LAMPIRAN

Lampiran 1: Kode Pemrograman Alat

```
#include <BlynkSimpleEsp32.h>
#include <WiFi.h>
#include <DHT.h>
#define DHTPIN 2
#define DHTTYPE DHT11
#define ledPin 13 //RL1
DHT dht(DHTPIN,DHTTYPE);
char ssid[] = "numpixel";
char pass[] = "tukoknokopi1";

//----- BLYNK Configuration -----//
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPL6Osv-ktnD"
#define BLYNK_TEMPLATE_NAME "Project DHT11 ESP32"
#define BLYNK_AUTH_TOKEN "ETKbR1EhkBiYz2JOSKcwJr18ExvzBRyb"

//----- pompa & Blower Configuration -----//
#define PWM_pompa    25
#define PWM_fan     26
float Temp, TempD, error_Temp;
float RH_Linear, RH_True, HumD, error_Hum;
int i, j;

//----- Fuzzy Variable -----//
float miu_EH[6];        // Error humidity fuzzyfication
float miu_ET[6];        // Error temperature fuzzyfication
float miu_pompa[6][6];   // pompaer fuzzyfication
float miu_fan[6][6];    // Fan fuzzyfication
```

```
float z[26];
float miu_A, miu_B;
float PWM;

//----- Fuzzy Logic Void -----//
// OMF Of pompaer
const int HM = 0;
const int HCH = 50;
const int HSH = 100;
const int HCP = 150;
const int HSP = 200;

// OMF Of Fan
const int BM = 0;
const int BSL = 20;
const int BML = 40;
const int BMC = 60;
const int BSC = 80;

// Error temperature fuzzy
void NLET();
void NSET();
void ZET();
void PSET();
void PLET();
void error_temperature_fuzzyfication();

// Error humidity fuzzy
void NLEH();
void NSEH();
```

```
void ZEH();
void PSEH();
void PLEH();
void error_humidity_fuzzyfication();


// Rule base
void pompaer_rulebase();
void fan_rulebase();

// Defuzzyfication
void pompaer_defuzzyfication();
void fan_defuzzyfication();

void setup()
{
  // put your setup code here, to run once:
  Serial.begin(115200);
  pinMode(ledPin,OUTPUT);
  pinMode(PWM_pompa,OUTPUT);
  pinMode(PWM_fan,OUTPUT);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  dht.begin();
}

void loop()
{
  // put your main code here, to run repeatedly:
   float dly;
   int RH_True = dht.readHumidity();
```

```
  int Temp= dht.readTemperature();
  error_Temp = TempD - Temp;
  error_Hum = HumD - RH_True;
  error_temperature_fuzzyfication();
  error_humidity_fuzzyfication();
  pompaer_rulebase();
  pompaer_defuzzyfication();
  fan_rulebase();
  fan_defuzzyfication();

  Serial.print("Kelembaban = ");
  Serial.print(humidity);
  Serial.print("%\t");
  Serial.print("Suhu = ");
  Serial.print(temperature);
  Serial.println("°C");
  Blynk.virtualWrite(V1,humidity);
  Blynk.virtualWrite(V0,temperature);
  dly = output*1000;
  Serial.print(input);Serial.print("   ");Serial.print(output);Serial.print("   ");
  Serial.println(dly);
  delay(1000);
  Blynk.run();
}

//----- Error Temperature Fuzzyfication -----//
void NLET()
{
 if(error_Temp < -10)
    miu_ET[1] = 1;
```

```
 if(error_Temp <= -10 && error_Temp <= -5)
    miu_ET[1] = (-error_Temp - 5)/5;
 if(error_Temp > -5)
    miu_ET[1] = 0;
}

