Päätelaitteiden automaatiotestaus

Jonni Nieminen

Haaga-Helia ammattikorkeakoulu

Amk-opinnäytetyö

2023

Tradenomi, tietojenkäsittely tutkinto

Tiivistelmä

Tekijä(t)
Jonni Nieminen

Tutkinto
Tradenomi, tietojenkäsittely

Raportin/Opinnäytetyön nimi
Päätelaitteiden automaatiotestaus

Sivu- ja liitesivumäärä
25 + 3

Opinnäytetyön toimeksiantajana oli Millog Oy, joka toimii Puolustusvoiminen elinkaaripalveluiden tuottajana. Tietoteknisiä palveluita Millog tarjoaa tietoturva- ja kyberratkaisuissa sekä tietoliikenteinfan ylläpitoon ja elinkaarenhallintaan. Opinnäytetyö toteutettiin ICT-palveluita tuottavassa yksikössä. Tarpeena päätelaitteiden automatisointi nousi esille, kun laitteita asennetaan isoja määriä, lisäksi asennuksia saattaa tehdä myös henkilöt riippumatta koulutustaustasta niin valmis ohjelmistorobotti tulisi tähän tarpeen. Lisäksi kun luodaan geneeriset testit laitteistojen tarkistukselle, säästetään tässä aikaa moneltakin taholta. Tässä vaiheessa toteutettava kokonaisuus on vain prototyyppi, jota tullaan jalostamaan yrityksen tarpeisiin.

Käytettäväksi viitekehykseksi valikoitui Robot Framework, koska se oli ollut yrityksessä jo jollakin tapaa käytössä ja lisäksi sen mahdollistaa testien määrittelyä ilman laajempaa ohjelmointi taitoa. Ohjelmistorobotin määrittely tehtiin ennalta annetuista määrittelyistä, jotka nousivat toimeksiantajan puolelta esille. Testattavaksi laitteistoksi otettiin mini-PC tietokone, Linux käyttöjärjestelmällä. Testattavista laitteista haluttiin raportoida laitetietoja, testata verkkoyhteyttä, asentaa tarvittavia paketteja sekä lukea GPS dataa erillisen GPS vastaanottimen kautta. GPS datan käsittelyyn tehtiin Python koodilla pieni ohjelman pätkä, jotta se saatiin luettavaan muotoon ja tulostettua Robot Frameworkin avulla.

Opinnäytetyön lopputuloksena muodostui Robot Frameworkia hyödyntävä testi kokonaisuus, jolla saadaan halutut tiedot laitteistoista raportoitua, sekä testattua että laitteet toimivat niin kuin on haluttu. Kaiken kaikkiaan testitapauksia muodostui 11 kappaletta. Tästä saadaan pohjaa asennettujen laitteiden dokumentoinnille, ja tarkoituksena tästä olisi jalostaa tuotantoympäristöön sopiva testikokonaisuus, jota voidaan suorittaa tarvittaessa erinäisissä olosuhteissa irrallaan laajemmasta infrastruktuurista.

Asiasanat
Testausautomaatio, Robot Framework, SSH

Sisällys

1. Johdanto

Opinnäytetyön toimeksianto tuli Millog Oyn tarpeesta. Millog Oy on erikoistunut huolto-, kunnossapito- ja logistiikkapalveluihin sekä kokonaisvaltaiseen elinkaarenhallintaan. Yrityksen pääpaino on kaluston ja järjestelmien huolto- ja kunnossapidossa. Millog Oy huoltaa ja ylläpitää Puolustusvoimien kalustoja ja järjestelmiä. Asiakkaina on myös joitakin yksityisen sektorin toimijoita, jotka ovat osana valtakunnallista huoltovarmuus ja turvallisuus tekijöitä. Yrityksessä uusia laitteistoja otetaan käyttöön ja vanhoihin tehdään jatkuvasti päivityksiä, joiden asennukset ja validointi teettää paljon manuaalista työtä ja vaatii tietynlaista osaamista, että voidaan tarvittavia asioita tarkistella.

Opinnäytetyön tavoitteena on toteuttaa ohjelmistorobotiikan avulla testiautomaatio kokonaisuus, jota voidaan operoida kevyiden ohjeiden kanssa, vaikka ei tietoteknistä osaamista löytyisi niin paljoa.

Työntoteuttamiseen valikoitui Robot Framework kehys, kun se on jo ollut testikäytössä yrityksen sisällä ja sen on todettu taipuvan moneen eri tarkoitukseen ja sen laajentamiseen on melko rajaton määrä mahdollisuuksia. Näin ollen itse automaation ja robotin tuottamiseen ei tarvita resursseja, vaan voidaan tuottaa testejä suoraan esimerkkitapauksia hyödyntäen, jossa tarvittavia testejä ja tarkistuksia haettaisiin kohde laitteilta käsin.

Robotin pääasiallinen tehtävä tulee olemaan päätelaitteeseen yhteydenmuodostuksen jälkeen kerätä laitteesta tekniset tiedot, järjestelmätiedot, varmistaa että laite toimii halutulla tavalla, tarkistaa laitteeseen asennetut ohjelmistot, asentaa haluttuja paketteja sekä tuottaa dokumentaatio, jossa nähdään tehdyt testit sekä laitetiedot.

Tositapaukset ovat yhdistelmä erilaista testausta, mukaan lukien järjestelmätason testausta, hyväksymistestausta sekä regressiotestausta.

Järjestelmätason testaus suoritetaan järjestelmän laitteiston ja ohjelmiston tilan ja suorituskyvyn tarkistamiseksi, kuten suorittimen, muistin, grafiikkasuorittimen ja asennettujen pakettien tarkistaminen. Hyväksymistestaus suoritetaan järjestelmän toimivuuden tarkistamiseksi, esimerkiksi sen tarkistamiseksi, että GPS-tietoja kerätään ja jäsennetään oikein. Regressiotestaus voidaan suorittaa myös "Asenna paketti" -testitapauksessa sen tarkistamiseksi, onko paketti asennettu oikein asennuksen jälkeen.

Testausympäristö opinnäytetyössä on CompuLabin FitLet2 mini-PC, johon on valmiiksi asennettuna Linux Mint käyttöjärjestelmä, ja oheislaitteena Globalsatin BU-353S4 GPS vastaanotin. Testit itsessään suoritetaan Windows 10 pohjaiselta työasemalta, jossa on asennettuna RobotFramework 6.0 sekä sen vaatima SSHLibrary kirjasto sekä Python 3.9 versio.

Keskeisimmät käsitteet

Avoin lähdekoodi	Avoin lähdekoodi tarkoittaa mitä tahansa ohjelmaa, jonka lähdekoodi on saatavilla käytettäväksi tai muokattavaksi käyttäjien tai muiden kehittäjien omilla työkaluilla ja -tavoilla. Sitä ei ole mitenkään piilotettu tai lukittu.
SSH	Secure Shell (SSH) -protokolla on verkkoprotokolla suojattua tiedonsiirtoa, etäkomentorivikirjautumista, etäkomentojen suorittamista ja muita suojattuja verkkopalveluita varten kahden verkossa olevan tietokoneen välillä.
SCP	SCP (secure copy) on komentorivityökalu, jonka avulla voit kopioida tiedostoja ja hakemistoja turvallisesti kahden sijainnin välillä.
Repository	Repository on tallennuspaikka, joka sisältää olennaisia ja suosittuja ohjelmistoja Linux-jakeluille ja jokaisella jakelulla on omat viralliset arkistot.
PXE	Preboot Execution Environment on asiaks-palvelin rajapinta, jonka avulla verkossa olevat tietokoneet voidaan käynnistää palvelimenlta ennen työaseman omaa käynnistys sektoria. Tätä käytetään pääasiallisesti laitteistojen asennuksien keskitetyissä asennuksissa.
Telnet	Telnet on protokolla, joka mahdollistaa käyttäjän tietokoneen tai muun laitteen yhdistämisen etäpalvelimeen tai tietokoneeseen Internetin tai muun verkon yli. Telnet-ohjelmiston avulla käyttäjä voi kirjautua etäjärjestelmään ja käyttää sen resursseja, kuten ohjelmistoja, tiedostoja ja tulostimia, ikään kuin hän olisi fyysisesti kyseisellä järjestelmällä.

2. Ohjelmistorobotiikka ja testiautomaatio

Lähtökohtaisesti ohjelmistorobotiikalla ja testiautomaatiolla saavutetaan paljon etuja. Ohjelmistorobotiikalla jäljitellään käyttäjän suorittamia toimintoja ohjelmistoa hyödyntäen. Ohjelmistorobotiikalla pystytään suorittamaan isoja määriä toistuvia prosessin osia. Tuoda tehokkuutta toimintaan, vähentää virheiden mahdollisuutta ja robotit voivat työskennellä taukoamatta. (CRG Solutions, 2020)

Testiautomaatio tulee usein kyseeseen, kun käsitellään isoja kokonaisuuksia tai jatkuvasti toistuvia testejä. Testiautomaatiolla kun voidaan määrittää testejä suoritettavaksi esimerkiksi aina kun muutoksia tapahtuu tietokannassa, lähdekoodissa tai aktivoidaan tietty vaihe sovelluksessa. Kerran luodut testit ovat usein käytettävissä uudelleen sovellusten päivittyessä tai kokonaan uusien sovellusten testauksessa. Automaattinen testaaminen ei kuitenkaan pois sulje manuaalisen testaamisen tarvetta, ohjelmistorobotille täytyy kuitenkin rakentaa toimiva testitapaus, jotta se saadaan automatisoitua. (Lozowska-Pereira, 2022)

