DAFTAR PUSTAKA

Alfa Rizzi, T., Aziz Abdillah, R., Efani Yancandra, Y., Wijaya, M., & Voutama, A. (2025). Implementasi React Vite Dalam Pengembangan Antarmuka Sistem Pemesanan Tiket Pesawat Dengan Metode Scrum. *Jurnal PROCESSOR*, *20*(1). https://doi.org/10.33998/processor.2025.20.1.2231

Alfian. (2024). *SISTEM PENYIRAMAN TANAMAN OTOMATIS DAN MONITORING KELEMBAPAN TANAH BERBASIS ESP8266, NODE.JS, DAN ANDROID*.

Asri, M., Abdullah, R. K., Wayan, I., & Ariawan, J. (2022). *Prototipe Perawatan Tanaman Hias Aglonema Menggunakan Sensor Yl-69 Berbasis IoT* (Vol. 11).

Daffa Ananda, M., Saragih, Y., Satrio Hadikusuma, R., Fadhlul Kamal, A., & Singaperbangsa Karawang, U. (2023). *Design of Smart Agricultural Systems Using MIT App and Firebase*.

Gabriel, I. D. (2024). *Implementasi Sistem Monitoring Indoor Hydroponic Farming Berbasis Website*.

Ifa Susuek Anselmus Talli, W., Dedy Irawan, J., & Xaverius Ariwibisono, F. (2023). *RANCANG BANGUN SISTEM MONITORING KUALITAS TANAH UNTUK TANAMAN CABAI BERBASIS IOT (INTERNET OF THINGS)*.

Islamy, I., & Wisudawati, L. M. (2023). Sistem Monitoring Smart Garden Tanaman Cabai Berbasis IoT Menggunakan Protokol MQTT, Node Red, dan Telegram Bot. *Jurnal Teknotan*, *17*(3), 197. https://doi.org/10.24198/jt.vol17n3.6

Nurhalimah, S., Yusa, A. M., & Fahmi, A. (2023). Rancang Bangun Sistem Monitoring KelembapanTanah dengan Konsep Smart Farming untuk Budidaya Tanaman Cabai Rawit Berbasis Internet of Things (IOT). *Software Development, Digital Business Intelligence, and Computer Engineering*, *1*(02), 49–54. https://doi.org/10.57203/session.v1i02.2023.40-54

Pitaloka, D. (2020). HORTIKULTURA: POTENSI, PENGEMBANGAN DAN TANTANGAN. *Jurnal Teknologi Terapan: G-Tech*, *1*(1), 1–4. https://doi.org/10.33379/gtech.v1i1.260

Rezki, A., Gde, I., Wirarama, P., Wirawan, W., & Zubaidi, A. (2021). *Rancang Bangun Sistem Monitoring dan Penyiraman Otomatis Pada Tanaman Bawang Merah Berbasis Internet of Things (Design of an Automatic Monitoring and Watering System for Shallot Plant Based on The Internet of Things)*.

Seetaram, J., Bhavya, A., Tarun, C., & Sameera, V. (2024). Internet of Things (IoT) Based Greenhouse Monitoring and Controlling System Using ESP-32. *IJARCCE*, *13*(6). https://doi.org/10.17148/ijarcce.2024.13605

Ubaform, A. S., & Iswari, L. (2021). *Penerapan React JS Pada Pengembangan FrontEnd*.

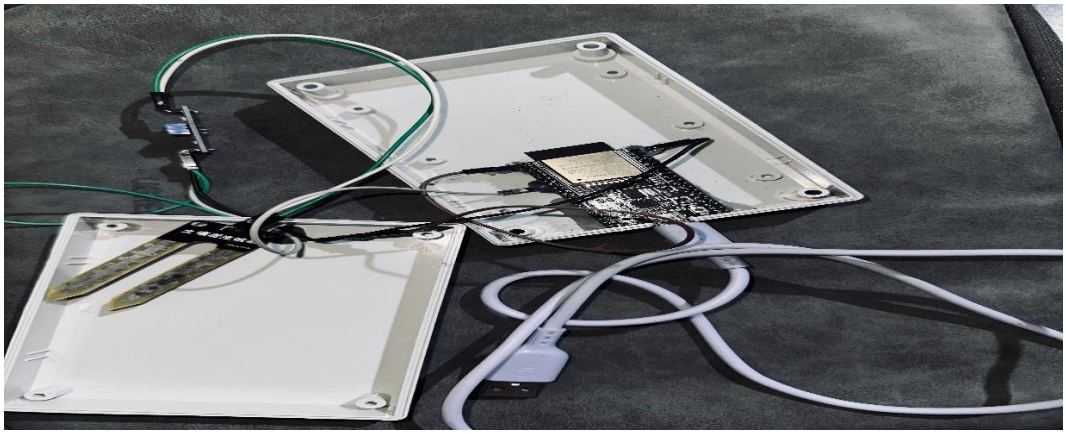
Wahyu, S., Syafaat, M., & Yuliana, A. (2020). Rancang Bangun Sistem Monitoring Pertumbuhan Tanaman Cabai Menggunakan Arduino Bertenaga Surya Terintegrasi Internet of Things (IoT). *Jurnal Teknologi*, *8*(1), 22–23. https://doi.org/10.31479/jtek.v1i8.63

Wira, J., Sistem, M. :, Mansa, J. W., Kainde, Q. C., & Sangkop, I. F. (2022). Sistem Monitor Kelembaban Tanah Berbasis Internet of Things. In *JOINTER : JOURNAL OF INFORMATICS ENGINEERING* (Vol. 03, Issue 01).

Yudha Purnama, H., & Supardi, A. S. (2024). *RANCANG BANGUN SISTEM KENDALI DAN MONITORING TANAMAN CABAI PADA GREENHOUSE BERBASIS INTERNET OF THINGS (IOT)*.

