Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра «Телекомунікації»

Методичні вказівки
до виконання лабораторної роботи №1
з дисципліни «Вбудовані системи»
на тему «Налаштування робочого середовища»

Львів 2021

Мета роботи: Встановити необхідні пакети та налаштувати робоче середовище.

Інсталяція необхідних пакетів для Linux Ubuntu.

```
# Інсталяція AVR Toolchain
 sudo apt-get install gcc-avr avr-libc avrdude

# Інсталяція VS Code
# Варіант №1
 sudo apt update
 sudo apt install snapd
 sudo snap install --classic code


# Варіант №2
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg --
dearmor > packages.microsoft.gpg

sudo install -o root -g root -m 644 packages.microsoft.gpg
/etc/apt/trusted.gpg.d/

sudo sh -c 'echo "deb [arch=amd64,arm64,armhf signed-
by=/etc/apt/trusted.gpg.d/packages.microsoft.gpg]
https://packages.microsoft.com/repos/code stable main" >
/etc/apt/sources.list.d/vscode.list'

# Оновіть кеш пакунка та встановіть пакет, використовуючи:
 sudo apt install apt-transport-https
 sudo apt update
 sudo apt install code  #or code-insiders


# Інсталяція додаткових розширень для VS Code
code --install-extension jkearins.action-buttons-ext
code --install-extension ms-vscode.cpptools
```

Інсталяція симулятора електронних схем SimulIDE
https://www.simulide.com/p/home.html

```
sudo apt-get install simulide

#якщо симулятор не запускається, то потрібно встановити додаткові
бібліотеки

sudo apt-get install libqt5core5a libqt5gui5 libqt5xml5 libqt5svg5
libqt5widgets5 libqt5concurrent5 libqt5multimedia5 libqt5multimedia5-
plugins libqt5serialport5 libqt5script5 libelf1
```

Створюємо дерево каталогів проекту для Visual Studio Code (VScode) з такою структурою:

```
/test_blink
      |__ Makefile
      |__ main.c
      |__ /.vscode
            |__ c_cpp_properties.json
            |__ tasks.json
            |__ settings.json
```

(test_blink та .vscode є папками (каталогами))

Відкриваємо папку проекту у ***VSCode*** вибираємо в меню **Terminal -> Run task… -> Build**, якщо компіляція пройшла успішно, то у робочому каталозі появиться папка build в якій буде файл з розширенням ***.hex** (`test_blink.hex`)**.**

Створюємо схему в симуляторі як показано на Рис 1., завантажуємо hex–файл (`test_blink.hex`) у модель мікроконтролера Рис 2. та запускаємо моделювання. Після перекомпіляції проекту потрібно перезавантажити hex-файл.

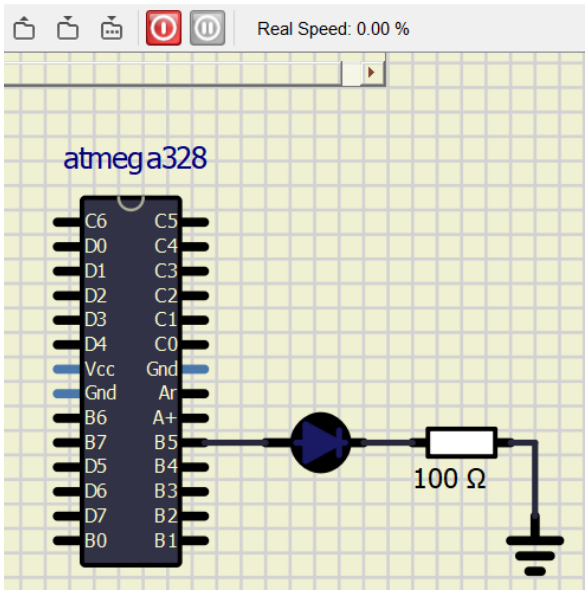

Рис 1.

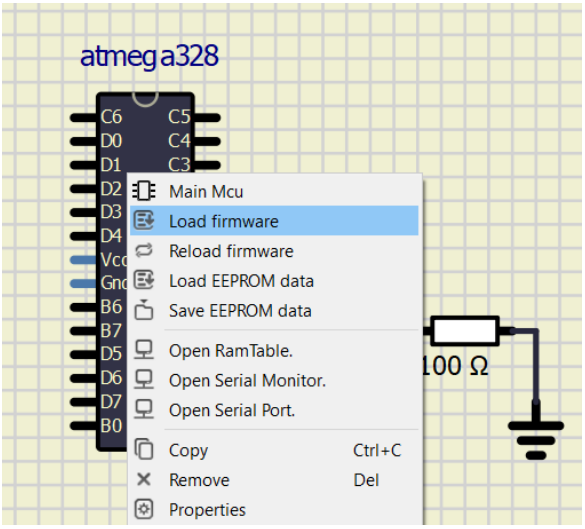

Рис 2.

Додаткове завдання.

Написати програму яка з періодом 1 секунда показує сигнал SOS (. . . - - - . . .) на світлодіоді (моргає світлодіодом короткі і довгі імпульси).

