tomas-fryza / **Digital-electronics-2**

Code Issues Pull requests Actions Projects Wiki Security Insights

master

Digital-electronics-2 / Labs / **01-tools** /

tomas-fryza … 3 days ago

..

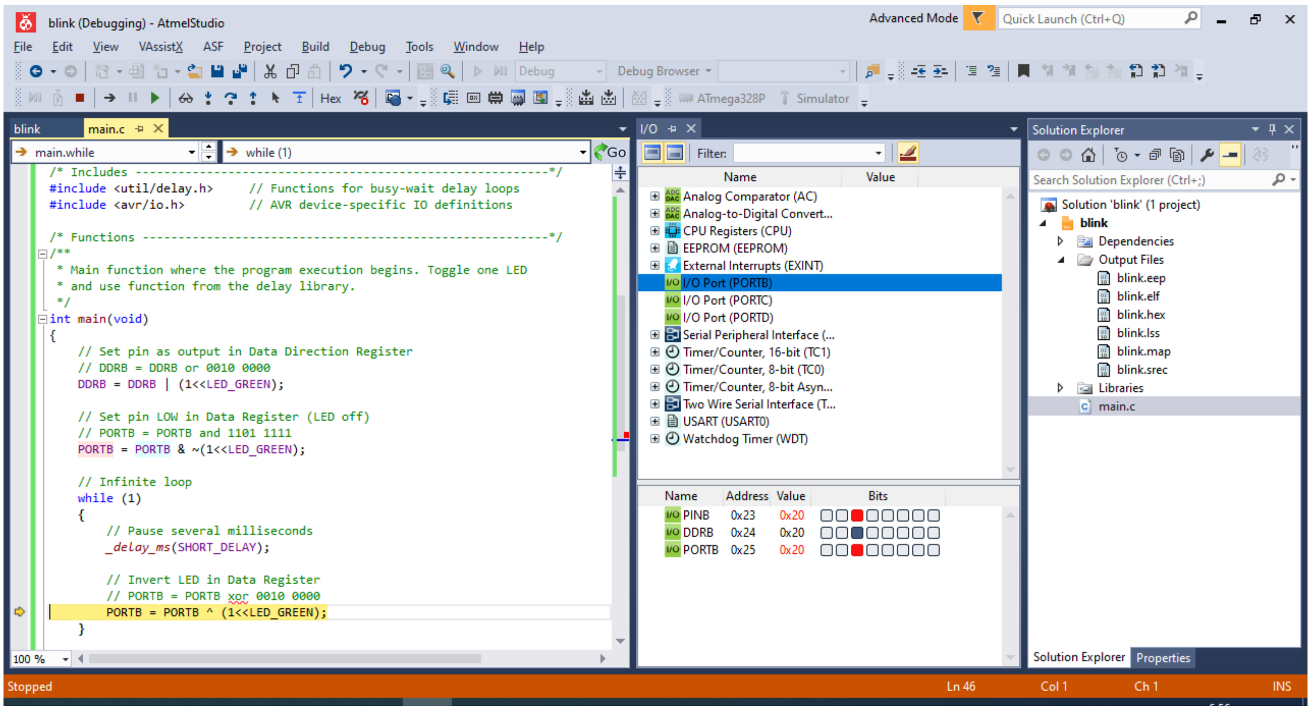
Images	26 days ago
README.md	3 days ago

README.md

Lab 1: Git version-control system, AVR tools

Learning objectives

The purpose of this laboratory exercise is to learn how to use the git versioning system, write the markdown readme file, learn the basic structure of C code, and learn how to use development tools to program a microcontroller on the Arduino Uno board.


