Niko Larmala

Lämpökamera ja matemaattinen kuvankäsittely

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Sähkö- ja automaatiotekniikka

Insinöörityö

14.12.2021

Tiivistelmä

Tekijä:	Niko Larmala
Otsikko:	Lämpökamera ja matemaattinen kuvankäsittely
Sivumäärä:	29 sivua
Aika:	14.12.2021
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Sähkö- ja automaatiotekniikka
Ammatillinen pääaine:	Elektroniikka
Ohjaajat:	Lehtori Anssi Ikonen

Insinöörityön tavoitteena oli lämpökameran rakentaminen sekä siihen tarvittavien osien toimintaperiaatteiden selvittäminen. Lisäksi työssä käytiin läpi matemaattisen kuvankäsittelyn perusteita. Työn tekninen painotus oli digitaalikameroiden toiminnan tutkiminen sekä kuvien käsittelyyn vaadittavan matematiikan selvittäminen.

Lämpökameran rakentamisen ohella tarkasteltiin erilaisten kamerasensoreiden toimintaan liittyviä ilmiöitä muun muassa sitä, kuinka kamerat havainnoivat maailmaa ja kuinka kuvat muunnetaan digitaaliseen muotoon.

Työssä käytiin läpi projektiin valitut osat ja lämpökameran suunnittelua. Valituille osille luotiin yhteensopiva piirilevy ja toimiva ohjelma. Ohjelmoinnissa toteutettiin vaadittavat ojaukset komponenteille ja toimivat kuvankäsittelyt, jotta kuva voidaan selkeästi piirtää näytölle.

Työn lopputuloksena syntyi toimiva lämpökamera, jota voidaan hyödyntää esimerkiksi lämpövuotojen etsimiseen. Tämä työ toimii hyvänä pohjana ja sen päälle voi kehittää omiin tarpeisiin sopivia ratkaisuja.

Avainsanat: lämpökamera, ohjelmointi, reunantunnistus

Abstract

Author:	Niko Larmala
Title:	Thermal Camera and Mathematical Image Processing
Number of Pages:	29 pages
Date:	14 December 2021
Degree:	Bachelor of Engineering
Degree Programme:	Electrical and Automation Engineering
Professional Major:	Electronics
Supervisors:	Anssi Ikonen, Senior lecturer

The goal of this bachelor's study was to build a functional thermal camera as well as study the working principles of the parts needed for this project. In addition, this study goes through the basics of mathematical image processing. The technical emphasis is to look how digital imaging works and find out the math needed for processing images.

In addition to building a thermal camera the aim of this research was to find how digital cameras sense the world and how these images are digitized.

This study goes through the parts chosen for this project and looks at the design process of the thermal camera. A functional PCB was created to fit all the needed parts. The device was programmed to control everything and to process images.

The result of this project is a functional thermal camera that can be used for example to find thermal leaks.

Keywords: thermal camera, programming, edge detection

Sisällys

Lyhenteet

APS:	Active pixel sensor. Aktiivipikselisensori. Sensorin jokaisella pikselillä on oma sisäänrakennettu vahvistin.
A/D-muunnin:	Analogia-digitaalimuunnin. Komponentti, joka muuntaa analogisen jännitteen digitaaliseksi arvoksi.
CCD-kenno:	Charge-coupled device. Valoherkkä kenno, muuntaa valonsäteet digitaalisiksi numeerisiksi arvoiksi.
CMOS-kenno:	Complementary Metal-oxide-semiconductor. CMOS-teknologiaan perustuva valoherkkä kenno.
FIR:	Far infrared. Infrapunasäteilyn pisimpien allonpituuksien osa.
I2C:	Inter-Integrated Circuit. Sarjakommunikaatioprotokolla. Kaksi johdinta.
I/O:	Input/output. Sisääntulo/ulostulo.
MCU:	Minimum coded unit. JPEG-tiedostomuodon pakkaus koostuu näistä. Usein kooltaan 8 x 8, 16 x 8 tai 16 x 16 pikseliä.
MOSFET:	Metal-oxide-semiconductor field-effect transistor. Metalli-oksidi-puolijohdekanavatransistori.
NMOS:	N-tyypin MOSFET. MOSFET, jonka puolijohdekanava on n-tyyppiä.
nm:	Nanometri.

PPS:	Passive pixel sensor. Passiivipikselisensori. Sensori, jonka pikseleiden varaukset siirretään yksitellen erilliseen vahvistimeen.
SPI:	Serial Peripheral Interface. Sarja kommunikaatio protokolla, jota käytetään oheislaitteiden ohjaukseen. Neljä johdinta.
UART:	Universal asynchronous receiver-transmitter. Sarja kommunikaatio protokolla. Kaksi johdinta.
V:	Voltage. Jännite.

1 Johdanto

Digitaalikamerat ovat yleistyneet paljon esimerkiksi teollisuuden ja turvajärjestelmien käytössä. Kotikäyttöön soveltuvat lämpökamerat ovat kuitenkin melko kalliita ja eivät välttämättä sovellu kaikkien budjettiin. Työn tavoite on osoittaa, kuinka lämpökameran voi rakentaa omaan käyttöön yksinkertaisesti helposti saatavilla olevista osista.

Tämä opinnäytetyö käsittelee digitaalikameroiden toimintaperiaatteita sekä otettujen kuvien käsittelyä matemaattisesti. Työssä käydään läpi, kuinka valokuvista saadaan matemaattisesti eroteltua paljon hyödyllistä tietoa erilaisilla reunantunnistusmenetelmillä. Tutustutaan, kuinka kuvia voidaan yhdistää ja piirtää päällekkäin.

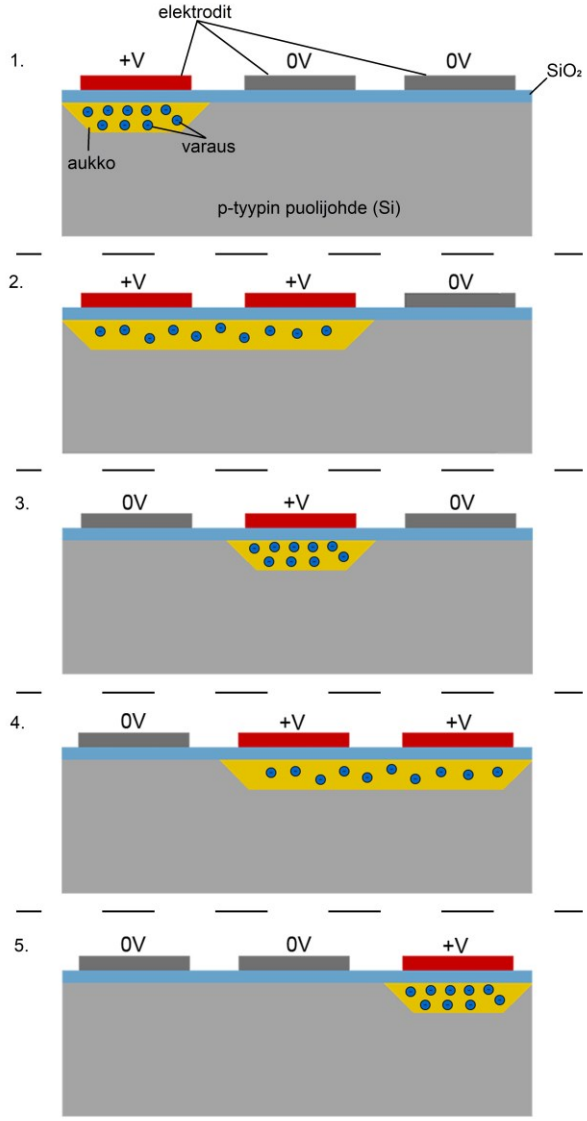
Rakennettua lämpökameraa voidaan hyödyntää esimerkiksi lämpövuotojen löytämisessä. Lopuksi käydään läpi kehitysideoita, kuinka projektia voidaan jatkaa eteenpäin.

2 Digitaalikamera

2.1 CCD-kenno

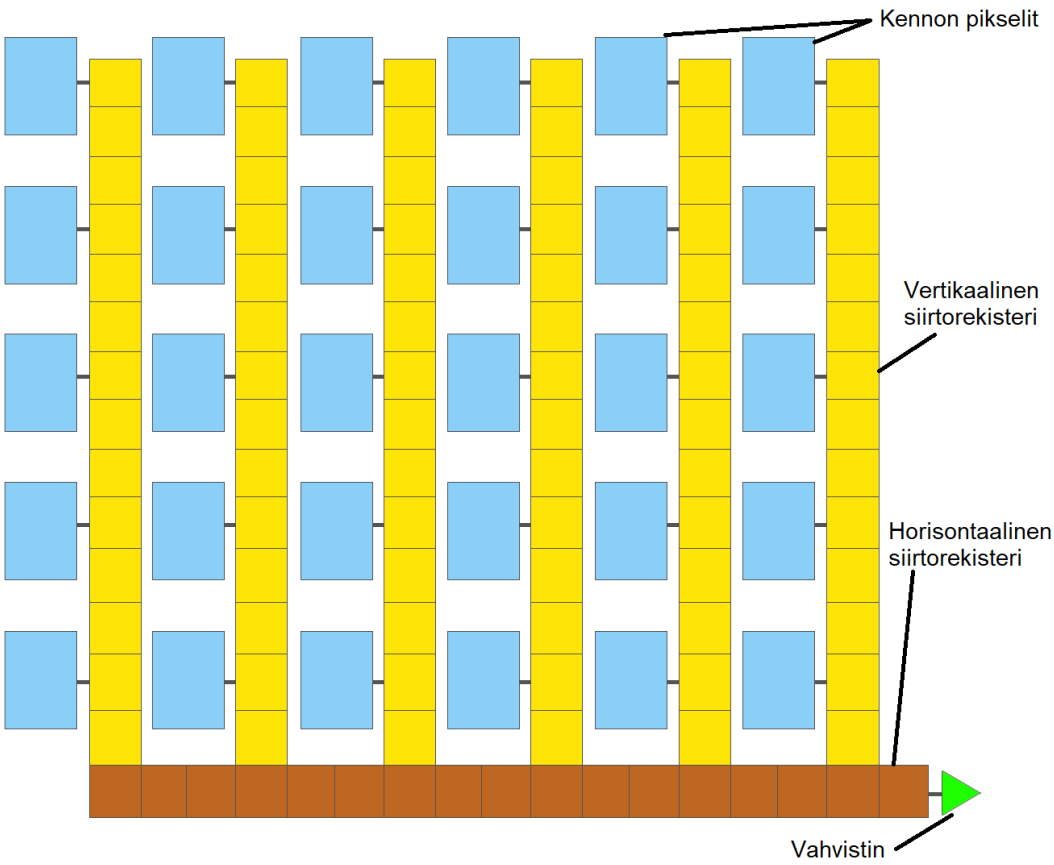
CCD-teknologian kehittivät Williard Boyle ja George E. Smith vuonna 1969 tutkiessa MOS-teknologiaan perustuvaa muistia. Vuotta myöhemmin ensimmäinen CCD-teknologiaan perustuva siirtorekisteri luotiin ja 1974 ensimmäiset CCD-kuvasensorit alkoivat ilmestymään markkinoille. [1.]

