- Prednáška Utorok 07:30 – 09:00 PK2_013(L5)
- Projektová práca Utorok 09:10 – 10:40; 10:50 - 12:20 PK2_013(L5)

- Literatúra:
  - Prednášky
  - Livovský: Mikroprocesorová technika ATmega328P.
  - Programovanie v jazyku C: http://people.tuke.sk/igor.podlubny/C/index.htm
  - *Datasheet* Atmega328P, Dokumentácia *MicroChip Studio*
  - https://onlinedocs.microchip.com/oxy/GUID-ECD8A826-B1DA-44FC-BE0B-5A53418A47BD-en-US-12/index.html

- Software:
  - *MicroChip Studio*; *Avrdude*; *Terminal v1.9b, Termite (CompuPhase)*
  - AVR *library* http://nongnu.org/avr-libc/user-manual/index.html
  - https://onlinedocs.microchip.com/oxy/GUID-317042D4-BCCE-4065-BB05-AC4312DBC2C4-en-US-2/index.html

- Informácie k predmetu: literatúra, softvér na stiahnutie, zdrojové príklady: http://kte.fei.tuke.sk/livovsky/ZMT
- Laboratórne cvičenie: notebook, vývojový kit s mikrokontrolérom AVR (napr. *Arduino UNO*)

- Predmet ZMT nadväzuje na predmety: „Základy algoritmizácie a programovania“, „Počítačové inžinierstvo v elektronike“, „Programovanie“ a „Základy elektroniky“.

- Priebežné hodnotenie (zápočet): 0 – 40 bodov, podľa náročnosti riešeného príkladu (12 a 13 týždeň).
- Skúška: 0 - 60 bodov písomná.

- Online konzultácia (na požiadanie študenta): Piatok 09:00-10:00.

Zápočtové zadania.

Študent si v 5. týždni zvolí jedno zadanie, ktoré bude samostatne realizovať s možnosťou konzultácií na cvičeniach počas 6. – 11. týždňa a obhajovať v 12. a 13. týždni. Minimálne 24 hodín pres konaním obhajoby je potrebné odoslať zdrojový kód s komentárom na kontrolu. Obhajoba bude pozostávať z praktickej ukážky riešenia a drobných modifikácií riešenia.

Bodové hodnotenie 0-25.

1. Pripojením logickej úrovne LOW na vývod PB0 sa LED (PB5) rozsvieti. LED zostáva rozsvietená aj po odpojení LOW z PB0. Ak sa pripojí na logická úroveň LOW na PD7, LED (PB5) zhasne a ostáva zhasnutá aj po odpojení LOW z PD7. **(25)**

Bodové hodnotenie 0-30.

1. Cyklickým pripájaním a odpájaním log. úrovne LOW na INT0 sa striedavo zapína a vypína LED. Realizovať pomocou prerušenia INT0. **(26)**
2. Ak sa pripojí na INT0 log. úroveň LOW, LED bude svietiť. Ak sa na INT1 pripojí LOW, LED zhasne. Riešiť pomocou prerušenia INT0 a INT1. **(28)**
3. Najskôr zakáž WDT a nastav blikanie LED s frekvenciou 0,5 Hz. Potom, povoľ WDT a pozoruj blikanie LED. Ak led nebliká správne (frekvenciou 0,5 Hz), uprav program tak, aby bola frekvencia blikania 0,5Hz. **(30)**

Bodové hodnotenie 0-35.

