TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Tarkvarateaduse instituut

Mihkel Kasepuu

153393IAPM

VEE TARBIMISE ANALÜÜS KAUGLOETAVATEST VEEMÕÕTJATEST

Magistritöö

Juhendaja: Marko Kääramees
PhD
Lauri Ilison PhD

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Mihkel Kasepuu

07.05.2017

Annotatsioon

Käesoleva töö põhiline temaatika on analüüsida asjade interneti mõjust tingitud tarbimisandmete informatsiooni veetarbimise valdkonnas. Töös kogutakse andmeid üle asjade interneti tarbeks mõeldud *LPWAN* võrgu reaalsete lõppkasutajate käest.

Andmeid analüüsitakse ning püütakse leida kasulikku informatsiooni, mille põhjal on võimalik väita, et perioodil toimis mingi tegevus. Antud informatsiooni kasutatakse ära sündmuste tuvastamisel, mis omakorda teostatakse klastritesse jagamise teel. Tulemusena valmib kirjeldav informatsioon katse objekti tüüpilisest käitumisest.

Käsitletakse andmete turvalisuse temaatikat ning võrreldakse kui tihe peab olema andmete edastus, et lõppkasutaja privaatsust ei eirataks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 63 leheküljel, 6 peatükki, 19 joonist, 10 tabelit.

Abstract

Usage analysis from remote water sensors

The main reason of the thesis is to analyse water usage following internet of things influence on gathering it information. Information is gathered using *LPWAN* network that was made for internet of things devices.

The data will be analysed to gather useful information for discovering certain activities in different periods. The gathered information will be used to classify usage events which will be discovered by clustering usage data. The result is describing information about test subjects' usual behaviour.

The thesis covers data security and privacy topic and compares how frequent data should be, so that the end users' privacy remains intact.

The thesis is in Estonian and contains 63 pages of text, 6 chapters, 19 figures, 10 tables.

Lühendite ja mõistete sõnastik

LoRaWAN	Tehnoloogia, *LPWAN-i* üks versioon
LPWAN	*Low Power Wide Area Network*
NAS	Nordic Automation Systems
API	Rakendusliides päringute tegemiseks
Iothub	NAS-i poolt pakutud API
JSON	Andmevahetusvorming: *Javascript Object Notation*
AuthKey	Iothub-i kasutaja identifikaator API poole pöördumiseks
EUI, DevEUI	Iothub-i seadme identifikaator API poole pöördumiseks
Python	Programmeerimiskeel
Jupyter	Interaktiivset tüüpi andmeteaduse projektide arenduskeskkond veebilehitsejas

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

1.1 Taust

Suurandmete kogumine on muutumas järjest populaarsemaks 21. sajandi infotehnoloogia maastikul. Rakenduste arendamisel ei arutletagi küsimuse üle, et kas esmapilgul kasutuid andmeid kustutada või mitte, soovitav on alles jätta, et hiljem vanade andmete analüüsimise tulemusest kasu tekitada. Asjade interneti kontseptsioon toob andmete kogumise, talletamise ja analüüsimise valdkonda täiendavaid probleeme ning ka võimalusi, kust võib järeldada, et suurandmete valdkond on asjade interneti kontseptsiooni osa ning sh. üks tähtsamaid osi.

Eestis on tänaseks loodud *LoRaWAN*-il [1] baseeruv *LPWAN*-i [2] võrk, mis on mõeldud asjade interneti suhtlusvõrguks ning millesse saab ühendada mitmesuguseid andureid. Käesoleva töö probleemi lahendus kasutab *LoRaWAN*-i võrku, millele andis ligipääsu Nordic Automation Systems[3] (edaspidi NAS).

NAS pakub *LPWAN* võrgu abil mitmeid asjade internetiga seonduvaid teenuseid ning plaanib antud valdkonnas laieneda. Nende üheks populaarsemaks tooteks on kaugloetav veemõõdik, mida kasutatakse ka käesolevas töös. Antud sensorit on müüdud nii Eesti turul kui ka mujal, mis tähendab, et ettevõttel esineb mitmeid klientide probleeme veetarbimise valdkonnas. Eestis on NAS-i kõige kõlavam temaatika lisaks avalikule *LPWAN* võrgule kauglugemise võimaldamine veemõõdikutelt.

1.2 Probleem

Valdkond, mida kauglugemine toetab on targa maja lahenduste pakkumine ning kuulub ka NAS-i huvi orbiiti. Selleks, et tark kodu või maja oleks tõeliselt nutikas, peab see oskama infot töödelda kodu kõikidelt võimalikelt anduritelt pärinevat informatsiooni, seal hulgas ka tarbimisandmetest. See püstitab käesoleva töö esmase probleemi, ehk kas ajateljel kujutatud tarbimisandmetest on võimalik leida kasulikku informatsiooni? Antud informatsiooni põhjal saab rakendusi treenida ning õpetada kindlate pere harjumuste raha

säästmise valdkonnas. Teades, iga kodus asuva eseme kasutuses oleku perioode, suudab seal elav pere mõne targa lahenduse abil igakuise kulu arvelt sääste suurendada, õppides ajavahemikke, millal pere kodus on.

Tänapäeva andmeteaduses on sageli võimalik järeldada andmetest palju enam kui algselt näib. Kui kauglugemise teel leitud andmestiku seas esineb sagedasti esitatud väärtusi, võivad andmed pakkuda informatsiooni, mis algul välja ei paista. Käesoleva töö raames püütakse vastata hüpoteesile, kas edastatud tarbimisandmetest võib järeldada erinevate sündmuste esinemist, kui tihti neid esineb ning kui tihedalt peab olema andmeid, et neid järeldada? Tuvastades sündmusi, on NAS-il võimalik pakkuda erinevaid teenuseid nii seadmete juhtimisel kui ka statistika kuvamiseks.

Kauglugemise puhul on oluline temaatika andmete turvalisus. Seoses andmete tiheda kogumisega, et tagada erinevate lahenduste pakkumine (nt. lekked), võib esineda võimalus, et nende seast on võimalik tuvastada enam käitumismustreid, mis riivavad lõppkasutaja huvisid juhul kui sellele luba pole antud. Neid antud andmeid saab müüa või teisteks ärisuundadeks kasutada (reklaam). Euroopa liidu andmekaitse reform [4] ainult tõstab probleemi olulisust infosüsteemides ning sellega peavad tulevased rakendused arvestama. Seega on probleemiks andmete kogumisest tulenev turvalisus lõppkasutajale, kas kliendi andmete küsimine kindla tihedusega on mõistlik ja turvaline. Seega püstitab käesolev töö küsimuse, et kui sage peab kaugloetav tarbimisinfo, et selle abil ei oleks võimalik kirjeldada eraisikute eraelu toimimist ning kas see on üldse võimalik.

1.3 Eesmärk

Leida objektid, kus esineb igapäevane tarbimine ning võimaldada neist kauglugemine üle Nordic Automation Systems-i pakutud *LoRaWAN* võrgu. Kirjutada esmane prototüüp rakendus, mis pöördudes Nordic Automation Systems-i veebiliidese poole kogub tarbimisandmed erinevatelt objektidelt analüüsiks. Analüüsida ja visualiseerida neid andmeid ning võrrelda eri objektide tulemusi omavahel.

Tuvastada tarbimisandmete seast perioodiliselt sagedased ning sarnased andmehulgad, defineerides neid kui sündmusi ja tegevusi, anda neile nimi ning kajastada neid visuaalselt. Selleks valitakse välja algoritm, et ajajoonelt sündmusi tuvastada. Klassifitseerida käitumismustrid andmetest ning kajastada neid ajajoonel visuaalselt.

Tuvastada, kui sagedasest näidu edastamisest võib välja lugeda enam, kui ainult näidu muutu. Selleks loetakse andmed erineva sagedusega ning võrreldakse omavahel, kasutades varem ülesehitatud lahendusi.

1.4 Motivatsioon

Tänu asjade interneti arengule on tekkinud palju uusi võimalusi andmeteaduse valdkonnas. Andmete hulga eskaleerumise kiirus on asjade interneti valdkonnas palju kiirem tavapärasest [5], millest järeldades on andmeteaduse head praktikad erinevate rakenduste loomisel äärmiselt olulised.

Asjade interneti valdkonna arengust tulenevalt on mitmete valdkondade mõõtmine palju põnevam kui varem. Kui varem korjati elektrinäitu kord kuus, saadetakse nüüd voolu tarbimisandmeid Eestis iga tunni vahetumisel, mis võimaldab pakkuda ka tunnipõhist hinda. Veenäidu valdkonnas andmete tihedat mõõtmist Eesti turul sageli ei leia ning käesolev töö püüab analüüsida, mis informatsiooni lühikeses ajavahemikus mõõtmine tegelikult omab.

1.5 Ülevaade tööst

Käesolev töö koosneb seitsmest peatükist. Peatükk 2 annab valdkonna ülevaate selles esinevate üldiste mõistete kirjelduste ja sarnaste tööde kirjeldamisega. Seejärel tutvustatakse NAS-i võrgu ülesehitust, tänu millele käesolev töö sai võimalikuks.

Peatükk 3 alustab andmete uurimist ning seda käesoleva töö probleemide lahendamise võtmes. Püütakse tuvastada kindlaid perioode, mis kirjeldavad elu toimimist. Kõik antud uurimised viiakse läbi kasutades programmeerimise võtteid.

Peatükk 4 käsitleb andmehulgast sündmuste tuvastamise võimalust ning püütakse reaalelulisest andmehulgast sündmusi tuvastada.

Peatükk 5 kirjeldab elu toimimist visuaalsel teel, lisades sinna leitud sündmuste osas informatsiooni. Lisaks käsitletakse andmete turvalisuse ning privaatsuse temaatikat. Püütakse pakkuda sagedust andmete edastamiseks, mis tagab privaatsuse.

Käesolev töö lõpetab lahendustega, mis leitud tulemusi kasutades võimaldaks tulevaste eesmärkide täideviimist. Käesolev töö kirjeldab kõiki tegevusi, mis autor tegi, et tulemusteni jõuda.

2 Asjade internet mujal ja Eestis

Asjade internet on tuntud arenev temaatika informatsiooni töötluse valdkonnas. Teema on kiiresti arenev, sest interneti võimekus mitmete seadmete puhul on pidevalt suurenev [6]. Temaatika põhiline punkt on, et kõik seadmed või asjad oleks interneti ühendatud ning nende vaheline suhtlus oleks pidev.

Uurime, mis on sarnased tööd ning vaatleme, millised võimalused on asjade interneti valdkonnas tegutsemiseks Eestis.

2.1 Sarnaseid töid

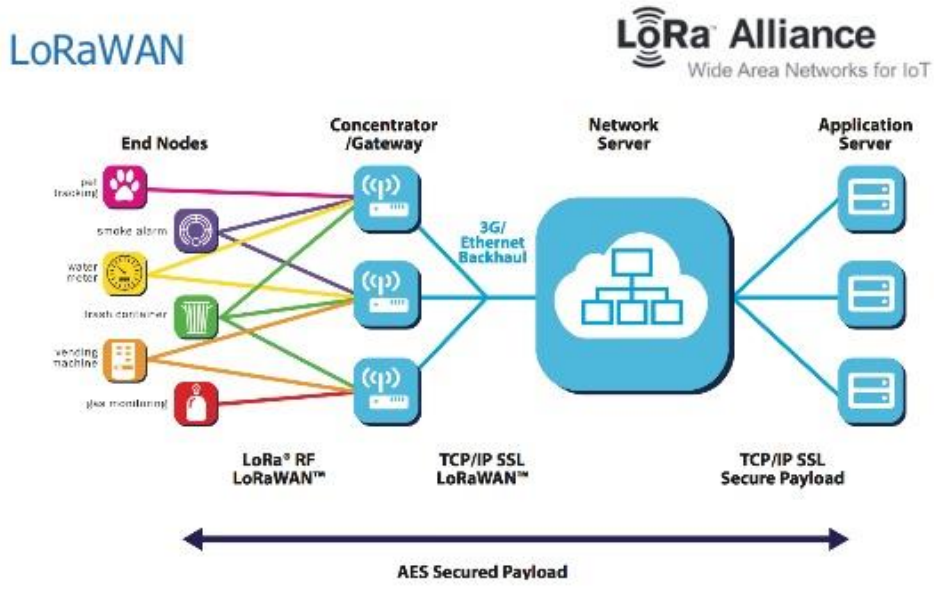
Sarnaselt käesolevale tööle on proovitud andmeid koguda ja klassifitseerida lisades andureid enam kui ühele seadmele koduses majapidamises [7][8], kus andmete täpsus iga seadme kohta on parem, kuid seetõttu on neid keerulisem üles seada igasse majapidamisesse, et sellest laialdaselt kasu leida. Antud juhul olid osad andurid mõõtmas lisaks veenäidule(tarbimisele) veel heli, või rõhku. Heli puhul näiteks püüti tuvastada, kas kasutati pesumasinat või tualetti [9]. Sarnaselt käesolevale tööle toodi ka välja, et lisaks näidu lugemisele on oluline ka hetke tarbimisvoo uurimine.

Ainuüksi sensorid ja mõõtmistulemused ei aita ühiskonna heaks väärtust luua. Põhiline vajalik informatsioon on siiski andmete analüüs. Näiteks on uuritud kuidas kõrge magnituudiga muutusi nagu näiteks vihmasadu saab ära kasutada kodutarbimises säästmiseks [10][11]. Ka siis uuriti andmehulgast vajalik info välja, identifitseeriti ja klassifitseeriti info ära ning leiti sellele vastav kasutuslugu, et lõppkasutaja ära tunda. Tugevalt lähtuti ka ajaperioodil toimunud suurtele muutustele veekulu osas.

2.2 Nordic Automation Systems

Nordic Automation Systems on Eestis baseeruv tööstusautomaatikat tootev ettevõte. 2016 aastal alustas ettevõte tegutsemist targa mõõtmise ning targa monitooringu valdkonnas, tänu millele on käesoleva töö teostamine võimalik. Nagu varasemalt mainitud on NAS ülesehitanud oma asjade interneti arhitektuuri Lora[1] poolt pakutud LPWAN võrgule LoraWAN[12].

2.2.1 Võrk ja sensor



Joonis 1 LoraWAN-il põhinevate rakenduste tüüpiline arhitektuuriskeem[13]

LoraWAN on LPWAN-i spetsifikatsioon, mis on mõeldud juhtmevabalt ning aku toitel põhinevatele asjadele suhtlusvahendiks[12]. LoraWAN-il põhinevate süsteemide arhitektuur on tüüpiliselt üles ehitatud tähevõrguna, kus iga täht on ühendatud rakendusserveriga ning on väravaks mitmetelt sensoritelt või asjadelt tulnud infole ja kus sensorid moodustavad veel omakorda tähtvõrgu värava tähe külge, nagu on näidatud Joonisel 1.

LoraWAN on defineerinud ära 3 klassi, kuidas sõnumivahetust teostada (Tabel 1).

Tabel 1 LoraWAN-i klassid[12]

Klass	Kirjeldus
A	Kahepoolne suhtlus, kus asjal on defineeritud saatmise reeglid, või tsükkel ning pärast igakordset infosaatmist avab lühiajalise vastuvõtu tsükli.
B	Kahepoolne suhtlus, kus asjal on defineeritud saatmise reeglid, või tsükkel ning avab vastuvõtu tsükli kindlatel ajaperioodidel.
C	Kahepoolne suhtlus, kus asjal on defineeritud saatmise reeglid, või tsükkel, ning omab avatud vastuvõtu tsüklit ajaperioodidel, millal ei toimu samaaegset saatmist.

Kuna käesolev töö keskendub veenäidu andmetele, on ka põhiliseks objektiks LoraWAN-i võrgus veenäidu lugemise sensor, mis kasutab klassi A ning millele ise sõnumeid ei saadeta. Käesolev töö kasutab NAS-i poolt pakutud CM3000A veenäidu lugemise sensorit (Joonis 2).

Joonis 2 Töös kasutatav veenäidu lugemise sensor

Antud sensor (Joonis 2) käib otse olemasoleva veemõõtja peale. Seade on tehniliselt impulsi lugeja, loendades kas veemõõtja tiiviku tiirlemise kordi või mõne veemõõtja poolt pakutavat optilist impulssi. Käesolevas töös loeb sensor impulsse ning saadab need

kindla ajavahemiku möödumisel serveri poole. Server salvestab andud väärtused ning võimaldab neid pärida üle pakutava API.

