Umsetzung

Umsetzung des Frontends in Flutter:

HomeScreen:

Der Home-Screen basiert auf einem Scroll-View, welches den Vorteil bietet ein einheitliches Design auf jeder Art von Bildschrimgröße zu gewährleisten, ohne die Lesbarkeit zu beeinträchtigen. Ein ScrollView in Flutter ist ein Widget, das verwendet wird, um eine Liste von Elementen anzuzeigen, die größer sind als der verfügbare Bildschirm. Es ermöglicht dem Benutzer, durch die Liste zu scrollen, um alle Elemente anzuzeigen. Es gibt verschiedene Arten von ScrollView in Flutter, darunter ScrollView, ListView, GridView und CustomScrollView. Das ScrollView enthält generisch selbst erstellte Objekte names DataCard.

Das DataCard Widget in Flutter ist ein vorgefertigtes Material Design-Widget, das zur Darstellung von Informationen in einem Kartenformat verwendet wird.

Es besteht aus einer rechteckigen Box mit abgerundeten Ecken, die in der Regel eine Hintergrundfarbe, einen Titel, eine Beschreibung und optional eine Aktion oder einen Button enthält.

Das DataCard Widget enthält verschiedene Eigenschaften, die angepasst werden können, um das Erscheinungsbild und das Verhalten der Karte zu steuern. Dazu gehören Eigenschaften wie Hintergrundfarbe, Ränder, Schatten, Größe, Padding und Ausrichtung.

DataCards verfügen über die Funktion „onPressed“ in der man angeben kann was geschieht, wenn man eine der 5 DataCards drückt. Wenn man eine DataCard die den Namen „Temperatur“, „Ph-Wert“ oder „Trübung“ drückt wird man an eine weitere Page der App weiter geleitet in der man Statistiken zu dem jeweiligen Wert erhält. Wenn man die DataCard mit dem Namen „Alarm“ drückt wird der Alarm den man erhält wenn verdächtige Bewegungen des Gyroskopssensors wahrgenommen werden aktiviert oder deaktiviert.

Die DataCards auf dem HomeScreen enthalten ansonsten den zuletzt gemessenen Wert, der durch einen WebSocket vom Backend bereitgestellt wird und stellen diesen dar. Die DataCards sind generisch erstellbar mit Übergabewerten, was eine einfache Vergrößerung des Funktionsumfangs des HomeScreens ermöglicht.

Umsetzung Backend:

Die generelle Funktionsweise des Backends basiert auf zwei TTGO-ESP32-LoRa Mikroprozessoren. Einer dieser Mikrochips(Sender) hat 4 verbaute Sensoren darunter:

* Temperatursensor
* Ph-Sensor
* NTU-Trübungssensor
* Beschleunigungssensor

Die Sensoren wurden über die vom ESP32 zur Verfügung gestellten Pins angelötet. Der Trübungssensor wurde bei einem ADC-Pin (Analog-Digital-Converter) angelötet, ebenso wie der Ph-Sensor, der Temperatursensor ist bei einem GPIO (General Purpose Input/Output) angebracht und der Beschleunigungssensor bei einem SDA(serial data) und dem SCL(serial clock) Anschluss. Der ESP32(Sender) wird alle 3 Stunden aus seinem Sleep-Zustand aufgeweckt und nimmt eine Messung der Daten vor. Er befindet sich während er inaktiv ist im Sleep damit er noch Stromsparender agiert da er nicht an einer Steckdose angebracht ist sondern mit einer externen Batterie betrieben wird.

Diese Daten werden direkt nach der Messung an den anderen ESP32(Reciever) gesendet. Die Daten werden nicht auf dem im Wasser schwimmenden Sender gespeichert da dieser äußerst stromsparend vorgehen soll. Die Messwerte werden mittels LoRa-Funk-System welches lokal an dem ESP32 verbaut ist an den Reciever gesendet und dort bei Erhalt mit einem Epoch-Timestamp versehen und in das SPIFF-File-System gespeichert.

