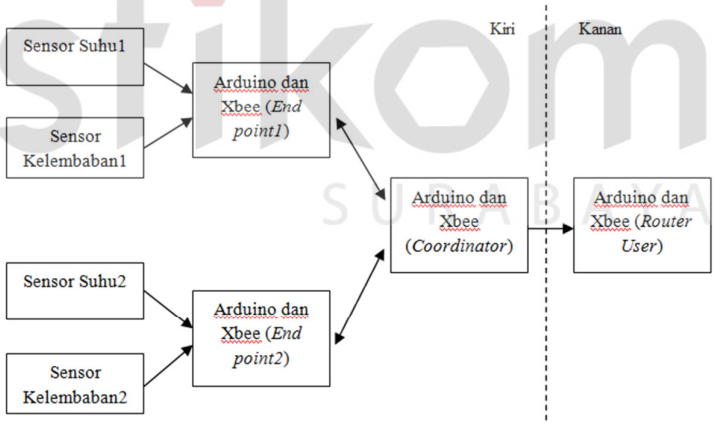
BAB III

METODE PENELITIAN

Metode penelitian yang digunakan pada pembuatan perangkat keras dan perangkat lunak yaitu dengan studi kepustakaan. Dengan cara ini penulis berusaha untuk mendapatkan dan mengumpulkan data-data, informasi, konsep-konsep yang bersifat teoritis dari buku, bahan-bahan kuliah dan internet yang berkaitan dengan permasalahan.

Dari data-data yang diperoleh maka dilakukan perencanaan rangkaian perangkat keras. Dalam perangkat keras ini, penulis akan melakukan pengujian perangkat keras dengan program-program yang telah dibuat. Pembuatan perangkat lunak adalah tahapan selanjutnya. Terakhir adalah perakitan perangkat keras dengan kerja perangkat lunak yang telah selesai dibuat.



Gambar 3.1 Blok Diagram Sebuah Sistem WSN untuk *Monitoring* Suhu dan Kelembaban Tanah pada Lahan Tanaman Jarak.

Pada bab ini dibahas mengenai masalah yang timbul dalam perencanaan dan pembuatan perangkat keras (*hardware*) maupun perangkat lunak (*software)*. Dari

kedua bagian tersebut akan dipadukan / diintegrasikan agar dapat bekerja sama menjalankan sistem dengan baik.

Dalam perancangan sistem *wireless sensor network* untuk *monitoring* suhu dan kelembaban tanah pada lahan tanaman jarak, digunakan blok diagram seperti dalam Gambar 3.1.

Pada tugas akhir ini hanya dibahas sampai dengan *coodinator* (bagian kiri) pada Gambar 3.1 yaitu:

1 *Endpoint*1 yang terdiri dari sensor suhu1 dan sensor kelembaban1 yang terhubung dengan kabel serta arduino dan xbee yang terhubung dengan *shield* xbee.
2 *Endpoint*2 yang terdiri dari sensor suhu2 dan sensor kelembaban2 yang terhubung dengan kabel serta arduino dan xbee yang terhubung dengan *shield* xbee.
3 *Coodinator* yang terdiri dari arduino dan xbee yang terhubung dengan *shield* xbee.

3.1 Perancangan Perangkat Keras

3.1.1 Koneksi sensor pada arduino sebagai *endpoint*

Pada perancangan ini membahas tentang koneksi sensor dengan arduino yang menggunakan kabel sebagai media penghubungnya, dan untuk program pada arduino dapat dilihat pada lampiran. Sensor yang digunakan adalah sensor DHT11 sebagai sensor suhu dan *soil moisture* adalah sebagai sensor kelembaban tanah, sebelum sensor dihubungkan dengan arduino harus menentukan pin dengan benar pada arduino. Supaya dalam pengiriman data dari sensor ke arduino sukses, Pin

input /output pada arduino ada dua tipe yaitu, digital dan analog. Dengan melihat data *sheet* masing-masing sensor, sensor DHT11 memiliki output digital sedangkan sensor *moisture* analog.

Pada Gambar 3.2 ditunjukkan dua buah sensor yang terhubung dengan arduino yaitu sensor DHT11 dan sensor *moisture*, dengan ketentuan:

1. Pin *out* vcc pada sensor DHT11 dengan pin *out* 5v pada arduino.
2. Pin *out* data pada sensor DHT11 dengan pin *in* digital 2 pada arduino.
3. Pin *out* gnd pada sensor DHT11 dengan pin *in* gnd pada arduino.
4. Pin *out* vcc pada sensor *moisture* dengan pin *out* 3.3v pada arduino.
5. Pin *out* data pada sensor *moisture* dengan pin *in* analog 3 pada arduino.
6. Pin *out* gnd pada sensor *moisture* dengan pin *in* gnd pada arduino.

