Niklas Kukkonen

Nopan heiton automatisointi robotilla

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

28.5.2024

Tiivistelmä

Tekijä:	Niklas Kukkonen
Otsikko:	Nopan heiton automatisointi robotilla
Sivumäärä:	58 sivua + 3 liitettä
Aika:	28.5.2024
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Älykkäät IoT-järjestelmät
Ohjaajat:	Lehtori Antti Laiho

Opinnäytetyön tarkoituksena on tutkia ja rakentaa laite, joka heittää 20-tahkoista noppaa käyttäjän lähettäessä signaalin digitaalisesti ja lukee nopan tuloksen kameran avulla ja lähettää nopan tuloksen digitaalisesti takaisin käyttäjälle. Tarkoituksena on rakentaa laite, jonka avulla voidaan luoda täysin randomisoitu nopanheiton tulos, jota voidaan hyödyntää digitaalisissa palveluissa tai peleissä.

Työssä käydään läpi nopalla pelaamisen historiaa ja teoriaa ja sitä, miten se on kehittynyt modernimpaan aikaan teknologian avulla integroituen online-peleihin ja digitaalisiin välineisiin. Tarkastellaan nopan käyttöä pelaamisessa ja miten sen mekaniikkaa hyödynnetään pelaamiskokemuksen rikastuttamisessa ja kehittämisessä.

Pohditaan lyhyesti automaation ja robotiikan kehitystä ja sen mahdollisia riskejä tulevaisuuden kannalta ja miten se tulee mahdollisesti vaikuttamaan ihmisiin. Pohditaan myös robotiikan, automaation ja kuvantunnistuksen merkitystä projektin tavoitteen saavuttamiseksi ja niiden käyttämistä laitteen rakentamisessa.

Työssä esitellään rakennettu laite, joka heittää noppaa käyttäjän signaalista ja lukee nopan lukeman kameran avulla ja palauttaa tuloksen nettisovellukseen. Lopputuloksena saatiin esitettyä kattava kuva nopalla pelaamisen historiasta ja merkityksestä nykyajan peleissä sekä rakennettua tätä merkitystä demonstroiva laite, jonka avulla voidaan luoda todellinen satunnainen luku ja prosessoida se digitaalisesti.

Avainsanat: automaattinen noppa, Python, kuvantunnistus, ohjelmistorobotiikka, Raspberry Pi

---

Abstract

Author:	Niklas Kukkonen
Title:	Automating Dice Rolling with a Robot
Number of Pages:	58 pages + 3 appendices
Date:	28 May 2024
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Smart IoT Systems
Supervisors:	Antti Laiho, Senior Lecturer

The purpose of the final year project was to investigate and construct a device that rolls a twenty-sided die when the user sends a signal digitally and reads the result with a camera. After this the roll result was to be sent digitally back to the user. The aim was to build a device that can generate a completely randomized die roll result, which can be utilized in digital services or games.

The history and theory of dice gaming are discussed in the thesis, along with how dice gaming has evolved in the modern era with the help of technology, integrating dice gaming into online games and digital tools.

The work briefly discusses the development of automation and robotics and their potential risks in the future, and how they might impact humankind. The significance of robotics, automation, and image recognition in achieving the project's goal and their use in building the device are also studied.

The work gives a comprehensive picture of the history and significance of dice gaming in modern games. It also describes the construction of a device demonstrating this significance, generating a genuine random number with a physical die and processing its results into a digital format. In conclusion, a device was built, which rolls the die upon a user's signal and reads the die's result with a camera and returns it to the web application for the user.

Keywords: automatic dice, Python, image processing, process automation, Raspberry Pi

Sisällys

Lyhenteet

AI:	Artificial Intelligence, eli tietokonepohjainen tekoäly.
AR:	Augmented Reality eli lisätty todellisuus on teknologia, joka yhdistää digitaalisen sisällön reaalimaailman näkymään.
CSS:	Cascading Style Sheets on kieli, jota käytetään verkkosivujen ulkoasun määrittämiseen ja tyylien hallintaan
D&D:	Lyhenne pelin nimelle Dungeons & Dragons.
D20:	20-tahkoinen noppa, jota kutsutaan nimellä D20.
DC:	Direct Current. Lyhennetty nimike tasavirralle. Esimerkiksi DC-moottori tarkoittaa tasavirtamoottoria.
GND:	Ground eli maadoitus yhteys sähköpiirissä.
GPIO:	General Purpose Input/Output, tapa jolla mikrokontrollerit ja tietokoneet voivat lähettää ja vastaanottaa digitaalisia signaaleja ulkoisista laitteista.
GPS:	Global Positioning System. Satelliittipaikannusjärjestelmä, joka mahdollistaa tarkkojen sijaintitietojen saamisen maapallolla.
HTML:	HyperText Markup Language on standardoitu kieli, jota käytetään verkkosivujen rakentamiseen ja niiden sisällön kuvaamiseen.
HTTP:	Hypertext Transfer Protocol on protokolla, jota käytetään tietojen siirtämiseen verkkosivujen ja verkkosovellusten välillä Internetissä.

RPA: Robotic Process Automation on teknologia, joka käyttää ohjelmistorobotteja automatisoimaan tehtäviä ja prosesseja, jotka normaalisti vaativat ihmisen interventiota tietojärjestelmissä.

SSL: Secure Sockets Layer on salaustekniikka, jota käytetään luomaan turvallisia ja salattuja yhteyksiä verkkosivustojen ja käyttäjien välillä Internetissä.

TAYS: Tampereen yliopistollinen sairaala.

TSL: Transport Layer Security on salaustekniikka, jota käytetään turvaamaan ja suojaamaan tietojen siirtoa Internetissä, mikä korvaa aiemman SSL-protokollan.

VR: Virtual Reality on teknologia, joka luo immersiivisen kokemuksen simuloimalla todellisuutta virtuaalimaailmassa käyttäjän ympärillä.

1 Johdanto

Pöytäpelien maailmassa nopan heittäminen on tunnistettu keskeiseksi mekaniikaksi, joka tuo peleihin satunnaisuuden ja ennalta-arvaamattomuuden elementin. Perinteinen nopan heittäminen luo pelikokemukseen taktiilisen ulottuvuuden, mikä edistää odotuksen tunnetta ja yhteenkuuluvuutta pelaajien kesken. Kuitenkin, digitaalisten teknologioiden myötä, syntyy mahdollisuus sulauttaa nopan heittämisen analoginen viehätysvoima modernien laskennallisten järjestelmien tehokkuuteen ja monipuolisuuteen.

Tämän projektin pyrkimyksenä on sillata tätä kuilua esittelemällä uudenlaisen ratkaisun – älykkään noppalaitteen. Laitteen ytimenä on mekanismi, joka kykenee simuloimaan kahdenkymmenen tahkolla varustetun nopan heittoa (D20), kameramoduuli, joka tallentaa nopan tuloksen, sekä yhteydet mahdollistaen datan siirron määrätylle palvelimelle. Tämä laitteiston ja ohjelmiston komponenttien yhdistelmä helpottaa nopanheiton tulosten automaattista generointia ja siirtoa, mikä rikastuttaa pelikokemusta samalla säilyttäen sen perinnäisen viehätyksen.

Projektin ensisijainen tavoite on kaksiosainen: kehittää toimiva laitteistoalusta, joka pystyy tarkasti simuloimaan nopanheittoja ja tallentamaan vastaavat tulokset ja luoda saumaton kommunikaatiokanava laitteen ja palvelimen välille. Tämä mahdollistaa nopanheiton datan siirron edelleen käsittelyä ja integrointia varten digitaalisiin sovelluksiin.

Yhteenvetona tämä projekti edustaa perinteisten pöytäpelikäytäntöjen yhteensovittamista tietoteknisten innovaatioiden kanssa, mikä tarjoaa vilauksen interaktiivisten pelikokemusten tulevaisuudesta. Hyödyntämällä teknologian keinoja tavoitteena on varustaa pelaajat monipuolisella ja immersiivisellä työkalulla, joka ylittää fyysisen ja digitaalisen maailman rajat edistäen luovuutta, yhteistyötä ja nautintoa lautapelien ja digitaalisten pelien maailmassa.

2 Pöytäpelien Historiaa

2.1 Perinteiset pöytäpelit

Monet yksilöt eivät ehkä ole tietoisia lautapelien esihistoriallisuudesta, jotka olivat olemassa ennen kirjoitettua kieltä. Lautapelien alkuperä on peräisin ennen kirjallisen viestinnän alkua. Näin ollen lautapelien alkuperän esihistoriallisuus herättää kysymyksen niiden alkuperästä. Pelaamisen varhaisin ilmentymä voidaan jäljittää noppaan, olennaiseen osaan monien nykyaikaisten lautapelien mekaniikkaa. Varhaisimmat todisteet tällaisista pelipaloista on löydetty 5000 vuotta vanhasta Başur Höyükin hautakummusta, Turkista (kuva 1). Samanlaisia paloja on löydetty Syyriasta ja Irakista, mikä osoittaa lautapelien alkuperän olevan Lähi-idästä. [1.]

Kuva 1: Başur Höyükin hautakummusta löydetyt pelaamiseen tarkoitetut palat. [2]

Pöytäpelaaminen on kokenut uuden suosion aallon viime vuosikymmeninä. Tämän suosion ytimessä on Dungeons & Dragons (D&D), 1970-luvun roolipeliklassikko. D&D:n esiintulo merkitsi merkittävää muutosta pöytäpelikulttuurissa, mikä toi mukanaan immersiivisiä tarinankerrontamekaniikoita, hahmon räätälöintiä ja yhteistyöpohjaisen pelidynamiikan. Varhaiset pöytäpelit keskittyivät strategiseen päätöksentekoon

ja pelaajien väliseen vuorovaikutukseen, niiden inspiraatio oli peräisin fantasiakirjallisuudesta ja historiallisista sodista. Dungeons & Dragonsin synty mullisti genren, mikä korosti narratiivista pelikokemusta ja mielikuvituksellista tutkimista fantastisissa maailmoissa. [3.]

Keskeistä pöytäpelaamisen kokemuksessa on yleisesti käytettyjen noppien heittäminen. Nopat toimivat satunnaisuuden tuojina, mikä toi mukaan ennalta arvaamattomuuden elementin pelikokemukseen samalla helpottaen päätöksentekoa ja konfliktien ratkaisua pelissä. Nopan heittäminen pöytäpelaamisessa ei ole pelkkä mekaaninen toimenpide vaan rituaalinen elementti, joka nostaa odotuksen tunnetta, vahvistaa yhteisöllisyyttä pelaajien keskuudessa ja muokkaa kehittyvää tarinaa. [4.]

Nopan heittäminen pöytäpelaamisessa ilmentää todennäköisyysteorian periaatteita, jossa lopputulokset määräytyvät nopan sivuilla olevien numeroiden jakautumisen mukaan. Nopan heittoon liittyvä satunnaisuus tuo mukaan ennalta-arvaamattomuuden elementin ja haastaa pelaajat sopeutumaan strategioihin ja omaksumaan epävarmuuden osana pöytäpelaamisen seikkailua.

Kuvassa 2 nähdään noppasetti, joita käytetään lautapelejä, esimerkiksi Dungeon & Dragonsia pelatessa. Kuvasta nähdään, että noppia on kuutta erilaista, joissa jokaisessa on eri määrä tahkoja eri satunnaistuloksen saamiseksi pelaamiseen. [4.]

Kuva 2: Yleinen noppasetti, mitä käytetään aktiivisesti Dungeons & Dragons -peleissä. [5]

2.2 Teknologiset edistyaskeleet pöytäpeleissä

Pöytäpelien maailmaa on kehitetty ja rikastettu teknologian avulla viimeisen vuosikymmenen aikana. Pöytäpelaamisen suosion kestävyys mahdollisesti riippuu kyseisten digitaalisten elementtien integroimisesta live- ja online-pelaamiseen.

Online-pelaaminen ja mobiilisovellukset

Internetin yleistymisen ja kehittymisen myötä monien asioiden saavutettavuus ja saatavuus on helpottunut. Tämä myös koskee pöytäpelien pelaamista. Monet pöytäpelit voi nykyään löytää verkosta ja mobiililaitteilta internetyhteyden avulla ja monet niistä myös tukevat online-pelaamista, jolloin voi pelata muiden ihmisten kanssa ilman live-tapaamista. Kuvassa 3 nähdään, kuinka Monopolin online-versioon on implementoitu digitaaliset nopat ylläpitämään pelin kulun satunnaisuutta.

Kuva 3: Suosittu pöytäpeliklassikko Monopoli on helposti pelattavissa online-versiona. [6]

Lisätty todellisuus eli Augmented Reality (AR)

Augmented Reality -sovellukset mahdollistavat pöytäpelien pelaamisen fyysisessä ympäristössä, samalla lisäten siihen digitaalisia elementtejä. AR-ympäristön luomisen perustana ovat tietokonetekniikka, esineiden tunnistus ja GPS-tietojen hyödyntäminen pelaajan todellisen maailman havaintojen ja kokemuksen rikastuttamiseen. Edistyneet anturit ja kamerat eri laitteissa taltioivat käyttäjän ympäristön, mikä mahdollistaa lisätyn todellisuuden tarkan kartoittamisen ja virtuaalisten elementtien integroimisen fyysiseen tilaan. Esimerkki AR:n käytöstä on nähtävillä kuvassa 4, jossa pelaajat näkevät hologrammit peliin kuuluvista hahmoista käyttämiensä AR-lasien avulla. [7.]

Kuva 4: Kuvassa pelaajat pelaavat The Tilt Five -peliä, jonka pelaamisessa käytetään AR-laseja, pelilautaa ja pelisauvaa. [8]

Tekoäly (Artificial Intelligence eli AI)

Tekoälyn kehitys ja sen implementointi lautapeleihin on suuri hyppy moderneissa lautapeleissä ja niiden monimuotoisuudessa. Tekoäly ei vain ole tietokoneen sisäinen vastapelaaja, vaan myös monimuotoinen työkalu, joka aktiivisesti muuttaa pelilaudan ulkonäköä ja maailmaa pelaajalle.

Yksi tekoälyn näkyvimmistä rooleista on antaa haasteita pelaajille, mikä muuttaa tekoälyn käyttäytymistä pelaajan oman käytöksen mukaan. Se adaptoituu ja muuttaa strategioita, mikä mahdollistaa dynaamisen ja kehittyvän haasteen pelaajalle. Tekoälyn käyttäminen on hyödyllistä yksinpelaajille taitojen kehittämisessä tai uusien strategioiden oppimisessa, sillä se voi omaksua erilaisia pelityylejä aggressiivisista puolustaviin, simuloiden ihmisen kaltaista arvaamattomuutta. [9.]

Virtuaalitodellisuus (VR)

Virtuaaliteknologia on tuonut uuden ulottuvuuden pöytäpeleihin, mikä mahdollistaa immersiivisen kokemuksen. VR-teknologia hyödyntää tietokonemallinnusta ja simulointia, mikä mahdollistaa henkilön vuorovaikutuksen keinotekoisen kolmiulotteisen ympäristön kanssa. VR-sovellukset upottavat käyttäjän tietokoneella luotuun ympäristöön, joka simuloi todellisuutta interaktiivisten laitteiden avulla. [10.]

Tämän teknologian kanssa on kehitetty monipuolisia digitaalisia versioita klassisista pöytäpeleistä, joissa pelaaja voi virtuaalisesti kommunikoida muiden pelaajien kanssa ja pelata pöytäpelejä immersiivisesti (kuva 5).

Kuva 5: All On Board! -VR-peli, jossa pelaajat voivat pelata yhdessä erilaisia lautapelejä tai luoda omia pelejä. [11]

Lautapelien ja niiden pelaaminen on kehittynyt huimasti, kun katsotaan, miten niiden historia tuhansia vuosia sitten lähti pienistä erimuotoisista pelinappuloista ja on nykypäivänä kehittynyt pitkälle virtuaaliseen todellisuuteen, jossa pelaajat voivat pelata eri maissa asuevien pelaajien kanssa.

3 Teoreettinen viitekehys

3.1 Nopan käyttö pelaamisessa

Nopan mekaniikka

Nopan mekaniikka

Nopan heittämisen mekaniikka on keskeinen osa monia lautapelejä, ja sen käytön tulos vaikuttaa pelin kulkuun ja satunnaisuuteen. Nopan käyttöön sisältyy monia periaatteita: fyysinen liike, satunnaisuus, todennäköisyys, pelimekaniikan vaikutus ja taktinen elementti.

Fyysinen liike: Useimmissa perinteisissä lautapeleissä pelaaja heittää fyysisen nopan pelilaudalle. Tämä prosessi sisältää fyysisen voiman käytön ja nopan liikuttamisen pelilaudalla. Fyysisen liikkeen ansiosta nopan lopullinen asento ja silmäluku määräytyvät satunnaisesti.

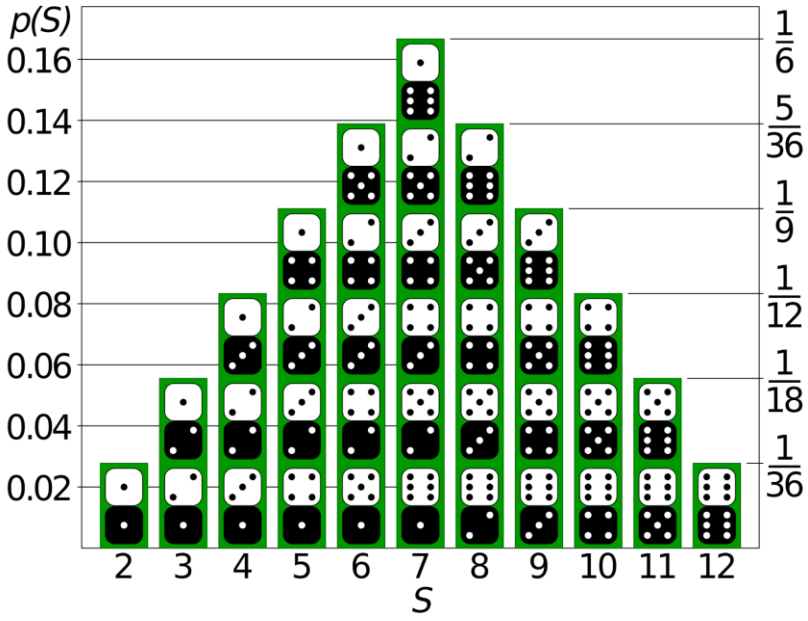
Satunnaisuus: Nopan heittämisessä keskeinen elementti on satunnaisuus. Nopan lopullinen asento ja siten silmäluvut määräytyvät monien tekijöiden, kuten heittäjän voiman, nopan muodon ja pinnan ominaisuuksien sekä ympäristön vaikutuksesta. Tämä luo jännitystä ja ennakoimattomuutta peliin.

Pelimekaniikan vaikutus: Nopan heittämisen mekaniikka voi vaihdella pelistä toiseen ja sillä voi olla erilainen vaikutus pelin kulkuun. Joissakin peleissä nopan heittäminen voi määrittää pelaajan liikkeen pelilaudalla, kun taas toisissa se voi vaikuttaa pelaajan saamiin pisteisiin tai etuihin.

