LAMPIRAN

Lampiran 1

Listing Program Alat Keseluruhan

```
#define DEBUG 0

#if DEBUG == 1
  #define debug(x)    Serial.print(x)
  #define debugln(x)  Serial.println(x)
#else
  #define debug(x)
  #define debugln(x)
#endif

#include "fis_header.h"
// #include <GyverMAX6675.h>
#include <GyverMAX6675_SPI.h>

#include <Fuzzy.h>

#define virtual_serial Serial2
#define PID 1
#define FUZZY 2

// MAX6675K
#define CLK_PIN   52  // SCK
#define DATA_PIN  50  // SO
#define CS_1 48
#define CS_2 46
#define CS_3 44
#define CS_4 42
// GyverMAX6675<CLK_PIN, DATA_PIN, CS_1> sensor1;
// GyverMAX6675<CLK_PIN, DATA_PIN, CS_2> sensor2;
// GyverMAX6675<CLK_PIN, DATA_PIN, CS_3> sensor3;
// GyverMAX6675<CLK_PIN, DATA_PIN, CS_4> sensor4;

#define MAX6675_SPI_SPEED 300000
GyverMAX6675_SPI<CS_1> sensor1;
GyverMAX6675_SPI<CS_2> sensor2;
GyverMAX6675_SPI<CS_3> sensor3;
GyverMAX6675_SPI<CS_4> sensor4;


int suhu_dec[4];
float suhu_float[4];

//HEATER SSR
#define pinssr 9  //PIN OUTPUT PWM
#define pinbuzzer 8  //PIN BUZZER

// Number of inputs to the fuzzy inference system
```

```
const int fis_gcI = 2;
// Number of outputs to the fuzzy inference system
const int fis_gcO = 3;
// Number of rules to the fuzzy inference system
const int fis_gcR = 25;

FIS_TYPE g_fisInput[fis_gcI];
FIS_TYPE g_fisOutput[fis_gcO];

int suhu;
float pembacaan_suhu;
float P, I, D;
float KP, KI, KD;
float KP_F, KI_F, KD_F;  // OUTPUT FUZZY
float kpa = 36;          //60;//36; // ini hitung hitungan gausah diubah kalau perlu
diubah diubah gapapa
float kia = 0.3;         //10;//0.3;//2.5; // berhubungan dengan grafik
float kda = 50;        //50;//1080

float error;
float last_error = 0;
float currentError = 0;
float errorx = 0;
float sumerr = 0;
// float output = 0;
int out_pwm_fuzzy;

//set poin suhu
int SP_suhu, waktu_menyala, metode;
unsigned long pointer_waktu, interval;

String receivedData = "";  // Variabel untuk menyimpan data serial yang
diterima
bool alat_on = false;      // Flag alat bekerja atau stanby
bool flag_setpoin, flag_selesai;

void changeTextOnNextion(int textObject, String message, int pp = 0);
FuzzySet *Hot = new FuzzySet(0, 0, 0, 1);
FuzzySet *Warm = new FuzzySet(1, 1.2, 2, 3);
FuzzySet *Cold = new FuzzySet(3, 4.5, 5.5, 10);
FuzzySet *VeryCold = new FuzzySet(10, 15, 20, 100);

FuzzySet *verylow = new FuzzySet(0, 0, 0, 30);
FuzzySet *low = new FuzzySet(90, 90, 100, 120);
FuzzySet *medium = new FuzzySet(220, 220, 220, 220);
FuzzySet *high = new FuzzySet(220, 220, 220, 220);

Fuzzy *fuzzy = new Fuzzy();

FuzzyInput *temperature = new FuzzyInput(1);
FuzzyOutput *heaterPower = new FuzzyOutput(1);
```

```
void setup() {
  init_fuzzy();
  Serial.begin(9600);
  Serial2.begin(9600);

  pinMode(pinssr, OUTPUT);
  pinMode(pinbuzzer, OUTPUT);
  digitalWrite(pinssr, 0);
  digitalWrite(pinbuzzer, 0);
  delay(500);
}

void loop() {
  if (virtual_serial.available() > 0) {
    char incomingChar = virtual_serial.read();  // Membaca satu karakter dari
serial
    receivedData += incomingChar;               // Menambahkan karakter ke
receivedData

    if (incomingChar == '*') {
      // Jika newline diterima (akhir dari data), proses receivedData
      processData(receivedData);
      alat_on = true;
      // delay(1000);
      receivedData = "";  // Reset receivedData untuk pesan berikutnya
    } else if (incomingChar == '&') {
      alat_on = false;
      debug("off");
      receivedData = "";  // Reset receivedData untuk pesan berikutnya
    }
  }

  if (alat_on == true) {
    // interval = (millis() - pointer_waktu) / 1000;
    if (interval < waktu_menyala && alat_on) {

      if(flag_setpoin) interval = (millis() - pointer_waktu) / 1000;

      pembacaan_suhu = 0.0;
      suhu = 0;
      for(int i = 0; i < 4; i++){
        bacaSemuaSensor();
        pembacaan_suhu +=
(suhu_float[0]+suhu_float[1]+suhu_float[2]+suhu_float[3]) / 4;
        suhu += (suhu_dec[0]+suhu_dec[1]+suhu_dec[2]+suhu_dec[3]) / 4;
        digitalWrite(13, (!(digitalRead(13))));
        delay(250);
      }
      pembacaan_suhu = pembacaan_suhu/4;
      suhu = suhu/4;

      Serial.println(pembacaan_suhu);
```

```
// changeTextOnNextion(0, (metode == 1) ? "PID" : "FUZZY");
changeTextOnNextion(1, String(pembacaan_suhu));
// changeTextOnNextion(2, String(interval / 60));
changeTextOnNextion(2, detik2time(interval));
changeTextOnNextion(3, String(suhu));
changeTextOnNextion(4, String(SP_suhu));

changeTextOnNextion(5, String(suhu_float[0]));
changeTextOnNextion(6, String(suhu_float[1]));
changeTextOnNextion(7, String(suhu_float[2]));
changeTextOnNextion(8, String(suhu_float[3]));
// delay(250);


if (metode == PID) {

