

## **Design Patterns in IoT with AI**

Chief Assist. Prof. Mariya Armyanova PhD  
University of Economics - Varna, Varna, Bulgaria  
armianova@ue-varna.bg

### *Abstract*

*Artificial intelligence (AI) relies on IoT to collect data in real time and is needed to enable self-management of devices. In addition to hardware, IoT devices require software and networks to function. Due to the fact that IoT devices transfer and process data, often sensitive data, the most important requirement for them is security. To ensure this, various techniques such as Design Patterns are used at different levels — architectural and for solving specific tasks. Their use has a number of advantages, such as improved security, reliability, adaptability and scalability, and most importantly, easier integration with AI. But there are also a number of limitations related to the limited resources of IoT devices, their performance, the complexity of their implementation and debugging, etc. In order to overcome these limitations, it is necessary to use various strategies, such as transferring heavy operations to the cloud or using federated learning, etc.*

*Keywords: Design patterns, Internet of Things, Architectural patterns, Artificial intelligence*

*JEL Code: C61, C88*

*DOI: 10.56065/IJUSV-ESS/2025.14.2.100*

### **Въведение**

Различни организации и изследователски институти реализират IoT, като се използват многобройните технологии, стандарти, производителите са различни и потребителите са от различни сектори, което често води до конфликт на интересите им. Всичко това поставя разнообразни комплицирани проблеми пред разработването на IoT системите. Проблемите се усложняват и от непрекъснатото нарастване на броя на устройствата включени в глобалната мрежа.

Изкуственият интелект (ИИ) разчита на IoT за събиране на данни в реално време и е нужен, за да позволи самоуправление на устройствата. За функционирането на IoT устройствата освен хардуер, е нужен софтуер и мрежи. Поради факта, че IoT пренасят и обработват данни, често пъти чувствителни, най-важното изискване към тях е сигурността. За да се използва опитът на разработчиците и да се улесни разработката се използват различни похвати като шаблони за проектиране от различни нива – архитектурни, логически и за решаване на конкретни задачи. От използването им има редица предимства като подобряване на сигурността, надеждността, адаптивността, мащабируемостта и най-важното улеснява се интегрирането с ИИ. Но има и редица предизвикателства свързани с ограничените ресурси на IoT устройствата, производителността им, сложността на внедряването и дебъгването им и др. За да могат да се преодолеят тези ограничения е нужно да се използват различни стратегии, като прехвърляне на тежките операции в облака или използване на федеративно обучение и др. Целта на статията е да разгледа шаблоните за IoT с ИИ и да се направи класификация, която да улесни откриването и проучването им от разработчиците.

### **1. Особенности на IoT с ИИ**

Internet of Things (IoT) се разглежда като концепция за обвързването на различни устройства с Интернет така, че да се осигури възможност, те да бъдат разпознавани от други устройства. IoT обединява водещи технологии, като облачни технологии, машинно обучение, изкуствен интелект, големи данни, data lake и интелигентни анализи. Всяка една технология поставя нови предизвикателства пред осигуряване на сигурността на системата. Например технологията data lake е добър подход за съхраняване на големи данни, но при неправилен дизайн и употреба носи много рискове свързани с качеството, сигурността и контрола на

достъпа и използването им (Sulova, 2019). Затова обвързването на устройствата в IoT, особено съчетано и с ИИ за самоуправление, е съпроводено с решаването на голям брой разнообразни проблеми и прилагането на шаблоните за проектиране би могло да подпомогне работата на разработчиците. Често отделните подсистеми и устройства се разработват от независими екипи от разработчици в различни фирми и по различни стандарти. Затова обединяването им в единна система изисква съобразяване с възможността за възникване на множество потенциални проблеми. Едно възможно решение е използването на шаблоните за проектиране, които да позволят многократно използване на опита на разработчиците при решаването на даден проблем. Шаблоните могат да се разглеждат като описания на често срещани проблеми, техните абстрактни решения и последствията им. Използването им до голяма предотвратява внасянето на неочаквани последствия в кода на системата при решаването на конкретния проблем.

За всеки елемент на IoT могат да се използват шаблони. Затова е важно най-напред да се представи модел на IoT платформа и да се намери мястото на ИИ в нея. IoT включва сензори, изчисления, комуникация и задействане.

Повечето автори се придържат към четирислойна архитектура с някои вариации, някои (Katlinsky, 2025) дори предлагат шестслойна архитектура с процесен слой над слоя на приложенията за управление, електронно правителство и други системи. Включва още и отделен слой за сигурност, който обхваща всички слоеве. Стандартните четири слоя са слой на сензорите, мрежови слой, слой на събирането и обработката на данните и слой на приложенията. В стандартната IoT архитектура данните се прехвърлят еднопосочно от сензорите, които могат да са за средата или в устройствата, през различните мрежови среди до хранилищата на данните и модулите за обработката им. Накрая получените резултати се изпращат до потребителите в удобен за ползване вид, през различни устройства, уеб или мобилни приложения.

Придържайки се към предложената четирислойна архитектура може да внесем някои изменения, за да определим мястото на модулите с ИИ за тези устройства (фиг. 1). Можем да добавим един слой, който да се изпълнява на самите устройства. Той може да служи за начална обработка на данните при модел на ИИ, базиран на федеративно обучение (ФО) или криптиране на данните.