CCD-teknologia perustuu sähköisenvarauksen siirtoon puolijohdemateriaalissa ja se toimii kuin siirtorekisteri. Oikein ajoitettu elektrodien kytkeminen päälle ja pois saa elektronit liikkumaan puolijohteessa aukosta toiseen. Kuvassa 1 on havainnollistettu varauksen siirto. [1.]



Kuva 1. Varauksensiirto aukosta toiseen puolijohteessa.

Digitaalisissa kuvakennoissa valoherkät pikselit saavat varauksen fotonien osumasta ja varauksen suuruus on suoraan verrannollinen valon määrään. Kuva luetaan CCD-kennosta ulos siirtämällä jokaisen pikselin varaus ensin vertikaaliseen siirtorekisteriin ja horisontaalisen siirtorekisterin kautta vahvistimeen ja analogia-digitaalimuuntimeen. Kuvassa 2 on merkitty CCD-kennon osat. [2.]



Kuva 2. CCD-kennon osat.

CCD-kennossa siirtorekisterit toimivat kuvan 1 osoittamalla tavalla ja varaukset luetaan riveittäin vahvistimeen ja siitä A/D-muuntimeen. CCD-kennoa kutsutaan passiivipikselisensoriksi, sillä sen pikselit itsessään ei sisällä yhtään aktiivisia komponentteja.

2.2 CMOS-kenno

Ensimmäisiä aktiivipikselisensoreita alettiin valmistaa 1980-luvulla ja niissä käytettiin alkuun NMOS-transistoreita. APS-kennossa jokaisella pikselillä on oma transistorivahvistin, joka vahvistaa pikselin varauksen suoraan pikselissä. CMOS-kennot alkoivat yleistyä 1990-luvulla, kun CMOS-teknologia oli kehittynyt jo hyvin pitkälle. Syy CMOS-kennon yleistymiselle oli CMOS-teknologian pienempi virran kulutus sekä helpompi valmistus. Vahvistimen sijoittaminen suoraan

pikseliin vähentää häiriötä sekä nopeuttaa pikseleiden lukemista huomattavasti. [2; 3.]

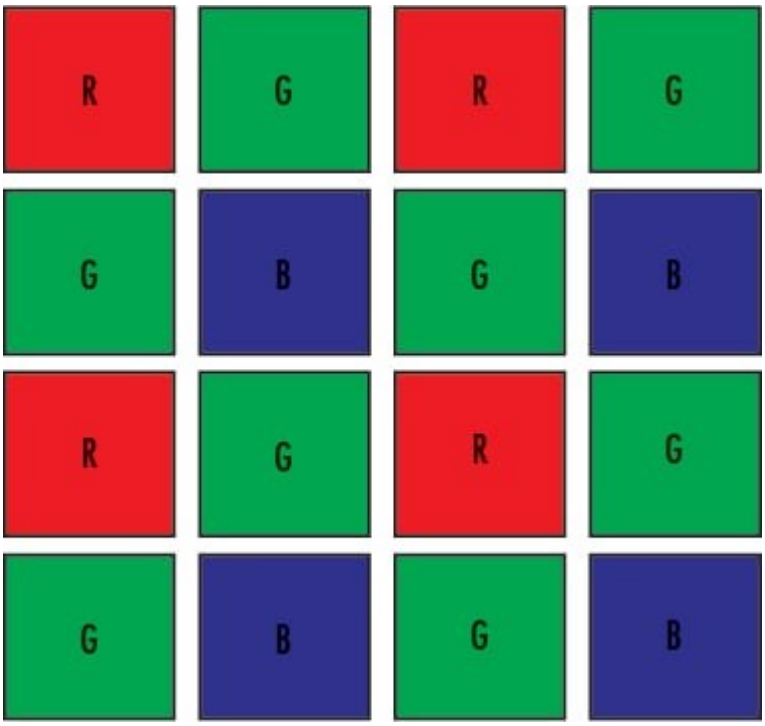
2.3 Valotus

Kuvasensorin toiminta pohjautuu elektromagneettiseen säteilyyn ja fotonien törmäyksen aiheuttamaan energiansiirtoon. Todellisuudessa näkyvä valo on vain pieni osa elektromagneettisen säteilyn spektriä, joka tarkoittaa, että joudumme suodattamaan kaikki ei-halutut aallonpituudet pois. Näitä ei-haluttuja aallonpituuksia voi olla esimerkiksi ultravioletti- ja infrapunasäteily, kun yritämme kuvata näkyvää valoa. Ylimääräiset aallonpituudet usein vääristävät kuvaa ja värejä.

Näkyvä valo ja värit

Itsessään CCD- ja CMOS-kennojen herkkyysalue on noin 350–1050 nm. Tämä kattaa näkyvän valon, 380–750nm aallonpituuksien, lisäksi osan ultravioletti- ja infrapunasäteilyä. Laajasta herkkyysalueesta johtuen pelkkä kenno ei pysty erottamaan eri värien aallonpituuksia toisistaan ja otettu kuva on harmaasävyinen.

Värien havaitsemiseksi jokaiselle pikselille pitää antaa oma aallonpituus suodattamalla muut värit pois. Kuvassa 3 on yksi yleisimpiä suodattimia värikameroille. Bayer-suodatin on kooltaan 2 x 2 pikseliä ja se osoittaa kaksi pikseliä vihreälle valolle, yhden punaiselle ja yhden siniselle. Syy kahdelle vihreälle pikselille pohjautuu ihmisen biologiaan ja siihen, että ihmissilmä on herkempi vihreälle valolle kuin muille väreille. [4.]



Kuva 3. Bayer-suodatin värikameroille [2.].

Värisuodattimet antavat meille väritietoa eri kohdista kuvaa, mutta emme saa kaikkia värejä jokaiselle pikselille. Esimerkiksi joudumme arvioimaan punaisen värin määrän jokaisen vihreän pikselin kohdalla. Puuttuvien värien täydentäminen tapahtuu interpoloimalla tiedossa olevien pikseleiden avulla. [5.]

Lämpökamera

Esineillä ja asioilla on mielenkiintoinen ominaisuus, sillä niiden vapauttaman infrapunasäteilyn määrä on verrannollinen lämpötilaan. Lämpimät ja kuumat materiaalit säteilevät enemmän infrapunasäteilyä kuin kylmät. Mittaamalla vapautuvan infrapunasäteilyn määrää pystymme arvioimaan lämpötiloja melko tarkasti. [6.]

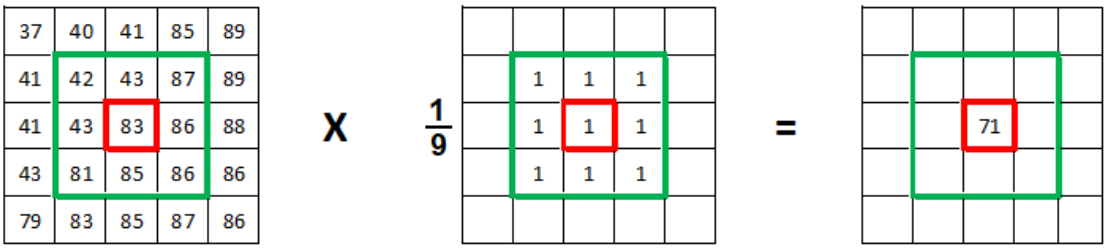
Infrapuna-aaltojen havainnointi toimii samalla tavalla kuin värien näkeminen, mutta lämpötilojen mittaaminen ei olekaan aivan niin helppoa. Eri materiaalit vapauttavat infrapunasäteilyä eri suhteessa, ilma väliaineena vääristää mittausta, sensorin jokainen pikseli täytyy olla kalibroitu. Lisäksi sensorin sisäinen lämpötila

ja jännitelähteen heilahtelut vaikuttavat mittauksiin. Lämpötilaa laskiessa on paljon huomioon otettavia asioita, jotka vaikuttavat lopulliseen tulokseen.

