DAFTAR PUSTAKA

Alfannizar, I., & Rahayu, Y. (2018). Perancangan dan Pembuatan Alat Home Electricity Based Home Appliance Controller Berbasis Internet of Things. *Jom FTEKNIK, 5(1)*, 1-6.

Anshori, M. S., Akbar, S. R., & Maulana, R. (2019). Implementasi Sistem Sensor Dan Aktuator Real Time Pada Tanaman Jamur. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 3(2)*, 1471-1479.

Efendi, Y. (2018). Internet of Things (IoT) Sistem Pengendalian Lampu Menggunakan Raspberry Pi Berbasis Mobile. *Jurnal Ilmiah Ilmu Komputer, 4(1)*, 19-26.

G, A. S. (2019). Teknologi IoT pada Monitoring dan Otomasi Kolam Pembesaran Ikan Lele berbasis Mikrokontroler.

Heryanto, A., Budiarto, J., & Hadi, S. (2020). Sistem Nutrisi Tanaman Hidroponik Berbasis Internet of Things Menggunakan NodeMCU ESP8266. *Jurnal BITe, 2(1)*, 31-39.

Hifdzullisan, J., Sumaryo, S., & Rizal, A. (2018). Desain dan Implementasi Sistem Operasi Waktu Nyata untuk Sistem Tertanam Biomedis Berbasis Penjadwalan. *e-Proceeding of Engineering, 5(3)*, 3851-3859.

Ibadarrohman, Salahuddin, N. S., & Kowanda, A. (2018). Sistem Kontroling dan Monitoring Hidroponik Berbasis Android. *Konferensi Nasional Sistem Informasi*, 177-182.

Jatmiko, W., Mursanto, P., Jati, G., Purnomo, D. J., Alhamidi, M. R., Habibie, N., & Dwinto, K. (2015). *RTOS Teori & Aplikasi.* Depok: Fakultas Ilmu Komputer Universitas Indonesia.

Madan, L., Anand, K., & Bhushan, B. (2014). Real-Time Operating System. *International Journal of Research in Science and Technology*, 39-50.

Newman, P. (2020, Maret 7). *THE INTERNET OF THINGS 2020: Here's what over 400 IoT decision-makers say about the future of enterprise connectivity and how IoT companies can use it to grow revenue*. Retrieved from Business Insider: https://www.businessinsider.com/internet-of-things-report?r=US&IR=T

Purnomo, D. M., Alhamidi, M. R., Jati, G., Habibie, N., Hardjono, B., & Wibisono, A. (2015). Comparative Study of RTOS and Primitive Interrupt in Embedded System. *Jurnal Ilmu Komputer dan Informasi*, 35-44.

Saputro, H. D., Maulana, R., & Ichsan, M. H. (2018). Implementasi Real Timepada Pergerakan Robot Quadruped menggunakan Multisensor dan RTOS. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 2(12)*, 6868-6875.

Septiawan, I. W., Akbar, S. R., & Syauqy, D. (2018). Rancang Bangun Sistem Multi-Sensor untuk Pengukuran Jarak secara Simultan. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 2(11)*, 5394-5401.

Setiadi, D., & Muhaemin, M. A. (2018). Penerapan Internet of Things (IoT) pada Sistem Monitoring Irigasi (Smart Irigasi). *Jurnal Infotoronik Volume 3 No. 2*, 96.

Setiawan, S. (2016). *Teknik Pemrograman dan Multithreading pada Mikrokontroler*. Yogyakarta: Penerbit ANDI.

Sulistiyo, N. T., Erwanto, D., & Rosanti, A. D. (2019). Alat Pengendali Derajat PH pada Sistem Hidroponik Tanaman Pakcoy berbasis Arduino Uno menggunakan Metode PID. *Multitek Indonesia Jurnal Ilmiah Vol. 13 No. 1*, 48.

Suprayitno, E. A., Dijaya, R., & Atho'illah, M. (2018). Otomasi Sistem Hidroponik DFT (Deep Flow Technique) Berbasis Arduino Android dengan Memanfaatkan Panel Surya Sebagai Energi Alternatif. *Elinvo, 3(2)*, 30-37.

Sutaya, I. W. (2015). Peningkatan Kinerja Perangkat Elektronik Berbasis Mikrokontroler AVR 8 Bit dengan Menggunakan RTOS (Real Time Operating System). *Jurnal Pendidikan Teknologi dan Kejuruan, 12(1)*, 11-19.

Yudhaprakosa, P., Akbar, S. R., & Maulana, R. (2019). Sistem Otomasi dan Monitoring Tanaman Hidroponik Berbasis Real Time OS. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 3(4)*, 3285-5293.