1. Nastav parametre sériového prenosu: 9600 baudov, 8 bitov, žiadna parita, 1 stop bit. Vysielaj 5x po sebe znak (napr. "A„). Skontroluj vyslanie znakov pomocou programu *Terminal* alebo *Data Visualizer*. **(31)**
2. Napíš program, ktorý pri prijatí znaku "Z" alebo "z", rozsvieti LED. Pri prijatí znaku "V" alebo "v", LED zhasne. Ak mikrokontroler prijme akýkoľvek iný znak, odošle ho späť a súčasne pošle aj znak "?". **(32)**
3. Napíš program na meranie teploty čipu a vypíš hodnotu teploty do programu *Terminal*. Ovládaj LED (svieti/nesvieti) podľa hodnoty teploty. **(33)**
4. Napíš program, ktorý pri prijatí slova "Zapni" zapne LED a pri prijatí slova "Vypni" LED vypne. Ak príjme ľubovoľnú inú skupinu piatich znakov, odošle ju späť a pridá znak "?". Pre riadenie komunikácie použi riadiace kódy STX a ETX z ASCII tabuľky. Príjem znakov rieš cez prerušenie. **(34)**
5. Použi časovač TC0, aby sa každých 5ms vygenerovalo prerušenie. Toto prerušenie použi na periodické blikanie LED, pričom LED bude 200ms svietiť a 800ms nebude svietiť. **(35)**

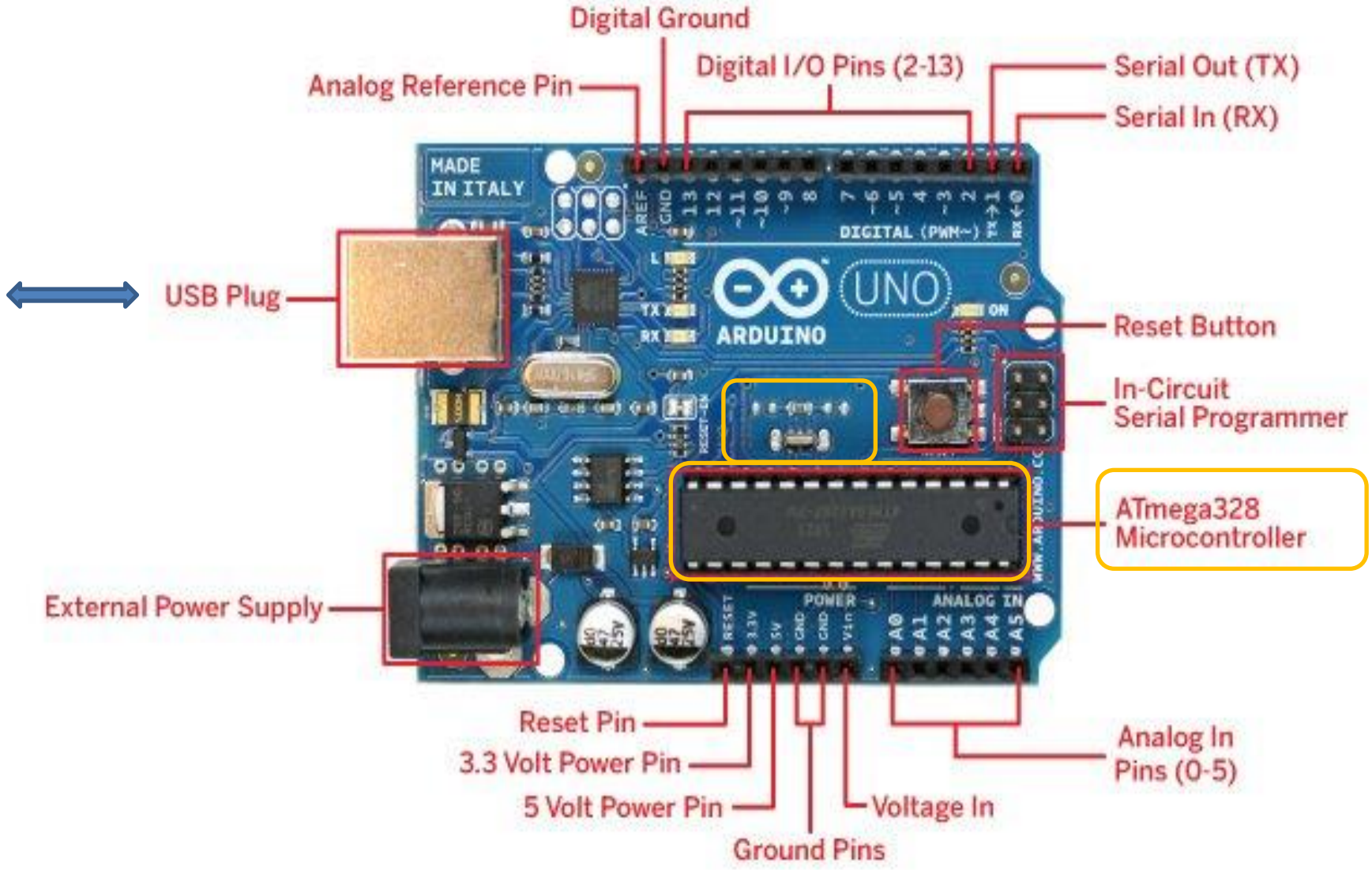
Bodové hodnotenie zápočtu 0-40.

