Tampereen yliopisto

Jaakko Kääriäinen

ELUA JA MICROPYTHON SULAUTETUISSA JÄRJESTELMISSÄ

Informaatioteknologian ja viestinnän tiedekunta
Kandidaatintutkielma
Huhtikuu 2020

TIIVISTELMÄ

Jaakko Kääriäinen: eLua ja MicroPython sulautetuissa järjestelmissä
Kandidaatintutkielma
Tampereen yliopisto
Tieto- ja sähkötekniikka, TkK
Huhtikuu 2020

Sulautetut järjestelmät ovat käytössä monessa arkipäivän laitteessa. Ne on suunniteltu suorittamaan vain yhteen käyttötarkoitukseen tarkoitettuja toimenpiteitä. Yleisin ohjelmointikieli sulautetussa ohjelmoinnissa on C-kieli, joka ei välttämättä ole aloittelevan ohjelmoijan ensimmäinen ohjelmointikieli. Tässä työssä tutkittiin eLuaa ja MicroPythonia, jotka on suunniteltu sulautettuun ohjelmointiin. Ne ovat uudelleentoteutuksia Lua- ja Python-ohjelmointikielistä. Tavoitteena oli selvittää, kumpi kielistä on käytännöllisempi sulautetussa ohjelmoinnissa. Tutkimus suoritettiin käyttämällä kirjallisia lähteitä.

Työssä on kaksi osaa. Ensimmäinen osa on kirjallisuustutkimus, jossa esitellään työssä tutkittavien kielten taustat ja ominaisuudet. Sen jälkeen käsitellään sulautettujen järjestelmien piirteitä, ohjelmointia ja muutamia sovelluksia. Lopuksi esitellään kielten toteutuksia sulautetuille järjestelmille. Toisessa osassa vertaillaan toteutusten ominaisuuksia ja esitetään niiden perusteella tutkimuksen johtopäätös. Vertailukriteereiksi käytännöllisyydessä nähdään totetusten helppokäyttöisyys tuotannossa, prosessoriarkkitehtuurien tuki, ohjelmistomoduulien saatavuus ja vähimmäismuistinkäyttö.

Tutkimuksen tuloksena MicroPython osoittautui käytännöllisemmäksi kieleksi kuin eLua. Pythonin laajan yhteisötuen ansiosta MicroPython toimii monessa prosessoriarkkitehtuurissa ja sillä on kattava laajennuksien tarjonta. MicroPython käyttää myös vähemmän RAM-muistia, joka on yksi niukoista resursseista sulautetuissa järjestelmissä. eLuan vahvempi puoli on kuitenkin kattavampi sulautetun ohjelmoinnin moduulivalikoima sisäänrakennettuna kieleen, mikä antaa valmiudet useille sulautettujen järjestelmien sovelluksille.

Avainsanat: eLua, Lua, MicroPython, Python, sulautettu ohjelmointi, sulautetut järjestelmät

SISÄLLYSLUETTELO

KUVALUETTELO

TAULUKKOLUETTELO

OHJELMA- JA ALGORITMILUETTELO

LYHENTEET JA MERKINNÄT

ARM	Advanced RISC Machine (entinen Acorn RISC Machine) on RISC-arkkitehtuuriin pohjautuvien prosessoreiden perhe.
CWI	Centrum Wiskunde & Informatica
I/O	Input/Output
IoT	Internet of Things
JTAG	Joint Test Action Group
LAN	Local-area network
Linux	Unix-kaltaisten käyttöjärjestelmien perhe, joiden toteutus perustuu Linux-ytimeen.
MIDI	Musical Instrument Digital Interface
MIT	Massachusetts Institute of Technology
PUC-Rio	Pontifícia Universidade Católica do Rio de Janeiro
RAD	Rapid Application Development
RAM	Random-access memory
REPL	Read–Eval–Print Loop. Interaktiivinen tulkki, joka ottaa vastaan syötteen, minkä jälkeen tulkki suorittaa sen ja tulostaa syötteen tulokset.
RISC	Reduced Instruction Set Computer
ROM	Read-only memory

1 JOHDANTO

Sulautetut järjestelmät ovat pienikokoisia tietokoneita, jotka on ohjelmoitu ja suunniteltu suorittamaan tiettyjä toimenpiteitä yhtä käyttötarkoitusta varten. Täten ne eivät ole yleiskäyttöisiä tietokoneita, kuten kannettava tietokone. Esimerkkinä sulautetusta järjestelmästä voidaan pitää ilmastointijärjestelmää, joka ylläpitää huoneen lämpötilaa. [1] Yleisin ohjelmointikieli sulautettujen järjestelmien ohjelmoinnissa on C-kieli, koska se antaa ohjelmoijalle suoran pääsyn laitteistoon uhraamatta korkean tason kielen ominaisuuksia [2]. C-kieli ei välttämättä ole jokaiselle ohjelmoijalle tuttu tai helppo kieli, minkä vuoksi on olemassa muita ohjelmointikieliä sulautetulle ohjelmoinnille.