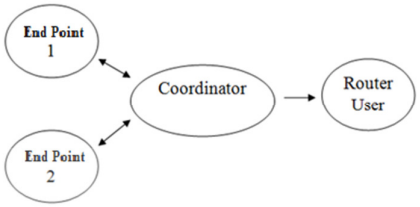
Gambar 3.2 Sensor yang Terhubung dengan Arduino.

3.2 Perancangan Perangkat Lunak

Selain perancangan *hardware* yang diperlukan pada perancangan sistem *wireless sensor network* untuk *monitoring* suhu dan kelembaban tanah pada lahan tanaman jarak meliputi algoritma dan program pada IDE arduino beserta *flowchart* yang menjelaskan alur proses program tersebut.

3.2.1 Program xbee sebagai *endpoint* dan *coodinator*

Sebelum menetukan program masing-masing xbee harus menetukan *toplogy* jaringannya, sehingga lebih mudah untuk proses konfigurasi xbee pada peranan masing-masing dalam protokol komunikasi atau pengalamatan yang ditujukan. Untuk pilihan *topology* menggunakan *topology tree*, karena dibutuhkan untuk komunikasi *point to point* dan *point to multipoint.*



Gambar 3.3 Diagram WSN.

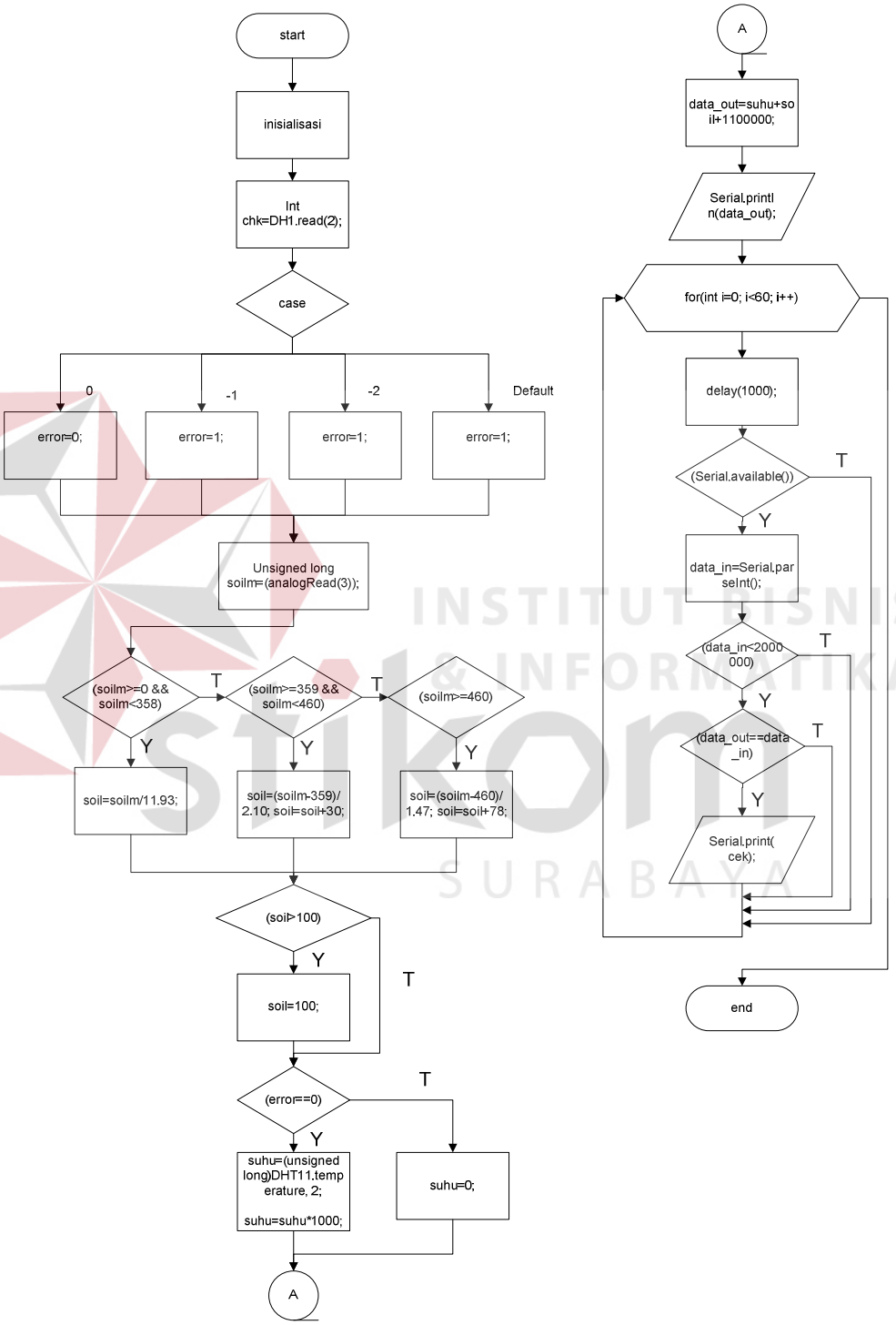
Pada Gambar 3.3 ditunjukan sebuah simbol panah dua arah, maksudnya *node endpoint1* maupun *node endpoint2* dapat menerima dan mengirim data tetapi hanya komunikasi *point to point.* Sedangkan pada *node coodinator* dapat menerima sekaligus mengirim data dan berkomunikasi *point to multipoint* pada *node* yang terhubung langsung yang disebut dengan *broadcast*. Untuk komunikasi antara *node coodinator* dengan *node* router *user, node coodinator* dapat mengirim data dan tidak dapat menerima data dari *node endpoint* user. Begitu juga untuk *node router user* tidak dapat mengirim data pada *node coodinator* dan hanya dapat menerima data dari *node coodinator* saja. Dan untuk konfigurasi xbee dapat dilihat pada lampiran.

