BAB III
METODE PENELITIAN

3.1 Metodelogi Penelitian

Metoda yang akan digunakan dalam penelitian ini adalah studi literatur dan eksperimen di laboratorium. Kegiatan penelitian ini dilakukan di Bengkel Observatorium Bosscha dan Laboratorium *Instrumentttasi* Prodi Fisika UPI. Perancangan, pembangunan serta untuk kerja dari sistem kendali dan telemetri untuk memantau memantau kinerja alat penghilang embun berbasis mikrokontroler ini terdiri dari beberapa tahapan proses, yaitu :

1. Perancangan perangkat keras
2. Perancangan server
3. Perancangan perangkat lunak menggunakan Arduino IDE
4. Perancangan *database* menggunakan XAMPP.
5. Perancangan pengiriman dan penerimaan data antara mikrokontroler dan *database* XAMPP menggunakan PHP, MySQL, *AJAX* dan Json.
6. Perancangan komunikasi antara *database* XAMPP dan halaman web.
7. Perancangan tampilan halaman web.
8. Pengujian semua komunikasi mikrokontroler, *database* dan *browser*.
9. Analisis hasil pengujian.

3.2 Waktu dan lokasi penelitian

Penelitian pembuatan sistem pemantauan dan kendali alat penghilang embun berbasis Arduino UNO R3 ATMEGA 328 dan *Ethernet Shield* dilaksanakan pada :

Waktu Pelaksanaan : Maret 2015 – September 2015.

Tempat pelaksanaan :
1. Bengkel Observatorium Bosscha.
2. Laboratorium Instrumentasi Prodi Fisika UPI.

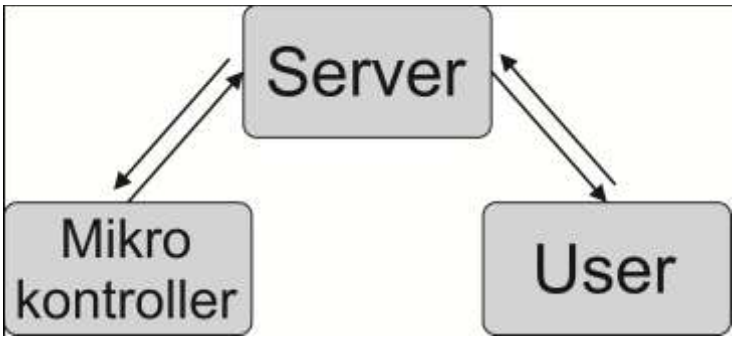
3.3 Diagram alir penelitian

Dalam penelitian ini terdapat tahapan-tahapan yang akan dilalui. Tahapannya adalah sebagai berikut:

Gambar 3. 1 Diagram alir penelitian

3.4 Perancangan sistem komunikasi

Sistem yang akan dibuat untuk komunikasi miktrokontroler ke web dijelaskan dalam gambar 3.2 berikut.



Gambar 3. 2 Sistem komunikasi mikrokontroler, server dan user.

Sistem yang dibuat akan bekerja sperti pada gambar 3.2. Data yang dihasilkan mikrkontroller akan dikirim ke server yang selanjutnya akan dikirim ke web(*user*). Sekalian dengan itu, *user* mengirim perintah berupa data yang dikirim ke server dan diteruskan ke mikrokontroler untuk di eksekusi oleh miktrokontroler. Dengan katalain, mikrokontroler ini dijadikan sebagai *client*

3.4.1 Perancangan Server

1. Setting akses pengguna server

   Server dibuat mengggunakan XAMPP. Dalam merancang server, dibutuhkan beberapa konfigurasi *network* seperti *IP Address, subnet mask* dan *domain name server* (DNS) untuk menginisialisasi server tersebut agar memiliki identitas.

   Agar server dapat diakses oleh pengguna lain, server harus disetting terlebih dahulu dengan membuka *httpd.conf* pada "...\xampp\apache\conf" dengan menggunakan *notepad*. Kemudian cari *Include "conf/extr1a/httpd-vhosts.conf*" dan *LoadModule vhost_alias_module modules/mod_vhost_alias.so*, jika bagian ini masih berupa komentar yang ditandai dengan "#" pada bagian

depannya, maka hilangkan tanda "#" pada bagian depannya agar fungsi *Load* dan *Include* dapat dijalankan sistem.

Setelah kedua fungsi tadi bekerja, tahap selanjutnya mengatur siapa saja yang dapat mengakses ke server dengan membuka *httpd-xampp.conf* pada "...\xampp\apache\conf\extra" dengan menggunakan *notepad*. Cari bagian "allow from *ip address",* kemudian ganti *Ip Address* dengan *IP* yang akan dapat mengakses server. Jika server ingin dibuka untuk umum, cari bagian "Deny from " menjadi "allow from all" seperti pada gambar dibawah ini

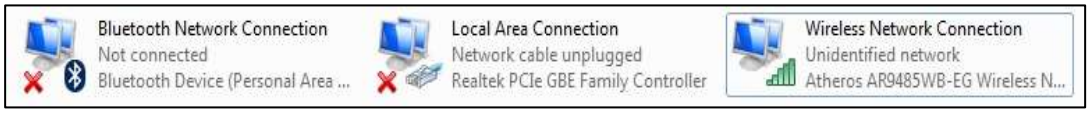
```
<LocationMatch "^/(?i:(?:xampp|security|licenses|phpmyadmin|webalizer|server-status|server-info))">
    Order deny,allow
    allow from all
    Allow from ::1 127.0.0.0/8 \
        fc00::/7 10.0.0.0/8 172.16.0.0/12 192.168.0.0/16 \
        fe80::/10 169.254.0.0/16
```

Gambar 3. 3 setting akses pengguna server

Gambar 3.9 merupakan settingan server yang dapat diakses oleh semua pengguna.

