LAMPIRAN A

LAMPIRAN

LAMPIRAN 1 Proses perakitan Sensor

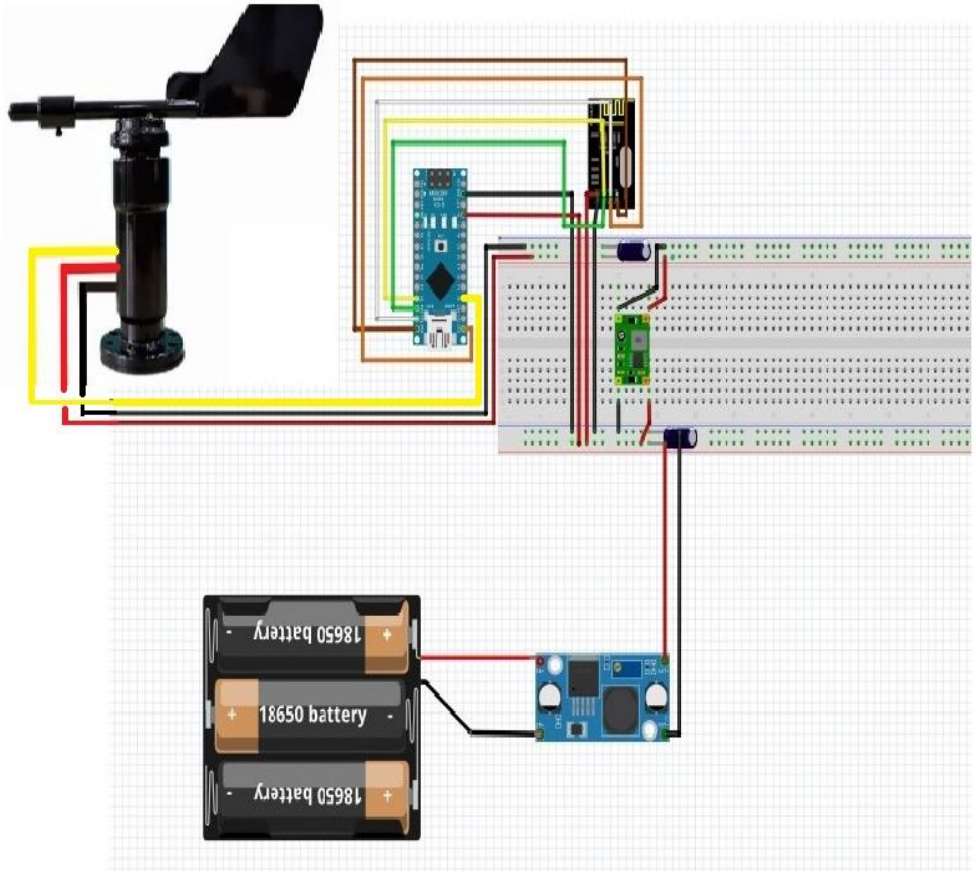
LAMPIRAN 2 Pemotongan Rangka Tiang *Weather Station*

