Grunnleggande innføring i STM32

Eirik Haustveit

22. mars 2025

Institutt for datateknologi, elektroteknologi og realfag

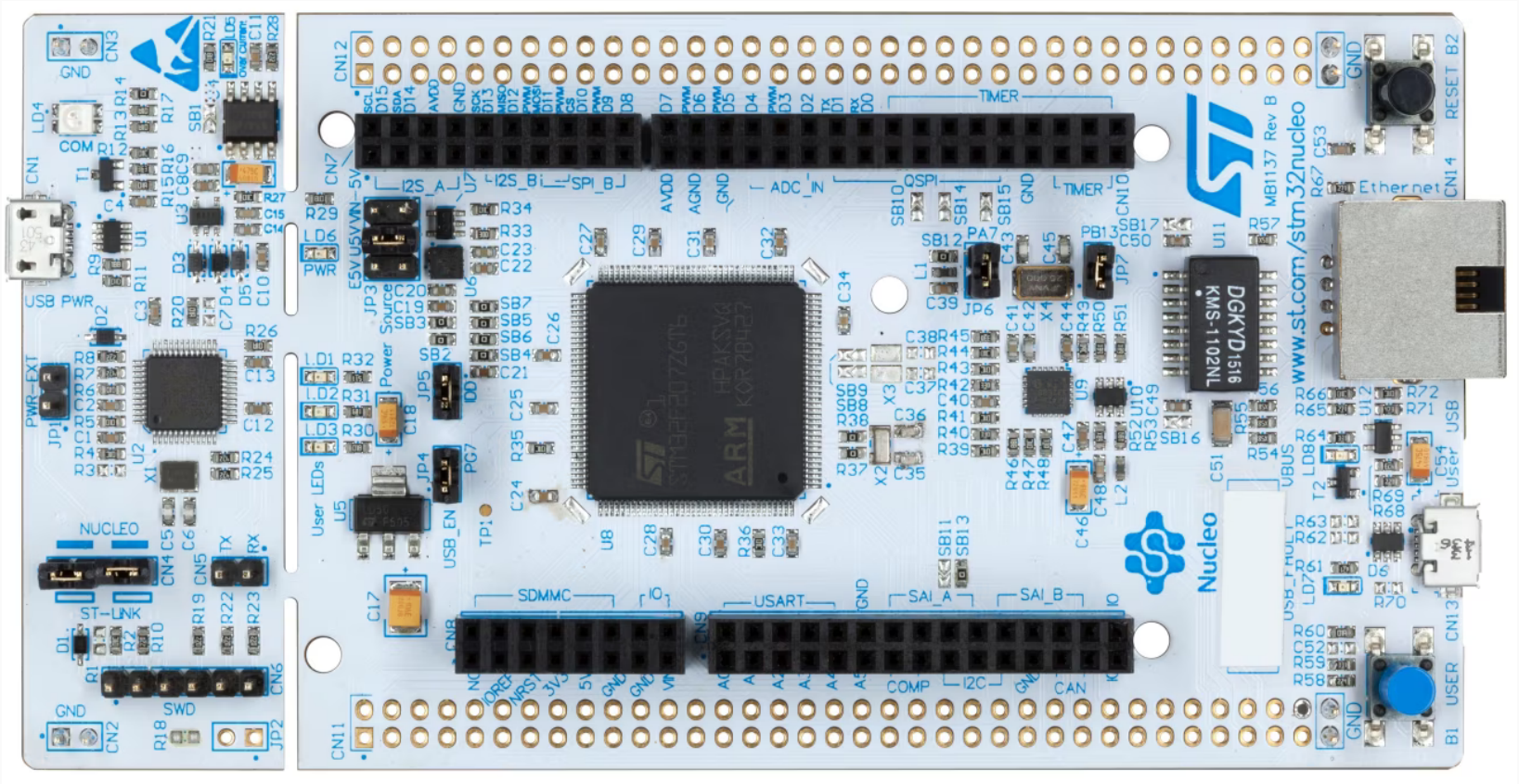
Plan for kurset

Plan

- Oppbygging til mikrokontrolleren og Nucleo-kortet
- Kretsskjema
- Klokketreet
- VSCode og PlatformIO
- Bruk av “Low layer”-biblioteket
- Bruk av CMSIS
- Bruk av libopencm3
- Bruk av Arduino-biblioteket
- Bruk av HAL-biblioteket og CubeMX (v. Gizem)
- U(S)ART
- USB
- Nettverkskommunikasjon

