BAB 5 PERANCANGAN DAN IMPLEMENTASI

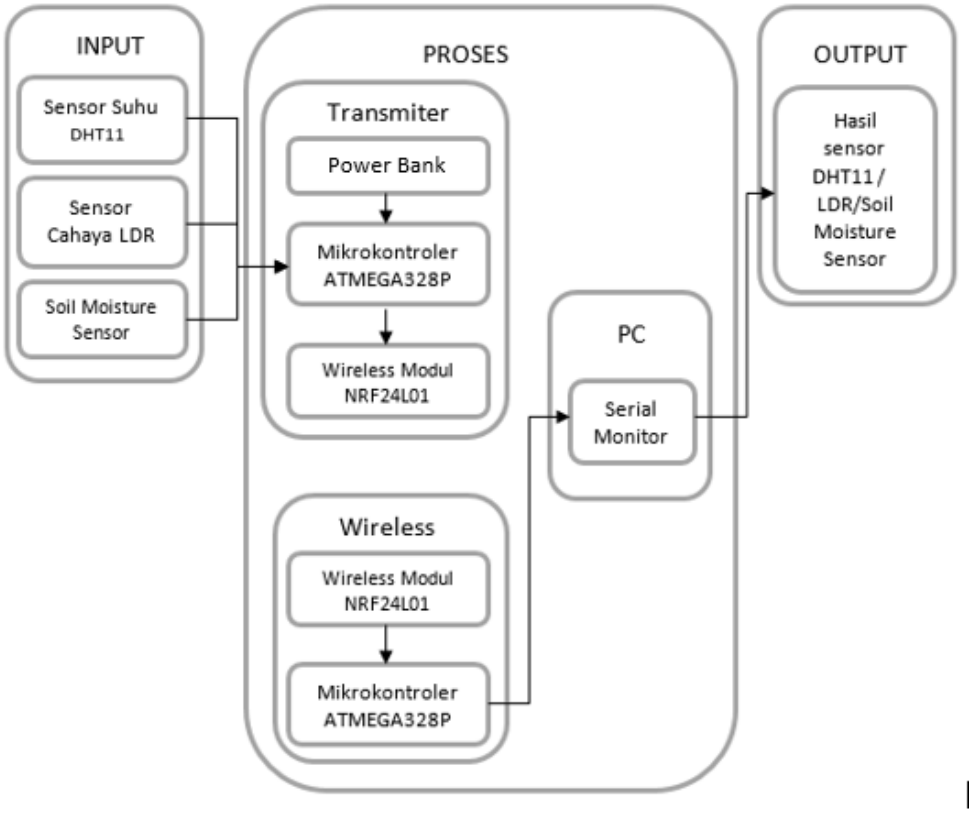
Bab ini menjelaskan tentang perancangan dan juga implementasi sistem, yang mana merupakan tindak lanjut dari rekayasa kebutuhan yang telah dibuat sebelumnya dengan tujuan agar sistem dirancang dan diimplementasikan sesuai dengan harapan.

5.1 Perancangan sistem

Pada tahap perancangan menjelaskan terkait tahapan perancangan *sensor node* meliputi perancangan perangkat keras dan perancangan perangkat lunak agar sistem tersebut dapat bekerja dengan tepat.

5.1.1 Gambaran umum sistem

Perancangan sistem terdiri dari dua bagian, yaitu *transmitter sensor node* dan *receiver sensor node*. Gambar 5.1 merupakan diagram blok sistem yang telah dirancang.



Gambar 5.1 Diagram blok sistem

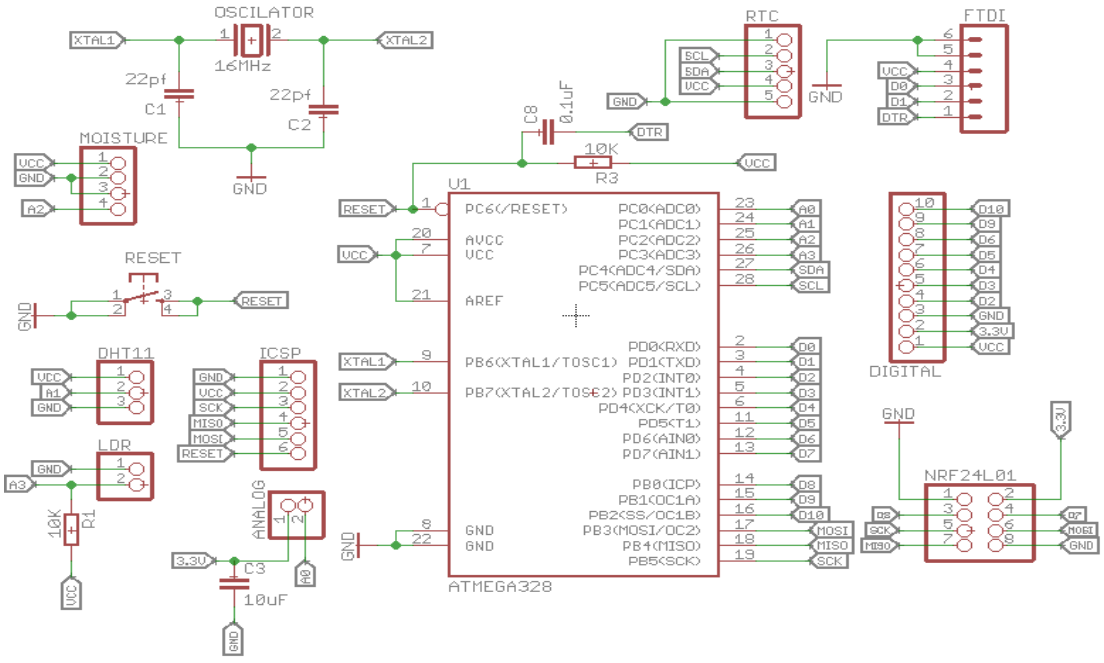
Diagram blok diatas merupakan gambaran umum terkait rancangan sistem yang dibuat, terdapat tiga bagian berupa *input*, proses, dan *output*. Pada bagian *input* pada *transmitter sensor node* terdapat modul sensor DHT11, LDR, dan *Soil Moisture.* Modul sensor DHT11 difungsikan hanya untuk mengakuisisi data suhu, modul sensor LDR (*Light Dependent Resistor*) untuk mengakuisisi data intensitas

cahaya, sedangkan modul sensor *Soil Moisture* untuk mengakuisisi data kelembaban tanah. Setiap data hasil *sensing* dari setiap sensor diteruskan pada mikrokontroler Atmega328P untuk diproses dan untuk menjalankan mekanisme *low power* dengan hasil keputusan apakah data hasil *sensing* harus dikirimkan pada *receiver sensor node* ataukah *transmitter sensor node* harus memasuki *sleep mode*. Pengiriman data dilakukan menggunakan modul NRF24L01 secara nirkabel melalui gelombang radio pada frekuensi 2,4 GHz.

