DESAIN DAN IMPLEMENTASI PROTOKOL *HANDOFF* DAN *ERROR CHECKING* PADA JARINGAN MWSN

(*MOBILE WIRELESS SENSOR NETWORKS*)

TUGAS AKHIR

Disusun oleh :

Nama : Septian Adi Herlambang

NIM : 09.41020.0083

Program : S1 (Strata Satu)

Jurusan :Sistem Komputer

SEKOLAH TINGGI

MANAJEMEN INFORMATIKA & TEKNIK KOMPUTER

SURABAYA

2014

Tugas Akhir

DESAIN DAN IMPLEMENTASI PROTOKOL *HANDOFF* DAN *ERROR CHECKING* PADA JARINGAN MWSN

(*MOBILE WIRELESS SENSOR NETWORKS*)

dipersiapkan dan disusun oleh

Septian Adi Herlambang

NIM :09.41020.0083

Telah diperiksa, diuji dan disetujui oleh Dewan Penguji

pada :September 2014

Susunan Dewan Penguji

Pembimbing

I. Dr. Jusak _____

II. Yuwono Marta Dinata, S.T., M.Eng. _____

Penguji

I. Helmy Widyantara, S.Kom., M.Eng. _____

II. Harianto, S.Kom., M.Eng. _____

Tugas Akhir ini telah diterima sebagai salah satu persyaratan

untuk memperoleh gelar Sarjana

Pantjawati Sudarmaningtyas, S.Kom., M.Eng., OCA
Pembantu Ketua Bidang Akademik

SEKOLAH TINGGI MANAJEMEN INFORMATIKA & TEKNIK KOMPUTER

SURABAYA

ABSTRAK

Selama dekade terakhir, perkembangan tekonologi komunikasi WSN (Wireless Sensor Networks) begitu pesat dilihat dari beberapa konsep yang dibangun menggunakan WSN seperti, sistem peringatan dini terhadap banjir, sistem pemantuan suhu dan kelembaban pada lahan tanaman jarak. WSN merupakan infrastruktur jaringan nirkabel atau tanpa kabel yang menggunakan sensor untuk memantau kondisi fisik atau lingkungan yang dapat terhubung ke jaringan, jaringan ini menggunakan gelombang radio sebagai media pengirimannya.

Pada umumnya penerapan WSN menggunakan model statis , namun pada kondisi tertentu model statis tidak cocok untuk diterapkan seperti pada kondisi lingkungan yang berubah-ubah. Model statis dapat mengalami masalah seperti, tidak semua daerah dapat terpantau, beberapa pengaplikasian WSN yang membutuhkan sensor dengan harga yang mahal cukup merepotkan.

Pada sistem MWSN ini menerapkan protokol *Handoff* atau *Handover*. *Handoff* atau *Hand Over* (HO) ialah proses perubahan pelayanan/peng-handle-an sebuah *Mobile Station* (MS) dari suatu *Base Station* (BS) ke BS lain dikarenakan adanya pergerakan MS yang menjauhi BS awal dan mendekati BS baru.

Berdasarkan hasil pengujian, didapatkan presentase *data loss* (data hilang) terhadap kecepetan *mobile* sebesar 70% - 80% dan persentase *data loss* (data hilang) terhadap jarak (di dalam ruangan) sebesar 30% - 80% sedangkan (di luar ruangan) sebesar 50% - 80%. Berdasarkan hasil pengujian didapatkan bahwa parameter jarak antar xbee sangat terpengaruh terhadap jumlah data yang diterima, sedangkan parameter kecepatan tidak terlalu terpengaruh.

DAFTAR ISI

Halaman

BAB I

PENDAHULUAN

1.1. Latar Belakang

Selama dekade terakhir, perkembangan tekonologi komunikasi WSN (Wireless Sensor Networks) begitu pesat dilihat dari beberapa konsep yang dibangun menggunakan WSN seperti, sistem peringatan dini terhadap banjir, sistem pemantuan suhu dan kelembaban pada lahan tanaman jarak. WSN merupakan infrastruktur jaringan nirkabel atau tanpa kabel yang menggunakan sensor untuk memantau kondisi fisik atau lingkungan yang dapat terhubung ke jaringan, jaringan ini menggunakan gelombang radio sebagai media pengirimannya.

Pada umumnya penerapan WSN menggunakan model statis , namun pada kondisi tertentu model statis tidak cocok untuk diterapkan seperti pada kondisi lingkungan yang berubah-ubah. Model statis dapat mengalami masalah seperti, tidak semua daerah dapat terpantau, beberapa pengaplikasian WSN yang membutuhkan sensor dengan harga yang mahal cukup merepotkan.

Dari beberapa kekurangan WSN itu dikembangkanlah sebuah model baru yaitu *Mobile Wireless Sensor Networks* (MWSN). Pada penelitian sebelumnya telah digambarkan tentang konsep MWSN (Javad Rezazadeh, 2012). Penelitian tersebut telah dibahas mengenai arsitektur MWSN, model dari mobilitas. Pada tugas akhir ini, akan dibuat sebuah protipe MWSN dengan menggunakan satu *node* sebagai coordinator, dua *node* sebagai BTS (*Base Transceiver Station*) dan 2 *mobile node*. selain itu sistem ini juga menggunakan protokol *handoff* untuk membantu perpindahan data pada *mobile node* serta juga dilengkapi *error checking* data untuk menjaga agar data yang diterima sesuai dan *error checking status* untuk mengetahui kondisi dari BTS.

Handoff atau *Hand Over* (HO) ialah proses perubahan pelayanan/peng-handle-an sebuah *Mobile Station* (MS) dari suatu *Base Station* (BS) ke BS lain dikarenakan adanya pergerakan MS yang menjauhi BS awal dan mendekati BS baru.

1.2. Rumusan Masalah

Berdasarkan latar belakang di atas, dapat dirumuskan permasalahan yaitu Bagaimana mendesain dan mengimplementasikan protokol *handoff* dan koreksi kesalahaan pada MWSN (*Mobile Wireless Sensor Networks*)?

1.3. Batasan Masalah

Batasan masalah dari pembahasan tugas akhir ini adalah :

1. Menggunakan modul mikrokontroler Arduino Uno.
2. Menggunakan modul Xbee *series* 2 sebagai komunikasi nirkabel.
3. Menggunakan *software* Arduino IDE untuk memprogram mikrokontroler pada Arduino Uno.
4. Menggunakan *software* X-CTU untuk menkonfigurasi modul Xbee *series* 2 dalam mode AT.
5. Bahasa pemrograman menggunakan bahasa C/C++ pada arduino IDE.
6. Masukan data berasal dari nilai keluaran potensio.
7. *Node* BTS diletakkan pada posisi yang telah ditentukan.
8. Koreksi kesalahan hanya pada *node coordinator* dan node BTS.

9. Untuk keperluan uji coba sistem, akan dilakukan pada daerah yang bebas dari penghalang.

1.4. Tujuan

Dalam rancang bangun prototipe MWSN ini bertujuan untuk :

Mendesain dan mengimplementasikan protokol *handoff* dan koreksi kesalahaan pada MWSN (*Mobile Wireless Sensor Networks*)

1.5. Sistematika Penulisan

Pembahasan Tugas Akhir ini secara garis besar tersusun dari 5 bab, yaitu diuraikan sebagai berikut :

BAB I. PENDAHULUAN

Pada bab ini akan dibahasa mengenai latar belakang masalah, batasan masalah, tujuan penulisan, dan sistematika penulisan.

BAB II. LANDASAN TEORI

Pada bab ini akan dibahas teori penunjang dari permasalahan, yaitu mengenai WSN, arsitektur MWSN, Protokol *handoff*, Arduino Uno, *software* arduino IDE, ZigBee (Xbee *series* 2) mode AT dan *software* X-CTU.

BAB III. METODE PENELITIAN DAN PERANCANGAN SISTEM

Pada bab ini akan dibahas tentang blog diagram sistem MWSN serta metode yang dilakukan dalam perancangan sistem MWSN, meliputi penerapan protokol

handoff pada MWSN, bagaimana menerapkan koreksi kesalahan pada *node coordinator* dan *node* BTS serta konfigurasi Xbee *series* 2 dalam mode AT pada *software* X-CTU.

BAB IV. HASIL DAN PEMBAHASAN

Pada bab ini akan dibahas mengenai pengujian pengaruh kecepatan *mobile node* dan jarak antara *mobile node* dengan *node* BTS terhadap paket *loss* dan pengujian aplikasi pada komputer apakah dapat menampilkan data yang telah dikirimkan oleh *mobile node* menggunakan *software visual basic*.

BAB V. KESIMPULAN DAN SARAN

Berisi kesimpulan yang didapat dari hasil penelitian berdasarkan rumusan masalah serta saran untuk perkembangan penelitian selanjutnya.

BAB II

LANDASAN TEORI

2.1 *Wireless Sensor Networks* (WSN)

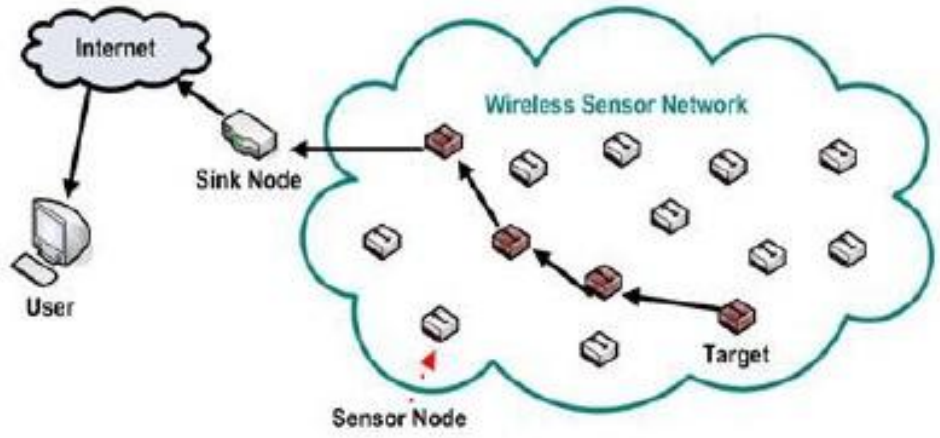
2.1.1 Konsep Dasar dan Pengertian *Wireless Sensor Networks*

Wilreless sensor networks adalah sebuah jaringan komunikasi sensor yang terhubung secara *wireless* / tanpa kabel untuk memantau kondisi lingkungan tertentu pada lokasi yang berbeda antara sensor dan pemrosesan datanya. Pada dasarnyaa jaringan komunikasi *wireless* ini digunakan pada industri ataupun aplikasi komersial lainnya yang kesulitan dengan pemasangan sistem perkabelan. Area penggunaan dari *wireless* sensor ini adalah seperti sistem pemantauan tingkat polusi atau kontaminasi udara, sistem deteksi kebakaran atau semburan panas bumi, *area habitat monitoring, object tracking, traffic monitoring* ataupun kondisi lainnya (Maribun,S. 2008).

Pada prinsipnya pembacaan kondisi oleh sensor ini akan diinformasikan secara realtime dan keamanan data yang terjamin hingga diterima oleh pengolah data. Beberapa karakteristik dari *wireless sensor* ini daiantaranya :

1. Dapat digunakan pada daya yang terbatas.
2. Dapat ditempatkan pada kondisi lingkungan yang keras.
3. Dapat digunakan untuk kondisi dan pemrosesan data secara *mobile*.
4. Mempunyai topologi jaringan yang dinamis, dengan sistem *node* yang heterogen.
5. Dapat dikembangkan untuk skala besar.

2.1.2 Arsitektur WSN

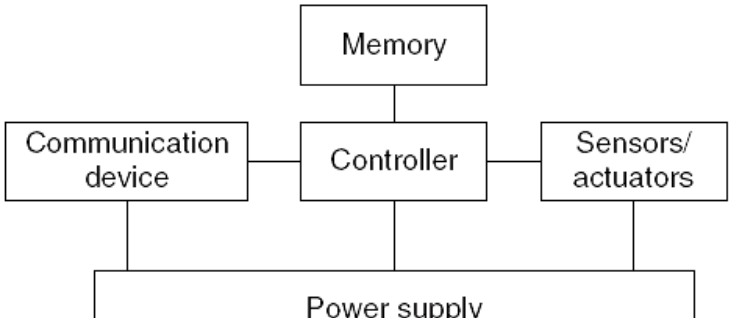


Gambar 2.1 Arstitektur WSN Secara Umum

Sumber : (http://telekom.ee.uii.ac.id)

Pada Gambar 2.1 dapat dilihat, *node* sensor disebar di sauatu area sensor. *Node* sensor tersebut memiliki kemampuan untuk merutekan data yang dikumpulkan ke *node* lain yang berdekatan. Data dikirimkan melalui transmisi radio kemudian diteruskan menuju *node* BS (*Base Station*) yang merupakan penghubung antara *node* sensor dan *user*. Informasi tersebut dapat diakses melalui berbagai *platform* seperti koneksi internet atau satelit sehingga memungkinkan *user* untuk dapat mengakses secara realtime melalui *remote coordinator* (http://digilib.ittelkom.ac.id).

Pada setiap *node* WSN terdiri dari 5 komponen yaitu : *controller ,memory, sensors /actuators, power supply* dan *communication device*. Komponen – komponen tersebut saling berkoordinasi, seperti yang ditunjukkan pada gambar 2.2 dibawah ini.



Gambar 2.2 Komponen-komponen Penyusun node WSN

1. *Communication Device* berfungsi untuk menerima dan mengirim data menggunakan protokol IEEE 802.15.4 atau IEEE 802.11b/g kepada *device* lain seperti *concentrator*, modem Wifi dan modem RF .
2. *Controller* berfungsi untuk melakukakn fungsi perhitungan, mengontrol dan memproses *device* yang terhubung dengan *controller*.
3. *Sensors/Actuators* berfungsi untuk men-sensing besaran-besaran fisis yang hendak diukur. *Sensor* adalah seuatu alat yang mampu untuk mengubah dari energy besaran yang diukur menjadi listrik yang kemudian diubah oleh ADC menjadi deratan pulsa trkuantisasi yang kemudian bisa dibaca oleh mikrokontroler.
4. *Power Supply* berfungsi sumber energy bagi sistem *Wireless Sensor* secara keseluruhan.
5. *Memory* berfungsi sebagai tambahan *memory* bagi sistem *Wireless Sensor*, pada dasarnya sebuah unit mikrokontroler memiliki *memory* sendiri.

2.2 *Mobile Wireless Sensor Networks* (MWSN)

2.2.1 Gambaran umum *Mobile Wireless Sensor Networks*

Pada umumnya penerapan WSN menggunakan sensor *node* model statis untuk memantau daerah yang ingin dipantau. Namun, karena kondisi lingkungan berubah-ubah, model statis ini mempunyai beberapa permasalahan seperti (Javad Rezazadeh, 2012). :

1. Penerapan WSN model statis tidak menjamin semua daerah cakupan dapat dipantau.
2. *Node* WSN menggunakan tenaga dari baterai, untuk beberapa *node* yang bertugas menjaga konketivitas jaringan sensor. Pada saat *node* ini mengalami kehabisan tenaga dapat berakibat terputusnya koneksi terhadap keseluruhan jaringan sensor.
3. Untuk beberapa pengaplikasian jaringan sensor yang membutuhkan sensor yang mahal sangat merepotkan.

2.2.2 Arsitektur *Mobile Wireless Sensor Network*

Pada arsitektur MWSN peran dari setiap *node* sangatlah penting maka dari itu,MWSN dapat dikategorikan berdasarkan peran mereka didalam jaringan (Javad Rezazadeh, 2012**).** :

1. **Mobile Embedded Sensor.** Sensor ini tidak dapat mengontrol pergerakan mereka sendiri melainkan dengan bantuan dari luar,misal sensor ditambatkan pada binatang.
2. **Mobile Actuated Sensor. S**ensor ini memiliki kemampuan untuk mengontrol pergerakan mereka sendiri, dengan kamampuan itu sensor dapat bergerak ke seluruh area dan memaksimalkan pembacaan sensor.
3. **Base Transcivier Station.** node ini berfungsi sebagai penghubung antara *mobile node* dengan *node* coordinator atau *base transcivier station* lainnya.
4. **coordinator.** node ini berfungsi sebagai pusat data.

1.2.3 Model Mobilitas

Model mobilitas yang dimaksud disini adalah bagaimana pergerakan dari *mobile sensor*. Model mobilitas ini dapat dibagi menjadi 2 tipe, (Javad Rezazadeh, 2012) :

1. **Random Way Model**. Model mobilitas ini *mobile sensor* dapat bergerak secara bebas namun tetap didalam daerah cakupan.
2. **Path Way Model.** Model mobilitas ini pergerakan *mobile sensor* sudah ditentukan dari awal.

2.3 Protokol *Handoff*

Handoff atau *Hand Over* (HO) ialah proses perubahan pelayanan/peng-handle-an sebuah *Mobile Station* (MS) dari suatu *Base Station* (BS) ke BS lain dikarenakan adanya pergerakan MS yang menjauhi BS awal dan mendekati BS baru (C, Andharini. 2008).

Mekanisme *Hand Over* dapat dibagi menjadi 2, yaitu :

1 ***Make Before Break***

Pada mekanisme ini, sebelum MS terhubung dan dilayani oleh BS yang baru, maka hubungan dengan BS lama tidak akan diputus. Hubungan dengan BS lama hanya akan diputus bila hubungan dengan agen baru dapat dilakukan.

2 ***Break Before Make***

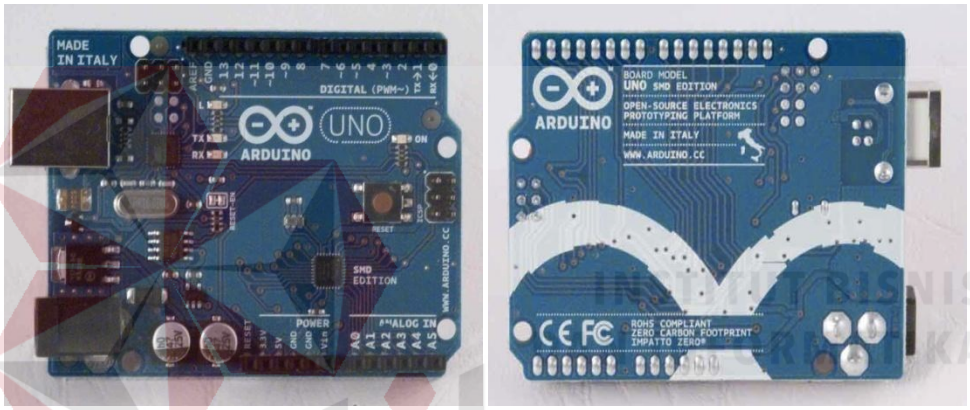
Pada mekanisme ini, MS akan memutuskan hubungan dengan agen lama walaupun hubungan dengan agen baru belum tercapai. Akibatnya aka nada suatu periode waktu yang singkat dimana MS tidak dilayani oleh agen manapun.