STM32

Introduksjon



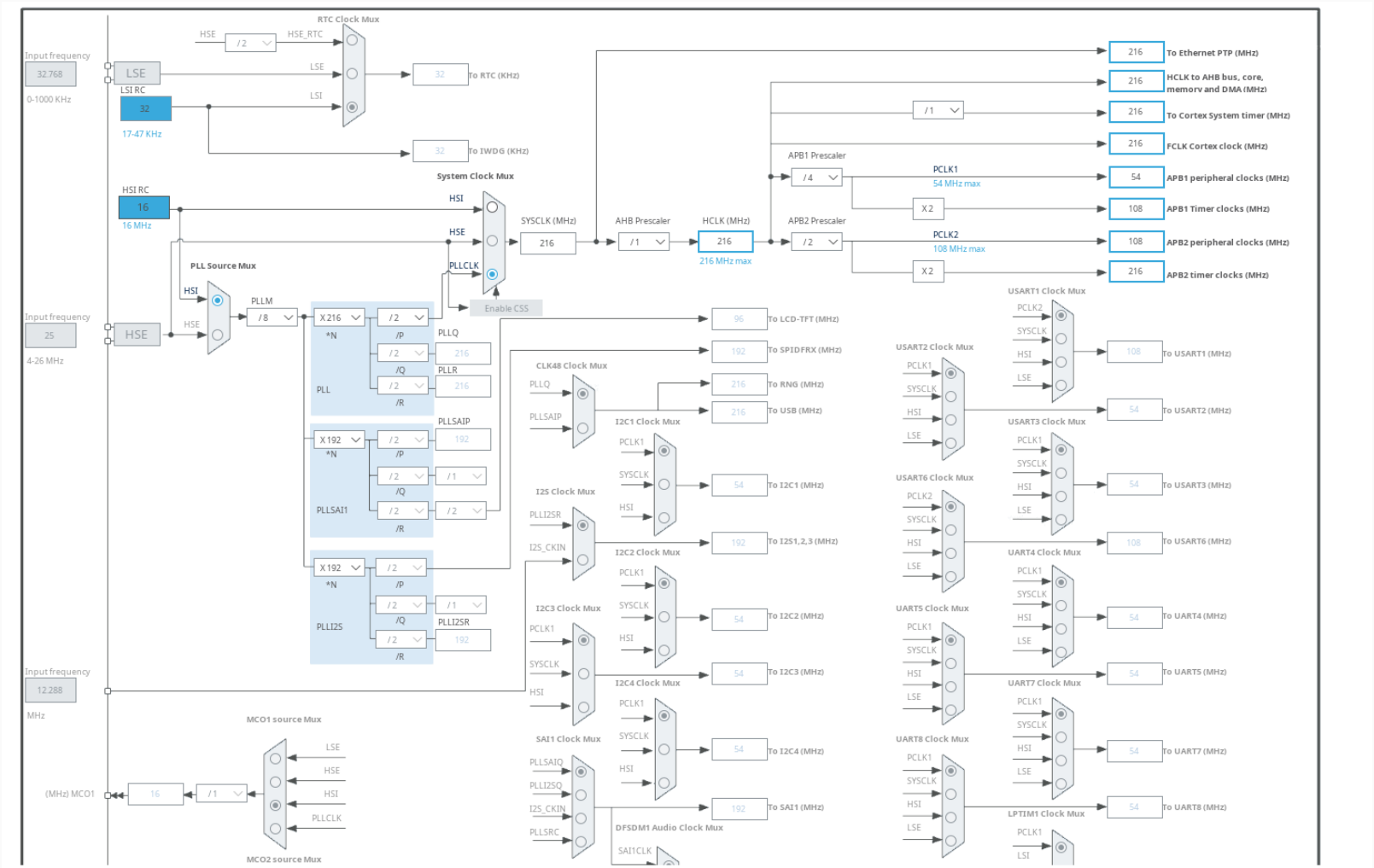
Figur 1: Nucleo-F767ZI

Introduksjon

Mikrokontrolleren har mange innebygde funksjonar i hardware:

