The code runs through ***main.cpp*** located the **Hardware** folder the project requires that the ***credentials.h*** must be filled, I have purposely removed these for security reasons, but I can provide the credentials if it is necessary.

I did not record presentation with alarm as I filmed late in the evening.

Note in the presentation the history is shown in interval of a minute this is just for demo purpose the code has since been changed to show average temperature hour for hour.

Case: Potato field monitoring system made with Flutter and Firebase.

Parts:

- 1 x ESP32
- 1 x DHT11
- 1 x Passive buzzer

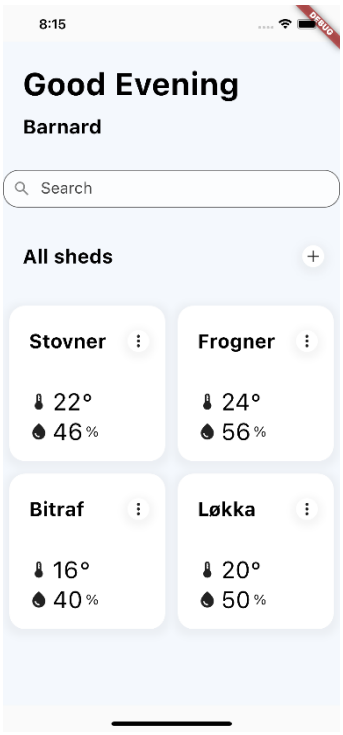
The device

The device is very basic consisting of just 3 parts, but I as I will explain in the code it is easy to extend with other peripheral.

Choosing the cloud service

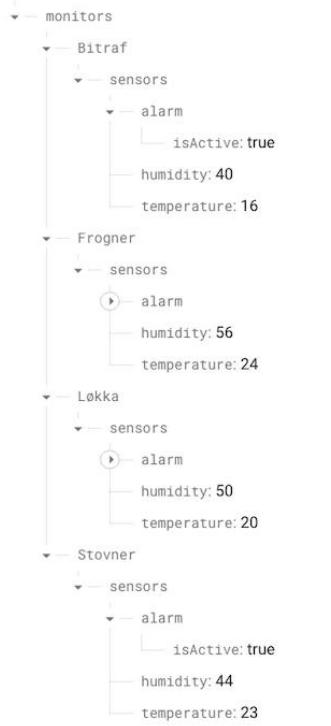
This was undoubtedly the hardest part, I wanted to challenge my self by not using any pre-made services like Blynk or Arduino cloud. First, I started with AWS's Amplify which offers MQTT communications, hooked it up with React Native using the Pub Sub library and ESP32 using the AWS IoT core library. This worked fine, but since my UI/UX skills are not that great the application did not really look the way I wished, obviously this could be fixed by using several UI libraries, but that was not the road I wanted to take.

After some searching for a cross platform alternative I came across Flutter, although I had heard of Flutter before I never gave it much attention until now. I spend a couple of days reading the documentations and from then it was very easy to build an eye pleasing application just with the very basics.

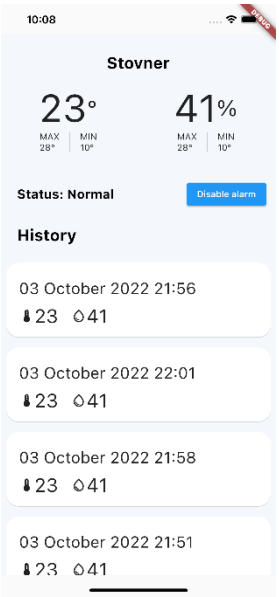


*The dashboard of the application note that **Stovner** is the only place that receives realtime data, **Frogner**, **Bitraf** and **Løkka** are just placeholders to fill the page. The **add**, **search,** and **edit** buttons are just for decorative purpose and might be implemented later. The application is cross platform, but I chose to base it for mobile as this is probably the most common use case.*

From there I needed a way to connect the frontend with the ESP32 which brings me to Firebase. Since both Flutter and Firebase are made by Google it was pretty much effortless to make them work together as it was a just a case of installing the necessary libraries. Firebase offers two databases the Real-time database and Firestore. I chose to use the Realtime database as I read that it is slightly faster than Firestore and since speed is essential for this project it was a no brainer.



*The realtime database the picture does not have the **history** property which logs the average temperature from the previous hour because the image was taken right after booting the device*



*Shows the history of temperatures and humidity, the **status** is hardcoded for now, but ideally this should come from the ESP32 via the database the **disable alarm** button works, if the alarm is on user can disable it after 10 min it will be enabled again of course given that the temperature is still below zero. This should also be conditionally rendered.*

The max and min fields are also hardcoded, but this is an easy fix for the future.

The code:

I only find it necessary to talk about the cpp code as this is most important,

The code consists of 4 classes

WiFimanager

- Sets up the wifi and that's pretty much it..

DBManager

- Sets up the database with the required credentials
- Sends and retrieves data using methods like ***update*** and ***getHistory***

SensorMonitor

- Sets up the DHT11 and active buzzer
- Reads and writes to the sensors

TimeKeeper

- Sets up local time using the nttp server.
- Used to get the time stamp to save average temperatures to the database.

Known errors:

One major flaw with the project is that the security rules in the database are public, this is obviously very bad, but since this is just a small testing prototype, I decided to keep it that way for simplicity's sake. This could be fixed by adding authentication using Firebase auth.

The time in history should be an hour less than what it is since I want to show the average temperature **from** the previous hour and the list should be sorted in chronological order.