Co-funded by the
Erasmus+ Programme
of the European Union

UbiLAB

UbiLAB

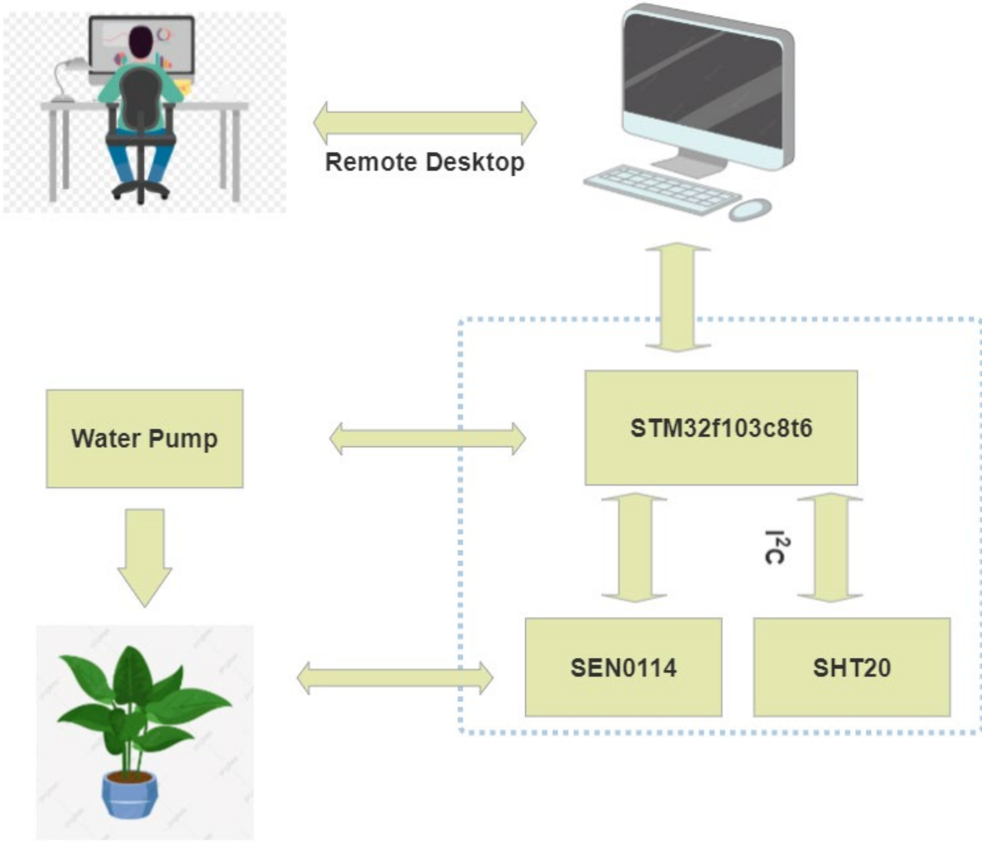
Virtual Hardware Laboratories examples (Embedded Systems Laboratory)

Host: Ss. Cyril and Methodius University in Skopje

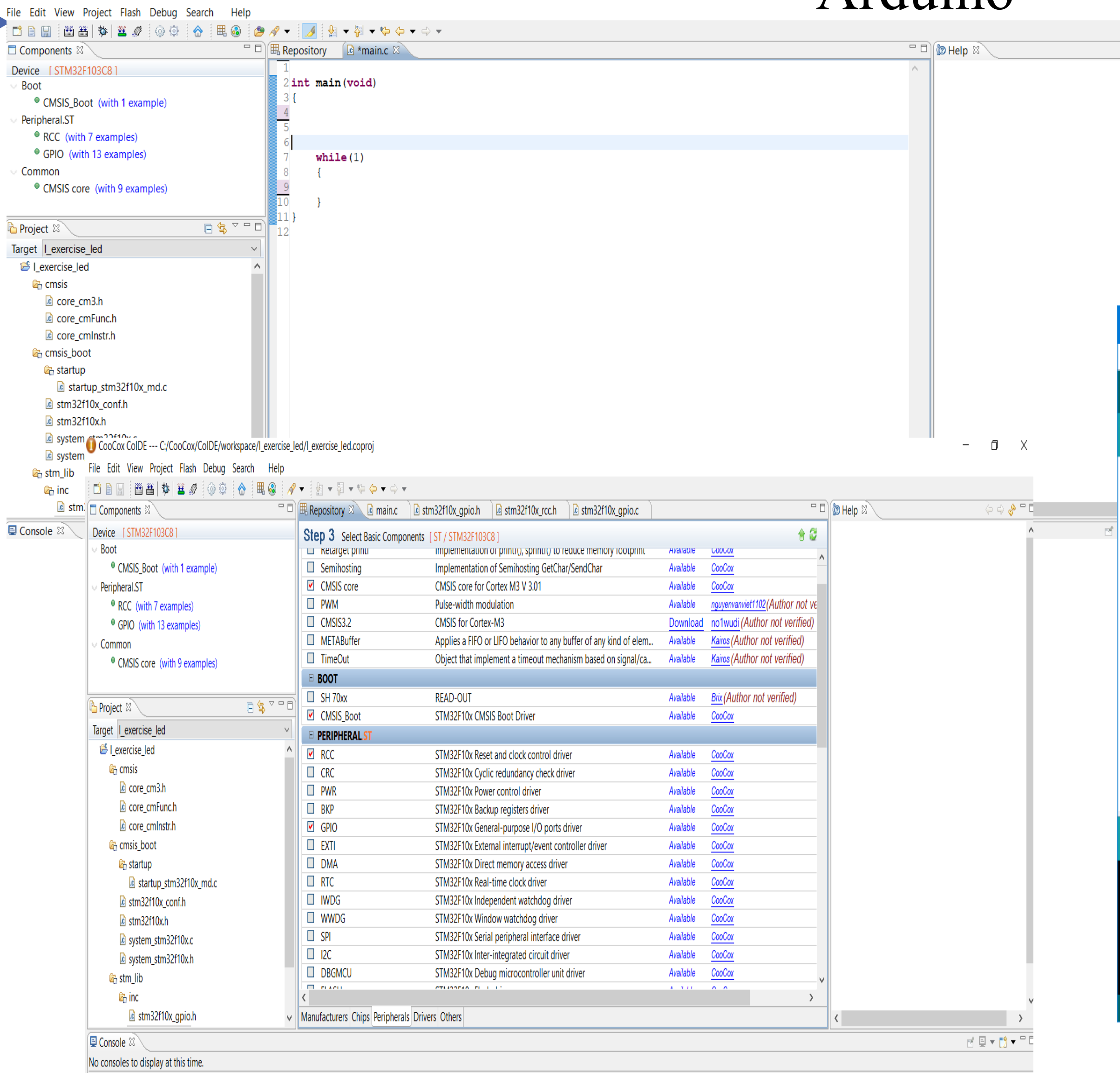
01.02.2023

- The course consists of five laboratory exercises each designed to be done in a single session. The topics that will be covered are:
  - General purpose I/O ports
  - Timers
  - ADC
  - Serial communication protocols (I^2C and UART/USART)

- The ultimate purpose of this project is to provide students with practical introduction to the field of embedded systems. Additionally by combining the five exercises an interesting and practical autonomous irrigation system can be created. The irrigation system consists of:
  - STM32f103c8t6 microcontroller (processing information)
  - SEN0114 soil moisture sensor (measure soil moisture)
  - SHT20 temperature and humidity sensor (measure ambient temperature and air humidity)



Development Environment –CooCox Coide and Arduino

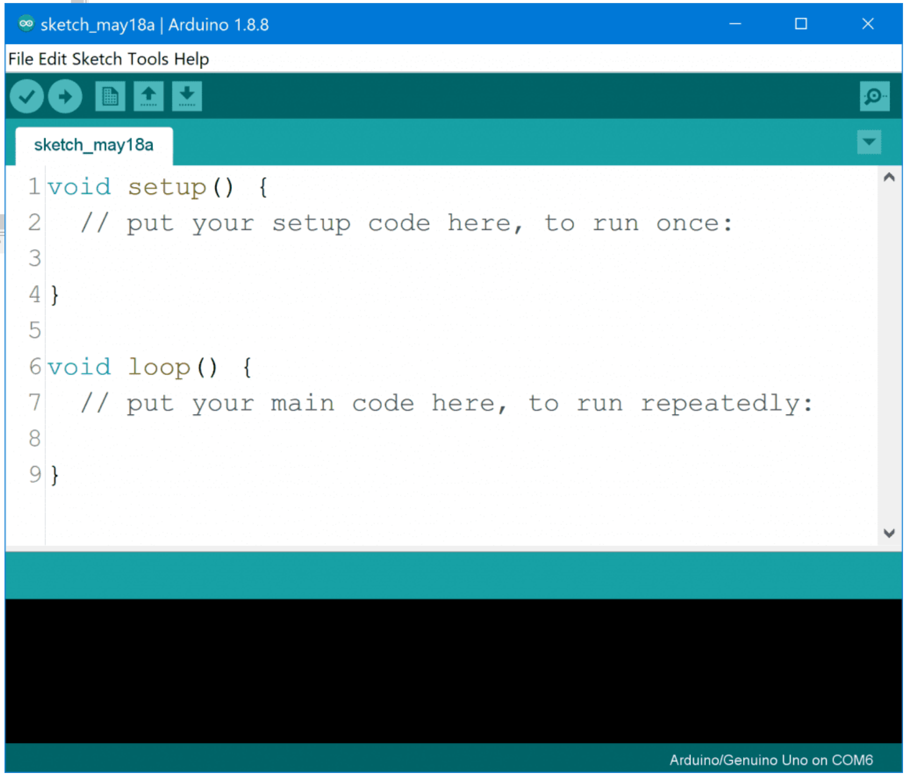


```
int main(void)
{


    while(1)
    {

    }
}
```

