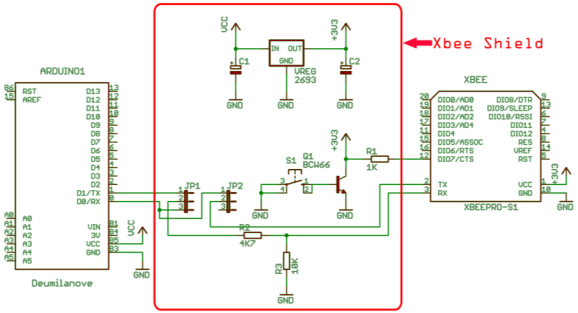
## Unit Komunikasi (modul RF)

Unit komunikasi merupakan bagian yang bertugas untuk mengirim dan menerima data dari tiap *node*. Dalam pembuatan jaringan sensor nirkabel ini terdapat 3 buah *node* yang terdiri dari 2 buah *router* dan 1 buah *gateway.* Antara *router*dan *gateway* memiliki perbedaan skematik dalam hal jalur komunikasi dengan mikrokontroler serta perbedaan algoritma yang digunakan. Sehingga dalam unit komunikasi dibagi menjadi dua buah yaitu *router* dan *gateway.*

### Skematik rangkaian modul RF pada router

Skematik rangkaian perantaraan modul RF (Xbee Pro Series 1) dengan mikrokontroler (Arduino Deumilanove) ditunjukan oleh gambar 4.28.



*Gambar 4.28 Skematik perantaraan Xbee* Pro *Series 1 dengan Arduino Deumilanove pada router*

Pada router digunakan *Xbee Shield* (kotak merah) sebagai perantara antara arduino dengan modul Xbee. Penggunaan *Xbee Shield* memberikan keuntungan sebagai berikut.

1. Instalasi xbee dengan arduino dipermudah karena tinggal menancapkan modul pada soket yang tersedia.
2. Tersedia regulator tegangan 3,3 Volt yang sesuai dengan tegangan suplai yang dibutuhkan modul Xbee sehingga tidak perlu membuat penyesuai tegangan lagi.
3. Dapat digunakan sebagai perantara untuk pemrograman Xbee dengan cara melepas mikrokontroler yang ada pada modul arduino.
4. Terdapat penyesuai level tegangan antara pin TX arduino dengan pin RX Xbee.

Bentuk *Xbee* Shield beserta Xbee Pro Series 1 yang telah terpasang pada *Xbee Shield* ditunjukan oleh gambar 4.30.



*Gambar 4.30. Xbee Shield yang telah terpasang modul Xbee Pro Series 1.*

Pada *router* , jalur komunikasi antara modul Xbee dengan arduino sama dengan jalur komunikasi yang digunakan untuk komunikasi serial arduino dengan komputer melalui perantara konverter *USB to Serial* menggunakan FT232. Sehingga apabila *Xbee Shield* terpasang pada arduino dan akan melakukan proses *download* program dari komputer ke arduino harus mengubah posisi sambungan pada jumper JP2 (gambar 4.8) yakni pin nomor 2 dihubungkan dengan pin no 3. Sehingga hubungan pin RX dari arduino dengan pin TX Xbee terputus dan program dapat ter*-download* dengan sempurna. Jika tidak dilakukan pemindahan posisi jumper JP2 maka akan menyebabkan *error* pada proses penulisan program dari komputer ke arduino.

### Algoritma Pengiriman Data Pada Router

Pembuatan algoritma program pengiriman data pada *router* menggunakan bantuan *library* untuk xbee (file “Xbee.h”) yang ditulis oleh Andrew Rapp dari <http://fsf.org/> (2009). Sebelum membuat program terlebih dahulu dilakukan seting parameter pada modul Xbee sebagai berikut.

|  |  |
| --- | --- |
| Parameter | Nilai |
| BD (*Baud rate*) | 9600 |
| DL (*Destination Address Low*) | 7E63 |
| AP (*API* *mode*) | 2 |

Setelah parameter di seting sesuai tabel diatas, langkah selanjutnya adalah penyusunan algoritma program. Pembuatan algoritma *router* mengacu pada *library* yang digunakan serta penyesuaian dengan karakteristik modul Xbee. Langkah – langkah pembuatan algoritma *router* adalah sebagai berikut.

1. Memasukan *library* Xbee.h kedalam program.
2. Membuat variabel bertipe Xbee.
3. Membuat variabel *buffer* untuk menampung data yang akan dikirim. Variabel ini bertipe integer positif 8 bit (1 *byte*) dan berbentuk *array* dengan besarnya *array* tergantung dari banyaknya data yang akan dikirim.
4. Membuat variabel yang bertipe *XbeeAddress64* dan nilainya diisi dengan *XbeeAdress64(DH,DL)* dimana DH merupakan 32 bit MSB dari alamat Xbee yang dituju dan DL merupakan 32 bit LSB dari alamat Xbee yang dituju.
5. Membuat variabel bertipe *Tx16Request* untuk menampung data keseluruhan yang yang akan dikirim yang mana variabel ini akan diisi dengan hasil dari fungsi *Tx16Request(addr16,payload, sizeof(payload)). Addr16* merupakan 16 bit LSB alamat pengirim.
6. Membuat variabel bertipe *TxStatusResponse*. Variabel ini akan menampung hasil dari fungsi *TxStatusResponse().*
7. Membuat algoritma pengiriman data yakni mengisi *payload* yang telah dibuat dengan data yang akan dikirim.

Pada program pengiriman data pada *router* dibuat variabel dengan nama *xbee* yang bertipe *Xbee* kemudian variabel *buffer* untuk menampung data yang akan dikirim bernama *payload* dengan tipe uint8\_t (*unsigned integer 8 bit*) atau 1 *byte* dan berbentuk *array* dengan panjang *array* 13. Panjang *array* ditentukan dengan menghitun banyaknya *byte* data yang akan dikirim. Data – data yang akan dikirim diantaranya adalah sebagai berikut.

