LAPIN AMK
Lapland University of Applied Sciences

USB-KAMERAN KÄYTTÖ VÄRIN- JA KUVANTUNNISTUKSESSA
Fischertechnik ROBOTICS TXT Controller

Pylkkönen Aleksi

Älykkään elinympäristön teknologiat -hanke
Tekniikka ja liikenne
Tieto- ja viestintätekniikka
Insinööri (AMK)

2017

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tekniikka ja liikenne
Tieto- ja viestintätekniikka
Insinööri (AMK)

Tekijä	Aleksi Pylkkönen	Vuosi	2017
Ohjaaja	Anssi Ylinampa		
Toimeksiantaja	Anssi Ylinampa, ÄET-hanke		
Työn nimi	USB-kameran käyttö värin- ja kuvantunnistuksessa		
Sivu- ja liitesivumäärä	47		

Opinnäytetyössä tarkoituksena oli tutkia Fischertechnikin opetuskäyttöön tarkoitettujen palikkarobottien käyttöä värin- ja kuvantunnistuksessa, käyttäen robotin omaa USB-kameraa ja Fischertechnikin luomaa ROBOPro-ohjelmointiympäristöä.

Värin- ja kuvantunnistuksen testaamista varten ohjelmoitiin pieniä testiohjelmia, joiden avulla niiden ominaisuuksia oli mahdollista tutkia. Robotin ja ohjelmointiympäristön tietoperustana käytettiin lähinnä internetistä löytyvää materiaalia ja kirjoittajan omia tietoja sekä tarvittaessa aiheeseen liittyvää kirjallisuutta.

Tuloksena on tutkimusdokumentaatio, jossa käsitellään tutkittujen aiheiden toimintaa ja mahdollisuuksia. Dokumentaatiossa myös hieman peilataan tutkittavaa robottia, sen toimintaa ja ominaisuuksia, nykypäivänä opetuskäytössä olevaan Lego Mindstorms EV3 -robottiin.

Pohdinnassa käydään läpi tutkimuksen lopputulos sekä pohditaan Fischertechnikin robotin mahdollisuuksia korvata Legon robotin tai toimia sen rinnalla.

Avainsanat: Fischertechnik, kuvantunnistus, robotiikka, sensori, väritunnistus

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

School of Technology,
Communication and Transport
Degree Programme in Information
and Communication Technology
Bachelor of Engineering

Author Aleksi Pylkkönen Year 2017
Supervisor Anssi Ylinampa
Commissioned by Anssi Ylinampa, the ÄET project
Subject of thesis Using an USB Camera for Color and Picture Recognition
Number of pages 47

The purpose of this thesis was to study the use of Fischertechnik's teaching kits for color and image recognition using the robot's own USB camera and the ROBOPro programming environment created by Fischertechnik.

To test the color and image recognition, small test programs were programmed to study those properties. The knowledge base for the robot and the programming environment was mainly the material found on the Internet and the author's own information and related literature where needed.

The result was a research document dealing with the activities and opportunities of the subjects studied. In the documentation, the robot, its function and features are also slightly mirrored in Lego Mindstorms EV3 robot which is used for teaching. The outcome of the study and the possibilities of Fischertechnik's robot to replace or operate with the Lego's robot were discussed.

Key words Fischertechnik, color recognition, picture, robotics, sensor

SISÄLLYS

1 JOHDANTO 6

2 ROBOTICS TXT CONTROLLER JA ROBOPRO 8

2.1 ROBOTICS TXT Controller 8
2.2 ROBOPro 9
2.2.1 Ensiaskeleet 10
2.2.2 Ohjelmointitasot 12

3 USB-KAMERA 14

3.1 USB-kameraan tutustuminen 14
3.2 USB-kameran sensorit 15

4 VÄRINTUNNISTUS 16

4.1 Päävärien tunnistaminen 16
4.1.1 Päävärien tunnistamisen kokeilu lainatulla ratkaisulla 16
4.1.2 Päävärien tunnistamisen tutkiminen omalla ratkaisulla 17
4.2 Päävärien erottaminen mustavalkoisuudesta 18
4.3 Välivärien tunnistaminen 20
4.4 Värintunnistuksessa ilmenneitä ongelmia 22
4.4.1 Valaistuksen vaikutus väreihin 22
4.4.2 Kuvan värivirheet 23
4.4.3 Väriarvojen muuttuminen kesken ohjelman 25
4.5 Värintunnistus Mindstorms EV3:lla 26

5 KUVANTUNNISTUS 29

5.1 Ympyräntunnistus 29
5.1.1 Ympyräntunnistuksen asetukset 29
5.1.2 Ensimmäinen ympyräntunnistuselementin testiohjelma 33
5.1.3 Ympyrään kohdistava testiohjelma 35
5.1.4 Ympyräntunnistuksen ongelmia 37
5.2 Viivantunnistus 38
5.2.1 Viivantunnistuksen käyttö 38
5.2.2 Viivantunnistuksen asetukset 40
5.3 Liikkeentunnistus 42
5.4 Kuvantunnistus Mindstorms EV3:lla 44

6 POHDINTA 46

LÄHTEET 47

1 JOHDANTO

Opinnäytetyöni aihe liittyy kesällä 2017 tieto- ja viestintätekniikan koulutusohjelman puolelle hankittuihin Fischertechnik GmbH:n Robotics TXT Controller -palikkarobotteihin. Aihetta minulle ehdotti opettajani Anssi Ylinampa, joka toimii sekä opinnäytetyöni ohjaajana että ÄET-hankkeen lähimpänä edustajana. Kiinnostuin aiheesta välittömästi, sillä en koskaan ollut kuullutkaan näistä roboteista, joten ajattelin opinnäytetyön olevan mahdollisuus oppia uutta ja päästä samalla sekä näpertelemään että ohjelmoimaan robotteja.

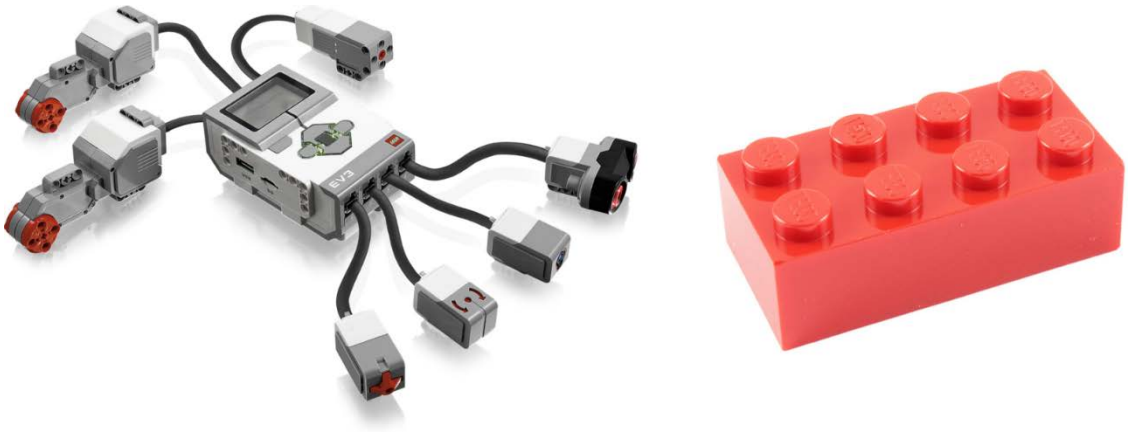
Robotit ovat osa Lapin ammattikorkeakoulun, Lapin liiton ja Euroopan Unionin ÄET eli Älykkään elinympäristön teknologiat -hanketta, joka rakentuu kolmesta samanaikaisesti toteutettavasta hankkeesta, jotka ovat Investoinnit, Kickstart ja Osaaminen. Hankkeiden yhteisenä tavoitteena on monialaisen ÄET-kehittämisympäristön muodostumisen ja tehokkaan toiminnan mahdollistaminen. Hankekokonaisuuden muodostavat neljä pääteemaa, jotka ovat Älykkäät järjestelmät, Älykäs energian hallinta, Digitaalinen rakennettu ympäristö ja Kestävä rakennettu ympäristö. (ÄET-hanke 2016a; 2016b; 2016c)

Opinnäytetyössäni tavoitteenani on tutkia sekä ROBOTICS TXT Controllerin värin- ja kuvantunnistuksen toimintaa, että siihen liittyviä mahdollisuuksia ja mahdollisia heikkouksia. Sivuutan työssäni myös olisiko näistä roboteista korvaamaan nykyisin opetuskäytössä olevia Lego Mindstorms EV3 –robotit tai toimimaan niiden rinnalla.

Käytän Fischertechnikin ROBOPro-ohjelmointiympäristöä ohjelmoidakseni tutkittavalle robotille testiohjelmia, joiden avulla pystyn testaamaan ja tutkimaan kyseisen robotin USB-kameran toiminnollisuuksia. Lopuksi peilaan saamiani tuloksia hieman Lego Mindstormsiin ja vertailen robotteja keskenään.

Fischertechnik GmbH on vuonna 1965 perustettu saksalainen yritys, jonka toiminta keskittyy samannimisten rakennussarjojen kehittämiseen ja valmistamiseen. Fischertechnik-rakennussarjoja myydään eri puolille maailmaa ja niiden käyttö keskittyy suuresti robotiikan, automaation ja ohjelmoinnin opetukseen. (Crunchbase 2017.)

Lego on vuonna 1932 perustettu tanskalainen rakennuspalikoiden kehittäjä ja valmistaja, joka tunnetaan ja tiedetään ympäri maailman. Vaikka tunnetuimmat Legon tuotteet ovatkin sen ikoniset rakennuspalikat, valmistaa yritys myös muunlaisia monimuotoisempia rakennussettejä. Näistä kehittyneimpiä ovat Legon Mindstorms rakennussarjat, joihin kuuluvat myös ohjelmoitavat tietokoneet (Kuvio 1). (Lego 2017c.)



Kuvio 1. Lego Mindstorms EV3 ja Legon ikoninen rakennuspalikka

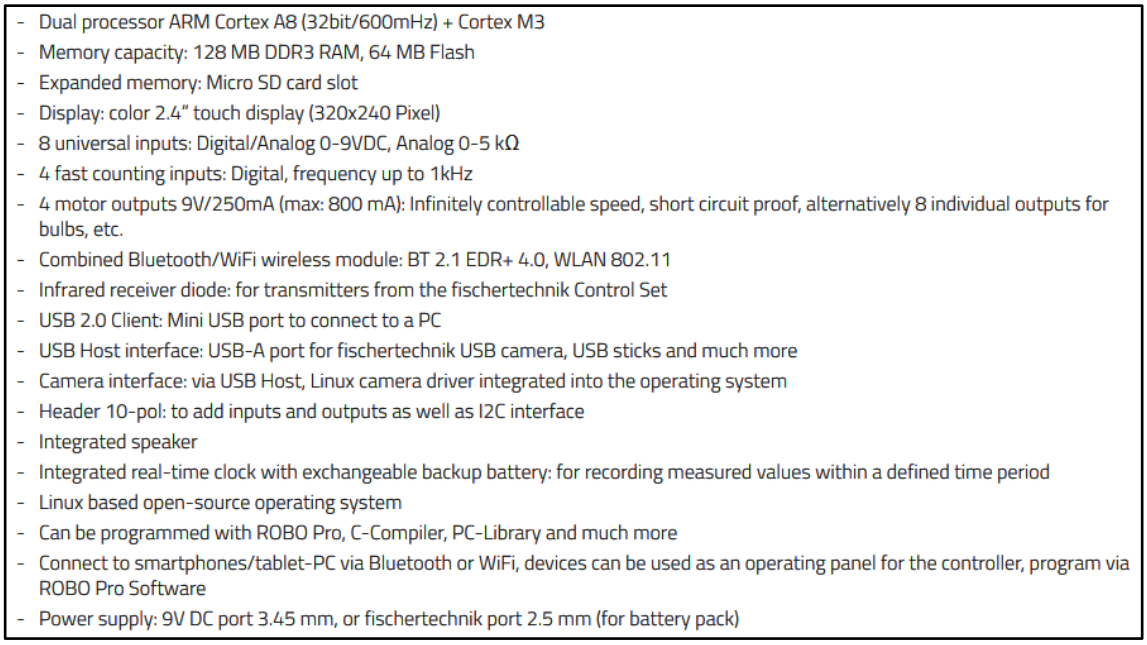
2 ROBOTICS TXT CONTROLLER JA ROBOPRO

2.1 ROBOTICS TXT Controller

Fischertechnikin ROBOTICS TXT Controller (Kuvio 2.) on kompakti tietokone, johon voidaan ladata ROBOPro-ohjelmointiympäristöllä luotuja ohjelmia ja liittää muita robottiin käytettäviä komponentteja, kuten moottoreita, kytkimiä, kameran, ja niin edelleen. Myös useampi robotti on mahdollista yhdistää toisiinsa. Robottia voidaan ohjata sen 2,4-tuumaisella, 320 x 240-pikselin kosketusnäytöllä, USB-väylän kautta PC:llä, tai vaihtoehtoisesti Bluetoothin tai WiFi:n kautta yhdistetyllä erillisellä ohjaimella, esimerkiksi älypuhelimella. (Fischertechnik GmbH 2017.)

Kuvio 2. ROBOTICS TXT Controller

Robotilla on kaksi 32-bittistä 600-megahertsin suoritinta, 128-megatavua RAM-muistia ja 64megatavua, microSD -kortilla laajennettavaa, flash -muistia (Kuvio 3.). Robotin käyttöjärjestelmä on Linux-pohjainen avoimen lähdekoodin käyttöjärjestelmä, ja ROBOPron lisäksi sille voidaan ohjelmoida ohjelmia myös RobotC:llä, käyttäen C-kielen kääntäjää. (Fischertechnik GmbH 2017.)

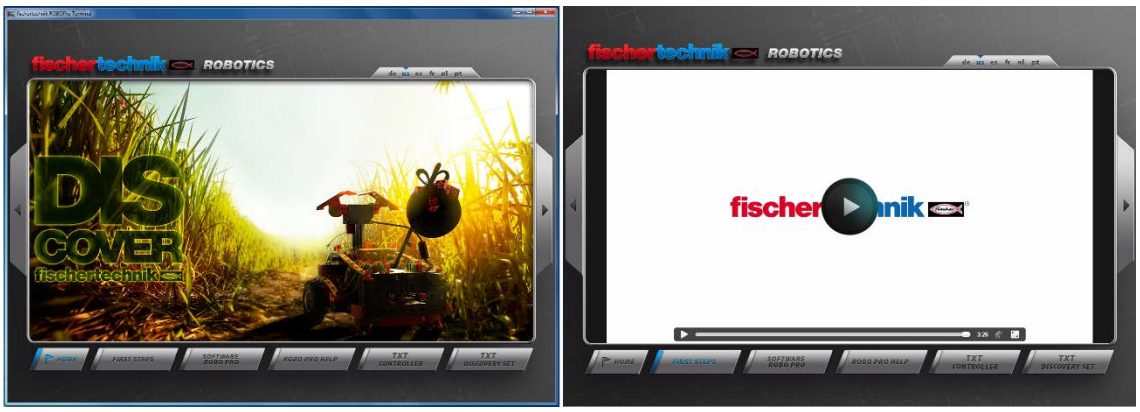


Kuvio 3. ROBOTICS TXT Controllerin järjestelmätiedot

2.2 ROBOPro

ROBOPro on Fischertechnik GmbH:n luoma ohjelmointiympäristö, jolla ROBOTICS TXT Controllerille voidaan ohjelmoida erilaisia ohjelmia. Ohjelmointiympäristö on aloittelijaystävällinen sen yksinkertaisen käyttöliittymän ja ohjelmointitavan ansiosta, sillä siinä ei tarvitse osata mitään varsinaista ohjelmointikieltä, vaan toiminta perustuu vuokaaviomaiseen ohjelmointiin. (Fischertechnik GmbH 2017.)

Aloitin ROBOProhon tutustumisen käynnistämällä ohjelman ”Fischertechnik ROBOPro Terminal” (Kuvio 4.), joka on Fischertechnikin aloitusterminaali. ROBOPro Terminalista löytyy ”First steps” (*Ensiaskeleet*) -kurssi sekä itse ROBOPro-ohjelmointiympäristö että ohjeita ROBOPron käyttöön.