Step 3 Select Basic Components [ST / STM32F103C8]

□	Retarget printf	Implementation of printf(), sprintf() to reduce memory footprint	Available	CooCox
□	Semihosting	Implementation of Semihosting GetChar/SendChar	Available	CooCox
☑	CMSIS core	CMSIS core for Cortex M3 V 3.01	Available	CooCox
□	PWM	Pulse-width modulation	Available	nguyenvanviet1102(Author not ve
□	CMSIS3.2	CMSIS for Cortex-M3	Download	no1wudi (Author not verified)
□	METABuffer	Applies a FIFO or LIFO behavior to any buffer of any kind of elem...	Available	Kairos (Author not verified)
□	TimeOut	Object that implement a timeout mechanism based on signal/ca...	Available	Kairos (Author not verified)
⊟	**BOOT**			
□	SH 70xx	READ-OUT	Available	Brix (Author not verified)
☑	CMSIS_Boot	STM32F10x CMSIS Boot Driver	Available	CooCox
⊟	**PERIPHERAL.ST**			
☑	RCC	STM32F10x Reset and clock control driver	Available	CooCox
□	CRC	STM32F10x Cyclic redundancy check driver	Available	CooCox
□	PWR	STM32F10x Power control driver	Available	CooCox
□	BKP	STM32F10x Backup registers driver	Available	CooCox
☑	GPIO	STM32F10x General-purpose I/O ports driver	Available	CooCox
□	EXTI	STM32F10x External interrupt/event controller driver	Available	CooCox
□	DMA	STM32F10x Direct memory access driver	Available	CooCox
□	RTC	STM32F10x Real-time clock driver	Available	CooCox
□	IWDG	STM32F10x Independent watchdog driver	Available	CooCox
□	WWDG	STM32F10x Window watchdog driver	Available	CooCox
□	SPI	STM32F10x Serial peripheral interface driver	Available	CooCox
□	I2C	STM32F10x Inter-integrated circuit driver	Available	CooCox
□	DBGMCU	STM32F10x Debug microcontroller unit driver	Available	CooCox

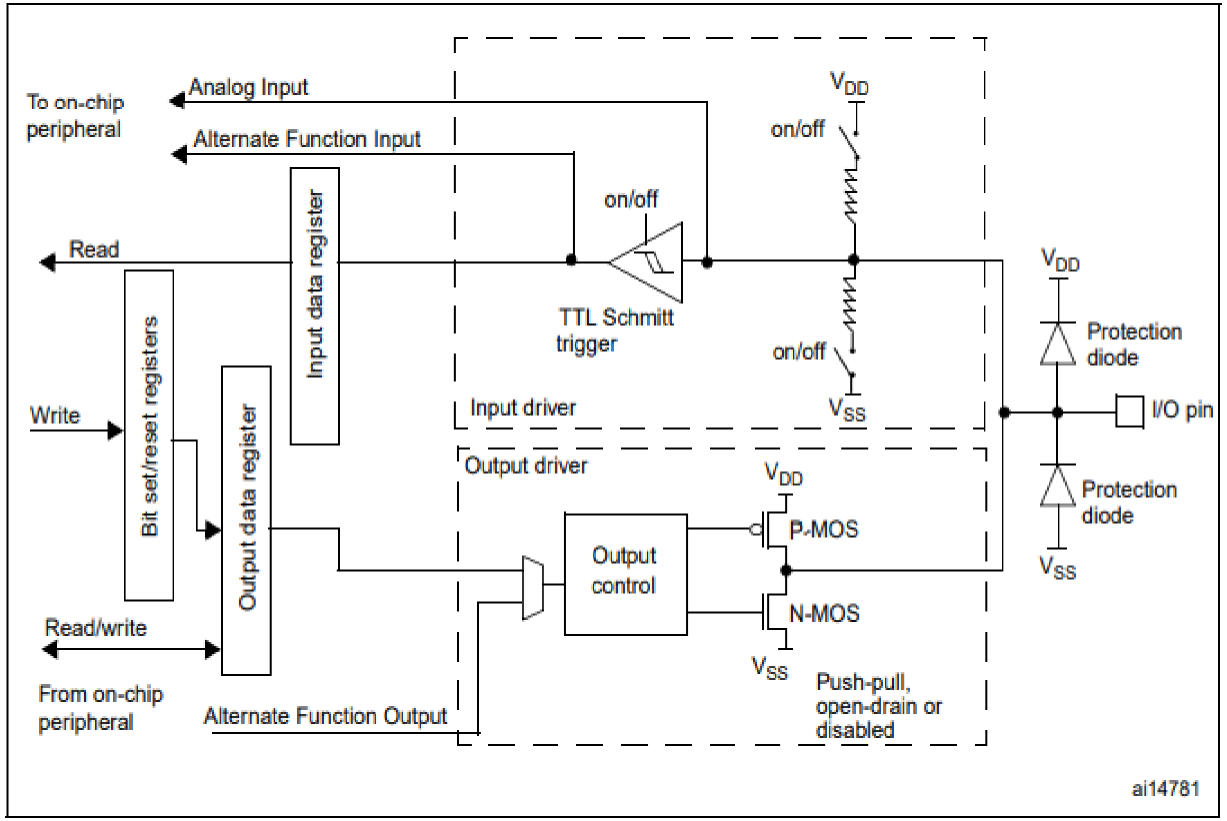
No consoles to display at this time.


sketch_may18a | Arduino 1.8.8
File Edit Sketch Tools Help
sketch_may18a
Arduino/Genuino Uno on COM6

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

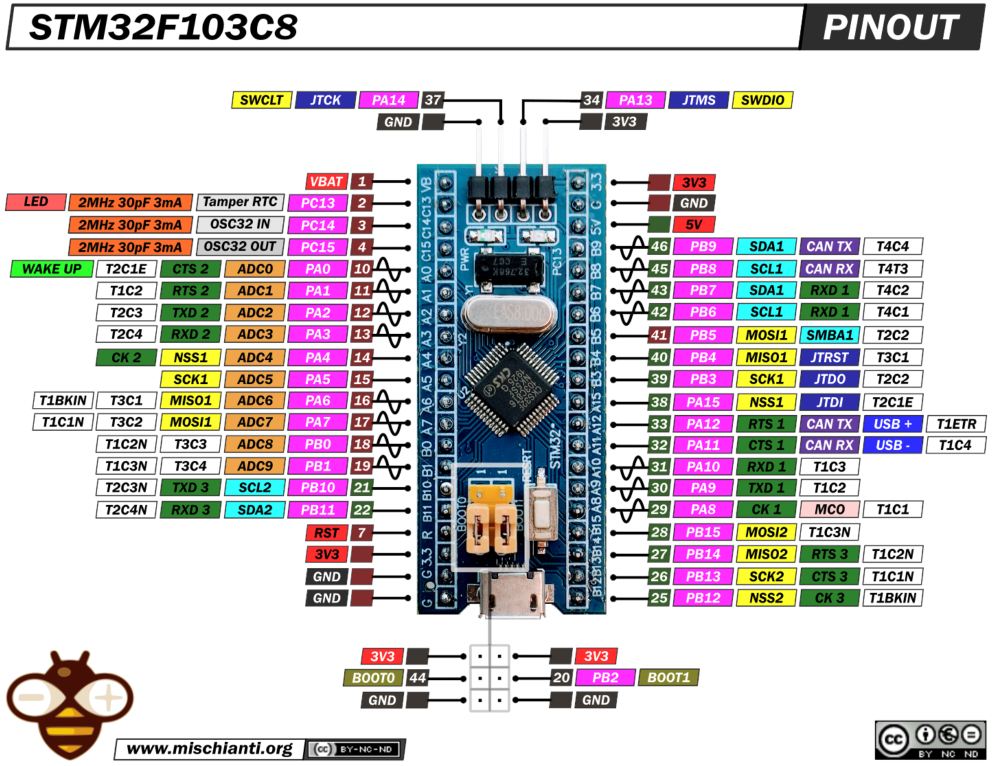
GPIO ports

- GPIO – General Purpose Input Output pins can be individually configured for specific purpose
- Relationship between microcontroller peripherals and devices
- GPIO functional modes:
  - Input floating
  - Input pull-up
  - Input push-up
  - Analog
  - Output open-drain
  - Output push-pull
  - Alternate function open-drain
  - Alternate function push-pull



