BAB III

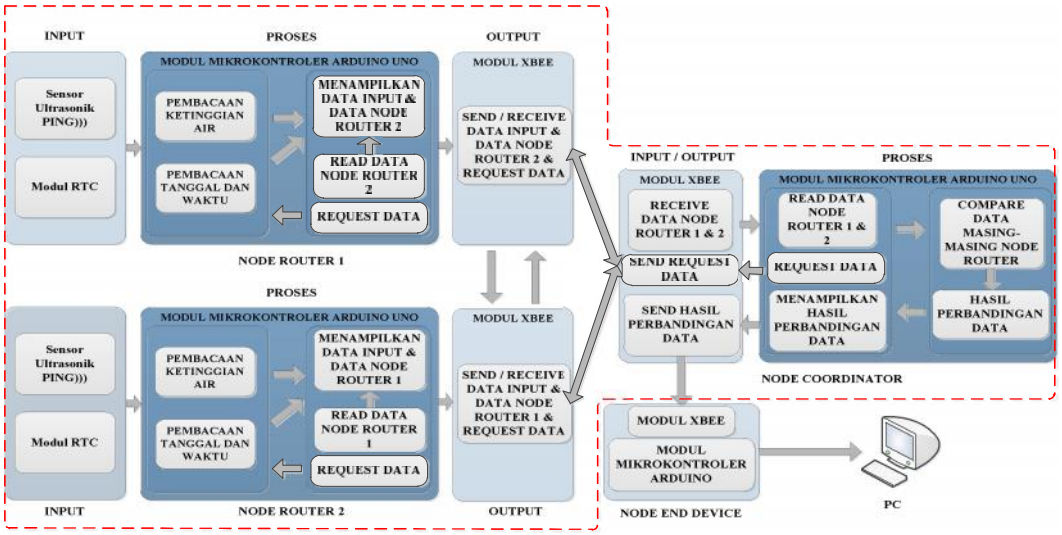
METODE PENELITIAN DAN PERANCANGAN SISTEM

3.1 Metode Penelitian

Metode penelitian yang digunakan pada tugas akhir ini melalui beberapa tahapan penelitian dan mencari informasi tentang data yang dibutuhkan dalam melakukan tugas akhir ini. Penelitian pertama adalah pengembangan konsep penelitian berdasarkan daftar pustaka. Selanjutnya perencanaan penelitian meliputi perancangan sistem perangkat keras dan perangkat lunak. Informasi data-data meliputi sensor ultrasonik PING dan modul RTC sebagai input data ketinggian air pada prototipe secara *real time*, kemudian diproses dalam modul mikrokontroler arduino uno. Modul Xbee *series* 2 sebagai *receiver* dan *transmitter* data secara nirkabel dari dan ke *node* lainnya serta informasi dalam penerimaan dan pengirimin data.

Setelah didapatkan informasi mengenai hal-hal yang dibutuhkan maka langkah selanjutnya adalah membuat skrip perancangan sistem WSN menggunakan *software* arduino IDE untuk menghasilkan informasi data yang nantinya akan digunakan dalam pengujian pengiriman dan penerimaan data ketinggian air secara *real time*.

Gambar 3.1 merupakan gambar blok diagram sistem yang merupakan penjelasan singkat dari perancangan sistem yang dibuat pada judul tugas akhir "Rancang Bangun Prototipe Aplikasi Wireless Sensor Network (WSN) Untuk Peringatan Dini Terhadap Banjir".



Gambar 3.1 Blok Diagram Sistem

Dari blok diagram sistem Gambar 3.1, terbagi menjadi 3 kelompok bagian, yaitu bagian *input*, proses dan *output*. Dalam tugas akhir ini, penulis hanya membahas tentang *node router* 1, *node router* 2, dan *node coordinator* yang terletak di dalam garis merah dari blok diagram sistem. Sedangkan pada blok diagram *node* end device dan PC dikerjakan dalam tugas akhir berjudul " Rancang Bangun Aplikasi Pemantau Data Wireless Sensor Network Untuk Peringatan Dini Terhadap Banjir" oleh Muhammad Syakir Kautsar.

3.1.1 *Input data*

Pada *node router* 1 dan 2, bagian *input* adalah proses dimana sensor ultrasonik sebagai detektor guna mendeteksi jarak ketinggian air pada prototipe yang akan dilakukan jika menerima perintah *request* data oleh *node coordinator*. Agar sensor dapat mendeteksi jarak, modul mikrokontroler arduino uno mengirimkan sinyal pulsa positif (*HIGH*) selama 2 «*s* sampai 5 «*s* ke pin SIG (I/O pin) pada sensor ultrasonik PING. Pemicu oleh mikrokontroler ini, menyebabkan sensor PING akan memancarkan gelombang suara 40 KHz.

Gelombang suara ini akan menghasilkan pantulan setelah mengenai permukaan air dan kembali ke sensor penerima. Bila sensor menerima sinyal pantulan, maka akan mengirimkan pulsa rendah (*LOW*) melalui pin SIG ke mikrokontroler. Sedangkan modul RTC adalah pencatat waktu pada saat pengambilan data ketinggian air pada prototipe. Pada *node coordinator*, bagian input adalah data-data yang berasal dari *node router* 1 dan 2.

Data ketinggian air oleh sensor ultrasonik dan pencatat waktu oleh RTC akan diproses di dalam mikrokontroler.

3.1.2 Bagian Proses

Pada bagian proses ini akan melakukan perhitungan agar mendapatkan ketinggian air dan proses lainnya pada modul mikrokontroler arduino uno menggunakan *software* arduino IDE. Disini penulis membuat skrip yang dibutuhkan untuk masing – masing *node* agar dapat berfungsi sesuai perancangan sistem dan blok diagram sistem.