2.2.2 API

Käesolev töö kasutab andmete korjamiseks NAS-i pakutud API-t, mis on REST liides ning on veebirakenduse Iothub (iothub.nasys.no) üks osa. Iothub pakub lihtsalt arusaadavaid võimalusi asjadelt informatsiooni kättesaamiseks ning neile saatmiseks. Iga asi või seade omab oma unikaalset identifikaatorit (nimetatakse EUI või DevEUI), mis on tekstiväärtuse kujul. Samuti ka iga kasutaja omab identifikaatorit (nimetatakse AuthKey-ks), millega ta saab ainult endale määratud asjade poole pöörduda. Iothub-i API tagastab UTF-8 formaadis JSON-i tekstiväärtusi ning selle poole saab pöörduda kasutades HTTP päringu meetodeid [14].

Käesolev töö kasutab peamiselt järgmist päringut:

```
GET
iothub.nasys.no/api/node/{AuthKey}/{EUI}/RX/{offset}/{limit}/{asc/desc}
```

Antud päring tagastab konkreetse seadme poolt saadetud informatsiooni, mis on antud hetkel Iothub-i serverisse salvestatud. Vastavalt parameetritele *limit* ning *offset* saab pärigu tulemuse suurust muuta. Lisaks saab muuta päringu järjekorda. Näide päringu tulemusest JSON kujul, kus *offset* on 0 ja *limit* on 1:


```
{
  "total": "15951",
  "result": [
      {
            "adr": "true",
            "bandwidth": "0",
            "codeRate": "",
            "data": "AAcAAAAVlJYA",
            "date": "2017-02-17 18:41:09",
            "fcnt": "2",
            "fport": "24",
            "frequency": "0",
            "gatewaycount": "1",
            "id": "885805",
            "modulation": null,
            "rssi": "-103",
            "spreadFactor": "0",
            "timestamp": "2147483647"
       }
   ]
}
```

Vastuse JSON-i väljal *total* on kirjas, mitu kirjet antud seadme kohta üldse serveril on, väljal *result* on jada kirje tulemustest, mille hulk sõltub serveris olevatest andmetest selle seadme kohta ning päringu parameetritest. Jada iga liige on JSON objekt, mis hõlmab kindla saatmiskorra hetkel eksisteerivat seadme poolt pakutavat informatsiooni.

Antud JSON objekti sees on mitmeid välju. Käesoleva töö raames on olulised väljad *fport*, *data* ning *date*, kus esimene väli määrab välja *data* tüübi, teine on seadme või asja väärtus ajahetkel, mis on määratud väljal *date*. Välja *data* väärtus on baidijada, mis on kodeeritud base64 tekstiväärtuseks. API-t kasutavad rakendused peavad eri tüüpi seadmete korral oskama vastavalt välja *fport* väärtusele erineval viisil antud baidijada kasulikuks infoks konverteerida. Käesolev töö arvestab *fport* väärtusega 24, mis on NAS-i poolt defineeritud kui staatuse formaat, kuid selle hulgas on ka veenäidu informatsioon, mille leidmist käsitletakse täpsemalt peatükis 3 Andmete analüüs.

2.3 Töö teostus / Metoodika

Käesolev töö käsitleb peamiselt veenäidu andmete analüüsi ning sealt kasuliku info leidmist. Selleks on vaja esmalt veenäite koguda ning need sobivasse formaati konverteerida. Et tagada parem võrdlusmoment, kogutakse andmeid enam kui ühest asukohast. Andmed kogutakse programmi, mis kirjutatakse käesoleva töö raames ning mille esmane ülesanne on neid kuvada graafikutena, et leida võimalikku kasulikku informatsiooni triviaalsel viisil, ehk uurides neid visuaalselt.

Kui andmed on programmi koondatud, uuritakse neid edasi erinevate statistiliste näitajate poolest ning tehakse tulemustest järeldused. Kuvatakse tulemusi eri perioodide vältel näiteks nädalapäevade vaheline võrdlus, õhtuste ja hommikuste perioodide võrdlus.

Andmete analüüsi järgselt töötatakse välja perioodi tarbimisandmete muudu põhjustanud sündmuste tuvastamise metoodika. Selleks valitakse sobivaim algoritm sündmuste tuvastamiseks töös kirjeldatud olukorrale. Kui sündmuste tuvastamine on võimaldatud, tuuakse tulemid välja käitumismustritena ning võrreldakse neid eri objektide vahel.

Sündmusi uuritakse ka turvalisuse valdkonnas. Kas tuvastatud informatsioon, mida kauglugemise teel on võimalik tuvastada, on mõistlik ning privaatne andmekaitse temaatikas.

2.3.1 Töövahendid

Käesolev töö kasutab andmete kogumiseks NAS-i poolt pakutud infosüsteemi Iothub, mille API kaudu laetakse andmed eraldi analüüsimiseks programmi. Programm on olemuselt prototüüp ning see kirjutatakse Python-i [15] tarkvarakeeles käesoleva töö raames. Python osutus valituks tänu autori soovi ning Python-i võimalustele andmeteaduse valdkonnas.

Prototüübi arendamiseks kasutatakse peamiselt järgnevaid kooditeeke: *pandas*[16], *numpy*[17], *statsmodels*[18], *matplotlib*[19] ning *scikit-learn*[20]. Kõik teegid on ka andmeteaduse valdkonnas populaarsed[21]. Python-i koodi arendatakse Jupyter-i [22] arenduskeskkonnas, mis on mugav andmeteaduse informatsiooni käitlemises, sest tegemist on interaktiivseid väljundeid lubava arendusvahendiga.

3 Andmete analüüs

Käesolev töö kogus andmeid NAS-i poolt üles seatud test-seadmetelt, mis seati üles külma vee näidikutele reaalsete testklientide koju. Nende kohta on teada järgnev informatsioon.

Tabel 2 Seadmete informatsioon

Anduri kood	**Seadme nimi**	**Täiskasvanud**	**Lapsed**	**Saatmise sagedus**	**Saatmise alguse kuupäev**
WMR - 35100025	A	2	2	5 minutit	17.02.2017
WMR - 3510003A	B	1	2	1 tund	31.03.2017
WMR - 35100076	C	Informatsiooni pole teada		1 tund	31.03.2017

Tabel 2 põhjal võib järeldada, et käesoleva töös on kõige olulisem seade A, mis on kõige kauem andmeid kogunud ning mille saatmise sagedus on kõige suurem. Kahel seadmel on erinev saatmise sagedus, andes võimaluse saavutada võrdlusmomenti erinevate kodude vahel.

Igal anduril oma unikaalne kood, mille abil seadme poolt saadetud informatsioon kätte saada. Käesolevas töös antud identifikaatori väärtusi ei avaldata, et tagada andmete tööjärgne turvalisus. Sama kehtib ka kasutaja identifikaatori koodi kohta, millega autenditakse vastu Iothub-i API-t, sest see on isikliku kasutajaga seotud informatsioon.

Käesolev töö talletab Iothub-i serverist pärinevad andmed esmalt enda rakenduse sisu hulka JSON failina, et tagada mugav arendamine andmetöötluse ning -analüüsi valdkonnas.

3.1 Andmete laadimine ja kogumine

Alustame andmete laadimisest Iothub-i serverist ning toome näite kasutades seadme A poolt serverisse saadetud andmeid. Laeme sealt esimesed kaks saadetud hulka.

```
jsonurl = urlopen(get_nas_url(response_count=2))
response = json.loads(jsonurl.read())
print(json.dumps(response, indent=4, sort_keys=True))
```

Meetod get_nas_url() tagastab päringu URL-i tekstiväärtuse, milles on seadme identifikaator ning kasutaja identifikaator olemas.

Väljakuvatud tulemus:


```
{
    "result": [
         {
            "adr": "true",
            "bandwidth": "0",
            "codeRate": "",
            "data": "AAcAAAAVlJYA",
            "date": "2017-02-17 18:41:09",
            "fcnt": "2",
            "fport": "24",
            "frequency": "0",
            "gatewaycount": "1",
            "id": "885805",
            "modulation": null,
            "rssi": "-103",
            "spreadFactor": "0",
            "timestamp": "2147483647"
        },
        {
            "adr": "true",
            "bandwidth": "0",
            "codeRate": "",
            "data": "LAkAAAATAJYA",
            "date": "2017-02-17 18:46:11",
            "fcnt": "3",
            "fport": "24",
            "frequency": "0",
            "gatewaycount": "1",
            "id": "885828",
            "modulation": null,
            "rssi": "-98",
            "spreadFactor": "0",
            "timestamp": "2147483647"
        }
    ],
    "total": "12806"
```


Nagu varasemalt mainitud, on antud töö raames olulised *data, fport ning date* väljad. Käesoleva töö raames kasutatavad veesensorid saadavad veenäidu infot *fport* 24 väärtusega, kuid mõned testimiseks seatud sensorid saadavad lisaks ka *fport* väärtusel 6

informatsiooni ning neid andmeid käesolev töö ei vaja. Seega tuleb antud informatsioon eemaldada. Lisaks on vaja muuta välja *data* base64 väärtus konkreetseks veenäiduks antud momendil. NAS-i veesensor saadab veenäitu üheksa baidise jadana, kuhu esimesse nelja baiti, mis on LSB baidijärjekorra formaadis (Tabel 3), on peidetud kogutarbimise kumuleeritud impulsside hulk täisarvulise väärtusena.

Tabel 3 NAS-i veesensori staatuse sõnum

Bait 1	**Bait 2**	**Bait 3**	**Bait 4**	Bait 5	Bait 6	Bait 7	Bait 8	Bait 9
Antud hetke kokku loendatud impulsside hulk (uint32)				Aku seis (int8)	Temperatuur (int8)	RSSI (int8)	Sensori staatus	

Käesolev töö teisendab sõnumis oleva väärtuse liitriteks ning eemaldab müra tekitavad andmed, ehk keskendub veenäidule väljal *data* ning kuupäevale ning kellaajale väljal *date*.

```
from urllib.request import urlopen
from mainlib import file_name
import json
import base64


def water_meter_data_parser(nas_data_str):
    btea_orig = bytearray(nas_data_str, 'UTF-8')
    btea = base64.b64decode(btea_orig)
    return int.from_bytes(btea[:4], byteorder='little', signed=False)

def map_nas_data(row):
    data_int = float(water_meter_data_parser(row['data']))
    return {
        'data': data_int / 100, # to l
        'date': row['date']
    }

usage_data_unparsed = response['result']
usage_data_list = list(map(map_nas_data,
                           filter(lambda row: row['fport'] == '24',
                           usage_data_unparsed)))

print(json.dumps(usage_data_list, indent=4, sort_keys=True))
```

Andmete dekodeerimiseks on defineeritud kaks meetodit. Meetod *water_meter_data_parser* dekodeerib üheksa baidise base64 tekstiväärtuse nelja

baidiseks täisarvuliseks väärtuseks ning *map_nas_data* muundab JSON-ist tulnud andmed reaalseks numbriliseks veenäiduks.

Tulemus:

```
[
    {
        "data": 17.92,
        "date": "2017-02-17 18:41:09"
    },
    {
        "data": 23.48,
        "date": "2017-02-17 18:46:11"
    }
]
```

Sarnast mustrit korrati iga seadme korral ning nende tulemused kirjutati eraldi JSON failidesse.

```
file = open(file_name, "w")
file.write(json.dumps(usage_data_list, indent=4, separators=(',', ': ')))
file.flush()
file.close()
```

3.2 Analüüs keskmiste väärtuste põhjal

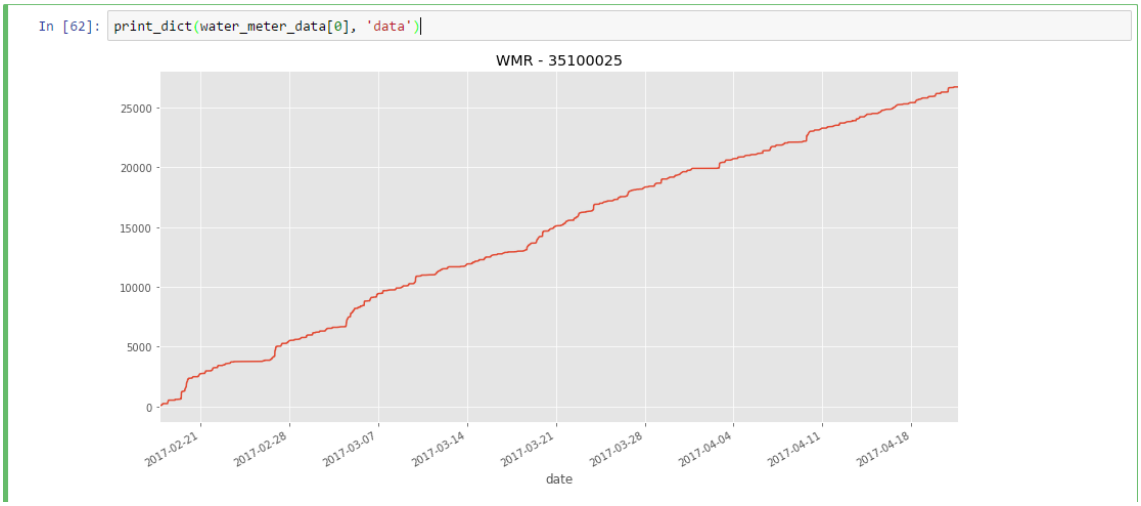
Uurime, mis informatsiooni andmetest võib leida, kasutades andmete filtreerimist, keskmistamist, summeerimist esialgsete andmete pealt. Püüame võrrelda kõiki katseobjekte omavahel ning tuvastada esmaseid sündmusi kõigist neist ning võrdleme tulemusi.

Käesoleva töö edasist koodi on arendatud Jupyter-i arendusvahendis. Kuna graafikute ning andmete hulk on mahukas, toome edasised näited seadme A veesensori andmete kohta.

Esmalt kuvame algsed andmed iga seadme kohta välja, selleks laeme faili kirjutatud andmed *pandas* teegi poolt pakutud andmeraami, mis pakub mitmeid mugavaid võimalusi andmete analüüsiks.

```
from matplotlib.pylab import rcParams
import matplotlib
import pandas as pd
import matplotlib.pylab as plt

matplotlib.style.use('ggplot')
%matplotlib inline
rcParams['figure.figsize'] = 14, 7

water_meter_data = [
    {'file_name': 'data/a.json', 'label': 'WMR - 35100025'},
    {'file_name': 'data/b.json', 'label': 'WMR - 3510003A'},
    {'file_name': 'data/c.json', 'label': 'WMR - 35100076'}
]

def print_dict(data_arr, elem='data', kind='line'):
    data_arr['data'][elem].plot(title=data_arr['label'], kind=kind)
    plt.show()


for wm in water_meter_data:
    data_df = pd.read_json(wm['file_name'], orient='records')
    wm['data'] = pd.DataFrame.from_records(data_df, index='date',
columns=['data', 'date'])
```

Seejärel kuvame Jupyter-i Notebook-is välja graafiku.



