Факултет Електроника и Автоматика
Катедра Компютърни Системи и Технологии

Дипломна Работа на тема:
“Проектиране на микроконтролерен модул с цел използване на AVR/PIC асемблер”

Дипломант:
Веселин Станчев
ФН: 614872
спец. КСТ

Научен Ръководител:
доц. д-р инж. Мария Маринова

Пловдив 2023 г.

Съдържание

Списък с използвани термини

Термин	Описание
ISA-Instruction Set Architecture	Сет от достъпни инструкции на процесора
RISC-Reduced Instruction Set	Редуциран сет от инструкции, като по-сложните операции биват изпълнени от основните инструкции
GNU Assembler	Свободен асемблер част от GNU проекта. Наличен в GNU Binutils пакета
AVR Assembler	Свободен асемблер. Поддържа множество AVR RISC базирани микроконтролери
PIC Assembler	Асемблер за PIC базирани микроконтролери
GNU Compiler Collection	Колекция от компилатори за ASM, C, C++, Ada
GPUTILS	Сет от инструменти за PIC микроконтролерите

Списък на фигурите

Увод

С развитието на езиците от високо ниво, като например C++, Python, Ruby, се развиват и Assembler-ните езици, като например NASM, GNU Assembler.

Assembler-ните езици са изключително важни при програмирането на:

- - микропроцесори
- - микро контролери
- - вградени устройства (embedded devices) изградени на основата на микро контролерите
- операционни системи от тип Real Time.

За различните видове микроконтролери съществуват различни асемблери.
За AVR-базираните микроконтролери като например:

- - ATtiny85
- - Atmega 16 A
- - rp2040

За AVR-базираните микроконтролери съществуват следните свободни асемблери:

- GNU Assembler
- AVR Assembler
- RISC-V Assembler

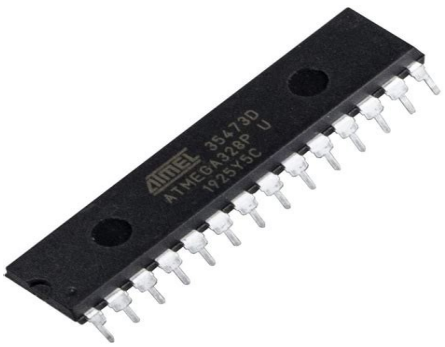
За PIC-базираните микроконтролери като напр.:

- pic16f84a
- pic16f877a
- pic10f320

За PIC-базираните микроконтролери съществуват следните асемблери:

- PIC Assembler

Фиг. У1 *Attiny13*



Фиг. У2 Atmega328

Фиг. У3 PIC16F84

Примери за микроконтролери и налични асемблери

Микроконтролери	Налични асемблери
Attiny13	AVR Assembler
Atmega328	GNU Assembler
PIC16F84A	PIC Assembler

У4 Микроконтролери и налични асемблери

Предпоставки за темата на дипломната работа

За предварителната подготовка при избор на темата на настоящата дипломна работа са налице следните предпоставки:

Проектирани AVR/PIC микроконтролерни модули (Приложение 1).

Използван софтуер за проектиране: Fritzing

Разработена дипломна работа за бакалавърска степен: Статична библиотека реализирана чрез GNU Assembler за ATtiny328P/RP2040

Асемблери според instruction set архитектурата

За различните видове instruction set architectures (ISA) съществуват различни асемблери. За x86_64 архитектурата може да бъде използван Netwide assembler. За ARM архитектурата може да бъде използван GNU Assembler. За PIC архитектурата може да бъде използван PIC Assembler.

Това е указано в таблицата по-долу:

Сет от инструкции (ISA)	Асемблер
x86_64	Nerwide assembler
ARM	GNU Assembler, AVR Assembler
RISC-V	RISC-V Assembler
PIC RISC	GNU PIC Assembler

У5 Асемблери за всеки вид архитектура (ISA)

Цел на дипломната работа:

Да бъде проектиран микроконтролерен модул с използване на AVR-базиран микроконтролер и PIC-базиран микроконтролер с цел комуникация между двата микроконтролера на асемблерно ниво. Платката ще може да бъде използвана за учебни и университетски цели. Проектът е изцяло open-source с цел по-доброто бъдещо развитие.

Причини за изпълнение на поставената цел:

- **Използване на два свободни асемблера.**
- **Използване на микроконтролери с малки размери с цел преносимост на микроконтролерния модул.**
- **Реализиране на софтуерната част за микроконтролерния модул като самостоятелна софтуерна библиотека.**
- **Намалямане на големината на elf файла чрез използване на асемблер.**
- **Изцяло практическа насоченост на проекта.**

Поставени задачи за изпълнение на целта:

- Обзор на съществуващи микроконтролерни модули с AVR/PIC базирани микроконтролери
- Изисквания към проекта
- Анализ на микроконтролерите които ще бъдат използвани за микроконтролерния модул
- Анализ на асемблерните инструкции които ще бъдат използвани
- Софтуерни инструменти за проектиране на микроконтролерният модул и програмиране
- Проектиране на микроконтролерния модул
- Програмиране на микроконтролерите
- Реализиране на кода като библиотека
- Постигнати резултати. Бъдещо развитие

Глави

I. Обзор на съществуващи микроконтролерни модули с AVR/PIC базирани микроконтролери.
II. Изисквания към проекта.
III. Анализ на микроконтролерите, които ще бъдат използвани за микроконтролерния модул
IV. Анализ на асемблерните инструкции, които ще бъдат използвани.
V. Софтуерни инструменти за проектиране на микроконтролерният модул и програмиране.
VI. Проектиране на микроконтролерния модул.
VII. Програмиране на микроконтролерите.
VIII. Реализиране на кода като библиотека.
IX. Постигнати резултати. Бъдещо развитие.

Приложения

Заключение

1. 1. Обзор на съществуващи микроконтролерни модули с AVR/PIC базирани микроконтролери

Въведение

В настоящата глава ще бъдат разгледани микроконтролерни модули (PCB), базирани върху AVR/PIC микроконтролери.

Съдържанието на главата се състои от следните подглави:

- **Предметна област**
- **Класификации на реализираните модули**
- **Анализ на модулите**
- **Резултати и обобщаване на резултатите**
- **Заключение**
- **Източници**

За всяки микроконтролерен модул ще бъдат описани:

- **обща характеристика на микроконтролерния модул**
- **микроконтролерите, които поддържа**
- **комуникационни протоколи, които поддържа**
- **интерфейсите, които поддържа**
- **асемблерите, на които могат да бъдат програмирани микроконтролерите**

Микроконтролерните модули, които ще бъдат анализирани са:

- **ANAVI MACRO PAD 2**
- **ANAVI LIGHT CONTROLLER**
- **AVR64DD32 BOARD**
- **RP2040 BOARD**

Ключови думи: P*CB, MCU, AVR/PIC, Assembly language*

I. Предметна област

Използването на свободни асемблери е изключително важно за разбирането на компютърните архитектури. Свободните асемблери са с отворен лиценз и могат да бъдат използвани за университетски цели. Могат да бъдат проектирани микроконтролерни модули чрез използването на свободен софтуер като например:

Fritzing

- ***LibrePCB***

Реализираните микроконтролерни модули може да бъдат програмирани на съвместим с микроконтролерите им асемблер. За програмирането на различните микроконтролери се използват различни асемблери в зависимост от техния сет от поддържани инструкции (ISA) и целта на проекта, който се постига чрез тях. Няколко вида асемблери са разгледани в [**Таблица 0.1]**.
Ще бъдат разгледани възможностите на различни реализирани микроконтролерни модули.

II. Класификации на реализираните модули

Класификация спрямо архитектурата (ISA) на микроконтролерите

RISC базирани

- **ANAVI MACRO PAD 2** - базирана на ATTiny 85 – AVR RISC архитектура
- **ANAVI LIGHT CONTROLLER** - базирана на ESP32 – RISC архитектура
  Raspberry Pi Pico - базирана на RP2040 – RISC архитектура

PIC базирани

- **AVR64DD32 Curiosity Nano -** базиран на AVR64DD32 – PIC базиран

III. Анализ на микроконтролерните модули

Модул ANAVI Macro Pad 2

Фиг. 1.1 Anavi Macro Pad 2

Обща характеристика на микроконтролерният модул

ANAVI Macro Pad 2 предоставя възможност за програмиране на два основни механични бутони, които могат да бъдат програмирани чрез Python-базиран фърмуер. Бутоните могат да бъдат използвани за стартирането на различни софтуерни приложения

- **Микроконтролери, които поддържа:**
  Използван е ATTiny85

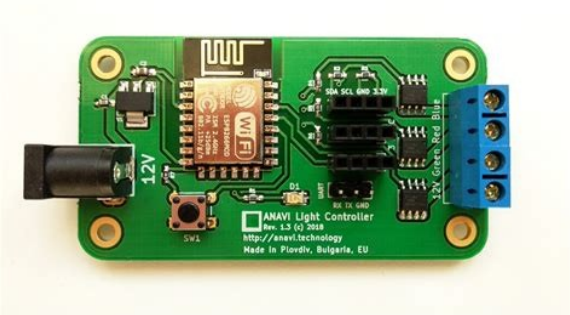
  Комуникационни протоколи, които поддържа:
  I2C

- **Интерфейси, които поддържа:**
  USB
  ISP

- **Описание на асемблерите, на които могат да бъдат програмирани микроконтролерите:**

  Асемблерите трябва да бъдат съвместими с AVR RISC архитектурата.
  Такива са:
  - **GNU Assembler**
  - **AVR Assembler**

Модул ANAVI Light Controller



Фиг. 1.2 Anavi Light Controller

Обща характеристика на микроконтролерния модул

ANAVI Light Controller предоставя възможност за използване на сензори за:

- светлина
- влажност
- температура

- **Микроконтролери ,които поддържа**:
  ESP32

  Комуникационни протоколи, които поддържа:
  I2C

- **Интерфейси, които поддържа**:
- USB
- WI-FI

- **Асемблери, на които могат да бъдат програмирани микроконтролерите**
  AVR Assembler

Модул AVR64DD32 Curiosity Nano

Фиг. 1.3 AVR64DD32 Curiosity Nano

Обща характеристика на микроконтролерният модул
Поддържа хардуерен дебъг. Базиран е на AVR64DD32. Поддържа UART.

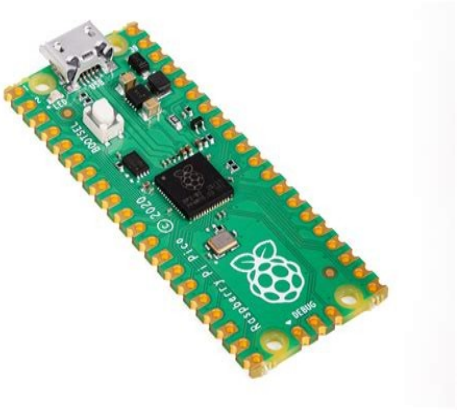
- **Описание на микроконтролерите, които поддържа**
- **AVR64DD32**
-
- **Комуникационни протоколи, които поддържа:**
  UART

- **Интерфейсите, които поддържа:**
- **USB**

Асемблери, на които могат да бъдат програмирани микроконтролерите

- **PIC Assembler**

Модул Raspberry Pi Pico



Фиг. 1.4 Raspberry Pi Pico

Обща характеристика на микроконтролерният модул

Raspberry Pi Pico е микроконтролерен модул базиран на RP2040, който се използва за различни проекти като например домашна автоматизация. Разполага с USB интерфейс за захранване и пренос на данни. Разполага с различни комуникационни интерфейси.