Фигура 1. Модел на IoT архитектура.

*Източник: собствена разработка*

Сензорният слой получава данните от устройствата или сензори в средата. Той служи за събиране и изпращане на данни за средата и устройствата, работещи в нея. Събраната информация е ключова за реализация на самоуправление и позволява на устройствата да се настройват към промените в средата.

Следващият слой е разширение на сензорния и може се разгледа като негово продължение. Отделен е в нов слой, защото може да се изпълнява и на друго устройство, например локален сървър. В слоя на локалното МО се извършва първоначалната обработка на събраните данни, а при ФО се изпълняват и някои алгоритми за МО в IoT устройствата. Федеративното обучение (ФО) е разновидност на МО, което се опитва да разреши предизвикателствата пред преноса и обработката на големи данни в IoT.

В някои случаи слоят на локалното МО може да се намира на устройствата, но може да е и на локален шлюз, възел или сървър, който управлява устройствата. Често устройствата имат слаби възможности или пък се събират данни от сензори, които не притежават възможности за сложни изчисления. В слоя, дори и да е избран централизиран модел на МО, се извършва криптиране и начална обработка и обобщаване на данните. Така се намалява трафика към сървъра и се повишава сигурността.

В слоя на локалното МО може да се намира модулът за автономно вземане на решения (Autonomous Decision-Making Module/AI Decision Engine). Той е ключов компонент в IoT системите с ИИ. Той се разполага в самите умни устройства, когато те изпълняват критични функции. Особено важно е в случаите, когато трябва да се осигури автономна работа, надеждност, дори и при слаба или нарушена мрежова връзка. Затова такива устройства имат повишени изчислителни възможности и на тях се изпълнява обработката с модела на ИИ, като те получават някои параметри на модела за обучение от централното звено. Те могат да функционират без постоянна мрежова връзка и да прехвърлят по-малко количество данни.

Следващият слой е комуникационният или мрежовият. По същество той може да се използва и за прехвърляне на данните между първите два слоя. Той осигурява инфраструктурата за прехвърляне на данни между устройствата и облачната инфраструктура. Целта на така определен комуникационен слой е да предава информацията, като гарантира нейната сигурност и надеждност. В комуникационния слой могат да работят различни шлюзове, които да събират и обобщават информация от устройства и локални мрежи. Този слой включва и комуникационните протоколи. Поради ограниченията на мрежовата инфраструктура се разработват, различни протоколи, които да ускорят преноса на данните.

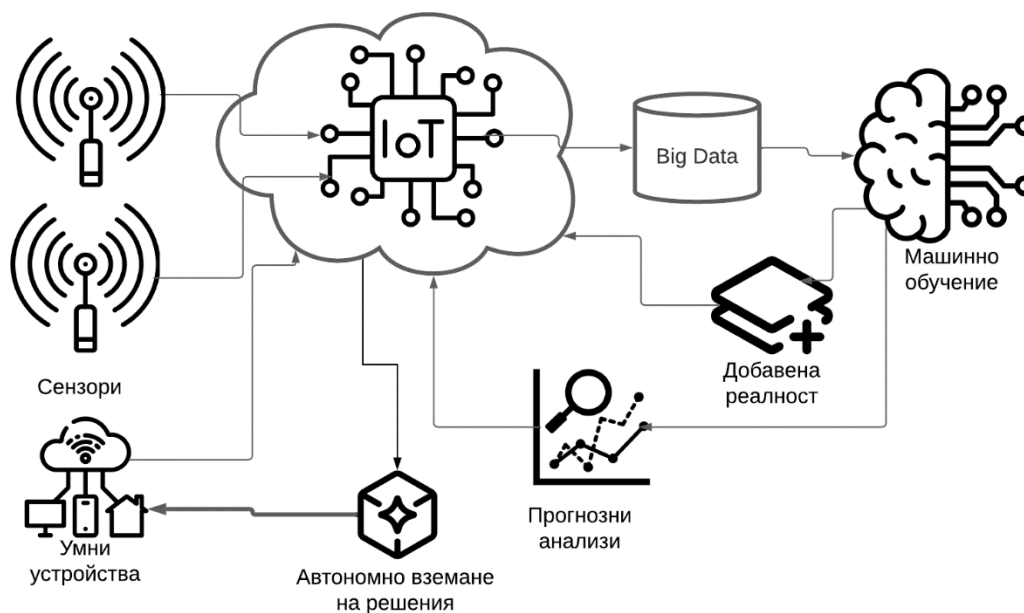
Слоят на системите за обработка картографира получените данни от комуникационния слой (Hatcher et.al., 2021), за да може да локализира различните ресурси. След това данните се подлагат на анализ с моделите на ИИ, базирани на машинно или дълбоко обучение. Резултатите от анализа и прогнозите могат да се използват за управление на устройствата и се предават на слоя на приложенията за крайния потребител. Този слой се изпълнява в инфраструктурата на IoT, която е най-често облачна. Извършва се съхраняване на алгоритмите за обучение, настройките за параметрите им и централното звено, което отговаря за обучението на моделите за обучение на ИИ. В слоя са и всички технологии за съхраняване на големи обеми от данни. Основната цел е да се поеме управлението на цялата система и да се извършат изчисленията, които не са във възможностите на устройствата.