Bagian proses terdiri dari beberapa tahap pada *node router* 1 dan 2. Beberapa tahap tersebut yaitu pembacaan ketinggian air yang dilakukan oleh sensor ultrasonik, pembacaan tanggal dan waktu oleh modul RTC, pembacaan data *node router* 2 jika proses terjadi pada *node router* 1, pembacaan data *node router* 1 jika proses terjadi pada *node router* 2. Kemudian *node router* 1 akan menampilkan data inputan dan data dari *node router* 2 pada serial monitor *software* arduino IDE, sedangkan *node router* 2 akan menampilkan data inputan dan data dari *node router* 1. Semua proses tersebut dilakukan apabila *node router* 1 dan 2 telah menerima pesan *request* data dari *node coordinator*. Pada *node coordinator* akan melakukan proses *request* data dan kemudian pembacaan data –

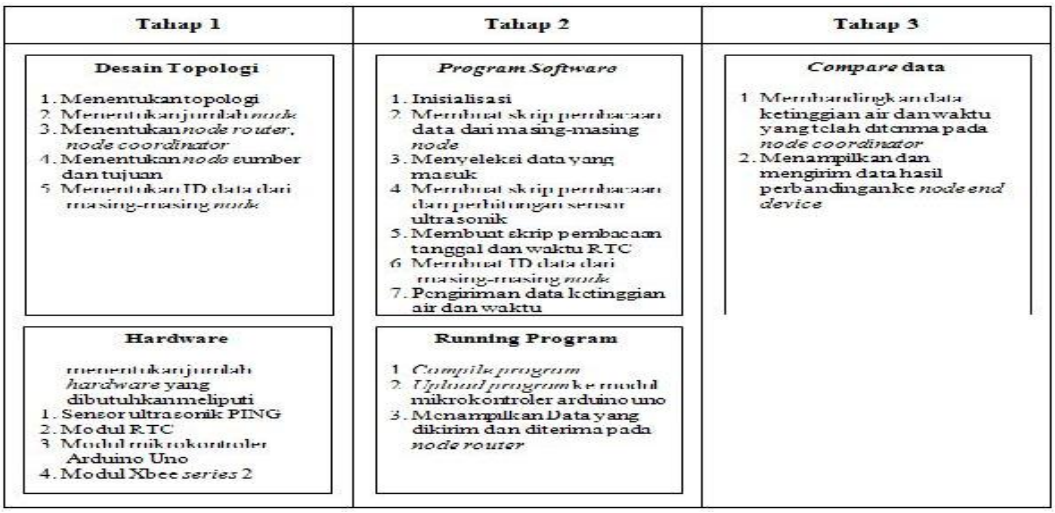
data dari *node router* 1 dan 2, kemudian membandingkan data tersebut apakah *valid* atau tidak *valid* (*error*). Selanjutnya hasil perbandingan tersebut akan ditampilkan di serial monitor pada *software* arduino IDE.

3.1.3 Bagian *output*

Setelah data ketinggian air dan waktu didapat pada *node router* 1, maka data tersebut akan dikirimkan ke *node router* 2 dan *node coordinator*. Sedangkan data yang didapat pada *node router* 2 akan dikirimkan ke *node* sensor 1 dan *node coordinator*. Data-data yang diterima *node coordinator* akan dibandingkan, kemudian hasil perbandingan data tersebut akan ditampilkan pada serial *monitor node coordinator* dan kemudian dikirimkan ke *node end device*. Data yang diterima pada *end device* akan ditampilkan pada PC menggunakan program yang telah dibuat dengan *software visual basic*.

3.2 Perancangan Sistem

Pada perancangan sistem WSN yang dilakukan, dapat dilihat berdasarkan tahap setiap proses yang akan di jalankan pada bagan Gambar 3.2.

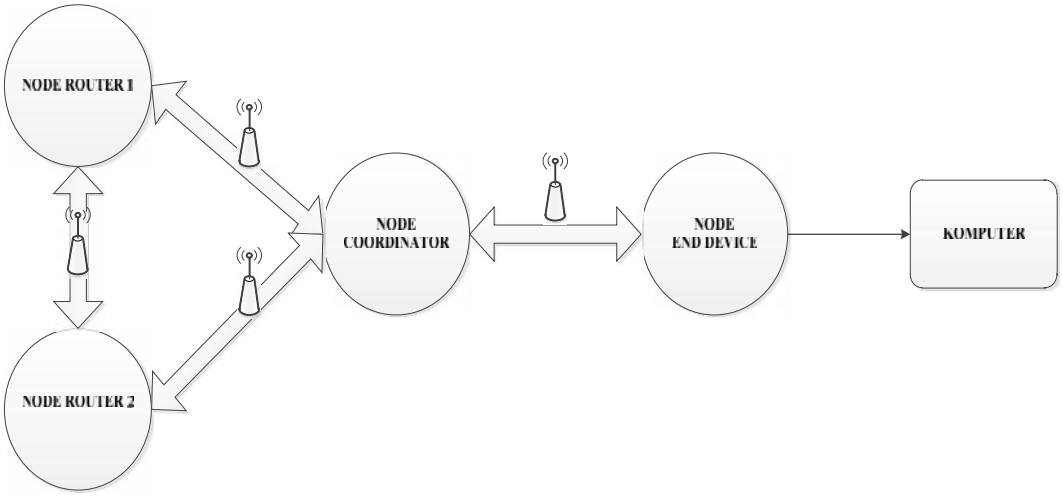


Gambar 3.2 Bagan Tahap Proses Perancangan Sistem

Gambar 3.2 merupakan Bagan tahap proses perancangan sistem yang akan dilakukan. Pada bagan ini proses sistem dapat dibagi ke dalam tiga kelompok. Tahap pertama yang dilakukan adalah menyiapkan hardware yang dibutuhkan dalam sistem dan membuat desain topologi. Tahap kedua yaitu membuat skrip pada *software* arduino IDE dan meng-*compile* program, yang selanjutnya di-*upload* ke dalam modul mikrokontroler arduino uno. Tahap yang ketiga adalah melakukan perbandingan data yang telah diterima oleh *node coordinator*, kemudian data hasil perbandingan tersebut dikirim ke *node end device* yang selanjutnya akan ditampilkan pada komputer menggunakan program *visual basic*.