1. Data dari unit RTC berupa data jam, menit, detik, tanggal, bulan dan tahun yang masing – masing 1 *byte* sehingga panjang data dari RTC berjumlah 6 *byte.*
2. Data jarak perpindahan modul hasil integrasi data percepatan yang dibaca oleh akselerometer. Data jarak perpindahan terdiri dari 3 sumbu yang tiap – tiap sumbu terdiri dari 2 *byte* sehingga jumlah data yang dikirim sebanyak 6 *byte*. Alasan mengapa data perpindahan membutuhkan alokasi 2 *byte* dikarenakan data perpindahan ada yang bernilai negatif sedangkan mengirimkan bilangan negatif pada komunikasi serial sering menimbulkan masalah pada perhitungan sehingga *payload* bertipe uint8\_t yang memiliki jangkauan nilai dari 0 hingga 255. Agar dapat mengirimkan data negatif maka ditambahkan 1 *byte* lagi sebagai penanda data positif atau negatif. Jika data positif , *byte* ini akan bernilai 0x00 sedangkan jika datanya negatif, *byte* ini bernilai oxff. Data perpindahan sendiri bertipe float dengan ketelitian 2 angka dibelakang koma sehingga agar data perpindahan dapat dikirim maka dilakukan konversi ke uint8\_t dengan dikalikan 100 terlebih dahulu.
3. Jumlah keseluruhan data yang akan dikirim adalah 12 *byte* yakni 6 *byte* data dari RTC dan 6 *byte* data jarak perpindahan. Dengan mengirimkan data sebanyak 12 *byte* maka panjang data (*data length*) yang diterima oleh *gateway* seharusnya adalah 17 *byte* karena mendapat tambahan 5 *byte* yang terdiri dari 1 *byte* data *API identifier,* 2 *byte* data *ID* *router*, 1 *byte* data RSSI (*Ricieve Signal Strength*) dan 1 *byte* data *options byte.* Desimal 17 dalam format heksadesimal adalah 0x11.
4. Dalam format API (*Aplication Programming Interface*) mode 2 terdapat *escape byte* yakni *byte – byte* yang berfungsi sebagai karakter kusus dalam format ASCI diantaranya 0x7E (karakter ~ / *start delimeter* pada jaringan Zigbee),0x11(karakter Xon) dan 0x13 (karakter Xoff / enter). Sehingga jika *data length* adalah 0x11 maka akan terdeteksi sebagai *escape byte* . Salah satu cara untuk menghindari *escape byte* adalah dengan menambahkan 1 *byte* data pada *payload* sehingga jumlah *payload* menjadi 13 dan *data length* menjadi 0x12.

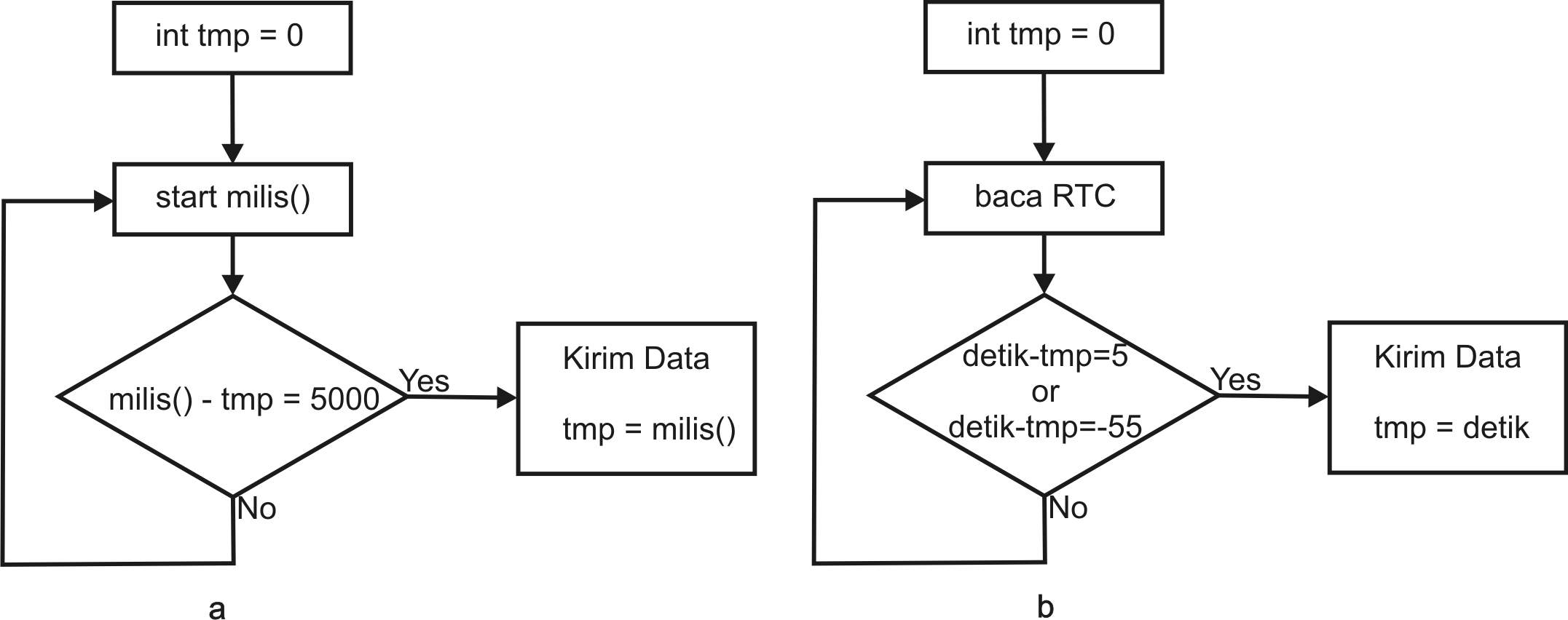
Setelah menentukan format data yang dikirim, langkah selanjutnya adalah menentukan periode pengiriman data.

Untuk menentukan periode pengiriman data dapat dilakukan dengan dua alternatif yaitu memanfaatkan *timer/counter* internal yang dimiliki oleh mikrokontroler atau dengan memanfaatkan *timer* eksternal yaitu dari modul RTC.

Jika menggunakan *timer/counter* internal mikrokontroler pada arduino dapat memanfaatkan fungsi milis()atau micros()*.*fungsi milis() merupakan fungsi penghitung naik (*count up* ) dengan nilai keluaran bertipe unsigned long sehingga jangkauan *counter* ini adalah dari 0 hingga 4.294.967.295 atau 4 *byte* (232 – 1). Nilai *counter* akan bertambah satu setiap 1 milidetik sehingga nilai maksimal 4.294.967.295 akan tercapai setelah 49 hari lebih 17 jam 2 menit 47 detik setelah arduino dijalankan. Dan pada saat ini *counter* akan mengalami *over flow* dan kembali ke – 0. Sedangkan fungsi micros() merupakan fungsi yang serupa dengan milis() akan tetapi nilai *counter* bertambah setiap 1 mikrodetik. Fungsi ini juga bertipe unsigned long sehingga akan mengalami *over flow* setelah 71 menit 35 detik arduino dijalankan.

Alternatif pewaktuan kedua adalah dengan memanfaatkan data yang diberikan oleh unit RTC. Data yang diberikan oleh RTC berupa data tahun, bulan, tanggal, hari, jam, menit dan detik. Untuk mengatur selang waktu pengiriman data dapat memanfaatkan data waktu ini.

Perbedaan algoritma perhitungan selang waktu pengiriman data menggunakan *timer/counter* internal dengan *timer* dari unit RTC ditunjukan pada flowchart pada gambar 4.30 dibawah. Kedua program sama - sama mengatur selang waktu pengiriman data tiap 5 detik.