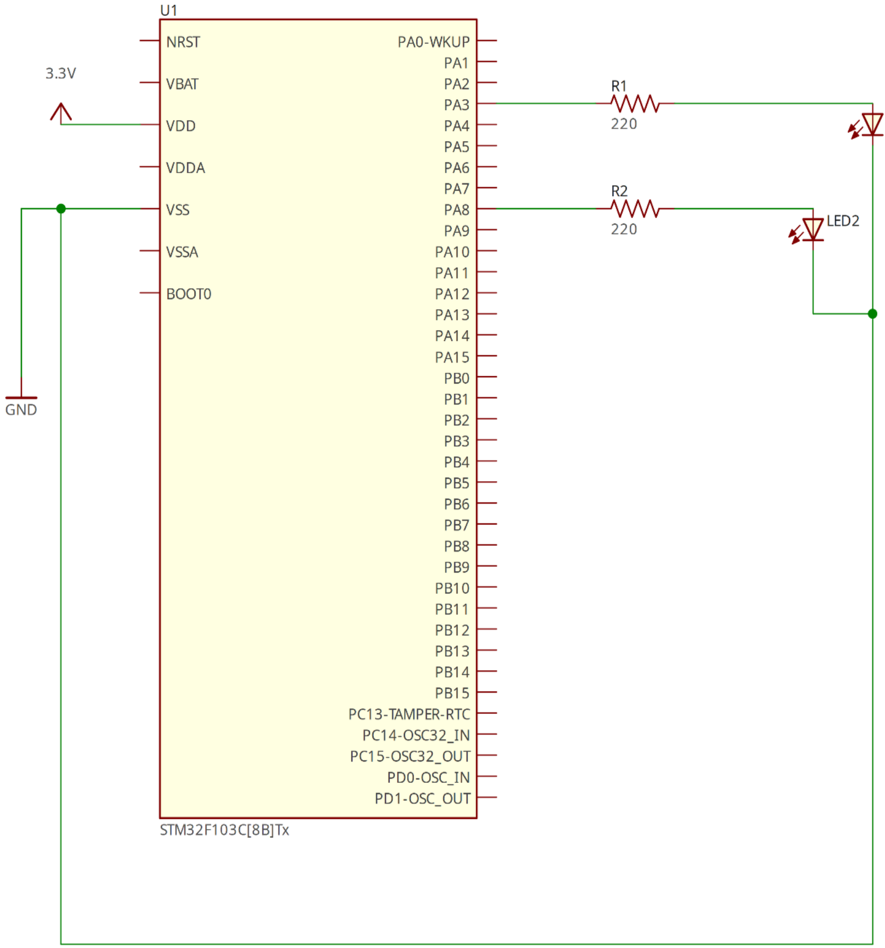
GPIO Configuration

- Enable peripheral clock
- Specify the GPIO pins to be configured
- Specify the speed for the selected pins
- Specify the operating mode for the selected pins
- Remap pin if alternate function is used

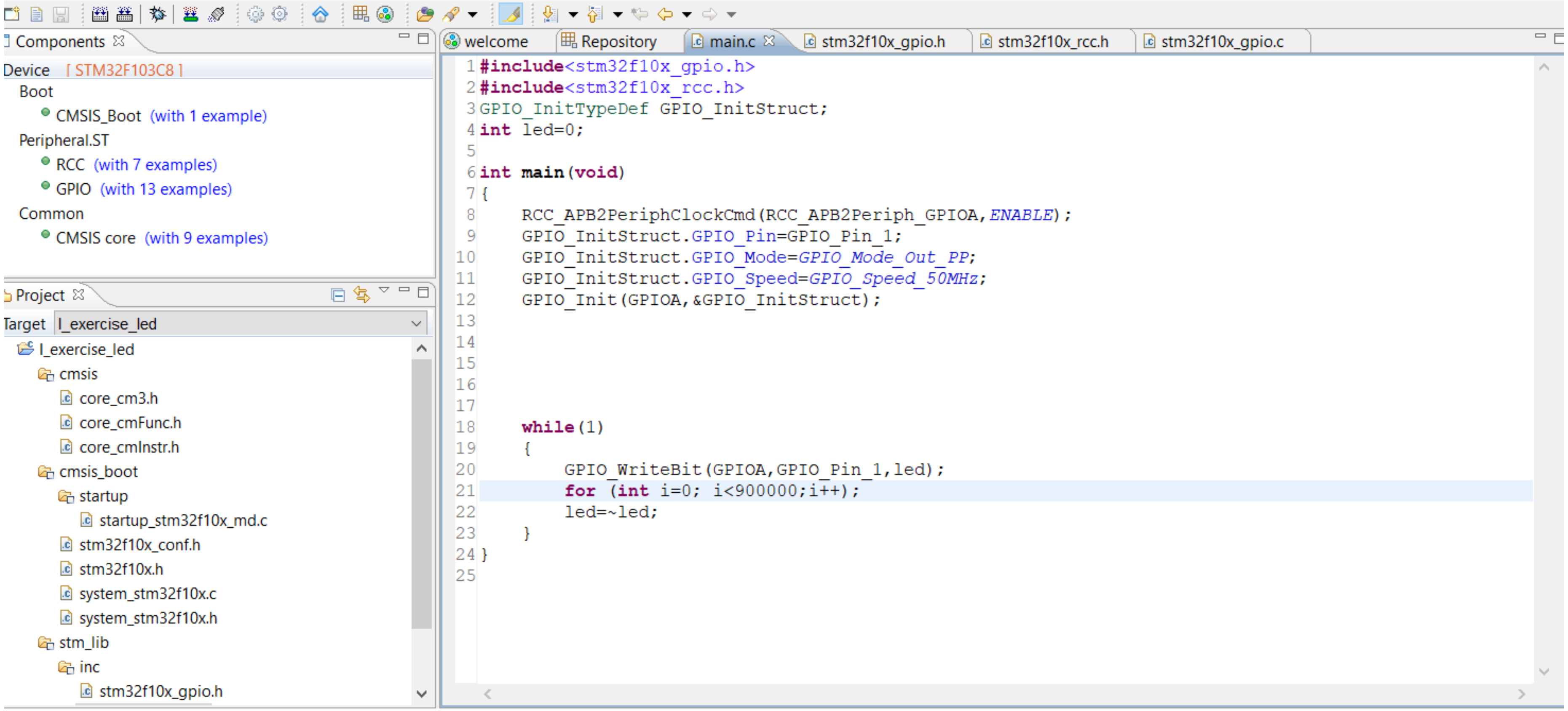


I exercise – Configuration the GPIO port

- The first exercise is a way of getting started and becoming familiar with the tools and environment, as well as understanding the general purpose I/O ports.
- It consists of a simple code that configures a selected I/O port which turns a LED diode connected through a 220 ohms resistor on or off dependent on the set logic level. In addition a second diode can be used to implement alternating operation.
- Additional exercise is to write code for traffic light where for loop is used for delay



Example with LED diode



Device [STM32F103C8]

Boot
- CMSIS_Boot (with 1 example)

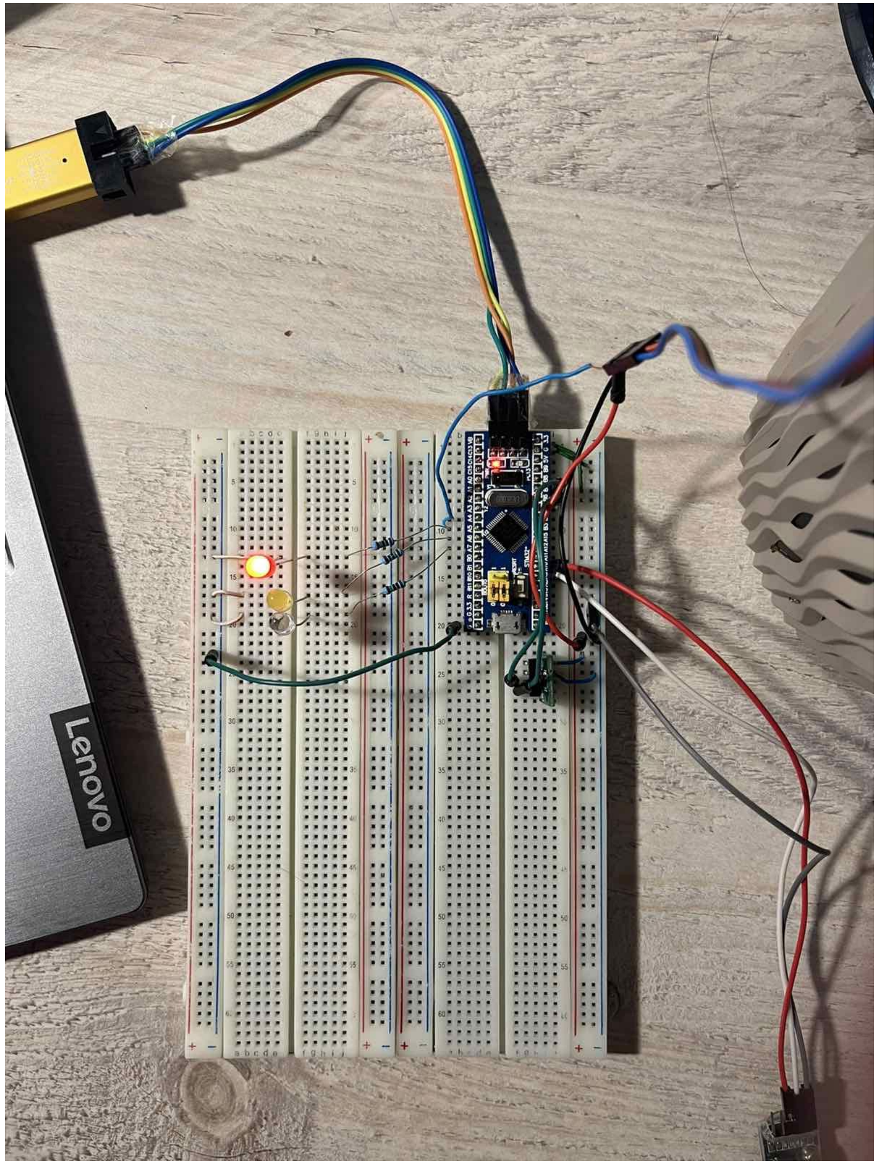
Peripheral.ST
- RCC (with 7 examples)
- GPIO (with 13 examples)

