AGH University of Science and Technology

Faculty of Mechanical Engineering and Robotics

Mechatronic Design

Project of the Automatic Pet Feeder 

Team members:

- Ivan Rybalko
- Michał Nowicki
- KyiMin Thant

Summary:

1. Introduction and Project Goals

An automatic pet feeder is a device that ensures regular feeding and water supply for a pet without direct human involvement. Our automatic pet feeder offers much more than just a basic timer-based feeder: it is an intelligent system controlled via a mobile application, equipped with video monitoring, sensor-based water and food level detection, and a fully customizable feeding schedule.

Project Goals:

- Simplify pet care routines;
- Provide remote monitoring and control;
- Prioritize sustainability using recyclable materials;

The automatic pet feeder is designed to meet the daily needs of owners of small to medium-sized pets. In modern households, pet owners often face challenges balancing their responsibilities with consistent feeding schedules, proper portion control, and real-time monitoring of their pet's well-being. This project addresses those challenges by offering a highly automated, connected, and user-friendly solution

2. Morphological Analysis

Morphological Analysis Table - Design

Aspects/Functions	Option 1 (Basic Feeder)	Option 2 (Mid-range)	Option 3 (Advanced Feeder)	Option 4 (Smart Feeder App-Controlled) ☑	Option 5 (Eco-Friendly Premium)
Food dispensing	Gravity-fed (5)	Manual dial portion control (7)	Programmable mechanical system (8)	**Web/Automatic app-controlled dispensing (10)**	Precise digital portion control (10)
Water dispensing	Simple bowl (3)	Manual valve (6)	Timed automatic valve (7)	**Web/App-controlled water pump (9)**	Hygienic water circulation (10)
Control mechanism	Manual (4)	Timer-based (7)	Simple electronic panel (8)	**Smartphone/Web app control (10)**	App + Voice command (10)
Materials	Basic plastic (3)	Standard plastic/metal (5)	Food-safe plastic & metal (7)	**Plywood + biodegradable filament (10)**	Premium eco-certified materials (10)
Scheduling options	None (1)	Basic timer (6)	Multi-setting timer (8)	**Customizable app schedules (10)**	Adaptive AI-based scheduling (10)
Connectivity	None (1)	Wired power only (5)	Battery-operated wireless (7)	**Wi-Fi app connectivity (10)**	Smart Home Integration (10)
Eco-Friendliness	Low (1)	Moderate (4)	Moderate-High (6)	**High (8)**	Maximum (biodegradable, recyclable) (10)
Target Dog Size	Universal (6)	Large dogs (4)	Medium dogs (7)	**Small to medium dogs (10)**	Customizable portions (all sizes) (10)

Functionality (1-10), Price (10-1):

Option 4 offers the best balance of functionality, technology, and value. With app-controlled food and water dispensing, customizable schedules, and Wi-Fi connectivity, it provides smart automation without the premium cost of Option 5. It also uses eco-friendly materials, making it both practical and sustainable. Overall, Option 4 is ideal for modern users seeking convenience, control, and efficiency.

Morphological Analysis Table Electronics

Functions / Components	Option 1	Option 2	Option 3	Option 4 ☑	Option 5
Controller & Connectivity	Basic Arduino (No connectivity) (3 🟢)	Raspberry Pi (Wi-Fi) (7 🟢)	Arduino Uno (WIFI) (9 🟢)	**ESP32 standalone (10 🟢)**	PLC (industrial) (6 🟢)
Food Level Detection	No sensor (manual check) (2 🟢)	Weight sensor (load cell) (6 🟢)	IR sensor (limited accuracy) (6 🟢)	**Ultrasonic sensor (accuracy, ease of use) (10 🟢)**	Laser distance sensor (high cost) (9 🟢)
Water Level Detection	Manual visual check/refill (2 🟢)	Float sensor (basic accuracy) (7 🟢)	Capacitive sensor (good accuracy) (8 🟢)	**Digital water level sensor (high accuracy, easy integration) (10 🟢)**	Ultrasonic sensor (complex for water) (8 🟢)
Food Dispensing Method	Gravity-fed (basic, no control) (3 🟢)	Servo motor + gate (7 🟢)	DC motor + auger (moderate control) (8 🟢)	**Stepper motor + feed screw (precise control, repeatable) (10 🟢)**	Pneumatic valve (expensive, noisy) (5 🟢)
Water Dispensing Method	Gravity bowl (no control) (2 🟢)	Solenoid valve (complex setup) (7 🟢)	Peristaltic pump (accurate, costly) (6 🟢)	**5-12V DC Water Pump Small Water Pump (efficient, reliable, compact)(10 🟢)**	Gravity-fed auto refill (limited control) (4 🟢)
Food Detection in Bowls	Manual check (no automation) (1 🟢)	Weight sensor (basic detection) (5 🟢)	IR sensor (limited accuracy) (4 🟢)	**Camera (ESP32-S3 with OV2640) (high accuracy, visual detection) (10 🟢)**	Camera + AI processing (complex, costly) (8 🟢)
Materials	Basic plastic (non-eco) (2 🟢)	Standard 3D filament (PLA) (6 🟢)	Acrylic/plexiglass (durable but non-eco) (5 🟢)	**Plywood + biodegradable 3D filament (eco, accessible) (10 🟢)**	Metal casing (expensive, heavy) (5 🟢)

Functionality (1-10), Price (10-1)

Option 4 stands out by combining high-performance components with smart automation at a reasonable cost. It uses the ESP32 controller for powerful connectivity, ultrasonic sensors for accurate food and water level sensor for water level detection, and stepper motors for precise dispensing making Option 4 the most balanced and advanced choice.

Morphological Analysis Table Mechanical Components

Functions / Components	Option 1	Option 2	Option 3 ☑	Option 4	Option 5
Case Material (Front/Back)	Basic plastic (4)	Standard 3D filament (6)	**Plywood (9)**	MDF (Medium-Density Fiberboard) (7)	Metal sheet (8)
Case Material (Sides)	Thin wood (4)	ABS filament (6)	**Bio-filament (7)**	Polycarbonate (7)	Aluminum (8)
Case Material (Container)	Glass (5)	PETG filament (7)	**Plexiglass + Biofilament (8)**	HDPE (High-Density Polyethylene) (6)	Stainless steel (9)
Assembly Method	Manual snap-fit (3)	Adhesive bonding (6)	**Screwed + Glued with epoxy (9)**	Bolted (7)	Welded (8)

Functionality (1-10), Price (10-1):

Option 3 offers the most practical and well-balanced mechanical design. It uses plywood for the front/back—strong, versatile, and cost-effective—and bio-filament sides, combining durability with eco-friendliness. The container is made of 3d printed filament with plexiglass windows , offering both strength and visibility. For assembly, it uses screwed and epoxy-glued joints, ensuring strong and reliable construction. This configuration provides a solid balance of durability, sustainability, and moderate cost, making Option 3 the most efficient and well-rounded mechanical solution.

2.1 Key Differentiators from Competitors

The automatic pet feeder stands out in a competitive market by offering a balanced blend of intelligent automation, sustainability, and user-centric design. While many feeders on the market offer either mechanical simplicity or app-based scheduling, our solution integrates multiple high-value features into one cohesive and accessible system.

Intelligent Features

- App-Controlled Dispensing : Food delivery is fully automated and triggered through a web or smartphone app interface.
- Water Pump Control : The water pump can also be activated remotely, ensuring pets always have access to fresh water.
- Live Camera Monitoring: Unlike many feeders, ours includes live video streaming via Seeed Xiao ESP32-S3 Sense - with OV2640 camera for real-time supervision.

User-Focused Interface

- Smartphone/Web App Control : The system is designed to be intuitive and fully operable through mobile devices.
- Customizable Schedules : Feeding times and water dispensing are easily programmable, down to the minute.

Eco-Friendly and Durable Construction

- Plywood + Biodegradable Filament : The enclosure is constructed from sustainable materials, prioritizing environmental responsibility without compromising durability.
- Compact and Targeted for Household Pets: Specifically optimized for small to medium-sized dogs

2.2 Advantages:

Remote Control and Monitoring: The feeder can be managed entirely via a smartphone or web application. Owners can dispense food or water, view current levels, and even activate a live video stream remotely.
Custom Schedules: Feeding and hydration times can be scheduled in advance, eliminating the risk of missed meals.
Sensor-Based Precision: Ultrasonic and water level sensors provide real-time feedback on how much food and water remains, ensuring pets are never left without resources.
Eco-Friendly Construction: Physical design is made using plywood and biodegradable PLA filament, reducing plastic waste and supporting sustainable practices.

3. Core Design

3.1 Concept

Concept proposed by the team, combining the best solutions identified during the market review along with justification, pros and cons compared to other solutions, including a simplified schematic of mechanical and electronic components

Dual-Container System: The feeder features a plexiglass container divided into two compartments — one for water, one for dry food. This design enables independent dosing of each resource while providing visual feedback for users to monitor levels.

Screw-Driven Dispensing Mechanism: Dry food is dispensed using a custom-designed feed screw , which is driven by a stepper motor for high precision and repeatability. This ensures consistent portioning without clogging or overfeeding.

Water Pumping System with Relay: A 5-12V DC water pump controlled via a relay module allows for controlled water dispensing. The relay is directly actuated by the ESP32-S3 board without the need for external transistors.

Eco-Friendly Structure: The outer body is built from plywood and biodegradable 3D printed filament. Only the container uses plexiglass for visibility, making the construction sustainable and environmentally friendly.

Cloud Integration (Arduino Cloud): The system uses Arduino Cloud for real-time control and monitoring. Through a smartphone or web app, users can view food/water levels, schedule feeding times, and trigger actions remotely.

Live Video Monitoring:
An Seeed Xiao ESP32-S3 Sense - with OV2640 camera module, connected via Wi-Fi, streams video to a web browser, allowing real-time observation of the pet and food tray. The camera works independently from the ESP32-S3.

Scheduling:
Feeding and watering times can be configured with hour, minute, and date via the dashboard. These settings are stored and processed by the ESP32-S3, which executes the schedule even without an active internet connection.

3.2 Drawings

Below is the exploded view of our project – an automatic pet feeder. The design focuses on sustainability, functionality, and ease of manufacturing, as reflected in both the material selection and the production methods used. All parts were designed in CAD software

As illustrated in Figure 1, the feeder consists of five key components, each made from different materials and manufactured using appropriate techniques:

1. Side Cylindrical Holders Parts 1 & 2:
   These parts were 3D printed using BIO-filament, a biodegradable and eco-friendly material. BIO-filament was chosen due to its minimal environmental impact and good structural properties The holders provide support for the rotation mechanism that controls the release of food.