Pada saat sebelum melakukan pengiriman data, terlebih dahulu mikrokontroler memberikan *time stamp* berdasarkan data waktu dan penanggalan yang tersimpan di modul RTC. Selain itu *time stamp* juga diberikan pada saat mikrokontroler memutuskan untuk *transmitter sensor node* untuk memasuki *sleep mode*. Mekanisme *low power* dimulai dengan membandingkan apakah data yang diterima mikrokontroler dari masing-masing sensor sama dengan data hasil *sensing* sebelumnya, dan rentang waktu *sleep mode* berlangsung selama satu menit. *Receiver sensor node* menerima data melalui media komunikasi nirkabel dengan modul NRF24L01, ketika mendeteksi adanya data mikrokontroler akan mengambil data tersebut untuk diproses. Proses yang dimaksudkan adalah proses *decoding* data tersebut untuk dapat ditampilkan melalui *serial monitor* dengan menggunakan modul FTDI *Break-out* menuju *personal computer*.

5.1.2 Perancangan *transmitter sensor node*

Perancangan *transmitter sensor node* terdiri dari beberapa modul yang dirangkai menjadi satu agar sistem dapat berjalan sesuai dengan tujuan. Gambar skematik diagram dari rangkaian *transmitter sensor node* ditunjukkan pada Gambar 5.2.



Gambar 5.2 Skematik diagram *transmitter sensor node*

Pada rangkaian *transmitter sensor node* terdapat beberapa komponen di antaranya adalah mikrokontroler ATmega328P, modul *USB-to-TTL FTDI Break-out*, modul NRF24L01, modul sensor DHT11, modul sensor LDR, modul sensor *Soil Moisture*, modul RTC, 16MHz crystal, 1 buah capasitor 0,1µf ,1 buah capasitor 10µf , 2 buah capasitor 22pf , dan 2 buah resistor 10k Ω.

5.1.2.1 Perancangan modul NRF24L01

Pada *transmitter sensor node* modul NRF24L01 digunakan sebagai media komunikasi nirkabel yang digunakan untuk mengirimkan data hasil *sensing* menuju *receiver sensor node*. Dalam perancangannya diperlukan tegangan sebesar 3,3 V yang diambil dari *suply* tegangan FTDI *Break-out*, namun di antara 3,3 V dan GND pada modul perlu ditambahkan sebuah kapasitor dengan nilai kapasitansi sebesar 10µf. Hal tersebut sebagai tindakan preventif apabila *suply* tegangan 3,3 V dari FTDI tidak stabil yang dapat berdampak pada performa komunikasi *sensor node*. Selain itu untuk dapat digunakan diperlukan Arduino Mirf *library* yang dipasangkan pada program, dan juga diperlukan konfigurasi pin yang berada pada Tabel 5.1.

Tabel 5.1 Keterangan Pin NRF24L01

Pin Modul NRF24L01	Pin ATmega328P
CSN	13
CE	14
MOSI	17
MISO	18
SCK	19
GND	GND
VCC	-
IRQ	-

5.1.2.2 Perancangan modul RTC

Modul RTC pada *transmitter sensor node* digunakan sebagai media penyedia data waktu dan penanggalan yang mana data tersebut diperlukan oleh *sensor node*. Untuk dapat digunakan diperlukan penambahan *library* Arduino RTClib pada program. Modul RTC memiliki 5 pin yang terdiri pin DS, SCL, SDA, GND, dan VCC dengan konfigurasi pin mikrokontroler yang berada pada Tabel 5.2.

Tabel 5.2 Keterangan Pin Modul RTC

Pin Modul RTC	Pin ATmega328P
SCL	28
SDA	27
GND	GND
VCC	VCC

5.1.2.3 Perancangan modul sensor DHT11

Modul sensor DHT11 digunakan oleh *transmitter sensor node* untuk mengakuisisi data *sensing* berupa besaran suhu mulai dari 0 ^{0}C hingga 50 ^{0}C. Cara kerja dari modul tersebut yakni dimulai dengan proses *sensing* oleh modul, yang kemudian hasilnya diteruskan pada mikrokontroler. Setelah mikrokontroler menerima *input*-an melalui pin DATA pada modul sensor, *input*-an tersebut diolah dengan menggunakan *library* DHTlib sehingga dapat memanifestasikan *input*-an menjadi data suhu dengan sesuai. Adapun konfigurasi pin modul sensor berada pada Tabel 5.3.

Tabel 5.3 Keterangan Pin Sensor DHT11

Pin Sensor DHT11	Pin ATmega328P
DATA	24
GND	GND
VCC	VCC

5.1.2.4 Perancangan modul sensor *soil moisture*

Pada *transmitter sensor node* modul sensor *soil moisture* merupakan modul yang berfungsi untuk mengakuisisi data kelembaban tanah mulai dari 0% hingga 100%. Secara garis besar modul sensor tersebut terdiri dari dua buah *probe sensor* dan modul pengkondisian sinyal dengan empat pin yang terdiri dari pin VCC, GND, ANALOG, dan DIGITAL. Konfigurasi pin pada rangkaian modul sensor dengan mikrokontroler ditunjukkan pada Tabel 5.4.

Tabel 5.4 Keterangan Pin Sensor *Soil Moisture*

Pin Sensor *Soil Moisture*	Pin ATmega328P
ANALOG	25
DIGITAL	-
GND	GND
VCC	VCC

5.1.2.5 Perancangan modul sensor LDR

Modul sensor LDR (*Light Dependent Resistor*) merupakan modul yang digunakan oleh *transmitter sensor node* untuk mengakuisisi data intensitas cahaya yang memiliki satuan lx. Pada perancangannya modul sensor yang digunakan adalah modul sensor LDR yang hanya berupa resistor saja, tanpa menggunakan modul pengkondisian sinyal. Seperti halnya resistor biasa, modul sensor yang digunakan memiliki dua kaki resistor yang mana diperlukan penambahan resistor dengan nilai resistansi sebesar 10k Ω. Modul sensor dipasangkan dengan resistor secara paralel terhadap pin data yang akan mengalirkan data hasil *sensing*, yang ditunjukkan pada Tabel 5.5