**

*Gambar 4.30 Flowchart algoritma pewaktuan pengiriman data. (a) menggunakan counter internal, (b) menggunakan RTC*

Kelemahan dari pewaktuan menggunakan *counter* internal yakni pada akurasi pewaktuan. Karena arduino menggunakan crystal 16 MHz tingkat akurasi *counter* sebesar 4 mikrodetik sehingga tiap penambahan nilai maka *error* sebesar 4 mikrodetik ini ikut bertambah juga. *Counter* internal ini juga akan mengalami gangguan jika mikrokontroler mengalami *hang* pada proses perhitungan ataupun proses transfer data antar register.

Sedangkan pemakaian unit RTC sebagai penentu waktu pengiriman data memiliki tingkat akurasi yang lebih baik selama tidak ada gangguan eksternal pada rangkaian. Penggunaan RTC eksternal juga lebih stabil karena sumber *clock* terpisah dari sistem mikrokontroler sehingga tidak terpengaruh oleh proses yang sedang terjadi pada internal mikrokontroler.

Agar pewaktuan pengiriman data berjalan konstan, proses perhitungan selang waktu pengiriman data dibuat dalam subrutin interupsi sehingga apabila nilai selang waktu telah didapat maka program interupsi berjalan dan menunda proses yang sedang terjadi pada program utama.

Program utama pada router terdiri dari pengambilan data dari akselerometer, perhitungan jarak perpindahan serta menyimpan nilai – nilai variabel yang akan dikirim kedalam *payload*. Jadi ketika rutin interupsi berjalan, nilai yang dikirim merupakan nilai terakhir yang disimpan didalam *payload*. Setelah menjalankan rutin interupsi, nilai – nilai pada variabel yang dikirim tadi kemudian direset kembali dan *payload*  diisi dengan nilai yang baru.

Programutama pengiriman data pada *router* dalam format bahasa C arduino adalah sebagai berikut.

void **loop**() {

getDateDs1307(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month, &year);

//==================routine interupsi==================//

noInterrupts();

if((second-tmp == 5) || (second-tmp == -55)){

xbee.send(tx);

Vx, Vy, Vz, Dx, Dy, Dz = 0;

flashLed(statusLed, 1, 300);

tmp = second;

}

interrupts();

//==================Data Waktu==========================//

payload[0] = hour & 0xff;

payload[1] = minute & 0xff;

payload[2] = second & 0xff;

payload[3] = dayOfMonth & 0xff;

payload[4] = month & 0xff;

payload[5] = year & 0xff;

//=================Mengukur Percepatan================//

Ax = ((float(getH48C(0,0) - 2048))\*0.0022)-Xawal;

Ay = ((float(getH48C(0,1) - 2048))\*0.0022)-Yawal;

Az = ((float(getH48C(1,0) - 2048))\*0.0022)-Zawal;

//====remove vibrasi dengan filter IIR ===============//

accelX = IIR(Ax, accelXInput, accelXOutput);

accelY = IIR(Ay, accelYInput, accelYOutput);

accelZ = IIR(Az, accelZInput, accelZOutput);

// ============ hitung kecepatan ==========//

Vx += (accelX \* 0.01);

Vy += (accelY \* 0.01);

Vz += (accelZ \* 0.01);

// =========== hitung perpindahan ========== //

Dx += (Vx \* 0.01);

Dy += (Vy \* 0.01);

Dz += (Vz \* 0.01);

//===========konversi float ke uint8\_t==============//

if (Dx < 0 ){

byteDx = byte(Dx\*(-100));

payload[6] = 0xff;

payload[7] = byteDx & 0xff;

}

else {

byteDx = byte(Dx\*100);

payload[6] = 0x00;

payload[7] = byteDx & 0xff;

}

if (Dy < 0 ){

byteDy = byte(Dy\*(-100));

payload[8] = 0xff;

payload[9] = byteDy & 0xff;

}

else {

byteDy = byte(Dy\*100);

payload[8] = 0x00;

payload[9] = byteDy & 0xff;

}

if (Dz < 0 ){

byteDz = byte(Dz\*(-100));

payload[10] = 0xff;

payload[11] = byteDz & 0xff;

}

else{

byteDz = byte(Dz\*100);

payload[10] = 0x00;

payload[11] = byteDz & 0xff;

}

payload[12] = 0x00;

}

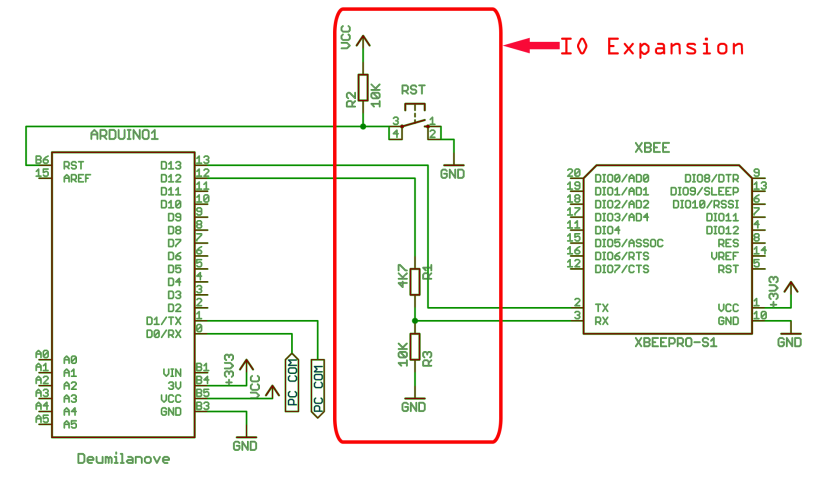
### Skematik rangkaian modul RF pada gateway

Fungsi utama dari *gateway* adalah untuk menerima data yang dikirim dari *router* serta mengirim data ke komputer server. Permasalahan yang dihadapi adalah perangkat modul RF dengan komputer server keduanya menggunakan mode komunikasi UART dengan arduino deumilanove sedangkan jalur komunikasi UART pada arduino deumilanove hanya terdapat 1 buah yakni pin 0 dan pin 1 (pin RX dan pin TX).

Apabila menggunakan Xbee Shield seperti halnya *router* maka diperlukan pemindahan jumper JP1 dan JP2 (gambar 4.28) secara terus menerus sehingga jalur komunikasi UART pada arduino dapat digunakan secara bergantian dan transfer data baik dari modul RF ke arduino atau dari arduino ke komputer server dapat dilakukan. Hal ini sangat tidak efisien apabila dilakukan secara manual. Apabila menggunakan *switching* secara elektronis misalnya menggunakan transistor atau optocoupler maka pewaktuan proses pensaklaran tergantung dari karakteristik komponen yang digunakan sehingga dapat dimungkinkan adanya *error* pada proses transfer data.

Salah satu solusi dari permasalahan tersebut adalah dengan menggunakan pendekatan *software* yakni dengan memanfaatkan *library* NewSoftSerial yang telah tersedia pada paket arduino IDE 0022. *Library* NewSoftSerial dibuat oleh David A. Mellis (2006). Dengan menggunakan *library* NewSoftSerial maka dapat membuat pin arduino yang bukan jalur komunikasi UART berfungsi sebagai jalur komunikasi UART sehingga dalam satu modul arduino deumilanove dapat dimungkinkan terdapat lebih dari satu jalur komunikasi UART.