LAMPIRAN

Lampiran 1 Spesifikasi Alat Pengukur Kelembaban Tanah



No	Komponen	Spesifikasi Teknis
1	Nama Alat	YL-69 Kin
2	Mikrokontroler	ESP32 Devkit 1, mendukung WiFi dan Bluetooth, tegangan kerja 3.3V.
3	Sensor Kelembaban Tanah	YL-69 – Output analog, tahan korosi, akurat untuk pemantauan tanah.
4	Sumber Daya	Kabel USB dan Power Bank kapasitas 10.000, output 5V/2.4A input 5V/2A mAh sebagai pemberi daya ke ESP32.
5	Metode Komunikasi	WiFi (Web Server) – Mengirim dan menampilkan data melalui dashboard web.
6	Antarmuka Web	Web Dashboard dibangun menggunakan react, vite, dan typescript.
7	Penyimpanan Data	Data disimpan melalui firebase yang bisa diexport ke Microsoft excel
8	Box Elektrik	Sebagai wadah pelindung komponen alat rancangan. • Tinggi: 4 cm • Lebar: 8 cm • Panjang: 12 cm

Lampiran 2 *Source Code* Web Dashboard

Kodingan login

```
import { useState } from "react";
import {
  Button,
  Card,
  Typography,
  Container,
  Box,
  TextField,
} from "@mui/material";
import { useNavigate, Link } from "react-router-dom";
import { firebaseAuth } from "../../firebase/auth";

const LoginView = () => {
  const navigate = useNavigate();

  const [userEmail, setuserEmail] = useState("");
  const [userPassword, setUserPassword] = useState("");

  const handleSubmit = async () => {
    try {
      try {
        await firebaseAuth.signIn(userEmail, userPassword);
        console.log("Login successful");
      } catch (error: unknown) {
        console.log(error);
      }

      navigate("/");
    } catch (error: unknown) {
      console.log(error);
    }
  };

  return (
    <>
      <Container maxWidth="xs">
        <Card
          sx={{
            mt: 5,
            p: 8,
            display: "flex",
            flexDirection: "column",
            justifyContent: "center",
            alignItems: "center",
          }}
        >
          <Typography
            variant="h4"
            marginBottom={5}
```

```
  color="primary"
  fontWeight={"bold"}
>
  Login
</Typography>
<Box
  component="form"
  style={{
    display: "flex",
    flexDirection: "column",
    justifyContent: "center",
  }}
>
  <TextField
    label="E-mail"
    id="outlined-start-adornment"
    sx={{ m: 1, width: "36ch" }}
    value={userEmail}
    type="email"
    onChange={(e) => {
      setuserEmail(e.target.value);
    }}
  />

  <TextField
    label="Password"
    id="outlined-start-adornment"
    sx={{ m: 1, width: "36ch" }}
    value={userPassword}
    type="password"
    onChange={(e) => {
      setUserPassword(e.target.value);
    }}
  />
  <Button
    sx={{
      m: 1,
      backgroundColor: "dodgerblue",
      color: "#FFF",
      fontWeight: "bold",
    }}
    variant={"contained"}
    onClick={handleSubmit}
  >
    Login
  </Button>
  <Typography sx={{ mt: 2, textAlign: "center" }}>
    Don't have an account?{" "}
    <Link
      to="/signup"
      style={{
        color: "primary.main",
        fontWeight: "bold",
        textDecoration: "underline",
```

```
                }}
              >
                Sign Up here
              </Link>
            </Typography>
          </Box>
        </Card>
      </Container>
    </>
  );
};

export default LoginView;
```

Kodingan signup

```
import { useState } from "react";
import {
  Button,
  Card,
  Typography,
  Container,
  Box,
  TextField,
} from "@mui/material";
import { useNavigate, Link } from "react-router-dom";
import { firebaseAuth } from "../../firebase/auth";

const SignUpView = () => {
  const navigate = useNavigate();

  const [userEmail, setuserEmail] = useState("");
  const [userPassword, setUserPassword] = useState("");

  const handleSubmit = async () => {
    try {
      await firebaseAuth.signUp(userEmail, userPassword);
      console.log("Signup successful");
      navigate("/");
    } catch (error: unknown) {
      console.log(error);
    }
  };

  return (
    <Container maxWidth="xs">
      <Card
        sx={{
          mt: 5,
          p: 8,
          display: "flex",
          flexDirection: "column",
          justifyContent: "center",
          alignItems: "center",
        }}
```

```
>
  <Typography
    variant="h4"
    marginBottom={5}
    color="primary"
    fontWeight={"bold"}
  >
    Sign Up
  </Typography>
  <Box
    component="form"
    style={{
      display: "flex",
      flexDirection: "column",
      justifyContent: "center",
    }}
  >
    <TextField
      label="E-mail"
      id="signup-email"
      sx={{ m: 1, width: "36ch" }}
      value={userEmail}
      type="email"
      onChange={(e) => setuserEmail(e.target.value)}
    />

    <TextField
      label="Password"
      id="signup-password"
      sx={{ m: 1, width: "36ch" }}
      value={userPassword}
      type="password"
      onChange={(e) => setUserPassword(e.target.value)}
    />

    <Button
      sx={{
        m: 1,
        backgroundColor: "dodgerblue",
        color: "#FFF",
        fontWeight: "bold",
      }}
      variant="contained"
      onClick={handleSubmit}
    >
      Sign Up
    </Button>
    <Typography sx={{ mt: 2, textAlign: "center" }}>
      Already have an account?{" "}
      <Link
        to="/login"
        style={{
          color: "primary.main",
          fontWeight: "bold",
```

```
                textDecoration: "underline",
              }}
            >
              Login here
            </Link>
          </Typography>
        </Box>
      </Card>
    </Container>
  );
};

export default SignUpView;
```

