LAMPIRAN

Lampiran 1 Source Code

sketch_jan13a

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266HTTPClient.h>
#include <Wire.h>
#include <ArduinoJson.h>

#define kipas1 12
#define kipas2 14

const char* ssid     = "FREEFIRE 1";
const char* password = "sempakmamel";
String incomingByte;
long kipas_1;
long kipas_2;
long modek;
char tampung[3];
int a,b,c,d;
char tmp[120];

WiFiClient client;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(kipas1,OUTPUT);
  pinMode(kipas2,OUTPUT);


        Serial.println("Connecting to ");
        Serial.println(ssid);
```

```
Invalid library found in C:\Users\Adi Nurdiansyah Y\Documents\A
Invalid library found in C:\Users\Adi Nurdiansyah Y\Documents\A
```

sketch_jan13a

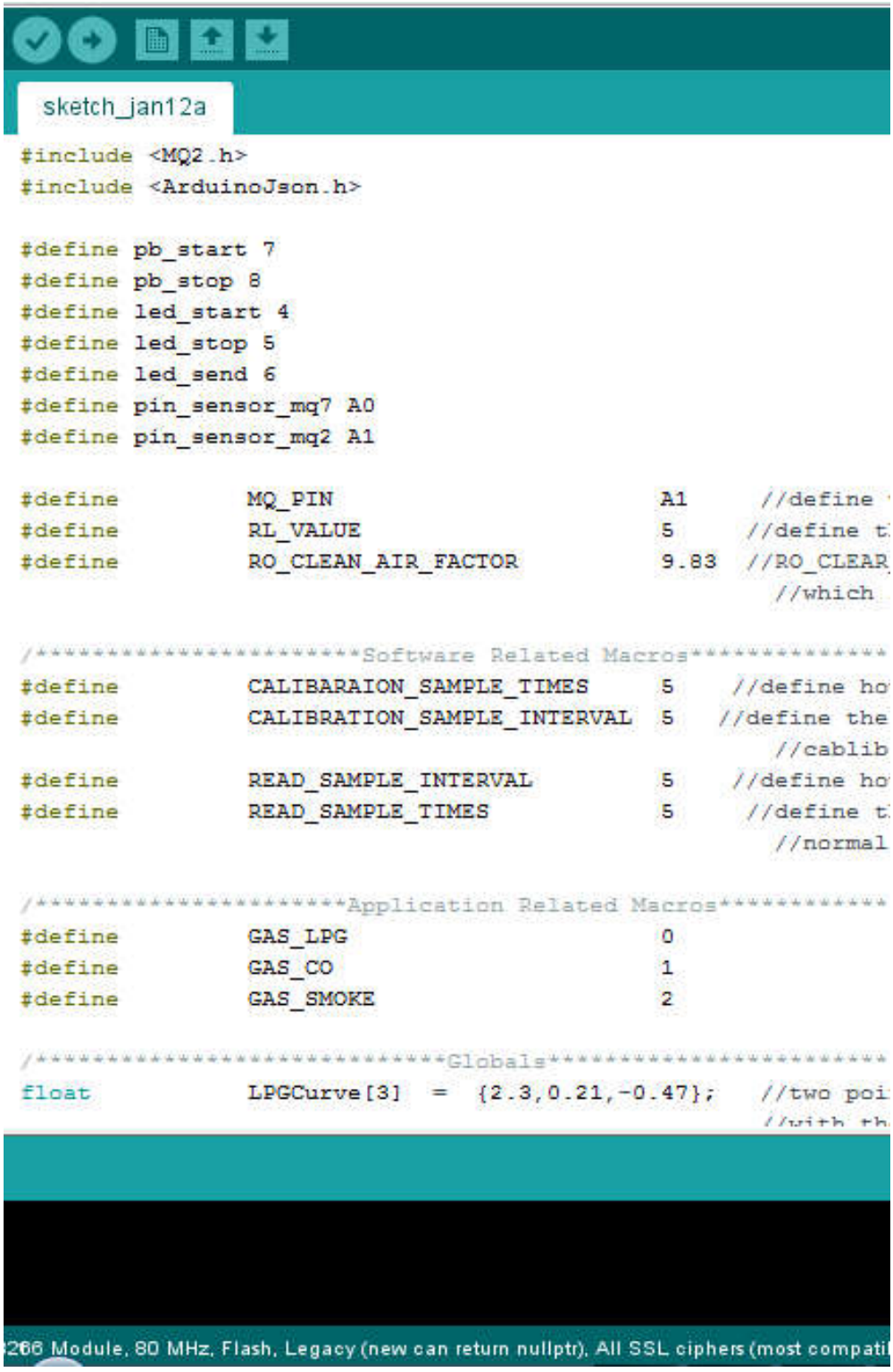
```
        Serial.println("Connecting to ");
        Serial.println(ssid);

        WiFi.begin(ssid, password);
        while (WiFi.status() != WL_CONNECTED)
            {
              delay(500);
              Serial.print(".");
            }
       Serial.println("");
       Serial.println("WiFi connected");

}

void loop() {
    get_kipas();
    //delay(100);
  if (Serial.available() > 0) {
      //read the incoming byte:
      //Serial.print("oke");
      parsing_data();
      if(d==0)
      {
       kirim_server();
      }
      //delay(1000);
  }
}
void get_kipas(){
