RANCANG BANGUN DASHBOARD IOT UNTUK MONITORING KOLAM AKUAPONIK BERBASIS ANDROID

TUGAS AKHIR

Program Studi

S1 TEKNIK KOMPUTER

Oleh:

Kevin Jonathan

19410200005

FAKULTAS TEKNOLOGI DAN INFORMATIKA

UNIVERSITAS DINAMIKA

2023

RANCANG BANGUN DASHBOARD IOT UNTUK MONITORING KOLAM AKUAPONIK BERBASIS ANDROID

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk menyelesaikan
Program Sarjana Teknik

Disusun Oleh:

Nama : Kevin Jonathan
NIM : 19410200005
Program : S1 (Strata Satu)
Jurusan : Teknik Komputer

FAKULTAS TEKNOLOGI DAN INFORMATIKA
UNIVERSITAS DINAMIKA

2023

TUGAS AKHIR

RANCANG BANGUN *DASHBOARD* IOT UNTUK *MONITORING* KOLAM AKUAPONIK BERBASIS ANDROID

Dipersiapkan dan disusun oleh:

Kevin Jonathan

NIM: 19410200005

Telah diperiksa, dibahas, dan disetujui oleh Dewan Pembahas

Pada: 5 Agustus 2023

Susunan Dewan Pembahas

Pembimbing:

I. **Harianto, S.Kom., M.Eng.**

NIDN 0722087701

cn=Harianto Harianto, o=Universitas Dinamika, ou=Prodi S1 Teknik Komputer, email=hari@dinamika.ac.id, c=ID 2023.08.09 09:36:34 +07'00'

II. **Pauladie Susanto, S.Kom., M.T.**

NIDN 0729047501

cn=Pauladie Susanto, o=FTI Undika, ou=Prodi S1 TK, email=pauladie@dinamika.ac.id, c=ID 2023.08.09 12:50:24 +07'00'

Pembahas:

Heri Pratikno, M.T., MTCNA., MTCRE.

NIDN 0716117302

Digitally signed by Heri Pratikno, M.T. DN: cn=Heri Pratikno, M.T., o=Universitas Dinamika, ou=S1 Teknik Komputer, email=heri@dinamika.ac.id, c=ID Date: 2023.08.09 13:47:50 +07'00' Adobe Acrobat version: 11.0.23

Tugas Akhir ini telah diterima sebagai salah satu persyaratan

untuk memperoleh gelar sarjana

Digitally signed by Universitas Dinamika Date: 2023.08.16 08:38:20 +07'00'

Tri Sagirani, S.Kom., M.MT.

NIDN: 0731017601

Dekan Fakultas Teknologi dan Informatika

UNIVERSITAS DINAMIKA

“Just do what you want to do.”

~Kevin Jonathan~

Dipersembahkan kepada Orang Tua saya dan semua orang yang selalu membantu, memberikan dukungan dan memberikan motivasi kepada saya.

PERNYATAAN

PERSETUJUAN PUBLIKASI DAN KEASLIAN KARYA ILMIAH

Sebagai mahasiswa **Universitas Dinamika**, Saya :

Nama	: Kevin Jonathan
NIM	19410200005
Program Studi	: S1 Teknik Komputer
Fakultas	: Fakultas Teknologi dan Informatika
Jenis Karya	: Laporan Tugas Akhir
Judul Karya	: **RANCANG BANGUN DASHBOARD IOT UNTUK MONITORING KOLAM AKUAPONIK BERBASIS ANDROID**

Menyatakan dengan sesungguhnya bahwa :

1. Demi pengembangan Ilmu Pengetahuan, Teknologi dan Seni, Saya menyetujui memberikan kepada **Universitas Dinamika** Hak Bebas Royalti Non-Eksklusif (***Non-Exclusive Royalty Free Right***) atas seluruh isi/sebagian karya ilmiah Saya tersebut diatas untuk disimpan, dialihmediakan, dan dikelola dalam bentuk pangkalan data (***database***) untuk selanjutnya didistribusikan atau dipublikasikan demi kepentingan akademis dengan tetap mencantumkan nama Saya sebagai penulis atau pencipta dan sebagai pemilik Hak Cipta.
2. Karya tersebut diatas adalah hasil karya asli Saya, bukan plagiat baik sebagian maupun keseluruhan. Kutipan, karya, atau pendapat orang lain yang ada dalam karya ilmiah ini semata-mata hanya sebagai rujukan yang dicantumkan dalam Daftar Pustaka Saya.
3. Apabila dikemudian hari ditemukan dan terbukti terdapat tindakan plagiasi pada karya ilmiah ini, maka Saya bersedia untuk menerima pencabutan terhadap gelar kesarjanaan yang telah diberikan kepada Saya.

Demikian surat pernyataan ini Saya buat dengan sebenar–benarnya.

Surabaya, 19 Juni 2023

Kevin Jonathan
NIM : 19410200005

ABSTRAK

Internet of Things atau biasa disingkat IoT merupakan teknologi yang sedang tren beberapa tahun terakhir karena teknologi ini dapat menghubungkan beberapa perangkat elektronik melalui jaringan internet. Teknologi IoT ini dapat diterapkan diberbagai sektor, contohnya pada Tugas Akhir ini IoT akan diimplementasikan dengan kolam Akuaponik sehingga pemilik kolam dapat mengetahui kondisi kolam akuaponik seperti pH, TDS, dan kekeruhan air melalui aplikasi *Smartphone* atau yang biasa disebut dengan *Dashboard* IoT. Saat ini sudah banyak *platform* yang menyediakan layanan *Dashboard* IoT namun fitur yang disediakan terbatas dan harus membayar untuk membuka semua fitur yang disediakan. Oleh karena itu pada Tugas Akhir ini penulis membuat *Dashboard* IoT sendiri dengan menggunakan *framework* Javascript yaitu React Native, lalu untuk komunikasi data antara aplikasi dan kontroler menggunakan salah satu fitur dari Firebase yaitu *Realtime Database*. Dengan membuat *Dashboard* IoT sendiri kita dapat menyesuaikan fitur apa saja yang akan dibutuhkan di dalam aplikasi kita.

Kata Kunci*:* IoT, *Dashboard* IoT, *Akuaponik, Monitoring, React*

KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Esa karena berkat dan rahmatNya sehingga penulis dapat menyelesaikan laporan Tugas Akhir dengan judul " Rancang Bangun *Dashboard* IoT Untuk *Monitoring* Kolam Akuaponik Berbasis Android". Pada kesempatan ini, penulis ingin mengucapkan terimakasih kepada:

1. Orang Tua, yang selalu memberikan motivasi dan dukungan sehingga penulis dapat mengerjakan dan menyusun laporan tugas akhir ini
2. Ibu Tri Sagirani, S.Kom., M.MT. selaku Dekan Fakultas Teknologi dan Informatika Universitas Dinamika.
3. Bapak Pauladie Susanto, S.Kom., M.T., selaku Ketua Program Studi S1 Teknik Komputer Universitas Dinamika dan Dosen Pembimbing II yang selalu meluangkan waktunya dan memberikan bimbingan kepada penulis sehingga dapat penulis menyelesaikan Tugas Akhir ini
4. Bapak Harianto, S.Kom., M.Eng., selaku Dosen Pembimbing I yang selalu meluangkan waktunya dan memberikan bimbingan kepada penulis sehingga dapat penulis menyelesaikan Tugas Akhir ini
5. Heri Pratikno, M.T., MTCNA., MTCRE. selaku Dosen Pembahas yang selalu memberi waktu dan bimbingan dalam menyelesaikan laporan Tugas Akhir ini.
6. Seluruh Dosen Program Studi S1 Teknik Komputer yang telah mendidik dan membimbing penulis selama masa studi di Universitas Dinamika
7. Teman-teman S1 Teknik Komputer angkatan 2019 dan Dinamika Robotik yang telah membantu dan memberikan dukungan kepada penulis selama proses pengerjaan Tugas Akhir ini
8. Semua pihak yang tidak dapat disebutkan satu persatu yang telah memberikan bantuan dan dukungan baik secara langsung dan tidak langsung dalam proses pengerjaan Tugas Akhir ini.

Penulis Berharap agar laporan ini dapat memberikan manfaat yang berarti dan dapat meningkatkan pemahaman bagi pembacanya, dan juga penulis menyadari adanya

kekurangan dalam penyusunan laporan ini. Oleh karena itu dengan rendah hati penulis menerima segala kritik yang konstruktif sehingga dapat memperbaiki kekurangan.

Surabaya, 20 Juni 2023

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Selama beberapa abad terakhir, perkembangan teknologi mengalami kemajuan yang sangat signifikan terutama sejak revolusi Industri pada awal abad ke-19. Pada saat itu teknologi seperti Mesin uap dapat membuat kegiatan manusia menjadi lebih efektif dan efisien.lalu pada abad ke-20, internet, televisi, dan komputer ditemukan. Tidak berhenti di situ, sampai saat ini di abad ke-21 beberapa teknologi canggih seperti GPS, Sosial Media, Kecerdasan Buatan dan IoT (Internet of Thing) telah ditemukan. Ini merupakan bukti bahwa semakin hari teknologi semakin berkembang dan menjadi semakin kompleks, akibatnya cara manusia bekerja, belajar, bersosialisasi dan berkomunikasi ikut berubah sesuai dengan teknologi yang berkembang.

Salah satu contoh teknologi masa kini yang beberapa tahun terakhir sedang trend adalah *Internet of Things* atau bisa disingkat IoT. Konsep dasar dari IoT adalah menghubungkan beberapa perangkat elektronik agar bisa saling berkomunikasi melalui jaringan internet, dengan begitu kita dapat mengumpulkan informasi dan data mengenai perangkat yang telah terhubung dan menggunakannya untuk berbagai macam tujuan. Sebagai contoh, pada tugas akhir ini, IoT akan diimplementasikan pada kolam akuaponik sehingga *user* bisa memantau pH, kekeruhan air dan nutrisi pada kolam akuaponik hanya dengan menggunakan *Smartphone*.

Tentunya untuk bisa memantau kolam akuaponik kita memerlukan *user* interface atau antarmuka yang dapat menghubungkan *user* dengan sistem, biasanya antarmuka ini disebut sebagai *Dashboard* IoT. *Dashboard* IoT ini sangat berperan besar karena segala informasi mengenai perangkat akan ditampilkan pada *Dashboard* ini, bahkan *user* bisa memantau perangkat secara real-time sehingga jika terjadi masalah *user* dapat mengambil tindakan yang tepat.