2.4 Arduino

Arduino adalah prototipe platform elektronik opensource yang terdiri dari mikrokontroler, bahasa pemrograman, dan IDE. Arduino adalah alat untuk membuat aplikasi interaktif, yang dirancang untuk mempermudah proyek pengguna (Banzi, 2009).

2.4.1 Arduino Uno SMD R3

Arduino Uno adalah papan mikrokontroler berbasis ATmega328. Dalam bahasa Italy "Uno" berarti satu, maka peluncuran arduino ini diberi nama Uno. Arduino ini berisi semua yang diperlukan untuk mendukung mikrokontroler, untuk mengaktifkan cukup menghubungkannya ke komputer dengan sebuah kabel USB atau mensuplainya dengan sebuah adaptor AC ke DC atau menggunakan baterai. (arduino.cc/ArduinoBoardUnoSMD, 2013)



Gambar 2.3 Arduino Uno SMD R3 Sisi Depan (Kiri) dan Belakang(Kanan)

Sumber : (arduino.cc, 2013)

Secara umum arduino terdiri dari dua bagian, yaitu:

1. *Hardware*: papan input/output (I/O)
2. *Software*: *software* arduino meliputi IDE untuk menulis program, driver untuk koneksi dengan komputer, contoh program dan *library* untuk pengembangan program. (Djuandi, 2011)

Berikut adalah Tabel 2.1 spesifikasi dari arduino uno smd R3:

Tabel 2.1 Spesifikasi Arduino Uno SMD R3

Mikrokontroler	ATmega328

Tegangan pengoperasian	5V
Tegangan input yang disarankan	7-12V
Batas tegangan input	6-20V
Jumlah pin I/O digital	14 (6 di antaranya menyediakan keluaran PWM)
Jumlah pin input analog	6
Arus DC tiap pin I/O	40 mA
Arus DC untuk pin 3.3V	50 mA
Memori Flash	32 KB (ATmega328), sekitar 0.5 KB digunakan oleh bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Arduino Uno dapat disuplai melalui koneksi USB atau dengan sebuah power suplai eksternal. Suplai eksternal (non-USB) dapat diperoleh dari sebuah adaptor AC ke DC atau baterai. Adaptor dapat dihubungkan dengan mencolokkan sebuah *center-positive* plug yang panjangnya 2,1 mm ke power jack dari *board*. Kabel lead dari sebuah battery dapat dimasukkan dalam *header*/kepala pin *Ground* (Gnd) dan pin Vin dari konektor *POWER*.

Board Arduino Uno dapat beroperasi pada sebuah suplai eksternal 6 sampai 20 Volt. Jika disuplai dengan yang lebih kecil dari 7 V, kiranya pin 5 Volt mungkin mensuplai kecil dari 5 Volt dan *board* Arduino Uno bisa menjadi tidak stabil. Jika menggunakan suplai yang lebih dari besar 12 Volt, *voltage* regulator bisa kelebihan panas dan membahayakan *board* Arduino Uno. *Range* yang direkomendasikan adalah 7 sampai 12 Volt. (arduino.cc/ArduinoBoardUnoSMD, 2013)

Pin-pin dayanya adalah sebagai berikut:

1. VIN. Tegangan input ke Arduino *board* ketika *board* sedang menggunakan sumber suplai eksternal (seperti 5 Volt dari koneksi USB atau sumber tenaga lainnya yang diatur). Kita dapat menyuplai tegangan melalui pin ini, atau jika penyuplaian tegangan melalui *power jack*, aksesnya melalui pin ini.
2. 5V. Pin output ini merupakan tegangan 5 Volt yang diatur dari regulator pada *board*. *Board* dapat disuplai dengan salah satu suplai dari DC *power jack* (7-12V), USB *connector* (5V), atau pin VIN dari *board* (7-12). Penyuplaian tegangan melalui pin 5V atau 3,3V mem*bypass* regulator, dan dapat membahayakan *board*. Hal itu tidak dianjurkan.
3. 3V3. Sebuah suplai 3,3 Volt dihasilkan oleh regulator pada *board*. Arus maksimum yang dapat dilalui adalah 50 mA.
4. GND. Pin *ground*.

2.4.3 Memori

ATmega328 mempunyai 32 KB yang bersifat *non-volatile*, digunakan untuk menyimpan program yang dimuat dari komputer. (dengan 0,5 KB digunakan untuk *bootloader*). ATmega 328 juga mempunyai 2 KB SRAM yang *volatile* (hilang saat daya dimatikan), digunakan oleh variable-variabel di dalam program. dan 1 KB EEPROM (yang dapat dibaca dan ditulis (RW/*read* and *written*). (arduino.cc/ArduinoBoardUnoSMD, 2013)

2.4.4 Input dan Ouput

Setiap 14 pin digital pada Arduino Uno dapat digunakan sebagai input dan output. Fungsi-fungsi tersebut beroperasi di tegangan 5 Volt. Setiap pin dapat memberikan atau menerima suatu arus maksimum 40 mA dan mempunyai sebuah resistor pull-up (terputus secara default) 20-50 kOhm. Selain itu, beberapa pin mempunyai fungsi-fungsi sebagai berikut:

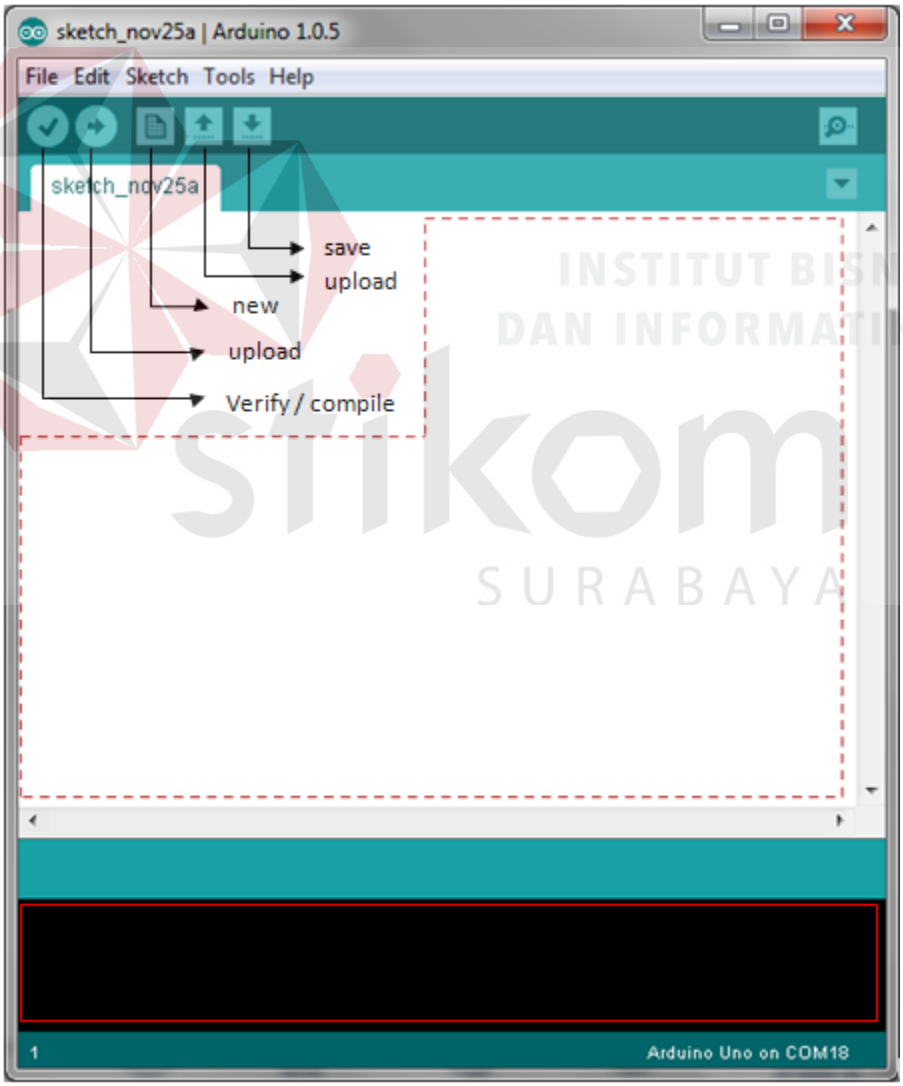
1. **Serial: 0 (RX) dan 1 (TX)**. Digunakan untuk menerima (RX) dan memancarkan (TX) serial data TTL (*Transistor-Transistor Logic*). Kedua pin ini dihubungkan ke pin-pin yang sesuai dari chip Serial Atmega8U2 USB-ke-TTL.
2. **External Interrupts: 2 dan 3**. Pin-pin ini dapat dikonfigurasikan untuk dipicu sebuah interrupt (gangguan) pada suatu nilai rendah, suatu kenaikan atau penurunan yang besar, atau suatu perubahan nilai.
3. **PWM: 3, 5, 6, 9, 10, dan 11**. Memberikan 8-bit PWM *output* dengan fungsi *analogWrite*().
4. **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK)**. Pin-pin ini men*support* komunikasi SPI menggunakan SPI *library*.
5. **LED: 13**. Ada sebuah LED yang terpasang, terhubung ke pin digital 13. Ketika pin bernilai *HIGH* LED menyala, ketika pin bernilai *LOW* LED mati.

Arduino UNO mempunyai 6 input analog, diberi label A0 sampai A5, setiapnya memberikan resolusi 10 bit. Secara default, 6 input analog tersebut mengukur tegangan dari *ground* sampai tegangan 5 Volt, dengan itu memungkinkan untuk mengganti batas atas dari rangenya dengan menggunakan pin AREF dan fungsi *analogReference*(). Di sisi lainnya, beberapa pin mempunyai fungsi spesifik yaitu pin A4 atau SDA dan pin A5 atau SCL. Mendukung komunikasi TWI dengan menggunakan *Wire library*. Ada sepasang pin lainnya pada board yaitu AREF referensi tegangan untuk input analog. Digunakan dengan *analogReference*(), dan reset untuk mereset mikrokontroler. (arduino.cc/ArduinoBoardUnoSMD, 2013)

2.5. Software Arduino IDE

Arduino IDE adalah *software* yang ditulis menggunakan java dan berdasarkan pengolahan seperti, avr-gcc, dan perangkat lunak *open source* lainnya (Djuandi, 2011). Arduino IDE terdiri dari:

1. *Editor program*, sebuah window yang memungkinkan pengguna menulis dan mengedit program dalam bahasa processing.
2. *Verify / Compiler*, sebuah modul yang mengubah kode program (bahasa processing) menjadi kode biner. Bagaimanapun sebuah mikrokontroller tidak akan bisa memahami bahasa processing, yang dipahami oleh mikrokontroller adalah kode biner.
3. *Uploader*, sebuah modul yang memuat kode biner dari komputer ke dalam memori mikrokontroller di dalam papan arduino.



Gambar 2.4 Tampilan Software Arduino IDE

Pada Gambar 2.4 terdapat *menu bar*, kemudian *toolbar* dibawahnya, dan sebuah area putih untuk *editing sketch*, area hitam dapat kita sebut sebagai *progress* area, dan paling bawah dapat kita sebut sebagai "*status bar*".

2.6. Bahasa Pemrograman Arduino

Arduino ini bisa dijalankan di komputer dengan berbagai macam *platform* karena didukung atau berbasis Java. *Source* program yang dibuat untuk aplikasi mikrokontroler adalah bahasa C/C++ dan dapat digabungkan dengan assembly. (arduino.cc/software, 2013)

2.7. Xbee Series *2* Chip Antenna *dan Xbee* Pro Series *2* Wire Antenna

Xbee *series* 2 modul RF dirancang untuk beroperasi dalam protokol ZigBee dengan biaya yang murah dan jaringan sensor nirkabel menggunakan daya yang rendah. Modul ini membutuhkan daya yang rendah dan dapat melakukan pengiriman data yang handal antara perangkat dengan jarak yang jauh. Modul ini beroperasi pada frekuensi 2.4 GHz. (Inc, XBee Series 2 OEM RF Modules, 2007)

Xbee *series* 2 ini mempunyai beberapa model antena, salah duanya adalah *chip antenna* dan *wire antenna*. *Chip antenna* merupakan suatu chip keramik yang terletak pada *board* modul Xbee, bentuknya lebih kecil. *Chip antenna* memiliki pola radiasi cardoid, yang artinya sinyal dilemahkan dalam berbagai arah dan sangat baik digunakan dalam area yang tidak terlalu besar atau kecil. Sedangkan *wire antenna* merupakan suatu antena kawat yang terletak pada board modul Xbee, *wire antenna* memiliki pola radiasi *omndirectional* yang artinya jarak transmisi maksimum hampir sama pada semua arah ketika antena tersebut tegak lurus terhadap modul. Gambar 2.7 merupakan gambar dari modul Xbee *series* 2 *chip antenna* dan Gambar 2.8 merupakan gambar dari modul Xbee *series* 2 *wire antenna*. (Faludi, 2011)

Gambar 2.7 Xbee Series 2 Chip Antenna

Gambar 2.8 Xbee Series 2 Wire Antenna

Berikut adalah spesifikasi dari modul Xbee pro *series* 2 *chip antenna* (Inc, XBee Series 2 OEM RF Modules, 2007):

1. Jarak jangkauan *indoor* 133 ft atau 40 meter.
2. Jarak jangkauan *outdoor line of sight* 400 ft atau 120 meter.
3. Transmit *power output* 2 mW (+ 3 dbm).
4. Radio Frekuensi data *rate* 250 Kbps.
5. Frekuensi 2.4 GHz.
6. *Receiver sensitivity* -98 dbm (1 % *pakcet error rate*).
7. Antena menggunakan *chip antenna.*

Berikut adalah spesifikasi dari modul Xbee pro *series* 2 *wire antenna* (Inc, 2012):

1. Jarak jangkauan *indoor* 300 ft (90 meter) dan 200 ft (60 meter)

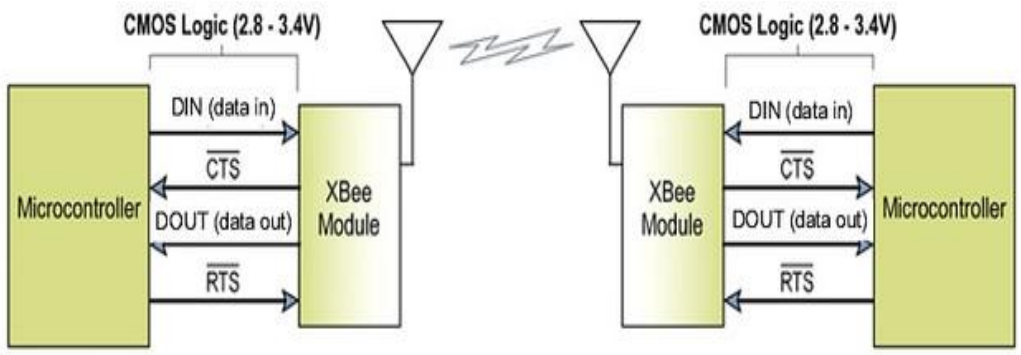
2. Jarak jangkauan *outdoor line of sight* 2 miles atau 3200 meter dan 5000 ft atau 1500 meter (variant lainnya).
3. Transmit *power output* 50 mW (+ 17 dbm).
4. Radio Frekuensi data *rate* 250 Kbps.
5. Frekuensi 2.4 GHz.
6. *Receiver sensitivity* -102 dbm.
7. Antena menggunakan *wire antenna.*

2.7.1 Mode Xbee AT/*Transparent*

Dalam mode transparent/AT, modul Xbee bertindak sebagai pengganti *serial line*. Semua data UART (*Universal Asincrhounus Recivier transmiter*) diterima melalui pin *input* akan ditransmisikan. Ketika data tersebut diterima maka data akan dikirimkan keluar (Xbee lainnya) melalui pin *output*. Data atau paket yang diterima bisa ditujukan ke satu sasaran (*point to point*) atau ke beberapa sasaran (*star/broadcast*). (Inc, 2007)

2.7.2 Komunikasi Serial Xbee *Series* 2

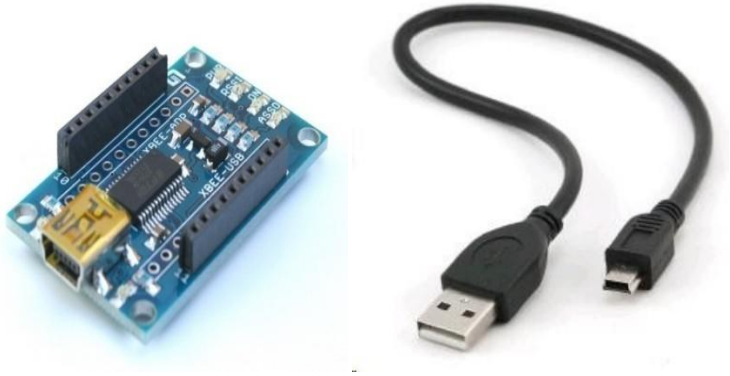
Xbee series 2 merupakan sebuah modul yang terdiri dari receiver dan tranmitter melalui port serial. Melalui port serial ini Xbee dapat berkomunikasi secara UART (*Universal Asincrhounus Recivier transmiter*). Gambar 2.9 menunjukkan diagram sistem aliran data secara UART. (Inc, 2007)



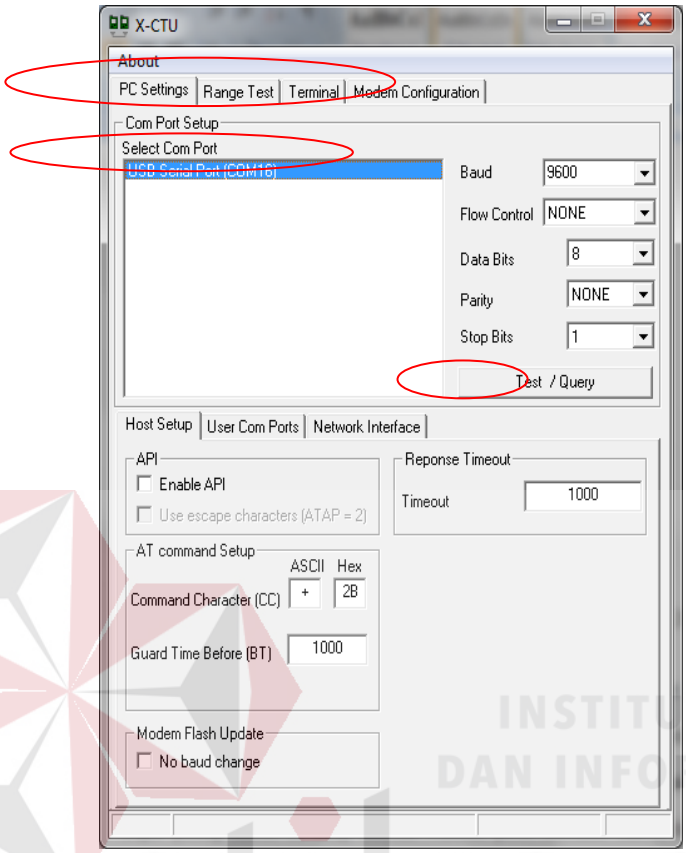
Gambar 2.9 Diagram Sistem Aliran Data UART pada Xbee

2.7.3 Xbee USB *Adapter* dan Software X-CTU

Xbee usb *adapter* (Gambar 2.10) merupakan alat untuk menghubungkan modul Xbee ke komputer dengan kabel mini usb (Gambar 2.10) dan selanjutnya dapat dikonfigurasi menggunakan *software* X-CTU (Gambar 2.11). *software* X-CTU merupakan *software* yang digunakan untuk mengkonfigurasi Xbee agar dapat berkomunikasi dengan Xbee lainya. Parameter yang harus diatur adalah PAN ID (*Personal Area Network*) ID yaitu parameter yang mengatur radio mana saja yang dapat berkomunikasi, agar dapat berkomunikasi PAN ID dalam satu jaringan harus sama. Xbee dapat berkomunikasi point to point dan point to multipoint (broadcast).



