LAPORAN TUGAS AKHIR

PEMANTAUAN SUHU DAN KELEMBABAN RUANGAN SENSOR *DIGITAL HUMIDITY AND TEMPERATURE*

Mata Kuliah Internet of Things

Disusun Oleh:

Achmad Rizky Zulkarnain	**2125250045**
Robert Antonius	**2125250057**
Rivaldo	**2125250065**

Program Studi Informatika
Fakultas Ilmu Komputer dan Rekayasa
Universitas Multi Data Palembang
Palembang
2024

ABSTRAK

Suhu adalah panas atau dinginnya suatu tempat dengan satuan derajat, lalu kelembapan adalah banyaknya air di dalam suatu titik tempat tersebut. Suhu dan kelembapan ini bisa menjadi kunci informasi untuk mengetahui tempat yang berpotensi terjadi kebakaran yang akan bisa mengurangi resiko kerusakan yang akan terjadi jika informasi kebakaran bisa cepat didapatkan. Di penelitian ini bertujuan untuk mengetahui suhu dan kelembapan udara yang akan dipantau melalui *smartphone* dengan bantuan sensor DHT11 yang jika terjadi kebakaran maka *buzzer* akan mengeluarkan bunyi untuk memberi tahu bahwa telah terjadi kebakaran. Adapun tahapan proses yang kami lakukan yaitu perencanaan, perancangan, lalu tahap terakhir yaitu pengujian. Di dalam penelitian ini dilakukan untuk menangkap angka suhu dan kelembapan dalam udara pada titik tertentu dengan 2 sensor DHT11 yang akan dikirim ke *database firebase* secara *real time* melalui NodeMCUv3. Setelah disimpan, data tersebut akan ditampilkan melalui *web* yang dibuat menggunakan *javascript* dengan *framework react* yang dapat dipantau melalui *smartphone*. Hasil pengujian kondisi suhu dan kelembapan untuk sensor DHT11 sebelah kiri dengan suhu rata-rata di 27,64 dan kelembapan di rata-rata 68.20, sedangkan untu DHT11 sebelah kanan dengan suhu rata-rata 27,82 dan kelembapan di rata-rata 61,63.

Kata Kunci: DHT11, Firebase, JavaScript, NodeMCUv3, React

Temperature is the heat or coldness of a place with units of degrees, then humidity is the amount of water in a point of the place. This temperature and humidity can be key information to find out where there is a potential fire that will be able to reduce the risk of damage that will occur if fire information can be quickly obtained. In this study aims to determine the temperature and humidity of the air which will be monitored via a smartphone with the help of a DHT11 sensor which in the event of a fire, a buzzer will make a sound to notify that a fire has occurred. The stages of the process that we do are planning, designing, then the last stage is testing. In this research, it is carried out to capture the temperature and humidity figures in the air at a certain point with 2 DHT11 sensors which will be sent to the firebase database in real time via NodeMCUv3. After being stored, the data will be displayed through a web created using javascript with the react framework that can be monitored via a smartphone. The test results of temperature and humidity conditions for the left DHT11 sensor with an average temperature of 27.64 and an average humidity of 68.20, while for the right DHT11 with an average temperature of 27.82 and an average humidity of 61.63.

***Keywords**: DHT11, Firebase, JavaScript, NodeMCUv3, React*

KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, yang memungkinkan kami menyelesaikan laporan tugas projek akhir mata kuliah Internet of Things (IoT) ini yang berjudul "PEMANTAUAN SUHU DAN KELEMBABAN RUANGAN SENSOR *DIGITAL HUMIDITY AND TEMPERATURE*". Laporan ini disusun sebagai salah satu bentuk penerapan ilmu yang telah kami peroleh selama perkuliahan. Kami mengucapkan terima kasih kepada dosen pengampu, Dedy Hermanto, S.Kom, M.T.I., atas bimbingan dan dukungannya selama pengerjaan *project* ini, dan kami menyadari bahwa banyak kekurangan dari *project* tugas akhir ini, oleh karena itu kami sangat berterima kasih atas dukungan dan bantuan dari dosen pengampu kami dalam pembangunan laporan ini. Penulis berharap agar laporan ini bisa menjadi manfaat bagi pembaca.

Palembang, 27 Desember 2024

Penulis

DAFTAR ISI

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Suhu adalah panas atau dinginnya udara yang dinyatakan dengan satuan derajat tertentu, sedangkan kelembaban adalah banyaknya air yang terkandung dalam udara, biasanya dinyatakan dengan persentase (Dewi, 2021). Suhu dan kelembaban merupakan suatu alat ukur yang sangat berguna untuk berbagai macam hal. contohnya untuk mengetahui situasi dari suatu ruangan yang akan sangat berguna untuk preservasi bahan pangan, obat-obatan, server dan mengetahui kondisi ruangan, khusunya di rumah tempat tinggal. Alat ukur suhu dan kelembaban masih jarang ditemui pada rumah-rumah di indonesia karena masih dianggap suatu hal yang tidak terlalu berguna di rumah. Buktinya bisa dilihat dari sisi smart home pada artikel oleh (Depok Pos, 2022) penetrasi smart home pada rumah tangga saat ini di Indonesia pada tahun 2022 sebesar 11,4% dan diperkirakan akan mencapai 18,3% pada tahun 2026, sedangkan jika dilihat data oleh (BPS, 2024) terdapat 84,95% keluarga di indonesia yang sudah memiliki rumah sendiri.

Informasi tentang suhu dan kelembaban di suatu ruangan sangatlah penting, menurut penelitian oleh (Hidayat & Sari, 2021) untuk menghabiskan waktu didalam ruangan/gedung manusia akan mencari kondisi yang nyaman dalam kehidupannya. Oleh karenanya salah satu faktor kenyamanan dan kesehatan yang optimal pada manusia bergantung pada suhu yang ada disekitarnya. Salah satu resiko yang paling fatal jika tidak mengetahui suhu di dalam ruangan adalah tidak

tahunya terjadi kebakaran. Resiko tidak adanya informasi tentang suhu dan kelembaban di dalam ruangan adalah tidak mengetahui adanya kebakaran. Kebakaran bisa terjadi saat tanpa ada yang bisa memprediksi, apabila tidak diatasi dengan cepat maka akan mengakibatkan kerugian yang sangat besar (Anam dkk., 2020). Dalam banyak kasus, hanya perlu waktu tiga menit dari saat kebakaran dimulai hingga seluruh ruangan dilalap api, yang berakibat sangat fatal (Lilley, 2012). Oleh karena itu, penting untuk memiliki sistem yang dapat mendeteksi perubahan suhu secara cepat dan memberikan peringatan dini kepada penghuni. Dengan adanya informasi suhu yang terpantau secara real-time.

Pentingnya suhu dan kelembaban menjadi suatu acuan untuk mengukur nilai suhu dan kelembaban di dalam ruangan. Masalah ini bisa dipecahkan dengan menggunakan termometer, namun menurut penelitian (Nugraha dkk., 2023) monitoring suhu dengan cara manual dengan mencatatnya sesuai dari termometer, tetapi hal ini belum dapat memonitoring suhu secara real-time. Karena jika termometer ditempel di ruangan maka kita tidak dapat mengetahui suhu ruangan jika kita berada di luar ruangan. Oleh karena itu diperlukan alat yang dapat mengukur suhu dan kelembaban dan menginformasikan ke kita secara real time. Masalah ini bisa diselesaikan dengan menggunakan alat IoT.

Menurut penilitan oleh (Waher, 2021) Internet of Things adalah sesuatu yang terhubung langsung ke internet tanpa perantara manusia dan dapat bekerja secara otomatis seperti kemampuan mengambil data secara otomatis. Menurut penelitian oleh (Manyika dkk., 2015) Internet of Things adalah sensor yang terhubung dengan jaringan untuk sistem komputasi sehingga dapat melakukan

pemantauan atau tindakan dari objek dan perangkat yang terhubung. Pangsa pasar Internet of Things (IoT) di Indonesia diperkirakan berkembang dengan nilai mencapai Rp 444 triliun di tahun 2022. Nilai tersebut disumbang dari berbagai macam teknologi, salah satunya adalah perangkat IoT 56 triliun (Saepudin, 2022). Salah satu alat IoT untuk menyelesaikan masalah ini adalah sensor suhu dan kelembaban yaitu DHT11. Menurut (Adiptya & Wibawanto, 2013) DHT11 memiliki output digital yang sudah terkalibrasi. Sensor ini terdiri dari komponen pengukur kelembaban tipe resistive dan pengukuran suhu melalui NTC serta terhubung dengan 8-*bit* uC sehingga memberikan hasil yang cukup baik, kecepatan respon yang cukup, memiliki ketahanan yang baik terhadap interferensi dan cukup murah dalam harga. Ini menjadikan DHT11 sebagai salah satu sensor yang sangat baik untuk penelitian ini, karena data yang didapatkan oleh sensor DHT11 nantinya akan ditampilkan dalam bentuk data yang dapat dikirimkan melalui internet.

Untuk penelitian terkait terdapat penelitian (Rangan dkk., 2020) bertujuan membangun sistem monitoring berbasis IoT menggunakan sensor DHT11 dan NodeMCU ESP8266, dengan data ditampilkan pada website. Sistem dikembangkan menggunakan metode prototipe untuk memastikan hasil sesuai kebutuhan, memanfaatkan perangkat lunak seperti Arduino IDE, XAMPP, dan Sublime. Dengan sistem ini, suhu dan kelembaban laboratorium dapat dipantau secara real-time, meningkatkan keamanan dan efisiensi. Terdapat juga penelitian (Baehaqi dkk., 2023) bertujuan membandingkan performa sensor suhu DHT11 dan DS18B20 dalam pengukuran suhu pada ruangan server. Sistem dikembangkan menggunakan pendekatan eksperimental untuk menguji nilai akurasi dan tingkat kesalahan

pembacaan kedua sensor, yang dibandingkan dengan alat Fluke 179 True-RMS Digital Multimeter. Dari 10 kali percobaan, hasil pengujian menunjukkan rata-rata tingkat kesalahan sebesar 3,37% untuk sensor DHT11 dan 1,17% untuk sensor DS18B20, dengan nilai akurasi masing-masing 96,63% dan 98,83%. Penelitian ini memberikan data objektif mengenai kelebihan dan kekurangan kedua sensor dalam aplikasi pengukuran suhu. Terakhir terdapat penelitian (Aspari dkk., 2024) yang bertujuan untuk membangun sistem monitoring suhu dan kelembaban pada inkubator penetas telur ayam berbasis IoT menggunakan aplikasi Android. Sistem ini memanfaatkan NodeMCU ESP8266 sebagai mikrokontroler dan sensor DHT11 untuk mendeteksi suhu dan kelembaban secara otomatis. Dengan aplikasi Android sebagai media monitoring, pengguna dapat memantau dan mengendalikan inkubator dari jarak jauh melalui akses internet. Sistem ini dirancang untuk meningkatkan efisiensi dan efektivitas dalam pengawasan inkubator penetas telur ayam.

Oleh karena itu, diperlukan solusi yang lebih canggih dengan memanfaatkan teknologi Internet of Things (IoT). IoT memungkinkan pemantauan suhu dan kelembaban secara otomatis dan real-time melalui sensor yang terhubung dengan jaringan internet. Salah satu sensor yang ideal untuk kebutuhan ini adalah DHT11, yang memiliki keunggulan seperti kecepatan respons yang baik, ketahanan terhadap interferensi, dan harga yang terjangkau. Dengan implementasi IoT menggunakan sensor DHT11, informasi suhu dan kelembaban dapat diperoleh secara real-time, sehingga memberikan solusi praktis dan efisien untuk menjaga kenyamanan, kesehatan, dan keamanan rumah tangga

1.2 Permasalahan

Berdasarkan latar belakang yang telah dijelaskan, terdapat beberapa permasalahan yang dapat dilihat. Pertama pada alat ukur suhu dan kelembaban yang masih jarang digunakan di rumah-rumah di Indonesia karena dianggap kurang bermanfaat bagi lingkungan rumah tangga. Padahal, informasi suhu dan kelembaban sangat penting untuk menjaga kenyamanan dan kesehatan penghuni rumah. Selanjutnya penggunaan termometer sebagai alat tradisional memiliki keterbatasan, karena pengukuran suhu secara manual tidak memungkinkan pemantauan secara real-time, terutama jika penghuni sedang berada di luar ruangan. Ketiga, ketidaktahuan mengenai kondisi suhu dan kelembaban ruangan dapat meningkatkan risiko kesehatan dan keamanan, seperti ketidaknyamanan penghuni hingga keterlambatan dalam mendeteksi kebakaran.

Masalah-masalah ini dapat diselesaikan dengan menggunakan IoT dengan implementasi DHT11 untuk mengukur suhu dan kelembaban dari suatu ruangan untuk mencegah terjadinya kebakaran dan membuat rumah menjadi lebih nyaman dan aman.