Sebagai contoh, pada penelitian yang berjudul “Sistem Monitoring Kualitas Air Pada Sistem Akuaponik Berbasis Iot” yang membuat sebuah sistem untuk memonitoring kolam akuaponik (Widodo, Alfia, Nurhayati, & Kholis, 2021). namun,

terdapat beberapa kekurangan pada penelitian tersebut karena sistem *monitoring* dibuat menggunakan web server dan telegram untuk memantau kolam aquaponik, kekurangan dari menggunakan web server adalah pengguna tidak bisa memonitoring melalui jarak jauh dikarenakan perangkat *monitoring* harus terhubung dengan jaringan lokal, lalu kekurangan menggunakan telegram pada penelitian tersebut adalah *user* harus mengirimkan pesan "Aquaponik" agar bot telegram dapat mengirimkan hasil *monitoring* dari kolam akuaponik. lalu pada penelitian yang berjudul "Sistem Monitoring Berbasis Internet of Thing (IoT) Untuk Pengendalian Kualitas Air dan Pakan Ikan pada Budidaya sistem Akuaponik" (Danih & Sugiyanto, 2021) yang membuat sistem *monitoring* dengan menggunakan *Platform* IoT pihak ketiga yaitu Blynk, kekurangan dari menggunakan aplikasi pihak ketiga seperti Blynk adalah kita tidak bisa dengan fleksibel untuk menentukan fitur apa saja yang ingin kita masukan ke dalam *Dashboard* IoT kita, karena ada beberapa fitur yang berbayar.

Oleh karena itu untuk melengkapi kekurangan dari dua penelitian di atas, pada tugas akhir ini, dibuatlah sebuah *Dashboard* IoT untuk memonitoring kolam akuaponik berbasis Android dengan menggunakan *framework* javascript yaitu react native dan menggunakan salah satu fitur dari Firebase yaitu *Realtime Database*. namun masih terdapat kekurangan pada tugas akhir ini karena kita harus memprogram *Dashboard* IoT dari awal agar bisa berkomunikasi dengan sensor dan menampilkan data hasil pembacaan sensor dengan benar.

1.2 Rumusan Masalah

Berdasarkan latar belakang maka dapat dirumuskan masalah pada Tugas Akhir adalah bagaimana mengembangkan sebuah aplikasi *Dashboard* IoT yang *user friendly* agar *user* dapat memonitoring kolam akuaponik melalui *smartphone*

1.3 Batasan Masalah

Dalam pembuatan tugas akhir ini, pembahasan masalah dibatasi pada pembuatan aplikasi *Dashboard* IoT pada sistem operasi Android yang meliputi :

1. *Framework* yang digunakan untuk membangun aplikasi *Dashboard* IoT adalah React Native.
2. Komunikasi data antara software dan hardware menggunakan fitur dari Firebase yaitu *Realtime Database*

1.4 Tujuan

Berdasarkan latar belakang dan rumusan masalah di atas didapatkan tujuan dari Tugas Akhir ini adalah dapat mengembangkan aplikasi *Dashboard* IoT yang *user friendly* agar *user* dapat memonitoring kolam akuaponik.

1.5 Manfaat

Adapun manfaat dari Tugas Akhir ini adalah mengimplementasikan IoT pada kolam akuaponik sehingga pemilik kolam bisa memantau kolamnya melalui jarak jauh dengan *Smartphone*

BAB II
LANDASAN TEORI

2.1 *Internet of Things*

IoT (*Internet of Things*) merupakan sebuah konsep yang memiliki tujuan untuk memperluas manfaat dari konektivitas internet yang tersambung secara terus-menerus. Pada dasarnya, IoT mengacu pada benda yang dapat diidentifikasikan secara unik sebagai representasi virtual dalam struktur berbasis internet (Salamah, Taqwa, & Wibowo, 2020)

IoT (*Internet of Things*) adalah sebuah teknologi yang dapat menghubungkan beberapa objek seperti peralatan rumah tangga, perangkat elektronik dan kendaraan melalui jaringan internet. Biasanya setiap objek IoT terhubung dengan sensor, sehingga dapat memungkinkan untuk mengumpulkan data dan bertukar data yang selanjutnya data tersebut dapat diolah dan dimanfaatkan sesuai dengan kebutuhan. Dapat dilihat teknologi ini mempunyai potensi untuk mengubah berbagai aspek kehidupan termasuk di bidang industri, kesehatan, transportasi dan sektor lainnya.

2.2 Firebase

Firebase adalah sebuah *platform* yang menyediakan layanan database dan backend (BaaS) yang bersifat *Realtime*. Firebase memiliki *Library* yang 5 pengembangan aplikasi web maupun mobile. Firebase telah menyediakan layanan API (Application Programming Interface) yang berfungsi untuk mengintegrasikan antara aplikasi dengan database atau dengan aplikasi lainnya. (Leonardo, Arwano, & Ratnawati, 2020). Tidak hanya menyediakan layanan *Database* dan backend *Realtime*, Firebase juga menyediakan berbagai fitur siap pakai yang dapat digunakan oleh pengembang untuk mengembangkan aplikasi web dan *mobile*. Beberapa fitur tersebut antara lain adalah Hosting, Authentication, Cloud Firestore, Cloud Storage, Cloud Messaging dan masih banyak lagi yang lainya.

Gambar 2.1 Logo Firebase

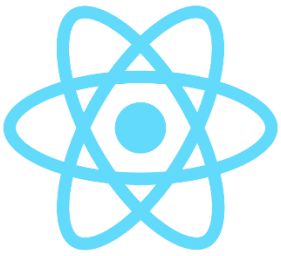
(Firebase, n.d.)

2.2.1 Firebase *Realtime Database*

Firebase *Realtime Database* adalah *database* NoSQL yang di-hosting di *cloud* dan dapat digunakan untuk menyimpan dan mensinkronkan data antar pengguna secara *Realtime*. **(Google, n.d.)**. Pada Tugas Akhir ini *Realtime Database* digunakan untuk menghubungkan kontroler dengan aplikasi *Dashboard* IoT agar dapat saling berkomunikasi.

2.3 React Native

React Native adalah sebuah *framework* berbasis Javascript yang digunakan untuk mengembangkan aplikasi *mobile* di dua sistem operasi secara bersamaan, yaitu Android dan iOS. React Native sendiri pertama kali diluncurkan pada tahun 2015 oleh Facebook dan bersifat *open source*. **(Setiawan, 2021)**. Dalam Tugas Akhir ini React Native digunakan sebagai kerangka utama untuk membuat aplikasi *Dashboard IoT* pada sistem operasi Android.

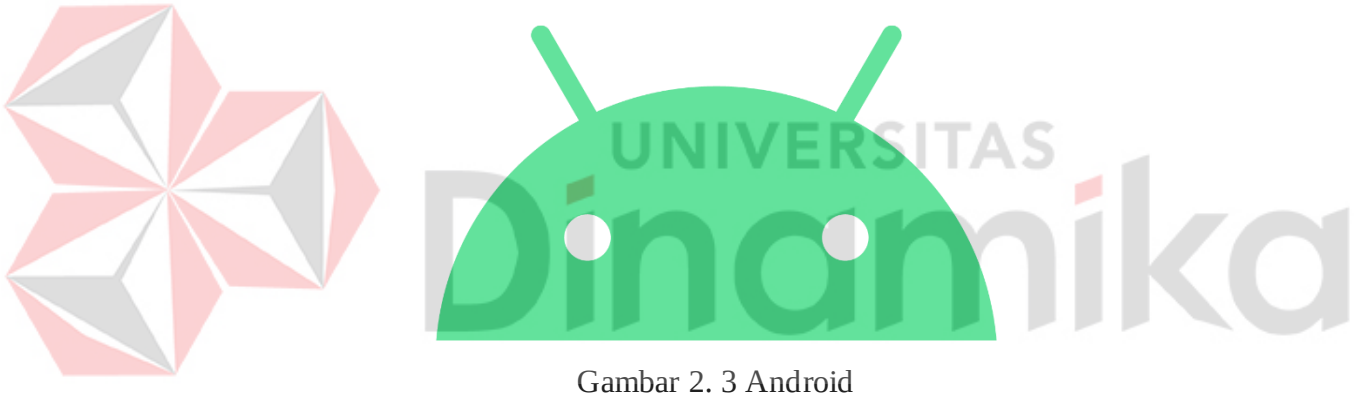


Gambar 2.2 Logo React Native

(McDaniel, 2020)

2.4 Android

Android merupakan sebuah sistem operasi yang bersifat open source. Artinya programmer diizinkan oleh pengembang sistem operasi untuk membuat, mengubah, mengembangkan dan menyebarluaskan aplikasi. (Irsan, 2015). Karena merupakan sistem operasi yang bersifat *open source*, pengembang diizinkan untuk membuat dan mengembangkan aplikasi dengan bebas. Dalam pembuatan *Dashboard* IoT, Android menjadi pilihan yang menarik karena beberapa alasan antara lain karena Android kompatibel dengan berbagai perangkat dengan berbagai spesifikasi dan ukuran layar yang berbeda. Android juga terintegrasi dengan layanan Google yang menyediakan berbagai layanan seperti Google Maps, Google Cloud dan Google Drive yang dapat memperkaya fungsionalitas aplikasi IoT.



Gambar 2. 3 Android

(Android, 2020)

2.5 Javascript

JavaScript adalah bahasa pemrograman web yang bersifat Client Side Programming Language. Client Side Programming Language adalah tipe bahasa pemrograman yang pemrosesannya dilakukan oleh *client*. Aplikasi *client* yang dimaksud merujuk kepada web browser seperti Google Chrome dan Mozilla Firefox (Miftah Farid Adiwisastra, Agung Baitul Hikmah, & Ai ilah Warnilah, 2019). Bahasa pemrograman Javascript digunakan karena aplikasi *Dashboard* IoT dibuat dengan menggunkan *framework* Javascript yaitu React Native.

Gambar 2. 4 Logo Javascript

(Dicoding, 2020)

2.6 Figma

Figma adalah salah satu *design tool* yang biasanya digunakan untuk membuat tampilan aplikasi *mobile*, desktop, website dan lain-lain. Figma bisa digunakan di sistem operasi windows, linux ataupun mac dengan terhubung ke internet (Muhyidin, Sulhan, & Sevtiana, 2020).