  currentError = (float)SP_suhu - (pembacaan_suhu + 0.5);
  // Calculate derivative of error
  float derivativeError = calculateDerivativeError(currentError);

  g_fisInput[0] = currentError;
  g_fisInput[1] = derivativeError;

  g_fisOutput[0] = 0;
  g_fisOutput[1] = 0;
  g_fisOutput[2] = 0;

  fis_evaluate();

  // Set output vlaue: Kpf
  KP_F = g_fisOutput[0];
  // Set output vlaue: Kif
  KI_F = g_fisOutput[1];
  // Set output vlaue: Kdf
  KD_F = g_fisOutput[2];

  // Update PID parameters
  KP = kpa * KP_F;
  KI = kia * KI_F;
  KD = kda * KD_F;

  // Compute PID output
  out_pwm_fuzzy = computePID(pembacaan_suhu);

  // Constrain output to prevent values outside PWM range
  if (out_pwm_fuzzy < 0) {
    out_pwm_fuzzy = 0;
  }
  if (out_pwm_fuzzy > 255) {
    out_pwm_fuzzy = 255;
  }
```

```
    // Apply control signal to heater
    if((int)pembacaan_suhu < SP_suhu)
      analogWrite(pinssr, out_pwm_fuzzy);
    else
      analogWrite(pinssr, 0);

    if((int)pembacaan_suhu >= SP_suhu && flag_setpoin == false)
    {
      bunyi(1);
      flag_setpoin = true;
      pointer_waktu = millis();
    }

    // Debug output
    debug("PID - Error: ");
    debug(currentError);
    debug(" DerivError: ");
    debug(derivativeError);
    debug(" Output: ");
    debugln(out_pwm_fuzzy);
  }

  else if (metode == FUZZY) {
    // Calculate current error and derivative
    currentError = SP_suhu - (pembacaan_suhu + 0.5);
    // float derivativeError = calculateDerivativeError(currentError);
    // int out_pwm_fuzzy;

    fuzzy->setInput(1, currentError);
    fuzzy->fuzzify();
    out_pwm_fuzzy = (int)fuzzy->defuzzify(1);

    // Debug output
    debug(" Output pwm fuzzy: ");
    debugln(out_pwm_fuzzy);
    debugln();

    // Apply control signal
    analogWrite(pinssr, out_pwm_fuzzy);

    if((int)pembacaan_suhu >= SP_suhu && flag_setpoin == false)
    {
      bunyi(1);
      flag_setpoin = true;
      pointer_waktu = millis();
    }
  }
  debugln();
}
else
{
```

```
      alat_on = false;
      bunyi(5);
      flag_selesai = true;
    }

  }

  else if (alat_on == false) {
    analogWrite(pinssr, 0);
  }
}

String detik2time(uint16_t detik) {
    uint16_t menit = detik / 60;
    uint16_t sisa_detik = detik % 60;

    // Format string dengan dua digit untuk menit dan detik
    char buffer[6];
    sprintf(buffer, "%02d:%02d", menit, sisa_detik);

    return String(buffer);
}

void bunyi(uint8_t berapa_kali){
  if(berapa_kali == 1){
    digitalWrite(pinbuzzer, HIGH);
    delay(3000);
    digitalWrite(pinbuzzer, LOW);
  }
  else{
    for(uint8_t n = 0; n < berapa_kali; n++){
      digitalWrite(pinbuzzer, HIGH);
      delay(500);
      digitalWrite(pinbuzzer, LOW);
      delay(500);
    }
  }
}

float calculateDerivativeError(float currentError) {
  static unsigned long previousTime = 0;

  // Waktu saat ini
  unsigned long currentTime = millis();

  // Tambahkan print debug untuk memahami nilai
  debug("Current Time: ");
  debug(currentTime);
  debug(" | Previous Time: ");
  debug(previousTime);

  // Menghitung perbedaan waktu dalam milidetik
```

```
  unsigned long timeDifference = (currentTime > previousTime)
                                 ? (currentTime - previousTime)
                                 : 0;

  // Debug print time difference
  debug(" | Time Difference: ");
  debugln(timeDifference);

  // Mencegah pembagian oleh nol
  float derivativeError = 0;
  if (timeDifference > 0) {
    // Menghitung perbedaan error
    float errorDifference = currentError - last_error;

    // Menghitung nilai error turunan (perubahan error per waktu)
    derivativeError = errorDifference / (float)timeDifference;
  }

  // Update waktu dan nilai error sebelumnya
  previousTime = currentTime;
  last_error = currentError;

  return derivativeError;
}


// Fungsi untuk memproses data nextion yang diterima
void processData(String data) {
  debug(data);
  // Pastikan data sesuai format
  if (data.startsWith("#") && data.endsWith("*")) {
    // Hapus tanda pagar awal dan akhir
    data.remove(0, 1);                // Hapus pagar pertama
    data.remove(data.length() - 1, 1);  // Hapus pagar terakhir

    // Temukan indeks tanda '#'
    int firstIndex = data.indexOf('#');
    int lastIndex = data.lastIndexOf('#');

    // Langsung parsing substring menjadi integer
    int var1 = data.substring(0, firstIndex).toInt();
    int var2 = data.substring(firstIndex + 1, lastIndex).toInt();
    int var3 = data.substring(lastIndex + 1).toInt();

    metode = var1;
    SP_suhu = var2;
    waktu_menyala = var3 * 60;
    interval = 0;
    // pointer_waktu = millis();

    flag_setpoin = false;
    flag_selesai = false;
```

