LAMPIRAN

Plagiarism Checker X Originality Report

Similarity Found: 23%

Date: Saturday, April 06, 2019
Statistics: 4751 words Plagiarized / 20420 Total words
Remarks: Medium Plagiarism Detected - Your Document needs Selective Improvement.

RANCANG BANGUN SISTEM MONIT ORING DAN KONTROL RUMAH DENGAN MODEL CLIENT -SER VER MENGGUNAKAN NODEMCU ESP-12E BERBASIS INTERNET OF THINGS (IoT) (Skripsi) Oleh: ADAM RABBANI ADNAN F AKUL T AS TEKNIK UNIVERSIT AS LAMPUNG BANDAR LAMPUNG 2019ABSTRAK RANCANG BANGUN SISTEM MONIT ORING DAN KONTROL RUMAH DENGAN MODEL CLIENT -SER VER MENGGUNAKAN NODEMCU ESP-12E BERBASIS INTERNET OF THINGS (IoT) Oleh Adam Rabbani Adnan T eknologi keamanan rumah yang digunakan saat ini berbasis web, sehingga pemantauan keamanan rumah hanya bila pemilik membuka alamat web. Salah satu solusi untuk mengetahui kondisi keamanan dan kontrol rumah tiap waktu maka diperlukan alat yang dapat memantau melalui aplikasi smartphone dengan koneksi internet menggunakan konsep Internet of Things (IoT). Pada penelitian ini membahas rancang bangun sistem pemantauan dan kontrol rumah menggunakan NodeMCU ESP-12E. Alat ini dirancang agar keamanan rumah tetap terjaga dengan memantau kondisi dan kontrol rumah dengan memasang alat yaitu client dipasang di tiap ruang dengan jumlah 4 client dan terdapat 1 server . Keseluruhan sistem alat ini dapat mendeteksi gas LPG, mendeteksi motion , kondisi lampu, kondisi pintu, mengontrol kunci pintu elektronik dan pemilik rumah dapat