1.3 Tujuan

Dalam penelitian ini ditentukan tujuan dari penelitian yang dirangkum sebagai berikut:

1. Mengembangkan alat berbasis Internet of Things (IoT) menggunakan sensor DHT11 untuk memantau suhu dan kelembaban secara real-time.
2. Mempromosikan alat berbasis Internet of Things (IoT) agar dapat mengetahui potensi dari alat IoT serta efektivitas dari alat IoT
3. Memberikan solusi praktis dan efisien yaitu teknologi Internet of Things (IoT) menggunakan sensor DHT11 untuk meningkatkan kenyamanan, kesehatan, dan keamanan di rumah tangga.

1.4 Manfaat

Manfaat dari implementasi alat berbasis IoT ini adalah menyediakan informasi suhu dan kelembaban secara real-time, sehingga penghuni rumah dapat menciptakan lingkungan yang nyaman dan sehat. Alat ini juga dapat membantu mendeteksi potensi bahaya, seperti risiko kebakaran, secara lebih dini. Selain itu, solusi ini mendukung adopsi teknologi IoT di lingkungan rumah tangga di Indonesia, sekaligus memanfaatkan sensor DHT11 yang memiliki keunggulan dalam akurasi, kecepatan respons, dan harga yang terjangkau.

1.5 Ruang Lingkup Proyek

Berikut ini merupakan ruang lingkup atau batasan pada proyek antara lain:

1. Sistem merekam kelembaban dan suhu pada ruangan dengan sensor DHT11 dan akan membunyikan *buzzer* saat kelembaban dan suhu melampaui *threshold* yang telah ditentukan.
2. Hasil deteksi dari sensor DHT akan disimpan pada *Firebase Firestore* untuk data rekaman sensor setiap 10 detik dan pada *Firebase Realtime Database* untuk data *live* sensor.
3. Data sensor akan ditampilkan pada GUI berbasis web yang dibangun dengan ReactJS *framework* untuk JavaScript.
4. GUI dapat diakses melalui *internet* pada perangkat *smartphone* atau komputer.

BAB 2

TINJAUAN PUSTAKA

2.1 NodeMCU

NodeMCU adalah mikrokontroler yang dilengkapi dengan modul WiFi ESP8266 dan merupakan salah satu komponen utama Internet of Things (IoT) (Tekkom, 2023). NodeMCU memiliki keunggulan sudah memiliki WIFI, sehingga sangat cocok untuk proyek IoT. NodeMCU ini sifatnya open source yang mengakibatkan banyak produsen memproduksinya dan mengembangkannya (Priyono, 2017).

Versi 3 NodeMCU menggunakan modul ESP-12E (ESP8266MOD) dan merupakan papan pengembangan yang mudah digunakan dengan pin analog dan digital, adapter *universal serial bus* (USB) ke *serial* dengan modul CH340g, dan soket *micro* USB (Loginov, 2018).

2.2 DHT11

Sensor DHT11 adalah modul sensor yang berfungsi untuk mensensing objek suhu dan kelembaban yang memiliki output tegangan analog yang dapat diolah lebih lanjut menggunakan mikrokontroler (Rangan dkk., 2020). Sensor DHT11 memiliki elemen pengukur suhu dan kelembaban yang sensitif tehadap perubahan suhu dan kelembaban disekitarnya (Pramoedya, t.t.). Sensor ini memiliki keunggulan berupa kalibrasi sinyal digital bawaan yang memberikan data yang stabil, mudah diakses, dan cukup akurat. DHT11 cocok untuk aplikasi dengan

kebutuhan pengukuran lingkungan yang sederhana. DHT11 beroperasi pada tegangan 3,3V hingga 5,5V dan memiliki frekuensi sampling sebesar 1 Hz (Zella, t.t.). Pengambilan data dari sensor DHT11 setiap 1 detik. pengukuran suhu antara 0°C hingga 50°C dengan akurasi ±2°C, serta kelembapan relatif (RH) antara 20% hingga 90% dengan akurasi ±5% (IoT Kece, 2022).

2.3 *Buzzer*

Buzzer adalah komponen elektronik yang digunakan untuk menghasilkan suara atau bip. Ini adalah perangkat output yang mengubah sinyal listrik menjadi suara. Buzzer sering digunakan dalam perangkat rumah tangga, jam alarm, komputer, dan banyak perangkat elektronik lainnya sebagai cara untuk memberikan indikasi audio kepada pengguna (Virtualab IoT, t.t.).

Buzzer terdiri dari dua jenis, yaitu passive buzzer dan active buzzer. Passive buzzer adalah jenis buzzer yang tidak dapat menghasilkan suara secara mandiri. Dengan kata lain, ketika diberi arus listrik, komponen ini tidak akan mengeluarkan suara, mirip dengan cara kerja speaker aktif. Sementara itu, active buzzer merupakan jenis buzzer yang dapat menghasilkan suara secara mandiri. Ketika dialiri listrik, komponen ini langsung menghasilkan suara tanpa memerlukan rangkaian tambahan (Hermawan & Abdurrohman, 2020).

2.4 Kabel *Jumper*

Kabel jumper adalah kabel elektrik yang memiliki pin konektor di kedua ujungnya, memungkinkan untuk menghubungkan dua komponen yang melibatkan Arduino tanpa perlu melakukan solder. Fungsi utama dari kabel jumper ini adalah

sebagai konduktor listrik untuk menyambungkan rangkaian listrik (Prastyo, 2022). Kabel ini umumnya terbuat dari inti tembaga yang dilapisi dengan isolasi plastik yang fleksibel dan tersedia dalam berbagai jenis ujung, seperti male-to-male, male-to-female, atau female-to-female, yang memungkinkan koneksi antara pin header, breadboard, atau perangkat lainnya (Kumparan, 2023).

2.5 Universal PCB

Universal PCB (printed circuit board) adalah PCB yang dirancang dan diproduksi sesuai dengan spesifikasi standar sehingga kompatibel dengan berbagai jenis komponen dan aplikasi. Alih-alih dirancang khusus untuk produk atau sistem tertentu, PCB universal menyediakan platform yang dapat digunakan kembali untuk berbagai proyek elektronik dengan sedikit modifikasi. Keberadaan PCB universal memungkinkan pembuatan prototipe lebih cepat dan mempermudah pengembangan perangkat elektronik dan gadget (Artist-3D, 2023). Universal PCB terdapat banyak jenis seperti breadboard, stripboard, perf board, protoboard, dan masih banyak lagi. Pada penelitian ini yang dipakai adalah jenis breadboard.

2.6 Kabel USB

Kabel USB adalah salah satu jenis penghubung elektronik yang paling umum digunakan saat ini, berfungsi untuk mengatur koneksi dan komunikasi antara komputer dan perangkat eksternal (Kabasi, 2021). Dalam pengembangan perangkat berbasis Internet of Things (IoT), kabel USB sering digunakan untuk mentransfer program ke NodeMCU serta menyediakan daya untuk menjalankan perangkat tersebut. Kabel USB terdiri dari beberapa jenis konektor, seperti USB-A, USB-B, USB-C, dan micro-USB, yang disesuaikan dengan kebutuhan perangkat IoT.

2.7 Power Bank

Powerbank adalah perangkat genggam yang berfungsi sebagai penyimpan energi listrik untuk digunakan sebagai sumber daya cadangan dalam mengisi ulang baterai perangkat elektronik, seperti ponsel, tablet, atau perangkat lainnya. Biasanya Powerbank berfungsi sebagai perangkat pengisi daya untuk gadget ketika pengguna berada di luar atau jauh dari sumber listrik. Secara fungsi, powerbank dapat diidentifikasi sebagai alat penyimpan daya yang berperan seperti baterai cadangan yang dapat digunakan kapan saja diperlukan (Apriani dkk., 2021).

2.8 Visual Studio Code

Visual Studio Code (VS Code) adalah sebuah teks editor untuk menulis kode program yang didukung oleh bahasa pemrograman JavaScript, Typescript, dan Node.js, serta bahasa pemrograman lainnya dengan bantuan plugin yang dapat

dipasang via marketplace Visual Studio Code (seperti C++, C#, Python, Go, Java, dan seterusnya) (Yuliadi dkk., 2021).

Salah satu ekstensi populer untuk pengembangan perangkat IoT yaitu *extension* PlatformIO pada Visual Studio Code digunakan untuk pembangunan perangkat lunak untuk perangkat ESP8266. PlatformIO pada Visual Studio Code digunakan untuk mempermudah pembangunan perangkat lunak pada perangkat ESP8266 dengan menyediakan fitur seperti manajemen dependensi, konfigurasi proyek, debugging, dan kompilasi kode secara otomatis. PlatformIO mendukung berbagai framework, seperti Arduino dan ESP-IDF (Visual Studio Marketplace, 2024).

2.9 Arduino

Arduino adalah platform prototyping elektronik berbasis open-source hardware yang menggabungkan perangkat keras dan perangkat lunak yang fleksibel serta mudah digunakan. Bahasa pemrograman Arduino adalah Bahasa perograman yang umum digunakan untuk membuat perangkat lunak yang ditanamkan pada Arduino board. Bahasa pemrograman Arduino mirip dengan Bahasa pemrograman C++ (Kusuma dkk., 2020).

2.10 Firebase

Firebase merupakan platform untuk aplikasi realtime. Ketika data berubah, maka aplikasi yang terhubung dengan firebase akan meng-update secara langsung melalui setiap device (perangkat) baik website ataupun mobile (Google Firebase, t.t.). Firebase menyediakan library yang lengkap untuk berbagai platform web dan

mobile, serta dapat diintegrasikan dengan berbagai framework lain seperti Node.js, Java, JavaScript, dan sebagainya. Manfaat Firebase juga berlaku untuk aplikasi web dengan secara signifikan mengurangi biaya infrastruktur, bahkan berpotensi menjadi nol, sambil tetap menawarkan kemampuan yang kuat. Firebase menyediakan fitur keamanan yang kuat, pengembangan dan penyebaran API yang cepat, serta kemampuan untuk meng-host file statis di CDN global (BlowStack, 2022).

2.11 React

ReactJS adalah pustaka JavaScript yang digunakan untuk mengembangkan komponen antarmuka pengguna (UI) yang dapat digunakan kembali. Berdasarkan dokumentasi resmi React, berikut adalah definisi React adalah perpustakaan untuk membangun antarmuka pengguna modular (Aggarwal, 2018). Dari sisi pengembang penggunaan ReactJs sangatlah membantu, selain mudah dipelajari, ReactJS juga mendapatkan hasil pengujian kinerja yang bagus, dan React cocok untuk digunakan dalam pembuatan website modular (Murti & Sujarwo, 2021). Kode yang dibuat pada React JS bisa digunakan kembali jika dibutuhkan seperti melakukan daur ulang, hal ini dapat menghemat waktu pengembangan yang membutuhkan kecepatan pengembangan (Nasution & Iswari, 2021).

BAB 3

METODE

3.1 Perangkat yang Digunakan

Dalam proses pembangunan *project* ini, terdapat beberapa alat dan *tools* yang digunakan oleh penulis, seperti *hardware* yang digunakan maupun *software* yang akan digunakan.

3.1.1 Perangkat Keras (*Hardware*)

Berikut beberapa *hardware* yang digunakan dalam pengembangan *project*.

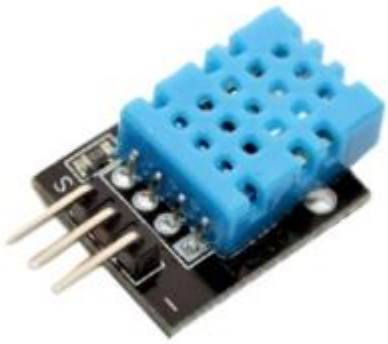
A. NodeMCUv3

NodeMCUv3 ini merupakan perangkat utama yang digunakan dalam *project* ini, berperan sebagai alat yang menerima data dari sensor DHT dan *buzzer* lalu mengirimkannya ke *firebase*. Gambar 3.1 merupakan contoh gambar dari perangkat NodeMCUv3 yang digunakan pada *project* ini.

