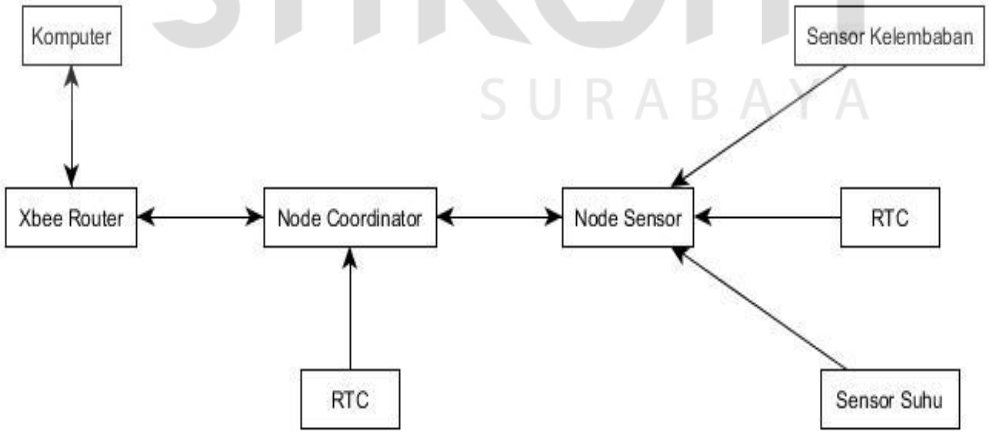
# BAB III

# METODE PENELITIAN

## 3.1. Metode Penelitian

Metode Penelitian yang digunakan pada pembuatan perangkat keras dan perangkat lunak yaitu dengan studi pustaka. Dengan cara ini penulis berusaha mendapatkan dan mengumpulkan data-data, informasi, dan konsep-konsep bersifat teori dari buku, dan bahan-bahan materi kuliah dan juga internet yang berkaitan dengan penelitian.

Dari data-data yang diperoleh maka disusun perancangan rangkaian perangkat keras. Dalam pengujian perangkat keras ini dilakukan pengujian yang didukung dengan program yang telah dibuat. Selanjutnya tahapan pembuatan perangkat lunak. Terakhir adalah perakitan perangkat keras dengan kerja perangkat lunak yang telah selesai dibuat.




Gambar 3.1 Blok diagram

Dari Gambar 3.1 menggambarkan blok diagram sistem, yang terdiri dari 4 buah *node* dan satu komputer yang berfungsi sebagai *monitoring* suhu dan kelembaban tanaman jarak. untuk *node router* terdiri dari *xbee* yang langsung

terhubung dengan komputer , fungsi dari *node router* adalah sebagai *monitoring* data yang dikirim dari *node* sensor. *Node coordinator* terdiri dari Mikrokontroler yang terhubung dengan *xbee* dan RTC, *coordinator* berperan sebagai jembatan untuk meneruskan paket data dari *node* sensor. *Node* sensor terdiri dari Mikrokontroler yang terhubung dengan *xbee* serta beberapa inputan sensor dan RTC. Fungsi keduanya untuk mengirimkan hasil sensor suhu dan kelembaban dan waktu menuju *node router* melalui perantara *node coordinator*.

