VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

APLIKAČNÉ MONITOROVANIE IOT ZARIADENÍ

APPLICATION MONITORING OF IOT DEVICES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **PATRIK KRAJČ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. PETR MATOUŠEK, Ph.D., M.A.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

Zadání diplomové práce

Student: **Krajč Patrik, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Počítačové sítě
Název: **Aplikační monitorování IoT zařízení**
Application Monitoring of IoT Devices
Kategorie: Počítačové sítě
Zadání:

1. Seznamte se s monitorování IoT sítí pomocí SNMP.
2. Vytvořte testbed reálných IoT zařízení, která budete dlouhodobě monitorovat pomocí SNMP. Na základě monitorování provozu vytvořte databázi obsahující historická data z monitorovaných IoT objektů.
3. Prostudejte metody pro dolování dat a detekce anomálií. Navrhněte způsob, jak modelovat aplikační data vybraných MIB objektů pro IoT.
4. Implementujte navržený model jakou součást monitorovacího systému.
5. Navrhněte scénáře pro ověření detekce typických anomálií, např. výpadků IoT zařízení, neočekávané změny hodnot, apod.
6. Diskutujte možnost využití navrženého systému pro správu IoT sítí.

Literatura:

- D. Hanes, G. Salqueiro, P. Grossetete, R. Barton, and J. Henry, IoTFundamentals. Networking Technologies, Protocol and Use Cases forthe Internet of Things. Cisco Press, 2017.
- Jiawei Han, Micheline Kamber, and Jian Pei. 2011. *Data Mining: Concepts and Techniques* (3rd. ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- MATOUŠEK Petr, RYŠAVÝ Ondřej and POLČÁK Libor. Unified SNMP Interface for IoT Monitoring. In: *Proceedings of the IEEE/IFIP International Workshop on Internet of Things Management*. Bordeaux.
- Byeongkwan Kang, Sunghoi Park, Tacklim Lee and Sehyun Park, "IoT-based monitoring system using tri-level context making model for smart home services," *2015 IEEE International Conference on Consumer Electronics (ICCE)*, 2015, pp. 198-199.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Matoušek Petr, Ing., Ph.D., M.A.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 18. května 2022
Datum schválení: 26. října 2021

Abstrakt

IoT zariadenia používajú rôzne štandardy na úrovni prenosového média a komunikačného protokolu. Cieľom práce je vytvoriť systém, ktorým dokážeme unifikovať heterogénnu IoT sieť pre účely monitorovania. Na zber dát z IoT siete bola použitá platforma Home Assistant, ktorá využíva nami vytvoreného agenta SNMP. Monitorovací systém pozostáva zo systému Nagios core, ktorý je rozšírený o detekciu anomálií založenej na strojovom učení.

Abstract

IoT devices use various standards at the level of the transmission medium and communication protocol. The aim of the work is to create a system, which we can unify a heterogeneous network of the Internet of Things for monitoring purposes. For data collection from the IoT network was used the Home Assistant platform which is uses SNMP agent we created. The monitoring system includes the Nagios core system, which is extended with machine learning-based anomaly detection.

Kľúčové slová

IoT sieť, systém SNMP, detekcia anomálií, strojové učenie, lineárna regresia, polynomialna regresia, izolačný les, lokálny faktor odľahlosti

Keywords

IoT network, SNMP system, anomaly detection, machine learning, linear regression, polynomial regression, isolation forest, local outlier factor

Citácia

KRAJČ, Patrik. *Aplikačné monitorovanie IoT zariadení.* Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Petr Matoušek, Ph.D., M.A.

Aplikačné monitorovanie IoT zariadení

Prehlásenie

Prohlašuji, že jsem tuto diplomovú práci vypracoval samostatně pod vedením pana Ing. Petra Matouška, Ph.D., M.A.. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Patrik Krajč
17. mája 2022

Poďakovanie

Touto cestou by som sa chcel poďakovať pánovi Ing. Petrovi Matouškovi, Ph.D., M.A., ktorý bol vždy ochotný poskytnúť odbornú pomoc pri vypracovaní práce.

Obsah

Kapitola 1

Úvod

Téma Internet of Things je každým rokom populárnejšia. IoT siete sa dostávajú nie len do domácností, ale aj do firiem či väčších organizácií. Takúto sieť môže tvoriť široké spektrum zariadení, pri ktorých dochádza k hardwarovým obmedzeniam, ktoré majú za následok použitia rôznych komunikačných štandardoch na úrovni prenosového média a to napríklad technológie Wi-Fi alebo ZigBee. Okrem použitia rôznych prenosových médií, IoT zariadenia používajú množstvo aplikačných protokolov na komunikáciu, jedná sa napríklad o protokoly MQTT, CoAP, HTTP.

Cieľom práce ja vytvoriť monitorovací systém, ktorý jednotným spôsobom dokáže monitorovať IoT sieť, bez obmedzenia na použitý komunikačný protokol alebo použité prenosové, ale bežnú IP sieť.

Použijeme platformu Home Assistant, ktorá bude predstavovať centrálny bod IoT siete, voči ktorému budeme vytvárať požiadavky na monitorované zariadenia zo systému Nagios. Ďalším cieľom práce je rozšíriť monitorovací systém o detekciu anomálií v IoT sieťach.

V kapitole 2 si ukážeme architektúru systému SNMP, ktorá zahŕňa oboznámenie s typmi zariadení, vytvorenie MIB databáze a ukážkou, ako môžeme monitorované dáta získať.

Vytvorili sme IoT sieť, ktorej podrobnosti si ukážeme v kapitole 3. Kapitola obsahuje informácie o topológií IoT siete, ale aj časť, ktorá zahŕňa monitorovací systém a to Home Assitenta a systém Nagios. V poslednej časti kapitoly sa oboznámime s použitými zariadeniami a prenášanými informáciami.

Detekciu anomálií si predstavíme v kapitole 5. Ukážeme si rôzne prístupy, ako je napríklad prístup založený na klasifikácií alebo prístup založený na štatistike. Ukážeme si taktiež metódy, ktoré budú použité na detekciu anomálií v našej IoT sieti.

V kapitole 4 si ukážeme akým spôsobom získavame dáta z IoT siete. Použijeme na to nami vytvoreného agenta SNMP, ktorý je integrovaný v prostredí Home Assistent. Získané dáta sú následne klasifikované ako normálne alebo abnormálne chovanie. Túto klasifikáciu vykonáva nástroj Learning core, ktorý si ukážeme v kapitole 6.

Pozrieme sa na vyhodnotenie jednotlivých modelov pre IoT zariadenia a ich presnosť. Tieto informácie sú zhrnuté v kapitole 7. Okrem vyhodnotenia sa budeme venovať aj testovaniu na reálnych dátach. Postupu pri testovaní sa budeme venovať v kapitole 8.

Kapitola 2

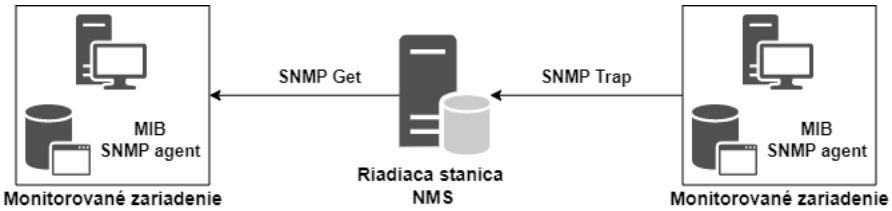
Systém SNMP

Architektúra systému SNMP pozostáva z dvoch typov zariadení. Prvý typ zariadenia predstavuje riadiacu stanicu, ktorú označujeme ako NMS (Network Management Systém). Jedná sa o serverovú aplikáciu, ktorej úlohou je riadiť a monitorovať zariadenia v IP sieti [14].

Druhý typ zariadenia v architektúre SNMP predstavuje monitorované zariadenie. Aby bolo možné zariadenia monitorovať, zariadenie musí mať spustený aktívny proces agenta SNMP. Úlohou agenta SNMP je spravovať SNMP objekty, ktoré predstavujú monitorovanú informáciu na danom zariadení. Monitorované objekty sú združované v MIB databáze (Management Information Base), štruktúru MIB databázy si ukážeme v kapitole 2.1.

Monitorovanie môže prebiehať v dvoch spôsoboch. Prvým spôsob je aktívne dotazovanie NMS na monitorované zariadenie, ktoré inicializuje riadiaca stanica. Monitorovanie môže inicializovať aj agent, ak dokáže rozoznať, že na zariadení vznikla mimoriadna situácia. Komunikácia prebieha prostredníctvom správy typu *trap*.

Na obrázku 2.1 sú znázornené dve monitorované zariadenia spolu s riadiacou stanicou. Monitorované zariadenie sa skladá z fyzického stroja, na ktorom je spustený proces agenta SNMP, ktorý pracuje s MIB databázou. Obrázok znázorňuje dva spôsoby komunikácie. SNMP Get predstavuje dotazovanie riadiacej stanice na aktuálny stav monitorovaného zariadenia. Správou SNMP Trap bežiaci proces agenta SNMP informuje NMS o zmene aktuálneho stavu.



Obr. 2.1: Architektúra SNMP

2.1 MIB databáza

MIB databáza [17] predstavuje virtuálne úložisko pre kolekciu objektov určených na monitorovanie a jej schéma vychádza z SMI (Structure of Management Information) [15].

Štruktúra MIB databáze je reprezentovaná ako acyklický strom, kde každý uzol predstavuje vlastný celočíselný identifikátor, ktorý môže nadobúdať nezáporné čísla.

Na listovej úrovni stromu sa nachádzajú MIB objekty. Každý objekt má vlastný identifikátor, nazývaný OID (object identifier), ktorý predstavujú cestu z koreňového uzlu k listu.

Objekty uložené v MIB databáze rozdeľujeme na dva typy. Môže sa jednať o objekt, ktorý obsahuje iba jednu hodnotu a tento objekt je označovaný ako skalárny. Druhým typom je zložený objekt, ktorý predstavuje položku ktorá sa skladá z už existujúcich MIB objektov. Môžeme vytvoriť napríklad tabuľku, ktorá sa skladá zo sekvencie objektov. Každý MIB objekt musí byť definovanými nasledujúcimi položkami:

- name - definuje názov objektu, ktorý môžeme použiť namiesto OID.
- syntax - definuje dátový typ objektu. SMI definuje jedenásť dátových typov ktoré sú odvodené z ASN.1. Môže sa jednať napríklad o jednoduchý dátový typ INTEGER alebo SEQUENCE OF, ktorá môže predstavovať sekvenciu zložených objektov.
- access mode - táto položka definuje spôsob prístupu k objektu, môžeme nastaviť prístup napríklad iba na čítanie, *Read-only*, prípadne môžeme povoliť aj zápis, *Read-write*.
- status - definuje či daný objekt je nutné implementovať na monitorovanom zariadení. Môže nadobúdať hodnotu *Mandatory*, ktorá definuje, že daný objekt musí byť implementovaný.
- description - táto položka obsahuje textový popisy objektu, jeho použitie a podobne.

Ukážeme si dva príklady vytvorenia MIB objektov. Prvý bude pre skalárnu číselnú veličinu, ktorú pomenujeme *ROInteger*. Definícia objektu nižšie obsahuje syntax INTEGER, ktorý definuje dátový typ pre celočíselné hodnoty. Položka max-access je nastavená tak, aby sme hodnotu objektu mohli iba čítať a status definuje, že daný objekt musí byť na monitorovanom zariadení implementovaný. V poslednom riadku definície môžeme vidieť *Scalars 1*, ktorá definuje predchádzajúci uzol v stromovej štruktúre, na ktorú je objekt viazaný. Pri objektoch, ktoré budú reprezentovať tabuľky bude predchádzajúci uzol *Tables 2*.

```
ROInteger OBJECT-TYPE
    SYNTAX      INTEGER
    MAX-ACCESS  Read-only
    STATUS      Mandatory
    DESCRIPTION "Objekt obsahuje číslenú hodnotu len na čítanie"
    ::= { Scalars 1 }
```

Druhým príkladom bude vytvorenie tabuľky, ktorá bude združovať objekty, obsahujúce jednu číselnú hodnotu, ktorá bude predstavovať index riadku v tabuľke a položku, ktorá bude obsahovať text.

Objekt nazveme *myTable* a budeme potrebovať tri typy definícií. Prvá definícia je určená pre samotnú tabuľku. Rozdiel medzi položkami pre skalárnu veličinu a tabuľku je v položke syntax, ktorá obsahuje hodnotu SEQUENCE OF <object>, a definuje tak postupnosť objektov v tabuľke.

Ďalším krokom je vytvorenie objektu, ktorý bude predstavovať šablónu riadku tabuľky. Tento objekt bude obsahovať syntax s rovnakým názvom ako samotný objekt, ktorý následne musíme definovať. Syntax *myTableEntry* bude obsahovať dva objekty id a text, ktoré definujeme ako skalárny objekt, ktorý sme si ukázali vyššie.

```
myTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF myTableEntry
    MAX-ACCESS  Read-only
    STATUS      Mandatory
    DESCRIPTION "Tabuľka reprezentujúca na každom riadku slovo"
    ::= { Tables 1 }

myTableEntry OBJECT-TYPE
    SYNTAX      myTableEntry
    MAX-ACCESS  Read-only
    STATUS      Mandatory
    DESCRIPTION "Riadok tabuľky"
    INDEX       { id }
    ::= { myTable 1 }

myTableEntry ::= SEQUENCE {
    id       INTEGER,
    text     OCTET STRING
}

text OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  Read-only
    STATUS      Mandatory
    DESCRIPTION "Textová hodnota"
    ::= { myTableEntry 2 }
```

2.2 Prístup k MIB objektom

Aby sme mohli pristupovať k MIB objektom, potrebujeme vytvoriť požiadavku, v ktorej definujem objekt, ktorého hodnota nás zaujíma. Môžeme použiť dva spôsoby. Môžeme definovať postupnosť uzlov oddelených bodkou až k objektu, 1.3.6.1.3.999.1.1.0, alebo môžeme použiť jeho názov z MIB databáze. Ak sa jedná o inštanciu objektu, ktorý nereprezentuje riadok v tabuľke, hodnota objektu je pod identifikátorom s hodnotou nula *ROInteger.0*.

Pri prístupe do tabuľky musíme zadať dlhšiu sekvenciu uzlov, aby sme získali hodnotu z tabuľky. Sekvencia uzlov je rozšírená o hodnotu položky v tabuľke a riadok na ktorom sa nachádza.

```
1.3.6.1.3.999.2.1     - OID prvej tabuľky.
1.3.6.1.3.999.2.1.2   - OID druhého stĺpca v prvej tabuľke.
1.3.6.1.3.999.2.1.2.1 - Hodnota druhého stĺpca z prvého riadku.
```

2.3 Protokol SNMP

Protokol SNMP [6] má tri verzie, z ktorých si predstavíme práve dve. Jedná sa o SNMPv2c [7] a SNMPv3 [4]. Protokol SNMPv2c je založený na komunitách, ktoré slúžia ako autentifikačný parameter voči agentovi SNMP. Aplikácia ktorá má k dispozícií názov komunity, môže pristupovať k MIB objektom.

Protokol SNMPv3 nie je založený na komunitách, ale na vytváraní politiky pomocou užívateľov a užívateľských skupín. Umožňuje nám to lepšiu správu a flexibilitu pri nastavení rôznych prístupov pre skupiny či užívateľov, pričom môžeme definovať aj spôsob overenia užívateľa zadaním hesla.

2.4 Zhrnutie

V tejto kapitole sme sa oboznámili so systémom SNMP a jeho architektúrou, ukázali sme si ako sa vytvára MIB databáza, a ako môžeme získať monitorované informácie.

Pri ďalšom postupe budeme používať experimentálnu vetvu MIB databázy, v ktorej môžeme vytvoriť vlastné MIB objekty, ktoré bude spravovať agent SNMP. Aby sme vytvorili zabezpečenie zo strany SNMP agenta môžeme použiť protokol SNMPv2c, alebo protokol SNMPv3.

Kapitola 3

IoT sieť

V tejto kapitole si ukážeme nami vytvorenú IoT sieť, ktorá sa bude skladať z dvoch typov sieti. Jedná sa o sieť typu PAN, personal area network, a Wi-Fi sieť.

Každá sieť obsahuje rôzne typy senzorov, ktoré komunikujú voči centrálnemu bodu, Home Assisentovi. IoT zariadenia používajú na komunikáciu s Home Assistentom dva aplikačné protokoly: MQTT a HTTP.

Ukážeme si aké zariadenia máme zapojené v IoT sieti a prejdeme dáta, ktoré sú prenášané. Okrem prenášaných dát sa budeme venovať, ako tieto dáta interpretuje Home Assistent.

3.1 Topológia

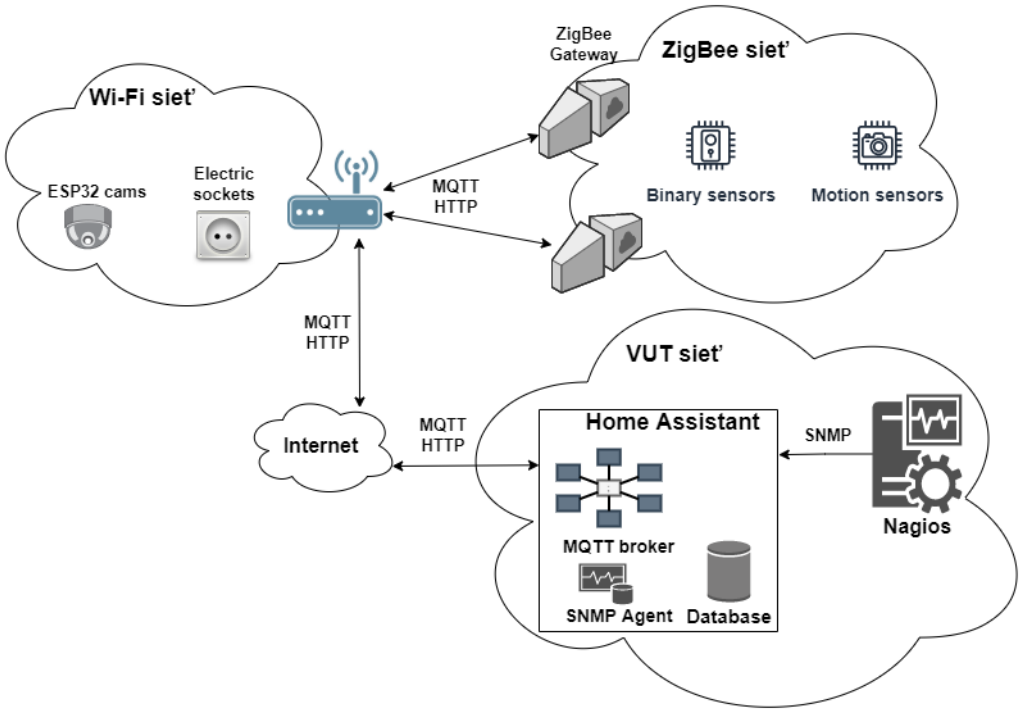
Centrálnym bodom našej topológie je Home Assitant a komunikácia medzi zariadeniami prebieha v troch rozdielnych sieťach, ktoré sú zobrazené na obrázku 3.1.

Ako prvú sieť si predstavíme sieť, ktorá používa technológiu ZigBee. ZigBee technológií je venovaná kapitola 3.2. V tejto sieti máme zapojené tri typy senzorov. Binárny senzor, ktorý monitoruje stav dverí a okien. Pohybový senzor, ktorý monitoruje pohyb, ale aj svietivosť v miestnosti. Posledné zariadenie, ktoré máme zapojené v tejto sieti je Sonoff ZigBee bridge, ktorý slúži ako brána do Wi-Fi siete a koordinátor v ZigBee sieti.

Druhá sieť je Wi-Fi sieť, v ktorej komunikujú zariadenia prostredníctvom protokolu MQTT a HTTP. Všetky požiadavky zo zariadení komunikujúcich protokolom MQTT majú spoločnú cieľovú adresu, a to adresu MQTT brokera, ktorý má pridelenú verejnú adresu v prostredí Home Assistenta vo VUT sieti.

Wi-Fi sieť pozostáva z ESP32 kamier, elektrických zásuviek a ZigBee brán. ZigBee brána umožňuje komunikovať zariadeniam zo ZigBee siete s Home Assistentom, a to prostredníctvom protokolu TCP/HTTP, ale aj protokolom MQTT. Aplikačný protokol, ktorý používa zariadenie závisí od použitého firmwaru.

Poslednou sieťou je VUT sieť, ktorá sa skladá z dvoch zariadení, spomínaný Home Assistent a monitorovacia stanica systému Nagios, na ktorej prebieha zber dát. Okrem zberu dát bude systém Nagios plniť ešte jednu úlohu. Systém Nagios bude rozšírený o nástroj, ktorý bude na základe trénovaných modelov vyhodnocovať prichádzajúce dátové vzorky.



Obr. 3.1: IoT sieť

3.2 ZigBee

Jednou z použitých technológií v našej IoT sieti je ZigBee. Tento štandard patrí do triedy s označením *IEEE 802*, ktorý popisuje siete LAN (Local Area Network) a MAN (Metropolitan Area Network), konkrétne sa jedná o štandard *IEEE 802.15.4* [3], ktorý je zameraný na PAN (Personal Area Network) siete. K tomuto štandardu patrí napríklad aj technológia *Bluetooth*.