Common
- CMSIS core (with 9 examples)

Target l_exercise_led

```
l_exercise_led
  cmsis
    core_cm3.h
    core_cmFunc.h
    core_cmInstr.h
  cmsis_boot
    startup
      startup_stm32f10x_md.c
    stm32f10x_conf.h
    stm32f10x.h
    system_stm32f10x.c
    system_stm32f10x.h
  stm_lib
    inc
      stm32f10x_gpio.h
```

```c
#include<stm32f10x_gpio.h>
#include<stm32f10x_rcc.h>
GPIO_InitTypeDef GPIO_InitStruct;
int led=0;

int main(void)
{
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);
  GPIO_InitStruct.GPIO_Pin=GPIO_Pin_1;
  GPIO_InitStruct.GPIO_Mode=GPIO_Mode_Out_PP;
  GPIO_InitStruct.GPIO_Speed=GPIO_Speed_50MHz;
  GPIO_Init(GPIOA,&GPIO_InitStruct);





  while(1)
  {
    GPIO_WriteBit(GPIOA,GPIO_Pin_1,led);
    for (int i=0; i<900000;i++);
    led=~led;
  }
}

```

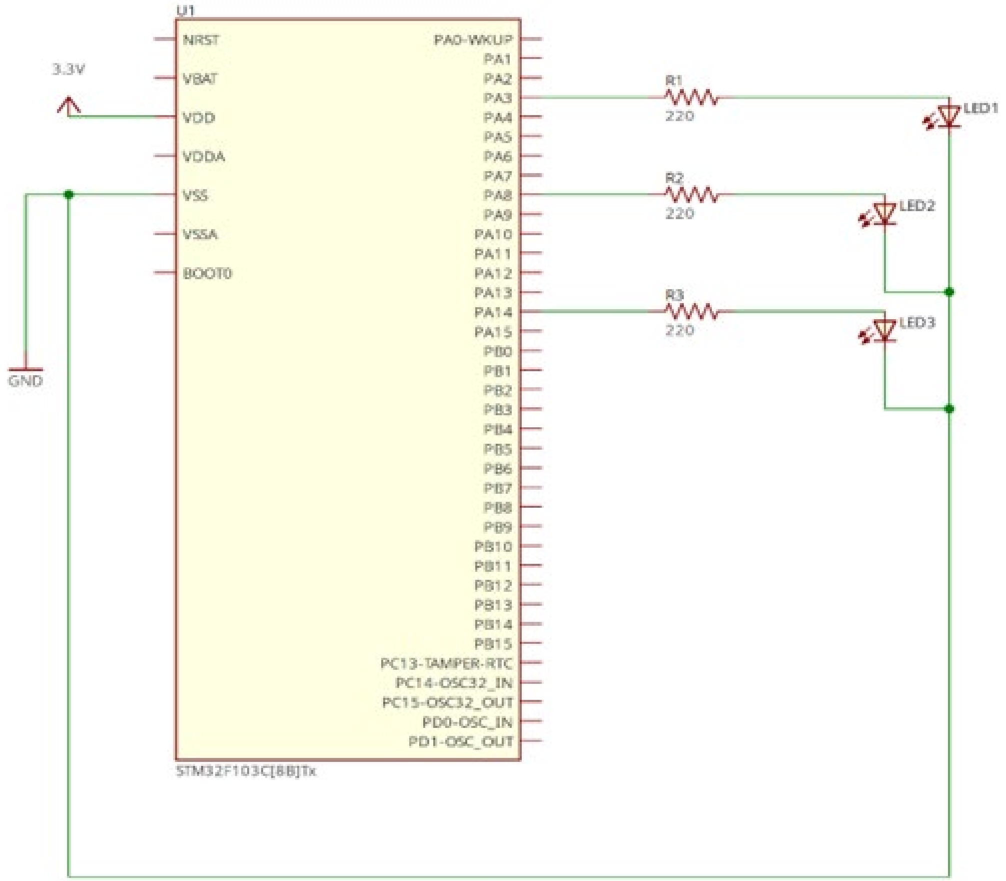


Basic Timers- TIM6 and TIM7

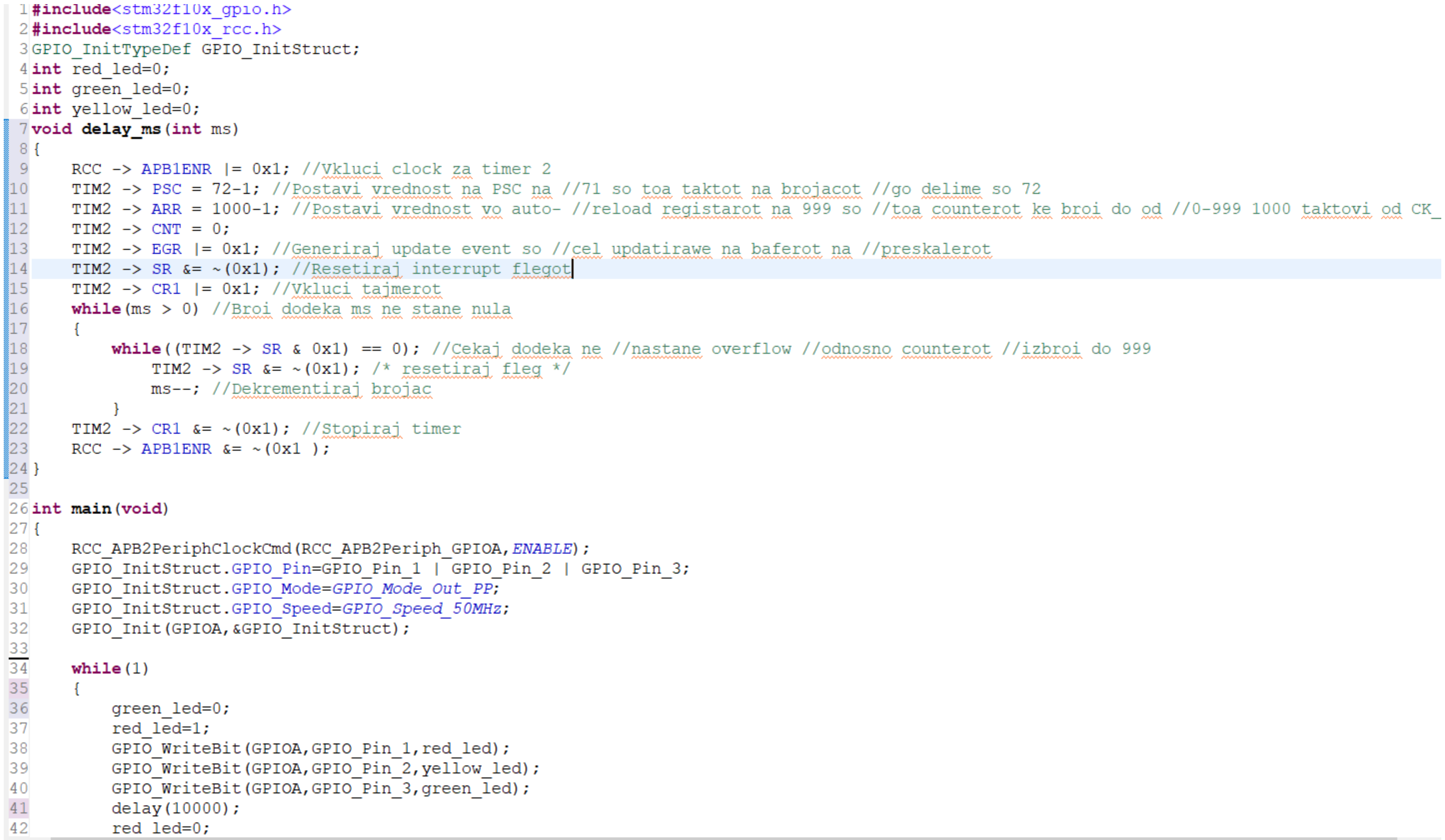
- TIM6 and TIM7 usually are used for time-base generation and to drive the digital-analog convertor
- Basic Timers consist of a 16-bit auto-reload counter driven by a programmable prescaler
- They are completely independent
- Main features are:
- 16-bit programmable prescaler used do divide the counter clock frequency by any factor between 1 and $2^{16} - 1$
- Synchronization circuit to trigger the DAC
- Interrupt on update event: counter overflow