Gambar 2. 5 Logo Figma

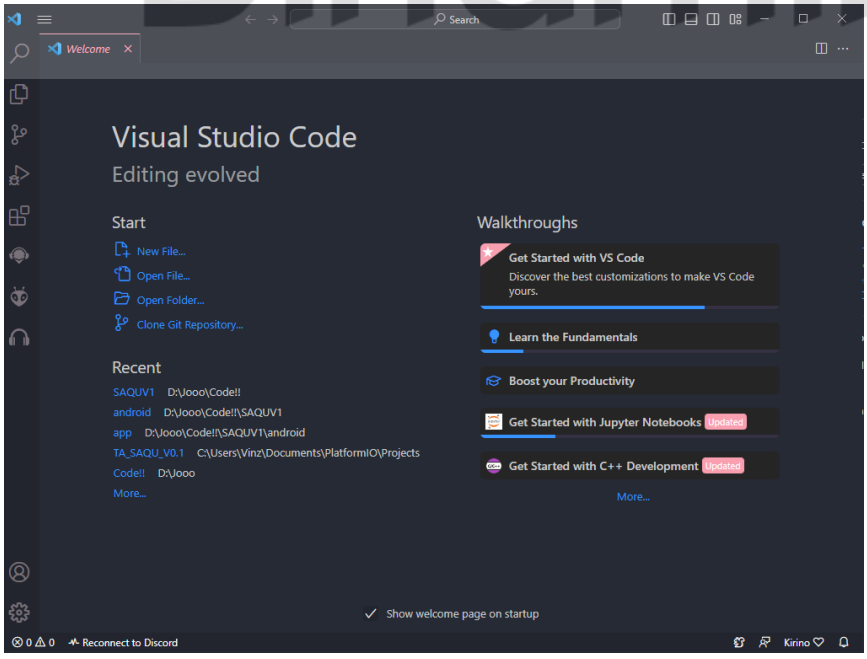
(Figma, 2017)

2.7 Visual Studio Code

Visual Studio Code ini adalah sebuah teks editor ringan dan handal yang dibuat oleh Microsoft untuk sistem operasi multiplatform, artinya tersedia juga untuk versi Linux, Mac, dan Windows. Teks editor ini secara langsung mendukung 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, dst) (Permana & Puji, 2019).

Gambar 2. 6 Logo Visual Studio Code

(Chris, 2021)

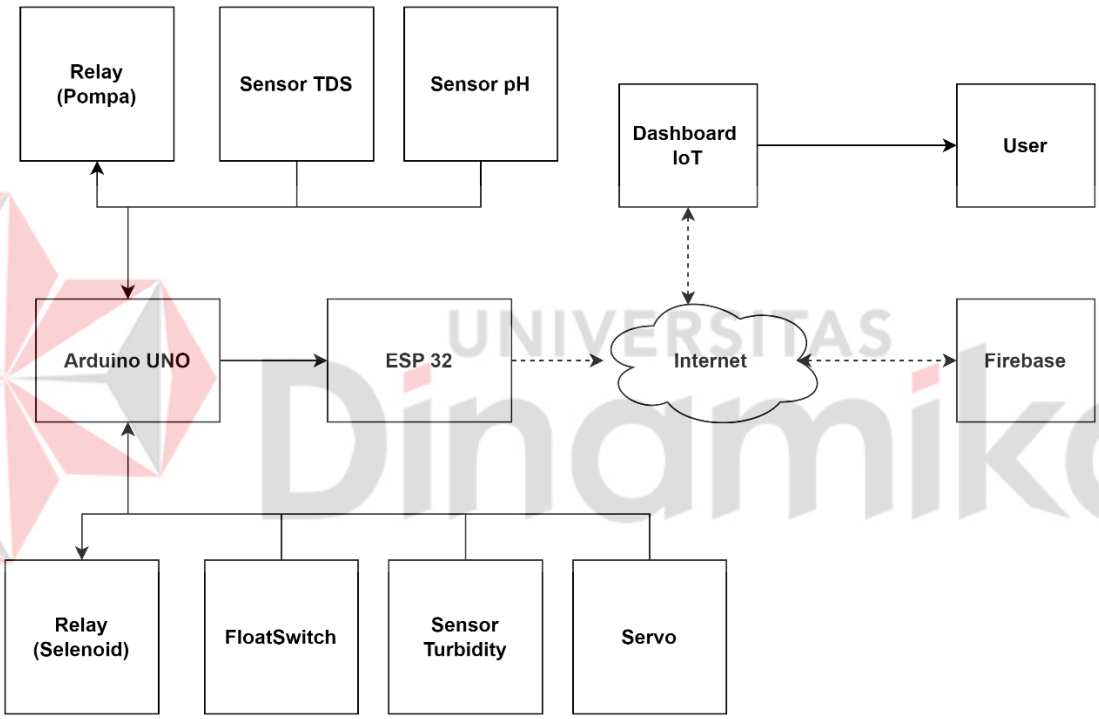


Gambar 2. 7 Tampilan Visual Studio Code

BAB III
METODOLOGI PENELITIAN

3.1 Blok Diagram

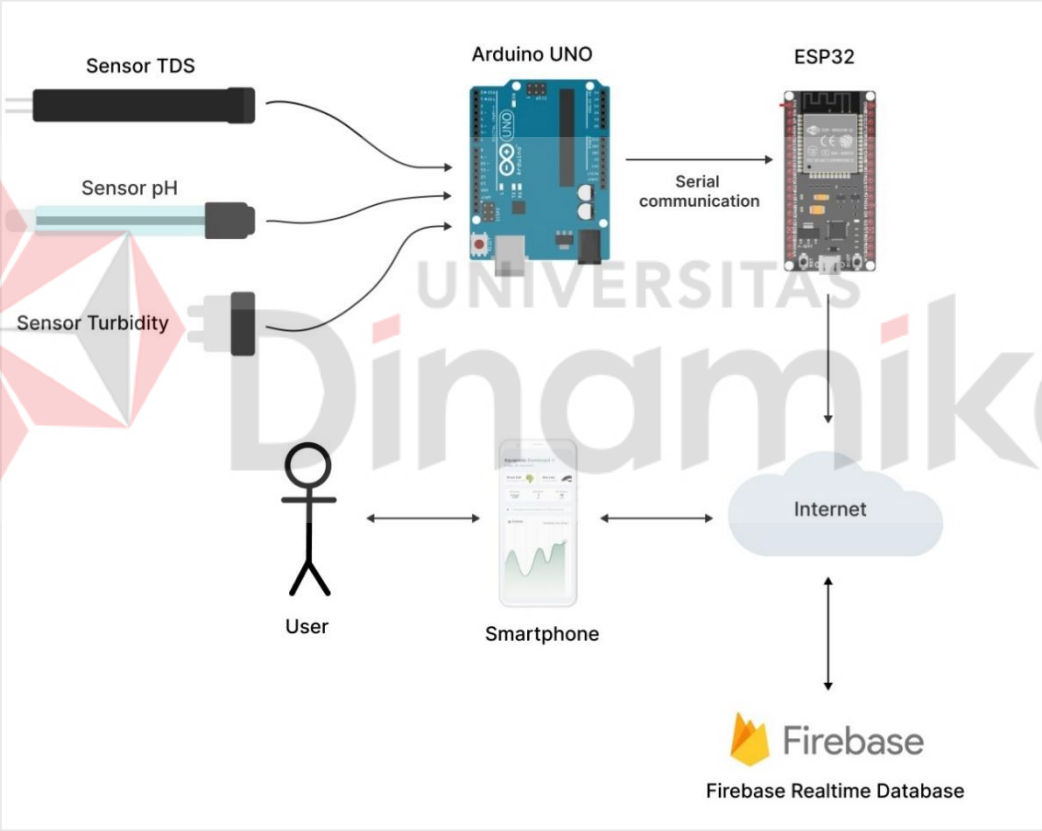
Pada Gambar 3.1 dapat dilihat data hasil pembacaan sensor pada kolam dan tanaman akan dikirimkan ke Firebase melalui kontroler lalu *Dashboard* akan mengambil data hasil pembacaan sensor yang tersedia di dalam Firebase untuk ditampilkan ke *user*



Gambar 3.1 Blok Diagram

3.2 Skema / Blok Diagram IoT

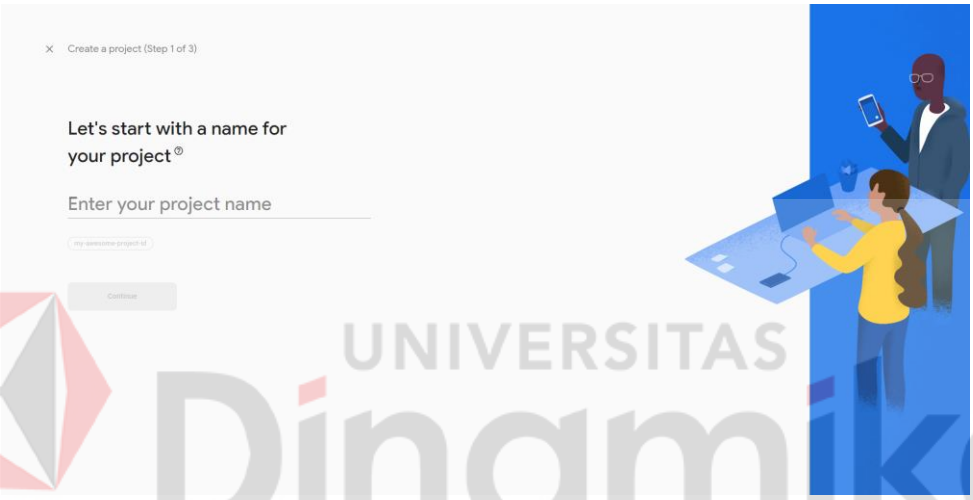
Gambar 3.2 merupakan skema / blok diagram mengenai *project* ini. Pada blok diagram Arduino UNO akan menerima hasil pembacaan sensor. Setelah Arduino UNO menerima hasil pembacaan sensor, data tersebut akan diteruskan Nodemcu ESP32 untuk dimasukkan ke dalam *Realtime Database* melalui jaringan internet, lalu *smartphone* yang terkoneksi dengan jaringan internet akan mengambil data yang ada di dalam *Realtime Database* dan ditampilkan ke dalam aplikasi sehingga *user* dapat melihat hasil pembacaan sensor.



Gambar 3.2 Blok Diagram IoT

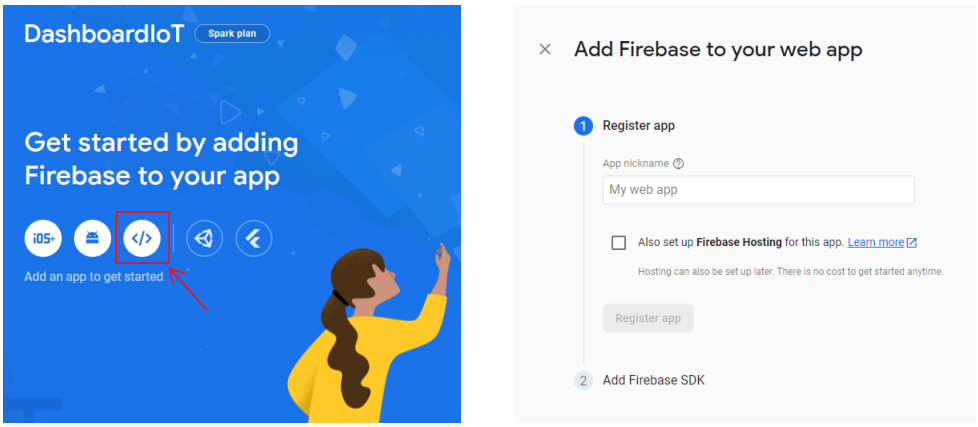
3.3 Mengambil Data dari Firebase

Untuk mengambil data yang tersimpan di dalam *Realtime Database*, React Native harus terhubung terlebih dahulu dengan *Realtime Database*. Untuk dapat terhubung ada beberapa tahapan yang harus dilakukan. Langkah pertama kita perlu membuat *project* baru pada Firebase dengan masuk ke halaman https://console.firebase.google.com/ sebagaimana tampak pada Gambar 3.3, setelah itu pilih *Add project* dan masukan nama *project*, lalu klik *continue* hingga proses pembuatan *project* selesai.