Cieľom tohto štandardu bolo vytvoriť bezdrôtový komunikačný protokol, ktorý bude využívať, čo najmenšie hardwarové zdroje, s dostatočnou efektivitou, aby bolo možné preniesť spoľahlivo dáta zo zariadení s minimálnymi hardwarovými požiadavkami, ako sú napríklad senzory [19].

Štandard rozdeľuje zariadenia do troch skupín, podľa ich funkčnosti a hardwarových nárokov.

- Koordinátor - v každej sieti sa nachádza len jedno zariadenie, ktoré označujeme ako koordinátor. Jeho úlohou je udržovať informácie o sieti a slúži aj ako centrálna autorita, ktorá združuje bezpečnostné kľúče. Koordinátor musí byť vždy dostupný a musí si uchovávať informácie/správy o zariadeniach, ktoré môžu byť neaktívne.
- Smerovač - ak to umožňujú hardwarové zdroje, zariadenia môžu používať aplikačnú funkcionalitu smerovača. Zariadenia s touto funkcionalitou môžu prijímať a vysielať správy od ostatných zariadení a zvýšiť tak dosah siete. Smerovač musí byť aktívny a musí si uchovávať informácie/správy o zariadeniach, ktoré môžu byť neaktívne.

- Koncové zariadenie - tento typ zariadenia potrebuje najmenej hardwarových zdrojov. Môže príjímať a vysielať správy, ale nedokáže opakovať/smerovať prichádzajúce správy. Koncové zariadenie musí byť pripojené ku koordinátorovi alebo smerovaču, ktorý si bude uchovávať jeho správy.

.

3.3 Zariadenia

V predchádzajúcich častiach sme si predstavili technológie, ktoré sme použili pri vytvorení našej IoT siete a naznačili sme, ako vyzerá zapojenie senzorov v sieti.

V tejto časti si ukážeme aké dáta prenášajú jednotlivé senzory a spôsob ich komunikácie. Niektoré zariadenia nemajú pôvodný firmware. Pôvodný firmware bol nahradený open source firmwarom Tasmota, ktorý nám umožňuje relatívnu flexibilitu v nastaveniach senzorov, aktualizáciu a kalibráciu senzorov. Tento firmware nám umožňuje používať na komunikáciu protokol MQTT/HTTP s Home Assistentom.

Aby sme umožnili zariadeniam komunikovať s Home Assistentom, musíme vykonať niekoľko nastavení na zariadení. Po úspešnej inštalácií firmwaru, zariadenie pozostáva z webového servera, ktorého rozhranie slúži na prvotnú konfiguráciu.

Prvotná konfigurácia spočíva v nastavení prístupového bodu, ktorú vykonáme tak, že sa pripojíme na zariadenie cez Wi-Fi na sieť s SSID, ktoré sa skladá zo slova tasmota a MAC adresy zariadenia.

Po úspešnom pripojení zariadenia do IoT siete, musíme nastaviť adresu MQTT brokra, užívateľa, pod ktorým sa bude prihlasovať a heslo. Aby sme mohli identifikovať o aké zariadenie sa jedná, je vhodné nastaviť položku *friendly name* a *TOPIC*, ktorý slúži ako identifikácia zdroja informácií, pod ktorým sa registruje zariadenie voči MQTT brokeru.

3.3.1 Blitzwolf SHP6 zásuvka

Elektrické zásuvky od firmy Blitzwolf majú nainštalovaný firmware Tasmota, ktorý bol spomenutý vyššie. Zariadenia majú nastavený MQTT broker, ktorý je spustený na Home assistentovi.

Pre podrobnejšiu analýzu prenášaných dát sme vytvorili súbor s názvom *mqtt.pcap*, ktorý obsahuje komunikáciu priamo na Home Assistentovi. Použili sme na to nástroj tcpdump.

Dáta prenášané zo zariadenia k Home Assistentovi sú vo formáte JSON[1] a posiela dva typy správ: *SENSOR a STATE*. Okrem správ, v ktorých sú prenášané informácie o senzore, si zariadenia vymieňajú *keep-alive* správy, ktoré sú označené ako *PING*.

Ako prvú si ukážeme správu, ktorá je označená ako *SENSOR*. Prenášané dáta sa skladajú z časovej značky a objektu s názvom *ENERGY*. Tento objekt sa skladá z jedenástich položiek. Obsahuje fyzikálne veličiny, ako je napríklad napätie alebo prúd. Okrem fyzikálnych veličín obsahuje informácie o spotrebovanej energii za aktuálny deň, predchádzajúci deň a celkovú spotrebu elektrickej energie.

```
{
  "Time": "2021-11-21T13: 30:35",
  "ENERGY": {
```

[1] viz https://en.wikipedia.org/wiki/JSON

```
    "TotalStartTime": "2020-07-08T14:31:04",
    "Total": 329.902,
    "Yesterday": 1.858,
    "Today": 1.029,
    "Period": 6,
    "Power": 66,
    "ApparentPower": 83,
    "ReactivePower": 51,
    "Factor": 0.79,
    "Voltage": 233,
    "Current": 0.358
  }
}
```


Nižšie môžeme vidieť príklad správy, ktorá je označená ako *STATE*. V tejto správe sa prenášajú všeobecné informácie o zariadení a jeho chovaní. Skladá sa z objektu Wi-Fi, ktorý obsahuje informácie, ako je názov prístupového bodu, MAC adresu prístupového bodu alebo intenzitu signálu.


```
{
   "Time":"2021-11-21T13: 30:35",
   "Uptime":"28T14:50:45",
   "UptimeSec":2469945,
   "Heap":31,
   "SleepMode":"Dynamic",
   "Sleep":50,
   "LoadAvg":19,
   "MqttCount":20,
   "POWER":"ON",
   "Wi-fi":{
      "AP":1,
      "SSID":"TEST IoT",
      "BSSId":"A8:5E:Ð45:AE:7E:61",
      "Channel":5,
      "RSSI":72,
      "Signal":-64,
      "LinkCount":15,
      "Downtime":"0T06:59:22"
   }
}
```


Teraz si ukážeme, ako dáta interpretuje Home Assistent. Každá položka z objektu *ENERGY* predstavuje v Home Assistentovi samostatný senzor. Pri interpretácií týchto senzorov Home Assistent z jednej položky v objekte *ENERGY* vytvorí objekt, ktorý obsahuje časové značky poslednej aktualizácie a poslednej zmeny. JSON objekt *attributes* obsahuje podrobnejšie informácie o senzore, môže obsahovať napríklad typ senzoru a fyzikálnu jednotku monitorovanej veličiny, ktorá je uložená v položke state.

```
{
   "entity_id":"sensor.shp6_energy_power",
   "state":"10",
   "attributes":{
      "unit_of_measurement":"W",
      "friendly_name":"SHP6 ENERGY Power",
      "device_class":"power"
   },
   "last_changed":"2021-11-26T09:30:50.722392+00:00",
   "last_updated":"2021-11-26T09:30:50.722392+00:00",
   "context":{
      "id":"b34862ea150dbcdd387b867af6eb73e2",
      "parent_id":null,
      "user_id":null
   }
}
```


Posledným príkladom pre zásuvku SHP6 bude interpretácia správy *STATUS* v Home Assistentovi. Podobne, ako to bolo v príklade predchádzajúcej správy, aj tu sa nachádzajú v objekte attributes informácie o Wi-Fi, aktuálne pridelenej IP adresy či nainštalovanom firmwaru. Ako sme spomenuli vyššie, položka state vyjadruje určitú veličinu, v tomto prípade state predstavuje intenzitu signálu, RSSI.


```
{
   "entity_id":"sensor.shp6_status_2",
   "state":"60",
   "attributes":{
      "Version":"8.1.0.2(tasmota)",
      "BuildDateTime":"2019-12-30T19:07:34",
      "Core":"2_6_1",
      "SDK":"2.2.2-dev(38a443e)",
      "Module":"BlitzWolf SHP",
      "RestartReason":"Power on",
      "Uptime":"35T18:57:04",
      "WiFi LinkCount":19,
      "WiFi Downtime":"0T18:08:52",
      "MqttCount":24,
      "BootCount":12,
      "SaveCount":435,
      "IPAddress":"192.168.11.220",
      "RSSI":"60",
      "LoadAvg":19,
      "unit_of_measurement":" ",
      "friendly_name":"SHP6 status",
      "icon":"mdi:information-outline"
   },
   "last_changed":"2021-11-28T16:55:32.315938+00:00",
   "last_updated":"2021-11-28T16:55:32.315938+00:00",
   "context":{
```

```
      "id":"2c77537183aa788ae014c648ac575140",
      "parent_id":null,
      "user_id":null
   }
}
```


3.3.2 ESP32 kamera

Pri analýze dát pre ESP32 kameru budeme postupovať rovnako, ako pri Blitzwolf SHP6 zásuvke. Odchytávanie komunikácie medzi zariadením a Home Assistentom bude prebiehať na Home Assistentovi a následne porovnáme zachytené dáta s interpretáciou Home assistenta.

Podobne, ako pri elektrickej zásuvke z kapitoly 3.3.1, sú prenášané dva typy správ *SENSOR* a *STATE*. Správy označené, ako *STATE* majú podobný formát ako spomínané zásuvky, obsahuje JSON objekt s informáciami o pripojení k Wi-Fi sieti.


```
{
   "Time":"2021-11-28T19:08:57",
   "Uptime":"0T00:25:19",
   "UptimeSec":1519,
   "Heap":97,
   "SleepMode":"Dynamic",
   "Sleep":50,
   "LoadAvg":19,
   "MqttCount":4,
   "Wifi":{
      "AP":1,
      "SSId":"TEST IoT",
      "BSSId":"BC:3D:85:4E:A2:D1",
      "Channel":6,
      "Mode":"11",
      "RSSI":92,
      "Signal":-54,
      "LinkCount":1,
      "Downtime":"0T00:00:13"
   }
}
```


Dáta prenášane pod správou *SENSOR* obsahujú informácie o aktuálnej teplote čipu a meranej fyzikálnej veličiny, ktorá je v tomto prípade stupeň Celzia, °C.


```
{
   "Time":"2021-11-28T18:58:57",
   "ESP32":{
      "Temperature":43.3
   },
   "TempUnit":"C"
}
```


Nižšie máme zobrazený príklad interpretácie dát z Home Assistenta.

```
{
   "entity_id":"sensor.esp32_cam01_esp32_temperature",
   "state":"43.3",
   "attributes":{
      "state_class":"measurement",
      "unit_of_measurement":"\u00b0C",
      "friendly_name":"ESP32 CAM01 ESP32 Temperature",
      "device_class":"temperature"
   }
}
```


3.3.3 Pohybový a magnetický senzor

Pohybové a magnetické senzory komunikujú prostredníctvom ZigBee siete so ZigBee bránou, ktorej úlohou je poskytnúť informácie Home Assistentovi. Nižšie zobrazená komunikácia bude predstavovať komunikáciu medzi ZigBee bránou a Home Assistentom namiesto priamej komunikácie jednotlivých senzorov.

ZigBee brána obsahuje firmware *Tasmota*, ale v tomto prípade sa nejedná o MQTT komunikáciu medzi bránou a Home Assistentom. Medzi bránou a Home Assistentom je vytvorené TCP spojenie, v ktorom si prostredníctvom protokolu HTTP vymieňajú aktuálne informácie o aktívnych zariadeniach.

Pohybový senzor interpretuje Home Assistant, ako tri samostatné senzory, pohybový senzor, senzor monitorujúci svietivosť a stav batérie.

Magnetický senzor interpretuje ako dva samostatné senzory, senzor batérie podobne, ako pri pohybovom senzore a aktuálny stav zopnutia. Pohybový a magnetický senzor nadobúdajú iba dve hodnoty, 0/1, ktoré sú interpretované, ako on/off pri pohybovom senzore a open/closed pri magnetickom senzore. Príklad získaných dat z pohybového senzoru je zobrazený nižšie.


```
{
  "entity_id":"binary_sensor.lumi_lumi_sensor_motion_aq2_occupancy",
  "state":"on",
  "attributes":{
     "device_class":"occupancy",
     "friendly_name":"MotionSensorTest occupancy"
  },
  "last_changed":"2022-01-16T18:51:56.673366+00:00",
  "last_updated":"2022-01-16T18:51:56.673366+00:00",
  "context":{
     "id":"96517d40d4b59699a98776f3b7579d19",
     "parent_id":null,
     "user_id":null
  }
}

{
  "entity_id":"sensor.lumi_lumi_sensor_motion_aq2_power",
  "state":"100",
  "attributes":{
```

```
      "state_class":"measurement",
      "battery_voltage":3.24,
      "unit_of_measurement":"%",
      "device_class":"battery",
      "friendly_name":"MotionSensorTest power"
   },
   "last_changed":"2022-01-16T18:51:56.673669+00:00",
   "last_updated":"2022-01-16T18:51:56.673669+00:00",
   "context":{
      "id":"853ec6a0dadb451832d2c1536ca85a5a",
      "parent_id":null,
      "user_id":null
   }
}
```


3.4 Zhrnutie

V kapitole sme si predstavili IoT sieť a zariadenia, ktoré budeme monitorovať. Použitá IoT sieť sa skladá z dvoch typov sieti: Wi-FI a Zigbee siete.

Aby sme vytvorili spojenie medzi Home Assistentom a ZigBee sieťou, použili sme ZigBee bránu, ktorá komunikuje prostredníctvom protokolu HTTP.

Ostatné zariadenia vo Wi-Fi sieti komunikujú prostredníctvom aplikačného protokolu MQTT s Home Assistentom.

Ukázali sme aké informácie môžeme získať z monitorovaných zariadení a akým spôsobom ich interpretuje Home Assistant.

Na základe týchto informácií, môžeme vytvoriť objekty MIB databáze, ktoré budeme monitorovať agentom SNMP a zbierať ich prostredníctvom systému Nagios.

Kapitola 4

Monitorovanie

Na monitorovanie IoT siete sme vytvoril kontajnerovú aplikáciu pre Home Assistant, ktorú je možné nainštalovať ako Add-on. Aplikáciu rozdelíme na dva funkčné bloky. Prvý blok nazveme ako manažment, ktorý slúži na správu agenta SNMP, druhý blok ako agent, ktorý slúži na získavanie informácií. Predstavíme si ešte plugin pre systém Nagios, ktorým budeme vytvárať požiadavky pre spomínaného agenta SNMP.

4.1 Správa SNMP agenta

V tejto časti si ukážeme akým spôsobom komunikuje Add-on s Home Assistentom a ako získané informácie použijeme pre nastavenie agenta.

Táto časť predstavuje webový server, ktorý komunikuje s Home Assistentom. Bola použitá technológia *ingress*, ktorá umožňuje komunikáciu s webovým serverom iba z prostredia Home Assistenta. Rozhranie Home Assitenta nám umožňuje získať informácie o všetkých senzoroch cez API **http://supervisor/core/api/states**, ktorého odpovede na všetky požiadavky sú vo formáte JSON.

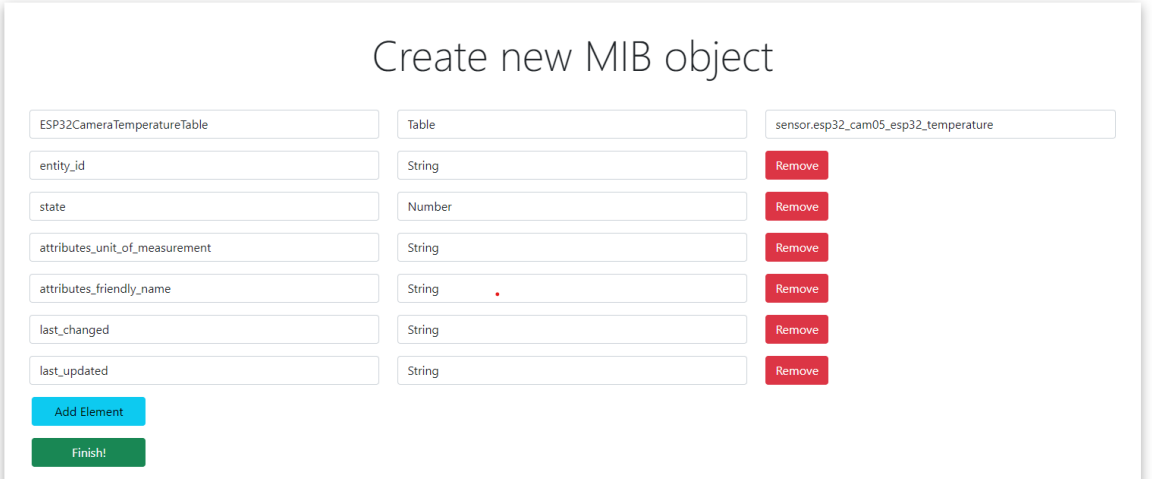
Máme k dispozícií informácie o všetkých pripojených senzoroch. Tieto informácie použijeme na prípravu konfiguračného súboru agenta, ktorý je vo formáte JSON.

Konfiguračný súbor slúži na vytvorenie MIB databáze a zároveň zoznam senzorov, priradených ku konkrétnemu MIB objektu. Konfiguračný súbor obsahuje dve hlavné položky, ktoré sú reprezentované ako pole objektov.

Prvá položka má názov *tables*. Položka obsahuje informácie o všetkých tabuľkách, ktoré budú súčasťou MIB databáze. Každý objekt v položke tables sa skladá z názvu tabuľky, položky attributes, ktorá je reprezentovaná ako pole objektov, ktoré definujú názov a typ atribútu a položky, ktorá označuje či daný atribút bude nutné monitorovať nástrojom Nagios. Názov a typ atribútu definuje syntax MIB objektov použitých pri vytváraní tabuľky. Posledná položka senzors, predstavuje zoznam senzorov, ktoré patria ku konkrétnej tabuľke.

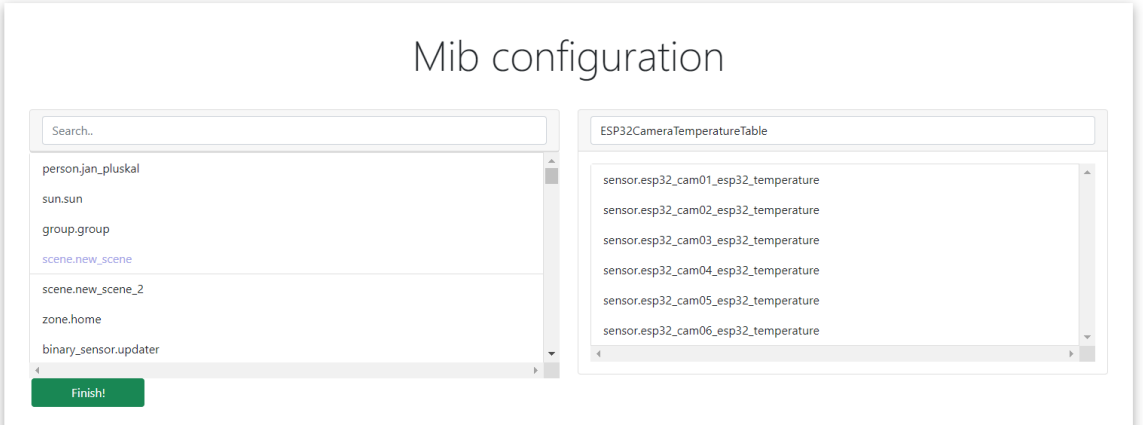
Posledná položka má názov taktiež senzors. Táto položka nám slúži len informačne. Obsahuje senzory, ktoré monitorujeme a môžeme ju použiť na vytvorenie zoznamu senzorov, ktoré ešte nemáme monitorované.

Prvá položka v grafickom rozhraní s názvom Vytváranie MIB, slúži na vytvorenie záznamu MIB objektu v konfiguračnom súbore. Môžeme vytvárať objekty podľa už existujúcich senzorov, prípadne si môžeme na definovať vlastné parametre. Ukážku môžeme vidieť na obrázku 4.1.



Obr. 4.1: Vytváranie MIB objektov

Akonáhle máme vytvorené MIB objekty, musíme k týmto objektom priradiť senzory, ktoré chceme monitorovať. Môžeme to urobiť cez ďalšiu záložku, v ktorej si vieme vyhľadať senzory a priradiť ich k danému MIB objektu. Príklad pridávania senzorov do tabuľky je znázornený na obrázku 4.2, kde sme priradili do tabuľky *ESP32CameraTemperatureTable* šesť senzorov.



Obr. 4.2: Priradenie senzorov k MIB objektom

Ak nepotrebujeme monitorovať všetky položky, môžeme ich vypnúť a v poslednom kroku sa nemám nevygenerujú služby pre konfiguráciu systému Nagios.

4.2 SNMP agent

Podobne, ako grafické rozhranie aj agent je napísaný v jazyku Node JS. Aplikácia používa voľne dostupnú knižnicu net-snmp[1].

[1] viz https://www.npmjs.com/package/net-snmp

Inicializácia agenta prebieha pri spustení Add-onu, prípadne jeho reštartu. Inicializácia prebieha na základe informácií z konfiguračného súboru, ktorý bol vytvorený v grafickom rozhraní.

