Automated Homemade Dog Food Production Machine

Group 33

Zekai Song, Zixi Zhao, Wenkai Zheng, Jingyang Chen

Abstract

This project presents the design of an automated homemade dog food production machine for small-batch pet feed preparation. The system addresses three common problems in manual pet food processing: inconsistent mixing, unstable feeding of moist material, and the need to keep molded feed warm during prototype operation. The proposed machine integrates a blade-and-funnel mixing subsystem, a 42-stepper-motor-driven extrusion subsystem, a low-power resistance-wire heat-preservation subsystem, and an STM32F103C8T6-based electrical control subsystem. The target batch size is approximately 200 g of moist feed material. The design goal is to maintain stable material flow without clogging, shape the material through a mold, and provide low-power heat preservation while maintaining safe prototype operation.

The main engineering challenges are material flow stability, geometric matching between the extrusion screw and outlet tube, heat distribution, and safe integration of moving blades, the resistance-wire heat-preservation assembly, and electronic control hardware. The final design uses a blade mechanism to homogenize the input material, a 42 stepper motor to rotate the extrusion screw and push material through a narrow channel, and an aluminum basin/mold with a low-power resistance-wire warming module wound around the outside for heat holding. The heat-preservation assembly is limited to a maximum electrical input of 24 W at 12 V and 2 A, so it is specified as a warming/holding module for prototype operation. Verification is organized around flow testing, heat-preservation operation, control reliability, and safety checks. Because the proposal does not contain final experimental measurements, the measured results in the verification section are left as clearly marked fields to be completed after testing.

1. Introduction

Producing small batches of pet feed from moist raw material is difficult because the material must be mixed uniformly, transported without clogging, shaped into a usable form, and kept warm during the prototype process. Manual processing is inefficient and inconsistent, especially when the moisture content and texture of the raw material vary from batch to batch. Commercial dog food products can also be expensive, while homemade production allows the user to select ingredients and control the amount produced.

To address this problem, the team designed an automated homemade dog food production machine. The machine combines material grinding and mixing, controlled feeding, molding, low-temperature resistance-wire heat preservation, and basic electronic control. The system is intended to process approximately 200 g of raw material per batch. After the user loads the raw material into the funnel, the material is mixed by rotating blades, pushed forward by the 42-stepper-motor extrusion system, distributed into a mold, and warmed by resistance wire wrapped around an aluminum basin.

The design is guided by the following high-level requirements:

1. The system shall maintain stable material flow without clogging during feeding and molding.

2. The system shall process approximately 200 g of material per batch in a controlled and repeatable manner.

3. The system shall provide low-power heat preservation for the molded material while maintaining safe and practical prototype operation. The final thermal module is limited to 12 V, 2 A, and 24 W maximum input power.

The project is treated as an engineering prototype rather than a certified food-grade product. Therefore, the focus of the final draft is mechanical feasibility, control feasibility, safety protection, and a clear verification method. Claims about final food safety and heat-preservation performance must be supported by measured test data before submission or public demonstration.

2. Project Requirements and Constraints

The design requirements are organized around function, performance, safety, and practicality.

2.1 Functional Requirements

The machine must accept moist pet feed material through a funnel, mix and grind the material with a blade assembly, feed the material through a controlled rotary extrusion mechanism, shape the material through a mold, and keep the molded material warm using a low-power resistance-wire module. The subsystems must operate in a coordinated sequence so that material is not trapped between the mixing, feeding, and molding stages.

2.2 Performance Requirements

The target batch mass is approximately 200 g. The feeding system must transport this mass through the designed geometry without blockage. The heat-preservation system uses a low-power resistance-wire heat-preservation module rated at 12 V, 2 A maximum current, and 24 W maximum electrical power. The resistance wire is wound around the aluminum basin so that heat is transferred through the basin wall into the molded material for warming and heat holding.

2.3 Safety Requirements

The prototype includes rotating blades, a motor-driven feeding mechanism, a low-power resistance-wire heat-preservation module, and electrical components. The blade area must be physically guarded to prevent direct hand contact during operation. The feeding path must reduce pinch-point exposure and avoid excessive pressure buildup. The heated aluminum basin and resistance wire must be insulated or separated from user contact surfaces. Electrical wiring and control hardware must be protected from heat, moisture, and moving parts.

2.4 Practical Constraints

The design uses commercially available components where possible, including an STM32F103C8T6 microcontroller, a 42 stepper motor and driver, a 12 V low-power resistance-wire warming module, a power supply, 3D printed parts, and an aluminum basin/mold. The estimated prototype cost is 640 RMB after removing the unused temperature-sensing module. The design also assumes that the user can disassemble or access relevant sections for cleaning after the machine has been powered off and cooled.

3. System Overview

The machine consists of three main mechanical subsystems and one electrical control subsystem. The overall process begins when the user loads raw material into the funnel and powers on the machine. The blade-and-

funnel subsystem cuts and homogenizes the material. The feeding subsystem then uses a 42 stepper motor and extrusion screw to push the material forward. The resistance-wire heat-preservation and molding subsystem receives the material, shapes it in the aluminum basin/mold, and keeps the molded material warm through controlled low-power heating. The electrical control subsystem coordinates the motor, blade, resistance wire, and servo through the STM32F103C8T6 controller.

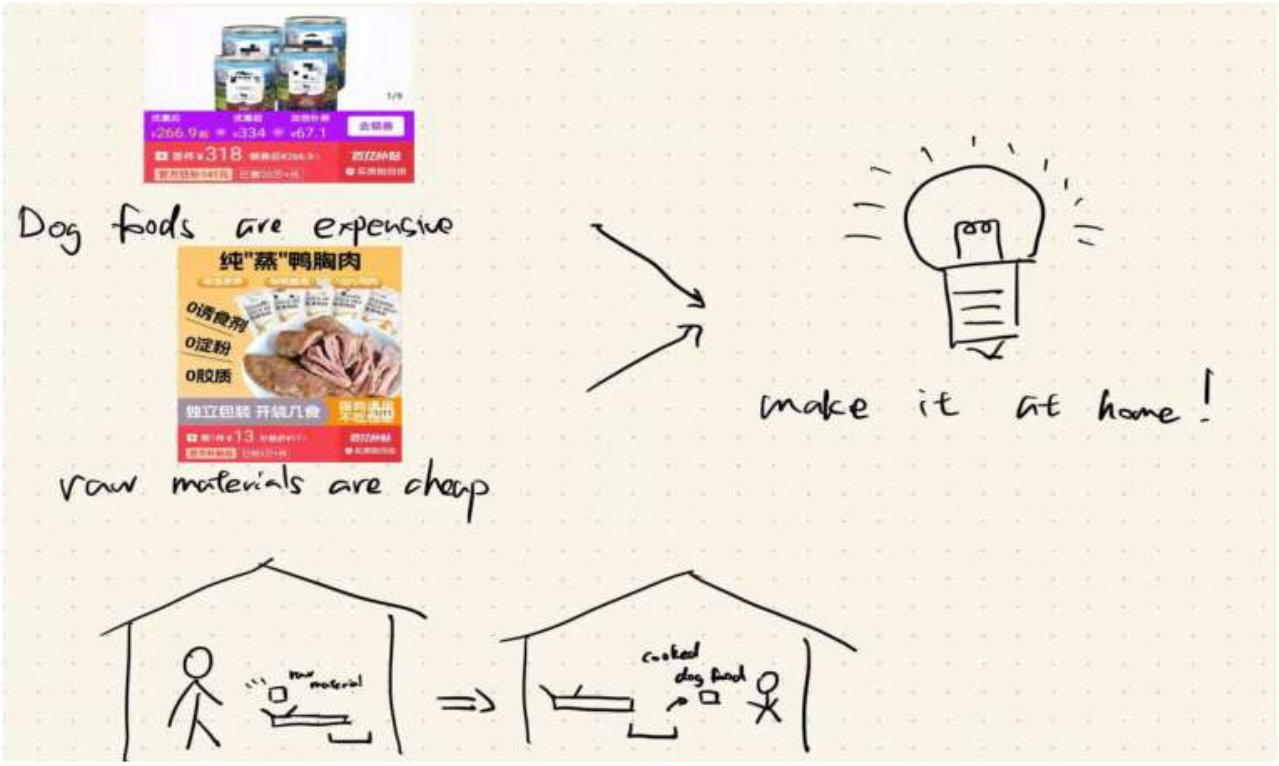

Figure 1. Pictorial representation of how the final solution is used in the context of pet feed production.

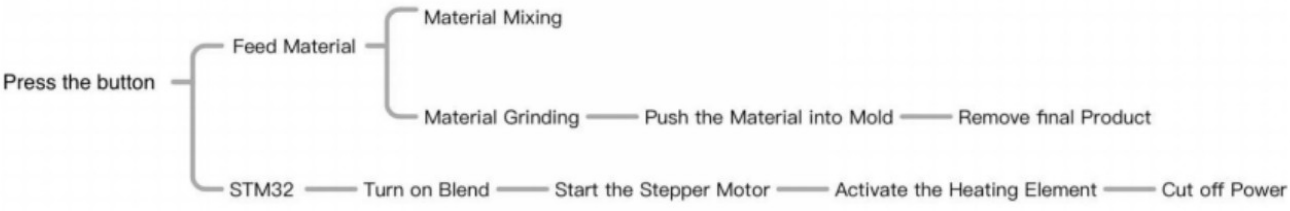

Figure 2. Block diagram of the operating sequence.

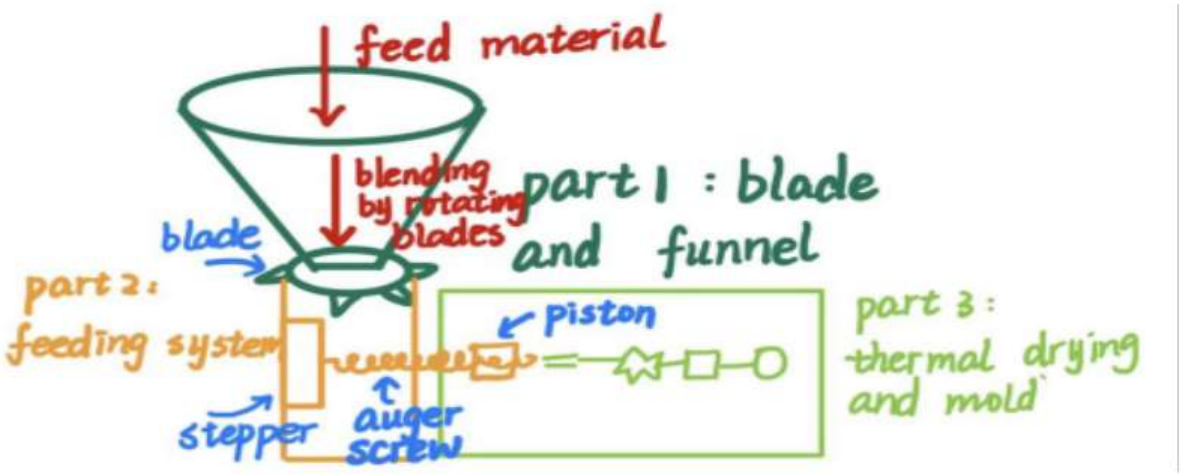

Figure 3. Subsystems overview showing the blade-and-funnel system, feeding system, and thermal heat-preservation and mold system.