Gambar 2.10 Xbee Usb Adapter dan Kabel Mini Usb

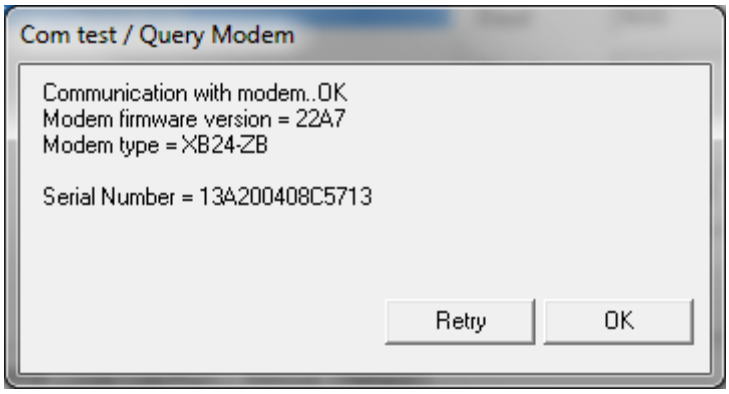


Gambar 2.11 Tampilan Software X-CTU

Pada Gambar 2.11 *software* X-CTU, terdapat empat tab di bagian atas program. Masing – masing tab mempunyai fungsi yang berbeda – beda (Inc, 2008). Berikut adalah ke empat tab tersebut:

1. *PC Settings*

   Pada tab ini mengijinkan pengguna untuk memilih *COM port* yang diinginkan dan mengkonfigurasi *port* tersebut sesuai pengaturan Xbee yang diinginkan. Terdapat tombol *Test* / *Query* pada tab *PC Settings*, tombol ini digunakan untuk menguji *COM port* yang telah dipilih, jika pengaturan dan *COM port* benar, maka akan muncul kotak dialog respon seperti pada Gambar 2.12.



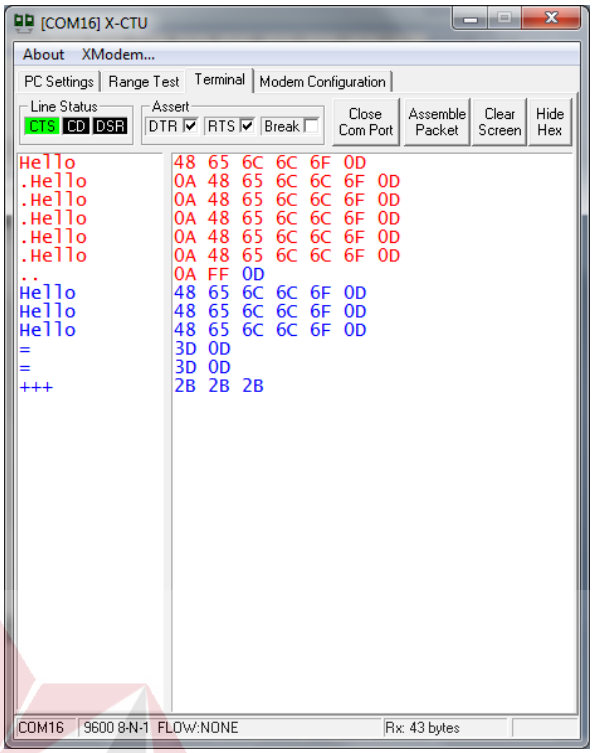
Gambar 2.12 Kotak Dialog Respon

2. *Range Test*

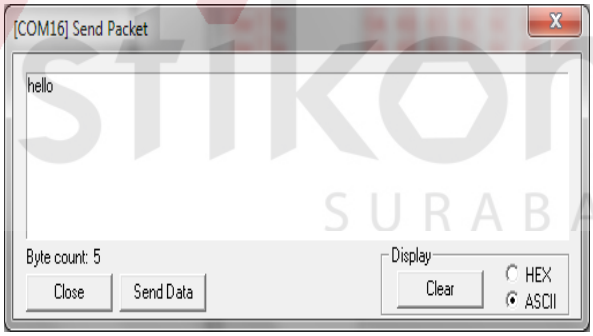
   Pada tab *range test* ini, pengguna dapat melakukan pengujian *range test* antara dua Xbee dengan mengirimkan paket data yang ditentukan pengguna dan memverifikasi apakah paket data yang dikirim sama dengan yang diterima.

3. *Terminal*

   Pada *tab* ini, pengguna memungkikan akses ke *COM port* komputer dengan program terminal *emulation*. Tab ini juga memungkinkan untuk mengakses *firmware* Xbee menggunakan AT *commands* dan dapat mengirim atau menerima data dalam format Hex dan ASCII dengan memilih *Assemble Packet* (Gambar.2.14).



Gambar 2.13 Tab Terminal



Gambar 2.14 Assemble Packet

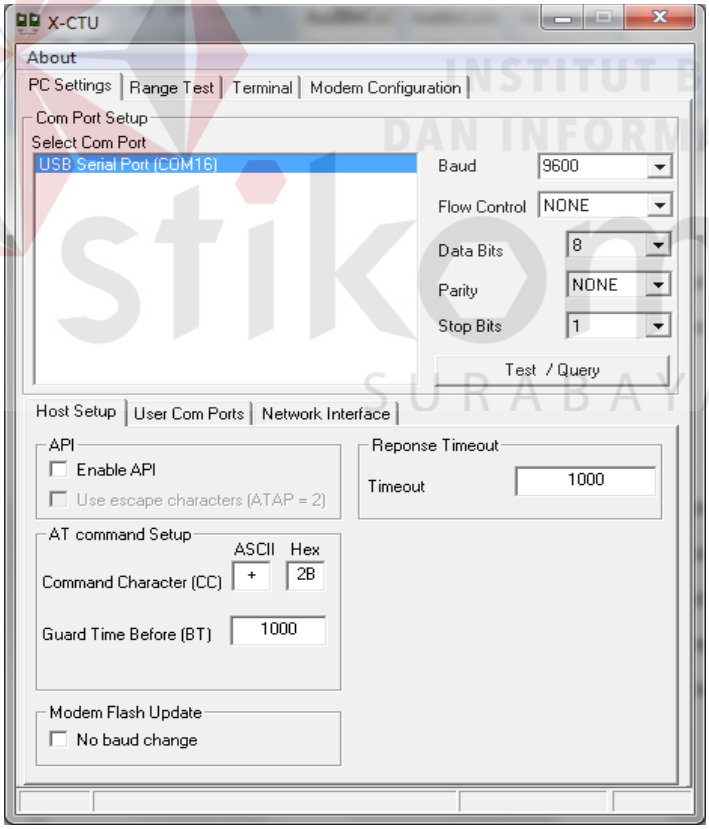
Pada Gambar 2.14 area putih dalam terminal *tab* ini berisi data informasi komunikasi yang terjadi antara 2 Xbee atau lebih. Teks yang berwarna biru merupakan teks yang telah diketik pengguna dan diarahkan ke *port* serial Xbee sedangkan teks merah merupakan data yang masuk dari *port* serial Xbee.

4. Modem Configuration Tab

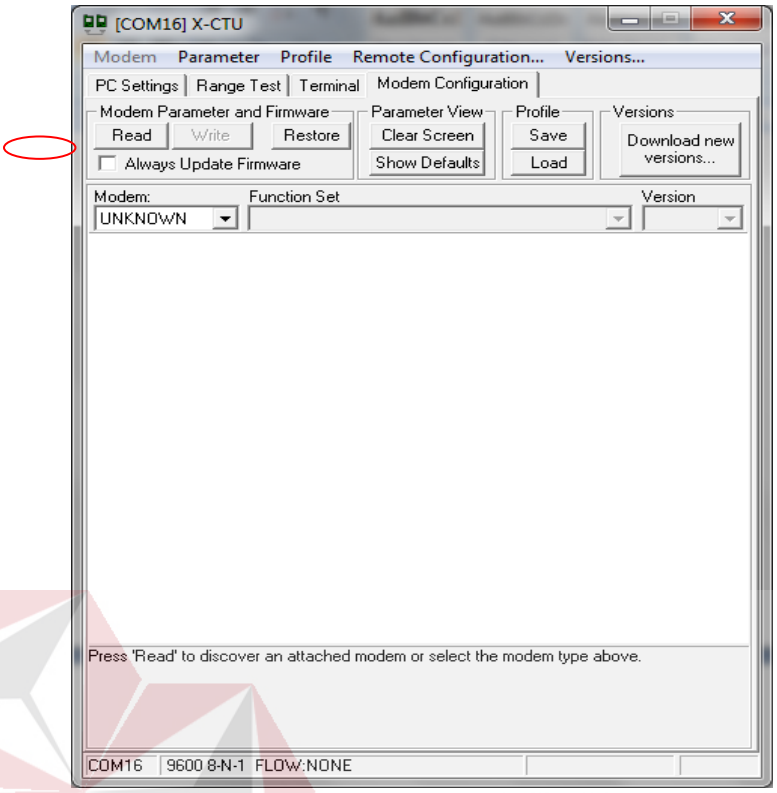
Pada tab ini, pengguna dapat melakukan pemrograman pada pengaturan *firmware* Xbee dan merubah versi *firmware*nya melalui *Graphical User Interface* (GUI). Terdapat 4 fungsi dasar dalam *modem configuration tab*, yaitu:

1. Menyediakan fasilitas GUI untuk mengatur *firmware* pada Xbee.
2. *Read* dan *write* firmware ke mikrokontroller Xbee.

Untuk dapat membaca (*read*) firmware Xbee, pertama hubungkan Xbee dengan Xbee usb adapter yang telah terhubung dengan kabel mini usb, kemudian hubungkan kabel tersebut ke komputer melalui *interface* usb. Jalankan aplikasi X-CTU pada komputer, selanjutnya atur *com port* pada tab PC *Settings* seperti Gambar 2.15. Pada *tab Modem Configuration*, klik "*Read*" pada bagian *Modem Parameter and Firmware* (Gambar 2.16).

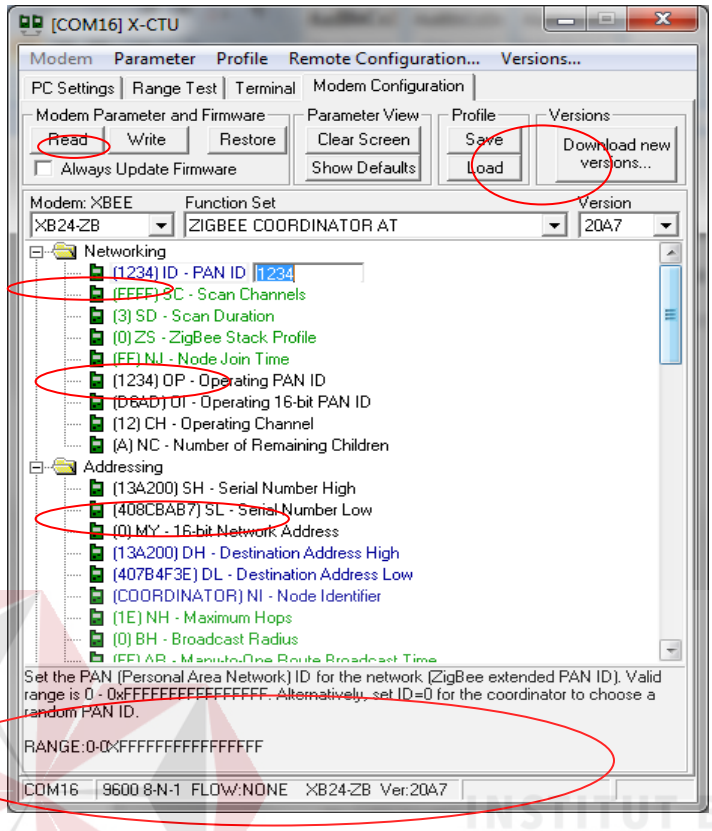


Gambar 2.15 Tab PC Settings



Gambar 2.16 Tab Modem Configuration

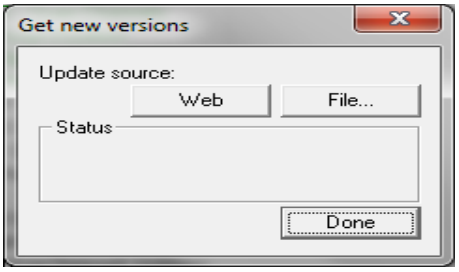
Setelah *firmware* Xbee telah terbaca, terdapat tiga warna dalam pengaturan konfigurasi (Gambar 2.17) yaitu hitam yang berarti *read-only* dan tidak bisa dirubah nilainya, kemudian hijau yang berarti nilai *default* Xbee dan biru yang berarti nilai yang pengguna tentukan sesuai keinginan. Untuk mengubah nilai parameter yang bisa dirubah, klik paramater tersebut kemudian ketik nilai yang baru sesuai keinginan pengguna. Untuk memudahkan pengisian nilai, terdapat deskripsi atau keterangan dalam pengisian nilai pada setiap parameter yang berada pada bagian bawah. Setelah semua nilai – nilai yang baru masuk, maka nilai tersebut dapat disimpan ke memori *non-volatile* pada Xbee. Klik tombol "*Write*" pada bagian *Modem Parameter and Firmware* untuk menyimpan konfigurasi parameter ke memori pada Xbee (Gambar 2.17).



Gambar 2.17 Firmware Xbee Telah Terbaca

3. Meng-*update firmware* pada Xbee

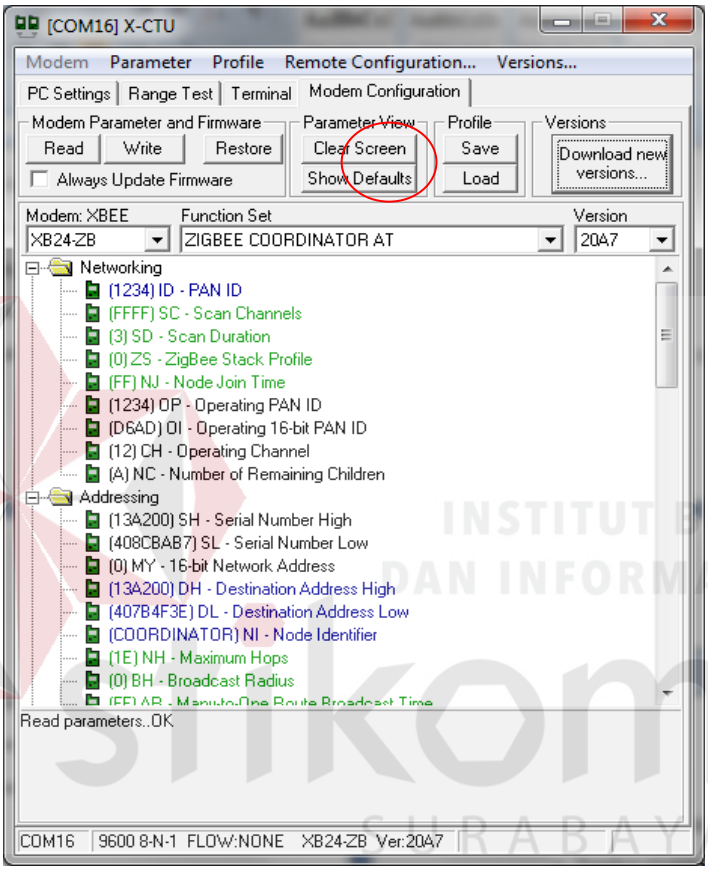
User atau pengguna dapat meng-*update firmware* pada Xbee baik melalui *web* atau menginstalnya dari *file.zip* atau *disk*. Klik "*Download New Versions*" pada bagian *Versions* (Gambar 2.17). Klik tombol *Web* jika ingin meng-*update file firmware* melalui *web* atau klik tombol *File* jika ingin meng-update melalui *file.zip* atau *disk* (Gambar 2.18).