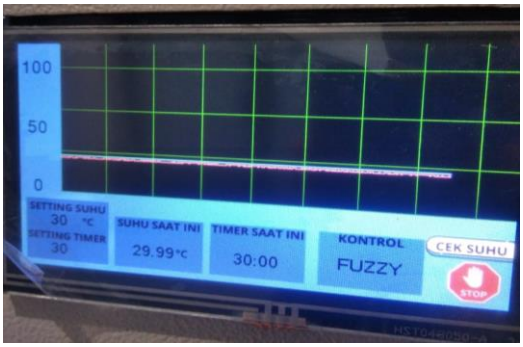
```
    // Tampilkan hasil di debug Monitor
    // debugln("Processed Data:");
    // debug("Var1: ");
    // debugln(var1);
    // debug("Var2: ");
    // debugln(var2);
    // debug("Var3: ");
    // debugln(var3);

    debug("on");
  } else {
    debugln("Invalid format!");
  }
}

void changeTextOnNextion(int textObject, String message, int pp = 0) {
  // Buat string untuk menyimpan perintah
  String command = "";
  if (textObject <= 2) {
    command += "t";
    command += String(textObject);  // Pastikan textObject adalah integer
    command += ".txt=\"";
    command += message;
    command += "\"";
  } else if (textObject <= 4) {
    command += "add 2,";
    command += String(textObject - 3);
    command += ",";
    command += message;
    // command += "\"";
  } else if (textObject <= 8) {
    command += "n";
    command += String(textObject-4);  // Pastikan textObject adalah integer
    command += ".txt=\"";
    command += message;
    command += "\"";
  }

  // Kirim perintah untuk mengubah teks pada objek tertentu
  virtual_serial.print(command);

  // Akhiri perintah dengan karakter akhir (\xFF\xFF\xFF)
  virtual_serial.write(0xFF);
  virtual_serial.write(0xFF);
  virtual_serial.write(0xFF);
}

float computePID(float inp) {
  // Calculate error
  error = SP_suhu - inp;
```

