OpenCV Raspberry Pi ympäristössä

Kohteen tunnistus väriin pohjautuen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Sähkö- ja automaatiotekniikan opinnäytetyö

Sähkö- ja automaatiotekniikka, insinööri (AMK)

Kevät 2021

Sami Juusonen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Sähkö- ja automaatiotekniikan AMK Tiivistelmä

HAMK Valkeakoski

Tekijä	Sami Juusonen	Vuosi 2021
Työn nimi	OpenCV Raspberry Pi -ympäristössä	
Ohjaajat	Juhani Henttonen	

TIIVISTELMÄ

Projektin tarkoituksena on esitellä OpenCV-konenäkökirjaston toimintaa ja madaltaa aloituskynnystä konenäköjärjestelmien rakentamisen suhteen. Projektin alkuperäinen tarkoitus oli tunnistaa kohde värinperusteella ja lähettää kohteen sijaintikoordinaatit sarjaväylään. Toimilaitteella on tarkoitus lukea koordinaatit väylästä sekä suorittaa kohteen poiminta annetuista koordinaateista, kuitenkin kohteen poiminta vapaasti osoittautui toimilaitteen tarkkuuden vuoksi liian haastavaksi ja lopullisessa projektissa kohde poimitaankin vain yhdestä sijainnista. Ohjelma on kuitenkin rakennettu niin, että kohteen sijainti kerrotaan toimilaitteelle ajatellen myöhempiä sovelluksia.

Työntuloksena on koodilistaukset sekä niiden dokumentaatio yhdessä havaittujen haasteiden kanssa, näiden on tarkoitus tarjota aloituspiste ja toimia tietolähteenä tulevissa OpenCV-projekteissa.

Ohjelmointikielenä toimivat Python 3.7 ja Arduino-kieli ja alustoina Raspberry Pi 2 Model B sekä Arduino Uno.

Projektilla ei ole toimeksiantajaa, vaan se pohjautuu omaan kiinnostukseen ja harrastuksiin.

Avainsanat Konenäkö, OpenCV, Ohjelmointi, Python, Raspberry Pi

Sivut 27 sivua ja liitteitä 16 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Electrical and Automation Engineering Bachelor´s Degree **Abstract**

HAMK Valkeakoski

Author Sami Juusonen Year 2021

Subject OpenCV Raspberry Pi ympäristössä

Supervisors Juhani Henttonen

ABSTRACT

The purpose of the project is to present the operation of the OpenCV machine vision library and to lower the start threshold for building machine vision systems. The original purpose of the project was to identify the object by color, to send the location coordinates of the object to the serial bus. The purpose of the actuator is to read the coordinates from the bus and to pick up the object from the given coordinates, however, picking up the object freely proved to be too challenging due to the accuracy of the actuator and in the final project the object will be picked from only one location. However, the program has built so that the location of the target is been told to the actuator for later applications.

The programming languages are Python 3.7 and Arduino, and the platforms are Raspberry Pi 2 Model B and Arduino Uno.

The project does not have a commissioner, but it based on one's own interests and hobbies.

Keywords Computer vision, OpenCV, Programming, Python, Raspberry Pi

Pages 27 pages and appendices 16 pages

Sisällys

Kuvat, taulukot ja kaavat

Liitteet

LYHENTEET JA MÄÄRITELMÄT

CCD

Charge-Coupled Device on kuvasensorityyppi.

CMOS

Complementary Metal-Oxide-Semiconductor on mikropiirien valmistustekniikka.

OpenCV

OpenCV on avoimeen lähdekoodiin perustuva konenäkökirjasto ohjelmointiin.

Arduino

On Atmel mikropiirien ympärille rakennettu kehitysalusta/ mikro-ohjain elektroniikalle.

Raspberry Pi

On Raspberry Pi Foundationin kehittämä mikro-ohjain, joka perustuu ARM suorittimeen.

Raspbian

Linux pohjainen käyttöjärjestelmä Raspberry Pi alustoille.

USB

Universal Serial Bus, Sarjamuotoinen tiedonsiirtoväylä.

LIDAR

(Light Detection and Ranging) on valonsäteeseen perustuva tutka.

Raspi-Config

Raspi-Config on ohjelmistotyökalu, joka mahdollistaa Raspberry Pi alustan laitteistoresurssien aktivoimisen/de-aktivoimisen sekä muokkaamisen. Työkalun kehitti alun perin Alex Bradbury.

WLAN

Wireless Local Area Network, Tarkoittaa käytännössä langatonta internet liittymää.

TFT

Thin Film Transistor on valmistus tekniikka, missä puolijohde transistorit valmistetaan ohuita kalvoja pinoamalla.

RISC

RISC (Reduced Instruction Set Computer) on suoritinarkkitehtuuri, jonka tarkoituksena on luoda suoritin joka käyttää lyhyempää ja selkeämpää käskykantaa.

ARM

ARM (Advanced RISK Maschines) on suoritin arkkitehtuuri, jonka kehitti Acorn yhtiö vuonna 1980.

HDMI

HDMI tulee sanoista High-Definition Multimedia Interface ja se on yleisesti käytössä oleva rajapinta kuvan sekä äänen siirtämiseen.

IDE

Integrated Development Environment on ohjelmisto joka sisältää tarvittavat perustyökalut ohjelmistojen kehittämiseen.

1 Johdanto

Tämän projektin ajatuksena oli opetella OpenCV-konenäköympäristön toimintoja ja ominaisuuksia, samalla avautui mahdollisuus myös opiskella Python-ohjelmointikieltä sekä Raspberry Pi ja Arduino alustojen käyttämistä yhdessä sarjaväylää rajapintana hyödyntäen.

Projektille ei ole tilaajaa, vaan se on syntynyt omasta kiinnostuksesta ja halusta kehittyä harrastuksen parissa, kuitenkin ajatuksena on myös tuottaa dokumentaatio joka mahdollistaisi OpenCV ympäristöön sekä mikro-ohjaimiin perehtymisen ja helpottaisi niiden parissa työskentelyä. Projektin tavoitteena on tunnistaa kameran näkökenttään asetettu tietyn värinen kohde ja syöttää kohteen koordinaatit USB-väylän kautta Arduino alustalle. Tarkoituksena on myös suorittaa kohteen siirto kahden pisteen välillä hyödyntäen koordinaatteja.

Projekti koostuu Raspberry Pi 2 Model B alustasta, johon on asennettu OpenCV-konenäkökirjasto Python-ympäristöön, Arduino Uno mikro-ohjaimesta sekä 3D tulostetusta robottitarttujasta. Arduino mikro-ohjaimen tehtävä on ohjata robottitarttujaa Raspberry Pi alustan lähettämien koordinaattien perusteella ja ilmoittaa, kun liike on suoritettu loppuun.

Raspberry Pi alustaan on asennettu Raspbian käyttöjärjestelmä ja konenäkö ympäristö toimii Python-ohjelmointikielen avulla. Arduino alusta toimiin Arduino-kielellä, joka on C-kielen kaltainen ohjelmointikieli. Rajapinta järjestelmien välillä on toteutettu USB-väylän kautta.

2 Digitaalikuvauksen lyhyt historia

Digitalisenkuvaamisen historia voidaan piirtää hyvinkin kauas, mikäli tarkastellaan hetkeä, milloin ajatus kuvan muuntamisesta sähköiseen muotoon on kehittynyt. Kuitenkin teknisen kehityksen lähtökohtana voidaan pitää vuotta 1880, kun alkuaine seleenin valosensitiivisyys löydettiin ja siitä kehitettiin ensimmäiset ”valokennot”. Vaikka digitalinen kuvantaminen on kehittynyt lähelle nykyistä muotoaan vain reilun 100 vuoden aikana, niin ensimmäiset 40 vuotta menivät varsin rauhallisesti, koska vasta 1920 hollantilainen tiedemies P.S. Hanna haki patenttia nimellä ”electrographic process”. Patentti esittelee kokeen missä sähkövirran

ja valon avulla pystyttiin siirtämään karkeaa kuvainformaatiota kuvasta alustana toimineelle levylle. (Digitalkamera Museum, n.d)

Tästä matka taittui Chester Carlsonin vuonna 1938 tekemän patentin ”Electron Photography” kautta vuoteen 1957 jolloin Russell A. Kirsch siirsi kuvan pojastaan magneettinauhalle ja siitä takaisin kuvaksi näytölle. Tuota kuvaa pidetään ensimmäisenä digitalisena kuvana historiassa. (The Washington post, 2020)

Vuoden 1957 jälkeen digitaalisenkuvauksen saralla kiihdytettiin teknisenkehityksen vauhtia, jo vuonna 1958 kehitettiin ensimmäinen mikropiiri ja 1960 NASA testasi analogisen kuvainformaation muuntamista digitaliseksi sekä lähettämistä avaruudesta maahan. Erinäisten teknistenvirstanpylväiden kautta vuonna 1967 kehitettiin TFT-tekniikkaan perustuva kamera joka osaltaan edisti digitalistenkuvakennojen kehitystä ja johti vuonna 1969 CCD-kuvakennon keksimiseen. (Digitalkamera Museum, n.d)

Konenäön osalta ei digitalinen kuvaustekniikka ole CCD-kennon jälkeen tuonut suuria kehitysaskelia, mutta mikropiirien kehityksen tuoma laskentateho on.

1958 keksityn mikropiirin jälkeen kehitys jatkui mikropiirien tuotannon sekä tekniikan kehittämisen kautta prosessoritekniikkaan ja vuonna 1972, 4-vuotta aiemmin perustettu yhtiö, Intel julkaisi 8008 prosessorinsa. (Computer Hope, 2020)

Motorola ehti prosessori kilpaan vuonna 1979 julkaisemalla 68000 suorittimensa, joka toimi perustana usealle 80- ja 90-luvun kotitietokoneista. (Computer Hope, 2020)

90- ja 2000-luvut menivätkin suoritinteknologian ”kilpajuoksuna” kahden toimijan välillä, nuo olivat vuonna 1968 perustettu Intel ja 1969 perustettu AMD. Kilpailu on osaltaan edistänyt laskentanopeuden kasvamista ja konenäköjärjestelmien kehitystä.

Teknisen kehityksen rinnalla voidaan konenäön kehityksen kannalta oleellisena virstanpylväänä pitää Lawrence Roberts:n 1963 julkaisemaa “*Machine perception of three-dimensional solids”* tutkimusta. Kyseisen tutkimuksen merkittävimpiä ajatuksia oli 3-ulotteisen informaation luominen kaksiulotteisesta kuvasta. (Corke, 2017)