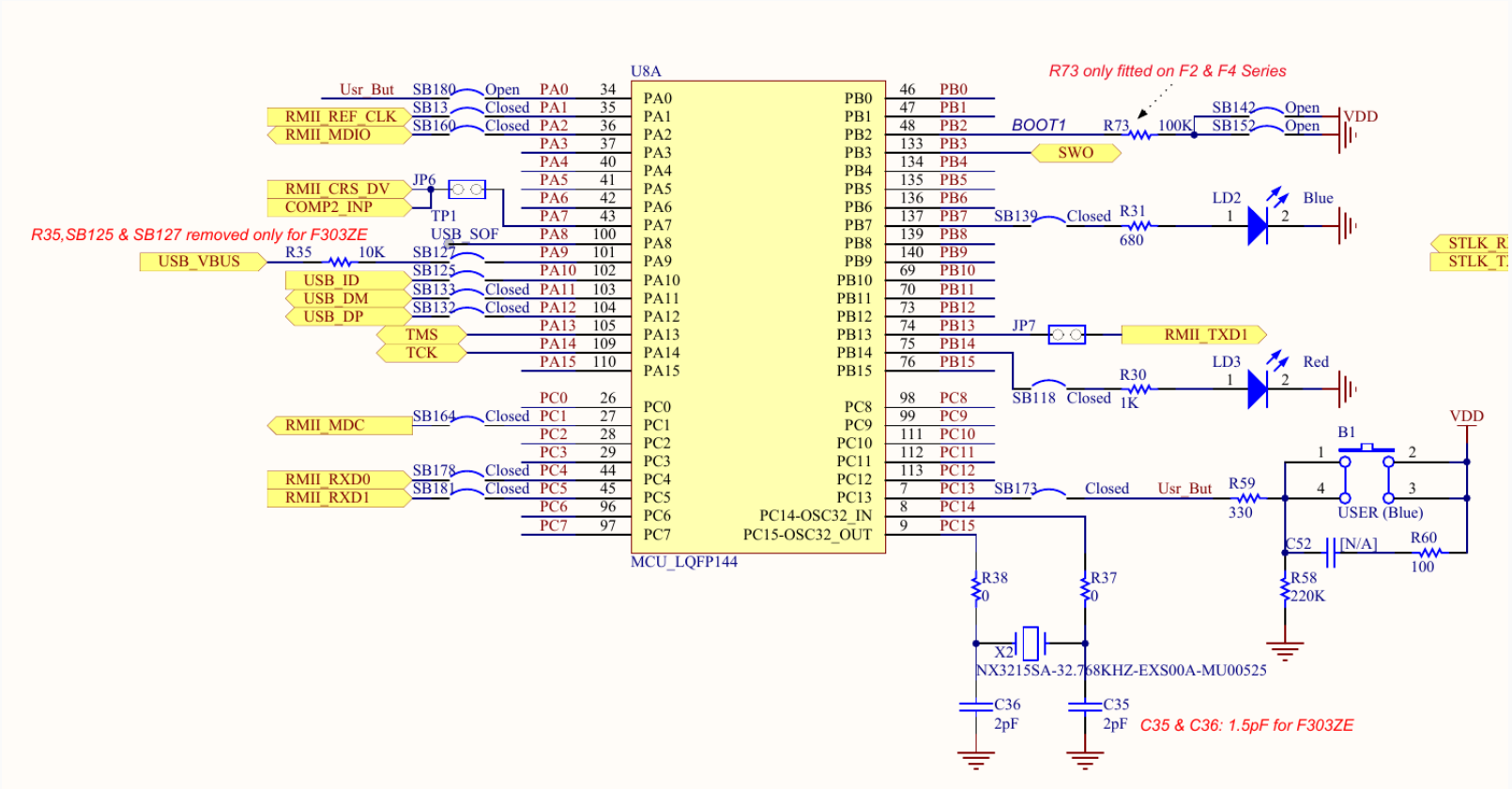
- Flyttalsstøtte (FPU)
- 12-bit Analog til digital omformar (ADC)
- 12-bit Digital til analog omformar (DAC)