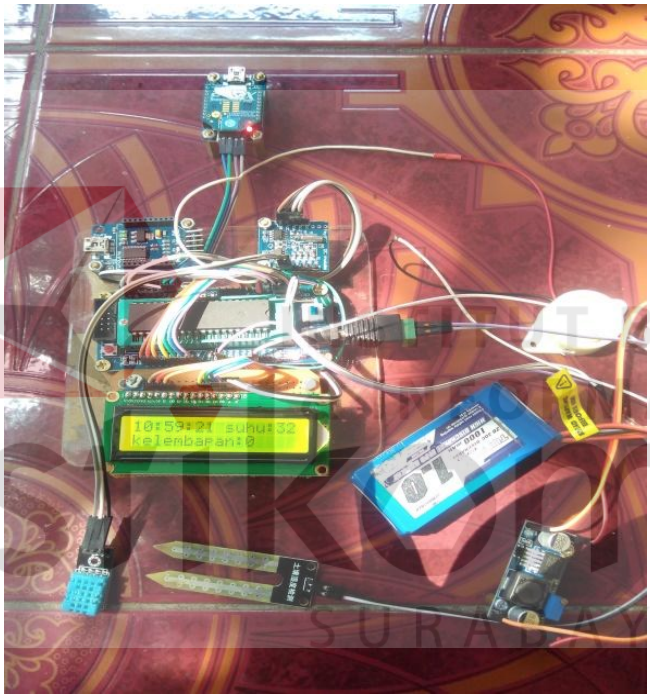
## 3.2 Perancangan Perangkat Keras

### 3.2.1 Koneksi sensor dengan mikrokontroler sebagai node sensor

Pada perancangan ini membahas koneksi sensor dengan mikrokontroler yang menggunakan kabel sebagai penghubung, dan untuk program dapat dilihat pada lampiran. Sensor yang digunakan adalah *soil moisture* sebagai sensor kelembaban tanah dan DHT11 sebagai sensor suhu, sebagai penunjuk waktu menggunakan RTC. Sebelum sensor terhubung pada mikrokontroler harus menentukan *pin* dengan benar pada mikrokontroler. Agar dalam pengiriman data dari sensor dapat terdeteksi dengan baik. Pada Gambar 3.2 ditunjukan ada dua buah sensor yang terhubung dengan mikrokontroler dan RTC, dengan ketentuan :

1. *Pin out* VCC pada sensor kelembaban tanah dengan *pin* 5V pada mikrokontroler.
2. *Pin out* GND pada sensor kelembaban tanah dengan *pin* GND pada mikrokontroler.
3. *Pin out* data pada sensor kelembaban tanah dengan *pin* A7 pada mikrokontroler.

4. *Pin out* VCC pada sensor suhu dengan *pin out* 5V pada mikrokontroler.
5. *Pin out* GND pada sensor suhu dengan *pin* GND pada mikrokontroler.
6. *Pin out* data pada sensor suhu tanah pada *pin* B0 pada mikrokontroler.
7. *Pin out* VCC pada RTC dengan *pin* 5V mikrokontroler.
8. *Pin out* GND pada RTC dengan *pin* GND mikrokontroler.
9. *Pin out* SDA pada RTC dengan *pin* D6 mikrokontroler.
10. *Pin out* SCL pada RTC dengan *pin* D7 mikrokontroler.