Tämä työ tutkii Luan ja Pythonin eLua- ja MicroPython-toteutuksia sulautetuille järjestelmille. Tutkimus on toteutettu käyttäen aiheeseen liittyviä kirjallisia lähteitä. Lähteiden avulla selvitetään, miten ja milloin Lua ja Python ovat käytännöllisiä kieliä sulautettujen järjestelmien ohjelmoinnissa. Käytännöllisyydeksi ymmärretään toteutusten helppokäyttöisyys tuotannossa, prosessoriarkkitehtuurien tuki, ohjelmistomoduulien saatavuus ja vähimmäismuistinkäyttö.

Luvussa 2 esitetään Lua- ja Python-kielten taustat ja ominaisuudet. Luku 3 käsittelee sulautettujen järjestelmien piirteitä ja niiden sovelluksia. Luvussa 4 tarkastellaan kielten toteutuksia sulautetuille järjestelmille. Luvussa 5 vertaillaan toteutusten ominaisuuksia, jotka määrittelevät kielen käytännöllisyyden sulautetussa ohjelmoinnissa. Vertailun jälkeen esitetään tulokset samassa luvussa. Luku 6 muodostaa yhteenvedon.

2 TARKASTELTAVIEN KIELTEN ESITTELY

Tässä luvussa käsitellään työssä tutkittavien kielten historiaa, toteutusta ja yleisiä käyttökohteita. Sulautetut järjestelmät jätetään tässä käsittelemättä, sillä niitä tarkastellaan omassa luvussaan.

2.1 Lua

Lua on proseduraalinen laajennusohjelmointikieli, joka tukee proseduraalisen ohjelmoinnin lisäksi muita paradigmoja: olio-ohjelmointia sekä funktionaalista ja data-pohjaista ohjelmointia. Lua on suunniteltu käytettäväksi ohjelmistoissa, jotka vaativat tehokkaan ja kevyen skriptikielen. [3] Lua-tulkin lähdekoodi on kirjoitettu C-kielellä, mikä sallii Lua-ohjelmien ajamisen monessa eri laiteympäristössä [4]. Luan pääominaisuuksiin kuuluvat seuraavat ominaisuudet [3, 5]:

- dynaaminen tyypitys
- vuorottaisrutiinit
- automaattinen/manuaalinen muistinsiivous
- merkkijonojen hallinta
- optimoitu häntärekursio
- lambda-funktiot
- metataulukot.

Lua kehitettiin PUC-Rion yliopistossa, ja sen ensimmäinen versio julkaistiin vuonna 1993. Kieltä on käytetty yleisesti videopeleissä, sulautetuissa järjestelmissä sekä muissa ohjelmissa ja palveluissa, kuten Adobe Photoshop Lightroomissa ja Wikipediassa. [4, 6] Luan toteutus sisältää vain 25 000 koodiriviä, ja sen binäärikoko on 200 kilotavua 64-bittisessä Linuxissa. [6] Lua on lisensoitu versiosta 5.0 eteenpäin MIT-lisenssillä [7].

Lua-kielessä taulukot ovat heterogeenisiä hakurakenteita, mikä tarkoittaa sitä, että taulukon indeksiksi kelpaavat numeroiden lisäksi muuntyyppiset arvot `nil`-arvoa lukuunottamatta. Arvo `nil` esittää puuttuvaa arvoa. [3]

Jokainen arvo sisältää metataulukon, joka määrittelee arvon käyttäytymisen operaattorien kanssa. Metataulukoita käsitellään kielessä samalla tavalla kuin tavallisia taulukoita. Arvojen käyttäytymistä muutetaan muuttamalla siihen kuuluvan metataulukon kenttiä, jotka sisältävät metodin. Jokainen avain metataulukossa on muotoiltu seuraavalla tavalla:

kaksi alaviivaa __ ja operaattorin nimi. Esimerkiksi lisäysoperaattorin metodi metataulukossa sijaitsee avaimessa `__add`. [3]

2.2 Python

Python on olio-ohjelmointikieli, joka tarjoaa tehokkaita tietorakenteita sekä käyttää dynaamista tyypitystä. Dynaamisen tyypityksen lisäksi kielen selkeä syntaksi tekee Pythonista käytännöllisen kielen ohjelmistokehityksessä, joka noudattaa nopean kehityksen mallia (RAD). Python-tulkin toiminnallisuutta voidaan täydentää lisäosilla, jotka on toteutettu C- tai C++-kielellä. [8] Python-kielen toiminnallisuutta pystyy laajentamaan pip-pakkauksenhallintajärjestelmällä, joka asentaa laajennukset sille määritellyistä repositorioista [9].

Guido van Rossum kehitti Pythonin 90-luvun alussa CWI:ssä Hollannissa. Vuonna 2001 perustettiin voittoa tavoittelematon yhdistys Python Software Foundation, jonka tarkoituksena on ylläpitää sekä Python-kieltä että sen yhteisöä. [10, 11] Koska Python on yleiskäyttöinen ohjelmointikieli, sitä sovelletaan monella eri alalla mukaan lukien sulautetut järjestelmät [12].