2. Front and Back Plates Parts 3 & 4:
   These elements were laser-cut from plywood. Plywood is lightweight, affordable, and sustainable, making it ideal for our needs. Laser cutting enabled us to achieve precise shapes and dimensions, including holes and slots necessary for assembly.

3. Food Container Part 6:
   The container is made of 3d printed filament with plexiglass windows , chosen for its transparency, light weight, and easy machinability. Transparency allows the user to quickly check the remaining food level. The plexiglass was cut to size and manually assembled using adhesive suitable for acrylic materials.

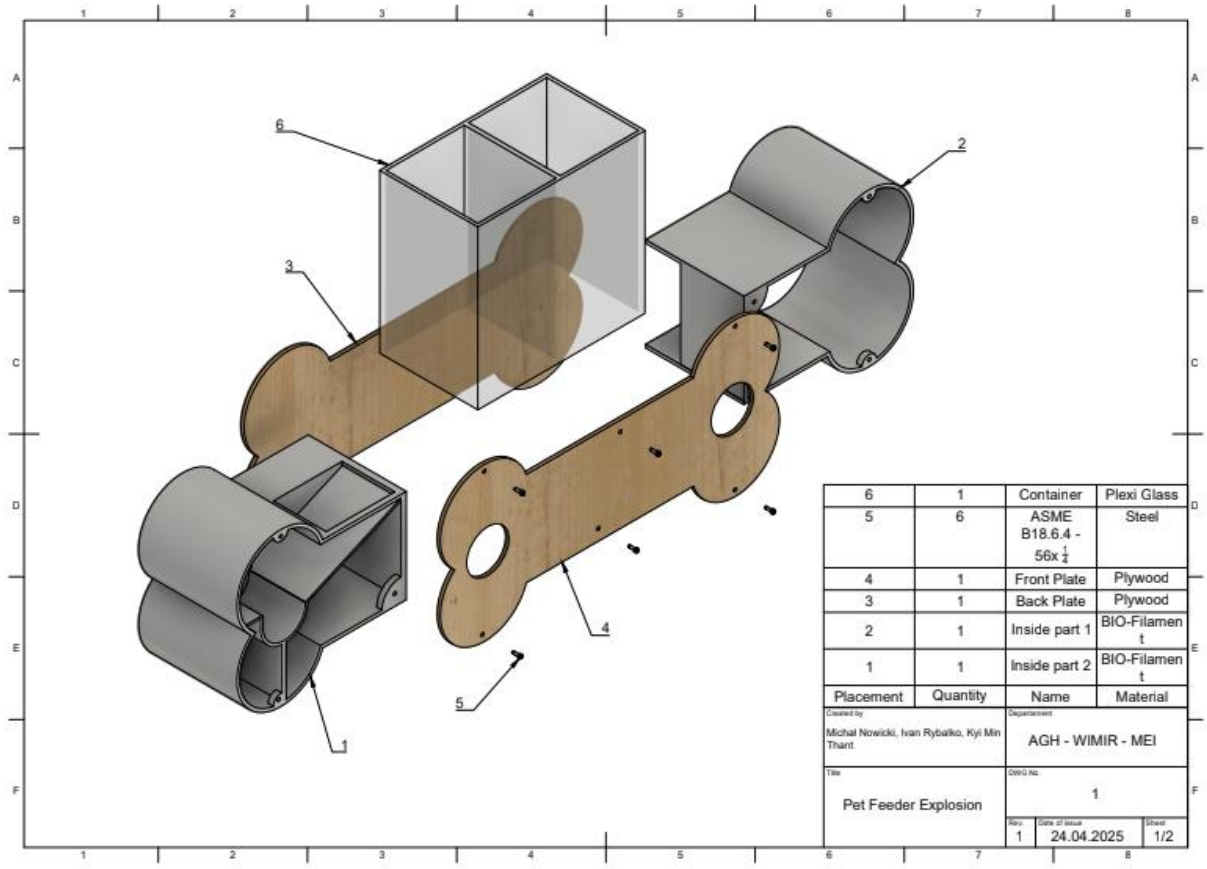
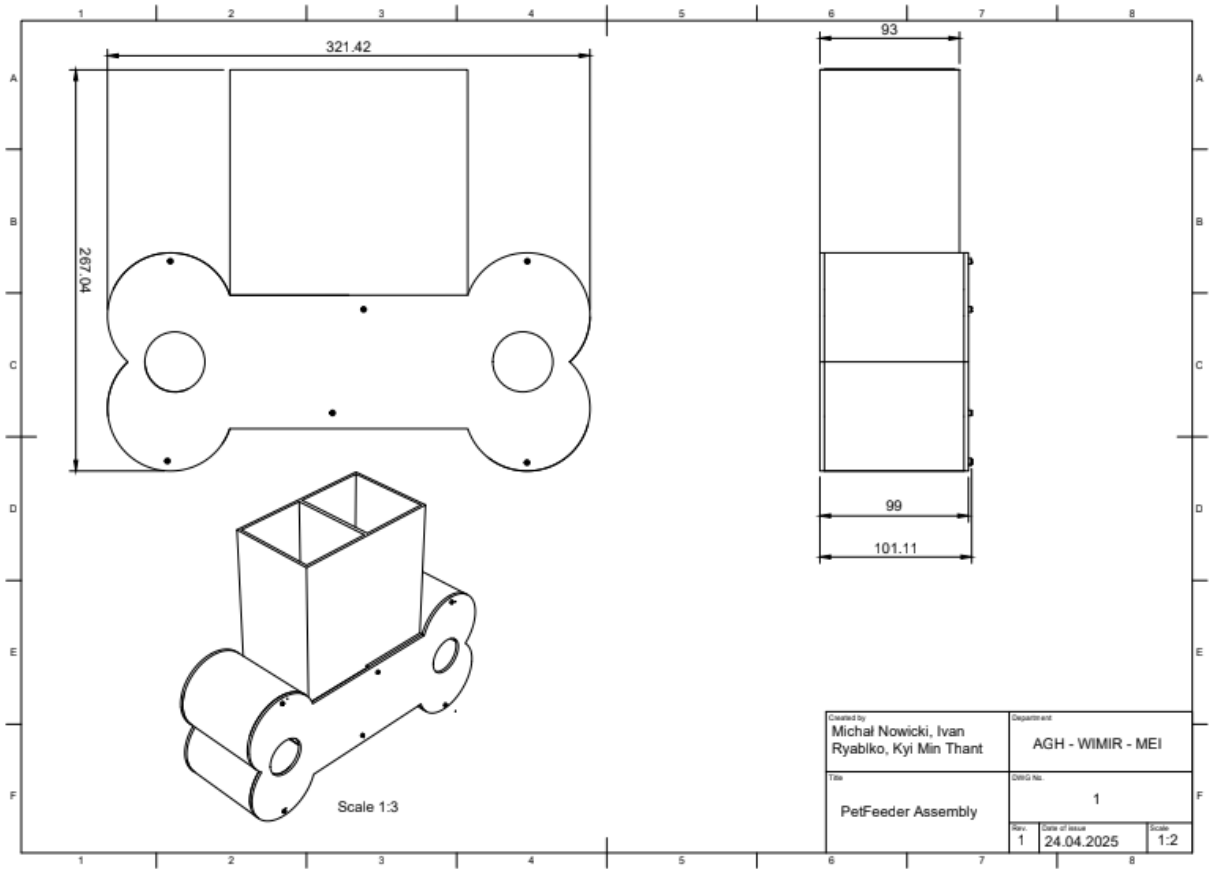

6
2
3
4
5
1
6 1 Container Plexi Glass
5 6 ASME B18.6.4 - 56x 1/2 Steel
4 1 Front Plate Plywood
3 1 Back Plate Plywood
2 1 Inside part 1 BIO-Filament
1 1 Inside part 2 BIO-Filament
Placement Quantity Name Material
Created by: Michał Nowicki, Ivan Ryablko, Kyi Min Thant
Department: AGH - WIMIR - MEI
Title: Pet Feeder Explosion
DWG No.: 1
Rev.: 1
Date of issue: 24.04.2025
Sheet: 1/2

Figure 1.


321.42
267.04
93
99
101.11
Scale 1:3
Created by: Michał Nowicki, Ivan Ryablko, Kyi Min Thant
Department: AGH - WIMIR - MEI
Title: PetFeeder Assembly
DWG No.: 1
Rev.: 1
Date of issue: 24.04.2025
Scale: 1:2