1.1 Ohjelmistorobotiikka etähallinnassa

Ohjelmistorobotiikkaa hyödynnetään monilla tavoin IT-infrastruktuurin etähallinnassa. Etäinfrastruktuurin hallinta on saanut nimityksen RIM (Remote Infrastructure Management). Konesalit kun usein sijaitsevat erillään toimistoista, joista työskentely tapahtuu, niin yhteyksien ollessa kunnossa lähestulkoon kaikki hallinta tapahtuu etäyhteydellä. Tämä toki edellyttää verkkoyhteyksien olevan kunnossa ja laitteistot määritetty hallittavaksi verkon yli. Automatisointia voidaan RPA työkaluilla tuoda hallintaan erilaisten raporttien, korjaustiedostojen, valvonnan ja esimerkiksi käyttäjähallinnan osalta. Lisäksi kun tarkastellaan kokonaisuuksia, on jo olemassa monella tavalla toteutettua automaatiota mm. kuormantasauksissa, varmuuskopioinneissa ja virtualisoinneissa. (Hunter, 2022)

1.2 Robot Framework ohjelmistotestauksessa

Robot Frameworkia käytetään useissa muodoissa automaatiotestaukseen ja avainsanalähtöiseen testaukseen. Joitakin yleisiä esimerkki kohteita on:
Verkkosovellusten testaus, Robot Frameworkia voidaan käyttää verkkosovellusten toimivuuden testaamiseen, mukaan lukien käyttöliittymän, tietokantayhteyksien ja taustaprosessien testaamiseen.
Työpöytäsovellusten testaus, Robot Frameworkia voidaan käyttää työpöytäsovellusten testaamiseen, mukaan lukien käyttöliittymän, tietokantayhteyksien ja muiden järjestelmien integroinnin testaamiseen.
Mobiilisovellusten testaus, Robot Frameworkia voidaan käyttää mobiilisovellusten testaamiseen, mukaan lukien käyttöliittymän, tietokantayhteyksien ja muiden järjestelmien integroinnin testaamiseen.

API-testaus, Robot Frameworkia voidaan käyttää API:iden (Application Programming Interface) testaamiseen lähettämällä pyyntöjä API:lle ja tarkistamalla vastaukset (Neova Tech Solutions Inc, 2022).
Integraatiotestaus, Robot Frameworkilla voidaan testata eri järjestelmien ja komponenttien integraatiota, mukaan lukien järjestelmien välisen tietovirran testaaminen ja järjestelmien toimivan yhdessä odotetulla tavalla.
Regressiotestaus, Robot Frameworkia voidaan käyttää regressiotestauksen suorittamiseen, mikä on prosessi, jossa testataan sovelluksen muutoksia sen varmistamiseksi, että ne eivät aiheuta uusia vikoja tai riko olemassa olevia toimintoja (Stresnjak & Hocenski, 2022).

1.3 Robot Framework lyhyesti

Robot Framework on avoimen lähdekoodin testiautomaatiokehys, joka on suunniteltu helppokäyttöiseksi ja laajennettavaksi. Se tukee monenlaisten sovellusten testausta, mukaan lukien web-, työpöytä- ja mobiilisovellusten testaus. Robot Frameworkin avulla voidaan kirjoittaa testitapauksia käyttämällä luonnollisen kielen syntaksia ja kehys huolehtii testien suorittamisesta ja raporttien luomisesta. Robot Frameworkin arkkitehtuuri on hyvin modulaarinen (Robot Framework, 2022b)

Robot Framework tarjoaa erilaisia avainsanoja ja kirjastoja, joita voi käyttää testitapauksissa vuorovaikutuksessa sovelluksien kanssa ja suorittamaan erilaisia tehtäviä. Voi esimerkiksi käyttää avainsanoja avatakseen verkkoselaimen, siirtyäkseen verkkosivustolle, täyttääkseen lomakkeen ja lähettääkseen lomakkeen. Voit myös käyttää kirjastoja vuorovaikutukseen tietokantojen, verkkopalvelujen ja muiden järjestelmien kanssa. (Robot Framework, 2022b)

Kun tehdään testejä, Robot Framework suorittaa testitapaukset ja luo tuloksista loki- ja raporttitiedoston. Lokitiedosto (Kuva 1) sisältää yksityiskohtaisia tietoja testitapauksen jokaisesta vaiheesta, mukaan lukien mahdolliset virheet tai epäonnistumiset. Raporttitiedosto sisältää yhteenvedon testituloksista, mukaan lukien läpäisseiden ja epäonnistuneiden testitapausten määrä. Raporttitiedosto on html muodossa ja kun sen avaa kertoo tausta väri onnistuneen tai epäonnistuneen testin tilan. (Kuva 2 & 3) (Robot Framework, 2022b)

Device Testing Log

Generated
20230113 13:00:34 UTC+02:00
19 minutes 7 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	10	10	0	0	00:01:17	

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
No Tags						

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Device Testing	10	10	0	0	00:01:17	

Test Execution Log

SUITE **Device Testing**

Full Name: Device Testing
Documentation: Test to validate installed devices, gathering information and install custom packages, read GPS data and parse it to readable format.
Source: C:\Users\Nieminen\Downloads\Device Testing.robot
Start / End / Elapsed: 20230113 12:59:17.337 / 20230113 13:00:34.470 / 00:01:17.133
Status: 10 tests total, 10 passed, 0 failed, 0 skipped

TEST **Open SSH Connection**

Full Name: Device Testing.Open SSH Connection
Start / End / Elapsed: 20230113 12:59:17.588 / 20230113 12:59:20.060 / 00:00:02.472
Status: PASS
Message: Connection established to RobotFramework (192.168.68.103)

KEYWORD SSHLibrary.**Open Connection** ${deviceip}
Documentation: Opens a new SSH connection to the given `host` and `port`.
Start / End / Elapsed: 20230113 12:59:17.589 / 20230113 12:59:17.589 / 00:00:00.000

KEYWORD SSHLibrary.**Login** ${username}, ${pwd}, delay=1
Documentation: Logs into the SSH server with the given `username` and `password`.
Start / End / Elapsed: 20230113 12:59:17.589 / 20230113 12:59:20.043 / 00:00:02.454

```
12:59:17.590    INFO    Logging into '192.168.68.103:22' as 'jonni'.
12:59:20.043    INFO    Read output: Last login: Fri Jan 13 12:58:01 2023 from 192.168.68.104

                        jonni@RobotFramework:~$
```

KEYWORD ${hostname} = SSHLibrary.**Execute Command** hostname

KEYWORD BuiltIn.**Set Test Message** Connection established to ${hostname} (${deviceip})

TEST **Sudo Access**

Full Name: Device Testing.Sudo Access
Start / End / Elapsed: 20230113 12:59:20.060 / 20230113 12:59:24.355 / 00:00:04.295
Status: PASS
Message: uid=0(root) gid=0(root) groups=0(root)

KEYWORD SSHLibrary.**Execute Command** sudo -s

KEYWORD SSHLibrary.**Write** ${pwd}

KEYWORD ${output} = SSHLibrary.**Execute Command** id, sudo=True, sudo_password=${pwd}

KEYWORD BuiltIn.**Set Test Message** ${output}

TEST **List installed Packages to a TXT file**

TEST **Hardware Info**

TEST **Serial**

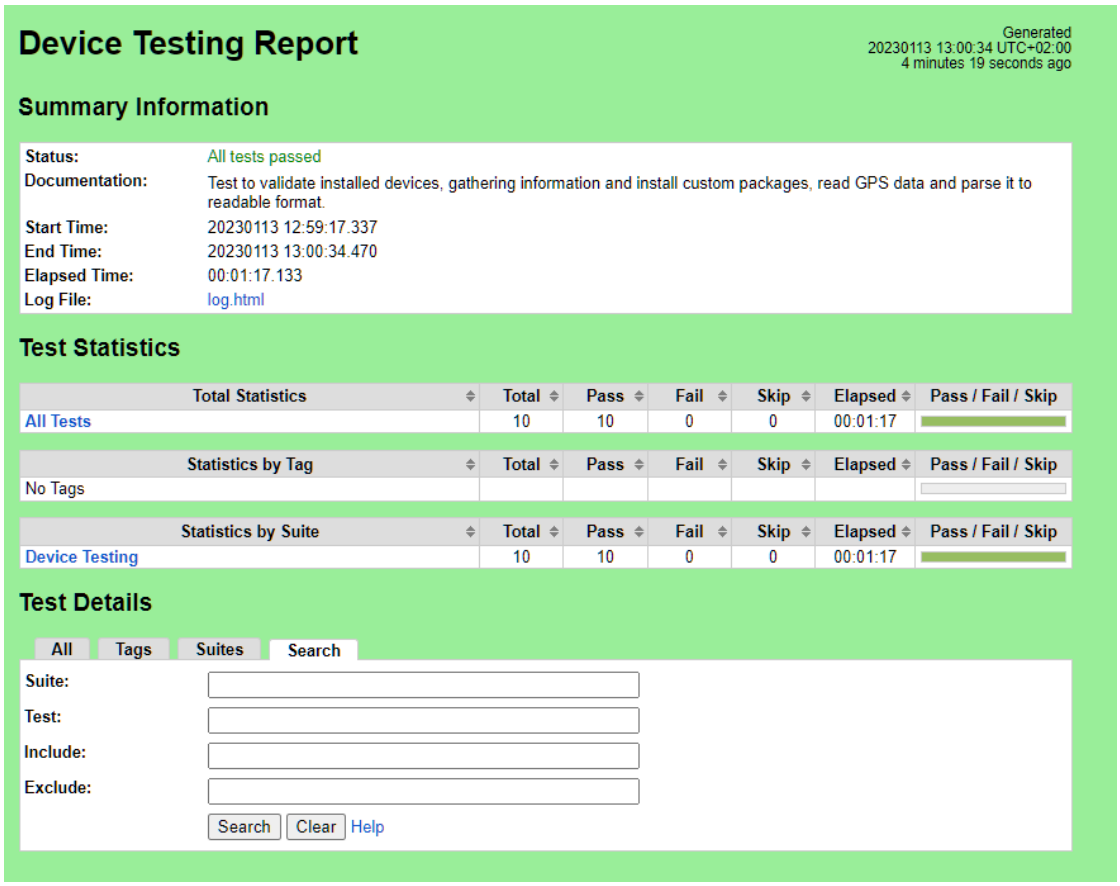
TEST **Mac Address**

TEST **Install Package**

TEST **Get Storage Usage Hours**

TEST **Reboot**

Kuva 1. Lokitiedosto



Device Testing Report

Generated
20230113 13:00:34 UTC+02:00
4 minutes 19 seconds ago

Summary Information

Status:	All tests passed
Documentation:	Test to validate installed devices, gathering information and install custom packages, read GPS data and parse it to readable format.
Start Time:	20230113 12:59:17.337
End Time:	20230113 13:00:34.470
Elapsed Time:	00:01:17.133
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	10	10	0	0	00:01:17	

