LAMPIRAN

Dokumentasi Penelitian

Sketch Lengkap

```
// ============================================
// SISTEM HIDROPONIK ESP32 - FREERTOS + BLYNK
// PIN: 16=pH UP | 17=pH DOWN | 18=NUTRISI
// KALIBRASI PERMANEN + SETTING AMBANG BATAS VIA BLYNK
// ============================================

#define BLYNK_TEMPLATE_ID "TMPL6KjLqhe7B"
#define BLYNK_TEMPLATE_NAME "monitoring hidroponik dft"
#define BLYNK_AUTH_TOKEN "ZSG50cQtDXTu2t76ue4cWayLyxJ8QjK3"

#include <Wire.h>
#include <Adafruit_ADS1X15.h>
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
#include <Preferences.h>

char wifi_ssid[] = "ANOMALI";           //SESUAIKAN
char wifi_password[] = "sadardiriaja";  //SESUAIKAN

Adafruit_ADS1115 modul_ads;
Preferences preferensi_kalibrasi;

const int pin_buzzer = 4;
const int pin_relay_pompa_ph_naik = 16;
const int pin_relay_pompa_ph_turun = 17;
const int pin_relay_pompa_nutrisi = 18;

const int channel_sensor_tds = 3;
const int channel_sensor_ph = 0;

float tegangan_netral_ph = 1.65;
float nilai_slope_ph = -5.70;
float suhu_air_asumsi = 25.0;
float tegangan_referensi_ads = 4.096;

float batas_ppm_minimum = 0;
float batas_ppm_maksimum = 1000;
float batas_ph_minimum = 0;
float batas_ph_maksimum = 14;

unsigned long durasi_pompa_ph_naik = 3000;
unsigned long durasi_pompa_ph_turun = 3000;
unsigned long durasi_pompa_nutrisi = 5000;

unsigned long interval_baca_sensor = 10;
unsigned long interval_kirim_blynk = 1000;
unsigned long interval_log_ringkas = 1000;

volatile float nilai_ph_terbaca = 7.0;
volatile float nilai_ppm_terbaca = 0.0;
volatile float tegangan_ph_terakhir = 0.0;

volatile bool status_pompa_ph_naik_nyala = false;
```

```
volatile bool status_pompa_ph_turun_nyala = false;
volatile bool status_pompa_nutrisi_nyala = false;
volatile bool status_buzzer_nyala = false;

volatile float tegangan_kalibrasi_ph7 = 0.0;
volatile float tegangan_kalibrasi_ph4 = 0.0;
volatile bool kalibrasi_ph7_tersimpan = false;
volatile bool kalibrasi_ph4_tersimpan = false;

String teks_status_ph = "BELUM ADA DATA";
String teks_status_ppm = "BELUM ADA DATA";

TaskHandle_t handle_task_baca_sensor;
TaskHandle_t handle_task_kontrol_ph;
TaskHandle_t handle_task_kontrol_ppm;
TaskHandle_t handle_task_tampilkan_data;
TaskHandle_t handle_task_blynk;

// ==========================================
// WAJIB DI LUAR TASK: CALLBACK BAWAAN BLYNK
// ==========================================
BLYNK_WRITE(V10) {
  int nilai_tombol = param.asInt();
  if (nilai_tombol == 1) {
    digitalWrite(pin_buzzer, HIGH);
    delay(200);
    digitalWrite(pin_buzzer, LOW);
    Serial.println("[BLYNK] Test buzzer ditekan.");
  }
}

BLYNK_WRITE(V11) {
  int nilai_tombol = param.asInt();
  if (nilai_tombol == 1) {
    tegangan_kalibrasi_ph7 = tegangan_ph_terakhir;
    kalibrasi_ph7_tersimpan = true;
    Serial.println("[KALIBRASI] pH 7 disimpan -> " +
String(tegangan_kalibrasi_ph7, 4) + "V");
  }
}

BLYNK_WRITE(V12) {
  int nilai_tombol = param.asInt();
  if (nilai_tombol == 1) {
    tegangan_kalibrasi_ph4 = tegangan_ph_terakhir;