II exercise – Pulse Width Modulation (PWM) and Delay

- This exercise begins with a brief overview of the operation of the integrated timers and prescalers.
- The STM microcontroller has to be programmed to use its internal RC oscillator as a clock signal. The students can write delay function using timers. Using delay can perform traffic light using three diode.
- For further practice students can generate PWM signal for intensity changes.
- Further, students can experiment with different oscillator as clock signal, different driving scheme and etc.)



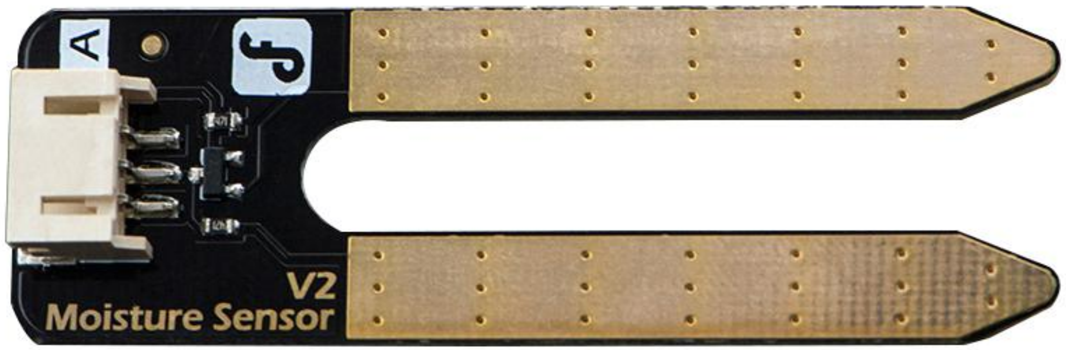
Example – Delay



```c
#include<stm32f10x_gpio.h>
#include<stm32f10x_rcc.h>
GPIO_InitTypeDef GPIO_InitStruct;
int red_led=0;
int green_led=0;
int yellow_led=0;
void delay_ms(int ms)
{
    RCC -> APB1ENR |= 0x1; //Vkluci clock za timer 2
    TIM2 -> PSC = 72-1; //Postavi vrednost na PSC na //71 so toa taktot na brojacot //go delime so 72
    TIM2 -> ARR = 1000-1; //Postavi vrednost vo auto- //reload registarot na 999 so //toa counterot ke broi do od //0-999 1000 taktovi od CK_
    TIM2 -> CNT = 0;
    TIM2 -> EGR |= 0x1; //Generiraj update event so //cel updatirawe na baferot na //preskalerot
    TIM2 -> SR &= ~(0x1); //Resetiraj interrupt flegot
    TIM2 -> CR1 |= 0x1; //Vkluci tajmerot
    while(ms > 0) //Broi dodeka ms ne stane nula
    {
        while((TIM2 -> SR & 0x1) == 0); //Cekaj dodeka ne //nastane overflow //odnosno counterot //izbroi do 999
            TIM2 -> SR &= ~(0x1); /* resetiraj fleg */
            ms--; //Dekrementiraj brojac
        }
    TIM2 -> CR1 &= ~(0x1); //Stopiraj timer
    RCC -> APB1ENR &= ~(0x1 );
}

int main(void)
{
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);
    GPIO_InitStruct.GPIO_Pin=GPIO_Pin_1 | GPIO_Pin_2 | GPIO_Pin_3;
    GPIO_InitStruct.GPIO_Mode=GPIO_Mode_Out_PP;
    GPIO_InitStruct.GPIO_Speed=GPIO_Speed_50MHz;
    GPIO_Init(GPIOA,&GPIO_InitStruct);

    while(1)
    {
        green_led=0;
        red_led=1;
        GPIO_WriteBit(GPIOA,GPIO_Pin_1,red_led);
        GPIO_WriteBit(GPIOA,GPIO_Pin_2,yellow_led);
        GPIO_WriteBit(GPIOA,GPIO_Pin_3,green_led);
        delay(10000);
        red_led=0;
```

III exercise – Conversion analog values to digital

- Exercise 3 gives a description of the operation of the soil moisture sensor and the process of A/D conversion.
- SEN0114 is a capacitive soil moisture sensor that conducts current through the soil its two probes and measures the soil resistivity.
- The goal of this exercise is to convert the measured values from analog to digital form and to store this information on the microcontroller memory.

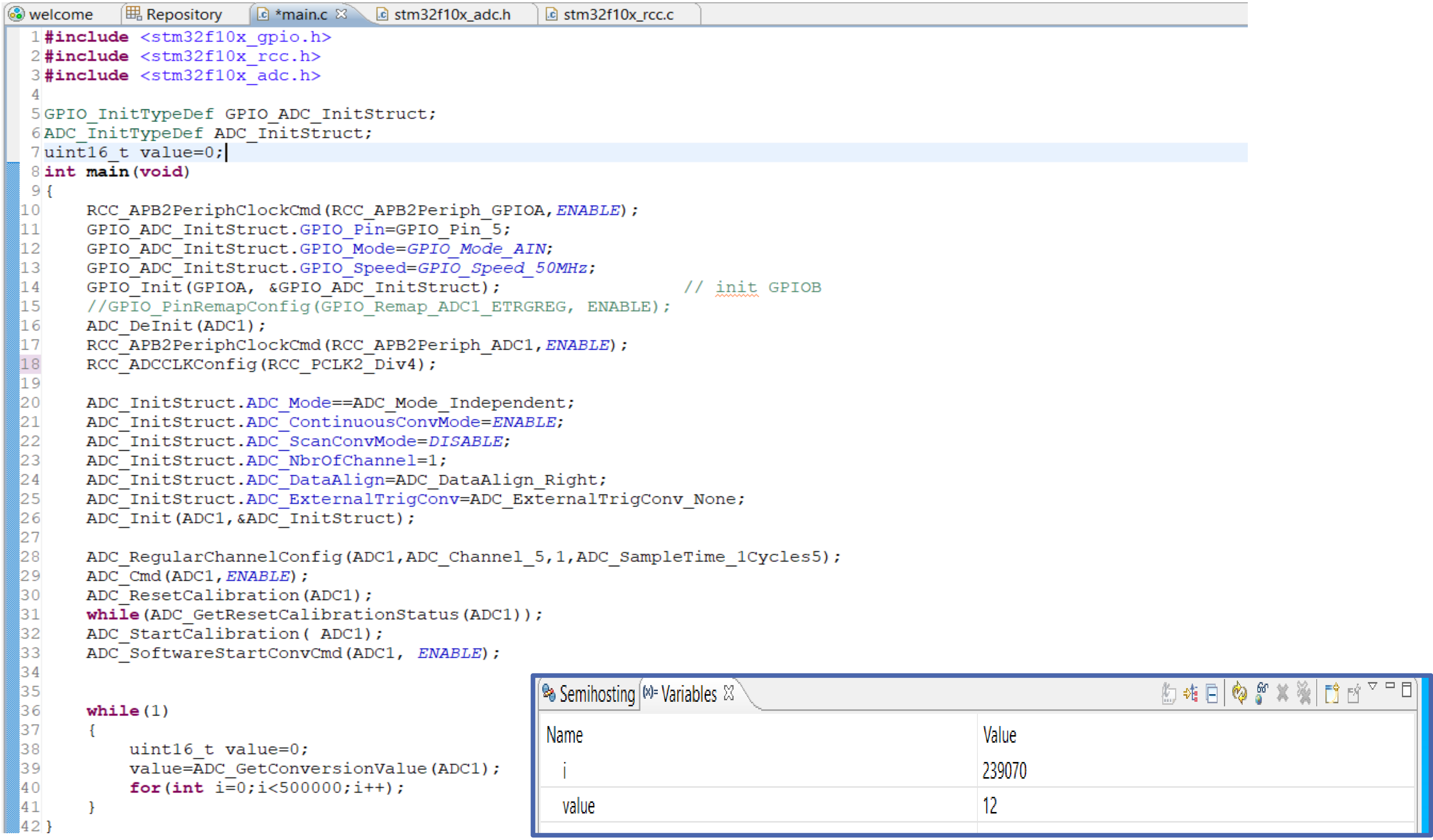


Measured values	Meaning
0-500	Very dry
500-1200	Good
>1200	Wet

Analog to digital converter - ADC

- ADC main features:
- 16 multiplexed channels
- 12-bit resolution
- Sampling frequency
- Single and continuous conversion modes

