BAB 5 PERANCANGAN DAN IMPLEMENTASI

Bab ini menguraikan proses implementasi dari dasar teori yang telah dipelajari sesuai analisis dan perancangan sistem.

5.1 Perancangan

Perancangan penelitian ini dilakukan untuk merencanakan tahapan yang harus dilakukan dalam melakukan penelitian. Tahapan penelitian yang harus dilakukan antara lain identifikasi kebutuhan yang akan digunakan dalam penelitian, baik pada sisi perangkat keras maupun perangkat lunak, serta perancangan proses implementasi. Diharapkan dengan dilakukannya tahapan perancangan penelitian dapat mempermudah pelaksanaan penelitian.

5.1.1 Pendefinisian dan Perancangan Penelitian

Penelitian dilakukan dengan menerapkan protokol TDMA dan metode penyetaraan waktu TPSN yang diimplementasikan pada Arduino Nano. Protokol TDMA digunakan pada sistem monitoring kualitas air kolam sebagai metode *anti-collision* untuk menghindari tabrakan pengiriman data secara *wireless*. Protokol ini berfungsi untuk menetapkan slot waktu pengiriman pada tiap *node client.* Metode TPSN digunakan untuk memberikan waktu yang setara pada tiap node.

Implementasi protokol TDMA dan metode TPSN pada sistem pengiriman akuisisi data air kolam ikan berbasis *wireless* ini dilakukan dengan merangkai seluruh perangkat keras yang dibutuhkan agar dapat bekerja sekaligus dapat berkomunikasi sesuai dengan kebutuhan penelitian. Perangkat keras yang digunakan meliputi Arduino Nano, modul komunikasi *wireless* nRF24L01 serta sensor pH, sensor suhu air dan sensor kekeruhan air. Protokol TDMA diterapkan pada mikrokontroler *Node base* agar dapat memberikan slot waktu penjadwalan pengiriman terhadap node *client*. Proses sebelumnya yaitu dilakukannya penyetaraan waktu antar *node*, baik *node client* maupun node *base* dengan menggunakan TPSN. Kemudian dilakukan pengambilan data oleh sensor pH, suhu air dan kekeruhan air pada masing-masing *node client*. Hasil monitoring nilai pH, suhu dan kekeruhan air tersebut akan dikirimkan oleh masing-masing node client ke node base dengan menggunakan modul komunikasi *wireless* nRF24L01. Hasil tersebut akan ditampilkan pada *serial monitor* node *base* sebagai ilustrasi tampilan pemilihan slot *node client*, ilustrasi data nilai pH, suhu dan kekeruhan air tiap *node client* serta tampilan ilustrasi *time-synchronization* seluruh *node*.

5.1.2 Perancangan Perangkat keras

Perancangan perangkat keras berdasarkan pada analisis kebutuhan penelitian yang ditetapkan di awal penelitian. Terdapat 6 komponen perangkat keras yang bekerja, yaitu : Arduino Nano, sensor pH, sensor suhu air, sensor kekeruhan air dan modul *wireless* nRF24L01. Untuk dapat memantau kualitas air kolam

diperlukan sensor pH, sensor suhu air dan sensor kekeruhan air sebagai alat deteksi. Hasil deteksi berupa data nilai pH, suhu air dan kekeruhan air yang kemudian akan diolah Arduino pada masing-masing node *client*. Pada *node base* diimplementasikan protokol TDMA untuk menentukan slot waktu pengiriman data agar tidak terjadi *collision* atau tabrakan data.

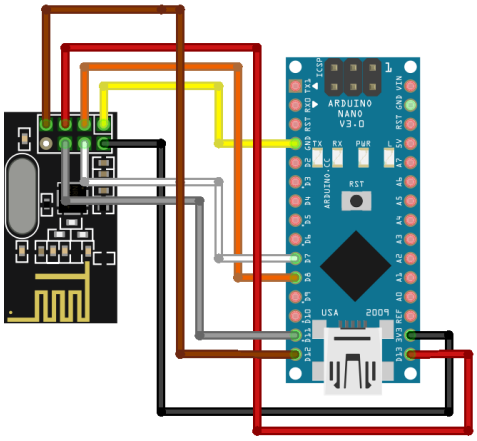
Perancangan pada sisi perangkat keras, dibagi menjadi 2 bagian yaitu perancangan pada sisi node *client* dan perancangan pada sisi *node base*.

5.1.2.1 Perancangan Subsistem *Node Base*

Gambar 5.1 Diagram Sistem *Node Base*

Penjelasan diagram sistem pada *node base* :

1. *Power Supply* adalah daya yang dibutuhkan node untuk menjalankan fungsinya. Pada penelitian ini, *power supply* berupa tegangan sebesar 5V yang didapatkan dari *port* USB komputer/laptop yang digunakan.
2. Mikrokontroler pada *node base* menggunakan Arduino Nano yang bertindak sebagai pengolah data.
3. nRF24L01 merupakan jenis *modul wireless* yang digunakan pada penelitian ini yang bertindak sebagai media komunikasi data.



Gambar 5.2 Perancangan Sistem *Node Base*

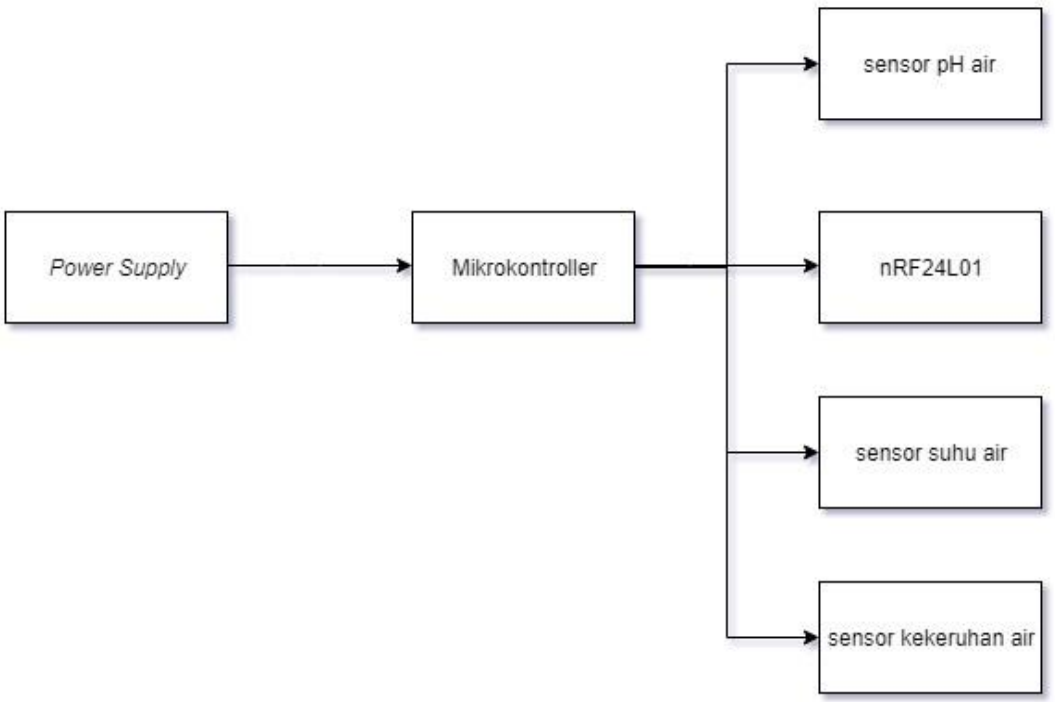
Pada penelitian ini, perancangan sistem node *base* yaitu dengan menghubungkan mikrokontroler Arduino Nano dengan modul komunikasi *wireless* nRF24L01. Modul komunikasi *wireless* pada *node base* digunakan untuk melakukan komunikasi dengan *node client*. Perancangan sistem dilakukan dengan menghubungkan pin-pin yang ada pada nRF24L01 dengan pin yang ada

pada *board* Arduino Nano. Bagaimana modul *wireless* nRF24L01 dan Arduino Nano terhubung dapat dilihat pada **Tabel 5.1.**

