A drone with propellers on a white surface

Description automatically generated

AeroBots

**AeroRobotiķi**

Oskars Plotnieks

Jānis Māris Broks

Reinis Krēsliņš

Pauls Ikvilds

05.04.2024 - 30.04.2024

**Apraksts**

**Darbības mērķis**

Projekta AeroBots ietver sevī pilnu četru propelleru bezpilota lidaparātu jeb lidrobotu jeb arī drona izveidi. Sākot 3D drukātu korpusu, kura pamatā ir Arduino platforma. Šis lidrobots ir aprīkots ar dažādiem sensoriem, tostarp žiroskopu, akselometru, barometru un 6 ultraskaņas sensoriem. Šie sensori nodrošina lidrobota autonomu noturēšanos gaisā vienā punktā, stabilitāti un drošu vadāmību ar pulti, novērš lidrobota ietriekšanos šķēršļos un palīdz tiem izvairīties. Pults, kas arī ir paštaisīta, ir aprīkota ar divām kursorsvirām, kas nodrošina vienkāršu un precīzu lidrobota vadību. Šāda pieeja ļauj radīt daudz lētāku risinājumu, kas ne tikai piedāvā līdzīgas funkcionalitātes, kā veikalu lidrobotu, bet arī dod iespēju iemācīties jaunas prasmes un iegūt pieredzi elektronikā, robotikā un programmēšanā.

**Mērķa auditorija**

Šī projekta mērķauditorija ir cilvēki, kuriem ir interese par elektroniku, robotiku un lidrobotu tehnoloģijām vai cilvēki, kuri vēlas padziļināt savas zināšanas šajās jomās. Projekts piedāvā praktisku un izzinošu veidu, kā apgūt sarežģītas tehnoloģijas, kas ir būtiskas mūsdienu tehnoloģiju pasaulē. Tāpat tas var būt interesanti arī tiem, kas vēlas paši izveidot savu elektronisko projektu un iegūt praktiskas iemaņas programmēšanā un robotikā.

**Funkcionalitāte**

Lidrobota funkcionalitāte balstās uz Arduino platformu, kas ievāc un apstrādā datus no dažādiem sensoriem, piemēram, žiroskopa, barometra un akselometra. Šie sensori nodrošina lidrobota stabilizāciju un precīzu pozicionēšanu gaisā. Turklāt, ultraskaņas sensori palīdz lidrobotam izvairīties no šķēršļiem, nodrošinot drošu un stabilu lidojumu. Lidrobota vadību nodrošina Bluetooth modulis, kas saņem komandas no pults. Pults izmanto divus džoisitkus, lai kontrolētu lidrobota kustību visos virzienos. Bluetooth savienojums starp lidrobotu un pulti nodrošina ātru un uzticamu komunikāciju, kas ir būtiski priekš drošas un precīzas vadības.

**Uzbūve**

3D drukātām lidrobota kājām ir pieskrūvētas divas plāksnes, kas savieno visas četras kājas un savā starpā un veido korpusu.

No augšas lidrobota kājām ir 16 caurumi ar vītni. Augšas pamatu (5 mm biezumā) pieskrūvē ar astoņām M4x12 skrūvēm un astoņām M4x16 skrūvēm pie četrām kājām vītņotajos caurumos.

No apakšas lidrobota kājām ir 8 caurumi ar vītni. Apakšas pamatu (3.5 mm biezumā) pieskrūvē ar astoņām M4x12 skrūvēm pie četrām kājām vītņotajos caurumos.

Skrūves M4x16 ir tuvāk korpusa centram, starp kurām var iebīdīt 2200 mAh 95C akumulatoru, kas to tur uz vietas stabili.