Klokketreet



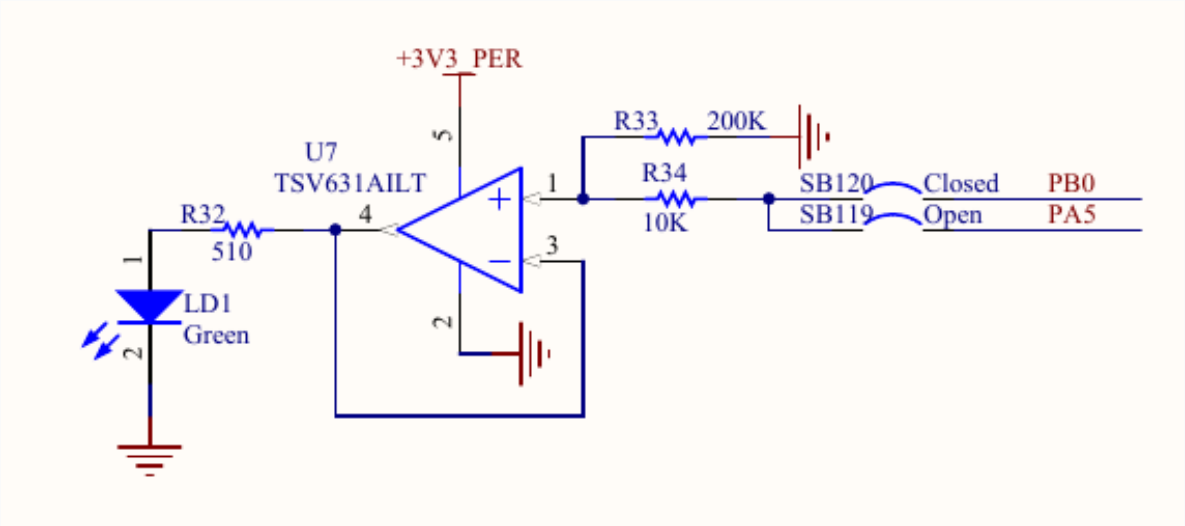
Figur 2: Utdrag frå klokketreet til STM32F767

Kretsskjema



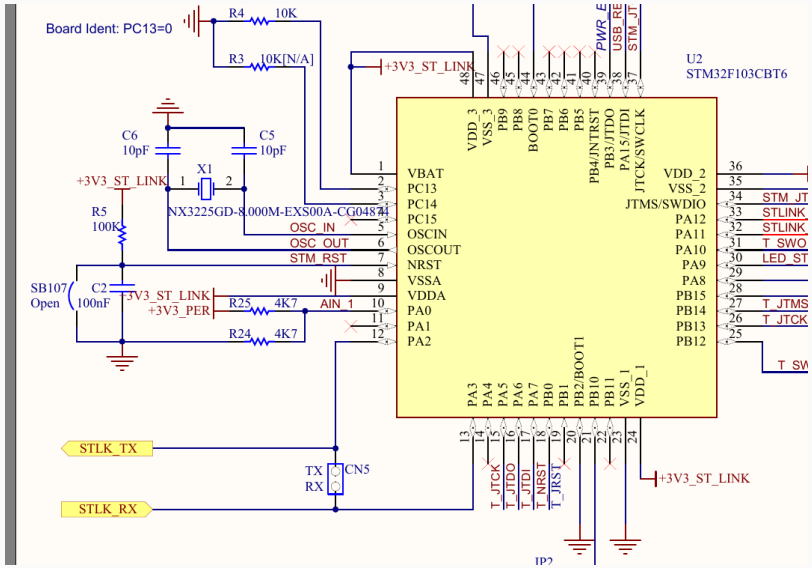
Figur 3: Tilkopling for trykknapp og lysdiodar