Описание на микроконтролерите, които поддържа
Използван е RP2040

Комуникационни протоколи, които поддържа:
I2C
UART
SPI

- **Описание на интерфейсите, които поддържа:**
  USB 1.1

- **Описание на асемблерите, на които могат да бъдат програмирани микроконтролерите**
  AVR Assembler
  GNU Assembler

IV. Резултати и обобщаване на резултатите

Анализът на модулите е обобщен чрез следната таблица:

Име	Поддържа микроконтролерите	Поддържа интерфейсите	Поддържа асемблерите
ANAVI Macro Pad 2	ATTiny85	ISP USB	AVR Assembler, GNU Assembler
ANAVI Light Controller	ESP-32	Wi-Fi USB	AVR Assembler
AVR64DD32 Curiosity Nano	AVR64DD32	USB	PIC Assembler
Raspberry Pi Pico	RP2040	USB	AVR Assembler, GNU Assembler

V. ЗАКЛЮЧЕНИЕ

Микроконтролери могат да бъдат програмирани на асемблер с цел ефективно използване на паметта им. Могат да бъдат използвани за различни проекти в сферата на компютърните архитектури. В следващата глава ще бъде направен нализ на микроконтролерите които ще бъдат използвани за реализацията на микроконтролерен модул. При програмирането на микроконтролерния модул ще бъде използван асемблер.

Използван източник **[1]**

2. Изисквания към проекта

Вследствие на направения обзор на съществуващи микроконтролерни модули и поставената цел на проекта, биват поставени следните изисквания:

Изисквания към проектирането на микроконтролерния модул:

- трябва да бъдат избрани AVR RISC-базиран микроконтролер както и PIC-базиран микроконтролер, между които да бъде извършена комуникацията
- трябва да бъдат избрани интерфейси, с които ще разполага микроконтролерния модул
- трябва да бъде избран софтуер за проектирането на микроконтролерния модул.
    - Възможен софтуер може да бъде:
    - LibrePCB
    - Fritzing
- трябва да бъде създадена блокова схема на микроконтролерния модул
- трябва да бъдат избрани подходящи допълнителни хардуерни компоненти
- с помощта на избрания софтуер за проектиране трябва да бъдат създадени:
    - символи на хардуерните компоненти (ако не са налични в основната библиотека с компоненти)
    - footprints на хардуерните компоненти (ако не са налични в основната библиотека с компоненти)
    - трябва да бъдат описани пиновете за комуникация между микроконтролерите
    - схема на микроконтролерния модул
    - трябва да бъде избрано разположение на хардуерните компоненти; размерите на микроконтролерния модул да бъде малък
- трябва да бъде редуциран фоновия шум на микроконтролерният модул

Възможности за реализация на микроконтролерният модул

Съществуват различни възможности за комуникация между два микроконтролера:

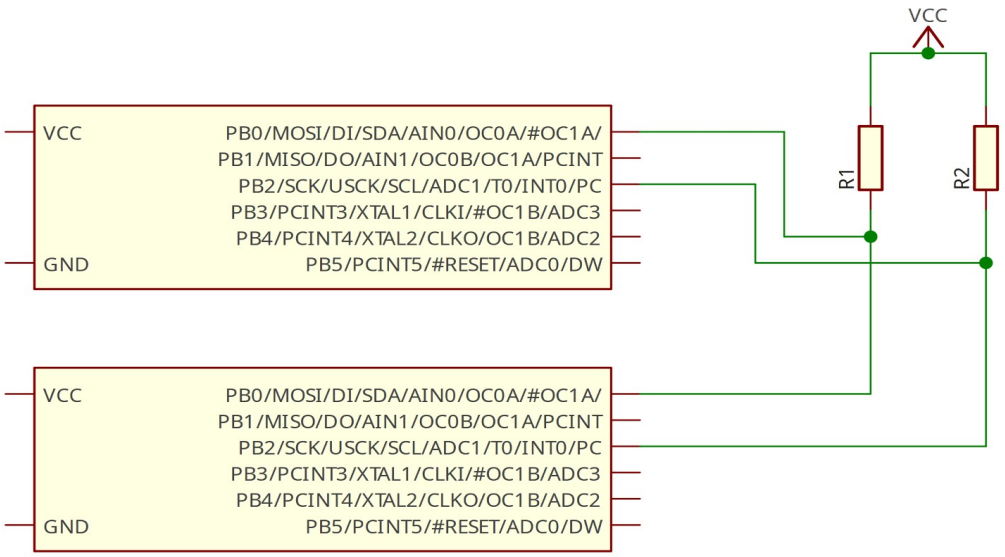
- Между два AVR RISC - базирани микроконтролери

  Могат да бъдат програмирани чрез:

  AVR Assembler

  GNU Assembler

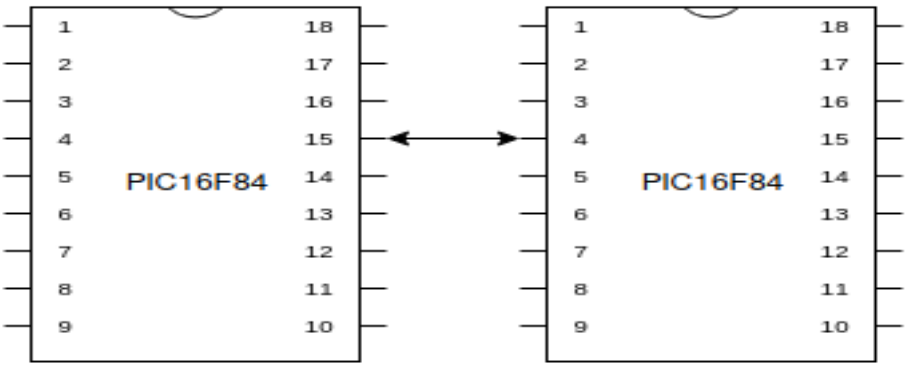
  Пример чрез използване на два микроконтролера ATTiny85:



\

Фиг. 2.1 Схема ATTiny85 UART

- Между два PIC - базирани микроконтролери



Фиг. 2.2 Схема PIC - базирани микроконтролери

Могат да бъдат програмирани чрез:

PIC Assembler

- Между AVR и PIC - базиран микроконтролер

Могат да бъдат програмирани чрез:

PIC Assembler

AVR Assembler

Възможни протоколи за използване:

- I2C
- UART
- UEXT
- vrstanchev

Изисквания към софтуерното осигуряване на микроконтролерния модул:

- трябва да бъдат избрани асемблери съвместими с архитектурата на микроконтролерите
- трябва да бъде реализиран протокол за комуникация между микроконтролерите
- трябва да бъде избран протокол за комуникация, спрямо който да бъде написан кодът
- кодът за микроконтролерите трябва да бъде реализиран чрез функции - labels
- функциите трябва да бъдат съобразени със избрания протокол за комуникация между микроконтролерите
- трябва да не се допуска излишно обръщение към регистрите
- трябва да бъде изложена логиката на компилацията
- трябва да бъде автоматизирана компилацията
- трябва кодът да бъде реализиран и като софтуерна библиотека поради:
    - бъдещото развитие на проекта
    - преносимост

Използван източник **[2]**

3. Анализ на микроконтролерите, които ще бъдат използвани за микроконтролерния модул

В предишната глава бяха разгледани съществуващи микроконтролерни модули, за които могат да бъдат използвани асемблер за avr или pic assembler.

За реализирането на микроконтролерния модул е нужно да бъдат използвани микроконтролери, които могат да бъдат програмирани чрез:

- **AVR Assembler**
- **PIC Assembler**

Избраните микроконтролери за реализация на проекта са :

Фиг. 3.2 Attiny85

Фиг. 3.1 PIC10F320

Анализ на Attiny85

ATTiny85 може да бъде използван за множество софтуерни и хардуерни проекти.

- домашна автоматизация

- микроконтролерни модули за игри

- летателни апарати

ATTiny85 разполага със следните възможности:

- 8Kb Flash памет
- 512 байта EEPROM памет
- 2 PWM канала
- 8 битов таймер
- комуникация през универсален сериен интерфейс (USI)
- възможност за I2C комуникация чрез пиновете PB0,PB2

Attiny85 е 8-битов микроконтролер, базиран върху AVR-RISC архитектура. Поддържа 32 регистъра с общо предназначение.

R0	0x00
R1	0x01
R2	0x02
...	
R13	0x0D
R14	0x0E
R15	0x0F
R16	0x10
R17	0x11
...	
R26	0x1A
R27	0x1B
R28	0x1C
R29	0x1D
R30	0x1E
R31	0x1F

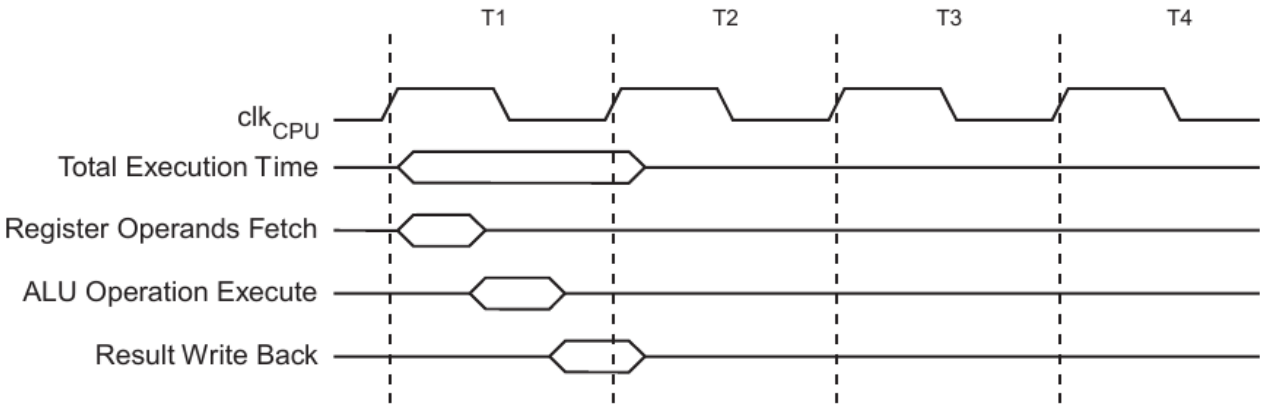
Фиг. 3.3 AVR-RISC регистри

Примери за регистър с общо предназначение:

- r0
- r1
- r2

Примери за регистър със специално предназначение:

- r15
- r16
- r14



Фиг. 3.4 Обработка на инструкция от процесора

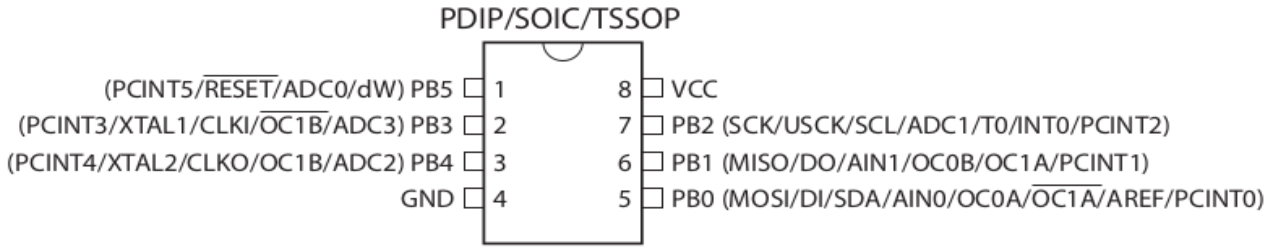
Както става ясно от фиг. 3.1 процесорът обработва инструкциите чрез:

- прихващане на инструкцията
- декодиране на инструкцията
- изпълнение на инструкцията