3 Matemaattinen kuvankäsittely

3.1 Kuvien käsittely matriiseilla

Yksi yleisimpiä tapoja käsitellä kuvia on matriisien avulla. Väri kuvat muodostuvat punaisesta, vihreästä ja sinisestä matriisi tasosta ja jokaista tasoa on käsiteltävä erikseen. Eri tehosteiden saavuttamiseksi näitä tasoja käsitellään suodinmatriiseilla. Tätä matriisin käsittelyä toisella matriisilla kutsutaan konvoluutioksi ja suotimia kutsutaan ytimiksi. Kuvassa 4 on esimerkki konvoluutiosta, jossa kuvamatriisi kerrotaan ytimellä. Kuvamatriisin arvot kerrotaan ytimen vastaavalla arvolla ja tulokset summataan yhteen. Punainen alue osoittaa uuden halutun pikselin ja vihreä rajaa ytimen toiminta-alueen. [7.]



Kuva 4. Konvoluutio, eli matriisin käsittely toisella matriisilla.

Kuvan 4 esimerkki sumentaa kuvaa. Jokaiseen uuteen pikseliin lisätään naapuri pikselit, joka vähentää kuvan yksityiskohtia. Kaava 1 osoittaa miten lopputulos on laskettu. Uusi kuvamatriisi saadaan käsittelemällä alkuperäisen kuvan jokainen pikseli ytimellä. Saatu arvo voidaan pyöristää yleisten laskentasääntöjen mukaan, mutta monessa ohjelmointikielessä desimaaliluvun muuntaminen kokonaisluvuksi poistaa desimaalit. Lopputulos on sama kuin luvun pyöristäminen alaspäin.

$$\frac{1}{9} * \big((42 * 1) + (43 * 1) + (87 * 1) + (43 * 1) + (83 * 1) + (86 * 1) + (81 * 1) + (85 * 1) + (86 * 1)\big) = 71.666 => 71 \quad (1)$$

Konvoluution kanssa kuvien reunat ovat usein hankalia erikoistapauksia, kun ydin on osittain kuvamatriisin ulkopuolella. Yksi vaihtoehto on jättää kuvan reunat käsittelemättä, jollain ulos saatu kuvamatriisi on pienempi kuin alkuperäinen kuva. Vaihtoehtoisesti voimme luoda puuttuvat arvot tiedossa olevien pikseleiden avulla, jolloin lopputulos ei välttämättä ole täysin halutunlainen, mutta kuva säilyttää alkuperäisen kokonsa.

3.2 Reunantunnistus

Ihmiselle reunantunnistus saattaa vaikuttaa helpolta, mutta tietokoneet näkevät maailman hyvin eri tavalla. Tietokoneet havainnoivat kuvat vain numeroina, jotka esittävät pikseleiden arvoja. Reunan tunnistuksessa käytetään yleensä harmaasävykuvia, jolloin jokaisella pikselillä on yksi arvo mustan (0) ja valkoisen (255) väliltä. Oikeanlaisella ytimellä voidaan vertailla pikseleiden arvoja toisiinsa. [8.]

Roberts Cross

Ensimmäisen reunantunnistusmenetelmän kehitti Lawrence Roberts vuonna 1963. Reunantunnistimelle Roberts vaati seuraavia ominaisuuksia:

1. tuotettujen reunojen pitäisi olla hyvin määriteltyjä

2. taustan tulisi tuottaa mahdollisimman vähän häiriötä

3. reunojen voimakkuuden tulisi vastata mahdollisimman lähelle sitä, mitä ihminen havaitsisi.

Nämä kriteerit mielessä Roberts kehitti seuraavat yhtälöt, jotka laskevat reunan voimakkuuden:

$$y_{i.j} = \sqrt{x_{i,j}} \quad (2)$$

$$z_{i,j} = \sqrt{(y_{i,j} - y_{i+1,j+1})^2 + (y_{i+1,j} - y_{i,j+1})^2} \quad (3)$$

X on alkuperäinen pikseli arvo, i ja j esittävät sijaintia kuvassa ja z on uusi laskettu arvo. Kuvassa 5 on näillä kaavoilla muodostetut kaksi konvoluutiomatriisia, joiden tulokset summataan yhteen. [9.]

$$\begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \qquad \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$$

Kuva 5. Roberts Cross -ytimet.

Roberts cross -menetelmä vertailee kulmikkain olevia pikseleitä keskenään. Mitä suurempi ero pikseleiden arvoilla on, sitä suurempi ulos saatu luku on. Jos kaikilla neljällä pikselillä on sama arvo, lopputulos on nolla ja uusi pikseli on musta. Kuvan käsittely näillä matriiseilla tuottaa valkoiset reunat kohtiin, joissa pikseleiden arvot muuttuvat selkeästi. Kuvassa 7 näkyy reunantunnistus eri menetelmillä.

Sobel-suodin

Sobel-reunantunnistinta käytetään samalla tavalla kuin Roberts cross -menetelmää. Sobel-suodin käyttää kahta 3 x 3 -ydintä, jotka tunnistavat reunoja vertikaalisessa ja horisontaalisessa suunnassa.

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \qquad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Kuva 6. Sobel-suotimen ytimet.

Roberts cross -menetelmään verrattuna Sobel-suodin tuottaa vahvemmat ja selkeämmät reunat. Reunan voimakkuus saadaan laskemalla yhteen kahden ytimen itseisarvot kaavan 4 mukaisesti. Kulman suunta puolestaan voidaan laskea kaavalla 5. [10.]

$$|G| = \sqrt{{G_x}^2 + {G_y}^2} \tag{4}$$

$$\angle G = \arctan\left(\frac{G_y}{G_x}\right) \tag{5}$$

Kuva 7. Eri reunantunnistusmenetelmät [9.].

Isomman ytimen takia Sobel-suodin on laskennallisesti hitaampi kuin Roberts cross, mutta samasta syystä se on vähemmän altis häiriölle. Häiriöksi reunantunnistuksessa lasketaan yksittäisten pikseleiden arvojen heilahtelut, jotka eivät ole oikeita reunoja.

Canny-menetelmä

Canny-menetelmä ei itsessään ole uusi reunantunnistin, vaan se on lisäkäsittelyä vanhalle Sobel-reunantunnistimelle. Canny-menetelmässä käytetään seuraavia vaiheita:

- sumennus
- reunojen voimakkuuden ja suunnan määrittäminen
- reunojen ohennus
- heikkojen reunojen poistaminen
- reunojen jäljitys

Ensimmäinen vaihe on alkuperäisen kuvan sumentaminen tai epäterävöittäminen, jolla poistetaan pieniä yksityiskohtia. Ylimääräistä kohinaa ei haluta reunantunnistimeen luomaan vääriä reunoja. Yleisin käytettävä sumennustapa on Gaussin-suodin, joka käyttää Gaussin käyrän mukaan luotua 5 x 5 -ydintä. [11; 12.]

Toinen vaihe on reunojen etsiminen Sobel-reunantunnistimella. Sobel-menetelmällä selvitetään reunojen voimakkuus ja suunta. Saadut reunat ovat selkeät ja vahvat, mutta paikoin usean pikselin levyiset.

Kolmas vaihe onkin reunojen ohennus non maximum suppression -menetelmällä, joka tarkoittaa, että säilytämme reunoista vain maksimiarvot ja poistamme kaikki muut asettamalla ne mustaksi. Non maximum suppression ottaa huomioon reunojen voimakkuuden sekä kulman. Jos reuna kulkee pystysuunnassa, tutkimme vaakasuunnassa, onko viereiset pikselit voimakkaampia. Tämä muuttaa reunat yhden pikselin levyisiksi. [13.]

Neljäs vaihe poistaa heikot arvot, jotka eivät välttämättä ole reunoja. Tämä tapahtuu kahdella kynnys arvolla. Kaikki reunat, jotka ovat ylärajaa korkeammat lasketaan vahvoiksi reunoiksi ja säilytetään. Kaikki alarajaa heikommat poistetaan automaattisesti, mutta näiden rajojen välissä olevat reunat päätetään seuraavassa vaiheessa.

Viides vaihe päättää, mitkä reunoista säilytetään ja mitkä poistetaan. Päätöksen teko perustuu reunojen jäljittämiseen. Jos heikko reuna kiinnittyy jossain kohtaa vahvaan reunaan, se säilytetään. Reunat, jotka eivät ole kytköksissä vahvoihin reunoihin poistetaan. Viidennen vaiheen reunojen jäljitys on monimutkainen prosessi ja osa Canny-menetelmän toteutuksista ei tätä käytä.

Canny-menetelmä tuottaa selkeät yhden pikselin leveät reunat, joista on poistettu ylimääräinen häiriö ja heikot reunat. [13.]

3.3 Kuvien yhdistäminen

Kahden kuvan yhdistämiseen on monta eri tapaa, riippuen siitä minkälainen efekti kuvilla halutaan luoda. Näitä eri tapoja yhdistää kuvia kutsutaan sekoitustiloiksi. Usein sekoitustiloja käyttäessä kuvien väriarvot normalisoidaan nollan ja yhden välille ja kuvien värikanavat käsitellään erikseen. Kaava 6 on kertova sekoitustila, joka yksinkertaisesti kertoo kuvien a ja b arvot keskenään. [14; 15.]

$$f(a,b) = ab \tag{6}$$

Sekoitustilat yhdistävät kuvien väriarvoja eri suhteissa. Lopputulos voi tummentaa tai vaalentaa kuvaa tai muokata värejä täysin. Sekoitustiloja pystytään tarvittaessa lisäämään vain haluttuihin osiin kuvasta, jolloin muokkaus tapahtuu rajatulla alueella. [16.]

3.4 Kuvan koon muuttaminen

Kuvan skaalaamiseen eli sen koon muuttamiseen löytyy monia tapoja, jotka kaikki vaikuttavat eri tavoin kuvan lopputulokseen. Käydään läpi yleisimmät tavat.