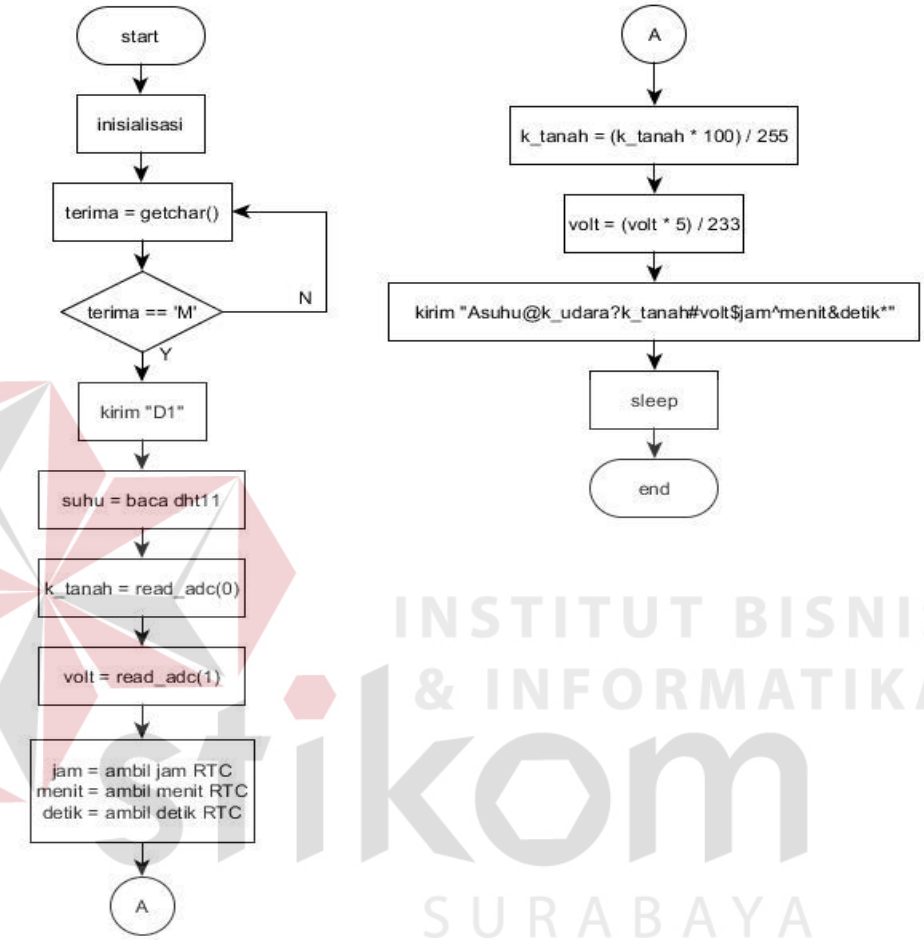

Gambar 3.2 Perancangan Perangkat keras *Node Sensor*

### 3.3 Perancangan Perangkat Lunak

Selain Perancangan perangkat keras juga diperlukan perancangan perangkat lunak agar sisem berjalan dengan baik, perancangan perangkat lunak meliputi algoritma dan program pada mikrokontroler, beserta *flowchart* yang berfungsi untuk menjelaskan jalannya program.

### 3.3.1 Perancangan Mikrokontroler sebagai *Node* sensor

Proses mikrokontroler sebagai *node* sensor dapat dilihat pada *flowchart* Gambar 3.3 :




Gambar 3.3 *Flowchart node* sensor

Pada *flowchart node* sensor terdapat 3 pokok proses utama yaitu :

1. Inisialisasi

Pada proses inisialisasi mikrokontroler terlebih dahulu membaca *port input* apa saja yang terhubung, dan pengolahan data pada setiap masing masing sensor. Untuk program pengolahan data masing masing sensor dapat dilihat pada lembar lampiran. Kemudian mikrokontroler siap untuk menerima data yang dikirim dari *node coordinator*, pada pemrogrman *coordinator* mengirim karakter

"M", yang artinya *node* sensor harus mengirimkan data yang diminta oleh *node coordinator* yang berisi sensor suhu, kelembaban, dan waktu. Berikut adalah cuplikan program ketika *node* sensor menerima karakter "M" :

```
if(data == 'M')
  {
    sleep_disable();
    PORTB.1 = 1;
    printf("D1");
    delay_us(10);
    baca_dht();
    suhu = suhu + 3;
    data_tmp = 255 - read_adc(7);                  //baca sensor SM
    k_tanah = (data_tmp*100)/255;                  //konversi presentase
    data_baterai = (float)(read_adc(2)*5.00)/233.00; //baca sensor baterai
    rtc_get_time(&jam,&menit,&detik);              //baca rtc
    data_tmp = (int)abs(data_baterai*100);
  }
```

2. Pengiriman data

Dalam pengiriman data sudah dikonsep format pengirman data, proses pengiriman data diawali dengan *header* dan diakhiri dengan *trailer*. *Header* data diawali dengan karakter "A" yang berasal dari *node* sensor kemudian diikuti dengna data suhu, kelembaban tanah dan penunjuk waktu. Setelah data telah terambil secara urutannya sesuai dengan format pengiriman, akan dikhiri dengan *trailer* berupa karakter "*" . Perlu diketahui untuk setiap urutan data yang telah tersusun harus ada pemisah berupa karakter antara data satu dengan selanjutnya. Berikut adalah format pengiriman data beserta karakter pemisah antar data, yang ditunjukan pada Gambar 3.4 :


Gambar 3.4 Format pengiriman data

Berikut adalah penjelasan dari gambar :

1. “A” : digunakan sebagai *Header* data.
2. “@” : Penanda karakter data suhu.
3. “#” : Penanda karakter data K_tanah.
4. “$” : Penanda karakter data Volt.
5. “^” : Penanda karakter data jam.
6. “&” : Penanda karakter data menit.
7. “*” : Penanda karakter data detik.