Lidrobota uzbūve ietver vairākas būtiskas komponentes. Arduino MEGA, kas atrodas lidrobota korpusā, ir pieslēgti visi nepieciešamie sensori, baterija, Bluetooth modulis un ESC(electronic speed controller), kuram ir pieslēgti 4 motori. Katram motoram ir pievienots propellers, kas nodrošina lidrobota pacelšanos un lidojumu. Pults, kas izmanto Arduino Nano, ir aprīkota ar diviem džoisitkiem un bateriju, kā arī otru Bluetooth moduli, kas nodrošina savienojumu ar lidrobotu. Šāda konstrukcija nodrošina stabilu un efektīvu lidrobota darbību, kā arī vieglu un precīzu vadību. Cinkotas skrūves savieno lidrobota pamatus ar kājām, tās tiek ieskrūvētas izvītņotos caurumos.

**Materiāli**

**Korpuss**

* **Pamati, kājas, pamata kājas: PLA (polilaktīds), 3D drukāts**
* **Propelleri: PLA, 3D drukāts**
* **Savienojumi: 16x 4x12 skrūves, 8x 4x16 mašīnskrūves, 4x16 skrūves**

Lidrobota korpuss un propelleri tiks veidoti no 3D drukātas plastmasas, PLA un ABS. Šis materiāls tika izvēlēts tā viegluma un izturības dēļ, kas ir būtiski, lai lidrobots būtu pietiekami viegls, lai tas paceltos bez nekādām problēmām un izturīgs pret triecieniem. 3D drukāšanas tehnoloģija ļauj izveidot sarežģītas formas un pielāgot dizainu atbilstoši specifiskām prasībām, kas ir būtiski, lai nodrošinātu optimālu funkcionalitāti un veiktspēju. Turklāt, 3D drukāšana sniedz iespēju ātri un lēti izgatavot un pielāgot nepieciešamās detaļas, kas padara pieeju ļoti izdevīgu salīdzinājumā ar tradicionālajām ražošanas metodēm.

**Vispārīga informācija**

* Lidrobota svars: 858g
* Pults svars: 74g
* Kopējā detaļu cena 383,18€
* Kopējās darba izmaksas 487,00€
* Prototipa izmaksas 870.18€
* Aptuvenās ražošanas izmaksas 250.00€
* Iespējamā pārdošanas cena 300.00€

**Lietošanas instrukcija**

1. Sagatavošanās darbības
   1. Pārliecinieties ka lidrobota baterija nav bojāta.
   2. Pārliecinieties ka ir pietiekami vietas lai lidinātu lidrobota.
   3. Pievienojiet bateriju ESC
   4. Pavelciet slēdzi kas savieno ESC ar lidrobota.
   5. Gaidiet vienu īsu mūzikas melodiju kurai sekos vēl viena īsa.
   6. Pēc tam ieslēdziet pulti.
   7. Atkāpjaties drošā attālumā no lidrobota.
2. Lidrobota vadīšana
   1. Pavirzot kreiso kursorsviru uz priekšu, tas palielina propeleru griešanās ātrumu un liek lidrobotam pacelties.
   2. Pavirzot kreiso kursorsviru uz savu pusi, tas samazina propeleru griešanās ātrumu un liek lidrobotam nolaisties.
   3. Pavirzot kreiso kursorsviru pa labi, tas liek griezīsies lidrobotam pulksteņrādītāj virzienā.
   4. Pavirzot kreiso kursorsviru pa kreisi, tas liek griezties lidrobotam pret pulksteņrādītāj virzienā.
   5. Pavirzot labo kursorsviru uz priekšu, tas liek lidrobotam lidotu uz priekšu.
   6. Pavirzot labo kursorsviru uz sevi, tas liek lidrobotam lidot atpakaļ.
   7. Pavirzot labo kursorsviru pa labi, tas liek lidrobotam lidot pa labi.
   8. Pavirzot labo kursorsviru pa kreisi, tas liek lidrobotam lidot pa kreisi.
3. Lidošana
   1. Lēnām stumjat kreiso kursorsviru uz priekšu lai lidrobots paceļas no zemes.
   2. Lidojiet uzmanīgi, izvairāties no šķēršļiem un lidojiet tikai atklātās vietās.
   3. Nekādā gadījumā nelidojiet ar lidrobotu dzīvju būtņu tuvuma un šaurās iekštelpās.
4. Pēc lidojuma
   1. Kad lidrobots ir pilnībā apstājies, pavelciet slēdzi kas savieno arduino ar ESC un atvienojiet bateriju no ESC.
   2. Izslēdziet pulti ar slēdzi.
5. Papildus darbības.
   1. Lādējiet bateriju uzmanīgi nevadot pārāk lielu spriegumu tai.
   2. Lādējiet bateriju drošā vietā tālu no viegli uzliesmojošām lietām, ieteicams uz akmens virsmām.
   3. Lādējot bateriju nestājiet to bez uzraudzības uz ilgu laiku.
   4. Uzmanīgi sekojiet līdzi baterijas uzlādes līmenim un nepārlādējiet to.
   5. Izmantojiet pareizo lādētāju baterijas lādēšanai, kas ir paredzēts LIPO 5S baterijai.
   6. Uzglabājiet lidrobotu un pulti drošā un sausā vietā.