3.2.2 Program arduino sebagai *endpoint* 1 dan *endpoint* 2

Pada program arduino sebagai *endpoint*1 maupun *endpoint*2, maka dapat dilihat *flowchart* pada Gambar 3.4 dan 3.5:

1. *Flowchart endpoint*1, pada label inisialisasi berisikan:

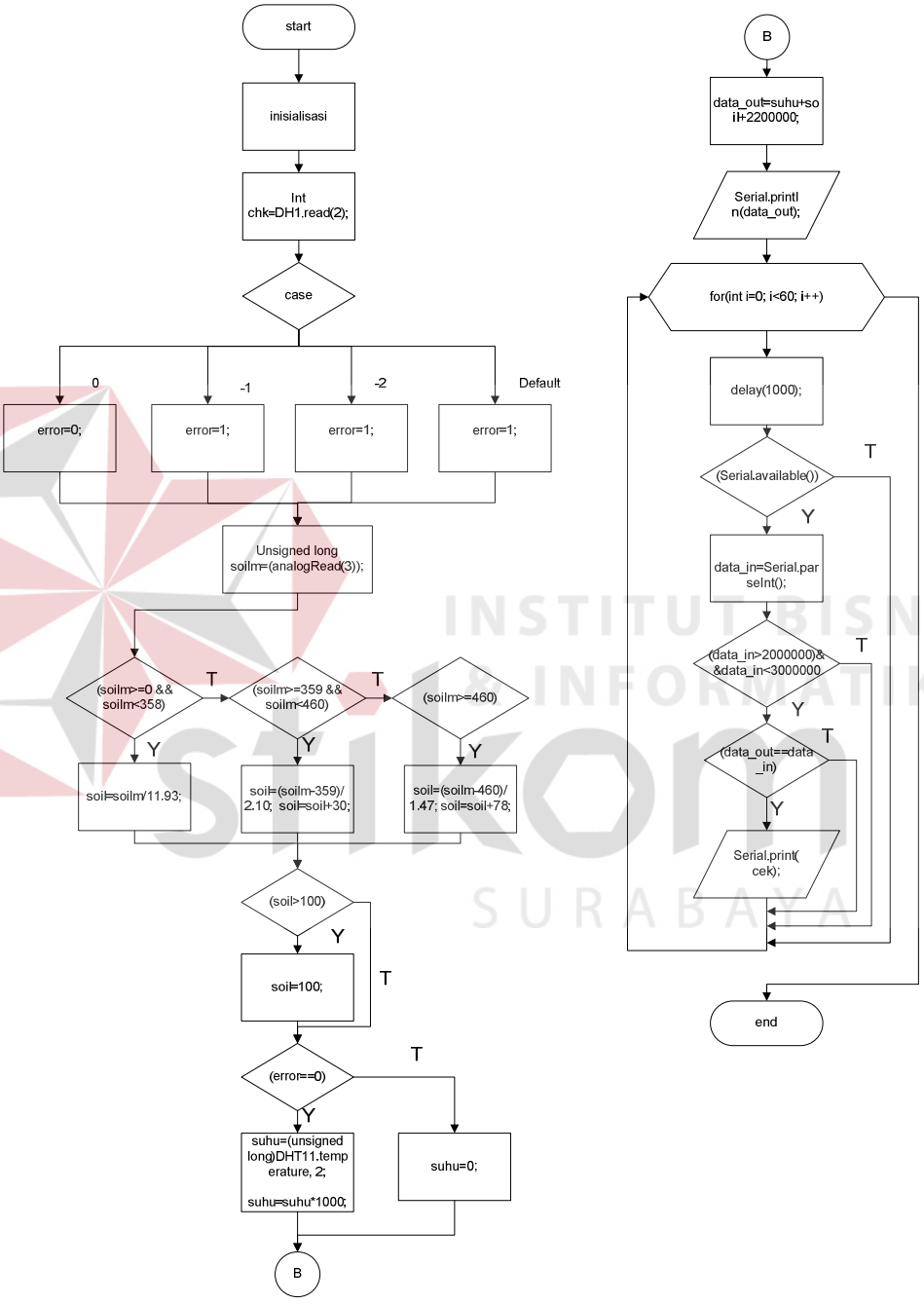
```
unsigned long soil, data_out, suhu, data_in, cek=11,
error;
```