Source Code Server :

```
#include <ESP8266WiFi.h>
#include <ESP8266mDNS.h>
#include <WiFiServer.h>
#include <WiFiClient.h>
#include <DNSServer.h>
#include <ESP8266WebServer.h>
#include <WiFiManager.h>
#include <WiFiUdp.h>

#include <FirebaseArduino.h>

#include <SimpleTimer.h>

#include <TimeLib.h>

char data[200] ={};
int packetsize = 0;
String receiveddata="";
const int port_server = 5555;
const int port_client = 5556;
WiFiUDP udp_client_server;

char json[200];

#define FIREBASE_HOST "rckd2018.firebaseio.com"
#define FIREBASE_AUTH
"T6oStlMlzFx4YMArLQ22XNa7b5XpTtIMQtDpSeyn"



char* ssid_AP = "ESP8266-RCKD";
char* pass_AP = "4adnanwifi";

char ssid[] = "CPPBT Wifi"    ;
char pass[] = "punyasini";
char* mdns = "iot";
```

```
unsigned int Udp_waktuPort = 2311;

static const char ntpServerName[] = "id.pool.ntp.org";
const int timeZone = 7;
time_t prevDisplay = 0;
uint8_t referensi_waktu = 0;
const unsigned long referensi_timestamp = 1357041600;

WiFiUDP Udp_waktu;
WiFiUDP Udp_broadcast;

unsigned int Udp_waktu_Port = 123;
unsigned int Udp_broadcast_Port = 2311;

WiFiClient client;

int port_web = 8000;
String str_port_web = String(port_web);

ESP8266WebServer server(port_web);

IPAddress ip(192, 168, 1, 177);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 255, 0);
IPAddress dns(8, 8, 8, 8);

IPAddress ip_AP(192, 168, 4, 1);
IPAddress subnet_AP(255, 255, 255, 0);

IPAddress broadcastIp;
IPAddress broadcastIp_AP;

const char* host = "www.google.com";

SimpleTimer timer;
byte interval_otomatis = 10;
byte interval_otomatis_num = 10;
```

```
int Respon_Interval_Relay_Otomatis;

WiFiServer    Server(9001);

int ClientESP;
int nclient;
int urutan_koneksi = 1;
int max_urutan_koneksi = 3;
byte Check_Firebase = 1;

const byte numKoneksi = 2;
int koneksi[numKoneksi];
byte gagal[2];

String IPlocal;
String Argumen_Server, Respon_Web_Client;
String lokasi_web_input_data = "/data";

int
Password_Beranda_Client,baru_Password_Beranda_Client,Respon_Password_Be
randa_Client;
int Password_Beranda_Server = 123456;
String Lokasi_Password_Beranda_Client = "/Password_Beranda";
String Lokasi_Password_Beranda_Server = "/status/Password_Beranda";

const byte numClient = 4;
byte gagal_Koneksi_Client[numClient];

byte Kondisi_Koneksi_numClient[numClient];
byte baru_Kondisi_Koneksi_numClient[numClient];
byte Kondisi_Koneksi_Client[numClient];
byte baru_Kondisi_Koneksi_Client[numClient];

String Lokasi_Kondisi_Koneksi_Client[numClient] =
{"Koneksi_Client_A","Koneksi_Client_B","Koneksi_Client_C","Koneksi_Client
_D"};
```

```
String Mode = "1";
String Lokasi_Mode = "/status/Mode";

const byte numPerubahan_Keadaan = 2;

String Perubahan_Kondisi = "1";
String Perubahan_Keadaan[numPerubahan_Keadaan] = {"1","1"};
String Lokasi_Perubahan_Kondisi = "/status/Perubahan";
String Lokasi_Perubahan_Keadaan = "Perubahan";

const byte numLampu_Otomatis = 2;
int Lampu_Otomatis_A[numLampu_Otomatis];
int Lampu_Otomatis_B[numLampu_Otomatis];

const byte numSensorA = 1;
const byte numSensorB = 2;
const byte numSensorC = 2;

String Lokasi_SensorA[numSensorA] = {"Sensor_A_0"};
String Lokasi_SensorB[numSensorB] = {"Sensor_B_0", "Sensor_B_1"};
String Lokasi_SensorC[numSensorC] = {"Sensor_C_0","Sensor_C_1"};

int SensorA[numSensorA];
int SensorB[numSensorB];
int SensorC[numSensorC];

int baru_SensorA[numSensorA];
int baru_SensorB[numSensorB];
int baru_SensorC[numSensorC];



const byte numRelayA = 2;
const byte numRelayB = 3;
const byte numRelayC = 1;
const byte numRelayD = 3;
```

```
int Kondisi_Relay_A[numRelayA] ; // = {0,0,0,0} ;
int Kondisi_Relay_B[numRelayB]; // ={0,0,0} ;
int Kondisi_Relay_C[numRelayC];//    = {0};
int Kondisi_Relay_D[numRelayD];

String Lokasi_Kondisi_Relay_A[numRelayA] = {"/status/Relay_A_0",
"/status/Relay_A_1"};
String Lokasi_Kondisi_Relay_B[numRelayB] = {"/status/Relay_B_0",
"/status/Relay_B_1", "/status/Relay_B_2"};
String Lokasi_Kondisi_Relay_C[numRelayC] = {"/status/Relay_C_0"};
String Lokasi_Kondisi_Relay_D[numRelayD] =
{"/status/Relay_D_0","/status/Relay_D_1","/status/Relay_D_2"};

String baru_Kondisi_Relay_A[numRelayA] ;
String baru_Kondisi_Relay_B[numRelayB] ;
String baru_Kondisi_Relay_C[numRelayC] ;
String baru_Kondisi_Relay_D[numRelayD] ;

const byte numKeadaan_RelayA = 2;
const byte numKeadaan_RelayB = 3;
const byte numKeadaan_RelayC = 2;
const byte numKeadaan_RelayD = 3;

int Keadaan_Relay_A[numKeadaan_RelayA]    ;
int Keadaan_Relay_B[numKeadaan_RelayB]    ;
int Keadaan_Relay_C[numKeadaan_RelayC]    ;
int Keadaan_Relay_D[numKeadaan_RelayD]    ;

int baru_Keadaan_Relay_A[numKeadaan_RelayA] ;
int baru_Keadaan_Relay_B[numKeadaan_RelayB] ;
int baru_Keadaan_Relay_C[numKeadaan_RelayC] ;
int baru_Keadaan_Relay_D[numKeadaan_RelayD] ;

String Lokasi_Keadaan_Relay_A[numKeadaan_RelayA] =
{"Keadaan_Relay_A_0", "Keadaan_Relay_A_1"};
String Lokasi_Keadaan_Relay_B[numKeadaan_RelayB] =
{"Keadaan_Relay_B_0", "Keadaan_Relay_B_1", "Keadaan_Relay_B_2"};
```

```
String Lokasi_Keadaan_Relay_C[numKeadaan_RelayC] =
{"Keadaan_Relay_C_0", "Keadaan_Relay_C_1"};
String Lokasi_Keadaan_Relay_D[numKeadaan_RelayD] =
{"Keadaan_Relay_D_0", "Keadaan_Relay_D_1", "Keadaan_Relay_D_2"};

boolean Client_Terkoneksi = false;
boolean Terakhir_Client_Terkoneksi = false;

const int numMax_Kondisi_Relay = (numRelayA - 1) + numRelayB +
numRelayC + numRelayD; // -1
int Max_Kondisi_Relay[numMax_Kondisi_Relay];

int channel = 1;
int ssid_hidden = 0;
int max_connection = 5;

void setup() {
   Serial.begin(115200);
   Serial.println("");
   Serial.println(ESP.getFullVersion());
   SetWifi();



   server.on("/keadaan", Server_Keadaan);
   server.on("/reset", Server_Reset);
   server.on("/respon", Respon_Client);

   server.on("/data", Server_Input_data);
   server.on("/",Server_Halaman_Utama);
   server.begin();

   Server.begin();
   MDNS.addService("http","tcp", 8000);
     udp_client_server.begin(port_client);

//Wifi_Status();
```

```
check_info();
Serial.println("==================");

   timer.setInterval(300L, Menerima_Client_Data);
   timer.setInterval(500L, Proses_Client_Data);
   timer.setInterval(15000L, Wifi_Status);
   timer.setInterval(4000L, Firebase_data);
   timer.setInterval(500L,check_info);
   timer.setInterval(5000L,Kondisi_ESP);
   timer.setInterval(1000L,Broadcast_data);



}

void Firebase_auth() {
   Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
}

void loop () {
   handle_client();
   timer.run();
}

void check_info(){
  Serial.println(ESP.getFreeHeap());
   }

void handle_client(){
   server.handleClient();
   }

void Wifi_Status() {

   if (WiFi.status() != WL_CONNECTED)
   {
      koneksi[0] = 0;
```

```
    SetWifi();
    Serial.println("tidak terkoneksi Wifi");
    Perubahan_Kondisi = "1";
  }
  else if (WiFi.status() == WL_CONNECTED) {
    koneksi[0] = 1;
    Serial.print("terkoneksi Wifi:");
    IPlocal = WiFi.localIP().toString();
    Serial.println(IPlocal);
    Test_Internet();
  }
}


void Test_Internet() {

  if (client.connect(host, 80))
  {
    Serial.println("Terkoneksi Internet");
    client.stop();
    koneksi[1] = 1;
      Zona_Waktu();
      Inisial_kondisi_keadaan_firebase();
  }

  else {
    Serial.println("Tidak Terkoneksi Internet");
    client.stop();
    koneksi[1] = 0;
    if(Mode=="1"){
      gagal[0] = gagal[0]+1;
      Perubahan_Kondisi = "1";
      }
  }

}
```

```
void SetWifi()
{

   WiFi.disconnect();
   WiFi.mode(WIFI_AP_STA);
   WiFi.config(ip, dns, gateway, subnet);
   WiFi.begin(ssid, pass);
   delay(4000);

WifiManager_Set();

      WiFi.softAP(ssid_AP, pass_AP, channel, ssid_hidden, max_connection);
   Serial.println("WIFI < " + String(ssid_AP) + " > ... Started");
   IPAddress IP = WiFi.softAPIP();
   Serial.println("AccessPoint IP : ");
   Serial.print(IP);

if (MDNS.begin(mdns)) {

Serial.println("MDNS ==> http://" + String(mdns) + ".local");

}

}



void WifiManager_Set() {

   WiFiManager wifiManager;
   wifiManager.setConfigPortalTimeout(10);
   wifiManager.setSTAStaticIPConfig(ip, gateway, subnet, dns);
   wifiManager.setAPStaticIPConfig(ip_AP, ip_AP, subnet);
//    wifiManager.autoConnect(ssid_AP, pass_AP);
}

void Zona_Waktu(){
```

```
    if ((koneksi[0] == 1) && (koneksi[1] == 1)) {
    if( day() != referensi_waktu) {

  Serial.println("Starting Udp_waktu");
  Udp_waktu.begin(Udp_waktu_Port);
  Serial.print("Local port: ");
 Serial.println(Udp_waktu.localPort());
  Serial.println("waiting for sync");
  setSyncProvider(getNtpTime);

        if( now() >= referensi_timestamp) {
      referensi_waktu = day();
      }
      else{
        referensi_waktu = 0;
        }
 // setSyncInterval(300);

  if (timeStatus() != timeNotSet) {
    if (now() != prevDisplay) { //update the display only if time has changed
      prevDisplay = now();
      Menampilkan_Waktu();
    }
  }
  }
}
}

    void Kondisi_ESP() {
  if (gagal[0] >= 3) {

  }

  Kondisi_Client();

}
```

```
void Kondisi_Client(){

   for(byte i = 0 ; i<numClient ; i++){

      if(Kondisi_Koneksi_numClient[i]==baru_Kondisi_Koneksi_numClient[i]){
         gagal_Koneksi_Client[i]= gagal_Koneksi_Client[i]+1;

         }

         else if
(Kondisi_Koneksi_numClient[i]!=baru_Kondisi_Koneksi_numClient[i]){
              Kondisi_Koneksi_Client[i] = 1;
            baru_Kondisi_Koneksi_numClient[i] = Kondisi_Koneksi_numClient[i];
            }

if(gagal_Koneksi_Client[i]>=1){
   Kondisi_Koneksi_Client[i] = 0;
   gagal_Koneksi_Client[i]=0;
   }


if((Kondisi_Koneksi_numClient[i]>=5)||(baru_Kondisi_Koneksi_numClient[i]>=
5)){
      Kondisi_Koneksi_numClient[i]=1;
      baru_Kondisi_Koneksi_numClient[i]=0;
      }

   }
   }


/*=========================================================
=========================================================
=======================*/
```

```
void Firebase_save() {

   if ((koneksi[0] == 1) && (koneksi[1] == 1)) {


for(byte i = 0; i<numSensorA;i++){

   if (SensorA[i] != baru_SensorA[i]) {
         Firebase.setInt(Lokasi_SensorA[i], SensorA[i]);
      }
   }

for(byte i = 0; i<numSensorB;i++){
   if (SensorB[i] != baru_SensorB[i]) {
   }

   for(byte i = 0; i<numSensorC;i++){
   if (SensorC[i] != baru_SensorC[i]) {
         Firebase.setInt(Lokasi_SensorC[i], SensorC[i]);
         baru_SensorC[i] = SensorC[i];
         Perubahan_Keadaan_Firebase();
      }
   }

   for(byte i = 0; i<numKeadaan_RelayA; i++){

       if (Keadaan_Relay_A[i] != baru_Keadaan_Relay_A[i]) {
      }

      }

   for(byte i = 0; i<numKeadaan_RelayB; i++){
       if (Keadaan_Relay_B[i] != baru_Keadaan_Relay_B[i]) {
         Firebase.setString(Lokasi_Keadaan_Relay_B[i],
String(Keadaan_Relay_B[i]));
         Firebase.setString(Lokasi_Kondisi_Relay_B[i],
String(Keadaan_Relay_B[i]));
```

```
      baru_Keadaan_Relay_B[i] = Keadaan_Relay_B[i];
      Perubahan_Keadaan_Firebase();
    }

    }

      for(byte i = 0; i<numKeadaan_RelayC; i++){

     if (Keadaan_Relay_C[i] != baru_Keadaan_Relay_C[i]) {
      baru_Keadaan_Relay_C[i] = Keadaan_Relay_C[i];
      Perubahan_Keadaan_Firebase();
    }

    }

            for(byte i = 0; i<numKeadaan_RelayD; i++){

     if (Keadaan_Relay_D[i] != baru_Keadaan_Relay_D[i]) {
    }

    }

    for (byte i = 0; i <numClient;i++){
      if (Kondisi_Koneksi_Client[i]!=baru_Kondisi_Koneksi_Client[i]){
    }

    if(Password_Beranda_Client!=baru_Password_Beranda_Client){
    baru_Password_Beranda_Client = Password_Beranda_Client;
    Perubahan_Keadaan_Firebase();
    }

  }

}
```

```
void Perubahan_Keadaan_Firebase(){
   Perubahan_Keadaan[1] = "1" ;
   if(Perubahan_Keadaan[1] == "1" ){
   Firebase.setString(Lokasi_Perubahan_Keadaan, Perubahan_Keadaan[0]);
   Perubahan_Keadaan[1] = "0";
   }
   }

void Firebase_baca() {

   if ((koneksi[0] == 1) && (koneksi[1] == 1)) {

      Perubahan_Kondisi = Firebase.getString(Lokasi_Perubahan_Kondisi)    ;

      Check_error_firebase();

      if (Perubahan_Kondisi == "1") {

         Serial.println("Ada Perubahan Data");
         Mode = Firebase.getString(Lokasi_Mode);

            Password_Beranda_Server =
(Firebase.getString(Lokasi_Password_Beranda_Server)).toInt();

         Firebase.setString(Lokasi_Perubahan_Kondisi, "0");
         Perubahan_Kondisi = "0";
         Firebase_sync_data();
      }

   }
}

void Firebase_sync_data() {
for(byte i = 0; i<numRelayA; i++){

      Kondisi_Relay_A[i] = baru_Kondisi_Relay_A[i].toInt();
   }
```

```
    for(byte i = 0; i<numRelayB; i++){

        Kondisi_Relay_B[i] = baru_Kondisi_Relay_B[i].toInt();

    }

    for(byte i = 0; i<numRelayC; i++){

        Kondisi_Relay_C[i] = baru_Kondisi_Relay_C[i].toInt();

    }

        for(byte i = 0; i<numRelayD; i++){

        Kondisi_Relay_D[i] = baru_Kondisi_Relay_D[i].toInt();

    }
}

void Menerima_Client_Data()
{
  if (Check_Firebase == 0) {

        char message = udp_client_server.parsePacket();
        packetsize = udp_client_server.available();

        if (message)
        {
  Serial.println(packetsize);

          udp_client_server.read(json,packetsize);
            Serial.println(json);
          //.readStringUntil('¥n')).toCharArray();
          IPAddress remoteip = udp_client_server.remoteIP();
          DataClient();
```

```
    memset(json, 0, sizeof(json));

    Serial.println(remoteip);
    Serial.println(udp_client_server.remotePort());

packetsize = 0;

        }

}}

void Proses_Client_Data()
{
  if (Check_Firebase == 0) {
broadcastIp_AP = WiFi.softAPIP();

            broadcastIp_AP[3] = 255;

          udp_client_server.beginPacket(broadcastIp_AP,port_server);

        if (i == 0) {

            DataKeClient_1();

        }

        else if (i == 1) {
            DataKeClient_2();

        }

        else if (i == 2) {
            DataKeClient_3();

        }
                else if (i == 3) {
```

```
            DataKeClient_4();

        }

void DataClient() {

    StaticJsonBuffer<200> jsonBuffer;

    JsonObject& root = jsonBuffer.parseObject(json);
    ClientESP = root["ClientESP"];

    if (ClientESP == 1) {

Kondisi_Koneksi_numClient[0] = Kondisi_Koneksi_numClient[0] + 1;

        Serial.print("SensorA");
        Serial.print(SensorA[0]);
        Serial.println(SensorA[1]);

        Serial.print("Keadaan_Relay_A");
        Serial.print(Keadaan_Relay_A[0]);
        Serial.print(Keadaan_Relay_A[1]);
        Serial.println("");


    }

    else if (ClientESP == 2) {

Kondisi_Koneksi_numClient[1] = Kondisi_Koneksi_numClient[1] + 1;

        Serial.print("SensorB");
        Serial.print(SensorB[0]);
        Serial.println(SensorB[1]);

        Serial.print("Keadaan_Relay_B");
        Serial.print(Keadaan_Relay_B[0]);
```

```
        Serial.print(Keadaan_Relay_B[1]);
        Serial.print(Keadaan_Relay_B[2]);
        Serial.println("");

    }

    else if (ClientESP == 3) {

Kondisi_Koneksi_numClient[2] = Kondisi_Koneksi_numClient[2] + 1;

        Serial.print("SensorC");
        Serial.println(SensorC[0]);

        Serial.print("Keadaan_Relay_C");
        Serial.print(Keadaan_Relay_C[0]);
        Serial.print(Keadaan_Relay_C[1]);
        Serial.println("");

    }

    else if (ClientESP == 4) {

Kondisi_Koneksi_numClient[3] = Kondisi_Koneksi_numClient[3] + 1;

        Serial.print("Keadaan_Relay_D");
        Serial.print(Keadaan_Relay_D[0]);
        Serial.print(Keadaan_Relay_D[1]);
        Serial.print(Keadaan_Relay_D[2]);
        Serial.println("");

    }

}
```

```
void DataKeClient_1() {

   udp_client_server.endPacket();
}



void DataKeClient_2() {
   StaticJsonBuffer<200> jsonBuffer2;

   root2.printTo(udp_client_server);
   udp_client_server.endPacket();
}



void DataKeClient_3() {
   StaticJsonBuffer<200> jsonBuffer3;
   JsonObject& root3 = jsonBuffer3.createObject();

   root3["Client"] = 3;

   root3.printTo(udp_client_server);
   udp_client_server.endPacket();
}

void DataKeClient_4() {
   StaticJsonBuffer<200> jsonBuffer4;
   JsonObject& root4 = jsonBuffer4.createObject();

   root4["Client"] = 4;


   root4.printTo(udp_client_server);
   udp_client_server.endPacket();
}
```

```
  void Broadcast_data(){
  if (Check_Firebase == 0) {
broadcastIp = WiFi.localIP();
Serial.println(broadcastIp);
broadcastIp[3] = 255;
broadcastIp_AP[3] = 255;
    }
  }

    void Kirim_kondisi_lokal(){

StaticJsonBuffer<2056> jsonBuffer0;
JsonObject& root0 = jsonBuffer0.createObject();

    JsonArray& koneksi_server = root0.createNestedArray("Koneksi_Server");
    for(byte i = 0 ;i<numKoneksi ; i++){
    koneksi_server.add(koneksi[i]);
    }

        JsonArray& koneksi_client = root0.createNestedArray("Koneksi_Client");
    for(byte i = 0 ;i<numClient ; i++){
    koneksi_client.add(Kondisi_Koneksi_Client[i]);
  }
        }

void Broadcast_data_keadaan_WiFi(){

        JsonArray& koneksi_client = root0.createNestedArray("Koneksi_Client");
    for(byte i = 0 ;i<numClient ; i++){
    koneksi_client.add(Kondisi_Koneksi_Client[i]);
    }

    JsonArray& relay_A = root0.createNestedArray("Relay_A");
    for(byte i = 0 ;i<numKeadaan_RelayA ; i++){
    relay_A.add(Keadaan_Relay_A[i]);
    }
```

```
  JsonArray& relay_B = root0.createNestedArray("Relay_B");
  for(byte i = 0 ;i<numKeadaan_RelayB ; i++){
  relay_B.add(Keadaan_Relay_B[i]);
  }

  JsonArray& relay_C = root0.createNestedArray("Relay_C");
  for(byte i = 0 ;i<numKeadaan_RelayC ; i++){
  relay_C.add(Keadaan_Relay_C[i]);
  }

    JsonArray& relay_D = root0.createNestedArray("Relay_D");
  for(byte i = 0 ;i<numKeadaan_RelayD ; i++){
  relay_D.add(Keadaan_Relay_D[i]);
  }

    JsonArray& sensor_A = root0.createNestedArray("Sensor_A");
  for(byte i = 0 ;i<numSensorA ; i++){
  sensor_A.add(SensorA[i]);
  }

        JsonArray& sensor_B = root0.createNestedArray("Sensor_B");
  for(byte i = 0 ;i<numSensorB ; i++){
  sensor_B.add(SensorB[i]);
  }

        JsonArray& sensor_C = root0.createNestedArray("Sensor_C");
  for(byte i = 0 ;i<numSensorC ; i++){
  sensor_C.add(SensorC[i]);
  }
  root0.printTo(Udp_broadcast);
  Udp_broadcast.endPacket();

  root0.printTo(Serial);
  Serial.println();
     }

void Broadcast_data_keadaan_AP(){
```

```
StaticJsonBuffer<1024> jsonBuffer0;
   JsonArray& koneksi_server = root0.createNestedArray("Koneksi_Server");
   for(byte i = 0 ;i<numKoneksi ; i++){
   koneksi_server.add(koneksi[i]);
   }

      JsonArray& koneksi_client = root0.createNestedArray("Koneksi_Client");
   for(byte i = 0 ;i<numClient ; i++){
   koneksi_client.add(Kondisi_Koneksi_Client[i]);
   }

   JsonArray& relay_A = root0.createNestedArray("Relay_A");
   for(byte i = 0 ;i<numKeadaan_RelayA ; i++){
   relay_A.add(Keadaan_Relay_A[i]);
   }

   JsonArray& relay_B = root0.createNestedArray("Relay_B");
   for(byte i = 0 ;i<numKeadaan_RelayB ; i++){
   relay_B.add(Keadaan_Relay_B[i]);
   }

   JsonArray& relay_C = root0.createNestedArray("Relay_C");
   for(byte i = 0 ;i<numKeadaan_RelayC ; i++){
   relay_C.add(Keadaan_Relay_C[i]);
   }

      JsonArray& relay_D = root0.createNestedArray("Relay_D");
   for(byte i = 0 ;i<numKeadaan_RelayD ; i++){
   relay_D.add(Keadaan_Relay_D[i]);
   }

void Server_Keadaan() {
   String message = "Koneksi_Wifi = [" + String(koneksi[0]) + "]";
   message += "¥n";
   message += "Koneksi_Internet = [" + String(koneksi[1]) + "]";
   message += "¥n";
```

```
   message += "Password_Beranda= [" + String(Password_Beranda_Client) + "]";

       for(byte i = 0; i<numClient;i++){
     message += "Kondisi_Koneksi_Client_"+String(i) +"= [" +
String(Kondisi_Koneksi_Client[i])+ "]";
   message += "¥n";
       }

       for(byte i = 0; i<numClient;i++){
     message += "Kondisi_Koneksi_numClient_"+String(i) +"= [" +
String(Kondisi_Koneksi_numClient[i])+ "]";
   message += "¥n";
       }

   for(byte i = 0; i<numKeadaan_RelayA;i++){
     message += "Keadaan_Relay_A_"+String(i) +"= [" +
String(Keadaan_Relay_A[i]) + "]";
   message += "¥n";
       }

          for(byte i = 0; i<numKeadaan_RelayB;i++){
     message += "Keadaan_Relay_B_"+String(i) +"= [" +
String(Keadaan_Relay_B[i]) + "]";
   message += "¥n";

       }

                for(byte i = 0; i<numKeadaan_RelayC;i++){
     message += "Keadaan_Relay_C_"+String(i) +"= [" +
String(Keadaan_Relay_C[i]) + "]";
   message += "¥n";

       }

                   for(byte i = 0; i<numKeadaan_RelayD;i++){
     message += "Keadaan_Relay_D_"+String(i) +"= [" +
String(Keadaan_Relay_D[i]) + "]";
```

```
  message += "¥n";

    }

          for(byte i = 0; i<numSensorA;i++){
   message += "Sensor_A_"+String(i) +"= [" + String(SensorA[i]) + "]";
  message += "¥n";
    }

        for(byte i = 0; i<numSensorB;i++){
   message += "Sensor_B_"+String(i) +"= [" + String(SensorB[i]) + "]";
  message += "¥n";

    }

            for(byte i = 0; i<numSensorC;i++){
   message += "Sensor_C_"+String(i) +"= [" + String(SensorC[i]) + "]";
  message += "¥n";

    }

  server.send(200, "text/html", message);
}

void Server_Mode_Online() {
  Mode = "1";
  String message = "berhasil";
  server.send(200, "text/plain", message);
}

void Server_Mode_Offline() {
  Mode = "0";
  String message = "berhasil";
  server.send(200, "text/plain", message);
}

void Server_Halaman_Utama(){
```

```
  String message = "Selamat Datang Alat Keamanan Rumah";
   server.send(200, "text/plain", message);

   }
                    const char* string2char(String Respon_Web_Client_Password){
      if(Respon_Web_Client_Password.length()!=0){
            const char *p =
const_cast<char*>(Respon_Web_Client_Password.c_str());
            return p;}}

void Server_Input_data(){
   String message = "<html>";
     message += "<head><title>Portal Input Data</title>";
      message += "<style>";
        message += "body { background-color: #E6E6FA; font-family: Arial,
Helvetica, Sans-Serif; Color: black;}";
      message += "</style>";
     message += "</head>";
     message += "<body>";
        message += "<form
action='http://"+IPlocal+":"+str_port_web+lokasi_web_input_data+"'
method='POST'>";
        message += "Masukan Password Beranda:<input type='text'
name='password_beranda'> <input type='submit' value='Enter'>";
      message += "</form>";
        message += "<form
action='http://"+IPlocal+":"+str_port_web+lokasi_web_input_data+"'
method='POST'>";
        message += "Masukan Interval Relay Otomatis:<input type='text'
name='interval_relay_otomatis'> <input type='submit' value='Enter'>";
      message += "</form>";
         message += "<form
action='http://"+IPlocal+":"+str_port_web+lokasi_web_input_data+"'
method='POST'>";
        message += "Kondisi Relay:<input type='text'
name='kondisi_relay'> <input type='submit' value='Enter'>";
```

```
      message += "</form>";
            message += "<form
action='http://"+IPlocal+":"+str_port_web+lokasi_web_input_data+"'
method='POST'>";
       message += "Masukan Pintu Beranda:<input type='text'
name='pintu_beranda'> <input type='submit' value='Enter'>";
      message += "</form>";

    message += "</body>";
   message += "</html>";
   server.send(200, "text/html", message);

   if (server.args() > 0 ) {
      for ( uint8_t i = 0; i < server.args(); i++ ) {
         Serial.print(server.argName(i));
         Argumen_Server = server.argName(i);

         if (server.argName(i) == "password_beranda") {
            Serial.print("Masukan datanya adalah: ");
            Serial.println(server.arg(i));
            Respon_Web_Client = server.arg(i);
            Respon_Password_Beranda_Client = Respon_Web_Client.toInt();

int digit_password = hitung_digit(Respon_Password_Beranda_Client);

if (digit_password==6){
      Password_Beranda_Server = Respon_Password_Beranda_Client;
   }
      }

    if (server.argName(i) == "interval_relay_otomatis") {
Serial.print("Masukan datanya adalah: ");
            Serial.println(server.arg(i));
            Respon_Web_Client = server.arg(i);
            Respon_Interval_Relay_Otomatis = Respon_Web_Client.toInt();
   }
```

```
        if (server.argName(i) == "pintu_beranda") {
Serial.print("Masukan datanya adalah: ");
            Serial.println(server.arg(i));
            Respon_Web_Client = server.arg(i);
            Kondisi_Relay_A[1] = Respon_Web_Client.toInt();
   }

        if (server.argName(i) == "kondisi_relay") {
Serial.print("Masukan datanya adalah: ");
            Serial.println(server.arg(i));
            Respon_Web_Client = server.arg(i);

            Serial.println(numMax_Kondisi_Relay);

String Respon_Kondisi_Relay[Respon_Web_Client.length()];

            for (int i=0; i<Respon_Web_Client.length(); i++) {
                 Respon_Kondisi_Relay[i] =      Respon_Web_Client[i];
}

for (int i=0; i<Respon_Web_Client.length(); i++){

   Max_Kondisi_Relay[i] = Respon_Kondisi_Relay[i].toInt();
   }

if(Respon_Web_Client.length() == numMax_Kondisi_Relay ){

              Pisah_Data_Kondisi_Relay();
              }
   }
      }
   }
}
```

```
void Respon_Client(){
   String message;
   message =    "<html>";
    message += "<head><title>Portal Hasil Data</title>";
     message += "<style>";
      message += "body { background-color: #E6E6FA; font-family: Arial,
Helvetica, Sans-Serif; Color: black;}";
     message += "</style>";
    message += "</head>";
    message += "<body>";
     message += "<h1><br>Portal Hasil Data terkahir diterima :</h1>";
           message += "<p>Respon Argumen: " +Argumen_Server+ "</p>";
     message += "<p>Respon diterima: " + Respon_Web_Client + "</p>";
    message += "</body>";
   message += "</html>";
   server.send(200, "text/html", message);



   }

   byte hitung_digit(long long int angka){
   byte count=0;
   while(angka){
      angka=angka/10;
      count++;
   }
   return count;
}

byte digit_ke_nilai(unsigned int angka, int digit) {
      for (int i=0; i<digit-1; i++) {
         angka /= 10;
      }
      return angka % 10;
}
```

```
void nilai_ke_array(int angka){
const byte max_digit = 32;
byte nilai_array[max_digit];

   byte nilai_digit = hitung_digit(angka);
   for (byte i=0; i<nilai_digit; i++) {
        nilai_array[i] = digit_ke_nilai(angka,nilai_digit-i);
   }
}

void Menampilkan_Waktu()
{

   Serial.print(hour());
   Tampil_Digit_Waktu(minute());
   Tampil_Digit_Waktu(second());
   Serial.print(" ");
   Serial.print(day());
   Serial.print(".");
   Serial.print(month());
   Serial.print(".");
   Serial.print(year());
   Serial.println();
}

void Tampil_Digit_Waktu(int digits)
{
   Serial.print(":");
   if (digits < 10)
      Serial.print('0');
   Serial.print(digits);
}


const int NTP_PACKET_SIZE = 48;
  byte packetBuffer[NTP_PACKET_SIZE];
```

```
time_t getNtpTime()
{
   IPAddress ntpServerIP;

   while (Udp_waktu.parsePacket() > 0) ;
   Serial.println("Transmit NTP Request");

   WiFi.hostByName(ntpServerName, ntpServerIP);
   Serial.print(ntpServerName);
   Serial.print(": ");
   Serial.println(ntpServerIP);
   sendNTPpacket(ntpServerIP);
   uint32_t beginWait = millis();
   while (millis() - beginWait < 1500) {
      int size = Udp_waktu.parsePacket();
      if (size >= NTP_PACKET_SIZE) {
         Serial.println("Receive NTP Response");
         Udp_waktu.read(packetBuffer, NTP_PACKET_SIZE);
         unsigned long secsSince1900;
         secsSince1900 =    (unsigned long)packetBuffer[40] << 24;
         secsSince1900 |= (unsigned long)packetBuffer[41] << 16;
         secsSince1900 |= (unsigned long)packetBuffer[42] << 8;
         secsSince1900 |= (unsigned long)packetBuffer[43];
         return secsSince1900 - 2208988800UL + timeZone * SECS_PER_HOUR;
      }
   }
   Serial.println("No NTP Response :-(");
   return 0;
}

void sendNTPpacket(IPAddress &address)
{
   memset(packetBuffer, 0, NTP_PACKET_SIZE);
   packetBuffer[0] = 0b11100011;
   packetBuffer[1] = 0;
   packetBuffer[2] = 6;
   packetBuffer[3] = 0xEC;
```

```
  packetBuffer[12] = 49;
  packetBuffer[13] = 0x4E;
  packetBuffer[14] = 49;
  packetBuffer[15] = 52;
  Udp_waktu.beginPacket(address, 123);
  Udp_waktu.write(packetBuffer, NTP_PACKET_SIZE);
  Udp_waktu.endPacket();
}
```