  HTTPClient http;
```

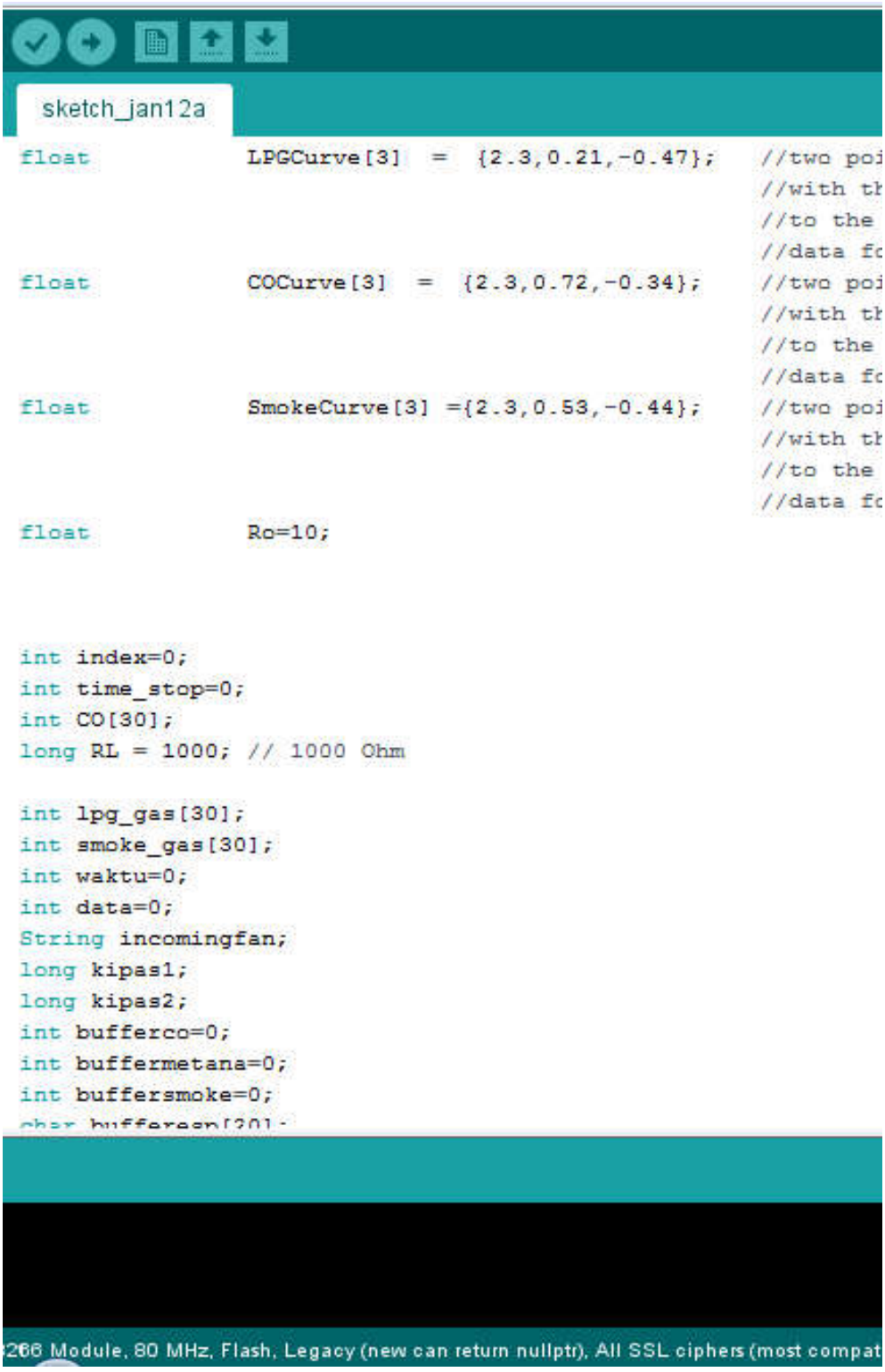
```
Invalid library found in C:\Users\Adi Nurdiansyah Y\Documents
Invalid library found in C:\Users\Adi Nurdiansyah Y\Documents
```

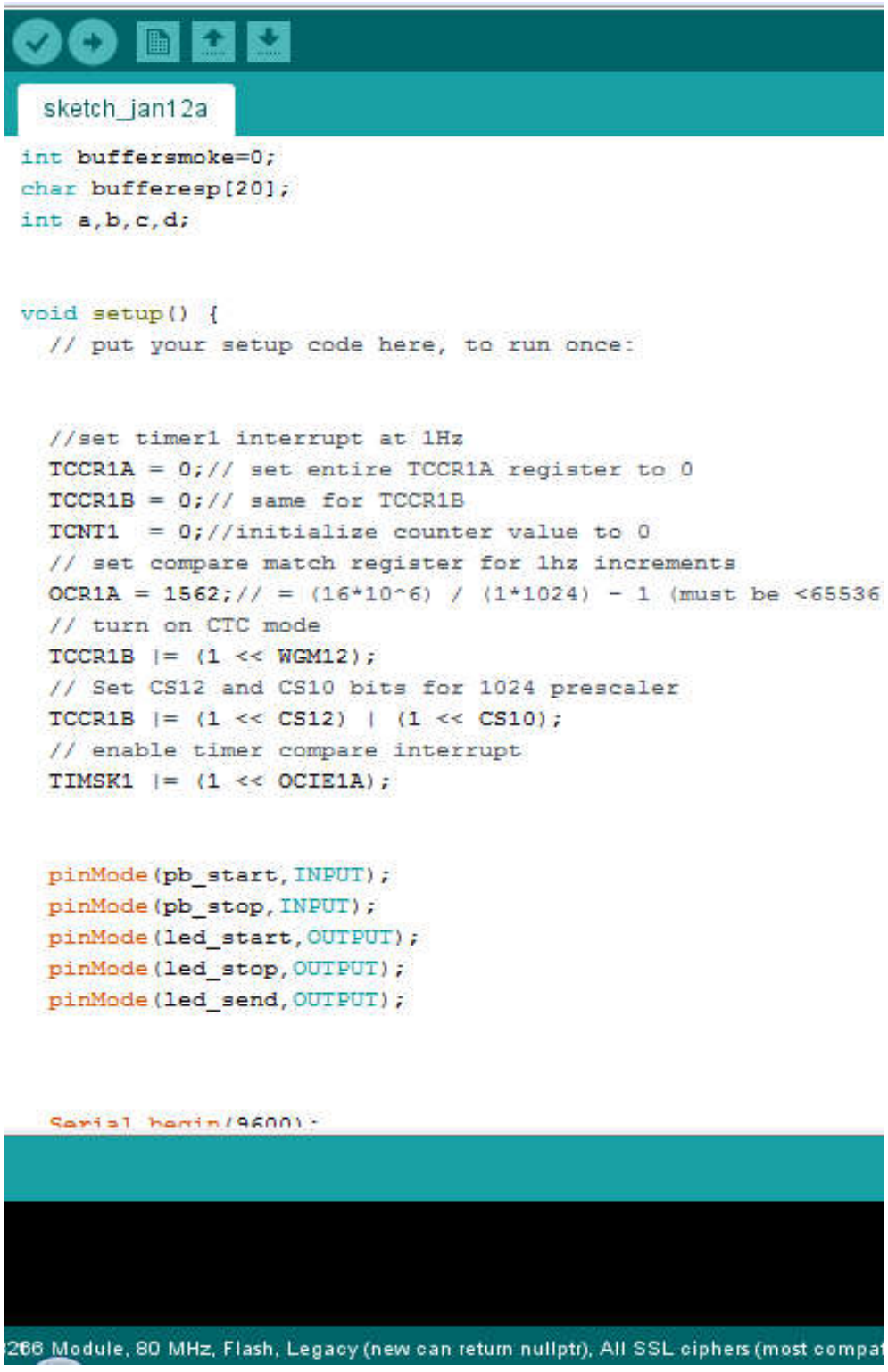

sketch_jan13a
}
}
void get_kipas(){
HTTPClient http;
http.begin("http://projekbareng.com/untag/getkipas.php");
int httpCode = http.GET();
if (httpCode > 0) { //Check the returning code
DynamicJsonDocument doc(2048);
deserializeJson(doc, http.getStream());
kipas_1=doc["kipas1"].as<long>();
kipas_2=doc["kipas2"].as<long>();
modek=doc["mode"].as<long>();
}
if(modek==1)
{
//Serial.print("modek1");
if(kipas_1==1)
{
digitalWrite(kipas1,HIGH);
}
else
{
digitalWrite(kipas1,LOW);
}
if(kipas_2==1)
{
digitalWrite(kipas2,HIGH);
}
else
{
digitalWrite(kipas2,LOW);
Invalid library found in C:\Users\Adi Nurdiansyah Y\Documents
Invalid library found in C:\Users\Adi Nurdiansyah Y\Documents
Module, 80 MHz, Flash, Legacy (new can return nullptr), All SSL ciphers (most compat

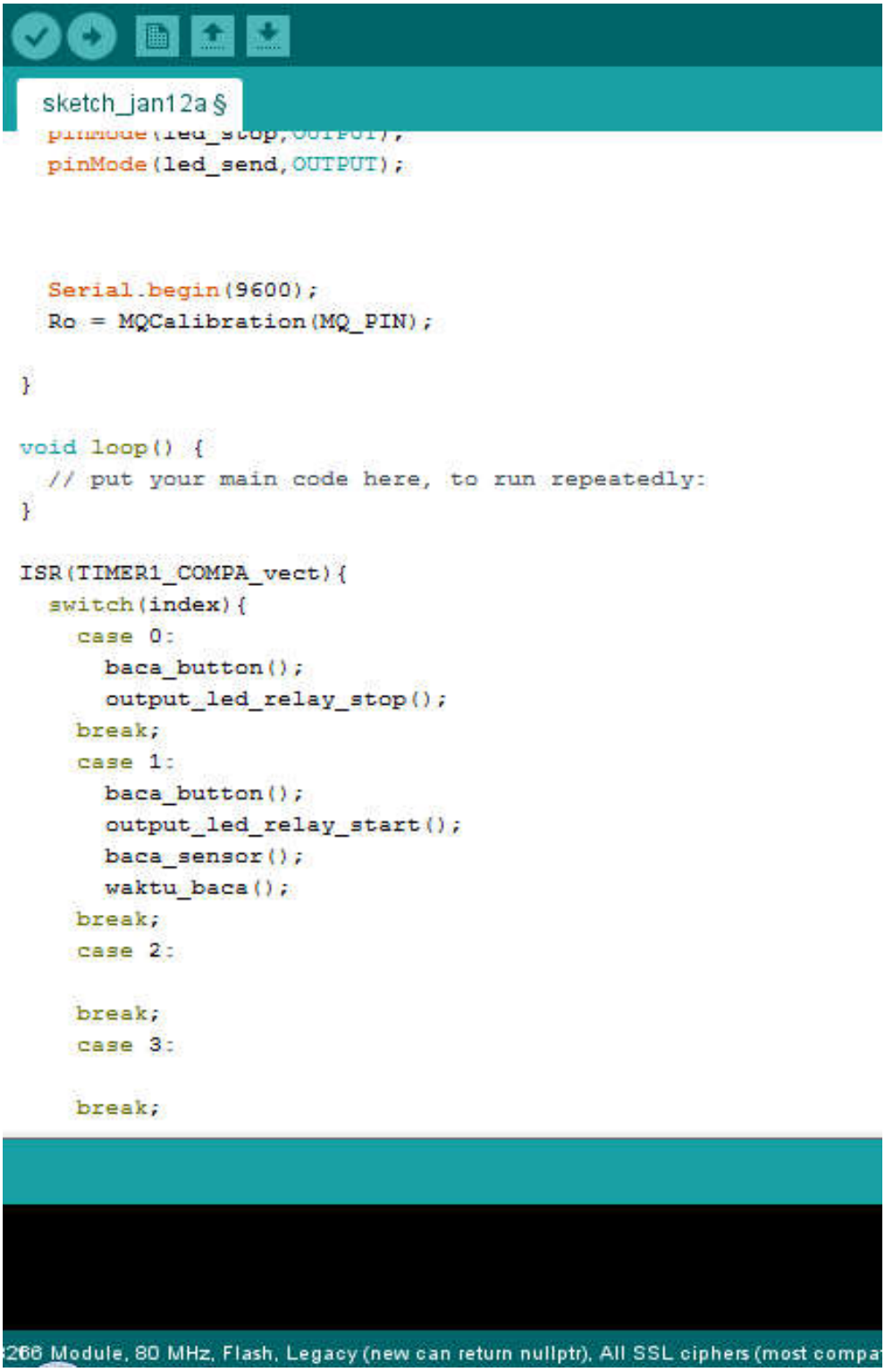

sketch_jan13a
digitalWrite(kipas2,LOW);
}
}
else
{
if(d>0)
{
//Serial.print("mode Stop");
digitalWrite(kipas1,LOW);
digitalWrite(kipas2,LOW);
}
else
{
//Serial.print("mode Start");
if(a>25||b>2||c>300)//a=CO b=metana c=smoke
{
//Serial.print("kipas nyala");
digitalWrite(kipas1,HIGH);