    kalibrasi_ph4_tersimpan = true;
    Serial.println("[KALIBRASI] pH 4 disimpan -> " +
String(tegangan_kalibrasi_ph4, 4) + "V");
  }
}

BLYNK_WRITE(V13) {
  int nilai_tombol = param.asInt();
  if (nilai_tombol == 1) {
```

```
    if (kalibrasi_ph7_tersimpan == false || kalibrasi_ph4_tersimpan
== false) {
      Serial.println("[KALIBRASI] GAGAL! Simpan pH 7 & pH 4 dulu
(V11 & V12).");
    } else if (tegangan_kalibrasi_ph7 == tegangan_kalibrasi_ph4) {
      Serial.println("[KALIBRASI] GAGAL! Tegangan pH 7 & pH 4 sama,
cek probe.");
    } else {
      nilai_slope_ph = (7.0 - 4.0) / (tegangan_kalibrasi_ph7 -
tegangan_kalibrasi_ph4);
      tegangan_netral_ph = tegangan_kalibrasi_ph7;

      preferensi_kalibrasi.begin("kalibrasi_ph", false);
      preferensi_kalibrasi.putFloat("tegangan_netral",
tegangan_netral_ph);
      preferensi_kalibrasi.putFloat("slope", nilai_slope_ph);
      preferensi_kalibrasi.end();

      Serial.println("[KALIBRASI] TERSIMPAN PERMANEN! netral=" +
String(tegangan_netral_ph, 4) + "V | slope=" + String(nilai_slope_ph,
4));

      kalibrasi_ph7_tersimpan = false;
      kalibrasi_ph4_tersimpan = false;
    }
  }
}

// ---------- SETTING AMBANG BATAS PH & PPM DARI DASHBOARD ---------
-
BLYNK_WRITE(V14) {
  batas_ph_minimum = param.asFloat();
  Serial.println("[SETTING] pH minimum = " + String(batas_ph_minimum,
2));
}

BLYNK_WRITE(V15) {
  batas_ph_maksimum = param.asFloat();
  Serial.println("[SETTING] pH maksimum = " +
String(batas_ph_maksimum, 2));
}

BLYNK_WRITE(V16) {
  batas_ppm_minimum = param.asFloat();
  Serial.println("[SETTING] PPM minimum = " +
String(batas_ppm_minimum, 1));
}

BLYNK_WRITE(V17) {
  batas_ppm_maksimum = param.asFloat();
  Serial.println("[SETTING] PPM maksimum = " +
String(batas_ppm_maksimum, 1));
}

void setup() {
```

```
  Serial.begin(115200);
  Wire.begin(21, 22);

  if (!modul_ads.begin()) {
    Serial.println("[BOOT] ERROR: ADS1115 gak kedetect!");
    while (1) {
      vTaskDelay(pdMS_TO_TICKS(1000));
    }
  }

  preferensi_kalibrasi.begin("kalibrasi_ph", true);
  bool ada_data_tersimpan =
preferensi_kalibrasi.isKey("tegangan_netral");
  if (ada_data_tersimpan) {
    tegangan_netral_ph =
preferensi_kalibrasi.getFloat("tegangan_netral", tegangan_netral_ph);
    nilai_slope_ph = preferensi_kalibrasi.getFloat("slope",
nilai_slope_ph);
    Serial.println("[BOOT] Kalibrasi lama dimuat -> netral=" +
String(tegangan_netral_ph, 4) + "V | slope=" + String(nilai_slope_ph,
4));
  } else {
    Serial.println("[BOOT] Belom ada kalibrasi tersimpan, pake nilai
default.");
  }
  preferensi_kalibrasi.end();

  pinMode(pin_buzzer, OUTPUT);
  pinMode(pin_relay_pompa_ph_naik, OUTPUT);
  pinMode(pin_relay_pompa_ph_turun, OUTPUT);
  pinMode(pin_relay_pompa_nutrisi, OUTPUT);

  digitalWrite(pin_buzzer, LOW);
  digitalWrite(pin_relay_pompa_ph_naik, LOW);
  digitalWrite(pin_relay_pompa_ph_turun, LOW);
  digitalWrite(pin_relay_pompa_nutrisi, LOW);

  delay(1000);

  digitalWrite(pin_buzzer, HIGH);
  delay(60);