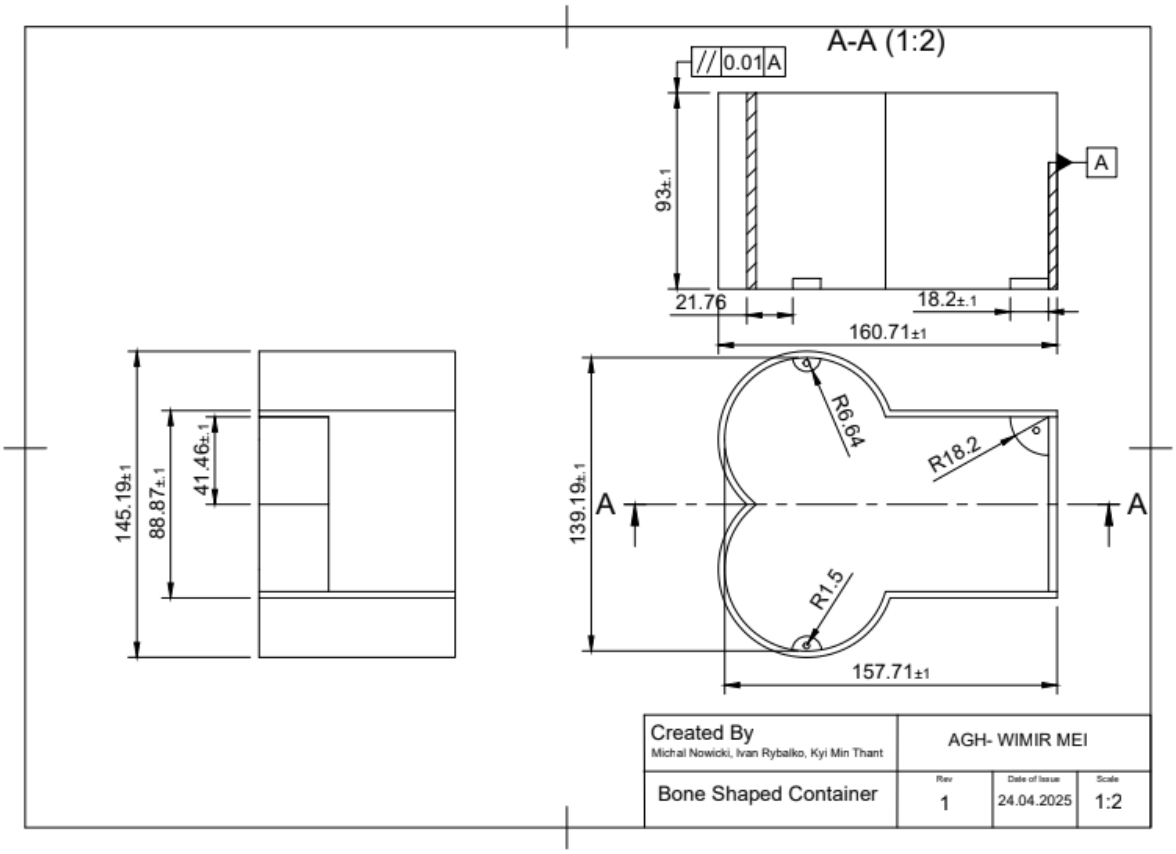

A-A (1:2)
// 0.01 A
93±.1
A
21.76
18.2±.1
160.71±1
R6.64
R18.2
139.19±1
A
A
R1.5
157.71±1
145.19±1
88.87±.1
41.46±.1
Created By
Michal Nowicki, Ivan Rybalko, Kyi Min Thant
AGH- WIMIR MEI
Bone Shaped Container
Rev
1
Date of Issue
24.04.2025
Scale
1:2

Figure 3.

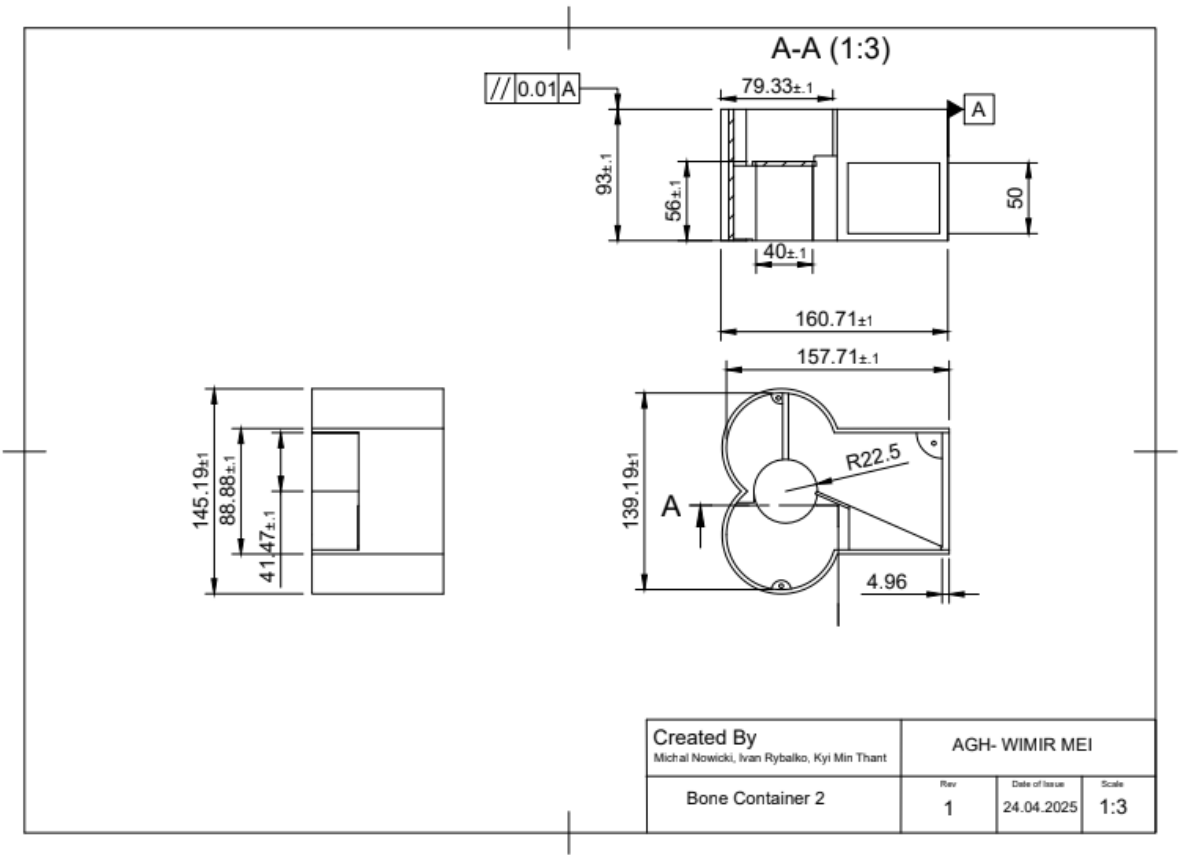

A-A (1:3)
// 0.01 A
79.33±.1
A
93±.1
56±.1
50
40±.1
160.71±1
157.71±.1
R22.5
139.19±1
A
4.96
145.19±1
88.88±.1
41.47±.1
Created By
Michal Nowicki, Ivan Rybalko, Kyi Min Thant
AGH- WIMIR MEI
Bone Container 2
Rev
1
Date of Issue
24.04.2025
Scale
1:3

Figure 4.

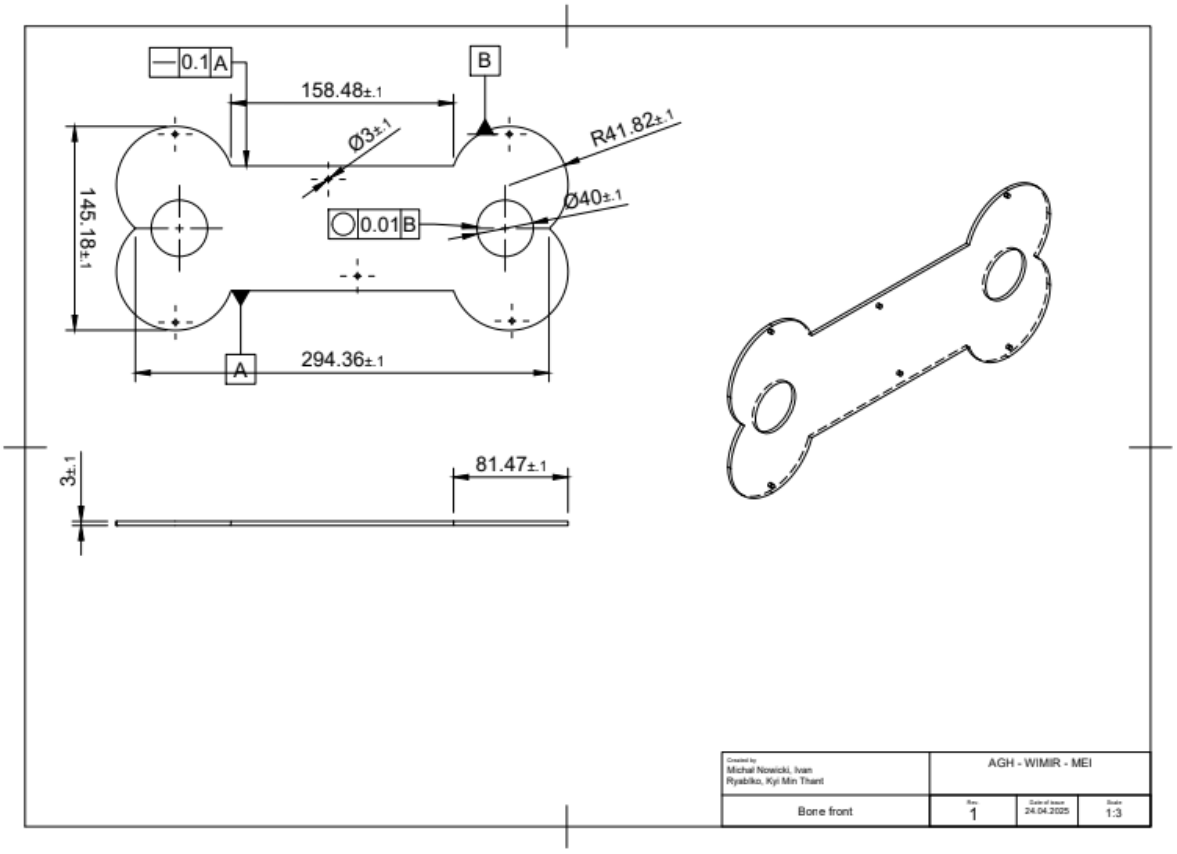


Figure 5.