The process sequence is:

1. The user loads feed material into the funnel.

2. The blade assembly mixes and grinds the material.

3. The processed material enters the feeding chamber.

4. The 42 stepper motor rotates the extrusion screw and drives the material forward.

5. The extrusion screw compresses the material into the outlet tube and mold.

6. The resistance wire wound around the aluminum basin warms the molded material and provides low-power heat preservation.

7. The system shuts off the resistance wire and the user removes the final product after cooling.

4. Mechanical Design

4.1 Blade and Funnel Subsystem

The blade-and-funnel subsystem is responsible for grinding and homogenizing the input material. A funnel directs the moist feed material toward the blade area, where rotating blades break down larger pieces and improve mixture uniformity. This stage is important because inconsistent particle size or uneven moisture distribution can increase the chance of clogging during feeding.

The primary requirement for this subsystem is that it grind and blend the feed material effectively without blockage. The verification method is visual inspection and flow testing using feed materials with different

moisture contents. The system should be checked for material accumulation near the blade, jamming at the funnel outlet, and uneven discharge into the feeding chamber.

Because this subsystem uses commercially available blade and motor components, the main design effort is not component feasibility but safe integration. The blade region must be covered or shielded so that the user cannot reach the moving blade during operation. The cover should still allow material to enter the intended path and should be removable only when the machine is powered off.

4.2 Feeding Subsystem

The feeding subsystem uses a 42 stepper motor, extrusion screw, and compression structure to push material forward into the mold. The stepper motor provides controllable rotary motion, while the screw converts motor rotation into extrusion pressure on the moist material. This controlled motion is necessary because moist feed material can behave as a paste: it may flow under compression, but it can also clog if the pressure rises too quickly or if the outlet geometry is too restrictive.

The target batch mass is 200 g. Assuming an approximate material density of 2 g/cm^3, the required volume is:

```
V = 200 g / 2 g/cm^3 = 100 cm^3
```

Using a 4 cm equivalent compression diameter for the extrusion chamber, the cross-sectional area is:

```
A = pi * (2 cm)^2 = 12.57 cm^2
```

The required equivalent displacement length is therefore:

```
L = V / A = 100 cm^3 / 12.57 cm^2 = 7.96 cm
```

An effective extrusion displacement of approximately 8 cm is sufficient to displace the target batch volume under the stated density assumption. However, the material is then pushed into a narrow tube with a diameter of approximately 1 cm. This reduction in flow area increases resistance and makes motor speed control critical. If the screw drives material too quickly, the material may compact at the outlet and clog. If the screw advances material too slowly, processing time increases and the material may remain in the mold longer than needed before removal.

The selected 42 stepper motor is appropriate for controlled extrusion because it can be driven at repeatable step rates through the STM32F103C8T6 controller and a motor driver. The final motor command profile should begin with a low feed rate, increase only after stable flow is observed, and stop immediately if excessive resistance, abnormal vibration, or material blockage is detected.

定格特性
RATED CHARACTERISTICS

相数 PHASES	2	
基本ステップ角度 FUNDAMENTAL STEP ANGLE	1.8	°
定格電圧 RATED VOLTAGE	2.88	V[DC]
定格電流 RATED CURRENT	1.2	A/PHASE
巻線抵抗 WINDING RESISTANCE	2.4	Ω±10% at 25 ℃
巻線インダクタンス WINDING INDUCTANCE	2.3	mH±20% at 1 kHz 1 V[rms]
ホールディングトルク HOLDING TORQUE	0.2	N・m MIN. at I=1.2 A/PHASE 2相励磁 2 PHASE EXCITATION.
脱出トルク PULL-OUT TORQUE	0.15	N・m MIN. at 1000 pulse/s
		負荷イナーシャ INERTIAL LOAD 0.94×10^{-4} kg・m^2 (ラバーカップリングイナーシャを含む) (INERTIA OF RUBBER COUPLING IS INCLUDED)
最大自起動周波数 MAXIMUM STARTING PULSE RATE	1600	pulse/s MIN. 無負荷時 at NO LOAD
最大連続応答周波数 MAXIMUM SLEWING PULSE RATE	3400	pulse/s MIN. 無負荷時 at NO LOAD
静止角度誤差 POSITIONAL ACCURACY	±0.09	°(0.18 °SPREAD MAX.) 2相励磁 2 PHASE EXCITATION.
温度上昇値 TEMPERATURE RISE	80	K MAX.
ロータイナーシャ ROTOR INERTIA	0.036×10^{-4} kg・m^2	NOMINAL
耐熱クラス THERMAL CLASS	B	
許容スラスト荷重 ALLOWABLE THRUST LOAD	10	N
許容ラジアル荷重 ALLOWABLE RADIAL LOAD	26	N 軸先端荷重 LOAD TO SHAFT END.

Figure 4. Stepper motor specifications.

4.3 Resistance-Wire Heat-Preservation and Mold Subsystem

The resistance-wire heat-preservation and mold subsystem shapes the material and provides low-temperature warming assistance. The updated design uses an aluminum basin/mold with resistance wire wound around the outside surface. The resistance wire operates from a 12 V supply, draws up to 2 A, and has a maximum electrical input power of 24 W. This value is calculated from the electrical power relation:

The working principle is similar to low-pressure extrusion and molding: the feeding subsystem pushes the moist material into the mold area, and the wound resistance wire warms the aluminum basin so that heat is transferred into the molded material. The final implemented module is a low-power heat-preservation module. Its function is to maintain warmth during the prototype sequence.

```
P = V * I = 12 V * 2 A = 24 W
```

For a 100-second PA6 heat-preservation interval, the maximum electrical energy delivered to the resistance-wire module is:

```
E = P * t = 24 W * 100 s = 2,400 J = 2.4 kJ
```

Uniform thermal contact is the main design objective of this subsystem. Non-uniform heating may create local hot spots on the basin surface. The aluminum basin helps distribute heat from the wound resistance wire. In the final prototype, the earlier PA5 temperature-sensing interface is not populated, because the heat-preservation module is low power and is controlled by a timed PA6 output rather than closed-loop temperature feedback. Therefore, thermal verification should use external measurement tools during testing, such as an infrared thermometer, contact thermometer, or surface temperature probe, while the controller limits the heat-preservation path mainly through fixed wiring, voltage, current, and operating time.

5. Electrical and Control Design

The electrical control subsystem coordinates blade operation, 42-stepper-motor extrusion, servo actuation, resistance-wire heat preservation, and shutdown by power removal. The final prototype uses an STM32F103C8T6-based control system because it can provide digital outputs for motor control, timed PA6 heat-preservation control, and PWM output for the servo. The custom PCB consolidates the power conversion circuit, MCU circuit, motor switching circuits, A4988 stepper driver interface, PA6 heat-preservation MOSFET path, reserved early-revision pads, and external connectors into one board.

Figure 5 shows the final circuit schematic used for the PCB implementation. The schematic combines the USB/power input, DC-DC conversion section, STM32F103C8T6 controller, MOSFET switching circuits, A4988 stepper driver interface, PA6 warming/heat-preservation output, and external connectors. In the final version, PA0 is not used as a total/start switch and PA5 is not used for temperature detection; the firmware starts automatically after power-on and does not read either pin.

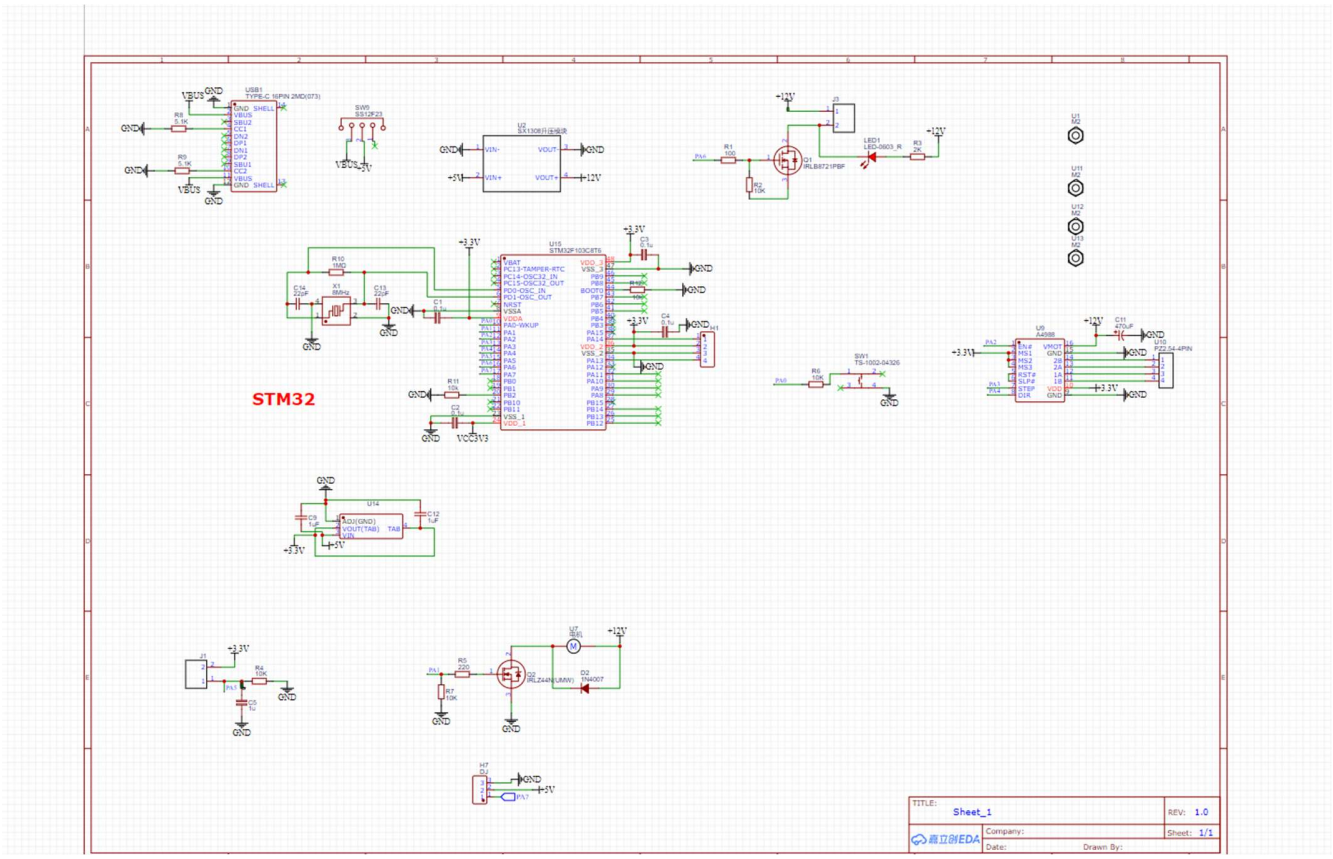

Figure 5. Final circuit schematic used for the PCB implementation.

