

CarpeDiem

Alarmklocka med ljud och ljus

Projekt för kursen Realtidssystem

Medlemmar

Dylan Saleh

Johan Kämpe

Pay-Shin Quach

Stefan Ekström

Handledare

Tomas Kindahl

1 Inledning	3
1.1 Syfte	3
1.2 Projektspecifikation	3
1.2.1 Design	3
1.2.2 Användningsfall	4
2 Genomförande och resultat	5
2.1 Metod	5
2.2 Komponenter	6
2.3 Användning av produkten	7
2.3.1 Display	7
2.3.2 Inställningar med joystick	8
2.4 Projektets kod	9
2.4.1 Använda bibliotek	9
2.4.2 Tasks och semaforer	9
3 Diskussion och slutsats	11
3.1 Problematik under utvecklingsarbetet	11

1 Inledning

1.1 Syfte

Syftet med projektet är att skapa en produkt för hemanvändning, med kombifunktionalitet. Funktionaliteterna ska exekveras parallellt och systemet ska använda sig av FreeRTOS.

Produkten som valdes att ta fram är en alarmklocka, Denna beskrivs i projektspecifikationen nedan.

1.2 Projektspecifikation

Projektet är en alarmklocka som används för att väcka sovande personer på morgonen. Ljus och eller ljud används vid väckning. Klockan visar också tiden.

Klockan kan sättas i olika lägen:

Ljudlöst	Endast ljus används vid väckning.
Ljusbäst	Endast ljud används vid väckning.
Full	Både ljus och ljud används vid väckning.
Klocka	Klockan används endast för att visa tid.

1.2.1 Design

Följande komponenter antogs behövas i projektet:

- **MCU**
- **RTC**
- **Ljuskälla:** Eventuellt LED-ring.
- **Ljudkälla**
- **Tidsvisning:** Eventuellt display eller LED-ring.
- **Input:** För att ställa klocka / mode / alarm

1.3 Länkar

Projektets GitHub-sida

<https://github.com/GoblinDynamiteer/CarpeDiemAlarmClock>

1.2.2 Användningsfall

I projektspecifikation finns ett användningsfall som beskriver ett scenario där produkten används:

En familj med en ensamstående mamma och tre tonårsbarn; mamman börjar jobbet tidigt och vill inte väcka sina barn som sover i samma rum.

Hon har satt sin "CarpeDiem" Alarm Clock på läge silent mode, vilket innebär att en gradvis ökande ljusintensitet används för att väcka personen, utan ljud.

Innan mamman åker till jobbet sätter hon ett nytt alarm, en timme senare, för att väcka sina barn, men denna gång i läge full mode, vilket innebär att alarmklockan väcker med både ljud och ljus.