Edellä esitelty historia on esitetty aikajanalla alla olevassa kuvassa. Kuva 1

Kuva 1. Tekninen aikajana

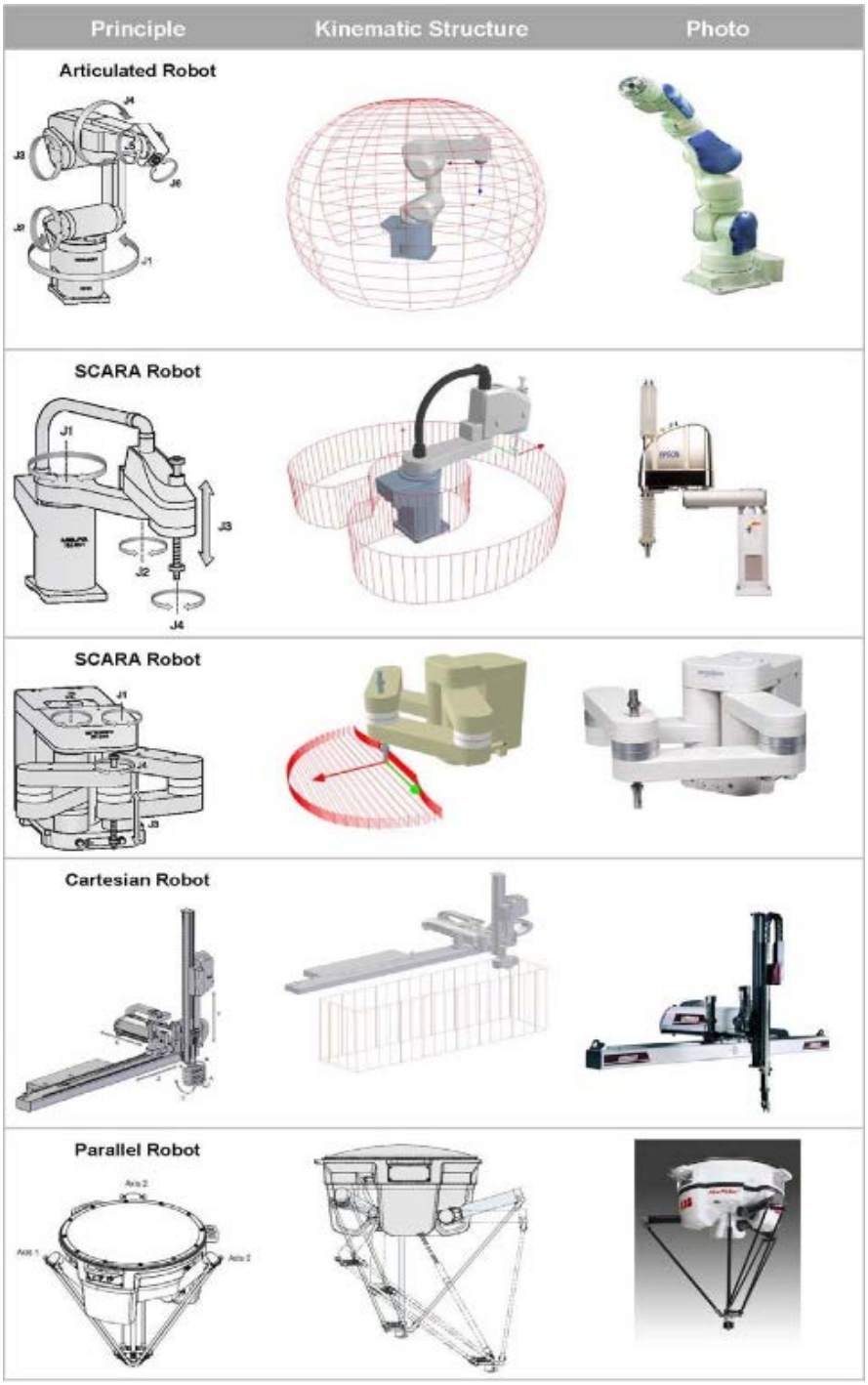
3 Robotiikka

Robotiikka on tutkimusala, joka keskittyy robottien suunnitteluun ja kehittämiseen. Robotin määrittely löytyy ISO 8373:2012 standardista, mutta karkeasti tulkittuna se on ohjelmoitava laite, jossa on 2 tai useampi akseli ja se kykenee suorittamaan tehtävänsä itsenäisesti sekä liikkumaan ympäristössään tarvittaessa. (iso.org, 2021)

Robotiikka on syntynyt tarpeesta kehittää tehokkaampia ja kestävämpiä robotteja eri tehtäviin. Koska ihminen ei ole kauhean hyvä suorittamaan toistavaa, mutta tarkkuutta vaativaa työtä, on robottien kehitys ja robotiikka tarjonnut ratkaisuja yhä useampaan tehtävään. Myös ikävien kotiaskareiden hoitaminen on saanut apua robotiikasta Robotti-imureiden kautta.

Robotiikassa robotit jaetaan kahteen ryhmään; Teollisuusrobotit sekä palvelurobotit. Teollisuusrobottien luokittelu perustuu niiden mekaaniseen rakenteeseen (Kuva 2) (International Federation of Robotics, n.d), kun taas palvelurobotit luokitellaan käyttötarkoituksen perusteella. (Alho;Neittaanmäki;Hänninen;& Tammilehto, 2018)

Kuva 2. Teollisuusrobottien luokat. (International Federation of Robotics, n.d)

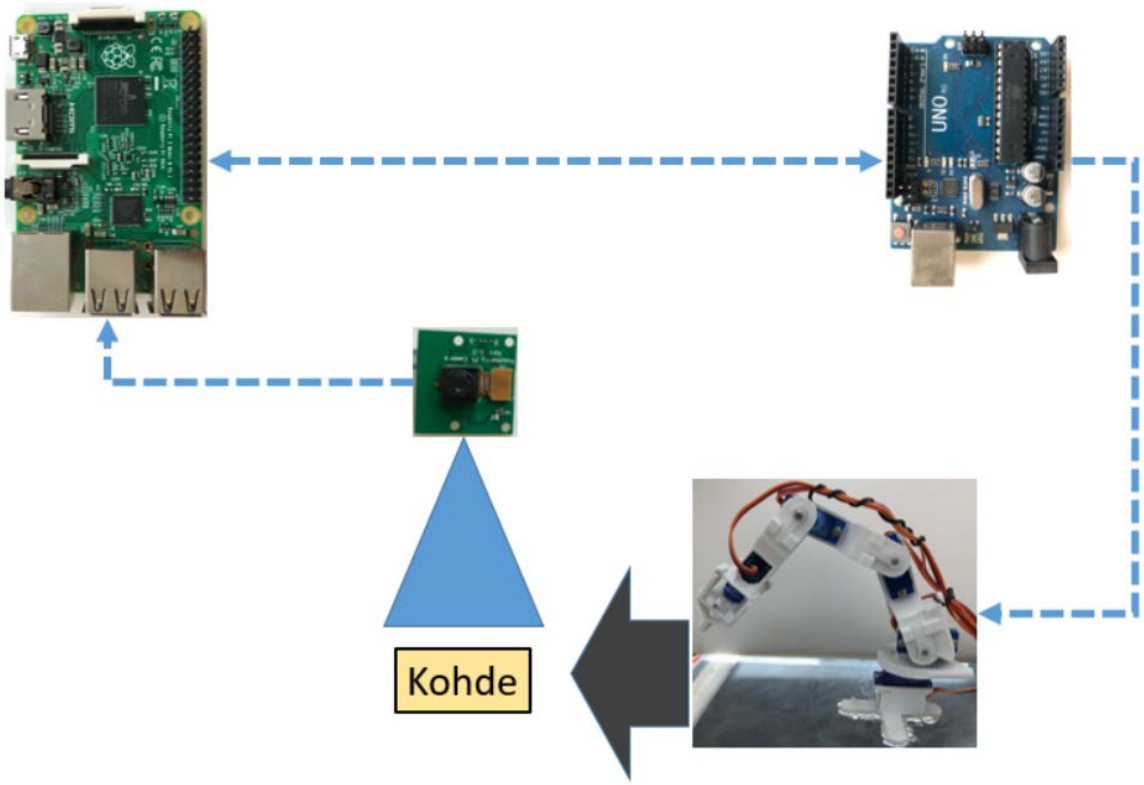


Robotiikka ja konenäkö liittyvät monissa ratkaisuissa toisiinsa ja laskentatehon kasvun myötä, yhä monimutkaisemmat toiminnot ja ohjaukset ovat olleet mahdollisia, tästä esimerkkinä kehittyneet ohjaukset ja tunnistukset teollisuusroboteissa sekä LIDAR tekniikan käyttäminen robotti-imureiden ympäristön havainnoinnissa.

4 Laitteisto

Laitteistona projektissa toimii Raspberry Pi 2 Model B pienoistietokone Raspbian käyttöjärjestelmällä sekä Arduino Uno mikro-ohjain, joka on kytkettynä 3D tulostettuun robottitarttujaan. Ohjelmistona on Python-kieleen integroitu OpenCV-konenäkö kirjasto sekä Arduino-kielellä toteutettu robottitarttujan ohjaus. (Kuva 3)
Seuraavissa kappaleissa esittelen laitteiston osat tarkemmin.

Kuva 3. Järjestelmän rakenne.



4.1 OpenCV

OpenCV on avoimenlähdekoodin konenäön- sekä koneoppimisenkirjasto, joka tarjoaa käyttöön suurimman osan konenäköön liittyvistä funktiosta, sekä lisäksi mahdollistaa tietokonegrafiikan lisäämisen kuviin. OpenCV pitää sisällään kattavan määrän toimintoja

kuten kuvan väriavaruuden muuttamisen, kuvan skaalaamisen, värien, muotojen sekä kappaleiden tunnistamisen ja kasvotunnistusalgoritmin. OpenCV-kirjasto on löytänyt tiensä myös monien suuryritysten tuotteiden perustaksi ja sen avulla tehdään nykyisin laaja-alaisesti eri toimintoja ympäri maailmaa. (Information Security Newspaper, 2020)

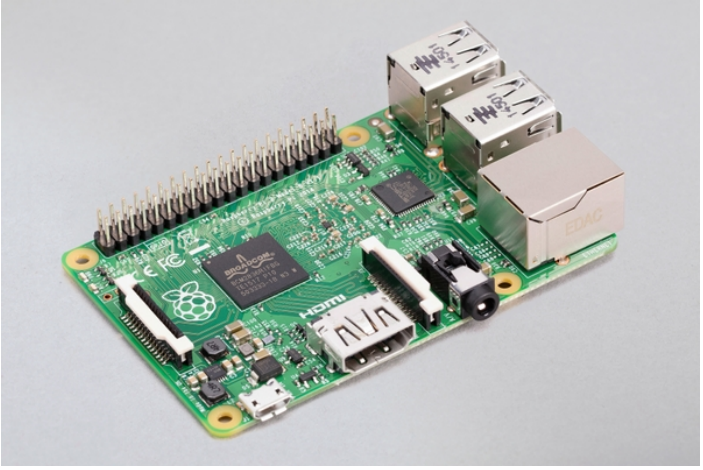
OpenCV tukee C++, Python, Java ja MATLAB ympäristöjä sekä Windows, Linux, Android ja Mac OS alustoja. Lisäksi laaja tuki eri alusta arkkitehtuureille on lisännyt OpenCV-kirjaston soveltuvuutta eri sovelluksiin niin mikro-ohjaimissa ,kuin sulautetuissa järjestelmissä. (OpenCV, 2021)

4.2 Raspberry Pi

Raspberry Pi on pienoistietokone, jonka on kehittänyt Englannista lähtöisin olevat hyväntekeväisyysjärjestö, Raspberry Pi Foundation. (Raspberry Pi Foundation, 2021) Tietokoneen arkkitehtuuri on toteutettu yhdelle piirilevylle, tämä mahdollistaa yhdessä edullisen hinnan kanssa Raspberry Pi -tietokoneen käyttämisen useissa erilaisissa käyttökohteissa.

Projektissa käytetty Raspberry Pi 2 Model vastaa konenäöstä ja toimii OpenCV-ympäristön alustana (Kuva 4). Raspberry Pi alustat käyttävät ARM Cortex suorittimia, jotka ovat RISC-arkkitehtuuriin perustuvia suorittimia ja ovat olleet käytössä myös matkapuhelin ympäristössä. Alusta mahdollistaa USB-rajapintojen lisäksi suoran IO-rajapinnan erillisten GPIO-pinnien kautta. Kortti voidaan liittää näyttölaitteeseen HDMI-liittimen kautta tai piirilevyn reunalla olevan Display-liittimen kautta, mikäli näyttölaite tukee tuota liitäntä mahdollisuutta. Verkkoyhteydet on mahdollista muodostaa verkkokaapelilla tai WLAN-yhteydellä, joskin projektissa käytetty Raspberry Pi 2 Model B ei sisällä integroitua WLAN-yhteyttä ja se on rakennettu erillisellä USB-sovittimella.

Kuva 4. Raspberry Pi 2 Model B (Raspberry Pi Foundation, 2021)



4.2.1 Raspbian

Raspbian (nykyisin Raspberry Pi OS) on ilmainen ja avoin käyttöjärjestelmä Raspberry Pi-ohjainalustalle. Se pitää sisällään perusohjelmistot ja laitteiston hallintaan liittyvät työkalut, kuten Raspi-Config-työkalun. Käyttöjärjestelmä ei ole Raspberry Pi Foundationin kehittämä, vaan on käyttäjien luoma ja sen ovat kehittäneet Mike Thompson sekä Peter Green yhteistyössä muiden käyttäjien kanssa. (Raspbian, 2021)

Ohjelmisto on ladattavissa Raspbianin sivuilta ja sen asentamiseen muistikortille löytyy ohjeet useammastakin lähteestä. (Pi, nd.) Projekti alustaksi valitsin Raspbianin, koska sen graafinen käyttöliittymä oli helppo omaksua ja lisäksi siihen sai työkalut mm. Arduino-alustan ohjelmoimiseksi.