3.3 Desain Topologi

Pada perancangan sistem ini menggunakan model topologi *point to multipoint*. Model topologi *point to multipoint* ini digunakan agar *node router* 1, 2 dan *coordinator* dapat berkomunikasi dengan dua atau lebih *node* yang berbeda pada satu jaringan. Gambar 3.3 merupakan topologi *point to multipoint* pada perencanaan sistem WSN.



Gambar 3.3 Topologi *Point to Multipoint*

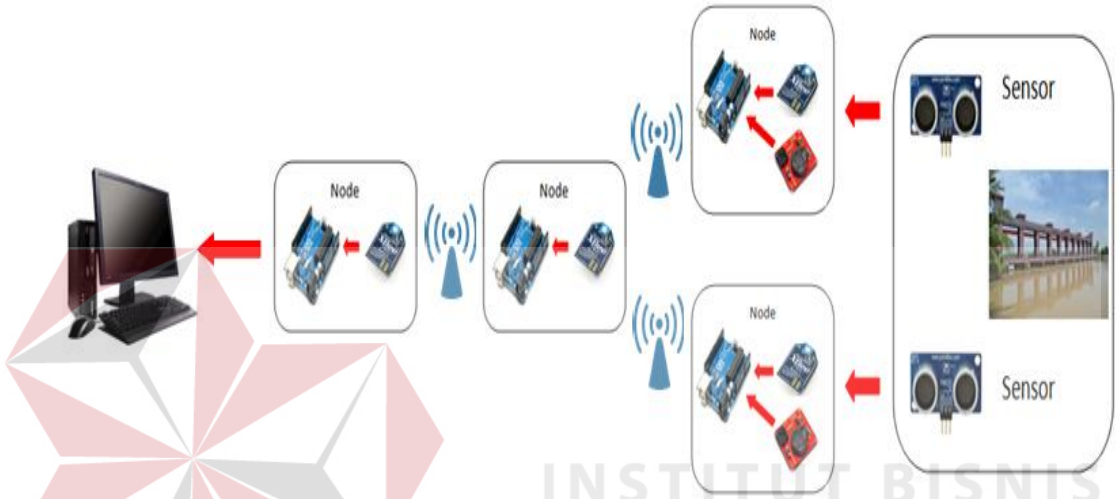
Pada perencanaan sistem WSN menggunakan topologi *point to multipoint* ini, diharapkan proses pengiriman masing-masing *node* dapat berjalan lancar.

a) Jumlah *node* yang digunakan berjumlah 4, yaitu *node router* 1, 2, *coordinator* dan *end device*.
b) *Node router* 1 akan mengirimkan data ketinggian air dan waktu secara *broadcast* dan nirkabel ke *node router* 2 dan *node coordinator*, jika terlebih dahulu mendapatkan *request* data dari *node coordinator*.
c) *Node router* 2, akan mengirimkan data ketinggian air dan waktu secara *broadcast* dan nirkabel ke *node router* 1 dan *node coordinator*, jika terlebih dahulu mendapatkan *request* data dari *node coordinator*.
d) *Node coordinator* mengirimkan *request* data ke *node router* 1 dan 2 secara *broadcast*. Jika data *node router* 1 dan 2 baik itu melalu *node router* itu sendiri maupun *node router* lainnya telah diterima oleh *node coordinator*, maka data-data tersebut akan dibandingkan. Kemudian hasilnya akan dikirimkan ke *node end device*.
e) *Node end device* menerima data dari *node coordinator*.
f) Komputer akan mengabil data yang terdapat di *node end device*s, selanjutnya akan menampilkannya melalui program yang telah dibuat.

3.4 *Hardware*

Gambar 3.4 merupakan perangkat keras yang dibutuhkan dalam tugas akhir ini. Pada *node router* 1 dan 2 perangkat keras yang dibutuhkan yaitu sensor ultrasonik PING untuk mendapatkan ketinggian air pada prototipe (Gambar 3.5), modul RTC untuk mendapatkan tanggal dan waktu, modul mikrokontroler

arduino uno untuk *processing*, Xbee *series* 2 untuk mengirim atau menerima data secara nirkabel, dan Xbee *shield* sebagai penghubung antara modul arduino uno dengan Xbee *series* 2. Sedangkan pada *node coordinator* perangkat keras yang dibutuhkan terdiri dari modul mikrokontroler arduino uno, Xbee *series* 2 dan Xbee *shield*.



Gambar 3.4 Perangkat Keras

Berdasarkan desain topologi sebelumnya *node router* 1, 2, dan *coordinator* dapat berkomunikasi secara *point to multipoint*. Agar dapat berkomunikasi *point to multipoint*, terdapat parameter-parameter yang harus diatur didalam Xbee *series* 2 menggunakan *software* X-CTU. Parameter-parameter tersebut adalah PAN ID dalam satu jaringan harus sama, kemudian DH dan DL yang diatur nilainya menjadi DH = 0 dan DL = FFFF, FFFF mempunyai arti bahwa data akan dikirim secara *broadcast* sehingga beberapa *node* dalam PAN ID yang sama akan mendapat data tersebut (Gambar 2.22).

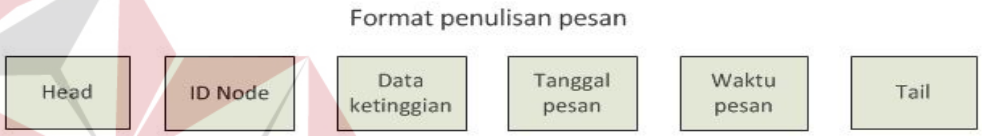
Gambar 3.5 Prototipe Pengukuran Ketinggian Air Sungai

3.5 Pemrogaman mikrokontroler Arduino Uno pada Software ArduinoIDE

Untuk melakukan perancangan sistem ini terlebih dahulu harus membuat skrip atau melakukan pemograman mikrokontorler arduino uno pada *software* arduino IDE agar mendapat hasil sesuai dengan yang dibutuhkan.

3.5.1 Format Penulisan Pesan

Setiap *node* dalam WSN ini akan mengirimkan pesan data dari satu *node* ke *node* lainnya. Format pengiriman atau penulisan pesan yang terjadi pada setiap *node* diatur seperti Gambar 3.6.