LAMPIRAN 3 Pengelasan Tiang Weather Station

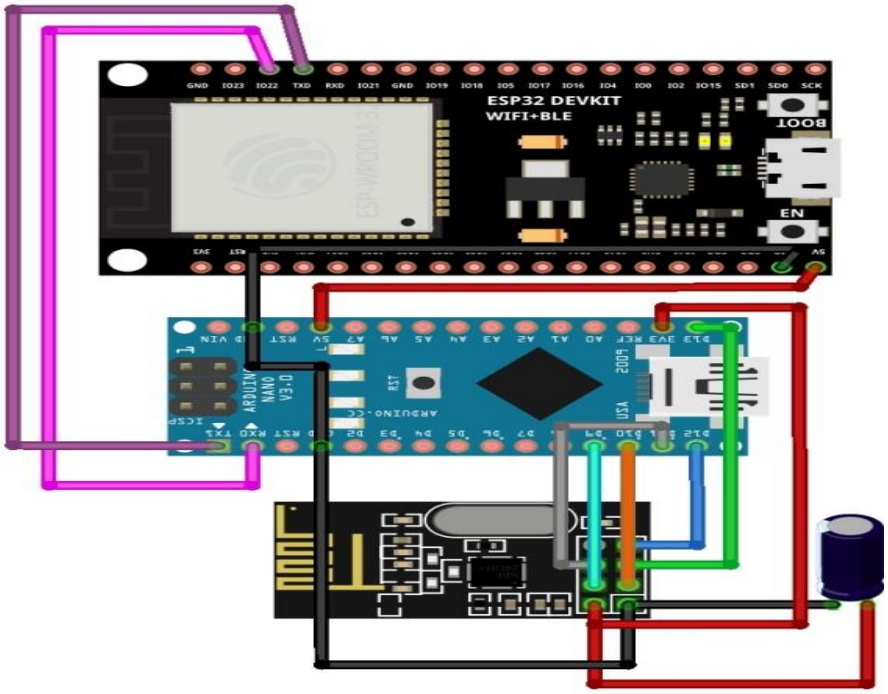
LAMPIRAN 4 Hasil Jadi Weather Station

LAMPIRAN 5 Pengujian Dengan Cuaca *Real*

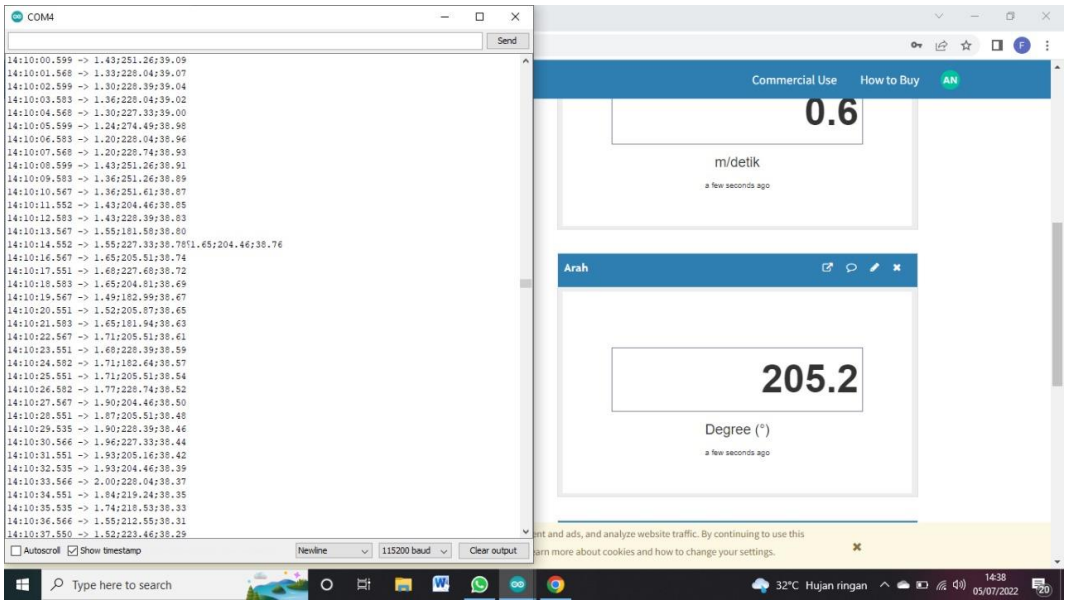
LAMPIRAN 6 Pengujian Jarak *Transmitter* Dan *Receiver*



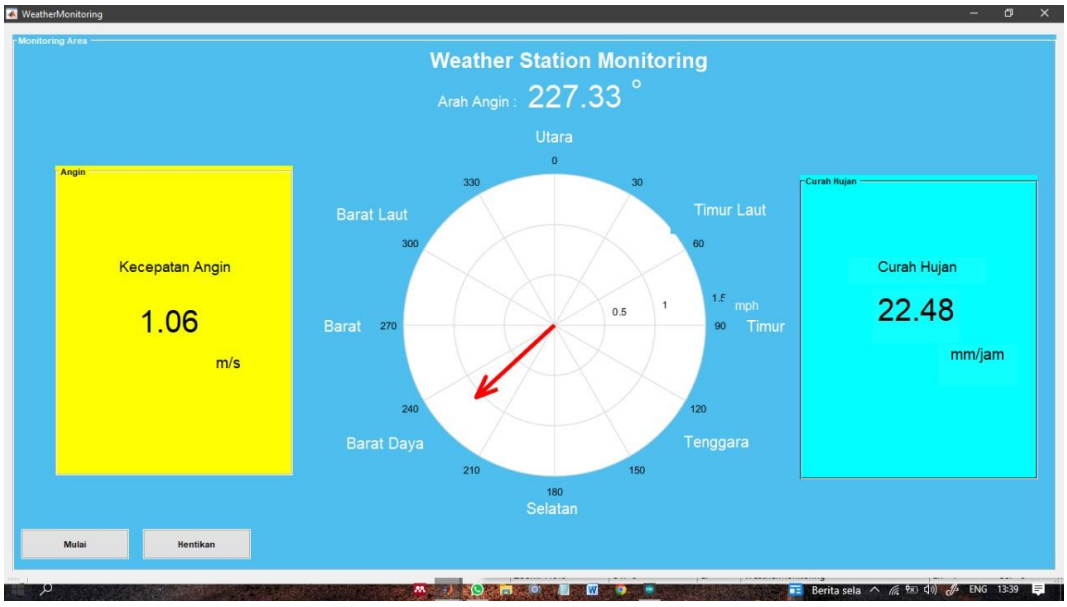
LAMPIRAN 7 Tata Letak Komponen Sensor Dan *Transmitter*