Example of Analog/Digital Conversion

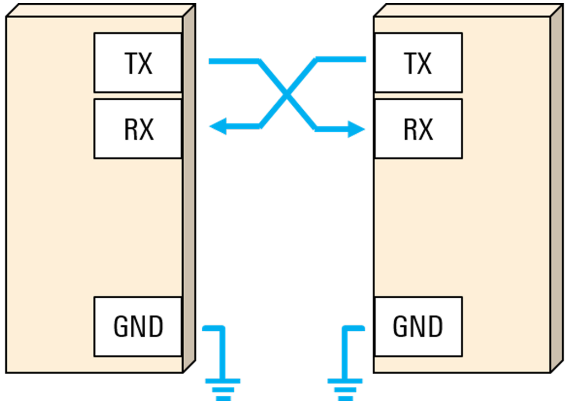


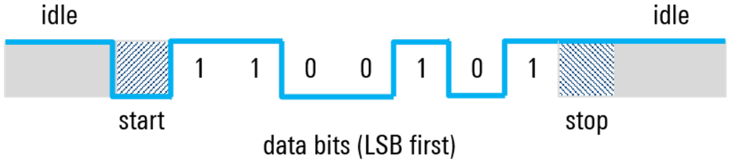
```
#include <stm32f10x_gpio.h>
#include <stm32f10x_rcc.h>
#include <stm32f10x_adc.h>

GPIO_InitTypeDef GPIO_ADC_InitStruct;
ADC_InitTypeDef ADC_InitStruct;
uint16_t value=0;
int main(void)
{
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);
    GPIO_ADC_InitStruct.GPIO_Pin=GPIO_Pin_5;
    GPIO_ADC_InitStruct.GPIO_Mode=GPIO_Mode_AIN;
    GPIO_ADC_InitStruct.GPIO_Speed=GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_ADC_InitStruct);              // init GPIOB
    //GPIO_PinRemapConfig(GPIO_Remap_ADC1_ETRGREG, ENABLE);
    ADC_DeInit(ADC1);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1,ENABLE);
    RCC_ADCCLKConfig(RCC_PCLK2_Div4);

    ADC_InitStruct.ADC_Mode==ADC_Mode_Independent;
    ADC_InitStruct.ADC_ContinuousConvMode=ENABLE;
    ADC_InitStruct.ADC_ScanConvMode=DISABLE;
    ADC_InitStruct.ADC_NbrOfChannel=1;
    ADC_InitStruct.ADC_DataAlign=ADC_DataAlign_Right;
    ADC_InitStruct.ADC_ExternalTrigConv=ADC_ExternalTrigConv_None;
    ADC_Init(ADC1,&ADC_InitStruct);

    ADC_RegularChannelConfig(ADC1,ADC_Channel_5,1,ADC_SampleTime_1Cycles5);
    ADC_Cmd(ADC1,ENABLE);
    ADC_ResetCalibration(ADC1);
    while(ADC_GetResetCalibrationStatus(ADC1));
    ADC_StartCalibration( ADC1);
    ADC_SoftwareStartConvCmd(ADC1, ENABLE);


    while(1)
    {
        uint16_t value=0;
        value=ADC_GetConversionValue(ADC1);
        for(int i=0;i<500000;i++);
    }
}
```

Name	Value
i	239070
value	12

IV exercise - USART communication

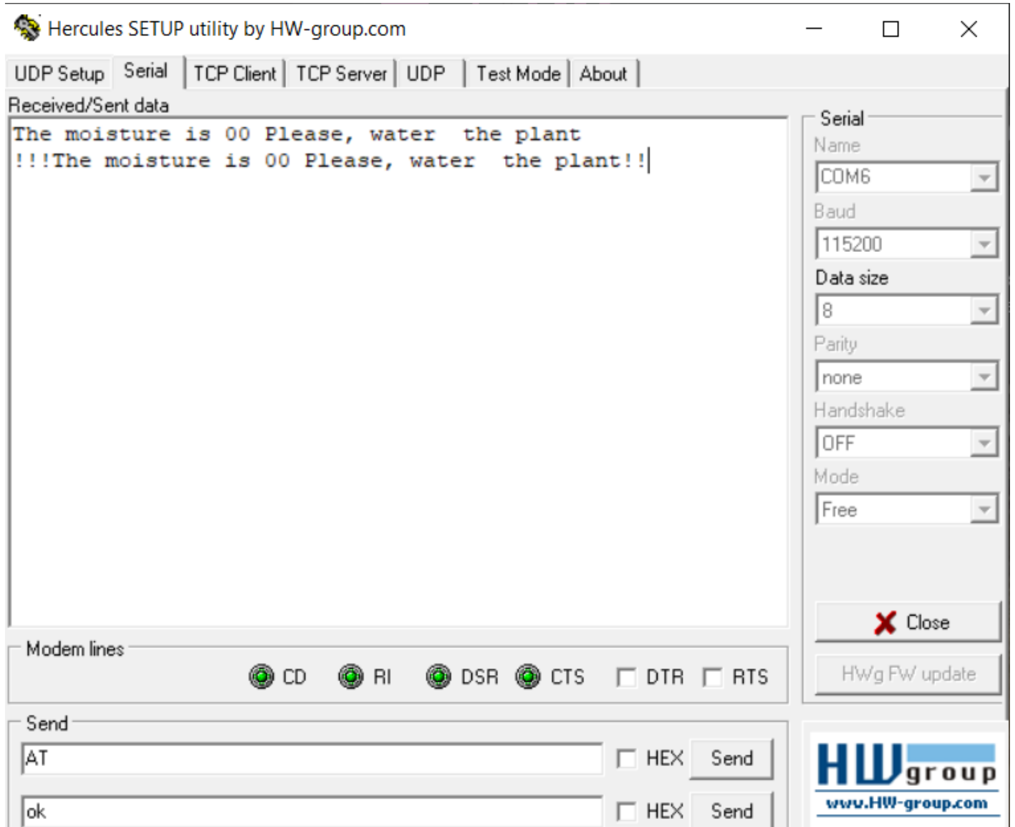
- Universal synchronous asynchronous receiver transmitter (USART) offers flexible data exchange through defined protocol or sets of rules for exchanging data between two devices.





IV exercise - USART communication

- In this exercise students get to know the basic principles of the UART/USART serial communication protocol.
- By using the USART modules of the STM microcontroller the can send the measured values stored on the microcontroller to the PC in order to view them on the serial monitor.



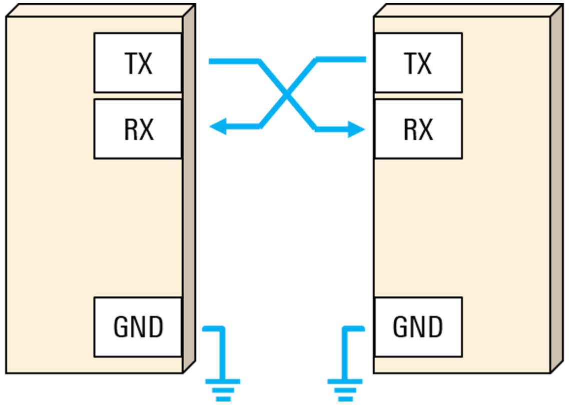
```
The moisture is 00 Please, water  the plant
!!!The moisture is 00 Please, water  the plant!!
```

Serial
Name: COM6
Baud: 115200
Data size: 8
Parity: none
Handshake: OFF
Mode: Free

Send
AT
ok

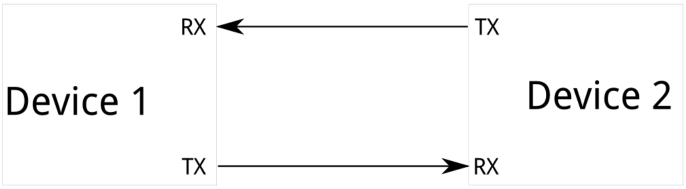
IV exercise - USART communication

- USART Features
  - Full duplex, asynchronous communications
  - A common programmable transmit and receive baud rates
  - Programmable data word length (8 or 9 bits)
  - Configurable stop bits - support for 1 or 2 stop bits
  - Transmitter clock output for synchronous transmission
  - Separate enable bits for Transmitter and Receive