Z konfiguračného súboru získa informácie o všetkých MIB objektoch, ktorým následne priradí OID z experimentálnej vetvy SMI. Každý objekt, ktorý predstavuje tabuľku má pridanú položku, ktorá slúži na indexáciu riadkov tabuľky. Hodnoty objektov získava agent podobne ako webová aplikácia, s rozdielom, že sa dotazujeme na konkrétny jeden objekt, **http://supervisor/core/api/states/{senzor}**.

Agent SNMP používa protokol SNMPv3. Pri použití SNMPv3 musíme definovať užívateľa a heslo. Ďalším parametrom, ktorým môžeme ovplyvniť chovanie agenta je interval, v ktorom sa dotazuje na objekty definované v MIB databáze. Pred definovaná hodnota je 3000 ms.

4.3 Nagios plugin

Vytvorili sme plugin s názvom *check_oid*, ktorý nám umožňuje vytvárať požiadavky na objekty SNMP v experimentálnej vetve. Plugin je implementovaný ako skript pre unixový shell Bash. Aby sme mohli tento skript použiť je požadovaná pre rekvizita, a to balíček *net-snpm-utils*, ktorý obsahuje program *snmpget* a program *net-cat*.

Plugin môžeme spustiť s nasledujúcimi prepínačmi, niektoré sú nevyhnutné pre spustenie:

- -H prepínač je určený na získanie IP adresy zariadenia, ktoré budeme monitorovať. Môžeme použiť IPv4 a IPv6 adresu.
- -o prepínačom definujem OID objektu, ktorý budeme monitorovať.
- -l prepínač určuje úroveň bezpečnosti počas komunikácie. Môže nadobúdať tri hodnoty, noAuthNoPriv, authNoPriv a authPriv.
- -u prepínač slúží na definíciu užívateľa v prípade SNMPv3.
- -A prepínač definuje frázu pre autentifikáciu.
- -a prepínač definuje autentifikačný algoritmus. Môžeme použiť MD5 alebo SHA.
- -X prepínač definuje frázu pre zabezpečenie dôvernosti správ.
- -x prepínač definuje protokol, ktorým zašifrujeme správy, aby sme zabezpečili dôvernosť správ. Môžeme použiť algoritmus DES alebo AES.

Nasledujúce parametre nie sú povinné. Slúžia len na definovanie formátu správy pre systém nagios.

- -pv určuje jednotku, ktorá bude zobrazená za získanou hodnotou.
- -m predstavuje doplnenie správy, ktorá bude zobrazená v systéme Nagios, môže sa jednať napríklad o názov veličiny.
- -s týmto prepínačom môžeme definovať názov senzoru, použijeme ho ako identifikátor služby systému Nagios.

Plugin získava hodnoty z Home Assistenta pomocou programu snmpget. Výstup programu snmpget spracuje regulárnymi výrazmi.

Získané dáta posiela na vyhodnotenie nástroju Learning core, ktorý si predstavíme v kapitole 6. Nástroj spracuje prichádzajúcu správu, a odpovedá správou, ktorá obsahuje hodnotu, ktorú použije plugin, ako návratový kód pri jeho ukončení. Na základe návratového kódu, systém Nagios vyhodnotí o aký typ udalosti sa jedná. Návratové kódy a udalosti môžeme vidieť nižšie.

- OK - 0
- WARNING - 1
- CRITICAL - 2
- UNKNOWN - 3

Plugin plní ešte jednu úlohou, a tou je vytváranie historických dát pre samostatné senzory. Dáta sa ukladajú do adresára */var/log/nagios/{dátum}/{OID}.log*. Každý riadok v logovacom súbore sa skladá z troch častí, dátumu a času zápisu, hodnoty a návratového kódu v textovej podobe. Aby sme nevytvárali príliš veľké množstvo opakujúcich záznamov, plugin pred zápisom skontroluje či nastala zmena voči predchádzajúcemu stavu. Ak nastala zmena pridá nový záznam na koniec súboru.

4.4 Získavanie dát

Akonáhle máme pripravený plugin na monitorovanie SNMP objektov, musíme ho integrovať do systému Nagios. Konfiguračný súbor systému Nagios definuje umiestnenie používaných nástrojov na monitorovanie, jedná sa o adresár */usr/local/nagios/libexec*, ktorý obsahuje z kompilované a spustiteľné programy, medzi ktoré pridáme nami vytvorený plugin *check_oid*.

V nasledujúcom kroku si ukážeme, ako vytvoríme príkaz pre systém Nagios. Príkaz v systéme Nagios predstavuje popis spôsobu, ako bude vykonaná kontrola. Zoznam príkazov je definovaný v konfiguračnom súbore *commands.cfg*, ktorý je uložený v adresári */usr/local/nagios/objects/*. Súbor obsahuje definície príkazov, ako je napríklad *PING* a podobne. Príkaz sa skladá z dvoch direktív [2]:

- command_name - predstavuje identifikátor príkazu, na ktorý sa môžeme odkazovať z definícií služieb a podobne.
- command_line - direktíva definuje činnosť, ktorá sa má vykonať pri spustení služby, spustenie programu a nastavenie prepínačov, na ktoré sa môžeme odkazovať syntaxou $*ARGX*$, kde X predstavuje poradie argumentov.

Schéma príkazu je znázornená nižšie.

```
define command {
    command_name ...
    command_line ...
}
```

Aby sme rozdelili senzory na samostatné celky, pre každý senzor musíme vytvoriť vlastný konfiguračný súbor, ktorý sa skladá z definície cieľového zariadenia a služieb, ktoré budeme monitorovať. Konfiguračné súbory sú uložené v adresári */usr/local/nagios/objects/*.

Konfiguračné súbory pre jednotlivé senzory musíme pridať v konfiguračnom súbore systému Nagios, */usr/lcoal/nagios/nagios.cfg*. Môžeme definovať jednotlivé súbory alebo definovať cestu k adresáru, v ktorom budeme ukladať všetky konfiguračné súbory.

Ako sme spomenuli vyššie, konfiguračný súbor senzorov sa skladá z dvoch častí, definície cieľového zariadenia a služieb, ktoré budeme monitorovať.

Pri definícii cieľového zariadenia, názov zariadenia predstavuje názov senzoru, ktorý monitorujeme a cieľová IP adresa je v našom prípade pri každom senzore rovnaká, jedná sa o IPv6 adresu Home Assistenta.

Posledným krokom je vytvorenie definícií služieb, ktoré budeme monitorovať nami vytvoreným pluginom *check_oid* z kapitoly 4.3. Pri definícii služby sa odkazujeme na príkaz *check_oid*, prípadne jeho varianty, ktoré si spomenieme v kapitole 4.5, v ktorej sa nachádza aj príklad vytvorenej služby.

4.5 Vizualizácia

Samostatný systém Nagios core neumožňuje vizualizovať monitorované dáta. Poskytuje rozhranie, ktorým môže predať monitorované dáta externým programom, ktoré si môžu získané dáta uložiť a spracovať na analýzu.

Existuje niekoľko spôsobov, ako môžeme docieliť vizualízáciu monitorovaných dát[2], pre nášu prácu sme si zvolili nástroj *PNP4Nagios* [3], ktorý je voľne dostupný. Tento nástroj nám umožňuje relatívne jednoduchú správu a flexibilitu grafického rozhrania. Po úspešnej inštalácií *PNP4Nagios*[4] si ukážeme ako docielime, aby získané dáta mohli byť graficky znázornené.

Aby sme mohli použiť *PNP4Nagios* musíme upraviť nami definované služby. Každá služba, ktorú chceme vizualizovať musí mať definovaný parameter *action_url*, ktorým sa odkazujeme na URL konkrétnej služby na danom senzore. Nižšie si ukážeme príklad konfigurácie služby spolu s parametrom pre *PNP4Nagios*.

```
define service {
    check_interval                  1
    use                             local-service
    host_name                       SHP6_fridge
    service_description             Current
    check_command                   check_sensor_current! -o 1.3.6.1.3...
    action_url /pnp4nagios/index.php/graph?host=$HOSTNAME$&srv=$SERVICEDESC$
}
```

V uvedenom príklade môžeme vidieť, že sme nepoužili príkaz *check_oid* z kapitoly 4.3, ale *check_sensor_current*. Jedná sa o nový príkaz, ktorý sa taktiež odkazuje na plugin *check_oid*. Túto zmenu musíme vykonať vždy, keď chceme upraviť spôsob zobrazovania dát pre jednotlivé služby. Nástroj *PNP4Nagios* musí mať definovanú šablónu zobrazovania pre každý definovaný príkaz.

Šablóna musí mať rovnaký názov, ako je názov príkazu s príponom *.php*. Šablóny, ktoré môžeme používať, musia byť uložené v adresári */usr/local/pnp4nagios/share/templates.dist*.

[2] viz https://support.nagios.com/kb/category.php?id=153
[3] viz https://docs.pnp4nagios.org/
[4] viz https://github.com/techarkit/nagios/blob/master/PNP4Nagios_Installation.md

Predtým ako si ukážeme príklad šablóny, ukážeme si konfiguračný súbor pre jednotlivé šáblóny, a snimi spojenú *Round Robin Databázu*[5].

Ako náhle máme upravené konfiguračné súbory pre senzory a reštartovali sme systém Nagios, nástroj *PNP4Nagios* vygeneroval v adresári */usr/local/pnp4nagios/var/perfdata* pre každý konfiguračný súbor samostatný adresár, ktorý obsahuje konfiguračné súbory vo formáte XML a RRD databázy pre všetky služby, pri ktorých sme definovali *action_url*.

Nižšie máme zobrazenú časť konfiguračného súboru vo formáte XML pre príkaz s názvom *check_sensor_current*. V konfiguračnom súbore môžeme vidieť napríklad element, ktorým definujeme miesto ukladania monitorovaných dát, RRDFILE, ak by nám nevyhovovalo základne umiestnenie.

Môžeme definovať hraničné hodnoty, ako interval alebo skalárnu hodnotu. Zároveň v tejto časti definujeme premenné, ktoré budeme môcť použiť pri vytváraní šablóny. Jedná sa napríklad o elementy s názvom *NAME*, *UNIT*, *WARN*, *CRIT*.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NAGIOS>
  <DATASOURCE>
    <TEMPLATE>check_sensor_current</TEMPLATE>
    <RRDFILE>/usr/local/pnp4nagios/var/perfdata/SHP6_ladnicka/Current.rrd</RRDFILE>
    <RRD_STORAGE_TYPE>SINGLE</RRD_STORAGE_TYPE>
    <RRD_HEARTBEAT>8460</RRD_HEARTBEAT>
    <IS_MULTI>0</IS_MULTI>
    <DS>1</DS>
    <NAME>var</NAME>
    <LABEL>var</LABEL>
    <UNIT></UNIT>
    <ACT>0.315</ACT>
    <WARN>0.35</WARN>
    <WARN_MIN></WARN_MIN>
    <WARN_MAX></WARN_MAX>
    <WARN_RANGE_TYPE></WARN_RANGE_TYPE>
    <CRIT>0.40</CRIT>
    <CRIT_MIN></CRIT_MIN>
    <CRIT_MAX></CRIT_MAX>
    <CRIT_RANGE_TYPE></CRIT_RANGE_TYPE>
    <MIN></MIN>
    <MAX></MAX>
  </DATASOURCE>
  .
  .
  .
```

Ukážeme si príklad vytvorenia šablóny pre službu *check_sensor_current*. V tomto príklade budeme vychádzať z už existujúcej šablóny, ktorá bola určená pre príkaz *check_load*.

Teraz si vysvetlíme, čo jednotlivé riadky znamenajú a čo ich modifikáciou môžeme získať. Riadok šesť nám slúži na definovanie osi Y zobrazovaného grafu spolu s názvom služby a senzoru, ktorého sa týka. Na riadku osem definujeme spôsob zobrazovania monitorovaných dát. Môžeme napríklad v jednom grafe zobrazovať viac hodnôt, na ktoré sa môžeme

[5]viz https://en.wikipedia.org/wiki/RRDtool

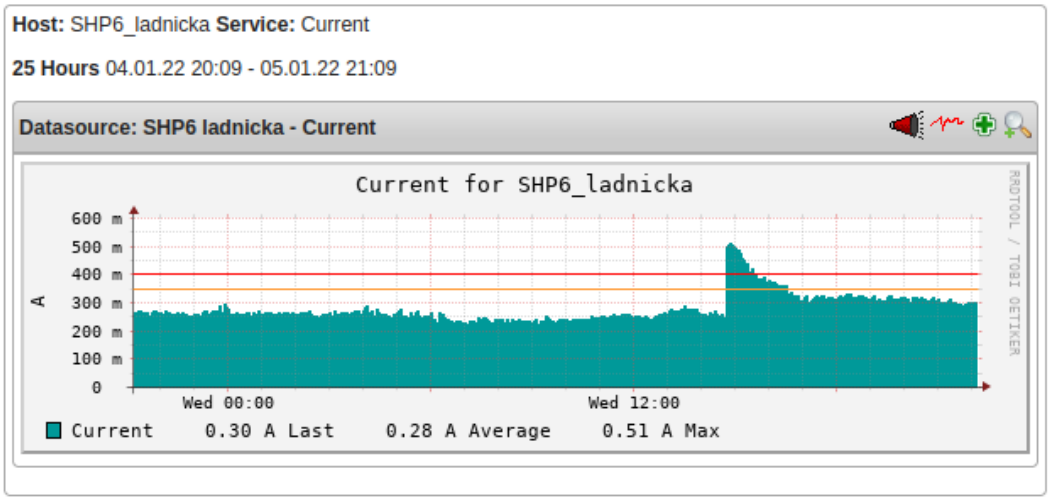
odkazovať cez premennú *DS* a definovať, ktorá hodnota bude zobrazovaná za časovú jednotku, môže sa jednať minimálnu, maximálnu alebo priemernú hodnotu. V tomto prípade sa snažíme zachytiť maximálnu hodnotu elektrického prúdu.

Na riadku šestnásť definuje štýl zobrazenia, konkrétne farbu pre konkrétny element. Na predposlednom riadku definujeme zobrazenie pre poslednú, priemernú a maximálnu získanú hodnotu.

Riadky desať až pätnásť slúžia na definíciu hraničných hodnôt. Vieme tak vytvoriť línie zobrazené v grafe, ktoré nám môžu slúžiť, ako vizuálna pomôcka pri odhadovaní hraničných hodnôt pre jednotlivé služby.

