Technische Dokumentation

Gruppe 1

Medieninformatik Projekt
Informationsvisualisierung WS 20/21

Björn Glawe
Taeeun Kim
Lanea Lilienthal

Inhaltsverzeichnis

1. Entwicklungsumgebung

1.1 Betriebssystem und Software

- Windows 10
- Visual Studio Code (Version: 1.52.1)
- VSCode Extension: Arduino from Microsoft (Version: 0.3.4)
- Arduino (1.8.13)

1.2 Arduino-Bibliotheken

verwendete Arduino-Bibliothek	**Version**
DHT sensor library	1.4.0
MQUnifiedsensor	2.0.1
Adafruit_MPU6050	2.0.3
OneWire	2.3.4
NeoPixelBus	2.6.0
TFT_eSPI	2.3.4
SPI	1.0.0

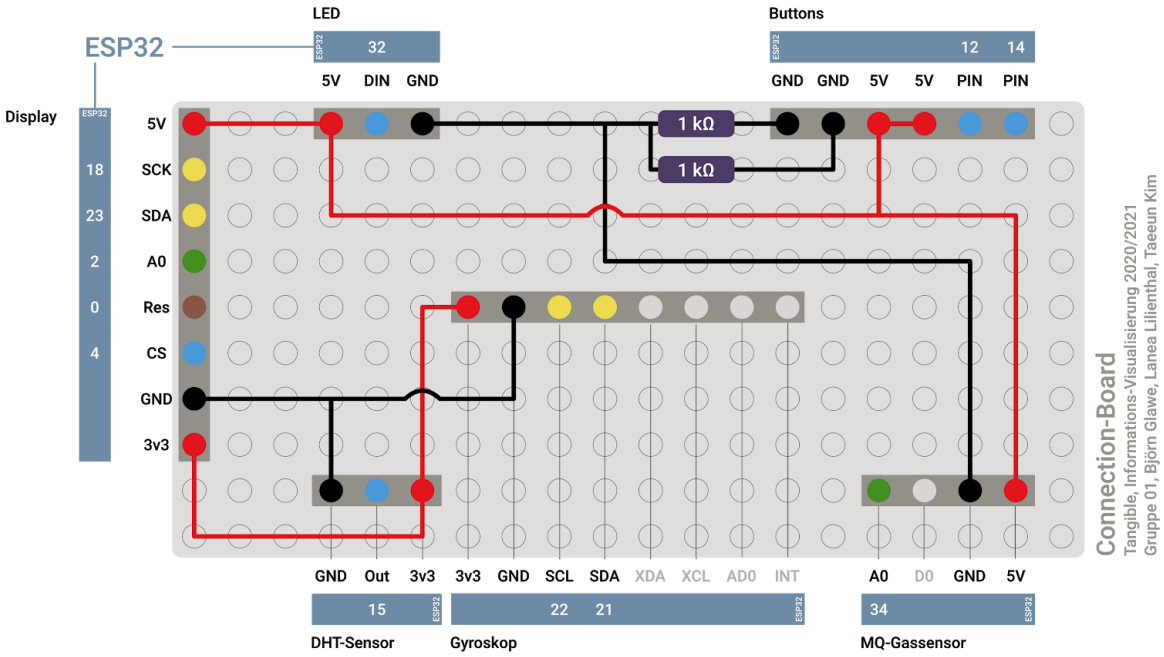
2. Hardware

2.1 Hardware-Komponenten

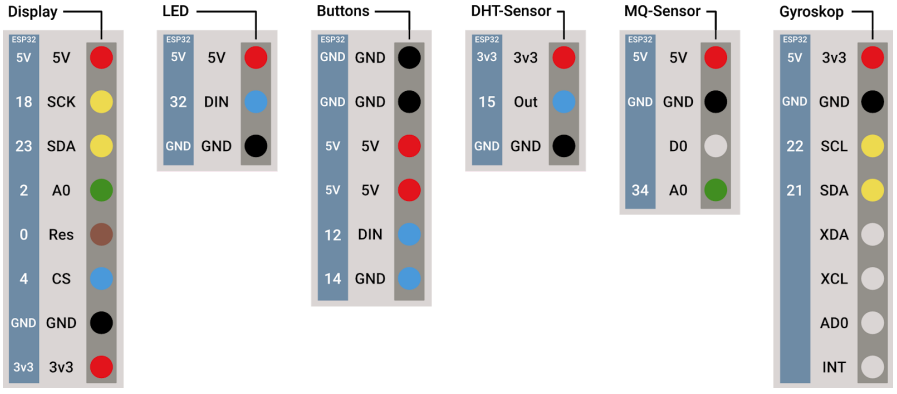
Bauteil	technische Bezeichnung	Funktion
ESP32	AZDelivery ESP32 Dev Kit C V4 NodeMCU WLAN WiFi Development Board (Link)	Mikrocontroller zur Steuerung aller Funktionen
Display	SPI TFT Display 1,8 Zoll mit 128 x 160 Pixeln (mit Kartenslot) (Link)	visuelle Wiedergabe von Informationen
LED	8 WS2812B LEDs adressierbare Streifen mit 5050 SMD LEDs	visuelle Wiedergabe von Informationen durch Farbdarstellung
Buttons	Drucktastenschalter	Eingaben durch Drücken des Tasters
DHT-Sensor	DHT22 AM2302 Temperatursensor und Luftfeuchtigkeitssensor mit Platine (Link)	Messung von Temperatur und Luftfeuchtigkeit
MQ-Sensor	MQ-135 Gas Sensor Luftqualität Modul (Link)	Messung von Luftqualität, misst Anteil an Luftpartikeln
Gyroskop	GY-521 MPU-6050 3-Achsen-Gyroskop und Beschleunigungssensor (Link)	Messung von Bewegung, hier Rotation

2.2 Hardware-Aufbau

Folgende Grafik zeigt die Verbindungen auf der Platine zu den einzelnen Bauteilen. Dabei dient die Platine als ein kompaktes Mittelstück, an dem alle Bauteile miteinander vereint werden. Das Display wird auf der Rückseite direkt verbunden, alle anderen Sensoren sind auf der Vorderseite direkt oder über Kabel verbunden. Alle notwendigen Verbindungen gehen von der Plantine aus zum ESP32. Durch die Platine werden vor allem 5V, 3v3 und GND nur jeweils einmal mit dem ESP32 verbunden. Dies ermöglicht eine relativ kompakte Bauweise im Tangible.



