

**ТЕХНОЛОГИЧНО УЧИЛИЩЕ “ЕЛЕКТРОННИ СИСТЕМИ”
към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

ДИПЛОМНА РАБОТА

Тема: Автономен пътен знак с възможност за дистанционно управление
през GPRS мрежа

Дипломант:

Илиян Антов

Научен ръководител:

инж. Михаил Ангелов

СОФИЯ
2020

Zadanie here

Otziv here

Използвани съкращения

APN - Access Point Name

CAD - Computer-Aided Design

EMI - Electromagnetic Interference

FLA - Flooded Lead-Acid

GPRS - General Packet Radio Service

GSM - Global System for Mobile Communications

GUI - Graphical User Interface

IDE - Integrated Development Environment

IMEI - International Mobile Equipment Identity

IMSI - International Mobile Subscriber Identity

IP - Internet Protocol

LCD - Liquid Crystal Display

LED - Light-Emitting Diode

LFP - Lithium Iron Phosphate (LiFePO₄)

MMS - Multimedia Messaging Service

MPP - Maximum Power Point

MPPT - Maximum Power Point Tracking

OLED - Organic Light-Emitting Diode

PCB - Printed Circuit Board

PSTN - Public Switched Telephone Network

RGB - Red-Green-Blue

SIM - Subscriber Identity Module

SIPO - Serial-In Parallel-Out

SLA - Sealed Lead-Acid

SMD - Surface-Mount Device

SMS - Short Message Service

SSL - Secure Sockets Layer

TLS - Transport Layer Security

UART - Universal Asynchronous Receiver-Transmitter

WAP - Wireless Application Protocol

ДКМ - Двоично-Кодова Модулация

КПД - Коефициент на Полезно Действие

МД-РКВ - Множествен Достъп с Разделяне на Каналите по Време

МД-РКК - Множествен Достъп с Разделяне на Каналите по Код

МД-ЧРК - Множествен Достъп с Честотно Разделяне на Каналите

ШИМ - Широчинно-Импулсна Модулация

УВОД

Модернизацията на транспорта е една от най-актуалните теми в съвременните информационни технологии. Големи компании започват да отделят все повече ресурси за автоматизацията на своите автомобили, като крайната им цел е пълна автономност. Автомобилите стават все по-интелигентни - в последните години се наблюдава засилено внимание към функции, подпомагащи водача при управлението. За тази цел автомобилите се оборудват с голям брой сензори както отвътре, така и отвън. С помощта на тези сензори, автомобилите могат да разпознават състоянието на водача, разстоянията до другите автомобили, опасностите на пътя, пътните знаци, лентите за движение и т.н., асистират водача и подобряват безопасността на пътя.

Но въпреки стремителното развитие в автомобилната електроника, една друга, директно свързана с нея област често остава пренебрегната - пътната инфраструктура. Една от основните задачи на управляващия софтуер на автономен автомобил е да разпознава с голяма степен на надеждност обозначенията на пътя - пътна маркировка, знаци и светофари. Тези средства за регулиране на трафика обаче не претърпяват никакво развитие и сякаш остават забравени от иначе бързо развиващите се технологии.

Пътните знаци са типичен пример за това - те не са претърпели никакви особени промени от своето създаване до днес. Един ограничителен знак например е само това - той не може да промени указанието, изписано върху него. Но пътната обстановка не е нещо статично - тя се променя постоянно и едно единствено указание често е недостатъчно, за да обозначи най-оптималното и безопасно ограничение за всички възможни състояния на пътя. Най-разпространеното решение на този проблем е поставянето на няколко пътни знака, всеки от които има условие за валидност - било то наличие на дъжд, сняг или мъгла. Това обаче рядко е оптимално, тъй като е свързано с повишено внимание от страна на водача и може лесно да доведе до объркване.

Друг пример за проблем, причинен от статичността на знаците, е липсата на метод за предизвестяване на водачите при настъпване на някакъв вид инцидент на пътя, по който се движат. Пътнотранспортните произшествия могат да се окажат изключително опасни не само за директно потърпевшите, но и за останалите участници в движението. В днешни дни предизвестяването може да се случи едва след пристигането на съответните държавни органи, които да вземат нужните мерки - било то отклоняване на движението или регулиране на преминаващите автомобили.

Настоящата дипломна работа има за цел решаването на тези проблеми чрез създаване на автономен пътен знак, чиито указания могат да бъдат променяни дистанционно. За да се реши проблемът с нуждата от няколко пътни знаци при различни условия, визуализацията на валидното ограничение или съобщение ще става чрез светодиодна матрица. За да се избегне закъснението на реакция при настъпване на произшествие, знакът ще може да бъде управляван дистанционно, като се осигури неговата постоянна свързаност към централен сървър, от където ще получава съответната информация. Свързаността ще бъде постигната чрез GPRS технологията, предоставяща широк обхват на покритието в цялата страна. За да се осигури постоянната и надеждна работа на устройството, ще му бъде осигурено захранване чрез фотоволтаичен панел и батерия. Това ще позволи поставянето на знака дори в местности, където изградена захранваща инфраструктура липсва или е твърде ненадеждна.

Първа глава

Технологии за реализация на микроконтролерна система за дистанционно управление на светодиоден пътен знак

1.1. Средства за визуализация на информация

За целите на микроконтролерната система е важен правилният избор на най-подходящата технология за визуализация на информация за съответното приложение. В т. 1.1.1. са разгледани най-разпространените видове дисплеи и са посочени техните основни предимства и недостатъци.

1.1.1. Видове дисплеи

Източниците на информация са дадени в [1] и [2]

1.1.1.1. Течнокристален дисплей (Liquid Crystal Display - LCD)

Работата му се основава на тънък слой течни кристали, които променят оптичните си качества при подаване на напрежение. Подобен дисплей е показан във фиг. 1.1.

- *Предимства:*
 - Ниска цена
 - Ниска консумация
 - Дълъг живот (>60 000 часа)
- *Недостатъци:*
 - Нужда от външен източник на светлина
 - Подсветката добавя към цената и намалява ефективния живот
 - Невъзможност за работа при ниски температури (<0°C) - този недостатък прави такъв вид дисплеи изключително неподходящи за нуждите на устройството, което ще бъде поставено на открито и трябва да запази функционалността си в широк температурен диапазон (от ~ -30°C до ~ +40°C)



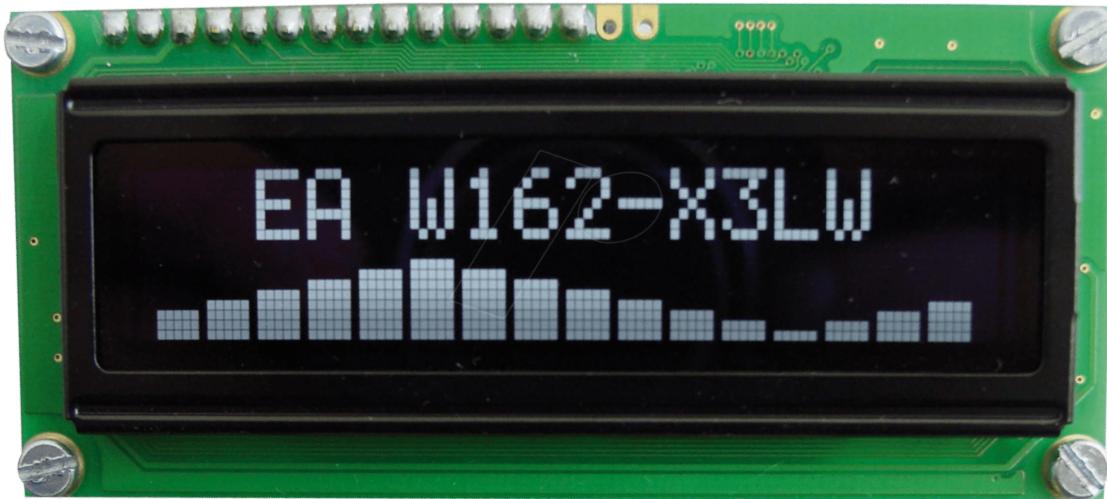
Фиг. 1.1. Течнокристален дисплей

1.1.1.2. Дисплей с органични светодиоди (Organic Light-Emitting Diode - OLED)

Визуализацията става чрез т.нар. органични светодиоди, всеки от които сам излъчва светлина, елиминирайки нуждата от подсветка. Такъв дисплей е изображен във фиг. 1.2.

- *Предимства:*
 - Точни и ярки цветове
 - Широк ъгъл на видимост
 - Могат да бъдат огъвани
- *Недостатъци:*
 - По-кратък живот от други видове дисплеи
 - Висока цена
 - Могат да бъдат повредени от влага
 - При продължително изобразяване на една и съща картина се наблюдава т.нар. прогаряне на екрана - ефект, при който се получава обезцветяване на екрана на определени места поради неравномерното използване на панела. При промяна на картината върху дисплея остава "призрачен образ" на предходното изображение.

Непоносимостта към влага и прогарянето на екрана правят този тип дисплеи неподходящи за целите на микроконтролерната система.

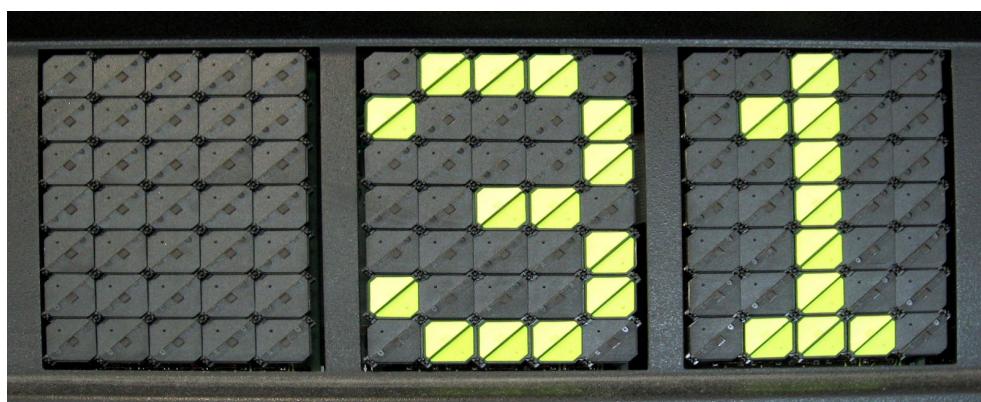


Фиг. 1.2. Дисплей с органични светодиоди

1.1.1.3. Дисплей с флип дискове (Flip-Disc Display)

Пикселите представляват малки дискове, които се обръщат при подаване на електрически импулс. Всеки диск е оцветен в матово черно от едната страна и флуоресцентно или светлоотразително от другата. Подобен вид дисплей е показан във фиг. 1.3.

- *Предимства:*
 - Ниска консумация (само при промяна на картината)
 - Много широк ъгъл на видимост
 - Теоретично дълъг живот (>100 000 часа)
- *Недостатъци:*
 - Голям брой движещи се части, които могат да се повредят
 - Изиска по-честа поддръжка от други видове технологии
 - Нужда от външно осветяване - неподходящи за използване през нощта, което е и основната причина този тип дисплеи да не са най-доброя избор за проектираното устройство.



Фиг. 1.3. Дисплей с флип дискове

1.1.1.4. Светодиоден дисплей (Light Emitting Diode Display)

Състои се от множество светодиоди - електронни елементи, които излъчват светлина в тесен спектър, когато през тях преминава електрически ток. Такъв дисплей е изображен във фиг. 1.4.

- *Предимства:*
 - Възможност за бързо превключване
 - Дълъг живот при нормални условия (>50 000 часа)
 - Висока яркост
 - Висока ефективност
 - Възможност за работа в широк температурен обхват
- *Недостатъци:*
 - Ниска резолюция
 - Сравнително висока цена
 - Цветът и ефективността варираят в зависимост от температурата



Фиг. 1.4. Светодиоден дисплей

Нуждите на проектираната микроконтролерна система предполагат дисплей, който:

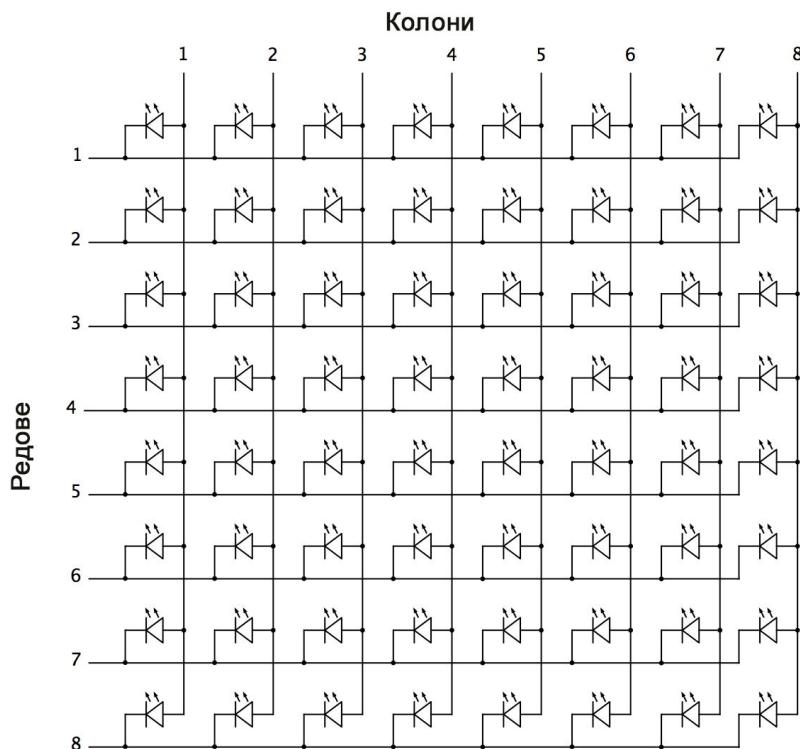
- Да бъде достатъчно устойчив на различни температурни условия;
- Да бъде с ниска консумация, за да се минимизират разходите за захранване;
- Да има достатъчно висока яркост, позволяваща му да бъде лесно различим дори при директна слънчева светлина.

Имайки предвид тези изисквания, както и предимствата и недостатъците на всеки от разгледаните видове дисплеи, беше преценено, че най-подходящ за нуждите на устройството е светодиодният дисплей.

1.1.2. Управление на матрици с голям брой светодиоди

Източникът на информация е даден в [3]. Съществуват различни методи за управление на светодиодни матрици:

- **Директно управление** - при него всеки светодиод се свързва към отделен извод на управляващото устройство (най-често микроконтролер). Най-простият за имплементация метод, но и най-неприложимият в реални условия поради непостижимо големия брой изводи, който се изисква при големи матрици;
- **Паралелно управление** - при този метод матрицата се разделя на редове и колони (фиг. 1.5.). Управляващото устройство едновременно контролира всеки от редовете и всяка от колоните, като по този начин с $2N$ извода могат да се управляват N^2 светодиода. На всяка уникална входна комбинация, която може да бъде подадена на изводите, съответства едно единствено състояние на матрицата. Предимство на този вид управление е бързодействието - опресняването на цялата матрица отнема един единствен такт на управляващото устройство. Този метод, макар и по-мащабируем от метода с директно управление, също страда от нарастващ брой изводи при увеличение на размерите на матрицата. За управление на трицветна матрица с широчина и височина 32×32 пиксела например, ще бъде нужно микроконтролерно устройство с общо 192 извода (формула (1.1.)). За това подобни устройства са скъпи и неподходящи за такова приложение;



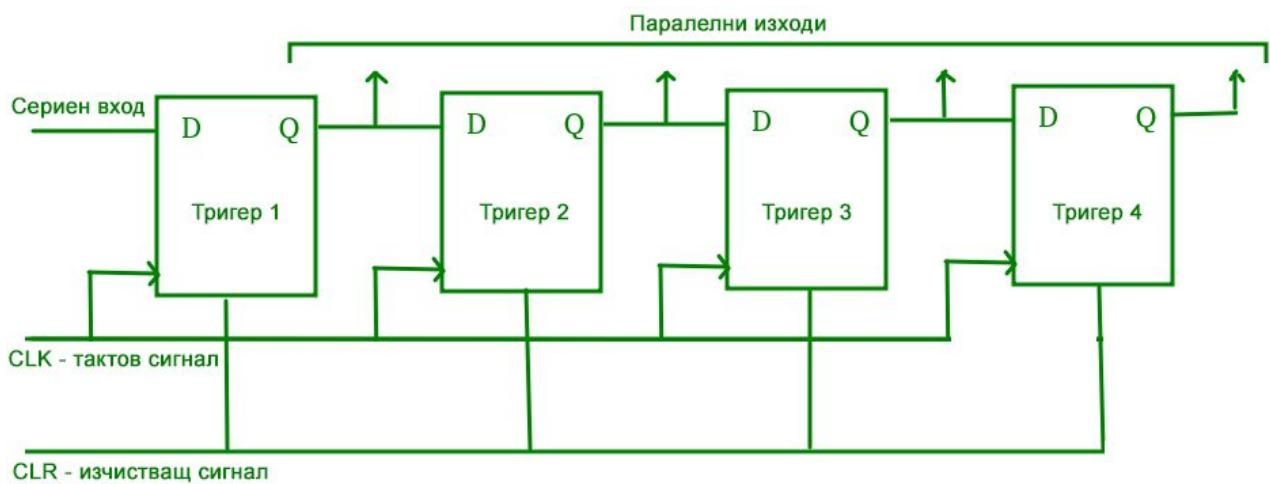
Фиг. 1.5. Паралелно свързване на светодиоди

$$\text{Брой изводи} = \text{Цветове} * (\text{Колони} + \text{Редове})$$

(1.1.)

$$\text{Брой изводи} = 3 * (32 + 32) = 192$$

- **Управление с преместващи регистри** - този метод се основава на използването на схеми с памет, наречени преместващи (shift) регистри. Преместващият регистър представлява група от последователно свързани тригери, при които при постъпване на тактов импулс информацията се предава към следващия тригер. Използваните преместващи регистри са от тип SIPO (Serial-In Parallel-Out) - те са снабдени с един сериен вход и няколко паралелни изхода. Обобщена схема на такъв регистър е дадена във фиг. 1.6. Тази конфигурация позволява поетапното въвеждане на информация в регистрите в продължение на няколко такта на управляващото устройство. По този начин теоретично е възможно управлението на неограничен брой изходи само с 2 входа. Недостатък на този метод е намаляването на бързодействието с увеличаването на броя изходи. Всеки допълнителен изход се равнява на един допълнителен тактов импулс.

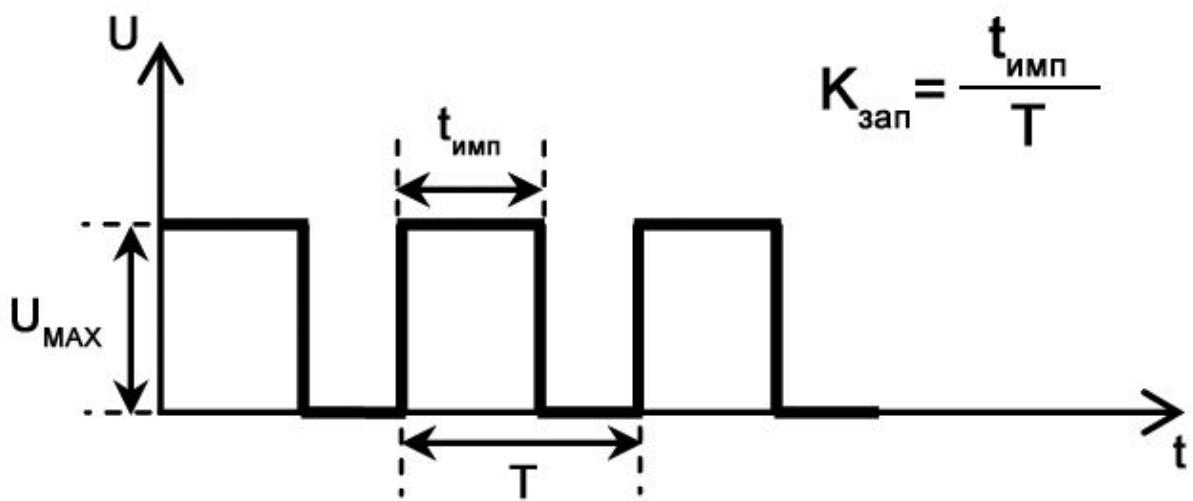


Фиг. 1.6. Обобщена схема на SIPO преместващ регистър

Въпреки по-бавната работа в сравнение с паралелното управление, този метод се предпочита при управление на матрици с голям брой светодиоди.

1.1.3. Постигане на различни цветове в големи светодиодни матрици

Източниците на информация са дадени в [4] - [6]. Най-разпространеният метод за управление на яркостта на светодиоди е т.нр. широчинно-импулсна модулация (ШИМ). ШИМ представлява управление на широчината на импулса в рамките на даден фиксиран период. Отношението между широчината на импулса и периода се нарича коефициент на запълване. Изчисляването му е показано във фиг. 1.7.



Фиг. 1.7. Изчисление на коефициента на запълване на ШИМ

Средното напрежение на изхода на ШИМ се изчислява по формула (1.2.).

$$U_{\text{ср}} = K_{\text{зап}} * U_{\text{max}} \quad (1.2.)$$

$U_{\text{ср}}$ - Средна стойност на напрежението на изхода на ШИМ

$K_{\text{зап}}$ - Коефициент на запълване на ШИМ

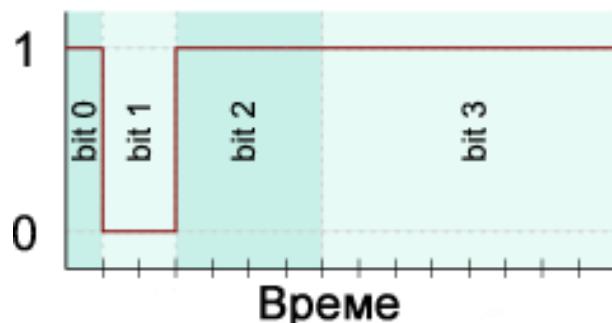
U_{max} - Максимално напрежение на сигнала на ШИМ

При достатъчно кратък период (съответно достатъчно висока честота), човешкото око възприема управляван от ШИМ светодиод като постоянно светещ, но с различна яркост според средното количество енергия, достигната до окото. С други думи, колкото по-висок е коефициентът на запълване, толкова по-ярък е светодиодът.

Съществуват 2 начина за постигане на ШИМ:

- **Хардуерен ШИМ** - използва хардуерни ШИМ генератори. Изключително бърз и надежден метод, но ограничен откъм броя налични канали.
- **Софтуерен ШИМ** - постига се чрез управляващ софтуер. Този метод не е толкова надежден, колкото хардуерния ШИМ, но броят канали е неограничен. Недостатъкът му е че процесорното време се увеличава с увеличаване на каналите.

Алтернативен метод за управление на яркостта на светодиоди е т.нар. двоично-кодова модулация (ДКМ). ДКМ използва ключово свойство на двоичните числа - всеки следващ бит (от най-младшия към най-старшия) е с двойно по-голяма тежест от предходния. При ДКМ с резолюция 4 бита например, времето може да се раздели на 15 времеви отрязъка. Най-младшият (първият) бит се равнява на 1 отрязък, вторият на 2, третият на 4 и четвъртият (най-старшият) на 8. При комбинирането на битовете са възможни общо 16 комбинации, които се равняват на 16 различни коефициента на запълване. Ако на даден светодиод се отреди двоичното число 1101 (13) например, то той ще има коефициент на запълване $13/15 = 0.87$. (фиг. 1.8.)



Фиг. 1.8. ДКМ с коефициент на запълване 13/15

Поради фиксираните интервали на промяна на състоянието на светодиодите, увеличаването на техния брой не е свързано с увеличаване на процесорното време. Тази характеристика на ДКМ я прави идеалният избор при управление на голям брой светодиоди с минимален процесорен ресурс, както в случая.

Без използване на модулация, RGB светодиодна матрица може да постигне 8 цвята ($3 \text{ бита} \Rightarrow 2^3 = 8$). Използването на ДКМ позволява постигането на много по-голям брой цветове, като броят им зависи от избраната резолюция.

1.2. Клетъчни технологии и протоколи

Съществуват множество различни методи за постигане на безжична свързаност. За целите на проектираната система е важно да се избере подходяща технология, съобразена с изискванията. От първостепенна важност за устройството е възможността му да запази работоспособността си при поставянето му на различни места из цялата страна. Поради тази причина, основно изискване при избора на безжична технология е тя да има възможно най-голям обхват. Скоростта на предаване не е от голямо значение, тъй като количеството изпращана и приемана от устройството информация е много малко. Двете съществуващи технологии с най-голямо покритие към днешна дата са клетъчната и сателитната. Поради сложността и високата цена на сателитната свързаност, за устройството е избрана клетъчната технология GPRS (General Packet Radio Service). Особеностите на клетъчната свързаност са описани в т. 1.2.1. Описание и сравнение на най-разпространените клетъчни технологии е направено в т. 1.2.2. GPRS технологията е описана в т. 1.2.3.

1.2.1. Особености на клетъчната свързаност

Източниците на информация са дадени в [7] и [8]. Клетъчната мрежа представлява комуникационна мрежа, при която крайната връзка е безжична. Осигуряването на услуги към крайните потребители става чрез разполагане на голям брой базови станции с лимитирана мощност. Земната площ условно се разделя на т. нар. "клетки", всяка от които се обслужва от една или повече базови станции. Лимитирането на мощността на излъчване на станциите прави възможно преизползването на едни и същи честоти през няколко клетки, без това да причинява смущения (интерференция), както е показано във фиг. 1.9.



Фиг. 1.9. Преизползване на честоти в отдалечени клетки.

Използването на този метод позволява осигуряването на покритие над неограничено голяма географска площ само с ограничен набор от честоти.

Когато даден потребител се премести от една клетка в друга, обменът на данни трябва да се прехвърли към нова базова станция. Това става чрез процес, известен като "предаване". Предаването представлява избор на канал за устройството от страна на новата базова станция и превключване към новия канал в потребителското устройство. В днешните клетъчни мрежи този процес е автоматичен и не причинява смущения в обмена на данни.

Най-разпространената клетъчна мрежа към дневна дата е мобилната телефонна мрежа. Тя използва радиовълни, за да обменя информация с крайни устройства, наречени мобилни (или клетъчни) телефони. Модерните мобилни мрежи се възползват от клетъчната технология, тъй като радиочестотите са споделен и ограничен ресурс, което прави преизползването им наложително.

За осигуряването на покритие се грижат големи компании, наречени мобилни оператори. Използването на мобилната мрежа на всеки оператор е свързано със заплащане на определени тарифи към него. В повечето случаи тези тарифи са под формата на месечен абонамент. При липса на договор, таксуването става на базата на време или количество обменена информация. Примерни такси за използването на клетъчна телефонна мрежа са дадени във фиг. 1.10.

Цени за разговори, SMS, MMS и мобилен интернет със стандартен предплатен план на Теленор

| | Цени |
|---|----------------------|
| Разговори към всички национални мрежи | 0,45 лв./мин. |
| Разговори към 5 номера от група За приятели | 0,29 лв./мин. |
| SMS към национални мрежи | 0,29 лв./ SMS |
| MMS в мрежата на Теленор | 0,69 лв./ MMS |
| Мобилен интернет в България | 2,40 лв./ MB |

Фиг. 1.10. Тарифи за използването на мрежата на мобилния оператор "Теленор"

За идентификацията на всяко устройство, свързано към мобилната мрежа, се грижи т.нар. SIM (subscriber identity module) карта. Тя представлява интегрална схема, в която е записан IMSI (international mobile subscriber identity) номерът и ключът, които се използват за идентификация и удостоверяване на мобилните абонати. В SIM картата се записват също контактите на абоната. Такава карта е показана във фиг. 1.11.



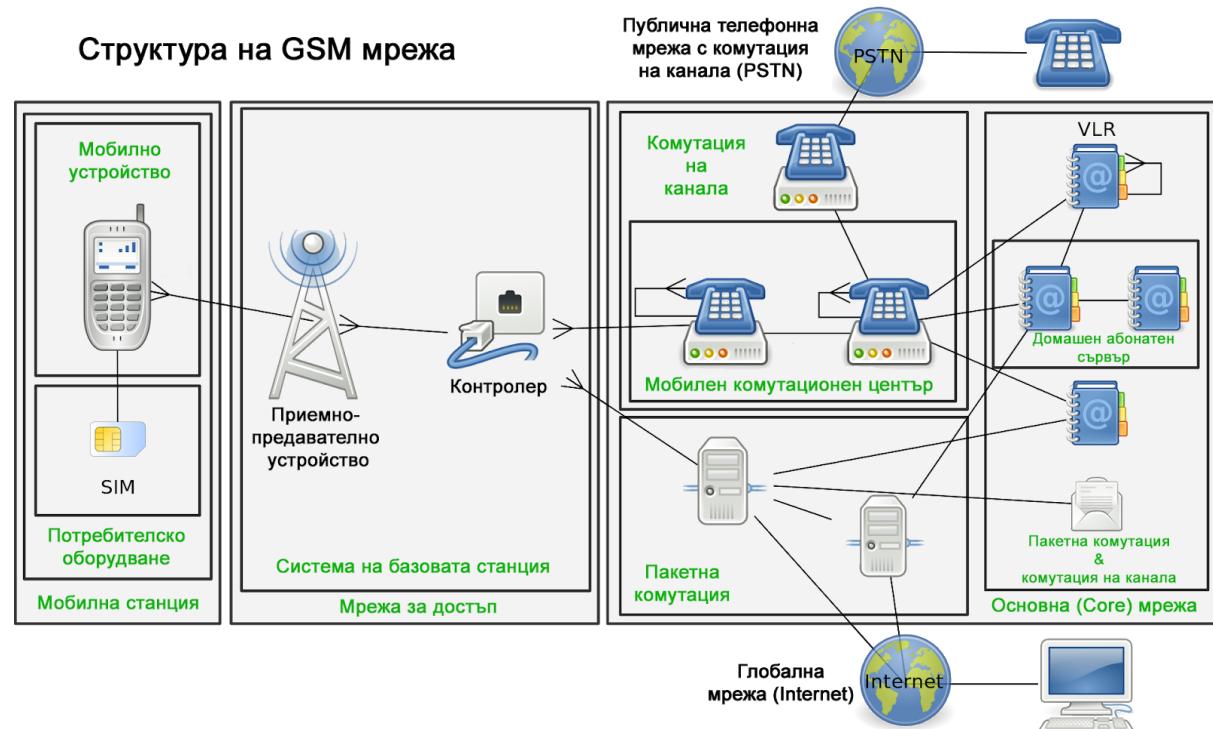
Фиг. 1.11. SIM карта

За да се разграничават сигнали от различни предаватели са разработени няколко различни методи за множествен достъп в клетъчните мрежи:

- **Множествен достъп с честотно разделяне на каналите (МД-ЧРК)**
 - всеки отделен канал се характеризира с различна честота. Този метод е стар и се използва рядко поради лимитирания брой канали, които могат да бъдат създадени;

- **Множествен достъп с разделяне на каналите по време (МД-РКВ)**
 - всеки канал има отреден времеви отрязък, в който само и единствено той може да предава и приема. Това налага въвеждането на изкуствено закъснение на сигнала. На този метод се базират най-ранните мобилни технологии с пакетна комутация, които позволяват предаването на IP (Internet protocol) пакети в клетъчната мрежа. Този метод се използва в GSM (Global System for Mobile Communications) мрежите;
- **Множествен достъп с разделяне на каналите по код (МД-РКК)** - всеки канал използва различен ключ за кодиране на информацията, което позволява едновременното използване на средата от множество канали. Този метод е най-новият от изброените и се използва в множество модерни мобилни технологии.

С развитието на клетъчната технология, скоростите на предаване на информация стават все по-бързи. Това позволява използването на мобилните мрежи както за предаване на гласови пакети, така и за връзка с глобалната мрежа чрез предаване на интернет пакети. Опростена диаграмма на GSM клетъчна мрежа с възможност за пренасяне на гласови и интернет пакети е показана във фиг. 1.12.



Фиг. 1.12. Структура на GSM мрежа

1.2.2. Сравнение между различни видове клетъчни технологии

Източниците на информация са дадени в [9] - [11]. Мобилните клетъчни технологии условно се разделят на няколко поколения (generations). Всяко от тях се означава с нотацията lG , където l е поредният номер на поколението. Всяко ново поколение бележи значителни подобрения в скоростта, качеството и надеждността на мобилните услуги. Към днешна дата най-новото поколение е 5-тото, по-известно като 5G. В таблица 1.1. е направено обстойно сравнение между различните поколения - техните максимални скорости на предаване, методи за множествен достъп, услуги, стандарти и т.н.

| Технология Особеност \ | 1G | 2G | 3G | 4G | 5G |
|---------------------------|-------------------------------------|---|--|---|-----------------------------------|
| Възникване / Внедряване | 1970 - 1980 | 1990 - 2004 | 2004 - 2010 | 2010+ | 2015+ |
| Скорост на предаване | 2 kbps | 14.4 kbps - 384 kbps | 2 Mbps | 1 Gbps | >1 Gbps |
| Стандарти | NMT, AMPS, Hicap, CDPD, TACS, ETACS | GSM, GPRS, EDGE, ... | WCDMA, CDMA 2000 | OFDMA, MC-CDMA, LTE | WWWW, CDMA, BDMA |
| Услуги | Аналогово гласово предаване | Дигитално гласово предаване, пакетизирани данни, по-висок капацитет | Интегрирано висококачествено аудио, предаване на видео и данни | Динамичен достъп до информация, подходящи за носене устройства, висококачествен стрийминг, глобален роуминг | Подобрени услуги, наследени от 4G |
| Множествен достъп | FDMA | TDMA CDMA | CDMA | CDMA | CDMA & BDMA |
| Вид комутация | Канална | Канална Пакетна | Канална Пакетна | Пакетна | Пакетна |
| Основна (Core) мрежа | PSTN | PSTN | Пакетна мрежа | Интернет | Интернет |

Таблица 1.1. Сравнение между всички поколения мобилни технологии

По-долу са разгледани някои от най-важните характеристики, принадлежащи на всяко от поколенията:

- **1G**

- Аналогова система;
- Позволява единствено гласово предаване;
- Ненадеждна мрежа;
- Проблеми със сигурността - липса на криптиране.

- **2G**

- Цифрова система;
- Подобрена сигурност;
- Подобрен капацитет (скорост);
- Позволява предаването на SMS (Short Message Service) и MMS (Multimedia Messaging Service) съобщения;
- GPRS стандартът (определен като 2.5G, защото работи както с 2G, така и с 3G мрежите) въвежда пакетна комутация и позволява използването на мобилната мрежа за нискоскоростна връзка с Интернет.

- **3G**

- По-високи скорости на предаване на информация (достигат няколко Mbps);
- Възможност за видео разговори;
- Споделяне на файлове;
- По-високоскоростна интернет връзка, позволяваща използването ѝ за онлайн телевизия, видеоигри и др.

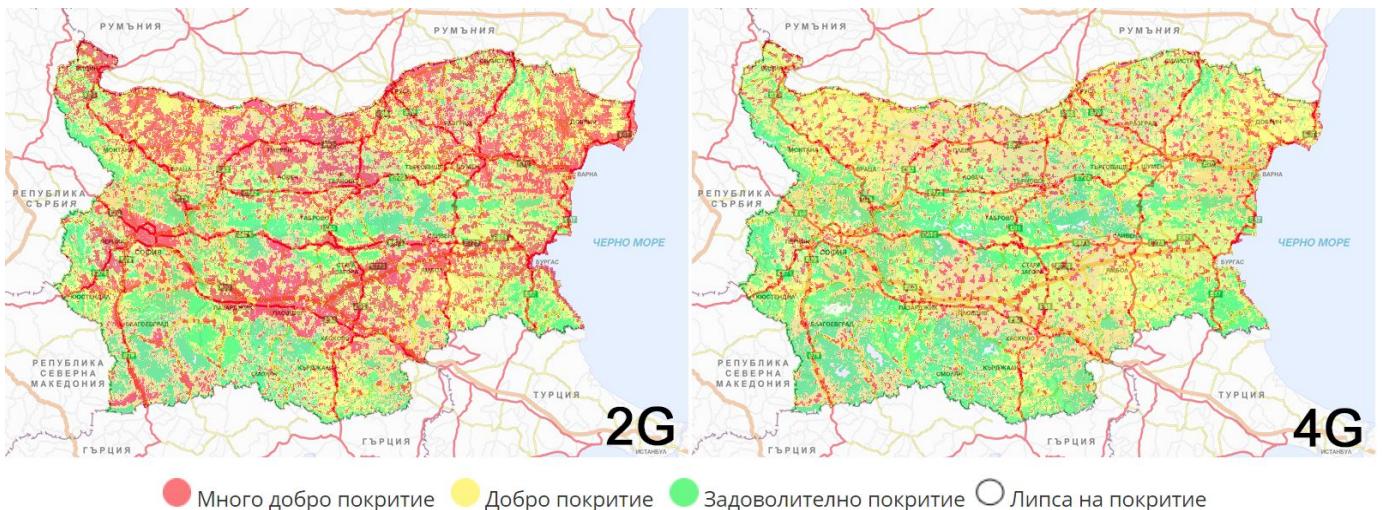
- **4G**

- Високи скорости, теоретично достигащи 1 Gbps;
- Висококачествен видео стрийминг (поточно предаване);
- Възможност за използване на облачни услуги;
- По-ниски закъснения;
- По-добро гласово качество;
- VoIP (Voice over IP).

- **5G**

- Нова технология, целяща постигането на възможно най-високи скорости;
- Предвижда се, че ще е в основата на развитието на "Интернет на нещата" (IoT - Internet of Things);
- Използване в големи бъдещи проекти като "умни" градове, свързани автономни автомобили, изкуствен интелект и т.н.;
- Подобрено качество на всички видове услуги, предоставени от предходящите поколения.

Както личи от сравнението, всяко следващо поколение клетъчни мрежи подобрява скоростите на предаване, предлагани от предишното. Това обаче е свързано с намаляване на покритието на услугите поради използване на по-високи честоти, както и поради не толкова големия брой клетки, актуализирани да поддържат новите стандарти и технологии. Покритието на всяко поколение се разширява постоянно, но както е показано на фиг. 1.13. по-старите технологии са тези, които са достъпни на най-много места в страната.



Фиг. 1.13. Покритие на 2G и 4G мрежите на мобилния оператор "Виваком"

Поради естеството на проектираното устройство е важно постигането на възможно най-голямо покритие, като скоростта на предаване е без голямо значение. Ето защо за проектираното устройство е избрана GPRS технологията, която използва 2G и 3G мобилни мрежи.

1.2.3. General Packet Radio Service технология

Използваните източници на информация са дадени в [12] - [14]. General packet radio service (GPRS) = Пакетна радиовръзка за общо ползване - представлява надстройка над мобилния стандарт от второ поколение - GSM (2G). GPRS е услуга за пакетна комутация, вградена в клетъчната GSM мрежа с комутация на канала. Основната GPRS мрежа позволява на 2G и 3G мобилни мрежи да предават IP пакети към външни мрежи, като например Интернет.

Изграждането на GPRS връзка става на базата на т. нар. име на точка за достъп или APN (Access Point Name). APN представлява името на шлюз, стоящ между клетъчната мрежа и друга компютърна мрежа, като това най-често е Интернет. Устройство, опитващо се да осъществи връзка с дадена пакетна мрежа, трябва да предостави предварително конфигуриран APN на мобилния

оператор. След това операторът изследва този APN, за да определи типът връзка, която трябва да бъде създадена, например какъв IP адрес трябва да бъде предоставен на устройството, какви методи за сигурност трябва да бъдат използвани и т.н. APN може да бъде използван и за определяне на типа услуга, предоставена от пакетната мрежа (Например WAP (Wireless Application Protocol), MMS (Multimedia Messaging Service) и др.). В допълнение към APN, някои мобилни оператори изискват и предоставянето на потребителско име (username) и парола (password) при осъществяване на GPRS връзка. Във фиг. 1.14. са показани GPRS настройки за клетъчната мрежа на мобилния оператор "Теленор".

| WAP през GPRS | |
|---------------|------------------|
| Bearer Type1 | GPRS |
| GPRS APN | telenorbg |
| User name | telenor |
| Profile name | Telenor Internet |

| MMS през GPRS | |
|-------------------|-----------------|
| Data bearer | GPRS |
| GPRS Access point | mms |
| User name | mms |
| IP Address | 192.168.087.011 |
| Port | 8004 |
| Profile Name | Telenor MMS |

Фиг. 1.14. GPRS настройки за WAP и MMS услуги на мобилния оператор "Теленор"

GPRS е "best-effort" услуга, което предполага променливи скорости и закъснения на връзката, зависещи от броя клиенти, свързани към мрежата по едно и също време. Това е така, защото GPRS използва неупотребените канали в система с метод за множествен достъп с разделяне на каналите по време (МД-РКВ), каквато е GSM мрежата.

Скоростта на връзката зависи от използваната схема на кодиране. Сравнение между различните схеми на кодиране е дадено в таблица 1.2.

| Схема на кодиране | Скорост на предаване (без служебна информация) (kbit/s/slot) |
|--------------------------|---|
| CS-1 | 8.00 |
| CS-2 | 12.00 |
| CS-3 | 14.40 |
| CS-4 | 20.00 |

Таблица 1.2. Схеми на кодиране, използвани в GPRS

По-надеждното кодиране позволява по-широко разпространение на сигнала за сметка на скоростта му. Определянето на използваното кодиране става на базата на силата на сигнала, който GPRS устройството получава. При много добър сигнал може да се използва най-бързият, но и най-несигурният вид кодиране (CS-4), а при слаб сигнал се използва най-сигурният, но и най-нискоскоростният вид кодиране (CS-1).

Другият определящ фактор за скоростта на GPRS връзката е т.нар. "мултислот клас". Той определя броя времеви (PKB) слотове, налични в двете посоки на връзката - предаване и приемане (uplink & downlink). Съществуват общо 45 класа, като някои от тях са изобразени в таблица 1.3. Слотовете за предаване и за приемане могат да бъдат комбинирани по произволен начин, но броят им не може да надвишава максималният (активният) брой слотове, характерни за всеки мултислот клас. Така например при избор на мултислот клас 10, разпределението може да стане по 2 начина:

- 4 слота за приемане и 1 слот за предаване ($4+1 = 5$)
- 3 слота за приемане и 2 слота за предаване ($3+2 = 5$)

| Мултислот клас | Времеви слотове за приемане | Времеви слотове за предаване | Активни времеви слотове |
|-----------------------|------------------------------------|-------------------------------------|--------------------------------|
| 1 | 1 | 1 | 2 |
| 5 | 2 | 2 | 4 |
| 8 | 4 | 1 | 5 |
| 10 | 4 | 2 | 5 |
| 32 | 5 | 3 | 6 |
| 14 | 5 | 5 | 6 |

Таблица 1.3. Мултислот класове в GPRS

Комбинацията между избрания мултислот клас, разделението на времевите слотове в него и избраната схема на кодиране определя максималната скорост на GPRS връзката.

Сериозен проблем на GPRS свързаността е липсата на надеждна защита на предаваните данни. Криптирането, използвано в GPRS мрежите, е старо и лесно разбиваемо (дори в реално време). Освен това, това криптиране се прилага единствено при предаване на информация от крайното устройство към най-близката базова станция. При предаване на пакети от базовата станция към определената дестинация, те се пренасят без никакво криптиране. Това налага използването на външни протоколи и методи за защита на предаваната информация.

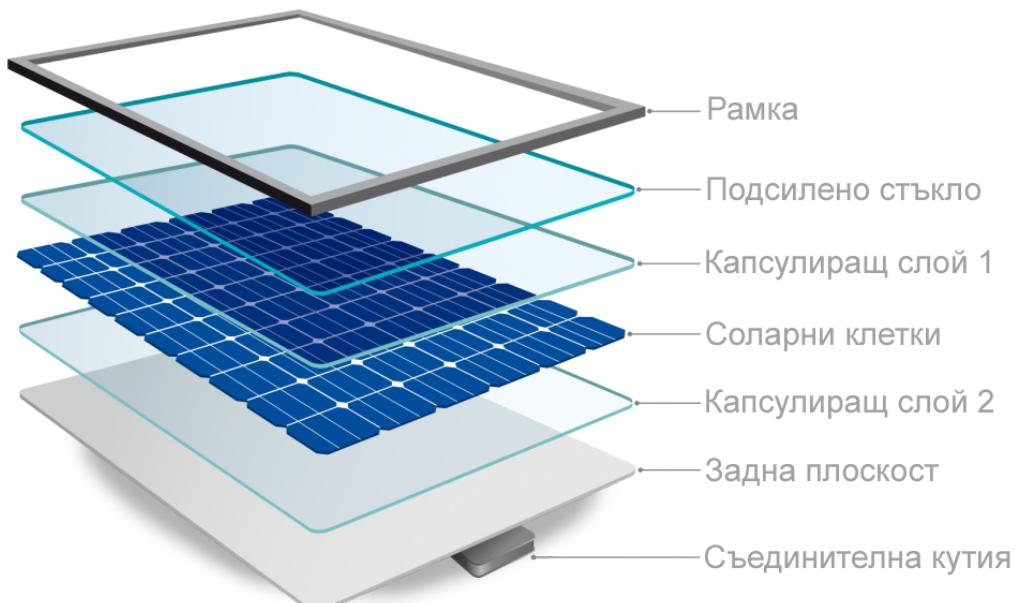
1.3. Фотоволтаично захранване

Фотоволтаичните панели, в комбинация с акумулаторни батерии, са най-достъпният и широко разпространен начин за осигуряване на преизползваемо независимо електрическо захранване. Слънчевата енергия е изключително подходяща за системи, които са разположени далеч от вече изградената електрозахранваща мрежа и/или са труднодостъпни. Поради все още високата му цена, фотоволтаичното захранване често се използва като резерв в случай на повреда в основното електрозахранване. Изискването за висока надеждност и непрекъсната работа на разработваното устройство предполага осигуряването на независимо захранване, за да се предотврати загубата на работоспособност при проблеми в електрическата мрежа. В т. 1.3.1. са разгледани основните особености и характеристики на фотоволтаичния панел. В т. 1.3.2. е обяснен изборът на подходящ тип акумулаторна батерия за фотоволтаични системи при различни приложения. Особеностите на Maximum Power Point Tracking (MPPT) метода, както и принципът на работа на контролери с тази функционалност, са разгледани в т. 1.3.3.

1.3.1. Особености на фотоволтаичния панел

Използваните източници на информация са дадени в [15] - [17]. Фотоволтаичните соларни панели използват слънчева светлина, за да генерират електрически ток. Опростена структура на соларен панел е показана във фиг. 1.15. Основните градивни елементи на всеки соларен панел са т.нар. соларни клетки. Необходимо е тези клетки да бъдат защитени от механични повреди и влага, като това най-често се прави с подсилено стъкло. Повечето

соларни панели са проектирани да издържат на дъжд, тежки снеговалежи и виелици. В задната част на панела се поставя съединителна кутия, която функционира като изходен интерфейс. Тя също е водо- и прахоустойчива.

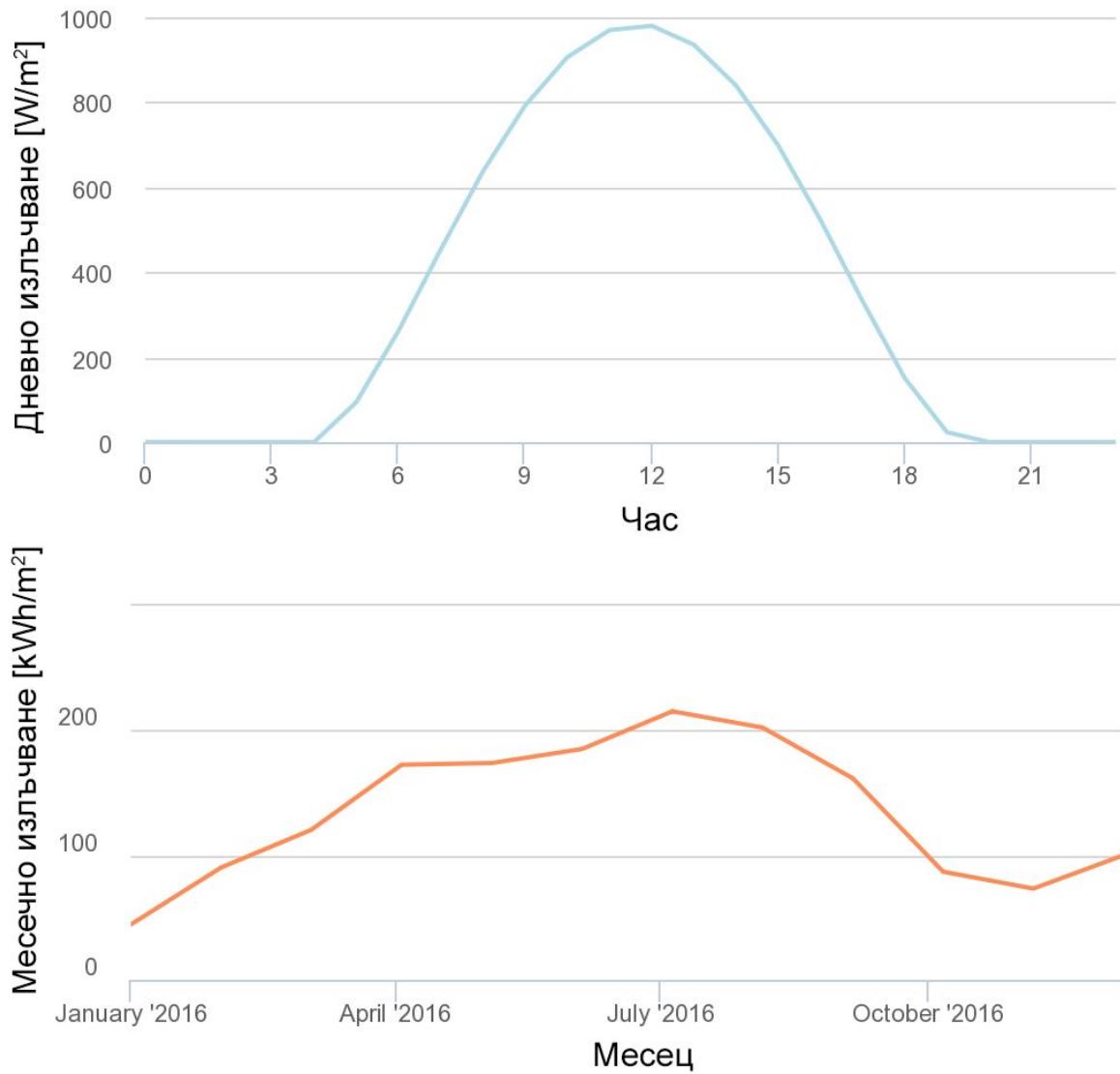


Фиг. 1.15. Структура на соларен панел

Обикновена силициева соларна клетка генерира максимално напрежение на празен ход $\sim 0.5\text{-}0.6\text{V}$. В соларния панел последователно се свързват определен брой клетки, за да се постигне желаното напрежение. В някои случаи няколко такива конструкции се свързват паралелно, за да се увеличи изходният ток.

Най-важните електрически характеристики на соларните панели са: напрежение при максимална мощност (V_{MPP}), ток при максимална мощност (I_{MPP}) и пикова мощност (P_{MPP}). Остаряло понятие, което все още се използва за характеризиране на повечето соларни панели, е номинално напрежение - то означава напрежението на батерия, която панелът е подходящ да зарежда. Често действителното напрежение на панела при директна слънчева светлина е много по-голямо от номиналното напрежение. При панел с 36 соларни клетки например, каквито са повечето 12V соларни панели, напрежението на празен ход достига почти 22V. V_{MPP} на такъв панел е около 18V.

Основният недостатък на соларните панели е лимитираният период от време, в който генерират електричество - те са силно зависими от множество метеорологични фактори (сезон, облачност, температура и т.н.). Както е показано на фиг. 1.16., количеството излъчена слънчева светлина силно варира както в рамките на дененощието, така и в рамките на годината.



Фиг. 1.16. Средно дневно (месец Юни) и средно месечно (2016 година) слънчево излъчване за гр. София

Тъй като соларните системи генерират електрически ток единствено при наличие на слънчева светлина, при нужда от денонощна работа на дадено устройство (както е в случая) е необходимо осигуряването на метод за съхранение на енергия. Най-често, при независимите от електрическата мрежа соларни системи, съхранението на електричество става с помощта на акумулаторни батерии, които се зареждат от соларните панели през деня и осигуряват електричество на системата през нощта или при липса на слънчева светлина.

1.3.2. Акумулаторни батерии за фотоволтаично захранване

Използваните източници на информация са дадени в [18] - [20]. Изборът на правилен тип батерии за съхранение на соларна енергия е от

изключително значение за цената и надеждността на крайния продукт. Съществуват множество различни видове акумулаторни батерии, но най-разпространени в соларните системи са:

- **Оловно-киселинни батерии с течен електролит (Flooded Lead-Acid или FLA)** - характерно за тях е че се пълнят с вода (оттук и името им). Това налага честа поддръжка и доливане на вода на всеки 1-3 месеца, за да се предотвратят неизправности.;
- **Запечатани оловно-киселинни батерии (Sealed Lead-Acid или SLA)** - за разлика от FLA, не изискват поддръжка и на практика са необслужваеми. Съществуват два основни типа SLA батерии - Absorbed Glass Mat (AGM) и Gell cell. Те също се наричат Valve Regulated Lead-Acid (VRLA) батерии, защото имат клапан за отвеждане на газовете при повишено налягане в клетката. Тяхно предимство е че могат да бъдат поставяни във всяка позиция. Те имат редица недостатъци пред по-скъпите литиеви батерии, но за сметка на това са много по-евтини;
- **Литиеви батерии (Lithium Iron Phosphate, съкратено LiFePO4 или LFP)** - нова технология с дълъг живот и висока надеждност. Не изискват никаква поддръжка. Недостатъкът им е високата цена.

Основните разлики между изброените типове акумулаторни батерии са:

- **Живот на батерията (в цикли на зареждане)** - най-често животът на една батерия се измерва в брой цикли на зареждане. За един цикъл се смята употребяване и зареждане на пълния капацитет на дадена батерия, като не е задължително това да стане наведнъж. В това отношение литиевите батерии са няколко пъти по-издръжливи от двата вида оловни батерии;
- **Дълбочина на разряд** - обозначава какъв процент от капацитета на батерията се препоръчва да бъде употребен преди зареждането ѝ. При оловните батерии, дълбочината на разряд не бива да надвишава 50%. При литиевите, тя може да достигне до над 80%. Това означава, че литиевите батерии имат по-висок използваем капацитет;
- **Ефективност** - означава каква част от енергията, използвана за зареждането на батерията, може да бъде употребена при разряд. За FLA и SLA батериите тя е в рамките на 80-85%. При LFP батериите ефективността достига 95%. Това означава, че при 1000W мощност за зареждане, оловните батерии ще разполагат с 800-850W, а литиевите - с 950W;
- **Скорост на зареждане** - обикновено се означава като част от капацитета С на батерията (например C/5). Оловна батерия обикновено се зарежда около два пъти по-дълго от литиева батерия със същия капацитет;

- **Пътност на енергията** - литиевите батерии имат много по-голяма енергийна пътност, което ги прави значително по-малки от оловни батерии с подобен капацитет.

Въпреки всички изброени предимства на LFP батериите, изключително високата им цена в сравнение с FLA и SLA батериите ги прави неподходящи за повечето приложения. LFP батерии най-често се използват като по-надеждна и неизискваща поддръжка алтернатива на FLA батериите за големи соларни системи, които не са свързани към електроснабдяващата мрежа. При рядко използване на батериите (например при използването им за резервен запас) или при ниска консумация, най-подходящи са евтините SLA батерии.

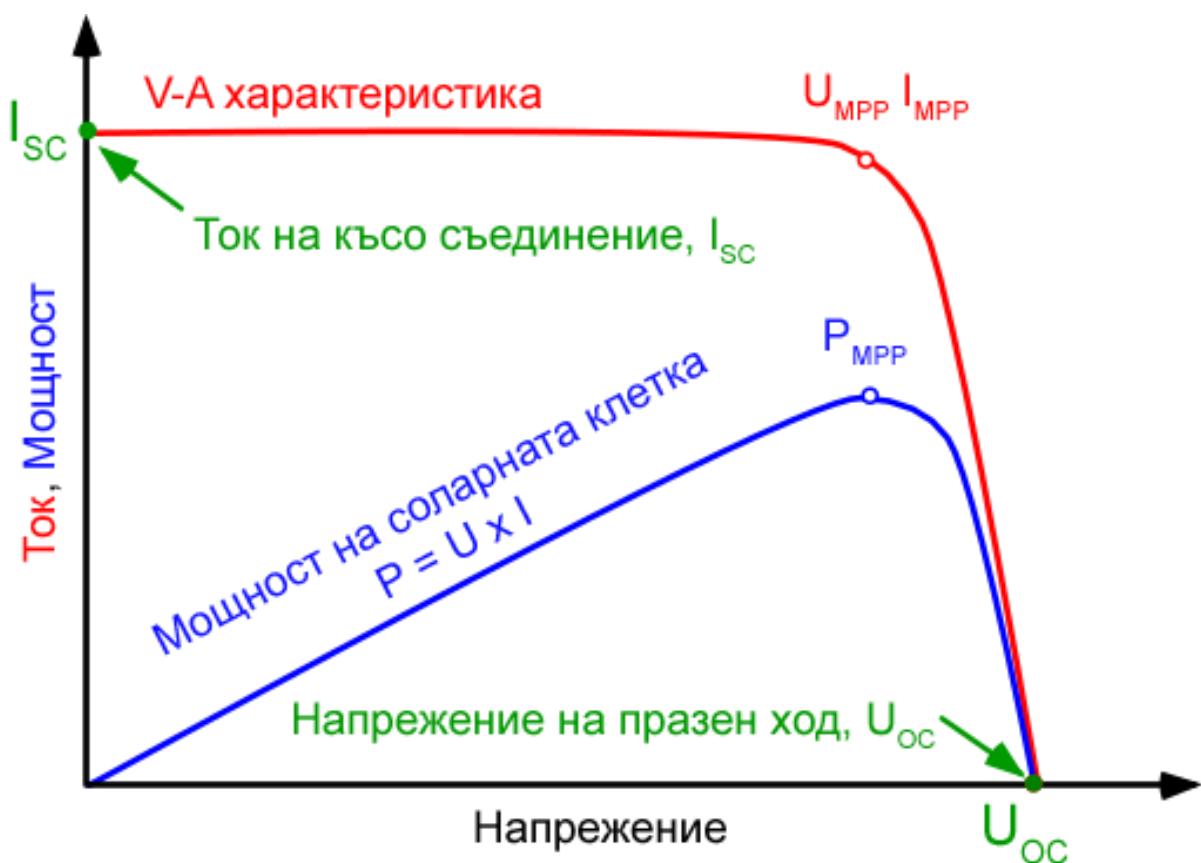
1.3.3. Maximum Power Point Tracking контролери

Използваните източници на информация са дадени в [21] - [25]. Съществуват два основни начина за подобряване на ефективността на фотоволтаична система. Единият от тях е чрез монтирането на соларния панел на т. нар. соларен трекер. Такъв е показан във фиг. 1.17. Той представлява стойка, задвижвана от електромотори, която автоматично променя положението си по такъв начин, че соларният панел винаги да е насочен към слънцето. Такива трекери обикновено предлагат около 15% увеличение на мощността през зимата и до 35% през лятото. Високата цена на соларните трекери ги прави неподходящи за използване в малки системи.



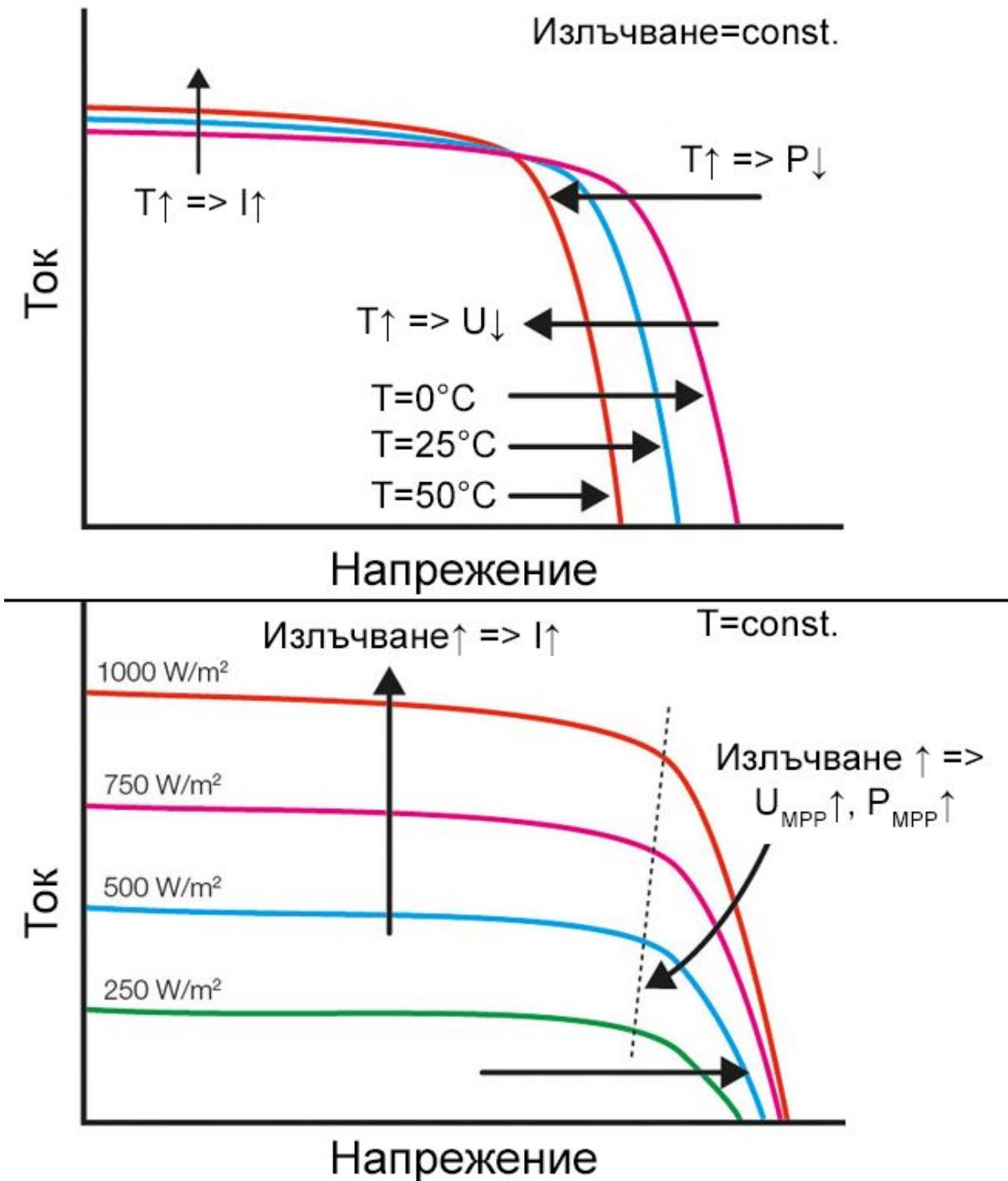
Фиг. 1.17. Соларен трекер

Другият метод за повишаване на добитата от соларните панели мощност е чрез т.нар. следене на точката на максимална мощност или Maximum Power Point Tracking (MPPT). Във фиг. 1.18. са показани двете основни електрически характеристики на соларните клетки - характеристика на мощността и волт-амперна характеристика. Както се вижда от фигурата, за всеки набор от експлоатационни условия съществува работна точка, в която произведението на тока и напрежението е максимално, т.е. мощността е максимална. Тази точка най-често се нарича maximum power point или MPP. Товар със съпротивление $R = U_{MPP} / I_{MPP}$ (От закон на Ом) черпи максималната възможна мощност от соларния панел.



Фиг. 1.18. Характеристики на соларна клетка

Както е показано на фиг. 1.19., волт-амперната характеристика на соларните клетки е силно зависима от външни фактори като температура и ниво на осветеност. Това предполага периодично отместване на MPP при промяна в атмосферните условия и като резултат от това - периодична промяна на оптималния товар. В основата си, процесът на следене на точката на максимална мощност (MPPT) представлява регулиране на товара на соларния панел с цел извлечане на максималната възможна мощност за конкретните условия.



Фиг. 1.19. Зависимост на волт-амперната характеристика на соларна клетка от температурата и слънчевото излъчване

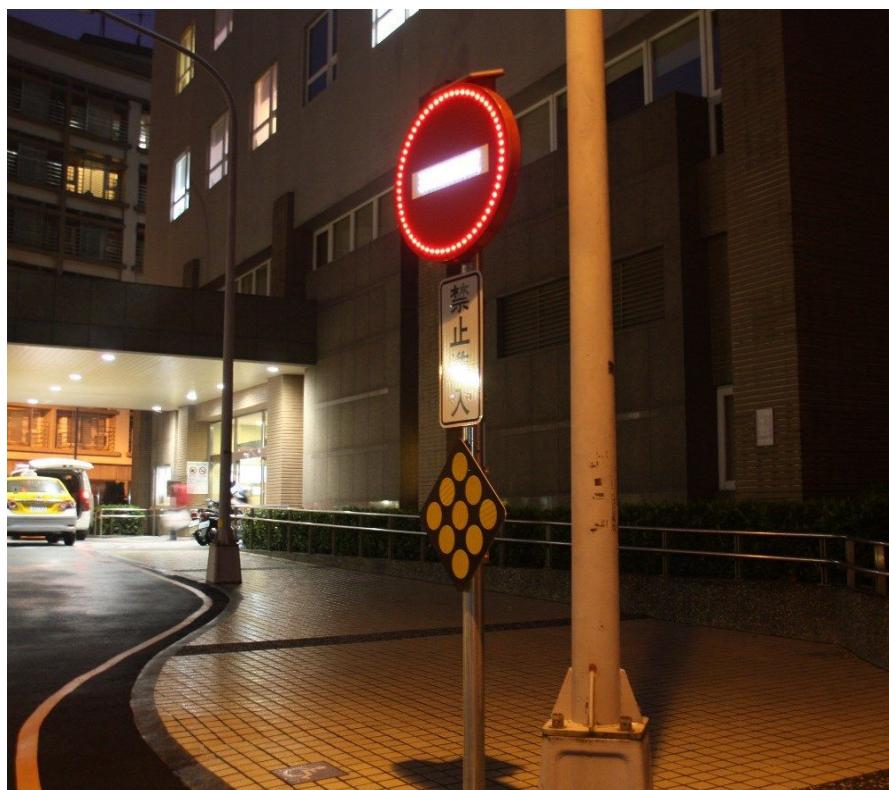
За регулиране на постояннотоково захранване без големи загуби най-често се използват т.нар. импулсни регулатори на напрежение, които обикновено са над 90% ефективни в преобразуването. Всеки MPPT контролер се състои от един такъв регулатор и микроконтролерно устройство, което се грижи за проследяване на точката на максимална мощност. При промяна на честотата на превключване (кофициента на запълване) на регулатора, се

променя и съпротивлението му от гледна точка на панела. Микроконтролерното устройство използва това явление, за да следи промяната в мощността на соларния панел при промяна на товара. Съществуват множество методи за MPPT, но всеки от тях цели максимално бързото откриване на MPP и следователно постигането на минимални загуби на енергия. Поради по-високите стойности на генерираната от панела мощност през зимата, MPPT контролерите предлагат от 20 до 45% увеличение на добитата мощност през зимата и от 10 до 15% през лятото. Това значително подобряне в ефективността, както и не толкова високата им цена, ги прави идеален избор за соларни системи от всякакви размери.

1.4. Съществуващи решения и реализации

Повечето съществуващи разработки, подобни на проектираното устройство, могат да бъдат разделени в две категории:

- **Пътни знаци със светодиодна подсветка** - подобен е показан във фиг. 1.20. При тях светодиодите служат за по-добро откряяване на пътния знак през ноцта и при понижена видимост. За разлика от разработвания проект, при такъв вид знаци светодиодите само осветяват предварително определени указанията, които остават статични;



Фиг. 1.20. Пътен знак със светодиодна подсветка

- **Пътни знаци със светодиодна матрица** - подобни знаци са показани във фиг. 1.21. Най-разпространеното им приложение е за отчитане и изобразяване на скоростта на преминаващите автомобили. Повечето такива пътни знаци се монтират на специални конструкции и се захранват от основната електрозахранваща мрежа. Съществуват и подобни преносими знаци с вграден соларен панел. Начинът за комуникация със знаците варира според производителя.



Фиг. 1.21. Пътни знаци със светодиодна матрица

Втора глава

Проектиране на блоковите схеми на микроконтролерна система за дистанционно управление на светодиоден пътен знак

2.1. Функционални и електрически изисквания към системата

2.1.1. Функционални изисквания към устройството

- Да се осигури визуализация на различни пътни знаци с помощта на светодиодна матрица;
- Да се осигури интернет достъп на системата чрез модул за връзка с GPRS мобилна мрежа;
- Да се осигури управление на интернет връзката и визуализацията на информация чрез микроконтролер;
- Да се осигури захранване чрез акумулаторна батерия;
- Да се осигури зареждане на акумулаторната батерия чрез фотоволтаичен панел;
- Да се оптимизира добитата от фотоволтаичния панел енергия чрез MPPT контролер;
- Да се проектира, поръча и оживи печатна платка на микроконтролерното устройство;
- Да се създаде работоспособен прототип на системата.

2.1.2. Софтуерни функционални изисквания към системата

- Да се създаде микроконтролерно приложение, което позволява свързване към уеб сървър, обработка и изпълнение на получени от него заявки, както и визуализация на различни пътни знаци според получените заявки;
- Да се създаде компютърно приложение за управление и мониторинг на устройствата;
- Да се създаде уеб сървър, който да може да обработва изпратените от компютърните приложения заявки и да изпраща заявки към микроконтролерните устройства;

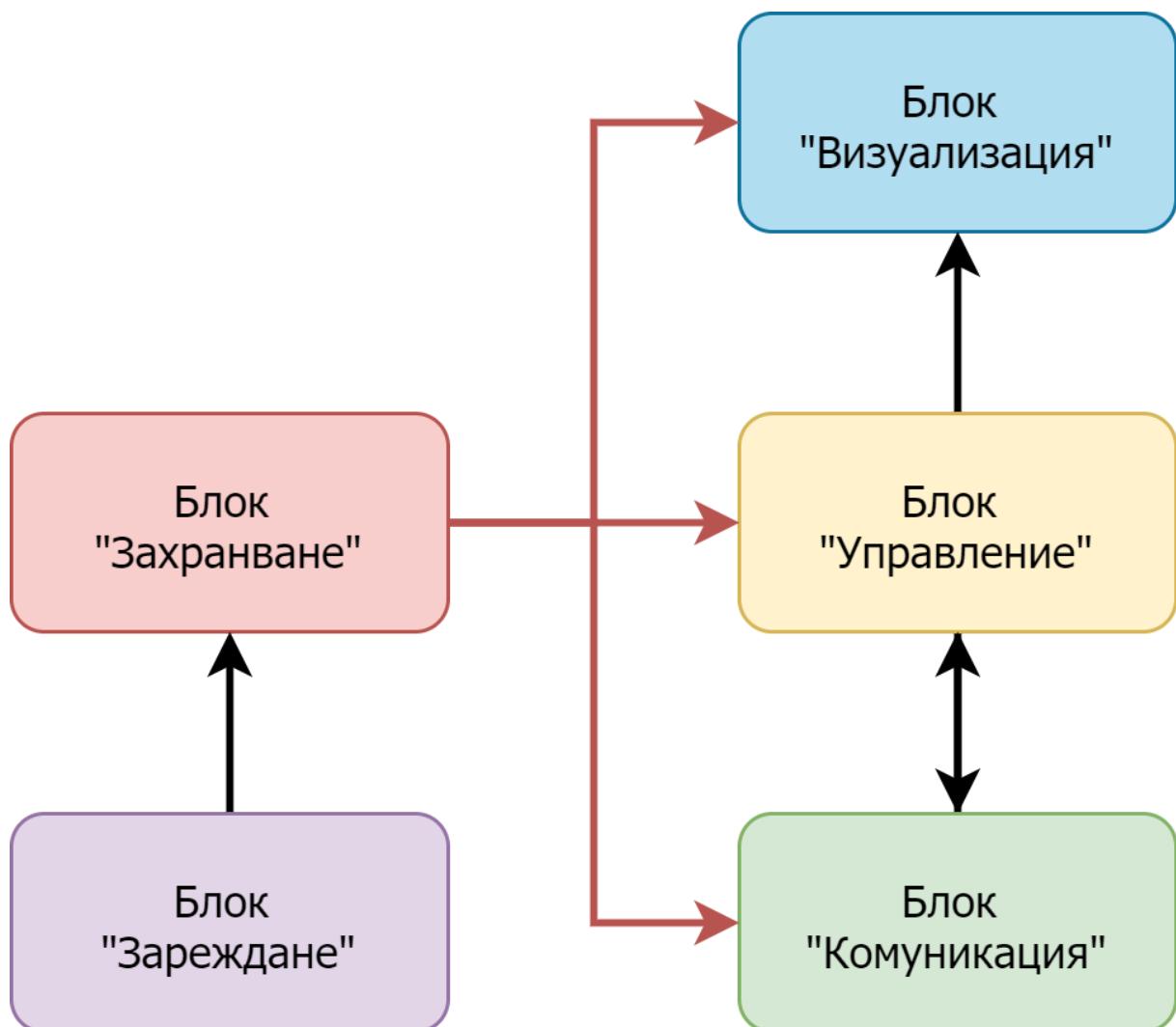
- Да се осигури надеждна връзка на микроконтролерните устройства с уеб сървъра;
- Да се осигури сигурна връзка между управляващото приложение и уеб сървъра;
- Да се осигури упълномощаване на потребителите, които се опитват да достъпят управляващото приложение;
- Да се създаде протокол, описващ всички видове заявки от страна на клиентите към уеб сървъра и от страна на уеб сървъра към микроконтролерните устройства.

2.1.3. Електрически изисквания към устройството

- Да се подбере подходящ микроконтролер с достатъчно оперативна памет и изчислителна мощ - ATmega328P-PU;
- Да се подбере подходящ GPRS модул за надеждна връзка с уеб сървър - SIM800L;
- Да се подбере подходяща светодиодна матрица с достатъчно висока резолюция за постигане на яснота на изобразявания пътен знак - Adafruit 32x32 RGB LED Matrix Panel;
- Да се подбере акумулаторна батерия с достатъчно голям капацитет за захранване на цялата система - 12V/14Ah SLA battery;
- Да се осигури зареждане на акумулаторната батерия чрез подходящ фотоволтаичен панел - 10W/12V solar panel;
- Да се подбере подходящ MPPT контролер за зареждане на оловна (SLA) акумулаторна батерия - BQ24650 charge controller;
- Да се проектира схема с импулсен регулатор на напрежение LM2596-5.0 с цел осигуряване на подходящо захранване за микроконтролера (5V), като се регулира захранването от акумулаторната батерия (12V);
- Да се проектира схема с импулсен регулатор на напрежение LM2596-ADJ с цел осигуряване на подходящо захранване за GPRS модула (4V), като се регулира захранването от акумулаторната батерия (12V);
- Да се осигури комуникация между микроконтролера и GPRS модула посредством серийна връзка;
- Да се осигури по-бърз външен осцилатор за микроконтролера;
- Да се подберат подходящи конектор и антена за GPRS модула с цел постигане на достатъчно голям обхват;
- Да се предвиди метод за лесно изключване на захранването към устройството;
- Да се подберат подходящи филтови кондензатори за интегралните схеми.

2.2. Проектиране на блоковата схема на устройството

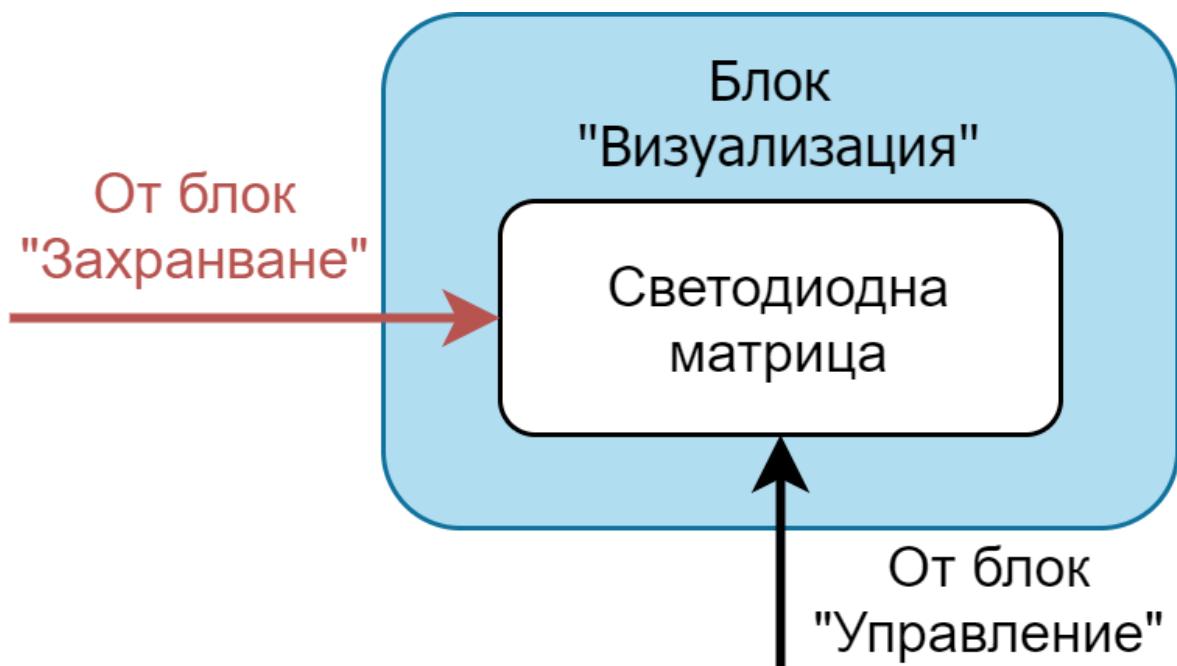
Във фиг. 2.1. е показана пълната блокова схема на проектираното микроконтролерно устройство. Подробни описания на предназначението и структурата на всеки от блоковете са дадени в т. 2.2.1. - т. 2.2.5.



Фиг. 2.1. Блокова схема на микроконтролерното устройство

2.2.1. Блок "Визуализация"

Структурата на този блок е показана във фиг. 2.2. Той се състои от светодиодна матрица, която се захранва от блок "Захранване". Основното му предназначение е изобразяването на информацията, получена от блок "Управление".

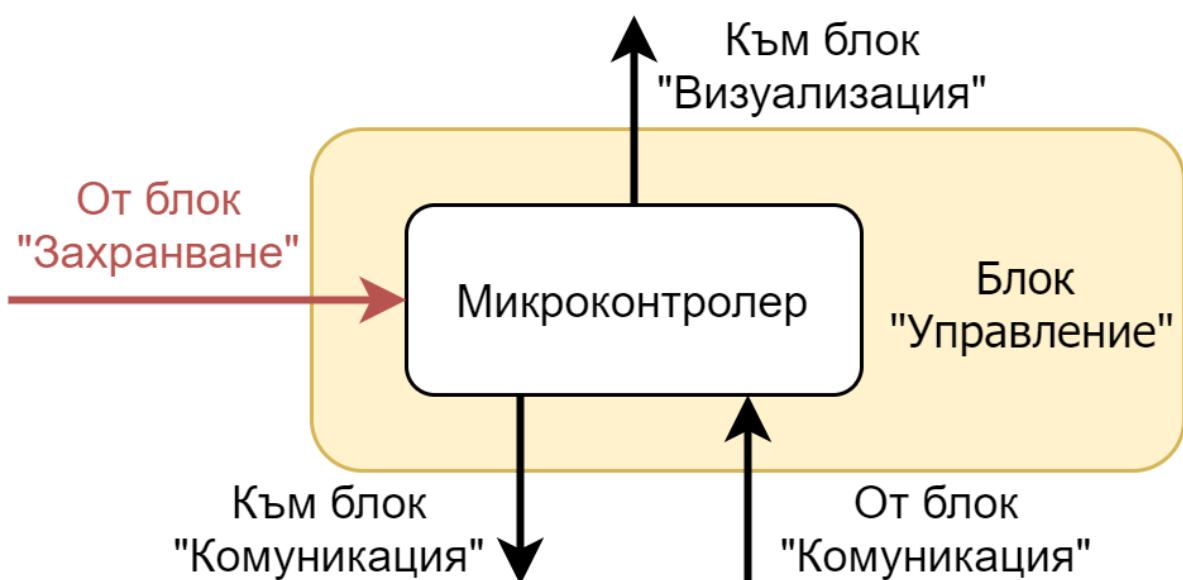


Фиг. 2.2. Структура на блок "Визуализация"

2.2.2. Блок "Управление"

Структурата на този блок е показана във фиг. 2.3. Той е изграден от микроконтролер, който се захранва от блок "Захранване". Този блок изпълнява всички функции, свързани с логиката на устройството:

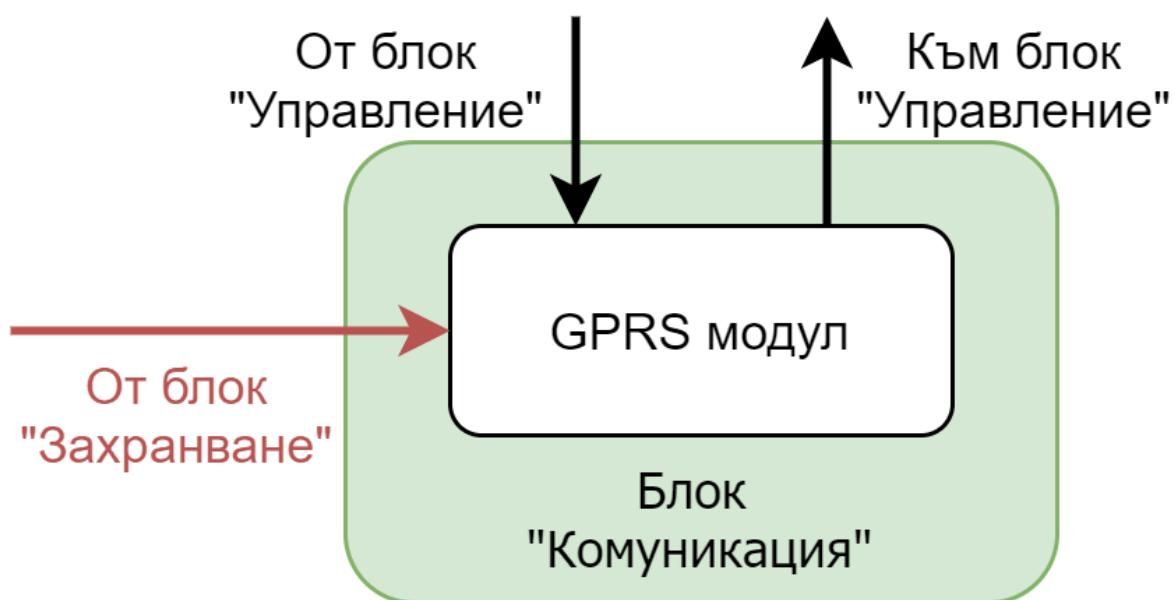
- Осъществява първоначалното свързване към уеб сървъра, като това става чрез изпращане на команди към GPRS модула от блок "Комуникация"
- Грижи се за поддържането на изградената сесия
- Обработва заявките, изпратени от уеб сървъра и прехвърлени към него от блок "Комуникация"
- Изпраща отговори на тези заявки чрез блок "Комуникация"
- Грижи се за постоянно изчисляване, зареждане и изпращане на информация към светодиодната матрица от блок "Визуализация" според получените заявки



Фиг. 2.3. Структура на блок "Управление"

2.2.3. Блок "Комуникация"

Структурата на този блок е показана във фиг. 2.4. Той е изграден от GPRS модул, чието захранване се осигурява от блок "Захранване". Блокът служи за осигуряване на връзка с интернет и изграждане на сесия с уеб сървъра, който управлява устройството. Основното му предназначение е да управлява GPRS връзката и при нужда да предава заявките от уеб сървъра към микроконтролера от блок "Управление" и обратно. При загуба на свързаност, този блок се грижи за повторното ѝ изграждане, като този процес се управлява от блок "Управление".



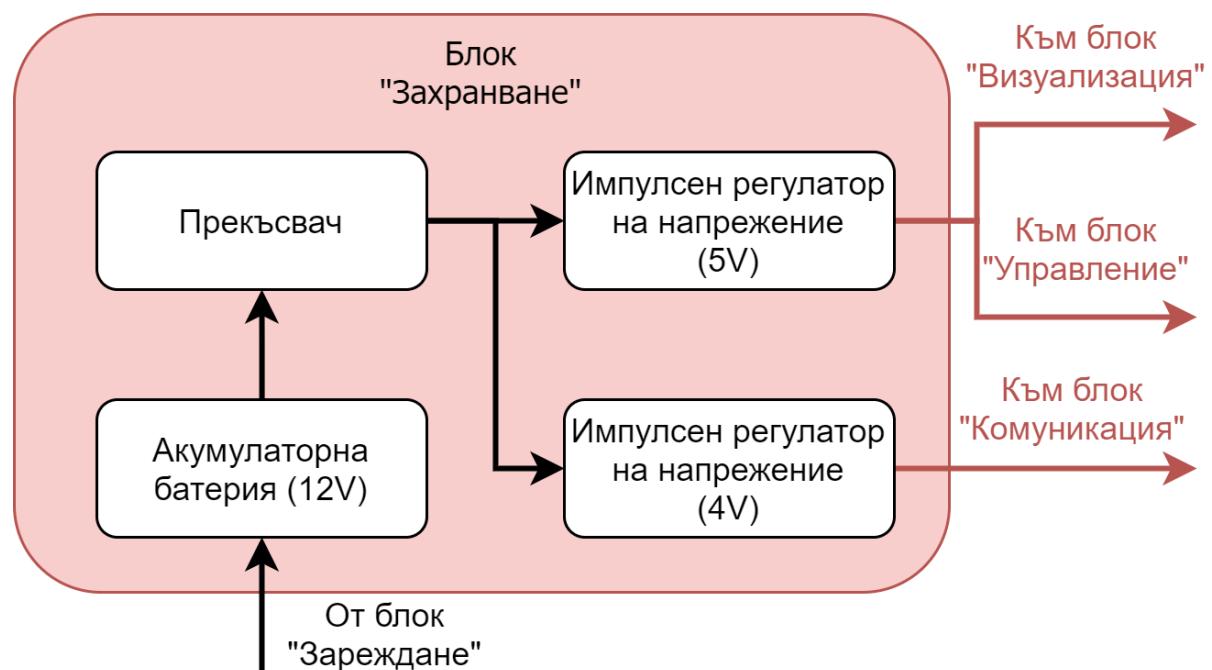
Фиг. 2.4. Структура на блок "Комуникация"

2.2.4. Блок "Захранване"

Структурата на този блок е показана във фиг. 2.5. Той е изграден от няколко елемента:

- **Акумулаторна батерия** - основният източник на захранване за системата;
- **Прекъсвач** - служи за включване и изключване на устройството;
- **Импулсни регулатори на напрежение** - служат за понижаване на захранващото напрежение от акумулаторната батерия в подходящо за захранване на другите блокове на устройството. Тъй като GPRS модулът от блок "Комуникация" работи с по-ниско захранващо напрежение, се налага използването на два такива регулатора.

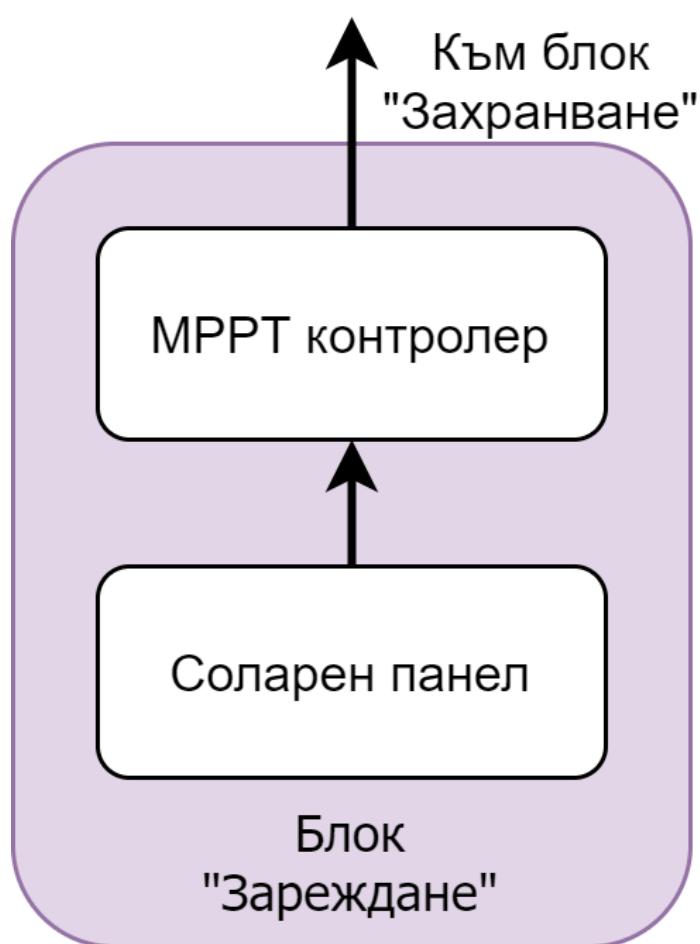
Основното предназначение на този блок е осигуряването на коректните захранващи напрежения за всички останали блокове на устройството (освен блок "Зареждане"). Също така, този блок се грижи за включването и изключването на цялото устройство посредством прекъсвач.



Фиг. 2.5. Структура на блок "Захранване"

2.2.5. Блок "Зареждане"

Структурата на този блок е показана във фиг. 2.6. Той се състои от фотоволтаичен (соларен) панел и MPPT контролер. Основното му предназначение е зареждането на акумулаторната батерия от блок "Захранване" чрез соларна енергия. Блокът се грижи за оптимизирането на добитата от соларния панел енергия, както и за осигуряването на коректно зареждащо напрежение за батерията.



Фиг. 2.6. Структура на блок "Зареждане"

Трета глава

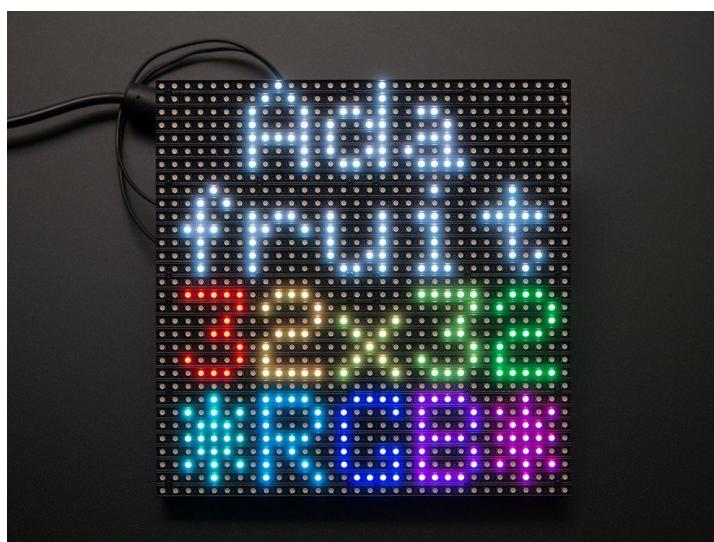
Проектиране на принципната електрическа схема на микроконтролерна система за дистанционно управление на светодиоден пътен знак

3.1. Проектиране на принципните електрически схеми на блоковете на устройството

Процесът на проектиране на принципните електрически схеми на всеки от блоковете на устройството, както и изборът на подходяща елементна база, са описани в т. 3.1.1. - т. 3.1.5. За проектиране на електрическите схеми и печатната платка на устройството е използван CAD (Computer-Aided Design) софтуерът с отворен код KiCAD

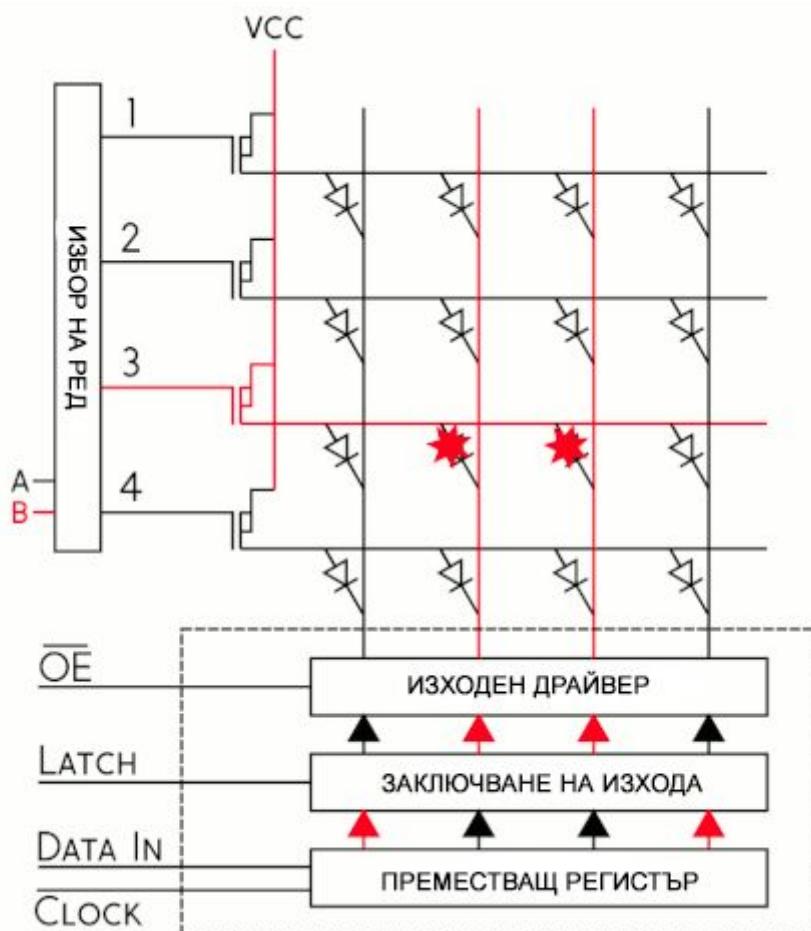
3.1.1. Принципна електрическа схема на блок "Визуализация"

Използваните източници на информация са дадени в [26] и [27]. Светодиодната матрица, избрана за устройството, е Adafruit 32x32 RGB (RGB - Red Green Blue) LED Matrix Panel (Показана във фиг. 3.1.). Каталожна информация за тази матрица е приложена в {1}.



Фиг. 3.1. Adafruit 32x32 RGB LED Matrix Panel

Поради големия брой светодиоди в матрицата, избраният от производителя начин на управление е чрез преместващи регистри (обяснен по-подробно в т. 1.1.2). Опростена схема на управлението на използваната матрица е дадена във фиг. 3.2.



Фиг. 3.2. Опростена схема на управлението на използваната LED матрица

Визуализацията на всеки от редовете минава през следните етапи:

- 1) **Записване на комбинацията за съответния ред в преместващия регистър** - в случая на проектираното устройство тази стъпка отнема време, равняващо се на 32 тактови импулса (32 светодиода на ред);
- 2) **Подаване на високи нива на LATCH и /OE изводите** - по този начин информацията в преместващия регистър достига до изходния драйвер. /OE изводът в неактивното си състояние (високо логическо ниво) не позволява активирането на светодиоди по време на тази стъпка, за да се предотврати паразитното светене при промяната на избрания ред;
- 3) **Промяна на избрания ред чрез A и B изводите;**
- 4) **Подаване на ниски нива на LATCH и /OE изводите** - позволява на записаните в изходния драйвер нива да достигнат съответния ред, като

същевременно "заключва" информацията, получена от преместващия регистър. По този начин коректните за съответния ред нива не се променят, докато в преместващия регистър се записват нивата за следващия ред.

Както личи от разгледания по-горе принцип на работа, матрицата визуализира всеки кадър ред по ред. По този начин се избягва едновременното светене на всички светодиоди, което е свързано с огромна консумация на ток - общо 61,44A (формула (3.1.), където $N = 32*32 = 1024$ активни светодиода).

$$I_{Matrix_{(MAX)}} = I_{LED_{(MAX)}} * N \quad (3.1.)$$

$I_{Matrix_{(MAX)}}$ - Максимална консумация на матрицата (Всички светодиоди активни, пълна яркост, бял цвят)

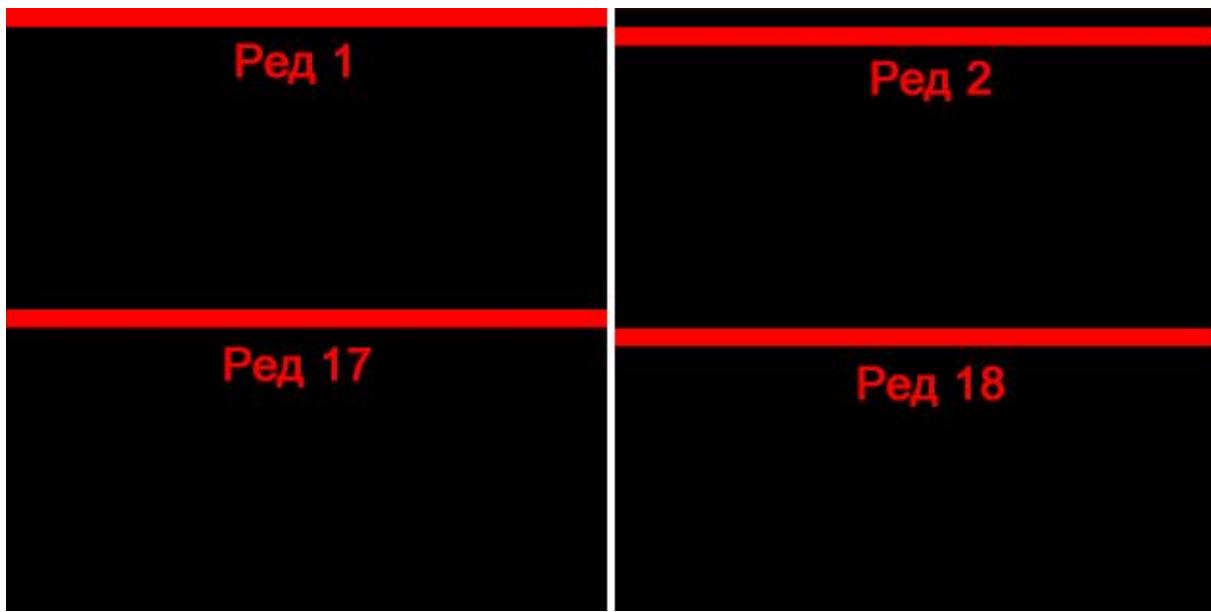
$I_{LED_{(MAX)}}$ - Максимална консумация на един RGB светодиод (Пълна яркост, бял цвят). Обикновено е $\approx 60mA$

N - Брой активни светодиоди във всеки един момент

$$I_{Matrix_{(MAX)}} = 60mA * (32 * 32) = 61\ 440mA = 61,44A$$

При достатъчно бързи скорости на опресняване (точните стойности зависят от множество фактори, но за повечето приложения се използват честоти $>300Hz$ за всеки светодиод), човешкото око не може да регистрира светенето на отделни редове и в него се създава илюзията за цялостна картина върху пълната матрица.

Като компромис между скорост и консумация, за използваната светодиодна матрица е избрана честота на сканиране 1:16. Това означава, че във всеки един момент от време 1/16 от всички пиксели, т.е. 2 реда, са активни. Както е показано на фиг. 3.3., единият активен ред се избира от горната половина на дисплея, а другият - от долната (ред 1 с ред 17, ред 2 с ред 18 и т.н.). Това се прави с цел да се намали трептенето при обновяване на кадрите.



Фиг. 3.3. Едновременно активни редове на матрицата

С така избраната честота на сканиране, максималната моментна консумация на матрицата е 3,84A (формула 3.1., където $N = 1024/16 = 64$ активни светодиода)

$$I_{Matrix_{(MAX)}} = 60mA * (1024 / 16) = 3840mA = 3,84A$$

Имайки предвид това, при проектиране на електрическа схема с такава матрица могат да се използват значително по-малки и евтини захранващи блокове. За захранване на матрицата е изведен съединител с 4 извода.

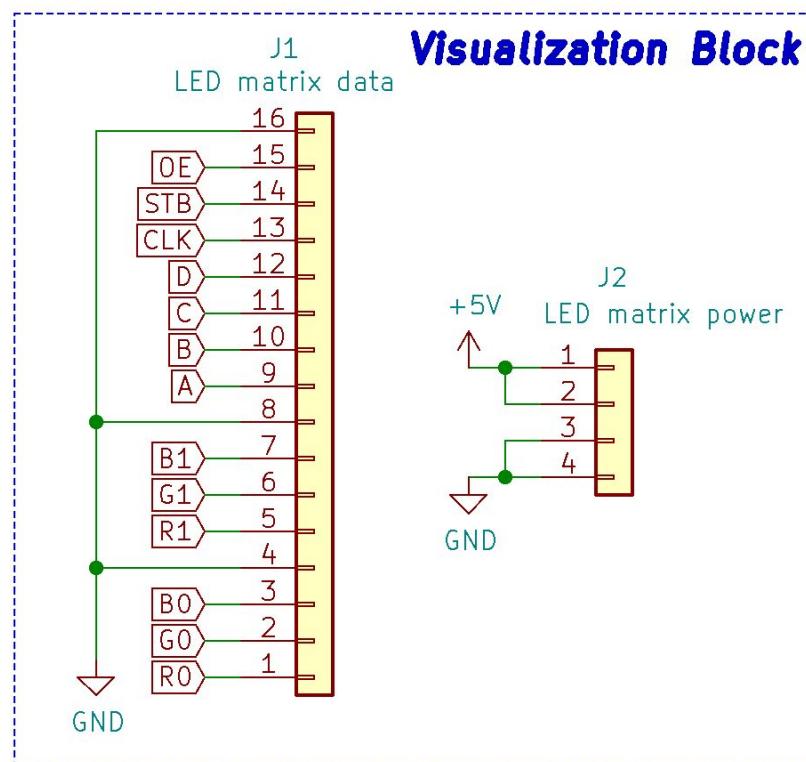
Връзката за изпращане на информация към светодиодната матрица става със съединител с 16 извода, като предназначението на всеки от тях е описано в таблица 3.1.

| Извод (означение) | Предназначение |
|----------------------|--|
| R0 | Състояние на червен светодиод (редове 1-16) |
| G0 | Състояние на зелен светодиод (редове 1-16) |
| B0 | Състояние на син светодиод (редове 1-16) |
| R1 | Състояние на червен светодиод (редове 17-32) |
| G1 | Състояние на зелен светодиод (редове 17-32) |

| | |
|--------|---|
| B1 | Състояние на син светодиод (редове 17-32) |
| A | Вход 1 на демултиплексора за избор на ред |
| B | Вход 2 на демултиплексора за избор на ред |
| C | Вход 3 на демултиплексора за избор на ред |
| D | Вход 4 на демултиплексора за избор на ред |
| CLK | Тактов сигнал (от фиг. 3.2. - CLOCK) |
| STB | Заключване на изхода (от фиг. 3.2. - LATCH) |
| OE | Разрешаване на изхода (от фиг. 3.2. - /OE) |
| 3x GND | Маса |

Таблица 3.1. Информационни изводи на светодиодната матрица

Принципната електрическа схема на блок “Визуализация” е показана във фиг. 3.4. Всички информационни изводи на матрицата са директно свързани към микроконтролера от блок “Управление”. Матрицата използва работно напрежение от 5V, което се осигурява от блок “Захранване”. На схемата с J1 е означен информационният съединител, а с J2 - захранващият.



Фиг. 3.4. Принципна електрическа схема на блок “Визуализация”

3.1.2. Принципна електрическа схема на блок “Управление”

Използваните източници на информация са дадени в [28] - [32]. Микроконтролерът, избран за устройството, е ATMega328P-PU на фирмата Atmel. Каталожна информация за него е приложена в {2}. Неговите основни характеристики, които имат отношение към разработката, са представени в таблица 3.2.

| Характеристика | Стойност |
|--|---|
| Микропроцесор | 8-битов AVR |
| Брой изводи | 28 |
| Работно напрежение | 1.8V до 5.5V |
| Брой програмируеми входни/изходни линии | 23 |
| Комуникационни интерфейси | 2x SPI (Serial peripheral interface) 1x UART (Universal asynchronous receiver-transmitter) 1x I ² C (Inter-integrated circuit) |
| PWM канали | 6 |
| Вътрешен осцилатор | 8MHz |
| Външен осцилатор | 0-4MHz @ 1.8V - 5.5V 0-10MHz @ 2.7V - 5.5V 0-20MHz @ 4.5V - 5.5V |
| Скорост на микропроцесора | 1MIPS (Million Instructions Per Second) за 1MHz на осцилатора |
| Програмна памет | 32KB Flash |
| Динамична памет | 2KB SRAM (Static random access memory) |
| EEPROM (Electrically erasable programmable read-only memory) | 1KB EEPROM |
| Работна температура | -40°C до +105°C |

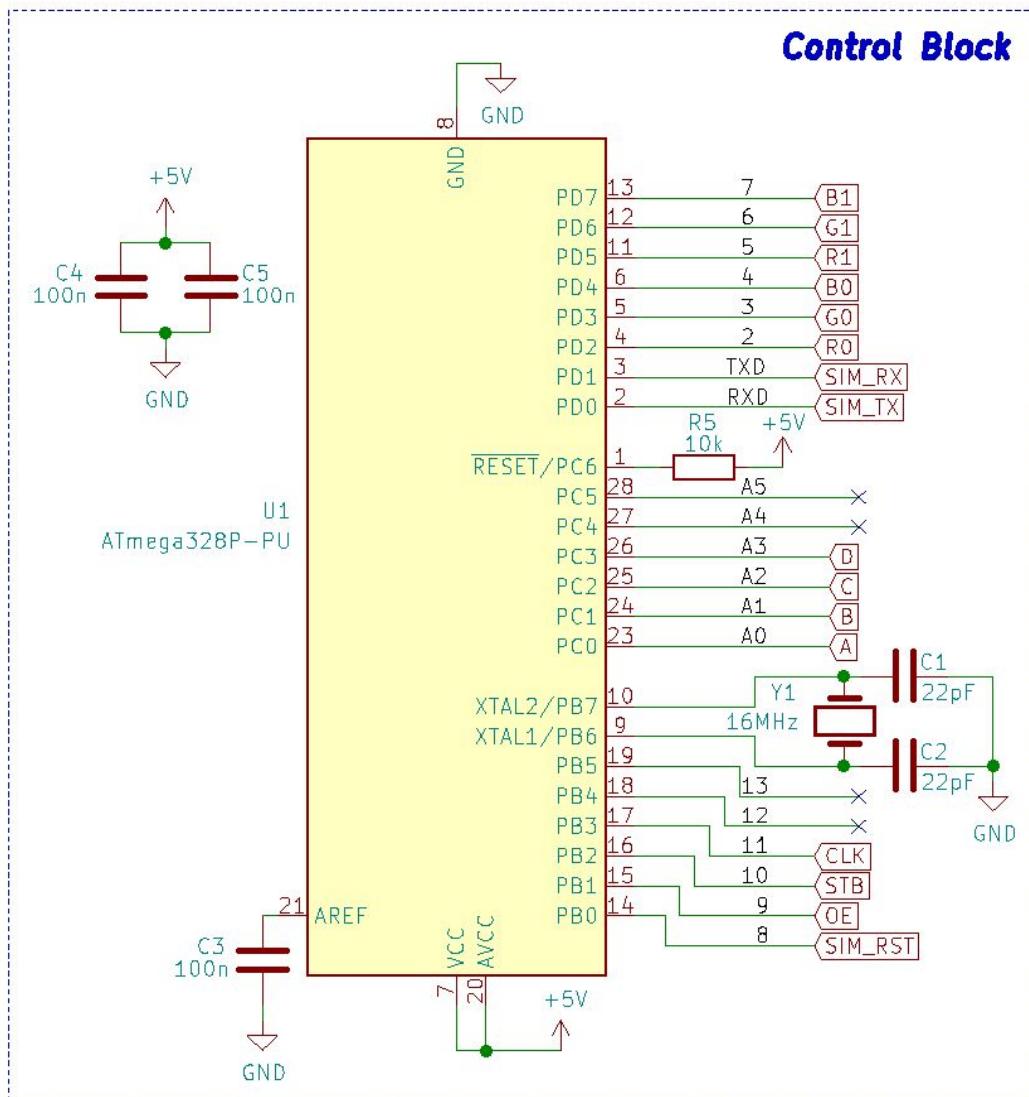
Таблица 3.2. Основни характеристики на микроконтролера ATMega328P

Изискванията, поставени при избора на подходящ микроконтролер за устройството, са:

- Използване на голяма част от наличните изводи;
- Високо натоварване на микропроцесора;
- Използване на голяма част от динамичната и програмната памет;
- Наличие на поне един UART интерфейс за комуникация с GPRS модула.

Тези изисквания имат за цел да минимизират цената на устройството. Предвиден е и известен допълнителен (неизползван) хардуерен ресурс, с цел лесно надграждане и актуализация на програмния код. Микроконтролерът ATMega328P отговаря на всички тези изисквания, като същевременно предлага надеждност и възможност за бъдещо развитие на системата.

Във фиг. 3.5. е показана принципната електрическа схема на блок "Управление". Всеки от изводите на микроконтролера ATMega328P е означен с еквивалентния му извод на развойната платка Arduino UNO R3 с цел улеснение на разработката.



Фиг. 3.5. Принципна електрическа схема на блок "Управление"

Предназначенията на всички изводи, използвани в устройството, са разгледани в таблица 3.3.

| Извод - означение (ATMega328P Arduino UNO) | Предназначение |
|---|--|
| VCC VCC | Положителен захранващ извод на микроконтролера |
| AVCC - | Захранващ извод на АЦП (Аналогово-цифров преобразувател) |
| GND GND | Отрицателен захранващ извод на микроконтролера (маса) |
| AREF AREF | Референтно аналогово напрежение за АЦП |
| $\overline{\text{RESET}}/\text{PC6}$ RESET | Нулиращ извод на микроконтролера |
| XTAL1/PB6 - | Извод 1 за свързване на външен осцилатор |
| XTAL2/PB7 - | Извод 2 за свързване на външен осцилатор |
| PD0 0 (RXD) | Входен информационен извод за UART комуникацията; свързан към изходния UART извод (TXD) на GPRS модула от блок "Комуникация" |
| PD1 1 (TXD) | Изходен информационен извод за UART комуникацията; свързан към входния UART извод (RXD) на GPRS модула от блок "Комуникация" |
| PB0 8 | Изходен извод; свързан към нулиращия извод (RESET) на GPRS модула от блок "Комуникация" |
| PD2 - PD7 2 - 7 | Изходни изводи; свързани към информационните изводи на светодиодната матрица от блок "Визуализация", които служат за зареждане на информацията за всеки от светодиодите в преместващите регистри (R0, G0, B0, R1, G1, B1) |
| PC0 - PC3 A0 - A3 | Изходни изводи; свързани към входовете на демултиплексора за избор на ред (A, B, C, D) на светодиодната матрица от блок "Визуализация" |

| | |
|----------|--|
| PB1 9 | Изходен извод; свързан към извода за разрешаване на изхода (OE) на светодиодната матрица от блок "Визуализация" |
| PB2 10 | Изходен извод; свързан към извода за заключване на изхода (STB) на светодиодната матрица от блок "Визуализация" |
| PB3 11 | Изходен извод; свързан към извода за тактов сигнал (CLK) на светодиодната матрица от блок "Визуализация" |

Таблица 3.3. Предназначение на изводите на микроконтролера за устройството

Захранващото напрежение, избрано за микроконтролера, е 5V. Захранването на ATMega328P става посредством изводите VCC и GND, които в схемата са свързани съответно към положителния и отрицателния терминал на регулатора на напрежение от блок "Захранване". Изводът AVCC също е свързан към +5V, тъй като производителят инструктира той да бъде свързан към захранващото напрежение, дори когато не се използва аналогово-цифровия преобразувател (както в случая). Референтното напрежение за АЦП (извод AREF) е заземено през 100nF кондензатор, за да се подобри шумоустойчивостта (отново по препоръка на производителя). За да се защитят захранващите изводи от шумове, причинени от други елементи в схемата, към тях са добавени и два броя развързвачи (decoupling) кондензатори. Използването на два кондензатора се налага, понеже захранващите изводи (VCC и AVCC) на микроконтролера физически са разположени от противоположни страни на корпуса.

Изборът на захранващо напрежение от 5V е повлиян главно от необходимостта за използване на бърз осцилатор (>8MHz). Това се налага поради нуждата от бързо сканиране на светодиодната матрица от блок "Визуализация", за да се намали трептенето на картина. Повишаването на бързодействието на микроконтролера става чрез осигуряването на високочестотен осцилатор. За устройството е подбран 16MHz кварцов осцилатор, заедно със съпътстващите го 22pF кондензатори (типично приложение, по препоръка на Atmel). Осцилаторът е свързан към изводите XTAL1/PB6 и XTAL2/PB7 на микроконтролера.

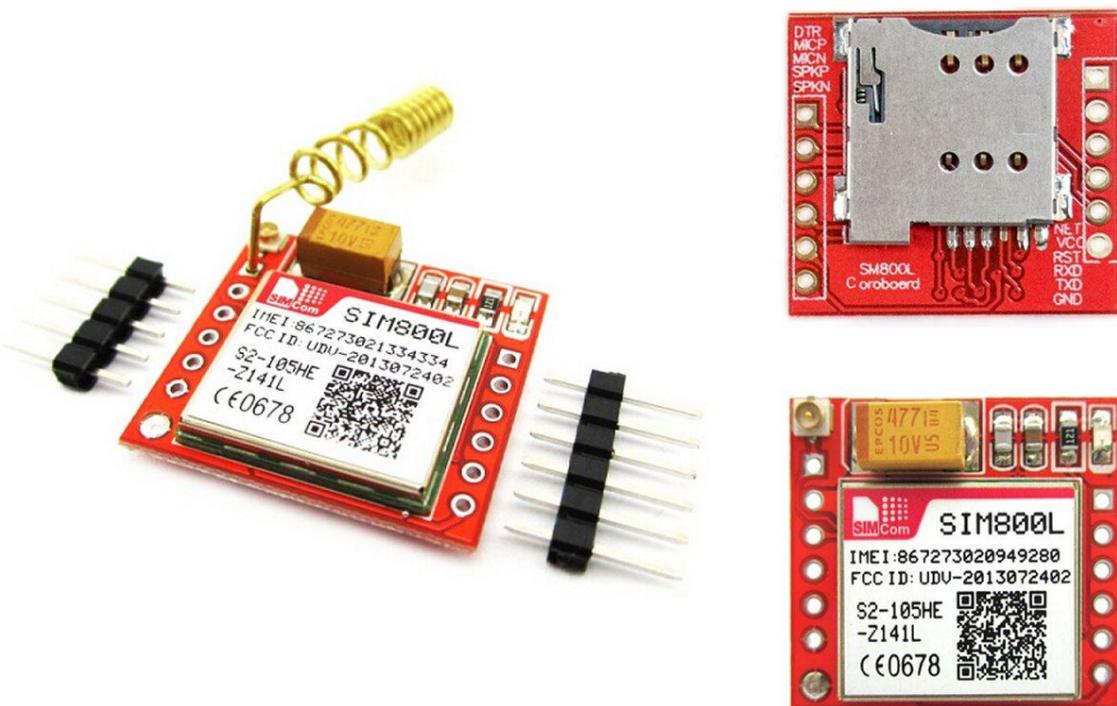
Нулиращият извод на микроконтролера (RESET/PC6) е свързан към +5V през 10kΩ резистор. При подаване на ниско логическо ниво на този извод, микроконтролерът се нулира. Свързването му към високо логическо ниво осигурява нормалната работа на устройството. R5 изпълнява ролята на pull-up резистор и позволява лесното добавяне на метод за нулиране при нужда в бъдещи варианти на разработката.

Връзката между ATMega328P и GPRS модула от блок "Комуникация" става посредством UART връзка. Тъй като този вид комуникация е серийна, за двупосочен обмен на информация се използват само два извода - приемащ (PD0/RXD) и предаващ (PD1/TXD). Тези изводи са съответно свързани към предаващия и приемащия извод на GPRS модула, като по този начин става възможно управлението му от микроконтролера.

3.1.3. Принципна електрическа схема на блок "Комуникация"

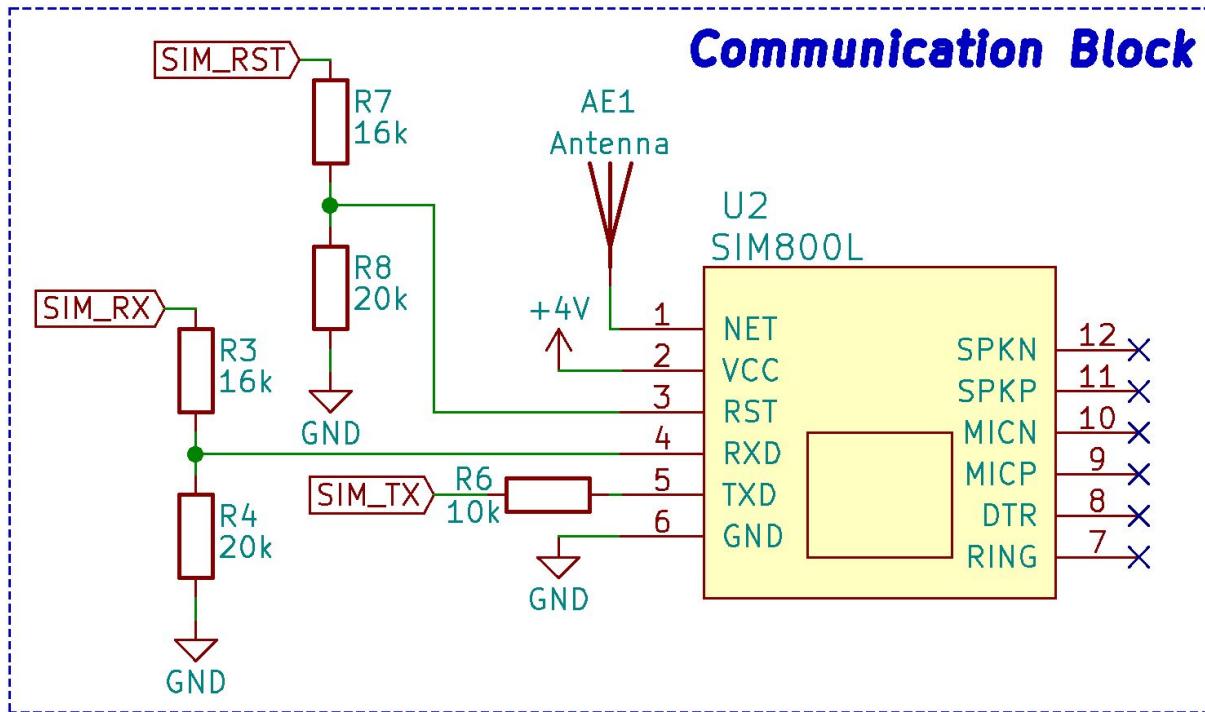
Използваните източници на информация са дадени в [33] - [35]. За устройството е избран GSM/GPRS модулът SIM800 на фирмата SIMCom. Каталожна информация за него е приложена в {3} - {6}. За улеснение на процеса на проектиране на устройството е използвана развойната платка SIM800L (показана във фиг. 3.6.). Някои от по-важните ѝ характеристики включват:

- Изведени всички необходими информационни изводи;
- Micro SIM цокъл;
- u.fl съединител и отделен извод за добавяне на антена;
- 470 μ F кондензатор на захранващите изводи;
- Светодиод за състоянието на мрежата.



Фиг. 3.6. Развойна платка SIM800L

Принципната електрическа схема на блок "Комуникация" е показана във фиг. 3.7. Предназначението на всеки от изводите на GSM/GPRS модула е разгледано в таблица 3.4.



Фиг. 3.7. Принципна електрическа схема на блок "Комуникация"

| Извод - означение | Предназначение |
|-------------------|--|
| VCC | Положителен захранващ извод на модула |
| GND | Отрицателен захранващ извод на модула (маса) |
| NET | Извод за свързване на антена |
| RST | Нулиращ извод на модула |
| RXD | Входен информационен извод за UART комуникацията |
| TXD | Изходен информационен извод за UART комуникацията |
| RING | Индикатор за позвъняване; променя състоянието си при входящо обажддане |

| | |
|-----------|--|
| DTR | Извод за активиране/деактивиране на sleep режим; при подаване на високо ниво на този извод, модулът деактивира серийната комуникация и влиза в режим на ниска консумация (sleep) докато отново не бъде подадено ниско ниво на извода |
| MICN/MICP | Диференциални входни изводи за свързване на микрофон |
| SPKN/SPKP | Диференциални изходни изводи за свързване на говорител |

Таблица 3.4. Предназначение на изводите на SIM800L

Тъй като в конкретното приложение не се използват възможностите на въпросния модул за изпращане и приемане на гласови данни и SMS съобщения, изводите, които имат отношение към тези услуги (изводи 7, 9-12), не са свързани. Изводът за активиране на sleep режима (извод 8, DTR) също не е свързан, тъй като при нормална работа модулът е постоянно свързан към уеб сървър и е в режим на изчакване и приемане на заявки от него.

Захранващото напрежение на интегралната схема SIM800 трябва да бъде в обхвата между 3.4V и 4.4V, като SIMCom съветват използването на 4.0V. По отношение на консумацията на ток, за връзка с GPRS мрежа е необходимо постоянно захранване от $\sim 450\text{mA}$. При изпращане на информация (т.нар. transmission burst), пиковата консумация може да достигне до 2A. Този пик налага използването на захранване, което може да осигури голям ток без значителен пад на напрежение. За да се подсигури нормалната работа при тези ситуации е силно препоръчително и добавянето на голям електролитен кондензатор близо до захранващите изводи на SIM800 модула. Развойната платка SIM800L е снабдена с вграден $470\mu\text{F}$ кондензатор с тази цел. В разработеното устройство, модулът се захранва от импулсен регулатор от блок "Захранване".

Към NET извода на модула е свързана GSM антена за постигане на по-голям обхват. Тя може да бъде изведена извън или в края на печатната платка на устройството, за да осигури запазване на свързаността при поставянето му в метална кутия.

При осъществяване на връзка между блоковете "Комуникация" и "Управление", е необходимо да се вземе под внимание следното: SIM800 модулът използва 2.8V вътрешна логика, докато ATMega328P микроконтролерът използва захранващото си напрежение като високо ниво за своите изводи (в случая 5V).

Входният информационен извод на серийната шина (RXD) на GPRS модула е свързан към изходния извод (TXD) на микроконтролера от блок "Управление" през резисторен делител на напрежение. Напрежението на

извода на GPRS модула при постъпване на високо ниво от микроконтролера е $\approx 2.78V$ (изчислено по формула (3.2.)). То не превишава максималното входно напрежение на модула ($V_{IH(MAX)} = 2.8V$), определено от документацията.

$$V_{out} = V_{in} * \frac{R2}{R1 + R2} \quad (3.2.)$$

V_{out} - Изходно напрежение на делителя (в случая извод RXD на SIM800L)

V_{in} - Входно напрежение на делителя (в случая маркер SIM_RX (извод TXD на ATMega328p))

$R1$ - Резистор между входа и изхода на делителя (в случая R3)

$R2$ - Резистор между изхода на делителя и маса (в случая R4)

$$V_{RXD} = 5V * \frac{20k\Omega}{16k\Omega + 20k\Omega} \approx 2.78V$$

Аналогичен е и случаят с нулиращия извод на модула (RST). Той също е свързан към микроконтролера през еквивалентен на разгледания резисторен делител.

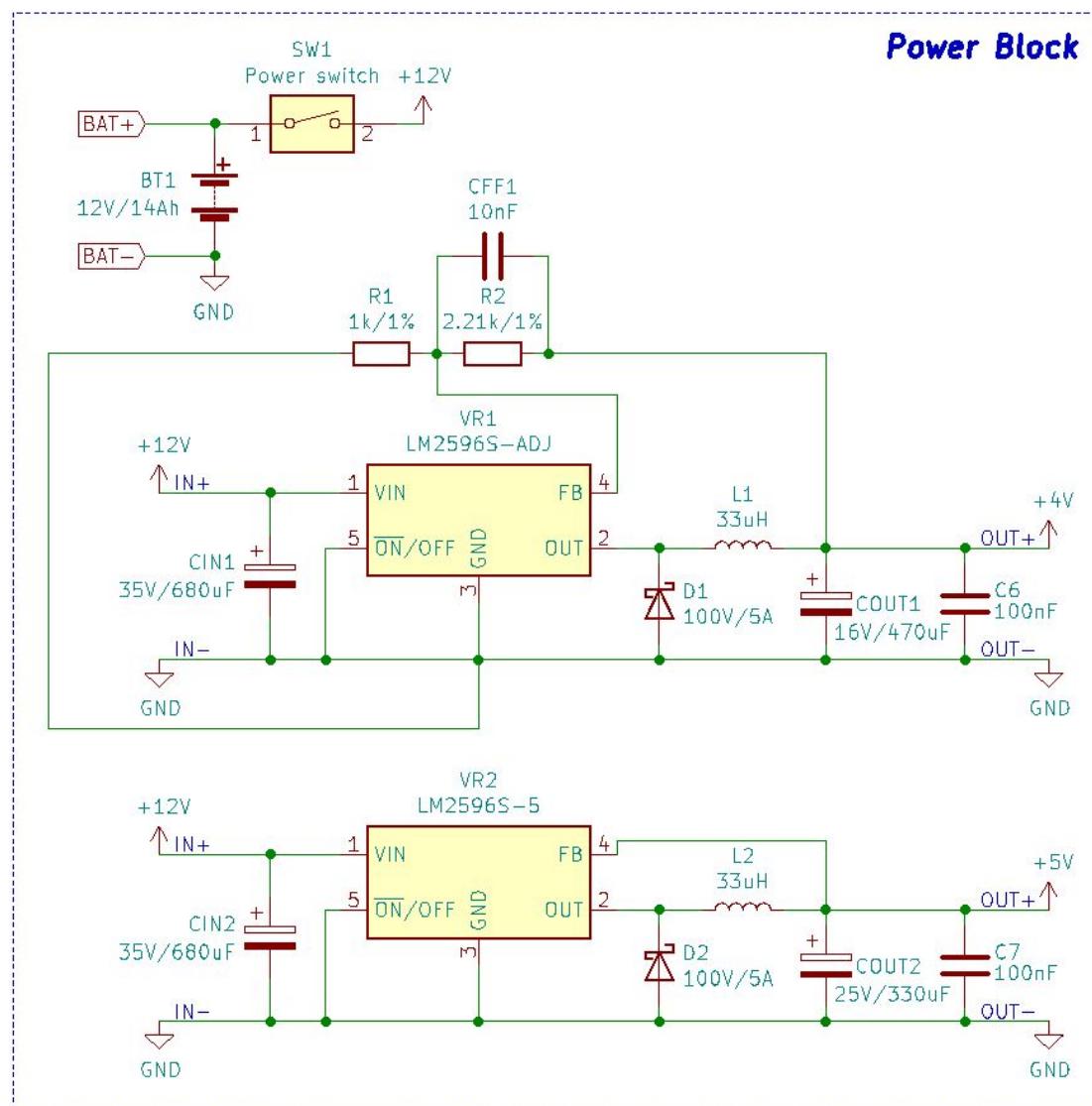
Изходният сериен извод (TXD) на GPRS модула е свързан директно към входния извод (RXD) на микроконтролера. Към него е добавен $10k\Omega$ токоограничаващ резистор (R6). Важно е да се отбележи, че изходното напрежение на извода на GPRS модула при установяване във високо ниво се равнява на $2.7-2.8V$. Това напрежение е по-ниско от минималното, нужно за надеждното отчитане на високо логическо ниво на входа на микроконтролера (според спецификацията на Atmel: $V_{IH(MIN)} = 0.6*V_{CC} = 3V$ за конкретното приложение). Въпреки това несъответствие, по време на функционалните тестове на устройството не бяха открити значителни грешки при предаването на информация от GPRS модула към микроконтролера. Запазването на тази схема на свързване намалява броя на нужните електронни елементи, което от своя страна намалява цената на крайния продукт. Въпреки това, като бъдеща цел за развитието на проекта може да се посочи подмяната на резисторните делители и директната свързаност с регулатори на логически нива, за да се подобри надеждността на устройството.

3.1.4. Принципна електрическа схема на блок “Захранване”

Използваният източник на информация е даден в [36]. Каталожна информация за използваните регулатори е приложена в {7}. Този блок може да бъде определен като най-комплексния за устройството по отношение на проектирани схеми. Той се състои от три основни части:

- **Акумулаторна батерия с прекъсвач;**
- **Схема с импулсен регулатор на напрежение LM2596S-ADJ за захранване на GPRS модула от блок “Комуникация”;**
- **Схема с импулсен регулатор на напрежение LM2596S-5 за захранване на светодиодната матрица от блок “Визуализация” и микроконтролера от блок “Управление”.**

Пълната принципна електрическа схема на този блок е показана във фиг. 3.8. Проектирането на схемите на всеки от отделните подблокове, както и изборът на подходящи елементи за тях, са обяснени в т. 3.1.4.1 - т. 3.1.4.3.



Фиг. 3.8. Принципна електрическа схема на блок "Захранване"

3.1.4.1. Акумулаторна батерия и прекъсвач

За захранване на цялото устройство е използвана 12V/14Ah SLA акумулаторна батерия. За поддържането на заряда ѝ се грижи MPPT контролерът от блок "Зареждане", към който е свързана батерията (маркери BAT- и BAT+ на схемата). Капацитетът на батерията (14Ah) е съобразен с голямата консумация на устройството и по-специално на светодиодната матрица. Изборът на запечатана оловно-киселинна батерия (SLA) е повлиян основно от ниската ѝ цена в сравнение с други по-качествени батерии с подобен капацитет. Въпреки това, в случай на реално внедряване на системата, по-удачният вариант за надеждността и живота на продукта би бил инвестиране в по-скъпи, но и по-качествени литиеви акумулаторни батерии.

Преди да се свърже с останалите елементи от схемата, положителният захранващ извод на батерията е свързан към обикновен ръчен превключвател (SW1), за да се осигури лесно включване и изключване на устройството. Превключвателят не прекъсва връзката между батерията и блок "Зареждане", т.е. дори при изключено устройство батерията продължава да се зарежда. По този начин тя остава заредена дори след дълъг период на престой без да бъде включвано устройството.

3.1.4.2. Схема с импулсен регулатор на напрежение LM2596S-ADJ за захранване на блок "Комуникация"

Изборът на импулсни регулатори вместо обикновени линейни стабилизатори за преобразуване на захранващото напрежение на устройството, е породен от две основни нужди:

- **Непрекъсната работа на системата** - импулсните регулатори са много по-ефективни при преобразуването на входни напрежения, които са доста по-големи от изходните - при обикновени условия тяхното КПД (коффициент на полезно действие) е >0.7 , докато при линейните стабилизатори ефективността зависи в много голяма степен от разликата между двете напрежения (в случая линеен стабилизатор би имал КПД $\approx 4V/12V = 0.33$). Повишаването на ефективността на преобразуването позволява използването на по-малки батерии и фотоволтаични панели, което намалява цената на крайния продукт;
- **Възможност за отдаване на голям захранващ ток** - подбранныте импулсни регулатори (LM2596) имат максимален постоянен изходен ток от 3А.

Предназначението на изводите на избрания регулатор е дадено в таблица 3.5.

| Извод - означение | Предназначение |
|------------------------------|--|
| VIN | Положителен захранващ извод на регулатора |
| GND | Отрицателен захранващ извод на регулатора (маса) |
| OUT | Изходен превключващ извод |
| FB | Извод за обратна връзка |
| ON/OFF | Разрешаващ/забраняващ извод |

Таблица 3.5. Предназначение на изводите на LM2596

При проектирането на схемата на регулатора е следвана описаната в документацията му процедура (**{7}**, стр. 22). Цели се постигането на изходно напрежение $\approx 4V$. Подборът на подходящи елементи е съобразен както с препоръката на производителя (TI - Texas Instruments), така и с наличността им на българския пазар. Извършени са няколко важни изчисления:

- **Изчисление на програмни резистори** (R1 и R2 на схемата) - по препоръка на TI, за R1 е подбран $1k\Omega / 1\%$ резистор. Стойността на R2 се изчислява по формула (3.3.). Избран е стандартен резистор със стойност, която максимално се доближава до изчислената - $2.21k\Omega / 1\%$. Използването на резистори с точност 1% осигурява по-голяма стабилност на изходното напрежение;

$$R2 = R1 * \left(\frac{V_{OUT}}{V_{REF}} - 1 \right) \quad (3.3.)$$

V_{OUT} - Изходно напрежение на регулатора

V_{REF} - Референтно напрежение (в случая $1.23V$)

$R1$ - $1k\Omega / 1\%$

$$R2 = 1k\Omega * \left(\frac{4V}{1.23V} - 1 \right) \approx 2.252k\Omega$$

- **Изчисление на волт-микросекунда (volt-microsecond) константата на индуктора** - използвана е формула (3.4.). Полученият резултат е използван при подбора на подходящ индуктор, като е взет под

внимание и максималният изходен ток на регулатора (3A). За избора е използвана таблицата, предоставена на стр. 19 от документацията на LM2596 ([7](#)).

$$E * T = (V_{IN} - V_{OUT} - V_{SAT}) * \left(\frac{V_{OUT} + V_D}{V_{IN} - V_{SAT} - V_D} \right) * \left(\frac{1000kHz}{150kHz} \right) \quad (3.4.)$$

V_{SAT} - Напрежение на насищане на вътрешния ключ (в случая 1.16V)

V_D - Пад на напрежение в права посока на диода (в случая 0.85V)

$$E * T = (12V - 4V - 1.16V) * \left(\frac{4V + 0.85V}{12V - 1.16V - 0.85V} \right) * \left(\frac{1000kHz}{150kHz} \right) \approx 22.14(V * \mu s)$$

Останалите елементи от схемата също са подбрани спрямо препоръките на TI:

- За електролитните кондензатори на входа и изхода на схемата (CIN и COUT) е предвидено достатъчно голямо работно напрежение (>1.5 пъти по-голямо от максималното напрежение на входа и изхода, съответно). Стойностите им са избрани спрямо документацията;
- Избраният диод е от тип Schottky - той има много добро време на превключване и нисък пад на напрежение в права посока. Неговото максимално обратно напрежение също е съобразено с напрежението на изхода на схемата;
- Кондензаторът CFF служи за внасяне на допълнителна стабилност към обратната връзка на регулатора.

Към изхода на схемата е добавен и още един филтриращ кондензатор (C6) със стойност 100nF. Той изпълнява ролята на развързващ кондензатор и служи за допълнителна защита на ИС от другите блокове от непредвидени флуктуации в захранването.

3.1.4.3. Схема с импулсен регулатор на напрежение LM2596S-5 за захранване на блокове "Управление" и "Визуализация"

Електрическата схема на този регулатор е почти напълно аналогична на предходната разгледана. Единствените съществени разлики са:

- Използван е регулатор с фиксирано изходно напрежение от 5V - LM2596S-5** (за разлика от програмируемото напрежение на LM2596S-ADJ);

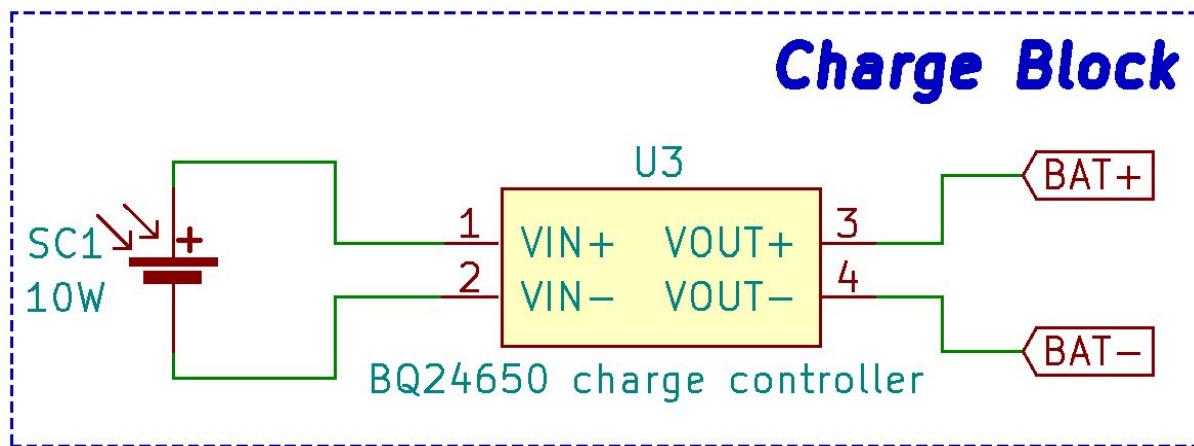
- Липсват двата резистора и единият кондензатор, които се използват за изграждане на обратна връзка - това е пряко следствие от фиксираното изходно напрежение на регулатора.

Подобно на другия импулсен регулатор от блока, към изхода на този регулатор също е добавен развързващ 100nF кондензатор (C7).

Въпреки че максималната консумация на светодиодната матрица (~4A) теоретично надвишава максималния изходен ток на регулатора (3A), в практическото приложение матрицата никога не използва повече от $\frac{1}{3}$ от максималния ток (т.е. 1A). Това е така, понеже при нито един от визуализираните пътни знаци не се използват повече от половината светодиоди в матрицата. В допълнение, не всички пиксели светят в бял цвят, което също предполага по-ниска консумация.

3.1.5. Принципна електрическа схема на блок “Зареждане”

Използваният източник на информация е даден в [37]. Каталожна информация за използваните елементи е приложена в {8} - {10}. Принципната електрическа схема на този блок е показана във фиг. 3.9. Той се състои от два основни елемента - фотоволтаичен панел и модул с MPPT контролер, които са разгледани в т. 3.1.5.1 и т. 3.1.5.2.



Фиг. 3.9. Принципна електрическа схема на блок “Зареждане”

3.1.5.1. Фотоволтаичен панел

Използваният в устройството соларен панел е LX-10M на фирмата LUXOR SOLAR. Неговата номинална мощност е 10W. Напрежението на максимална мощност U_{MPP} е 17.39V. Токът на максимална мощност I_{MPP} е 0.58V (значението на тези параметри е описано в т. 1.3.3.). Свързването на панела с MPPT контролера става посредством съединителната кутия на гърба му.

Причините за избора на този панел са основно финансови, тъй като по-мощните соларни панели са значително по-скъпи, а при изработването на прототип на устройството се цели основно доказване на работоспособността. Към подбрания MPPT контролер може без проблем да бъде свързан няколко пъти по-мощен фотоволтаичен панел без да се налага коригиране на схемата.

Процесът на избор на подходящ соларен панел зависи от множество фактори, като най-важните са свързани с физическото му разположение. При внедряване на подобни устройства, оптималното решение би било да се направи задълбочен анализ на избраната географска област, часовете слънчева светлина в продължение на цялата година, температурата на околната среда и т.н. По този начин цената на продукта може да бъде намалена максимално, като все пак се спази изискването за непрекъсната работа. В случай че такъв анализ не бъде направен, за системата може да бъде избран голям фотоволтаичен панел (например $>100W$), който да може за минимално време да зареди акумулаторната батерия. Така ще може да се осигури достатъчен заряд дори при продължителни периоди с много малко или почти никакво слънчево излъчване.

3.1.5.2. Модул с MPPT контролер

Избраният модул е базиран на ИС BQ24650 на Texas Instruments и е показан във фиг. 3.10.



Фиг. 3.10. Модул с MPPT контролер BQ24650

За настройка на модула се използва следната процедура (приложена в {9}):

- **Към входните изводи V_{IN+} и V_{IN-} се подава постоянно напрежение от лабораторно захранване** - то трябва да се равнява на напрежението на максимална мощност (U_{MPP}) на фотоволтаичния панел, което в случая е 17.39V;
- **Регулира се MPPT потенциометърът докато на изходните изводи V_{OUT+} и V_{OUT-} не се измери никакво напрежение** - така модулът се конфигурира да регулира работната точка по такъв начин, че входното напрежение винаги да се стреми към подаденото в момента на регулация (U_{MPP}). По този начин добитата от фотоволтаичния панел мощност ще е максимална (P_{MPP});
- **Регулира се изходното напрежение с втория потенциометър** - то зависи от вида на използваната батерия. Акумулаторната батерия на устройството (12V SLA) има препоръчано напрежение на зареждане 13.6V, затова изходното напрежение се регулира до достигане на тази стойност. По този начин се подсигурява, че модулът няма да надвиши това изходно напрежение и зареждането ще спре при достигането му, за да се предотврати презареждане.

Съществуват няколко вариации на въпросния MPPT модул, които се различават единствено по максималния изходен ток - 2A, 4A, 8A или 10A. Поради ниската мощност на използвания соларен панел, за устройството е използван модул с максимален изходен ток 2A. В случай на подмяна на фотоволтаика с по-мощен такъв, е необходима и замяната на модула съобразно с по-високия изходен ток.

3.2. Проектиране на пълната принципна електрическа схема на устройството

Във фиг. 3.11. е показана пълната принципна електрическа схема на микроконтролерна система за дистанционно управление на светодиоден пътен знак

Full schematic here

Четвърта глава

Проектиране на алгоритми и управляващ софтуер за
микроконтролерна система за дистанционно управление на
светодиоден пътен знак

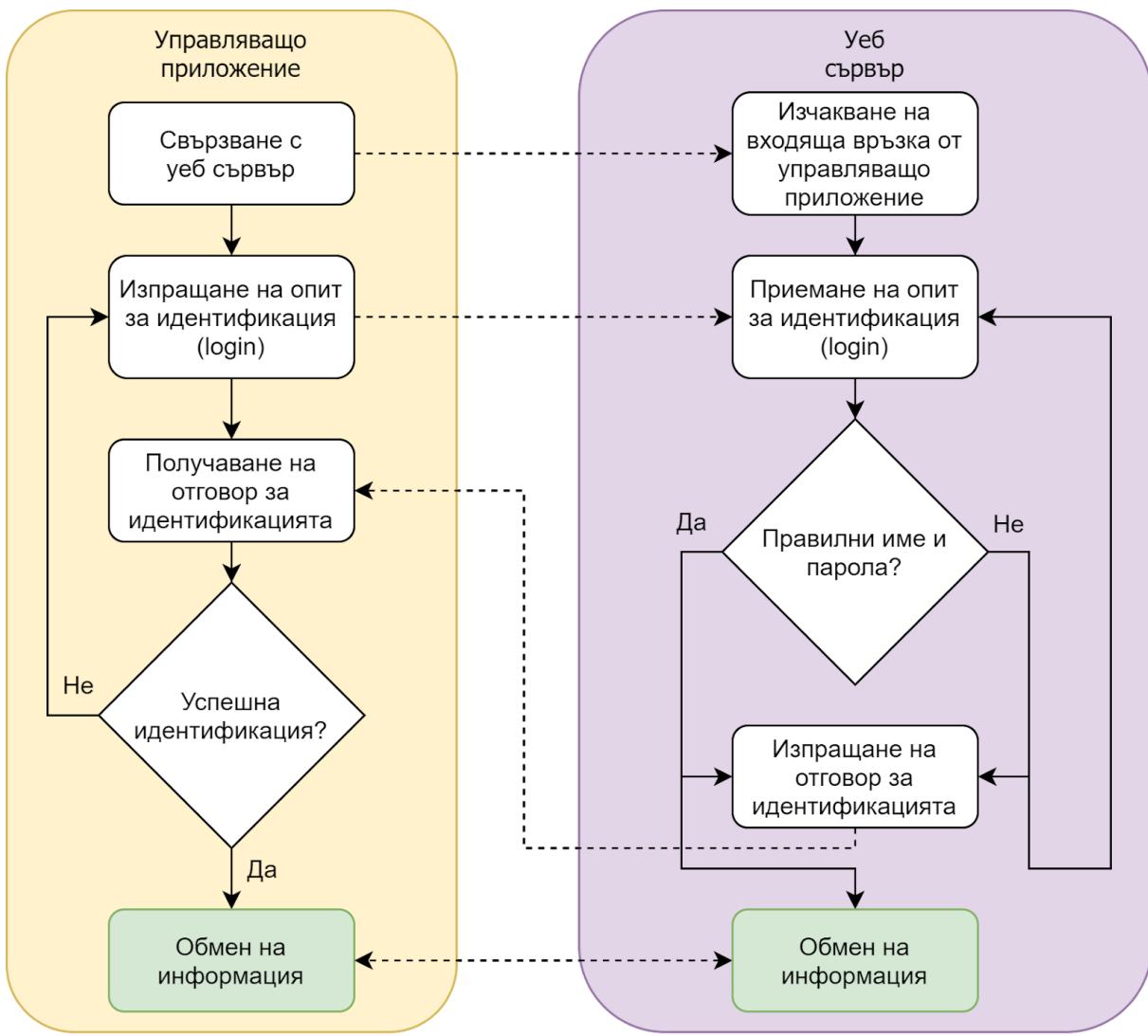
Целият програмен код на системата е приложен в **{11}**.

4.1. Проектиране на алгоритми за осъществяване на свързаност между различните софтуерни продукти

В т. 4.1.1. - т. 4.1.3 са разгледани блоковите схеми на алгоритъма на системата за осъществяване на свързаност и обмен на данни между управляващите приложения, уеб сървъра и пътните знаци. В т. 4.1.4. е разгледан персонализираният протокол, по който се извършва комуникацията между програмните продукти.

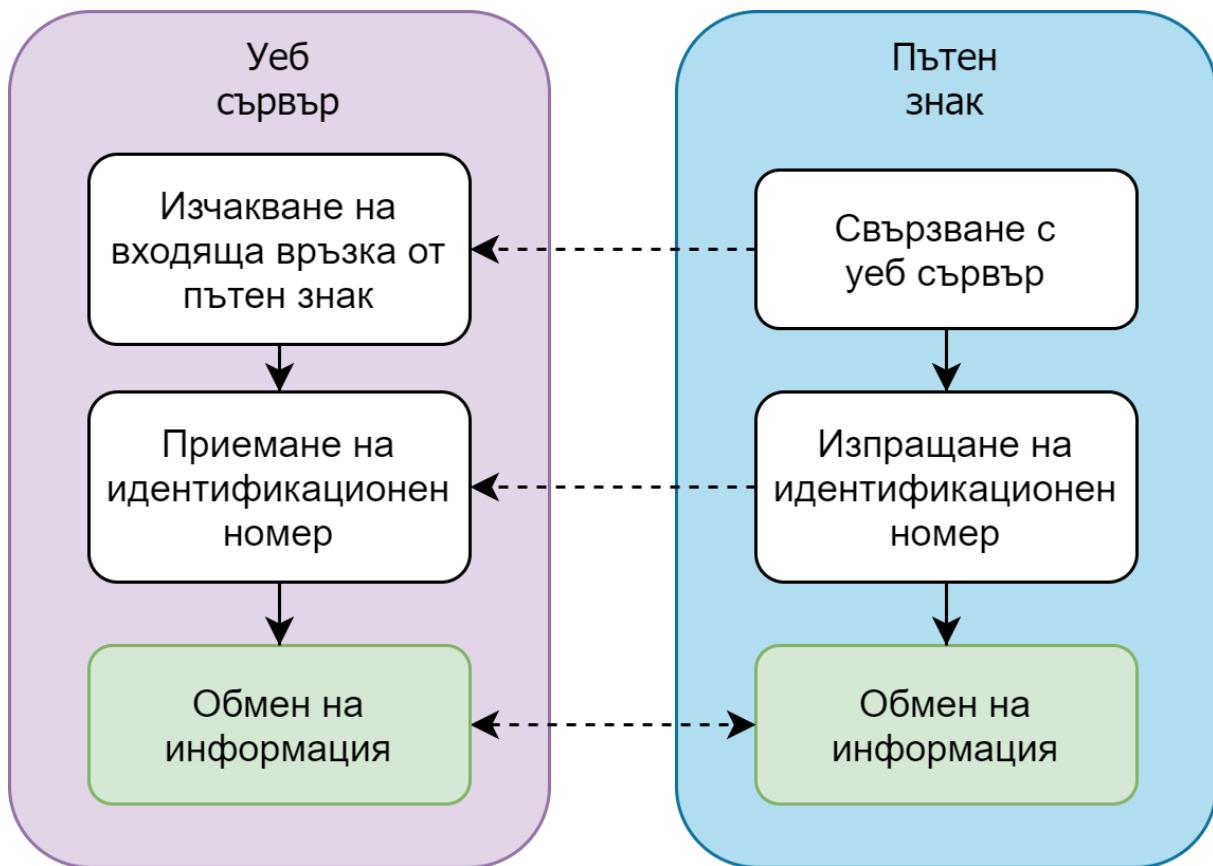
4.1.1. Осъществяване на връзка между управляващо приложение и уеб сървър

Във фиг. 4.1. е показана блоковата схема на алгоритъма за осъществяване на връзка между управляващо приложение и уеб сървър. Важна при този процес е идентификацията на потребителя, който се опитва да достъпи и да управлява наличните пътни знаци. В случай на успешна връзка и идентификация, между уеб сървъра и управляващото приложение се създава сесия за обмен на информация.



4.1.2. Осъществяване на връзка между уеб сървър и пътен знак

Във фиг. 4.2. е показана блоковата схема на алгоритъма за осъществяване на връзка между уеб сървър и пътен знак. Тази процедура е сравнително по-проста от предходната, тъй като при осъществяване на връзка към пътен знак не се налага идентификация на потребител. Единствената особеност на този процес, е че преди създаването на сесия за обмен на информация, свързаният пътен знак изпраща уникален номер, с който се идентифицира.



4.1.3. Процес на обмен на информация

Във фиг. 4.3. е показана блоковата схема на алгоритъма за обмен на информация между управляващите приложения, уеб сървъра и пътните знаци. Както се вижда на фигурата, управлението на пътните знаци от потребителя става посредством заявки към уеб сървъра. Уеб сървърът се грижи за обработване на тези заявки, като той има две функции:

- **Ако потребителят изпрати заявка за информация, която се намира на сървъра** - достъпва нужните ресурси и връща отговор с поисканата информация;
- **Ако потребителят изпрати заявка към определен пътен знак** - преобразува заявката в коректен формат и я препраща към пътния знак; получава отговор от пътния знак и го препраща към потребителя.

Connection block scheme here

4.1.4. Протокол за комуникация

Заявките, които обменят различните програмни продукти на системата, са дефинирани в персонализиран протокол. Той се състои от два основни типа заявки:

- SET заявки - дефинират изпращане на информация
- GET заявки - дефинират поискване на информация

Отговорите на заявките зависят от вида и резултата от изпълнението им.

Възможните заявки, които управляващото приложение може да изпрати към уеб сървъра, са показани във фиг. 4.4.

```
"GET ":
  1) "devices"
    {response}:
      [message 0]: <Number of available devices>
      [message 1-n]: <Device IMEI>
  2) "details <IMEI>"
    {response}:
      [message 0]: <Device details string>

"SET <IMEI> ":
  1) "speed <speed limit>"
    {response}:
      [message 0]: <Confirmation message>
  2) "warning <warning sign>"
    {response}:
      [message 0]: <Confirmation message>
```

Фиг. 4.4. Заявки от управляващо приложение към уеб сървър

Описание на заявките:

- “**GET devices**” - поиска от уеб сървъра информация за всички пътни знаци, които са свързани към него
 - Отговор - няколко на брой съобщения, като първото е броят на наличните устройства, а всяко следващо - уникален идентификатор на съответния пътен знак (до изчерпване на устройствата в списъка)
- “**GET details <IMEI>**” - поиска от уеб сървъра детайлна информация за състоянието на дадено устройство. Тази заявка се препраща към пътния знак със съответния уникален идентификатор <IMEI>

- Отговор - детайлна информация за пътния знак (може да бъде и съобщение за грешка)
- “**SET <IMEI> speed <speed limit>**” - изпраща заявка за изобразяване на ограничение на скоростта върху даден пътен знак. Тази заявка се препраща към пътния знак със съответния уникален идентификатор **<IMEI>**, като стойността на ограничението се определя от **<speed limit>** аргумента
 - Отговор - потвърждение в случай на коректно прието съобщение от пътния знак (може да бъде и съобщение за грешка)
- “**SET <IMEI> warning <warning sign>**” - изпраща заявка за изобразяване на предупредителен знак върху даден пътен знак. Тази заявка се препраща към пътния знак със съответния уникален идентификатор **<IMEI>**, като видът на предупреждението се определя от **<warning sign>** аргумента
 - Отговор - потвърждение в случай на коректно прието съобщение от пътния знак (може да бъде и съобщение за грешка)

Възможните заявки, които уеб сървърът може да изпрати към определен пътен знак, са разгледани във фиг. 4.5.

```

"GET ":
  1) "dtl" = device details
    {response}:
      [message 0]: <Device details string>

"SET ":
  1) "spd <speed limit>"
    {response}:
      [message 0]: <Confirmation message>

  2) "wrn <warning sign code>"
    {response}:
      [message 0]: <Confirmation message>

```

Фиг. 4.5. Заявки от уеб сървър към пътен знак

Описание на заявките:

- “**GET dtl**” - поиска от устройството информация относно изобразения в момента пътен знак. Тази заявка се изпраща при получаване на “**GET details <IMEI>**” заявка от управляващо приложение

- Отговор - информация за състоянието на пътния знак
- “**SET spd <speed limit>**” - изпраща заявка за изобразяване на ограничение на скоростта. Тази заявка се изпраща при получаване на “*SET <IMEI> speed <speed limit>*” заявка от управляващо приложение
 - Отговор - потвърждение в случай на коректно получена заявка.
- “**SET wrn <warning sign code>**” - изпраща заявка за изобразяване на предупредителен знак. Тази заявка се изпраща при получаване на “*SET <IMEI> warning <warning sign>*” заявка от управляващо приложение
 - Отговор - потвърждение в случай на коректно получена заявка.

4.2. Проектиране на софтуер за микроконтролерно устройство

Управляващият софтуер на микроконтролерното устройство е отговорен за:

- Изграждане и поддържане на сесия с уеб сървъра;
- Обработване на заявки, получени от уеб сървъра;
- Изпращане на отговори и потвърждения на получени заявки;
- Визуализиране на коректния пътен знак.

4.2.1. Използвани инструменти за разработка

При проектирането на управляващия софтуер за микроконтролерното устройство са използвани следните инструменти:

- **Интегрирана среда за разработка Arduino IDE** - на използванятия за устройството микроконтролер (ATMega328P) е качен Arduino bootloader с цел улесняване на функционалните тестове. Интегрираната среда за разработка на Arduino предлага лесен метод за качване на код на микроконтролера;
- **Arduino C/C++ език за програмиране;**
- **Инструмент за автоматично форматиране на код** - за оформянето на програмния код е използван вграденият в интегрираната среда за разработка Visual Studio Code инструмент.

4.2.2. Използвани библиотеки

Документацията на използваните библиотеки е приложена в **{12}** и **{13}**. Управляващият софтуер използва следните библиотеки:

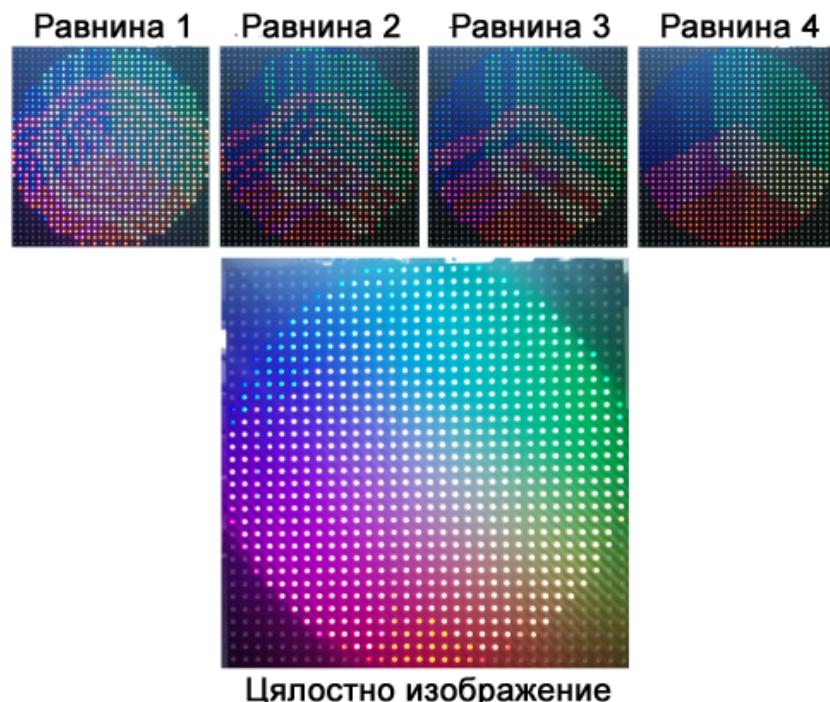
- **<TinyGsmClient.h>** - служи за лесно управление на GSM/GPRS модула. Дефинира функции за инициализация на модула, свързване към GPRS

мрежата на доставчика, свързване към уеб сървър, изпращане и четене на информация от него, поддържане на сесия и др.;

- **<RGBmatrixPanel.h>** - служи за лесно управление на светодиодната матрица. Дефинира функции за инициализация на матрицата, както и за писане на форми и символи в различни цветове върху нея.

При разработката на устройството се наложи преработване на **<RGBmatrixPanel.h>** библиотеката поради недостатъчната динамична памет на избрания микроконтролер. За да се разгледат внесените промени, е необходимо първо да се обясни принципът на работа на библиотеката.

Библиотеката използва 4-битова ДКМ (т. 1.1.3.), за да увеличи максималния брой цветове, които могат да бъдат изобразявани. За да се осигури бързодействие, в паметта на микроконтролера се създава буфер. В него се запазват определен брой състояния на матрицата, равняващи се на резолюцията на модулацията. В случая на използваната матрица, в буфера се запазват 4 "равнини", всяка от които пази състоянието на матрицата, характерно за всеки от 16-те времеви отрезъка (първата равнина ще бъде изобразявана в продължение на 1 времеви отрезък, втората - в продължение на 2 и т.н.). Ако бъдат разгледани поотделно, всяка от равнините съдържа 8-цветно изображение на матрицата. При достатъчно бързото им редуване (чрез ДКМ), различните коефициенти на запълване на всеки отделен светодиод създават илюзията за различни цветове. Равнините, както и резултатът от комбинирането им, са изобразени във фиг. 4.6.



Фиг. 4.6. Равнини, изобразявани през различните времеви отрезъци + цялостно изображение, получено чрез ДКМ

Недостатък на този подход е голямото количество памет, което се използва за запаметяване на всички състояния на матрицата. За всяка равнина, състоянието на един пиксел може да бъде запазено в 3 бита (R,G,B). Това означава, че за запазване на 4 състояния са нужни общо $4 \times 3 = 12$ бита или 1,5 байта. Използвайки тази информация може да бъде пресметнат общий размер на буфера, необходим за цялата матрица чрез формула (4.1.).

$$\text{Размер на буфера} = \text{Брой пиксели} * \text{Памет за 1 пиксел} \quad (4.1.)$$

$$\text{Размер на буфера} = (32 * 32) * (1,5 \text{ байта}) = 1536 \text{ байта}$$

Тези 1536 байта, заети от буфера на библиотеката, заемат $>75\%$ от общата динамична памет на микроконтролера. Оставащите <500 байта са крайно недостатъчни за надеждна реализация на останалата функционалност на устройството - управление на GSM модул през серийна връзка, комуникация с уеб сървър и т.н.

Най-удачното решение на този проблем е чрез намаляване на размера на буфера. Това става чрез редуциране на броя равнини, които се записват в паметта на микроконтролера. Това намалява и броя налични цветове. За изобразяване на пътни знаци са напълно достатъчни 8-те цвята, които могат да бъдат запазени в една единствена равнина. Поради тази причина внесените корекции в библиотеката премахват функционалността, която е свързана с изобразяване на множество равнини. Основните редакции са във файла *RGBMatrixPanel.cpp* и включват:

- **Намаляване на броя равнини, с които работи библиотеката** (фиг. 4.7.) - стойността на константата е променена на 2 поради проблем, който възниква при въвеждане на 1. Въпреки това в паметта се пази информация само за едно състояние на матрицата;

```
93 // #define nPlanes 4  
94 #define nPlanes 2
```

Фиг. 4.7. Намаляване на броя равнини

- **Създаване на буфер в статичната памет, който заменя динамичното заделяне в библиотеката** (фиг. 4.8.) - по този начин може по-лесно да се следи заетата от библиотеката памет и да се идентифицират грешки от препълване на стека;

[Забележка: минималният размер на необходимия буфер е $(32*32*3/8) = 384$ байта, но оптимизирането му е свързано с пълно реконструиране на логиката на библиотеката. Затова за улеснение е използван по-голям буфер, който побира $(32*32*1/2) = 512$ байта]

```
56     uint8_t          buff[512];
134    // if(NULL == (matrixbuff[0] = (uint8_t *)malloc(allocsize))) return;
135    matrixbuff[0] = buff;
```

Фиг. 4.8. Дефиниране на буфер в статичната памет

- **Промяна на местата, на които се записват състоянията на всеки светодиод** (фиг. 4.9.) - това се налага поради по-малкия размер на новия буфер. С внесените корекции състоянията на всички светодиоди се запазват в старшите 6 бита на всеки байт от буфера;

```
579    // ptr[WIDTH*2] &= ~B00000011;           // Plane 0 R,G mask out in one op
580    // if(r & 1) ptr[WIDTH*2] |= B00000001; // Plane 0 R: 64 bytes ahead, bit 0
581    // if(g & 1) ptr[WIDTH*2] |= B00000010; // Plane 0 G: 64 bytes ahead, bit 1
582    // if(b & 1) ptr[WIDTH]   |= B00000001; // Plane 0 B: 32 bytes ahead, bit 0
583    // else      ptr[WIDTH]   &= ~B00000001; // Plane 0 B unset; mask out
584
585    *ptr &= ~B00011100;           // Mask out R,G,B in one op
586    if (r & 1) *ptr |= B00000100; // Plane N R: bit 2
587    if (g & 1) *ptr |= B00001000; // Plane N G: bit 3
588    if (b & 1) *ptr |= B00010000; // Plane N B: bit 4
589
604    // *ptr &= ~B00000011;           // Plane 0 G,B mask out in one op
605    // if(r & 1) ptr[WIDTH] |= B00000010; // Plane 0 R: 32 bytes ahead, bit 1
606    // else      ptr[WIDTH] &= ~B00000010; // Plane 0 R unset; mask out
607    // if(g & 1) *ptr      |= B00000001; // Plane 0 G: bit 0
608    // if(b & 1) *ptr      |= B00000010; // Plane 0 B: bit 0
609
610    *ptr &= ~B11100000;           // Mask out R,G,B in one op
611    if (r & 1) *ptr |= B00100000; // Plane N R: bit 5
612    if (g & 1) *ptr |= B01000000; // Plane N G: bit 6
613    if (b & 1) *ptr |= B10000000; // Plane N B: bit 7
```

Фиг. 4.9. Записване на състоянията на светодиодите на нови места в буфера поради по-малкия му размер

- Корекция на функциите за запълване на буфера предвид новия му размер (фиг. 4.10.);

```

630  // memset(matrixbuff[backindex], c, WIDTH * nRows * 3);
631  memset(matrixbuff[backindex], c, WIDTH * nRows);
654  // memcpy(matrixbuff[backindex], matrixbuff[1-backindex], WIDTH * nRows *
655  // 3);
656  memcpy(matrixbuff[backindex], matrixbuff[1 - backindex], WIDTH * nRows);

```

Фиг. 4.10. Корекция на дължината на запълваното пространство

- Премахване на логиката за изобразяване на информацията в последните 2 бита на всеки байт от буфера (фиг. 4.11.) - това се прави, понеже с внесените корекции на тези места вече не се запазва информация за никои от светодиодите.

```

964  // for(i=0; i<WIDTH; i++) {
965  //   DATAPORT =
966  //     ( ptr[i]          << 6)
967  //     ((ptr[i+WIDTH]    << 4) & 0x30) |
968  //     ((ptr[i+WIDTH*2] << 2) & 0x0C);
969  //   CLKPORT = tick; // Clock lo
970  //   CLKPORT = tock; // Clock hi
971  // }

```

Фиг. 4.11. Премахване на логиката за изобразяване на последните 2 бита на всеки байт

С тези корекции, библиотеката може успешно да изобразява 8-те цвята, използвани при визуализацията на пътните знаци, като същевременно заема едва $\frac{1}{3}$ от паметта, нужна на оригиналната библиотека.

4.2.3. Блокова схема на алгоритъма

Във фиг. 4.12. е показана блоковата схема на алгоритъма на управляващия софтуер за микроконтролерното устройство

Microcontroller block scheme

4.2.4. Функции и променливи на програмата

Целият програмен код на микроконтролера се намира във файла */TrafficSignCode/MicrocontrollerCode/MicrocontrollerCode.ino*. Нужните за работата на устройството глобални променливи са дефинирани в началото на файла и са показани във фиг. 4.13.

```
33  char command[4]; // Last received command
34  char request[4]; // Last received request
35  char value[4]; // Last received value
36  int index = 0; // Index (Used for loops)
37  char current; // Current char (Used for reading from the serial port)
38
39  char currentState[8] = {"unk unk"}; // Current device state (Default)
40
41 // LED matrix object (RGBmatrixPanel.h library)
42 RGBmatrixPanel matrix(A, B, C, D, CLK, LAT, OE, false);
43 // GPRS modem object (TinyGsmClient.h library)
44 TinyGsm modem(SerialAT);
45 // GPRS connection client object (TinyGsmClient.h library)
46 TinyGsmClient client(modem);
```

Фиг. 4.13. Глобални променливи

Предназначенията на дефинираните функции са:

- **void setup()** - Първата функция, която се изпълнява при включване на устройството:
 - Инициализира необходимите изводи на GSM модула;
 - Извършва опит за свързване към уеб сървъра;
 - В случай, че връзката е неуспешна - рестартира модула и извършва повторен опит за свързване;
 - Инициализира светодиодната матрица.
- **void loop()** - Основният цикъл на програмата. Повтаря се докато устройството не бъде изключено:
 - Изчаква, прочита и изпълнява получените от уеб сървъра заявки;
 - Ако връзката към сървъра бъде прекъсната - рестартира GSM модула и прави повторен опит за свързване.
- **void InitModulePins()** - Инициализира серийната комуникация и RST извода на GSM модула;
- **bool RunConnectionProcedure()** - Изпълнява всички функции, нужни за изграждане на връзка към уеб сървъра. Връща "true" ако сесията е изградена успешно и "false" ако е възникнал проблем;
- **void InitModem()** - Инициализира GSM модула;

- **bool ConnectToAPN()** - Свързва GSM модула към GPRS мрежата на интернет доставчика. Връща “true” ако връзката е изградена успешно и “false” ако е възникнала грешка;
- **bool ConnectToServer()** - Свързва GSM модула към уеб сървъра. Връща “true” ако връзката е успешна и “false” ако е възникнала грешка;
- **void SendIMEI()** - Изпраща уникалния идентификатор на устройството на уеб сървъра. За идентификация се използва т.нар. IMEI (International Mobile Equipment Identity) номер, който е уникален за всяко мобилно устройство. Той е избран вместо IMSI (International Mobile Subscriber Identity) номерът, тъй като не се променя при подмяна на SIM картата;
- **void RestartModule()** - Извършва хардуерен рестарт на GSM модула чрез неговия RST извод;
- **void InitMatrix()** - Инициализира светодиодната матрица и изписва информационно съобщение върху нея (“Wait for data...”);
- **void ReadRequest()** - Основната функция, която приема и обработва заявки от уеб сървъра. Процедурата по четене и записване на всеки от елементите на заявката е показана във фиг. 4.14. Тъй като заявките винаги се състоят от един или повече 3-буквени кода, прочитането на останалите елементи на заявката е идентично;

```

197 // Read the first character
198 current = (char)client.read();
199 // Keep reading characters until whitespace or newline is received
200 while (current != '\n' && current != ' ') {
201     // Store the character
202     command[index] = current;
203     // Increment the index
204     index++;
205     // If the command was longer than 3 chars (This shouldn't happen):
206     if (index > 3) {
207         // Reset the index to prevent out of bounds write
208         index = 0;
209     }
210     // Read another character
211     current = (char)client.read();
212 }
213
214 // Set the last character to a terminating zero
215 // (required for proper storage of char array)
216 command[index] = '\0';

```

Фиг. 4.14. Прочитане и записване на елемент от заявката

- **void HandleSet()** - Обработва получени “SET” заявки, като преди това прочита стойността им (3-тия аргумент от заявката). При успешно получена и изпълнена заявка - актуализира запазеното състояние на

устройството и изпраща потвърждение към уеб сървъра. Тялото на тази функция е показано във фиг. 4.15.

- При получаване на “SET spd <speed limit>” заявка - превръща стойността в число; визуализира съответното ограничение;
- При получаване на “SET wrn <warning sign code>” заявка - визуализира съответния предупредителен знак.

```
273 // Handles a SET command received from the server
274 void HandleSet() {
275     // Read the value of the command
276     ReadValue();
277
278     // If the request is "spd" (Speed limit sign):
279     if (strcmp(request, "spd") == 0) {
280         // Convert the received value to int for easier calculations later
281         int limit = ConvertSpeed();
282         // Visualize the speed limit traffic sign with the appropriate value
283         VisualizeSpeedLimit(limit);
284     }
285     // If the request is "wrn" (Warning sign):
286     else if (strcmp(request, "wrn") == 0) {
287         // Visualize the appropriate warning traffic sign
288         VisualizeWarning();
289     }
290     // If something went wrong:
291     else {
292         // Do nothing
293         return;
294     }
295
296     // Update the currently stored device state
297     ChangeCurrentState();
298     // Send confirmation to the web server
299     client.print('k');
300 }
```

Фиг. 4.15. Обработване на получени SET заявки

- **void HandleGet()** - Обработва получени “GET” заявки. Изпраща поисканата информация на уеб сървъра. Тялото на тази функция е показано във фиг. 4.16.
 - При получаване на “GET dtl” заявка - изпраща запазеното състояние на устройството на уеб сървъра.

```

302 // Handles a GET command received from the server
303 void HandleGet() {
304     // If the request is "dtl" (Device details):
305     if (strcmp(request, "dtl") == 0) {
306         // Send the current device details to the web server
307         client.print(currentState);
308     }
309     // If something went wrong:
310     else {
311         // Do nothing
312         return;
313     }
314 }
```

Фиг. 4.16. Обработване на получени GET заявки

- **void ReadValue()** - Прочита и записва стойността на “SET” заявка (3-тия аргумент);
- **int ConvertSpeed()** - Превръща стойността на “SET” заявка (3-тия аргумент) от char[] в int. Използва се за конвертиране на ограничението на скоростта при получаване на “SET spd <speed limit>” заявка;
- **void ChangeCurrentState()** - Актуализира променливата, която пази текущото състояние на устройството;
- **void VisualizeSpeedLimit(int speedLimit)** - Изобразява пътен знак за ограничение на скоростта със стойност *speedLimit*;
- **void VisualizeWarning()** - Визуализира определен предупредителен пътен знак според заявката, получена от уеб сървъра;
- **void VisualizeStopSignWarning()** - Изобразява пътен знак “СТОП” върху LED матрицата;
- **void VisualizeGeneralWarning()** - Изобразява пътен знак “Внимание!” върху LED матрицата;
- **void VisualizeTrafficLightWarning()** - Изобразява пътен знак “Светофар” върху LED матрицата;
- **void VisualizeNoEntryWarning()** - Изобразява пътен знак “Забранено влизането” върху LED матрицата;
- **void VisualizeForwardOnlyWarning()** - Изобразява пътен знак “Движение само напред след знака” върху LED матрицата;
- **void VisualizeRightOnlyWarning()** - Изобразява пътен знак “Движение само надясно след знака” върху LED матрицата;
- **void VisualizeLeftOnlyWarning()** - Изобразява пътен знак “Движение само наляво след знака” върху LED матрицата.

4.3. Проектиране на софтуер за управляващ уеб сървър

Уеб сървърт е отговорен за:

- Изграждане и поддържане на сесии с управляващите приложения;
- Изграждане и поддържане на сесии с пътните знаци;
- Получаване и обработване на заявки от управляващите приложения;
- Изпращане на заявки и получаване на отговори от пътните знаци;

4.3.1. Използвани инструменти за разработка

При проектирането на софтуера за управляващия уеб сървър бяха използвани следните инструменти:

- **Python език за програмиране** - избран поради неговата гъвкавост и наличие на множество библиотеки, подходящи за изпълнението на поставените към софтуерния продукт изисквания;
- **Интегрирана среда за разработка Visual Studio Code**;
- **Black** - инструмент за автоматично форматиране на python код;
- **Amazon Web Services** - платформа за облачни услуги. Използвана е за хостинг и присвояване на статичен публичен IPv4 адрес на сървъра;
- **OpenSSL** - инструмент за създаване на SSL/TLS (Secure Sockets Layer/Transport Layer Security) сертификати и ключове. Използван е за създаване на self-signed сертификата, който се използва за защита на комуникацията между управляващите приложения и уеб сървъра.

4.3.2. Използвани библиотеки

Документацията на използваните библиотеки е приложена в {14} - {21}. Работата на сървъра зависи от следните библиотеки:

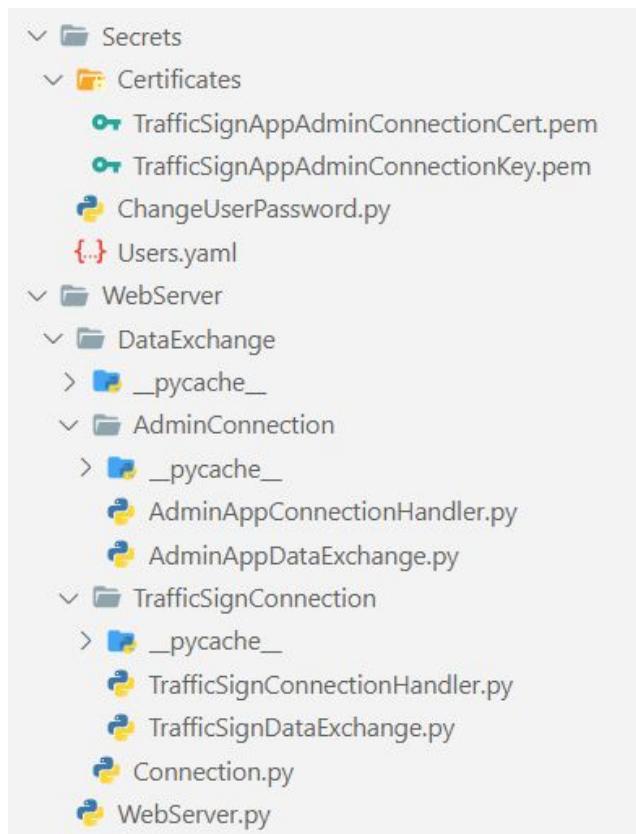
- **socket** - мрежов интерфейс от ниско ниво. Служи за изграждане на между процесна комуникация (Inter-process communication (IPC)). Позволява създаването на socket обекти, които имат свой IP адрес и порт и могат да бъдат използвани като сървър или като клиент:
 - socket сървър - приема входящи връзки на съответния IP адрес и порт;
 - socket клиент - осъществява изходяща връзка към съответния IP адрес и порт.
- **ssl** - предоставя метод за въвеждане на SSL/TLS за socket обектите. SSL/TLS протоколите предлагат криптиране на информация и удостоверяване на източника чрез т. нар. сертификати. Сертификатът

съдържа публичен ключ и доказателство за самоличността на организацията, на която той принадлежи;

- **yaml** - предоставя методи за преобразуване и записване на python обекти в .yaml файлове и обратно;
- **hashlib** - предоставя методи за изпълнение на хеширащи алгоритми върху дадена информация;
- **select** - използва се за избягване на безкрайното блокиращо изчакване за получаване на информация на даден socket. Позволява въвеждането на максимално време за изчакване, след което изпълнението на програмата продължава;
- **struct** - позволява конвертирането на Python стойности в стандартни C структури, представени като Python bytes обекти и обратно;
- **threading** - позволява създаването на допълнителни процесни нишки, които се изпълняват паралелно в програмата;
- **os** - предоставя методи за използване на специфична за операционната система функционалност.

4.3.3. Структура на хранилището на уеб сървъра

Във фиг. 4.17. е показана структурата на файловата система на уеб сървъра.



Фиг. 4.17. Файловата система на уеб сървъра

Предназначението на всички папки и файлове (разгледани от горе надолу) е следното:

- **Secrets/** - съдържа всички файлове, които имат отношение към някакъв вид секретна информация. При внедряване на системата тази папка трябва да бъде изключена от системата за контрол на версии и достъп до нея трябва да бъде даден само на администратори, които имат достъп до виртуалната машина на уеб сървъра;
- **Certificates/** - съдържа създадените за системата SSL сертификат и частен ключ, които се използват за защита на връзката между уеб сървъра и управляващите приложения;
- **TrafficSignAppAdminConnectionCert.pem** - SSL сертификат;
- **TrafficSignAppAdminConnectionKey.pem** - частен ключ;
- **ChangeUserPassword.py** - прост скрипт, който се използва за промяна на паролите на потребителските акаунти, които имат достъп до системата. Може да бъде използван и за добавяне на нови акаунти към системата;
- **Users.yaml** - съдържа информация за всички потребители с достъп до системата, като за всеки потребител пази хеширана парола и "salt" (случайно число, използвано при генерирането на хеша). Информацията в този файл се използва от уеб сървъра за проверка на получените потребителски идентификационни данни;
- **WebServer/** - съдържа всички файлове, нужни за работата на уеб сървъра;
- **DataExchange/** - съдържа всички файлове, които отговарят за обмена на информация между уеб сървъра и другите програмни продукти на системата;
- **AdminConnection/** - съдържа всички файлове, които отговарят за връзката между уеб сървъра и управляващите приложения;
- **AdminAppConnectionHandler.py** - отговаря за създаването и удостоверяването на връзки с управляващите приложения;
- **AdminAppDataExchange.py** - отговаря за обмена на данни между уеб сървъра и управляващите приложения;
- **TrafficSignConnection/** - съдържа всички файлове, които отговарят за връзката между уеб сървъра и пътните знаци;
- **TrafficSignConnectionHandler.py** - отговаря за създаването на връзки с пътните знаци;
- **TrafficSignDataExchange.py** - отговаря за обмена на данни между уеб сървъра и пътните знаци;
- **Connection.py** - служи за форматиране, изпращане и получаване на съобщения от различните връзки;
- **WebServer.py** - съдържа основната логика на приложението. Изпълнението на този файл стартира уеб сървъра.

4.3.4. Блокова схема на алгоритъма

Във фиг. 4.18. е показана блоковата схема на алгоритъма на управляващия уеб сървър.

Web server block scheme

4.3.5. Класове, методи и променливи на програмата

Промяната на паролите на потребителските акаунти и добавянето на нови потребители става с помощта на скрипа *ChangeUserPassword.py*. Той зарежда съществуващите потребители от файла *Users.yaml*, позволява на администратора да въведе нова информация и накрая записва новите данни отново в същия файл. Процедурата по генериране на хеш на въведена парола е показана във фиг. 4.19.

```
32     # Read the new password for the user
33     newPassword = input("New password: ")
34
35     # Generate a truly random 32 byte salt
36     salt = os.urandom(32)
37
38     # Generate the password hash using pbkdf2
39     passwordHash = hashlib.pbkdf2_hmac(
40         "sha512", # Hash algorithm
41         newPassword.encode("utf-8"), # Convert the password to bytes
42         salt, # Salt
43         100000, # Iterations of the hash algorithm
44     )
45
46     # Store the user's new salt and password hash
47     users[username] = {"salt": salt, "password": passwordHash}
```

Фиг. 4.19. Хеширане и записване на парола

Тъй като бързодействието не е от особено значение, а сигурността е критична за конкретното приложение, избраният хеширащ алгоритъм е SHA-512, който генерира 64-байтов хеш. "salt" представлява 32-байтово число, получено чрез функцията на операционната система за генериране на случаини числа, подходящи за криптографска употреба (`os.urandom(size)`). То се добавя към паролата и се използва за подсигуряване, че две еднакви пароли на различни потребители ще имат различни хешове. При създаването на хеш, хеширащият алгоритъм SHA-512 се изпълнява последователно 100 000 пъти, за да се забави процесът и да се затрудният бъдещи опити за хакерски атаки.

За по-лесна навигация и по-подредена структура на програмния код на уеб сървъра, всички класове са изведени в отделни файлове. Предназначенията на всички класове и методи са както следва:

- **class AdminAppConnectionHandler** - служи за: създаване на socket сървър, към който управляващите приложения се свързват; установяване на връзки; удостоверяване на свързаните потребители; създаване на процеси за обработване на заявките на всеки свързан клиент

- **def __init__(self)** - конструктор методът на класа. Създава socket сървър, който изчаква входящи връзки (фиг. 4.20.);

```

15  # Constructor method
16  def __init__(self):
17      super().__init__()
18      self.address = "0.0.0.0" # IPv4 address used for the web server socket (listen on all addresses)
19      self.port = 26418 # Admin application port
20
21      # Create an unwrapped IPv4 TCP socket object
22      self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
23      # Allow the server to bind to an address which is in a TIME_WAIT state
24      self.socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
25      # Bind the socket to the given address and port
26      self.socket.bind((self.address, self.port))
27      # Enable the server to accept connections
28      self.socket.listen()
29
30      self.users = {} # Known users
31      # Load all known users
32      self.LoadUsers()

```

Фиг. 4.20. Създаване на socket сървър

- **def LoadUsers(self)** - зарежда списъка с потребители от *Users.yaml* файла;
- **def WaitForConnection(self)** - приема входяща връзка от управляващо приложение и добавя SSL/TLS към socket обекта, отговорен за комуникацията (фиг. 4.21.);

```

51  # Accept the connection
52  adminSocket, adminAddress = self.socket.accept()
53  # Create an SSLContext object
54  context = ssl.SSLContext(
55      ssl.PROTOCOL_TLS_SERVER # Server side connection (auto-negotiate highest supported SSL/TLS version)
56  )
57  # Load the certificate and private key used for the admin application connection
58  context.load_cert_chain(
59      certfile="../Secrets/Certificates/TrafficSignAppAdminConnectionCert.pem",
60      keyfile="../Secrets/Certificates/TrafficSignAppAdminConnectionKey.pem",
61  )
62  # Create a protected SSL/TLS TCP socket (server side)
63  adminSocket = context.wrap_socket(adminSocket, server_side=True)

```

Фиг. 4.21. Създаване на SSL/TLS обект за защита на връзка с клиент

- **def WaitForLogin(self, adminSocket, adminAddress)** - приема опити за идентификация (login) на потребители; връща отговор след всеки опит за идентификация; терминира връзката при получаване на прекалено много опити за идентификация от един и същи потребител; в случай на успешна идентификация - създава и стартира нишка (thread), която отговаря за обмена на информация с този клиент (фиг. 4.22.);

```

92     # Username and password correct:
93     elif returnCode == 1:
94         print("Authenticated")
95         # Send an authentication (auth) message to the client
96         Connection().SendMessage(adminSocket, b"auth")
97         # Create a data exchange thread for the connection
98         adminThread = AdminAppDataExchange(adminSocket)
99         # Start the thread
100        adminThread.start()
101        return True

```

Фиг. 4.22. Създаване на нишка за обмен на информация

- **def AuthenticateUser(self, adminSocket, adminAddress)** - приема име и парола от даден клиент; сравнява хеша на въведената от потребителя парола с правилния;
- **def GetPasswordHash(self, password, salt)** - генерира хеш на въведената парола по същата процедура, използвана при създаването на хешове за потребителските акаунти;
- **class AdminAppDataExchange(Thread)** - наследява от Thread класа, което позволява изпълнението му в отделна нишка (процес). Служи за: приемане и обработване на заявки от управляващите приложения; препращане на заявки към съответните пътни знаци; връщане на отговори на управляващите приложения
 - **def __init__(self)** - конструктор методът на класа;
 - **def run(self)** - изпълнява се при стартиране на нишката;
 - **def ListenToUser(self)** - приема заявки от управляващото приложение;
 - **def HandleUserRequest(self, data)** - обработва приети заявки;
 - **def HandleGetRequest(self, request)** - обработва получени GET заявки; при необходимост изпраща заявка към пътен знак; връща отговор на управляващото приложение;
 - **def HandleSetRequest(self, commands)** - обработва получени SET заявки; изпраща заявка към необходимия пътен знак; връща отговор на управляващото приложение.
- **class TrafficSignConnectionHandler** - служи за: създаване на socket сървър, към който пътните знаци се свързват; установяване на връзки; създаване на процеси за обмен на информация с пътните знаци
 - **def __init__(self)** - конструктор методът на класа. Създава socket сървър, който изчаква входящи връзки;

- **def WaitForConnection(self)** - приема входяща връзка от пътен знак; създава и стартира нишка (thread), която отговаря за обмена на информация с този пътен знак.
- **class TrafficSignDataExchange(Thread)** - наследява от Thread класа, което позволява изпълнението му в отделна нишка (процес). Служи за: приемане на уникален идентификатор от всеки свързан пътен знак; добавяне на свързан пътен знак към речника с активни устройства
 - **def __init__(self)** - конструктор методът на класа;
 - **def run(self)** - изпълнява се при стартиране на нишката;
 - **def AddDevice(self, socket)** - приема уникален идентификатор от свързания пътен знак и добавя неговия socket към речника с активни устройства, който се пази на уеб сървъра.
- **class Connection** - статичен клас. Служи за: съхраняване на речник със свързани към сървъра пътни знаци; изпращане на съобщения към управляващите приложения в коректен формат; получаване на съобщения от управляващите приложения; изпращане на съобщения към пътните знаци в коректен формат
 - **def SendMessage(sock, data)** - пакетира информация и я изпраща към даден socket. Осигурява получаването на точно една заявка в отсрецната страна като първо изпраща дължината на съобщението. Това се налага, понеже socket обектите нямат метод за различаване на различни изпращания и при приемане на съобщения с променлива дължина не може да се предвиди колко символа трябва да бъдат прочетени до достигане на края на заявката. Този метод е показан във фиг. 4.23.;

```

15 # Sends a message to the socket in the agreed format
16 @staticmethod
17 def SendMessage(sock, data):
18     length = len(data)
19     # Send the length of the message
20     sock.sendall(
21         struct.pack("!I", length) # Pack the length in unsigned int format
22     )
23     # Send the message itself
24     sock.sendall(data)

```

Фиг. 4.23. Изпращане на съобщение към даден socket

- **def ReceiveMessage(sock)** - получава и разпакетира съобщения, пакетирани по гореописания метод;
- **def ReceiveAll(sock, count)** - чете *count* символа от даден socket;
- **def SendSetRequest(targetIMEI, request, value)** - изпраща SET заявка към даден пътен знак и изчаква получаване на

потвърждение; ако не получи такова - премахва пътния знак от речника със свързани устройства (фиг. 4.24.);

```
74     # Wait for device response; timeout after 10 seconds
75     ready = select.select([deviceSocket], [], [], 10)
76     if ready[0]:
77         # Read the acknowledgement
78         data = deviceSocket.recv(2)
79         return "success"
80     else:
81         print("Error, device not responding")
82         # Remove the device from the list of available devices
83         Connection().deviceList.pop(targetIMEI)
84         return "noresp"
```

Фиг. 4.24. Изчакване на потвърждение от пътен знак

- **def CompressRequest(request)** - преобразува 2-рия аргумент на получена от управляващо приложение SET заявка в 3-буквено съкращение, подходящо за изпращане на пътен знак;
- **def CompressValue(value)** - преобразува стойността (3-тия аргумент) на получена от управляващо приложение SET заявка в 3-буквено съкращение, подходящо за изпращане на пътен знак;
- **def SendGetRequest(targetIMEI, request)** - изпраща GET заявка към даден пътен знак и изчаква получаване на отговор; ако не получи такъв - премахва пътния знак от речника със свързани устройства;

Във файла *WebServer.py* е дефинирана главната функция, която се изпълнява при стартиране на уеб сървъра (фиг. 4.25.). Тя създава 2 нишки, които едновременно приемат връзки от управляващи приложения и пътни знаци.

```
28 # Main method of the application - executes first
29 if __name__ == "__main__":
30
31     # Create an AdminAppConnectionHandler() object
32     adminConnections = AdminAppConnectionHandler()
33     # Create a TrafficSignConnectionHandler() object
34     deviceConnections = TrafficSignConnectionHandler()
35
36     # Create a thread for handling admin application connections
37     threading.Thread(target=HandleAdminConnections).start()
38     # Create a thread for handling traffic sign device connections
39     threading.Thread(target=HandleDeviceConnections).start()
40
```

Фиг. 4.25. Главна (*main*) функция на уеб сървъра

При успешно осъществяване на връзка, функциите **HandleAdminConnections()** и **HandleDeviceConnections()** рекурсивно извикват себе си в нови нишки, за да се позволи едновременното свързване на множество управляващи приложения и пътни знаци (фиг. 4.26.).

```
11 # Handles admin application connections
12 def HandleAdminConnections():
13     # Wait for a connection
14     adminSocketTuple = adminConnections.WaitForConnection()
15     # Create a thread for another connection
16     threading.Thread(target=HandleAdminConnections).start()
17     if adminSocketTuple:
18         # Handle the login process for the established connection
19         adminConnections.WaitForLogin(adminSocketTuple[0], adminSocketTuple[1])
20
21 # Handles traffic sign device connections
22 def HandleDeviceConnections():
23     # Wait for a connection
24     connected = deviceConnections.WaitForConnection()
25     # Create a thread for another connection
26     threading.Thread(target=HandleDeviceConnections).start()
```

Фиг. 4.26. Функции за осъществяване на връзки с управляващи приложения и пътни знаци

4.4. Проектиране на компютърен софтуер за управление на пътни знаци

Компютърното приложение за управление на пътни знаци е отговорно за:

- Изграждане и поддържане на сесия с уеб сървъра;
- Изпращане на заявки и получаване на отговори от уеб сървъра;
- Предоставяне на удобен графичен интерфейс за управление на устройствата;
- Извеждане на информационни съобщения за резултатите от изпратените към пътните знаци заявки.

4.4.1. Използвани инструменти за разработка

Инструментите, които бяха използвани при разработката на управляващото приложение, са:

- **Python език за програмиране** - избран заради библиотеката PyQt, която позволява създаване на междуплатформени графични интерфейси;
- **Интегрирана среда за разработка Visual Studio Code;**
- **Black** - инструмент за автоматично форматиране на python код;
- **Qt Designer** - инструмент за проектиране и изграждане на графични потребителски интерфейси (Graphical User Interfaces (GUIs)).

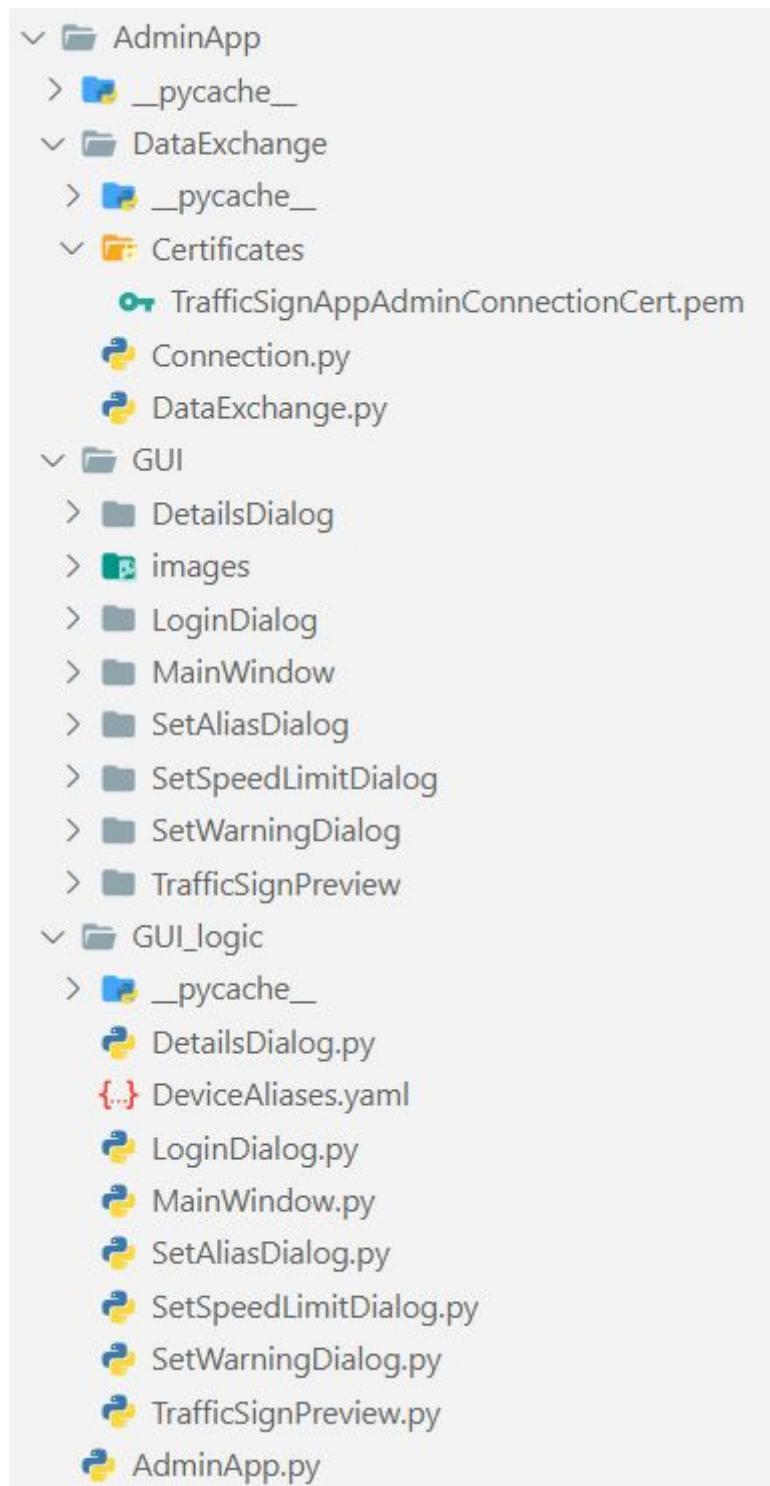
4.4.2. Използвани библиотеки

Документацията на използваните библиотеки е приложена в {14} - {19} и {22}. Работата на сървъра зависи от следните библиотеки:

- **socket** - (т. 4.3.2.);
- **ssl** - (т. 4.3.2.);
- **yaml** - (т. 4.3.2.);
- **hashlib** - (т. 4.3.2.);
- **select** - (т. 4.3.2.);
- **struct** - (т. 4.3.2.);
- **yaml** - (т. 4.3.2.);
- **PyQt5** - Qt представлява набор от междуплатформени (cross-platform) C++ библиотеки за създаване на графични потребителски интерфейси. PyQt5 е изчерпателен набор от връзки в Python за Qt библиотеките, който позволява езикът да се използва като алтернатива на C++ при разработката на Qt приложения.

4.4.3. Структура на хранилището на управляващото приложение

Във фиг. 4.27. е показана структурата на файловата система на управляващото приложение



Фиг. 4.27. Файлова система на управляващото приложение

Предназначението на всички папки и файлове (разгледани от горе надолу) е следното:

- **AdminApp/** - съдържа всички файлове на приложението;
- **DataExchange/** - съдържа всички файлове, които отговарят за връзката между приложението и уеб сървъра;
- **Certificates/** - съдържа сертификатите на уеб сървъра ;
- **TrafficSignAppAdminConnectionCert.pem** - SSL сертификат;
- **Connection.py** - служи за форматиране, изпращане и получаване на съобщения от уеб сървъра;
- **DataExchange.py** - служи за свързване към уеб сървъра и за обмен на информация с него;
- **GUI/** - съдържа всички файлове, които имат отношение към графичния интерфейс на приложението;
- **DetailsDialog/ ; LoginDialog/ ; MainWindow/ ; SetAliasDialog/ ; SetSpeedLimitDialog/ ; SetWarningDialog/ ; TrafficSignPreview/** - всяка от тези папки съдържа 2 файла:
 - .ui файл - Qt Designer файл, който съдържа конфигурацията на съответния прозорец от графичния интерфейс;
 - .py файл - автоматично генериран Python код, който изобразява графичния интерфейс на прозореца при изпълнение на setupUi() метода на съответния клас.
- **images/** - съдържа всички изображения, които приложението използва;
- **GUI_logic/** - съдържа всички файлове, които имат отношение към логиката на графичния интерфейс;
- **DetailsDialog.py ; LoginDialog.py ; MainWindow.py ; SetAliasDialog.py ; SetSpeedLimitDialog.py ; SetWarningDialog.py ; TrafficSignPreview.py** - служат за определяне на логиката на всеки от съответстващите им прозорци от графичния интерфейс на приложението
- **DeviceAliases.yaml** - съдържа всички въведени от потребителя; псевдоними за идентификатори на пътни знаци;
- **AdminApp.py** - съдържа основната логика на приложението. Изпълнението на този файл стартира графичния интерфейс.

4.4.4. Блокова схема на алгоритъма

Във фиг. 4.28. е показана блоковата схема на алгоритъма на компютърното приложение за управление на пътни знаци

Admin app block scheme

4.4.5. Класове, методи и променливи на програмата

Предназначените на всички класове и методи на управляващото приложение са както следва:

- **class Connection** - статичен клас. Служи за: добавяне на SSL/TLS към socket обекта, отговорен за комуникацията с уеб сървъра и проверка на сертификата, получен от него; поддържане на списък с налични на сървъра пътни знаци; изпращане на съобщения към уеб сървъра в коректен формат; получаване и прочитане на съобщения от него; терминиране на връзката при нужда. Във фиг. 4.29. е показано създаването на защищен socket чрез въвеждане на SSL/TLS и проверка на името на организацията, създала сертификата, получен от уеб сървъра.

```
10 class Connection:  
11     # Create an unwrapped IPv4 TCP socket object  
12     unwrapped = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
13     # Create an SSLContext object  
14     context = ssl.SSLContext(  
15         ssl.PROTOCOL_TLS_CLIENT # Client side connection (auto-negotiate highest supported SSL/TLS version)  
16     )  
17     # Load the certificate used for the application  
18     context.load_verify_locations(  
19         "./DataExchange/Certificates/TrafficSignAppAdminConnectionCert.pem"  
20     )  
21     # Create a protected SSL/TLS TCP socket if the server hostname is correct  
22     client_socket = context.wrap_socket(  
23         sock=unwrapped, server_hostname="TrafficSignAdminApp"  
24     )  
25     # Blocking socket operations timeout after 20 seconds  
26     client_socket.settimeout(20)
```

Фиг. 4.29. Въвеждане на SSL/TLS за връзката с уеб сървъра и проверка на получени сертификат

В този клас се пази и списък със свързани към уеб сървъра пътни знаци, както и речник, който свързва уникалния идентификатор на всяко устройство с въведен от потребителя псевдоним (фиг. 4.30.)

```
28     deviceList = [] # All currently available devices' IMEIs  
29     knownDevices = {} # All known custom user aliases (key == Alias, value == IMEI)
```

Фиг. 4.30. Структури от данни за запазване на информация за налични пътни знаци

- **def SendMessage(sock, data)** - еквивалентен метод на използвания от уеб сървъра (стр. 86);
- **def ReceiveMessage(sock)** - еквивалентен метод на използвания от уеб сървъра (стр. 86);
- **def ReceiveAll(sock, count)** - еквивалентен метод на използвания от уеб сървъра (стр. 86);

- **def Close()** - терминира socket обекта, свързан към уеб сървъра.
- **class DataExchange** - служи за: изграждане на връзка с уеб сървъра; обмен на информация с него
 - **def __init__(self)** - конструктор методът на класа. Дефинира IPv4 адреса на уеб сървъра и порта, на който работи приложението;
 - **def AttemptConnect(self)** - опитва да изгради връзка между приложението и уеб сървъра;
 - **def AttemptLogin(self, username, password)** - изпраща въведената от потребителя идентификационна информация към сървъра; прочита получения отговор за съответния опит за идентификация (login);
 - **def GetDevices(self)** - изпраща заявка за получаване на информация за всички пътни знаци, свързани към уеб сървъра и актуализира запазения локално списък с устройства;
 - **def GetDeviceDetails(self, target)** - изпраща заявка за получаване на детайлна информация за състоянието на определен пътен знак;
 - **def SetSpeedLimit(self, target, speedLimit)** - изпраща заявка за визуализиране на ограничение на скоростта върху даден пътен знак;
 - **def SetWarning(self, target, request)** - изпраща заявка за визуализиране на предупреждение върху даден пътен знак;
 - **def WaitForData(self)** - изчаква получаване на отговор от уеб сървъра. Връща съобщение за грешка ако такъв не бъде получен в рамките на timeout периода (2 сек).

Някои от методите, които се използват в голяма част от класовете, дефиниращи логиката на графичния интерфейс, изпълняват почти или напълно идентични функции. Общото предназначение на тези методи е разгледано преди съответните класове:

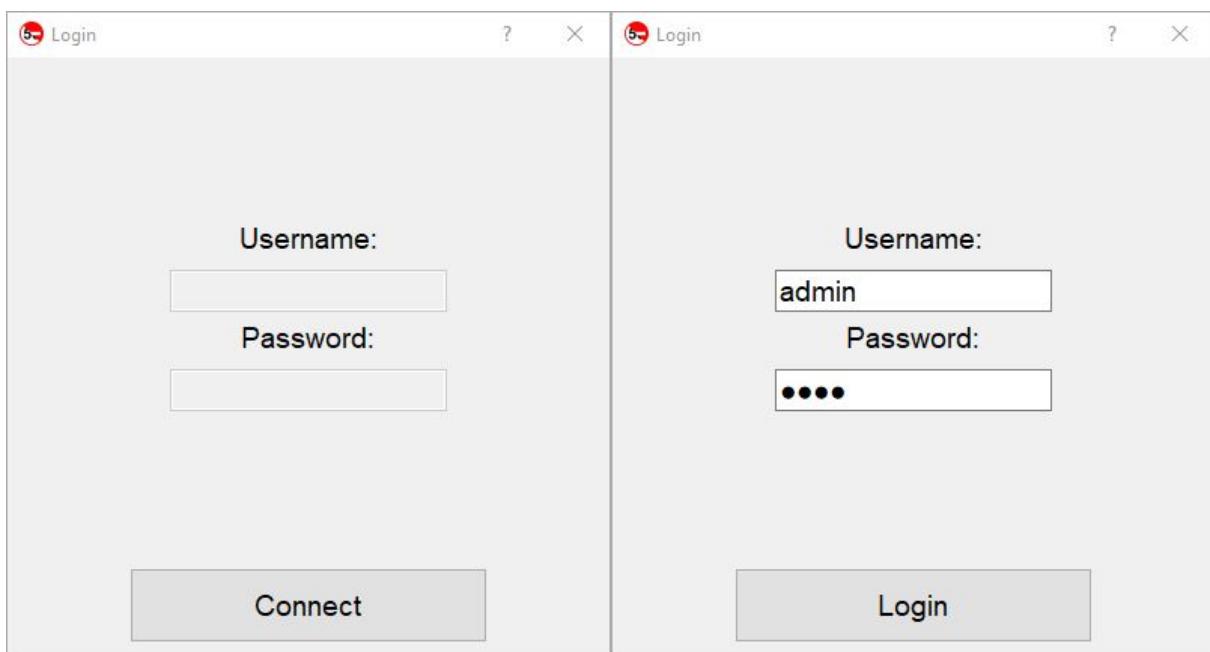
- **def __init__(self)** - конструктор метод на даден клас. Извиква .setupUi(self) метода на автоматично генерирания файл, в който се пази графичния интерфейс на съответния прозорец, като по този начин го изобразява. Също така извиква self.SetupFunctionality() метода на своя клас и дефинира променливи, които той използва;
- **def SetupFunctionality(self)** - дефинира допълнителната логика, нужна за функционирането на даден графичен прозорец. При нужда променя елементи от графичния интерфейс. Добавя икона с логото на приложението;

- **def QuitDialog(self)** - затваря графичния интерфейс на диалоговия прозорец и връща резултат от изпълнението му на метода, който го е изобразил;
- **def keyPressEvent(self, event)** - отговаря за дефинирането на преки пътища в графичния интерфейс, които се достъпват чрез клавиатурата. Този метод се извиква всеки път, когато бъде натиснат някой клавиш;
- **def closeEvent(self, event)** - извиква се при затваряне на съответния прозорец. Обикновено се конфигурира да пренасочва към друг метод, който служи за коректното затваряне на прозореца.

Графичният интерфейс на всеки от прозорците на приложението, заедно със съществуващата го управляваща логика, са разгледани в т. 4.4.5.1. - т.4.4.5.7.

4.4.5.1. Графичен интерфейс и управляваща логика на диалоговия прозорец за влизане в системата (Login Dialog)

Графичният интерфейс на този диалогов прозорец е показан във фиг. 4.31.



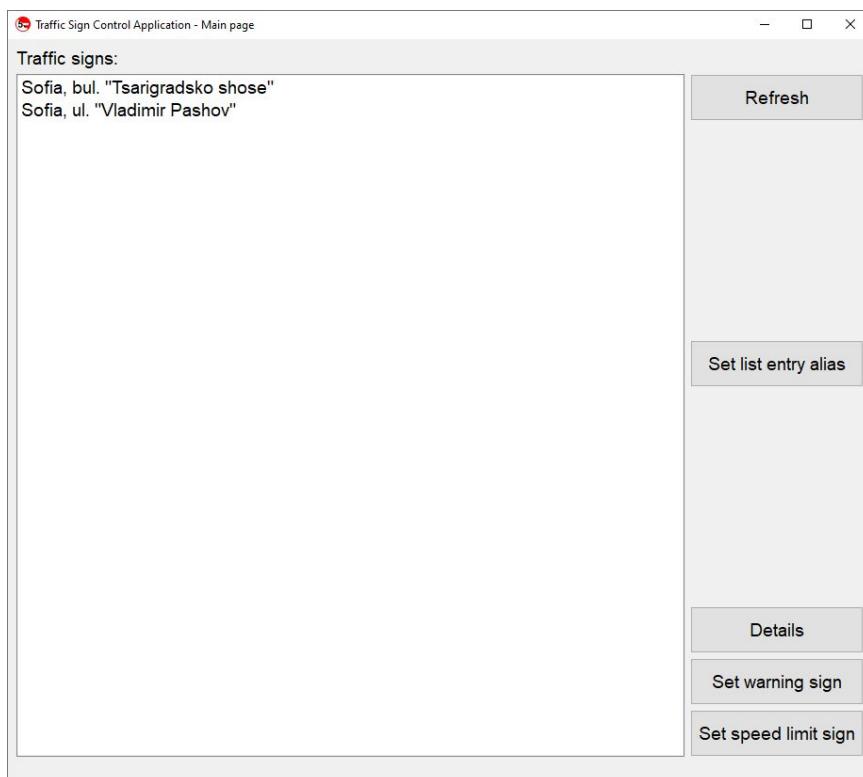
Фиг. 4.31. Графичен интерфейс на диалоговия прозорец за влизане в системата

Управляващата логика на този диалогов прозорец е дефинирана в класа **class DetailsDialog(QDialog)**. Методите на този клас изпълняват следните функции:

- **def EnableInput(self)** - позволява въвеждането на име и парола;
- **def DisableInput(self)** - забранява въвеждането на име и парола;
- **def TryConnect(self)** - опитва да установи връзка с уеб сървъра; при неуспешно свързване изобразява съобщение за грешка и дава възможност за повторен опит за връзка;
- **def ConnectClick(self)** - изпълнява се при натискане на Connect/Login бутона на прозореца. Изпълнява 2 функции в зависимост от състоянието на връзката:
 - Ако приложението не е свързано към уеб сървъра - извиква метода за свързване;
 - Ако приложението е свързано към уеб сървъра - изпраща въведените потребителско име и парола и приема отговор. Ако опитът за идентификация е бил успешен - затваря диалоговия прозорец. Ако опитът за идентификация е бил неуспешен - извежда съобщение за грешка. Ако връзката е била отхвърлена от уеб сървъра - извежда съобщение за грешка, затваря диалоговия прозорец и прекъсва работата на приложението.

4.4.5.2. Графичен интерфейс и управляваща логика на основния прозорец на приложението (Main Window)

Графичният интерфейс на този прозорец е показан във фиг. 4.32.



Фиг. 4.32. Графичен интерфейс на основния прозорец на приложението

Управляващата логика на този прозорец е дефинирана в класа **MainWindow(QMainWindow)**. Методите на този клас изпълняват следните функции:

- **def UpdateDeviceList(self)** - изпълнява се при натискане на *Refresh* бутона. Изисква информация от уеб сървъра за наличните пътни знаци и обновява изобразения в прозореца списък с налични устройства;
- **def GenerateDeviceList(self)** - генерира списък, подходящ за изобразяване в прозореца, като извежда въведените псевдоними за всеки от наличните пътни знаци. В случай че определен пътен знак е нов и за него липсва запис в речника с псевдоними - създава запис, като за псевдоним използва идентификатора му (IMEI). Този метод е показан във фиг. 4.33.;

```
77 # Generates the device list that is to be displayed to the user
78 def GenerateDeviceList(self):
79     displayList = []
80     # Loop through every device in the list
81     for device in Connection().deviceList:
82         # If the device is new and the IMEI is unknown:
83         if device not in Connection().knownDevices.values():
84             # Add it to the list of known devices
85             Connection().knownDevices[device] = device # Alias == IMEI (initially)
86
87         # Find the device alias from its IMEI and add it to the list
88         # (Search dictionary by value)
89         displayList.append(
90             list(Connection().knownDevices.keys())[list(Connection().knownDevices.values()).index(device)])
91
92     )
93
94     # Return the updated display list
95     return displayList
```

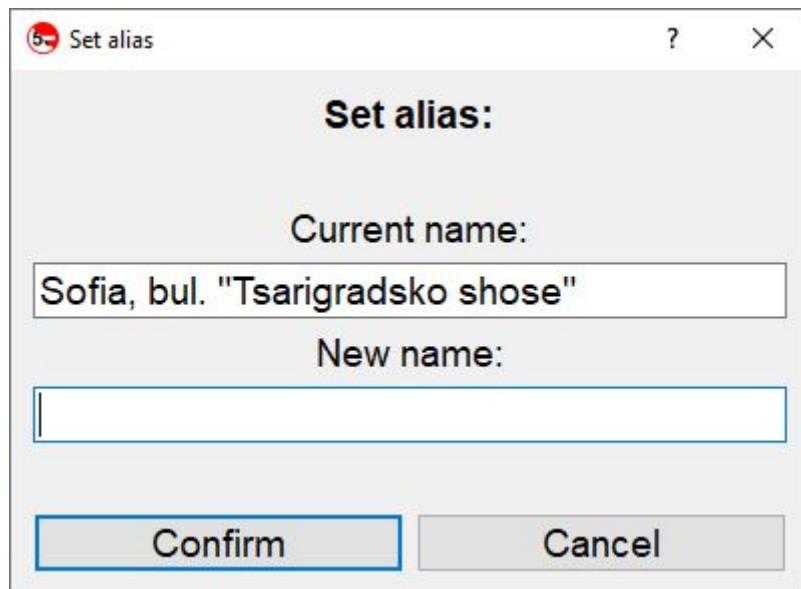
Фиг. 4.33. Извеждане на псевдоними за наличните пътни знаци

- **def LoadKnownDevices(self)** - прочита и зарежда речника с псевдоними, запазен в *DeviceAliases.yaml* файла;
- **def SaveKnownDevices(self)** - запазва речника с псевдоними в *DeviceAliases.yaml* файла;
- **def ShowSetSpeedLimitDialog(self)** - създава и показва диалогов прозорец за изобразяване на ограничение на скоростта върху избрания пътен знак при натискане на *Set speed limit sign* бутона;
- **def ShowSetWarningDialog(self)** - създава и показва диалогов прозорец за изобразяване на предупреждение върху избрания пътен знак при натискане на *Set warning sign* бутона;

- **def ShowSetAliasDialog(self)** - създава и показва диалогов прозорец за промяна на псевдонима на избрания пътен знак при натискане на *Set list entry alias* бутона;
- **def ShowDetailsDialog(self)** - създава и показва диалогов прозорец за изобразяване на детайлна информация за избрания пътен знак при натискане на *Details* бутона;
- **def ConvertStatus(self, status)** - преобразува получения от пътния знак 3-буквен код за вида на изобразяваното съобщение в лесно четима информация;
- **def ConvertValue(self, value)** - преобразува получения от пътния знак 3-буквен код за стойността на изобразяваното предупредително съобщение в лесно четима информация;
- **def GetSelectedDevice(self)** - връща псевдонима на избрания от потребителя пътен знак от списъка;
- **def Exit(self)** - запазва псевдонимите на устройствата в *DeviceAliases.yaml* файла, терминира връзката с уеб сървъра и затваря прозореца.

4.4.5.3. Графичен интерфейс и управляваща логика на диалоговия прозорец за промяна на псевдонима на пътен знак (Set Alias Dialog)

Графичният интерфейс на този прозорец е показан във фиг. 4.34.



Фиг. 4.34. Графичен интерфейс на диалоговия прозорец за промяна на псевдонима на пътен знак

Управляващата логика на този диалогов прозорец е дефинирана в клас **SetAliasDialog(QDialog)**. Методите на този клас изпълняват следните функции:

- **def SetupFunctionality(self)** (освен стандартната функционалност) - въвежда регулярен израз (Regular expression - RegEx), който служи за осигуряване на валидност на псевдонима, въведен от потребителя (фиг. 4.35.);

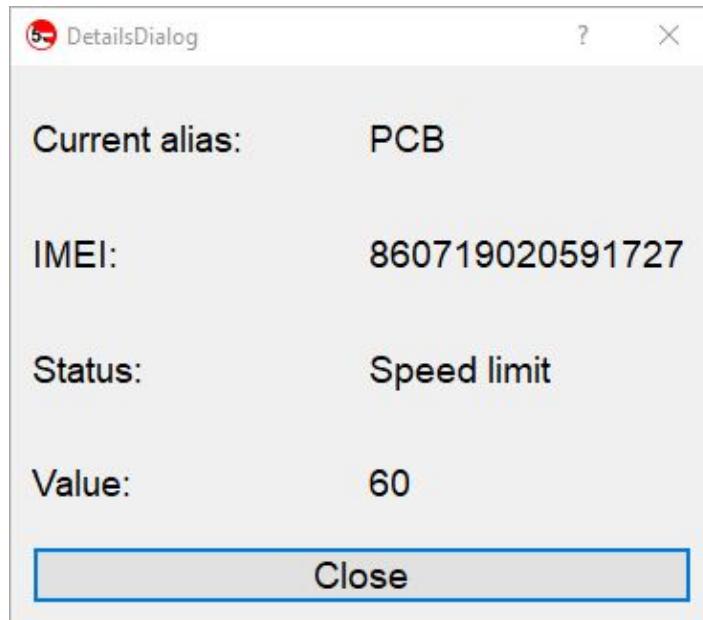
```
35     # Create a regular expression to prevent invalid data entry
36     inputRegEx = QRegExp(
37         "\S.*" # First symbol not whitespace, any length, any character
38     )
39     # Create a validator object
40     validator = QRegExpValidator(inputRegEx)
41     # Attach the validator to the input textbox
42     self.ui.NewNameTextBox.setValidator(validator)
```

Фиг. 4.35. Въвеждане на регулярен израз за потребителския вход

- **def SetAlias(self)** - актуализира избрания псевдоним при натискане на Confirm бутона. Извежда съобщение за грешка при въвеждане на невалидна информация.

4.4.5.4. Графичен интерфейс и управляваща логика на диалоговия прозорец за изобразяване на детайлна информация за избран пътен знак (Details Dialog)

Графичният интерфейс на този прозорец е показан във фиг. 4.36.



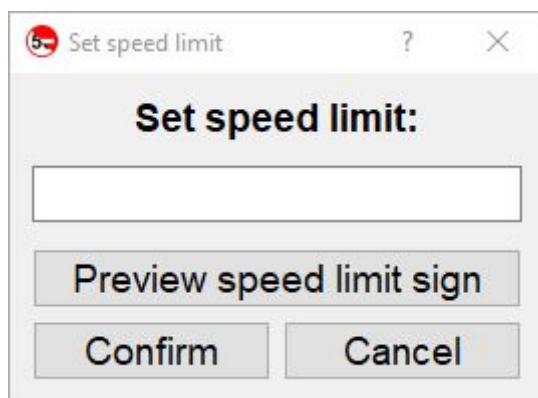
Фиг. 4.36. Графичен интерфейс на диалоговия прозорец за изобразяване на детайлна информация за избран пътен знак

Управляващата логика на този диалогов прозорец е дефинирана в класа **class DetailsDialog(QDialog)**. Методите на този клас изпълняват следните функции:

- **def __init__(self, alias, IMEI, status, value)** (освен стандартната функционалност) - записва получената информация за пътния знак в променливи, които се изобразяват в прозореца.

4.4.5.5. Графичен интерфейс и управляваща логика на диалоговия прозорец за изобразяване на ограничение на скоростта върху избран пътен знак (Set Speed Limit Dialog)

Графичният интерфейс на този прозорец е показан във фиг. 4.37.



Фиг. 4.37. Графичен интерфейс на диалоговия прозорец за изобразяване на ограничение на скоростта върху избран пътен знак

Управляващата логика на този диалогов прозорец е дефинирана в класа **class SetSpeedLimitDialog(QDialog)**. Методите на този клас изпълняват следните функции:

- **def SetupFunctionality(self)** (освен стандартната функционалност) - въвежда регулярен израз (Regular expression - RegEx), който служи за осигуряване на валидност на ограничението, въведено от потребителя (фиг. 4.38.);

```

36      # Create a regular expression to prevent invalid data entry
37      inputRegEx = QRegExp(
38          "[1-9]\d{0,2}" # Max 3 symbols
39          # First symbol - digit between 1 and 9
40          # Second/third symbol - any whole digit
41      )
42      # Create a validator object
43      validator = QRegExpValidator(inputRegEx)
44      # Attach the validator to the input textbox
45      self.ui.SpeedLimitTextBox.setValidator(validator)

```

Фиг. 4.38. Въвеждане на регулярен израз за потребителския вход

- **def SetSpeedLimit(self)** - изпраща заявка на уеб сървъра за изобразяване на ограничение на скоростта върху даден пътен знак при натискане на *Confirm* бутона. Извежда съобщение за грешка при въвеждане на невалидна информация;
- **def HandleResponse(self, response)** - прочита и обработва получения отговор на заявката. При нужда извежда съобщения за грешка;
- **def DisplayPreview(self)** - създава и показва диалогов прозорец с предварителен изглед на пътния знак, който ще бъде изобразен в следствие на изпращането на заявката. Изпълнява се при натискане на *Preview speed limit sign* бутона.

4.4.5.6. Графичен интерфейс и управляваща логика на диалоговия прозорец за изобразяване на предварителен изглед на пътен знак (Traffic Sign Preview Dialog)

Графичният интерфейс на този прозорец е показан във фиг. 4.39.



Фиг. 4.39. Графичен интерфейс на диалоговия прозорец за изобразяване на предварителен изглед на пътен знак

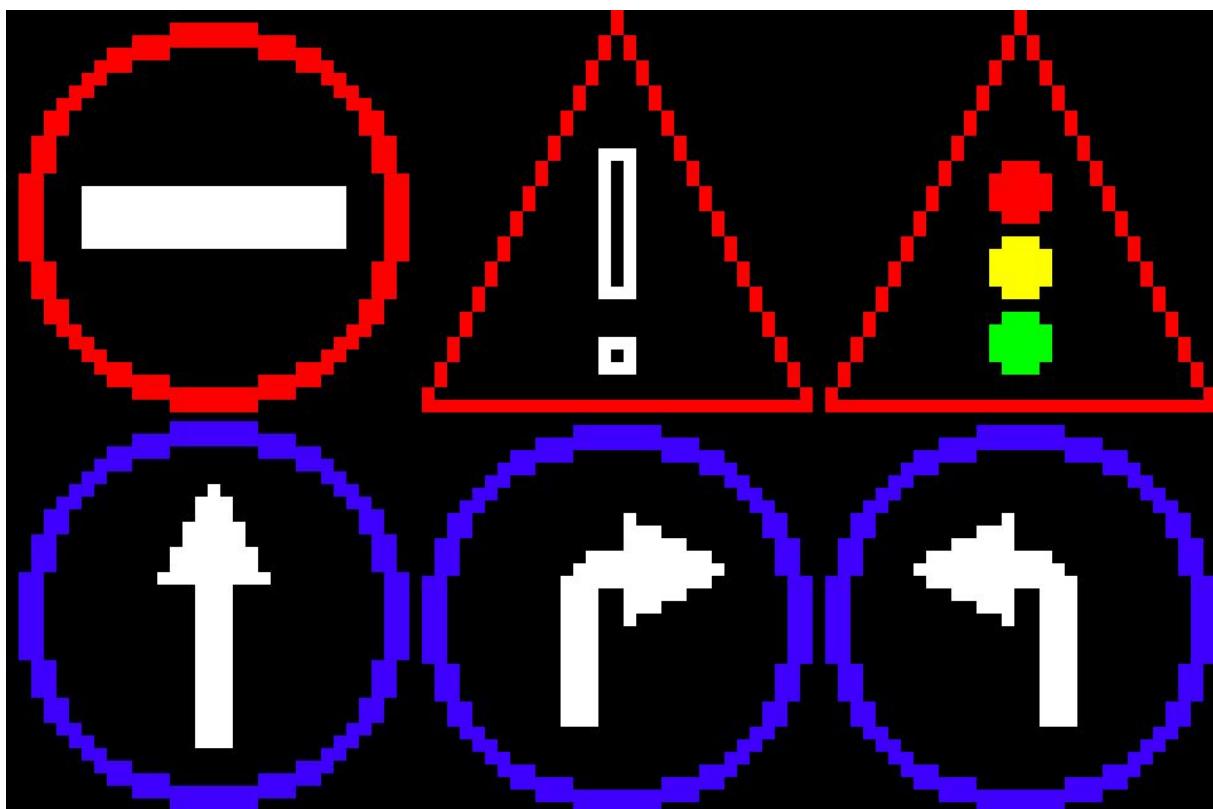
Управляващата логика на този диалогов прозорец е дефинирана в класа **class TrafficSignPreview(QDialog)**.

4.4.5.7. Графичен интерфейс и управляваща логика на диалоговия прозорец за изобразяване на предупреждение върху избран пътен знак (Set Warning Dialog)

Графичният интерфейс на този прозорец е показан във фиг. 4.40. Изображенията на всички налични предупредителни знаци (освен показания във фиг. 4.40. знак “СТОП”) са показани във фиг. 4.41.



Фиг. 4.40. Графичен интерфейс на диалоговия прозорец за изобразяване на ограничение на скоростта върху избран пътен знак



Фиг. 4.41. Предупредителни пътни знаци

Управляващата логика на този диалогов прозорец е дефинирана в класа **class SetWarningDialog(QDialog)**. Методите на този клас изпълняват следните функции:

- **def UpdateImage(self)** - обновява предварителния изглед, който изобразява избрания в момента предупредителен знак. Изпълнява се при промяна на избрания от потребителя знак. Този метод е показан във фиг. 4.42.;

```

42     # Updates the image preview to show the currently selected traffic sign
43     def UpdateImage(self):
44         # Get the currently selected traffic sign name
45         self.currentSelection = self.GetCurrentSelection()
46         # If the selection is valid:
47         if self.currentSelection:
48             # Generate a path to the image of the selected traffic sign
49             image = "./GUI/images/" + self.currentSelection + ".png"
50             # Create a QPixmap() object with the image
51             pixmap = QPixmap(image)
52             # Display the image
53             self.ui.ImageLabel.setPixmap(pixmap)

```

Фиг. 4.42. Промяна на изображението на предварителния изглед

- **def GetCurrentSelection(self)** - връща името на избрания предупредителен знак във формат, който може да се използва при

- генериране на път към изображението на предварителния му изглед;
- **def SetWarning(self)** - изпраща заявка на уеб сървъра за изобразяване на предупреждение върху даден пътен знак при натискане на *Confirm* бутона;
- **def HandleResponse(self, response)** - прочита и обработва получения отговор на заявката. При нужда извежда съобщения за грешка.

Пета глава

Проектиране на графични оригинали на печатни платки за микроконтролерна система за дистанционно управление на светодиоден пътен знак

5.1. Принципна електрическа схема на печатната платка на устройството

За проектирането на печатната платка на микроконтролерното устройство е използван модифициран вариант на пълната принципна електрическа схема. Той е показан във фиг. 5.1. Основните разлики между двата варианта на схемата са:

- **Премахване на блок “Зареждане”** - това се налага, тъй като елементите от този блок се разполагат отделно от печатната платка;
- **Премахване на акумулаторната батерия и прекъсвача от блок “Захранване”** - тези елементи също не се поставят върху печатната платка и затова са премахнати;
- **Добавяне на захранващ конектор J3** - служи за свързване на акумулаторната батерия, която захранва устройството, към печатната платка;
- **Добавяне на монтажни дупки H1-H4** - при проектирането на печатната платка са предвидени и 4 монтажни дупки, които служат за по-лесното ѝ фиксиране.

PCB schematic

5.2. Особености на печатната платка на устройството

Основните характеристики на печатната платка на устройството са разгледани в таблица 5.1.

| | |
|----------------------------|-----------------------|
| Размери на платката: | 106.68mm / 90.424mm |
| Брой слоеве: | 2 |
| Материал: | FR4 |
| Дебелина на материала: | 1.55mm |
| Дебелина на медното фолио: | 35µm |
| Покритие: | Sn-Pb (калай-олово) |
| Зашитна маска: | Двустранна (зелена) |
| Ситопечат: | Едностраниен (отгоре) |

Таблица 5.1. Основни характеристики на печатната платка на устройството

Проектирането на платката е съобразено с технологичните възможности на производителя, като основните изисквания са разгледани в таблица 5.2.

| | |
|---------------------------------|---------------------------|
| Минимален размер на отвор: | 0.5mm (20 mils) |
| Минимално разстояние м/у писти: | 0.2mm (8 mils) |
| Минимална широчина на писта: | 0.3mm (12 mils) |
| Обща мед върху пистите: | 60 - 70µm |
| Диаметър на площадка: | Диаметър на отвор + 0.4mm |

Таблица 5.2. Изисквания към печатната платка

5.3. Разположение на елементите върху печатната платка на устройството

5.3.1. Съображения при разполагането на елементите върху печатната платка

При избирането на подходящи места за всеки от елементите на устройството са спазени някои важни изисквания:

- **Съединителите да бъдат лесно достъпни** - съединителите представляват места за свързване на външни електрически вериги към печатната платка. Към тях обикновено многократно се включват и изключват други съединители. Поради тази причина те трябва да бъдат лесно достъпни, което налага разполагането им в краищата на платката;
- **Антената да бъде изведена максимално в края на печатната платка** - по този начин платката може да бъде поставена в затворена кутия;
- **Да се осигури лесен достъп до SIM картата на GPRS модула** - тъй като може да се наложи подмяна на SIM картата на устройството с нова, GPRS модулът трябва да се разположи по начин, който позволява достъп до нея;
- **Да се осигури достатъчно голяма площ за охлаждане на импулсните регулатори** - поради големите токове, които преминават през тях, импулсните регулатори обикновено трябва да разсейват големи количества мощност. Това налага осигуряването на достатъчно големи метализирани площици, които да подобрят охлаждането на регулаторите;
- **Да се осигури достатъчно разстояние между индукторите и сигналните писти** - тъй като индукторите са източници на големи електромагнитни смущения, е необходимо да бъдат разположени на достатъчно големи разстояния от сигналните писти, за да се подсигури правилната работа на устройството;
- **Филцовите кондензатори да бъдат разположени максимално близо до интегралните схеми** - това осигурява максимално добра защита от смущения, причинени от други елементи на схемата;
- **Монтажните дупки да бъдат разположени на подходящи места** - обикновено те се поставят в краищата на платката, тъй като там са най-лесно достъпни;

- **Елементите да се разположат по функционални блокове с минимално разстояние между тях** - по този начин се осигурява по-голяма шумоустойчивост на устройството.

5.3.2. Разполагане на елементите върху печатната платка

Във фиг. 5.2. е показано крайното разположение на елементите върху печатната платка. При проектирането са следвани всички изисквания, разгледани в т. 5.3.1. Монтажните дупки са разположени в 4-те края на платката. Елементите са групирани по блокове, за да се минимизират дължините на пистите между тях.

Element placement here

5.4. Опроводяване на печатната платка на устройството

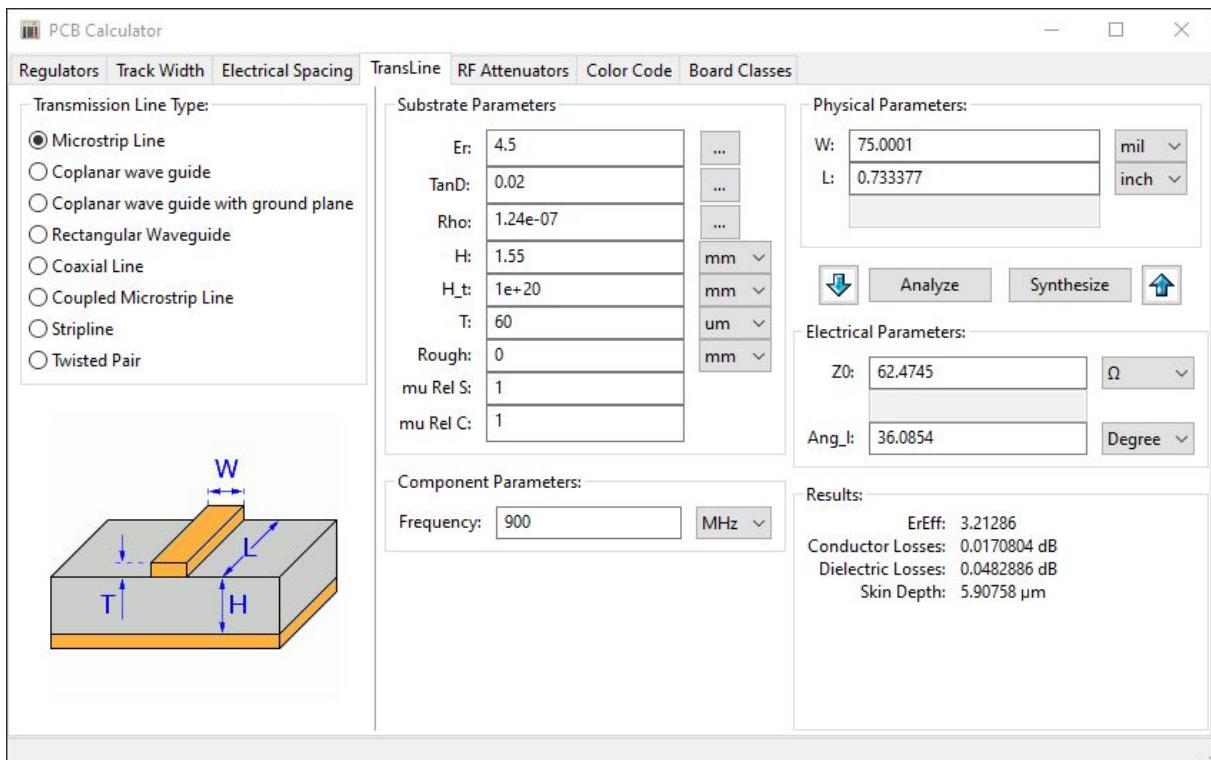
5.4.1. Съображения при опроводяването на печатната платка

Следваните изисквания при опроводяването на печатната платка са:

- **Захранващите писти да бъдат достатъчно широки** - това е от изключително значение особено за пистите на импулсните регулатори, тъй като те пренасят токове от порядъка на няколко ампера. Широките писти имат по-ниско съпротивление от тънките, с което осигуряват по-малки топлинни загуби;
- **Да се осигурят достатъчни разстояния между пистите и между пистите и площеадките** - по този начин се намалява риска от грешки при производството на платката;
- **Да се използват 45° ъгли на пистите вместо 90°** - така се намалява възможността за неправилно ецване при производство;
- **Да се осигури меден слой за подобряване на шумоустойчивостта** - добавянето на меден слой носи множество предимства, но най-важното е подобрената защитеност от електромагнитна интерференция (Electromagnetic interference (EMI)). Това е особено важно, особено при наличието на високочестотна антена, както е в случая с GSM антената;
- **Да се добавят достатъчно проходни отвори за свързване към медния слой** - това е от особена важност за SMD (Surface-mount device) елементите, които са разположени от противоположната страна на медния слой;
- **Да се добавят проходни отвори и медни слоеве под индукторите** - служат за подобряване на шумоизолацията;
- **Да се осигурят thermal relief площеадки на медния слой** - улесняват запояването на елементите върху него;
- **Да се пресметне дължината и широчината на пистата за антената** - за целта се използва калкулатор за пресмятане на т. нар. microstrip линии. Целта е постигане на 50Ω съгласуван импеданс, което минимизира загубите от предаването на високочестотни сигнали. Минимизирането на дължината на пистата също спомага за намаляване на загубите.

5.4.2. Опроводяване на печатната платка

Във фиг. 5.3. е изобразена напълно опроводената печатна платка на устройството. Върху горния слой на платката са опроводени всички сигнални и захранващи писти, а долният слой е изцяло GND (маса). Захранващите писти на импулсните регулатори са с широчина 125 mils. Сигналните писти са с широчина 15 mils. Елементите, между които има силнотокови връзки, са опроводени с максимално къси и широки писти (както е по препоръка на производителя на импулсните регулатори). Антената е опроводена с максимално къса писта. Пресмятането на съгласувания импеданс е показано във фиг. 5.4.



Фиг. 5.4. Пресмятане на съгласувания импеданс на пистата за антената

Получената стойност на импеданса ($\sim 62\Omega$) надвишава препоръчаната (50Ω), но е в рамките на допустимото при такава къса писта ($< 2\text{cm}$).

Във фиг. 5.5. е показан 3D изглед от страна елементи на печатната платка.

Routed board here

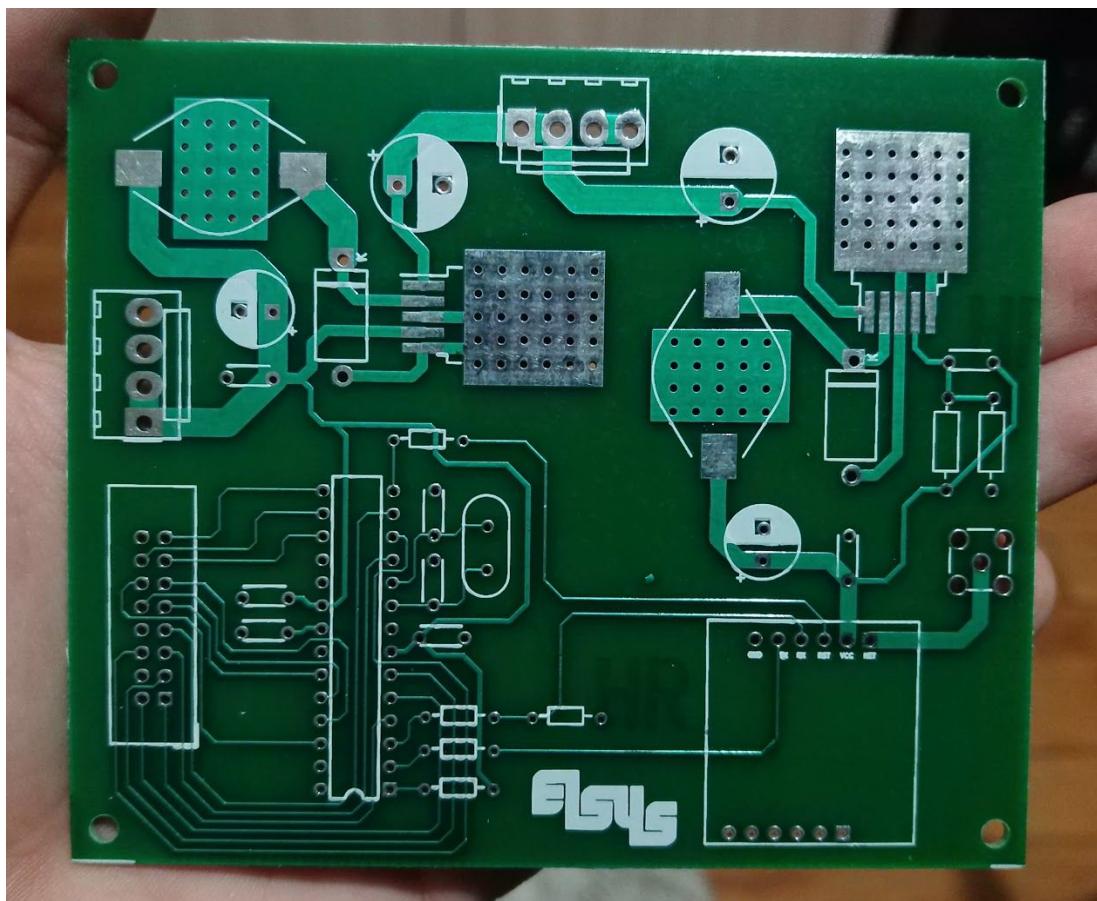
3D view here

Шеста глава

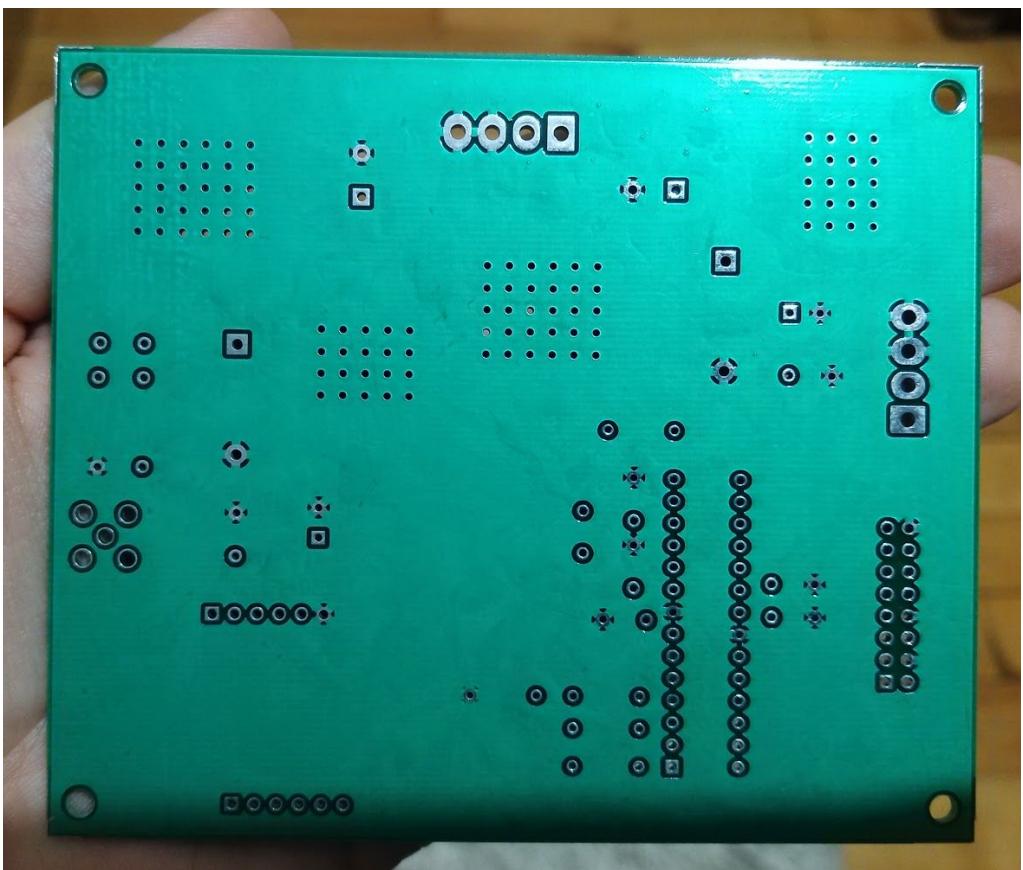
Създаване на работоспособен модел на микроконтролерна система за дистанционно управление на светодиоден пътен знак

6.1. Поръчване, насищане и оживяване на печатна платка

Ненаситената печатна платка на устройството е показана на фиг. 6.1. и фиг. 6.2.



Фиг. 6.1. Ненаситена печатна платка на устройството, страна елементи



Фиг. 6.2. Ненаситена печатна платка на устройството, страна спойки

Наситената с компоненти печатна платка е показана във фиг. 6.3.



Фиг. 6.3. Наситена печатна платка на устройството

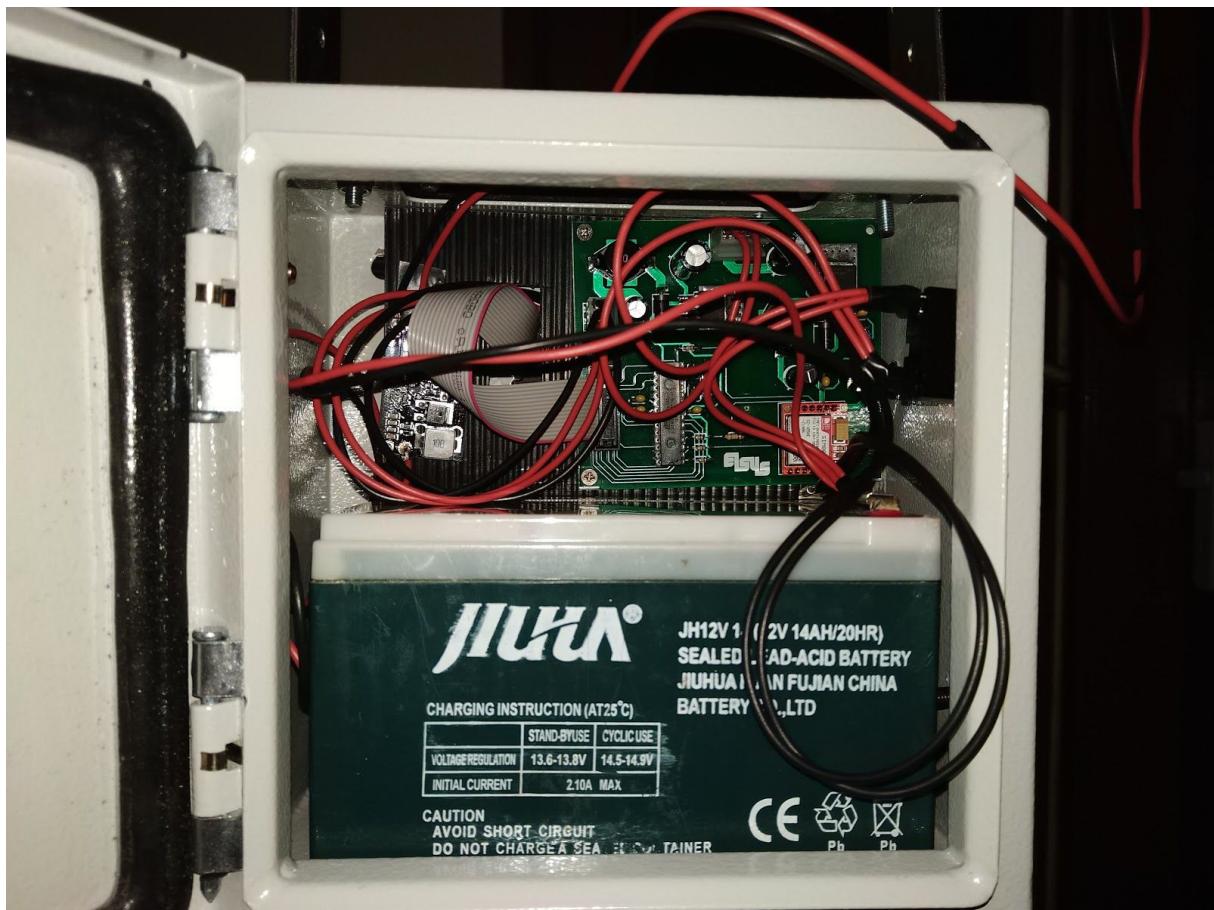
6.2. Разработване на прототип на пътен знак

С цел демонстрация е изработена прототипна стойка за устройството. Тя е направена от кутия за електрическо табло, дълъг стол за маса, стойка за чадър и метални профили. Показана е във фиг. 6.4.



Фиг. 6.4. Прототипна стойка за устройството

Всички електрически компоненти на устройството са разположени в кутията за електрическо табло. Съдържанието ѝ е показано във фиг. 6.5.



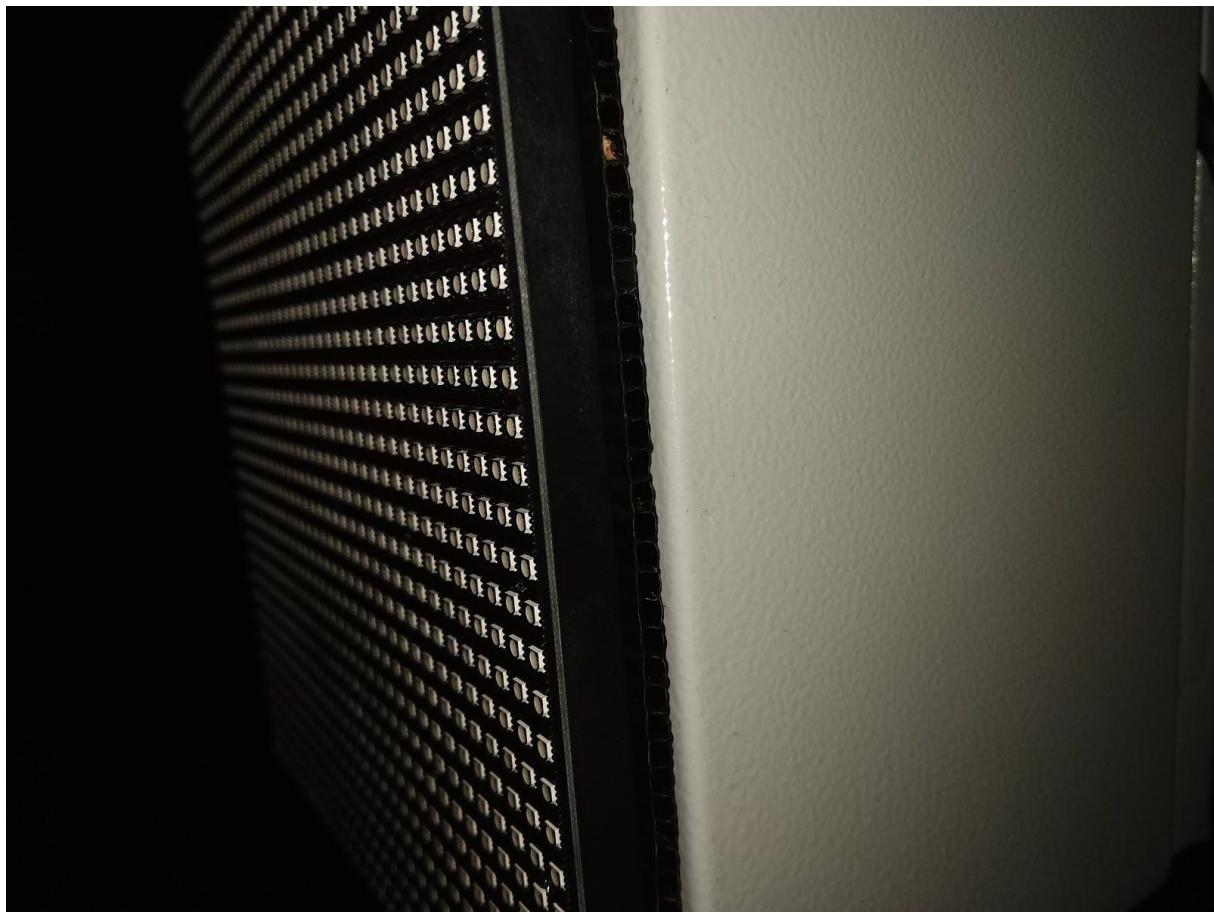
Фиг. 6.5. Кутия с компонентите на устройството

Антената и прекъсвачът са изведени отстрани на кутията, където са на открито и са лесно достъпни (фиг. 6.6.)



Фиг. 6.6. Антена и прекъсвач на устройството

Светодиодната матрица е прикрепена в предната част на кутията към изолационен материал (фиг. 6.7.). В него са пробити дупки, през които минават проводниците на матрицата.



Фиг. 6.7. Монтаж на светодиодна матрица върху изолационен материал

Фотоволтаичният панел е монтиран върху кутията с помощта на метални профили (фиг. 6.8.). Те са прикрепени по такъв начин, че панелът да може да се върти около една ос. Свързаните към съединителната кутия на соларния панел проводници влизат в кутията през отвор в горната ѝ част.



Фиг. 6.8. Монтаж на соларен панел

Изобразяването на различни пътни знаци върху пътния знак е показано във фиг. 6.9. - фиг. 6.13.



Фиг. 6.9. Изобразяване на знак "движение само наляво след знака"



Фиг. 6.10. Изобразяване на знак за ограничение на скоростта (60 км/ч)



Фиг. 6.11. Изобразяване на знак за ограничение на скоростта (5 км/ч)



Фиг. 6.12. Изобразяване на знак "СТОП"



Фиг. 6.13. Изобразяване на знак "забранено влизането"

6.3. Изчисление на цената на продукта

Общата цена на крайния продукт (закръглена до цяло число) е изчислена в таблица 6.1.

| Продукт: | Цена: |
|---|----------------|
| Печатна платка | 25 лв. |
| Светодиодна матрица | 34 лв. |
| GPRS модул | 12 лв. |
| Соларен панел | 37 лв. |
| Модул с MPPT контролер | 17 лв. |
| Акумулаторна батерия | 54 лв. |
| Прекъсвач | 2 лв. |
| Микроконтролер | 4 лв. |
| Импулсни регулятори | 8 лв. |
| Пасивни компоненти, кабели, съединители | 18 лв. |
| Кутия за електрическо табло | 34 лв. |
| Крак за маса | 15 лв. |
| Основа за чадър | 16 лв. |
| Метални профили | 8 лв. |
| Доставки | 50 лв. |
| ОБЩО: | 334 лв. |

Таблица 6.1. Цена на крайния продукт

В изчислената цена не е включен интелектуалният труд за разработката. При реално внедряване на системата, цената ще се повиши от: използването на по-мощен соларен панел, използването на по-качествени акумулаторни батерии с по-голям капацитет, подобряването на стойката на пътния знак; разходите за хостинг на уеб сървъра и др. Въпреки това продукта е значително по-евтини от повечето налични подобни разработки на пазара, които обикновено са от порядъка на над \$2000.

Заключение

В настоящата дипломна работа са разгледани технологии за визуализация на информация чрез светодиодни матрици, клетъчни технологии и протоколи за предаване на интернет пакети, особености на фотоволтаичните панели и подходящите за тях батерии и контролери на зареждането. Проектирана е електрическа схема на микроконтролерно устройство за управление на светодиодна матрица и GPRS модул. Осигурено е захранване на устройството чрез акумулаторна батерия и е осигурено зареждане на батерията чрез фотоволтаичен панел. Проектиран е управляващ код на микроконтролера за свързване с уеб сървър през GPRS мрежа и за визуализация на пътни знаци върху светодиодната матрица. Разработен е компютърен софтуер с графичен интерфейс за управление на указанията на пътните знаци. Разработен е уеб сървър за осъществяване на връзки с управляващите приложения и пътните знаци и е проектиран протокол за комуникация между различните програмни продукти. Изработена е печатна платка на микроконтролерното устройство. Създаден е работоспособен прототип на устройството и е изчислена цената на крайния продукт.

Основните идеи за бъдещото развитие на този проект са:

- **Добавяне на криптиране и валидиране на източника за връзката между уеб сървъра и пътните знаци** - това може да стане сравнително лесно чрез подмяна на GPRS модула на устройството (SIM800L) с еквивалентния по управление SIM800C. Той поддържа различни протоколи за защита на GPRS връзката като например SSL и TLS;
- **Добавяне на допълнителни пътни знаци, които микроконтролерното устройство може да изобразява** - възможно е да се имплементира и пълен графичен редактор, който да позволява изобразяването на произволни изображения върху пътните знаци;
- **Проектиране на персонализирани схеми, които да заменят използваните модули;**
- **Добавяне на методи за достъп до различна информация според потребителския профил** - в уеб сървъра вече съществува метод за разграничаване на различни потребителски профили. Възможно е да се ограничи достъпът на всеки акаунт до предварително определен набор от пътни знаци, които той има право да управлява. Това позволява групирането на пътни знаци според някакъв признак, като например физическо разположение или пък отговорен управляващ орган;
- **Добавяне на методи за сортиране на пътни знаци** - при увеличаването на броя пуснати в употреба пътни знаци ще възникне

нуждата от метод за тяхното сортиране и филтриране на базата на определени характеристики;

- **Добавяне на възможности за автоматична промяна на указанията на пътните знаци** - възможно е към устройствата да бъдат добавени различни сензори (например за влага, температура и др.), чрез които автоматично да се променят указанията на пътните знаци. Тези сензори могат да отчитат промени в метеорологичните условия и например да намаляват ограничението на скоростта при дъжд или температури под 0°C;
- **Оптимизиране на консумацията на светодиодната матрица** - това може да стане чрез сензор за светлина (или по-евтино - чрез следене на деня и часа). Информацията от него може да бъде използвана за автоматично регулиране на яркостта на матрицата. По този начин през нощта и при липса на слънце консумацията ще бъде намалена;
- **Добавяне на допълнителни мерки за предпазване на устройството** - например чрез добавяне на бушон за защита при токов удар, схема за предпазване от обратен поляритет на захранването и т.н.;
- **Добавяне на механизъм за поддържане на връзката между уеб сървъра и пътните знаци** - например чрез изпращане на периодични KeepAlive съобщения от уеб сървъра към пътните знаци.

Използвана литература

- [1]http://research.bfu.bg:8080/jspui/bitstream/123456789/529/1/BFU_2013_T_XXI_X_Popova_Letskovska_Seymenliyski.pdf
- [2]<https://www.maximintegrated.com/en/design/technical-documents/app-notes/1/1193.html>
- [3]<https://www.geeksforgeeks.org/shift-registers-in-digital-logic/>
- [4]http://www.batsocks.co.uk/readme/art_bcm_1.htm
- [5]<http://www.rotormind.com/blog/2011/Fast-Modulation-A-New-Kind-of-PWM/>
- [6]<https://medium.com/@tiemenwaterreus/4-bit-angle-modulating-16-leds-using-arduino-and-shift-registers-8b2b738d4ced>
- [7]https://en.wikipedia.org/wiki/Cellular_network
- [8]<http://www.telecomabc.com/c/cellular.html>
- [9]https://www.researchgate.net/figure/COMPARISON-OF-MOBILE-TECHNOLOGIES_tbl1_318673817
- [10]<https://www.semanticscholar.org/paper/Evolution-of-Mobile-Wireless-Communication-to-5G-as-sharma/ee92210c6be1dca049f80b898ff03713b402c74f/figure/1>
- [11]<https://danielmiessler.com/study/cellular/>
- [12]https://en.wikipedia.org/wiki/General_Packet_Radio_Service
- [13]<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.699.7280&rep=rep1&type=pdf>
- [14]<https://security.stackexchange.com/questions/21363/how-secure-is-http-https-over-3g?rq=1>
- [15]https://re.jrc.ec.europa.eu/pvg_tools/en/tools.html
- [16]https://en.wikipedia.org/wiki/Solar_cell
- [17]https://en.wikipedia.org/wiki/Solar_panel
- [18]<https://greencoast.org/best-batteries-for-solar/>
- [19]<https://www.wholesalesolar.com/blog/lead-acid-vs-lithium-batteries/>
- [20]<https://www.solartechnology.co.uk/support-centre/calculating-your-solar-requirements>
- [21]https://en.wikipedia.org/wiki/Maximum_power_point_tracking
- [22]http://www.leonics.com/support/article2_14j/articles2_14j_en.php
- [23]<https://www.electroschematics.com/maximizing-solar-panel-efficiency-and-output-power/>
- [24][https://electronics.stackexchange.com/questions/376751/is-a-mppt-controller-basically-a-buck-converter-with-a-feedback-geared-towards-p](https://electronics.stackexchange.com/questions/376751/is-a-mppt-controller-basically-a-buck-converter-with-a-feedback-gearred-towards-p)
- [25]<https://www.solar-electric.com/learning-center/mppt-solar-charge-controllers.html/>
- [26]<https://learn.sparkfun.com/tutorials/rgb-panel-hookup-guide/all>
- [27]<https://www.sparkfun.com/sparkx/blog/2650>

- [28]<https://components101.com/microcontrollers/atmega328p-pinout-features-data-sheet>
- [29]<https://www.theengineeringprojects.com/2017/08/introduction-to-atmega328.html>
- [30]<https://www.electroschematics.com/build-arduino-bootload-atmega-microcontroller-part-1/>
- [31]<https://www.instructables.com/id/Bootload-an-ATmega328/>
- [32]<https://electronics.stackexchange.com/questions/28897/how-to-choose-value-of-resistor-in-voltage-divider>
- [33]<https://www.electroschematics.com/introducingsim800/>
- [34]https://exploreembedded.com/wiki/Setting_up_GPRS_with_SIM800L
- [35]<https://lastminuteengineers.com/sim800l-gsm-module-arduino-tutorial/>
- [36]<http://www.mbedgeek.com/2019/01/power-supply-sim800l-core-board.html>
- [37]https://www.youtube.com/watch?v=yjvV_yBvZdw

Приложения

- {1}<https://cdn-learn.adafruit.com/downloads/pdf/32x16-32x32-rgb-led-matrix.pdf>
- {2}http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf
- {3}https://img.filipeflop.com/files/download/Datasheet_SIM800L.pdf
- {4}https://www.elecrow.com/wiki/images/2/20/SIM800_Series_AT_Command_Manual_V1.09.pdf
- {5}https://simcom.ee/documents/SIM800x/SIM800%20Series_SSL_Application%20Note_V1.02.pdf
- {6}<https://simcom.ee/documents/SIM800L/SIM800L%20SPEC170914.pdf>
- {7}<http://www.ti.com/lit/ds/symlink/lm2596.pdf>
- {8}<https://vikiwat.com/userfiles/productimages/6503/files/solare-panel-10w-12v-058a-lx-10m-0.pdf>
- {9}https://drive.google.com/file/d/0B1jlW3s_UNY5bjdHcmNWQ09vS28/view
- {10}<http://www.ti.com/lit/ds/symlink/bq24650.pdf>
- {11}<https://github.com/IliyanAntov/TrafficSign>
- {12}<https://github.com/vshymanskyy/TinyGSM>
- {13}<https://github.com/adafruit/RGB-matrix-Panel>
- {14}<https://docs.python.org/3/library/socket.html>
- {15}<https://docs.python.org/3/library/ssl.html>
- {16}<https://pyyaml.org/wiki/PyYAMLDocumentation>
- {17}<https://docs.python.org/3/library/hashlib.html>
- {18}<https://docs.python.org/3/library/select.html>
- {19}<https://docs.python.org/3/library/struct.html>
- {20}<https://docs.python.org/3/library/threading.html>
- {21}<https://docs.python.org/3/library/os.html>
- {22}<https://doc.qt.io/qtforpython/>

Съдържание

Първа глава

| | |
|---|-----------|
| 1.1. Средства за визуализация на информация | 7 |
| 1.1.1. Видове дисплеи | 7 |
| 1.1.1.1. Течнокристален дисплей (Liquid Crystal Display - LCD) | 7 |
| 1.1.1.2. Дисплей с органични светодиоди (Organic Light-Emitting Diode - OLED) | 8 |
| 1.1.1.3. Дисплей с флип дискове (Flip-Disc Display) | 9 |
| 1.1.1.4. Светодиоден дисплей (Light Emitting Diode Display) | 10 |
| 1.1.2. Управление на матрици с голям брой светодиоди | 11 |
| 1.1.3. Постигане на различни цветове в големи светодиодни матрици | 13 |
| 1.2. Клетъчни технологии и протоколи | 15 |
| 1.2.1. Особености на клетъчната свързаност | 15 |
| 1.2.2. Сравнение между различни видове клетъчни технологии | 19 |
| 1.2.3. General Packet Radio Service технология | 21 |
| 1.3. Фотоволтаично захранване | 24 |
| 1.3.1. Особености на фотоволтаичния панел | 24 |
| 1.3.2. Акумулаторни батерии за фотоволтаично захранване | 26 |
| 1.3.3. Maximum Power Point Tracking контролери | 28 |
| 1.4. Съществуващи решения и реализации | 31 |

Втора глава

| | |
|--|-----------|
| 2.1. Функционални и електрически изисквания към системата | 33 |
| 2.1.1. Функционални изисквания към устройството | 33 |
| 2.1.2. Софтуерни функционални изисквания към системата | 33 |
| 2.1.3. Електрически изисквания към устройството | 34 |
| 2.2. Проектиране на блоковата схема на устройството | 35 |
| 2.2.1. Блок "Визуализация" | 36 |
| 2.2.2. Блок "Управление" | 37 |
| 2.2.3. Блок "Комуникация" | 38 |
| 2.2.4. Блок "Захранване" | 39 |
| 2.2.5. Блок "Зареждане" | 40 |

Трета глава

| | |
|--|-----------|
| 3.1. Проектиране на принципните електрически схеми на блоковете на устройството | 41 |
| 3.1.1. Принципна електрическа схема на блок "Визуализация" | 41 |
| 3.1.2. Принципна електрическа схема на блок "Управление" | 46 |
| 3.1.3. Принципна електрическа схема на блок "Комуникация" | 50 |
| 3.1.4. Принципна електрическа схема на блок "Захранване" | 54 |
| 3.1.4.1. Акумулаторна батерия и прекъсвач | 55 |
| 3.1.4.2. Схема с импулсен регулатор на напрежение LM2596S-ADJ за захранване на блок "Комуникация" | 55 |
| 3.1.4.3. Схема с импулсен регулатор на напрежение LM2596S-5 за захранване на блокове "Управление" и "Визуализация" | 57 |
| 3.1.5. Принципна електрическа схема на блок "Зареждане" | 58 |
| 3.1.5.1. Фотоволтаичен панел | 58 |
| 3.1.5.2. Модул с MPPT контролер | 59 |
| 3.2. Проектиране на пълната принципна електрическа схема на устройството | 60 |

Четвърта глава

| | |
|---|-----------|
| 4.1. Проектиране на алгоритми за осъществяване на свързаност между различните софтуерни продукти | 62 |
| 4.1.1. Осъществяване на връзка между управляващо приложение и уеб сървър | 62 |
| 4.1.2. Осъществяване на връзка между уеб сървър и пътен знак | 63 |
| 4.1.3. Процес на обмен на информация | 64 |
| 4.1.4. Протокол за комуникация | 66 |
| 4.2. Проектиране на софтуер за микроконтролерно устройство | 68 |
| 4.2.1. Използвани инструменти за разработка | 68 |
| 4.2.2. Използвани библиотеки | 68 |
| 4.2.3. Блокова схема на алгоритъма | 72 |
| 4.2.4. Функции и променливи на програмата | 74 |
| 4.3. Проектиране на софтуер за управляващ уеб сървър | 78 |
| 4.3.1. Използвани инструменти за разработка | 78 |
| 4.3.2. Използвани библиотеки | 78 |
| 4.3.3. Структура на хранилището на уеб сървъра | 79 |
| 4.3.4. Блокова схема на алгоритъма | 81 |
| 4.3.5. Класове, методи и променливи на програмата | 83 |

| | |
|--|-----------|
| 4.4. Проектиране на компютърен софтуер за управление на пътни знаци | 89 |
| 4.4.1. Използвани инструменти за разработка | 89 |
| 4.4.2. Използвани библиотеки | 89 |
| 4.4.3. Структура на хранилището на управляващото приложение | 90 |
| 4.4.4. Блокова схема на алгоритъма | 91 |
| 4.4.5. Класове, методи и променливи на програмата | 93 |
| 4.4.5.1. Графичен интерфейс и управляваща логика на диалоговия прозорец за влизане в системата (Login Dialog) | 95 |
| 4.4.5.2. Графичен интерфейс и управляваща логика на основния прозорец на приложението (Main Window) | 96 |
| 4.4.5.3. Графичен интерфейс и управляваща логика на диалоговия прозорец за промяна на псевдонима на пътен знак (Set Alias Dialog) | 98 |
| 4.4.5.4. Графичен интерфейс и управляваща логика на диалоговия прозорец за изобразяване на детайлна информация за избран пътен знак (Details Dialog) | 99 |
| 4.4.5.5. Графичен интерфейс и управляваща логика на диалоговия прозорец за изобразяване на ограничение на скоростта върху избран пътен знак (Set Speed Limit Dialog) | 100 |
| 4.4.5.6. Графичен интерфейс и управляваща логика на диалоговия прозорец за изобразяване на предварителен изглед на пътен знак (Traffic Sign Preview Dialog) | 101 |
| 4.4.5.7. Графичен интерфейс и управляваща логика на диалоговия прозорец за изобразяване на предупреждение върху избран пътен знак (Set Warning Dialog) | 102 |

Пета глава

| | |
|---|------------|
| 5.1. Принципна електрическа схема на печатната платка на устройството | 105 |
| 5.2. Особености на печатната платка на устройството | 107 |
| 5.3. Разположение на елементите върху печатната платка на устройството | 108 |
| 5.3.1. Съображения при разполагането на елементите върху печатната платка | 108 |
| 5.3.2. Разполагане на елементите върху печатната платка | 109 |
| 5.4. Опроводяване на печатната платка на устройството | 111 |
| 5.4.1. Съображения при опроводяването на печатната платка | 111 |
| 5.4.2. Опроводяване на печатната платка | 112 |

Шеста глава

| | |
|---|------------|
| 6.1. Поръчване, насищане и оживяване на печатна платка | 115 |
| 6.2. Разработване на прототип на пътен знак | 117 |
| 6.3. Изчисление на цената на продукта | 127 |