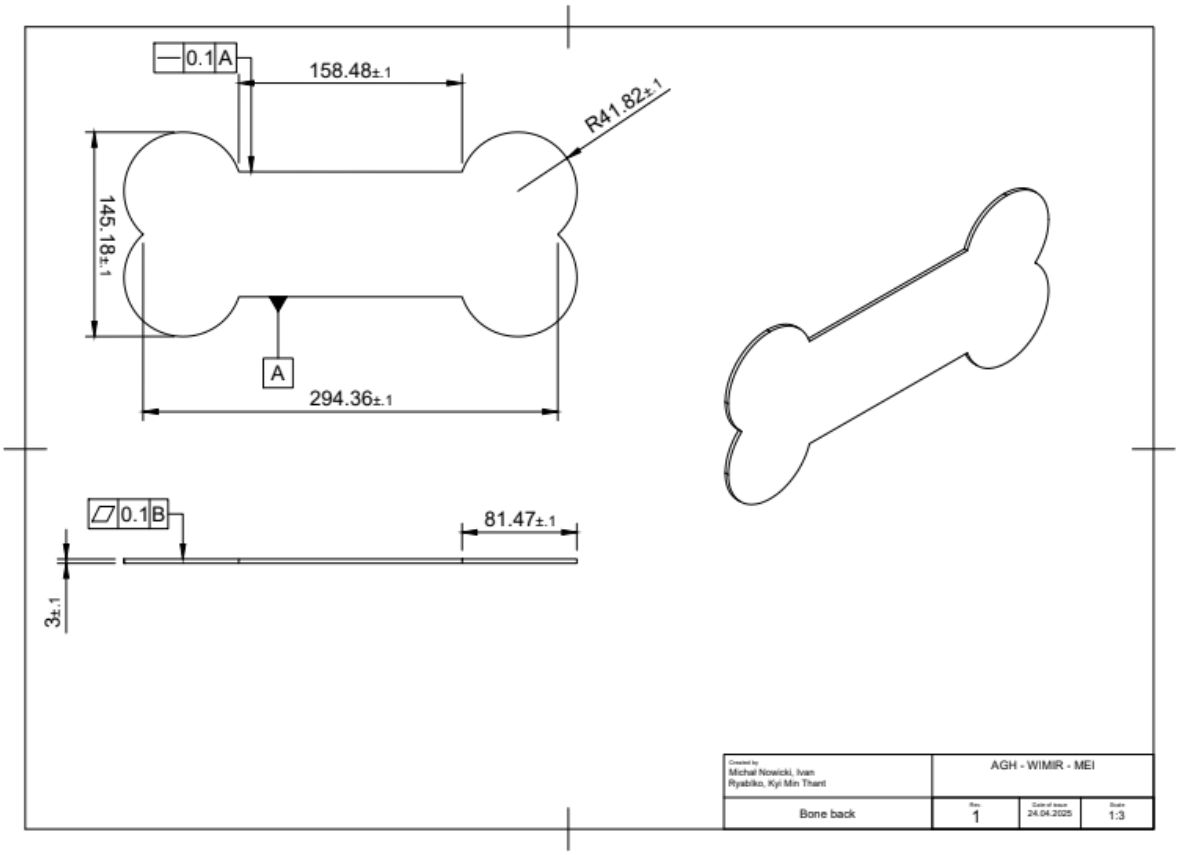


Figure 6.

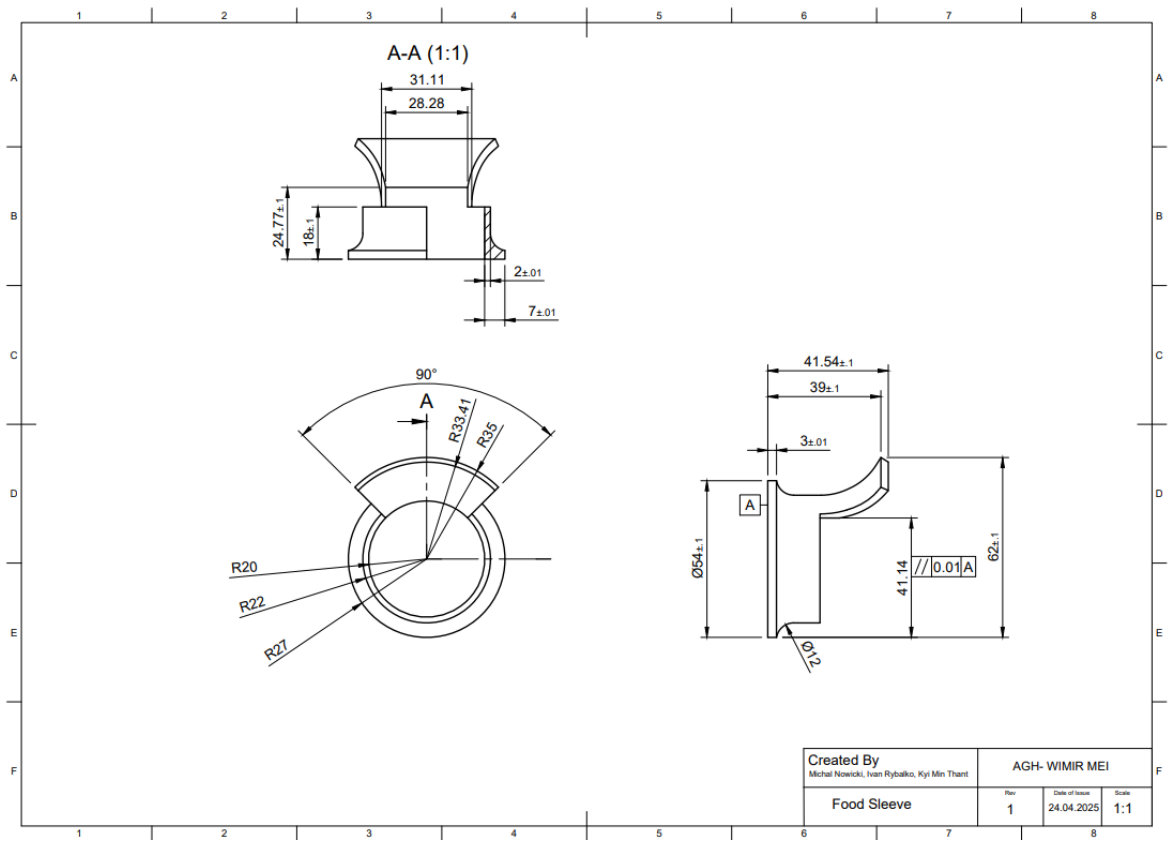
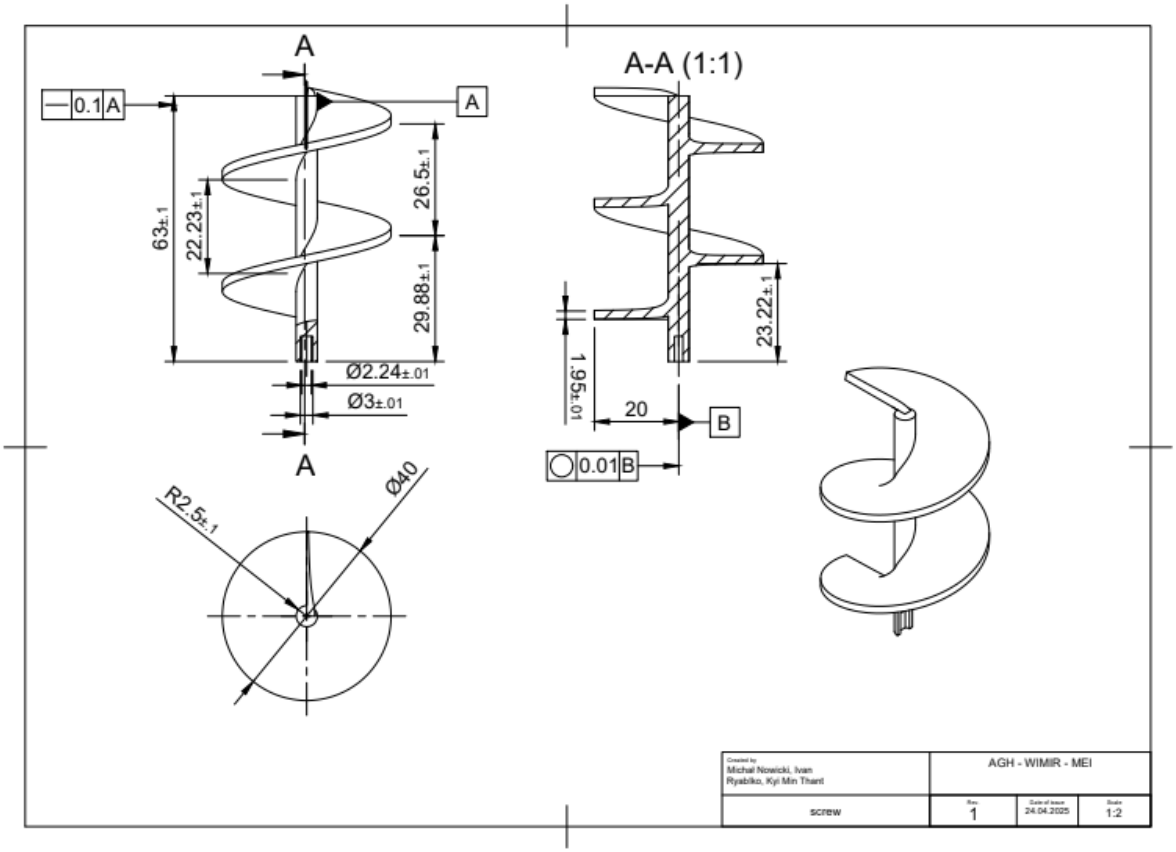


Figure 7.