Tabel 5.5 Keterangan Pin Sensor LDR

Pin Sensor LDR	Pin ATmega328P
Kaki Resistor 10k	VCC
DATA	26
Kaki Sensor LDR	GND

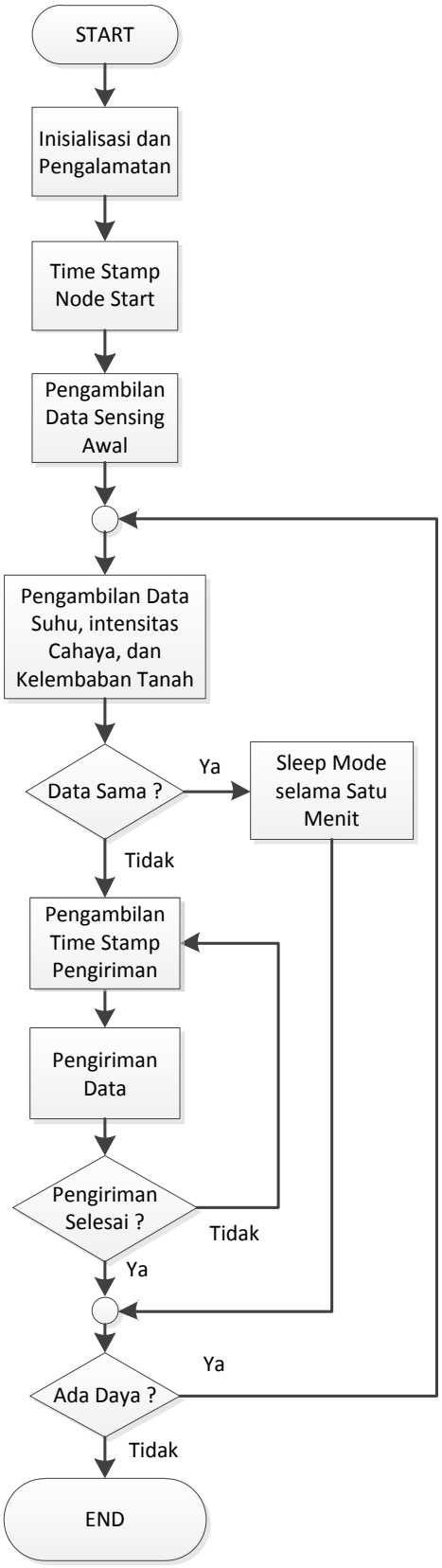
5.1.2.6 Perancangan modul FTDI *Break-out*

Pada *transmitter sensor node* modul FTDI *Break-out* digunakan sebagai sumber tegangan dari *sensor node*. Yang mana modul tersebut akan menerima *input*-an berupa sumber energi yang kemudian ditransformasikan menjadi tegangan 5V, terdapat dua pilihan tegangan yang dapat dilairkan yakni tegangan 5V dan tegangan 3,3V yang dipilih dengan menggunakan *jumper.* Pada modul tersebut memiliki enam buah pin utama dan juga beberapa pin tambahan yang dapat digunakan, pin utama yang dimaksud terdiri dari pin DTR, RX, TX, VCC, CTS, dan GND, disamping itu dalam perancangan sistem juga menggunakan pin tambahan untuk mengalirkan tegangan 3,3V. Tabel 5.6 menjelaskan bagaimana konfigurasi pin utama FTDI *Break-out* dengan mikrokontroler.

Tabel 5.6 Keterangan Pin FTDI *Break-out*

Pin FTDI *Break-out*	Pin Atmega328P
DTR	1
RX	3
TX	2
VCC	VCC
CTS	-
GND	GND

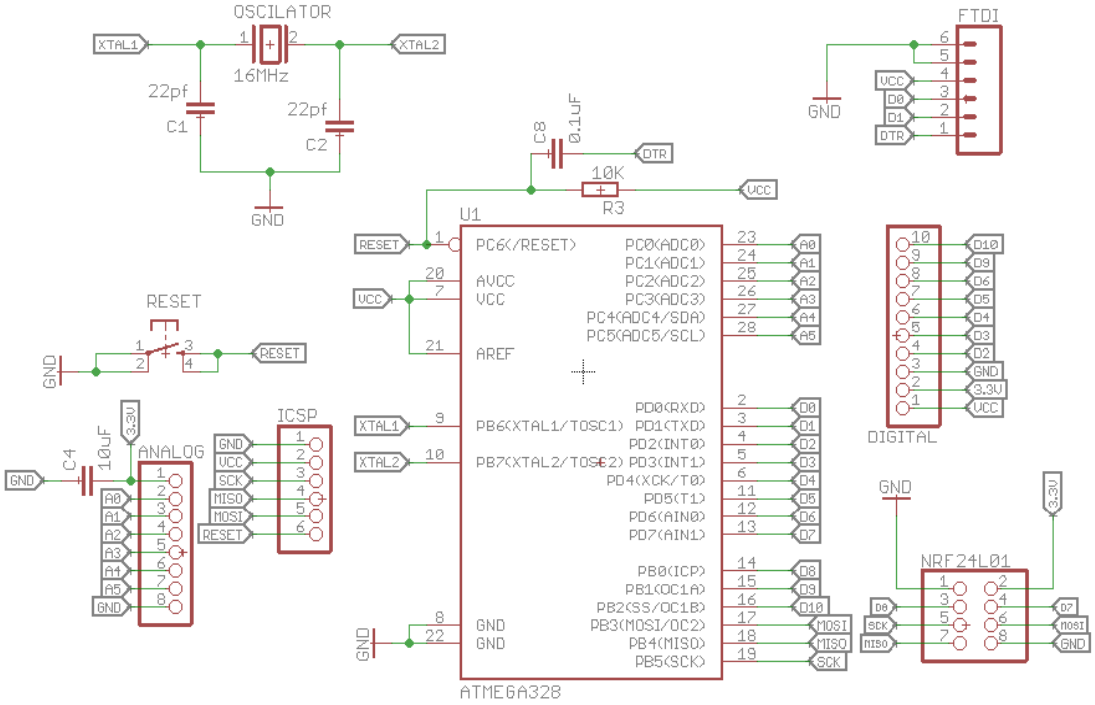
Alur berjalannya sistem pada *transmitter sensor node* ditampilkan dalam bentuk *flowchart diagram* yang berada pada Gambar 5.3. Pertama sistem melakukan inisialisasi *node* dengan memberikan alamat pengiriman pada *sensor node,* setelah itu mikrokontroler mengambil data waktu pada RTC (*Real Time Clock*) untuk mengetahui waktu pada saat ini sebagai *time stamp* awal berjalannya *sensor node*. Kemudian *sensor node* melakukan akuisisi data *sensing* untuk pertama kali sejak *sensor node* mulai berjalan. Pada proses akuisisi data berikutnya *sensor node* mulai menjalankan mekanisme *low power* dengan melakukan perbandingan apakah data yang diambil masih sama hasilnya dengan data sebelumnya, apabila sama maka *sensor node* memasuki *sleep mode* selama satu menit, namun jika tidak maka data tersebut kemudian dikirimkan. Setiap kali *sensor node* melakukan pengiriman maupun sebelum memasuki *sleep mode*, *sensor node* akan memberikan *time stamp* yang diambil dari modul RTC.