Gambar 3.1 NodeMCUv3 (ESP8266MOD)

B. Sensor *Digital Humidity and Temperature*

Sensor DHT yang kami gunakan adalah DHT11 yang berfungsi sebagai sensor untuk mendapatkan nilai kelembapan dan suhu yang akan dikirim ke NodeMCUv3. Seperti pada Gambar 3.2 merupakan contoh dari DHT11 yang kami gunakan Gambar 3.2



Gambar 3.2 *Digital Humidity and Temperature* 11 (DHT11) Sensor

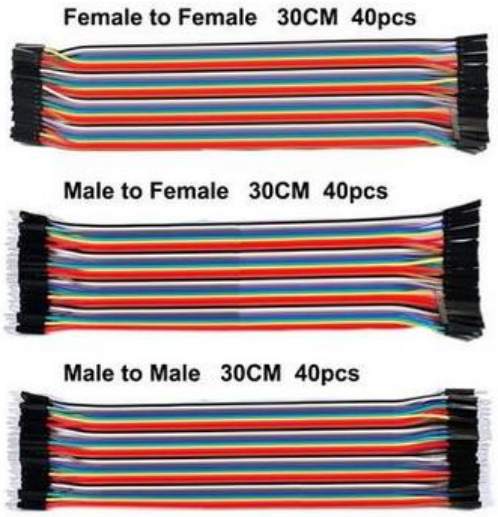
C. *Buzzer*

Buzzer yang digunakan adalah *buzzer* aktif yang digunakan untuk mengeluarkan suara jika suhu yang ditangkap dari sensor DHT11 mencapai suhu tertentu. Gambar 3.3 adalah gambar Buzzer yang digunakan pada *project* ini.

Gambar 3.3 *Buzzer*

D. Kabel *Jumper*

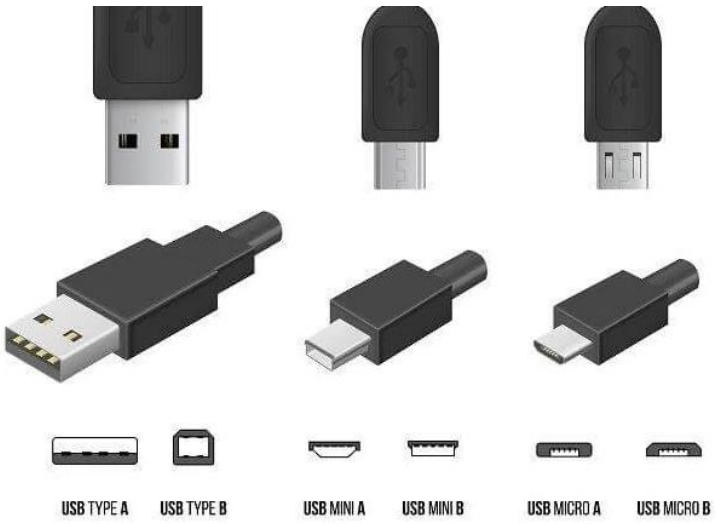
Kabel *jumper* ini berfungsi sebagai media penguhubung antar NodeMCUv3 dengan perangkat keras lain seperti DHT11 dan *buzzer*. Pada Gambar 3.4 adalah gambar kabel *jumper* yang kami gunakan pada *project* ini.



Gambar 3.4 *Jumper male to female*

E. Kabel USB

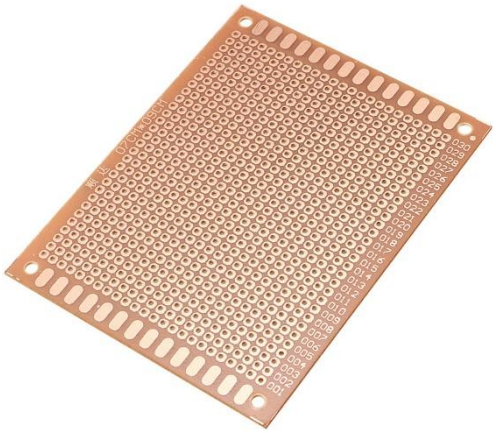
Kabel USB yang digunakan adalah *micro* USB berfungsi sebagai media penghubung untuk menyalurkan listrik dari *powerbank* ke NodeMCUv3. Pada Gambar 3.5 dilampirkan contoh perangkat kabel USB.



Gambar 3.5 Kabel USB

F. *Universal* PCB

Universal PCB ini menjadi media penghubung dan wadah untuk menaruh NodeMCUv3 ini dengan perangkat keras lainnya. Pada Gambar 3.6 dilampirkan contoh papan *universal* PCB.



Gambar 3.6 *Universal* PCB

G. *Power Bank*

Powerbank ini menjadi *power supply* sebagai penyedia listrik untuk NodeMCUv3 pada *project* ini. Gambar 3.7 merupakan contoh *powerbank* yang kami gunakan pada *project* ini.

Gambar 3.7 *Power Bank*

3.1.2 Perangkat Lunak (*Software*)

Berikut perangkat lunak yang dipakai dalam pembangunan *project* ini.

A. Visual Studio Code

Visual Studio Code atau *VSC* merupakan IDE yang digunakan untuk membuat program NodeMCUv3 dan juga program react.

B. Firebase

Firebase ini digunakan untuk tempat penyimpanan nilai angka yang ditangkap oleh perangkat DHT11 lalu dikirim oleh NodeMCUv3 secara *realtime.*

C. React

React adalah *framework* yang menggunakan bahasa pemograman JavaScript, yang digunakan untuk membuat *laman dashboard* pemantauan suhu dan kelembapan udara pada *project* ini.

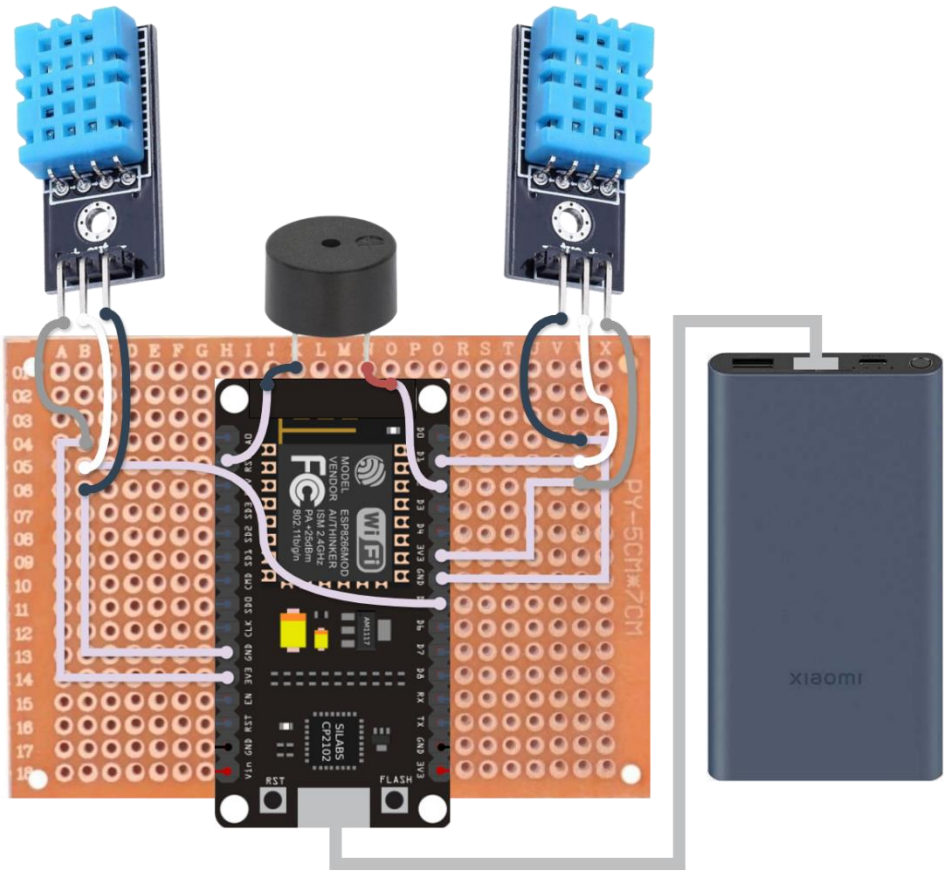
3.2 Pembangunan Perangkat

Berikut beberapa tahapan dalam pembangunan perangkat sensos kelembapan dan suhu udara yang dibangun pada *project* ini.

3.2.1 Pembangunan Perangkat Keras

Perangkat keras yang digunakan dihubungkan pada sebuah *universal* PCB supaya perangkat dapat diganti jika diperlukan. NodeMCU ESP8266 dihubungkan ke *universal* PCB pada *slot female* yang di solder. DHT11 dihubungkan ke *universal* PCB dengan kabel *jumper*. *Buzzer* disolder ke *universal* PCB.

Universal PCB berperan sebagai *daughterboard* dan menghubungkan *output* DHT sebelah kiri ke pin D5 (GPIO14), menghubungkan *output* DHT sebelah kanan ke pin D1 (GPIO5), dan menghubungkan *input buzzer* ke D2 (GPIO4). Pin *power* DHT sebelah kiri dan *power* DHT sebelah kanan dihubungkan ke pin 3.3V terdekat dengan posisi *slot* DHT dan pin *ground* DHT sebelah kiri, *ground* DHT sebelah kanan, dan *ground buzzer* dihubungkan ke pin *ground* terdekat. Skematik perancangan perangkat keras tertera pada Gambar 3.8.



Gambar 3.8 Skematik Perancangan

3.2.2 Penggunaan Arduino

Program pada ESP8266 akan dibuat dengan pustaka PlatformIO dari Visual Studio Code untuk melakukan pembangunan dan pengunggahan program untuk ESP8266. Kode program dapat diakses melalui URL https://github.com/roberika/iot-esp.

Program melakukan beberapa pengaturan global seperti deklarasi penggunaan pustaka pada program dan konfigurasi program. Pustaka yang digunakan meliputi pustaka Arduino, pustaka-pustaka untuk koneksi *Wi-Fi* pada ESP8266, pustaka FirebaseClient, pustaka AdafruitSensor dan DHT unutk

penggunaan DHT11, pustaka JSON oleh nlohmann, dan pustaka *time* untuk pengaturan pencatatan waktu. Pengaturan jaringan *Wi-Fi* dan konfigurasi layanan *Firebase* dilakukan melalui kode program. Kode program untuk melakukan pengaturan tersebut dilampirkan pada Gambar 3.9.

```
// Impor pustaka
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <FirebaseClient.h>
#include <nlohmann/json.hpp>
#include <Adafruit_Sensor.h>
#include "DHT.h"
#include "time.h"

 // Setelan database Firebase
#define WIFI_SSID "Wifi Rumah"
#define WIFI_PASSWORD "1992kunci"
#define DHT_TYPE DHT11
#define FIREBASE_PROJECT_ID "dht-firebase-if51"
#define API_KEY "AIzaSyDwOZzT_L0wlNby5OrAscTdL8sBKDmZ1ik"
#define THRESHOLD_COLLECTION_ID "thresholds/threshold"
#define THRESHOLD_INTERVAL 5000
#define RECORDS_COLLECTION_ID "records"
#define RECORDS_INTERVAL 10000
#define USER_EMAIL "dht-firebase@if51.mdp.ac.id"
#define USER_PASSWORD "janganlupo"
#define DATABASE_URL "https://dht-firebase-if51-default-rtdb.asia-
southeast1.firebasedatabase.app/"
#define DATABASE_SECRET "seXLmFU5gfHCwQ8SE4Y9yK1iDxLv1CAMJZtAkTed"
#define MONITORING_INTERVAL 1000
#define BUZZER 4
#define DHT_LEFT 14
#define DHT_RIGHT 5

// Konfigurasi perangkat
DHT dhtLeft(DHT_LEFT, DHT_TYPE);
DHT dhtRight(DHT_RIGHT, DHT_TYPE);
DefaultNetwork network;
using json = nlohmann::json;

// Konfigurasi Firebase
FirebaseApp appFirestore, appRealtime;
UserAuth userAuth(API_KEY, USER_EMAIL, USER_PASSWORD, 3000);
LegacyToken legacyToken(DATABASE_SECRET);
RealtimeDatabase database;
AsyncResult resultFirestore, resultRealtime;
Firestore::Documents docs;
WiFiClientSecure sslFirestore, sslRealtime;
using AsyncClient = AsyncClientClass;
AsyncClient clientFirestore(sslFirestore, getNetwork(network)),
clientRealtime(sslRealtime, getNetwork(network));
```

Gambar 3.9 Kode Program Konfigurasi Perangkat

Program menggunakan beberapa fungsi untuk mengolah data, seperti fungsi untuk menangani otentikasi, fungsi untuk mengambil *timestamp string* dari waktu pencatatan, fungsi untuk mengambil waktu saat ini dalam detik, fungsi untuk mencetak *payload Firebase* untuk *debugging*, dan fungsi untuk mencetak hasil eksekusi fungsi *Firebase* untuk *debugging*. Program menggunakan beberapa variabel untuk operasinya, yaitu variabel untuk menyimpan hasil rekaman sensor secara sementara, variabel untuk menyimpan *threshold* yang dimuat dari *Firebase Cloud Firestore* untuk perbandingan dan aktivasi *buzzer*, dan variabel untuk menyimpan jeda waktu dari penyimpanan rekaman terakhir. Fungsi dan variabel tersebut dideklarasikan pada Gambar 3.10.

```
void authHandler(FirebaseApp appFirestore);
String getTimestampString(uint64_t sec);
unsigned long getTime();
void printPayload(AsyncClient client, String payload);
void printResult(AsyncResult &aResult);
double thresholdDHTLeftTemperature;
double thresholdDHTLeftHumidity;
double recordedDHTLeftTemperature;
double recordedDHTLeftHumidity;
double thresholdDHTRightTemperature;
double thresholdDHTRightHumidity;
double recordedDHTRightTemperature;
double recordedDHTRightHumidity;
unsigned long lastThresholdUpdate = 0;
unsigned long lastFirestoreUpdate = 0;
unsigned long lastRealtimeUpdate = 0;
```