Gambar 3.4 *Flowchart Endpoint*1.

2. *Flowchart endpoint*2 pada label inisialisasi berisikan:

```
unsigned long soil, data_out, suhu, data_in,
cek=22, error;
```



Gambar 3.5 *Flowchart Endpoint*2.

Pada Gambar 3.4 dan 3.5 terdiri tiga bagian utama yaitu:

a. Inisialisasi.

Pada dasarnya alur dari *endpoint*1 dengan *endpoint*2 sama hanya terdapat perbedaan kode *id endpoint* masing-masing dan logika program penerimaan masing-masing. Maka dalam pemrograman *endpoint*1 ini diberi kode *id* 11 sedangkan *endpoint*2 adalah 22. Pada pemrograman ini diasumsikan pada *endpoint*1 yaitu dimulai dari pembacaan sensor DHT11 dan sensor *moisture,* lalu data dari setiap sensor disimpan didalam *variable* masing-masing yaitu variabel suhu untuk *output* data dari sensor DHT11 dan variabel soil untuk *output* data dari sensor *moisture.* Selanjutnya dilakukan pengecekan status dari sensor DHT11 apakah sensor tidak ada *error* misalnya ada kabel yang tidak terhubung dengan baik. Apabila sensor tersebut terdapat *error* maka sensor akan memberikan informasi bahwa *error*=1, sedangkan jika tidak terdapat error maka *error*=0. Pada program ditulis sepert ini:

```
int chk = DHT11.read(2);
  switch (chk)
  {
    case 0: error=0; break;
    case -1: error=1; break;
    case -2: error=1; break;
    default: error=1; break;
  }
```

Kemudian pembacaan sensor dari sensor *moisture* dan hasil *output* sensor disimpan ke dalam *variabel soil*, supaya nilai dari *soil* menjadi satuan %RH dan terkalibrasi dengan alat ukur yang ditetapkan. Pada sensor *moisture* ini memiliki rumus 3 macam kondisi, yaitu kering, lembab dan basah. Begitu juga pada alat ukur kelembaban tanah juga menunjukkan 3 macam tipe kondisi yang dapat diukur, pada alat ukur kondisi kering

menunjukkan nilai 0-30%RH, kemudian pada kondisi lembab menunjukkan nilai 31-79%RH dan pada kondisi basah menunjukkan 80-100%RH. Sedangkan pada sensor kondisi kering menunjukkan 0-358, lembab 359-460 dan basah 461-495. Dari data yang diperoleh. maka dapat menggunakan rumus perbagian untuk mencari nilai kelembaban 1%RH pada *output* dari sensor.

$$1\%RH = \frac{\text{nilai max sensor} - \text{nilai min sensor}}{\text{nilai max alat ukur} - \text{nilai min alat ukur}} \quad (3.1)$$

Misalnya:

1. Kondisi kering

   $$1\%RH = \frac{358-0}{30-0} \rightarrow 1\%RH = 11.93 \quad (3.2)$$

   Maka untuk dapat persentase kelembaban yang terukur oleh sensor pada kondisi kering adalah sebagai berikut:

   $$\text{Kelembaban}(\%RH) = \frac{\text{Pengukuran Sensor}}{11.93} \quad (3.3)$$

2. Kondisi lembab

   $$1\%RH == \frac{460-359}{79-31} \rightarrow 1\%RH = 2.10 \quad (3.4)$$

   Maka untuk dapat persentase kelembaban yang terukur oleh sensor pada kondisi lembab adalah sebagai berikut:

   $$\text{Kelembaban}(\%RH) = \frac{\text{Pengukuran Sensor}}{2.10} \quad (3.5)$$

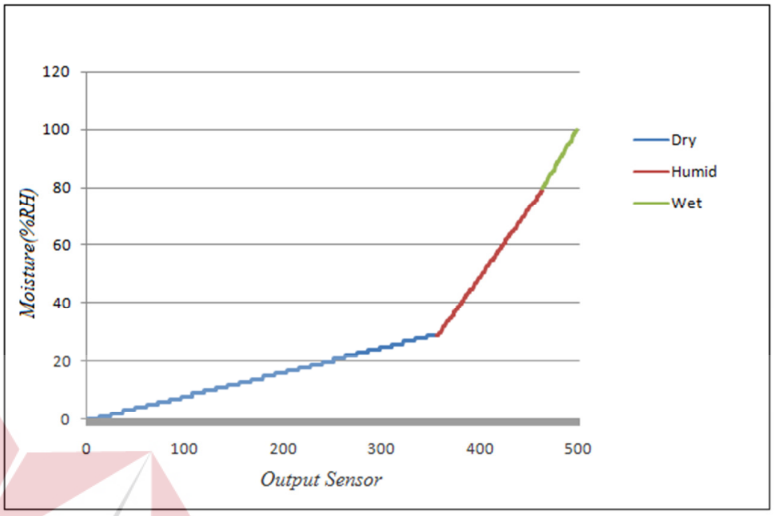
3. Kondisi basah

   $$1\%RH == \frac{495-461}{100-80} \rightarrow 1\%RH = 1.7 \quad (3.6)$$

   Maka untuk dapat persentase kelembaban yang terukur oleh sensor pada kondisi basah adalah sebagai berikut:

   $$\text{Kelembaban}(\%RH) = \frac{\text{Pengukuran Sensor}}{1.7} \quad (3.7)$$

Secara keseluruhan karakteristik persentase kelembaban terhadap pengukuran sensor berdasarkan persamaan 3.3, 3.5 dan 3.7 ditunjukkan dengan grafik seperti pada Gambar 3.6.



Gambar 3.6 Grafik Karakteristik Sensor Kelembaban

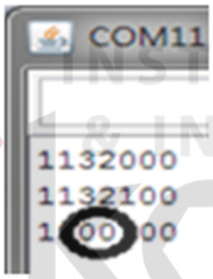
Untuk potongan *source code* pada *endpoint*1/*endpoint*2, mulai dari pembacaan nilai dari sensor sampai dengan perhitungan untuk kalibrasi dengan alat ukur kelembaban yang sudah ditentukan yaitu:

```
unsigned long soilm=(analogRead(3));
   if(soilm>=0 && soilm<=358)
   {
   soil=soilm/11.93;}
   else if(soilm>=359 && soilm<=460)
   {
   soil=(soilm-359)/2.10;
   soil=soil+31;
   }
     else if(soilm>=461)
   {
   soil=(soilm-461)/1.7;
   soil=soil+80;
   }
```

Karena sensor *moisture* adalah sensor analog maka kemungkinan data kurang stabil, kejadian muncul nilai 101 akan akan terjadi, maka apabila

terjadi demikian dibuat sebagai antipasipasi jika muncul nilai >100 maka nilai harus muncul 100. Kemudian lanjut ke proses pembacaan sensor DHT11, akan tetapi sebelum itu harus melihat nilai *error* dari pengecekan sensor pada awal program. Apabila nilai *error*==0 maka proses pembacaan dimulai, sedangkan nilai *error*==1 maka nilai suhu=0. Untuk potongan *source code* seperti dibawah ini:

```
if (error==0)
   {
    suhu=(unsigned long)DHT11.temperature, 2;
    suhu=suhu*1000;
   }
   else
   {
    suhu=0;
   }
```



Gambar 3.7 *Output* Sensor Suhu Saat Terdapat *Error*.

b. Pengiriman data.