  digitalWrite(pin_buzzer, LOW);
  delay(60);
  digitalWrite(pin_buzzer, HIGH);
  delay(60);
  digitalWrite(pin_buzzer, LOW);

  delay(2000);

  Blynk.begin(BLYNK_AUTH_TOKEN, wifi_ssid, wifi_password);

  xTaskCreatePinnedToCore(task_baca_sensor, "task_baca_sensor", 4096,
NULL, 1, &handle_task_baca_sensor, 1);
  xTaskCreatePinnedToCore(task_kontrol_ph, "task_kontrol_ph", 4096,
NULL, 1, &handle_task_kontrol_ph, 1);
```

```
  xTaskCreatePinnedToCore(task_kontrol_ppm, "task_kontrol_ppm", 4096,
NULL, 1, &handle_task_kontrol_ppm, 1);
  xTaskCreatePinnedToCore(task_tampilkan_data, "task_tampilkan_data",
4096, NULL, 1, &handle_task_tampilkan_data, 1);
  xTaskCreatePinnedToCore(task_blynk, "task_blynk", 4096, NULL, 1,
&handle_task_blynk, 1);

  Serial.println("[BOOT] Sistem live.");
}

void loop() {
}

// ==========================================
// TASK 1 : BACA SENSOR PH & PPM
// ==========================================
void task_baca_sensor(void *parameter) {
  unsigned long waktu_baca_terakhir = 0;

  while (true) {
    unsigned long waktu_sekarang = millis();

    if (waktu_sekarang - waktu_baca_terakhir >= interval_baca_sensor)
{
      waktu_baca_terakhir = waktu_sekarang;

      int16_t nilai_mentah_ph =
modul_ads.readADC_SingleEnded(channel_sensor_ph);
      float tegangan_ph = (nilai_mentah_ph * tegangan_referensi_ads)
/ 32767.0;
      nilai_ph_terbaca = 7.0 + ((tegangan_ph - tegangan_netral_ph) *
nilai_slope_ph);
      tegangan_ph_terakhir = tegangan_ph;

      int16_t nilai_mentah_tds =
modul_ads.readADC_SingleEnded(channel_sensor_tds);
      float tegangan_tds = (nilai_mentah_tds *
tegangan_referensi_ads) / 32767.0;
      float faktor_kompensasi_suhu = 1.0 + 0.02 * (suhu_air_asumsi -
25.0);
      float tegangan_terkompensasi = tegangan_tds /
faktor_kompensasi_suhu;

      float hasil_ppm = (133.42 * tegangan_terkompensasi *
tegangan_terkompensasi * tegangan_terkompensasi
                         - 255.86 * tegangan_terkompensasi *
tegangan_terkompensasi
                         + 857.39 * tegangan_terkompensasi)
                        * 0.5;

      if (hasil_ppm < 0) {
        hasil_ppm = 0;
      }
      nilai_ppm_terbaca = hasil_ppm;
    }
```

```
    vTaskDelay(pdMS_TO_TICKS(10));
  }
}

// ==========================================
// TASK 2 : KONTROL PH
// ==========================================
void task_kontrol_ph(void *parameter) {

  while (true) {

    float ph_saat_ini = nilai_ph_terbaca;

    // Default: semua relay OFF
    digitalWrite(pin_relay_pompa_ph_naik, LOW);
    digitalWrite(pin_relay_pompa_ph_turun, LOW);

    status_pompa_ph_naik_nyala = false;
    status_pompa_ph_turun_nyala = false;

    if (ph_saat_ini < batas_ph_minimum) {

      teks_status_ph = "TERLALU RENDAH (ASAM)";

      digitalWrite(pin_relay_pompa_ph_naik, HIGH);
      status_pompa_ph_naik_nyala = true;

    } else if (ph_saat_ini > batas_ph_maksimum) {

      teks_status_ph = "TERLALU TINGGI (BASA)";

      digitalWrite(pin_relay_pompa_ph_turun, HIGH);
