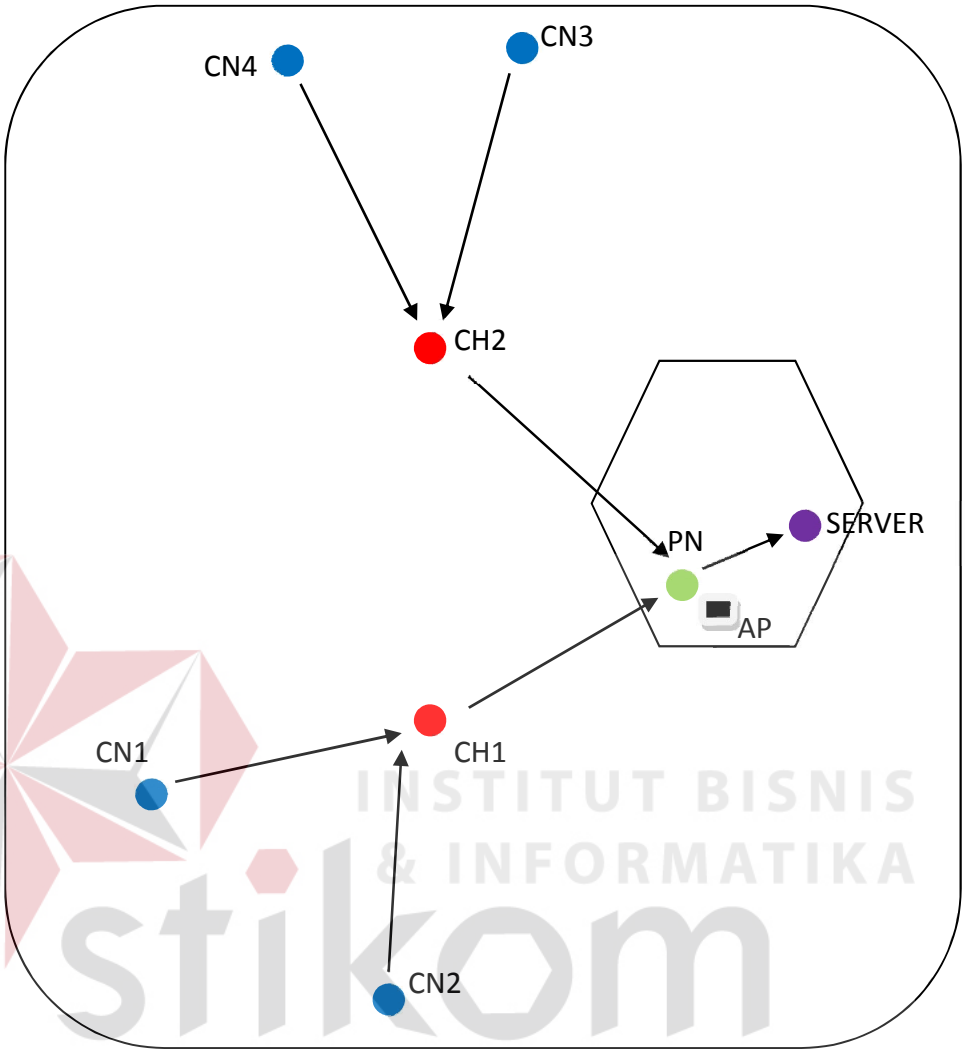
BAB III

METODE PENELITIAN DAN PERANCANGAN SISTEM

3.1 Metode Penelitian

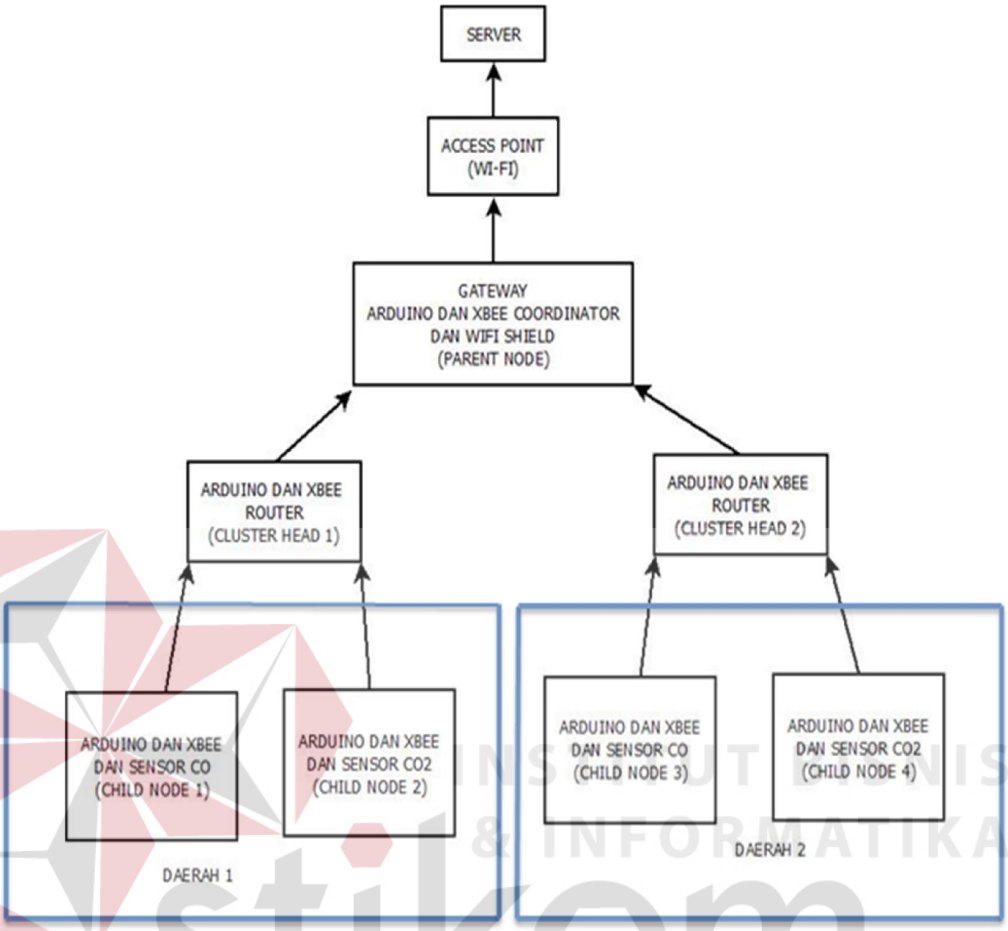
Pada metode penelitian tugas akhir ini dibuat rancang bangun *wireless sensor networks* untuk pemantauan kualitas udara di daerah Intitut Bisnis Stikom Surabaya. Pemantauan dilakukan untuk mendeteksi kadar CO dan CO_2. Terdapat sensor pada masing – masing *Child Node* (CN) dan *Child Node* tersebut akan diletakkan pada 2 titik daerah yang berbeda yang terhubung pada sebuah *cluster*. Sensor CO (CN 1) dan CO_2 (CN 2) akan diletakkan pada sebuah daerah disekitar Intitut Bisnis Stikom Surabaya, yang akan mentransmisikan rata – rata data setiap menit kepada *Cluster Head* 1 (CH 1). Sedangkan kedua sensor yang lain yaitu CO (CN 3) dan CO_2 (CN 4) akan diletakkan di daerah berbeda disekitar Intitut Bisnis Stikom Surabaya, yang akan mentransmisikan nilai tertinggi data setiap menit kepada *Cluster Head* 2 (CH 2). Data yang telah diterima oleh masing – masing *Cluster Head* (CH) akan ditransmisikan kembali kepada sebuah *Parent Node* (PN). Data yang diterima oleh PN yang telah terintegrasi dengan jaringan *wifi*, agar *user* dapat memantau keadaan udara, melalui sebuah aplikasi yang dibangun dari Visual Basic, disekitar Institut Bisnis dan Informatika Stikom Surabaya. Sedangkan pengiriman data yang berasal dari masing – masing CN hingga diterima oleh PN ditransmisikan menggunakan *wireless zigbee networks* . Peletakan sensor dapat dilihat pada gambar 3.1 dibawah ini:



Gambar 3.1 Gambar Peletakan *Child Node (*CN), *Cluster Head* (CH), *Parent Node* (PN), *Access Point* (AP) dan Server

3.2 Model Perancangan

Pada perancangan ini penulis menggambarkan perancangan sistemnya seperti pada gambar 3.2 berikut.



Gambar 3.2 Gambar Perancangan

Dari Gambar 3.2 didapatkan bahwa setiap *node* WSN memiliki tugas berbeda-beda seperti berikut:

a) *Child Node* 1

Pada *node* ini, *node* bertanggung jawab sebagai pencatat hasil pemantauan CO pada titik 1 daerah 1 kemudian mengirimkan datanya pada *Cluster Head* 1.

b) *Child Node* 2

Pada *node* ini, *node* bertanggung jawab sebagai pencatat hasil pemantauan CO_2 pada titik 2 daerah 1 kemudian mengirimkan datanya pada *Cluster Head* 1.

c) *Child Node* 3

Pada *node* ini, *node* bertanggung jawab sebagai pencatat hasil pemantauan CO pada titik 1 daerah 2 kemudian mengirimkan datanya pada *Cluster Head 2.*

d) *Child Node* 4

Pada *node* ini, *node* bertanggung jawab sebagai pencatat hasil pemantauan CO_2 pada titik 2 daerah 2 kemudian mengirimkan datanya pada *cluster head* 2.

e) *Cluster Head* 1

Cluster Head 1 merupakan *node router*. Pada *node* ini, *node* menerima data dari rata – rata CO *Child Node* 1 dan rata – rata CO_2 *Child Node* 2. *Node* ini bertanggung jawab atas data yang sudah dikirim oleh *Child Node.* Masing – masing *child node* akan diberikan ID yang berbeda, ini bertujuan agar data tidak tertukar. Dan kemudian akan mentransmisikannya pada *Parent Node.*

f) *Cluster Head* 2.