```
1  <?php
2     #
3     # Copyright (c) 2006-2010 Joerg Linge (http://www.pnp4nagios.org)
4     #
5
6     $opt[1] = "--vertical-label A -l0  --title \"Current for $hostname\" ";
7
8     $def[1]  = rrd::def("var1", $RRDFILE[1], $DS[1], "MAX");
9
10    if ($WARN[1] != "") {
11      $def[1] .= "HRULE:$WARN[1]#FFFF00 ";
12    }
13    if ($CRIT[1] != "") {
14        $def[1] .= "HRULE:$CRIT[1]#FF0000 ";
15    }
16    $def[1] .= rrd::area("var1", "#009999", "Current") ;
17    $def[1] .= rrd::gprint("var1", array("LAST", "AVERAGE", "MAX"), "%6.2lf A");
18 ?>
```

Na obrázku 4.3 je zobrazená ukážka vizualizácie dát získaných agentom SNMP.



Obr. 4.3: Príklad vizualizácia nástrojom PNP4nagios

4.6 Zhrnutie

V tejto kapitole sme si ukázali, aký spôsobom budeme zbierať dáta z IoT siete.

Aby sme mohli získať informácie z IoT siete, vytvorili sme agenta SNMP, ktorý predstavuje kontajnerovú aplikáciu, ktorú je možné použiť ako Add-on pre platformu Home Assistant.

Agenta SNMP sme rozdelili na dva funkčné bloky. V prvá časť predstavuje grafické rozhranie, v ktorom môžeme vytvoriť konfiguračný súbor.

Druhá časť predstavuje už samotného agenta SNMP, ktorý používa definované API Home Assistenta. Použitím tohto API dokáže získať potrebné informácie o senzoroch, ktoré má definované v konfiguračnom súbore.

Ďalším krokom na získanie dát z IoT siete bolo vytvorenie pluginu pre systém Nagios, ktorým sa môžeme dotazovať na monitorované zariadenia. Plugin plní ešte druhú úlohu, a to vytváranie logovacích súborov.

V poslednej časti sme spomenuli akým spôsobom môžeme docieliť vyzualizáciu nazbieraných dát. Použili sme nástroj *PNP4Nagios*. Vizualizácia dát nám umožní lepšie pochopiť nazbierané dáta a ich chovanie.

Kapitola 5

Detekcia anomálií

Detekcia anomálií je proces, pri ktorom sa snažíme nájsť dátové objekty, ktorých chovanie sa výrazne líši od ostatných objektov. Za normálne chovanie označujeme dátové objekty, ktoré sa najčastejšie objavujú a majú podobné vlastnosti. Ako anomáliu môžeme označovať dátové objekty, ktorých chovanie sa výrazne líšia voči väčšiny dátových objektov, ktoré sme označili ako objekty s normálnym chovaním [11].

Predpokladáme, že anomálie sa vyskytujú len zriedka, a teda predstavujú neočakávaný stav, ktorý by mohol predstavovať hrozbu, poškodenie zariadenia a podobne. Anomálie rozdeľujeme do troch skupín bodové, kontextové a kolektívne anomálie.

- Bodové anomálie predstavujú dátové objekty, ktoré sa výrazne líšia od ostatných, práve pri jednom pozorovaní [11].
- Pri kontextových anomáliách rozdeľujeme atribúty dátového objektu na dve skupiny, atribúty reprezentujúce kontext a atribúty reprezentujúce chovanie objektu. Kontextové atribúty môžu predstavovať napríklad čas alebo miesto. Ako príklad si môžeme uviesť meranie teploty. Kontextový atribút predstavuje miesto merania, pretože tá istá nameraná teplota môže predstavovať na jednom mieste normálne chovanie a na druhom abnormálne. Atribúty, ktoré reprezentujú chovanie objektu v tomto prípade je spomenutá teplota. Kontextovú anomáliu určuje chovanie objektu za určitých podmienok [11].
- Kolektívna anomália predstavuje zhluk objektov, ktoré ako samostatné objekty nemusia predstavovať anomálie, ale ich zoskupenie môže predstavovať neobvyklú postupnosť udalostí [11].

5.1 Metódy detekcie anomálií

Metódy detekcie anomálií rozdeľujeme do viacerých kategórií. Rozdeľujeme ich na základe trénovacích dát, a to podľa toho, či trénovacie dáta obsahujú informáciu o tom, či chovanie dátového objektu je normálne alebo abnormálne. Rozdeľujeme ich do troch základných skupín.

- Detekcia anomálií pod dohľadom (Supervised Anomaly Detection) - táto metóda požaduje správne označené trénovacích dát pre normálne, ale aj abnormálne chovanie. Nevýhodou tejto metódy je, že v niektorých prípadoch nie sme schopný definovať všetky abnormálne chovania [8].

- Detekcia anomálií s čiastočným dohľadom (Semisupervised Anomaly Detection) - metóda požaduje správne označené trénovacích dát iba pre objekty, ktoré predstavujú normálne chovanie. Definujeme tak iba normálne chovanie, čo nám umožňuje detekovať širšiu škálu anomálií ako v prípade detekcie pod dohľadom [8].

- Detekcia anomálií bez dozoru (Unsupervised Anomaly Detection) - pri metóde bez dozoru nepožadujeme informáciu o tom či sa jedná o anomáliu alebo normálne chovanie. Vychádza z predpokladu, že výskyt vzoriek s normálnym chovaním je oveľa väčší ako výskyt anomálií [8].

5.1.1 Prístup založený na klasifikácii

Klasifikácia je proces rozhodovania, ktorý vstupné dáta rozdeľuje do tried na základe spoločných vlastností. Model označujeme aj ako klasifikátor, ktorého úlohou je rozdeľovať vstupné dáta do jednotlivých tried.

Aby sme mohli začať s klasifikáciou, musíme najprv vytvoriť model, klasifikátor, ktorý vytvoríme na základe trénovacích dát, ktoré sa skladajú z dvoch častí: dátového objektu, ktorý označujeme ako objekt, a atribút, ktorý označuje triedu, kam objekt patrí. Existuje niekoľko spôsobov, ako môžeme vytvoriť klasifikátor. Môžeme použiť napríklad rozhodovací strom.

Pri detekcii anomálií je komplikované definovať všetky prípady anomálií. Na detekciu anomálií sa používa model jednej triedy. Trieda definuje normálne chovanie. Definuje rozhodovacie hranice, ktoré určujú, či daná vzorka patrí medzi normálne chovanie alebo abnormálne. Tým že nedefinujeme triedy s abnormálnym chovaním, nám tento spôsob umožňuje detekovať aj chovania, ktoré sú odlišné od všetkých predchádzajúcich prípadov a zároveň neboli obsiahnuté v trénovacej sade.

5.1.2 Prístup založený na štatistike

Úlohou štatistických metód je naučiť sa z vstupných dát generatívny model, na základe ktorého bude model rozdeľovať vzorky s vysokou pravdepodobnosť ako normálne a vzorky s nízkou pravdepodobnosťou ako anomálie. Musíme ale dodržať predpoklad, že vstupné dáta pochádzajú z normálneho rozdelenia [11].

Štatistické metódy rozdeľujeme do dvoch skupín, a to parametrické a neparametrické metódy. Parametrické metódy sa snažia naučiť parametre normálneho rozloženie z vstupných dát. Neparametrické metódy sa snažia naučiť model zo vstupných dát, ale nie sú obmedzené na apriórne štatistické modely. Nedefinujú fixný počet parametrov [11].

Pri výbere vhodnej metódy, musíme brať do úvahy počet atribútov. Ak berieme do úvahy dáta zložené z viacerých atribútov, existujú rozširujúce varianty, ktoré transformujú viac atribútovú detekciu anomálií na detekciu anomálií s jedným atribútom, môžeme použiť napríklad χ^2 - kvadrát[1] alebo Mahalonobisovú vzdialenosť[2].

5.1.3 Prístup založený na blízkosti

Prístupy založené na blízkosti rozdeľujeme na dve skupiny. Prvá skupina predstavuje metódy založené na vzdialenosti medzi objektami. Vychádzame z predpokladu, že okolo kaž-

[1] viz https://biopedia.sk/genetika/chi-kvadrat-test
[2] viz https://sk.hrvwiki.net/wiki/Mahalanobis_distance

dého objektu sa môžu nachádzať objekty v určitej vzdialenosti. Objekt, ktorý vo svojom okolí s priemerom r nemá dostatočné množstvo susedov označujeme ako anomáliu [11].

Druhá skupina predstavuje metódy založené na hustote. Na dáta sa pozeráme z globálneho hľadiska a skúmame hustotu výskytov objektov s podobnými vlastnosťami. Pri vyhodnotení každého objektu sa snažíme vyjadriť faktor, ktorým vyjadrujeme, ako veľmi je daná vzorka odlišná od n najbližších susedov.

5.1.4 Prístup založený na zhlukovaní

Pri tomto prístupe sa snažíme vyjadriť vzťah medzi dátovým objektom a najbližším zhlukom. Detekciu anomálií môžeme vykonávať na základe troch znakov [11].

- Dátový objekt nepatrí do žiadneho zhluku.
- Vzdialenosť medzi dátovým objektom a zhlukom je príliš veľká.
- Dátový objekt patrí do zhluku objektov, ktoré sú vyhodnotené ako anomálie.

Na základe spomenutých znakov, nám tento prístup umožňuje vykonať detekciu anomálií bez dozoru. Nepotrebujeme pre trénovacie dáta označenie normálneho a abnormálneho chovania. Metódy porovnávajú vlastnosti objektu voči vytvoreným zhlukom a na základe toho identifikujú, či sa jedná o odľahlú hodnotu.

5.2 Modely založené na štatistike

Na detekciu anomálií môžeme použiť prístup založený na štatistike z kapitoly 5.1.2 a na vyhodnotenie môžeme použiť rôzne rozdelenia. V našej práci si spomenieme dve rozdelenia: normálne a logaritmicko-normálne rozdelenie.

Vychádzame z predpokladu, že 99.7% dát sa nachádza v rozložení od strednej hodnoty. Pre jednotlivé rozdelenia budeme musieť zistiť strednú hodnotu a hraničné hodnoty, pre ktoré môže model rozhodnúť, či pochádzajú z daného rozdelenia.

5.2.1 Test rozdelenia

Pred tým ako použijeme nejaký model, musíme overiť predpoklad, či vzorky pochádzajú z daného rozdelenia. Aby sme tento predpoklad overili, môžeme použiť test dobrej zhody. Existuje množstvo testov dobrej zhody, môžeme si uviesť napríklad [16]:

- Pearsonov test
- Kolmogorovov test
- Kolmogorovov-Smirnovov test
- Shapiro-Wilkov test

5.2.2 Model normálneho rozdelenia

Model normálneho rozdelenia nazývame aj ako Gausovský model. Cieľom bude naučiť model parametre normálneho rozloženia, strednú hodnotu μ a štandardnú odchýlku σ zo vstupných dát. Aby sme odhadli parametre normálneho rozloženia, môžeme použiť metódu *maximum-likelihood* [11].

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$
$$\hat{\delta}^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathrm{x}_i - \bar{x})$$

Model normálneho rozdelenia používa na určenie hraničných hodnôt od strednej hodnoty μ smerodajnú odchýlku σ. Pri pozorovaní hodnôt sa vyskytne pri prvej odchýlke 68% dát, pri druhej odchýlke, 95% a pri tretej až 99.7%.

Pri vyhodnotení novej vzorky zistíme rozdiel voči odhadu strednej hodnoty μ. Ak rozdiel voči strednej hodnote μ nadobúda väčšie hodnoty ako je 3σ, túto vzorku označíme ako anomáliu. Tento spôsob vyhodnotenia označujeme ako pravidlo tri sigma.

5.2.3 Model logaritmicko normálneho rozdelenia

Podobne ako to bolo v prípade normálneho rozdelenia cieľom bude naučiť model parametre logaritmicko normálneho rozdelenia strednej hodnoty μ a štandardnej odchýlky σ. Na odhad parametrov môžeme použiť metódu maximum-likelihood [10]. Vzorce podľa ktorých môžeme odhadnúť parametre rozdelenia môžeme vidieť nižšie.

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n}\ln(\mathrm{x}_i)$$

$$\hat{\delta}^2 = \frac{1}{n}\sum_{i=1}^{n}(\ln(\mathrm{x}_i) - \frac{\mu}{n})$$

Pri vyhodnotení novej vzorky sa budeme pozerať, ako ďaleko sa nachádza od nami odhadnutej strednej hodnoty, podobne, ako to bolo v prípade normálneho rozdelenia.

5.3 Regresný model

V tejto časti si predstavíme regresné modely. Medzi najznámejšie regresné modely patria napríklad modely založené na lineárnej alebo polynomiálnej regresii. Úlohou regresných modelov je vyjadriť vzťah medzi atribútmi, kde vzniká závislosť, napríklad časová. Závislý atribút sa zvykne označovať ako Y alebo anglicky dependent. Atribút, ktorý je nezávislý a na ktorom vzniká závislosť označujeme ako X [12].

5.3.1 Lineárna a polynomiálna regresia

Ako prvú si ukážeme lineárnu regresiu. Pri použití tejto metódy predpokladáme, že medzi závislou a nezávislou premenou je lineárny vzťah. Tento vzťah môžeme vyjadriť priamkou [18]:

$$y_t = \beta_0 + \beta_1 x_t + \epsilon_t \text{ , kde}$$

- koeficient β_0 vyjadruje posun regresnej priamky od nulového bodu
- koeficient β_1 vyjadruje sklon regresnej priamky
- koeficient ϵ_t vyjadruje náhodnú chybu, ktorá vyjadruje odchýlku od lineárneho modelu

- x_t predstavuje vstupný parameter regresnej priamky
- y_t predstavuje výstupnú predikovanú hodnotu regresného modelu

Pri vytváraní odhadu polynomiálnou regresiou, potrebujeme niekoľko vstupov, ktoré definujú jednotlivé prediktory. Vzťah medzi odhadovanou hodnotou a polynomiálnou regresiou vyjadrujeme vzťahom zobrazeným nižšie [12]:

$$y_t = \beta_0 + \beta_1 x_{t,1} + \beta_2 x_{t,2} + \beta_k x_{t,k} + \epsilon_t$$

Úlohou regresie je odhadnúť koeficienty $\beta_0, \beta_1, ..\beta_k$ z kolekcie predchádzajúcich dát, aby účelová funkcia bola minimálna, a teda rozdiel medzi skutočnou hodnotou a predikovanou hodnotou minimálny. Najčastejšia používaná účelová funkcia je MSE (Mean Squared Error), ale existujú aj iné, ako je napríklad MAE (Mean Absolute Error). Vzorec na výpočet účelových funkcií môžeme vidieť nižšie [18].

$$MSE = \frac{1}{n} \sum_{ni=1}^{n} (y_i - \bar{y})^2$$

$$MAE = \frac{1}{n} \sum_{ni=1}^{n} |y_i - \bar{y}|$$

Polynomiálna a lineárna regriesa majú rovnaké využitie. Polynomiálnú regresiu môžeme použiť napríklad v prípade, keď vstupné dáta sú príliš rozptýlené a priamka by bola v tomto prípade veľmi nepresná. Pri polynomiálnej regresii si ale musíme dať pozor na stupeň polynomu. Ak by sme použili vysoký stupeň polynomu, môže byť odhad hodnôt príliš presný a nebudeme schopný predikovať iné hodnoty.

5.4 Lokálny faktor odľahlosti (Local Outlier Factor)

Na detekciu anomálií budeme pracovať s lokálnym faktor odľahlosti, anglicky *Local outlier factor*, LOC. Tento faktor pre každú dátovú vzorku vyjadruje mieru odľahlosti. Faktor vychádza iba z obmedzeného okolia danej vzorky, preto sa nazýva lokálny. Tento prístup súvisí so zhlukovaním na základe hustoty [5].

Aby sme mohli vyjadriť LOF, musíme najprv uviesť niekoľko definícií.

- K-vzdialenosť - pre akékoľvek kladné číslo, k-vzdialenosť objektu p označovaná ako k-vzdialenosť(p), je definovaná ako vzdialenosť medzi bodmi p a o, $d(p, o)$ kde $p, o \in D$ a pre ktoré platí súčasne:
  - Pre minimálne k objektov $o' \in D \backslash \{p\}$ platí, že $d(p, o') \leq (p, o')$.
  - Pre maximálne k-1 objektov $o' \in D \backslash \{p\}$ platí, že $d(p, o') < (p, o')$.
- K-najbližších susedov - je množina objektov vzhľadom ku k-vzdialenosti objektu p, ktorá je definovaná ako $N_{k-distance}(p) = \{q \in D \backslash \{p\} \mid d(p, q) \leq k\text{-}vzdialenos(p)\}$
- Hustota dosiahnuteľnosti - je definovaná ako maximálna vzdialenosť objektov *k-distance(o)* a vzdialenosťami $d(p, o)$. $reach\text{-}dist_k(p, o) = max\ \{k\text{-}distance(o), d(p, o)\}$

- Hustota lokálnej dosiahnuteľnosti objektu p predstavuje prevrátenú hodnotu priemernej vzdialenosti dosiahnuteľností, na základe paramteru MinPts, ktorý predstavuje minimálny počet susedov. Označujeme ju ako LRD.

$$LRD_{MinPts}(p) = 1/\frac{\sum_{o \in N_{MinPts}(p)} reach\text{-}dist_{MinPts}(p,o)}{|N_{MinPts}(p)|}$$

- Odľahlý faktor objektu p zachytáva mieru, ako je odľahlý daný objekt od svojho okolia. Odľahlý faktor objektu je definovaný ako pomer hustoty lokálnej dosiahnuteľnosti objektu p a hustoty p s MinPts najbližšíimi susedmi.

$$LOF_{MinPts}(p) = \frac{\sum_{o \in N_{MinPts}(p)} \frac{LRD_{MinPts}(o)}{LRD_{MinPts}(p)}}{|N_{MinPts}(p)|}$$

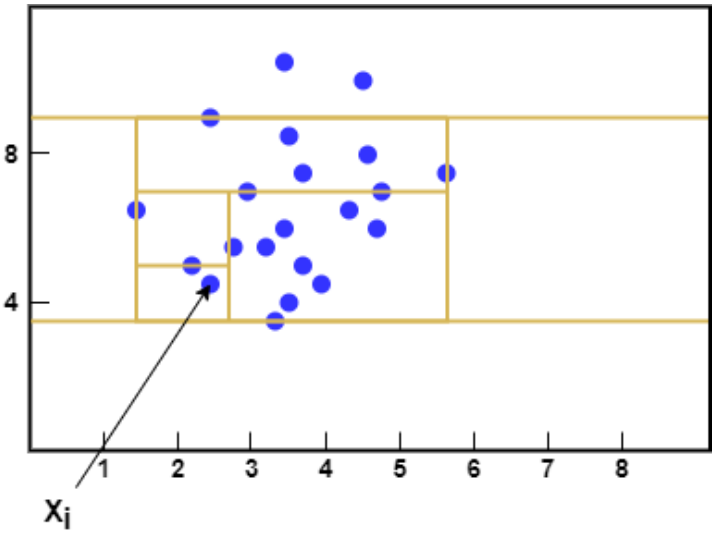
5.5 Izolačný les (Isolation forest)

V tejto časti sa budeme venovať metóde detekcii anomálií, ktorá sa značne líši od prechádzajúcich metód. Táto metóda vychádza z predpokladu, že počet anomálií je výrazne nižší a ich vlastnosti sa výrazne líšia od ostatných [13].

Zameriava sa na izoláciu anomálií namiesto definovania normálneho chovania a počítania vzdialeností medzi vzorkami. Na vyhodnotenie používa súbor stromov izolácie, ktoré tvoria izolačný les [13].

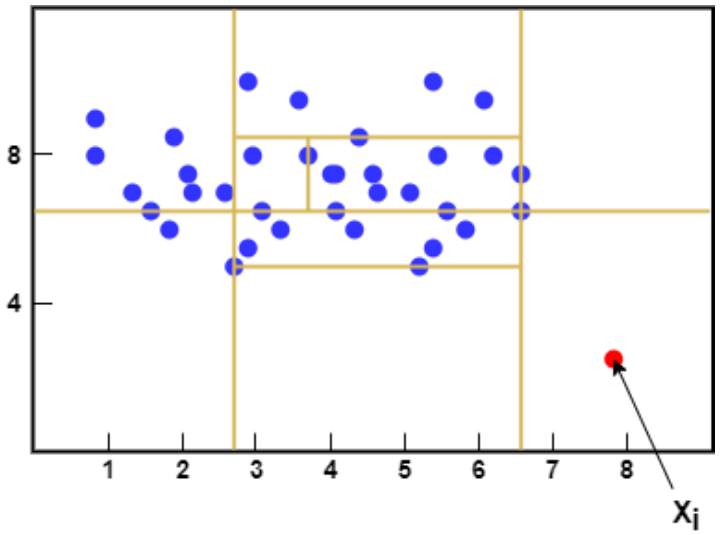
Základným princípom tejto metódy je izolovať všetky dátové vzorky. Prebieha rekurzívne delenie priestoru podľa náhodne zvolenej vzorky a jéj náhodne zvolenej vlastnosti, podľa hodnoty, ktorá môže nadobúdať hodnoty v rozmedzí jej minimálnej a maximálnej hodnoty. Toto rozdelenie je znázornené na obrázku 5.1 a 5.2.

Na obrázku 5.1 je znázornená izolácia vzoriek bez anomálií. Rekurzívne sa priestor rozdeľuje a na izoláciu jednotlivých vzoriek je nutné vytvoriť množstvo delení. Na obrázku 5.1 je znázornená situácia, pri ktorej na izoláciu vzorky X_i je nutné vykonať sedem delení.



Obr. 5.1: Rozdelenie dátových vzoriek bez anomálií

V druhom prípade už uvažujeme situáciu s anomáliou. Na obrázku 5.2 anomáliu predstavuje vzorka označená ako X_i a je vidieť, že na jej izoláciu nám postačuje výrazne menší počet rozdelení.



Obr. 5.2: Rozdelenie dátových vzoriek s anomáliou

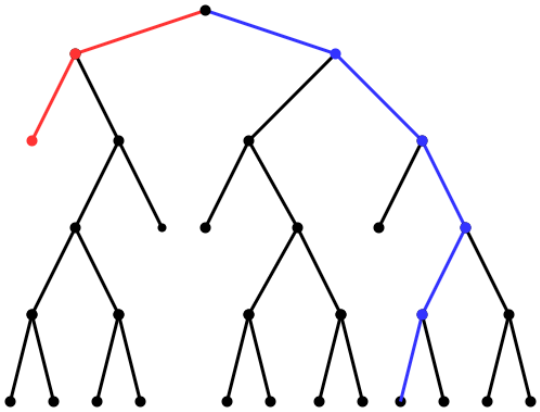
Izolačný strom predstavuje úplný binárny strom, kde každý uzol má práve žiadneho alebo dvoch potomkov. Vetvenie izolačného stromu predstavuje delenie priestoru, ako sme si ukázali na obrázkoch 5.1 a 5.2. Počet rozdelení potrebných na izoláciu nám udáva dĺžku cesty z koreňového uzla izolačného stromu k dátovej vzorke.

Izolačný strom je znázornený na obrázku 5.3, v ktorom sú znázornené prípady pre obvyklú a neobvyklú vzorku.

- Neobvyklá vzorka sa výrazne líši od ostatných, čo má za následok, že budeme potrebovať nižší počet delení priestoru k izolovaniu. Počet delení udáva dĺžku cesty z koreňového uzla, z toho vyplýva, že anomálie sa budú nachádzať v blízkosti koreňového uzla izolačného stromu. Tento prípad je zvýraznený červenou farbou.

- Obvyklá vzorka je znázornená na obrázku modrou farbou. Aby bolo možné izolovať danú vzorku bolo potrebných aspoň päť delení. Vieme že podozrivé dátové vzorky sa nachádzajú v blízkosti koreňového uzla, a preto stačí prechádzať izolačný strom len do určitej hĺbky, aby sme označili dátovú vzorku ako obvyklú.

Detekcia anomálií touto metódou pozostáva z dvoch fáz. Začíname fázou trénovania, v ktorej definujeme počet izolačných stromov l a veľkosť pod vzorkovania ψ, od ktorého sa odvíja približná výška izolačného stromu definovaným vzťahom $h = \log_2(\psi)$. Počas tejto fáze prebieha konštrukcia izolačných stromov rekurzívnym delením dátových vzoriek, pokiaľ nie sú všetky dátové vzorky izolované alebo výška stromu nedosiahne špecifickú výšku [13].



Obr. 5.3: Izolačný strom

Vo fáze vyhodnotenie prebieha niekoľko výpočtov. Začneme priemernou dĺžkou cesty v izolačnom lese pre vstupnú dátovú vzorku, ktorú označujeme ako $E(h(x))$, kde $h(x)$ predstavuje dĺžku cesty v jednom izolačnom strome. Pre výpočet finálneho skóre vychádzame z rovnice $s(x, \psi)$ uvedenej nižšie [13].

$$s(x, \psi) = 2^{-\frac{E(h(x))}{c(n)}}$$

$$c(n) = H(n-1) - ((2 * n/2) - 1)$$

$$H(i) = \lg i + \gamma$$

Výsledná hodnota $s(x, \psi)$ sa nachádza v intervale $< 0, 1 >$, kde hodnoty vyššie, ako zvolená prahová hodnota, môžeme označovať za pravdepodobné anomálie.

5.6 Zhrnutie

V tejto kapitole sme sa venovali základným pojmom spojených s detekciou anomálií. Predstavali sme si jednotlivé typy anomálií a metódy detekcie, ktoré sme rozdelili na základe trénovacích dát.

Venovali sme sa rôznym prístupom ako je napríklad prístup založený na zhlukovaní alebo prístup založený na štatistike. Pre vybrané prístupy sme ukázali rôzne metódy, ktorými môžeme definovať správanie IoT zariadení.

Prvý model, ktorému sme sa venovali bol Gausovský model 5.2.2, ktorého prístup je založený na štatistike. Pri tomto modele musíme najprv overiť predpoklad, že sa jedná o normálne rozdelenie. Na overenie môžeme použiť napríklad Pearsonov test dobrej zhody.

Prístupy založené na blízkosti rozdeľujeme do dvoch kategórií. Na prístup založený na vzdialenosti a prístup založený na hustote. Použili sme metódu nazývanú lokálny faktor odľahlosti 5.4, ktorá používa prístup založený na hustote.

Poslednej časti sme sa venovali metóde izolačný les. Táto metóda sa výrazne líši od ostatných. Vytvára les izolačných stromov, ktorých skóre je použité ako klasifikátor.

Získané poznatky použijeme na popis chovania IoT zariadení, ktoré boli spomenuté v kapitole 3.3.

Kapitola 6

Nástroj na detekciu anomálií v systéme Nagios

V tejto časti sa budeme venovať implementačnej časti nástroja, ktorého činnosťou je spracovať a vyhodnotiť prichádzajúce dátové vzorky monitorované agentom SNMP z kapitoly 4. Tento nástroj sme pomenovali ako *Learning Core*, učebné jadro.

Na implementáciu bol použitý jazyk Python 3.8[1], ktorý nám umožňuje pracovať s knižnicou scikit-learn[2], pomocou ktorej boli implementované modely z kapitoly 5.

6.1 Štruktúra programu

Základnou štruktúrou programu je päť tried, ktoré riadia beh programu a spracuvávajú dáta. Zvyšné triedy predstavujú použité modely určené na klasifikáciu a ich rozhranie, ktoré musí každá trieda implementovať, aby ich bolo možné aplikovať.

Diagram tried je znázornený na obrázku 6.1, a postupne si prejdeme činnosť jednotlivých tried.

Začneme triedou *Sensor*, ktorá obsahuje všetky podstatné informácie o senzore ako takom a informácie, ktoré sa týkajú trénovania jednotlivých modelov. Medzi základné informácie o senzore patrí napríklad jeho jedinečný identifikátor, *object_id*. Medzi informácie, ktoré sú potrebné pre trénovanie sú napríklad použitý model pre klasifikáciu a atribúty použité na trénovanie. Umožňuje nám napríklad definovať časové rozmedzie trénovacích dátových vzoriek.

Ako trénovacie dáta môžeme použiť logovacie súbory, ktoré sú generované Nagios pluginom. Spracovanie týchto súborov ma za úlohu trieda s názvom LogManager. Úlohou triedy LogManger je vytvoriť z príslušných logovacích súborov jeden súbor, ktorý bude použitý pri trénovaní. Tento súbor je vo formáte CSV a obsahuje časovú značku pre jednotlivé pozorovania, z ktorej sa vypočítajú parametre ako sú deň v týždni alebo čas v sekundách. Zároveň prebieha kontrola logovacích súborov či neobsahujú nepovolené hodnoty.

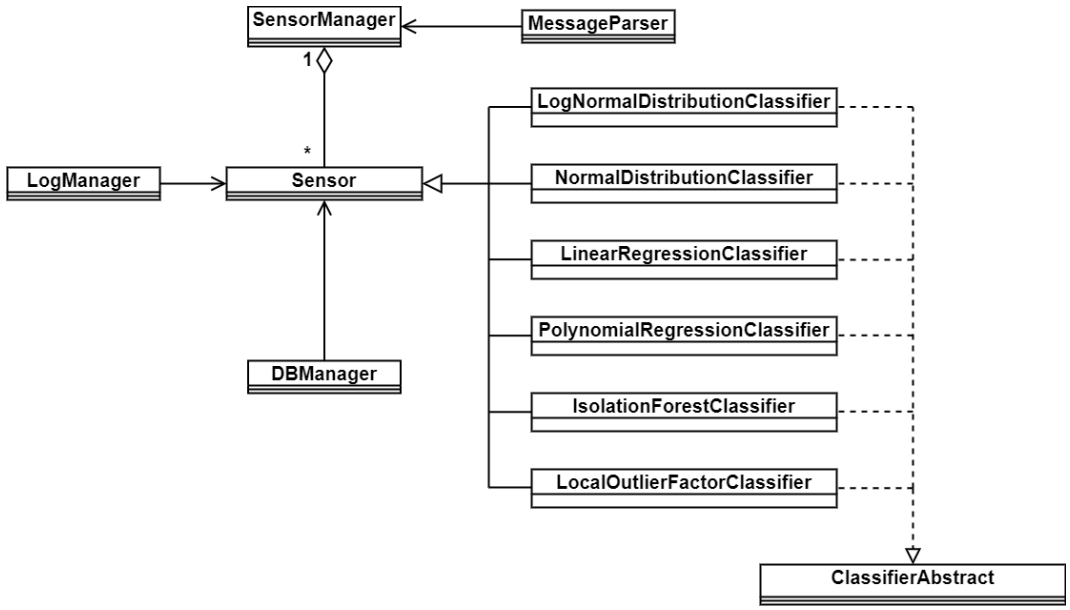
Okrem spracovania súborov prebieha spracovanie dát aj na úrovni databáze. Túto činnosť vykonáva trieda DBManager, ktorá umožňuje evidovať IoT objekty ich vlastnosti a dáta s nimi spojené. Spôsob ukladania dát si ukážeme v kapitole 6.3.

Trieda MessageParser sa používa na extrahovanie dát z UDP správ. Získané dáta predstavujú jednoznačný identifikátor senzoru a jeho hodnotu. Tieto informácie sú predávané

[1] viz https://www.python.org/downloads/release/python-380/
[2] viz https://scikit-learn.org/stable/

triede SensorManager, ktorej úlohou je združovať všetky monitorované objekty a na základe identifikátora vybrať model senzoru, ktorý vyhodnotí prichádzajúcu dátovú vzorku.

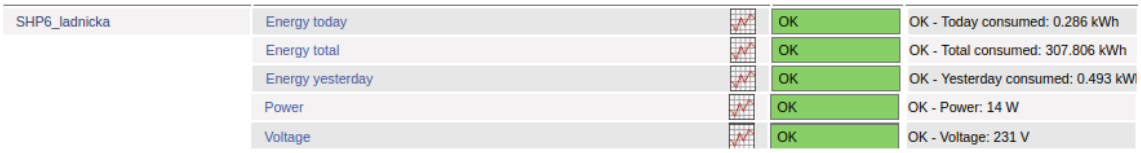
Posledné triedy, ktoré si spomenieme, sú triedy určené na klasifikáciu. Tieto triedy musia implementovať funkcie z abstraktnej triedy ClassifierAbstract, ktorá definuje metódy určené na trénovanie, predikciu aktuálneho stavu a spracovanie vlastností modelov.



Obr. 6.1: Diagram tried - Learning Core

6.2 Komunikácia

Pri vytváraní nástroja Learning Core bolo cieľom, aby detekcia anomálií bola súčasťou systému Nagios. Tento cieľ sa skladal z dvoch častí. Prvá časť predstavuje reportovanie anomálií v pôvodnom prostredí ako sú služby systému Nagios, ktoré sú znázornené na obrázku 6.2.



Obr. 6.2: Nagios - monitorované služby

Aby sme túto časť splnili, prebieha komunikácia medzi pluginom, ktorý sa dotazuje na jednotlivé položky a nástrojom Learning Core. Aby nevznikla závislosť, ktorá by určovalo súčinnosť nástrojov v rovnakom prostredí, prebieha komunikácia medzi pluginom a nástrojom Learning Core nad protokolom IP.

Pri vyhodnocovaní dátovej vzorky prebiehajú dve správy medzi nástrojmi. Iniciátor komunikácie je vždy plugin systému Nagios. Prenášané dáta sú vo formáte JSON a obsahujú informácie, ktoré jednoznačne identifikujú senzor, názov senzoru, časovú značku, deň v týždni, čas v sekundách a monitorovanú hodnotu. Príklad zaslanej správy z Nagios pluginu je znázornená nižšie.