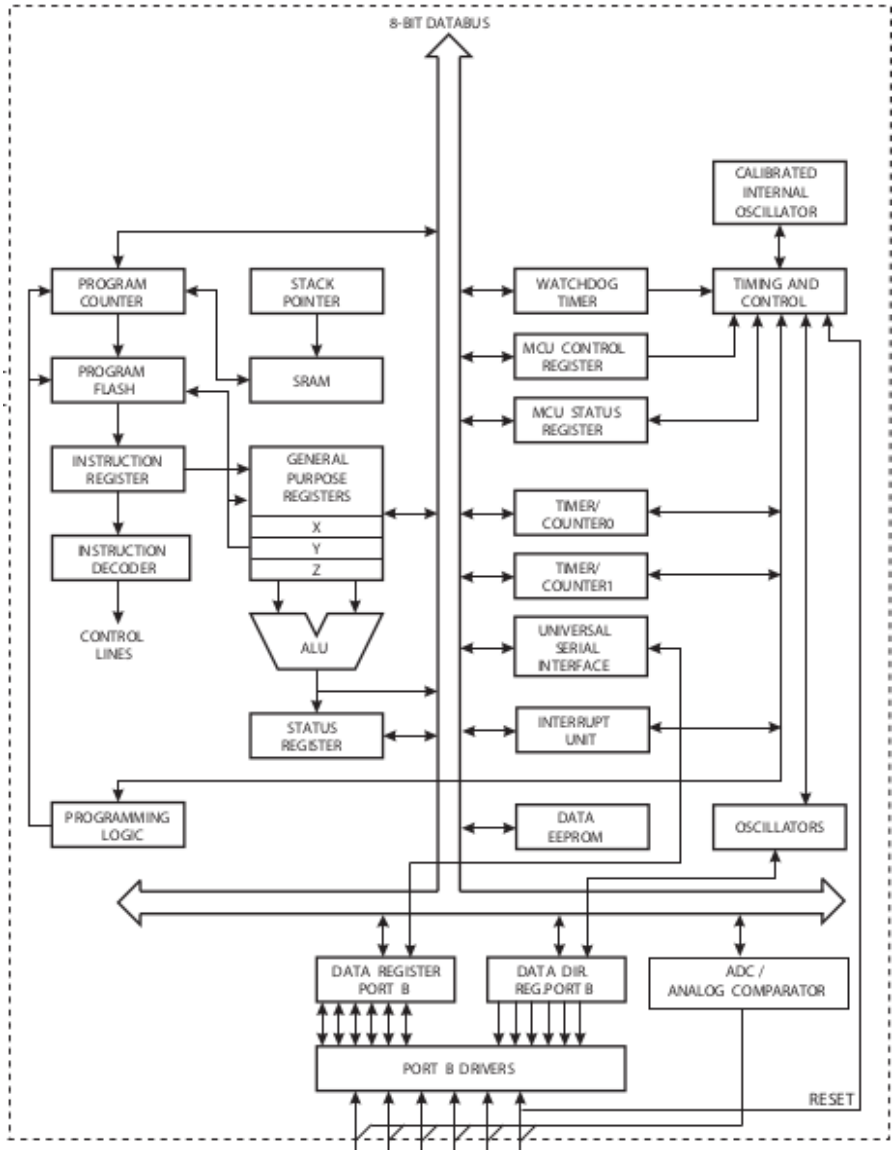
За програмирането му могат да бъдат използвани следните асемблери:

- **GNU Assembler**
- **AVR Assembler**

Следните пинове са достъпни за използване:



Фиг. 3.5 Пинове на ATTiny85



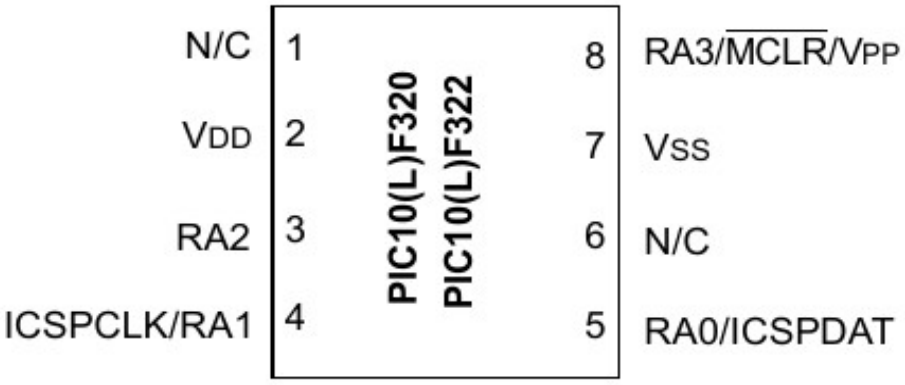
Фиг. 3.6 Блокова схема на ATTiny85

Анализ на PIC10F320

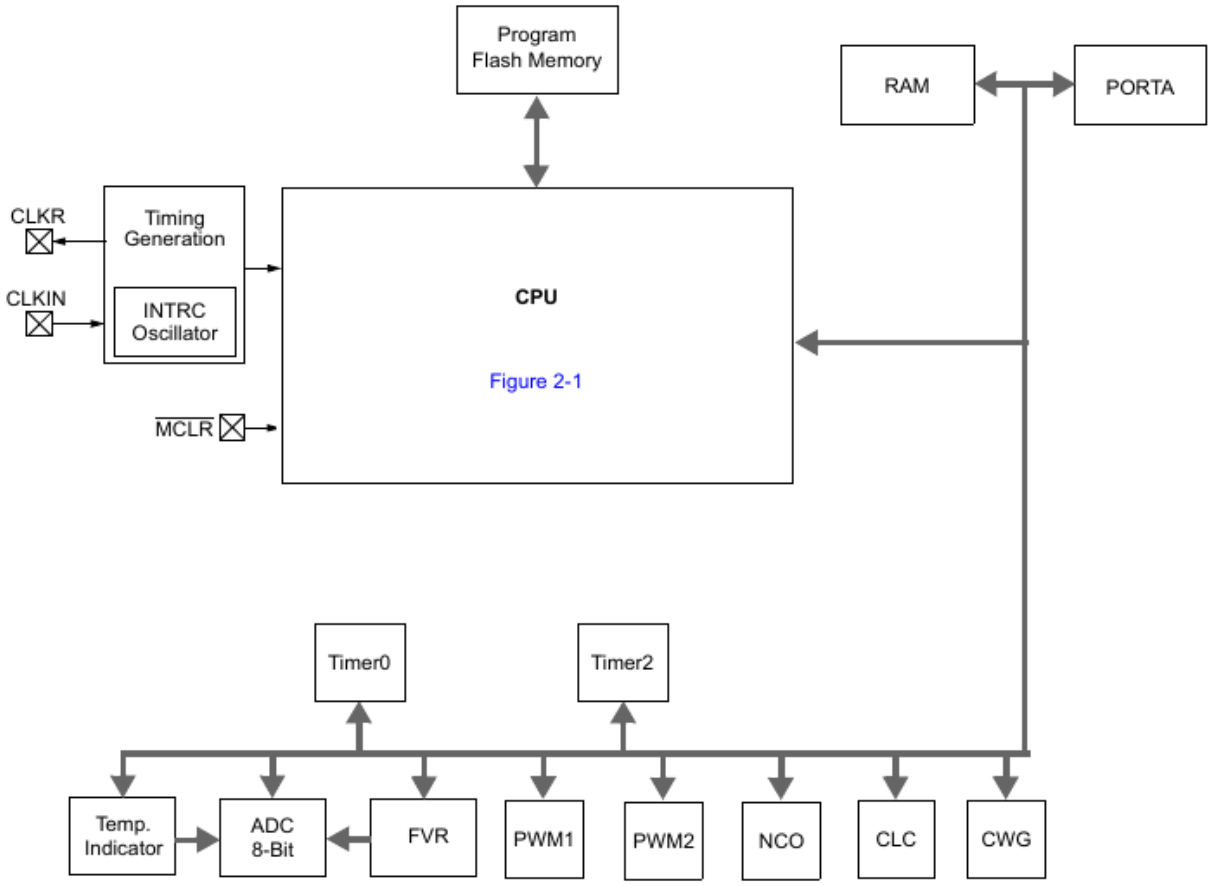
PIC10F320 разполага със следните възможности:

- 128 байта памет
- 2 PWM канала
- 8 битов таймер
- ADC-конвентер
- комуникация през вътрешен интерфейс (ICSP)
- waveform генератор
- възможност за комуникация между хост компютъра и микроконтролера

Следните пинове са достъпни за използване:



Фиг. 3.7 Пинове на PIC10F320



Фиг. 4.8 Блокова схема на PIC10F320

Използвани източници [2], [3]

4. Анализ на асемблерните инструкции, които ще бъдат използвани (ISA)

Във връзка с поставените изисквания към софтуерното осигуряване на микроконтролерния модул е необходимо да се анализират достъпните за използване инструкции на микроконтролерите, чрез които ще бъде изграден микроконтролерният модул.

За ATTiny85 са достъпни следните видове инструкции:

- Аритметико-логически инструкции
- инструкции за разклоняване на потока от данни
- инструкции за работа с битове
- инструкции за пренос на данни

За PIC10F320 са достъпни следните видове инструкции:

- инструкции за разклоняване на потока от данни
- инструкции за работа с битове
- инструкции за работа с байтове

ATTiny85:

Аритметико-логически инструкции:

Пример ADD
LDI r0, #4
LDI r1,#5
ADD r0,r1

Пример SUB
LDI r0, #7
LDI r1,#4
SUB r0,r1

Побитови операции:
Пример AND
LDI r2,#1
LDI r3, #0
AND r2,r3

Пример OR
LDI r4,#1
LDI r5, #0
OR r4,r5

ATTiny85:

Инструкции за работа с битове:
Пример присвояване стойност на бит SBI
LDI r0, #4
LDI r1,#5
ADD r0,r1

Пример изчистване стойност на бит CBI
LDI r0, #7
LDI r1,#4
SUB r0,r1

Инструкции за разклоняване на потока от данни:
Пример за сравнение cp
LDI r2,#5
LDI r3, #5
CP r2,r3

След като стойностите в регистрите са сравнени логиката може да бъде разклонена BREQ

Пример OR
LDI r4,#1
LDI r5, #0
CP r4,r5
BREQ label1

PIC10F320:

Както се вижда от анализа в трета глава, микроконтролерът PIC10F320 има PORTA
За да може PORTA да работи са нужни три регистъра с общо предназначение:

Инструкции за работа с битове:

Пример присвояване стойност на бит BSF
BSF «0b000001», «0h9877»

Пример изчистване стойност на бит BCF
BCF «0b000001», «0h9877»

Пример изчистване стойност на бит BCF
BCF «0b000001», «0h9877»

Пример изчистване стойност на бит BCF
BCF «0b000001», «0h9877»

Използвани източници [2], [3]

5. Софтуерни инструменти за проектиране на микроконтролерният модул и програмиране

За реализиране на проекта е избран свободен сет от инструменти с отворен код или са част от GNU проекта. Сетът от инструменти е с терминален интерфейс с цел по-удобна работа.

За реализацията на софтуерната част на проекта са избрани следните инструменти:

- Git
- GNU Make
- avra
- gputils
- gcc
- avrdude
- Vim
- ar
- gplib

За реализацията на хардуерната част на проекта е избран следният инструмент:

- LibrePCB

GIT

Git е разпределена система за контрол на версиите с отворен код, която може да бъде използвана за проследяване на промените за всеки тип файл. Използван е за проследяване на файловете в настоящия проект. Пример за използването й е изложен по-долу:

- git clone https://github.com/vrstanchev/Degrees.git – клониране на отдалеченото хранилище
- cd Degrees – достъпване на хранилището
- git add . - добавяне на файловете на проекта към хранилището
- git commit -m “Master Degree” - потвърждаване на направените промени във файловете
- *git push origin main* - Изпращане на направените промени към отдалеченото хранилище

GNU MAKE

GNU Make е система за автоматизация на компилацията или тестването на програма. Част е от GNU проекта.
Може да бъде използван за множество програмни езици, като например:

- C
- C++
- Assembler
- BASH

Може да бъде използвана не само за автоматизация на програмирането, но и за автоматизация на различни услуги.

Всеки makefile се състои от следните компоненти:

цели (targets)
- файлове, нужни за постигане на целта
- команда или поредица от команди които се изпълняват в рамките на целта

Пример за използването й е изложен по-долу:

цели: add,commit, push
нужни файлове: file1
примерна команда: git add file1

```
add: file1
    git add file1

commit: file1
        git commit -m “Msg“

push:file1
     git push origin main
```

AVRA

avra е инструмент за компилиране на avr assembler.