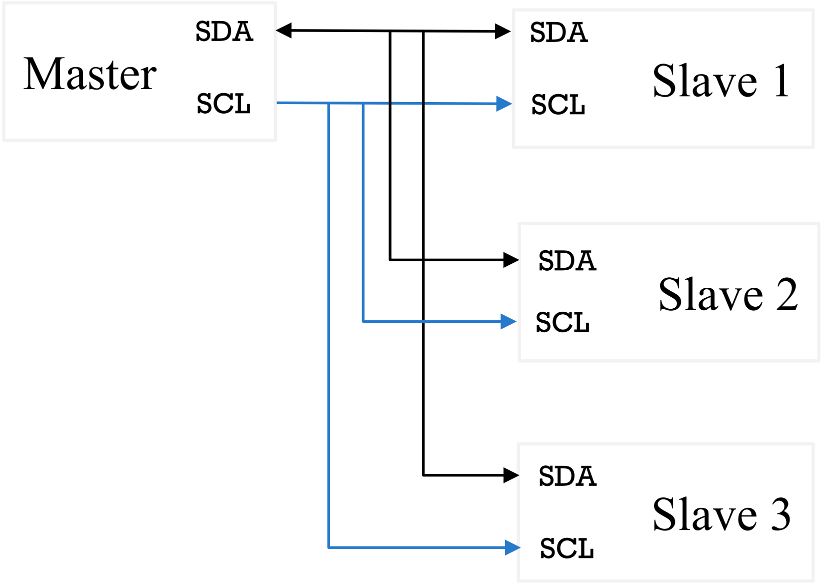


V exercise – I2C communication

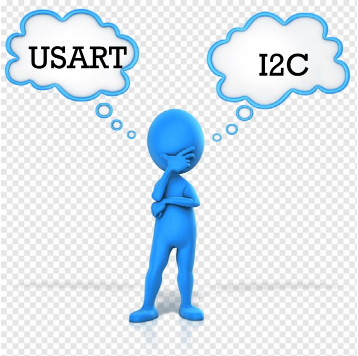
- **Why Inter-Intergrated Circuit (I2C) Protocol?**



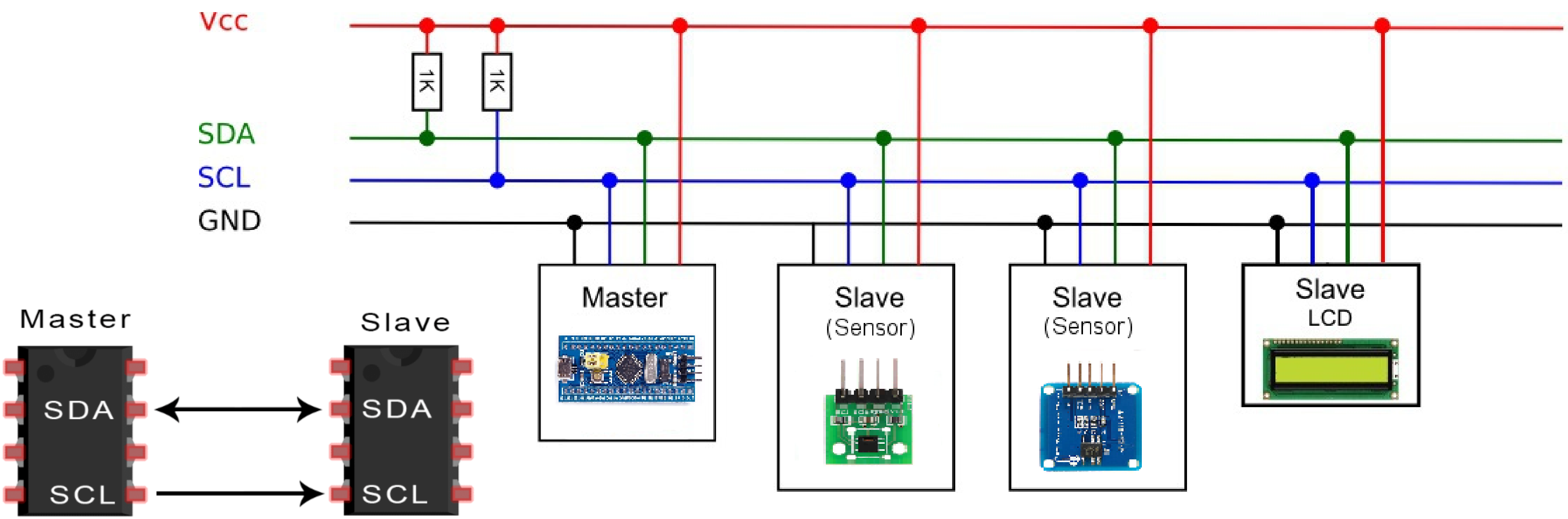
USART



I2C



V exercise – I2C communication



Start	**7 or 10 bits**	**Read/ Write Bit**	**ACK/ NACK Bit**	**8 Bits**	**ACK/ NACK Bit**	**8 Bits**	**ACK/ NACK Bit**	**Stop**