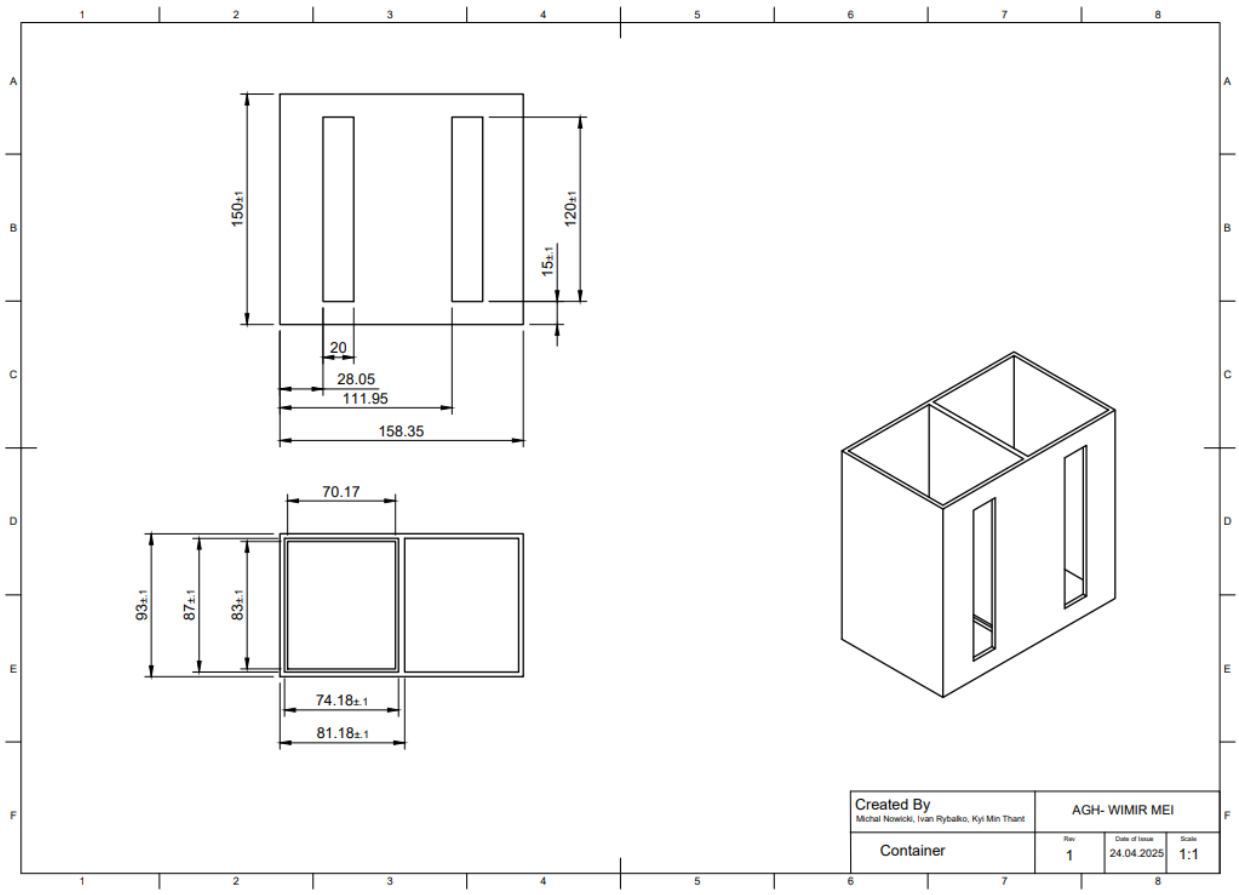
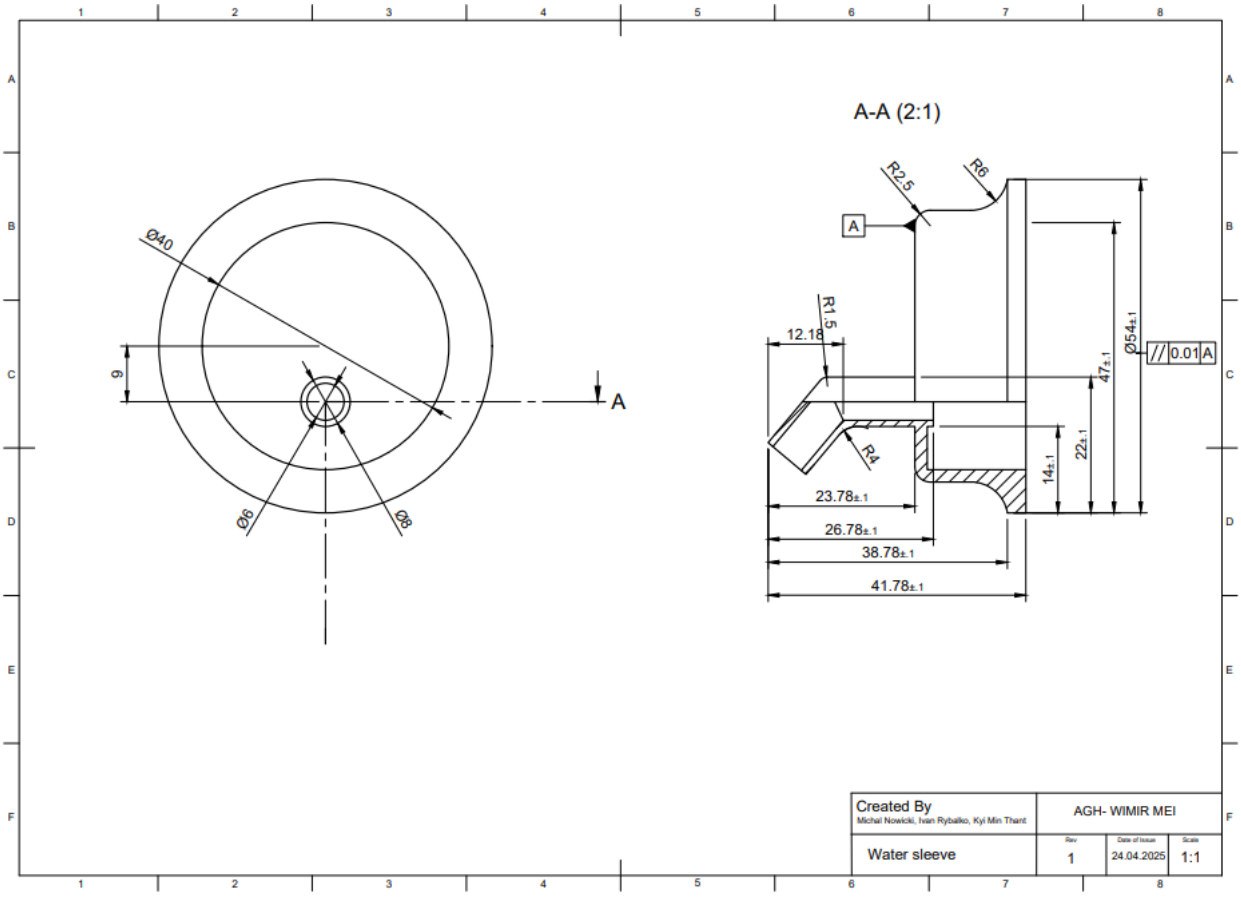

150±1
120±1
15±1
20
28.05
111.95
158.35
70.17
93±1
87±1
83±1
74.18±1
81.18±1
Created By
Michał Nowicki, Ivan Rybalko, Kyi Min Thant
AGH- WIMIR MEI
Container
Rev
1
Date of Issue
24.04.2025
Scale
1:1

Figure 9.


A-A (2:1)
Ø40
9
A
Ø6
Ø8
R2.5
R6
A
R1.5
12.18
R4
23.78±1
26.78±1
38.78±1
41.78±1
14±1
22±1
47±1
Ø54±1
// 0.01 A
Created By
Michał Nowicki, Ivan Rybalko, Kyi Min Thant
AGH- WIMIR MEI
Water sleeve
Rev
1
Date of Issue
24.04.2025
Scale
1:1

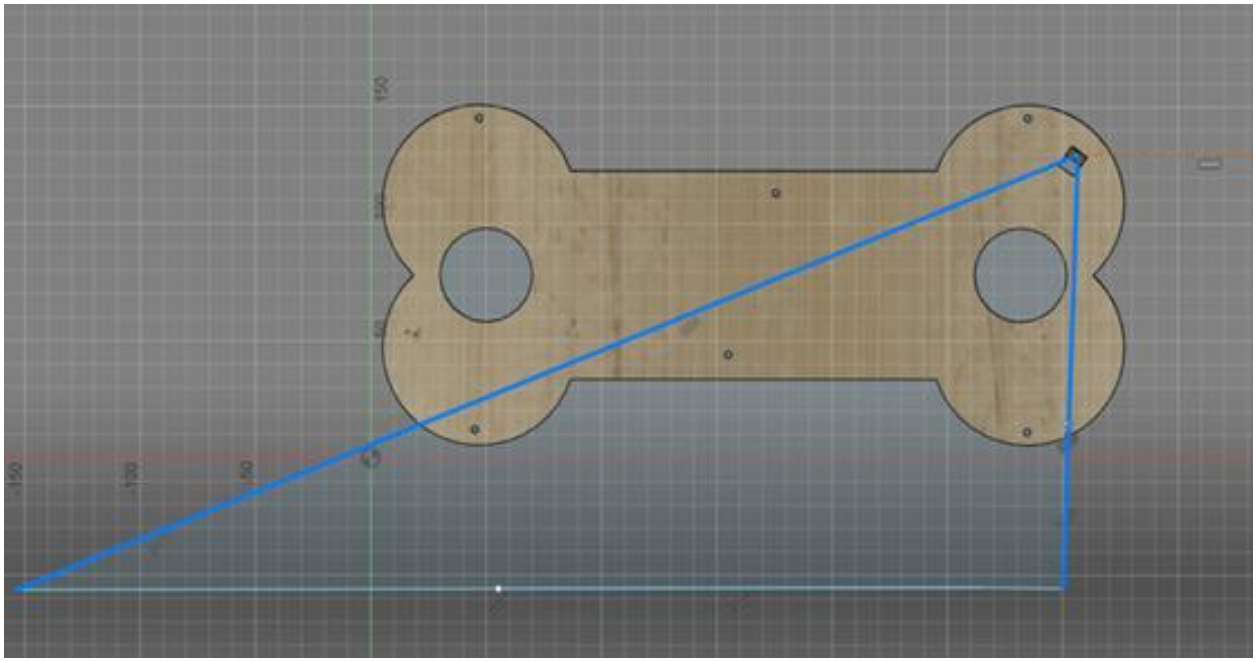


Figure 11.

Presented in Figure 11 is the 66-degree wide angle of camera view. It was positioned to fully cover the area with both feeding bowls.

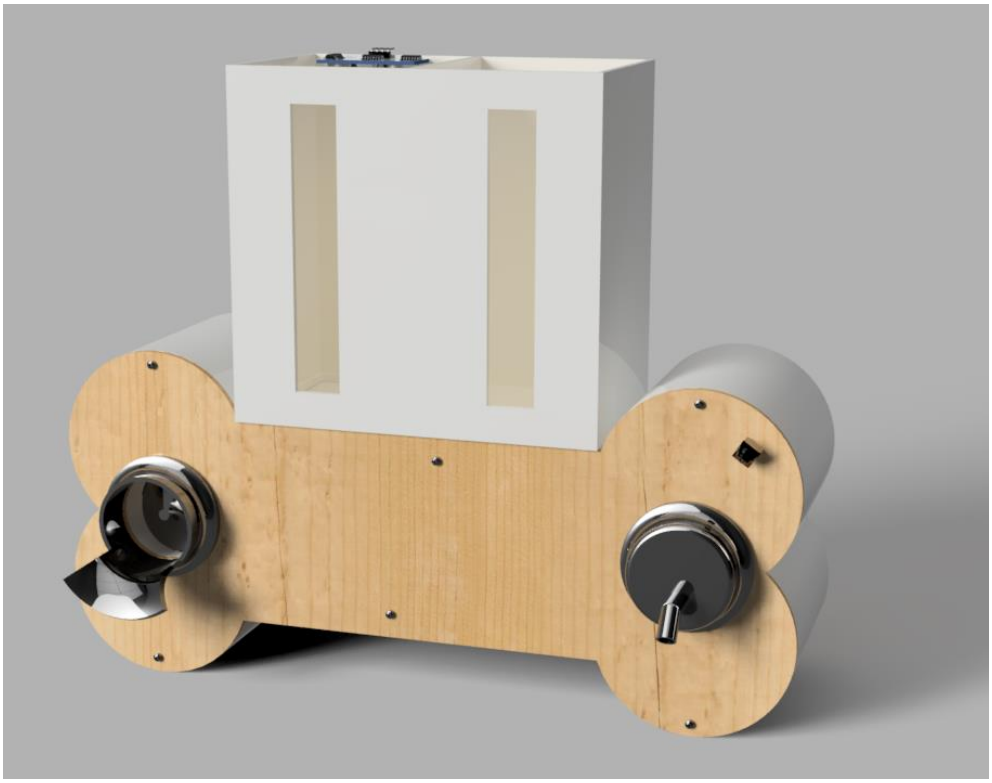


Figure 12.