Cluster Head 2 merupakan *node router*. Pada *node* ini, *node* menerima data dari *Child Node* sensor 3 dan *Child Node* sensor 4 Dan kemudian akan mentransmisikannya pada *Parent Node.*

g) *Parent Node*

Pada *Parent Node* ini, *node* menerima data dari *Child Node* 1, *Child Node* 2, *Child Node* 3 dan *Child Node* 4 melalui masing – masing *Cluster Head*. Selanjutnya data diproses untuk di integrasikan yang semula *zigbee network* menuju ke *wifi network*. Perlu diketahui bahwa *zigbee network* berjalan pada *layer* 2 (*data link*) dan *layer* 1 (*physical*) sehingga tidak dapat melakukan proses *routing* dan tidak mengetahui IP *address* yang biasa dilakukan pada *layer* 3 (*network*), dan tidak dapat

menjalankan protokol *transport* yaitu UDP dan TCP yang ada pada *layer* 4 (*transport*), tetapi data kualitas udara yang berasal dari *parent node* ke server akan dikirim kan melalui jaringan WiFi yang memiliki kemampuan tersebut, maka dari itu diperlukan suatu modul yang memungkinkan untuk melakukannya, yaitu menggunakan modul *tonylabs* CC3000 *wifi shield* alat ini dapat melakukan proses *routing* dan dapat menggunakan protokol *transport* UDP atau TCP. Dan pada penelitian ini digunakan protokol UDP untuk mengirimkan datanya. Pada modul *wifi shield* dikonfigurasi SSID dan *password access point* (AP) yang dituju dan data kualitas udara tersebut diamankan menggunkaan WPA2, konfigurasi ini dilakukan pada dua sisi yaitu sisi *node coordinator* dengan CC3000 *wifi shield* dan AP.

h) *Access Point*

Access Point adalah suatu alat yang dapat menyediakan akses kepada banyak *user* untuk dapat mengakses internet menggunkaan jaringan *wireless*, fungsinya banyak, digunakan pada perkantoran, rumah sakit, dan tempat outdour.

Pada penelitian ini difungsikan untuk menyediakan akses komunikasi pada komputer server dengan *Parent Node* yang berkomunikasi dengan menggunakan jaringan *wifi.*

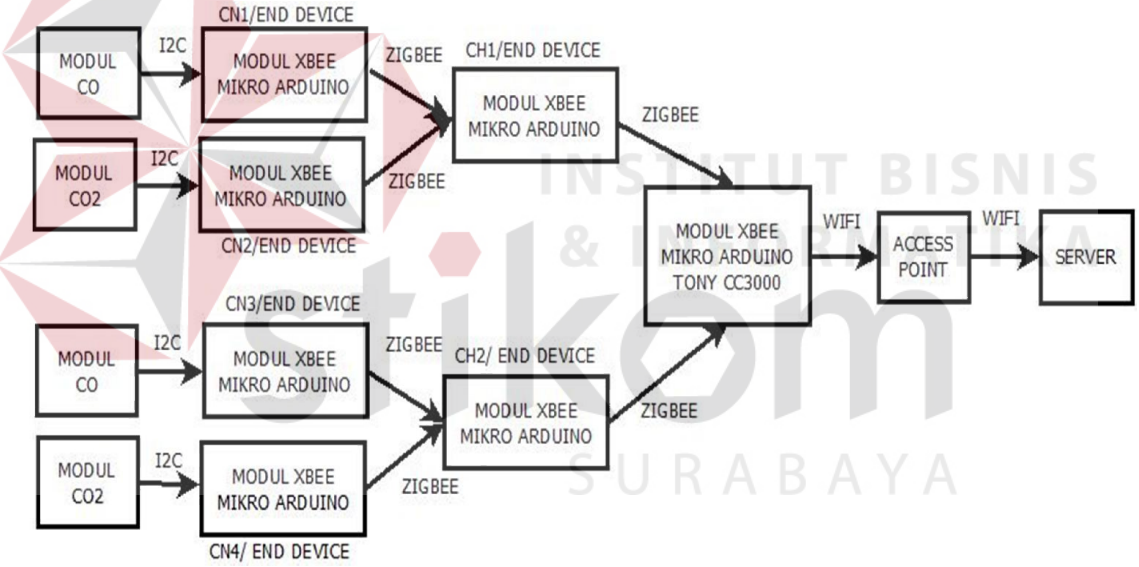
i) Komputer *Server*

Komputer *server* disini sebagai pusat dari penerima data dan digunakan oleh user untuk melihat hasil monitoring kualitas udara dari CN 1, CN 2, CN 3, dan CN 4 dari aplikasi yang sudah dibuat menggunakan *visual basic* (VB). Komputer *server*

berkomunikasi secara *unicast* (*point to point*) dengan *Parent Node* melalui *access point*.

3.3 Perancangan Sistem

Dalam tugas akhir ini, penulis hanya akan memfokuskan pengiriman data berupa rata – rata atau nilai tertinggi dari masing – masing *cild node* kepada server dengan menggunakan metode cluster. Dan juga mengamati hasil unjuk kerja jaringan pada model *cluster* yang telah ditentukan sebelumnya. Adapun perancangan blok diagram ditunjukkan sebagaimana gambar 3.3:



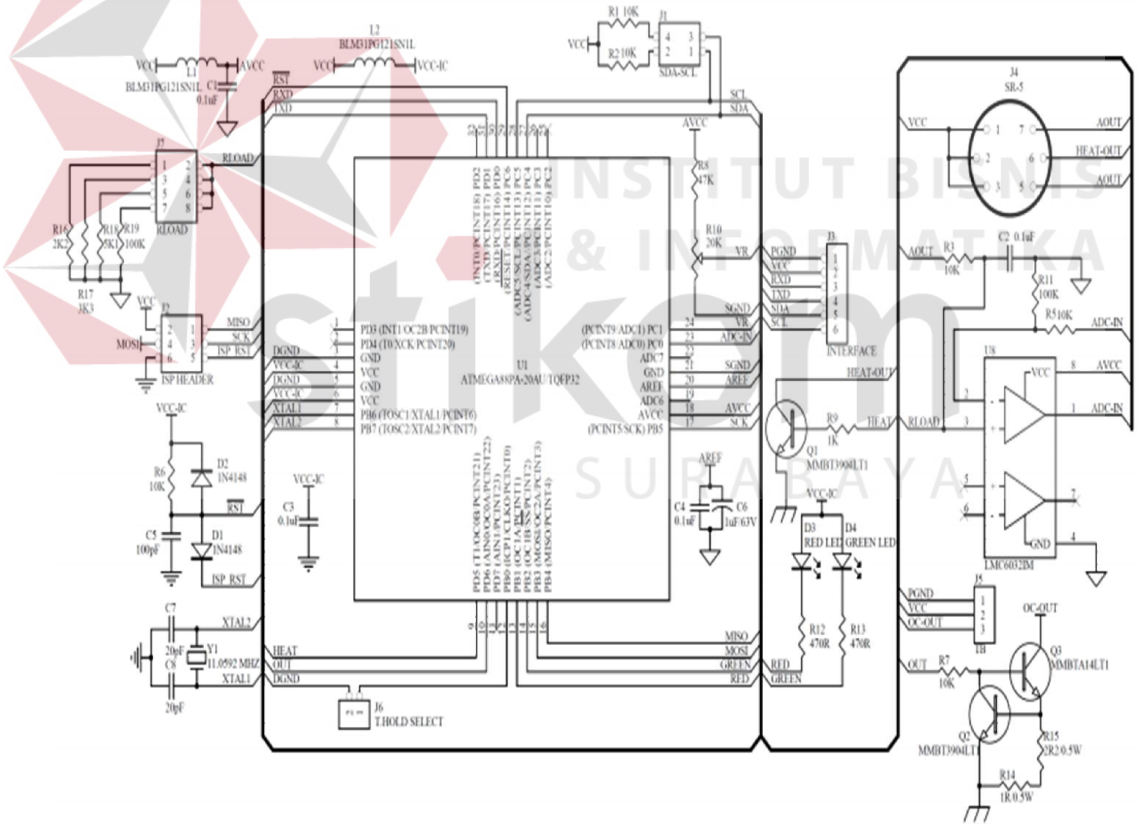
Gambar 3.3 Blok Diagram

3.4 Perancangan Perangkat Keras

3.4.1 Perancangan DT-SENSE CARBON DIOXIDE SENSOR

Untuk dapat mendeteksi kadar CO dalam udara maka dibutuhkan sebuah sensor. Sensor yang digunakan pada penelitian ini adalah DT-SENSE CARBON

MONOXIDE. SENSOR.DT-SENSE CARBON MONOXIDE SENSOR merupakan modul sensor gas yang dapat digunakan untuk menentukan kadar karbon monoksida yang terdapat pada udara. Modul ini berbasiskan sensor MQ-7. Adapun perancangan rangkaian DT-SENSE CARBON MONOXIDE SENSOR menggunakan komunikasi I2C(Inter-Integrated Circuit).Data yang dikeluarkan berupa data digital sesuai yang sudah dijelaskan pada BAB II. Dan skenmatik sensor dapat dilihat pada gambar di bawah ini. Yang membedakan modul sensor MQ-7 dengan MG-811 adalah pemakaian resistor pada MQ-7 resistor yang digunakan adalah 3k3Ω

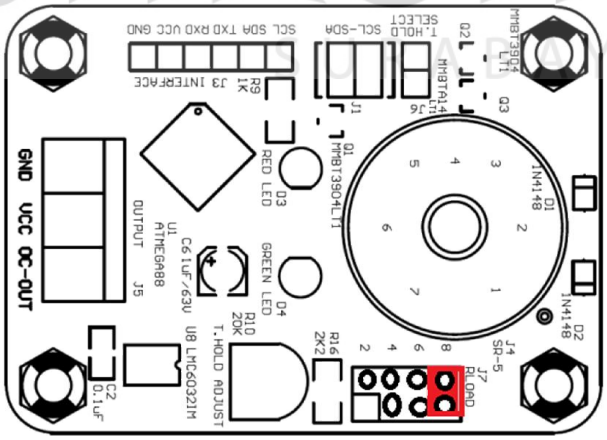


Gambar 3.4 Skematik Sensor DT-SENSE CARBON MONOXIDE SENSOR

(data sheet DT-SENSE)