The control system includes:

Component	Function
STM32F103C8T6 microcontroller	Main controller for process sequence, timing, motor control, heat-preservation control, and servo PWM output
Power conversion circuit	Provides the logic rail and actuator rail used by the controller, driver, motors, and resistance-wire load
42 stepper motor and driver	Rotates the extrusion screw at controlled step rates
MG90S micro servo	Provides auxiliary actuation in the prototype assembly
Grinding motor or blade motor driver	Controls blade rotation
12 V resistance-wire warming	Provides low-power heat holding through wire wound around the

module	aluminum basin
12 V power supply	Provides the resistance wire with up to 2 A; additional current margin is needed if the motors share the same supply
Connectors and wiring	Connect the controller, motors, servo, and resistance-wire heat-preservation module

5.1 Circuit Block Design

The power supply block provides the voltage rails required by the prototype. The PCB includes a USB/5 V input path and a DC-DC conversion section for the 12 V actuator/heating rail. A 3.3 V regulator supplies the STM32F103C8T6 and low-voltage logic. Bulk capacitors are placed near the power input and motor/heater rails to reduce low-frequency voltage dips, while smaller ceramic capacitors are placed near logic IC power pins to suppress high-frequency noise.

The main controller block is built around the STM32F103C8T6. The final control program starts the prototype sequence automatically after power-on, then generates the control outputs for the MOSFET switches, A4988 driver, and servo. Decoupling capacitors are placed close to the MCU power pins to improve controller stability during motor switching and heater operation.

The dispenser motor control block uses a logic-level MOSFET as a low-side switch for the DC motor path. The MCU drive signal passes through a small series gate resistor to limit switching transients, and a gate pull-down resistor keeps the MOSFET off when the MCU pin is floating during startup. A flyback diode is connected across the motor load to provide a discharge path for inductive voltage spikes when the motor is switched off.

The stepper motor block uses an A4988 driver module. The MCU controls ENABLE, STEP, and DIR signals so the 42 stepper motor can rotate the extrusion screw at repeatable speeds and directions. The driver uses a 3.3 V logic supply and a separate motor supply rail. A bulk capacitor near VMOT helps absorb current pulses from the stepper motor and reduces the risk of driver reset or overvoltage damage.

The early schematic included a PA5 temperature-sensing interface, but the PA5 temperature-detection module is removed from the final prototype. Since the heat-preservation module is limited to low power, the final design treats temperature measurement as an external verification step rather than an embedded feedback loop. This avoids claiming closed-loop temperature control that is not implemented in the current PCB/software combination.

The heat-preservation control block uses a logic-level MOSFET to switch the resistance-wire heat-preservation load from the MCU. The PA6 control signal drives the MOSFET gate through a resistor, while a pull-down resistor prevents accidental turn-on during reset. The LED indicator provides visible feedback when the heat-preservation path is active. The heater connector routes the 12 V load current to the resistance wire wound around the aluminum basin.

The active MCU pin assignments are summarized below. PA0 and PA5 are listed only to clarify that they are not

part of the final control logic.

MCU signal	Connected circuit	Function
PA0	Not used in final version	No total/start switch is implemented in the final firmware; the system starts automatically after power-on
PA1	DC motor MOSFET gate	Controls the dispenser or blade motor switching path
PA2	A4988 ENABLE	Enables or disables the stepper motor driver
PA3	A4988 STEP	Sends step pulses to the 42 stepper motor driver
PA4	A4988 DIR	Selects stepper motor rotation direction
PA5	Not used in final version	Temperature-detection module removed; no embedded temperature input is read by the firmware
PA6	Heat-preservation MOSFET gate	Switches the 12 V low-power warming/heat-preservation load
PA7	MG90S servo PWM	Generates the servo position control signal

The intended control sequence is:

1. Power on the control board and allow the STM32 to initialize.

2. Keep the DC motor control active by default for the demonstration sequence.

3. After the startup delay, start the 42 stepper motor at the selected extrusion pulse rate.

4. Drive the MG90S servo through the programmed PWM timing sequence.

5. Keep the PA6 heat-preservation output active for the programmed low-power warming interval, then turn it off in software or by external power removal.

6. Remove power and allow cooling before the user opens the mold and removes the product.

For final implementation, the resistance-wire heat-preservation output should still use a MOSFET switching path, even though the final control program uses only a timed PA6 output rather than closed-loop temperature feedback. At 12 V and 2 A maximum current, the resistance wire requires wiring, connectors, and switching components rated above 2 A with appropriate margin. Because the embedded temperature-sensing path is not used, prototype safety depends on low heating power, fixed operating conditions, external temperature checks

during validation, insulation from user contact, and manual power removal when the test is complete.

6. Engineering Analysis

6.1 Material Flow Stability

Material flow is the highest-risk mechanical issue because moist feed material can clog in several locations: under the funnel, around the blade outlet, in the extrusion chamber, at the 1 cm outlet tube, or inside the mold channels. The design addresses this by first homogenizing the material and then controlling the extrusion screw speed. The most important variables are moisture content, particle size, extrusion chamber diameter, outlet diameter, feed rate, and mold channel geometry.

Stable feeding is defined by continuous material movement without visible blockage, sudden motor stall, excessive pressure buildup, or inconsistent discharge into the mold. If clogging occurs, the design can be adjusted by reducing the compression area, increasing the effective extrusion length, enlarging the outlet channel, smoothing the internal geometry, or reducing the material moisture content before feeding.

6.2 Geometric Matching

The feeding calculation shows that a 4 cm equivalent compression diameter needs approximately 8 cm of effective displacement to move 100 cm^3 of material. The difficulty is not only displacement volume but also transition geometry. A 4 cm chamber feeding a 1 cm outlet creates a large reduction in flow area. This geometry can work only if the material remains soft enough to deform, the feed rate is slow enough to avoid compacting the material, and the outlet path is smooth enough to avoid local buildup.

The final design should treat the outlet and mold channels as tolerance-sensitive features. Sharp corners, rough 3D printed surfaces, and sudden diameter reductions should be avoided. The transition from the extrusion chamber to the outlet tube should be filleted or tapered when possible.

6.3 Heat-Preservation Requirement

The final thermal requirement is heat preservation. The resistance-wire module is rated for a maximum input of 12 V and 2 A, so the maximum electrical power is 24 W. The STM32 does not perform closed-loop temperature regulation because the PA5 temperature-detection module is removed. Instead, PA6 switches the heat-preservation load for a programmed time interval. This means the thermal subsystem should be evaluated by electrical input, external surface-temperature measurement, and safe operation.

6.4 Heat Distribution and Safety

Thermal uniformity is necessary for user safety and consistent warming. Uneven heating may create local hot spots on the aluminum basin or resistance wire. The most practical verification method is to measure resistance-wire voltage and current, calculate electrical power, and record basin surface temperature at several points during the programmed PA6-on interval. The acceptance criterion should be that the module remains within the 12 V, 2 A, 24 W limit and that the surface temperature remains acceptable for prototype handling after cooling.

7. Prototype Implementation

The prototype implementation is organized as an integrated mechanical-electrical system. The mechanical assembly includes the funnel, blade region, extrusion chamber, screw drive, outlet tube, and aluminum basin/mold. The electrical assembly includes the STM32F103C8T6 controller, 42 stepper motor driver, blade motor control, MG90S servo connection, 12 V resistance-wire heat-preservation control, power supply, and external connectors.

The blade subsystem is mounted below the funnel so that raw material falls into the mixing region by gravity. After mixing, the material enters the feeding chamber. The 42 stepper motor and screw drive are aligned with the extrusion path to reduce side loading and friction. The rotating screw compresses material toward the outlet tube and mold. The heated aluminum basin/mold is designed so that the finished dog food pieces can be removed after cooling.

The controller software is organized around an automatic power-on demonstration sequence. After initialization, the DC motor and PA6 warming output are enabled, the stepper motor waits for a short startup delay before cycling through forward, stop, and reverse phases, and the MG90S servo follows its programmed PWM timing sequence. In the latest code, PA6 is controlled by a 100-second timed heat-preservation interval. This software structure matches the simplified final PCB usage: PA0 is not a total/start switch, and PA5 is not a temperature-detection input.

The implementation should still support manual interruption at the system level through power removal. This is especially important because the prototype contains both mechanical and thermal hazards. During testing, the board should be powered from a supply or switch that can be disconnected quickly.

7.1 Control PCB Implementation

The custom PCB was produced to make the final prototype wiring more stable and easier to assemble. The board places the USB/power input, voltage conversion module, STM32 controller, A4988 stepper driver socket, MOSFET switching sections, and external connectors on a single panel. If PA0 and PA5 footprints/signals appear on the early schematic or PCB, they should be treated as unused reserved points only; they are not connected to the final automatic-start logic and are not used by the firmware. This layout reduces long jumper wires, gives the motor and heat-preservation connections clearer terminals, and makes debugging easier because the key control points are fixed on the PCB.

Figure 6 shows the PCB routing layout. The cleaned layout image focuses on the board outline, copper traces, pads, and component footprints instead of the design-software interface. The routing places the STM32 controller near the middle of the board, keeps the stepper-driver socket and output connector on the right side, and routes the MOSFET-controlled loads through wider traces to reduce voltage drop and wiring confusion during assembly.

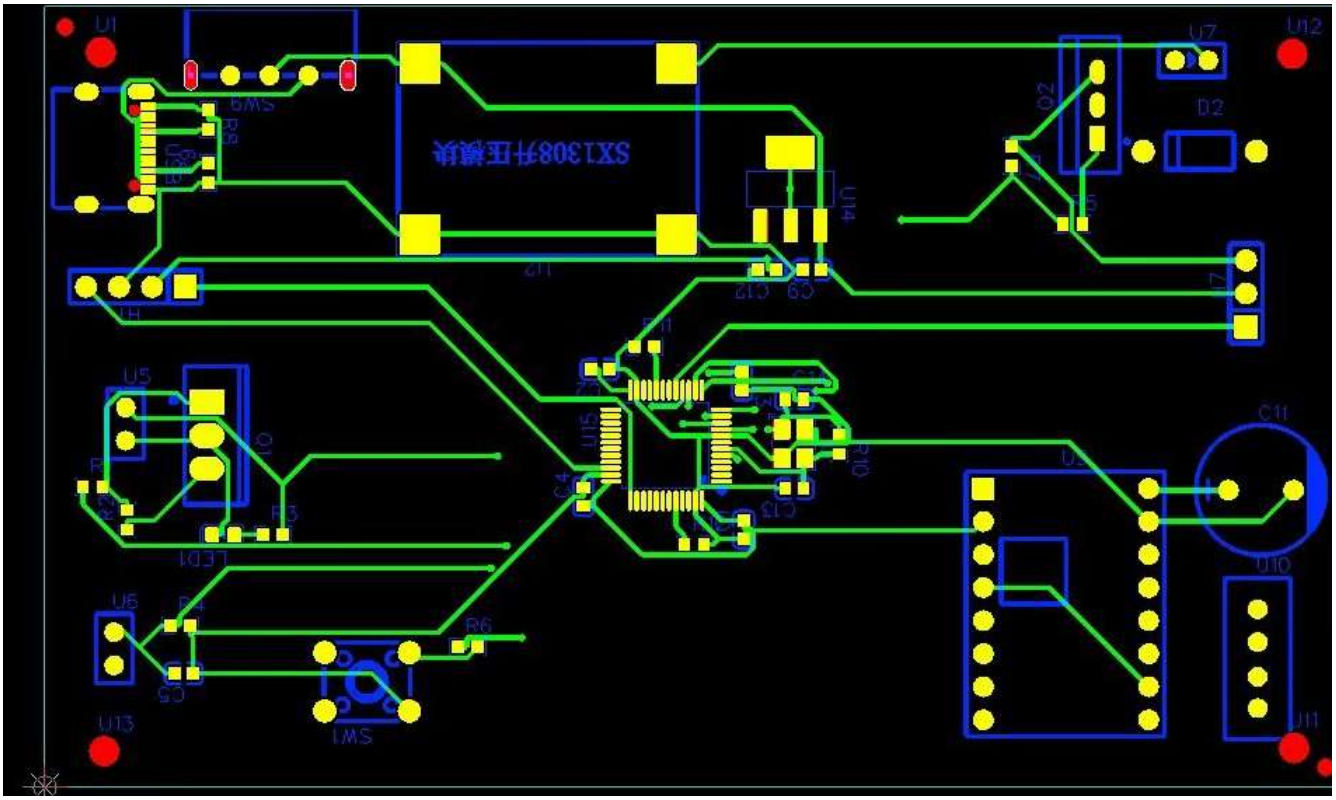

Figure 6. PCB routing layout for the electrical control board.