1. Napíš program na ovládanie LED. Ak MCU príjme znak:"1„ - LED sa rozsvieti na 500ms a potom zhasne; "2„ - LED sa rozsvieti na 1000ms a potom zhasne; "3„ - LED sa zapne a bude svietiť; "4" - LED zhasne a ostane zhasnutá. Časovanie realizuj pomocou časovača TC0. **(36)**
2. Nastav časovač TC0 na režim Fast PWM a napíš program, ktorý umožní meniť jas externej LED (s rezistorom v sérií) od 0% do 100%, na základe prijatého textu v rozsahu „000“ až „100“. **(40)**
3. Napíš program na meranie napätia na AD vstupe. Uprav hodnotu meraného napätia tak, aby menila jas externej LED pomocou PWM v intervale od 0% do 100%. Na zmenu napätia použi potenciometer. Hodnotu napätia a odpovedajúcu hodnotu jasu zobraz v programe *Terminal*. (40+10)

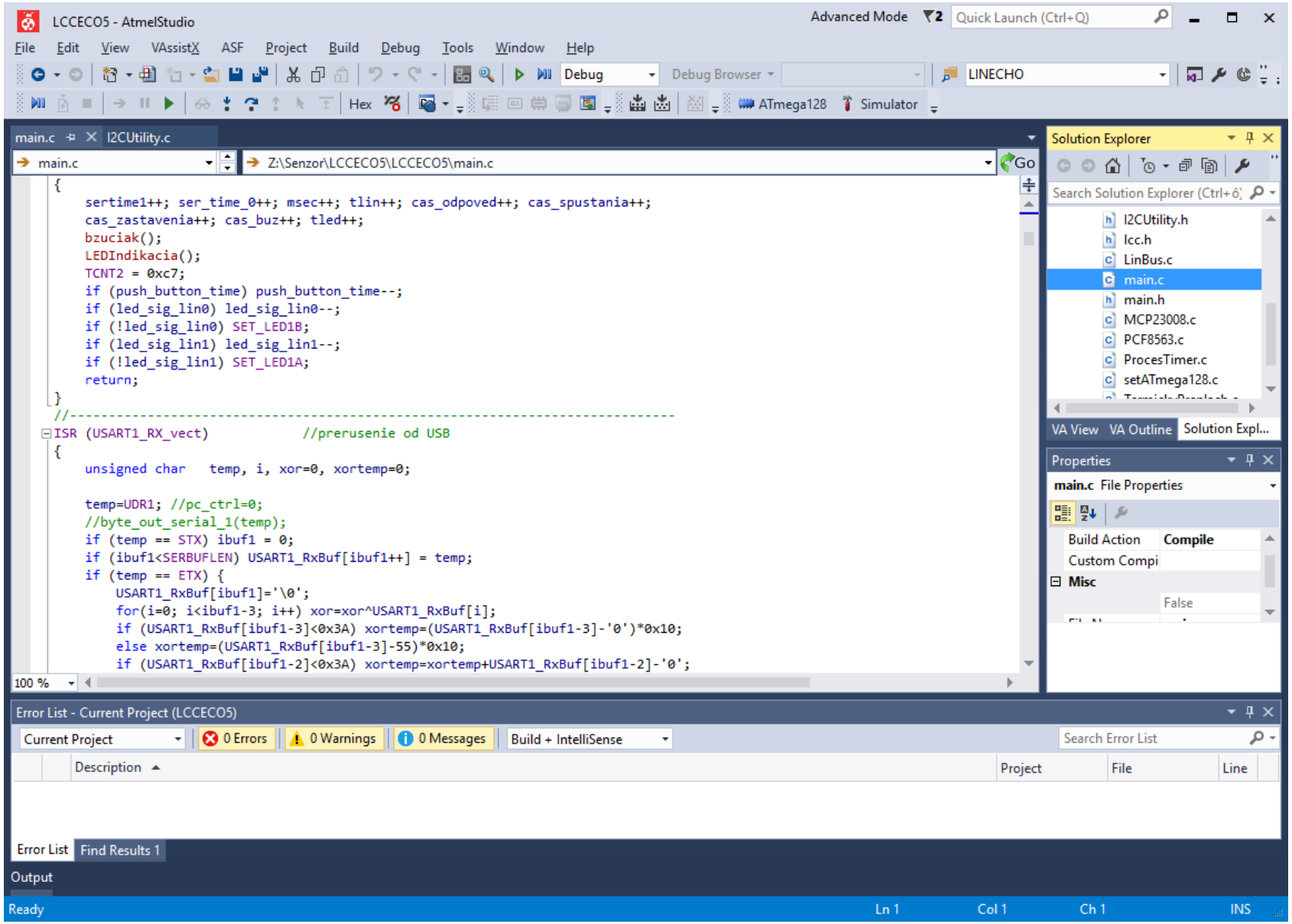
❑ Plán prednášok a laboratórnych cvičení

Týždeň	Prednášky	Cvičenia
1.	Úvod do predmetu. Číselné sústavy v elektronike, kódovane čísel. Jazyk C. Úvod do digitálnej elektroniky	Inštalácia: - IDE *Microchip Studio* - *driver* pre *Arduino kit*
2.	Polovodičové pamäte	Práca s *Microchip Studio*, definovanie projektu, preklad projektu