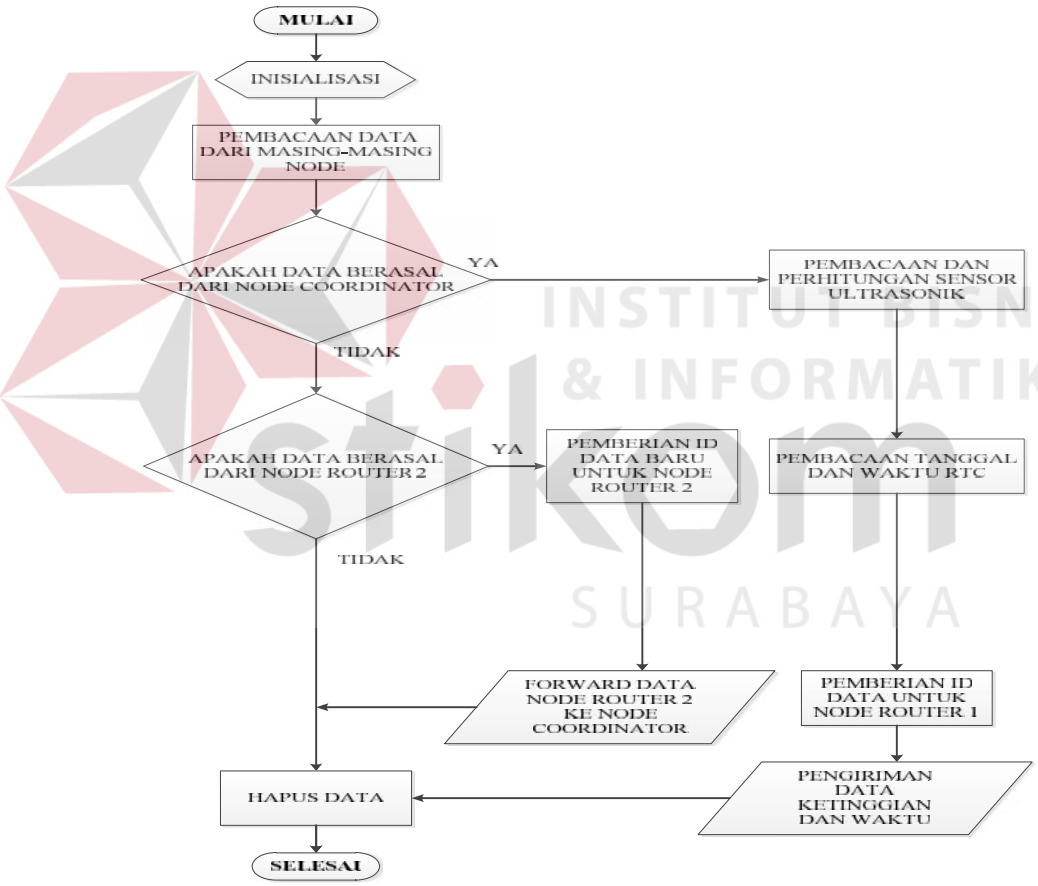


Gambar 3.6 Format Penulisan Pesan

Format penulisan pesan ini terinspirasi oleh pengiriman pesan Xbee dengan mode API. Mode ini dapat memberi informasi *source address* pengirim ke Xbee lainnya yang dituju pada saat pengiriman data. Dengan demikian, maka setiap *node* pada jaringan WSN ini dapat mengetahui data yang diterima berasal dari *node* yang mana (pengirim). Setiap *node* memiliki *head* dan id *node* yang harus dikenali. Semua penulisan tersebut ditulis semua dan dipisahkan oleh tanda titik. Dalam tugas akhir ini penulis menggunakan simbol % untuk digunakan sebagai *head*. Maka setiap *node* akan mengetahui apabila data yang diterima harus mempunyai awalan %. Simbol *head* disini akan mengantisipasi sebuah *node* ilegal yang tiba-tiba mengirimkan data ke *node*. Apabila data yang diterima tidak memiliki *head*, maka data tersebut akan dianggap ilegal dan *node* akan membuang data tersebut.

Dari penjelasan di atas, *node router* 1 akan memiliki id “N1”, *node router* 2 akan memiliki id “N2”, data *node router* 1 yang melewati *node router* 2 akan memiliki id “S1”, data *node router* 2 yang melewati *node router* 1 akan memiliki id “S2”, data *node coordinator* akan memliki 2 id yaitu id “C1” untuk data yang berasal dari *node router* 1 dan id “C2” untuk data yang berasal dari *node router* 2.

3.5.2 Membuat Skrip untuk *Node router* 1



Gambar 3.7 Diagram Alir Pembuatan Skrip pada *Node router* 1

Berikut ini penjabaran dari Gambar 3.7 proses pembuatan skrip pada *node router* 1.

1. Inisialisai

Pembuatan skrip ini dimulai dengan menuliskan skrip berikut ini. skrip ini merupakan inisialisasi dan harus ada dalam setiap program yang dibuat pada *node router*.

```
// library yang digunakan
#include <Wire.h>
#include "RTClib.h"
// inisialisasi variabel
String id,indata,tampung;
const int pingPin = 7;
// inisialisasi RTC
RTC_DS1307 RTC;
void setup() {
  // inisialisasi komunikasi serial
  Serial.begin(9600);
  // inisialiasi Wire (komunikasi I2C)
  Wire.begin();
  // start RTC
  RTC.begin();
  // hanya sekali digunakan untuk mengatur tanggal dan waktu
  // pada modul RTC
  RTC.adjust(DateTime(__DATE__, __TIME__));
}
//semua proses yang dilakukan secara berulang-ulang akan
dimasukkan ke dalam fungsi loop ini
void loop(){}
```