Figure 7 shows the assembled physical PCB used in the prototype. The upper-left area contains the USB/power input and programming-related connectors, the blue module near the top center is the DC-DC conversion module, the STM32 controller is placed near the center of the board, and the red A4988 module at the lower right drives the 42 stepper motor. The right-side connectors route power and actuator wiring to the stepper motor, warming/heat-preservation module, servo, and other prototype loads. This physical board replaces loose jumper wiring and makes the final demonstration easier to assemble and troubleshoot.

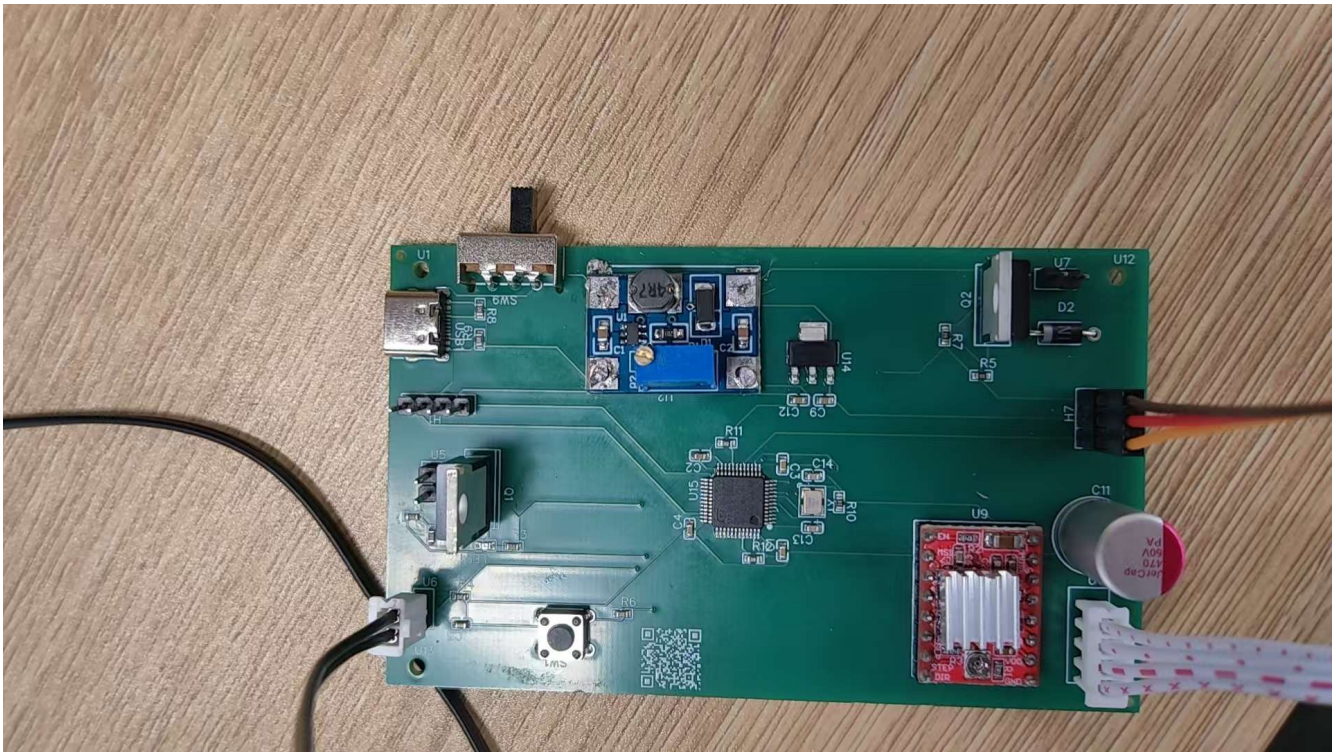

Figure 7. Assembled physical control PCB used in the prototype.

7.2 Final Model and Prototype Photos

Figure 8 shows the final CAD model of the overall machine. The model organizes the upper mixing unit, middle extrusion structure, lower receiving/heating area, and supporting frame into a single vertical layout.

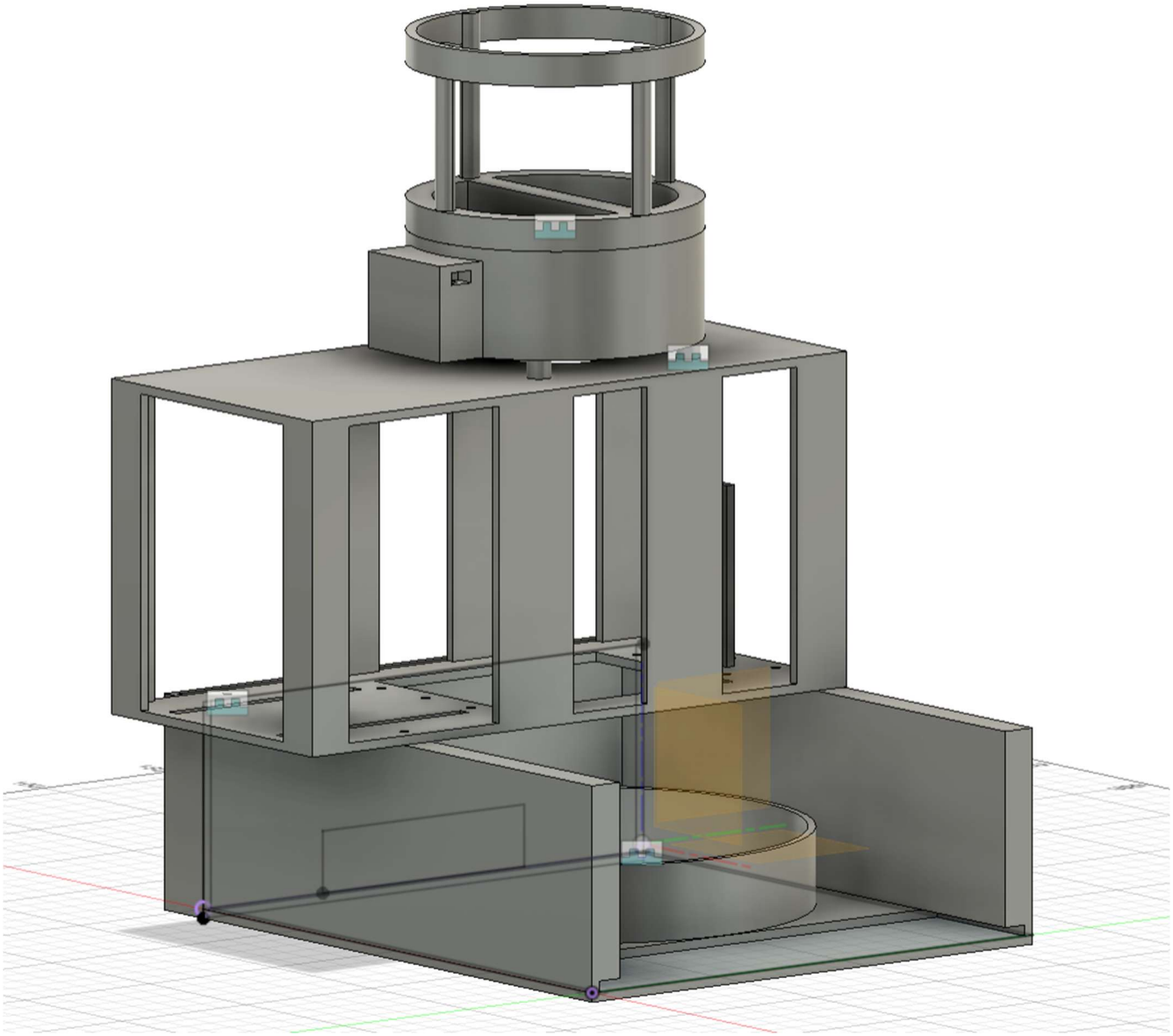

Figure 8. Final CAD model of the automated dog food production machine.

Figure 9 shows the physical stirring subsystem. The orange 3D-printed frame holds the transparent mixing container and supports the blade/mixing region.

Figure 9. Physical stirring subsystem.

Figure 10 shows the resistance-wire heat-preservation module. The resistance wire is wound around the outside of the aluminum basin so that the basin wall spreads heat into the food material.

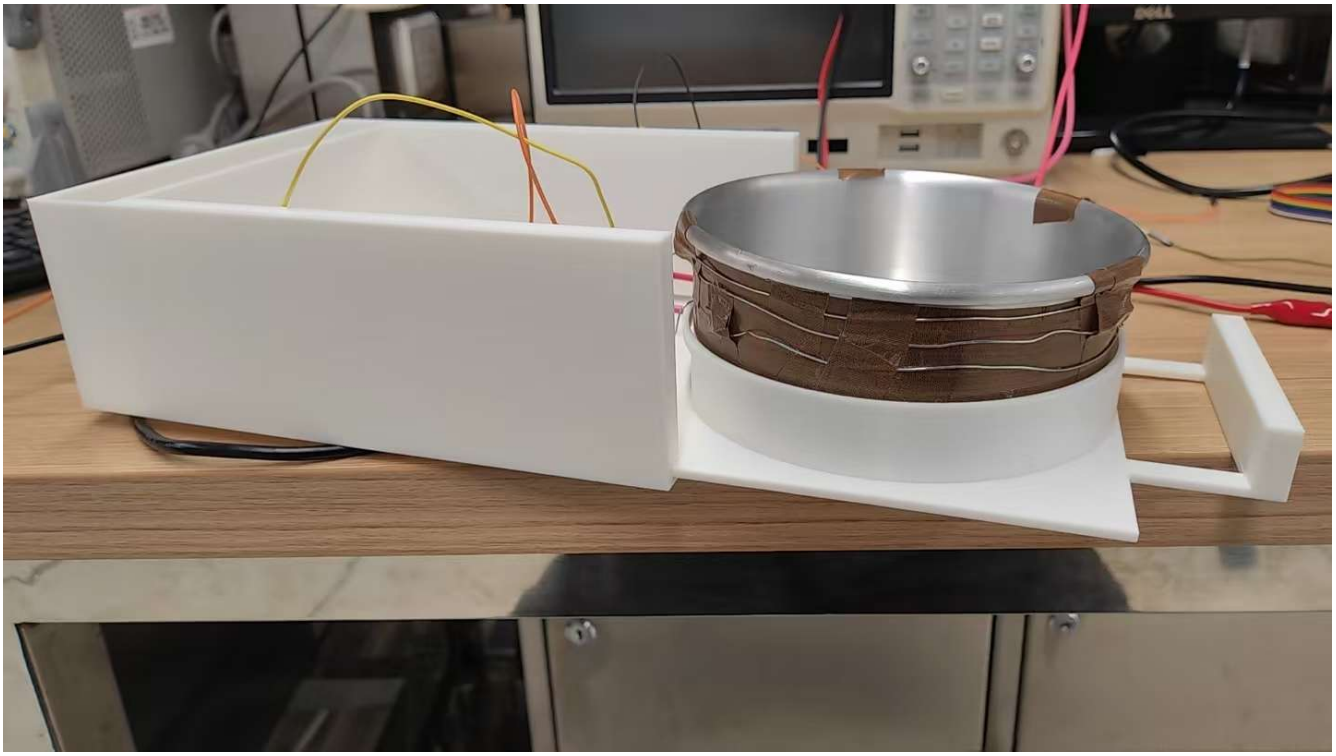

Figure 10. Resistance-wire heat-preservation module wound around the aluminum basin.