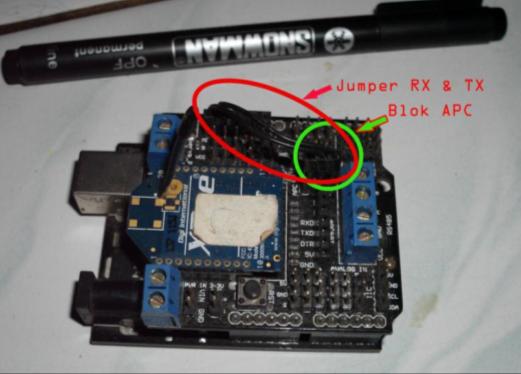
Skematik rangkaian pada *gateway* sesuai dengan gambar 4.31 berikut.



**Blok APC**

*Gambar 4.31 Skematik* rangkaian *pada gateway*

Pada *gateway*, perantaraan Xbee dengan arduino dibantu menggunakan arduino IO Expansion. Posisi pin TX dan RX modul Xbee terhubung pada blok APC yang terlebih dahulu melalui penyesuai level tegangan pada jalur RX Xbee. Untuk menghubungkan pin TX dan pin RX yang terdapat pada blok APC dari arduino IO Expansion dengan pin 12 dan pin 13 arduino deumilanove dilakukan dengan menambahkan kabel *jumper* seperti pada gambar 4.32.

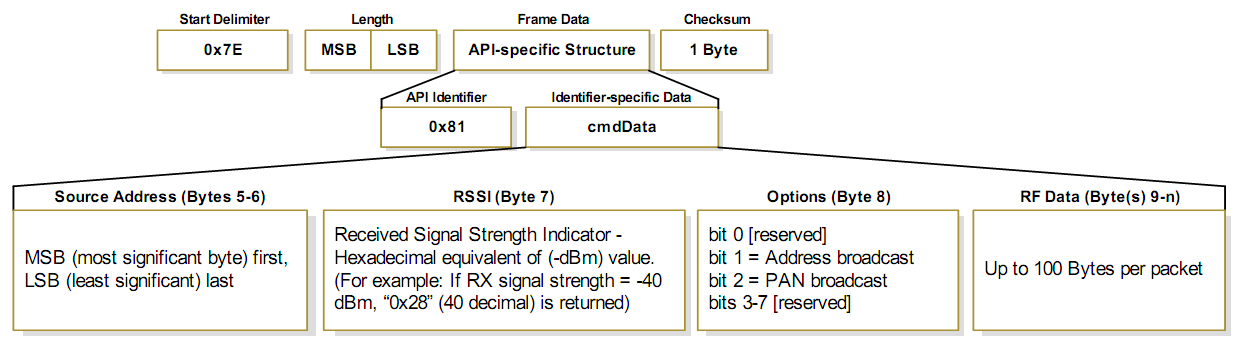


*Gambar 4.32 Jumper RX dan TX* pada *blok APC dari arduino IO Expansion dengan pin 12 dan pin 13*

### Algoritma pembacaan data pada gateway

Fungsi utama *gateway* adalah sebagai penerima data yang dikirim secara *wireless* oleh *router* dan mengirimkan kembali ke komputer server melalui komunikasi serial. Tugas ini sebenarnya dapat dilakukan dengan cara melewatkan langsung data yang diterima oleh Modul RF kemudian mengirimkan ke komputer server. Akan tetapi data yang diterima merupakan data yang ter-enkripsi dalam format API, sedangkan pengamatan terhadap data yang ter-enkripsi relatif lebih sulit jika dibandingkan dengan data yang tidak ter-enkripsi. Sehingga proses enkode frame data yang diterima dilakukan pada modul *gateway*.

Langkah pertama proses enkode frame data yang diterima adalah dengan mengenali tipe frame yang dikirim oleh *router*. Pengalamatan modul yang digunakan adalah dengan menggunakan 16 bit alamat sehingga tipe frame API pada protokol zigbee dengan mode 16 bit alamat adalah 0x81 (Maxstream.Inc, 2007). Tipe frame data pada API *identifier* = 0x81 ditunjukan pada gambar 4.33 berikut.



*Gambar 4.33 Tipe frame API untuk Xbee RX packet pada mode pengalamatan 16 bit (Maxstream.Inc, 2007)*

Dapat dilihat format frame data pada tipe frame 0x81 bahwa dalam satu frame terdiri dari *start delimeter*(*byte* ke-1), data *length* (*byte* ke-2 dan ke-3), *Frame Data* (*byte* ke-4 hingga ke-n) dan *checksum* (*byte* terakhir).

Setelah mengetahui tipe frame API yang dikirim oleh router, langkah selanjutnya adalah menganalisa format frame data yang diterima. Frame data terdiri dari *Api Identifier* (*byte* ke-4) dan *identifier Specific Data* yang terdiri dari *Source Address* (*byte* ke-5 dan ke-6), *RSSI* (*byte* ke-7), *Options* (*byte* ke-8), serta *RF data* (*byte* ke-9 hingga ke-n). Didalam *RF Data* inilah data yang kita kirim berada.

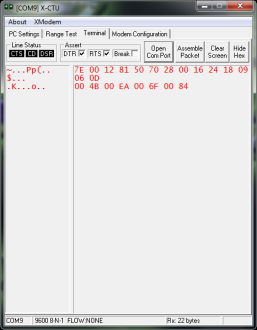
Data yang kita kirim merupakan data yang disimpan dalam variabel *payload* pada modul *router* yang telah kita bahas sebelumnya. Sehingga kita telah mengetahui jumlah *payload* yakni sebanyak 13 *byte*. Sehingga data dari *payload* akan menempati *byte* ke-9 hingga ke-21.

Sampai disini kita dapat menentukan format frame data yang akan diterima oleh *gateway* yaitu sebagai berikut.

|  |  |  |
| --- | --- | --- |
| *byte* ke-1 | = | 0x7E |
| *byte* ke-2 | = | 0x00 |
| *byte* ke-3 | = | 0x12 |
| *byte* ke-4 | = | 0x81 |
| *byte* ke-5 | = | MSB address |
| *byte* ke-6 | = | LSB address |
| *byte* ke-7 | = | RSSI |
| *byte* ke-8 | = | 0x00 |
| *byte* ke-9 | = | jam |
| *byte* ke-10 | = | menit |
| *byte* ke-11 | = | detik |
| *byte* ke-12 | = | tanggal |
| *byte* ke-13 | = | bulan |
| *byte* ke-14 | = | tahun |
| *byte* ke-15 | = | flag Dx |
| *byte* ke-16 | = | Dx |
| *byte* ke-17 | = | flag Dy |
| *byte* ke-18 | = | Dy |
| *byte* ke-19 | = | Flag Dz |
| *byte* ke-20 | = | Dz |
| *byte* ke-21 | = | 0x00 |
| *byte* ke-22 | = | checksum |

Langkah selanjutnya adalah melakukan uji coba membaca data yang diterima oleh modul Xbee, dengan cara menghubungkan modul Xbee ke komputer server kemudian mengamati data yang diterima pada *softwere* terminal. Untuk uji coba ini dapat memanfaatkan X-CTU , CoolTerm atau softwere terminal yang lain yang dapat menampilkan bilangan heksadesimal dari *byte* data yang diterima.