LAMPIRAN 8 Tata Letak Komponen *Receiver*



LAMPIRAN 9 Pembacaan Sensor Pada *Serial Monitor* Dan *Thingspeak*



LAMPIRAN 10 Monitoring Di Software Matlab

LAMPIRAN B

Wind Direction Sensor

OVERVIEW

The MBMet 110 series Wind Direction Sensors are wind direction indicators that provides visual indication of wind direction. Digital circuits are capable of strong RFI and EMI resistance and the automatic temperature compensation are built in. MBMet 110 series construction reflects high reliability and durability providing minimal maintenance. The sensor material design offers superb dirt and weather resistance for long-term measurement stability in dirty and cloudy environments.

BENEFITS AND FEATURES

- **Low starting threshold**
- **Strong corrosion resistant ability**
- **Various Output signal options**
- **Double bearing design**
- **Long-term stability in dirt and dusty environment**
- **Easy Installation**

APPLICATIONS

- **Weather Station**
- **Smart Cities**
- **Solar and Wind Power Generation Plants**
- **Marine Vessels**
- **Airports**
- **Road Management**

Resolution : 22.5°

Resolution : 1°

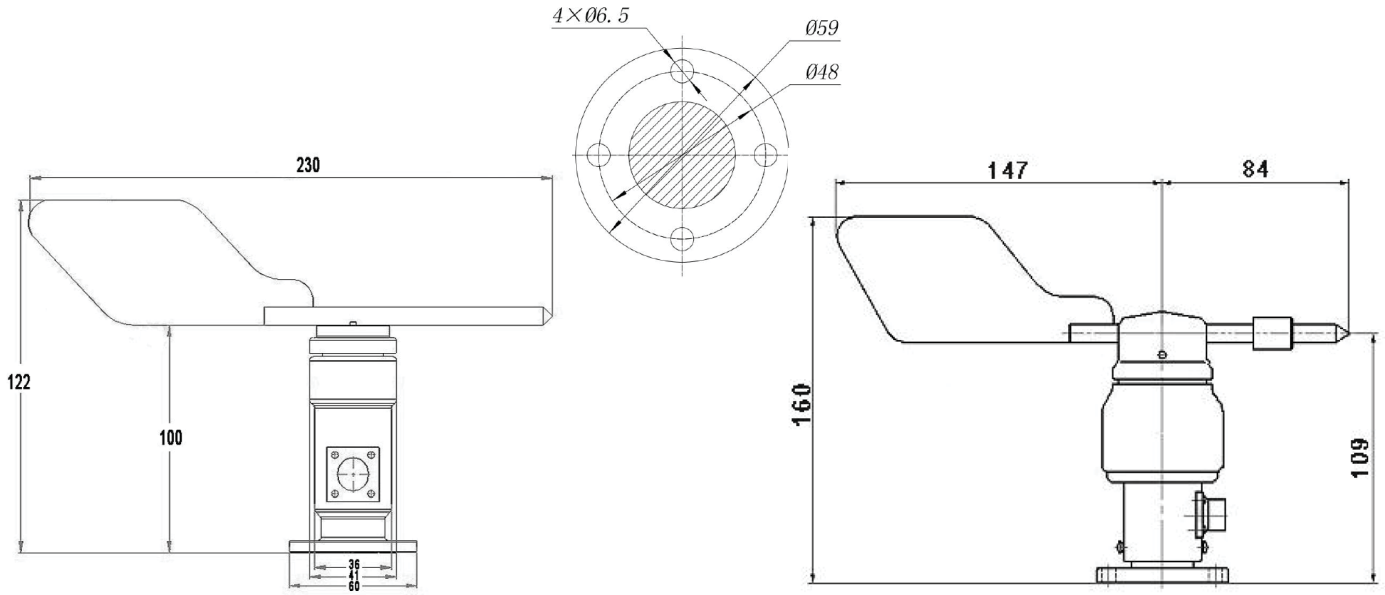
TECHNICAL DESCRIPTION

Model/Parameters	MBMet-110A	MBMet-110A-B	MBMet-110B	MBMet-110B-B	MBMet-110C	MBMet-110C-B	MBMet-110D	MBMet-110D-B
Resolution	A 1°	B 22.5°	A 1°	B 22.5°	A 1°	B 22.5°	A 1°	B 22.5°
Communication Output	4-20mA		RS-485 Modbus		0-5V		0-10V	
Measuring Range	0 to 360°							
Accuracy	±3°							
Measurement Performance								
Starting Wind Speed	<0.5m/s							
Survival Wind Speed	70m/s							
Electrical Specification								
Input Voltage	12 to 24VDC							
General Specifications								
Operating Temperature	-30°C to +70°C							
Ingress Protection Rating	IP65							
Storage Condition	10°C to 60°C @ 20% to 90%RH							
Cable Type/Length	UV protected cable/3 meters							
Body Material	Vane: 304 stainless steel, Main Body: Aluminum alloy							
Weight (unpacked)	410g							