Das äußerst kleine Speichersystem ermöglicht eine Speicherung der Daten in einem Intervall von allen 3 Stunden über 1 Jahr durchgehend.

Die Daten werden mittels Web-Sockets(für die Live-Daten-Aktualisierung) bereitgestellt und mit einem http-Request der alle Daten an das Frontend übergibt falls das Fenster mit den Zeit-Diagrammen aufgerufen wird um sie in die jeweiligen Diagramme einzufügen. Die dazu benötigten Libraries um dies alles in C umzusetzen sind unter „Technologien| Software“ zu finden.

Die Daten die für eine Routine-Übertragung alle 3 Stunden nötig sind, lauten:

* Datum
* Temperatur
* Ph-Wert
* NTU-Wert

Die Daten des Beschleunigungssensors werden nicht übertragen, um Speicherplatz zu sparen, die Aktualisierungsgeschwindigkeit zu erhöhen und eine Echt-Zeit-Benachrichtigung zu ermöglichen, sondern der Sender sendet eine extra Benachrichtigung falls der Beschleunigungssensor Werte erreicht hat die auf verdächtige Aktivitäten im Wasser hinweisen.

Der (Reciever) ESP32 ist mit dem W-Lan verbunden und permanent eingeschaltet um die Daten 24 Stunden 7 Tage in der Woche zu erhalten zu verarbeiten und für das Frontend für eine http-Request bereitzustellen.

Funktion von LoRa-Funk in diesem Projekt:

Die Kommunikationstechnologie, die auf der Basis von Spread-Spectrum-Techniken entwickelt wurde ermöglicht eine drahtlose Kommunikation über große Entfernungen bei niedriger Leistungsaufnahme. In deinem Fall werden die Daten von drei Sensoren (Temperatursensor, pH-Sensor und NTU-Trübungssensor) mithilfe eines ESP32-Mikrocontrollers und eines LoRa-Transceivers an einen zweiten ESP32, der als Empfänger fungiert, übertragen.

Hier ist das Grundkonzept, wie LoRa funktioniert und die Daten überträgt:

Sensoren erfassen Daten: Die drei Sensoren (Temperatur, pH und NTU-Trübung) erfassen Umgebungsdaten und übermitteln diese an den ESP32-Mikrocontroller.

Datenverarbeitung: Der ESP32-Mikrocontroller verarbeitet die empfangenen Daten, konvertiert sie in ein geeignetes Format und erstellt ein Datenpaket. Dieses Paket enthält die Sensordaten.

LoRa-Modulation: Der LoRa-Transceiver, der am ESP32 angeschlossen ist, wandelt das Datenpaket in ein Funksignal um, das für die Übertragung über das LoRa-Protokoll geeignet ist. LoRa verwendet eine Chirp Spread Spectrum (CSS)-Technologie, bei der die Frequenz des Funksignals kontinuierlich über eine bestimmte Bandbreite variiert. Diese Technik erhöht die Störfestigkeit und ermöglicht eine effiziente Nutzung des Funkspektrums.

Datenübertragung: Das modulierte Funksignal wird über die LoRa-Antenne ausgesendet und kann über große Entfernungen übertragen werden. LoRa ist besonders für Anwendungen mit geringem Energieverbrauch und geringer Datenrate geeignet.

Empfang des Funksignals: Der zweite ESP32, der als Empfänger fungiert, ist ebenfalls mit einem LoRa-Transceiver und einer Antenne ausgestattet. Dieser empfängt das Funksignal, das von der Senderantenne ausgesendet wurde.

Demodulation und Datenextraktion: Der LoRa-Transceiver am Empfänger demoduliert das empfangene Funksignal und extrahiert das ursprüngliche Datenpaket. Dabei wird die Spread-Spectrum-Technik rückgängig gemacht, um die übertragenen Informationen zurückzugewinnen.

Datenverarbeitung: Der empfangende ESP32-Mikrocontroller verarbeitet das extrahierte Datenpaket, um die Daten der einzelnen Sensoren (Temperatur, pH und NTU-Trübung) zu erhalten. Anschließend werden die Daten dann für die Zugriffe des Frontends vorbereitet.