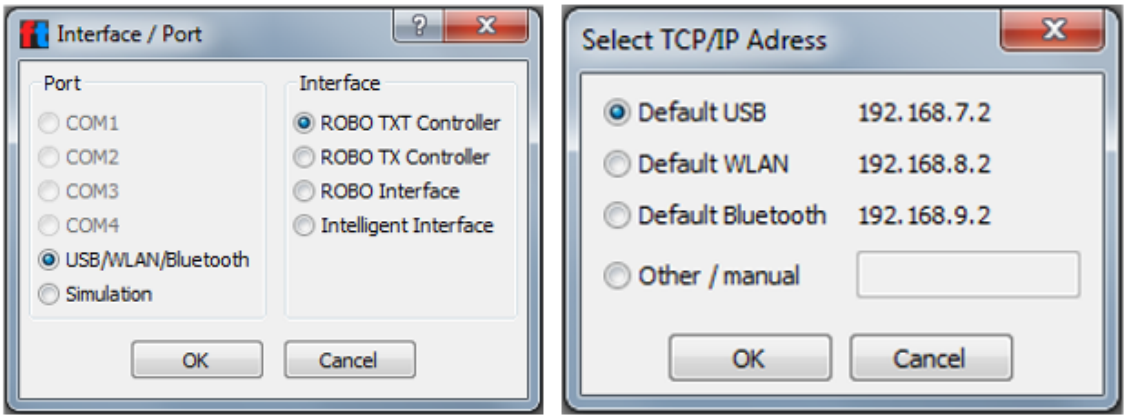


Kuvio 4. Fischertechnik ROBOPro Terminal aloitussivu ja ROBOPro Ensiaskeleet -kurssi

2.2.1 Ensiaskeleet

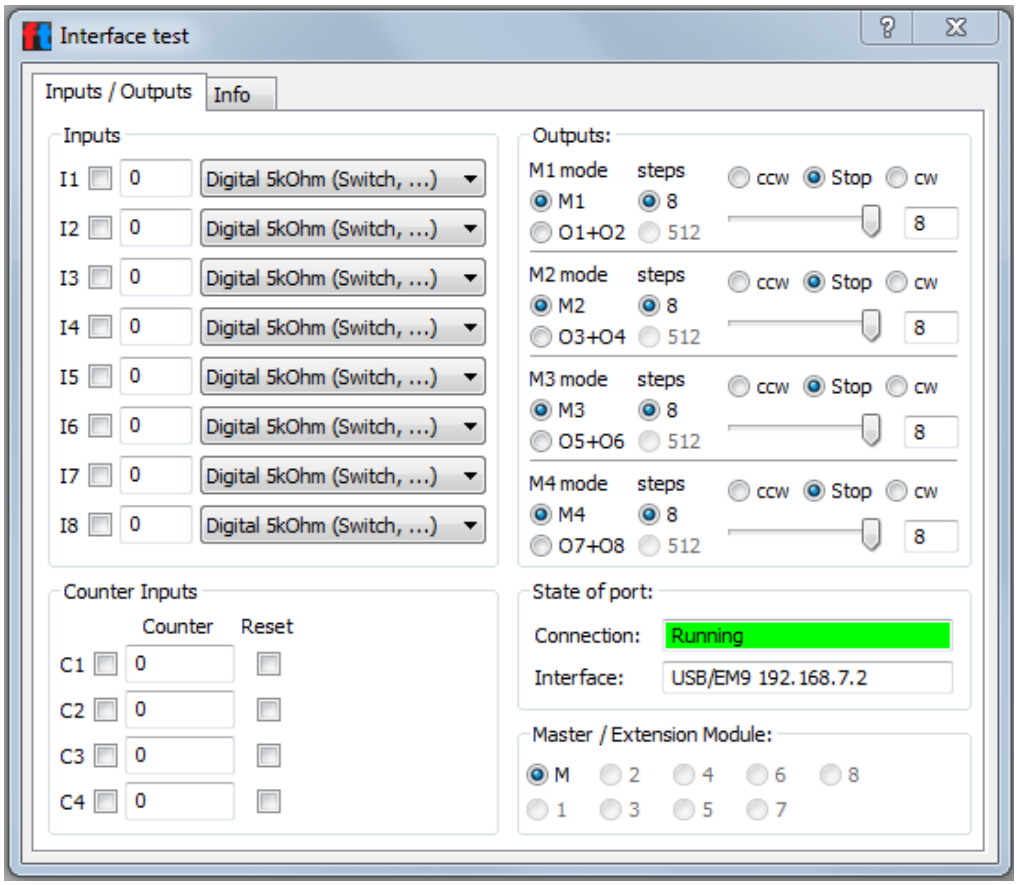
Terminaalin käynnistämisen jälkeen käynnistin ROBOPro-ohjelman sekä terminaalista Ensiaskeleet -kurssin. Seurasin kurssin ohjeita ensimmäisen yksinkertaisen rakennelman, joka on kytkimellä toimiva tuuletin, luomiseen ja käyttöön.

Ensimmäisenä minun täytyi määrittää USB-yhteys ohjelmointiympäristön ja robotin keskusyksikön välille. Tämä tapahtui painamalla työkalupalkin COM/USB-painiketta, jonka takaa avautui valikko, josta valittuna olivat kohdat ”USB/WLAN/Bluetooth” ja ”ROBO TXT Controller”. Painettuani OK, avautui näytölle TCP/IP –osoitteen valitsin (Kuvio 5.). Tässä valitsin kohdan ”Default USB” ja painoin OK. Tämän tehtyäni yhteys robottiin oli luotu.



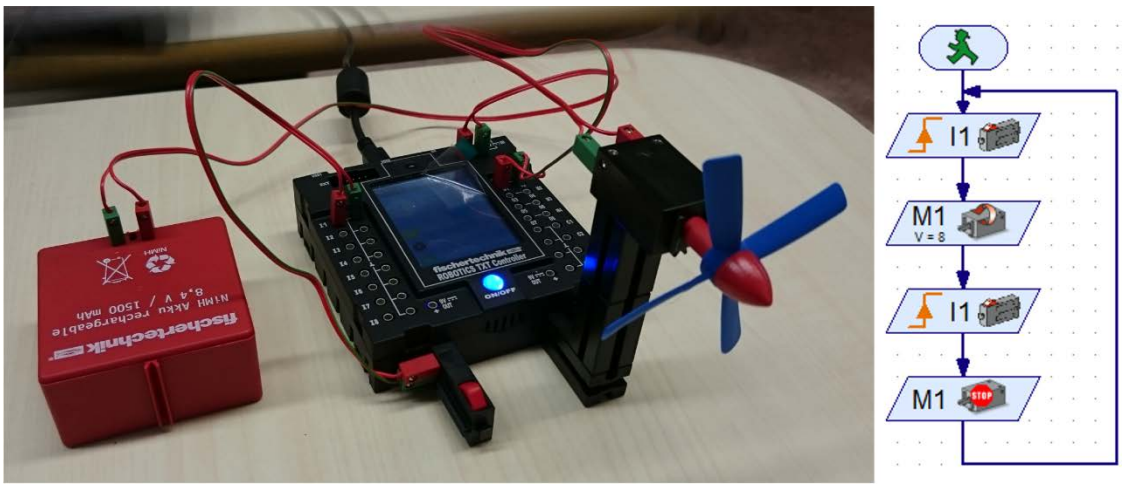
Kuvio 5. Portin valinta, ja TCP/IP-osoitteen valitsin

Toisena testasin luomaani yhteyttä ohjelmointiympäristön ja robotin välillä. Valitsin työkalupalkin kohdan ”Test”, jota painamalla avautui rajapinnan testeri (Kuvio 6.).



Kuva 6. Rajapinnan testeri

Rakentamani testilaitteen (Kuvio 7.) testaukseen valitsin Inputs-osiosta ensimmäisen inputin I1, joka vastaa fyysistä, robotin liittimiin kytkettyä painiketta. Moottorin käynnistin Outputs-osion, M1-kohdasta valitsemalla painikkeen ”ccw” (counter-clockwise, suom. vastapäivään). Moottorin nopeutta pystyy säätämään arvojen 1 – 8 välillä, joista 1 on hitain ja 8 nopein. Ensiaskeleet läpikäytyäni jatkoin ROBOPro-ympäristön testailua pienin rakennelmin ja ohjelmin.

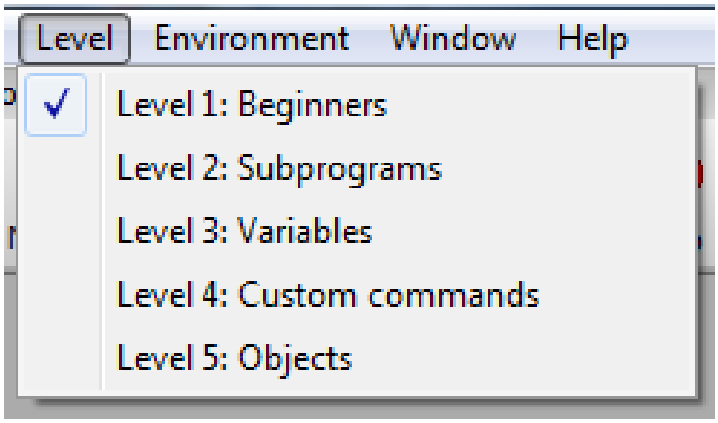


Kuvio 7. ROBOPron Ensiaskeleet -testirobotti ja -ohjelma

2.2.2 Ohjelmointitasot

ROBO Pro sisältää viisi eri ohjelmointitasoa (Kuvio 8.), jotka ovat

- taso 1: Aloittelijat
- taso 2: Aliohjelmat
- taso 3: Muuttujat
- taso 4: Mukautetut komennot
- taso 5: Oliot.



Kuvio 8. Tason valitsin

Jokainen taso tuo mukanaan uusia ja monipuolisempia ohjelmointielementtejä ja -mahdollisuuksia käytettäväksi. Ensimmäinen taso on kaikista yksinkertaisin ja

sisältää siten kaikista yksinkertaisimmat ja helpoiten käytettävät ohjelmointielementit, kuten kytkimen, moottorin ja ajastimen. Ensiaskeleiden ensimmäisen ohjelman ohjelmointi tapahtui tätä tasoa käyttäen.

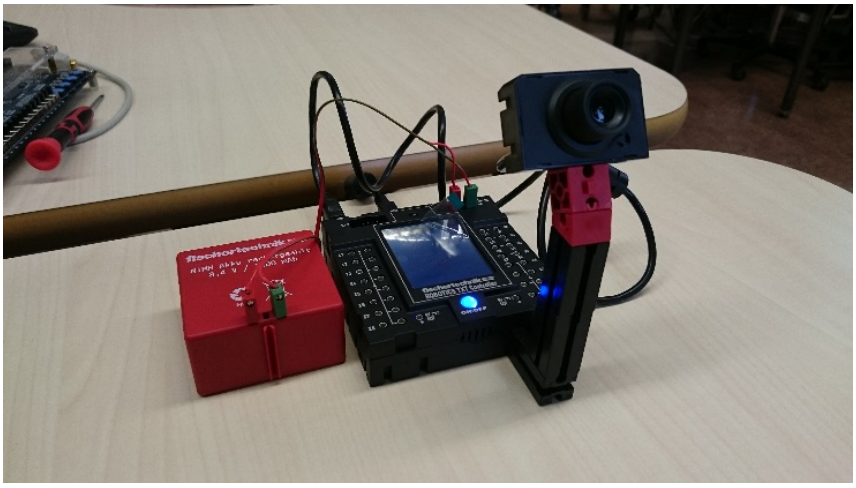
Toinen taso tuo mukanaan aliohjelmissa käytetyt aloitukset ja lopetukset sekä robottien väliseen kommunikointiin tarkoitetut lähetin- ja vastaanotinelementit että niissä hyödynnettäviä valmiita aliohjelmia esimerkiksi sijainnin tai asennon ilmoittamiseen. Kolmannella tasolla käytettävien elementtien määrä kasvaa suuresti, sen avatessa mahdollisuuden käyttää muuttujia, ajastimia, komentoja, vertailuja, haaroituksia, sisään- ja ulostuloelementtejä sekä operaattoreita.

Neljäs ja viides taso täydentävät kolmannella tasolla avattuja elementtejä lisäämällä mahdollisuuksia linkittää komentoja ja elementtejä toisiinsa. Viidennellä tasolla lisätään myös mahdollisuuksia värjätä linjoja ja elementtejä eri väreillä.

3 USB-KAMERA

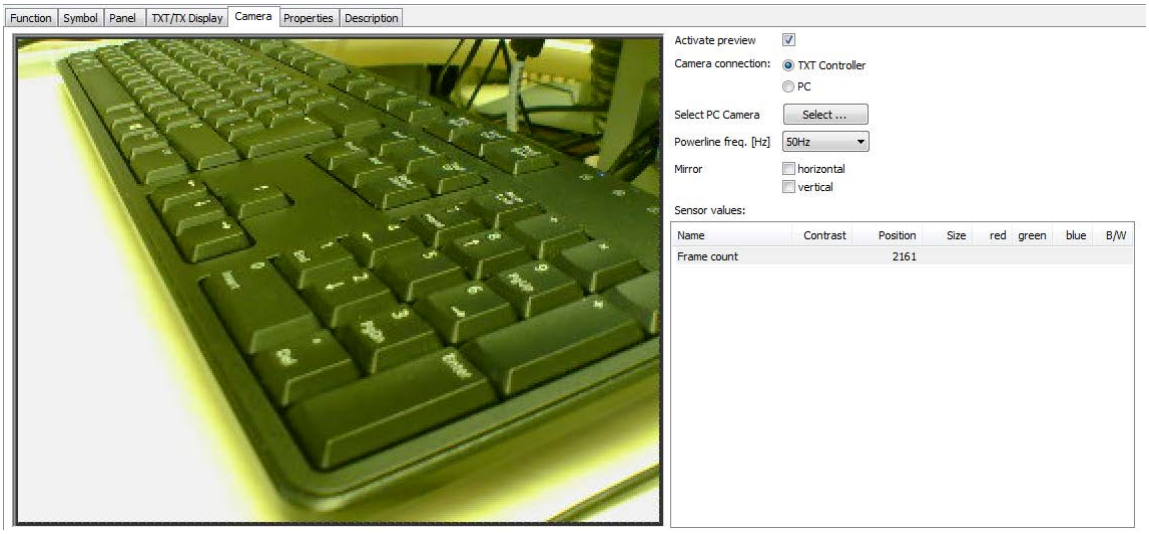
3.1 USB-kameraan tutustuminen

Fischertechnikin robottiin on myös mahdollista asentaa rakennussarjan mukana tullut USB-kamera (Kuvio 9.). Kameran kuvatarkkuus on yksi megapikseli, ja kamera voidaan liittää robotin takana olevaan USB 2.0 -porttiin tai vaihtoehtoisesti suoraan tietokoneeseen.



Kuvio 9. USB-kamera robotissa

Kameranäkymän saa esille avaamalla ROBOPron päänäkymästä Camera-välilehden ja valitsemalla ”Activate preview” (Kuvio 10.). Jos kamera on kiinni suoraan tietokoneessa, vaihdetaan kameran liitäntä TXT Controllerista PC:ksi.

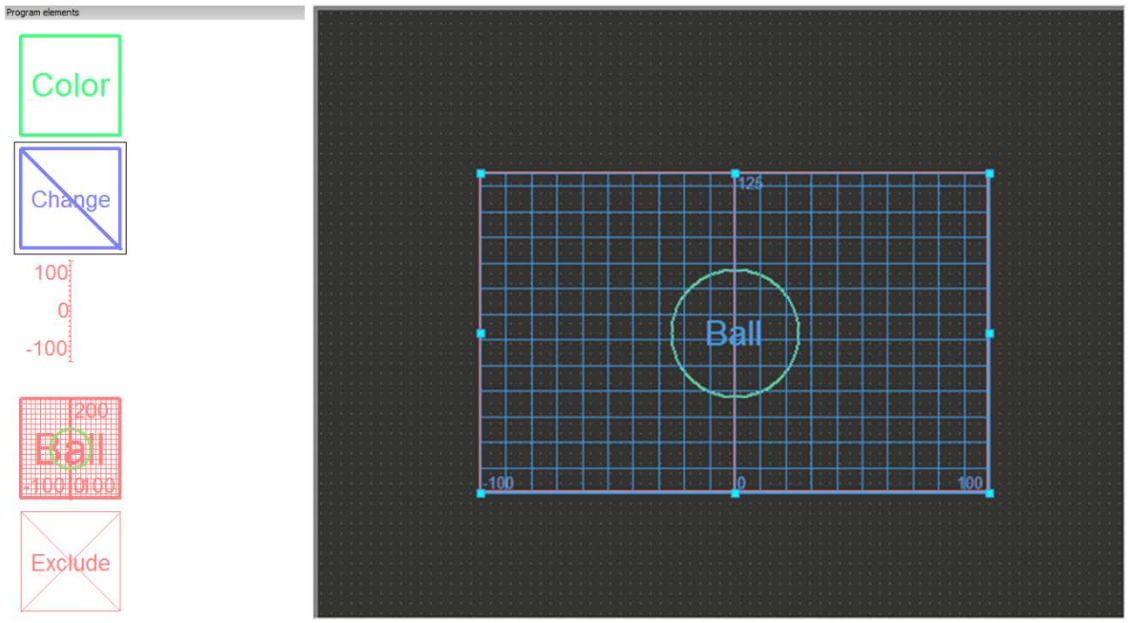


Kuvio 10. ROBOPron Camera-välilehti kameranäkymällä

3.2 USB-kameran sensorit

Fischertechnikin USB-kameran saa ohjelmoitua tunnistamaan erilaisia asioita, kuten värin, liikkeen tai muodon. Haluamansa sensorin saa valittua ROBOPron Camera-välilehdeltä, kohdasta ”Element groups” ja ”Sensor fields”, jolloin alapuolelle aukeaa valikko kameran ohjelmointielementeistä.

Sensori valitaan klikkaamalla ja piirtämällä se kameranäkymään haluamansa kokoisena (Kuvio 11.). Sensorialueen voi asettaa mihin kohtaa kameranäkymää tahansa ja alueen kokoa voi tarvittaessa muuttaa. Kameralle on mahdollista asettaa useampia eri sensoreita, myös monta samaa sensoria, joista jokaisen voi nimetä erikseen tunnistamisen helpottamiseksi.

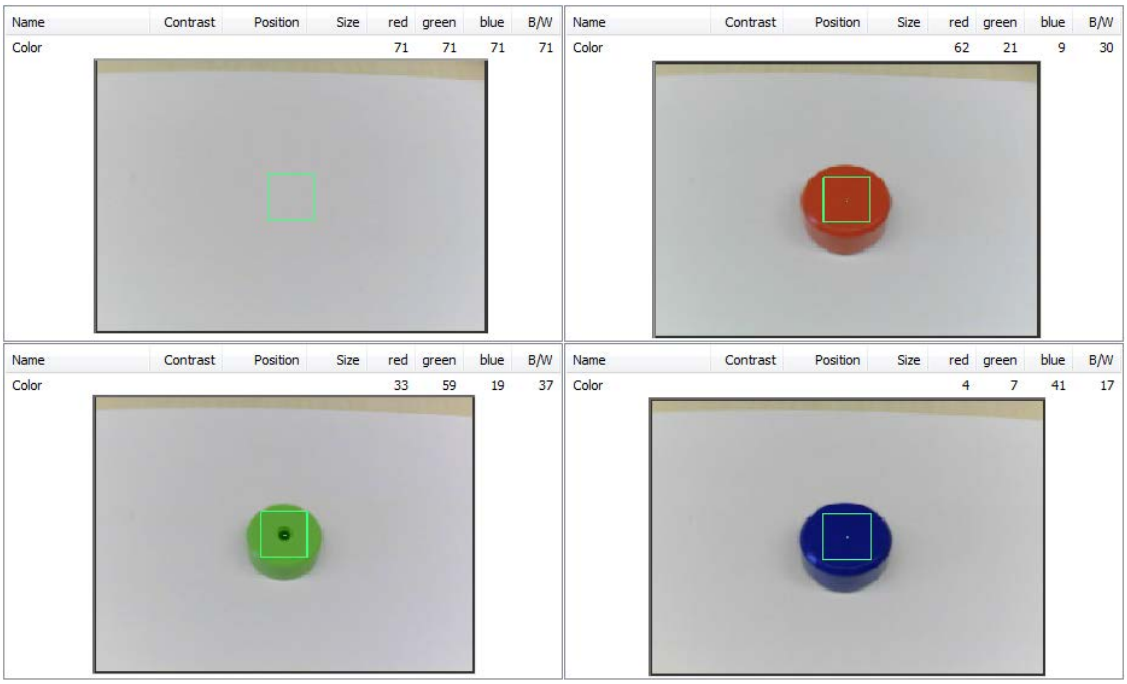


Kuvio 11. Kameran sensorivalikko ja sensorialueen asettaminen kameranäkymään

4 VÄRINTUNNISTUS

Kuten aiemmin jo on tullut mainittua, ROBOProssa on mahdollista tuoda eri sensoreita USB-kameran kuvaan ja yksi sensoreista on värintunnistin. Värintunnistusta voidaan käyttää eri värien tunnistamiseen RGB-värimallin mukaisesti. Tarkoitus on tutkia värintunnistimen kykyä sekä havaita että tunnistaa mallin mukaiset päävärit, jotka ovat punainen (R), vihreä (G) ja sininen (B).