Hasil pengujian menggunakan X-CTU dapat dilihat pada gambar 4.33 dibawah.



*Gambar 4.33 Data yang diterima oleh modul Xbee*

Dari data yang diterima oleh modul Xbee yang ditunjukan pada gambar 4.33 dapat dianalisa sebagai berikut.

*Byte* ke-1 bernilai 0x7E yang berarti*start delimeter*. Kemudian dilanjutkan dengan *byte* ke-2 dan ke-3 secara berturut – turut merupakan MSB dan LSB dari *Data Length* yaitu jumlah *byte* dari *API Identifier*  hingga *bye* terakhir sebelum *checsum* atau dari *byte* ke-4 hingga *byte* ke-21 yang berjumlah 17 *byte*. (dalam format heksadesimal = 0x12).

Modul Xbee yang digunakan pada router terdiri dari 2 buah yang masing – masing memiliki alamat 5070 dan 4D36 sehingga jika *byte* ke-5 digeser ke kiri 8 bit kemudian ditambahkan dengan *byte* ke-6 akan memiliki dua kemungkinan nilai yaitu 0x5070 atau 0x4D36. Nilai ini tergantung dari modul mana data ini dikirimkan. Karena data menunjukan 0x5070 maka modul Xbee pengirim mempunyai alamat 0x5070.

*Byte* ke-7 bernilai 0x28, sedangkan *byte* ini merupakan nilai RSSI negatif dalam skala dBm (desibell miliwat). Sehingga pada pengujian diatas menunjukan bahwa RSSI bernilai -40 dBm.

*Byte* ke-8 bernilai 0x00 menunjukan bahwa opsi pengiriman yang digunakan oleh pengirim secara *reserved* yang artinya data yang dikirim hanya untuk alamat tertentu saja*.* Jika *byte* ini bernilai 0x01 maka pengiriman yang digunakan secara broadcast dalam satu PAN (*Personal Area Network*) sedangkan jika bernilai 0x02 maka opsi pengiriman yang digunakan oleg pengirim adalah secara broadcast antar PAN.

*Byte* ke-9 hingga 14 merupakan data waktu pengiriman data yang didapat dari unit RTC pada modul *router*. Secara berturut – turut data ini menujukan waktu pukul 22:36 .24 tanggal 9/6/13.

*Byte* ke-15, ke-17, dan ke-19 merupakan *flag* atau penanda nilai bilangan pada *byte* selanjutnya yang telah diset sebelumnya pada program pengiriman data pada *router*. Sehingga dari data yang diterima didapat informasi bahwa *byte* ke-16, ke-18 dan ke-20 semuanya bernilai positif.

*Byte* ke-16, ke-18 dan ke-20 secara berturut – turut merupakan data perpindahan pada sumbu X, Sumbu Y dan sumbu Z dan nilainya menunjukan perpindahan sebesar Dx = 75 cm, Dy = 234 cm dan Dz = 111 cm.

*Byte* ke-21 merupakan *byte* yang didapat dari *payload[13]* pada program *router* yang berfungsi untuk menghindari adanya *escape byte*. Sedangkan *byte* ke-22 adalah *byte* terakhir atau *checksum.* *Byte* ini dapat digunakan sebagai pengkoreksi data yang dikirim benar atau tidak. Untuk menguji jikadata dikirim dengan sempurna dilakukan dengan cara menjumlahkan seluruh *byte* yang diterima kecuali *byte* ke-1 sampai *byte* ke-3. Data yang dikirim tidak cacat jika hasil penjumlahan pada 1 *byte* LSB bernilai 0xff. Jadi hasil verifikasi dari data yang diperoleh diatas adalah sebagai berikut.

0x81 + 0x50 + 0x70 + 0x28 + 0x00 + 0x16 + 0x24 + 0x18 + 0x06 + 0x0D + 0x00+ 0x4B + 0x00 + 0xEA + 0x00 + 0x6F + 0x00 + 0x84 = 0x3FF

Nilai 1 *byte* LSB dari hasil perhitungan diatas adalah 0xFF sehingga dapat dipastikan bahwa data yang dikirim tidak ada cacat.

Langkah selanjutnya adalah menyusun algoritma yang akan digunakan untuk memprogram mikrokontroler. Penyusunan algoritma program pada *gateway* didasarkan pada 2 pokok permasalahan yaitu :

1. Jalur komunikasi.
2. Informasi apa saja yang akan dikirim ke komputer server.

Penentuan jalur komunikasi telah dibahas sebelumya yaitu dengan pendekatan software dengan memanfaatkan *library* NewSoftSerial sehingga diawal program perlu dimasukan header NewSoftSerial.h. Setelah memasukan *header* kemudian membuat *object* dengan *class* NewSoftSerial dan menentukan pin Rx dan Tx yang digunakan untuk komunikasi UART. *Class* NewSoftSerial memiliki beberapa *public methods* berikut.

NewSoftSerial(uint8\_t receivePin, uint8\_t transmitPin, bool inverse\_logic = false);

~NewSoftSerial();

void begin(long speed);

void end();

int read();

uint8\_t available(void);

bool active()

bool overflow()

static int library\_version()

static void enable\_timer0(bool enable);

void flush();

Jalur komunikasi yang terbentuk akan digunakan untuk transfer data antara modul Xbee dengan mikrokontroler sehingga *object* ini diberi nama xbee dengan pin Tx pada pin 12 dan pin Rx pada pin 13. Sampai disini kita telah memiliki jalur komunikasi UART pada pin 12 dan pin 13.

Potongan listing program untuk konstruktor jalur UART ini sebagai berikut.

#include <NewSoftSerial.h>

NewSoftSerial xbee (13,12);

Selanjutnya adalah menyiapkan variabel untuk menampung *byte* data yang diterima dari modul Xbee. Mengingat bahwa didalam 1 frame terdapat 22 *byte* data sehingga variabel ini berbentuk *array* dari uint8\_t dengan lebar *array* 22.

Langkah selanjutnya adalah mengisi *array*  dengan *byte* data yang diterima dari modul Xbee dan kemudian memilih data mana saja yang akan dikirim ke komputer server. Proses ini dilakukan dengan membaca *byte* data satu persatu dan memasukanya kedalam tiap – tiap indeks *array*.

