

BAB IV

HASIL AKHIR DAN PEMBAHASAN

4.1 Pengujian Pembacaan Sensor MH-Z19

Diketahui bahwa sensor CO₂ MH-Z19 berjenis NDIR (*non-dispersive infra red*) yang menggunakan *infra red*, *optical filter* dan *detector* sebagai media pengukuran CO₂. Dalam pembacaan data hasil pengukuran sensor MH-Z19 dibutuhkan *array* berisi 9 *byte* data guna sebagai *command* kepada sensor dan data dari pembacaan. Berdasarkan *datasheet*, untuk memperoleh data pengukuran, antarmuka I²C atau mikrokontroler harus mengirim sebuah *command* berupa *array* berisi 9 *byte* yang berisi data sebagai berikut.

Command [9] = {0xFF,0x01,0x86,0x00,0x00,0x00,0x00,0x00,0x79}

Sensor MH-Z19 akan menerima *command* di atas melalui komunikasi serial atau UART. Sensor lalu turut melakukan pengukuran CO₂ dan mengirim hasil pengukuran tersebut dalam sebuah *array* yang berisi 9 *byte* data. Namun, tidak semua data dalam *array* merupakan data pengukuran. Hanya 2 *byte* data yang merupakan data pengukuran CO₂ yaitu *byte* ke-2 sebagai *high level concentration* dan *byte* ke-3 sebagai *low level concentration*. Untuk memperoleh data pengukuran utuh, kedua *byte* ini harus dikombinasikan menjadi sebuah data 2 *byte*. Berikut adalah rumus dan penjelasan kombinasi 2 *byte* data.

$$\text{CO}_2 = (\text{byte ke-2} \times 256) + \text{byte ke-3}$$

Sebagai contoh perhitungan, misalnya *byte* ke-2 adalah 3 dan *byte* ke-3 adalah 135. Maka nilai pengukuran CO₂ berdasarkan kedua data tersebut adalah sebagai berikut

$$\text{CO}_2 = (3 \times 256) + 135$$

$$\text{CO}_2 = 904$$

Perkalian antara 256 dan *byte* ke-2 berfungsi untuk memindahkan posisi *byte* ke-2 tepat disebelah kanan *byte* ke-3. Sehingga jika digabungkan antara kedua *byte* tersebut akan dihasilkan data utuh bernilai 2 *byte*. Jika pada perhitungan dilakukan dalam format *byte* atau biner, dan variabel *array* adalah “data”, maka rumus dalam program sebagai berikut.

$$\text{CO}_2 = \text{data}[2] \ll 8 \mid \text{data}[3]$$

Pada rumus di atas “ \ll ” berfungsi sebagai operator *left shift* dimana biner akan digeser atau pindah ke kiri. Sementara itu, 8 adalah jumlah pergeseran biner yaitu 8 kali pergeseran. Tanda “|” adalah sebuah operator “or” yang berfungsi saat penggabungan 2 atau lebih *operand* dengan menggunakan sifat dari logika “or”. Berikut adalah contoh perhitungan dengan nilai yang sama seperti pada contoh sebelumnya yaitu $\text{data}[2] = 3$ atau dalam biner “0000 0011” dan $\text{data}[3] = 135$ atau dalam biner “1000 0111”.

$$\text{CO}_2 = \text{data}[2] \ll 8 \mid \text{data}[3]$$

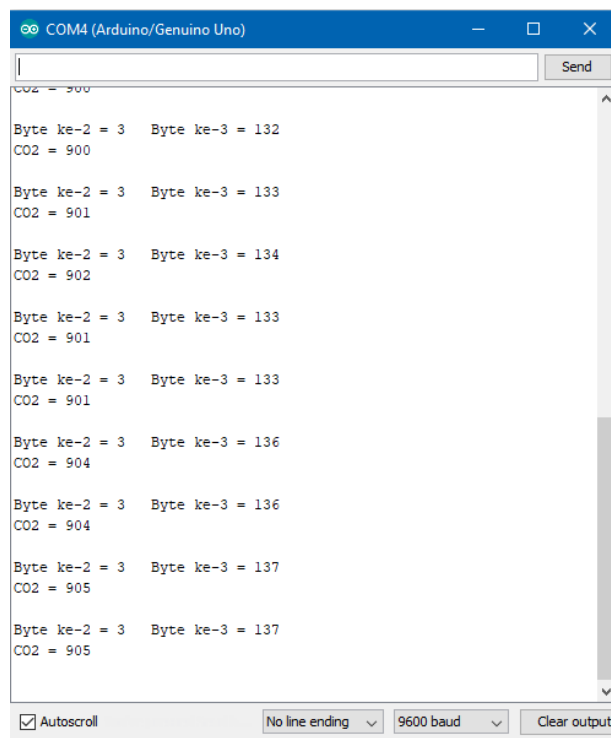
$$\text{CO}_2 = 0000\ 0011 \ll 8 \mid 1000\ 0111$$

$$\text{CO}_2 = 0000\ 0011\ 0000\ 0000 \mid 1000\ 0111$$

$$\text{CO}_2 = 0000\ 0011\ 1000\ 0111$$

$$\text{CO}_2 = 903$$

Kedua perhitungan sebelumnya akan memiliki hasil yang sama dan yang membedakannya hanyalah format data yang digunakan. Dengan kedua contoh perhitungan di atas, nilai pengukuran CO₂ dapat diperoleh secara utuh tanpa ada perhitungan lainnya. Sesuai dengan program CO₂ MH-Z19 pada gambar 2.24 pada perancangan perangkat lunak, diperoleh data pengukuran yang ditampilkan pada serial monitor arduino IDE. Berikut adalah gambar dari hasil pengukuran sensor MH-Z19 pada gambar 4.1 dibawah ini.



Gambar 4.1 Hasil Program CO₂ Sensor MH-Z19

Pada arduino terdapat 2 metode untuk melakukan komunikasi UART atau serial dengan sensor atau *device* lainnya, yaitu dengan menggunakan *software serial* atau *hardware serial*. Pada *hardware serial*, komunikasi UART harus melalui pin khusus UART yaitu Rx dan Tx. Namun, pada *software serial* bisa menjadikan pin lain sebagai Rx dan Tx dengan tambahan *library*

SoftwareSerial pada program arduino. Setelah dilakukan percobaan diantara kedua jenis metode tersebut terdapat perbedaan kecil dari hasil pembacaan pada serial monitor di arduino IDE. Pada *software serial* tidak terjadi masalah, namun pada *hardware serial* terdapat penambahan karakter tertentu pada awal data. Tetapi, data pengukuran tidak mengalami kesalahan sama sekali. Berikut adalah program CO₂ sensor MH-Z19 yang menggunakan *software serial* dan hasil serial monitor pada hardware serial.

The image shows two windows from the Arduino IDE. Window (a) displays the source code for a program named 'B_MH-Z19_UART_SOFTWARESERIAL'. The code includes the `<SoftwareSerial.h>` header, defines a `SoftwareSerial` object named `mySerial` on pins 2 and 3, and uses `mySerial` for all serial communication. Key lines are highlighted with red boxes: `#include <SoftwareSerial.h>`, `SoftwareSerial mySerial (2,3); //RX DAN TX`, `mySerial.begin(9600);`, and the `mySerial` write and read functions in the `loop()`. Window (b) shows the serial monitor output for the same program. The output consists of multiple lines of data, each starting with a header: `<Byte ke-2 = 2 Byte ke-3 =` followed by a CO₂ value. The values range from 195 to 224. The output is clean and matches the data from the `software serial` implementation.