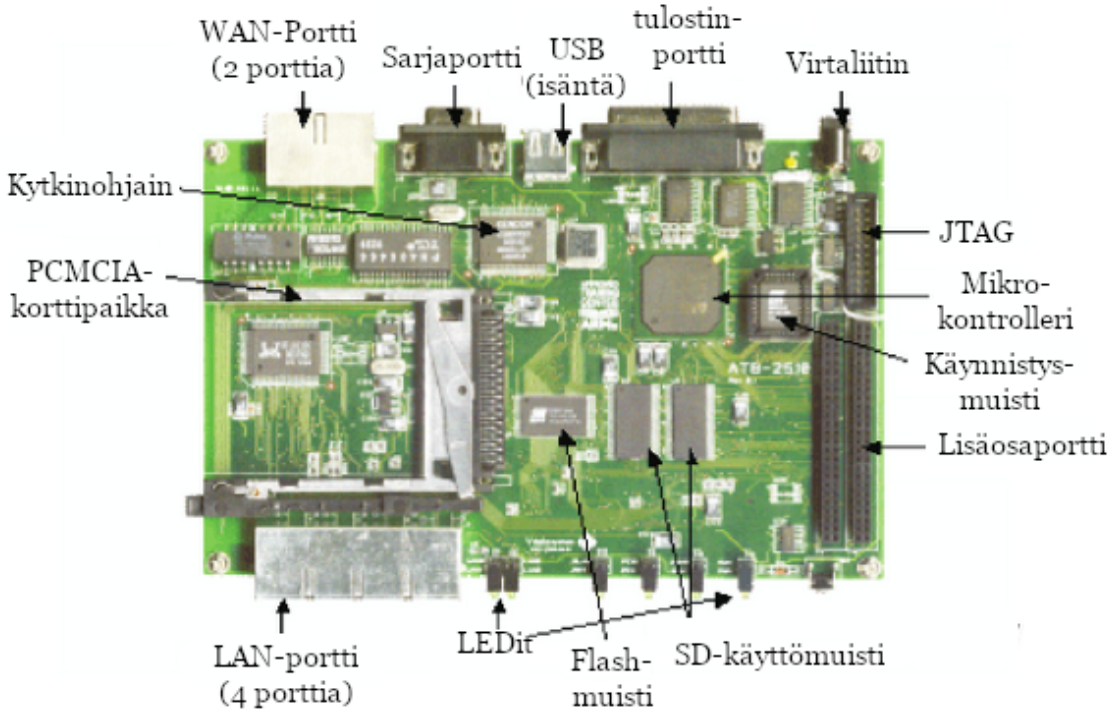
Pythonista on olemassa kaksi eri versiota: 2 ja 3. Edellisen version tuki päättyi vuoden 2020 alussa, joten sille ei tarjota enää päivityksiä [13]. Tämän vuoksi tässä työssä viitataan jatkossa Pythonin versioon 3.

3 SULAUTETUT JÄRJESTELMÄT

Tässä luvussa tutustutaan sulautettujen järjestelmien piirteisiin, ohjelmointiin ja niiden sovelluksiin. Luvussa keskitytään niihin ominaisuuksiin, joiden osalta sulautetut järjestelmät eroavat tavallisista järjestelmistä.

3.1 Piirteet

Sulautetun järjestelmän peruskomponentit ovat mikrokontrolleri, erilaiset I/O-portit ja käyttömuisti (kuva 3.1). Mikrokontrolleri sisältää ohjelmamuistin sekä osoite- ja dataväylät järjestelmän muihin komponentteihin. Porttien tarkoituksena on lähettää ja vastaanottaa dataa muilta komponenteilta ja oheislaitteilta. [1]



Kuva 3.1. *Esimerkki sulautetusta järjestelmästä (suomennettu) [1].*

Sulautetun järjestelmän tarkoitus on suorittaa yhtä tehtävää toistuvasti. Muihin piirteisiin kuuluvat rajoitetut järjestelmäresurssit, reaktiivisuus sekä monimutkaisten algoritmien suorittaminen. Käyttömuistin määrä on rajoitettu, ja mikrokontrollerin mikroprosessorin kellotaajuus on pienempi tavalliseen kotitietokoneeseen verrattuna. Reaktiivisuudella tar-

koitetaan järjestelmän kykyä reagoida odottamattomiin tapahtumiin, esimerkiksi auton jarrutukseen. Järjestelmän sovelluksesta riippuen mikroprosessori joutuu suorittamaan monimutkaisia algoritmeja järjestelmälle määriteltyjen tehtävien suorittamiseksi. [1]

Sulautetut järjestelmät jakaantuvat kolmeen eri kategoriaan: pienet, keskisuuret ja kehittyneet sulautetut järjestelmät. Kategorioihin jako perustuu niiden suunnittelun monimutkaisuuteen, valmistuskustannuksiin sekä käyttövaatimuksiin. Pieniä järjestelmiä ei ole suunniteltu raskaisiin laskennallisiin tehtäviin, mutta niille ohjelmointi on yksinkertaisempaa kehittyneisiin järjestelmiin verrattuna. [1]