LAMPIRAN

Lampiran 1 *Source Code* program tanpa RTOS

```
#include <AntaresESP32MQTT.h>
#define ACCESSKEY       "ACCESS KEY AKUN ANTARES"
#define WIFISSID        "SSID WIFI"
#define PASSWORD        "PASSWORD WIFI"
#define projectName     "NAMA PROJECT ANTARES"
#define deviceName      "NAMA DEVICE ANTARES"

#include <virtuabotixRTC.h>

#include <DHTesp.h>

#include <OneWire.h>
#include <DallasTemperature.h>

#include <Servo.h>
//---------------------------------------------------------------------
AntaresESP32MQTT antares(ACCESSKEY);

virtuabotixRTC rtcKu(14, 12, 13);

int dhtPin = 4;
DHTesp dht;

const int oneWirePin = 25;
OneWire oneWire(oneWirePin);
DallasTemperature sensorSuhuAir(&oneWire);

int tdsPin = 27;
int sensorValue;

const int trigPin = 2;
const int echoPin = 5;
long durasi;
int tinggiEmber = 30;

const int phPin = 35;
int buf[10], temp, phVol, avgValue;

int servoPin = 26;
Servo servo;
int pos = 0;

const int relayPin = 19;

//---------------------------------------------------------------------
int jam;
int suhuUdara, jmlSuhuUdara;
int kelembabanUdara;
int suhuAir, jmlSuhuAir;
int konduktivitasAir, jmlKonduktivitasAir;
int tds, jmlTds;
int tinggiAir, jmlTinggiAir;
```

```
int phAir, jmlPhAir;

int jmlRelay;
int jmlServo;
int jmlKirim;
int jmlTotal;

void setup() {
  Serial.begin(9600);

  Serial.println("CLEARDATA");
  Serial.println("LABEL, Pukul, sU, kU, task1, sA, task2, kA, task3,
tds, task4, tA, task5, pH, task6, task7, task8, kirim, jTot");
  Serial.println("RESETTIMER");

  antares.setDebug(true);
  antares.wifiConnection(WIFISSID, PASSWORD);
  antares.setMqttServer();

  rtcKu.setDS1302Time(00, 30, 10, 6, 18, 12, 2020); // (detik,
menit, jam, hari, tanggal, bulan, tahun)

  dht.setup(dhtPin, DHTesp::DHT11);

  sensorSuhuAir.begin();

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  pinMode(relayPin, OUTPUT);

  servo.attach(servoPin);
}

void loop() {

  Serial.print("DATA, TIME, TIMER,");

  rtcKu.updateTime();
  jam = rtcKu.hours;

  TempAndHumidity lastValues = dht.getTempAndHumidity();
  suhuUdara = lastValues.temperature;
  kelembabanUdara = lastValues.humidity;
  jmlSuhuUdara += 1;
  jmlTotal += 1;

  sensorSuhuAir.requestTemperatures();
  suhuAir = sensorSuhuAir.getTempCByIndex(0);
  jmlSuhuAir += 1;
  jmlTotal += 1;

  sensorValue = analogRead(tdsPin);
  konduktivitasAir = ((0.2142*sensorValue)+494.93);
  jmlKonduktivitasAir += 1;
  jmlTotal += 1;
```

```
sensorValue = analogRead(tdsPin);
tds = (0.3417*sensorValue)+281.08;
jmlTds += 1;
jmlTotal += 1;

digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
durasi = pulseIn(echoPin, HIGH);

tinggiAir = tinggiEmber - (durasi*0.034/2);

jmlTinggiAir += 1;
jmlTotal +=1;

for (int i=0; i<10; i++) {
  buf[i] = analogRead(phPin);
}
for (int i=0; i<9; i++) {
  for (int j=i+1; j<10; j++) {
    if (buf[i] > buf[j]) {
      temp = buf[i];
      buf[i] = buf[j];
      buf[j] = temp;
    }
  }
}
avgValue = 0;
for(int i=2; i<8; i++) {
  avgValue += buf[i];
}
phVol = avgValue/6;
if ( phVol > 0 && phVol < 300) {
  phAir = 14;
} else if ( phVol > 300 && phVol < 600) {
  phAir = 13;
} else if ( phVol > 600 && phVol < 900 ) {
  phAir = 12;
} else if ( phVol > 900 && phVol < 1200 ) {
  phAir = 11;
} else if ( phVol > 1200 && phVol < 1500 ) {
  phAir = 10;
} else if ( phVol > 1500 && phVol < 1800 ) {
  phAir = 9;
} else if ( phVol > 1800 && phVol < 2100 ) {
  phAir = 8;
} else if ( phVol > 2100 && phVol < 2400 ) {
  phAir = 7;
} else if ( phVol > 2400 && phVol < 2700) {
  phAir = 6;
} else if ( phVol > 2700 && phVol < 3000 ) {
  phAir = 5;
} else if ( phVol > 3000 && phVol < 3300) {
  phAir = 4;
```

```
} else if ( phVol > 3300 && phVol < 3600) {
  phAir = 3;
} else if ( phVol > 3600 && phVol < 3900) {
  phAir = 2;
} else if ( phVol > 3900 && phVol < 4096 ) {
  phAir = 1;
}
jmlPhAir += 1;
jmlTotal += 1;

if (rtcKu.hours > 9 && rtcKu.hours < 15) {
  digitalWrite(relayPin, LOW);
  delay(1500);
  digitalWrite(relayPin, HIGH);
  delay(10000);
  jmlRelay += 1;
  jmlTotal += 1;
} else if ( suhuUdara > 35) {
  digitalWrite(relayPin, LOW);
  delay(1500);
  digitalWrite(relayPin, HIGH);
  delay(10000);
  jmlRelay += 1;
  jmlTotal += 1;
} else {
  digitalWrite(relayPin, HIGH);
  delay(10000);
  jmlRelay += 1;
  jmlTotal += 1;
}

if (tds < 1000) {
  for (pos = 0; pos <= 45; pos+=1) {
    servo.write(pos);
    delay(15);
  }
  for (pos = 45; pos >=0; pos -= 1) {
    servo.write(pos);
    delay(15);
  }
  jmlServo += 1;
  jmlTotal += 1;
} else {
  jmlServo += 1;
  jmlTotal += 1;
}

antares.checkMqttConnection();

jmlKirim += 1;
jmlTotal += 1;

antares.add("sU",     suhuUdara);
antares.add("kU",     kelembabanUdara);
antares.add("task1",  jmlSuhuUdara);
antares.add("sA",     suhuAir);
```

```
antares.add("task2",  jmlSuhuAir);
antares.add("kA",     konduktivitasAir);
antares.add("task3",  jmlKonduktivitasAir);
antares.add("tds",    tds);
antares.add("task4",  jmlTds);
antares.add("tA",     tinggiAir);
antares.add("task5",  jmlTinggiAir);
antares.add("ph",     phAir);
antares.add("task6",  jmlPhAir);
antares.add("task7",  jmlRelay);
antares.add("task8",  jmlServo);
antares.add("kirim",  jmlKirim);
antares.add("jTot",   jmlTotal);

antares.publish(projectName, deviceName);

Serial.print(suhuUdara);
Serial.print(",");
Serial.print(kelembabanUdara);
Serial.print(",");
Serial.print(jmlSuhuUdara);
Serial.print(",");
Serial.print(suhuAir);
Serial.print(",");
Serial.print(jmlSuhuAir);
Serial.print(",");
Serial.print(konduktivitasAir);
Serial.print(",");
Serial.print(jmlKonduktivitasAir);
Serial.print(",");
Serial.print(tds);
Serial.print(",");
Serial.print(jmlTds);
Serial.print(",");
Serial.print(tinggiAir);
Serial.print(",");
Serial.print(jmlTinggiAir);
Serial.print(",");
Serial.print(phAir);
Serial.print(",");
Serial.print(jmlPhAir);
Serial.print(",");
Serial.print(jmlRelay);
Serial.print(",");
Serial.print(jmlServo);
Serial.print(",");
Serial.print(jmlKirim);
Serial.print(",");
Serial.println(jmlTotal);

jmlSuhuUdara          = 0;
jmlSuhuAir            = 0;
jmlKonduktivitasAir   = 0;
jmlTds                = 0;
jmlTinggiAir          = 0;
jmlPhAir              = 0;
```

```
  jmlRelay              = 0;
  jmlServo              = 0;
  jmlKirim              = 0;
  jmlTotal              = 0;

  delay(5000);
}
```