Tabel 5.1 Konfigurasi Pin Arduino Nano dan nRF24L01 *Node Base*

nRF24L01	Arduino Nano
Nama Pin	Nama Pin
Pin 1 GND	GnD
Pin 2 Vcc	3.3V
Pin 3 CE	D8
Pin 4 CS	D7
Pin 5 SCK	D13
Pin 6 MOSI	D11
Pin 7 MISO	D12

5.1.2.2 Perancangan Sistem *Node Client*

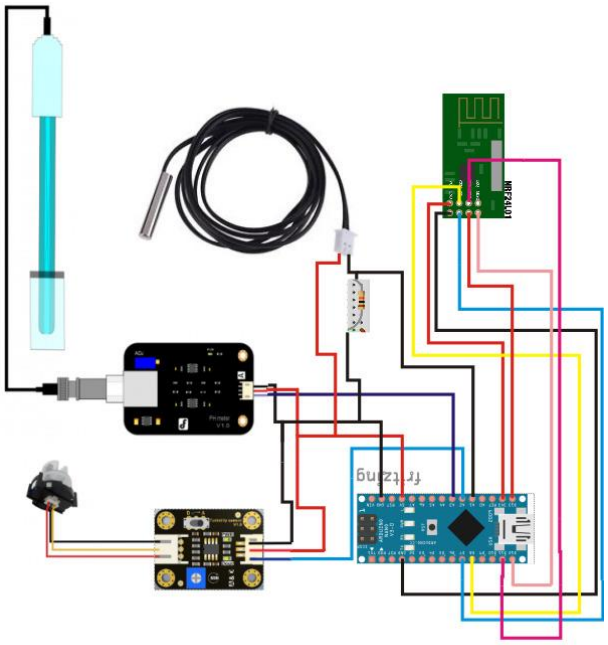


Gambar 5.3 Diagram Sistem *Node Client*

Berikut penjelasan diagram sistem pada *node Client* :

1. *Power Supply* adalah daya yang dibutuhkan node untuk menjalankan fungsinya. Pada penelitian ini, *power supply* berupa tegangan sebesar 5V yang didapatkan dari *port* USB komputer/laptop yang digunakan.

2. Mikrokontroler pada *node client* menggunakan Arduino Nano yang bertindak sebagai pengolah data hasil *sensing* sensor ph, sensor suhu air dan kekeruhan air serta sebagai kontrol komunikasi dengan *receiver node base*.
3. Pada penelitian ini, sensor pH diperlukan sebagai modul *input* yang bertindak sebagai komponen pemantau objek dengan hasil *sensing* berupa data nilai pH air kolam.
4. Pada penelitian ini, sensor suhu air diperlukan sebagai modul *input* yang bertindak sebagai komponen pemantau objek dengan hasil *sensing* berupa data nilai suhu air kolam.
5. Pada penelitian ini, sensor kekeruhan air digunakan sebagai modul *input* yang bertindak sebagai komponen pemantau objek dengan hasil *sensing* berupa data nilai kekeruhan air kolam.



Gambar 5.4 Perancangan Sistem *Node Client*

Pada penelitian ini, perancangan sistem *node client* terdiri dari 2 bagian. Bagian yang pertama adalah perancangan mikrokontroler Arduino Nano dengan sensor pH air, sensor suhu air dan sensor kekeruhan sebagai komponen *input*. Perancangan sistem dilakukan dengan cara menghubungkan pin-pin yang ada pada sensor pH, sensor suhu air dan sensor kekeruhan air dengan pin yang ada pada *board* Arduino Nano. Bagaimana sensor pH air, sensor suhu air, sensor kekeruhan air dan Arduino Nano terhubung dapat dilihat pada **Tabel 5.2** (Sensor pH Air); **Tabel 5.3** (Sensor Suhu Air); dan **Tabel 5.4** (Sensor kekeruhan Air).

Tabel 5.2 Konfigurasi Pin Sensor pH Air dan Arduino Nano *Node Client*

Sensor pH air	Arduino Nano
Nama Pin	Nama Pin
VCC	5V
A	A3
GND	GnD

Tabel 5.3 Konfiguriasi Pin Sensor Suhu Air dan Arduino Nano *Node Client*

Sensor suhu air	Arduino Nano
Nama Pin	Nama Pin
VcC	5V
DQ	A1
GND	GND

Tabel 5.4 Konfigurasi Pin Sensor Kekeruhan Air dan Arduino Nano *Node Client*

Sensor Kekeruhan air	Arduino Nano
Nama Pin	Nama Pin
VCC	5V
A	A2
GND	GND

Keempat komponen tersebut akan saling terhubung melalui kabel *jumper.* Pin *power* atau Vcc pada sensor pH air, sensor suhu air dan sensor kekeruhan air terhubung dengan pin *power output* 5V pada Arduino Nano. Hal ini dimaksudkan agar semua sensor dapat aktif dan bekerja dengan mendapatkan *input* daya dari Arduino.

Bagian yang kedua yaitu perancangan mikrokontroler Arduino Nano dengan modul *wireless* nRF24L01. Perancangan sistem dilakukan dengan menghubungkan pin-pin yang ada pada nRF24L01 dengan pin yang ada pada *board* Arduino Nano. Bagaimana modul *wireless* nRF24L01 dan Arduino Nano terhubung bisa dilihat pada **Tabel 5.5**

Tabel 5.5 Konfigurasi Pin Arduino Nano dan nRF24L01 *Node Client*

nRF24L01	Arduino Nano
Nama Pin	Nama Pin
Pin 1 GND	GND
Pin 2 Vcc	3.3V
Pin 3 CE	D8
Pin 4 CS	D7
Pin 5 SCK	D13
Pin 6 MOSI	D11
Pin 7 MISO	D12

5.1.3 Perancangan Komunikasi Perangkat keras

Komunikasi antara node *client* dan *node base* pada penelitian ini menggunakan komunikasi *wireless* dengan menggunakan modul *wireless* nRF24L01. Pada penelitian ini menggunakan komunikasi data serial, dimana data akan dikirimkan per-*bit* data. Pengiriman data secara *wireless* pada prinsipnya menumpangkan data pada pita frekuensi pembawa. Frekuensi yang digunakan untuk melewatkan data adalah pada 2,4GHz global ISM Band.

Frekuensi dan *channel* yang digunakan dalam penelitian ini diatur secara otomatis. Modul komunikasi *wireless* nRF24L01 menggunakan *range channel* antara 0-125. Penggunaan *channel* juga mempengaruhi frekuensi yang digunakan oleh tiap *node* dalam jaringan sensor. Penggunaan frekuensi pada modul komunikasi *wireless* nRF24L01 ini dimulai dari 2.400 GHz sampai dengan 2.525 GHz. Pada data *sheet* nRF24L01 tersedia rumus untuk menentukan frekuensi yang digunakan, yaitu

$$F0 = 2400 + RF\ Channel\ [MHz] \quad (5.1)$$

Pada penelitian ini menggunakan *channel* 101 atau 1MHz, maka frekuensi yang digunakan adalah :