3.4.2 Perancangan DT-SENSE CARBON DIOXODE SENSOR

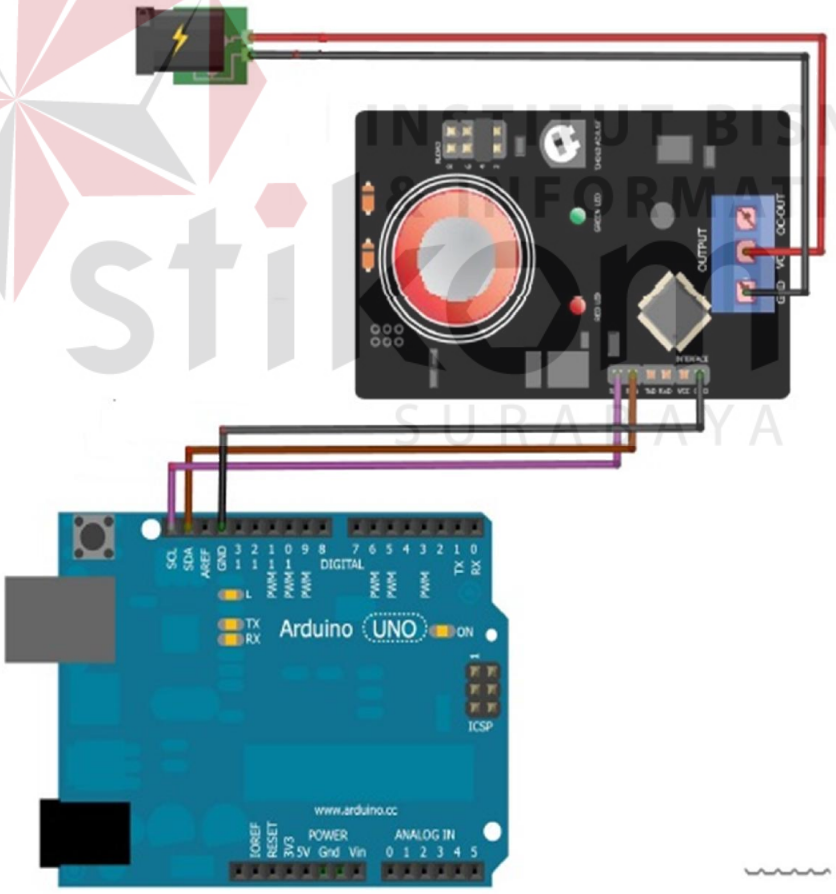
Untuk dapat mendeteksi kadar CO_2 dalam udara maka dibutuhkan sebuah sensor. Sensor yang digunakan pada penelitian ini adalah DT-SENSE CARBON DIOXODE SENSOR. DT-SENSE CARBON DIOXODE SENSOR merupakan modul sensor gas yang dapat digunakan untuk menentukan kadar karbon dioksida yang terdapat pada udara.Modul ini berbasiskan sensor MG-811. Adapun perancangan rangkaian DT-SENSE CARBON DIOXODE SENSOR menggunakan komunikasi I2C sam halnyad engan MQ-7.Dan pada dasarnya skematik sensor MG-811 sama dengan MQ-7, hanya saja yang membedakan pemakaian resistor antara modul pada MG-811 dan MQ-7. Pada MG-811 menggunakan resistor 100kΩ, jika dilihat pada modul sensor yang digunakan maka penggunaan resistor yang dikaitkan seperti pada gambar 3.5 dibawah ini, dan jika dilihat pada gambar skematik pada gambar 3.4 diatas maka menunjukan bahwa posisi tersebut merupakan resistor dengan nilai 100kΩ



Gambar 3.5 Perancangan Sensor DT-SENSE CARBON DIOXIDE SENSOR (data sheet DT-SENSE)

3.4.3 Perancangan Sensor dengan Arduino

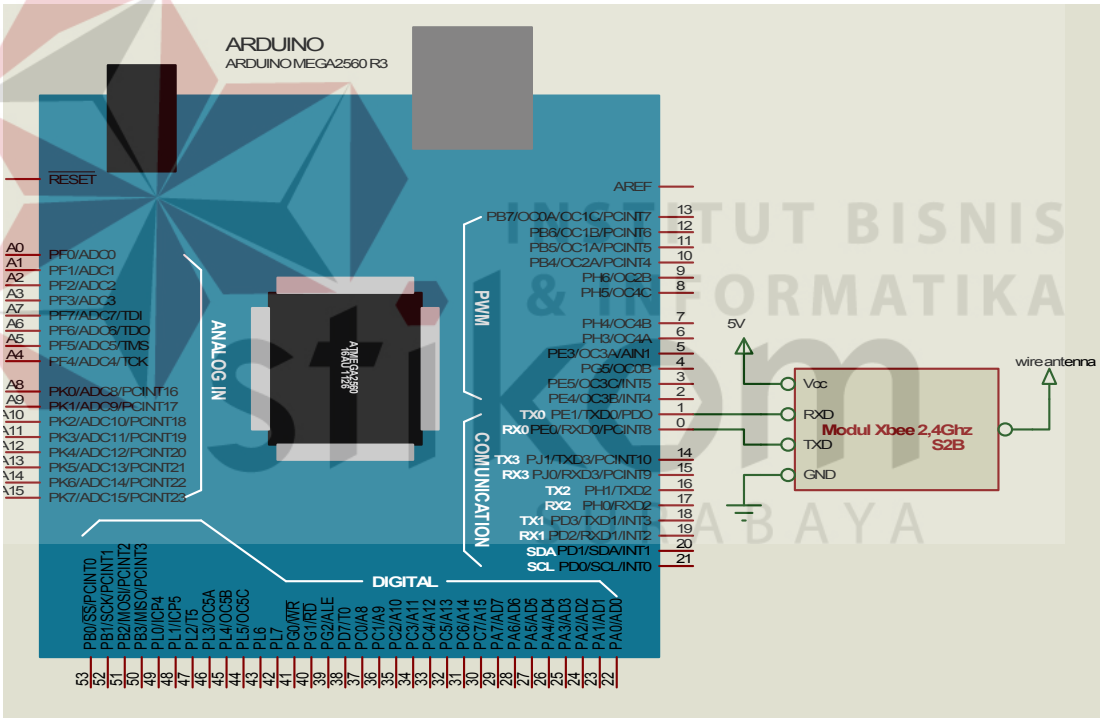
Untuk dapat mendeteksi kadar CO dan CO_2, maka data yang dipperoleh oleh masing-masing sensor harus dioleh kembali oleh mikrokontroler agar hasilnya dapat dipergunakan oleh user. Seperti yang telah dijelaskan diatas, bahwa pembacaan sensor dilakukan melalui komunikasi *Inter Integrated Circuit* (I2C). sehingga pembacaan sensor dilakukan melalui pin *Serial Clock* (SCL) dsn *Serial Data* (SDA) pada Arduino, seperti yang digambarkan pada gambar dibawah ini. Hal ini berlaku pada modul MG-811 maupun MQ-7.



Gambar 3.6 Perancangan Sensor dengan Arduino

3.4.4 Perancangan Rangkaian Xbee Zigbee S2B

Agar modul arduino dapat berkomunikasi secara serial *wireless* dengan perangkat lain, maka dibutuhkan rangkaian *wireless* yang dalam perancangan ini menggunakan modul Zigbee S2B. modul *zigbee* dapat berkomunikasi *wireless* dan diakses menggunakan komunikasi serial TTL (*Time to Live*). Adapun port serial yang digunakan untuk pengendalian dan pembacaan modul Xbee adalah TX0 dan RX0 pada modul arduino sebagaimana ditunjukkan pada Gambar 3.7:



Gambar 3.7 Hubungan Rangkaian Xbee dan Arduino

3.4.5 Arduino Mega 2560

Pada gambar 3.3 terdapat 4 arduino yang memiliki fungsi yang sama yaitu membaca sensor yang memiliki karakteristik sama dalam cara pembacaan data sensor, pembacaan data dilakukan dengan cara inputan yang berasal dari sensor

diletakkan pada PORT SCL dan SDA, untuk membaca nilai dari sensor tersebut digunakan komunikasi I2C di dalam modul arduino.

Pada modul arduino juga dilakukan pemberian identitas pada data yang akan ditrasmisikan. Yang artinya data yang dikirim mendapatkan tambahan identitas setiap *Child Node* seperti pada gambar 3.8. Dari tambahan itu yang akan membuat *Cluster Head* dapat mengenali darimana asal data. XX pada gambar 3.8 terdiri dari DR untuk *Child Node* 1, MR untuk *Child Node 2*, DT untuk *Child Node 3* dan MT untuk *Child Node 4*. Identitas ini akan dikirim bersama dengan inti data yang *Child Node* kirimkan ke *Cluster Head.* Dari *Cluster Head* mengirimkan data ke *Parent Node* dengan menambahkan identitas tambahan. Pada isi data yang diterima *Cluster Head* 1 nantinya terdapat simbol DR (untuk data dari *Child Node* 1) dan MR (untuk data dari *Child Node* 2) sedangkan *Cluster Head 2* terdapat simbol DT (untuk data dari *Child Node 3*) dan MT (untuk data dari *Child Node 4*) selanjutnya diikuti inti pesan yang dikirim masing – masing *Child Node.*

FORMAT PENGIRIMAN DATA CN KE CH			
XX	#	*DATA*	#

Gambar 3.8 Format pengiriman data dari *Child Node* ke *Cluster Head*

Berikut penjelasan dari gambar 3.8 :

1. MR : Identitas dari *Child Node* darimana data berasal
2. # : Pemisah antara identitas dengan data

3. DATA : Data dari sensor yang dikirimkan
4. # : Penanda akhir pengiriman

3.4.6 Xbee

Untuk mengirimkan data dari masing – masing CN ke CH kemudian dari CH ke PN diperlukan sebuah pemancar data. Dalam penelitian ini penulis menggunakan Xbee Series 2 untuk pemancar data.Konfigurasi yang dilakukan pada Xbee sangat penting, agar data dapat dikirimkan ke alamat yang sesuai.

Untuk mengkonfigurasi Xbee tersebut dibutuhkan sebuah *software*. *Software* yang biasa digunakan untuk mengkonfigurasi Xbee salah satunya ialah X-CTU.

Xbee dikonfigurasi untuk menjadi *end device* dalam mode AT untuk Xbee yang terdapat pada CN dan CH pada X-CTU menggunakan pilihan *Function Set* "*ZNET 2.5 ROUTER/END DEVICE AT",* dan Xbee dikonfigurasi untuk menjadi*coordinator*dalam mode ATuntuk Xbee yang terdapat pada *parent node*pada X-CTU menggunakan pilihan *Function Set* "*ZNET 2.5 COORDINATOR AT"*. Dalam mengkonfigurasi Xbee *series* 2 hal yang terpenting ialah mengisi nilai PAN ID, DH dan DL.