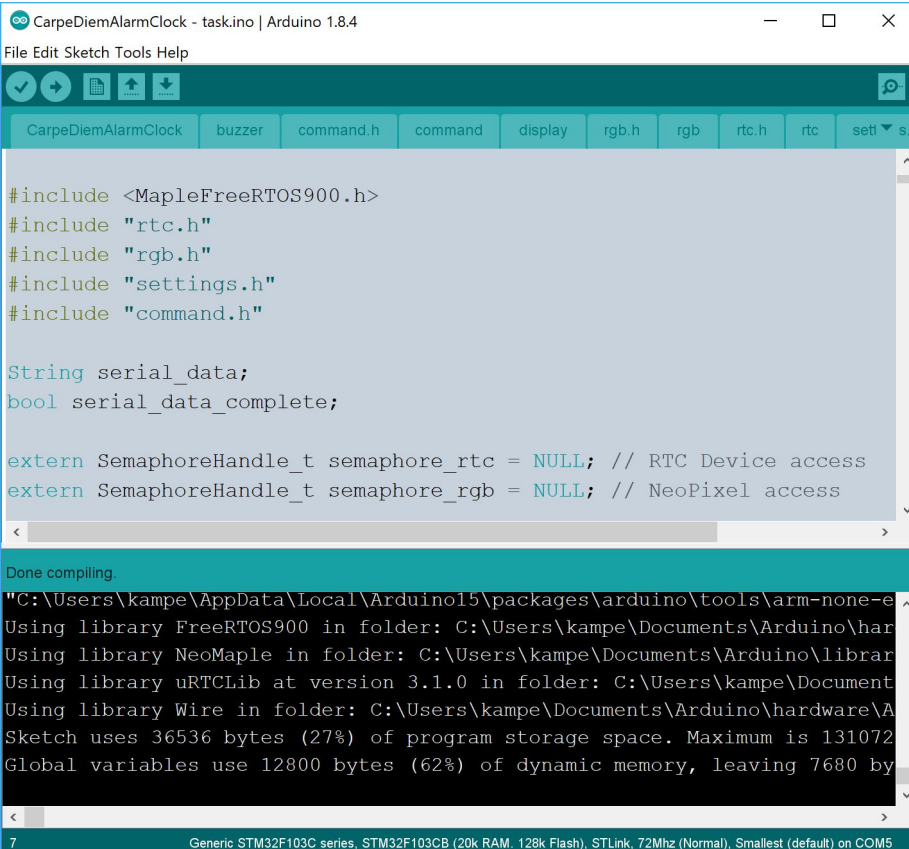
2 Genomförande och resultat

2.1 Metod

Projektets kod skrevs i programspråket C++. Verktyget *STM32duino* användes för att kunna skriva "Arduino-kod" till det STM32-kort som användes.

Komponenter som skulle användas i projektet testades var för sig, oftast på en Arduino UNO-enhet innan det implementerades i projektet.

Kod skrevs i en texteditor eller i utvecklingsmiljön Arduino, och kompilerades och laddades upp till STM32-kortet med Arduino.



```
CarpeDiemAlarmClock - task.ino | Arduino 1.8.4
File Edit Sketch Tools Help

CarpeDiemAlarmClock buzzer command.h command display rgb.h rgb rtc.h rtc sett s.h

#include <MapleFreeRTOS900.h>
#include "rtc.h"
#include "rgb.h"
#include "settings.h"
#include "command.h"

String serial_data;
bool serial_data_complete;

extern SemaphoreHandle_t semaphore_rtc = NULL; // RTC Device access
extern SemaphoreHandle_t semaphore_rgb = NULL; // NeoPixel access

Done compiling.
"C:\Users\kampe\AppData\Local\Arduino15\packages\arduino\tools\arm-none-e
Using library FreeRTOS900 in folder: C:\Users\kampe\Documents\Arduino\har
Using library NeoMaple in folder: C:\Users\kampe\Documents\Arduino\librar
Using library uRTCLib at version 3.1.0 in folder: C:\Users\kampe\Document
Using library Wire in folder: C:\Users\kampe\Documents\Arduino\hardware\A
Sketch uses 36536 bytes (27%) of program storage space. Maximum is 131072
Global variables use 12800 bytes (62%) of dynamic memory, leaving 7680 by
7 Generic STM32F103C series, STM32F103CB (20k RAM, 128k Flash), STLink, 72Mhz (Normal), Smallest (default) on COM5
```