blink (Debugging) - AtmelStudio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug Debug Browser ATmega328P Simulator
blink main.c
main.while while (1)
I/O Filter:
Analog Comparator (AC)
Analog-to-Digital Convert...
CPU Registers (CPU)
EEPROM (EEPROM)
External Interrupts (EXINT)
I/O Port (PORTB)
I/O Port (PORTC)
I/O Port (PORTD)
Serial Peripheral Interface (...
Timer/Counter, 16-bit (TC1)
Timer/Counter, 8-bit (TC0)
Timer/Counter, 8-bit Asyn...
Two Wire Serial Interface (T...
USART (USART0)
Watchdog Timer (WDT)
Solution Explorer
Solution 'blink' (1 project)
blink
Dependencies
Output Files
blink.eep
blink.elf
blink.hex
blink.lss
blink.map
blink.srec
Libraries
main.c
Stopped Ln 46 Col 1 Ch 1 INS

```c
/* Includes -----------------------------------------------------------*/
#include <util/delay.h>     // Functions for busy-wait delay loops
#include <avr/io.h>         // AVR device-specific IO definitions

/* Functions ----------------------------------------------------------*/
/**
 * Main function where the program execution begins. Toggle one LED
 * and use function from the delay library.
 */
int main(void)
{
    // Set pin as output in Data Direction Register
    // DDRB = DDRB or 0010 0000
    DDRB = DDRB | (1<<LED_GREEN);

    // Set pin LOW in Data Register (LED off)
    // PORTB = PORTB and 1101 1111
    PORTB = PORTB & ~(1<<LED_GREEN);

    // Infinite loop
    while (1)
    {
        // Pause several milliseconds
        _delay_ms(SHORT_DELAY);

        // Invert LED in Data Register
        // PORTB = PORTB xor 0010 0000
        PORTB = PORTB ^ (1<<LED_GREEN);
    }
```

Name	Address	Value
PINB	0x23	0x20
DDRB	0x24	0x20
PORTB	0x25	0x20

Preparation tasks (done before the lab at home)

Create an account on GitHub.

According to your preferences, choose one of the variants below and prepare the development chain on your own computer.

Windows and Atmel Studio 7

Download and install:

- Atmel Studio 7,
- SimulIDE, and
- git.

If you have the option to use Arduino Uno board and logic analyzer, also download and install:

- Arduino IDE, which contains all USB drivers and
- Saleae logic.

To make it easier to work with git, you can install a graphical client:

- GitKraken or
- GitHub Desktop.

Windows and command-line toolchain

Follow the instructions for Windows and create an entire comand-line toolchain instead of using Atmel Studio.

Download and install SimulIDE.

To make it easier to work with git, you can install a graphical client:

- GitKraken or
- GitHub Desktop.

Ubuntu-based Linux distributions

Follow the instructions for Linux and create an entire comand-line toolchain.

Download and install SimulIDE.

To make it easier to work with git, you can install a graphical client GitKraken.

Part 1: GitHub

GitHub is a code hosting platform for collaboration and version control. GitHub lets you (and others) work together on projects.

In GitHub, create a new public repository titled **Digital-electronics-2**. Initialize a README, .gitignore, and MIT license.

Use one of the available git manuals, such as 1, 2, or 3, and add the following sections to your README file.

- Headers
- Emphasis (italics, bold)
- Lists (ordered, unordered)
- Links
- Table
- Listing of C source code

Part 2: Local repository

Run Git Bash (Windows) of Terminal (Linux) and create your own home folder inside `Documents`.

```
## Windows Git Bash:
$ cd d:/Documents/
$ mkdir your-name
$ cd your-name/

## Linux:
$ cd
$ cd Documents/
$ mkdir your-name
$ cd your-name/
```

With help of `git` command, clone a local copy of your public repository.

```
## Windows Git Bash or Linux:
$ git clone https://github.com/your-github-account/Digital-electronics-2
$ cd Digital-electronics-2/
$ ls
LICENSE  README.md
```

Download `Docs` and `Examples` folders from this repository and copy them to your `Digital-electronics-2` local repository.

```
## Windows Git Bash or Linux:
$ ls
Docs  Examples  LICENSE  README.md
```

Create a new working folder `Labs/01-tools` for this exercise.

```
## Windows Git Bash or Linux:
$ mkdir Labs
$ cd Labs/
$ mkdir 01-tools
```

Part 3: Test AVR tools

Version: Windows and Atmel Studio 7

Follow any online tutorial, such as 1 or 2, create a new GCC C Executable Project for ATmega328P within `01-tools` working folder and copy/paste blink example code to your `main.c` source file. Examine all lines of source code. What is the meaning of individual commands?

