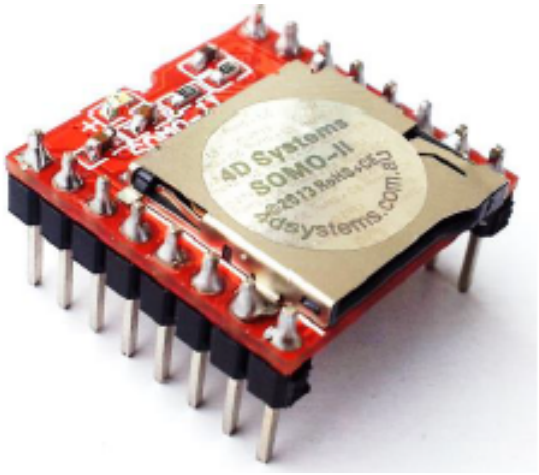
Bilag S: SOMO

2. juni 2023

Studienummer	Navn	Studieretning
202008660	Michelle Valentine Petersen	SW



Indhold

1 Indledning

Dette bilag er en samlet oversigt over udviklingen af lyd til produktets alarm.
Den omfatter de overvejelser, der er gået forud for valg af lydmodul og højtaler, her med henblik på fordele og ulemper ved hver kandidat.
Dernæst gennemgås designovervejelser, der naturligt fører videre til selve implementeringen. Til sidst testes med en modultest.

2 Design

2.1 Valg af lyd

Under overvejelserne omkring lyd til alarmen, lå valget mellem en buzzer og SOMO-II.
Valget faldt på SOMO, da dens fordele overskyggede fordelene for buzzeren. Fordele og ulemper ved hhv. den ene og den anden er diskuteret i afsnittene under.

2.2 Buzzer

En buzzer er en elektronisk enhed, der producerer en lyd, når strøm løber gennem den. Derudover kræver det blot en strømforsyning og selvfølgelig ground. Den påkrævede volt afhænger selvfølgelig af den valgte buzzer og ser ud til typisk at kræve mellem 1.5 til 15V[2]. Det ville derfor være forholdsvist simpelt at implementere en buzzer, og en buzzer ville leve op til projektets behov for en alarm.
I embedded stock er der kun en buzzer tilgængelig, der kan tage op til 12V. Den vurderes dog ret stor og vil derfor skabe pladsproblemer i systemet.

2.3 SOMO

SOMO er en indlejret audio-lyds modul, der kan afspille MP3 filer gemt på et micro SD-kort. Her bliver det en mulighed for brugeren at kunne vælge en personlig alarm til potten i en fremtidig version.
Det personlige valg kunne fremme tilknytningen til planten og gøre interaktionen med potten des mere givende for brugeren. SOMO er desuden velkendt fra første semester, så der ville være mere tilgængeligt data at finde.
Dimensionerne for en SOMO er også kun på 21.0 mm lang, 20.5 mm bred, og 11.3 mm høj, hvilket er en fordel, når pladsen er en faktor.
SOMO kan dog virke overkvalificeret, da kun en brøkdel af dens potentiale udnyttes på nuværende tidspunkt, og derfor sagtens kunne have været erstattet med et simplere alternativ som buzzeren.

2.4 Valg af højtaler

Ved valg af højtaler blev der vægtet størrelse, her især bredden, over lydkvalitet. Til projektets formål er det vigtigere med en højtaler, hvis dimensioner er minimale, end det er at kunne levere en høj kvalitet i lyd. Pladsfaktoren spiller en vigtig rolle i og med, at højtaleren skal kunne sidde i kassen sammen med skærmen, SOMO, Raspberry Pi, samt en masse ledninger, og der er derfor simpelthen ikke plads til en større højtaler. Den valgte højtaler er tynd, lille i størrelse og leverer en lyd, hvis kvalitet rammer projektets forventninger.