***Source Code Client 1* :**

```
#include <ESP8266WiFi.h>
#include <ArduinoJson.h>
#include <SimpleTimer.h>
#include <TimeLib.h>
#include <WiFiUdp.h>
#include <Wire.h>

#include <Keypad.h>
#include <Keypad_I2C.h>
#include <Password.h>

#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define OLED_RESET D3
Adafruit_SSD1306 display(OLED_RESET);

#define LOGO16_GLCD_HEIGHT 16
#define LOGO16_GLCD_WIDTH    16
static const unsigned char PROGMEM logo16_glcd_bmp[] =
{ B00000000, B11000000,
   B00000001, B11000000,
   B00000001, B11000000,
   B00000011, B11100000,
   B11110011, B11100000,
   B11111110, B11111000,
   B01111110, B11111111,
   B00110011, B10011111,
   B00011111, B11111100,
   B00001101, B01110000,
   B00011011, B10100000,
   B00111111, B11100000,
   B00111111, B11110000,
   B01111100, B11110000,
   B01110000, B01110000,
```

```
   B00000000, B00110000 };

#define I2CADDR 0x20

const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
   {'1','2','3','A'},
   {'4','5','6','B'},
   {'7','8','9','C'},
   {'*','0','#','D'}
};


byte rowPins[ROWS] = {4, 5, 6, 7};
byte colPins[COLS] = {0, 1, 2, 3};

Keypad_I2C kpd( makeKeymap(keys), rowPins, colPins, ROWS, COLS,
I2CADDR, PCF8574 );

const char* ssid = "ESP8266-RCKD";
const char* password = "4adnanwifi";
const char* host = "192.168.4.1";
const int    Port = 9001;
const byte ClientESP = 1;

int info_Client;

WiFiUDP udp_client_server;
const int udp_server_port = 5555;
const int udp_client_port=5556;
int packetsize = 0;

   IPAddress         Server(192,168,4,1);

SimpleTimer timer0;
SimpleTimer timer1;
```

```
SimpleTimer timer2;
SimpleTimer timer3;

byte waktu_tanda[2];

int Waktu_Server;
int Interval_Lampu_Otomatis[2] = {10,10};
byte interval_otomatis[2] = {10,10};
byte interval_otomatis_num[2];

const byte numRelay = 2;
long Relay[numRelay];
long new_Relay[numRelay];

const byte numKeadaan_Relay = 2;
long Keadaan_Relay[numKeadaan_Relay];

const byte numSensor = 2;
long Sensor[numSensor];

int Password_Client ;
int Password_Server = 123456;
int Password_sync ;

const byte numpass_set = 7;
char pass_set[numpass_set];
byte num_salah_password;

Password password_set = Password(pass_set);

const byte num1 = 200;
char json[num1];

const byte numAck_Server = 2;
byte Ack_Server[numAck_Server];

byte kondisi_pintu;
```

```
byte operasi_pintu;

#define Sensor1 A0
#define sensor_pintu D7
#define selonoid_1 D6
#define buzzer D5
#define motion D0
#define LED D4


float Ro = 490000.0;
int val = 0;
float Vrl = 0.0;
float Rs = 0.0;
float ratio = 0.0;

void setup()
{
      Wire.begin( );
      kpd.begin( makeKeymap(keys) );
      display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
Serial.begin(115200);

   pinMode(selonoid_1,OUTPUT);
   pinMode(buzzer,OUTPUT);
   pinMode(sensor_pintu,INPUT);
   pinMode(motion,INPUT);
   pinMode(LED, OUTPUT);

   digitalWrite(LED, LOW);
   digitalWrite(buzzer, LOW);

Check_Wifi();

     timer0.setInterval(1000L, ClientServer);
//     timer0.setInterval(100L, Data_Server);
     timer1.setInterval(50L, Kontrol);
```

```
    timer1.setInterval(500L, Sensor_lup);
    timer1.setInterval(15000L,Kondisi_Client);
    timer1.setInterval(200L, sync_data);
udp_client_server.begin(udp_server_port);

Display_Awal_Setup();
kpd.addEventListener(Keypad_Kunci); //add an event listener for this keypad
}


void loop()
{
  char key = kpd.getKey();
  Data_Server();
   timer0.run();
  timer1.run();



    }

    void Keypad_Kunci(KeypadEvent key){

      switch (kpd.getState()) {
    case PRESSED:
       switch (key) {

           case 'A':
Cek_Password();
              break;

       case '0':
display.print('*');
display.display();
password_set.append(key);
              break;

       case '1':
```

```
display.print('*');
display.display();
password_set.append(key);
                break;

        case '2':
display.print('*');
display.display();
password_set.append(key);
                break;

        case '3':
display.print('*');
display.display();
password_set.append(key);
                break;

        case '4':
display.print('*');
display.display();
password_set.append(key);
                break;

        case '5':
display.print('*');
display.display();
password_set.append(key);
                break;

        case '6':
display.print('*');
display.display();
password_set.append(key);
                break;

                case '7':
display.print('*');
```

```
display.display();
password_set.append(key);
                break;

case '8':
display.print('*');
display.display();
password_set.append(key);
                break;

                case '9':
display.print('*');
display.display();
password_set.append(key);
                break;

            default:
                {
                   Serial.println(key);
                   break;
                }

          }
          break;

       case RELEASED:
          switch (key) {

                break;
          }
          break;

       case HOLD:
          switch (key) {
             case '*':      ;

             Display_Passsword();
```

```
            Display_Awal();
            break;
          }

          switch (key) {
            case '#':      ;

      digitalWrite(selonoid_1,HIGH);
      operasi_pintu = 1;
      Display_Tutup_Kunci();
      delay(2000);

            Display_Awal();
            break;
          }

          break;

                  }
}

void Cek_Password(){


if (password_set.evaluate()){
num_salah_password = 0;

Display_Valid();
delay(1000);
digitalWrite(selonoid_1,LOW);
operasi_pintu = 0;
Display_Buka_Kunci();
Keadaan_Relay[1] = 1;

display.setCursor(0, 1);
display.println(" ");
```

```
for(byte i = 5; i>0 ; i--){
      display.print(i);
      display.print("..");
      display.display();
delay(1000);
   }
password_set.reset();
Display_Awal();

}
else{
   num_salah_password = num_salah_password +1;
   digitalWrite(selonoid_1,HIGH);
   operasi_pintu = 1;
   display.clearDisplay();
   delay(10);
   display.setTextSize(1.5);
   display.setTextColor(WHITE);
   display.setCursor(0, 0);
   display.println("Password Salah");
   display.display();
   delay(2000);
   Display_Awal();
   password_set.reset();
   }

}

void Display_Valid(){
   display.clearDisplay();
   delay(10);
   display.setTextSize(1.5);
   display.setTextColor(WHITE);
   display.setCursor(0, 0);
   display.println("Berhasil");
     display.println("");
   display.display();
```

```
    }

void Display_Awal_Setup() {

    display.invertDisplay(0);
    display.setRotation(0);
    //display.setRotation(2);
    display.clearDisplay();

    // miniature bitmap display

    display.drawBitmap(30, 16,    logo16_glcd_bmp, 16, 16, WHITE); //(x, y,
array name, sizex, sizey, color(black/white));
    display.display();
    delay(2000);
      Display_Awal();

}

void Display_Awal(){
        display.clearDisplay();
    delay(10);
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(0, 0);
    display.println("Keamanan Rumah");
      display.println("");
    display.display();
    display.setTextSize(1);
    display.println("Masukan Password:");
    display.display();

    }

void Display_Passsword() {
    display.clearDisplay();
    delay(10);
```

```
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(0, 0);
    display.println(pass_set);
    display.println("");
    display.display();
    delay(2000);
    display.display();
}

void Display_Buka_Kunci(){
        display.clearDisplay();
    delay(10);
    display.setTextSize(1.5);
    display.setTextColor(WHITE);
    display.setCursor(0, 0);
    display.println("Buka Kunci");
      display.println("");
    display.display();
    }

void Display_Tutup_Kunci(){
        display.clearDisplay();
    delay(10);
    display.setTextSize(1.5);
    display.setTextColor(WHITE);
    display.setCursor(0, 0);
    display.println("Tutup Kunci");
      display.println("");
    display.display();
    }

void Check_Wifi(){

WiFi.disconnect();
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
```

```
delay(4000);

      if (WiFi.status() == WL_CONNECTED)
   {
      Serial.println("Terkoneksi Wifi");
      digitalWrite(LED, HIGH);
      Serial.println("Menghubungkan ke          : " + String(WiFi.SSID()));
      Serial.println("Kekuatan Sinyal           : " + String(WiFi.RSSI()) + " dBm");
      Serial.print    ("Server IP Address : ");
      Serial.println(Server);
      Serial.print    ("Device IP Address : ");
      Serial.println(WiFi.localIP());
         }
}

         void Kondisi_Client(){

          if (WiFi.status() != WL_CONNECTED)
   {
      digitalWrite(LED, LOW);
      Check_Wifi();
   }

    }

      void ClientServer()
   {
if(Ack_Server[0] == 1){
 udp_client_server.beginPacket(host,udp_client_port);

      Serial.print("Terkoneksi ke Server, Client=");
      Serial.println(ClientESP);

 }
```

```
void DataServer(){

        Serial.print("Relay:");
            Serial.print(Relay[0]);
            Serial.print("-");
            Serial.print(Relay[1]);

            Serial.print("Waktu:");
            Serial.println(Waktu_Server);

            Serial.print("Lampu_Otomatis");
            Serial.print(Interval_Lampu_Otomatis[0]);
            Serial.print("-");
            Serial.println(Interval_Lampu_Otomatis[1]);
            processSyncMessage();
            }
            }



void Kontrol(){

   loop_kunci_pintu();
   motion_loop();

if(digitalRead(sensor_pintu) == HIGH){
   kondisi_pintu = 1;
   }
   else{
      kondisi_pintu = 0;
      }

Keadaan_Relay[1] = loop_sensor_pintu();

if(((Sensor[0] == 1) && ( Relay[0] == 1) )|| (num_salah_password >=3)){
   digitalWrite(buzzer, HIGH);
```

```
   Keadaan_Relay[0] = 1;
   }

   else if((Relay[0] == 0) && (num_salah_password < 3)){
      digitalWrite(buzzer, LOW);
      Keadaan_Relay[0] = 0;
      }

     Serial.println(kondisi_pintu);
  Serial.println(operasi_pintu);

                    }

long loop_sensor_pintu(){

long pintu_keadaan ;

      if ((kondisi_pintu == 1)&&(operasi_pintu == 1)){
return pintu_keadaan = 3;
   }

      else if ((kondisi_pintu == 0)&&(operasi_pintu == 1)){

return    pintu_keadaan = 1;
      }

         else if ((kondisi_pintu == 1)&&(operasi_pintu == 0)){

return    pintu_keadaan = 2;
      }

   else if ((kondisi_pintu == 0)&&(operasi_pintu == 0)){

return    pintu_keadaan = 0;
      }

      }
```

```
void loop_kunci_pintu(){
   if((Relay[1] == 0)&&(new_Relay[1] != Relay[1])){
   new_Relay[1] = Relay[1];
   digitalWrite(selonoid_1,LOW);
   operasi_pintu = 0;
   Display_Buka_Kunci();
   delay(2000);
   Display_Awal();
   }

   else if ((Relay[1] == 1)&&(new_Relay[1] != Relay[1])){
      new_Relay[1] = Relay[1];
      digitalWrite(selonoid_1,HIGH);
      operasi_pintu = 1;
      Display_Tutup_Kunci();
      delay(2000);
      Display_Awal();
      }

   }

void motion_loop(){

         if (digitalRead(motion) == HIGH){
         Sensor[0] = 1;

         }
         else {
              Sensor[0] = 0;
            }
   }


void sync_data(){
   Password_sync = Password_Client;
```

```
int digit_password = hitung_digit(Password_Server);

if(digit_password == 6){


   if (Password_Client!= Password_Server){

      Password_Client = Password_Server;

String str;
str = String(Password_Client);
str.toCharArray(pass_set,numpass_set);

      password_set.set(pass_set);
      }
   Serial.println(Password_sync);
   }
}

   void Menampilkan_Waktu()
{

   Serial.print(hour());
   Tampil_Digit_Waktu(minute());
   Tampil_Digit_Waktu(second());
   Serial.print(" ");
   Serial.print(day());
   Serial.print(".");
   Serial.print(month());
   Serial.print(".");
   Serial.print(year());
   Serial.println();
}

void Tampil_Digit_Waktu(int digits)
{
   Serial.print(":");
```

```
    if (digits < 10)
      Serial.print('0');
    Serial.print(digits);
}

    byte hitung_digit(long long int angka){
    byte count=0;
    while(angka){
      angka=angka/10;
      count++;
    }
    return count;
}

byte digit_ke_nilai(unsigned int angka, int digit) {
      for (int i=0; i<digit-1; i++) {
        angka /= 10;
      }
      return angka % 10;
}

void nilai_ke_array(int angka){
const byte max_digit = 32;
byte nilai_array[max_digit];

    byte nilai_digit = hitung_digit(angka);
    for (byte i=0; i<nilai_digit; i++) {
        nilai_array[i] = digit_ke_nilai(angka,nilai_digit-i);
    }
}
```