Lampiran 2 *Source Code* Program dengan freeRTOS

```
#include <AntaresESP32MQTT.h>
#define ACCESSKEY       "ACCESS KEY AKUN ANTARES"
#define WIFISSID        "SSID WIFI"
#define PASSWORD        "PASSWORD WIFI"
#define projectName     "NAMA PROJECT ANTARES"
#define deviceName      "NAMA DEVICE ANTARES"

#include <virtuabotixRTC.h>

#include <DHTesp.h>

#include <OneWire.h>
#include <DallasTemperature.h>

#include <Servo.h>
//---------------------------------------------------------------------
AntaresESP32MQTT antares(ACCESSKEY);

virtuabotixRTC rtcKu(14, 12, 13);

int dhtPin = 4;
DHTesp dht;

const int oneWirePin = 25;
OneWire oneWire(oneWirePin);
DallasTemperature sensorSuhuAir(&oneWire);

int tdsPin = 27;
int sensorValue;

const int trigPin = 2;
const int echoPin = 5;
long durasi;
int tinggiEmber = 30;

const int phPin = 35;
int buf[10], temp, phVol, avgValue;

int servoPin = 26;
Servo servo;
int pos = 0;

const int relayPin = 19;

//---------------------------------------------------------------------
int jam;
int suhuUdara, jmlSuhuUdara;
int kelembabanUdara;
int suhuAir, jmlSuhuAir;
int konduktivitasAir, jmlKonduktivitasAir;
int tds, jmlTds;
int tinggiAir, jmlTinggiAir;
int phAir, jmlPhAir;
int jmlRelay;
```

```
int jmlServo;
int jmlKirim;
int jmlTotal;

//----------------------------------------------------------------
void TaskSuhuUdara();
void TaskSuhuAir();
void TaskKonduktivitasAir();
void TaskTds();
void TaskTinggiAir();
void TaskPhAir();
void TaskRelay();
void TaskServo();
void TaskKirim();

void setup() {
  Serial.begin(9600);

  Serial.println("CLEARDATA");
  Serial.println("LABEL, Pukul, sU, kU, task1, sA, task2, kA, task3,
tds, task4, tA, task5, pH, task6, task7, task8, kirim, jTot");
  Serial.println("RESETTIMER");

  antares.setDebug(true);
  antares.wifiConnection(WIFISSID, PASSWORD);
  antares.setMqttServer();

  rtcKu.setDS1302Time(00, 55, 12, 6, 18, 12, 2020); // (detik,
menit, jam, hari, tanggal, bulan, tahun)

  dht.setup(dhtPin, DHTesp::DHT11);

  sensorSuhuAir.begin();

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  pinMode(relayPin, OUTPUT);

  servo.attach(servoPin);

  //TASK
  xTaskCreate(
    TaskSuhuUdara,
    "TaskSuhuUdara",
    10000,
    NULL,
    1,
    NULL);

  xTaskCreate(
    TaskSuhuAir,
    "TaskSuhuAir",
    10000,
    NULL,
    1,
```

```
  NULL);

xTaskCreate(
  TaskKonduktivitasAir,
  "TaskKonduktivitasAir",
  10000,
  NULL,
  1,
  NULL);

xTaskCreate(
  TaskTds,
  "TaskTds",
  10000,
  NULL,
  1,
  NULL);

xTaskCreate(
  TaskTinggiAir,
  "TaskTinggiAir",
  10000,
  NULL,
  1,
  NULL);

xTaskCreate(
  TaskPhAir,
  "TaskPhAir",
  10000,
  NULL,
  1,
  NULL);

xTaskCreate(
  TaskRelay,
  "TaskRelay",
  10000,
  NULL,
  1,
  NULL);

xTaskCreate(
  TaskServo,
  "TaskServo",
  10000,
  NULL,
  1,
  NULL);

xTaskCreate(
  TaskKirim,
  "TaskKirim",
  10000,
  NULL,
  2,
```

```
    NULL);
}

void loop() {
  rtcKu.updateTime();
  jam = rtcKu.hours;
}

void TaskSuhuUdara (void *pvParameters) {
  TickType_t xLastWakeTime;
  const TickType_t xDelaySuhuUdara = 1000/portTICK_PERIOD_MS;

  for(int i;;) {
    vTaskDelayUntil(&xLastWakeTime, xDelaySuhuUdara);
    TempAndHumidity lastValue = dht.getTempAndHumidity();
    suhuUdara = lastValue.temperature;
    kelembabanUdara = lastValue.humidity;
    jmlSuhuUdara += 1;
    jmlTotal += 1;
  }
}

void TaskSuhuAir (void *pvParameters) {
  TickType_t xLastWakeTime;
  const TickType_t xDelaySuhuAir = 2000/portTICK_PERIOD_MS;

  for(int i;;){
    vTaskDelayUntil(&xLastWakeTime, xDelaySuhuAir);
    sensorSuhuAir.requestTemperatures();
    suhuAir = sensorSuhuAir.getTempCByIndex(0);
    jmlSuhuAir += 1;
    jmlTotal += 1;
  }
}

void TaskKonduktivitasAir (void *pvParameters) {
  TickType_t xLastWakeTime;
  const TickType_t xDelayKonduk = 1000/portTICK_PERIOD_MS;

  for(int i;;){
    vTaskDelayUntil(&xLastWakeTime, xDelayKonduk);
    sensorValue = analogRead(tdsPin);
    konduktivitasAir = ((0.2142*sensorValue)+494.93);
    jmlKonduktivitasAir += 1;
    jmlTotal += 1;
  }
}

void TaskTds (void *pvParameters) {
  TickType_t xLastWakeTime;
  const TickType_t xDelayTds = 1000 / portTICK_PERIOD_MS;

  for(int i;;) {
    vTaskDelayUntil(&xLastWakeTime, xDelayTds);
    sensorValue = analogRead(tdsPin);
    tds = (0.3417*sensorValue)+281.08;
```

```
    jmlTds += 1;
    jmlTotal += 1;
  }
}

void TaskTinggiAir (void *pvParameters) {
  TickType_t xLastWakeTime;
  const TickType_t xDelayTinggi = 1000 / portTICK_PERIOD_MS;
  const TickType_t xDelayLow = 2;
  const TickType_t xDelayHigh = 10;

  for(int i;;) {
    vTaskDelayUntil(&xLastWakeTime, xDelayTinggi);
    digitalWrite(trigPin, LOW);
    vTaskDelay(xDelayLow);
    digitalWrite(trigPin, HIGH);
    vTaskDelay(xDelayHigh);
    digitalWrite(trigPin, LOW);
    durasi = pulseIn(echoPin, HIGH);

    tinggiAir = tinggiEmber - ((durasi*0.034)/2);

    jmlTinggiAir += 1;
    jmlTotal += 1;
  }
}

void TaskPhAir (void *pvParameters) {
  TickType_t xLastWakeTime;
  const TickType_t xDelayPh = 1000 / portTICK_PERIOD_MS;

  for(int i;;) {
    vTaskDelayUntil(&xLastWakeTime, xDelayPh);
    for (int i=0; i<10; i++) {
      buf[i] = analogRead(phPin);
    }
    for (int i=0; i<9; i++) {
      for (int j=i+1; j<10; j++) {
        if (buf[i] > buf[j]) {
          temp = buf[i];
          buf[i] = buf[j];
          buf[j] = temp;
        }
      }
    }
    avgValue = 0;
    for(int i=2; i<8; i++) {
      avgValue += buf[i];
    }
    phVol = avgValue/6;
    if ( phVol > 0 && phVol < 300) {
      phAir = 14;
    } else if ( phVol > 300 && phVol < 600) {
      phAir = 13;
    } else if ( phVol > 600 && phVol < 900 ) {
      phAir = 12;
```

```
    } else if ( phVol > 900 && phVol < 1200 ) {
      phAir = 11;
    } else if ( phVol > 1200 && phVol < 1500 ) {
      phAir = 10;
    } else if ( phVol > 1500 && phVol < 1800 ) {
      phAir = 9;
    } else if ( phVol > 1800 && phVol < 2100 ) {
      phAir = 8;
    } else if ( phVol > 2100 && phVol < 2400 ) {
      phAir = 7;
    } else if ( phVol > 2400 && phVol < 2700) {
      phAir = 6;
    } else if ( phVol > 2700 && phVol < 3000 ) {
      phAir = 5;
    } else if ( phVol > 3000 && phVol < 3300) {
      phAir = 4;
    } else if ( phVol > 3300 && phVol < 3600) {
      phAir = 3;
    } else if ( phVol > 3600 && phVol < 3900) {
      phAir = 2;
    } else if ( phVol > 3900 && phVol < 4096 ) {
      phAir = 1;
    }

    jmlPhAir += 1;
    jmlTotal += 1;
  }
}

void TaskRelay (void *pvParameters) {
  const TickType_t xRelayOff = 15000/portTICK_PERIOD_MS;
  const TickType_t xRelayOn = 1500/portTICK_PERIOD_MS;

  for(int i;;) {
    if (rtcKu.hours > 9 && rtcKu.hours < 15) {
      digitalWrite(relayPin, LOW);
      vTaskDelay(xRelayOn);
      digitalWrite(relayPin, HIGH);
      vTaskDelay(xRelayOff);
    } else if ( suhuUdara > 35) {
      digitalWrite(relayPin, LOW);
      vTaskDelay(xRelayOn);
      digitalWrite(relayPin, HIGH);
      vTaskDelay(xRelayOff);
    } else {
      digitalWrite(relayPin, HIGH);
      vTaskDelay(xRelayOff);
    }
    jmlRelay += 1;
    jmlTotal += 1;
  }
}

void TaskServo (void *pvParameters) {
  TickType_t xLastWakeTime;
  const TickType_t xDelayServo = 1000 / portTICK_PERIOD_MS;
```

```
  const TickType_t xWriteServo = 15/portTICK_PERIOD_MS;

  for(int i;;) {
    vTaskDelayUntil(&xLastWakeTime, xDelayServo);
    if (tds < 1000) {
      for (pos = 0; pos <= 45; pos+=1) {
        servo.write(pos);
        vTaskDelay(xWriteServo);
      }
      for (pos = 45; pos >=0; pos -= 1) {
        servo.write(pos);
        vTaskDelay(xWriteServo);
      }
    } else {

    }
    jmlServo += 1;
    jmlTotal += 1;
  }
}

void TaskKirim (void *pvParameters) {
  TickType_t xLastWakeTime;
  const TickType_t xDelay = 5000/portTICK_PERIOD_MS;

  for (int i;;) {
    vTaskDelayUntil(&xLastWakeTime, xDelay);

    Serial.print("DATA, TIME, TIMER,");

    antares.checkMqttConnection();

    jmlKirim += 1;
    jmlTotal += 1;

    antares.add("sU",     suhuUdara);
    antares.add("kU",     kelembabanUdara);
    antares.add("task1",  jmlSuhuUdara);
    antares.add("sA",     suhuAir);
    antares.add("task2",  jmlSuhuAir);
    antares.add("kA",     konduktivitasAir);
    antares.add("task3",  jmlKonduktivitasAir);
    antares.add("tds",    tds);
    antares.add("task4",  jmlTds);
    antares.add("tA",     tinggiAir);
    antares.add("task5",  jmlTinggiAir);
    antares.add("ph",     phAir);
    antares.add("task6",  jmlPhAir);
    antares.add("task7",  jmlRelay);
    antares.add("task8",  jmlServo);
    antares.add("jkirim", jmlKirim);
    antares.add("jTot",   jmlTotal);

    antares.publish(projectName, deviceName);

    Serial.print(suhuUdara);
```

```
    Serial.print(",");
    Serial.print(kelembabanUdara);
    Serial.print(",");
    Serial.print(jmlSuhuUdara);
    Serial.print(",");
    Serial.print(suhuAir);
    Serial.print(",");
    Serial.print(jmlSuhuAir);
    Serial.print(",");
    Serial.print(konduktivitasAir);
    Serial.print(",");
    Serial.print(jmlKonduktivitasAir);
    Serial.print(",");
    Serial.print(tds);
    Serial.print(",");
    Serial.print(jmlTds);
    Serial.print(",");
    Serial.print(tinggiAir);
    Serial.print(",");
    Serial.print(jmlTinggiAir);
    Serial.print(",");
    Serial.print(phAir);
    Serial.print(",");
    Serial.print(jmlPhAir);
    Serial.print(",");
    Serial.print(jmlRelay);
    Serial.print(",");
    Serial.print(jmlServo);
    Serial.print(",");
    Serial.print(jmlKirim);
    Serial.print(",");
    Serial.println(jmlTotal);

    jmlSuhuUdara        = 0;
    jmlSuhuAir          = 0;
    jmlKonduktivitasAir = 0;
    jmlTds              = 0;
    jmlTinggiAir        = 0;
    jmlPhAir            = 0;
    jmlRelay            = 0;
    jmlServo            = 0;
    jmlKirim            = 0;
    jmlTotal            = 0;
  }
}
```