Kod i Arduino

2.2 Komponenter

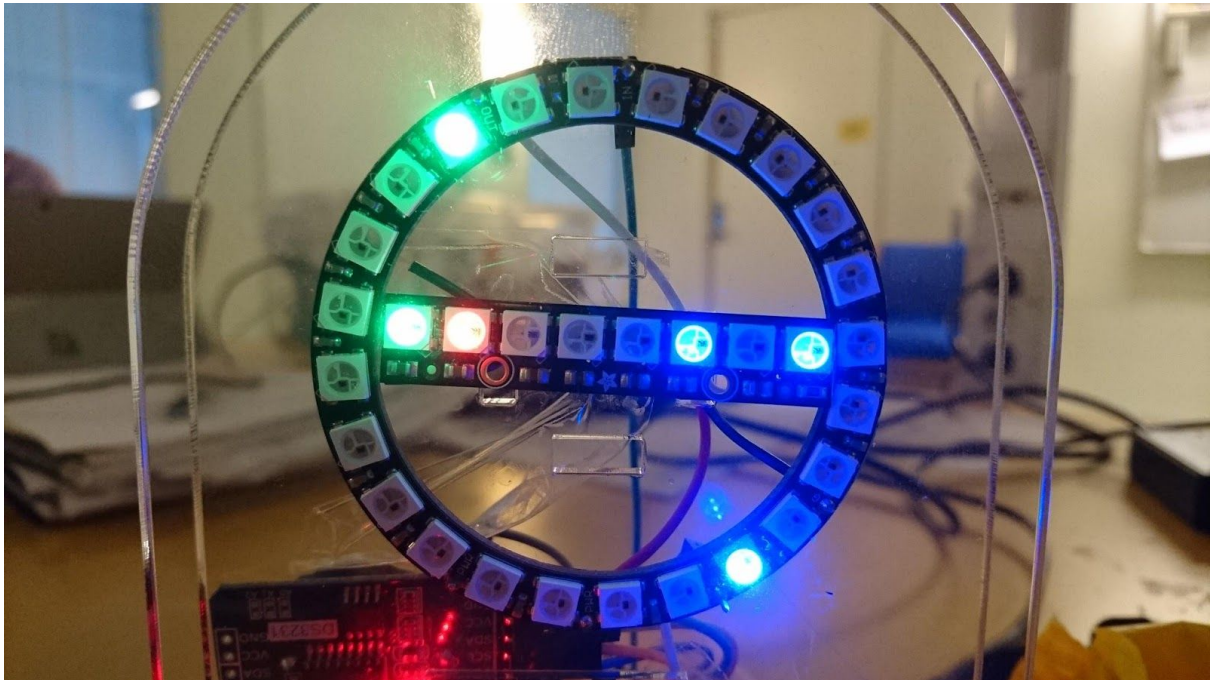
Följande komponenter valdes att användas för produkten:

Komponent	Användning
STM32F103C "Bluepill"-utvecklingskort	MCU av ARM-typ.
Adafruit NeoPixel LED-ring (24 LED)	LED-ring, används för att visa tid och ljuskälla för alarm. Används också för att visa "ljusshower" om användaren inte vill ha tidsvisning.
Adafruit NeoPixel LED-strip (8 LED)	Visar sekunder i binär form på 6 LEDs, resterande två LEDs används som statusindikatorer för buzzer och alarm.
Joystick	Analog rörelse i X-led och Y-led, samt knapp.
HC-06 Bluetooth-modul	För UART-kommunikation.
Buzzer	För ljud vid alarm.
DS3231 RTC-modul	Realtidsklocka.

Kopplingsschema finns bifogat som en bilaga.

2.3 Användning av produkten

2.3.1 Display



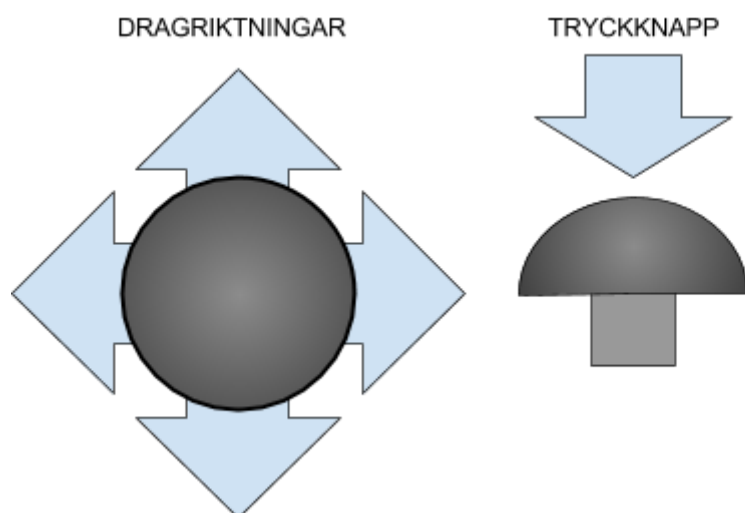
LED-ring

Grönt ljus representerar klockans timme och blått ljus representerar klockans minut.

LED-strip

Den första biten är en indikator på om buzzern är på/av, den andra biten en indikator på om alarmet är på/av. **Grön** = på. **Röd** = av. De resterande sex bitarna representerar sekunderna binärt; **Blå** = 1, avstängda = 0.

2.3.2 Inställningar med joystick



Joystick-riktningar: Vyer uppifrån och sida

Display på/av

Styr joysticken uppåt och släpp, för att starta alarmklockan, då kommer tiden på klockan visas. Dra joysticken uppåt igen för att stänga av.