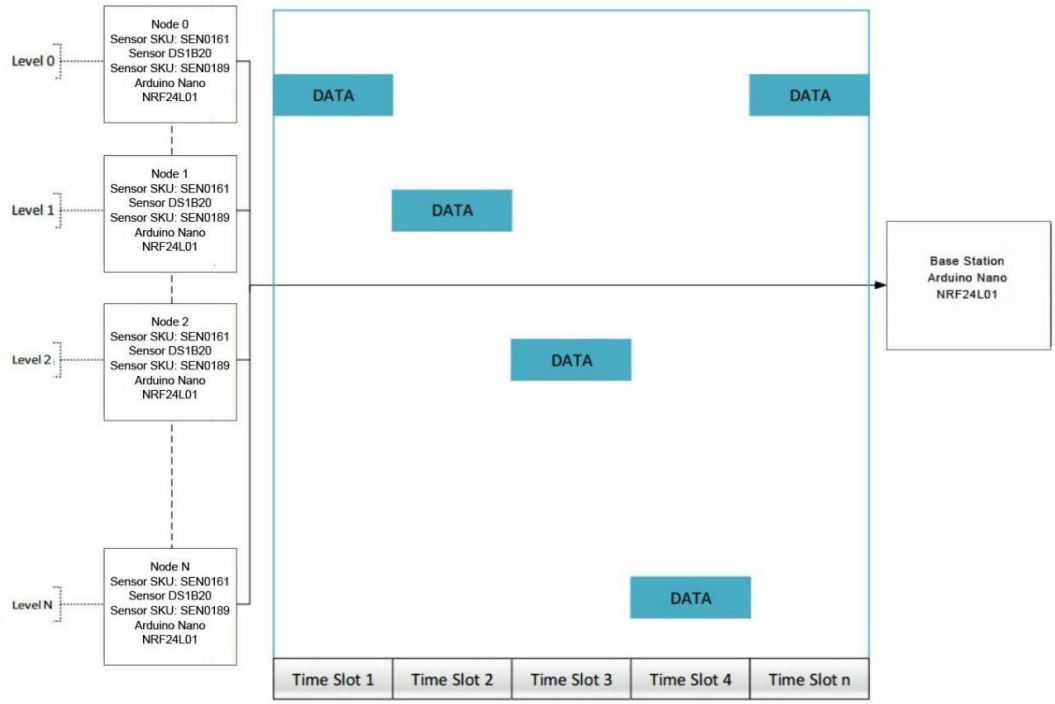
$$F0 = 2400 + 101 \quad (5.2)$$

$$F0 = 2101\ MHz \quad (5.3)$$

5.1.4 Perancangan Penyetaraan Waktu TPSN dan Protokol TDMA

Perancangan penyetaraan waktu pada node *client* dibagi menjadi 2 level, yaitu level 1 dan level 2, meskipun pada implementasinya nantinya dibatasi cukup 2 *node client* saja. Dimana level tersebut akan diberi secara urut berdasarkan urutan *client* yang aktif. Diagram perancangan sinkronisasi waktu TPSN dan protokol TDMA ditunjukkan pada Gambar 5.5. Sinkronisasi waktu dengan TPSN dibagi menjadi 2 fase atau tahap. Fase pertama adalah *Discovery*

Phase dimana pada fase ini akan membentuk topologi hirarki *tree* dan level-level pada *node client.* Node *base* berperan selayaknya *root* yang menentukan struktur topologi *tree* dan menentukan node yang mengisi masing-masing level topologi. Dilanjutkan fase kedua adalah *Synchronization phase* yang merupakan fase penyetaraan waktu antar *node base* dan node *client.*

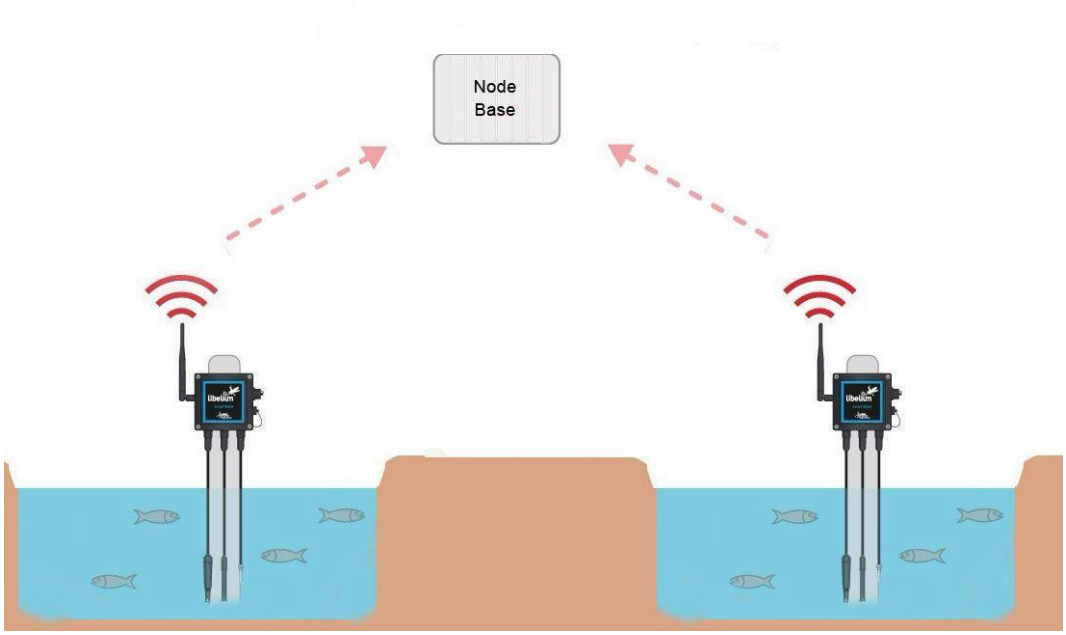


Gambar 5.5 Perancangan Metode TPSN dan Protokol TDMA

Perancangan penjadwalan sebagai metode *anti-collision* pengiriman data dengan protokol TDMA adalah membuat slot-slot waktu seperti **Gambar 5.5.** Pengiriman data sensor dilakukan ketika tiap node *client* telah menerima slot waktu pengiriman. Slot waktu yang didapatkan oleh *node client* secara *random* berdasarkan ketentuan konfigurasi. Pada penelitian ini slot waktu disediakan sebanyak 5 buah slot waktu dengan tiap slot diberikan waktu 2 detik untuk mengirim data.

5.1.5 Perancangan Peletakan Sensor

Pada penelitian ini digunakan sensor pH, sensor suhu dan sensor kekeruhan air. Air yang dipantau yang dimaksud adalah air kolam ikan yang digunakan sebagai nilai *input* pada sistem. Agar dapat memberikan nilai *input* yang akurat, diperlukan adanya perancangan peletakan sensor sebagai komponen pengukur nilai pH, suhu dan kekeruhan air. Ilustrasi peletakkan sensor pH air, sensor suhu air dan sensor kekeruhan air dapat dilihat pada **Gambar 5.6.**