(a)

(b)

Gambar 4.2 Program *Software Serial* (a) dan Hasil *Hardware Serial*

Berdasarkan gambar 4.2 poin (a) di atas, *software serial* menggunakan *library* tambahan yaitu `<SoftwareSerial.h>` dan harus menginisialisasikan nama serial beserta pin yang akan digunakan sebagai pengganti Rx dan Tx asli. Aktifitas dan fungsi yang menggunakan serial harus mengikuti dari nama

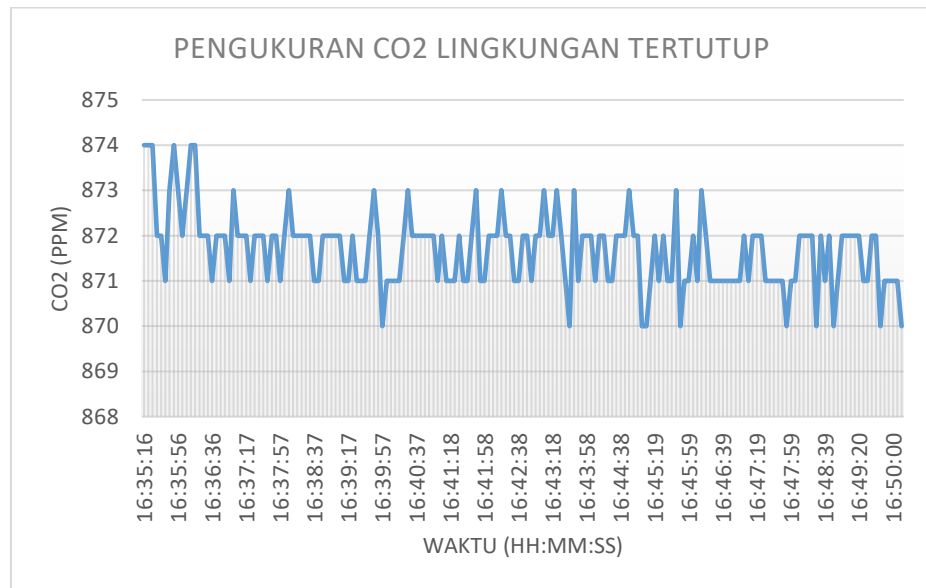
software serial yang telah diinisialisasikan tersebut. Pada gambar 4.2 poin (b) di atas dapat diperhatikan pada kotak merahnya bahwa pada tiap awal data hasil dari `serial.print()`, terdapat tambahan karakter tertentu yang muncul tanpa perintah. Karakter ini selalu muncul di awal kalimat dan konstan. Karakter tersebut adalah “?” yang tidak diketahui penyebab kemunculannya. Namun, hal ini tidak mengganggu dari hasil pengukuran CO₂ sensor MH-Z19.

4.2 Pengujian dan Analisis Pengukuran Sensor MH-Z19

4.2.1 Konsistensi data

Tujuan pengujian ini adalah untuk mengetahui konsistensi pengukuran kadar CO₂. Oleh karena itu, dalam pengujian ini sensor diletakkan pada wadah tertutup sehingga tidak terjadi pertukaran atau sirkulasi udara. Hal ini untuk mencegah adanya perbedaan kadar CO₂ yang bisa dipengaruhi oleh berbagai faktor seperti pengaruh sumber CO₂ di sekitar lokasi pengujian, angin dan lain sebagainya.

Pengujian dilakukan selama 15 menit (180 kali pengukuran dan interval pengukuran 5 detik), dengan lokasi pengujian diluar ruangan dan pengukuran awal kadar CO₂ sebelum wadah ditutup rapat adalah sebesar 874 PPM. Pengukuran CO₂ oleh sensor MH-Z19 ini dikombinasikan dengan aplikasi CO₂ *reader* yang dibuat menggunakan *processing IDE* sebagai monitoring sensor dan penyimpanan data pengukuran. Grafik hasil pengujian pengukuran CO₂ pada lingkungan tertutup ini ditunjukkan pada gambar 4.3 pada halaman berikutnya.



Gambar 4.3 Grafik Pengukuran CO₂ Lingkungan Tertutup

Berdasarkan grafik pada gambar 4.3 di atas, ditunjukkan bahwa pengukuran kadar CO₂ memiliki perbedaan nilai hingga 2 PPM di tiap pengukurannya tiap 5 detik. Tidak menutup kemungkinan kadar CO₂ yang diukur akan mengalami penurunan jika terus dilakukan pengukuran dalam waktu lama. Hal ini ditunjukkan dari kadar CO₂ yang pada awalnya 875 PPM dan yang terus turun mengalami fluktuasi dengan titik terendah 870 PPM. Dan di waktu akhir pengujian, data kadar CO₂ yang terukur semakin banyak bernilai 870 PPM.

4.2.2 Pengujian Pengukuran Sensor di berbagai Jenis Lingkungan

Tujuan dari pengujian ini adalah untuk mengetahui kinerja sensor saat mengukur kadar CO₂ pada beberapa jenis lingkungan berbeda baik lokasi maupun jenis sumber CO₂. Terdapat beberapa jenis lingkungan yang diujikan yaitu lingkungan udara bebas pada siang hari dan malam

hari, asap knalpot kendaraan motor, dan ruangan kamar tertutup berisi manusia.

Pada lingkungan udara bebas dilakukan 2 kali pengujian untuk mengetahui perbedaan kadar CO₂ pada siang dan malam hari. Hal ini dikarenakan terdapat perbedaan kegiatan pada siang dan malam hari. Pengujian asap knalpot dilakukan dengan menggunakan motor beat 110cc sebagai media pengujian. Sedangkan pengujian ruangan kamar tertutup dilakukan pada sebuah kamar dengan ventilasi kecil dan dimensi kamar ± 3,7m x 2m x 3m serta sumber CO₂ adalah seorang manusia. Seperti pada pengujian sebelumnya, durasi pengukuran kadar CO₂ adalah 15 menit dengan interval pengukuran sebanyak 5 detik. Data pengukuran disimpan dengan bantuan aplikasi *processing*. Data hasil pengujian tiap lingkungan ditampilkan pada tabel 4.1 di bawah ini dengan menghitung nilai rata-rata dari data yang diperoleh.

Tabel 4.1 Hasil Pengujian MH-Z19 Berbagai Lingkungan

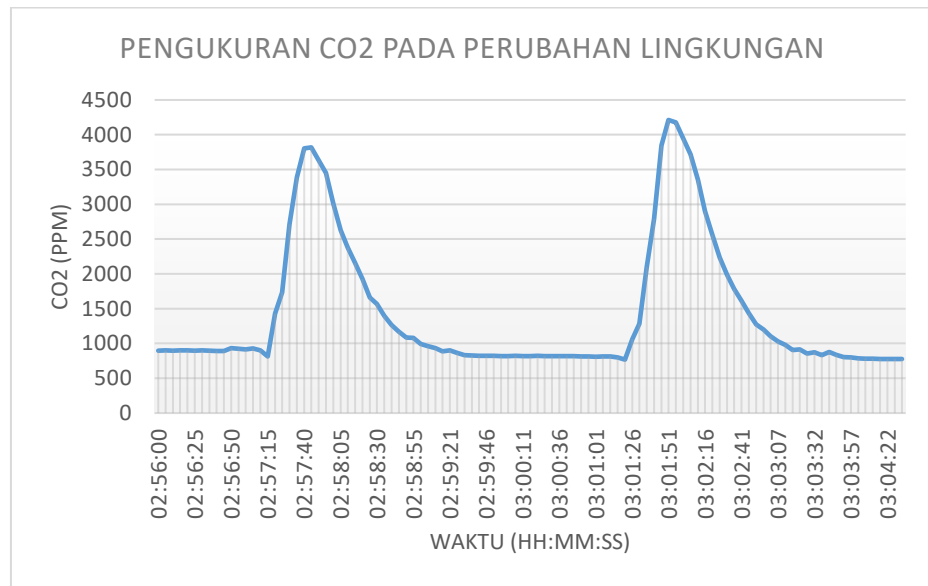
NO	LINGKUNGAN	CO ₂ (PPM)
1.	Udara Bebas (Siang)	564,6
2.	Udara Bebas (Malam)	906,4
3.	Asap Kendaraan Motor	1309,9
4.	Ruang Kamar Tertutup	970,5

Berdasarkan data dari tabel di atas, diketahui bahwa kadar CO₂ pada siang dan malam hari memiliki perbedaan yang signifikan. Hal ini

dikarenakan adanya perubahan kegiatan tumbuhan yang pada siang hari berfotosintesis dan malam hari lebih berfokus pada respirasi. Lingkungan dengan kadar CO₂ tertinggi adalah asap kendaraan bermotor yang diketahui menjadi salah satu sumber CO₂ dan CO terbesar. Ruang kamar tertutup dengan manusia didalamnya juga memiliki kadar CO₂ yang terbilang tinggi, namun masih di batas normal jika tidak melebihi 1000 PPM. Hal ini dikarenakan sirkulasi yang kurang baik dan tidak adanya penyerap CO₂ seperti pohon. Data tabel 4.1 tidak bisa dibilang valid, karena tidak adanya perbandingan dengan sensor CO₂ yang lebih presisi dan digunakan oleh lembaga pengukuran cuaca dan sebagainya. Namun, bisa diketahui bahwa sensor bekerja dengan cukup baik.

