

Politecnico di Torino

Politecnico di Torino

Master's Degree in Mechatronics Engineering
A.Y. 2025/2026
Graduation Session September 2026

Design and Manufacture of a Biomimetic Bird Robot for Song Learning Research

Supervisors:
Ph.D. Pangcheng David Cen Cheng
Prof. Alexandre Pitti

Candidate:
Stefano Marrone

Abstract

Songbirds learn their song by imitating adults during a sensitive period of development, which makes them the standard model for studying vocal learning. Laboratory work on this process faces a trade-off: playing recordings gives complete control over the stimulus but produces poor imitation, whereas a live adult tutor produces near-perfect copies but cannot be controlled or repeated. A robotic bird is an attempt to have both — a physical object that moves and reacts to the subject, but which sings exactly what the protocol specifies: which songs are played, at what moments, how often, and in response to what behaviour of the subject.

This thesis describes a work carried out at the ETIS laboratory (CY Cergy Paris Université, ENSEA, CNRS) within the ANR project nIrVAna, on the Canaribot: a biomimetic canary robot used as a song tutor by the ethology laboratory of Université Paris Nanterre. The robot consists of a 3D-printed bird mounted on an electronics enclosure, with two rotation axes, a loudspeaker, instrumented perches, and a microphone. It is controlled by a Raspberry Pi running a C++ application, which drives the hardware through the pigpio library and exposes the robot over a WebSocket interface to a browser-based client used to prepare and run experimental sessions. The bird models were prepared in Blender and SolidWorks and produced by fused deposition modelling.

The work had two objectives: to produce and repair units in response to the requests of the end user, and to improve the existing mechatronics in terms of robustness and reproducibility. The bird models inherited at the start were dimensionally inaccurate, with problems such as the base not passing through the opening in the lid, the bore not engaging the drive shaft firmly enough to transmit rotation, and the head motor shaft not protruding far enough for the head to seat, so every unit had to be adjusted by hand during assembly and could not be reproduced in numbers.

The geometry was revised so that printed parts fit as produced, and the fixation of the head was redesigned to withstand the pecking the robots receive in use. Two new actuator mechanisms were developed for the beak and the tail, adding visual components to accompany the vocalisation: the beak opens in time with the song, so that what the subject sees matches what it hears, and the tail moves alongside it as a further visual cue. Both mechanisms were built and verified on the assembled bird, though neither was driven from the electronics box within the time available. A batch of ten electronics boards was assembled, with commissioning carried out on the completed ones, and the electronics and its interfaces with the firmware and the bird were documented where they previously were not.

Contents

Abstract **i**

1 Introduction **1**
1.1 Scientific context . . . 1
1.2 Why a robot . . . 1
1.3 Host laboratory and project organisation . . . 2
1.4 Requested deliverables . . . 3
1.5 Objectives . . . 5
1.6 Contributions . . . 5
1.7 Outline of the thesis . . . 6

2 Background **7**
2.1 What the robot has to do, and why . . . 7
2.2 The zebra finch variant . . . 8
2.3 What follows for the design . . . 9

3 The Canaribot System **11**
3.1 The experimental scenario . . . 12
3.2 Physical architecture . . . 13
3.3 Degrees of freedom of the bird . . . 18
3.4 Sensor system setup . . . 18
3.5 Audio subsystem . . . 19
3.6 Control and communication . . . 20
3.7 Starting point . . . 20
3.8 The paired-box configuration . . . 21

4 Mechanical Design of the Bird Unit **23**
4.1 Design constraints . . . 23
4.2 Baseline geometry . . . 24
4.3 Redesign of the bird . . . 25
4.3.1 Base and interface to the body motor . . . 25
4.3.2 Head and neck . . . 27
4.3.3 Tail . . . 30
4.3.4 Female variant model . . . 31
4.4 Additive manufacturing . . . 32
4.5 Open problems . . . 33