Taktinen elementti: Vaikka nopan heittämiseen liittyy suuri satunnaisuus, osa pelistrategiasta voi liittyä siihen, miten pelaaja hallitsee heittämistään.

Esimerkiksi jotkut pelaajat saattavat yrittää tietoisesti muuttaa heiton voimakkuutta tai kulmaa saadakseen haluamansa tuloksen.
Nämä periaatteet yhdessä luovat nopan heittämisen mekaniikan, joka on olennainen osa monien lautapelien dynamiikkaa ja vaikuttaa pelaajien päätöksiin ja pelin lopputulokseen.

Todennäköisyys: Vaikka nopan heittämiseen liittyy satunnaisuutta, jotkin silmäluvut ovat todennäköisempiä kuin toiset. Esimerkiksi kuusisilmäisen nopan heittämisessä jokainen silmäluku on teoreettisesti yhtä todennäköinen, mutta kahta noppaa käytettäessä tietyt silmälukuyhdistelmät, kuten 7, ovat yleisempiä kuin toiset, kuten 2 tai 12 (kuva 6).

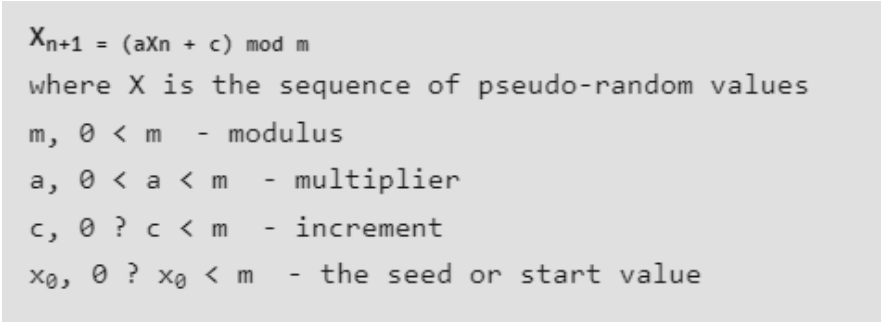


Kuva 6: Kahdella nopalla saatavien tulosten todennäköisyys. [12]

Nopan käyttö tietokone- ja online-peleissä

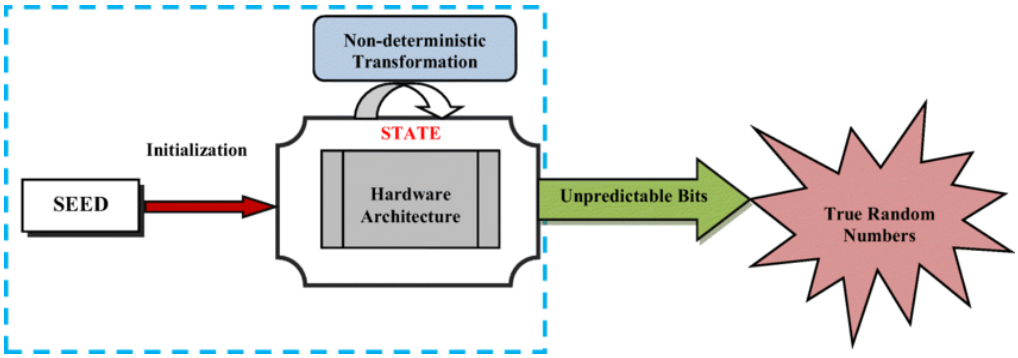
Pelien kehittyessä teknologisemmiksi ja siirtyessä traditionaalisista pöytäpeleistä online-peleiksi, myös niissä käytetty nopan mekaniikka pitää saada siirrettyä uuteen alustaan. Nopan toiminnallisuutta on opittu simuloimaan teknologisesti monella eri tavalla.

Pseudosatunnaislukugeneraattori (Pseudo Random Number Generator) Yleisimpiä satunnaislukugeneraattoreita ovat pseudosatunnaislukugeneraattorit ohjelmistoissa. Niiden tuotokset eivät ole todellisia satunnaislukuja. Sen sijaan ne luottavat algoritmeihin matkiakseen arvon valintaa todellisen satunnaisuuden mukaan. Pseudosatunnaislukugeneraattorit toimivat käyttäjän asettaessa alueen, josta satunnaisluku valitaan (kuva 7).



Kuva 7: Lineaarinen kongruenssigeneraattori on yleisin algoritmi pseudosatunnaisten numeroiden generointiin. Generointi määrittyy rekursiivisesti kuvan osoittamalla suhteella. [13]

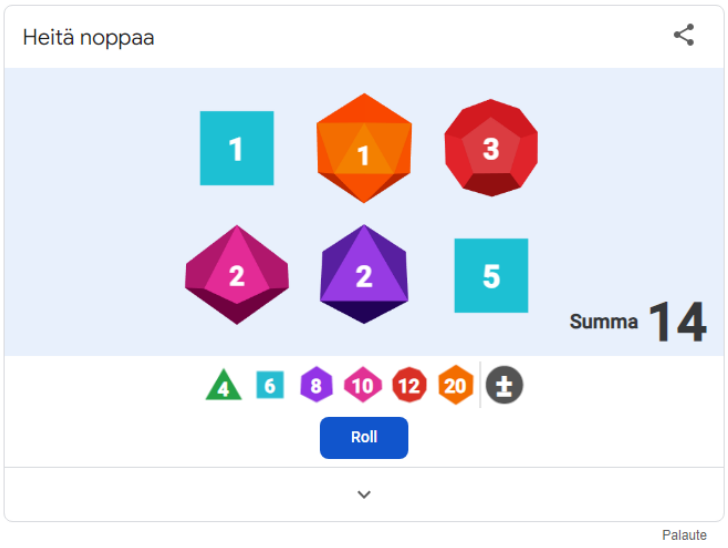
Todellinen satunnaislukugeneraattori (True Random Number Generator) Todellinen satunnaislukugeneraattori, laitteistopohjainen satunnaislukugeneraattori, on kryptografisesti turvallinen ja ottaa huomioon fyysiset ominaisuudet kuten ilmakehän tai lämpötilan olosuhteet. Satunnaisuus tulee ilmakehän kohinasta, joka moniin tarkoituksiin on parempi kuin tyypillisesti tietokoneohjelmissa käytetyt pseudosatunnaiset numerogenerointialgoritmit (kuva 8). [14.]



Kuva 8: Todellinen satunnaislukugeneraattorin diagrammi. [15]

Virtuaalinen noppa

Virtuaalinen noppa on graafisesti esitetty digitaalinen noppa, joka simuloi nopan heittämistä näytöllä. Pelaaja pystyy suoraan näkemään tuloksen luvun. Jotkut virtuaaliset nopat tarjoavat myös mahdollisuuden muokata nopan ulkonäköä, efektejä tai esimerkiksi nopan sivujen määrää eri tarkoituksiin (kuva 9). Koodillisesti virtuaaliset nopat käyttävät pseudo-satunnaislukugeneraattorialgoritmia. [16.]



Kuva 9: Googlen virtuaalinoppia voi käyttää esimerkiksi googlaamalla ‘virtual dice’. [17]

3.2 Automaatio ja robotiikka

Robotiikan määritelmä ja kehitys

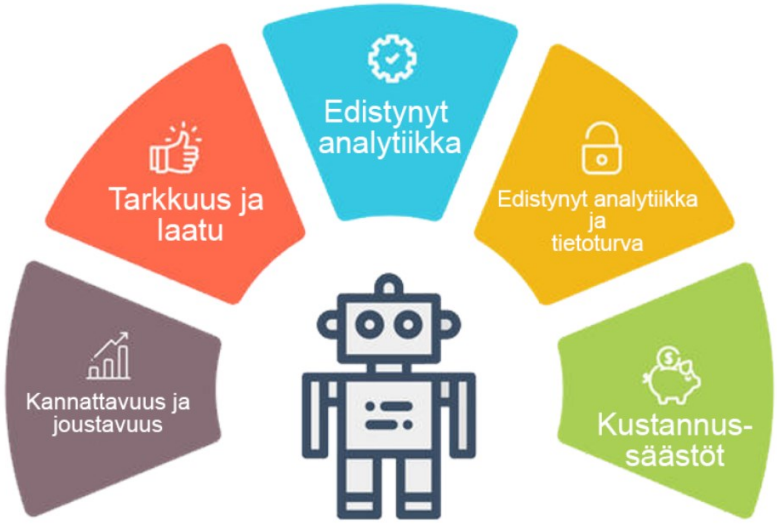
Vaikka robotiikka ja robotit ovat olleet olemassa pitkään, niiden määrittely ei ole yksiselitteistä. Yleisesti ottaen robotit viittaavat tietokoneohjattuihin laitteisiin tai työkaluihin, jotka suorittavat monenlaisia tehtäviä ja ovat monikäyttöisiä. [18.]

Robotiikan kehityksen alkua havaitaan jo Aristoteleen pohdinnoista 320 eKr. Hän pohtii työkalujen kykyä tehdä työtä itsenäisesti ja sen mahdollista merkitystä. Seuraava askel kohti nykyaikaista robotiikkaa tapahtuu 1750-luvulla, kun Jacques de Vaucanson kehittää huilua soittavan robotin. Vuonna 1913 Henry Ford ottaa käyttöön ensimmäisen liukuhihnan autonvalmistuksessa. Teollisuusrobotiikan merkkipaaluksi nousee Unimate, joka otetaan käyttöön General Motorsin tuotantolinjalla vuonna 1961. Vuonna 1970 julkaistaan ensimmäinen ihmisen kaltainen robotti Japanissa. Suomessa ensimmäinen lypsyrobotti otetaan käyttöön vuonna 2000, ja vuonna 2008 TAYSissa (Tampereen yliopistollinen sairaala) otetaan käyttöön Suomen ensimmäinen leukkaussalirobotti. [18.]

Robotiikalla on pitkä historia, ja sen merkitys ja monipuolisuus ovat kehittyneet pitkälle nykymaailman tahdissa. Sen avulla on pystytty oppimaan ja tutkimaan syvemmin ja monipuolisemmin monia asioita. Robotiikka voidaan yksinkertaisimmillaan määritellä järjestelmäksi, joka koostuu laitteista ja osaamisesta, jotka mahdollistavat yhteistyön robotin kanssa. Lisäksi ohjelmistorobotiikka muodostaa oman osa-alueensa, joka eroaa fyysisestä robotiikasta. [18.]

Ohjelmistorobotiikka (Robotic Process Automation, RPA) on teknologia, joka hyödyntää etukäteen määriteltyä bisneslogiikkaa, sääntöjä ja strukturoitua dataa liiketoimintaprosessien automatisointiin. Ohjelmistorobotit poimivat tietoa ja tulkitsevat ohjelmistoista dataa transaktioiden prosessoimiseksi, datan käsittelemiseksi, vastausten generoimiseksi sekä toisten järjestelmien kanssa keskustelemiseksi.

Ohjelmistorobotiikka on ihanteellinen teknologia moniin tietotyön rutiinitehtäviin. Sen avulla voidaan erityisen hyvin automatisoida toistuvat, sääntöpohjaiset ja ison volyymin tehtävät. Kuvassa 10 nähdään ohjelmistorobotiikan monet eri hyödyt. [20.]



Kuva 10: Ohjelmistorobotiikan hyötyjä havainnollistettuna. [19]

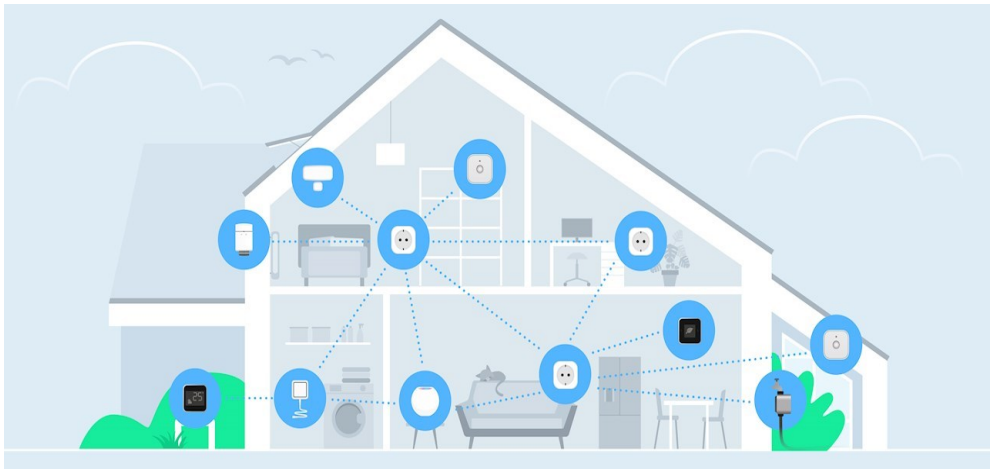
Automaation periaatteet ja merkitys

Automaatio viittaa ohjelmoituun, itsenäisesti toimivaan laitteeseen tai järjestelmään. Yksinkertaisimmillaan automaatio ilmenee esimerkiksi hissin toiminnassa tai ovien automaattisessa avautumisessa ihmiselle. Teollisuudessa automaatio tarkoittaa tietokoneilla ohjaamista ja koneiden ja erilaisten tuotantoprosessien hallitsemista niiden avulla. Automaation käyttö tarjoaa monia etuja, kuten nopeuden, helpon ja yhdenmukaisen toistettavuuden, laadunhallinnan parantamisen, jätemäärän vähenemisen ja työvoiman tarpeen pienenemisen. Nämä hyödyt tekevät automatisoinnista suositun ratkaisun, mikä usein johtaa tuotannon kasvuun.

Automaatio voidaan jakaa useisiin eri muotoihin, kuten teollisuusrobotiikkaan ja testausautomaatioon. Teollisuusrobotiikassa käytetään tietokoneohjattuja

yleiskäyttöisiä koneita, jotka käsittelevät työvälineitä ja muita kappaleita. Ohjelmistot ovat tyypillisesti helposti muokattavissa, ja koneet ovat monikäyttöisiä eri tarkoituksiin.

Testausautomaatiossa tietokone ohjelmoidaan matkimaan ihmisen toimintaa ja sen avulla vältetään manuaalisten testien virhemahdollisuudet. Automaatiojärjestelmissä on yleensä valvomo, jossa on PC-laitteisto ja mahdollisuus valvojille monitoroida järjestelmää. Kuvassa 11 nähdään esimerkki normaaliin kotiin rakennetusta älykkäästä automaatiojärjestelmästä, mikä koostuu monesta eri laitteesta, jotka ovat yhteydessä toisiinsa. [21.]

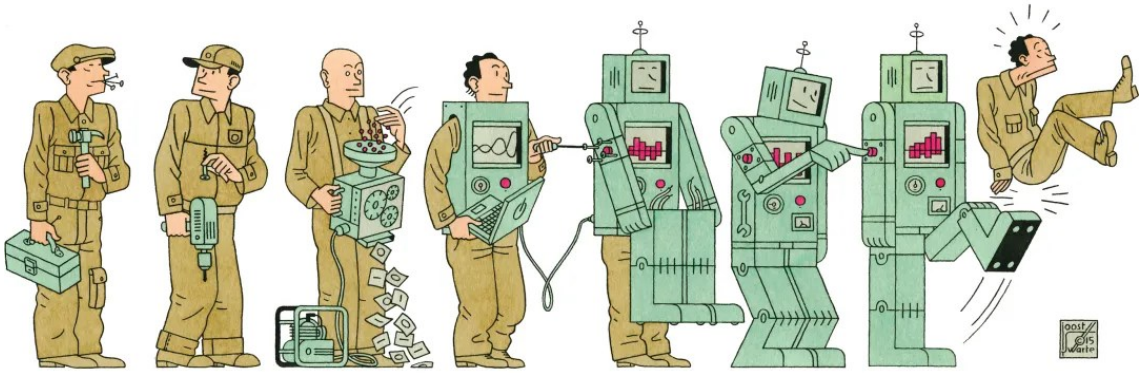


Kuva 11: Kuvassa esimerkkinä älykoti-automaatiojärjestelmä. [22]d

Robotiikka ja automaatio teknologisessa kehityksessä

Robotiikan ja automaation kehitys ilmenee niiden kasvavana globaalina markkina-arvona. Nykyään käytämme monia automatisoituja palveluita, ja robotiikkaa kehitetään jatkuvasti ihmisiä palveleviin tehtäviin. Suurimpia hyötyjä on nähty esimerkiksi lääketieteessä, missä robotiikka mahdollistaa vaarallisten toimenpiteiden suorittamisen mikrokirurgisesti. Esimerkkinä vuonna 2019 kiinalainen kirurgi Tian Wei suoritti tarkkuutta vaativan leikkauksen hyödyntäen kirurgirobottia ja 5G-yhteyttä. [23.]

Teknologian kehitys ei kuitenkaan etene ilman haasteita. Robotiikan ja automaation kehitys herättää huolta tulevaisuuden työllistymisen mahdollisuuksista ja ihmisten tarpeellisuudesta työelämässä. On kuitenkin huomattava, että robottien määrän kasvaessa työpaikkojen määrä ei ole merkittävästi vähentynyt, vaan työn laatu ja tehtävät ovat kehittyneet vastaamaan uusia osaamistarpeita. Tulevaisuudessa robotteja saatetaan käyttää yhä enemmän teollisuudessa, palveluissa ja vapaa-ajan aktiviteeteissa, mutta niiden toiminnan valvomiseen ja huoltamiseen tarvitaan edelleen ihmisiä. Ei ole siis uskottavaa, että robotit korvaisivat ihmiset teollisessa työssä, niin kuin kuvasta 12 voisi ymmärtää. [23.]