2. konfigurasi alamat server

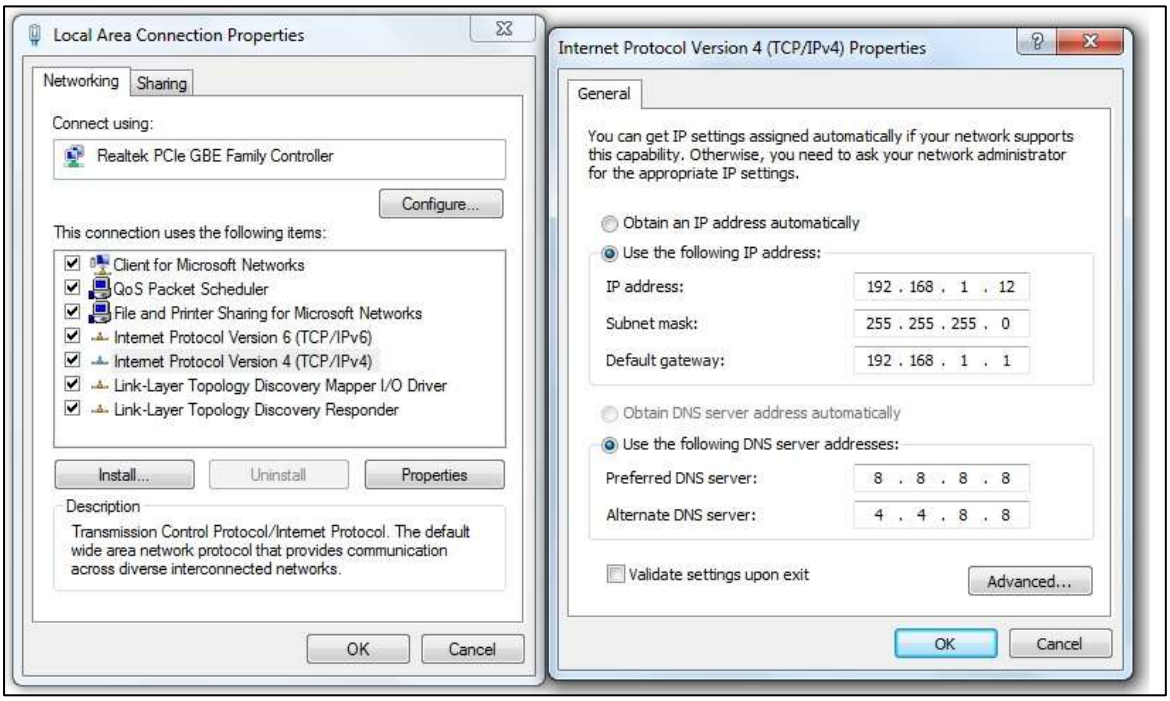
Server yang dibuat harus memiliki alamat agar data yang dikirimkan oleh mikrokontroler sesuai pada tujuannya. Konfigurasi *network* yang harus disetting adalah *IP Address, subnet mask* dan *domain name server* (DNS). Untuk mengisi *IP Address, subnet mask* dan DNS adalah dengan memasuki "...*control panel\network and internet\network and sharing center"* . pilih *change adapter setting*, maka akan muncul tampilan seperti berikut. :



Gambar 3. 4 Opsi Koneksi

Pada gambar 3.10 dapat dilihat beberapa opsi untuk mengatur sambungan kepada server. Untuk alat penghilang embun, pilihan yang digunakan adalah *Local Area Networ* (LAN) dengan menggunakan perantara *router* sebagai *getway*.

Untuk masuk settingan pilihan, klik pada *local area connection*, kemudian pilih *properties* maka akan muncul tampilan seperti berikut :



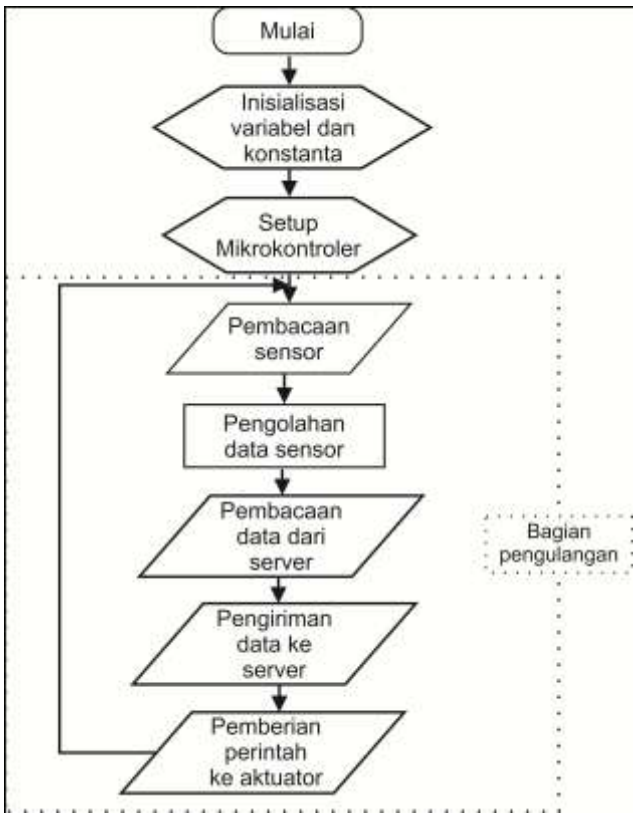
Gambar 3. 5 Local area conection properties

Setelah masuk pada *local area conection properties* pilih opsi *Internet Protocol version 4* seperti pada Gambar 3.11 sebelah kanan, maka akan muncul formulir untuk mengisi *IP Address*, *subnet mask*, *Default Getway* dan *domain name server* (DNS).

Isi IP address dengan alamat 192.168.1.12, *subnet mask* dengan kode 225.225.225.0, *Default Getway* dengan alamat *router* yang di setel 192.168.1.10 dan *domain name server* (DNS) dengan kode 8.8.8.8 dan *alternative* 4.4.8.8 setelah semua terisi klik "ok". Dengan demikian, server telah memiliki identitas seperti pada Gambar 3.11 bagian kanan.