5 Actuator Development **35**
5.1 What each actuator has to do . . . 35
5.2 Beak . . . 36
5.2.1 Working principle . . . 36
5.2.2 Drive electronics . . . 37
5.2.3 Current and dissipation . . . 37
5.2.4 Status . . . 37
5.3 Tail . . . 38
5.3.1 Transmission . . . 38
5.3.2 Drive electronics . . . 38
5.3.3 Current and dissipation . . . 38
5.3.4 Status . . . 40
5.4 Female variant: bowing mechanism . . . 40
5.4.1 The display and what it requires . . . 40
5.4.2 What was built . . . 40
5.4.3 Proposed actuation . . . 41
5.5 Comparison of the three actuator types . . . 44

6 Electronic System **47**
6.1 Overview . . . 47
6.2 Power distribution . . . 51
6.3 Stepper drivers . . . 52
6.4 Beak and tail driver . . . 53
6.5 Load cell front end . . . 55
6.6 Audio . . . 55
6.7 Connectors and harness . . . 56
6.8 Assembly . . . 58
6.9 Commissioning the assembled boards . . . 59
6.9.1 Method . . . 59
6.9.2 Faults found . . . 59
6.9.3 The beak pin assignment . . . 61
6.9.4 The converter return path . . . 61
6.9.5 The amplifier shutdown input . . . 61
6.9.6 Rebuilding rather than continuing . . . 62

7 Firmware and Control Software **63**
7.1 Environment . . . 63
7.2 Structure . . . 64
7.3 Concurrency model . . . 64
7.4 Command protocol . . . 66
7.5 Motion control . . . 66
7.6 Actuator control . . . 67
7.7 Vocalisation detection . . . 68
7.8 Client application . . . 68
7.9 Software modifications . . . 68

8 Results and Discussion **71**
8.1 Assessment against the deliverables and objectives 71
8.2 What was improved 73
8.3 Reproducibility 74
8.4 Limitations 75

9 Conclusions and Future Work **77**
9.1 Summary 77
9.2 Completing the work begun here 78
9.3 Further development 79
9.4 Experiments the work makes possible 79

References **81**

List of Figures

3.1 A complete Canaribot unit: the bird mounted on the lid of the electronics box, with the two instrumented perches on either side of it. . 11

3.2 System-level block diagram. The enclosure holds the subject bird and the robot, which is mounted on the lid of the electronics box together with the two instrumented perches and the limit switch. Inside the box are the Raspberry Pi, the main board and the power supply. Sensing paths run from the perches and the microphone into the box, actuation paths run from the board to the robot, and a WebSocket link connects the Raspberry Pi to the laptop running the client application. 13

3.3 Two experimental sessions, seen from the monitoring station. Because the bird and the electronics box are separate units, the robot presented to a subject need not be of the subject's own species, which allows the appearance of the tutor and the song it delivers to be varied independently of one another. 14

3.4 Interior of the electronics box, showing the Raspberry Pi, the main board, the body motor, the USB sound card and the various connecting cables. 16

3.5 Lid of the box. (a) Seen from above: the two perches on either side of the central opening through which the drive shaft and the wiring pass. (b) Seen from below: the load cells supporting the perches, and the limit switch against which the locking block of the bird comes to rest. 17

3.6 Cross-section of the two-unit architecture. 17

4.1 The parts of the bird, shown as CAD models. The body carries the interface to the drive shaft at its base and houses the head motor holder at its upper end; the neck disc fits onto the motor shaft and carries the head, into which the beak is inserted through the mouth opening; the two wings are held by the fixation pin passing through them inside the body; and the tail is fitted at the rear. The parts are not shown to the same scale. 26