В последния приложен слой потребителят изпраща различни заявки до приложения, за да получи актуална информация и прогнози и чрез приложенията може да въздейства и да управлява реалните устройства. Тук също може да бъде разположен модулът за автономно вземане на решения. Това е особено важно за устройства, чието решение е критично, за да може човешките оператори да оказват контрол върху решенията на ИИ. В този слой се следи за сигурността и контрол на достъпа според правата на потребителите и приложенията.

Предложената многослойна архитектура варира според предназначението на устройствата, възможностите им и избраният модел на ИИ. IoT съчетава много компоненти,

които имат различни задачи. Обобщавайки различните модели на архитектурата може да се създаде концептуален модел (фиг. 2), който представя взаимодействията между основните компоненти на IoT.

Най-напред в IoT се събират данни от устройства сензорите. За прехвърлянето на данните е необходима надеждна връзка, която да позволи включване на устройствата в IoT. Разчита се на технологиите за съхранение и обработка на големите данни, за да се съберат данните от IoT устройствата, които се предоставят на системата за МО. МО се използва за прогнози и обратна връзка и за откриване на стратегии за смекчаване на нежеланите резултати. Трябва да се гарантира сигурността на данните, които са събрани, както и сигурността на преноса им между различните компоненти. Разчита се на протоколи за сигурност при споделяне и механизми за оторизация и удостоверяване.



Фигура 2. Концептуален модел на IoT с ИИ.

*Източник: собствена разработка*

Необходимостта от осигуряване на надеждност, точност, бързина и сигурност при преноса и обработката на непрекъснато нарастващия обем данни поставя много високи изисквания към комуникационните и изчислителните ресурси. Проблемът се усложнява от нуждата за ефективност и мащабиране на обработките. В опит да се решат тези предизвикателства се въвежда идеята за разпределена, а не централизирана обработка на данните и децентрализирано обучение. Целта е да се използват подобряващите се възможности на устройствата, като моделите за МО се обучават с ресурсите на устройствата, без да се налага данните събрани от тях да се изпращат на централизирано място. По този начин се гарантира поверителността и сигурността на данните, които не напускат устройствата в суров вид или дори изобщо не се прехвърлят към централизирано хранилище. Особено необходима е тактиката за сфери, които използват чувствителна информация. Чрез намаляване на необходимостта от преместване на данни, ФО помага да се избегнат забавяния и намалява търсенето в мрежите, което прави възможно анализирането на данните бързо и локално. Освен това ФО може да работи с различни видове и количества данни на много устройства, което помага на IoT системите да се мащабират ефективно, като същевременно запазва информацията за потребителите поверителна и защитена (Yadav et.al, 2022). ФО има потенциал да се използва, защото повишава ефективността, надеждността и гъвкавостта.

## **2. Предизвикателства и възможности за решаването им с шаблони за проектиране**

IoT с ИИ съчетава различни технологии. Прехвърлянето и обработката на данните трябва да са сигурни, надеждни, точни и в реално време. И тук възниква нуждата от ефективно използване на изчислителни ресурси, мрежовите ресурси и ресурсите за съхранение. Могат да възникнат проблеми с хардуера или мрежови повреди, които да намалят надеждността. Включването на голям брой взаимодействащи си устройства предполага възникване на проблеми с претоварването на мрежите и високата сложност на алгоритъма за управление. Възможно решение е използване на механизми за ФО, както и осигуряване на надеждна комуникация.

Шаблоните за проектиране имат различни цели. Едни са свързани с реализиране на определена функционалност в системата, а други със съпровождането ѝ. За да позволят лесното развитие на IoT системите, те следва да отразяват тенденциите в развитието им. Затова те следва да поддържат очакваната им бъдеща функционалност.

Шаблоните за проектиране трябва да позволят да се приложи съществуващият опит на разработчиците при създаването на нови софтуерни решения. Те могат да преследват постигане на различни цели и се използват само в определен контекст. При неправилно разпознаване на контекста или целите на разработката, т.е. използването на шаблоните по неподходящ за създаването им начин, те не носят полза. Дори могат да усложнят и въведат нови проблеми в софтуерното решение. Обаче при правилно откриване и прилагане могат да направят софтуера гъвкав, елегантен и многократно използваем. Разработчиците, могат да прилагат известните им шаблони за проектиране без да се налага да преоткриват работещи решения за известни проблеми. Можем да направим извода, че за да се използват са важни две неща. Първо разработчиците трябва да знаят за съществуването на подходящ шаблон за текущия софтуерен проблем. И второ трябва да познават шаблона, да знаят правилния контекст на прилагането му и ограниченията му. Тъй като шаблоните се създават непрекъснато от различни екипи, които описват опита си при решаването на различни проблеми и се публикуват непрекъснато, е трудно разработчикът да открие шаблон за даден проблем, освен ако преди това не е попаднал на него. Затова непрекъснато актуална област е създаването на класификации, които да обобщят и представят шаблоните, използвани в дадена предметна област. С такава класификация разработчиците лесно могат да открият подходящ шаблон за даден проблем и да потърсят по-задълбочена информация за него, ако потенциално е подходящ за решение на проблема.