2. Pembacaan Data *Node*

Pembacaan data dilakukan secara berulang-ulang oleh masing-masing *node* ketika ada data yang dikirimkan dari *node* lain dan kemudian data tersebut diterima (data masuk diakhiri dengan karakter \n atau *enter*). Contoh skrip pembacaan data dari suatu *node* dalam jaringan.

```
void loop(){
while(Serial.available()>0){
    char data_masuk=(char)Serial.read();
     id += data_masuk;
    if(data_masuk =='\n'){}
}
```

3. Menyeleksi Data dari *Node coordinator*

Setelah data yang diakhiri karakter \n diterima, dan data tersebut berasal dari *node coordinator* yang melakukan *request* data (mengirim angka %1), maka

node router akan melakukan penyeleksian. Berikut skrip penyeleksian data yang berasal dari *node coordinator*.

```
void loop(){
if(data_masuk =='\n'){
if((id[0]=='%') && (id[1]=='1')) //skrip penyeleksian
     {
       delay(1000);
       sph(); //merupakan fungsi pembacaan sensor dan waktu
       id="";
     }
  }
}
```

4. Pembacaan dan Perhitungan Sensor Ultrasonik

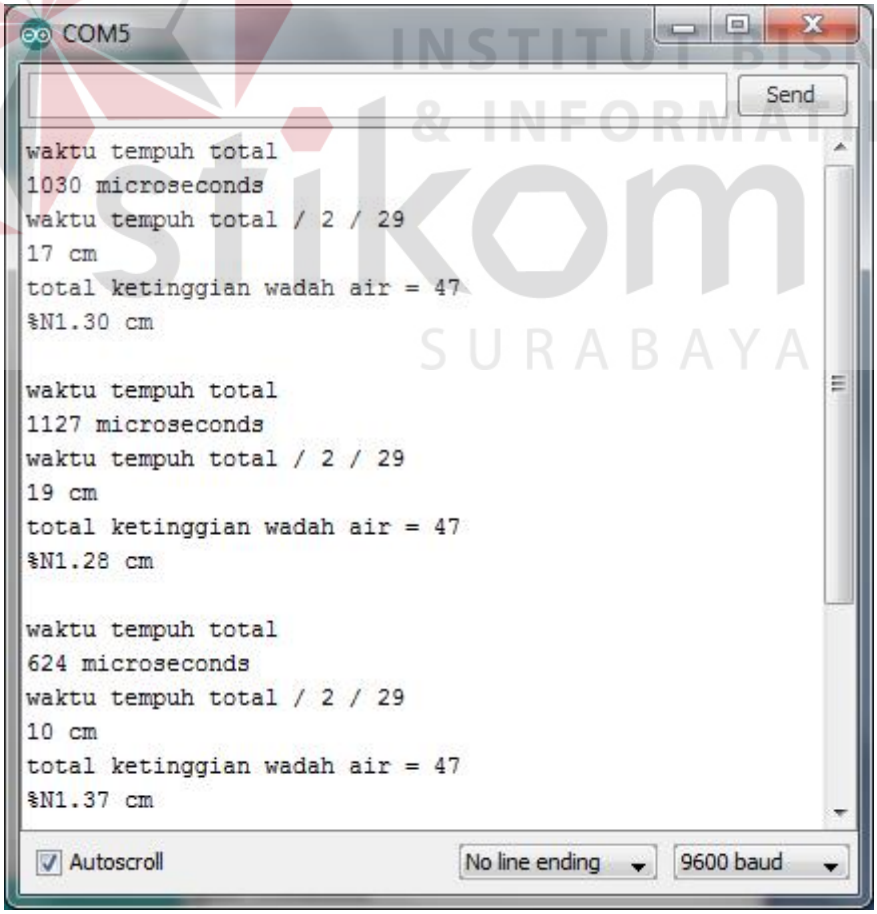
Dengan adanya permintaan *request* data dari *node coordinator*, maka *node router* akan melakukan pembacaan dan perhitungan ketinggian air yang dilakukan sensor ultrasonik. Contoh skrip sebagi berikut.

```
//merupakan fungsi konversi microsecond ke centimeter
long microsecondsToCentimeters(long microseconds)
{
    return microseconds / 29 / 2;
  }
//merupakan fungsi untuk memicu sensor ultrasonik agar dapat
membaca ketinggian air
void sph()
 {
    long duration, cm, i;
    pinMode(pingPin, OUTPUT);
    digitalWrite(pingPin, LOW);
    delayMicroseconds(2);
    digitalWrite(pingPin, HIGH);
    delayMicroseconds(5);
    digitalWrite(pingPin, LOW);
    pinMode(pingPin, INPUT);
    duration = pulseIn(pingPin, HIGH);
    cm = microsecondsToCentimeters(duration);
    i = total ketinggian wadah air - cm;
 }
```

Fungsi konversi *microsecond* ke centimeter merupakan pengukuran jarak sensor ultrasonik dengan memanfaatkan konstanta kecepatan gelombang ultrasonik 1130 *feet/second* atau 344 m/s. Sehingga untuk menempuh jarak 1 cm dibutuhkan waktu : 29 µs. Dengan menghitung waktu tempuh ultrasonik kemudian dibagi dengan 29 µs maka kita akan mendapatkan jarak tempuh.

Fungsi sph ( ) untuk membuat sinyal ultrasonik dengan frekuensi 40 KHz dengan cara pin SIG dibuat *HIGH* selama 2 µs s/d 5 µs. Gelombang ultrasonik akan terpancar sampai mengenai sasaran kemudian akan terpantul menuju kembali ke sensor PING. Selama ultrasonik belum diterima kembali oleh sensor, kondisi logika pin SIG adalah *HIGH*. Tunggu sampai gelombang ultrasonik diterima kembali (setelah terpantul) dengan tanda pin SIG berubah menjadi LOW.

Jika sinyal ultrasonik sudah diterima kembali, waktu tempuh yang terhitung adalah 2 kali jarak tempuh yaitu kirim – terima (atau bahasa umumya pergi – pulang). Waktu tempuh sensor dengan target berarti waktu tempuh total dibagi 2. Jarak antara sensor dengan target = waktu tempuh sensor ke target/29 µs (cm) (inkubator-teknologi.com, 2012).