4.2.3 Pengujian Perubahan Lingkungan

Pada pengujian ini sensor MH-Z19 akan dihadapkan pada kondisi disaat normal dan disaat diberi gas CO₂ hasil respirasi atau pernafasan secara langsung. Tujuannya untuk mengetahui kinerja sensor saat terjadi perubahan lingkungan secara mendadak. Selain itu juga mengetahui kecepatan sensor dalam mengukur kadar CO₂ setelah diberikan gas CO₂. Pengujian dilakukan dengan melakukan respirasi tepat di dekat sensor sebanyak 2 kali. Respirasi dilakukan selama ± 20 detik lalu menjauhkan sensor. grafik hasil pengukuran CO₂ terhadap perubahan yang mendadak dengan sumber CO₂ berupa hasil respirasi manusia yang ditunjukkan pada gambar 4.4 pada halaman selanjutnya.

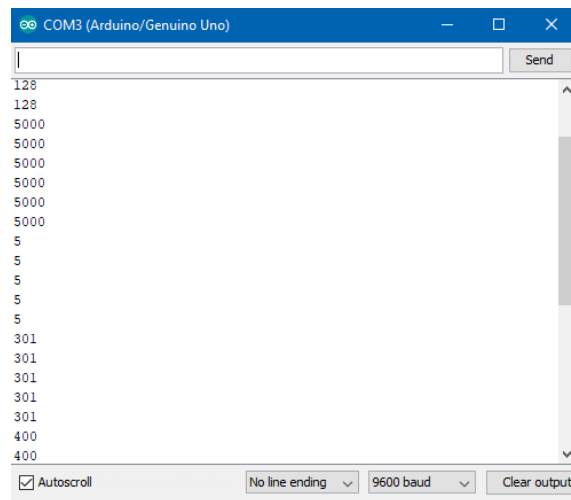


Gambar 4.4 Grafik Pengukuran CO₂ Pada Perubahan Lingkungan

Berdasarkan grafik pada gambar 4.4 di atas dapat diamati bahwa kadar CO₂ hasil respirasi manusia bisa mencapai 4000 PPM saat sensor di dekatkan dengan mulut. Kemampuan sensor MH-Z19 untuk mengukur perubahan mendadak kadar CO₂ tidak bisa dibilang cepat. Hal ini dikarenakan sensor membutuhkan beberapa detik untuk mendapatkan hasil pengukuran yang tepat. Hal ini dibuktikan dari grafik kenaikan kadar CO₂ yang meningkat secara bertahap dan tidak langsung menuju nilai sebenarnya. Begitu pula saat sensor sudah tidak lagi diberi CO₂. Sensor membutuhkan lebih dari 40 detik untuk memperoleh kembali hasil pengukuran saat kondisi normal sebelum diberi CO₂. Kemungkinan hal ini disebabkan karena sirkulasi udara pada tabung pengukuran sensor sehingga butuh waktu untuk udara didalam tabung sensor sama dengan udara di lingkungan luar sensor.

4.3 Analisis Kondisi Start Sensor MH-Z19

Setelah melakukan beberapa percobaan dan pengujian terhadap kinerja sensor CO₂ MH-Z19, diketahui bahwa sensor membutuhkan beberapa waktu saat sensor baru dihidupkan untuk melakukan pengukuran CO₂ yang ditunjukkan pada gambar di bawah ini.



Gambar 4.5 Hasil Pengukuran Kondisi Awal Sensor MH-Z19

Berdasarkan gambar 4.5 di atas, hasil pengukuran CO₂ yang diterima menunjukkan hasil yang tidak sesuai. Namun, ketidaksesuaian data ini hanya berlangsung beberapa detik saat sensor baru dihidupkan. Oleh karena itu, dilakukan analisis waktu optimal sensor MH-Z19 untuk dapat melakukan pengukuran kadar CO₂ secara tepat. Pengujian dilakukan dengan memperhatikan data pengukuran saat sensor baru dihidupkan dan dilakukan selama 10 kali percobaan. Selain itu, juga diperhatikan nilai-nilai yang diberikan sensor sebelum sensor menghasilkan data pengukuran yang sesuai. Hasil pengujian ini ditunjukkan pada tabel 4.2 di halaman selanjutnya.

Tabel 4.2 Hasil Pengamatan Kondisi Awal Sensor

NO	PENGUJIAN	DATA AWAL	DATA AKHIR	JUMLAH DATA ERROR	WAKTU (DETIK)
1	KE-1	128	900	90	90
2	KE-2	128	900	92	92
3	KE-3	128	900	91	91
4	KE-4	128	900	92	92
5	KE-5	128	900	91	91
6	KE-6	128	900	90	90
7	KE-7	128	900	91	91
8	KE-8	128	900	91	91
9	KE-9	128	900	90	90
10	KE-10	128	900	90	90
RATA-RATA				90,8	90,8

Dari tabel 4.2 yang menunjukkan hasil pengamatan kondisi awal sensor dapat diketahui bahwa sensor membutuhkan ± 90 detik sampai sensor mampu memberikan data hasil pengukuran CO₂ yang tepat. Kondisi ini biasa disebut sebagai kondisi *preheating* sensor. Tidak hanya sensor MH-Z19, beberapa sensor juga memiliki kondisi *preheating* layaknya suatu mesin yang butuh pemanasan hingga mencapai kondisi stabil.

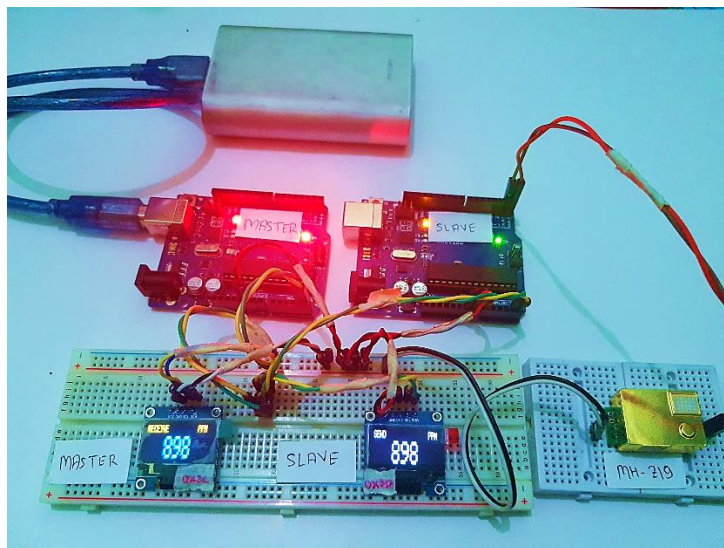
Pada tahap *preheating* ini, data awal yang dihasilkan selalu dimulai dari 128, 5000, 5, 301, 400 dan diakhiri dengan 900. Setelah itu sensor akan mampu mengukur kadar CO₂ disekitar dengan lebih tepat.

Refresh-rate sensor MH-Z19 adalah 5 detik. Jadi, sensor akan mengukur kadar CO₂ dalam interval 5 detik. Dari data mentah pada analisis kondisi awal sensor, data sensor pada nilai 128,5000,5,301 dan 900 hanya muncul selama 5 detik atau dalam sekali *refresh* sensor. Sementara itu nilai 400 terus muncul dalam jangka waktu ± 65 detik.