***Source Code Client 2* :**

```
#include <ESP8266WiFi.h>
#include <ArduinoJson.h>
#include <SimpleTimer.h>
#include <TimeLib.h>
#include <WiFiUdp.h>

const char* ssid = "ESP8266-RCKD";
const char* password = "4adnanwifi";
const char* host = "192.168.4.1";
const int    Port = 9001;
const byte ClientESP = 2;

int info_Client;

WiFiUDP udp_client_server;
const int udp_server_port = 5555;
const int udp_client_port=5556;
int packetsize = 0;

   IPAddress          Server(192,168,4,1);

SimpleTimer timer;
SimpleTimer timer2;
SimpleTimer timer3;

byte waktu_tanda[2];

int Waktu_Server;
const byte numInterval_Lampu = 3;
int Interval_Lampu_Otomatis[numInterval_Lampu] = {10,10,10};
byte interval_otomatis[numInterval_Lampu] = {10,10,10};
byte interval_otomatis_num[numInterval_Lampu];

const byte numRelay = 3;
long Relay[numRelay];
```

```
long new_Relay[numRelay];
const byte numSensor = 3;
long Sensor[numSensor];


const byte numKeadaan_Relay = 3;
long Keadaan_Relay[numKeadaan_Relay];

const byte num1 = 128;
char json[num1];

const byte numAck_Server = 2;
byte Ack_Server[numAck_Server];

#define lampu_1 D5
#define lampu_2 D6
#define buzzer D7
#define motion_1 D0
#define LED D4

void setup()
{

Serial.begin(115200);
   pinMode(lampu_1,OUTPUT);
   pinMode(lampu_2,OUTPUT);
   pinMode(buzzer,OUTPUT);
   pinMode(motion_1,INPUT);
   pinMode(LED, OUTPUT);

   digitalWrite(LED, LOW);
   digitalWrite(lampu_1, LOW);
   digitalWrite(lampu_2, LOW);
   digitalWrite(buzzer, LOW);
```

```
Check_Wifi();

    timer.setInterval(1000L, ClientServer);
    timer.setInterval(50L, Kontrol);
    timer.setInterval(15000L,Kondisi_Client);
     timer2.setInterval(2000L, switch_Lampu_Kedip_0);
    timer2.setInterval(3000L, switch_Lampu_Otomatis_0);
    timer3.setInterval(3000L, switch_Lampu_Otomatis_1);
    udp_client_server.begin(udp_server_port);
}


void loop()
{
   Data_Server();
timer.run();
 Lampu_Kedip();
      }

void Check_Wifi(){

WiFi.disconnect();
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
delay(4000);

      if (WiFi.status() == WL_CONNECTED)
   {
      Serial.println("Terkoneksi Wifi");
      digitalWrite(LED, HIGH);
      Serial.println("Menghubungkan ke          : " + String(WiFi.SSID()));
      Serial.println("Kekuatan Sinyal           : " + String(WiFi.RSSI()) + " dBm");
      Serial.print    ("Server IP Address : ");
      Serial.println(Server);
      Serial.print    ("Device IP Address : ");
      Serial.println(WiFi.localIP());
         }
```

```
}

          void Kondisi_Client(){

            if (WiFi.status() != WL_CONNECTED)
   {
      digitalWrite(LED, LOW);
      Check_Wifi();
   }

     }

      void ClientServer()
   {
if(Ack_Server[0] == 1){

      Serial.print("Terkoneksi ke Server, Client=");
      Serial.println(ClientESP);

      }

void Data_Server(){

   char message = udp_client_server.parsePacket();
      packetsize = udp_client_server.available();
      if (message)
      {

        udp_client_server.read(json,packetsize);
            DataServer();
            memset(json, 0, sizeof(json));
            Ack_Server[0] = 1;
         }

   }
```

```
void DataServer(){
                Serial.print("Relay");
                Serial.print(Relay[0]);
                Serial.print("-");
                Serial.print(Relay[1]);
                Serial.print("-");
                Serial.print(Relay[2]);

                Serial.print("Waktu");
                Serial.print(Waktu_Server);

                processSyncMessage();
                }
                }

void Kontrol(){

  interval_otomatis[0] =    Interval_Lampu_Otomatis[0] ;
  interval_otomatis[1] =    Interval_Lampu_Otomatis[1] ;
  interval_otomatis[2] =    Interval_Lampu_Otomatis[2] ;
Menampilkan_Waktu();

  if((Relay[0] == 0) || (Relay[0] == 1)){

  if (Relay[0] == 0){
digitalWrite(lampu_1, LOW);
Keadaan_Relay[0] = 0;
}

else if (Relay[0] == 1){
    digitalWrite(lampu_1, HIGH);
    Keadaan_Relay[0] = 1;

    }
    }

    else if(Relay[0] > 1){
```

```
        Lampu_Otomatis_loop_0();

        }

                if((Relay[1] == 0) || (Relay[1] == 1)){

  if (Relay[1] == 0){
digitalWrite(lampu_2, LOW);
Keadaan_Relay[1] = 0;
}

else if (Relay[1] == 1){
    digitalWrite(lampu_2, HIGH);
      Keadaan_Relay[1] = 1;

    }
    }

    else if(Relay[1] > 1){
        Lampu_Otomatis_loop_1();

        }

motion_loop();



if((Sensor[0] == 1) && ( Relay[2] == 1)){
    digitalWrite(buzzer, HIGH);
    Keadaan_Relay[2] = 1;
    }

    else if(Relay[2] == 0){
        digitalWrite(buzzer, LOW);
        Keadaan_Relay[2] = 0;
        }
```

```
                    }

void motion_loop(){

                        if (digitalRead(motion_1) == HIGH){
          Sensor[0] = 1;

          }
          else {
               Sensor[0] = 0;
             }
    }



void Lampu_Kedip(){

if(Keadaan_Relay[0] == 2){

        timer2.run();
        }


    if(Keadaan_Relay[1] == 3){
         timer3.disable(0);
        timer3.restartTimer(0);

        }else    if(Keadaan_Relay[1] == 2){

      timer3.enable(0);
      timer3.run();
        }


    }

void switch_Lampu_Kedip_0(){
```

```
   int count ;
   if(Keadaan_Relay[0]==2){
   if ((count%2) == 0){
      digitalWrite(lampu_1, HIGH);
      Serial.println("HIGH");
      }
      else{
         digitalWrite(lampu_1, LOW);
         Serial.println("LOW");
         count = 0;
         }
         }
    count++;
   }

void Lampu_Otomatis_loop_0(){


   Serial.print("interval_otomatis_num[0]:");
   Serial.println(interval_otomatis_num[0]);

   if (18 <= hour() < 6){


            if(waktu_tanda[0] == 0){
      digitalWrite(lampu_1, HIGH);
      interval_otomatis_num[0] = minute();
      waktu_tanda[0] =1;
}


                  if(interval_otomatis_num[0] >= 60){
      interval_otomatis_num[0] = interval_otomatis_num[0]%60;
      }

            if((minute()) == interval_otomatis_num[0]){
```

```
        interval_otomatis_num[0] = interval_otomatis_num[0] +
interval_otomatis[0];

digitalWrite(lampu_1, LOW);
Keadaan_Relay[0] = 2;

timer2.restartTimer(0);



        }
    }
    }



    void Lampu_Otomatis_loop_1(){
    Serial.print("interval_otomatis_num[1]");
    Serial.println(interval_otomatis_num[1]);



    if (18 <= hour() < 6){

if(waktu_tanda[1] == 0){
        interval_otomatis_num[1] = minute();
        waktu_tanda[1] =1;
}
                    if(interval_otomatis_num[1] > 60){
           interval_otomatis_num[1] = interval_otomatis_num[1]%60;
        }

        if((minute()) == interval_otomatis_num[1]){
        interval_otomatis_num[1] = interval_otomatis_num[1] +
interval_otomatis[1];

        digitalWrite(lampu_2, LOW);
        Keadaan_Relay[1] = 2;
```

```
        }
        }
    }

void switch_Lampu_Otomatis_0(){

    digitalWrite(lampu_1, HIGH);
        Keadaan_Relay[0] = 3;

    }

void switch_Lampu_Otomatis_1(){
    digitalWrite(lampu_2, HIGH);
        Keadaan_Relay[1] = 3;
    }


    void Menampilkan_Waktu()
{

    Serial.print(hour());
    Tampil_Digit_Waktu(minute());
    Tampil_Digit_Waktu(second());
    Serial.print(" ");
    Serial.print(day());
    Serial.print(".");
    Serial.print(month());
    Serial.print(".");
    Serial.print(year());
    Serial.println();
}
```