4.2.2 Python

Python on korkeantason ohjelmointikieli, jonka suunnitteli Guido van Rossum vuonna 1991 ja kehitti Python Software Foundation. Python on suunniteltu koodin luettavuutta ajatellen ja se mahdollistaa syntaksien ilmaisun vähemmillä koodiriveillä. (Pramanick, 2019)

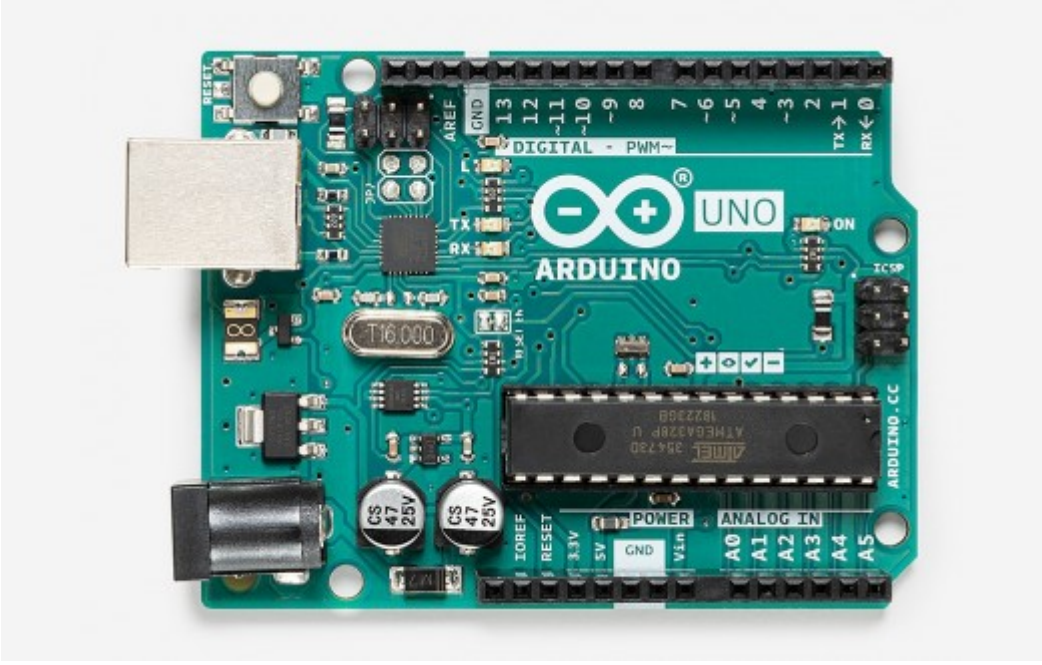
Projektissa on käytetty Python 3.7 ohjelmointikieltä ja se valikoitui käytettäväksi koska Raspbian käyttöjärjestelmään oli asennettu Python valmiiksi ja OpenCV-ympäristö tukee myös Python-kieltä.

4.3 Arduino

Arduino on avoin kehitysalusta joka mahdollistaa helpon lähestymisen mikro-ohjainten maailmaan. Arduino on rakennettu Atmel AT -mikropiirien ympärille ja oman boot laturin ansiosta, mikropiirejä on mahdollista ohjelmoida Arduino IDE -työkalulla.
Projektissa päätin käyttää Arduino-ohjainta, koska kyseisiä ohjaimia oli minulla valmiina kotona ja koska ohjelmointikoodin kääntäminen C-ympäristöön ei olisi vaikeaa myöhemmin johtuen Arduino-kielen pohjautumisesta C-kielen kaltaiseen processing-kieleen. (Badami, nd)

Projektissa käytettävä mikro-ohjain on Arduino UNO (Kuva 5), joka on yksi Arduino tuoteperheen ohjainkortteja. Arduino-ohjaimen merkittävimmät komponentit ovat Atmel ATmega328P mikro-ohjain, ATmega16u2 USB-ohjain, jänniteregulaattorit sekä kellokide. Vastaavat komponentit löytyvät myös muista Arduino-perheen tuotteista.

Kuva 5. Arduino Uno (Arduino, 2018)



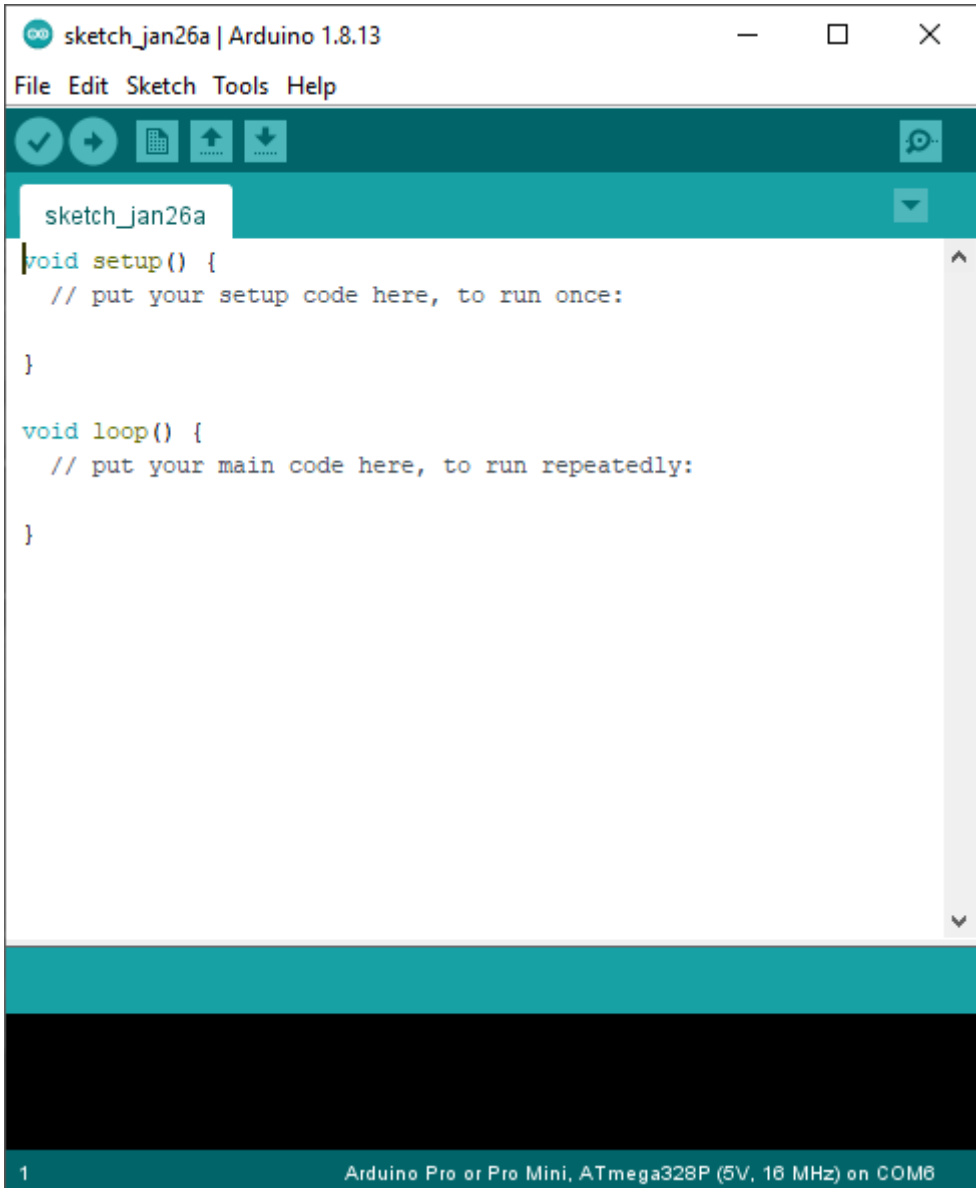
4.3.1 Arduino IDE

Arduino IDE on avoimeenlähdekoodiin perustuva kehitysympäristö Arduino-ohjaimille. Arduino IDE pitää sisällään perustyökalut koodin kirjoittamiseen, kääntämiseen ja lataamiseen kaikille Arduino-ohjaimille, lisäksi sen avulla on mahdollista kommunikoida

ohjaimen kanssa sarjaväylää hyödyntäen, ladata kirjastoja komponenteille ja tarkistaa ohjaimen tiedot.

Arduino IDE perusnäkymä (Kuva 6) koostuu koodi-ikkunasta, ilmoitusikkunasta sekä työkalu- ja valikkoriveistä.

Kuva 6. Arduino IDE näkymä.



4.3.2 Arduino-kieli

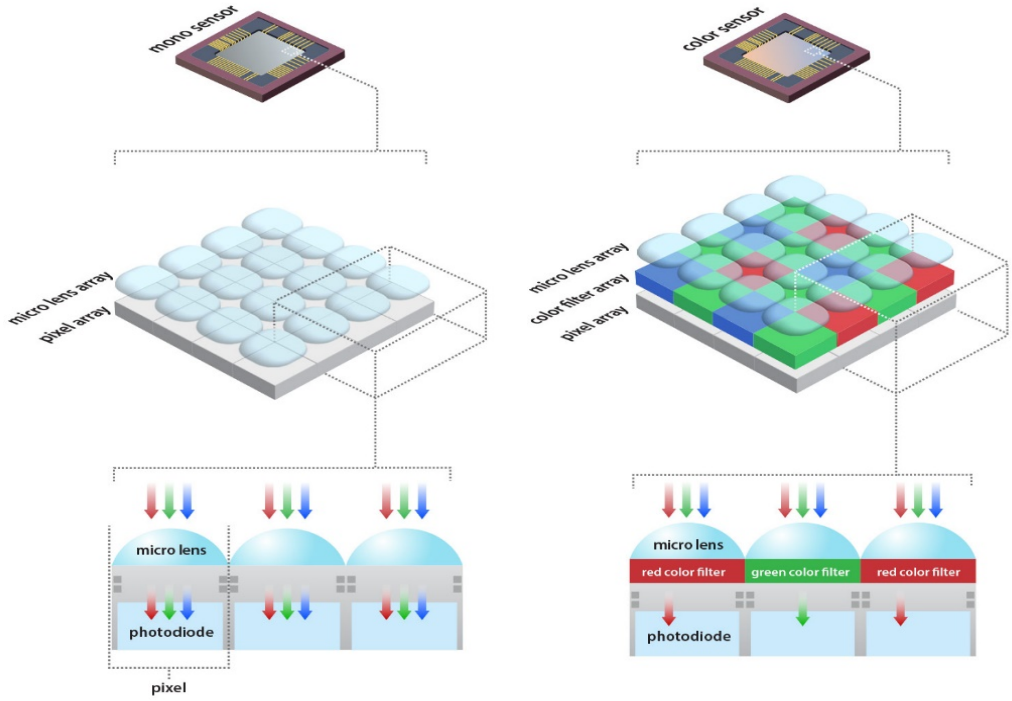
Arduino-kieli voidaan jakaa kolmeen pääkomponenttiin: funktiot, muuttujat sekä struktuurit. Koodin rakenne on hyvin paljon samanlainen kuin C-sukuisissa kielissä ja useimmat funktiot kirjoitetaankin samoin kuin C-kielessä.

Koodin rakenne koostuu kahdesta osasta, ”void setup” suoritetaan vain ohjaimen käynnistyksen yhteydessä ja tässä voidaan mm. määrittää laitteisto resursseja koskevat komennot sekä alustaa muuttujat. ”void loop” on varsinainen ohjelmasilmukka joka jatkaa ”setup” –alustuksen jälkeen toimintaa niin kauan, kuin ohjainkortissa on virrat. ”void loop” vastaa ohjelmallisesti samaa kuin ”while(1)” –komento jossa ”while” funktion alle kirjotettu koodi muodostaa ohjelmasilmukan.

5 Konenäkö

” Konenäön toiminnan perusperiaate on kuvata näkyvän valon aallonpituudella tapahtuvia ilmiöitä.” (Opetushallitus, 2021)
Konenäkölaitteisto rakentuu kamerasta, valaisulaitteistosta, sekä ohjelmistosta ja käyttöliittymästä. Toteutuksesta riippuen, voivat mittausohjelmisto ja kuvankäsittelyohjelmisto olla hajautettu kahdeksi erilliseksi kokonaisuudeksi. Konenäköjärjestelmän toiminta perustuu digitaaliseen kuvainformaatioon ja siitä tehtyyn analyysiin. Kamera ottaa kohteesta digitalisenkuvan, tämä tarkoittaa, että kuva muodostuu pikseleistä eli pienistä pisteistä joille annetaan kirkkausarvo. Kuitenkin kirkkaudella voidaan muodostaa vain mustavalkokuvia, joten värikomponenttia varten osa pikseleistä omaa väriä suodattavan linssin ja niiden avulla muille pikseleille lasketaan väriarvo. Kuvassa 7 on esitelty yksi mahdollinen tapa toteuttaa kyseinen ratkaisu.