3.	ATmega328, pamäť, generovanie hodinových signálov, manažment napájania, reset	Práca s *Microchip Studio* simulácia projektu, porty
4.	Vstupno – výstupné porty	Použitie vstupno-výstupných portov
5.	Externé a interné prerušenia, Watchdog časovač	Práca s prerušeniami Výber zadania na zápočet
6.	Sériové komunikačné prostriedky USART	Použitie USART. Práca na zadaní
7.	TC0 - Normálny mód	Práca s TC0. Práca na zadaní
8.	TC0 – Rýchly PWM mód, Fázovo korektná a CTC PWM	Práca s TC0. Práca na zadaní
9.	Sériové komunikačné prostriedky SPI	Použitie SPI. Práca na zadaní
10.	AC komparátor, AD prevodník	Použitie A/D. Práca na zadaní
11.	EEPROM, *FUSE* bity	Programovanie EEPROM. Práca na zadaní
12.	Sériové komunikačné prostriedky - I2C	**Zápočet - Praktická práca s MCU**
13.	Predtermín	**Zápočet - Praktická práca s MCU**

- Úspešné absolvovanie predmetu ZMT znamená zvládnutie programovania hardvérových zariadení mikrokontroléra **AVR - ATmega328P** v **jazyku C** s využitím integrovaného vývojového prostredia **IDE *Microchip Studio*** a vývojovej dosky (*Arduino UNO*).