Model	Series	Output	Resolution	Accessories	Cable Length	
MBMet						
	110					
		A				4-20mA
		B				RS-485 Modbus
		C				0 to 5V
		D				0 to 10V
			A			1°
			B			22.5°
				A		With Pole Mounting Accessories
				X		Without Pole Mounting Accessories
					3000	Units : mm (default)
					.....	Units : mm (on request)

DIMENSIONS & MOUNTING

Flange mounted, fix four screws on the bracket and install the product horizontal.



Resolution : 22.5° **Resolution : 1°**

Note : There is one red or white marked point on the sensor, it should be pointed towards north direction while installing.

ORDERING STRING

For example : MBMet-110B-AA3000
Parameter : Wind Direction,
Output : RS-485 Modbus,
Resolution : 1° with Pole Mounting
Accessories, Cable Length : 3 meters

SEE ALSO

- MBMet 100 Wind Direction Sensor
- MBMet 120 Wind Speed & Direction Sensor
- MBMet 140 Ultrasonic Wind Speed & Direction Sensor
- MBMet 800 PV Module Temperature Sensor
- MBMet 901 Air Temperature, Humidity & Pressure Sensor
- View our complete range of Weather Sensors

****Specifications are subject to change without notice.**

LAMPIRAN 5 Datasheet Sensor *Wind Direction*

Parameter Teknis

Jenis keluaran	Jenis output saat ini	Jenis output tegangan	Model RS485
Lingkup	16 arah atau 8 arah atau 0-360 derajat		
Tegangan suplai	DC12V ~ 24 V		
Sinyal keluaran	4-20mA (sistem tiga kawat).	0,4-2V atau 0-2V 1-5V atau 0-5V	Protokol MODBUS
kesalahan	±3°	±3°	±3°
Kapasitas beban	≤500Ω	Ω ≥2K	
Lingkungan penggunaan	-20 °C ~ +55 °C, kelembaban relatif 35-85% non-kondensasi		
Mulai angin	≥0.8m/dtk		
Konsumsi daya secara keseluruhan (DC24V)	≤700mW	≤300mW	≤300mW
berat	≤0.15Kg		

Harap beri tahu kami secara terpisah jika Anda memiliki spesifikasi khusus

Keempat, karakteristik fungsional

Sejak produk ini dipasarkan, ia telah memenangkan pujian dari sebagian besar pengguna dengan kualitas yang sangat baik dan kinerja yang sangat baik, dan memiliki karakteristik sebagai berikut:

◆ Ukuran kecil, mudah dibawa, instalasi sederhana, penampilan indah;

◆ Ketahanan korosi yang kuat dan tahan cuaca;

◆ Akurasi pengukuran tinggi, jangkauan pengukuran yang luas, stabilitas yang baik;

◆ Konsumsi daya rendah, kemampuan anti-interferensi yang kuat, pekerjaan stabil jangka panjang;

◆ Rentang adaptasi catu daya yang luas, linearitas informasi data yang baik, jarak transmisi sinyal yang jauh.

Metode Tetap

Sensor harus dipasang secara horizontal untuk memastikan keakuratan data arah angin; Metode pemasangan flensa, diameter flensa instalasi di bawah sensor Φ70mm, empat lubang pemasangan adalah Φ3.3mm, empat lubang pemasangan didistribusikan secara merata dan kemudian pada keliling Φ 62.5mm, instalasi diperbaiki dengan flensa, dan dimensi pemasangan adalah sebagai berikut:

8. Definisi warna garis

Jenis garis	**Warna yang umum digunakan**	**Warna alternatif**
Warna Garis Listrik	Merah	
Warna Garis Ground	Hitam	
Warna Garis Sinyal	Kuning -------- A+	
	Biru --------- B-	

10. Korespondensi keluaran sinyal

Korespondensi output sinyal tegangan (0-5V) (akurasi komprehensif tegangan ±2%)

Utara: 5V N N NW: 0.31V NW: 0.63V N-N: 0.94V

Timur: 1.25V N-N: 1.56V NW: 1.88V NSW: 2.19V

Selatan: 2.5V NSW: 2.81V WW: 3.13V West-NSW: 3.44V

Barat: 3.75V West-N: 4.06V NW: 4.38V NNW: 4.69V

Korespondensi output sinyal tegangan (1-5V) (akurasi komprehensif tegangan ±2%)