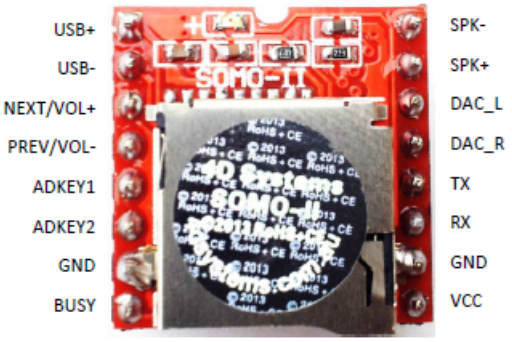
2.5 Kommunikationsprotokol

Ifølge SOMO's datablad[1] opererer den over enten KEY-MODE, hvor modulet kan styres direkte med blot to knapper uden brug af en ekstern processor, eller SERIAL-MODE, hvor Raspberry Pi styrer modulet igennem UART. Da alarmen ikke skal kaldes af brugeren, men skal kaldes af interface controlleren, Raspberry Pi, vælges den sidstnævnte tilstand.

2.6 Hardware

SOMO kan styres igennem dens TTL seriel UART-interface via dens TX og RX pins. Da SOMO kun skal modtage kommandoer fra Raspberry Pi, og dermed ikke sende beskeder tilbage, benyttes TX ikke.

Til gengæld benyttes RX, GND, VCC og SPK-/+, der alt sammen på nær SPK-/+ tilkobles Raspberry Pi. Højtaleren forbindes til SPK-/+. SOMO'ens pins kan ses på figur 1.
Der indsættes et mikro SD-kort med den valgte MP3-fil på.
Modulet understøtter TTL 3.3V UART-interface til enhver micro-controller. Raspberry Pi's interface er nøjagtigt på 3.3V. Så der kan kobles en ledning direkte mellem RX på SOMO og TX (gpio14) på Raspberry Pi.
Forbindelserne mellem SOMO og Raspberry Pi kan ses på tabel 1.



Figur 1: SOMO pins

Tabel 1: Forbindelser mellem SOMO og Raspberry Pi

SOMO	Raspberry Pi
VCC	gpio (3.3 V)
GND	gpio (GND)
RX	gpio14 (TX)

2.7 Software

Softwaren er designet som en klasse, der agerer driver. Dens eneste funktion i softwarearkitekturen, playsound(), er at afspille en lyd, der sender kommandoen til afspilning af første fil på SD-kort.

Klassen er ydermere blevet udvidet til at opsætte en seriel port og sætte baud rate til 9600 i constructor, da det er den SOMO kører med. Faktisk er ttyS0, som er den anvendte seriel port, per default konfigureret til 9600 baudrate, men det er god skik at sikre sig, at den nu også er sat til den ønskede baudrate.
I destructor lukkes selv samme seriel port.
For at afprøve SOMO's funktionalitet og bygge videre til fremtiden er der tilføjet ekstra metoder til bl.a. at justere volumen eller sæt den i en sovende tilstand med hensigt på at spare på strømforbruget, når den ikke afspiller alarm.

Det var hensigten at skrive en driver til SOMO'en, men det blev droppet, da det ikke gav mening med det lave niveau af kompleksitet, som der kræves af modulet. Samtidig er der allerede et veletableret, stabilt og simpelt interface, der kan køres fra userspace, som vil være det smarteste at anvende i vores projekt.
Desuden bør man ikke pille ved kernelspace, jo mindre man intet andet valg har, da det ikke er uden fare.

3 Implementering

Klassen der styrer SOMO-modulet er implementeret med brug af det allerede eksisterende UART interface, komplimenteret med metoder fra boost/asio biblioteket, da det hjælper med at udføre asynkrone I/O operationer.

3.1 asio::bost bibliotek

Metode attributten io_ er et objekt af klassen asio::io_context, der repræsenterer udførelseskonteksten for I/O operationer, såsom seriel kommunikation, hvor asio::serial_port er en klasse, der repræsenterer en seriel port til seriel kommunikation, der tillader at åbne, konfigurere og udføre I/O operationer. Et eksempel på dette kan ses i contructoren på figur 2, hvor port_ kalder open() metoden på argumentet device, der i vores tilfælde ville være ttyS0.
Et samlet billede af header filen for klassen kan ses på figur 3.

```
SoundModuleDriver::SoundModuleDriver(const std::string& device)
        : io_(), port_(io_), device_(device)
    {
        // Open the serial port
        port_.open(device_);

        // Set the baud rate to 9600
        port_.set_option(asio::serial_port_base::baud_rate(9600));
}
```