Sensori ilmoittaa tunnistamansa värit prosentteina ja se väri jolla on isoin arvo, on kuvassa hallitseva väri (Kuvio 12.). Sensori ilmoittaa myös kuvan kirkkauden, jolloin pieni arvo tarkoittaa tummempaa ja suurempi arvo kirkkaampaa kuvaa. Kirkkauden arvo on aina RGB-värien arvojen keskiarvo. Jos jokaisella värillä on lähes samat arvot, on kyseessä joko musta, valkoinen tai harmaansävyinen kuva.



Kuvio 12. Väriarvot sekä valkoisesta taustasta että värillisistä kappaleista

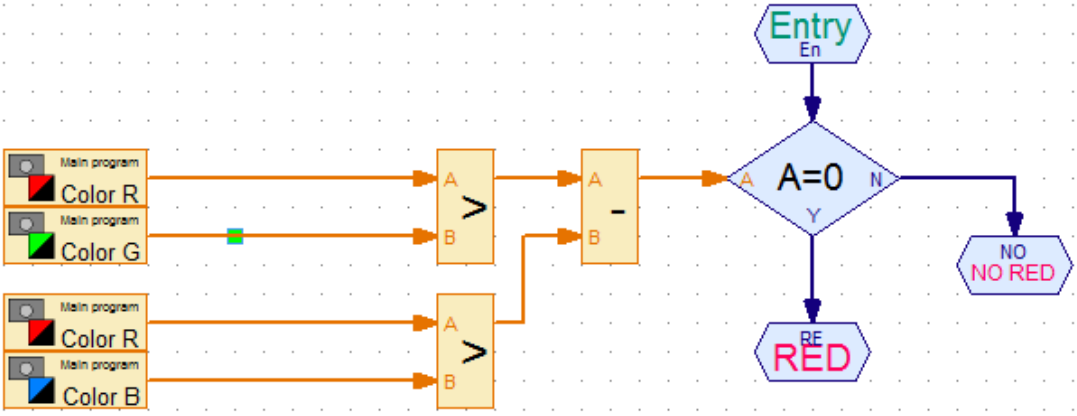
4.1 Päävärien tunnistaminen

4.1.1 Päävärien tunnistamisen kokeilu lainatulla ratkaisulla

Yritin ensi alkuun etsiä internetistä jonkinlaista ohjetta, millaisen ROBOPro-ohjelman minun kannattaisi tehdä, jotta voisin helposti alkaa tutkia kyseistä aihetta.

Ensimmäinen lupaavan näköinen osuma, minkä löysin, oli YouTube-video, jossa näytettiin jonkinlainen ohjelma värintunnistamiseen. Kokeilin tehdä tämän ohjelman itsekin, mutta en saanut sitä toimimaan.

Esimerkkiohjelman värintunnistus perustuisi siihen, että ensimmäisen vaiheen aliohjelmassa (Kuvio 13.) verrattiin sekä värin X-arvoa värin Y-arvoon että värin X-arvoa värin Z-arvoon. Näistä aliohjelma valitsi vertailuparien suuremmat arvot ja siirtyi seuraavaan vaiheeseen, jossa arvot vähennettiin toisistaan ja verrattiin, onko erotuksen arvo yhtä suuri kuin nolla. Tämän kohdan tarkoituksena oli selvittää, onko molemmista aikaisemmista vertailukohdista saatu arvo sama, jolloin niiden erotus olisi tasan nolla. Arvojen erotuksen ollessa yhtä suuri kuin nolla, aliohjelma siirtyisi kohtaan ”Väri X” ja ohjelma tekisi jotain mitä sen olisi tarkoitus tehdä tässä tapauksessa. Erotuksen ollessa jotain muuta kuin nolla, aliohjelma siirtyisi kohtaan ”Ei väriä” ja pääohjelma jatkaisi seuraavaan samanlaiseen aliohjelmaan, jossa vertailukohteena olisi seuraava väri. (Van Niekerk 2015.)



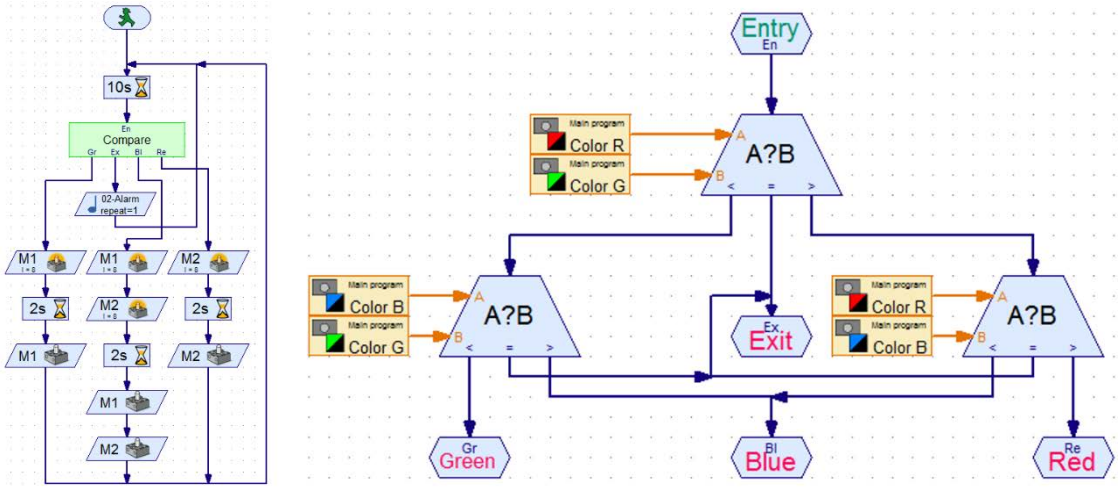
Kuvio 13. YouTubesta löytyneen ohjelman aliohjelma, jossa verrattavana punainen väri

4.1.2 Päävärien tunnistamisen tutkiminen omalla ratkaisulla

Kokeiltuani tätä ohjelmaa ja todettuani sen toimimattomuuden, päätin jättää kokonaan pois internetin ohjeet ja tehdä itse kokonaan oman ohjelmani. Ohjelmani ensimmäisen version aliohjelmassa (Kuvio 14.) vertasin värejä suoraan toisiinsa, pari kerrallaan. Ensimmäisessä kohdassa vertasin punaisen värin arvoa A, vihreän värin arvoon B, jonka jälkeen aliohjelman logiikka vertaa, kumpi arvoista on suurempi vai ovatko ne yhtä suuria. Punaisen arvon A ollessa suurempi kuin

vihreän arvo B, siirtyy ohjelma tämän määrittämää polkua seuraavaan samanlaiseen vertailuun, jossa verrattavina arvoina ovat punainen A ja sininen B. Ensimmäisessä kohdassa vihreän värin arvon B ollessa suurempi kuin punaisen värin arvo A, siirtyy ohjelma toiseen vertailuun vertaamaan värien sininen A ja vihreä B, arvoja. Sekä sinisen että vihreän tai vaihtoehtoisesti sekä punaisen että vihreän vertailun jälkeen, aliohjelma siirtyy vertailujen suurinta väriä vastaavaan loppuun. Väriparien vertailtavien arvojen ollessa missä tahansa kohdassa yhtä suuria, saa aliohjelma tehtävänsä päätökseen ja palaa pääohjelman alkuun.

Aliohjelman annettua jonkin arvon, pääohjelma (Kuvio 14.) suoritti sille arvolle määritetyn komentojonon. Tuloksen ollessa tunnistamaton väri, soitti ohjelma hälytysäänen ja siirtyi takaisin ohjelman alkuun. Vaihtoehtoisesti tuloksen ollessa joko punainen, vihreä tai sininen väri, sytytettiin näitä värejä vastaavat lamput kahden sekunnin ajaksi, jonka jälkeen ohjelma siirtyi takaisin alkuunsa.

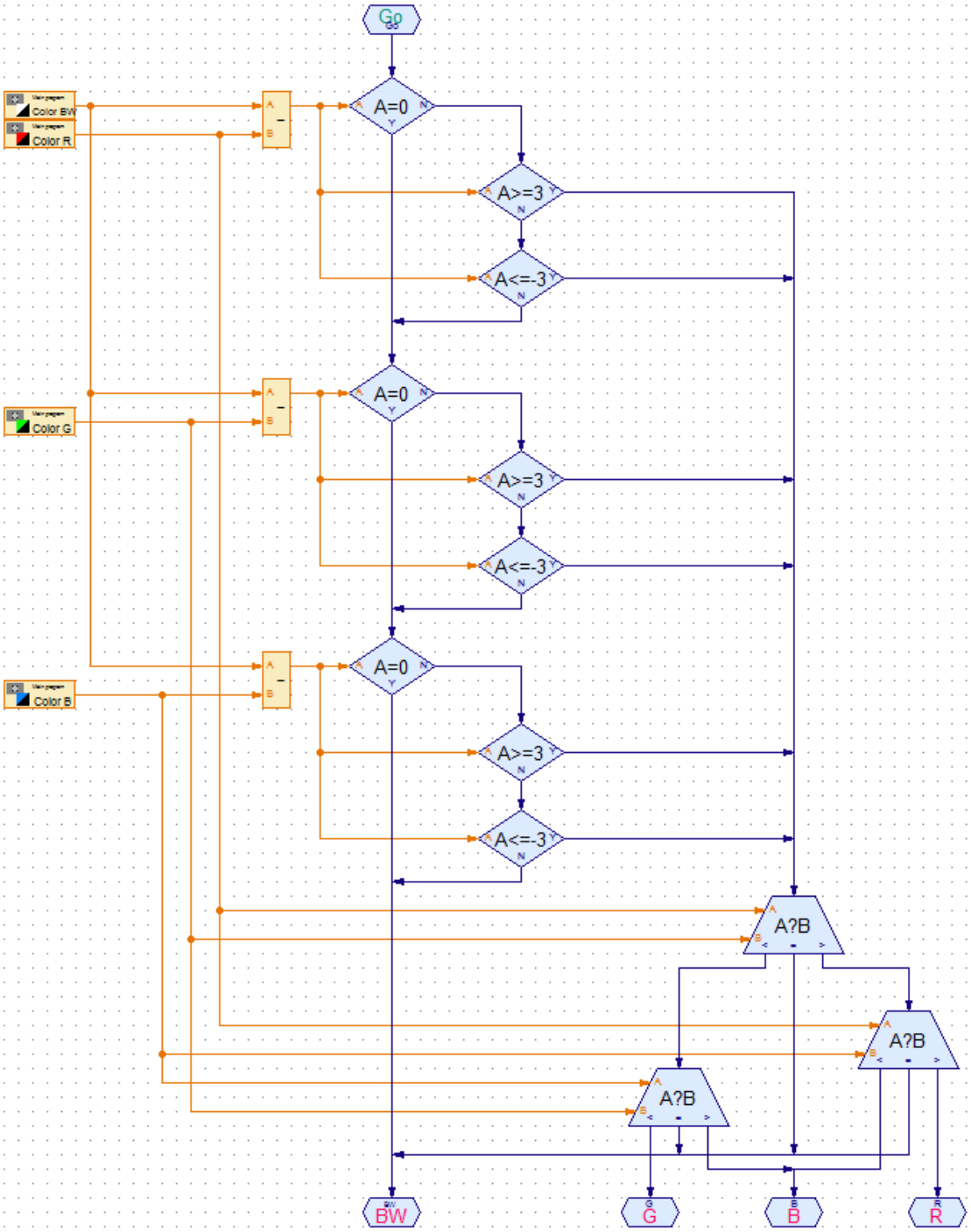


Kuvio 14. Ohjelman ensimmäisen version pää- ja aliohjelma

4.2 Päävärien erottaminen mustavalkoisuudesta

Ohjelmani ensimmäinen versio ei kuitenkaan ollut vielä tarpeeksi tarkka erottamaan värejä toisistansa, sillä se vertaili värejä toisiinsa hyvin karkeasti. Yhden värin arvon ollessa vaikkapa vain yhden prosentin muiden värien arvoja suurempi, olisi ohjelma tulkinnut tämän värin hallitsevaksi väriksi, vaikka kuva olisi todennäköisimmin ollut todellisuudessa väriltään mustavalkoinen.

Ohjelman karkeuden ratkaisemiseksi tein lisäyksen aliohjelmaan (Kuvio 15.), jossa ennen värien arvojen vertailua toisiinsa, vertaillaan jokaisen värin arvoa kuvan kirkkauden arvoon. Tällä vertailulla saadaan selvitettyä, onko kyseessä värillinen vai mustavalkoinen kuva.



Kuvio 15. Päävärien erottaminen mustavalkoisesta

Ensimmäisenä vertailtavana arvona on punaisen värin arvo, joka vähennetään kirkkauden arvosta, jolloin ohjelmalta kysytään, onko erotus yhtä suuri kuin nolla.

Jos arvojen erotus ei ole nolla, kysytään onko se joko suurempi tai yhtä suuri kuin 3, tai yhtä suuri tai pienempi kuin -3. Jos kumpikaan näistä väitteistä on tosi, siirtyy ohjelma värien vertailuun, mutta jos erotus on välillä [-2, +2], toistaa ohjelma saman vertailun käyttäen vähentäjänä muita värejä.

Kirkkauden arvon ollessa jokaisen värin arvon kanssa joko yhtä suuria tai lähes yhtä suuria kahden prosentin toleranssilla, siirtyy aliohjelma mustavalkoisen vaihtoehdon lopetukseen. Muussa tapauksessa aliohjelma käy läpi värien arvojen vertailun ja lopettaa voimakkaimpaan väriin. Tilanteessa, jossa aliohjelma ei vielä tässäkään vaiheessa tunnista mitään väriä hallitsevaksi, siirrytään ohjelmassa mustavalkoiseen lopetukseen.

4.3 Välivärien tunnistaminen

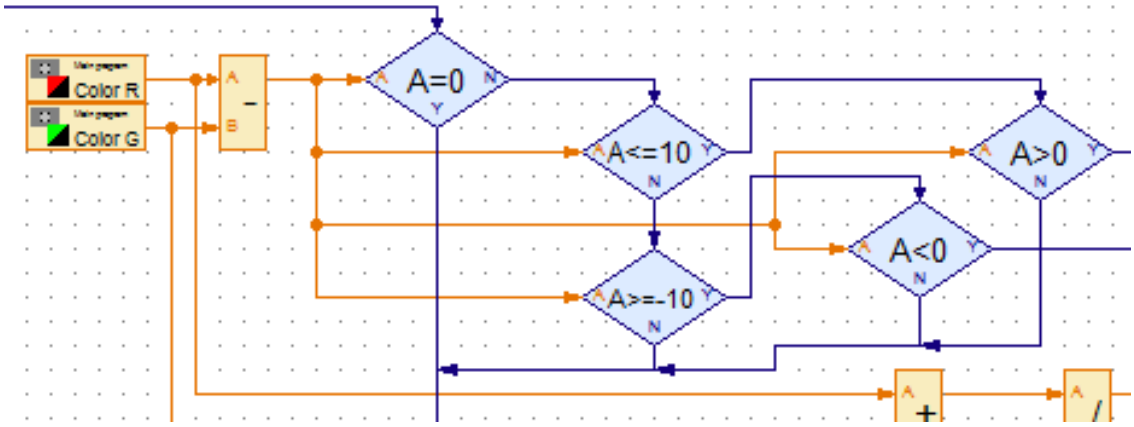
Halusin myös tuoda testeihin mukaan RGB-värimallin mukaiset välivärit, jotka ovat keltainen (Y), magenta (M) ja syaani (C). Nämä välivärit ovat arvoiltaan päävärien tasaisia sekoituksia, eli jokainen väliväri saadaan antamalla kahdelle päävärille yhtä suuret arvot. Keltainen on sekoitus punaista ja vihreää, magenta on sekoitus punaista ja sinistä, ja syaani on sekoitus vihreää ja sinistä. (Huttunen 2005.) Näiden tunnistamista varten annoin kuitenkin värien arvojen vertailuun kymmenen prosentin toleranssin, jolla saadaan vähennettyä kameran ja valaistuksen aiheuttamien värivirheiden vaikutusta. Välivärin tunnistamista varten lisäsin kirkkauden ja värien arvojen vertailuiden perään lisää kohtia, joihin siirryttäisiin siinä tapauksessa, jos edeltävässä vaiheessa tunnistettiin jonkin värin arvo suuremmaksi kuin kirkkauden arvo.