Data-data tersebut dikirim secara bersamaan pada saat *node coordinator* meminta data, berikut adalah cuplikan program *node* sensor mengirim data :

```
printf("A%d@%d?%d#%d$%d^%d&%d*", suhu, k_tanah, k_udara, data_tmp, jam, menit, detik);
```

*3. Sleep mode*

Setelah mikrokontroler mengirimkan seluruh data yang diinginkan oleh *router* atau *end user*, mikrokontroler akan kembali pada kondisi *sleep mode* guna menghemat daya pada *node* sensor. berikut adalah cuplikan program *sleep* pada mikrokontroler :

```
interrupt [TIM2_OVF] void timer2_ovf_isr(void)
{
   sleep_enable();
   idle();
}
```

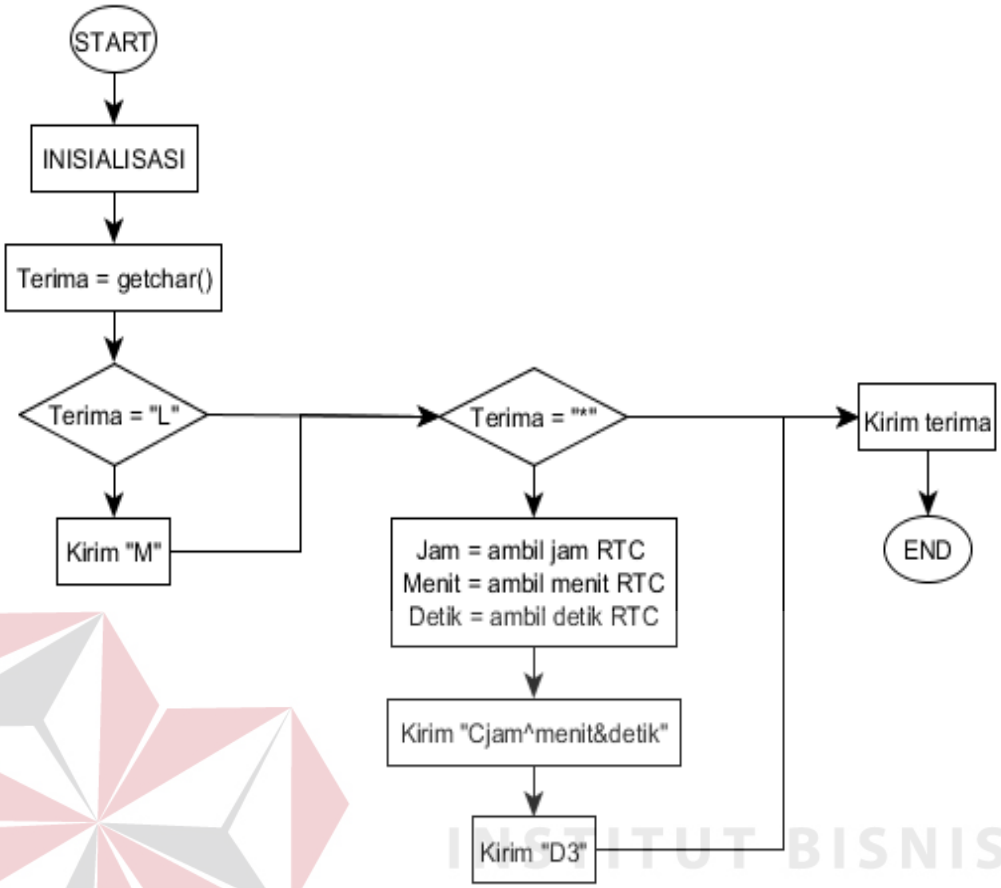
4. Sinkronisasi Waktu

Sinkronisasi waktu dilakukan untuk mengatur waktu pada *node* sensor agar sinkron dengan waktu yang ada pada aplikasi. Jadi data yang diambil menunjukan data *realtime*. Untuk mengatur waktu pada *node* sensor masih dilakukan dengan cara manual, dengan mengatur RTC yang ada pada *node* sensor, RTC tetap berjalan meskipun dalam kondisi *sleep mode*, dikarenakan pada modul RTC memiliki sumber tegangan sendiri. Jadi ketika aplikasi meminta data, secara otomatis *node* sensor juga mencantumkan waktu pada saat pengambilan data dan dikirimkan dalam satu paket. Berikut adalah potongan program untuk mengatur waktu pada *node* sensor.

```
void set_rtc1()
{
        rtc_get_time(&jam,&menit,&detik);
        if(jam == 0)
        {
                rtc_set_time(5,30,0);       //jam:menit:detik
                rtc_set_date(3,17,7,2016);  //minggu,hari,bulan,tahun
        }
}
```