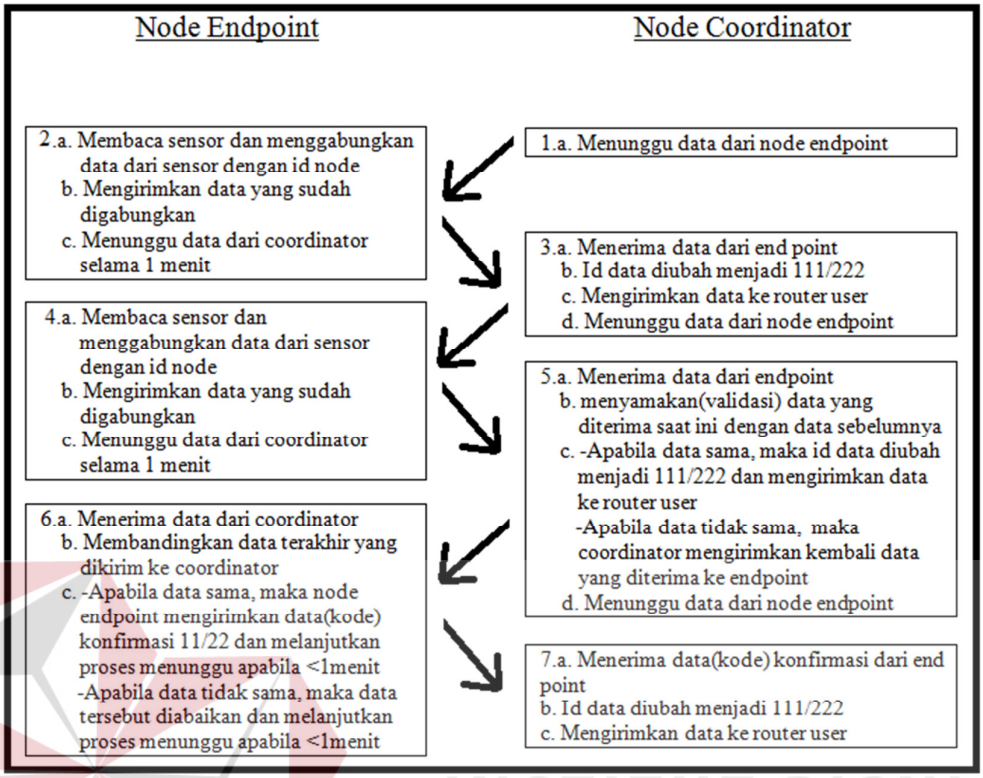
Dalam suatu jaringan selalu memiliki perbedaan *id node* masing-masing. Selanjutnya nilai dari *id* digabungkan/encapsulasi dengan nilai-nilai sensor masing-masing. Misanya pada *endpoint*1 nilai suhu=23 dan kelembaban=100 maka data yang akan dikirimkan kepada *node* yang dituju (*coodinator*) adalah 1123100, data ini berasal dari nilai *id* dikalikan 1100000 kemudian dijumlahkan dengan nilai dari suhu dikalikan 1000 dan terakhir dijumlahkan dengan nilai kelembaban.

Gambar 3.8 Contoh *Endpoint* Menunjukkan Validasi.

c. Validasi data.

Setelah data dikirimkan ke *coodinator*, *endpoint*1 menunggu selama 60 detik untuk menunggu kabar dari *coodinator*. Apabila *coodinator* tidak mengirim data ke *endpoint*1 selama proses menunggu selesai *endpoint* akan membaca data dari sensor kembali, tetapi apabila *coodinator* mengirim data kembali ke *endpoint*1 berarti terjadi perubahan data. Maksudnya perubahan data yaitu data yang diterima oleh *coordinator* berbeda dengan data yang sebelumnya(contohnya pada Gambar 3.8 data yang diterima oleh *coordinator* adalah 1131071 selanjutnya *coordinator* menerima data 1131072). Data yang telah dikirimkan ke *endpoint*1 akan dicek lagi oleh *endpoint*1 apakah data benar atau sama dengan nilai data terakhir yang telah dikirimkan ke *coodinator* dari *node endpoint.* Apabila data sama, maka *endpoint*1 akan mengirim kode(konfirmasi) *id* saja ke *coodinator* yang seperti ditunjukkan Gambar 3.8.

Pada Gambar 3.9 menjelaskan keseluruhan proses pengiriman data atau protokol komunikasi, dimulai dari pembacaan sensor kemudian digabung dengan *id node* masing-masing sampai dengan pengiriman data ke *router user.*

