Rapport

Projekt skapat utav:

Rene Ahumada – Rene.ahumada@yh.nackademin.se

Innehållsförteckning

Introduktion

I denna rapport så förklarar jag utvecklingen av en LED-hanteringsfunktion för en STM32-mikrokontroller.

Syftet med projektet var att styra olika LED-lampor med olika färger och tillstånd genom att använda GPIO-pinnar och UART-kommunikation.

Förklaring

Projektet består av sex filer:

Led.cpp, Led.h, uart.h, uart.cpp, main.cpp och stm32f4xx.h.

Vi börjar med att förklara vad alla filer gör:

Led.cpp:

Här implementeras medlemsfunktionerna för LED-klassen.

I konstruktorn ”Led::Led()” initialiseras LED-lamporna med den färgen och statusen som angavs.

Färg och status bestämmer vilken port läge som sätts och LED-pinnar kontrolleras och konfigureras.

Medlemsfunktionen ”Led::setState()” används för att ändra statusen för en LED. Beroende på statusen så sätts eller stängs LED-pinnen av.

Funktionen ”Led::getState()” skickar tillbaka statusen för en LED-lampa.

Led.h:

I led.h-filen definieras LED-klassen med attribut och medlemsfunktion. Attributen ”color” och ”state” representerar färgen och statusen för LED-lampan. Klassen har en konstruktor som får information om färgen och statusen som argument och initierar attributen.
Medlemsfunktionerna ”setState()” och ”getState()” används för att ändra och hämta statusen för led-lampan.

uart.h:

uart.h-filen innehåller deklarationen för funktionerna ”USART2_Init()” och ”test_setup()”.
Funktionen ”USART2_Init()” initierar UART-protokollet och alla beståndsdelar, inklusive konfigurering av klocksignaler och GPIO-portarna för UART-kommunikationen.

uart.cpp:

I uart.cpp-filen så implementeras funktionerna ”USART2_Init(), USART2_write() och USART2_read()”.
Funktionen ”USART2_Init()” initierar UART2 genom att aktivera klocktillgång och konfigurerar GPIO-pinnar för alternativt funktion. Baudhastigheten och överföringsparametrar sätts också för UART2.

Funktionen ”USART2_write()” används för att skicka data till terminalen via UART. Den väntar på att överföringsregistret ska bli tom och överför sedan dataregistret med den angivna datan.

Funktionen ”USART2_read()” används för att läsa data från UART-mottagaren. Den väntar på data som ska hämtas och skickar sedan den lästa data tillbaka.

main.cpp:

I denna fil så finns huvudfunktionen ”main()”, som hanterar LED-lamporna och UART-kommunikationen.
Först initieras UART-kommunikationen genom att anropa ”USART2_Init()”.
Sedan skapas tre LED-objekt: ”led1” &”led2” skapas som automatiska variabler medan ”led3” skapades som dynamiskt med hjälp av operator ”new”.
”Led1” har färg: RED och status ON. ”led2” har färg: BLUE och status ON. ”led3” har färg: YELLOW och status ON.
Sedan hämtas och lagras statusen för led1 i variabeln ”led1_state”. Efter det så ändras statusen för ”led1” till OFF genom att anropa ”led1.setState(OFF);
Slutligen tas minnet som allokerats för ”led3” bort genom att anropa ”delete led3;”.
Huvudfunktionen avslutas med en evig ”while(1){}” loop för att programmet ska fortsätta köra.

stm32f4xx.h:

Slutligen så används denna header fil för att Inkluderar alla definitioner, funktioner och initialiserings koden som krävs för att interagera med STM32F4xx-mikrokontrollern och periferienheten.

Detta möjliggör styrning och konfigurering av alla aspekter hos enheten.

Sammanfattning

Genom att kombinera STM32 mikrokontroller och UART-kommunikation erbjuder detta projekt en bra och anpassningsbar lösning för att styra tre LED-lampor. Detta kan tillämpas inom flera olika områden som IoT, hemautomation, elektronikprototyper och belysningsdesign osv.
Slutligen så vill säga att jag lärde mig väldigt mycket och jag kommer fortsätta arbeta med stm32 plattformen.

Länk till Github

https://github.com/KabyIot/LED_stm32