4.4 Pengujian Transmisi Data Antar Arduino Via I²C

Pengujian ini dilakukan untuk mengetahui tingkat keberhasilan pengiriman data melalui komunikasi I²C. Dibutuhkan 2 arduino untuk melakukan pengujian ini. Salah satu arduino akan bertindak sebagai *slave* yang bertugas mengirim data pengukuran CO₂ dari sensor MH-Z19 dan arduino lain sebagai *master* yang bertugas sebagai penerima data dari *slave*.

Keberhasilan pengujian ini akan berdampak baik pada antarmuka I²C sensor MH-Z19, dikarenakan antarmuka I²C ini akan berperan seperti arduino *slave*. Untuk mempermudah pengamatan, tiap arduino dikoneksikan dengan *display* oled untuk membandingkan hasil dari data pengukuran yang dikirim dan diterima. Semua komunikasi pada pengujian menggunakan komunikasi I²C dan khusus pada sensor menggunakan komunikasi UART. Pengujian ini ditunjukkan pada gambar 4.6 di bawah ini



Gambar 4.6 Pengujian Transmisi Data CO₂ Antar Arduino

Berdasarkan hasil pengamatan yang ditunjukkan pada gambar 4.6 sebelumnya, dapat disimpulkan bahwa sangat memungkinkan untuk mentransmisikan data hasil pengukuran sensor MH-Z19 yang berasal dari komunikasi UART menggunakan antarmuka I²C. Hal ini dibuktikan dengan kesamaan data yang diterima oleh sisi *master* dengan data yang dikirim oleh pihak *slave*. Untuk membuktikannya dilakukan lagi pengujian dengan menyimpan data baik dari *slave* maupun *master*. Berikut ini adalah tabel data berdasarkan waktu transmisi dengan interval transmisi selama 1 detik dan pengamatan pada sampel 20 detik durasi transmisi.

Tabel 4.3 Hasil Pengamatan Transmisi Data

TANGGAL	MASTER		SLAVE	
	WAKTU	DATA	WAKTU	DATA
10/05/2018	00:52:00	774	00:52:00	774
10/05/2018	00:52:01	774	00:52:01	774
10/05/2018	00:52:02	774	00:52:02	774
10/05/2018	00:52:03	774	00:52:03	774
10/05/2018	00:52:04	774	00:52:04	774
10/05/2018	00:52:05	775	00:52:05	775
10/05/2018	00:52:07	775	00:52:07	775
10/05/2018	00:52:08	775	00:52:08	775
10/05/2018	00:52:09	775	00:52:09	775
10/05/2018	00:52:10	775	00:52:10	775
10/05/2018	00:52:11	775	00:52:11	775
10/05/2018	00:52:12	775	00:52:12	775
10/05/2018	00:52:13	775	00:52:13	775
10/05/2018	00:52:14	775	00:52:14	775
10/05/2018	00:52:15	775	00:52:15	775
10/05/2018	00:52:16	775	00:52:16	775
10/05/2018	00:52:17	775	00:52:17	775
10/05/2018	00:52:18	775	00:52:18	775
10/05/2018	00:52:19	775	00:52:19	775
10/05/2018	00:52:20	776	00:52:20	776

Berdasarkan tabel 4.3 pada halaman sebelumnya, ditunjukkan bahwa data baik yang dikirim oleh *slave* maupun yang diterima oleh *master* bernilai sama. Selain itu juga tidak ada *delay* dalam transmisi data. Data langsung dikirim menuju *master* dalam waktu yang sama saat pengiriman. Meskipun ada keterlambatan, kemungkinan hanya dalam hitungan *millisecond* dan tidak terlalu berpengaruh pada saat transmisi.

Bersamaan dengan pengujian transmisi data antar arduino, terdapat beberapa data yang mengalami kerusakan atau *error*. Pada arduino *slave* yang bertugas mengirim data pengukuran CO₂ terdapat beberapa data yang melebihi nilai maksimal yang bisa diukur oleh sensor. Beberapa data berukuran lebih besar dari 2000 PPM. Berikut ini adalah tabel persentase *error* pada arduino *slave*.

Tabel 4.4 Persentase *Error* Data Arduino *Slave*

NO	WAKTU	JUMLAH DATA	JUMLAH DATA ERROR	PERSENTASE ERROR (%)
1	1 JAM KE-1	6804	20	0,293945
2	1 JAM KE-2	6804	25	0,367431
3	1 JAM KE-3	6804	30	0,440917
4	1 JAM KE-4	6804	22	0,323339
5	1 JAM KE-5	6804	22	0,323339
TOTAL		34020	119	1,74897119
RATA-RATA			24,2	0,3497942

Pengujian pada tabel 4.4 di atas dilakukan selama 5 jam dengan interval transmisi data 0,5 detik. Data yang *error* bernilai > 2000 PPM, karena

maksimal pengukuran sensor adalah 2000 PPM. Dari pengujian ini, diketahui bahwa kerusakan data pada arduino *slave* mencapai $\pm 0,3497942\%$.

Selain pada arduino *slave* sebagai *transmitter*, arduino *master* sebagai *receiver* yang menerima data pengukuran sensor MH-Z19 juga mendapatkan beberapa data yang *error*. Sedikit berbeda dengan data *error* pada arduino *slave*, data *error* pada arduino *master* terdiri dari beberapa jenis yaitu data *error* yang melebihi 2000 PPM, data *error* yang bernilai minus (-) dan data *error* yang bernilai konstan yaitu 255. Nilai 255 diketahui adalah nilai dari 1 *byte* data atau 1111 1111. Untuk mengetahui persentase *error* pada arduino *master*, dilakukan pengujian yang sama dengan pengujian pada arduino *slave* sebelumnya. Berikut ini adalah persentase *error data* pada arduino *master* setelah dilakukan pengujian selama 5 jam.

Tabel 4.5 Persentase *Error Data* Arduino *Master*

NO	WAKTU	JUMLAH DATA	JUMLAH DATA ERROR	PERSENTASE ERROR (%)
1	1 JAM KE-1	6804	106	1,557907
2	1 JAM KE-2	6804	99	1,455026
3	1 JAM KE-3	6804	109	1,601999
4	1 JAM KE-4	6804	89	1,308054
5	1 JAM KE-5	6804	101	1,484421
TOTAL		34020	504	7,407407
RATA-RATA			100,8	1,481481

Berdasarkan data hasil pengamatan pada tabel 4.5 di atas dapat disimpulkan bahwa pada sisi arduino *master* atau data yang diterima mengalami kerusakan sebesar $\pm 1,481481\%$. Seperti yang disinggung

sebelumnya, terdapat beberapa jenis data *error* yang diterima. Jumlah data *error* terbesar adalah data yang bernilai 255. Berikut adalah tabel keterangan data *error* berdasarkan data yang sama pada tabel 4.5.

Tabel 4.6 *Error Data Arduino Master*

NO	WAKTU	ERROR >2000	ERROR <0	ERROR 255	TOTAL
1	1 JAM KE-1	6	14	86	106
2	1 JAM KE-2	6	19	74	99
3	1 JAM KE-3	12	18	79	109
4	1 JAM KE-4	10	12	67	89
5	1 JAM KE-5	10	12	79	101

Jenis program transmisi data yang digunakan pada pengujian ini adalah program *master request to slave*. Kode program ini lebih sederhana dan *responsive* dibandingkan kode program transmisi yang lain. *Master* tidak perlu mengirim *command* tertentu setiap transmisi seperti pada program *master send command to slave*. Program *master send command to slave* lumayan rumit dikarenakan *master* harus mengirim dahulu sebuah *command* berukuran 1 *byte* kepada *slave* untuk *slave* mengirim ke *master*. Dari sisi memori, program ini lebih besar dibandingkan program *master request to slave*. Dan program *master receive from slave* juga kurang baik. Pemberian *address* dilakukan pada sisi *master*, sehingga jika digunakan pada antarmuka I²C akan mempersulit *master* untuk mendeteksi ketersediaan antarmuka I²C. Berikut ini adalah kode program yang digunakan pada pengujian transmisi data antar arduino yang ditunjukkan pada halaman berikutnya.


```

slave_sender

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);

byte request[9] = {0xFF,0x01,0x86,0x00,0x00,0x00,0x00,0x00,0x79};
unsigned char response[9];

unsigned int co2;

void setup() {
  Serial.begin(9600);
  Wire.begin(0x47);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3D);
  Wire.onRequest(requestEvent);
}

void displayco2() {

  Serial.write(request, 9);
  memset(response,0,9);
  Serial.readBytes(response,9);

  unsigned int HLconcentration = (unsigned int) response[2];
  unsigned int LLconcentration = (unsigned int) response[3];
  co2 = (256*HLconcentration) + LLconcentration;

  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  display.setTextSize(1);
  display.print("SEND          PPM");
  display.setCursor(0,10);
  display.setTextSize(3);
  if (co2 < 1000) {
    display.print(" ");
  }
  display.print(" ");
  display.print(co2);
}

void requestEvent() {
  byte buffer[2];
  buffer[0] = co2 >> 8;
  buffer[1] = co2 & 255;
  Wire.write(buffer, 2);
}

void loop() {
  displayco2();
  display.display();
  delay(1000);
}

```

Done Saving.

