

Ryhmä 23: Ihmistä seuraava Creeper

Jäsenet/Members:

Nimi	Pääaine	<p> </p>
Niklas Ritalahti	Elektroniikka ja sähkötekniikka	
Silja Heinonen	Elektroniikka ja sähkötekniikka	
Matti Kivelä	Elektroniikka ja sähkötekniikka	
Leevi Leinonen	Elektroniikka ja sähkötekniikka	

Asistentti: Justus Ojala

PROJEKTIVIDEO (Youtube)



Keijo Kriipperi.zip

Keijon CAD-tiedostot

1. Yleistä
2. Tavoitteet
3. Suunnittelu
 - 3.1. Paperiluonnos
 - 3.2. Pahviprototyppi
 - 3.3. 3D-mallinnus
4. Rakennusvaihe
 - 4.1. 3D-tulostus
 - 4.2. Testirakentelu
 - 4.3. Juottaminen
 - 4.4. Laserleikkaus
 - 4.5. Kasaus
5. Kytkentä- ja lohkokaaviot
6. Ohjelmakoodi
 - 6.1. ESP32
 - 6.2. ESP32-CAM & objektintunnistus
7. Osat ja materiaalit
 - 7.1. Komponentit
 - 7.2. Liittimet
 - 7.3. Kiinnikeet
8. Ongelmat & ratkaisut
 - 8.1. 3D-printauksen ongelmat
 - 8.1.1. Printteri hajosi
 - 8.1.2. Printit epäonnistuivat todella paljon ja todella usein
 - 8.1.2.1. "Liftaus"
 - 8.1.2.2. Suutinkoko
 - 8.1.2.3. Printtiresoluutio
 - 8.1.2.4. Printtimateriaali
 - 8.1.2.5. Osa printteistä hajosi kasatessa
 - 8.1.3. Printattujen osien suunnittelu
 - 8.1.3.1. Puristussovitus
 - 8.1.3.2. Muotolujitteet
 - 8.2. Muut ongelmat
 - 8.2.1. Ruuviliitokset
 - 8.2.2. Diagonaaliseesti asetetut moottorit eivät kääntäneet Keijoaa toiseen suuntaan
 - 8.2.3. Virranjako komponentteihin
 - 8.2.4. ESP32CAM tarkkuus ihmisen tunnistuksessa
 - 8.2.5. Laserleikkuri oli rikki
 9. Kehittävää
 - 9.1. Liitostyyppit
 - 9.1.1. Muovi-muovi -liitokset
 - 9.1.2. Akryyli-akryyli -liitokset
 - 9.2. Tunnistustarkkuus
 - 9.3. Servon valinta
 - 9.4. Ohjelmakoodi
 - 9.5. Johtosarjat
 - 9.6. Lineaariregulaattorin jäähdytys
 - 9.7. Piirien suojaus
 - 9.8. Helposti vaihdettava akkumoduuli
 - 9.9. Nappulat ja kytkimet
 - 9.9.1. Virtakytkin
 - 9.9.2. Reset ja boot -nappulat ESP32:lle
 - 9.10. USB-jatkojohto
 - 9.11. Muuta kivaa
 10. Lähteet
 - 10.1. Kirjaimelliset lähteet
 - 10.2. Ei-kirjaimelliset lähteet



1. Yleistä

Kurssiprojektimme on Minecraft-videopelistä tuttu Creeper-hahmo, joka tunnistaa ihmisen ja lähtee seuraamaan sitä. Idea projektiin saatiin siitä, että meillä oli syksyllä fuksiryhmän yhteinen Minecraft-serveri, jossa pelasimme yhdessä pelin läpi.

Keijo Kriipperi on noin 50 cm korkea figuuri, joka liikkuu neljällä renkaalla. Jokaisella renkaalla on oma harjallinen DC-moottori, joita ohjataan kahdella kaksikanavaisella moottorinohjaimella. Keijo pystyy liikkumaan eteenpäin ja käännytymään oikealle ja vasemmalle tankkikäänöksellä. Keijon "silmät" ovat ESP32-CAM ja ultraäänisensori. ESP32-CAM:ssa on ihmisen tunnistukseen tekoälymalli, jonka perusteella Keijo käantää servomoottorilla varustettua päättään ja tekee päätöksen liikkumisesta. Ultraäänisensori esittää törmäykset eteenpäin liikuessa. Ohjautuminen on siis täysin autonomista, sillä kaikki liikkumiseen liittyvä tapahtuu Keijon omassa koodissa eikä siihen puituta ulkopuolelta. Virtalähteenä Keijo käyttää LiPo-akkuja. Keijon ulkokuori on akryylia, jonka päälle on liimattua paperinen kuviointi. Keijo on valmistettu useita osia 3D-mallintamalla ja -tulostamalla.

	<p>Valmis Keijo</p> <p>Ominaisuudet, kuten autonomisuus, toimivat halutulla tavalla. Onnistuimme kurssin aikana tavoitteisamme, ja valmis Kriipperi toimii kuten suunnitellessa haluimme.</p> <p>Jätimme Keijoon takapuolelle kurkistusalueen, josta pääsee tarkkailemaan akryylin läpi sen sisällä olevia komponentteja.</p> <p>& a m p; lt; p & a m p; gt; ; & a m p; lt; /p & a m p; gt;</p> <p>V id e o K ei jo n to i m in n a sta</p> 
--	---

2. Tavoitteet

- Saada Kejo sihisemään ja pitämään räjähdyssämenen
- Välkkymään ja "räjähtämään" LED-valojen avulla
- Onnistua ihmisen tunnistuksesta ja seuraamisesta

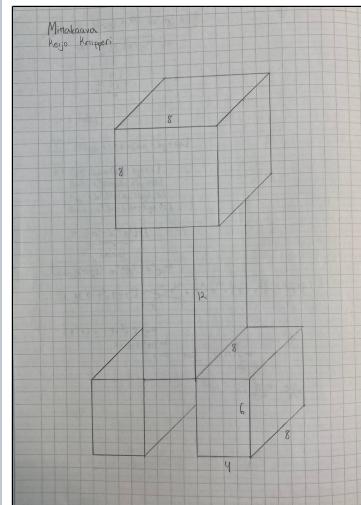
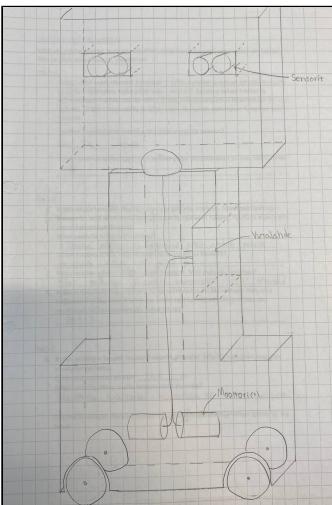
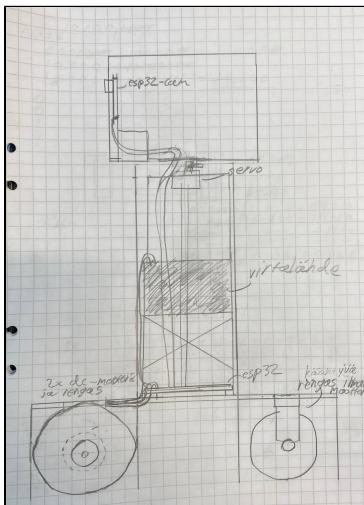
- Oppia elektronikasta
- Harjoitella mallintamista ja printtaamista
- Tutustua erilaisiin komponentteihin
- Koodata
- Pitää hauskaa

3. Suunnittelu

3.1. Paperiluonnos

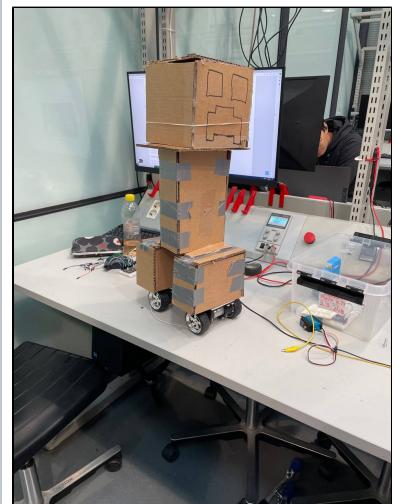
Keijon suunnittelu alkoi luennolla, kun tehtävään oli piirtää raakasuunnitelma paperille. Malleja tuli useita, joista tässä on muutamia:

(Parempi tarkkuus klikkaessa)



Oikeanpuoleisin kuva on ensimmäinen oikeassa mittakaavassa oleva luonnos. Mittakaava on sama kuin Minecraft-videopelin Creeperillä.

3.2. Pahviprototyppi



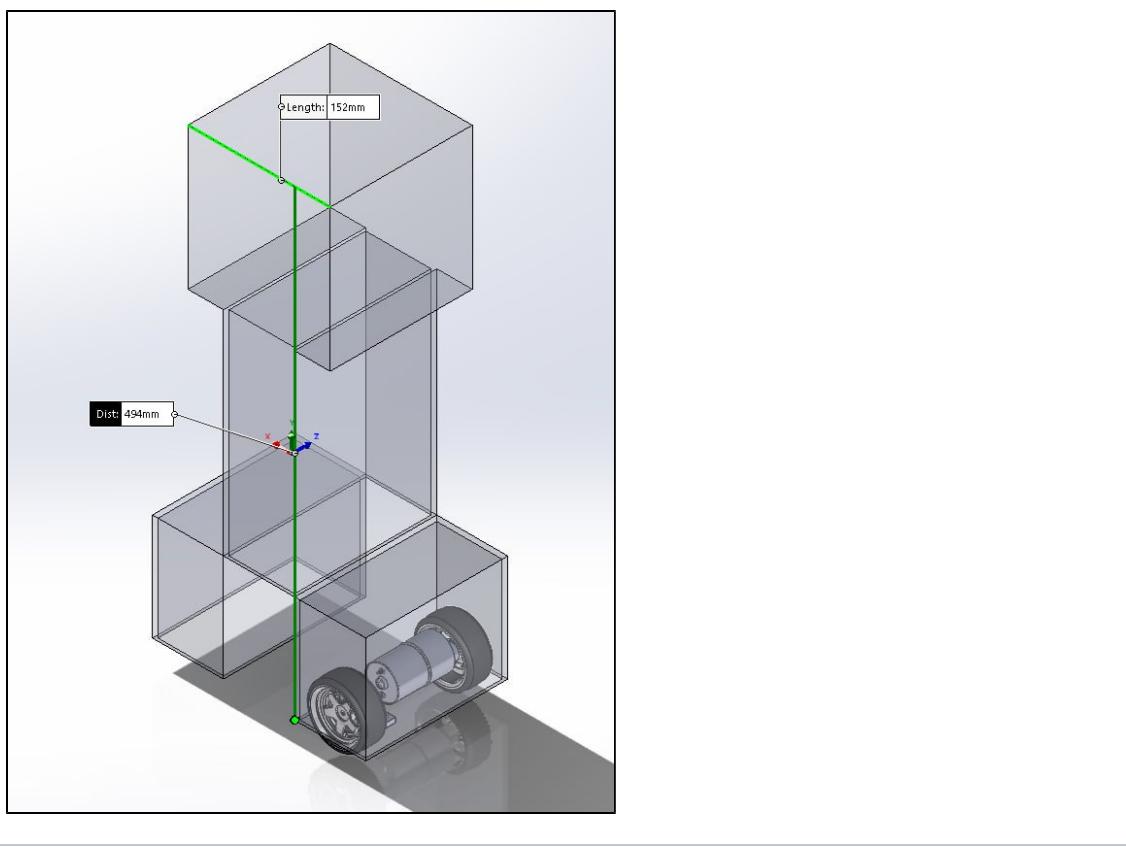
Kurssin aikana oli tehtävään mallintaa projektityötä myös prototyppinä. Ryhmämme väsäsi sitä varten pahvi-teippi-taktikalla ihka oikean Kriipperin. Mallinnus ei ole täysin tarkka lopullisen suunnitelman mukaisesti, sillä ajatuksena oli hahmottaa vain Keijon oikea koko paremmin.

Esimerkiksi renkaat tulevat lopulta Keijon sisälle toisin kuin mitä pahviprototypissä. Prototypin tekemisestä oli hyötyä projektin kannalta, sillä reaalimailman malli auttaa huomaamaan mahdollisia epäkohtia. Pahvimalli saikin meidät hieman huolestumaan Keijon pystyssä pysymisestä. Se on osasyy siihen, miksi koodiin lisättiin moottoreiden kiihyttäminen sen sijaan, että ne lähtisivät heti täydelle teholle.

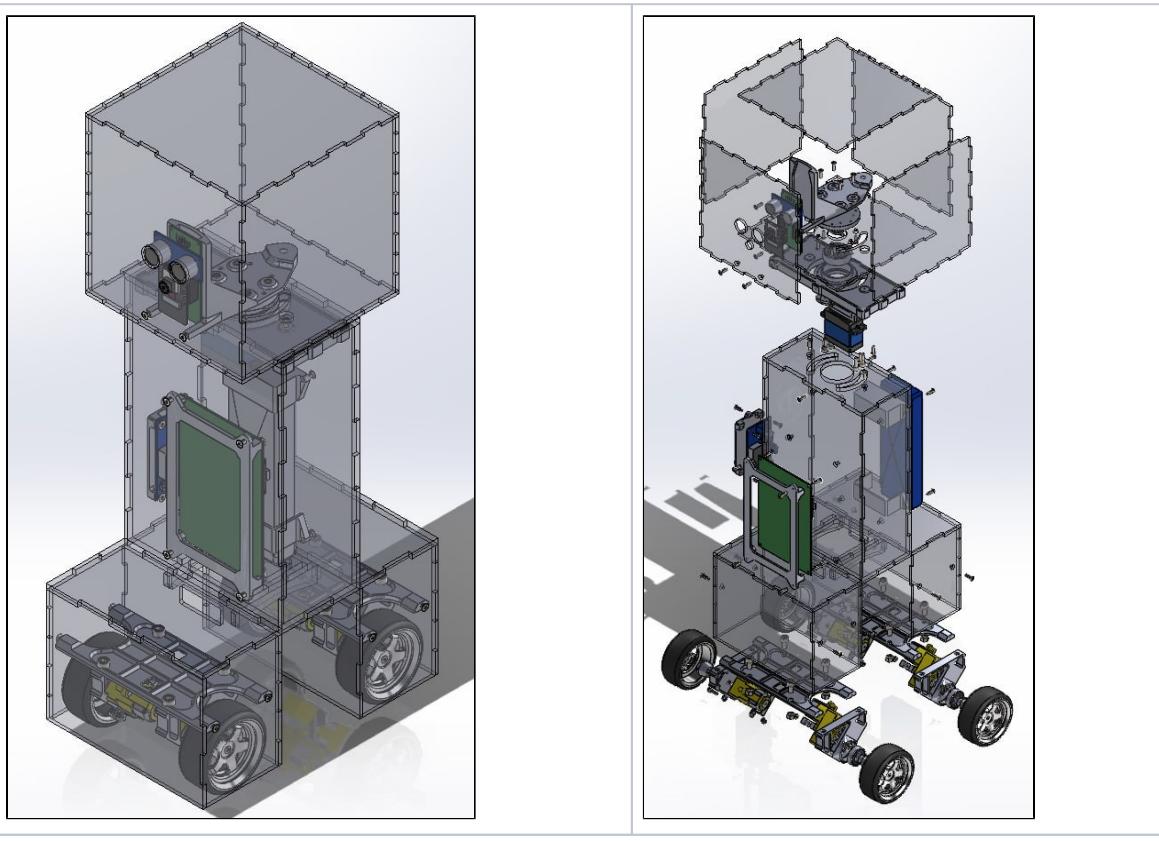
Pahvi-Keijo ei valitettavasti selvinnyt itse näkemään valmista Keijoaa mahdollisen sabotaasin vuoksi. Pahvinen versio oli siis mystisesti kadonnut kevään aikana.

3.3. 3D-mallinnus

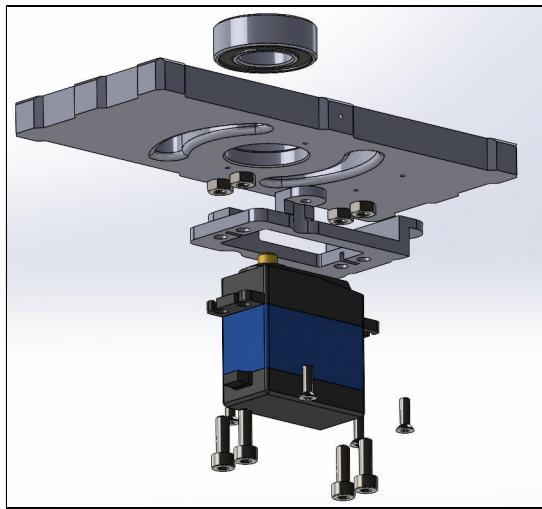
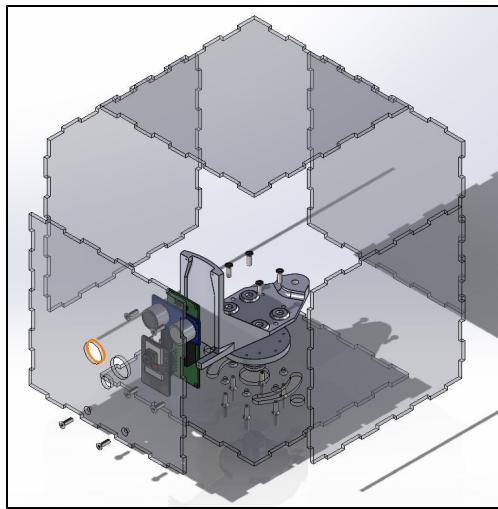
Piirrettyjen ja askarreltujen luonnosten lisäksi kurssin aikana kehittyi täydellinen SolidWorksilla^[9] rakennettu 3D-mallinnus Keijon jokaisesta osasta oikeissa mittasuhteissa tarkasti mitattuna.



Alunperin tankkikäänöksellä käännyvä suunitelma, siis Keijossa oli tarkoitus olla yhteensä vain 2 moottoria, nykyisen 4 sijaan.



Tässä mallissa näkyy käytännössä jokainen osa, joka Keijoon tulee. Sen vieressä malli, miten Keijo kasataan ja osat tulevat kiinni.

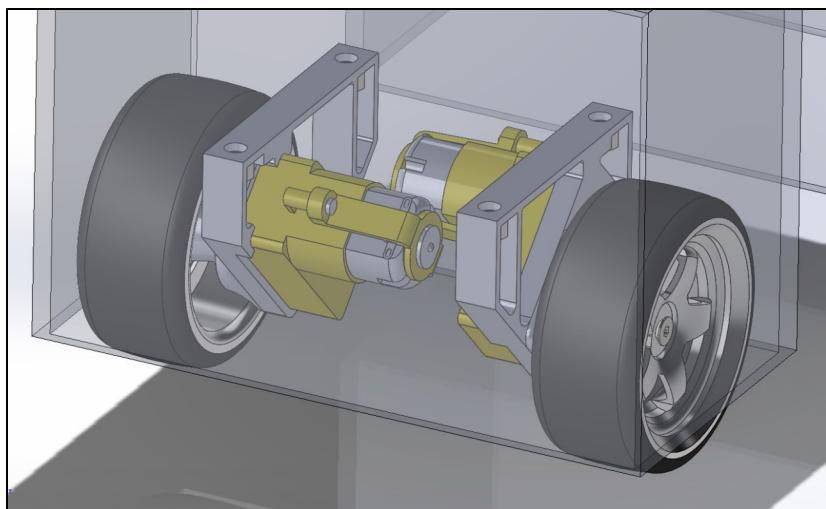


Kejon pään osien kiinnittäminen, ja vieressä päätä pyörityväni servosysteeminen kokoamistapa. Pään sisällä on ultraääänisensori ja ESP32Cam.

4. Rakennusvaihe

4.1. 3D-tulostus

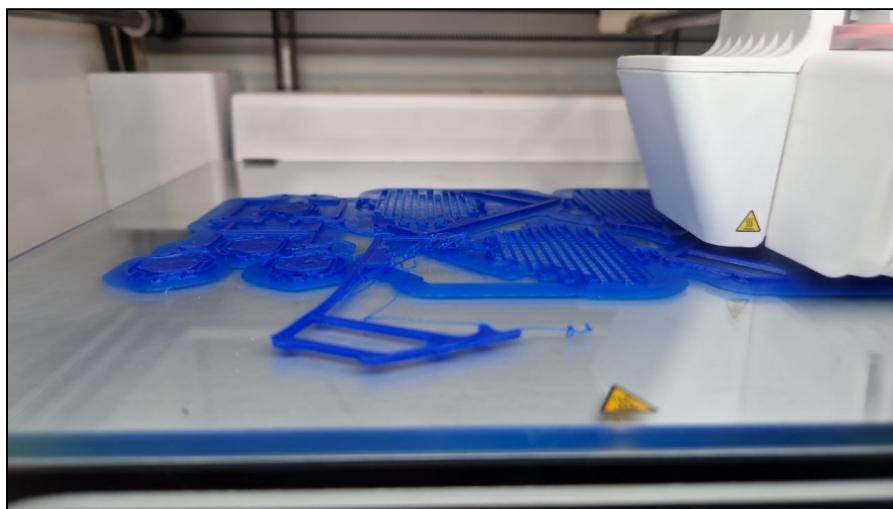
Aloitimme rakentamisen 3D tulostamalla adapterit renkaista moottoreihin, jonka jälkeen tulostimme moottoripidikkeet. Jouduimme tulostamaan useammat, koska kaikki tulostimet ja asetukset eivät olleet tarpeeksi tarkkoja meille. Kokeilun jälkeen löysimme sopivan koneen ja tulostusmateriaalin, joka oli tarpeeksi tarkkaa meille.



Ensimmäinen mallinnus siitä, miltä neljän moottorin versio näyttäisi toisessa jalassa.



Ensimmäiset rengasadapterit tulostumassa. Ne eivät kuitenkaan olleet sopivan kokoisia. Oikealla oleva sininen versio oli jo sopiva.

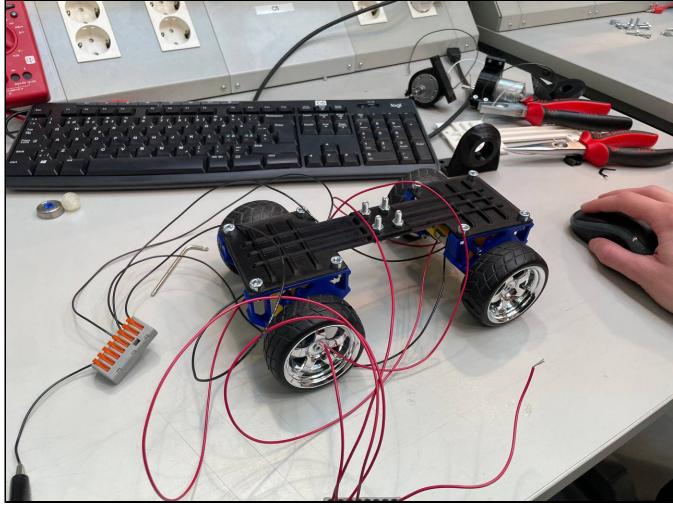


Moottorinpidikkeiden tulostuksen alkuvaiheet. Näiden tulostus kesti yli 10 tuntia. Moni meni pieleen, mutta lopulta onnistui. Niiden piti sopia juuri eikä melkein.

3D-tulostus oli yksi kurssin eniten aikaa vievistä välivaiheista, sillä tulostusajat olivat pitkiä ja tulostimet olivat usein varattuja. Lisäksi pilalle menneet printit pitkittivät projektia. Tulevaisuudessa tällaisessa projektissa koneistaminen voisi olla oivallinen vaihtoehto joidenkin osien kanssa.

Pääpiirteisesti printtaus kuitenkin oli yksi parhaista ja tehokkaimmista tavoista saada halutunlaisia osia pajaprojektiin.

4.2. Testirakentelu



Tällä alustalla testasimme käännylyä ja yleisesti liikkumista, ennen kuin jalat olivat valmiit. Suuren johtomääärän takia piti miettiä, miten johtoja pitää testiajonaikana. Myöhäisemmässä rakennusvaiheessa päästiin luopumaan testialustasta ja kiinnittämään renkaat lopullisiin pidikkeisiin.

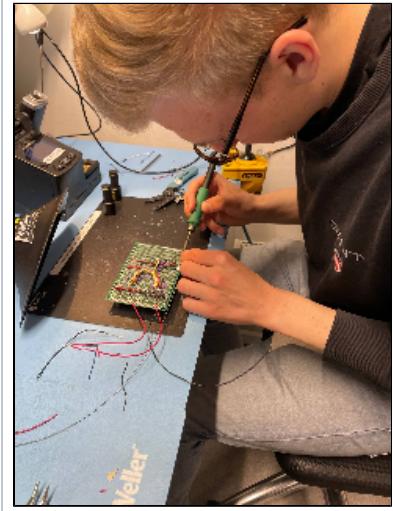
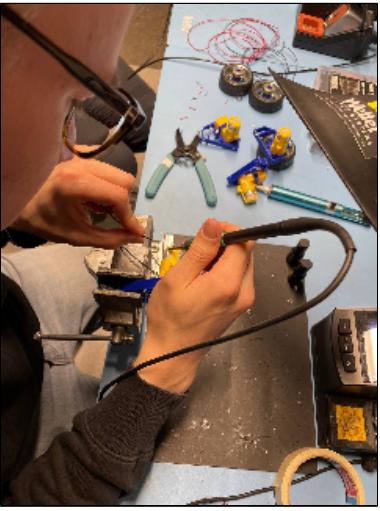
Osana testirakentelua oli myös diagonaalisten vetävien rengaiden testaamista. Se oli täysi farssi, sillä Keijo ei suostunut käänymään toiseen suuntaan olleenkaan. Eteen- ja taaksepäin liikkuminen kävi vaivattomasti, mutta tarvisimme ehdottomasti myös mahdollisuuden liikkua muuhinkin suuntiin. Tässä vaiheessa projektia tuli paljon takapakkia, sillä jouduimme suunnittelemaan täysin uuden tavan liikkua. Onneksemme löysimme sopivat neljä moottoria pajan varastosta, ja pystyimme aloittamaan neljän vetävän renkaan suunnittelun.

Testailu on tärkeä osa projektia, sillä mahdollisia suunnitteluvirheitä on vaikeaa huomata paperilla. Emme vieläkään tiedä, miksi alkuperäinen kahden vetävän renkaan suunnitelma ei toiminut. Jos emme olisi testanneet ajoissa, ei Keijo olisi valmistunut aikataulun mukaisesti.

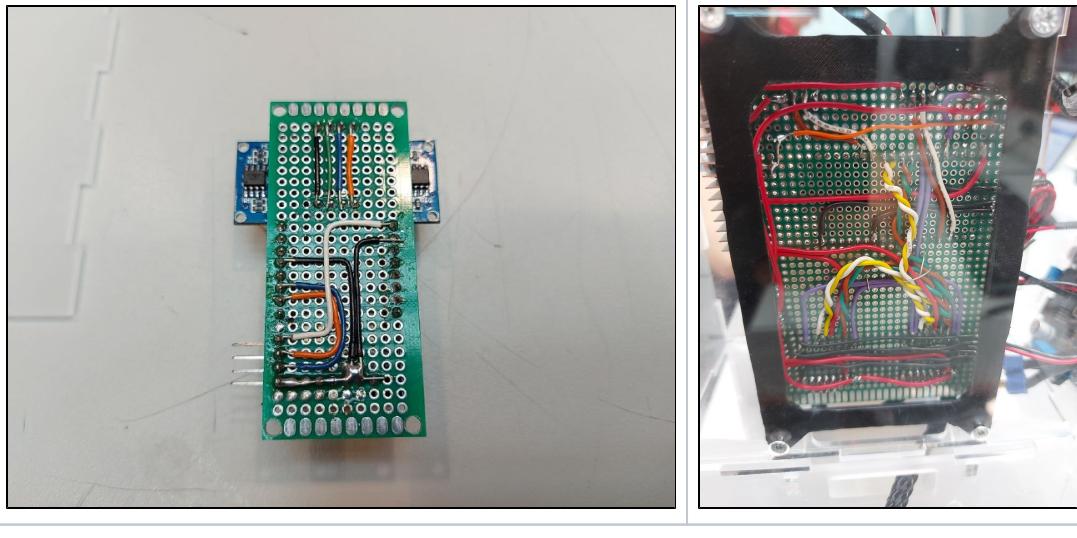
4.3. Juottaminen

Emme halunneet käyttää lopullisessa työssämme breadboard-typpistä ratkaisua, sillä johtojen irtaaminen ja mahdollisen katastrofin riski oli mielestämme liian suuri. Lisäksi tila Keijon sisällä ei olisi riittänyt isoihin johtoviritelmiin. Sen vuoksi päätimme jo kurssin alussa tehdä lopulliseen versioon protoboardin itse. Breadboardeja käytimme vain projektin alkuvaiheissa, kun lopullisista kytkennoista ei vielä ollut varmuutta. Siihen se oli erittäin hyvä vaihtoehto.

Rakenteluviiseen kuului paljon juottamista. Esimerkiksi johdot kiinnitettiin moottoreihin ja protoboardin jokainen johto on itse juotettu. Jälki ei aina ollut ensiluokkalista, mutta oikosulkutilanteilta vältyttiin.



Kuvissa tilannekuvia johtojen juottamisesta ja protoboardin viimeistelyistä



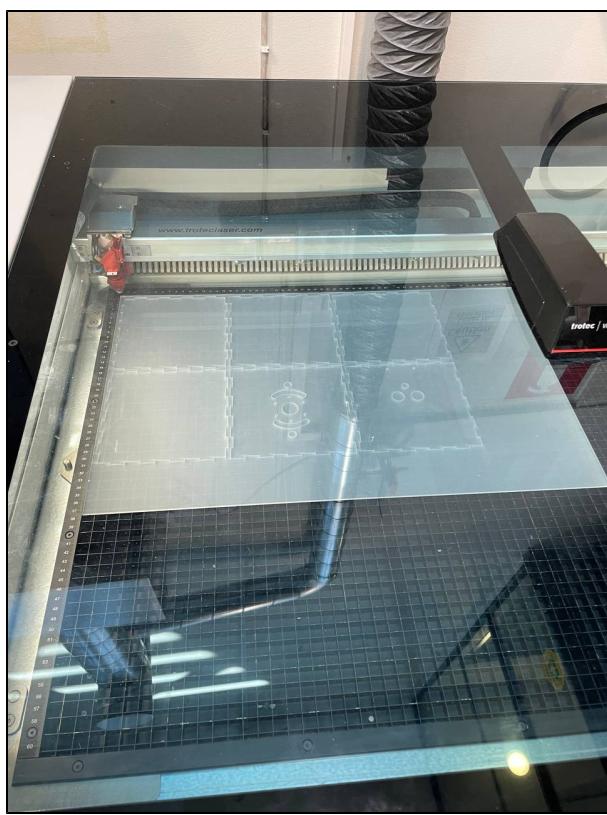
Kuvat valmiista protoboardeista. Vasemmalle ultraäänisensorin ja ESP32-CAMin, oikealla kaikkien muiden osien.

Haluimme tehdä protoboardin myös siksi, että jokainen meistä saisi lisää kokemusta tekniikan perusteista. Osalle kolvaaminen oli tuttu juttu, mutta osa meistä oli ensimmäisiä kertoja tekemässä. Kurssin aikana jokainen pääsi harjoittamaan taitoa, josta on varmasti hyötyä myös myöhemmässä elämässä.

4.4. Laserleikkaus

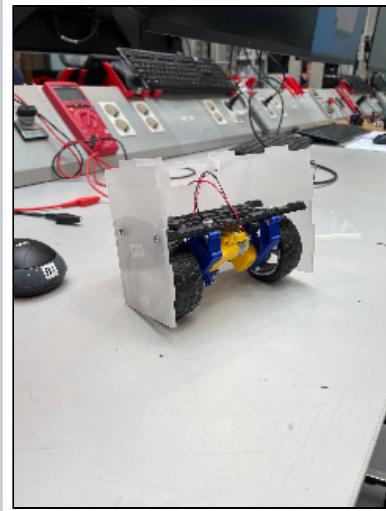
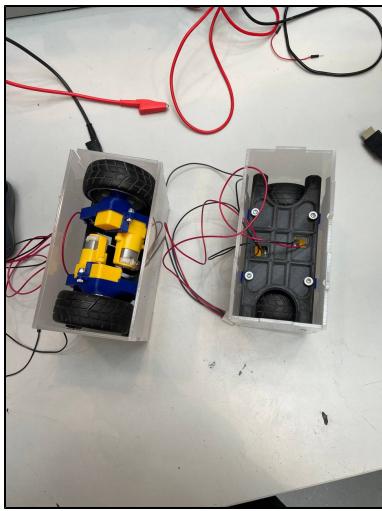
Assistantin avustuksella leikattiin läpinäkyvästä akryylistä Keijon kuoret. Tämä(kään) ei mennyt ykkösellä purkkiin, sillä muutama osa oli leikkautunut väärän paksuisille akryylilevyille. Uudelleen leikkaamalla saatin ongelma korjattua, ja oikeanlaiset palat leikkattua.

Kuoren liitännät ovat sinkkaliitoksia, joten Keijolle on tarvittaessa helppo suorittaa ruumiinavaus.



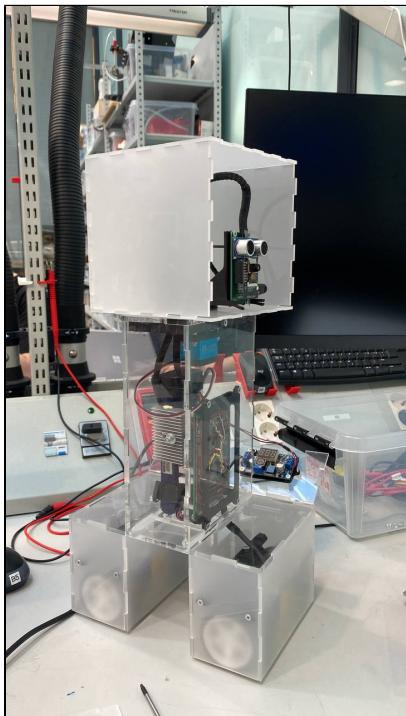
Keijon pään ulkokuori leikkautumassa

4.5. Kasaus



Valmiiksi kootut jalat.

Laserleikatut kuoret oli helppo naksutella paikoilleen. Yksittäisiä osia, kuten 3D-printattuja kokonaisuuksia, oli pystytty yhdistelemään jo aikaisemmin. Loppukasaus koostui lähinnä 3D-osien ja komponenttien ruuvaamisesta kuoriin, jolloin Keijo Kripperi saatiin jopa näyttämään Creeperiltä. Joitain liitoksia jouduttiin vahvistamaan kuumaliimalla, jotta romahtamisvaaralta välttyttäisiin.



Keijo ennen vaatteita

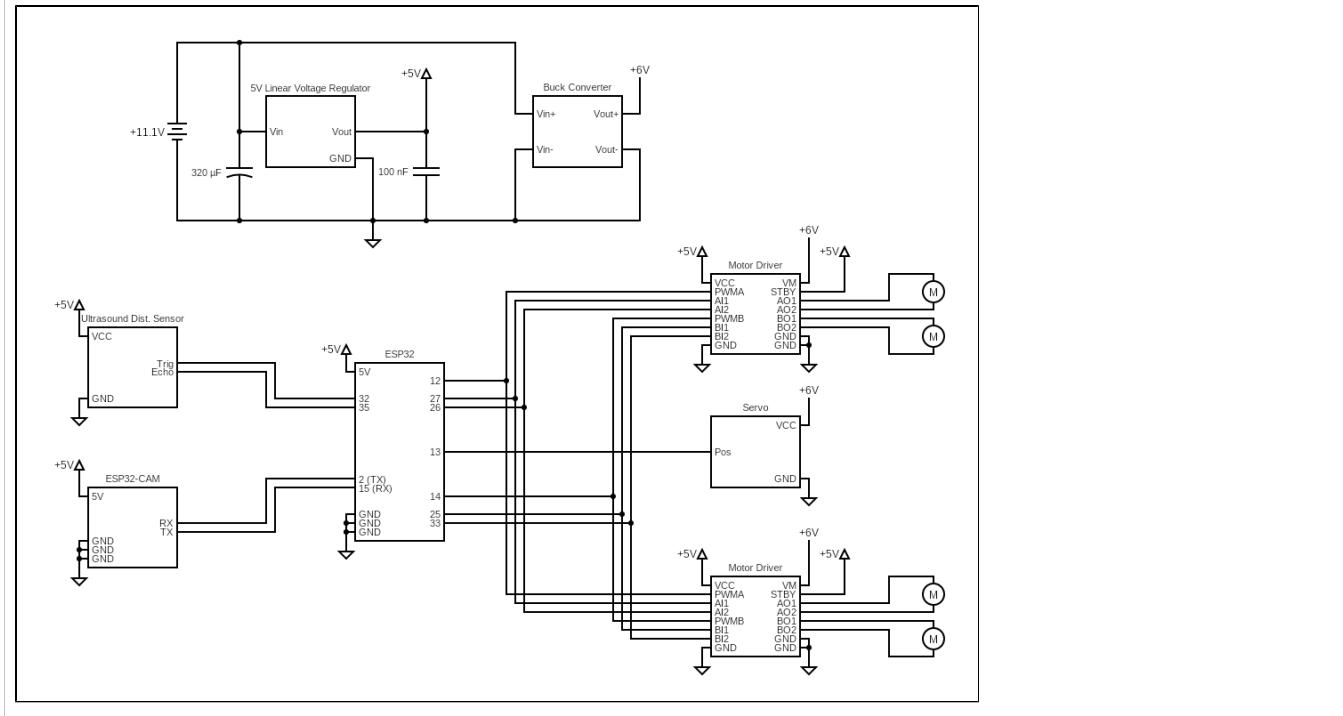
Keijo vaatteilla

Viimeisenä vaiheena kasausta tulostimme ja leikkasimme paperista Keijolle Creeperin pikselit, jotta hänet tunnistaa helpommin. Paperit liimattiin spray-liimalla kehoon kiinni.

Suosittelemme muita ryhmää aloittamaan lopullisen mallin rakentamisen ajoissa, jotta mahdollisiin ongelmatilanteisiin on varattu aikaa. Me kasasimme Keijon valmiiksi kurssin viimeisellä viikolla, jonka takia ilmestyneet ongelmat aiheuttivat turhaa stressiä ja harmaita hiukseja. Emme myöskään voineet osallistua lavaesitykseen, sillä päätöshetkellä Keijo ei ollut vielä tarpeeksi valmis ja emme uskoneet että olisimme kerenneet suunnitella esityksen.

5. Kytkentä- ja lohkokaaviot

Kaksi eri versiota hahmottamista varten. Ensimmäisen piirtoon on käytetty Circuit Diagram^[3]-selainsovellusta. Keskimäinen on rakennettu käyttäen GIMP^[4]-kuvankäsittelysovellusta ja netistä löytyneitä png-kuvatiedostoja, sillä valmiissa ohjelmistoissa ei ollut kaikkia osia saatavilla. Lohkokaavio GIMP.



6. Ohjelmakoodi

6.1. ESP32

Koko Keijon ohjaus tapahtuu autonomisesti koodin perusteella. Koodi on hyvin pitkälti kirjoitettu kokonaan itse, mutta joissakin kohdissa on lainattu valmista koodia muualta. Koodissa on käytössä kaksi kirjasto, joista toinen, ESP32Servo, pitää ladata library managerin kautta. Koodi on pyritty pitämään mahdollisimman yksinkertaisena, jotta se olisi helposti muokattavissa ja ymmärrettävissä. Koodiin on tehty useita funktioita eri toiminnoille ja ne on jaettu useammalle sivulle selkeyden vuoksi. Moottorien ohjausta varten tehdyt funktiot perustuvat moottoriohjainten valmistajan ohjeisiin^[2]. Ultaäänisensoria hyödyntävä distance -funktio on tehty valmiin ohjeen pohjalta^[5]. Servo funktiot muuttavat servoPos -muuttujan arvoa ja kirjoittavat sen servokirjaston valmiilla funktiolla. UART:ia käyttävässä offSet -funktiossa hyödynnetään YouTube-tutoriaalissa olevaa dekoderia^[6]. OffSet funktoit toimii siten, että se lähettää dataa ESP32-CAM:n bufferiin. Kun esp32cam bufferissa on dataa, se tietää, että ESP32 haluaa dataa ja lähettää sitä. ESP32 odottaa niin kauan, että se saa dataa, minkä jälkeen se dekoodaa saadun viestin ja muuttaa sen numeroksi.

Yksinkertaisesti selitettyyn pääaloopin alussa ESP32 pyytää ESP32-CAM:ta UART:n välityksellä tämän havaitseman koteenen sijainnin eron keskikohdasta. Sitten ESP32 tekee päätkösen siitä pitääkö servoaa käännytä johonkin suuntaan. Kun koteenen sijainnin ero keskikohdasta on riittävän pieni, tekee ESP32 päätkösen siitä, pitääkö sen käännytä, vai voiko se mennä eteenpäin. Tarkempi koodi ja sen funktioiden toiminta selviää helpoiten lukemalla koodia ja sen kommentteja.

ESP-32

```
//UART kirjasto ja serial portin määrittely
#include <HardwareSerial.h> //otetaan käyttöön tämä kirjasto, ei tarvitse ladata erikseen
HardwareSerial SerialPort(2); //käytetään UART2
const int camRes = 96; //kameran resoluutio pikseleinä x aksellilla

//Servo kirjasto ja pinni
#include <ESP32Servo.h> //otetaan käyttöön tämä kirjasto, tämä pitää kidata Arduino IDEn library managerin
kautta
#define SERVO_PIN 13
Servo servoMotor; //määritellään nimi servolle
const int turnAmount = 5; //tämä on paljonko servoaa käännetään yhdellä kerralla
const int maxAngle = 100; //tämä on servon maksimi kulma
int servoPos = maxAngle/2; //asetataan servoPos muuttuja puoleen väliin
```

```

//motoottoriohjaimen pinnit
#define AIN1 27
#define BIN1 25
#define AIN2 26
#define BIN2 33
#define PWMA 12
#define PWMB 14
//PWM kanavat joita käytetään
#define PWMA_ch 2
#define PWMB_ch 3

//ultraäänisensorin pinnit
#define TRIG 32
#define ECHO 35

void setup() {
    Serial.begin(115200); //avaa serial portin debugausta varten

    SerialPort.begin(115200, SERIAL_8N1, 15, 2); //avaa serial port 2 datan siirto varten ja käytetään pinnejä 15
ja 2

    //Moottoriohjainten AIN ja BIN pinnien määrittely output pinneiksi
    pinMode(AIN1, OUTPUT);
    pinMode(AIN2, OUTPUT);
    pinMode(BIN1, OUTPUT);
    pinMode(BIN2, OUTPUT);

    //Moottoriohjainten PWM pinninen määrittely, käytettään apuna valmiita ledc funktioita
    ledcSetup(PWMA_ch,10000,8);
    ledcSetup(PWMB_ch,10000,8);

    ledcAttachPin(PWMA,PWMA_ch);
    ledcAttachPin(PWMB,PWMB_ch);

    //Umääritetellään ultraäänisensorin pinnit
    pinMode(TRIG, OUTPUT);
    pinMode(ECHO, INPUT);

    //Määritellään servo, käytetään hyödyksi ESP32Servo kirjaston funktioita
    pinMode(SERVO_PIN, OUTPUT);
    servoMotor.attach(SERVO_PIN, 500, 2500);
    servoCenter(); //asetetaan servo keskikohtaan
}

void loop() {
    int off = offSet(); //pytää offsettiä espcamilta

    if (off == 0) { //jos off = 0, kamera ei näe mitään eikä tapahdu mitään
    }
    else if (off > (camRes/2)+1+5) { //jos off yli 5 pikseliä keskeltä vasemmalla niin käänny servoa vasemmalle,
+ 1 on sen takia koska espcamilla on lisätty arvoihin 1 uartia varten
        servoLeft();
    }
    else if (off < (camRes/2)+1-5) { //jos off yli 5 pikseliä keskeltä oikealla niin käänny servoa oikella, + 1
on sen takia koska espcamilla on lisätty arvoihin 1 uartia varten
        servoRight();
    }
    else { //muulloin objekti on lähes keskellä kuvaan jolloin voidaan liikuttaa autoa
        if (servoPos > ((maxAngle/2)+2*turnAmount)) { //jos servoPos isompi kuin keskikohta käännetään autoa
vasemmalle, toleranssina keskikohdalle on 2*yhden kerran servon käänötömäärä
            int t = (servoPos-(maxAngle/2))*4; //lasketaan servon kulman ja keskikohdan ja ero neljällä, kerroin on

```

```

määritelty testaamalla
    turnLeft(t); //kääntää autoa vasemmalle ajan t verran
    servoCenter(); //asetetaan servo keskelle
}
else if (servoPos < ((maxAngle/2)-2*turnAmount)) { //jos servoPos pienempi kuin keskikohta käännetään
autoa oikealla
    int t = ((maxAngle/2)-servoPos)*4; //lasketaan servon kulman ja keskikohdan ja ero neljällä, kerroin on
määritelty testaamalla
    turnRight(t); //kääntää autoa oikella ajan t verran
    servoCenter(); //asetetaan servo keskelle
}
else { //muulloin servo on lähes keskellä jolloin voidaan ajaa autoa eteenpäin
    servoCenter(); //asetetaan servo keskelle
    forward(1500); //ajetaan autoa 1500 ms eteenpäin, tämäkin arvo on määritelty sopivaksi testaamalla
}
}
}
}

```

Moottorinohjaus

```

//Mootoriohjaimia ohjataan säätmällä sen input pinnejen. Mootoreiden suunta määritetään säätmällä
moottoriohjaimien input pinnejen jännitettä ylös ja alas.
//Moottorien pyörimisnopeutta ohjataan pwm pinnejä säätmällä.
//Tältä sivulta https://learn.sparkfun.com/tutorials/tb6612fng-hookup-guide/all löytyy tarkemmin miten pinnejä
pitää säättää jotta saadaan haluttu pyörimissuunta moottoreille.
//Mootoriohjaimen Standby pinnejä ei ole kytketty esp32:n, vaan se on vedetty protoboardissa suoraan ylös

void forward(int t){ //Auto liikkuu eteenpäin, jos sen edessä ei ole esteitä
    digitalWrite(AIN1,HIGH);
    digitalWrite(AIN2,LOW);

    digitalWrite(BIN1,LOW);
    digitalWrite(BIN2,HIGH);

    if (distance() > 20 || distance() == 0) { //aloitetaan kiihyttämään moottoreita jos ultraäänisensori ei
havaitse mitään liian lähellä, 0 cm kelpaa, koska sensori antaa sen joksuks pitkällä matkalla
        accelerate();
    }

    int j = 0;
    int times = t/100;
    while (j < times && (distance() > 20 || distance() == 0)) { //ajetaan moottori ajan t verran ja samalla
katsoaan ultraäänisensorilla, ettei törmätä
        j++;
        delay(100);
    }
    stop(); //pysäytetään moottorit
}

void turnLeft(int t) { //Auto kääntyy oikealle
    digitalWrite(AIN1,LOW);
    digitalWrite(AIN2,HIGH);

    digitalWrite(BIN1,LOW);
    digitalWrite(BIN2,HIGH);

    accelerate(); //kiihytetään sulavasti, ettei liike olisi nykivää
    delay(t); //käännytään ajan t verran
    stop(); //pysäytetään moottorit
}

void turnRight(int t) { //Auto kääntyy vasemmalle
    digitalWrite(AIN1,HIGH);
    digitalWrite(AIN2,LOW);
}

```

```

digitalWrite(BIN1,HIGH);
digitalWrite(BIN2,LOW);

accelerate(); //kiihdytetään sulavasti, ettei liike olisi nykivää
delay(t); //käännytään ajan t verran
stop(); //pysäytetään moottorit
}

void stop() { //Pysäyttää moottorit
  digitalWrite(AIN1,LOW);
  digitalWrite(AIN2,LOW);

  digitalWrite(BIN1,LOW);
  digitalWrite(BIN2,LOW);
}

int distance() { //palauttaa matkan senttimetreinä lähimpään esteeseen ultraäänini sensorista
  //tämä on tehty tämän ohjeen pohjalta: https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/
  //koodi muokattu vähän poistamalla ylimääräiset asiat
  digitalWrite(TRIG, LOW); //asetaan ensin trigger alas
  delayMicroseconds(5); //odotetaan että varmasti alhaalla
  digitalWrite(TRIG, HIGH); //sitten asetetaan se ylös
  delayMicroseconds(10); //odotetaan että se ehtii varmasti lähetää
  digitalWrite(TRIG, LOW); //sitten asetetaan se takaisin alas

  pinMode(ECHO, INPUT); //tämä oli jostain syystä tutoriaalin koodissa, mutta tuskin tarvitsee koska määritelty jo aiemmin, en kuitenkaan enää kerkeä testa joten annetaan sen olla täällä
  long duration = pulseIn(ECHO, HIGH); //lasketaan kauanko kestää, että pulssi havaitaan
  long x = (duration/2) / 29.1; //jaetaan äänennopeudella ja saadaan etäisyys senttimetreinä
  return x;
}

void accelerate() { //Accelerate-funktio kiihdyttää moottoreita, jotta liike ei olisi nykivää
  nollasta sataan
  for (int i = 0; i < 255; i++) { //nostetaan for loopilla arvo nollasta maksimiin eli 255
    ledcWrite(PWMA_ch,i);
    ledcWrite(PWMB_ch,i);
    delay(10); //odotetaan joka välissä 10 ms, ettei kiihdytys ole liian nopeaa
  }
}

```

Servo

```
void servoRight() {    //kääntää servoa oikealla turnAmount verran
  if (servoPos >= 0+turnAmount){  //tarkistaa ettei servo ole jo oikella ääriasennossaan
    servoPos -= turnAmount;
    servoMotor.write(servoPos); //käytetään apuna servon kulman asettamiseen servo kirjaston funktioita
  }
}

void servoLeft() {    //kääntää servoa vasemmalla turnAmount verran
  if (servoPos <= maxAngle-turnAmount){  //tarkistaa ettei servo ole jo vasemmalla ääriasennossaan
    servoPos += turnAmount;
    servoMotor.write(servoPos); //käytetään apuna servon kulman asettamiseen servo kirjaston funktioita
  }
}

void servoCenter() {  //asettaan servon keskikohtaan
  servoPos = maxAngle/2;
  servoMotor.write(servoPos); //käytetään apuna servon kulman asettamiseen servo kirjaston funktioita
}
```

UART

```
int offSet() {    //pyytää espcamilta offsetin kohteseen jonka havaitsee
  //tässä apuna decoderin teossa on käytetty tästä tutoriaali https://www.youtube.com/watch?v=nSGnCT080d8
  int number = 0; //defaulttina 0, sama jos ei näe mitään

  char message[12];
  unsigned int message_pos = 0;

  while (SerialPort.available() > 0){SerialPort.read();} //tyhjentää oman bufferin, ettei siellä varmasti ole
mitään häiriötä

  SerialPort.println(1); //lähetää espcamin bufferiin dataa, jolloin se tietää että sen pitää lähettää nyt
dataa

  while(SerialPort.available() < 1){} //odotetaan niin kauan, että espcam lähetää jotain
delay(10); //odotetaan hetki, että kaikki data ehtii varmasti tulla

  int i = 0;
  while (SerialPort.available() > 0 && i < 12){  //decoodaan saadun datan ja muutetaan sen integeriksi, tämä on
suoraan videolta ja sen tarkempi toiminta kannattaa katso sieltä
    char inByte = SerialPort.read(); //luetaan dataa bufferista yksi tavu kerralla

    if (inByte != '\n'){  //jos ei ole viimeinen tavu, lisätään se message char arrayhin
      message[message_pos] = inByte;
      message_pos++;
    }
    else{
      message[message_pos] = '\0'; //lisätään lopetus merkki, jos oli viimeinen tavu
      number = atoi(message); //muutetaan integeriksi atoi funktiolla
      message_pos = 0;
    }
    i++;
  }
  return number; //palautetaan numero
}
```

6.2. ESP32-CAM & objektintunnistus

Keijossa on ESP32-CAM eli ESP32 kehityskortti, jossa on valmiaksi kytketty kamera. ESP32:n matala suorituskyky ja pieni muisti asettaa rajoituksia ja haasteita objektintunnistukselle. Projektissa ESP-CAM ei siis ole liitetty tietokoneeseen tai mihinkään muuhun, joka objektintunnistuksen tekisi, vaan kaikki tapahtuu siinä. Espcamin käyttämä ihmisen tunnistusmalli on tehty Edge Impulse palvelulla^[7]. Mallia varten on aluksi otettu useita erilaisia kuvia käytteään espcamia, jotta mallin harjoittamisessa käytetystä kuvat vastaisivat analysoitavia kuvia mahdollisimman paljon. Sitten kuvat on ladattu Edge Impulseen ja jokaiseen kuvaan on manuaalisesti merkitty laatikolla, missä kohdassa ihmisen on. Sen jälkeen luodaan Edge Impulsella koneoppimismalli. Tässä vaiheessa kuvien resoluutio pitää säättää pieneksi (96x96 pikseliä), jotta lopullinen malli mahtuu espcamiin. Sitten valmis ihmisen tunnistusmalli ja mukana tulevat kirjasto voidaan ladata esimerkiksi Arduino IDE:n. Tämä prosessi ei ole lopulta kovin mutkainen, mutta suhteellisen työläs. DroneBot Workshop -sivustolle on kirjoitettu objektiin tunnistuksesta ja Edge Impulsen käyttämisestä kattava teksti ja ohje^[8].

ESP32-CAM:n koodi on Edge Impulse kirjaston mukana tuleva esimerkkikoodi, joka antaa havaitut objektit ja niiden sijainnin. Koodi on tominnaltaan melko monimutkainen ja se käyttää useita kirjastoja. Sitä on muokattu siten, että se laskee havaitun objektiin sijainnin eron keskikohdasta ja lähettää sen ESP32:le, jos tämä on pyytänyt dataa. Dataa lähetetään vain pyydettäessä, jotta ESP32:n bufferi ei mene täyteen ja se saa aina tuoreimman tiedon. Koodiin on kommentoitu vain kohdat joihin on tehty itse muokkauksia.

ESP32-CAM

```
/* Edge Impulse Arduino examples
 * Copyright (c) 2022 EdgeImpulse Inc.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 */
/*
 * Includes -----
 */
#include <PersonDetection2_inferencing.h>
#include "edge-impulse-sdk/dsp/image/image.hpp"

#include "esp_camera.h"

// Select camera model - find more camera models in camera_pins.h file here
// https://github.com/espressif/arduino-esp32/blob/master/libraries/ESP32/examples/Camera/CameraWebServer
//camera_pins.h

#ifndef CAMERA_MODEL_ESP_EYE // Has PSRAM
#define CAMERA_MODEL_AI_THINKER // Has PSRAM
#endif

#if defined(CAMERA_MODEL_ESP_EYE)
#define PWDN_GPIO_NUM -1
#define RESET_GPIO_NUM -1
#define XCLK_GPIO_NUM 4
#define SIOD_GPIO_NUM 18
#define SIOC_GPIO_NUM 23

#define Y9_GPIO_NUM 36
#define Y8_GPIO_NUM 37
#define Y7_GPIO_NUM 38
#define Y6_GPIO_NUM 39
#define Y5_GPIO_NUM 35
#define Y4_GPIO_NUM 14
#define Y3_GPIO_NUM 13
#define Y2_GPIO_NUM 34
#define VSYNC_GPIO_NUM 5
#define HREF_GPIO_NUM 27
#define PCLK_GPIO_NUM 25
#endif
```

```

#elif defined(CAMERA_MODEL_AI_THINKER)
#define PWDN_GPIO_NUM 32
#define RESET_GPIO_NUM -1
#define XCLK_GPIO_NUM 0
#define SIOD_GPIO_NUM 26
#define SIOC_GPIO_NUM 27

#define Y9_GPIO_NUM 35
#define Y8_GPIO_NUM 34
#define Y7_GPIO_NUM 39
#define Y6_GPIO_NUM 36
#define Y5_GPIO_NUM 21
#define Y4_GPIO_NUM 19
#define Y3_GPIO_NUM 18
#define Y2_GPIO_NUM 5
#define VSYNC_GPIO_NUM 25
#define HREF_GPIO_NUM 23
#define PCLK_GPIO_NUM 22

#else
#error "Camera model not selected"
#endif

/* Constant defines ----- */
#define EI_CAMERA_RAW_FRAME_BUFFER_COLS 320
#define EI_CAMERA_RAW_FRAME_BUFFER_ROWS 240
#define EI_CAMERA_FRAME_BYTE_SIZE 3

/* Private variables ----- */
static bool debug_nn = false; // Set this to true to see e.g. features generated from the raw signal
static bool is_initialised = false;
uint8_t *snapshot_buf; //points to the output of the capture

static camera_config_t camera_config = {
.pin_pwdn = PWDN_GPIO_NUM,
.pin_reset = RESET_GPIO_NUM,
.pin_xclk = XCLK_GPIO_NUM,
.pin_sscb_sda = SIOD_GPIO_NUM,
.pin_sscb_scl = SIOC_GPIO_NUM,

.pin_d7 = Y9_GPIO_NUM,
.pin_d6 = Y8_GPIO_NUM,
.pin_d5 = Y7_GPIO_NUM,
.pin_d4 = Y6_GPIO_NUM,
.pin_d3 = Y5_GPIO_NUM,
.pin_d2 = Y4_GPIO_NUM,
.pin_d1 = Y3_GPIO_NUM,
.pin_d0 = Y2_GPIO_NUM,
.pin_vsync = VSYNC_GPIO_NUM,
.pin_href = HREF_GPIO_NUM,
.pin_pclk = PCLK_GPIO_NUM,

//XCLK 20MHz or 10MHz for OV2640 double FPS (Experimental)
.xclk_freq_hz = 20000000,
.ledc_timer = LEDC_TIMER_0,
.ledc_channel = LEDC_CHANNEL_0,

.pixel_format = PIXFORMAT_JPEG, //YUV422,GRAYSCALE,RGB565,JPEG
.frame_size = FRAMESIZE_QVGA, //QQVGA-UXGA Do not use sizes above QVGA when not JPEG

.jpeg_quality = 12, //0-63 lower number means higher quality
.fb_count = 1, //if more than one, i2s runs in continuous mode. Use only with JPEG
.fb_location = CAMERA_FB_IN_PSRAM,
.grab_mode = CAMERA_GRAB_WHEN_EMPTY,
};

/* Function definitions ----- */
bool ei_camera_init(void);
void ei_camera_deinit(void);
bool ei_camera_capture(uint32_t img_width, uint32_t img_height, uint8_t *out_buf) ;

```

```

/**
 * @brief Arduino setup function
 */
void setup()
{
// put your setup code here, to run once:
Serial.begin(115200);
//comment out the below line to start inference immediately after upload
while (!Serial);
//Serial.println("Edge Impulse Inferencing Demo");
if (ei_camera_init() == false) {
//ei_printf("Failed to initialize Camera!\r\n");
}
else {
//ei_printf("Camera initialized\r\n");
}

//ei_printf("\nStarting continious inference in 2 seconds...\n");
ei_sleep(2000);
}

/**
 * @brief Get data and run inferencing
 *
 * @param[in] debug Get debug info if true
 */
void loop()
{

// instead of wait_ms, we'll wait on the signal, this allows threads to cancel us...
if (ei_sleep(5) != EI_IMPULSE_OK) {
return;
}

snapshot_buf = (uint8_t*)malloc(EI_CAMERA_RAW_FRAME_BUFFER_COLS * EI_CAMERA_RAW_FRAME_BUFFER_ROWS *
EI_CAMERA_FRAME_BYTE_SIZE);

// check if allocation was successful
if(snapshot_buf == nullptr) {
//ei_printf("ERR: Failed to allocate snapshot buffer!\n");
return;
}

ei::signal_t signal;
signal.total_length = EI_CLASSIFIER_INPUT_WIDTH * EI_CLASSIFIER_INPUT_HEIGHT;
signal.get_data = &ei_camera_get_data;

if (ei_camera_capture((size_t)EI_CLASSIFIER_INPUT_WIDTH, (size_t)EI_CLASSIFIER_INPUT_HEIGHT, snapshot_buf) ==
false) {
//ei_printf("Failed to capture image\r\n");
free(snapshot_buf);
return;
}

// Run the classifier
ei_impulse_result_t result = { 0 };

EI_IMPULSE_ERROR err = run_classifier(&signal, &result, debug_nn);
if (err != EI_IMPULSE_OK) {
//ei_printf("ERR: Failed to run classifier (%d)\n", err);
return;
}

// print the predictions
// ei_printf("Predictions (DSP: %d ms., Classification: %d ms., Anomaly: %d ms.): \n",
// result.timing.dsp, result.timing.classification, result.timing.anomaly);

#if EI_CLASSIFIER_OBJECT_DETECTION == 1
bool bb_found = result.bounding_boxes[0].value > 0;
for (size_t ix = 0; ix < result.bounding_boxes_count; ix++) {

```

```

auto bb = result.bounding_boxes[ix];
if (bb.value == 0) {
continue;
}
//ei_printf(" %s (%f) [ x: %u, y: %u, width: %u, height: %u ]\n", bb.label, bb.value, bb.x, bb.y, bb.width, bb.height);
sendOffset(bb.x, bb.width); //laske offset ja lähetä se
}
if (!bb_found) {
//ei_printf(" No objects found\n");
if (Serial.available() > 0) { //tarkista onko bufferiin tullut dataa
while(Serial.available()) {Serial.read();} //tyhjentää bufferin
delay(10);

Serial.println(0, DEC); //lähetä 0 jost ei näe mitään
}
}
#endif

#if EI_CLASSIFIER_HAS_ANOMALY == 1
ei_printf(" anomaly score: %.3f\n", result.anomaly);
#endif

free(snapshot_buf);
}

void sendOffset(int x, int w){ //laskee paljonko tunnistettu objekti on sivussa keskikohdasta
int o;
o = 97 - (x + (w/2));
// 96 + 1 , koska halutaan varata nolle sille ettei havaitse mitään
if (Serial.available() > 0) { //tarkistaa onko omassa bufferissa dataa
while(Serial.available()) {Serial.read();} //tyhjentää oman bufferin
delay(10);

Serial.println(o, DEC); //lähetää lasketun offsetin
}
}

/***
* @brief Setup image sensor & start streaming
*
* @retval false if initialisation failed
*/
bool ei_camera_init(void) {

if (is_initialised) return true;

#if defined(CAMERA_MODEL_ESP_EYE)
pinMode(13, INPUT_PULLUP);
pinMode(14, INPUT_PULLUP);
#endif

//initialize the camera
esp_err_t err = esp_camera_init(&camera_config);
if (err != ESP_OK) {
Serial.printf("Camera init failed with error 0x%x\n", err);
return false;
}

sensor_t * s = esp_camera_sensor_get();
// initial sensors are flipped vertically and colors are a bit saturated
if (s->id.PID == OV3660_PID) {

```

```

s->set_vflip(s, 1); // flip it back
s->set_brightness(s, 1); // up the brightness just a bit
s->set_saturation(s, 0); // lower the saturation
}

#ifndef CAMERA_MODEL_M5STACK_WIDE
s->set_vflip(s, 1);
s->set_hmirror(s, 1);
#endif
#ifndef CAMERA_MODEL_ESP_EYE
s->set_vflip(s, 1);
s->set_hmirror(s, 1);
s->set_awb_gain(s, 1);
#endif

is_initialised = true;
return true;
}

/**
 * @brief Stop streaming of sensor data
 */
void ei_camera_deinit(void) {

//deinitialize the camera
esp_err_t err = esp_camera_deinit();

if (err != ESP_OK)
{
ei_printf("Camera deinit failed\n");
return;
}

is_initialised = false;
return;
}

/**
 * @brief Capture, rescale and crop image
 *
 * @param[in] img_width width of output image
 * @param[in] img_height height of output image
 * @param[in] out_buf pointer to store output image, NULL may be used
 * if ei_camera_frame_buffer is to be used for capture and resize/cropping.
 *
 * @retval false if not initialised, image captured, rescaled or cropped failed
 */
bool ei_camera_capture(uint32_t img_width, uint32_t img_height, uint8_t *out_buf) {
bool do_resize = false;

if (!is_initialised) {
ei_printf("ERR: Camera is not initialized\r\n");
return false;
}

camera_fb_t *fb = esp_camera_fb_get();

if (!fb) {
ei_printf("Camera capture failed\n");
return false;
}

bool converted = fmt2rgb888(fb->buf, fb->len, PIXFORMAT_JPEG, snapshot_buf);

esp_camera_fb_return(fb);

if(!converted){
ei_printf("Conversion failed\n");
return false;
}
}

```

```

if ((img_width != EI_CAMERA_RAW_FRAME_BUFFER_COLS)
|| (img_height != EI_CAMERA_RAW_FRAME_BUFFER_ROWS)) {
do_resize = true;
}

if (do_resize) {
ei::image::processing::crop_and_interpolate_rgb888(
out_buf,
EI_CAMERA_RAW_FRAME_BUFFER_COLS,
EI_CAMERA_RAW_FRAME_BUFFER_ROWS,
out_buf,
img_width,
img_height);
}

return true;
}

static int ei_camera_get_data(size_t offset, size_t length, float *out_ptr)
{
// we already have a RGB888 buffer, so recalculate offset into pixel index
size_t pixel_ix = offset * 3;
size_t pixels_left = length;
size_t out_ptr_ix = 0;

while (pixels_left != 0) {
out_ptr[out_ptr_ix] = (snapshot_buf[pixel_ix] << 16) + (snapshot_buf[pixel_ix + 1] << 8) + snapshot_buf
[pixel_ix + 2];

// go to the next pixel
out_ptr_ix++;
pixel_ix+=3;
pixels_left--;
}
// and done!
return 0;
}

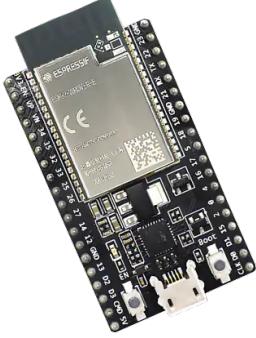
#ifndef EI_CLASSIFIER_SENSOR
#error "Invalid model for current sensor"
#endif

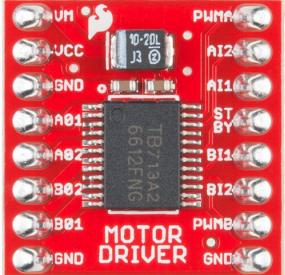
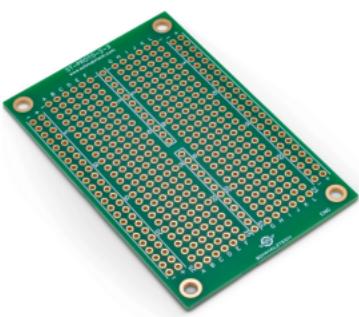
```

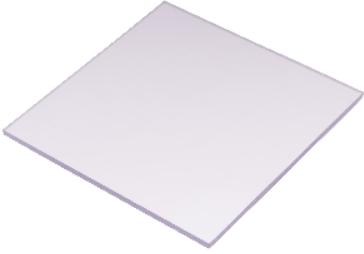
7. Osat ja materiaalit

7.1. Komponentit

Osa	Määrä	Kuva	Tarkoitus	Datasheet
-----	-------	------	-----------	-----------

ESP32 DevKitC V4	1		Keijon aivot; ESP32 hoittaa pääasiallisesti kaiken logiikan	https://docs.espressif.com/projects/esp-idf/en/stable/esp32/hw-reference/esp32/get-started-devkitc.html#related-documents
AZ-Delivery ESP32-CAM AI-Thinker Model	1		Kamera ihmisten tunnistukseen	https://cdn.shopify.com/s/files/1/1509/1638/files/ESP32-Cam_datasheet.pdf?v=1689236064
Ultraäänisen sori HC-SR04	1		Ultraäänisensori tunnistaa Keijon etäisyden kohteisiin ja auttaa pysäytämään Keijon, kun se on törmäämässä johonkin	https://pdf1.alldatasheet.com/datasheet-pdf/view/1132203/ETC2/HC-SR04.html
Turnigy LiPo-akku 4000mAh 35-70C	1		Akusta saadaan käyttöön kaikkien komponenttien tarvitsema virta	
Sunfounder SF3218MG servo	1		Servomoottoria käytetään Keijon pään kääntämiseen	http://wiki.sunfounder.cc/index.php?title=TM-8120MG_Digital_Servo http://wiki.sunfounder.cc/images/9/9a/TD-8120MG_Digital_Servo.pdf

SparkFun Dual Motor Driver TB6612FNG	2		Moottorinohjaimilla ajetaan moottoreita https://www.sparkfun.com/datasheets/Robotics/TB6612FNG.pdf https://learn.sparkfun.com/tutorials/tb6612fng-hookup-guide/all	
Protoboard	2		Protoboardille juotettiin kaikki johdot ja komponentit, jotta välttyään leipälaudan aiheuttamalta johtosotkulta ja pienennetään riskiä johtojen irtoamisesta	
Harjallinen DC moottori	4		Moottorit ovat yhdistettyinä renkaisiin. Jokaisella renkaalla on oma moottori, joita ohjataan moottorinohjaimilla.	
Rengas	4		Renkaat pyörivät ja liikuttavat Keijoja	
DFRobot 20W DC- DC buck- konverterri	1		Konverterillä pystytään säätelemään akusta saatavia jännitemääriä eri komponenteille sopiviksi; 6V-kisko moottoreille ja servolle	https://www.mouser.com/pdfDocs/ProductOverview-DFRobot-DFR0379.pdf
STMicroelec tronics L7805CV 5V Fixed Linear Regulator	1		5V-kisko elektroniikalle	https://www.st.com/content/ccc/resource/technical/document/datasheet/41/4f/b3/b0/12/d4/47/88/CD00000444.pdf/files/CD00000444.pdf/jcr:content/translations/en.CD00000444.pdf

Kondensaattori (alumiinielektrolyytti) 330 F	1		Lineaariregulaattorin sisääntulojännitteen tasaaminen (bulk decoupling)	
Kondensaattori (keräaminen) 100 nF	1		Lineaariregulaattorin ulostulojännitteen transienttien poistoon	
Akryylilevy 3 ja 5 mm	~3900 cm ²		Keijon ulkokuori ja runkorakenteet	
Filamenttia	Kyllä		3D-printtausten materiaali	
Kutistesukkaa (eri kokoja)	Useita		Johtosotkujen selvittämiseen ja juotoksiin eristämiseen	
Nylonsukkaa	~1 m		Johtosotkujen selvittämiseen	
Jäähdityssiiili	1		Lineaariregulaattorin jäähdytys	

7.2. Liittimet

Liitin	Määrä

Molex KK Female	
2-pin	2
4-pin	4
Molex KK Male	
2-pin	2
4-pin	4
0,1 in Pin Header Female	
4-pin	2
16-pin	6
19-pin	2
0,1 in Pin Header Male	
3-pin	1
4-pin	1
0,1 in Pin Header Male, 90 deg	
4-pin	2
XT60 Male	1

7.3. Kiinnikkeet

Kiinnike	Määrä
M3x10 uppokantaruuvi	54
M3x10 koneruuvi	8
M3 Nyloc-mutteri	8
M4x12 koneruuvi	4
M4x20 koneruuvi	1
M4 mutteri	5
M4 aluslevy	1
M5x10 koneruuvi	8
M5 mutteri	8
M6 aluslevy	1

8. Ongelmat & ratkaisut

8.1. 3D-printtauksen ongelmat

8.1.1. Printteri hajosi



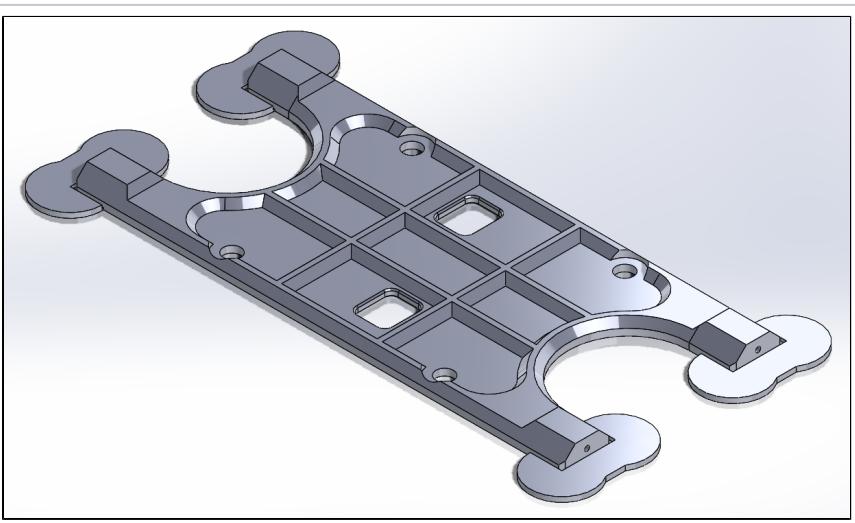
- odotettiin, että se korjattiin 😊
- lue: me ei rikottu..
- Printerri korjattiin

8.1.2. Printit epäonnistuvat todella paljon ja todella usein

8.1.2.1. "Liftaus"

Useat osat epäonnistuvat materiaalin irrottua printtilästasta osan reunolla. Kurssin aikana (erityisesti loppuviikkoina) irtoamismekanismiksi osoittautui hyvin samankaltainen ilmiö kuin hitsaussaumoissa esiintyvä lämpövääristymä. Kun hitsisauma jäähtyy, se kutistuu, joka aiheuttaa merkittäviä sisäisiä voimia kappaleeseen, mikä vetää ja väristää hitsattua osaa kutistuvan sauman suuntaan. Samalla tavalla 3D-tulostaessa jokaista tulostuskerrosta voi ajatella pienennä hitsisaumana, jotka yhdessä kutistuessaan aiheuttavat kappaleeseen vääristävän voiman, joka pyrkii väentämään litteät osat kuperaksi.

Kun vääristävän voiman mekanismin tuntee, voi osiin suunnitella printtausta helpottavia apurakenteita. Ensinnäkin terävät kulmat tahtovat irrota helposti, koska terävän kulman ympäällä on vain vähän kulmaa tukevaa materiaalia. Toisekseen, paksut rakenteet kappaleen reunalla aiheuttavat suurempia vääristäviä voimia (suurempi määrä "hitsisaumoja"). Molempien näihin ongelmien ratkaisuna ovat "kolikot" printatut osan reunalla (erityisesti terävissä kulmissa), jotka pyöristävät terävät kulmat ja antavat suuren tarjumapinta-alan printtilästään. Kolikot eivät kuitenkaan saa olla itsessään liian paksuja, tai kolikot tuottavat omia sisäisiä vääristäviä voimia. Osien printtausta suunnitellessa on hyvä noudattaa "pyramidiperiaatetta", eli paksut piirteet kappaleen keskelle, ja piirteiden tulisi ohentua kappaleen reunaa lähestyessä. Suuret kulmaviisteet alustan läheisyydessä osoittautuvat myös toimiviksi edellä mainitusta syystä.



Esimerkki "kolikoista" moottorialustan kulmissa.

8.1.2.2. Suutinkoko

Pienempi suutin on parempi kuin suurempi suutin kaikessa muussa, kuin printtinopeudessa. Ainakin näin vaikuttaa asia olevan omien kokemuksiemme mukaan. Suuremmat suuttimet soveltuvat lähinnä vain hyvin yksinkertaisiin ja suuriin muotoihin.

8.1.2.3. Printiresoluutio

Riippuen printeristä, Ultimaker Cura antaa valita joitakin eri resoluutioita printtiohjelmaa tehdessä. Voisi kuvitella, että tarkin resoluutio antaisi aina parhaimman laadun, mutta kokemuksiemme mukaan näin kuitenkin harvoin kävi. Esimerkiksi 0,4 mm suuttimella 0,1 mm resoluutio antoi harvoin laadukkaita osia. Parhaimman laadun saimme, kun resoluutio oli hieman suuttimen kokoa pienempi (esim. 0,4 mm suuttimella 0,15 - 0,20 mm resoluutio).

8.1.2.4. Printtimateriaali

Muovin valinnalla oli merkittävä vaikuttus printtien onnistumiseen. Esimerkiksi valkoinen ja läpinäkyvä muovi eivät meidän käytössä antaneet kertaakaan onnistuneita osia. Parhaimmat tulokset saimme mustalla ja sinisellä muovilla.

Tutkimustyötä tehdessä, selitys lienee muovin laatu ja kosteus, tai väärät printerin asetukset (pursotuslämpötila ja -nopeus, alustan lämpötila, jne.) kyseiselle muoville. Joidenkin valmistajien muovit ovat laadukkaampia ja jotkin muovit voivat väärin varastoituna kerätä kosteutta, minkä seurauksena printtilaatu kärssi.

8.1.2.5. Osa printteistä hajosi kasatessa

- tulostettiin uudelleen paranneltuina
- kohdeltiin paremmin
- eivät hajonneet uudestaan

8.1.3. Printattujen osien suunnittelu

8.1.3.1. Puristussovitus

Puristussovitusliitokset osoittautuivat alkuun vaikeaksi, mutta loppussa helpoksi liitäntätavaksi (ajattele laakerin asennusta prässäämällä laakeripesään). Alkuun suunnittelijamme puristussovitettyt osat samaan tapaan kuin mitkä tahansa muut metalli-metalli -sovitukset, eli sisäosan ulko-osaa suuremmaksi esimerkiksi puolikkaan millin kymmenynksen verran, jotta osat prässäytyisivät yhteen luotettavasti. Printattuun muovin kanssa tulee kuitenkin toleranssiongelmia puristussovitteita suunnitellessa. Muovin printtaaminen riittävään toleranssiin oikeaan sovitukseen saamiseksi osoittautui käytännössä mahdottomaksi. Esimerkiksi samoilla printtiasetuksilla yksi osa saattaa saada kevyen puristussovituksen, mutta toinen osa jää merkittävästi väljäksi.

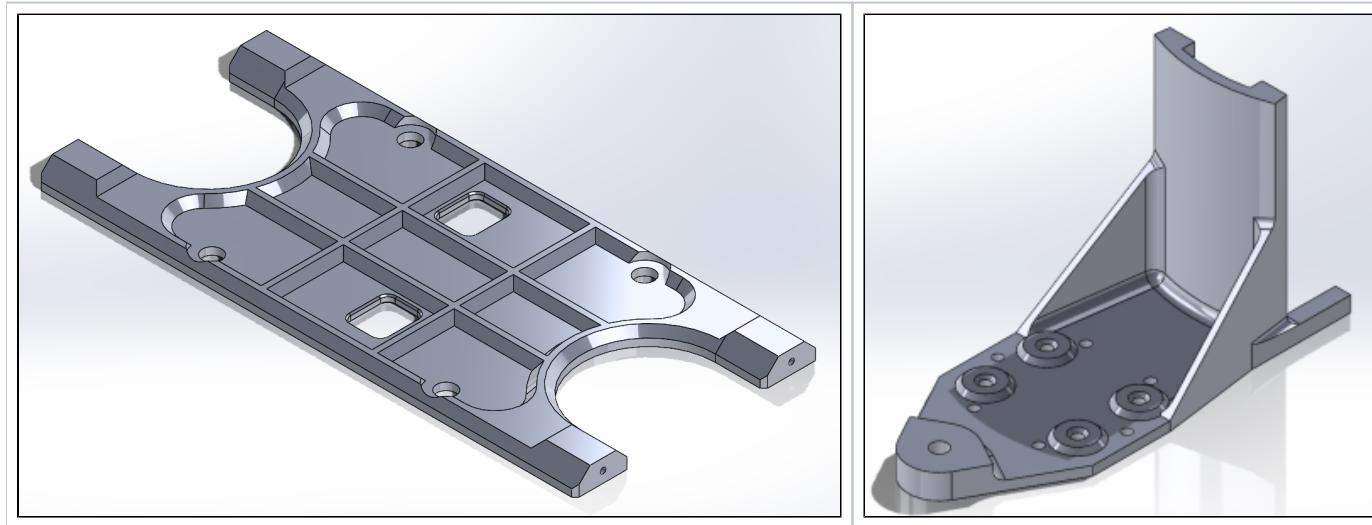
Ratkaisu on yksinkertainen: muovi on suorastaan pehmeää verrattuna metalliin, eli muovin voi pakottaa mukautumaan metallin muotoon voimakkaasti prässäämällä (pylväsporakoneen istukka osoittautui todella käteväksi prässäksi). Kurssin lopussa suunnittelijamme kaikki puristusliitokset noin 0,2 mm liian ahtaiksi, ja pakotimme osat yhteen. Lopputulos on todella luotettava, voimakas ja helposti kasattava liitos erityisesti muovi-metalli -liitoksille. Muovi-muovi -puristussovitteissa on syytä suunnitella ulko-osa riittävän tukevaksi, jotta sisemmän osan syrjäyttävä voima ei halkaise ulko-osaa (printattu PLA on osoittaunut heikohkoksi vedossa).

Muovi-metalli -liitosta kasattaessa on kuitenkin hyvä ottaa huomioon, että printattu PLA on melko hauras materiaali. Liitoksia ei kannata edes yrityää sovittaa vasaroimalla, koska muovi halkeaa helposti iskuista. Liitos on käytännöllinen vain prässäämällä.

8.1.3.2. Muotolujititteet

Yksi 3D-tulostuksen eduista sekä muovin että metallin tulostuksessa ovat muodot, jotka olisivat vailleita tai jopa mahdottomia valmistaa lastuavilla menetelmillä. Muoviosia kannattaa harvoin valmistaa paksuista piirteistä jo pelkästään aikaisemmin mainitun lämpövääristymän vuoksi. Muotolujitteiden käyttö osien suunnittelussa tekee osista kevyempää (mikä ei Keijon tapauksessa ollut oleellista), mutta myöskin käyttää vähemmän muovifilamenttia, mikä nopeuttaa printtiprosessia.

CAM-ohjelmat tekevät automaatisesti muotolujitteet paksujen piirteiden sisälle käyttäjän valitsemaa kennorakennetta käyttäen. Kokemuksiemme mukaan osia kuitenkin kannattaa harvoin rakentaa paksuiksi. Esimerkiksi ero tasapaksun 4 mm levy ja 2 mm muotolujitetun levyn jälkikäytön välillä on mitätöin, vaikka ohuempi levy käyttäisi merkittävästi vähemmän muovia ja täten myös tulostuu nopeammin. Tilavuudeltaan suuria osia kannattaa suosia lähinnä kiertolujuttaa vaativissa kohteissa.



Moottorialusta on tehty 2mm paksusta levystä, jota on lujitettu rimoilla osan reunaa myöten, sekä levyn keskellä kahdella pitius- ja leveyssuuntaisella rimalla. Alusta on jäykäväänössä pitius- ja leveysakseleilla. Pituusakselin suunnassa alusta kiertyy helposti kappaleen pienien poikkipinta-alan vuoksi, mutta tämä ei ole Keijon tapauksessa merkittävä, koska alusta kiinnittyää jäykään akryylikuoreen alustan kulmista, mikä estää kierron.

Päänsisältämää anturipidikettä on lujitettu kahdella pystytuella, sekä kaartamalla pystyseinämää. Pidikkeen seinämään kiinnitetty anturikokooppano pysyy jäykänä ja tärinävapaana suhteessa pään akseliin, vaikka korkea pystyseinämä tarjoaa pitkän vipuvarren alustaan näden.

8.2. Muut ongelmat

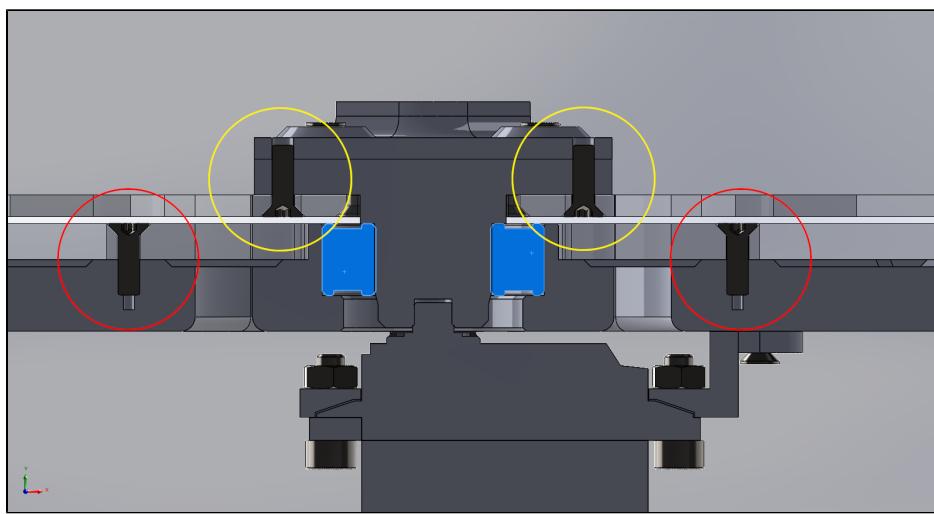
8.2.1. Ruuviliitokset

Valtaosaan muoveista on erittäin vaikeaa tehdä kestäviä kierteitä (poislukien koneistettavat muovit, kuten Delrin), eli koneruuviliitos on epäkäytännöllinen liitäntätapa. Käytimme paljon pultti-mutteri-liitoksia, mutta jokainen pultti-mutteripari tuplaa kiinnikemäären yhteen ruuviin verrattuna, mikä ei ole ekonominen ja elegantti ratkaisu ja lisää asennusaikaa.

Lämmöllä asennettavat "kierreinsertit" voisivat olla elegantti, vahva ja kestävä liitäntätapa koneruuvin kanssa muoville, mutta vaatisivat ylimääräisiä tilattavia osia ja asennusaikaa. Insertit eivät myöskään ole tarkoitukseen sopivia, jos projektin ei olisi pitkäaikaisessa käytössä, jossa metallikierteen tarjoama kestävyys ja asennushelpous uudelleenasennuksissa olisi merkittävä etu.

Käytimme paljon muoviosien kiinnitykseen upkokantaruuveja, mikä osoittautui erittäin toimivaksi liitäntätavaksi erityisesti akryyli-muovi -liitoksille. Akryylireiät on helppo tehdä ja sijoittaa tarkasti laserleikkurilla, ja reikien senkkaaminen (joka onnistuu yllättävän helposti, vaikka akryyli onkin melko hauras materiaali) upkokantaa varten upottaa ruuvin pään akryylipinnan alle, mikä tekee liitoksesta siistin ja toimivan ahtaissa raoissa. On kuitenkin huomioitava, että printattujen kappaleiden sisäisten rakenteiden vuoksi ruuviliitosta ei voi tehdä ilman valmistelua (ruuvi asetettu vinoon, kun se kohtaa kappaleen sisäisen kennarakenteen). On kuitenkin helppoa tehdä printtiin pieni pilottireikä ja mallinnusvaiheessa, jonka perusteella CAM-ohjelma rakentaa seinämän reiän ympärille, mikä ohjaa ruuvin suoraan ja tarjoaa sille materiaa johon pureutua kiinni.

Ruuviliitos oli kuitenkin vaikeaho osiin, jossa reiät olivat lähellä printatun kappaleen reunoja. Jos pilottireikä on vähänkään liian pieni, ruuvi syrjäyttää muovia reiän ympäillä, mikä voi aiheuttaa osan murtumisen lähellä reunaan. Jos pilottireikä on taas liian suuri, ei ruuvilla ole riittävästi pureutumisvoimaa. $0,5 * D_{\text{ruuvi}}$ osoittautuu hyväksi pilottireiän halkaisijaksi, kun murtumisvaaraa ei ollut. Hauraisiin osiin oli tehtävä jopa $D_{\text{ruuvi}} - 0,2 \text{ mm}$ -kokoisia pilottireikiä, jolloin ruuveilla ei ollut juurikaan pureutumisvoimaa, mutta muovi ei syrjäytynyt riittävästi aiheuttaakseen murtumisvaaraa.



Esimerkki ruuviliitoksesta pään asennukseen. Kuva on poikkileikkaus pään akselistasta, jossa laakeri on korostettu sinisellä. Upotetut ruuvit kiinnittävät sekä pään akselin pään kuoreen (korostettu keltaisella), että torson yläpään laakeripesänä toimivaan printattuun kappaleeseen (korostettu punaisella). Pään ja torson välissä on vain 1 mm rako, joka onnistuu vaivatta upotetuilla ruuveilla.

8.2.2. Diagonaaliseksi asetetut moottorit eivät käänneet Keijoaa toiseen suuntaan

- Vaihdettiin pois alkuperäisestä suunnitelmasta:
- kahden moottorin sijaan neljä moottoria, jolloin ohjaus tapahtuu yhden moottorihaimen sijaan kahdella ja jokainen rengas on ohjelmoitavissa
- ongelma ratkesi

8.2.3. Virranjako komponentteihin

- Osa tarvitsi 5V jänniteen ja osa 6V jännitteen. Meillä ei ollut välineitä tällaiseen
- saatiin buck-konverteeri, jolla sen pystyi tekemään.
- ongelma ratkesi

8.2.4. ESP32CAM tarkkuus ihmisen tunnistuksessa

- Ensimmäiset ihmisen tunnistusmallit olivat epätarkkoja

- ottettiin lisää parempia kuvia ja tehtiin uusi malli, joka toimii huomattavasti paremmin, muttei ihan täydellisesti
- ongelma ratkesi, ainakin osittain ✓

8.2.5. Laserleikkuri oli rikki

- Odotettiin korjausta!
- korjattiin!

9. Kehitettäväää

Vaikka Keijo onkin kaunis, ei hänen ole täydellinen.

9.1. Liitostyypit

9.1.1. Muovi-muovi -liitokset

Loppukurssia kohden, kun 3D-printtailuteknikka (erityisesti toleranssit) alkoi olla hanskassa, projektissa olisi avautunut mahdollisuus käyttää upotettuja muttereita insertteinä, joihin voisi ruuvata suoraan koneruuveja. Tässä vaiheessa osien uudelleensuunnittelu ja printtaus ei enää ollut mahdollista. Kaikkien pultti-mutteri -liitosten korvaaminen muoviin upotetulla mutterilla ja koneruuvilla pitäisi kiinnikkeiden määrän samana, mutta helpottaisi ja nopeuttaisi asennusta.

9.1.2. Akryyli-akryyli -liitokset

Mekaanisen suunnittelun yksi suurimmista kehityskohteista on akryylikuoren liitosten parantaminen. Akryylit oli suunniteltu kiinnitettäväksi toisiinsa vain kuumaliimaamalla, mutta tämä osoittautui kovin kehnaksi kiinnitystavaksi paristakin eri syystä:

- Kuumaliimaaminen on hidasta
- Kuumaliimaliitokset irtosivat hyvin helposti
- Liimaaminen on pysyvä kiinnitysratkaisu, joka estää helpon ja nopean purkamisen ja uudelleenasentamisen
- Monien akryyiliitosten liimaaminen oli mahdollista vain kasattuna, jolloin liimapistoolin käytöö robotin sisällä oli vaikeaa tai mahdotonta
- Kuumaliimaliitos ei ole elegantti tai esteettisesti mieluisa liitosmenetelmä

Jos suunnittelisimme akryyliiitoksia uudelleen, käyttäisimme mahdollisimman paljon ruuveja kiinnitykseen, esim printattuja kulmakiiloja hyödyntäen. Akryylin osoittauduttua kovaksi, mutta helposti lastuavaksi materiaaliksi, olisi myös mielenkiintoista kokeilla kierteen tekemistä suoraan akryyliin joitakin liitoksia varten.

9.2. Tunnistustarkkuus

Vaikka tunnistust toimiikin tällähetkellä suhteellisen hyvin voisi tunnistustarkkuutta yrittää parantaa hieman. Nykyisen mallin tunnistustarkkuus tunnistaa ihmisen taustasta 85.0 % kerroista ja antaa sen sijainnin melko tarkasti alhaisesta 96x96 pikselin resoluutiosta huolimatta. Tunnistusmallin tarkkuutta voisi parantaa lisäämällä mallin kehitysvaiheeseen kuavia, joissa olisi useampia erilaisisia ihmisiä, koska tällä hetkellä kuvissa on vain ryhmämme jäseniä. Mallin kheityksessä voisi myös yrittää etsiä ja säätää optimaalisempia asetuksia. Myös mallin resoluutiota voisi yrittää nostaa, mutta sitten sen mahdutuminen esp32-camilla olisi epävarmaa ja lataaminen todella hidasta. Resoluution nostamisen myötä myös prosessoointinepos voisi laskea merkittävästi, mikä taas aiheuttaisi muita ongelmia. Myös ESP32-CAM:n asetuksia voisi yrittää säätää vähän paremmiksi siten, että ne olisivat optimaaliset käytettävään ympäristön.

9.3. Servon valinta

Valitsemamme servo on kohtuullisen voimakas, ja se on suunniteltu saavuttamaan käskyttelyn sijaintinsa mahdollisimman äkillisesti. Tämä aiheuttaa päälle nykivää liikehdintää, joka vaikuttaa kameran ihmisen tunnistusta.

Tarkoitukiimme sopisi huomattavasti hitaammin pyörivä moottori, jota voisi mahdollisesti kiihdyttää tasaisesti nykimisen välttämiseksi. Olisi myös mielenkiintoista kokeilla hitaaksi välitettyä moottoria, jota ohjataan *kooderin* (eng. *rotary encoder*) signaalin avulla.

9.4. Ohjelmakoodi

Ohjelmakoodi on toiminnallisesti melko hyvä, mutta sen selkeyttä voisi yrittää parantaa hieman. Siinä voisi esimerkiksi korvata joitakin arvoja muuttujilla, joita olisi helpompi muokata. Myös joidenkkin testaamalla löydettyjen arvojen kalibrointia voisi yrittää parantaa testaamalla esimerkiksi käännytmistä lisää, jolloin käännyminen voisi olla tarkempaa. Myös joitakin funktioita voisi tehdä vähän helpommin muokkattavaksi. Koodissa on myös joitakin mahdollisesti turhia kohtia (kuten delayt, joista ei ole hyötyä) joita vois poistaa. Tämä kuitenkin vaatii jokaisen kohdan testaamista, mikä taas veisi paljon aikaa siihen nähdien, että niistä ei ole merkittävää haittaa toimivuuden kannalta.

9.5. Johtosarjat

Lähtökohtaisesti olemme todella tytyväisiä johtosarjojen laatuun, helppoon käytettävyyteen ja ulkonäköön. Pienenä parannuksena olisi kuitenkin parempien liittimien käyttö erityisesti *pin header* -liitinten tilalla. Pää anturikokoonpanossa on tilaa vain 90-asteen liittimille, minkä seurauskena jouduimme käyttämään *pin headereitä* johtosarjoissa (ainoat 90-asteen liittimet, jotka löysimme pajalta), joissa ei ole sisäänrakennettua estoa liittimen väärinkytkemiselle. Olisi ollut asiaankuuluvaa tilata 90-asteen Molex KK -liittimiä, joita käytimme kaikkialla muualla sähköliitoksiin.

9.6. Lineaariregulaattorin jäähdytys

Käytämme 5V lineaariregulaattoria elektroniikan ajoon, joka ottaa virtansa suoraan akulta. Alunperin ajatus oli käyttää 2S LiPo-akku, joka kurssin aikana vaihtui 3S-akuksi. Ylimääräinen akkujännite aiheuttaa regulaattorin yli suuremman jännitteenalenneman, minkä seurauskena regulaattoriin aiheutuu merkittävästi suurempi hukkalämpöteho.

Huomasimme regulaattorin kuumenevan merkittävästi ilman jäähdytyssililiä (sormitestillä arvio 60-70 °C). Jouduimme jälkkikäteen asentamaan jäähdytyssililin regulaattoria varten, joka on merkittävästi ylmitoitettu regulaattorille ja hukkaa paljon tilaa. Siili on myöskin kiinnitetty vain suoraan TO-220 -pakkauksen kiinnitysreikään, minkä seurauskena regulaattorin juotoksiin kohdistuu merkittäviä voimia siiliin suuren painon ja regulaattorin pitkän vipuvarren seurauskena. Siili tarvitisi kipeästi kunnollisen mekaanisen kiinnitysratkaisun.

9.7. Piirien suojaus

Pajalta olivat loppuneet kaikki XT60-liittimet, joita tarvitsimme akun kytkemiseen. Löysimme erään yksinäisen liittimen, joka ei sopinut akkuun, mutta jonka pinnit sopivat yksinään akun naaraspulon pinneihin. Jouduimme poistamaan liittimen ympäriltä kuoren, jotta saisimme kiinnitettyä pinnit suoraan akun liittimeen kiinni.

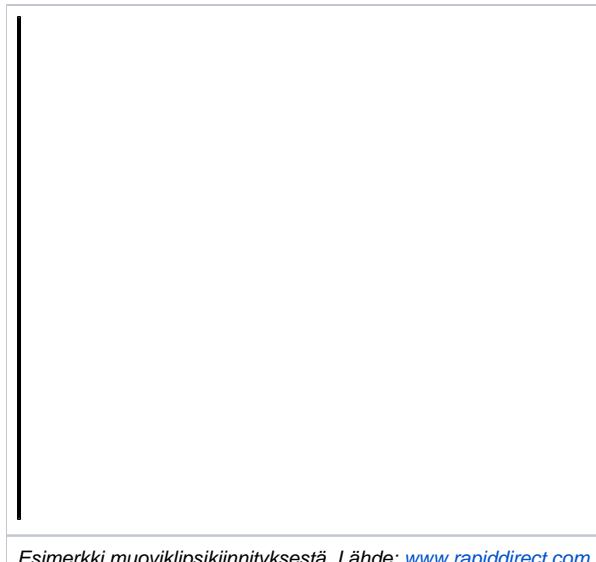
Tästä syystä akun väärinkytkemiselle on suuri riski. Piirissä olisi asiallista olla käänteisen polarisaation suojaus komponenttien suojelemiseksi.

9.8. Helposti vaihdettava akkumoduuli

Akku on kiinnitetty torson takaseinään kahdella akkupidikkeellä, jotka on ruuvattu seinään neljällä ruuvilla. Torson sisällä olevan vähäisen tilan vuoksi akun poistaminen Keijosta on työlästä, joka tekee akun lataamisesta vaikeampaa.

Olimme alunperin ideoineet akun kiinnityksen niin, että se liukuisi Keijon pohjasta sisään kiskoilla. Oli myös ajatuksena kokeilla, miten 3D-tulostus selviytyisi muovikliprien tuottamisesta, jolla akku voitaisiin kiinnittää runkoon ilman työkaluja. Akkujohdon kiinnittäminen Keijon ulkopuolella olisi myös helpompaa, mikä tekisi akun asentamisesta helpompaa.

Ajan käydessä vähii, päätimme käyttää mekaanisesti paljon yksinkertaisempaa kiinnitystä.



Esimerkki muoviklipiskiinnityksestä. Lähde: www.rapiddirect.com

9.9. Nappulat ja kytkimet

Projektiin olisi ollut syytä lisätä muutamia eri nappuloita tai kytkimiä, jotka helpottaisivat käyttöä ja debuggausta.

9.9.1. Virtakytkin

Virtakytkin tekisi käytöstä merkittävästi helpompaa. Nyt ainut tapa kytkeä virta päälle on poistaa kylki ja kytkeä akkujohdot kiinni.

9.9.2. Reset ja boot -nappulat ESP32:lle

ESP32:n uudelleenohjelointi vaatii boot-nappulan painamista, ja ohjelman uudelleenkäynnistys reset-nappulan painamista. Nappulat ovat melko huonossa paikassa Keijon sisällä, mikä hankaloittaa ohjelmakoodin muokkausta. ESP32:n nappuloiden rinnalle olisi voinut juottaa omat nappulat, jotka olisi voinut asettaa Keijon pohjaan.

9.10. USB-jatkojohto

ESP32:n uudelleenohjelointi vaatii USB-kaapelin liittämistä. Nykyinen USB-portin sijainti on huonosti saavutettavassa paikassa Keijon sisällä. ESP32:n USB-portin olisi voinut jatkaa jatkojohtoa käyttäen Keijon pohjaan, josta uudelleenohjelointi olisi vaivatonta.

9.11. Muuta kivaa

Voisi lisätä kaiuttimen ja ledejä, joilla saisi Keijon "räjähäitämään". Tämä oli alkuperäinen idea, mutta aikaa ei riittänyt kaikkeen ja päätimme jättää nämä toteuttamatta.

10. Lähteet

10.1. Kirjaimelliset lähteet



Läheen lähde: botanic.cam.ac.uk

10.2. Ei-kirjaimelliset lähteet

1. www.google.com
2. <https://learn.sparkfun.com/tutorials/tb6612fng-hookup-guide/all>
3. <https://www.circuit-diagram.org/>
4. <https://www.gimp.org/>
5. <https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/>
6. <https://www.youtube.com/watch?v=nSGnCT080d8>
7. <https://edgeimpulse.com/>
8. <https://dronebotworkshop.com/esp32-object-detect/>
9. <https://www.solidworks.com/>