3.2 Ohjelmointi

Sulautetussa ohjelmoinnissa työskennellään lähellä laitteistotasoa. Ohjelmoijan pitää olla tietoinen laitteen ominaisuuksista ja siihen liitetyistä oheislaitteista, jotta hän voi kirjoittaa laitteelle tarvittavan laiteohjelmiston. Laiteohjelmiston kirjoittamisessa ja testaamisessa käytetään yleensä apuna mikropiirien kytkentäkaavioita ja oskillokooppia. [2]

Laiteohjelmiston suunnittelussa tulee ottaa huomioon laitteen rajoitukset muistin ja nopeuden näkökulmista. Jokaisella koodirivillä on merkitystä, sillä jokainen rivi vie prosessorilta suoritusaikaa. Muistinhallinnallakin tulee huolehtia, ettei laitteen muisti lopu kesken laiteohjelmiston ajoaikana. [2]

Laiteohjelmisto on erikoistunut, sillä laitteen jokaisella komponentilla on oma tapansa käsitellä aktiviteetteja. Komponenttien välisen kommunikoinnin toteuttaminen on tärkeää, jotta saadaan kaikki hyöty irti laitteen hallitsemisesta. Suurin osa laiteohjelmiston koodista keskittyy komponenttien kommunikointiin. [2]

3.3 Sovellukset

Yleisimmät sovellukset sulautetuille järjestelmille löytyvät arkipäivän laitteista, kuten mikroaaltouuneista, pankkiautomaateista, modeemeista tai laskimista [1]. Lähiaikoina sulautettujen järjestelmien merkitys on muuttunut merkittävästi IoT-laitteiden (esineiden internet) mainonnan vuoksi, mikä puolestaan on lisännyt mielenkiintoa IoT:n tarjoamien mahdollisuuksien tutkimiseen. [14]

IoT-laitteet ovat fyysisiä esineitä, jotka voidaan kytkeä Internetiin. Nämä laitteet voivat vuorovaikuttaa ulkopuoliseen ympäristöön Internetin välityksellä. Esimerkiksi älykotien turvajärjestelmät, valaistus ja muu elektroniikka pohjautuvat IoT-tekniikkaan. [15]

Usein sulautettu järjestelmä toimii komponenttina suuremmassa kokonaisuudessa. Autot sisältävät useamman kuin yhden sulautetun järjestelmän toimiakseen. Esimerkiksi yksi järjestelmistä ohjaa auton jarrujen toimintaa, toinen monitoroi ja ohjaa auton päästöjä ja kolmas näyttää tietoja kojelaudalla. [1]

Sulautettuja järjestelmiä hyödynnetään myös mekatroniikan alalla. Yleistä mekatroniikan sovelluksissa on anturien hyödyntäminen mittatiedon noutamisessa ja käsittelyssä jär-

jestelmän toimintalogiikan mukaisesti. Esimerkiksi robotiikassa niitä hyödynnetään polunseurannassa ja konenäössä. [16]

4 KIELTEN TOTEUTUKSET SULAUTETUILLE JÄRJESTELMILLE

Tässä luvussa käsitellään niitä kielten toteutuksia, jotka on suunniteltu juuri sulautettuja järjestelmiä varten. Näistä toteutuksista on joko poistettu kielten tarpeettomat ominaisuudet tai niihin on lisätty uusia ominaisuuksia, jotka ovat hyödyllisiä ohjelmistojen kehittämisessä sulautetuille järjestelmille. Jokaisen toteutuksen kohdalla esitellään yksi käyttökohde sekä laite, joka hyödyntää alaluvussa käsiteltävää toteutusta.

4.1 eLua

eLua (Embedded Lua) on kokonainen toteutus Lua-kielestä, johon on lisätty sulautettuun ohjelmointiin tarkoitettuja ominaisuuksia. Ominaisuudet on suunniteltu siten, että ne käyttävät mahdollisimman vähän muistia ja ovat tehokkaampia Luan ydinominaisuuksiin verrattuna. eLua ei ole sidottu tiettyyn laiteympäristöön, mikä helpottaa laiteohjelmiston kehittämistä usealle eri ympäristölle. eLua ei ole käyttöjärjestelmässä ajettava ohjelma, vaan se pyörii *bare-metal*-ympäristössä, mikä tarkoittaa sitä, että eLualla on täysi kontrolli alustan toiminnan hallitsemisessa. [5] eLuan vähimmäisvaatimuksiin kuuluvat 32-bittinen keskusyksikkö, 256 kilotavua flash-muistia ja 64 kilotavua käyttömuistia [17]. Yksi eLuaa käyttävistä sovelluksista on ohjelmistokehys MIDI-laitteiden ja ohjaimien protokollalle [18].

eLua on kehitetty ja testattu monelle ARM-arkkitehtuurin mikrokontrollerille, mutta eLua tukee myös x86-arkkitehtuuria. eLuan moduulien tuki eri mikrokontrollereissa kuitenkin vaihtelee, mikä tulee ottaa huomioon, jos kehittää laiteohjelmistoa samalla monelle muulle mikrokontrollerille. [19] eLuaa voi käyttää esimerkiksi Texas Intrumentsin EK-LM3S9B92-laitteessa (kuva 4.1). Laite kytketään USB-johdolla tietokoneeseen, minkä jälkeen eLuan lähdekoodin viimeisin koontiversio voidaan konfiguroida laitteen laiteympäristön mukaisesti ja siirtää se laitteen ROM-muistin tiedostojärjestelmään. Jos `autorun.lua`-tiedosto löytyy tiedostojärjestelmästä, se ajetaan automaattisesti laitteen käynnistyessä. Kyseinen tiedosto toimii laiteohjelmiston suorituksen lähtökohtana. Kun laite on käynnistynyt, sen komentotulkkiin pääsee käsiksi Telnet-protokollalla. [17]