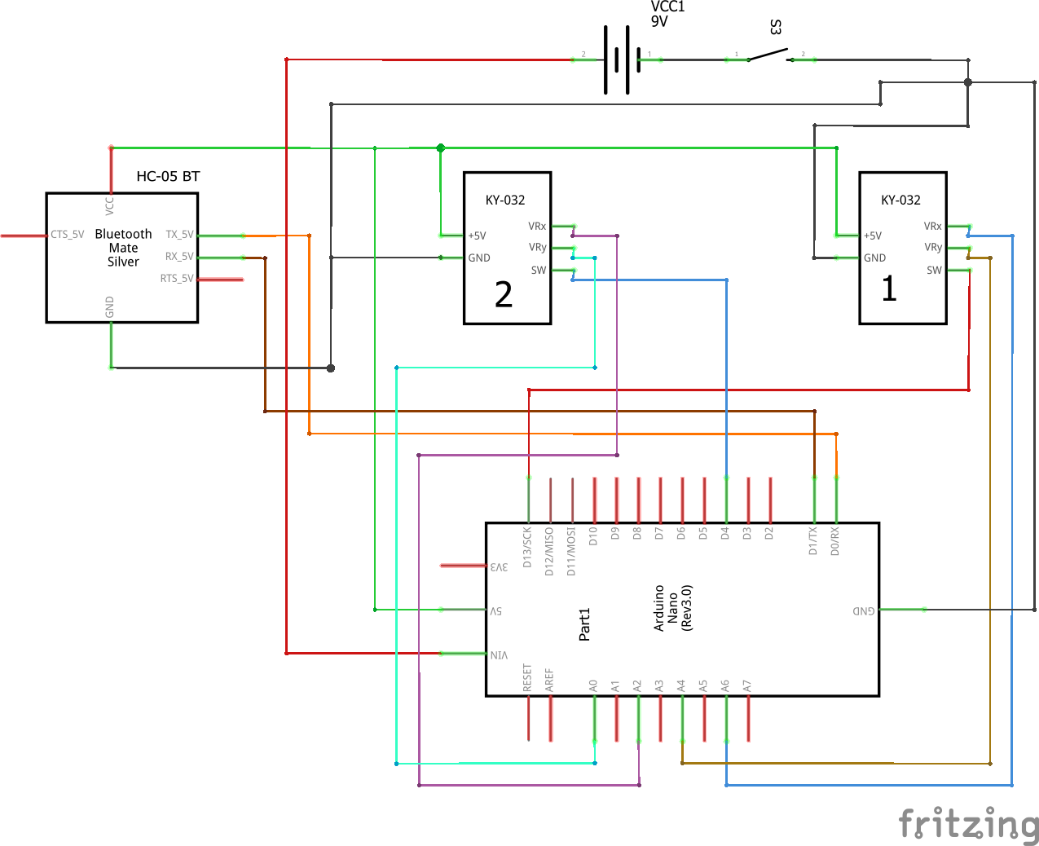
**Shematiskais zīmējums**

**Lidrobota**

Attēls, kurā ir diagramma, plāns, teksts, rasējums

Apraksts ģenerēts automātiski

**Pults**



**Programmas skripts**

Tas ir pagaidu skripts, lai vienkārši paceļas un nosēžas lidrobots, demonstrācijai.