Gambar 5.3 Flowchart diagram *transmitter sensor node*

5.1.3 Perancangan *receiver sensor node*

Secara garis besar dalam perancangan *receiver sensor node* tidak jauh beda dengan *transmitter sensor node*, perbedaannya terletak pada modul RTC dan modul sensor yang mana pada *transmitter sensor node* dilengkapi dengan modul RTC dan juga tiga buah modul sensor, sedangkan pada *receiver sensor node* ini tidak. Skematik diagram rangkaian pada *receiver sensor node* di tunjukkan pada Gambar 5.4.



Gambar 5.4 Skematik diagram *receiver sensor node*

5.1.3.1 Perancangan modul NRF24L01

Pada *receiver sensor node* modul NRF24L01 digunakan sebagai media komunikasi nirkabel yang berfungsi untuk menerima data hasil *sensing* dari *transmitter sensor node*. Dalam perancangannya diperlukan tegangan sebesar 3,3V yang diambil dari *suply* tegangan FTDI *Break-out*, namun di antara 3,3 V dan GND pada modul perlu ditambahkan sebuah kapasitor dengan nilai kapasitansi sebesar 10µf. Hal tersebut sebagai tindakan preventif apabila *suply* tegangan dari FTDI kurang stabil yang dapat berdampak pada performa komunikasi *sensor node*. Selain itu untuk dapat digunakan diperlukan Arduino Mirf *library* yang dipasangkan pada program, dan juga diperlukan konfigurasi pin yang ditunjukkan pada Tabel 5.7.

Tabel 5.7 Keterangan Pin NRF24L01

Pin Modul NRF24L01	Pin ATmega328P
CSN	13
CE	14

MOSI	17
MISO	18
SCK	19
GND	GND
VCC	-
IRQ	-

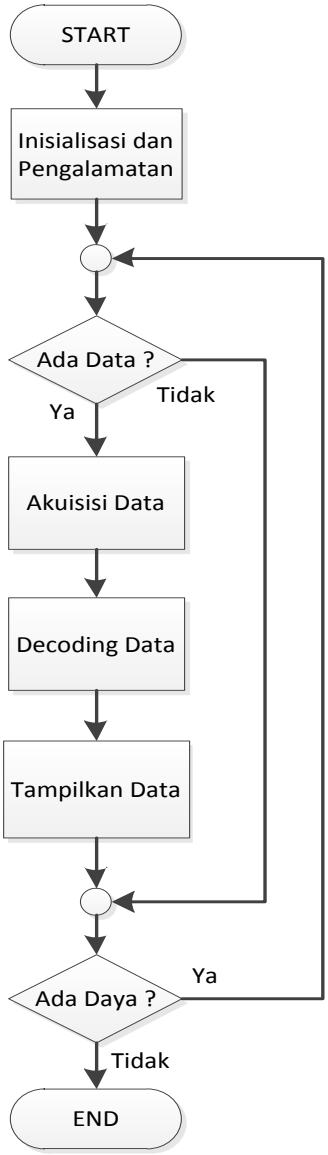
5.1.3.2 Perancangan modul FTDI *Break-out*

Pada *receiver sensor node* modul FTDI *Break-out* digunakan sebagai modul komunikasi serial antara *sensor node* dan PC (*Personal Computer*) untuk keperluan *monitoring* hasil *sensing*. Selain sebagai modul komunikasi serial, seperti halnya fungsi modul tersebut pada *transmitter sensor node,* yakni digunakan sebagai penyedia sumber energi bagi *sensor node.* Namun yang membedakan adalah *input*-an energi yang diterima oleh modul ini pada *receiver sensor* node adalah dari port USB (*Universal Serial Bus*) pada PC. Pada modul tersebut memiliki enam buah pin utama dan juga beberapa pin tambahan yang dapat digunakan, pin utama yang dimaksud terdiri dari pin DTR, RX, TX, VCC, CTS, dan GND, disamping itu dalam perancangan sistem juga menggunakan pin tambahan untuk mengalirkan tegangan 3,3V. Tabel 5.8 menjelaskan bagaimana konfigurasi pin utama FTDI *Break-out* dengan mikrokontroler.

Tabel 5.8 Keterangan Pin FTDI *Break-out*

Pin FTDI *Break-out*	**Pin ATmega328P**
DTR	1
RX	3
TX	2
VCC	VCC
CTS	-
GND	GND

Alur jalannya sistem pada *receiver sensor node* disajikan dalam bentuk *flowchart diagram* yang berada pada Gambar 5.5. Pertama sistem melakukan inisialisasi *node* beserta pengaturan alamat penerimaan data, setelah itu *node* menyatakan untuk siap menerima data. Kemudian *sensor node* menunggu hingga mendeteksi adanya data yang diterima, data yang diterima kemudian diambil dan di-*decode* oleh mikrokontroler agar dapat diketahui apakah data yang diterima merupakan data hasil sensing suhu, intensitas cahaya, atau kelembaban tanah dan kemudian ditampilkan melalui *serial monitor* pada *personal computer*. Apabila tidak ada data yang diterima maka *sensor node* akan terus menunggu hingga *sensor node* dimatikan.



Gambar 5.5 Flowchart diagram *receiver sensor node*

5.1.4 Perancangan mekanisme *Low Power*

Perancangan mekanisme *low power* bertujuan untuk mengaplikasikan *sleep mode* serta meminimalisir beban kerja dan panjang durasi aktif *sensor node*. Cara kerja mekanisme *low power* yakni dengan membandingkan hasil *sensing* terbaru dengan hasil *sensing* sebelumnya, yang kemudian menghasilkan keputusan apakah *sensor node* akan memasuki *sleep mode* atau akan mengirimkan data. Mikrokontroler berada pada *sleep mode* dengan durasi satu menit sejak mekanisme *low power* menghasilkan keputusan agar *sensor node* memasuki *sleep mode*.