void NSET()
{
 if(error_Temp < -10)
    miu_ET[2] = 0;
 if(error_Temp <= -10 && error_Temp < -5)
    miu_ET[2] = (error_Temp + 10)/5;
 if(error_Temp <= -5 && error_Temp <= 0)
    miu_ET[2] = -(error_Temp/5);
 if(error_Temp > 0)
    miu_ET[2] = 0;
}

void ZET()
{
 if(error_Temp < -5)
    miu_ET[3] = 0;
 if(error_Temp <= -5 && error_Temp < 0)
    miu_ET[3] = (error_Temp + 5)/5;
 if(error_Temp <= 0 && error_Temp <= 5)
    miu_ET[3] = (-error_Temp + 5)/5;
 if(error_Temp > 5)
    miu_ET[3] = 0;
}
```

```
void PSET()
{
 if(error_Temp < 0)
    miu_ET[4] = 0;
 if(error_Temp <= 0 && error_Temp < 5)
    miu_ET[4] = error_Temp/5;
 if(error_Temp <= 5 && error_Temp <= 10)
    miu_ET[4] = (-error_Temp + 10)/5;
 if(error_Temp > 10)
    miu_ET[4] = 0;
}

void PLET()
{
 if(error_Temp < 5)
    miu_ET[5] = 0;
 if(error_Temp <= 5 && error_Temp <= 10)
    miu_ET[5] = (error_Temp - 5)/5;
 if(error_Temp > 10)
    miu_ET[5] = 1;
}

void error_temperature_fuzzyfication()
{
 NLET();
 NSET();
 ZET();
 PSET();
 PLET();
}
```

```
//----- Error Humidity Fuzzyfication -----//
void NLEH()
{
 if(error_Hum < -20)
    miu_EH[1] = 1;
 if(error_Hum <= -20 && error_Hum <= -10)
    miu_EH[1] = (-error_Hum - 10)/10;
 if(error_Hum > -10)
    miu_EH[1] = 0;
}

void NSEH()
{
 if(error_Hum < -20)
    miu_EH[2] = 0;
 if(error_Hum <= -20 && error_Hum < -10)
    miu_EH[2] = (error_Hum + 20)/10;
 if(error_Hum <= -10 && error_Hum <= 0)
    miu_EH[2] = -(error_Hum/10);
 if(error_Hum > 0)
    miu_EH[2] = 0;
}

void ZEH()
{
 if(error_Hum < -10)
    miu_EH[3] = 0;
 if(error_Hum <= -10 && error_Hum < 0)
    miu_EH[3] = (error_Hum + 10)/10;
```

```
 if(error_Hum <= 0 && error_Hum <= 10)
    miu_EH[3] = (-error_Hum + 10)/10;
 if(error_Hum > 10)
    miu_EH[3] = 0;
}

void PSEH()
{
 if(error_Hum < 0)
    miu_EH[4] = 0;
 if(error_Hum <= 0 && error_Hum <10)
    miu_EH[4] = error_Hum/10;
 if(error_Hum <= 10 && error_Hum <= 20)
    miu_EH[4] = (-error_Hum + 20)/10;
 if(error_Hum > 20)
    miu_EH[4] = 0;
}

void PLEH()
{
 if(error_Hum < 10)
    miu_EH[5] = 0;
 if(error_Hum <= 10 && error_Hum <= 20)
    miu_EH[5] = (error_Hum - 10)/10;
 if(error_Hum > 20)
    miu_EH[5] = 1;
}

void error_humidity_fuzzyfication()
{
```

```
 NLEH();
 NSEH();
 ZEH();
 PSEH();
 PLEH();
}

//----- Mamdani Rule Base -----//
void pompaer_rulebase()
{for(i=1; i<=5; i++)
   {for(j=1; j<=5; j++)
      {
       if(miu_ET[i] < miu_EH[j])
          miu_pompa[i][j] = miu_ET[i];
       if(miu_ET[i] > miu_EH[j])
          miu_pompa[i][j] = miu_EH[j];
       if(miu_ET[i] == miu_EH[j])
          miu_pompa[i][j] = miu_ET[i];
      }
   }
}


void fan_rulebase()
{for(i=1; i<=5; i++)
   {for(j=1; j<=5; j++)
      {
       if(miu_ET[i] < miu_EH[j])
          miu_fan[i][j] = miu_ET[i];
```

```
        if(miu_ET[i] > miu_EH[j])
          miu_fan[i][j] = miu_EH[j];
        if(miu_ET[i] == miu_EH[j])
          miu_fan[i][j] = miu_ET[i];