1.4.2 Perancangan pemograman menggunakan Arduino IDE

Pemograman yang yang dibuat dalam arduino IDE, secara garis besar akan bekerja seperti pada diagram alir berikut pada gambar 3.12.



Gambar 3. 6 Diagram alir program *Arduino*

Pada gambar 3.12, dapat dilihat bahwa program akan bekerja dengan menginisialisasi variabel dan konstanta yang digunakan, selanjutnya menjalankan setup yang sesuai dengan instrumen yang digunakan. Setelah setup ini selesai, program memasuki bagian pengulangan. karena adanya pengulanangan (*looping*) dibagian akhir, pemograman arduino yang dibuat tidak akan memiliki titik ahir (bekerja terus-menerus). Selain dari bagian looping, program akan dieksekusi sekali saja.

Untuk mengisi program pada mikrokontroller ATMEGA 328 digunakan aplikasi Arduino IDE yang menggunakan bahasa C++. Penggunaan instrument sebagai input ataupun output dibutuhkan library sebagai driver dari instrumen yang diguakan. Penulisan program pada arduino terdapat tiga bagian. Antara lain *inisialisasi*, *setup* dan *looping*. Jika dilihat pada gambar 3.12, program dalam *looping* terdapat beberapa bagian yang berisi perintah yang berbeda.

Pada bagian inisialisasi ini digunakan untuk menginisilisasi library, *integer*, karakter dan variabel yang digunakan. Untuk alat penghilang embun sendiri terdapat inisialisasi seperti berikut :

```
#include <SPI.h>
#include <Ethernet.h>
#include <JsonParser.h>
#include <SHT1x.h>

using namespace ArduinoJson::Parser;
byte mac[]         = { 0xDE, 0xAD, 0xBE, 0xEF,
0xFE, 0xED }; // Mac Address Default
byte ip[]          = { 192,168,1,11 }; // Ip
untuk Ethernet shield
byte dnsserver[]   = { 8,8,8,8 }; // DNS Server
untuk Ethernet shield
byte gateway[]     = { 192,168,1,1}; // Gateway
untuk Ethernet shield
byte subnet[]      = { 255, 255, 255, 0 }; //
Subnet untuk Ethernet shield
EthernetClient client;
#define datain 5
#define dataout 6
#define clock 7
float a = 17.625;         // konstanta magnus
float b = 243.04;         // konstanta magnus
SHT1x shtin(datain, clock);
SHT1x shtout(dataout, clock);
boolean lastConnected = false;
char* statdriv = "0";
char* statspry = "0";
String strLineResult;
boolean startRead = false;
int sensor_hujan = 2;
int limswitch1 = 3  ;
int limswitch2 = 4;
int sprayer = 8;
int driver1 = 9;
int driver2 = 10;
int driver = A1;
int s;
int u;
String k;
String embi;
String dsci;
String hjn;
```

Dari teks boks di atas dapat dilihat bahwa *library* yang digunakan pada alat ini adalah *SPI , Ethernet, JsonParser* dan *SHT1x* . setelah inisial *library* selanjutnya menginisialisasi *Json Parser* untuk dieksekusi selanjutnya menginisialisasi *ethernet* atau menyeting alamat untuk mikrokontroler. Seperti pada textbox di atas. Alamat yang disetting dengan *IP Address* 192.168.1.11 agar memiliki *IP* setingkat dengan *ip server* dan *ip getway*. *subnet mask, Default Getway* dan *domain name server* (DNS) di setel sama dengan server. *Byte mac* di *setting default* sesuai dengan pabrikan.

Selanjutnya menginisialisasi SHT *#define datain 5 #define dataout 6 #define clock 7* SHT yang digunakan disini dua, SHT pertama data pinnya di setting ke pin 5 mikrokontroler dan yang kedua data pinnya di setting ke pin 6 dan kedua *clock* yang digunakan kedua SHT ini di satukan ke pin 7. SHT11 ini akan bekerja dengan memanggil data dan *clock* sehingga SHT pertama di deklarasi *SHTin(datain,clock)* dan *sht* kedua di *setting SHTout (dataout,clock).*

Float a dan *float b* digunakan untuk menginisialisasi variabel *magnus* yang akan digunakan untuk mencari *dewpoint* terjadinya kabut. *lastConnected* digunakan untuk mengdeklarasikan koneksi terakhir yang akan digunakan tidak berhenti dan terus melakukan pengulangan. *Char* statdriv* dan *char* statspry;* digunakan sebagai pengguna untuk pembanding karakter yang akan dibaca oleh mikrokontroler dari *database*. Selanjutnya mesetting *integer* dari sensor dan aktuator agar hasil dari sensor dan data yang dikirim ke aktuator menjadi bilangan bulat.

sensor hujan disetting menjadi *integer* adalah sensor hujan, *limit switch*, dan *driver*. *Integer s* dan *u* digunakan untuk variabel yang akan mengambil data dari *database* yang akan digunakan sebagai *set point* terjadinya kabut dan pengaman agar alat dapat bekerja sebelum berembun. Tahap selanjutnya menginisialkan *string*. *String* yang digunakan pada program ini bekerja untuk memberikan keterangan yang akan dikirim ke database sehingga alat dapat mengirim data berupa teks. *String k;* digunakan untuk memberika komentar keadaan kabut, *String embi;* digunakan untuk memberikan komentas terjadinya

embun pada bagian dalam *String dsci;* digunakan untuk memberikan kometar keadaan *Silika gel* dan *String hjn;* digunakan untuk komentar keadaan hujan atau tidak hujan.