Gambar 2.18 Kotak Dialog Get New Versions

4. *Save* atau *load modem profile*

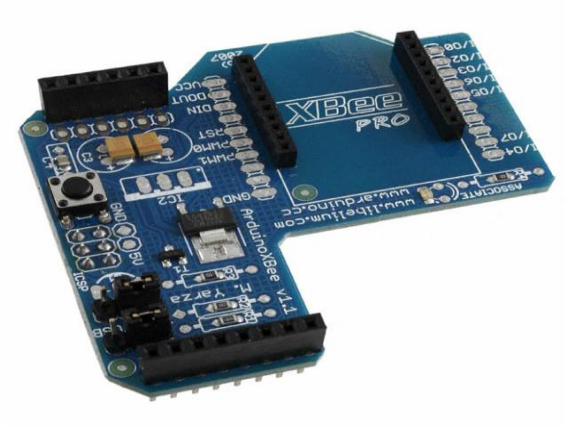
X-CTU dapat menyimpan dan men-*load* profil modem atau konfigurasi yang telah disimpan, ini sangat berguna ketika paramater konfigurasi yang sama ditetapkan pada beberapa Xbee yang berbeda (Gambar 2.19).



Gambar 2.19 Save dan Load Modem Profile

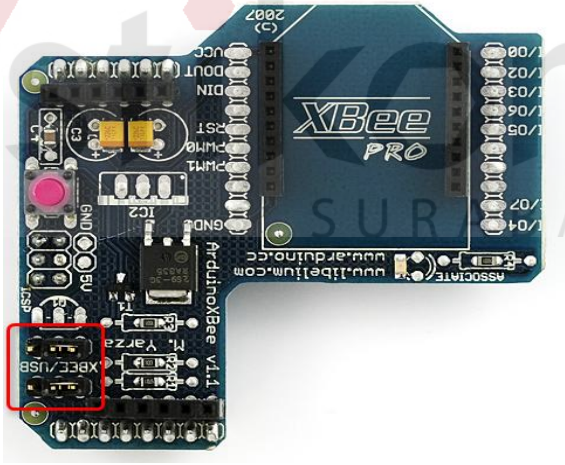
2.8. Xbee Shield

Xbee *shield* merupakan suatu *board* yang dapat menghubungkan *board* arduino untuk berkomunikasi secara nirkabel atau *wireless* menggunakan modul Xbee atau Zigbee (Gambar 2.20). (arduino.cc/ArduinoXbeeShield, 2013)



Gambar 2.10 Xbee Shield

Xbee *shield* memiliki dua jumper terbuat dari plastik yang dapat di *removeable* dari tiga pin pada *shield* yang berlabel Xbee/USB (Gambar 2.21). Jumper ini menentukan komunikasi serial Xbee agar terhubung pada komunikasi serial antar mikrokontroller atau USB pada *board* arduino. (arduino.cc/ArduinoXbeeShield, 2013)



Gambar 2.11 Jumper pada Xbee Shield

2.9. Parity Bit

Parity Bit adalah sistem pengecekan data yang pertama kali ada. Parity bit code menunjukan ganjil atau genapnya jumlah bit yang bernilai 1. Parity bit dibagi menjadi 2 jenis, yaitu even dan odd. Even parity akan menunjukkan nilai 1 bila jumlah bit 1 genap, sebaliknya odd parity akan memberikan nilai 1 jika jumlah bit satu ganjil.Parity bit code ini bebas diletakkan dimanapun sesuai kesepakatan penerima dan juga pengirim.

7 bits of data	Byte with parity bit code	
	even	odd
0101010	**0**0101010	**1**0101010
1001000	**1**1001000	**0**1001000
1010010	**0**1010010	**1**1010010
0101100	**0**0101100	**1**0101100
0000000	**1**0000000	**0**0000000
1111111	**0**1111111	**1**1111111

Gambar 12 Contoh *byte* dengan *parity bit code* pada awal *byte*

Parity bit memiliki kelemahan yang tidak dapat mendeteksi bila terjadi dua buah kesalahan sekaligus. Contohnya bila 1111111 terkirim sebagai 11011110 pada odd parity code yang seharusnya adalah 11111111. Hingga saat ini parity code masih digunakan pada perangkat SCSI dan PCI buses pada komputer. Penggunan dari parity code ini dikarenakan minimnya kesalahan yang dilakukan pada SCSI dan PCI buses saat mengcopi data dari memory. Error correction code yang hanya dapat mengecek sebuah kesalahan akan menjadi masalah untuk data yang panjang, sehingga penggunaan error correction code biasnya digunakan dengan cara membagi-bagi data ke dalam ukuran bit yang lebih kecil. Pengunaan parity bit code biasanya digunakan untuk setiap 7 bit memiliki satu parity code. 7 bit juga dapa disebut sebagai jarak kode. Semakin kecil jarak kode, maka kesalahan pun dapat dikurangi, sebaliknya untuk jumlah data yang sama, jumlah parity bit kode menjadi lebih banyak, dan ukuran data yang diterima pun lebih besar.

Pengulangan jarak selalu sama untuk setiap data yang dimasudkan sesuai dengan sistem yang sudah diperjanjikan sebelumnya. Kebanyakan dari error correcting code bertipe SECDED (single error correction and double error detecting). Untuk tiga buah kesalahan atau lebih dapat tidak terdeteksi (Tirtawinata,K).

BAB III

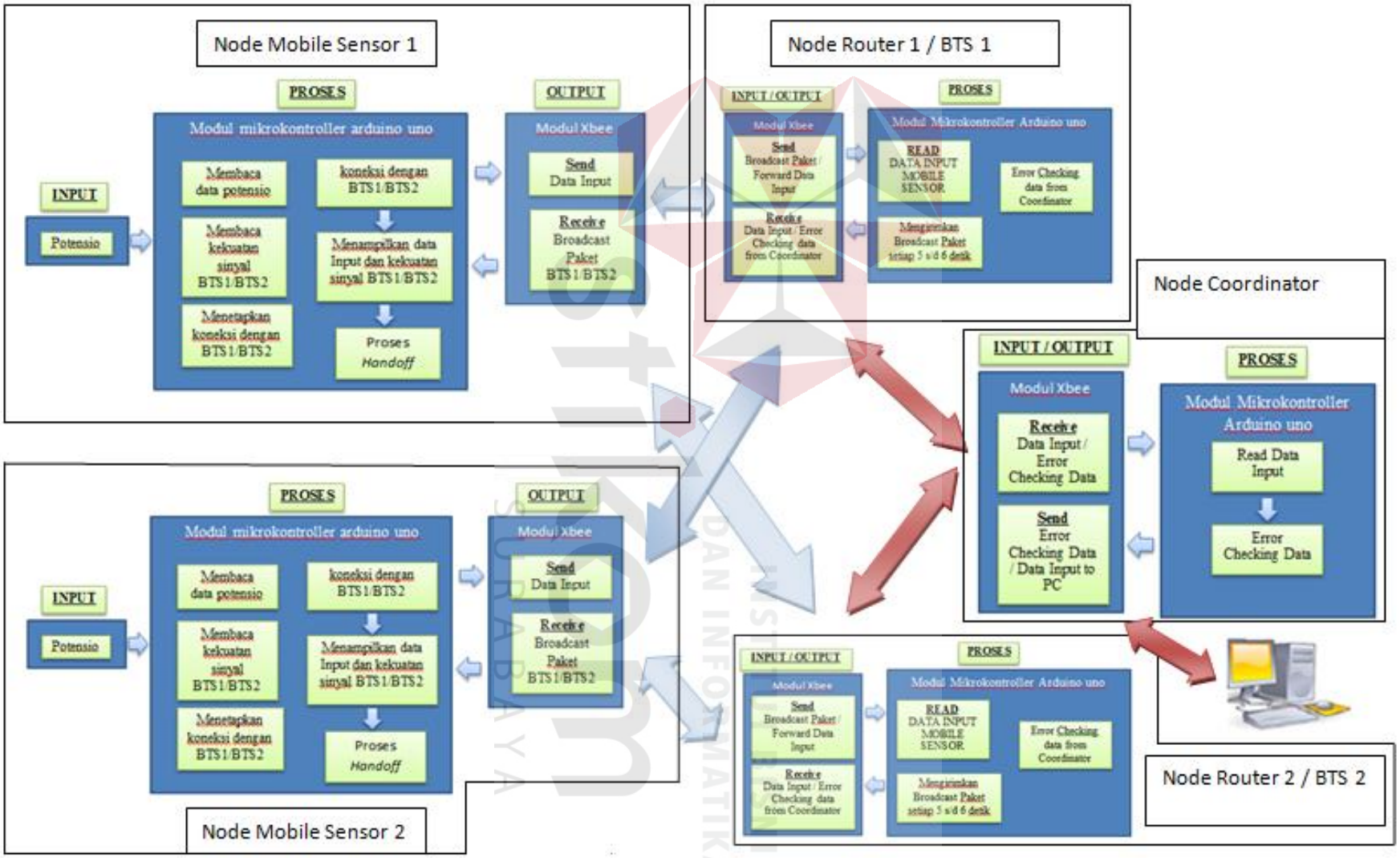
METODE PENELITIAN DAN PERANCANGAN SISTEM

1.1. Metode Penelitian

Metode peneltian yang digunakan pada tugas akhir ini melalui beberapa tahapan penelitian dan mencari informasi tentang data yang dibutuhkan dalam mengerjakan 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 arsitektur MWSN dan penerapan protokol *handoff* pada MWSN. Modul Xbee *series* 2 sebagai *receiver* dan *transmitter* data secara nirkabel antar node serta informasi dalam penerimaan dan pengiriman data.

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

Gambar 3.1 merupakan gambar blok diagram sistem yang merupakan penjelasan singkat dari perancangan sistem yang dibuat pada tugas akhir "Desain dan Implementasi Protokol *Handoff* dan Koreksi Kesalahan pada Jaringan MWSN".



Gambar 3.1 Blok diagram sistem

1.1.1 *Input* Data

Pada *node mobile* sensor 1 dan 2, *input* didapat dari nilai keluaran dari potensio yang dihubungkan ke pin analog arduino agar rentang nilai potensio dapat lebih banyak. Data tersebut yang akan diproses dan dikirimkan ke *node* BTS 1 atau BTS 2 dan diteruskan ke *node coordinator* dan dibaca oleh aplikasi pada PC.

3.1.2 Bagian Proses

Pada bagian proses dilakukan oleh 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.

Pada bagian proses *node* BTS 1 dan BTS 2 mengirimkan *broadcast packet* setiap 1 detik. *Broadcast packet* ini digunakan oleh *mobile node* sebagai penetapan koneksi. Selain itu digunakan untuk membaca kekuatan sinyal dari *node* BTS 1 atau *node* BTS 2. Sedangkan pada *node coordinator* digunakan sebagai pembacaan kondisi dari *node* BTS 1 dan *node* BTS 2 apakah dalam kondisi aktif atau tidak aktif. Pada sistem ini pembacaan kekuatan sinyal dari *node* BTS 1 dan *node* BTS 2 memanfaatkan pin 6 pada Xbee *serires* 2. Pin 6 pada Xbee dapat digunakan sebagai RSSI (*Recieve Signal Strength Indicator*) PWM (*Pulse Width Modulation*) dan sebagai *digital input*/*output*(ZigBee RF Modules,2013).Pada sistem ini pin 6 difungsikan sebagai RSSI PWM, keluaran dari PWM tersebut dapat direpresentasikan sebagai kekuatan sinyal(ZigBee RF Modules,2013). Kemudian pada arduino nilai PWM dirubah menjadi bilangan positif *integer* menggunakan fungsi *pulseIn*. Cara kerja fungsi ini membaca saat *pulse* (pada saat HIGH atau LOW) pada pin 6 (arduino.cc,2014). Kemudian nilai tersebut dapat digunakan dalam proses *handoff* pada *mobile node*. Berikut ini adalah mekanisme penetapan koneksi dan proses

handoff pada *mobile node* : pada saat *mobile node* berada pada kawasan BTS 1 atau BTS 2, *mobile node* akan mendapat *broadcast packet*. *Packet* tersebut digunakan oleh *mobile node* sebagai

Koneksi	Nilai PWM (Output pulseIn)	Handoff

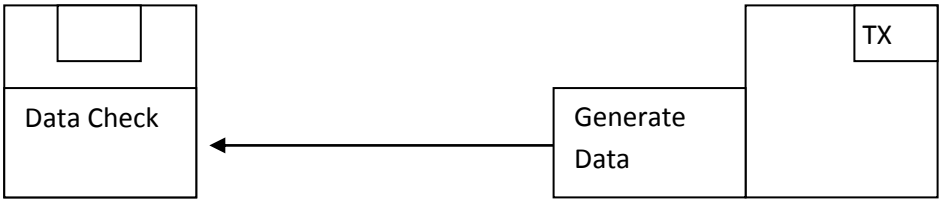
penetapan koneksi bahwa *mobile sensor* tersebut telah terkoneksi dengan BTS 1 atau BTS 2 langkah selanjutnya *mobile node* akan membaca berapa nilai kekuatan sinyal dari BTS 1 atau BTS 2.

Pada saat *mobile node* hanya terkoneksi dengan satu BTS data akan dikirimkan ke BTS tersebut. Apabila *mobile node* terkoneksi dengan dua BTS, *mobile node* akan membandingkan nilai PWM kedua BTS tersebut. BTS mana yang mempunyai nilai PWM mendekati nol, *mobile node* akan mengirimkan pesan data melalui BTS tersebut. Apabila kedua BTS tersebut mempunyai nilai yang sama *mobile node* akan memilih BTS mana yang pertama kali terkoneksi. Tabel 3.1 adalah tabel kebenaran koneksi dan *handoff* pada *mobile node*.

Tabel 3.1 Tabel kebenaran koneksi dan handoff pada mobile node

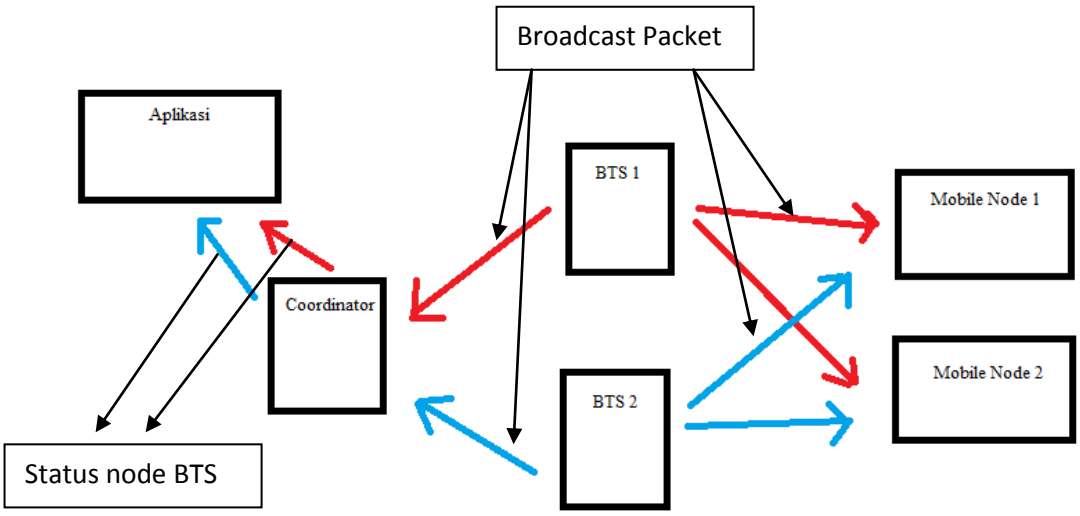
Hanya BTS1	Tidak ada perbandingan	Kirim Ke BTS 1
Hanya BTS2	Tidak ada perbandingan	Kirim Ke BTS 2
BTS1 & BTS2	BTS1 > BTS2	Kirim Ke BTS 2
BTS1 & BTS2	BTS1 < BTS2	Kirim Ke BTS 1
BTS1 & BTS2	BTS1 = BTS2	Pertama kali terkoneksi (BTS 1 / BTS 2)

Gambar 3.2 merupakan gambaran dari proses *error checking* data.



Gambar 3.2 Proses Error Checking data

Pada Gambar 3.2 Proses *Error Checking data* pada sistem ini menggunakan logika *even parity bit* dimana penambahan *bit parity* berdasarkan jumlah dari bit yang bernilai "1" adalah ganjil. Pada pengirim *bit parity* disisipkan pada akhir data kemudian pada penerima *bit parity* akan dicek jika sama dengan *bit parity* yang dikirimkan data benar dan dapat diteruskan ke *node* selanjutnya jika data yang diterima salah penerima akan mengirimkan *request packet* untuk mengirimkan data lagi dan data yang sebelumnya akan dibuang. Proses ini dilakukan setiap melakukan komunikasi antar *node*. Gambar 3.3 merupakan gambaran dari proses *error checking status* pada *node coordinator*.



Gambar 3.3 Proses error checking status

Broadcast packet yang dikirimkan oleh *node* BTS pada *mobile node* digunakan sebagai pembacaan kekuatan sinyal, sedangkan pada *node coordinator* digunakan sebagai pesan *status*. *Status* yang dimaksud disini adalah kondisi dari *node* BTS 1 atau BTS 2 dalam kondisi aktif atau tidak aktif.

3.1.3 *Output*

Setelah data dari potensio didapat oleh *mobile node.mobile node* hanya akan mengirimkan data ke *node BTS mobile node* tidak akan bisa langsung mengirimkan ke *node coordinator*. kemudian pada *node* BTS akan meneruskan pada *node coordinator*. Pada *node coordinator* akan mengirimkan ke PC/laptop dan ditampilkan pada aplikasi.