Ställa klockan

Tryck på joysticken, då visas först timme och pixel väljs genom att dra joysticken åt höger för medsol eller vänster för motsol. Godkänn med ett tryck.

Ställ nu in minut på samma sätt som timme och godkänn med ett tryck. Nu kommer tiden att visas.

Ställa alarmet

Tryck på joysticken och håll in i minst två sekunder. Ställ in alarmet på samma sätt som klockan. Nu ska alarmet vara på.

Ljusshower

Dra joysticken nedåt en gång och släpp, för att växla mellan ljusshower. Efter den sista ljusshowen visas klockläget igen.

Buzzer på/av

Dra joysticken åt vänster för att sätta på eller stänga av buzzern. Dra den en eller två gånger tills önskat läge visas.

Alarm på/av

Dra joysticken åt höger för att sätta på eller stänga av alarmet. Dra den en eller två gånger tills önskat läge visas.

2.4 Projektets kod

Projektets kod skrevs i C++. FreeRTOS användes för implementation av realtidssystem. Projektets kod delades upp i flera källkodsfiler och header-filer:

buzzer.ino	rgb.ino
CarpeDiemAlarmClock.ino	rtc.h
command.h	rtc.ino
command.ino	settings.h
display.ino	settings.ino
rgb.h	task.ino

2.4.1 Använda bibliotek

Förutom *STM32duino* och *FreeRTOS* användes följande två bibliotek i projektet:

Bibliotek	Användning
NeoMaple	Portning av AdaFruit NeoPixel-biblioteket för att styra LED-ringar / LED-stripar.
uRTCLib	Enkelt bibliotek för RTC

2.4.2 Tasks och semaforer

Följande tasks skapades för projektets kod

Namn	Användning
time_handler	Pollar RTC-modulen om tid och uppdaterar sekunder på LED-stripen.
alarm_handler	Hanterar alarm
rgb_display_handler	Sätter RGB-värden på LED-ring och LED-strip.
rgb_updater	Uppdaterar (visar) aktuella RGB-värden på LED-ring och LED-strip.
serial_command	Hanterar inkommande UART-kommandon från användare. Används för debugging och testning.
joystick_input	Hanterar användarinput med joystick.

Samtliga tasks skapades samma inställningar för minnesanvändning och prioritet, och ingen tar emot några parametrar.

```
xTaskCreate(  
    time_handler,  
    "time_handler",  
    configMINIMAL_STACK_SIZE,  
    NULL,  
    tskIDLE_PRIORITY + 2,  
    NULL);
```

Skapande av task

Två semaforer användes i projektet:

Namn	Typ	Användning
semaphore_rtc	Mutex	Tillgång till RTC-klockan och funktioner i rtc.ino
semaphore_rgb	Mutex	Tillgång till NeoPixels och funktioner som använder sig av dessa.

3 Diskussion och slutsats

3.1 Problematik under utvecklingsarbetet

Under utvecklingen av produkten testades att visa tiden på två stycken LED-ringar, samt LED-strip. Timmar skulle visas på en ring med 24 st LED och minuter på en med 12 st. Dock visade det sig att det inte gick att driva dessa tre samtidigt och idén skrotades.

NeoMaple-biblioteket använder en fördefinierad pin för data. Denna pin inte gå att ändra, därför var det ej möjligt att koppla in olika NeoPixels på olika pins. Lösningen som användes för detta var att seriekoppla NeoPixel-ringen och strippen. Dessa kan då användas som en enda enhet.

3.2 Förbättringsförslag

Byggnation

I skrivande stund är komponenter fasttejpade på en fixtur och drivs med en powerbank via en ST-LINK-programmerare. En mer attraktiv produkt skulle kunna byggas, och en bättre strömförsörjning skulle kunna väljas.

En front / urtavla skulle kunna tillverkas, som visar siffror på urverket, och har förklaring för status-LEDarna.

Kod

Programkoden innehåller flera småbuggar, dessa skulle kunna åtgärdas.

En app skulle kunna utvecklas, för att sätta alarm / klocka via en telefon. Produkten har redan stöd för att ta emot kommandon via UART.

Bilaga 1 - Kopplingsschema