digitalWrite(kipas2,HIGH);
}
else
{
digitalWrite(kipas1,LOW);
digitalWrite(kipas2,LOW);
}
}
}
http.end(); //Close connectio
Invalid library found in C:\Users\Adi Nurdiansyah Y\Document
Invalid library found in C:\Users\Adi Nurdiansyah Y\Document
Module, 80 MHz, Flash, Legacy (new can return nullptr), All SSL ciphers (most compa

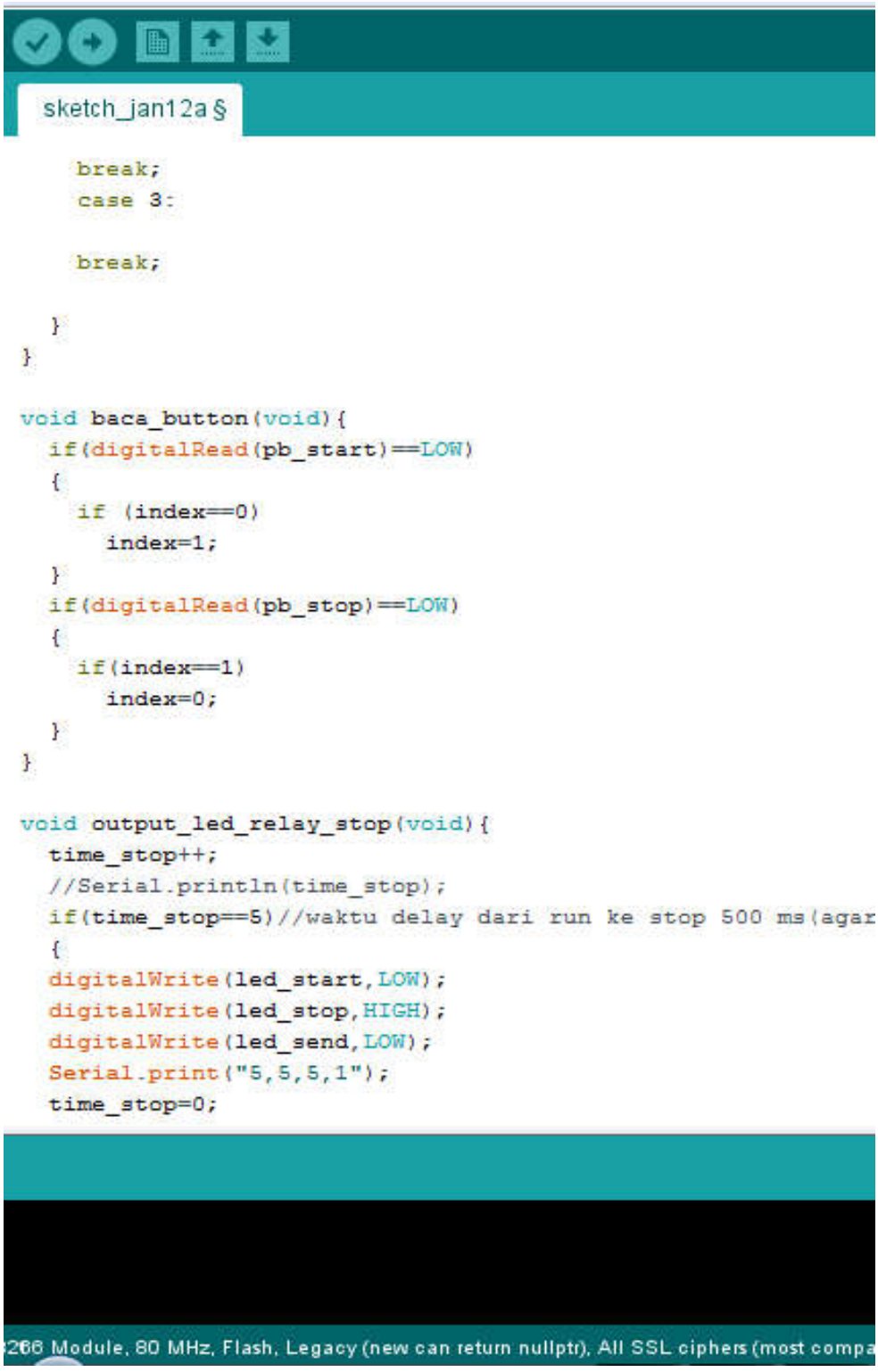

sketch_jan13a
http.end(); //Close connectio
}
void parsing_data(){
a=Serial.parseInt();
b=Serial.parseInt();
c=Serial.parseInt();
d=Serial.parseInt();
//Serial.print(a);
//Serial.print(b);
//Serial.print(c);
//Serial.print(d);
}
void kirim_server(){
if (WiFi.status() == WL_CONNECTED) { //Check WiFi connec
//Serial.print("oke");
HTTPClient http; //Declare an object of class HTTPCli
sprintf(tmp,"http://projekbareng.com/untag/send.php?se
http.begin(tmp);
int httpCode = http.GET();
if (httpCode > 0) { //Check the returning code
String payload = http.getString(); //Get the request
}
http.end();
//Serial.println(tmp);
}
}
Invalid library found in C:\Users\Adi Nurdiansyah Y\Documents
Invalid library found in C:\Users\Adi Nurdiansyah Y\Documents
Module, 80 MHz, Flash, Legacy (new can return nullptr), All SSL ciphers (most compat


sketch_jan12a
#include <MQ2.h>
#include <ArduinoJson.h>
#define pb_start 7
#define pb_stop 8
#define led_start 4
#define led_stop 5
#define led_send 6