Figure 11 shows the motor and servo assembly used during prototype integration. The MG90S micro servo is used as an auxiliary actuator, while the stepper motor system provides the main extrusion motion.

Figure 11. Motor and servo components used in the prototype.

Figure 12 shows the integrated rotary extrusion module with the 42 stepper motor, screw drive, control PCB, and servo assembly. This module demonstrates the final feeding mechanism used to rotate and press the material forward.

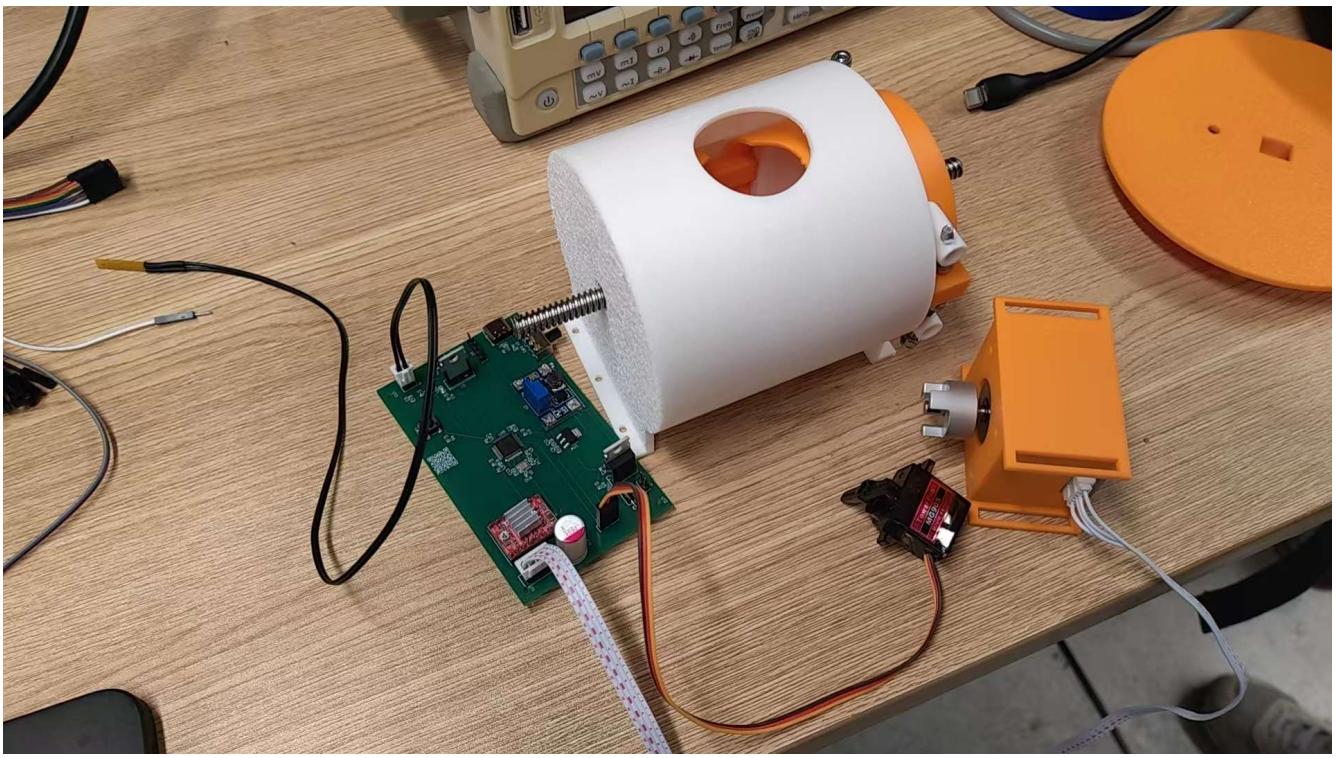

Figure 12. Rotary extrusion system driven by the 42 stepper motor with servo assembly.

8. Verification and Validation

The verification plan is based on the project requirements. The proposal does not include final measured data, so the result column below must be completed after prototype testing.

Requirement	**Verification method**	**Acceptance criterion**	**Measured result**	**Status**
Grind and homogenize feed material	Test materials with different moisture contents and visually inspect mixture quality	Material exits blade region without jamming or large unmixed pieces	[Insert measured observation]	[Pass/Revise]
Feed approximately 200 g per batch	Load a 200 g batch and run the 42-stepper-motor extrusion system	Batch is transported into the mold without major residual material in the chamber	[Insert measured mass and observation]	[Pass/Revise]
Prevent clogging	Test 20%, 30%, and 40% moisture	No clogging, motor stall, or unsafe	[Insert test result]	[Pass/Revise]

during feeding	materials at selected motor speeds	pressure buildup		
Provide low-power heat preservation	Run the 12 V resistance-wire heat-preservation module during the programmed PA6 interval and measure surface temperature externally	Molded material remains warm during the prototype sequence	[Insert warmth and temperature observation]	[Pass/Revise]
Stay within thermal-module power limit	Measure resistance-wire voltage and current during operation	Electrical input does not exceed 12 V, 2 A, or 24 W	[Insert voltage, current, and power]	[Pass/Revise]
Maintain controlled thermal operation	Monitor heating-wire voltage, current, basin temperature, and chamber temperature with external tools	Heating wire remains at or below 12 V, 2 A, and 24 W; surface temperature remains acceptable for prototype testing	[Insert electrical and temperature range]	[Pass/Revise]
Operate electrical control sequence	Run full power-on motor-stepper-servo-heating sequence	Motors, servo, and resistance wire activate only in intended states	[Insert control test result]	[Pass/Revise]
Provide safe prototype operation	Inspect blade cover, hot-surface isolation, wiring, and manual power-disconnect behavior	No direct access to moving blades; heating and wiring are protected; power can be removed quickly	[Insert safety check result]	[Pass/Revise]

8.1 Feeding Test Matrix

The feeding subsystem should be tested before heating tests. Recommended test conditions are:

Test	Moisture content	Batch mass	Motor setting	Expected observation
F1	20%	200 g	Low speed	Stable flow, low clogging risk
F2	30%	200 g	Low to medium speed	Stable flow if material is homogeneous
F3	40%	200 g	Low speed	Higher clogging risk; observe outlet carefully

The best operating point is the material condition that feeds reliably without clogging, because the final thermal module is intended only for warming and heat preservation.

8.2 Heat-Preservation Test Matrix

The thermal subsystem should be tested using controlled batch mass, resistance-wire power, and holding time. Since the updated heat-preservation system is limited to 12 V, 2 A, and 24 W and does not include the PA5 temperature-detection module, the test matrix should record both electrical input and externally measured temperature:

Test	Initial moisture	Heating input	Time	Data to record
D1	25%	Actual 12 V resistance-wire input, up to 24 W	100 s	Initial mass, final mass, heating voltage/current, basin temperature, texture
D2	30%	Actual 12 V resistance-wire input, up to 24 W	100 s	Initial mass, final mass, heating voltage/current, basin temperature, texture
D3	35%	Actual 12 V	100 s	Temperature

		resistance-wire input, up to 24 W		uniformity, hot-spot check, post-cooling handling observation
D4	Best-feeding material condition	Actual 12 V resistance-wire input, up to 24 W	Selected safe interval	Confirm that wiring, MOSFET, connector, and basin temperature remain within safe prototype limits
Optional	Extended demonstration check	Actual 12 V resistance-wire input, up to 24 W	100 s	Confirm safe warming behavior within the 12 V, 2 A, 24 W limit

The final report should identify the voltage-current-time condition that gives stable warming and safe operation for the final 12 V, 2 A heat-preservation design.

9. Cost and Project Management

The estimated prototype cost is summarized below.

Part	**Cost (RMB)**
USB-to-TTL serial module for PC-to-STM32 connection	15
PCB components, including STM32, motor driver, power module, connectors, and related parts	85
Feeding motor set, including 42 stepper motor, driver, and rotary extrusion mechanism	95
Grinding motor and blade assembly	60

Aluminum basin/mold and related support parts	120
12 V low-power resistance-wire heat-preservation module, up to 24 W	25
12 V switching power supply or desktop adapter, at least 2 A for resistance wire with additional margin if shared with motors	65
Funnel, outlet tube, fasteners, and miscellaneous mechanical parts	35
3D printed parts	60
Reserved contingency cost	80
Total	640

The project schedule ran from initial concept discussion through final presentation preparation. The major work packages were divided among the four team members:

Team member	**Main responsibility**
Jingyang Chen	Overall project framework, electrical architecture, key control modules, PCB and control verification
Wenkai Zheng	Electrical direction, power supply research, circuit connections, control and power debugging
Zixi Zhao	Mechanical concept, mechanical feasibility, CAD modeling, mechanical prototype construction and stability verification
Zekai Song	Structural concept, CAD refinement, fabrication preparation, prototype assembly support and testing assistance

The project management plan supported parallel mechanical and electrical development. Mechanical design

focused on CAD modeling, feeding structure, and mold integration. Electrical design focused on the STM32 control system, power connections, motor driver, reserved interfaces, and PCB preparation. System-level integration combined these tasks into a prototype suitable for final testing and presentation.

10. Ethics, Safety, and Environmental Considerations

10.1 Ethics

The machine processes ground raw material into shaped pet food products. Therefore, the design must prioritize honest performance claims, responsible testing, and contamination awareness. The prototype should not be presented as a certified food-grade manufacturing device unless the materials, cleaning procedure, and process controls are upgraded and validated against applicable food and animal feed requirements.