4.2 The assembled bird. 27

4.3 The interface between the bird and the body motor. The base of the bird carries a D-shaped bore that engages the flat of the drive shaft, a rectangular block projecting from the side of the cylindrical boss, which locks the bird under the lid and reaches the limit switch as the body rotates, and an opening through which the wiring leaves the body and enters the box. The flattened profile of the shaft is what prevents the two from rotating relative to one another. 28
4.4 Bird mounted on the lid with the lid raised to expose the underside of the mounting interface. The rectangular block on the base is shown in contact with the limit switch at the homing position of the body axis. 29
4.5 The head-to-body interface with the head removed. (a) The shaft of the head motor protruding from the upper part of the body, with the TPU disc fitted over it. (b) The neck disc, showing the TPU insert around the opening that receives the shaft. 30
4.6 The split body of the female variant. The body is divided horizontally between the base and the lower belly, with lugs on either side of the split carrying the screw that forms the pivot, and cavities that allow the wiring to cross the joint. The upper body rotates forward about this pivot while the base stays fixed to the drive shaft. 32
4.7 Preparation of a print in Bambu Studio [19]. The parts are positioned and oriented on the build plate so that supported surfaces end up inside the assembly rather than on the visible outside of the bird, and so that features loaded transversely are not stressed across the layer interfaces. The slicer then generates the support material and the tool path. 33

5.1 Components of the beak actuation mechanism: (a) the lower mandible carrying the bonded magnet, (b) the lever mechanism mounted inside the head, and (c) the driving coil mounted in its holder on the neck disc (the coil shown is a worn specimen used during various tests). . . 36
5.2 Tail actuation mechanism as built, showing the coil mounted inside the 3D-printed body shell and the actuator lever directly engaging the recess in the tail. 39
5.3 Kinematic scheme of the proposed bowing mechanism. The base stays fixed to the drive shaft and the upper body rotates about the pivot at the split, into the position shown dashed. The servo is mounted on the base below the lid of the box and draws a steel wire upward to the pin that joins the wings; since that point lies ahead of the pivot, the body tilts forward. The spring runs between the same point and an anchorage on the base above the lid, so that tilting extends it and releasing the wire lets it return the body upright. 42

6.1 Complete schematic of the Canaribot V1.0 board, showing the module sockets, the power distribution, and the connectors towards the Raspberry Pi, the bird and the sensors. 49

6.2 The assembled Canaribot V1.0 board. The modules are: (1) the Traco Power TEN 25-1211 converter, which derives the 5 V rail from the external supply; (2) the two SilentStepStick TMC2130 drivers for the head and body axes; (3) the TB6612FNG dual H-bridge on its Pololu carrier, which drives the beak and tail coils; (4) the two HX711 modules amplifying and digitising the load cell signals; and (5) the PAM8302A audio amplifier. The interfaces towards the Raspberry Pi are (6) the 40-pin IDC header carrying the whole GPIO and (7) the USB connector supplying it. Power enters through (8) the barrel jack and is distributed by (13) the screw terminals for the 5 V and 12 V rails. The remaining connectors serve the robot and the sensors: (9) the Hirose DF11 connector towards the bird, (10) the load cell headers, (11) the limit switch headers, (12) the body motor header, and (14) the audio input jack from the USB sound card. 50
6.3 Audio section of the schematic. The signal from the USB sound card enters at the jack connector J18, whose tip and sleeve drive the two signal inputs of the PAM8302A amplifier (U6) while the ring is left unconnected, the source being monophonic. The screw terminal J19 carries the loudspeaker connection towards J2 and the harness; the output stage of the amplifier is not routed on the board and reaches J19 through wires fitted during assembly. The shutdown input SD is not driven by the board, the carrier providing its own pull-up. 56

7.1 Structure of the embedded application. The `Server` owns the `Client` and holds a reference to `Canari`, which owns the two `HX711` objects and calls the two stepper modules and the microphone module, shown in grey as collections of free functions rather than classes. 65
7.2 The client interface. The operator first enters the network address of the robot and selects the bird whose stimulus library is to be used; the manual controls then allow each function to be exercised individually, which is how a unit is checked before a session and during repair work; and the planner runs the timed sequence of events, with every command, stimulus, perch trigger and detected vocalisation recorded in the log below it. 69

List of Tables

1.1 Work carried out in response to the needs of Nanterre, with the technical content of each item and its outcome. The dated entries are specific requests; the continuous ones ran throughout. 4

3.1 Degrees of freedom of the bird, with the actuator and driver used for each, its location, and its state at the end of the work. The first four belong to the standard configuration; the fifth exists only in the female variant. 19