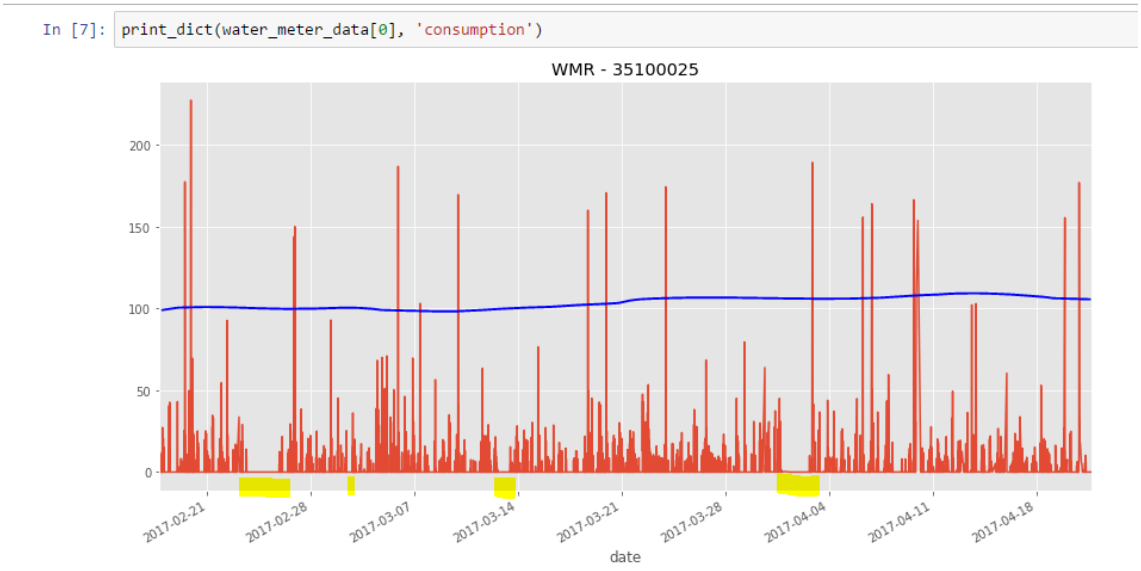
Joonis 3 Seadme A algsete andmete graafik Jupyter-i töökeskkonnas

Joonis 3 näitab pere kumulatiivset veekasutust 17-ndast veebruarist kuni aprilli viimase nädalani. Esialgsetest andmetest on visuaalselt keeruline muud informatsiooni tuvastada peale veenäidu. Antud pere on pigem stabiilse tarbimisega, sest graafik sarnaneb lineaarse

sirgega. Andmetest visuaalselt suurema hulga informatsiooni saamiseks, uurime iga saatmisvahemiku vahelist tarbimist.

Selleks lahutame iga eelneva näidu järgnevast ning kuvame tulemuse graafikuna.

```
for wm in water_meter_data:
    wm['data']['consumption'] = wm['data']['data'] - wm['data']['data'].shift()
    wm['data']['consumption'] = wm['data']['consumption'].fillna(0)
```



Joonis 4 Seadme A tarbimise graafik

Antud graafik (Joonis 4) pakub palju rohkem informatsiooni võrreldes eelnevaga. Esmalt saab tuvastada, et pere kõige sagedasemad tarbimised oma ajavahemiku vältel on alla 50 liitri, kus on enam kui kord pikemaid perioode, mille hulgas tarbimine on 0 liitri lähedal (graafikul märgitud kollasega). Samuti leiab graafikult mitmeid väärtuseid, mis ületavad 100 liitri piiri (graafikul märgitud sinisega).

Antud näidet saab vaadelda kui visuaalsel teel tuvastatud sündmuseid. Kui tarbimine on pikema perioodi vältel olnud ligi null, võib väita, et antud perioodil pere kodus ei olnud. Näidud, mis on ületanud 100 liitri piiri, võib vaadelda kui pere pesemiskäike, kuid see on pigem oletuslik.

Jätkame andmete analüüsi ajaliste määratluste alusel. Selleks teisendame veeru *date*, mis on kuupäeva veerg, erinevateks ajalisteks osadeks.

```
for wm in water_meter_data:
    data = wm['data']
    data['year'] = data.index.year
    data['month'] = data.index.month
    data['day'] = data.index.day
    data['hour'] = data.index.hour
    data['minute'] = data.index.minute
    data['weekday_nb'] = data.index.dayofweek
    data['weekday'] = data.index.weekday_name
    wm['data'] = data

water_meter_data[0]['data']
```

Kuvame koostatud objekti välja (Tabel 4):

Tabel 4 Ajalisteks osadeks jaotatud tarbimisinformatsiooni tulemus

	data	consumption	year	month	day	hour	minute	weekday_nb	weekday
date									
2017-02-17 18:41:09	17.92	0.00	2017	2	17	18	41	4	Friday
2017-02-17 18:46:11	23.48	5.56	2017	2	17	18	46	4	Friday
2017-02-17 18:51:13	23.48	0.00	2017	2	17	18	51	4	Friday
2017-02-17 18:56:15	23.48	0.00	2017	2	17	18	56	4	Friday
2017-02-17 19:01:17	23.48	0.00	2017	2	17	19	1	4	Friday

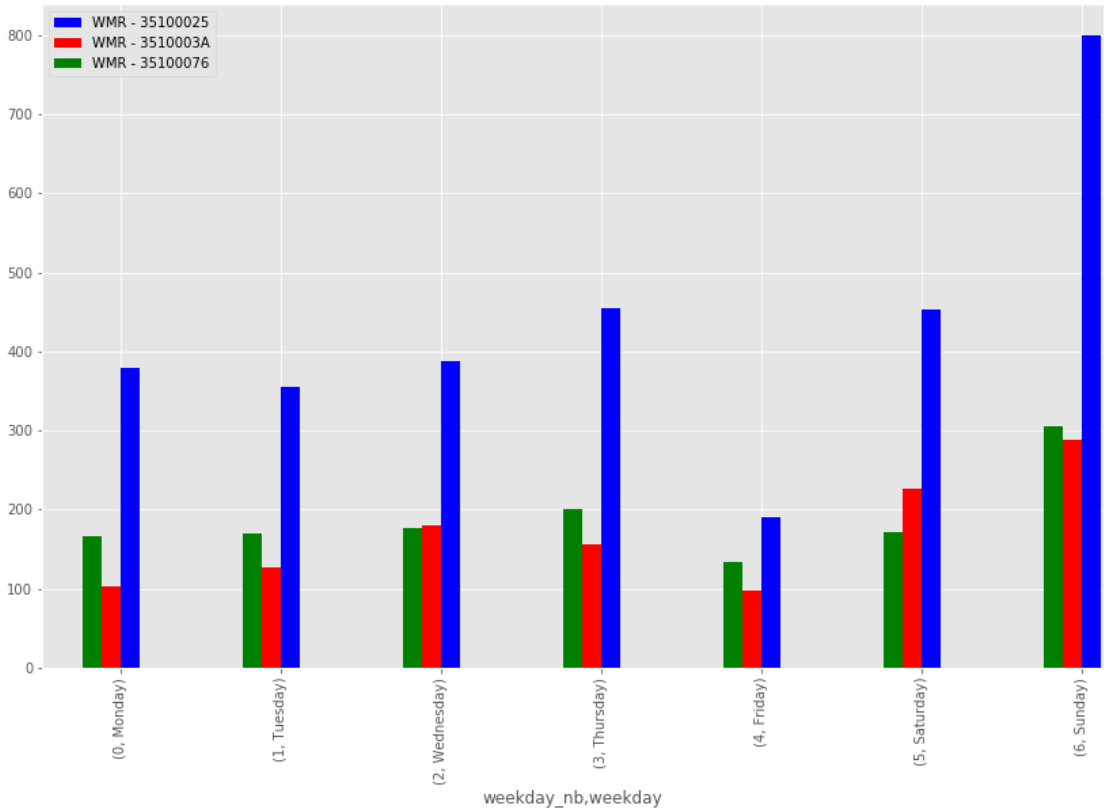
Osadeks jagamine on vajalik, et muuta antud tabelile erinevate päringute tegemist mugavamaks, nagu on grupeerimine, summeerimine ja keskmistamine. Järgnevalt analüüsime kõigi kolme seadme poolt salvestatud tarbimise informatsiooni.

3.2.1 Päevade kaupa võrdlus

Uurime tarbimist nädalapäevade kaupa. Näitame välja iga üksiku nädalapäeva kogutarbimise summa keskmist kogu tarbimisajaloo vältel. Selleks grupeerime esmalt iga kuu, iga päeva ning iga nädalapäeva kaupa andmed ning summeerime iga grupi ajavahemiku tarbimised kokku (väli *consumption*) ning võtame iga summeeritud väärtuse iga erineva nädalapäeva kohta keskmise. Lisaks võtame arvestusse ainult need päevad, kus tarbimine on suurem kui üks liiter, et tagada andmete parem täpsus, ning et arvestada neid päevi kui keegi on reaalselt kodus.

```
def plot_bar(label, kind='bar'):
    ax = water_meter_data[0][label].plot(kind=kind, color='blue',  position=0, width=0.12)
    ax2 = water_meter_data[1][label].plot(ax=ax, kind=kind, color='red', position=1, width=0.12)
    water_meter_data[2][label].plot(ax=ax2, kind=kind, color='green', position=2, width=0.12)
    ax.legend(labels=map(lambda wm: wm['label'] , water_meter_data))

for wm in water_meter_data:
    df = wm['data']
    df_by_weekday = df.groupby(['month', 'day', 'weekday_nb', 'weekday'], as_index=False)
    df_by_weekday = df_by_weekday['consumption'].sum()
    df_by_weekday = df_by_weekday[df_by_weekday['consumption']  >  1]
    df_by_weekday = df_by_weekday.groupby(['weekday_nb', 'weekday'])['consumption'].mean()
    wm['df_by_weekday'] = df_by_weekday

plot_bar('df_by_weekday')
```

Tulemuseks saame tulpdiagrammi, kus on kõigi kolme katsesensori tulemused antud kriteeriumitel.



Joonis 5 Keskmine tarbimine iga nädalapäeva kohta

Jooniselt 6 on ilmne järeldada, et seadme A (sinine) kodus on tarbimine oluliselt suurem kui ülejäänutel, kuid kõigi objektide vahel on lisaks ka sarnasusi. On huvitav, et kõigi kolme puhul on reedene tarbimine kõige madalam võrreldes ülejäänud päevadega ning

ka tööpäevadega. Lisaks on sarnane kõigil pühapäevane päev, mis on võrreldes teiste nädalapäevadega kõige kõrgema tarbimisega. See annab võimaluse järeldada, esmaseid mustreid, kus pühapäeviti, võrreldes ülejäänud nädalapäevadega, veedetakse kõige enam summeeritud perioode ehk aega kodus.

Antud informatsiooni alusel on võimalik järeldada, et päevapõhistest andmetest on võimalik järeldada päevi, millal pere kodus ei olnud. Töö raames uuriti päevi, kus päeva kogu tarbimiste summa on alla ühe liitri. Liiter valiti autori poolt, et arvestada võimalike pisikeste leketega). Käesoleva töö raames selgus, et ainult seadme A perel oli päevi, millal kedagi kodus ei olnud ning neid oli kokku kaks (24. veebruar ning 1. aprill). Lisades kitsendusse aktiivse päevase ning õhtuse perioodi (autor valis kella 09.00 ning 22.00 vahelise perioodi), tuvastati lisaks seadme A osas veel üks päev (21. aprill) ning seadme C osas üks päev (31. märts).

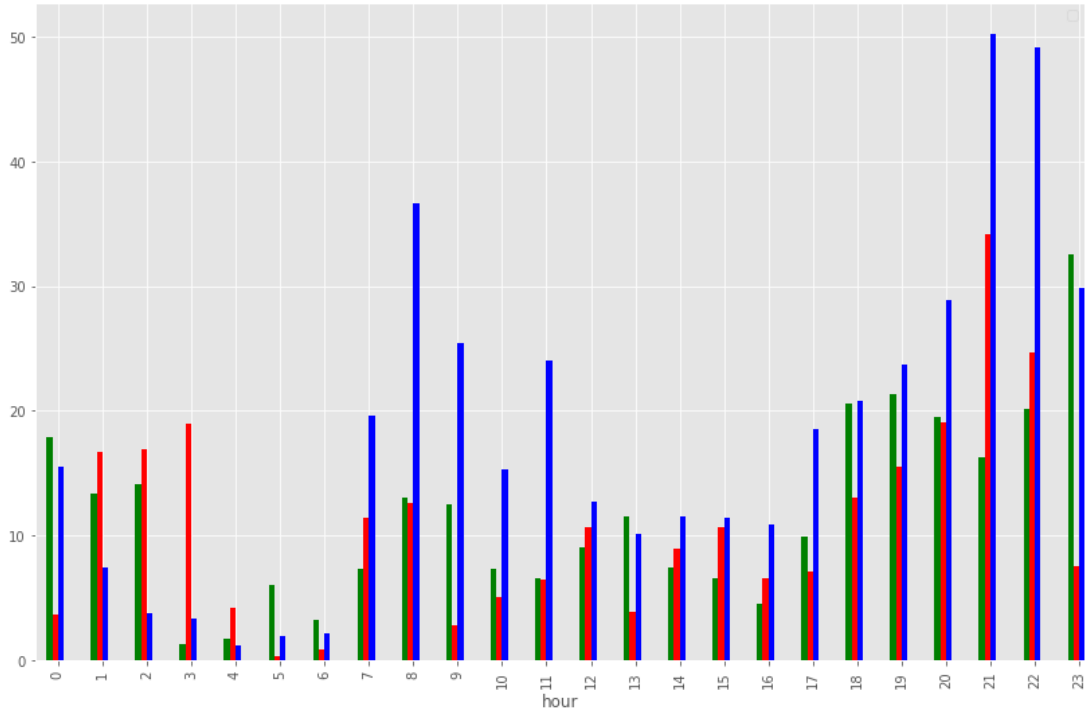
3.2.2 Tundide kaupa võrdlus

Liigume andmete analüüsi osas täpsemaks ning uurime neid tunnipõhiselt. Selleks genereerime objektide tunnipõhise andmestiku kõikide päevade kohta. Seejärel keskmistame tulemused, kuid kuna võrdlus on aja määratluselt täpsem, siis arvestusse võetakse kõik tarbimised.

```
def plot_bar_single(wm, label, color, kind='bar'):
    ax = wm[label].plot(kind=kind, color=color, title=wm['label'])
    ax.legend(labels=[wm['label']])

def manage_hourly_data(result_label, map_df=lambda x: x):
    for wm in water_meter_data:
        df = wm['data']
        df = map_df(df)
        df_by_hour = df.groupby(['month', 'day', 'hour'], as_index=False)
        df_by_hour = df_by_hour['consumption'].sum()
        df_by_hour = df_by_hour.groupby(['hour'])['consumption'].mean()
        wm[result_label] = df_by_hour

manage_hourly_data('df_by_hour')
plot_bar('df_by_hour')
```

Defineerisime meetodi *manage_hourly_data*, mille abil andmeid muundada ja kuvada. Meetodi teine parameeter on meetod, mille abil on võimalik andmeid erinevalt filtreerida.

Lisaks lõime meetodi *plot_bar_single*, mille abil saame üksikuid tulpdiagramme Jupyter-i keskkonnas visuaalselt kuvada.

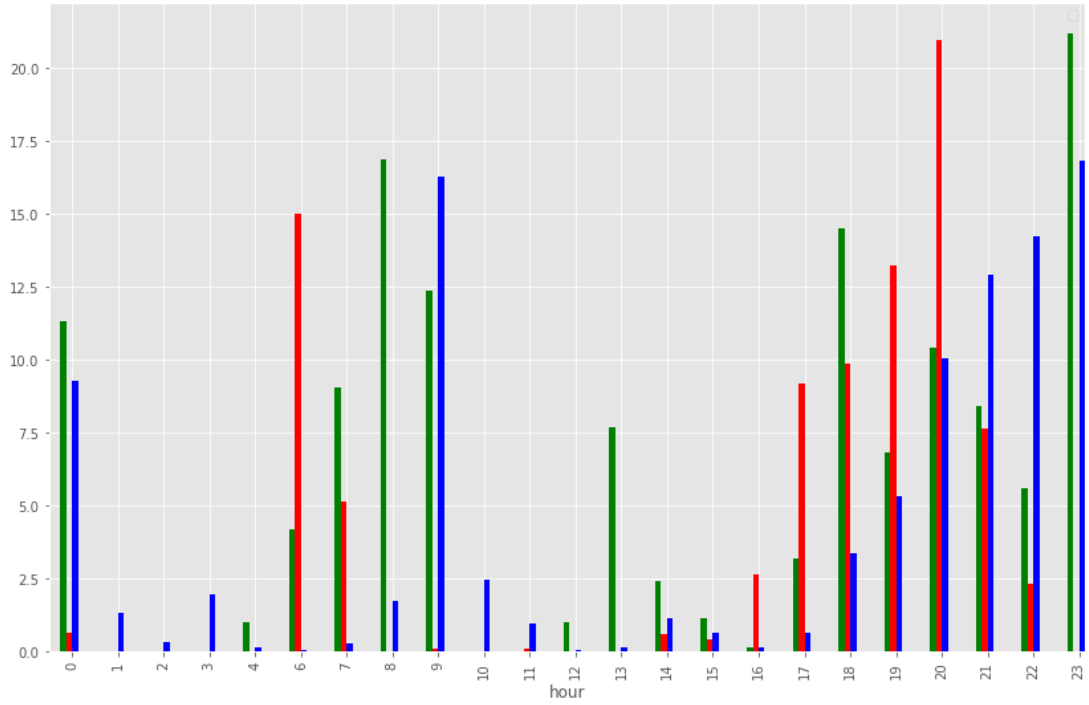


Joonis 6 Tunnipõhiste tarbimiste keskmised

Joonis 6 võimaldab võrrelda katse-sensorite perede erinevusi tarbimisharjumuste perioodide osas ning näitab, et need on pigem erinevad. Kõige sarnasem periood kõigi kolme objekti vahel on kella nelja ning kella kuue vahel hommikul. Püüame leida konkreetseid perioode, millal tüüpiliselt tarbimine eksisteerib ning millal on tarbimine väike. Uurime tarbimisi tööpäevade lõikes, sest võib eeldada, et neil päevadel on perioodid päeva lõikes paremini väljakujunenud.