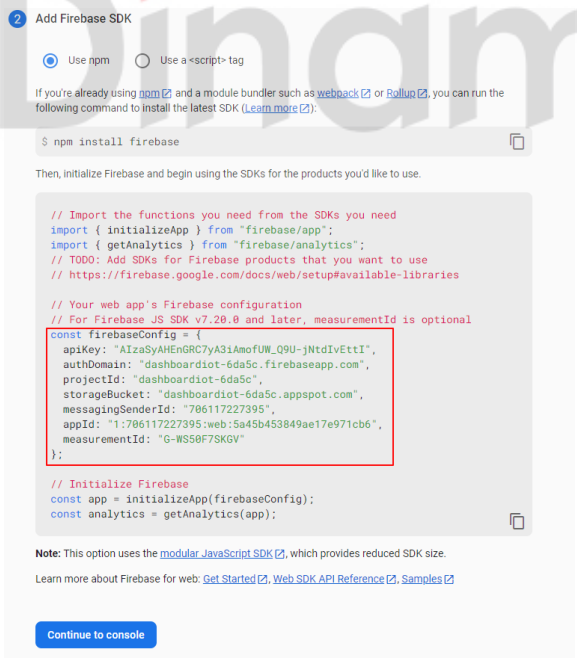
Gambar 3.3 Membuat *Project* Firebase

Setelah selesai membuat *project* baru akan diarahkan ke halaman *Dashboard* Firebase, pada halaman tersebut terdapat tiga tombol untuk membuat aplikasi Firebase pada tiga *platform* yang berbeda, yaitu IOS, Android dan Web. Selanjutnya, klik tombol Web yang berada di samping tombol Android. Pada kolom App *nickname* masukan nama aplikasi, lalu klik tombol *Register* App untuk lanjut ke tahap berikutnya sebagaimana tampak pada Gambar 3.4.



Gambar 3.4 Menambahkan Firebase Ke Aplikasi

Langkah selanjutnya *copy* variabel dengan nama firebaseConfig seperti pada Gambar 3.5 agar aplikasi dapat terhubung dengan Firebase. Setelah itu buat *file* dengan nama config.js dan pastikan pastikan *Library* Firebase telah terinstal di dalam *project* React Native. Jika sudah terinstal import *function* initializeApp, getDatabase dan get Auth dari *library* Firebase dan masukan variabel firebaseConfig yang telah di-*copy* pada tahap sebelumnya ke dalam file config.js



Gambar 3.5 Menambahkan Firebase ke aplikasi

```
import {initializeApp} from 'firebase/app';
import {getDatabase} from 'firebase/database';
import {getAuth} from 'firebase/auth';

const firebaseConfig = {
  apiKey: AIzaSyAHEnGRC7yA3iAmofUW_Q9U-jNtdIvEttI,
  authDomain: 'aquasense-ceae3.firebaseapp.com',
  databaseURL:
    XXXXXXXXXXXXXXXXXX-rtdb.asia-southeast1.firebasedatabase.app',
  projectId: 'aquasense-ceae3',
  storageBucket: 'aquasense-ceae3.appspot.com',
  messagingSenderId: '419014169226',
  appId: '1:419014169226:web:5ac6b55151726cb2b99373',
  measurementId: 'G-FX262Q4JQB',
};

const app = initializeApp(firebaseConfig);
const auth = getAuth(app);
const db = getDatabase();
export {db, auth};
```

Gambar 3.6 Program config.js

Dengan menambahkan kode seperti pada Gambar 3.6, React Native sudah dapat terhubung dengan Firebase dan sudah bisa mengambil data yang ada. Untuk mengambil data yang ada pada *Realtime Database* buat file baru dengan nama fetchdata.js.

```
import { getDatabase, ref, set, onValue, limitToLast } from 'firebase/database';
import { getAuth } from 'firebase/auth';
import { db } from './config.js';
import React, { useEffect, useState } from 'react';

const Fetchdata = ({ onDataLoaded }) => {
    const [turbidity, setTurbidity] = useState('');
    const [ph, setPH] = useState('');
    const [tds, setTDS] = useState('');

    useEffect(() => {
        const data = ref(db, 'sensor1');
        onValue(data, snapshot => {
            const datasensor = snapshot.val();
            setTurbidity(datasensor.valueTurbidity);
            setPH(datasensor.valuePH);
            setTDS(datasensor.valueTDS);

            onDataLoaded({
                turbidity: datasensor.valueTurbidity,
                ph: datasensor.valuePH,
                tds: datasensor.valueTDS,
            });
        });
    }, []);
};

export default Fetchdata;
```

Gambar 3.7 Program fetchdata.js

Pada program sebagaimana terlihat pada Gambar 3.7 di atas, pastikan file config.js telah ter-import dengan benar. komponen Fetchdata dibuat dengan tujuan untuk mengambil data yang ada pada *Realtime Database* dengan memanfaatkan Hook dari React yaitu useEffect. Dengan menggunakan *function* useEffect data akan yang ada pada *Realtime Database* akan diambil setiap kali aplikasi merender komponen. Lalu setelah data pada *Realtime Database* diambil dan data tersebut akan disimpan ke dalam props onDataLoaded yang nantinya data tersebut dapat diakses oleh setiap komponen pada *project* React Native

3.4 Desain *Dashboard* IoT

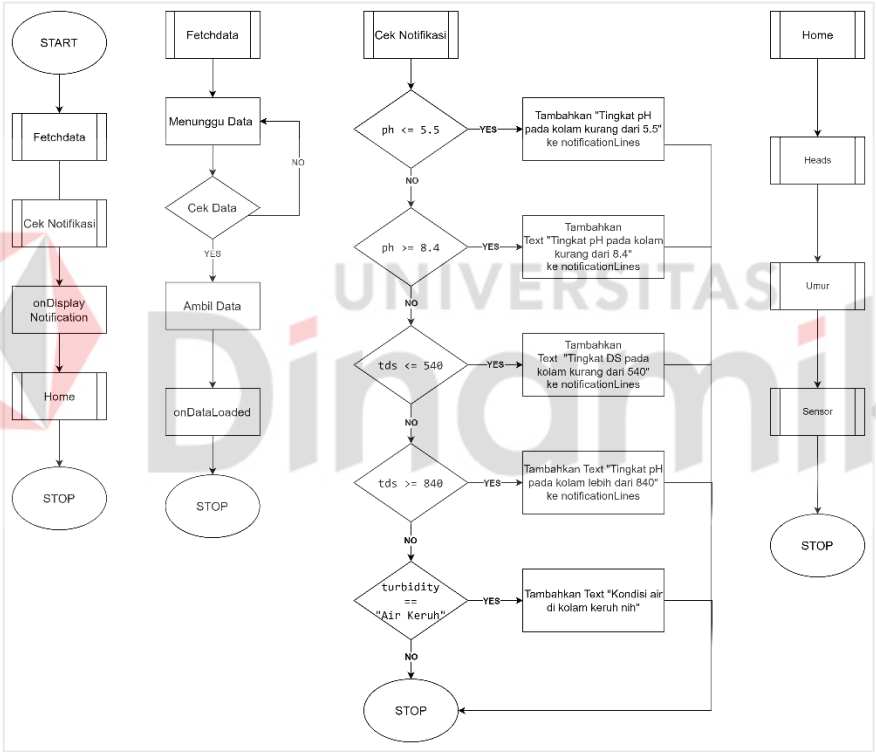
Gambar 3.8 merupakan desain dari *Dashboard* IoT, *Dashboard* IoT ini di desain dengan menggunakan Figma dengan menggunakan ukuran 900x1600. Desain aplikasi ini tersusun dari beberapa komponen, yang pertama ada komponen untuk mengukur umur tanaman dan umur ikan. Lalu ada komponen yang menampilkan data pembacaan sensor. Dan yang terakhir ada *line Chart* untuk memvisualisasikan data.

Gambar 3.8 Desain *Dashboard* IoT

Semua komponen yang ada pada *Dashboard* IoT dibuat dengan menggunakan *tools* bawaan dari figma, yaitu: komponen Head dibuat dengan menggunakan *text*, Komponen Umur dan Sensor dibuat menggunakan *rectangle tool* dan *text* dan komponen *Chart* dibuat dengan menggunakan *rectangle tool* dan *pen tool*

3.5 *Flowchart Dashboard* IoT

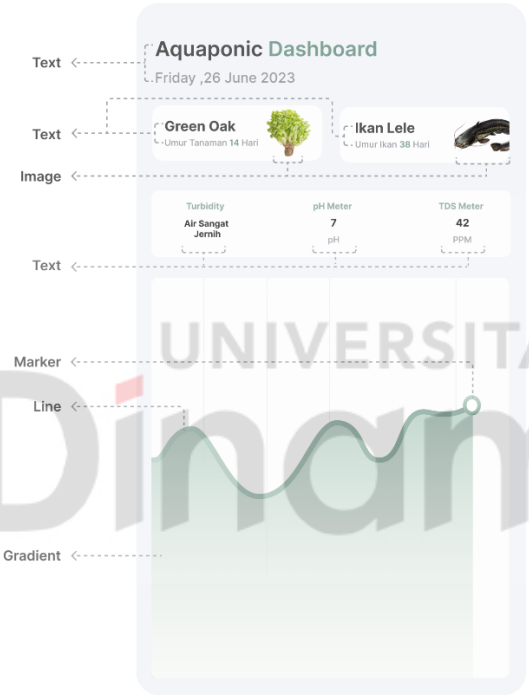
Gambar 3.9 merupakan *Flowchart Dashboard* IoT secara garis besar. Dalam *Flowchart* tersebut terdapat komponen komponen yang menyusun aplikasi *Dashboard* IoT. Berikut penjelasannya.