Lähin naapuri -menetelmä

Yksinkertaisin lähin naapuri -menetelmä kopioi vanhasta kuvasta pikselin, joka osuu lähimmäksi uuden pikselin sijaintia. Tämä luo kuvaa suurentaessa ”pikselöityneen” tyylin ja soveltuu parhaiten, kun terävät reunat halutaan säilyttää. [17.]

Bilineaarinen interpolaatio

Lähin naapuri -menetelmää edistyneempi tapa on bilineaarinen interpolaatio, joka arvioi pikseleiden välisiä arvoja lineaarisesti kahdessa ulottuvuudessa. Bilineaarinen interpolaatio luo tasaisempia muutoksia värien välille. [18.]

Bicubic interpolaatio

Bicubic interpolaation uusi ulottuvuus lisää yhtälöön värin derivaatan. Bilineaariseen interpolaatioon verrattuna bicubic interpolaatio antaa parempaa jälkeä, kun useampi väri sekoittuu keskenään. Kuvassa 8 on vertailu eri menetelmien välillä. [19.]

Kuva 8. Skaalausmenetelmien erot kuvaa suurennettaessa.

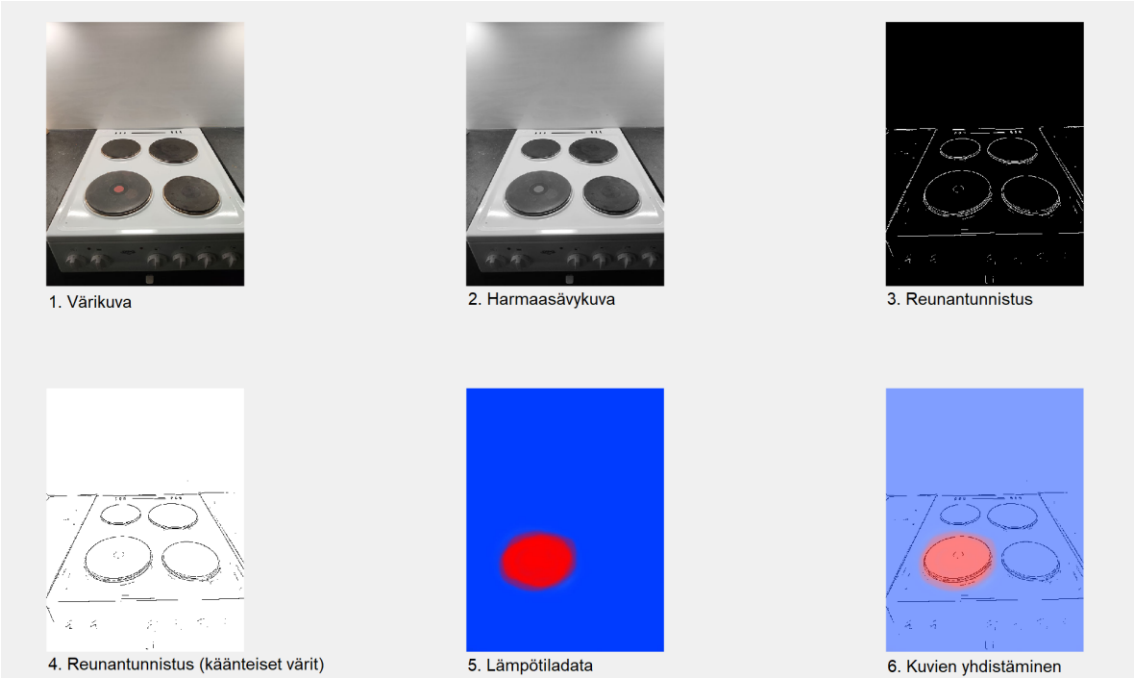
Skaalausmenetelmä kannattaa valita käyttötarkoituksen perusteella. Useimpiin valokuviin bicubic interpolaatio toimii parhaiten, mutta lähin naapuri menetelmää kannatta harkita, jos kuvassa on paljon teräviä reunoja.

4 Lämpökameran rakentaminen

4.1 Suunnittelu

Lämpökameran suunnittelu alkoi 2021 vuoden alussa ja kehittyi paljon kevään aikana. Ideana oli käyttää helposti saatavilla olevaa infrapunakamera moduulia ja lisätä kuvaan tarkkuutta erillisellä värikameralla. Värikameran kuvalle tehtäisiin reunantunnistus Roberts cross tai Sobel-menetelmällä, jonka jälkeen reunat piirrettäisiin lämpötiladatan kanssa näytölle. Näytön reunaan tulisi lämpötilaskaala, joka osoittaa mitä lämpötilaa eri värit tarkoittavat.

Kuvassa 9 näkyy suunnitelma kuvien käsittelystä visualisoituna. Ensimmäinen vaihe on ottaa kuva värikameralla. Kuva muutetaan harmaasävyiseksi ja syötetään reunantunnistimeen, josta halutaan ulos käänteiset värit. Viides vaihe esittää lämpötiladataa, jossa punainen on lämmin ja sininen kylmä. Lopuksi nämä kaksi kuvaa yhdistetään ja lämpötiladata näkyy reunatunnistetun kuvan päällä.



Kuva 9. Suunnitelma lämpökameran kuvasta.

Lämpötilojen esittämiseen neljä väriä voisi olla sopiva määrää. Värejä saadaan riittävästi eri lämpötilojen esittämiseksi, ja koko kuva ei ole yksivärinen.

4.2 Lämpökameran osat

MLX90640-infrapunakameramoduuli

Projektissa käytetty infrapunakamera on Adafruitin kytkentäpiirissä oleva Melexis MLX90640 FIR-sensori. Kyseisessä sensorissa on 24 x 32 -pikseliä eli se kykenee lukemaan 768 lämpötila-arvoa ja pystyy mittaamaan lämpötiloja -40 ºC ja 300 ºC väliltä. Kamerasensorin sisäisessä muistissa on valmistajan valmiiksi luomat kalibrointiarvot, joiden avulla pystytään laskemaan oikeat lämpötilat 2 ºC tarkkuudella 0 ºC ja 100 ºC väliltä. Sensorin valmistaja on jakanut kirjaston, jota käytetään lämpötilojen lukemiseen ja laskemiseen. MLX90640 käyttää I2C-protokollaa ohjaukseen sekä tiedonsiirtoon. [20; 21.]

VC0706-kameramoduuli

VC0706-kamera sisältää CMOS-sensorin sekä kuvankäsittelypiirin, joka pakkaa kuvat automaattisesti JPEG-muotoon. Kamera ohjaa automaattisesti valotusta ja manuaalisesti voimme säätää kuvan kokoa, pakkaussuhdetta. Suurempi JPEG-pakkaussuhde pienentää tiedoston kokoa, mutta yli kolmekymmentä prosenttia alkaa jo kadottaa paljon yksityiskohtia. Kameraa ohjataan UART-protokollaa käyttäen. [22; 23.]

ILI9341-näyttö

ILI9341 on 2,8 tuuman 240 x 320 -pikselin kosketusnäyttö. Kyseisellä näytöllä on oma kuvapuskuri, joka mahdollistaa yksittäisen pikselin asettamisen ilman koko näytän päivittämistä. Näyttö käyttää väreille 16-bittistä RGB565-formaattia, joka antaa punaiselle ja siniselle värille viisi bittiä ja vihreälle värille kuusi bittiä. Näyttö käyttää SPI-protokollaa ohjaukseen ja kuvapuskurin täyttämiseen. [24; 25.]

Teensy 4 -kehitysalusta

Lämpökameran aivoina toimii PJRCn Teensy 4 -kehitysalusta, joka pohjautuu ARM Cortex-M7 mikrokontrolleriin. Teensy 4 -alustan 32-bittinen mikrokontrolleri on kellotettu 450 MHz:n taajuuteen ja pitäisi olla riittävästi laskentatehoa tähän projektiin. Teensy 4:ssä löytyy jopa 40 i/o-pinniä. [26.]

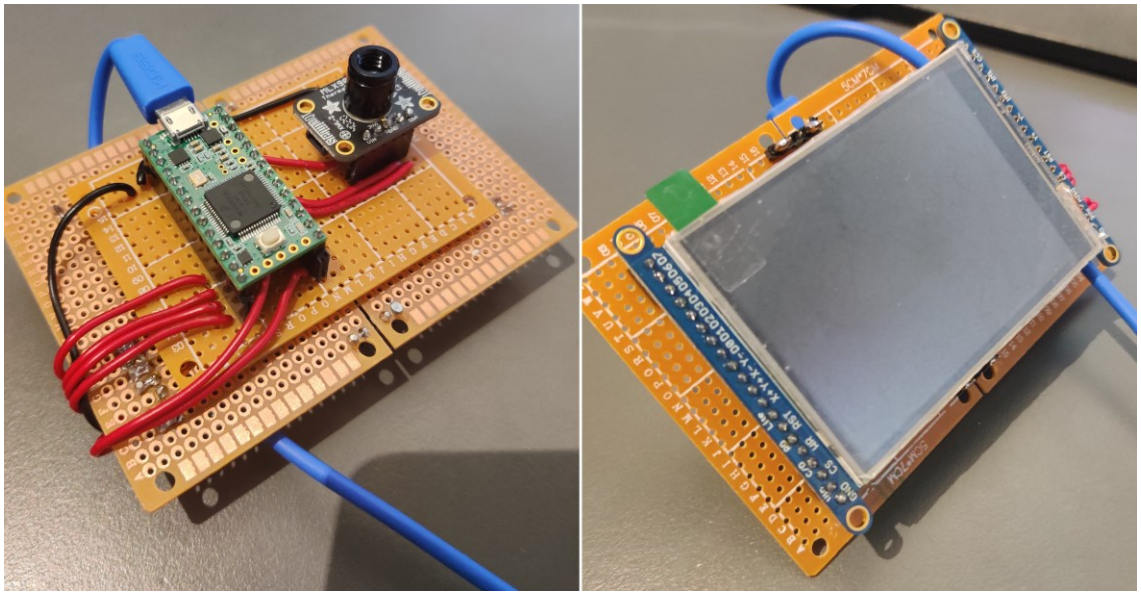
Virtalähde

Kaikki projektissa käytetyt komponentit toimivat 3,3 voltin jännitteellä. Teensy-kehitysalustassa on oma jännitteensäädin, jota voidaan hyödyntää. Itse virtalähteenä toimii tavallinen 9 voltin paristo. Pariston jännite syötetään DC/DC-muuntimeen ja siitä suoraan Teensy-kehitysalustaan sekä muihin komponentteihin.