#define pin_sensor_mq7 A0
#define pin_sensor_mq2 A1
#define MQ_PIN A1 //define
#define RL_VALUE 5 //define t
#define RO_CLEAN_AIR_FACTOR 9.83 //RO_CLEAR
//which
/************************Software Related Macros***************
#define CALIBARAION_SAMPLE_TIMES 5 //define ho
#define CALIBRATION_SAMPLE_INTERVAL 5 //define the
//cablib
#define READ_SAMPLE_INTERVAL 5 //define ho
#define READ_SAMPLE_TIMES 5 //define t
//normal
/***********************Application Related Macros*************
#define GAS_LPG 0
#define GAS_CO 1
#define GAS_SMOKE 2
/*********************************Globals************************
float LPGCurve[3] = {2.3,0.21,-0.47}; //two poi
266 Module, 80 MHz, Flash, Legacy (new can return nullptr), All SSL ciphers (most compati


sketch_jan12a
float LPGCurve[3] = {2.3,0.21,-0.47}; //two poi
//with th
//to the
//data fo
float COCurve[3] = {2.3,0.72,-0.34}; //two poi
//with th
//to the
//data fo
float SmokeCurve[3] ={2.3,0.53,-0.44}; //two poi
//with th
//to the
//data fo
float Ro=10;
int index=0;
int time_stop=0;
int CO[30];
long RL = 1000; // 1000 Ohm
int lpg_gas[30];
int smoke_gas[30];
int waktu=0;
int data=0;
String incomingfan;
long kipas1;
long kipas2;
int bufferco=0;
int buffermetana=0;
int buffersmoke=0;
266 Module, 80 MHz, Flash, Legacy (new can return nullptr), All SSL ciphers (most compat


sketch_jan12a