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
No Tags						

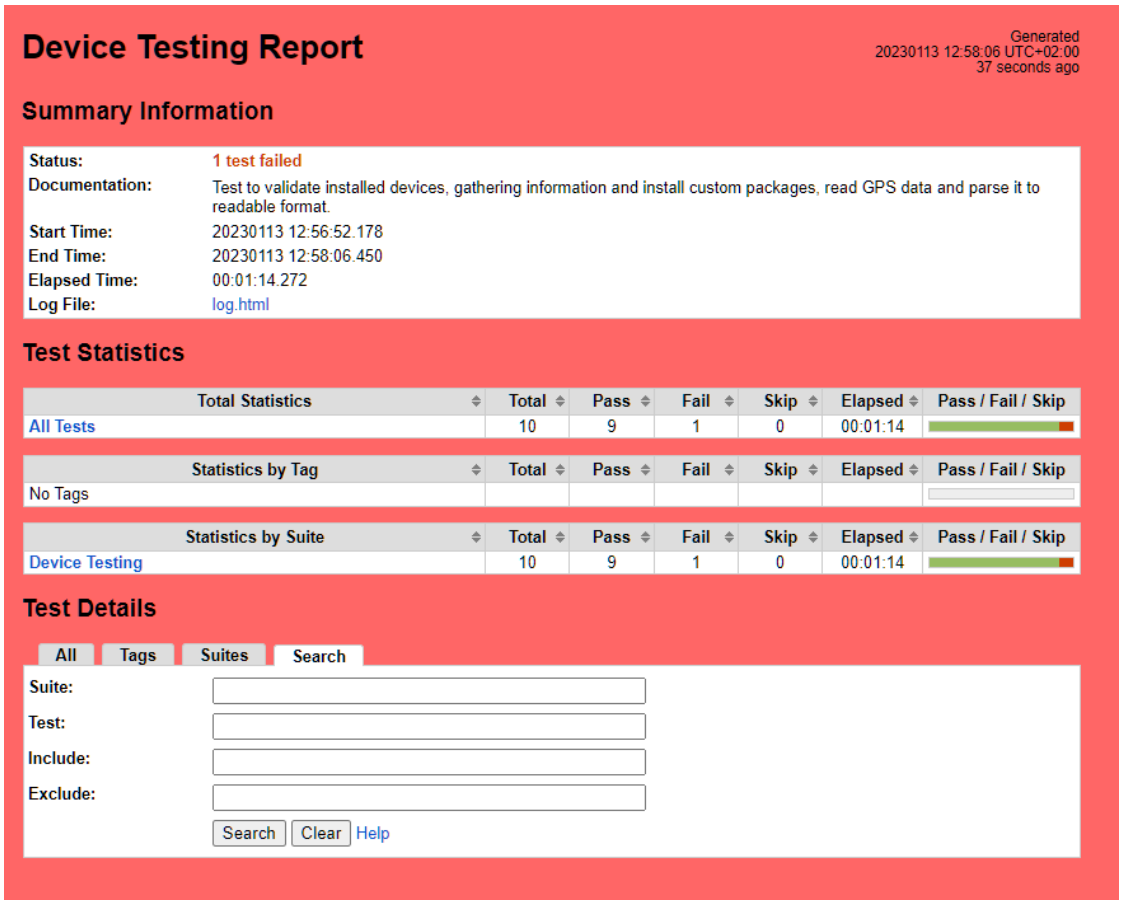
Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Device Testing	10	10	0	0	00:01:17	

Test Details

All | Tags | Suites | Search

Suite:
Test:
Include:
Exclude:
Search Clear Help

Kuva 2. Esimerkki onnistuneesta testistä



Device Testing Report

Generated
20230113 12:58:06 UTC+02:00
37 seconds ago

Summary Information

Status:	1 test failed
Documentation:	Test to validate installed devices, gathering information and install custom packages, read GPS data and parse it to readable format.
Start Time:	20230113 12:56:52.178
End Time:	20230113 12:58:06.450
Elapsed Time:	00:01:14.272
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	10	9	1	0	00:01:14	

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
No Tags						

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Device Testing	10	9	1	0	00:01:14	

Test Details

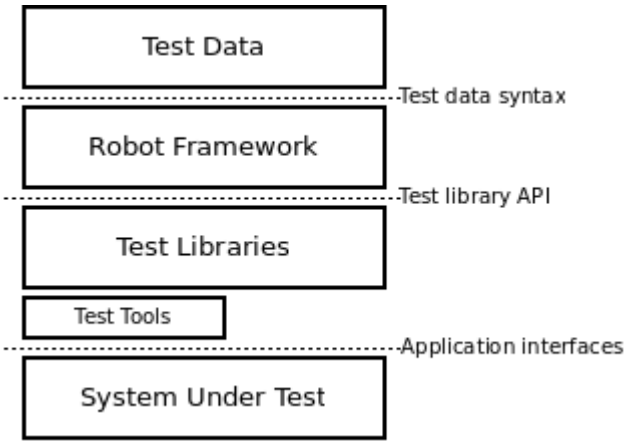
All | Tags | Suites | Search

Suite:
Test:
Include:
Exclude:
Search Clear Help

Kuva 3. Esimerkki epäonnistuneesta testistä

Kaiken kaikkiaan Robot Framework on tehokas ja joustava työkalu ohjelmistosovellusten testauksen automatisointiin. Se on laajasti käytössä organisaatioissa ympäri maailmaa, ja se soveltuu hyvin testattavaksi ohjelmistokehityksen elinkaaren kaikissa vaiheissa. (Robot Framework, 2022b)

Robot Frameworkissa on myös rikas ekosysteemi kirjastoja ja työkaluja kykyjensä laajentamiseen, ja se on monialustainen ja tukee Windowsia, Linuxia ja macOS:ää. Robot Frameworkin arkkitehtuuri on modulaarinen, joka mahdollistaa erinäisten toiminnallisuuksien rakentamista. (Kuva 4) Robot Framework on laajalti organisaatioiden käytössä ohjelmistosovellustensa automaattiseen testaukseen, ja se soveltuu testaukseen missä tahansa ohjelmistokehityksen elinkaaren vaiheessa. (Robot Framework, 2022b)



Kuva 4. Robot Framework arkkitehtuuri (Robot Framework, 2022b)

Robot Frameworkin käyttöliittymiä on useita, kuitenkin perusperiaatteelta kaikki on samanlaisia, Robot Frameworkille asetetaan halutut asetukset (Settings) joilla voidaan kutsua mm. halutut kirjastot sekä dokumentaatiot ja suorittaa mahdollisia globaaleita testisarjoja koskevia komentoja. Niitä ovat Suite Setup, jotka suoritetaan ennen testitapauksia ja Suite Teardown, jotka suoritetaan vuorostaan kaikkien testitapausten jälkeen. Näissä usein on haluttujen selainten käynnistykset ja sulkemiset määriteltynä. Sitten määritellään muuttujia, jotka ovat testien sisällä käytettäviä arvoja, jotka ovat laitesidonnaisia, eli kun testataan eri laitteita, tarvitsee jokaiselle määritellä tiettyjä muuttujia, jotta testi toimii juuri kyseiseen laitteeseen. (Koodi 1), joita sitten pystytään kutsumaan itse testitapauksissa avainsanoja käyttämällä.

```
***Settings***
Documentation           This is Device Testing Robot for theseis

Library                 SSHLibrary
Library                 Selenium

Suite Teardown          Close All Connections

***Variables***
${ipaddress}            10.10.10.1
${username}             operator
${password}             Ch4ng3!
${url}                  www.google.com
${configfile}           //SetupServer/Configurations/setup.conf
```

Koodi 1. Esimerkki robot tiedoston asetuksista ja muuttujista

1.4 Avainsanapohjainen testaus

Avainsanalähtöinen testaus (Keyword-Driven Testing) on testaustapa, jossa testitapaukset luodaan käyttämällä ennalta määritettyjä avainsanoja, jotka edustavat testin aikana suoritettavia toimia. Tässä lähestymistavassa testitapaus kirjoitetaan luonnollisella kielellä (kuten englanniksi) ja käytetään avainsanoja määrittämään suoritettavat vaiheet. (Lambdatest, 2022)

Avainsanapohjainen testaus on hyödyllinen, koska sen avulla testaajat voivat luoda testitapauksia ilman varsinaisen koodin kirjoittamista. Tämä helpottaa ei-teknisten käyttäjien testien luomista ja suorittamista ja mahdollistaa testitapausten ymmärtämisen ja ylläpitämisen laajemmalle kohdeyleisölle. (Lambdatest, 2022)