Kuva 4.1. *Texas Instruments EK-LM3S9B92 [17].*

Taulukko 4.1. *Lista eLuan moduuleista ja niiden tiloista (suomennettu) [19].*

nimi	kuvaus	tila
pio	Ohjelmoitava I/O	valmis
tmr	Fyysiset ja virtuaaliset ajastimet	valmis
pwm	Pulssinleveysmodulaatio	valmis
uart	Universaali asynkroninen lähetin-vastaanotin	valmis
spi	Serial Peripheral Interface	valmis
net	TCP/IP-pino	valmis
adc	Analogia–digitaalimuunnin	valmis
dac	Digitaali–analogiamuunnin	ei toteutettu
cpu	Matalan tason pääsy järjestelmään	valmis
pd	Alustadata	valmis
term	Pääsy ANSI-päätteeseen	valmis
bit	Bittioperaatiot	valmis
pack	Binääridatan pakkaus/purku	valmis
cmp	Analoginen komparaattori	ei toteutettu
i2c	Integroitujen piirien protokolla	valmis
cnt	Tapahtumalaskuri	ei toteutettu
can	CAN-väylä	testaus kesken
mrpc	Etäproseduurikutsu	valmis
i2s	Integroitujen piirien äänimoduuli	ei toteutettu
elua	eLua järjestelmäohjaus	valmis

Taulukko 4.2. *Lista alustariippuvaisista eLuan moduuleista ja niiden tiloista (suomennettu) [19].*

nimi	kuvaus	alustat	tila
lm3s.disp	OLED-näyttötuki	EK-LM3S8962, EK-LM3S6965	valmis
str9.pio	Laajennettu I/O-konfiguraatio	STR-E912, STR9-comStick	valmis
mbed.pio	Laajennettu I/O-konfiguraatio	mbed	valmis
str9.rtc	Reaaliaikainen kello	STR-E912, STR9-comStick	valmis

Taulukosta 4.1 nähdään eLuan yleiset ohjelmistomoduulit ja niiden kehityksen tila. Osa moduuleista ei ole toteutettu tai niiden testaus on kesken. Ei-toteutettujen moduulien kehitys on kuitenkin suunniteltu jatkossa. eLuan alustariippuvaiset moduulit on listattu taulukossa 4.2. Nämä moduulit toimivat vain niille määritetyissä alustoissa. [19]