Vertailun ensimmäisen version ensimmäisessä kohdassa vähennetään kahden värin arvot toisistaan ja verrataan, onko niiden erotus yhtä suuri kuin nolla. Jos väite pitää paikkansa, siirtyy ohjelma seuraavaan samanlaiseen vertailuun, jossa vertailtavana ovat seuraavan väriparin arvot. Erotuksen ollessa erisuuri kuin nolla, jatketaan samojen värien vertailua tarkistamalla, onko niiden erotus pienempi tai yhtä suuri kuin 10. Jos väite ei pidä paikkaansa, siirtyy ohjelma seuraavaan kohtaan, jossa tarkistetaan, onko erotus yhtä suuri tai suurempi kuin -10, mutta jos väite pitää paikkansa, siirrytään tarkistamaan, onko kahden vertailtavan värin arvojen keskiarvo suurempi kuin kolmannen värin arvo. Väitteen pitäessä

paikkansa, voidaan päätellä kyseessä olevan väliväri. Keskiarvon ollessa jostain syystä yhtä suuri kolmannen värin arvon kanssa, siirtyy ohjelma mustavalkoisen lopetukseen. Keskiarvon ollessa pienempi kuin kolmannen värin arvo, siirtyy ohjelma päävärien vertailuun. Jos erotus kuitenkin on pienempi kuin -10, jatkaa ohjelma kulkuaan seuraavan väriparin vertailuun.

Ensimmäisessä versiossa ensimmäinen vastaan tullut ongelma oli kohdassa, kun tarkistettiin, onko erotus pienempi tai yhtä suuri kuin 10. Väitteen ollessa epätosi, esimerkiksi tilanteessa, jossa erotuksen arvo oli 15, siirryttiin tarkistamaan, onko erotus yhtä suuri tai suurempi kuin -10, jolloin tämä väite oli tosi. Näin tapahtuessa ohjelma pääsi jatkamaan suuntaan, jonne sen ei olisi oikeasti pitänyt jatkaa, koska se ei täyttänyt ehtoa ”pienempi tai yhtä suuri kuin 10”. Sama ongelma oli myös tilanteessa, jossa erotus olisi ollut esimerkiksi -15, jolloin se olisi pienempi kuin 10.

Ratkaisin ongelman lisäämällä kohtien 10 ja -10 suhteen vertailujen jälkeen vertailut, joissa tutkittiin, onko erotus suurempi tai pienempi kuin nolla (Kuvio 16.). Vasta näiden ehtojen täyttymisen jälkeen ohjelma voisi jatkaa keskiarvon ja kolmannen värin vertailuun.

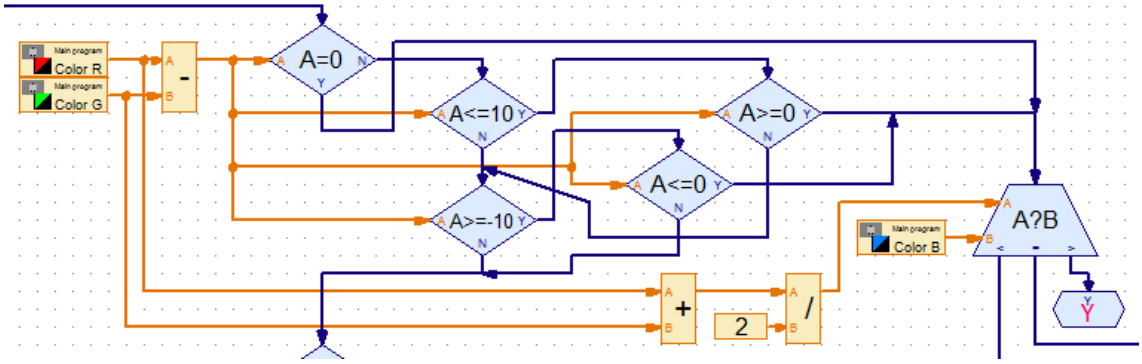


Kuvio 16. Välivärin tunnistus vertailuilla nollan suhteen väitteellä ”arvo on suurempi/pienempi kuin nolla”

En kuitenkaan ollut ottanut huomioon sitä mahdollisuutta, että erotuksen arvo voisi olla tasan nolla, jolloin se ei olisi täyttänyt kumpaakaan väitettä ja ohjelma olisi virheellisesti jatkanut seuraavan väriparin vertailuun. Korjasin tämän virheen muuttamalla väitteet ”suurempi kuin nolla” ja ”pienempi kuin nolla”, väitteiksi ”yhtä

suuri tai suurempi kuin nolla” ja ”pienempi tai yhtä suuri kuin nolla”, jolloin erotuksen arvoksi sallittaisiin nolla. Tämän vertailun pitäisi kuitenkin olla ohjelmassa irrelevantti, koska erotuksen arvo tässä vaiheessa ei enää pitäisi voida olla nolla, sillä sen tulisi olla jo tarkastettu ja todettu epätodeksi. Virheiden välttämiseksi muutin kuitenkin tämän kohdan sen nykyiseen muotoonsa.

Myöskin haaroituksia täytyi korjata ja muuttaa (Kuvio 17.). Erotuksen ja nollan yhtäsuuruuden vertailun jälkeinen haara jolle ohjelma jatkoi, jos väite piti paikkansa, piti yhdistää suoraan keskiarvon ja kolmannen värin vertailuun. Siinä tapauksessa erotuksen arvo täytti automaattisesti seuraavien välivaiheiden ehdot. Vaihdoin myös ehdolta ”yhtä suuri tai suurempi kuin nolla” haarautuvan Ei-haaran kulkemaan seuraavaksi vertailuun -10 suhteen. Aiemmin se jatkoi seuraavan väriparin vertailuun, mutta tarkoitus on tämänkin vaiheen jälkeen selvittää, onko arvo pienempi kuin nolla, mutta yhtä suuri tai suurempi kuin -10.



Kuvio 17. Välivärin tunnistus kokonaisuudessaan kaikki kohdat korjattuina

4.4 Värintunnistuksessa ilmenneitä ongelmia

4.4.1 Valaistuksen vaikutus väreihin

Värien tunnistamisessa ilmeni joitain ongelmia, jotka johtuvat lähinnä sekä laitteiston että ympäristön aiheuttamista epätarkkuuksista ja muuttuvista olosuhteista, aiheuttaen vääristymiä väreissä. Värien arvot riippuvat todella suuresti käytetystä valaistuksesta, sillä lähteestä riippuen siinä voi olla suuriakin eroja. Jo pelkästään loisteputkissa on eri arvoja valon värilämpötilalle, joka ilmaistaan kelvineinä (K).

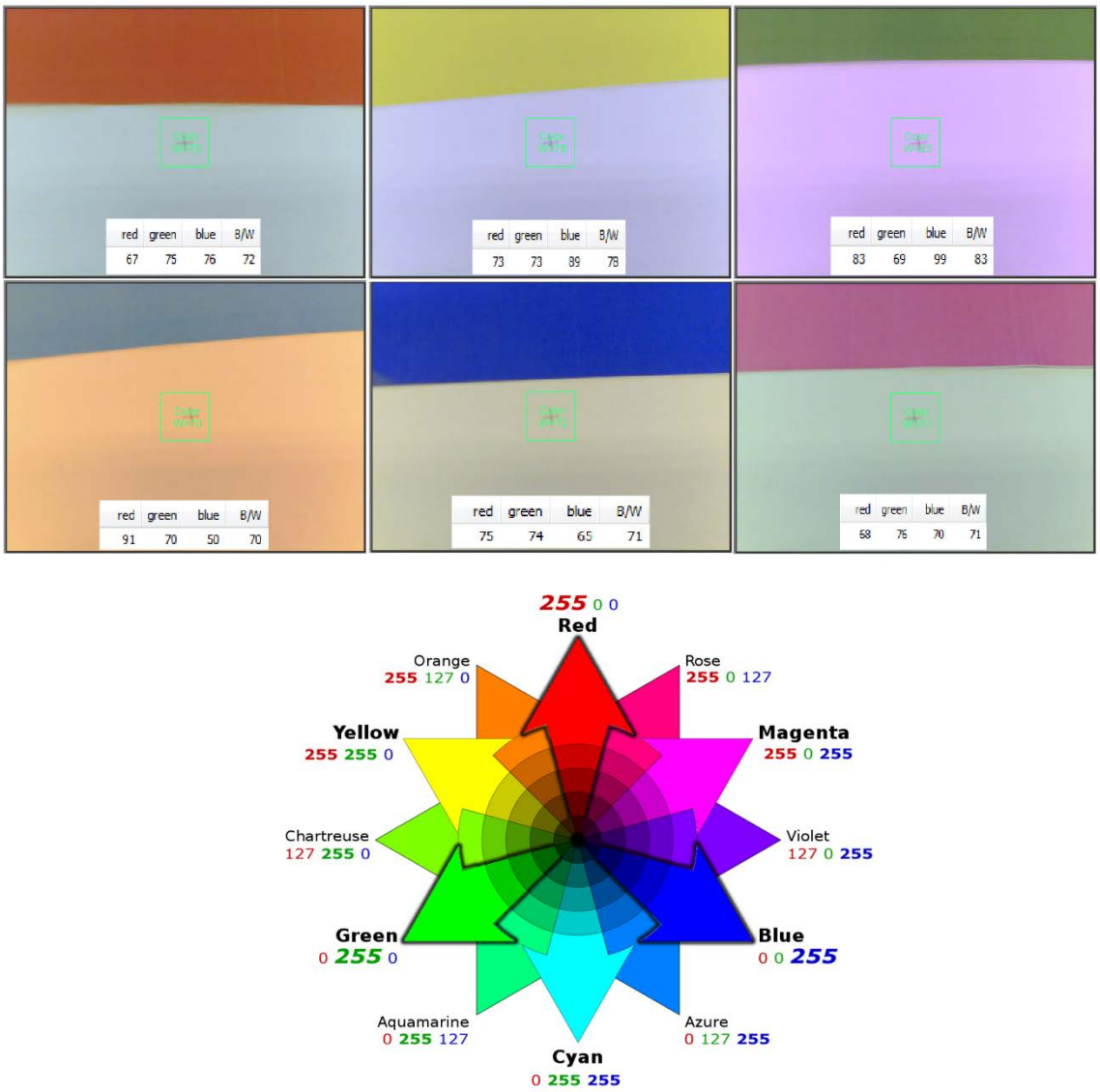
Värilämpötilat voidaan karkeasti luokitella kolmeen luokkaan: lämpimänvalkoiset, viileänvalkoiset ja luonnonvalo. Lämpimänvalkoiset värilämpötilat ovat väliltä 2000 – 3000 K ja ilmenevät oranssin/keltaisen sävyisenä valkoisena. Viileänvalkoiset ovat väliltä 3100 – 4500 K ja ilmenevät enemmän neutraalin/sinisen sävyisenä valkoisena. Luonnonvalo sijoittuu välille 4600 – 6500 K ja on enemmän sinisävyinen, kuin viileänvalkoiset värilämpötilat. Toisin sanoen mitä suurempi värilämpötilan arvo on, sitä kylmempi ja sinertävämpi sen sävy on. (Westinghouse 2017.) Loisteputkien värilämpötilat vaihtelevat välillä 2700 – 6500 K, kun auringonvalo on noin 5500 Kelviniä (Motiva Oy 2017).

Tämä aiheuttaa herkästi ongelmia varsinkin välivärien tunnistamisessa, jolloin niiden toinen hallitseva väri voi voimistua liikaa. Esimerkiksi jos tarkoituksena olisi tunnistaa syaani ja vallitsevan valaistuksen värilämpötila olisi esimerkiksi noin 6000 K tai enemmän, voisi sininen väri voimistua kuvassa liikaa, aiheuttaen sen, että syaani tunnistuisikin sinisenä. Liian lämpimän sävyinen valaistus puolestaan voisi aiheuttaa magentan tunnistamisen liian punaisena.

Valon voimakkuudella myöskin on suuri merkitys värien tunnistamisessa. Liian voimakas valo aiheuttaa kuvan puhki palamisen, toisin sanoen kuvasta tulee niin kirkas, että se on täysin valkoinen. Vastavuoroisesti liian himmeä/tumma valaistus aiheuttaa sen, että eri värit eivät välttämättä erotu kuvasta riittävästi.

4.4.2 Kuvan värivirheet

Laitteiston omalla USB-kameralla kuvatessa, kuvassa olevan valkoisen taustan ja testattava väri saattoivat vääristyä välillä hyvinkin suuresti. Tuodessani kuvan alueelle jonkin kuudesta testattavasta väristä, lisääntyi kuvan taustassa testattavan värin käänteisväriä vastaavat väriarvot (Kuvio 18.). Käänteisväri on RGB-värimallin mukaisen väriympyrän väri, joka on testattavan värin vastakkaisella puolella.

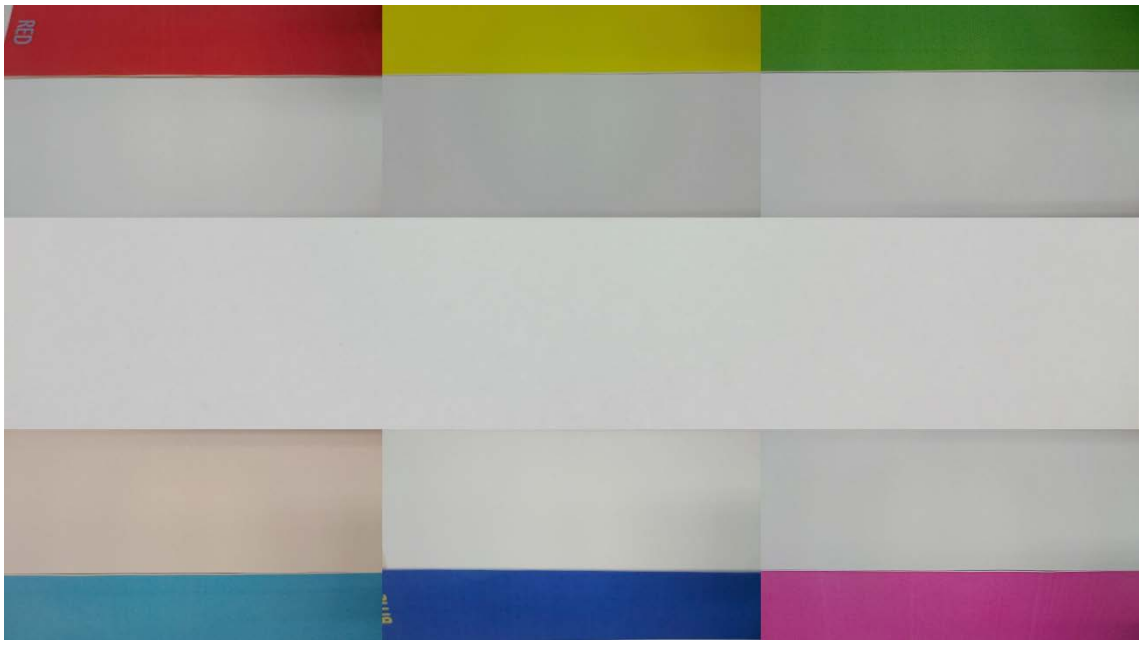


Kuvio 18. Jokaisen testattavan väriliuskan taustan arvot ja RGB-värikartta

Tämä ilmiö johtuu lähinnä siitä, että USB-kameralla ei ole erikseen mahdollista säätää valkotasapainon asetuksia tai ne puuttuvat ohjelmiston säädöistä kokonaan. Valkotasapainolla tarkoitetaan kuvan kohteen värissä aiheutuvaa muutosta, jonka valon väri aiheuttaa, ja sen tavoitteena on saada kuvan valkoiset ja harmaat sävyt näyttämään mahdollisimman neutraaleilta korostamalla valon käänteisväriä. Valkotasapainon säätöjen puuttuminen voi aiheuttaa sen, että sekä valonlähteistä tulevan valon väri ja eri pinnoista aiheutuvat heijastukset muuttavat kuvan väriä suuresti. (DigiFAQ 2008.)

Puhelimella otetuissa kuvissa (Kuvio 19.) ei ilmennyt samaa ongelmaa, ainakaan yhtä suuresti, kuin USB-kameralla otetuissa kuvissa. Ainoa selvä ero on syaanin

värisen testiliuskan kuvassa, jonka tausta on korostuneesti hieman punaisen sävyinen. Muuten puhelimella otetut kuvat ovat suhteellisen tasapainoisia, sillä puhelimen kamera yrittää automaattisesti säätää valkotasapainoa.



Kuvio 19. Puhelimen kameralla otetut vertailukuvat, keskellä kuva valkoisesta taustasta ilman muita värejä.

4.4.3 Väriarvojen muuttuminen kesken ohjelman

Erittäin pieni, mutta joissain tapauksissa hyvin relevantti ongelma on kuvan väriarvojen mahdollinen muuttuminen ohjelman suorituksen aikana. Tämän ongelma tuli itselle ilmi, kun testailin omien ohjelmieni eri versioita hidastettuna, jolloin saatoin tahallani vaihtaa väriliuskaa kesken ohjelman. Todellisuudessa tämän ongelman vaikutus nopeassa ja yksinkertaisessa värintunnistuksessa on jokseenkin epätodennäköistä, sillä esimerkiksi tekemäni testiohjelman aliohjelman suoritusaika on sekunnin murto-osa, joten suuria vaihteluita ei pääse siinä ajassa helposti syntymään.

Merkittävämmäksi ongelma muodostuu sitä mukaan, mitä pidempikestoisemmaksi ohjelma tehdään ja jos samoja väriarvoja pitää käyttää monessa eri kohdassa. Esimerkiksi omaan ohjelmaani peilaten, jos olisin lisännyt aliohjelman suorituksessa jokaisen pääkohdan väliin ajastimen tai jonkin muun komennon,

jonka loppuun suorittamista ohjelma joutuu odottamaan, olisi väriarvojen muuttuminen paljon todennäköisempää.