Kodingan dashboard

```
/* eslint-disable @typescript-eslint/no-explicit-any */
import React, { useState, useEffect } from "react";
import { Card, Box, Stack, Typography } from "@mui/material";
import Grid from "@mui/material/Grid2";
import WaterIcon from "@mui/icons-material/Water";
import ThermostatIcon from "@mui/icons-material/Thermostat";
import OpacityIcon from "@mui/icons-material/Opacity";
import ScienceIcon from "@mui/icons-material/Science";
import { IDeviceModel } from "../../models/deviceModel";
import { firebaseDb } from "../../firebase/db";
import { removeDotsFromEmail } from
"../../utilities/removeDotsFromEmail";
import { firebaseAuth } from "../../firebase/auth";

const DashboardView: React.FC = () => {
  const [groupedDevices, setGroupedDevices] = useState<{
    [key: string]: IDeviceModel[];
  }>({});
  const [currentUser, setCurrentUser] = useState<any>(null);

  useEffect(() => {
    const fetchDevices = async () => {
      try {
        const currentUser = await firebaseAuth.getCurrentUser();
        setCurrentUser(currentUser);
        const email = removeDotsFromEmail(currentUser?.email ?? "");
        const data = await firebaseDb.readAll<{ [key: string]:
IDeviceModel }>(
          `${email}/devices`
        );

        const devicesArray = Object.entries(data).map(([deviceId,
device]) => ({
          ...device,
          id: deviceId,
          deviceId,
        }));

        const grouped = devicesArray.reduce((acc, device: any) => {
```

```
          const type = device.deviceType;
          acc[type] = acc[type] || [];
          acc[type].push(device);
          return acc;
        }, {} as { [key: string]: IDeviceModel[] });

        // setDevices(devicesArray);
        setGroupedDevices(grouped);
      } catch (error) {
        console.error("Error fetching devices:", error);
      }
    };

    fetchDevices();
  }, []);

  const iconMapping = {
    TDS: <WaterIcon fontSize="large" sx={{ color: "#2196f3" }} />,
    TEMPERATURE: <ThermostatIcon fontSize="large" sx={{ color: "#ff5722"
}} />,
    HUMIDITY: <OpacityIcon fontSize="large" sx={{ color: "#4caf50" }}
/>,
    PH: <ScienceIcon fontSize="large" sx={{ color: "#9c27b0" }} />,
    DEFAULT: <ScienceIcon fontSize="large" />,
  };

  return (
    <Box sx={{ p: { xs: 2, md: 3 }, bgcolor: "#f4f6f8", minHeight:
"100vh" }}>
      {currentUser && (
        <Typography
          variant="h4"
          gutterBottom
          sx={{
            fontWeight: 600,
            color: "text.primary",
            mb: 3,
            display: "flex",
            alignItems: "center",
            gap: 1,
          }}
        >
          Welcome back, {currentUser.displayName || "User"}!
        </Typography>
      )}

      {Object.keys(groupedDevices).length === 0 ? (
        <Box
          sx={{
            display: "flex",
            justifyContent: "center",
            alignItems: "center",
            height: "70vh",
            textAlign: "center",
```

```
          }}
        >
          <Typography variant="h5" color="textSecondary">
            No devices found. <br />
            <Typography variant="body1" component="span">
              Add your first device to see dashboard metrics
            </Typography>
          </Typography>
        </Box>
      ) : (
        <Grid container spacing={2}>
          {Object.entries(groupedDevices).map(([deviceType, devices]) =>
(
            <Grid
              container
              rowSpacing={2}
              columnSpacing={{ xs: 1, sm: 2, md: 3 }}
              key={deviceType}
            >
              <Card
                sx={{
                  p: 3,
                  // height: 180,
                  boxShadow: 3,
                  borderRadius: 4,
                  transition: "transform 0.3s, box-shadow 0.3s",
                  "&:hover": {
                    boxShadow: 6,
                    transform: "translateY(-4px)",
                  },
                  background: "linear-gradient(145deg, #ffffff,
#f8f9fa)",
                }}
              >
                <Stack direction="row" spacing={3} alignItems="center">
                  <Box
                    sx={{
                      p: 2,
                      bgcolor: "#f8f9fa",
                      borderRadius: 2,
                      display: "flex",
                      boxShadow: 1,
                    }}
                  >
                    {iconMapping[deviceType as keyof typeof iconMapping]
||
                      iconMapping.DEFAULT}
                  </Box>
                  <Stack justifyContent="center">
                    <Typography
                      color="textSecondary"
                      variant="subtitle1"
                      sx={{ fontWeight: 600, textTransform: "uppercase"
}}
```

```
                    >
                      {deviceType === "HUMADITY" ? "HUMIDITY" :
deviceType}
                    </Typography>
                    <Typography
                      fontSize="2rem"
                      fontWeight="bold"
                      sx={{ color: "primary.main" }}
                    >
                      {devices.length}
                      <Typography
                        component="span"
                        variant="body2"
                        sx={{ ml: 1, color: "text.secondary" }}
                      >
                        devices
                      </Typography>
                    </Typography>
                    {/* <Typography
                      variant="caption"
                      sx={{
                        color: "text.secondary",
                        mt: 1,
                        display: "block",
                      }}
                    >
                      Name: {devices[0]?.deviceName || "N/A"}
                    </Typography> */}
                  </Stack>
                </Stack>
              </Card>
            </Grid>
          ))}
        </Grid>
      )}

      {/* <Grid container spacing={3} mt={3}>
        {[
          {
            name: "TDS",
            data: tdsData,
            timestamps: tdsTimestamps,
            color: "#00bcd4",
          },
          {
            name: "Temperature",
            data: temperatureData,
            timestamps: tdsTimestamps,
            color: "#ff5722",
          },
          {
            name: "Humidity",
            data: humidityData,
            timestamps: tdsTimestamps,
```

```
            color: "#4caf50",
          },
          {
            name: "pH",
            data: phData,
            timestamps: tdsTimestamps,
            color: "#8e44ad",
          },
        ].map((sensor, index) => (
          <Grid item xs={12} sm={6} key={index}>
            <Card sx={{ p: { xs: 2, md: 4 }, boxShadow: 2, borderRadius:
2 }}>
              <Typography variant="h6" gutterBottom>
                {sensor.name} Sensor Data Over Time
              </Typography>
              {loading ? (
                <CircularProgress />
              ) : (
                <ReactApexChart
                  options={{
                    chart: { height: 350, type: "area" },
                    colors: [sensor.color],
                    dataLabels: { enabled: false },
                    stroke: { curve: "smooth", width: 2 },
                    xaxis: { type: "datetime", categories:
sensor.timestamps },
                    tooltip: { x: { format: "dd/MM/yy HH:mm" } },
                  }}
                  series={[
                    { name: `${sensor.name} (units)`, data: sensor.data
},
                  ]}
                  type="area"
                  height={350}
                />
              )}
            </Card>
          </Grid>
        ))}
      </Grid> */}
    </Box>
  );
};
```