Gambar 3.10 Kode Program Deklarasi Fungsi dan Variabel

Program perlu menjalankan serangkaian perintah sebelum memulai merekam sensor. Kode program ini tertulis dalam fungsi *setup()* sehingga dilaksanakan sebelum *loop* awal dimulai dan meliputi fungsi untuk menyetel ESP8266, menghubungkan ke jaringan, menghubungkan ke server NTP,

menghubungkan ke *Cloud Firestore*, dan menghubungkan ke *Realtime Database*. Kode program untuk setelan awal perangkat dapat dibaca pada Gambar 3.11.

```
void setup() {
  // Setup perangkat
  Serial.begin(115200);
  Serial.println(" ");
  pinMode(BUZZER, OUTPUT);
  dhtLeft.begin();
  dhtRight.begin();

  // Konek ke internet
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to ");
  Serial.print(WIFI_SSID);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("Connected to ");
  Serial.println(WiFi.localIP());

  // Konek ke server NTP
  //.. UTC+7 & no daylight savings
  configTime(0, 0, "id.pool.ntp.org");

  // Konek ke Firebase
  Serial.print("Connecting to ");
  Serial.println(FIREBASE_PROJECT_ID);

  //.. Konek ke cloud firestore
  sslFirestore.setInsecure();
  sslFirestore.setBufferSizes(1024, 1024);
  initializeApp(clientFirestore, appFirestore,
    getAuth(userAuth), resultFirestore);
  authHandler(appFirestore);
  Serial.println("Authentication Information");
  Firebase.printf("User UID: %s\n", appFirestore.getUid().c_str());
  Firebase.printf("Auth Token: %s\n", appFirestore.getToken().c_str());
  Firebase.printf("Refresh Token: %s\n",
    appFirestore.getRefreshToken().c_str());
  appFirestore.getApp<Firestore::Documents>(docs);
  clientFirestore.setAsyncResult(resultFirestore);

  //.. Konek ke stream realtime database
  sslRealtime.setInsecure();
  sslRealtime.setBufferSizes(1024, 1024);
  initializeApp(clientRealtime, appRealtime, getAuth(legacyToken));
  appRealtime.getApp<RealtimeDatabase>(database);
  database.url(DATABASE_URL);
  clientRealtime.setAsyncResult(resultRealtime);

  Serial.print("Connected to ");
  Serial.println(FIREBASE_PROJECT_ID);
}
```

Gambar 3.11 Kode Program Setelah Perangkat

Program berjalan mengikuti fungsi *loop()* dari Arduino. Sebelum melakukan fungsi lain pada *loop* program pertama memastikan apakah klien *Firebase* sudah terhubung, melakukan otentikasi dengan *Cloud Firestore*, dan membiarkan fungsi *looping* dari *Realtime Database* dan *Cloud Firestore* telah berjalan terlebih dahulu. Kode program awal *loop* tercantumkan pada Gambar 3.12.

```
void loop() {
  if (!appFirestore.ready() || !appRealtime.ready()) {
    return;
  }

  authHandler(appFirestore);
  database.loop();
  docs.loop();
  //...
}
```

Gambar 3.12 Kode Program Awal *Loop*

Program memuat *threshold* dari *Cloud Firestore* setiap 5 detik untuk memastikan *threshold* sesuai dengan setelan dari pengguna namun tidak yang dimuat mencakup *threshold* untuk kelembaban dan suhu pada sensor DHT sebelah kiri dan kelembaban dan suhu pada sensor DHT sebelah kanan. Kode program untuk memuat *threshold* dicantumkan pada Gambar 3.13.

```
void loop() {
  //...

  // Ambil nilai threshold yang disimpan pada Firebase
  if (millis() - lastThresholdUpdate > THRESHOLD_INTERVAL) {
    lastThresholdUpdate = millis();

    //.. Ambil Document dht1 dan dht2 pada Collection
    //.. threshold yang berisi threshold peringatan buzzer
    Serial.println("Fetching thresholds...");
    String payload = docs.get(clientFirestore,
      Firestore::Parent(FIREBASE_PROJECT_ID), THRESHOLD_COLLECTION_ID,
      GetDocumentOptions(DocumentMask()));

    //.. Muat threshold
    json decoded = json::parse(payload);
    json threshold = decoded.at("fields");
    thresholdDHTLeftTemperature = threshold.at("leftTemperature")
      .at("doubleValue").template get<double>();
    thresholdDHTLeftHumidity = threshold.at("leftHumidity")
      .at("doubleValue").template get<double>();
    thresholdDHTRightTemperature = threshold.at("rightTemperature")
      .at("doubleValue").template get<double>();
    thresholdDHTRightHumidity = threshold.at("rightHumidity")
      .at("doubleValue").template get<double>();
    Serial.println(thresholdDHTLeftTemperature);
    Serial.println(thresholdDHTLeftHumidity);
    Serial.println(thresholdDHTRightTemperature);
    Serial.println(thresholdDHTRightHumidity);
    Serial.println(" ");
  }
  //...
}
```

Gambar 3.13 Kode Program Memuat *Threshold*

Program mengukur kelemababan dan suhu melalui sensor dan menyimpannya ke *Realtime Database* setiap 1 detik. Pada bagian ini program juga membandingkan rekaman dengan nilai *threshold* untuk mengecek apakah salah satu sisi melewati *threshold* dan apakah *buzzer* perlu dibunyikan. Kode program pengukuran dan pembandingan ini dilampirkan pada Gambar 3.14.

```
void loop() {
  //...

  if (millis() - lastRealtimeUpdate > MONITORING_INTERVAL) {
    lastRealtimeUpdate = millis();
    // Ambil data suhu dan kelembaban dari dht
    Serial.println("Recording room condition...");
    recordedDHTLeftTemperature = dhtLeft.readTemperature(false);
    recordedDHTLeftHumidity = dhtLeft.readHumidity();
    recordedDHTRightTemperature = dhtRight.readTemperature(false);
    recordedDHTRightHumidity = dhtRight.readHumidity();
    Serial.println(recordedDHTLeftTemperature);
    Serial.println(recordedDHTLeftHumidity);
    Serial.println(recordedDHTRightTemperature);
    Serial.println(recordedDHTRightHumidity);
    Serial.println(" ");

    // Cek apakah suhu dan kelembaban melebihi nilai threshold
    bool leftDanger = recordedDHTLeftHumidity >= thresholdDHTLeftHumidity
      && recordedDHTLeftTemperature >= thresholdDHTLeftTemperature;
    bool rightDanger =  recordedDHTRightHumidity >= thresholdDHTRightHumidity
      && recordedDHTRightTemperature >= thresholdDHTRightTemperature;

    if(!leftDanger && !rightDanger) {
      digitalWrite(BUZZER, LOW);
      Serial.println("calm...");
    } else {
      if(leftDanger) {
        digitalWrite(BUZZER, HIGH);
        Serial.println("LEFT, DANGER!!!");
      }
      if(rightDanger) {
        digitalWrite(BUZZER, HIGH);
        Serial.println("RIGHT, DANGER!!!");
      }
    }
    Serial.println(" ");

    // Simpan data pengukuran ke Realtime Database
    Serial.println("Updating monitor values...");
    Serial.println("Left Humidity...");
    database.set<double>(clientRealtime, "/leftHumidity",
      recordedDHTLeftHumidity);
    Serial.println("Left Temperature...");
    database.set<double>(clientRealtime, "/leftTemperature",
      recordedDHTLeftTemperature);
    Serial.println("Right Humidity...");
    database.set<double>(clientRealtime, "/rightHumidity",
      recordedDHTRightHumidity);
    Serial.println("Right Temperature...");
    database.set<double>(clientRealtime, "/rightTemperature",
      recordedDHTRightTemperature);
    Serial.println(" ");
  }
  //...
}
```

Gambar 3.14 Kode Program Mengecek dan Menyimpan Rekaman

Program menyimpan catatan rekaman ke *Cloud Firestore* setiap 10 detik. Catatan memuat sisi DHT mana yang digunakan untuk rekaman, tingkat kelembaban dan suhu yang direkam, dan waktu pengambilan rekaman. Kode program untuk menyimpan catatan dilampirkan pada Gambar 3.15.

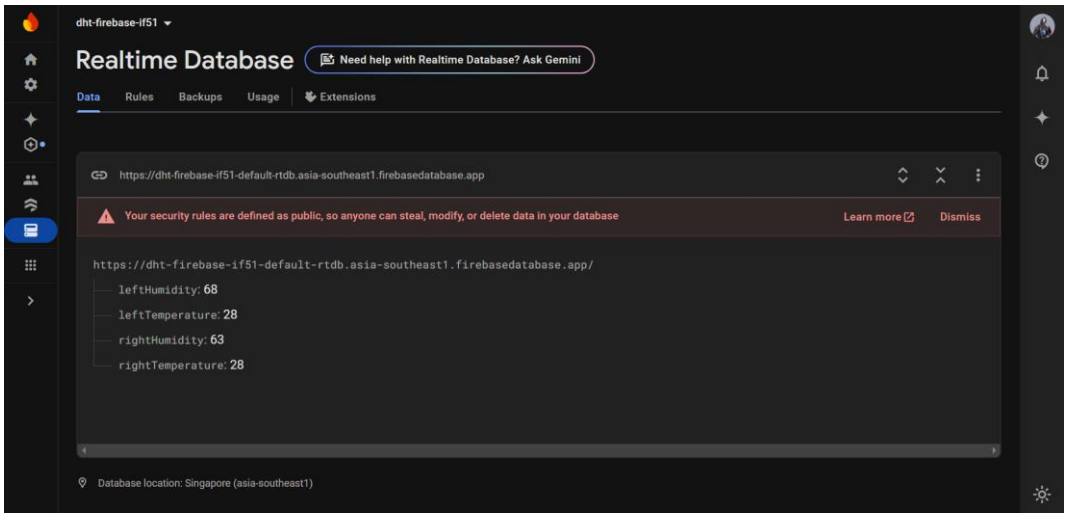
```
void loop() {
  //...

  // Simpan rekaman catatan ke Cloud Firestore
  if (millis() - lastFirestoreUpdate > RECORDS_INTERVAL) {
    lastFirestoreUpdate = millis();
    for(int i = 0; i < 2; i++) {
      //.. Susun jadi 1 dokumen
      Serial.println("Saving recorded conditions...");
      Serial.println("Timestamp...");
      Values::TimestampValue recordTime(getTimestampString(getTime()));
      Serial.println("Temperature...");
      Values::DoubleValue recordTemperature((i == 0) ?
        recordedDHTLeftTemperature : recordedDHTRightTemperature);
      Serial.println("Humidity...");
      Values::DoubleValue recordHumidity((i == 0) ?
        recordedDHTLeftHumidity : recordedDHTRightHumidity);
      Serial.println("Type...");
      Values::IntegerValue recordDHTID(i);

      Document<Values::Value> doc("dhtid", Values::Value(recordDHTID));
      doc.add("timestamp", Values::Value(recordTime));
      doc.add("temperature", Values::Value(recordTemperature));
      doc.add("humidity", Values::Value(recordHumidity));

      //.. Buat ID baru untuk dokumen, jadikan payload, dan kirim
      String documentPath = RECORDS_COLLECTION_ID;
      Serial.println("Saving...");
      String payload = docs.createDocument(clientFirestore,
        Firestore::Parent(FIREBASE_PROJECT_ID),
        documentPath, DocumentMask(), doc);

      //.. Tampilkan payload atau error
      // printPayload(clientFirestore, payload);
      Serial.println(" ");
    }
  }
}
```

