NACKADEMIN

Author: Marcus Peterson, Mjukvaruutvecklare Inbyggda System & IoT

Dokumentation för LED-kontroll, UART, STM32

Det här projektet är uppbyggt kring flera huvudkomponenter som tillsammans fungerar för att kontrollera LED-belysningen.
Add Headings (Format > Paragraph styles) and they will appear in your table of contents.

Mikrokontroller

(STM32F411xC / STM32F411xE): Denna är hjärtat i hela systemet. Den styr alla andra komponenter och utför logiken som definieras i din kod.

```
#include "stm32f4xx.h"//Hämtar källkoden och hänvisningar för arbete på STM-Hårdvara
```

Den har också många inbyggda funktioner som vi använder i det här projektet, inklusive timers, DMA, och flera kommunikationsgränssnitt. LED-objekt: Dessa är de faktiska LED-lamporna som vi kontrollerar.

Vi har tre av dem i vår kod (en röd, en blå och en gul), men du kan lägga till så många du vill.

```
// Definerar de olika LED-färgerna som finns
```

```
typedef enum
{
  RED = 0,
  GREEN,
  YELLOW,
  BLUE
} LedColor_Type;
```

Varje LED-objekt har en tillståndsvariabel som kan vara antingen ON eller OFF, och en färgvariabel som bestämmer LED-lampans färg. USART2: Detta är det seriella gränssnittet som vi använder för att kommunicera med mikrokontrollern.
Vi använder det för att skicka kommandon till mikrokontrollern och få tillbaka statusinformation.

```
Led led1(RED, ON);
Led led2(BLUE, ON);
Led *led3 = new Led(YELLOW, ON);
```

LED.cpp och LED.h filer:

Dessa filer innehåller koden för LED-klassen, som vi använder för att skapa våra LED-objekt. LED-klassen har metoder för att ställa in och få tillståndet för en LED, samt för att ändra dess färg.
main.cpp:

Detta är huvudfilen i projektet, där all logik körs.
Den innehåller en oändlig loop som kontinuerligt kontrollerar tillståndet för varje LED och styr den i enlighet med det. Och inuti while loopen har vi felhantering utifall det skulle vara något fel med lamporna.

```
while (1) {
    // Kontrollera om någon av LED-objekten är
null
```

```
    if(isLedNull(&led1) || isLedNull(&led2) ||
isLedNull(led3))
    {
      printf("Error: a LED object is null\n");
      return 1;
    }
}
```

Alla dessa komponenter är sammankopplade och interagerar med varandra för att uppnå det önskade resultatet.

Mikrokontrollern läser in kommandon från USART2, bearbetar dem och ändrar tillståndet för LED-objekten i enlighet med dessa kommandon.

LED-objekten skickar sedan signaler till de fysiska LED-lamporna för att tända eller släcka dem.

Teknologier som används

C++: Hela projektet är skrivet i C++, vilket gör det möjligt att skapa objektorienterad kod med klasser och objekt. Detta gör koden mer strukturerad och lättare att förstå och underhålla..USART2: Detta är det seriella gränssnittet vi använder för att kommunicera med mikrokontrollern.

Det gör att vi kan skicka kommandon till mikrokontrollern och få tillbaka statusinformation. STM32CubeIDE: Detta är den integrerade utvecklingsmiljön (IDE) som vi använder för att skriva och kompilera vår kod.
Den är speciellt designad för att arbeta med STM32 mikrokontroller och har många inbyggda funktioner som gör det enklare att utveckla vår kod. Serial Terminal (som PuTTY eller Tera Term):
Vi använder en seriell terminal för att kommunicera med vårt system. Detta gör att vi kan skicka kommandon till mikrokontrollern och se resultatet i realtid.

Protokoll och hur de används

I det här projektet använder vi ett antal olika protokoll för att kommunicera mellan olika delar av systemet. USART (Universal Synchronous/Asynchronous Receiver/Transmitter): Detta är protokollet vi använder för att kommunicera mellan vår dator och mikrokontrollern.
Vi skickar kommandon via detta gränssnitt till mikrokontrollern, som sedan utför dessa kommandon och skickar tillbaka ett svar.

GPIO (General Purpose Input/Output): Detta är protokollet vi använder för att kontrollera LED-lamporna. Varje LED är ansluten till en GPIO-pinne på mikrokontrollern, och vi kan styra om pinnen ger ut en hög eller låg signal för att tända eller släcka LED-lamporna.

DMA (Direct Memory Access): Detta är ett protokoll som mikrokontrollern använder för att snabbt och effektivt flytta data mellan olika delar av systemet utan att belasta CPU:n. I vårt projekt används DMA för att skicka data mellan mikrokontrollerns minne och USART2-gränssnittet.

I2C (Inter-Integrated Circuit):
Detta är ett protokoll som används för att kommunicera mellan mikrokontrollern och eventuella andra integrerade kretsar i systemet. I vårt projekt används det inte aktivt, men mikrokontrollern har stöd för det om det behövs i framtiden.

ADC (Analog-to-Digital Converter): Detta är ett protokoll som används för att konvertera analoga signaler (som de som kommer från sensorer) till digitala värden som mikrokontrollern kan bearbeta. I vårt projekt används det inte aktivt, men mikrokontrollern har stöd för det om det behövs i framtiden.

Sammanfattning

Vi dra den slutsatsen att projektet öppnar upp för möjligheter att kunna bygga vidare, tack vare hur vi valt att bygga vår kod-
Visserligen kommer vi behöva göra vissa tillägg i exempelvis LED.h (som att skapa nya definitioner om vi vill lägga till flera LED) och aktivera flera pinnar för vår mikrokontroller och registrera dessa