Kodingan profil

```
// import { useEffect, useState } from "react";
// import { Button, Card, Typography, Box, TextField, Stack } from
"@mui/material";
// import { useNavigate, useParams } from "react-router-dom";
// import { IUserUpdateRequestModel } from "../../models/userModel";
```

```
// export default function EditProfileView() {
//   const navigate = useNavigate();
//   const { userId } = useParams();

//   const [user, setUser] = useState<IUserUpdateRequestModel>({
//     userId: userId!,
//     userName: "",
//     userPassword: "",
//   });

//   return (
//     <>
//       <Card
//         sx={{
//           mt: 5,
//           p: 8,
//         }}
//       >
//         <Typography
//           variant="h4"
//           marginBottom={5}
//           color="primary"
//           fontWeight={"bold"}
//         >
//           Edit My Profile
//         </Typography>
//         <Box
//           component="form"
//           style={{
//             display: "flex",
//             flexDirection: "column",
//             justifyContent: "center",
//           }}
//         >
//           <TextField
//             label="User Name"
//             id="outlined-start-adornment"
//             sx={{ m: 1 }}
//             value={user?.userName}
//             type="text"
//             onChange={(e) => {
//               setUser({
//                 ...user,
//                 userName: e.target.value,
//               });
```

```
//             }}
//           />
//           <TextField
//             label="password"
//             id="outlined-start-adornment"
//             sx={{ m: 1 }}
//             value={user?.userPassword}
//             type="password"
//             onChange={(e) => {
//               setUser({
//                 ...user,
//                 userPassword: e.target.value,
//               });
//             }}
//           />

//           <Stack direction={"row"} justifyContent="flex-end">
//             <Button
//               sx={{
//                 m: 1,
//                 width: "25ch",
//                 backgroundColor: "dodgerblue",
//                 color: "#FFF",
//                 fontWeight: "bold",
//               }}
//               variant={"contained"}
//             >
//               Submit
//             </Button>
//           </Stack>
//         </Box>
//       </Card>
//     </>
//   );
// }
```

Sensors

```
import { useEffect, useState } from "react";
import { v4 as uuidv4 } from "uuid";
import { firebaseDb } from "../../firebase/db";
import {
  Button,
  Card,
  Typography,
  Box,
```

```
  TextField,
  Stack,
  MenuItem,
  FormControl,
  InputLabel,
  Select,
} from "@mui/material";
import { IDeviceCreateModel } from "../../models/deviceModel";
import { firebaseAuth } from "../../firebase/auth";
import { removeDotsFromEmail } from
"../../utilities/removeDotsFromEmail";

export default function CreateSensorView() {
  const [device, setDevice] = useState({
    deviceName: "",
    deviceId: uuidv4(),
    deviceType: "",
  });

  const [currentUser, setCurrentUser] = useState({
    email: "",
    uid: "",
  });

  const createDevice = async () => {
    const dbPath = `${removeDotsFromEmail(currentUser.email)}/devices`;

    if (device.deviceType === "DHT11") {
      const payload1: IDeviceCreateModel = {
        deviceName: device.deviceName,
        deviceId: uuidv4(),
        deviceUserName: currentUser.email,
        deviceType: "DHT_TEMPERATURE",
      };

      const payload2: IDeviceCreateModel = {
        deviceName: device.deviceName,
        deviceId: uuidv4(),
        deviceUserName: currentUser.email,
        deviceType: "DHT_HUMIDITY",
      };

      await firebaseDb.create(dbPath, payload1);
      await createDeviceData(payload1.deviceId);

      await firebaseDb.create(dbPath, payload2);
      await createDeviceData(payload2.deviceId);
    } else {
      const payload: IDeviceCreateModel = {
        deviceName: device.deviceName,
        deviceId: device.deviceId,
        deviceUserName: currentUser.email,
        deviceType: device.deviceType,
      };
```

```
    await firebaseDb.create(dbPath, payload);

    await createDeviceData(device.deviceId);
  }
};

const createDeviceData = async (deviceId: string) => {
  const email = removeDotsFromEmail(currentUser.email);
  const dbPath = `${email}/deviceData/${deviceId}`;

  const payload = {
    value: 0,
    timeStamp: Date.now(),
  };

  await firebaseDb.create(dbPath, payload);
};

const handleSubmit = async () => {
  try {
    await createDevice();
    window.history.back();
  } catch (error: unknown) {
    console.log(error);
  }
};

const getCurrentUser = async () => {
  const user = await firebaseAuth.getCurrentUser();
  console.log(user);
  setCurrentUser({
    email: user?.email || "",
    uid: user?.uid || "",
  });
};

useEffect(() => {
  getCurrentUser();
}, []);

return (
  <>
    <Card
      sx={{
        mt: 5,
        p: 8,
      }}
    >
      <Typography
        variant="h4"
        marginBottom={5}
        color="primary"
        fontWeight={"bold"}
```

```
>
  Add Sensor
</Typography>
<Box
  component="form"
  style={{
    display: "flex",
    flexDirection: "column",
    justifyContent: "center",
  }}
>
  <TextField
    label="Sensor Name"
    id="outlined-start-adornment"
    sx={{ m: 1 }}
    value={device.deviceName}
    type="text"
    onChange={(e) => {
      setDevice({
        ...device,
        deviceName: e.target.value,
      });
    }}
  />
  <FormControl sx={{ m: 1 }}>
    <InputLabel>Sensor Type</InputLabel>
    <Select
      value={device.deviceType}
      label="Sensor Type"
      onChange={(e) =>
        setDevice({
          ...device,
          deviceType: e.target
            .value as IDeviceCreateModel["deviceType"],
        })
      }
    >
      <MenuItem value="LDR">LDR</MenuItem>
      <MenuItem value="SOIL_MOISTURE">SOIL_MOISTURE</MenuItem>
      <MenuItem value="DS18B20">DS18B20</MenuItem>
      <MenuItem value="DHT11">DHT11</MenuItem>
    </Select>
  </FormControl>
  <Stack direction={"row"} justifyContent="flex-end">
    <Button
      sx={{
        m: 1,
        width: "25ch",
        backgroundColor: "dodgerblue",
        color: "#FFF",
        fontWeight: "bold",
      }}
      variant={"contained"}
      onClick={handleSubmit}
```

```
            >
              Submit
            </Button>
          </Stack>
        </Box>
      </Card>
    </>
  );
}
```

Setting

```
/* eslint-disable @typescript-eslint/no-explicit-any */
import { useEffect, useState } from "react";
import { Box, Card, Typography, Button } from "@mui/material";
import { DataGrid, GridColDef } from "@mui/x-data-grid";
import { firebaseAuth } from "../../firebase/auth";
import { removeDotsFromEmail } from
"../../utilities/removeDotsFromEmail";
import { Prism as SyntaxHighlighter } from "react-syntax-highlighter";
import { vscDarkPlus } from "react-syntax-
highlighter/dist/esm/styles/prism";
import { firebaseConfig } from "../../firebase/configs";

export default function SettingView() {
  const [loading, setLoading] = useState(true);
  const [esp32Code, setEsp32Code] = useState<string>("");
  const currentUser = firebaseAuth.getCurrentUser();
  const email = removeDotsFromEmail(currentUser?.email ?? "");

  const esp32Config = {
    firebaseHost: firebaseConfig.databaseURL,
    firebaseAuth: firebaseConfig.apiKey,
    userEmail: email,
  };

  useEffect(() => {
    const fetchDeviceData = async () => {
      try {
        generateEsp32Code();
      } catch (error) {
        console.error("Error fetching device data:", error);
      } finally {
        setLoading(false);
      }
    };

    fetchDeviceData();
  }, [email]);

  const generateEsp32Code = () => {
    const code = `
#include <WiFi.h>
#include <FirebaseESP32.h>
```