Informasi yang perlu ditampilkan di komputer server diantaranya adalah alamat pengirim data, data RSSI, data waktu pengiriman, serta data perpindahan sehingga *array* yang akan diproses nilainya pada indeks 5,6,7,9 hingga 20. Mengingat kembali bahwa data akselerometer memiliki *flag* pada *byte* ke-15, 17 dan 19 maka untuk menentukan tanda bilangan pada data percepatan dilakukan dengan memanfaatkan instruksi if (conditional) dengan parameter yang diuji adalah array yang menunjukan *flag* untuk data perpindahan yakni pada indeks ke-14, 16 dan 18 (indek *array* pertama adalah 0). Potongan program pada algoritma utama penerimaan data dari modul Xbee dan mengirimkan data ke komputer server adalah sebagai berikut.

void **loop**() {

int sX, sY, sZ;

while (xbee.available() > 0) {

int b = readByte();

if (b == 0x7E) {

packet[0] = b;

packet[1] = readByte();

packet[2] = readByte();

int Length = ((packet[1] << 8) | packet[2]);

for(int i=1;i<=(Length+1);i++) {

packet[2+i] = readByte();

}

printPacket(Length+4);

int frametype = packet[3];

if (frametype == 0x81) {

int router = (packet[4] << 8) | packet[5];

**Serial**.print("Router : ");

**Serial**.println(router,HEX);

int RSSI = packet[6];

**Serial**.print("RSSI :");

**Serial**.println(RSSI, HEX);

int hour = packet[8];

int minute = packet[9];

int second = packet[10];

int dayOfMonth = packet[11];

int month = packet[12];

int year = packet[13];

Serial.print(hour,DEC);

Serial.print(":");

Serial.print(minute,DEC);

Serial.print(":");

Serial.print(second,DEC);

Serial.print(" ");

Serial.print(dayOfMonth,DEC);

Serial.print("/");

Serial.print(month,DEC);

Serial.print("/");

Serial.println(year,DEC);

//byte fX = packet[14];

//byte fY = packet[16];

//byte fZ = packet[18];

if (packet[14] == 0x00){

sX = packet[15] ; }

if (packet[14] == 0xff){

sX = packet [15]\*(-1);}

if (packet[16] == 0x00){

sY = packet[17] ; }

if (packet[16] == 0xff){

sY = packet [17]\*(-1);}

if (packet[18] == 0x00){

sZ = packet[19] ; }

if (packet[18] == 0xff){

sZ = packet [19]\*(-1);}

Serial.print("sX :");

Serial.print(sX,DEC);

Serial.print("cm sY :");

Serial.print(sY,DEC);

Serial.print("cm sZ :");

Serial.print(sZ,DEC);

Serial.print("cm ");

Serial.println();

}

}

}

}

int month = packet[12];

int year = packet[13];

**Serial**.print(hour,DEC); **Serial**.print(":");

**Serial**.print(minute,DEC); **Serial**.print(":");

**Seria**l.print(second,DEC); **Serial**.print(" ");

**Serial**.print(dayOfMonth,DEC); **Serial**.print("/");

**Serial**.print(month,DEC); **Serial**.print("/");

**Serial**.println(year,DEC);

if (packet[14] == 0x00){

sX = packet[15] ; }

if (packet[14] == 0xff){

sX = packet [15]\*(-1);}

if (packet[16] == 0x00){

sY = packet[17] ; }

if (packet[16] == 0xff){

sY = packet [17]\*(-1);}

if (packet[18] == 0x00){

sZ = packet[19] ; }

if (packet[18] == 0xff){

sZ = packet [19]\*(-1);}

**Serial**.print("Perpindahan X :");

**Serial**.println(sX,DEC);

**Serial**.print("Perpindahan Y :");

**Serial**.println(sY,DEC);

**Serial**.print("Perpindahan Z :");

**Serial**.println(sZ,DEC);

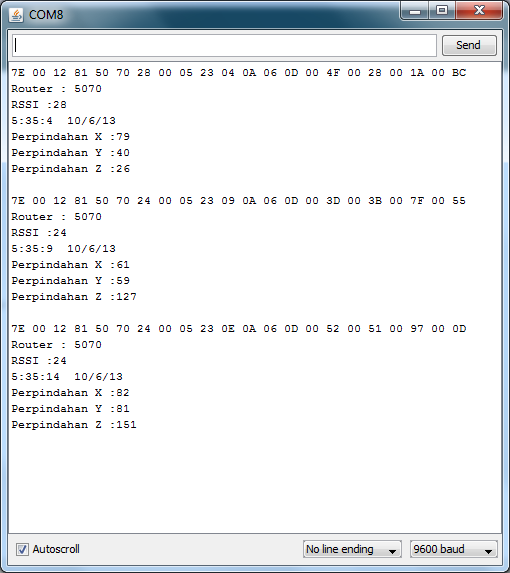
}

}

}

}

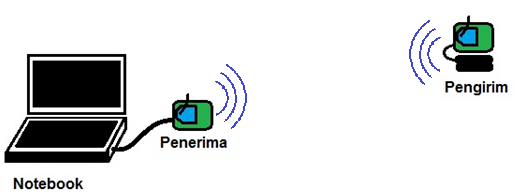
hasil yang didapat pada program pembacaan dan pengiriman data pada *gateway* ditunjukan oleh gambar 4.34 dibawah ini.



*Gambar 4.34 Hasil runing program pada gateway*

## Pengujian RSSI (*Receive Signal Strength Indicator*)

Untuk pengujian RSSI dilakukan dengan 2 cara yakni menguji kekuatan sinyal pada daerah yang tidak ada halangan atau LOS (*Line Of Sight*) dan menguji kekuatan sinyal pada daerah yang terdapat halangan. Skematik yang digunakan untuk pengujian ini adalah sebagai berikut (gambar 4.35 )



*Gambar 4.35 Skematik rangkaian pengujian RSSI*

Perangkat yang dibutuhkan untuk pengujian ini adalah sebagai berikut.

* Notebook sebagai media penampil dan penyimpan data.
* Modul penerima terdiri dari modul Xbee Pro Series 1 dan Arduino.
* Modul pengirim terdiri dari modul Xbee Pro Series 1, Arduino dan batere *lithium* 7,4 Volt sebagai catu daya modul.
* Meteran gulung
* Tali/benang dengan panjang 20 meter.
* Tiang penyangga modul dengan panjang 1,3 meter hingga 1,5 meter.

Untuk seting parameter dari tiap modul Xbee Pro yang digunakan sesuai dengan tabel 4.10 berikut.

Tabel 4.10 Seting parameter Xbee untuk pengujian RSSI

|  |  |  |
| --- | --- | --- |
| Parameter | Pengirim | Penerima |
| BD (*Baud Rate*) | 3 (9600 bps) | 3 (9600 bps) |
| DL (*Destination Address Low)* | 7E63 | FFFF |
| MY (*16 bit Source Address*) | 4D36 | 7E63 |
| PL (*Power Level*) | 4 (10 dBm) | 4 (10 dBm) |

Untuk pengujian RSSI ini berkaitan erat dengan *Power Level*dari modul yang digunakan. Pada parameter PL (Power *Level*) diseting 4 dimana modul yang digunakan adalah Xbee Pro yang varian internasional sehingga nilai 4 berarti *power level* dari modul diseting 10 dBm.

