Arbejdsportfolio

Dette dokument fungerer som projektets arbejdsportfolio. Formålet er at dokumentere processen "fra idé til færdig løsning". Her beskrives alle ideer og beslutninger, som fører til løsning af projektets elementer – fra den spæde idé, via skitser, til den endelige implementering.

Produktkrav

- Anordningen har 4 knapper, med tilhørende smileys.
- Der er respons på LED ved hver knap.
- På knapperne implementeres debounce funktionalitet.
- Lysdioden ud for den trykkede knap lyser
- Anordningen tilsluttes internet, via wi-fi access-punkt i undervisningslokalet.
- Anordningens ur synkroniseres løbende med NTP, og korrekt tidszone.
- Knaptryk-data (værdi og timestamp), overføres via MQTT:
  - Der bruges brugernavn og password, og hele kommunikationen er sikret ved brug af TLS-kryptering.
- Anordningen er designet strømbesparende, ved brug af DeepSleep og batteridrift:
  - Anordningen skal spare strøm ved at skifte til deep-sleep, når den er ubeskæftiget i et stykke tid.
  - Eksperimenter og analyser skal vise de bedste tidsgrænser for hvor længe anordningen er ubeskæftiget, men aktiv.
  - Samt, hvordan skal opvågnen fungere, så anordningen opdage hvad der sker, og falder i søvn igen.
  - En god analyse og overvejelse danner grundlag for hvor længe anordningen sover, inden den vågner rutinemæssigt.

Teknologier

Vi har valgt at skrive projektets software i **Rust**.

Til dette benytter vi **esp-rs/esp-hal**, som er officielt udviklet af **Espressif** specifikt til at understøtte Rust på ESP32-platformen. Dette giver os fordelene ved Rusts hukommelsessikkerhed kombineret med direkte adgang til hardwaren gennem værktøjer og biblioteker, der er målrettet netop denne platform.

NTP

Efter WiFi var implementeret, så det var tid til at implementere NTP. Satte til at bruge Danmarks NTP server (dk.pool.ntp.org) og lavede offset logik til sommer og vinter tid.

MQTT

Hvordan sætte MQTT broker op med TLS? Der var flere tanker om dette: Egen domain, så TLS er automatisk tilgængelig. Reverse proxy med certbot og letsencrypt

- nginx eller caddy? Det viser sig at eclipse-mosquitto image har sin egen TLS konfiguration og den brugte vi for at gøre det simpelt. Skulle generate certs og keys med openssl, og så konfigurere mosquitto til at bruge dem. Satte en nem MQTT broker op med docker compose.

WIFI

Skulle vælge mellem esp-wifi og esp-radio. Endte med esp-radio, da det understøtte bedre esp-hal version vi er på. For at nemt skifte WiFi SSID og password (og gøre det sikkert - ikke hardcode i source code), så satte vi WiFi config i .env fil. I build.rs skulle der blive loaded ind contents fra .env fil.