Figur 2: SOMO konstruktor

```
#include <boost/asio.hpp>

using namespace boost;

class SoundModuleDriver {
public:
    SoundModuleDriver(const std::string& device);
    ~SoundModuleDriver();
    void playSound();
    void increaseVolume(int x);
    void decreaseVolume(int x);
    void sleep();
    void reset();
private:
    asio::io_context io_;
    asio::serial_port port_;
    std::string device_;
};
```

Figur 3: Header fil til SOMO

3.2 playSound()

Hovedfunktionen for klassen er at sende en kommando til afspilning af første fil på SD-kort, hvilket sker i playSound(), som set på figur 4.
I databladet for SOMO kan man se kommandoen for play, som er vist på figur 5.

```
void SoundModuleDriver::playSound() {
        // Send the command to play a sound
        uint8_t play[] = {0x7E, 0x0D, 0x00, 0x00, 0x00, 0xFF, 0xF3, 0xEF};
        asio::write(port_, asio::buffer(play, sizeof(play)));
}
```

Figur 4: Metoden playSound()

PLAY	7E 0D 00 00 00 FF F3 EF	Play the audio track selected (if selected) else the first track copied on to the media (See Section 6)

Figur 5: Kommando til afspilning af første fil på SOMO

4 Modultest

SOMO modultestes ved at skrive en main, der opretter et klasse objekt, der derefter kører playSound() metoden for at teste om UART forbindelsen er sat korrekt op, og om den korrekte kommando når frem til SOMO, samt om MP3-filen er lagt ind i den rigtige struktur.

4.1 Fil og Mappe Struktur

For at SOMO kan finde og afspille ens MP3-fil skal der indsættes en mappe, der hedder "01", hvor den første fil skal starte med "001". SOMO er faktisk i stand til at adressere 99 mapper, hver med 255 filer i.
Men til vores system er det blot nødvendigt med én fil, der navngives "001 Alarm".

4.2 Linux Opsætning

Undervejs i forløbet med testningen på Raspberry Pi har det været nødvendigt at deaktivere konsol output ved at ændre i tekstfilen /boot/cmdline.txt. Figur 6 viser, hvad der skal ændres fra og til.

```
dwc_otg.lpm_enable=0 console=serial0 root=/dev/mmcblk0p2 rootfstype=ext4 rootwait modules-load=dwc2,g_ether
```

to

```
dwc_otg.lpm_enable=0 root=/dev/mmcblk0p2 rootfstype=ext4 rootwait modules-load=dwc2,g_ether
```

Figur 6: /boot/cmdline.txt: Konsol output deaktiveres

Desuden skulle login funktionalitet deaktiveres ved at skrive i terminalen: systemcl disable serial-getty@ttyS0.service
Før klassen blev sat op til at håndtere opsætningen af baudrate kunne det gøres med: stty -F /dev/ttyS0 9600.

4.3 main()

Klassen testes ved at køre følgende main(), som kan ses på figur 7, hvor målet er at sende play-kommandoen til SOMO, så den afspiller MP3-filen på micro SD-kortet.

```cpp
#include "somo_class.hpp"
#include <boost/asio.hpp>

using namespace boost;


int main() {
    // Create an instance of the SoundModuleDriver and specify the device name
    SoundModuleDriver driver("/dev/ttyS0");

    // Play a sound using the driver
    driver.playSound();
    //driver.increaseVolume(20);
    //driver.sleep();
    return 0;
}
```

Figur 7: main program til test af SOMO klassen

Litteratur

[1] Mouser Electronics. *SOMO II Datasheet*. 2021. URL: https://www.mouser.com/datasheet/2/451/SOMO_II_Datasheet_R_1_6-1627245.pdf.

[2] Quisure. *What is the Working Principle of the Buzzer?* 2023. URL: https://www.quisure.com/blog/faq/what-is-the-working-principle-of-the-buzzer.