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

#define SOIL_MOISTURE_PIN 16
#define LDR_PIN 35
#define DHT_PIN 4
#define DHT_TYPE DHT11
#define DS18B20_PIN 5

DHT dht(DHT_PIN, DHT_TYPE);
OneWire oneWire(DS18B20_PIN);
DallasTemperature ds18b20(&oneWire);

#define WIFI_SSID ""
#define WIFI_PASSWORD ""

#define FIREBASE_HOST "${firebaseConfig.databaseURL}"
#define FIREBASE_AUTH "${firebaseConfig.apiKey}"

#define USER_NAME "${esp32Config.userEmail}"

// isi sensor ID-nya supaya bisa kirim data
const char* SOIL_SENSOR_ID = "";
const char* LDR_SENSOR_ID  = "";
const char* DHT_TEMPERATURE_SENSOR_ID  = "";
const char* DHT_HUMIDITY_SENSOR_ID = "";
const char* DS18B20_SENSOR_ID = "";

const char* ntpServer = "id.pool.ntp.org";
const long gmtOffset_sec = 7 * 3600;
const int daylightOffset_sec = 0;

FirebaseData firebaseData;
FirebaseAuth auth;
FirebaseConfig config;
WiFiUDP udp;

void setup() {
  Serial.begin(115200);

  pinMode(SOIL_MOISTURE_PIN, INPUT);
  pinMode(LDR_PIN, INPUT);

  dht.begin();
  ds18b20.begin();

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi...");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
```

```
  Serial.println(" Connected!");

  config.host = FIREBASE_HOST;
  config.signer.tokens.legacy_token = FIREBASE_AUTH;
  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);

  Serial.println("Setting up NTP...");
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    Serial.println("Failed to obtain time from NTP.");
  } else {
    Serial.print("Current time: ");
    Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");
  }
}

void loop() {
  struct tm timeinfo;
  time_t epochTime;
  if (getLocalTime(&timeinfo)) {
    epochTime = mktime(&timeinfo);
  } else {
    epochTime = millis() / 1000;
    Serial.println("Failed to get NTP time for timestamp, using
uptime.");
  }

  Serial.println("time");
  Serial.println(epochTime);

  if (strlen(SOIL_SENSOR_ID) > 0) {
    int soilMoistureValue = analogRead(SOIL_MOISTURE_PIN);

    FirebaseJson soilJson;
    soilJson.set("value", soilMoistureValue);
    soilJson.set("timestamp", (unsigned long)epochTime);

    if (Firebase.pushJSON(firebaseData, String("/") + USER_NAME +
"/deviceData/" + SOIL_SENSOR_ID, soilJson)) {
      Serial.println("Soil Moisture sent to Firebase");
    } else {
      Serial.print("Soil Moisture data failed: ");
      Serial.println(firebaseData.errorReason());
    }

    Serial.print("Soil Moisture: ");
    Serial.println(soilMoistureValue);
  }

  if (strlen(LDR_SENSOR_ID) > 0) {
    int ldrValue = analogRead(LDR_PIN);
```

```
    FirebaseJson ldrJson;
    ldrJson.set("value", ldrValue);
    ldrJson.set("timestamp", (unsigned long)epochTime);

    if (Firebase.pushJSON(firebaseData, String("/") + USER_NAME +
"/deviceData/" + LDR_SENSOR_ID, ldrJson)) {
      Serial.println("LDR data sent.");
    } else {
      Serial.print("LDR data failed: ");
      Serial.println(firebaseData.errorReason());
    }

    Serial.print("LDR: ");
    Serial.println(ldrValue);
  }

  float temperature = dht.readTemperature();
  if (strlen(DHT_TEMPERATURE_SENSOR_ID) > 0 && !isnan(temperature)) {
    FirebaseJson tempJson;
    tempJson.set("value", temperature);
    tempJson.set("timestamp", (unsigned long)epochTime);

    if (Firebase.pushJSON(firebaseData, String("/") + USER_NAME +
"/deviceData/" + DHT_TEMPERATURE_SENSOR_ID, tempJson)) {
      Serial.println("Temperature sent.");
    } else {
      Serial.print("Temperature failed: ");
      Serial.println(firebaseData.errorReason());
    }

    Serial.print("Temperature: ");
    Serial.println(temperature);
  }

  float humidity = dht.readHumidity();
  if (strlen(DHT_HUMIDITY_SENSOR_ID) > 0 && !isnan(humidity)) {
    FirebaseJson humidJson;
    humidJson.set("value", humidity);
    humidJson.set("timestamp", (unsigned long)epochTime);

    if (Firebase.pushJSON(firebaseData, String("/") + USER_NAME +
"/deviceData/" + DHT_HUMIDITY_SENSOR_ID, humidJson)) {
      Serial.println("Humidity sent.");
    } else {
      Serial.print("Humidity failed: ");
      Serial.println(firebaseData.errorReason());
    }

    Serial.print("Humidity: ");
    Serial.println(humidity);
  }

  if (strlen(DS18B20_SENSOR_ID) > 0) {
    ds18b20.requestTemperatures();
```

```
    float dsTemp = ds18b20.getTempCByIndex(0);

    if (!isnan(dsTemp)) {
      FirebaseJson dsJson;
      dsJson.set("value", dsTemp);
      dsJson.set("timestamp", (unsigned long)epochTime);

      if (Firebase.pushJSON(firebaseData, String("/") + USER_NAME +
"/deviceData/" + DS18B20_SENSOR_ID, dsJson)) {
        Serial.println("DS18B20 temperature sent.");
      } else {
        Serial.print("DS18B20 failed: ");
        Serial.println(firebaseData.errorReason());
      }

      Serial.print("DS18B20 Temp: ");
      Serial.println(dsTemp);
    }
  }

  delay(60000 * 30); // Delay 60 seconds x 30 = 30 menit
}
`;

    setEsp32Code(code);
  };

  const handleCopyCode = () => {
    navigator.clipboard.writeText(esp32Code);
  };

  const configRows = [
    {
      id: 1,
      variable: "FIREBASE_HOST",
      value: esp32Config.firebaseHost,
    },
    { id: 2, variable: "FIREBASE_AUTH", value: esp32Config.firebaseAuth
},
    {
      id: 3,
      variable: "USER_NAME",
      value: esp32Config.userEmail,
    },
  ];

  return (
    <Box>
      <Typography variant="h4" gutterBottom>
        ESP32 Configuration
      </Typography>