       }
    }
}

//----- pompaer Defuzzyfication (Takagi Sugeno Kang) -----//
void pompaer_defuzzyfication()
{
 z[1] = HM;  z[6] = HCH;  z[11] = HSP; z[16] = HSH; z[21] = HSP;
 z[2] = HCH; z[7] = HCH;  z[12] = HCP; z[17] = HCP; z[22] = HSP;
 z[3] = HCH; z[8] = HCH;  z[13] = HM;  z[18] = HCP; z[23] = HSP;
 z[4] = HCH; z[9] = HCH;  z[14] = HCH; z[19] = HCP; z[24] = HCP;
 z[5] = HCH; z[10] = HCH; z[15] = HM;  z[20] = HCP; z[25] = HCP;
 miu_A = 0;
 miu_B = 0;

 miu_A = miu_pompa[1][1] * z[1] +
        miu_pompa[1][2] * z[2] +
        miu_pompa[1][3] * z[3] +
        miu_pompa[1][4] * z[4] +
        miu_pompa[1][5] * z[5] +

        miu_pompa[2][1] * z[6] +
        miu_pompa[2][2] * z[7] +
        miu_pompa[2][3] * z[8] +
        miu_pompa[2][4] * z[9] +
        miu_pompa[2][5] * z[10] +
```

```
        miu_pompa[3][1] * z[11] +
        miu_pompa[3][2] * z[12] +
        miu_pompa[3][3] * z[13] +
        miu_pompa[3][4] * z[14] +
        miu_pompa[3][5] * z[15] +

        miu_pompa[4][1] * z[16] +
        miu_pompa[4][2] * z[17] +
        miu_pompa[4][3] * z[18] +
        miu_pompa[4][4] * z[19] +
        miu_pompa[4][5] * z[20] +

        miu_pompa[5][1] * z[21] +
        miu_pompa[5][2] * z[22] +
        miu_pompa[5][3] * z[23] +
        miu_pompa[5][4] * z[24] +
        miu_pompa[5][5] * z[25];

for(i=1; i<=5; i++)
   {for(j=1; j<=5; j++)
      {
        miu_B = miu_B + miu_pompa[i][j];
      }
   }
 PWM = miu_A/miu_B;
 PWM_pompa=PWM*255;


}
```

```
//----- Fan Defuzzyfication (Takagi Sugeno Kang) -----//
void fan_defuzzyfication()
{
 z[1] = BSC; z[6] = BMC;  z[11] = BML; z[16] = BSL; z[21] = BM;
 z[2] = BSC; z[7] = BMC;  z[12] = BML; z[17] = BSL; z[22] = BM;
 z[3] = BSC; z[8] = BMC;  z[13] = BML; z[18] = BSL; z[23] = BM;
 z[4] = BSC; z[9] = BMC;  z[14] = BML; z[19] = BSL; z[24] = BM;
 z[5] = BSC; z[10] = BMC; z[15] = BML; z[20] = BSL; z[25] = BM;
 miu_A = 0;
 miu_B = 0;

 miu_A = miu_fan[1][1] * z[1] +
         miu_fan[1][2] * z[2] +
         miu_fan[1][3] * z[3] +
         miu_fan[1][4] * z[4] +
         miu_fan[1][5] * z[5] +

         miu_fan[2][1] * z[6] +
         miu_fan[2][2] * z[7] +
         miu_fan[2][3] * z[8] +
         miu_fan[2][4] * z[9] +
         miu_fan[2][5] * z[10] +

         miu_fan[3][1] * z[11] +
         miu_fan[3][2] * z[12] +
         miu_fan[3][3] * z[13] +
         miu_fan[3][4] * z[14] +
         miu_fan[3][5] * z[15] +
```

```
        miu_fan[4][1] * z[16] +
        miu_fan[4][2] * z[17] +
        miu_fan[4][3] * z[18] +
        miu_fan[4][4] * z[19] +
        miu_fan[4][5] * z[20] +

        miu_fan[5][1] * z[21] +
        miu_fan[5][2] * z[22] +
        miu_fan[5][3] * z[23] +
        miu_fan[5][4] * z[24] +
        miu_fan[5][5] * z[25];

 for(i=1; i<=5; i++)
   {for(j=1; j<=5; j++)
      {miu_B = miu_B + miu_fan[i][j];
      }
   }
 PWM = miu_A/miu_B;
 PWM_fan=PWM*255;
}
```