Pada bagian setup dibuat program seperti berikut :

```
Serial.begin(9600);
Ethernet.begin(mac, ip, dnsserver, gateway, subnet);
pinMode(driver, OUTPUT);
digitalWrite(driver, LOW);
pinMode(sprayer, OUTPUT);
digitalWrite(sprayer, LOW);
pinMode(sensor_hujan, INPUT);
```

Pada bagian *Serial.begin(9600);* berfungsi untuk menyetelkan kecepatan transmisi data dengan kecepatan 9600 *bits* per detik. Perintah selanjutnya adalah *Ethernet.begin(mac, ip, dnsserver, gateway, subnet)* yang berfungsi untuk menjalankan modul *ethernet* dengan *mac, ip, dns server, gateway dan subnet* berasal dari hasil inisialisasi sebelumnya.

Pinmode digunakan untuk mendeklarasikan bahwa *pin* yang digunakan akan bekerja sebagai *input* atau *output*. Pada kode ini *driver* dan relay untuk *sprayer* di set sebagai *output* sedangkan sensor hujan sebagai *input*. Sedangakan *digitalWrite* dipasang *LOW* untuk *driver* dan *relay sprayer*. Hal ini dilakukan untuk pengaman agar saat miktrokontroler digunakan, aktuator dalam kondisi mati. Meski resikonya sedikit, namun jika aktuator langsung bekerja saat mikro kontroler mulai dijalankan, *shock* dari input luar bisa saja mempengaruhi dan menjadikan alat cepat rusak.

Setelah bagian *setup* selesai, selanjutnya membuat bagian *looping* dengan membuat perintah. Pada bagian *looping*, semua program digunakan untuk perintah yang akan di ulang terus menerus. Di bagian ini diawali dengan perintah mengambil data hasil dari pengukuran sensor hujan dan SHT11 yang telah di inisialisasi sebelumnya. data yang diambil dari sensor hujan adalah data digital yang berarti data itu hanya menampilkan 0 dan 1. Sedangkan untuk memanggil data hasil pengukuran SHT11 digunakan perintah *read*. Data yang akan di ambil

dari hasil pengukuran tersebut diambil dua angka di belakang koma sehingga digunakan fungsi *float*. Sehingga perintahnya akan ditulis sebagai berikut:

```
int HUJAN = digitalRead(sensor_hujan);
  float Ti = shtin.readTemperatureC();
  float Hi = shtin.readHumidity();
  float To = shtout.readTemperatureC();
  float Ho = shtout.readHumidity();
```

Pada kode ini dapat dilihat bahwa data yang dibaca dari sensor hujan adalah data digital yang perintahnya dituliskan dengan *digitalRead*. Sehingga jika data yang di ambil kurang dari 5 volt, data akan dijadikan nol (0) atau *LOW* dan jika data yang di ambil lebih atau samadengan 5 volt, data akan dijadikan satu (1) atau *HIGH*.

Pengambilan data hasil pengukuran *SHT11* digunakan *float* untuk mengambil data dua angka dibelakang koma. Variable *Ti* dan *Hi* digunakan untuk mengambil data dari *SHTin*, sedangkan *To dan H*o digunakan untuk membaca data dari *SHTout* *shtin.readTemperatureC()* untuk membaca data suhu dari hasil pengukuran *SHTin* dalam satuan ^{o}C dan *sht.readHumidity()* untuk membaca data kelembaban dari *SHTin* dalam satuan %.

Data yang dibaca dari senson *SHT11* ini biasanya belum standar, sehingga harus dikalibrasi ulang. Untuk mengkalibrasi, hal yang pertama adalah melihat hasil dari sensor yang digunakan dan sensor standar. Untuk melihat hasil pengukuran *SHT11* dalam *Arduino IDE*, dapat menuliskan *Serial.print(variabel yang diukur)* sehingga hasil baca dari *SHT11* dapat dilihat pada serial monitoring. Maka kodenya akan menjadi seperti berikut :

```
Serial.print("Ti= ");
Serial.print(Tin);
Serial.print(" Hi = ");
Serial.print(Hin);
Serial.print("T0= ");
Serial.print(Tout);
Serial.print(" H0 = ");
Serial.print(Hout);
```

Pada kode ini terdapat *Serial.print("nama=")* dimaksudkan untuk memberikan nama agar data yang ditampilkan dalam serial monitor menjadi lebih

rapih. Dari *serial monitoring* pada *arduino IDE* ini, dapat dilihat hasil pengukuran dari semua variabel temperatur dan kelembaban pada sensor dalam dan sensor luar. Dengan mengetahui hasil pengukuran sensor *SHT11*, selanjutnya dibandingkan dengan hasil pengukuran sensor standar dengan tempat yang sama.

Dari hasil perbandingan didapatkan perbedaan suhu antara sensor SHT11 dan sensor standar sekitar 1,5 ^{o}C dan perbedaan kelembaban sekitar 8%. Untuk mengkalibrasi sensor SHT11 ini hasil dari pembacaan sensor SHT11 pada suhu dikurangi 1,5 ^{o}C dan pada kelembaban dikurangi 8% pada programnya. Maka dalam pemogramannya akan menjadi seperti berikut

```
float Tin = Ti - 1.5;
float Tout = To - 1.5;
float Hin = Hi - 8;
float Hout = Ho - 8;
```

Setelah sensor terkalibrasi, langkah selanjutnya adalah menghitung suhu dewpoint. Dimana suhu *dewpoint* ini adalah suhu *set point* terjadinya embun. Untuk mendapatkan suhu *dewpoint* ini digunakan persamaan sebagai berikut:

$$Tdi = \frac{b\left(log\left(\left(\frac{Hi}{100}\right)+\left(\frac{(a.Ti)}{(b+Ti)}\right)\right)\right)}{a-\left(log\left(\left(\frac{Hi}{100}\right)-\left(\frac{(a.Ti)}{(b+Ti)}\right)\right)\right)} \qquad (2.11)$$