      <Card sx={{ p: 3, mb: 3 }}>
        <Typography variant="h6" gutterBottom>
```

```
          Firmware Constants
        </Typography>
        <Box sx={{ width: "100%" }}>
          <DataGrid
            rows={configRows}
            columns={configColumns}
            disableRowSelectionOnClick
            hideFooterPagination
            loading={loading}
            autoHeight
          />
        </Box>
      </Card>

      <Card sx={{ p: 3, mb: 3 }}>
        <Typography variant="h6" gutterBottom>
          📌 Petunjuk Upload Kode ke ESP32
        </Typography>
        <Typography
          variant="body2"
          component="div"
          sx={{ whiteSpace: "pre-line" }}
        >
          {`
1. Install Libarary:
   - FirebaseESP32
   - NTPClient
   - DHT11
   - OneWire by Paul Stoffregen
   - DallasTemperature by Miles Burton
2. Sambungkan ESP32 ke komputer via USB
3. Copy kode di bawah dan paste ke Arduino IDE
4. Isi:
  # AUTHENTICATION
   - WIFI_SSID dan WIFI_PASSWORD
   - FIREBASE_HOST dan FIREBASE_AUTH
   - USER_NAME

  # SENSOR PIN
   - SOIL_MOISTURE_PIN
   - LDR_PIN
   - DHT_PIN
   - DS18B20_PIN

  # SENSOR ID
   - SOIL_SENSOR_ID
   - LDR_SENSOR_ID
   - DHT_TEMPERATURE_SENSOR_ID
   - DHT_HUMIDITY_SENSOR_ID
   - DS18B20_SENSOR_ID

5. Upload ke board
6. Lihat Serial Monitor, atur baudrate (115200)
    `}
```

```
        </Typography>
      </Card>

      <Card sx={{ p: 3, mb: 3 }}>
        <Typography variant="h6" gutterBottom>
          ESP32 Code
        </Typography>
        <Button variant="outlined" onClick={handleCopyCode} sx={{ mb: 2
}}>
          Copy Code
        </Button>
        <Box sx={{ borderRadius: 1, overflow: "hidden" }}>
          <SyntaxHighlighter
            language="cpp"
            style={vscDarkPlus}
            customStyle={{
              margin: 0,
              padding: "16px",
              maxHeight: "500px",
            }}
            showLineNumbers
          >
            {esp32Code}
          </SyntaxHighlighter>
        </Box>
      </Card>
    </Box>
  );
}

const configColumns: GridColDef[] = [
  {
    field: "variable",
    headerName: "Variable Name",
    flex: 1,
    renderCell: (params) => <code>{params.value}</code>,
  },
  {
    field: "value",
    headerName: "Value",
    flex: 2,
    renderCell: (params) => (
      <Typography variant="body2">{params.value}</Typography>
    ),
  },
];

pendro
const chartOptions: ApexOptions = {
  chart: {
    type: "area",
    height: 350,
    zoom: { enabled: false },
  },
```

```
  xaxis: {
    type: "datetime",
  },
  dataLabels: {
    enabled: false,
  },
  fill: {
    type: "gradient",
    gradient: {
      shadeIntensity: 1,
      opacityFrom: 0.7,
      opacityTo: 0.9,
      stops: [0, 100],
    },
  },
  tooltip: {
    x: { format: "dd/MM/yy HH:mm" },
  },
};

// Modular function to adjust timestamp to WIB (UTC+7)
const adjustToWIB = (timestamp: number): Date => {
  const WIB_OFFSET = 7 * 60 * 60 * 1000; // 7 hours in milliseconds
  return new Date(timestamp * 1000 + WIB_OFFSET);
};

// New modular function to filter data by date range
const filterDataByDateRange = (
  data: any[],
  startDate: Date | null,
  endDate: Date | null
): any[] => {
  if (!startDate || !endDate) return data; // Return all data if either
date is null
  return data.filter((entry) => entry.x >= startDate && entry.x <=
endDate);
};

export default function DetailSensorView() {
  const { email, sensorId } = useParams();
  const [chartSeries, setChartSeries] = useState<any[]>([]);
  const [deviceName, setDeviceName] = useState("Sensor");
  const [deviceType, setDeviceType] = useState("GENERIC");
  const [deviceValue, setDeviceValue] = useState<any[]>([]);
  const [allData, setAllData] = useState<any[]>([]); // Store unfiltered
data
  const [startDate, setStartDate] = useState<Date | null>(null); //
Default to null for all data
  const [endDate, setEndDate] = useState<Date | null>(null); // Default
to null for all data

  const fetchDeviceData = async () => {
    try {
      if (!sensorId) return;
```

```
      const path = `${email}/deviceData/${sensorId}`;
      const result: any = await firebaseDb.read(path);

      const sensorPath = `${email}/devices`;
      const sensors = (await firebaseDb.read(sensorPath)) as
IDeviceModel[];

      if (sensors) {
        const sensorDetil = Object.values(sensors).find(
          (device: IDeviceModel) => device.deviceId === sensorId
        );

        if (sensorDetil) {
          setDeviceName(sensorDetil?.deviceName);
          setDeviceType(sensorDetil?.deviceType);
        }
      }

      if (!result) {
        setChartSeries([]);
        setDeviceValue([]);
        setAllData([]);
        return;
      }

      const entries = Object.values(result) as any[];
      const formattedData = entries
        .filter((entry: any) => entry.timestamp &&
!isNaN(entry.timestamp))
        .map((entry: any) => ({
          x: adjustToWIB(entry.timestamp),
          y: entry.value,
        }))
        .filter((data) => !isNaN(data.x.getTime())); // Ensure valid
Date objects

      setAllData(formattedData); // Store unfiltered data
      const filteredData = filterDataByDateRange(
        formattedData,
        startDate,
        endDate
      ); // Apply date range filter
      setDeviceValue(filteredData);
      setChartSeries([
        {
          name: "Sensor Value",
          data: filteredData,
          group: "apexcharts-axis-0",
        },
      ]);
    } catch (error) {
      console.error("Error fetching device data:", error);
      setChartSeries([]);
```