***Source Code Client 3* :**

```
#include <ESP8266WiFi.h>
#include <ArduinoJson.h>
#include <SimpleTimer.h>
#include <TimeLib.h>
#include <WiFiUdp.h>

const char* ssid = "ESP8266-RCKD";
const char* password = "4adnanwifi";
const char* host = "192.168.4.1";
const int    Port = 9001;
const byte ClientESP = 3;

int info_Client;

WiFiUDP udp_client_server;
const int udp_server_port = 5555;
const int udp_client_port=5556;
int packetsize = 0;

   IPAddress          Server(192,168,4,1);

SimpleTimer timer;

int Waktu_Server;

const byte numRelay = 3;
long Relay[numRelay];
const byte numSensor = 3;
long Sensor[numSensor];

const byte numKeadaan_Relay = 2;
long Keadaan_Relay[numKeadaan_Relay];

const byte num1 = 128;
char json[num1];
```

```
const byte numGagal = 2;
byte Gagal[numGagal];

#define Sensor1 A0

#define Sensor_Gas D3
#define buzzer D5
#define motion D6
#define Blower D0


const byte numAck_Server = 2;
byte Ack_Server[numAck_Server];

void setup()
{

Serial.begin(115200);
   pinMode(buzzer,OUTPUT);
     pinMode(Sensor_Gas,INPUT);
       pinMode(motion,INPUT);
       pinMode(Blower,OUTPUT);
   pinMode(2, OUTPUT);

   digitalWrite(2, LOW);
       digitalWrite(buzzer, LOW);

Check_Wifi();


     timer.setInterval(1000L, ClientServer);
     timer.setInterval(50L, Kontrol);
     timer.setInterval(15000L,Kondisi_Client);
     udp_client_server.begin(udp_server_port);
}
```

```
void loop()
{
   Data_Server();
timer.run();
       }

void Check_Wifi(){

WiFi.disconnect();
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
delay(4000);

       if (WiFi.status() == WL_CONNECTED)
   {
       Serial.println("Terkoneksi Wifi");
       digitalWrite(2, LOW);
       Serial.println("Menghubungkan ke            : " + String(WiFi.SSID()));
       Serial.println("Kekuatan Sinyal             : " + String(WiFi.RSSI()) + " dBm");
       Serial.print    ("Server IP Address : ");
       Serial.println(Server);
       Serial.print    ("Device IP Address : ");
       Serial.println(WiFi.localIP());
          }
}

void Kondisi_Client(){

            if (WiFi.status() != WL_CONNECTED)
   {
       Check_Wifi();
       digitalWrite(2, HIGH);
   }

     }
```

```
void ClientServer()
   {
if(Ack_Server[0] == 1){
      Serial.print("Terkoneksi ke Server, Client=");
      Serial.println(ClientESP);

   }

void Data_Server(){

   char message = udp_client_server.parsePacket();
      packetsize = udp_client_server.available();
      if (message)
      {

       udp_client_server.read(json,packetsize);
       Serial.println(json);
           DataServer();
           memset(json, 0, sizeof(json));
           Ack_Server[0] = 1;
         }


   }

void DataServer(){

              Serial.print("Relay");
              Serial.println(Relay[0]);

                Serial.print("Waktu");
              Serial.println(Waktu_Server);
              processSyncMessage();
              }
}
```

```
void Kontrol_Relay(){
     if (Relay[0] == 0){
digitalWrite(2, LOW);
}

else if (Relay[0] == 1){
   digitalWrite(2, HIGH);

   }

   }

   void Kontrol_Blower(){


   if(digitalRead(Sensor_Gas) == LOW){
      digitalWrite(Blower, LOW);
      Keadaan_Relay[0] = 1;
      Sensor[1] = 1;
      }

         else if (digitalRead(Sensor_Gas) == HIGH) {                    /
      digitalWrite(Blower,    HIGH);
      Keadaan_Relay[0] = 0;
        Sensor[1] = 0;
      }

      }

void Kontrol(){

Menampilkan_Waktu();

  Kontrol_Relay();
Kontrol_Blower();

if (digitalRead(motion) == HIGH){
```

```
        Sensor[0] = 1;

        }
        else if(digitalRead(motion) == LOW){
             Sensor[0] = 0;
           }

if((Sensor[0] == 1) && ( Relay[0] == 1) ||(Sensor[1] == 1) ){
   digitalWrite(buzzer, HIGH);
   Keadaan_Relay[1] = 1;
   }

   else if(Relay[0] == 0){
      digitalWrite(buzzer, LOW);

      Keadaan_Relay[1] = 0;
      }
                  }

   void Menampilkan_Waktu()
{

   Serial.print(hour());
   Tampil_Digit_Waktu(minute());
   Tampil_Digit_Waktu(second());
   Serial.print(" ");
   Serial.print(day());
   Serial.print(".");
   Serial.print(month());
   Serial.print(".");
   Serial.print(year());
   Serial.println();
}
```