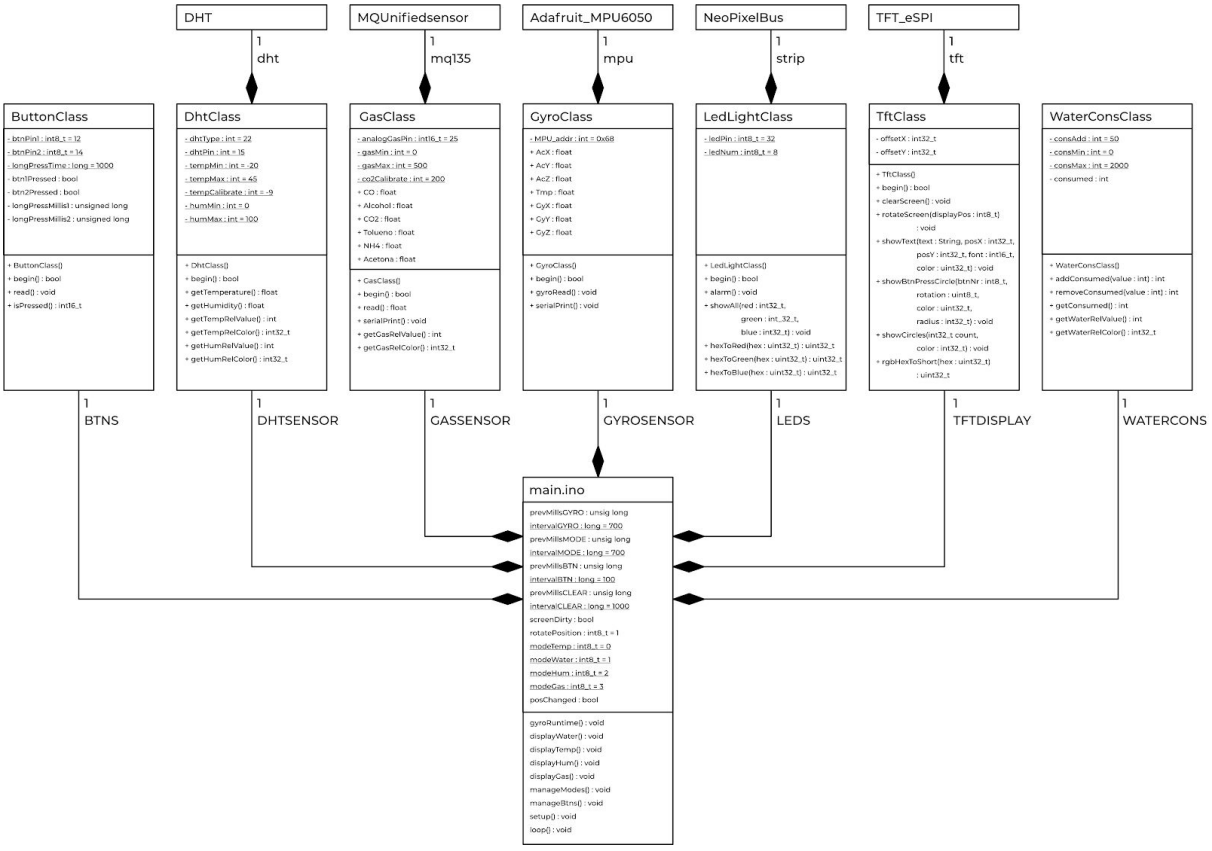
In der folgenden Abbildung finden sich die Verbindungen von der jeweiligen Komponente zum direkten Pin-Anschlüsse am ESP32. Auf der linken Seite befinden sich jeweils die ESP32-Pins und auf der rechten Seite die Komponenten-Pins. Graue Kreise (Pins) werden nicht angeschlossen.



3. Software

3.1 Vereinfachtes Klassendiagramm

Zentral ist die **`main.ino`**-Datei. Alle Funktionen werden hier initialisiert, aufgerufen und verknüpft. Zu jedem Modul, bzw. für den Wasserkonsum zusätzlich, existiert eine eigene Klasse, in der für das Projekt notwendige Funktionen implementiert wurden. Diese können jeweils von der **`main.ino`** aufgerufen und verwendet werden.



3.2 Kompilierte Binärdateien und Upload

Unter `./build` sind alle kompilierten Dateien und Sourcecode nach dem letzten build-Vorgang zu finden, inklusive Bibliotheken.
In der Konfigurationsdatei `.vscode/arduino.json` werden durch die Angabe von `"output": "build"` alle Dateien die im Projekt zur Kompilierung, sowie auch die kompilierten Binärdateien, im Ordner `./build` abgelegt.
Für den Upload auf den ESP32 werden folgende Dateien benötigt:

- `build\button.cpp.bin`
- `build\button.cpp.partitions.bin`

  Die Bezeichnung *button* innerhalb des Dateinamen spielt dabei keine Rolle.

  Für den Upload der Binärdateien lautet der Konsolenbefehl wie folgt:

farbliche Codierung:
`ESP-Uploader-Files` (kommen mit der ESP32-Board-Packages)
`Controller-Configurations` (spezifische Einstellungen für den Upload)
`kompilierte Projektdateien` (zuvor beschriebene Dateien unter `./build`)

schematischer Upload-Konsolen-Befehl ([...] kennzeichnet den Systempfad zur Datei):

```
[...]/esptool.exe --chip esp32 --port COM3 --baud 921600 --before default_reset
--after hard_reset write_flash -z --flash_mode dio --flash_freq 80m --flash_size
detect 0xe000 [...]/boot_app0.bin 0x1000 [...]/bootloader_qio_80m.bin 0x10000
[...]\button.cpp.bin 0x8000 [...]\button.cpp.partitions.bin
```