Kretsskjema

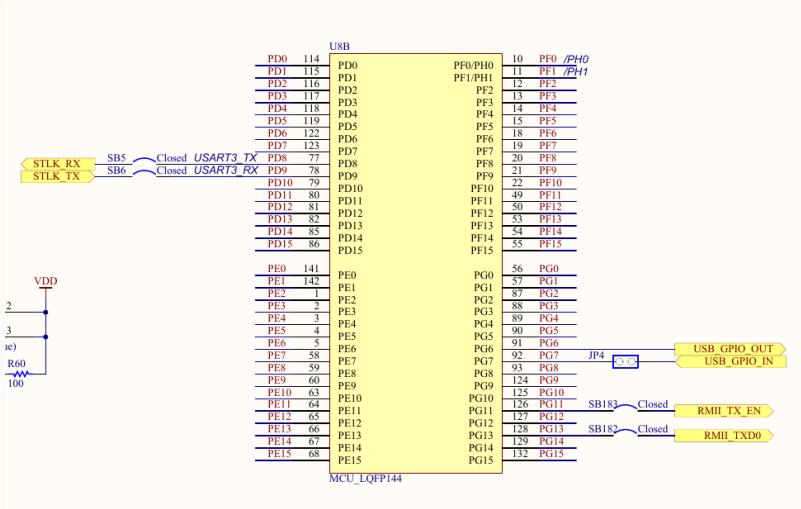


Figur 4: Lysdioden LD1 er av ein eller anna grunn kopla til ein forsterkar

Kretsskjema



Figur 5: Tilkopling for UART på STLink



Figur 6: Tilkopling for UART på mikrokontrolleren

VSCode og PlatformIO

...

Bruk av STM32Cube LL

Introduksjon til low-level biblioteket

STM32Cube tilbyr to ulike biblitotek for å snakka med dei ulike HW modulane i mikrokontrolleren.

- Low Layer (LL)
- Hardware Abstraction Layer (HAL)

LL-biblioteket tilbyr eit mindre abstrakt lavnivå API for direkte kommunikasjon med HW.

HAL og LL er heilt uavhengige. Både HAL og LL snakkar direkte med lavnivå I/O-register ved hjelp av peikarar til dei ulike minneadressene.

STM32Cube LL

```
#include <stm32f7xx_ll_gpio.h>
#include <stm32f7xx_ll_bus.h>
#include <stm32f7xx_ll_utils.h>

#define USER_LED_PORT           GPIOB
#define USER_LED1_PIN           LL_GPIO_PIN_0
#define USER_LED2_PIN           LL_GPIO_PIN_7
#define USER_LED3_PIN           LL_GPIO_PIN_14

void init_gpio(void) {

LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOB);

LL_GPIO_SetPinMode(USER_LED_PORT, USER_LED1_PIN,
        LL_GPIO_MODE_OUTPUT);
LL_GPIO_SetPinOutputType(USER_LED_PORT, USER_LED1_PIN,
        LL_GPIO_OUTPUT_PUSHPULL);
}
```

STM32Cube LL

```
int main(void) {
        init_gpio();

        SystemCoreClockUpdate();
        SysTick_Config(SystemCoreClock / 1000);

        while(1){

                LL_GPIO_TogglePin(USER_LED_PORT, USER_LED1_PIN);
                LL_mDelay(500);
        }
}
```

Bruk av CMSIS

Introduksjon til CMSIS

CMSIS (Common Microcontroller Software Interface Standard) er eit leverandøruavhengig abstraksjonslag for Cortex-M baserte mikrokontrollerar.

CMSIS

```
void gpio_pin_config(){
RCC->AHB1ENR |= RCC_AHB1ENR_GPIOBEN;

// 0b00 = input. 0b01 = output. 0b10 = alternate. 0b11 = analog.
GPIOB->MODER |= GPIO_MODER_MODER7_0;

// 0 = push-pull. 1 = open-drain.
GPIOB->OTYPER &= ~(GPIO_OTYPER_OT7);

// 0b00 = low. 0b01 = medium. 0b10 = high. 0b11 = very high.
GPIOB->OSPEEDR |= (GPIO_OSPEEDER_OSPEEDR7_0 | GPIO_OSPEEDER_OSPEEDR7_1

// 0b00 = no. 0b01 = pull-up. 0b10 = pull-down. 0b11 = reserved.
GPIOB->PUPDR &= ~(GPIO_PUPDR_PUPDR7_0 | GPIO_PUPDR_PUPDR7_1);

}
```