Kuva 12: Humoristinen kuvitus robottien “vallankaappauksesta”. [24]

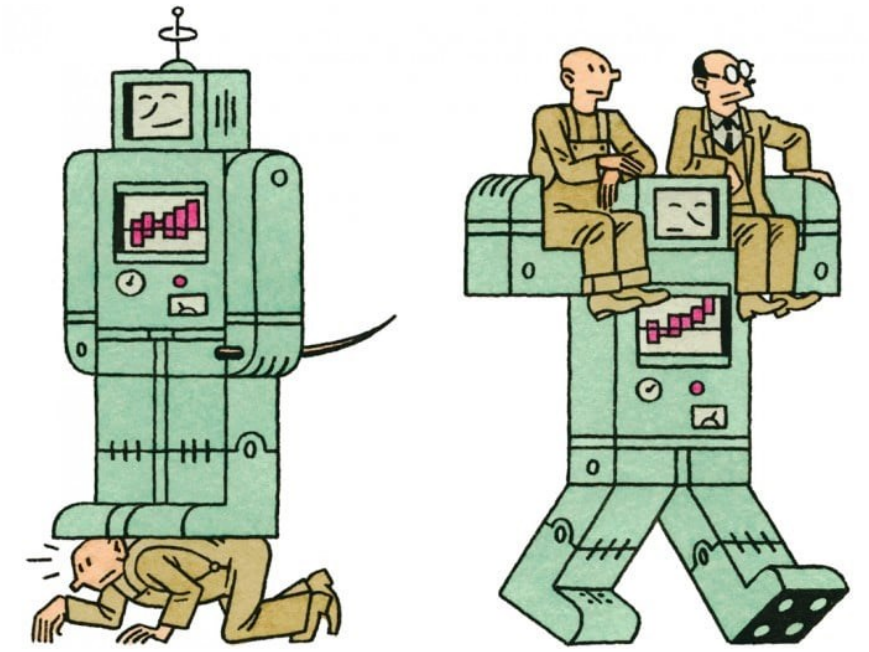
Tulevaisuuden mahdollisuudet ja haasteet

Tekoälyn ja koneoppimisen nopea kehitys muuttaa merkittävästi työmarkkinoita tulevaisuudessa. Monet ennusteet viittaavat siihen, että automaatio ja robotiikka korvaavat monia nykyisiä työtehtäviä, erityisesti rutiininomaisia ja matalan taitotason töitä. Tämä voi aiheuttaa työttömyyttä tietyillä aloilla ja edellyttää uudelleenkoulutusta työvoimalle vastaamaan uusia teknologisia vaatimuksia. Toisaalta automaatio voi myös luoda uusia työpaikkoja, erityisesti teknologian kehittämisen, ylläpidon ja valvonnan aloilla.

Automatisaation laajentuessa esiin nousee myös eettisiä ja sosiaalisia kysymyksiä. Miten esimerkiksi varmistetaan, että tekoälypohjaiset järjestelmät tekevät oikeudenmukaisia päätöksiä ja välttävät syrjintää? Miten käsitellään

työntekijöiden mahdollista työpaikkojen menetystä automatisaation seurauksena? Lisäksi, miten turvataan ihmisten ja robotiikan välinen turvallisuus ja yhteistyö erilaisissa ympäristöissä, kuten tehtaissa tai sairaaloissa?

Tutkimukset ja ennusteet tarjoavat erilaisia näkemyksiä siitä, miten robotiikka ja automaatio muokkaavat tulevaisuuden työelämää ja yhteiskuntaa. On tärkeää, että nämä kehityskulut otetaan huomioon päätöksenteossa ja politiikan muotoilussa, jotta voidaan varmistaa, että teknologian kehitys tukee ihmisten hyvinvointia ja yhteiskunnan kestävää kehitystä. Kuvassa 13 tulkitaan robotiikan ja automaation kehittymisen johdosta syntyviä huonoja ja hyviä tulevaisuuden mahdollisuuksia.

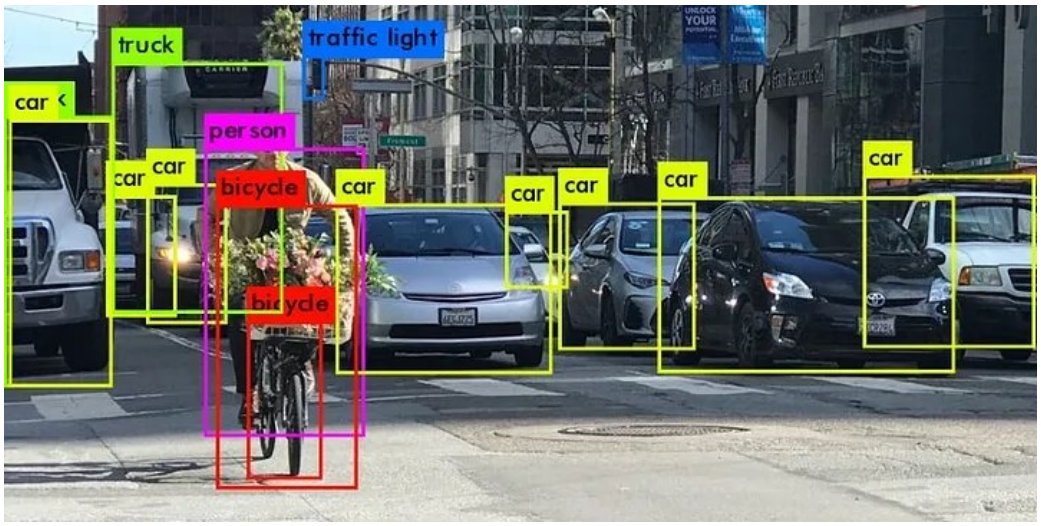


Kuva 13: Humoristinen kuvitus tulevaisuuden eri mahdollisuuksista robotiikan ja automaation kehityksestä. [24]

3.3 Kuvantunnistus ja kameroiden rooli

Kuvantunnistuksen perusteet

Kuvantunnistus viittaa yleisiin metodeihin ja tekniikoihin tietokonenäössä, joiden tarkoituksena on tunnistaa ja nimetä kuvassa esiintyviä asioita perustuen objektien ja asioiden tunnettuihin ominaisuuksiin. Se on katsottu kaikkein haastavimmaksi tietokoneella suoritettavaksi visuaaliseksi tehtäväksi. Kuvantunnistuksen tehtävänä on esimerkiksi tunnistaa paikkoja, ihmisiä, objekteja ja muita tunnistettavia elementtejä. Se on osa tietokonenäön päättelyä, ja siihen sisältyy muun muassa kuvien luokittelu tunnistettujen ominaisuuksien perusteella, tunnistettujen ominaisuuksien paikallistaminen ja rajaaminen, hahmojen ääriviivojen rajaus (ns. maskin luominen), hahmon seuranta ja kasvojentunnistus. Hahmoilla tarkoitetaan tässä yhteydessä kaikenlaisia havaittavia objekteja ja kokonaisuuksia. Koska kuvat voivat sisältää monimutkaisia tilanteita ja maisemia, kuvantunnistusjärjestelmiä opetetaan tunnistusongelmaa vastaavilla merkityillä tietoaineistoilla (dataseteillä). Opettamisen tuloksena algoritmit kykenevät ennustamaan eli tunnistamaan ja luokittelemaan kuvassa esiintyvää tietoa (kuva 14). [25.]



Kuva 14: Kuvantunnistusjärjestelmä tunnistaa ja erottelee eri hahmot ja esineet kuvassa ja merkitsee ne. [26]

Kuvantunnistuksen opettamisessa käytetään nykyään koneoppimistekniikoita ja syväoppimista, joissa hyödynnetään erityisesti neuroverkkoja (Convolutional Neural Networks eli CNNs). Neuroverkko on tietokonejärjestelmä, joka pyrkii jäljittelemään ihmisen aivojen toimintaa. Neuroverkossa yksittäiset "neuronit" vastaavat ihmisen aivojen hermosoluja, mutta ne ovat matemaattisia funktioita. Neuroverkko koostuu syötteestä, parametreistä ja tulosteesta. [25.]

Kameroiden käyttö tietokonenäössä

Kamerat ovat olennainen osa tietokonenäön järjestelmiä, ja ne toimivat syötteiden antureina tietokoneprosessoinnille. Kameran valinta on keskeistä järjestelmän menestyksekkäälle käyttöönotolle, sillä se vaikuttaa suoraan järjestelmän suorituskykyyn ja luotettavuuteen. 1980-luvun alussa kameroiden valikoiman rajoitukset rajoittivat tietokonenäön kehitystä, kunnes kiinteän tilan kameroiden hinnat laskivat ja niiden teknologinen kehitys vastasi paremmin tietokonenäön tarpeita. Nykyään kamerayritykset jatkavat tuotetarkennusten kehittämistä vastaamaan tietokonenäön markkinoiden vaatimuksiin, mikä osoittaa kameroiden merkityksen tietokonenäön kehityksessä ja menestyksessä. [27.]

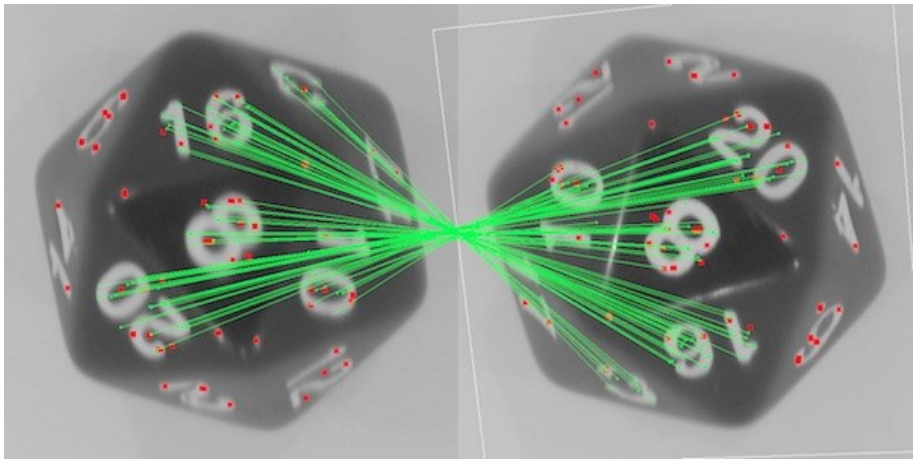
Kameran rooli automaattisessa nopanheittolaitteessa

Automaattisessa nopanheittolaitteessa keskeinen tavoite on nopan luvun tunnistaminen. Tämä osa-alue on ratkaisevan tärkeä projektin onnistumisen kannalta. Heti nopan heittämisen jälkeen laite hyödyntää kameraa nopan lukeman havaitsemiseksi, jonka jälkeen se välittää tiedon nopan tuloksesta edelleen käsiteltäväksi. Kamera toimii tässä prosessissa syötteen anturina, joka odottaa nopanheiton tapahtumista ja sitten lukee nopan antaman tuloksen tarkasti.

Kameran rooli laitteessa on kriittinen, sillä se on vastuussa nopan lukeman havaitsemisesta ja oikean tuloksen varmistamisesta. Kamera voidaan opettaa tunnistamaan eri nopan luvut syöttämällä sille valmiiksi kerättyä dataa, joka sisältää nopan numerot eri kulmissa ja suunnissa. Tällä opetusmetodilla pyritään varmistamaan, että kamera kykenee luotettavasti ja tarkasti

tunnistamaan nopan lukeman erilaisissa tilanteissa. Tämä taas takaa laitteen toiminnan luotettavuuden ja oikeudenmukaisuuden eri nopanheittojen aikana. Kameran opettaminen tähän tehtävään vaatii aikaa ja paljon dataa.

Kuvassa 15 nähdään esimerkki kuvantunnistuksen toimimisesta D20-nopan tuloksesta kameran ottamien kuvien pohjalta. Kuvassa näkyvät punaiset pisteet ja vihreät viivat ovat osana kuvaprosessoinnin tekniikkaa, missä kahdesta kuvasta tulkitaan yhteensopivia ominaisuuksia ja ne merkitään. Punaiset pisteet tarkoittavat avainkohtia tai mielenkiintoisia elementtejä, jotka löytyvät molemmista kuvista, esimerkiksi kulmat tai reunat. Vihreät viivat osoittavat punaisten avainkohtien yhteensopivuutta kahden kuvan välillä. Näiden vastaavuuksien löytämisen avulla kaksi kuvaa voidaan kohdistaa toisiinsa nähden ja niistä voidaan tulkita esimerkiksi orientaatio tai suhteellinen sijainti.



Kuva 15: Esimerkki Mark Fickettin tekemästä kuvantunnistuksen toiminnasta kameran ottaman kuvan suhteen. [28]

Kuvantunnistuksen haasteet ja ratkaisut

Kuvantunnistuksessa kohtaa monia haasteita, kuten olemassa olevan datan eheys, valaistusolosuhteet, kuvien tarkkuus, nopan epäsäännöllinen liike, perspektiivi, taustan häiriöt ja kameran kuvan laatu. Nämä kaikki tekijät vaikuttavat kuvantunnistuksen tuloksiin ja tarkkuuteen. Siksi on olennaista luoda suotuisat olosuhteet kuvantunnistukselle kameran laadun ja tulosten eheyden takaamiseksi. Tähän sisältyy esimerkiksi huolehtiminen siitä, että kuvan tausta

ei ole täynnä häiriötekijöitä, laitteen vakaa asento nopan heiton jälkeen, riittävä valaistus kameralle ja kameran optimaalinen sijainti nopan yläpuolella. [29.]

Lisäksi koneoppimista ja neuroverkkoja voidaan hyödyntää tulosten eheyden varmistamiseksi. Laitteen tietokantaan voidaan tallentaa valmiiksi dataa nopan eri luvuista ja tuloksista. Esimerkiksi voidaan tallentaa kuvia nopan eri silmäluvuista eri kulmissa ja suunnissa, mikä helpottaa numeron tunnistamista kameran toimesta ja varmistaa nopean ja tarkan tunnistamisen.

3.4 Haasteet ja testaus

Tekniset haasteet

Automaattisen nopanheittolaitteen suorituskykyyn ja luotettavuuteen voi vaikuttaa useita teknisiä haasteita, jotka voidaan jakaa mekaanisiin ja ohjelmistollisiin tekijöihin.

Mekaaniset haasteet voivat liittyä nopan pyörittämisen nopeuteen ja vakioon, joka vaikuttaa nopan tuloksen satunnaisuuteen ja lopputuloksen luotettavuuteen. Lisäksi kameran tarkkuus ja optimaalinen sijoittelu ovat tärkeitä tekijöitä, jotka vaikuttavat nopan tuloksen luotettavaan lukemiseen. Valon määrä ja laitteen vakaus voivat myös vaikuttavat kuvan laatuun ja siten kuvantunnistuksen tarkkuuteen.

Ohjelmistolliset haasteet voivat liittyä ohjelmiston koodauksen laatuun ja monimutkaisuuteen, mikä voi vaikuttaa kuvantunnistuksen tarkkuuteen ja luotettavuuteen. Lisäksi reaaliaikaisen tiedonsiirron vakautta on tärkeää varmistaa, jotta nopan heiton tulos välitetään serverille nopeasti ja virheettömästi.

Nämä haasteet edellyttävät huolellista suunnittelua ja testausta, jotta automaattisen nopanheittolaitteen suorituskyky ja luotettavuus voidaan optimoida. Mekaanisten osien ja kameran valinta, valaistuksen hallinta, laitteen vakauden varmistaminen sekä ohjelmiston kehitys ja testaus ovat kaikki avainasemassa laitteen toiminnan varmistamisessa. Ajankäytön optimointi ja

haasteiden huomioon ottaminen projektin tekemisessä vaikuttaa lopputulokseen ja projektin valmistumiseen.

Testausmenetelmät ja validointi

Kokonaisuuden testaus voidaan suorittaa manuaalisin testein käyttämällä projektin verkkosivua ja varmistamalla annettujen tulosten olevan validoituja ja oikeita tarkistamalla nopan tulos laitteesta suoraan. Tämä varmistaa, että laitteen ja palvelimen välinen tiedonsiirto toimii odotetusti ja että lopputulokset vastaavat todellisia heittoja. Lisäksi voidaan suorittaa järjestelmätestejä, joissa koko laitetta testataan kokonaisuutena varmistaakseen sen toiminnan luotettavuuden eri tilanteissa ja käyttöympäristöissä.

3.5 Olemassa olevia ratkaisuja

Nopan käytön historian ja yleisen suosion pohjalta on oletettavaa, että noppia on myös implementoitu digitaalisesti ja rakennettu erilaisia ratkaisuja vuosien varrella.

Adafruit automaattinen nopan pyörittäjä

Adafruit on kehittänyt ja rakentanut automaattisen nopan pyörittäjän, jossa nappia painamalla laitteen sisällä oleva DC-moottori pyörittää noppien alustaa ja saa ne liikkumaan kupolin sisällä. Kuvasta 16 nähdään, että tämä ratkaisu ei hyödynnä digitaalista yhteyttä mihinkään palveluun vaan käyttäjän tulee itse lukea noppien tulos kupolin läpi. [30.]

Kuva 16: Adafruitin rakentama automaattinen noppalaite [30]

Noppa-skanneri

Belgialainen Arranged BV on kehittänyt laitteen, joka skannaa laitteen alustalle heitetyt nopat ja palauttaa tuloksen käyttäjälle digitaalisesti. Laitteen käyttö perustuu teknologiaan, jossa käytetään miljoonia fotodiodeja alustan skannaamiseen ja nopan tulosten lukemiseen. Skannerin käyttöä varten noppien heitto tapahtuu kuitenkin käyttäjän toimesta. Kuvassa 17 nähdään kyseisen yrityksen suunnittelema laite. [31.]

Kuva 17: Arranged BV:n suunnittelema noppaskannerin alusta [31]

Mark Fickettin nopanheittäjä

Projektin periaatetta lähimpänä oleva ratkaisu on mahdollisesti Mark Fickettin rakentama nopanheittäjälaite. Laite on automatisoitu, se heittää noppaa purkin sisällä, purkin yläpuolella oleva kamera ottaa nopasta kuvan ja nopan luku luetaan digitaalisesti käyttämällä kuvantunnistusohjelmaa. Hänen projektinsa periaatteena on testata eri D20-noppien reiluutta ja luotettavuutta tulosten suhteen. Tämän perusteella hän testaa laitetta monilla eri nopilla ja kehittää taulukoita, joissa näkyvät eri noppien keskimääräiset tulokset. Kuvassa 18 nähdään hänen projektinsa noppalaitteen ja kameran kokoonpano. [28.]

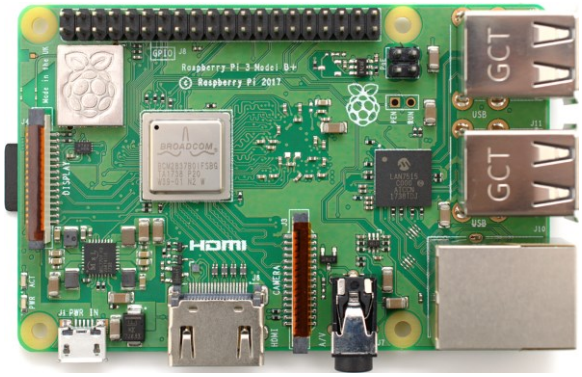
Kuva 18: Mark Fickettin nopanheittäjäprojektin asetelma. [28]

4 Nopanheittorobotin rakentaminen

4.1 Robotin laitteiston komponentit

Raspberry Pi

Projektin ohjausyksikkönä toimii Raspberry Pi 3b+. Se suorittaa ohjelmistoa, joka hallitsee nopan heittoa, kommunikoi kameran kanssa ja lähettää tiedot palvelimelle. Raspberry Pi voidaan liittää muihin projektin laitteisiin ja antureihin, kuten moottoriin ja kameraan, GPIO-porttien kautta (kuva 19).



Kuva 19: Kuvassa Raspberry Pi 3b+. [32]

Moottori

Moottori on vastuussa nopan heittämisestä. Moottorina toimii Multicomp Pro MM10 direct motor (DC) -moottori (kuva 20), joka ohjataan Raspberry Pi:n avulla pyörittämään alustaa, jossa noppa liikkuu. Moottori käynnistyy Raspberry Pi:n ohjelmiston käskystä ja pyörittää moottoria tietyn ajanjakson verran nopan heittoa varten. Virta moottoriin saadaan moottorikontrollerin ja 9 V:n alkaliparistolla avulla.