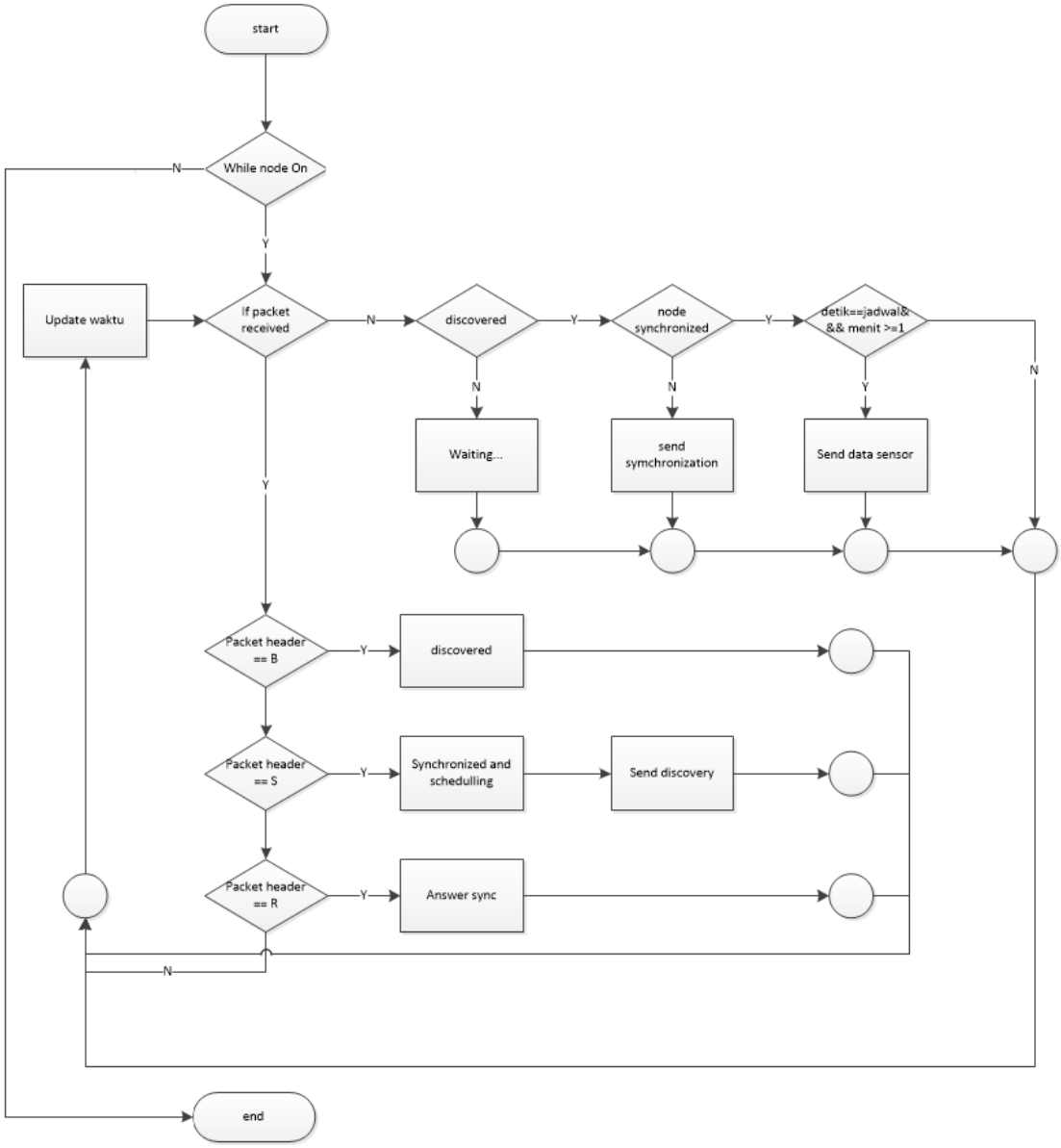


Gambar 5.6 Ilustrasi Peletakkan Sensor pH, suhu, dan kekeruhan air

Semua sensor akan diletakkan di dalam air kolam. Hal ini dimaksudkan ketika ada perubahan kualitas air kolam, sensor dapat secara langsung mendeteksi nilai pH, suhu air dan kekeruhan air kolam ikan.

5.1.6 Perancangan Pengiriman Data *Node* Client

Node *client* berperan sebagai pengirim data atau *transmitter* yang mengakuisisi data nilai pH, suhu dan kekeruhan air objek yang dipantau. Alur kerja pengiriman data *client* dapat dilihat pada **Gambar 5.7.**



Gambar 5.7 Diagram Alur Pengiriman Data *Client*

Proses pengiriman data pada sisi *client* diawali dengan melakukan proses inisialisasi. Proses inisialisasi ini meliputi inisialisasi *baudrate,* pin sensor, dan *channel*. Inisialisasi *baudrate* digunakan untuk menentukan mikrokontroler yang dipilih bekerja pada *rate* yang telah ditentukan. Inisialisasi pin sensor berguna untuk menentukan sensor bekerja pada pin yang telah ditetapkan. Inisialisasi *channel* berfungsi untuk menentukan *channel* komunikasi *wireless* nRF24L01 antara *node base* dan *node client.*

Setelah node *client* aktif maka node akan melakukan proses penyetaraan waktu. Proses penyetaraan waktu ini memanfaatkan metode TPSN, *node client* akan menerima paket *discovery* dari *root* (Node *base)*. Ketika salah satu node *client* telah berhasil terhubung dengan node *root* (Node *base)* dan memiliki waktu yang setara, lalu *node* tersebut akan mengirimkan paket *discovery* ke *node* lainnya berdasarkan dengan konsep hierarki pada metode sinkronisasi waktu TPSN.

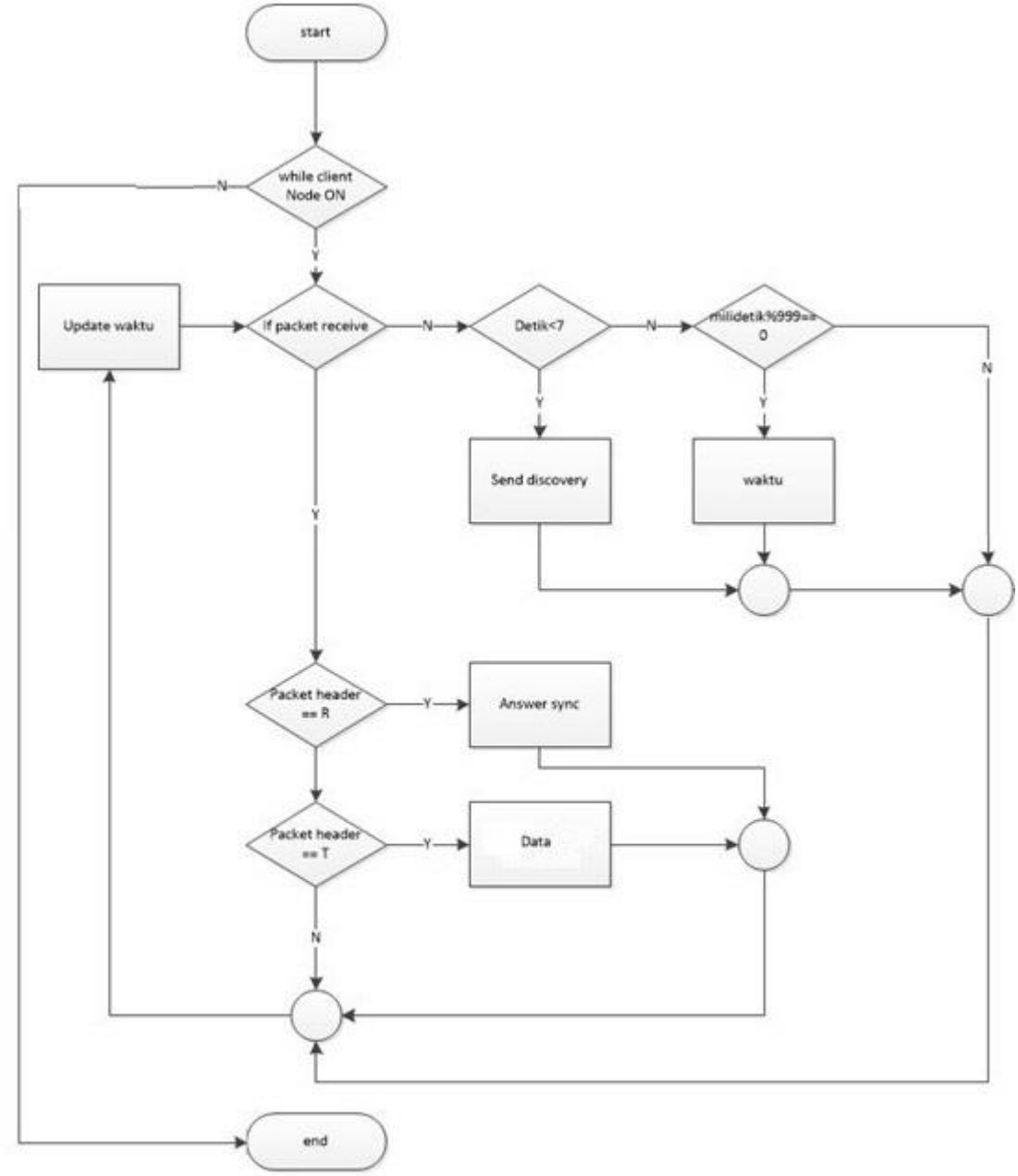
Proses penyetaraan waktu pada TPSN terdiri dari dua langkah, pertama adalah tahap *Discovery Phase* yang berfungsi untuk menentukan level pada masing-masing *node*. Yang kedua adalah tahap *Synchronization Phase* dimana *node root* (*Node base*) akan melakukan proses penyetaraan waktu dengan *node client.* Penyetaraan waktu diawali dengan melakukan pengiriman permintaan penyetaraan waktu oleh *node client* melalui paket header. *Node client* dan *Root* (*Node base*) akan saling mengirimkan waktu yang dimiliki dan kemudian melakukan pembaharuan atau pencocokan. Dalam proses penyetaraan waktu antar *node*, terdapat pembentukan jadwal pengiriman yang berupa slot waktu. Slot waktu ditujukan agar pengiriman data yang dilakukan secara bersamaan tidak saling bertabrakan atau *collision.* Metode penjadwalan pengiriman yang digunakan adalah TDMA. Setelah semua *node client* berhasil menyetarakan waktu dan mendapatkan slot pengiriman data, maka akan dilakukan proses pengiriman data nilai pH, suhu dan kekeruhan air menuju *node base.*