```
{
    "object_id": "1.3.6.1.3.999.1.12.3.1",
    "friendly_name": "ESP32 Cam01 temperature",
    "timestamp": 1647255916,
    "day": 0,
    "seconds": 39916,
    "value": 42
}
```


Každá z položiek má svoj špecifický význam. Položka *object_id* predstavuje unikátnu hodnotu v celom prostredí, ktorá identifikuje senzor a službu v systéme Nagios.

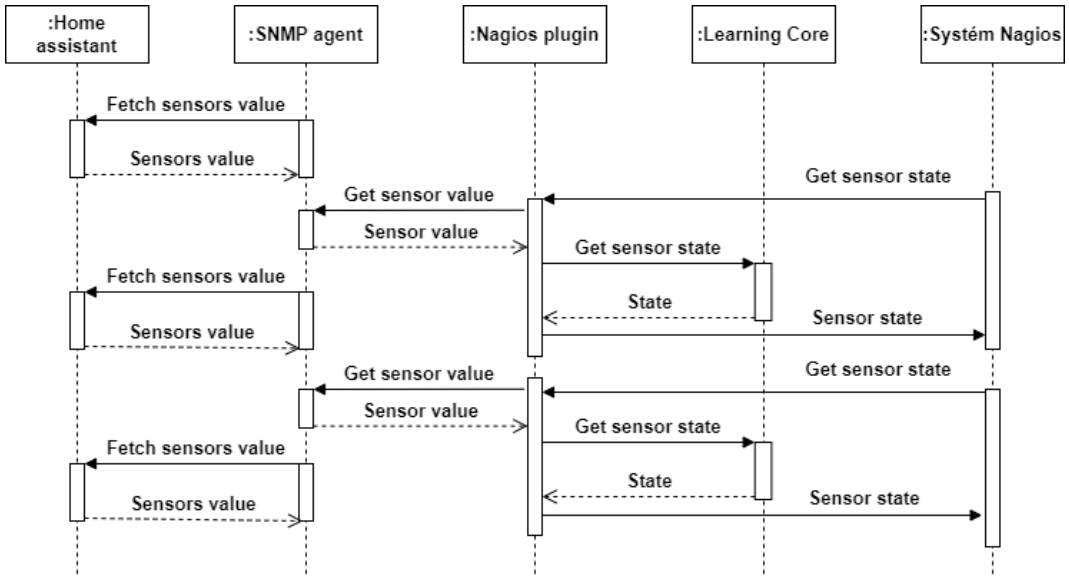
Nástroj Learning Core umožňuje detekovať služby, ktoré neboli zahrnuté v počiatočnej konfigurácií. Aby sme docielili jednoduchšie dohľadávanie takýchto služieb, správa obsahuje položku *friendly_name*, ktorá predstavuje názov služby definovanej v systéme Nagios.

Položky časovú značka, deň v týždni a čas v sekundách sú pridané len na účely testovania nástroja, ktoré si bližšie spomenie v kapitole 8.

Poslednou položkou je *value*, ktorá nesie informácie o aktuálnom stave monitorovaného senzoru. Všetky monitorované hodnoty predstavujú číselné hodnoty.

Nástroj Learning Core na takúto správu odpovedá celo číselnou hodnotou v rozsahu $0 - 3$. Hodnota predstavuje návratový kód, ktorým sa ukončí Nagios plugin. Podľa návratovej hodnoty systém Nagios následne vyhodnotí stav služby ako jeden zo štyroch možných stavov: OK, WARNING, CRITICAL alebo UNKNOWN.

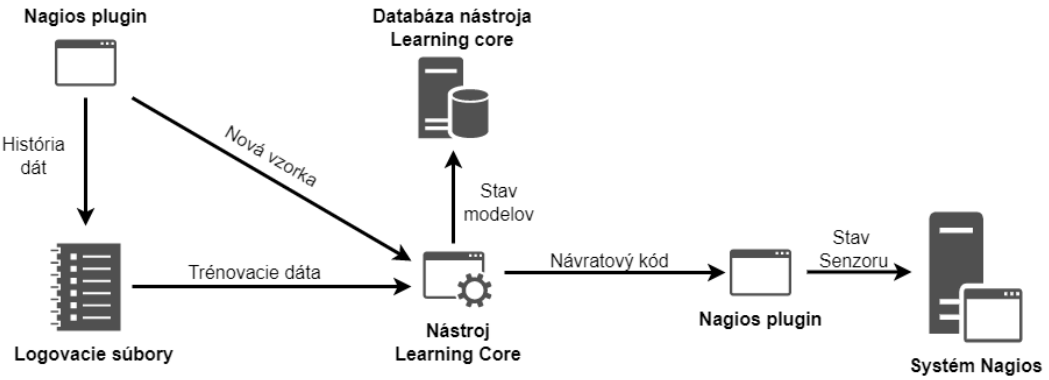
Sekvenčným diagramom na obrázku 6.3 zobrazujeme komunikáciu medzi jednotlivými súčasťami. Home assistant združuje informácie o zapojených IoT zariadeniach, na ktoré sa periodicky dotazuje agent SNMP. Podobne ako agent SNMP, aj Nagios plugin sa v intervaloch dotazuje na konkrétnu hodnotu, ktorú poskytne nástroju Learning Core. Následne je dátova vzorka vyhodnotená a výsledny stav je predaný Nagios pluginu, ktorý svojím návratovým kódom informuje o aktuálnom stave.



Obr. 6.3: Sekvenčný diagram monitorovacieho systému

Schéma monitorovacieho systému, ktorá zahŕňa aj systém Nagios je zobrazená na obrázku 6.4. Nágios plugin vykonáva tri činnosti: vytvára históriu monitorovaných dát, predáva dátové vzorky na vyhodnotenie nástroju Learning Core a reportuje stav senzoru systému Nagios.

Nástroj Learning Core na základe historických dát z logovacích súborov vytvorí modely chovania pre jednotlivé senzory. Pre nové vzorky si ukladá ich stav do databáze, ktorý predstavuje návratový kód pre Nagios plugin. Ukoženie Nagios pluginu reportuje stav služby do systému Nagios.



Obr. 6.4: Schéma monitorovacieho systému

6.3 Ukladanie dát

Počas behu programu uchovávame dva typy informácií: informácie o senzoroch a prenášané dáta. Na ukladanie týchto informácií bola použitá voľne dostupná databáza *postgreSQL*[3].

Začneme s informáciami o senzoroch. Pre tento typ informácií je vytvorená tabuľka s názvom *iot_objects*, ktorej popis môžeme vidieť nižšie:

```
CREATE TABLE iot_objects(
    OBJECT_ID TEXT NOT NULL,
    FRIENDLY_NAME TEXT NOT NULL,
    METHOD TEXT,
    UNIT TEXT,
    SENSOR_TYPE TEXT,
    LEARNING_VALUES TEXT[],
    DATE_FROM DATE,
    PARAMS json,
    PRIMARY KEY(OBJECT_ID)
);
```

Tabuľka obsahuje dva rôzne typy informácií: informácie týkajúce sa vyhodnotenia stavu senzoru a informácie, ktoré sú použité pri vizualizácií.

Medzi informácie, ktoré sa týkajú vyhodnotenia stavu sú položky LEARNING_VALUES, DATE_FROM a METHOD. Položka LEARNING_VALUES predstavuje pole atribútov, z ktorých bude vychádza trénovanie vybraného modelu. Môžeme definovať dátum, ktorým

[3]viz https://www.postgresql.org/

určujeme začiatok dátového setu použitého pri trénovaní, a tak obmedziť množstvo dat. Poslednou položkou určujeme, aký model sa ma použiť pri trénovaní pre daný senzor. Ako náhle máme všetky tieto informácie, môže prebehnúť trénovanie. Definovanie trénovacích informácií prebieha v zdrojom súbore a bližšie informácie k tejto tému sú zhrnuté v kapitole 6.4.

Zvyšné položky sú použité pri spracovaní grafického rozhrania, pri ktorom zobrazujeme jednotlivé služby. Pri ich zobrazovaní berieme do úvahy typ senzoru, na čo nám slúži položka SENSOR_TYPE, podľa ktorej rozlišujeme, či sa jedná napríklad o senzor, ktorý prenáša binárny stav ako je magnetický či pohybový senzor alebo len číselnú hodnotu, ktorá vyjadruje meranú veličinu. Poslednou položkou je PARAMS, ktorá je dátového typu JSON. Táto položka obsahuje informácie o trénovaní a výsledných parametrov jednotlivých modelov.

V časti 6.2 sme si ukázali, ako prebieha komunikácia medzi pluginom a nástrojom Learning Core. Všetky tieto správy sú ukladané do tabuľky s názvom *iot_data*. Spolu s prenášanými hodnotami sa ukladá aj informácia o stave, ktorý nástroj vyhodnotí na základne dostupných informácií. Schému tabuľky môžeme vidieť nižšie.

```
CREATE TABLE iot_data(
    INDEX BIGSERIAL,
    OBJECT_ID TEXT NOT NULL,
    TIMESTAMP INT NOT NULL,
    VALUE REAL NOT NULL,
    STATE INT NOT NULL,
    PRIMARY KEY(INDEX)
);
```

6.4 Inicializácia objektu Senzor

Inicializácia senzorov, ktoré budú spracované našim nástrojom prebieha v súbore *learning_core.py* vo funkcii *make_registration()*. V tejto časti pre jednotlivé senzory vytvárame inštancie triedy Sensor. Na príklade nižšie je zobrazený príklad definície senzoru, ktorý je označený ako *Door C305*. Inicializácia sa skladá z identifikátora senzoru, názvu služby, metódy vyhodnotenia, referencie na inštanciu triedy, ktorá spravuje prístup k databáze a typ senzoru.

Môžeme dodatočne definovať parametre k trénovaciemu modelu, v tomto prípade definujeme úroveň kontaminácie dát a množstvo vzoriek, ktoré má model spracovať. V poslednom riadku určujeme parametre trénovania, a to atribúty, ktoré majú byť brané do úvahy, a ich formát.

```
door_c305 = Sensor('1.3.6.1.3.999.1.15.3.1', 'Door C305',
                   IsolationForestClassifier, db_manager, 'magnetic')

door_c305.set_param({"contamination": 0.005, "max_samples": 5000})