Avainsanoihin perustuvaa testausta voidaan yhdistää myös muihin testausmenetelmiin, kuten datalähtöiseen testaukseen tai käyttäytymiseen perustuvaan kehitykseen, jotta voidaan luoda joustavampi ja tehokkaampi testauskehys. Tämä voi auttaa testaajia luomaan ja suorittamaan laajemman valikoiman testejä ja helpommin sieppaamaan ja vahvistamaan testattavan järjestelmän käyttäytymistä. (Lambdatest, 2022)

Avainsanoihin perustuvaa testausta käytetään useissa eri kohteissa, mukaan lukien ohjelmistokehitys, laitteistotestaus ja liiketoiminta-analyysi. Ohjelmistokehityksen yhteydessä avainsanalähtöistä testausta käytetään usein luomaan ja suorittamaan suuri määrä testitapauksia, jotka kattavat

laajan valikoiman skenaarioita ja käyttäjän toimia. Tämä voi auttaa varmistamaan, että sovellus toimii oikein ja pystyy käsittelemään monenlaisia syötteitä ja reunatapauksia. (Lambdatest, 2022)

Avainsanapohjaista testausta käytetään myös laitteistotestauksessa, jossa sen avulla voidaan testata laitteen tai järjestelmän toimivuutta useissa eri olosuhteissa. Esimerkiksi avainsanoihin perustuvaa testausta voitaisiin käyttää uuden laitteen testaamiseen käyttämällä ennalta määritettyjä avainsanoja edustamaan erilaisia käyttäjän toimintoja (esim. napauttamalla painiketta, kirjoittamalla tekstiä, käynnistämällä sovelluksia) ja varmistamalla, että laite vastaa oikein kaikissa tapauksissa. (Lambdatest, 2022)

Liiketoiminta-analyysin yhteydessä avainsanalähtöistä testausta käytetään tietojoukkojen ja liiketoimintamallien tarkkuuden ja luotettavuuden vahvistamiseen. Esimerkiksi avainsanoihin perustuvaa testaustapaa voidaan käyttää taloudellisen ennustemallin testaamiseen käyttämällä ennalta määritettyjä avainsanoja edustamaan erilaisia syöttöskenaarioita (esim. erilaiset markkinaolosuhteet, taloudelliset mittarit ja riskitekijät) ja vertaamaan mallin tuotoksia todellisiin tuloksiin. Tämä voi auttaa tunnistamaan tiedoissa tai mallissa olevat virheet tai harhakohdat ja antaa yritykselle mahdollisuuden tehdä tarkempia ja luotettavampia ennusteita. (Lambdatest, 2022)

Avainsanoihin perustuva testaus on eräänlainen ohjelmistotestaus, jossa testitapaukset määritellään avainsanoilla. Tämä voi olla hyödyllinen lähestymistapa testausjärjestelmiin, koska se mahdollistaa uudelleenkäytettävien testitapausten luomisen ja helpottaa ei-teknisten sidosryhmien ymmärtämistä ja osallistumista testausprosessiin. Se ei kuitenkaan välttämättä ole paras lähestymistapa kaikissa tilanteissa, eikä se välttämättä sovellu monimutkaisempiin tai dynaamisempiin järjestelmiin. On tärkeää harkita tarkasti avainsanalähtöisen testauksen vahvuudet ja rajoitukset suhteessa testattavaan järjestelmään sekä testausprosessin tavoitteet ja resurssit. (Lambdatest, 2022)

Avainsanoihin perustuvaan testaukseen on saatavilla monia erilaisia työkaluja, ja paras tiettyyn tilanteeseen riippuu tekijöistä, kuten testattavan järjestelmän erityisvaatimuksista, testausprosessin tavoitteista sekä testauksen resursseista. Joitakin suosittuja avainsanapohjaisia testityökaluja ovat HPE Unified Functional Testing, Micro Focus UFT Pro, Ranorex, TestComplete ja Robot Framework. Nämä työkalut tarjoavat yleensä joukon ominaisuuksia testitapausten luomiseen ja järjestämiseen, testien suorittamiseen ja tulosten raportointiin. Voi olla hyödyllistä tutkia ja vertailla erilaisia työkaluja löytääksesi sellaisen, joka sopii hyvin haluttuihin testaustarpeisiin.

Robot Framework on avainsanalähtöinen testaustyökalu. Robot Framework on ilmainen ja avoimen lähdekoodin testauskehys, joka käyttää avainsanoja testitapausten määrittämiseen ja testauksen automatisoimiseen. Se tukee useita ohjelmointikieliä, ja sitä voidaan laajentaa mukautetuilla avainsanoilla ja kirjastoilla. Robot Frameworkia voidaan käyttää avainsanalähtöiseen testaukseen

sekä muun tyyppiseen testaukseen, kuten käyttäytymislähtöiseen kehitykseen (BDD) ja hyväksymistestiin perustuvaan kehitykseen (ATDD).

1.5 SSH Library

Robot Frameworkin SSHLibrary on kirjasto vuorovaikutukseen etäpalvelimien ja laitteiden kanssa Secure Shell (SSH) -protokollan kautta. Sen avulla voit suorittaa komentoja etäpalvelimella, siirtää tiedostoja paikallisen ja etäjärjestelmän välillä sekä hallita ja automatisoida erilaisia tehtäviä etäpalvelimella. (Robot Framework, 2022a)

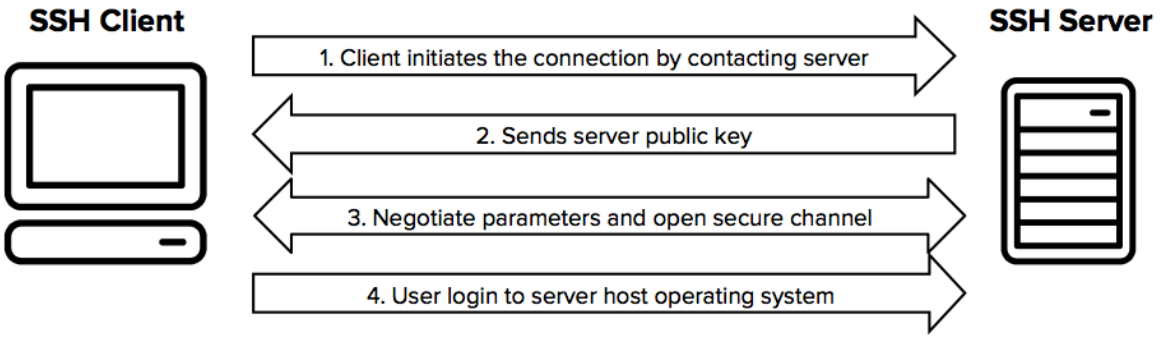
Suoritettavia tehtäviä voivat muun muassa olla, yhdistää etäpalvelimeen käyttäjätunnuksella ja salasanalla tai SSH-avaimella, suorittaa komentoja etäpalvelimella ja palauta tulos, siirtää tiedostoja paikallisen ja etäjärjestelmän välillä käyttämällä erilaisia protokollia (esim. SFTP, SCP), muokata tiedostojen käyttöoikeuksia ja omistajuutta etäjärjestelmässä sekä suorittaa komentosarjat etäjärjestelmässä. SSHLibrary on toteutettu Paramiko-kirjastolla, joka on Python-kirjasto SSH2-protokollayhteyksille. Tämä mahdollistaa sen käytön minkä tahansa järjestelmän kanssa, johon on asennettu Python ja Paramiko, mukaan lukien Windows, Linux ja macOS. (Robot Framework, 2022a)

1.6 SSH

Secure Shell (SSH) -protokolla on verkkoprotokolla suojattua tiedonsiirtoa, etäkomentorivikirjautumista, etäkomentojen suorittamista ja muita suojattuja verkkopalveluita varten kahden verkossa olevan tietokoneen välillä. Se on suunniteltu korvaamaan Telnet- ja rlogin-protokollat, jotka ovat turvattomia ja lähettävät tietoja pelkkänä tekstinä. (SSH, 2022)

SSH käyttää julkisen avaimen salausta ja turvallisia hajautusalgoritmeja salatakseen kahden järjestelmän välillä siirrettävät tiedot, mikä tekee siitä salakuuntelun ja peukaloinnin kestävän. Se tarjoaa myös erilaisia todennusmenetelmiä, kuten salasanoja, biometristä todennusta ja avainpohjaista todennusta digitaalisilla varmenteilla. (SSH, 2022)

SSH-protokolla koostuu kolmesta pääkomponentista, SSH-asiakas, joka on ohjelmisto, joka toimii paikallisessa järjestelmässä ja jonka avulla voit muodostaa yhteyden etäpalvelimeen verkon kautta. (Kuva 5) SSH-palvelin, joka on ohjelmisto, joka toimii etäjärjestelmässä ja kuuntelee SSH-asiakkaiden yhteyksiä. Sekä itse SSH-protokolla, joka määrittelee muodon ja menetelmät datan ja komentojen siirtoon asiakkaan ja palvelimen välillä. (SSH, 2022)



Kuva 5. SSH visuaalinen kuvaus (SSH, 2022)

SSH:ta käytetään laajalti verkkoviestinnän suojaamiseen, ja sitä käytetään yleisesti kirjautumiseen etäpalvelimille, suorittamaan komentoja etäjärjestelmissä ja siirtämään tiedostoja järjestelmien välillä. Sitä käytetään myös luomaan suojattuja tunneleita muille verkkoprotokolleille, kuten Secure Sockets Layer (SSL) ja Virtual Private Network (VPN) yhteyksille. (SSH, 2022)

3. Päätelaite testauksen automatisointi

Päätelaitteeksi toimeksiannossa valittiin geneerinen saatavilla oleva laite ja se oli Compulabin Fitlet2 minitietokone, johon asennettiin Linux Mint Cinnamon 19.1 versio. (Kuva 6) Lisäksi oheislaitteena minitietokoneeseen kytkettiin GPS vastaanotin, Globalsatin BU-353S4. Testit rakennettin ja suoritettiin erilliseltä päätelaitteelta, jossa on Windows 10 käyttöjärjestelmä ja laitteet kytkettiin yksinkertaisen kytkimen välityksellä toisiinsa.