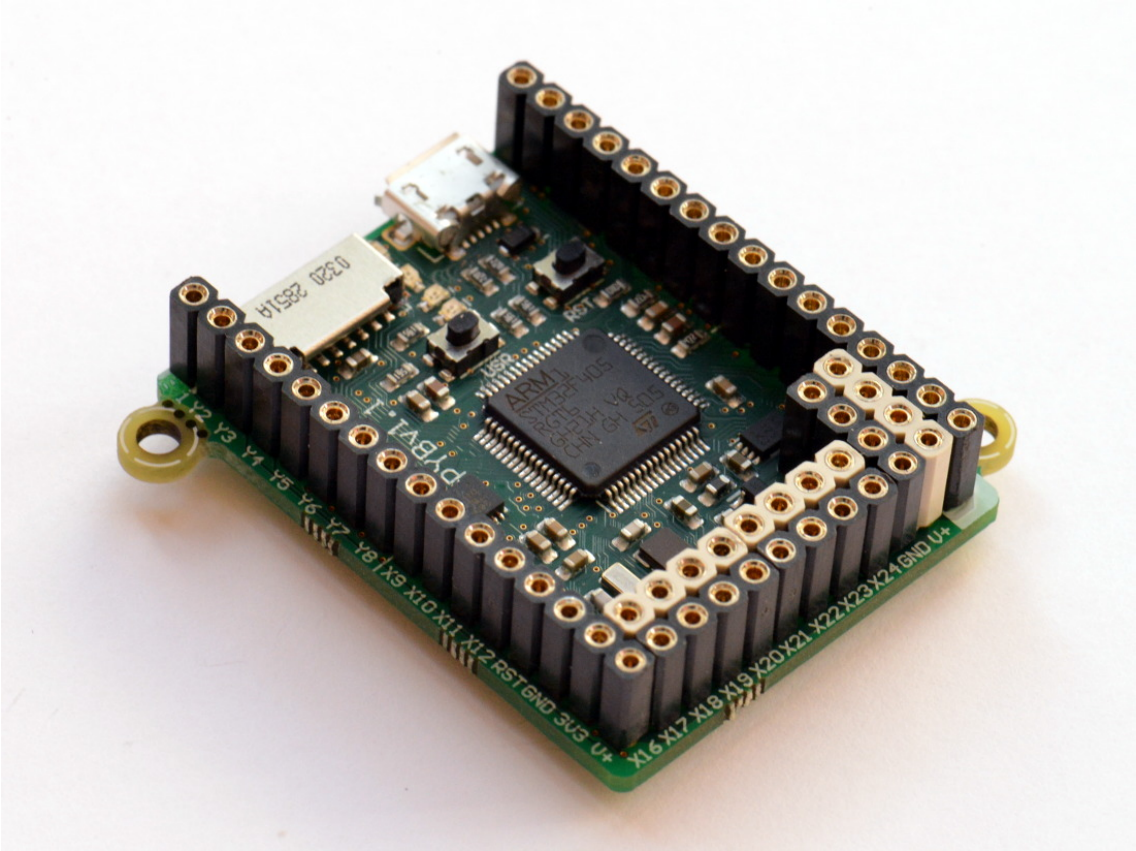
4.2 MicroPython

MicroPython on uudelleentoteutettu versio Python-kielestä. MicroPython on suunniteltu mikrokontrollereita ja sulautettuja järjestelmiä varten. Koska MicroPython toimii monessa eri laitteessa, kielen ominaisuusvalikoima saattaa vaihdella (yleensä Pythonin vakiokirjaston osajoukko on saatavilla). MicroPython ajetaan *bare-metal*-ympäristössä. [14] MicroPythonin vähimmäislaitteistovaatimuksina ovat 256 kilotavua ROM-muistia sekä 16 kilotavua käyttömuistia [20]. MicroPython tukee seuraavia prosessoriarkkitehtuureja: x86, x86-64, ARM, ARM Thumb ja XTensa [21]. Eräs MicroPythonia hyödyntävistä sovelluksista on ohjelmoinnin opetukseen suunniteltu liikkuva robotti (ohjelma 4.1) [22].

```
1 while True:
2     while (robot.sendUltra(1)) > 10:
3         robot.forward_always()
4     robot.sendBuzzer(1,3)
5     pyb.delay(1000)
6     robot.sendBuzzer(0,3)
7     robot.stop()
```

Ohjelma 4.1. *Esimerkkiohjelma robotin ohjaamiseen [22].*

Ensimmäinen valmistettu laite, joka tukee MicroPythonia, on Pyboard (kuva 4.2). Laitteen Flash-muisti sisältää laitteen tiedostojärjestelmän. Jos se sisältää tiedoston `main.py`, MicroPython suorittaa sen laitteen käynnistyessä. Pyboardin tallennustilaa voidaan tarvittaessa täydentää micro SD -kortilla. Pyboard voidaan yhdistää micro USB -kaapelilla tietokoneelle, jolloin sen kooditiedostoja voidaan muokata tai suorittaa sille Python-komentoja REPL-päätteen kautta. [14]



Kuva 4.2. *MicroPython Pyboard v1.1 [14].*

MicroPythonissa osa Python-kirjastojen toiminnoista on karsittu jättäen vain kirjastojen ydinominaisuudet. Taulukosta 4.3 nähdään lista kavennetuista kirjastoista. Osa kirjastojen nimistä sisältää u-etuliitteen, joka ilmaisee, että kyseinen kirjasto on mikrokirjasto. Tämä nimeämiskäytäntö antaa käyttäjälle mahdollisuuden itse toteuttaa Python-tasoisen moduulin paremman yhteensopivuuden takaamiseksi CPythonin kanssa. [23] CPython on viitetoteutus Python-kielestä, johon moni muu Python-toteutus perustuu [24]. Taulukko 4.4 näyttää listan MicroPythonin omista moduuleista [23].

Taulukko 4.3. *Lista Pythonin moduuleista, joiden toiminnallisuutta on kavennettu MicroPythonissa (suomennettu) [23].*

nimi	kuvaus
cmath	Matemaattiset funktiot kompleksiluvuille
gc	Automaattinen roskienkeräys
math	Matemaattiset funktiot
sys	Järjestelmäkeskeiset funktiot
uarray	Taulukot numeeriselle datalle
uasyncio	Asynkroninen I/O-ajastin
ubinascii	Binääri/ASCII-muuntimet
ucollections	Kokoelma- ja säiliötietorakenteet
uerrno	Järjestelmävirhekoodit
uhashlib	Hajautusalgoritmit
uheapq	Kekojonoalgoritmi
uio	I/O-virrat
ujson	JSON-koodaus ja -dekoodaus
uos	Käyttöjärjestelmän peruspalvelut
ure	Yksinkertaiset säännölliset lausekkeet
uselect	Tapahtumien odotus useista datavirroista
usocket	Verkkopistokemoduuli
ussl	SSL/TLS-moduuli
ustruct	Tietotyyppien pakkaus/purku
utime	Ajankäsittelyyn liittyvät funktiot
uzlib	Zlib-purku
_thread	Rinnakkaisuustuki

Taulukko 4.4. *Lista Micropythonin moduuleista (suomennettu) [23].*

nimi	kuvaus
btree	Yksinkertainen binäärinen hakupuutietorakenne
framebuf	Kuvapuskurin käsittely
machine	Laitteistofunktiot
micropython	MicroPythonin sisäisen datan käsittely
network	Verkon konfigurointi
ubluetooth	Matalan tason Bluetooth ohjaus
ucryptolib	Salausalgoritmit
uctypes	Jäsennetyn binääridatan luku