Ketika proses akuisisi data sensor, nilai mentah hasil akuisisi data sensor kekeruhan dikonversikan ke dalam satuan NTU menggunakan formula persamaan 5.1 Dimana x adalah nilai voltage dalam satuan volt dan y adalah nilai kekeruhan dalam satuan NTU.

$$y = -1120{,}4 \times x^2 + 5742{,}3 \times x - 4352{,}9 \quad (5.1)$$

5.1.7 Perancangan Penerimaan Data *Node Base*

Node *base* adalah sebuah *node* yang berperan sebagai *receiver* data yang dikirimkan oleh node *client.* Alur kerja node *base* bisa dilihat di **Gambar 5.8**



Gambar 5.8 Diagram Alur Penerimaan Data *Node Base*

Setelah *root* (Node *Base)* aktif maka *root* akan mengirim paket *discovery* selama 7 detik. *Root* menerima paket permintaan penyetaraan waktu bila ada *node client* yang menerima paket *discovery* tersebut. *Root* akan mengirimkan paket sinkronisasi waktu menuju node yang mengirimkan permintaan. Selanjutnya *root* akan mengirimkan penjadwalan waktu pengiriman data sensor dengan metode TDMA. Terakhir, *root* (Node *Base)* akan menerima nilai data

sensor pH, suhu dan kekeruhan air secara bersama-sama dari 2 node *client* berdasarkan slot waktunya masing-masing.

5.2 Implementasi

Pada tahap ini akan dikaji tentang implementasi dari semua perancangan penelitian yang telah disusun sebelumnya. Pada tahap ini dijelaskan langkah-langkah penelitian dalam mengimplementasikan protokol TDMA sebagai metode *anti-collision* pada sistem monitoring kualitas air kolam ikan berbasis *wireless* serta akan dijelaskan mengenai batasan penelitian yang dilakukan.

5.2.1 Spesifikasi Perangkat keras

Perancangan penelitian "Implementasi Pengiriman Data *Wireless* dengan Metode *Time Division Multiple Access* Dan *Timing-Sync Protocol for Sensor Networks* pada Kolam Ikan" ini menerapkan *anti-collision* data menggunakan beberapa komponen perangkat keras, diantaranya sensor pH, sensor suhu air, sensor kekeruhan air, modul *wireless* nRF24L01 dan Arduino Nano. Spesifikasi dari komponen-komponen tersebut adalah :

a. Sensor pH Air

Sensor pH yang digunakan dalam penelitian ini adalah sensor pHSKU:SEN0161. Spesifikasi dari sensor pH dapat dilihat pada Tabel 5.6.

Tabel 5.6 Spesifikasi Sensor pH SKU:SEN0161

Sensor pHSKU:SEN0161	
Module Power	DC 5V
Module Size	43mmx32mm
Measuring range	0-14 Ph
Accuracy	± 0.1pH (25 ℃)
Response Time	≤ 1min

b. Sensor Suhu Air

Sensor suhu air yang digunakan dalam penelitian ini adalah sensor suhu DS18B20. Spesifikasi dari sensor suhu air dapat dilihat pada Tabel 5.7.

Tabel 5.7 Spesifikasi Sensor Suhu DS18B20

Sensor Suhu DS18B20	
Module Power	DC 3-5V
Module dimension	1x1x1cm
Measuring range	-55°C - +125°C
Accuracy	-10°C to +85°C: ±0.5°C

c. Sensor Kekeruhan Air

Sensor kekeruhan air yang digunakan dalam penelitian ini adalah sensor kekeruhan air SKU:SEN0189. Spesifikasi dari sensor kekeruhan air dapat dilihat pada **Tabel 5.8**.

Tabel 5.8 Spesifikasi Sensor Kekeruhan Air SKU:SEN0189

Sensor Kekeruhan Air SKU:SEN0189	
Power module	DC 5V
Module Dimension	38mmx28mm10mm
Measuring range	0-3000 NTU
Working current	40 Ma

d. Perangkat Mikrokontroler

Mikrokontroler yang dipakai dalam penelitian ini adalah Arduino Nano. Spesifikasi dari Arduino Nano dapat dilihat pada **Tabel 5.9.**

Tabel 5.9 Spesifikasi Arduino Nano

Arduino Nano	
Microcontroller	Atmel ATMega168 or ATMega328
Operating Voltage (logic level)	5 V
Input Voltage (Recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14 (6 provide PWM output)
Analog Input Pins	8
DC Current per I/O Pins	40 Ma
Flash Memory	16 KB (ATMega168) or 32 KB (ATMega 328) of which 2 KB used by bootloader
SRAM	1 KB (ATMega168) or 2 KB (ATMega328)
EEPROM	512 bytes (ATMega168) or 1 KB (ATMega328)
Clock Speed	16 MHz

e. Modul *wireless* nRF24L01

Modul *wireless* nRF24L01 berfungsi untuk mengirimkan data (komunikasi data) antar *node client* dan node *base*. Spesifikasi modul *wireless* nRF24L01 bisa dilihat di Tabel 5.10.

Tabel 5.10 Spesifikasi Modul *Wireless* nRF24L01

Modul NRF24L01	
Frequency	2,4GHz ISM Band
Channels	126 RF
Data Rate	250Kbps, 1 and 2Mbps
Transmitter	11,3mA at 0dBm output power
Digital Interface	SPI Speed 0-8Mhz
Voltage	1,9 ~ 3,6V

5.2.2 Batasan Penelitian

Batasan-batasan dalam penelitian "Implementasi Pengiriman Data *Wireless* dengan Metode *Time Division Multiple Access* Dan *Timing-Sync Protocol For Sensor Networks* Pada Kolam Ikan" adalah sebagai berikut :

1. Implementasi dilakukan kolam ikan yang berukuran mini.
2. Peletakan *node client* berada di dekat kolam ikan dengan semua sensor dimasukkan ke dalam air kolam ikan.
3. *Input* dari implementasi sistem adalah hasil pantau oleh sensor pH air, suhu air dan kekeruhan air pada *node client*.
4. Metode TPSN diimplementasikan pada tiap node baik *node client* dan *node base* sebagai metode penyetaraan waktu.
5. Protokol TDMA diimplementasikan pada node *base* dan *node client* untuk memberikan slot waktu pengiriman pada masing-masing node *client* yang sudah memiliki waktu yang setara.
6. Hasil pantau sensor pH air, suhu air dan kekeruhan air dikirimkan secara *wireless* oleh *node client* menuju node *base* menggunakan *transmitter* nRF24L01 yang digunakan.
7. Hasil pantau sensor pH air, suhu air dan kekeruhan air diterima oleh *node base* menggunakan *receiver* nRF24L01 yang digunakan.
8. *Output* dari hasil deteksi sensor yang diterima oleh node *base* akan ditampilkan pada *serial monitor* node *base. Output* yang ditampilkan berupa alamat tiap node *client*, waktu pengiriman dan hasil *sensing* sensor berupa nilai pH, suhu dan kekeruhan akuisisi data air kolam ikan.