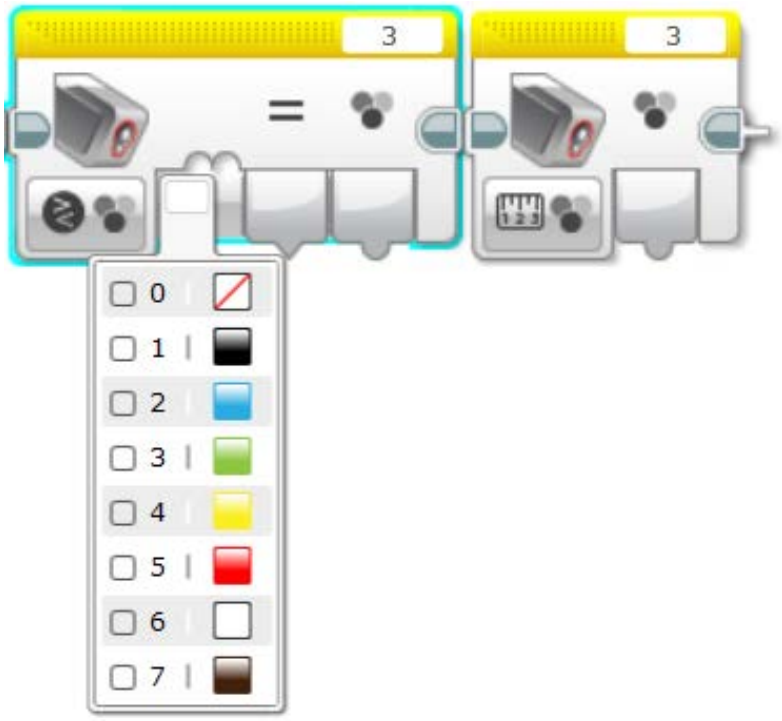
Värien muuttumisen aiheuttamasta ongelmasta eroon pääsemiseksi tulisi värien arvot tallettaa heti ohjelman alussa ohjelmointielementeistä löytyvään listaan, jossa ne pysyvät muuttumattomina ja josta niitä voidaan tarvittaessa käyttää ohjelman eri kohdissa. Itse en tätä toimenpidettä tehnyt omissa testeissäni, sillä en kokenut sen olevan tarpeellista tavoitteisiini nähden.

4.5 Värintunnistus Mindstorms EV3:lla

Värintunnistus Lego Mindstorms EV3 -robotilla tapahtuu käyttäen sen mukana tullutta värintunnistussensoria (Kuvio 20.). Sensorilla pystyy sekä tunnistamaan seitsemää eri väriä että mittaamaan valon intensiteettiä. (Lego 2017b.)

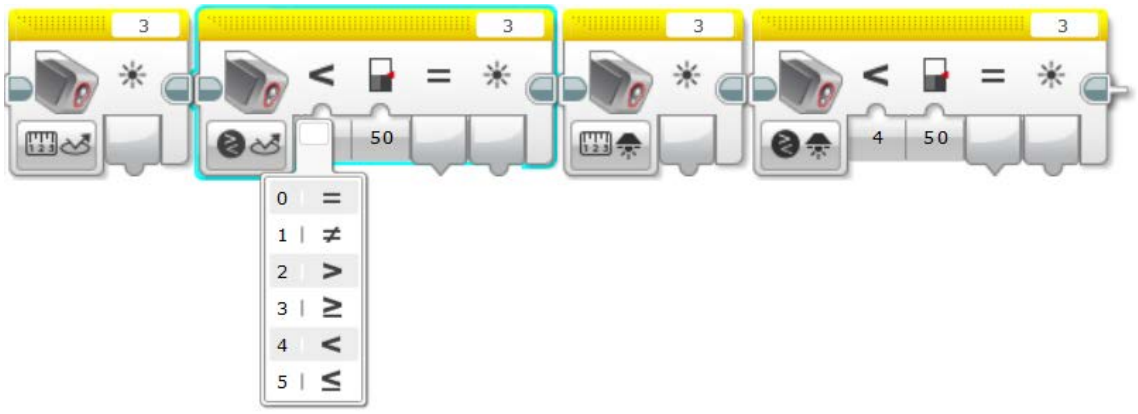
Kuvio 20. Lego Mindstorms EV3 värisensori (Lego 2017a)

Värejä voidaan tunnistaa kahdella eri ohjelmointielementtivariaatiolla (Kuvio 21.), joista ensimmäisellä voidaan valita suoraan yksi tai useampi väri mikä halutaan tunnistaa. Tunnistettavia värejä ovat musta, sininen, vihreä, keltainen, punainen, valkoinen ja ruskea. Elementin tunnistaessa edes yhden valituista väreistä, antaa elementti tunnistetun värin arvoksi ”true”. Toinen elementti mittaa heijastuvaa väriä ja antaa sille arvon väliltä 1 – 7, mutta jos sensori ei tunnista mitään väreistä, annetaan arvoksi 0. (Lego 2017b.)



Kuvio 21. Värintunnistuksen ohjelmointielementit

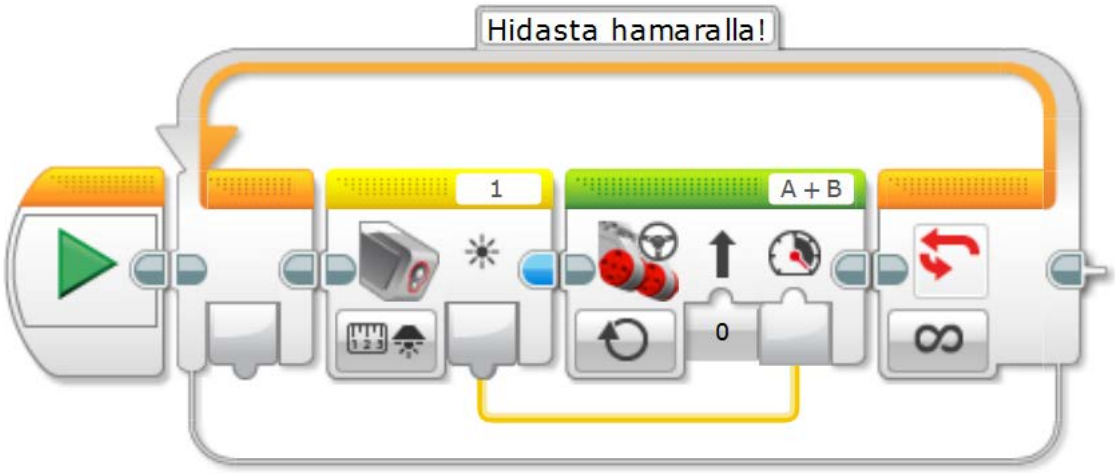
Valon intensiteetin eli valon määrän ohjelmointielementtien (Kuvio 22.) avulla nimensä mukaan mitataan valon määrää joko ympäristöstä tai heijastuksesta (Valostore 2017). Intensiteetin vertailulausekkeita voidaan käyttää esimerkiksi tapauksessa, jossa halutaan sytyttää robotin omat valot joko ympäröivän tai heijastuvan valon määrän ollessa liian vähäinen. Vastavuoroisesti voidaan robotin valot myös sammuttaa muun valon määrän ollessa tarpeeksi suuri.



Kuvio 22. Valon intensiteetin sensorit

Intensiteetin mittausta voidaan käyttää esimerkiksi ohjelmassa, jonka aikana robotti kulkee vaihtelevan kirkkaassa ympäristössä (Kuvio #.) tai mittaa eriväristen

pintojen intensiteettiä. Tällöin valon määrän arvoa voidaan käyttää määrittämään moottoreiden pyörimisnopeuden esimerkiksi hidastamalla niitä robotin siirtyessä tummemmalle alueelle tai päinvastoin.



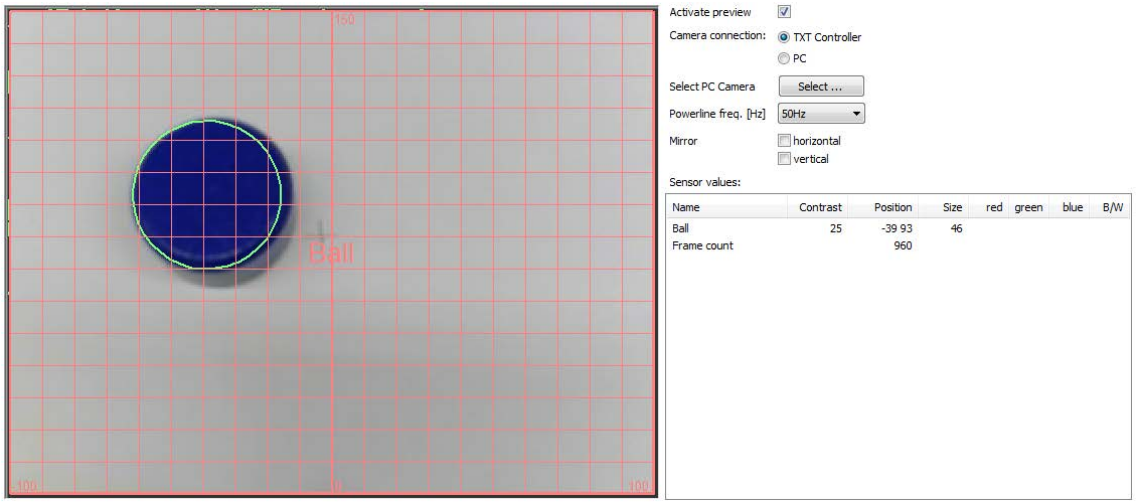
Kuvio 23. Ympäristön intensiteetin arvon käyttäminen moottoreiden nopeutena

5 KUVANTUNNISTUS

5.1 Ympyräntunnistus

Värintunnistuksen lisäksi kamera voidaan ohjelmoida tunnistamaan sekä erilaisia muotoja, kuten ympyröitä ja viivoja, että kuvassa tapahtuvaa liikettä. Ympyräntunnistusta (Kuvio 24.) voidaan käyttää pyöreiden muotojen sijainnin, koon ja kontrastin tunnistamiseen.

Ympyräntunnistusta voi käyttää esimerkiksi jalkapallon tai jääkiekon tunnistamiseen ja seuraamiseen. Pelissä tätä voisi hyödyntää epäselvissä tilanteissa maalin sisällä tunnistamaan käykö pallo tai kiekko maalin sisällä.



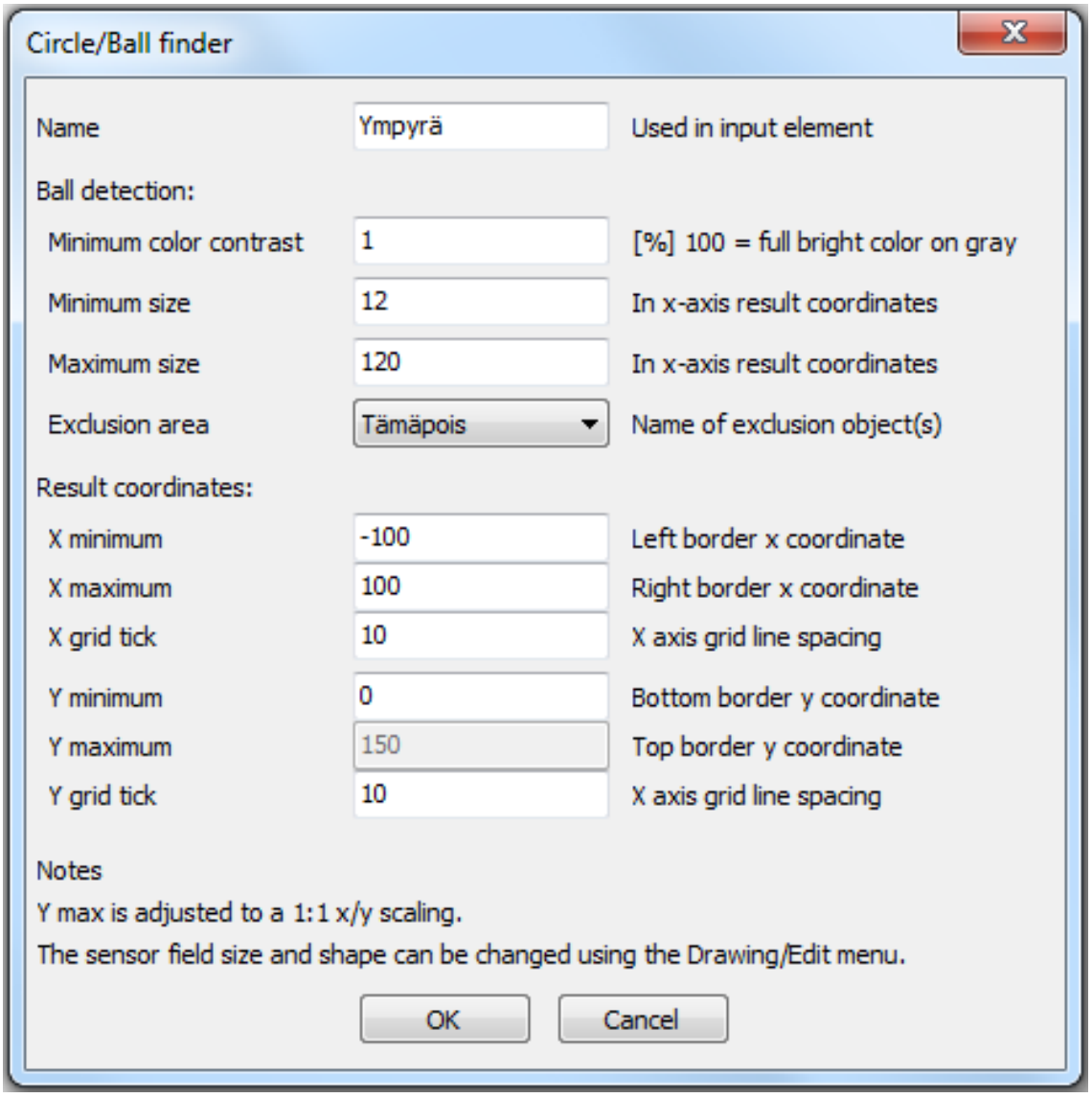
Kuvio 24. Ympyräntunnistimen näkymä ja tunnistettu ympyrä kamera-välilehdellä

5.1.1 Ympyräntunnistuksen asetukset

Ympyräntunnistuksen asetuksista (Kuvio 25.) saadaan säädettyä sekä ympyrään että tunnistusalueeseen liittyviä arvoja. Ympyrään liittyvät arvot ovat sen kontrasti sekä koon pienin ja suurin arvo.

Kontrastille annettava arvo määrittelee sen, kuinka värikäs tai väriltään taustan väristä poikkeava ympyrä on tarkoitus kuvasta tunnistaa. Tämä ilmaistaan prosentteina, jolloin sata prosenttia tarkoittaisi erittäin värikkään kohteen tunnistamista harmaansävyistä taustaa vasten, ja nollalla prosentilla ilmaistaan vain sitä, tunnistetaanko kuvasta kohdetta ollenkaan.

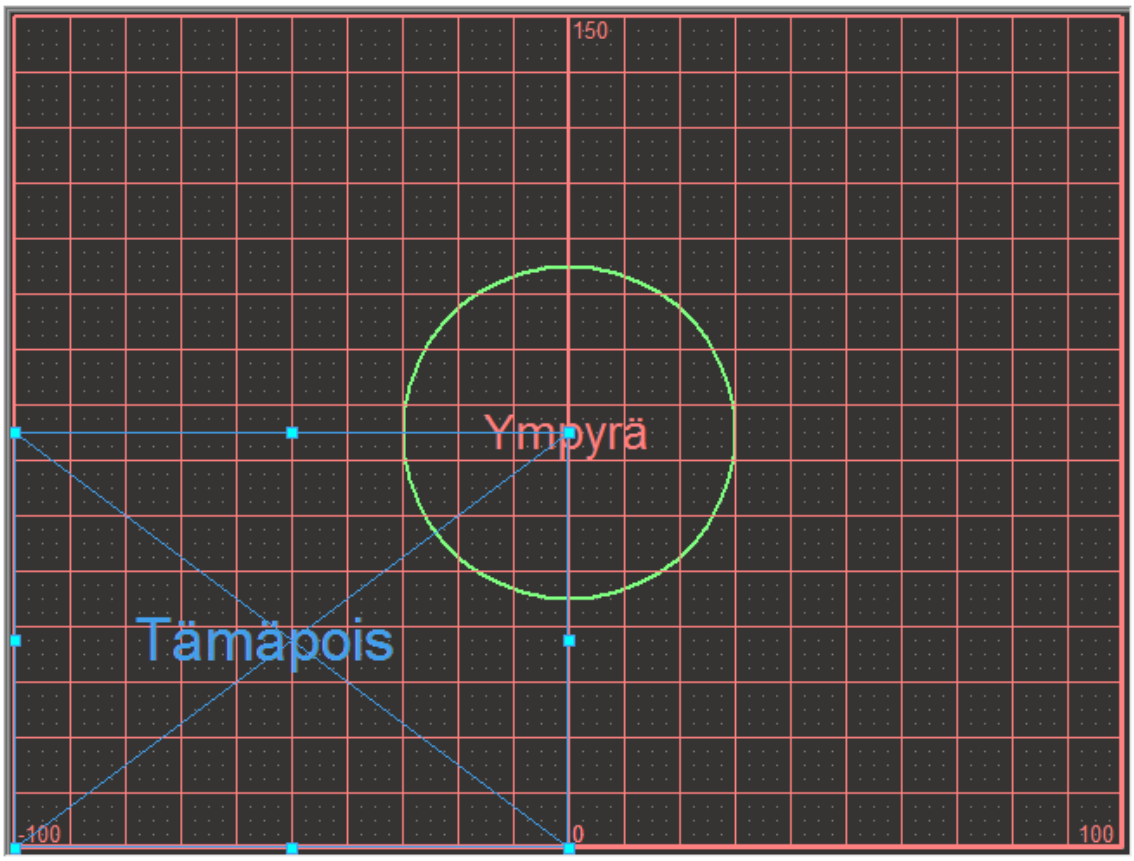
Ympyrän koko määritellään antamalla sille sekä pienin että suurin arvo, jotka mitataan sensorialueen ruudukon x-akselin koordinaattien mukaan. Halutun koon määrittämisessä kuitenkin tulisi huomioida se, ettei suurimman koon arvosta tehtäisi liian suurta, jolloin laskenta-aika kasvaisi ja voisi aiheuttaa ei-haluttujen kohteiden tunnistamisen. Myöskään liian pieni arvo ei ole toivottavaa, koska esimerkiksi liikkeen aiheuttama sumeus tai kuvan epätarkkuus voi tehdä kohteesta todellista suuremman näköisen.