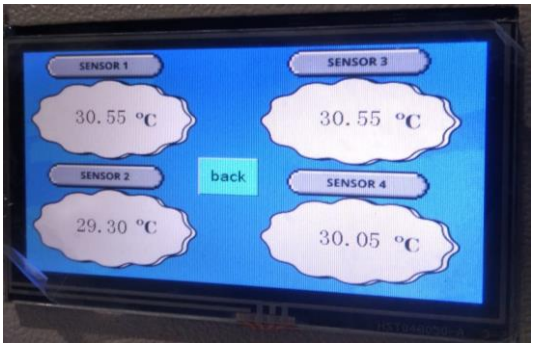
```
  // Proportional term
  P = error * KP;

  // Integral term
  sumerr = error + errorx;
  I = KI * sumerr;

  // Derivative term
  D = KD * (error - errorx);

  // Total PID output
  float out = P + I + D;

  // Store current error for next iteration
  errorx = error;

  return out;
}

// Fungsi untuk baca semua sensor sekaligus
void bacaSemuaSensor() {
  bacaSensor(0, sensor1);
  bacaSensor(1, sensor2);
  bacaSensor(2, sensor3);
  bacaSensor(3, sensor4);
}

// Fungsi baca satu sensor (pakai index)
template <typename T>
void bacaSensor(int index, T& sensor) {
  if (sensor.readTemp()) {
    suhu_dec[index] = sensor.getTempInt();
    suhu_float[index] = sensor.getTemp();
  } else {
    suhu_dec[index] = -1;
    suhu_float[index] = -1;

    debug("Sensor ");
    debug(index + 1);
    debugln(" | Error reading temperature!");
  }
}

void init_fuzzy(){
  temperature->addFuzzySet(VeryCold);
  temperature->addFuzzySet(Cold);
  temperature->addFuzzySet(Warm);
  temperature->addFuzzySet(Hot);

  heaterPower->addFuzzySet(high);
  heaterPower->addFuzzySet(medium);
  heaterPower->addFuzzySet(low);
```

```
  heaterPower->addFuzzySet(verylow);

  fuzzy->addFuzzyInput(temperature);
  fuzzy->addFuzzyOutput(heaterPower);

  // Define fuzzy rules
  FuzzyRuleAntecedent *ifVeryCold = new FuzzyRuleAntecedent();
  ifVeryCold->joinSingle(VeryCold);

  FuzzyRuleConsequent *thenHighPower = new FuzzyRuleConsequent();
  thenHighPower->addOutput(high);

  FuzzyRuleAntecedent *ifCold = new FuzzyRuleAntecedent();
  ifCold->joinSingle(Cold);

  FuzzyRuleConsequent *thenMediumPower = new FuzzyRuleConsequent();
  thenMediumPower->addOutput(medium);

  FuzzyRuleAntecedent *ifWarm = new FuzzyRuleAntecedent();
  ifWarm->joinSingle(Warm);

  FuzzyRuleConsequent *thenLowPower = new FuzzyRuleConsequent();
  thenLowPower->addOutput(low);

  FuzzyRuleAntecedent *ifHot = new FuzzyRuleAntecedent();
  ifHot->joinSingle(Hot);

  FuzzyRuleConsequent *thenVeryLowPower = new FuzzyRuleConsequent();
  thenVeryLowPower->addOutput(verylow);


  FuzzyRule *rule1 = new FuzzyRule(1, ifHot, thenVeryLowPower);
  FuzzyRule *rule2 = new FuzzyRule(1, ifWarm, thenLowPower);
  FuzzyRule *rule3 = new FuzzyRule(1, ifCold, thenMediumPower);
  FuzzyRule *rule4 = new FuzzyRule(1, ifVeryCold, thenHighPower);

  // Add fuzzy rules to Fuzzy object
  fuzzy->addFuzzyRule(rule1);
  fuzzy->addFuzzyRule(rule2);
  fuzzy->addFuzzyRule(rule3);
  fuzzy->addFuzzyRule(rule4);
}
```