***Source Code Client 4* :**

```
#include <ESP8266WiFi.h>
#include <ArduinoJson.h>
#include <SimpleTimer.h>
#include <TimeLib.h>
#include <WiFiUdp.h>

const char* ssid = "ESP8266-RCKD";
const char* password = "4adnanwifi";
const char* host = "192.168.4.1";
const int    Port = 9001;
const byte ClientESP = 4;

int info_Client;

WiFiUDP udp_client_server;
const int udp_server_port = 5555;
const int udp_client_port=5556;
int packetsize = 0;

   IPAddress          Server(192,168,4,1);

SimpleTimer timer;
SimpleTimer timer2;
SimpleTimer timer3;

byte waktu_tanda[2];

int Waktu_Server;
const byte numInterval_Lampu = 3;
int Interval_Lampu_Otomatis[numInterval_Lampu] = {10,10,10};
byte interval_otomatis[numInterval_Lampu] = {10,10,10};
byte interval_otomatis_num[numInterval_Lampu];

const byte numRelay = 3;
long Relay[numRelay];
```

```
const byte numSensor = 3;
long Sensor[numSensor];


const byte numKeadaan_Relay = 3;
long Keadaan_Relay[numKeadaan_Relay];

const byte num1 = 128;
char json[num1];

const byte numAck_Server = 2;
byte Ack_Server[numAck_Server];

#define lampu_1 D5
#define lampu_2 D6
#define buzzer D7
#define motion_1 D4


void setup()
{

Serial.begin(115200);
   pinMode(lampu_1,OUTPUT);
   pinMode(lampu_2,OUTPUT);
   pinMode(buzzer,OUTPUT);
   pinMode(motion_1,INPUT);
   pinMode(2, OUTPUT);

   digitalWrite(2, LOW);
   digitalWrite(lampu_1, LOW);
   digitalWrite(lampu_2, LOW);
   digitalWrite(buzzer, LOW);


Check_Wifi();
```

```
    timer.setInterval(1000L, ClientServer);
  // timer.setInterval(100L, Data_Server);
    timer.setInterval(100L, Kontrol);
    timer.setInterval(15000L,Kondisi_Client);
      timer2.setInterval(2000L, switch_Lampu_Kedip_0);
    timer2.setInterval(3000L, switch_Lampu_Otomatis_0);
    timer3.setInterval(3000L, switch_Lampu_Otomatis_1);
    udp_client_server.begin(udp_server_port);
}


void loop()
{
   Data_Server();
timer.run();
  Lampu_Kedip();
      }

void Check_Wifi(){

WiFi.disconnect();
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
delay(4000);

      if (WiFi.status() == WL_CONNECTED)
   {
      Serial.println("Terkoneksi Wifi");
      digitalWrite(2, HIGH);
      Serial.println("Menghubungkan ke          : " + String(WiFi.SSID()));
      Serial.println("Kekuatan Sinyal           : " + String(WiFi.RSSI()) + " dBm");
      Serial.print    ("Server IP Address : ");
      Serial.println(Server);
      Serial.print    ("Device IP Address : ");
      Serial.println(WiFi.localIP());
         }
}
```

```
        void Kondisi_Client(){

          if (WiFi.status() != WL_CONNECTED)
   {
      digitalWrite(2, LOW);
      Check_Wifi();
   }

    }

      void ClientServer()
   {
if(Ack_Server[0] == 1){
 udp_client_server.beginPacket(host,udp_client_port);

      Serial.print("Terkoneksi ke Server, Client=");
      Serial.println(ClientESP);
            }

void Data_Server(){

   char message = udp_client_server.parsePacket();
      packetsize = udp_client_server.available();
      if (message)
      {

        udp_client_server.read(json,packetsize);
            DataServer();
            memset(json, 0, sizeof(json));
            Ack_Server[0] = 1;
         }


   }
```

```
void DataServer(){

                Serial.print("Relay");
                Serial.print(Relay[0]);
                Serial.print("-");
                Serial.print(Relay[1]);
                Serial.print("-");
                Serial.print(Relay[2]);

                Serial.print("Waktu");
                Serial.print(Waktu_Server);

                processSyncMessage();
                }
                }

void Kontrol(){

  interval_otomatis[0] =    Interval_Lampu_Otomatis[0] ;
  interval_otomatis[1] =    Interval_Lampu_Otomatis[1] ;
  interval_otomatis[2] =    Interval_Lampu_Otomatis[2] ;
Menampilkan_Waktu();

  if((Relay[0] == 0) || (Relay[0] == 1)){

  if (Relay[0] == 0){
digitalWrite(lampu_1, HIGH);
Keadaan_Relay[0] = 0;
}

else if (Relay[0] == 1){
    digitalWrite(lampu_1, LOW);
    Keadaan_Relay[0] = 1;

    }
    }
```

```
    else if(Relay[0] > 1){
       Lampu_Otomatis_loop_0();

       }

                if((Relay[1] == 0) || (Relay[1] == 1)){

  if (Relay[1] == 0){
digitalWrite(lampu_2, HIGH);
Keadaan_Relay[1] = 0;
}

else if (Relay[1] == 1){
    digitalWrite(lampu_2, LOW);
      Keadaan_Relay[1] = 1;

    }
    }

    else if(Relay[1] > 1){
       Lampu_Otomatis_loop_1();

       }

    else if(Relay[2] > 1){
       Lampu_Otomatis_loop_2();

       }
motion_loop();

if((Sensor[0] == 1) && ( Relay[3] == 1)){
    digitalWrite(buzzer, HIGH);
    Keadaan_Relay[3] = 1;
    }

    else if(Relay[3] == 0){
       digitalWrite(buzzer, LOW);
```