Kuva 7. Värien muodostaminen digitaliselle kuvakennolle. (Lucid vision lab, 2021)

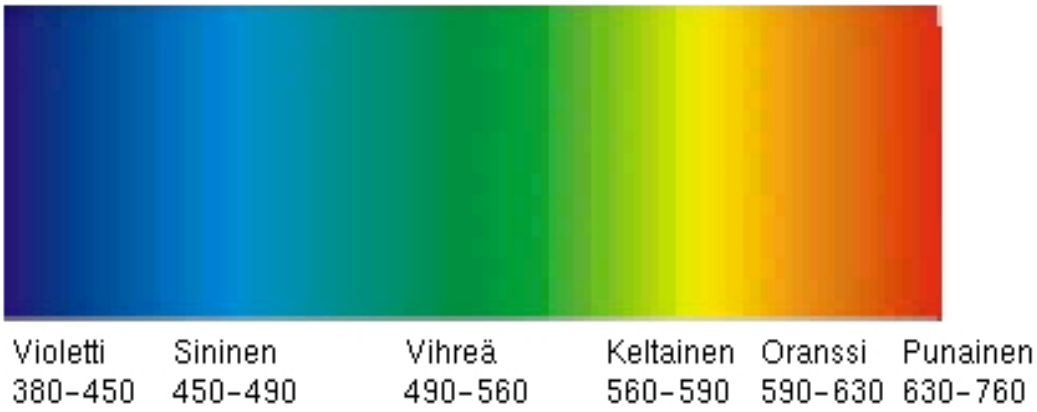


5.1 Väriteoria

Näkyvä valo on sähkömagneettista säteilyä jonka aallonpituus osuus välille 400-700nm. (Kuva 8) Valon aallonpituus korreloi väriin niin, että pidempi aallonpituus näkyy havaitsijalle punaisena ja aallonpituuden pieneneminen johtaa värin muuttumiseen oranssin, keltaisen ja vihreän kautta siniseksi. (Ilmatieteen laitos, 2021)

Näkyvän valon lisäksi, molemmissa päissä skaalaa on silmälle näkymätöntä valoa, aallonpituuden ollessa yli 760nm kutsutaan tuota valoa infrapunaksi ja aallonpituuden ollessa alle 380nm nimenä on ultravioletti. Infrapunavalo samoin kuin ultraviolettivalo ovat silmälle näkymätöntä, mutta voidaan havaita digitalisella kuvasensorilla. Tästä johtuen useimmissa kameroissa on suodattimet infrapunavaloa sekä ultraviolettivaloa vastaan.

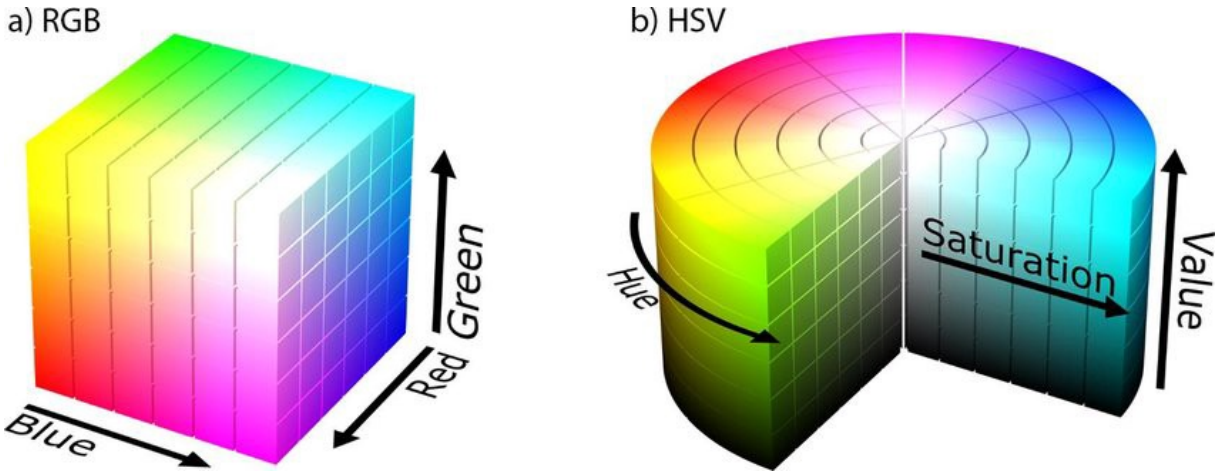
Kuva 8. Näkyvän valon aallonpituudet. (Ilmatieteen laitos, 2021)



Väri voi koostua kolmesta peruskomponentista, joita voidaan sekoitta joko vähentämällä tai lisäämällä ne toisiinsa. Esimerkkinä lisäävästä sekoittamisesta voisi toimia televisio, jossa kaikkien värien summa on valkoinen ja vähentävässä sekoittamisessa maalien sekoitus jossa kaikkien välien summa on musta.
Digitalisen kuvantamisen avuksi on luotu tapoja tehdä väreistä koordinaatistoja, yksi näistä RGB (Red, Green, Blue) -koordinaatisto jossa annetaan punaiselle, vihreälle ja siniselle arvo, näitä arvoja säätämällä on mahdollista luoda eri värejä.

Ihmisen luontaista värien hahmottamista on pyritty ajan saatosta jäljittelemään ja siihen tarkoitukseen luotiin HSV (Hue, Saturatio, Value) ja HSB (Hue, Saturatio, Brightness) värimallit. (Kuva 9)
Näissä värimalleissa ”hue” (sävy) määrittää värisävyn, ”saturaatio” määrittää kuinka saturoitunut tai ”kyllästetty” väri on, viimeinen arvo määrittää värin kirkkauden.

Kuva 9. RGB ja HSV koordinaatistot. (Abenza, 2018)



5.2 Konenäön haasteet

Konenäön tunnistus nojaa vahvasti laskentaan ja suorittimien kehitys onkin mahdollistanut yhä pienempien ja monimutkaisempien järjestelmien rakentamisen, myös tunnistusalgoritmit ovat kehittyneet ajan saatossa huomattavasti. Projektissani käytin tunnistusta, joka pohjautuu kohteen väriin koska hyvällä taustan sekä valaistuksen määrityksellä ja värialueen tarkalla asettamisella voidaan tällä menetelmällä saavuttaa hyvä tarkkuus kohtuullisella laskenta teholla.

OpenCV, kuten useimmat konenäköjärjestelmät, voi tunnistaa kohteita sekä video- että valokuvasta, kuitenkin kohteentunnistuksen kannalta tunnisteiden määrittely on erityisen tärkeää ja siinä tulee huomioida myös ympäristön vaikutukset.

Valaistus on suuressa roolissa, riippumatta siitä, tehdäänkö tunnistus muodon vai värin perusteella.

Konenäköjärjestelmissä monimutkaisten muotojen ja symmetrioiden tunnistaminen vaatii paljon laskentaa, mutta sen avulla on mahdollista segmentoida kohteita tehokkaammin ja myös tarkentaa sekä nopeutta tunnistamisprosessia. Tämä kuitenkin vaatii huolellista määrittelyä tunnistuspisteille, jotta vältytään virheellisiltä tunnistuksilta.

5.3 Kuvakennot

Digitalisen kuvantamisen perustana toimii kuva-sensori, jonka avulla heijastunut kuva voidaan muuttaa sähköiseksi informaatioksi. Sensorien toiminta perustuu valosähköiseen

ilmiöön missä fotonin kuljettama energia irrottaa pii-kiteestä elektronin. Kuvakennot voidaan jakaa fyysisen arkkitehtuurin perusteella karkeasti kahteen ryhmään, CCD ja CMOS kennot.

CCD:n (Charge-Coupled Device) kehitys sai alkunsa AT & T Bell Labs:n laboratoriossa, missä George E. Smith ja Willard S. Boyle pyrkivät kehittämään uudenlaista muistia vuonna 1970. Hyvin pian he kuitenkin huomasivat, että kyseinen piiri soveltui myös digitalisen valokuvan muodostamiseen. (Janesick, 2001)

CCD-piirin toiminta voidaan yksinkertaistetusti ajatella peräkkäisinä kondensaattoreina sekä niiden välissä sijaitsevina kytkiminä, kytkimiä sopivasti ohjaamalla, voidaan vierekkäisten kondensaattorien varauksia siirtää eteenpäin ja näin luoda tavallaan siirtorekisteri. (Janesick, 2001) Käytännössä toteutus on kuitenkin tehty puolijohdekondensaattoreilla ja on monimutkaisempi.

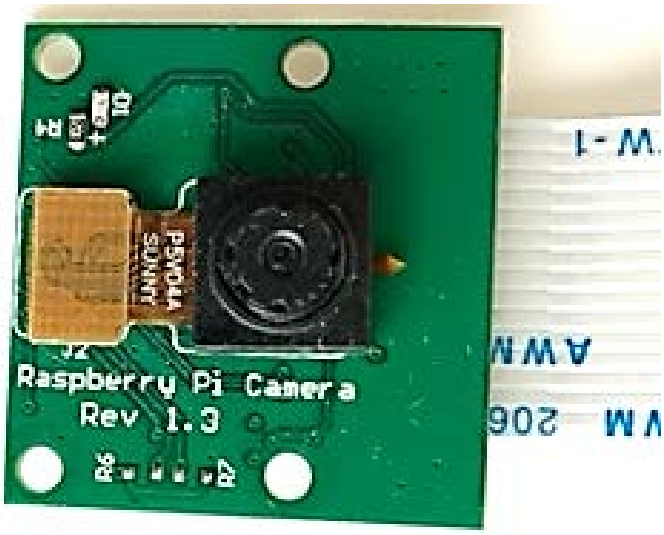
Arkkitehtuurista johtuen CCD-kennot kuluttavat suhteellisen paljon energiaa verrattuna CMOS-kennoihin, mutta toisaalta tuottavat korkealaatuista kuvaa koska pikselien ympärille ei tarvitse rakentaa omia vahvistinpiirejä ja näin ollen sensorin pikselit ovat lähempänä toisiaan sekä altistuvat pienemmille sähköisillehäiriöille. CCD-kennot ovat kalliimpia valmistaa kuin CMOS-kennot ja ne ovat myös hitaampia käyttää. Kuitenkin, jos halutaan korkealaatuista kuvaa, jonka kohinasuhde on pieni, on CCD kenno edelleen hyvä valinta ja sitä käytetään esimerkiksi tähtien kuvaamiseen tarkoitetuissa kamerayksiköissä.

CMOS-Sensorin (complimentray metal oxide semiconductor) arkkitehtuurissa jokaisen pikselin yhteyteen on integroitu oma vahvistin piiri. Tämä osaltaan parantaa signaalin laatua ja mahdollistaa paremmat kuvaus ominaisuudet, mutta kasvattaa pikselin vaatimaa tilaa ja näin ollen myös heikentää kennon valo-ominaisuuksia.
Kuitenkin useimmat nykyiset digikamerat perustuvat CMOS teknologiaan paljolti siksi että CMOS sensorit ovat nopeampia ja halvempia valmistaa. (Tokyo Electron, n.d)

Projektissa kamerana toimii Raspberry Pi Camera module rev 1.3 (Kuva 10), joka on 5 megapikselin CMOS kamera kenno. Kamera kykenee ottamaan 2592x1944 pikselin still-kuvia sekä HD video kuvaa 1080p ja 720p resoluutiolla. Kamera kytketään Raspberry Pi alustaan 15

pinnisellä lattakaapelilla Camera Serial Interfacea (CSI) hyödyntäen ja sen ohjaamista varten on python ympäristöön saatavilla oma kirjasto.

Kuva 10. Raspberry Pi Camera.



5.4 Tunnistaminen

Projektissa kohteentunnistamiseen hyödynnetään väriä, mutta konenäköjärjestelmissä on mahdollista tunnistaa kohteita useilla muillakin parametreilla. Väritunnistuksessa kohteelle annetaan värialue ja ohjelma etsii tuolle värille tai värialueelle sopivia pisteitä, tunnistuksen perusteella kuvasta luodaan mustavalkoinenmaski, jota ohjelmiston on helpompi käsitellä.