Compile the project.

Simulate the project in Atmel Studio 7.

Run external programmer in menu **Tools** > **Send to Arduino UNO** and download the compiled code to Arduino Uno board. Note that, this external tool is configured according to How to Flash AVR from Atmel Studio.

Version: Windows and command-line toolchain

Copy `main.c` and `Makefile` files from blink example to `Labs\01-tools` folder.

Copy `Example\Makefile.in` settings file to `Labs` folder. Note that, this file contains parameters and settings that are identical for all (future) projects located in this folder. Uncomment the Windows settings in this file. Make sure the values for `PREFIX` and `AVRDUDE` contain the correct paths and `USBPORT` contains port where Arduino board is connected.

```
## Linux
#PREFIX  = /opt/avr8-gnu-toolchain-linux_x86_64
#AVRDUDE = avrdude
#RM      = rm -f
## See "dmesg" command output
#USBPORT = /dev/ttyUSB0
```

```
## Windows
PREFIX  = C:\APPZ\Atmel\Studio\7.0\toolchain\avr8\avr8-gnu-toolchain
AVRDUDE = C:\APPZ\avrdude\avrdude.exe
RM      = del
# See USB-SERIAL CH340 port in Device Manager
USBPORT = COM3
```

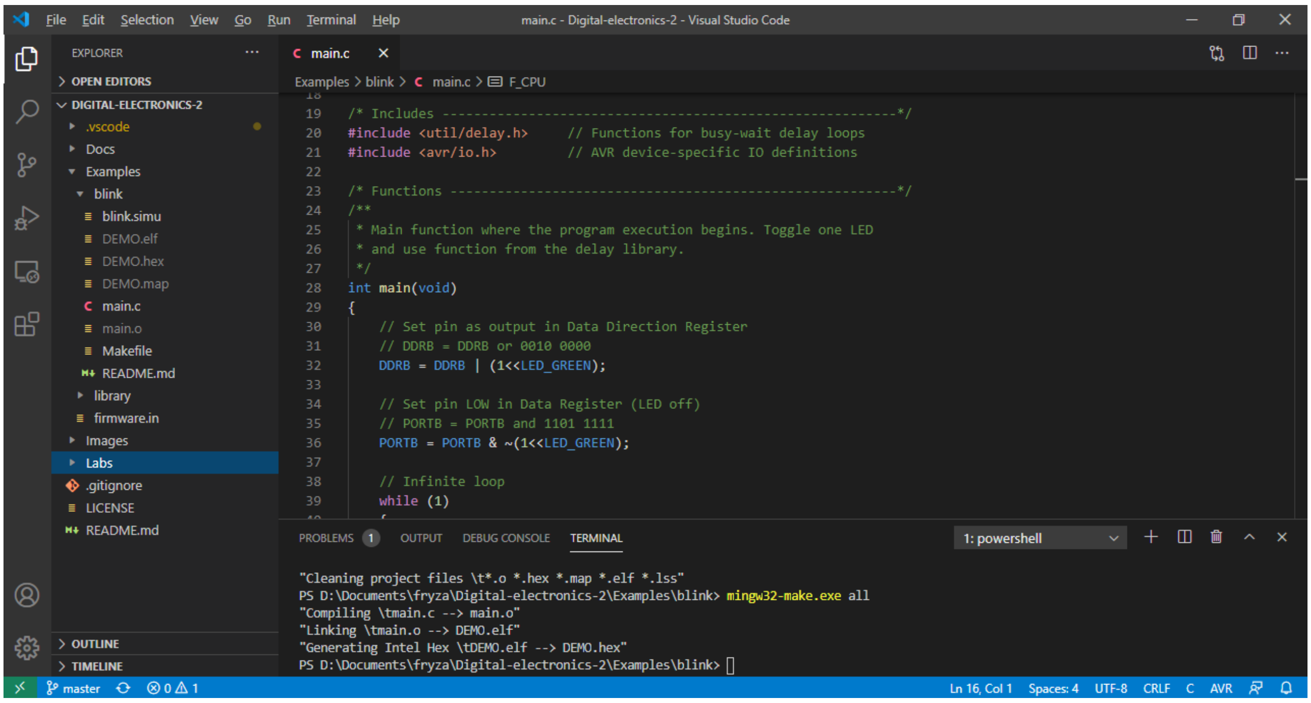
Run Visual Studio Code source code editor, open your `Digital-electronics-2` working folder, run internal terminal in menu **Terminal > New Terminal**, and change path to `Labs\01-tools`.

```
cd Labs\01-tools\
```