In Figure 12, the fully assembled model can be seen, created and visualized in the CAD software. This final assembly view demonstrates how all individual components fit together and provides a clear representation of the functional layout of the pet feeder.

4. System Overview and Components

4.1 System Overview

The ESP32 board controls and connects to various components,

such as the water level sensor, ultrasonic sensor, stepper motor driver and relay module.

Water level sensor is communicating via SDA(serial DATA Line) and SCL(serial clock line) pins, in reality it looks a bit different that this presented in Figure12 but its connected in correct way.

Whole system is powered by 5V 4A power supply. Estimated current draw of a system is around, 3 Amps.

Relay module is designed to be controlled directly by pins without need of additional transistors.

Seeed Xiao ESP32-S3 Sense - with OV2640 camera is not controlled via main microconstroller. Its streaming image to html site via Wifi and will be accessed by software

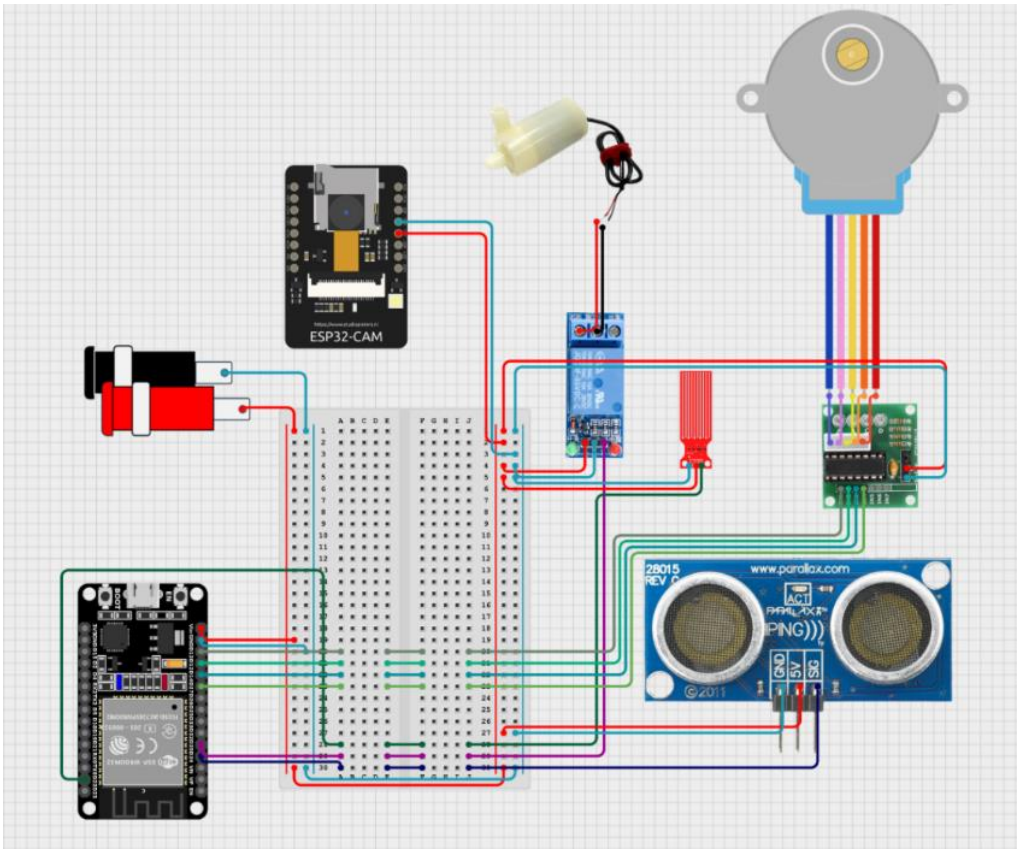


Figure 13.

4.2 Components

Component	Purpose	Justification
ESP32-S3	Main controller	Powerful, supports Wi-Fi, BLE, I2C
Seeed Xiao ESP32-S3 Sense - with OV2640 camera	Independent camera module	Easy Wi-Fi HTML streaming
28BYJ-48 + ULN2003 driver	Stepper motor for food dosing	Accurate, stable, low-power
5-12V DC Water Pump	Dispenses water	Compact and efficient
Iduino Relay Module	Controls pump power	Simple, works with ESP32 directly
HC-SR04 Ultrasonic Sensor	Measures remaining food	Low cost, reliable
Water Level Sensor	Measures water level	High precision, easy I2C integration
5V 4A Power Supply	Powers the full system	Stable enough for ~2.5 A peak load

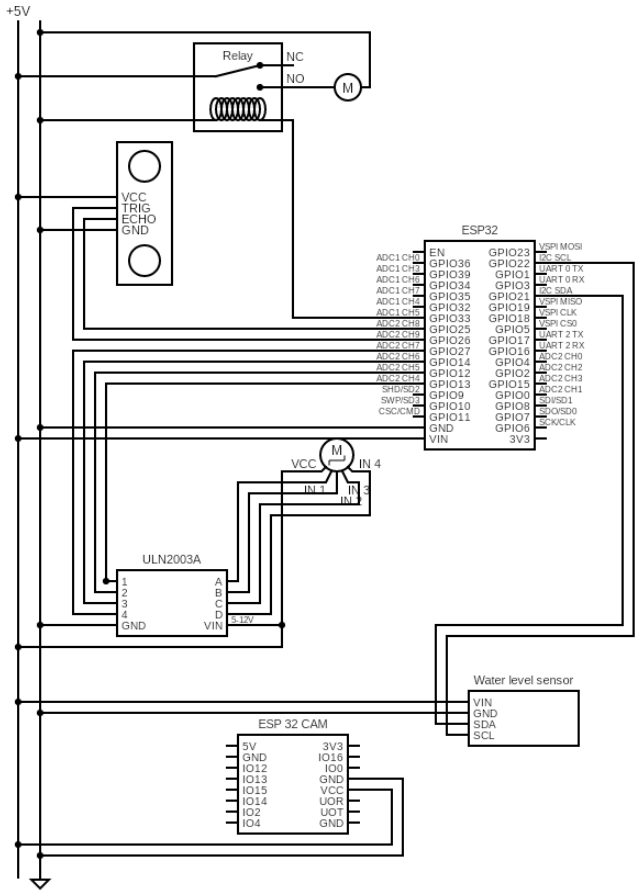


Figure 14

4.3 Power and Current Consumption

The entire system is powered by a 5V 4A switching power supply. Based on our measurements and manufacturer specifications, the average current draw is:

Component	Consumption
ESP32-S3 Sense	200 mA
Stepper Motor + ULN2003	900 + 100 mA
HC-SR04	20 mA
Relay Module	75 mA
Grove Water Sensor	20 mA
Seeed Xiao ESP32-S3 Sense - with OV2640 camera	50 mA
5-12V DC Water Pump	1200 mA
Total Average	≈ 2565 mA

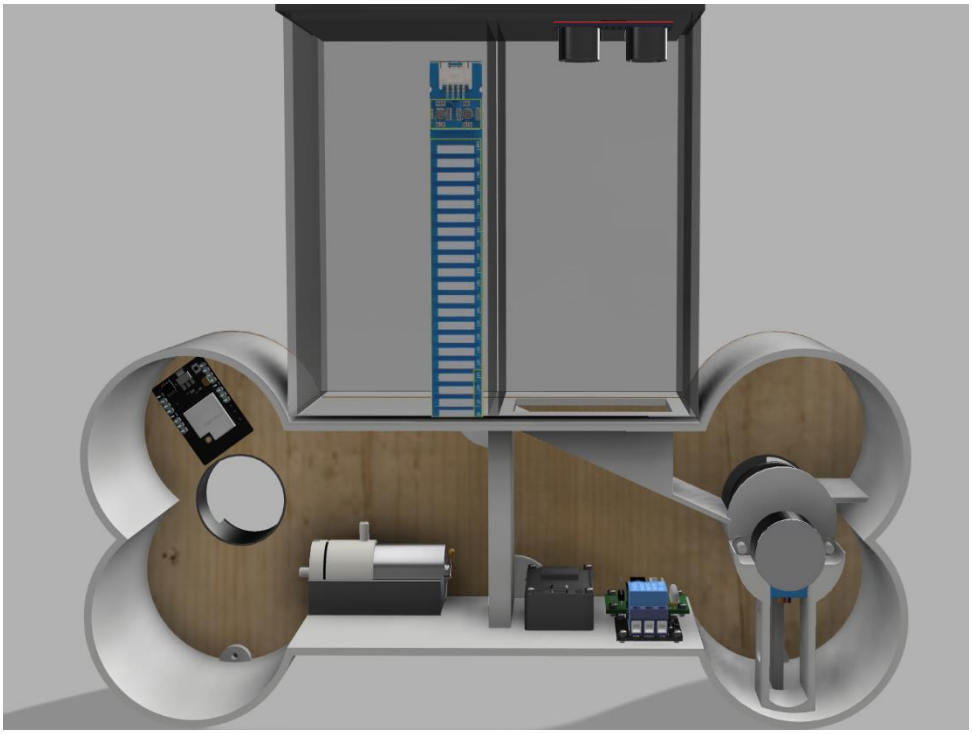


Figure 15

Figure 15 provides a close-up view. Wires connections are not visible to make the placement more clearly to see. All electrical connections are fully detailed in schemes Figures 13 and 14

5. Software – Arduino Cloud

The entire system logic of the automatic pet feeder is implemented using Arduino Cloud, an integrated IoT platform that allows remote device control, real-time data visualization, and scheduling—all accessible through a web dashboard or mobile app.