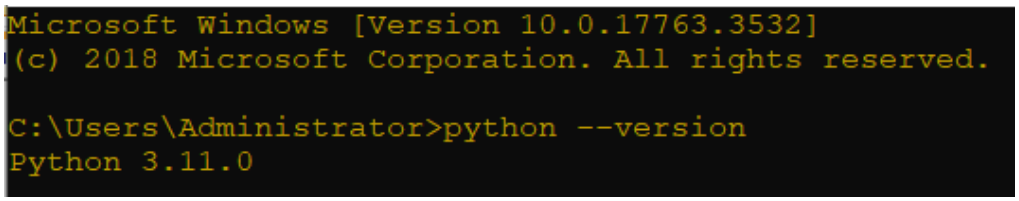
Kuva 6. Päätelaite Compulab Fitlet2 (Copulab Ltd s. a)

1.7 Päätelaitteen asennus

Itse päätelaitteen asennus oli nopea ja yksinkertainen prosessi. Ensin ladattiin haluttu käyttöjärjestelmän levykuva, Linux Mint Cinnamon (v19.1) Windows tietokoneella ja luotiin siitä käynnistettävä USB-media ja sen jälkeen kytkettiin se Fitlet2 tietokoneeseen ja käynnistyksen yhteydessä painamalla F7 päästiin kertakäynnistys valikkoon, josta valittiin USB laite ja sen jälkeen alkoi normaali Linux käyttöjärjestelmän asennusprosessi. Määritettiin pääkäyttäjä sekä tunnistetiedot laitteelle. Asennuksen jälkeen käyttöjärjestelmälle ajettiin tarpeelliset päivitykset sekä asenneltiin tiedossa olevia tarpeellisia paketteja. Asennettavia paketteja oli mm. Python, skriptien ajamiseen, gpsd ja gpsd-clients paketit GPS datan tutkimiseen (gpsd.io 2022) ja lshw laitetietojen keräämiseen (Batool, 2021).

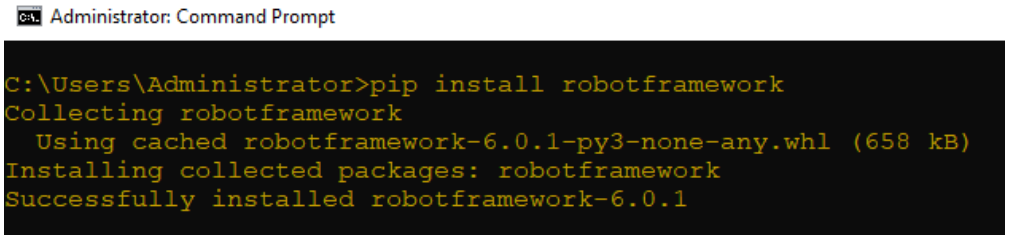
1.8 Robot Frameworkin asennus

Robot Frameworkin asennuksessa on olemassa useita eri tapoja. Esivaatimuksena on kuitenkin, että päätelaitteessa on asennettuna Python. Asennuksen jälkeen Python tulee lisätä Windowsin ympäristömuuttujiin, mikäli tuota ei asennuksen aikana sinne ole jo lisätty. Tämä tulee tehdä siksi että komentokehote osaa etsiä Pythonin tarvittavat komponentit suorittaakseen Python skriptejä. Pythonin version pystyy tarkistamaan halutessaan komentoriviltä komennolla python –version. (Kuva 7)

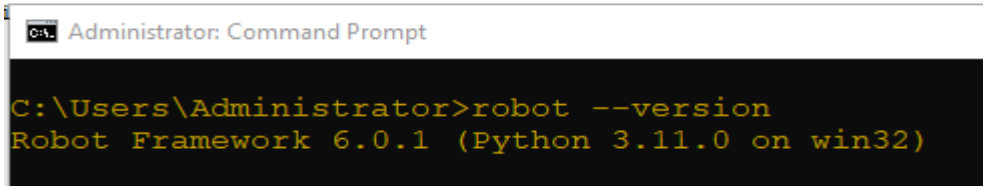


Kuva 7. Python versio tarkistus

Robot Frameworkin asennukseen päästään, kun Python on asennettu ja lisätty ympäristömuuttujiin. Robot Frameworkin asennus onnistuu käyttämällä Pythonin paketinhallintakomentoa pip. Eli Windowsin komentoriviltä suoritetaan komento pip install robotframewrok. (Kuva 8) Sovelluksen voi päivittää viimeisimpään versioon antamalla komento pip install –upgrade robotframewok. Asennuksen jälkeen Windowsin komentoriviltä suorittamalla komento robot –version saadaan varmistus siitä, että ohjelma on asentunut oikein ja mikä versio on kyseessä. (Kuva 9)

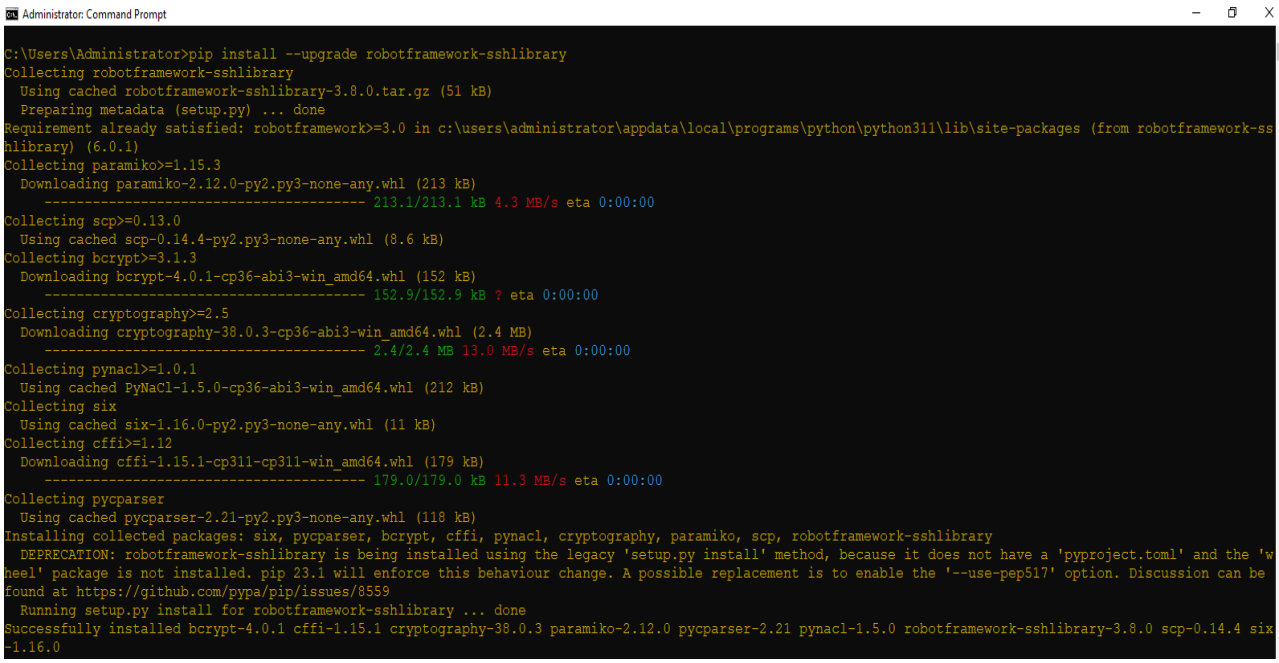


Kuva 8. Robot Frameworkin asennus



Kuva 9. Robot Frameworkin versio

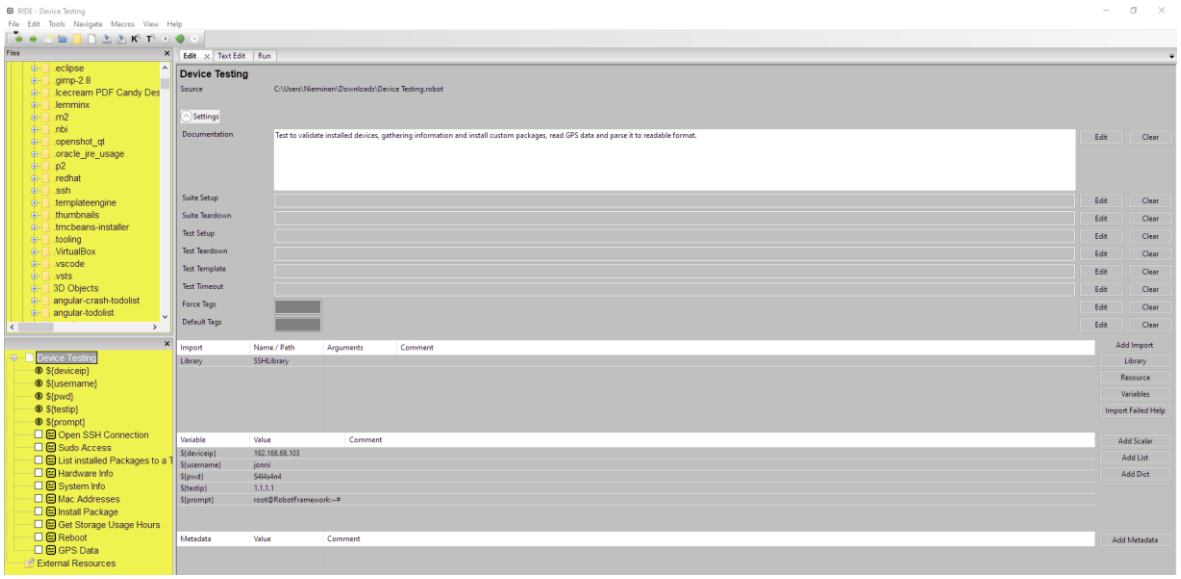
Robot Frameworkin erinäisiä kirjastoja voidaan asentaa suoraa komentoriviltä tässä tapauksessa, kun käytettiin SSHLibraryä niin komento on seuraavanlainen: pip install –upgrade robotframework-sshlibrary. (Kuva 10)



Kuva 10. SSHLibraryn asennus

Robot Frameworkin käyttämiseen on monia eri työkaluja, tässä tapauksessa käyttöön valikoitui RIDE. Ride on Robot Frameworkin integroitu kehitysympäristö. Integrated Development Environment (IDE) on ohjelmistosovellus, joka tarjoaa tietokoneohjelmoijille kattavat mahdollisuudet koodin kirjoittamiseen ja virheenkorjaukseen. RIDE:n käytön voi aloittaa joko komentoriviltä kirjoittamalla komennoksi ride.py tai sitten vaihtoehtoisesti suoraan pikakuvakkeesta RIDE.