Untuk dapat menerapkan mekanisme *low power*, dibutuhkan Arduino *Lightweight Low Power library* yang diaplikasikan ke dalam program agar *sensor node* dapat menjalankan *sleep mode*. Program yang telah berisi mekanisme *low power* beserta *library*-nya di-*upload* pada *transmitter sensor node.* Agar dapat

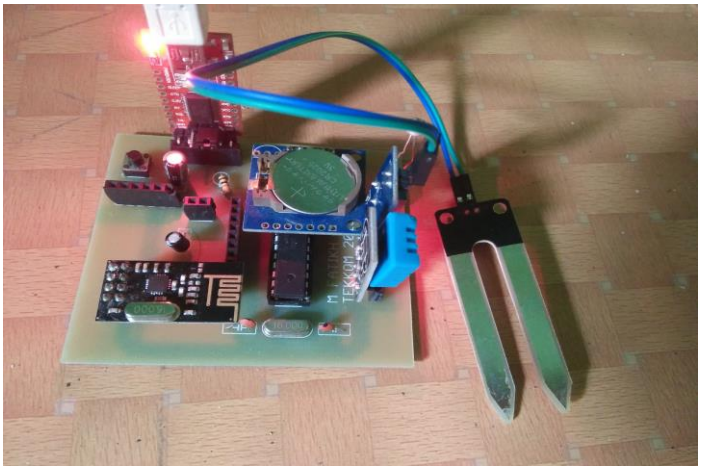
diaktifkan maka didalam program yang di-*upload* diperlukan fungsi untuk membandingkan hasil *sensing* dari masing masing *sensor node*, menginisialisasi penggunaan *library* yang dibutuhkan, dan memanggil fungsi "LowPower" dengan pilihan mode "poweDown" yang dijalankan selama 4s dengan menonaktifkan ADC (*Analog to Digital Converter*) dan BOD (*Brown-out Detection*) sehingga ketika fungsi tersebut diulang selama 15 kali, maka akan menempatkan mikrokontroler pada mode *power down* dengan ADC dan BOD yang dinonaktifkan selama satu menit.

5.2 Implementasi sistem

Tahapan implementasi sistem dapat dilaksanakan apabila proses perancangan sistem telah terpenuhi, hal tersebut dikarenakan tahapan perancangan merupakan acuan dalam implementasi sistem. Pada bagian ini menjelaskan tentang pembahasan pengimplementasian sistem pada *transmitter sensor node* dan *receiver sensor node.*

5.2.1 Implementasi sistem pada *transmitter sensor node*

Implementasi sistem pada *transmitter sensor node* dilakukan sesuai dengan pembahasan perancangan sistem. Dalam perangkaian *sensor node* disesuaikan dengan perancangan sistem, yang mana rangkaian tersebut terdiri dari mikrokontroler ATmega328P, modul *USB-to-TTL* FTDI *Break-out,* modul komunikasi nirkabel NRF24L01, modul RTC sebagai penyedia data waktu dan penanggalan, modul sensor DHT11, modul sensor LDR, dan modul sensor *soil moisture*. Gambar 5.6 menunjukan hasil implementasi sistem pada *transmitter sensor node*.



Gambar 5.6 Implementasi sistem pada *transmitter sensor node*

Sedangkan dalam implementasi sistem pada bagian perangkat lunak, diperlukan Arduino IDE (*Integrated Development Environment*) yang mendukung bahasa pemrograman C, selain itu juga dibutuhkan beberapa *library* yang menunjang berjalannya setiap fungsi modul pada *sensor node*. *Library* tersebut adalah Arduino Mirf *library* + SPI.h untuk menunjang penggunaan modul NRF24L01, Arduino RTClib + Wire.h untuk menunjang penggunaan modul RTC,

Arduino DHTlib untuk menunjang penggunaan modul sensor DHT11, dan Arduino *Lightweight Low Power library* untuk menunjang pengaplikasian mekanisme *low power* pada sistem. Penggunaan *library* diatas dapat dilihat pada Gambar 5.7.

node_send_mirf02

```
#include <Mirf.h>
#include <nRF24L01.h>
#include <MirfHardwareSpiDriver.h>
#include <LowPower.h>
#include <RTClib.h>
#include <SPI.h>
#include <Wire.h>
#include <dht.h>
```

Gambar 5.7 Library yang digunakan pada *transmitter sensor node*

```
dht DHT;
#define DHT11_PIN A1
RTC_DS1307 rtc;

char daysOfTheWeek[7][12] = {"Minggu", "Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu"};
int dht[2], ldr[2], mois[2]; //inisialisasi variable hasil sensing
int ai[3]; //inisialisasi marking variable
int start[2];
int msg[8];
int LDR_Pin = A3;
int Moisture_Pin = A2;

void setup(void) {
  Serial.begin(57600);
  pinMode(LDR_Pin, INPUT);
  pinMode(Moisture_Pin, INPUT);
```

Gambar 5.8 Inisialisasi dan konfigurasi modul pada *transmitter sensor node*

Pada Gambar 5.8 menunjukan tentang inisialisasi modul, variabel, dan konfigurasi pin sensor yang digunakan pada *transmitter sensor node* yang berfungsi untuk menunjang berjalannya sistem. Diantara variabel yang digunakan yakni *array integer* "dht[2], ldr[2], mois[2], ai[3], start[2], msg[8]" kemudian pin yang digunakan untuk modul sensor yakni pin A1 untuk modul sensor DHT11, pin A2 untuk modul sensor LDR, dan pin A3 untuk modul sensor *Soil Moisture*.

```
if (! rtc.begin()) {
  Serial.println("Couldn't find RTC");
  while (1);
}
rtc.adjust(DateTime(F(__DATE__), F(__TIME__))); //penyesuaian data waktu dan penanggalan
if (! rtc.isrunning()) {
  Serial.println("RTC is NOT running!");

}
Mirf.spi = &MirfHardwareSpi;
Mirf.init();
Mirf.setTADDR((byte *)"FGHIJ"); //inisialisasi alamat pengiriman data
Mirf.payload = sizeof(msg); //inisialisasi payload pengiriman data
Mirf.channel = 90; //inisialisasi chanel komunikasi
Mirf.config();
```

Gambar 5.9 Aktifasi modul RTC dan konfigurasi pengalamatan *node*

Pada Gambar 5.9 menunjukan tentang bagaimana *sensor node* mulai menggunakan modul RTC, yakni dengan melakukan pengecekan apakah RTC dapat berjalan dengan baik ataukah tidak, proses penyesuaian data waktu serta penanggalan yang akan disimpan pada modul RTC untuk dapat diproses oleh mikrokontroler, dan juga konfigurasi pengalamatan pada *transmitter sensor node*.

```
dht[0] = 0; //encoding data suhu
ldr[0] = 1; //encoding data intensitas cahaya
mois[0] = 2; //encoding data persentase kelembaban tanah
for (int i = 0; i < 3; i++) { //instansiasi marking variable
  ai[i] = 0;
}
```

Gambar 5.10 Proses *encoding* data dan inisialisasi *marking variable*

Pada Gambar 5.10 menunjukan terkait proses *encoding* variabel yang digunakan sebagai penampung data sensor yang siap untuk dikirimkan, beserta *marking variable* yang berfungsi untuk menjalankan mekanisme *low power*.