3.2. Perancangan Sistem

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

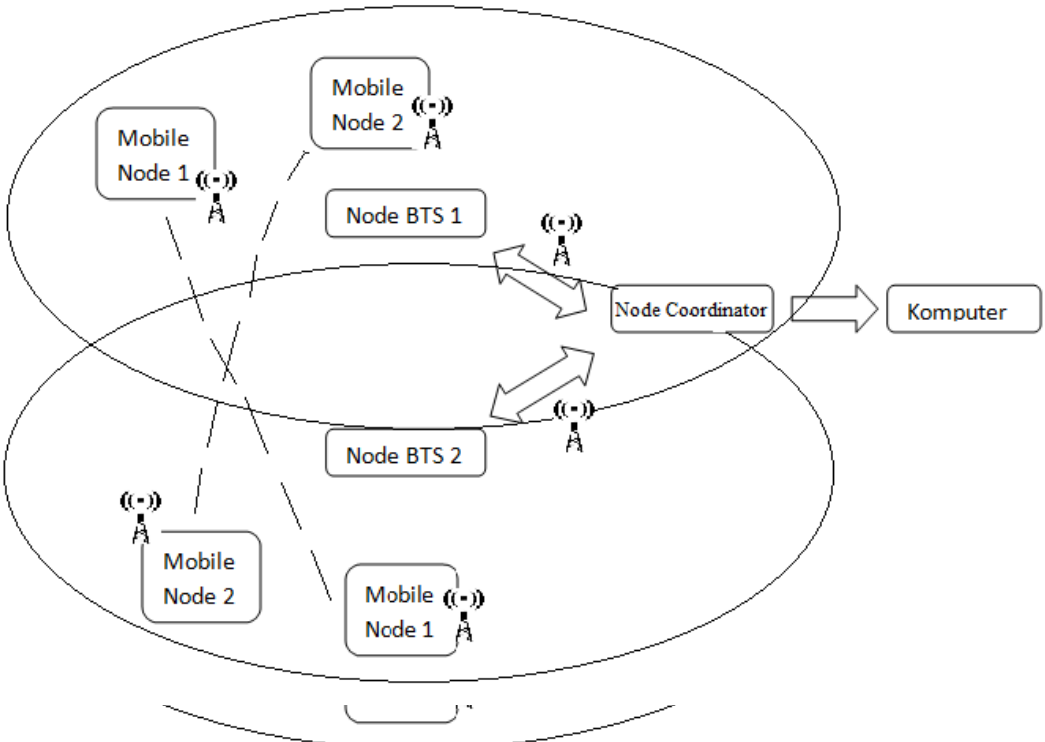
TAHAP 1	TAHAP 2	TAHAP 3
Desain Topologi 1. Menentukan topologi 2. Menentukan jumlah *node* 3. Menentukan jumlah *mobile node, node BTS* 4. Menentukan ID data dari masing - masing *node*	*Program Software* 1. Inisialisasi 2. Membuat skrip pembacaan data dari masing-masing *node* 3. Menyeleksi data yang masuk 4. Membuat skrip pembacaan potensio 5. Membuat algoritma *error checking* data 6. Membuat algoritma protokol *handoff* 7. Pengiriman data potensio 8. Membuat aplikasi pada komputer	Cek Protokol dan *Error Checking* 1. Memastikan protokol *handoff* berjalan dengan baik pada *mobile node* 2. Memastikan *error checking* data pada *node* BTS
Hardware Menentukan jumlah *hardware* yang dibutuhkan meliputi 1. Potensio 2. Modul Mikrokontroler Arduino Uno 3. Modul Xbee *series* 2	*Running Program* 1. *Compile program* 2. *Upload program* ke modul mikrokontroler arduino uno 3. Menampilkan data yang dikirim dan diterima pada setiap *node*	Aplikasi 1. Membandingkan data potensio yang dikirimkan oleh *mobile node* dengan data yang diterima oleh aplikasi

Gambar 3.4 Bagan Tahapan Proses Perancangan Sistem

Gambar 3.4 merupakan bagan proses perancangan sistem yang akan dilakukan. Pada bagan ini proses sistem dibagi ke dalam tiga kelompok . Tahap pertama yang dilakukan adalah menyiapkan *hardware* yang dibutuhkan oleh sistem dan membuat desain topologi. Tahap kedua yaitu membuat skrip untuk arduino menggunakan *software* arduino IDE dan membuat skrip aplikasi pada komputer dengan menggunakan *software visual basic*. Pada tahap ketiga melakukan pengecekkan terhadap apakah sistem berjalan dengan benar.

3.3. Desain Topologi

Pada perancangan sistem ini menggunakan model topologi *point to multipoint*. Model topologi ini memungkinkan *node* untuk berkomunikasi dengan 2 atau lebih namun tetap dalam 1 jaringan. Pada *mobile node* dapat berkomunikasi dengan *node* BTS 1 dan *node* BTS 2 kemudian *node* BTS 1 atau 2 dapat berkomunikasi dengan *mobile node* 1 atau *mobile node* 2 dan *node coordinator* dapat berkomunikasi dengan *node* BTS 1 dan *node* BTS 2 Gambar 3.3 merupakan topologi *point to multipoint* pada perancangan sistem MWSN.



Gambar 3.5 Topologi Point to Multipoint

1. Jumlah *node* yang digunakan berjumlah 5 buah, yaitu *mobile node* 2 buah , *node* BTS 2 buah dan *node coordinator* 1 buah.

2. *mobile node* 1 akan mulai mengirimkan data potensio ketika sudah memasuki kawasan BTS 1 atau 2. *Mobile node* dapat bergerak melewati *node* BTS 1 terlebih dahulu atau *node* BTS 2 namun *mobile node* tidak bisa mengirim data langsung ke *node coordinator*.

3. *mobile node* 1 akan mulai mengirimkan data potensio ketika sudah memasuki kawasan BTS 1 atau 2. *Mobile node* dapat bergerak melewati *node* BTS 1 terlebih dahulu atau *node* BTS 2 namun *mobile node* tidak bisa mengirim data langsung ke *node coordinator*.

4. *node* BTS 1 berada pada posisi yang telah ditentukan dan tidak bergerak. fungsi node ini untuk menerima data dari *mobile node* yang kemudian diteruskan ke *node coordinator*.

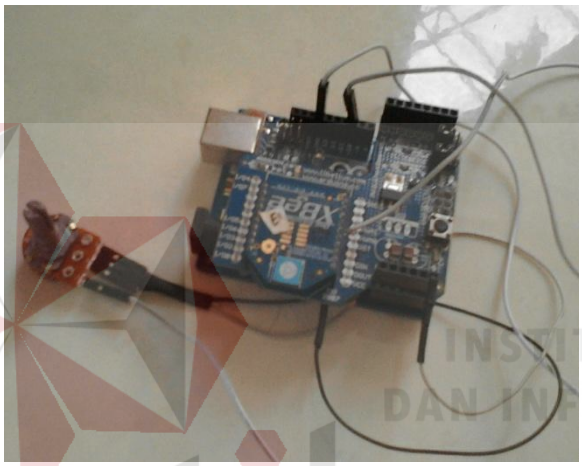
5. *node* BTS 2 berada pada posisi yang telah ditentukan dan tidak bergerak. fungsi node ini untuk menerima data dari *mobile node* yang kemudian diteruskan ke *node coordinator*.

6. *node coordinator* menerima data dari *node* BTS

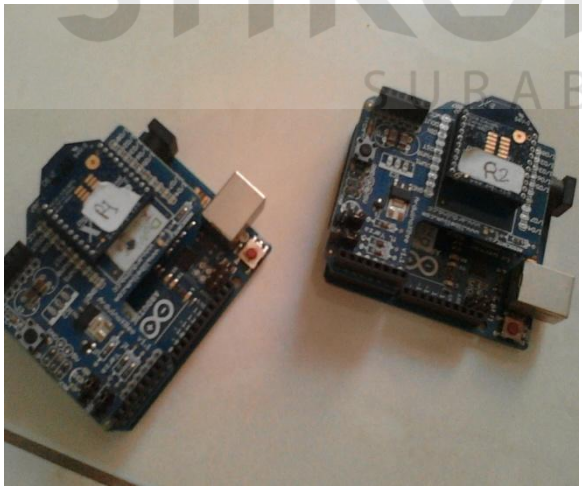
7. Komputer akan menerima data dari *node coordinator* melalui kabel serial, selanjutnya akan menampilkannya melalui program yang telah dibuat.

3.4 Hardware (Perangkat Keras)

Pada *mobile node* 1 dan 2 perangkat keras yang dibutuhkan yaitu potensio hanya digunakan untuk membuat data, modul mikrokontroler arduino uno untuk memproses data, modul xbee *series* 2 untuk mengirim dan menerima data secara nirkabel, modul xbee *shield* sebagai penghubung antara modul ardino uno dengan Xbee *series* 2 (Gambar 3.6). Sedangkan pada *node* BTS dan *node coordinator* perangkat keras yang dibutuhkan terdiri dari modul mikrokontroler arduino uno, Xbee *series* 2 dan Xbee *shield* (Gambar 3.7).



Gambar 3.6 Mobile node



Gambar 3.7 Node BTS dan node coordinator

Pada *mobile node* (gambar3.6) pin 6 pada xbee *series* 2 di hubungkan ke pin digital 10 arduino, kemudian pin analog 5 digunakan untuk potensio , pin GND dan 5 v digunakan oleh potensio.

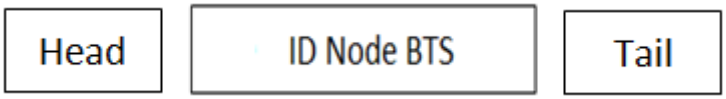
3.5. Pemrograman mikrokontroler Arduino Uno pada Software Arduino IDE

3.5.1 Format Penulisan Pesan

Pada setiap *node* dalam MWSN ini akan mengirimkan pesan dari satu *node* ke *node* lainnya. Format penulisan pesan pada *setiap node* diatur seperti Gambar 3.6.

Gambar 3.8 Format Penulisan Pesan Mobile Node

Pada *mobile node* 1 jika data akan dikirim ke BTS 1 dengan *header* "%" namun jika data akan dikirim ke BTS 2 dengan *header* "&". *Tailer* pada *mobile node* 1 dan 2 dengan "@". Pada *node* BTS 1 , 2 *header* akan diubah menjadi "!" dimaksudkan untuk setiap pesan dengan *header* itu ditujukan ke *node coordinator*. Selain itu pada *node* BTS 1 dan 2 juga mengirimkan *broadcast packet* yang terjadi setiap 5 s/d 6 detik, format penulisan pesan diatur seperti gambar 3.7.

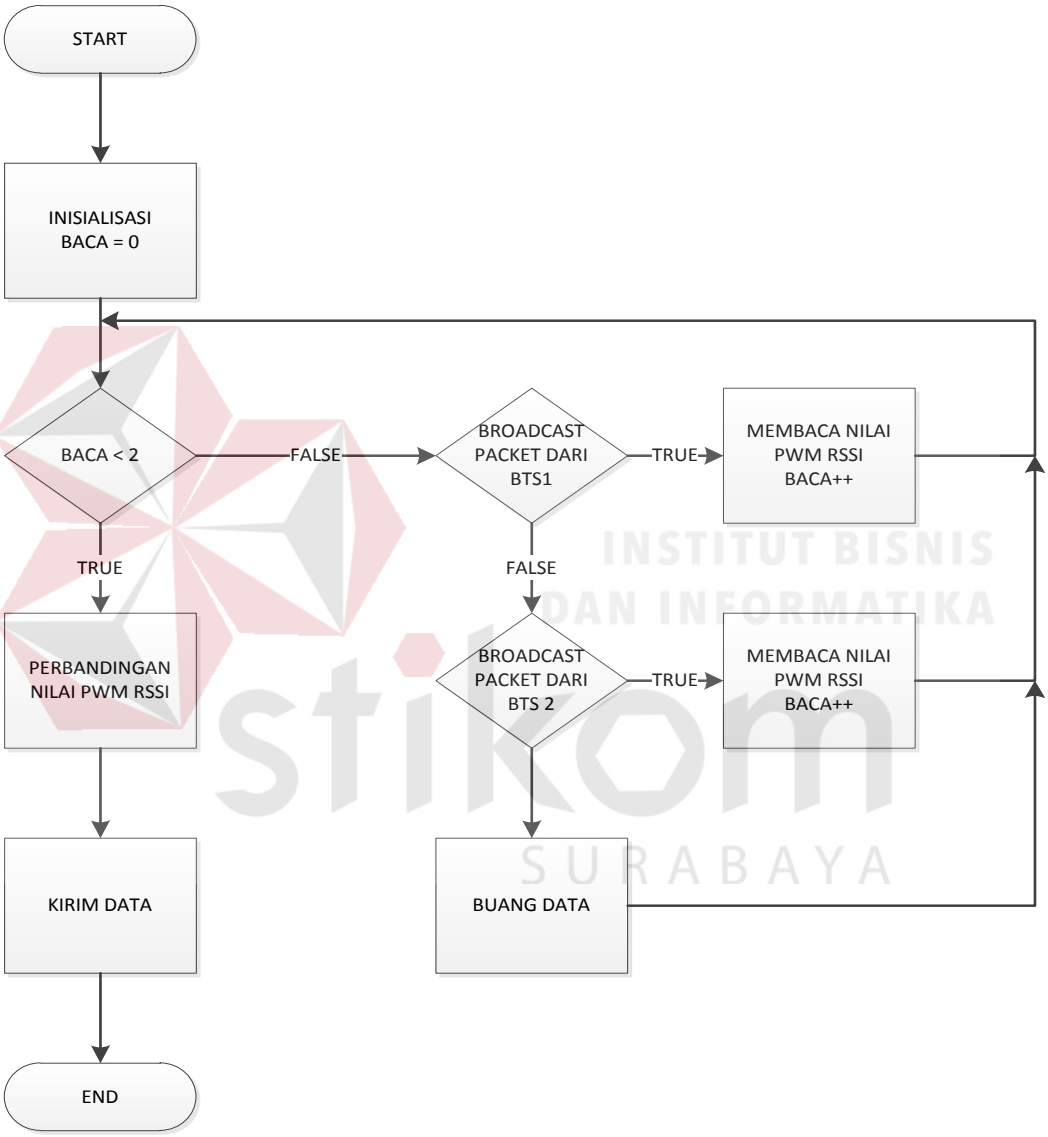


Gambar 3.9 Format penulisan broadcast packet

Pada *node* BTS 1 dan 2 menggunakan *header* "$" dengan *tailer* "*". Pada *node coordinator* untuk proses *error checking* data *header* dirubah menjadi "-" pada *node* BTS apabila data tersebut sama dengan data sebelumnya *header* akan dirubah menjadi "+" dan kemudian pada *node coordinator* akan ditampilkan pada aplikasi. Format penulisan pesan dibuat seperti ini

agar memudahkan setiap *node* dalam pembacaan pesan dan memudahkan untuk mengenali dari mana data berasal dan kemana data akan dikirim.

3.5.2 Skrip untuk *Mobile Node*



Gambar 3.10 Diagram alir pembuatan skrip mobile node

Berikut ini penjabaran gambar 3.10 Skrip ini digunakan untuk *mobile node* 1 dan 2.

1. Inisialisasi

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

```
//inisialisasi variabel
//pin analog yang digunakan potensio
int potensio=5

int koneksi_R1=0
int koneksi_R2=0;
long rssi_R1;
long rssi_R2;
String buff;
//pin digital yang digunakan untuk membaca rssi dari xbee series
2
int rssi_pin=10;

int baca=0;
int R1_flag=0;
int R2_flag=0;
int val=0;

//inisialisasi serial
void setup()
{
     Serial.begin(9600);
     pinMode(rssi_pin,INPUT);
}
```

2. Pembacaan Data *Node*

Pembacaan data dilakukan secara berulang-ulang oleh masing - masing *node* ketika ada data yang dikirmkan 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()
{
     if(Serial.available()>0)
     {
          char data_masuk=(char)Serial.read();
          buff += data_masuk;
          if(data_masuk == '\n') {}
     }
}
```

3. Menyeleksi Pesan dari *Node* BTS 1 dan 2

Setelah pesan yang diakhiri karakter '\n' diterima dan pesan tersebut berasal dari *node BTS* 1 atau 2 yang melakukan *broadcast packet* ($R1* / $R2*), maka *mobile node* akan melakukan penyelesksian. Berikut skrip penyeleksian pesan.

```
void loop()
{
   if(baca<=1)
   {
    if(Serial.available()>0)
    {
      char data_masuk=(char)Serial.read();
      buff += data_masuk;

      if(data_masuk == '\n')
      {
       if(buff[3]=='*')
       {
        //dari BTS 1
        if((buff[0]='$') && (buff[2]=='1'))
        {
          //pembacaan nilai RSSI dari node BTS 1
          long R1=pulseIn(rssi_pin,LOW,500);
          //nilai RSSI disimpan pada variabel RSSI_R1
          rssi_R1=R1;
          //BTS 1 terkoneksi dengan mobile node
          koneksi_R1=1;
          Serial.print("R1: ");
          Serial.println(rssi_R1);
          buff="";
          baca++;
          if(R2_flag==0)
          {
            R1_flag=1;
          }
          else R1_flag=0;
        }
        //dari BTS 2
        else if((buff[0]='$') && (buff[2]=='2'))
        {
          //pembacaan nilai RSSI dari node BTS 2
          long R2=pulseIn(rssi_pin,LOW,500);
```

```
            //nilai RSSI disimpan pada variabel RSSI_R1
            rssi_R2=R2;
            //BTS 1 terkoneksi dengan mobile node
            koneksi_R2=1;
            Serial.print("R2: ");
            Serial.println(rssi_R2);
            buff="";
            baca++;
            if(R1_flag==0)
            {
              R2_flag=1;
            }
            else R2_flag=0;
          }
        }
      }
    }
  }
}
```

Pembacaan hanya dilakukan dua kali untuk membandingan nilai PWM RSSI dari BTS 1 dan 2 (Tabel 3.1). Setelah dilakukan pembacaan dua kali akan dilakukan perbandingan. proses perbandingan nilai PWM RSSI inilah *handoff* pada sistem ini. Berikut adalah skrip untuk melakukan perbandingan.

```
void banding()
{
     if(koneksi_R1==1 && koneksi_R2==0)
    {
        kirim_data1();
    }

    if(koneksi_R2==1 && koneksi_R1==0)
    {
        kirim_data2();
    }
    if(koneksi_R1==1 && koneksi_R2==1)
    {
        if(rssi_R1<rssi_R2)
        {
            kirim_data1();
        }
```

```
        if(rssi_R1>rssi_R2)
        {
            kirim_data2();
        }
        if(rssi_R1==rssi_R2)
        {
            if(R1_flag==1)
            {
               kirim_data1();
            }
            if(R2_flag==1)
            {
               kirim_data2();
            }
        }
    }
}
```

4. Mengirim pesan ke *node* BTS

Setelah melakukan perbandingan pesan akan dikirimkan ke *node* BTS 1 atau 2, berikut adalah skrip untuk mengirimkan pesan.