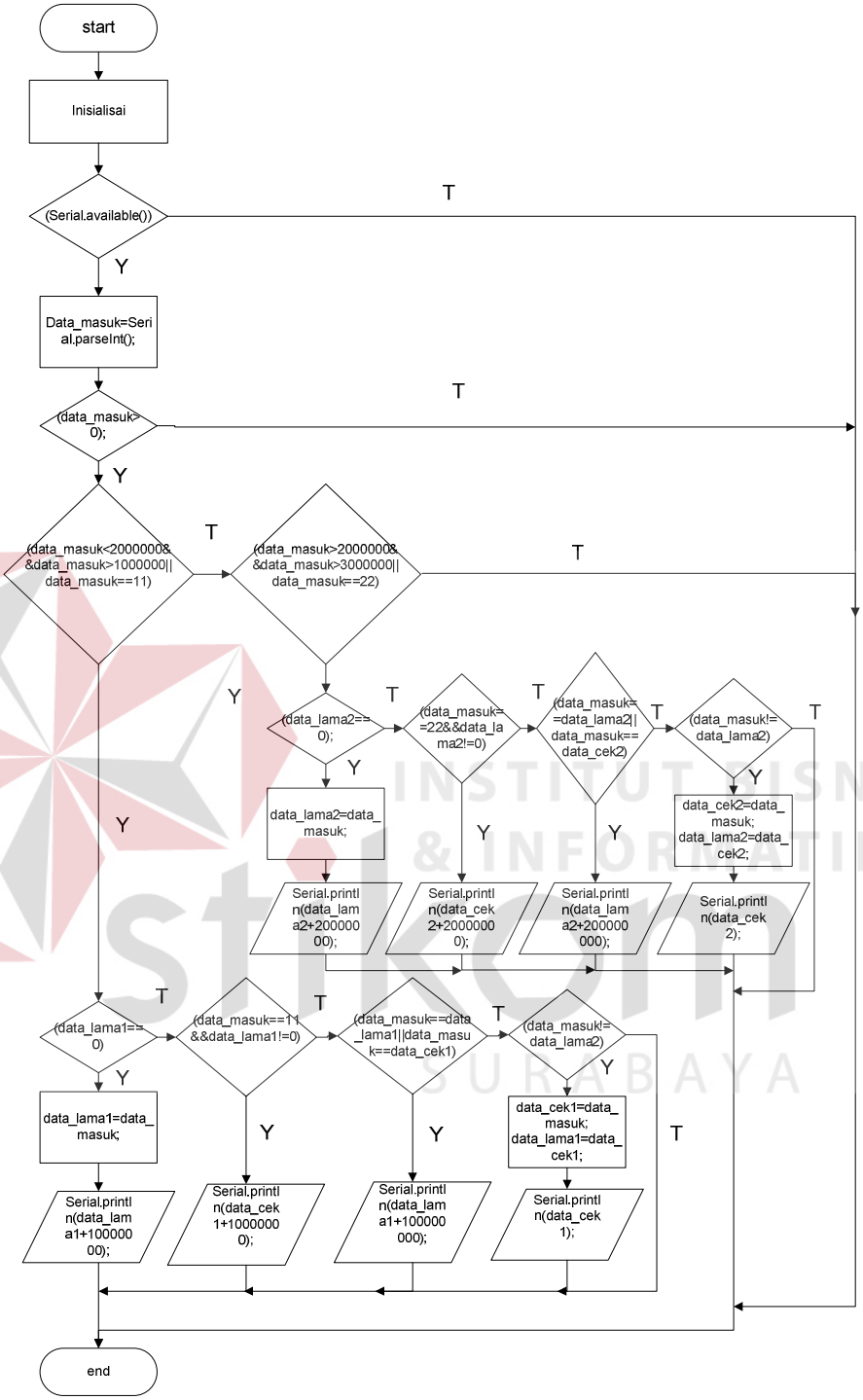


Gambar 3.9 Proses Pengiriman Data

3.2.3 Program arduino sebagai *coodinator*

Coodinator sebenarnya hanya sebagai pengantar data ke alamat yang dituju, tetapi disini *coodinator* pengiriman bersifat broadcast sehingga data sebenarnya dikirim ke semua *node* yang terhubung langsung. Tetapi *node* yang menerima data dari *coodinator* mengerti apakah data itu untuk *node* tersebut, jadi kesimpulannya *coodinator* mengirim data secara *broadcast* tetapi data yang terkirim selalu diterima pada *node* yang ditujukan. Untuk alur program pada *coodinator* dapat dilihat *flowchart* dalam Gambar 3.10, pada bagian inisialisasi berisikan:

```
unsigned long data_lama1, data_lama2=0, data_masuk,
data_cek1, data_cek2;
```



Gambar 3.10 *Flowchart Coodinator*.

Program arduino sebagai *coodinator* dimulai dari menerima data awal dari masing-masing *endpoint*, karena sebagai data awal maka *coodinator* melanjutkan