A primary ethical concern is food-related contamination. All material-contact parts should reduce residue buildup, loose fasteners, degraded prototype material, and unclean surfaces. Since the current design is a prototype, claims about product safety, final moisture content, and storage stability must be based on actual testing rather than assumptions.

Another ethical concern is responsible process selection. The team considered heating approaches such as oil-bath heating and water-bath heating, but the final concept uses molding followed by a low-power 12 V resistance-wire heat-preservation system wound around an aluminum basin. This choice reduces the risk of oil contamination, simplifies cleaning, lowers electrical and thermal hazard severity, and is more suitable for prototype testing.

Environmental responsibility is also relevant. The 24 W thermal limit reduces unnecessary energy use compared with excessive heating and keeps the thermal subsystem aligned with a heat-preservation function. Failed material batches, damaged electronic components, disposable printed parts, and other prototype waste should be minimized and handled properly.

10.2 Safety

The main safety risks come from the blade mechanism, feeding and molding structure, thermal subsystem, and electrical control system.

The blade area is the most direct mechanical hazard. A plastic barrier or cover should isolate the blade from the user during operation. The cover should prevent hand access while allowing material to pass through the designed path.

The thermal subsystem introduces burn and overheating risks even though the module is low power. The thermal region should include insulation or physical separation so users are not directly exposed to hot surfaces. Voltage, current, temperature, and heating duration should be limited through the control system. The machine should be opened for maintenance or product removal only after power is removed and the thermal section has cooled.

The feeding and molding structure may create pinch points or pressure buildup if the material is too wet or if clogging occurs. Feed rate and motor speed should be limited to maintain stable operation. The feeding path

should be guarded so that users cannot insert fingers into moving or compressed areas.

The electrical subsystem must be arranged so that wires, the 12 V resistance-wire heat-preservation system, and control components do not create additional hazards. Electronic components should be separated from hot zones and moving parts. Exposed wiring should be insulated, and the heating-wire switching path should be rated above 2 A with appropriate margin. The system should include controlled power delivery and a practical manual power-disconnect method during testing.

11. Limitations and Future Work

The current design is suitable for prototype validation, but several limitations remain before the system could be treated as a practical consumer device.

First, the material-contact surfaces must be upgraded for food-safe use. Aluminum basin/mold surfaces, printed parts, fasteners, and seals should be evaluated for cleaning, corrosion, residue buildup, and long-term use with moist feed material.

Second, the feeding system needs measured torque or motor current data. Without this data, clogging risk can only be evaluated visually. A future version should include stall detection, current monitoring, or pressure sensing to stop the motor before the material compacts dangerously.

Third, the heat-preservation verification needs measured electrical and temperature data under the 24 W resistance-wire heating limit. Future tests should record voltage, current, calculated power, basin surface temperature, and material surface warmth during the programmed PA6 interval. These tests should confirm safe warming behavior.

Fourth, the mold should be optimized for thermal uniformity, easy cleaning, and safe product removal after cooling. A removable mold insert or simpler tray geometry may improve usability during early prototype testing.

Finally, the control system could be expanded from timer-based operation to feedback-based operation in a future revision. The current final prototype removes the PA5 temperature-detection module, so voltage, current, and surface temperature checks for the 12 V resistance-wire heat-preservation system should be recorded manually during testing. A future revision could add a new temperature/current sensing circuit if closed-loop thermal control becomes necessary.

12. Conclusion

The automated homemade dog food production machine provides a feasible prototype architecture for small-batch pet feed processing. The design integrates mixing, controlled rotary extrusion, molding, and low-power resistance-wire heat preservation into one coordinated system. The main mechanical challenge is stable transport of moist material through a narrow outlet and mold. The main thermal challenge is keeping the molded material warm with a 12 V, 2 A, 24 W maximum resistance-wire heat-preservation module while avoiding overheating or uneven heat distribution. The main safety challenge is protecting users from rotating blades, moving feeding components, hot surfaces, and electrical hazards.

Based on the proposal analysis, the design is technically feasible if motor speed, outlet geometry, material

moisture content, and thermal-module operation are controlled carefully. The updated 24 W thermal subsystem should be treated as a warming and heat-preservation module. The next critical step is completing the verification table with measured feeding, heat-preservation, electrical, temperature, and safety results. Once those data are added, the report can support a final pass/fail assessment of the prototype.

References

[1] IEEE, "IEEE Code of Ethics." [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[2] Occupational Safety and Health Administration, "29 CFR 1910.212 - General requirements for all machines." [Online]. Available: https://www.osha.gov/laws-regs/regulations/standardnumber/1910/1910.212

[3] U.S. Food and Drug Administration, "21 CFR Part 507 - Current Good Manufacturing Practice, Hazard Analysis, and Risk-Based Preventive Controls for Food for Animals." Electronic Code of Federal Regulations. [Online]. Available: https://www.ecfr.gov/current/title-21/chapter-I/subchapter-E/part-507

[4] International Electrotechnical Commission, IEC 60204-1: Safety of machinery - Electrical equipment of machines - Part 1: General requirements. Geneva, Switzerland: IEC.

[5] Bilibili, "STM32 control of stepper motor tutorial" [Video]. [Online]. Available: https://b23.tv/kL3xbDn

Appendix A. Figure List

Appendix B. Data Needed Before Final Submission

1. Measured feeding result for 200 g batch.

2. Clogging observations at 20%, 30%, and 40% moisture content.

3. Heat-preservation test results: resistance-wire voltage, current, calculated power, basin surface temperature, and material surface warmth for each test condition.

4. Maximum measured aluminum-basin/chamber temperature during operation of the 12 V resistance-wire heat-preservation system.

5. Confirmation of actual heating-wire voltage, current, and power, with maximum values not exceeding 12 V, 2 A, and 24 W.

6. Confirmation that PA0 and PA5 are unused in the final prototype, with PA6 used as the timed low-power heat-preservation/warming output.

7. Safety check result for blade guard, hot-surface isolation, wiring, and manual power-disconnect behavior.

Appendix C. STM32 Control Program

This appendix explains the STM32F103C8T6 control program by functional blocks. The code was written for Keil MDK and is intended to be downloaded through ST-Link. The final program does not use PA0 as a total/start switch and does not read PA5 as a temperature input. Instead, the board starts its demonstration sequence automatically after power-on. PA1 controls the DC motor MOSFET, PA2 to PA4 control the A4988 stepper driver, PA6 controls the low-power heat-preservation/warming MOSFET, and PA7/TIM3_CH2 generates the MG90S servo PWM signal.

C.1 Pin Mapping and Timing Constants

This first block defines the electrical connection between the STM32 and the PCB. It also defines the operating timing for the stepper motor, servo, startup delay, and PA6 heat-preservation interval. These constants make the program easier to tune because motor speed, servo timing, and heater duration can be changed from one location.

```
#define DC_MOTOR_PIN         1U  /* PA1: DC motor MOSFET control */
#define A4988_EN_PIN         2U  /* PA2: A4988 enable, active low */
#define A4988_STEP_PIN       3U  /* PA3: A4988 step pulse */
#define A4988_DIR_PIN        4U  /* PA4: A4988 direction */
#define HEATER_PIN           6U  /* PA6: heat-preservation/warming MOSFET control */
#define SERVO_PIN            7U  /* PA7: MG90S servo PWM */

#define STEPPER_PPS          4500UL
#define STEP_INTERVAL_US     (1000000UL / STEPPER_PPS)

#define STEPPER_FORWARD_MS   8000UL
#define STEPPER_STOP_MS      5000UL
#define STEPPER_REVERSE_MS   8000UL
#define LOOP_PERIOD_MS       21000UL

#define SERVO_0_US           500U
#define SERVO_90_US          1500U
#define SERVO_WAIT_0_MS      12000UL
#define SERVO_MOVE_TO_90_MS  1000U
#define SERVO_HOLD_90_MS     6000U
#define SERVO_MOVE_TO_0_MS   2000UL

#define HEATER_ON_MS         100000UL
#define START_DELAY_MS       5000UL
```

In this version, PA6 is not a temperature-controlled closed-loop output. It is a timed warming/heat-preservation output: after `start_system()` is called, PA6 remains high for `HEATER_ON_MS`, which is 100 seconds in the current code. The heat-preservation module itself is low power, so the thermal behavior must still be checked using external temperature measurement during validation.

C.2 Software Time Base

The program avoids defining a `SysTick_Handler` because the Keil project already had a SysTick-related symbol conflict. Instead, TIM2 is used as a microsecond counter and TIM4 is used as a software millisecond counter. This gives the program two useful time scales: microseconds for step pulses and milliseconds for motor/servo/heater scheduling.

```
static uint16_t micros16(void)
{
    return (uint16_t)TIM2->CNT;
}

static void timebase_update(void)
{
    if (TIM4->SR & TIM_SR_UIF) {
        TIM4->SR &= (uint16_t)~TIM_SR_UIF;
        ms_count++;
    }
}

static uint32_t millis(void)
{
    timebase_update();
    return ms_count;
}

static void delay_us(uint16_t us)
{
    uint16_t start;

    start = micros16();
    while ((uint16_t)(micros16() - start) < us) {
    }
}
```

`micros16()` reads TIM2 directly. `millis()` updates and returns the software millisecond count. `delay_us()` is used only for short delays, such as the 3 microsecond A4988 STEP pulse and the small delay after changing the DIR pin.

C.3 GPIO Configuration and Output Writing

This block configures the STM32 GPIO pins used by the PCB. PA1, PA2, PA3, PA4, and PA6 are configured as push-pull outputs. PA7 is configured as an alternate-function push-pull output because TIM3_CH2 generates the MG90S PWM signal on that pin.

```
static void gpio_mode(GPIO_TypeDef *port, uint8_t pin, uint8_t mode)
{
    uint32_t shift;

    shift = (uint32_t)pin * 4U;
    port->CRL &= ~(0x0FUL << shift);
    port->CRL |= ((uint32_t)mode << shift);
}

static void pin_write(uint8_t pin, uint8_t level)
{
    if (level) {
        GPIOA->BSRR = 1UL << pin;
    } else {
        GPIOA->BRR = 1UL << pin;
    }
}

static void gpio_init_all(void)
{
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN | RCC_APB2ENR_AFIOEN;

    gpio_mode(GPIOA, DC_MOTOR_PIN, 0x02U);
    gpio_mode(GPIOA, A4988_EN_PIN, 0x02U);
```

```
    gpio_mode(GPIOA, A4988_STEP_PIN, 0x02U);
    gpio_mode(GPIOA, A4988_DIR_PIN, 0x02U);
    gpio_mode(GPIOA, HEATER_PIN, 0x02U);
    gpio_mode(GPIOA, SERVO_PIN, 0x0BU);

    AFIO->MAPR &= ~AFIO_MAPR_TIM3_REMAP;

    pin_write(DC_MOTOR_PIN, 1U);
    pin_write(HEATER_PIN, 1U);
    pin_write(A4988_EN_PIN, 1U);
    pin_write(A4988_STEP_PIN, 0U);
    pin_write(A4988_DIR_PIN, 0U);
}
```

The important practical point is that PA1 and PA6 are set high after GPIO initialization. If an oscilloscope shows PA1 or PA6 low during reset, that does not necessarily mean the program is wrong. During reset, STM32 GPIO pins are still in their default input state. If the MOSFET gate has an external pull-down resistor, the pin can appear low until the firmware has started and configured the pin as an output.