Възниква необходимостта някои IoT устройства да работят и самостоятелно без необходимост от непрекъснато управление от интерактивната платформа. Платформата, осигуряваща интеграцията на IoT устройствата, има за задачи да събира данните от отделните подсистеми и да осигури взаимодействието им. Различните подсистеми, като алармата за сигурност например се очаква да могат да работят и при откъсване от Интернет мрежата. Следва да се гарантира, че и когато системите нямат достъп до платформата, те продължават да изпълняват поставените им задачи.

Чрез платформата следва да се осигури взаимодействието между устройствата на различните производители, които използват различни мрежови протоколи и информационни технологии. Платформата следва да осигури единен начин на потребителите да управляват различните подсистеми и устройства, като им осигури общи възможности за настройка. Чрез нея потребителят реализира дистанционен контрол и управление. Различните системи обменят през платформата данни, за да изпълнят ефективно поставените задачи от потребителя. Това налага тя да изпълнява и функцията на шлюз за различните подсистеми и устройства.

При разработката на системите се започва от най-високо ниво и постепенно се детайлизира. Затова е по-лесно да се потърсят шаблони според слоя на архитектурата, чието

разработване подпомагат. Има три нива на абстракция – концептуален, логически и физически модел на архитектурата на системата. Шаблоните, използвани при разработването на концептуалния модел са архитектурни шаблони. Те определят начина на организация и взаимодействие на елементите в цялата система. Подпомагат дефинирането на базовите характеристики и поведение на системата (Richards, 2015). Шаблоните, участващи в разработването на логическия слой на архитектурата или шаблоните за проектиране, имат по-малко влияние – само върху отделен елемент, а не върху цялостното функциониране на системата. Те подпомагат решаването на конкретен проблем на разработката, например за сигурността. Шаблоните, които подпомагат създаването на технологичния (физическия) архитектурен модел участват в разработването на архитектурата на системата от най-ниско ниво и представят модели, които отразяват особеностите на конкретен елемент като интерфейс.

### 3. Шаблони за проектиране за IoT с ИИ и класификацията им.

За областта на IoT са създадени значителен брой шаблони за програмиране. Някои са създадени за други сфери на софтуерни решения и са подходящо за областта на IoT, защото добре адресират проблемите в нея.



Фигура 3. Класификационна схема на категориите IoT шаблони.

*Източник: собствена разработка*

Основната категория шаблони за IoT са архитектурните шаблони (фиг. 3). От архитектурата, предложена в точка 1, може да се направи извод че най-подходяща за IoT е многослойната архитектура. Тя е представена в шаблона Layered. Предимството на използването на този шаблон е по-лесна поддръжка и мащабируемост. Друг шаблон, който се използва при облачните системи е Microservices. При него функционалността се разделя на малки независими услуги, които могат да се актуализират и мащабират независимо.

Освен тези шаблони има специално разработени за IoT (Weyrich & Ebert, 2016). В IoT се въвеждат компоненти, които да позволят създаването на различен тип интеграция и комуникация на компонентите ѝ. Те включват Device-to-Device (D2D), Networking-to-Networking (N2N), Middleware-to-Middleware (MW2MW), Application&Services-to-Application&Services (AS2AS), Data& Semantics-to-Data & Semantics (DS2DS) и защита от различни нива на архитектурата, която използва различни механизми от автентификация на идентификационни данни, използване на маркери за удостоверяване или слой за сигурни сокети (SSL). Включването на едни или други архитектурни елементи се влияе и от предназначението на системата, тъй като системите са разнородни. Например системи, като

умен дом и верига за доставка си имат специфики. Друг архитектурен шаблон (Alreshidi & Ahmad, 2019, p.4) за IoT предлага, разделение на функционалността в три слоя. Потребителският слой позволява на потребителите да следят и управляват дистанционно устройствата. Слойът на комуникация и логика осигурява координацията и съвместната работа на устройствата в мрежата. Слойът на изчисления и съхранение управлява интелигентния анализ и съхранението на данните.

Шаблоните от логическия слой или така наречените шаблони за проектиране решават конкретен проблем в дадена подсистема. Затова шаблоните за проектиране могат да се разделят в следните основни групи, определени от основните категории проблеми: шаблони за интеграция, шаблони за комуникация, шаблони за сигурност, шаблони за ИИ модели, шаблони за енергоспестяване и шаблони за автономно управление.

Шаблоните за **сигурност** имат за цел да предотвратят случайното вмъкване на уязвимости в кода и да смекчат последствията от тези уязвимости. Те обединяват познанията за сигурността и системната структура, като дават възможност за развитие и усъвършенстване на софтуера. Позволяват да се интегрира политиката на сигурност с проектирането и разработването на софтуера. Шаблоните за сигурност са многобройни. Част от проблемите по сигурността са общи за всички системи. Затова при разработката на IoT система могат да се използват както специализирани шаблони за IoT, така и шаблони, които реализират един или друг аспект на сигурността при всички системи.