5.2.3 Implementasi Komunikasi Perangkat keras

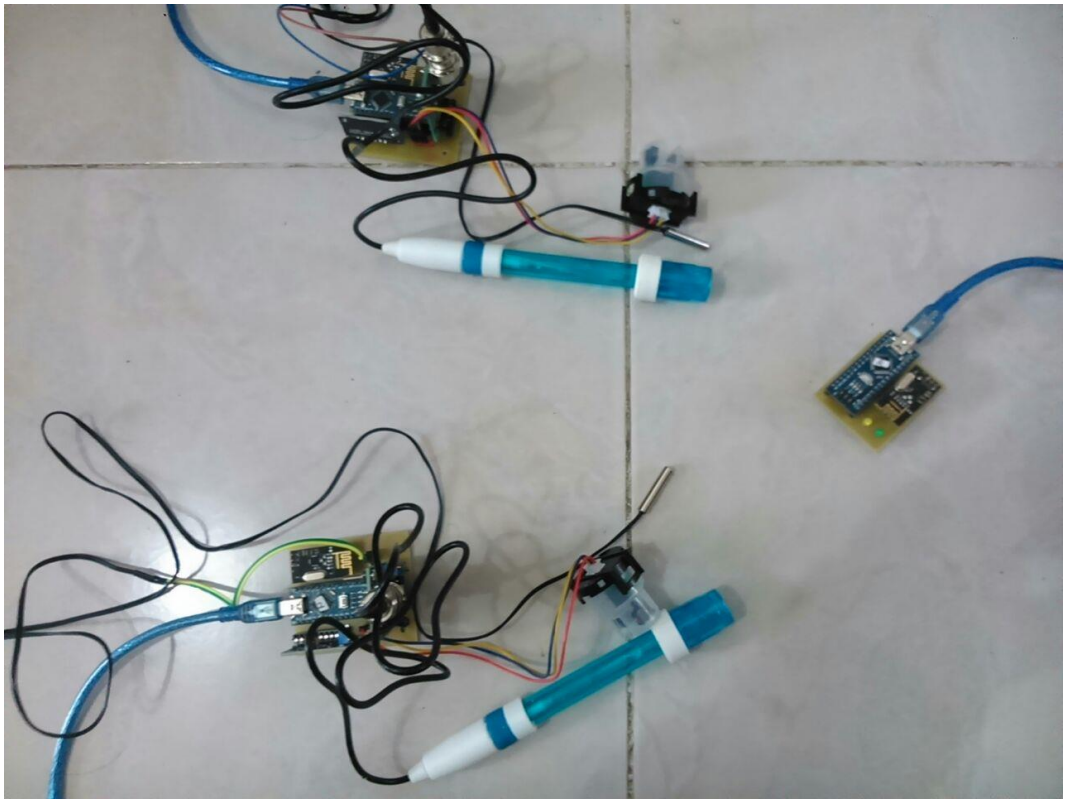
Pada tiap node, implementasi komunikasi perangkat keras dilakukan secara serial dengan menghubungkan *board* Arduino ke laptop untuk pemrograman dan pengujian. Pada *node base* dan node *client*, untuk proses komunikasi atau pengiriman data akan dilakukan secara *wireless* dengan menggunakan modul *wireless* nRF24L01.

Gambar 5.9 Implementasi nRF24L01 sebagai Komunikasi *Wireless*

Gambar 5.9 adalah implementasi komponen sistem berupa nRF24L01 dan Arduino Nano untuk melakukan komunikasi data *wireless* antar *node*.

5.2.4 Implementasi Perangkat keras

Pada penelitian ini, implementasi perangkat keras adalah pemasangan seluruh komponen-komponen yang digunakan. Terdapat 2 pengelompokan perangkat keras, yaitu; perangkat keras pada *Node base* dan perangkat keras pada *node client*. Pemasangan perangkat keras pada *node base* meliputi Arduino Nano untuk menampilkan data hasil deteksi sensor dan *receiver* modul komunikasi *wireless* nRF24L01 yang berperan menerima data. Pada *node client*, pemasangan perangkat keras meliputi sensor pH air, sensor suhu air, sensor kekeruhan air, Arduino Nano sebagai pengolah data dan protokol TDMA, dan *transmitter* modul komunikasi *wireless* nRF24L01 sebagai perangkat pengirim data. Implementasi perangkat keras secara keseluruhan dapat dilihat pada **Gambar 5.10.**



Gambar 5.10 Komponen Perangkat Keras

Implementasi perangkat keras pada *node base* dilakukan dengan menghubungkan nRF24L01 terhadap Arduino Nano. Implementasi perangkat keras pada tiap *node client* dilakukan dengan menghubungkan semua sensor dan nRF24L01 dengan tiap mikrokontroler, yaitu Arduino Nano.

5.2.5 Implementasi Perangkat lunak

Pada penelitian ini diperlukan *library* pada sisi pemrograman perangkat lunak supaya sistem bisa berjalan sesuai fungsinya. Fungsi *library* sendiri untuk membantu kinerja atau fungsionalitas dari komponen-komponen yang sudah dirancang. *Library* dimasukan ke dalam program yang tertanam di dalam perangkat pengolah data, yaitu Arduino Nano. Implementasi *library code* pada sisi *Node client* dapat dilihat pada **Tabel 5.11.** Dan implementasi *library code* pada sisi *Node base* dapat dilihat pada **Tabel 5.12.**

Tabel 5.11 *Library Code Node Client*

Library Code node client	
1	`#include <Ports.h>`
2	`#include <LowPower.h>`
3	`#include <SPI.h>`
4	`#include <Mirf.h>`
5	`#include <nRF24L01.h>`
6	`#include <MirfHardwareSpiDriver.h>`
7	`#include <stdio.h>`
8	`#include <string.h>`
9	`#include <stdlib.h>`
10	`#include "HardwareSerial.h"`

Library<SPI.h> berfungsi untuk pengaktifan pin SPI pada modul komunikasi *wireless* nRF24L01. *Library*<Mirf.h> berfungsi untuk pengiriman data pada modul komunikasi *wireless* nRF24L01. *Library*<Stdio.h> berfungsi untuk melakukan perintah *input-output* pemrograman pada Bahasa C atau C++. *Library*<string.h> berfungsi untuk memberikan nilai pada suatu karakter string, sedangkan <stdlib.h> berfungsi untuk melakukan proses perhitungan matematis dalam program.

Tabel 5.12 *Library Code Node Base*

Library Code node base	
1	`#include <SPI.h>`
2	`#include <Mirf.h>`
3	`#include <nRF24L01.h>`
4	`#include <MirfHardwareSpiDriver.h>`
5	`#include <stdio.h>`
6	`#include <string.h>`

Library<SPI.h> berfungsi untuk mengaktifkan pin SPI pada modul komunikasi *wireless* nRF24L01. *Library*<Mirf.h> berfungsi untuk melakukan pengiriman data pada modul komunikasi *wireless* nRF24L01. *Library*<Stdio.h> berfungsi untuk melakukan perintah *input-output* pemrograman pada Bahasa C atau C++. *Library*<string.h> berfungsi untuk memberikan nilai pada suatu karakter string,

sedangkan <stdlib.h> berfungsi untuk melakukan proses perhitungan matematis dalam program.