### 3.3.2 Perancangan Mikrokontroler sebagai *Node Coordinator*

Proses mikrokontroler sebagai *node coordinator* dapat dilihat pada *flowchart* Gambar 3.5 :




Gambar 3.5 *Flowchart Node Coordinator*

Dalam Gambar 3.5 sebagai *flowchart* proses *node coordinator* terdapat 2 proses utama yaitu :

1. Menerima Perintah.

Pada proses menerima perintah *node coordinator* mendapat perintah dari *node router* untuk mengambil data pada *node* sensor, kemudian perintah tersebut diteruskan oleh *node coordinator* kepada *node* sensor. Dari *flowchart* di atas ditunjukan pada Terima = “L”, jika *node coordinator* menerima karakter “L” yang berasal dari *node router*, maka *node coordinator* harus meneruskan perintah tersebut ke *node* sensor dengan mengirimkan perintah yang ditandai dengan karakter “M”. Setelah karakter “M” diterima oleh *node* sensor kemudian diproses

oleh *node* sensor yang mana proses tersebut telah dijelaskan pada penjelasan sebelumnya.

2. Pengiriman Data.

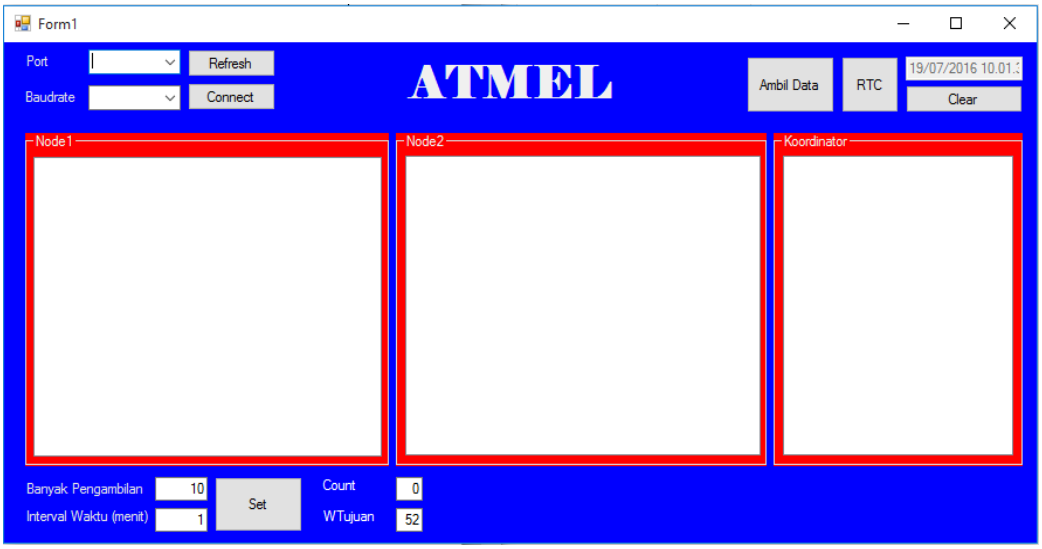
Proses pengiriman data *node coordinator* bertugas untuk meneruskan paket yang dikirim dari *node sensor* menuju *node router*. Setelah tugas meneruskan data dari *node* sensor selesai kemudian *node coordinator* mengirimkan paket data yang berisi jam, menit dan detik yang berasal dari *node coordinator* sendiri dan di akhiri dengan mengirim karakter "D3", yang menandakan bahwa *node* tersebut aktif.

3. Sinkronisasi waktu

Pada *node coordinator* juga dilakukan proses sinkronisasi waktu, supaya data yang akan diinformasikan pada aplikasi juga secara *realtime*. Untuk mengatur waktu pada *node* coordinator sama seperti penjelasan pada pengaturan waktu *node* sensor.