Gambar 3.8 Proses Pembacaan Sensor Ultrasonik

Pada Gambar 3.8 merupakan proses pembacaan sensor ultrasonik hingga dapat menghasilkan jarak ketinggian air sungai pada prototipe. Waktu tempuh total merupakan waktu sinyal gelombang ultrasonik pada saat dikirim yang kemudian terpantul karena mengenai permukaan air dan diterima kembali oleh sensor (2 kali jarak tempuh). Waktu tempuh total akan dibagi 2 dan dibagi 29 karena untuk menempuh jarak 1 cm dibutuhkan waktu 29 µs. Setelah itu akan didapat jarak antara sensor dengan permukaan air dalam satuan cm. karena yang dibutuhkan adalah jarak ketinggian air sungai pada prototipe, maka total ketinggian wadah air dikurangi jarak antara sensor dengan permukaan air untuk mendapatkan jarak ketinggian air pada prototipe yang sebenarnya.

5. Pembacaan Tanggal dan Waktu RTC

Pembacaan tanggal dan waktu digunakan untuk mengetahui waktu pada saat pengambilan data ketinggian air oleh sensor ultrasonik. Dengan cara ini dapat mengetahui perubahan ketinggian air secara *realtime*.

```
void sph()
 {
   DateTime now = RTC.now();  //mengambil tanggal dan waktu
 }
```

6. Pemberian ID Data dan Pengiriman Data

Pemberian ID data ini dilakukan untuk mempermudah pengguna melihat data ketinggian air dan waktu yang berasal dari beberapa *node router*. *Node router* 1 memilik simbol “%” dan ID data berawalan “N1” dan *node router* 2 memiliki ID data berawalan “N2” yang diikuti dengan data ketinggian air dan waktu. Kemudian data tersebut dikirim ke *coordinator* dan *node router* lainnya.

```
void sph()
 {
    //mengirimkan simbol ”%” dan ID “N1”
    Serial.print('%');
    Serial.print("N1.");
```

```
   //mengirim ketinggian air melalui variabel i
    Serial.print(i);
    Serial.print('.');
    //mengirim tanggal
    DateTime now = RTC.now();
    Serial.print(now.day(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.year(), DEC);
    //mengirim waktu
    Serial.print('.');
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.print('$');
    Serial.print('\n');
 }
```

7. Menyeleksi Data dari *Node router* 2

Skrip untuk menyeleksi jika data dari *node router* 2 yang masuk ke dalam *node router* 1 sebagai berikut.

```
if(data_masuk =='\n'){
else if ((id[0]=='%') && (id[1]=='N'))
{
       delay(4500);
       sensor(); // fungsi pemberian ID data baru untuk node
                 // router 2 dan meneruskan data tersebut ke
                // node coordinator
     }
}
```

Jika data masuk berawalan “%” dan “N”, maka akan menunggu selama 4,5 detik sebelum melakukan panggilan terhadap fungsi sensor ().

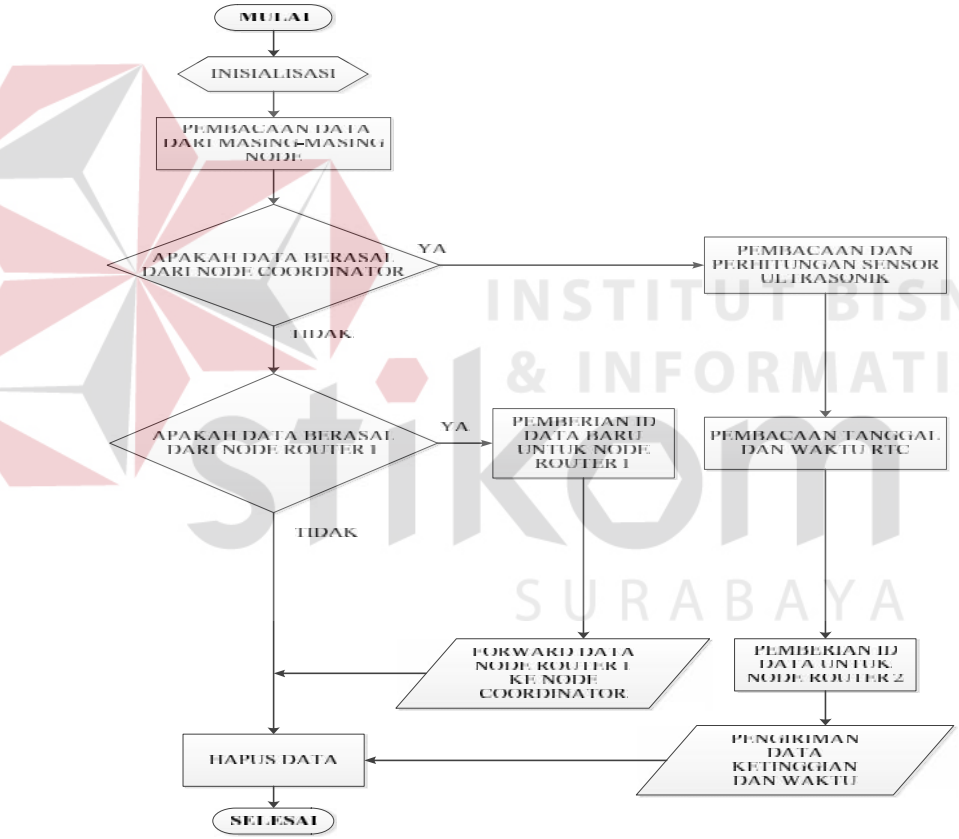
8. Pemberian ID Data baru dan Meneruskan Data *Node router* 2

Pemberian ID data baru ini dilakukan untuk mempermudah pengguna mengetahui ID data *node router* 2 yang melalui *node router* 1 untuk selanjutnya akan diteruskan ke *node coordinator*. ID data *node router* 2 akan berubah dari “N2” menjadi “S2” jika melalui *node router* 1.

```
void sensor ()
{
    if (id[0]=='%')
```

```
    {
      if (id[1]=='N')
      {
        id[1]='S'; //perubahan ID “N” menjadi “S”
        if (id[2]=='2')
        {
        indata=id;
        }
    }
    Serial.print(id); //meneruskan atau mengirim data ke node
                      //coordinator
  }
  id="";
}
```