RIDE:n käyttöliittymässä on selkeä kansionäkymä ja testit eriteltynä vasemmassa reunassa. (Kuva 11) Lisäksi RIDE tarjoaa graafisen käyttöliittymän avainsanojen käyttöön sekä koodieditorin.



Kuva 11. RIDE käyttöliittymä

1.9 Päätelaitteen testaus

Päätelaitteen testaaminen aloitettiin ensin testattavan laitteen käynnistämisellä ja sieltä tarvitsee etsiä IP-osoite, joka sille on määritelty. Tämä IP-osoite sitten annettiin Robot Frameworkin muuttujiin tässä tapauksessa ${deviceip} muuttujaksi. Tunnistetiedot, tulee nekin määritellä etukäteen ${username} ja ${password} muuttujiin, sekä jokin mahdollinen yhteyspiste, johon voidaan kokeilla verkkoyhteyden toimintaa. Testilaite kun oli julkisessa verkossa niin testattava IP osoite oli Cloudflaren DNS palvelin 1.1.1.1. (Koodi 2) Vaihtoehtoisesti jos ollaan suljetussa ympäristössä, niin voidaan käyttää esimerkiksi verkon gateway IP-osoitetta tai muuta vastaavaa.

```
*** Settings ***
Documentation     Test to validate installed devices, gathering information and install
custom packages, read GPS data and parse it to readable format.
Library           SSHLibrary

*** Variables ***
${deviceip}       192.168.68.103
${username}       jonni
${pwd}            S4l4s4n4
${testip}         1.1.1.1
${prompt}         root@RobotFramework:~#
```

Koodi 2. Asetukset ja muuttujat

Testitapaukset aloitetaan SSH yhteyden muodostamisella kohde laitteeseen. Tähän käytetään avainsanaa Open Connection. Avainsanalle tulee antaa laitteen IP osoite ja tarvittaessa portti, mutta tässä testitapauksessa hyödynnettiin SSH yhteyden oletusporttia 22. Kirjautuminen tunnistetiedoilla ja varmistetaan linux komennolla hostname laitteen nimi, johon ollaan yhteydessä ja asetetaan se testin viestiin. (Koodi 3)

```
*** Test Cases ***
Open SSH Connection
    Open Connection    ${deviceip}
    Login    ${username}    ${pwd}    delay=1
    ${hostname}    Execute Command    hostname
    Set Test Message    Connection established to ${hostname} (${deviceip})
```

Koodi 3. SSH yhteyden avaus

Seuraavassa vaiheessa korotetaan käyttöoikeudet root tasolle ja varmistetaan pwd komenolla nykyinen hakemisto, että ollaan root käyttäjän hakemistossa. Robot Frameworkissa sudo oikeudet tulee antaa vielä erillisinä argumentteina sudo=True ja sudo_password. (Koodi 4)

```
Sudo Access
    Execute Command    sudo -s    #Grant Sudo Access
    Write    ${pwd}
    ${output}    Execute Command    id    sudo=True    sudo_password=${pwd}    #Print
Linux 'id' output, which tells you current user
    Set Test Message    ${output}
```

Koodi 4. Käyttöoikeuksien korotus

Tämän jälkeen listattiin kohde laitteeseen asennetut ohjelmistopaketit. Pakettien määrä, kun on melko suuri Linux Mintin kokoisessa käyttöjärjestelmässä niin parhaaksi vaihtoehdoksi tässä valikoitui tulostaa pakettilistaus erilliseen tekstitiedostoon ja hakea sieltä muutama esimerkki mitä laitteeseen on asennettuna. (Koodi 5)

```
List installed Packages to a TXT file
    ${packagecount}    Execute Command    dpkg --get-selections >> package_list.txt |
wc -l package_list.txt | grep -o '[0-9]'    #List packages to a text file
    ${checkpackages}    Execute Command    head -10 package_list.txt
    ${listofpackages}    Execute Command    cat package_list
    Set Test Message    Firt 10 installed Packages, check rest from package_list.txt \n
${checkpackages}
```

Koodi 5. Asennettujen pakettien listaus

Sitten laitteesta haettiin laitteisto tietoja, prosessori, muisti, käynnissä oloaika sekä näytönohjaimen tiedot. Nämä tiedot täytyi eritellä isosta määrästä tietoa egrep, sed, cut ja head komentoja käyttäen, jotta saadaan testin viesti osioon kirjoitettua selkeästi haluttua tietoa. (Koodi 6)

```
Hardware Info
    ${cpu}=    Execute Command    cat /proc/cpuinfo | egrep 'cpu cores|model name' |
sed 's/:[[:blank:]]*/: /' | sort -r | uniq    #CPU Specification
    ${meminfo}=    Execute Command    cat /proc/meminfo | egrep 'MemTotal' | sed
's/:[[:blank:]]*/: /'    #Memory Specification
    ${uptime}=    Execute Command    uptime | cut -d' ' -f4-5 | head --bytes -2    #Uptime
    ${gpu}=    Execute Command    lspci | grep VGA | cut -d ' ' -f 5-10    #GPU Specifica-
tion
    Set Test Message    CPU=${cpu}\nMemory=${meminfo}\nGPU=${gpu}\nUp-
time=${uptime}
```

Koodi 6. Laitteistotietojen noutaminen

Tämän jälkeen laitteesta haettiin sarjanumero, valmistaja ja laitteen nimi dmidecode -t system komennolla ja siitä eriteltiin grep toiminnolla halutut kentät. Lisäksi omana testinään haettiin verkkokorttien MAC osoitteet ifconfig komennolla ja sieltä eriteltiin tiettyä muotoa oleva merkkijono grep komennolla. (Koodi 7)

```
System Info
    ${sysinfo}=    Execute Command    dmidecode -t system | grep -E 'UUID|Serial Num-
ber|Manufacturer|Product Name' | sed 's/:[[:blank:]]*/: /'    sudo=True    sudo_pass-
word=${pwd}    #Get Serialnumber
    Set Test Message    ${sysinfo}

Mac Addresses
    ${macaddress}=    Execute Command    ifconfig | grep -o -E
'([[:xdigit:]]{1,2}:){5}[[:xdigit:]]{1,2}'    #List every Interface MAC Address
    Set Test Message    ${macaddress}
```

Koodi 7. Laitetietojen noutaminen

Seuraavana vaiheena oli edessä paketin asennus. Tässä kohdepakettina oli Smart Monitor Tools paketti, jolla oli tarkoitus kerätä kiintolevyn käyttötunnit, koska historiassa esiintyneiden tapausten kohdalla on ollut laitteita, joissa levyn ohjelmistoversion bugi aiheuttaa laitteiston vikaantumisen 32768 tunnin jälkeen. Asennuksen jälkeen varmistetaan vielä, että smartmontools niminen paketti on installed tilassa. (Koodi 8)

```
Install Package
    Execute Command    sudo apt install smartmontools    sudo=True    sudo_pass-
word=${pwd}    #Install package and check if its installed after
    Set Test Message    Smart Monitor Tools package installed
    ${installcheck}    Execute Command    apt list --installed | grep "smartmontools"
    Should End With    [installed]    [installed]
```

Koodi 8. Pakettien asennus

Ja viitaten edellisen vaiheen paketin valinta kriteereihin, haettiin laitteistosta kiintolevyn käyttötunnit smartmontools pakettia hyödyntäen. Tulosta pitää taas siistiä käyttäen cut komentoa, jotta saadaan testituloksiin kirjattua vain tarpeellinen tieto. (Koodi 9)

```
Get Storage Usage Hours
    ${usagehours}=    Execute Command    smartctl --all /dev/sda | grep
Power_On_Hours | cut -d' ' -f4    sudo=True    sudo_password=${pwd}
    Set Test Message    Storage usage hours: ${usagehours}    #Output is available only
for supported devices on smartctl database
```

Koodi 9. Levyn käyttötuntien noutaminen

Tässä vaiheessa testiä oli uudelleen käynnistyksen vaihe, joka onnistuu root käyttäjän oikeuksin antamalla laitteelle komento reboot. Robot Framework asetettiin Sleep tilaan 45 sekunniksi ja sen jälkeen kokeilemaan Open Connection avainsanan suorittamista niin kauan että komentokehotteen ulosanti antaa merkin $. (Koodi 10)

```
Reboot
    Execute Command    reboot    sudo=True    sudo_password=${pwd}    #Reboots sys-
tem, wait 2 min to get response from Open Connection
    Sleep    45 sec
    Wait Until Keyword Succeeds    2 min    15 sec    Open Connection    ${deviceip}
prompt=$
    Login    ${username}    ${pwd}
    Set Test Message    Device reboot successfully
```