```
void kirim_data1() // kirim ke Node BTS 1
{
      val=analogRead(potensio);
      gen_biner(val);
      simpan1+="%E1.R1.";
      simpan1+=String(biner);
      for (int y=1;y<12;y++)
      {
          simpan1+=biner[y];
      }
      simpan1+="@";
      koreksi1=simpan1;
      Serial.println(simpan1);
      //Serial.flush();
      simpan1="";
      ulang();
}

void kirim_data2() // kirim ke Node BTS 2
{
      //delay(1000);
      val=analogRead(potensio);
      gen_biner(val);
```

```
      simpan2+="&E1.R2.";
      //simpan2+=String(biner);
      for (int y=1;y<12;y++)
      {
          simpan2+=biner[y];
      }
      simpan2+="@";
      koreksi2=simpan2;
      Serial.println(simpan2);
      //Serial.flush();
      simpan2="";
      ulang();
}
```

Setelah itu semua variabel kembali di set sesuai inisialisasi, berikut adalah skripnya.

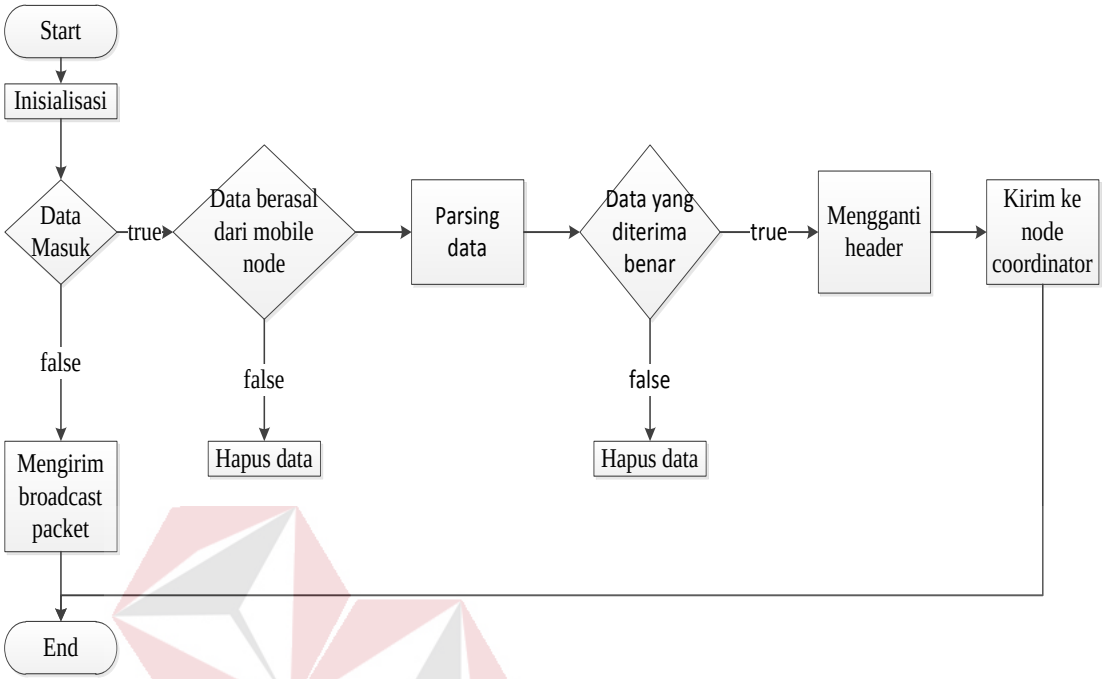
```
void ulang()
{
    koneksi_R1=0;
    koneksi_R2=0;
    baca=0;
    R1_flag=0;
    R2_flag=0;
    buff="";
    sisa_bagi=0;
    sisa=0;
    jml_1=0;
    pariti=0;
}
```

Berikut ini adalah fungsi tambahan untuk keperluan *error checking*.

```
void gen_biner(int data2)
{
    sisa_bagi=data2%2;
    sisa=data2/2;
    for(int i=1 ; i<11 ; i++)
    {
        if(sisa_bagi==1)
        {
           biner[x]='1';
           x=x-1;
           jml_1+=1;
        }
        else if(sisa_bagi==0)
        {
           biner[x]='0';
```

```
            x=x-1;
        }
        sisa_bagi=sisa%2;
        sisa=sisa/2;
    }

    gen_pariti(jml_1);


}

void gen_pariti(int data4)
{
    pariti=data4%2;
    if(pariti==0)
    {
        biner[11]='0';
    }
    else if(pariti==1)
    {
        biner[11]='1';
    }

}
```

3.5.3 Skrip untuk *node* BTS



Gambar 3.11 Diagram alir pembuatan skrip node BTS 1

Berikut ini penjabaran dari Gambar 3.11 proses pembuatan skrip pada *node* BTS 1.

1. Inisialisasi

Pada saat inisialisasi digunakan untuk menyiapkan variabel - variabel yang dibutuhkan dan memberikan nilai awal pada variabel - variabel tersebut. berikut adalah inisialisasi pada *node* BTS 1

```
String buff;
String simpan1,simpan2;
void setup()
{
    Serial.begin(9600);
}
```

2. Mengirimkan *Broadcast Packet*

Broadcast packet pada *node* BTS 1 dikirmkan setiap 5 detik sekali. Berikut adalah skrip untuk mengirimkan *broadcast packet*.

```
void rssi()
{
    delay(1000);
    Serial.println("$R1*");
}
```

3. Menyeleksi Pesan dari *Mobile node*.

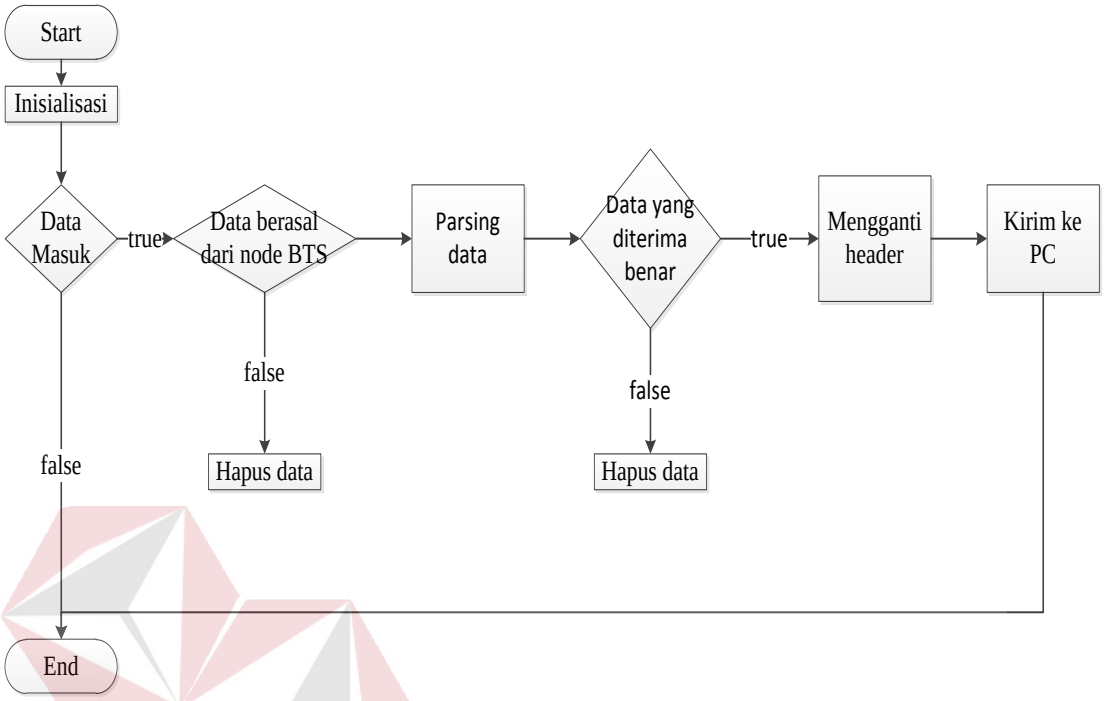
```
void loop()
{
    if(Serial.available()>0)
    {
        char data_masuk=(char)Serial.read();
       buff+=data_masuk;
       if(data_masuk=='\n')
                {
                    buff="";
                }
       if(data_masuk=='@')
       {
                if((buff[0]=='%') && (buff[2]=='1'))
                {
                    //fungsi untuk melakukan parsing data
                    parsing();
                    simpan1=buff;
                    buff="";
                }
                else if((buff[0]=='%') && (buff[2]=='2'))
                {
                    //fungsi untuk melakukan parsing data
                    parsing();
                    simpan2=buff;
                    buff="";
            }
                else buff="";

       }
    }
    else rssi();

}
```

Pada *node* BTS 1 ini pesan yang akan diproses hanya pesan dengan *header* '%' meskipun itu dari *end device* 1 atau 2. Kemudian pesan tersebut akan dipisah menggunakan fungsi parsing.

```
void parsing()
{
   for(int y=8;y<19;y++)
   {
       cek_data=int(buff[y]);
       if(cek_data==1)
       {
           pariti_cek+=1;
           cek_data=0;
       }
   }

   pariti=pariti_cek%2;

   if(buff[19]==pariti)
   {
       buff[0]='!';
       Serial.println(buff);
   }
   else if(buff[19] != pariti)
   {
       Serial.println("*1");
   }
}
```

Setelah data dilakukan pemilahan data akan dicek jika data yang diterima benar data akan diteruskan ke *node coordinator*, jika data salah data akan dibuang kemudian *node* BTS akan mengerimkan sebuah *request packet* untuk mengirimkan data lagi.

3.5.5 Skrip untuk *node coordinator*



Gambar 3.12 Diagram alir pembuatan skrip node coordinator

Berikut ini penjabaran dari Gambar 3.13 proses pembuatan skrip *node coordinator*.

1. Inisialisasi

Pada *node coordinator* ini juga melakukan inisialisasi. Berikut adalah skrip inisialisasi pada *node coordinator*.

```
String buff;
boolean node1=false;
boolean node2=false;
int error1=0;
int error2=0;

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

Pada saat *node coordinator* baru saja dinyalakan / aktif kondisi *node* BTS 1 dan *node* BTS 2 di ibaratkan mati / tidak aktif

2. Menyeleksi pesan dari *node* BTS 1 dan *node* BTS 2

Pada *node coordinator* hanya akan memproses dua, pesan pesan data dan *broadcast packet*. Pertama *node coordinator* akan memastikan apakah *node* BTS 1 atau *node* BTS 2 dalam keadaan aktif atau tidak aktif dengan cara membaca *broadcast packet* dari masing-masing *node* BTS.Berikut adalah skrip membaca *broadcast packet*.

```
if(node1==false) // error checking node 1
{
   if(data_masuk=='*')
   {
     if((buff[0]=='$') && (buff[2]=='1'))
     {
         Serial.println("node1hidup@");
         node1=true;
         error1=0;
     }
   }

}
if(node2==false) //error checking node 2
{
   if(data_masuk=='*')
   {
      if((buff[0]=='$')&&(buff[2]=='2'))
      {
         Serial.println("node2hidup@");
         node2=true;
         error2=0;
      }
  }
}
```

Setelah *node coordinator* mendapatkan *broadcast packet* dari *node* BTS 1 atau *node* BTS 2, *node coordinator* akan memberikan pemberitahuan ke PC dengan cara mengirimkan pesan "node1hidup@" untuk *node* BTS 1 jika hidup , "node2hidup@" untuk *node* BTS 2 jika hidup.

Setelah kondisi *node* BTS 1 atau *node* BTS 2 dinyatakan hidup oleh *node coordinator* baru dapat memproses pesan data dari *node* BTS 1 atau *node* BTS 2. Berikut adalah skrip yang digunakan.

```
if(node1==true)
        {

              if(data_masuk=='@')
              {
                if(buff[0]=='!')
                {
                    if((buff[4]=='R') && (buff[5]=='1'))
                    {
                      //delay(100);
                      parsing1();
                      error1=0;
                      ulang();
                    }

                }
                else buff="";
              }

        }
        if(node2==true)
         {

              if(data_masuk=='@')
              {
                if(buff[0]=='!')
                {
                    if((buff[4]=='R') && (buff[5]=='2'))
                    {
                      //delay(100);
                      parsing2();
                      error2=0;
                      ulang();
                    }

                }
                else buff="";
              }

        }
```

Pada *node coordinator* pesan data yang diterima terlebih dahulu akan dipilah apakah dari *node* BTS 1 atau *node* BTS 2 kemudian akan dipilah untuk mendapatkan nilai potensio yang dikirimkan oleh *node* BTS menggunakan fungsi parsing.

```
void parsing1()
{

   for(int y=8;y<19;y++)
   {
       cek_data=int(buff[y]);
       if(cek_data==1)
       {
           pariti_cek+=1;
           cek_data=0;
           hasil+=pangkat;
           pangkat/=2;
       }
       if(cek_data==0)
       {
           pangkat/=2;
       }

   }
   Serial.println(hasil);
   pariti=pariti_cek%2;

   if(buff[19]==pariti)
   {
      buff+="+E1.R1.";
      buff+=String(hasil);
      buff+="@";
      Serial.println(buff);
      hasil=0;
   }
   else if(buff[19] != pariti)
   {
       Serial.println("-1");
   }
}

void parsing2()
{
   for(int y=8;y<19;y++)
   {
       cek_data=int(buff[y]);
```

```
        if(cek_data==1)
        {
            pariti_cek+=1;
            cek_data=0;
            hasil+=pangkat;
            pangkat/=2;
        }
        if(cek_data==0)
        {
            pangkat/=2;
        }

    }

    pariti=pariti_cek%2;

    if(buff[19]==pariti)
    {
       buff+="+E1.R2.";
       buff+=String(hasil);
       buff+="@";
       Serial.println(buff);
       hasil=0;
    }
    else if(buff[19] != pariti)
    {
        Serial.println("-2");
    }
}
```

Setelah data dilakukan pemilahan data akan dicek jika data yang diterima benar data akan diteruskan ke PC, jika data salah data akan dibuang kemudian *node coordinator* akan mengerimkan sebuah *request packet* kepada *node BTS* untuk mengirimkan data lagi.

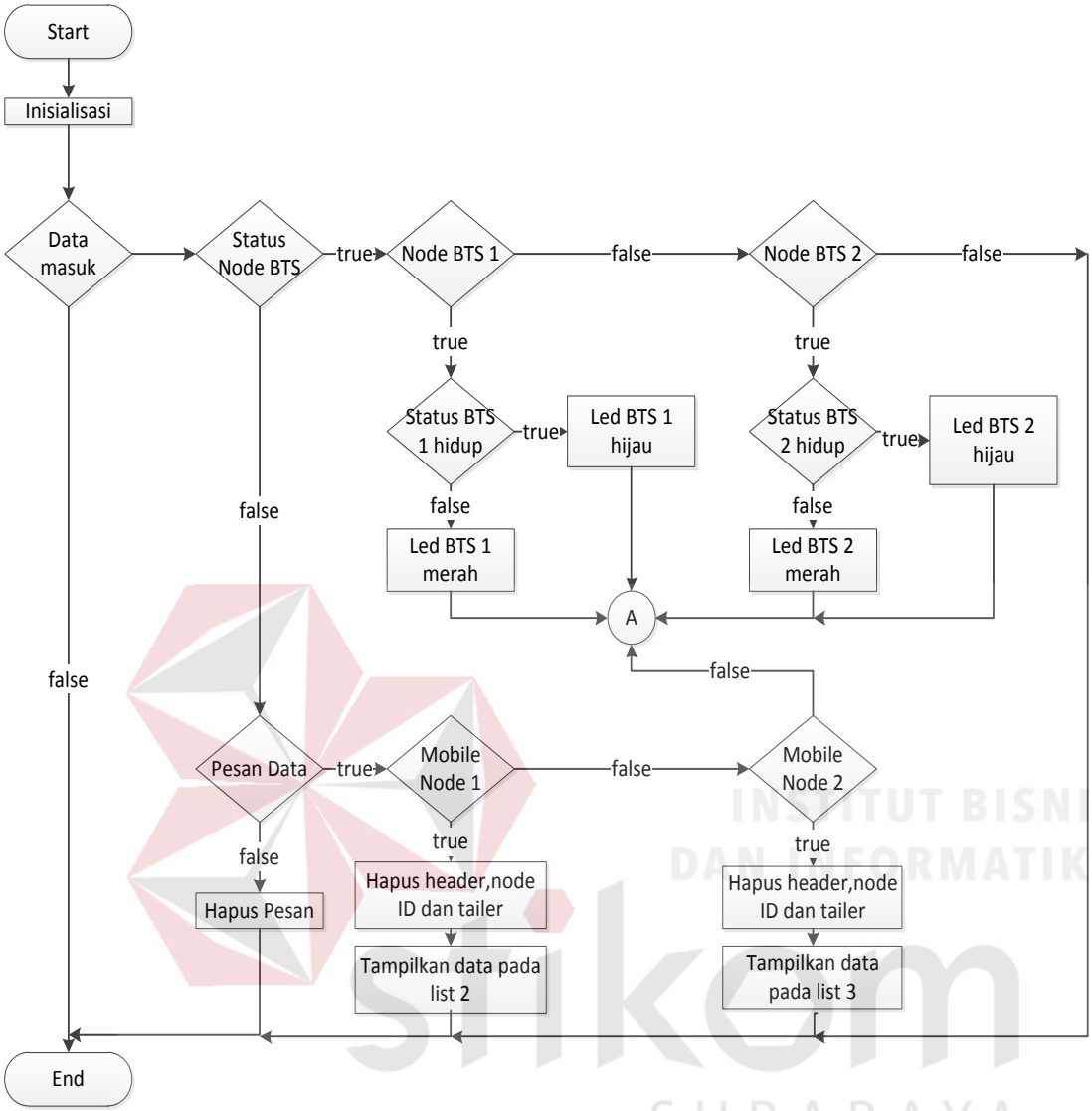
3. *Error checking* koneksi pada *node coordinator*

Setelah kondisi *node* BTS 1 atau *node* BTS 2 aktif, jika selama 9 detik tidak menerima pesan apapun dari *node* BTS 1 atau *node* BTS 2 *node coordinator* akan menyatakan bahwa *node* BTS 1 atau *node* BTS 2 dalam kondisi tidak aktif. Berikut ini adalah skrip *error checking* koneksi pada *node coordinator*.

```
void error_node()
{
    if(error1<=9)
    {
        delay(1000);
        error1++;
    }
    else
    {
        Serial.println("node1mati@");
        node1=false;
        error1=0;
    }

    if(error2<=9)
    {
        delay(1000);
        error2++;
    }
    else
    {
        Serial.println("node2mati@");
        node2=false;
        error2=0;
    }
}
```


3.6. Aplikasi pada PC (*Personal Computer*) menggunakan *software Visual basic*

Aplikasi ini melakukan pemantauan pesan data yang dikirimkan oleh *mobile node* sampai ke *node coordinator* dan juga menampilkan status dari *node* BTS 1 dan *node* BTS 2 dalam keadaan aktif atau tidak aktif. Berikut adalah blok diagram dari aplikasi.