Dengan Tdi merupakan titik embun (dewpoint) dibagian dalam, atau titik refensi terjadinya embun. *a* dan *b* adalah konstranta magnus yang memiliki nilai *a = 17.625* dan *b = 243.04*. *Hi* adalah Kelembaban bagian dalam, dan *Ti* adalah suhu bagian dalam. Dengan memiliki persamaan ini maka program yang akan ditulis untuk mendapatkan titik embun adalah sebagai berikut:

```
  float Tdin = b*(log(Hin/100) + ((a*Tin)/(b+Tin)))/a-log(Hin/100) -
((a*Tin)/(b+Tin));

  float Tdout = b*(log(Hout/100) + ((a*Tout)/(b+Tout)))/a-
log(Hout/100) - ((a*Tout)/(b+Tout));
```

Titik embun ini akan digunakan untuk mendeklarasikan terjadinya embun. Embun didalam akan terjadi jika suhu luar lebih kecil dari titik embun *(Tou < Tdin)*. Sesuai yang diharapkan alat ini, pemberitahuan terjadinya embun akan

dilakukan sebelum terjadinya embun. Sehingga program akan diberi perintah untuk memberitahukan pengguna jika terjadi embun didalam ketika *(Tou - u < Tdin*) agar sebelum terjadinya embun, pengguna dapat mengganti *silika gel (silica gel)*. Maka program yang akan digunakan fungsi *if* seperti berikut:

```
if (Tout - s < Tdin){
    dsci= "Ganti%20Silika gel"  ; //ganti dsc
     //embun dalam
  }
  else if (Tout - s >= Tdin){
      dsci= "Silika gel%20Bagus"  ; //ganti dsc
      // tidak berembun didalam
    }
```

Perintah yang dilakukan untuk mendefinisikan terjadinya embun didalam dilakukan dengan permberitahuan keadaan *silika gel*. Maka perintah yang digunakan saat terjadinya embun diberi peringatan untuk mengganti silika gel dengan kode *dsci = "Ganti%20Dessican" dan dsci= "Silika gel%20Bagus"* untuk keadaan normal. Maksud dari %20 disana adalah sebagai tanda ganti spasi untuk pada bahasa SQL. Hasil pemberitahuan ini berupa karakter, seperti yang telah di inisialisasikan sebelumnya untuk *dsci*.

Selain pemberitahuan untuk pengembunan didalam, pemberitahuan keadaan juga dibuat untuk mengetahui keadaan hujan dan kabut. Untuk memberitahukan keadaan hujan dan kabut ini digunakan fungsi yang sama dengan pemberitahuan pada keadaan pengembunan didalam dengan menggunakan fungsi if dan pemberitahuan dengan karakter. Kode pemograman untuk kedua pemberitahuan ini adalah sebagai berikut

```
if (HUJAN == LOW){
hjn = "Hujan";
}
  else if (HUJAN == HIGH){
  hjn = "Tidak%20Hujan";
  }
if (Hout > u){
 k= "Berkabut" ; //berkabut
 }
  else if (Hout <= u){
   k= "Tidak%20Berkabut"; //tidak berkabut
   }
```

Pada perintah ini, keadaan hujan akan diberitakan "hujan" dan "tidak hujan" dan untuk keadaan kabut akan diberitakan "berkabut" dan "tidak Berkabut". Seperti halnya pemberitahuan keadaan pada pengembunan, penlisan untuk kode ini juga digunakan %20 untuk spasinya.

Pada fungsi *if* untuk kabut, terdapat variablel u sebagai *set point* terjadinya kabut. Variabel *u* ini digunakan untuk pengkalibrasian terjadinya kabut, karena berdasarkan eksperimen terjadinya kabut berbeda di tiap ketinggian. Agar alat dapat digunakan di tempat yang berbeda, kondisi terjadinya kabut ini dapat disetting ulang melalui internet.

Setelah bagian dari pengolahan data dan oprasi matematis selesai, langkah selanjutnya adalah menerima dan mengirim data dari database ke mikrokontroler. Alur kerja penerimaan dan pengiriman ini dilakukan bersamaan. Untuk penerimaan data, mikrokontroler di set untuk membaca *array* yang telah disesuaikan pada SQL yang menghubungkan mikrokontroler dan *database*. Dengan begitu, program yang dibuat dalam mikrokontroler haruslah berupa *array* juga. Pembacaan *array* ini menggunakan *library JsonParse* yang telah di inisialisasikan sebelumnya. pembacaan *array* ini dilakukan dengan membaca *array* per kolom dalam satu baris. Pembacaan ini selanjutnya akan dijadikan referensi untuk pengontrolan alat dan pengkalibrasian. Berikut kode yang telah dibuat :

```
while (client.available()) {
   char c = client.read();
            if (c == '<' ) { //'<' is our begining character
              strLineResult = "";
              startRead = true; //Ready to start reading the part
              }
              else if(startRead){
                    if(c != '>'
                    strLineResult += c;
              }
               else{
               startRead = false;
               char charBuf[(strLineResult.length() + 1)];
               strLineResult.toCharArray(charBuf,
(strLineResult.length() + 1));
               JsonParser<32> parser;
               JsonObject root = parser.parse(charBuf);
          if (root.success())
          {
            }
```

```
//perangkat 1
char* nyala1 = root["status"][0];
if (strcmp (nyala1,"0") == 0)
{
  if(strcmp (statdriv,"1") == 0)
  {
    digitalWrite(drv, LOW);
  }
}
else if (strcmp (nyala1,"1") == 0)
{
   if(strcmp (statdriv,"0") == 0)
  {
    digitalWrite(drv, HIGH);
  }
```

Fungsi *while* digunakan disini agar perintah yang untuk membaca dan mengirim data diprioritaskan, sehingga proses akan dilanjutkan jika proses penerimaan dan pengiriman data telah selesai dijalankan. Fungsi *if client.available* digunakan untuk membaca ketersediaan koneksi mikrokontroler untuk menerima dan mengirim data.

Char c digunakan untuk membaca string sql yang diawali dengan "<" dan diakhiri dengan ">". Setelah pembacaan *string* selesai, *char c* akan membaca data *array* yang telah dirubah di dalam sql untuk digunakan sebagai kontrol dan *set point* yang akan di panggil oleh *Json parser*.