Utara: 1V Utara-timur laut: 1.25V NW: 1.5V Timur-timur laut: 1.75V

Timur: 2V Timur-Tenggara: 2.25V Tenggara: 2.5V Selatan-Tenggara: 2.75V

Selatan: 3V Selatan-NSW: 3.25V Barat Daya: 3.5V Barat-NSW: 3.75V

Barat: 4V Barat-N: 4.25V NW: 4.5V N-N: 4.75V

Korespondensi output sinyal tegangan (0-2V) (akurasi komprehensif tegangan ±2%)

Utara: 2V N N NW: 0.13V NW: 0.25V NW: 0.38V

Timur: 0.5V NSW: 0.62V NW: 0.75V NSW: 0.88V

Selatan: 1.0V Selatan-NSW: 1.12V WW: 1.25V Barat-NSW: 1.38V

Barat: 1.5V West-N: 1.62V NW: 1.75V NW: 1.88V

Korespondensi output sinyal tegangan (0,4-2V) (akurasi komprehensif tegangan ±2%)

Utara: 2V Utara-timur laut: 0.4V NW: 0.51V Timur-timur laut: 0.61V

Timur: 0.72V N-N: 0.82V NSW: 0.93V N-N: 1.04V

Selatan: 1.14V Selatan-NSW: 1.25V WW: 1.35V Barat-NSW: 1.46V

Barat: 1.57V West-N: 1.67V NW: 1.78V NNW: 1.88V

Korespondensi output sinyal saat ini (akurasi komprehensif saat ini ±2%)

Utara: 4mA Utara-timur laut: 5mA NW: 6mA Timur-timur laut: 7mA

Timur: 8mA Timur-Tenggara: 9mA Tenggara: 10mA Selatan-Tenggara: 11mA Selatan: 12mA Barat daya: 13mA Barat Daya: 14mA Barat-Barat Daya: 15mA Barat: 16mA Barat-N: 17mA NW: 18mA N-N: 19mA

11. **Protokol komunikasi RS485/232**

Subset perintah menggunakan protokol MODBUS-RTU, menggunakan perintah read register (03) (06).

1. Mode transmisi data:

8 bit data, 1 bit stop bit, tidak ada digit cek.

2. Tingkat transmisi data:

Baud rate default adalah 9600bps, yang tidak dapat dimodifikasi, jadi jika Anda ingin menggunakan baud rate lain, harap nyatakan saat memesan. Tingkat baud dukungan: 9600bps, 4800bps, 2400bps, 1200bps.

3. Format pesan data

(1) Kode fungsi 0x03--- menanyakan isi register perangkat slave

Menguasai pesan perangkat	**Memperbaiki pesan dari perangkat**
Alamat perangkat Slave (0x01-0xFE 1 byte).	Alamat perangkat Slave (0x01-0xFE 1 byte).
Kode fungsi (0x03 1 byte).	Kode fungsi (0x03 1 byte).
Mulai alamat register (2 byte).	Jumlah byte di area data (2* mendaftar dengan 1 byte).
Jumlah register (2 byte).	Area data (daftar isi 2* Jumlah register 1 byte).
Digit pemeriksaan CRC (2 byte)	Digit pemeriksaan CRC (2 byte)

(2) Kode fungsi 0x06--- jumlah register perangkat slave

Menguasai pesan perangkat	**Memperbaiki pesan dari perangkat**
Alamat perangkat Slave (0x01-0xFE 1 byte).	Alamat perangkat Slave (0x01-0xFE 1 byte).
Kode fungsi (0x06 1 byte).	Kode fungsi (0x06 1 byte).
Mulai alamat register (2 byte).	Jumlah byte di area data (2* mendaftar dengan 1 byte).
Data yang ditulis ke register (2* register adalah 1byte).	Area data (daftar isi 2* Register nomor 1 byte).
Digit pemeriksaan CRC (2 byte)	Digit pemeriksaan CRC (2 byte)

Catatan: 1, kode uji CRC rendah di depan, bit tinggi di belakang, alamat register, jumlah register, data tinggi di depan, rendah di belakang; 2. Panjang kata register adalah 16bit (dua byte);

4. Daftarkan deskripsi dan format perintah

(1) Tabel definisi register data parameter

Alamat Register (Hex)	Daftarkan konten	Jumlah register	Status pendaftaran	Rentang Data (Hex)
0x002A	Angin	1	hanya membaca	0 ~ 3600 (0x00-0x0E10)

Tip: Sejak 20 Desember 2013, semua alamat register sensor telah dimodifikasi untuk 0x002A, dan alamat register nilai kecepatan angin yang digunakan oleh pelanggan lama 0x0010, 0x0002, 0x0000, Perjanjian yang direvisi masih mendukung alamat di atas dan tidak perlu dimodifikasi oleh pelanggan.