Algoritma yang digunakan untuk pengujian RSSI ini adalah sebagai berikut.

Inisialisasi

Xbee.Avaliable >0

Xbee Kirim Data ke Penerima

Yes

No

Inisialisasi

Xbee.Avaliable >0

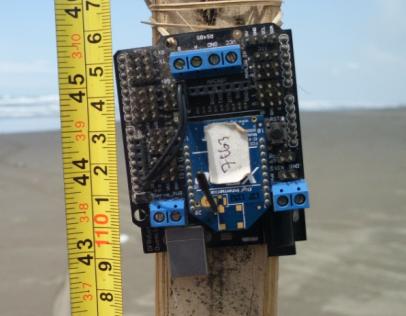
Cetak ke Port Serial COM

Yes

No

*Gambar 4.36 Flowchart algoritma modul penerima (kiri) dan modul pengirim (kanan) yang digunakan pada pengujian RSSI*

Untuk pengujian LOS dilakukan di pantai Widarapayung kecamatan Binangun kabupaten Cilacap. Pengujian dilakukan hingga jarak 500 meter dengan data yang diambil per 20 meter. Untuk menghindari adanya halangan sehingga antara modul pengirim dan penerima benar – benar LOS, tiap modul diletakan pada ketinggian 1 meter hingga 1,3 meter diatas tanah (gambar 4.37).



*Gambar 4.37 Tinggi modul dari* permukaan *tanah (kiri : Pengirim, kanan : Penerima)*

Data yang diperoleh dari pengujian ini adalah sebagai berikut (Tabel 4.11).

Tabel 4.11 Data pengujian RSSI modul terhubung secara LOS

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| NO | Jarak (m) | RSSI Sample ke -n (dalam Heksadesimal) | | | | | | | | | | | RSSI | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Rata - Rata | Rata - Rata | |
| 1 | 20 | 3C | 3C | 3C | 3C | 3C | 3C | 3C | 3C | 3C | 3C | 3C | -60 | dBm |
| 2 | 40 | 49 | 4C | 4D | 4D | 4D | 4D | 4D | 4D | 4E | 4A | 4C | -76 | dBm |
| 3 | 60 | 4F | 4D | 4F | 4F | 50 | 50 | 4F | 55 | 53 | 4D | 4F | -79 | dBm |
| 4 | 80 | 54 | 57 | 58 | 62 | 5F | 5D | 59 | 58 | 62 | 53 | 5A | -90 | dBm |
| 5 | 100 | 5D | 59 | 62 | 5A | 59 | 5F | 62 | 65 | 57 | 5D | 5D | -93 | dBm |
| 6 | 120 | 62 | 5F | 62 | 62 | 5D | 5F | 5D | 5F | 65 | 5F | 60 | -96 | dBm |
| 7 | 140 | 56 | 5A | 62 | 58 | 62 | 5A | 65 | 5D | 62 | 65 | 5E | -94 | dBm |
| 8 | 160 | 65 | 62 | 62 | 62 | 62 | 62 | 65 | 5F | 65 | 65 | 62 | -98 | dBm |
| 9 | 180 | 5D | 65 | 62 | 62 | 62 | 65 | 62 | 62 | 5C | 62 | 61 | -97 | dBm |
| 10 | 200 | 5D | 62 | 65 | 65 | 62 | 5F | 62 | 65 | 62 | 62 | 62 | -98 | dBm |
| 11 | 220 | 62 | 65 | 65 | 65 | 65 | 65 | 65 | 65 | 65 | 65 | 64 | -100 | dBm |
| 12 | 240 | 62 | 59 | 54 | 56 | 54 | 54 | 55 | 5C | 58 | 54 | 57 | -87 | dBm |
| 13 | 260 | 65 | 5C | 58 | 58 | 59 | 5A | 5C | 5C | 5A | 5A | 5B | -91 | dBm |
| 14 | 280 | 53 | 57 | 52 | 57 | 54 | 52 | 51 | 50 | 4F | 52 | 52 | -82 | dBm |
| 15 | 300 | 59 | 59 | 57 | 57 | 57 | 57 | 57 | 54 | 54 | 56 | 56 | -86 | dBm |
| 16 | 320 | 5D | 5A | 5C | 5A | 5A | 5D | 5F | 5C | 5D | 62 | 5C | -92 | dBm |
| 17 | 340 | 55 | 53 | 50 | 56 | 58 | 5A | 62 | 5C | 5F | 5D | 59 | -89 | dBm |
| 18 | 360 | 62 | 5D | 62 | 5F | 5F | 5D | 5D | 5F | 62 | 5F | 5F | -95 | dBm |
| 19 | 380 | 65 | 5D | 5F | 62 | 62 | 5C | 5D | 5C | 5D | 5C | 5E | -94 | dBm |
| 20 | 400 | 62 | 62 | 5D | 65 | 65 | 65 | 65 | 65 | 65 | 65 | 63 | -99 | dBm |
| 21 | 420 | 65 | 65 | 62 | 62 | 65 | 5D | 62 | 62 | 62 | 65 | 62 | -98 | dBm |
| 22 | 440 | 5F | 5C | 62 | 62 | 62 | 5F | 5F | 62 | 5D | 5F | 5F | -95 | dBm |
| 23 | 460 | 62 | 62 | 5F | 62 | 65 | 65 | 65 | 62 | 62 | 65 | 62 | -98 | dBm |
| 24 | 480 | 65 | 62 | 62 | 62 | 5F | 62 | 62 | 62 | 55 | 62 | 60 | -96 | dBm |
| 25 | 500 | 65 | 65 | 65 | 65 | 62 | 62 | 62 | 65 | 62 | 65 | 63 | -99 | dBm |

Berdasarkan *data sheet* Xbee Pro Series 1 (Maxstream, 2007) sensitivitas penerimaan sinyalnya adalah dari -36 dBm hingga -100 dBm sehingga dari data yang ada dapat dibuat grafik persentase RSSI terhadap jarak dengan perhitungan sebagai berikut.

Dan hasilnya adalah sebagai berikut.(gambar 4.38)

*Gambar 4.38 Grafik perbandingan nilai RSSI terhadap jarak*

Pengujian RSSI berikutnya adalah pengujian modul pada daerah berpenghalang. Pengujian ini dilakukan di pesisir Pantai Widara Payung dimana daerah tersebut merupakan daerah perkebunan yang ditanami pohon kelapa dan dibawahnya ditanami palawija serta ada beberapa yang tidak ditanami palawija sehingga ditumbuhi tumbuhan liar dan perdu. Gambar 4.39 menunjukan lokasi pengujian RSSI untuk daerah yang berpenghalang.



*Gambar 4.39 Lokasi pengujian nilai RSSI pada medan berpenghalang*

Data RSSI yang diambil tiap 10 meter dan dilakukan hingga tidak ada data yang terkirim. Dari hasil percobaan, jarak terjauh data dapat diterima oleh modul penerima adalah 140 meter.