3.5.3 Membuat Skrip untuk *Node router* 2



Gambar 3.9 Diagram Alir Pembuatan Skrip pada *Node router* 2

Pada Gambar 3.9 proses diagram alir pembuatan skrip pada *node router* 2 hampir sama dengan diagram alir pada *node router* 1. Perbedaannya hanya terdapat pada pembuatan skrip untuk pemberian ID data *node router* 2 itu sendiri

dan pemberian ID data baru untuk *node router* 1 yang melalui *node router* 2. ID data *node router* 2 diberi awalan “N2”. Sedangkan ID data *node router* 1 yang sebelumnya berawalan “N1” berubah menjadi “S1” karena data tersebut melalui *node router* 2 untuk diteruskan ke *node coordinator*. Berikut ini merupakan perbedaan skrip yang terdapat pada *node router* 2.

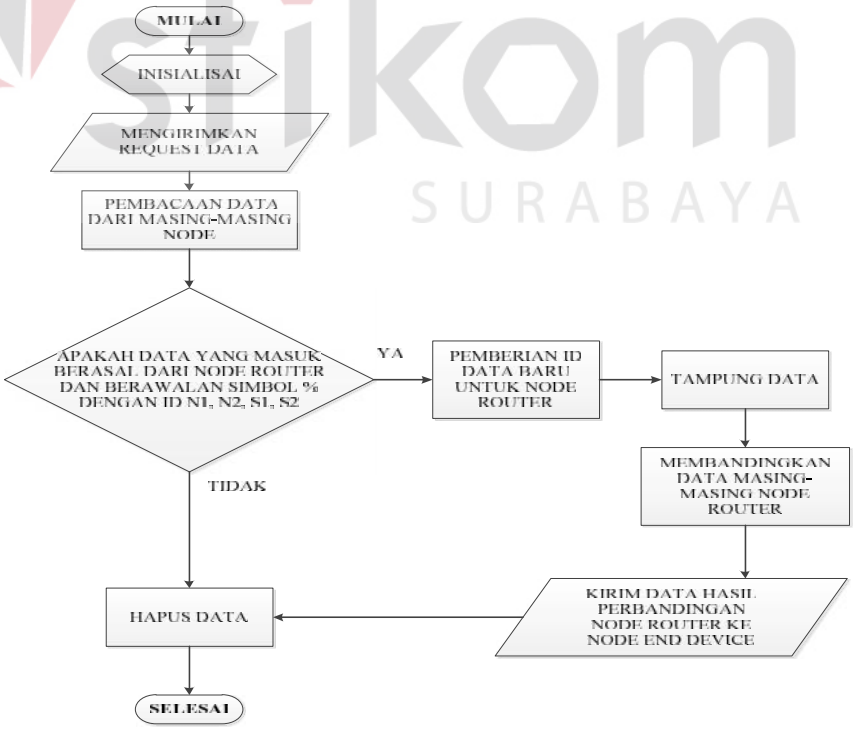
```
void sph()
 {
    //mengirimkan simbol ”%” dan ID “N2”
    Serial.print('%');
    Serial.print("N2.");
    //mengirim ketinggian air melalui variabel i
    Serial.print(i);
    Serial.print('.');
    //mengirim tanggal
    DateTime now = RTC.now();
    Serial.print(now.day(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.year(), DEC);
    //mengirim waktu
    Serial.print('.');
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.print('$');
    Serial.print('\n');
 }
```

Kemudian skrip perubahan ID data *node router* 1.

```
void sensor ()
{
    if (id[0]=='%')
    {
      if (id[1]=='N')
      {
        id[1]='S'; //perubahan ID “N” menjadi “S”
        if (id[2]=='1')
        {
        indata=id;
        }
    }
    Serial.print(id); //meneruskan atau mengirim data ke node
                      //coordinator
  }
  id="";  }
```

3.5.4 Membuat Skrip untuk *Node coordinator*

Pada perencanaan sistem ini, *node coordinator* akan mengirimkan pesan "%1" yang berarti meminta data atau *request* data secara *broadcast* setiap 20 detik sekali yang akan diterima oleh *node router* 1 dan 2. Setelah mengirimkan *request*, *node coordinator* akan menunggu data dari *node router* 1 dan 2. *Node* ini hanya menerima data dari *node router* 1 dan 2 dengan format pesan data berawalan "%" dan memiliki ID "N1", "N2", "S1" dan "S2". Ketika keempat data ini diterima, maka seluruh ID akan dirubah menjadi "C" dan kemudian disimpan dalam variabel tampung yang sudah dibuat. Jika semua tampung sudah terisi oleh keempat data tersebut, *node* ini akan membandingkan data tersebut sesuai masing-masing *node router*. Dan hasil perbandingan data ini akan dikirimkan ke *node end device* (Gambar 3.10) yang kemudian akan ditampilkan di komputer menggunakan program *visual basic*.



Gambar 3.10 Diagram Alir Proses *Node coordinator*

Berdasarkan Gambar 3.10, berikut adalah contoh pemrogaman modul mikrokontroler arduino uno pada *node coordinator* yang diprogam pada *software* Arduino IDE :

1. Inisialisasi

Pembuatan skrip ini dimulai dengan menuliskan skrip berikut ini. skrip ini merupakan inisialisasi dan harus ada dalam program yang dibuat pada *node coordinator*.

```
int b=0;
String indata,tampung1,tampung2,tampung3,tampung4;
// variabel tampung0 yang menyimpan info errors pada node 1
String tampung0="%C1.errors";
// variabel tampung yang menyimpan info errors pada node 2
String tampung="%C2.errors";

void setup()
{
  Serial.begin(9600);
}
void loop(){}
```