5.2.2 Urutan berjalannya *transmitter sensor node*

Setelah melewati bagian inisialisasi dan *setup*, *transmitter sensor node* kemudian menjalankan *code* program pada Gambar 5.11 dijalankan diawal *sensor node* mulai berjalan yang dimulai dengan mengirimkan *time stamp node* start pada *receiver sensor node*, dan dilanjutkan dengan proses akuisisi data oleh setiap modul sensor.

```
kirim(start);
bacaDHT();
bacaLDR();
bacaMoisture();
```

Gambar 5.11 Fungsi yang mengawali bekerjanya *sensor node*

```
void bacaLDR() {
  int cahaya;
  double Vo = analogRead(LDR_Pin) * 0.004882815;
  cahaya = (2500 / Vo - 500) / 10;
  if (ldr[1] != 0) {
    banding(cahaya, ldr);
  } else {
    ldr[1] = cahaya;
    Serial.print("Intensitas Cahaya sekarang ");
    Serial.print(ldr[1]);
    Serial.println(" lux");
  }
}
```

Gambar 5.12 Fungsi bacaLDR()

Pada Gambar 5.12 merupakan program pada fungsi bacaLDR() yang berfungsi untuk mengakuisisi data intensitas cahaya dengan menggunakan modul sensor *Light Dependent Resistor*, yang kemudian data yang didapatkan akan diolah oleh mikrokontroler sehingga dapat dipahami sebagai data intensitas cahaya dengan satuan “lx”.

```
void bacaDHT() {
  int suhu, chk;
  chk = DHT.read11(DHT11_PIN);
  if (chk == 0) {
    delay(500);
    suhu = DHT.temperature;
  } else {
    bacaDHT();
  }
  if (dht[1] != 0) {
    banding(suhu, dht);
  } else {
    dht[1] = suhu;
    Serial.print("Suhu sekarang ");
    Serial.print(dht[1]);
    Serial.println(" *C");
  }
}
```

Gambar 5.13 Fungsi bacaDHT()

Pada Gambar 5.13 merupakan program pada fungsi bacaDHT() yang berfungsi hanya untuk mengakuisisi data suhu dengan menggunakan modul sensor DHT11, proses akuisisi data dapat langsung dilakukan dengan penambahan *library* DHTlib sehingga data yang diterima oleh mikrokontroler sudah dapat dipahami sebagai data suhu dengan satuan "^{0}C".

```
void bacaMoisture() {
  int lembab, min = 300, max = 1000;
  int raw = analogRead(Moisture_Pin);
  lembab = constrain(raw, 485, 1023);
  lembab = map(lembab, 485, 1023, 100, 0);
  if (mois[1] != 0) {
    banding(lembab, mois);
  } else {
    mois[1] = lembab;
    Serial.print("Persentase Kelembaban Tanah sekarang ");
    Serial.print(mois[1]);
    Serial.println(" %");
  }
}
```

Gambar 5.14 Fungsi bacaMoisture()

Pada Gambar 5.14 merupakan program pada fungsi bacaMoisture() yang berfungsi untuk mengakuisisi data kelembaban tanah dengan menggunakan modul sensor *Soil Moisture*, yang kemudian data yang didapatkan akan diolah oleh mikrokontroler sehingga dapat dipahami sebagai data persentase kelembaban tanah dengan satuan "%".

Kemudian setelah menjalankan rentetan fungsi tersebut, program akan dilanjutkan dengan mengeksekusi *code* program pada fungsi loop() seperti yang ditunjukkan pada Gambar 5.15.

```
void loop(void) {
  Serial.println("---------------------------------");
  bacaDHT();
  bacaLDR();
  bacaMoisture();
  if (ai[0] == 0 && ai[1] == 0) { //proses pemeriksaan nilai marking variable
    if (ai[1] == 0 && ai[2] == 0) {
      waktu();
      delay(300);
      Serial.println("Zzzzz_Sleep_Mode_zzzzZ");
      delay(300);
      sleep();
    }
  }
  delay (1000);
}
```

Gambar 5.15 Fungsi loop() pada *transmitter sensor node*

Seperti yang ditunjukkan pada Gambar 5.12, 5.13, dan 5.14, ketiga fungsi pembacaan data sensor terdapat fungsi banding() yang memiliki dua parameter fungsi, yakni data hasil *sensing* saat ini dan data pada variable yang siap atau telah dikirimkan. Sementara didalamnya terdapat mekanisme analisa untuk membandingkan hasil *sensing* apakah masih sama dengan sebelumnya atau tidak Gambar 5.16 menunjukkan program pada fungsi banding().

```
void banding(int a, int *d) {
  if (a != d[1]) {
    Serial.println("Terjadi Perubahan!");
    d[1] = a; kirim(d);
    if (d[0] == 0) {
      Serial.print("Suhu sekarang ");
      Serial.print(d[1]);
      Serial.println(" *C");
      ai[0] = 1; //merubah nilai marking variable suhu ketika terjadi perubahan data sensing
    } else if (d[0] == 1) {
      Serial.print("Intensitas Cahaya sekarang ");
      Serial.print(d[1]);
      Serial.println(" lux");
      ai[1] = 1; //merubah nilai marking variable intensitas cahaya ketika terjadi perubahan
                 //data sensing
    } else {
      Serial.print("Persentase Kelembaban Tanah sekarang ");
      Serial.print(d[1]);
      Serial.println(" %");
      ai[2] = 1; //merubah nilai marking variable persentase kelembaban tanah ketika terjadi
                 //perubahan data sensing
    }
  } else { penilaian(d);
  }
}
```

Gambar 5.16 Fungsi banding() pada *transmitter sensor node*

Pada Gambar 5.16 yang merupakan program pada fungsi banding(), terdapat dua fungsi yang dieksekusi di dalamnya, yakni fungsi kirim() dengan parameter alamat variabel yang siap dikirimkan, dan juga fungsi penilaian() dengan parameter variabel yang akan dieksekusi. Fungsi kirim() hanya akan dieksekusi ketika proses perbandingan menghasilkan perbedaan hasil *sensing* saat ini dengan

sebelumnya, sebaliknya jika ternyata hasilnya adalah sama maka akan mengeksekusi fungsi penilaian(). Program pada fungsi kirim() berada pada Gambar 5.17 dan 5.19 untuk fungsi penilaian().

```
void kirim(int d[]) {
  msg[0]=d[0]; //proses encoding data
  msg[1]=d[1]; //memasukkan hasil sensing suhu kedalam paket pengiriman
  DateTime now = rtc.now();
  msg[2]=now.year(); //pemberian time stamp pada paket pengiriman
  msg[3]=now.month();
  msg[4]=now.day();
  msg[5]=now.hour();
  msg[6]=now.minute();
  msg[7]=now.second();
  Mirf.send((byte *)msg); //paket dikirimkan
  while ( Mirf.isSending() ) {
    waktu(); //mencetak time stamp
  }
  Mirf.powerDown();
}
```