**Pults**

#include <SoftwareSerial.h>

SoftwareSerial BTSerial(3, 6); // RX, TX

const int joystickX = A2;

const int joystickY = A0;

const int joystick1X = A6;

const int joystick1Y = A4;

void setup() {

  BTSerial.begin(38400);

  Serial.begin(38400);

}

void loop() {

  int joyXValue = analogRead(joystickX);

  int joyYValue = analogRead(joystickY);

  int joy1XValue = analogRead(joystick1X);

  int joy1YValue = analogRead(joystick1Y);

  if (joyYValue > 900) {

    BTSerial.println("a");

    Serial.println("augsa");

    delay(2000);

  } else if (joyYValue < 100) {

    BTSerial.println("b");

    Serial.println("leja");

    delay(2000);

  // } else if (joyXValue > 800 && joyYValue > 500) {

  //   BTSerial.println("c");

  //   Serial.println("grieztiesL");

  // } else if (joyXValue > 1000 && joyYValue > 500) {

  //   BTSerial.println("d");

  //   Serial.println("grieztiesK");

  }

  // if (joy1YValue > 800) {

  //   BTSerial.println("e");

  //   Serial.println("prieksu");

  // } else if (joy1YValue < 400) {

  //   BTSerial.println("f");

  //   Serial.println("apakal");

  // } else if (joy1XValue > 800) {

  //   BTSerial.println("g");

  //   Serial.println("labi");

  // } else if (joy1XValue < 400) {

  //   BTSerial.println("h");

  //   Serial.println("kreisi");

  // } else {

  //   BTSerial.println("z");

  //   Serial.println("halt");

  // }

}

**Lidrobota**

#include <Servo.h>

Servo esc1;

Servo esc2;

Servo esc3;

Servo esc4;

const int trigPinH = 6;

const int echoPinH = 7;

enum DroneState { IDLE, ASCENDING, DESCENDING, HOVERING };

DroneState droneState = IDLE;

void setup() {

  Serial3.begin(38400);

  Serial.begin(38400);

  Serial.println("Iedarbinajas");

  pinMode(trigPinH, OUTPUT);

  pinMode(echoPinH, INPUT);

  esc1.attach(25);

  esc2.attach(29);

  esc3.attach(33);

  esc4.attach(37);

  esc1.writeMicroseconds(1000);

  esc2.writeMicroseconds(1000);

  esc3.writeMicroseconds(1000);

  esc4.writeMicroseconds(1000);

  delay(5000);

}

long measureDistance() {

  long duration;

  int distance;

  digitalWrite(trigPinH, LOW);

  delayMicroseconds(2);

  digitalWrite(trigPinH, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPinH, LOW);

  duration = pulseIn(echoPinH, HIGH);

  distance = duration \* 0.034 / 2;

  return distance;

}

void hover() {

  // Set motors to hover state (adjust pulse width as necessary for your setup)

  int hoverPulseWidth = 1200;

  esc1.writeMicroseconds(hoverPulseWidth);

  esc2.writeMicroseconds(hoverPulseWidth);

  esc3.writeMicroseconds(hoverPulseWidth);

  esc4.writeMicroseconds(hoverPulseWidth);

}

void stopMotors() {

  esc1.writeMicroseconds(1000);  // Ensure motors are stopped

  esc2.writeMicroseconds(1000);

  esc3.writeMicroseconds(1000);

  esc4.writeMicroseconds(1000);