Gambar 3.9 *Flowchart Dashboard* IoT

Flowchart pada Gambar 3.9 merupakan *Flowchart* secara keseluruhan. *Flowchart* dimulai dari memanggil komponen Fetchdata, komponen ini bertugas untuk mengambil data yang ada pada *Realtime Database*. Setelah data berhasil di ambil. prop dengan nama onDataLoaded akan dieksekusi. Prop onDataLoaded ini berfungsi untuk mengirimkan data dari komponen Fetchdata ke komponen yang lain sehingga

komponen lain dapat mengakses data yang telah diambil dari *Realtime Database*. Selanjutnya, program akan melakukan pengecekan notifikasi. Jika ada data pembacaan sensor yang sesuai dengan kondisi yang telah ditentukan, maka, pesan notifikasi akan disiapkan dan ditambahkan ke notificationLine sesuai dengan kondisi, lalu *function* onDisplayNotification akan dieksekusi. *Function* onDisplayNotification inilah yang akan menampilkan notifikasi. Selanjutnya, komponen *Home* akan dipanggil, komponen *Home* ini berisikan 3 komponen, yaitu Head, Umur dan Sensor. Ketiga komponen ini akan ditampilkan jika komponen Home dipanggil.

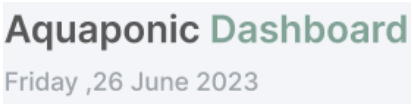


Gambar 3.10 Komponen React Native

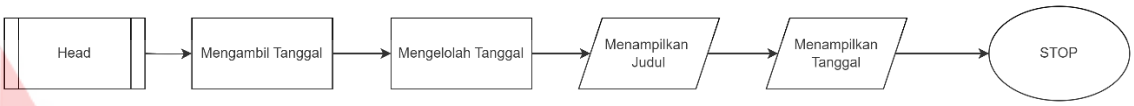
Gambar 3.10 merupakan komponen-komponen yang dipakai untuk menyusun komponen Head, Umur, sensor dan Chart. Dapat dilihat bahwa komponen Head, Umur, sensor dan Chart tersusun dari komponen seperti Text, Image, Line, Gradient, dan Marker

3.5.1 *Flowchart* Head

Komponen Head ini adalah komponen yang pertama kali dimunculkan di dalam *Dashboard* . Di dalam komponen ini terdapat tulisan dan tanggal seperti pada Gambar 3.11. Gambar 3.12 Merupakan *Flowchart* dari komponen Head. Dalam pembuatan komponen ini tidak diperlukan *library* tambahan. Berikut penjelasan *Flowchart* komponen *Head*.



Gambar 3.11 Komponen Head



Gambar 3.12 *Flowchart* Head

```
const tanggal = new Date();
```

Gambar 3.13 *Function* NewDate()

Flowchart dimulai dari Mengambil Tanggal. Untuk mengambil tanggal kita bisa menggunakan *function* newDate() seperti pada Gambar 3.13. Diblok selanjutnya, yaitu ”Mengelolah Tanggal”, tanggal akan di olah untuk mendapatkan format Hari, Bulan dan Tahun. Untuk mengolah tanggal tambahkan kode seperti pada Gambar 3.14.

```
    const options = {
      weekday: 'long',
      month: 'long',
      day: 'numeric',
      year: 'numeric',
    };
    const formattedDate =
tanggal.toLocaleDateString('en-US', options);
    const year = tanggal.getFullYear();
    const formattedDateTahun = `${formattedDate}
${year}`;
```

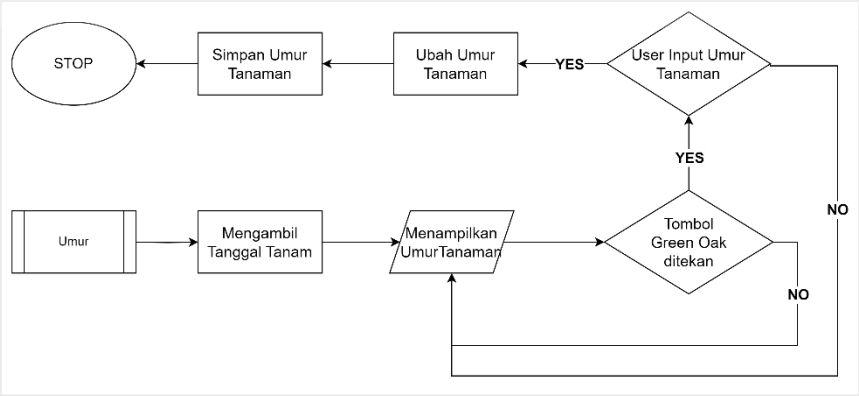
Gambar 3.14 Mengolah Tanggal

Setelah itu tanggal diolah untuk mendapatkan format tertentu, tulisan "Aquaponic *Dashboard* " dan Tanggal akan ditampilkan ke dalam *Dashboard* IoT.

3.5.2 *Flowchart* Umur

Komponen dengan nama Umur, merupakan komponen yang dapat menyimpan umur tanaman dan umur ikan sebagaimana tampak pada Gambar 3.15 . Untuk dapat menyimpan umur tanaman dan ikan yang berasal dari input *user*, digunakan *library* AsyncStorage sehingga umur tanaman dan ikan dapat disimpan secara lokal dan tidak membutuhkan koneksi internet untuk mengaksesnya. Untuk menginstall *library* AsyncStorage cukup dengan membuka terminal dan mengetikan perintah npm install @react-native-async-storage/async-storage. Jika *library* telah terinstal pastikan untuk meng-link *library* tersebut dengan perintah npx react-native link @react-native-async-storage/async-storage .

Gambar 3.15 Komponen Umur



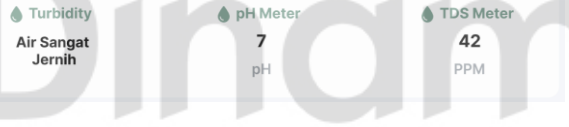
Gambar 3.16 *Flowchart* Umur Tanaman

Gambar 3.16 merupakan *Flowchart* dari komponen umur tanaman. *Flowchart* dimulai dari mengambil tanggal tanam. Pada blok mengambil tanggal tanam, *function* dengan nama calculateUmurTanaman() akan dijalankan. *Function* ini akan

mengambil tanggal saat ini. Lalu pada blok kedua umur tanaman akan ditampilkan.Selanjutnya jika tombol dengan nama “Green Oak” diklik akan muncul *pop up* untuk memasukkan umur tanaman yang di mana jika *user* memasukkan nama tanaman, proses akan dilanjutkan ke blok selanjutnya, yaitu mengubah umur tanaman dan menyimpan umur tanaman. Untuk komponen umur ikan langkah nya sama seperti umur tanaman

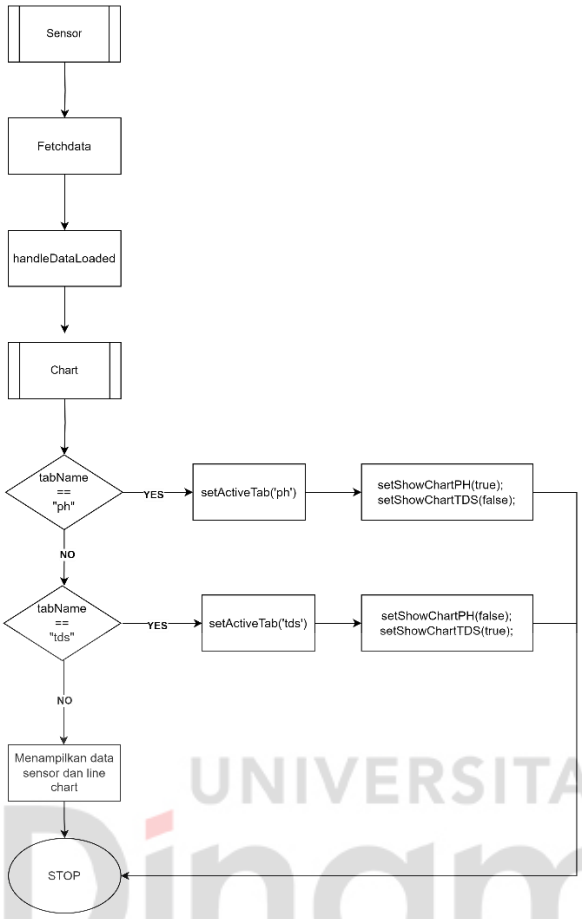
3.5.3 *Flowchart* Sensor

Gambar 3.17 merupakan komponen dengan nama “Sensor”. Komponen ini berfungsi untuk menampilkan data yang diambil dari *Realtime Database* dan berfungsi sebagai tombol untuk membuka Chart. Komponen “Sensor” merupakan komponen yang sangat penting karena komponen ini yang menampilkan data yang dikirimkan oleh kontroler. Untuk mengetahui alur dari program komponen sensor, silahkan lihat *Flowchart* Gambar 3.17



Gambar 3.17 Komponen Sensor

Flowchart pada Gambar 3.18 dimulai dengan memanggil komponen Fetchdata. Komponen ini dipanggil agar komponen Sensor dapat mengakses data yang ada di dalam komponen Fetchdata. untuk memanggil komponen Fetchdata, import terlebih dahulu komponen tersebut seperti pada Gambar. 3.19**.** Jika komponen Fetchdata telah dimport, kita dapat mengakses data yang ada didalam komponen tersebut dengan memanggil *function* handleDataLoaded. *Function* handeDataLoaded dipanggil untuk memasukan data dari komponen Fetchdata ke dalam state variabel SensorData (Gambar 3.20)



Gambar 3.18 *Flowchart* Komponen Sensor

```
import Fetchdata from './fetchdata.js';
```

Gambar 3. 19 Import Komponen Fetchdata

```
const [sensorData, setSensorData] = useState({});

const handleDataLoaded  = data => {
   setSensorData(data);
};
```

Gambar 3.20 *Function* handleDataLoaded

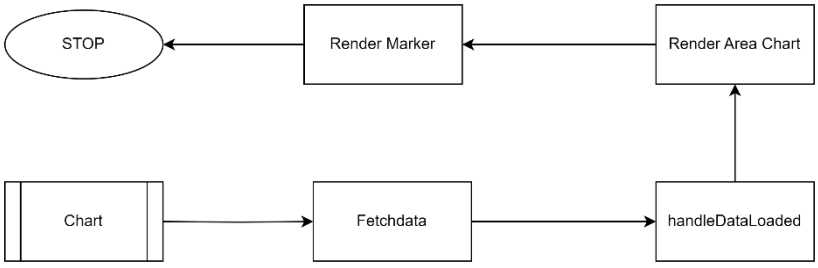
```
<Fetchdata onDataLoaded={handleDataLoaded}  />
```

Gambar 3.21 Memanggil Komponen Fetchdata

Dan terakhir tambahkan baris seperti Gambar 3.21 pada bagian return agar dapat mengakses data yang ada pada komponen Fetchdata. Setelah blok Fetchdata dan handleDataLoaded selesai dijalankan, komponen ChartPH akan ditampilkan. Komponen ChartPH ini akan ditampilkan pertama kali bersamaan dengan data dari komponen Fetchdata. Selanjutnya, ketika tombol "TDS" diklik oleh pengguna, *function* handleTabPress akan dipanggil dan menjalankan program yang ada di dalamnya, yaitu mengubah nilai state activeTab menjadi "tds", showChartPH menjadi false, dan showChartTDS menjadi true.