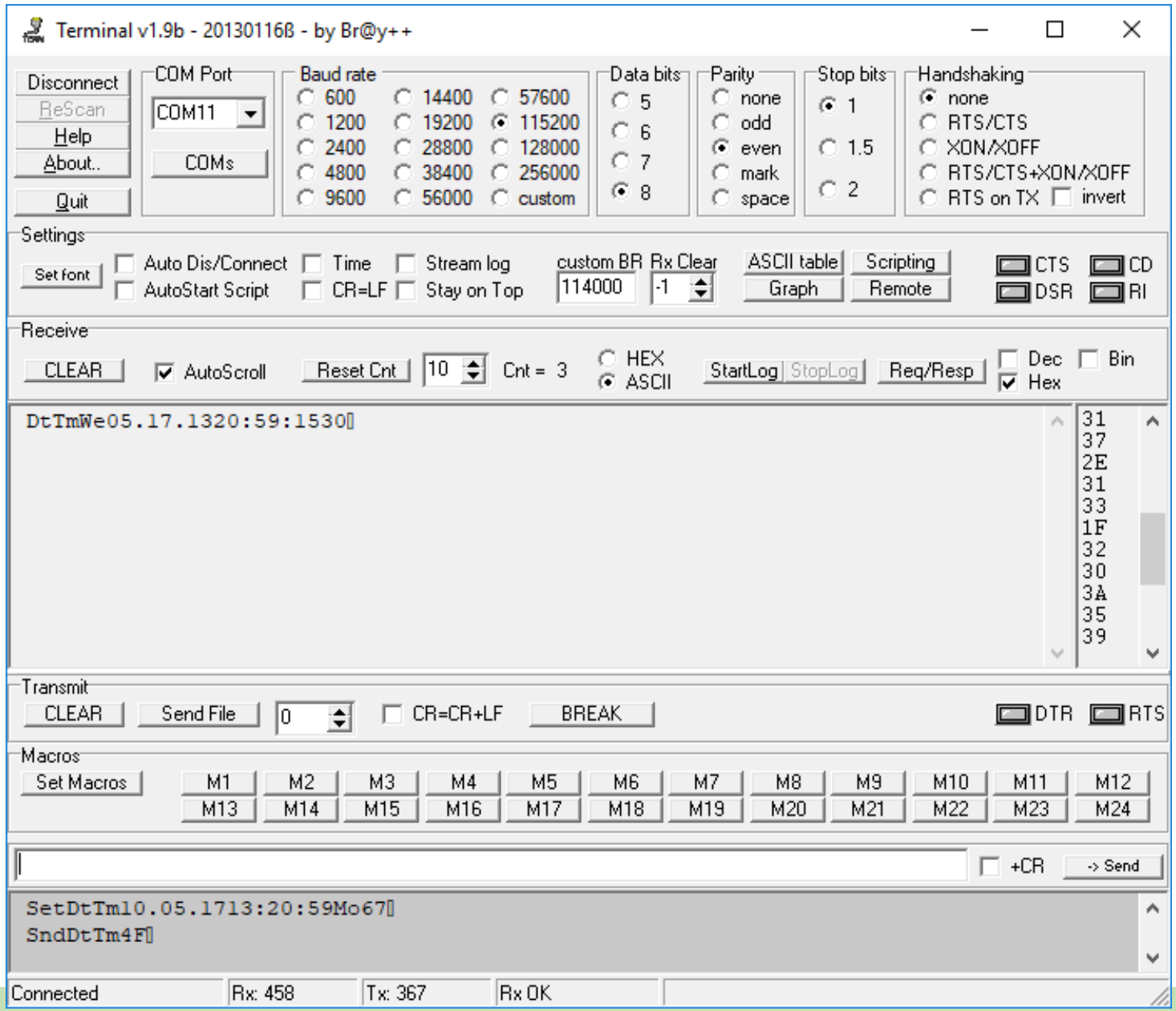


- Integrované vývojové prostredie *Microchip Studio* – voľba inštalácie len pre 8-bitové AVR mikrokontrolery