}

void performMotorAction(char command, int &lastCommand, bool &isAscending, int distance) {

  if (command == 'a' && lastCommand != 'a') {

    Serial.println("augsa");

    for (int pulseWidth = 1000; pulseWidth < 1400; pulseWidth += 3) {

      if (distance > 40) { // Stop ascending if distance is more than 40

        break;

      }

      esc1.writeMicroseconds(pulseWidth);

      esc2.writeMicroseconds(pulseWidth);

      esc3.writeMicroseconds(pulseWidth);

      esc4.writeMicroseconds(pulseWidth);

      delay(50); // Short delay between updates

    }

    isAscending = true;

    droneState = ASCENDING;

  }

  else if (command == 'b' && lastCommand != 'b') {

    Serial.println("leja");

    for (int pulseWidth = 1400; pulseWidth > 1000; pulseWidth -= 2) {

      esc1.writeMicroseconds(pulseWidth);

      esc2.writeMicroseconds(pulseWidth);

      esc3.writeMicroseconds(pulseWidth);

      esc4.writeMicroseconds(pulseWidth);

      delay(50); // Short delay between updates

    }

    isAscending = false;

    stopMotors();

    droneState = IDLE;

    lastCommand = 0;  // Reset lastCommand to ensure no residual action

  }

  lastCommand = command;

}

void loop() {

  static int lastCommand = 0;

  static bool isAscending = false;

  int distance = measureDistance();

  if (Serial3.available() > 0) {

    char incomingByte = Serial3.read();

    Serial3.flush();

    if (incomingByte == 'a' && distance < 20 && !isAscending) {

      performMotorAction(incomingByte, lastCommand, isAscending, distance);

    }

    else if (incomingByte == 'b') {

      performMotorAction(incomingByte, lastCommand, isAscending, distance);

      stopMotors();

      isAscending = false;

      droneState = IDLE;

      lastCommand = 0;

    }

    // Uncomment and adjust the following sections if needed

    // else if (incomingByte == 'c' && lastCommand != 'c') {

    //   Serial.println("grieztiesL");

    //   esc1.writeMicroseconds(1200);

    //   esc2.writeMicroseconds(1000);

    //   esc3.writeMicroseconds(1200);

    //   esc4.writeMicroseconds(1000);

    //   lastCommand = incomingByte;

    // }

    // else if (incomingByte == 'd' && lastCommand != 'd') {

    //   Serial.println("grieztiesK");

    //   esc1.writeMicroseconds(1000);

    //   esc2.writeMicroseconds(1200);

    //   esc3.writeMicroseconds(1000);

    //   esc4.writeMicroseconds(1200);

    //   lastCommand = incomingByte;

    // }

    // else if (incomingByte == 's' && lastCommand != 's') {

    //   Serial.println("stop");

    //   esc1.writeMicroseconds(1100);

    //   esc2.writeMicroseconds(1100);

    //   esc3.writeMicroseconds(1100);

    //   esc4.writeMicroseconds(1100);

    //   lastCommand = incomingByte;

    // }

    // else if (incomingByte == 'e' && lastCommand != 'e') {

    //   Serial.println("prieksu");

    //   esc1.writeMicroseconds(1000);

    //   esc2.writeMicroseconds(1000);

    //   esc3.writeMicroseconds(1200);

    //   esc4.writeMicroseconds(1200);

    //   lastCommand = incomingByte;

    // }

    // else if (incomingByte == 'f' && lastCommand != 'f') {

    //   Serial.println("apakal");

    //   esc1.writeMicroseconds(1200);

    //   esc2.writeMicroseconds(1200);

    //   esc3.writeMicroseconds(1000);

    //   esc4.writeMicroseconds(1000);

    //   lastCommand = incomingByte;

    // }

    // else if (incomingByte == 'g' && lastCommand != 'g') {

    //   Serial.println("labi");