```
      setDeviceValue([]);
      setAllData([]);
    }
  };
```

Lampiran 2 *Source Code* Pemograman ESP32 Arduino IDE

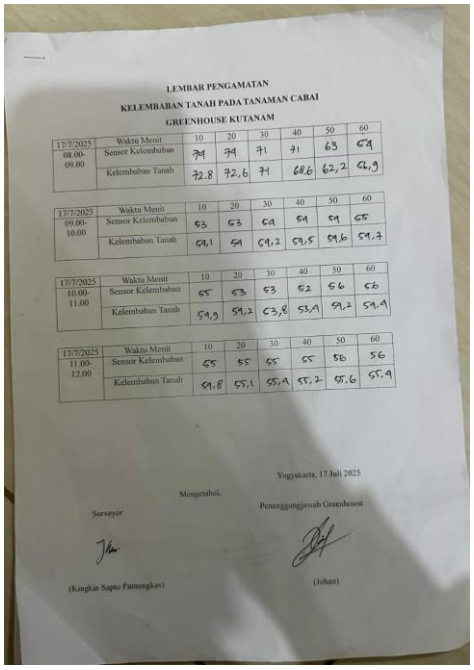
```
#include <WiFi.h>
#include <FirebaseESP32.h>
#include <WiFiUdp.h>
#include <DHT.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define SOIL_MOISTURE_PIN 32
#define LDR_PIN 35
#define DHT_PIN 2
#define DHT_TYPE DHT22
#define DS18B20_PIN 5

DHT dht(DHT_PIN, DHT_TYPE);
OneWire oneWire(DS18B20_PIN);
DallasTemperature ds18b20(&oneWire);

#define WIFI_SSID "Kutanam JP"
#define WIFI_PASSWORD "j4minsukses"

#define FIREBASE_HOST "https://tugas-akhir-iot-da5f2-default-rtdb.asia-
southeast1.firebasedatabase.app/"
#define FIREBASE_AUTH "AIzaSyDf61bPDPkIMsDRn9C9sP-lT50PRldEvpg"

#define USER_NAME "apendroanggara12@gmailcom"

// isi sensor ID-nya supaya bisa kirim data
const char* SOIL_SENSOR_ID = "3ba0bab5-bdc1-4ff5-86de-5071e5a7c987
";
const char* LDR_SENSOR_ID  = "";
const char* DHT_TEMPERATURE_SENSOR_ID  = "";
const char* DHT_HUMIDITY_SENSOR_ID = "";
const char* DS18B20_SENSOR_ID = "";

const char* ntpServer = "id.pool.ntp.org";
const long gmtOffset_sec = 7 * 3600;
const int daylightOffset_sec = 0;

FirebaseData firebaseData;
FirebaseAuth auth;
FirebaseConfig config;
WiFiUDP udp;

void setup() {
  Serial.begin(115200);

  pinMode(SOIL_MOISTURE_PIN, INPUT);
  pinMode(LDR_PIN, INPUT);
```

```
  dht.begin();
  ds18b20.begin();

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi...");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println(" Connected!");

  config.host = FIREBASE_HOST;
  config.signer.tokens.legacy_token = FIREBASE_AUTH;
  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);

  Serial.println("Setting up NTP...");
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    Serial.println("Failed to obtain time from NTP.");
  } else {
    Serial.print("Current time: ");
    Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");
  }
}

void loop() {
  struct tm timeinfo;
  time_t epochTime;
  if (getLocalTime(&timeinfo)) {
    epochTime = mktime(&timeinfo);
  } else {
    epochTime = millis() / 1000;
    Serial.println("Failed to get NTP time for timestamp, using
uptime.");
  }

  Serial.println("time");
  Serial.println(epochTime);

  if (strlen(SOIL_SENSOR_ID) > 0) {
    int soilMoistureValue = analogRead(SOIL_MOISTURE_PIN);

    FirebaseJson soilJson;
    soilJson.set("value", soilMoistureValue);
    soilJson.set("timestamp", (unsigned long)epochTime);

    if (Firebase.pushJSON(firebaseData, String("/") + USER_NAME +
"/deviceData/" + SOIL_SENSOR_ID, soilJson)) {
      Serial.println("Soil Moisture sent to Firebase");
    } else {
      Serial.print("Soil Moisture data failed: ");
```

```
      Serial.println(firebaseData.errorReason());
    }

    Serial.print("Soil Moisture: ");
    Serial.println(soilMoistureValue);
  }

  if (strlen(LDR_SENSOR_ID) > 0) {
    int ldrValue = analogRead(LDR_PIN);

    FirebaseJson ldrJson;
    ldrJson.set("value", ldrValue);
    ldrJson.set("timestamp", (unsigned long)epochTime);

    if (Firebase.pushJSON(firebaseData, String("/") + USER_NAME +
"/deviceData/" + LDR_SENSOR_ID, ldrJson)) {
      Serial.println("LDR data sent.");
    } else {
      Serial.print("LDR data failed: ");
      Serial.println(firebaseData.errorReason());
    }

    Serial.print("LDR: ");
    Serial.println(ldrValue);
  }

  float temperature = dht.readTemperature();
  if (strlen(DHT_TEMPERATURE_SENSOR_ID) > 0 && !isnan(temperature)) {
    FirebaseJson tempJson;
    tempJson.set("value", temperature);
    tempJson.set("timestamp", (unsigned long)epochTime);

    if (Firebase.pushJSON(firebaseData, String("/") + USER_NAME +
"/deviceData/" + DHT_TEMPERATURE_SENSOR_ID, tempJson)) {
      Serial.println("Temperature sent.");
    } else {
      Serial.print("Temperature failed: ");
      Serial.println(firebaseData.errorReason());
    }

    Serial.print("Temperature: ");
    Serial.println(temperature);
  }

  float humidity = dht.readHumidity();
  if (strlen(DHT_HUMIDITY_SENSOR_ID) > 0 && !isnan(humidity)) {
    FirebaseJson humidJson;
    humidJson.set("value", humidity);
    humidJson.set("timestamp", (unsigned long)epochTime);

    if (Firebase.pushJSON(firebaseData, String("/") + USER_NAME +
"/deviceData/" + DHT_HUMIDITY_SENSOR_ID, humidJson)) {
      Serial.println("Humidity sent.");
    } else {
```

```
      Serial.print("Humidity failed: ");
      Serial.println(firebaseData.errorReason());
    }