Konenäköä käytetään usein tunnistamaan kappaleiden muotoja sekä symmetrioita, tunnistaminen useiden parametrien perusteella mahdollistaa tarkemman tunnistamisen sekä erottelun. Muotoon perustuvassa tunnistuksessa voidaan käyttää apuna valmista maskia, mihin ohjelma vertaa tunnistettua kappaletta tai pisteiden välisiä suhteita voidaan laskea ohjelmallisesti.

6 Projekti

Projekti aloitettiin tekemällä Arduino-alustalle servojenohjauskoodi, missä liike kutsutaan yhdenrivinfunktioilla. Tämä helpottaa koodin lukemista sekä kirjoittamista. Python-koodin muodostaminen alkoi tutustumisella OpenCV-kirjaston funktioihin sekä niiden syntakseihin.

Aluksi Raspberry Pi -alustalle piti asentaa OpenCV-kirjasto Python-kielelle ja raspi config työkalulla piti kytkeä kameraportti käyttöön Raspberry Pi -alustasta.

Pyrin pitämään koodin selkeänä ja yksinkertaisena, jotta sen seuraaminen olisi helpompaa, tämä osaltaan tarkoittaa sitä, että koodi on kompromissi luettavuuden ja tehokkuuden optimoinnin väliltä. Python koodin rakenne koostuu alustuksesta, kuvan ottamisesta, analyysistä sekä suorittavasta osasta, missä tehdään kohteelle tunnistuspiste, grafiikka sen ympärille, sekä lähetetään tieto ja odotetaan liikkeenkuittausta. Arduino-koodin tehtävä on saada kohteen koordinaatit, määrittää poimintapiste ja poimia kohde sekä kuljettaa se haluttuun pisteeseen. Lopuksi Arduino kuittaa toiminnon suoritetuksi, jotta Raspberry Pi voi ottaa uuden kuvan, tällä vältetään liikkeenaikainen turha informaatio ja säästetään laskentakapasiteettia.

Toiminnan kannalta olennaiset funktiot ovat selitettynä seuraavissa kappaleissa.

6.1 Python-koodin toiminnot.

Koodin alussa esitellään ohjelmalle tarpeelliset kirjastot sekä luodaan tekstin tulostamista varten aliohjelma, myös tarpeelliset muuttujat alustetaan tässä vaiheessa.
Alustuksessa määritellään myös ohjelmalle käytettävä USB-portti joka pitää käyttäjän määrittää komennolla ”ser = serial.Serial('/dev/ttyACM0', 9600)”. Sarjaportin määrityskomennossa ’dev/tty/ttyACM0’ on USB portin osoite ja osoite pitää tarkistaa käytössä olevan USB portin perustella.
Itse tarkistin ’/dev’ kansiossa näkyvät laitteet ja vertasin listaa tilanteeseen, kun USB-laite ei ole kytkettynä. Lista tulostuu, kun konsolissa syöttää ensin komennon ”cd /dev” joka avaan kansion ”dev”, tämän jälkeen komennolla ”ls” saa tulostettua listan liitetyistä laitteista ja USB-laite on aina muodossa ”ttyXXX”.
Kuvassa 11 on esitetty itse ohjelmasilmukan ensimmäinen vaihe, joka pitää sisällään kuvan ottamisen sekä värimaskin luomisen ja siihen liittyvät korjaukset.

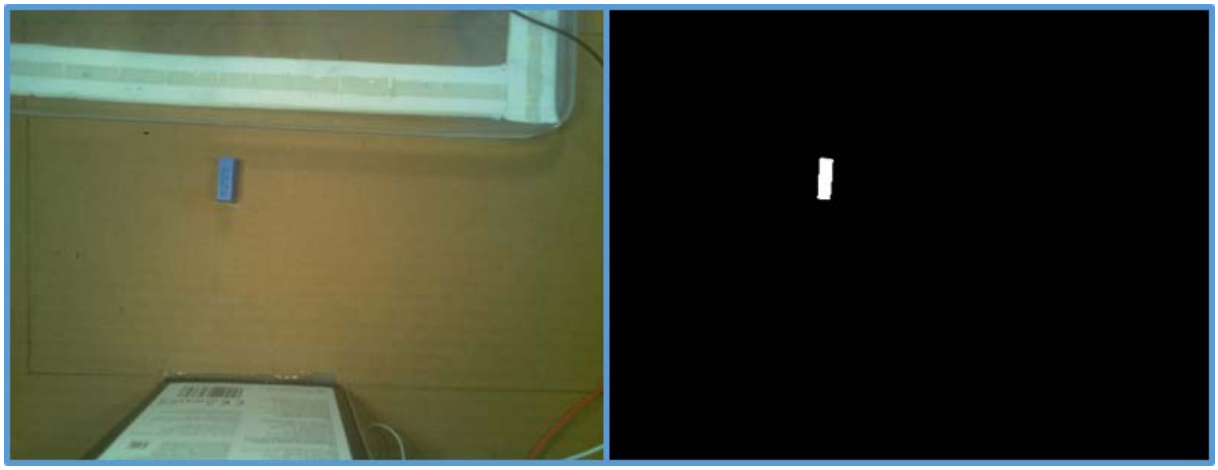
Kuva 11. Python koodi, kuvan ottaminen.

```
26  # Kohteen tunnistus loop
27  while True:
28      i=0                                     #Alustetaan muuttuja i arvoon 0
29      camera.start_preview()                  #Käynnistetään kamera
30      camera.capture('/home/pi/snapshot.jpg')  #Otetaan kuva ja tallennetaan se
31      camera.stop_preview()                   #Sammutetaan kamera
32
33      frame = cv2.imread('snapshot.jpg', 1)   #Luetaan kuva tiedostosta
34      frame = imutils.resize(frame, width=500)  #skaalataan kuvan leveys 500 kuvapisteeseen
35      hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)  #muutetaan kuvan värikoordinaatisto BGR (Blue,Green,Red) HSV:hen (Hue, Saturatio, Value)
36
37      # Luodaan maski tunnistukselle
38      # Poistetaan maskista häiriöt käyttämällä "erode" ja "dilate" funktioita
39      mask = cv2.inRange(hsv, colLow, colUp)
40      mask = cv2.erode(mask, None, iterations=1)
41      mask = cv2.dilate(mask, None, iterations=1)
```

Kuvassa rivi 34 ’frame = imutils.resize(fram, width=500)’ muuttaa kuvankokoa niin, että leveydeksi tulee 500 kuvapistettä ja korkeus skaalautuu kuvasuhteen mukaan. Tämän tarkoitus on nopeuttaa kuvankäsittelyprosessia, eikä se vaikuta käytetyltä etäisyydeltä tunnistustarkkuuteen. Kuvantarkkuus ennen skaalausta on kameran suurin tarkkuus, joka on 2592 kuvapistettä vaakasuunnassa.

Riviltä 39 eteenpäin kohteelle luodaan maski niin, että kaikki kuvapisteet jotka osuvat ’colLow ja colUp’ arvojen väliin, merkitään maskiin arvolla 1 joka näkyy alla olevassa kuvassa (Kuva 12) oikealla puolella valkoisena alueena. Vasemmalla kuvassa näkyy kameran ottama alkuperäinenkuva ilman kuvasuhteenskaalausta.

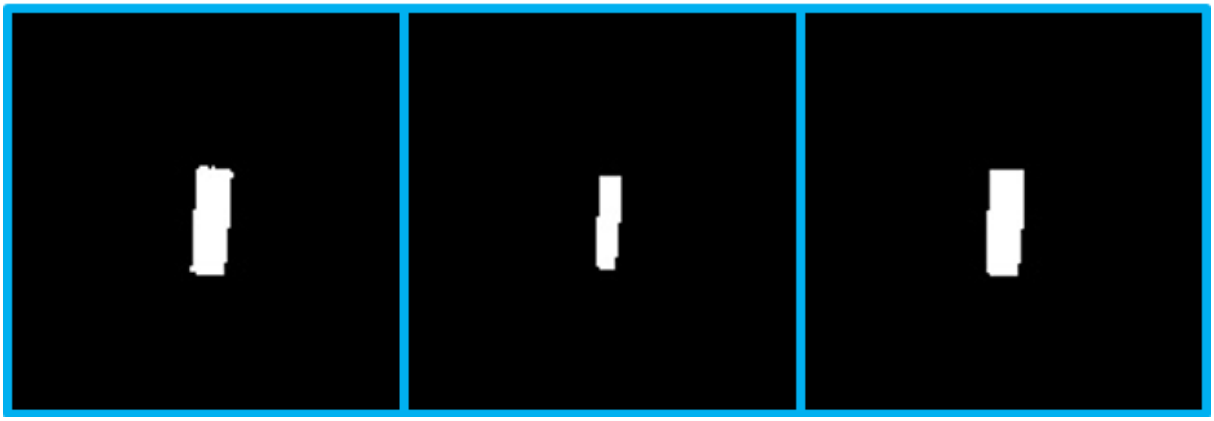
Kuva 12. Tunnistusmaskin luominen kohteesta.



Kuva 13 koostuu kolmesta kuvasta, vasemmalta oikealle luettuna kuva esittää ohjelman riviltä 39 alkavan prosessin jossa kuvalle luodaan värimaski, maskista poistetaan häiriöt ja hajanaiset pikselit erode-funktiolla, sekä lopuksi dilate-funktiolla suurennetaan kohde vastaamaan alkuperäistä maskin kokoa. Vasemman ja oikean kuvan vertailulla voidaan huomata, että kohteen reunat ovat tasaisemmat oikean puoleisessa kuvassa. Erode-funktion

voisi yksinkertaisesti selittää niin, että kuvasta ”leikataan pois epätasaiset reunat” ja dilate-funktiolla lisätään tasaiset reunat jotka mukailevat alkuperäisen kuvan muotoja.

Kuva 13. Erode ja dilate funktiot.



Kuva 14 esittää koodin rivit 53-56 missä maskista etsitään suurin kohde, sekä lasketaan sille tunnistuspisteen koko ja keskipiste. Demonstraatiokuvassa kohteita on vain yksi ja se oli suhteellisen selkeä tunnistaa taustasta, joten suurimman kohteen tunnistus toiminto ei ole tässä tapauksessa täysin relevantti. Halusin tuon kuitenkin jättää koodiin, koska se helpottaa koodin sovittamista eri ympäristöihin. Rivillä 55 luodaan muuttuja ’M’ valitulle muodolle ’c’ käyttämällä ’cv2.moments’ funktiota. ”Moments” funktio yksinkertaistettuna laskee kuvasta valkoisten pikselien ”massakeskipisteen” ja siitä voidaan johtaa kaava kuvan keskipisteen koordinaatteihin.

Kuva 14. Python koodi, Tunnistuspiste kohteelle.

```
53          c = max(cnts, key=cv2.contourArea)                          #Etsitään suurin muoto
54          ((x, y), radius) = cv2.minEnclosingCircle(c)                #Lasketaan ympyrän koko
55          M = cv2.moments(c)                                          #Alustetaan muuttuja M
56          center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"])) #Lasketaan keskipiste moments-arvojen avulla
```

Kuva 15 esittää python-koodin loppuosan, jossa piirretään kuvaan ympyrät tunnistuspisteen päälle, lähetetään koordinaatit sarjaväylään, sekä tulostetaan käyttäjälle kuva kohteesta ja konsoliin koordinaatit kohteelle. Ohjelma odottaa, kunnes Arduino-ohjain on lähettänyt kuittausviestin joka tarkoittaa, että liike on suoritettu loppuun. Tällöin konsoliin tulostetaan teksti ”ok”. Muuttuja ’i’ on tarkoitettu while-funktion käyttämiseen ja sen ollessa ’0’ siirrytään suorittamaan ’while’ silmukka.