Головний файл тестової програми **main.c**

```
#include <avr/io.h>
#include <util/delay.h>
#define LED_PIN PB5 // вбудований в Arduino nano світлодіод

int main(void)
{
    DDRB |= 1 << LED_PIN;
    while (1) {
        PORTB |= 1 << LED_PIN;     // включити світлодіод
        _delay_ms(100);            // затримка
        PORTB &= ~(1 << LED_PIN); // виключити світлодіод
        _delay_ms(400);
    }
  return 0;
}
```

Вміст файлу c_cpp_properties.json

```
{
    "configurations": [
        {
            "name": "vscode_avr_c_properties",
            "includePath": [
                "${workspaceFolder}/**",
                "/usr/lib/avr/include"
            ],
            "defines": [
                "__AVR_ATmega328P__"
            ],
            "cStandard": "c11",
            "intelliSenseMode": "gcc-x64",
            "compilerPath": "avr-gcc"
        }
    ],
    "version": 4
}
```

Вміст файлу tasks.json

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "label": "Build",
            "type": "shell",
            "command": "make",
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "problemMatcher": ["$gcc"]
```

```
            ]
        },
        {
            "label": "Flash",
            "type": "shell",
            "args": ["flash"],
            "command": "make",
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "problemMatcher": ["$gcc"]
            ]
        }
    ]
}
```

Вміст файлу settings.json

```
{
    "actionButtons": {
        "defaultColor": "#FFFFFF",
        "loadNpmCommands":false,
        "reloadButton":"↻",
        "commands": [
            {
                // Початкова папка терміналу ${workspaceFolder}
                "cwd": "${workspaceFolder}",
                "name": "♻ Build Projekt",
                "color": "yellow",
                "singleInstance": true,
                // Команда яка виконається при кліку на actionButton
                "command": "make",
            },
            {
                "cwd": "${workspaceFolder}",
                "singleInstance": true,
                "name": "▲ Flash hex",
                "color": "white",
                "command": "make flash",
            },
            {
                "name": "$(split-horizontal) Split editor",
                "color": "orange",
                "useVsCodeApi": true,
                "command": "workbench.action.splitEditor"
            }
        ]
    },
    "C_Cpp.errorSquiggles": "Disabled"
}
```

Вміст файлу **Makefile**

```
# Назва проекту
PROJECT_NAME = test_blink
# Модель мікроконтролера
MCU = atmega328p
# Частота тактового генератора
F_CPU = 16000000UL
# Порт для програмування
UPLOAD_PORT=/dev/ttyUSB0
# Шридкість передачі даних [19200 -(nano atmega168p), 57600 -(nano
atmega328p), 115200 -(uno)]
UPLOAD_PORT_BAUD=57600

# Sources files needed for building the application
#SRC = $(wildcard *.c)
SRC = main.c
SRC +=

# The headers files needed for building the application
INCLUDE = -I.
INCLUDE +=

#-------------------------------------------------------------------
PROJECT_NAME := $(strip $(PROJECT_NAME))
THIS_DIR := $(dir $(abspath $(firstword $(MAKEFILE_LIST))))

#OBJS:=$(SRC: .c=.o)
OBJS:= $(addsuffix .o,$(basename $(SRC)))
OBJS:= $(addprefix build/,$(OBJS))

CC = avr-gcc
OBJCOPY = avr-objcopy
AVRDUDE = avrdude
AVRDUDE_CONF = /etc/avrdude.conf
TARGET_DIR = build
RM = rm -rf

CFLAGS =-Os
CFLAGS +=-Wall
CFLAGS +=-fpack-struct
CFLAGS +=-fshort-enums
CFLAGS +=-ffunction-sections
CFLAGS +=-fdata-sections
CFLAGS +=-std=gnu99
CFLAGS +=-funsigned-char
CFLAGS +=-funsigned-bitfields

.PHONY: all

all: $(TARGET_DIR) clean build/$(PROJECT_NAME).hex sizedummy

build/$(PROJECT_NAME).hex: build/$(PROJECT_NAME).elf
	@echo Create exec file $@
	@$(OBJCOPY) -O ihex -R .eeprom $< $@
	@echo BUILD SUCCESS!
```

```
build/$(PROJECT_NAME).elf: $(OBJS)
	@echo Building target: $@
	@$(CC) -Os -mmcu=$(MCU) -o $@ $^

build/%.o: %.c
	@echo Compile file: $<
	@$(CC) $(CFLAGS) -mmcu=$(MCU) -DF_CPU=$(F_CPU) -o $@ -c $<

$(TARGET_DIR):
	mkdir $(TARGET_DIR)

.PHONY: clean sizedummy

sizedummy:
	@echo
	@avr-size --format=avr --mcu=$(MCU) build/$(PROJECT_NAME).elf

flash:
	@$(AVRDUDE) -C$(AVRDUDE_CONF) -F -v -p$(MCU) -carduino -P$(UPLOAD_PORT)
-b$(UPLOAD_PORT_BAUD) -D -Uflash:w:build/$(PROJECT_NAME).hex:i

clean:
	@$(RM) build/*.o build/$(PROJECT_NAME).elf build/$(PROJECT_NAME).hex
```