CMSIS

```
int main(){

        gpio_pin_config();
        SysTick_Config(SystemCoreClock / 80);

        for(;;){ //ever

                // Set pin 7
                GPIOB->BSRR |= GPIO_BSRR_BS7;

                delay_ms(100);

                // Reset pin 7
                GPIOB->BSRR |= GPIO_BSRR_BR7;

                delay_ms(100);
        }
}
```

Bruk av libopencm3

Introduksjon til libopencm3

libopencm3 er eit prosjekt for utvikling av eit ikkje-proprietært åpen kjeldekode bibliotek for ARM Cortex mikrokontrollerar.

Det har mellom anna støtte for:

- ST STM32
- Ti Tiva og Stellaris
- NXP LPC
- Atmel SAM3
- Energy Micro EFM32
- og mange fleire

libopencm3

```c
#include <libopencm3/stm32/rcc.h>
#include <libopencm3/stm32/gpio.h>

static void gpio_setup(void)
{
        /* Enable GPIOB clock. */
        rcc_periph_clock_enable(RCC_GPIOB);

        /* Set GPIO14 (in GPIO port B) to 'output push-pull'. */
        gpio_mode_setup(GPIOB, GPIO_MODE_OUTPUT,
                GPIO_PUPD_NONE, GPIO14);
}
```

libopencm3

```
int main(void)
{
        uint32_t i;

        gpio_setup();

        while (1) {
                /* Using API function gpio_toggle(): */
                gpio_toggle(GPIOB, GPIO14);        /* LED on/off */
                for (i = 0; i < 2000000; i++) /* Wait a bit. */
                __asm__("nop");

        }

        return 0;
}
```

Bruk av Arduino

Introduksjon til Arduno

Med Arduino i denne konteksten meiner eg *biblioteket* `<Arduino.h>`.

Eit problem med Arduino er at det har ein veldig forenkla API, som ikkje gir brukaren full kontroll på (eller forståelse for) eigenskapane til dei HW modulane ein nyttar. Til dømes har alle dei digitale utgangane på ein STM32 mikrokontroller ei innstilling for hastighet og push-pull vs open-drain, men `pinMode(USER_LED_1, OUTPUT);` har ikkje mulighet for å endra på dei innstillingane.

Arduino

```
#include <Arduino.h>

#define USER_LED_1 LED_BUILTIN
#define USER_LED_2 7
#define USER_LED_3 14

void setup() {
        pinMode(USER_LED_1, OUTPUT);
}

void loop() {
        digitalWrite(USER_LED_1, HIGH);
        delay(100);
        digitalWrite(USER_LED_1, LOW);
        delay(100);
}
```

Arduino og LL

Arduino nyttar LL biblioteket internt (når du er på ein STM32), og det er derfor enkelt å kombinera kode frå begge rammeverk:

```
#include <Arduino.h>

#include <stm32f7xx_ll_gpio.h>
#include <stm32f7xx_ll_bus.h>

#define USER_LED_PORT           GPIOB
#define USER_LED1_PIN           LL_GPIO_PIN_0

void setup() {

LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOB);

LL_GPIO_SetPinMode(USER_LED_PORT, USER_LED1_PIN,
        LL_GPIO_MODE_OUTPUT);
LL_GPIO_SetPinOutputType(USER_LED_PORT, USER_LED1_PIN,
        LL_GPIO_OUTPUT_PUSHPULL);
}
```