1. Implementasi *Main program*

Pada *main program* ada beberapa fungsi yang akan dijalankan secara berurutan. Implementasi *main program* pada sisi *node client* dapat dilihat pada **Tabel 5.13**, sedangkan implementasi *main program* pada sisi *node base* bisa dilihat di **Tabel 5.14.**

Tabel 5.13 *Main program* Sistem *Node Client*

Main program sistem *node client*

```
1  void loop(){
2
3    if(Mirf.dataReady()){
4
5             Mirf.getData((byte *)&DataTerima);
6
7
8              LevelCheck[0] = DataTerima[8];
9              LevelClear = strtoul(LevelCheck,NULL,0);
10             if(!terdaftar && DataTerima[0] == 'B' && DataTerima[1] !=
11 'B'){
12               Serial.print("Data RAW: ");
13               Serial.println(DataTerima);
14               Serial.print("Ukuran: ");
15               Serial.print(sizeof(DataTerima));
16               Serial.print(" byte ");
17               Serial.print(sizeof(DataTerima)*8);
18               Serial.println(" bit");
19
20               Serial.print("Header: ");
21               Serial.println(DataTerima[0]);
22               discovered();
23               Serial.print("waktu tunggu ");
24               Serial.print(tunggu);
25               Serial.println(" uS");
26               Serial.print("idle...");
27               terdaftar = true;
28             }else if(DataTerima[0] == 'S'){
29               waktu();
30               TS4 = milidetik;
```

```
              detiklama = detik;
              buffering();
              synchronized();
              waktu();
            }else if(DataTerima[0] == 'R' && LevelCheck != LevelKirim && tersinkron){
               waktu();
               sprintf(TimeStamp2,"%d",milidetik);
               TimeStampInt = milidetik;
               answer_sync();
               waktu();

            }else if(DataTerima[0] == 'T' && LevelClear == (Level+1)){
               TDMArecv();
            }
      }else{
        if (!tersinkron && terdaftar && millis()%tunggu==1){
            synchronize();
              tunggu = 20000;
            delay(1);

        }

        if(tersinkron){
            if(milidetik%999==0){
                cetakwaktu();
                delay(3);
                x++;
                if(x < 7){
                  discovery();
                }
            }
            if(menit >= 1)
            {
              detikx=detik%10;
            }
            if(detikx==jadwal && menit >= 1){
                TDMAsend();
               delay(1000);
                if((jadwal-jadwalpatent)>3){jadwal = jadwalpatent;}
            }
```

```
78          }
71
72        }
73     waktu();
74  }
```

Kemudian akan ada kondisi dimana *node client* akan melakukan pengecekan paket yang diterima. Jika paket yang diterima adalah paket dengan header “B” maka akan berlanjut pada proses *discovered.* Jika paket yang diterima adalah paket dengan header “S” maka akan berlanjut pada proses *synchronized.* Jika paket yang diterima adalah paket dengan header “R” maka akan berlanjut pada proses *answer-sync.*

Tabel 5.14 *Main program* Sistem *Node Base*

Main program Sistem *Node Base*

```
1  void loop(){
2    if(Mirf.dataReady()){
3      Mirf.getData((byte *)&DataTerima);
4      LevelCheck[0] = DataTerima[8];
5      //konversi str ke ul, string ke unsigned long, untuk perbandingan
6  di if nya
7      LevelClear = strtoul(LevelCheck,NULL,0);
8      //dicek dulu ini mau terima data tahap TPSN ataukah tahap TDMA
9  ngirim2 data
10     if(DataTerima[0] == 'R' && LevelClear<2){
11       waktu();
12       sprintf(TimeStamp2,"%d",milidetik);
13       TimeStampInt = milidetik;
14       //ini TPSN
15       synchronize();
16     }else if(DataTerima[0] == 'T' ){
17       //terima data semua hasil akuisisi air kolam ikan melalui
18 method TDMA
19       TDMArecv();
20     }
21   }else{
22     //root broadcast dulu kalau Mirf.dataready tdk terima data
23 apapun, broadcast diberi kesempatan "scara subjektif 7 sekon"
24     if(millis()<7000){
25     discovery();
26     delay(1000);
27     }else if(milidetik%999==0){
28       cetakwaktu();
     delay(3);
```

```
29      }
30     }
31    //jalankan method konversi waktu jam, menit, detik nya.
32    waktu();
    }
```

Ketika *node base* telah aktif dan siap, *node base* akan menjalankan program secara berurutan. Pada *node base* akan dilakukan pengecekan paket sesuai dengan headernya. Jika data yang diterima adalah data dengan header “R”, maka akan menjalankan proses *answer-sync* sebagai fungsi untuk membalas paket permintaan penyetaraan waktu. Jika paket yang diterima adalah paket dengan header “T”, maka akan menjalan proses penerimaan data sensor dari *node client.* Jika *node base* tidak menerima paket dengan header “R” ataupun “T”, makan *node base* akan melakukan proses *broadcast* paket *discovery.*

2. Implementasi Metode Penyetaraan Waktu TPSN

Pada penelitian ini diimplementasikan metode sinkronisasi waktu TPSN. Sinkronisasi waku dilakukan untuk membantu pemberian jadwal pengiriman data pada masing-masing *node client* yang saling terkoneksi. Sinkronisasi waktu TPSN dibagi menjadi 2 tahap, tahap pertama adalah *Discovery Phase* dan tahap kedua adalah *Synchronization Phase.* Pada tahap *Discovery Phase*, *root node* yang pada penelitian ini adalah node *base* memiliki tugas mem-*broadcast* paket *discovery* yang berisi identitas dan level pengiriman node *root* ke node terdekat atau *node client*. Tahap *Discovery Phase* akan membentuk hierarki yang dimulai dari *node base* sebagai *node root* yang menjadi Level 0 dan berlanjut ke *node* Level 1 dan Level 2*.* Pada tahap ini seluruh *node* akan mengenali posisi dan levelnya serta mengenali *parent*nya. Tahap kedua adalah *Synchronization Phase*, pada tahap ini semua *node* akan mulai menyetarakan waktu. Diawali dari node dengan level paling kecil, node Level 1 akan mengirimkan paket permintaan sinkronisasi waktu atau *request synchronization* ketika T1 ke *node root* (*Node base).* Selanjutnya *root* mendapatkan paket permintaan tersebut pada waktu T2 dan membalas paket tersebut dengan *Acknowledgement Packet* pada waktu T3. T3 akan berisikan nomor atau alamat hierarki T1, T2, dan T3. Selanjutnya, *node* Level 1 akan melakukan perhitungan dari aliran skenario yang telah terbentuk dengan melakukan pencocokan waktu. Skenario serupa akan dilakukan oleh *node* pada Level 2 ke *node* Level 1 sebagai *parent,* dan berulang hingga node pada Level n.

Berdasarkan penjelasan sebelumnya, bahwa proses sinkronisasi waktu pada TPSN dibagi menjadi dua tahap, diantaranya adalah tahap *Discovery Phase* dan *Synchronization Phase.* Tahap *Discovery Phase* dibagi menjadi 2 *method*, yaitu *method discovery*() dan *method discovered*(). *Method discovery*() berfungsi melakukan pengiriman atau *broadcast* paket *discovery* dari *root* ke semua *node client*. Sedangkan *Method discovered*() berfungsi untuk menerima paket *discovery* dari *root*. Implementasi kode program yang digunakan dapat dilihat di **Tabel 5.15.**

Tabel 5.15 *Method Discovery*() dan *Discovered*() pada *Node Client*

Method Discovery() dan Discovered() pada *node client*

```
1  void discovery(){
2          sprintf(LevelKirim,"%d", Levelx);
3          Mirf.setTADDR((byte *)"node");
4          strcpy(DataKirim,"B");
5          strcat(DataKirim,AlamatSendiri);
6          strcat(DataKirim,LevelKirim);
7          Mirf.send((byte *)&DataKirim);
8          Serial.println("Paket discovery terkirim!");
9          while(Mirf.isSending()){
10          }
11 }
12
13 void discovered(){
14   for (int d=0; d<4; d++){
15   AlamatParent[d] = DataTerima[d+1];
16   }
17   Serial.println("Paket Broadcast diterima!");
18   Serial.print("Alamat Parent adalah: ");
19   Serial.println(AlamatParent);
20   Leveltemp[0] = DataTerima[5];
21   Level = strtoul(Leveltemp, NULL, 0);
22   Serial.print("Node bergabung di hierarki level: ");
23   Serial.println(Level);
24   Level = Level+1;
25   Levelx = Level;
26
27 }
```