int buffersmoke=0;
char bufferesp[20];
int a,b,c,d;
void setup() {
// put your setup code here, to run once:
//set timer1 interrupt at 1Hz
TCCR1A = 0;// set entire TCCR1A register to 0
TCCR1B = 0;// same for TCCR1B
TCNT1 = 0;//initialize counter value to 0
// set compare match register for 1hz increments
OCR1A = 1562;// = (16*10^6) / (1*1024) - 1 (must be <65536
// turn on CTC mode
TCCR1B |= (1 << WGM12);
// Set CS12 and CS10 bits for 1024 prescaler
TCCR1B |= (1 << CS12) | (1 << CS10);
// enable timer compare interrupt
TIMSK1 |= (1 << OCIE1A);
pinMode(pb_start,INPUT);
pinMode(pb_stop,INPUT);
pinMode(led_start,OUTPUT);
pinMode(led_stop,OUTPUT);
pinMode(led_send,OUTPUT);
Serial.begin(9600);
266 Module, 80 MHz, Flash, Legacy (new can return nullptr), All SSL ciphers (most compat


sketch_jan12a §
pinMode(led_send,OUTPUT);
Serial.begin(9600);
Ro = MQCalibration(MQ_PIN);
}
void loop() {
// put your main code here, to run repeatedly:
}
ISR(TIMER1_COMPA_vect){
switch(index){
case 0:
baca_button();
output_led_relay_stop();
break;
case 1:
baca_button();
output_led_relay_start();
baca_sensor();
waktu_baca();
break;
case 2:
break;
case 3:
break;
266 Module, 80 MHz, Flash, Legacy (new can return nullptr), All SSL ciphers (most compa

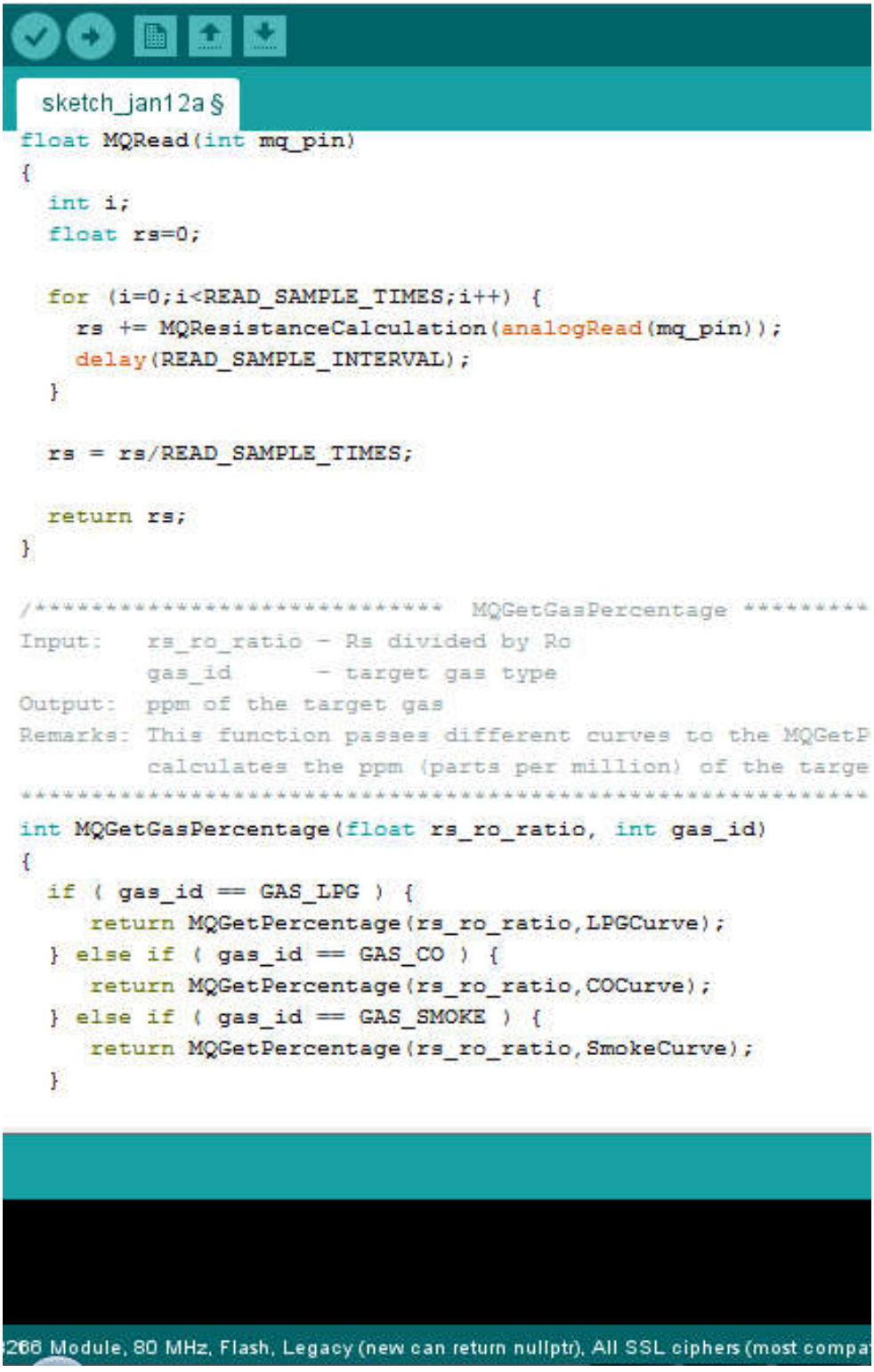

sketch_jan12a §
break;
case 3:
break;
}
}
void baca_button(void){