4.3 Osien yhdistäminen

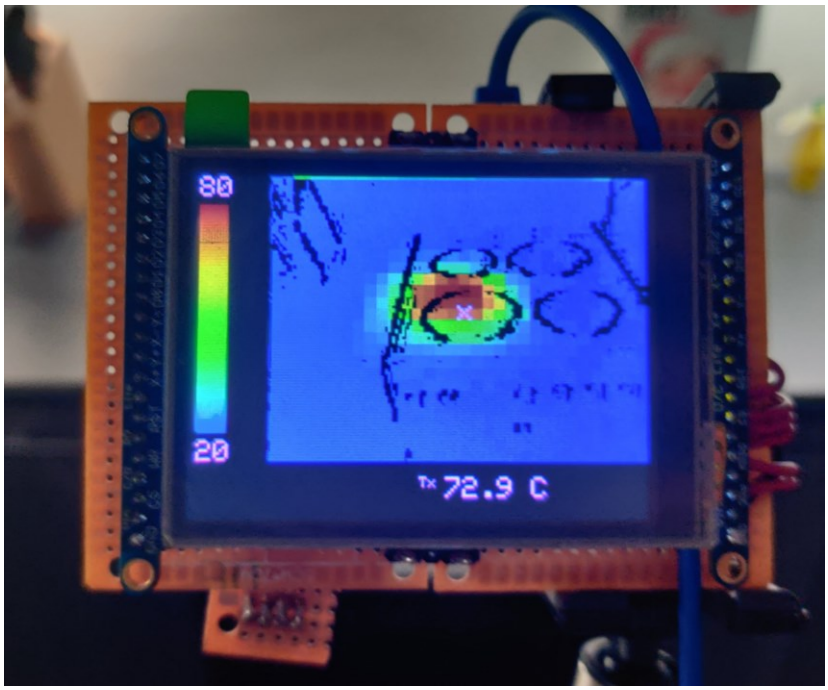
Komponenttien toimivuus testattiin ensiksi yksitellen kytkemällä ne hyppylangoilla kiinni. Komponentteja testatessa kokeiltiin, että niitä pystytään ohjaamaan mikrokontrollerin avulla ja ne tuottavat odotetun ulostulon.

Ensimmäinen raaka versio lämpökamerasta tehtiin kytkentälevyillä ja johdoilla. Kuvassa 10 näkyy kytkentä näytön, mikrokontrollerin ja infrapunakameran kanssa.



Kuva 10. Lämpökameran testikytkentä.

Kuvassa 11 näkyy lämpökameran kokeilua, kun testikytkentä oli saatu tehtyä ja mikrokontrollerin ohjelmointia alettiin hioa kohdilleen. Kuvan tilanne on sama kuin suunnitteluvaiheen kuvassa 9 ks. s.14.



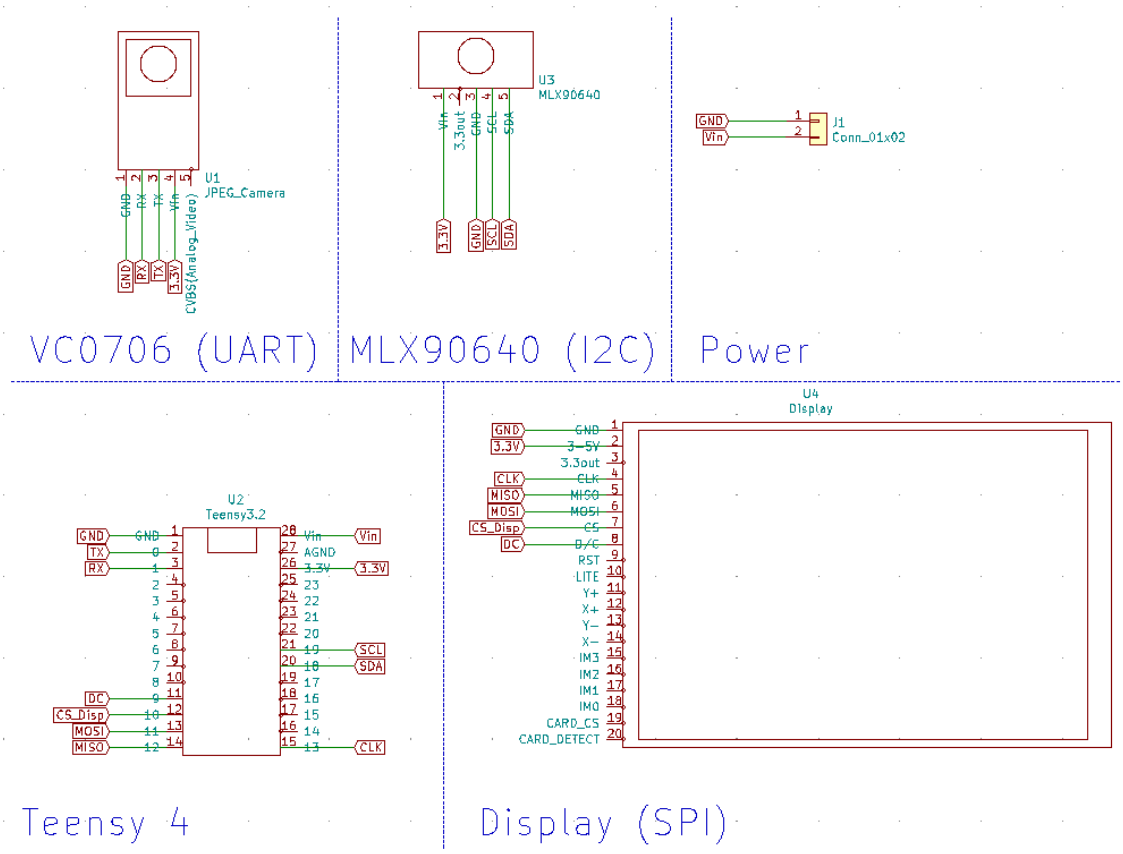
Kuva 11. Lämpökameran kokeilu.

Kaikki osat oli päätetty ja niiden sijoittelu lopulliseen laitteeseen oli suunniteltu. Seuraava vaihe luonnollisesti olisi poistaa kaikki johdot ja korvata ne piirilevyllä.

Piirilevy

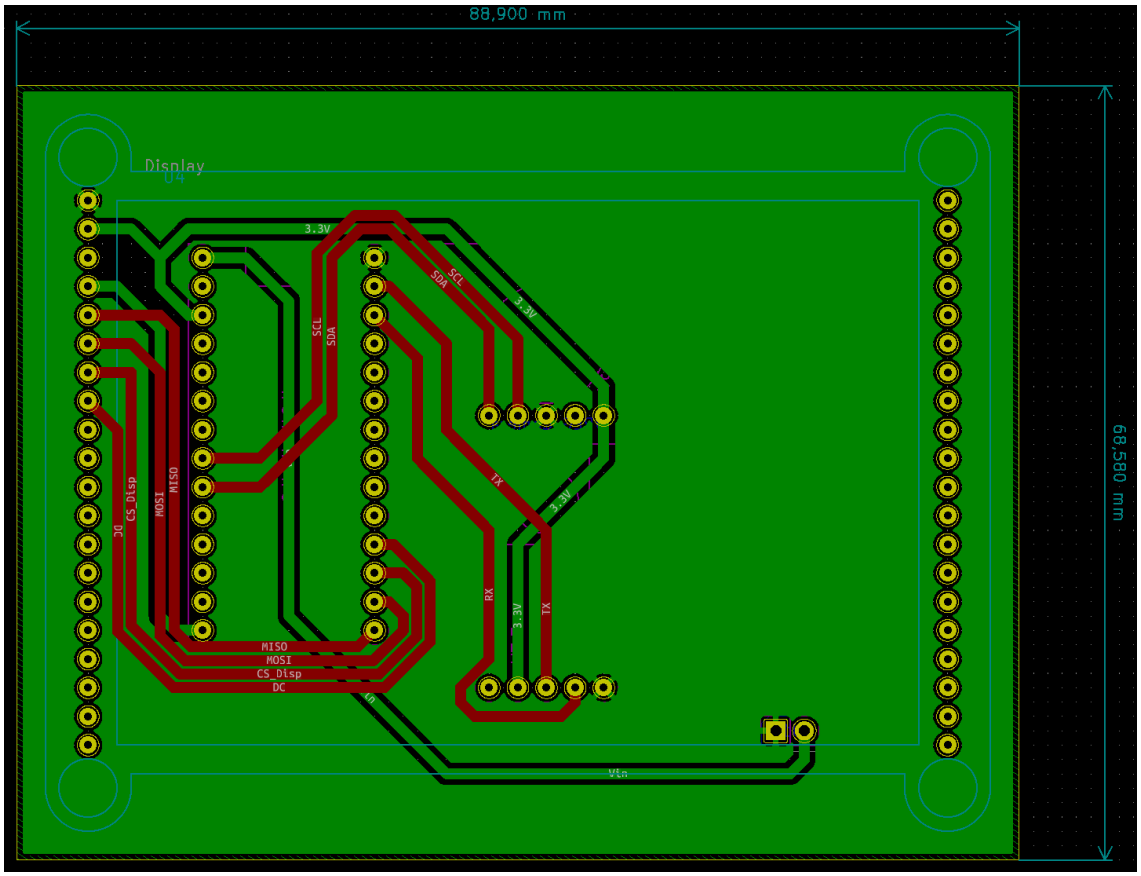
Piirilevyn suunnittelu alkoi heti, kun oli tiedossa, mitä komponentteja projektissa käytettäisiin. Piirilevyn suunnitteluun käytettiin ohjelmaa nimeltä KiCad, joka on avoimen lähdekoodin elektroniikkasuunnittelutyökalu. KiCadissa ei ole valmiita malleja käytössä oleville osille, joten ne täytyi luoda tyhjästä. [27.]