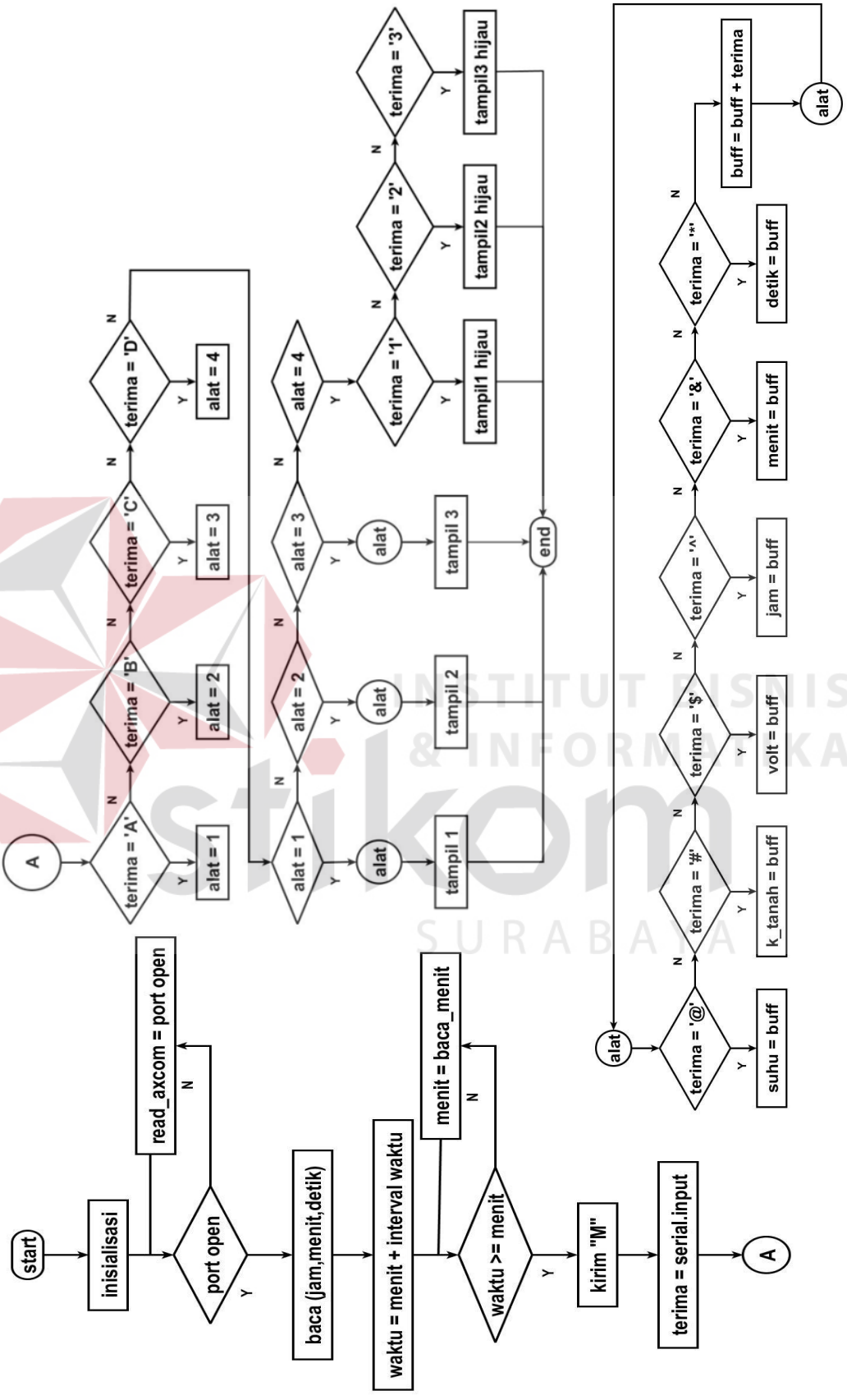
### 3.3.3 Perancangan *Visual Basic*

Aplikasi *visual basic* digunaka pada komputer sebagai *end device* yang berfungsi untuk *monitoring* data yang telah dikirm dari *node* sensor. berikut adalah tampilan aplikasi *monitoring* :