Gambar 5.17 Program fungsi kirim() pada *transmitter sensor node*

Pada Gambar 5.17 dapat dilihat bahwa data yang akan dikirimkan adalah data pada variabel *array* dengan tipe data *integer*, yang kemudian data hasil *sensing* tersebut akan dimasukkan kedalam variabel yang dilengkapi dengan *encoding* data serta *time stamp* pengirimannya. Pada saat modul NRF24L01 sedang melakukan pengiriman data, program akan terus menerus menjalankan fungsi waktu() yang akan mengembalikan data waktu dan penanggalan sebagai *time stamp* sampai proses pengiriman data selesai dilakukan. Setelah modul NRF24L01 selesai mengirimkan data, modul akan dimatikan dengan instruksi "Mirf.powerDown()" untuk menghemat konsumsi arus modul pada *sensor node*.

```
void waktu() {
  DateTime now = rtc.now();
  Serial.print(now.year(), DEC);
  Serial.print('/');
  Serial.print(now.month(), DEC);
  Serial.print('/');
  Serial.print(now.day(), DEC);
  Serial.print(" (");
  Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);
  Serial.print(") ");
  Serial.print(now.hour());
  Serial.print(':');
  Serial.print(now.minute());
  Serial.print(':');
  Serial.print(now.second());
  Serial.println();
}
```

Gambar 5.18 Fungsi waktu()

Gambar 5.18 merupakan fungsi waktu() yang digunakan untuk mencetak data waktu dan penanggalan yang tersimpan pada modul RTC sebagai *time stamp* pada sisi *transmitter sensor node.*

Pada Gambar 5.19 yang merupakan program fungsi penilaian() akan dilakukan perubahan nilai dari *marking variable* yang menandakan data hasil *sensing* yang terletak pada variabel yang menjadi parameter bahwa hasil *sensing* tidak mengalami perubahan, sehingga akan berpengaruh pada hasil keputusan mekanisme *low power* yang terletak pada fungsi loop().

```
void penilaian (int *d) {
  if (d[0] == 0) {
    ai[0] = 0; //mengembalikan nilai marking variable suhu bernilai 0 ketika data sensing
               //tidak mengalami perubahan
  } else if (d[0] == 1) {
    ai[1] = 0; //mengembalikan nilai marking variable intensitas cahaya bernilai 0 ketika
               //data sensing tidak mengalami perubahan
  } else if (d[0] == 2) {
    ai[2] = 0; //merubah nilai marking variable persentase kelembaban tanah bernilai 0
               //ketika data sensing tidak mengalami perubahan
  } else {
    ;
  }
}
```

Gambar 5.19 Proggram fungsi penilaian() pada *transmitter sensor node*

Setelah mikrokontroler mengeksekusi ketiga fungsi akuisisi data sensor beserta rangkaian program didalamnya, eksekusi program akan dilanjutkan pada pembacaan *marking variable* untuk pengambilan keputusan apakah *sensor node* harus memasuki *sleep mode* ataukah tidak. *Marking variable* merupakan variabel *array integer* yang berfungsi sebagai penanda ketika hasil pembacaan sensor terjadi perubahan atau tidak. Apabila variable tersebut secara keseluruhan berisikan nilai angka 1, maka *sensor node* akan memasuki *sleep mode*, jika tidak maka akan mengulang program pada fungsi loop().

Sleep mode akan dimulai dengan mencetak *time stamp* yang ditandai dengan pengeksekusian fungsi waktu(), setelah itu barulah menjalankan fungsi sleep() yang berisikan kode program untuk menempatkan mikrokontroler pada mode *power down*. Program didalam fungsi sleep() ditunjukkan pada Gambar 5.20.

```
void sleep() {
  for (int i = 1; i <= 15; i++) { //proses mengulang eksekusi fungsi selama 15 kali
    LowPower.powerDown(SLEEP_4S, ADC_OFF, BOD_OFF);
    //mikrokontroler berada pada mode power down selama 4 detik dengan menonaktifkan Analog
    //Digital Converter dan Brown-out Detection
  }
}
```

Gambar 5.20 Proggram fungsi sleep() pada *transmitter sensor node*

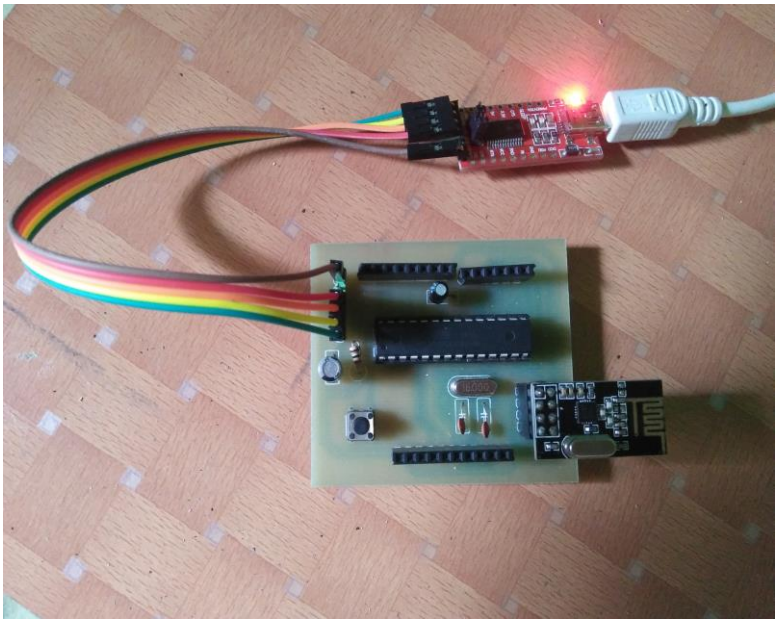
Gambar 5.20 menunjukkan program berfungsi untuk menempatkan mikrokontroler pada mode *power down* selama 4 detik, namun kondisi tersebut akan di ulang selama 15 kali, sehingga mikrokontroler akan berada pada mode *power down* dengan ADC (*Analog to Digital Converter*) dan BOD (*Brown-out Detection*) yang dinonaktifkan selama satu menit. Setelah itu mikrokontroler akan malanjutkan program pada fungsi loop() dengan *delay* selama satu detik yang bertujuan agar data hasil *sensing* dapat terbaca dengan baik melalui *serial monitor*.

Sensor node akan terus menerus menjalankan program yang berada pada fungsi loop() pada Gambar 5.16 hingga *transmitter sensor node* dimatikan atau kehabisan sumber daya.