5.1 Control signals of the beak channel and the General-Purpose Input/Output (GPIO) lines assigned to them on the board. 37
5.2 Control signals of the tail channel and the GPIO lines assigned to them on the board. 39
5.3 Specification of the micro servo used for the bowing mechanism. . . . 41
5.4 Comparison of the three actuation technologies used on the robot, against the criteria that governed the choice in each case. 44

6.1 Principal components of the Canaribot V1.0 board, with the function of each. 48
6.2 Supply rails distributed on the board and the loads each serves. . . . 52
6.3 Mapping between the control signals of the TB6612FNG, the pins of connector J1, and the GPIO lines of the Raspberry Pi, as implemented on the board. 54
6.4 Correspondence between the contacts of connector J2, the conductors of the harness and their destinations inside the bird, established by continuity testing. Contacts are numbered as in the board design, starting from the end at which the green conductor sits. 58
6.5 Faults identified during the commissioning of the assembled boards, with the symptom observed, the cause where one was established, and the outcome. 60

7.1 Commands accepted by the robot over the WebSocket interface, with the argument each takes and its effect. 66

List of Acronyms

ADC Analog-to-Digital Converter 48

ALSA Advanced Linux Sound Architecture 19, 63, 68

ANR Agence Nationale de la Recherche 1, 77

CNRS Centre National de la Recherche Scientifique 2

DC Direct Current 47, 53

ENSEA École Nationale Supérieure de l'Électronique et de ses Applications 2

ETIS Équipes Traitement de l'Information et Systèmes 1–3, 13, 21, 25, 47, 68, 71, 77

GPIO General-Purpose Input/Output xi, 6, 21, 37, 39, 47, 54, 57, 59, 61, 63, 73

IDC Insulation-Displacement Connector 57

JSON JavaScript Object Notation 20, 66

LECD Laboratoire Éthologie Cognition Développement 1

OS Operating System 63

PLA Polylactic Acid 29, 32, 37

PWM Pulse-Width Modulation 19, 37, 41, 44, 53, 63, 67, 78

SPI Serial Peripheral Interface 52, 54, 63, 64

TPU Thermoplastic Polyurethane 30, 32, 34, 74

WAV Waveform Audio File Format 19

Chapter 1

Introduction

1.1 Scientific context

Songbirds are one of the few animal groups in which the young learn their vocalisations rather than producing them innately. A young bird has a genetic predisposition towards the sounds of its own species, but it has to hear an adult during a limited period of its development in order to sing normally as an adult [1]. This makes birdsong one of the closest available parallels to the way human infants acquire speech, and it is why it has become the standard model for studying how vocal learning works [2].

The work described in this thesis was carried out within the Agence Nationale de la Recherche (ANR) project nIrVAna[1] [3], coordinated by Sébastien Derégnaucourt at the Laboratoire Éthologie Cognition Développement (LECD) of Université Paris Nanterre, with Équipes Traitement de l'Information et Systèmes (ETIS) among its partners. The project studies the role of variability in vocal learning, following the song development of young zebra finches and young canaries exposed to song models of controlled variability, and using the results to build computational models in neurorobotics. Its stated aim reaches beyond birds: to develop tools for strategies that improve the quality of learning in birds and, to some extent, in human children.

Within this project, ETIS is responsible for the robots. The internship described here was advertised for the design and construction of biomimetic bird robots, with the stated objective of producing several units and improving the existing mechatronics, in terms of both robustness and performance. That is the work this thesis describes.

1.2 Why a robot

Studying how a young bird learns to sing means exposing it to a song model, and this is where the difficulty arises that the robot exists to solve.

A common approach is to play recordings, which gives complete control: the experimenter decides which song is heard, when, and how often. The problem is that it works poorly. Comparing three tutoring methods on the same protocol,

[1] ANR-23-CE37-0025.

Derégnaucourt et al. [4] found that passive playback produced poor imitation, that letting the bird trigger the playback itself produced better but highly variable results, and that housing a young bird with a live adult male produced close-to-perfect copies. Their conclusion was that social factors dominate the outcome.

So the method that works best offers the least control. A live tutor sings when it chooses and interacts with the pupil in ways the experimenter cannot specify or repeat, and two birds tutored by two adults have not received the same treatment in any controlled sense. Conversely, the method that offers full control produces the weakest learning.