MicroPythonin tekniset ominaisuudet tarjoavat ketterän ja käyttäjäkeskeisen tavan sulautettuun kehitykseen. Python-ohjelmoijat voivat hyödyntää aikaisempaa osaamista sulautetuissa järjestelmissä. Esimerkiksi web-kehittäjä voi oppia sulautetun ohjelmoinnin periaatteet nopeasti oppimatta täysin uutta ohjelmointikieltä. Pythonin yksinkertainen syntaksi ja nopea sovelluskehitys tekevät MicroPythonista hyvän valinnan aloitteleville ohjelmoijille ja alan ammattilaisille. [14]

5 ELUAN JA MICROPYTHONIN VERTAILU

Tässä luvussa pohditaan niitä eLuan ja MicroPythonin eroavaisuuksia ja yhtäläisyyksiä, joiden avulla voidaan päätellä niiden käytännöllisyys. Aluksi verrataan toteutusten helppokäyttöisyyttä tuotannossa, ja sen jälkeen muistinkäyttöä, ohjelmistomoduulien saatavuutta ja arkkitehtuuritukea. Toteutusten asentamisen helppoutta sulautetuille järjestelmille ei verrata tässä työssä, koska asennusprosessi riippuu paljon kulloisenkin kohdejärjestelmän ominaisuuksista.

Molemmat toteutukset toimivat samalla tavalla, kun laite käynnistetään. Ne etsivät käynnistystiedoston, joka toimii sulautetun järjestelmän toiminnan lähtökohtana. MicroPythonin toiminnallisuutta pystyy laajentamaan pip-työkalulla, jolla voidaan asentaa laajennuksia Python-kieleen. eLualla ei kuitenkaan ole vastaavanlaista ominaisuutta, koska itse Lua-kielellä ei ole olemassa standardia pakkauksenhallintajärjestelmää.

Taulukoiden 4.1 ja 4.4 perusteella voidaan päätellä, että eLualla on enemmän sulautettuun ohjelmointiin tarkoitettuja ohjelmistomoduuleja kuin MicroPythonilla, mikä tekee eLuasta sopivan monille sulautettujen järjestelmien sovelluksille. MicroPython kuitenkin sisältää Pythonin ohjelmistokirjastoja (taulukko 4.3), jotka lisäävät Pythonin ydintoiminnallisuuden MicroPythoniin, mikä helpottaa Python-ohjelman siirtämistä MicroPythoniin.

Taulukosta 5.1 nähdään, että molemmat kielet käyttävät saman määrän ROM-muistia, mutta MicroPython käyttää vähemmän RAM-muistia eLuaan verrattuna. Lisäksi MicroPython tukee enemmän prosessoriarkkitehtuureja kuin eLua, mikä tekee MicroPythonista sekä muistiekonomisen että yhteensopivan vaihtoehdon sulautetulle ohjelmoinnille monessa eri alustassa.

Taulukko 5.1. *Toteutuksien vähimmäisjärjestelmävaatimukset ja arkkitehtuurituki.*

kieli	ROM (kt)	RAM (kt)	arkkitehtuurituki
eLua	256	64	x86, ARM
MicroPython	256	16	x86, x86-64, ARM, ARM Thumb, XTensa

Vertailukriteerien perusteella MicroPython on käytännöllisempi kieli kuin eLua sulautetussa ohjelmoinnissa. MicroPythonilla on laajempi yhteisötuki, minkä vuoksi se toimii monella alustalla ja sille on myös olemassa monia Python-yhteisön tarjoamia laajennuksia. eLua tarjoaa enemmän ohjelmistomoduuleja sulautettuun ohjelmointiin, mutta siltä puuttuvat yleishyödylliset funktiot, jotka ovat valmiina MicroPythonissa. eLuan korkeampi RAM-muistin käyttö saattaa johtua Lua-tulkin muistinhallinnan oletuskäyttäytymisestä.

6 YHTEENVETO

Työssä vertailtiin eLuaa ja MicroPythonia sulautetuille järjestelmille. Kyseiset ohjelmointikielet ovat uudelleentoteutuksia Luasta ja Pythonista. Vertailun tavoitteena oli selvittää kumpi kieli on käytännöllisempi sulautetussa ohjelmoinnissa. Vertailukriteereiksi valittiin niiden käytännöllisyys tuotannossa, ohjelmistomoduulien saatavuus, vähimmäismuistinkäyttö ja arkkitehtuurituki. Vertailun pohjana käytettiin aiheeseen liittyviä kirjallisuuslähteitä.

Vertailun johtopäätöksenä MicroPython osoittautui käytännöllisemmäksi kieleksi kuin eLua. Johtopäätöstä tukevat laaja Python-yhteison tuki, pienempi vähimmäiskäyttömuistivaatimus sekä laajempi arkkitehtuurituki eLuaan verrattuna. eLuan kattavampi sulautetun ohjelmoinnin moduulivalikoima kuitenkin kattaa monen eri sulautetun järjestelmän sovellusvaatimukset.