Gambar 3.15 Kode Program Menyimpan Catatan Rekaman

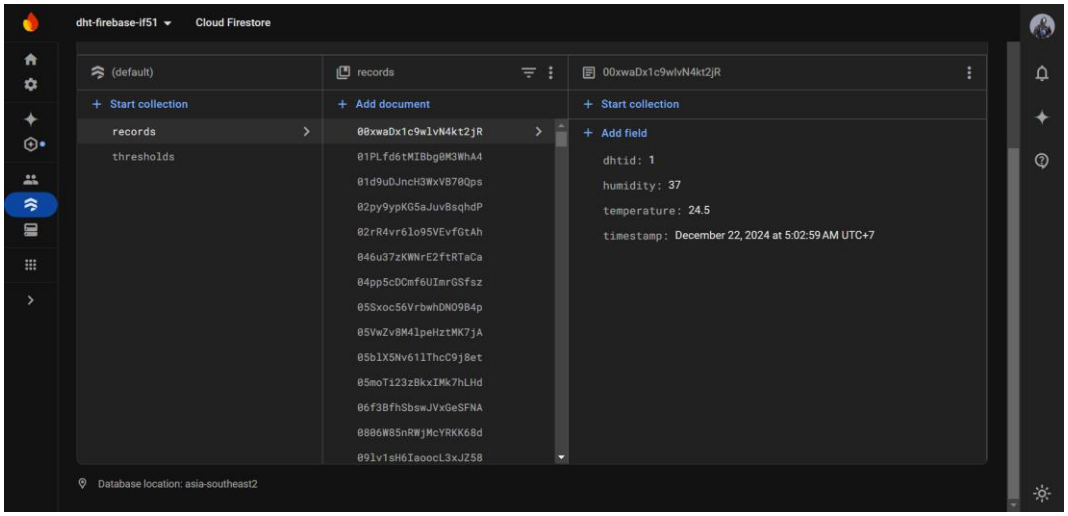
3.2.3 Penggunaan Firebase

Nilai sensor yang di dapat dari DHT11 akan diproses dan disimpan pada *Firebase*. Data hasil rekaman sensor akan diperbaharui setiap 1 detik dan disimpan pada *Firebase Realtime Database* untuk mengakses sensor secara *realtime*/*live*. Data catatan rekaman sensor akan disimpan setiap 10 detik pada *Firebase Cloud Firestore* untuk mengakses catatan rekaman sensor saat diperlukan. *Threshold* yang digunakan untuk membunyikan *buzzer* juga disimpan pada *Cloud Firestore* dan dimuat ulang pada ESP8266 setiap 5 detik.

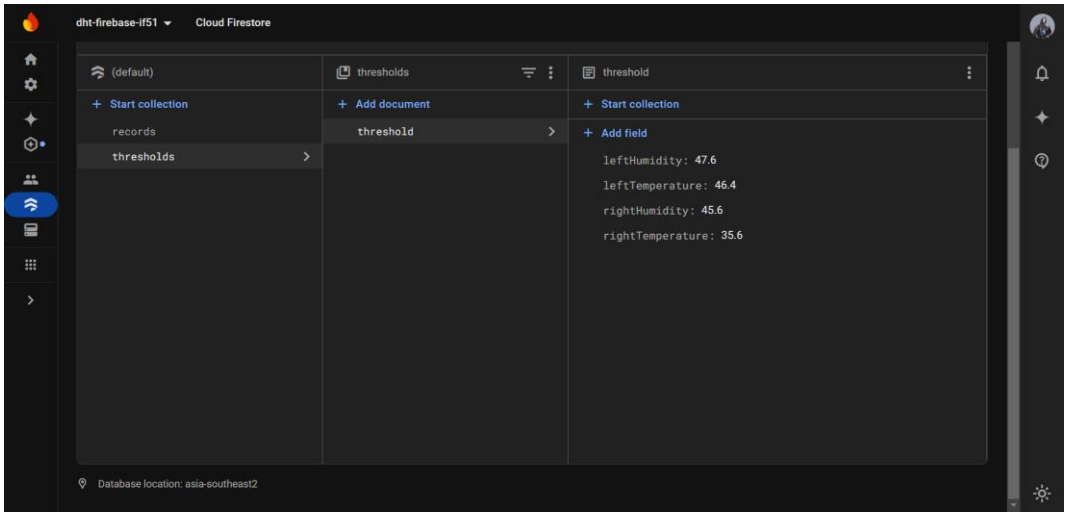
Contoh *live* rekaman sensor pada *Firebase Realtime Database* dapat dilihat pada Gambar 3.16. Contoh catatan rekaman sensor pada *Firebase Cloud Firestore* dapat dilihat pada Gambar 3.17. Contoh *threshold* pada *Firebase Cloud Firestore* dapat dilihat pada Gambar 3.18.



Gambar 3.16 *Live* Rekaman Sensor pada *Realtime Database*



Gambar 3.17 Catatan Rekaman Sensor pada *Cloud Firestore*



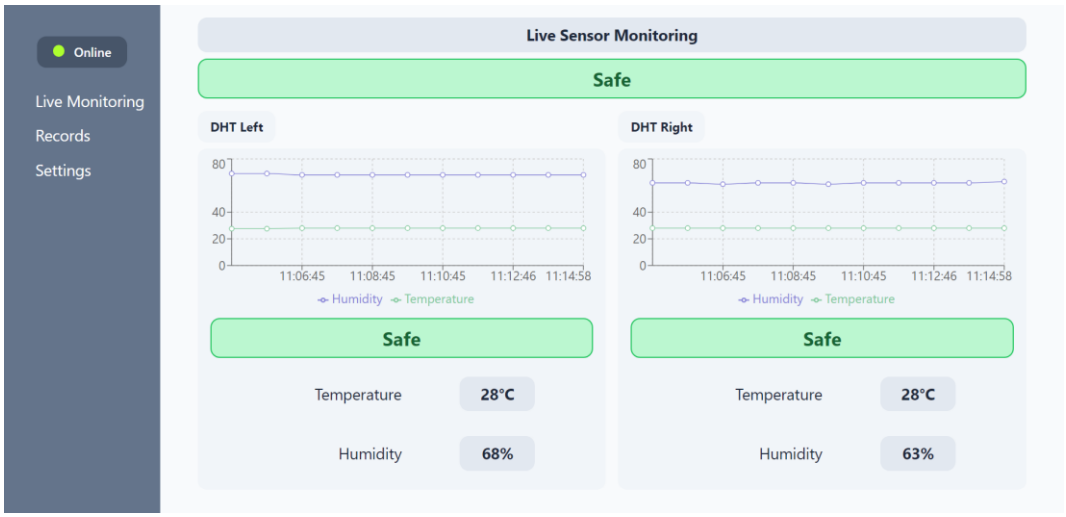
Gambar 3.18 *Threshold* pada *Cloud Firestore*

3.2.4 Pembangunan Aplikasi Berbasis Web

Aplikasi berbasis web dapat digunakan untuk melihat hasil pengukuran dari sensor DHT, melihat catatan pengukuran lalu sensor, dan menyetel *threshold* yang digunakan sensor untuk mengetahui kapan untuk berbunyi. Aplikasi berbasis web bekerja dengan mengolah dan menampilkan data yang tersimpan pada database *Firebase* saat pengguna mengakses aplikasi pada Github Pages lewat URL

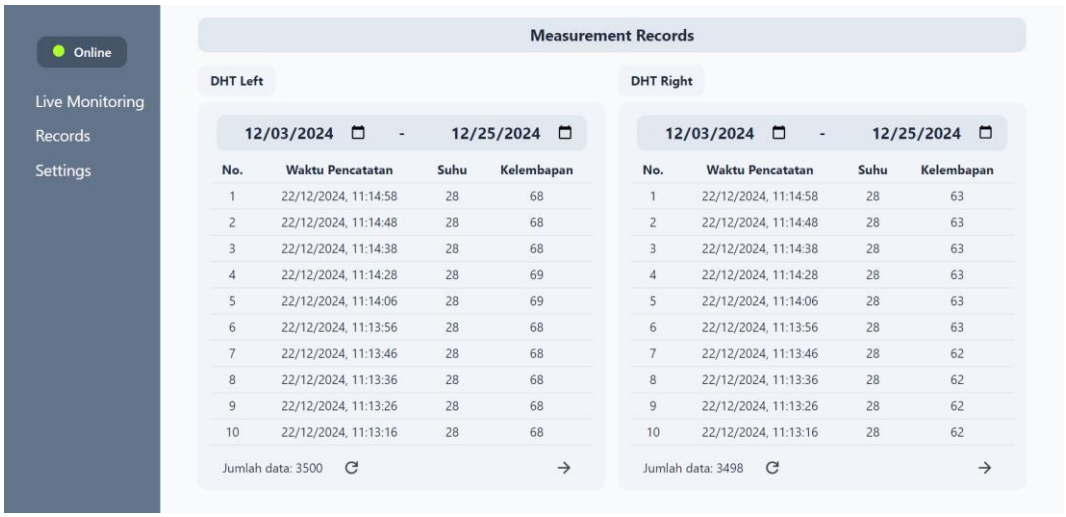
https://roberika.github.io/iot-gui/. Aplikasi berbasis web dibangun dengan *framework* JavaScript, ReactJS, *framework* CSS, Tailwind, dan *build tool* Vite.

Halaman *Live Monitoring* menampilkan apakah kelembaban dan suhu pada sensor melewati *threshold*, nilai *realtime* dari kelembaban dan suhu, dan grafik berisi 10 catatan terakhir dalam jeda 1 menit untuk setiap sensor DHT11. Tampilan halaman *Live Monitoring* dapat dilihat pada Gambar 3.19.



Gambar 3.19 Halaman *Live Monitoring*

Halaman *Records* berisi catatan pengukuran kelembaban dan suhu ditampilkan dalam sebuah tabel yang dapat disetel untuk memuat rekaman dari dan sampai tanggal berapa. Halaman *Records* dapat dilihat pada Gambar 3.20.



Gambar 3.20 Halaman *Records*

Halaman *Settings* dapat digunakan untuk melihat dan menyetel *threshold* yang digunakan oleh perangkat sebagai indikator untuk membunyikan *buzzer*. Tampilan halaman *Settings* digambarkan pada Gambar 3.21.

Gambar 3.21 Halaman *Settings*

Aplikasi berbasis web merupakan aplikasi *Single Page Application* mengikuti batasan dari Github untuk meng-*host* aplikasi pada Github Pages. Aplikasi menggunakan pustaka *Firebase* untuk mengambil dan mengolah data yang tersimpan pada *Cloud Firestore* dan *Realtime Database*. Kode program untuk aplikasi berbasis web dapat diakses melalui repositori Github pada URL https://github.com/roberika/iot-gui.

Halaman *Live Monitoring* mengambil data 60 catatan kelembaban dan suhu terakhir dari *Cloud Firestore* untuk menyusun grafik pengukuran selama 10 menit terakhir dan mengambil data pengukuran *live* dari *Realtime Database* untuk menampilkan status, kelembaban, dan suhu dari masing-masing sensor DHT11. Kode program yang digunakan untuk melakukan fungsi ini dilampirkan pada Gambar 3.22 dan Gambar 3.23.

```
import { useEffect, useState } from 'react'

import { firestore, realtime } from './Firebase';
import { doc, getDoc } from 'firebase/firestore';
import { onValue, ref } from 'firebase/database';

import './App.css'

import LiveChart from './LiveChart';

const Units = {
    TEMPERATURE: "°C",
    HUMIDITY: "%",
}

const DHTID_LEFT = 0;
const DHTID_RIGHT = 1;

function LiveMonitoring() {
    const [leftDanger, setLeftDanger] = useState(false);
    const [rightDanger, setRightDanger] = useState(false);
    const [leftTemperature, setLeftemperature] = useState(0);
    const [leftHumidity, setLeftHumidity] = useState(0);
    const [rightTemperature, setRightTemperature] = useState(0);
    const [rightHumidity, setRightHumidity] = useState(0);

    const setDanger = async (
        leftTemperature,
```

```
        leftHumidity,
        rightTemperature,
        rightHumidity
    ) => {
        const thresholdRef = doc(firestore, "thresholds", "threshold");
        const thresholdSnapshot = await getDoc(thresholdRef);
        const threshold = thresholdSnapshot.data();

        setLeftDanger(
            (threshold.leftTemperature <= leftTemperature) &&
            (threshold.leftHumidity <= leftHumidity)
        );
        setRightDanger(
            (threshold.rightTemperature <= rightTemperature) &&
            (threshold.rightHumidity <= rightHumidity)
        );
    }

    const getDangerSide = () => {
        if (leftDanger && rightDanger) {
            return "Both"
        }
        if (leftDanger) {
            return "Left"
        }
        if (rightDanger) {
            return "Right"
        }
        return "Safe"
    }

    useEffect(() => {
        const liveRef = ref(realtime);
        onValue(liveRef, (snapshot) => {
            const newVal = snapshot.val()
            const leftTemperature = newVal.leftTemperature;
            const leftHumidity = newVal.leftHumidity;
            const rightTemperature = newVal.rightTemperature;
            const rightHumidity = newVal.rightHumidity;
            setLeftemperature(leftTemperature);
            setLeftHumidity(leftHumidity);
            setRightTemperature(rightTemperature);
            setRightHumidity(rightHumidity);
            setDanger(
                leftTemperature, leftHumidity,
                rightTemperature, rightHumidity
            );
        });
    }, [])

    return (
        <div className='live-monitoring'>
            <p className='content-title'>
                Live Sensor Monitoring
            </p>
            <div className={'card card-background live-measure-status ' +
                ((leftDanger || rightDanger)
                    ? "live-measure-danger" : "live-measure-safe")}>
                {getDangerSide()}
```

```
            </div>
            <div className='lg:content'>
                <div className='grow'>
                    <div className='card-unimportant card-title'>
                        <p>DHT Left</p>
                    </div>
                    <div className='card card-background'>
                        <div className='live-measure-row'>
                            <LiveChart dhtid={DHTID_LEFT} />
                            <LiveMonitor value={leftDanger} />
                            <LiveMeasure unit={Units.TEMPERATURE}
                                value={leftTemperature} />
                            <LiveMeasure unit={Units.HUMIDITY}
                                value={leftHumidity} />
                        </div>
                    </div>
                </div>
                <div className='grow'>
                    <div className='card-unimportant card-title'>
                        <p>DHT Right</p>
                    </div>
                    <div className='card card-background'>
                        <div className='live-measure-row'>
                            <LiveChart dhtid={DHTID_RIGHT} />
                            <LiveMonitor value={rightDanger} />
                            <LiveMeasure unit={Units.TEMPERATURE}
                                value={rightTemperature} />
                            <LiveMeasure unit={Units.HUMIDITY}
                                value={rightHumidity} />
                        </div>
                    </div>
                </div>
            </div>
        </div>
    )
}

function LiveMonitor({ value }) {
    return (
        <p className={'live-measure-status ' +
            (value ? "live-measure-danger" : "live-measure-safe")}>
            {value ? "Danger" : "Safe"}
        </p>
    )
}

function LiveMeasure({ unit, value }) {
    return (
        <div className='sm:live-measure'>
            <p className='ml-auto my-auto sm:text-xl'>
                {unit == Units.TEMPERATURE
                    ? "Temperature" : "Humidity"}
            </p>
            <p className='live-measure-units min-w-24'>
                {value + unit}
            </p>
        </div>
    )
}
```

```
export default LiveMonitoring
```