2. Mengirimkan *request* data dan pembacaan data

Skrip ini merupakan pengiriman *request* data (angka %1) selama 20 detik sekali ke *node router* 1 dan 2. Kemudian akan membaca data yang masuk dari *node router* 1 dan 2 selama 4 kali. Ketika pembacaan keempat kalinya, program akan memanggil atau menjalankan fungsi banding ( ) untuk membandingkan data yang diterima *node coordinator*.

```
void kirim(int dat)
{
  if (dat == 1)
  {
    //kondisi tampung dikosongkan setiap akan mengirimkan request
    //data
    tampung1="";
    tampung2="";
    tampung3="";
    tampung4="";
  Serial.print('%');
  Serial.print(dat);
  Serial.print(" : request data :");
  Serial.print('\n');
```

```
  }
}

void loop(){
b=1;
//merupakan pembacaan data yang dilakukan selama 4 kali dengan
//selang waktu 5 detik pada setiap kali pembacaannya.
while(b<5){
  kirim(b);
  while(Serial.available()>0){
      char data_masuk=(char)Serial.read();
      indata += data_masuk;
      if(data_masuk=='\n'){}
        seleksi(); //memanggil atau menjalankan fungsi seleksi
                   //data
        indata="";
      }
    }
    indata="";
    if (b == 4)
    {
      banding(); //memanggil atau menjalankan fungsi banding data
    }
delay(5000);
b++;
}
}
```

3. Menyeleksi data, pemberian ID baru, dan tampung data

Void seleksi merupakan fungsi yang berisi skrip untuk menyeleksi data yang masuk dan pemberian ID baru "C" yang selanjutnya akan ditampung sebelum data tersebut dibandingkan. Karena *node* ini hanya menerima data dari *node router* 1 dan 2 yang masing-masing mempunyai format pesan dengan simbol awal "%" dan ID "N1", "N2", "S1" dan "S2", maka skrip penyeleksian data dan pemberian ID data yang baru dalam hal ini ID "C" sebagai berikut.

```
void seleksi()
{
  if ((indata[0]=='%') && (indata[1]=='N') )
  {
      //pemberian ID baru C
        indata[1]='C';
      if (indata[2]=='1')
      {
        tampung1=indata;// tampung data untuk N1
        indata="";// mengkosongkan data
       Serial.print("N1");
        Serial.print(tampung1);

      }
```

```
      if (indata[2]=='2')
      {
        tampung3=indata;// tampung data untuk N2
        indata="";// mengkosongkan data
        Serial.print("N2");
        Serial.print(tampung3);

      }
    }
    else if ((indata[0]=='%') && (indata[1]=='S'))
    {
      //pemberian ID baru C
      indata[1]='C';
      if (indata[2]=='1')
      {
        tampung2=indata;// tampung data N1 melalui N2
        indata="";// mengkosongkan data
       Serial.print("N1viaN2");
        Serial.print(tampung2);

      }
      if (indata[2]=='2')
      {
        tampung4=indata;// tampung data N2 melalui N1
        indata="";// mengkosongkan data
       Serial.print("N2viaN1");
        Serial.print(tampung4);
      }
    }
  else
  {
    indata="";
  }
}
```

4. Membandingkan data dan pengiriman data

Skrip untuk membandingkan data dan pengiriman hasil perbandingan dijadikan dalam satu fungsi. Fungsi tersebut adalah *void* banding. Fungsi ini akan melakukan proses perbandingan data dari masing-masing *node router*.

```
void banding()
{
  Serial.println("Hasil Perbandingan :");
  //jika seluruh tampung tidak terisi atau node coordinator tidak
  //menerima data
if ((tampung1 == "") && (tampung2 == ""))
{
  Serial.println(tampung0);
}
if ((tampung3 == "") && (tampung4 == ""))
{
  Serial.println(tampung);

}
```

```
  // jika tampung terisi 1
if ((tampung1 != "") && (tampung2 == ""))
{
  Serial.print(tampung1);
}
if ((tampung1 == "") && (tampung2 != ""))
{
  Serial.print(tampung2);
}
if ((tampung3 != "") && (tampung4 == ""))
{
  Serial.print(tampung3);
}
if ((tampung3 == "") && (tampung4 != ""))
{
  Serial.print(tampung4);
}
// dua tampung terisi
if ((tampung1 != "") && (tampung2 != ""))
{
  if (tampung1 == tampung2)
  {
    Serial.print(tampung1);
  }
}

if ((tampung3 != "") && (tampung4 != ""))
{
  if (tampung3 == tampung4)
  {
    Serial.print(tampung3);
  }s
 }
}
```

Terdapat beberapa kondisi perbandingan yang dilakukan oleh *node coordinator*. Tampung1 akan dibandingkan dengan tampung2, sedangkan tampung3 akan dibandingkan dengan tampung4. Tampung data yang dibandingkan selalu sama dalam setiap kondisi perbandingannya. Berikut kondisi perbandingannya:

1. Jika *node coordinator* tidak menerima data dari *node router* 1 atau 2, maka *node coordinator* akan mengirimkan pesan *error* yang terjadi pada *node router* 1 atau 2 ke *node router end device*.
2. Jika *node coordinator* hanya menerima salah satu data dari *node router* 1 atau 2, maka *node* ini akan membandingkan data tersebut. Jika salah satu tampung

terisi, *node coordinator* akan mengirimkan data sesuai dengan tampung yang diterima ke *node router end device*.

3. Jika *node coordinator* menerima semua data dari *node router* 1 dan 2 atau tampung1, 2, 3, 4 terisi, maka akan dilakukan perbandingan data. Jika data tampung1 sama dengan tampung2, maka *node* ini akan mengirimkan data salah satu dari tampung ke *node router end device* (dalam skrip ini akan mengirimkan data tampung1). Jika data tampung3 sama dengan tampung4, maka *node* ini akan mengirimkan data salah satu dari tampung ke *node router end device* (dalam skrip ini akan mengirimkan data tampung3).

Setelah membandingkan data-data tersebut, *node* ini akan mengirimkan hasil perbandingan data ke *node end device*. Kemudian komputer akan mengambil data hasil perbandingan dari *node end device* untuk ditampilkan pada program yang telah dibuat.