Освен общите шаблони са създадени и специфични за нуждите на IoT системите. Целта ни е да разгледаме именно тези шаблони. Такива са шаблоните на Reinfurt (Reinfurt et al., 2017). Той предлага шаблоните за проектиране Trusted Communication Partner, Outbound-Only Connection, Permission Control, Personal Zone Hub, Whitelist и Blacklist. Шаблонът Trusted Communication Partner има за цел да предотврати достъпа до устройството или неговата мрежа на случайни комуникационни партньори чрез предоставяне на възможност за комуникация с устройството единствено на списък от разрешени предварително известни партньори. Друго решение за предотвратяване на неоторизиран достъп е предложено в шаблона Outbound-Only Connection, които позволява отговор само на тези входящи заявки за комуникация, които са в отговор на връзка, иницирана от устройството. Шаблонът Permission Control гарантира сигурността на данните на устройството чрез запазване върху сървър на правата за достъп до данните на комуникационните партньори. Personal Zone Hub предполага създаване на хъб, който да позволи управлението на правата, споделянето на данни и контролът на всички устройства на даден потребител. Шаблонът Whitelist предполага създаване на списък с потребители, които имат права за достъп, като никой извън този списък няма достъп. Blacklist е на обратния принцип. Предполага създаването на списък с ненадеждни партньори, на които се отказва правото за достъп.

Като шаблони за сигурност могат да се разглеждат шаблоните на Гама Адаптер (Adapter) и Фасада (Facade), които улесняват интеграцията с различни протоколи и устройства, и осигуряват стандартен интерфейс. Те позволяват да се въведат редица протоколи за действия като оторизиране и криптиране на достъпа до данните. Шаблонът декоратор (Decorator) на Гама добавя функционалност без промяна на основния клас. Така може да се добави действие криптиране или логване за достъп до данните.

Другата основна група шаблони са за **интегриране**, които осигуряват комбинирането на устройствата, разположени на различни крайни възли с различни протоколи в IoT платформа. Koster предлага шаблони, които могат да използват различна мрежова и устройствена технология за решаване на проблеми с физическата инфраструктура на IoT (Koster, 2014). Qanbari (Qanbari et al., 2016) също предлага предлага шаблони за комбиниране на устройствата, които се отнасят за възлите от периферията. Edge Provisioning Pattern осигурява контрол върху голям брой труднодостъпни разпръснати крайни устройства, като осигурява възможност за улесното им преконфигуриране, а също и за лесно включване на

нови. Edge Code Deployment Pattern осигурява възможност за лесно поддържане на софтуера на устройствата чрез децентрализиран Git контрол на версиите, разположен на Provisioning Server. Edge Orchestration Pattern разпределя функционалността така, че оставя възможност крайните възли сами да контролират, конфигурират и управляват крайните си устройства, да следят състоянието им, да откриват нужните им услуги. Edge DOT Pattern изисква създаване на Metering Server, който да уеднакви начина на измерване на извършените услуги в различните крайни възли, тъй като различните доставчици могат да използват различни модели, базирани на събития или базирани на време.

Към шаблоните за интегриране могат да се отнесат и някои класически шаблони на Гама свързани с унифицираното управление на устройства и данни. Например шаблонът Единствен обект (Singleton) осигурява единна точка за достъп до ресурс, като устройство, база от данни или MQTT брокер. А шаблонът Фабрика (Factory или Factory Method) създава достъп до различни видове IoT устройства или съобщения до тях по стандартизиран начин. Шаблонът Команда (Command) позволява изпращане на команди към устройствата по унифициран начин и лесно добавяне на нови команди.

Следващата голяма група шаблони са за **комуникация** Петер (Peter, 2016) предлага такива шаблоните Request/Response, Event Subscription, Asynchronous Messaging, Reliable Messaging, Multicasting, Publish/Subscribe, Queues, Message Brokers, Federation, Discovery and Delegation of Trust. Вместо шаблона Publish/Subscribe на Петер може да се използва класическият шаблон на Гама Наблюдател (Observer). Това е възможно, защото устройства и сензори изпращат данни към заинтересовани страни, като приложения, сървър или облак без директно обвързване.

Reinfurt също предлага шаблони за решаване на проблемите с комуникацията (Reinfurt et al., 2017 b). Device Gateway, Device Shadow, Rules Engine, Device Wakeup Trigger, Remote Lock and Wipe, Delta Update, Remote Device Management, Visible Light Communication. Шаблонът Device Gateway се използва в случаите, когато част от устройствата не поддържат комуникационната технология или протокол на мрежата. Шаблонът предполага използване на посредник, като Gateway, които да превежда протоколите от и към устройството. Понякога вместо шаблона Gateway може да се използва шаблонът Прокси (Proxy) за централизирана точка за управление на множество IoT устройства, филтриране на данни и превод между протоколи.

Device Shadow се използва за изпращане на заявки към устройства, които не са постоянно включени в мрежата. Шаблонът предполага създаване на виртуално обръщение към устройството и обикновено пряко си взаимодейства с шаблоните от другата следващата група. Rules Engine позволява на потребителите да настройват системата чрез съвкупност от прости правила. Device Wakeup Trigger предполага изпращане на съобщение по комуникационния канал до управляващия блок на устройството, което не е постоянно включено в мрежата. Remote Lock and Wipe гарантира сигурност на данните съхранявани на устройството, като позволява при отключване на устройството от системата данните му да се изтриват. Delta Update подпомага намаляването на мрежовия трафик чрез изпращане само на променените от последното изпращане данни на устройството. Remote Device Management позволява локално управление на устройството чрез клиент инсталиран на устройството, който интерпретира сървърните команди. Visible Light Communication предполага изпращане на светлини сигнали за комуникация до или от отдалечено устройство. Jung (2013) въвеждат три шаблона за проектиране, за да се справят с хетерогенността на IoT устройствата.