Kytkentäkaaviossa on kaikki lämpökameran osat. VC0706.kameramoduuli on kytketty Teensy-kehitysalustan pinneihin 0 ja 1, jotka vastaavat mikrokontrollerin UART-pinnejä. Teensy 4:n SPI toimii pinneissä 9–13 ja I2C puolestaan pinneissä 18 ja 19. Käytössä on siis vain 9 i/o-pinniä.



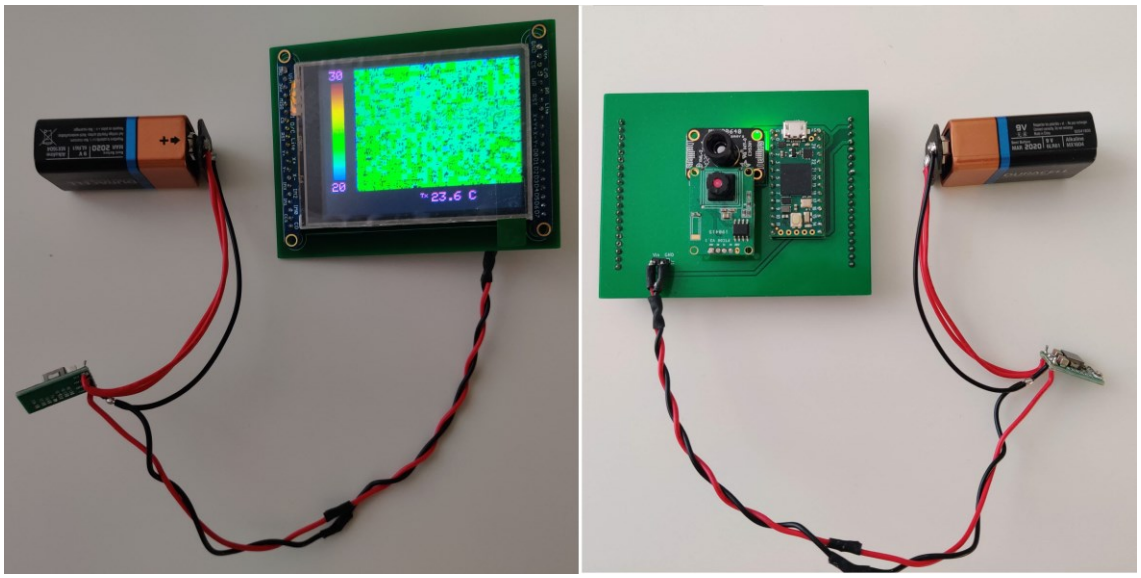
Kuva 12. Kytkentäkaavio.

Piirilevy on kaksipuolinen ja mitoiltaan hieman isompi kuin käytössä oleva näyttö. Kaikki virrat (+5 V sisääntulo, +3,3 V ja maa) kulkevat piirilevyn takapuolella ja muut kytkennät piirilevyn etupuolella. Kupari jälkien leveys on 1 mm.



Kuva 13. Piirilevyn etupuoli.

Komponenttien suhteen tärkeintä oli saada molempien kameroiden linssit mahdollisimman lähekkäin. Näin kameroiden kuvakulmat olisivat samat ja kuvien kohdistaminen olisi helpompaa. Näyttö on kiinnitetty piirilevyn etupuolelle ja kaikki muut komponentit takapuolelle. Kuvassa 14 näkyy, kuinka osat ovat kiinni piirilevyssä.



Kuva 14. Kasattu piirilevy.

Piirilevy in hieman isompi kuin käytössä oleva näyttö. Kaikille osille on hyvin tilaa piirilevyllä. DC/DC-muunnin lisättiin projektiin vasta piirilevyn suunnittelun jälkeen, joten se on toistaiseksi kiinni johdolla. Laitetaan DC/DC-muunnin toistaiseksi kehitysideoiden listaan.

4.4 Ohjelmointi

Mikrokontrolleri on ohjelmoitu C++ -kielellä Arduino ohjelmistoympäristöä käyttäen. Ohjelma vaatii kaksi pääfunktiota, setup ja loop. Setup juostaan kerran laitteen käynnistyessä ja sen sisällä tehdään kaikki vaadittavat alustukset. Eri kommunikaatio protokollat käynnistetään ja oheislaitteet valmistellaan. Esimerkkikoodissa 1 setup kutsuu alifunktiot, jotka alustavat oheislaitteet (näyttö, värikamera ja infrapunakamera). Nämä alifunktiot valmistelevat kommunikaation oheislaitteisiin ja asettavat laitteiden asetukset, kuten kameran resoluution, päivitystaajuuden ja näytön suunnan. [28.]

```
//================================================================
//                          Setup
//================================================================
void setup(){

  ILI9341_init();                              // tft display setup

  VC0706_init();                               // VC0706 setup

  MLX90640_init();                             // MLX90640 setup

} // setup()
```

Esimerkkikoodi 1. Setup-funktio

Setup funktion jälkeen kutsutaan loop-funktio, jota suoritetaan, kunnes laite sammutetaan. Loop-funktio näkyy esimerkkikoodissa 2. Funktio alkaa ottamalla kuvan molemmilla kameramoduuleilla. Kuvat halutaan ottaa ajallisesti mahdollisimman lähekkäin, jotta saamme kuvan samasta tilanteesta. Mitä pidempi aikaväli kuvien ottamisella on, sitä todennäköisemmin kamera ehtii liikkua ja lämpödata ei ole kohdillaan tunnistettujen reunojen kanssa.

Cam.takePicture() käskee VC0706-kameraa ottamaan kuvan. UART-protokollaa käyttäen lähetetään kameramoduulille heksadesimaaliarvot 0x36, 0x01 ja 0x00. ReadImage() kutsuu alifunktion cam.readPicture(), joka lukee kuvan ulos osissa ja tallentaa ne kuvapuskuriin.

VC0706-kamera antaa kuvan ulos pakatussa JPEG-formaatissa, jotta voimme käyttää kuvaa se täytyy ensin purkaa. JPEG-kuva muodostuu MCU-paloista (minimum coded unit), jotka VC0706-kameran tapauksessa ovat 16x8 pikseliä. JPEG-pakkauksen purkamisessa käytetään picojpeg-nimistä työkalua. Picojpeg on suunniteltu JPEG-kuvien purkuun mikrokontrollereilla ja vähäisellä muisti määrällä. JpegDec.decodeArray() kutsuu picojpeg-kirjaston, joka muuntaa MCU-palat yhtenäiseksi kuvamatriisiksi. [29.]

Värikameran kuvalle tehdään reunantunnistus muokatulla Roberts cross -menetelmällä, jossa värit ovat käänteiset. Eli reunat ovat mustat ja muu kuva on valkoinen. Reunatunnistettua kuvaa voidaan näin käyttää maskina. Reunatunnistettuun kuvaan lisätään vielä lämpötiladata väreinä. Värit skaalataan pienimmän ja

suurimman lämpötilan mukaan järjestyksessä sinisestä, vihreään, keltaiseen ja punaiseen.

Lämpötiladata haetaan mlx.getFrame-funktiolla. MLX90640-infrapunakameran kirjasto lähettää tarvittavat komennot ja laskee oikeat lämpötilat kalibrointiarvojen avulla.

```
//================================================================
//                          Loop
//================================================================
void loop() {
  LOOP:

  // Make sure the camera is rolling
  cam.resumeVideo();
  // Take picture
  if(! cam.takePicture()){
    printError("Failed to take picture!");
  }

  else{
    jpglen = cam.frameLength();
    // If image is too large (won't fit into buffer) restart the loop
    if(jpglen > 4000){
      printError("Image file too large!");
      goto LOOP;
    }
  }
  // Read image to buffer
  readImage();

  // Get temp data
  if (mlx.getFrame(frame) != 0){
    printError("No temp data");
    goto LOOP;
  }

  // Decode JPEG
  boolean decoded = JpegDec.decodeArray(buffer, jpglen);

  if(decoded){
    // Transform decoded JPEG MCUs into 565color array
    imageArrayptr = jpeg2array();

    // Edge detection
    edges();

    // Add temperature data
    temperature();

    // Draw output
    drawOutput();

  }
  else printError("Failed to decode image!");

} // loop()
```

Esimerkkikoodi 2. Loop-funktio.

Loop-funktioon on lisätty tarkistuksia, jotka varmistavat, että kaikki vaiheet toimivat oikein ja tulostaa mahdolliset vikakoodit näytölle. Vikakoodit helpottavat

huomattavasti vikojen selvittämistä, kun tietää missä vaiheessa koodia virhe tapahtuu.