Kuva 20: Kuvassa Multicomp Pro MM10 DC -moottori. [33]

Moottoriohjainkortti

Moottorin toimimiseen tarvitaan vähintään kolme volttia virtaa, mikä ylittää Raspberry Pi:n GPIO-porttien tarjoaman jännitteen. Tämän vuoksi Raspberry Pi:hin on kytketty moottoriohjainkortti L298N (kuva 21), joka toimii välittäjänä moottorin ja virtalähteen välillä. Moottoriohjainkortti saa virtansa suoraan alkalisesta 9 voltin paristosta, ja se pystyy jakamaan tarvittavan virran moottorille moottorin ohjaussignaalien mukaan, jotka se saa Raspberry Pi:ltä. Tällä tavoin moottorin ohjaus ja tehon jakelu on eriytetty Raspberry Pi:stä, mikä varmistaa laitteen vakauden ja suorituskyvyn.

Kuva 21: Kuvassa moottoriohjainkortti L298N. [34]

Kamera

Kamera on olennainen osa projektia, koska se lukee nopan numeron heiton jälkeen. Projektissa käytetään Raspberry Pi:n RPI 8 megapikselin kamerasensoria (kuva 22), joka voidaan kytkeä suoraan Raspberry Pi:hin ribbon-liittimellä. Raspberry Pi:n ohjelmisto käsittelee kameran kuvaa ja tunnistaa nopan numeron kuvantunnistusalgoritmien avulla.

Kuva 22: Kuvassa Raspberry Pi Camera V2. [35]

PC / Monitori

Projektin palvelimen pyöriessä alustavasti Raspberry Pi:llä lokaalisena verkkosivuna tarvitaan verkkosivun näkemistä ja käyttöä varten monitori ja hiiri. Raspberry Pi voidaan suoraan yhdistää monitoriin ja hiireen verkkosivun käyttöä varten laitteen toimivuuden testaamiseksi. Monitori ja hiiri -yhdistelmä toimivat käyttöliittymänä, jonka avulla käyttäjä voi seurata nopan heittoa ja nähdä tuloksen reaaliajassa. Projektin tulevan kehityksen kannalta monitori tai lokaaliset päätelaitteet eivät välttämättä ole tarpeellisia, jos palvelimen ja nopanheittäjän välinen yhteys saadaan toimimaan verkkoyhteyden avulla routerin kautta.

Kuori ja kupoli

Kuoren ja kupolin käyttö on olennainen projektin toimivuuden ja tuloksen eheyden kannalta. Pyöritettävän nopan on oltava suljetussa tilassa, jotta sen liikettä eivät häiritse ulkoiset tekijät, vaan ainoastaan laitteen moottorin aiheuttama liike. Tämän saavuttamiseksi laitteessa on kuori, joka suojaa moottoria ja laitteistoa ja tarjoaa niille turvallisen ympäristön. Lisäksi tarvitaan lasinen tai muuten läpinäkyvä kupoli, joka mahdollistaa nopan turvallisen liikkumisen ja helpottaa sen tuloksen lukemista.

Projektin tuote on alkeellinen, minkä vuoksi siihen ei käytetä erityisiä tai kalliita materiaaleja. Alustan rakentamiseen käytetään saatavilla olevia materiaaleja, jotka toimivat perusrakenteena. Kupolina puolestaan toimii tyhjä lasipurkki, jonka pohjaan voidaan mahdollisesti porata reikä kameraa varten, mikäli kamera ei kykene lukemaan tulosta purkin lasin läpi.

Patteri ja patterin klipsi

Moottorin pyörittämiseen tarvitaan virtalähde, joka saadaan 9 voltin alkaliparistosta (kuva 23). Pariston tulee olla vaihdettavissa helposti virran loppuessa, joten se on kytketty laitteeseen patteri-klipsin avulla (kuva 24). Klipsin johdot kulkevat virranjakajaa varten tarkoitetun moottorin ohjaimen kautta.

Kuva 23: Kuvassa 9 V:n alkaliparisto. [36]

Kuva 24: Kuvassa patterin klipsi [37]

Hyppylanka

Hyppylangat ovat tarpeellisia tässä laitteessa, koska ne mahdollistavat joustavan ja helpon kytkennän eri osien välillä. Koska laite koostuu useista eri komponenteista, kuten Raspberry Pi:stä, moottorista ja moottoriohjaimesta, tarvitaan useita liitäntöjä ja kytkentöjä näiden osien välillä. Kuvassa 25 on useita naaras-uros-hyppylankoja.

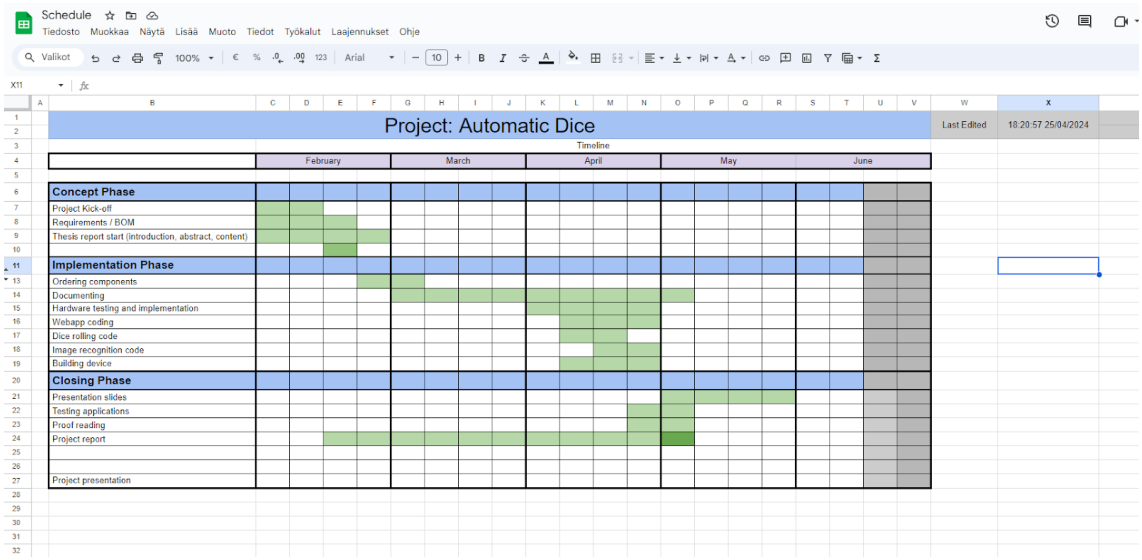
Kuva 25: Kuvassa Multicomp naaras-uros-hyppylankoja. [38]

4.2 Kehitystyökalut

Projektin päätoiminen kehitys tapahtuu Raspberry Pi:n avulla ja suurin osa koodaamisesta, laitteen toiminnasta ja projektin työstä tapahtuu Raspberryn käyttöjärjestelmässä. Työn ollessa yhden henkilön tekemä, ei ole tarvetta usealle eri kehitystyökalulle. Seuraavat työkalut koettiin projektin kannalta kattaviksi.

Google Drive

Google Driveen luotiin oma kansio, missä on projektin aikataulutusta ja tehtäviä varten luotu Excel-taulukko (kuva 26), materiaaleja ja tarvikkeita varten tehty BOM-lista (kuva 27) ja itse projektia koskeva raporttipohja ja sen tekoa tukevat tiedostot.



Schedule

Tiedosto Muokkaa Näytä Lisää Muoto Tiedot Työkalut Laajennukset Ohje

Project: Automatic Dice

Last Edited: 18:20:57 25/04/2024

Timeline

	February	March	April	May	June
Concept Phase					
Project Kick-off					
Requirements / BOM					
Thesis report start (introduction, abstract, content)					
Implementation Phase					
Ordering components					
Documenting					
Hardware testing and implementation					
Webapp coding					
Dice rolling code					
Image recognition code					
Building device					
Closing Phase					
Presentation slides					
Testing applications					
Proof reading					
Project report					
Project presentation					

Kuva 26: Kuvassa projektin tekijän kehittelemä aikataulu projektille.

Bill of Materials

Finished Product	Automatic Dice
Total Number Of Pieces	11
Total Price	88,70 €

Name	Product #	Category	Amount	Unit Cost	Cost	Interface	Where to get it	
Raspberry Pi 4	RPI4-MODBP-2GB	PC	1	41,38 €	41,38 €	-	https://fi.farnell.com/raspberry-pi/rpi4-mod	Koululta
Raspberry Pi 4 PSU	SC0212	Power Supply	1	9,56 €	9,56 €	DC	https://fi.farnell.com/raspberry-pi/sc0212/r	Koululta
Raspberry Pi Kotelo	PI4B_CASE_RED/WHITE	Case	1	4,42 €	4,42 €	-	https://fi.farnell.com/raspberry-pi/pi4b-cas	Koululta
Raspberry Camera V2	RPI 8MP CAMERA BOARD	Camera	1	13,58 €	13,58 €	CSi	https://fi.farnell.com/en-FI/raspberry-pi/rpi-	Tilaa
Moottorinohjain	L298N	Motor driver	1	3,28 €	3,28 €	DC	https://kauppa.sintosen.com/product/605/	Tilaa
Moottori	MM10	DC Motor	1	1,48 €	1,48 €	DC	https://fi.farnell.com/en-FI/multicomp/mm1	Tilaa
Battery 9V	MP-PSG91114	Battery	1	2,39 €	2,39 €		https://fi.farnell.com/pro-elec/psg91114/alk	Tilaa
Hyppylanka Naaras-Naaras	MP006284	Jumperwire	1	3,45 €	3,45 €		https://fi.farnell.com/en-FI/multicomp-pro/r	Tilaa
Hyppylanka Uros-Naaras	MP006283	Jumperwire	1	3,45 €	3,45 €		https://fi.farnell.com/en-FI/multicomp-pro/r	Tilaa
Patterin klipsi	29-130	Battery clip	1	0,81 €	0,81 €		https://fi.farnell.com/mcm/29-130/battery-s	Tilaa
Lasinen purkki					0,00 €			
Pienempi purkin kansi/alusta nopalle					0,00 €			
Styroksi					0,00 €			Käytä olemassa olevia materiaaleja
Vaneri / Kova muovi					0,00 €			
Pikaliima					0,00 €			
Kontaktimuovi					0,00 €			
D20 noppa			1	4,90 €	4,90 €		https://www.puolenkuunpelit.com/kauppa/	Tilaa
Total			**11**		**88,70 €**			

Kuva 27: Kuvassa projektin tekijän luoma Bill Of Materials -taulukko.

Geany

Geany on Raspberry Pi:ssä toimiva tehokas ja kevyt kehitysalusta tekstin editoimiseen esimerkiksi koodaamista varten. Sitä tukee GTK+ -lisäosa ja Scintilla-kirjastot, jotka tukevat yli 50 eri koodikielen kirjoittamista. Se sisältää halutuimmat ohjelmistoympäristön ominaisuudet kuten syntaksin korostus, koodin taitto, HTML- ja XML- tunnisteiden automaattinen sulkeminen.

4.3 Robotin ohjelmointiin tarvittavat alustat

Projektin ohjelmointi tapahtuu Raspberry Pin avulla, ja siinä käytetään Pythonin kevyttä "virtual environments" -moduulia. Tämän avulla projektiin tarvittavat alustat ja riippuvuudet voidaan pitää eristettynä koko muusta ympäristöstä. Virtuaalisen ympäristön käyttö on päätetty, koska tiettyjen komponenttien ohjelmointi vaatii oletuksesta poikkeavat kirjasto- ja riippuvuusversiot.

Ohjelmointikielet

Python

Projektin päätoiminen ohjelmointi tapahtuu Pythonin avulla. Pythonin avulla luodaan projektin ohjelmistokoodi, mikä suorittaa verkkosovelluksen rakennuksen, nopan pyörittämisen ja nopan silmäluvun lukemisen. Koodi myös lähettää nopan tuloksen nettisivulle käyttäjän nähtäville.

HTML

Projektin verkkosovellusta varten tarvitaan HTML:ää, jotta sivun komponentit ovat selkeästi nähtävissä ja käytettävissä. HTML määrittää nettisivun osioiden sijoittelun ja päätoimisen tekstin. HTML:n avulla myös määritetään sovelluksen nappi, jota painamalla lähetetään signaali nopan pyörittämiselle.

Javascript

Yhdistettynä nettisivun HTML:n kanssa käytetään myös hieman Javascriptiä jotta nettisivuille voidaan hakea sovelluksen ohjelman prosessoima nopan tulos ja näyttää se nettisivuilla. Javascriptin avulla nopan tulos päivittyy aina, kun nappia painetaan ja ohjelma heittää noppaa uudelleen.

Kirjastot ja riippuvuudet

Flask

Flask on Pythonille tarkoitettu mikroverkkokehys. Sillä on oma HTTP-palvelin, mutta sillä on rajallinen kapasiteetti. Flask on tehokas työkalu verkkosovelluksen luontiin ja python-funktioiden implementointiin HTML-sivulla, mikä mahdollistaa kommunikaation sovelluksen ja laitteiston välillä. Ylikuormituksen estämiseksi yleisesti suositaan käyttämään Apachen palvelinta yhdessä Flaskin kanssa, mutta Apache ei ole oleellinen projektin alkeellisuuden takia. [39.]

OpenCV

OpenCV, lyhenne sanoista Open Source Computer Vision Library, on olennainen osa tätä projektia. Se on avoimen lähdekoodin ohjelmistokirjasto, joka on suunniteltu tukemaan monenlaisia tietokoneiden näköön ja kuvankäsittelyyn liittyviä tehtäviä. Kirjasto tarjoaa laajan valikoiman työkaluja ja algoritmeja, jotka mahdollistavat kuvien ja videoiden käsittelyn, tunnistamisen, seurannan ja analysoinnin erilaisissa sovelluksissa. OpenCV:llä voidaan toteuttaa monenlaisia toimintoja, kuten kasvojen tunnistus, liikkeentunnistus, objektien seuranta ja kuvien rekonstruktio. Tämä kirjasto on suosittu ja laajasti käytetty monilla eri aloilla, kuten robotiikassa, lääketieteessä, turvallisuussovelluksissa, liikenteen valvonnassa ja teollisuusautomaatiossa. [40.]

Tässä projektissa OpenCV on keskeisessä roolissa kuvantunnistuksen toteuttamisessa. Sen avulla pyritään lukemaan nopan numero kameran ottamasta kuvasta ja käsittelemään tätä tietoa eteenpäin.

Picamera2

Picamera2 on libcamera-pohjainen korvaaja Picamera-sovellukselle, joka oli Python-liittymä Raspberry Pi:n vanhalle kamerapinon käytölle. Picamera2 tarjoaa myös helppokäyttöisen Python-ohjelmointirajapinnan. Picamera2 tukee ainoastaan Raspberry Pi OS Bullseye (tai uudempi) -kuvia, sekä 32- että 64-

bittisiä versioita. Picamera2 tarjoaa kehittyneitä toimintoja ja parannuksia verrattuna alkuperäiseen Picameraan, kuten nopeampaa suorituskykyä, parempaa yhteensopivuutta ja laajempaa tukialustaa. Lisäksi Picamera2 mahdollistaa monipuolisempia kuvan- ja videonkäsittelytoimintoja Raspberry Pi -alustalla. Tämä tekee siitä välttämättömän työkalun monenlaisiin Raspberry Pi -pohjaisiin kuvantunnistus- ja kameraprojekteihin, kuten esimerkiksi automaattisen noppalaitteen rakentamisessa. [41.]

RPi.GPIO

RPi.GPIO tarjoaa Python-moduulin, jolla voidaan ohjata Raspberry Pi:n GPIO-liittimiä. Tämä moduuli tarjoaa käyttäjälle helpon tavan hallita Raspberry Pi:n GPIO-liittimiä Python-ohjelmoinnin avulla. Se sisältää joukon toimintoja ja metodeja, jotka mahdollistavat pinnien tilan asettamisen, lukemisen ja muuttamisen sekä erilaisten GPIO-laitteiden ohjaamisen Raspberry Pi -alustalla. RPi.GPIO on laajalti käytetty ja suosittu työkalu Raspberry Pi -projekteissa, joissa tarvitaan pinnien ohjausta ja vuorovaikutusta ulkoisten laitteiden kanssa. Projekti käyttää RPi.GPIO:ta lähettääkseen GPIO-signaalin moottorille nopan pyörittämistä varten. [42.]

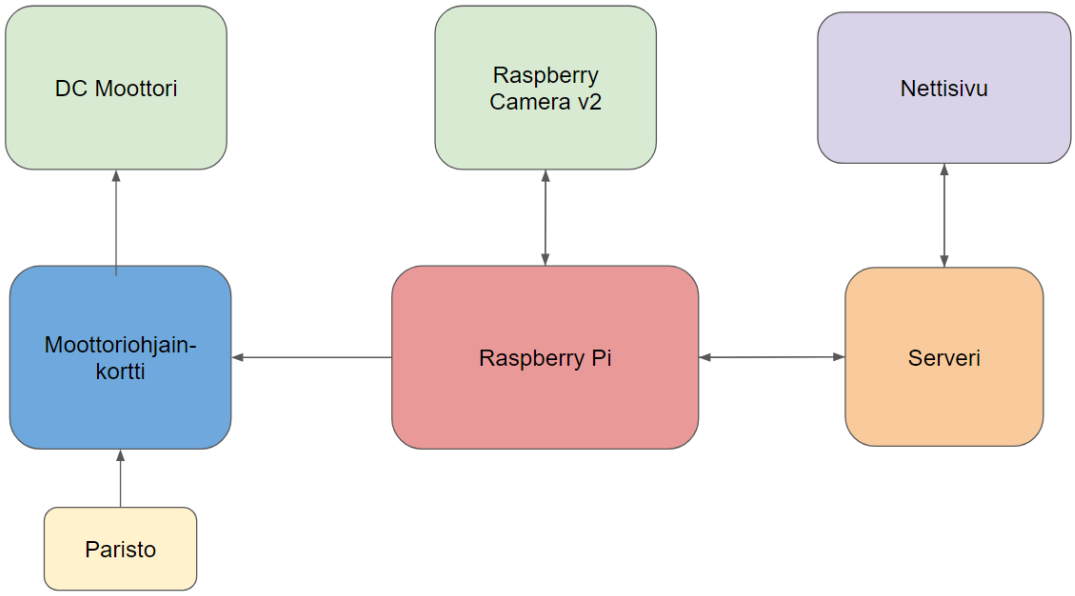
Time

Aikaan liittyvät toiminnot ovat tärkeitä monissa Raspberry Pi -projekteissa, ja Pythonin time-moduuli tarjoaa tehokkaita työkaluja ajan käsittelyyn ja hallintaan. Time-moduuli mahdollistaa ajan ja odotusten hallinnan, mikä on olennaista monissa erilaisissa sovelluksissa, kuten antureiden lukemisessa, tehtävien ajoituksessa ja tapahtumien synkronoinnissa.