Open `main.c` source file. What is the meaning of each line of this source code?

Use the following commands step by step in the internal terminal to find out what they mean. Note: these commands are defined in `Makefile`.

```
mingw32-make.exe all
mingw32-make.exe clean
mingw32-make.exe size
mingw32-make.exe flash
```



```c
/* Includes ----------------------------------------------------------*/
#include <util/delay.h>     // Functions for busy-wait delay loops
#include <avr/io.h>         // AVR device-specific IO definitions

/* Functions ---------------------------------------------------------*/
/**
 * Main function where the program execution begins. Toggle one LED
 * and use function from the delay library.
 */
int main(void)
{
    // Set pin as output in Data Direction Register
    // DDRB = DDRB or 0010 0000
    DDRB = DDRB | (1<<LED_GREEN);

    // Set pin LOW in Data Register (LED off)
    // PORTB = PORTB and 1101 1111
    PORTB = PORTB & ~(1<<LED_GREEN);

    // Infinite loop
    while (1)
```

```
"Cleaning project files \t*.o *.hex *.map *.elf *.lss"
PS D:\Documents\fryza\Digital-electronics-2\Examples\blink> mingw32-make.exe all
"Compiling \tmain.c --> main.o"
"Linking \tmain.o --> DEMO.elf"
"Generating Intel Hex \tDEMO.elf --> DEMO.hex"
PS D:\Documents\fryza\Digital-electronics-2\Examples\blink>
```

Version: Ubuntu-based Linux distributions

Copy `main.c` and `Makefile` files from blink example to `Labs/01-tools` folder.

Copy `Example/Makefile.in` settings file to `Labs` folder. Note that, this file contains parameters and settings that are identical for all (future) projects located in this folder. Uncomment the Linux settings in this file. Make sure the values for `PREFIX` and `AVRDUDE` contain the correct paths and `USBPORT` contains port where Arduino board is connected.

```
## Linux
PREFIX  = /opt/avr8-gnu-toolchain-linux_x86_64
AVRDUDE = avrdude
RM      = rm -f
# See "dmesg" command output
USBPORT = /dev/ttyUSB0

## Windows
#PREFIX  = C:\APPZ\Atmel\Studio\7.0\toolchain\avr8\avr8-gnu-toolchain
#AVRDUDE = C:\APPZ\avrdude\avrdude.exe
#RM      = del
## See USB-SERIAL CH340 port in Device Manager
#USBPORT = COM3
```

Run Visual Studio Code source code editor, open your `Digital-electronics-2` working folder, run internal terminal in menu **Terminal > New Terminal**, and change path to `Labs/01-tools`.

```
cd Labs/01-tools/
```

Open `main.c` source file. What is the meaning of each line of this source code?