if(digitalRead(pb_start)==LOW)
{
if (index==0)
index=1;
}
if(digitalRead(pb_stop)==LOW)
{
if(index==1)
index=0;
}
}
void output_led_relay_stop(void){
time_stop++;
//Serial.println(time_stop);
if(time_stop==5)//waktu delay dari run ke stop 500 ms(agar
{
digitalWrite(led_start,LOW);
digitalWrite(led_stop,HIGH);
digitalWrite(led_send,LOW);
Serial.print("5,5,5,1");
time_stop=0;
266 Module, 80 MHz, Flash, Legacy (new can return nullptr), All SSL ciphers (most compa


sketch_jan12a §
time_stop=0;
}
}
void output_led_relay_start(void){
digitalWrite(led_start,HIGH);
digitalWrite(led_stop,LOW);
digitalWrite(led_send,LOW);
}
void baca_sensor(){
//sensor MQ7
int sensorvalue = analogRead(pin_sensor_mq7); // membaca n
float VRL= sensorvalue*5.00/1024; // mengubah nilai ADC (
float Rs = (5.00 * RL / VRL) - RL;
CO[data] = int(100 * pow(Rs / Ro,-1.53)); // ppm = 100 * (
/*Serial.print("CO : ");
Serial.print(CO[data]);
Serial.println(" ppm");*/
//sensor MQ2
//Serial.println(MQ2value);
lpg_gas[data] = MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GAS_
smoke_gas[data] = MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GA
/* Serial.print("Metana : ");
Serial.print(lpg_gas[data]);
266 Module, 80 MHz, Flash, Legacy (new can return nullptr), All SSL ciphers (most compa


sketch_jan12a §
lpg_gas[data] = MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GAS_