Kuva 15. Python koodi, koordinaattien lähetys.

```
58          # If lauseke jos halkaisija on yli 10 kuvapistettä
59          if radius > 10:
60              cv2.circle(frame, (int(x), int(y)), int(radius),
61                  (255, 0, 255), 2)
62              cv2.circle(frame, center, 5, (255, 255, 0), -1)
63
64              #Servon ohjaus
65              ser.write(struct.pack('>2H', int(x),int(y)))
66              mapPos(int(x), int(y))
67              ser.flushOutput()
68
69              while i==0:
70                  number = ser.read()
71                  if int.from_bytes(number,byteorder='big')==18:
72                      print("ok")
73                      i=1
74
75
76      # Tulostetaan kuva kehykseen
77      cv2.imshow("Frame", frame)
78      time.sleep(1)
```

6.2 Arduino-koodin toiminnot

Arduino-koodin alussa esitellään ohjelmalle tarpeelliset muuttujat ja funktiot, sekä asetetaan tarvittavat lähdöt, tulot ja sarjaväylä asetukset. Arduino-koodissa on lisäksi ’void setup’ funktio, jonka avulla alustetaan osa toiminnoista, tämä koodi ajetaan kerran ennen varsinaisen ohjelman käynnistymistä ja siellä on mahdollista esimerkiksi asettaa servoille alkupositiot. Kuvassa 16 riviltä 50 alkaa varsinainen ohjelmasilmukka, jossa aluksi alustetaan x- ja y-muuttujat, sekä luodaan sarjaliikennettä varten viestipuskuri. Ensimmäisellä If-lausekkeella varmistetaan, että sarjaväylä on käytössä ja sieltä tulee dataa ’Serial.available()’ -komento antaa vastaukseksi, kuinka monta tavua sarjaväylästä on tulossa ja vertailulla varmistetaan, että arvo on suurempi kuin nolla. Rivillä 59 luetaan ’buffer’ puskuriin neljä ensimmäistä tavua sarjaväylästä.

Riviltä 61 ja 65 alkavat x- sekä y-koordinaattien tallennus, Python lähettää kaksi 16-bittistä muuttujaa, jotka jakautuvat neljälle 8-bittiselle tavulle. Koska arvo on väliltä 0-255 niin se voidaan esittää yhdellä 8-bittisellä tavulla ja käsitellä arduinossa integerinä (int). Tästä johtuen ohjelmassa luetaan bufferista vain tavut 1 ja 3 koska niiden sisältämä informaatio on relevanttia laitteiston toiminnalle.

Rivillä 70 on if-lauseke, joka vertailee, onko kohteenkoordinaatit oikeat, jotta liike voidaan suorittaa. Molemmille koordinaateille annetaan lausekkeessa yläarvo sekä ala-arvo ja jos kohde on näiden arvojen alueella, voidaan robotin liike suorittaa.

Kuva 16. Arduino-koodi, sarjaväylän luku.

```
49  //Ohjelma loop
50  void loop() {
51
52
53
54    int x = 0;
55    int y = 0;
56    char buffer[4];
57
58    if (Serial.available() > 0) {
59      int size = Serial.readBytesUntil('\n', buffer, 4);
60
61      x = buffer[1];
62      Serial.print("X is: ");
63      Serial.println(x,DEC);
64
65       y = buffer[3];
66      Serial.print("Y is: ");
67      Serial.println(y,DEC);
68      delay(2000);
69
70      if ((y<= 142 && y>= 135) && (x<= 190 && x>= 175))
71      {
72         int buttonState = digitalRead(pushButton);
73         Serial.println(buttonState);
```

Kuvassa 17 rivillä 75 on if-lauseke, jonka tehtävä on tarkistaa onko Arduinon tuloon kytketty kytkin päällä, jotta liikesarja voidaan aloittaa. Jos kytkin on painettuna, voidaan liikesarja suorittaa ja riviltä 81 alkaa liikefunktioiden kutsusarja. Kytkimen tarkoitus on simuloida turvakytkintä järjestelmässä.

Liikefunktiot on esitelty ohjelmalle jo alustusosiossa ja niiden rakenne koostuu nimestä, edellisestä servonarvosta sekä halutusta servonarvosta. Lopuksi palautettu arvo tallennetaan ohjelmaan servonarvoksi, jota käytetään seuraavalla kutsukerralla funktion edellisenä arvona. Kun kaikki ohjelmakoodiin kuuluvat liikkeet on suoritettu, niin ohjelmasilmukka pitää 200ms tauon, tauon tarkoitus on antaa Arduinolle hetki sarjaliikennepuskurin tyhjentymiseen.

Kun liikesarja on suoritettu, lähettää Arduino sarjaväylään koodin ’18’ jonka tarkoitus on pyytää kuvantunnistukselta seuraavat koordinaatit. Jos koordinaatit eivät täsmää Kuvassa 15 rivillä 70 asetettuihin arvoihin, suorittaa ohjelma kuvassa 17 rivillä 104 olevan else-lausekkeen ja pyytää konenäöltä uudet koordinaatit.

Kuva 17. Arduino koodi, liikeohjelma.

```
75        if(buttonState == 1)
76
77          {
78
79
80
81            varsilarvo = varsilliike(varsilarvo,90);
82            varsi2arvo = varsi2liike(varsi2arvo,110);
83            varsi3arvo = varsi3liike(varsi3arvo,120);
84            kaantoarvo = kaantoliike(kaantoarvo,90);
85            kouraarvo = kouraliike(kouraarvo,135);
86            varsi2arvo = varsi2liike(varsi2arvo,90);
87            varsi3arvo = varsi3liike(varsi3arvo,110);
88            varsilarvo = varsilliike(varsilarvo,138);
89            kouraarvo = kouraliike(kouraarvo,55);
90            varsilarvo = varsilliike(varsilarvo,90);
91            kaantoarvo = kaantoliike(kaantoarvo,5);
92            varsilarvo = varsilliike(varsilarvo,125);
93            kouraarvo = kouraliike(kouraarvo,136);
94            varsilarvo = varsilliike(varsilarvo,80);
95            varsi3arvo = varsi3liike(varsi3arvo,180);
96
97            delay(200);
98
99             Serial.write(18);
100            Serial.println("kuittaus");
101         }
102     }
103
104     else
105     {
106        Serial.write(18);
107        Serial.println("kuittaus");
108     }
```

Kuvassa 18 on esitetty liikealiohjelma käännön osalta ja kaikki liikealiohjelmat ovat identtisiä, joten kaikkia niistä en esittele erikseen.

Liikealiohjelma vertailee aluksi, onko edellinen arvo suurempi vai pienempi kuin pyydetty arvo. Tämän jälkeen ohjelma suorittaa for-silmukan joko vähentäen tai lisäten servon arvoa ja lähettää arvon servon ohjausjohtimeen. Riveillä 130 ja 140 olevat ’delay’ käskyt toimivat liikenopeudensäätönä ja muuttuja ’t’ asetetaan ohjelman alustusvaiheessa. For-silmukka tallentaa servon ’value’ arvon ’retval’ muuttujaksi ja lopuksi aliohjelma palauttaa ’retval’ muuttujan arvon takaisin pääohjelmaan.

Kuva 18. Arduino koodi, liikealiohjelma.

```
122  int kaantoliike(int edarvo, int arvo)
123  {
124     int retval;
125     if(edarvo <= arvo)
126     {
127     for(int value = edarvo; value <= arvo; value ++)
128    {
129      kaanto.write(value);
130      delay(t);
131      retval = value;
132    }
133    }
134
135    else
136    {
137       for(int value = edarvo; value >= arvo; value --)
138    {
139      kaanto.write(value);
140      delay(t);
141      retval = value;
142    }
143    }
144    return retval;
145  }
```

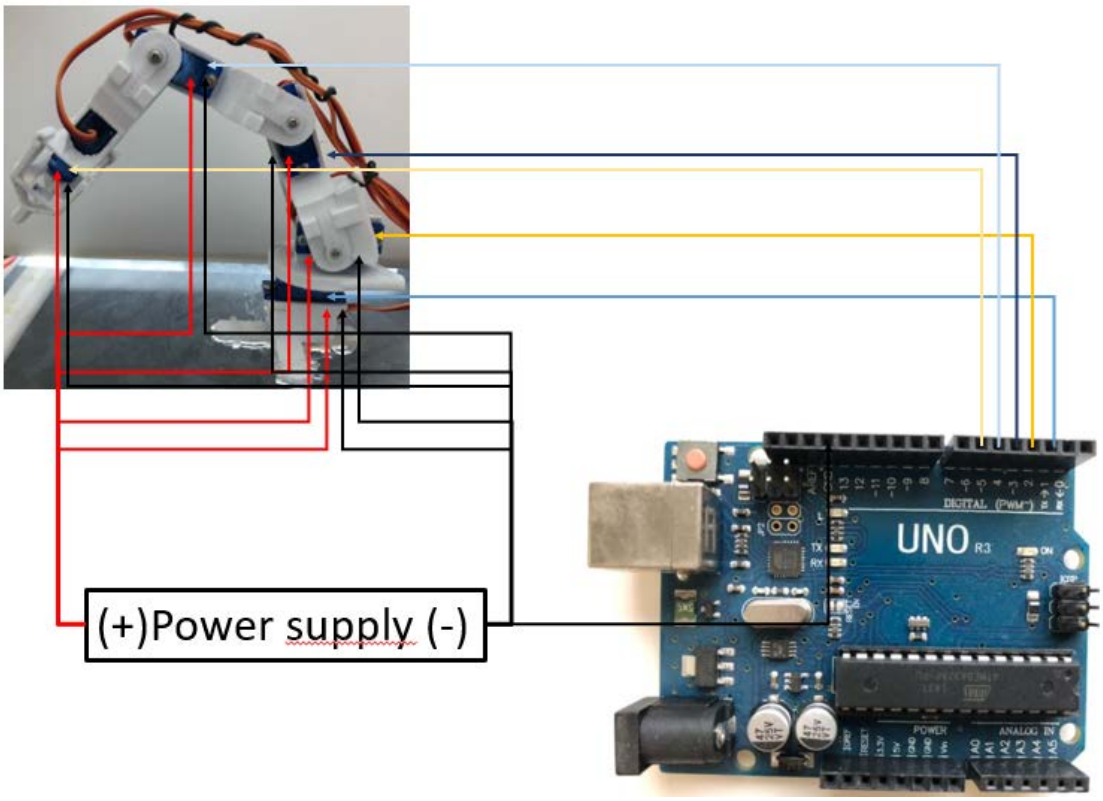
6.3 Robottitarttuja

Robotti-varsi on varustettu mikroservoilla, joiden ohjaaminen on suoraan mahdollista Arduino-ohjaimella. Kuitenkin liikeratakokeiden aikana huomasin servoissa epätarkkuutta, joka osaltaan vaikeutti kappaleen poimimista käytettäessä vapaata paikoitusta. Ratkaisuksi ongelmaan valitsin kohteen sijoittamisen alueelle ja robotin ohjaamisen kiinteästi valittuihin pisteisiin, tämä ratkaisu vastaa toiminnaltaan enemmän kohteen tunnistamista värin perusteella ja sen tiedon liittämistä osaksi automaatiojärjestelmän toimintaa. Robottivarsi

toimii siis projektissa vain toimilaitteena, joka voitaisiin korvata indikaattorilla tai toisen mallisella toimilaitteella sovelluksesta riippuen. Kuva 18

Servo kytkentä vaatii jännitteensyötön, miinuksen, sekä PWM-signaalin jolla määritetään servon haluttu positio. Järjestelmä sisältää viisi (5) servoa joiden syöttöjännitejohtimet on kytketty virtalähteen plus johtimeen ja miinusjohtimet virtalähteen miinukseen. Ohjainsignaali syötetään Arduinolta pinneistä 2,3,4,5,6. Lisäksi Arduinon miinusjohdin on kytketty virtalähteen miinukseen, jotta vältytään sähköisiltähäiriöiltä. Kuva 19

Kuva 19. Servojen kytkentä.



6.4 Tulokset

Johtuen ympäröivistä olosuhteista (CoVid19), oli testilaitteiston rakentamisessa käytettävä luovia menetelmiä. Kameralaitteistonteline rakentui kuumaliiman avulla vanhoista pahvipakkauksista ja valaisuun käytin hajonneesta LED-varoitusvalosta saatuja LED- sarjoja. Testilaitteistonvirransyöttönä toimii säädettävä laboratoriovirtalähde, joka virroittaa servot

sekä valaisun. Raspberry Pi -kortti on virroitettu omalla 2 ampeerin virtalähteellä ja Arduino-kortti saa virtansa Raspberry Pi-kortin USB-portista.