За да бъде получен краен .hex файл се изпълнява следното:

avra myfile.asm

За да бъде получен обектен файл .o от файла myfile.asm се изпълнява следното:

avra -o myfile.o myfile.asm

Кодът може да бъде реализиран и като самостоятелна статична библиотека,

чрез архив от обектни файлове.

GNU C Compiler

Gcc е компилатор, част от GNU Compiler Collection, който може да бъде използван за:

- C
- C++
- Assembler

Може да бъде използван за C-компилация по следния начин:

gcc -c -o myfile.o myfile.c – получаване на обектен файл
gcc myfile.o -o myfile – създаване на краен изпълним файл
./myfile – изпълняване

Може да бъде използван и за създаване на краен файл от асемблерски обектен файл:

gcc myasm.o -o myfile

GPUTILS

GPUTILS е сет от инструменти за програмиране на PIC микроконтролери. Състои се от :

- gpasm – използва се за компилиране на PIC Assembler
- gpdasm – използва се за декомпилиране на PIC Assembler
- gplink – линкер за PIC Assembler
- gplib – архиватор на обектни файлове с цел създаване на библиотека

Архиваторът на обектни файлове за GNU Assembler е ar

Съвместим е с множество PIC-базирани микроконтролери като например:

- PIC10F200
- PIC16F84
- PIC16F877A

Vim

Vim е текстов редактор с терминален интерфейс, който може да бъде използван на всички ОС. Разполага с няколко режими:

- INSERT – за въвеждане
- NORMAL – за редакция
- SEARCH / – за търсене или търсене и заместване
- REPLACE – за заместване
- VISUAL – за редакция
- VISUAL BLOCK– за редакция на множество редове

GNU AR

ar е архиватор на обектни файлове, който може да бъде използван за създаването на статични библиотеки. Статичните библиотеки може да бъдат базирани на:

- C/C++
- GNU Assembler
- AVR Assembler

GPLIB

gplib е архиватор за статични библиотеки, част от GPUTILS. Може да бъде използван за PIC Assembler.
Пример за използване:

```
gpasm -c pic_func.asm
gplib libpic_func.a pic_func.o
```

AVRDUDE

AVRDUDE е инструмент за качване на компилирания краен файл за микроконтролер

LibrePCB

LibrePCB е сет от инструменти за проектиране на микроконтролерни модули, който има следните възможности :

- създаване и редактиране на символи и footprints на хардуерни компоненти
- създаване и редактиране на схеми за микроконтролерните модули
- създаване и редактиране на микроконтралерните модули създадени от схемите

Използван източник [4]

6. Проектиране на микроконтролерния модул

Изпълнение на изискванията към проектирането на микроконтролерния модул:

За реализацията на микроконтролерният модул са избрани микроконтролерите:

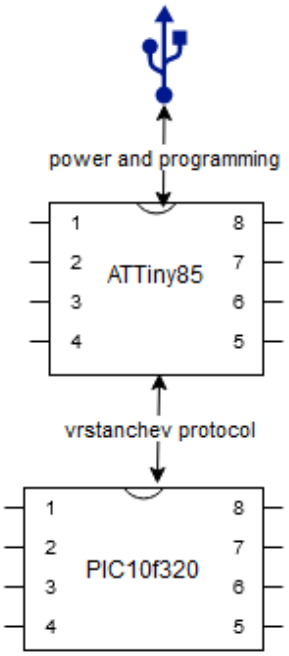
- **ATTiny85**
- **PIC10F320**

Интерфейси, с които ще разполага микроконтролерния модул:

- USB за захранване
- vrstanchev за комуникация

Избран софтуер за проектирането на микроконтролерния модул:

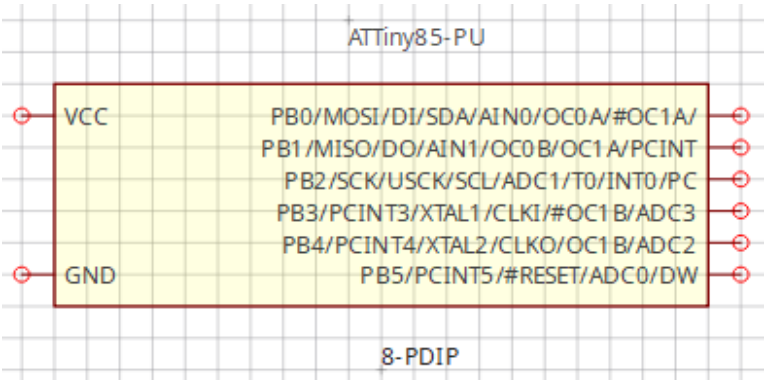
- LibrePCB



Фиг .6.1 Блокова схема на микроконтролерния модул

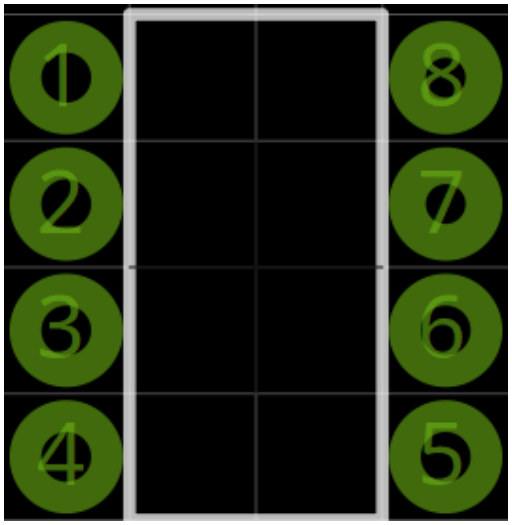
Хардуерни компоненти

Компонент Attiny85



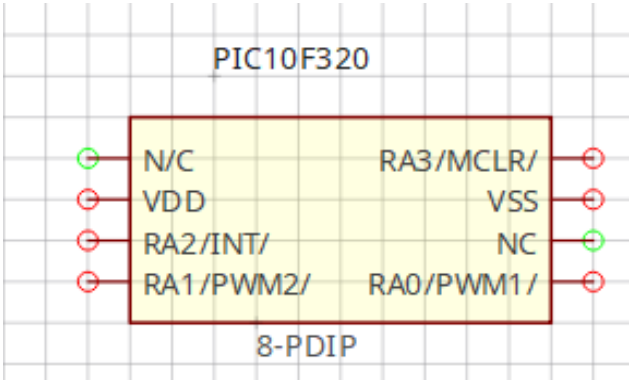
Фиг .6.2 Символ Attiny85

•



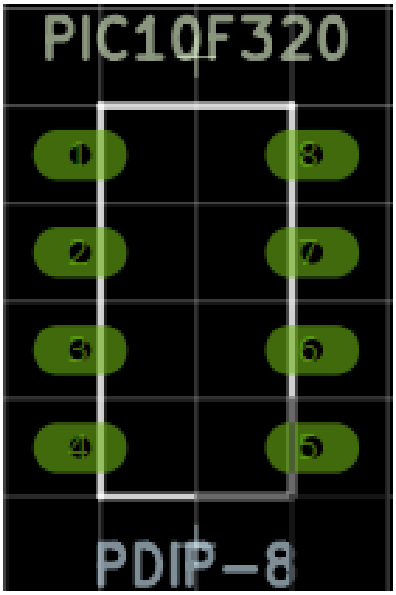
Фиг. 6.3 Footprint Attiny85

Компонент PIC10F320



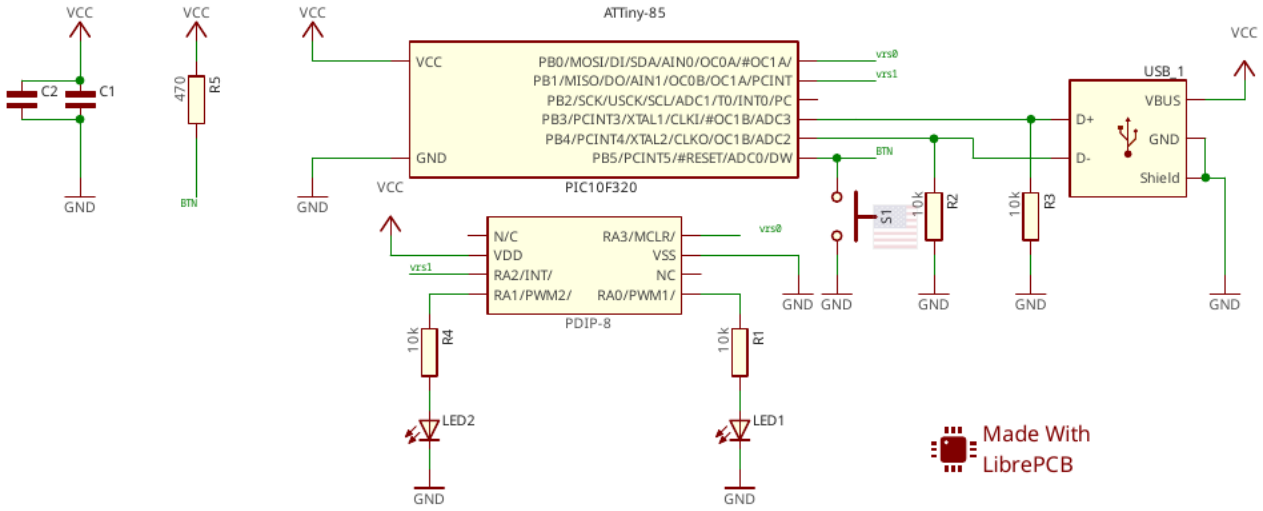
Фиг. 6.4 Символ PIC10F320

•



Фиг. 6.5 Footprint PIC10F320

Схема на микроконтролерния модул



Фиг. 6.6 Схема на микроконтролерният модул

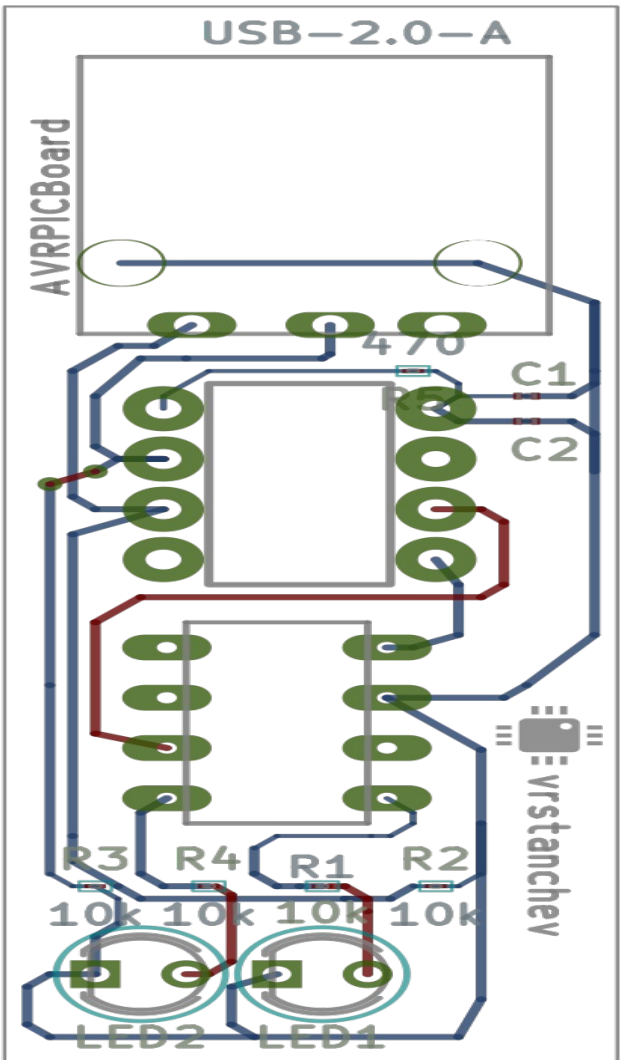
Описание на пиновете за комуникация между микроконтролерите