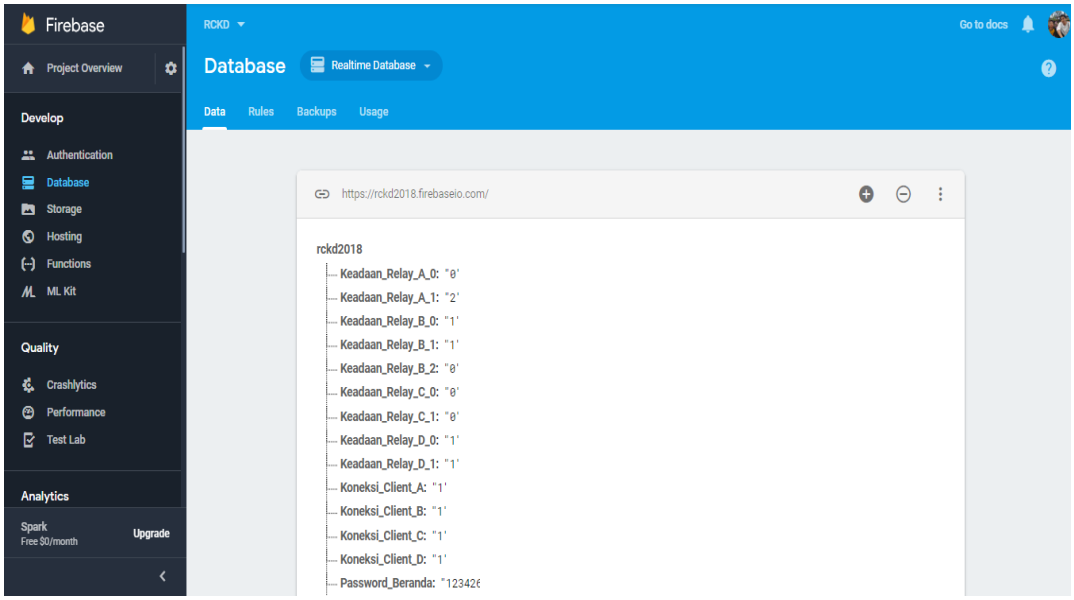
```
        Keadaan_Relay[3] = 0;
        }

                    }

void motion_loop(){

                        if (digitalRead(motion_1) == HIGH){
          Sensor[0] = 1;

          }
          else {
               Sensor[0] = 0;
             }
   }

void Lampu_Kedip(){

if(Keadaan_Relay[0] == 2){
        timer2.run();
        }

void switch_Lampu_Kedip_0(){
   int count ;
   if(Keadaan_Relay[0]==2){
   if ((count%2) == 0){
      digitalWrite(lampu_1, HIGH);
      Serial.println("HIGH");
      }
      else{
         digitalWrite(lampu_1, LOW);
         Serial.println("LOW");
         count = 0;
         }
         }
     count++;
   }
```

```
void Lampu_Otomatis_loop_0(){


   Serial.print("interval_otomatis_num[0]:");
   Serial.println(interval_otomatis_num[0]);

   if (18 <= hour() < 6){


            if(waktu_tanda[0] == 0){
        digitalWrite(lampu_1, HIGH);
      interval_otomatis_num[0] = minute();
      waktu_tanda[0] =1;
}


                  if(interval_otomatis_num[0] >= 60){
         interval_otomatis_num[0] = interval_otomatis_num[0]%60;
      }

            if((minute()) == interval_otomatis_num[0]){
      interval_otomatis_num[0] = interval_otomatis_num[0] +
interval_otomatis[0];

digitalWrite(lampu_1, LOW);
Keadaan_Relay[0] = 2;

timer2.restartTimer(0);
      }

   void Lampu_Otomatis_loop_1(){
   Serial.print("interval_otomatis_num[1]");
   Serial.println(interval_otomatis_num[1]);

   if (18 <= hour() < 6){

if(waktu_tanda[1] == 0){
```