3.5.4 *Flowchart* Chart

Komponen Chart, pada komponen ini *terdapat line chart* yang gunanya untuk memvisualisasikan data hasil pembacaan sensor secara *Realtime* agar *user* dapat melihat *history* sebanyak 30 data. Proses / alur dari komponen Chart dapat dilihat pada Gambar 3.22. Untuk membuat *line Chart* diperlukan *library* react-native -svg-charts, pastikan *library* tersebut sudah terinstal pada *project* React Native, untuk menginstall *library* tersebut cukup ketikan perintah npm install react-native-svg-charts pada terminal. Setelah itu untuk dapat menampilkan data kita perlu mengakses data yang ada pada komponen Fetchdata, sama seperti komponen sebelumnya, kita perlu membuat *function* handleDataLoaded.



Gambar 3.22 *Flowchart* Chart

```
const [cdTDS, setcdTDS] = useState([]);
const [latestData, setLatestData] = useState(null);
const [sensorData, setSensorData] = useState({});

const handleDataLoaded = data => {
   setcdTDS(prevData => {
      const newData = [...prevData, data.tds];
      return newData.slice(-30); // simpan 30 data
   });
   setLatestData(data.tds); //update nilai
};
```

Gambar 3.23 *Function* handleDataLoaded

Program pada Gambar 2.23 berfungsi untuk mengakses data yang ada pada komponen Fetchdata lalu data tersebut disimpan ke dalam variabel state array cdTDS dan data tersebut akan selalu diperbarui melalui *function* setcdTDS. prevData memiliki nilai sebelumnya dari cdTDS. didalam *function* ini prevData memiliki nilai data.tds yang ditambahkan melalui spread operator sehingga array newData akan memiliki 30 data TDS. Dan untuk memotong agar hanya 30 data yang disimpan kita dapat menggunkaan *function* newData.slice(30). Jadi array newData hanya akan menampung 30 data saja.

```
const Gradient = () => (
   <LinearGradient id="gradient" x1="0%" y1="0%" x2="0%" y2="100%">
      <Stop offset="0%" stopColor={backgroundColorStart} stopOpacity={1} />
      <Stop offset="100%" stopColor={backgroundColorEnd} stopOpacity={1} />
   </LinearGradient>
);

const GradientLine = () => (
   <LinearGradient id="gradientLine" x1="0%" y1="0%" x2="0%" y2="100%">
      <Stop offset="0%" stopColor={lineColorStart} stopOpacity={1} />
      <Stop offset="100%" stopColor={lineColorEnd} stopOpacity={1} />
   </LinearGradient>
);

const Line = ({ line }) => (
   <Path d={line} stroke="url(#gradientLine)" strokeWidth={5} fill="none" />
);
```

Gambar 3.24 Komponen Gradient, GradientLine, Komponen Line

Baris program pada Gambar 3.24 adalah baris program yang digunakan untuk membuat garis dan *background* pada chart yang memiliki warna gradasi. Untuk gradasi

pada *background* dimulai dengan warna # BAD2C8 dan diakhiri dengan warna #E6EDE4. Sedangkan gradasi pada garis dimulai dari warna #81A395 dan diakhiri dengan warna # BAD2C8. Langkah selanjutnya kita perlu marker atau komponen berbentuk bulat untuk menandai ujung garis. Untuk membuat nya kita dapat menambahkan program pada Gambar 3.25. Potongan program pada Gambar 3.25 berfungsi untuk mengatur letak marker agar posisinya tepat berada diujung garis. Prop cx digunakan untuk menentukan koordinat secara *horizontal* dengan menggunakan *function* x(cdTDS.length -1). Sehingga data terakhir dalam array dapat dikonversikan menjadi koordinat x. sama seperti prop cx, prop cy digunakan untuk menentukan koordinat secara *vertical* dan menggunakan *function* y untuk konversi nilai terakhir dari cdTDS yaitu cdTDS[cdTDS.length-1] menjadi koordinat y.

```
const Marker = ({ x, y }) => (
  <Circle
    cx={x(cdTDS.length - 1)}
    cy={y(cdTDS[cdTDS.length - 1])}
    r={6}
    stroke="#fff"
    fill="#81A395"
  />
);
```

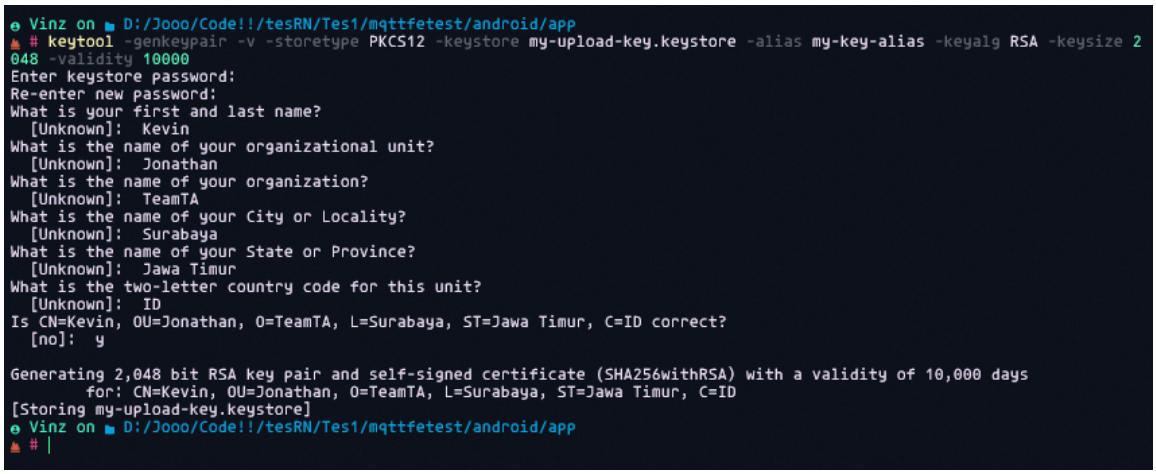
Gambar 3.25 Komponen Marker

3.6 Compile Aplikasi React Native Untuk Android

Agar aplikasi yang telah dibuat menggunakan React Native dapat berjalan di *Smartphone*. Langkah pertama kita perlu menjalankan terminal, pastikan lokasi berada pada folder *project* React Native lalu /android/app. setelah itu ketikan perintah seperti Gambar 3.26

```
keytool -genkeypair -v -storetype PKCS12 -keystore my-upload-key.keystore -alias my-key-alias -keyalg RSA -keysize 2048 -validity 10000
```

Gambar 3.26 Generate a private signing key



Gambar 3.27 Generate Keystore

Lalu masukan semua data yang diperlukan, saat membuat password pastikan lebih dari 6 karakter. Jika data sudah dimasukan dengan benar maka tampilan terminal akan seperti Gambar 3.27 dan akan ada file baru dengan nama my-upload-key.keystore pada file /android/app. Selanjut nya buka folder dengan nama gradle lalu buka file gradle.properties dan *copy* program pada Gambar 3.28 ke dalam gradle.properties

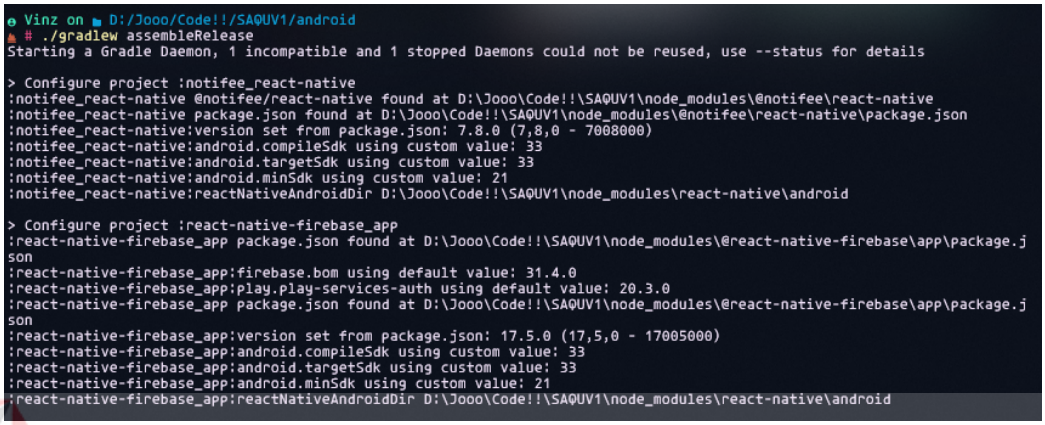
```
MYAPP_UPLOAD_STORE_FILE=my-upload-key.keystore
MYAPP_UPLOAD_KEY_ALIAS=my-key-alias
MYAPP_UPLOAD_STORE_PASSWORD=*****
MYAPP_UPLOAD_KEY_PASSWORD=*****
```

Gambar 3.28 Setting Gradle Variables

```
...
android {
    ...
    defaultConfig { ... }
    signingConfigs {
        release {
            if (project.hasProperty('MYAPP_UPLOAD_STORE_FILE')) {
                storeFile file(MYAPP_UPLOAD_STORE_FILE)
                storePassword MYAPP_UPLOAD_STORE_PASSWORD
                keyAlias MYAPP_UPLOAD_KEY_ALIAS
                keyPassword MYAPP_UPLOAD_KEY_PASSWORD
            }}}
    buildTypes {
        release {
            ...
            signingConfig signingConfigs.release
        }}}
...
```

Gambar 3.29 Menambahkan Signing Config

Selanjut nya *copy* pada Gambar 3.29 dan letakkan di dalam file build.gradle yang ada didalam folder /android/app. Buka kembali terminal lalu kembali ke folder android dengan mengetikan *command* cd android setelah itu ketikan perintah ./gradlew assembleRelease seperti pada Gambar 3.30 dan tunggu hingga proses compile selesai



Gambar 3.30 Compile Aplikasi

Jika sudah selesai, aplikasi dapat ditemukan di folder android/app/build/outputs/apk/release. Agar bisa berjalan di *smartphone*, silahkan pindahkan aplikasi ke dalam *smartphone* lalu di instal.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Pengujian Data Sensor

Pengujian Data Sensor ini dilakukan untuk membandingkan data hasil pembacaan sensor yang ditampilkan di *Serial Monitor* dengan data yang ada di aplikasi *Dashboard* IoT, tujuan nya agar data yang ditampilkan pada *Dashboard* IoT sesuai dengan pembacaan sensor

4.1.1 Prosedur Pengujian Data Sensor

Prosedur yang dilakukan untuk melakukan pengujian ini:

1. Menghubungkan kontroler dengan laptop dan jaringan internet.
2. Menampilkan hasil pembacaan sensor ke dalam *Serial Monitor*.
3. Memastikan kontroler mengirimkan data pembacaan sensor ke dalam *Realtime Database*.
4. Memastikan *Dashboard* IoT terhubung dengan internet.
5. Menjalankan aplikasi *Dashboard* IoT.