Untuk memanggil data *array* pertama, digunakan bagian *root["status"][0]*, dan untuk menggambil data *array* kedua digunakan root["status"][1];, hal ini dikarenakan *array* dimulai dari kolom nol.

Pada bagian *char* nyala1*, digunakan untuk menginisialisasi data yang didapatkan oleh nyala satu adalah sebuah karaker atau string. Sehingga jika data ini akan diambil untuk pengkalibrasian, data ini harus diubah menjadi bentuk integr dengan menggunakan perintah *atoi*. Perintah atoi ini bekerja agar data yang diambil hanya berupa *integer* dan mengabaikan karakter. Maka, untuk mendapatkan *integer* dari *array* yang akan digunakan sebagai *set point* digunakan kode sebagai berikut:

```
char* nyala3 = root["status"][2];
u = atoi (nyala3) ;
```

Proses penerimaan data dilakukan bersamaan dengan pengirimana data. Pengiriman data ini sebenarnya bukan datanya yang di kirim, melainkan *Sql* lah yang memeriksa ketersedian data pada mikrokontroler. Saat pemeriksaat ketersedian data inilah sql memberikan data.

Proses pengiriman data ini dilakukan dengan menggunkan perintah *client.print*. yang artinya, data diprint pada *client* sehingga data dapat diambil oleh *sql*. Dengan telah diperiksanya koneksi *sql* pada fungsi *while*, pengiriman ini hanya memeriksa koneksi saja dengan perintah *if(!client.connected())*. Jika sebelumnya pada fungsi *while* koneksi telah terdeteksi maka koneksi *client* akan disambungkan. Setelah fungsi ini menemukan jawaban dari if tadi selanjutnya digunak fungsi *if (client.connect("192.168.1.12",80))* untuk memeriksa alamat tujuan pengiriman data tersebut. Jika alamat telah sesuai dengan fungsi *if* tadi, maka selanjutnya adalah proses pengiriman data dengan menggunakan *client.print* .

Agar data yang dikirimkan sesuai dengan *sql* yang akan menerimannya maka dituliskan nama *sql* tersebut yang nantinya akan dibaca sebagai *string* seperti *Client.print("GET /add.php?")*. *"GET /add.php?"* ini diartikan bahwa data akan dikirim ke sql yang bernama *add.php*. tanda tanya disana berfungi sebagai penginput data. Setelah itu ditambahkanlah data yang akan diinputkan dibagian berikutnya dengan perintah *client.print("Ti="), client.print(Tin), client.print("&Hi=") dan client.print(Hin)*. *"Ti="* merupakan inisial yang akan dibaca oleh sql dan *Tin* adalah variabel yang akan diinputkan ke *database*. Pada perintah selanjutnya "*&Hi*" adalah inisial untuk variabel kedua. Untuk variabel kedua, ketiga dan seterusnya. Digunakan tambahan "&" diawalnya sebagai pembatas data pertama dan kedua.

Dari perintah diatas jika dituliskan dalam addres browser akan sama dengan *localhost/add.php?Ti=Tin&Hi=Hin/* yang artinya data *Tin* dan *Hin* akan

dimasukan ke *sql* bernama *add.php* dengan inisial *Ti* dan *Hi*. Maka pemograman untuk pengiriman selengkapnya adalah sebagai berikut:

```
if(!client.connected()) {
      if (client.connect("192.168.1.12",80)) {
        Serial.println("connected");
        client.print("GET /add.php?");
        client.print("Ti=");
        client.print(Tin);
        client.print("&Hi=");
        client.print(Hin);
        client.print("&To=");
        client.print(Tout);
        client.print("&Ho=");
        client.print(Hout);
        client.print("&Hujan=");
        client.print(hjn);
        client.print("&kbt=");
        client.print(k);
        client.print("&dsc=");
        client.print(dsci);
        client.print("&emb=");
        client.print(embi);
        client.print("&hujann=");
        client.print(HUJAN);
        client.println(" HTTP/1.1");
        client.println("Host: 192.168.1.12");
        client.println("Connection: close");
        client.println();
      }
  }
```

Setelah proses pengiriman selesai, selanjutnya memasukan kendali otomatis seperti pada gambar 2.16 dimana sistem tidak akan bekerja saat terjadi hujan sampai menunggu hujan selesai yang selanjutnya sprayer dan wiper bekerja dan kembali ke awal *looping*. Dalam pemograman ini, kerja alat yang sesuai dengan keadaan ini dapat menggunakan fungsi if dan while seperti pada kode berikut :

```
      if (HUJAN == LOW)  {
          digitalWrite(sprayer, LOW);
          digitalWrite(driver1, LOW);
          digitalWrite(driver2, LOW);
          hujan = true;
          hujan_selesai = false;
           }
        else if (HUJAN == HIGH)  {
           if (hujan == true)  {
            hujan = false;
            hujan_selesai = true;
            if (wiper_ON == false) {
              wiper_ON = true;



        i = 0;
        while (i < 5) { gerakkan_wiper();  }
        wiper_ON = false;
      }
     }
  }

if (hujan == false && hujan_selesai == true)  {
      if (Tout - 5 < Tdin)  {
      Serial.println("GANTI SILIKA GEL ");
    }
    else if (Tout - 5 >= Tdin)  {                    // kondisi aman
      Serial.print("TAK BEREMBUN DI DALAM");
      if (Hout > 85)  {                             // kondisi kabut
        if (wiper_ON == false) {
          wiper_ON = true;
          i = 0;
          while (i < 5) { gerakkan_wiper();  }
          wiper_ON = false;
else if (Hout <= 90)  {                     // kondisi tidak berkabut
      }
    }
  }
}
```

Program ini menunjukkan bahwa ketika hujan terjadi semua sistem dimatikan sedangkan *bolean hujan* dibuat status *true* dan *bolean hujan_selesai* dibuat status *false*. Ketika hujan berhenti dan status *bolean hujan true*, maka status *hujan_selesai* menjadi *true* dan wiper dijalankan.