Gambar 3.22 Kode Program Halaman *Live Monitoring*

```
import { useEffect, useState } from 'react'

import { firestore } from './Firebase';
import {
    collection, getDocs, limit,
    orderBy, query, where
} from 'firebase/firestore';
import {
    CartesianGrid, Legend, Line, LineChart,
    ResponsiveContainer, Tooltip, XAxis, YAxis
} from 'recharts';

import './App.css'

const Units = {
    TEMPERATURE: "°C",
    HUMIDITY: "%",
}
const RECORDS_PER_PAGE = 61;

function LiveChart({ dhtid }) {

    const getDate = (timestamp) => {
        return timestamp.toDate().toLocaleDateString("en-GB", {
            day: "2-digit",
            month: "2-digit",
            year: "numeric",
        });
    }

    const getTime = (timestamp) => {
        return timestamp.toDate().toLocaleTimeString("en-GB", {
            hour: "2-digit",
            minute: "2-digit",
            second: "2-digit",
        });
    }

    const [records, setRecords] = useState(null);

    const loadData = async () => {
        const querySnapshot = await getDocs(query(
            collection(firestore, "records"),
            where("dhtid", "==", dhtid),
            orderBy("timestamp", "desc"),
            limit(RECORDS_PER_PAGE)
        ));
        setRecords(querySnapshot);
    }

    const getData = () => {
        const data = !records ? [] : records.docs.map((doc, index) => {
```

```
            if (index % 6 == 0) return {
                "dht": dhtid == 0 ? "Left" : "Right",
                "humidity": doc.data().humidity,
                "temperature": doc.data().temperature,
                "date": getDate(doc.data().timestamp),
                "time": getTime(doc.data().timestamp),
            }
            return null;
        }).filter((e) => e != null);
        return data.reverse();
    }

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

    return (
        <ResponsiveContainer width="99%" height={200} className="mx-auto">
            <LineChart data={getData()} syncId={"dht-records"}
                margin={{ top: 5, right: 10, left: 5, bottom: 5 }}>
                <CartesianGrid strokeDasharray="3 3" />
                <XAxis dataKey="time" />
                <YAxis width={20} />
                <Tooltip />
                <Legend />
                <Line name="DHT" type="monotone" dataKey="dht"
                    stroke="#724892" legendType='none' hide='true' />
                <Line name="Date" type="monotone" dataKey="date"
                    stroke="#d66884" legendType='none' hide='true' />
                <Line name="Humidity" type="monotone"
                    dataKey="humidity" stroke="#8884d8"
                    unit={Units.HUMIDITY} />
                <Line name="Temperature" type="monotone"
                    dataKey="temperature" stroke="#82ca9d"
                    unit={Units.TEMPERATURE} />
            </LineChart>
        </ResponsiveContainer>
    )
}

export default LiveChart
```

Gambar 3.23 Kode Program Grafik pada *Live Monitoring*

Halaman *Records* mengambil data 10 catatan kelembaban dan suhu terakhir dari *Cloud Firestore* untuk menyusun tabel catatan pengukuran. Tabel juga dapat digunakan untuk menyeleksi jangkauan waktu catatan yang akan dimuat pada tabel dan dapat memilih untuk menampilkan 10 data catatan selanjutnya dengan prinsip *pagination* untuk tidak memberatkan *database*. Kode program yang

digunakan untuk membangun fitur ini dilampirkan pada Gambar 3.24 dan Gambar 3.25.

```
import './App.css'

import RecordsTable from './RecordsTable';

const DHTID_LEFT = 0;
const DHTID_RIGHT = 1;

function Records() {
    return (
        <div className='records'>
            <p className='content-title'>
                Measurement Records
            </p>
            <div className='lg:content'>
                <div className='grow'>
                    <div className='card-unimportant card-title'>
                        <p>DHT Left</p>
                    </div>
                    <div className='card card-background'>
                        <RecordsTable dhtid={DHTID_LEFT} />
                    </div>
                </div>
                <div className='grow'>
                    <div className='card-unimportant card-title'>
                        <p>DHT Right</p>
                    </div>
                    <div className='card card-background'>
                        <RecordsTable dhtid={DHTID_RIGHT} />
                    </div>
                </div>
            </div>
        </div>
    )
}

export default Records
```

Gambar 3.24 Kode Program Halaman *Records*

```
import { useEffect, useState } from 'react'

import { firestore } from './Firebase';
import {
    collection, query, where, getDocs,
    orderBy, getCountFromServer, limit,
    startAfter, endBefore, limitToLast
} from "firebase/firestore";

import './App.css'

import RefreshIcon from './assets/refresh-icon.svg'
import PrevIcon from './assets/prev-icon.svg'
import NextIcon from './assets/next-icon.svg'
```

```
const RECORDS_PER_PAGE = 10;

function RecordsTable({ dhtid }) {

    const getToday = () => {
        const today = new Date();
        return today.toISOString().slice(0, 10);
    }

    const getYesterday = () => {
        const yesterday = new Date();
        yesterday.setDate(yesterday.getDate() - 1);
        return yesterday.toISOString().slice(0, 10);
    }

    const [records, setRecords] = useState(null);
    const [recordsCount, setRecordsCount] = useState(0);
    const [page, setPage] = useState(0);
    const [startDate, setStartDate] = useState(getYesterday())
    const [endDate, setEndDate] = useState(getToday())

    const loadData = async (cursor = null, forward = true, page = 0) => {
        const countSnapshot = await getCountFromServer(query(
            collection(firestore, "records"),
            where("dhtid", "==", dhtid),
            where("timestamp", ">=", getSeconds(startDate, false)),
            where("timestamp", "<=", getSeconds(endDate, true)),
        ));
        setRecordsCount(countSnapshot.data().count);

        let querySnapshot;
        if (page == 0) {
            querySnapshot = await getDocs(query(
                collection(firestore, "records"),
                where("dhtid", "==", dhtid),
                where("timestamp", ">=", getSeconds(startDate, false)),
                where("timestamp", "<=", getSeconds(endDate, true)),
                orderBy("timestamp", "desc"),
                limit(RECORDS_PER_PAGE)
            ));
        } else if (forward) {
            querySnapshot = await getDocs(query(
                collection(firestore, "records"),
                where("dhtid", "==", dhtid),
                where("timestamp", ">=", getSeconds(startDate, false)),
                where("timestamp", "<=", getSeconds(endDate, true)),
                orderBy("timestamp", "desc"),
                startAfter(cursor),
                limit(RECORDS_PER_PAGE)
            ));
        } else {
            querySnapshot = await getDocs(query(
                collection(firestore, "records"),
                where("dhtid", "==", dhtid),
                where("timestamp", ">=", getSeconds(startDate, false)),
                where("timestamp", "<=", getSeconds(endDate, true)),
                orderBy("timestamp", "desc"),
                endBefore(cursor),
                limitToLast(RECORDS_PER_PAGE)
```

```
            ));
        }
        setRecords(querySnapshot);
    }

    const nextPage = () => {
        loadData(records.docs[records.docs.length - 1], true, page + 1);
        setPage(page + 1);
    }

    const prevPage = () => {
        loadData(records.docs[0], false, page - 1);
        setPage(page - 1);
    }

    const getRecords = () => {
        return !records ? [] : records.docs.map((doc, _) => doc.data())
    }

    const getDate = (timestamp) => {
        return timestamp.toDate().toLocaleDateString("en-GB", {
            day: "2-digit",
            month: "2-digit",
            year: "numeric",
            hour: "2-digit",
            minute: "2-digit",
            second: "2-digit",
        });
    }

    const getSeconds = (dateString, end = false) => {
        const millis = Math.floor(Date.parse(dateString)) +
            (end ? 86399999 : 0);
        return new Date(millis);
    }

    useEffect(() => {
        loadData();
    }, [startDate, endDate]);

    return (
        <div className='records-table'>
            <div className='card card-emphasis bg-slate-200'>
                <div className='flex flex-col sm:flex-row'>
                    <input value={startDate}
                        className='records-table-date' type='date'
                        onChange={(e) => { setStartDate(e.target.value) }}>
                    </input>
                    <p className='grow hidden sm:block'>-</p>
                    <hr className='bg-slate-300 h-0.5' />
                    <input value={endDate}
                        className='records-table-date' type='date'
                        onChange={(e) => { setEndDate(e.target.value) }}>
                    </input>
                </div>
            </div>
            <table>
                <thead>
                    <tr>
                        <th className='hidden sm:table-cell'>No.</th>
```

```
                        <th>Waktu Pencatatan</th>
                        <th>Suhu</th>
                        <th>Kelembapan</th>
                    </tr>
                </thead>
                <tbody>
                    {getRecords().map((record, index) => {
                        return <tr key={index}>
                            <td className='hidden sm:table-cell'>
                                {index + (page * RECORDS_PER_PAGE) + 1}
                            </td>
                            <td>{getDate(record.timestamp)}</td>
                            <td>{record.temperature}</td>
                            <td>{record.humidity}</td>
                        </tr>
                    })}
                </tbody>
            </table>
            <div className='flex px-4 pt-2'>
                <div className='flex-shrink pr-4 py-2 hidden sm:block'>
                    {`Jumlah data: ${recordsCount}`}
                </div>
                <button className='records-table-button' type='button'
                    onClick={() => loadData(null, false)}>
                    <img src={RefreshIcon} />
                </button>
                <div className='flex-grow' />
                {page > 0 ? (
                    <button className='records-table-button'
                        type='button' onClick={prevPage}>
                        <img src={PrevIcon} />
                    </button>
                ) : (
                    <button className='records-table-button'
                        type='button' disabled>
                        <img className='invisible' src={PrevIcon} />
                    </button>
                )}

                {(page + 1) * RECORDS_PER_PAGE < recordsCount ? (
                    <button className='records-table-button'
                        type='button' onClick={nextPage}>
                        <img src={NextIcon} />
                    </button>
                ) : (
                    <button className='records-table-button'
                        type='button' disabled>
                        <img className='invisible' src={NextIcon} />
                    </button>
                )}
            </div>
        </div>
    )
}

export default RecordsTable
```