    Serial.print("Humidity: ");
    Serial.println(humidity);
  }

  if (strlen(DS18B20_SENSOR_ID) > 0) {
    ds18b20.requestTemperatures();
    float dsTemp = ds18b20.getTempCByIndex(0);

    if (!isnan(dsTemp)) {
      FirebaseJson dsJson;
      dsJson.set("value", dsTemp);
      dsJson.set("timestamp", (unsigned long)epochTime);

      if (Firebase.pushJSON(firebaseData, String("/") + USER_NAME +
"/deviceData/" + DS18B20_SENSOR_ID, dsJson)) {
        Serial.println("DS18B20 temperature sent.");
      } else {
        Serial.print("DS18B20 failed: ");
        Serial.println(firebaseData.errorReason());
      }

      Serial.print("DS18B20 Temp: ");
      Serial.println(dsTemp);
    }
  }

  delay(6000000); // Delay 10 menit
}
```

Lampiran 3 Tabel Hasil Pengamatan Di Greenhouse



LEMBAR PENGAMATAN
KELEMBABAN TANAH PADA TANAMAN CABAI
GREENHOUSE KUTANAM

17/7/2025	Waktu Menit	10	20	30	40	50	60
08.00-09.00	Sensor Kelembaban	79	79	71	71	63	59
	Kelembaban Tanah	72.8	72,6	71	68,6	62,2	61,9

17/7/2025	Waktu Menit	10	20	30	40	50	60
09.00-10.00	Sensor Kelembaban	63	63	69	69	69	65
	Kelembaban Tanah	59,1	59	59,2	59,5	59,6	59,7

17/7/2025	Waktu Menit	10	20	30	40	50	60
10.00-11.00	Sensor Kelembaban	65	63	63	62	66	66
	Kelembaban Tanah	59,3	59,2	63,8	63,9	59,2	59,9

17/7/2025	Waktu Menit	10	20	30	40	50	60
11.00-12.00	Sensor Kelembaban	65	55	55	55	56	56
	Kelembaban Tanah	59,8	55,1	55,9	55,2	55,6	55,9

Yogyakarta, 17 Juli 2025

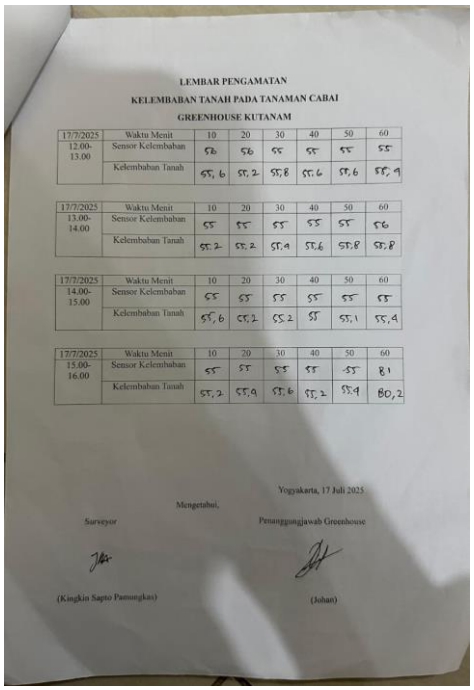
Mengetahui,

Surveyor

(Kingkin Sapto Pamungkas)

Penanggungjawab Greenhouse

(Johan)



LEMBAR PENGAMATAN
KELEMBABAN TANAH PADA TANAMAN CABAI
GREENHOUSE KUTANAM

17/7/2025	Waktu Menit	10	20	30	40	50	60
12.00-13.00	Sensor Kelembaban	56	56	55	55	55	55
	Kelembaban Tanah	55,6	55,2	55,8	55,6	55,6	55,9

17/7/2025	Waktu Menit	10	20	30	40	50	60
13.00-14.00	Sensor Kelembaban	55	55	55	55	55	56
	Kelembaban Tanah	55,2	55,2	55,9	55,6	55,8	55,8

17/7/2025	Waktu Menit	10	20	30	40	50	60
14.00-15.00	Sensor Kelembaban	55	55	55	55	55	55
	Kelembaban Tanah	55,6	55,2	55,2	55	55,1	55,9

17/7/2025	Waktu Menit	10	20	30	40	50	60
15.00-16.00	Sensor Kelembaban	55	55	55	55	55	81
	Kelembaban Tanah	55,2	55,9	55,6	55,2	55,9	80,2

Yogyakarta, 17 Juli 2025

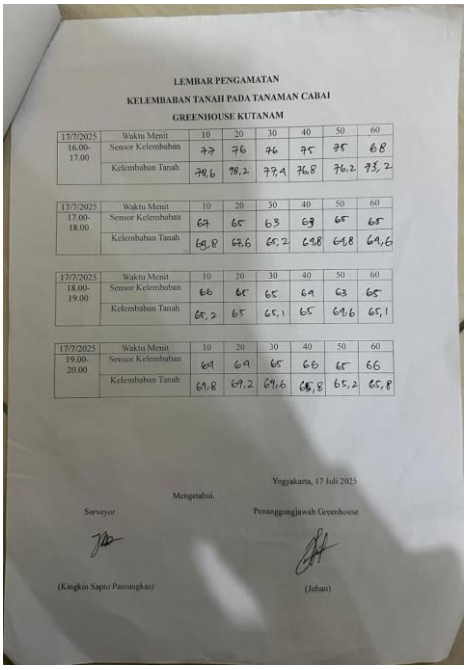
Mengetahui,

Surveyor

(Kingkin Sapto Pamungkas)

Penanggungjawab Greenhouse

(Johan)



LEMBAR PENGAMATAN

KELEMBABAN TANAH PADA TANAMAN CABAI

GREENHOUSE KUTANAM

17/7/2025	Waktu Menit	10	20	30	40	50	60
16.00-17.00	Sensor Kelembaban	77	76	76	75	75	68
	Kelembaban Tanah	78,6	78,2	77,9	76,8	76,2	73,2

17/7/2025	Waktu Menit	10	20	30	40	50	60
17.00-18.00	Sensor Kelembaban	67	65	63	69	65	65
	Kelembaban Tanah	68,8	67,6	65,2	69,8	69,8	69,6

17/7/2025	Waktu Menit	10	20	30	40	50	60
18.00-19.00	Sensor Kelembaban	66	65	65	69	63	65
	Kelembaban Tanah	65,2	65	65,1	65	69,6	65,1

17/7/2025	Waktu Menit	10	20	30	40	50	60
19.00-20.00	Sensor Kelembaban	69	69	65	66	65	66
	Kelembaban Tanah	69,8	69,2	69,6	65,8	65,2	65,8

Yogyakarta, 17 Juli 2025

Mengetahui,

Surveyor

(Kingkin Sapto Pamungkas)

Penanggungjawab Greenhouse

(Johan)

Lampiran 4 Dokumentasi Penelitian