door_c305.train(['Day in week', 'Seconds', 'Value'],
                {'Day in week': int, 'Seconds': int, 'Value': int})
```

6.5 Zhrnutie

V tejto kapitole sme sa venovali popisu nástroja na detekciu anomálií v IoT sieťach. Jedným z cieľov bola integrácia tohto nástroja do systému Nagios, čo sa nám podarilo vytvorením komunikácie medzi Nagios pluginom a nástrojom Learning Core.

Nagios plugin plní dve činnosti: vytváranie historických dát, ktoré sú určené na trénovanie modelov nástrojom Learning Core a predávanie nových dátových vzoriek priamo nástroju Learning Core, ktorého činnosťou je vyhodnotiť prichádzajúce dátové vzorky a ukladanie aktuálneho stavu senzoru do databáze.

Databáza pozostáva z dvoch tabuliek: z tabuľky, ktorá slúži na evidenciu IoT zariadení, ktorých veličiny sú vyhodnotené nástrojom Learning Core a tabuľka, ktorej úlohou je evidovať stav týchto IoT zariadení.

V poslednej časti sme si ukázali, ako môžeme definovať objekt Senzor, ktorý slúži na popis chovania monitorovaného IoT zariadenia.

Kapitola 7

Vyhodnotenie implementovaných modelov

V tejto časti sa budeme venovať spôsobu aplikácie jednotlivých modelov na konkrétne senzory a ich veličiny. Použitá dátová sada bola rozdelená na dve časti, trénovacia sada a testovacia sada v pomere dva ku trom.

Jedným z dôležitých faktorov pri vyhodnotení modelov je validácia. Pri trénovaní sa môžeme stretnúť s javmi ako sú:

- Pretrénovanie (*Overfitting*) - je jav, kedy parametre modelu sú príliš presné a model nedokáže detekovať odchýlky.
- Podtrénovanie (*Underfitting*) - môže nastať napríklad v prípade kedy nemáme dostatočné množstvo trénovacích dát, a nedokázali sme naučiť model optimálne parametre.

Preto dátová sada určená na trénovanie je ešte rozdelená na validačnú časť.

Ak máme k dispozícií menší počet vzoriek, môžeme použiť napríklad metódu, ktorá sa nazýva *Leave-one-out* [1]. Priebeh validácie je zdĺhavejší a závisí od počtu vzoriek. Ako príklad uvedieme dátovú sadu, ktorá obsahuje sto vzoriek. Validácia prebieha v iteráciách a v tomto prípade prebehne 100-krát na 99 vzorkách. Výsledná presnosť modelu odpovedá priemernej hodnote presnosti z jednotlivých iterácií.

Keď máme k dispozícií väčší počet vzoriek, môžeme použiť metódu krížovej validácie [1], *Cross-validation*. Táto metóda nám umožní znížiť časovú náročnosť validácie. Trénovacia dátová sada je rozdelená na K blokov, kde $K-1$ blokov je vyhradených na trénovanie a jeden blok na validáciu. Výsledná presnosť je taktiež priemerná hodnota presností z jednotlivých iterácií, ako to bolo v prípade *Leave-one-out*.

7.1 Model lineárnej a polynomiálnej regresie

Modely lineárnej a polynomiálnej regresie boli aplikovaný na niekoľko senzorov, ktoré si postupne prejdeme. Model lineárnej regresie bol použitý na detekciu anomálií senzorov, ktoré monitorujú stav celkovo spotrebovanej elektrickej energie elektrickými zásuvkami a model polynomiálnej regresie bol použitý na popis chovania spotrebovanej energie za aktuálny deň.

7.1.1 R-kvadrát

V kapitole 5.3 sme si ukázali, ako dokážeme odhadnúť parametre pre regresný model. Ako náhle máme vytvorený model, skúmame, či nám daný model vyhovuje. Pri regresných modeloch môžeme použiť na vyjadrenie kvality modelu R-kvadrát [9], označujeme ho R^2, ktorý je známy aj ako koeficient determinácie. R-kvadrát vyjadruje mieru, ako sa približujú odhadované hodnoty z regresného modelu skutočným hodnotám. Môže nadobúdať hodnoty z intervalu $< 0, 1 >$, kde nula vyjadruje úplnú nezávislosť medzi premennými, a jednotka práve úplnú závislosť.

Vzorec na výpočet R kvadrátu je znázornený nižšie, kde čitateľ vyjadruje reziduálnu sumu štvorcov a teda sumu štvorcov medzi skutočnými hodnotami Y_i a predikovanými hodnotami X_i z regresného modelu. Menovateľ vyjadruje celkovú sumu štvorcov [9].

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(Y_i - X_i)^2}{\sum_{i=1}^{m}(Y_i - \bar{Y})^2}$$

7.1.2 Interval spoľahlivosti

Po definovaní regresného modelu a vyjadrení kvality sa zameriame na interval spoľahlivosti, ktorý kvantifikuje neistotu predikovanej hodnoty. Regresné modely predstavujú predikčné modely, ktorých výstup je jeden bod. Táto hodnota nám nie je postačujúca, a preto potrebujeme zistiť rozsah platných hodnôt k predikovanému bodu, ktorý označujeme ako interval spoľahlivosti. Interval spoľahlivosti je znázornený na obrázku 7.1.

Obr. 7.1: Interval spoľahlivosti

Z obrázku je vidieť, že predikované hodnoty predstavujú strednú hodnotu, pre ktorú definujeme horné a dolné ohraničenie. Toto rozmedzie určuje rozsah platných hodnôt, v ktorom sa môže nachádzať vyhodnocovaná vzorka. Aby sme mohli definovať toto ohraničenie, musíme si zvoliť interval spoľahlivosti. Podľa intervalu spoľahlivosti vyberieme parameter z, ktorý predstavuje hodnotu z tabuľky normálneho rozdelenia. Časť tabuľky normálneho rozdelenia je zobrazená nižšie.

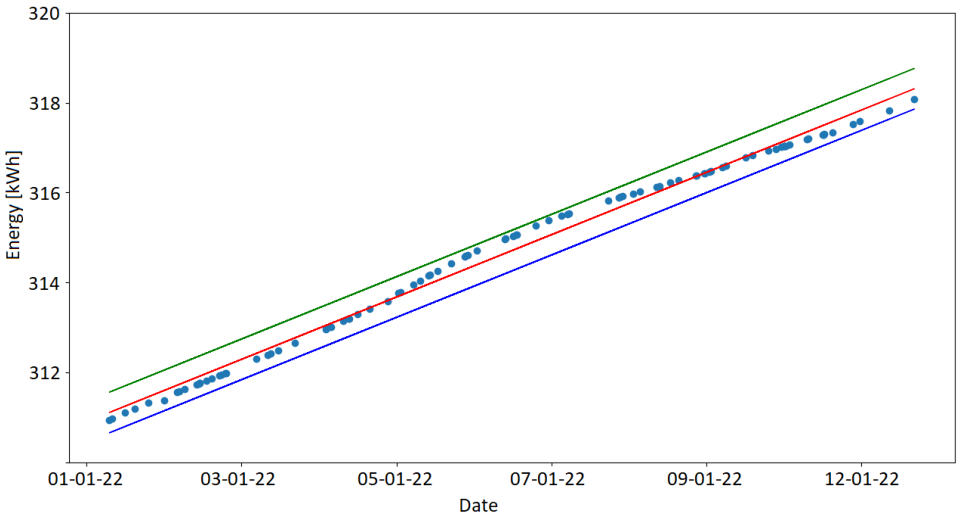
Interval spoľahlivosti	Parameter z
98 %	2.33
95 %	1.96
90 %	1.65
80 %	1.28
68 %	1

Aby sme zistili hraničné hodnoty, potrebuje vypočítať štandardnú odchýlku regresného modelu. Štandardnú odchýlku v tomto prípade budeme označovať ako RMSE. Na základe vypočítanej štandardnej odchýlky a parametru z, vypočítame rozsah platných hodnôt od predikovanej hodnoty.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y_i})^2}$$

$$< -z.RMSE\ ,\ z.RMSE >$$

Na obrázku 7.2 zobrazujeme príklad hraničných intervalov so spoľahlivosťou 98% pre senzor moniturujúci celkovú spotrebu elektrickej energie.



Obr. 7.2: Hraničné intervaly pre celkovú spotrebu elektrickej energie

7.1.3 Merenia celkovej spotreby

Elektrická zásuvka nám poskytuje pohľad na celkovú spotrebovanú elektrickú energiu. Pri skúmaní nadobúdaných hodnôt je vidieť, že celková spotreba elektrickej energie má lineárný trend, ktorý je závislý od času. Použitý bol preto model lineárnej regresie, ktorý na základe vstupnej časovej značky predpovedá očakávanú hodnotu.

V kapitole 5.3.1 sme si ukázali ako vyzerá rovnica priamky, a taktiež sme si spomenuli dve účelové funkcie MSE (Mean Squared Error) a MAE (Mean Absolute Error). Tieto parametre modelu sú z zhrnuté v tabuľke 7.1 spolu s parametrom RMSE, ktorý predstavuje druhú odmocninu MSE a parametrom R^2, koeficientom determinácie. Parametre β_0 a β_1, predstavujú parametre priamky.

β_0	$6.648.10^{-6}$
β_1	-10598.547
MSE	0.494
MAE	0.598
$RMSE$	0.703
R^2	0.998

Tabuľka 7.1: Parametre lineárneho modelu pre celkovú spotrebu elektrickej energie

Dátová sada pozostávala z nazbieraných dát za mesiac február, ktorá bola rozdelená v pomere dva ku trom pre trénovací a testovací dataset.

Na testovacej sade bolo vykonaných šesť experimentov s rôznymi intervalmi spoľahlivosti, ktoré sú zhrnuté v tabuľke 7.2. Začínali sme na intervale so spoľahlivosťou 68%, ktorej rozsah odpovedá smerodajnej odchýlke, RMSE. Postupným zvyšovaním intervalu spoľahlivosti sme dosiahli presnosť 100% na testovacej sade. Presnosť v tomto prípade vyjadruje pomer vzoriek označených ako *False Positive* voči celkovému počtu testovacích vzoriek. Testovaniu modelu sa budeme venovať v kapitole 8.

Počet vzoriek	**Počet False Positive**	**Interval spolahlivosti**	**Násobiteľ z**	**Rozsah intervalu**	**Presnosť Accuracy**
5603	2377	68%	1.00	±0.699	57.576 %
5603	1026	80%	1.28	±0.896	81.688 %
5603	383	90%	1.64	±1.156	93.164 %
5603	197	95%	1.96	±1.383	96.484%
5603	0	98%	2.33	±1.636	100.0 %
5603	0	99.7%	3.00	±2.127	100.0 %

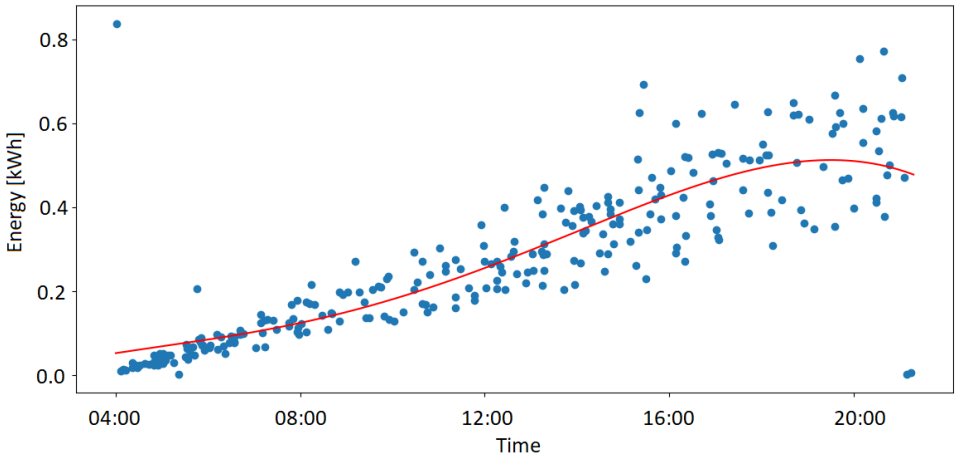
Tabuľka 7.2: Vyhodnotenie modelu pre celkovú spotrebu elektrickej energie

7.1.4 Meranie dennej spotreby

Na meranie dennej spotreby elektrickej energie sme použili polynomiálny model z kapitoly 5.3.1. Keďže sa tento jav opakuje denne, zvolili sme ako parametre čas, ktorý je interpretovaný v sekundách, a spotrebovanú elektrickú energiu. Vizualizácia dennej spotreby je zobrazená na grafe 7.3.

Podobne ako to bolo v predchádzajúcom prípade, dátová sada pozostávala z nazbieraných dát za mesiac február. Dátová sada bola rozdelená v rovnakom pomere.

Podobne ako v predchádzajúcom prípade, vychádzame z kapitoly 5.3.1 a parametre modelu sú z zhrnuté v tabuľke 7.3, ktorá je rozdelená na dve časti. Prvá časť definuje parametre modelu, ako je stupeň polynómu, vyhodnotenie účelových funkcií a koeficientu determinácie. V druhej časti sú zobrazené koeficienty polynómu, ktorým sa snažíme popísať chovanie senzoru. Rovnicu polynómu sme si ukázali v kapitole 5.3.1.



Obr. 7.3: Denná spotrebu elektrickej energie

$Degree$	4
MSE	0.007
MAE	0.053
$RMSE$	0.082
R^2	0.813

β_0	1
β_1	0.00
β_2	$7.148.10^{-15}$
β_3	$1.47.10^{-10}$
β_4	$-5.536.10^{-18}$
β_5	$-1.20.10^{-20}$

Tabuľka 7.3: Parametre polynomiálneho modelu pre dennú spotreba elektrickej energie

Presnosť modelu dosiahla takmer 99% keď sme zvolili interval spoľahlivosti 99.7 %. Obsah všetkých experimentov je zobrazený v tabuľke 7.6.

Počet vzoriek	**Počet False Positive**	**Interval spolahlivosti**	**Násobiteľ Z**	**Rozsah intervalu**	**Presnosť**
5523	1019	68.0%	1.00	±0.0833	80.549 %
5523	704	80.0%	1.28	±0.107	87.253 %
5523	393	90.0%	1.64	±0.133	92.884 %
5523	220	95.0%	1.96	±0.163	95.016 %
5523	113	98.0%	2.33	±0.197	97.954 %
5523	73	99.7%	3.00	±0.246	98.678 %

Tabuľka 7.4: Vyhodnotenie modelu pre dennú spotrebu elektrickej energie

t

7.2 Modely založené na štatistike

V kapitole 5.2 sme sa venovali dvom modelom založeným na štatistike: model normálneho a logaritmicko-normálneho rozdelenia. Týmto prístupom sme sa rozhodli vyhodnocovať senzor monitorujúci elektrické napätie a senzor monitorujúci teplotu čipu ESP32 kamery.

V časti 5.2.1 sme spomenuli, že ak chceme použiť dané rozdelenie, je vhodné overiť hypotézu, či vzorky pochádzajú z daného rozdelenia. Na tento účel sme použili *Shapirow-*

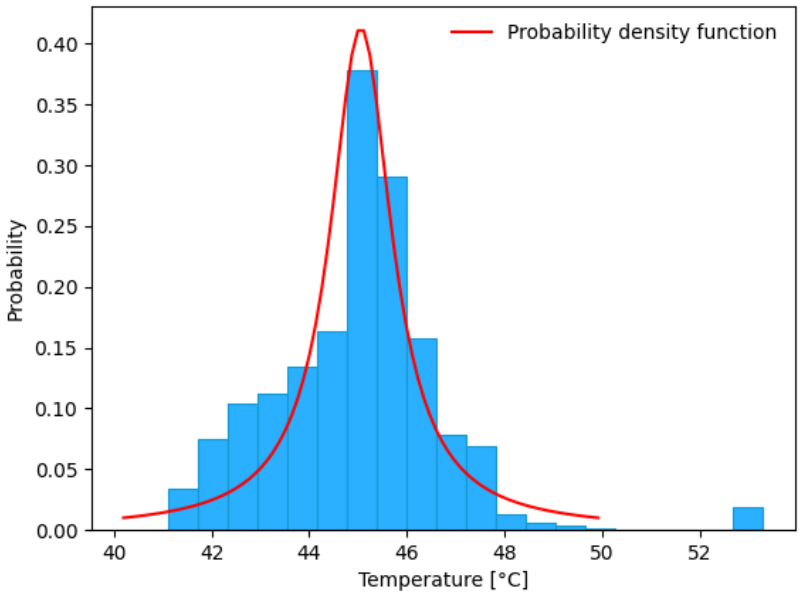
Wilkov test. Pri vyhodnotení dátových vzoriek *Shapirow-Wilkov testom* sme zistili, že v oboch prípadoch sa nejedná o normálne rozdelenie.

Aby sme určili o aké rozdelenie sa jedná, použili sme knižnicu *Fitter*[1]. Táto knižnica nám umožňuje nájsť rozdelenie, ktorému dátové vzorky najviac odpovedajú.

7.2.1 Teplotný senzor ESP32 kamery

Chovanie teplotného senzoru ESP32 kamery knižnica *Fitter* vyhodnotila, že sa jedná o logaritmicko-normálne rozdelenie, ktoré sme si spomenuli v kapitole 5.2.3.

Namerané hodnoty a priebeh Logaritmicko-normálnej funkcie je znázornený na obrázku 7.4. Histogram pozostáva nazbieraných dát za mesiac február a znázorňuje nám, že najčastejšia nameraná teplota v tom období bola približne 45°C.



Obr. 7.4: Histogram teplotného senzoru ESP32 kamery

Aj v tomto prípade sme skúmali presnosť na základe intervalu spoľahlivosti a s intervalom 99.7% sme dosiahli presnosť na testovacie sade 100%. V tomto prípade môže kolísať teplota necelé štyri stupne Celzia. Zhrnutie môžeme vidieť v tabuľke 7.5.

[1] viz https://fitter.readthedocs.io/

Počet vzoriek	Počet False Positive	Interval spolahlivosti	Násobiteľ Z	Rozsah intervalu	Presnosť
1560	417	68.0%	0.758	±1.303	73.269 %
1560	338	80.0%	0.977	±1.679	78.333 %
1560	184	90.0%	1.255	±2.155	88.205 %
1560	79	95.0%	1.495	±2.568	94.935 %
1560	13	98.0%	1.775	±3.041	99.167 %
1560	0	99.7%	2.265	±3.889	100.00 %

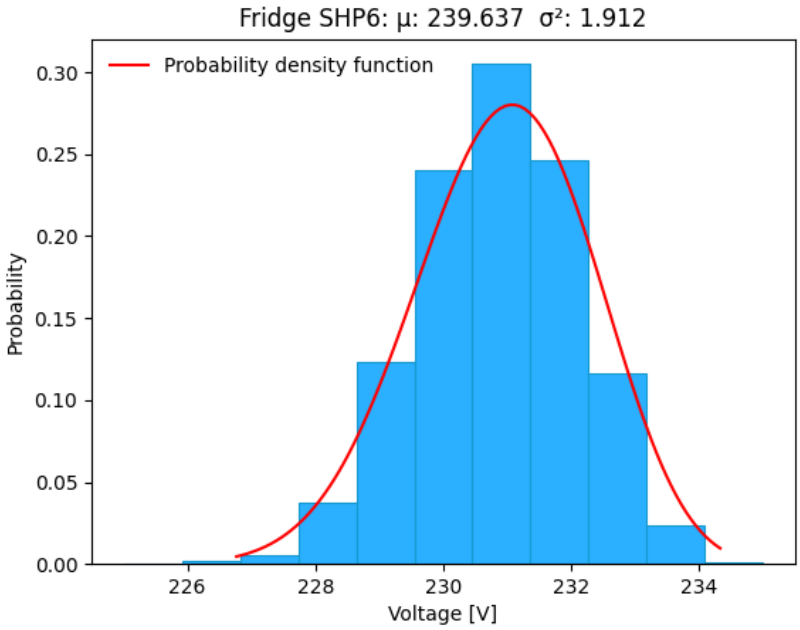
Tabuľka 7.5: Vyhodnotenie modelu pre teplotný senzor ESP32 kamery

7.2.2 Senzor merania napätia

Pri skúmaní dátových vzoriek sme zistili, že sa nejedná o normálne rozdelenie. Podobne ako v predchádzajúcom prípade sme sa snažili nájsť rozdelenie, ktoré najlepšie odpovedá nazbieraným dátam.

Nejedná sa o normálne rozdelenie podľa *Shapiro-Wilkovho* testu, ich rozloženie má ale k normálnemu rozdeleniu najbližšie, a preto sme ho v tomto prípade použili.

Stredná hodnota pre napäťový senzor je 230 V. Z pozorovaní sme zistili, že 99 % všetkých nameraných hodnôt sa nachádza v rozsahu ±3.252 od strednej hodnoty.



Obr. 7.5: Histogram a priebeh Gausovej krivky pre napäťový senzor

Počet vzoriek	Počet False Positive	Interval spoľahlivosti	Násobiteľ Z	Rozsah intervalu	Presnosť
6503	1809	68%	1.00	±1.396	72.182 %
6503	1809	80%	1.28	±1.786	72.182 %
6503	404	90%	1.64	±2.228	93.787 %
6503	404	95%	1.96	±2.735	93.787 %
6503	54	98%	2.33	±3.252	99.169 %
6503	0	98%	3.00	±4.188	100.00 %

Tabuľka 7.6: Vyhodnotenie modelu pre teplotný senzor ESP 32 kamery

7.3 Model izolačného lesa

Tento model sme použili na detekciu anomálií pre pohybový a magnetický senzor. Magnetický senzor monitoruje stav okien a dverí, či sú otvorené alebo zatvorené, a pohybový senzor monitoruje pohyb v rovnakých miestnostiach.

Hodnoty magnetického a pohybového senzoru môžu nadobúdať iba binárne hodnoty. V prípade magnetického senzoru hodnota 0 označuje stav, kedy je magnetický senzor zopnutý, čo predstavuje v našom prípade zatvorené okno či dvere.

V prípade pohybového senzoru hodnota 0 predstavuje stav, počas ktorého nebol detekovaný žiadny pohyb.

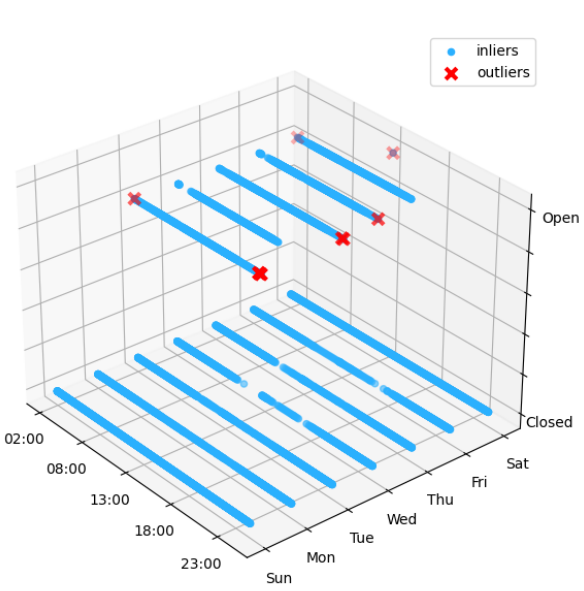
7.3.1 Magnetický senzor

Pre tento model sme brali do úvahy tri parametre. Senzory boli umiestnené v kancelárskych priestoroch, a preto sme brali do úvahy deň v týždni, pretože zmeny zachytávané týmto senzorom sa budú líšiť počas pracovného týždňa voči zmenám cez víkend. Zároveň sa pozeráme aj na čas, kedy bola evidovaná zmena stavu senzoru.

Pri použití tohto modelu je nutné vhodne definovať parametre, ktoré ovplyvňujú počet použitých dátových vzoriek a ich kontamináciu. V našom prípade sme brali do úvahy, že ak sa nemení stav senzoru, tak tento stav zapisujeme s rozdielom desiatich minút v súbore, ktorý obsahuje dátové vzorky určené na trénovanie. Zvolenou hodnotou pre počet vzoriek bola hodnota 5000. Táto hodnota predstavuje počet vzoriek, ktorými sme boli schopný pokryť viacej ako jeden mesiac nazbieraných dát.

K tomuto počtu dátových vzoriek sme zvolili najprv úroveň kontaminácie 1%, ktoré ale viedlo k množstvu vyhodnotení označovaných ako *False Positive*, a teda nesprávne identifikovaných anomálií. Tento jav nastával pri dátových vzorkách, ktorých čas bol vždy v blízkosti 24 hodiny, ±10 min. Tento jav sa nám nepodarilo odstrániť zvyšovaním vzoriek v nezmenenom stave. Kvôli tejto vlastnosti sme znížili presnosť kontaminácie na 0.05%.

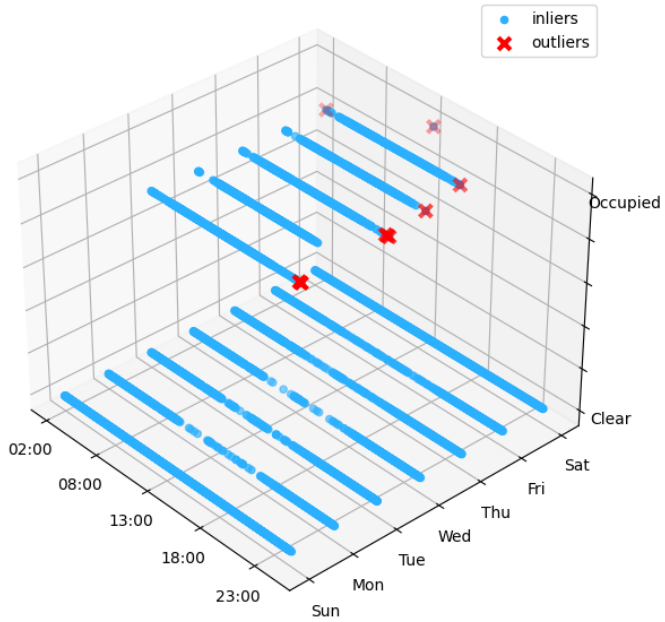
Na obrázku 7.6 je znázornené trojrozmerný graf, v ktorom znázorňujeme normálny priebeh spolu s detekovanými anomáliami. Môžeme si všimnúť, že väčšina anomálií vznikla práve vo večerných hodinách. Ale okrem týchto anomálií sú detekované anomálie počas víkendu alebo v skorých ranných hodinách.



Obr. 7.6: Grafické zobrazenie testovacej sady pre magnetický senzor

7.3.2 Pohybový senzor

Pohybový senzor používa na trénovanie rovnaké atribúty ako magnetický senzor. A teda berie do úvahy deň v týždni, čas a stav senzoru. Pri trénovaní modelu sme použili 5000 vzoriek, s ich kontamináciou 0.01%. Tými to parametrami sme boli schopný na základe testovacej sady vyhodnotiť ako anomálie dátové vzorky, ktoré predstavovali pohyb vo večerných hodinách či cez víkend. Vizualizácia testovacej sady je znázornená na obrázku 7.7.



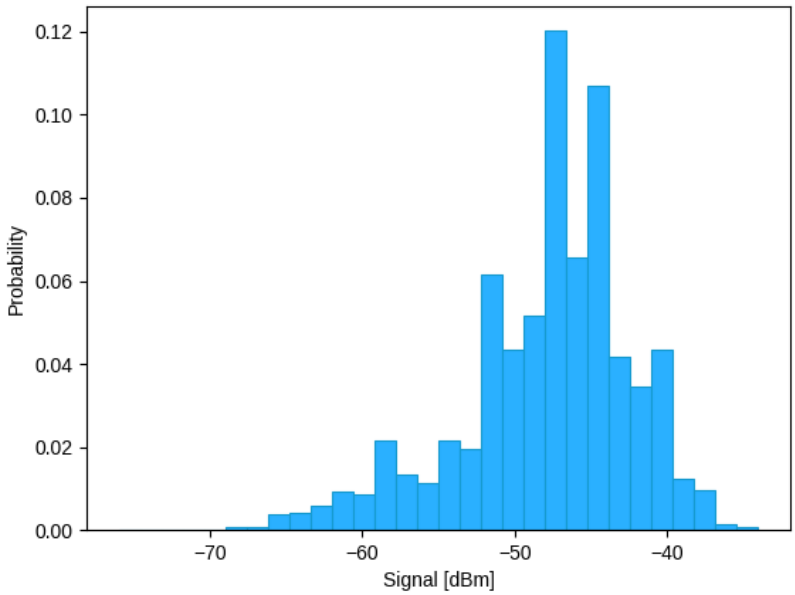
Obr. 7.7: Grafické zobrazenie testovacej sady pre pohybový senzor

7.4 Model Local outlier factor

Senzor ESP32 kamery nám poskytuje možnosť monitorovať intenzitu Wi-Fi signálu. A práve pre tento typ senzoru sme vybrali metódu založenú na zhlukovaní, Local outlier factor. Tento algoritmus je možné použiť v dvoch spôsoboch. Prvým spôsobom je tzv. novelty spôsob, pri ktorom sa model učí na prichádzajúcich sa dátových vzorkách. Druhý spôsob je použitie iba trénovacieho a testovacieho datasetu.

7.4.1 Intenzita signálu ESP32 kamery

Pre tento typ senzoru sme vybrali ako trénovacie atribúty iba hodnoty intenzity signálu. Na obrázku 7.8 je znázornený histogram nameraných hodnôt, na ktorom vidíme, že hodnota signálu sa pohybuje v rozmedzí -35 dBm až -70 dBm. Tieto hraničné hodnoty budeme považovať ako hodnoty, ktoré sa výrazne líšia od ostatných, a teda sa ich budeme snažiť vyhodnotiť ako anomálie.



Obr. 7.8: Histogram signálu ESP32 kamery

Pri vyhodnotení aktuálneho signálu berieme do úvahy parameter MinPts, ktorým určujeme počet najbližších susedov a úroveň kontaminácie datasetu. Pri nastavení parametru kontaminácia väčšou hodnotou ako 1 % dochádzalo k situácií, kedy model vyhodnotil ako anomáliu aj hodnoty v rozsahu -40 dBm až -50 dBm. Kvôli tomu chovaniu bola nastavená hodnota kontaminácie na 1 %. V tabuľke 7.7 bolo vykonaných šesť testov, v ktorých sme pozorovali aké hodnoty vyhodnotí model ako anomálie, a na základe tohto pozorovania sme určili pre kombinácie susednosti a kontamináciu intervaly povolených hodnôt. Na základe týchto pozorovaní, vidíme že interval platných hodnôt sa nezväčšuje od počtu susedov.

Počet vzoriek	Počet False/Positive	Počet susedov	Interval platných hodnôt	Presnosť
2061	1	10	<-68, -38>	99.952 %
2061	1	20	<-68, -38>	99.952 %
2061	1	30	<-67, -37>	99.952 %
2061	1	40	<-67, -38>	99.952 %
2061	1	50	<-67, -37>	99.952 %
2061	1	60	<-67, -37>	99.952 %

Tabuľka 7.7: Vyhodnotenie modelu pre senzor intenzity signálu

Pri vyhodnotení sme brali do úvahy aj použité algoritmy na výber najbližších susedov a algoritmy na výpočet vzdialenosti medzi susedmi. Knižnica scikit-learn nám umožňuje použiť tri algoritmy na zistenie najbližších susedov, BallTree, KDTree, Brute-force.

Pri použití algoritmus BallTree pri väčšom počte susedov dochádzalo k situácií, pri ktorej interval platných hodnôt obsahoval celú doménu hodnôt. Pri metóde Brute-force nedochádzalo k tomu to javu ani pri použití v rádovo väčšom počte susedov. V našom prípade sme ale použili algoritmus KdTree, ktorý prejavoval stabilitu aj pri väčšom počte susedov a zároveň jeho vyhodnotenie je rýchlejšie, ako pri metóde Brute-force.

7.5 Zhrnutie

V tejto kapitole sme sa venovali vytvoreniu modelov pre jednotlivé senzory. Pozerali sme sa na ich chovanie a úspešnosť na testovacej sade.

Ako prvý sme si ukázali model lineárnej regresie, ktorý bol použitý na meranie celkovej spotreby elektrickej energie. Aby sme nebrali do úvahy len predikovaný pod modelu lineárnej regresie definovali sme rozsah intervalu na základe intervalu spoľahlivosti. Týmto spôsobom sa nám podarilo dosiahnuť presnosti na testovacej sade 100%.

Pri polynomiálnej regresii sme postupovali rovnako. Tento model bol určený na popis chovania dennej spotreby elektrickej energie. Dosiahli sme presnosť takmer 99%.

V ďalšej časti sme sa venovali modelom založených na štatistike. Na základe testu dobrej zhody sme zistili, že vstupné dáta nepochádzajú z normálneho rozdelenia. Teplotný senzor ESP32 kamery najlepšie vystihovalo logaritmicko-normálne rozdelenie. Pri vyhodnotení senzoru, ktorý monitoruje stav napätia na elektrickej zásuvke sme použili normálne rozdelenie, aj keď podľa testu dobrej zhody hodnoty nepochádzajú z tohto rozdelenia, jeho chovanie najlepšie vystihovalo práve toto rozdelenie.

Model izolačného lesa bol použitý na popis chovania magnetického a pohybového senzoru, ktoré nadobúdajú iba binárne hodnoty. Pri vytvorení modelu sme brali do úvahy čas a deň v týždni. Na základe týchto parametrov sme boli schopný na testovacej sade detekovať neobvyklé situácie, ako napríklad detekovaný pohyb počas víkendu v kancelárií.

Posledný model, ktorý sme použili je Local Outlier Factor. Použili sme ho na popis chovania intenzity signálu ESP32 kamery. Pri nastavenej kontaminácií 1% sme menili počet susedov, braný na vyhodnotenie novej vzorky. Týmto parametrom sme nemenilo chovanie modelu. Pri vyhodnotení presnosti na testovacej sade sme dosiahli presnosť takmer 100%.

Kapitola 8

Testovanie monitorovacieho systému

V tejto časti si ukážeme akým spôsobom reaguje monitorovací systém na neočakávané hodnoty a pozrieme sa aj na úspešnosť vytvorených modelov z kapitoly 7.

Pri tomto testovaní máme dva ciele. Prvým cieľom je overenie či navrhnutý systém dokáže detekovať neočakávané situácie. Môže sa jednať napríklad o nedostupnosť senzoru alebo zmena jeho chovania. Druhým cieľom je testovanie vytvorených modelov. Budeme sa pozerať ako budú vytvorené modely reagovať na dátové sady, ktoré boli nazbierané v priebehu následujúcich mesiacov.

8.1 Test č. 1: Detekcia výpadku senzoru

V tomto teste sa budeme pozerať na reakciu monitorovacieho systému na situáciu, pri ktorej dôjde k výpadku senzoru. Použijeme pri tom ESP32 kameru, ktorú odpojíme od napájania.

Medzi Home Assistantom a ESP32 kamerou prebieha komunikácia nad aplikačným protokolom MQTT. Okrem informácií o stave senzoru si posielajú aj *keep-alive* správy, ktoré sú označované ako *PING*. Prí výpadku dôjde k prerušenia týchto *keep-alive* správ a na túto zmenu stavu reaguje ako prvý Home Assistant a označí senzor ako nedostupný.

Pri výpadku senzoru neprebieha interakcia Home Assisenta a agentom SNMP. Taktiež sa negenerujú SNMP správy označované ako trap, preto sa táto zmena neprejaví okamžite, ale až po niekoľkých sekundách.

Na prejavenie tejto zmeny vplývajú tri faktory. Prvým faktorom je doba, kým označí Home Assistent senzor ako nedostupný na základe *keep-alive* správ. Toto vyhodnotenie trvá približne pätnásť sekúnd. Ďalším faktorom je aktualizácia MIB databáze agentom SNMP. V našom prípade prebieha aktualizácia každú minútu. Posledným faktorom je dotazovanie na MIB objekty systémom Nagios, ktoré je tiež nastavené na jednu minútu.

Nižšie sú zobrazené SNMP dotazy na ESP32 kameru v dostupnom a nedostupnom stave.