Pada bagian *looping* selanjutnya, keadaan berembun didalam akan dipenuhi jika Tdin>Tou. Jika Tdin>Tout maka dibuat pemberitahuan bahwa silika gel sudah tidak layak digunakan dan tidak terjadi pengembunan di dalam. tetapi

jika Tdin<Tout maka dibuat pemberitahuan bahwa Silika gel masih bagus dan tidak terjadi pengembunan di dalam. Pada bagian *looping*selanjutnya, dideklarasikan bahwa pemberitahuan berkabut digunakan juka Hout>s atau set point untuk terjadinya kabut.

Pada bagian selanjutnya adalah penggerak aktuator yang dipisahkan dalam *void*. Bagian ini berisi perintah untuk menjalankan relay spray dan wiper bulak-balik selama hitungan 5 yang diinisialkan sebagai *i* jika *boolean* wiper_bergerak *true*

1.4.3 Perancangan *database* menggunakan XAMPP

Pada database MySql ini akan dibuat satu database dan dua tabel. Tabel pertama digunakan untuk data masuk dari mikrokontroler dan tabel yang kedua untuk data yang dikirim ker mikrokontroler. Database ini memiliki nama database "ape" dan nama tabel pertam "monitoring" sedangkan tabel kedua diberi nama "kontrol".

Pada tabel monitoring terdapat 9 kolom yang berisi *no, waktu, suhuluar, kelembabanluar, suhudalam, kelembabandalam, hujan, dsc, kabut, embun dan hujann*. Dengan stuktur sebagai berikut

Gambar 3. 7 Struktur *database*

Pada gambar 3.7 dapat dilihat bahwa yang dijadikan kunci (*primary)* adalah kolom no dengan tipe *integer* yang berisi maksimal 11 digit. Maka untuk kolom no, isi yang akan masuk tidak boleh ada data yang sama. kolom no ini dijadikan *Auto increment* agar data yang diisikan otomatis terisi dengan nilai

selanjutnya tanpa harus *diinsert* pada perintah di sql. kolom waktu memiliki tipe data waktu yang di setting *datetime* yang artinya data waktu yang dituliskan di kolom ini akan memiliki format waktu YYYY-MM-DD ; hh:mm:ss.

Kolom *suhuluar, kelembabanluar, suhudalam, kelembabandalam dan hujann* memiliki format *integer* dengan jumlah digit maksimal 11 digit. Kolom ini akan diisi oleh data hasil pengukuran SHT11 yang berada di luar ,SHT11 yang berada didalam dan sensor hujan. Kolom selanjutnya yaitu kolom hujan, dsc, kabut dan embun menggunakan tipe *text* yang memiliki jumlah digit yang panjang sehingga dapat menampung komentar yang memiliki jumlah digit yang banyak. data yang dikirim ke kolom ini adalah data keaadan dari silika gel, embun, hujan dan embun yang telah dideklarasikan didalam mikrokontroler.

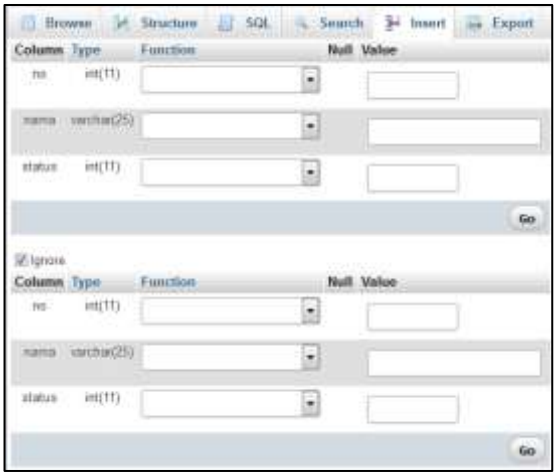
Pada tabel kedua, isi kolom yang dibuat antaralain no, nama dan status yang memiliki struktur seperti gambar 3.8 berikut.

	#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
☐	1	no	int(11)			No	None	AUTO_INCREMENT	Change Drop Primary Unique Index
☐	2	nama	varchar(25)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index
☐	3	status	int(11)			No	None		Change Drop Primary Unique Index

Gambar 3. 8 Struktur tabel kontrol

Pada gambar 3.8 tedapat Kolom no menggunakan struktur type yang sama dengan kolom no pada tabel "monitoring". Kolom nama menggunakan tipe *varchar* yang memiliki batas maksimal digit 25 digit, dengan *collation latin1_swedish_ci* . kolom status menggunakan tipe *integer* dengan batas maksimal digit 11 digit.

Pada tabel kontrol ini, data yang akan masuk ke *database* digunakan metode *update*, sehingga data pada tabel ini harus ada terlebih dahulu. Untuk menginput data kedalam tabel digunakan *menu insert* yang terdapat pada *php myAdmin*. Pada fungsi insert ini terdapat formulir seperti pada gambar 3.8.



Gambar 3. 9 Formulir fungsi insert

Pada gambar 3.9 terdapat kolom formulir disebelah kanan dan fungsi disebelah kiri. Pada kolom value untuk baris pertama *no* tuliskan "1" , *nama* tuliskan "driver" dan *status* tuliskan "0". Pada baris kedua *no* tuliskan "2" , *nama* tuliskan "*sprayer*" dan *status* tuliskan "0". Pada baris ketiga *no* tuliskan "3" , *nama* tuliskan "kabut" dan *status* tuliskan "90". Pada baris keempat *no* tuliskan "4" , *nama* tuliskan "td" dan *status* tuliskan "12". Setelah semua terisi, klik go untuk menginsert dan hasilnya akan seperti gambar 3.10.

Gambar 3. 10 Isi tabel kontrol

Pada Gambar 3.10, data yang berada pada tabel "kontrol" telah terdapat data *driver, sprayer, kabut dan td* yang nantinya data dari kolom "status" akan dikirim ke mikrokontroler, maka kolom "status" yang datanya akan di*update* sesuai dengan data yang diinginkan.