A robot is an attempt to have both. It is a physical object in the cage that moves and can react to the young bird, so it provides something of what a live tutor provides; and it is a machine, so it delivers exactly the stimuli the experimental protocol specifies — which songs are played, at what moments, how often, and in response to what behaviour of the subject — identically for every subject.

The approach works. Araguas et al. [5], at Nanterre, tutored young zebra finches with a robot for five weeks and found no significant difference in song imitation between them and birds tutored by a live adult, and better results than any artificial method used previously. The Canaribot described in this thesis is a robot of this class, built for the canary.

1.3 Host laboratory and project organisation

The work described in this thesis was carried out at the ETIS laboratory, a joint research unit of CY Cergy Paris Université, École Nationale Supérieure de l'Électronique et de ses Applications (ENSEA) and the Centre National de la Recherche Scientifique (CNRS), located in Cergy, France.

The Canaribot is one of two variants of the same platform; the other, built around a zebra finch, is described in Section 2.2.

The platform is not the product of a single development effort but of a sequence of contributions accumulated over several years, by staff researchers, doctoral students and successive interns. The work described in this thesis was supervised by Prof. Alexandre Pitti. The V1.0 electronics were designed at ETIS by Fred de Melo. Much of the embedded C++ application is the work of Arnaud Blanchard, with subsequent contributions by Léo Coquet, whose doctoral work centres on the analysis of birdsong and who developed the client application described in Section 7.8. The software is maintained in the project repository [6].

Both the mechanical design of the bird and the embedded software carry the traces of these successive contributions. Alternative implementations of the same function coexist within the codebase, and modifications made to the hardware were not always recorded. This organisational context is relevant to the technical content of what follows: when a platform is developed by several people over several years, information about how it works tends to remain with the people who built each part rather than being written down.

Within that context, the work described in this thesis was mostly hands-on: producing units and repairing those that came back, revising the models the birds are printed from so that they go together as produced, and assembling and commis-

sioning the electronics. The new actuators described in Chapter 5 were developed alongside that, in the time it left, and a good part of the effort went into resolving problems of the kind just described as they arose. Section 1.6 states the contributions in more detail, and Chapter 8 assesses how far each was carried.

The Canaribot is developed at ETIS but used by the ethology laboratory of Université Paris Nanterre, which is the end user: they run the experiments with the birds, and the work at ETIS is carried out in response to their needs. In practice this took several forms. Nanterre would request a specific capability for an experiment they wanted to run, or ask for an additional unit, or return a unit that had broken and needed repair, and new experimental ideas came out of discussions between the two groups. In all cases, the deadlines were set by their experimental schedule rather than by the development work.

One consequence deserves mention, because it shapes several of the design decisions described later. The people who use the Canaribot are not engineers, so it has to be usable and, to some extent, repairable by people whose expertise is in ethology rather than in electronics or mechanics. This is part of the reason for the emphasis on documentation and on serviceability throughout this thesis.

1.4 Requested deliverables

The scope of the internship was set out in the advertisement for it. A first working version of the mechanical and electronic components already existed, including the body design, the pressure sensors, the stepper motors, a microphone, a loudspeaker, a Raspberry Pi and a dedicated board, and the objective was to produce several units of these robots and to improve the existing mechatronics in terms of both robustness and performance.

What was available at ETIS at the start was an electronics box that worked and a set of models for the bird; the units built from those models earlier were in use at Nanterre. The first weeks therefore went into understanding what was there — the components, their interconnection, the embedded software — and into producing a first bird locally, which turned out to require correcting the printed parts by hand before they would go together at all. That work is described in Section 4.3.

Within that scope, the work was driven by requests from Nanterre, which were mostly specific and short-term: a unit needed for a particular experiment, an additional unit, or the repair of one that had broken. These requests carry deadlines that the development work does not control. Most experiments require birds within a specific age range, from about 30 to about 100 days old, since this is the period in which song learning takes place. The experimental schedule therefore follows the availability of birds of the right age, and a unit delivered after that window has closed is of no use for that experiment, however well it works. Table 1.1 lists the requests received and their outcome.