C.4 Servo PWM Generation

The MG90S servo is driven from PA7 using TIM3 channel 2. The timer is configured for a 20 ms period, which is the normal 50 Hz servo-control frame. The program changes the CCR2 compare value to change the high-pulse width and therefore the servo angle.

```
static void servo_write_us(uint16_t us)
{
    TIM3->CCR2 = us;
}

static uint16_t servo_lerp(uint16_t from, uint16_t to, uint16_t elapsed, uint16_t total)
{
    int32_t diff;

    if (elapsed >= total) {
        return to;
    }

    diff = (int32_t)to - (int32_t)from;
    return (uint16_t)((int32_t)from + diff * (int32_t)elapsed / (int32_t)total);
}

static void tim3_init_servo_pwm(void)
{
    RCC->APB1ENR |= RCC_APB1ENR_TIM3EN;

    TIM3->PSC = (uint16_t)(SystemCoreClock / 1000000U - 1U);
    TIM3->ARR = 20000U - 1U;
    TIM3->CCR2 = SERVO_0_US;

    TIM3->CCMR1 &= ~((3UL << 8) | (7UL << 12) | TIM_CCMR1_OC2PE);
    TIM3->CCMR1 |= (6UL << 12) | TIM_CCMR1_OC2PE;
    TIM3->CCER &= ~TIM_CCER_CC2P;
    TIM3->CCER |= TIM_CCER_CC2E;
    TIM3->CR1 |= TIM_CR1_ARPE;
    TIM3->EGR = TIM_EGR_UG;
    TIM3->CR1 |= TIM_CR1_CEN;
}
```

`servo_lerp()` creates a smooth transition between two pulse widths instead of jumping immediately from one angle to another. This reduces sudden movement in the mechanical linkage.

C.5 Stepper Motor Control

The 42 stepper motor is driven through the A4988 module. PA2 enables the driver, PA3 sends STEP pulses, and PA4 selects the direction. The code uses a three-state cycle: forward, stop, and reverse.

```
typedef enum {
    STEPPER_FORWARD = 0,
    STEPPER_STOP,
    STEPPER_REVERSE
} StepperState;

static void stepper_one_pulse(void)
{
    pin_write(A4988_STEP_PIN, 1U);
    delay_us(3U);
    pin_write(A4988_STEP_PIN, 0U);
}

static void stepper_task(void)
{
    uint32_t now_ms;
    uint16_t now_us;

    now_ms = millis();

    if (stepper_state == STEPPER_FORWARD) {
        if ((now_ms - stepper_state_start_ms) >= STEPPER_FORWARD_MS) {
            stepper_state = STEPPER_STOP;
            stepper_state_start_ms = now_ms;
            pin_write(A4988_STEP_PIN, 0U);
            return;
        }
        pin_write(A4988_DIR_PIN, 0U);
    } else if (stepper_state == STEPPER_STOP) {
        pin_write(A4988_STEP_PIN, 0U);
        if ((now_ms - stepper_state_start_ms) >= STEPPER_STOP_MS) {
            stepper_state = STEPPER_REVERSE;
            stepper_state_start_ms = now_ms;
            stepper_next_step_us = micros16();
            pin_write(A4988_DIR_PIN, 1U);
            delay_us(20U);
        }
        return;
    } else {
        if ((now_ms - stepper_state_start_ms) >= STEPPER_REVERSE_MS) {
            stepper_state = STEPPER_FORWARD;
            stepper_state_start_ms = now_ms;
            stepper_next_step_us = micros16();
            pin_write(A4988_DIR_PIN, 0U);
            delay_us(20U);
            return;
        }
        pin_write(A4988_DIR_PIN, 1U);
    }

    now_us = micros16();
    if ((uint16_t)(now_us - stepper_next_step_us) >= STEP_INTERVAL_US) {
        stepper_one_pulse();
        stepper_next_step_us = (uint16_t)(stepper_next_step_us + STEP_INTERVAL_US);
    }
}
```

The current pulse rate is defined by `STEPPER_PPS`. If the motor vibrates, stalls, or loses steps under load, this value should be reduced first before changing the mechanical design.

C.6 Heat-Preservation Output and Main Loop

The startup function enables the DC motor, enables the A4988 driver, sets the stepper direction, turns PA6 high, and initializes the servo position. The heat-preservation task then keeps PA6 high until 100 seconds have elapsed, after which it turns PA6 low and marks the heat-preservation interval as finished.

```
static void start_system(void)
{
    stepper_state = STEPPER_FORWARD;
    stepper_state_start_ms = millis();
    stepper_next_step_us = micros16();
    heater_finished = 0U;
    heater_start_ms = millis();

    pin_write(DC_MOTOR_PIN, 1U);
    pin_write(A4988_EN_PIN, 0U);
    pin_write(A4988_DIR_PIN, 0U);
    pin_write(A4988_STEP_PIN, 0U);
    pin_write(HEATER_PIN, 1U);
    servo_write_us(SERVO_0_US);
}

static void heater_task(void)
{
    uint32_t now;

    now = millis();

    if (heater_finished || (now - heater_start_ms) >= HEATER_ON_MS) {
        pin_write(HEATER_PIN, 0U);
        heater_finished = 1U;
        return;
    }

    pin_write(HEATER_PIN, 1U);
}
```

The main loop keeps PA1 high, waits 5 seconds before stepper/servo motion begins, then repeatedly calls the stepper task, servo task, and heat-preservation task. This matches the final simplified operation: the board starts automatically after power-on rather than waiting for a PA0 button input, and no PA5 temperature-detection input is read.

```
while (1) {
    timebase_update();
    pin_write(DC_MOTOR_PIN, 1U);

    elapsed_ms = millis() - start_time;

    if (!cycle_started) {
        pin_write(A4988_STEP_PIN, 0U);
        servo_write_us(SERVO_0_US);

        if (elapsed_ms < START_DELAY_MS) {
            heater_task();
            continue;
        }

        cycle_started = 1U;
        start_time = millis();
        elapsed_ms = 0U;
        stepper_state = STEPPER_FORWARD;
        stepper_state_start_ms = start_time;
        stepper_next_step_us = micros16();
        pin_write(A4988_DIR_PIN, 0U);
```

```
        pin_write(A4988_STEP_PIN, 0U);
    }

    cycle_ms = (uint16_t)(elapsed_ms % LOOP_PERIOD_MS);

    stepper_task();
    servo_task(cycle_ms);
    heater_task();
}
```