Visual Studio Code (VSCode) ist eine plattformübergreifende Entwicklungsumgebung (IDE), die von Microsoft entwickelt wurde. Mit einer breiten Palette von Funktionen und einer hohen Anpassungsfähigkeit ermöglicht VSCode die Programmierung in verschiedenen Programmiersprachen und unterstützt Entwickler durch zahlreiche Erweiterungen wie in diesem Projekt verwendeten Flutter- und Espressif IDF-Erweiterung.

Visual Studio Code bietet Entwicklern eine einheitliche und benutzerfreundliche Schnittstelle, um Code in einer Vielzahl von Programmiersprachen zu schreiben und zu bearbeiten. Die IDE unterstützt Syntaxhervorhebung, Code-Navigation, Debugging und Versionskontrolle.

Eine der Stärken von Visual Studio Code ist das Erweiterungssystem, das es ermöglicht, den Funktionsumfang der IDE durch den Einsatz von Erweiterungen anzupassen und zu erweitern. Diese Erweiterungen werden von Microsoft, der Community oder Drittanbietern entwickelt und bieten zusätzliche Funktionen, die auf bestimmte Programmiersprachen, Frameworks oder Plattformen zugeschnitten sind.

Eine solche Erweiterung ist die Flutter-Extension, die die Entwicklung von plattformübergreifenden mobilen, Web- und Desktop-Anwendungen mit dem Flutter-Framework von Google ermöglicht. Die Flutter-Extension bietet Funktionen wie Autovervollständigung, Syntaxhervorhebung, Fehlererkennung und Integration von Flutter- und Dart-SDKs. Darüber hinaus unterstützt die Erweiterung das Hot-Reload-Feature von Flutter, das es ermöglicht, Änderungen am Code in Echtzeit anzuzeigen, ohne die Anwendung neu starten zu müssen.

Eine weitere nützliche Erweiterung ist die Espressif IDF-Extension, die die Entwicklung von Anwendungen für ESP32- und ESP-IDF-basierte Geräte erleichtert. Diese Erweiterung bietet Integration mit dem Espressif IoT Development Framework (IDF) und unterstützt Funktionen wie Codevervollständigung, Projektgenerierung, Debugging und Flashing von Firmware auf ESP32-Geräte. Dazu kommt die automatische Erkennung von angeschlossenen ESP-Geräten an den Computer die separat ausgewählt werden können.

Umsetzung in C:

Reciever:

In diesem Ausschnitt des Reciever-Codes wird lora\_recieve() gestartet. Dadurch wartet der Empfänger auf ein LoRa-Datenpacket. Wenn ein Datenpacket Empfangen wird, wird die „while-Schleife“ ausgeführt und die Packete in den Buffer gelesen. Danach werden die Daten in den File geschrieben. Dann werden alle Web-Server-Clients abgefragt und die neuen Daten an sie gesendet. Dies wird in der „for-Schleife“ für jeden einzelnen Client wiederholt. Wenn der Client einen Web-Socket ist wird ein anderer Code ausgeführt aber in beiden Fällen werden die Daten an den Client gesendet.

Im Laufe dieser Diplomarbeit wurden sehr viele Erkenntnisse über die Programmiersprachen Dart und C gemacht. Es wurde C im Rahmen der Mirkoprozessor-Programmierung kenngelernt und Einblicke in eine Hardware nahe Entwicklung gewonnen, die in der Fachrichtung-Informatik normalerweise nicht üblich sind, der Aufbau neuer Programmiersprachen kennengelernt und ein tiefergehendes Verständnis zur Entwicklung einer

Full-Stack-Application wurde entwickelt.

Ebenso wurde bewusst wie wichtig ein Arbeitsplan und Zeit-Management sind um nicht die Zeit aus den Augen zu verlieren und mögliche Probleme die während der Entwicklungszeit auftreten können einplanen zu können.