Lampiran 3 *Source code* program dengan TridentTD_EasyFreeRTOS32

Source code utama

```
#include <TridentTD_EasyFreeRTOS32.h>

#include <AntaresESP32MQTT.h>
#define ACCESSKEY "ACCESS KEY AKUN ANTARES"
#define WIFISSID  "SSID WIFI"
#define PASSWORD  "PASSWORD WIFI"

#include <virtuabotixRTC.h>

AntaresESP32MQTT antares(ACCESSKEY);

virtuabotixRTC rtcKu(14, 12, 13);

TridentOS   task1, task2, task3, task4, task5, task6, task7, task8,
task9;
void  task1_func(void*), task2_func(void*),
      task3_func(void*), task4_func(void*),
      task5_func(void*), task6_func(void*),
      task7_func(void*), task8_func(void*),
      task9_func(void*);

int xJam;
int xSuhuUdara, xJmlSuhuUdara;
int xKelembabanUdara;
int xSuhuAir, xJmlSuhuAir;
int xKonduktivitasAir, xJmlKonduktivitasAir;
int xTds, xJmlTds;
int xTinggiAir, xJmlTinggiAir;
int xPhAir, xJmlPhAir;
int xJmlRelay;
int xJmlServo;
int xJmlKirim;
int xJmlTotal;

long previousMillis = 0;
long interval = 1000;

//-------------------------------------------------------

void setup(){
  Serial.begin(9600);

  Serial.println("CLEARDATA");
  Serial.println("LABEL, Pukul, sU, kU, task1, sA, task2, kA, task3,
tds, task4, tA, task5, pH, task6, task7, task8, kirim, jTot");
  Serial.println("RESETTIMER");

  antares.setDebug(true);
  antares.wifiConnection(WIFISSID, PASSWORD);
  antares.setMqttServer();
```

```
  rtcKu.setDS1302Time(00, 43, 16, 6, 18, 12, 2020); // (detik,
menit, jam, hari, tanggal, bulan, tahun)

  task1.start( task1_func );
  task2.start( task2_func );
  task3.start( task3_func );
  task4.start( task4_func );
  task5.start( task5_func );
  task6.start( task6_func );
  task7.start( task7_func );
  task8.start( task8_func );
  task9.start( task9_func );
}

void loop(){}
```

Source code Task1

```
#include <DHTesp.h>

#define TASK1_INTERVAL  1000

int dhtPin = 4;
DHTesp dht;

void task1_func(void*)  {

  VOID SETUP() {
    dht.setup(dhtPin, DHTesp::DHT11);
  }

  VOID LOOP() {
    TempAndHumidity lastValues = dht.getTempAndHumidity();
    xSuhuUdara = lastValues.temperature;
    xKelembabanUdara = lastValues.humidity;
    xJmlSuhuUdara += 1;
    xJmlTotal += 1;

    DELAY(TASK1_INTERVAL);
  }
}
```

Source code Task2

```
#include <OneWire.h>
#include <DallasTemperature.h>

const int oneWirePin = 25;
OneWire oneWire(oneWirePin);
DallasTemperature sensorSuhuAir(&oneWire);

void task2_func(void*)
```

```
  VOID SETUP()
    sensorSuhuAir.begin();
  }

  VOID LOOP()
    sensorSuhuAir.requestTemperatures();
    xSuhuAir = sensorSuhuAir.getTempCByIndex(0);
    xJmlSuhuAir += 1;
    xJmlTotal += 1;

    DELAY(2000);
  }
}
```

Source code Task3

```
void task3_func(void*)  {

  VOID SETUP() {
  }

  VOID LOOP() {
    sensorValue = analogRead(tdsPin);
    xKonduktivitasAir = ((0.2142*sensorValue)+494.93);
    xJmlKonduktivitasAir += 1;
    xJmlTotal += 1;

    DELAY(1000);
  }
}
```

Source code Task4

```
void task4_func(void*)  {
  int tdsPin = 27;
  int sensorValue;

  VOID SETUP() {
  }

  VOID LOOP() {
    sensorValue = analogRead(tdsPin);
    xTds = (0.3417*sensorValue)+281.08;
    xJmlTds += 1;
    xJmlTotal += 1;

    DELAY(1000);
  }
}
```

Source code Task5

```
const int trigPin = 2;
const int echoPin = 5;

void task5_func(void*)  {
  long durasi;
  int tinggiEmber = 30;

  VOID SETUP() {
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
  }

  VOID LOOP() {
    digitalWrite(trigPin, LOW);
    DELAY(2);
    digitalWrite(trigPin, HIGH);
    DELAY(10);
    digitalWrite(trigPin, LOW);
    durasi = pulseIn(echoPin, HIGH);

    xTinggiAir = tinggiEmber - (durasi*0.034/2);
    xJmlTinggiAir += 1;
    xJmlTotal += 1;

    DELAY(1000);
  }
}
```

Source code Task6

```
const int phPin = 35;

void task6_func(void*)  {
  int buf[10], temp, phVol, avgValue;

  VOID SETUP() {
  }

  VOID LOOP() {
    for (int i=0; i<10; i++) {
      buf[i] = analogRead(phPin);
    }
    for (int i=0; i<9; i++) {
      for (int j=i+1; j<10; j++) {
        if (buf[i] > buf[j]) {
          temp = buf[i];
          buf[i] = buf[j];
          buf[j] = temp;
        }
      }
    }
```

```
    avgValue = 0;
    for(int i=2; i<8; i++) {
      avgValue += buf[i];
    }
    phVol = avgValue/6;
    if ( phVol > 0 && phVol < 300) {
      xPhAir = 14;
    } else if ( phVol > 300 && phVol < 600) {
      xPhAir = 13;
    } else if ( phVol > 600 && phVol < 900 ) {
      xPhAir = 12;
    } else if ( phVol > 900 && phVol < 1200 ) {
      xPhAir = 11;
    } else if ( phVol > 1200 && phVol < 1500 ) {
      xPhAir = 10;
    } else if ( phVol > 1500 && phVol < 1800 ) {
      xPhAir = 9;
    } else if ( phVol > 1800 && phVol < 2100 ) {
      xPhAir = 8;
    } else if ( phVol > 2100 && phVol < 2400 ) {
      xPhAir = 7;
    } else if ( phVol > 2400 && phVol < 2700) {
      xPhAir = 6;
    } else if ( phVol > 2700 && phVol < 3000 ) {
      xPhAir = 5;
    } else if ( phVol > 3000 && phVol < 3300) {
      xPhAir = 4;
    } else if ( phVol > 3300 && phVol < 3600) {
      xPhAir = 3;
    } else if ( phVol > 3600 && phVol < 3900) {
      xPhAir = 2;
    } else if ( phVol > 3900 && phVol < 4096 ) {
      xPhAir = 1;
    }
    xJmlPhAir += 1;
    xJmlTotal += 1;

    DELAY(1000);
  }
}
```

Source code Task7

```
void task7_func(void*)  {
  const int relayPin = 19;

  VOID SETUP() {
    pinMode(relayPin, OUTPUT);
  }

  VOID LOOP() {
    unsigned long currentMillis = millis();
    rtcKu.updateTime();

    if (currentMillis - previousMillis > interval) {
```

```
      previousMillis = currentMillis;
      xJam = rtcKu.hours;
    }

    if (rtcKu.hours > 9 && rtcKu.hours < 15) {
      digitalWrite(relayPin, LOW);
      DELAY(1500);
      digitalWrite(relayPin, HIGH);
      DELAY(15000);
    } else if ( xSuhuUdara > 35) {
      digitalWrite(relayPin, LOW);
      DELAY(1500);
      digitalWrite(relayPin, HIGH);
      DELAY(15000);
    } else {
      digitalWrite(relayPin, HIGH);
      DELAY(15000);
    }
    xJmlRelay += 1;
    xJmlTotal += 1;
  }
}
```

Source code Task8

```
#include <Servo.h>

void task8_func(void*)  {
  int servoPin = 26;
  Servo servo;
  int pos = 0;

  VOID SETUP() {
    servo.attach(servoPin);
  }

  VOID LOOP() {
    if (xTds < 1000) {
      for (pos = 0; pos <= 45; pos+=1) {
        servo.write(pos);
        DELAY(15);
      }
      for (pos = 45; pos >=0; pos -= 1) {
        servo.write(pos);
        DELAY(15);
      }
    } else {
    }
    xJmlServo += 1;
    xJmlTotal += 1;

    DELAY(1000);
  }
}
```