Selleks, et uurimist alustada, filtreerime esmalt nädalavahetuse päevad andmehulgast välja.

```
manage_hourly_data('df_by_hour_workdays', lambda df: df[df['weekday_nb'] < 5])
plot_bar('df_by_hour')
```



Joonis 7 Katseseadmete tunnipõhiste tarbimiste keskmised tööpäevade lõikes

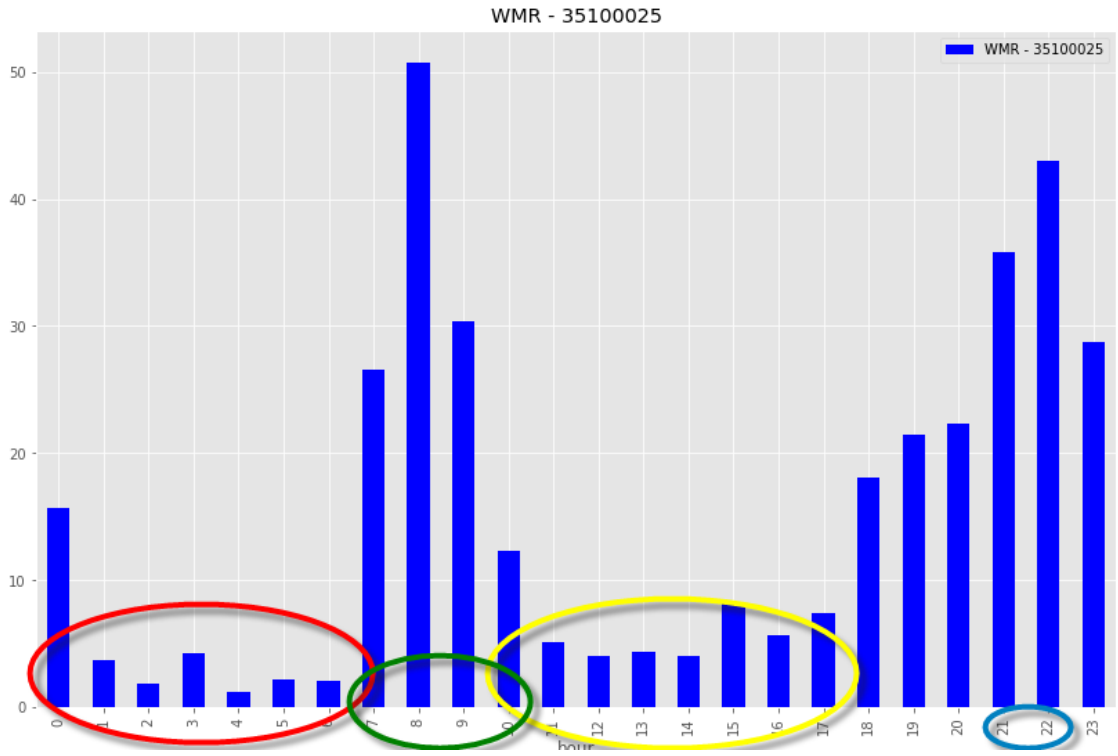
Joonis 7 võimaldab mõista, et perioode kus tarbimine on tundide lõikes madalam kui tüüpiliselt, on võimalik tuvastada ning neid perioode on katseobjektide vahel rohkem kui üks. Uurime kõikide perede tarbimisharjumusi eraldi ning püüame leida kõikide seadmete vahel sarnaseid perioode varasemalt kirjeldatud tingimustel.

3.2.2.1 Seade A

Uurime esmalt seadme A pere tööpäevaseid harjumusi, uurides tööpäeva tundide keskmisi tulemusi.

Kuvame tulemused välja:

```
plot_bar_single(water_meter_data[0], 'df_by_hour_workdays', 'blue')
```



Joonis 8 Seadme A tööpäeva tundide keskmised tarbimised

Antud graafik (Joonis 8) võimaldab tuvastada palju enam. Näiteks on võimalik tuvastada pere minimaalne une pikkus keskmiselt, mis antud juhul on kella kaheteistkümne ning kella kuue vahel ehk umbes seitse tundi (graafikul punasega märgitud).

On võimalik tuvastada pere keskmine ülestõusmise ning kodunt ära minemise periood (märgitud rohelisega), kust võib ka eeldada, et see on pigem pereliikmetel erinev kui sarnane. Antud näite puhul jääb see periood kella seitsme ning kella kümne vahemikku, kus kella kümnene kodunt lahkumise aeg on võrreldes ülejäänud perioodiga tundidega harvem nähtus. Kuna on teada, et peres kasvavad lapsed, võib eeldada, et nende ärkamise aeg on kõige varasem, sest kooli- ning lasteaedade päevad algavad pigem vara ning pärast kella üheksasi aegu on koduses toimetamises aktiivseks pooleks üks või teine või mõlemad täiskasvanud, kuid jällegi on see oletuslik.

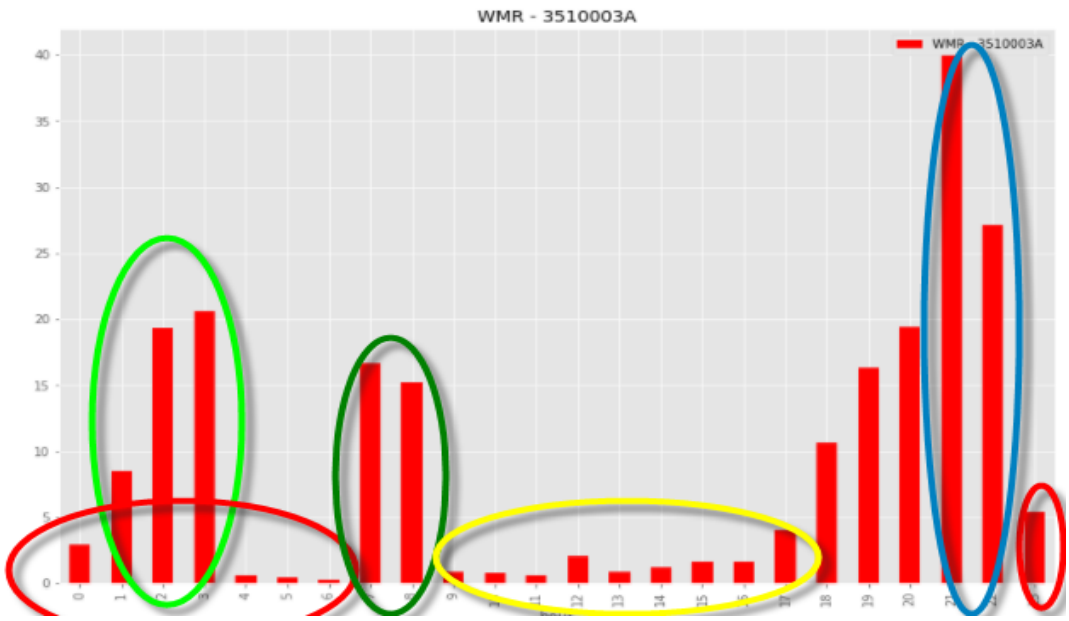
Kella üheteistkümnest kella kuueni on graafikul märgata vähest keskmist tarbimist (graafikul märgitud kollasega). Kuna tegemist on tööpäevadega, võime eeldada, et see on keskmine tööl olemise periood, sest selle perioodi veetarbimine on oluliselt erinev võrreldes õhtuste kella aegadega kui võib eeldada, et pere on kodus.

Graafikul kujutatud ajajoonel on märgata, et päeva lõpupoole on kella üheksa ning kümne ajal keskmine tarbimine märgatavalt kõrgem kui ülejäänud õhtutundidel (märgitud sinisega). Seda perioodi võib klassifitseerida kui pere harjumuslikku pesemas käimise aega. Varasemat, kahe perioodi vahelist aega võib vaadelda kui pere õhtuseid tegevusi, mille hulgas on näiteks söögi tegemine, nõudepesu jne. Antud graafikult ei saa kahjuks tuvastada, et mis sündmused antud juhul toimusid.

3.2.2.2 Seade B

Vaatleme, kas on võimalik sarnaseid perioode leida ka teiste katseobjektide keskmistes tunnipõhistes tarbimistes tööpäevade hulgas.

```
plot_bar_single(water_meter_data[1], 'df_by_hour_workdays', 'green')
```



Joonis 9 Seadme B tööpäeva tundide keskmised tarbimised

Võrreldes eelmise graafikuga, on antud joonis natuke erinev. Sarnaselt varasemaga on hästi eraldatav kodust eemal oleku periood, mis antud on objekti puhul kella kaheksast kella kuueni (graafikul märgitud kollasega). Samuti on leitav hommikuste tegevuste periood, mis on võrreldes eelmisega lühem, kella seitsme ja kaheksa aeg (graafikul märgitud rohelisega) ning ka õhtune pesuperiood (graafikul märgitud sinisega).

Huvitav on vaadelda magamise perioodi. Kui muidu on märgata pärast kella kümmet õhtul, et tarbimine on langenud, siis keset ööd, kella ühe, kahe ja kolme ajal, see järsult tõuseb ning hiljem langeb tagasi pea olematuks, kuniks jõuab hommikune periood. Kuna tegemist on keskmiste tarbimisega, võib eeldada, et öösel sarnast tarbimist juhtub sageli. Arvestades muude valdkondade rütmi, võib eeldada, et tegemist on mingi vett tarbivate masinate tööga (nõudepesumasin, pesumasin), mis on pandud öösel käima, et öise odavama elektri abil säästa, või mingi muu tegevus, mida antud hetkel ei ole võimalik määrata. Uurime lähemalt ning kuvame antud anduri näidud välja tööpäeviti kella ühe ja kolme vahel ning kus tarbimine on suurem kui üks liiter.

```
ch_data = water_meter_data[1]['data']
ch_data = ch_data[ch_data['weekday_nb'] < 5]
ch_data = ch_data[ch_data['hour'] >= 1]
ch_data = ch_data[ch_data['hour'] <= 3]
ch_data = ch_data[ch_data['consumption'] > 1]

ch_data
```

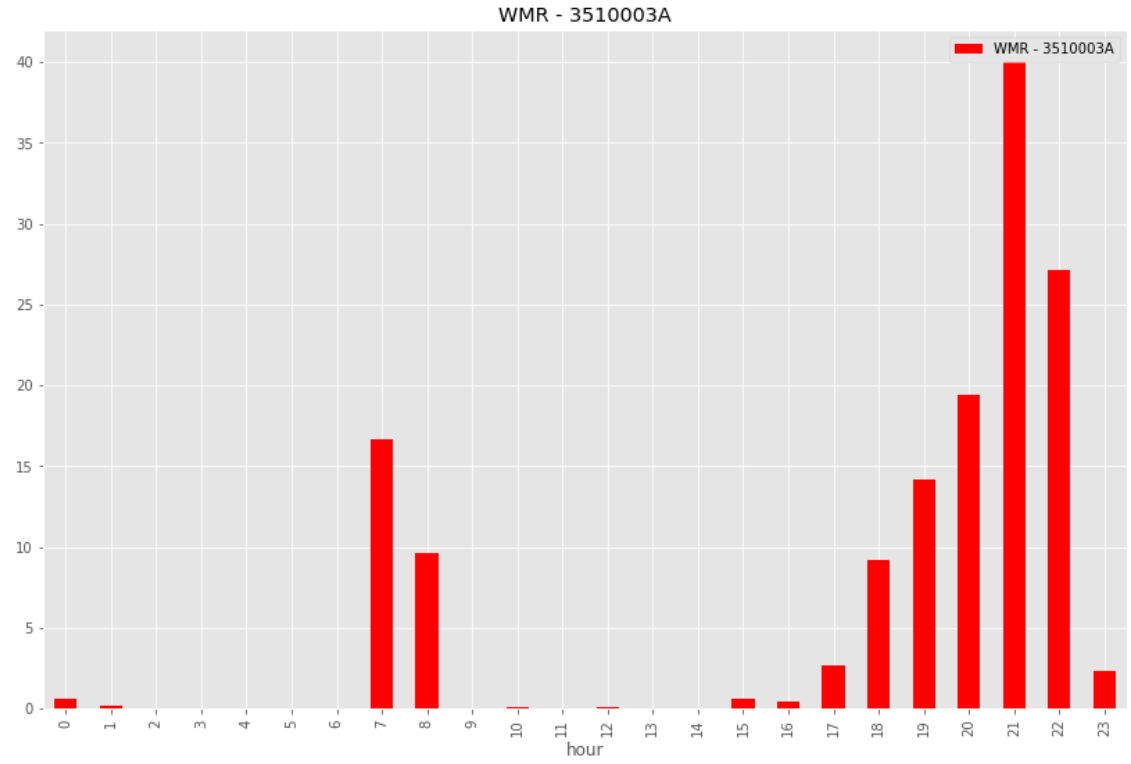
Tabel 5 Seadme B üksikute perioodide kõrgete tarbimiste uurimise tulemus

	data	consumption	year	month	day	hour	minute	weekday_nb	weekday
date									
2017-04-11 01:50:41	1922	50.0	2017	4	11	1	50	1	Tuesday
2017-04-12 03:50:52	2067	103.0	2017	4	12	3	50	2	Wednesday
2017-04-18 02:51:49	2973	67.0	2017	4	18	2	51	1	Tuesday
2017-04-21 02:52:18	3457	68.0	2017	4	21	2	52	4	Friday

Antud tulemus näitab, et tegemist on pigem üksikute kordadega, kus veetarbimine oli hästi suur ning tänu lühikesele ajateljele tõstavad üksikud juhtumid keskmist väärtust oluliselt ning standardhälve on kõrge. Seetõttu antud olukorras välistame need juhtumid. Magamistundide pikkus antud perel on standardselt tööpäeviti kella kümnest õhtul, kella seitsmeni hommikul. See tekitab antud juhul käesoleva töö jaoks probleemi, kus keskmiste väärtuste uurimine nii lühikesel perioodil (nagu on seadme B ning seadme C puhul) ei taga piisava täpsusega keskmisi tulemusi pere tarbimisharjumustest.