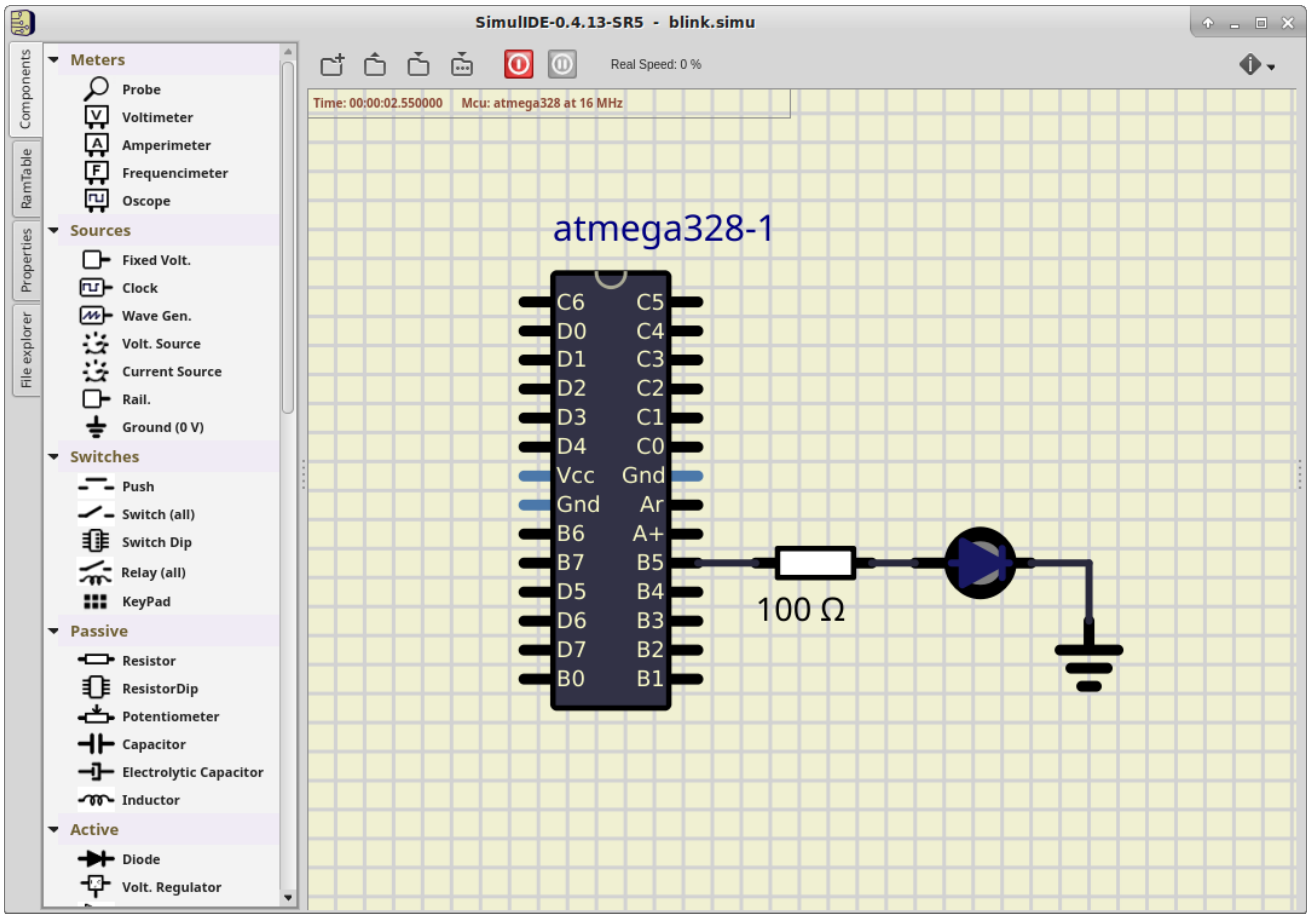
Use the following commands step by step in the internal terminal to find out what they mean. Note: these commands are defined in `Makefile`.

```
$ make all
$ make clean
$ make size
$ make flash
```

Part 4: SimulIDE

Run SimulIDE, use online tutorials, and create a circuit with ATmega328 AVR microcontroller.

All circuit and control elements are available in the **Components** tab. Use the following components ATmega328 (**Micro > AVR > atmega > atmega328**), resistor (**Passive > Resistor**), LED (**Outputs > Led**), and GND (**Sources > Ground (0 V)**) and connect them as shown.