Gambar 3.6 Aplikasi monitoring

Dari Gambar 3.6 aplikasi *monitoring* terdapat fungsi masing masing bagian. Untuk kolom port berfungsi sebagai pilihan *com* USB *xbee* adapter yang terhubung pada komputer atau *end device*. Kolom *baudrate* berfungsi untuk memilih *baudrate* dan nilainya harus sama pada settingan yang digunakan pada *xbee*. pada kolom *text node* 1, *node* 2, dan *coordinator* berfungsi untuk menampilkan data yang telah dikirim oleh masing-masing *node*. Banyak pengambilan data bersangkutan dengan *interval* waktu kirim, untuk *interval* waktu kirim menggunakan hitungan menit. Jadi *user* ingin mengambil berapa banyak data dalam *interval* waktu beberapa menit. Misalkan banyak data diisi 10, *interval* waktu 1, artinya *user* akan menerima 10 data dalam waktu 1 menit sekali. Pada aplikasi tersebut juga terdapat penunjuk waktu. Untuk alur program pada aplikasi dapat dilihat pada gambar 3.7 :




Gambar 3.7 *flowchart* aplikasi *monitoring*

Pada Gambar 3.7 *flowchart* aplikasi *monitoring* terdiri 3 bagian pokok proses yaitu :

1. Inisialisasi

Pada inisialisasi adalah proses awal aplikasi dibuka dan menseting beberapa bagian. Pertama user harus memilih *com* USB *adapter xbee* yang terhubung dengan komputer *end user* misal pilihannya adalah *com* 2, jadi kita pilih *com* tersebut. Kemudian pilih *baudrate* sesuai dengan nilai *baudrate* pada *xbee*, setelah memilih baudrate isikan berapa banyak data yang akan diambil pada kolom pengambilan data, beserta *interval* waktunya. Setelah semua sudah di seting klik *button connect*, apabila tulisan *connect* berubah menjadi *disconnect* itu tandanya bahwa aplikasi telah *connect* atau terhubung dengan *xbee* dan aplikasi telah berjalan.

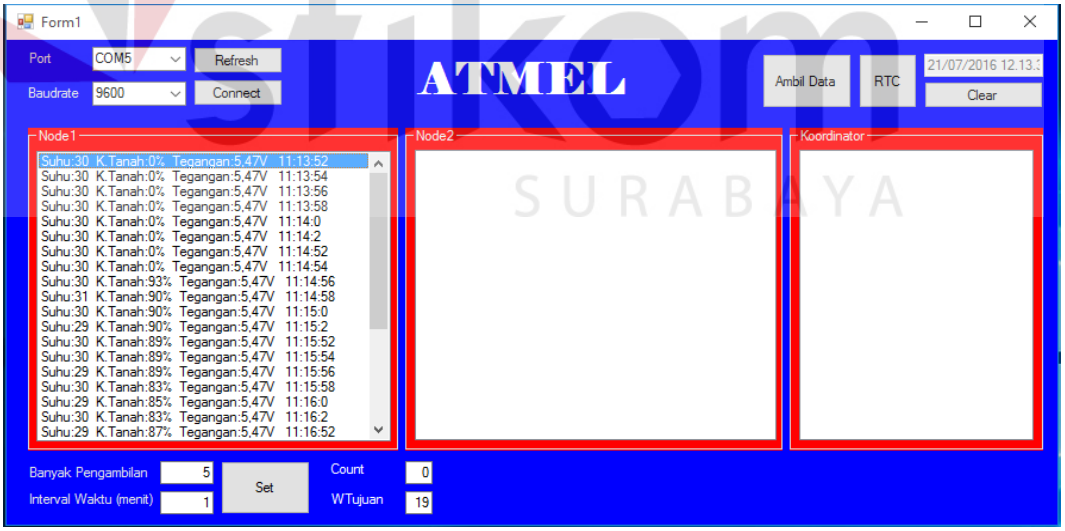
2. Proses *request* data

Proses pengambilan data berawal pada pembacaan *com*, jika *com* terbuka maka aplikasi langsung menuju proses pembacaan waktu jam,menit,dan detik. Kemudian waktu akan ditambahkan dengan interval waktu yang diseting oleh user. Waktu akan mengecek berlualang-ulang apakah waktu lebih dari atau sama dengan menit yang user seting, jika kondisi tersebut terpenuhi maka aplikasi mengirimkan Karakter “M” yang akan ditujukan kepada *node* sensor melalui *coordinator*.