Seetõttu täiendame *manage_hourly_data* meetodit ning püüame kõrged ebatavalised keskmist tõstvad olukorrad eemaldada. Arvutame kõikide tööpäevade konkreetsete tundide osas keskmised nagu varem, kuid seekord seda kaks korda. Esimesel korral arutame keskmist tavalisel viisil ning leiame hälbe keskmisest väärtusest. Teistkorda arvutades, arvutame sama andmehulga pealt, mille hulgast oleme eemaldanud väärtused, mille hälve on esialgest keskmisest kaks korda suurem. Antud juhul peaks see tagama lühikese perioodi jooksul tüüpilise tarbimise ilmestamise, mille hulgast on kaotatud juhuslikud keskmist oluliselt tõstvad ekstreemumid.

```
def manage_hourly_data(result_label, map_df=lambda x: x):
    for wm in water_meter_data:
        df = wm['data']
        df = map_df(df)
        df_by_hour = df.groupby(['month', 'day', 'hour'], as_index=False)
        df_by_hour = df_by_hour['consumption'].sum()

        means_by_hours = df_by_hour.groupby(['hour'])['consumption'].mean()

        df_by_hour['mean_val'] = df_by_hour.apply(lambda row:
                                    means_by_hours[row['hour']] , axis=1)

        df_by_hour['deviation']= abs(df_by_hour['mean_val'] –
                                          df_by_hour['consumption'])
        df_by_hour['deviation_times_diff'] = df_by_hour['deviation'] /
                                                 df_by_hour['mean_val']
        df_by_hour = df_by_hour[df_by_hour['deviation_times_diff'] < 2]

        df_by_hour = df_by_hour.groupby(['hour'])['consumption'].mean()
        wm[result_label] = df_by_hour

manage_hourly_data('df_by_hour_workdays', lambda df: df[df['weekday_nb'] < 5])
plot_bar_single(water_meter_data[1], 'df_by_hour_workdays', 'red')
```



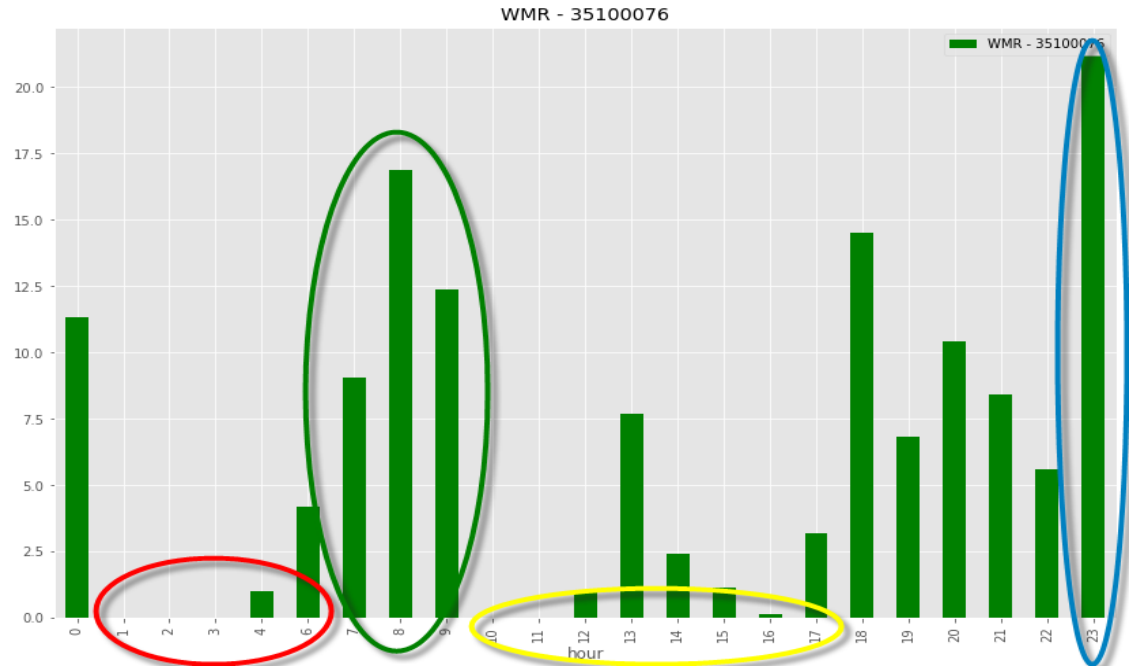
Joonis 10 Kärbitud seadme B graafik

Joonis 10 on une aeg selgelt väljaloetav ning selleks on kella üheteistkümne ning kella seitsme vaheline periood.

3.2.2.3 Seade C

Viimaks uurime tööpäeva lõikes seadme C pere tarbimisharjumusi. Kuna tegemist on samuti lühikese perioodiga, rakendame koheselt sarnast metoodikat, nagu seadme B puhul.

```
plot_bar_single(water_meter_data[2],'df_by_hour_workdays', 'green')
```



Joonis 11 Seadme C tööpäeva tundide keskmised tarbimised

Viimase graafiku pealt (Joonis 11) võib lugeda, et antud pere keskmine une aeg on pärast kesköö ning kella seitsme vaheline periood, millele järgneb hommikune rutiin, mis kestab kella üheksani. Tööaja pikkus on perel kümne ja õhtu kella viie või kuue vahel ning õhtune pesemas käimine on pigem hiline, kella üheteist ajal öösel.

Kuvame analüüsi tulemused tööpäeva lõikes iga katseobjekti vahel tabelina (Tabel 6).

Tabel 6 Analüüsi tulemustest järeldunud tööpäevade perioodid

Objekt	**Une periood (Kokku)**	**Hommikuste tegevuste periood (Kokku)**	**Tööaeg (Kokku)**	**Pesus käik õhtuti (Kokku)**
Seade A	00-6 (6h)	7-10 (4h)	11-17 (7h)	21-22 (2h)
Seade B	23-7 (9h)	7-8 (2h)	9-17 (9h)	22-23 (2h)
Seade C	00-7 (7h)	7-9 (3h)	10-17 (8h)	23 (1h)

4 Sündmustamine

Hüpotees, kas sündmuste leidmine tarbimisvoo ajajoonelt on võimalik, on käesoleva töö kõige tähtsam probleem. Konkreetseid sündmuseid on parem tuvastada, kui ajajoonel on erinevaid väärtusi kuvatud sagedasti. Seega alustame uurimist seadme A poolt edastatud informatsiooniga.

4.1 K-means

Kasutame sündmuste tuvastamiseks *k-means* [23] algoritmi. Valik langes antud meetodile järgnevatel põhjustel:

- Andmeid on vähe, seega keeruka klasterdamise meetodi järele puudub vajadus.
- Ta võimaldab jagada klastritesse ka ühe-dimensionaalsel juhul (näiteks veetarbimine)
- *K-means* on lihtsasti arusaadav ja kuna sündmustamise näited on esmane samm suuremast tööst, võimaldab see aega säästa.
- *K-means* on kiire ning energiatõhus, mis suurendab tõenäosust, et algoritmi kasutada ka anduri tasemel (lähemalt alapeatükis 6.1).

Olemuselt on algoritm klastrite ning klastrite keskpunktide leidmise meetod, mille loogika on järgnev:

1. Määratakse otsitav klastrite arv (ütleme, et see on *k).*
2. Valitakse algandmetest juhuslikud *k* väärtust, mis määratakse iga klastri keskpunktideks ning arvutatakse kõikide arvude kaugus klastri keskmest.
3. Iga klastri hulka määratakse konkreetne arv, millele see on kõige lähem.
4. Seejärel määratakse iga arvu hulgast, mis on klastri keskmele kõige lähedamal omakorda järgmine omavaheline keskpunkt - nii, et summaarne kaugus oleks vähim. Selle põhjal leitakse uued klastrite keskpunktid.
5. Naastakse punkti 2, kuid seekord uute keskmetega.

Python-is on *k-means-i* algoritm implementeeritud mitmetes tuntud teekides. Käesolev töö kasutab *sklearn* teeki kuuluvat objekti *KMeans* [24], mis pärineb *scikit-learn* masinõppe algoritmide kogust. Teek lubab kasutada parameetreid, mille abil saab algoritmi tööd muuta. Käesoleva töö raames on algoritmi valik teegis on *full*, mis tähendab, et teostatakse klassikaline EM-algoritmil [23] *(expectation-maximation)* põhinev *k-means*-i algoritm, kus maksimeerimise samm (ingl. keeles *maximation step*) on varem defineeritud keskväärtuste valimine klastrile ning ootuse samm (ingl. keeles *expectation step*) on nende ümberkalkuleerimine.

Teek soovib andmete klastrite koostamise sisendiks kahedimensioonilist maatriksit. Käesoleva töö andmed on kõik arvulised väärtused ühes jadas, kuid ka see sobib andmete klassifitseerimiseks, eeldusel, et andmed muudetakse ühe-elemendilisteks jadadeks, mis asuvad omakorda suurema jada sees. Tegemist on lihtsalt teegi sisendandmete struktuuri sooviga, mille tõttu sisendandmete muudatus on vajalik. Käesoleva töö raames on see lihtne, sest on võimalik kasutada *numpy* meetodit *reshape* parameetritega -1 ja 1, et jada elemendid muuta omakorda üksikelemendi jadaks. Esimene parameeter -1 määrab, et tuleb luua uus jada teise parameetri (1) sammuga liikmeid. Näide -1 ja 1 parameetrite puhul: [A, B] -> [[A], [B]] ning kui parameetriteks on -1, 2: [A, B] -> [[A, B]].

Loome Python-is *sklearn Kmeans* objekti. Seejärel toome näite, kuidas *kmeans* töötab.

```
from sklearn.cluster import KMeans

def do_kmeans(time_series, n_clusters):
    values = time_series.values
    values_reshape_for_kmeans = values.reshape(-1, 1)
    kmeans_obj = KMeans(n_clusters=n_clusters,random_state=0, algorithm='full')
    kmeans_obj = kmeans_obj.fit(values_reshape_for_kmeans)
    return kmeans_obj
```

Lõime meetodi *do_kmeans*, mille kaheks sisendparameetriks on Python-i teegi *pandas DataFrame* objekt ning kastrite arv. Esmalt muundame *pandas* objekti *numpy* arvujadaks, seejärel kasutame *numpy* meetodit *reshape* sarnaselt nagu eelnevalt kirjeldatud. Seejärel anname tulemuse *sklearn KMeans* objekti andmestikuks, mille abil sarju looma hakatakse ning tagastame loodud objekti. Anname andmed meetodile ning näitame milline on tulemus kolme klastri otsimisel. Selleks küsime loodavalt objektilt kõikide klastrite keskmisi väärtusi.

```
test_arr = [1.2, 2.2, 2.3, 1.1, 5, 6, 3]
test = pd.DataFrame(test_arr, columns=['data'])
kmeans_test = do_kmeans(test['data'], 3)
print(kmeans_test.cluster_centers_)
```

Tulemus:

```
[[ 5.5 ]
 [ 2.5 ]
 [ 1.15]]
```

Antud töö raames on antud veemõõtja andmestikku kasutades iga leitud klaster mingi konkreetne sündmus, mis toimus seadme kahe edastamise perioodi vahel, ehk antud juhul iga viie minutilise perioodi vahemikus. Käesolev töö kasutabki antud viisi sündmuste leidmiseks, kus genereeritud klastritele määratletakse oma nimi ning klassifitseeritakse antud klastrit kui konkreetse sündmuse kogumit. Kui kõik klastrid on leitud, uuritakse andmestik läbi ning määratakse, kuhu ja millisesse klastrisse antud tarbimine kuulub. Näiteks kui on tarbimise suurus 3 liitrit, saab kasutada *KMeans* objekti kuuluvat meetodit *predict*, et ennustada millisese klastrisse see kuulub. Toome näite:

```
print(kmeans_test.predict(3))
>>>[1]
```

Väärtus [1] on leitud klastri indeks jada ühe elemendina, ehk [2.5] keskmise väärtusega klaster.

4.1.1 Klastrite hulga tuvastamine

Probleemiks on klastrite hulga tuvastamine, ehk kuidas tuvastada mitu klastrit, või erinevat sündmust on andmehulgas olemas. Kui tüüpiliselt märgitakse *k-means* algoritmi kasutades *k* väärtus enne klasterdamist [25], siis käesoleva töö puhul on võimalus genereerida mitmete k väärtustega klastrid ära ning nende uurimise kaudu leida tulemus. Genereerime klastrid mitmete *k* väärtustega ning uurime, mis on nende tsentrid.

```
result_arr = []
range_start = 3
range_end = 14

def int_round_str(num):
    return str(int(num))

for k in range(range_start, range_end):
    cpy_df = df.copy()
    cluster = do_kmeans(cpy_df['consumption'], k)
    mean_values = cluster.cluster_centers_.flatten().tolist()
    mean_values = map(int_round_str, mean_values)
    result_arr.append({"k": k, "mean_values": ', '.join(mean_values)})

for value in result_arr:
    print(str(value['k']) + ":\t" + str(value['mean_values']))
```

Tulemus:

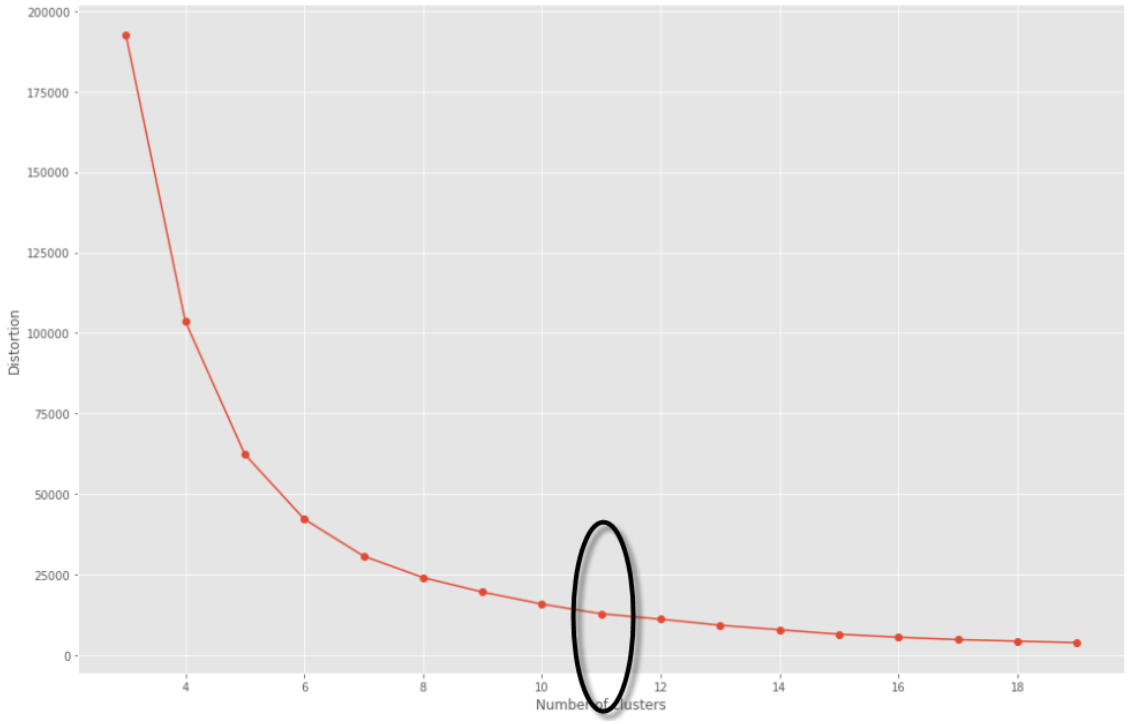
```
3:  0, 139, 24
4:  0, 159, 14, 56
5:  0, 159, 25, 9, 67
6:  0, 97, 49, 21, 167, 7
7:  0, 97, 53, 14, 167, 6, 27
8:  0, 103, 11, 36, 167, 61, 21, 5
9:  0, 167, 46, 17, 4, 104, 28, 69, 10
10: 0, 67, 26, 183, 9, 44, 152, 16, 102, 4
11: 0, 104, 22, 160, 9, 47, 4, 32, 69, 200, 15
12: 0, 174, 37, 16, 71, 9, 4, 149, 51, 102, 25, 227
13: 0, 69, 153, 15, 6, 32, 104, 23, 176, 10, 3, 227, 47
14: 0, 102, 10, 71, 174, 6, 29, 53, 227, 21, 149, 40, 15, 3
```

Tulemuselt saab tuvastada, et vahemikus 9-14 klastrit hakkavad numbrid muutuma sündmuste osas pigem sarnaseks ning vahemikud klastrite vahel lühemaks. Kasutame klastrite hulga tuvastamiseks *elbow* metoodit [26], mis uurib momente, kus andmetele lisa klastri tekitamine ei ole enam optimaalne (ehk klastrid on juba piisavalt sarnased). Selle uurmiseks kuvame kõikide klastrite moonutused visuaalselt ning võrdleme neid. Klastrite moonutuseks loetakse selles olevate andmete kauguste summat nende kesk

väärtustest [24]. Moonutust on võimalik arvutada ka *KMeans* objekti kasutades, kutsudes välja funktsiooni *inertia_*.

```
range_start = 3
range_end = 20

distortions = []
for k in range(range_start, range_end):
    cpy_df = df.copy()
    cluster = do_kmeans(cpy_df['consumption'], k)
    distortions.append(cluster.inertia_)


plt.plot(range(range_start, range_end), distortions, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Distortion')
plt.tight_layout()
plt.show()
```

Tulemus (Joonis 12):