Koodi 10. Laitteen uudelleen käynnistäminen

GPS vastaanottimen testaus toteutettiin yksinkertaisella Python skriptillä, (Pythonkoodi 1) jossa luetaan USB porttiin liitetyn GPS vastaanottimen tulostamaa dataa sieltä haarukoidaan $GPGGA formaatissa oleva data (Kuva 12) ja se viedään tekstitiedostoon. Tekstitiedostossa oleva data käsitellään pynmea2 python kirjaston avulla, joka on kirjasto NMEA0183 protokollalle. (Pythonkoodi 2) Näin GPS vastaanottimen data saadaan luettavaan muotoon (Kuva 13)

```
#!/usr/bin/python3.6
import serial
import time
import pynmea2
import re

def readgps():
    ser = serial.Serial('/dev/ttyUSB0', 4800, timeout=60)

    # Printataan rivejä N kappaletta
    line_count = 0

    # Määritellään whilelooppi ja montako riviä tulostetaan
    while line_count < 3:
        try:
            line = bytes(ser.readline())
            line = line.decode('utf-8')
            if line.startswith('$GPGGA'):
                print(line, end='')
                line_count += 1
        except Exception as e:
            print(e)

    # Katkaistaan yhteys
    ser.close()

readgps()
```

Pythonkoodi 1. GPS datan sieppaaminen

```
$GPGGA,134857.120,6044.6828,N,02445.4171,E,1,03,3.0,81.5,M,19.7,M,,0000*66
$GPGGA,134858.120,6044.6861,N,02445.4176,E,1,03,3.0,81.5,M,19.7,M,,0000*63
$GPGGA,134859.120,6044.6867,N,02445.4184,E,1,03,3.0,81.5,M,19.7,M,,0000*69
```

Kuva 12: Käsittelemätön GPS data

```
#!/usr/bin/python3.6
import pynmea2
import datetime

try:
    with open('gps_data.txt') as log:
        for line in log:
            gps = pynmea2.parse(line)
            # print(repr(gps))
            ts = str(gps.timestamp)
            lat = str(gps.lat)
            lon = str(gps.lon)
            lon_dir = str(gps.lon_dir)
            lat_dir = str(gps.lat_dir)
            sat = str(gps.num_sats)
            qual = str(gps.gps_qual)
            print('| Time: ' + ts + ' UTC ' + ' | GPS quality FIX = ' + qual + ' | Latitude:  ' + lat +
' ' + lat_dir +  ' |  Longi$except Error as e:
    print(e)
```

Pythonkoodi 2. GPS datan käsittely

```
| Time: 13:48:57.120000 UTC  | GPS quality FIX = 1 | Latitude:  6044.6828 N |  Longitude: 02445.4171 E | Using 03 satelites |
| Time: 13:48:58.120000 UTC  | GPS quality FIX = 1 | Latitude:  6044.6861 N |  Longitude: 02445.4176 E | Using 03 satelites |
| Time: 13:48:59.120000 UTC  | GPS quality FIX = 1 | Latitude:  6044.6867 N |  Longitude: 02445.4184 E | Using 03 satelites |
```

Kuva 13. Käsitelty GPS data

1.10 Lopputulos ja raportointi

Lopputuloksena Robot Frameworkin onnistuneista testeistä saatiin raportti, jossa käy ilmi mitä testejä kohdelaitteelle on suoritettu, sekä halutut tiedot testituloksista. Raportin ”Test Message” kenttään on asetettu halutut tiedot jokaisesta testistä, näin ollen syntyy tarvittavan kattava raportti mikä säilötään tarpeen mukaan.

Raportin Test Message kenttään tuotiin mm. yhteyden muodostamisen onnistumisviesti, asennettujen pakettien listausta sekä tiedostonimi, johon lista tulostettu täydellisesti. Lisäksi laitteesta tuotiin hardwaretietoa, niin prosessorin, muistien, näytönohjaimen osalta sekä laitteen valmistaja, sarjanumero, UUID ja käynnissä oloaika (uptime). (Kuva 14)

Verkkomääritysten osalta laitteistosta haettiin jokaisen verkkosovittimen osalta fyysinen MAC-osoite, sekä laitteella tehtiin yhteystesti ping komennolla tässä tapauksessa Cloudfaren DNS palvelimeen, joka on muotoa 1.1.1.1. Muissa tapauksissa voidaan tarvittaessa käyttää verkkoinfrastruktuurin yhdyskäytäviä tai palvelimia vastaamaan ping komentoon. Laitteistotiedot sekä verkkotestin tulokset tuotiin raportille näkyviin.

GPS tietoa haettaessa testin suorituspaikan mukaan saatiin useampaan otteeseen kohdistaa ja parannella GPS vastaanottimen yhteyttä, mutta kun laitteesta saatiin tarpeeksi dataa, niin pystyttiin suorittamaan tiedonkeräys, sen siistiminen ja tulostaminen luettavaan ja ymmärrettävään formaattiin. Tämä siistitty tieto sitten tulostettiin myös raportille.

Testituloksia pyrittiin validoimaan mahdollisimman monia sen mukaan, kun tiedetään mitä onnistuneiden testien tulisi tulostaa niin näitä ehtoja lisättiin myös testitapausten loppuun mm. Should Contain, Should Match Regexp, Should End With ja Wait Until Keyword Succeeds. Testien saatujen tulosten automaattista varmistusta voi tehdä varmasti monella eri tavallakin, mutta tämä todettiin riittäväksi näissä tapauksissa.

Status	Message
PASS	Connection established to RobotFramework (192.168.68.103)
PASS	uid=0(root) gid=0(root) groups=0(root)
PASS	Firt 10 installed Packages, check rest from /home/jonnipackage_list.txt accountsservice install acl install acpi-support install acpid install add-apt-key install adduser install adobe-flashplugin install adwaita-icon-theme install alsa-base install alsa-utils install
PASS	CPU=model name : Intel(R) Celeron(R) CPU J3455 @ 1.50GHz cpu cores : 4 Memory=MemTotal: 16244788 kB GPU=Intel Corporation Device 5a85 (rev 0b) Uptime=2 days
PASS	Manufacturer: Compulab Product Name: fitlet2 Serial Number: 1190411-00211 UUID: 26B0C100-6FC1-11E9-990B-94171BD00500
PASS	00:01:c0:24:a3:3b 00:01:c0:23:da:e9 00:01:c0:24:a3:34 00:01:c0:23:db:10
PASS	PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data. 64 bytes from 1.1.1.1: icmp_seq=1 ttl=58 time=20.5 ms 64 bytes from 1.1.1.1: icmp_seq=2 ttl=58 time=19.5 ms 64 bytes from 1.1.1.1: icmp_seq=3 ttl=58 time=16.4 ms 64 bytes from 1.1.1.1: icmp_seq=4 ttl=58 time=17.5 ms --- 1.1.1.1 ping statistics --- 4 packets transmitted, 4 received, 0% packet loss, time 3005ms rtt min/avg/max/mdev = 16.485/18.541/20.567/1.614 ms
PASS	Smart Monitor Tools package installed
PASS	Storage usage hours: Power_On_Hours
PASS	Device reboot successfully
PASS	| Time: 13:48:57.120000 UTC | GPS quality FIX = 1 | Latitude: 6044.6828 N | Longitude: 02445.4171 E | Using 03 satelites | | Time: 13:48:58.120000 UTC | GPS quality FIX = 1 | Latitude: 6044.6861 N | Longitude: 02445.4176 E | Using 03 satelites | | Time: 13:48:59.120000 UTC | GPS quality FIX = 1 | Latitude: 6044.6867 N | Longitude: 02445.4184 E | Using 03 satelites |

Kuva 14. Testiraportti

4. Pohdinta

Robot Frameworkin soveltuvuus erinäisiin päätelaitteita koskeviin testeihin todettiin meidän tapauksessamme aivan riittäväksi. Yrityksestä kun löytyy ohjelmointiosaamistakin melkoisesti, pystymme tarvittaessa luomaan ja jalostamaan jo olemassa olevia kirjastoja käyttöömme.

Avainsanojen avustuksella testien suunnitteleminen ja kirjoittaminen kävi helposti, sekä hahmotteleminen ei vielä alkutaipaleella tuottanut haasteita, kun ei tarvinnut miettiä miten mikäkin testi teknisesti toimii. SSHLibraryn ja Linuxin komennoilla kokonaisuus saatiin yksinkertaiseen formaattiin melko nopeasti.

Testien kirjoittamisessa nousi kuitenkin esille se, että testit sisältävät melkoisen paljon monimutkaisia Linux-komentosarjoja ja näin ollen, mikäli testissä havaitaan virheitä niin niiden selvittely vaatii mahdollisesti lisäresursseja sitten ohjelmointitaustaiselta henkilöltä tai vähintäänkin Linuxia ymmärtävältä taholta. Kuitenkin testin käyttötarkoituksena on jatkokehittää sitä ja sen jälkeen jalkauttaa tuotantoon niin tässä vaiheessa voitaisiin hankalimmat testitapaukset kirjoittaa erilliseen kirjastoon ja näin ollen vain kutsua tarvittavaa testiä ja pois sulkea se mahdollisuus, että testien vaikeimmat komennot vahingossa sotkettaisiin. Esimerkkinä mainittakoon, vaikka paketin asennus, niin luodaan ”apt install” komennosta avainsana ja kirjataan muuttujiin haluttuja paketteja ja sitten kutsutaan testissä tätä esim. Install Package ${chrome}

Listaus laitteeseen asennetuista paketeista halutaan jatkoa ajatellen listata, koska kyseessä tulee olemaan kustomoitu ja rajattu ympäristö Linux käyttöjärjestelmästä niin tällä voidaan tarkistaa, että kaikki on asentunut laitteeseen niin kuin halutaan.