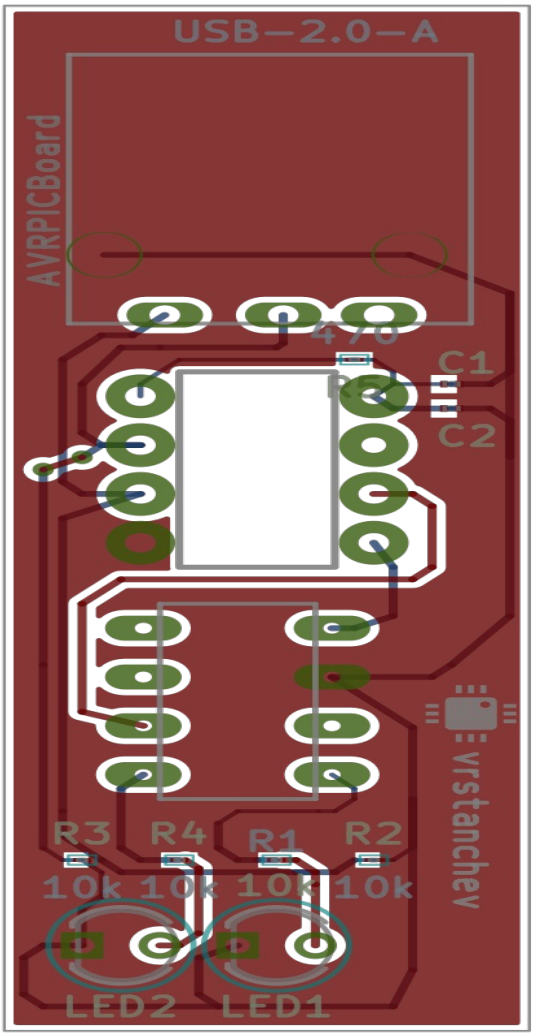
Интерфейс	Свързаност	Използвани пинове
USB	Attiny85 ↔ USB	PB3,PB4
vrstanchev	Attiny85 ↔ PIC10F320	PB0, PB1 → RA2, RA3

Избрани хардуерни компоненти

Quantity	Designators	Value	Device	Package
1	PIC1	PTH-8	PIC10F320	PIC10F320
1	AT1	PTH-8	ATTINY-85-PU	ATTINY-85-PU
2	"LED1	LED ⌀3.0x4.5/2.54mm Clear	LED	LED-THT-P254D300H450-CLEAR
	LED2"	LED ⌀3.0x4.5/2.54mm Clear	LED	LED-THT-P254D300H450-CLEAR
2	C1		C-0201	C-0201
	C2		C-0201	C-0201
4	"R1	10k	"R1	Resistor 0402 (01005)
	R2	10k	R2	Resistor 0402 (01005)
	R3	10k	R3	Resistor 0402 (01005)
	R4"	10k	R4"	Resistor 0402 (01005)
1	USB_1	USB-A-2.0	USB-2.0-A	USB-A-2.0

Разположение на хардуерните компоненти





Фиг. 6.7 Общо разположение

Фиг. 6.8 Редуциран шум чрез добавяне на GND

Фоновият шум на микроконтролерния модул е редуциран чрез:

- добавяне на GND
- пътеките между компонентите (trace) са под 45°

Използван източник [5]

7. Програмиране на микроконтролерите

Изпълнение на изисквания към софтуерното осигуряване на микроконтролерния модул:

- Избрани са следните асемблери:
  - AVR Assembler
  - PIC Assembler
- Избран е протокол за комуникация vrstanchev
- кодът за ATTiny85 ще бъде във функцията att_func()
- кодът за PIC10F320 ще бъде във функцията pic_func()
- функциите ще бъдат съобразени с протокола vrstanchev
- трябва да не се допуска излишно обръщение към регистрите
- трябва да бъде изложена логиката на компилацията
- трябва да бъде показана компилацията
- трябва кодът да бъде реализиран и като софтуерна библиотека поради:
  - бъдещото развитие на проекта
  - използване за различни проекти

Логика на кода

Софтуерната част на проекта се състои от:

`att_func`

`pic_func`

`vrstanchev`

Във функцията att_func ще бъде описано:

- в съответствие със схемата прави пиновете PB0 и PB1 на ATTINY-85 да бъдат изходни за сигнала, който достига до PIC10F320, след което са прави обръщение към избрания комуникационен протокол

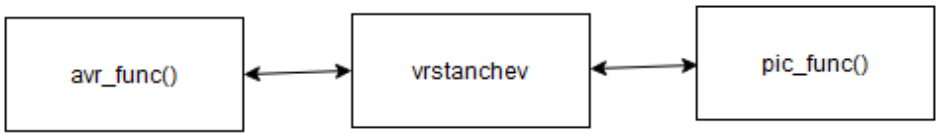
- биват записани стойности в регистри, след като пиновете са направени изходни. Тези стойности след обръщането към протокола биват обработени и изпратени към другия микроконтролер.

Във функцията pic_func спрямо описаното в схемата пиновете RA2 и RA3 са входни. Чрез директивата call се обръщаме към протокола, за да получим данните.

Протокол за комуникация vrstanchev

Протоколът за комуникация между микроконтролерите трябва да бъде подобен на UART и написан на асемблер.

Схема на комуникацията



Използвани пинове относно протокола

Компонент	**Използвани пинове**
ATTiny85	PB0 , PB1
PIC10F320	RA2, RA3
LED1, LED2	RA0, RA1

Сравнение между UART и самостоятелния протокол vrstanchev

Протокол	**UART**	**VRSTANCHEV**
Възможности	Half duplex, full duplex	Half duplex
Реализиран чрез	C	Assembler

Описание на пиновете

Тип на пина I/O	**Пин**
output	PB0
output	PB1
input	RA2
input	RA3

Код на функцията за ATTiny85

```
.include “tn85def.inc”
.equ pb0_out=0b0000001
.equ pb1_out=0b0000010
.cseg
.org 0x00
att_func:
sbi r16, pb0_out
sbi r17, pb1_out
out DDRB, r16
out PORTB,r16
out DDRB, r17
out PORTB,r17
ldi r0, #1
ldi r1, #0
ldi r3, #1
ldi r4, #0
rjmp vrstanchev
```

Код на функцията за PIC10F320

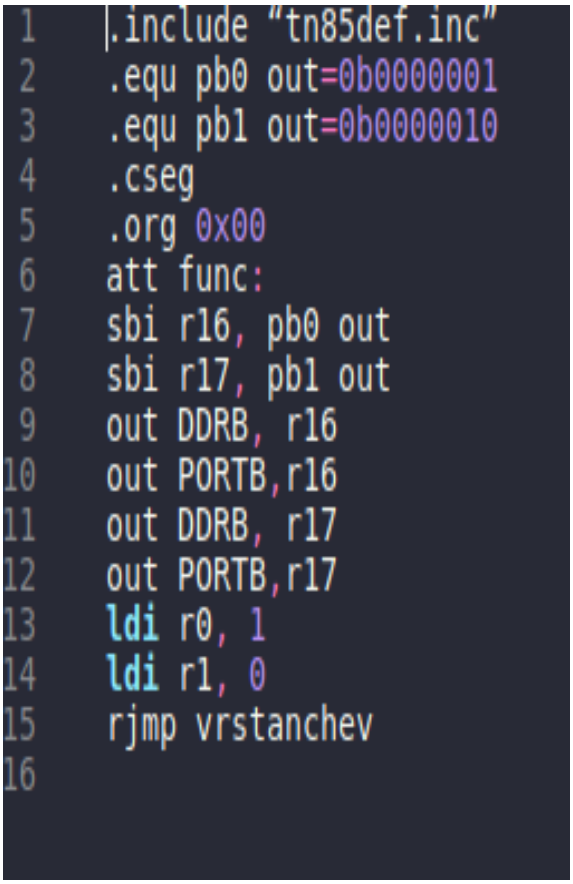
Функцията е реализирана в съответствие със схемата на микроконтролерния модул

```
.include “pic10f320.inc”
.org 0
pic_func:
CLRF PORTA
MOVLW B‘00000100’
MOVLW B‘00001000’
MOVLW B‘00000100’
MOVLW B‘00001000’
call vrstanchev
```

Протокол vrstanchev

```
vrstanchev:
mov B‘00000100’, r0
mov B‘00001000’r1
mov B‘00000100’,r3
mov B‘00001000’ ,r4
```

Скрийншоти на кода



```
1   .include "tn85def.inc"
2   .equ pb0 out=0b0000001
3   .equ pb1 out=0b0000010
4   .cseg
5   .org 0x00
6   att func:
7   sbi r16, pb0 out
8   sbi r17, pb1 out
9   out DDRB, r16
10  out PORTB,r16
11  out DDRB, r17
12  out PORTB,r17
13  ldi r0, 1
14  ldi r1, 0
15  rjmp vrstanchev
16
```

Използвани източници [7]

8. Реализиране на кода като библиотека

Функциите разработени в предишната глава трябва да бъдат реализирани и като софтуерна библиотека с цел преносимост и последващо използване в подобни проекти.

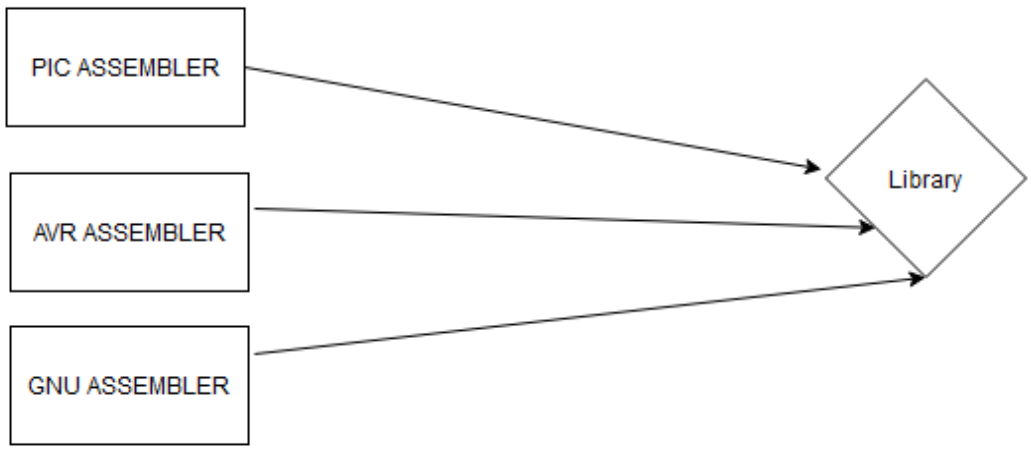
Възможности за реализацията на библиотеката

Библиотеката може да бъде реализирана като архив от обектни файлове чрез:

- GNU ar
- GNU PIC Lib

Може да бъде реализирана от:

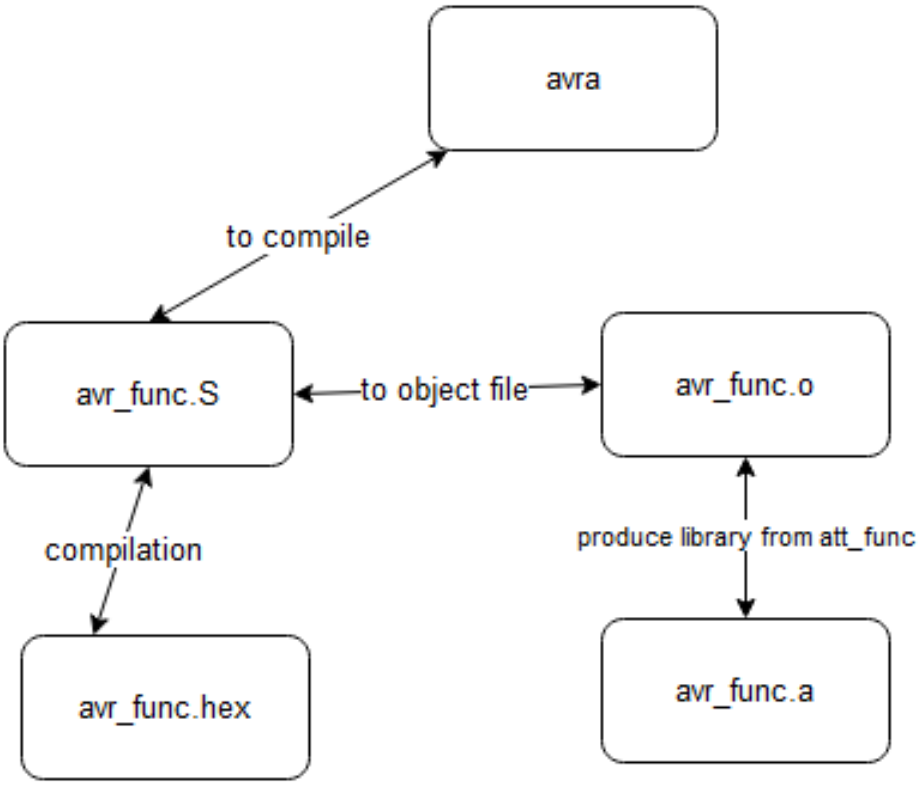
- функции написани чрез GNU Assembler
- функции написани чрез PIC Assembler
- функции написани чрез AVR Assembler
- функции написани чрез AVR Assembler и PIC Assembler и използването на самостоятелен протокол