Kuvio 25. Ympyräntunnistimen asetukset-ikkuna

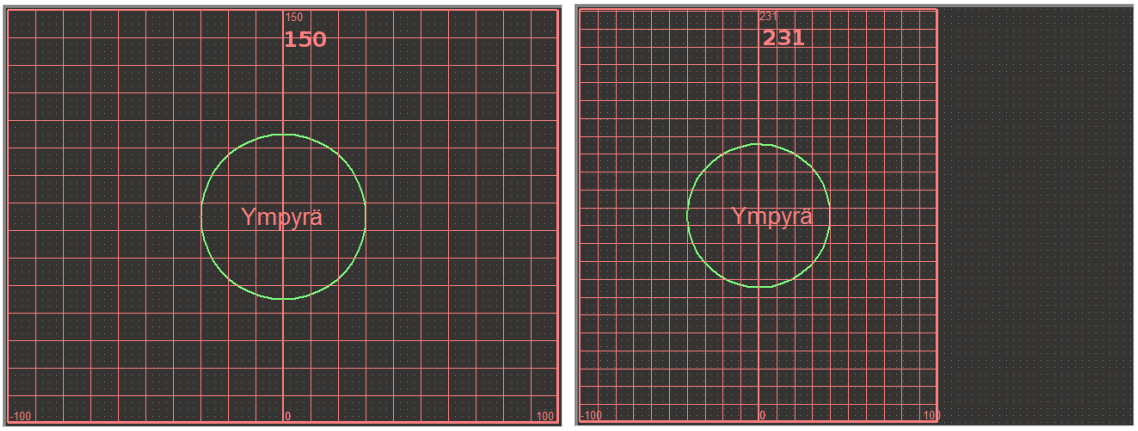
Tunnistusalueelta voidaan myös rajata poissulkevan elementin avulla kohtia, joista ei haluta ympyrää tunnistaa. Poissulkeva alue asetetaan ympyräntunnisti-

men alueelle ja nimetään (Kuva 26.). Saadakseen poissulkevan alueen toimimaan yhdessä ympyräntunnistimen kanssa, täytyy se nimetä ympyräntunnistuksen poissulkeva alue -kohtaan. Näitä alueita voidaan asettaa useampiakin saman tunnistimen alueelle, mutta jotta jokainen niistä toimii, täytyy jokaiselle antaa sama nimi.



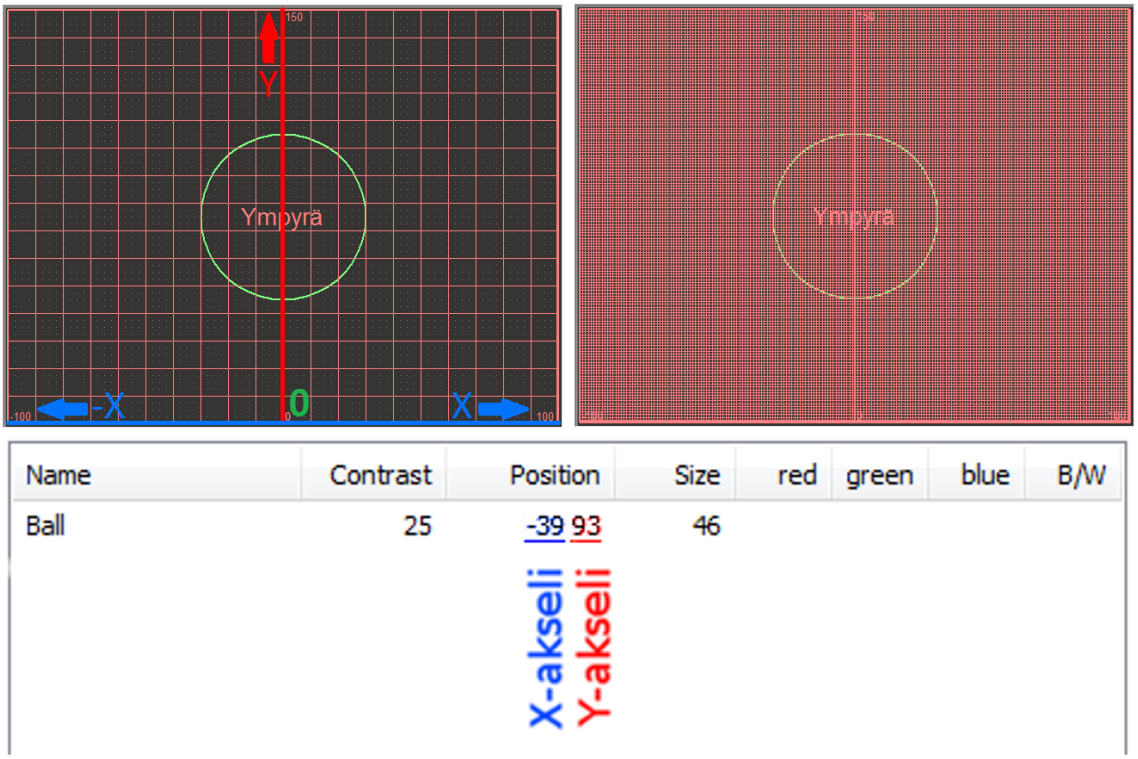
Kuvio 26. Poissulkeva alue asetettuna ympyräntunnistimen alueelle

Tunnistusalueen kokoa ja mallia kameranäkymässä voidaan säätää sekä sen asettamisen yhteydessä että myöhemmin ROBOPron piirrä-valikon kautta. Oletusarvoisesti tunnistusalueen koordinaatiston X-akselille on annettu arvot välille [-100, +100], ja Y-akselin pienimmäksi arvoksi on määritelty nolla (Kuvio 27.). Koordinaatistoa voidaan tarvittaessa säätää antamalla tahdotut arvot X-akselin pienimmälle arvolle [-1000, 0] väliltä, ja suurimalle arvolle [0, +1000] väliltä. Y-akselin suurin arvo määräytyy annettujen X-akselin arvojen ja Y-akselin pienimmän arvon mukaan automaattisesti, jotta molemmat akselit pysyvät samassa mittakaavassa.



Kuvio 27. Tunnistusalue erikokoisina ja automaattisesti säätyvä Y-akselin arvo

Koordinaatiston ruudukon välistystä on mahdollista säätää muuttamalla X- ja Y-akselin välistysarvoja, jotka oletusarvoisesti ovat kooltaan 10 x 10 pistettä (Kuvio 28.). Varsinaista ohjelmallista hyötyä tästä ei ole, mutta silmämääräisessä kappaleiden kokojen arvioinnissa se auttaa, kun välistyksen kooksi voidaan säätää esimerkiksi vain 1 x 1 pistettä.

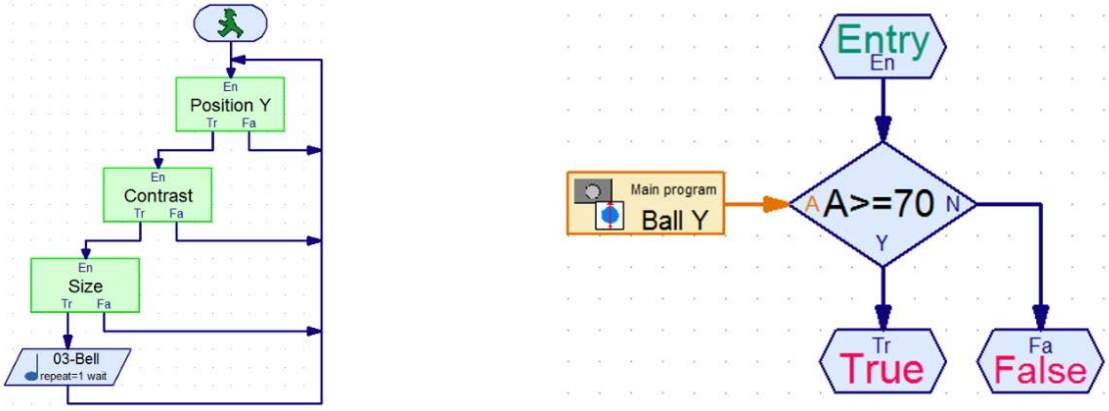


Kuvio 28. Tunnistusalueen koordinaatiston akselit, koordinaatit ja ruudukon välistys asetettuna 1 x 1:een

5.1.2 Ensimmäinen ympyräntunnistuselementin testiohjelma

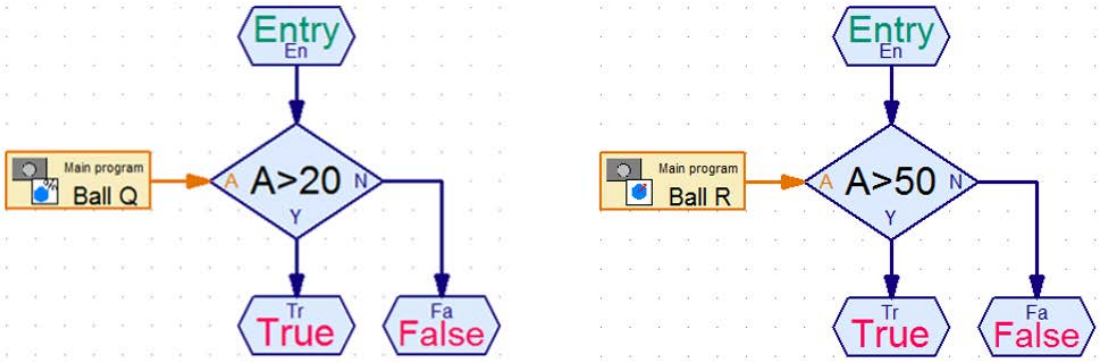
Testatakseni ympyräntunnistuksen toimivuutta, tein yksinkertaisen testiohjelman, jossa robotti tarkistaa onko annetut väitteet ympyrän arvoista tosia, jolloin se jatkaa seuraavaan kohtaa. Väitteen ollessa epätosi, siirtyy ohjelma takaisin alkuun.

Ohjelma koostuu kolmesta aliohjelmasta, joista ensimmäisessä (Kuvio 29.) tutkitaan ympyrän sijaintia Y-akselilla. Asetin vertailtavaksi väitteeksi ”yhtä suuri tai suurempi kuin”, ja vertailtavaksi Y-akselin arvoksi 70, jolloin ohjelma jatkaa kulkuaan true-haaran kautta vain, jos ympyrän sijainti Y-akselilla on 70 tai enemmän.



Kuvio 29. Pääohjelma ja ympyrän sijaintia Y-akselilla tutkiva aliohjelma

Seuraavassa aliohjelmassa tutkittavana on ympyrän kontrasti (Kuvio 30.). Vertailtavaksi väitteeksi annoin ”suurempi kuin 20”, jolloin ympyrän tunnistettuaan ohjelma vertaili ympyrän ja taustan välistä eroa, ja jatkaa true-haaraan arvon ollessa suurempi kuin 20. Kolmannessa ja viimeisessä aliohjelmassa laskettiin ympyrän halkaisijan koko ja sen ollessa suurempi kuin 50, jatketaan ohjelmassa seuraavaan vaiheeseen.



Kuvio 30. Aliohjelmat ympyrän kontrastin ja halkaisijan vertailuun

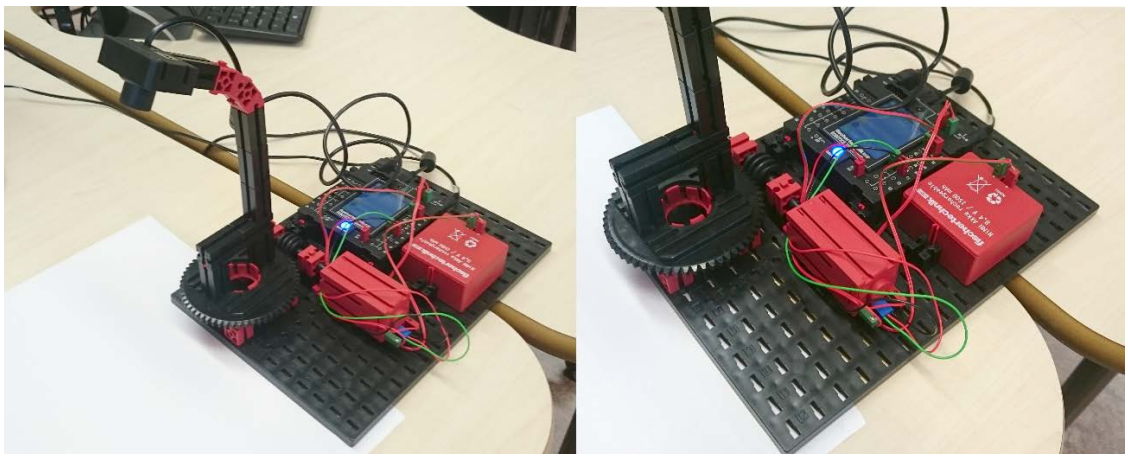
Käytössä tässä minulla oli neljä eriväristä ja -kokoista ympyrän muotoista nappulaa (Kuvio 31.), joiden kokoa ja kontrastia tutkin ennen kuin valmistin ensimmäisen testiohjelmani. Asetin ohjelman arvot siten, että kappaleista keltainen täyttäisi niitä koskevat väitteet ja ohjelma pääsisi loppuun asti vain sitä verratessa. Kuitenkin kappaleista myös sininen ja punainen antoivat suurimmassa osassa testejä lähes samat arvot kontrastille kuin keltainen, joten myös ne pääsivät jatkamaan kyseisen vertailun jälkeen oikeaa reittiä.

Keltaisella kuitenkin on verrattavista kappaleista suurin halkaisija, joten ainoastaan se pääsee viimeiseen kohtaan asti. Tämän varmistaakseni jouduin asettamaan ohjelmaan siinä käytetyn ensimmäisen aliohjelman, jossa tutkittiin tunnistettujen kappaleiden sijaintia Y-akselilla, sillä kamera on jalustallaan hieman vinosti, joka aiheuttaa sen, että mitä alempana Y-akselia kappale on, sitä suuremmalta se kuvassa vaikuttaa. Tämän tapahtuessa sekä sininen, että punainen kappale olisi täyttänyt kokoa vertailevan väitteet ja ohjelma olisi näin ollen virheellisesti päässyt jatkamaan loppuun asti. Vastaavasti ensimmäisen aliohjelman korvaajaksi olisin voinut asettaa ympyräntunnistusalueelle poissulkevan alueen, jolloin kyseistä aliohjelmaa ei olisi tarvittu ollenkaan.

Kuvio 31. Testissä käytetyt kappaleet

5.1.3 Ympyrään kohdistava testiohjelma

Toiseksi ympyräntunnistuksen testiksi tein ohjelman, jossa tutkitaan tunnistetun ympyrän sijaintia tunnistusalueen X-akselilla. Mikäli ympyrän sijainti on jotain muuta kuin nolla, eli kameran tunnistusalueen keskellä, kääntyy robotin kamera (Kuvio 32.) ympyrän suuntaan.

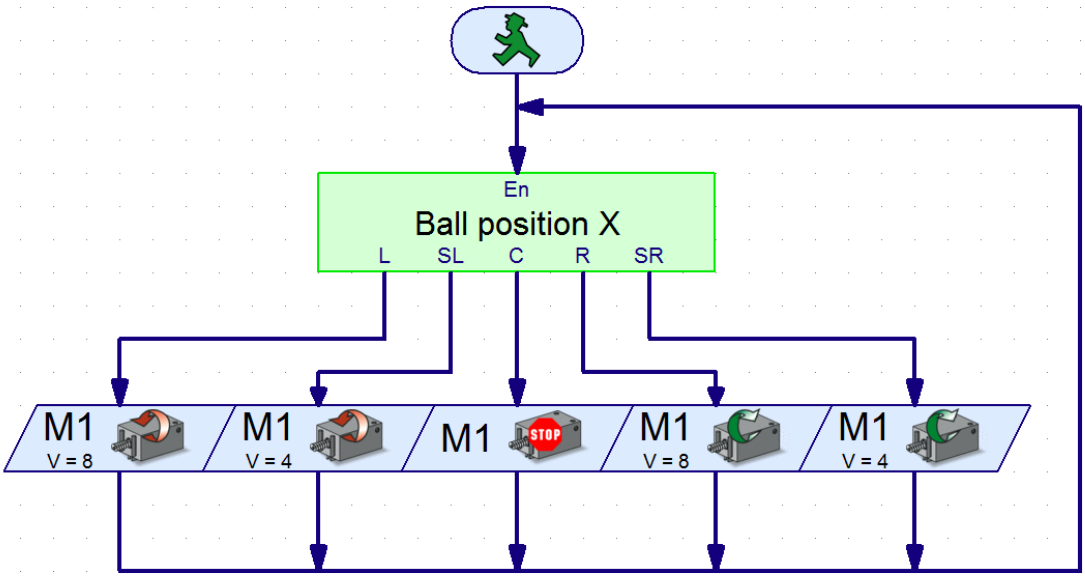


Kuvio 32. Ympyrään kohdistavan robotin laitteisto

Ympyrän sijainnin arvon ollessa pienempi kuin nolla, kiertää ohjelma (Kuvio 33.) robottiin kytkettyä moottoria vastapäivään, jolloin kamera kääntyy vasemmalle,

eli X-akselin negatiiviseen suuntaan. Vastaavasti sijainnin arvon ollessa suurempi kuin nolla, kiertää robotin moottori myötäpäivään, jolloin kamera kääntyy oikealle, eli X-akselin positiiviseen suuntaan. Sijainnin ollessa nolla, moottori pysähtyy, jolloin kamera pysyy kohdistettuna ympyrään.

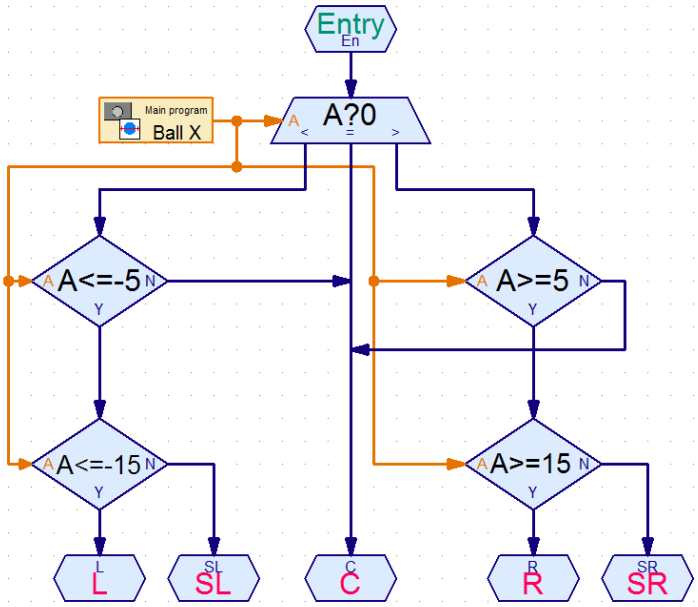
Asetin moottoreiden nopeuden suurimmaksi mahdolliseksi, jotta kamera pysyisi mahdollisimman hyvin ympyrän sijainnin muutoksen mukana. Tämä kuitenkin aiheutti ongelman, jossa moottori ei kerennyt pysähtyä riittävän nopeasti ympyrän ollessa alueen keskellä, jolloin moottori jäi nykimään edestakaiseen liikkeeseen yrittäessään kohdistaa ympyrää kuvan keskelle. Tämän yritin ratkaista lisäämällä ympyrän sijainnille viiden pisteen toleranssin, jolloin moottori pysähtyisi, mikäli ympyrän sijainti ei olisi viittä kuvapistettä kauempana nollapisteestä. Tämä paransi kohdistuksen tarkkuutta huomattavasti pitämällä kuitenkin ympyrän mahdollisimman keskellä tunnistusaluetta.