1.4.4 Perancangan pengiriman dan penerimaan data antara mikrokontroler dan database XAMPP

Perancangan ini akan dilakukan dengan membuat file bertipe *php* dengan fungsi *Sql* dari *MySql* dengan kode sebagai berikut.

```
<?php
    function Connection(){
          $server="localhost";
          $user="root";
          $pass="";
          $db="ape";
          $connection = mysql_connect($server, $user, $pass);
          if (!$connection) {
          die('MySQL ERROR: ' . mysql_error());
          }
          mysql_select_db($db) or die( 'MySQL ERROR: '.
mysql_error() );
          return $connection;
    }
```

```
   $link=Connection();
      $to1=$_GET["To"];
      $ho1=$_GET["Ho"];
      $ti1=$_GET["Ti"];
      $hi1=$_GET["Hi"];
      $hjn1=$_GET["Hujan"];
      $dsc1=$_GET["dsc"];
      $kbt1=$_GET["kbt"];
      $emb1=$_GET["emb"];
      $hjnn=$_GET["hujann"];

      $query = "INSERT INTO `monitoring` (`waktu`,`suhuluar`,
`kelembabanluar`
,`suhudalam`,`kelembabandalam`,`hujan`,`dsc`,`kabut`,`embun`,`hujann`)
VALUES NOW(),$to1,$ho1,$ti1,$hi1,'".$hjn1."','".$dsc1."','".$kbt1."',
'".$emb1."',$hjnn)";

mysql_query($query,$link) or die(mysql_error());
      mysql_close($link);
      $link       =Connection();
      $sql        = "SELECT status FROM kontrol";
      $result           = mysql_query($sql,$link);
      if(mysql_num_rows($result) > 0)
      {
            $records          = array();
            while($row = mysql_fetch_array($result)) {
                  $records[] = $row[0];
            }
            mysql_close($link);
            $data = json_encode($records);
            //$data = str_replace("[","",$data);
            //$data = str_replace("]","",$data);
            echo '<{"status":'.$data."}>";
      }
?>
```

Pada kode sql ini, hal yang pertama dilakukan adalah membuat fungsi koneksi dengan *database* dengan kode *function Connection()* yang berisi alamat database diletakan. Alamat *database* disetting sesuai dengan identitas *database* yang digunakan maka *server* disetting *localhost*, *user* disetting *root*, *pass* dikosongkan dan *db* di setting *ape*.

Setelah php ini terhubung dengan *database* selanjutnya mengambil data mikrokontroler dengan membuat kode *$_GET[""];* setelah data diambil data kemudian dikirim dengan *query INSERT INTO*. Untuk mengambil data dari database dengan *query SELECT*. Data yang dipanggil ini berupa *string* yang akan

dirubah menjadi *array*. Dari hasil array ini akan di panggil oleh JsonParser yang telah dimasukan kedalam mikrokontroler.

1.4.5 Perancangan tampilan

Perancangan tampilan yang menggunakan *jquery bootstrap* untuk menampilkan data yang berada di database. Ada tiga macam yang akan ditampilkan dalam halaman web ini, diantaranya tampilan pengukuran suhu dan kelembaban dalam grafik, pemberitahuan keadaan *dessicant*, kabut dan hujan dalam tabel, serta tampilan pengontrolan dan penentuan *setpoint*.

Pada tampilan grafik, digunakan *ajax* untuk memanggilnya, dalam kode *ajax* ini telah terdapat perintah untuk memanggil data dan mengirim data dengan *database*. Data yang telah terpanggil akan dimasukan ulang kedalam *fungsi* yang akan menampilkan grafik. *Query* untuk tampilan grafik digunakan *Highchart*. Untuk mendapatkan grafik *realtime*, digunakan *auto refresh* pada halaman agar bisa memanggil data jika ada data yang masuk.

Pada tampilan tabel data yang diambil merupakan 100 data terakhir dari pengukuran. Pada tampilan tabel juga digunakan *auto refresh* agar data yang baru bisa langsung ditampilkan. Tampilan pada tabel digunakan *query bootstrap table* sehingga pengaturan untuk lebar dan hasil tampilan grafik dapat disesuaikan dengan mengubah *query* pada *bootstrap table*

Pada tampilan kontrol dan setpoint akan dibuat dua tombol on dan off untuk driver dan sprayer. Sementara itu untuk set point sendiri diberikan opsi untuk di pilih yang kemudian disediakan tombol kirim untuk mengirim set point yang telah ditentukan.

Tampilan ini didesain dengan menggunakan *jquery bootstrap*. Penempatan tampilan ini diatur dengan menggunakan *grid*. Sehingga penempatan tampilan grafik, tabel dan kendali tidak dapat sembarangan di simpan dimana saja.

1.4.6 Perancangan komunikasi database dan tampilan

Komunikasi database ke halaman web menggunaankan *ajax*. *Ajax* ini bekerja membaca data yang berada di *database* dan mengirimnya ke halaman web. Begitu juga dalam pengiriman data dari halaman web ke *database*, *ajax* digunakan untuk membaca keadaan pada halaman web yang kemudian dikirimkan ke *database*.

Untuk setiap tampilan digunakan *ajax* yang berbeda. pada bagian kontrol terdapat empat *ajax* yang digunakan untuk mengirim dua data driver *on* dan *off*, dan untuk mengirim dua data sprayer *on* dan *off*. Data *on* dan *off* ini akan dirubah menjadi nol dan satu, sehingga data yang masuk database 0 dan 1. Sedangkan Pada bagian *set point* digunakan dua *ajax* untuk mengirim data *set point* terjadi kabut dan *set point* untuk pengaman terjadinya kabut.

Untuk tampilan tabel dan grafik digunakan masing masing satu *ajax* yang bekerja memanggil data dari database untuk dimasukan kedalam fungsi grafik danpun tabel. Data untuk grafik dipanggil menggunakan inisial hari. Sedangkan data untuk tabel, hanya dipanggil 100 data terakhir.