Нова група са шаблоните за **енергоспестяване**. Те набират все по-голяма актуалност. Reinfurt (Reinfurt et al., 2017) също предлага шаблони за тази категория, като ги разделя на два типа. Първите описват различните типове устройства според изискванията им на енергия и затова не представляват интерес за софтуерното обезпечаване на IoT системата. Вторият тип описват начини за комуникация с устройствата, така че да се спести енергия. Шаблонът

Always-On Device се отнася до случаите, когато пестенето на енергия е неефективно. Другият шаблон Normally-Sleeping Device се използва в случаите, когато не е необходимо устройството да работи непрекъснато и се реализира чрез деактивиране на всички му енергоелементи. При необходимост от работата на такова устройство управляващият блок подава енергия и събужда устройството. Възможно дори управляващият блок да не е постоянно подклучен в мрежата, а да се включва на определени интервали, за да провери за необходимост от събуждане на устройството. Шаблоните, с чиято помощ се реализира комуникацията с подобни устройства от разгледаните са Device Shadow, Device Wakeup Trigger. Cruz и Abreu (2019) обобщават 22 шаблона за енергийна ефективност в мобилни приложения. Тези шаблони могат да бъдат прилагани и в областта на IoT.

Друга голяма група са **шаблоните за ИИ**. Такъв е шаблонът Model Lifecycle Pattern (Washizaki et al., 2022), който осигурява управление на моделите за МО обучение в облака, тестване и внедряване на възела или устройството. Друг е шаблонът за Federated Learning Pattern, който изисква обучение на модели на ИИ директно върху IoT устройства със събраните от тях данни. Така не се прехвърля сурова информация и се повишава сигурността. Шаблонът Anomaly Detection Pipeline Pattern се използва за откриване на аномалии в реално време. Шаблонът Reinforcement Learning Control Loop е за автоматизирана оптимизация, като агент с ИИ управлява устройства според обратна връзка от средата.

IoT се използва за разработката на интелигентен дом, транспорт, град, здравеопазване, вериги за доставка, роботизирана индустрия или интелигентно управление на енергията. Всички тези системи могат автономно да се настройват към средата. В IoT системите с ИИ се предполага да се поддържа автономното управление на устройствата. Затова могат да се използват шаблони от групата за **автономното управление**.

Такава е целта на шаблона Cloudin-the-Loop (Bloom et al., 2018). Той позволява и в двата слоя за локално МО на възела и в слоя за обработка на възела или в облака да се включи интелигентен анализ, базиран на МО и алгоритми за прогнозиране, които подкрепят вземането на решения. Това позволява самоадаптиране на устройствата на няколко нива. Така е възможно да се осъществява мониторинг върху всички устройства. От друга страна децентрализираното, локално обучение на устройствата изисква разпространение на информация в рамките на слоя за локално МО. Такъв пример е реализиран за областта на координиращи се роботи. Два робота разделят производствена задача на подзадачи и могат да реорганизират разделението им в случай на неизправности на робот (Krupitzer et al., 2018). Други подходящи шаблони могат да бъдат за координирано управление, като Master-Slave или регионално планиране. Шаблоните за проектиране Sensor-Factory, както и Content-based Routing (Ramirez, 2008) могат да структурират и да организират информационния поток в разпределени настройки за мониторинг.

Шаблонът за проектиране SAS Analyzer (Abuseta & Swesi, 2015) поддържа приложение на многостепенен подход за анализ чрез възможността за ясно присвояване на съответната функционалност за слоя, както и ефективни механизми за агрегиране за извличане от ситуации на симптоми, например, използвайки подхода за осъзнаване на ситуацията.

На локално ниво Ramirez (2008) предлага няколко шаблона, като шаблонът Case-based Reasoning за адаптиране. Шаблонът Gossip (Fernandez-Marquez et al., 2013) има за цел да постигне споделено споразумение относно стойностите на параметрите, например може да подпомогне вземането на решения в група устройства. За вземане на решения на много нива е подходящ шаблонът Divide and Conquer. Той може да раздели отговорностите и различните обхвати на вземане на решения между нивата, най-вече защото поддържа комбинацията от няколко плана за реконфигурация. Шаблонът TradeOff пък предлага да се избере план, който балансира няколко цели и помага да се балансират решенията за компромиси между нивата.

Шаблонът Digital Pheromones (Fernandez-Marquez et al., 2013) позволява координацията на устройства, включително работи в производството, а шаблонът Collective Sort for MAS

(Gardelli et al., 2007) осигурява клъстериране на информация между слоевете на обучение. Шаблонът Collective Sort (Snyder et al., 2012) предлага проверки за правдоподобност на данните, особено важно за ФО.

Тъй като целта на представянето на шаблоните е да позволи по-лесното им откриване при необходимост в таблица 1 е направено обобщение на основните групи шаблони според класификацията им.