Kuvio 33. Ohjelma, joka kohdistaa kameran ympyrään

Jotta saisin kohdistuksesta vielä tarkemman, lisäsin ohjelmaan vertailukohdat, joissa tutkittiin, onko ympyrän sijainnin arvo pienempi tai yhtä suuri kuin -15, tai yhtä suuri tai suurempi kuin 15 (Kuvio 34.). Arvon ollessa lähempänä nollaa, kuin -15 tai 15, antaa ohjelma moottorille käskyn vähentää nopeutta puoleen maksi-

minopeudesta, joka puolestaan vähentää mahdollisten edestakaisten korjausliikkeiden tarvetta ympyrän lähestyessä nollapistettä hitaammin ja näin ollen kykeni pysähtymään nopeammin.

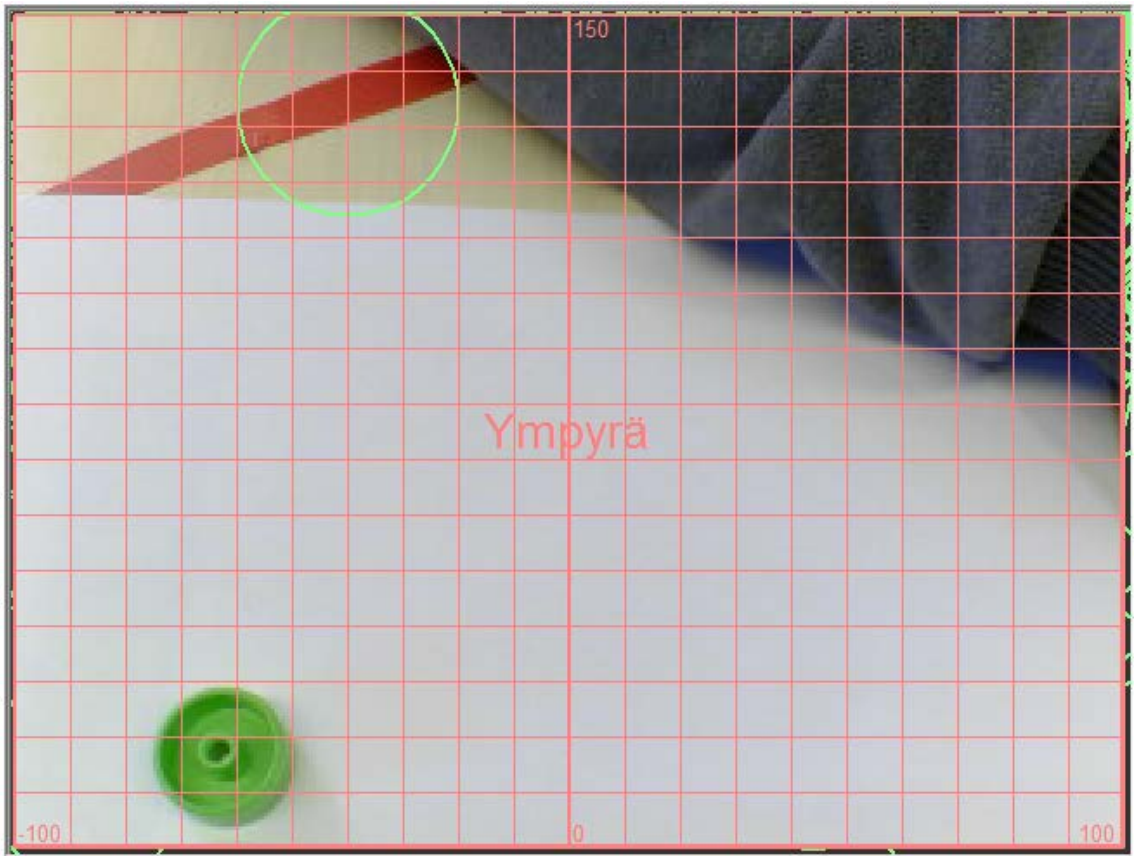


Kuvio 34. Sijainnin tunnistus toleranssilla ja moottorin hidastuksilla

5.1.4 Ympyräntunnistuksen ongelmia

Ympyräntunnistuksen suurin ongelma liittyi sensorin tunnistamiin haamuympyröihin, eli ympyröihin, joita kuvassa ei oikeasti ollut, mutta jotka sensori kuitenkin kuvasta tunnisti. Ongelman ilmetessä sensori ei samaan aikaan pystynyt tunnistamaan kuvassa näkyvää oikeaa ympyrää.

Nämä ilmenivät erityisesti robotin kääntyessä tasaiselta taustaväriltä värikkäämmälle/sekavammalle taustalle (Kuvio 35.). Ongelman olisi pitänyt poistua säätämällä ympyrän halkaisijan suurinta mahdollista pituutta, mutta testeissäni tämä ei kuitenkaan toiminut, sillä ympyröitä kuitenkin ilmestyi tyhjästä.

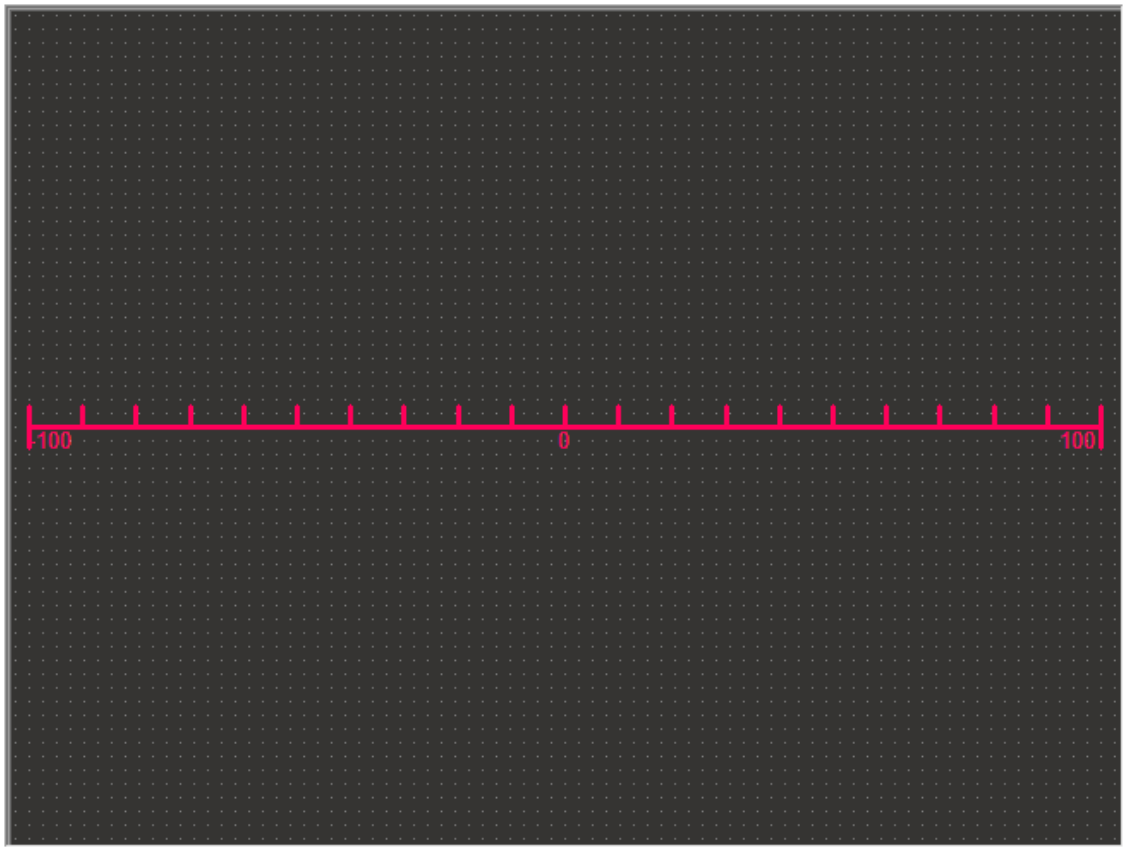


Kuvio 35. Haamuympyrä kuvan kohdassa, jossa ympyrää ei pitäisi olla

5.2 Viivantunnistus

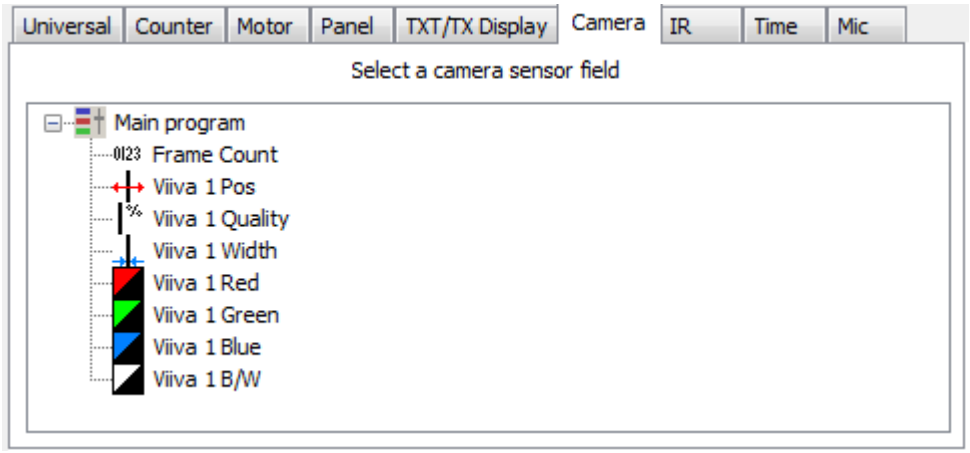
5.2.1 Viivantunnistuksen käyttö

Viivantunnistuselementillä (Kuvio 36.) on mahdollista tunnistaa sekä sen poikki kulkevia viivoja ja linjoja että niiden sijainnit, leveydet ja värit. Tätä sensoria voidaan käyttää robotilla tiettyä linjaa pitkin kulkemiseen ja eri linjojen valintaan, sillä robotilla pystyy tunnistamaan jopa viisi erillistä viivaa samanaikaisesti, joista jokainen saa omat käytettävät sisääntuloelementtinsä.



Kuvio 36. Viivantunnistuselementti kameranäkymässä

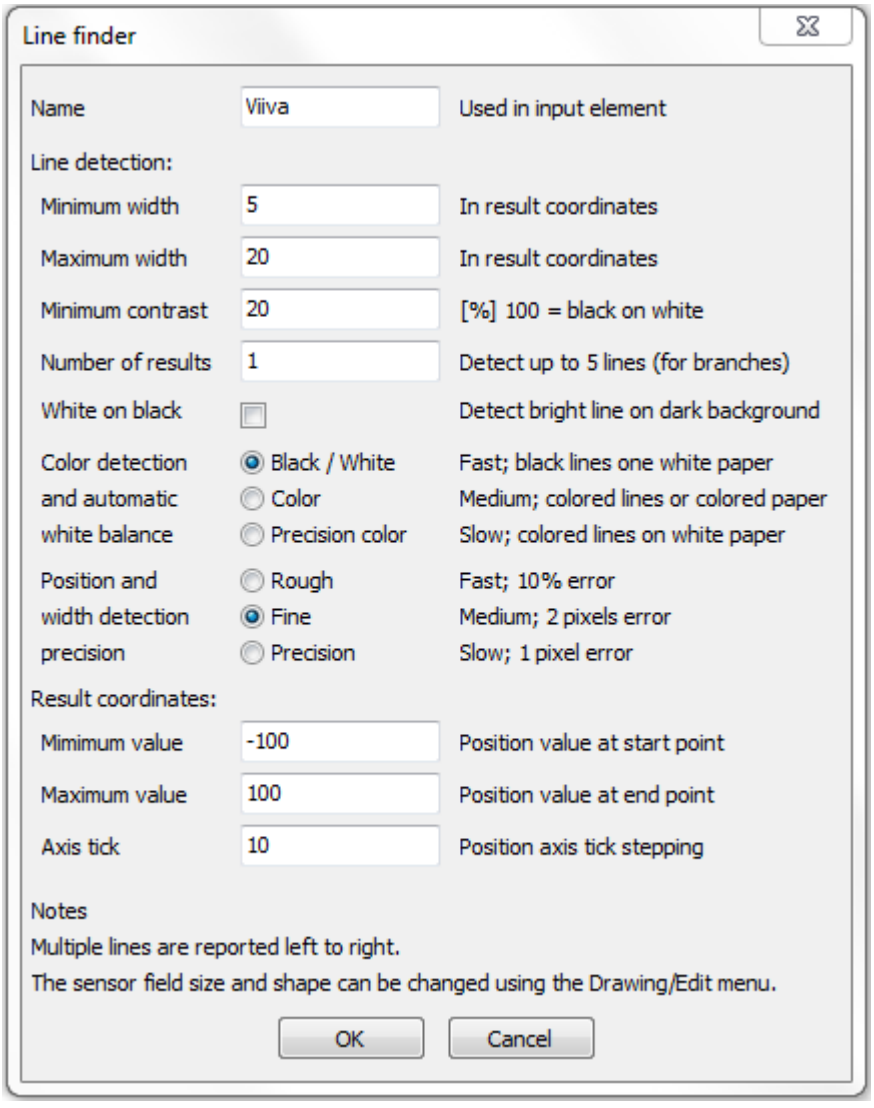
Käytettäviä elementtejä ovat viivan sijainti, laatu, leveys ja väri (Kuvio 37.). Valittavat värit ovat riippuvaisia asetuksissa määritellystä värimoodista, jolloin jos tarkoitus on tunnistaa pelkästään musta valkoisesta, ei RGB-värejä ole valittavissa.



Kuvio 37. Ohjelmoitavat viivantunnistuksen sensorit

5.2.2 Viivantunnistuksen asetukset

Viivantunnistuselementin eri arvoja säädetään samankaltaisesta asetukset-ikkunasta (Kuvio 38.) kuin ympyräntunnistuksessakin. Ensimmäiset säädettävät arvot ovat tunnistettavien viivojen vähimmäis- ja enimmäisleveydet. Leveyttä ei tulisi säätää tarpeettoman suureksi, koska viivantunnistusprosessi vaatisi turhan paljon laskenta-aikaa tunnistaessaan viivan viivaksi. Liikkeen ja kameran aiheuttaman rajojen hämärtymisen vuoksi arvoja ei tulisi myöskään asettaa liian pieniksi. Parhaimmat arvot vähimmäis- ja enimmäisleveyksille saadaan mittaamalla viivan leveyden ROBOPron kameranäkymässä viivantunnistuselementtiä käyttäen, ja muuttamalla arvoja +/- 20 – 50 prosentilla.



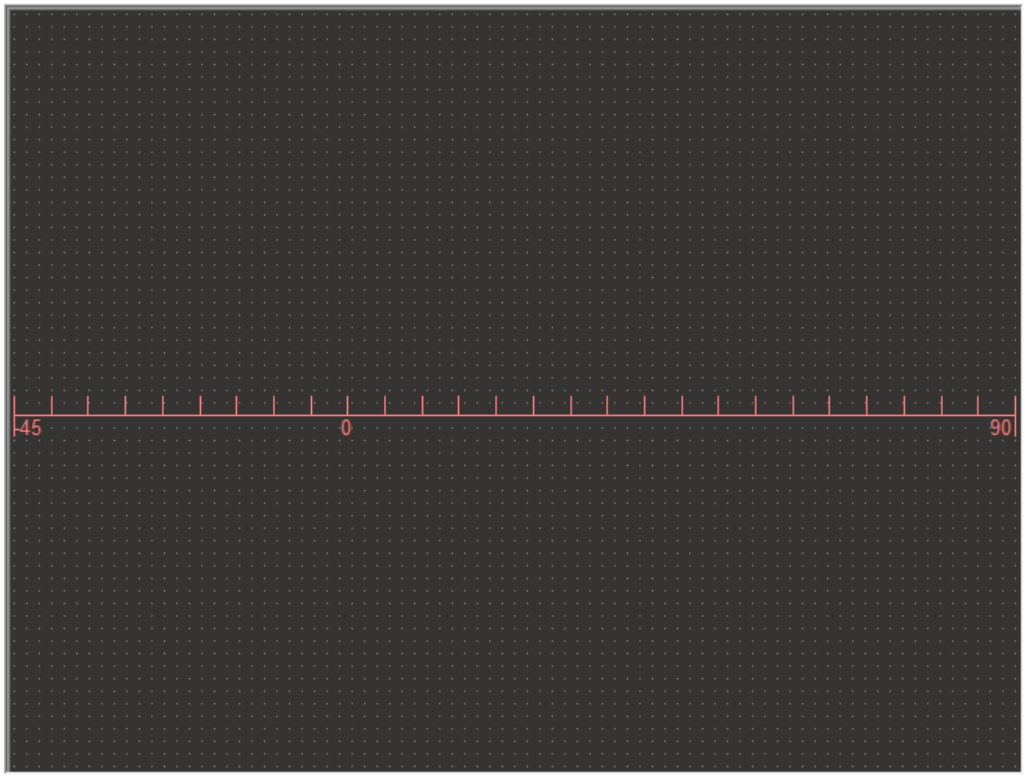
Kuvio 38. Viivantunnistuksen asetukset

Kontrastilla säädetään sitä, kuinka paljon linjan väri tulee poiketa taustasta. Kontrastin arvo sata prosenttia vastaa pikimustaa viivaa valkoisella taustalla, mutta tätäkin säätäessä tulee huomioida sekä liikkeen aiheuttama epätarkkuus että kuvassa näkyvät mahdolliset heijastumat. Mikäli tarkoituksena on tunnistaa kirkkaita viivoja tummalta taustalta, on asetuksista valittava kohta ”White on black”.