Right-click on the ATmega package and select **Load firmware**. In your project folder, find the `*.hex` file that was created by compiling in the previous point.

Register values can be displayed in the **RamTable** tab. In the **Reg.** column, type `DDRB` on the first line and `PORTB` on the second.

Click to **Power Circuit** button, simulate the project, and monitor the LED status and register values. The simulation can be paused with the **Pause Simulation** button and stopped by pressing the **Power Circuit** button again.

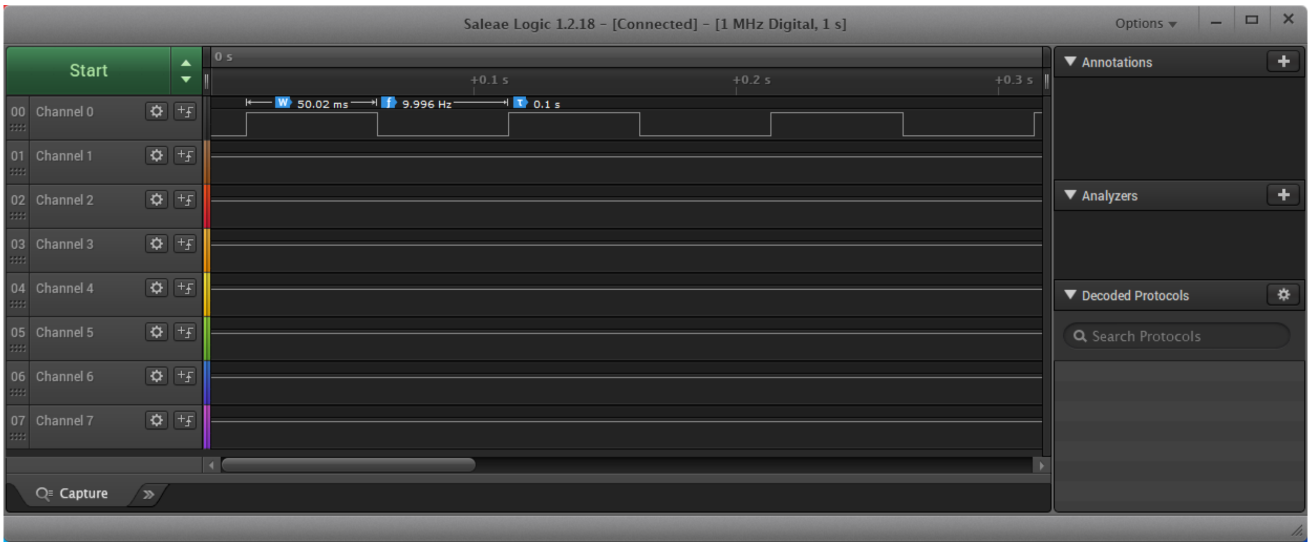
You can connect a probe (**Meters** > **Probe**), an oscilloscope (**Meters** > **Oscope**), or a voltmeter (**Meters** > **Voltimeter**) to output B5 and observe the voltage.

The properties of each component can be found/changed in the **Properties** tab.

Part 5: Logic analyzer

Run Saleae Logic software, use wire and connect Channel 0 to Arduino board pin 13 (pin PB5 is connected here), and verify the duration of delay function.

To start sampling, press the green button with two arrows, set the sampling rate to 1 MS/s and the recording time to 1 second. Click the Start button.