Source code Task9

```
#define projectName     "NAMA PROJECT ANTARES"
#define deviceName      "NAMA DEVICE ANTARES"

void task9_func(void*)  {
  VOID SETUP() {
  }

  VOID LOOP() {
    Serial.print("DATA, TIME, TIMER,");

    antares.checkMqttConnection();

    xJmlKirim += 1;
    xJmlTotal += 1;

    antares.add("sU",   xSuhuUdara);
    antares.add("kU",   xKelembabanUdara);
    antares.add("task1",xJmlSuhuUdara);
    antares.add("sA",   xSuhuAir);
    antares.add("task2",xJmlSuhuAir);
    antares.add("kA",   xKonduktivitasAir);
    antares.add("task3",xJmlKonduktivitasAir);
    antares.add("tds",  xTds);
    antares.add("task4",xJmlTds);
    antares.add("tA",   xTinggiAir);
    antares.add("task5",xJmlTinggiAir);
    antares.add("ph",   xPhAir);
    antares.add("task6",xJmlPhAir);
    antares.add("task7",xJmlRelay);
    antares.add("task8",xJmlServo);
    antares.add("kirim",xJmlKirim);
    antares.add("jTot", xJmlTotal);

    antares.publish(projectName, deviceName);

    Serial.print(xSuhuUdara);
    Serial.print(",");
    Serial.print(xKelembabanUdara);
    Serial.print(",");
    Serial.print(xJmlSuhuUdara);
    Serial.print(",");
    Serial.print(xSuhuAir);
    Serial.print(",");
    Serial.print(xJmlSuhuAir);
    Serial.print(",");
    Serial.print(xKonduktivitasAir);
    Serial.print(",");
    Serial.print(xJmlKonduktivitasAir);
    Serial.print(",");
    Serial.print(xTds);
    Serial.print(",");
    Serial.print(xJmlTds);
    Serial.print(",");
```

```
    Serial.print(xTinggiAir);
    Serial.print(",");
    Serial.print(xJmlTinggiAir);
    Serial.print(",");
    Serial.print(xPhAir);
    Serial.print(",");
    Serial.print(xJmlPhAir);
    Serial.print(",");
    Serial.print(xJmlRelay);
    Serial.print(",");
    Serial.print(xJmlServo);
    Serial.print(",");
    Serial.print(xJmlKirim);
    Serial.print(",");
    Serial.println(xJmlTotal);

    xJmlSuhuUdara       = 0;
    xJmlSuhuAir         = 0;
    xJmlKonduktivitasAir= 0;
    xJmlTds             = 0;
    xJmlTinggiAir       = 0;
    xJmlPhAir           = 0;
    xJmlRelay           = 0;
    xJmlServo           = 0;
    xJmlKirim           = 0;
    xJmlTotal           = 0;

    DELAY(5000);
  }
}
```

Lampiran 4 *Source Code* program 3 *task* freeRTOS

```
#include <AntaresESP32MQTT.h>
#define ACCESSKEY       "ACCESS KEY AKUN ANTARES"
#define WIFISSID        "SSID WIFI"
#define PASSWORD        "PASSWORD WIFI"
#define projectName     "NAMA PROJECT ANTARES"
#define deviceName      "NAMA DEVICE ANTARES"

#include <virtuabotixRTC.h>

#include <DHTesp.h>

#include <OneWire.h>
#include <DallasTemperature.h>

#include <Servo.h>
//---------------------------------------------------------------------
AntaresESP32MQTT antares(ACCESSKEY);

virtuabotixRTC rtcKu(14, 12, 13);

int dhtPin = 4;
DHTesp dht;

const int oneWirePin = 25;
OneWire oneWire(oneWirePin);
DallasTemperature sensorSuhuAir(&oneWire);

int tdsPin = 27;
int sensorValue;

const int trigPin = 2;
const int echoPin = 5;
long durasi;
int tinggiEmber = 30;

const int phPin = 35;
int buf[10], temp, phVol, avgValue;

int servoPin = 26;
Servo servo;
int pos = 0;

const int relayPin = 19;

//---------------------------------------------------------------------
int jam;
int suhuUdara;
int kelembabanUdara;
int suhuAir;
int konduktivitasAir;
int tds;
int tinggiAir;
int phAir;
int jmlTask1;
```

```
int jmlTask2;
int jmlKirim;
int jmlTotal;

long previousMillis = 0;
long interval = 1000;

//---------------------------------------------------------------------
void Task1();
void Task2();
void TaskKirim();

void setup() {
  Serial.begin(9600);

  Serial.println("CLEARDATA");
  Serial.println("LABEL, Pukul, sU, kU, sA, kA, tds, task1, tA,
pH, task2, kirim, jTot");
  Serial.println("RESETTIMER");

  antares.setDebug(true);
  antares.wifiConnection(WIFISSID, PASSWORD);
  antares.setMqttServer();

  rtcKu.setDS1302Time(00, 52, 17, 6, 18, 12, 2020); // (detik,
menit, jam, hari, tanggal, bulan, tahun)

  dht.setup(dhtPin, DHTesp::DHT11);

  sensorSuhuAir.begin();

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  pinMode(relayPin, OUTPUT);

  servo.attach(servoPin);

  //TASK
  xTaskCreate(
    Task1,
    "Task1",
    10000,
    NULL,
    1,
    NULL);

  xTaskCreate(
    Task2,
    "Task2",
    10000,
    NULL,
    1,
    NULL);

  xTaskCreate(
```

```
    TaskKirim,
    "TaskKirim",
    10000,
    NULL,
    2,
    NULL);
}

void loop() {
  unsigned long currentMillis = millis();
  rtcKu.updateTime();

  if (currentMillis - previousMillis > interval) {
    previousMillis = currentMillis;
    jam = rtcKu.hours;
  }
}

void Task1(void *pvParameters) {
  const TickType_t xDelayTask1 = 2000/portTICK_PERIOD_MS;

  for (int i;;) {
    TempAndHumidity lastValues = dht.getTempAndHumidity();
    suhuUdara = lastValues.temperature;
    kelembabanUdara = lastValues.humidity;

    sensorSuhuAir.requestTemperatures();
    suhuAir = sensorSuhuAir.getTempCByIndex(0);

    sensorValue = analogRead(tdsPin);
    konduktivitasAir = ((0.2142*sensorValue)+494.93);

    sensorValue = analogRead(tdsPin);
    tds = (0.3417*sensorValue)+281.08;

    jmlTask1 += 1;
    jmlTotal += 1;
    vTaskDelay(xDelayTask1);
  }
}

void Task2(void *pvParameters) {

  const TickType_t xDelayTask2 = 1000/portTICK_PERIOD_MS;
  const TickType_t xDelayTrig = 2;
  const TickType_t xDelayTrigN = 10;
  const TickType_t xRelayOff = 15000/portTICK_PERIOD_MS;
  const TickType_t xRelayOn = 1500/portTICK_PERIOD_MS;

  for (int i;;) {
    digitalWrite(trigPin, LOW);
    vTaskDelay(xDelayTrig);
    digitalWrite(trigPin, HIGH);
    vTaskDelay(xDelayTrigN);
    digitalWrite(trigPin, LOW);
    durasi = pulseIn(echoPin, HIGH);
```

```
tinggiAir = tinggiEmber - (durasi*0.034/2);

for (int i=0; i<10; i++) {
  buf[i] = analogRead(phPin);
}
for (int i=0; i<9; i++) {
  for (int j=i+1; j<10; j++) {
    if (buf[i] > buf[j]) {
      temp = buf[i];
      buf[i] = buf[j];
      buf[j] = temp;
    }
  }
}
avgValue = 0;
for(int i=2; i<8; i++) {
  avgValue += buf[i];
}
phVol = avgValue/6;
if ( phVol > 0 && phVol < 300) {
  phAir = 14;
} else if ( phVol > 300 && phVol < 600) {
  phAir = 13;
} else if ( phVol > 600 && phVol < 900 ) {
  phAir = 12;
} else if ( phVol > 900 && phVol < 1200 ) {
  phAir = 11;
} else if ( phVol > 1200 && phVol < 1500 ) {
  phAir = 10;
} else if ( phVol > 1500 && phVol < 1800 ) {
  phAir = 9;
} else if ( phVol > 1800 && phVol < 2100 ) {
  phAir = 8;
} else if ( phVol > 2100 && phVol < 2400 ) {
  phAir = 7;
} else if ( phVol > 2400 && phVol < 2700) {
  phAir = 6;
} else if ( phVol > 2700 && phVol < 3000 ) {
  phAir = 5;
} else if ( phVol > 3000 && phVol < 3300) {
  phAir = 4;
} else if ( phVol > 3300 && phVol < 3600) {
  phAir = 3;
} else if ( phVol > 3600 && phVol < 3900) {
  phAir = 2;
} else if ( phVol > 3900 && phVol < 4096 ) {
  phAir = 1;
}

if (rtcKu.hours > 9 && rtcKu.hours < 15) {
  digitalWrite(relayPin, LOW);
  vTaskDelay(xRelayOn);
  digitalWrite(relayPin, HIGH);
  vTaskDelay(xRelayOff);
} else if ( suhuUdara > 35) {
  digitalWrite(relayPin, LOW);
```

```
      vTaskDelay(xRelayOn);
      digitalWrite(relayPin, HIGH);
      vTaskDelay(xRelayOff);
    } else {
      digitalWrite(relayPin, HIGH);
      vTaskDelay(xRelayOff);
    }

    if (tds < 1000) {
      for (pos = 0; pos <= 45; pos+=1) {
        servo.write(pos);
        delay(15);
      }
      for (pos = 45; pos >=0; pos -= 1) {
        servo.write(pos);
        delay(15);
      }
    } else {
    }

    jmlTask2 += 1;
    jmlTotal += 1;
    vTaskDelay(xDelayTask2);
  }
}

void TaskKirim(void *pvParameters) {
  TickType_t xLastWakeTime;
  const TickType_t xDelay = 5000/portTICK_PERIOD_MS;

  for (int i;;) {

    vTaskDelayUntil(&xLastWakeTime, xDelay);

    Serial.print("DATA, TIME, TIMER,");

    antares.checkMqttConnection();

    jmlKirim += 1;
    jmlTotal += 1;

    antares.add("sU",       suhuUdara);
    antares.add("kU",       kelembabanUdara);
    antares.add("sA",       suhuAir);
    antares.add("kA",       konduktivitasAir);
    antares.add("tds",      tds);
    antares.add("jmlTask1", jmlTask1);

    antares.add("tA",       tinggiAir);
    antares.add("ph",       phAir);
    antares.add("jmlTask2", jmlTask2);

    antares.add("jk",       jmlKirim);
    antares.add("jTot",     jmlTotal);

    antares.publish(projectName, deviceName);
```

```
    Serial.print(suhuUdara);
    Serial.print(",");
    Serial.print(kelembabanUdara);
    Serial.print(",");
    Serial.print(suhuAir);
    Serial.print(",");
    Serial.print(konduktivitasAir);
    Serial.print(",");
    Serial.print(tds);
    Serial.print(",");
    Serial.print(jmlTask1);
    Serial.print(",");
    Serial.print(tinggiAir);
    Serial.print(",");
    Serial.print(phAir);
    Serial.print(",");
    Serial.print(jmlTask2);
    Serial.print(",");
    Serial.print(jmlKirim);
    Serial.print(",");
    Serial.println(jmlTotal);

    jmlTask1 = 0;
    jmlTask2 = 0;
    jmlKirim = 0;
    jmlTotal = 0;
  }
}
```