Избран е последният вариант

Компилиране на функцията att_func() чрез Makefile и avra:

Схема на компилацията:



Фиг. 8.1

Компилиране до .hex файл и до обектен файл:

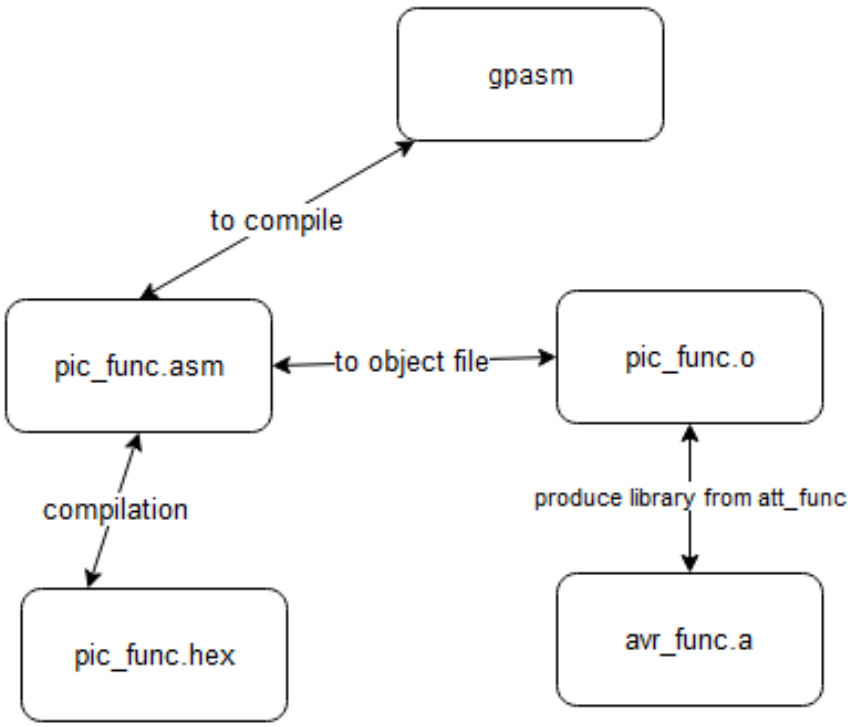
```
compile:att_func.S
        avra att_func.S
att_func.o:att_func.S
           avra -o att_func.o att_func.S
```

Реализиране на функцията att_func() като библиотека

```
libatt:att_func.o
ar rcs libatt_func.a att_func.o
```

Компилиране на функцията pic_func() чрез Makefile и gplib:

Схема на компилацията:



Фиг. 8.2

Компилиране до .hex файл и до обектен файл:

```
compile:pic_func.asm
        gpasm pic_func.asm
pic_func.o:pic_func.asm
           gpasm -c pic_func.asm
```

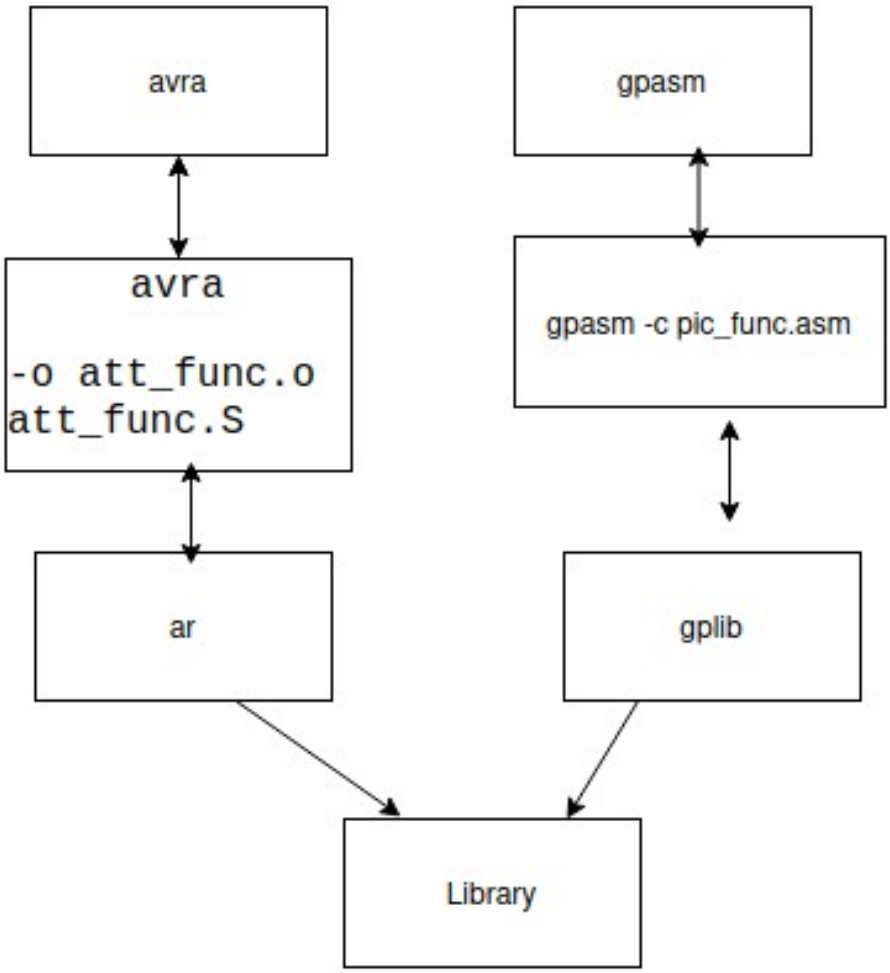
дебъгване на pic_func:

```
pic_func_debug:pic_func.asm
           gpasm -d pic_func.asm
```

Реализиране на функцията att_func() като библиотека

```
piclib:pic_func.o
gplib libpic_func.a pic_func.o
```

Обща схема за библиотеката



Изпълнение на Makefile-овете за тестване на модулите

Компилиране до pic_func.hex файл и до обектен файл:

make compile

Компилиране на att_func до обектен файл

make att_func.o

Компилиране на pic_func до обектен файл

make pic_func.o

Дебъгване на pic_func:

```
make pic_func_debug
```

9. Постигнати резултати. Бъдещо развитие

В настоящата дипломна работа е поставена целта да бъде проектиран микроконтролерен модул, който да осъществява комуникация мужду AVR/PIC базирани микроконтролери. Във въведението са поставени 9 цели. Разгледани са предпоставките при избор на темата на дипломната работа. Всяка глава представя изпълнение на съответна цел.

В първа глава беше направен обзор на съществуващи AVR/PIC микроконтролерни модули. Характеризирани са 4 микроконтролерни модули. Направена е характеристика на предметната област както и класификация на микроконтролерите.

За всеки микроконтролер е направено изследване, което обхваща:

- обща характеристика на микроконтролерният модул
- микроконтролерите, които поддържа
- комуникационните протоколи, които поддържа
- интерфейсите, които поддържа
- асемблери, на които може да бъде програмирани микроконтролерите

Направено е обобщение на получените резултати и заключение за съответната глава.

Във втора глава са поставени изисквания към хардуерната и софтуерна част на проекта.

Към хардуерната част са поставени изисквания за използване на собствен протокол за комуникация между микроконтролерите. Разгледани са различни възможности за реализация на микроконтролерния модул. Описани са изисквания към неговото проектиране. Изискванията са изпълнени в шеста глава.

Към софтуерната част на проекта са избрани асемблери съвместими с микроконтролерите, от които е изграден модулът.

В трета глава е направен анализ на микроконтролерите, които са използвани за проектирането на микроконтролерния модул. Изложени са примери за регистри с общо и специално предназначение, които микроконтролерите поддържат.

В четвърта глава е направен анализ на инструкциите, които микроконтролерите поддържат. Представено е използване на инструкции за двата микроконтролера.

В пета глава е представен сет от инструменти използвани за последващото изпълнение на проекта. Всички използвани инструменти са със свободен лиценз, за да бъде проекта разширен и допълнен.

В шеста глава е разгледано проектирането на микроконтролерния модул. Представени са символи и footprints на микроконтролерите изграждащи модула. Създадена е схема на микроконтролерния модул и разположение на компонентите на модула. Изпълнени са изискванията към хардуерната част на проекта.

В седма глава е реализирана софтуерната част на проекта. Реализирани са функциите за Attiny85 и PIC10F320 – целевите микроконтролери. Реализиран е самостоятелно проектиран протокол за комуникацията между микроконтролерите. Изискванията към софтуерната част на проекта са изпълнени.

В осма глава е представена реализацията на функциите и протокола като софтуерни библиотеки. Разгледани са възможни асемблери за създаване на библиотеката. Използван е сетът от инструменти описан в пета глава.

В настоящата глава са обобщени постигнатите резултати.

Насоченост на проекта

Насочеността на проекта е изцяло практическа.

Проектираният микроконтролен модул може да бъде използван за университетски и учебни цели. Може да бъде използван при:

- провеждане на лабораторни упражнения
- изучаване на GNU Tools
- изучаване на взаимодействие между два асемблера
- програмиране на AVR базирани микроконтролери
- изучаване на протоколи за комуникация

Тъй като за реализацията му са използвани инструменти с отворен код, може функционалността му да бъде надградена чрез добавяне на хедъри към настоящия модул и проектирането на помощни модули към него. По отношение на софтуерната част могат да бъдат проектирани библиотеки за различните видове ISA.

Може да бъде от полза при изучаването на микроконтролери и използването на свободни асемблери.

Възможности за бъдещо развитие на проекта

1. Разширяване на възможностите на микроконтролерния модул - хардуерна част:
    - добавяне на допълнителни интерфейси към модула
    - разработване и използване на допълнителни помощни модули към основния
    - реализиране на различни версии на модула
    - реализиране на модул с микроконтролери

      AVR-AVR

      PIC-PIC

      референция: **Възможности за реализация на микроконтролерният модул**
    - реализация на модул с друг сет от инструкции (ISA)

2. Разширяване на възможностите на микроконтролерния модул - софтуерна част:
    - адаптиране на библиотеката и реализираните функции за друга архитектура (ISA)
    - разширяване на възможностите на комуникационния протокол vrstanchev

Съществуващи научни статии по темата на дипломната работа:

Digitally controlled 4-channel constant current source for LED luminary based on PIC microcontroler

Programming of AVR Microcontrollers-Ansgar Meroth

Board design and implementation for 8-bit avr microcontrollers in an educational environment-Gideon van Roon

Заключение

В рамките на сегашната дипломна работа беше разгледана целта:

Да бъде проектиран микроконтролерен модул с използване на AVR-базиран микроконтролер и PIC-базиран микроконтролер с цел комуникация между двата микроконтролера на асемблерно ниво.