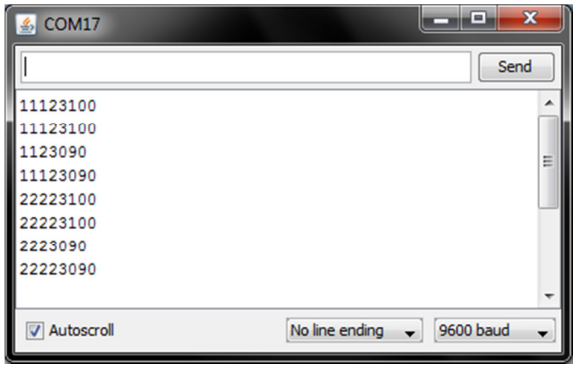
data dengan mengirim ke *endpoint* user, tetapi sebelum mengirim data tersebut *coodinator* menambahkan *id* dari data tersebut. Misalnya data yang diterima adalah 1123100, berarti data tersebut dari *endpoint*1 karena digit awal adalah 11 sebagai *id* dari *endpoint*1. Berikut potongan *source code* pada penerimaan data dari *endpoint*1 :

```
if(data_masuk>0)
   {
      if(data_masuk<2000000 && data_masuk>1000000
 || data_masuk==11)//masuk dari data node1

 {
        if(data_lama1==0)
        {
           data_lama1=data_masuk;
           Serial.println(data_lama1 + 10000000);
        }
        else if(data_masuk==11 && data_lama1!=0)
        {
         Serial.println(data_cek1+10000000);
        }

    else if(data_masuk==data_lama1 ||
data_masuk==data_cek1)
        {
          Serial.println(data_lama1 + 10000000);
        }
        else if(data_masuk!=data_lama1)
        {
          data_cek1=data_masuk;
          data_lama1=data_cek1;
          Serial.println(data_cek1);
        }
     }
```

Kemudian *coodinator* merubah *id* tersebut sebagai penandaan bahwa data ini sudah di koreksi *coodinator* dan dikirim ke *router user* menjadi 111xxxxx, yang berasal dari data yang diterima di tambah dengan nilai 10000000. Maka data yang dikirm ke *router user* adalah 11123100.



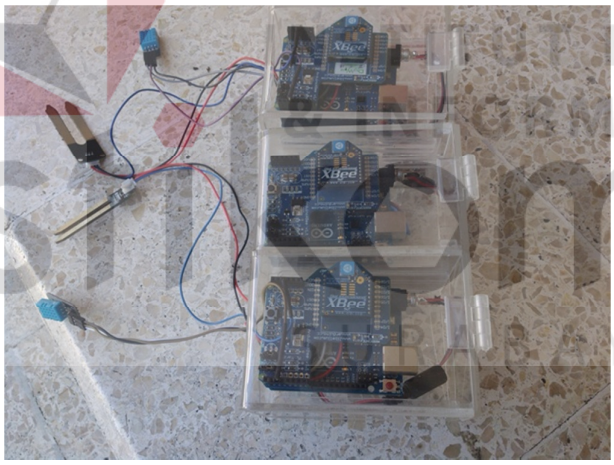
Gambar 3.11 Contoh Data yang Sudah di Cek oleh *Coodinator*.

Selanjutnya apabila sudah terdapat nilai awal yang tersimpan pada variabel data_lama, maka *coodinator* pada proses selanjutnya yaitu menerima data lagi dari *endpoint*1 ataupun *endpoint*2. Pada saat menerima data *coodinator* akan memilah-milah data misalnya data yang diterima <2000000 dan data yang diterima >1000000 , maka *coodinator* akan memproses bahwa data ini dari *endpoint*1. Pada Gambar 3.11 terlihat bahwa *coodinator* mengirim data 11123100, data ini berasal dari *endpoint*1 dengan data 1123100 saat *coodinator* menerimanya. Saat pertama kali menerima *coodinator* akan langsung mengirim ke *router user* karena masih belum ada nilai sebelumnya yang akan dibandingkan dan merubah *id* terlebih dahulu. Selanjutnya *coodinator* mengirimkan data yang sama pada *router user* karena *coodinator* menerima data yang sama dari *endpoint*1 yaitu 1123100. Tetapi pada saat *coordinator* menerima data 1123090, maka *coodinator* mengirim dengan nilai 1123090 ke *endpoint*1 karena terdapat perbedaan data dari data sebelumnya saat *coodinator* menerima data ke *endpoint*1. Dan apabila *coodinator* menerima data dengan nilai 11 maka *coodinator* akan mengirim data yang sebelumnya sudah dikirim ke *endpoint*1, tetapi *coodinator* merubah *id* supaya *router user* mengerti bahwa data tersebut untuk *router user*

yaitu menjadi 11123090 (8digit). Proses ini tidak ada perbedaan saat menerima data dari *endpoint*1.

3.3 Perakitan

Setelah melakukan percobaan koneksi sensor dengan arduino serta mengkonfigurasi xbee untuk peranan/ parameter masing-masing yang dibutuhkan pada setiap *node* yaitu *endpoint*1, *endpoint*2 dan *coodinator*. Saatnya menggabungkan antara perangkat keras dengan perangkat lunak yaitu xbee yang sudah dikonfigurasi dan arduino dengan program *endpoint*1/*endpoint*2/*coodinator* beserta sensor-sensor yang sudah terhubung sehingga menjadi terlihat dalam Gambar 3.12.



Gambar 3.12 Hasil Perakitan Setiap *Node*.