Start Condition

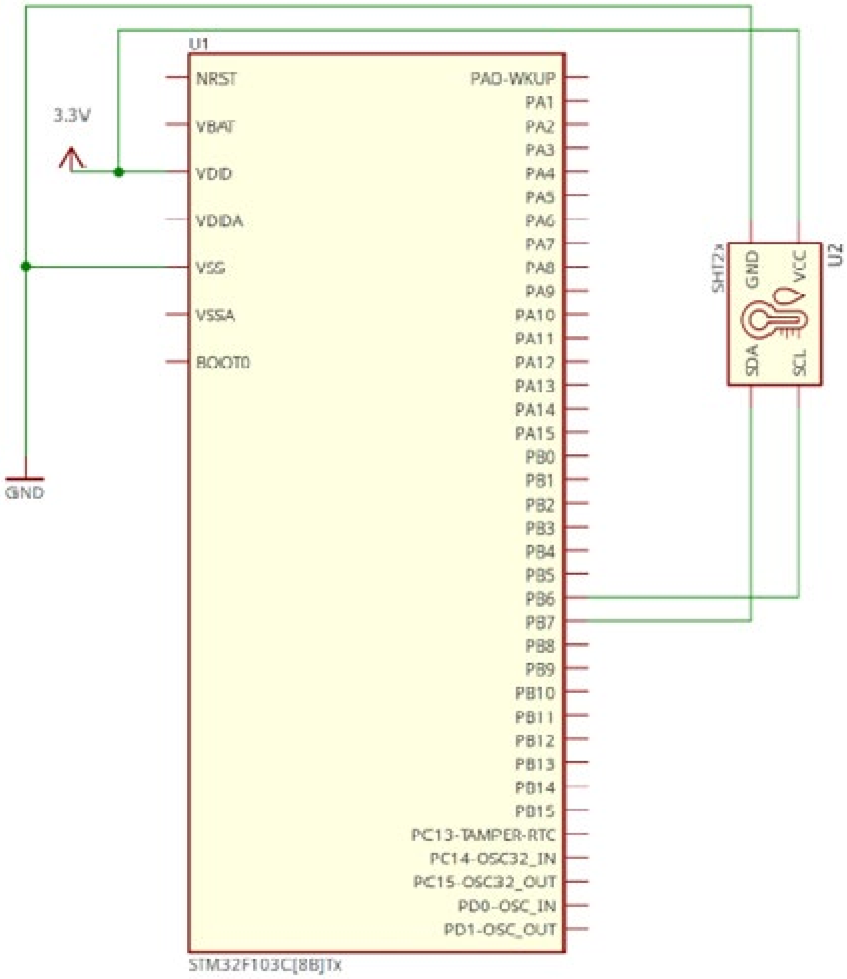
Address Frame

Data Frame

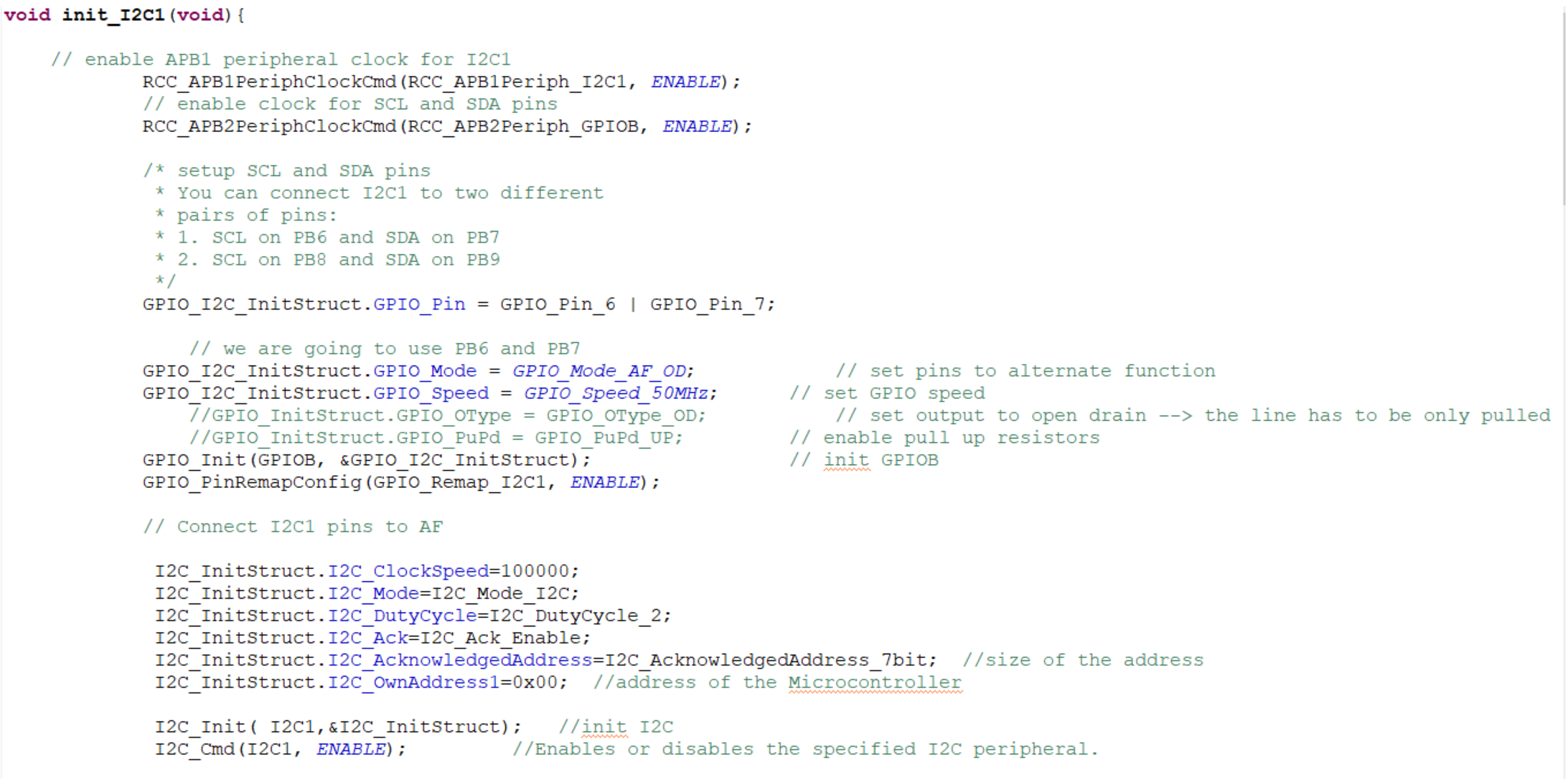
Data Frame

V exercise – Temperature and Humidity sensor with I2C communication

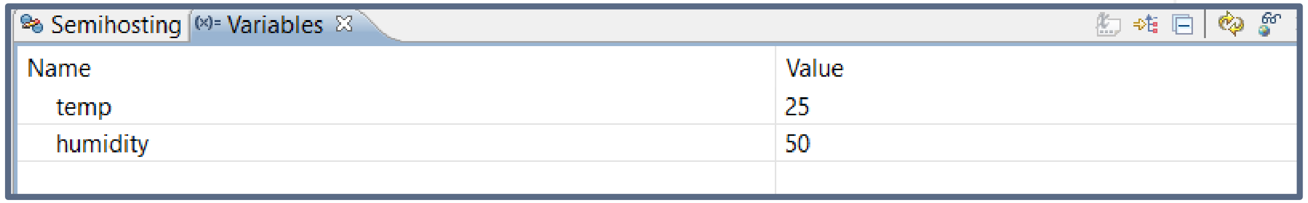
- This exercise gives a brief overview of the operation of the temperature and humidity sensor as well as the I^{2}C communication protocol.
- SHT20 is a combination of a capacitive humidity sensor and ambient temperature sensor.
- By using I^{2}C communication the measured data is sent to the STM memory.



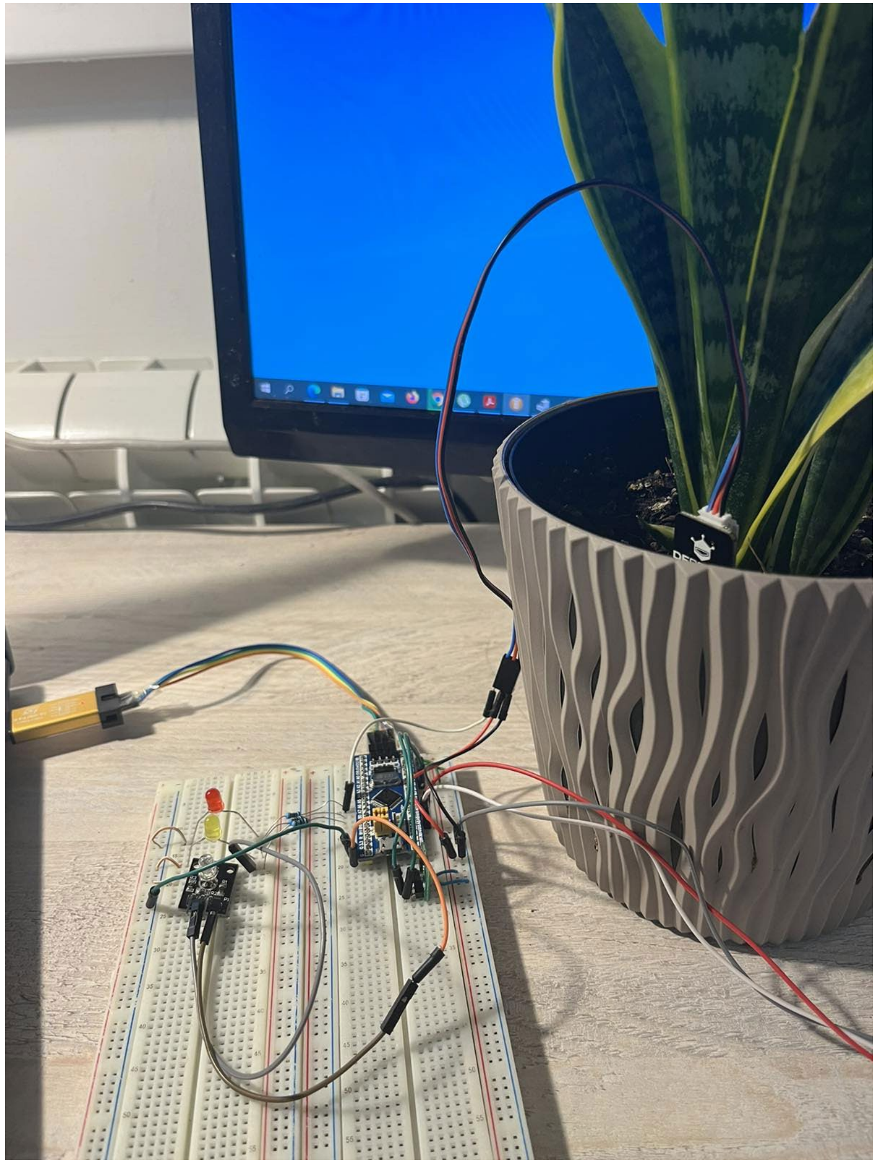
Example with I2C communication



```
void init_I2C1(void){

    // enable APB1 peripheral clock for I2C1
            RCC_APB1PeriphClockCmd(RCC_APB1Periph_I2C1, ENABLE);
            // enable clock for SCL and SDA pins
            RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);

            /* setup SCL and SDA pins
             * You can connect I2C1 to two different
             * pairs of pins:
             * 1. SCL on PB6 and SDA on PB7
             * 2. SCL on PB8 and SDA on PB9
             */
            GPIO_I2C_InitStruct.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7;

                // we are going to use PB6 and PB7
            GPIO_I2C_InitStruct.GPIO_Mode = GPIO_Mode_AF_OD;            // set pins to alternate function
            GPIO_I2C_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;     // set GPIO speed
                //GPIO_InitStruct.GPIO_OType = GPIO_OType_OD;           // set output to open drain --> the line has to be only pulled
                //GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_UP;         // enable pull up resistors
            GPIO_Init(GPIOB, &GPIO_I2C_InitStruct);                 // init GPIOB
            GPIO_PinRemapConfig(GPIO_Remap_I2C1, ENABLE);

            // Connect I2C1 pins to AF

             I2C_InitStruct.I2C_ClockSpeed=100000;
             I2C_InitStruct.I2C_Mode=I2C_Mode_I2C;
             I2C_InitStruct.I2C_DutyCycle=I2C_DutyCycle_2;
             I2C_InitStruct.I2C_Ack=I2C_Ack_Enable;
             I2C_InitStruct.I2C_AcknowledgedAddress=I2C_AcknowledgedAddress_7bit;  //size of the address
             I2C_InitStruct.I2C_OwnAddress1=0x00;  //address of the Microcontroller

             I2C_Init( I2C1,&I2C_InitStruct);   //init I2C
             I2C_Cmd(I2C1, ENABLE);         //Enables or disables the specified I2C peripheral.

}
```



Name	Value
temp	25
humidity	50



Co-funded by the
Erasmus+ Programme
of the European Union

UbiLAB

Thank You!

Erasmus+ project no. 2020-1-MK01-KA226-HE-094548