Rapport

Linda Kjellberg IoT22

Linda.kjellberg@yh.nackademin.se

## Innehållsförteckning

[Inledning 3](#_Toc6624)

[Dagbok 4](#_Toc6625)

[Resultat 6](#_Toc6626)

[Referenser 8](#_Toc6627)

# Inledning

Skriv en inledning som förklarar:

* Vad är syftet med rapporten
* Vad är syftet med arbetet som vi har gjort
* Vad är det vi ska åstadkomma?

Förtydligande:

I den här rapporten används begreppen main och huvudfunktion omväxlande. Även UART-protokollet och USART2.

SYFTE

Syftet med denna rapport är att ge läsaren insikt i den plattforms-lösning som rapporten beskriver, den förklarar de funktioner som används i lösningen, hur dessa fungerar och vilket samspel som finns mellan de olika källkod-filerna.

Den här lösningen består av drivrutiner till STM32F411-platformen skapade med ett UART kommunikationsprotokoll, med syftet att kontrollera periferienheter bestående av LED lampor.

Tanken med projektet är att få valda LED-lampor av fyra möjliga färger (röd, grön, gul och blå), att blinka vid kommando. Denna funktionalitet styrs av de två funktionerna ”USART2\_read” som ansvarar för att ta emot data, och ”USART2\_write” som ansvarar för skickad data. Kommunikationen mellan plattformen och periferienheterna möjliggörs med hjälp av USART2, vilket är en half-duplex, Syncrom variant av UART- kommunikationsprotokollet.

Syftet bakom detta projekt var att uppnå en fördjupad förståelse av vanligt förekommande kommunikationsprotokoll inom inbyggda system, hur dessa fungerar och hur vi kan tillämpa dem för kommunikation mellan enheter.

----------------------------------------------------------------------------------------------------------------

# Dagbok

Hur vi utvecklade drivrutins projektet:

1. Bekantade oss med hårdvaran och utvecklingsmiljön
2. Bekantade oss med referensmanualen och databladet
3. Bekantade oss med att utveckla UART-protokollet
4. Bekantade oss med att utveckla Periferi-drivrutinerna som kontrollerar LED lamporna
5. Satte ihop projektet med en main funktion som kallar på allt

Ist för dagbok skriv en flytande text som innefattar:

* Vad man har gjort
* Varför man gjorde det
* Hur man gjorde det

Inför uppstarten av detta projekt har vi bekantat oss med några vanligt förekommande kommunikationsprotokoll som används vid utveckling och drift av inbyggda system, och hur man kan implementera dessa. Däribland UART (Universal Asynchronous Receiver/Transmitter) vilket är det kommunikationsprotokoll vi kommer använde oss av i det här projektet.

Inledningsvis fick vi bekanta oss med den hårdvara och den utvecklingsmiljö som var tänkt att vi skulle använda i projektet. Detta gjordes genom att titta på en YouTube-tutorial som läraren hade hänvisat till. I videon sattes det upp ett liknande projekt till det vi har utfört, och en genomgång av hur man använder STM32 CubeIDE – den programvaran (kodeditor) som var tänkt att vi skulle använda i det här projektet.

Nästa steg var att bli bekant med relevant dokumentation tillhörande hårdvaran. Detta innefattade datablad, referensguide samt användarmanual för STM32F4-plattformen för att få en bättre förståelse för hur plattformen är uppbyggd och hur man arbetar med den. Det blev en stor utmaning för mig att läsa igenom all dokumentation och förstå hur jag skulle kunna applicera den i projektet. Jag tyckte att det både var svårt att förstå hur man läser av dokumentationen och även vilka delar som vi skulle arbeta med i projektet. Trotts att vi innan uppstart grundligt gick igenom vilka delar som var viktiga att kunna läsa av så upplevde jag en hel del förvirring kring hur jag skulle gå till väga för att få fram den informationen jag skulle söka.

Därefter gick jag vidare med att bekanta mig med hur man utvecklar drivrutiner med hjälp av UART-protokollet, detta gjordes med en uppgift där vi fick ett stycke kod som vi skulle skriva kommentarer till och förklara koden med hjälp av den dokumentation vi tidigare fått. Även det var en stor utmaning för mig, jag fann det svår att förstå hur jag skulle söka få fram information om hårdvaran i dokumentationen. Och jag kände mig vilsen i uppgiften.