Lampiran 2

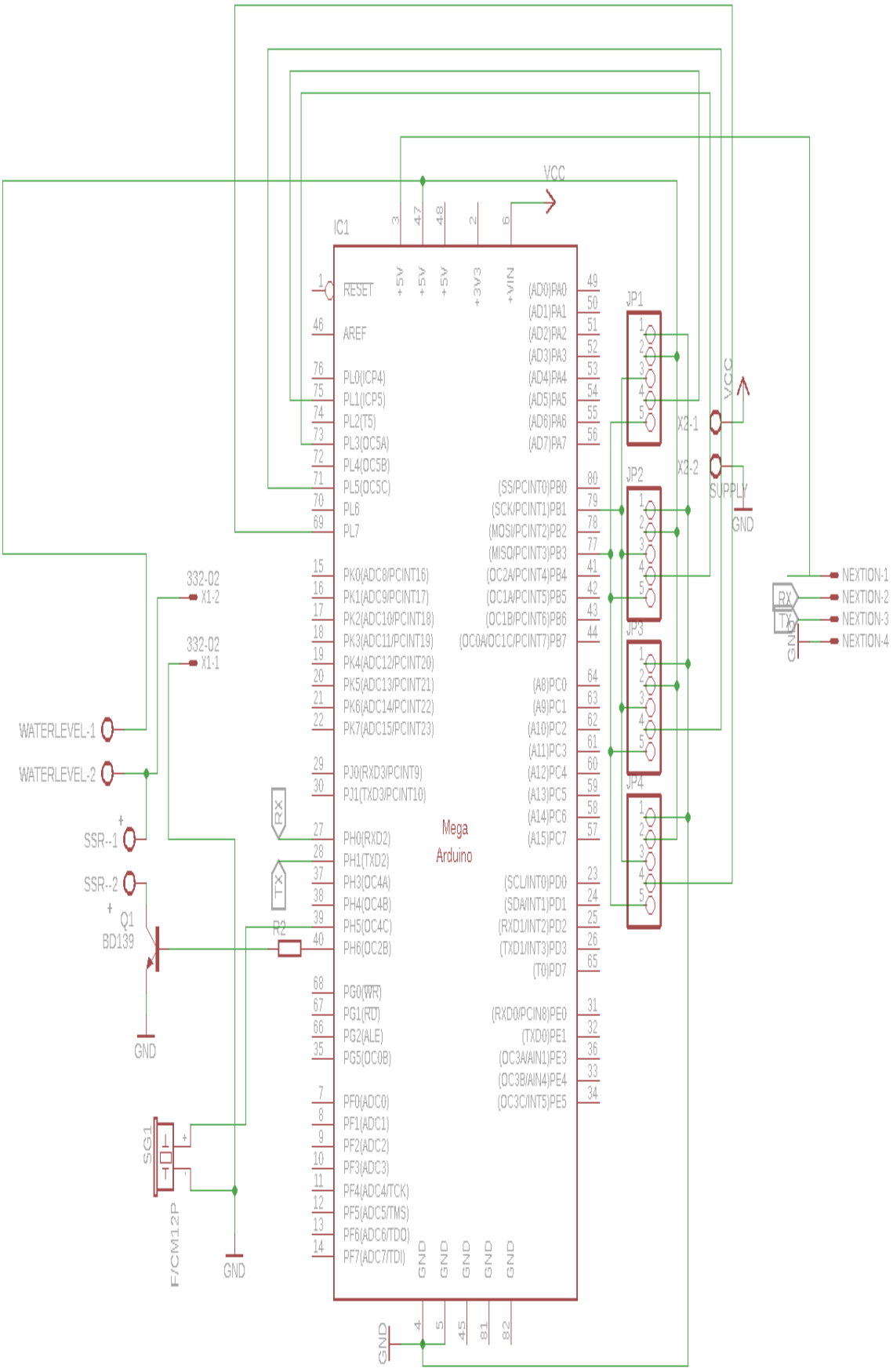
Dokumentasi Alat Waterbath





Proses pengambilan data

Rangkaian Modul Keseluruhan



Spesifikasi Thermometer Digital HTC-2

Digital HTC-2
In/Out Temperature, Humidity Meter,
Date, Clock

Nama	:	Digital thermometer
Merk	:	Aventru
Type	:	HTC-2
Spesifikasi	:	• 1 x Baterai AAA 1.5V • Precision: +1% • Measuring range: -50℃ - 70℃ • Overall size: 48X28.5X16CM • Installation dimension: 46.2X26.7MM • LCD Besar