```
SNMPv2-SMI::experimental.999.1.12.3.1 = STRING: "43.3"
SNMPv2-SMI::experimental.999.1.12.3.1 = STRING: "unavailable"
```

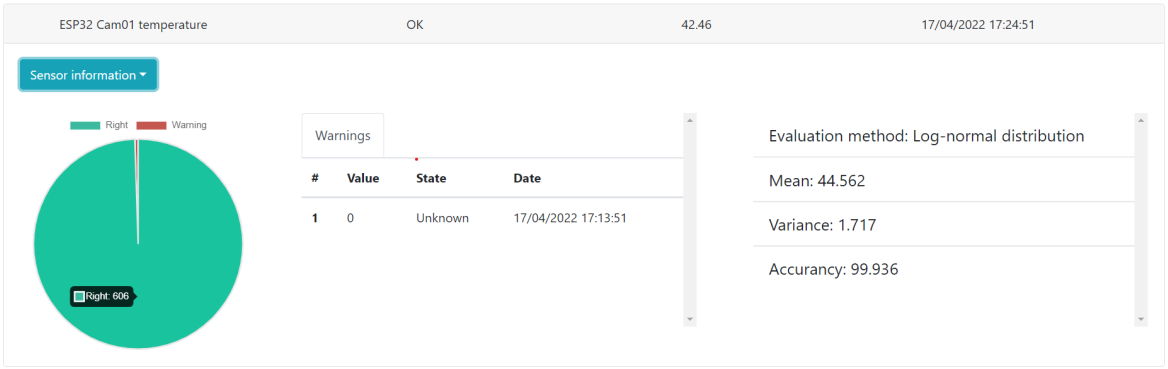
Informácia o nedostupnosti senzoru sa prejaví na dvoch miestach. V grafickom rozhraní nástroja Learning Core a v systéme Nagios v sekcii služby.

Nedostupnosť senzoru ESP32 kamery v sekcii služby môžeme vidieť na obrázku 8.1.

ESP32_cam01	ESP32_cam01_signal	UNKNOWN	05-04-2022 22:07:50	Signal - N/A
	ESP32_cam01_temperature	UNKNOWN	05-04-2022 22:07:31	Temperature - N/A

Obr. 8.1: Zobrazenie nedostupnosti senzoru systémom Nagios

Nedostupnosť sa prejaví aj v nástroji Learning Core. Nedostupnosť senzoru môžeme vidieť na obrázku 8.2. Informáciu o nedostupnosti označujeme ako *UNKNOWN*. Tento výraz môžeme vidieť na hornej lište. Táto informácia sa prejaví aj v časti *Warnings*, v ktorej je zobrazený aj dátum, kedy nastal výpadok.



Obr. 8.2: Zobrazenie nedostupnosti senzoru nástrojom Learning Core

Ako sme spomenuli na začiatku, tento typ výpadku detekuje ako prvý Home Assistent a preto rovnaký prístup vyhodnotenia bude prebiehať aj pri ostatných senzoroch.

8.2 Test č. 2: Teplotný senzor ESP32 kamery

V následujúcich testoch sa zameriame na reakciu modelov, na dátové vzorky, ktoré boli nazbierané počas následujúcich mesiacov. V tomto teste sa budeme venovať kamere ESP32, konkrétne teplotnému senzoru.

V kapitole 6.2 sme spomínali formát správ, ktoré prijíma nástroj Learning core. Nižšie môžeme vidieť ukážku tejto správy.