Säätämällä värintunnistusta voidaan määrittää, halutaanko tunnistaa mustia viivoja valkoiselta pohjalta, vai tietyn värinen viiva ei-valkoiselta pohjalta. Värimoodia ei tulisi käyttää mustan viivan tunnistamiseen, sillä siinä tilassa sekä musta että punainen viiva antavat lähestulkoon saman kontrastin arvon valkoisella taustalla. Tarkkuustilaa (Precision color) käyttäessä anturielementti käyttää valkoista taustaa valkoisen värin tasapainotukseen. Tätä käyttämällä voidaan värejä tunnistaa tarkemmin, mutta haittapuolena se lisää laskenta-aikaa, joten sitä ei tulisi käyttää tarpeettomasti.

Sijainnin ja leveyden havaitsemisen tarkkuudella voidaan määrittää robotille annettavaa aikaa, jonka se käyttää viivan tunnistamiseen. Jos ohjelmassa on vain yksi tunnistettava viiva, voidaan sen tarkkuus säätää karkeaksi, jolloin kameran anturi antaa viivan koolle ja sijainnille kymmenen prosentin toleranssin, jolloin laskenta-aika on lyhyempi. Erikokoisten ja väristen viivojen tunnistamisessa on järkevämpää käyttää tarkempia valintoja, jolloin virheiden todennäköisyys laskee, mutta tässäkin tapauksessa laskenta-aika pitenee.

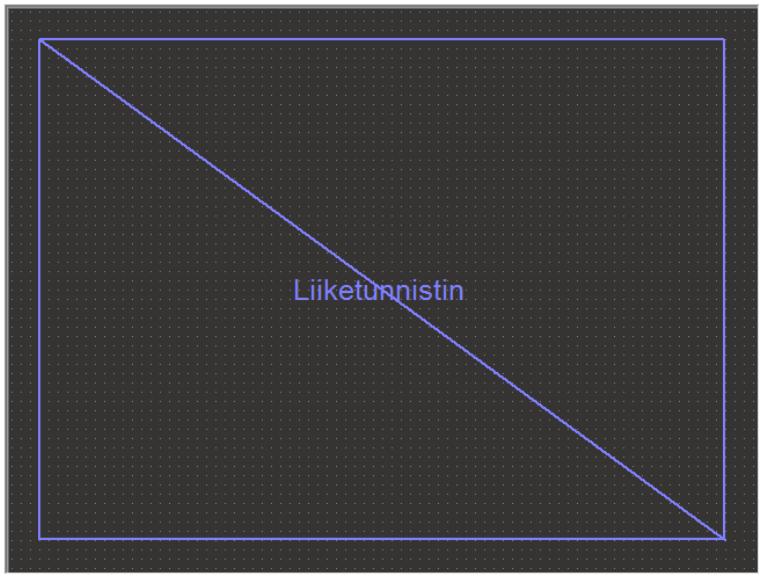
Viimeisessä osassa on mahdollista muuttaa koordinaatiston arvoja, sekä niiden merkintätiheyttä (Kuvio 39.). Arvoja muutettaessa on muistettava skaalata viivantunnistuksen arvot samoille arvoille. Säätämällä koordinaattien merkintätiheyden esimerkiksi viiteen, saadaan tunnistettavat viivat mitattua silmämääräisesti viiden pisteen tarkkuudella, kun oletusarvoisesti merkintätiheys on säädetty kymmeneen pisteeseen.



Kuvio 39. Muokatut viivantunnistuselementin koordinaatit ja merkintätiheys

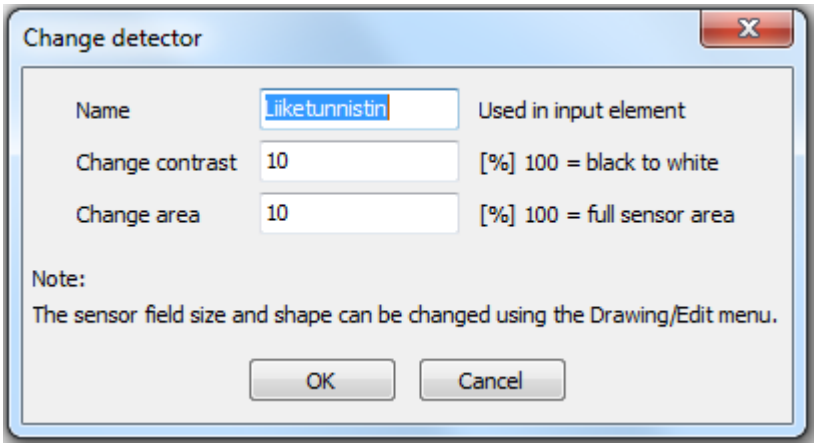
5.3 Liikkeentunnistus

Liikkeentunnistuselementillä (Kuvio 40.) voidaan havaita sensorialueen sisällä tapahtuvia muutoksia. Tätä voidaan käyttää esimerkiksi murtohälyttimissä tai riistakameroissa.



Kuvio 40. Liikkeentunnistuselementti asetettuna kameranäkymään

Liikkeentunnistuksen asetuksista (Kuvio 41.) täytyy asettaa kaksi arvoa, joihin kameran kuvassa tapahtuvia muutoksia verrataan. Näillä arvoilla määritellään muutoksen kontrasti ja muutosalueen koko.

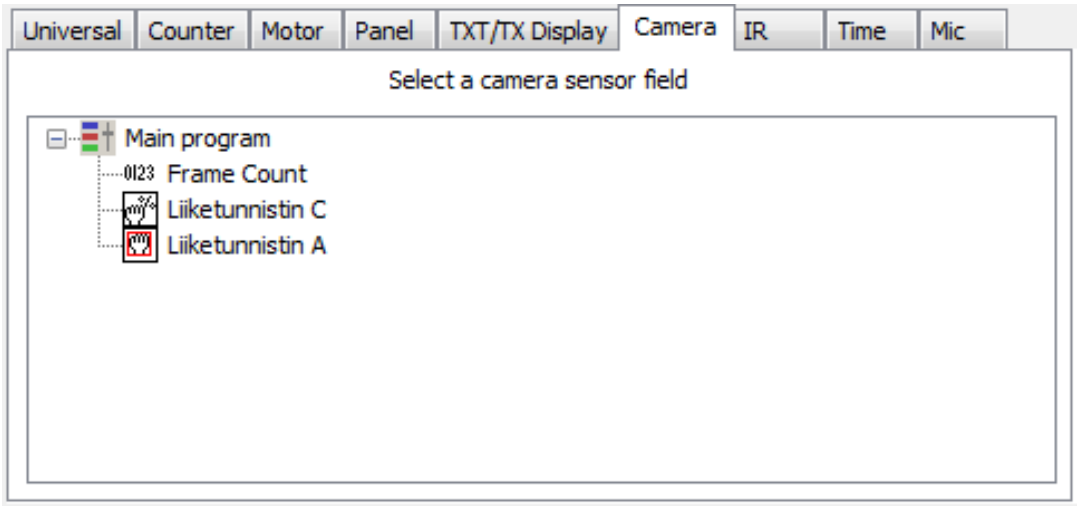


Kuvio 41. Liikkeentunnistuksen asetukset

Kontrastilla määritellään, kuinka suuresti pikselin kirkkautta muutetaan. Jos tarkoituksena on tunnistaa esimerkiksi musta kiekko jäältä, eli täydellinen muutos mustavalkoisella skaalalla, asetetaan kontrastin arvoksi sata prosenttia.

Muutosalueen koolla määritetään tunnistusalueen osuus prosentteina, jolla tapahtuva kontrastin muutos huomioidaan. Jos kyseessä olisi terassia kuvaava valvontakamera, olisi muutosalueen arvon oltava riittävän suuri, jotta esimerkiksi putoava lehti ei riittäisi laukaisemaan sitä.

Molempia arvoja pystytään vertailemaan ohjelmassa erikseen niiden omilla syötöillä (Kuvio 42.). Tällöin vertailtavaksi arvoksi asetetaan vain kontrasti esimerkiksi tilanteessa, jossa tarkoituksena on tutkia ainoastaan kuvassa tapahtuvaa liikettä, riippumatta siitä, kuinka suurella alueella liike tapahtuu.



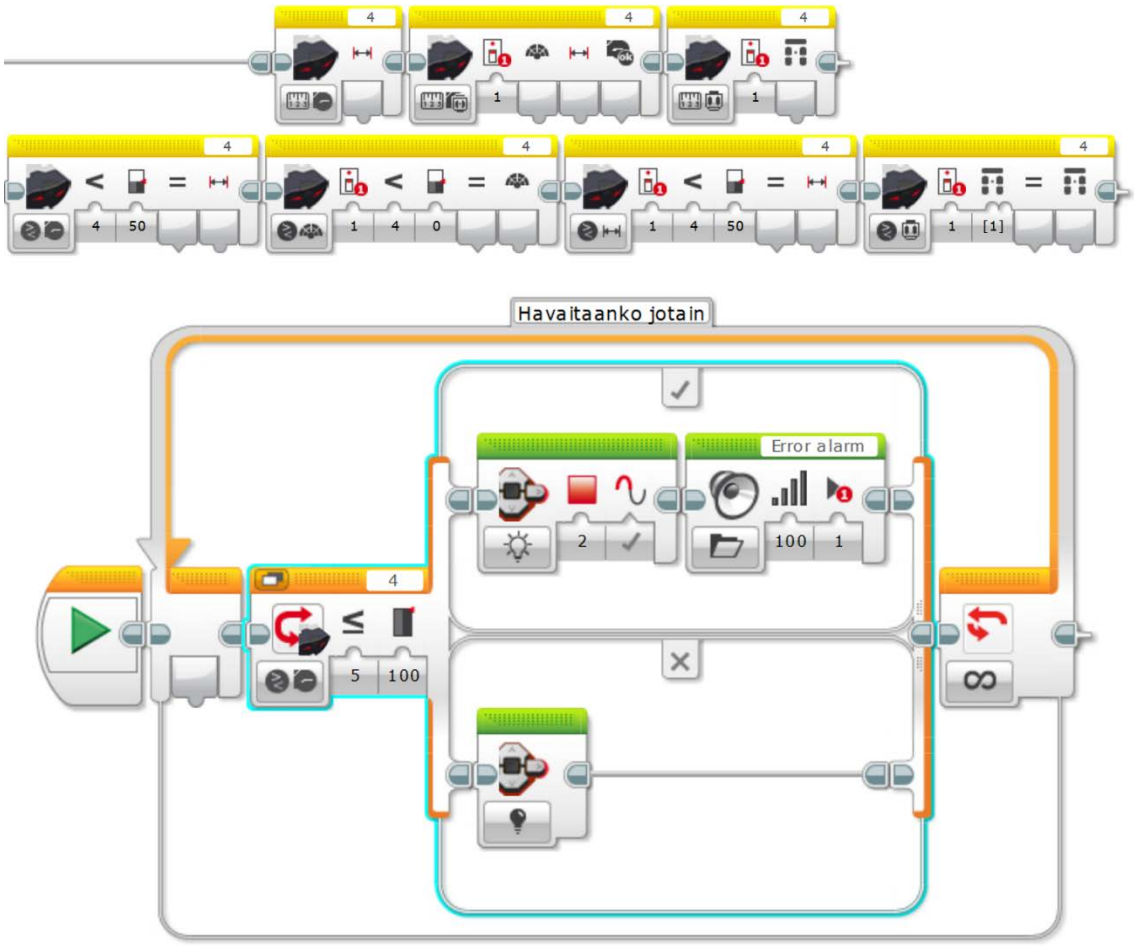
Kuvio 42. Liikkeentunnistukseen liittyvät ohjelman asetukset

5.4 Kuvantunnistus Mindstorms EV3:lla

Legon Mindstormsin mukana ei tule Fischertechnikin USB-kameran kuvantunnistuksen ohjelmointielementtejä vastaavia sensoreita, joilla voitaisiin tunnistaa muotoja tai liikettä. On kuitenkin mahdollista käyttää robotin mukana tulevaa infrapunasensoria (Kuvio 43.) sen edessä tapahtuvien muutosten, esimerkiksi esteiden tai sensorin ohitse kulkevien objektien havaitsemiseen. Viivan seuraaminen voitaisiin hoitaa värintunnistussensorin kanssa.

Kuvio 43. Lego Mindstorms EV3 infrapunasensori (Lego 2017a)

Infrapunasensorilla on mahdollista mitata kohteen etäisyyttä, mukana tulevan infrapunalähettimen suuntaa ja etäisyyttä, ja tunnistaa mitä infrapunalähettimen painikkeita on painettu (Kuvio 44.). Liikkeentunnistimenakin sensoria on mahdollista käyttää, mutta liikkeen on tapahduttava suoraan sensorin edessä.



Kuvio 44. Infrapunasensorin ohjelmointielementit ja yksinkertainen liikkeentunnistusohjelma

6 POHDINTA

ROBOTICS TXT Controller ja ROBOPro ovat erittäin monipuolisia robotiikan opettelun työkaluja. Graafinen ohjelmointiympäristö on mieluinen käytettävä ja sen käytön perusperiaate on helppo opetella. Huonona puolena ROBOPron käytössä on Fischertechnikin huono dokumentaatio ohjelmointielementtien käyttöön. Suuri osa käytettävien ohjelmointielementtien käyttötarkoituksista ja tavoista jäi minulle epäselväksi, sillä en löytänyt niiden käyttöön lyhyttä selitystä kummoisempaa dokumentaatiota. Ajan puutteellisuuden ja opinnäytetyöni rajauksen takia en ruvennut jokaista ohjelmointielementtiä tutkimaan tarkemmin, vaan keskityin vain tarvitsemiini elementteihin.

Värintunnistuksessa suurimmaksi ongelmaksi nousi kameran epätarkkuuden ja valkotasapainotuksen puuttumisen vuoksi värien vääristyminen. Päävärejä tunnistaessa tätä ongelmaa ei niinkään esiintynyt, mutta välivärien tunnistamisen vaikeus tuntui liian suurelta, varsinkin Mindstormsin kohtuullisen toimivaan ja valmiiseen värintunnistussensoriin verrattuna.

Legon Mindstorms EV3:een verrattuna ROBOTICS TXT Controllerin mahdollisuudet ovat laajemmat, mutta se vaatisi tarkempaa ohjelmaan tutustumista ja sen käytön opettelua. Mindstormsin vahvana puolena mainittakoon sen kattava dokumentaatio sekä internetissä että Mindstorms ohjelmointiympäristön sisällä. Edellä mainittujen syiden vuoksi en olisi valmis jättämään Mindstormsia kokonaan pois robotiikan opetuksesta, sillä niiden käyttöön on helpompi päästä sisälle ja sitä kautta myös mahdollisesti oppia graafisen ohjelmoinnin periaatteita.

Jatkokehitysideana ROBOTICS TXT Controllerin käyttöön ehdottaisin kattavamman dokumentaation tekemistä. Se sisältäisi kuvauksen jokaisesta ohjelmointielementistä, niiden käyttötarkoitukset ja esimerkit niiden käyttöön ohjelmoinnissa. Dokumentaation voisi toteuttaa opinnäytetyönä tai koulun tarjoamana harjoitteluna tieto- ja viestintätekniikan opiskelijoille.

LÄHTEET

Crunchbase 2017. Fischertechnik GmbH. Viitattu 23.11.2017 https://www.crunchbase.com/organization/fischertechnik-gmbh.

DigiFAQ 2008. Mikä on valkotasapaino/värilämpötila? Viitattu 4.10.2017 http://digifaq.info/digifaq/3_valko.html.

Fischertechnik GmbH 2010. History. Viitattu 22.11.2017 https://web.archive.org/web/20110722040829/http://www.fischertechnik.de:80/en/desktopdefault.aspx/tabid-54/.

Fischertechnik GmbH 2017. ROBOTICS TXT Controller. Viitattu 6.11.2017 https://www.fischertechnik.de/en/products/playing/robotics/522429-robotics-txt-controller.

Huttunen, M. 2005. Värit Pintaa Syvemmältä. Helsinki: WSOY.

Lego 2017a. 31313 Mindstorms EV3. Viitattu 22.11.2017 https://www.lego.com/en-us/mindstorms/products/mindstorms-ev3-31313.

Lego 2017b. LEGO Mindstorms EV3 Help.

Lego 2017c. Lego History Timeline. Viitattu 23.11.2017 https://www.lego.com/en-us/aboutus/lego-group/the_lego_history

Motiva Oy 2017. Värilämpötila – kelvin-arvo. Viitattu 25.9.2017 https://lamputieto.fi/lampun-valinta/lamppujen-ominaisuuksia/kelvin-varilampotila/.

Valostore 2017. Lumenit, luxit ja candelat – Ja miten ne mitataan. Viitattu 22.11.2017 http://www.valostore.fi/lumen_lux_candela_integroiva_pallo/.

Van Niekerk, H. 2015. Fischertechnik TXT Controller: How Colour Detection in ROBOPro Should Be Working #1. Viitattu 10.9.2017 https://www.youtube.com/watch?v=RtK0_nK_3Gg.

Westinghouse 2017. Understanding Color Temperature. Viitattu 25.9.2017 http://www.westinghouselighting.com/color-temperature.aspx.

Wheeler, S. 2017. RGB Color Wheel. Viitattu 11.9.2017 https://www.pinterest.com/pin/158963061820841322/.

ÄET–hanke 2016a. Investoinnit. Viitattu 3.11.2017 http://www.lapinamk.fi/fi/Tyoelamalle/Tutkimus-ja-kehitys/Lapin-AMKin-hankkeet?RepoProject=521824.

ÄET–hanke 2016b. Kickstart. Viitattu 3.11.2017 http://www.lapinamk.fi/fi/Tyoelamalle/Tutkimus-ja-kehitys/Lapin-AMKin-hankkeet?RepoProject=521621.

ÄET–hanke 2016c. Osaaminen. Viitattu 3.11.2017 http://www.lapinamk.fi/fi/Tyoelamalle/Tutkimus-ja-kehitys/Lapin-AMKin-hankkeet?RepoProject=521627.