Gambar 4.7 Program *Slave* Pengujian Transmisi Antar Arduino

```

master_receive | Arduino 1.8.5
File Edit Sketch Tools Help

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);

int y;

void setup() {
  Wire.begin();
  Serial.begin(9600);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
}

void loop() {

  int check = Wire.requestFrom(0x47, (byte)2);
  if(check == 2)
  {
    y = Wire.read() << 8 | Wire.read();
  }

  Serial.println(y);
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  display.setTextSize(1);
  display.print("RECEIVE          PPM");
  display.setCursor(0,10);
  display.setTextSize(3);
  if (y < 1000) {
    display.print(" ");
  }
  display.print(" ");
  display.print(y);
  display.display();
  delay(500);
}
Done Saving.

```

Gambar 4.8 Program *Master* Pengujian Transmisi Antar Arduino

Pada pembahasan 4.1 mengenai pengujian pembacaan sensor MH-Z19 terdapat kode program kombinasi 2 data berukuran masing-masing 1 *byte* menjadi sebuah data utuh berukuran 2 *byte*. Kode program

ini digunakan pada bagian *master*, dikarenakan *master* menerima data dari *slave* berupa 2 data berukuran 1 *byte*.

Pada bagian *slave*, data pengukuran CO₂ yang telah dikombinasikan menjadi sebuah data 2 *byte* dipecah kembali menjadi 2 buah data 1 *byte* dan disimpan dalam sebuah *array*. Hal ini dikarenakan komunikasi I²C hanya bisa mengirim per *byte* dan bertahap, tidak bisa langsung mengirim sebuah data berukuran 2 *byte* secara langsung. Oleh karena terdapat cara untuk memecah data 2 *byte* tersebut. Kodenya adalah sebagai berikut.

```
byte buffer[2];
buffer[0] = co2 >> 8;
buffer[1] = co2 & 255;
Wire.write(buffer, 2);
```

Gambar 4.9 Kode Pemecahan Data

Data yang akan dikirim berupa *array* yang berisi 2 *byte* data. Seperti pada kombinasi 2 data biner pada pembahasan 4.1. Pemecahan data juga memiliki sistem seperti itu yaitu pergeseran *byte* dan penggabungan 2 operand atau *byte*. Pada gambar 4.9 di atas, data *co2* akan di jadikan 2 data yang berbeda. Data pertama atau pada gambar yaitu *buffer[0]* akan mengalami *rightshift* sebanyak 8 kali. Dan data kedua yaitu *buffer[1]* akan mengalami penggabungan dengan *byte* 255 secara logik “and”. Contoh perhitungan jika nilai CO₂ adalah 447 atau 0000 0001 1011 1111.

```
buffer [0] = CO2 >> 8
```

```
buffer [0] = 0000 0001 1011 1111 >> 8
```

```
buffer [0] = 0000 0001
```

```
buffer [0] = 1
```

Perhitungan pada data kedua.

```
buffer [1] = CO2 & 255
```

```
buffer [1] = 0000 0001 1011 1111 & 0000 0000 1111 1111
```

```
buffer [1] = 1011 1111
```

```
buffer [1] = 191
```

Kedua data di atas akan dikirim melalui komunikasi I²C dan *master* menerima serta mengkombinasikan kedua data ini menjadi sebuah data 2 *byte*. Namun, terdapat sebuah metode lagi yaitu dengan mengirim langsung 2 buah data *byte* mentah hasil pengukuran sensor MH-Z19 yang belum dikombinasikan yaitu data berikut.

```
unsigned int HLconcentration = (unsigned int) response[2];
unsigned int LLconcentration = (unsigned int) response[3];
```

Gambar 4.10 Data Mentah MH-Z19

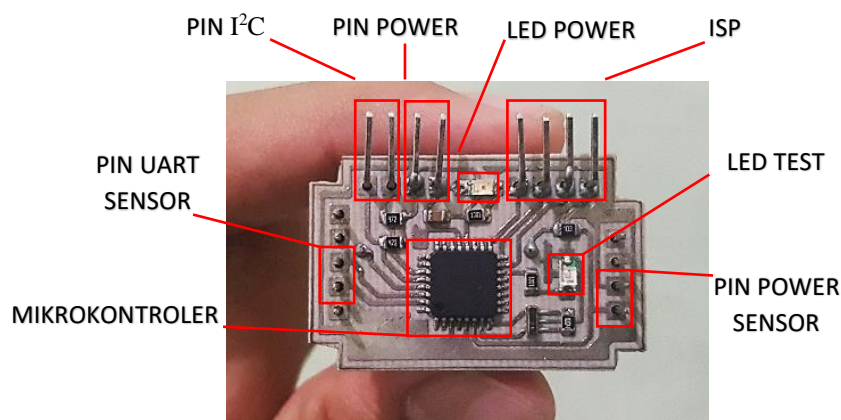
Kedua data ini bisa langsung dikirimkan menuju *master* dengan memasukkannya pada sebuah *array*. Hal ini sama seperti kode pada gambar 4.9 dan hasilnya akan menjadi seperti kode pada gambar 4.11.

```
byte buffer[2];
buffer[0] = HLconcentration;
buffer[1] = LLconcentration;
Wire.write(buffer, 2);
```

Gambar 4.11 Pengiriman Data Mentah MH-Z19

4.5 Pengujian Antarmuka I²C dengan Arduino

Antarmuka I²C adalah tujuan utama dari tugas akhir ini. Perangkat ini memiliki fungsi untuk mengubah *output* atau keluaran sensor yang awalnya menggunakan komunikasi UART menjadi komunikasi I²C. Sistem kerjanya adalah membaca data dari pengukuran CO₂ sensor MH-Z19 melalui komunikasi UART dan meneruskan data tersebut ke *device* atau *peripheral* lain yang membutuhkan melalui komunikasi I²C. Seperti modul, sensor dan alat lain yang memiliki antarmuka I²C, perangkat ini juga memiliki *address* tersendiri. *Address* ini bisa diatur pada program dan pada saat ini *address*-nya adalah 0x47. Sebagai pengontrolnya, antarmuka I²C ini menggunakan mikrokontroler atmega 8. Berikut adalah bentuk dari antarmuka I²C tersebut.



Gambar 4.12 Antarmuka I²C Sensor MH-Z19

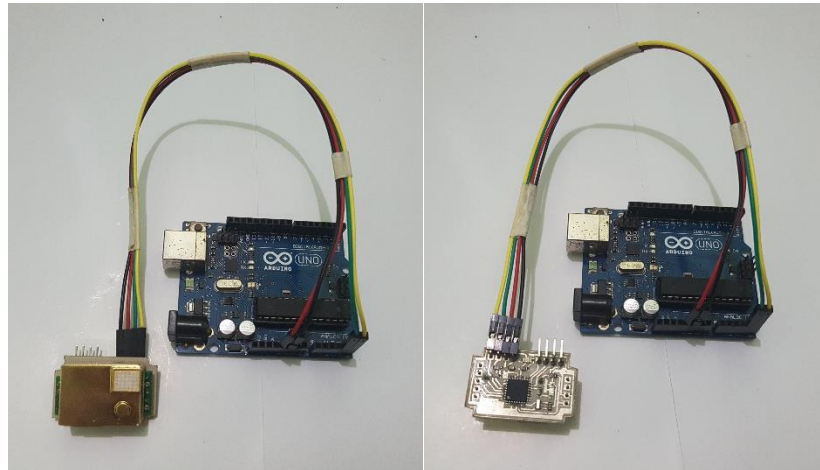
Berdasarkan gambar 4.12, berikut ini adalah tabel spesifikasi dari perangkat antarmuka I²C pada sensor CO₂ MH-Z19.