Arduino og LL

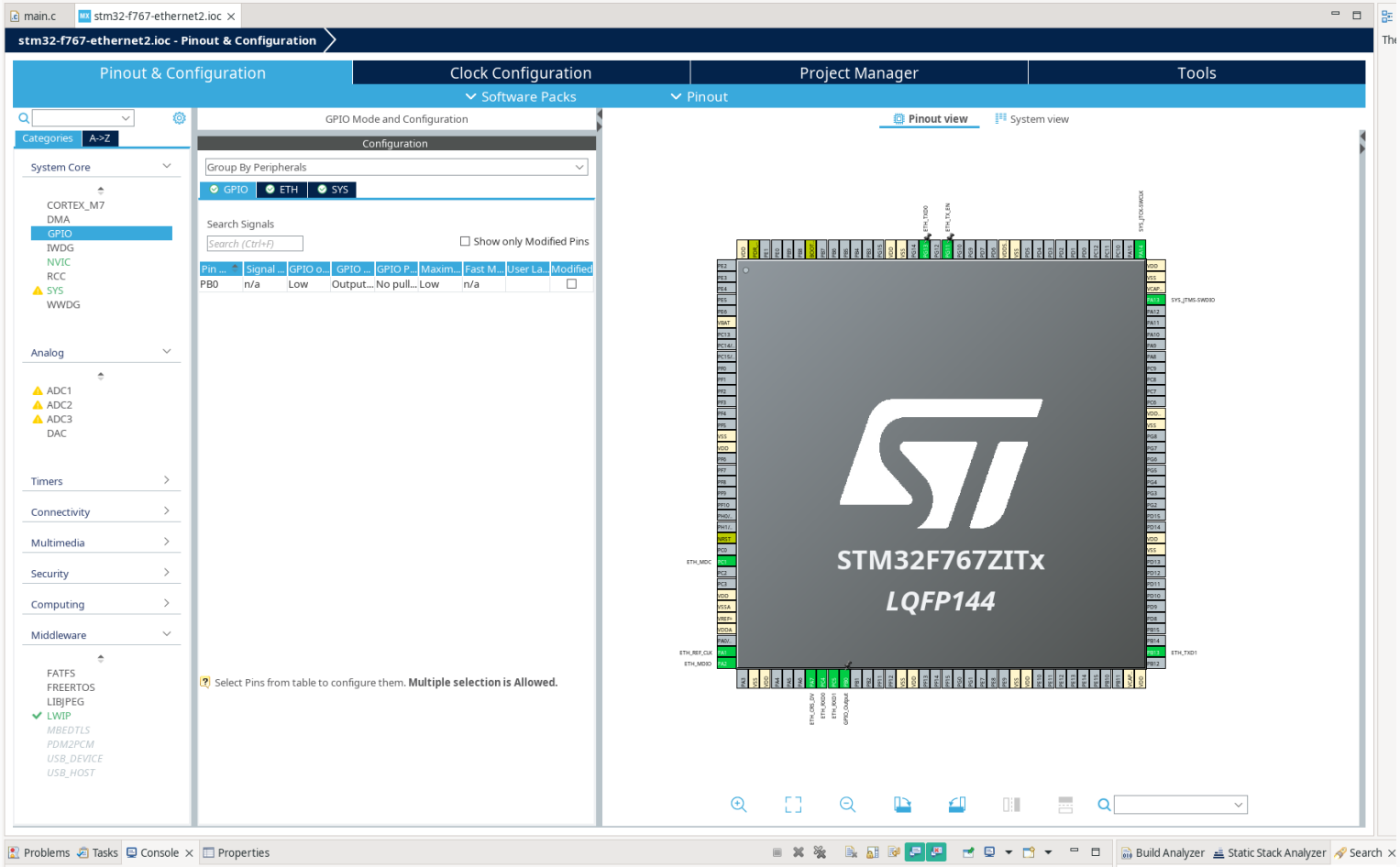
```
void loop() {

        LL_GPIO_TogglePin(USER_LED_PORT, USER_LED1_PIN);
        delay(100);
}
```

For å kunna utnytta all funksjonalitet i mikrokontrolleren er ein avhengig av eit kraftigare bibliotek enn Arduino, og dette kan i slike situasjonar vera ei god nødløysing.

STM32CubeMX og STM32CubeIDE

Introduksjon



Figur 7: Skjermbilete av pin-konfigureringa i STM32CubeMX

Introduksjon

STM32CubeIDE er basert på Eclipse. Det er tungvindt å bruka, treigt og ustabilt. Det kan derimot vera nyttig å autogenerera delar av oppsettkoden for dei ulike pinnane på mikrokontrolleren ved hjelp av STM32CubeMX som kan nyttast som eit “stand alone” program, eller integrert i STM32CubeIDE.