    //   esc1.writeMicroseconds(1200);

    //   esc2.writeMicroseconds(1000);

    //   esc3.writeMicroseconds(1200);

    //   esc4.writeMicroseconds(1000);

    //   lastCommand = incomingByte;

    // }

    // else if (incomingByte == 'h' && lastCommand != 'h') {

    //   Serial.println("kreisi");

    //   esc1.writeMicroseconds(1000);

    //   esc2.writeMicroseconds(1200);

    //   esc3.writeMicroseconds(1000);

    //   esc4.writeMicroseconds(1200);

    //   lastCommand = incomingByte;

    // }

    // else if (incomingByte == 'z' && lastCommand != 'z') {

    //   Serial.println("halt");

    //   esc1.writeMicroseconds(1100);

    //   esc2.writeMicroseconds(1100);

    //   esc3.writeMicroseconds(1100);

    //   esc4.writeMicroseconds(1100);

    //   lastCommand = incomingByte;

    // }

  }

  // Check distance and transition to hover if more than 40

  if (isAscending && distance > 40) {

    hover();

    droneState = HOVERING;

    isAscending = false;

    lastCommand = 0;

  }

  // Default behavior: Ensure motors are off when no command is active and not hovering

  if (droneState == IDLE) {

    stopMotors();

  }

}

**Telefona vadāma lidrobota**

#include <Servo.h>

const int pin45 = 45;

char burts;

bool pogaVal;

Servo esc1;

Servo esc2;

Servo esc3;

Servo esc4;

bool motorsRunning = false;

void setup() {

pinMode(pin45, INPUT\_PULLUP);

Serial3.begin(38400);

Serial.begin(38400);

esc1.attach(25);

esc2.attach(29);

esc3.attach(33);

esc4.attach(37);

esc1.writeMicroseconds(1000);

esc2.writeMicroseconds(1000);

esc3.writeMicroseconds(1000);

esc4.writeMicroseconds(1000);

delay(5000);

}

void loop() {

pogaVal = digitalRead(pin45);

if (Serial3.available() > 0) {

burts = Serial3.read();

Serial.println(burts);

if (burts == 'a' && !motorsRunning) {

motorsRunning = true;

takeoff();

} else if (burts == 'b') {

stopMotors();

motorsRunning = false;

}

}

if (motorsRunning) {

// Hovering phase

hover();

// Landing phase

land();

motorsRunning = false;

}

}

void takeoff() {

// Increase throttle gradually

for (int pulseWidth = 1000; pulseWidth < 1300; pulseWidth += 3) {

esc1.writeMicroseconds(pulseWidth);

esc2.writeMicroseconds(pulseWidth);

esc3.writeMicroseconds(pulseWidth);

esc4.writeMicroseconds(pulseWidth);

delay(50); // Short delay between updates

}

}

void hover() {

// Hovering at 1300 microseconds for 3 seconds

for (int i = 0; i < 20; i++) { // 60 \* 50ms = 3000ms (3 seconds)

esc1.writeMicroseconds(1300);

esc2.writeMicroseconds(1300);

esc3.writeMicroseconds(1300);

esc4.writeMicroseconds(1300);

delay(50); // Short delay between updates

}

}

void land() {

// Decrease throttle gradually

for (int pulseWidth = 1300; pulseWidth > 1000; pulseWidth -= 2) {

esc1.writeMicroseconds(pulseWidth);

esc2.writeMicroseconds(pulseWidth);

esc3.writeMicroseconds(pulseWidth);

esc4.writeMicroseconds(pulseWidth);

delay(50); // Short delay between updates

}

}

void stopMotors() {

// Set all motors to the minimum throttle (shutdown)

esc1.writeMicroseconds(1000);

esc2.writeMicroseconds(1000);