5.2.3 Implementasi sistem pada *receiver sensor node*

Implementasi sistem pada *receiver sensor node* dilakukan sesuai dengan pembahasan perancangan sistem. Dalam perangkaian *sensor node* yang disesuaikan dengan perancangan sistem yang mana perangkat keras yang digunakan hanya mikrokontroler ATmega328P, modul komunikasi nirkabel NRF24L01, dan modul *USB-to-TTL* FTDI *Break-out.*

Pada *transmitter sensor node* modul FTDI hanya digunakan sebagai penyalur sumber daya bagi *sensor node*, sedangkan pada *receiver sensor node* selain digunakan sebagai penyalur sumber daya, modul FTDI juga digunakan sebagai media komunikasi serial antara *personal computer* dengan *sensor node*, sehingga aktifitas penerimaan data dapat dipantau melalui *serial monitor*. Perangkat keras pada *receiver sensor node* lebih sedikit jika dibandingkan dengan *transmitter sensor node*, hal tersebut disebabkan fungsi utama *sensor node* hanya untuk menerima dan menampilkan data. Implementasi perangkat keras pada *receiver sensor node* ditunjukan pada Gambar 5.21.



Gambar 5.21 Implementasi perangkat keras *receiver sensor node*

Sedangkan dalam implementasi sistem pada bagian perangkat lunak berbeda dengan *transmitter sensor node*, diantara perbedaan tersebut yakni tidak adanya mekanisme *low power* dan *library* yang digunakan lebih sedikit karena modul yang digunakan hanya modul NRF24L01 dan fungsi utama dari sensor node hanya menerima dan menampilkan data. *Library* yang digunakan yakni Arduino Mirf *library* dan SPI.h seperti pada Gambar 5.22.

node_recv_mirf01 §

```
#include <nRF24L01.h>
#include <Mirf.h>
#include <MirfHardwareSpiDriver.h>
#include <SPI.h>

int msg[2];
```

Gambar 5.22 Library yang digunakan pada *receiver sensor node*

Pada Gambar 5.23 menunjukkan tentang inisialisasi variabel dan konfigurasi pengalamatan *sensor node* untuk keperluan penerimaan data dari *transmitter sensor node* yang digunakan pada *receiver sensor node* sebagai penunjang berjalannya sistem.

```
int msg[8]; // inisialisasi variable penampung data

void setup(void){
 Serial.begin(57600);
  Mirf.spi = &MirfHardwareSpi;
  Mirf.init();
  Mirf.setRADDR((byte *)"FGHIJ"); //inisialisasi alamat penerimaan
  Mirf.payload = sizeof(msg); //inisialisasi payload penerimaan data
  Mirf.channel = 90; //inisialisasi chanel komunikasi
  Mirf.config();
  Serial.println("Listening...");

 }
```

Gambar 5.23 Implementasi pada *receiver sensor node*

5.2.4 Urutan berjalannya *receiver sensor node*

Bagian utama program *receiver sensor node* berada pada fungsi loop(), yang mana fungsi loop() akan mulai dijalankan ketika tahapan inisialisasi dan *setup* telah selesai dilakukan. Berikut ini merupakan program pada fungsi loop() yang ditunjukan oleh Gambar 5.24.

```
void loop(void){
 if (Mirf.dataReady()){
   bool done = false;
   while (!done){
         Mirf.getData((byte *)msg);
     done=true;
     cetak();
   }
 }
}
```

Gambar 5.24 Fungsi loop() pada *receiver sensor node*

Pada Gambar 5.24 dapat dilihat bahwa cara kerja dari program dimulai dengan pengecekan terlebih dahulu apakah modul NRF24L01 mendeteksi penerimaan data ataukah tidak, apabila didapati adanya penerimaan data maka *sensor node* akan melakukan pembacaan data yang diterima dan dimasukkan pada *variable array integer* dengan identitas "msg" untuk kemudian ditampilkan. Dalam menampilan data yang diterima, program akan menjalankan fungsi cetak() yang mana program pada fungsi tersebut terdapat di Gambar 5.25.

```
void cetak(){
    if(msg[0]==0){ //proses decoding data
      tstamp(); //pemanggilan fungsi untuk mencetak time stamp
      Serial.print("Suhu sekarang ");
      Serial.print(msg[1]);
      Serial.println(" *C");
    }
    if(msg[0]==1){ //proses decoding data
      tstamp(); //pemanggilan fungsi untuk mencetak time stamp
      Serial.print("Intensitas Cahaya sekarang ");
      Serial.print(msg[1]);
      Serial.println(" lx");
    }
    if(msg[0]==2){ //proses decoding data
      tstamp(); //pemanggilan fungsi untuk mencetak time stamp
      Serial.print("Persentase Kelembaban Tanah sekarang ");
      Serial.print(msg[1]);
      Serial.println(" %");
    }
    if(msg[0]==8){ //proses decoding data
      Serial.println("=== Node Start ===");
      tstamp(); //pemanggilan fungsi untuk mencetak time stamp
    }
}
```

Gambar 5.25 Fungsi cetak() pada *receiver sensor node*

Alur program pada fungsi cetak() yakni mula-mula setelah *sensor node* mengambil data yang telah diterima, dilakukan proses *decoding* yang berfungsi untuk mengartikan data apakah yang sedang diterima saat ini. Proses *decoding* tersebut yakni dengan melakukan pembacaan blok *array* pertama dari data yang diterima, setelah melewati proses *decoding* barulah data tersebut dapat ditampilkan pada *serial monitor* sesuai dengan identitas datanya.

Pada bagian *decoding* data akan dibedakan menjadi 4 jenis data, yakni dengan pengkodean 0 yang merupakan data suhu, pengkodean 1 sebagai data intensitas cahaya, pengkodean 2 sebagai data persentase kelembaban tanah, dan pengkodean 8 sebagai data *time stamp node start,* selain mencetak data hasil *sensing* yang diterima *receiver sensor node* juga akan mencetak *time stamp* dengan menjalankan fungsi tstamp() seperti pada Gambar 5.26.

```
void tstamp(){
  Serial.print(msg[2]); //mencetak data tahun
  Serial.print('/');
  Serial.print(msg[3]); //mencetak data bulan
  Serial.print('/');
  Serial.print(msg[4]); //mencetak data tanggal
  Serial.print('\t');
  Serial.print(msg[5]); //mencetak data jam
  Serial.print(':');
  Serial.print(msg[6]); //mencetak data menit
  Serial.print(':');
  Serial.print(msg[7]); //mencetak data detik
  Serial.println();
}
```

Gambar 5.26 Fungsi tstamp() pada *receiver sensor node*

Pada Gambar 5.26 merupakan *code* program pada fungsi tstamp(), yang mana fungsi tersebut digunakan untuk mencetak data *time stamp* yang dikirmkan oleh *transmitter sensor node* untuk memberikan tanda terjadinya sebuah *event* baik ketika *transmitter sensor node* mulai dijalankan ataupun mengirimkan data ketika mendeteksi perubahan hasil *sensing*.

Sensor node akan terus menerus menjalankan program yang berada pada fungsi loop() pada Gambar 5.24 hingga *receiver sensor node* dimatikan.