Tabel 4.7 Spesifikasi Antarmuka I²C

NO	Spesifikasi	Keterangan
1	Jenis Modul	Antarmuka I ² C
2	Target Sensor	Sensor CO ₂ MH-Z19
3	<i>Chip Microcontroller</i>	Atmega 8
4	Tegangan Operasi	5 volt
5	Memori <i>Flash</i>	8 Kb,0,5 Kb dipakai bootloader
6	SRAM	1 Kb
7	EEPROM	512 <i>bytes</i>
8	<i>Clock Speed</i>	16 Mhz
9	I/O	2 Pin Power 2 Pin I ² C 4 Pin ISP
10	<i>Output Data</i>	I ² C
11	Input Data	UART
12	Speed-rate I ² C	100 KHz
13	<i>Adress</i>	0x47
14	<i>Programming System</i>	Arduino
15	Indikator	LED Power LED Test
16	Reset	Software
17	Dimensi	23 mm x 35 mm

Dikarenakan menggunakan sistem arduino, speed-rate antarmuka I²C secara otomatis diatur pada kecepatan 100 KHz yang merupakan kecepatan

standar I²C. Hampir semua perangkat, sensor dan mikrokontroler dipasarkan menggunakan kecepatan standar dalam komunikasi I²C.



Gambar 4.13 Pengujian Antarmuka I²C dengan Arduino

Pada gambar 4.13 di atas adalah pengujian antarmuka I²C dengan arduino sebagai *master* dari antarmuka I²C. Hasil dari pengujian ini sama seperti hasil dari pengujian transmisi data antar arduino melalui I²C sebelumnya. Data dapat diakses oleh antarmuka I²C dan dikirim menuju arduino sehingga arduino juga dapat mengakses data hasil pengukuran MH-Z19.

Dari segi program yang membedakan hanyalah penggunaan *display* oled yang dihilangkan khususnya untuk antarmuka I²C. Hal ini dikarenakan program dengan *display* oleh memiliki ukuran memori *flash* yang cukup besar hingga mencapai 12 Kb. Sementara kapasitas memori *flash* yang dimiliki atmega 8 hanya sebesar 8 Kb dan sudah terisi 0,5 Kb oleh *bootloader* arduino. Program akan dimasukkan melalui port ISP (*in-circuit serial programming*). Program yang diupload pada antarmuka I²C ini menggunakan arduino sebagai

downloader atau *arduino as isp*. Selain itu, terdapat tambahan program yaitu pada *test* LED. LED ini dijadikan indikator transmisi, sehingga saat antarmuka I²C mengirim data maka LED ini akan berkedip. Berikut ini adalah program antarmuka I²C pada uji coba dengan arduino.

The image shows a screenshot of the Arduino IDE interface. The title bar reads 'CONVERTER_FIX | Arduino 1.8.5'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu bar is a toolbar with various icons. The main text area contains the following C++ code:

```
#include <Wire.h>

#define SLAVE_ADDRESS 0x47

byte request[9] = {0xFF,0x01,0x86,0x00,0x00,0x00,0x00,0x00,0x79};
unsigned char response[9];

unsigned int co2;
int LED =10;

void setup() {
  Wire.begin(SLAVE_ADDRESS);
  Serial.begin(9600);
  Wire.onRequest(requestEvent);
  pinMode(LED, OUTPUT);
}

void requestEvent() {
  byte buffer[2];
  buffer[0] = co2 >> 8;
  buffer[1] = co2 & 255;
  Wire.write(buffer, 2);
  digitalWrite(LED, HIGH);
  delay(100);
  digitalWrite(LED, LOW);
  delay(50);
}

int getco2() {
  Serial.write(request, 9);
  memset(response,0,9);
  Serial.readBytes(response,9);

  unsigned int HLconcentration = (unsigned int) response[2];
  unsigned int LLconcentration = (unsigned int) response[3];
  co2 = (256*HLconcentration) + LLconcentration;
}

void loop() {
  getco2();
  delay(1000);
}
```

Gambar 4.14 Program Antarmuka I²C

Berdasarkan pengujian transmisi antar arduino ditemukan beberapa data yang mengalami *error*. Oleh sebab itu, pada pengujian antarmuka I²C dilakukan pengamatan *error* data dan penyimpanan data yang diterima oleh arduino *master*. Pada antarmuka I²C tidak dilakukan pengamatan dan penyimpanan data karena pin komunikasi serial sudah terpakai, sehingga tidak bisa dilakukan penyimpanan data melalui aplikasi *processing* pada laptop. Berikut ini adalah tabel persentase *error* hasil pengamatan data *error* yang diterima oleh arduino.

Tabel 4.8 Persentase *Error* Data Antarmuka I²C oleh Arduino

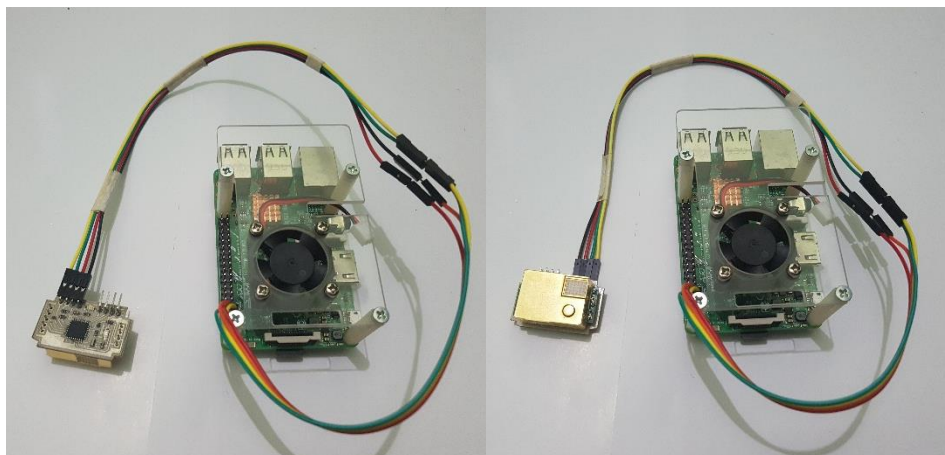
NO	WAKTU	JUMLAH DATA	JUMLAH DATA ERROR	PERSENTASE ERROR (%)	LOSSES DATA
1	1 JAM KE-1	3488	1	0,0287	112
2	1 JAM KE-2	3486	1	0,0287	114
3	1 JAM KE-3	3487	1	0,0287	113
4	1 JAM KE-4	3488	2	0,0573	112
5	1 JAM KE-5	3487	0	0	113
TOTAL		17435	5	0,1434	564
RATA-RATA			1	0,02868	112,8

Berdasarkan data dari tabel 4.7 di atas, diketahui bahwa *error* pada antarmuka I²C lebih kecil dibandingkan dengan percobaan transmisi antar arduino sebelumnya yang memiliki *error* mencapai $\pm 1,48$ %. Data antarmuka I²C ditransmisikan tiap 1 detik menuju arduino sehingga jika dihitung data ideal harusnya berjumlah 3600 data. Namun, data yang diterima rata-rata

berjumlah 3847 buah. Sehingga bisa diasumsikan interval transmisi bisa lebih dari 1 detik atau terjadi *delay* saat transmisi.