Fortsatt bekantade vi oss med att utveckla Periferi-drivrutiner som kontrollerar LED lamporna på samma sätt som med UART-övningen – vi fick färdiga kodstycken som vi med hjälp av dokumentationen skulle kommentera för att själva få en förståelse för vad koden gör och hur man konstruerar drivrutiner till den här typen av projekt.

Jag upplevde det som förvirrande och hade liknande motgångar med att hitta den information jag behövde. I efterhand tror jag att jag hade behövt mer förståelse för dokumentationshantering.

Slutligen satte vi ihop projektet i main.cpp-filen med huvudfunktionen main som kör projektet. I filen skapas inledningsvis tre objekt upp som instanser av LED lampor globalt, dessa kommer kunna utnyttja den funktionalitet vi har satt upp åt lamporna med hjälp av objektens konstruktorer. Det vill säga – de kommer kunna anta en av de för-konstruerade färgerna (röd, blå, grön eller gul), samt anta ett utav två tillstånd (på/av). Därefter initialiseras den första lampan led1 samt ställs in till färgen röd och tänds utanför huvudfunktionen.

Fortsatt startar vi upp huvudfunktionen som först initialiserar USART2 protokollet. Och inuti main initialiserar vi även led2 till färgen blå och tänder lampan.

Sen skapar vi ett nytt Led objekt med färgen gul som vi också tänder. Och den här lampan allokerar vi minne dynamiskt för, samt med en pekare tilldelas det nya objektet variabelnamnet led3. Senare tar vi bort objektet så det allokerade minnet frigörs.

Vi \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Huvudfunktionen avslutas med en oändlig loop som upprepar funktionen så länge programmet körs.

!!! KOLLA GENOMG AV KODEN FÖR GETSTATE!!!

----------------------------------------------------------------------------------------------------------------

Led.cpp:

Det är i LED.c filen som vi drar igång funktionaliteten och där vi pekar med konstruktorn (struct) till LED.h (header filen) samt sätta funktionerna som hämtar- och sätter statusen.

I LED.h har vi skapat en led när den har en färg och ett tillstånd (på/av)

Vi konstruerar en struct som inehåller (tar argumenten) (Led\_Type som pekar på konstanten const me, och LedColor\_Type som betonas av \_color, och LedState\_Type som betäcknas av \_state)

Led.h:

UART.cpp:

UART.h:

I uart.h initialiserar vi periferienheterna med enum function

stm32f4xx.h:

Detta är en headerfil som automatiskt lades till i projektet vid uppstart i CubeIDE. Den innehåller nödvändiga funktionalitet för STM32-platformen.

Som avslut i projektet var det tänkt att vi skulle testköra projektet i simuleringsprogrammet Proteus, men det gick inte då vi stötte på problem med kodhanteringen efter en ny uppdatering av mjukvarans demoversion som vi skulle använda.

----------------------------------------------------------------------------------------------------------------

Baud rate är bits per second som vi kan skicka över seriekommunikation

PINSEN ÄR INTE RIKTIGT OFF AND ON, UTAN MER FUNKTIONSHANTERING

USART2 kommunikationsprotokollet, vilket är en half-duplex, Syncrom variant av UART som använder sig av en klocksignal för att \_\_\_\_\_\_\_\_, till skillnad från UART som är ett full-duplex protokoll utan klocksignal.

----------------------------------------------------------------------------------------------------------------

# Resultat

Resultatdel:

* Länka till mitt Github repo
* Repot ska innehålla en READme som förklarar projektet är
* Hur det ska användas
* Vilka funktioner projektet innehåller (write och read)
* Ett exempel på hur man konfigurerar UART med hjälp av vår drivrutin

----------------------------------------------------------------------------------------------------------------

# Referenser

Lektionsmaterial och muntlig information från Ludwig Simonsson

Dokumentation tillhörande STM32F411-platform:

* Datablad
* Referensmanual
* Användarmanual