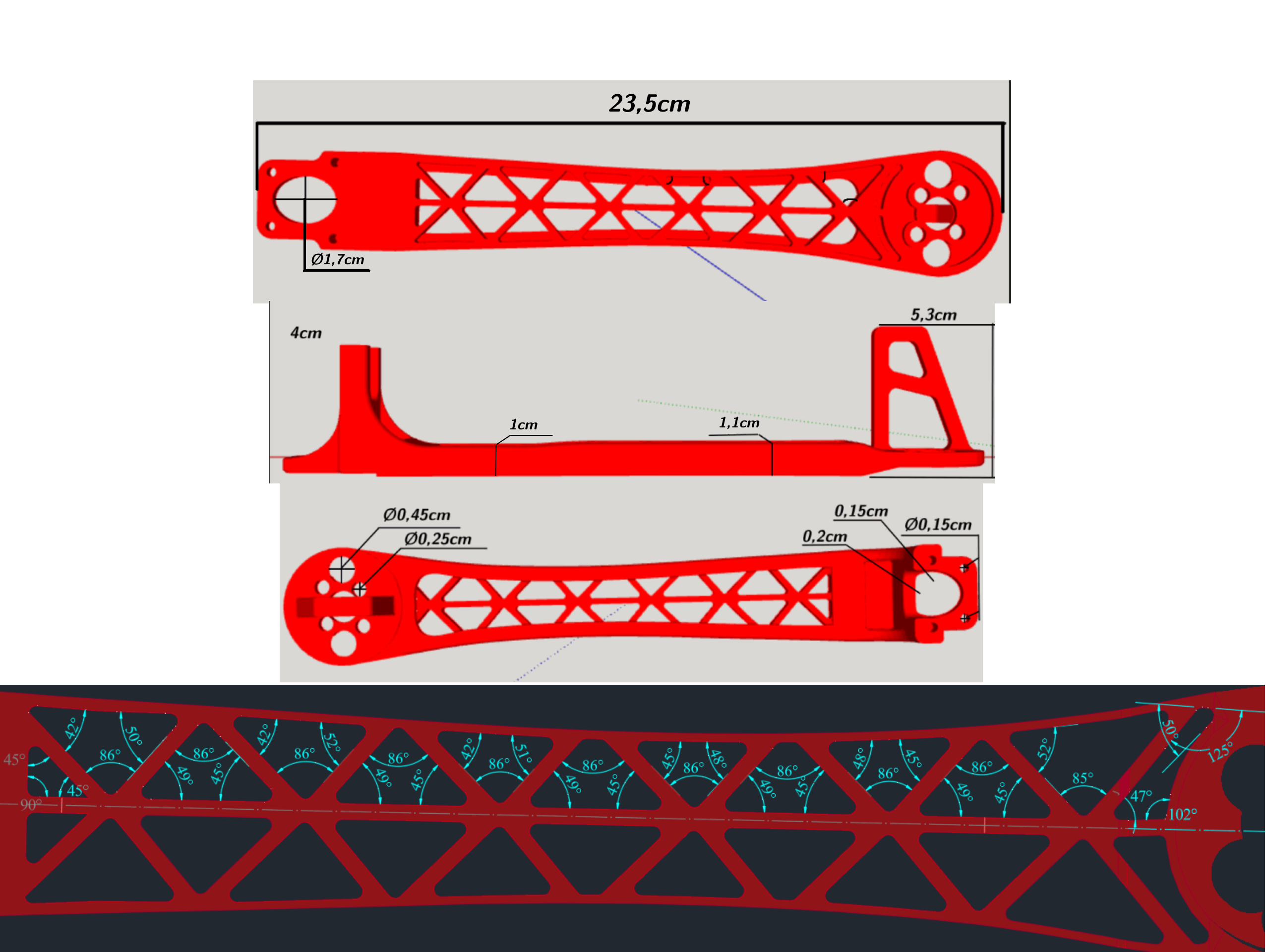
esc3.writeMicroseconds(1000);

esc4.writeMicroseconds(1000);