Laitteiston kasaamisen jälkeen kalibroin kohteentunnistuksenvärin sekä kohteen poimintapisteen. Robottitarttujankoura vaati myös hieman säätöä, koska liike piti saada niin suureksi kuin mahdollista, jotta kohteen poiminta oli mahdollista kääntöservon epätarkkuudesta huolimatta.

Tunnistamisen epätarkkuuteen vaikutti myös valaistus sekä kappaleen sijainti, tämä osaltaan johti siihen, että tunnistuspisteen X/Y koordinaatit eivät olleet jokaisella tunnistuskerralla täysin identtiset. Ongelmaan soveltui ratkaisuksi vertailulauseke, jolla voitiin määrittää tunnistettavien koordinaattien yläarvot ja ala-arvot.

Lopulta kohteen poiminta onnistui ja operaatio oli toistettavissa useamman kerran, myös järjestelmän uudelleenkäynnistyksen jälkeen. Lisäksi järjestelmä saa kappaleen koordinaatit, vaikka kohde ei ole poiminta alueella, ja tämä mahdollistaa sovelluksen laajentamisen myöhemmin.

6.5 Jatkokehitys

Osa projektiin tarkoitetuista toiminnoista piti jättää pois tarttujan epätarkkuuden vuoksi, mutta tarkoituksenani on lisätä ne osaksi laitteistoa myöhemmässä kehityksessä. Robotti-tarttuja on yksi kehityskohteista ja tarkoituksenani on vaihtaa tarttuja versioon, jossa on mahdollisuus takaisinkytkennälle. Tämä tarkoittaa, että tarttujaan asennetaan anturit, joiden avulla ohjelman on mahdollista tietää tarttuja tarkka asema.
Takaisinkytkennän lisäämisen jälkeen lisään ohjelmaan toiminnon, jolla kappale voidaan poimia vapaasti mistä tahansa sijainnista kameran näköalueella. Jotta kappale voidaan poimia vapaasti, tarvitsee tarttujanohjaimeen tehdä algoritmi, joka laskee tarttujan varsien asennot niin, että tarttuja löytää x ja y koordinaateilla kerrotun aseman.
Muita kehityskohteita, joita olen suunnitellut, ovat Raspberry Pi 2 -ohjaimen vaihtaminen Raspberry Pi 3 tai 4 -alustaan, sekä valaistuksenohjauksen liittäminen Raspberry Pi ohjaimeen.

Toiminnallisena muutoksena on tarkoitus muuttaa ohjelman toimintaa niin, että tunnistus tapahtuisi videokuvasta, eikä still-kuvasta, kuten nykyisessä versiossa. Videokuvan käyttäminen vaatii laitteistolta paljon resursseja, tätä vaatimusta voidaan kuitenkin alentaa laskemalla kuvan resoluutiota ja käyttämällä laskennallisesti kevyempiä algoritmeja. Tästä johtuen kameran vaihtaminen tehokkaampaan tuskin on tarpeellista, koska käsiteltävän kuvadatanresoluutiota lasketaan jo nykyisessäkin versiossa merkittävästi.
Tämän projektin myötä laitteisto on saanut alkupisteen, jonka ympärille on mahdollista rakentaa ja lisätä useita toimintoja.

Lähteet

Abenza, J. A. (2018). *Medium.com.* Noudettu osoitteesta https://medium.com/neurosapiens/segmentation-and-classification-with-hsv-8f2406c62b39

Alho, T.;Neittaanmäki, P.;Hänninen, P.;& Tammilehto, O. (2018). *Jyväskylän yliopisto.* Noudettu osoitteesta https://www.jyu.fi/it/fi/tutkimus/julkaisut/tekes-raportteja/tekoaly_ja_palvelurobotiikka.pdf

Arduino. (2018). *Arduino.cc.* Noudettu osoitteesta https://www.arduino.cc/en/Guide/ArduinoMega2560

Badami, V. (nd). *Hacker Earth*. Noudettu osoitteesta https://www.hackerearth.com/blog/developers/arduino-programming-for-beginners/

Computer Hope. (2020). *Computer Hope* . Noudettu osoitteesta https://www.computerhope.com/history/processor.htm

Corke, P. (2017). *Robotics, Vision and Control.* Queensland: Springer.

Digitalkamera Museum. (n.d). *Digitalkamera Museum*. Noudettu osoitteesta https://www.digitalkameramuseum.de/en/history

Illlmatieteen laitos. (2021). *VALO JA SPEKTRI.* Noudettu osoitteesta https://space.fmi.fi/oppimateriaali/envisat/valonsade/spektri.html

Information Security Newspaper. (2020). *Information Security Newspaper*. Noudettu osoitteesta https://www.securitynewspaper.com/2020/01/20/google-yahoo-microsoft-intel-ibm-sony-honda-toyota-among-others-affected-by-critical-vulnerabilities-in-opencv/

International Federation of Robotics. (n.d). *ifr.org.* Noudettu osoitteesta https://ifr.org/img/office/Industrial_Robots_2016_Chapter_1_2.pdf

iso.org. (2021). *ISO*. Noudettu osoitteesta https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:en

Janesick, J. R. (2001). *Scientific charge-coupled device.* The International Society of Optical Engineering.

Lucid vision lab. (2021). *Tech brief.* Noudettu osoitteesta https://thinklucid.com/tech-briefs/understanding-digital-image-sensors/

OpenCV. (2021). *OpenCV kotisivu*. Noudettu osoitteesta https://opencv.org/about/

Opetushallitus. (2021). *Puutuoteprosessit*. Noudettu osoitteesta http://www03.edu.fi/oppimateriaalit/puutuoteteollisuus/automaatio/konenako/index.html

Pi, T. (nd.). *The Pi io*. Noudettu osoitteesta https://thepi.io/how-to-install-raspbian-on-the-raspberry-pi/

Pramanick, S. (2019). *geeksforgeeks*. Noudettu osoitteesta https://www.geeksforgeeks.org/history-of-python/

Raspberry Pi Foundation. (2021). *Raspberry.org.* Noudettu osoitteesta https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

Raspbian. (2021). *Rasbian.org*. Noudettu osoitteesta https://www.raspbian.org/RaspbianAbout

The Washington post. (2020). *The Washington post*. Noudettu osoitteesta https://www.washingtonpost.com/local/obituaries/russell-kirsch-computer-scientist-who-scanned-first-digital-image-dies-at-91/2020/08/13/a234a22c-dccd-11ea-b205-ff838e15a9a6_story.html

Tokyo Electron. (n.d). *Nanotec Museum*. Noudettu osoitteesta https://www.tel.com/museum/exhibition/principle/cmos.html