C.7 Complete Source Listing

The complete source code used for the final demonstration is included below.

```
#include "stm32f10x.h"
#include <stdint.h>

/*
 * MCU: STM32F103C8T6
 * MCU 是 Micro Controller Unit，中文通常叫“单片机/微控制器”。
 * 这颗 STM32 负责按照程序输出 GPIO 电平、PWM 和步进电机脉冲。
 *
 * Keil MDK + STM32F10x Standard Peripheral/CMSIS header
 *
 * 引脚和电路图对应关系：
 * PA1: 直流电机 MOS 管控制，高电平 = 电机运行
 * PA2: A4988 ENABLE，低电平 = 使能步进驱动器
 * PA3: A4988 STEP，步进脉冲脚，每个脉冲让步进电机走一步
 * PA4: A4988 DIRECTION，方向脚，0/1 对应两个方向
 * PA5: 未使用
 * PA6: 加热/负载 MOS 管控制，高电平 = 加热，持续 100 秒后关闭
 * PA7: MG90S 舵机 PWM，TIM3_CH2 输出
 *
 * 断电后再上电不会保持原状态：本程序没有把状态写入 Flash、EEPROM
 * 或 RTC backup register，所以上电后会从 main() 重新开始执行。
 *
 * This version does not define SysTick_Handler.
 * It avoids the Keil link error:
 * "Symbol SysTick_Handler multiply defined".
 */

#define DC_MOTOR_PIN         1U  /* PA1: 直流电机控制 */
#define A4988_EN_PIN         2U  /* PA2: A4988 使能，低电平有效 */
#define A4988_STEP_PIN       3U  /* PA3: A4988 步进脉冲 */
#define A4988_DIR_PIN        4U  /* PA4: A4988 方向 */
#define HEATER_PIN           6U  /* PA6: 加热/负载控制 */
#define SERVO_PIN            7U  /* PA7: MG90S 舵机 PWM */

#define STEPPER_PPS          4500UL                  /* 步进电机目标脉冲频率：4500 pulse/s */
#define STEP_INTERVAL_US     (1000000UL / STEPPER_PPS) /* 两个 STEP 脉冲之间的间隔，单位 us */

#define STEPPER_FORWARD_MS   8000UL  /* 步进电机正转 8 秒 */
#define STEPPER_STOP_MS      5000UL  /* 停止 5 秒 */
#define STEPPER_REVERSE_MS   8000UL  /* 反转 8 秒 */
#define LOOP_PERIOD_MS       21000UL /* 舵机动作周期：12 + 1 + 6 + 2 = 21 秒 */

#define SERVO_0_US           500U    /* MG90S 约 0 度脉宽，实际角度需要按舵机校准 */
#define SERVO_90_US          1500U   /* MG90S 约 90 度脉宽 */
#define SERVO_WAIT_0_MS      12000UL /* 每个周期前 12 秒保持 0 度 */
#define SERVO_MOVE_TO_90_MS  1000U   /* 用 1 秒从 0 度转到 90 度 */
#define SERVO_HOLD_90_MS     6000U   /* 保持 90 度 6 秒 */
#define SERVO_MOVE_TO_0_MS   2000UL  /* 用 2 秒从 90 度回到 0 度 */

#define HEATER_ON_MS         100000UL /* PA6 加热输出保持 100 秒 */
#define START_DELAY_MS       5000UL   /* 上电后步进电机和舵机先等待 5 秒 */
```

```
typedef enum {
    STEPPER_FORWARD = 0, /* 正转阶段 */
    STEPPER_STOP,        /* 停止阶段 */
    STEPPER_REVERSE      /* 反转阶段 */
} StepperState;

static uint16_t stepper_next_step_us = 0U;       /* 下一次 STEP 脉冲的时间点，单位 us */
static uint32_t ms_count = 0U;                   /* TIM4 维护的软件毫秒计数 */
static uint32_t stepper_state_start_ms = 0U;     /* 当前步进电机阶段的开始时间 */
static StepperState stepper_state = STEPPER_FORWARD;
static uint8_t heater_finished = 0U;             /* 1 表示加热 100 秒已经结束 */
static uint32_t heater_start_ms = 0U;            /* 保温开始时间 */

/* 读取 TIM2 当前计数值。TIM2 被配置成 1us 计数一次。 */
static uint16_t micros16(void)
{
    return (uint16_t)TIM2->CNT;
}

/* TIM4 每 1ms 产生一次更新标志，看到标志后清除并让 ms_count 加 1。 */
static void timebase_update(void)
{
    if (TIM4->SR & TIM_SR_UIF) {
        TIM4->SR &= (uint16_t)~TIM_SR_UIF;
        ms_count++;
    }
}

/* 返回系统运行的毫秒数。 */
static uint32_t millis(void)
{
    timebase_update();
    return ms_count;
}

/* 基于 TIM2 的短延时，主要用于 STEP 脉冲和方向切换后的等待。 */
static void delay_us(uint16_t us)
{
    uint16_t start;

    start = micros16();
    while ((uint16_t)(micros16() - start) < us) {
    }
}

/* 配置 GPIO 模式。STM32F103 的 PA0~PA7 都在 CRL 寄存器中，每个引脚占 4 bit。 */
static void gpio_mode(GPIO_TypeDef *port, uint8_t pin, uint8_t mode)
{
    uint32_t shift;

    shift = (uint32_t)pin * 4U;
    port->CRL &= ~(0x0FUL << shift);
    port->CRL |= ((uint32_t)mode << shift);
}

/* 写 GPIOA 输出电平：level=1 输出高电平，level=0 输出低电平。 */
static void pin_write(uint8_t pin, uint8_t level)
{
    if (level) {
        GPIOA->BSRR = 1UL << pin;
    } else {
        GPIOA->BRR = 1UL << pin;
    }
}

/* 设置 PA7/TIM3_CH2 的 PWM 高电平时间，MG90S 靠这个脉宽判断角度。 */
static void servo_write_us(uint16_t us)
{
    TIM3->CCR2 = us;
}

/* 舵机平滑移动：按经过时间 elapsed，在 from 和 to 两个脉宽之间线性插值。 */
```

```
static uint16_t servo_lerp(uint16_t from, uint16_t to, uint16_t elapsed, uint16_t total)
{
    int32_t diff;

    if (elapsed >= total) {
        return to;
    }

    diff = (int32_t)to - (int32_t)from;
    return (uint16_t)((int32_t)from + diff * (int32_t)elapsed / (int32_t)total);
}

/* 给 A4988 的 STEP 脚输出一个 3us 高脉冲。 */
static void stepper_one_pulse(void)
{
    pin_write(A4988_STEP_PIN, 1U);
    delay_us(3U);
    pin_write(A4988_STEP_PIN, 0U);
}

/* 初始化所有用到的 GPIO。
 * 0x02 = 通用推挽输出，最高 2MHz；0x0B = 复用推挽输出，最高 50MHz。
 */
static void gpio_init_all(void)
{
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN | RCC_APB2ENR_AFIOEN;

    gpio_mode(GPIOA, DC_MOTOR_PIN, 0x02U);
    gpio_mode(GPIOA, A4988_EN_PIN, 0x02U);
    gpio_mode(GPIOA, A4988_STEP_PIN, 0x02U);
    gpio_mode(GPIOA, A4988_DIR_PIN, 0x02U);
    gpio_mode(GPIOA, HEATER_PIN, 0x02U);
    gpio_mode(GPIOA, SERVO_PIN, 0x0BU);

    /* TIM3 默认映射：CH2 从 PA7 输出。 */
    AFIO->MAPR &= ~AFIO_MAPR_TIM3_REMAP;

    /* PA1 和 PA6 初始化后直接置高，减少程序初始化期间的低电平时间。
     * 注意：STM32 刚复位时 GPIO 仍是默认输入状态；如果外部有 10k 下拉，
     * 从复位到执行到这里之前，引脚仍可能被外部电阻拉低一小段时间。
     */
    pin_write(DC_MOTOR_PIN, 1U);
    pin_write(HEATER_PIN, 1U);

    /* A4988 的 ENABLE 是低电平有效。初始化时先关闭，start_system() 再打开。 */
    pin_write(A4988_EN_PIN, 1U);
    pin_write(A4988_STEP_PIN, 0U);
    pin_write(A4988_DIR_PIN, 0U);
}

/* TIM2 做 1us 计数器，用于产生步进脉冲间隔和短延时。 */
static void tim2_init_1us(void)
{
    RCC->APB1ENR |= RCC_APB1ENR_TIM2EN;

    TIM2->PSC = (uint16_t)(SystemCoreClock / 1000000U - 1U);
    TIM2->ARR = 0xFFFFU;
    TIM2->CNT = 0U;
    TIM2->EGR = TIM_EGR_UG;
    TIM2->CR1 = TIM_CR1_CEN;
}

/* TIM4 做 1ms 软件时基，ms_count 相当于系统运行毫秒数。 */
static void tim4_init_1ms(void)
{
    RCC->APB1ENR |= RCC_APB1ENR_TIM4EN;

    TIM4->PSC = (uint16_t)(SystemCoreClock / 10000U - 1U);
    TIM4->ARR = 10U - 1U;
    TIM4->CNT = 0U;
    TIM4->EGR = TIM_EGR_UG;
    TIM4->SR = 0U;
```

```
    TIM4->CR1 = TIM_CR1_CEN;
}

/* TIM3_CH2 输出 50Hz 舵机 PWM: 周期 20ms，高电平约 0.5ms~1.5ms。 */
static void tim3_init_servo_pwm(void)
{
    RCC->APB1ENR |= RCC_APB1ENR_TIM3EN;

    TIM3->PSC = (uint16_t)(SystemCoreClock / 1000000U - 1U);
    TIM3->ARR = 20000U - 1U;
    TIM3->CCR2 = SERVO_0_US;

    TIM3->CCMR1 &= ~((3UL << 8) | (7UL << 12) | TIM_CCMR1_OC2PE);
    TIM3->CCMR1 |= (6UL << 12) | TIM_CCMR1_OC2PE;
    TIM3->CCER &= ~TIM_CCER_CC2P;
    TIM3->CCER |= TIM_CCER_CC2E;
    TIM3->CR1 |= TIM_CR1_ARPE;
    TIM3->EGR = TIM_EGR_UG;
    TIM3->CR1 |= TIM_CR1_CEN;
}

/* 启动系统动作：电机运行、A4988 使能、保温模块打开、舵机回到 0 度。 */
static void start_system(void)
{
    stepper_state = STEPPER_FORWARD;
    stepper_state_start_ms = millis();
    stepper_next_step_us = micros16();
    heater_finished = 0U;
    heater_start_ms = millis();

    pin_write(DC_MOTOR_PIN, 1U);
    pin_write(A4988_EN_PIN, 0U);
    pin_write(A4988_DIR_PIN, 0U);
    pin_write(A4988_STEP_PIN, 0U);
    pin_write(HEATER_PIN, 1U);
    servo_write_us(SERVO_0_US);
}

/* 步进电机任务：8 秒正转，5 秒停止，8 秒反转，然后循环。 */
static void stepper_task(void)
{
    uint32_t now_ms;
    uint16_t now_us;

    now_ms = millis();

    if (stepper_state == STEPPER_FORWARD) {
        if ((now_ms - stepper_state_start_ms) >= STEPPER_FORWARD_MS) {
            stepper_state = STEPPER_STOP;
            stepper_state_start_ms = now_ms;
            pin_write(A4988_STEP_PIN, 0U);
            return;
        }

        pin_write(A4988_DIR_PIN, 0U);
    } else if (stepper_state == STEPPER_STOP) {
        pin_write(A4988_STEP_PIN, 0U);

        if ((now_ms - stepper_state_start_ms) >= STEPPER_STOP_MS) {
            stepper_state = STEPPER_REVERSE;
            stepper_state_start_ms = now_ms;
            stepper_next_step_us = micros16();
            pin_write(A4988_DIR_PIN, 1U);
            delay_us(20U);
        }
        return;
    } else {
        if ((now_ms - stepper_state_start_ms) >= STEPPER_REVERSE_MS) {
            stepper_state = STEPPER_FORWARD;
            stepper_state_start_ms = now_ms;
            stepper_next_step_us = micros16();
            pin_write(A4988_DIR_PIN, 0U);
            delay_us(20U);
            return;
```

```
        }

        pin_write(A4988_DIR_PIN, 1U);
    }

    now_us = micros16();
    if ((uint16_t)(now_us - stepper_next_step_us) >= STEP_INTERVAL_US) {
        stepper_one_pulse();
        stepper_next_step_us = (uint16_t)(stepper_next_step_us + STEP_INTERVAL_US);
    }
}

/* MG90S 舵机任务：0 度等待 12 秒，1 秒转到 90 度，保持 6 秒，2 秒回 0 度。 */
static void servo_task(uint16_t cycle_ms)
{
    if (cycle_ms < SERVO_WAIT_0_MS) {
        servo_write_us(SERVO_0_US);
    } else if (cycle_ms < (SERVO_WAIT_0_MS + SERVO_MOVE_TO_90_MS)) {
        servo_write_us(servo_lerp(SERVO_0_US, SERVO_90_US, (uint16_t)(cycle_ms - SERVO_WAIT_0_MS),
SERVO_MOVE_TO_90_MS));
    } else if (cycle_ms < (SERVO_WAIT_0_MS + SERVO_MOVE_TO_90_MS + SERVO_HOLD_90_MS)) {
        servo_write_us(SERVO_90_US);
    } else {
        servo_write_us(servo_lerp(SERVO_90_US, SERVO_0_US,
                                  (uint16_t)(cycle_ms - SERVO_WAIT_0_MS - SERVO_MOVE_TO_90_MS -
SERVO_HOLD_90_MS),
                                  SERVO_MOVE_TO_0_MS));
    }
}

/* 保温任务：从 start_system() 开始计时，PA6 保持高电平 100 秒，然后关闭。 */
static void heater_task(void)
{
    uint32_t now;

    now = millis();

    if (heater_finished || (now - heater_start_ms) >= HEATER_ON_MS) {
        pin_write(HEATER_PIN, 0U);
        heater_finished = 1U;
        return;
    }

    pin_write(HEATER_PIN, 1U);
}

/* 主程序入口。断电重启后会从这里重新开始，不会记住上次运行到哪里。 */
int main(void)
{
    uint32_t start_time;
    uint32_t elapsed_ms;
    uint16_t cycle_ms;
    uint8_t cycle_started;

    start_time = 0U;
    cycle_started = 0U;

    SystemCoreClockUpdate();

    gpio_init_all();
    tim2_init_1us();
    tim4_init_1ms();
    tim3_init_servo_pwm();

    start_system();
    start_time = millis();

    while (1) {
        timebase_update();

        /* PA1 直流电机始终保持高电平运行。 */
        pin_write(DC_MOTOR_PIN, 1U);
```

```
        elapsed_ms = millis() - start_time;

        if (!cycle_started) {
            /* 上电前 5 秒：步进电机不发脉冲，舵机保持 0 度，保温模块仍然打开。 */
            pin_write(A4988_STEP_PIN, 0U);
            servo_write_us(SERVO_0_US);

            if (elapsed_ms < START_DELAY_MS) {
                heater_task();
                continue;
            }

            /* 5 秒延时结束后，开始步进电机和舵机的周期动作。 */
            cycle_started = 1U;
            start_time = millis();
            elapsed_ms = 0U;
            stepper_state = STEPPER_FORWARD;
            stepper_state_start_ms = start_time;
            stepper_next_step_us = micros16();
            pin_write(A4988_DIR_PIN, 0U);
            pin_write(A4988_STEP_PIN, 0U);
        }

        cycle_ms = (uint16_t)(elapsed_ms % LOOP_PERIOD_MS);

        stepper_task();
        servo_task(cycle_ms);
        heater_task();
    }
}
```