

**Inbyggda System Arkitektur och Design**

**Inlämning**

Inledning…….

Komponenter….

Kod….

Slutsats…

IoT 2022

Ellära

Jonatan Ghirmay

[jonatan.ghirmay@yh.nackademin.se](mailto:jonatan.ghirmay@yh.nackademin.se)

**Inledning**

I denna rapport så kommer jag att gå igenom ett projekt för en LED kopplad till en STM32 mikrokontroller. Denna LED kommer att kommunicera med mikrokontrollern med hjälp av UART protokollet.

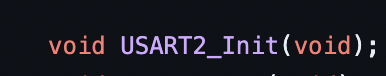
Syftet och målet med detta arbete är att påvisa kunskap och en övergripande förståelse inom “Inbyggda System” och i det här fallet genom detta LED projekt med STM32 mikrokontrollern.

I projektet så kommer det att finnas en kod “mapp” där all kod finns deklarerad med kommentarer som förklarar vad varje kod gör och har för funktion i kommunikationen mellan LED och mikrokontroller hårdvaran. I denna rapport så kommer jag främst att förklara koden och hur programmet för denna LED med STM32 mikrokontroller i UART-protokollet är uppbyggt och strukturerat.

Github länk: <https://github.com/jonghi11/UART_LED>

**Kod**

Filerna med kod är upplagda så att det finns en “Main” fil som initierar programmet och kallar på funktionen som skall sätta igång eller stänga av lampan på önskat sätt. Sedan så har vi två “Header” filer. I den ena “Headern” som är för LED så finns definitionerna för att enklare kunna läsa programmet samt vissa klasser och funktioner. Sedan så finns en “Header” fil för själva UART:en där vi främst har funktionen som initierar UART-protokollet inklusive en testfunktion, “STM32” mikrokontrollerns header inkluderad.



De två filerna med mest kod i är i detta fall LED och UART vardera “huvudfil” där vi i UART:ens huvudfil sätter pins, diverse register, portar till de värdena vi vill med hjälp av “bitwise operations". I LEDens huvudfil så har vi främst funktionerna för LEDen, där exempelvis ON/OFF funktioner finns samt switch cases med if-satser för hur LEDen med pins etc bör te sig när man skall byta lampa och liknande.

LED.h

Till en början så definierar vi porten på STM32 kortet som vi vill använda för LED funktionen. Där väljs "GPIOB"-porten som både kan användas för input och output. Sedan så definierar vi klocksignal för nämnda port genom att sätta den första biten som 1 och resten kvarstår som 0. Denna konstant sätts för att kontrollera klocksignal för LED-porten.





Efter detta så definieras bits som styr skall styra pins och moden till specifika LED färger.





I exemplen ovan så ser man hur vi definierat den 14:e biten från höger till 1 för att kontrollera röd pin och den 28:e biten från höger till 1 för att kontrollera moden på denna röda LED. Detta görs även sedan med resten av LED färgerna grön, gul blå på samma sätt med olika bits.

I koden så sätts sedan värdena med “enum” inkrementering för LED färgerna där varje färg har sitt eget värde från 0-3 och samma sak görs med ON = 0/OFF = 1 funktionen

RED = 0,

GREEN = 1

YELLOW = 2

BLUE = 3



Dessa kommer sedan att kunna kallas på från klassen “Led” med vardera argument för att kalla på attributen(color, state). I och med att dessa två attribut är privata så kommer vi göra en setter och getter funktion för att kunna kalla på dem i main filen.

LED.cpp

I LED filens huvudfil så börjar vi med att “include” LED headerfilen till programmet. Sedan så tar vi klassen “Led” från “Led.h” och för att sätta attribut, konfigurera i konstruktorn som vi skapat från klassen Led.



“RCC” som kontrollerar klocksignal på en på en lämplig frekvens för kärnan och “peripherals”. Den sätter vi till AHBENR1 som är ett register som har en högsta frekvens på 100 MHz. Dessa värden sätts med en OR bitwise i LED\_PORT\_CLOCK(GPIOB porten) för att enable:a klockan i.

**REFERENS & BILAGOR**