За изпълнението на тази цел бяха поставени следните задачи:

- **Обзор на съществуващи микроконтролерни модули с AVR/PIC базирани микроконтролери**
- **Изисквания към проекта**
- **Анализ на микроконтролерите, които ще бъдат използвани за микроконтролерния модул**
- **Анализ на асемблерните инструкции, които ще бъдат използвани**
- **Софтуерни инструменти за проектиране на микроконтролерният модул и програмиране**
- **Проектиране на микроконтролерния модул**
- **Програмиране на микроконтролерите**
- **Реализиране на кода като библиотека**
- **Постигнати резултати. Бъдещо развитие**

Изискванията към библиотеката бяха изпълнени.

Поставените цели бяха успешно изпълнени.

Целта на дипломната работа е постигната.

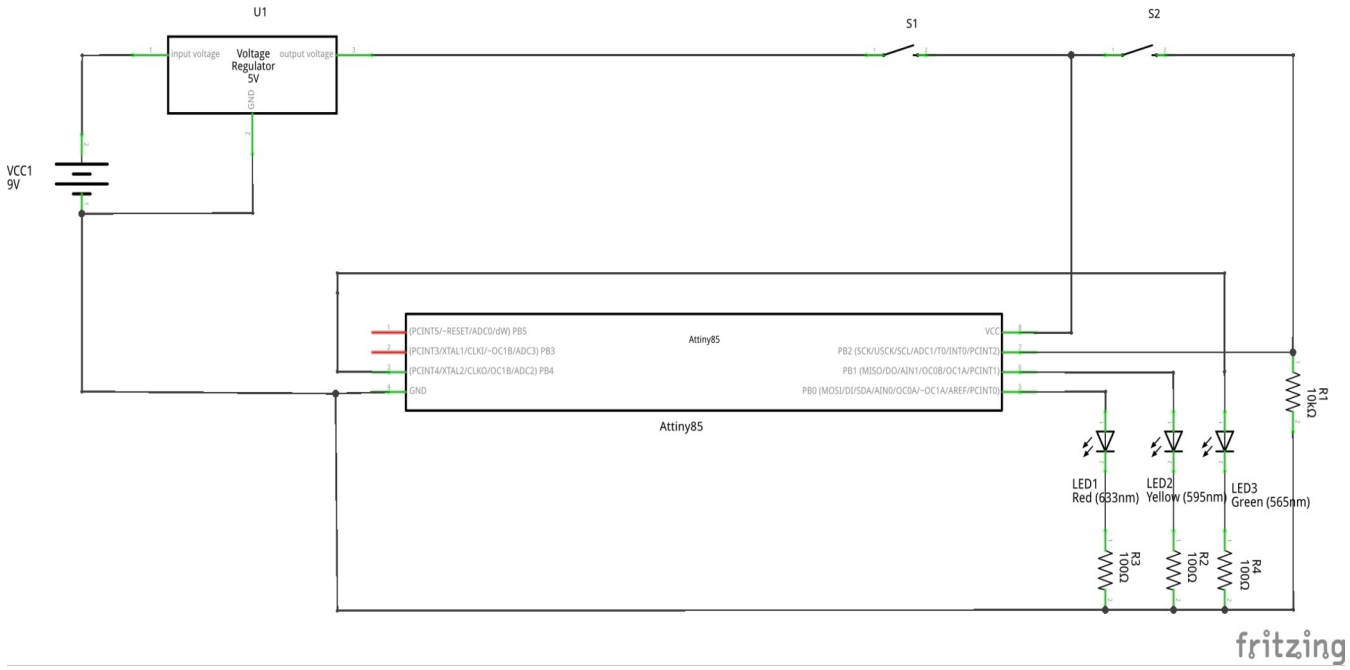
Дипломната работа ще бъде допълнително развита, чрез използване на някоя от посочените възможности.

Приложение 1:

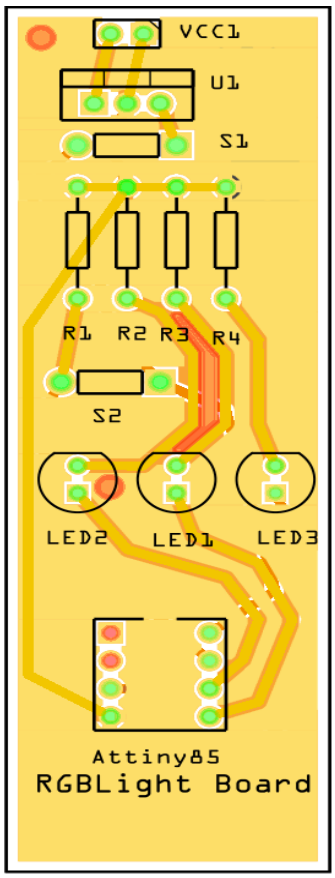
Проектирани AVR/PIC микроконтролерни модули използвайки свободно -достъпни схеми и програма за проектиране Fritzing