Liite 1: Arduino-koodilistaus

```
//Tuodaan servo kirjasto
#include <Servo.h>

//Alustetaan servot
Servo kaanto;
Servo varsi1;
Servo varsi2;
Servo varsi3;
Servo koura;

//Alustetaan painonappi pin7:ään
int pushButton = 7;

//Funktioiden alustus
int kaantoliike(int[],int[]);
int varsi1liike(int[],int[]);
int varsi2liike(int[],int[]);
int varsi3liike(int[],int[]);
int kouraliike(int[],int[]);

//Lähtö arvot
int kaantoarvo = 5;
int varsi1arvo = 90;
int varsi2arvo = 90;
int varsi3arvo = 144;
int kouraarvo = 140;

//Liikenopeus säätö
int t = 20;

//Alustus
void setup() {

  //Liikeiden setup asennot, piste josta liike aloitetaan
  kaantoarvo = kaantoliike(kaantoarvo,5);
  kouraarvo = kouraliike(kouraarvo,135);
  varsi1arvo = varsi1liike(varsi1arvo,80);
  varsi3arvo = varsi3liike(varsi3arvo,180);

  //Sarjaportin alustus 9600baud
  Serial.begin(9600);

  //Servojen pinnit
  kaanto.attach(2);
  varsi1.attach(3);
  varsi2.attach(4);
  varsi3.attach(5);
  koura.attach(6);

  //Määritetää pushButton (pin7) tuloksi
  pinMode(pushButton, INPUT);

}

//Ohjelma loop
void loop() {


  //Alustetaan muuttuja 'x' arvoon 0
  //Alustetaan muuttuja 'y' arvoon 0
  //Alustetaan merkki bufferi 4 tavun mittaiseksi
  int x = 0;
  int y = 0;
  char buffer[4];
```

```
  //Jos sarjaportti on aktiivinen
  //Luetaan 4 tavua puskuriin sarjaportista
  if (Serial.available() > 0) {
    int size = Serial.readBytesUntil('\n', buffer, 4);

     //Luetaan arvo 'x' puskurin ensimmäisestä tavusta
     //Tulosteaan teksti "X is: "
     //Tulostetaan muuttuja 'x' desimaalina
     x = buffer[1];
     Serial.print("X is: ");
     Serial.println(x,DEC);

     //Luetaan arvo 'x' puskurin ensimmäisestä tavusta
     //Tulosteaan teksti "Y is: "
     //Tulostetaan muuttuja 'y' desimaalina
     //2000ms tauko
     y = buffer[3];
     Serial.print("Y is: ");
     Serial.println(y,DEC);
     delay(2000);

     //If-lauseke joka tarkistaa onko kappaleen koordinaatit oikeat (y=135-142 ja x=175-
190)
     if ((y<= 142 && y>= 135) && (x<= 190 && x>= 175))
     {
        //Alustetaan muuttuja 'buttonState' johon luetaan digitali arvo pushButton
pinnistä (pin 7)
        //Tulostetaan konsoliin muuttuja 'buttonState'
        int buttonState = digitalRead(pushButton);
        Serial.println(buttonState);

        //If-lause jos buttonState on arvossa 1 (Nappi painettuna)
        if(buttonState == 1)

          {
            //Kutsutaan liike funktiota ja annetaan sille alkuarvoksi edellinen
tallennettu arvo
            //sekä haluttu arvo mihin halutaan liikkua.
            //Lopuksi tallennettaan haluttu arvo edelliseksi arvoksi.

            //Tehdään liikesarja varsilla
            varsi1arvo = varsi1liike(varsi1arvo,90);
            varsi2arvo = varsi2liike(varsi2arvo,110);
            varsi3arvo = varsi3liike(varsi3arvo,120);
            kaantoarvo = kaantoliike(kaantoarvo,90);
            kouraarvo = kouraliike(kouraarvo,135);
            varsi2arvo = varsi2liike(varsi2arvo,90);
            varsi3arvo = varsi3liike(varsi3arvo,110);
            varsi1arvo = varsi1liike(varsi1arvo,138);
            kouraarvo = kouraliike(kouraarvo,55);
            varsi1arvo = varsi1liike(varsi1arvo,90);
            kaantoarvo = kaantoliike(kaantoarvo,5);
            varsi1arvo = varsi1liike(varsi1arvo,125);
            kouraarvo = kouraliike(kouraarvo,136);
            varsi1arvo = varsi1liike(varsi1arvo,80);
            varsi3arvo = varsi3liike(varsi3arvo,180);

            //200ms viive jotta sarjaväylä bufferi ehtii tyhjentyä
            delay(200);

             //Kirjoitetaan sarjaväylään arvo '18', joka pyytää openCv:ltä uusia
koordinaatteja
             //Tulostetaan käyttäjälle ilmoitus 'kuittaus' konsoliin
             Serial.write(18);
             Serial.println("kuittaus");
          }
     }

     else
```

```
     {
        //Kirjoitetaan sarjaväylään arvo '18', joka pyytää openCv:ltä uusia
koordinaatteja
        //Tulostetaan käyttäjälle ilmoitus 'kuittaus' konsoliin
        Serial.write(18);
        Serial.println("kuittaus");
     }

     //Tulostetaan käyttäjälle ilmoitus 'New position:'
     //Nollataan bufferin tavu 1
     //Nollataan bufferin tavu 3
     Serial.println("New position: ");
     buffer[1] = 0x0000000;
     buffer[3] = 0x0000000;

  }
}


//---------------------------------------------------------------
//FUNKTIOT
//---------------------------------------------------------------

//Alustetaan funktio 'kaantoliike' ja siihen kuuluvat muuttujat
int kaantoliike(int edarvo, int arvo)
{
   //Alustetaan 'retval' muuttuja
   int retval;
   //If-lauseke jos 'edarvo' on pienempi tai yhtäsuuri kuin 'arvo'
   if(edarvo <= arvo)
   {
   //For funktio liikkeelle.
   //Alkuarvo on 'edarvo', lopetusarvo on 'arvo' ja jokaisella iteraatio kerralla
lisätään arvoon 1
   for(int value = edarvo; value <= arvo; value ++)
  {
    //Kirjoitetaan muuttuja 'value' servolle
    //Viive muutujan 't' verran (20ms)
    //Tallennetaan muuttuja 'value' muuttujaksi 'retval'
    kaanto.write(value);
    delay(t);
    retval = value;
  }
  }

  //Else-lauseke jos 'edarvo' on suurempi kuin 'arvo'
  else
  {
     //For funktio liikkeelle.
     //Alkuarvo on 'edarvo', lopetusarvo on 'arvo' ja jokaisella iteraatio kerralla
vähennetään 1 arvosta
     for(int value = edarvo; value >= arvo; value --)
  {
    //Kirjoitetaan muuttuja 'value' servolle
    //Viive muutujan 't' verran (20ms)
    //Tallennetaan muuttuja 'value' muuttujaksi 'retval'
    kaanto.write(value);
    delay(t);
    retval = value;
  }
  }
  //Palautetaan muuttuja 'retval' alkuperäiseen ohjelmaan
  return retval;
}



//Alustetaan funktio 'varsi1liike' ja siihen kuuluvat muuttujat
int varsi1liike(int edarvo, int arvo)
{
```

```
   //Alustetaan 'retval' muuttuja
   int retval;
   //If-lauseke jos 'edarvo' on pienempi tai yhtäsuuri kuin 'arvo'
   if(edarvo <= arvo)
   {
   //For funktio liikkeelle.
   //Alkuarvo on 'edarvo', lopetusarvo on 'arvo' ja jokaisella iteraatio kerralla
lisätään arvoon 1
   for(int value = edarvo; value <= arvo; value ++)
  {
    //Kirjoitetaan muuttuja 'value' servolle
    //Viive muutujan 't' verran (20ms)
    //Tallennetaan muuttuja 'value' muuttujaksi 'retval'
    varsi1.write(value);
    delay(t);
    retval = value;
  }
  }

  //Else-lauseke jos 'edarvo' on suurempi kuin 'arvo'
  else
  {
     //For funktio liikkeelle.
     //Alkuarvo on 'edarvo', lopetusarvo on 'arvo' ja jokaisella iteraatio kerralla
vähennetään 1 arvosta
     for(int value = edarvo; value >= arvo; value --)
  {
    //Kirjoitetaan muuttuja 'value' servolle
    //Viive muutujan 't' verran (20ms)
    //Tallennetaan muuttuja 'value' muuttujaksi 'retval'
    varsi1.write(value);
    delay(t);
    retval = value;
  }
  }
  //Palautetaan muuttuja 'retval' alkuperäiseen ohjelmaan
  return retval;
}



//Alustetaan funktio 'varsi2liike' ja siihen kuuluvat muuttujat
int varsi2liike(int edarvo, int arvo)
{
   //Alustetaan 'retval' muuttuja
   int retval;
   //If-lauseke jos 'edarvo' on pienempi tai yhtäsuuri kuin 'arvo'
   if(edarvo <= arvo)
   {
   //For funktio liikkeelle.
   //Alkuarvo on 'edarvo', lopetusarvo on 'arvo' ja jokaisella iteraatio kerralla
lisätään arvoon 1
   for(int value = edarvo; value <= arvo; value ++)
  {
    //Kirjoitetaan muuttuja 'value' servolle
    //Viive muutujan 't' verran (20ms)
    //Tallennetaan muuttuja 'value' muuttujaksi 'retval'
    varsi2.write(value);
    delay(t);
    retval = value;
  }
  }

  //Else-lauseke jos 'edarvo' on suurempi kuin 'arvo'
  else
  {
     //For funktio liikkeelle.
     //Alkuarvo on 'edarvo', lopetusarvo on 'arvo' ja jokaisella iteraatio kerralla
vähennetään 1 arvosta
     for(int value = edarvo; value >= arvo; value --)
```

```
  {
    //Kirjoitetaan muuttuja 'value' servolle
    //Viive muutujan 't' verran (20ms)
    //Tallennetaan muuttuja 'value' muuttujaksi 'retval'
    varsi2.write(value);
    delay(t);
    retval = value;
  }
  }
  //Palautetaan muuttuja 'retval' alkuperäiseen ohjelmaan
  return retval;
}



//Alustetaan funktio 'varsi3liike' ja siihen kuuluvat muuttujat
int varsi3liike(int edarvo, int arvo)
{
   //Alustetaan 'retval' muuttuja
   int retval;
   //If-lauseke jos 'edarvo' on pienempi tai yhtäsuuri kuin 'arvo'
   if(edarvo <= arvo)
   {
   //For funktio liikkeelle.
   //Alkuarvo on 'edarvo', lopetusarvo on 'arvo' ja jokaisella iteraatio kerralla
lisätään arvoon 1
   for(int value = edarvo; value <= arvo; value ++)
  {
    //Kirjoitetaan muuttuja 'value' servolle
    //Viive muutujan 't' verran (20ms)
    //Tallennetaan muuttuja 'value' muuttujaksi 'retval'
    varsi3.write(value);
    delay(t);
    retval = value;
  }
  }

  //Else-lauseke jos 'edarvo' on suurempi kuin 'arvo'
  else
  {
     //For funktio liikkeelle.
     //Alkuarvo on 'edarvo', lopetusarvo on 'arvo' ja jokaisella iteraatio kerralla
vähennetään 1 arvosta
     for(int value = edarvo; value >= arvo; value --)
  {
    //Kirjoitetaan muuttuja 'value' servolle
    //Viive muutujan 't' verran (20ms)
    //Tallennetaan muuttuja 'value' muuttujaksi 'retval'
    varsi3.write(value);
    delay(t);
    retval = value;
  }
  }
  //Palautetaan muuttuja 'retval' alkuperäiseen ohjelmaan
  return retval;
}



//Alustetaan funktio 'kouraliike' ja siihen kuuluvat muuttujat
int kouraliike(int edarvo, int arvo)
{
   //Alustetaan 'retval' muuttuja
   int retval;
   //If-lauseke jos 'edarvo' on pienempi tai yhtäsuuri kuin 'arvo'
   if(edarvo <= arvo)
   {
   //For funktio liikkeelle.
```

```
   //Alkuarvo on 'edarvo', lopetusarvo on 'arvo' ja jokaisella iteraatio kerralla
lisätään arvoon 1
   for(int value = edarvo; value <= arvo; value ++)
  {
    //Kirjoitetaan muuttuja 'value' servolle
    //Viive muutujan 't' verran (20ms)
    //Tallennetaan muuttuja 'value' muuttujaksi 'retval'
    koura.write(value);
    delay(t);
    retval = value;
  }
  }

  //Else-lauseke jos 'edarvo' on suurempi kuin 'arvo'
  else
  {
     //For funktio liikkeelle.
     //Alkuarvo on 'edarvo', lopetusarvo on 'arvo' ja jokaisella iteraatio kerralla
vähennetään 1 arvosta
     for(int value = edarvo; value >= arvo; value --)
  {
    //Kirjoitetaan muuttuja 'value' servolle
    //Viive muutujan 't' verran (20ms)
    //Tallennetaan muuttuja 'value' muuttujaksi 'retval'
    koura.write(value);
    delay(t);
    retval = value;
  }
  }
  //Palautetaan muuttuja 'retval' alkuperäiseen ohjelmaan
  return retval;
}
```

Liite 2: Python-koodilistaus.

```
#  Tuodaan kirjastot
from __future__ import print_function #Tuodaan "print" funktio Python 3 ympäristöstä
from picamera import PiCamera #PiCamera kirjasto
import imutils #Tuodaan perus kuvankäsittely funktiot
import time #Time kirjasto mahdollistaa viiveiden käytön
import cv2 #Tuodaan konenäkökirjasto (OpenCV)
import os #Tuodaan kirjasto joka mahdollistaa käyttöjärjestelmän ominaisuuksien käytön
import serial #Sarjaviestintä kirjasto
import struct #Kirjasto joka mahdollistaa structuurien käytön

#Määritellään sarjaportti: ttyACM0 = arduino, 9600 = baudrate
ser = serial.Serial('/dev/ttyACM0', 9600)

#Alustetaan PiCamera cameraksi
camera = PiCamera()

#  Määritetään koordinaattien tuostus funktio
#  Aliohjelman nimi ja syötetyt arvot
#  Tulostetaan x ja y koordinaatit

def mapPos (x, y):
    print ("Coordinates X = {0} and Y =  {1}".format(x, y))

# Määritetään tunnistuksen värialue
# HSV avaruudessa (H=0-179, S=0-255, V=0-255)
# 'colLow' Alempi arvo (H,S,V)
# 'colUp' Ylempi arvo (H,S,V)

colLow = (100, 100, 100)
colUp = (120, 255, 255)

# Kohteen tunnistus loop
while True:

    #Alustetaan muuttuja i arvoon 0
    #Käynnistetään kamera
    #Otetaan kuva ja tallennetaan se
    #Sammutetaan kamera
    i=0
    camera.start_preview()
    camera.capture('/home/pi/snapshot.jpg')
    camera.stop_preview()

    #Luetaan kuva tiedostosta
    #skaalataan kuvan leveys 500 kuvapisteeseen
    #muutetaan värikoordinaatisto
    frame = cv2.imread('snapshot.jpg', 1)
    frame = imutils.resize(frame, width=500)
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # Luodaan maski tunnistukselle
    # Poistetaan maskista häiriöt käyttämällä "erode" ja "dilate" funktioita
    mask = cv2.inRange(hsv, colLow, colUp)
    mask = cv2.erode(mask, None, iterations=1)
    mask = cv2.dilate(mask, None, iterations=1)


    #Etsitään maskista ääriviivoja
    #nopeuden vuoksi luetaan vain ulommaiset ääriviivat ja kulma pisteet.
    #Tarkistetaan ja sovitetaan 'cnts' muuttuja CV version mukaan.
    #Alustetaan muuttuja 'center'.
    #2 sekunnin tauko ohjelman rauhoittamiseksi.
    cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    cnts = cnts[0] if imutils.is_cv2() else cnts[1]
    center = None
    time.sleep(2)
```

```
#If lauseke jos ääriviivoja havaitaan
if len(cnts) > 0:

      #Etsitään suurin muoto
      #Lasketaan ympyrän koko
      #Alustetaan muuttuja M
      #Lasketaan keskipiste moments-arvojen avulla
      c = max(cnts, key=cv2.contourArea)
      ((x, y), radius) = cv2.minEnclosingCircle(c)
      M = cv2.moments(c)
      center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))

      # If lauseke jos halkaisija on yli 10 kuvapistettä
      #Piirretään ympyrä x ja y koordinaattien mukaan ja säteeksi valitaan 'radius'
      #Määritetään väriksi '255,0,255*
      #Piirretään piste kohteen 'center'-pisteen päälle värillä '255,255,0'
      if radius > 10:
            cv2.circle(frame, (int(x), int(y)), int(radius),
                  (255, 0, 255), 2)
            cv2.circle(frame, center, 5, (255, 255, 0), -1)

            #Servon ohjaus
            #Kirjotetaan sarjaväylään strucktuuri joka sisältää x ja y arvot.
            ##formaatti on 2kpl Big endian unsigned short.
            ser.write(struct.pack('>2H', int(x),int(y)))

            #Kutsutaan 'mapPos' funktiota ja annetaan sille x ja y koordinaatit
            #Nollataan sarjaportin bufferi
            mapPos(int(x), int(y))
            ser.flushOutput()

            #While loop jolla odotetaan arduinon kuittausta tehdystä liikkeestä
            #Alustetaan ja luetaan sarjaväylästä muuttuja number
            #Tarkistetaan että arvo on 18
            #Tulostetaan viesti 'ok' konsoliin
            #Muutetaan muuttuja 'i' arvoon 1 jotta while loop päättyy
            while i==0:
                number = ser.read()
                if int.from_bytes(number,byteorder='big')==18:
                    print("ok")
                    i=1


# Tulostetaan kuva kehykseen
#Kehyksen nimi 'Frame' ja tulostettava kuva 'frame'
#1 sekunnin tauko ohjelman rauhoittamiseksi
cv2.imshow("Frame", frame)
time.sleep(1)
```

Liite 3: Servojen kytkentä