Lampiran 5 *Source Code* program 5 *task* freeRTOS

```
#include <AntaresESP32MQTT.h>
#define ACCESSKEY       "ACCESS KEY AKUN ANTARES"
#define WIFISSID        "SSID WIFI"
#define PASSWORD        "PASSWORD WIFI"
#define projectName     "NAMA PROJECT ANTARES"
#define deviceName      "NAMA DEVICE ANTARES"

#include <virtuabotixRTC.h>

#include <DHTesp.h>

#include <OneWire.h>
#include <DallasTemperature.h>

#include <Servo.h>
//---------------------------------------------------------------------
AntaresESP32MQTT antares(ACCESSKEY);

virtuabotixRTC rtcKu(14, 12, 13);

int dhtPin = 4;
DHTesp dht;

const int oneWirePin = 25;
OneWire oneWire(oneWirePin);
DallasTemperature sensorSuhuAir(&oneWire);

int tdsPin = 27;
int sensorValue;

const int trigPin = 2;
const int echoPin = 5;
long durasi;
int tinggiEmber = 30;

const int phPin = 35;
int buf[10], temp, phVol, avgValue;

int servoPin = 26;
Servo servo;
int pos = 0;

const int relayPin = 19;

//---------------------------------------------------------------------
int jam;
int suhuUdara;
int kelembabanUdara;
int suhuAir;
int konduktivitasAir;
int tds;
int tinggiAir;
int phAir;
int jmlTask1;
```

```
int jmlTask2;
int jmlTask3;
int jmlTask4;
int jmlKirim;
int jmlTotal;

long previousMillis = 0;
long interval = 1000;

//---------------------------------------------------------------
void Task1();
void Task2();
void Task3();
void Task4();
void TaskKirim();

void setup() {
  Serial.begin(9600);

  Serial.println("CLEARDATA");
  Serial.println("LABEL, Pukul, sU, kU, sA, task1, kA, tds, task2,
tA, pH, task3, task4, kirim, jTot");
  Serial.println("RESETTIMER");

  antares.setDebug(true);
  antares.wifiConnection(WIFISSID, PASSWORD);
  antares.setMqttServer();

  rtcKu.setDS1302Time(00, 8, 11, 6, 12, 11, 2020); // (detik,
menit, jam, hari, tanggal, bulan, tahun)

  dht.setup(dhtPin, DHTesp::DHT11);

  sensorSuhuAir.begin();

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  pinMode(relayPin, OUTPUT);

  servo.attach(servoPin);

  //TASK
  xTaskCreate(
    Task1,
    "Task1",
    10000,
    NULL,
    1,
    NULL);

  xTaskCreate(
    Task2,
    "Task2",
    10000,
    NULL,
```

```
    1,
    NULL);

  xTaskCreate(
    Task3,
    "Task3",
    10000,
    NULL,
    1,
    NULL);

  xTaskCreate(
    Task4,
    "Task4",
    10000,
    NULL,
    1,
    NULL);

  xTaskCreate(
    TaskKirim,
    "TaskKirim",
    10000,
    NULL,
    2,
    NULL);
}

void loop() {
  unsigned long currentMillis = millis();
  rtcKu.updateTime();

  if (currentMillis - previousMillis > interval) {
    previousMillis = currentMillis;
    jam = rtcKu.hours;
  }
}

void Task1(void *pvParameters) {
  const TickType_t xDelayTask1 = 1000/portTICK_PERIOD_MS;

  for (int i;;) {
    TempAndHumidity lastValues = dht.getTempAndHumidity();
    suhuUdara = lastValues.temperature;
    kelembabanUdara = lastValues.humidity;

    sensorSuhuAir.requestTemperatures();
    suhuAir = sensorSuhuAir.getTempCByIndex(0);
    jmlTask1 += 1;
    jmlTotal += 1;

    vTaskDelay(xDelayTask1);
  }
}

void Task2(void *pvParameters) {
```

```
  const TickType_t xDelayTask2 = 1000/portTICK_PERIOD_MS;

  for (int i;;) {
    sensorValue = analogRead(tdsPin);
    konduktivitasAir = ((0.2142*sensorValue)+494.93);

    sensorValue = analogRead(tdsPin);
    tds = (0.3417*sensorValue)+281.08;
    jmlTask2 += 1;
    jmlTotal += 1;

    vTaskDelay(xDelayTask2);
  }
}

void Task3(void *pvParameters) {
  const TickType_t xDelayTask3 = 1000/portTICK_PERIOD_MS;
  const TickType_t xDelayTrig = 2;
  const TickType_t xDelayTrigN = 10;

  for (int i;;) {
    digitalWrite(trigPin, LOW);
    vTaskDelay(xDelayTrig);
    digitalWrite(trigPin, HIGH);
    vTaskDelay(xDelayTrigN);
    digitalWrite(trigPin, LOW);
    durasi = pulseIn(echoPin, HIGH);
    tinggiAir = tinggiEmber - (durasi*0.034/2);

    for (int i=0; i<10; i++) {
      buf[i] = analogRead(phPin);
    }
    for (int i=0; i<9; i++) {
      for (int j=i+1; j<10; j++) {
        if (buf[i] > buf[j]) {
          temp = buf[i];
          buf[i] = buf[j];
          buf[j] = temp;
        }
      }
    }
    avgValue = 0;
    for(int i=2; i<8; i++) {
      avgValue += buf[i];
    }
    phVol = avgValue/6;
    if ( phVol > 0 && phVol < 300) {
      phAir = 14;
    } else if ( phVol > 300 && phVol < 600) {
      phAir = 13;
    } else if ( phVol > 600 && phVol < 900 ) {
      phAir = 12;
    } else if ( phVol > 900 && phVol < 1200 ) {
      phAir = 11;
    } else if ( phVol > 1200 && phVol < 1500 ) {
      phAir = 10;
```

```
    } else if ( phVol > 1500 && phVol < 1800 ) {
      phAir = 9;
    } else if ( phVol > 1800 && phVol < 2100 ) {
      phAir = 8;
    } else if ( phVol > 2100 && phVol < 2400 ) {
      phAir = 7;
    } else if ( phVol > 2400 && phVol < 2700) {
      phAir = 6;
    } else if ( phVol > 2700 && phVol < 3000 ) {
      phAir = 5;
    } else if ( phVol > 3000 && phVol < 3300) {
      phAir = 4;
    } else if ( phVol > 3300 && phVol < 3600) {
      phAir = 3;
    } else if ( phVol > 3600 && phVol < 3900) {
      phAir = 2;
    } else if ( phVol > 3900 && phVol < 4096 ) {
      phAir = 1;
    }
    jmlTask3 += 1;
    jmlTotal += 1;

    vTaskDelay(xDelayTask3);
  }
}

void Task4(void *pvParameters) {
  const TickType_t xDelayTask4 = 1000/portTICK_PERIOD_MS;

  const TickType_t xRelayOff = 15000/portTICK_PERIOD_MS;
  const TickType_t xRelayOn = 1500/portTICK_PERIOD_MS;

  for (int i;;) {
    if (rtcKu.hours > 9 && rtcKu.hours < 15) {
      digitalWrite(relayPin, LOW);
      vTaskDelay(xRelayOn);
      digitalWrite(relayPin, HIGH);
      vTaskDelay(xRelayOff);
    } else if ( suhuUdara > 35) {
      digitalWrite(relayPin, LOW);
      vTaskDelay(xRelayOn);
      digitalWrite(relayPin, HIGH);
      vTaskDelay(xRelayOff);
    } else {
      digitalWrite(relayPin, HIGH);
      vTaskDelay(xRelayOff);
    }

    if (tds < 1000) {
      for (pos = 0; pos <= 45; pos+=1) {
        servo.write(pos);
        delay(15);
      }
      for (pos = 45; pos >=0; pos -= 1) {
        servo.write(pos);
        delay(15);
```

```
      }
    } else {
    }
    jmlTask4 += 1;
    jmlTotal += 1;

    vTaskDelay(xDelayTask4);
  }
}

void TaskKirim(void *pvParameters) {
  const TickType_t xDelay = 5000/portTICK_PERIOD_MS;
  TickType_t xLastWakeTime;

  for (int i;;) {

    vTaskDelayUntil(&xLastWakeTime, xDelay);

    Serial.print("DATA, TIME, TIMER,");

    antares.checkMqttConnection();

    jmlKirim += 1;
    jmlTotal += 1;

    antares.add("sU", suhuUdara);
    antares.add("kU", kelembabanUdara);
    antares.add("sA", suhuAir);
    antares.add("jT1", jmlTask1);

    antares.add("kA", konduktivitasAir);
    antares.add("tds", tds);
    antares.add("jT2", jmlTask2);

    antares.add("tA", tinggiAir);
    antares.add("ph", phAir);
    antares.add("jT3", jmlTask3);

    antares.add("jT4", jmlTask4);

    antares.add("jk", jmlKirim);
    antares.add("jTot", jmlTotal);

    antares.publish(projectName, deviceName);

    Serial.print(suhuUdara);
    Serial.print(",");
    Serial.print(kelembabanUdara);
    Serial.print(",");
    Serial.print(suhuAir);
    Serial.print(",");
    Serial.print(jmlTask1);
    Serial.print(",");
    Serial.print(konduktivitasAir);
    Serial.print(",");
    Serial.print(tds);
```

```
    Serial.print(",");
    Serial.print(jmlTask2);
    Serial.print(",");
    Serial.print(tinggiAir);
    Serial.print(",");
    Serial.print(phAir);
    Serial.print(",");
    Serial.print(jmlTask3);
    Serial.print(",");
    Serial.print(jmlTask4);
    Serial.print(",");
    Serial.print(jmlKirim);
    Serial.print(",");
    Serial.println(jmlTotal);


    jmlTask1 = 0;
    jmlTask2 = 0;
    jmlTask3 = 0;
    jmlTask4 = 0;
    jmlKirim = 0;
    jmlTotal = 0;
  }
}
```