}

**Rasējumi**

**Lidrobota kāja**

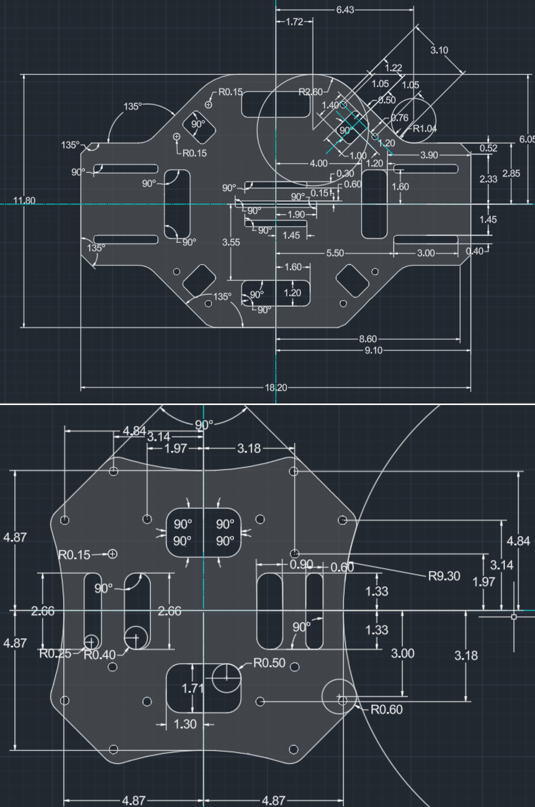


**Propellers**

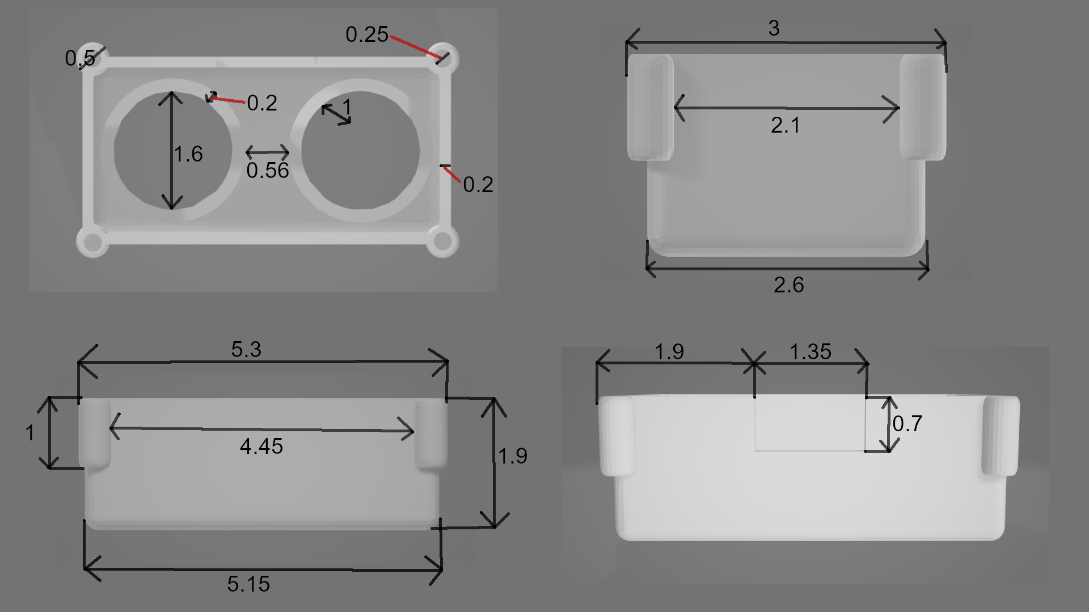
A drawing of a rectangle and a rectangle with a square and a square with a square and a square with a square and a square with a square and a square with a square and a

Description automatically generated

**Lidrobota plāksnes**



**Ultraskaņas sensora kaste**



**Ultraskaņas sensora kastes vāks**