Gambar 3.13 **Blok diagram aplikasi MWSN pada Visual basic**

Berikut ini penjabaran dari Gambar 3.14 proses pembuatan skrip aplikasi pada PC menggunakan *software visual basic*.

1. Inisialisasi

Pada proses inisialisasi aplikasi akan menentukan nilai awal untuk MScomm seperti pengaturan *baudrate, parity, data bit, stop bit parameter*. Pada aplikasi ini pengaturan tersebut menggunakan pengaturan *default* (9600,n,8,1). selain itu pengaturan *RTSEnable* bernilai *true* , *RThreshold* bernilai 1 dan *InputLen* bernilai 1. Pada proses ini juga menentukan led BTS 1

(*shape1*) dan led BTS 2 (*shape 2*) berwarna merah yang berarti BTS 1 dan BTS 2 dalam keadaan tidak aktif.

2. Pembacaan *port serial*

Pada saat aplikasi baru dijalankan aplikasi akan memberitahukan *port serial* berapa saja yang aktif dan dapat digunakan kemudian akan ditampilkan pada *combo box*. Berikut ini adalah skrip pembacaan *port serial* yang sedang aktif dan dapat digunakan.

```
Private Function serial()
On Error Resume Next

  Dim i As Integer

    For i = 2 To 100
        MSComm1.CommPort = i
        MSComm1.PortOpen = True

        If MSComm1.PortOpen Then
            Combo1.AddItem "COM" & i
            MSComm1.PortOpen = False
        Else
        End If
    Next
End Function
```

3. Pembacaan data masuk

Komunikasi *serial* pada *visual basic* ada beberapa *event* yang dapat digunakan namun pada aplikasi ini hanya menggunakan *event* data masuk atau pembacaan data yang diterima aplikasi melalui komunkasi *serial*. Berikut ini adalah skrip yang digunakan ketika ada data yang masuk.

```
Private Sub MSComm1_OnComm()
    If MSComm1.CommEvent = 2 Then
        ........
    Else
    End If
End Sub
```

4. Menyeleksi pesan dari *node coordinator*

Pesan yang diterima oleh aplikasi terlebih dahulu akan diseleksi, pesan status *node* BTS 1 dan *node* BTS 2. Berikut ini adalah skrip untuk menyeleksi pesan status *node*.

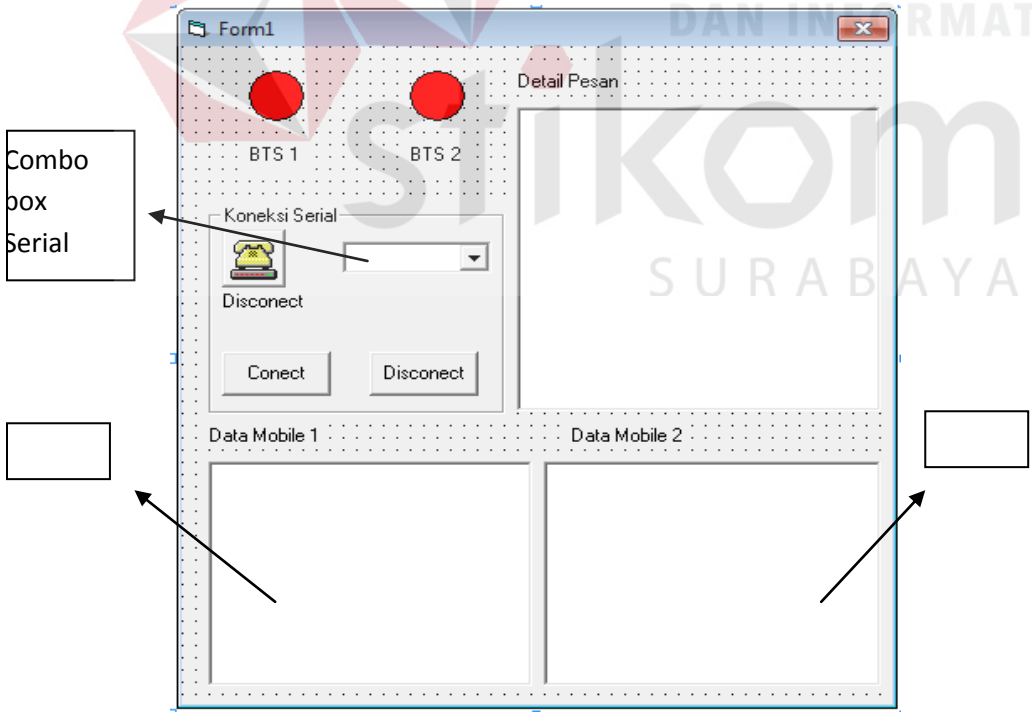
```
If MSComm1.CommEvent = 2 Then
        data_masuk = MSComm1.Input
        data = data + data_masuk

   If data_masuk = "@" Then
      If Mid(data, 1, 5) = "node1" Then
         If Mid(data, 6, 5) = "hidup" Then
            Shape1.FillColor = vbGreen
         End If
         If Mid(data, 6, 4) = "mati" Then
            Shape1.FillColor = vbRed
         End If
      ElseIf Mid(data, 1, 5) = "node2" Then
         If Mid(data, 6, 5) = "hidup" Then
            Shape2.FillColor = vbGreen
         End If
         If Mid(data, 6, 4) = "mati" Then
            Shape2.FillColor = vbRed
         End If
      End If
   End If
Else
End If
```

Berikut ini adalah skrip untuk menyeleksi pesan data.

```
If MSComm1.CommEvent = 2 Then
        data_masuk = MSComm1.Input
        data = data + data_masuk

  If data_masuk = "@" Then
     If Mid(data, 1, 1) = "+" Then
        If Mid(data, 2, 6) = "E1.R1." Or Mid(data, 2, 6) =
"E1.R2." Then
        If Len(data) = 12 Then
          List2.AddItem Mid(data, 8, 4)
        ElseIf Len(data) = 11 Then
          List2.AddItem Mid(data, 8, 3)
        ElseIf Len(data) = 10 Then
          List2.AddItem Mid(data, 8, 2)
        ElseIf Len(data) = 9 Then
          List2.AddItem Mid(data, 8, 1)
        End If
```

```
     ElseIf Mid(data, 2, 6) = "E2.R1." Or Mid(data, 2, 6) =
"E2.R2." Then
        If Len(data) = 12 Then
          List3.AddItem Mid(data, 8, 4)
        ElseIf Len(data) = 11 Then
          List3.AddItem Mid(data, 8, 3)
        ElseIf Len(data) = 10 Then
          List3.AddItem Mid(data, 8, 2)
        ElseIf Len(data) = 9 Then
          List3.AddItem Mid(data, 8, 1)
        End If
      End If
    End If

Else
End If
```

Pada saat pesan data berasal dari *mobile node* 1 data tersebut akan ditampilkan pada list 2, sedangkan jika berasal dari *mobile node* 2 akan ditampilkan pada list 3. Berikut ini adalah Gambar 3.13 tampilan aplikasi pada PC.



Gambar 3.14 Tampilan Aplikasi

Pada sistem MWSN ini aplikasi yang dibuat hanya sebagai pemantau pesan data yang dikirimkan oleh *mobile node* berhasil dikirimkan setelah melewati proses *handoff* pada *mobile node* dan *error checking* pada *node* BTS. Aplikasi ini juga sebagai pemantau status dari *node* BTS1 dan *node* BTS2 apakah dalam kondisi aktif atau tidak aktif.

BAB IV

HASIL DAN PEMBAHASAN

4.1. Kebutuhan *Hardware* dan *Software*

4.1.1. Kebutuhan Perangkat Keras (*Hardware*)

Penulis membutuhkan perangkat keras sebagai berikut :

1. Komputer atau laptop
2. Arduino SMD
3. Xbee *series* 2 dengan Xbee *shield*
4. Potensio

4.1.2. Kebutuhan Perangkat Lunak (*Software*)

Perangkat lunak yang digunakan dalam pengujian sistem ini menggunakan *software* arduino IDE.

4.2. Pengujian Sistem (Protokol *Handoff , Error checking* dan Aplikasi)

Pengujian ini dilakukan untuk mengetahui apakah protokol tersebut sudah berjalan sesuai dengan perancangan sistem. Terdapat dua buah *node* BTS yang diletakkan berjauhan supaya pergerakkan dari *mobile node* dapat lebih luas. Setiap *node* BTS akan mengirimkan *broadcast paket* ke *mobile node* yang kemudian akan didapatkan nilai RSSI dari *node* BTS yang telah terhubung dengan *mobile node*, kemudian *mobile node* akan melakukan perbandingan nilai RSSI *node* BTS yang telah terhubung dengan *mobile node*. *mobile node* akan digerakkan menggunakan sepeda motor.

4.2.1. Peralatan

Peralatan yang dibutuhkan dalam pengujian ini adalah sebagai berikut :

1. Arduino Uno
2. Xbee *series* 2 dengan Xbee *shield*
3. Komputer atau laptop
4. Potensio
5. Sepeda motor
6. *Software* arduino IDE
7. Kabel USB

4.3.2. Prosedur Pengujian

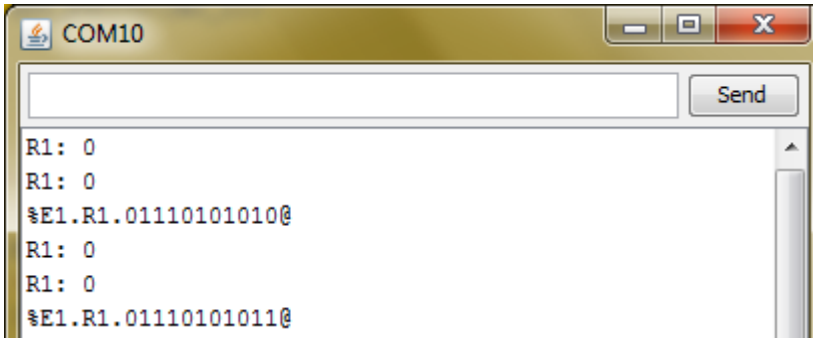
Langkah-langkah dalam pengujian ini adalah sebagai berikut :

1. Hubungkan potensio ke arduino uno menggunakan kabel pada pin yang telah ditentukan.
2. Hubungkan pin 6 Xbee menggunakan kabel pada pin yang telah ditentukan (langakah 1 dan 2 untuk *mobile node*)
3. Hubungkan arduino (*mobile node*) pada laptop menggunakan kabel USB
4. Letakkan *node* BTS 1 dan *node* BTS 2 berjauhan.
5. Letakkan *node coordinator* diantara *node* BTS 1 dan *node* BTS 2
6. Hubungkan setiap *node* BTS dan *node coordinator* pada laptop menggunakan kabel USB
7. Aktifkan laptop pada *node* BTS 1,2 dan *mobile node* kemudian jalankan *software* arduino IDE
8. Aktifkan laptop pada *node coordinator* kemuidan jalankan aplikasi yang telah dibuat menggunakan *software visual basic*

9. *Upload* skrip yang telah dibuat kedalam arduino uno (*node* BTS 1, *node* BTS 2 , *mobile node, node coordinator*)
10. Buka serial monitor pada *software* arduino IDE untuk melihat pengiriman pesan setiap *node*
11. Jalankan *mobile node* menggunakan sepeda motor
12. Amati setiap pesan yang dikirim dan diterima oleh masing-masing *node*
13. Amati pesan yang diterima oleh aplikasi

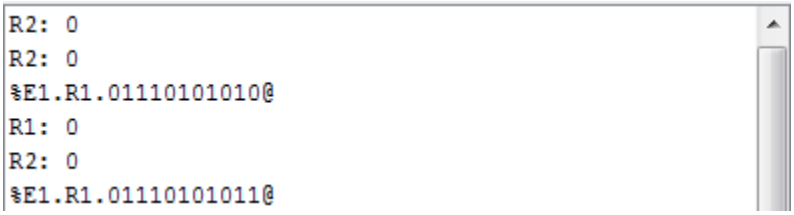
4.3.3. Hasil Pengujian

Pada pengujian ini *mobile node* setelah terhubung dengan salah satu *node* BTS akan menerima *broadcast paket* dari *node* BTS, kemudian *mobile node* akan membaca berapa nilai RSSI dari *node* BTS tersebut. Apabila *mobile node* terhubung dengan semua *node* BTS akan menerima semua *broadcast packet* dari *node* BTS 1 dan 2. *Mobile node* hanya akan membaca *broadcast paket* dua kali untuk melakukan perbandingan apakah dari *node* BTS 1 saja, *node* BTS 2 saja, atau *node* BTS 1 dan *node* BTS 2. Setelah itu baru data akan dikirimkan sesuai hasil dari perbandingan nilai RSSI dari masing-masing *node* BTS.



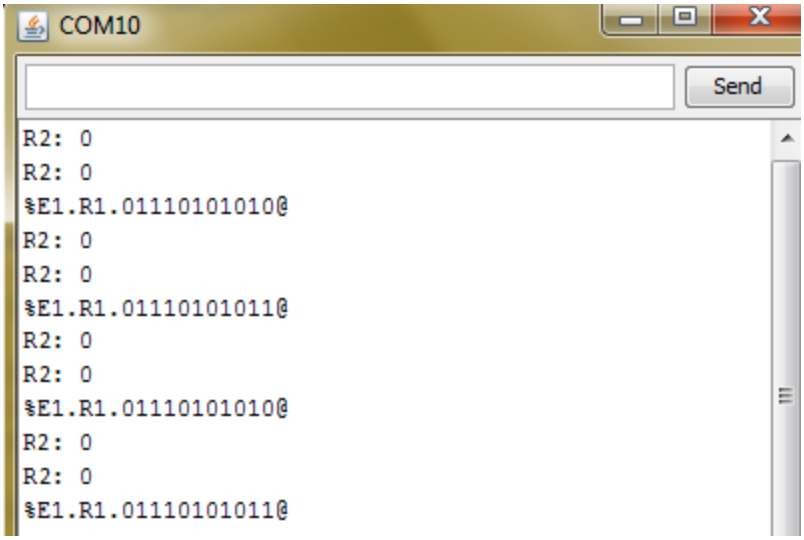
Gambar 4.1 Mobile node hanya terhubung dengan BTS 1

Setelah melakukan pembacaan dua kali *mobile* node akan mengirimkan pesan data. Dengan fomat penulisan pesan “ %E1.R1.01110101010@ “ karakter “%” sebagai *header*, “E1” sebagai ID *mobile node* , karakter “.” sebagai pemisah , “R1” sebagai ID *node* tujuan , “0111010101” adalah data dari potensio, "0" *parity bit* dan karakter “@” sebagai *tailer*. Gambar 4.2 menunjukkan ketika *mobile node* mendapat pesan dari BTS 1 dan BTS 2.



Gambar 4.2 Mobile node terhubung dengan BTS 1 dan BTS 2

Pada saat *mobile node* hanya menerima pesan *broadcast paket* dari BTS 2 *mobile node* juga akan membaca nilai RSSI dari *node* BTS 2.

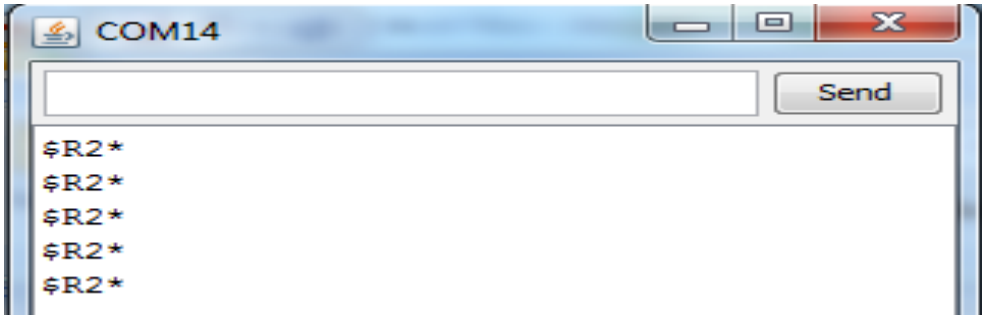


Gambar 4.3 Mobile node hanya terhubung dengan BTS 2

Pada Gambar 4.2 menunjukkan nilai RSSI dari BTS1 adalah nol dibandingkan dengan nilai RSSI dari BTS2 maka pesan data akan dikirimkan ke BTS1 dengan format pesan data yang telah ditentukan.

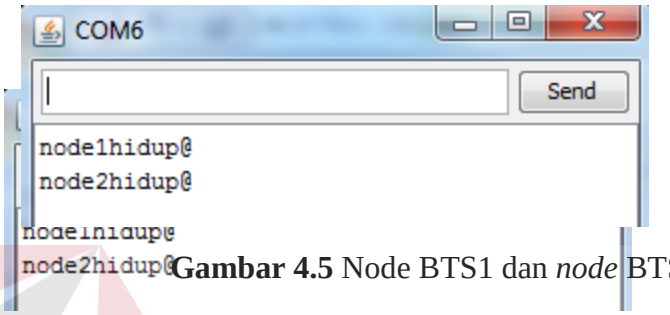
Pada *node* BTS, *broadcast paket* yang dikirimkan akan diproses sebagai *error checking* status pada *node coordinator* namun pada *mobile node* diproses sebagai pembacaan nilai RSSI *node* BTS selain *error checking* status pada sistem ini juga terdapat *error checking* data antara *node coordinator* dengan *node* BTS. Pada gambar 4.4 menunjukkan *node* BTS1 mengirimkan *broadcast packet* dan gambar 4.5 menunjukkan *node* BTS2 mengirimkan *broadcast packet*.