Projektissa time-moduuli määrittelee tiettyjen funktioiden ajoituksen eri kutsujen välillä, jotta kerätty data on ehyttä ja luotettavaa. Esimerkkinä voidaan käyttää time-moluudin sleep()-funktiota, jonka avulla ohjelma odottaa tietyn ajan ennen seuraavan komennon tekemistä. [43.]

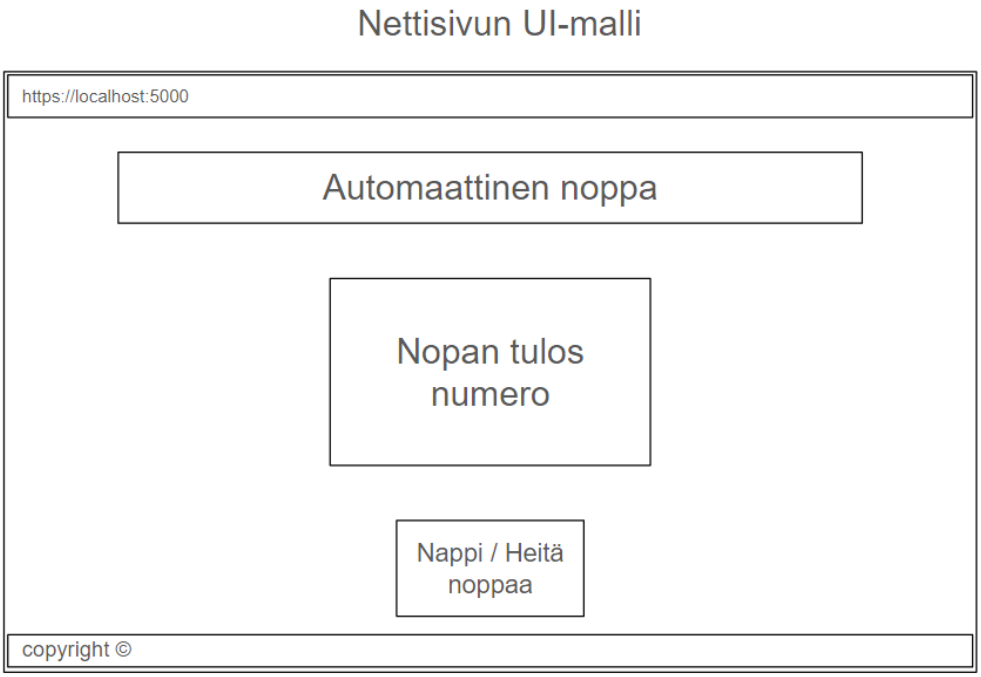
4.4 Robotin skeemakaavio ja nettisovelluksen UI-malli

Työn skeema on havainnollistettu kuvassa 28, jossa näkyvät geneeriset yhteydet komponenttien välillä. Raspberry Pi ylläpitää projektin kevyttä Flask-serveriä, missä projektin nettisivu sijaitsee. Flask toimii samaa aikaa serverinä ja nettisovelluksena. Raspberryyn myös yhdistyy laitteiston kamera ja moottoriohjainkortti, joka johtaa tarvittavan virran laitteen DC-moottorille ohjainkorttiin yhdistetystä paristosta.



Kuva 28: Projektin skeema, jossa on havainnollistettu projektin komponenttien geneerinen yhdistäminen.

Projektin nettisivu on mallinnettu kuvassa 29, josta näkee, että sivun ulkonäkö on yksinkertainen ja pelkistetty. Nettisivulla on vain sivun otsikko, nappi nopan pyörittämiseen ja ruutu, johon nopan lukema lähetetään laitteelta.

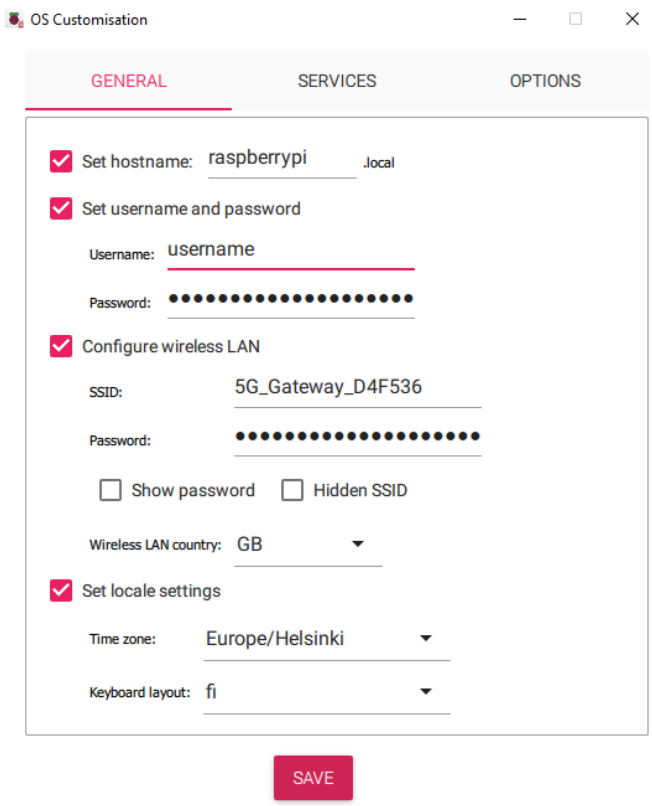


Kuva 29: Projektin nettisovelluksen alkeellinen UI-malli, missä näkyvät tarvittavat komponentit.

4.5 Robotin laitteiston kokoonpano

Raspberryn asennus

Raspberryn käyttöjärjestelmä asennettiin Raspberry Pi Imager -ohjelmiston avulla (kuva 30). Ohjelmisto tukee monia eri Raspberry-malleja, kuten projektissa käytettyä Raspberry Pi 3b+ -mallia. Sen avulla voi kustomoida itselleen sopivan käyttöjärjestelmän Raspberryyn. Tämän projektin kohdalla valitsimme täyden 64 bitin käyttöjärjestelmän, Debian 11 “Bullseye”n.

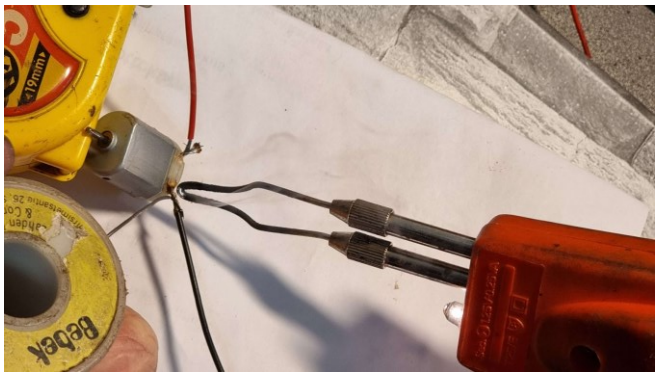


Kuva 30: Kuva Raspberry Pi Imager -ohjelman asetuksista käyttöjärjestelmän asentamisen yhteydessä.

Laitteen rakentaminen

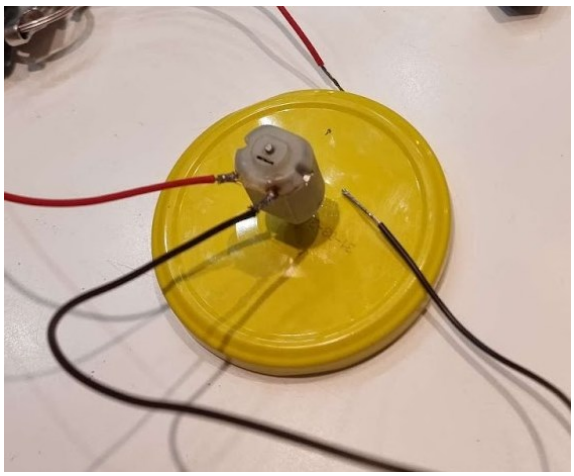
Laitteen rakentamisessa piti aluksi suunnitella alustava ulkomuoto. Tätä varten tehtiin tutkimusta olemassa olevista noppaa pyörittävistä tai heitettävistä laitteista, joista yksi helposti lähestyttävissä on Adafruitin suunnittelema Automatic Dice Roller. Adafruitin laitteessa on implementoitu DC-moottori, mikä pyörittää alustaa, jossa nopat liikkuvat läpinäkyvän kupolin sisällä. [33.] Tästä ulkonäöstä lähdettiin suunnittelemaan sopivaa laitteen ulkomuotoa projektille, jossa tarvitaan nopan näkyvyyttä kameraa varten ja pyörivää DC-moottoria noppien liikuttamiselle. Raskaisiin ja hintaviin materiaaleihin ei kuitenkaan ole vielä tarvetta projektin alkeellisuuden ja prototyyppisen luonteen vuoksi.

Laitteen rakentaminen aloitetaan kuoren rakentamisesta ja moottorin asentamisesta laitteeseen. Aluksi moottoriin kolvataan sähköjohdot virran saamiseksi (kuva 31).



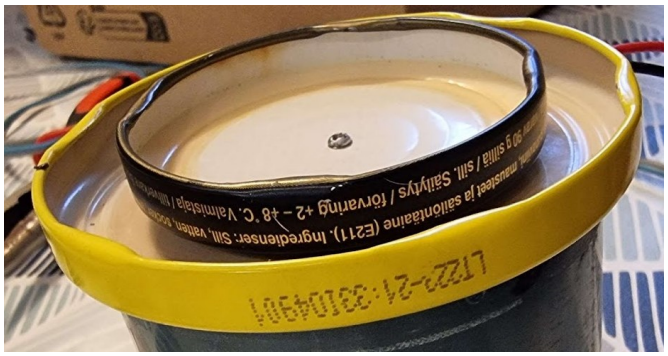
Kuva 31: Sähköjohtojen kolvaus.

Projektin kupoliksi sovelletaan läpinäkyvää lasipurkkia, jossa on metallikansi. Purkin kanteen porataan reikä, josta saadaan purkin sisälle laitettua akseli, joka pyörittää purkin sisällä olevaa alustaa, jossa 20-sivuinen noppa liikkuu. Akselin toinen pää on yhdistettynä DC-moottoriin, joka sijaitsee lasipurkin alapuolella (kuva 32). Nopan alustaksi lasipurkin sisälle sovelletaan pienempää purkin kantta, missä on hieman epätasaisuutta reunalla, mikä edesauttaa nopan ennakoimattoman liikkeen.



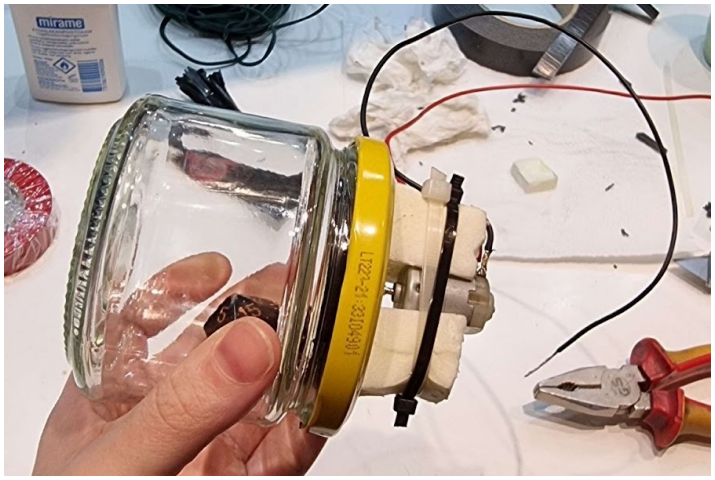
Kuva 32: Lasipurkin kanteen on porattu reikä, mikä on hieman suurempi kuin moottoriin yhdistetty akseli.

Akselin toiseen päähän kiinnitetään pienempi kansi alustaksi, akselin ollessa lasipurkin kannen sisäpuolella. Kiinnitys tapahtuu pienen ruuvin avulla. Keltaisen kannen reikä on tarpeeksi iso, jotta akseli mahtuu pyörimään moottorin avulla sujuvasti (kuva 33).



Kuva 33: Lasipurkin sisälle tuleva pienemmän kannen kiinnitys.

Moottori pitää kiinnittää keltaiseen kanteen tukevasti pystysuoraan, jotta se ei liiku tai tärise kannen pyörityksen aikana. Tämä myös varmistaa sisäisen kannen vapaan pyörimisen. Kanteen liimattiin kaksi paksua styroksin palaa moottorin molemmin puolin ja ne vielä kiristettiin moottorin ympärille nippusiteillä. Sähköjohdot myös vietiin nippusiteiden alapuolelle, jotta ne eivät liiku ja vaurioidu käytössä (kuva 34).



Kuva 34: Moottori kiinnitetään kanteen styroksin ja nippusiteiden avulla.

Moottorin suojaksi käytetään muovista putkea, mikä leveydeltään sopii moottorin ympärille, mikä jättää tilaa johdoille. Putkesta leikataan 7 cm:n pituinen pätkä, johon leikataan pienet ulostuloaukot johdoille. Putken pätkä liimataan keltaiseen kanteen, samalla pitäen huolen että mottorin sähköjohdot tulevat läpi ulostuloaukoista (kuva 35).

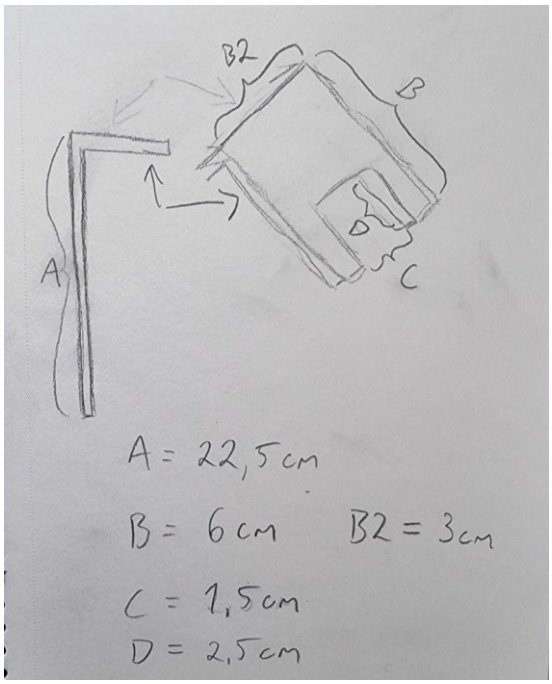
Kuva 35: Kanteen liimattu putkenpätkä.

Laitteen stabiloimiseksi ja liikkuvuuden estämiseksi myös leikattiin iso neliö styroksista, johon laitteen putki upotettiin. Styroksinen alusta ja putken osa päällystetään kontaktimuovilla visuaalisen ulkonäön takia (36).

Kuva 36: Laitteen kuori on rakennettu ja moottori asennettu. Moottorin sähköjohdot tulevat ulos pienestä aukosta kannen alapuolella.

Kameran yhdistäminen

Laitteeseen asennetaan Raspberry Pi Camera V2, joka on suoraan raspberryyn yhdistettävä kameramoduuli. Projektin alussa se suunniteltiin asennettavaksi suoraan lasipurkin päälle, johon porataan pieni reikä kameran linssille. Laitteen rakennuksen aikana ja kameraa testatessa ilmeni kameran kuvan sameudesta purkin ollessa matala ja nopan alustan ollessa vain noin 4 cm päässä kamerasta. Ratkaisuksi kameralle rakennetaan jalusta, joka pitää sitä noin 8 cm lasipurkin yläpuolella, mikä luo kameran ja nopan etäisyydeksi noin 12-13 cm. Tämä mahdollistaa kameran kuvan luotettavamman tarkkuuden ja kuvan ottamisen stabiloimisen. Jalusta tehdään vanerista. Kuvassa 37 on hahmoteltu jalustan ulkomuotoa.



Kuva 37: Suunnittelu-piirros kameran jalustalle.

Jalustan kokonaiset mitat ovat:

- korkeus 23 cm
- leveys 3 cm
- paksuus 0,7 cm.

Jalustaan kiinnitetään taso, johon kamera kiinnitetään.
Tason mitat ovat:

- pituus 6,5 cm
- leveys 3 cm
- paksuus 0,7 cm.

Jalustaan leikataan kameralle sopiva aukko, mikä on 3 cm pitkä ja 1,5 cm leveä. Kamera kiinnitetään jalustaan kahdella pienellä ruuvilla ja kameran ribbon-johto kiinnitetään toistaiseksi kuminauhoilla. Jalusta on upotettu styroksiseen alustaan, mutta on irroitettavissa tarpeen mukaan (kuva 38).

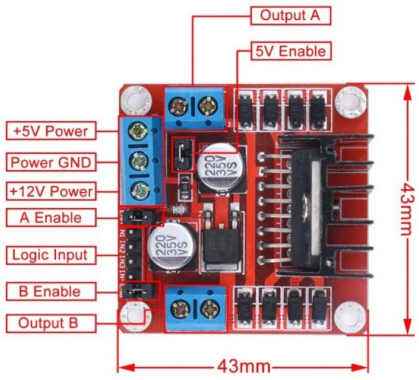
Kuva 38: Kamera kiinnitetään jalustaan ruuvien ja kuminauhojen avulla.

Laitteen yhdistäminen Raspberryyn

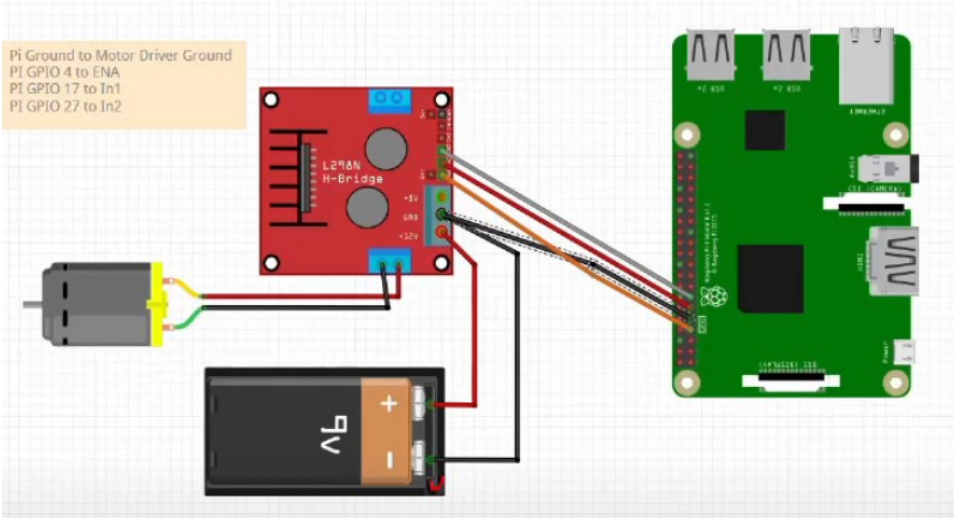
Laitteen yhdistäminen Raspberry Pi:hin tapahtuu kahdella eri tapaa. Laitteen kamera yhdistetään suoraan Raspberry Pi:hin kameran ribbon-kaapelin avulla. Raspberry Pi 3b+ -piirilevyssä on kameran paikka, mihin se yhdistetään suoraan.