Cloud Integration Overview

Arduino Cloud handles: Device management (ESP32-S3 Sense); Live synchronization of variables; Scheduling and time-based tasks; User interface rendering (custom dashboard widgets).

System Features and Logic:

Feeding and Water Dose	Controlled via sliders; values are read by the ESP32-S3 and trigger actuators
Level Monitoring	Sensor data (HC-SR04 & Grove Water Sensor) is displayed live in dashboard
Schedule Configuration	User selects hour/minute/date via time picker; stored in cloud and synced
Camera Activation	A dashboard button opens local HTML site with camera image is tunnelled through ngrok software to make it available publicly with https protocol

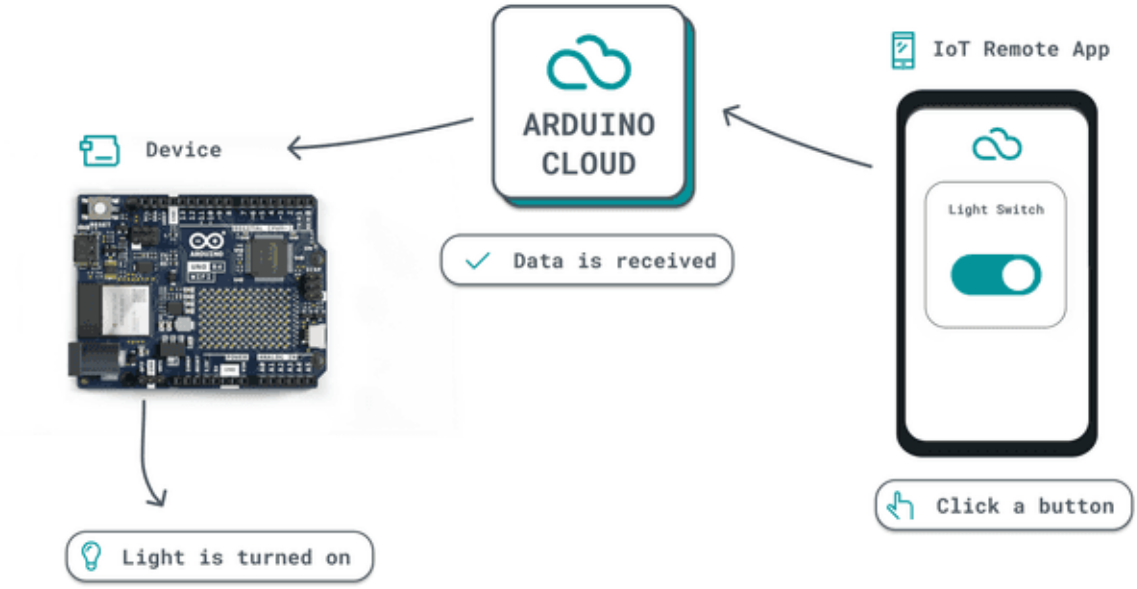


Figure 16

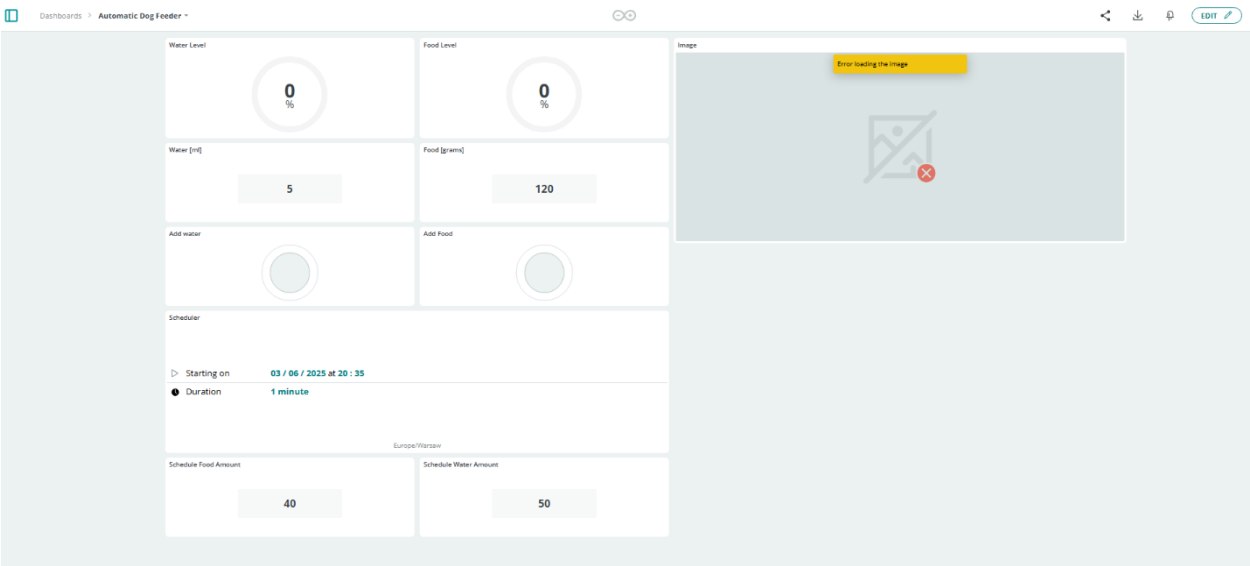


Figure 17

Figure 17 shows the interface concept developed in Arduino Cloud for controlling the device. This cloud-based dashboard allows real-time monitoring and remote control of key system functions, such as food dispensing, water level status. The interface is designed to be user-friendly and accessible from any internet-connected device.

Cloud Integration

The device connects to the Arduino IoT Cloud using Wi-Fi credentials and unique device keys. Several cloud variables are defined to report water and food levels and to receive user commands and schedules. The initProperties() function registers these variables, enabling seamless communication between the device and cloud dashboard.

Sensor Inputs

Two types of sensors provide environmental feedback:

- Ultrasonic Sensor measures the distance to the food surface, converting it into a percentage with calibration to account for empty bin offset.
- Grove I²C Water Sensor detects the number of “wet” pads across two sections, calculating the water level as a percentage based on active segments.

Actuation

The system dispenses:

- Water via a relay-controlled pump, with timing based on a fixed flow rate (PUMP_FLOW_ML_PER_MIN).
- Food using a stepper motor, where the number of steps is proportional to the required grams (FOOD_PER_REV_G).

Safety and Scheduling

Manual activation is throttled by a 10-second delay to prevent overuse. Additionally, a cloud-based schedule trigger can initiate both food and water dispensing, enabling automated daily routines

5.1 Working program code

```
#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>
#include <WiFi.h>
#include <Wire.h>
#include <AccelStepper.h>
const char SSID[]   = "UPC3216837";
const char PASS[]   = "WifiPass";
const char THING_ID[]           = "############################";
const char DEVICE_LOGIN_NAME[] = "############################";
const char DEVICE_KEY[]         = "############################;
WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
#define STEPPER_IN1    13
#define STEPPER_IN2    12
#define STEPPER_IN3    14
#define STEPPER_IN4    27
#define TRIG_PIN       26
#define ECHO_PIN       25
#define RELAY_PIN      33
#define I2C_SDA        21
#define I2C_SCL        22
const float   PUMP_FLOW_ML_PER_MIN = 1200.0;
const float   FOOD_PER_REV_G       = 20.0;
const float   FOOD_BIN_HEIGHT_CM   = 12.5;
const unsigned long SAFETY_DELAY   = 10000;

const float   FOOD_EMPTY_RAW_PCT   = 7.0;
const float   FOOD_FULL_RANGE_PCT  = 100.0 - FOOD_EMPTY_RAW_PCT;

const uint8_t WATER_LOW_ADDR  = 0x77;  // low-section (8 pads)
const uint8_t WATER_HIGH_ADDR = 0x78;  // high-section (12 pads)
const uint8_t TOUCH_THRESHOLD = 100;   // default threshold for “wet”

float          waterLevelPct, foodLevelPct;
int            waterAmountInput, foodAmountInput;
bool           waterButton, foodButton;
int            scheduleWaterAmount, scheduleFoodAmount;
CloudSchedule  waterScheduleTrigger;

AccelStepper stepper(AccelStepper::HALF4WIRE,
                     STEPPER_IN1, STEPPER_IN3,
                     STEPPER_IN2, STEPPER_IN4);

unsigned long lastWaterMillis = 0,
              lastFoodMillis  = 0;
bool prevWaterActive = false;
void initProperties();
void onWaterButtonChange();
void onFoodButtonChange();
void deployWater(int ml);
void deployFood(int g);
void readI2CWaterSensor();
void setup() {
```

```
  Serial.begin(115200);
  delay(500);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);
  Wire.begin(I2C_SDA, I2C_SCL);
  stepper.setMaxSpeed(1000);
  stepper.setAcceleration(150);
  initProperties();

  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}

void loop() {
  ArduinoCloud.update();


  static unsigned long lastSensMillis = 0;
  if (millis() - lastSensMillis > 1000) {
    lastSensMillis = millis();
    readI2CWaterSensor();


    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    long dur = pulseIn(ECHO_PIN, HIGH);
    float dist = dur * 0.034 / 2.0;  // measured distance (cm)
    float levelRawCm = FOOD_BIN_HEIGHT_CM - dist;
    float rawPct = constrain((levelRawCm / FOOD_BIN_HEIGHT_CM) * 100.0, 0.0,
100.0);

    float adjusted   = rawPct - FOOD_EMPTY_RAW_PCT;
    float scaledPct  = adjusted * (100.0 / FOOD_FULL_RANGE_PCT);
    foodLevelPct     = constrain(scaledPct, 0.0, 100.0);
  }
  if (waterButton && millis() - lastWaterMillis > SAFETY_DELAY) {
    deployWater(waterAmountInput);
    lastWaterMillis = millis();
  }
  waterButton = false;
  if (foodButton && millis() - lastFoodMillis > SAFETY_DELAY) {
    deployFood(foodAmountInput);
    lastFoodMillis = millis();
  }
  foodButton = false;
  bool nowWaterActive = waterScheduleTrigger.isActive();
  if (nowWaterActive && !prevWaterActive) {

    Serial.println("Scheduler activated: deploying scheduled water AND
food");
    deployWater(scheduleWaterAmount);
    deployFood(scheduleFoodAmount);
  }
  prevWaterActive = nowWaterActive;
}
void initProperties() {
  ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
  ArduinoCloud.setSecretDeviceKey(DEVICE_KEY);
```

```
  ArduinoCloud.setThingId(THING_ID);
  ArduinoCloud.addProperty(waterLevelPct, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(foodLevelPct,  READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(waterAmountInput, READWRITE, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(foodAmountInput,  READWRITE, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(waterButton,      READWRITE, ON_CHANGE,
onWaterButtonChange);
  ArduinoCloud.addProperty(foodButton,       READWRITE, ON_CHANGE,
onFoodButtonChange);
  ArduinoCloud.addProperty(scheduleWaterAmount, READWRITE, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(scheduleFoodAmount,  READWRITE, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(waterScheduleTrigger, READWRITE, ON_CHANGE, NULL);
}

void onWaterButtonChange() {
  if (waterButton && millis() - lastWaterMillis > SAFETY_DELAY) {
    deployWater(waterAmountInput);
    lastWaterMillis = millis();
  }
  waterButton = false;
}

void onFoodButtonChange() {
  if (foodButton && millis() - lastFoodMillis > SAFETY_DELAY) {
    deployFood(foodAmountInput);
    lastFoodMillis = millis();
  }
  foodButton = false;
}

void readI2CWaterSensor() {
  uint8_t low_data[8], high_data[12];
  uint32_t mask = 0;

  Wire.beginTransmission(WATER_LOW_ADDR);
  Wire.endTransmission();
  Wire.requestFrom(WATER_LOW_ADDR, (uint8_t)8);
  for (int i = 0; i < 8 && Wire.available(); i++) {
    low_data[i] = Wire.read();
    if (low_data[i] > TOUCH_THRESHOLD) mask |= 1UL << i;
  }
  Wire.beginTransmission(WATER_HIGH_ADDR);
  Wire.endTransmission();
  Wire.requestFrom(WATER_HIGH_ADDR, (uint8_t)12);
  for (int i = 0; i < 12 && Wire.available(); i++) {
    high_data[i] = Wire.read();
    if (high_data[i] > TOUCH_THRESHOLD) mask |= 1UL << (8 + i);
  }
  uint32_t touch = mask;
  uint8_t  sections = 0;
  while (touch & 0x01) {
    sections++;
    touch >>= 1;
  }
  waterLevelPct = constrain(sections * 5.0, 0.0, 100.0);
}

void deployWater(int ml) {
  unsigned long onTime = (unsigned long)(ml / PUMP_FLOW_ML_PER_MIN *
60000.0);
  digitalWrite(RELAY_PIN, HIGH);
  delay(onTime);
  digitalWrite(RELAY_PIN, LOW);
}
```

```
// one rev = 4096 half-steps for a 28BYJ-48, etc.
void deployFood(int g) {
  long steps = (long)(g / FOOD_PER_REV_G * 4096.0);
  stepper.moveTo(stepper.currentPosition() - steps);
  // run() will accelerate to max speed then decelerate at end
  while (stepper.distanceToGo() != 0) {
    stepper.run();
  }
```