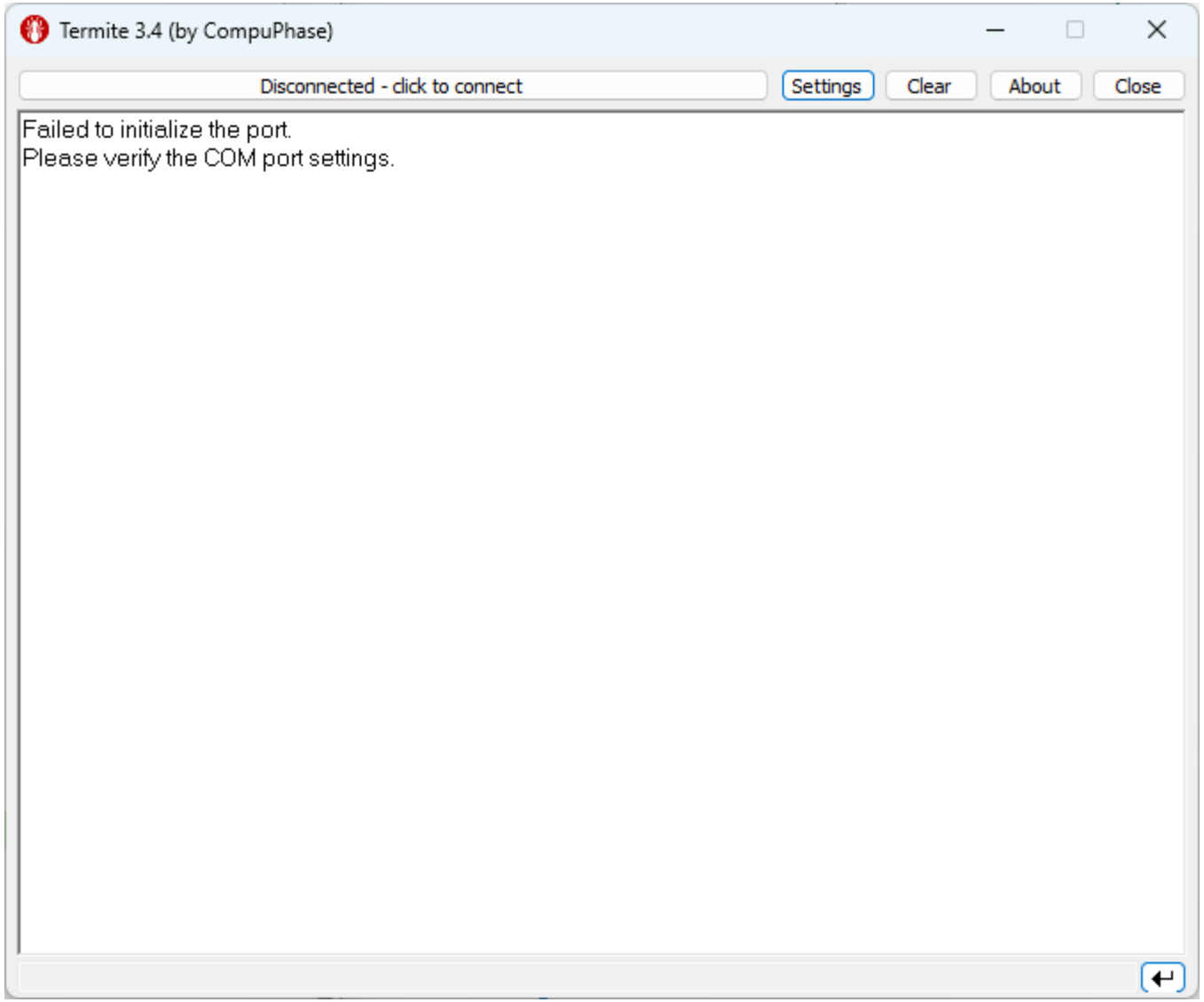


https://www.microchip.com/en-us/tools-resources/develop/microchip-studio

- Komunikačný program *Terminal* - https://sites.google.com/site/terminalbpp/



- Komunikačný program *Termite 3.4* - *https://www.compuphase.com/software_termite.htm*