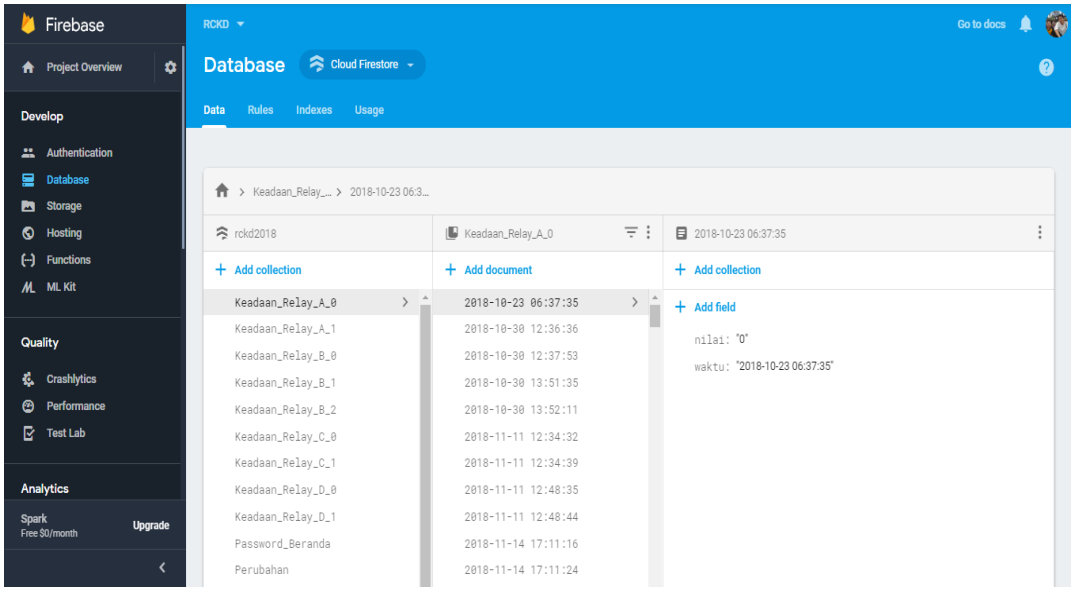
```
        interval_otomatis_num[1] = minute();
        waktu_tanda[1] =1;
}
                    if(interval_otomatis_num[1] > 60){
           interval_otomatis_num[1] = interval_otomatis_num[1]%60;
        }

        if((minute()) == interval_otomatis_num[1]){
        interval_otomatis_num[1] = interval_otomatis_num[1] +
interval_otomatis[1];

    //    digitalWrite(2, LOW);
        Keadaan_Relay[1] = 2;

        }
        }
    }

    void Lampu_Otomatis_loop_2(){
    Serial.print("interval_otomatis_num[1]");
    Serial.println(interval_otomatis_num[2]);



    if (18 <= hour() < 6){

if(waktu_tanda[2] == 0){
        interval_otomatis_num[2] = minute();
        waktu_tanda[2] =1;
}
                    if(interval_otomatis_num[2] > 60){
           interval_otomatis_num[2] = interval_otomatis_num[2]%60;
        }

        if((minute()) == interval_otomatis_num[1]){
        interval_otomatis_num[2] = interval_otomatis_num[2] +
interval_otomatis[2];
```

```
    //    digitalWrite(2, LOW);
        Keadaan_Relay[2] = 2;

        }
        }
    }

void switch_Lampu_Otomatis_0(){

    digitalWrite(lampu_1, HIGH);
        Keadaan_Relay[0] = 3;

    }

void switch_Lampu_Otomatis_1(){
  // digitalWrite(2, HIGH);
        Keadaan_Relay[1] = 3;
    }

    void switch_Lampu_Otomatis_2(){
  // digitalWrite(2, HIGH);
        Keadaan_Relay[2] = 3;
    }
```

Gambar Tampilan Firebase pada Realtime Database:



Gambar Tampilan Firebase pada Cloud Firestore:



Gambar Alat Server:

Gambar Alat Client 3 pada Blower:

Gambar penampakan dalam alat: