



**СОФИЙСКА ПРОФЕСИОНАЛНА ГИМНАЗИЯ ПО ЕЛЕКТРОНИКА**

**„ДЖОН АТАНАСОВ“**

ул. „Райко Алексиев“ № 48 е – mail [spgelectron@yahoo.co.uk](mailto:spgelectron@yahoo.co.uk); spge – bg.com

**ДЪРЖАВЕН ИЗПИТ ЗА ПРИДОБИВАНЕ НА ТРЕТА СТЕПЕН НА  
ПРОФЕСИОНАЛНА КВАЛИФИКАЦИЯ**

**по професията код 523050 „Техник на компютърни системи“**

**по специалността код 5230501 „Компютърна техника и технологии“**

**ДИПЛОМЕН ПРОЕКТ**

**НА ТЕМА: СЪРВЪРИ И СЪРВЪРНИ КОНФИГУРАЦИИ. ИЗИСКВАНИЯ,  
ХАРАКТЕРИСТИКИ И РЕШЕНИЯ**

**ПРАКТИКА: ПРОЕКТИРАНЕ НА ЕФЕКТИВНА СХЕМА ЗА ПОДДЪРЖАНЕ И  
РЕГУЛИРАНЕ НА ОСНОВНИТЕ ХАРАКТЕРИСТИКИ В СЪРВЪРНО ПОМЕЩЕНИЕ  
ЧРЕЗ МИКРОКОНТРОЛЕРИ /ARDUINO/, ПРОГРАМИРАНИ НА C++.**

**ИЗГОТВИЛ: НИКОЛА ГЕДЕФАУ АДМАСУ УЧЕНИК ОТ 12<sup>Г</sup> КЛАС**

**Дипломант:**

*/Никола Адмасу/*

**Научен ръководител:**

*/инж. Цветана Пунева/*

**София, 2023г.**

## СЪДЪРЖАНИЕ

---

УВОД.....стр. 3

**I. ПЪРВА ЧАСТ - ОСНОВНИ ИЗИСКВАНИЯ ПРИ АСЕМБЛИРАНЕ НА СЪРВЪРНИ  
КОНФИГУРАЦИИ.....стр.5 - 35**

1.Сървъри/Server – определение, изисквания при конфигуриране и основни  
характеристики.....стр.5 – 15

2.Конфигуриране на дискове.....стр.15 – 15

3.RAID масиви.....стр.15 – 22

II. Най – често използвани сървъри – функционалности и изисквания.....стр.23 – 30

III. ОСНОВНИ ИЗИСКВАНИЯ КЪМ СЪРВЪРНИТЕ ПОМЕЩЕНИЯ.....стр.30– 35

**II. ВТОРА ЧАСТ - ПРОЕКТИРАНЕ НА ЕФЕКТИВНА СХЕМА ЗА ПОДРЕЖДАНЕ И  
РЕГУЛИРАНЕ НА ОСНОВНИТЕ ХАРАКТЕРИСТИКИ В СЪРВЪРНО ПОМЕЩЕНИЕ  
ЧРЕЗ МИКРОКОНТРОЛЕРИ /ARDUINO/. ПРОГРАМИРАНИ НА C++.....стр.36– 51**

1.Идейно предложение.....стр.36 – 36

2. Техническо решение.....стр.37 – 51

ИЗВОДИ, ОБОБЩЕНИЕ И ЗАКЛЮЧЕНИЕ.....стр.51 – 54

ИЗТОЧНИЦИ.....стр.54

## УВОД

Сървърите са обслужващи мрежите устройства. Отдавна отмина времето, когато файл сървърът беше единственият представител на този клас оборудване. Днес съществува голямо разнообразие от сървъри. Зад всяка една функция на мрежата или предлагана услуга стои съответен сървър. Разпределението на дейностите в мрежата значително облекчава работата ѝ и я прави по - ефективна. Сървърите са програмни или програмно -апаратни комплекси. Мрежовият сървър и операционната система работят като едно цяло.

Независимо колко е мощен сървърът, той е неизползваем без операционната система, която да се възползва от физическите му ресурси. Сървърът е компютър, който работи съвместно с други компютри в компютърна мрежа, като се различава от тях по това, че за работата му не е необходимо човешко участие.

В мрежите сървърът предоставя услуги към останалите компютри, наричани клиенти. Терминът „сървър“ е пряко свързан с архитектурата клиент - сървър. Сървърът е компютърна програма, която предоставя услуги на други програми, наречени в този контекст клиентски софтуер (Client). Сървърът стартира като услуга, която обслужва заявки на други програми („клиенти“), които могат, а могат и да не бъдат стартирани на същия компютър.

Сървърът е също така компютър, стартиращ сървърен софтуер и предоставящ една или повече услуги (като например хост)на други компютри в същата мрежа. В повечето случаи хардуерните изисквания към този компютър са по-високи от изискванията към хардуера на стандартния настолен компютър, който не функционира като сървър. В допълнение, може да има специални изисквания за архивиране, защиты и др.

Днес този термин се използва най-вече за завършени компютърни системи. Една такава софтуерна/хардуерна система представлява софтуерна услуга, стартираща на специално предназначен за нея компютър. Това са например сървър база данни, файлов сървър, mail сървър или print сървър. Linux или [GNU]/Linux е общото название на всички операционни системи, основаващи се на ядрото „Linux“ (Linux kernel) и системните инструменти и библиотеки, обикновено от проекта GNU. Повечето от тези операционни системи се наричат Linux дистрибуции, но Фондацията за свободен софтуер използва наименованието GNU/Linux, за да подчертае значимостта на GNU софтуера, което предизвиква известни противоречия.

Linux е флагман и един от най-известните представители на свободния софтуер и софтуера с отворен код (F(L)OSS – free software, libre software, and open-source software). Проектът и движение GNU, чиято цел е създаване на нова операционна система свободен софтуер, е основан от Ричард Столман през 1984 г. Системата съдържа голям брой инструменти и програми, например компилатори, текстови редактори и сървъри. Софтуерът се разпространява с лиценза GNU GPL, което гарантира бъдещата му свободна достъпност. С Linux, GNU става напълно работеща операционна система и това спомага за бързото ѝ разпространение.

Фактът, че повечето от най-бързите суперкомпютри в света, работещи на Linux, може да се отдаде на скоростта му. Linux има репутацията на бърз и гладък, докато за Windows 10 е известно, че с времето става бавен и бавен. Linux работи по-бързо от Windows 8.1 и Windows 10, заедно с модерна работна среда и качества на операционната система, докато Windows е бавен за по - стария хардуер. Говорейки за основните възможности на операционната система като планиране на нишки, управление на паметта, обработка на входове и изходи, управление на файлова система и основни инструменти, като цяло Linux превъзхожда Windows. Има много причини Linux да бъде по-бърз от Windows:

**Целта на дипломния ми проект е да** представя основните изисквания при асемблиране на сървърни конфигурации, да представя основните видове сървъри, както и изискванията при проектиране и архитектурни изисквания при оборудване на сървърни помещения. Да програмирам измервателен модул, на основните параметри които трябва да се поддържат в сървърните помещения. Микроконтролера Arduino да се програмира на C++ за измерване на температура, влажност и сеизмична активност.

За постигане на тази цел си поставям **задачите:**

- Представяне на основните изисквания при конфигуриране на сървъри;
- Представяне на основните видове сървъри с техните функционалности и изисквания;
- Да разгледам основните изисквания и архитектурни решения на сървърни помещения;
- Да проектирам измервателен модул на основните параметри, които трябва да се поддържат във всяко сървърно помещение.

### СЪРВЪРИ И СЪРВЪРНИ КОНФИГУРАЦИИ. ИЗИСКВАНИЯ, ХАРАКТЕРИСТИКИ И РЕШЕНИЯ

#### 1. ОСНОВНИ ИЗИСКВАНИЯ ПРИ АСЕМБЛИРАНЕ НА СЪРВЪРНИ КОНФИГУРАЦИИ:

##### 1. Сървъри/Server – определение, изисквания при конфигуриране и основни характеристики

**1.1. Сървър определение** – обслужващо мрежите устройство. Зад всяка функция на мрежа или предлагана мрежова услуга, стои сървър. Сървърът е компютър, който работи съвместно с други компютри, като се различава по това, че за работата му не е необходимо човешко участие. Предназначен е да обработва заявки, да предоставя данни и услуги към останалите компютри/клиенти, по интернет или в локална мрежа. Той е или софтуерно, или хардуерно устройство. Наименованието идва от архитектурата клиент-сървър. Някои сървъри са ангажирани с конкретна задача – специализирани, но повечето съвременни сървъри са споделени – поемат отговорността (за електронна поща, DNS, FTP и за множество уебсайтове в случай на уеб сървър). Разпределението на дейностите в мрежата, облекчава работата ѝ и я прави по-ефективна.



Фиг. 1

##### 1.2. Компоненти на сървъра

► Хардуерни компоненти – изискванията към хардуера зависят от предназначението на сървъра и софтуера му. Обикновено се състоят от: системна платка/motherboard, един или повече процесори, RAM, памет за съхранение (обикновено харддиск, в някои случаи SSD), мрежов

интерфейс, шаси за монтиране в шкаф и захранване/power supply. Повечето сървърни хардуерни устройства поддържат извънлентово управление чрез специален мрежов порт. То позволява управление и наблюдение на сървъра. Системите за извънлентово управление могат да се използват за дистанционно включване или изключване на сървъра, за инсталиране на операционна система и наблюдение на състоянието. Тъй като достъпът до сървърите масово се осъществява по мрежа, голяма част от тях работят без надзор, без монитор или входно устройство, аудиоапаратура и USB интерфейси. Много нямат графичен потребителски интерфейс (GUI). Дистанционното управление може да се извършва чрез различни методи, като някои от тях са: Microsoft Management Console (MMC), PowerShell и т.н.



Фиг.2

► Софтуер – сървърът се нуждае най-малко от два софтуерни компонента – операционна система и приложения. Операционната система действа като платформа за изпълнение на сървърното приложение. Тя осигурява достъп до основните хардуерни ресурси и дава възможност за мрежова свързаност. Също така осигурява средствата за комуникация на клиентите със сървърното приложение. Преобладаващите операционни системи сред сървърите са подобни на UNIX дистрибуции с отворен код, като – базирани на Linux и FreeBSD. Windows Server също имат значителен дял. Собствени операционни системи като z/OS и macOS Server също се използват, но са с по-малък дял. Специализираните сървърно ориентирани операционни системи обикновено имат характеристики като: графичен потребителски интерфейс, който не е задължителен; възможност за преконфигуриране и актуализиране на хардуера и софтуера до известна степен без рестартиране; усъвършенствани средства за архивиране, позволяващи редовно и често онлайн архивиране на критични данни; прехвърляне на данни между различни токове/volumes или устройства; гъвкави и усъвършенствани възможности за работа в мрежа; възможности за

автоматизация и др.; строга сигурност на системата с усъвършенствана защита на потребителите, ресурсите, данните и паметта; усъвършенствано откриване и предупреждаване за състояния като прегряване, отказ на процесора и диска. Днес много настолни и сървърни операционни системи споделят сходни бази данни, различаващи се най-вече по конфигурацията.

### 1.3. Архитектура на сървъра

- Паралелна компютърна архитектура. Операционната система работеща с четири и повече процесора ще балансира товара на задачата така, че да се постига добра производителност на сървъра;
- Код за коригиране на грешките /ECC/ подсилва сигурността на сървъра и целостта на данните;
- Кеширането на данните в паметта оптимизира производителността на сървъра;
- Два мрежови адаптера позволяват сегментиране товара на клиентите на сървъра. Софтуерът обединява двата порта в един. Вторият адаптер служи за осигуряване на по-висока честотна лента или като резервен;
- Смяна на дисковете по време на работа. Използване на технологията RAID Level 5;
- Връзка към устройства за бързо съхранение на данни;
- Непрекъсваеми по време на работа захранвания с двойни мрежови кабели и два допълнителни вентилатора.

**Уеб сървър/Web server** – компютърна програма, която обслужва заявени HTML страници или файлове. В този случай уеб браузърът действа като клиент – Internet Explorer, Chrome, Firefox, Opera или Safari. Използват се за много задачи в допълнение към доставянето на прост текст и изображения, например за качване и архивиране на файлове онлайн чрез услуга за съхранение в облак или онлайн услуга за архивиране.

**Сървър за приложения** – програма в компютър в разпределена мрежа, която осигурява бизнес логиката за дадена програма за приложения. Правят данните и сървърната част на приложенията от типа клиент/сървър достъпни за клиентите. При сървъра за приложения базата данни се намира на сървъра, а на клиентският компютър чрез форми се попълват данни или чрез заявки се извлича информация.

**Пощенски сървър** – управлява електронните съобщения между потребителите на мрежата. Получава входящи имейли от местни потребители – хора в рамките на един и

същи домейн – и отдалечени изпращачи и препраща изходящи имейли за доставка.

**Файлов сървър/File server** – компютър, който отговаря за централното съхранение и управление на файлове с данни, така че други компютри в същата мрежа да имат достъп до тях. Синхронизира достъпа до общите ресурси. **Упълномощен сървър/Proxy сървър** – софтуер, който действа като посредник между крайно устройство, например компютър, и друг сървър, от който потребителят или клиентът заявява услуга.

**Виртуален сървър/Virtual server** – програма, работеща на споделен сървър, която е конфигурирана по такъв начин, че за всеки потребител изглежда, че има пълен контрол върху сървъра.

**Блейд сървър/Blade server** – сървърно шаси, в което са разположени множество тънки модулни електронни платки, известни като сървърни блейдове. Всеки блейд е самостоятелен сървър, често посветен на едно-единствено приложение.

**Сървър за политики/Policy server** – компонент за сигурност на мрежа, базирана на политики, който предоставя услуги за оторизация и улеснява проследяването и контрола на файловете.

**Сървър за бази данни/Database server** – този сървър отговаря за хостването на една или повече бази данни. Клиентските приложения изпълняват заявки към базата данни, които извличат данни или записват данни в базата данни, която се хоства на сървъра.

**Сървър за печат/Print server** – този сървър осигурява на потребителите достъп до един или повече мрежови принтери – или устройства за печат, както ги наричат някои доставчици на сървъри. Сървърът за печат действа като опашка за заданията за печат, които потребителите подават. Някои сървъри за печат могат да приоритизират задачите в опашката за печат въз основа на типа на задачата или на това кой е подал задачата за печат.

**FTP сървър** – FTP сървърите преместват файлове чрез инструменти на протокола за прехвърляне на файлове. FTP сървърите са достъпни от разстояние с помощта на FTP клиентски програми, които се свързват с файловия дял на сървъра чрез вградените FTP възможности на сървъра или със специална FTP сървърна програма.

**Сървър за идентичност/User identity server** – сървърите за идентичност поддържат влизания и роли за сигурност за оторизирани потребители.

**Сървър за IP адресиране/DNS/Domain Name Server** – извършва услуги по управление на IP адреси



**Сървър, осъществяващ защитната стена** – сървърът може да има три адаптера:

- За връзка с Интернет;
- Създава защитна стена и пълен контрол върху маршрутизирането на постъпващите пакети;
- Адаптер за локална мрежа, който дава възможност на потребителите да се включват в другикорпоративни локални мрежи;
- Адаптер за защитааната LAN.

**Факс сървър** – управлява трафика на факсове към и от мрежата. Може чрез общественатателефонна мрежа да предава факсове към факс апарати.

**Сървъри за директорийни услуги** – позволяват на потребителите да локализируют, съхраняват изащитават информацията в мрежата.

**Други сървъри** – колкото функции изпълнява мрежата, толкова вида сървъри могат да бъдатсъздадени. Това е въпрос на проектиране, оптимизация на структурата и естествено пари.

**Общи видове сървъри** – докато някои специализирани сървъри се фокусират върху една функция, като например сървър за печат или сървър за бази данни, някои реализации използват единсървър за множество цели. Голяма мрежа с общо предназначение, която поддържа средна по големина компания, вероятно разполага с няколко вида сървъри, включително: стотици специализирани типове сървъри поддържат компютърни мрежи. Освен обичайните корпоративни типове, домашните потребители често си взаимодействат със сървъри за онлайн игри, чат сървъри, сървъри за аудио и видео стрийминг и др. Някои сървъри съществуват с конкретна цел, но не е задължително да се взаимодейства с тях по някакъв значим начин. Пример за това са DNS сървъритеи прокси сървърите.

**Видове мрежови сървъри** – в много мрежи в интернет се използва мрежов модел клиент-сървър, който интегрира уебсайтове и комуникационни услуги. Алтернативният модел, наречен **peer-to-peerмрежа**, позволява на всички устройства в мрежата да функционират като сървър или клиент при необходимост. Равнопоставените мрежи предлагат по-голяма степен на поверителност, тъй като комуникацията между компютрите е тясно насочена. Въпреки това, отчасти поради ограниченията на честотната лента, повечето реализации на peer-to-peer мрежи не са достатъчно надеждни, за да поддържат големи скокове на трафика.

**Облачен сървър; Сървър за бази данни; Специализиран сървър; Услуга за име на домейн; Сървър за печат; Самостоятелен сървър**

**1.4.Изисквания към сървъра:**

Средства улесняващи инсталирането – софтуерни пакети, които предпазват от допускане на големи грешки при инсталиране и експлоатация на сървъра и оказват ценна помощ на мрежовия администратор.

Средства за наблюдение и контрол – продукти за визуализиране състоянието на сървъра. Те позволяват автоматизиране на отделни задачи, които претоварените системни администратори пренебрегват.

Запаметяване и защита на информацията – прилагат се различни техники за дублиране на информацията, без това да се отразява чувствително върху крайната цена на сървъра. Използват се три метода:

- Огледален диск – два диска използват един и същи контролер, като и двата записват напълно идентично копие на данните. Недостатък – при повреда на контролера информацията се губи;
- Дублиране на дискове – всеки от дублиращите дискове разполага със собствен контролер. Методът е по-скъп, но по надежден и осигурява по-голямо бързодействие;
- RAID – обобщено наименование на няколко метода за повишаване на надеждността. Данните се записват на няколко диска по определена схема.

Средства за отдалечен контрол/Remote access – много полезни, когато мрежовият администратор не е наблизо. Той може да се свърже по телефона и да получи информация за състоянието на мрежата, да тества или рестартира системата. Има и възможност, където при възникване на проблем, сървърът автоматично да потърси помощ по телефона.

Допълнителни осигуровки на сървъра – хардуер, който сам фиксира възникналите проблеми в сървъра; оперативна памет с автоматична корекция на грешките. Ако това не може да стане, повреденият чип се изолира и се сигнализира за събитието. Двойна осигуровка на захранването на сървъра. Включване на два UPS, като допълнителна осигуровка на захранването. Двете устройства работят в паралел; дублиране на сървъра (клъстериране) – две или повече машини се свързват чрез високоскоростна връзка и се прави абсолютно огледален образ на данните и паметта.

**1.5.Физически и виртуални сървъри** – на база тяхната реализация, се различават физически и виртуални сървъри.

Физически сървър – повечето сървъри са от този тип. Всеки физически сървър включва сървър с дънна платка, памет, процесор, мрежова връзка, твърд диск и операционна система за изпълнение на програми и приложения. Счита се за "bare-metal"/"чист сървър", тъй като хардуерът му се

използва директно от операционна система вместо от платформа за виртуализация.

Виртуален сървър – виртуално представяне на физически сървър. Подобно на физическия, виртуалният включва собствена операционна система и приложения. Те са отделени от всички други виртуални сървъри, които могат да работят на физическия сървър. Процесът на създаване на виртуални машини включва инсталиране на олекотен софтуерен компонент – хипервайзор, на физическия сървър. Задачата му е да позволи на физическия сървър да функционира като хост за виртуализация. Хостът за виртуализация предоставя хардуерните ресурси на физическия сървър – като процесорно време, памет, място за съхранение и мрежова честотна лента – на разположение на една или повече виртуални машини.

**1.6. Типове сървъри според големината/натоварването им** – сървърите могат да бъдат разделени и според големината и натоварването, на което са подложени:

Бизнес/Корпоративни сървъри – големите сървъри работят за дълги периоди, без прекъсване. Функционалността на този тип сървъри трябва да бъде непрекъсната, което прави надеждността и издръжливостта на хардуера ключови. Критично важните корпоративни сървъри трябва да са много устойчиви на грешки, да използват специализиран хардуер с ниска честота на повреда, за да се увеличи времето на работа. В системите се включат непрекъсваеми източници на захранване, за да се предпазят от прекъсване на електрозахранването. Сървърите обикновено включват хардуерно резервиране, като двойни захранвания, RAID дискови системи и ECC памет, както и обширно тестване и проверка на паметта преди стартиране на системата. Критичните компоненти могат да бъдат подменяни в горещо състояние/hot swapping, а за да се предпазят от прегряване, сървърите могат да имат по-мощни вентилатори или да използват водно охлаждане. Често срещано е да бъдат конфигурирани, включвани, изключвани или рестартирани от разстояние, като се използва извънлентово управление. Корпусите на сървърите обикновено са плоски и широки и са предназначени за монтаж в 19-инчови шкафове или в отворени шкафове. Тези типове сървъри се намират в специализирани центрове за данни. Шумът е малък проблем, но консумацията на енергия и отделянето на топлина могат да имат негативно влияние. За тази цел, сървърните помещения са оборудвани с климатични устройства.

Малки бизнес и домашни сървъри – всеки компютър може да действа като сървър с подходящия софтуер. Може да се изтегли програма за FTP сървър на настолен компютър, за споделяне на файлове между потребители в направената мрежа. Някои твърди дискове, работещи в мрежа, използват протокола на сървъра Network Attached Storage, за да позволят на различни компютри в домашната мрежа да имат достъп до споделен набор от файлове. Софтуерът на медийния сървър

Plex, позволява на потребителите да гледат цифрови медии на телевизори и устройства за развлечение, независимо дали данните се намират в облака или на локален компютър. За да работи компютър като сървър, той трябва постоянно да работи. Има риск за атаки при свързаност в интернет. Ако има голямо натоварване, компютърът може да не разполага с необходимите ресурси да извърши всички заявки.

Мобилни – за разлика от големите центрове за данни или сървъри в шкафове, мобилният сървър е предназначен за работа на пътя, като в спешни случаи, при бедствия или във временна среда, където традиционните сървъри са неприложими, поради изискванията им за хранване, размера и времето за пускане. Основните ползватели на технологията "сървър в движение" са мрежовите мениджъри, разработчиците на софтуер или бази данни, центрове за обучение, военните, правоприлагащите органи, криминалистите, групите за спешна помощ и организациите за услуги. За да се улесни преносимостта, в шасито са интегрирани функции като клавиатура, дисплей, батерия (непрекъсваемо хранване, за да се осигури резервиране на хранването в случай на повреда) и мишка.

**1.7.Работен режим на сървърите** – обикновено се използват за предоставяне на услуги, които са постоянно необходими и за това повечето никога не се изключват. Колкото по-голям е сървърът, толкова е по-важно работата му да не спира, тъй като повече хора разчитат/зависят на него. Когато се повредят могат да причинят проблеми на потребителите и компанията. За това се настройват, така че да са устойчиви на грешки. Тъй като времето за работа е критично важно за повечето сървъри, те работят непрекъснато. Понякога се изключват умишлено за планирана поддръжка, като в подобни ситуации, някои уеб сайтове и услуги уведомяват потребителите предварително. Сървърите могат да се изключат и

непреднамерено по време на нещо като DDoS атака. Когато даден уеб сървър сваля информация за постоянно или временно, е възможно все още да има достъп до нея, ако услуга на трета страна ги архивира. Wayback Machine е пример за уеб архиватор, който съхранява снимки на уеб страници и файлове, съхранявани на уеб сървъри.

Големите предприятия, които разполагат с множество сървъри, обикновено нямат локален достъп до тях с клавиатура и мишка, а чрез отдалечен достъп. Понякога тези сървъри са и виртуални машини, което означава, че на едно устройство за съхранение могат да бъдат разположени няколко сървъра. По този начин се спестява физическо пространство и пари.

**1.8.Начини на свързване на устройствата към сървъра** – при локална мрежа, сървърът се свързва с маршрутизатор, който се използва от всички останали компютри в мрежата. След като

се свържат към мрежата, другите компютри имат достъп до този сървър и неговите функции. При уеб сървър, потребител може да се свърже със сървъра, за да разглежда уебсайт, да търси и да комуникира с други потребители в мрежата. Интернет сървърът работи по същия начин като сървъра на локалната мрежа, но в по-голям мащаб. На сървъра се присвоява статичен IP адрес от InterNIC или от уеб хоста. Обикновено потребителите се свързват със сървъра, като използват името на домейна. Когато потребителите се свързват с името на домейна (например "starwars.com"), името автоматично се преобразува в IP адрес на сървъра от DNS резолвера.

**1.9.Физическо съхранение на сървърите** – в бизнес или корпоративна среда сървърът и другото мрежово оборудване се съхраняват в шкаф/т.н. рак. Тези помещения помагат да се изолират чувствителните компютри и оборудване от хора, които не трябва да имат достъп до тях. Сървърите, които са отдалечени или не се хостват на място, се намират в център за данни. При тези видове сървъри хардуерът се управлява от друга компания и се конфигурира дистанционно от потребителя или от компанията му.

За малките бизнес/домашните сървъри, не е необходимо специално помещение – може да се намира в някоя стаите, където живее или работи даден потребител. Проблемът би бил евентуалният шум.

**1.10.Съображения при избор на подходящ сървър** – при избора, трябва да се прецени важноста на функциите, въз основа на случаите на използване. Възможностите за сигурност също са важни и има доста функции за защита, откриване и възстановяване, които трябва да се разгледат. Ако сървърът ще разчита на вътрешна памет, изборът на типове дискове и капацитет е важен, тъй като може да окаже значително влияние върху входа/изхода (I/O) и устойчивостта. Много организации намаляват броя на физическите сървъри в своите центрове за данни, тъй като виртуализацията позволява на по-малко сървъри да се хостват повече работни натоварвания. Навлизането на облаците е променило броя на сървърите, които организацията трябва да хоства на място. Събирането на повече възможности в по-малко кутии, може да намали общите разходи, площта на центровете за данни и нуждите от енергия и охлаждане. Обаче хостването на повече работни натоварвания в по-малко кутии може да представлява и повишен риск за бизнеса, тъй като повече работни натоварвания ще бъдат засегнати, ако сървърът се повреди или трябва да бъде изключен за рутинна поддръжка. Контролният списък за поддръжка на сървъра трябва да обхваща физическите елементи, както и критичната конфигурация на системата.

**1.11.Изграждане на домашен сървър или такъв за нуждите на малка фирма** – в съвременното почти всички сървърни услуги може да се ползват чрез облак/cloud, или да се ползват сървъри

под наем. Но има причини, поради които изграждането на собствен сървър е предпочитано начинание. За много потребители, от най-голямо значение е контролът на данните. Големи корпорации като Google Drive твърдят, че не притежават данните, които потребителите качват в облака, но в действителност притежават лиценз за възпроизвеждане и модифициране на файловете на потребителите. Ползването на собствен сървър гарантира стриктен контрол на потребителя върху собствените му данни.

**1.12.Цена** – цената на хардуера е само едно от перата, за които трябва да се планира бюджет. Други важни разходи са – операционна система; сървърен софтуер; поддръжка на хардуера и софтуера; подмяна на захранването и охлаждане в случай на необходимост и т.н.

**1.13.Движение на цените** – благодарение на технологичната еволюция има драстичен спад на цените, не само на професионалните сървърни конфигурации, но и на потребителските системи. Всичко това прави достъпно изграждането на сървърна конфигурация за лични цели или за нуждите на малка фирма. На българския пазар от финансова гледна точка, за такива цели са още по-достъпни и вносните, втора употреба професионални компютърни конфигурации. От 1999 г. насам, цената на сървър се е понижила четирикратно в сравнение с пика през 1999 г., когато много RISC/Unix и IBM мейнфрейм машини са се продавали страшно добре. От 20 000-30 000 долара в края на XX век, днес цените на сървърните конфигурации за бизнес цели са към 5 000 долара. В подобни диапазони е и движението на цените на персоналните компютри.

#### **1.14.Глобална консумация на енергия, дължаща се на сървъри**

През 2010 г. центровете за данни (сървъри, охлаждане и друга електрическа инфраструктура) са употребили 1,1-1,5 % от електрическа енергия в света. Според една от оценките, общото потребление на енергия за информационни и комуникационни технологии спестява повече от 5 пъти въглеродния отпечатък в останалата част на икономиката чрез повишаване на ефективността. Глобалното потребление на енергия се увеличава поради нарастващото търсене на данни и широчина на честотната лента. Съветът за защита на природните ресурси (NRDC) посочва, че през 2013 г. центровете за данни са използвали 91 млрд. киловатчаса (kWh) електрическа енергия, което представлява 3 % от световното потребление на електроенергия. Групите за защита на околната среда поставят акцент върху въглеродните емисии на центровете за данни, тъй като те отделят 200 млн. метрични тона въглероден диоксид годишно.

## **2.Конфигуриране на дискове**

Най-разпространената реализация за конфигуриране на множество твърди дискове е чрез RAID масиви, която се предлага в редица стандартни и нестандартни конфигурации. Макар RAID масивите да са най-ефективния и разпространен начин за управление на множество дискови устройства, не е единствената възможност. Съществуват и други като:

JBOD – Just a Bunch Of Disks/Купчина дискове – множество твърди дискове, работещи като отделни независими такива. Не предлага никакво подобрение на производителността и няма никаква резервираност, но предимство е, че не се интересува от това, какви дискове влизат в него.

SPAN или BIG – метод за комбиниране на свободното пространство на няколко твърди диска от "JBOD" за създаване на обхванат том. Такова обединяване понякога се нарича също BIG/SPAN. SPAN или BIG обикновено е само разтеглен том, тъй като често съдържа несъвместими типове и размери на твърди дискове (препратка 2, 3 в приложение).

MAID – Massive Array of Idle Drives/Озромен масив от неактивни устройства – архитектура използваща стотици до хиляди твърди дискове за осигуряване на близко до линията съхранение на данни, предназначена предимно за приложения от типа "Write Once, Read Occasionally" (WORO), при които увеличената плътност на съхранение и намалената цена се търгуват за увеличена латентност и намалена резервираност.

### **3.RAID масиви**

**3.1.Същност на RAID масивите** – Redundant Array of Independent Disks – набор от дискови устройства, които се управляват от специален контролер, като компютъра ги счита за един независим диск с голям капацитет . Основната цел е подобряване на производителността и намаляване на риска от загуба на данни при възникване на технически проблеми или унищожаване на хард дисковете. Райд масивите функционират, като разпределят балансирано информацията върху определено количество дискове. Съществуват различни логически схеми за разпределяне на информацията и те са пряко свързани с производителността на райд масива, с дублирането и защитаването на данните, а или и двете едновременно. Различните дискове се свързват към така наречените райд хардуерни контролери.

**3.2.Техники за разпределение** – съществуват две техники за организиране данните в нива на HDD. Това са – data mirroring, data striping.

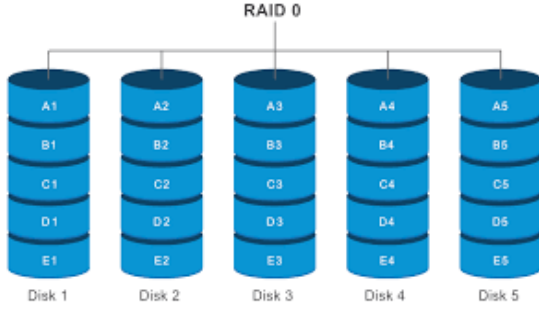
Data mirroring/Дублиране – информацията се записва огледално на 2 диска. При този вид има резервираност на данните при отпадане на единия хард диск. Това означава, че данните ще бъдат изцяло налични при възникване на технически проблем. Недостатъкът е, че ефективно използваното пространство е по-малко. Използват се 2 пъти повече носители и времето за запис е по-бавно, защото трябва да се запише на 2 места едновременно.

Data striping/Разпределяне – данните се разделят на части, които трябва да се запишат – четни на единия диск и нечетни – на другия. По този начин, те са равномерно разпределени из дисковия масив. Това е предпоставка за по-бързото им прочитане. Дисковото пространство се използва по-пълноценно, защото информацията не се копира на няколко диска едновременно, но за сметка на това се губи резервираността на данните.

*! Разпределянето на данните между самите дискове се извършва на различни нива – райд нива.*

**3.3.Стандартни/Основни RAID нива са** – RAID 0, RAID 1, RAID 2, RAID 3, RAID 4, RAID 5, RAID 6, RAID 7 - фиг.

**3.4.Комбинирани/Хибридни RAID нива са** – RAID 0+1, RAID 1+0, RAID 3+0, RAID 5+0, RAID 5+3 и т.н.

Стандартни RAID нива:	Схема:
<b>RAID 0:</b>  Минимум 2 устройства, няма защита на информацията.  При увеличаване на броя на дисковете в масива, расте и скоростта на запис/четене. Предлага най-добрата производителност, но не позволява защита на данните.	 <p>The diagram illustrates a RAID 0 configuration with five disks labeled Disk 1 through Disk 5. A horizontal line at the top represents the RAID 0 controller, with vertical lines connecting it to each disk. Each disk is represented as a vertical cylinder with five segments labeled A, B, C, D, and E from top to bottom. The data is striped across the disks: Disk 1 contains A1, B1, C1, D1, E1; Disk 2 contains A2, B2, C2, D2, E2; Disk 3 contains A3, B3, C3, D3, E3; Disk 4 contains A4, B4, C4, D4, E4; and Disk 5 contains A5, B5, C5, D5, E5.</p>



### RAID 1:

Минимум 2 устройства, опростена защита на данните.

Производителността при запис е същата както при единичен диск.

Недостатъкът на това ниво е, че ефективният свободен обем ще бъде двойно по-малък от сбора на инсталираните устройства.

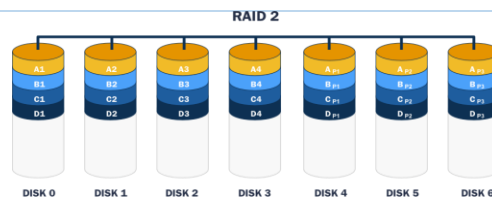


### RAID 2:

Това ниво не се използва от дълги години.

Препоръчват се поне 10 диска за съхранение на данни и още 4 за генерираните кодове. Това прави общ дисков масив от 14 устройства.

Информацията се разделя на блокове и се записва последователно на дисковете от масива.

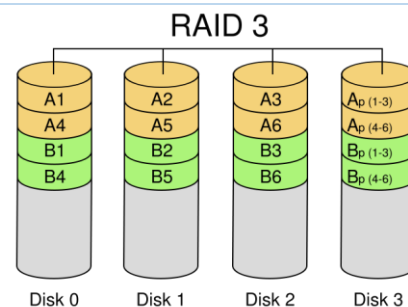


### RAID 3:

Принципът на действие е подобен на този на RAID 2.

Надеждността на съхранение на информация е същата като в предходното RAID 2.

Един от дисковете е заделен за съхранение на битове за четност.



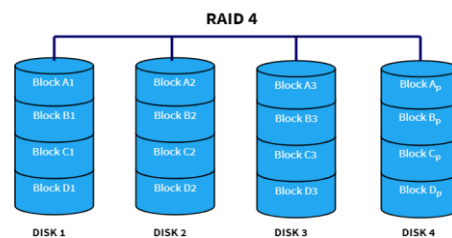
### RAID 4:

Изискват се поне 3 твърди диска с еднакъв капацитет

Разпределянето на информацията се осъществява като предходните две нива(2, 3), като отново е предвидено устройство за съхранение на кодовете за четност.

Едновременно обработване на две заявки е възможно.

Ефективността при това ниво се увеличава с добавяне на повече устройства към масива.



## RAID 5:

Необходими са минимум 3 диска.

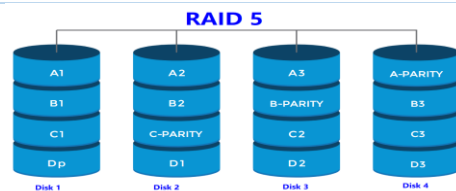
Основава се на разпределението на блокове от данни с кодове за четност.

Както при останалите нива, блоковете от данни се записват последователно в дисковете от масива, кодовете за четност се разпределят последователно по дисковете в масива.

Функционирането се запазва дори при отпадането на един от дисковете.

При отпадне на един от дисковете и бива заменен с друг, производителността на масива рязко спада през времето, което е необходимо за възстановяването на отпадналият диск.

По времето през което се възстановява изгубеният диск, масива е много податлив към отпадане на втори диск, като в този случай информацията се изгубва безвъзвратно.

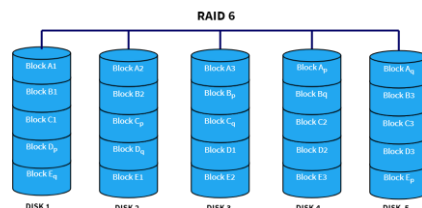


## RAID 6:

Минималният брой необходими устройства са четири.

подобен на RAID 5, но при него има двойно разпределяне на битовете за четност.

Нивото предлага по-голяма надеждност на съхраняваната информация за сметка на по-малко ефективно пространство и по-ниска производителност в сравнение с RAID 5.



Фиг.3

## Хибридни RAID нива (Комбинация от RAID нива):

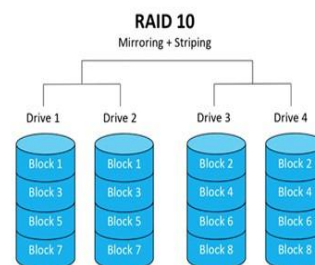
### RAID 1+0 (10):

Необходимостта от минимум 4 диска

Това ниво представлява комбинация от RAID 1 и RAID 0.

В този случай блоковете от данни се записват огледално (информацията се дублира) и едновременно с това се разпределят между двойка дискове.

## Схема:



Тук има по-голяма производителност и защита на данните от RAID 1, но цената му е много по-висока.

Масивът може да понесе множество загуби на устройства, стига едно огледално копие да не загуби всичките дискове.

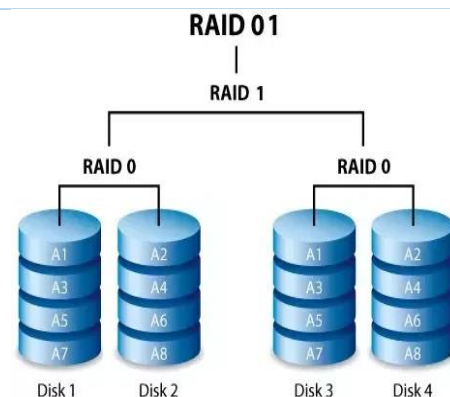
### RAID 0+1:

Отново минималното изискване е 4 дискови устройства

Тук схемата на функциониране е обратна на 1+0.

Първо информацията се разпределя последователно на блокове и след това те се дублират.

Рискът от загуба на информация е по-голям, отколкото при RAID 1+0.

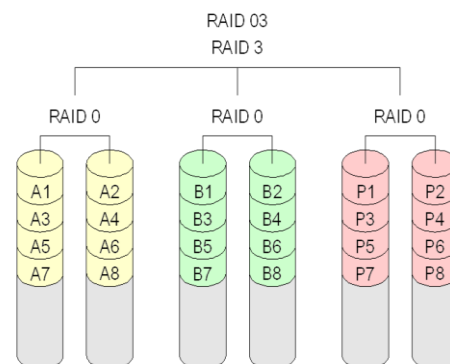


### RAID 0+3:

Нивото обединява начинът на работа на RAID 0 и RAID 3.

Използва се разделяне (striping) както при RAID 0, но при разпределение на блоковете по схемата на RAID 3.

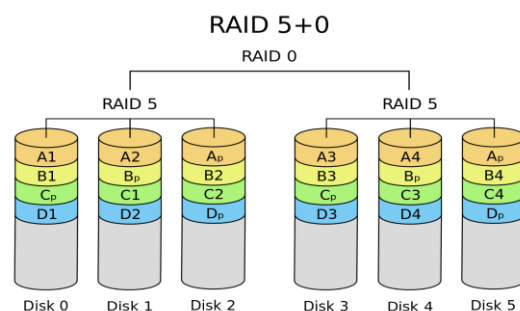
Така се увеличава надеждността на RAID 3, но респективно с това се повишава и цената.



### RAID 5+0:

Съчетание между разпределението на битовите за четност при RAID 5 със striping - а на RAID 0.

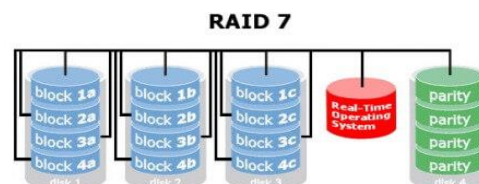
Подобрява се надеждността на RAID 5 без да се намалява защитата на данните.



### RAID 7:

Този RAID е запазена марка на Storage Computer Corporation.

Показателите на максималния трансфер при четене на файлове, независимо малки или големи, е изключително



висок. Начинът на работа обединява РАЙД 3 и РАЙД 4, но предлага много по-голяма производителност от тях. Той трудно се намира на пазара и държи висока цена.	
--	--

Фиг.4

**3.5.Типове райд масиви** – един RAID масив може да бъде създаден както чрез отделно хардуерно устройство, така и софтуерно като част от операционната система. Най-често хардуерните RAID устройства се наричат RAID контролери.

Хардуерен RAID масив – изгражда се с помощта на хардуерно устройство, наречено RAID контролер. Те представляват отделен микропроцесор със собствена памет, обединени на една обща платка, която се добавя към разширителите слотове на основния компютър или сървър. Модерните RAID контролери имат вграден кеш за увеличаване на бързодействието. Дисковете, които ще участват в масива се свързват към този RAID контролер. Най-често RAID контролерите са производство на дадена компания и разполагат със собственически затворен код. По-съвременните RAID контролери имат вграден кеш за увеличаване на бързодействието. Съдържанието на този кеш би било загубено при спиране на захранването на контролера. Поради тази причина, съдържанието на кеш паметта често е резервирано от BBU (Backup Battery Unit) или BBN (Battery Backup Module). Те позволяват на RAID контролера да запаzeti информацията, която все още не е записана на диска. Това устройство може да осигури резервно захранване на контролера, така че той да запази информацията до 72 часа без захранване. След като бъде възстановено основното захранване на системата, BBU частта ще запише съдържанието на кеш паметта на дисковете на масива. Цялото съдържание на буфера на RAID контролера е невъзстановимо, ако той няма BBU. Много RAID контролери биха функционирали оптимално и с най-висока производителност, ако имат инсталирано BBU устройство.

- Предимства на хардуерния RAID масив – висока производителност при по-сложни RAID системи и при ползването на битове за четност; не се използват ресурсите на системата (процесор, памет); възможност за замяна на дискове в работещо състояние (disk hot swapping); по-малко време за възстановяване на масива при отпадане на някой от дисковете; лесен за инсталация – не е нужно инсталиране и конфигуриране на допълнителен софтуер; при наличие на батерия (BBU) изчакващите записи в кеша на контролера няма да бъдат загубени при спиране на захранването.

- Недостатъци на хардуерния RAID масив – допълнителни разходи – хардуерните RAID контролери, струват повече отколкото обикновените дискове контролери; алгоритмите и кодът са затворени собственически от компанията произвела контролера; ограничена възможност за замяна и миграция на дисковия масив; липса на гъвкавост (невъзможност за по-големи промени на масива); високите клас контролери могат да бъдат доста скъпи.

Софтуерен RAID масив – един RAID масив също може да бъде изграден с помощта на отворен код, който се вписва към операционната система. Софтуерният RAID използва споделените ресурси на системата – памет и процесор. Той е по-евтиният вариант в сравнение с хардуерния RAID.

- Предимства на софтуерния RAID масив – по-евтин в сравнение с хардуерния RAID, част е от операционната система и не се изискват допълнителни средства за хардуер; отворен код – RAID масивът е независим от хардуера на системата; стандартизирана конфигурация за всяка една операционна система; сравнително добра производителност – процесорите стават все по-бързи; за сравнително не големи натоварвания и задачи, софтуерният RAID е отлично и евтино решение; гъвкав метод – позволява преконфигурирането на RAID масива по множество различни начини.
- Недостатъци на софтуерния RAID масив – по-голяма сложност за изграждане в сравнение с хардуерния RAID. При по-сложни RAID системи и по-високи натоварвания отстъпва по производителност спрямо хардуерните контролери; невъзможна замяна на диск в работещо положение; използва част от ресурсите на системата (процесор и памет); няма възможност за BBU, тоест при спиране на захранването незаписаната информация в кеша се губи; по-бавно възстановяване на масива; по-голяма подготовка при инсталацията на операционната система.

## II. Най – често използвани сървъри – функционалности и изисквания

### 1.Файловият сървър

е вид сървър, който се използва за съхраняване и споделяне на файлове в мрежа. Файловите сървъри осигуряват централизирано съхранение и позволяват на потребителите да имат достъп до файлове от всяко място в мрежата. Те са от съществено значение за фирми и организации, които трябва да споделят файлове между служители, отдели и местоположения. Файловият сървър обикновено работи на специален компютър или сървър, който е оптимизиран за съхранение и управление на файлове. Сървърът е оборудван с един или повече твърди дискове или други устройства за съхранение и файловете се съхраняват на тези дискове. Сървърът е свързан към мрежата и е конфигуриран да споделя файловете с оторизирани потребители.

*Файловите сървъри предоставят няколко предимства, включително:*

Централизирано съхранение: Файловите сървъри осигуряват централно място за съхранение на файлове, което улеснява достъпа и споделянето на файлове от потребителите.

Контрол на достъпа: Файловите сървъри позволяват на администраторите да контролират кой има достъп до файлове и папки, като гарантират, че чувствителната информация е достъпна само за оторизирани потребители.

Архивиране и възстановяване: Файловите сървъри могат да бъдат конфигурирани за автоматично архивиране на файлове на регулярна основа, намалявайки риска от загуба на данни в случай на хардуерен срив или друго бедствие.

Отдалечен достъп: Файловите сървъри могат да бъдат достъпни дистанционно през интернет, което позволява на потребителите да имат достъп до файлове от всяко място с интернет връзка.

Мащабируемост: Файловите сървъри могат лесно да бъдат мащабирани с нарастването на нуждите от съхранение на една организация, чрез добавяне на допълнителни твърди дискове или други устройства за съхранение към сървъра.

Файловите сървъри могат да бъдат настроени с помощта на различни протоколи и технологии, включително:

Мрежова файлова система (NFS): Този протокол се използва за споделяне на файлове между Unix-базирани системи. Позволява на отдалечени потребители достъп до файлове, сякаш се съхраняват локално на собствения им компютър.

Блокиране на сървърни съобщения (SMB): Този протокол се използва за споделяне на файлове между базирани на Windows системи. Позволява на отдалечени потребители достъп до файлове, сякаш се съхраняват локално на собствения им компютър.

Network Attached Storage (NAS): Тази технология се използва за осигуряване на централизирано съхранение на файлове и други данни. NAS устройствата обикновено са малки, специализирани компютри, които са проектирани да съхраняват и споделят файлове в мрежа.

Мрежа за съхранение (SAN): Тази технология се използва за осигуряване на високопроизводително съхранение на големи количества данни. SAN обикновено се използват в среди на корпоративно ниво и са проектирани да осигурят високоскоростен достъп до големи количества данни.

Като цяло файловете сървъри са основен компонент на съвременните бизнес и организационни мрежи. Те осигуряват централизиран, сигурен и ефективен начин за съхраняване и споделяне на файлове между потребители, отдели и местоположения.

**2. Уеб сървърите** - са критичен компонент на интернет и играят важна роля при обслужването на уеб страници и друго онлайн съдържание. С прости думи, уеб сървърът е компютърна програма, която обработва заявки от уеб браузъри и обслужва уеб страници в отговор. Ето всичко, което трябва да знаете за уеб сървърите:

#### ■ Как работят уеб сървърите:

Уеб сървърите използват различни протоколи и технологии, за да доставят уеб страници на клиенти. Най-често използваният протокол е Hypertext Transfer Protocol (HTTP), който определя как уеб браузърите и сървърите комуникират помежду си.

Когато потребител въведе уеб адрес или щракне върху връзка, неговият браузър изпраща заявка до сървъра за определената уеб страница. След това сървърът обработва заявката, извлича съответната уеб страница и я изпраща обратно към браузъра на потребителя.

#### ■ Видове уеб сървъри:

Налични са няколко вида уеб сървъри, включително Apache, Nginx, IIS и Lighttpd. Всеки от тези сървъри има свои собствени уникални характеристики и предимства, а изборът на сървър зависи от специфичните нужди на уебсайта или приложението.

#### ■ Конфигурация:

Уеб сървърите могат да бъдат конфигурирани по много различни начини, в зависимост от нуждите на уебсайта или приложението. Конфигурационните настройки могат да включват опции за сигурност, настройки за кеширане, настройки за компресиране и други.

#### ■ Изпълнение:

Производителността на уеб сървъра е от решаващо значение за осигуряване на бързо зареждане на страницата и добро потребителско изживяване. Факторите, които могат да повлияят на производителността на сървъра, включват хардуер, натоварване на сървъра, скорост на мрежата и софтуерна оптимизация.

#### ■ Сигурност:

Уеб сървърите могат да бъдат уязвими към различни заплахи за сигурността, включително хакерство, злонамерен софтуер и атаки за отказ на услуга. Мерки за сигурност като защитни стени, SSL сертификати и системи за откриване на проникване могат да помогнат за защитата на уеб сървърите от тези заплахи.

#### ■ Поддръжка:

Уеб сървърите изискват редовна поддръжка, за да се гарантира, че работят гладко и ефективно. Задачите по поддръжката могат да включват софтуерни актуализации, хардуерни надстройки и наблюдение на регистрационните файлове на сървъра за грешки или необичайна дейност. Като цяло, уеб сървърите са критичен компонент на интернет и играят жизненоважна роля в доставянето на онлайн съдържание до потребителите по целия свят. Правилната конфигурация, оптимизирането на производителността и мерките за сигурност са от съществено значение, за да се гарантира, че уеб сървърите работят с максимална ефективност и осигуряват безпроблемно потребителско изживяване.

### **3.Mail servers**

Пощенските сървъри, известни също като имейл сървъри, са компютърни системи, които отговарят за изпращането, получаването и съхраняването на имейл съобщения. Те са основен компонент на съвременната комуникация и се използват от хора, фирми и организации по целия свят. Пощенските сървъри работят по модел клиент-сървър, където клиент (като настолен имейл клиент или интерфейс за уеб поща) комуникира със сървъра, за да изпраща или получава имейл съобщения. Когато потребител изпрати имейл съобщение, то първо се предава на сървъра за изходяща поща на подателя, който след това го препраща към сървъра за входяща поща на получателя. След това получателят може да получи достъп до имейл съобщението чрез своя имейл клиент или интерфейс за уеб поща. Пощенските сървъри използват различни протоколи за управление на имейл съобщения, включително Simple Mail Transfer Protocol (SMTP) за изпращане на съобщения, Post Office Protocol (POP) и Internet Message Access Protocol (IMAP) за получаване на съобщения и Domain Name System (DNS) за маршрутизиране на имейл до съответния пощенски сървър.

Има два основни типа пощенски сървъри: входящи и изходящи. Сървърите за входяща поща получават имейл съобщения от външни източници и ги доставят до съответната пощенска кутия на получателя. Сървърите за изходяща поща отговарят за изпращането на имейл съобщения от пощенската кутия на потребителя до външни получатели. Внедряване, включително локални и базирани на облак. Локалните пощенски сървъри се инсталират и управляват на място от организацията, осигурявайки пълен контрол и персонализиране на конфигурацията на пощенския сървър. Базираните в облак пощенски сървъри се хостват и



управляват от доставчици трети страни, осигурявайки лекота на използване и мащабируемост, без да изискват значителни хардуерни ресурси или ресурси за поддръжка.

Някои популярни софтуерни програми за имейл сървъри включват Microsoft Exchange Server, Postfix, Sendmail и Zimbra.

Като цяло пощенските сървъри играят критична роля в съвременната комуникация и са необходим компонент от ИТ инфраструктурата на всяка организация. Те позволяват ефективна и сигурна комуникация по имейл, позволявайки на лица и организации да се свързват с други по целия свят.

#### **4.Database servers**

Сървърът на база данни е специализиран тип сървър, който е проектиран да съхранява, управлява и извлича данни за използване от други приложения или потребители. Тези сървъри са оптимизирани за работа с големи обеми данни и обикновено се използват в среди на корпоративно ниво, където съхранението и извличането на данни са критични за бизнес операциите.

*Ето някои ключови характеристики и функции на сървърите на бази данни:*

Съхранение и управление на данни: Сървърите на бази данни са проектирани да съхраняват и управляват големи обеми данни по структуриран и организиран начин. Те използват система за управление на база данни (СУБД), за да създават, модифицират и извличат данни в отговор на потребителски заявки.

Висока производителност: Сървърите на бази данни са оптимизирани за високоскоростен достъп и обработка на данни. Те обикновено използват специализирани хардуерни и софтуерни компоненти за постигане на високи нива на производителност, надеждност и мащабируемост.

Сигурност: Сървърите на бази данни са проектирани да защитават данните от неоторизиран достъп и да гарантират поверителност и сигурност на данните. Те използват различни механизми за сигурност като контрол на достъпа, криптиране и архивиране и възстановяване на данни, за да гарантират целостта и наличността на данните.

Архивиране и възстановяване: Сървърите на бази данни са проектирани да предоставят механизми за архивиране и възстановяване за защита срещу загуба или повреда на данни. Това включва функции като автоматично архивиране, репликация на данни и възстановяване на данни в случай на системна повреда или бедствие.

Запитване и отчитане: Сървърите на бази данни поддържат сложна функционалност за заявки и отчитане, което позволява на потребителите да извличат и анализират данни по различни начини. Това включва функции като агрегиране на данни, филтриране и сортиране, както и поддръжка за персонализирани отчети и анализи.

**Налични са няколко типа сървъри за бази данни, включително:**

Сървъри за релационни бази данни: Тези сървъри са проектирани да съхраняват данни по структуриран и организиран начин, използвайки таблици, колони и връзки между елементи от данни.

Обектно-ориентирани сървъри на бази данни: Тези сървъри са проектирани да съхраняват данни под формата на обекти, които могат да бъдат достъпни и манипулирани с помощта на техники за обектно-ориентирано програмиране.

NoSQL сървъри за бази данни: Тези сървъри са проектирани да обработват неструктурирани или полуструктурирани данни, като текст, изображения или мултимедийно съдържание, които не могат лесно да бъдат организирани в таблици и колони.

Като цяло сървърите на бази данни са критичен компонент на съвременните бизнес и организационни мрежи. Те предоставят централизиран, ефективен и сигурен начин за съхраняване, управление и извличане на данни за използване от други приложения и потребители.

## **5.Application servers**

Сървърите за приложения са тип сървър, който е проектиран да хоства и управлява уеб приложения. Тези сървъри предоставят платформа за хостване, управление и доставка на приложения до клиенти по мрежа. Те позволяват на разработчиците да внедряват и управляват приложения на централизирано място, което улеснява поддръжката и актуализирането им.

*Сървърите на приложения обикновено предоставят набор от функции, включително:*

Мидълуер услуги: Сървърите на приложения предоставят мидълуер услуги, които позволяват на разработчиците да пишат и внедряват приложения, които могат да взаимодействат с различни бекенд услуги като бази данни, системи за съобщения и други сървъри.

Услуги за сигурност: Сървърите на приложения предоставят функции за сигурност като удостоверяване на потребителя, контрол на достъпа и криптиране за защита на чувствителни данни.

Балансиране на натоварването: Сървърите на приложения могат да бъдат конфигурирани да балансират натоварването между множество сървъри, за да се гарантира, че приложенията са налични и реагират дори по време на периоди с голям трафик.

Управление на сесии: Сървърите на приложения предоставят услуги за управление на сесии, които позволяват на сървъра да управлява потребителски сесии, да поддържа данни за сесиите и да наблюдава активността на потребителите.

Висока наличност и толерантност към грешки: Сървърите за приложения са проектирани да осигурят висока наличност и толерантност към грешки чрез репликиране на компоненти на приложения в множество сървъри, осигурявайки автоматично превключване и възстановяване. Има много различни типове сървъри за приложения, всеки със собствен набор

от функции и възможности. Някои популярни сървъри за приложения включват Apache Tomcat, JBoss, WebSphere и WebLogic.

В обобщение, сървърите на приложения са критичен компонент на съвременната разработка и внедряване на софтуер. Те предоставят на разработчиците платформа за внедряване, управление и мащабиране на уеб приложения и предлагат набор от функции за гарантиране на сигурност, достъпност и производителност.

## **6.Proxy servers**

Прокси сървърът е вид сървър, който действа като посредник между клиенти и сървъри. Когато клиент поиска ресурс като уеб страница или файл, заявката първо се изпраща до прокси сървъра, който след това препраща заявката към съответния сървър от името на клиента. След това отговорът от сървъра се изпраща обратно към прокси сървъра, който от своя страна препраща отговора към клиента. Този процес може да помогне за подобряване на сигурността, производителността и поверителността за клиентите, които имат достъп до ресурси в интернет.

*Прокси сървърите могат да се използват за различни цели, включително:*

Кеширане: Прокси сървърите могат да кешират често искани ресурси, като уеб страници или файлове, за да намалят честотната лента и времето, необходимо за достъп до тях. Това може да помогне за подобряване на производителността на клиентите, особено в ситуации, в които честотната лента на мрежата е ограничена.

Филтриране: Прокси сървърите могат да бъдат конфигурирани да филтрират трафика въз основа на различни критерии, като URL или тип съдържание. Това може да помогне за блокиране на достъпа до нежелани или вредни ресурси, като злонамерен софтуер или неподходящо съдържание.

Анонимност: Прокси сървърите могат да се използват за защита на самоличността и поверителността на клиентите, които имат достъп до ресурси в интернет. Чрез използването на прокси сървър клиентите могат да маскират своя IP адрес и друга идентифицираща информация от сървърите, до които имат достъп.

Балансиране на натоварването: Прокси сървърите могат да се използват за разпределяне на трафик между множество сървъри, което спомага за подобряване на производителността и наличността за клиенти, които имат достъп до ресурси в интернет.

*Има няколко вида прокси сървъри, включително:*

- Прокси за препращане: Прокси за препращане е прокси сървър, който се използва от клиенти за достъп до ресурси в интернет. Когато клиент направи заявка, тя първо се изпраща до препращащия прокси, който след това препраща заявката към съответния сървър от името на клиента;

- Обратно прокси: Обратното прокси е прокси сървър, който се използва от сървърите за достъп до ресурси от името на клиенти. Когато клиент направи заявка, тя първо се изпраща до обратния прокси, който след това препраща заявката към съответния сървър от името на сървъра;
- Прозрачен прокси: Прозрачният прокси е прокси сървър, който не променя заявката или отговора на клиента. Използва се предимно за целите на кеширане и филтриране;
- Анонимен прокси: Анонимният прокси е прокси сървър, който скрива IP адреса и друга идентифицираща информация на клиента от сървърите, до които има достъп.

В обобщение, прокси сървърът е междинен сървър, който може да се използва за различни цели, включително кеширане, филтриране, анонимност и балансиране на натоварването. Има няколко вида прокси сървъри, всеки със собствен набор от функции и случаи на използване.

## 7.Virtual servers

Виртуалните сървъри, известни още като виртуални частни сървъри (VPS), са вид сървър, който работи като физическа машина, но всъщност е софтуерно дефинирана виртуализация на физически сървър. Това позволява създаването на множество виртуални сървъри на една физическа машина, всеки със собствена операционна система, приложения и ресурси.

Виртуалните сървъри обикновено се използват в ситуации, в които специален физически сървър може да е твърде скъп или непрактичен за нуждите на организацията. Те предлагат няколко предимства пред физическите сървъри, включително:

Ефективност на разходите: Виртуалните сървъри могат да бъдат значително по-евтини от физическите сървъри, тъй като позволяват множество сървъри да работят на една физическа машина, намалявайки разходите за хардуер.

Мащабируемост: Виртуалните сървъри могат лесно да бъдат мащабирани нагоре или надолу в зависимост от нуждите на организацията, което им позволява да добавят или премахват ресурси според нуждите.

Гъвкавост: Тъй като виртуалните сървъри работят независимо един от друг, те могат да бъдат конфигурирани и персонализирани, за да отговарят на специфични изисквания.

Надеждност: Виртуалните сървъри са проектирани да бъдат високо достъпни, с вградени автоматичен отказ и резервиране, за да осигурят непрекъсната работа.

Виртуалните сървъри могат да бъдат настроени с помощта на различни технологии за виртуализация, включително хипервайзори като VMware, Hyper-V и KVM. Те обикновено се използват за уеб хостинг, хостинг на бази данни, хостинг на приложения и други подобни цели.

Въпреки предимствата си, виртуалните сървъри имат някои ограничения. Те могат да бъдат ограничени от физическите ресурси на основния хардуер и може да не са подходящи за

приложения, които изискват високопроизводителен хардуер или специализирани хардуерни конфигурации. Освен това виртуалните сървъри може да са по-сложни за управление от физическите сървъри, изискващи специализирани знания и опит за конфигуриране и поддръжка.

### **III. ОСНОВНИ ИЗИСКВАНИЯ КЪМ СЪРВЪРНИТЕ ПОМЕЩЕНИЯ**

В сървърната зала има разпределителни устройства, както и голям брой различни телекомуникационни съоръжения. В него могат да се помещават и пасивни разпределителни уредби, различни разпределителни пунктове. Ако се основава на стандарти, тогава те нямат критерии, които определят вида на телекомуникационната зала по броя на оборудването, инсталирано в него. Следователно типът на помещението трябва да се определя от монтажника или клиента. Наложително е сървърната зала да се намира в непосредствена близост до магистралните комуникационни канали. Препоръчително е да поставим сървъра до основната точка на разпространение. И в този случай, ако е възможно, тогава главната разпределителна точка трябва да бъде инсталирана в сървърната стая. Поставяме сървърно помещение в непосредствена близост до асансьорни шахти, стълби, вентилационни камери, както и други строителни елементи, които ограничават разширяването в бъдеще. Необходимо е да се избере размерът, въз основа на коя площ се обслужва в работната зона, както и броя на оборудването. Важно е да се вземе предвид не само големината на оборудването, но и вида на инсталацията.

Най-важното е, че след инсталирането на оборудването достъпът до оборудването е бил улеснен във възможно най-голяма степен. Също така трябва да имате предвид, че е възможно по-късно да се инсталират различни допълнителни устройства. Стаята трябва да има височина над 244 м. Що се отнася до площта, тя трябва да бъде повече от 14 кв. м. Площта на сървърната стая може да се изчисли самостоятелно. За всеки 10 кв.м. м от работната площ, която се обслужва, се нуждае от 009 кв. м. m стаи. Не е възможно да се постави сървърно помещение в сутерена, ако не е защитено от проникване на вода отвън. Не е възможно да има тръбопроводи, дренажни системи в помещението, освен ако, разбира се, те се използват за нормалното функциониране на системите и оборудването, разположени в сървърното помещение.

В случай, че има вероятност от изтичане на вода, трябва да се извърши дренажна инсталация. Например направете дренажен отвор в пода. Внимателно проучете всички технически изисквания за сървърната зала, така че оборудването да работи без проблеми. За сървърните помещения е най-добре да се използват стаи, в които изцяло липсват прозорци. В случай, че няма такива стаи, трябва напълно да се отървете от прозорците - поставете ги в тухли. Що се

отнася до вратата, ширината му трябва да бъде повече от 19 м, височината е повече от 2 м. Задължително е вратата да бъде затворена до ключалката, в който случай ще можете да ограничите достъпа до стаята. Можете да използвате плъзгащи конструкции на вратите. Що се отнася до вратите на панти, е необходимо те да се отварят отвън. В случай, че в бъдеще се планира поставянето на димензионно оборудване в помещението, е възможно да се инсталира двойна врата с прорез, Ширина от 182 м, височина трябва да бъде повече от 228 м. Що се отнася до тавана, забранено е да се използват окачени типове в сървърното помещение. По стените трябва да бъдат покрити тавани и подове, което ще усложни отделянето, натрупването и отлагането на прах. На тавана трябва да бъде хидроизолация, за да се изключи напълно възможността от изтичане на вода. В случай, че в сървърното помещение е монтирано тежко оборудване, трябва да се изчисли статично и динамично натоварване на подовото покритие (подови настилки). Тежкото оборудване може да включва акумулаторни батерии, голям брой различно оборудване, сглобени в една монтажна кутия, чиято маса е повече от 500 кг. Системата за контрол и контрол на климата трябва да бъде монтирана в стаята на сървъра. Необходимо е да се осигури постоянна температура и влажност в помещението, така че цялото активно оборудване да е максимално и стабилно функциониращо. Системата за контрол на микроклимата трябва да осигурява подкрепа за зададения температурен режим както през лятото, така и през зимата. Освен това е необходимо тя да работи непрекъснато и денонощно. Ако централизираната система за отопление и вентилация в сградата не може да осигури непрекъсната работа, разрешено е да се инсталира автономна система за конкретна стая. При температури от 20 до 25 Измерването на влажността и температурата трябва да бъде фиксирано на височина от един и половина метра от пода в зоната, където се подава студен въздух. Ако водата е произведена, температурата и влажността се измерват в работното оборудване директно в монтажния шкаф. Наложително е да се осигури натиск в сървърната зала повече, отколкото в съседните помещения. Ако в сървърната стая има персонал постоянно, въздухът трябва да се сменя поне веднъж на час. Също така е желателно да се използва филтрираща система и пречистване на въздуха, които влизат в помещението. Ако в цялата къща е инсталирана резервна система, тогава контролът на микроклимата в сървърната стая трябва да бъде свързан с него. Градуса е необходимо относителната влажност на въздуха да бъде 40-55%. Цялата сървърна зала трябва да бъде максимално защитена от вредни вещества и прах, тъй като те имат неблагоприятно въздействие върху работата на цялото оборудване. В сървърното помещение има максимално допустими концентрации на вредни вещества:

Хлор не повече от 001 ppm.

Допустимо е не повече от 100 mg прах на куб. м на ден.

Не повече от 4 mg на кубичен метър. m на ден сероводород не повече от 005 милисекунди.

Азотни оксиди не повече от 01 милилитра.

Серен диоксид не надвишава 03 ppm.

Ако е необходимо, използвайте система за филтриране и пречистване на въздуха, какво се случва Прилагането на маслени филтри в сървърните стаи е строго забранено. Не забравяйте да обърнете внимание на изискванията за помещенията на сървъра на защитната стена. По подобен начин се извършва и отстраняването на дим и вредни вещества. По отношение на вибрациите, които влияят неблагоприятно на работата на цялото оборудване, е необходимо амплитудата на колебанията при 25 Hz да не надвишава 01 mm. Не забравяйте да осигурите висококачествено осветление в сървърните стаи, то трябва да бъде поне 500 lm. Желателно е в помещението да има поне два блока двойни електрически контакти. Те трябва да се изискват от различни кабели, максималното натоварване е до 16 A. Препоръчително е също така да се монтират допълнителни блокове от двойни контакти в стената на разстояние от 18м. един от друг. Разстоянието от пода трябва да бъде най-малко 15 cm. Стаята на сървъра трябва да се захранва от захранващ кабел, директно от основната сметка за разпределение. Това е едно от изискванията за пожар за сървърните стаи. За нюансите ще се каже малко по-ниско. Ако сградата има резервно електрозахранване, е необходимо стаята на сървъра да е свързана с нея. Не забравяйте да инсталирате отделно разпределително табло за цялото помещение. Разрешава се инсталирането на UPS до 100 kVA директно в сървърната зала. Ако източниците на непрекъснато захранване имат мощност над 100 kVA, те трябва да бъдат монтирани в отделно помещение.

В случай на пожар, тези упори ще се разширят, което ще доведе до припокриване на пространството, дим и огън няма да се разпространи. Тавани, стени, всички прегради в сървърното помещение трябва да бъдат направени от негорими материали, а също така да осигуряват минимална стойност на огнеустойчивост от 45 минути. Що се отнася до входната врата, е необходимо да има пожароустойчивост над 36 минути. Разрешено е да се произвеждат врати от огнеупорни материали, с дебелина: 40 mm. Можете да използвате врата от дърво, но те трябва да са сигурнипокрити с азбест или обшита стомана с двете страни. Задължително е да се монтират изпускателни колектори с автоматично или ръчно отваряне. Тяхната площ трябва да бъде повече от 02% от общата площ на помещението. При монтажа на оборудването и избора на стая, трябва да се разчита на всички изисквания и препоръки, споменати в дипломния проект.

**1.Контрол на температурата:** Измерването на нивото на осветеност трябва да се извършва на един метър от пода. Необходимо е осветлението на осветителните устройства и на цялото телекомуникационно оборудване да се извършва от различни табла. Всички тела трябва да бъдат поставени на тавана. Можете да използвате един или няколко превключватели, които се намират в близост до вратата, един и половина фута от пода, за да контролирате светлинните

устройства. Използването на устройства за плавно управление на осветлението е забранено. Стаята за сървърното оборудване трябва да се намира далеч от източници на смущения от електромагнитния тип. Силата на полето трябва да бъде не повече от 3 V/m.

Сървърите генерират много топлина и контролът на температурата е от решаващо значение, за да се гарантира, че работят ефективно и надеждно. Препоръчителният температурен диапазон за сървърно помещение е от 18°C до 27°C . Този температурен диапазон може да бъде постигнат чрез използване на климатик или специална охладителна система. Температурата трябва да се следи редовно с помощта на температурни сензори и трябва да се настроят сигнали за уведомяване на съответния персонал, ако температурата се качи над или падне под препоръчания диапазон.

## **2.Контрол на влажността:**

В допълнение към контрола на температурата, контролът на влажността също е важен за сървърните стаи. Препоръчителният диапазон на влажност е от 40% до 60%, тъй като високите нива на влажност могат да причинят кондензация, която може да повреди оборудването. Нивата на влажност могат да се контролират чрез използването на овлажнители и изсушители, както и чрез подходяща вентилация. Влажността трябва да се следи редовно с помощта на сензори за влажност и трябва да се настроят сигнали за уведомяване на съответния персонал, ако влажността падне извън препоръчания диапазон.

## **3.Вентилация:**

Необходима е адекватна вентилация, за да се отстрани горещият въздух от сървърното помещение и да се замени със студен въздух. Това може да се постигне чрез използване на вентилатори или климатични инсталации, а вентилационната система трябва да е проектирана така, че да гарантира, че студеният въздух се разпределя равномерно в цялата стая. Също така е важно да се гарантира, че има достатъчно пространство между стелажите за оборудване, за да се осигури правилен въздушен поток.

## **4. Електрическо захранване:**

Надеждното и стабилно захранване е от решаващо значение за правилното функциониране на сървърите. Сървърното помещение трябва да има специален източник на захранване с резервни захранвания, като непрекъсваеми захранващи устройства (UPS), за да се предотврати загуба на данни или повреда поради прекъсване на захранването или пренапрежение. Електрическата система също трябва да бъде проектирана така, че да предотвратява



претоварване, с подходящи прекъсвачи и блокове за разпределение на мощността (PDU), за да се гарантира, че мощността се разпределя равномерно към стелажите за оборудване.

## **5. Потушаване на пожар:**

Системата за потушаване на пожар е от съществено значение за защитата на сървърната стая и нейното съдържание в случай на пожар. Системата може да бъде на водна основа или на газ (като система FM200). Системата трябва да бъде проектирана така, че да не повреди оборудването или да навреди на персонала в случай на активиране. Редовната поддръжка и тестване на пожарогасителната система също е важно, за да се гарантира, че тя функционира правилно.

## **6. Сигурност:**

Достъпът до сървърната стая трябва да бъде ограничен и наблюдаван, за да се предотврати неоторизиран достъп. Това може да се постигне чрез използване на системи за контрол на достъпа, охранителни камери и охрана. Физически мерки за сигурност, като подсилени врати и стени, също трябва да бъдат приложени, за да се предотврати проникване с взлом. Целият персонал с достъп до сървърната стая трябва да бъде обучен относно протоколите и процедурите за сигурност.

## **7. Стелаж:**

Сървърната стая трябва да има достатъчно пространство за стелажи, за да побере всички сървъри, мрежови устройства и друго оборудване, с място за бъдещо разширяване. Оборудването трябва да бъде организирано по начин, който позволява лесен достъп и поддръжка. Трябва да се осигури достатъчно пространство между стелажите, за да се осигури правилна вентилация и окабеляване.

## **8. Окабеляване:**

Правилното управление на кабелите е от съществено значение, за да се гарантира, че кабелите са добре организирани и не създават никакви препятствия или опасности. Кабелите трябва да бъдат етикетирани и организирани по начин, който позволява лесна идентификация и поддръжка. Излишният кабел трябва да се съхранява спретнато, за да се предотвратят опасности от спъване и да се гарантира, че не пречи на вентилацията.

## **9. Мониторинг на околната среда:**

Сървърната стая трябва да бъде оборудвана със сензори за наблюдение на температурата, влажността, потреблението на енергия и други фактори на околната среда. Тези данни трябва да се събират и анализират, за да се идентифицират потенциални проблеми, преди да се превърнат в проблеми. Трябва да се настроят сигнали за уведомяване на съответния персонал, ако някой от наблюдаваните фактори е извън препоръчаните диапазони.

### ПРОЕКТИРАНЕ НА ЕФЕКТИВНА СХЕМА ЗА ПОДДЪРЖАНЕ И РЕГУЛИРАНЕ НА ОСНОВНИТЕ ХАРАКТЕРИСТИКИ В СЪРВЪРНО ПОМЕЩЕНИЕ ЧРЕЗ МИКРОКОНТРОЛЕРИ /ARDUINO/, ПРОГРАМИРАНИ НА C++.

#### 1.Идейно предложение:

Да се проектира и реализира измервателен модул на основните параметри - температура, влажност и физическа стабилност. Да се представи функционална схема на модула в който основен компонент е микроконтролер Arduino, програмиран на езика C++. Да се представят техническите параметри, които ще се отчитат през определен период на денонощието. Данните да се събират в база данни и периодически да се прави анализ на отчитаните резултати.

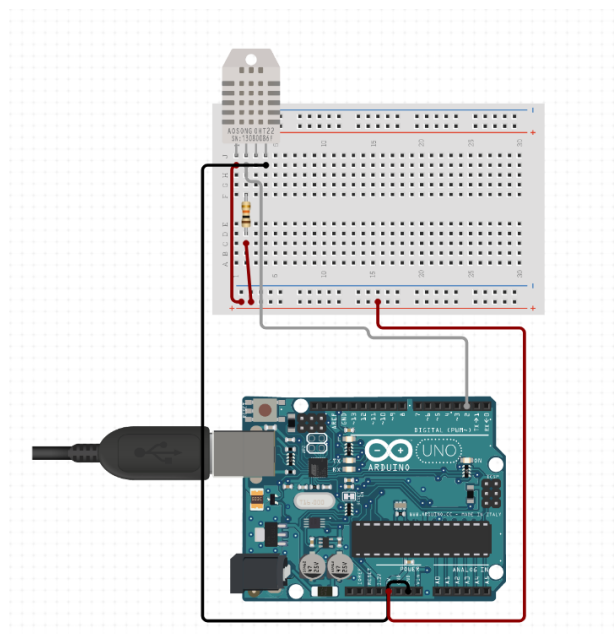
#### 2. Техническо решение

##### 2.1. Функционална схема на свързване на измервателния модул и описание:

Свързвам VCC пин на Ардуино към + лента на прототипна платка.

Свързвам GND пин на Ардуино към GND пин на DHT11 сензор.

Свързвам 2 пин на Ардуино към 3 пин на DHT11 сензор.



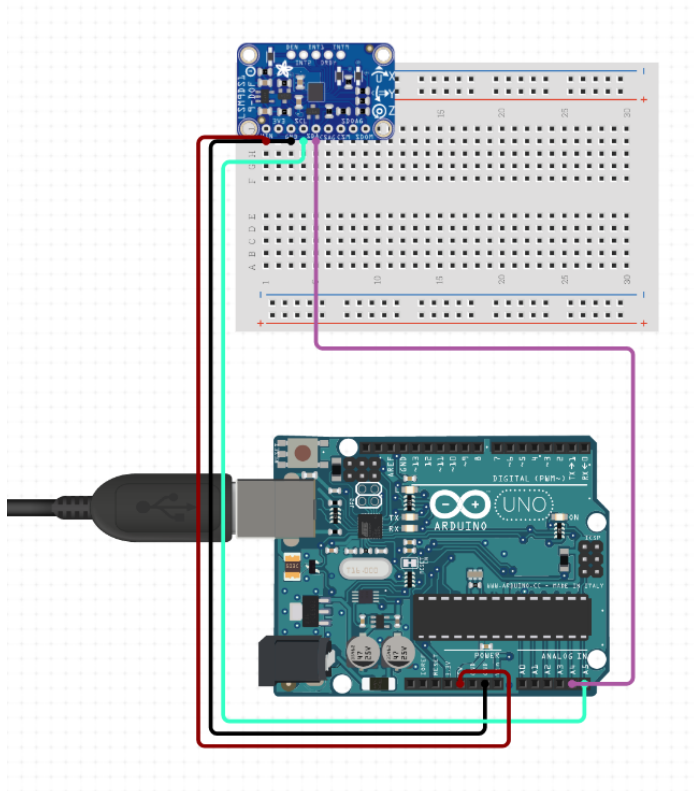
Фиг.5

Свързвам VCC пин на Ардуино към VIN пин на GY-521 сензор.

Свързвам GND пин на Ардуино към GND пин на GY-521 сензор.

Свързвам A1 пин на Ардуино към 4 пин на GY-521 сензор.

Свързвам A2 пин на Ардуино към 3 пин на GY-521 сензор.



Фиг.6

## 2.2. Ардуино код на C++

```
#include <Wire.h>
#include <MPU6050.h>
#include <DHT.h>

#define DHTPIN 2
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);
MPU6050 mpu;

void setup() {
    Serial.begin(9600);

    dht.begin();
    mpu.initialize();

    mpu.CalibrateGyro();
    mpu.CalibrateAccel();
}

void loop() {
    int16_t ax, ay, az;
    int16_t gx, gy, gz;
    int16_t temp;

    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
```

```

temp = mpu.getTemperature();
float humidity = dht.readHumidity();
float gyroX = gx / 131.0;
float gyroY = gy / 131.0;
float gyroZ = gz / 131.0;
float tempC = (temp / 340.0) + 36.53;

Serial.print("H:");
Serial.print(humidity);
Serial.print(",T:");
Serial.print(tempC);

if(gyroX > 2 || gyroY > 2 || gyroZ > 2)
{
    Serial.print(",S:");
    Serial.print("danger");
} else {
    Serial.print(",S:");
    Serial.print("ok");
}

Serial.println();
delay(250);
}

```

Фиг. 7

### 2.3. Сървърен код за сайт на javascript, ejs, html, css

```

const express = require("express");
const app = express();
const server = require("http").createServer(app);
const io = require("socket.io")(server);
const ejs = require("ejs");
const bodyParser = require('body-parser');
const { SerialPort } = require('serialport');
const { ReadlineParser } = require('@serialport/parser-readline');
const { exec } = require('child_process');
const requestIp = require('request-ip');
const PORT = 3000;
const {CreateUser, ValidateLogin, ClearActiveUsers,
    CheckActiveUsers, UpdateActiveUser, CheckIfIpActive,
    CreateRoomData, WaitConnection } = require(__dirname +
"/Database/middleware.js");

app.set('view engine', 'ejs');
app.use(bodyParser.urlencoded({extended: true}));
app.use(express.static(__dirname + '\\public'));

app.post("/heartbeat", async (req, res) => {
    let ip = requestIp.getClientIp(req);
    UpdateActiveUser(ip);
    res.sendStatus(200);
});

```

```

app.get("/RequestSocket", (req, res) => {
  res.sendFile(__dirname + '/node_modules/socket.io/client-dist/socket.io.js');
});

app.get('/', (req, res) => {
  res.render(__dirname + '/views/index.ejs', {message: " "});
});

app.get('/Login', (req, res) => {
  res.render(__dirname + '/views/index.ejs', {message: ""});
});

app.post('/Login', (req, res) => {
  let username = req.body.username;
  let password = req.body.password;
  let ip = requestIp.getClientIp(req);
  ValidateLogin(username, password, ip).then(result =>{
    if(result.Error)
    {
      res.render(__dirname + '/views/index.ejs', {message: result.Message});
    } else {
      console.log("Logged");
      res.sendFile(__dirname + '/views/Information.html');
    }
  });
});

app.get('/Register', (req, res) => {
  res.render(__dirname + '/views/Register.ejs', {message: ""});
});

app.post('/Register', (req, res) => {
  let username = req.body.regsiterUsername;
  let password = req.body.regsiterPassword;
  let confirmPassword = req.body.confirmPassword;
  if(username && password && confirmPassword)
  {
    if(password === confirmPassword)
    {
      CreateUser(username, password).then(result =>{
        if(result.Error)
        {
          res.render(__dirname + '/views/Register.ejs', {message: result.Message});
        } else {
          console.log("Registered");
          res.render(__dirname + '/views/index.ejs', {message: ""});
        }
      });
    } else {

```

```

        res.render(__dirname + '/views/Register.ejs', {message: "Passwords dont
match"}});
    }

}
else
{
    res.render(__dirname + '/views/Register.ejs', {message: "Fill all fields!"});
}

});

app.get('/Information', (req, res) => {
    let ip = requestIp.getClientIp(req);
    CheckIfIpActive(ip).then(result => {
        if(result)
        {
            res.sendFile(__dirname + '/views/Information.html');
        }
        else {
            res.render(__dirname + '/views/index.ejs', {message: ""});
        }
    })
});

app.post('/Information', (req, res) => {
    let username = req.body.username;
    let password = req.body.password;
    let ip = requestIp.getClientIp(req);
    CheckIfIpActive(ip).then(result => {
        if(result)
        {
            res.sendFile(__dirname + '/views/Information.html');
        }
        else {
            ValidateLogin(username, password, ip).then(result =>{
                if(result.Error)
                {
                    res.render(__dirname + '/views/index.ejs', {message: result.Message});
                } else {
                    console.log("Logged");
                    res.sendFile(__dirname + '/views/Information.html');
                }
            });
        }
    });
});

io.on('connection', (socket) => {
    socket.on('shutdown', () =>{
        exec('shutdown /s /t 0', (error, stdout, stderr) => {
            if (error) {

```

```

        console.error(`Error executing command: ${error}`);
        return;
    }
    })
});
});

const port = new SerialPort({
    path: 'COM3',
    baudRate: 9600,
}, (err => {
    if (err)
    {
        console.log('No device on serial port.');
        io.emit('SerialError');
    }
    else {
        console.log('Device connected');
    }
}));

const parser = port.pipe(new ReadlineParser({delimiter: '\r\n'}));

let lastReading = "";
let count = 0;
parser.on('data', async (data) => {
    let output = data.toString().split(",");
    let DecodedData = { };
    for (let i = 0; i < output.length; i++) {
        DecodedData[output[i].split(':')[0].toString()] =
output[i].split(':')[1].toString();
    }
    count++;
    if (count > 4) {
        count = 0;
        CreateRoomData(DecodedData);
    }
    console.log(DecodedData);
    if (lastReading === 'danger' && DecodedData['S'] === 'danger')
    {
        console.log(`Error executing command: `);
        exec('shutdown /s /t 0', (error, stdout, stderr) => {
            if (error) {
                console.log(`Error executing command: ${error}`);
                return;
            }
        });
    }
    lastReading = DecodedData['S'];
    if (DecodedData['T'] >= 30)
    {

```



```

    exec('shutdown /s /t 0', (error, stdout, stderr) => {
      if (error) {
        console.log(`Error executing command: ${error}`);
        return;
      }
    });
  }
  io.emit('ArduionoData', DecodedData);
});

WaitConnection().then(() =>
{
  ClearActiveUsers();
  CheckActiveUsers();
});

server.listen(PORT, () =>
{
  process.env.TZ = "Europe/Kiev";
  console.log("Server is listening. http://localhost:3000")
});

```

Фиг.7

## 2.2. Код на страниците.

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset=utf-8>
    <link rel="stylesheet" href="/styles/Information.css">
    <title>Server info</title>
    <script src="/RequestSocket"></script>
    <script >
      const socket = io();

      socket.on('connect', () => {
        console.log('Connected to server');
      });

      socket.on('ArduionoData', (data) => {
        let DecodedData = data;

        const HumidityOutput = document.getElementById('HumidityOutput');
        HumidityOutput.textContent = DecodedData["H"] + "%" ;

        const TemperatureOutput =
document.getElementById('TemperatureOutput');
        TemperatureOutput.textContent = DecodedData["T"] + "C" ;

        const LevelOutput = document.getElementById('LevelOutput');
        LevelOutput.textContent = DecodedData["S"];

```

```

});

socket.on("SerialError", () => {
    window.location.href = "NoSerialDevice.html";
});

function StartShutdown(){
    socket.emit('shutdown');
    console.log("Shutdown sent");
}
</script>
</head>
<body>
    <h2>Информация за сървърно помещение</h2>

    <form class="table">
        <div class="row">
            <label>Температура: </label>
            <a id="TemperatureOutput"></a>
        </div>

        <div class="row">
            <label>Влажност: </label>
            <a id="HumidityOutput"></a>
        </div>

        <div class="row" style="border-bottom: 0">
            <label>Стабилност: </label>
            <a id="LevelOutput"></a>
        </div>
        <div class="button">
            <button class="pepe" onclick="StartShutdown()">Изключване на
сървър</button>
        </div>
    </form>

</body>
<script>
window.setInterval(function() {
    fetch("/heartbeat", {
        method: "POST",
        headers: {
            "Content-Type": "application/json"
        },
        body: JSON.stringify("")
    });
}, 5000);
</script>
</html>

```

Фиг.8

```

const { Sequelize, DataTypes, Op } = require('sequelize');

// const sequelize = new Sequelize('kurami_', 'kurami_', '', {
//   host: 'sql.bsite.net/MSSQL2016',
//   dialect: 'mssql',
//   port: '',
//   database: 'kurami_',
//   logging: false,
//   dialectOptions: {
//     options: {
//       encrypt: false // If you are connecting to a server with SSL encryption
//     }
//   }
// });

const sequelize = new Sequelize('haaaaaaaaaaaaaa', 'haaaaaaaaaaaaaa',
'Ts0BHETj_~SF', {
  dialect: 'mssql',
  host: 'den1.mssql8.gear.host',
  logging: false,
  dialectOptions: {
    options: {
      server: "MSSQL8"
    }
  }
});

const User = sequelize.define('Users', {
  UserId: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true
  },
  Username: {
    type: DataTypes.STRING,
    allowNull: false
  },
  Password: {
    type: DataTypes.STRING,
    allowNull: false
  }
});

const UserIP = sequelize.define('UserIPs', {
  Id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true
  },

```

```

    UserId:
    {
      type: DataTypes.INTEGER,

      references:
      {
        model: User,
        key: 'UserId'
      }
    },
    IP:
    {
      type: DataTypes.STRING,
    }
  });

const Data = sequelize.define('RoomData', {
  Id: {
    type: DataTypes.BIGINT,
    primaryKey: true,
    autoIncrement: true
  },
  Humidity: {
    type: DataTypes.STRING,
    allowNull: true
  },
  Temperature: {
    type: DataTypes.STRING,
    allowNull: true
  },
  Level: {
    type: DataTypes.STRING,
    allowNull: true
  }
});

sequelize.sync({ force: false });

module.exports = {
  sequelize,
  User,
  UserIP,
  Data,
  Op
};

```

*Fig.9*

```

const { Time, DateTime } = require('mssql');
const { sequelize, User, UserIP, Data, Op } = require('./models');

async function CheckConnection()

```

```

{
  try {
    await sequelize.authenticate().then(() =>
      {
        return true;
      }
    );
  } catch (error) {
    return false;
  }
}

async function WaitConnection()
{
  await new Promise(resolve => setTimeout(resolve, 10000));
}

async function CreateUser(Username, Password)
{
  let result = {Error: false , Message: ""};
  let entity = await User.findOne({ where: { Username: Username } });
  if(entity === null)
  {
    let test = await User.build({
      Username: Username,
      Password: Password
    });
    await test.save();
  }
  else{
    result.Error = true;
    result.Message = "Username already exist!";
  }
  return result;
}

async function ValidateLogin(Username, Password, IP)
{
  let result = {Error: false , Message: ""};
  let entity = await User.findOne({ where: { Username: Username, Password: Password } });
  if(entity != null)
  {
    let ipEntity = await UserIP.findOne({where: {UserId: entity.UserId}});
    if(ipEntity === null)
    {
      let test = await UserIP.build({
        UserId: entity.UserId,
        IP: IP
      });
      await test.save();
    }
  }
}

```

```

        } else {
            result.Error = true;
            result.Message = "Account is logged on another device!";
        }

    } else {
        result.Error = true;
        result.Message = "No user with this username and password!";
    }
    return result;
}

async function CheckIfIpActive(IP)
{
    let entity = await UserIP.findOne({where: {Ip: IP}});
    if(entity)
    {
        return true;
    } else {
        return false;
    }
}

async function UpdateActiveUser(IP){
    await UserIP.update({IP: IP},
        {
            where: {
                IP: IP
            }
        });
}

async function CreateRoomData(DecodedData){
    let test = await Data.build({
        Humidity: DecodedData['H'],
        Temperature: DecodedData['T'],
        Level: DecodedData['S']
    });
    await test.save();
}

async function ClearActiveUsers()
{
    await UserIP.destroy({
        truncate: true
    });
}

async function CheckActiveUsers()
{
    while(true)
    {

```

```

        let temp = new Date(new Date().getTime() - (2 * 60 * 1000));
        await UserIP.destroy({
            where:{
                updatedAt: {
                    [Op.lte]: temp
                }
            }
        });
        await new Promise(resolve => setTimeout(resolve, 60000));
    }
}

module.exports = {
    CreateUser,
    ValidateLogin,
    ClearActiveUsers,
    CheckActiveUsers,
    UpdateActiveUser,
    CheckIfIpActive,
    CreateRoomData,
    WaitConnection
};

```

*Фиг.10*

```

body {
    text-align: center;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    font-size: 16px;
    color: black;
    background-color: darkgray;
}

form{
    background-color: #4f4f4f;;
    width: 400px;
    height: 250px;
    display: grid;
    margin: 50px auto;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0px 0px 10px 0px rgba(0,0,0,0.75);
}

form div.row{
    text-align: start;
}

form div button{
    background-color: #c6aadd;
    color: #424242;;
    padding: 10px;
}

```

```

border: none;
border-radius: 5px;
cursor: pointer;
box-shadow: 0px 0px 5px 0px rgba(0,0,0,0.1);
transition: background-color 0.7s ease;;
}

button:hover {
  background-color: red;
  transform: scale(1.1);
}

form div.button{
  text-align: end;
}

h1 {
  text-align: left;
}

.container{
  display: flex;
  align-items: start;
  justify-content: left;
  width: 40%;
  height: 20%;
}

.row {
  display: inline-flexbox;
  justify-content: space-between;
  padding: 10px 0;
  border-bottom: 1px solid #ccc;
  margin-right: 10px;
}

.row label {
  font-weight: bold;
  margin-right: 10px;
}

.row a {
  color: white;
}

```



```

body {
    background-color: #919191;
}
form {
    background-color: #4f4f4f;;
    width: 400px;
    height: 250px;
    display: grid;
    margin: 50px auto;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0px 0px 10px 0px rgba(0,0,0,0.75);
}
div.label{
    text-align: center;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}
small{
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}
label {
    display: block;
    font-weight: bold;
    margin-bottom: 10px;
    color: white;
}
input[type="text"],
input[type="password"] {
    background-color: #FEFBD0;
    padding: 10px;
    margin-bottom: 20px;
    width: 95%;
    border-radius: 5px;
    border: none;
    box-shadow: 0px 0px 5px 0px rgba(0,0,0,0.1);
}
button[type="submit"] {
    background-color: #c6aadd;
    color: #424242;;
    padding: 10px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    box-shadow: 0px 0px 5px 0px rgba(0,0,0,0.1);
}
button[type="submit"]:hover {
    background-color: #d3b4b4;
}
small{
    color: white;
}
a{

```

```
color: white;  
}
```

Фиг.12

## ИЗВОДИ:

След като се запознаем със същността на **сървърите**, стигаме до заключението, че те се делят на 2 големи групи – хардуерни и софтуерни. Хардуерният сървър е нормален, докато виртуалният се позиционира върху хардуерен – точно както при РС-тата, виртуалната машина се изгражда софтуерно върху компютъра. Също така, сървърите могат да се подразделят на такива, които имат конкретна задача и на такива, които извършват няколко функции едновременно. Според това, под какво натоварване са подложени, има 3 основни групи сървъри – бизнес клас, обслужващи малка фирма/домашни сървъри и мобилни. Важно да се спомене е, че масово сървърите работят непрестанно. Това е така, защото други потребители зависят от него и дори ако бъде спрян за малък период от време, това може да нанесе щети както финансово, така и репутационно. За правилен избор на сървър, трябва да се прецени каква ще е основната му бъдеща функция, освен това от изключителна важност за правилно и нормално функциониране на сървърите е микроклимата при който трябва да работят. Проследяването на параметрите на микроклимата като температура, влажност и подвижност на въздуха в затворени помещения дава възможност за поддържането им в подходящи граници, които да гарантират добър комфорт, благоприятна работна среда и висока производителност на работещите в помещението. Необходимостта от проследяване на параметрите на микроклимата в последните години е все по-осезаема. От една страна, това е свързано със засиленото прилагане на мерки за повишаване енергийната ефективност на сградите и най-вече с тяхното по-добро изолиране, а от друга - със значителното време, което хората прекарват в затворените пространства. Поддържането на параметрите на микроклимата в подходящи граници гарантира не само благоприятен микроклимат, но и запазване здравето на работещите. Стойностите на параметрите на микроклимата в производствените помещения зависят от редица фактори, сред тях са климатичните условия и сезоните; видът на протичащия технологичен процес и използваното оборудване; условията на въздухообмен; размерът на помещението; броят на работещите и т. н. Също така те могат да варират в рамките на работния ден или да бъдат с различни стойности в отделни участъци на едно и също производствено помещение. Подобни помещения се характеризират със сумарното действие и на трите основни параметъра температура, влажност, подвижност на въздуха.

## **ОБОБЩЕНИЕ:**

Температурата на въздуха, измервана в С, е един от основните параметри, характеризиращи топлинното състояние на микроклимата. Обикновено при измерване на температурата се измерва температурата на въздуха, температурата на ограждащите повърхности, като се отчита тази на ограждащите конструкции – стени, под, таван на различни устройства като екрани например, а така също на технологичното оборудване и други. Температурата на повърхностите и интензивността на топлинното излъчване се отчитат при наличие на съответния източник на топлина.

При определяне влажността на въздуха обикновено се измерва относителната влажност на въздуха. Известно е, че под влажност на въздуха се има предвид съдържанието на водна пари във въздуха. Различават се съответно абсолютна и относителна влажност. Абсолютната влажност отразява количеството водни пари, съдържащи се в единица обем. Измерва се в  $\text{g/m}^3$  и не зависи от околните температура и налягане. Относителната влажност, от своя страна, отразява в проценти съдържанието на водни пари спрямо максимално възможното при същите условия. Максималното количество на водните пари зависи от температурата и от налягането на околната среда.

### **Измерване на температурата**

Уредите, използвани за измерване параметрите на работната среда, имат за цел да подпомогнат поддържането на комфортна среда и при отчитане на отклонение навременно предприемане на корективни действия. Основно изискване към използваните уреди е те да дават точни и коректни данни за параметрите на въздуха. При измерване на температурата като фактор на работната среда приложение намират и безконтактните методи за измерване, работещи на принципа на топлинното излъчване на телата. Използват се предимно за определяне на излъчваната от повърхностите топлинна радиация. Тези измервателни уреди се характеризират със сравнително висока точност на измерване на кратки разстояния. Отчетените показания могат да се използват и като индиректен показател на температурата на въздуха. Сред използваните уреди са и термографите, които дават възможност за непрекъснато измерване на температурата и регистрират изменението ѝ в определен период от време.

Основни елементи в реализираните схеми могат да бъдат различни видове термистори, а на последък масово като основен компонент навлизат микроконтролерите.

### **Измерване на влажност**

За определяне на влажността на въздуха се използват предимно преносими аспирационни психрометри, по-рядко стационарни психрометри и влагомери.

Психрометричният метод се основава на измерването на психрометричната температурна разлика между две температури - тази на “сухия” и тази на “мокрия” термометър. Сухият термометър е в пряк контакт с въздуха, а мокрият термометър е увит с влажна тъкан и към него постоянно постъпва вода по специален фитил. За сметка на изпарението на водата, този термометър се охлажда до температура по-ниска от тази на въздуха. С увеличаването на влажността на въздуха изпарителното охлаждане на термометъра намалява, вследствие на което температурата се повишава, т. е. разликата между двете температури зависи от влажността на въздуха. Съответно, при 100% влажност на въздуха, водата не би се изпарявала и температурите на двата термометъра ще са сравними.

Психрометричният метод се характеризира с висока чувствителност при извършване на измерването в условия на положителни температури. Този метод не се счита за подходящ при измерване на влажността при ниски температури, тъй като при работа в подобни условия се получава висока грешка при измерването.

При отчитането на относителната влажност на въздуха с помощта на тези прибори е необходимо да се вземе предвид разликата в показанията на сухия и мокрия термометър, която е обратнопропорционална на относителната влажност на въздуха. При извършване на измерването е необходимо психрометърът да се постави на определеното място за около 15 минути, след което се отчитат показанията и се определя температурната разлика. Полученият резултат се умножава по корекционен коефициент за движение на въздуха. При аспирационните психрометри подобна корекция не е необходима, тъй като в повечето случаи конструкцията им ограничава въздействието на слънчевата радиация или топлинни излъчвания.

*!Те се характеризират с ефективност и точност на измерването.*

### **Измерване подвижността на въздуха**

За измерване на подвижността на въздуха се използват предимно крилчати и чашкови анемометри. При чашковите анемометри чувствителният елемент е система от полусферични чашки (три или четири на брой), които се въртят около вертикална ос. Принципът им на действие се основава на разликата между съпротивлението при обтичане на вдлъбнатата и изпъкналата част на чашката. Тъй като съпротивлението зависи от числото на Рейнолдс, при измерване на високи скорости е необходима индивидуална градуировка.

При крилчатите анемометри като приемник се използва пропелер (вентилатор). Съответно, броят на оборотите на пропелера за единица време е пропорционален на скоростта на измервания поток.

### **ЗАКЛЮЧЕНИЕ:**

В последно време за определяне на параметрите на микроклимата в производствени и сървърни помещения успешно се използват многофункционални преносими уреди, които позволяват измерването на няколко климатични параметъра. Те са лесни за използване, окомплектовани са с необходимата измервателна апаратура и са подходящи за използване при различни условия. Обикновено са оборудвани със сонди, позволяващи измерването освен на влажността и температурата, така също и на нивото на вредни газове като CO<sub>2</sub>, например. Получените данни могат да се съхраняват в паметта на уреда.

С цел интегриране измерванията на основните параметри на микроклимата в сървърните помещения бе проектирана ефективна схема за поддържане и регулиране на основните характеристики в сървърно помещение чрез микроконтролери /ARDUINO/, програмирани на C++.

## **ИЗТОЧНИЦИ:**

<https://sequelize.org/docs/v6/getting-started/>

<https://docs.npmjs.com/getting-started/>

<https://nodejs.org/ro/docs>

<https://en.wikipedia.org/wiki/RAID/>

[https://en.wikipedia.org/wiki/Server\\_\(computing\)/](https://en.wikipedia.org/wiki/Server_(computing)/)

<https://www.w3schools.com/css/>

<https://www.w3schools.com/js/default.asp>

<https://www.w3schools.com/sql/default.asp>

<https://www.w3schools.com/cpp/default.asp>

<https://www.w3schools.com/html/default.asp>

[https://en.wikipedia.org/wiki/Server\\_room](https://en.wikipedia.org/wiki/Server_room)

[https://en.wikipedia.org/wiki/Server\\_farm](https://en.wikipedia.org/wiki/Server_farm)

[https://en.wikipedia.org/wiki/Blade\\_server](https://en.wikipedia.org/wiki/Blade_server)

[https://en.wikipedia.org/wiki/File\\_server](https://en.wikipedia.org/wiki/File_server)

[https://en.wikipedia.org/wiki/Database\\_server](https://en.wikipedia.org/wiki/Database_server)

[https://en.wikipedia.org/wiki/Virtual\\_private\\_server](https://en.wikipedia.org/wiki/Virtual_private_server)

[https://en.wikipedia.org/wiki/Web\\_server](https://en.wikipedia.org/wiki/Web_server)