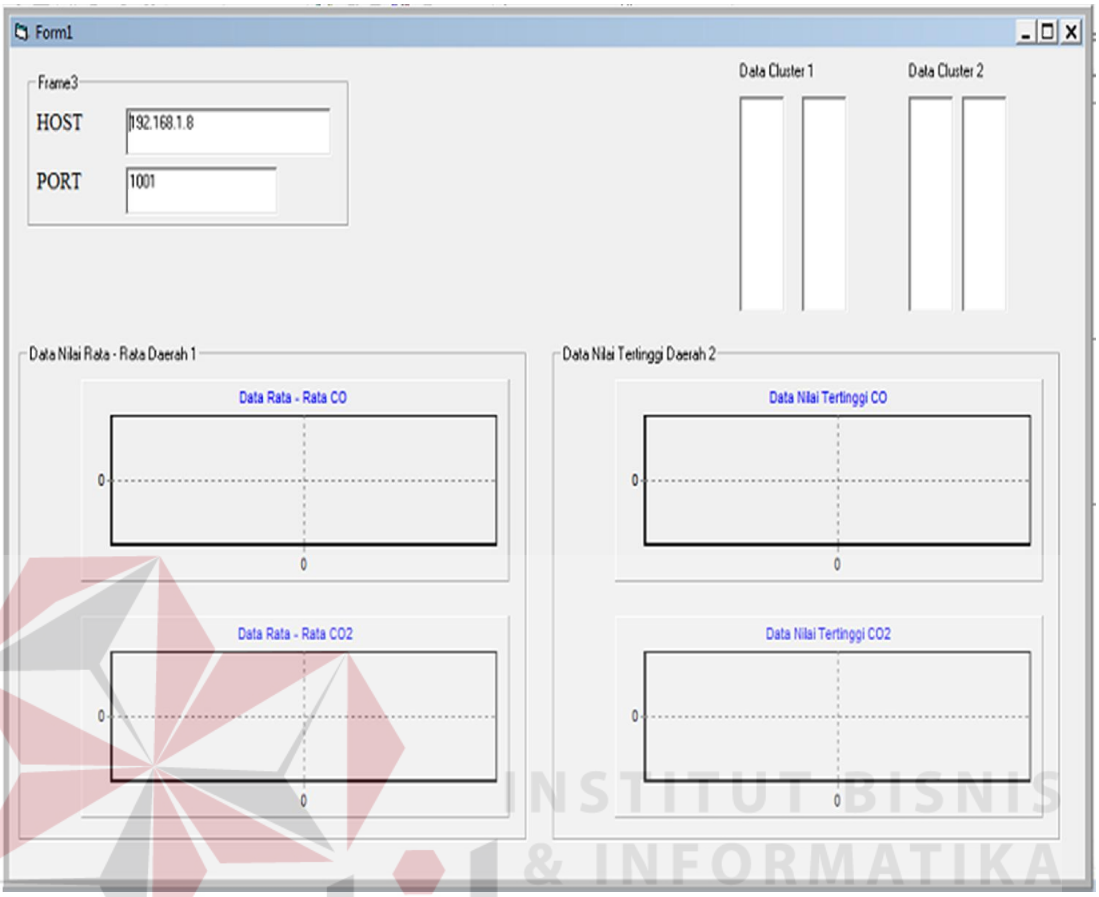
Langkah pertama untuk dapat berkomunikasi dalam satu jaringan, maka PAN ID antar Xbee harus diisi dengan nilai yang sama. Langkah kedua yaitu mengisi DH dengan ID yang terdapat pada Xbee. Dan Langkah terakhir yaitu mengisi DL, pada CN nilai DL diisi sesuai dengan nilai Xbee yang digunakan sebagai CH, pada CH nilai DL diisi sesuai dengan nilai Xbee pada PN, sedangkan pada PN nilai DL diisi dengan nilai Xbee pada CH Hal ini dilakukan agar Xbee yang digunakan pada

CN hanya dapat berkomunikasi dengan Xbee pada CH dan Xbee yang digunakan pada CH hanya dapat berkomunikasi dengan Xbee pada PN.

3.4.7 Visual Basic

Visual basic pada komputer atau *end device* berfungsi untuk mengolah data yang dikirimkan oleh PN. Data yang diterima tersebut masih berupa sekumpulan informasi dan kode yang masih lengkap yang berup *header* dan data (sesuai dengan protokol), sehingga diperlukan pemisahan data serta pengelompakan pada data tersebut agar didapatkan sebuah data beserta informasi yang diinginkan dari data tersebut. Seperti yang dijelaskan pada gambar 3.8 pengelompokan data sesuai dengan kode yang terdapat pada satu paket data. Selanjutnya data yang sudah dipisah di simpan sesuai dengan pengelompokan data. Hal ini dilakukan agar data yang diperoleh nantinya dapat dianalisa, sehingga dapat diketahui kemapuan algoritma dari sistem transmisi dengan *cluster* ini. Selanjutnya dari data yang telah dikelompokkan dan dipisah ditampilkan pada sebuah grafik agar dapat dilihat oleh *user*.

Karakter dari pengiriman data ini adalah *real time,* sehingga *protocol* yang digunakan adalah *protocol* UDP. Dikarenakan data yang diperoleh menggunakan *protocol* UDP, maka diperlukan adanya pengaturan IP pada komputer secara otomatis untuk membuat computer server dengan alat berada dalam sebuah jarngan. Perancangan tampilan yang digunakan pada Visual basic dapat dilihat pada gambar 3.9 dibawah ini :

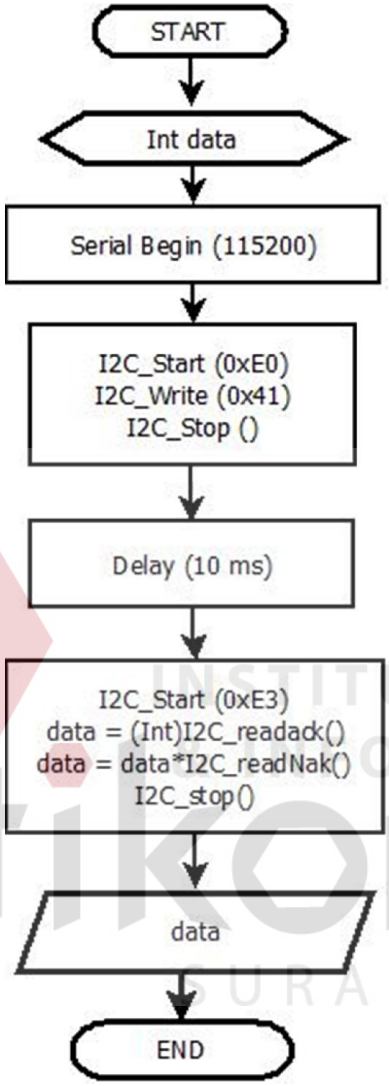


Gambar 3.9 Tampilan pada *End Device*

3.5 Perancangan Perangkat Lunak

Dari perancangan sistem diatas, selain perancangan *hardware,* juga dibutuhkan perancangan perangkat lunak untuk menjalankan perancangan *hardware* yang telah dibuat.Perangkat lunak terdiri dari beberapa algoritma perancangan dari sistem yang ditangani oleh pengontrol.

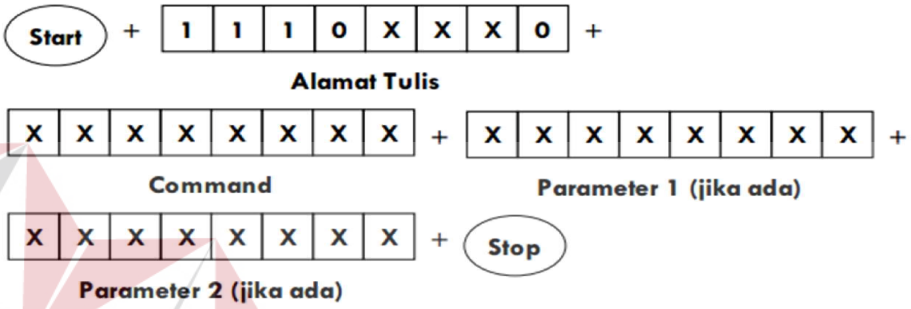
3.5.1 Algoritma Pembacaan Sensor



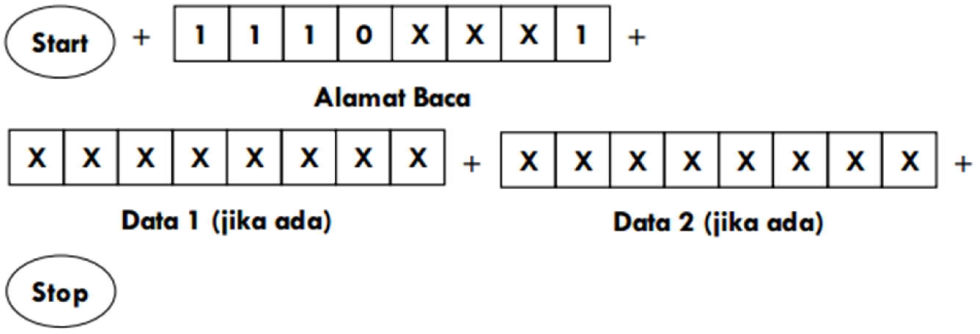
Gambar 3.10 *Flowchart* pembacaan sensor

Seperti yang sudah dijelaskan diatas, cara pembacaan data sensor DT-SENSE CARBON MONOXIDE SENSOR dan DT-SENSE CARBON DIOXIDE SENSOR sama, sehingga algoritma yang digunakanpun sama. Dan data keluaran dari kedua sensor tersebut sudah berupa data digital. Pada dasarnya, cara pembacaan sensor melalui I2C adalah *interface* I2C akan bertindak sebagai slave dengan alamat