Joonis 12 Klastrite moonutused

Antud meetodi järgi on optimaalne klastrite hulk momendil, kui moonutuste hulga muut on oluliselt väiksem kui antud klastri lisamisel. Sisuliselt näitab see, kohta kus uue klastri lisamine ei tee klastrite seesolevaid väärtusi sarnasemaks. Käesoleva töö antud andmestiku osas näib, et optimaalne klastrite hulk on üksteist (Joonis 12 mustaga

märgitud), sest tõusu langu kiirus on alates nii mitmenda klastri lisamisel pigem aeglane. Kuvame antud tulemuse klastrite keskmised väärtused välja.

```
kmeans = do_kmeans(df['consumption'], 11)

print(list(map(lambda num: int(num),
kmeans.cluster_centers_.flatten().tolist())))
```

Tulemus ümardatud alla täisarvu kujule liitrites:

```
[0, 104, 22, 160, 9, 47, 4, 32, 69, 200, 15]
```

4.1.2 Sündmuste nimetamine

Nagu varasemalt mainitud on tulemuseks klastrite keskmised väärtused. Esmalt leiame igale tarbimisele tema klastri indeksi väärtuse.

```
cluster_idx_fn = lambda row: kmeans.predict(row['consumption'])[0]
water_meter_data[0]['data']['cluster_idx'] =
water_meter_data[0]['data'].apply(cluster_idx_fn, axis=1)
water_meter_data[0]['data'].head(10)
```

Tulemus esimese kümne väärtuse kohta, kus nüüdseks klastri indeks märgitud (Tabel 7):

Tabel 7 Tulemus Jupyter-ist pärast klastri indeksi arvutamist

	data	consumption	year	month	day	hour	minute	weekday_nb	weekday	cluster_idx
date										
2017-02-17 18:41:09	17.92	0.00	2017	2	17	18	41	4	Friday	0
2017-02-17 18:46:11	23.48	5.56	2017	2	17	18	46	4	Friday	6
2017-02-17 18:51:13	23.48	0.00	2017	2	17	18	51	4	Friday	0
2017-02-17 18:56:15	23.48	0.00	2017	2	17	18	56	4	Friday	0
2017-02-17 19:01:17	23.48	0.00	2017	2	17	19	1	4	Friday	0
2017-02-17 19:06:20	35.83	12.35	2017	2	17	19	6	4	Friday	10
2017-02-17 19:11:22	35.84	0.01	2017	2	17	19	11	4	Friday	0
2017-02-17 19:16:24	40.93	5.09	2017	2	17	19	16	4	Friday	6
2017-02-17 19:21:26	40.93	0.00	2017	2	17	19	21	4	Friday	0
2017-02-17 19:26:28	50.40	9.47	2017	2	17	19	26	4	Friday	4

Järgnevalt tuleb defineerida, missugune indeks, millise sündmuse kohta käib. Kuna katseobjektide kohta ei ole informatsiooni, mis veeseadmed neil kodus on, püüame tarbimisi oletada ning uurime kui suur on veetarbimine keskmiselt mingite seadmete kohta.

Käesoleva töö autor viis läbi katsed viie indiviidi vahel, kelle abil püüti tuvastada nende tarbimisi kuni viie minuti jooksul. Koondati informatsiooni peamiselt kolme tegevuse

kohta: WC, pesemine(vann, dušš) ning koondati hambapesu, kätepesu, lihtsad söögi jaoks vee tarbimised mõiste kraanikausi kasutus alla, sest kõik antud tarbimise suurused on pigem sarnased. Valiti kogutud informatsiooni kohta maksimum ning miinimum väärtused ning leiti neile käesoleva töö käigus leitud klastrite indeksid.

Tabel 8 Koostatud tarbimise sündmused

Tegevus	Vahemik katsetest	Valitud indeks	Klastri indeksi keskmine väärtus	Kood
Vann	150-200l	3; 9	160l; 200l	BATH_SHOWER
Dušš	35-120l	1; 5; 7; 8	104l; 47l; 32l; 69l	BATH_SHOWER
Kraanikausi kasutus	2-5l	6	4l	SINK_USE
WC	6-10l	4; 10	9l; 15l	WC
Tegevust ei olnud antud ajavahemikul	~0l	0	0	NO_USAGE

Antud tabelis (
Tabel 8) on kirjeldatud autori arvamus, mis antud tarbimiste puhul võiksid olla tegevused, määrame nende tegevuste koodid kõikidele tarbimiste juurde. Nagu tabel kirjeldab liitsime tegevused vannis käimise ning dušši abil pesemise kokku. Selleks loome käsitsi *dictionary* tüüpi objekti, milles määrame iga indeksi kohta koodi ning lisame koodi tarbimiste hulka lisa veeruks. Käesoleva töö raames jäid ka osad klastrid üle, mida ei osatud autori poolt vastavalt küsimustikule määratleda.

```
event_dict={0: 'NO_USAGE',
            1: 'BATH_SHOWER',
            3: 'BATH_SHOWER',
            8: 'BATH_SHOWER',
            9: 'BATH_SHOWER',
            6: 'SINK_USE',
            4: 'WC',
            10: 'WC'}

def cluster_name_fn(row):
    if row['cluster_idx'] in event_dict:
        return event_dict[row['cluster_idx']]
    return 'ACTIVITY_NOT_SPECIFIED'


water_meter_data[0]['data']['activity'] = water_meter_data[0]['data'].apply(cluster_name_fn, axis=1)
water_meter_data[0]['data'].head(4)
```

Tulemus (Tabel 9):

Tabel 9 Tarbimiste tegevuste lisamine

	data	consumption	year	month	day	hour	minute	weekday_nb	weekday	cluster_idx	activity
date											
2017-02-17 18:41:09	17.92	0.00	2017	2	17	18	41	4	Friday	0	NO_USAGE
2017-02-17 18:46:11	23.48	5.56	2017	2	17	18	46	4	Friday	6	SINK_USE
2017-02-17 18:51:13	23.48	0.00	2017	2	17	18	51	4	Friday	0	NO_USAGE
2017-02-17 18:56:15	23.48	0.00	2017	2	17	18	56	4	Friday	0	NO_USAGE

4.1.3 Valideerimine

Nagu varasemalt mainitud, on antud sündmustamise tulemused autori poolt oletuslikud. Käesoleva töö raames ei ole kahjuks võimalus valideerida, kas objektidel esinenud tarbimine sarnastele sündmustele vastab, sest andurite perede poole pöördumine selle informatsiooni küsimiseks ei ole NAS-i poolt antud ligipääsu, et tagada katseperede anonüümsus. See on ka NAS-i poolt õige samm, kuid seda käsitletakse peatükis 6 Turvalisus.

Käesoleva töö raames see ei ole oluline, sest keskenduda saab antud andmete sündmustamise osas ka teistele eesmärkidele. Kui piltlikult kujutada, et tegemist on vee teenuseid pakkuva ettevõttega, siis ka neil on kindla ajavahemiku järgi mõõdetud mõõtetulemused ning ka nemad saavad antud informatsiooni hulgast teha oletuslikke järeldusi perede käitumise kohta. Seega on käesolev töö justkui uuring vee teenuseid pakkuvate ettevõtete jaoks, et mis andmeid nad tuvastada saavad, ning mis ajavahemik

tuleks valida andmete edastamiseks, et ei seataks ohtu inimeste privaatsusele. Seega järgnevalt püüame kujutada statistikat leitud sündmuste baasil.

Antud peatükk võimaldas meil kujutada, kuidas andmeid klastritesse jagada vastavalt nende suuruste sarnasusele. Püüdsime klastritest leida tuvastatavad sündmused, mis on realistlikud viie minutilisele tarbimisaknale. Antud ülesande järeldus, et viie minuti jooksul on võimalik sündmusi tuvastada, mille kestvus on sarnane antud ajavahemikuga. Olemuslikult on palju mõistlikum ehitada sündmuste tuvastamine iga minuti järgsele tarbimise hulga edastusele, kus sündmuste tuvastus on palju täpsem. Kui uurida sündmuseid kasutades ülejäänud käesoleva töö katseobjektide andmeid, saame tulemuse, mis on sarnane peatükis 3 Andmete analüüs leitud tulemustele, ehk andmetest on võimalik eristada perioode ning eeldada, mis ehk on antud perioodi tegevus, kuid veetarbimist põhjendavaid sündmuseid on pigem raske uurida. Käesoleva peatüki alguses kirjeldatud hüpoteesile, kas sündmuste leidmine tarbimisvoo ajajoonelt on võimalik, on autor seisukohal, et kindlasti on, kuid mida tihedamalt on tarbimisvoogu kirjeldatud, seda parem on tuvastada. Käesoleva töö autor leiab, et vahemik, millal sündmusi saab tuvastada parimal mõistlikul viisil on pigem alla viie minuti.

5 Mustrid

Jätkame käesoleva töö peatükis 4 tuvastatud sündmustega ning püüame kirjeldada visuaalselt pere käitumist vastavalt leitud andmetele. Kasutame lisaks sündmustele ära peatükis 3 Andmete analüüs leitud tulemuste andmeid ning uurime tuvastatud mustrite paiknevust ajajoonel erinevate perioodide lõikes. Jätkame uurimist seadme A poolt edastatud informatsiooniga.

5.1 Sündmuste hulkade leidmine perioodides

Esmalt loome meetodi, mis võimaldab tuvastada konkreetsete sündmuste keskmist hulka ajateljel päevade lõikes. Uurime konkreetseid sündmusi, kui toimus mingi tegevus, ehk välistame read, kus tarbimine oli nulli lähedane.

```
df = water_meter_data[0]['data']
events = ['BATH_SHOWER', 'SINK_USE', 'WC']

def find_avg_event_count(df, period=None, map_df=lambda x, y: x):
    df = df[df['activity'].isin(events)].copy()
    df = map_df(df, period)
    df_by_hour = df.groupby(['month', 'day', 'activity']).size()
    df_by_hour = df_by_hour.reset_index(name='size')
    df_by_hour = df_by_hour.groupby('activity')['size'].mean().reset_index(name='mean')
    df_by_hour['mean'] = df_by_hour.apply(lambda row: ceil(row['mean']), axis=1)
    return df_by_hour

res = find_avg_event_count(df)
res
```

Tulemus:

Tabel 10 Sündmuste keskmine hulk päevas

	ACTIVITY	**MEAN**
1	BATH_SHOWER	4
2	SINK_USE	13
3	WC	13

Tabel 10 kirjeldab kõikide seadme A poolt mõõdetud sündmuse kirjega tarbimiste seast erinevate sündmuste päevas esinemiste kordade keskmisi, ehk vastab küsimusele kui mitu korda esineb mingi sündmus päevas kõikide eetris olnud päevade lõikes keskmiselt. Pesemas käike on pere (kus elab neli inimest) omanud keskmiselt neli korda päevas, kraanikausi kasutusi ning tualetis käike on mõlemaid keskmiselt kolmteist. Meetod (*find_avg_event_count*), mis andmed antud kujule viis, sisendiks on kolm parameetrit, millest üks on *pandas* ajaraam ning teiseks on periood, millal keskmisi sündmuste hulka ennustada ning kolmandaks filtreerimise meetod, mida kasutatakse edaspidi samuti filtreerimisteks.

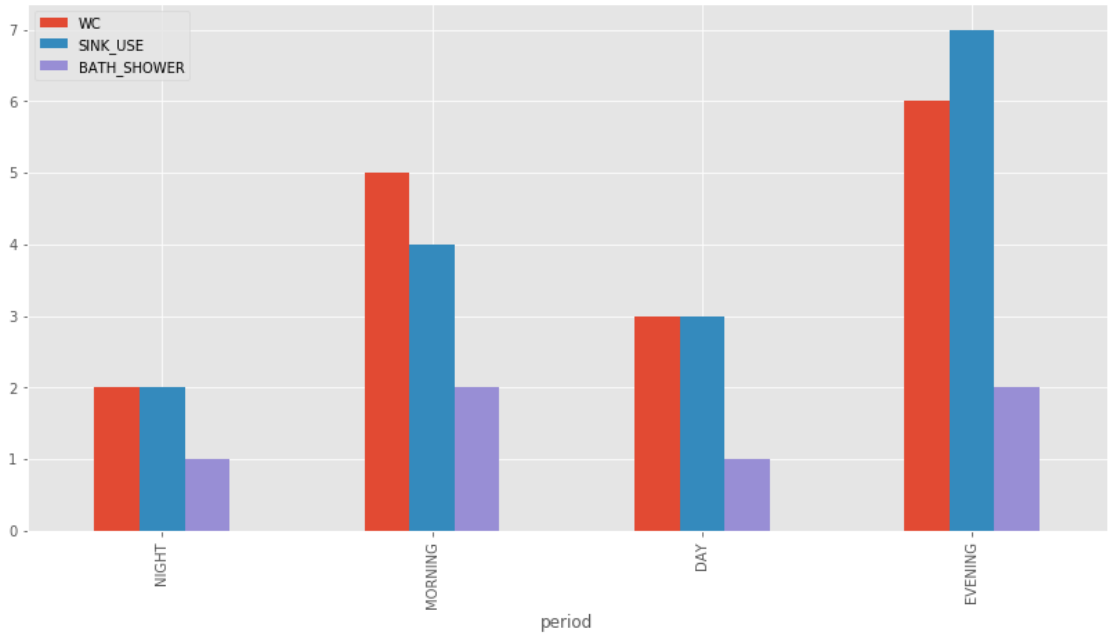
Käesolev töö püüab leida sarnaseid keskmisi väärtuseid nagu kirjeldab Tabel 10 erinevate päeva perioodide kohta, võttes peatükis 3 Andmete analüüs leitud tulemusi, et kirjeldada antud pere tüüpilisi käitumismustreid tööpäevade lõikes. Koostame struktuuri, mille abil saame käitumismustrit keskmise tarbimise osas visualiseerida.

Esmalt valime perioodid vastavalt peatükk 3 Andmete analüüs tulemustelt. Antud informatsiooni põhjal saame perioodid järgmiselt: Uneperiood, hommikused tegevused, tööaeg, pesus käik. Iga antud perioodi kohta on teada tundidel põhinev ajavahemik, millal mingi periood algas ning millal lõppes. Lisaks saame loogiliselt järeldada mõne perioodi kohta konkreetse tegevuse, ehk nimetame une perioodi ööks ning teame, et antud perioodi jooksul esineb sündmus magamine. Nimetame tööaeg tööks ning antud perioodi sündmus on töötamine. Hommikuse tegevuse nimetame hommikuks ning kogu ülejäänud aja nimetame õhtuks, kuigi on teada, et antud perioodil on defineeritud ka tüüpiline pesemas käimise periood, kuid selle perioodi sündmused peaksid ka sündmustega esile kerkima. Nii on semantilised mõisted päeva kirjeldamiseks tuntumad ning perioodid tunni täpsusega teada. Loome perioodid Python-is vastavalt kogutud informatsioonile ning anname neile esmase nime kirjelduse, mis antud perioodil toimus.

```
from collections import OrderedDict

periods = OrderedDict()
periods['NIGHT'] = { 'start': 0, 'end': 6, 'activity': 'SLEEP', 'color': 'black' }
periods['MORNING'] = { 'start': 7, 'end': 10, 'activity': 'HOME', 'color': 'orange' }
periods['DAY'] = { 'start': 11, 'end': 17, 'activity': 'WORK', 'color': 'yellow' }
periods['EVENING'] = { 'start': 18, 'end': 23, 'activity': 'HOME', 'color': 'orange' }
```

Järgnevalt uurime, mitu korda esines mingit sündmust neil perioodidel eri päevil ning leiame antud tundide vahemikus päevade lõikes keskmised esinemiste kordade hulgad. Püüame luua uue *pandas* ajaraami objekti, et tagada mugav informatsiooni kuvamine. Tulemus (Joonis 13):

```
columns = ['WC', 'SINK_USE', 'BATH_SHOWER']
res = []

def filter_by_period(data_df, period_label, start, end):
    data_df = data_df[data_df[period_label] >= start]
    data_df = data_df[data_df[period_label] <= end]
    return data_df

def filter_by_hour(data_df, start, end):
    return filter_by_period(data_df, 'hour', start, end)

def filter_by_day(data_df, start, end):
    return filter_by_period(data_df, 'weekday_nb', start, end)

def get_filter_by_workdays_fn(data_df, period):
    data_df = filter_by_day(data_df, 0, 4)
    data_df = filter_by_hour(data_df, period['start'], period['end'])
    return data_df

def find_period_events(period, result_arr=[]):
    event_df = find_avg_event_count(df, period, get_filter_by_workdays_fn)
    for column in columns:
        value = 0
        if column in event_df['activity'].values.tolist():
            value = event_df[event_df['activity'] == column].get('mean').values[0]
        result_arr.append(value)
    result_arr.append(period['start'])
    result_arr.append(period['end'])
    result_arr.append(period['activity'])
    result_arr.append(period['color'])
    return result_arr

for period in periods:
    res.append(find_period_events(periods[period], [period]))

pd_columns = ['period'] + columns + ['start', 'end', 'activity', 'color']
pd_columns
pattern_df = pd.DataFrame.from_records(res, index='period', columns=pd_columns)
pattern_df[columns].plot(kind='bar')
pattern_df
```