Gambar 3.25 Kode Program Tabel pada *Records*

Halaman *Settings* dapat digunakan untuk menyetel *threshold* kelembaban dan suhu untuk masing-masing sensor DHT11. Kode program pada penyetelan *threshold* dapat dilihat pada Gambar 3.26.

```
import { useEffect, useState } from 'react'

import { firestore } from './Firebase';
import { doc, getDoc, updateDoc } from "firebase/firestore";

import './App.css'

const Units = {
    TEMPERATURE: "°C",
    HUMIDITY: "%",
}

const Sensors = {
    LEFT: "0",
    RIGHT: "1",
}

function Settings() {
    const [threshold, setThreshold] = useState({
        leftTemperature: '0',
        leftHumidity: '0',
        rightTemperature: '0',
        rightHumidity: '0',
    });

    const getThresholds = async () => {
        const thresholdRef = doc(firestore, "thresholds", "threshold");
        const thresholdSnapshot = await getDoc(thresholdRef);
        setThreshold(thresholdSnapshot.data());
    }

    const handleChange = (e) => {
        setThreshold(prev => ({ ...prev, [e.target.name]: e.target.value }));
    }

    const handleSubmit = async (e) => {
        e.preventDefault(); //Don't refresh the page
        const thresholdRef = doc(firestore, "thresholds", "threshold");
        await updateDoc(thresholdRef, {
            leftTemperature: parseFloat(threshold.leftTemperature),
            leftHumidity: parseFloat(threshold.leftHumidity),
            rightTemperature: parseFloat(threshold.rightTemperature),
            rightHumidity: parseFloat(threshold.rightHumidity),
        })
    }

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

    return (
        <div className='settings'>
```

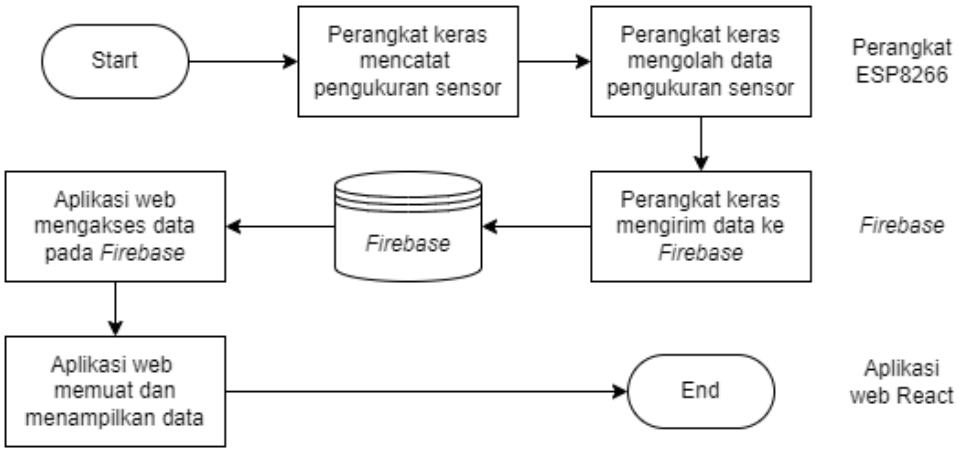
```
            <p className='content-title'>
                Sensor Settings
            </p>
            <div className='lg:content'>
                <form className='grow lg:grow-0 lg:mx-auto'
                    onSubmit={handleSubmit}>
                    <div className='card-unimportant card-title'>
                        <p>Threshold</p>
                    </div>
                    <div className='card card-background'>
                        <SettingsThreshold
                            sensor={Sensors.LEFT}
                            unit={Units.TEMPERATURE}
                            value={threshold.leftTemperature}
                            onChange={handleChange}
                        />
                        <SettingsThreshold
                            sensor={Sensors.LEFT}
                            unit={Units.HUMIDITY}
                            value={threshold.leftHumidity}
                            onChange={handleChange}
                        />
                        <SettingsThreshold
                            sensor={Sensors.RIGHT}
                            unit={Units.TEMPERATURE}
                            value={threshold.rightTemperature}
                            onChange={handleChange}
                        />
                        <SettingsThreshold
                            sensor={Sensors.RIGHT}
                            unit={Units.HUMIDITY}
                            value={threshold.rightHumidity}
                            onChange={handleChange}
                        />
                        <div className='w-full'>
                            <button className='card
                            card-emphasis block ml-auto
                            bg-yellow-200 focus:bg-green-200'>
                                Save
                            </button>
                        </div>
                    </div>
                </form>
            </div>
        </div>
    )
}

function SettingsThreshold({ sensor, unit, value, onChange }) {
    const getName = () => {
        return (
            (sensor == Sensors.LEFT ? "left" : "right") +
            (unit == Units.TEMPERATURE ? "Temperature" : "Humidity")
        );
    }

    const getLabel = () => {
        return (
            (sensor == Sensors.LEFT ? "Left" : "Right") + " " +
```

```
            (unit == Units.TEMPERATURE ? "Temperature" : "Humidity")
        )
    }

    return (
        <div className='sm:threshold'>
            <label htmlFor={getName()}
                className='mr-auto my-auto sm:text-xl smcol-span-1'>
                {getLabel()}
            </label>
            <div className='flex sm:col-span-2 my-auto'>
                <input type='number' step="0.1"
                    id={getName()} name={getName()} value={value}
                    className='threshold-units grow w-24 sm:text-right'
                    onChange={onChange} size={0} required />
                <p className='threshold-units'>
                    {unit}
                </p>
            </div>
        </div>
    )
}

export default Settings
```

Gambar 3.26 Kode Program Halaman *Settings*

3.3 Cara Kerja Perangkat

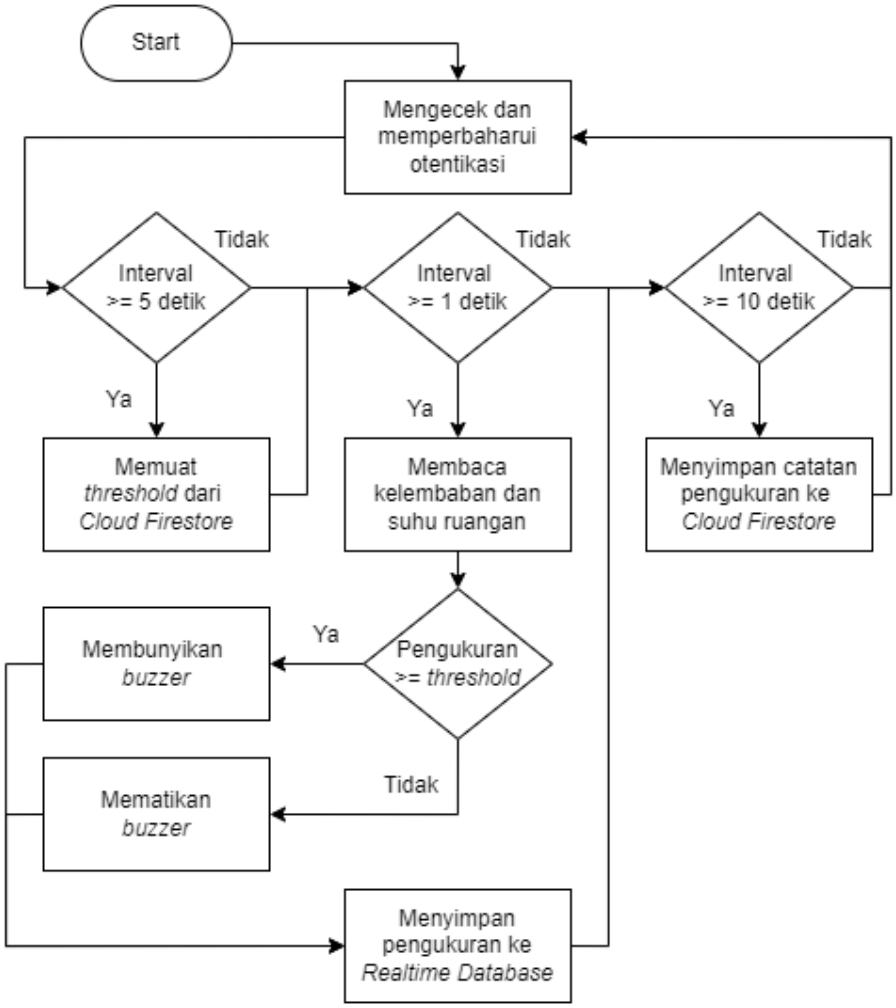
Sistem pemantauan kelembaban dan suhu ruangan bekerja dengan pertama perangkat keras mengukur kelembaban dan suhu melalui sensor DHT11 lalu kemudian mengelolanya untuk lebih mudah digunakan dan disimpan. Perangkat lalu mengirimkan data pengukuran ke layanan *database Firebase* sehingga dapat diakses oleh aplikasi berbasis web saat diperlukan pengguna. Saat aplikasi berbasis web digunakan, aplikasi akan memuat dan menampilkan hasil pengukuran kelembaban dan suhu oleh sensor. Alur penggunaan sistem pemantauan digambarkan pada Gambar 3.27.



Gambar 3.27 Alur Penggunaan Sistem

Perangkat keras pengukuran kelembaban dan suhu bekerja dengan pertama mengecek dan memperbaharui otentikasi. Setiap 5 detik, perangkat akan memuat *threshold* untuk *buzzer* dari *Cloud Firestore*. Setiap 1 detik perangkat akan membaca kelembaban dan suhu pada sensor DHT11, memutuskan untuk membunyikan atau mematikan *buzzer* berdasakan apakah pengukuran masing-masing sensor melebihi atau sama dengan *threshold*, lalu memperbaharui nilai

pengukuran yang tersimpan pada *Realtime Database*. Setiap 10 detik perangkat akan menyimpan catatan hasil pengukuran kelembaban dan suhu ke *Cloud Firestore*. Setelah itu perangkat akan kembali ke awal perulangan dimulai dari mengecek dan memperbaharui otentikasi jika token yang diberikan telah *expired*. Alur kerja perangkat keras pengukuran kelembaban dan suhu diilustrasikan pada Gambar 3.28.



Gambar 3.28 Alur Kerja Perangkat Keras

BAB 4

HASIL PENGUJIAN

4.1 Hasil Pengujian

Pengujian dilakukan untuk memastikan implementasi perangkat pemantauan kelembaban dan suhu ruangan sudah sesuai dengan perancangan. Hasil pengujian tertera pada Tabel 4.1.

Tabel 4.1 Hasil Pengujian Perangkat

No	Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Keterangan
1	Mengukur tingkat kelembaban dan suhu dengan sensor DHT11 sebelah kiri	Mendapatkan nilai tingkat kelembaban dan suhu ruangan dari sensor DHT11 sebelah kiri	Mendapatkan nilai tingkat kelembaban dan suhu ruangan dari sensor DHT11 sebelah kiri	Berhasil
2	Mengukur tingkat kelembaban dan suhu dengan sensor DHT11 sebelah kanan	Mendapatkan nilai tingkat kelembaban dan suhu ruangan dari sensor DHT11 sebelah kanan	Mendapatkan nilai tingkat kelembaban dan suhu ruangan dari sensor DHT11 sebelah kanan	Berhasil
3	Koneksi *Wi-Fi* ESP8266	ESP8266 terhubung ke *Wi-Fi*	ESP8266 terhubung ke *Wi-Fi*	Berhasil
4	Memperbaharui data pengukuran ke *Realtime Database*	Data sensor berhasil berhasil diperbaharui pada *Realtime Database*	Data sensor berhasil berhasil diperbaharui pada *Realtime Database*	Berhasil
5	Menambah data pengukuran ke *Cloud Firestore*	Data sensor berhasil berhasil ditambah ke *Cloud Firestore*	Data sensor berhasil berhasil ditambah ke *Cloud Firestore*	Berhasil

No	Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Keterangan
6	Memuat *threshold* dari *Cloud Firestore*	*Threshold* pada *Cloud Firestore* berhasil dimuat pada ESP8266	*Threshold* pada *Cloud Firestore* berhasil dimuat pada ESP8266	Berhasil
7	Membunyikan *buzzer* saat melewati *threshold*	*Buzzer* berbunyi saat pengukuran dari salah satu sensor lebih *threshold*	*Buzzer* berbunyi saat pengukuran dari salah satu sensor lebih *threshold*	Berhasil
8	Mematikan *buzzer* jika kurang dari *threshold*	*Buzzer* berhenti berbunyi saat pengukuran dari kedua sensor kurang *threshold*	*Buzzer* berhenti berbunyi saat pengukuran dari kedua sensor kurang *threshold*	Berhasil
9	Web memuat data pengukuran sensor DHT11 sebelah kiri	Data pengukuran sensor kiri dapat diakses oleh aplikasi web	Data pengukuran sensor kiri dapat diakses oleh aplikasi web	Berhasil
10	Web memuat data pengukuran sensor DHT11 sebelah kanan	Data pengukuran sensor kanan dapat diakses oleh aplikasi web	Data pengukuran sensor kanan dapat diakses oleh aplikasi web	Berhasil
11	Web memuat data dari *Realtime Database*	Data pada *Realtime Database* dapat diakses oleh aplikasi web	Data pada *Realtime Database* dapat diakses oleh aplikasi web	Berhasil
12	Web menampilkan data pengukuran sensor	Menampilkan kelembaban dan suhu yang tersimpan pada *Realtime Database*	Menampilkan kelembaban dan suhu yang tersimpan pada *Realtime Database* untuk setiap sisi bersama satuan unitnya.	Berhasil