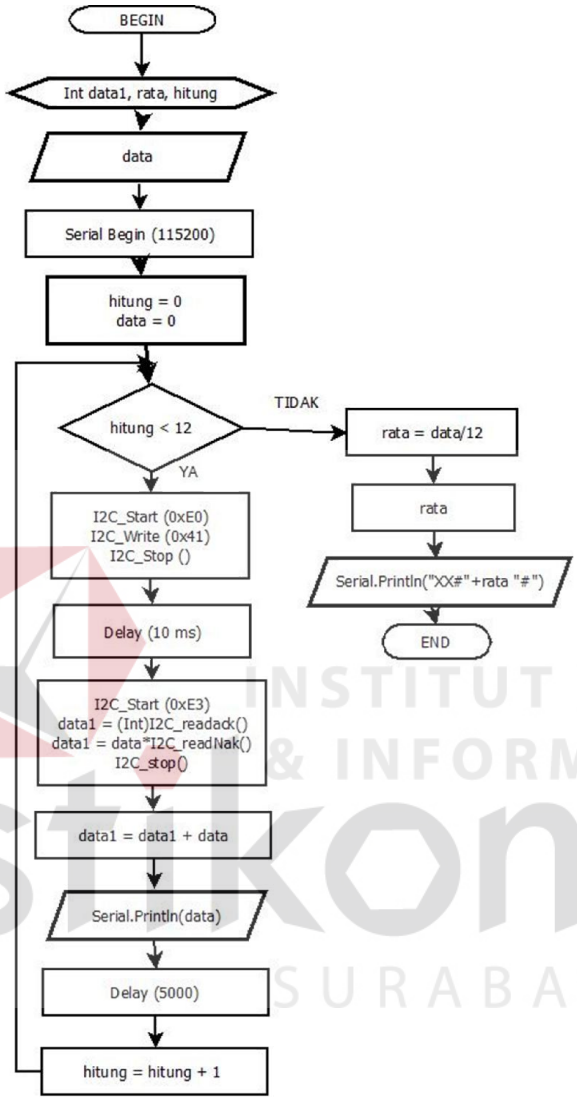
sesuai yang telah ditentukan sebelumnya, semua perintah data yang dikirim melalui I2C diawali dengan *start condition* dan kemudian diikuti dengan pengiriman alamat modul DT- SENSE GAS SENSOR, selanjutnya master mengrimkan 1 byte data pengiriman perintah, dan setelah seluruh parameter perintah terkirim, selanjutnya data yang dikirim adalah *end condition*, seperti yang dijelaskan pada ilustrasi dibawah ini:



Sedangkan pembacaan data sensor, dilakukan secara 2 tahap, karena sebuah data / parameter memiliki range yang lebih besar dari 255 desimal (lebih besar dari 1 byte) Satu byte data MSB dikirim lebih dahulu kemudian diikuti dengan data LSB. Misalnya parameter <dataSensor> yang memiliki range 0 – 1023. Jika (dataSensor) bernilai 1000 maka byte MSB yang dikirim adalah 3 dan byte LSB yang dikirim adalah 232 (0xE8). Dan karena data yang diperoleh berupa digital, sehingga deperlukan perhitungan untuk menjadikan data berupa persentase.



3.5.2 Algoritma Pengiriman Data Sensor



Gambar 3.11 *Flowchart* pengiriman sensor

Pada dasarnya konsep dari pengiriman sensor DT-SENSE CARBON MONOXIDE SENSOR DAN DT-SENSE CARBON DIOXIDE SENSOR ini adalah data dikirim setiap 5 detik tanpa identitas asal data, setelah satu menit data yang terkumpul tersebut dirata- rata atau dicari nilai tertinggi dan diberi identitas kemudian dikirim ke CH. Pada flowchart tersebut tertulis hitung < 12 karena satu menit ada 60

detik dan data dikirim setiap 5 detik sehingga 60 detik dibagi 5 detik ada 12 kali hitung tanpa identitas. Untuk analisa data, dilakukan pengambilan data selama 30 menit, sehingga data yang terkumpul berjumlah ± 30 data

Pada tugas akhir ini, penulis menggunakan Arduino Mega2560 sebagai mikrokontrolernya. *Software* yang digunakan untuk memprogram arduino tersebut ialah *software* Arduino IDE. Dan dari algoritma yang dibuat diatas maka dibuatlah program seperti pada penjelasan dibawah ini:

Berikut contoh pemrogaman modul arduino Mega 2560 pada *child node* yang diprogam pada Arduino IDE

a. Pembuatan variable

Dalam pembuatan variable, terdapat beberapa variable yang digunakan oleh penulis seperti pada algoritma di atas. Berikut sebagai contoh :

```
int data, rata, data1, hitung;
```

b. Fungsi void setup

Dalam fungsi void setup perintah akan dibaca 1 kali setelah program berjalan. Berikut sebagai contoh :

```
void setup()
{
  i2c_init();
  Serial.begin(115200);
}
```

Dikarenakan pengiriman data menggnakan *interface* I2C, maka deperlukan adanya inisialisasi I2C pada awal program.

c. Fungsi void loop

Dalam void loop perintah akan dibaca berulangkali selama fungsi bacasensor masih berjalan dan akan menampilkan rata atau nilai tinggi dari sensor .

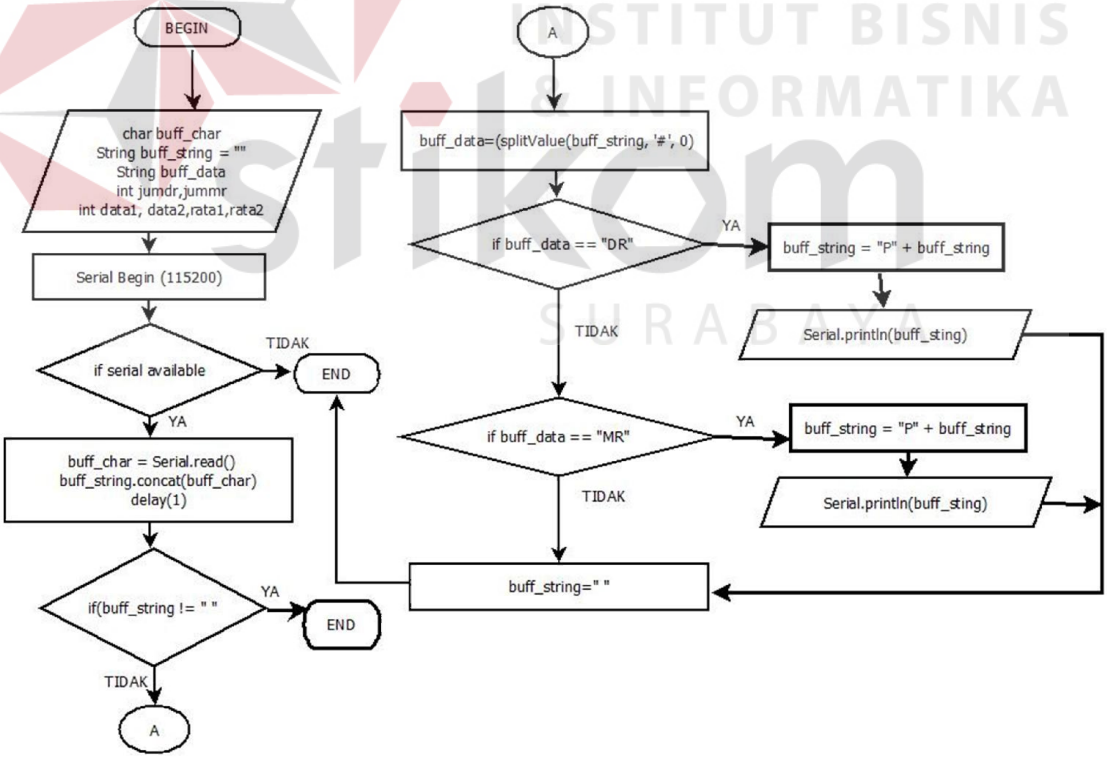
```
      bacaSensor();
      Serial.print("DR#");
{     Serial.println(rata);         //send data gas sensor A
to
}
```

d. Fungsi void bacaSensor

Dalam void bacaSensor dilakukan pengambilan data berulang kali selama hitung belum memenuhi target yaitu 12, seperti yang digambarkan pada *flowchart* gambar 3.10. Selama itu, setiap lima detik mikrokontroler akan mengambil data yang berasal dari sensor. Cara pembacaan sensor adalah sensor akan mengirimkan frekuensi dan menerima data setiap 10ms, serta meletakkan data sensor pada alamat yang sudah ditentukan. Data tersebut akan ditampilkan tanpa menggunakan header agar penulis dapat menganalisis hasil rata – rata yang akan dikirimkan nantinya. Setelah target terpenuhi maka data akan di rata- rata lalu dikirimkan ke *Cluster Head.* Berikut sebagai contoh :

```
        while ( hitung < 12 )
        {
          i2c_start(0xE0);  //  Alamat  I2C  modul
               gas Sensor A
          i2c_write(0x41);
          i2c_stop();
          delay(10);
          i2c_start(0xE1);
          data = (int) i2c_readAck();
          data = data*255 + i2c_readNak();
          i2c_stop();
          data1= data1+data;
          Serial.println(data);
          delay(5000);
          hitung = hitung + 1;
        }
        rata = data1/12;
```

3.5.3 Algoritma Pengiriman Data dari *Cluster Head* ke *Parent Node*



Gambar 3.12 Flowchart pengiriman Data dari *Cluster Head* ke *Parent Node*

Berikut contoh pemrogaman modul arduino Mega 2560 pada *Cluster Head* yang diprogam pada Arduino IDE

a. Pembuatan variable

Seperti pada pengiriman data pada *Child Node*, *Cluster Head* juga mempunyai variable. Berikut sebagai contoh :

```
char buff_char;
String buff_string = "";
String buff_data;
int jumdr,jummr;
int data1, data2,rata1,rata2;
```

b. Fungsi void setup

Dalam fungsi void setup perintah akan dibaca 1 kali setelah program berjalan. Penggunaan *baudrate* adalah 115200, dikarenakan data yang digunakan untuk mengirimkan data minimum harus menggunakan baudrate 115200 agar data dapat dikirim sampai dengan server. Berikut sebagai contoh :

```
{
   Serial.begin(115200);
   Serial.println("Data yang berasal dari Node 1
   dan Node 2");
   Serial.println("");
   delay(1000);
}
```

c. Fungsi void loop

Pada void loop data yang diterima dikumpulkan menjadi satu di dalam sebuah variable dikarenakan data yang diterima berkarakter *char*. Selanjutnya program akan mendeteksi apakah data memiliki header yang tepat atau tidak, header tersebut akan dikelompokkan kembali menurut asal sensornya agar diketahui

darimana asal sensor. Hal ini juga berfungsi, untuk menyering data yang diterima hanya merupakan data utuh yang berasal dari *Child Node.* Data yang sudah dikelompokkan akan dikirim kembali dengan menggunakan header yang baru. Hal ini berfungsi agar data berulang kali dikirim dan diterima karena kemungkinan salah satu dari tujuan parent node merupakan *Cluster Head*. Sehingga format pengiriman data menjadi sebagai berikut :