Lampiran 6 *Source Code* program 7 *task* freeRTOS

```
#include <AntaresESP32MQTT.h>
#define ACCESSKEY       "ACCESS KEY AKUN ANTARES"
#define WIFISSID        "SSID WIFI"
#define PASSWORD        "PASSWORD WIFI"
#define projectName     "NAMA PROJECT ANTARES"
#define deviceName      "NAMA DEVICE ANTARES"

#include <virtuabotixRTC.h>

#include <DHTesp.h>

#include <OneWire.h>
#include <DallasTemperature.h>

#include <Servo.h>
//---------------------------------------------------------------------
AntaresESP32MQTT antares(ACCESSKEY);

virtuabotixRTC rtcKu(14, 12, 13);

int dhtPin = 4;
DHTesp dht;

const int oneWirePin = 25;
OneWire oneWire(oneWirePin);
DallasTemperature sensorSuhuAir(&oneWire);

int tdsPin = 27;
int sensorValue;

const int trigPin = 2;
const int echoPin = 5;
long durasi;
int tinggiEmber = 30;

const int phPin = 35;
int buf[10], temp, phVol, avgValue;

int servoPin = 26;
Servo servo;
int pos = 0;

const int relayPin = 19;

//---------------------------------------------------------------------
int jam;
int suhuUdara;
int kelembabanUdara;
int suhuAir;
int konduktivitasAir;
int tds;
int tinggiAir;
int phAir;
int jmlTask1;
```

```
int jmlTask2;
int jmlTask3;
int jmlTask4;
int jmlTask5;
int jmlTask6;
int jmlKirim;
int jmlTotal;

long previousMillis = 0;
long interval = 1000;

//---------------------------------------------------------------------
void Task1();
void Task2();
void Task3();
void Task4();
void Task5();
void Task6();
void TaskKirim();

void setup() {
  Serial.begin(9600);

  Serial.println("CLEARDATA");
  Serial.println("LABEL, Pukul, sU, kU, sA, task1, kA, tds, task2,
tA, task3, pH, task4, task5, task6, kirim, jTot");
  Serial.println("RESETTIMER");

  antares.setDebug(true);
  antares.wifiConnection(WIFISSID, PASSWORD);
  antares.setMqttServer();

  rtcKu.setDS1302Time(00, 42, 20, 6, 18, 12, 2020); // (detik,
menit, jam, hari, tanggal, bulan, tahun)

  dht.setup(dhtPin, DHTesp::DHT11);

  sensorSuhuAir.begin();

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  pinMode(relayPin, OUTPUT);

  servo.attach(servoPin);

  //TASK
  xTaskCreate(
    Task1,
    "Task1",
    10000,
    NULL,
    1,
    NULL);

  xTaskCreate(
```

```
    Task2,
    "Task2",
    10000,
    NULL,
    1,
    NULL);

  xTaskCreate(
    Task3,
    "Task3",
    10000,
    NULL,
    1,
    NULL);

  xTaskCreate(
    Task4,
    "Task4",
    10000,
    NULL,
    1,
    NULL);

  xTaskCreate(
    Task5,
    "Task5",
    10000,
    NULL,
    1,
    NULL);

  xTaskCreate(
    Task6,
    "Task6",
    10000,
    NULL,
    1,
    NULL);

  xTaskCreate(
    TaskKirim,
    "TaskKirim",
    10000,
    NULL,
    2,
    NULL);
}

void loop() {
  unsigned long currentMillis = millis();
  rtcKu.updateTime();

  if (currentMillis - previousMillis > interval) {
    previousMillis = currentMillis;
    jam = rtcKu.hours;
  }
```

```
}

void Task1(void *pvParameters) {
  const TickType_t xDelayTask1 = 1000/portTICK_PERIOD_MS;

  for (int i;;) {
    TempAndHumidity lastValues = dht.getTempAndHumidity();
    suhuUdara = lastValues.temperature;
    kelembabanUdara = lastValues.humidity;

    sensorSuhuAir.requestTemperatures();
    suhuAir = sensorSuhuAir.getTempCByIndex(0);

    jmlTask1 += 1;
    jmlTotal += 1;
    vTaskDelay(xDelayTask1);
  }
}

void Task2(void *pvParameters) {
  const TickType_t xDelayTask2 = 1000/portTICK_PERIOD_MS;

  for (int i;;) {

    sensorValue = analogRead(tdsPin);
    konduktivitasAir = ((0.2142*sensorValue)+494.93);
    tds = (0.3417*sensorValue)+281.08;

    jmlTask2 += 1;
    jmlTotal += 1;
    vTaskDelay(xDelayTask2);
  }
}

void Task3(void *pvParameters) {
  const TickType_t xDelayTask3 = 1000 / portTICK_PERIOD_MS;
  const TickType_t xDelayTrig = 2;
  const TickType_t xDelayTrigN = 10;

  for (int i;;) {
    digitalWrite(trigPin, LOW);
    vTaskDelay(xDelayTrig);
    digitalWrite(trigPin, HIGH);
    vTaskDelay(xDelayTrigN);
    digitalWrite(trigPin, LOW);
    durasi = pulseIn(echoPin, HIGH);

    tinggiAir = tinggiEmber - (durasi*0.034/2);
    jmlTask3 += 1;
    jmlTotal += 1;
    vTaskDelay(xDelayTask3);
  }
}

void Task4(void *pvParameters) {
  const TickType_t xDelayTask4 = 1000 / portTICK_PERIOD_MS;
```

```
  for (int i;;) {
    for (int i=0; i<10; i++) {
      buf[i] = analogRead(phPin);
    }
    for (int i=0; i<9; i++) {
      for (int j=i+1; j<10; j++) {
        if (buf[i] > buf[j]) {
          temp = buf[i];
          buf[i] = buf[j];
          buf[j] = temp;
        }
      }
    }
    avgValue = 0;
    for(int i=2; i<8; i++) {
      avgValue += buf[i];
    }
    phVol = avgValue/6;
    if ( phVol > 0 && phVol < 300) {
      phAir = 14;
    } else if ( phVol > 300 && phVol < 600) {
      phAir = 13;
    } else if ( phVol > 600 && phVol < 900 ) {
      phAir = 12;
    } else if ( phVol > 900 && phVol < 1200 ) {
      phAir = 11;
    } else if ( phVol > 1200 && phVol < 1500 ) {
      phAir = 10;
    } else if ( phVol > 1500 && phVol < 1800 ) {
      phAir = 9;
    } else if ( phVol > 1800 && phVol < 2100 ) {
      phAir = 8;
    } else if ( phVol > 2100 && phVol < 2400 ) {
      phAir = 7;
    } else if ( phVol > 2400 && phVol < 2700) {
      phAir = 6;
    } else if ( phVol > 2700 && phVol < 3000 ) {
      phAir = 5;
    } else if ( phVol > 3000 && phVol < 3300) {
      phAir = 4;
    } else if ( phVol > 3300 && phVol < 3600) {
      phAir = 3;
    } else if ( phVol > 3600 && phVol < 3900) {
      phAir = 2;
    } else if ( phVol > 3900 && phVol < 4096 ) {
      phAir = 1;
    }
    jmlTask4 += 1;
    jmlTotal += 1;

    vTaskDelay(xDelayTask4);
  }
}

void Task5(void *pvParameters) {
```

```
  const TickType_t xRelayOff = 15000/portTICK_PERIOD_MS;
  const TickType_t xRelayOn = 1500/portTICK_PERIOD_MS;

  for (int i;;) {
    if (rtcKu.hours > 9 && rtcKu.hours < 15) {
      digitalWrite(relayPin, LOW);
      vTaskDelay(xRelayOn);
      digitalWrite(relayPin, HIGH);
      vTaskDelay(xRelayOff);
    } else if ( suhuUdara > 35) {
      digitalWrite(relayPin, LOW);
      vTaskDelay(xRelayOn);
      digitalWrite(relayPin, HIGH);
      vTaskDelay(xRelayOff);
    } else {
      digitalWrite(relayPin, HIGH);
      vTaskDelay(xRelayOff);
    }
    jmlTask5 += 1;
    jmlTotal += 1;
  }
}

void Task6(void *pvParameters) {
  const TickType_t xDelayTask6 = 1000 / portTICK_PERIOD_MS;

  for (int i;;) {
    if (tds < 1000) {
      for (pos = 0; pos <= 45; pos+=1) {
        servo.write(pos);
        delay(15);
      }
      for (pos = 45; pos >=0; pos -= 1) {
        servo.write(pos);
        delay(15);
      }
    } else {
    }
    jmlTask6 += 1;
    jmlTotal += 1;

    vTaskDelay(xDelayTask6);
  }
}

void TaskKirim(void *pvParameters) {
  const TickType_t xDelay = 5000/portTICK_PERIOD_MS;
  TickType_t xLastWakeTime;

  for (int i;;) {

    vTaskDelayUntil(&xLastWakeTime, xDelay);

    Serial.print("DATA, TIME, TIMER,");

    antares.checkMqttConnection();
```

```
jmlKirim += 1;
jmlTotal += 1;

antares.add("sU",   suhuUdara);
antares.add("kU",   kelembabanUdara);
antares.add("sA",   suhuAir);
antares.add("jT1",  jmlTask1);
antares.add("kA",   konduktivitasAir);
antares.add("tds",  tds);
antares.add("jT2",  jmlTask2);
antares.add("tA",   tinggiAir);
antares.add("jT3",  jmlTask3);
antares.add("ph",   phAir);
antares.add("jT4",  jmlTask4);
antares.add("jT5",  jmlTask5);
antares.add("jT6",  jmlTask6);
antares.add("jk",   jmlKirim);
antares.add("jTot", jmlTotal);

antares.publish(projectName, deviceName);

Serial.print(suhuUdara);
Serial.print(",");
Serial.print(kelembabanUdara);
Serial.print(",");
Serial.print(suhuAir);
Serial.print(",");
Serial.print(jmlTask1);
Serial.print(",");
Serial.print(konduktivitasAir);
Serial.print(",");
Serial.print(tds);
Serial.print(",");
Serial.print(jmlTask2);
Serial.print(",");
Serial.print(tinggiAir);
Serial.print(",");
Serial.print(jmlTask3);
Serial.print(",");
Serial.print(phAir);
Serial.print(",");
Serial.print(jmlTask4);
Serial.print(",");
Serial.print(jmlTask5);
Serial.print(",");
Serial.print(jmlTask6);
Serial.print(",");
Serial.print(jmlKirim);
Serial.print(",");
Serial.println(jmlTotal);

jmlTask1 = 0;
jmlTask2 = 0;
jmlTask3 = 0;
jmlTask4 = 0;
```

```
    jmlTask5 = 0;
    jmlTask6 = 0;
    jmlKirim = 0;
    jmlTotal = 0;
  }
}
```