Setelah *client* mendapatkan paket *discovered,* selanjutnya *client* merubah alamatnya. Perubahan alamat *client* dimaksudkan supaya *root* bisa membedakan masing-masing *client* dari alamat baru yang dimiliki. Implementasi kode program agar mengubah alamat *client* ada pada **Tabel 5.16.**

Tabel 5.16 *Method* Ubah Alamat

Method Ubah alamat	
1	`void UbahAlamat(){`
2	`  int Acak = random(analogRead(A0),9999); //generate random number`
3	`  while(Acak<1000){`
4	`  Acak = random(analogRead(A0),9999);`
5	`  }`
6	`  sprintf(AlamatSendiri,"%d", Acak);`
7	`  }`

Tahap kedua adalah *Synchronization-time* yang terbagi menjadi 2 yaitu fungsi *synchronize*() dan fungsi *synchronized* (). *Synchronize*() berfungsi untuk melakukan permintaan penyetaraan waktu terhadap *node base.* Sedangkan *Synchronized* berfungsi untuk menerima permintaan sinkronisasi waktu dari *root.* Implementasi kode program bisa dilihat di **Tabel 5.17.**

Tabel 5.17 *Method Synchronize*() dan *Synchronized*()

Method Synchronize() dan Synchronized()

```
1  void synchronize(){
2    UbahAlamat();
3    waktu();
4    Mirf.setTADDR((byte *)Alamat Parent);
5    strcpy(DataKirim,"R");
6    strcat(DataKirim,AlamatSendiri);
7    sprintf(TimeStamp1,"%d",milidetik);
8    if(milidetik<10){
9    strcat(DataKirim,"0");
10   strcat(DataKirim,"0");
11   }else if(milidetik<100){
12   strcat(DataKirim,"0");
13   }
14   strcat(DataKirim,TimeStamp1);
15   sprintf(LevelKirim,"%d", Level);
16   strcat(DataKirim,LevelKirim);
17   Latency = millis();
18   Mirf.send((byte *)&DataKirim);
19   while(Mirf.isSending()){
20     Serial.println();
21     Serial.println("Request Sinkronisasi terkirim!");
22     Serial.print("Alamat Node: ");
23     Serial.println(AlamatSendiri);
24     delay(2);
25     }
26 }
27 void synchronized(){
28    Serial.print("Waktu awal: ");
29    waktu();
30    cetakwaktu();
31    waktu();
32    Latency = millis() - Latency;
33    ubahwaktu();
```

```
34      waktu();
35      cetakwaktu();
36      Serial.println("(Tersinkronisasi!)");
37      Serial.print("Dari paket: ");
38      Serial.println(DataTerima);
39      Serial.print("Ukuran: ");
40      Serial.print(sizeof(DataTerima));
41      Serial.print(" byte ");
42      Serial.print(sizeof(DataTerima)*8);
43      Serial.println(" bit");
44      Serial.print(", delay propagasi: ");
45      Serial.println(delayPropagasi);
46      Serial.print("Jadwal: ");
47      Serial.println(jadwalpatent);
48      Serial.print("Slot: ");
49      if(jadwalpatent%2==0)
50      {
51        Serial.println(jadwalpatent/2);
52      }
53      else
54      {
55        Serial.println((jadwalpatent/2)+1);
56      }
57      for(int i=0;i<Level;i++)
58      {
59        Serial.print("jadwal terpakai : ");
60        Serial.println(simpanjadwalint[i]);
61      }
62      if (delayPropagasi<50 && delayPropagasi>0){//20
63      tersinkron = 1;
64      }
65  }
```

Setelah mendapatkan paket *Synchronized*, *client* lalu merubah waktunya. Perubahan waktu ini supaya masing-masing *node* memiliki waktu yang setara. Implementasi kode program bisa dilihat di **Tabel 5.18.**

Tabel 5.18 *Method* UbahWaktu()

Method Ubah waktu	
1	`void ubahwaktu(){`
2	`  ubahmili = milibaru - milidetik;`
3	`  ubahdetik = detikbaru - detiklama;`

```
4    ubahmenit = menitbaru - menit;
5    ubahjam = jambaru - jam;
6    ubahmili = ubahmili +clockdrift+delayPropagasi;
7  }
```

3. Implementasi Protokol TDMA

Setelah waktu masing-masing *client* sudah tersinkronisasi, selanjutnya adalah dilakukan pengiriman data berdasarkan dengan jadwal yang diterima dengan menerapkan potokol TDMA. Pada penelitian ini waktu pengiriman yang tersedia adalah 10 detik dengan slot waktu sebanyak 5 slot. Dalam 5 slot waktu yang telah ditentukan, pada masing-masing slotnya ditentukan 2 detik waktu yang dapat digunakan untuk mengirimkan data. Slot waktu ini kan disebar ke *node client* secara *random*. Implementasi penyediaan slot waktu pengiriman data bisa dilihat pada **Tabel 5.19**.

Tabel 5.19 *Method* TDMAsched()

Method TDMAsched()

```
1  void TDMAsched(){
2    Sched = random(0,9);
3  }
```

Setelah waktu pengiriman ditentukan, tahap selanjutnya adalah membagi waktu pengiriman tersebut menjadi slot-slot waktu. Implementasi kode program pembagian slot waktu bisa dilihat pada **Tabel 5.20.**

Tabel 5.20 Kode Program Pembagian Slot Waktu

Method Pembagian slot waktu

```
1  if(jadwalpatent%2==0)
2     {
3       Serial.println(jadwalpatent/2);
4     }
5     else
6     {
7       Serial.println((jadwalpatent/2)+1);
8     }
9     for(int i=0;i<Level;i++)
10    {
11      Serial.print("jadwal terpakai : ");
12      Serial.println(simpanjadwalint[i]);
13    }
```

4. Implementasi Pengiriman Data

Setelah waktu masing-masing *node* telah setara, selanjutnya dilakukan implementasi pengiriman data dengan modul komunikasi *wireless* nRF24L01. Data dikirimkan menyesuaikan slot yang waktu pengiriman yang didapatkan supaya data yang dikirimkan tidak bertabrakan. Data yang dikirimkan adalah nilai pH, suhu dan kekeruhan air kolam yang didapatkan dari hasil *sensing* sensor pH, sensor suhu air, dan sensor kekeruhan air pada *node client.* Implementasi kode program *sensing* sensor dan pengiriman data bisa dilihat pada **Tabel 5.21** dan **Tabel 5.22.**

Tabel 5.21 KodeProgram *Sensing* data sensor pH, suhu air, dan kekeruhan air

Kode program sensing sensor	
1	`int dataSuhu = analogRead(A1);`
2	`  int dataPh = analogRead(A3);`
3	`  int dataTurbi = analogRead(A2);`
4	`  sprintf(x,"%4d%4d%4d", dataSuhu, dataPh,dataTurbi);`

Tabel 5.22 Kode Program Pengiriman Data

Kode program pengiriman data	
1	`strcat(DataKirim,x);`
2	`  Mirf.send((byte *)&DataKirim);`
3	`  Serial.print("Data yg dikirim: ");`
4	`  Serial.print(DataKirim);`
5	`  Serial.print(" ukuran: ");`
6	`  Serial.print(sizeof(DataKirim));`
7	`  Serial.print(" byte ");`
8	`  Serial.print(sizeof(DataTerima)*8);`
9	`  Serial.println(" bit");`
10	`  Serial.print("Suhu: ");`
11	`  Serial.print(dataSuhu);`
12	`  Serial.print(" valPh: ");`
13	`  Serial.print(dataPh);`
14	`  Serial.print(" valTurbi: ");`
15	`  Serial.println(dataTurbi);`
16	`  Serial.print("Dari alamat: ");`
17	`  Serial.println(AlamatSendiri);`
18	`  Serial.println("Data TDMA terkirim!");`