Loop-funktiossa komento edges() kutsuu reunantunnistimen. Esimerkkikoodi 3 näyttää, kuinka reunantunnistin on ohjelmoitu. Kaksi for-rakennetta iteroi kuvan läpi. Kuvasta haetaan tarvittavat neljä pikselin arvoa, joilla lasketaan uusi arvo Lawrence Robertsin luoman kaavan avulla. Tätä uutta arvoa verrataan asetettuun raja-arvoon, jolla päätetään, tuleeko pikselistä musta (reuna) vai valkoinen. Lopuksi palautetaan osoitin kuvamatriisiin.

```
//==============================================================
//              Edge detection with Roberts cross
//==============================================================
uint16_t *edges(){

  uint16_t pixels[4];
  for(int y = 0; y < imageHeight-1; y++){
    for(int x = 0; x < imageWidth-1; x++){

      pixels[0] = imageArray[ (y*imageWidth)    + x     ] & 0xFF;
      pixels[1] = imageArray[ (y*imageWidth+1)  + (x+1) ] & 0xFF;
      pixels[2] = imageArray[ (y*imageWidth+1)  + x     ] & 0xFF;
      pixels[3] = imageArray[ (y*imageWidth)    + (x+1) ] & 0xFF;

      uint16_t temp=abs(pixels[0]-pixels[1])+abs(pixels[2]-pixels[3]);

      if(temp < threshold) imageArray[ (y*imageWidth) + x ] = 0xFFFF;

      else imageArray[ (y*imageWidth) + x ] = 0x0000;
    }
  }
  return imageArray;
} // *edges()
```

Esimerkkikoodi 3. Roberts cross -reunantunnistin.

Esimerkkikoodissa 4 lisätään lämpötiladata reunatunnistettuun kuvaan. Lämpötiladataa lisätessä on huomioitava, että infrapunakameran kuva on viisi kertaa pienempi kuin värikameran kuva. Kuvan skaalaamiseen on käytetty lähin naapuri -menetelmää. Reunatunnistettu kuva toimii maskina värien lisäämiselle. Jos reunantunnistimen antama pikseli on valkoinen, väri lisätään. Jos pikseli on musta, se jätetään mustaksi. Tämä on toteutettu ohjelmoinnissa bittitason AND -operaatiolla (imageArray[i] = imageArray[i] & colorGradient(Temp)).

```
uint8_t y = 23;
uint8_t x = 0;

// Loop through the image
for (uint8_t h = 1; h < imageHeight; h++){
  if(h % 5 == 0) y--;

  for (uint8_t w = 1; w < imageWidth; w++){
     if(w % 5 == 0) x++;

     int Temp = frame[y * 32 + x];
     Temp = map(Temp, low, high, 0, 400);
     imageArray[(h-1)*imageWidth+(w-1)]=imageArray[(h-1)*imageWidth+
     (w-1)]&colorGradient(Temp);

  }
  x = 0;
}
```

Esimerkkikoodi 4. Lämpötiladatan lisääminen.

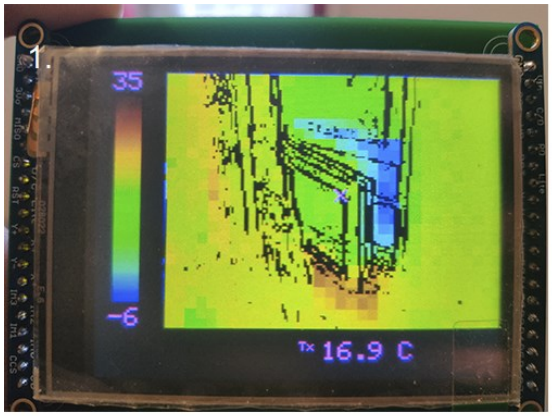
Lämpötilat kartoitetaan 0: ja 400:n välille ja colorGradient-funktio antaa ulos lämpötilaan sopivan värin. Ennen lämpötiladatan lisäämistä imageArray sisältää reunatunnistetun kuvan ja siihen lisätään väriä.

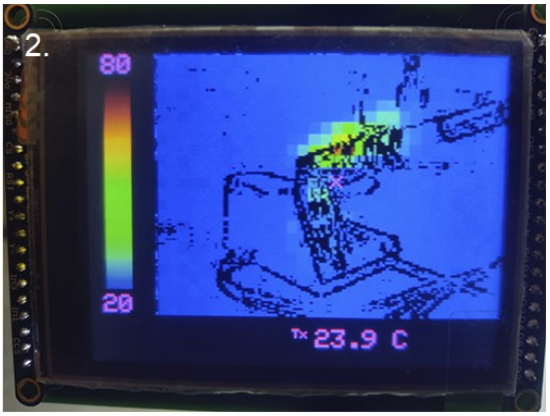
Kun kuvan on kasattu, se piirretään näytölle. Kuva on tässä vaiheessa kooltaan 160 x 120 -pikseliä, mutta se piirretään näytölle 266 x 200 -pikselin kokoisena. Skaalaukseen käytetään lähin naapuri -menetelmää. Kaksi for-rakennetta iteroi halutun kuvakoon verran (266 x 200). Kuvan indeksit kerrotaan kuvaskaalalla, joka on alkuperäisen kuvan korkeus jaettuna halutulla kuva korkeudella. Esimerkkikoodi 5 on kuvan piirtämisestä vastaava funktio.

```
//================================================================
//                 Draw final output to display
//================================================================
#define drawPosX 0
#define drawPosY 40
#define drawSizeH 200
#define drawSizeW 266
#define imgScale ((float)imageHeight / (float)drawSizeH)

void drawOutput(){

  for(uint16_t y = drawPosY; y < drawPosY + drawSizeH; y++){
    for(uint16_t x = drawPosX; x < drawPosX + drawSizeW; x++){

      tft.drawPixel(x, y, imageArrayptr[(int)(imgScale * (y - draw-
PosY)) * imageWidth + (int)((x - drawPosX) * imgScale)]);
    }
  }


  tft.setTextColor(ILI9341_PINK);
  tft.setTextSize(2);

  // Draw x in the middle of the image
  tft.setCursor(drawPosX+(drawSizeW/2)-10, drawPosY+(drawSizeH/2)-5);
  tft.print("x");

  tft.setRotation(3);

  // function call to draw the temperature color scale
  tempScale(15, 20, 20, 160);


  // Temp scale high value
  tft.setCursor(12, 0);
  tft.fillRect(0, 0, 320 - drawSizeW - drawPosX, 20, 0);
  tft.print(high);

  // Temp scale low value
  tft.setCursor(12, 185);
  tft.fillRect(0, 180, 320 - drawSizeW - drawPosX, 20, 0);
  tft.print(low);

  // Print Tx
  tft.setCursor(160, 210);
  tft.setTextSize(1);
  tft.fillRect(160, 200, 160, 40, 0);
  tft.print("Tx ");
  tft.setTextSize(2);
  sprintf(str, "%.1f C", T_x);
  tft.print(str);

  tft.setRotation(1);

}// drawOutput()
```

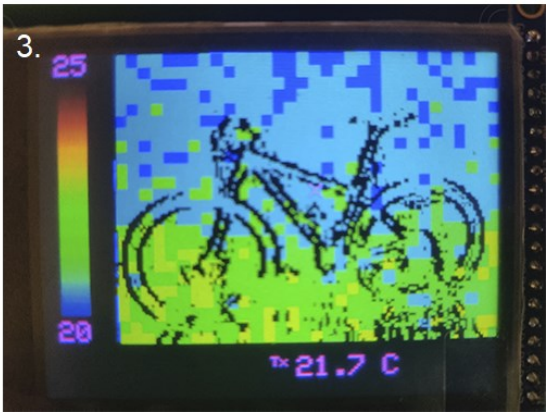
Esimerkkikoodi 5. Kuvan piirtäminen näytölle.

Kun itse lämpökameran kuva on piirretty näytölle, sen ympärille lisätään muuta hyödyllistä dataa. Näytön alareunassa näkyy kuvan keskipisteen lämpötila, jolloin lämpötiloja voidaan lukea osoittamalla kameraa eri kohteisiin. Näytön vasempaan laitaan piirretään palkki, joka näyttää missä järjestyksessä värit menevät kylmästä kuumaan.





1. Avoin ikkuna talvella
2. Kuuma kolvi
3. Pyörä



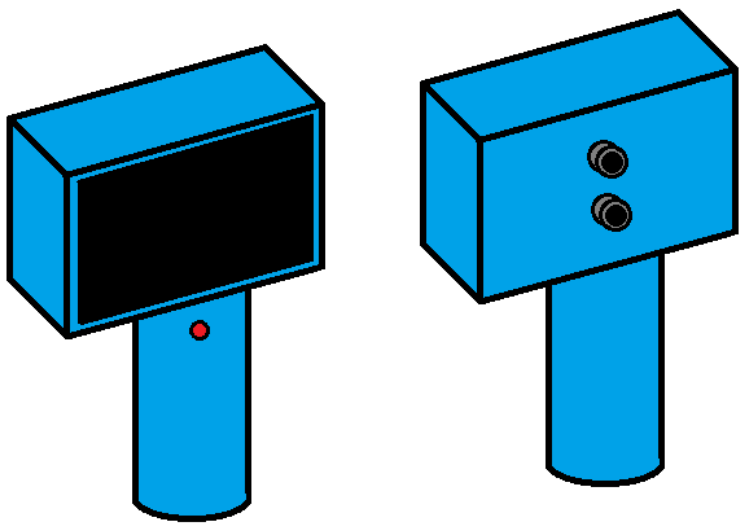
Kuva 15. Lämpökameran testaus.