      status_pompa_ph_turun_nyala = true;

    } else {
      digitalWrite(pin_relay_pompa_ph_naik, LOW);
      digitalWrite(pin_relay_pompa_ph_turun, LOW);
      teks_status_ph = "AMAN";
    }

    vTaskDelay(pdMS_TO_TICKS(100));
  }
}

// ==========================================
// TASK 3 : KONTROL PPM
// ==========================================
void task_kontrol_ppm(void *parameter) {

  while (true) {

    float ppm_saat_ini = nilai_ppm_terbaca;

    digitalWrite(pin_relay_pompa_nutrisi, LOW);
    digitalWrite(pin_buzzer, LOW);
```

```
    status_pompa_nutrisi_nyala = false;
    status_buzzer_nyala = false;

    if (ppm_saat_ini < batas_ppm_minimum) {

      teks_status_ppm = "TERLALU RENDAH";

      digitalWrite(pin_relay_pompa_nutrisi, HIGH);
      status_pompa_nutrisi_nyala = true;

    } else {
      digitalWrite(pin_relay_pompa_nutrisi, LOW);
      teks_status_ppm = "AMAN";
    }

    vTaskDelay(pdMS_TO_TICKS(100));
  }
}

// ==========================================
// TASK 4 : LOG RINGKAS
// ==========================================
void task_tampilkan_data(void *parameter) {
  unsigned long waktu_tampil_terakhir = 0;

  while (true) {
    unsigned long waktu_sekarang = millis();

    if (waktu_sekarang - waktu_tampil_terakhir >=
interval_log_ringkas) {
      waktu_tampil_terakhir = waktu_sekarang;

      Serial.println("[STATUS] pH=" + String(nilai_ph_terbaca, 2) +
"(" + teks_status_ph + ") | PPM=" + String(nilai_ppm_terbaca, 1) +
"(" + teks_status_ppm + ") | Batas pH=" + String(batas_ph_minimum, 1)
+ "-" + String(batas_ph_maksimum, 1) + " | Batas PPM=" +
String(batas_ppm_minimum, 0) + "-" + String(batas_ppm_maksimum, 0) +
" | WiFi=" + String(WiFi.status() == WL_CONNECTED ? "OK" :
"TERPUTUS") + " | Blynk=" + String(Blynk.connected() ? "OK" :
"TERPUTUS"));
    }

    vTaskDelay(pdMS_TO_TICKS(200));
  }
}

// ==========================================
// TASK 5 : BLYNK RUN + KIRIM DATA
// ==========================================
void task_blynk(void *parameter) {
  unsigned long waktu_kirim_terakhir = 0;
  bool status_terputus_sebelumnya = false;

  while (true) {
    Blynk.run();
```

```
    unsigned long waktu_sekarang = millis();

    if (waktu_sekarang - waktu_kirim_terakhir >=
interval_kirim_blynk) {
      waktu_kirim_terakhir = waktu_sekarang;

      if (Blynk.connected()) {
        Blynk.virtualWrite(V0, nilai_ph_terbaca);
        Blynk.virtualWrite(V1, nilai_ppm_terbaca);
        Blynk.virtualWrite(V2, status_pompa_ph_naik_nyala ? "ON" :
"OFF");
        Blynk.virtualWrite(V3, status_pompa_ph_turun_nyala ? "ON" :
"OFF");
        Blynk.virtualWrite(V4, status_pompa_nutrisi_nyala ? "ON" :
"OFF");
        Blynk.virtualWrite(V5, status_buzzer_nyala ? "ON" : "OFF");
        status_terputus_sebelumnya = false;
      } else {
        if (status_terputus_sebelumnya == false) {
          Serial.println("[BLYNK] Terputus!");
          status_terputus_sebelumnya = true;
        }
      }
    }

    vTaskDelay(pdMS_TO_TICKS(1000));
  }
}
```