6. System Demonstration and Functional Validation

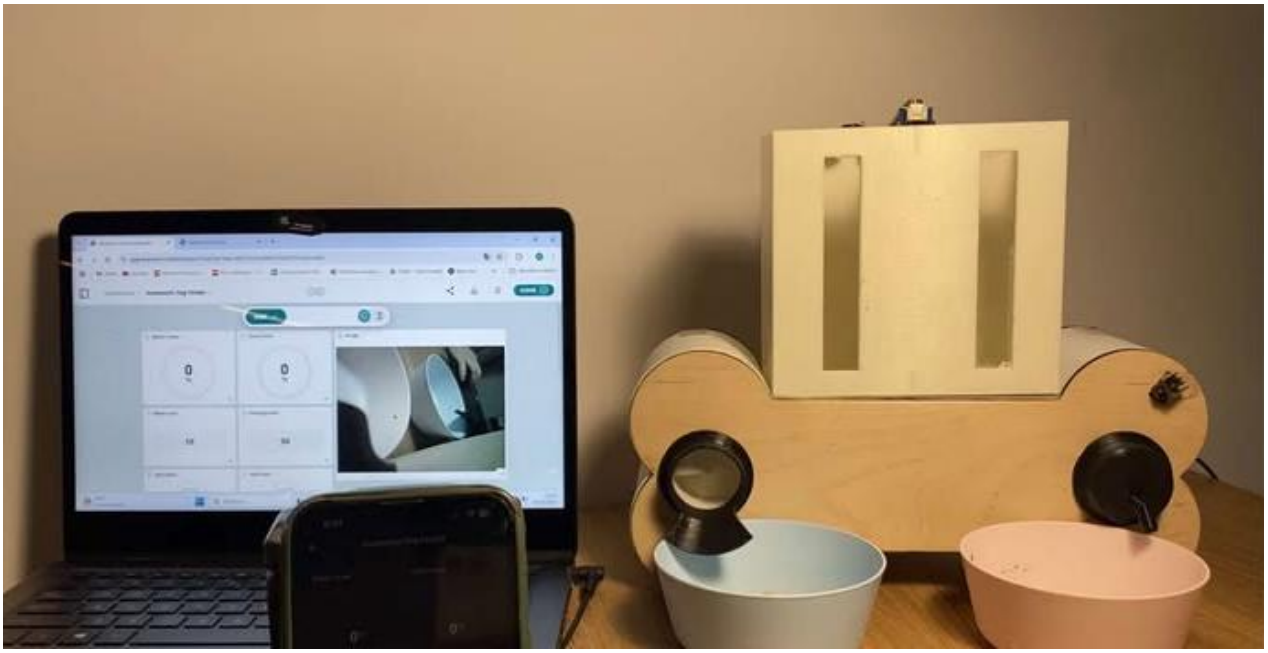


Figure 18

Figure 18 presents a real-life view of the working prototype along with the Arduino Cloud interface.

Figure 19

Figure 20

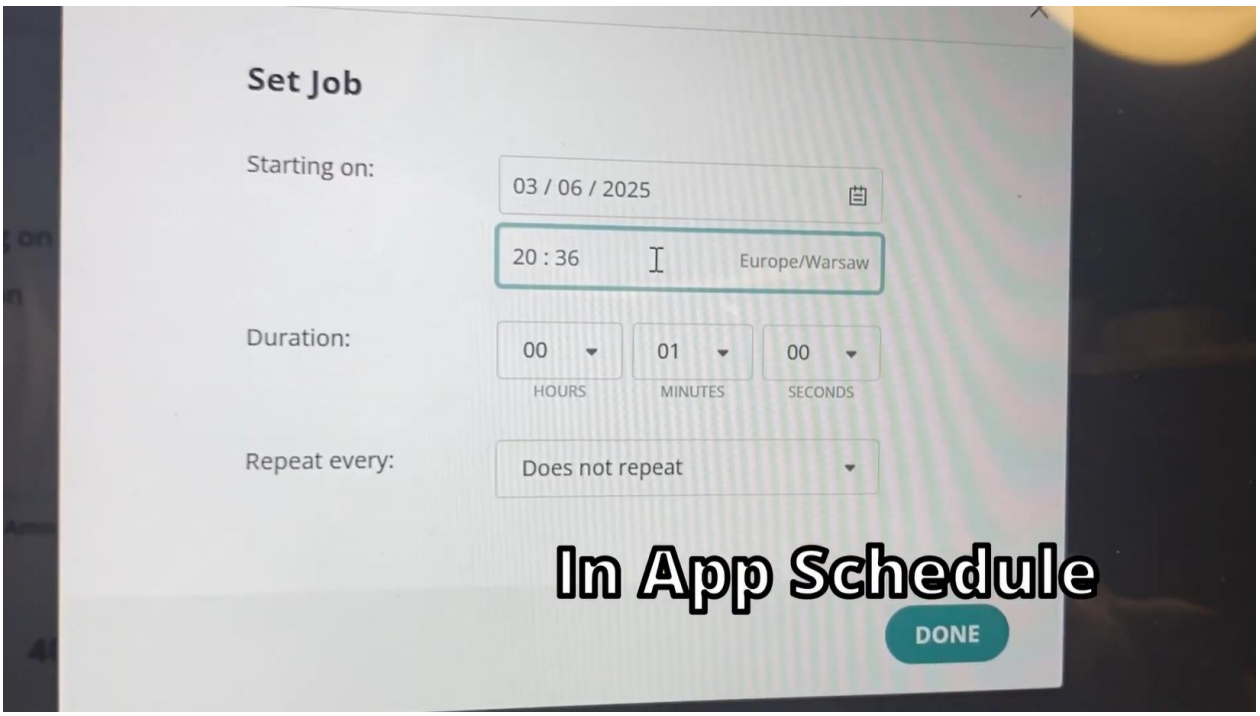


Figure 21

Figures 19, 20, 21 demonstrate the working prototype during various stages of operation. For clarity, these images showcase that all previously mentioned functions — including automated food and water dispensing, water level and food level monitoring and camera feedback — operate exactly as intended. This visual documentation confirms the successful implementation and integration of both the hardware and software components.

7. Future Improvements

- Physical Control Buttons:
  Adding physical buttons on the device would offer direct control without needing to access the online interface, improving accessibility and usability.
- Food Density Adjustment:
  Introducing a feature that allows the user to input or select the type and density of pet food would improve the accuracy of portioning by weight rather than volume, making the system more precise for various food types.
- Own Domain and Cloud Hosting:
  Hosting the control interface and live camera feed on a custom domain would provide easier and more reliable access. This would allow the user to manage the feeder remotely through a dedicated web application, improving scalability

8. Conclusion:

The presented project successfully demonstrates the design, construction, and testing of a automatic pet feeder. Through a combination of CAD modeling, 3D printing, laser cutting, and embedded electronics, the team developed a fully functional prototype capable of automated food and water dispensing, real-time monitoring, and remote control.

As shown in Figures 19–21, the system performs reliably under real conditions. The integration with Arduino Cloud enables intuitive control via a web-based dashboard and mobile devices, offering features such as food and water dispensing, water level and food level monitoring and camera feedback.

The physical prototype, combined with software automation, validates all design goals: sustainability, ease of use, and functionality

Overall, the project proves to be a scalable, practical, and eco-conscious solution for modern pet owners, combining smart-home features with mechatronic engineering principles.