smoke_gas[data] = MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GA
/* Serial.print("Metana : ");
Serial.print(lpg_gas[data]);
Serial.println(" ppm");
Serial.print("Smoke : ");
Serial.print(smoke_gas[data]);
Serial.println(" ppm");
Serial.println("");*/
}
void waktu_baca(){
waktu++;
//Serial.println(waktu);
if(waktu==300){//1 menit sekali membaca sensor 300*100=300
waktu=0;
data++;
baca_sensor();
if (data==2){//1 menit sekali kirim ke tabel
for(int i=1;i<=2;i++)
{
bufferco=bufferco+CO[i];
buffermetana=buffermetana+lpg_gas[i];
buffersmoke=buffersmoke+smoke_gas[i];
}//ngejumlah 2 data
bufferco=bufferco/2;//proses mendapatkan rata rata
buffermetana=buffermetana/2;
buffersmoke=buffersmoke/2;
data=0;
kirim_esp();
bufferco=0;
266 Module, 80 MHz, Flash, Legacy (new can return nullptr), All SSL ciphers (most compa


sketch_jan12a §
buffersmoke=0;
}
}
}
void kirim_esp(){
sprintf(bufferesp,"%d,%d,%d,0",bufferco,buffermetana,buffe
Serial.print(bufferesp);
sprintf(bufferesp,"
");
digitalWrite(led_send,HIGH);
}
/******************** MQResistanceCalculation ****************
Input: raw_adc - raw value read from adc, which represents
Output: the calculated sensor resistance
Remarks: The sensor and the load resistor forms a voltage di
across the load resistor and its resistance, the re
could be derived.