4.6 Pengujian Antarmuka I²C dengan Raspberry Pi

Pengujian antarmuka I²C adalah pengujian terakhir. Fungsi dari pengujian ini adalah untuk mengetahui tingkat fleksibilitas antarmuka I²C saat digunakan oleh *peripheral* atau *device* yang berbeda-beda. Untuk *wiring*-nya sama seperti saat pengujian arduino yaitu menggunakan pin I²C dan *power*. Selain itu, program pada antarmuka I²C juga tetap sama, yaitu dengan mengirim data *array* yang berisi 2 buah data dengan kapasitas 1 *byte* tiap data. Berikut ini adalah gambar pengujian antarmuka I²C menggunakan raspberry pi 3 sebagai *master* dan antarmuka I²C sebagai *slave* yang langsung terhubung dengan sensor MH-Z19.



Gambar 4.15 Pengujian Antarmuka I²C dengan Raspberry Pi

Program utama yang bertugas membaca data pin I²C dan mengkombinasikan data yang diterima pada raspberry pi adalah ditunjukkan pada gambar 4.16 di halaman selanjutnya.

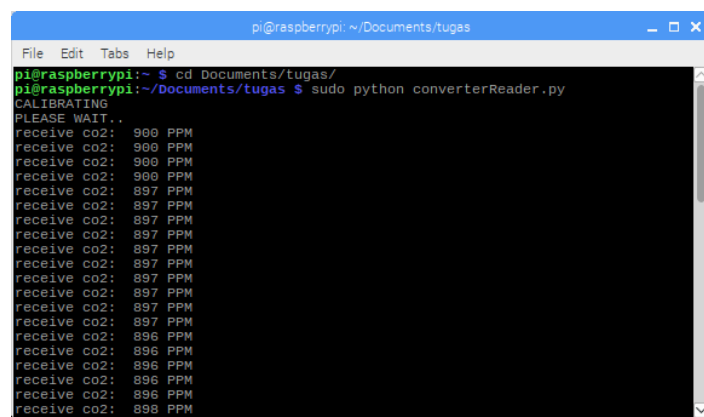
```

def readCo2():
    co2 = bus.read_i2c_block_data(address,0,2)
    y = co2[0]
    z = co2[1]
    co2 = (256*y)+z
    return co2

```

Gambar 4.16 Kode Utama Raspberry Pi

Hampir seluruh data yang diterima oleh raspberry tidak mengalami kerusakan atau kecacatan seperti yang ditunjukkan pada gambar 4.17. Namun tidak dipungkiri bahwa akan terjadi beberapa *error* baik pada data yang diterima seperti pada gambar 4.18 maupun pada *wiring* atau jalur datanya seperti pada gambar 4.19. Berikut ini adalah beberapa hasil pengujian antarmuka I²C dengan raspberry pi.

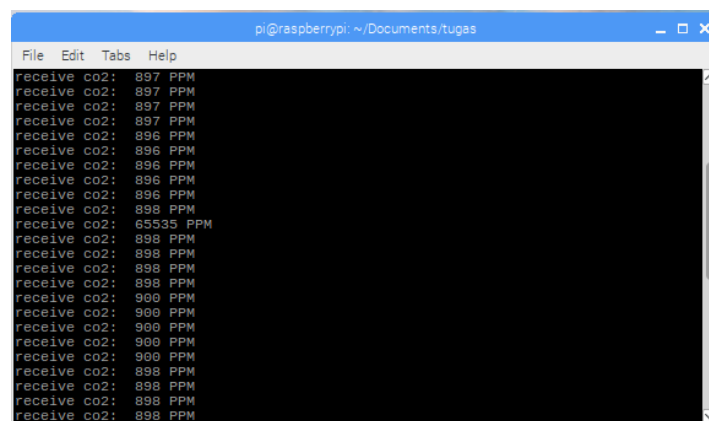


```

pi@raspberrypi: ~/Documents/tugas
File Edit Tabs Help
pi@raspberrypi:~$ cd Documents/tugas/
pi@raspberrypi:~/Documents/tugas$ sudo python converterReader.py
CALIBRATING
PLEASE WAIT..
receive co2: 900 PPM
receive co2: 900 PPM
receive co2: 900 PPM
receive co2: 900 PPM
receive co2: 897 PPM
receive co2: 897 PPM
receive co2: 897 PPM
receive co2: 897 PPM
receive co2: 897 PPM
receive co2: 897 PPM
receive co2: 897 PPM
receive co2: 897 PPM
receive co2: 897 PPM
receive co2: 896 PPM
receive co2: 896 PPM
receive co2: 896 PPM
receive co2: 896 PPM
receive co2: 896 PPM
receive co2: 898 PPM

```

Gambar 4.17 Hasil Penerimaan Data Raspberry (tanpa *error*)



```

pi@raspberrypi: ~/Documents/tugas
File Edit Tabs Help
receive co2: 897 PPM
receive co2: 897 PPM
receive co2: 897 PPM
receive co2: 897 PPM
receive co2: 896 PPM
receive co2: 896 PPM
receive co2: 896 PPM
receive co2: 896 PPM
receive co2: 896 PPM
receive co2: 898 PPM
receive co2: 65535 PPM
receive co2: 898 PPM
receive co2: 898 PPM
receive co2: 898 PPM
receive co2: 898 PPM
receive co2: 900 PPM
receive co2: 900 PPM
receive co2: 900 PPM
receive co2: 900 PPM
receive co2: 898 PPM
receive co2: 898 PPM
receive co2: 898 PPM

```

Gambar 4.18 Hasil Penerimaan Data Raspberry (data *error*)

Berdasarkan gambar 4.17 terdapat beberapa data yang terkirim mengalami kerusakan. Data tersebut selalu bernilai 65535 atau dalam biner berjumlah 1111 1111 1111 1111 yaitu 2 *byte* data penuh. Masih belum diketahui penyebab terjadinya kerusakan data tersebut.

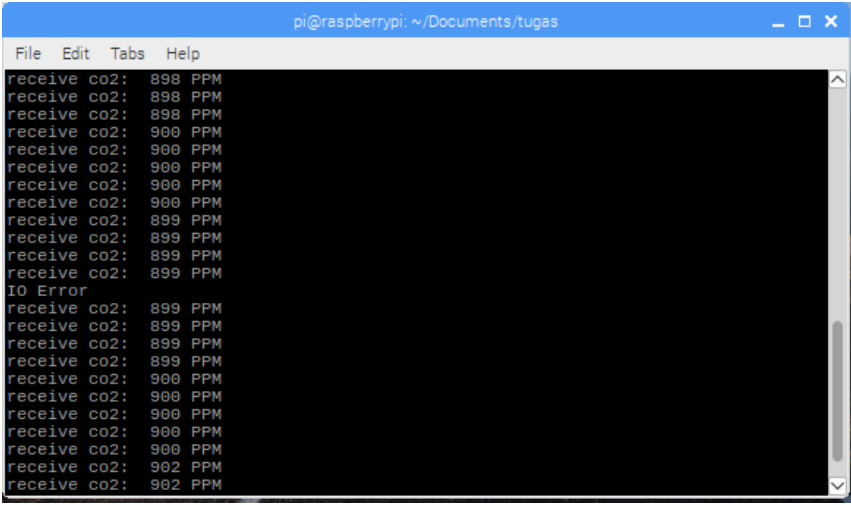
Diperlukan pengujian tambahan untuk mengetahui jumlah data *error* yang diterima oleh raspberry pi. Oleh karena itu, dilakukan pengujian dengan melakukan transmisi data antara antarmuka I²C yang meneruskan data pengukuran CO₂ dan raspberry pi sebagai penerima data pengukuran. Pada raspberry pi ditambahkan program untuk penyimpanan data pengukuran dalam format .csv sebagai *data logger*. Sistem pengujian adalah dengan melakukan pengukuran CO₂ sekaligus transmisi data selama 5 jam berturut-turut dengan interval transmisi data selama 1 detik. Setiap 1 jam akan dihitung presentase data *error* yang diterima. Berikut ini adalah hasil perhitungan persentase data *error* yang ditunjukkan pada tabel 4.9 di bawah ini.