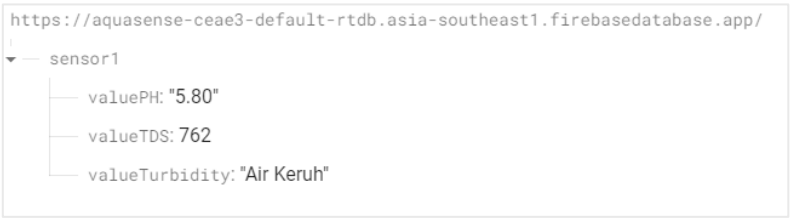
4.1.2 Hasil Pengujian Data Sensor

Pada tabel 4.1 pengujian dilakukan dengan cara membandingkan data hasil pembacaan ketiga sensor yang ditampilkan pada *Serial Monitor* dengan data yang ditampilkan pada aplikasi yang telah dibuat. Seperti pada Tabel 4.1 Data yang ditampilkan di *Dashboard* sesuai dengan data yang ada pada *Serial Monitor.*

Tabel 4. 1 Tabel Pengujian Data Sensor

No	*Serial Monitor*			Data Yang Ditampilkan Aplikasi			Keterangan
	TDS	pH	Turbidity	TDS	pH	Turbidity	
1	762	5.80	Air Keruh	762	5.80	Air Keruh	Sesuai
2	765	5.14	Air Keruh	765	5.14	Air Keruh	Sesuai
3	781	5.28	Air Keruh	781	5.28	Air Keruh	Sesuai
4	793	5.64	Air Keruh	793	5.64	Air Keruh	Sesuai

No	*Serial Monitor*			Data Yang Ditampilkan Aplikasi			Keterangan
	TDS	pH	Turbidity	TDS	pH	Turbidity	
5	797	5.42	Air Keruh	797	5.42	Air Keruh	Sesuai
6	761	5.92	Air Keruh	761	5.92	Air Keruh	Sesuai
7	781	5.50	Air Keruh	781	5.50	Air Keruh	Sesuai
8	800	5.98	Air Keruh	800	5.98	Air Keruh	Sesuai
9	766	5.91	Air Keruh	766	5.91	Air Keruh	Sesuai
10	782	5.41	Air Keruh	782	5.41	Air Keruh	Sesuai
11	776	5.94	Air Keruh	776	5.94	Air Keruh	Sesuai
12	779	5.47	Air Keruh	779	5.47	Air Keruh	Sesuai
13	751	5.48	Air Keruh	751	5.48	Air Keruh	Sesuai
14	758	5.30	Air Keruh	758	5.30	Air Keruh	Sesuai
15	784	5.85	Air Keruh	784	5.85	Air Keruh	Sesuai
16	795	5.43	Air Keruh	795	5.43	Air Keruh	Sesuai
17	797	5.33	Air Keruh	797	5.33	Air Keruh	Sesuai
18	770	5.06	Air Keruh	770	5.06	Air Keruh	Sesuai
19	755	5.42	Air Keruh	755	5.42	Air Keruh	Sesuai
20	797	5.87	Air Keruh	797	5.87	Air Keruh	Sesuai
21	788	5.80	Air Keruh	788	5.80	Air Keruh	Sesuai
22	768	5.40	Air Keruh	768	5.40	Air Keruh	Sesuai
23	775	5.38	Air Keruh	775	5.38	Air Keruh	Sesuai
24	761	5.11	Air Keruh	761	5.11	Air Keruh	Sesuai
25	785	5.29	Air Keruh	785	5.29	Air Keruh	Sesuai
26	773	5.73	Air Keruh	773	5.73	Air Keruh	Sesuai
27	767	5.62	Air Keruh	767	5.62	Air Keruh	Sesuai
28	761	5.86	Air Keruh	761	5.86	Air Keruh	Sesuai
29	751	5.51	Air Keruh	751	5.51	Air Keruh	Sesuai
30	779	5.45	Air Keruh	779	5.45	Air Keruh	Sesuai



Gambar 4. 1 Struktur Database

Gambar 4. 2 Tampilan Sensor

Gambar 4.1 merupakan struktur dari database. terdapat 3 *key* dengan nama valueTurbidity, valuePH, dan value TDS. Ketiga *key* ini memiliki *value* yang dapat berubah jika terdapat data baru yang dikirimkan oleh kontroler. Dan pada gambar 4.2 dapat dilihat bahwa semua hasil pembacaan sensor yang telah dimasukan ke dalam *Realtime Database* dapat diambil dan ditampilkan ke dalam *Dashboard* IoT dengan keakuratan mencapai 100%

4.2 Pengujian *Delay*

Pengujian ini dilakukan untuk mengetahui seberapa cepat aplikasi dapat menampilkan data yang dikirimkan

4.2.1 Prosedur Pengujian *Delay*

Dalam pengujian ini dilakukan tahapan- tahapan berikut:

1. Menghubungkan laptop dan dengan jaringan internet
2. Membuat program terpisah untuk mengirimkan data dari kontroler
3. Menambahkan program untuk menampilkan kecepatan pengambilan data ke dalam aplikasi *Dashboard* IoT dan menampilkan nya ke dalam console.
4. Membuka aplikasi *Dashboard* IoT

4.2.2 Hasil Pengujian *Delay*

Pada Tabel 4.2 merupakan pengujian untuk mengukur seberapa besar *delay* saat data dikirimkan dari kontroler dan ditampilkan ke aplikasi. Dari 30 kali percobaan didapatkan *delay* paling rendah 1 detik dan paling tinggi 2 detik. Pengujian dilakukan menggunakan internet dengan kecepatan 24.08 Mbps

Tabel 4. 2 Tabel Pengujian Delay

No	Data Terkirim	Waktu Pengiriman	Data Ditampilkan	Waktu Ditampilkan	*Delay*
1	1	05:23:24	1	05:23:25	1 detik
2	2	05:23:31	2	05:23:33	2 detik
3	3	05:23:38	3	05:23:40	2 detik
4	4	05:23:46	4	05:23:47	1 detik
5	5	05:23:52	5	05:23:53	2 detik
6	6	05:23:59	6	05:24:01	2 detik
7	7	05:24:06	7	05:24:08	2 detik
8	8	05:24:11	8	05:24:13	2 detik
9	9	05:24:18	9	05:24:20	2 detik
10	10	05:24:26	10	05:24:28	2 detik
11	11	05:24:33	11	05:24:34	1 detik
12	12	05:24:39	12	05:24:40	1 detik
13	13	05:24:45	13	05:24:46	1 detik
14	14	05:24:51	14	05:24:53	2 detik
15	15	05:24:58	15	05:24:59	1 detik
16	16	05:25:04	16	05:25:05	1 detik
17	17	05:25:11	17	05:25:12	1 detik
18	18	05:25:17	18	05:25:18	1 detik
19	19	05:25:23	19	05:25:24	1 detik
20	20	05:25:29	20	05:25:31	2 detik
21	21	05:25:36	21	05:25:38	2 detik
23	23	05:25:41	23	05:25:43	2 detik
24	24	05:25:48	24	05:25:50	2 detik
25	25	05:25:56	25	05:25:57	1 detik
26	26	05:26:02	26	05:26:04	2 detik
27	27	05:26:09	27	05:26:11	2 detik
28	28	05:26:16	28	05:26:17	1 detik
29	29	05:26:23	29	05:26:24	1 detik
30	30	05:26:29	30	05:26:31	2 detik
Rata-Rata					1.47 detik

4.3 Pengujian Notifikasi

Pengujian notifikasi ini dilakukan dengan tujuan agar *user* dapat menerima notifikasi jika kondisi kolam tidak sesuai dengan kondisi yang telah ditetapkan

4.3.1 Prosedur Pengujian Notifikasi

Pengujian ini dilakukan dengan langkah-langkah berikut:

1. Memastikan kontroler terhubung dengan jaringan internet.
2. Menampilkan hasil pembacaan sensor ke *Serial Monitor*
3. Menjalankan aplikasi *Dashboard* IoT
4. Membuat hasil pembacaan sensor tidak sesuai dengan kondisi yang sudah ditentukan agar *Dashboard* dapat menerima notifikasi

4.3.2 Hasil Pengujian Notifikasi Sensor Turbidity

Pengujian notifikasi dilakukan dengan cara memasukan probe sensor ke dalam air kotor sehingga sensor akan membaca bahwa air keruh dan akan mengirimkan notifikasi ke *Dashboard* IoT

Tabel 4. 3 Hasil Pengujian Notifikasi Sensor Turbidity

Pengujian	Pembacaan Awal Turbidity	Pembacaan Akhir Turbidity	Pesan Notifikasi	Keterangan
1	Sangat jernih	Air Keruh	Air Kolam Keruh	Notifikasi Diterima
2	Sangat jernih	Air Keruh	Air Kolam Keruh	Notifikasi Diterima
3	Sangat jernih	Air Keruh	Air Kolam Keruh	Notifikasi Diterima
4	Sangat jernih	Air Keruh	Air Kolam Keruh	Notifikasi Diterima
5	Sangat jernih	Air Keruh	Air Kolam Keruh	Notifikasi Diterima
6	Sangat jernih	Air Keruh	Air Kolam Keruh	Notifikasi Diterima
7	Sangat jernih	Air Keruh	Air Kolam Keruh	Notifikasi Diterima
8	Sangat jernih	Air Keruh	Air Kolam Keruh	Notifikasi Diterima
9	Sangat jernih	Air Keruh	Air Kolam Keruh	Notifikasi Diterima
10	Sangat jernih	Air Keruh	Air Kolam Keruh	Notifikasi Diterima