- Avr-libc - http://nongnu.org/avr-libc/user-manual/index.html

avr-libc: Modules

nongnu.org/avr-libc/user-manual/modules.html

Main Page | User Manual | Library Reference | FAQ | Example Projects

Modules

Here is a list of all modules:

- <alloca.h>: Allocate space in the stack
- <assert.h>: Diagnostics
- <ctype.h>: Character Operations
- <errno.h>: System Errors
- <inttypes.h>: Integer Type conversions
- <math.h>: Mathematics
- <setjmp.h>: Non-local goto
- <stdint.h>: Standard Integer Types
- <stdio.h>: Standard IO facilities
- <stdlib.h>: General utilities
- <string.h>: Strings
- <time.h>: Time
- <avr/boot.h>: Bootloader Support Utilities
- <avr/cpufunc.h>: Special AVR CPU functions
- <avr/eeprom.h>: EEPROM handling
- <avr/fuse.h>: Fuse Support
- <avr/interrupt.h>: Interrupts
- <avr/io.h>: AVR device-specific IO definitions
- <avr/lock.h>: Lockbit Support

avr-libc: A simple project

nongnu.org/av 67 %

avr-libc 2.0.0

Standard C library for AVR-GCC

AVR Libc Home Page | AVR Libc Development Pages

Main Page | User Manual | Library Reference | FAQ | Example Projects

A simple project

Demo projects

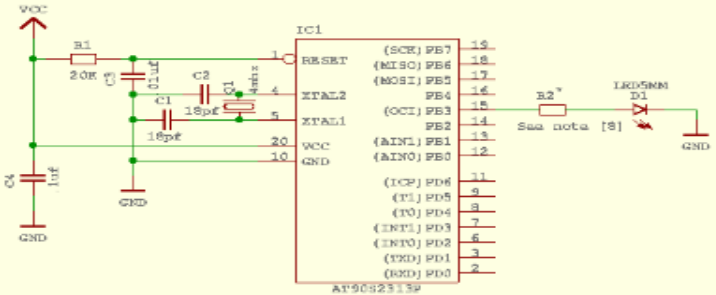
At this point, you should have the GNU tools configured, built, and installed on your system. In this chapter, we present a simple example of using the GNU tools in an AVR project. After reading this chapter, you should have a better feel as to how the tools are used and how a `Makefile` can be configured.

The Project

This project will use the pulse-width modulator (PWM) to ramp an LED on and off every two seconds. An AT90S2313 processor will be used as the controller. The circuit for this demonstration is shown in the schematic diagram. If you have a development kit, you should be able to use it, rather than build the circuit, for this project.

Note

Meanwhile, the AT90S2313 became obsolete. Either use its successor, the (pin-compatible) ATtiny2313 for the project, or perhaps the ATmega8 or one of its successors (ATmega48/88/168) which have become quite popular since the original demo project had been established. For all these more modern devices, it is no longer necessary to use an external crystal for clocking as they ship with the internal 1 MHz oscillator enabled, so C1, C2, and Q1 can be omitted. Normally, for this experiment, the external circuitry on /RESET (R1, C3) can be omitted as well, leaving only the AVR, the LED, the bypass capacitor C4, and perhaps R2. For the ATmega8/48/88/168, use PB1 (pin 15 at the DIP-28 package) to connect the LED to. Additionally, this demo has been ported to many different other AVRs. The location of the respective OC pin varies between different AVRs, and it is mandated by the AVR hardware.



Schematic of circuit for demo project