USART

STM32F767 har 4 USART modular i HW. Av dei er *USART3* kopla til STLink som igjen presenterer den som ein serieport via USB.

USART

```
LL_USART_InitTypeDef USART_InitStruct = {0};
LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

/* Peripheral clock enable */
LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_USART3);
LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOD);

/*
* USART3 GPIO Configuration
*
* PD8   ------> USART3_TX
* PD9   ------> USART3_RX
*/
GPIO_InitStruct.Pin = LL_GPIO_PIN_8 | LL_GPIO_PIN_9;
GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_VERY_HIGH;
GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
GPIO_InitStruct.Pull = LL_GPIO_PULL_UP;
GPIO_InitStruct.Alternate = LL_GPIO_AF_7;
LL_GPIO_Init(GPIOD, &GPIO_InitStruct);
```

USART

```
/* USART configuration */
USART_InitStruct.BaudRate = 115200;
USART_InitStruct.DataWidth = LL_USART_DATAWIDTH_8B;
USART_InitStruct.StopBits = LL_USART_STOPBITS_1;
USART_InitStruct.Parity = LL_USART_PARITY_NONE;
USART_InitStruct.TransferDirection = LL_USART_DIRECTION_TX_RX;
USART_InitStruct.HardwareFlowControl = LL_USART_HWCONTROL_NONE;
USART_InitStruct.OverSampling = LL_USART_OVERSAMPLING_16;
LL_USART_Init(USART3, &USART_InitStruct);
LL_USART_ConfigAsyncMode(USART3);

/* Enable USART */
LL_USART_Enable(USART3);
```

USART

```
char demo_string[] = "Dette er ein test av Nucleo-L767ZI\r\n";

while(1) {

        for(uint8_t i = 0; i < strlen(demo_string); i++){
                while(!LL_USART_IsActiveFlag_TC(USART3));
                LL_USART_TransmitData8(USART3, demo_string[i]);
        }

        LL_GPIO_TogglePin(USER_LED_PORT, USER_LED1_PIN);
        LL_mDelay(1800);
};
```

USB

Introduksjon til USB for STM32

STM32F767 (og mange av dei andre mikrokontrollerane i STM32 serien) har innebygd støtte for USB. Det betyr at mikrokontrolleren kan levera mykje meir enn ein serieport via USB.

- Fleire serieportar via samme USB kabel
- Mus eller tastatur (eller begge delar)
- Lydkort
- USB-minne (minnepinne)
- ..og veldig mykje meir

TinyUSB (`https://github.com/hathach/tinyusb`) er eit veldig bra bibliotek for USB på STM32.

USB

```c
int main(void) {
        board_init();

        // init device stack on configured roothub port
        tusb_rhport_init_t dev_init = {
                .role = TUSB_ROLE_DEVICE,
                .speed = TUSB_SPEED_AUTO
        };
        tusb_init(BOARD_TUD_RHPORT, &dev_init);

        if (board_init_after_tusb) {
                board_init_after_tusb();
        }

        while (1) {
                tud_task(); // tinyusb device task
                led_blinking_task();

                cdc_task();
        }
}
```

Ethernet

Introduksjon til Ethernet og UDP,TCP/IP for STM32

STM32F767 har støtte for kabla ethernet. Det er fleire ulike bibliotek tilgjengelig for nettverksapplikasjonar:

- lwIP (`https://savannah.nongnu.org/projects/lwip/`)
- Mongoose (`https://github.com/cesanta/mongoose`)

Biblioteka kan nyttast "stand alone" eller saman med eit sanntidsoperativsystem som til dømes FreeRTOS (`https://www.freertos.org/`).

Ein kan setta opp mikrokontrolleren for enkel demonstrasjon av TCP og UDP, den kan fungera som ein sensor som leverer data på MQTT, ein kan setta opp ein fullverdig HTTP(S) webserver, og ein heil haug av andre applikasjonar.

lwIP

STM32CubeMX har støtte for å oppretta og konfigurera eit lwIP-basert ethernetprosjekt.

lwIP

```c
int main(void)
```

Referansar