DC-moottorin kiinnitys tapahtuu L298N-moottoriohjaimen avulla. Moottorista tulevat kaapelit yhdistetään ohjaimen IN1 ja IN2 ulostuloihin. Ohjaimessa on ruuvit, mitkä voi avata, kaapelit asentaa ruuvin alle ja ruuvin voi kiristää kaapeleiden paikalla pitämiseksi (kuva 39).
Moottoriohjaimeen yhdistetään moottorin virtalähde, 9 V:n alkaliparisto. Paristo kiinnittyy paristoklipsiin, jonka sähköjohdot kiinnitetään moottoriohjaimen GND ja +12V sisääntuloihin.
Moottoriohjain yhdistetään Raspberry Pi:hin neljän hyppyjohtimen avulla. Yksi johdoista yhdistetään moottoriohjaimen GND-liittimestä Raspberryn GND liittimeen (kuva 40).

Loogisiin sisääntuloihin asennetut johdot yhdistetään Raspberry Pi:hin seuraavasti:
ENA (A Enable) yhdistetään GPIO 4
In1 (Logical Input 1) yhdistetään GPIO 17
In2 (Logical Input 2) yhdistetään GPIO 27.



Kuva 39: Moottoriohjainkortti L298N liittimet ja ulkonäkö [44]



Kuva 40: Havainnollistava kuva komponenttien yhdistämisestä moottorin käyttöä varten. [45]

Projektin komponenttien kytkentäkaavio löytyy liitteestä 1.

4.6 Robotin ja verkkosovelluksen ohjelmointi

Virtuaaliympäristön luominen

Alkuperäisesti projektin ohjelmisto rakennettiin suoraan Raspberry Pi:n ympäristöön, mutta eri kirjasto ja tarvittavat riippuvuusversiot aiheuttivat ongelmia ohjelmiston kokoamisessa. Ratkaisuksi päätettiin luoda pythonin virtuaaliympäristö, mihin ohjelmistolle ladattiin tarvittavat kirjastot ja riippuvuudet ja luotiin kansiot ohjelmiston käyttämiselle.

```
python -m venv diceProject
source diceProject/bin/activate
```

Esimerkkikoodi 1. Käytetyt käskyt Raspberryn shellissä virtuaaliympäristön luomiselle ja käynnistämiselle

Riippuvuuksien asentaminen

Kun virtuaaliympäristö on luotu ja käynnistetty, voidaan asentaa projektia varten tarvittavat kirjastot ja paketit. Tutkimisen ja testaamisten tuloksena seuraavat

kirjastot todettiin tarpeellisiksi projektin toimivuuden kannalta ja asennetaan pip-komennolla, mikä on python-kielen oletuskomento kirjastojen asentamiselle:

```
pip install python3-picamera2
pip install opencv-python
pip install pytesseract
pip install flask
pip install numpy
pip install imutils
```

Esimerkkikoodi 2. Kirjastojen pakettien asentaminen virtuaaliympäristöön

Projekti voidaan rakentaa virtuaaliympäristön oletuskansioon tai sille voidaan luoda erillinen oma kansionsa. Projektille luodaan temp-kansio, johon ohjelma tallentaa kuvantunnistuksessa prosessoitavat kuvat eri funktioiden jälkeen. Temp-kansio ei ole pakollinen, mutta se auttaa kuvien näkemisessä ja ohjelman toimimisen tarkkailua vielä testausvaiheessa ja ohjelmoinnin optimoimisen aikana.

Verkkosovelluksen koodin kehitys

Laitteen ohjelmointi voidaan suorittaa yhden Python-ohjelman sisällä. Ohjelma ensin vastaanottaa napin painalluksen POST-signaalin ja lähettää sen jälkeen signaalin GPIO-pinnien kautta moottorin käynnistämiselle. Moottori pyörii kahden sekunnin ajan ja sitten pysähtyy. Se antaa tarpeeksi liikettä alustalle nopan pyörimiselle. Tämän jälkeen ohjelma avaa kameran, asettaa sen tarkennuksen ja ottaa kuvan nopasta. Ohjelma käsittelee kuvan pytesseract ja opencv-funktioiden avulla, jotta kuvasta saadaan erotettua nopan muoto ja numerot, joita ohjelma lähtee lukemaan digitaaliseen muotoon. Lopuksi ohjelma lähettää saadun tuloksen nettisovellukseen käyttäjän nähtäville. Nettisivu hakee tuloksen GET-signaalina.

Kuvassa 41 on kuvankaappaus Python-ohjelman kirjastojen tuonti ja muuttujien määrittely sekä nettisovelluksen aloitussivun määrittely.

```python
import RPi.GPIO as GPIO
from flask import Flask, render_template, Response, request, redirect, url_for
from picamera2 import Picamera2, Preview
import time
import json
import cv2
import pytesseract
from pytesseract import Output

GPIO.setwarnings(False)
app = Flask(__name__)

# Motor pins
in1 = 17
in2 = 27
en_a = 4

# Image And Mask Var
raw_image = ""
mask_image = cv2.imread("Mask.jpg")
processed_image = ""

# tesseract Settings
custom_oem = "outputbase digits"
min_conf = 0


@app.route('/')
def index():
    return render_template('index.html')

```

Kuva 41: Ohjelman kirjastojen tuonti ja muuttujien määrittely

Moottorin pyöritys tapahtuu kuvassa 42 tapahtuvan koodin avulla. GPIO-pinit asetetaan, jonka jälkeen määritellään moottorin voimakkuuden suuruus. Moottori käynnistetään, odotetaan kaksi sekuntia ja sen jälkeen moottori sammutetaan.

```python
def roll_dice():
    try:
        #Set GPIO
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(in1,GPIO.OUT)
        GPIO.setup(in2,GPIO.OUT)
        GPIO.setup(en_a,GPIO.OUT)

        #Set motors power level
        q=GPIO.PWM(en_a,100)
        q.start(20)

        #Turn the motor on
        GPIO.output(in1,GPIO.HIGH)
        GPIO.output(in2,GPIO.LOW)

        #Wait 2 second
        time.sleep(2)

        #Turn the motor off
        GPIO.output(in1,GPIO.LOW)
        GPIO.output(in2,GPIO.LOW)


    finally:
        #Cleanup GPIO
        GPIO.cleanup()

```

Kuva 42: Moottorin pyöritykseen kirjoitettu koodi, jossa käytetään GPIO-pinnejä signaalin lähettämiseen

Kuvan ottaminen tapahtuu picamera2-kirjaston funktioiden avulla, jolla määritellään ensin kameran konfiguraatiot, aloitetaan esikatselu, määritellään linssin tarkennus ja otetaan kuva. Kuvan ottamisen jälkeen kamera suljetaan (kuva 43).

```
#Function to take the picture with the camera module
def takePicture():
    #Define camera
    picam2 = Picamera2()

    #configure preview and resolution
    config = picam2.create_preview_configuration(main={"size": (1280, 720)}
    picam2.configure(config)

    #Start preview window
    picam2.start_preview(Preview.QTGL)

    #start camera
    picam2.start()

    #Set lens focus to focal length 5.0
    picam2.set_controls({"AfMode": controls.AfModeEnum.Manual,"LensPosition": 5.0})

    #Wait 5 seconds for focus
    time.sleep(5)

    #Take a picture
    picam2.capture_file("picture.jpg")

    #Stops preview and shuts down camera. Returns picture
    picam2.stop_preview()
    picam2.stop()
    return "picture.jpg"

```

Kuva 43: Kuvan ottaminen Raspberry Pi Camera V2 -moduulin avulla

Edellä mainitun funktion avulla saadaan otettua kuva nopasta purkin sisällä, niin kuin kuvassa 44 nähdään.

Kuva 44: Kameran ottama kuva purkin kannen läpi

Kuvan prosessointi tapahtuu OpenCV-kirjaston funktioiden avulla (kuva 45). Kuvan prosessoinnissa ensin kuva muutetaan binääriseen muotoon, jotta ohjelma voi käsitellä ja lukea siitä dataa. Tämä muuttaa kuvan mustavalkoiseksi ja erottaa numerot mahdollisimman selkeästi muusta taustasta. Maskeeraus-funktion avulla kuvaa verrataan olemassa olevaan kuvaan purkista ilman noppaa, jotta kuvasta voidaan erotella nopan sijainti. Grayscale-funktio muuttaa kuvan harmaaksi jatkokäsittelyä varten. noiseRemoval()-funktio vähentää kuvassa tapahtuvaa kohinaa ja parantaa näin datan käsittelyä ja numeroiden havaitsemista. inverImage()-funktio kääntää kuvan värit vastakohdikkain, mikä helpottaa tuloksen saamista kuvasta.

```
66  #Process image into binary
67  def binarizing(image):
68      return cv2.threshold(image, 110, 220, cv2.THRESH_BINARY)
69
70  #Image masking. Compares taken image to an image of the dice platform without the dice
71  def mask(mask, image):
72      return cv2.subtract(mask, image)
73
74  #Turns image into grayscale
75  def grayscale(image):
76      return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
77
78  #Adjust noise pollution of the image and smoothens it out
79  def noiseRemoval(image):
80      import numpy as np
81      kernel = np.ones((1, 1), np.uint8)
82      image = cv2.dilate(image, kernel, iterations=1)
83      kernel = np.ones((1, 1), np.uint8)
84      image = cv2.erode(image, kernel, iterations=1)
85      image = cv2.morphologyEx(image, cv2.MORPH_CLOSE, kernel)
86      image = cv2.medianBlur(image, 3)
87      return (image)
88
89  #Inverts image
90  def invertImage(image):
91      return cv2.bitwise_not(image)
92
```

Kuva 45: Otetun kuvan prosessointiin käytettävät funktiot

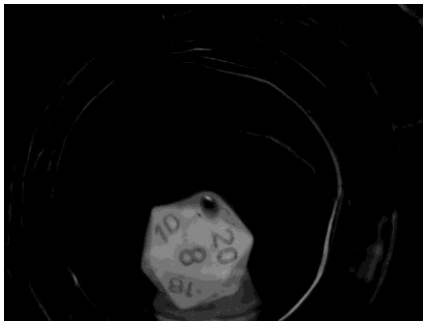
Kuvassa 46 edellä mainittuja funktioita kutsutaan otetun kuvan prosessoimiseksi ja käsittelyksi. Raw_imageksi määritellään kuva, mikä saadaan takePicture()-funktiolta, joka ottaa kuvan nopasta. Sen jälkeen sitä käsitellään ensin mask()-funktiolla mask, jonka jälkeen maskeerattu kuva käsitellään grayscale()-funktiolla. Harmaaksi muutettu kuva käsitellään binääriseksi binarizing()-funktiolla, jonka jälkeen kuvan värit käännetään

invert()-funktion avulla ja lopuksi kuva tallennetaan processed_image-muuttujaan.

```
raw_image = cv2.imread(takePicture())
cv2.imwrite("temp/rawImage.jpg", raw_image)

maskedImage = mask(mask_image, raw_image)
cv2.imwrite("temp/masked.jpg", maskedImage)

grayImage = grayscale(maskedImage)
cv2.imwrite('temp/gray.jpg', grayImage)

thresh, im_bw = binarizing(grayImage)
cv2.imwrite("temp/bw_image.jpg", im_bw)

invertedImage = invertImage(im_bw)
cv2.imwrite("temp/inverted.jpg", invertedImage)

processed_image = invertedImage
cv2.imwrite("temp/Pimage.jpg", processed_image)
```

Kuva 46: Otetun kuvan prosessointi aiemman esimerkin funktioilla

Kuva 47 ja 48 näyttävät kuvan prosessoinnin tuloksia eri ohjelmoinnin vaiheissa.



Kuva 47: Kuvan prosessoinnin tulos greyscale-funktioilla.

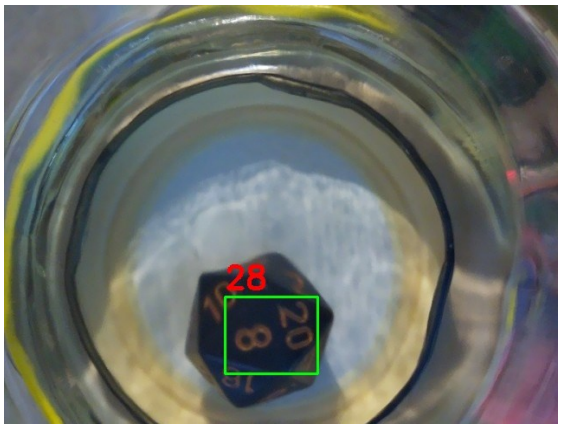
Kuva 48: Kuvan prosessoinnin tulos inverted-funktioilla.

Prosessoidusta kuvasta lähdetään tunnistamaan nopan tulosta pytesseract-funktioiden avulla. Pytesseract on kevyt kuvantunnistus kirjasto, joka on opetettu tunnistamaan tekstiä ja numeroita AI-tyylisesti. Sen avulla prosessoidusta kuvasta pyritään etsimään kaikki tekstin paikannukset ja niistä eritellään suurin todennäköisyys, mitä se havaitsee numeroksi tai virheellisesti kirjaimeksi (kuva 49).

```
    # Perform OCR (Optical Character Recognition) using Tesseract
    # Path to Tesseract executable
    pytesseract.pytesseract.tesseract_cmd = "/usr/bin/tesseract"
    results = pytesseract.image_to_data(
        processed_image, output_type=Output.DICT, config=custom_oem)

    # loop over each of the individual text localizations
    for i in range(0, len(results["text"])):
        # extract the bounding box coordinates of the text region from
        # the current result

        x = results["left"][i]
        y = results["top"][i]
        w = results["width"][i]
        h = results["height"][i]
        # extract the OCR text itself along with the confidence of the
        # text localization
        text = results["text"][i]
        conf = int(results["conf"][i])

        # filter out weak confidence text localizations
        if conf > min_conf:
            # display the confidence and text to our terminal
            print("Confidence: {}".format(conf))
            print("Text: {}".format(text))
            print("")
            # strip out non-ASCII text so we can draw the text on the image
            # using OpenCV, then draw a bounding box around the text along
            # with the text itself
            text = "".join([c if ord(c) < 128 else "" for c in text]).strip()
            cv2.rectangle(raw_image, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cv2.putText(raw_image, text, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX,
                        1.2, (0, 0, 255), 3)
    # show the output image
    cv2.imwrite("temp/Image.jpg", raw_image)
```

Kuva 49: Prosessoitu kuva käsitellään Pytesseractin avulla, joka hahmottaa kuvasta nopan alueen ja yrittää lukea nopan numeron

Lopuksi ohjelma näyttää alkuperäisen kuvan, mihin on rajattu tunnistettu teksti ja sen saama tulos. Pytesseractin opetettu kirjasto ei ohjelmoinnin ja testaamisen tuloksena ole tarpeeksi kattava luotettavan tuloksen saamiseksi (kuva 50).



Kuva 50: Ohjelman luoma numeroiden tunniste nopasta. Koodia pitää vielä optimoida oikean tuloksen saamiseksi.

Numeron lähetys verkkosovellukseen tapahtuu kuvassa 51 näkyvän koodin kautta. Kuvasta saatu tulos tallennetaan muuttujaan, mikä lähetetään JSON:ina palvelimelle, joka päivittää tuloksen nettisivulle.

```
150         textjason = results["text"][i]
151         print(textjason)
152         # print(text)
153         cv2.waitKey(0)
154         return json.dumps({"roll_message": textjason})
155
```

Kuva 51: Numeron tiedon lähetys tapahtuu JSON:in avulla, mikä mahdollistaa numeron vaihtumisen uuten tulokseen verkkosovelluksessa napin painamisen jälkeen.

Kuvassa 52 nähdään nettisovelluksen käynnistykseen tarvittava koodi, mikä määrittelee host-osoitteen ja sen käyttämän portin nettisovelluksen toimimiselle. Nämä ovat lokaaliset asetukset nettisovellukselle.

```
157 if __name__ == '__main__':
158     app.run(debug=True, host='0.0.0.0', port="5001")
159
```

Kuva 52: Python-ohjelman lopussa määritellään verkkosovelluksen rakentaminen ja portit

Koko ohjelman koodi on nähtävillä liitteessä 2.

Käyttöliittymä

Projektin käyttöliittymänä toimii yksinkertainen HTML-nettisivu, joka rakentuu sivun otsikosta, nopan heittämiseen tarvittavasta napista ja nopan tuloksen kertovasta ruudusta. HTML-koodiin on lisätty Javascriptillä kirjoitettu funktio, joka hakee nopan luvun ja päivittää sen sivulle (kuva 53).

```
<html>
<head>
<meta name="viewport" content="width=device-widrh" />
<title>Automatic Dice</title>
</head>
    <body>
    <center>
    <h1>Automatic Dice<h1>
    <br>
    <br>
    <form id="rollForm" action="/roll" method="post">
        <button name="rollBtn" type="submit">Roll Dice!</button>
    </form>
    <br>
    <h3 id="numberResult"></h3>
    </center>
    <script>
    // Function to handle form submission
    document.getElementById("rollForm").addEventListener("submit", function (event) {
        event.preventDefault(); // Prevent the default form submission behavior

        // Make an AJAX request to the /roll/ route
        fetch("/roll/", {
            method: "POST",
        })
        .then(response => response.json()) // Parse the response as JSON
        .then(data => {
            // Update the displayed random number
            document.getElementById("numberResult").textContent = data.roll_message;
        })
        .catch(error => {
            console.error("Error fetching data:", error);
        });
    });
    </script>
    </body>
</html>
```

Kuva 53: HTML-ohjelmakoodi verkkosovellukselle. Koodissa on myös JSON-skripti nopan tuloksen näyttämiseksi.

Kuvassa 54 nähdään rakennettu nettisivu ja sinne ilmestyvä nopan lukeman tulos, mikä päivittyy aina uuteen tulokseen, kun nappia painetaan, eli palvelimelle lähetetään uusi POST-komento nopan heittämistä varten.

Kuva 54: Kuva projektin HTML-sivusta, jossa käyttäjä painaa nappia ja ohjelma palauttaa nopan lukeman.

Vaihtoehtoinen ratkaisu nopan tuloksen näyttämiseksi

Koska Pytesseractin ja OpenCV:n avulla ei saavutettu haluttua lopputulosta, työhön suunniteltiin vaihtoehtoinen ratkaisu. Numeron tunnistuksen sijaan nopasta otettu kuva lähetettäisiin verkkosovellukseen. Nopasta otettu kuva käsitellään OpenCV:n avulla ja kuvantunnistusta hyödynnetään siten, että kuvasta rajataan ainoastaan noppa ja rajattu kuva lähetetään nettisovellukseen käyttäjän nähtäville.

Työn alkuperäistä koodia muutetaan, jotta nopasta otettu kuva saadaan rajattua ja lähetettyä nettisivulle. Noppalaitteen moottorin pyöritys tapahtuu samalla tavalla, kuin aikaisemmassa koodissa. Nopasta otetun kuvan käsittelyä varten ohjelmaan implementoidaan staattiset tiedostosijainnit (kuva 55).

```
# Static file paths
MASK_PATH = "Images/mask.jpg"
CAM_IMAGE = "temp/rolledDice.jpg"
REF_PATH = "Images/ref.jpg"
```

Kuva 55: Kuvien staattiset tiedostosijainnit.

Koska uusi koodi käyttää staattisia tiedostosijainteja, on tärkeää saada kameran ottama kuva oikeaan kansioon. Picamera-kirjasto ei mahdollista kuvan sijainnin määrittelyä kuvan ottamisen aikana, joten kuvalle tarkoitetun kansion sijainti määritellään ennen kuvan ottamista. Tämä tapahtuu os-kirjaston avulla, joka tuodaan mukaan koodiin. Kuvassa 56 nähdään, kuinka os.chdir-komennon avulla siirrrytään oikeaan tiedostosijaintiin ennen kuvan ottamista kameralla. Kuvan ottamisen jälkeen kamera sammutetaan ja tiedostosijainti muutetaan takaisin alkuperäiseen kansioon, jotta muut staattiset tiedostosijainnit toimivat oikein.

```
# Function to take the picture and save it in temp folder with the name "rolledDice.jpg"
def takePicture():
    # Taking a photo with picam
    # Save image to CAM_IMAGE location
    os.chdir('/home/niklask/Thesis/temp')
    picam2 = Picamera2()
    config = picam2.create_preview_configuration(main={"size": (1280, 960)})
    picam2.configure(config)
    picam2.start()
    picam2.set_controls({"AfMode": controls.AfModeEnum.Manual,"LensPosition": 5.0})
    time.sleep(2)
    picam2.capture_file("rolledDice.jpg")
    picam2.stop()
    picam2.close()
    os.chdir('/home/niklask/Thesis')
    return 0
```

Kuva 56: Uusi funktio kuvan ottamiseen kameralla, jossa myös tiedostosijainti muuttuu ennen kuvan ottoa ja kuvan ottamisen jälkeen.

Kuvassa 57 nähdään vaihtoehtoisen koodin main-funktio eli pääohjelma, joka kutsuu eri funktioita halutulla tavalla ja lopuksi palauttaa prosessoidun kuvan nettisovellukseen käyttäjän nähtäville.

```
30  # This is the route for the roll, tells flask to run the roll_dice function
31  # and then take a picture and then run the precessing_logic function
32  @app.route("/roll/", methods=['POST'])
33  def main():
34
35      roll_dice()
36
37      time.sleep(3)
38
39      takePicture()
40
41      time.sleep(2)
42
43      final_Location = precessing_logic()
44      return jsonify({'url': final_Location})
45
```

Kuva 57: Uusi main-ohjelma sovellukselle.

Kuvan prosessointi tapahtuu erilailla, koska siitä ei enään yritetä erottaa numeroita, vaan kuvasta tunnistetaan noppa ja kuva rajataan nopan mukaan ja suurennetaan sopivaan kokoon. Kuvan prosessointi tapahtuu monen pienemmän funktion avulla, joita kutsutaan erikseen precessing_logic-funktiossa (kuva 58).

```
# Main Image Processing Logic for finding and cutting out the die
def precessing_logic():

    # Main Image processing
    image = read_image(CAM_IMAGE)
    maskImage = read_image(MASK_PATH)
    save_image(maskImage, "temp/maskImage.jpg")
    masked = mask_image(image, maskImage)
    save_image(masked, "temp/masked.jpg")
    gray = gray_image(masked)

    # Quick Ref Image Processing
    ref_mod = read_image(REF_PATH)
    gray_ref = gray_image(ref_mod)
    save_image(gray_ref, "temp/gray_ref.jpg")

    # Find and cutout die
    top, bottom = find_die(gray, gray_ref)
    cutout = cutout_die(image, top, bottom)
    save_image(cutout, "temp/rolledDice.jpg")  # pick a good folder
    # change name based on saved image name above DO NOT TOUCH /image/
    return "/image/rolledDice.jpg"
```

Kuva 58: Päätoiminen kuvan prosessointi-funktio.

Lopputuloksena ohjelma lähettää nopasta rajatun kuvan nettisovellukselle, josta käyttäjä voi nähdä nopan tuloksen (kuva 59). Painamalla ”Roll Dice!” -nappia ohjelma pyörittää noppaa uudelleen ja kuva päivittyy uuteen nopan tulokseen.

Kuva 59: Nopasta otettu kuva on rajattu ja lähetetty nettisovellukseen.

Koko ohjelman koodi on nähtävillä liitteessä 3.

5 Työn lopputulos

Opinnäytetyön rakentamisen ja koodiesimerkkien perusteella projekti eteni haluttua lopputulosta kohden. Saadut tulokset eivät välttämättä vastanneet kaikkia odotuksia, mitä projektille asetettiin työtä aloitettaessa, mutta työn tulokset ovat silti nähtävillä. Laitteen osat toimivat Raspberryn lähettämien signaalien mukaan, ja ohjelmisto käsittelee annetun datan funktioiden mukaan. Datan eheys ja selkeys selkeästi vaikuttaa saatuun lopputulokseen, sekä myös funktioiden optimointi ja tarkkuus. Tulosten tarkkuutta ja oikeutta voidaan parantaa laitteen ja funktioiden kehittämisen avulla.

Projektin työ on osa suurempaa projektia, joten työn kehittäminen ja optimointi ovat suositeltavia. Nopan tuloksen validoinnin ja luotettavuuden parantaminen on mahdollista kehittämällä laitteen ja ohjelmoinnin ominaisuuksia.
Insinöörityön tekijän ehdottamia kehityskohteita ovat:

- laitteen taustan optimointi, kuten kansi- ja sisäpintojen maalaaminen yksiväriseksi, esimerkiksi valkoiseksi
- hyvän valaistuksen varmistaminen, esimerkiksi led-nauhan asentaminen laitteen kupolin sisäpuolelle
- kuvaprosessoinnin parametrien optimointi ja nopan kuvan selkeämpi prosessointi ohjelmoinnin avulla
- mahdollinen ohjelman kouluttaminen dataseteillä eri tyylisistä nopista.
- verkkosovelluksen integroiminen serveri-palveluun, kuten Apache, ja sen julkaiseminen testikäyttöön.

6 Yhteenveto

Insinöörityön tarkoituksena oli tutkia ja rakentaa automaattinen noppalaite, joka mahdollistaa täysin satunnaisen luvun generoinnin digitaaliseen pelikäyttöön säilyttäen samalla perinteisen nopan käytön. Työssä käytiin läpi nopan käytön historiaa ja kehitystä pöytäpeleistä nykyaikaiseen digitaalisiin peleihin. Projektin teoreettinen osa koostui nopan mekaniikan tutkimisesta ja projektissa käytettävien teknologioiden käsittelemisestä.

Projektin laajuus muuttui projektin edetessä, mutta pääperiaate ja halutun laitteen rakentaminen pysyivät ennallaan. Työssä käytetään monen teknologian ja kädentaidon osaamista laitteen rakentamisesta komponenttien yhdistämiseen ja niiden ohjelmointiin digitaalisessa ympäristössä. Projektin lopputuloksena valmistui toimiva laite, joka pyörittää D20-noppaa, kun käyttäjä painaa nettisovelluksen nappia. Laite lukee nopan tuloksen kameran avulla ja lähettää tiedon takaisin nettisovellukselle. Projekti toimii vain lokaalisena verkkosovelluksena ja laitteen antamia nopan tuloksia ei saatu validoitua täysin luotettaviksi.

Kuvantunnistusta on käytetty ja kehitetty teknologiassa ja erilaisissa projekteissa jo monta vuotta, mutta suurimmissa osissa näissä projekteissa koneille annettavat numerot ovat oikein päin kameraa kohden. Tämän projektin vaikeus tuli selväksi tämän kulmakiven myötä, sillä nopan numeron suuntaa on mahdotonta tietää etukäteen, jolloin kameran ottamassa kuvassa noppa ei välttämättä ole oikein päin kameraan nähden. Tämä vaikeuttaa kuvantunnistusprosessia ja saatujen tuloksien validointia oikeiksi.

Robotiikan ja kuvantunnistus-teknologioiden hyödyntämistä perinteisten hyödykkeiden ja palveluiden digitalisoinnissa on jatkuvassa kasvussa. Ne mahdollistavat monimuotoisten ryhmien ja kommunikaation synnyn, ja monet yksilöt hyötyvät näiden asioiden digitoitumisesta nykyaikaan. Pelaaminen on useimmiten hauskempaa ryhmässä kuin yksin.

Lähteet

1 The Full History of Board Games. Medium. Verkkoaineisto. Saatavilla: https://medium.com/@peterattia/the-full-history-of-board-games-5e622811ce89. Luettu 24.3.2024.

2 Oldest known gaming tokens dug up in Bronze Age Turkish graves. NBC News. Verkkoaineisto. Saatavilla: https://www.nbcnews.com/sciencemain/oldest-known-gaming-tokens-dug-bronze-age-turkish-graves-6c10920354. Luettu 24.3.2024.

3 Dungeons & Dragons. Britannica. Verkkoaineisto. Saatavilla: https://www.britannica.com/topic/Dungeons-and-Dragons. Luettu 24.3.2024.

4 The Role of Dice in Dungeons and Dragons. Medium. Verkkoaineisto. Saatavilla: https://medium.com/@pauljones_85805/the-role-of-dice-in-dungeons-and-dragons-732cf9a4a749. Luettu 24.3.2024.

5 DND Dice Set. Amazon. Verkkoaineisto. Saatavilla: https://www.amazon.com/CREEBUY-Dungeon-Dragons-Playing-Polyhedral/dp/B091MLJSHL?th=1. Luettu 24.3.2024.

6 Monopoly Go! is now available on mobile devices. Gamingtrend. Verkkoaineisto. Saatavilla: https://gamingtrend.com/news/monopoly-go-is-now-available-on-mobile-devices/. Luettu 24.3.2024.

7 The Role of Augmented Reality in Gaming. Medium. Verkkoaineisto. Saatavilla: https://medium.com/technology-buzz/the-role-of-augmented-reality-in-gaming-68fb31a9d167. Luettu 24.3.2024.

8 AR system brings holographic board games to the table. New Atlas. Verkkoaineisto. Saatavilla: https://newatlas.com/games/tilt-five-augmented-reality-tabletop-gaming-system/. Luettu 24.3.2024.

9 The Impact of Technology on Board Games: Apps, AI, AR and Beyond. Joyful Games. Verkkoaineisto. Saatavilla: https://joyful-games.com/blogs/card-and-board-games-101/impact-of-technology-on-board-games-apps-ai-ar-and-beyond. Luettu 24.3.2024.

10 Understanding how virtual reality works. Talespin. Verkkoaineisto. Saatavilla: https://www.talespin.com/reading/understanding-how-virtual-reality-works. Luettu 24.3.2024.

11 “All on Board” brings board game nights to virtual reality. Mixed News. Verkkoaineisto. Saatavilla: https://mixed-news.com/en/all-on-board-brings-board-game-nights-to-virtual-reality/. Luettu 24.3.2024.

12 Stelios Avramidis. Why is the sum of the rolls of two dices a Binomial Distribution?. [Foorumikysymys]. Mathematics. 24.3.2015. Saatavilla: https://math.stackexchange.com/questions/1204396/why-is-the-sum-of-the-rolls-of-two-dices-a-binomial-distribution-what-is-define. Luettu 15.4.2024.

13 Pseudo Random Number Generator (PRNG). Geeks for Geeks. Verkkoaineisto. Saatavilla: https://www.geeksforgeeks.org/pseudo-random-number-generator-prng/. Luettu 15.4.2024.

14 Random Number Generator. Hypr. Verkkoaineisto. Saatavilla: https://www.hypr.com/security-encyclopedia/random-number-generator. Luettu 15.4.2024.

15 Basic True Random Number Generator (TRNG). Researchgate. Verkkoaineisto. Saatavilla: https://www.researchgate.net/figure/Basic-True-Random-Number-Generator-TRNG-block-diagram_fig8_3340063199. Luettu 15.4.2024.

16 Virtual Dice. Google. Verkkoaineisto. Saatavilla: https://www.google.com/search?q=virtual+dice. Luettu 15.4.2024.

17 Free Online Dice. Verkkoaineisto. Saatavilla: https://freeonlinedice.com/. Luettu 15.4.2024.

18 Tiesitkö tämän robotiikasta? Jamk. [pdf-dokumentti]. Saatavilla: https://www.jamk.fi/sites/default/files/2021-10/robocoutryside-valijulkaisu-web-small.pdf. Luettu 20.4.2024.

19 Process Automation. Smartkrow. Verkkoaineisto. Saatavilla: https://smartkrow.com/rpa.html. Luettu 20.4.2024.

20 Ohjelmistorobotiikka (RPA). Digital Workforce. Verkkoaineisto. Saatavilla: https://digitalworkforce.com/fi/digityontekija/rpa-ohjelmistorobotiikka/. Luettu 20.4.2024.

21 Automaatio ja automaatiojärjestelmät. Valmistajat. Verkkoaineisto. Saatavilla: https://valmistajat.fi/menetelmat/elektroniikka/automaatio-ja-automaatiojarjestelmat. Luettu 20.4.2024.

22 Welcoming Eve - A major acquisition for Smart Buildings. ABB. Verkkoaineisto. Saatavilla: https://new.abb.com/news/detail/104186/welcoming-eve-a-major-acquisition-for-smart-buildings. Luettu 20.4.2024.

23 Robotiikan ja automaation tulevaisuus sekä standardisointi Suomessa. Metsta. Verkkoaineisto. Saatavilla: https://metsta.fi/robotiikan-ja-automaation-tulevaisuus-seka-standardisointi-suomessa/. Luettu 20.4.2024.

24 Who Will Own the Robots?. Technology Review. Verkkoaineisto. Saatavilla: https://www.technologyreview.com/2015/06/16/11184/who-will-own-the-robots/. Luettu 20.4.2024.

25 What is image recognition?. Deepomatic. Verkkoaineisto. Saatavilla: https://deepomatic.com/blog/what-is-image-recognition. Luettu 20.4.2024.

26 Image detection, recognition and image classification with machine learning. Medium. Verkkoaineisto. Saatavilla: https://medium.com/ai-techsystems/image-detection-recognition-and-image-classification-with-machine-learning-92226ea5f595. Luettu 20.4.2024.

27 Cameras in Machine Vision Applications. Automate. Verkkoaineisto. Saatavilla: https://www.automate.org/vision/industry-insights/cameras-in-machine-vision-applications. Luettu 20.4.2024.

28 Mark Fickett Art: Dice Roller. Mark Fickett. Verkkoaineisto. Saatavilla: http://www.markfickett.com/stuff/artPage.php?id=389. Luettu 28.4.2024.

29 Seeing Through the Machine's Eyes: Top Challenges in Image Recognition. Verkkoaineisto. Saatavilla: https://chisw.com/blog/image-recognition-challenges/. Luettu 28.4.2024.

30 Automatic Dice Roller. Adafruit. Verkkoaineisto. Saatavilla: https://learn.adafruit.com/automatic-dice-roller/overview. Luettu 28.4.2024.

31 Instant scanning of dice results. Dice-Scanner. Verkkoaineisto. Saatavilla: https://www.dice-scanner.com. Luettu 28.4.2024.

32 File: Raspberry Pi 3 B+. Wikipedia. Verkkoaineisto. Saatavilla: https://en.m.wikipedia.org/wiki/File:Raspberry_Pi_3_B%2B_%283990636925%29.png. Luettu 28.4.2024.

33 Multicomp Pro MM10. Farnell. Verkkoaineisto. Saatavilla: https://fi.farnell.com/en-FI/multicomp/mm10/motor-miniature-1-5-3-0v-16-3krpm/dp/599104. Luettu 28.4.2024.

34 L298N Moottoriohjainkortti. Sintosen palvelut. Verkkoaineisto. Saatavilla: https://kauppa.sintosen.com/product/605/l298n-moottoriohjainkortti. Luettu 28.4.2024.

35 Raspberry Pi Camera Board V2. Farnell. Verkkoaineisto. Saatavilla: https://fi.farnell.com/en-FI/raspberry-pi/rpi-8mp-camera-board/raspberry-pi-camera-board-v2/dp/3677845. Luettu 28.4.2024.

36 Multicomp PRO MP-PSG91114. Farnell. Verkkoaineisto. Saatavilla: https://fi.farnell.com/pro-elec/psg91114/alkaline-battery-9v-pp3-1pk/dp/2503729. Luettu 28.4.2024.

37 MCM 29-130. Farnell. Verkkoaineisto. Saatavilla: https://fi.farnell.com/mcm/29-130/battery-snap-connector-9v/dp/4245111. Luettu 28.4.2024.

38 MULTICOMP PRO MP006283. Farnell. Verkkoaineisto. Saatavilla: https://fi.farnell.com/en-FI/multicomp-pro/mp006283/jumper-wire-kit-male-to-female/dp/3617771. Luettu 28.4.2024.

39 Deploying a Flask Application via the Apache Server. Open Source For U. Verkkoaineisto. Saatavilla: https://www.opensourceforu.com/2023/03/deploying-a-flask-application-via-the-apache-server/. Luettu 28.4.2024.

40 OpenCV. Verkkoaineisto. Saatavilla: https://opencv.org/. Luettu 28.4.2024.

41 Picamera2. Github. Verkkoaineisto. Saatavilla: https://github.com/raspberrypi/picamera2. Luettu 28.4.2024.

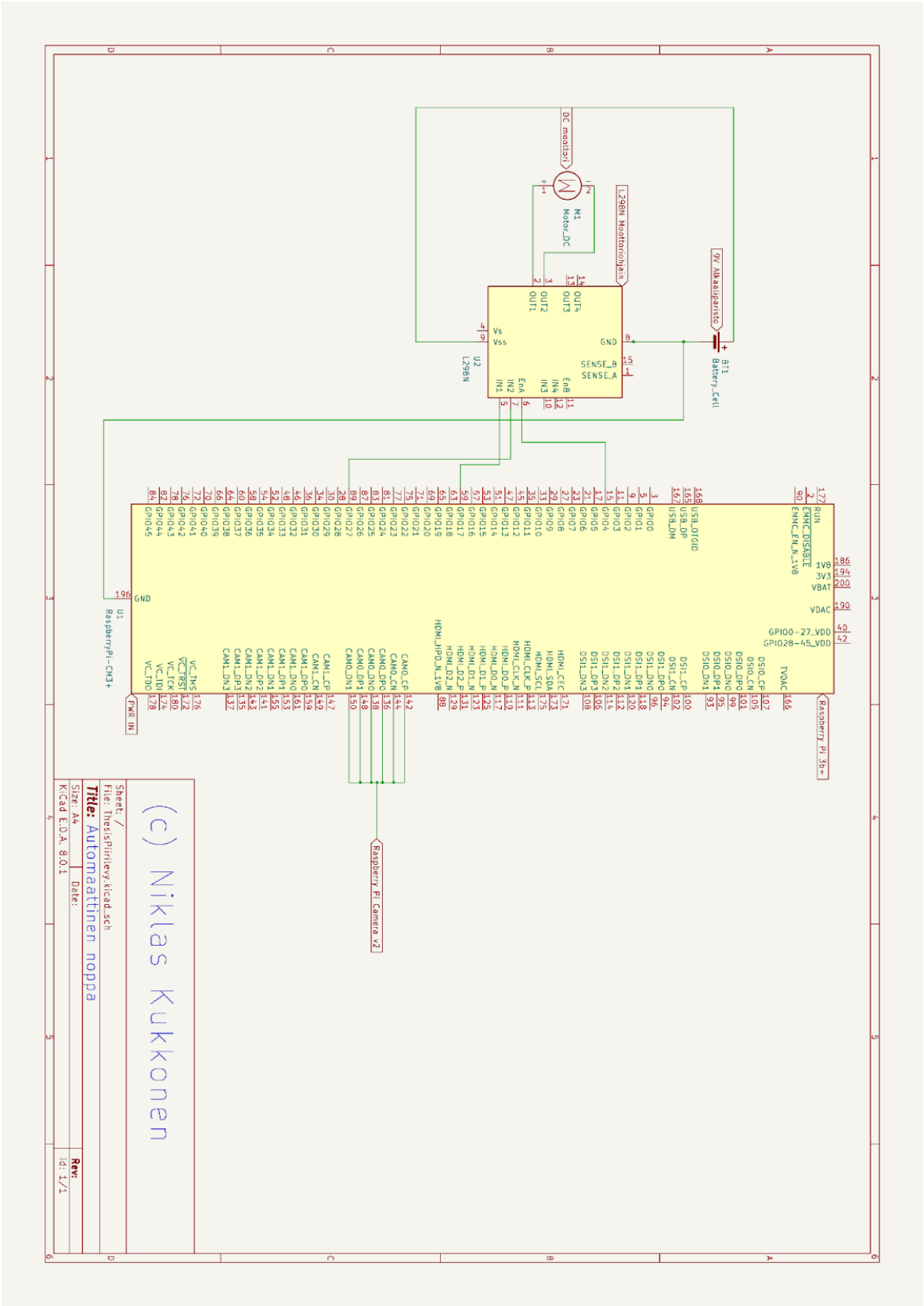
42 Control Raspberry Pi GPIO Pins from Python. ICS. Verkkoaineisto. Saatavilla: https://www.ics.com/blog/control-raspberry-pi-gpio-pins-python. Luettu 28.4.2024.

43 Time access and conversions. Python. Verkkoaineisto. Saatavilla: https://docs.python.org/2/library/time.html. Luettu 28.4.2024.

44 L298N Dual H-Bridge Motor Driver. Handsontec. Verkkoaineisto. Saatavilla: https://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf. Luettu 28.4.2024.

45 Collvy. Controlling DC Motor with Raspberry Pi and L298N motor driver. Video. Youtube. 8.8.2022. Saatavilla: https://www.youtube.com/watch?v=OV0S_3KMj2A. Katsottu 28.4.2024.

Laitteen kytkentäkaavio



Verkkosovelluksen koodi

main.py

```
import RPi.GPIO as GPIO
from flask import Flask, render_template, Response, request, redirect,
url_for
import time
import json
import cv2
import pytesseract
from pytesseract import Output
import numpy as np
import imutils

GPIO.setwarnings(False)

app = Flask(__name__)

# Motor pins
in1 = 17
in2 = 27
en_a = 4

# Image And Mask Var

raw_image = ""
mask_image = cv2.imread("Mask.jpg")
processed_image = ""

# tesseract Settings

custom_oem = "outputbase digits"
min_conf = 0


@app.route('/')
def index():
    return render_template('index.html')

#Function to turn the motor on for rolling the die
def roll_dice():
    try:
        # Set GPIO
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(in1, GPIO.OUT)
        GPIO.setup(in2, GPIO.OUT)
        GPIO.setup(en_a, GPIO.OUT)
```

```
        #Define motors power
        q = GPIO.PWM(en_a, 100)
        q.start(20)

        # Turn the motor on
        GPIO.output(in1, GPIO.HIGH)
        GPIO.output(in2, GPIO.LOW)

        # Wait 2 second
        time.sleep(2)

        # Turn the motor off
        GPIO.output(in1, GPIO.LOW)
        GPIO.output(in2, GPIO.LOW)

    finally:
        # Cleanup GPIO
        GPIO.cleanup()

#Function for taking picture of the die
def takePicture():
    #define camera and configurations
    picam2 = Picamera2()
    config = picam2.create_preview_configuration(main={"size": (1280,
720)}
    picam2.configure(config)

    #start preview and camera
    picam2.start_preview(Preview.QTGL)
    picam2.start()

    #set lens focus and wait 5 seconds
    picam2.set_controls({"AfMode": con-trols.AfModeEnum.Manual,"LensPosi-
tion": 5.0})
    time.sleep(5)

    #take a picture and stop preview and camera. Return image
    picam2.capture_file("picture.jpg")
    picam2.stop_preview()
    picam2.stop()
    return "picture.jpg"

#Binarize the image
def binarizing(image, a, b):
    return cv2.threshold(image, a, b, cv2.THRESH_BINARY)

#Comparing image to empty mask to find the die
```

```
def mask(mask, image):
    return cv2.subtract(mask, image)

#Grayscaling the image
def grayscale(image):
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

#Removing noise from the image
def noiseRemoval(image):
    import numpy as np
    kernel = np.ones((1, 1), np.uint8)
    image = cv2.dilate(image, kernel, iterations=1)
    kernel = np.ones((1, 1), np.uint8)
    image = cv2.erode(image, kernel, iterations=1)
    image = cv2.morphologyEx(image, cv2.MORPH_CLOSE, kernel)
    image = cv2.medianBlur(image, 3)
    return (image)

#Inverting the image colors
def invertImage(image):
    return cv2.bitwise_not(image)

#Tesseract processing
def pitess(image):
    # Perform OCR using Tesseract
    # Path to Tesseract executable
    pytesseract.pytesseract.tesseract_cmd = "/usr/bin/tesseract"
    results = pytesseract.image_to_data(
        image, output_type=Output.DICT, config=custom_oem)
    # print(type(results))
    return results

#Resize
def resize(img):
    height, width = img.shape[:2]
    res = cv2.resize(img, (2*width, 2*height), interpolation=cv2.IN-
TER_CUBIC)
    return res

#Crop
def crop(img, pix, pix2):
    height, width = img.shape
    center_x, center_y = width // 2, height // 2

    # Define the desired crop size (adjust as needed)
    crop_width, crop_height = pix, pix2

    # Calculate the top-left corner coordinates for cropping
```

```
    x = center_x - crop_width // 2
    y = center_y - crop_height // 2

    # Crop the image
    crop_img = img[y:y+crop_height, x:x+crop_width]

    # Save the cropped image to a file
    cv2.imwrite("temp/Cropped_Image.jpg", crop_img)
    return crop_img

#Degine font for image
def thin_font(image):
    import numpy as np
    image = cv2.bitwise_not(image)
    kernel = np.ones((2, 2), np.uint8)
    image = cv2.erode(image, kernel, iterations=5)
    image = cv2.bitwise_not(image)
    return (image)

#Main route where functions are called and the program runs on
@app.route("/roll/", methods=['POST'])
def main():
    roll_dice()

    time.sleep(5)

    raw_image = cv2.imread(takePicture())
    cv2.imwrite("temp/rawImage.jpg", raw_image)

    maskresized = resize(mask_image)

    maskedImage = mask(maskresized, raw_image)
    cv2.imwrite("temp/masked.jpg", maskedImage)

    grayImage = grayscale(maskedImage)
    cv2.imwrite('temp/gray.jpg', grayImage)

    thresh, im_bw = binarizing(grayImage, 65, 600)
    cv2.imwrite("temp/bw_image.jpg", im_bw)

    edges = cv2.Canny(im_bw, 100, 200)
    cv2.imwrite("temp/edge.jpg", edges)

    kernel_size = 5
    kernel = np.ones((kernel_size, kernel_size), np.uint8)

    dilated_mask = cv2.dilate(edges, kernel, iterations=1)
```

```
    # Find contours in the edges
    contours, _ = cv2.findContours(
        dilated_mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cv2.imwrite("temp/dilated_mask.jpg", dilated_mask)

    min_contour_area = 30  # Example threshold
    filtered_contours = [
        c for c in contours if cv2.contourArea(c) > min_contour_area]

# Sort contours by area (largest first)
    contours = sorted(filtered_contours, key=cv2.contourArea, re-
verse=True)

# Assume the largest contour corresponds to the D20 die
    d20_contour = contours[0]

# Calculate the bounding rectangle for the D20 die
    x, y, w, h = cv2.boundingRect(d20_contour)

# Crop the D20 die from the original image
    d20_cropped = maskedImage[y:y+h, x:x+w]
    cv2.imwrite("temp/newmask.jpg", d20_cropped)

#   no_noise =
    dieComp = grayscale(d20_cropped)
    thresh, im_bw2 = binarizing(dieComp, 90, 600)
    cv2.imwrite("temp/bw_image.jpg", im_bw2)
    invertedImage = invertImage(im_bw2)
    cv2.imwrite("temp/inverted.jpg", invertedImage)
    nr = noiseRemoval(im_bw2)
    resized = resize(nr)
    resized = noiseRemoval(resized)
    crp = crop(nr, 120, 120)
    nr2 = noiseRemoval(crp)
    eroded_image = thin_font(nr2)
    cv2.imwrite("temp/asd.jpg", eroded_image)
    invertedImage2 = invertImage(eroded_image)
    cv2.imwrite("temp/inverted.jpg", invertedImage2)
    processed_image = invertedImage2
    cv2.imwrite("temp/Pimage.jpg", processed_image)

    results = []

    for angle in np.arange(0, 360, 15):
        rotated = imutils.rotate(processed_image, angle)
        results = pitess(rotated)
        cv2.imwrite("temp/rotated.jpg", rotated)
        if len(results["text"]) != 0:
```

```
            for i in range(0, len(results["text"])):
                # text = results["text"][i]
                # print("Text: {}".format(text))
                x = results["left"][i]
                y = results["top"][i]
                w = results["width"][i]
                h = results["height"][i]
                # extract the OCR text itself along with the confidence
of the
                # text localization
                text = results["text"][i]
                conf = int(results["conf"][i])
                # filter out weak confidence text localizations
                if conf > min_conf:
                    # display the confidence and text to our terminal
                    print("Confidence: {}".format(conf))
                    print("Text: {}".format(text))
                    print("")
                    # strip out non-ASCII text so we can draw the text on
the image
                    # using OpenCV, then draw a bounding box around the
text along
                    # with the text itself
                    text = "".join(
                        [c if ord(c) < 128 else "" for c in
text]).strip()
                    cv2.rectangle(d20_cropped, (x, y),
                                  (x + w, y + h), (0, 255, 0), 2)
                    cv2.putText(d20_cropped, text, (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX,
                                1.2, (0, 0, 255), 3)
                    cv2.imwrite("temp/Image.jpg", d20_cropped)
    textjason = results["text"][i]
    return json.dumps({"roll_message": textjason})

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port="5000")
```

index.html

```
<html>
<head>
<meta name="viewport" content="width=device-widrh" />
<title>Automatic Dice</title>
</head>
    <body>
    <center>
    <h1>Automatic Dice<h1>
    <br>
    <br>
    <form id="rollForm" action="/roll" method="post">
        <button name="rollBtn" type="submit">Roll Dice!</button>
    </form>
    <br>
    <h3 id="numberResult"></h3>
    </center>

    <script>
        //Function to handle form submission
    document.getElementById("rollForm").addEventListener("submit", func-
tion (event) {
        event.preventDefault(); //precent the default form submission be-
haviour

        //Make an AJAX request to the /roll/ route
        fetch("/roll/", {
            method: "POST",
        })
        .then(response => response.json()) //Parse the response as JSON
        .tehn(data => {
            //Update the displayed random number
            document.getElementById("numberResult").textContent =
data.roll_message;
        })
        .catch(error => {
            console.error("Error fetching data:", error);
        });
    });
    </script>
    </body>
</html>
```

Vaihtoehtoinen verkkosovelluksen koodi

main.py

```
import RPi.GPIO as GPIO
import cv2
import numpy
from flask import Flask, render_template, send_from_directory, jsonify,
Response, request, redirect, url_for
import time
from picamera2 import Picamera2
from libcamera import controls
import os


# GPIO and Flask setup
GPIO.setwarnings(False)
app = Flask(__name__)

# Motor pins
in1 = 17
in2 = 27
en_a = 4

# Static file paths
MASK_PATH = "Images/mask.jpg"
CAM_IMAGE = "temp/rolledDice.jpg"
REF_PATH = "Images/ref.jpg"

# This is the route for the index, tells flask to render the index.html
file
@app.route('/')
def index():
    return render_template('index.html')

# This is the route for the roll, tells flask to run the roll_dice func-
tion
# and then take a picture and then run the precessing_logic function
@app.route("/roll/", methods=['POST'])
def main():

    roll_dice()

    time.sleep(3)

    takePicture()

    time.sleep(2)
```

```
    final_Location = precessing_logic()
    return jsonify({'url': final_Location})

# This is the route for the image, tells flask where to pull the image
from
@app.route('/image/<path:filename>')
def serve_image(filename):
    # ***!!!change to the folder where the image is saved!!!***
    return send_from_directory('/home/niklask/Thesis/temp/', filename)

# Main Image Processing Logic for finding and cutting out the die
def precessing_logic():

    # Main Image processing
    image = read_image(CAM_IMAGE)
    maskImage = read_image(MASK_PATH)
    save_image(maskImage, "temp/maskImage.jpg")
    masked = mask_image(image, maskImage)
    save_image(masked, "temp/masked.jpg")
    gray = gray_image(masked)

    # Quick Ref Image Processing
    ref_mod = read_image(REF_PATH)
    gray_ref = gray_image(ref_mod)
    save_image(gray_ref, "temp/gray_ref.jpg")

    # Find and cutout die
    top, bottom = find_die(gray, gray_ref)
    cutout = cutout_die(image, top, bottom)
    save_image(cutout, "temp/rolledDice.jpg")  # pick a good folder
    # change name based on saved image name above DO NOT TOUCH /image/
    return "/image/rolledDice.jpg"

# Find the image size
def find_image_size(image):
    height = numpy.size(image, 0)
    width = numpy.size(image, 1)
    return height, width

# Scale the image (upsacle or downscale based in scale_percent)
def scale_image(image, scale_percent):
    height, width = find_image_size(image)
    height = int(height * scale_percent / 100)
    width = int(width * scale_percent / 100)
    dim = (width, height)
    resized_image = cv2.resize(image, dim, interpolation=cv2.INTER_AREA)
    return resized_image
```

```
# Convert the image DATA to gray scale
def gray_image(image):
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    return gray_image

# Read the image from the path
def read_image(image_path):
    image = cv2.imread(image_path)
    return image

# Mask the image
def mask_image(image, mask):
    masked_image = cv2.subtract(mask, image)
    return masked_image

# Display the image in a poput window
def display_image(image, window_name="Image"):
    cv2.imshow(window_name, image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# Save the image
def save_image(image, image_path):
    cv2.imwrite(image_path, image)

# Find the die in the image, return location on ORIGINAL image
def find_die(image, reference_image):
    result = cv2.matchTemplate(image, reference_image, cv2.TM_CCO-
EFF_NORMED)
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)
    top_left = max_loc
    bottom_right = (top_left[0] + reference_image.shape[1],
                    top_left[1] + reference_image.shape[0])
    return top_left, bottom_right

# Draw a rectangle on the image based on top left and bottom right coor-
dinates
def draw_rectangle(image, top_left, bottom_right):
    cv2.rectangle(image, top_left, bottom_right, (0, 255, 0), 2)
    return image

# Cutout the die from the image
def cutout_die(image, top_left, bottom_right):
    cutout = image[top_left[1]:bottom_right[1], top_left[0]:bot-
tom_right[0]]
    return cutout

# Threshold the image
```

```
def threshold_image(image, threshold):
    _, threshold_image = cv2.threshold(
        image, threshold, 255, cv2.THRESH_BINARY)
    return threshold_image

# Remove glare from the image
def remove_glare(image):
    mask = cv2.inRange(image, (200, 200, 200), (255, 255, 255))
    result = cv2.inpaint(image, mask, 3, cv2.INPAINT_TELEA)
    return result

# Function to Roll the die
def roll_dice():
    try:
        # Set GPIO
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(in1, GPIO.OUT)
        GPIO.setup(in2, GPIO.OUT)
        GPIO.setup(en_a, GPIO.OUT)

        q = GPIO.PWM(en_a, 100)
        q.start(20)

        # Turn the motor on
        GPIO.output(in1, GPIO.HIGH)
        GPIO.output(in2, GPIO.LOW)

        # Wait 2 second
        time.sleep(2)

        # Turn the motor off
        GPIO.output(in1, GPIO.LOW)
        GPIO.output(in2, GPIO.LOW)

    finally:
        # Cleanup GPIO
        GPIO.cleanup()


# Function to take the picture and save it in temp folder with the name
"rolledDice.jpg"
def takePicture():
    # Taking a photo with picam
    # Save image to CAM_IMAGE location
    os.chdir('/home/niklask/Thesis/temp')
    picam2 = Picamera2()
    config = picam2.create_preview_configuration(main={"size": (1280,
960)})
```

```
    picam2.configure(config)
    picam2.start()
    picam2.set_controls({"AfMode": controls.AfModeEnum.Manual,"LensPosi-
tion": 5.0})
    time.sleep(2)
    picam2.capture_file("rolledDice.jpg")
    picam2.stop()
    picam2.close()
    os.chdir('/home/niklask/Thesis')
    return 0

if __name__ == "__main__":
    app.run(debug=True, host='0.0.0.0', port="5000")
```

index.html

```
<html>
<head>
<meta name="viewport" content="width=device-widrh" />
<title>Automatic Dice</title>
</head>
<body>
<center>
<h1>Automatic Dice<h1>
<br>
<br>
<img id="diceImage" src="" alt="Dice Image">
<br>
<br>
<br>
<form id="rollForm" action="/roll" method="post">
    <button name="rollBtn" type="submit">Roll Dice!</button>
</form>
</center>

<script>
// Function to handle form submission
document.getElementById("rollForm").addEventListener("submit", function
(event) {
    event.preventDefault(); // Prevent the default form submission behav-
ior

    // Make an AJAX request to the /roll/ route
    fetch("/roll/", {
        method: "POST",
    })
    .then(response => response.json()) // Parse the JSON from the re-
sponse
```

```
    .then(data => {
        // Update the displayed image
        var img = document.getElementById("diceImage");
        img.src = data.url + '?' + new Date().getTime(); // Append the
current timestamp as a query parameter
    })
    .catch(error => {
        console.error("Error fetching data:", error);
    });
});
</script>
</body>
</html>
```