Eniten haasteita testien toteuttamisessa aiheutui GPS datan keräämisestä, sekä sen siistimisestä. Tietoa GPS datasta sai toki ulos useammallakin Linux käyttöjärjestelmän ohjelmistolla, mutta se että siitä saatiin muokattua haluttuun formaattiin ja ymmärrettävään muotoon, niin jouduin turvautumaan Python-ohjelmointikielen skripteihin, jonka avulla tietoa käsiteltiin. Valmiista GPS ohjelmista voidaan mainita, vaikka cgps, jota yritettiin käyttää ensin, mutta sen tulostamaa dataa ei saatu Robot Frameworkille helposti pihalle.

Myös testattavan laitteen kiintolevyn käyttötuntien lukemiseen joutui kuluttamaan jonkin verran aikaa ennen kuin oli todettava, että tähän ei nyt tällä erää saada ratkaisua, kun käytettävä ohjelmisto vaatisi laitteiston osien kirjaamisen heidän tietokantaansa. (Kuva 23) Tämä tulee jatkoa ajatellen tarkistaa laite ja komponentti kohtaisesti, että onko tätä järkeä pitää osana testiä, mikäli käytössä oleva laitekanta ei löydy kyseisetä tietokannasta.

```
=== START OF INFORMATION SECTION ===
Device Model:     M2SCF-6M 256GB
Serial Number:    20190320017310100048
LU WWN Device Id: 0 000000 000000000
Firmware Version: P0810C
User Capacity:    253 403 070 464 bytes [253 GB]
Sector Size:      512 bytes logical/physical
Rotation Rate:    Solid State Device
Form Factor:      M.2
Device is:        Not in smartctl database [for details use: -P showall]
ATA Version is:   ACS-2 (minor revision not indicated)
SATA Version is:  SATA 3.1, 6.0 Gb/s (current: 6.0 Gb/s)
Local Time is:    Fri Jan 13 15:49:11 2023 EET
SMART support is: Available - device has SMART capability.
SMART support is: Enabled
```

Kuva 23. Laitetiedot, ”Not in smartctl database”

Testausympäristöjä ajatellen, jatkokehitys vaiheessa robotille täytyy määritellä asennettavien sovellusten pakettitiedostot vietäväksi testaustyöasemalta kohdekoneelle SCP komentoa hyödyntäen, niin ei kohdelaitteiston tarvitse olla yhteydessä erilliseen repositoryyn tai internettiin paketteja asentaakseen. Tai vaihtoehtoisesti kun laitteita asennetaan usein erilliseltä asennuspisteeltä Miradore järjestelmän kautta, jossa laitteet käynnistetään PXE käynnistykseltä ja tarvittava levykuva asennetaan asennuspalvelimelta, niin tässä yhteydessä osana prosessia voitaisiin viedä tarvittavat asennuspaketit laitteelle.

5. Lähteet

Batool, S. 2021. Linux lshw Command Luettavissa: https://linuxhint.com/use-linux-lshw-command/ Luettu: 24.1.2023
Compulab Ltd. s.a https://fit-iot.com/web/products/fitlet2/

CRG Solutions, 2020. Robotic Process Automation (RPA) in infrastructure management. Luettavissa: https://www.crgsolutions.co/blogs/robotic-process-automation-rpa-in-infrastructure-management/ Luettu 24.1.2023

gpsd.io, 2022. gpsd — a GPS service daemon Luettavissa: https://gpsd.io/ Luettu: 24.1.2023

Hunter, A. 2022. Discover Remote Infrastructure Management. Luettavissa: https://www.parallels.com/blogs/ras/remote-infrastructure-management/ Luettu: 25.1.2023

Lambdatest, 2022. Keyword Driven Testing Tutorial: A Comprehensive Guide with Examples and Best Practices. Luettavissa: https://www.lambdatest.com/learning-hub/keyword-driven-testing Luettu: 16.12.2022.

Lozowska-Pereira, M. 2022. Why and When to Use Automation in testing. Luettavissa: https://www.netguru.com/blog/automation-in-testing-importance Luettu: 24.1.2023

Neova Tech Solutions Inc, 2022. API Automation Testing using Robot Framework with Python. Luettavissa: https://www.neovasolutions.com/2022/08/04/api-automation-testing-using-robot-framework-with-python/ Luettu: 2.1.2023.

Robot Framework, 2022a. SSHLibrary. Luettavissa: http://robotframework.org/SSHLibrary/SSHLibrary.html Luettu 22.12.2022

Robot Framework, 2022b. Robot Framework User Guide. Luettavissa: https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html Luettu: 22.12.2022

SSH Communications Security, 2022. SSH (Secure Shell). Luettavissa: https://www.ssh.com/academy/ssh Luettu 22.12.2022

Stresnjak & Hocenski. ICSEA 2011: The Sixth International Conference on Software Engineering Advances. Luettavissa: http://personales.upv.es/thinkmind/dl/conferences/icsea/icsea_2011/icsea_2011_2_10_10057.pdf Luettu 2.1.2023

6. Liitteet

Liite 1. Robot Framework .robot tiedoston lähdekoodi

```
*** Settings ***
Documentation     Test to validate installed devices, gathering information and install custom packages, read GPS data and parse it to readable format.
Suite Teardown    Close All Connections
Library           SSHLibrary

*** Variables ***
${deviceip}       192.168.68.103
${username}       jonni
${pwd}            S4l4s4n4
${testip}         1.1.1.1
${prompt}         root@RobotFramework:~#

*** Test Cases ***
Open SSH Connection
    Open Connection    ${deviceip}
    Login    ${username}    ${pwd}    delay=1
    ${hostname}    Execute Command    hostname
    Set Test Message    Connection established to ${hostname} (${deviceip})

Sudo Access
    Execute Command    sudo -s    #Grant Sudo Access
    Write    ${pwd}
    ${output}    Execute Command    id    sudo=True    sudo_password=${pwd}    #Print Linux 'id' output, which tells you current user
    Should Contain    ${output}    uid=0(root)
    Set Test Message    ${output}

List installed Packages to a TXT file
    ${packagecount}    Execute Command    dpkg --get-selections >> package_list.txt | wc -l package_list.txt | grep -o '[0-9]'    #List packages to a text file
    ${checkpackages}    Execute Command    head -10 package_list.txt
    ${listofpackages}    Execute Command    cat package_list
    ${listlocation}    Execute Command    pwd
    Set Test Message    Firt 10 installed Packages, check rest from ${listlocation}package_list.txt \n${checkpackages}
```

```
Hardware Info
    ${cpu}=    Execute Command    cat /proc/cpuinfo | egrep 'cpu cores|model name' | sed 's/:[[:blank:]]*/: /' | sort -r | uniq    #CPU Specification
    ${meminfo}=    Execute Command    cat /proc/meminfo | egrep 'MemTotal' | sed 's/:[[:blank:]]*/: /'    #Memory Specification
    ${uptime}=    Execute Command    uptime | cut -d' ' -f4-5 | head --bytes -2    #Uptime
    ${gpu}=    Execute Command    lspci | grep VGA | cut -d ' ' -f 5-10    #GPU Specification
    Set Test Message    CPU=${cpu}\nMemory=${meminfo}\nGPU=${gpu}\nUptime=${uptime}

System Info
    ${sysinfo}=    Execute Command    dmidecode -t system | grep -E 'UUID|Serial Number|Manufacturer|Product Name' | sed 's/:[[:blank:]]*/: /'    sudo=True    sudo_password=${pwd}    #Get Serialnumber
    Set Test Message    ${sysinfo}

Check MAC Addresses
    ${macaddress}=    Execute Command    ifconfig | grep -o -E '([[:xdigit:]]{1,2}:){5}[[:xdigit:]]{1,2}' >> macaddresses.txt    #List every Interface MAC Address
    ${checkmac}=    Execute Command    cat macaddresses.txt | grep -o -E '([[:xdigit:]]{1,2}:){5}[[:xdigit:]]{1,2}' | head -n 1
    Should Match Regexp    ${checkmac}    ^[a-fA-F0-9]{2}(:[a-fA-F0-9]{2}){5}$
    Set Test Message    ${macaddress}

Ping Test
    ${pingresult}    Execute Command    ping -c 4 ${testip}    #Ping against testip and take 4 rows of results
    Should Contain    ${pingresult}    64 bytes from    #Verifies that Ping result give output as it should
    Set Test Message    ${pingresult}

Install Package
    Execute Command    sudo apt install smartmontools    sudo=True    sudo_password=${pwd}    #Install package and check if its installed after
    Set Test Message    Smart Monitor Tools package installed
    ${installcheck}    Execute Command    apt list --installed | grep "smartmontools"
    Should End With    [installed]    [installed]

Get Storage Usage Hours
    ${usagehours}=    Execute Command    smartctl --all /dev/sda | grep Power_On_Hours | cut -d' ' -f4    sudo=True    sudo_password=${pwd}
    Set Test Message    Storage usage hours: ${usagehours}    #Output is available only for supported devices on smartctl database
```

```
Reboot
    Execute Command    reboot    sudo=True    sudo_password=${pwd}    #Reboots system, wait 2 min to get response from Open Connection
    Sleep    45 sec
    Wait Until Keyword Succeeds    2 min    15 sec    Open Connection    ${deviceip}    prompt=$
    Login    ${username}    ${pwd}
    Set Test Message    Device reboot successfully

GPS Data
    Execute Command    ./testgps.py >> gps_data.txt    #Run Python Script to get GPS data to a txt file
    ${cleangps}    Execute Command    ./readgps.py    #Run Python Script to print GPS data in a clean understandable format
    ${gps_data}=    Execute Command    cat gps_data.txt | head -n 1
    Should Contain    ${gps_data}    $GPGGA,
    Should Contain    ${gps_data}    ,N,
    Should Contain    ${gps_data}    ,E,
    Set Test Message    ${cleangps}
```