Data hasil pengujian ditunjukan oleh tabel 4.12 dan grafik perbandingan nilai RSSI terhadap jarak ditunjukan oleh gambar 4.40.

Tabel 4.12 Data hasil pengujian RSSI pada medan berpenghalang

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| NO | Jarak (m) | RSSI Sample ke -n (dalam Heksadesimal) | | | | | | | | | | | RSSI | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Rata - Rata | Rata - Rata | |
| 1 | 10 | 42 | 42 | 42 | 3F | 3F | 40 | 3C | 43 | 42 | 47 | 41 | -65 | dBm |
| 2 | 20 | 4F | 49 | 4C | 4E | 46 | 4E | 47 | 46 | 44 | 4F | 4A | -74 | dBm |
| 3 | 30 | 4F | 50 | 50 | 58 | 4A | 49 | 4D | 51 | 49 | 4E | 4E | -78 | dBm |
| 4 | 40 | 58 | 56 | 53 | 52 | 52 | 59 | 53 | 52 | 5F | 53 | 55 | -85 | dBm |
| 5 | 50 | 54 | 5F | 57 | 5F | 56 | 54 | 4D | 62 | 59 | 57 | 58 | -88 | dBm |
| 6 | 60 | 4F | 58 | 59 | 56 | 59 | 4C | 5A | 52 | 53 | 5F | 55 | -85 | dBm |
| 7 | 70 | 62 | 5D | 65 | 65 | 62 | 59 | 65 | 65 | 65 | 59 | 61 | -97 | dBm |
| 8 | 80 | 59 | 5D | 5A | 5F | 59 | 5D | 5C | 65 | 5F | 65 | 5D | -93 | dBm |
| 9 | 90 | 5D | 5D | 59 | 5D | 5A | 62 | 58 | 65 | 57 | 5A | 5C | -92 | dBm |
| 10 | 100 | 5A | 5F | 58 | 5D | 5A | 5A | 5C | 5C | 5D | 53 | 5A | -90 | dBm |
| 11 | 110 | 65 | 5F | 65 | 65 | 5F | 65 | 62 | 62 | 5F | 62 | 62 | -98 | dBm |
| 12 | 120 | 62 | 5D | 65 | 5F | 58 | 5C | 5D | 62 | 62 | 5D | 5E | -94 | dBm |
| 13 | 130 | 65 | 65 | 5F | 65 | 62 | 65 | 65 | 62 | 62 | 65 | 63 | -99 | dBm |
| 14 | 140 | 65 | 5F | 5F |  |  |  |  |  |  |  | 61 | -97 | dBm |

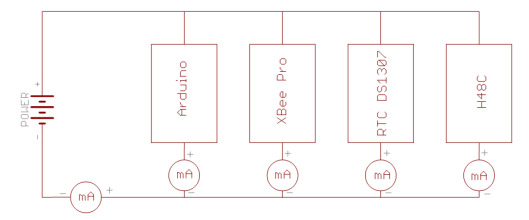
*Gambar 4.40 Grafik perbandingan nilai RSSI terhadap jarak pada medan berpenghalang*

## Konsumsi Daya

Salah satu aspek terpenting dalam jaringan sensor nirkabel adalah pengetahuan mengenai konsumsi daya dari modul yang digunakan. Hal ini dikarenakan *node* akan diletakan didaerah tertentu dan terkadang jauh jala – jala PLN. Sehingga informasi konsumsi daya penting untuk menentukan tipe catu daya yang akan digunakan khususnya konsumsi daya *router.*

Dalam sebuah *node router* terdiri dari 4 bagian utama yaitu kontroler, unit RF, unit RTC dan Sensor. Sehingga untuk mengetahui konsumsi daya sebuah *router* diperlukan pengukuran pada masing – masing bagian.

Pengukuran daya dapat dilakukan dengan mengukur konsumsi arus modul. Skematik pengukuran arus dari sebuah *router* ditunjukan oleh gambar 4.41 dibawah.



*Gambar 4.41 Skematik pengukuran daya dari sebuah router*

Pada pengukuran digunakan catu daya berupa 2 buah sel baterai *Lithium Polimer* dengan tegangan masing – masing 3.6 V dengan kapasitas 2600 mAH dan disusun seri sehingga tegangan catu daya keseluruhan adalah 7.2 V.

Pengukuran dilakukan pada modul yang sedang menjalankan program *router* dan mengirimkan data melalui modul RF ke *gateway* dengan interval pengiriman data 5 detik.

Hasil yang didapat dari pengukuran ditunjukan pada tabel 4.13 dibawah.

Tabel 4.13 Hasil pengukuran konsumsi arus modul *router*

|  |  |
| --- | --- |
| Unit | Arus |
| Kontroler ( Arduino) | 20 mA |
| RF ( Xbee Pro ) | 62 mA |
| RTC ( DS1307 ) | 2 mA |
| Akselerometer ( H48C ) | 1 mA |
| Arus Total | 85 mA |

Dengan arus total 85 mA dan tegangan kerja adalah 7.2 V sehingga konsumsi daya modul adalah 612 mW. Dengan kapasitas baterai 2600 mAH idealnya baterai dapat mensuplai modul selama 30.5 jam akan tetapi pada kenyataanya dapat bervariasi.

Jika menggunakan baterai sebagai catu daya modul, kapasitas baterai sangat menentukan lamanya waktu kerja modul. Sehingga untuk membuat modul bekerja lebih lama dapat dilakukan dengan beberapa cara seperti menambah kapasitas baterai, meminimalisir kerja kontroler, serta memperpanjang jeda waktu pengiriman data.

Menambah kapasitas baterai dapat dilakukan dengan memilih baterai yang kapasitasnya tinggi atau menggabungkan beberapa sel baterai secara paralel. Akan tetapi baterai memiliki hambatan dalam yang menyebabkan baterai mengalami *self-discharge* meskipun baterai tidak digunakan.

Semakin tinggi kerja kontroler menyebabkan kontroler mengkonsumsi arus lebih banyak sehingga untuk memperkecil konsumsi arus dapat dilakukan dengan membuat kerja kontroler seringan mungkin. Ini dapat dilakukan dengan beberapa cara yaitu membuat algoritma yang lebih ringkas, menghindari adanya operasi matematikal yang kompleks, mengganti operasi pembagian dengan operasi perkalian atau dengan operasi penggeseran bit jika mungkin, mengurangi penggunaan tipe *float* karena tipe ini membutuhkan alokasi 32 bit untuk *single precision* dan 64 bit untuk *double precision* sehingga melibatkan operasi bilangan pada *float* akan lebih lama dari pada operasi pada bilangan bulat (*integer*).

Selang waktu pengiriman data berpengaruh sangat besar pada konsumsi daya dari sebuah modul karena modul RF membutuhkan arus yang tinggi pada saat mengirim data. XbeePro sendiri membutuhkan arus 250 mA untuk melakukan pengiriman data dan hanya 55 mA pada saat *idle*. Sehingga semakin cepat interval pengiriman data akan berakibat pada semakin besarnya konsumsi daya dari modul.