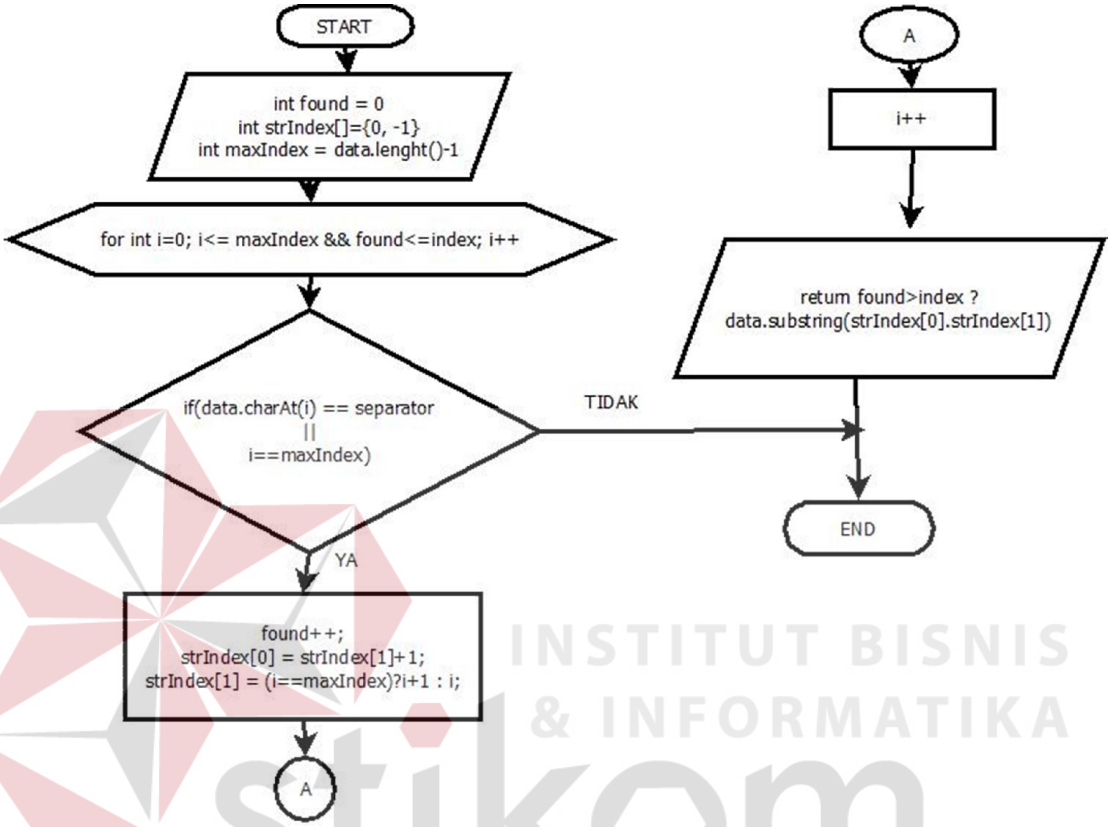
FORMAT PENGIRIMAN DATA CH KE PN			
PXX	**#**	**DATA**	**#**

Gambar 3.13 Format pengiriman data dari *Cluster Head* ke *Parent Node*

Dan progam ditulis sesuai dengan algoritma yang telah dibuat seperti pada gambar 3.11.Berikut contohnya :

```
    while(Serial.available())
  {
    buff_char = Serial.read();
    buff_string.concat(buff_char);
    delay(1);
  }
  if(buff_string != "")
  {
    buff_data=(splitValue(buff_string, '#', 0));
    if(buff_data == "DR")
  {
     buff_string = "P" + buff_string;
     //Serial.print("RATA DR:");
     Serial.println(buff_string);
  }
    if(buff_data == "MR")
  {
    buff_string = "P" + buff_string;
    //Serial.print("RATA MR:");
    Serial.println(buff_string);
  }
    buff_string = "";
  }
```

d. Fungsi string splitValue



Gambar 3.14 Fungsi untuk memisahkan data

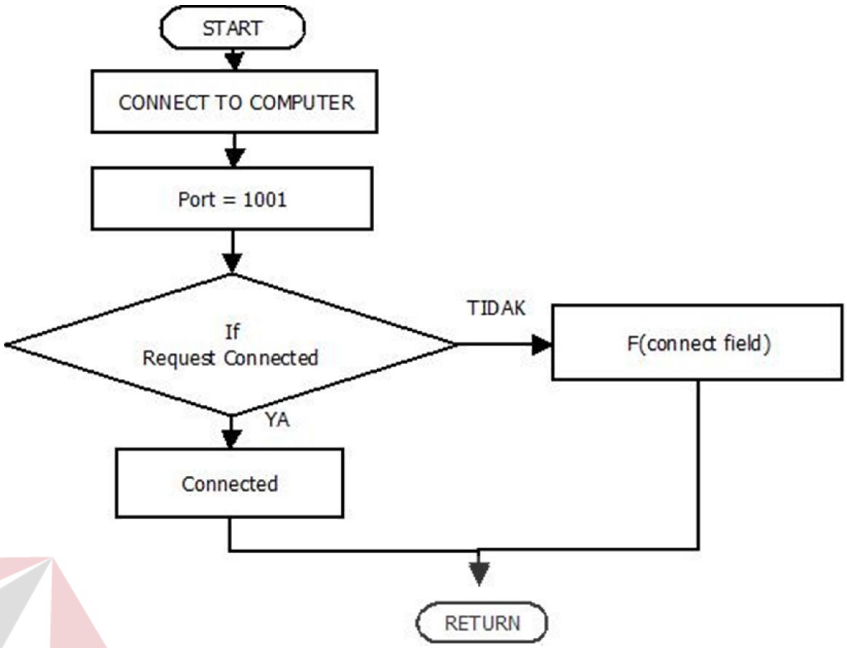
Untuk meringankan kerja algoritma inti *Cluster Head* maka dibuatkan sebuah fungsi pemisah data. Dan progam ditulis sesuai dengan algoritma yang telah dibuat seperti pada gambar 3.13.Berikut contohnya :

```
String splitValue(String data, char separator, int
index)
{
   int found = 0;
   int strIndex[]={0, -1};
   int maxIndex = data.length()-1;
   for(int i=0; i<=maxIndex && found<=index; i++)
   {
      if(data.charAt(i) == separator || i==maxIndex)
      {
         found++;
         strIndex[0] = strIndex[1]+1;
         strIndex[1] = (i == maxIndex) ? i+1 : i;
      }
   }
   return found>index ? data.substring(strIndex[0],
    strIndex[1]) :  "";
}
```

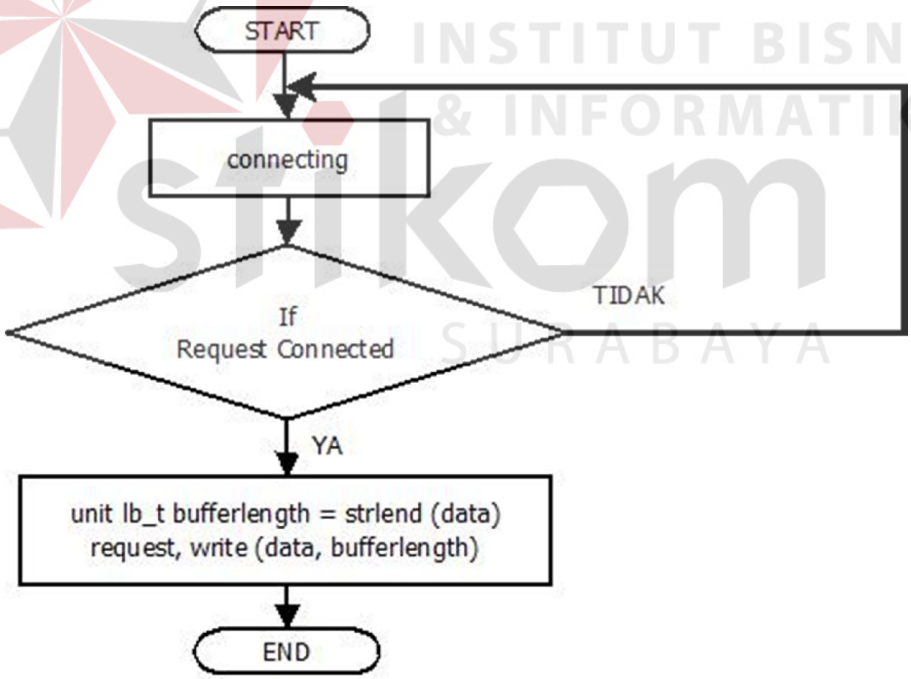
3.5.4 Algoritma Pengiriman Data dari *Parent Node* ke server

Berbeda dengan pengiriman data yang dilakukan oleh *Child Node* sampai menuju *Parent Node.* Pengiriman data yang berasal dari *Parent Node* delakukan dengan cara menggunakan *wifi*. Hal ini menyebabkan perbedaan cara pengiriman dan penerimaan data yang dilakukan pada *Parent Node.* Seperti yag telah dijelaskan, bahwa pengiriman dilakukan denggan menggunakan protokol UDP.