Näyteohjelman kirjoittaminen molemmilla kielillä samoille laiteympäristöille ja ohjelman suorittaminen voisivat antaa lisätietoa kielten välisistä suorituskykyeroista. Suorituskykyeroihin kuuluvat ohjelman suoritusaika ja tulkin muistinhallinta ajoaikana. Kielten suorituskykyä voitaisiin myös mitata tarkastelemalla tulkkien suorituskykyä hyödyntäen erilaisia suoritustestejä.

LÄHTEET

[1] Kothari, D. P. *Embedded systems*. New Delhi: New Age International, 2012. ISBN: 81-224-3498-3.

[2] Barr, M. *Programming embedded systems : with C and GNU development tools*. 2nd ed. Beijing: O'Reilly. ISBN: 0-596-51912-5.

[3] Ierusalimschy, R., De Figueiredo, L. ja Celes, W. *Lua 5.1 Reference Manual*. Lua.Org, 2006. ISBN: 8590379833.

[4] Jung, K. *Beginning Lua programming*. Wrox beginning guides. Indianapolis, IN: Wiley Pub., 2007. ISBN: 1-280-74112-0.

[5] eLua Project. *Overview - eluaproject*. 23. helmikuuta 2020. URL: http://www.eluaproject.net/overview (viitattu 23. 02. 2020).

[6] Ierusalimschy, R., De Figueiredo, L. ja Celes, W. A Look at the Design of Lua. *Association for Computing Machinery. Communications of the ACM* 61.11 (2018), 114–123. ISSN: 00010782. URL: http://search.proquest.com/docview/2133823510/.

[7] *Lua: license*. 1. kesäkuuta 2019. URL: https://www.lua.org/license.html (viitattu 31. 01. 2020).

[8] *The Python Tutorial — Python 3.8.1 documentation*. 31. tammikuuta 2020. URL: https://docs.python.org/3/tutorial/index.html (viitattu 31. 01. 2020).

[9] *pip · PyPI*. 21. maaliskuuta 2020. URL: https://pypi.org/project/pip/ (viitattu 21. 03. 2020).

[10] *History and License — Python 3.8.1 documentation*. 26. tammikuuta 2020. URL: https://docs.python.org/3/license.html (viitattu 26. 01. 2020).

[11] *Python Software Foundation | Python Software Foundation*. 2. helmikuuta 2020. URL: https://www.python.org/psf/ (viitattu 02. 02. 2020).

[12] Kalb, I. *Learn to Program with Python*. 1st ed. 2016. Berkeley, CA: Apress, 2016. ISBN: 1-4842-2172-9.

[13] *Sunsetting Python 2 | Python.org*. 17. maaliskuuta 2020. URL: https://www.python.org/doc/sunset-python-2/ (viitattu 17. 03. 2020).

[14] Tollervey, N. H. *Programming with MicroPython : embedded programming with microcontrollers and Python*. Beijing, 2017.

[15] Stroud, F. *Internet of Things (IoT) Definition Webopedia*. 28. maaliskuuta 2020. URL: https://www.webopedia.com/TERM/I/internet_of_things.html (viitattu 28. 03. 2020).

[16] Tao, Y., Tan, J., Shao, Z. ja Wei, H. Mechatronics and Embedded Systems. *Advances in Mechanical Engineering* 6.2014 (2014), 2. ISSN: 1687-8132.

[17] *Running Embedded Lua On Microcontrollers | Electronic Design*. 8. lokakuuta 2013. URL: https://www.electronicdesign.com/technologies/dev-tools/article/21798490/running-embedded-lua-on-microcontrollers (viitattu 19. 03. 2020).

[18] *eLuaMIDI - eLua Wiki*. Helmikuu 2010. URL: http://web.archive.org/web/20170625013731/http://wiki.eluaproject.net/eLuaMIDI (viitattu 19. 03. 2020).

[19] *eLua Doc*. 2. huhtikuuta 2015. URL: http://www.eluaproject.net/doc/v0.8/en_index.html (viitattu 19. 03. 2020).

[20] *Introduction to MicroPython*. 1. maaliskuuta 2020. URL: https://www.digikey.fi/en/maker/blogs/2018/introduction-to-micropython (viitattu 01. 03. 2020).

[21] *MicroPython - Python for microcontrollers*. 20. maaliskuuta 2020. URL: https://micropython.org/ (viitattu 20. 03. 2020).

[22] Khamphroo, M., Kwankeo, N., Kaemarungsi, K. ja Fukawa, K. MicroPython-based educational mobile robot for computer coding learning. *2017 8th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)*. IEEE, 2017, 1, 6. ISBN: 9781509048090.

[23] *Overview — MicroPython 1.12 documentation*. 27. maaliskuuta 2020. URL: https://docs.micropython.org/en/latest/index.html (viitattu 27. 03. 2020).

[24] Shaw, A. *Your Guide to the CPython Source Code – Real Python*. 21. elokuuta 2019. URL: https://realpython.com/cpython-source-code-guide/#part-1-introduction-to-cpython (viitattu 28. 03. 2020).