Gambar 4.3 Broadcast packet node BTS 1



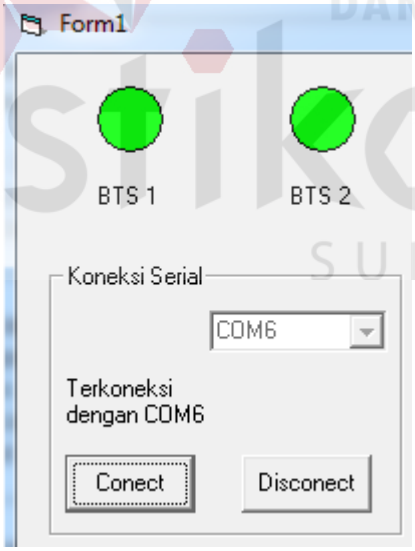
Gambar 4.4 Broadcast packet node BTS 2

Pesan *broadcast paket* yang kirimkan oleh *node* BTS akan diproses sebagai *error checking* status pada *node coordinator* namun pada *mobile node* diproses sebagai pembacaan nilai RSSI *node* BTS. Pada *node coordinator broadcast packet* dari *node* BTS1 akan diproses sebagai pesan status bahwa *node* BTS1 dalam kondisi aktif. Berikut ini adalah ketika *node coordinator* menyatakan *node* BTS1 dan *node* BTS2 dalam kondisi aktif.



Gambar 4.5 Node BTS1 dan *node* BTS2 aktif

Kemudian pada aplikasi led BTS1 dan BTS2 akan berwarna hijau. Berikut ini adalah tampilan pada aplikasi ketika *node* BTS1 dan BTS2 dalam keadaan aktif.



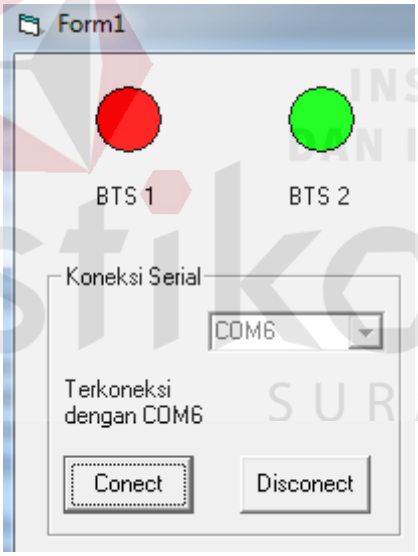
Gambar 4.6 Pada aplikasi *node* BTS1 dan *node* BTS2 aktif

Apabila dalam selang waktu sembilan detik *node coordinator* tidak menerima *broadcast packet* dari *node* BTS1 *node coordinator* akan menyatakan bahwa *node* BTS1 dalam keadaan

tidak aktif. Berikut ini adalah ketika *node coordinator* menyatakan *node* BTS1 dalam kondisi tidak aktif.

Gambar 4.7 Node BTS1 tidak aktif

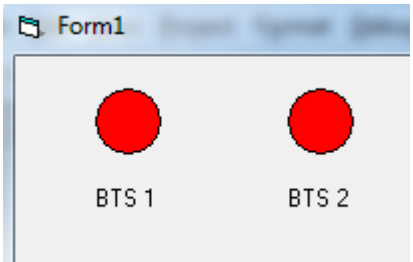
Kemudian pada aplikasi led BTS1 akan berwarna merah. Berikut ini adalah tampilan pada aplikasi ketika *node* BTS1 dalam keadaan tidak aktif.



Gambar 4.8 Pada aplikasi node BTS1 tidak aktif

Proses tersebut juga terjadi pada *node* BTS2 ketika *node* BTS2 dalam kondisi tidak aktif.

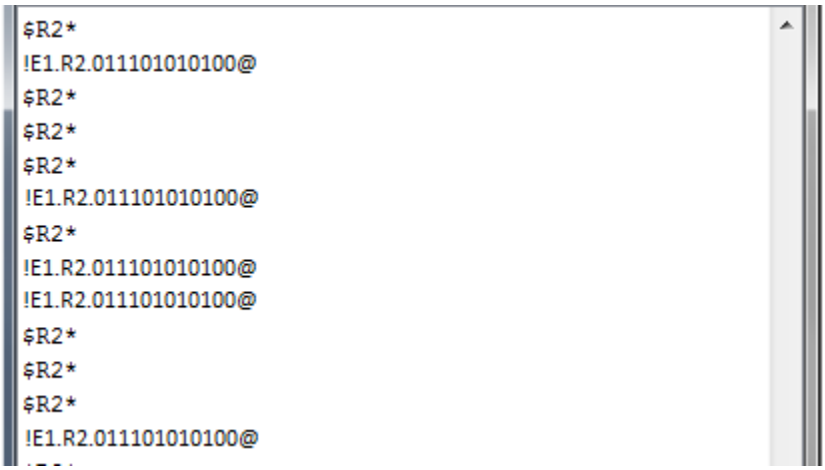
Gambar 4.9 node BTS2 tidak aktif



Gambar 4.10 Pada aplikasi node BTS 2 tidak aktif

Berikut ini adalah pesan data yang diterima oleh *node* BTS.

Gambar 4.11 Pesan data pada *node BTS 1*



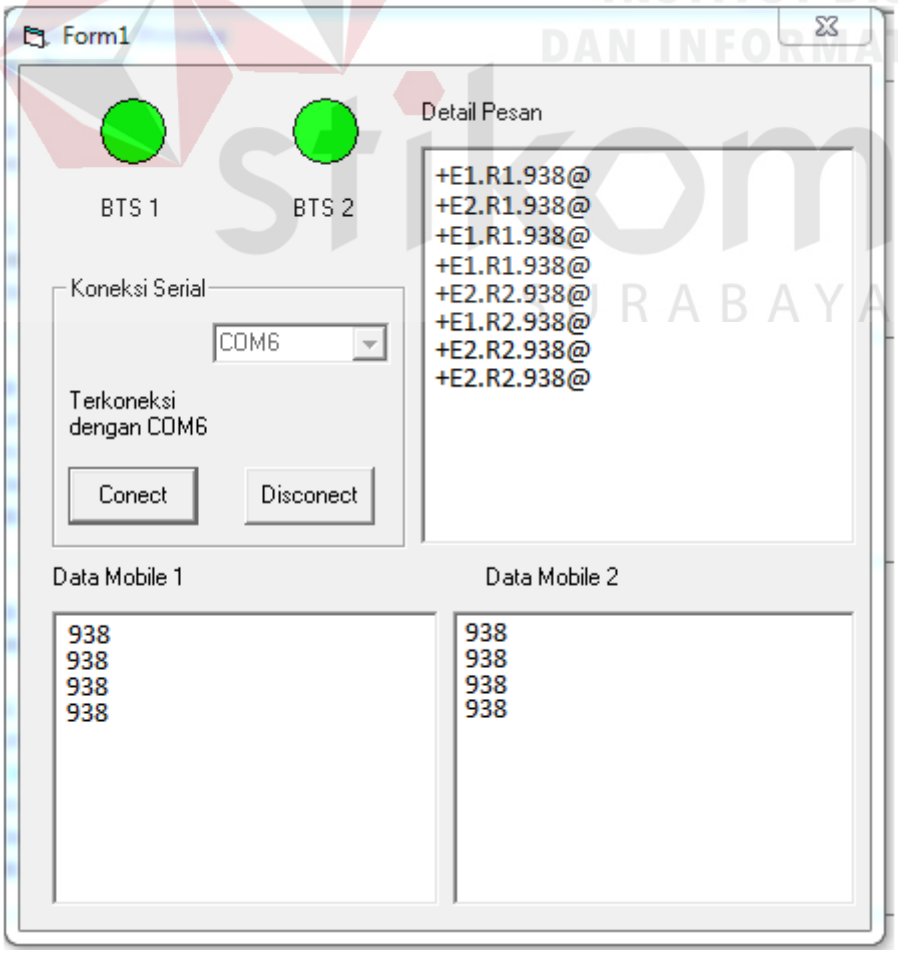
Gambar 4.11 Pesan data pada node BTS 2

Kemudian pesan data pada *node coordinator* akan diteruskan ke aplikasi pada PC / laptop. Berikut ini adalah pesan data pada *node coordinator*.

```
node1hidup@
node2hidup@
+E1.R2.938@
+E1.R2.938@
+E1.R2.938@
```

Gambar 4.12 Pesan data pada node coordinator

Berikut ini adalah pesan data yang telah diterima oleh aplikasi.



Gambar 4.13 Pesan data pada aplikasi

Pada aplikasi sistem ini hanya dibuat untuk membaca status dari *node* BTS dan membaca pesan data yang dikirimkan oleh *mobile node*.

Pengujian pada sistem dilakukan sebanyak enam kali percobaan dengan sampel data sebanyak tiga puluh data yang dikirimkan oleh *mobile node* dan berapa banyak data yang diterima oleh aplikasi. Pengujian ini dilakukan untuk dapat mengetahui presentase *paket loss* (paket/data hilang) terhadap kecepatan *mobile*. Pengujian dilakukan di luar ruangan / pada lapangan yang bebas dari halangan / hambatan

Tabel 4.1 Pengujian Pakcet Loss (Data Hilang) terhadap kecepatan *mobile*

Percobaan	**Kecepatan *Mobile***	**Data yang dikirim**		**Data yang diterima**		**Presentase**	
		Mobile Node 1	**Mobile Node 2**	**Mobile Node 1**	**Mobile Nodec2**	**Mobile Node 1**	**Mobile Node 2**
1	5 km/h	30 data	30 data	27	28	81%	84%
2	10 km/h	30 data	30 data	26	27	78%	81%
3	15 km/h	30 data	30 data	26	26	78%	78%
4	20 km/h	30 data	30 data	28	27	84%	84%
5	25 km/h	30 data	30 data	27	27	81%	81%
6	30 km/h	30 data	30 data	28	26	84%	78%

Setiap percobaan, *mobile node* akan mengirimkan data sebanyak tiga puluh data dengan awal kecepatan *mobile* 5 km/h kemudian pada aplikasi berapa banyak data yang dapat diterima.

4.4. Pengujian Jarak Kemampuan Pengiriman dan Penerimaan Xbee

Pada pengujian ini bertujuan untuk mengetahui kemampuan jangkauan atau jarak pengiriman dan penerimaan data.

4.4.1. Peralatan

Peralatan yang digunakan dalam pengujian ini adalah sebagai berikut:

1. Arduino uno
2. Xbee S2 beserta *shield*
3. Potensio
4. Kabel USB
5. Laptop
6. Baterai 9v (pada *node* BTS)

4.4.2. Prosedur Pengujian

Langkah – langkah dalam pengujian ini sebagai berikut :

1. Hubungkan *mobile node* pada laptop dengan kabel USB
2. Aktifkan laptop dan jalankan program arduino IDE
3. *Upload* skrip atau program yang digunakan pada *mobile node*
4. Setelah *upload* program selesai , buka serial monitor pada IDE
5. *Upload* skrip atau program yang digunakan pada *node* BTS1 dan BTS2
6. Hubungkan *node* BTS1 dan BTS2 dengan catu daya atau baterai pada masing-masing *node*
7. Ukur jarak antar *node* dan carilah jangkauan jarak maksimal pengiriman dan penerimaan data.

4.4.3. Hasil Pengujian

Pengujian ini dilakukan di luar ruangan dan di dalam ruangan, didapatkan hasil jarak atau jangkauan pengiriman data sebagai berikut :

Tabel 4.2 Hasil Pengujian Jarak Pengiriman dan Penerimaan data pada Xbee (Di luar ruangan)

.	Jarak (meter)	Nilai PWM RSSI	Data yang dikirim	Data yang diterima	Presentase
	10	+/- 20	30	28	84%
	20	+/- 40	30	27	81%
	30	+/- 60	30	27	81%
	40	+/- 80	30	26	78%
	50	+/- 100	30	23	69%
	60	+/- 110	30	20	60%
	70	+/- 130	30	19	57%
	80	+/- 140	30	17	51%
	90	+/- 150	30	14	42%
	100	+/- 160	30	10	30%
	105	-	30	-	-
	110	-	30	-	-

Tabel 4.3 Hasil Pengujian Jarak Pengiriman dan Penerimaan data pada Xbee (Di dalam ruangan)

o.	Jarak (Meter)	Nilai PWM RSSI	Data yang dikirim	Data yang diterima	Presentase
	10	+/- 20	30	27	81%
	20	+/- 40	30	25	75%
	30	+/- 60	30	20	60%
	40	+/- 80	30	17	51%
5.	45	-	30	-	-

Pada kondisi di luar ruangan dengan jarak 1 - 100 meter pengiriman dan penerimaan data antar *node* dapat berkomunikasi dengan baik, tetapi pada jarak lebih dari 100 meter tidak dapat berkomunikasi. Artinya komunikasi antar *node* tersebut terputus, sehingga *node* yang dituju tidak dapat menerima data yang dikirimkan oleh *node* lain.Sedangkan pada kondisi di dalam ruangan dengan jarak 1 – 40 meter dapat berkomunikasi dengan baik. Setelah jarak melebihi 40 meter komunikasi terputus. Dengan demikian hasil yang didapat sama dengan spesifikasi pada *datasheet* xbee S2.

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Kesimpulan dari tugas akhir "DESAIN DAN IMPLEMENTASI PROTOKOL *HANDOFF* DAN *ERROR CHECKING* PADA JARINGAN MWSN (*MOBILE WIRELESS SENSOR NETWORKS*)" adalah :

1. Sistem MWSN menggunakan 5 *node* yaitu *node* BTS 1, BTS 2, *coordinator*, *mobile node* 1 dan *mobile node* 2. Pada *mobile node* 1 dan 2 terdiri dari modul mikrokontroler arduino uno sebagai otak keseluruhan sistem untuk melakukan *handover* dan pembacaan data dari potensio. Data potensio akan dikirimkan ke *node* lainnya secara nirkabel menggunakan modul Xbee *series* 2 beserta *shield*. Sedangkan *node coordinator*, *node* BTS 1 dan *node* BTS 2 terdiri dari modul mikrokontroler arduino uno dan modul Xbee *series* 2 beserta *shield*.
2. Sistem melakukan pengukuran terhadap paket yang hilang / *packet loss* melalui komunikasi nirkabel dan didapatkan hasil pengujian sebagai berikut:
   a) Rata – rata persentase paket hilang / *packet loss* terhadap kecepatan *mobile node* sebesar 70 % sampai 85 %.
   b) Rata - rata persentase paket hilang / packet loss terhadap jarak (di dalam ruangan) sebesar 30% - 80% sedangkan (di luar ruangan) sebesar 50% - 80%
   c) Berdasarkan hasil pengujian didapatkan bahwa parameter jarak antar xbee sangat terpengaruh terhadap jumlah data yang diterima, sedangkan parameter kecepatan tidak terlalu terpengaruh.

5.2. Saran

Agar diperoleh hasil yang lebih baik dan sebagai pengembangan pada penelitian berikutnya sebaiknya mempertimbangkan langkah ini:

1. Jika ingin mendapatkan jangkauan jarak komunikasi nirkabel yang lebih luas, dapat menggunakan Xbee Pro *series* 2 pada setiap *node*.
2. Coba diterapkan pada sebuah pengaplikasian yang nyata untuk dapat mengembangkan sistem ini.

Daftar Pustaka

arduino.cc/ArduinoXbeeShield. 2013. *Arduino Xbee Shield,* [online], (http://arduino.cc/en/Main/ArduinoXbeeShield, diakses tanggal 24 November 2013)

arduino.cc/Reference. 2013. *Arduino Program Language Reference,* [online], (http://arduino.cc/en/Reference/HomePage, diakses tanggal 24 November 2013)

arduino.cc/ArduinoBoardUnoSMD. 2013. *Arduino Board Uno SMD,* [online] (http://arduino.cc/en/Main/ArduinoBoardUnoSMD, diakses tanggal 24 November 2013)

arduino.cc/software. 2013. *Software Arduino IDE,* [online] (http://arduino.cc/en/main/software, diakses tanggal 24 November 2013)

Arrosyid, M. H., Tjahjono, I. A., & Epyk Sunarno, S. 2009. *Implementasi Wireless Sensor Network untuk Monitoring Parameter Energi Listrik sebagai Peningkatan Layanan bagi Penyedia Energi Listrik*. Surabaya: PENS

Banzi, M. 2009. *Getting Started with Arduino.* America: O'Reilly.

Andharini,C. 2008. Analisa Kinerja Virtual Interface Pada Vertikal Handover 802.11 (Wireless Area Network) Dan 802.15 (Personal Area Network/Bluetooth). Yogyakarta: SNATI 2008

Djuandi, F. 2011. *Pengenalan Arduino.* Banten: tobuku.com.

Muhammad E I., Sugiarto, B., & Sakti, I. 2009. *Rancang Bangun Sistem Monitoring Kualitas Udara Menggunakan Teknologi Wireless Sensor Network (WSN).* Jakarta: INKOM.

Faludi, R. 2011. *Building Wireless Sensor Networks.* America: O'Reilly.

*XBee-Pro ZB RF Modules.*2012. Minnetonka: Digi International Inc.

*XBee Series 2 OEM RF Modules.*2007. Lindon: MaxStream.

*X-CTU Configuration & Test Utility Software.*2008. Minnetonka: Digi International Inc.

Rezazadeh,J, Marjan,M. Abdul,S,S. 2012. Mobile Wireless Sensor Networks Overview. IJCCN : Volume 2, Issue 1

Maribun , S. 2008. Implementasi Sistem *Wireless Sensor Networks* Berbassis *Internet Protocol* (IP) untuk Pemantauan Tingkat Polusi Udara. Jakarta: UI

Tirtawinata, K. Penggunaan Logika Even Parity pada Beberapa Error Correction Code Terutama pada Hamming Code. Bandung: ITB