Alamat Register (Hex)	Daftarkan konten	Jumlah register	Status pendaftaran	Rentang Data (Hex)
0x2000	Alamat perangkat	1	Membaca dan menulis	1 ~ 127 (0x01 ~ 0x7F)

(2) Contoh perintah:

Semua byte alamat register, bagian nomor register, byte data di urutan pertama dan rendah dalam perintah;

CRC memeriksa digit byte rendah di depan, byte bit tinggi di belakang;

Baca nilai sensor saat ini:

(Dari alamat perangkat 02, baud rate 9600, N, 8, 1)

Alamat perangkat Slave	Kode fungsi	Alamat daftar awal		Jumlah register		CRC-L	CRC-H
0x02	0x03	0x00	0x2A	0x00	0x01	0xA5	0xF1

Tanggapi dari perangkat:

Alamat perangkat Slave	Kode fungsi	Jumlah byte di area data	Daftarkan data		CRC-L	CRC-H
0x02	0x03	0x02	0x00	0x00	0xFC	0x44

Untuk mengubah alamat perangkat:

(Dari alamat perangkat 02, ubah ke 03)

Alamat perangkat Slave	Kode fungsi	Alamat daftar awal		Data yang dimodifikasi		CRC-L	CRC-H
0x02	0x06	0x20	0x00	0x00	0x03	0XC2	0x38

Tanggapi dari perangkat:

Alamat perangkat Slave	Kode fungsi	Alamat daftar awal		Data yang dimodifikasi		CRC-L	CRC-H
0x02	0x06	0x20	0x00	0x00	0x03	0XC2	0x38

Setelah memodifikasi alamat perangkat, Anda harus menyalakannya kembali.

Catatan Khusus: Jika alamat perangkat tidak berubah setelah alamat 2000 diubah, alamat tersebut akan diubah 4000 b20>Alamat.

Petunjuk pengkabelan: garis merah ke elektroda positif, kawat hitam ke elektroda negatif, kawat kuning ke A +, kawat biru ke B-

XII. Pemeliharaan dan pemeliharaan

Instrumen ini adalah produk elektronik presisi, perawatan dan pemeliharaan yang benar untuk membantu melindungi kinerja instrumen, memperpanjang masa pakai instrumen, harap perhatikan poin-poin berikut:

1. Harap gunakan instruksi manual dengan benar sesuai dengan persyaratan instruksi manual, dan uang yang salah dapat menyebabkan kerusakan pada instrumen.
2. Jangan menyeka instrumen dengan cairan yang mudah menguap, jika tidak maka dapat menyebabkan perubahan warna dan deformasi instrumen; Bersihkan dengan kain lembut untuk menghindari goresan pada lapisan pelindung eksternal instrumen dan memperpanjang masa pakai instrumen.
3. Instrumen harus ditangani dengan ringan, dan tidak boleh dijatuhkan atau diberi tekanan, jika tidak maka akan menyebabkan instrumen berubah bentuk dan papan sirkuit internal rusak.
4. Jangan menyentuh bagian penginderaan saat instrumen diisi untuk mempengaruhi hasil pengukuran atau menyebabkan kerusakan pada sirkuit internal instrumen.
5. Jangan membongkar dan memodifikasi instrumen tanpa izin untuk menghindari kerusakan pada instrumen.
6. Saat instrumen digunakan, sekrup harus dipasang dengan kuat, jika tidak maka dapat merusak instrumen.
7. Periksa tegangan catu daya instrumen secara teratur untuk memastikan operasi normal instrumen