No	Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Keterangan
13	Web menampilkan status ruangan berdasarkan pengukuran dan *threshold*	Menampilkan status ruangan sebagai "*Safe*" jika pengukuran tidak melebihi *threshold* dan "*Danger*" jika melebihi *threshold*	Menampilkan status ruangan sebagai "*Safe*" jika pengukuran tidak melebihi *threshold* dan "*Danger*" jika melebihi *threshold* dengan sebuah *overall* status yang menampilkan "*Safe*" saat keduanya "*Safe*" dan "*Danger*" saat salah satunya "*Danger*"	Berhasil
14	Web memuat data dari *Cloud Firestore*	Data pada *Cloud Firestore* dapat diakses oleh aplikasi web	Data pada *Cloud Firestore* dapat diakses oleh aplikasi web	Berhasil
15	Web menampilkan grafik pengukuran	Menampilkan grafik pengukuran beberapa catatan terakhir	Menampilkan grafik pengukuran 10 catatan terakhir dengan jeda 1 menit antar catatan	Berhasil
16	Web menampilkan tabel catatan pengukuran	Menampilkan tabel catatan pengukuran sensor	Menampilkan tabel catatan pengukuran sensor yang memiliki fitur *pagination* 10 catatan per *page* dan *filter* berdasarkan rentang waktu	Berhasil
17	Web menampilkan setelan *threshold*	Menampilkan *threshold* yang digunakan perangkat untuk membunyikan *buzzer*	Menampilkan *threshold* yang digunakan perangkat untuk membunyikan *buzzer*	Berhasil

No	Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Keterangan
18	Web mampu mengubah setelan *threshold*	Pengguna dapat mengubah *threshold* yang digunakan perangkat untuk membunyikan *buzzer*	Pengguna dapat mengubah *threshold* yang digunakan perangkat untuk membunyikan *buzzer*	Berhasil

Dilampirkan pada Tabel 4.2 hasil uji coba pengukuran kelembaban dan suhu yang dilakukan selama 2 jam dari jam 09:25 – 11:15 WIB. Uji coba mengukur kelembaban dan suhu setiap 10 detik dan menghitung rata-rata setiap 60 catatan atau dalam periode 10 menit.

Tabel 4.2 Uji Coba Pengukuran Kelembaban dan Suhu

Jam	DHT Kiri		DHT Kanan	
	Suhu	Kelembaban	Suhu	Kelembaban
09:25	27.51	67.93	27.6	61
09:35	27.6	67.48	27.6	61
09:45	27.56	67.9	27.6	61.85
09:55	27.6	67.9	27.6	61.9
10:05	27.6	67.85	27.6	62
10:15	27.6	68	27.79	61.58
10:25	27.6	68.03	28	61
10:35	27.6	68.87	28	61.55
10:45	27.6	69	28	62
10:55	27.7	68.88	28	62
S11:05	27.71	68.37	28	61.73
11:15	27.94	68.2	28	61.88
Rata-rata	**27.64**	**68.20**	**27.82**	**61.63**

BAB 5

PENUTUP

5.1 Kesimpulan

Berdasarkan pengujian yang telah penulis lakukan, bisa disimpulkan bahwa penggunaan sensor DHT11 sangat berfungsi dengan baik untuk memantau suhu dan kelembapan di titik tertentu atau tempat yang memang mempunyai potensi kebakaran. Dengan pemantauan yang dilakukan secara *real time* dan melihat informasi secara mudah yang hanya dengan menggunakan *smartphone* yang sudah terhubung ke *internet*. Dari hasil pengujian dan analisis dengan 2 sensor DHT11 yang telah dilakukan, didapati hasil kondisi suhu dan kelembapan untuk sensor DHT11 sebelah kiri dengan suhu rata-rata di 27,64 dan kelembapan di rata-rata 68.20, sedangkan untu DHT11 sebelah kanan dengan suhu rata-rata 27,82 dan kelembapan di rata-rata 61,63.

5.2 Saran

Berikut saran yang dapat dipertimbangkan untuk pengembangan lanjutan terkait penelitian yang serupa.

1. *Framework* bisa diganti dengan *framework* lain yang mendukung.
2. Bisa digabungkan dengan alat penyiram air otomatis dengan pemicu sensor DHT11 ini.

REFERENSI

Adiptya, M. Y. E., & Wibawanto, H. (2013). *Sistem pengamatan suhu dan kelembaban pada rumah berbasis mikrokontroller atmega8*. *5*(1), 15–17.

Aggarwal, S. (2018). Modern web-development using reactjs. *International Journal of Recent Research Aspects*, *5*(1), 133–137.

Anam, S., Wijaya, I. D., & Rismanto, R. (2020). Rancang bangun sistem deteksi dan pemadam kebakaran pada smart home menggunakan metode fuzzy. *Jurnal Informatika Polinema*, *6*(4), 9–16. https://doi.org/10.33795/jip.v6i4.298

Apriani, N. D., Rachmatullah, M. A., Sukamto, R., & Apriani, Y. (2021). Powerbank laptop portable sebagai sumber energi mobile. *Jurnal Rekayasa Elektro Sriwijaya*, *3*(1), 205–212. https://doi.org/10.36706/jres.v3i1.44

Artist-3D. (2023, November 6). *What is a universal pcb?* Artist-3D. https://artist-3d.com/universal-pcb/

Aspari, R., Lalu Delsi Samsumar, Emi Suryadi, Ardiyallah Akbar, & Zaenudin. (2024). Sistem monitoring suhu dan kelembapan pada kandang ayam broiler berbasis internet of things untuk meningkatkan produksi. *Journal of Computer Science and Information Technology*, *1*(4), 351–358. https://doi.org/10.70248/jcsit.v1i4.1285

Baehaqi, M., Rosyid, A., Siswanto, A., & Subiyanta, E. (2023). Pengujian performa sensor dht11 dan ds18b20 sebagai sensor suhu ruang server. *Jurnal Mestro*, *5*(2), 6–11.

BlowStack. (2022, Oktober 16). *What is firebase and how to use it with web apps*. BlowStack. https://blowstack.com/blog/what-is-firebase-and-how-to-use-it-with-web-apps

BPS. (2024). *Persentase rumah tangga menurut provinsi dan status kepemilikan bangunan tempat tinggal yang ditempati milik sendiri (persen)*. https://www.bps.go.id/id/statistics-table/2/ODQ5IzI=/persentase-rumah-tangga-menurut-provinsi-dan-status-kepemilikan-rumah-milik-sendiri.html

Depok Pos. (2022, Desember 7). *Gandeng Tuya, Telkom kembangkan ekosistem smart home di Indonesia*. Depok Pos.

https://www.depokpos.com/2022/12/gandeng-tuya-telkom-kembangkan-ekosistem-smart-home-di-indonesia/

Dewi, L. P. (2021). *Kajian suhu kelembaban pencahayaan dan kelelahan kerja pada pekerja industri batik "x" di Pijenan Wijirejo Pandak Bantu* [Diploma]. Politeknik Kesehatan Kementrian Kesehatan Yogyakarta.

Google Firebase. (t.t.). *Products build*. Google Firebase. Diambil 2 Januari 2025, dari https://firebase.google.com/products-build

Hermawan, R., & Abdurrohman, A. (2020). Pemanfaatan teknologi internet of things pada alarm sepeda motor menggunakan nodemcu lolin v3 dan media telegram. *Infotronik : Jurnal Teknologi Informasi dan Elektronika*, *5*(2), 58. https://doi.org/10.32897/infotronik.2020.5.2.453

Hidayat, D., & Sari, I. (2021). Monitoring suhu dan kelembaban berbasis internet of things (iot). *JURNAL TEKNOLOGI DAN ILMU KOMPUTER PRIMA (JUTIKOMP)*, *4*(1), 525–530. https://doi.org/10.34012/jutikomp.v4i1.1676

IoT Kece. (2022, Maret 6). *Cara kerja sensor dht11 (sensor suhu dan kelembaban)*. IoT Kece. https://iotkece.com/cara-kerja-sensor-dht11-sensor-suhu-dan-kelembaban

Kabasi. (2021, Juni 10). *Apa itu kabel usb?* Kabasi. https://id.kbs-connector.com/news/what-is-usb-cable-46819198.html

Kumparan. (2023, November 21). *Definisi kabel jumper beserta fungsi dan jenisnya*. Kumparan. https://kumparan.com/berita-update/definisi-kabel-jumper-beserta-fungsi-dan-jenisnya-21ERiC1PFiD/full

Kusuma, A. Y., Pratikno, H., & Puspasari, I. (2020). Rancang bangun alat pelipat baju otomatis menggunakan arduino uno. *Journal of Control and Network Systems*, *9*(2), 8–18.

Lilley, D. (2012, Juni 26). Three minutes from ignition to inferno: It's not unusual in fires. *1st International Energy Conversion Engineering Conference (IECEC)*. https://doi.org/10.2514/6.2003-6052

Loginov, D. (2018, Desember 12). Quick start with esp8266 or esp32 in the arduino ecosystem using platformio ide. *Medium*. https://loginov-rocks.medium.com/quick-start-with-nodemcu-v3-esp8266-arduino-ecosystem-and-platformio-ide-b8415bf9a038

Manyika, J., Chui, M., Bisson, P., Woetzel, J., Dobbs, R., Bughin, J., & Aharon, D. (2015). *The internet of things: Mapping the value beyond the hype*.

Murti, S. K., & Sujarwo, A. (2021, Agustus 31). Membangun antarmuka pengguna menggunakan reactjs untuk modul manajemen pengguna. *Prosiding Automata*.

Nasution, & Iswari, L. (2021, Agustus 31). Penerapan react js pada pengembangan frontend aplikasi startup ubaform. *Prosiding Automata*.

Nugraha, P. D., Soekarta, R., & Amri, I. (2023). Rancang bangun alat monitoring suhu dan kelembaban berbasis internet of things (iot) pada gudang obat Rumah Sakit Aryoko Sorong. *Framework : Jurnal Ilmu Komputer dan Informatika*, *2*(01), 21–30. https://doi.org/10.33506/jiki.v2i01.3044

Pramoedya, D. N. (t.t.). *Sistem suhu dan kelembapan esp8266 dan dht11*. Bisa.AI. Diambil 2 Januari 2025, dari https://bisa.ai/portofolio/detail/MzA4Nw

Prastyo, E. A. (2022, November 21). *Pengertian, jenis dan cara kerja kabel jumper arduino*. Arduino Indonesia. https://www.arduinoindonesia.id/2022/11/pengertian-jenis-dan-cara-kerja-kabel-jumper-arduino.html

Priyono, N. Y. (2017). *Sistem peringatan dini banjir berbasis protocol mqtt menggunakan nodemcu esp8266* [Diploma]. Sekolah Tinggi Manajemen Informatika dan Komputer Akakom .

Rangan, A. Y., Amelia Yusnita, & Muhammad Awaludin. (2020). Sistem monitoring berbasis internet of things pada suhu dan kelembaban udara di laboratorium kimia xyz. *Jurnal E-Komtek (Elektro-Komputer-Teknik)*, *4*(2), 168–183. https://doi.org/10.37339/e-komtek.v4i2.404

Saepudin, A. (2022). Teknologi internet of things dalam proses monitoring suhu dan kelembaban di gudang penyimpanan bahan kulit. *Jurnal Teknik Informatika dan Sistem Informasi*, *9*(4), 2712–2719.

Tekkom. (2023, Januari 18). *Apa itu internet of things?* Tekkom. https://tekkom.upi.edu/2023/01/apa-itu-internet-of-things-nodemcu-esp8266/

Virtualab IoT. (t.t.). *Buzzer*. Virtualab IoT. Diambil 2 Januari 2025, dari https://te.eng.uho.ac.id/virtualab/manager/buzzer.html

Visual Studio Marketplace. (2024). *PlatformIO IDE* (v3.3.3). Visual Studio Marketplace.

https://marketplace.visualstudio.com/items?itemName=platformio.platformio-ide

Waher, P. (2021). *Mastering internet of things: Design and create your own iot applications using raspberry pi 3* (1 ed., Vol. 1). Packt Publishing.

Yuliadi, Solihat, N. M., & Herfandi. (2021). Rekayasa aplikasi center rumah kost berbasis web di Kabupaten Sumbawa. *Jurnal Manajemen Informatika dan Sistem Informasi*, *4*(2).

Zella. (t.t.). *Percobaan esp32 sensor dht 11*. Scribd. Diambil 2 Januari 2025, dari https://www.scribd.com/document/676342815/Percobaan-ESP32-Sensor-DHT-11

LAMPIRAN


lexar