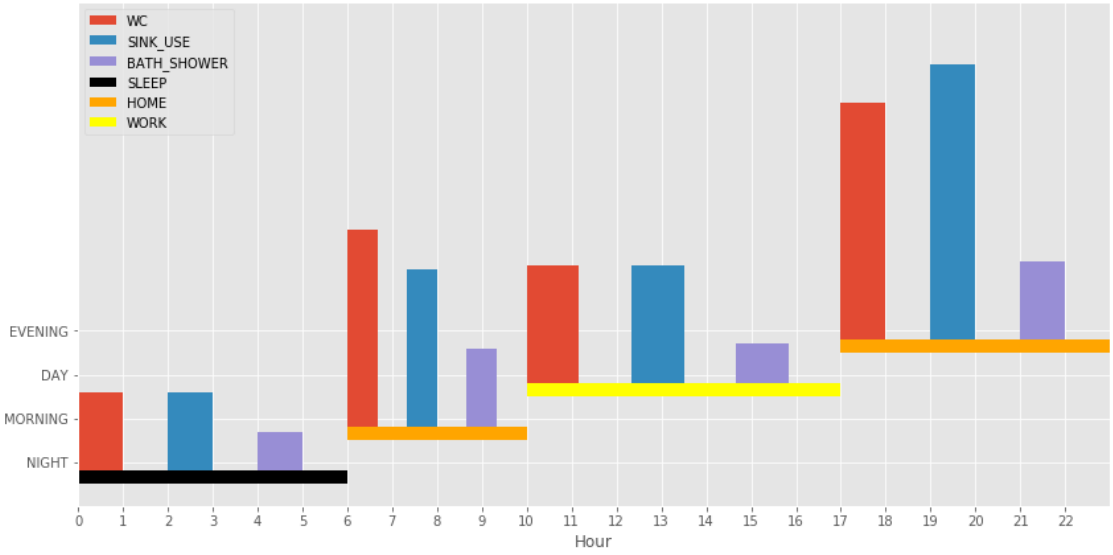


Joonis 13 Perioodide keskmised tarbimised

Nagu Joonis 13 näitab, oleme kirjeldanud pere tüüpilised tarbimised tööpäeviti perioodide kaupa. Kuid see ei ole kogu informatsioon, mis pere käitumiste kohta käesolevas töös on tuvastatud. On teada ka kõikide perioodide pikkused, mis olid tuvastatud peatükk 3-s. Kasutame antud peatükis loogiliselt eeldatud sündmusi, mis antud perioodidel peaksid esinema ning lisame need graafikule koos ajateljega, mis näitab perioodide pikkusi.

```
pattern_df
column_c={'WC':'#E24A33', 'SINK_USE': '#348ABD', 'BATH_SHOWER':'#988ED5'}

fig, ax = plt.subplots(figsize=(14,7))

x = 10
y_ticks = []
y_ticklabels = []
used_labels = []

def get_first(df):
    return df.values[0]

def do_bar(ax, begin, length, color, label):
    if label not in used_labels:
        used_labels.append(label)
        return ax.broken_barh(begin, length, facecolors=color, label=label)
    else:
        return ax.broken_barh(begin, length, facecolors=color)

for p in pattern_df.index.values.tolist():
    p_df = pattern_df[pattern_df.index == p]
    start = get_first(p_df['start'])
    end = get_first(p_df['end'])
    label = get_first(p_df['activity'])
    e = end-start
    w = e / len(columns) / 2
    begin = start
    for c in columns:
        value = get_first(p_df[p_df.index == p][c])
        do_bar(ax, [(begin, w)], (x + 3, 9 * value), column_c[c], c)
        begin += w*2
    color = get_first(p_df['color'])

    do_bar(ax, [(start, end-start)], (x, 3), color, label)
    y_ticks.append(x+5)
    y_ticklabels.append(p)
    x+=10

ax.set_ylim(5, 120)
ax.set_xlim(0, 23)
ax.set_xticks(range(0, 23))
ax.set_xlabel('Hour')
ax.set_yticks(y_ticks)
ax.set_yticklabels(y_ticklabels)
ax.grid(True)
ax.legend(loc='upper left')

plt.show()
```

Kasutame kogutud informatsiooni kuvamiseks *matplot* teeki, mis lubab meil ehitada katkendlikke tulpdiagramme. Kuvame iga perioodi põhilise tegevuse peale omakorda Joonis 10 peal tuvastatud väärtused.



Joonis 14 Keskmiste väärtuste alusel koostatud käitumismuster

Joonis 14 on käesoleva töö lõplikud tulemused, mida tuvastati viie minutilise vahemiku järel edastatud andmestikult. Joonis kirjeldab perioodi pikkust, perioodil esinevate keskmiste sündmusi, nende keskmist esinemise arvu ning vähest tarbimist põhjustanud ajavahemike kaudu eeldatud perioodi tegevusi.

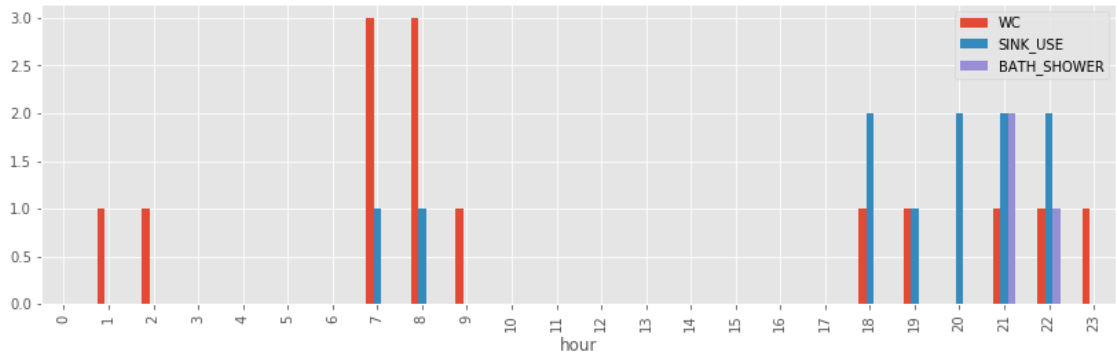
Püüame tulemust valideerida. Uurime tulemust mõne konkreetse nädala lõikes tunnipõhisel ajateljel ning vaatleme, kas tulemused on pigem sarnased keskmisega. Valime nädalaks 2017 aasta aprilli esimese täisnädala ning kuvame tulemuse iga päeva kohta.

```
month = 4
d_start = 3
d_end = 7

data_df = water_meter_data[0]['data']
data_df = data_df[data_df['month'] == month]
data_df = data_df[data_df['day'] >= d_start]
data_df = data_df[data_df['day'] <= d_end]

by_day_activities = None

hours = data_df['hour'].unique().tolist()
days = data_df['day'].unique().tolist()

res_arr_cols = ['hour'] + columns

day_pd_arr = []
def create_day_pd(day_arr):
    return pd.DataFrame.from_records(day_arr, index='hour',
columns=res_arr_cols)
```

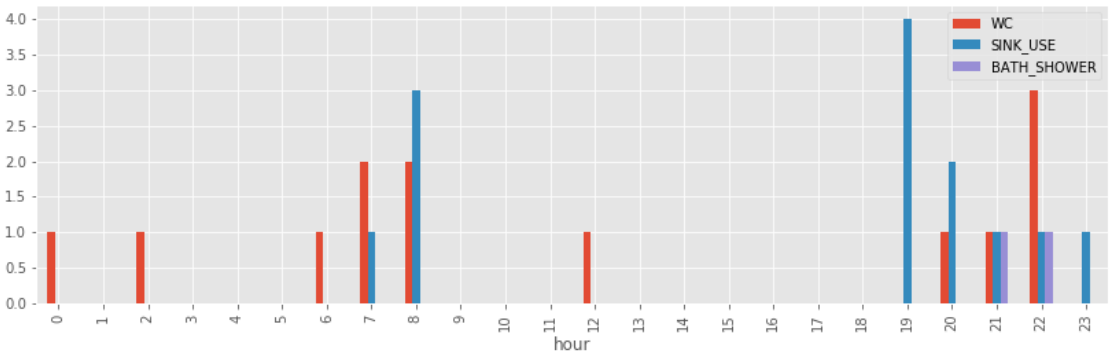
```
for day in days:
    result = []
    for hour in hours:
        res_arr = []
        res_arr.append(hour)
        by_hour_df = data_df[data_df['day'] == day].copy()
        by_hour_df = by_hour_df[by_hour_df['hour'] == hour]
        by_hour_df = by_hour_df[by_hour_df['activity'].isin(columns)]
        by_hour_df = by_hour_df.groupby('activity').size()
        by_hour_df = by_hour_df.reset_index(name='count')
        for col in columns:
            if by_hour_df[by_hour_df['activity'] == col].size > 0:
                value = get_first(by_hour_df[by_hour_df['activity'] ==
col]['count'])

            else: value = 0
            res_arr.append(value)
        result.append(res_arr)
    day_pd_arr.append(create_day_pd(result))
```

Tulemused:

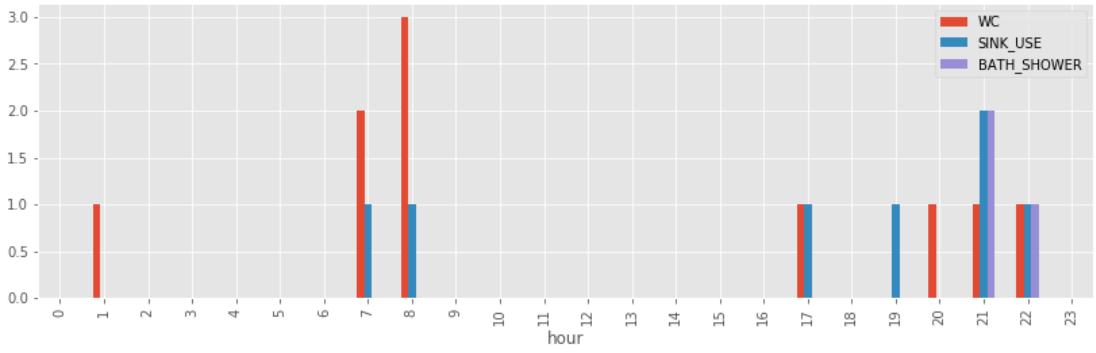
Järgnevatel graafikutel on esitatud tööpäevade kaupa tunnipõhiselt tuvastatud sündmuste arv. Graafikud võimaldavad lihtsat ülevaadet nädala sees toimuvast käitumismustrist.



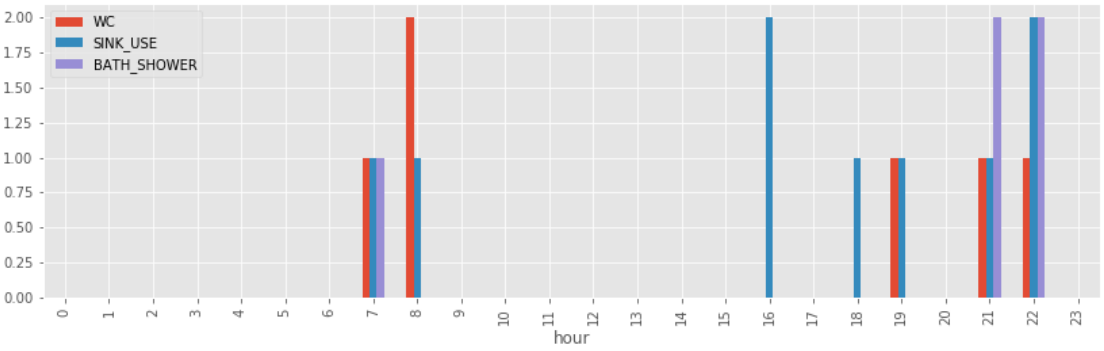
Joonis 15 Esmaspäeva sündmuste graafik



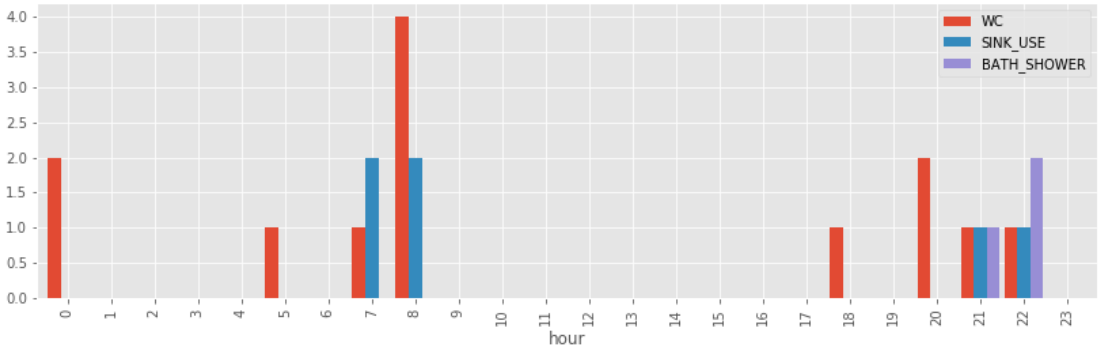
Joonis 16 Teisipäeva sündmuste graafik



Joonis 17 Kolmapäeva sündmuste graafik



Joonis 18 Neljapäeva sündmuste graafik



Joonis 19 Reede sündmuste graafik

Joonis Joonis 15, Joonis 16, Joonis 17, Joonis 18 ja Joonis 19 võimaldavad meil valideerida kirjeldatud käitumismustri ning mingi reaalelulise nädala pealt (antud andmehulgast) tuvastatud sündmuste sarnasust. Tulemused on sarnased pigem öö, õhtu ning hommiku perioodidel. Antud nädala jooksul on päevane periood pigem sündmuste vaene. Samas on võimalik valideerida ka sündmuste abil päevaseid oleku perioode. Näeme, et päevane aeg ning öine on pigem sündmusteta, õhtuti perioodil, mis märgiti ka peatükis 3 pesuskäigu perioodiks on täpselt samad perioodid. Antud nädala puhul hommikused ülestõusmise sündmused algavad küll veidi hiljem ning päevane töö periood veidi varem. Kui uurida tarbimisi päevasel perioodil, on näha, et perioodil 18. märts kuni 26 märts on tarbimine kõrgem. Antud periood oli Eestis koolivaheaeg ning mille tarbimisi arvestas ka käitumismustri graafik.

Käesolev töö püüab vastata probleemidele, mis on kirjeldatud peatükis 1. Joonis 14 kirjeldab keskmiste tulemuste kaudu sisuliselt objekti pere harjumusi, mida võib vaadelda kui elu toimimise kirjeldamist, mis on peatükis 1 üheks uuritavaks probleemiks. Käesolev töö on tuvastanud pere keskmise magamise pikkuse, tüüpilise perioodi, millal see sageli esineb, kui pikalt ning mis perioodidel ollakse kodunt eemal ja palju muud. Vaatleme, mis antud informatsioon andmete turvalisuse osas tähendab.

5.2 Andmete turvalisus ja privaatsus

Käesolev töö on tuvastanud mitmesuguseid andmeid perede tarbimisharjumuste kohta. Tulemustest on ilmne järeldada, et tihedast tarbimisandmete kogumisest, et pakkuda teenuseid nagu lekete tuvastamine, tegevuste juhtimine või mingi seadme oleku muutmine, võimaldab lõppkasutaja privaatse informatsiooni kohta küllaltki palju teada saada, eeldades, et lõppkasutaja on tsiviilisik, kellele on pakutud kauglugemist võimaldavad seadmed.