****
float MQResistanceCalculation(int raw_adc)
{
return (((float)RL_VALUE*(1023-raw_adc)/raw_adc));
}
/***************************** MQCalibration ****************
Input: mq_pin - analog channel
Output: Ro of the sensor
Remarks: This function assumes that the sensor is in clean a
266 Module, 80 MHz, Flash, Legacy (new can return nullptr), All SSL ciphers (most compa


sketch_jan12a §
Output: Ro of the sensor
Remarks: This function assumes that the sensor is in clean a
MQResistanceCalculation to calculates the sensor re
and then divides it with RO_CLEAN_AIR_FACTOR. RO_CL
10, which differs slightly between different sensor
float MQCalibration(int mq_pin)
{
int i;
float val=0;
for (i=0;i<CALIBARAION_SAMPLE_TIMES;i++) {
val += MQResistanceCalculation(analogRead(mq_pin));
delay(CALIBRATION_SAMPLE_INTERVAL);
}
val = val/CALIBARAION_SAMPLE_TIMES;
val = val/RO_CLEAN_AIR_FACTOR;
return val;
}
MQRead
Input: mq_pin - analog channel
Output: Rs of the sensor
Remarks: This function use MQResistanceCalculation to cacula
The Rs changes as the sensor is in the different co
gas. The sample times and the time interval between
by changing the definition of the macros.
float MQRead(int mq_pin)
{
266 Module, 80 MHz, Flash, Legacy (new can return nullptr), All SSL ciphers (most compa


sketch_jan12a §
float MQRead(int mq_pin)
{
int i;
float rs=0;
for (i=0;i<READ_SAMPLE_TIMES;i++) {
rs += MQResistanceCalculation(analogRead(mq_pin));
delay(READ_SAMPLE_INTERVAL);
}
rs = rs/READ_SAMPLE_TIMES;
return rs;
}
/***************************** MQGetGasPercentage **********
Input: rs_ro_ratio - Rs divided by Ro
gas_id - target gas type
Output: ppm of the target gas
Remarks: This function passes different curves to the MQGetP
calculates the ppm (parts per million) of the targe
****
int MQGetGasPercentage(float rs_ro_ratio, int gas_id)
{
if (gas_id == GAS_LPG) {
return MQGetPercentage(rs_ro_ratio,LPGCurve);
} else if (gas_id == GAS_CO) {
return MQGetPercentage(rs_ro_ratio,COCurve);
} else if (gas_id == GAS_SMOKE) {
return MQGetPercentage(rs_ro_ratio,SmokeCurve);
}
266 Module, 80 MHz, Flash, Legacy (new can return nullptr), All SSL ciphers (most compa


sketch_jan12a §
Input: rs_ro_ratio - Rs divided by Ro
gas_id - target gas type
Output: ppm of the target gas
Remarks: This function passes different curves to the MQGetP
calculates the ppm (parts per million) of the targe
int MQGetGasPercentage(float rs_ro_ratio, int gas_id)
{
if (gas_id == GAS_LPG) {
return MQGetPercentage(rs_ro_ratio,LPGCurve);
} else if (gas_id == GAS_CO) {
return MQGetPercentage(rs_ro_ratio,COCurve);
} else if (gas_id == GAS_SMOKE) {
return MQGetPercentage(rs_ro_ratio,SmokeCurve);
}
return 0;
}
/****************************** MQGetPercentage ************
Input: rs_ro_ratio - Rs divided by Ro
pcurve - pointer to the curve of the target ga
Output: ppm of the target gas
Remarks: By using the slope and a point of the line. The x(l
of the line could be derived if y(rs_ro_ratio) is p
logarithmic coordinate, power of 10 is used to conv
value.
int MQGetPercentage(float rs_ro_ratio, float *pcurve)
{
return (pow(10,(((log(rs_ro_ratio)-pcurve[1])/pcurve[2])
}
266 Module, 80 MHz, Flash, Legacy (new can return nullptr), All SSL ciphers (most compa