4.3.3 Hasil Pengujian Notifikasi Sensor TDS (PPM < 540)

Pada pengujian notifikasi pada sensor TDS ini dilakukan dengan cara mencelupkan probe sensor TDS ke dalam air yang sudah dicampur dengan cairan ABMix agar ppm bisa di atas 540. Lalu saat pengujian selanjutnya ujung probe di lap dengan menggunakan kain , agar hasil pembacaan sensor bisa dibawah 540. Dapat dilihat pada Tabel 4.4 ketika nilai ppm dibawah 540 maka *Dashboard* IoT menerima notifikasi bahwa PPM kurang dari 540

Tabel 4. 4 Hasil Pengujian Notifikasi Sensor TDS(PPM<540)

Pengujian	Pembacaan Awal TDS	Pembacaan Akhir TDS	Pesan Notifikasi	Keterangan
1	722	0	PPM Dibawah 540	Notifikasi Diterima
2	722	2	PPM Dibawah 540	Notifikasi Diterima
3	722	5	PPM Dibawah 540	Notifikasi Diterima
4	721	2	PPM Dibawah 540	Notifikasi Diterima
5	722	0	PPM Dibawah 540	Notifikasi Diterima
6	729	0	PPM Dibawah 540	Notifikasi Diterima
7	721	21	PPM Dibawah 540	Notifikasi Diterima
8	722	20	PPM Dibawah 540	Notifikasi Diterima
9	722	0	PPM Dibawah 540	Notifikasi Diterima
10	722	6	PPM Dibawah 540	Notifikasi Diterima

4.3.4 Hasil Pengujian Notifikasi Sensor TDS (PPM > 840)

Sama seperti saat menguji notifikasi untuk hasil pembacaan sensor dibawah 540. Pengujian ini dilakukan dengan cara membuat hasil pembacaan sensor TDS tidak sesuai dengan kondisi yang telah ditentukan dengan cara, menyiapkan dua wadah dengan air yang nutrisi nya berbeda. Pada pengujian ini penulis menggunakan wadah berisi air dengan ppm normal yaitu 722-734 dan air berisi cairan ABmix dengan ppm di atas 840. Dapat dilihat pada tabel 4.5 di atas ketika ppm di atas 840 maka akan notifikasi akan diterima oleh *Dashboard* IoT dan ditampilkan ke *user*

Tabel 4. 5 Hasil Pengujian Notifikasi Sensor TDS (PPM > 840)

Pengujian	Pembacaan Awal TDS	Pembacaan Akhir TDS	Pesan Notifikasi	Keterangan
1	729	1027	PPM Di atas 840	Notifikasi Diterima
2	721	1027	PPM Di atas 840	Notifikasi Diterima
3	722	1026	PPM Di atas 840	Notifikasi Diterima
4	722	1027	PPM Di atas 840	Notifikasi Diterima
5	722	1027	PPM Di atas 840	Notifikasi Diterima
6	722	1023	PPM Di atas 840	Notifikasi Diterima
7	722	1020	PPM Di atas 840	Notifikasi Diterima
8	722	1026	PPM Di atas 840	Notifikasi Diterima
9	724	1027	PPM Di atas 840	Notifikasi Diterima
10	722	1027	PPM Di atas 840	Notifikasi Diterima

4.3.5 Hasil Pengujian Notifikasi Sensor pH (pH < 6)

Untuk pengujian notifikasi pH, kurang lebih sama seperti dengan pengujian TDS, untuk mendapatkan pH dibawah 6, digunakan *BufferSolution* dengan tingkat pH 4.01. Dapat dilihat pada Tabel 4.6 hasil pembacaan sensor adalah 3.95 pH lalu *Dashboard* menerima notifikasi bahwa nilai pH dibawah 6.

Tabel 4. 6 Hasil Pengujian Notifikasi Sensor pH (pH < 6)

Pengujian	Pembacaan Awal ph	Pembacaan Akhir pH	Pesan Notifikasi	Keterangan
1	7.52	3.97	pH Dibawah 6	Notifikasi Diterima
2	7.52	3.97	pH Dibawah 6	Notifikasi Diterima
3	7.52	3.97	pH Dibawah 6	Notifikasi Diterima
4	7.53	3.97	pH Dibawah 6	Notifikasi Diterima
5	7.53	3.97	pH Dibawah 6	Notifikasi Diterima
6	7.52	3.97	pH Dibawah 6	Notifikasi Diterima
7	7.52	3.97	pH Dibawah 6	Notifikasi Diterima
8	7.53	3.95	pH Dibawah 6	Notifikasi Diterima
9	7.53	3.95	pH Dibawah 6	Notifikasi Diterima
10	7.53	3.95	pH Dibawah 6	Notifikasi Diterima

4.3.6 Hasil Pengujian Notifikasi Sensor pH (pH > 7)

Pada pengujian notifikasi pH dengan nilai di atas 7 digunakan *Buffer Solution* dengan nilai pH 9,18. Dapat dilihat pada Tabel 4.7 bahwa hasil pembacaan sensor pH di atas 7 dan *Dashboard* IoT menerima notifikasi bahwa pH melebihi batas yang telah ditentukan yaitu 7.

Tabel 4. 7 Hasil Pengujian Notifikasi Sensor pH (pH > 7)

Pengujian	Pembacaan Awal ph	Pembacaan Akhir pH	Pesan Notifikasi	Keterangan
1	7.53	8.80	pH Di atas 7	Notifikasi Diterima
2	7.52	8.80	pH Di atas 7	Notifikasi Diterima
3	7.52	8.85	pH Di atas 7	Notifikasi Diterima
4	7.52	8.85	pH Di atas 7	Notifikasi Diterima
5	7.53	8.85	pH Di atas 7	Notifikasi Diterima
6	7.52	8.85	pH Di atas 7	Notifikasi Diterima
7	7.54	8.85	pH Di atas 7	Notifikasi Diterima
8	7.53	8.85	pH Di atas 7	Notifikasi Diterima
9	7.52	8.85	pH Di atas 7	Notifikasi Diterima
10	7.52	8.85	pH Di atas 7	Notifikasi Diterima

BAB V
PENUTUP

5.1 Kesimpulan

Dari hasil pengerjaan dan pengujian yang dilakukan oleh penulis, diperoleh beberapa kesimpulan sebagai berikut:

1. Aplikasi *Dashboard* IoT dapat mengambil dan menampilkan data hasil pembacaan sensor yang dikirim oleh kontroler dengan rata rata *delay* 1,43 detik dan data yang diambil sesuai dengan data yang dikirimkan oleh mikrokontroler.
2. *Dashboard* IoT membutuhkan *Realtime Database* untuk menerima data yang dikirim oleh mikrokontroler
3. Aplikasi dapat menampilkan notifikasi kepada *user* ketika ada hasil pembacaan sensor yang tidak sesuai dengan kondisi yang telah ditentukan
4. Terdapat *line Chart* yang dapat memvisualisasikan data hasil pembacaan sensor secara *Realtime*.

5.2 Saran

Agar aplikasi *Dashboard* IoT ini dapat berkembang menjadi lebih baik, berikut beberapa saran yang dapat dipertimbangkan:

1. Menambahkan indikator aksi agar *user* dapat mengetahui aksi apa yang sedang dijalankan di kolam akuaponik
2. Menambahkan fitur *widget* agar *user* tidak perlu membuka aplikasi untuk memonitoring kolamnya

DAFTAR PUSTAKA

Android. (2020). *Google Play, Brand guidlines, developers android*. Diambil kembali dari developers android: https://developer.android.com/distribute/marketing-tools/brand-guidelines

Chris, K. (2021, Agustus 10). *10 VS Code Extensions to Increase Your Productivity*. Diambil kembali dari freecodecamp: https://www.freecodecamp.org/news/10-vscode-extensions-to-increase-productivity/

Danih, & Sugiyanto. (2021). Sistem Monitoring Berbasis Internet of Thing (IoT) Untuk Pengendalian Kualitas Air dan Pakan Ikan pada Budidaya sistem Akuaponik. *Journal of Students' Research in Computer Science (JSRCS), 2*(1), 89-98.

Dicoding. (2020, Desember 2). *Apa Itu JavaScript? Fungsi dan Contohnya*. Diambil kembali dari Dicoding: https://www.dicoding.com/blog/apa-itu-javascript-fungsi-dan-contohnya/

Figma. (2017, March 30). *Figma's new icon*. Diambil kembali dari Figma: https://www.figma.com/blog/figmas-new-icon/

Firebase. (t.thn.). *Builth With Firebase*. Diambil kembali dari Firebase: https://firebase.google.com/brand-guidelines

Google. (t.thn.). *Firebase Realtime Database*. Dipetik March 20, 2023, dari https://firebase.google.com/products/realtime-database?hl=id

Irsan, M. (2015). RANCANG BANGUN APLIKASI MOBILE NOTIFIKASI BERBASIS . *JustIN (Jurnal Sistem dan Teknologi Informasi)*.

Leonardo, R., Arwano, I., & Ratnawati, D. E. (2020). PEMANFAATAN TEKNOLOGI FIREBASE DALAM PENGEMBANGAN APLIKASI

PENGELOLAAN STOK BARANG BERBASIS MOBILE PADA RUMAH MAKAN NAKAMASE MALANG. *Jurnal Sistem Informasi, Teknologi Informasi, dan Edukasi Sistem Informasi, 1*(1), 1-11.

McDaniel, L. (2020, June 9). *Animations in ReactJS*. Diambil kembali dari blog.lawrencemcdan: https://blog.lawrencemcdaniel.com/animations-in-reactjs/

Miftah Farid Adiwisastra, S., Agung Baitul Hikmah, S., & Ai ilah Warnilah, S. (2019). *Dasar Pemograman Web.* Purwodadi-Grobogan, Jawa Tengah: CV. SARNU UNTUNG.

Muhyidin, M. A., Sulhan, M. A., & Sevtiana, A. (2020). PERANCANGAN UI/UX APLIKASI MY CIC LAYANAN. *JURNAL DIGIT*, 211.

Permana, A. Y., & P. R. (2019). PERANCANGAN SISTEM INFORMASI PENJUALAN PERUMAHAN MENGUNAKAN METODE . *Jurnal SIGMA*, 155.

Salamah, I., Taqwa, A., & Wibowo, A. T. (2020). RANCANG BANGUN SISTEM KEAMANAN SEPEDA MOTOR BERBASIS IOT. *JURNAL FASILKOM, 10*(2), 103-112.

Setiawan, R. (2021). *Dicoding.com*. Dipetik March 21, 2022, dari https://www.dicoding.com/blog/apa-itu-react-native/

Sinaga, G. A. (2022). *Information Technology Certification Center*. Dipetik Maret 20, 2023, dari https://itcc.itpln.ac.id/firebase-cloud-messaging/

Widodo, A., Alfia, R., Nurhayati, & Kholis, N. (2021). Sistem Monitoring Kualitas Air Pada Sistem Akuaponik Berbasis IoT. *JURNAL TEKNIK ELEKTRO, 10*(3).