Kuigi käesolev töö kasutas andmeid veekasutuse valdkonnast, võib eeldada, et sarnaselt saab tarbimisinformatsiooni kasutades tuvastada informatsiooni ka energeetika valdkonnast, kus tänapäeval on tunnipõhine kauglugemine pea kõigis kodudes Eestis [27]. Kui eeldada, et elektrinäitude andmetest, saab sarnast informatsiooni kaevata, mis vee informatsiooni puhul, siis võib järeldada et pea kõik Eestis paiknevad eramud saadavad informatsiooni oma keskmise une aja, kodus olemise perioodide ning kodunt

eemal oleku perioodide kohta. Kui asjade interneti areng jõuab momenti, kui kauglugemine veetarbimise valdkonnas on jõudnud iga tunni tasemele samal hulgal nagu täna on energeetika valdkonnas, on vee ning elektrit pakkuvate ettevõtete kätte koondunud pea piisavalt andmeid, et ennustada inimeste käitumist äärmiselt kõrge täpsusega.

Käesoleva töö andmed põhinesid täiesti tavaliste perede reaalsete tarbimisandmete pealt ning NAS ei avaldanud pere kohta muud informatsiooni, kui pereliikmete arv, mis on nende poolt õige, sest teisel juhul korjaks käesoleva töö autor informatsiooni, mida saab ära kasutada.

Antud võimalused seavad mitmeid riske asjade tarbimisandmete kogumise valdkonnas ning ettevõtted, mis pakuvad asjade interneti teenuseid, või üldiselt tegelevad andmete kogumise ning analüüsiga peavad tegelema andmekaitse muredega. Üks murekohti antud ettevõtete jaoks on 2018 aasta mais jõustuv Euroopa andmekaitse reform [4].

5.3 Euroopa andmekaitse reform

Euroopa andmekaitse reform on mõeldud Euroopa liidu poolt eraisiku andmete kaitseks, mis peaks tagama, et erinevad infosüsteemid peavad andmete kogumisel arvestama uusi põhimõtteid[28]:

- Andmed, mida eraisikute kohta korjatakse on eraisiku omand, ehk neid ei tohi kasutada kõrvaliseks otstarbeks või andmete müügiks (reklaam, statistika).
- Andmete korjamiseks on ettevõttel õigus kui:
    - Seadus on seda lubanud.
    - Neil on leping eraisikuga andmete defineeritud andmete kasutamiseks või korjamiseks.
    - Eraisik on andnud eraldi kinnituse (ingl *concent*).
- Andmete minimaalsuse põhimõte:
    - Teenuse osutamiseks korjata nii vähe kui võimalik. Käesoleva töö kontekstis saab tuua näite, et kui arveid edastatakse teenuse eest iga kuu,

siis andmeid ei tohi korjata tihedama sagedusega kui just selle eest pole küsitud eraldi luba.

Käesoleva töö kontekstis on kõige aktuaalsem neist viimane punkt, kus iga astme jagu täpsema andmete küsimise korral tuleb küsida, kas luba või lepingu muudatus või võib ettevõte olla sunnitud tasuma määruses märgitud rahatrahvi.

Käesoleva töö algul otsitakse lahendust probleemile kui sagedasest näidu edastamisest võib välja lugeda enam, kui ainult näidu muutu. Käesoleva töö varasematest tulemustest võib järeldada, et sagedus on kindlasti pikem, kui üks tund või viis minutit. Autor pakub, et suuruseks, mis lõppkasutaja privaatsust oluliselt ei eiraks on üks kuu. Sellise edastusvahemiku puhul on privaatsete andmete tuvastus viidud minimaalsele tasemele.

6 Kokkuvõte

Töö koosnes sisuliselt kolmest osast: probleemide püstitusest, neile vastuste leidmisest ning diskussioonist tulemuste üle. Käesoleva töö raames põhiliselt analüüsiti reaalsete tsiviilisikute perede tarbimisandmeid tööpäevade lõikes.

Andmete põhjal tuvastati perioodid, kus tarbimine oli suur või pea olematu ning ajalise määratluse põhjal (põhiliselt 1 tund) eeldati neil momentidel tegevusi. See võimaldas tuvastada, et tarbimisandmetes on kasulikku informatsiooni. Seejärel uuriti andmeid klastritesse jagamise teel, mille põhjal üritati tuvastada sündmusi. Teostuse käik näitas, et kindlaid sündmusi on võimalik tuvastada viie minuti informatsiooni aknas. Siiski võimaldaks paremat tuvastamise lahendust sagedam saatmise edastus kui viis minutit.

Antud informatsioon koondati, mis võimaldas kirjeldada käitumismustreid ning ajendas diskussiooni andmete turvalisuse ning privaatsuse valdkonnas. Tuvastati, et tarbimisandmete põhjal on võimalik kirjeldada elu toimimist, mis tekitab erinevate teenuste pakkumisel, nagu on lekete tuvastus, probleeme andmekaitse valdkonnas. Pakuti välja, et sisuliselt privaatsuse tagamiseks on vaja sarnase teenus loogika viia anduri tasemele ning näidu informatsioon saata privaatsust tagava perioodi puhul, mis autori arvates on üks kuu.

Töö kõige probleemseim valdkond on leitud sündmuste ning perioodide valideerimine. Autor soovis töö käigus katseobjekti üles ehitada oma majapidamise raames, mille abil oleks võimaldatud valideerimine ning erinevate perioodide muutmine hõlpsal viisil (näiteks ehitades valideerimiseks mingi mobiilirakenduse), kuid seda ei võimaldatud. Siiski oli võimalik valideerida tuvastatud perioodide ning neil toimuvate sündmuste sarnasust. Tulemus oli sarnane.

Käesoleva töö käigus püstitatud probleemidele oli NAS-i poolt tarnitud andmehulga abil võimalik vastata ning töö tulemit võib määrata edukaks.

6.1 Tööjärgsed eesmärgid

Käesoleva töö põhjal saab kirjeldada eesmärke, mida võib lugeda töö jätku osadeks. Kõige olulisem tegevus, mis seni koondatud materjaliga teostada tuleks on valideerimine

tarbimisinformatsiooni sündmustamise osas, mis kahjuks käesoleva töö raames asjade interneti valdkonna uudsuse tõttu ei olnud võimalik.

Kauglugemine ning selle abil sündmuste tuvastamine võimaldab andmeid hästi kirjeldada ning tänu sellele on võimalik luua mitmeid tarku lahendusi. Näiteks kauglugemine võimaldab üles ehitada elektripakkujatel erinevaid hinnastamise mudeleid ning palju dünaamilisemat kuluarvestust lõppkasutaja jaoks [29]. Pigem tähendab see lõppkasutajale hinnalangust, sest vee-ettevõtted saavad pakkuda hinnaplaani vastavavalt konkreetse perioodi võimekuse ning sarnaste andurite poolt pakutud info alusel nagu kirjeldab käesolev töö [30]. Samuti võimaldaks selline mõtteviis antud ettevõtetel ka ennustada suure magnituudiga koormuse tõuse veepumbajaamadele, ennustada ja tuvastada lekkeid ning planeerida reservide kasutust [31]. Lõppkasutajad võivad olla nõus selle nimel ka privaatsusest loobuma ning on nõus andmeid jagama mitmete osapoolte vahel, et tagada mõni mugav teenus. Samas on selline andmete jagamine äärmiselt ohtlik, sest tänasel hetkel on neid andmeid korjamas asjade interneti näol eraettevõtted, kes kõik püüavad ehitada esimest minimaalset töötavat toodet. Andmete lekke korral sealsetest süsteemidest saab ennustada, millal keegi kodus on, õppida tema käitumist, ennustada tema asukohta ja isegi harjumusi või terviseprobleeme kui saatmissagedus on tihe. Seega on üks tööjärgsetest eesmärkidest turvalisuse temaatika uurimine asjade interneti valdkonnas. Turvalisuse peale peaks enne iga asjade internetiga seonduva toote väljastamist ning vältida ühendatud seadmete kuritahtliku kasutamist enne kui ilmneb *miljardi dollari bugi* [32].
Käesoleva töö järgsetest eesmärkide tehnilistes võtmes tuleks katsetada mõne targa lahenduse ehitamist näiteks nutikodu näitel. Uurida, kas on võimalik ehitada intelligentne süsteem, mis sündmuste ning perioodide analüüsi põhjal oskab majapidamises suurendada säästmist, näiteks tarkade lülituste abil, kütte, veesoojendamise ning ka elektrit tarbivate seadmete seas.

Nagu varem mainitud on üheks targaks lahenduseks lekete tuvastus, mille probleemidele suuresti käesolev töö on ülesehitatud ning mille lahendamist NAS oluliseks peab. Lekete tuvastusele hea lahenduse leidmine hõlmab sündmuste tundmist ning käitumismustrite analüüsi, kuid seda on eraisikutest tarbijate puhul ettevõttele andmekaitsemäärusest tulenevalt riskantne lahendada serverilahendusena, sest see tähendab selgitamist erinevate andmete kogumise vajaduse osas. Käesoleva töö loomise raames on jõutud arusaamale, et lekete tuvastamine peab antud kontekstis jõudma anduri tasemele, kuid see

tähendab tehnilist katsumust, sest tüüpiliselt on *LPWAN* võrgu seadmed patarei toitel. See on aluseks autori valikule kasutada *k-means* algoritmi peatükis 4, mille mälu vajadus on pigem väike, sest vaja on salvestada ainult keskväärtused ning andmeobjektid. Sellise töötava lahenduse puhul on võimalik saata lekete tuvastamise korral häire eraldi parameetrina ning näidu informatsiooni perioodil, mis on vastav ettevõtte ning kliendi poolt, või seaduse abil kokkulepitud andmekaitse nõuetele.

Teine variant on valida lekketuvastusele lihtsad parameetrid, et millise pideva voo puhul on maksimaalne kulunud tarbimishulk, mis ei ole leke. Siiski tähendab see sagedaid ning asjatuid häireid staatilise määratud väärtuse, mis tehniliselt keerukamat, kuid targemat lahendust õigustavad.

Kasutatud kirjandus

[1] Lora-Alliance.org, “LoRa® Technology,” 2016. [Online]. Available: https://www.lora-alliance.org/What-Is-LoRa/Technology. [Accessed: 22-Mar-2017].

[2] “What is LPWAN (low-power wide area network)? - Definition from WhatIs.com.” [Online]. Available: http://internetofthingsagenda.techtarget.com/definition/LPWAN-low-power-wide-area-network. [Accessed: 22-Mar-2017].

[3] “Nordic Automation Systems | LoRaWAN™ solutions & products, sensor technologies, data analysis and monitoring solutions.” [Online]. Available: http://www.nasys.no/. [Accessed: 22-Apr-2017].

[4] “Euroopa Andmekaitse reform | Andmekaitse Inspektsioon.” [Online]. Available: http://www.aki.ee/et/eraelu-kaitse/euroopa-andmekaitse-reform. [Accessed: 10-Apr-2017].

[5] “IoT data consumption a key challenge in edge device scalability.” [Online]. Available: http://www.theserverside.com/podcast/IoT-data-consumption-a-key-challenge-in-edge-device-scalability. [Accessed: 24-Apr-2017].

[6] “A Simple Explanation Of ‘The Internet Of Things.’” [Online]. Available: https://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/#6977e0431d09. [Accessed: 05-May-2017].

[7] D. Carboni, A. Gluhak, J. A. McCann, and T. H. Beach, “Contextualising water use in residential settings: A survey of non-intrusive techniques and approaches,” *Sensors (Switzerland)*, vol. 16, no. 5, 2016.

[8] A. R. Bodhe, R. Singh, and A. Bawa, “An Internet of Things solution for sustainable domestic water consumption,” *2016 Int. Conf. Comput. Syst. Inf. Technol. Sustain. Solut.*, pp. 224–229, 2016.

[9] J. Fogarty, C. Au, and S. E. Hudson, “Sensing from the Basement : A Feasibility Study of Unobtrusive and Low-Cost Home Activity Recognition,” *19th Annu. ACM Symp. User interface Softw. Technol.*, pp. 91–100, 2006.

[10] R. Cardell-oliver, J. Wang, and H. Gigney, “Smart Meter Analytics to Pinpoint Opportunities for Reducing Household Water Use,” vol. c, no. 6, pp. 1–9, 2014.

[11] D.-Y. Kim, Y.-S. Jeong, and S. Kim, “Data-Filtering System to Avoid Total Data Distortion in IoT Networking,” *Symmetry (Basel).*, vol. 9, no. 1, p. 16, 2017.

[12] “LoRa Technology.” [Online]. Available: https://www.lora-alliance.org/What-Is-LoRa/Technology. [Accessed: 22-Apr-2017].

[13] “low-power-wireless-sensor-network-technologies-and-standards-for-the-internet-of-things-38-638.jpg (638×452).” [Online]. Available: https://image.slidesharecdn.com/connect2systemsiop12sep16finalversion-160912141117/95/low-power-wireless-sensor-network-technologies-and-standards-for-the-internet-of-things-38-638.jpg?cb=1473689971. [Accessed: 22-Apr-2017].

[14] “HTTP request methods - HTTP | MDN.” [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods. [Accessed: 23-Apr-2017].

[15] “Welcome to Python.org.” [Online]. Available: https://www.python.org/. [Accessed: 23-Apr-2017].

[16] “Python Data Analysis Library — pandas: Python Data Analysis Library.” [Online]. Available: http://pandas.pydata.org/. [Accessed: 27-Apr-2017].

[17] “NumPy — NumPy.” [Online]. Available: http://www.numpy.org/. [Accessed: 27-Apr-2017].

[18] “StatsModels: Statistics in Python — statsmodels 0.8.0 documentation.” [Online]. Available: http://www.statsmodels.org/stable/index.html. [Accessed: 27-Apr-2017].

[19] “Matplotlib: Python plotting — Matplotlib 2.0.0 documentation.” [Online]. Available: https://matplotlib.org/. [Accessed: 27-Apr-2017].

[20] “scikit-learn: machine learning in Python — scikit-learn 0.18.1 documentation.” [Online]. Available: http://scikit-learn.org/stable/. [Accessed: 01-May-2017].

[21] “Choosing R or Python for data analysis? An infographic.” [Online]. Available: https://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis#gs.l7O7kmQ. [Accessed: 23-Apr-2017].

[22] “Project Jupyter | Home.” [Online]. Available: http://jupyter.org/. [Accessed: 27-Apr-2017].

[23] D. Mining, “Springer Series in Statistics The Elements of,” *Math. Intell.*, vol. 27,

no. 2, pp. 83–85, 2009.

[24] “sklearn.cluster.KMeans — scikit-learn 0.18.1 documentation.” [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html. [Accessed: 01-May-2017].

[25] C. Science and I. Technology, “SCIENTIFIC JOURNAL OF RIGA TECHNICAL UNIVERSITY Computer Science Information Technology and Management Science 2009,” *Inf. Technol. Manag.*, no. 2005, pp. 81–86, 2009.

[26] J. (Computer scientist) Bell, *Machine learning : hands-on for developers and technical professionals*. Wiley, 2015.

[27] “Elektriturg ja elektrisüsteem - Elektrilevi.” [Online]. Available: https://www.elektrilevi.ee/elektriturg. [Accessed: 04-May-2017].

[28] “EUROOPA PARLAMENDI JA NÕUKOGU MÄÄRUS (EL) 2016/679.” 2016.

[29] C. Rougé, P. Garrone, and M. Pulido-velazquez, “Smart Meter Enabled Dynamic Pricing of Water,” *Int. Congr. Environ. Model. Softw.*, 2016.

[30] T. R. Gurung, R. A. Stewart, C. D. Beal, and A. K. Sharma, “Smart meter enabled informatics for economically efficient diversified water supply infrastructure planning,” *J. Clean. Prod.*, vol. 135, pp. 1023–1033, 2016.

[31] C. D. Beal, R. A. Stewart, and K. Fielding, “A novel mixed method smart metering approach to reconciling differences between perceived and actual residential end use water consumption,” *J. Clean. Prod.*, vol. 60, pp. 116–128, 2013.

[32] P. McDaniel and S. McLaughlin, “Security and privacy challenges in the smart grid,” *IEEE Secur. Priv.*, vol. 7, no. 3, pp. 75–77, 2009.

[33] “Git.” [Online]. Available: https://git-scm.com/. [Accessed: 07-May-2017].

Lisa 1 – Käesoleva töö käigus kirjutatud kood

Kood on leitav *git*-i [33] versioonihalduse süsteemi repostooriumist, mis asub järgneval aadressil: https://github.com/mkasep/magister