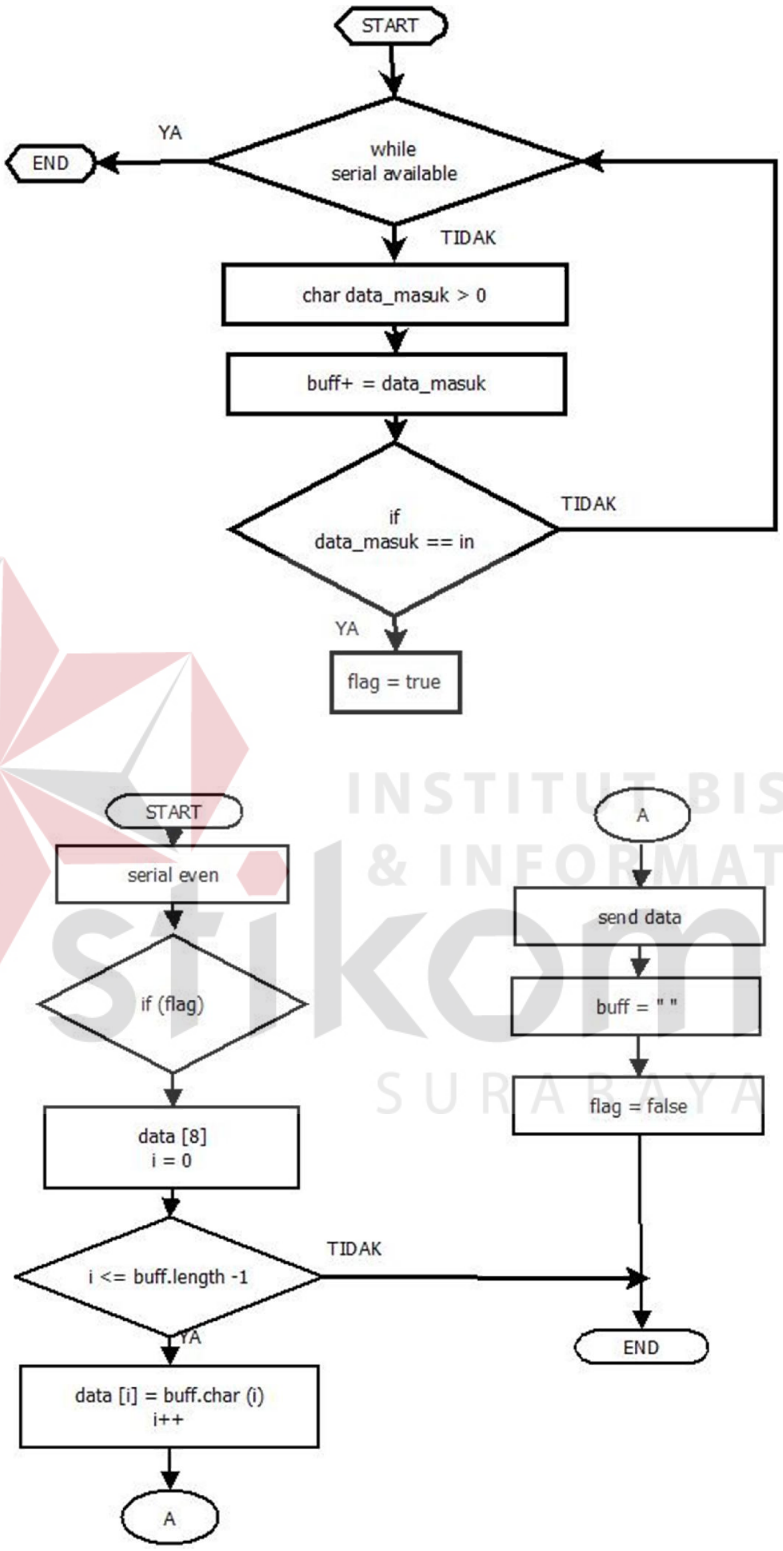
Sebelum data dikirimkan ke server data ditampilkan terlebih dahulu pada sebuah *end device,* hal ini dilakukan agar penulis dapat menganalisa hasil pengiriman data yang telah dilakukan, pengkodingan dilakukan pada arduino. Selanjutnya pengiriman data kepada *server* dilakukan melaui komputer, dalam hal ini menggunakan program pada Visual Basic.Maka, dibuatlah algoritma pengiriman data pada *parent node* pada arduino seperti gambar 3.14 dibawah ini:



Gambar 3.15 *Flowchart* koneksi antara alat dengan *wifi*



Gambar 3.16 *Flowchart* pengiriman data menggunakan *wifi*

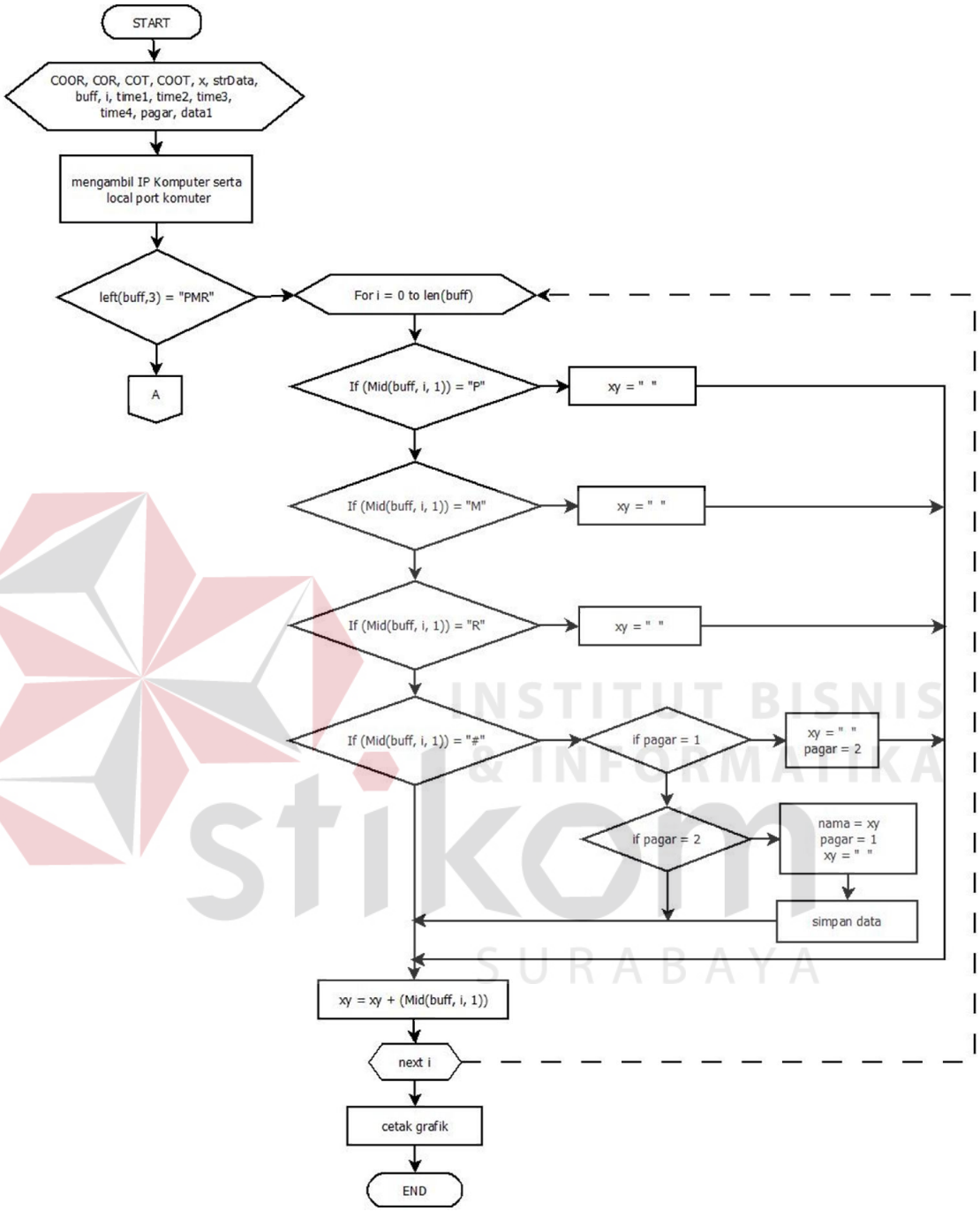


Gambar 3.17 *Flowchart* pengiriman Data dari *Parent Node* ke *server*

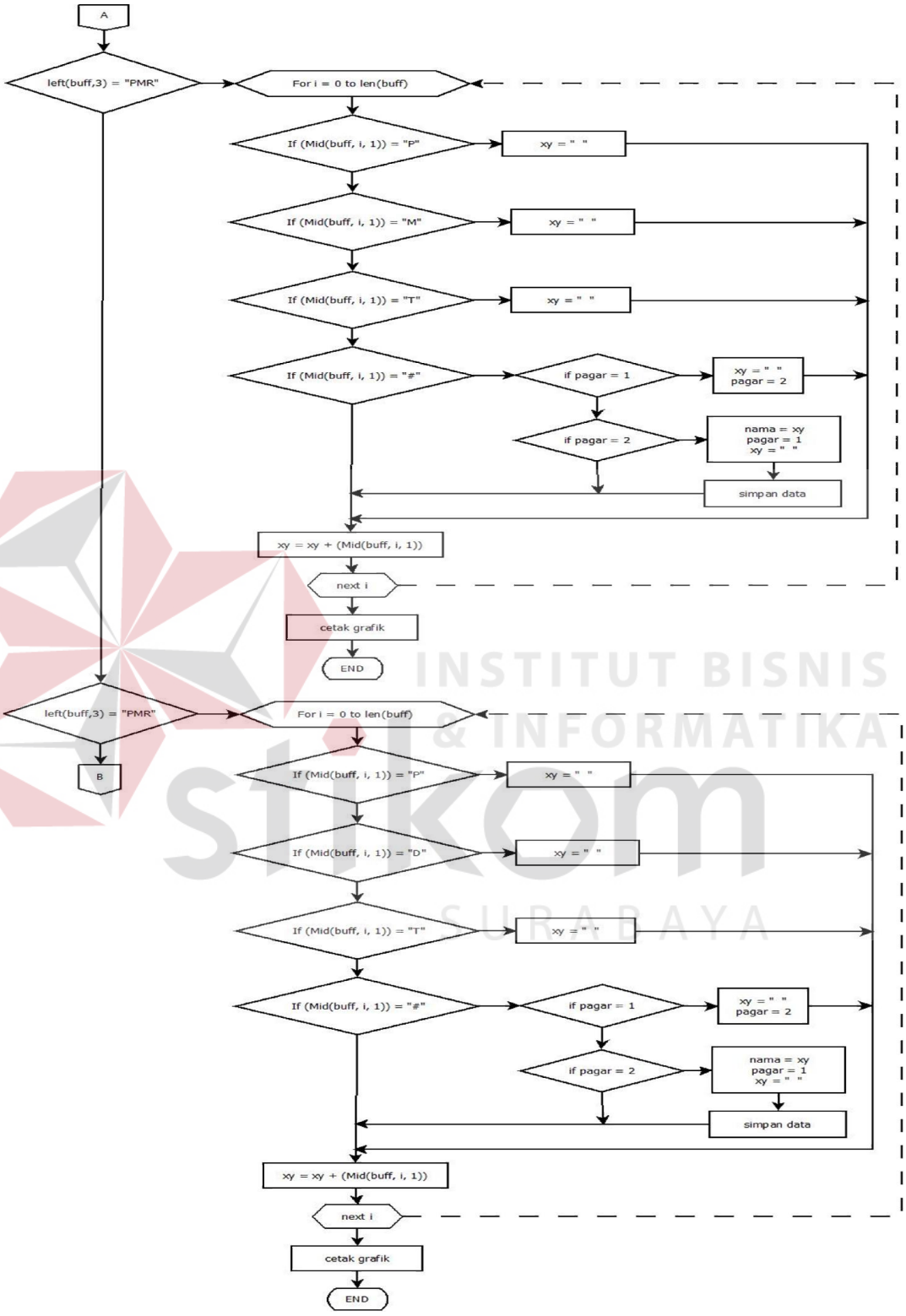
Dalam pengiriman data menggunakan *wifi*, diperlukan modul pendukung, yang dalam hal ini penulis menggunakan CC3000 milik Tony Lab. Untuk itu diperlukan adanya penambahan *library* yang berasal dari modul untuk megakses modul tersebut. Dan, diperlukan adanya pengaturan IP pada alat juga computer yang terhubung dalam sebuah jaringan. Pemberian IP pada alat akan secara otomatis terjadi saat alat terkoneksi pada sebuah jaringan, sama halnya dengan computer yang terkoneksi dalam sebuah jaringan computer. Akan tetapi, diperlukan adanya pengaturan IP tujuan yang dilakukan pada alat agar data yang dikirim diterima oleh computer yang dituju.