3. Proses Penerimaan data

Setelah proses *request* data aplikasi telah siap menerima data, *node* sensor mengirim karakter “D” yang artinya node sensor telah bangun dari *slee mode* dan siap mengirim data. *Node* sensor mengirim data berdasarkan format pengiriman

data. Ketika aplikasi menerima data yang dikirimkan dari *node* sensor tidak langsung ditampilkan, namun terlebih dahulu melalui proses pencacahan data agar aplikasi mengetahui data yang dikirimkan. Dimulai dari data suhu yang ditandai dengan karakter "@", kemudian disimpan pada variabel suhu. Data kelembaban tanah ditandai dengan karakter "#" kemudian disimpan pada variabel k_tanah. Data Volt ditandai dengan karakter "$" kemudian disimpan pada variabel Volt. Data jam ditandai dengan karakter "^" kemudian disimpan pada variabel jam. Data menit ditandai dengan karakter "&" kemudian disimpan pada variabel menit. Data detik ditandai dengan karakter "*" kemudian disimpan pada variabel detik. Setelah data terkumpul aplikasi akan menampilkan seleruh data pada kolom *node* dan kolom *node* yang berwarna merah menjadi hijau pertanda bahwa *node* tersebut telah aktif. Berikut adalah Gambar 3.8 yang menunjukan aplikasi menampilkan data :




Gambar 3.8 Aplikasi Menerima dan Menampilkan Data

4. Sinkronisasi Waktu

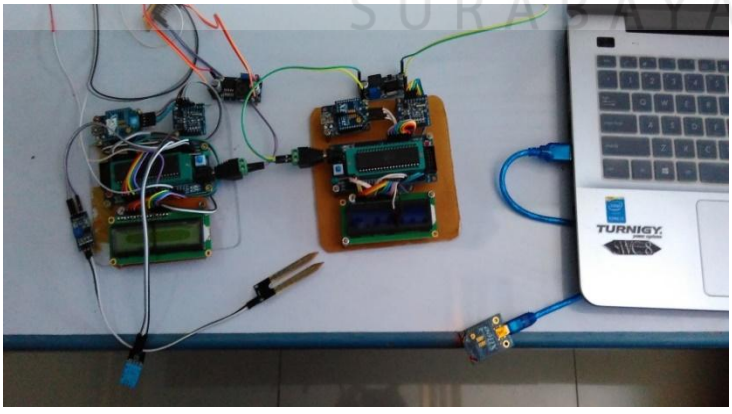
Pada aplikasi juga dilakukan sinkronisasi waktu. Waktu pada aplikasi digunakan sebagai waktu utama, yang menjadi barometer waktu pada *node*

*coordinator* dan *node* sensor. Aplikasi langsung mengambil data waktu dari komputer, jadi waktu pada aplikasi juga harus secara *realtime*. Berikut adalah potongan program pada aplikasi pada saat mengambil data waktu komputer.

```
Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles
Timer1.Tick
jam_leptop = Now.Hour
If jam_leptop > 12 Then
   jam_leptop = jam_leptop - 12
End If
   menit_leptop = Now.Minute
   detik_leptop = Now.Second
   TextBox1.Text = Now
End Sub
```

### 3.4 Perakitan Seluruh Alat

Setelah proses inisialisasi sensor yang telah terhubung pada mikrokontroler dan telah di uji coba kemudian menguji koneksi *xbee* antar *node* sesuai peranannya. Sistem *sleep mode* juga berjalan dengan baik, aplikasi pada sistem *monitoring* telah slesai di konsep sesuai kebutuhan, maka selanjutnya adalah perakitan keseluruhan alat mulai dari *node* sensor dan *coordinator* dirangkai beserta sensor,dapat dilihat pada Gambar 3.9.



Gambar 3.9 Perakitan keseluruhan Alat