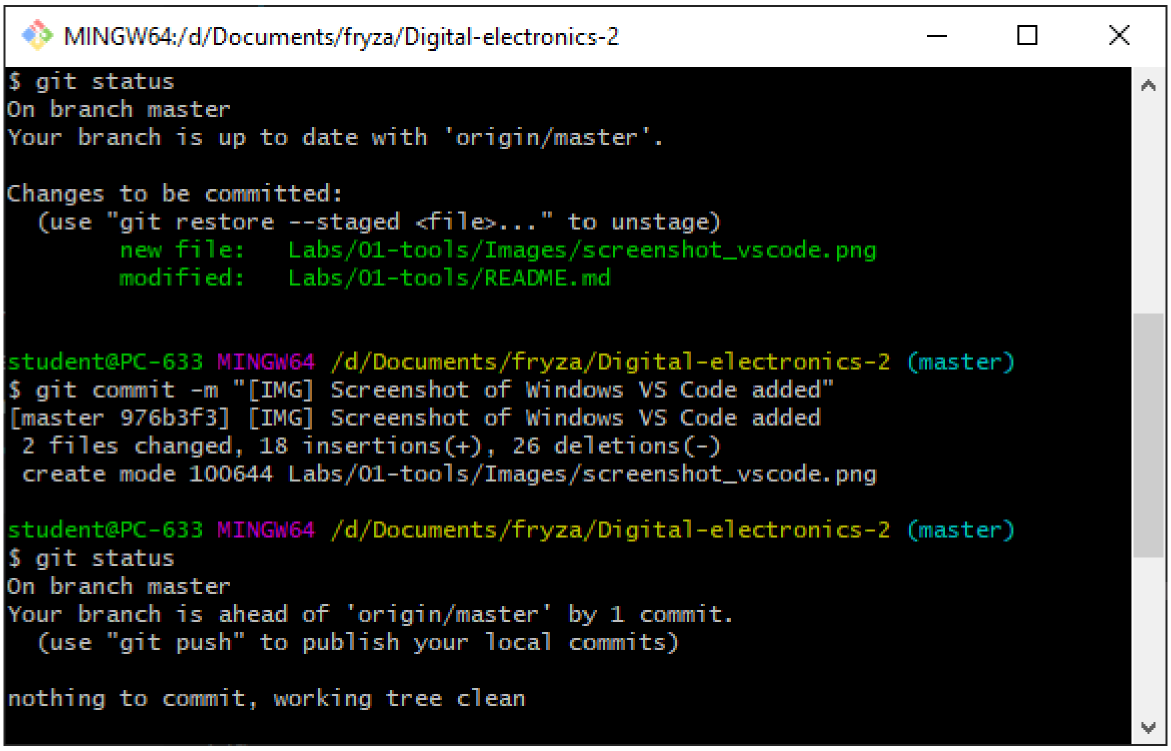


Synchronize repositories

When you finish working, always synchronize the contents of your working folder with the local and remote versions of your repository. This way you are sure that you will not lose any of your changes.

Use git commands to add, commit, and push all local changes to your remote repository. Note that, a detailed description of all git commands can be found here. Check the repository at GitHub web page for changes.

```
## Windows Git Bash or Linux:
$ git status
$ git add <your-modified-files>
$ git status
$ git commit -m "[LAB] AVR toolchain tested"
$ git status
$ git push
$ git status
```


MINGW64:/d/Documents/fryza/Digital-electronics-2

```
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Labs/01-tools/Images/screenshot_vscode.png
        modified:   Labs/01-tools/README.md


student@PC-633 MINGW64 /d/Documents/fryza/Digital-electronics-2 (master)
$ git commit -m "[IMG] Screenshot of Windows VS Code added"
[master 976b3f3] [IMG] Screenshot of Windows VS Code added
 2 files changed, 18 insertions(+), 26 deletions(-)
 create mode 100644 Labs/01-tools/Images/screenshot_vscode.png

student@PC-633 MINGW64 /d/Documents/fryza/Digital-electronics-2 (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Experiments on your own

1. Choose one variant and install the AVR development tools on your computer.
2. Modify the `01-tools` application so that the string `DE2` is repeatedly displayed on the LED in the Morse code.
3. Simulate the Morse code application in SimulIDE.

Lab assignment

1. Submit the GitHub link to your `Digital-electronics-2` repository.
2. Blink example. Submit:
   - Answers to questions: What is the meaning of `|`, `&`, `^`, `~`, `<<` binary operators? Write a truth table and explain the use of operators with examples.
3. Morse code application. Submit:
   - Listing of C code (`main.c`).

The deadline for submitting the task is the day before the next laboratory exercise. Use BUT e-learning web page and submit a single PDF file.