```
{
    "object_id": "1.3.6.1.3.999.1.12.3.1",
    "friendly_name": "ESP32 Cam01 temperature",
    "timestamp": 1647255916,
    "day": 0,
    "seconds": 39916,
    "value": 42
}
```

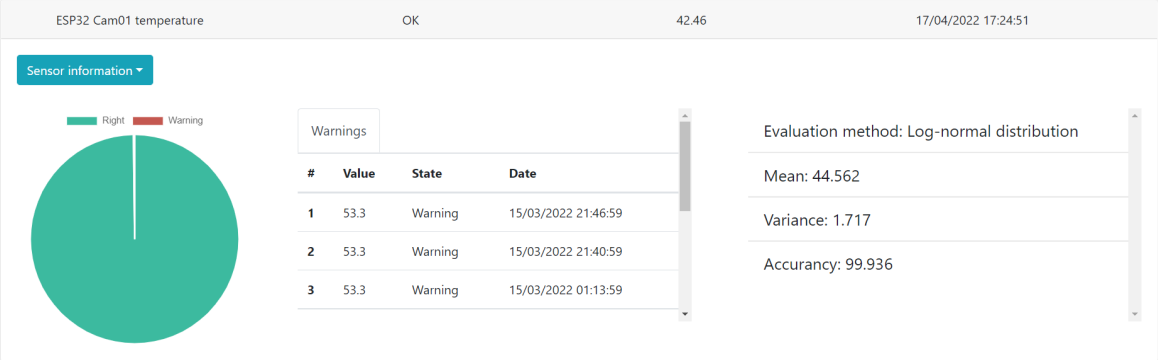
Položky timestamp, day a seconds boli pridané práve kvôli testovaniu a použijeme ich pri posielaní testovacích správ. Pri trénovaní modelov, sme nebrali do úvahy všetky dostupné dátové vzorky, ale len za mesiac február. V týchto testoch si ukážeme, ako budú reagovať modely na dátové vzorky, ktoré boli nazbierané za mesiac marec.

Na testovanie sme vytvorili program v jazyku Python, ktorý používa triedu LogManager z nástroja Learning Core. Táto trieda nám umožňuje vytvoriť logovací súbor, ktorý obsahuje všetky záznamy za mesiac marec. Tieto záznamy následne spracujeme a vytvoríme testovacie správy, ktoré budú odpovedať komunikácií za mesiac marec vďaka pridaným položkám timestamp, day a seconds. V tabuľke 8.1 je znázornené vyhodnotenie testu. Na testovacom datasete za mesiac február sme dosiahli presnosť takmer 100%. Presnosť predstavuje pomer anomálií voči celkovému počtu.

Počet vzoriek	**Počet anomálií**	**Interval spoľahlivosti**	**Násobiteľ Z**	**Rozsah intervalu**	**Presnosť**
5191	8	99.7%	2.265	±3.889	99.845 %

Tabuľka 8.1: Testovanie modelu pre teplotný senzor ESP32 kamery

Model zaznamenal osem dátových vzoriek, ktoré označil ako anomálie. Jednalo sa o vzorky, ktorých hodnota presahovala 53 $°C$. Tieto hodnoty môžeme vidieť časti *Warnings* na obrázku 8.3.



Obr. 8.3: Zobrazenie odľahlých hodnôt nástrojom Learning Core

Na obrázku 8.3 môžeme ešte vidieť v pravej časti informácie o použitom modele. Táto časť obsahuje názov použitej metódy a jej parametre, ktoré sú v tomto prípade stredná hodnota, rozptyl a presnosť na testovacej sade vstupného datasetu použitého pri trénovaní.

8.3 Test č. 3: Napäťový senzor

Na modelovanie chovania napäťového senzoru bola použitá metóda normálneho rozdelenia. Postup testovania bol totožný ako v prípade testu č.2 8.2. Vytvorili sme testovaciu dátovú sadu, ktorá obsahovala dáta za mesiac marec.

V tabuľke 8.2 je znázornený výsledok testovania. Použili sme interval spoľahlivosti 99.7% s ktorým sme dosiahli pri trénovaní na testovacej sade 100% presnosť. V tomto prípade to dopadlo rovnako. S rozdielom jedného mesiaca, model vyhodnotil všetky hodnoty ako normálne. Všetky dátové vzorky sa pohybovali ±4.188 V od strednej hodnoty.

Počet vzoriek	Počet anomálií	Interval spoľahlivosti	Násobiteľ Z	Rozsah intervalu	Presnosť
4504	0	99.7%	3.00	±4.188	100.00 %

Tabuľka 8.2: Testovanie modelu pre napäťový senzor

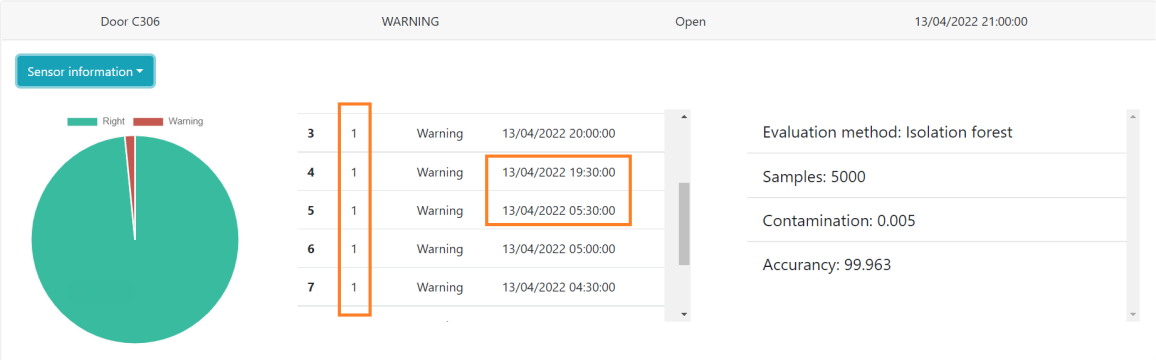
8.4 Test č. 4: Magnetický senzor

Pre magnetický senzor bol použitý model Isolation Forest, ktorého vyhodnotenie závisí na troch parametroch, a to aktuálnom čase, dni v týždni a stavu senzoru. Pre tento test sme si vybrali senzor, ktorý sníma stav dverí či sú otvorené alebo zatvorené.

V tomto teste použijeme nami vytvorené testovacie dáta, ktorými chceme ukázať časové okno, počas ktorého môžu byť dvere otvorené. Testováciu sadu rozdelíme na dve časti. Prvá testovacia sada obsahuje dátové vzorky počas týždňa, druhá práve počas víkendu.

Testovacie sady obsahovali vzorky, ktorými sme simulovali, že sú dvere otvorené počas dňa. Časový rozostup medzi jednotlivými vzorkami bol tridsať minút.

Na obrázku 8.4 môžeme vidieť vyhodnotenie testovacej sady, ktorá simulovala otvorené dvere počas týždňa. V strednej časti vidíme zoznam hodnôt, ktoré boli vyhodnotené ako anomálie. V tomto prípade 1 reprezentuje stav senzoru kedy sú dvere otvorené. Na obrázku si ešte môžeme všimnúť časové okno medzi 5:30 až 19:30, kedy neboli detekované anomálie testovacou sadou.



Obr. 8.4: Vyhodnotenie testovacej sady pre magnetický senzor

V prípade testovacej sady určenej pre víkend, boli vyhodnotené všetky vzorky, ktorých hodnoty predstavovali stav otvorených dverí ako anomálie. Tento stav bol očakávaný, pretože sa jedná o kancelárske priestory.

8.5 Test č. 5: Pohybový senzor

Pri pohybovom senzore budeme postupovať rovnako ako v prípade magnetického senzoru. Testovacia sada obsahuje dva prípady, detekciu pohybu počas pracovného dňa a počas víkendu. Obsah testov predstavuje pohyb každých 30 minút a to od 3 hodiny rannej.

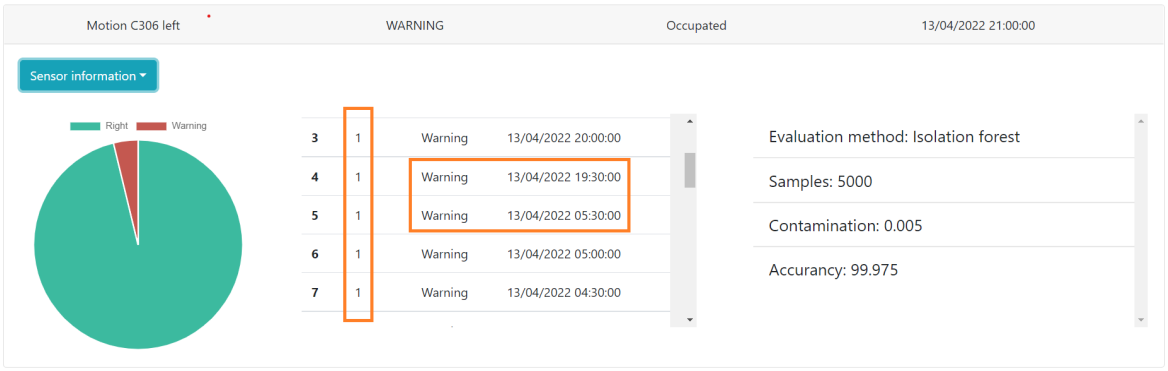
Na obrázku 8.5 si môžeme všimnúť, že rozsah očakávaných hodnôt, časové okno, počas ktorého môže byť detekovaný pohyb sa začína pred 6 hodinou rannou a končí okolo 19 hodiny.

Rovnako ako v prípade magnetického senzoru, boli vyhodnotené všetky testy určené pre detekciu pohybu počas víkendu ako anomálie.

Môžeme si všimnúť, že hodnoty v prípade magnetického senzoru na obrázku 8.4 a pohybového senzoru 8.5 sú identické a to preto, lebo tieto dva senzory sa nachádzajú v rovnakých priestoroch.

Ako anomálie sme označili vzorky, ktoré predstavovali otvorené dvere počas skorých ranných hodín.

Medzi nesprávne identifikované vzorky sme zaradili tie, ktoré predstavovali bežný pracovný deň a čas otvorených dverí spadal približne do 20:00 hodiny.



Obr. 8.5: Vyhodnotenie testovacej sady pre pohybový senzor

8.6 Test č. 6: Magnetický senzor

V teste č. 5 sme si ukázali časové okno, počas ktorého môže nadobúdať magnetický senzor hodnoty, ktoré predstavujú otvorené dvere. V tomto teste sa znova budeme venovať vyhodnoteniu na dátových vzorkách za následujúci mesiac.

Súhrn testu sme z hrnuli v tabuľke 8.3. Model vyhodnotil 21 vzoriek ako anomálie.

Pri skúmaní jednotlivých hodnôt, sme vyhodnotili, že z týchto vzoriek predstavovalo neobvyklé hodnoty iba sedem. Tieto vzorky reprezentovali stav otvorených dverí počas skorých ranných hodín.

Zvyšných štrnásť hodnôt bolo nesprávne identifikovaných, *False Positive*. Jednalo sa hlavne o prípady, kedy boli otvorené dvere medzi 19 a 20 hodinou.

Počet vzoriek	**Počet anomálií**	**Počet False Positive**
4291	7	14

Tabuľka 8.3: Testovanie magnetického senzoru

8.7 Test č. 7: Pohybový senzor

V tomto teste sme použili dátovú sadu za nasledujúci mesiac na vyhodnotenie na trénovaného modelu pre pohybový senzor. Na dátovej sade bolo vyhodnotených šestnásť vzoriek ako anomálie. Pri analýze týchto anomálií sme zistili, že desať z nich predstavovali neobvyklú situáciu. Jednalo sa pohyb v kancelárií v skorých ranných hodinách, podobne, ako to bolo v prípade magnetického senzoru.

Počet vzoriek	Počet anomálií	Počet False Positive
6476	10	6

Tabuľka 8.4: Testovanie pohybového senzoru

8.8 Test č. 8: Intenzita signálu ESP32 kamery

Pri testovaní modelu Local outlier factor, ktorý sme použili na vyhodnotenie intenzity signálu ESP32 kamery sme vychádzali z poznatkov z kapitoly 7.4.1. V tabuľke 7.7 sú zhrnuté testy, v ktorých sme skúmali intervaly platných hodnôt a pre tento prípade sme si vybrali počet susedov s hodnotou 20. Tento počet bol postačujúci, aby sa model naučil stabilný interval platných hodnôt.

Podobne ako v predchádzajúcich prípadoch, model bol trénovaný na datasate za mesiac február a testovanie pozostávalo z dát, ktoré boli nazbierané za mesiac marec. Výsledky testu sú zhrnuté v tabuľke 8.5. Bola dosiahnutá presnosť 100% na intervale platných hodnôt <-65, -37>.

Počet vzoriek	Počet anomálií	Rozsah intervalu	Presnosť
6753	0	$< -65, -37 >$	100.00 %

Tabuľka 8.5: Testovanie modelu intenzity signálu ESP32 kamery

8.9 Test č. 9: Denná spotreba elektrickej energie

Na popis spotrebovanej elektrickej energie za jeden deň sme použili model polynomiálnej regresie. Pri vyhodnotení datasetu za mesiac marec sme dosiahli presnosť 96%.

Dochádzalo k situácií, kedy bolo meranie spotrebovanej elektrickej energie reštartované pred 24 hodinou, čo spôsobilo značnú časť detekovaných anomálií.

Ďalšie vzorky, ktoré model vyhodnotil ako anomálie, boli situácie na konci dňa, kedy spotrebovaná elektrická energia presiahla hraničné hodnoty. Súhrn testu môžeme vidieť v tabuľke 8.6.

Počet vzoriek	Väčšia spotreba elektrickej energie	Predčasný reštart merania
8608	209	124

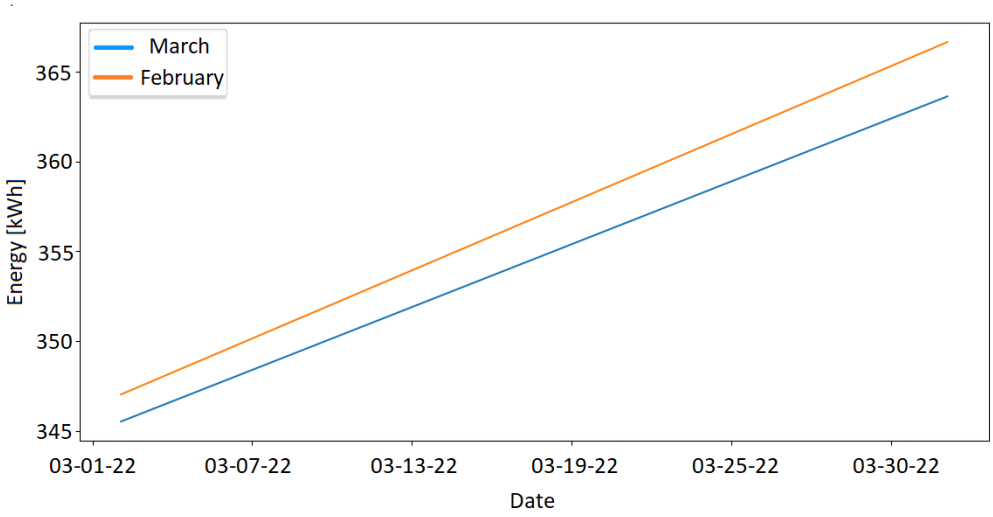
Tabuľka 8.6: Testovanie modelu pre dennú spotrebu elektrickej energie

8.10 Test č. 10: Celková spotreba elektrickej energie

Celkovú spotrebu elektrickej energie sme sa pokúsili modelovať lineárnou regresiou. V kapitole 7.1.3 sme uviedli, že pri trénovaní sme dosiahli presnosť na testovacej sade 100%. Tento stav sa nám nepodarilo zopakovať pri vyhodnotení dátovej sady za mesiac marec.

Dosiahnutá presnosť sa pohybovala okolo 95%. Od 29. dňa v mesiaci nedokázal model predikovať správne hodnoty. Na obrázku 8.6 je graf, ktorý znázorňuje dve priamky. Tieto priamky reprezentujú lineárny trend za mesiac február a marec. Môžeme si všimnúť, že

vzdialenosť medzi priamkami na konci mesiaca je väčšia ako na začiatku. V tabuľke 7.2 sme uviedli, že s intervalom spoľahlivosti 99,7% sa môže skutočná hodnota pohybovať od predikovanej hodnoty ±2.127. Od 29. dňa nastala situácia, že skutočné hodnoty presahovali hraničné hodnoty a boli následne označené ako anomálie.



Obr. 8.6: Lineárne modely pre mesiace február a marec

8.11 Zhrnutie

V tejto kapitole sme sa venovali testovaniu vytvorených modelov pre jednotlivé senzory. Trénovanie týchto modelov prebiehalo na nazbieraných dátach za mesiac február. Na vyhodnotenie modelov sme použili program v jazyku Python a nazbierané dáta za mesiac marec, ktoré sme spracovali a posielali nástroju Learning Core na vyhodnotenie.

Všetky modely dosahovali presnosť na testovacej sade vyše 95%. Pri meraní dennej spotreby elektrickej energie dochádzalo k situácií, kedy senzor reštartoval proces merania skôr, ako by mal, čo malo vplyv na výslednú presnosť.

V prípade modelu pre meranie celkovej spotreby elektrickej energie došlo na konci mesiaca k väčšej odchýlke, ako bol predikovaný stav.

Pri magnetickom a pohybovom senzore sme sa zamerali aj na testovanie časových okien, v ktorých napríklad môže byť detekovaný pohyb.

Okrem použitej testovacej sady, prebehol aj test, v ktorom došlo k výpadku senzoru. Monitorovací systém na túto udalosť reagoval a označil senzor ako nedostupný.

Kapitola 9

Záver

Cieľom diplomovej práce bolo vytvoriť monitorovací systém, ktorý nám umožní monitorovať v rovnakom prostredí IoT zariadenia, ale aj bežné zariadenia.

IoT zariadenia môžu čeliť kybernetickým útokom. Kvôli hardwárovým obmedzeniam im chýbajú bezpečnostné prvky, ako je napríklad firewall. Preto v rámci sieťovej infraštruktúry môžeme vytvoriť pre IoT zariadenia virtuálnu LAN sieť, ktorá bude zahŕňať nie len IoT zariadenia, ale aj centrálny bod, Home Assistenta.

Použitím Home Assistenta dokážeme minimalizovať komunikáciu medzi IoT sieťou, a zvyšnou sieťovou infraštruktúrou. Zároveň môžeme vytvoriť obmedzenie, ktorým umožníme komunikovať IoT zariadeniam iba v rámci ich virtuálnej LAN siete.

Medzi najpopulárnejšie voľne dostupné monitorovacie systémy patrí Nagios core, ktorý sme použili v našej práci. Systém Nagios sme rozšírili o plugin, ktorým vytvárame logovacie súbory a nástroj Learning Core, ktorého úlohou je vytvoriť model, ktorým definuje chovanie IoT zariadenia. Nástroj Learning Core môžeme použiť na popis chovania nie len IoT zariadení, ale aj na popis chovania rôznych služieb bežných zariadení.

Práve nástroj Learning Core slúži na detekciu anomálií v IoT sieti. Vytvorené modely nástrojom Learning Core slúžia na popis bežného chovania monitorovaných zariadení a dátové vzorky, ktoré sa výrazne líšia od vytvoreného modelu vyhodnotí ako anomálie.

V našom prostredí sme sa dotazovali na stav IoT zariadení priebežne každú minútu. Snažili sme sa zachytiť, čo najpresnejšie chovanie monitorovaných zariadení. V prípadoch, ako sú senzory, ktoré monitorujú celkovú a dennú spotrebu elektrickej energie môžeme znížiť interval dotazovania na stav senzoru systémom Nagios. Správanie týchto senzorov má lineárny trend a prípadná odchýlka od normálneho chovania sa prejaví aj pri menej častom dotazovaní. V prípade pohybového alebo magnetického senzoru môžeme požadovať presnejšie informácie. Interval na dotazovanie stavu senzoru upravíme na požadovanú hodnotu.

Aktívne monitorovanie IoT siete prebiehalo niekoľko mesiacov. Na základe nazbieraných dát z IoT siete sa nám poradilo vytvoriť modely, ktoré definujú normálne chovanie jednotlivých senzorov. Počas tohto monitorovania sa nám podarilo detekovať neobvyklé situácie, ktoré sa následne reportovali systému Nagios.

Výstupom práce je agent SNMP určený pre platformu Home Assistent a nástroj Learning Core, ktorý môžeme použiť ako rozšírenie pre systém Nagios na detekciu anomálií.

Literatúra

[1] The Robotics Institute, School of Computer Science, Carnegie Mellon University. Dostupné z: https://www.cs.cmu.edu/~schneide/tut5/node42.html.

[2] *Object Definitions*. Dostupné z: https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/3/en/objectdefinitions.html#command.

[3] IEEE Standard for Low-Rate Wireless Networks. *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*. 2016, s. 1–709.

[4] Blumenthal, U. *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)*. IETF RFC 3414. December 2002. 1-88 s.

[5] Breunig, M. M., Kriegel, H.-P., Ng, R. T. a Sander, J. LOF: Identifying Density-Based Local Outliers. New York, NY, USA: Association for Computing Machinery. May 2000, zv. 29, č. 2, s. 93–104. ISSN 0163-5808.

[6] Case, J., Fedor, M., Schoffstall, M. a Davin, J. *A Simple Network Management Protocol (SNMP)*. IETF RFC 1157. May 1990. 1-36 s.

[7] Case, J., McCloghrie, K., Rose, M. a Waldbusser, S. *Introduction to Community-based SNMPv2*. IETF RFC 1901. December 2002. 1-26 s.

[8] Chandola, V., Banerjee, A. a Kumar, V. Anomaly Detection: A Survey. New York, NY, USA: Association for Computing Machinery. 2009, zv. 41, č. 3. ISSN 0360-0300.

[9] Chicco, D., Warrens, M. J. a Jurman, G. The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. *PeerJ Comput Sci.* 2021, zv. 7, s. 24.

[10] Ginos, B. F. *Parameter estimation for the lognormal distribution*. Brigham Young University, November 2009.

[11] Han, J., Kamber, M. a Pei, J. *Data Mining: Concepts and Techniques*. 3nd. USA: Morgan Kaufmann, June 2011. ISBN 9780123814807.

[12] Hyndman, R. a Athanasopoulos, G. *Forecasting: Principles and Practice*. 2nd. Australia: OTexts, 2018.

[13] Liu, F. T., Ting, K. M. a Zhou, Z.-H. Isolation Forest. In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, s. 413–422. DOI: 10.1109/ICDM.2008.17.

[14] Matoušek, P. *Síťové služby a jejich architektura.* Publishing house of Brno University of Technology VUTIUM, 2014. 396 s. ISBN 978-80-214-3766-1.

[15] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., McCloghrie, K. et al. *Structure of Management Information Version 2 (SMIv2).* IETF RFC 2578. April 1999. 1-36 s.

[16] Ostertagova, E. *Aplikovaná štatistika v počítačovom prostredí MATLABu.* June 2015. 175 s. ISBN 978-80-553-2089-2.

[17] Presuhn, R., Case, J., McCloghrie, K., Rose, M. a Waldbusser, S. *Management Information Base (MIB) for the Simple Network Management Protocol (SNMP).* IETF RFC 3418. December 2002. 1-26 s.

[18] Rong, Shen a Bao-wen, Zhang. The research of regression model in machine learning field. *MATEC Web Conf.* 2018, zv. 176, s. 01033.

[19] Safaric, S. a Malaric, K. ZigBee wireless standard. In: *Proceedings ELMAR 2006.* 2006, s. 259–262.

Príloha A

Obsah DVD

- learning_core_install/ - adresár obsahuje zdrojové kódy pre nástroj Learning Core, nagios plugin, konfiguračný súbor httpd, konfiguračný súbor pre vytvorenie služby v operačnom systéme. Obsahuje ešte skript *install.sh*, ktorý obsahuje všetky požadované knižnice, programy a umiestnenia jednotlivých súborov.
- snmp_agent/ - adresár obsahuje zdrojové kódy SNMP agenta. Jedná sa o Add-on pre Home Assistant platformu.
- log/ - adresár obsahuje logovacie súbory monitorovaných zariadení.
- text_prace/ - adresár obsahuje súbory k textu diplomovej práve.
- xkrajc17.pdf - text diplomovej práce.

Príloha B

Inštalácia a spustenie

Adresár learning_core_install nakopírujeme na cieľový server, na ktorom budeme mať spustený nástroj Learning Core a systém Nagios. Pred tým ako spustíme inštalačný skript, musí už byť nainštalovaný systém Nagios.

Inštalačný skript je určený pre operačný systém CentOS. Po jeho aplikácií, budeme mať k dispozícii službu core, ktorou spustíme nástroj Learning Core.

```
cd learning_core_install
chmod +x install.sh
./install.sh
service core start
```

Pred spustením nástroja Learning Core, musíme ešte skontrolovať, či boli správne nekopírované logovacie súbory *ls /var/log/nagios*. Ak adresár neobsahuje logovacie súbory, je nutné ich tam prekopírovať z inštalačného adresára *install_dir/log/*. V prípade neúspešného kopírovania služby, je tiež nutné vykonať kopírovanie manuálne. Kopírujeme z adresára *install_dir/service/*.

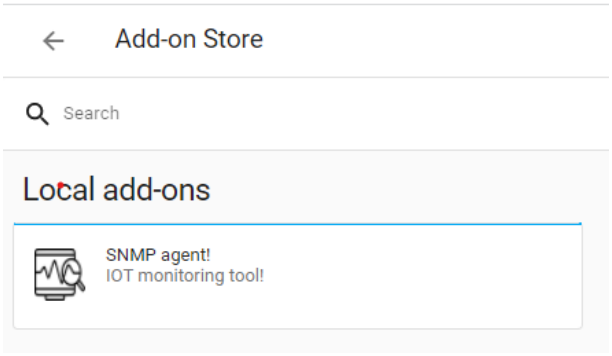
```
cp ./service/* /usr/lib/systemd/system/
systemctl enable core.service
systemctl start core.service
```

Posledný krok, ktorý musíme vykonať, je zmena IP adresy pre databázový server. Tieto zmeny musíme vykonať v */usr/local/learning_core/core/learning_core.py* a */usr/local/learning_core/share/global.php*.

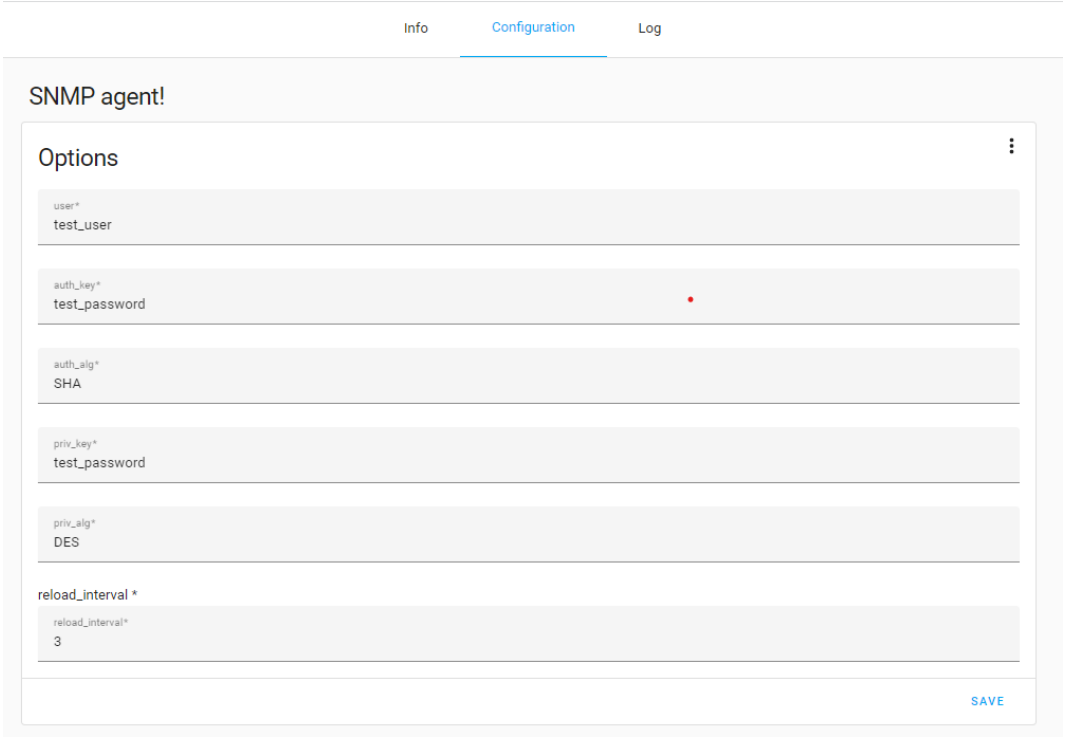
Inštalácia agenta SNMP prebieha priamo na Home Assistentovi. Inštalácia prebieha z lokálneho uložiska. Adresár snmp_agent nakopírujeme do adresára */usr/share/hassio/addons/local*. Následne sa môžeme presunúť do grafického rozhrania Home Assistenta, v ktorom vykonáme nasledujúce kroky:

- V menu prejdeme do zložky *Configuration*.
- Klikneme na záložku *Add-ons, backups & Supervisor*.
- V pravej dolnej časti klikneme na *add-on store*.
- V pravej hornej časti klikneme na menu znázornené bodkami a aktualizujeme dostupné Add-on kliknutím na *Check for updates*.

Po týchto krokoch by sme mali mať dostupný Add-on agenta SNMP, ako môžeme vidieť na obrázku B.1. Po otvorení Add-on môžeme spustiť inštaláciu. Po inštalácií môžeme vykonať nastavenie agenta SNMP, a to upraviť prihlasovacie údaje a interval aktualizácie MIB databáze. Zmeny môžeme vykonať v záložke *Configuration*. Zmena prihlasovacích údajov je znázornená na obrázku B.2



Obr. B.1: Agent SNMP v prostredí Home Assistenta



Obr. B.2: Konfigurácia agent SNMP