Beispiel von der eigenen Entwicklungsumgebung:

```
C:\Users\rumpe\AppData\Local\Arduino15\packages\esp32\tools\esptool_py\2.6.1/esptool.exe
--chip esp32 --port COM3 --baud 921600 --before default_reset --after hard_reset
write_flash -z --flash_mode dio --flash_freq 80m --flash_size detect 0xe000
C:\Users\rumpe\AppData\Local\Arduino15\packages\esp32\hardware\esp32\1.0.4/tools/partition
s/boot_app0.bin 0x1000
C:\Users\rumpe\AppData\Local\Arduino15\packages\esp32\hardware\esp32\1.0.4/tools/sdk/bin/b
ootloader_qio_80m.bin 0x10000
C:\Users\rumpe\CloudStation\Drive\UNI_5\InfoVis\infovis-2021-gruppe-1\build\button.cpp.bin
0x8000
C:\Users\rumpe\CloudStation\Drive\UNI_5\InfoVis\infovis-2021-gruppe-1\build\button.cpp.par
titions.bin
```

Um den Konsolenbefehl auf dem eigenen System zu finden, geht man wie folgt vor.
In der Arduino IDE unter `File>Preferences` die Option `"Show verbose output during" > "upload"` den Haken setzen (true).
Daraufhin einen Upload mit Arduino IDE auf einen ESP32 starten und in der letzten Zeile vom Arduino Output, bevor der Upload startet, ist der Uploadbefehl mit allen Pfadangaben in der eigenen eingerichteten Entwicklungsumgebung zu finden.
Diesen Befehl kopieren und entsprechende Pfade (in dem Beispiel `grün` markiert) durch die kompilierten Binärdatei-Pfade ersetzen.

4. Bedienungsanleitung

Das Tangible hat vier grundlegende Funktionen, die für den Nutzer relevant sind:

- zwischen den verschiedenen Ausgabewerten wechseln
- den aktuellen Wert nachschauen
- Getrunkenes eintragen
- Warnfunktion zum Trinken.

Es gibt vier unterschiedliche Ansichten (Wasserkonsum, Temperatur, Luftfeuchtigkeit und Luftqualität) zwischen denen der Nutzer wechseln kann. Das Wechseln funktioniert, indem das Tangible um 90° gedreht wird. Das Tangible ist hierbei so designt, dass es auf allen dieser vier Seiten stehen kann.
Sobald das Tangible gedreht wird, wechselt die Ansicht. Wenn das Tangible also z.B. auf Temperatur eingestellt ist und es gedreht wird, wechselt es zur nächsten Ansicht (z.B. Luftfeuchtigkeit).
Jede dieser Ansichten zeigt den jeweiligen aktuellen Wert im Raum in dem das Tangible sich befindet. Also die aktuelle Temperatur in °C, Luftfeuchtigkeit in %, Luftqualität in ppm und die Menge des bisher Getrunkenen des Tages in ml. Zusätzlich zur textuellen Anzeige des Wertes befindet sich am Rand des Displays eine visuelle Anzeige in Form von Kreisen, deren Anzahl und Farbe je nach Wert variiert. Ein höherer Temperaturwert bedeutet also auch mehr Punkte.
Die Farbe der Punkte orientiert sich auch am aktuellen Wert und bezieht sich auf die Farbskala der jeweiligen Ansicht (z.B. Dunkelrot für sehr hohe Temperaturen). Zusätzlich dazu leuchtet das Tangible in der entsprechenden Farbe.
Bei der Wasserkonsum-Ansicht gibt es des Weiteren die Funktion des Eintragens. Dazu befinden sich zwei Knöpfe ober- und unterhalb des Displays. Der obere Knopf ist hierbei dazu da die Milliliterzahl des Getrunkenen anzupassen. Sobald der Knopf gedrückt wird, erhöht sich die Anzahl in 50 ml Schritten und das Display und die Farbe des Tangibles passen sich entsprechend an. Mit der unteren Taste kann die Zahl jeweils um 50 ml verringert werden (z.B. bei Fehleingaben).
Letztlich gibt es noch eine Warnfunktion, die aber bisher noch nicht umgesetzt ist. Diese soll den Nutzer in bestimmten Zeitabschnitten warnen, wenn über einen längeren Zeitraum nichts oder nicht genug getrunken wurde. Diese Warnung erfolgt über ein Aufleuchten und eine Vibration des Tangibles, die ein paar Sekunden anhält.