LAMPIRAN C

LAMPIRAN 6 Coding *Transmitter*

```
#include <nRF24L01.h>
#include <RF24.h>
#define CE_PIN 10
#define CSN_PIN 9

// Anemometer
int anemo_Adc = 0;
float anemo_Val = 0.0;
float anemo_Sensor = 0.0;

// Direction
int dir_Adc = 0;
float dir_Val = 0.0;
float dir_Sensor = 0.0;
int kali;

// Rain
int tiping = 3;
long int tipingAmount;
double rain =0.0;
double rain_minute= 0.0;

int fsec, sec;
```

```
// NRF
const byte address[6] = "76543";
RF24 radio(CE_PIN, CSN_PIN);
float data[10]; // depending on the number of sensors you want to use
int header = 9251; //Header node 1

uint32_t timer = millis(), timer2 = millis();



void setup() {
  Serial.begin(115200);
  pinMode(tiping, INPUT);
   attachInterrupt(digitalPinToInterrupt(tiping), tipingCount, FALLING);

  radio.begin();
  radio.openWritingPipe(address);
  Serial.println("NRF Receiver");

}

void loop() {
  // put your main code here, to run repeatedly:
  if (millis() - timer > 1000)
  {
```

```
    timer += 1000;
    Baca_Sensor();
    data[0] = header;
    data[1] = anemo_Sensor;
    data[2] = dir_Sensor;
    data[3] = rain_minute;
    data[4] = tipingAmount;
    data[5] = rain;
    radio.write( &data, sizeof(data) );
    String message = String(data[1],1) + ";" + String(data[2],1) + ";" +
String(data[3],2) +";"+String(data[4],1) +";"+ String(data[5],1);
    //Serial.println("Data Sending : " + message);
    //Serial.println("\n");


  }
}


void Baca_Sensor()
{
    sensorDirection();
    //Serial.print("\t | \t");
    sensorAnemo();
    //Serial.print("\t | \t");
    sensorRain();
}
```

```
void sensorAnemo()
{
  anemo_Adc = analogRead(A1);
  anemo_Val = (anemo_Adc * 5.0) / 1023;
  anemo_Sensor = mapfloat(anemo_Val, 0, 5, 0, 30);
  Serial.print("\t |ADC Anemo (1023) \t");
  Serial.print(anemo_Adc);

  Serial.print("\t |Voltage Anemo (V) \t");
  Serial.print(anemo_Val);

  Serial.print("\t |Anemo Value (m/s) \t");
  Serial.print(anemo_Sensor);
}

void sensorDirection()
{
  dir_Adc = analogRead(A0);
  dir_Val = (dir_Adc * 5.0) / 1023;
  dir_Sensor = mapfloat(dir_Val, 0, 5, 0, 360);
  Serial.print("\t |ADC Dir (1023) \t");
  Serial.print(dir_Adc);

  Serial.print("\t |Voltage Dir (V) \t");
```

```
  Serial.print( dir_Val);
    Serial.print("\t |Wind Direction (°) \t");
  Serial.print(dir_Sensor);


}


void tipingCount()
{
  tipingAmount++;
}



void sensorRain()
{
  /*
      - Curah hujan 1 mm adl jml air hujan yang jatuh di permukaan per satuan
        luas (m2) dengan volume sbnyk 1 liter tanpa ada yang menguap,
meresap/mengalir.


      - Tinggi curah hujan (cm) = volume yang dikumpulkan (mL) / area
pengumpulan (cm2)
      - Luas kolektor 5,5cm x 3,5cm = 19,25 cm2
      - Koleksi per ujung tip kami dapat dengan cara menuangkan 100ml air
        ke kolektor kemudian menghitung berapa kali air terbuang dari tip.
      - Dlm perhitungan yang dilakukan air terbuang sebanyak 70 kali. 100ml / 70=
1.42mL per tip.
```

```
      Jadi 1 tip bernilai 1.42 / 19.25 = 0,07cm atau 0.70 mm curah hujan
     */

  rain = 0.74 * tipingAmount;
//   Serial.print("Jumlah tipintg : ");
  Serial.print("\t |tiping (tip) \t");
  Serial.print(tipingAmount);
  Serial.print("\t |Volume tinggi (mm) \t");
  Serial.print(rain);
  hitung_jam();
  Serial.print("\t |Curah Hujan (mm/hour) \t");
  Serial.println(rain_minute);
//   Serial.println("mm");
}

void hitung_jam()
{
   rain_minute = rain;
   if (millis() - timer2 > 1000)
  {
   timer2 += 1000;
   if (tipingAmount>=1 ){
   sec++;
   }
   rain_minute = rain_minute/sec * 3600;
```

```
    if (sec > 3600)
    {
      sec = 0;
      tipingAmount=0;
    }
  }
}

float mapfloat(float x, float in_min, float in_max, float out_min, float out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

LAMPIRAN 7 Coding *Receiver*

```
#include <SPI.h>
 #include <nRF24L01.h>
 #include <RF24.h>
 #define CE_PIN 10
 #define CSN_PIN 9

const byte address[6] = "76543";

RF24 radio(CE_PIN, CSN_PIN);
float data[6]; // depending on the number of sensors used

unsigned long lastReceiveTime = 0;
unsigned long currentTime = 0;

//float data1;
//float data2, data3, data4;
float dt[10];

void setup()
 {
   Serial.begin(115200);
   delay(1000);
//   Serial.println("Nrf24L01 Receiver Starting");
   radio.begin();
```

```
  radio.openReadingPipe(1,address);
  radio.startListening();

 }

void loop()
 {
 if ( radio.available() )
 {
    radio.read(&data, sizeof(data));
    lastReceiveTime = millis(); // At this moment we have received the data

   if (data[0] == 9251) // cek header if correct
   {
     dt[0] = data[0];
     dt[1] = data[1];
     dt[2] = data[2];
     dt[3] = data[3];
     dt[4] = data[4];
     dt[5] = data[5];

     String Message1 = String(dt[1],2) + ";" + String(dt[2],2) + ";" +
String(dt[3],2)+";"+ String(dt[4],2) +";"+String(dt[5],2);
     Serial.println(Message1);
   }
```

}

}

LAMPIRAN 8 Coding ESP32 Integrasi Ke *Thingspeak*

```
/*
  WriteMultipleFields

  Description: Writes values to fields 1,2,3,4 and status in a single ThingSpeak update every 20 seconds.