Таблица 1. Списък на шаблоните според целта им.

| Група шаблони  | Примери   |
|--|---|
| Архитектурни шаблони   | Шаблон на Алрашиди и на Уейрик  |
| Шаблона за сигурност:  |   |
| <ul style="list-style-type: none"> <li>Шаблона за сигурност за облачните системи</li> </ul>        | Шаблона на Хафиз, на Шумахер, на Романовски, на Кинзъл, шаблони за симетрично и асиметрично кодиране на достъпа |
| <ul style="list-style-type: none"> <li>Шаблона за сигурност специфични за IoT системите</li> </ul> | Рейнфърт и Гама   |
| Шаблона за интегриране   | Шаблона на Koster, на Кенбъри, на Рейнфърт  |
| Шаблона за комуникация   | Шаблона на Петер, на Рейнфърт и на Джънг  |
| Шаблона за енергоспестяване:   | Шаблона на Рейнфърт и на Круз   |
| Шаблона за ИИ  | Шаблона на Уашизаки   |
| Шаблона за автономно управление  | Шаблона на Блум, на Крупицер, на Рамирес  |

*Източник: собствена разработка*

### 3. Резултати и дискусия

Шаблоните предлагат решения на проблемите за разработка, но при неподходящото им използване могат да доведат до нови проблеми. За почти всеки шаблон съществуват редица ограничения при използването му. За да се преодолеят трябва да се използва подходящ шаблон съгласно контекста, дефиниран при описанието му.

Първата голяма група ограничения са **техническите**. Те са свързани с особеностите на хардуера на устройствата и мрежите. Много IoT устройства нямат достатъчна изчислителна мощ и памет за използване на сложни модели с ИИ, като например реализация на ФО. При ФО устройства трябва да обучават локални модели, което е енергоемко и изисква синхронизация. Микроконтролерите на устройствата са с ограничена памет и не могат да създават много екземпляри от абстракциите. В някои случаи мрежата не е стабилна и не е в състояние своевременно да се прехвърлят данни между реалното устройство и неговата виртуална версия, например при модел на дигитален близък. Разнообразието на протоколи, използвани от различни производители, мрежи, затруднява обмена на данни и се изисква използване на допълнителни шаблони като Адаптер, които забавят обработката и са ресурсоемки.

Други трудности идват от **особеностите на IoT** средите с ИИ. Те се променят динамично, затова моделите им остаряват бързо и изискват чести актуализации. Налага се да се правят непрекъснати актуализации, които са разходи на ресурси. Това води до непрекъснато обучение и управление на версиите на моделите с ИИ. Изисква се сложна инфраструктура, която не винаги е налична при всички IoT устройства. Старите устройства нямат ресурси за поддържане на новите архитектурни подходи. Възможно решение е да се добави шаблон за адаптиране на възможностите, а това води до излишна сложност. ИИ в IoT може да обработва лични данни, което означава проблеми със сигурността, поверителността, регулациите и прозрачността.

Всички тези ограничения са преодолими с използване на подходящи шаблони. Например проблемът за ниските изчислителни възможности на устройствата, може да се преодолее с използване на опростени шаблони, като TinyML. Проблемът с нестабилните мрежи, също е преодолим с обработка на данните локално и използване на асинхронни и кеширащи механизми за комуникация.

За да се преодолеят различните ограничения е нужно да се използват различни стратегии, като прехвърляне на тежките операции в облака или обратното - използване на федеративно обучение и др.

### **Заклучение**

IoT системите с ИИ са силно разпределени системи с хетерогенни хардуерни и софтуерни ресурси. IoT системите подобряват функционалността на приложенията, увеличават достъпа до ресурси, спестяват енергия и намаляват разходите. Но разработването им е съпроводено с множество проблеми. В доклада са представени основните проблемите на IoT системите и възможностите за решаването им с помощта на шаблоните. Целта е да се улеснят разработчиците в откриването на подходящ шаблон, когато те са представени в определени групи. Разработчиците се специализират в конкретна проблематика и разделението на шаблоните според критерия цел позволява те да се фокусират върху нея.

Трябва да обобщим, че използването на шаблоните не преодолява всички ограничения. В някои случаи използването на шаблон без да се съобрази контекстът може да увеличи проблемите. Затова за да се преодолеят различните ограничения е нужно да се използват различни стратегии строго съобразени с дадената ситуация.