KARTU BIMBINGAN SKRIPSI

Prodi S1 Teknik Informatika Universitas Hasanuddin

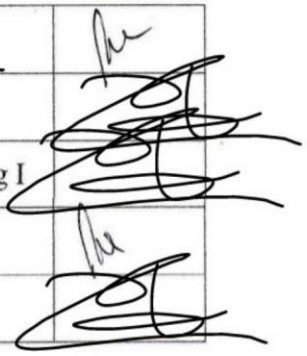
Stb.	Nama Mahasiswa
D42115010	Ryan Rafli

Pembimbing.	Nama Pembimbing	Paraf & Tgl. Persetujuan Ujian Akhir
I	Adnan, S.T., M.T., Ph.D.	
II	Dr. Eng. Zulkifli Tahir, S.T., M.Sc.	
No SK Pemb:		

Judul Skripsi:	Penerapan *Multitasking* pada Sistem Hidroponik Terotomasi Berbasis *Internet of Things*

No.	Tanggal Bimbingan	Uraian Kegiatan Bimbingan	Paraf Pemb.
1	01-04-2020	Laporan progress pengerjaan skripsi ke Pembimbing I	
2	16-04-2020	Laporan progress pengerjaan skripsi ke Pembimbing I	
3	16-05-2020	Laporan progress pengerjaan skripsi ke Pembimbing I	
4	01-06-2020	Laporan progress pengerjaan skripsi ke Pembimbing I	
5	17-07-2020	Laporan progress pengerjaan skripsi ke Pembimbing I	
6	28-07-2020	Laporan progress pengerjaan skripsi ke Pembimbing I	
7	19-08-2020	Laporan progress pengerjaan skripsi ke Pembimbing I	
8	29-09-2020	Konsultasi penulisan skripsi ke Pembimbing II	
9	03-10-2020	Konsultasi penulisan skripsi ke Pembimbing II	
10	12-10-2020	Konsultasi penulisan skripsi ke Pembimbing II	
11	16-10-2020	Laporan progress penulisan skripsi ke Pembimbing II	
12	26-10-2020	Laporan progress penulisan skripsi ke Pembimbing II	

13	27-10-2020	Asistensi skripsi ke Pembimbing II	
14	27-10-2020	Asistensi skripsi ke Pembimbing I	
15	01-12-2020	Laporan progress pengerjaan skripsi ke Pembimbing I	
16	24-12-2020	Asistensi skripsi ke Pembimbing II	
17	29-12-2020	Asistensi skripsi ke Pembimbing I	



LEMBAR PERBAIKAN SKRIPSI

"PENERAPAN MULTITASKING PADA SISTEM HIDROPONIK TEROTOMASI BERBASIS INTERNET OF THINGS"

OLEH:

RYAN RAFLI

D421 15 010

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana tanggal 27 Januari 2021, Telah dilakukan perbaikan penulisan dan isi skripsi berdasarkan usulan dari penguji dan pembimbing skripsi.

Persetujuan perbaikan oleh tim penguji:

	Nama	**Tanda Tangan**
Ketua	Adnan, S.T., M.T., Ph.D.	
Sekretaris	Dr. Eng. Zulkifli Tahir, S.T., M.Sc.	
Anggota	Dr. Ir. Zahir Zainuddin, M.Sc.	
	Ir. Christoforus Yohannes, M.T.	

Persetujuan perbaikan oleh pembimbing:

Pembimbing	**Nama**	**Tanda Tangan**
I	Adnan, S.T., M.T., Ph.D.	
II	Dr. Eng. Zulkifli Tahir, S.T., M.Sc	