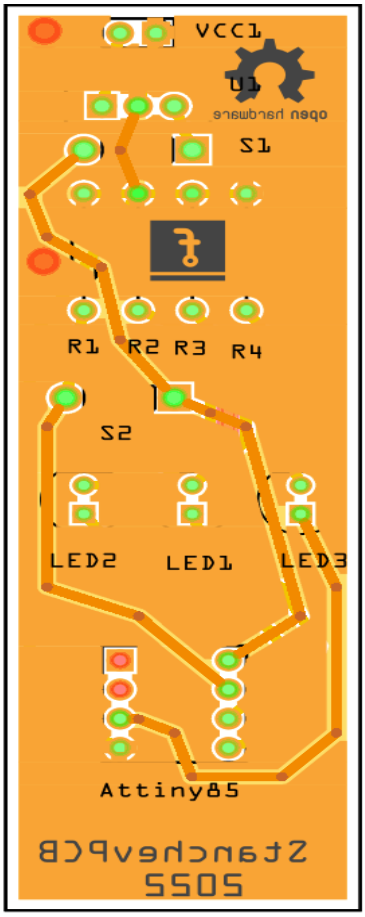
1. Attiny 85 Traffic Lights

- **Схема Attiny 85 Traffic Lights**

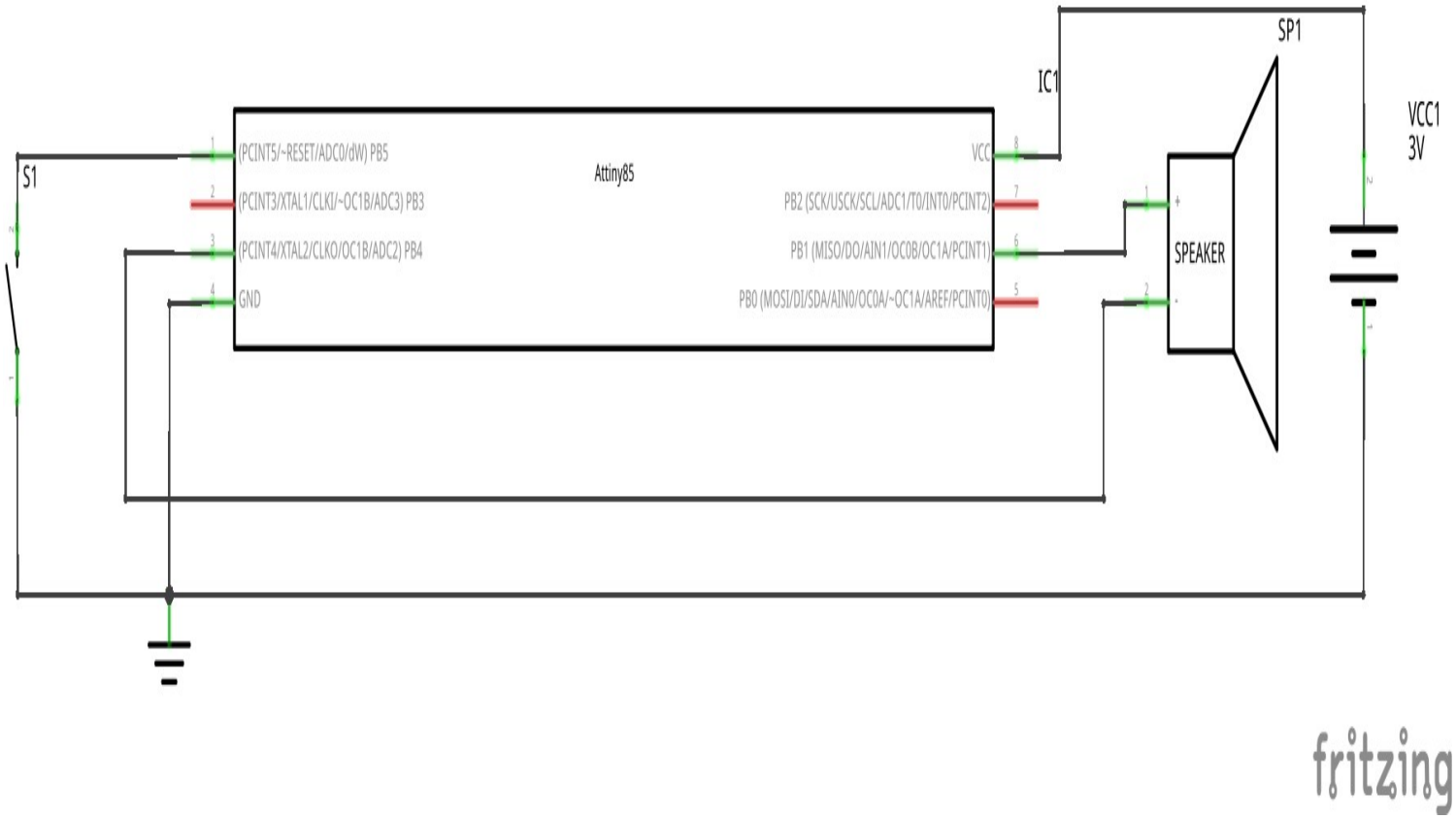


- **Преглед на PCB Attiny 85 Traffic Lights**

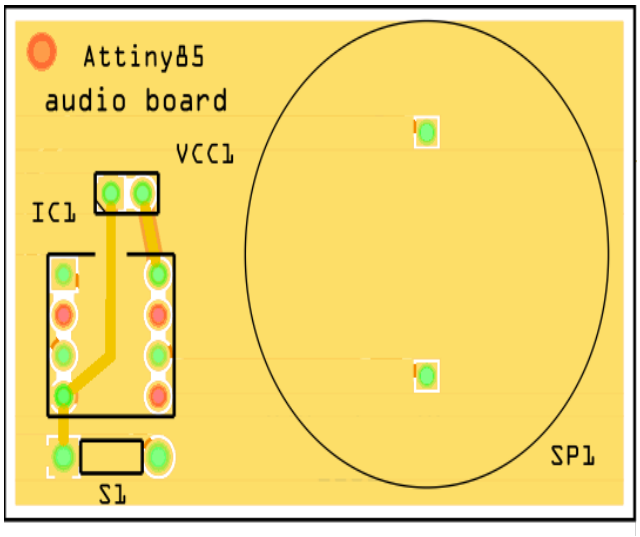


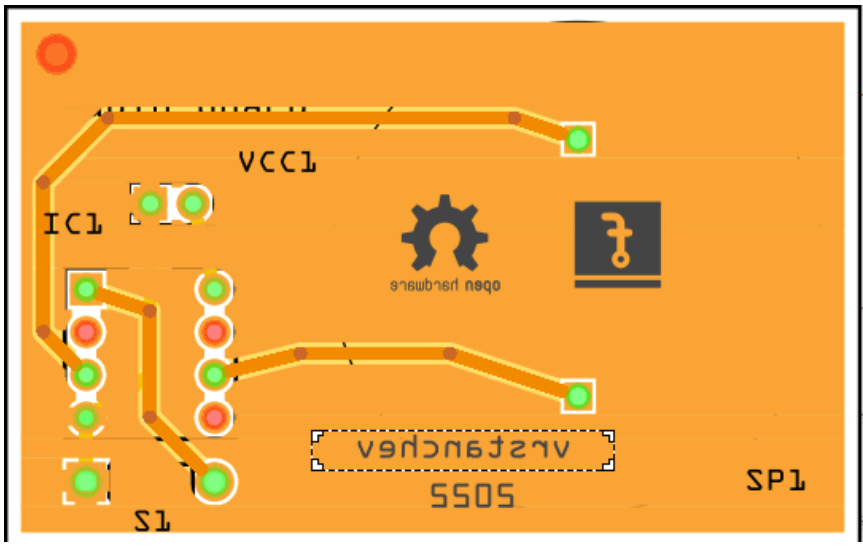


2. **Attiny 85 Audio Player**
   - **Схема Attiny 85 Audio Player**



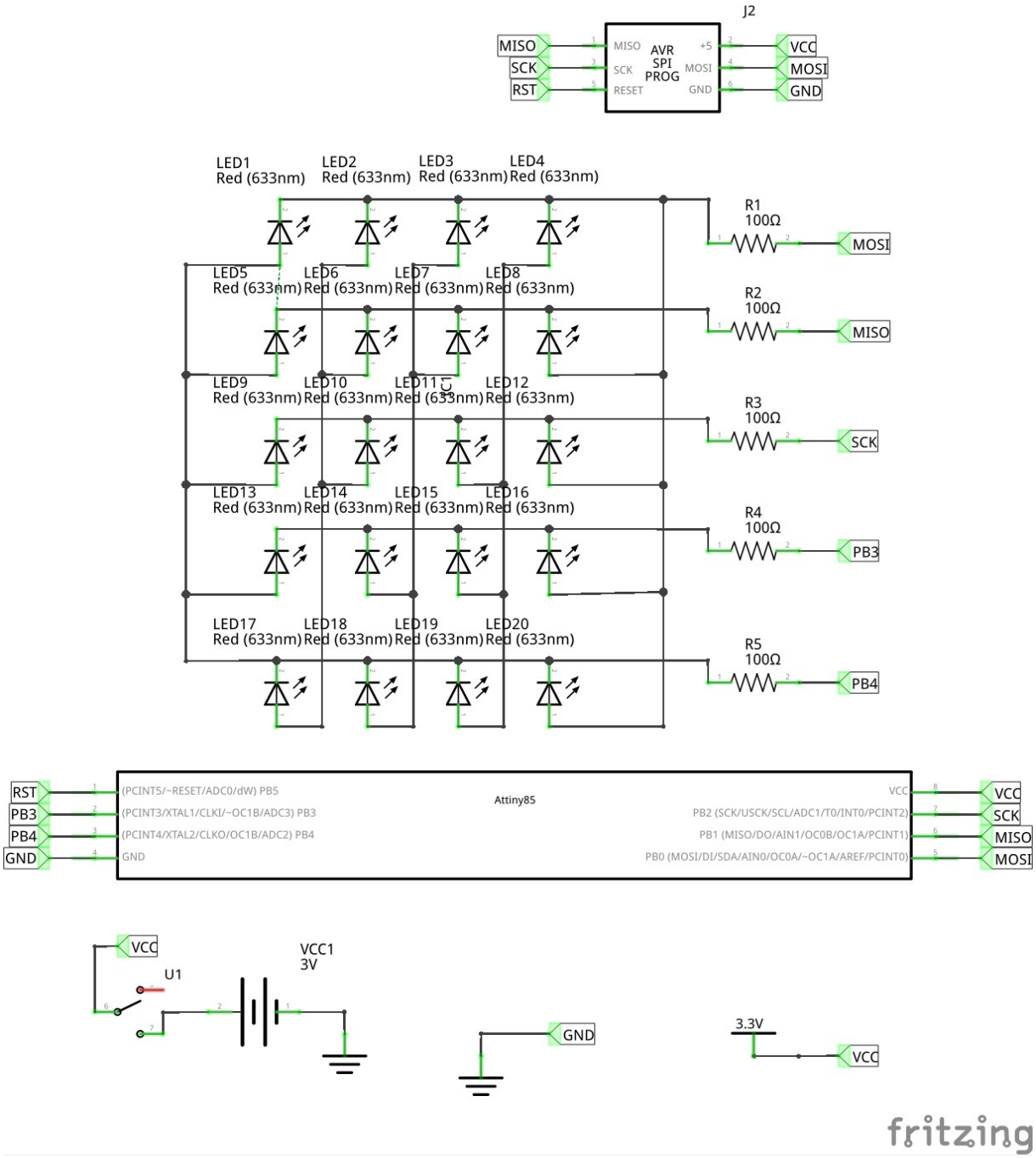
   - **Преглед на PCB Attiny 85 Audio Player**





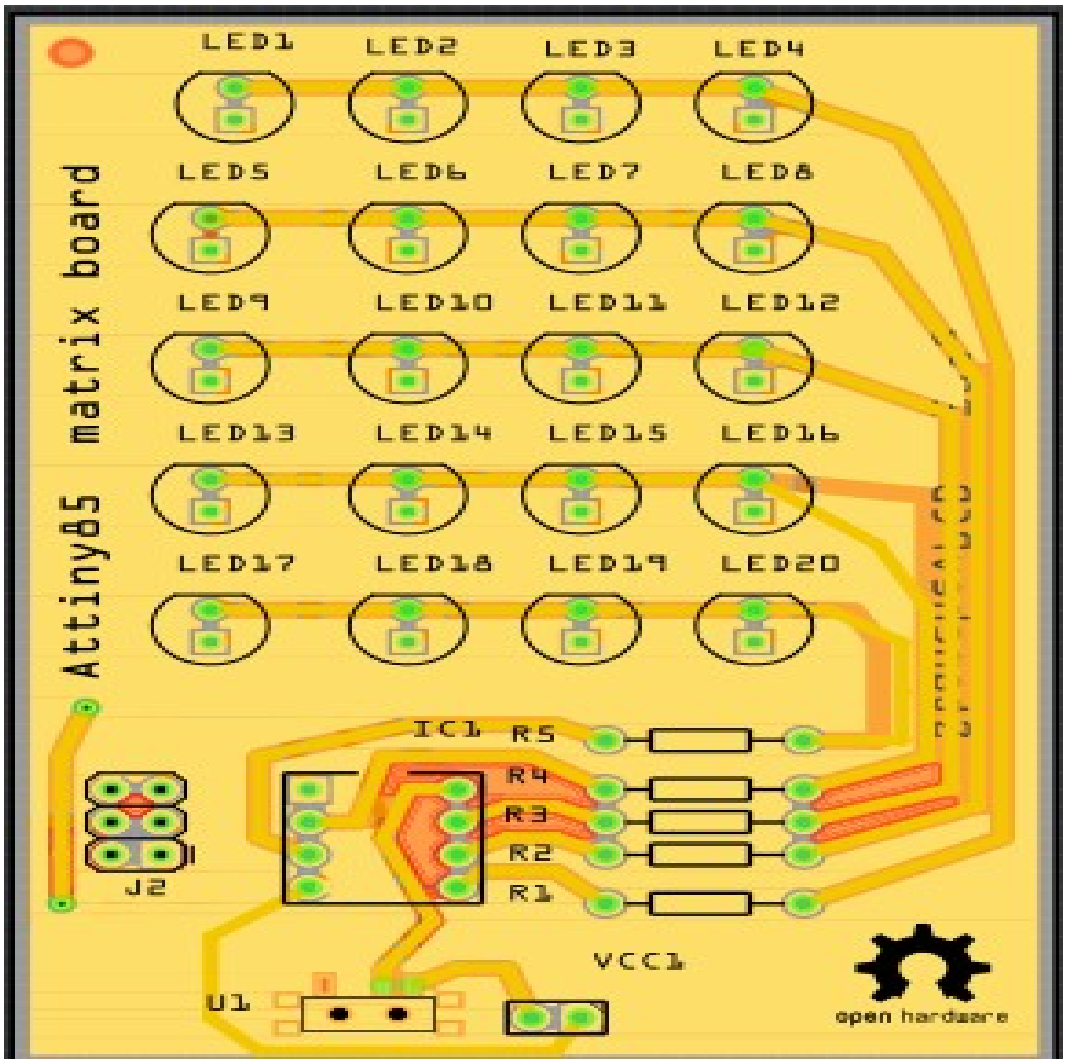
3. Attiny 85 LED matrix

- **Схема Attiny 85 LED matrix**



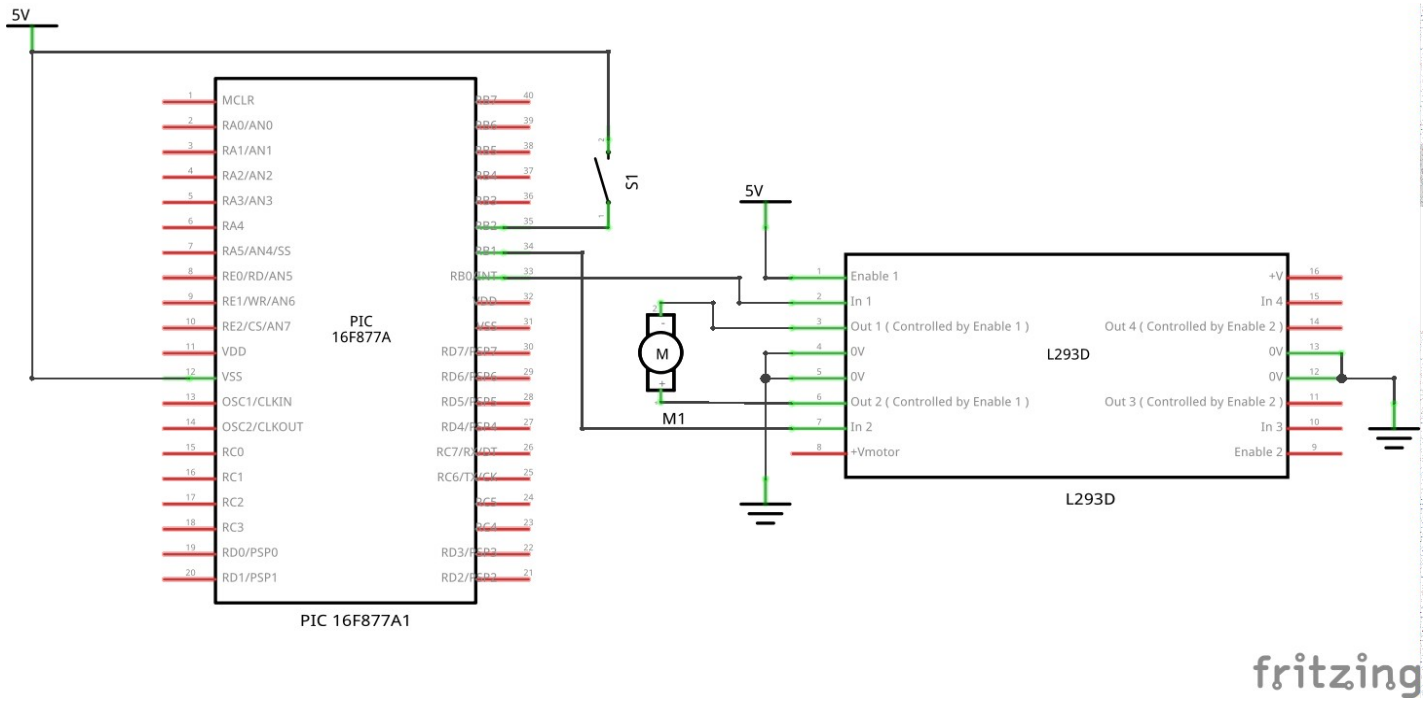
Преглед на PCB Attiny 85

LED matrix



3. pic16f877a Motor

- Схема pic16f877a Motor



[1]={[[1.1]],[[1.2]],[[1.3]],[[1.4]]}

[[1.1]] ANAVI Macro Pad 2 DataSheet
[[1.2]] ANAVI Light Controller DataSheet
[[1.3]] avr64dd32 DataSheet
[[1.4]] rp2040 board DataSheet

[[2]]ATTiny85 Datasheet

[[3]] PIC10F320 Datasheet

[[4]]Linux Man Pages

[[5]] LibrePCB-Documentation

[[6]] Проектирани AVR/PIC микроконтролерни модули