  Hardware: ESP32 based boards

  !!! IMPORTANT - Modify the secrets.h file for this project with your network connection and ThingSpeak channel details. !!!

  Note:
  - Requires installation of EPS32 core. See https://github.com/espressif/arduino-esp32/blob/master/docs/arduino-ide/boards_manager.md for details.
  - Select the target hardware from the Tools->Board menu
  - This example is written for a network using WPA encryption. For WEP or WPA, change the WiFi.begin() call accordingly.

  ThingSpeak ( https://www.thingspeak.com ) is an analytic IoT platform service that allows you to aggregate, visualize, and
  analyze live data streams in the cloud. Visit https://www.thingspeak.com to sign up for a free account and create a channel.

  Documentation for the ThingSpeak Communication Library for Arduino is in the README.md folder where the library was installed.
```

See https://www.mathworks.com/help/thingspeak/index.html for the full ThingSpeak documentation.

For licensing information, see the accompanying license file.

*/

```
#include <WiFi.h>

#include "secrets.h"

#include "ThingSpeak.h" // always include thingspeak header file after other header files and custom macros


char ssid[] = SECRET_SSID;   // your network SSID (name)

char pass[] = SECRET_PASS;   // your network password

int keyIndex = 0;            // your network key Index number (needed only for WEP)

WiFiClient  client;


unsigned long myChannelNumber = SECRET_CH_ID;

const char * myWriteAPIKey = SECRET_WRITE_APIKEY;


// Initialize our values

int number1 = 0;

int number2 = random(0, 100);

int number3 = random(0, 100);
```

```
//int number4 = random(0,100);
String myStatus = "";
uint32_t timer = millis()+5000;

String data;

float Index1, Index2, Index3;
String firstValue, secondValue, thirdValue;



void setup() {
  Serial.begin(115200);  //Initialize serial
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo native USB port only
  }
  WiFi.begin(SECRET_SSID, SECRET_PASS);

  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(200);
  }
  Serial.println("");
  Serial.println("WiFi connected");

  //WiFi.mode(WIFI_STA);
```

```
  ThingSpeak.begin(client);  // Initialize ThingSpeak
}
int adaserial = 0;
void loop() {

  adaserial = 0;
  while (Serial.available())
  {
    char message = Serial.read();
    data += message;
    adaserial++; //mengambil status ada serial
  }
  if (adaserial > 0) {
    Index1 = data.indexOf(';');
    Index2 = data.indexOf(';', Index1 + 1);
    Index3 = data.indexOf(';', Index2 + 1);
    //Index4 = data.indexOf('\r', Index3 + 1);
    firstValue = data.substring(0, Index1);
    secondValue = data.substring(Index1 + 1, Index2);
    thirdValue = data.substring(Index2 + 1, Index3);
    //fourthValue = data.substring(Index3 + 1, Index4);
    Serial.print("data 1:"); Serial.print(firstValue); Serial.println(" m/s");
    Serial.print("data 2:"); Serial.print(secondValue); Serial.println(" °");
    Serial.print("data 3:"); Serial.print(thirdValue); Serial.println(" mm/hour");
    //Serial.print("data 3:"); Serial.println(fourthValue);
```

```
  data = "";
 }


 if (millis() > timer)
 {
  timer += 15000;
  // Connect or reconnect to WiFi
  if (WiFi.status() != WL_CONNECTED) {
   Serial.print("Attempting to connect to SSID: ");
   Serial.println(SECRET_SSID);
   while (WiFi.status() != WL_CONNECTED) {
    WiFi.begin(ssid, pass);  // Connect to WPA/WPA2 network. Change this line if using open or WEP network
    Serial.print(".");
    delay(5000);
   }
   Serial.println("\nConnected.");
  }


  // set the fields with the values
  ThingSpeak.setField(1, firstValue);
  ThingSpeak.setField(2, secondValue);
  ThingSpeak.setField(3, thirdValue);


  // write to the ThingSpeak channel
```

```
  int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
  if (x == 200) {
    Serial.println("Channel update successful.");
  }
  else {
    Serial.println("Problem updating channel. HTTP error code " + String(x));
  }

  // change the values
  //  number1++;
  //  if(number1 > 99){
  //    number1 = 0;
  //  }
  //  number2 = random(0,100);
  //  number3 = random(0,100);
  //  number4 = random(0,100);
 }
 delay(200);
}
```