Tabel 4.9 Persentase *Error* Data Antarmuka I²C

NO	WAKTU	JUMLAH DATA	JUMLAH DATA ERROR	PERSENTASE ERROR (%)
1	1 JAM KE-1	3590	0	0
2	1 JAM KE-2	3590	0	0
3	1 JAM KE-3	3591	0	0
4	1 JAM KE-4	3589	0	0
5	1 JAM KE-5	3591	0	0
TOTAL		17951	0	0
RATA-RATA			0	0

Berdasarkan hasil perhitungan persentase *error* data pada tabel 4.9 sebelumnya, diketahui bahwa tiap 1 jam transmisi data antar antarmuka I²C dan raspberry pi tidak terjadi kerusakan data sama sekali. Dari 3590 data yang diterima semuanya tidak mengalami kerusakan. Berdasarkan data tersebut, dapat diperhitungkan bahwa tingkat keberhasilan transmisi antara antarmuka I²C dan raspberry pi mencapai 100%. Oleh karena itu, pembuatan antarmuka I²C pada sensor MH-Z19 ini dinyatakan berhasil, dikarenakan tidak terdapat *error* sama sekali.

Selain *error* pada data yang diterima, terdapat *error* pada bagian jalur I²C yang digunakan sebagai media transmisi data. Terjadinya *error* bisa disebabkan oleh kualitas kabel, koneksi antar *device* dan lain sebagainya. Raspberry mendadak tidak kehilangan komunikasi dengan antarmuka I²C sehingga data tidak diterima. Berikut adalah gambar saat terjadi *error* pada jalur I²C yang ditunjukkan pada gambar 4.19 di bawah ini.



```
pi@raspberrypi: ~/Documents/tugas
File Edit Tabs Help
receive co2: 898 PPM
receive co2: 898 PPM
receive co2: 898 PPM
receive co2: 900 PPM
receive co2: 900 PPM
receive co2: 900 PPM
receive co2: 900 PPM
receive co2: 900 PPM
receive co2: 900 PPM
receive co2: 899 PPM
receive co2: 899 PPM
receive co2: 899 PPM
receive co2: 899 PPM
IO Error
receive co2: 899 PPM
receive co2: 899 PPM
receive co2: 899 PPM
receive co2: 899 PPM
receive co2: 900 PPM
receive co2: 900 PPM
receive co2: 900 PPM
receive co2: 900 PPM
receive co2: 900 PPM
receive co2: 902 PPM
receive co2: 902 PPM
```

Gambar 4.19 Hasil Penerimaan Data Raspberry (Jalur I²C *error*)

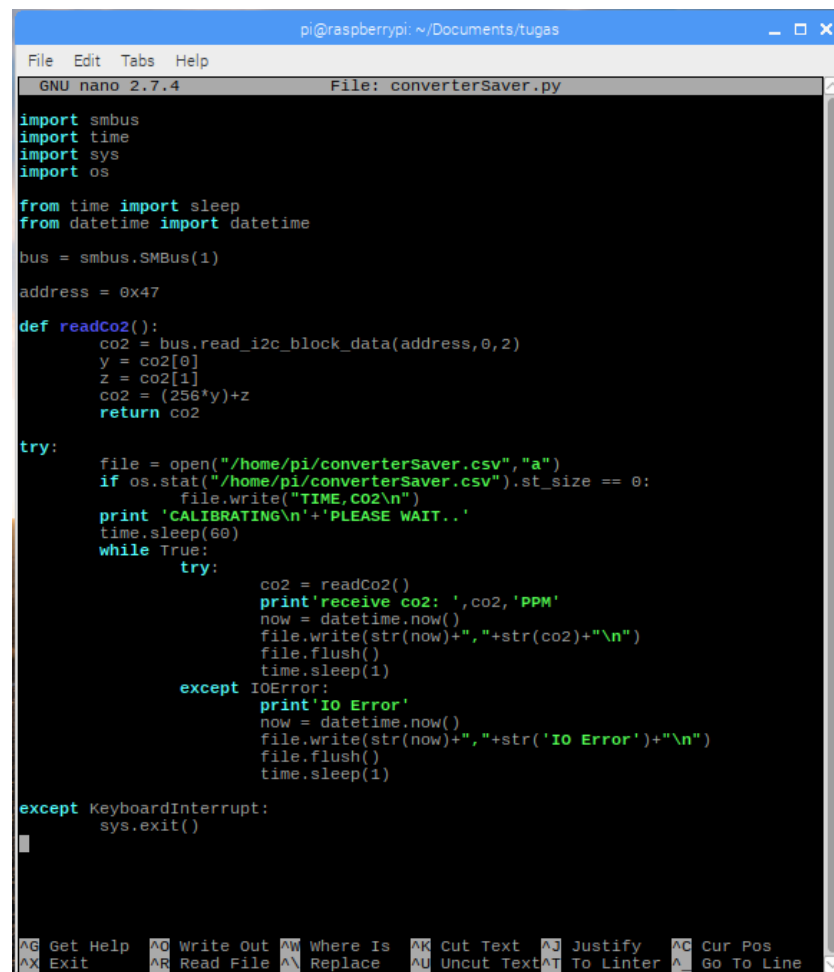
Terjadinya *error* pada jalur I²C sangat jarang terjadi karena penyebab utamanya kemungkinan adalah masalah hardware, kualitas kabel atau kondisi di sekitar *device*. Tabel 4.10 di bawah ini adalah perhitungan persentase *error* berdasarkan data yang sama pada pengujian perhitungan persentasi data *error*.

Tabel 4.10 Persentase *Error* Jalur Antarmuka I²C

NO	WAKTU	JUMLAH DATA	JUMLAH DATA JALUR	PERSENTASE ERROR (%)
1	1 JAM KE-1	3593	0	0
2	1 JAM KE-2	3592	0	0
3	1 JAM KE-3	3593	0	0
4	1 JAM KE-4	3593	0	0
5	1 JAM KE-5	3593	0	0
RATA-RATA			0	0

Dalam pengambilan data antarmuka I²C selama 5 jam berturut-turut, pada raspberry tidak terjadi kesalahan jalur atau kesalahan I/O (*input output*). Hal ini dibuktikan dari tabel 4.10 sebelumnya, dimana persentase *error* hanya 0% pada tiap jam. Jadi, dapat disimpulkan bahwa antarmuka I²C yang telah dikerjakan dapat berfungsi dengan semestinya.

Pada pengujian ini, program penerimaan data dari antarmuka I²C pada raspberry mengalami penambahan program. Program tersebut adalah program penyimpanan data dalam format .csv yang bisa ditampilkan dalam *microsoft excel* dan *libre office calc*. program pada raspberry secara keseluruhan ditunjukkan pada gambar 4.20 di halaman selanjutnya.



```
pi@raspberrypi: ~/Documents/tugas
GNU nano 2.7.4 File: converterSaver.py
import smbus
import time
import sys
import os

from time import sleep
from datetime import datetime

bus = smbus.SMBus(1)

address = 0x47

def readCo2():
    co2 = bus.read_i2c_block_data(address,0,2)
    y = co2[0]
    z = co2[1]
    co2 = (256*y)+z
    return co2

try:
    file = open("/home/pi/converterSaver.csv", "a")
    if os.stat("/home/pi/converterSaver.csv").st_size == 0:
        file.write("TIME,CO2\n")
    print 'CALIBRATING\n'+ 'PLEASE WAIT..'
    time.sleep(60)
    while True:
        try:
            co2 = readCo2()
            print'receive co2: ',co2,'PPM'
            now = datetime.now()
            file.write(str(now)+","+str(co2)+"\n")
            file.flush()
            time.sleep(1)
        except IOError:
            print'IO Error'
            now = datetime.now()
            file.write(str(now)+","+str('IO Error')+"\n")
            file.flush()
            time.sleep(1)
except KeyboardInterrupt:
    sys.exit()

```

⌘ Get Help ⌘ Write Out ⌘ Where Is ⌘ Cut Text ⌘ Justify ⌘ Cur Pos
⌘ Exit ⌘ Read File ⌘ Replace ⌘ Uncut Text ⌘ To Linter ⌘ Go To Line

Gambar 4.20 Program Penyimpanan Data Raspberry Pi