### **References**

1. Abuseta, Y. & Swesi, K. (2015). Design patterns for self adaptive systems engineering. arXiv:1508.01330. [Online]. Available: <http://arxiv.org/abs/1508.01330> [Accessed 28/11/2025].
2. Alreshidi, A., & Ahmad, A. (2019). Architecting Software for the Internet of Thing Based Systems. *Future Internet*, 11(7), 153. <https://doi.org/10.3390/fi11070153>
3. Bloom, G., Alsulami, B., Nwafor, E. & Bertolotti, I. C. (2018). Design patterns for the industrial Internet of things. *14th IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, pp. 1-10. doi: 10.1109/WFCS.2018.8402353.
4. Cruz, L. & Abreu. R. (2019). Catalog of energy patterns for mobile applications. *Empirical Softw. Eng.*, vol. 24, no. 4, pp. 2209-2235. <https://doi.org/10.1007/s10664-019-09682-0>
5. Fernandez-Marquez, J. L., Marzo Serugendo, G. Di, Montagna, S., Viroli, M. & Arcos, J. L. (2013). Description and composition of bio-inspired design patterns: A complete overview. *Natural Comput.*, vol. 12, no. 1, pp. 43-67. <https://doi.org/10.1007/s11047-012-9324-y>
6. Gardelli, L., Viroli, M. & Omicini, A. (2007). Design patterns for selforganising systems. *Multi-Agent Systems and Applications V*. Berlin. *Springer*, pp. 123\_132. [https://doi.org/10.1007/978-3-540-75254-7\\_13](https://doi.org/10.1007/978-3-540-75254-7_13).
7. Hatcher, W. G. et.al. (2021). Towards Efficient and Intelligent Internet of Things Search Engine. *IEEE Access*, 9, pp. 15778-15795. doi: 10.1109/ACCESS.2021.3052759.
8. Jung, E., Cho, I. & Kang, S. M. (2013). An agent modeling for overcoming the heterogeneity in the IoT with design patterns. *Proc. MUSIC*, pp. 69-74. [https://doi.org/10.1007/978-3-642-40675-1\\_11](https://doi.org/10.1007/978-3-642-40675-1_11)
9. Karaduman, B. et.al. (2022). An Architecture and Reference Implementation for WSN [Online] Available from: <https://bentleyjoakes.github.io/assets/publications/Karaduman2022-An%20Architecture%20and%20Reference%20Implementation%20for%20WSN-Based%20IoT%20Systems.pdf> [Accessed 28/11/2025].
10. Katlinsky, I. (2025). IoT architecture: key layers, components & use cases. [Online] Available from: <https://www.itransition.com/iot/architecture> [Accessed 28/11/2025].

11. Koster, M., (2014) Design Patterns for an Internet of Things. Available from: <http://community.arm.com/groups/internet-of-things/blog/2014/05/27/design-patterns-for-an-internet-of-things> Available from: <http://iot-datamodels.blogspot.com/2014/05/design-patterns-for-internet-of-things.html> [Accessed 12/08/2025].
12. Krupitzer, C. et.al. (2018). Using spreadsheet-defined rules for reasoning in self-adaptive systems. *PerCom Workshops*, pp. 462-467. ISBN 978-1-5386-3228-4, 978-1-5386-3227-7, 978-1-5386-3226-0.
13. Peter W. Communication Patterns for the Internet of Things, June 1, 2016. Available from: <https://software.intel.com/en-us/articles/communication-patterns-for-the-internet-of-things> [Accessed 12/08/2025].
14. Qanbari, S. et.al. (2016) IoT Design Patterns: Computational Constructs to Design, Build and Engineer Edge Applications. *IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pp. 277-282. doi: 10.1109/IoTDI.2015.18.
15. Reinfurt, L., Breitenbücher, U., Falkenthal, M., Leymann, F. & Riegg, A., (2019) Internet of Things Patterns for Communication and Management. *Springer-Verlag*. Available from: <https://www.iaas.uni-stuttgart.de/publications/ART-2019-08-Internet-of-Things-Patterns-for-Communication-and-Management.pdf> [Accessed 12/08/2025].
16. Ramirez, A. J. (2008). *Design Patterns for Developing Dynamically Adaptive Systems*. East Lansing, MI, USA: Michigan State Univ. ISBN 9781605589718.
17. Reinfurt, L., Breitenbücher, U., Falkenthal, M., Leymann, F. & Riegg, A. (2017) Internet of Things Patterns for Devices [Online] Available from: <https://www.iaas.uni-stuttgart.de/publications/INPROC-2017-15-Internet-of-Things-Patterns-for-Devices.pdf> [Accessed 12/08/2025].
18. Reinfurt, L., Breitenbücher, U., Falkenthal, M., Fremantle, P., & Leymann, F. (2017) Internet of Things Security Patterns. *HILLSIDE Proc. of Conf. on Pattern Lang. of Prog.* Available from: <https://www.hillside.net/plop/2017/papers/proceedings/papers/20-reinfurt.pdf> [Accessed 12/08/2025].
19. Richards, M. (2015). *Software Architecture Patterns*, O'Reilly Media. ISBN: 9781491971437.
20. Snyder, P. L., Valetto, G., Fernandez-Marquez, J. L. & Serugendo, G. D. M. (2012). Augmenting the repertoire of design patterns for self-organized software by reverse engineering a bio-inspired P2P system. *Proc. IEEE 6th Int. Conf. Self-Adapt. Self-Organizing Syst.*, pp. 199\_204. doi: 10.1109/SASO.2012.23.
21. Sulova, S., (2019). The Usage of Data Lake for Business Intelligence Data Analysis. *International Conference Information and communication technologies in business and education*, 18 October. UE-Varna, Science and economics, pp. 135-144. ISBN 978-954-21-1004-0
22. Washizaki, H., Khomh, F., Gueheneuc, Y., Takeuchi, H., Natori, N., Doi, T. & Okuda, S. (2022). Software Engineering Design Patterns for Machine Learning Applications. *Computer* 2022.03, pp. 30-39, vol. 55 ISBN 9781941652183
23. Weyrich, M., Ebert, C. (2016) Reference Architectures for the Internet of Things. *IEEE Softw.* 33, pp. 112–116. ISSN: 1937-4194
24. Yadav, S.P., Bhati, B.S., Mahato, D.P. & Kumar, S. (2022) *Federated Learning for IoT Applications*. Springer. ISBN 978-3-030-85561-1