# LAMPIRAN

**LAMPIRAN 1  Proses perakitan Sensor**

**LAMPIRAN 2 Pemotongan Rangka Tiang Weather Station**

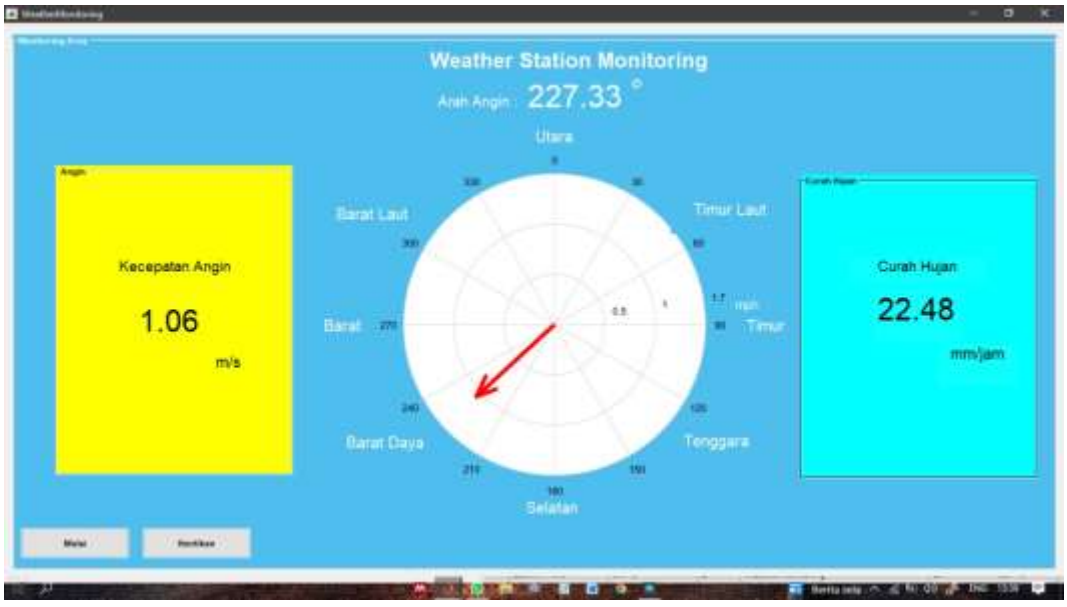
**LAMPIRAN 3 Pengelasan Tiang Weather Station**

**LAMPIRAN 4 Hasil Jadi Weather Station**

**LAMPIRAN 5 Pengujian Dengan Cuaca *Real***

**LAMPIRAN 6 Pengujian Jarak *Transmitter* Dan *Receiver***


**LAMPIRAN 5 Pembacaan Sensor Pada *Serial Monitor* Dan *Thingspeak***




**LAMPIRAN 6 Monitoring Di Software Matlab**

## LAMPIRAN 7 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 8 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 9 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.

  Copyright 2020, The MathWorks, Inc.

*/
```

```
#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);
```