Kuvassa 15 on lämpökameran testausta eri tilanteissa. Ensimmäisessä tilanteessa näkyy, kuinka kylmää ilmaa tulee avoimesta ikkunasta. Ikkunan alla oleva patteri näkyy selkeästi lämpöisenä. Toisessa tilanteessa reunantunnistus erottaa kolvin telineen selkeästi ja lämpökamera näyttää, kuinka lämpö hohkaa kolvista. Kolmas tilanne on reunantunnistus testi. Pyörän tunnistaa kuvasta hyvin, vaikka ääriviivat katkeilevat paikoittain.

4.5 Kehitysmahdollisuudet

Lämpökamera ei ole vielä valmis. Sitä pystyisi vielä kehittämään pidemmälle, niin ohjelmoinnin kuin fyysisen rakenteen kannalta. Yksi ideoista oli lisätä kuvanotto-painike, joka tallentaa kuvan muistikortille. Kuvan tallennuksen lisääminen ei ole erityisen vaikea toteuttaa. Kuva täytyisi vain muuttaa muotoon, jonka pystytään lukemaan muilla laitteilla. Esimerkiksi bitmap-tiedostomuoto vaatii oikeanlaisen ylätunnisteen, jonka perään lisätään pikseleiden väriarvot järjestyksessä. Koska kuvan koko ei muutu ylätunniste pysyy samana.

Lämpökamera on tällä hetkellä vain piirilevy, johon osat on kiinnitetty. Seuraava vaihe olisi luoda jonkinlainen kotelo 3D-tulostamalla. Kuvassa 16 on luonnos ko-telon ideasta. Kameraan tulisi kahva, jonka sisälle jäisi tilaa paristolle ja kahvaan lisättäisiin painike kuvanottoa varten.



Kuva 16. Luonnos lämpökameran kotelosta.

Piirilevyn suunnittelussa mainittiin myös, että DC/DC-muunnin lisättiin projektiin vasta myöhemmin ja ei tästä syystä ole kiinni piirilevyssä. DC/DC-muuntimelle

on kuitenkin paljon tilaa piirilevyn takapuolella ja sen lisääminen olisi hyvin yksinkertaista.

5 Yhteenveto

Opinnäytetyön aikana tutustuttiin digitaalikameroiden toimintaperiaatteisiin. Opeteltiin, kuinka kamerasensorit havainnoivat maailmaa ja tulkitsevat värejä suodattamalla muut elektromagneettisen säteilyn aallonpituudet pois. Bayer-suodatinta käyttämällä havainnoidaan kolme päväriä. Oikeanlaisia suodattimia käyttämällä kamerat pystyvät havaitsemaan ihmissilmälle näkymättömiä elektromagneettisia säteitä.

Luvussa 3 käytiin läpi, kuinka digitaaliset valokuvat muodostuvat punaisesta, vihreästä ja sinisestä matriisitasosta. Näitä tasoja käsiteltiin erilaisilla suodinmatriiseilla, jotka muokkasivat kuvien ulkonäköä eri tavoin. Konvoluution avulla kuvia pystytään tarkentamaan ja sumentamaan tai niistä voitiin tunnistaa esineiden reunoja.

Lämpökameran suunnittelu alkoi ideasta käyttää edullista infrapunakameraa ja lisätä kuvaan tarkkuutta värikameralla. Värikameran kuvalle tehtiin reunantunnistus, jonka päälle lämpötiladata piirrettiin. Lopullinen kuva oli paljon selkeämpi kuin infrapunakameran kuva yksinään.

Lämpökameralle luotiin osiin sopiva piirilevy, jolla osat kytkettiin yhteen. Piirilevy saa rakennelman näyttämään valmiimmalta, kun johtoja ei roiku joka paikassa. Ohjelmointi sekä tarvittava kuvienkäsittely toteutettiin C++ -kielellä. Reunantunnistus vaikuttaa toimivan hyvin ja kuvista tunnistaa elementtejä selkeästi.

Kaikkea ei pystytty toteuttamaan tämän opinnäytetyön aikana, mutta työ toimii hyvänä pohjana jatko kehitykselle. Lämpökameraa tullaan kehittämään vielä opinnäytetyön ulkopuolella aiemmin mainittujen ideoiden mukaan.

Lähteet

1 What is a CCD? Verkkoaineisto. Spectral Instruments. <https://specinstcameras.com/what-is-a-ccd/>. Luettu 24.11.2021.

2 Imaging Electronics 101: Understanding camera sensors for machine vision applications. Verkkoaineisto. Edmund optics. <https://www.edmundoptics.com/knowledge-center/application-notes/imaging/understanding-camera-sensors-for-machine-vision-applications/>. Luettu 29.9.2021.

3 Difference between CMOS and NMOS technology. Verkkoaineisto. El Pro Cus <https://www.elprocus.com/difference-between-nmos-cmos-technology/>. Luettu 29.9.2021.

4 Bayer filter: What is it and how does it work? 2015. Verkkoaineisto. What digital camera. <https://www.whatdigitalcamera.com/technology_guides/bayer-filter-work-60461>. 8.1.2015. Luettu 24.11.2021.

5 Linear interpolation. 2021. Verkkoaineisto. Wikipedia. <https://en.wikipedia.org/wiki/Linear_interpolation>. Päivitetty 14.9.2021. Luettu 24.11.2021.

6 Infrared. 2021. Verkkoaineisto. Wikipedia. <https://en.wikipedia.org/wiki/Infrared>. Päivitetty 20.9.2021. Luettu 29.9.2021.

7 Convolution. Verkkoaineisto.Gimp. <https://docs.gimp.org/fi/gimp-filter-convolution-matrix.html >. Luettu 29.9.2021.

8 Introduction to edge detection. 2021. Verkkoaineisto. Great Learning <https://www.mygreatlearning.com/blog/introduction-to-edge-detection/>. 16.2.2021. Luettu 29.9.2021.

9 Roberts cross. 2019. Verkkoaineisto. Wikipedia. <https://en.wikipedia.org/wiki/Roberts_cross >. Päivitetty 22.1.2019. Luettu 24.11.2021.

10 Sobel edge detector. Verkkoaineisto. HIPR2. <https://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm >. Luettu 24.11.2021.

11 Canny-menetelmä. Verkkoaineisto. Cmap Tools <https://cmapspublic3.ihmc.us/rid=1HGFTQ1TB-11C16C4-16GN/Cannyn%20menetelm%C3%A4.cmap>. Luettu 24.11.2021.

12 Gaussian filter. 2021. Verkkoaineisto. Wikipedia. <https://en.wikipedia.org/wiki/Canny_edge_detector#Gaussian_filter>. Päivitetty 29.9.2021. Luettu 24.11.2021.

13 Liang, Justin. Canny edge detection. Verkkoaineisto. justin-liang. <http://www.justin-liang.com/tutorials/canny/>. Luettu 29.9.2021.

14 Sekoitustilojen kuvaukset. 2020. Verkkoaineisto. Adobe. <https://helpx.adobe.com/fi/photoshop/using/blending-modes.html >. Päivitetty 23.6.2020. Luettu 29.9.2021.

15 Blend modes. 2021. Verkkoaineisto. Wikipedia. <https://en.wikipedia.org/wiki/Blend_modes>. Päivitetty 1.9.2021. Luettu 16.11.2021.

16 Ramirez, Jesus. 2017. Blending Modes Explained. Verkkoaineisto. Photoshop training channel. <https://photoshoptrainingchannel.com/blending-modes-explained/>. Luettu 3.12.2021.

17 Nearest-neighbor interpolation. 2021. Verkkoaineisto. Wikipedia. <https://en.wikipedia.org/wiki/Nearest-neighbor_interpolation>. Päivitetty 1.9.2021. Luettu 3.12.2021.

18 Bilinear interpolation. 2021. Verkkoaineisto. Wikipedia. <https://en.wikipedia.org/wiki/Bilinear_interpolation>. Päivitetty 31.11.2021. Luettu 3.12.2021.

19 Bicubic interpolation. 2021. Verkkoaineisto. Wikipedia. <https://en.wikipedia.org/wiki/Bicubic_interpolation>. Päivitetty 6.10.2021. Luettu 3.12.2021.

20 MLX90640. Verkkoaineisto. Adafruit. <https://www.adafruit.com/product/4407>. Luettu 15.11.2021.

21 I2C. 2021. Verkkoaineisto. Wikipedia. <https://en.wikipedia.org/wiki/I%C2%B2C>. Päivitetty 5.11.2021. Luettu 15.11.2021.

22 Adafuit VC0706. Verkkoaineisto. Adafruit. <https://www.adafruit.com/product/1386>. Luettu 15.11.2021.

23 UART. 2021. Verkkoaineisto. Wikipedia. <https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter >. Päivitetty 1.11.2021. Luettu 15.11.2021.

24 Adafruit ILI9341. Verkkoaineisto. Adafruit. <https://www.adafruit.com/product/1770>. Luettu 15.11.2021.

25 SPI. 2021. Verkkoaineisto. Wikipedia. <https://en.wikipedia.org/wiki/Serial_Peripheral_Interface>. Päivitetty 10.11.2021. Luettu 15.11.2021.

26 Teensy 4.0 Development board. Verkkoaineisto. PJRC. <https://www.pjrc.com/store/teensy40.html >. Luettu 15.11.2021.

27 KiCad EDA. Verkkoaineisto. KiCad. <https://www.kicad.org/>. Luettu 15.11.2021.

28 Arduino IDE. Verkkoaineisto. Arduino. <https://www.arduino.cc/en/software>. Luettu 24.11.2021.

29 picojpeg. 2010. Verkkoaineisto. GitHub <https://github.com/richgel999/picojpeg>. Päivitetty 23.3.2020. Luettu 29.11.2021.