3.5.5 Algoritma Penerimaan dan Pemisahan Data pada *server*

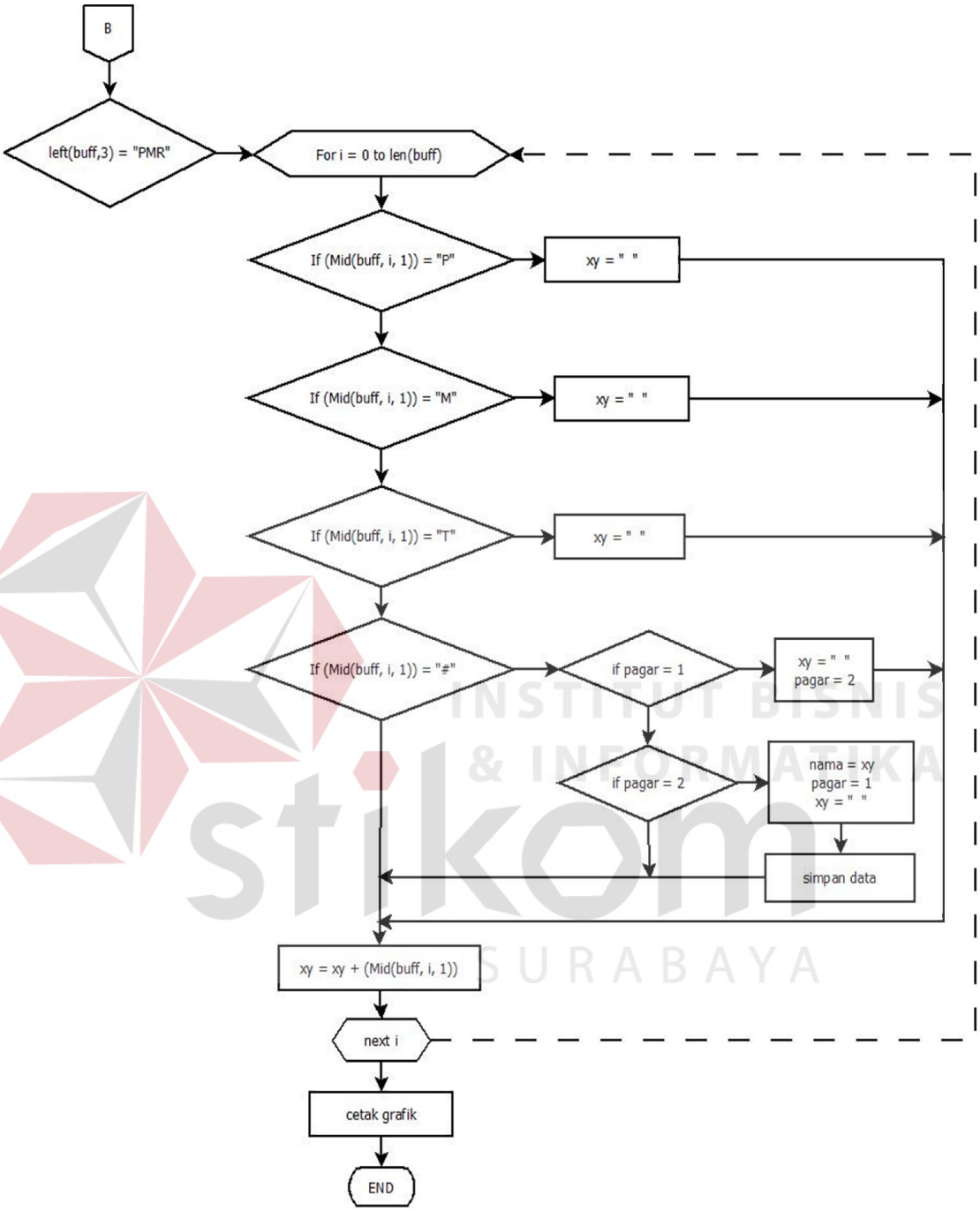
Data yang diterima oleh server masih merupakan data mentah dengan header yang mengikuti data, untuk itu diperlukan adanya pemisahan data agar data dapat diolah kembali menjadi sebuah grafik yang yang terpisah antara CO dan CO_2. Selain itu, fungsi dari pemisahan data itu sendiri adalah untuk penyimpanan data pada setiap file yang telah dikelompokkan. Hasil dari pengelompokan file tersebut adalah seperti gambar 3.17 dibawah ini :



Gambar 3.18a. *Flowchart* penerimaan dan pemisahan data pada server

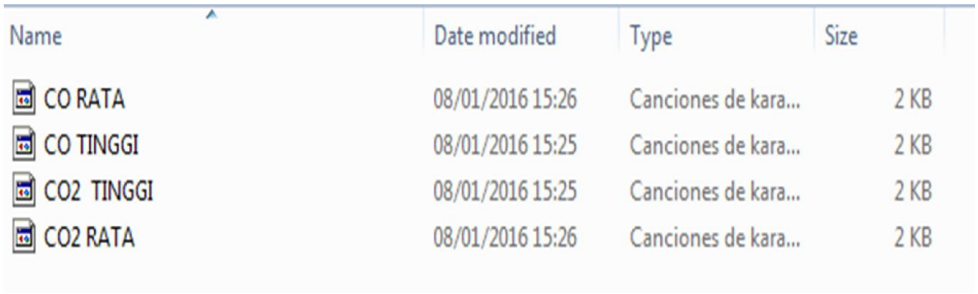


Gambar 3.18b. *Flowchart* penerimaan dan pemisahan data pada server



Gambar 3.18c. *Flowchart* penerimaan dan pemisahan data pada *server*

Pada diagram diatas dijelaskan bahwa data akan tersimpan secara otomatis sesuai dengan kelompok datanya, dalam sebuah file, sepert gambar 3.18 dibawah ini :



Gambar 3.19 Nama file yang tersimpan

3.6 Metode Analisa

Dalam memonitor kualitas udara ini, selain pembuatan algoritma pengiriman data hal terpenting lainnya adalah analisa dari hasil pengiriman itu sendiri agar dapat diketahui seberapa baik sistem yang telah dibangun.

4.6.1 Peletakan Sensor

Dalam memonitoring kualitas udara ini yang terpenting adalah peletakkan masing – masing *Child Node, Cluster Head* dan *Parent Node.* Pada penelitian ini, *Child Node* 1 dan *Child Node* 2 akan diletakkan pada KOPMA atas Institut Bisnis dan Informatika Stikom Surabaya dan *Cluster Head* 1 diletakkan jam menara Institut Bisnis dan Informatika Stikom Surabaya. Sedangkan *Child Node 3* dan *Child Node 4* akan diletakkan pada daerah disekitar gedung Serba Guna Institut Bisnis dan Informatika Stikom Surabaya dan *Cluster Head* 2 diletakkan pada daerah parkir Institut Bisnis dan Informatika Stikom Surabaya. Dan *Parent Node* diletakkan pada Lobi Institut Bisnis dan Informatika Stikom Surabaya. Sedangkan *server* akan

diletakkan di ruang kemahasiswaan Institut Bisnis dan Informatika Stikom Surabaya Denah peletakan masing – masing *node* digambarkan pada gambar 3.1.

4.6.2 Analisa Penerimaan Data pada *server*

Cara menganalisa hasil monitoring udara ini adalah dengan menghitung *delay* dan juga data *loss* yang didapat saat pengiriman data terjadi, dengan menggunakan topologi yang telah dibuat. Langkah – langkah yang yang dilakukan untuk mendapatkan *delay* adalah dengan cara membuat rekaman video saat pengiriman dilakukan, dengan demikian akan diketahui waktu saat data dikirim dan saat data diterima, hal ini penting dilakukan, karena data udara yang diterima merupakan data *streaming,* karena penelitian ini berhubungan dengan pencemaran udara, ketika udara termonitoring buruk, maka ada tindakan – tindakan yang cepat dilakukan.

Sedangkan cara untuk memperoleh data *loss* adalah dengan cara membandingkan data yang telah dikirim dan diterima. Ada beberapa tingkatan yang akan dianalisa pada penelitian ini, yaitu :

1. Data *loss* antara masing – masing *child node* dengan masing – masing *cluster head*
2. Data *loss* antara masing – masing *cluster head* dengan *parent node*
3. Data *loss* antara masing – masing *child node* dengan *parent node*
4. Data *loss* antara *parent node* dengan *server*
5. Data *loss* antara *child node* dengan *server*

Hal ini dilakukan agar kita dapat mengetahui pada bagian mana data terkirim dengan baik dan kurang baik, sehingga nantinya dapat diperbaharui dikemudian hari. Cara memperoleh data *loss* dengan melakukan penghitungan seperti dibawah ini :

$$packet\ loss = \frac{jumlah\ paket\ hilang}{jumlah\ data\ masuk}\ x\ 100\ \%$$

Pemisahan data diperlukan dalam hal ini, karena fungsi dari penambahan *header* sebenarnya hanya untuk memberi identitas asal data. Sehingga perlu dipisahkan untuk memudahkan *user* dalam melihat data pada *end device.*