Debounce

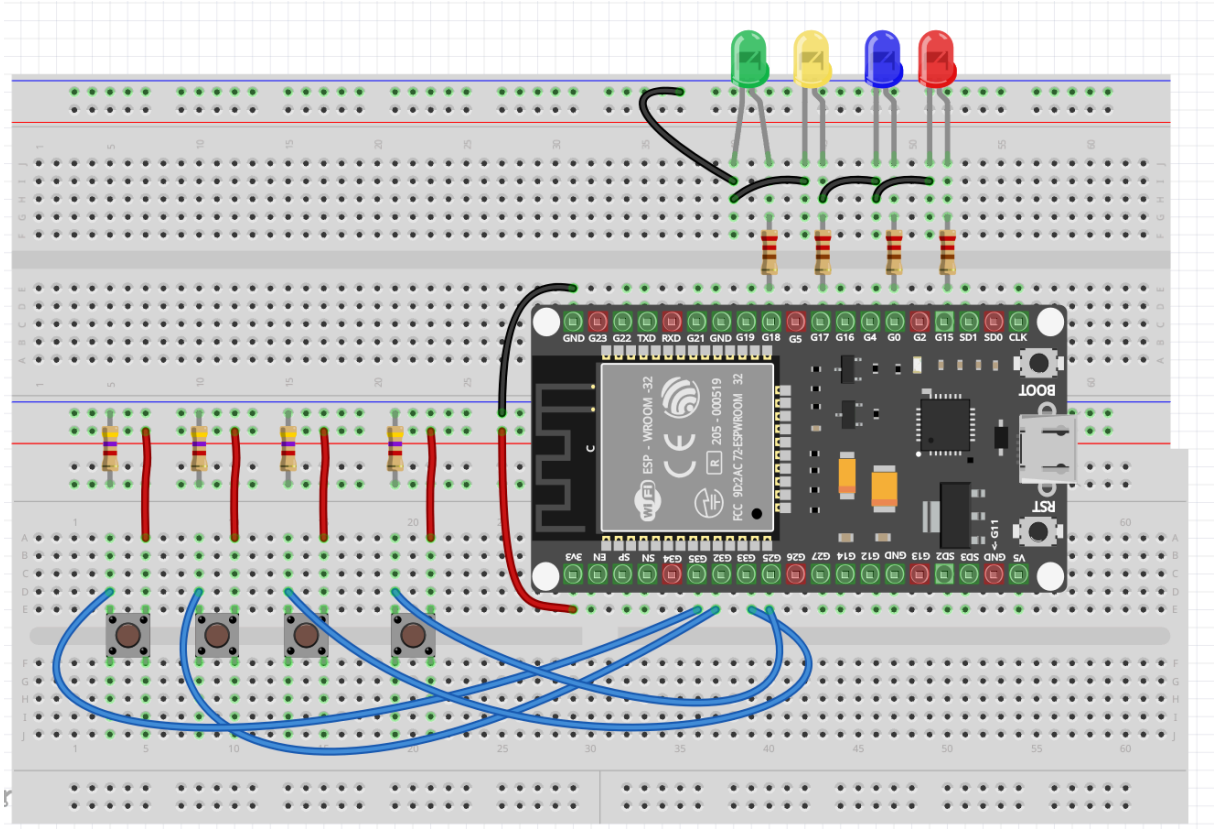
Deep sleep

Der skulle omkonfigureres på knappernes porte, så de brugte RTC GPIOs til at være kompatibelt med deepsleep. Deep sleep bliver aktiveret når enheden (ESP32) er inatkiv i mere end 10 sekunder.

TLS

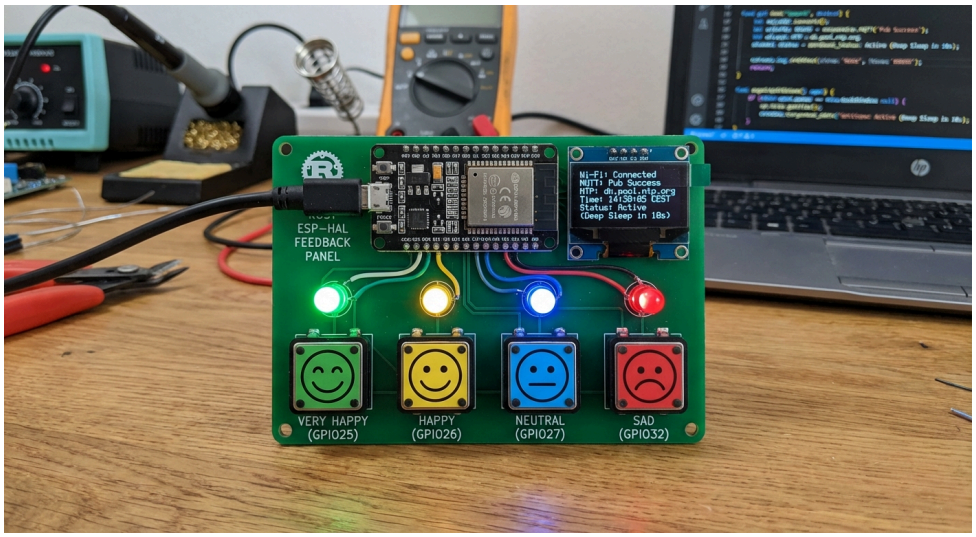
TLS har været SUPER SVÆRT at implementere. Prøvet adskillige libaries og refactor fra esp-hal til esp-idf-hal, som er community driven der inkludere std, som der ikke gør i bare-metal esp-hal. Prøvet embedded-tls - virkede ikke for os. Prøvet esp-mbedtls - virkede heller ikke. Skulle fork esp-mbedtls og ret i koden så den er mere kompatibelt med vores hardware og software versioner. - Den skulle ikke depend på esp-wifi. Secure TLS 1.3 kryptering med esp32 certs verifikation. Det lykkedes til sidst.

Kredsløbsdiagram



Figur 1: Fritzing kredsløbsdiagram

Figur 2: Projekt billede



Figur 3: Projekt billede