The dated entries are the ones with deadlines attached, and they are the ones against which the work was formally measured. The continuous entries account for a comparable share of the time: units were supplied and repaired as the laboratory needed them, and the mechanisms described in Chapter 5 were arrived at through successive printed prototypes rather than designed once and built. The outcome of

Table 1.1: Work carried out in response to the needs of Nanterre, with the technical content of each item and its outcome. The dated entries are specific requests; the continuous ones ran throughout.

Period	**Request**	**Technical content**	**Outcome**
April	One fully working bird unit	Body and head rotation, audio playback, instrumented perches	Delivered
May	Paired-box configuration for a comparison experiment	One complete unit and one audio-only unit, driven from the same host	Delivered; not yet used
September	One additional box, three canary units and three zebra finch units	Complete units of two species	Two canary units delivered; zebra finch units not undertaken
Continuous	Additional units as they were needed	Units assembled to the current geometry and supplied on request	Carried out throughout
Continuous	Repair of units in service	Returned units diagnosed and repaired	Carried out throughout
Continuous	Development of the beak and tail mechanisms	Successive printed prototypes of the head, neck and tail, tested and revised	Mechanisms built and verified in place

each is assessed in Chapter 8.

1.5 Objectives

Alongside the deliverables and repairs described above, a set of development objectives was pursued throughout the internship. These are the upgrades and new capabilities added to the system, as opposed to the production and maintenance of units based on the existing design. Some were requested directly by Nanterre, others came out of discussions between the two groups about what the robot could usefully do.

1. Improve the mechanical design of the bird so that it can be assembled from the printed parts as produced, without the manual adjustment the inherited models required, and so that it stands up better to use.

2. Design and integrate the beak and tail actuators, adding the visual components that accompany vocalisation and postural signalling in a live bird.

3. Produce a batch of units built around the revised board, which supports the beak and tail actuators where the earlier version did not. The aim was not to meet a particular request but to renew the equipment available to Nanterre and to increase the number of units, so that experiments could be run in parallel rather than one at a time.

4. Improve the reliability of units in service, in particular the fixation of the head, which was the most frequent cause of units being returned for repair.

5. Document the system so that it can be maintained and extended by later contributors without relying on knowledge held by individuals.

The first three are the main development work. The last two were not planned at the start: they emerged from the repair activity and from the difficulties encountered when bringing new units into service.

A sixth objective was added partway through, in circumstances worth recording because they bear on how far it was taken. Production was repeatedly held up by materials that had to be ordered and had not yet arrived, and during one such wait the question arose of what could usefully be done in the meantime. The result was the female variant described in Section 5.4: a bird capable of the bowing posture, which would allow experiments in which the robot provides female feedback to a live male rather than only presenting to him. It was taken as far as a validated kinematic arrangement and a tested actuator, and no further, since the materials arrived and the production and actuator work resumed.

1.6 Contributions

The work resulted in the following contributions to the platform.

On the **mechanical** side, the inherited bird models were revised in Blender and SolidWorks so that the printed parts assemble as produced, without the manual correction previously required, and the fixation of the head was redesigned to withstand the pecking the robots receive in use. The revised geometry also provides the interfaces required by the new actuators, and a split body was developed for the female variant.

Two **actuators** were developed and built: an electromagnetic beak mechanism, characterised by measurement and sized against the available drive circuit, and a tail transmission selected from three candidate arrangements. A servo-driven bowing mechanism was designed for the female variant and its actuator validated on the bench.

On the **electronics** side, a batch of revised boards was assembled, and the completed boards were commissioned, and two interfaces that had not previously been documented were reconstructed by measurement: the mapping between the driver control signals and the host GPIO lines, obtained from the fabrication data of the board, and the pinout of the harness between the box and the bird, established by continuity testing. Seven faults were identified and classified in the course of this work, four of them with their cause established.

On the **software** side, the tail actuation subsystem was implemented, and two defects affecting the beak control path were found and corrected.

Alongside these developments, units were produced and repaired throughout in response to the requests listed in Table 1.1, and the paired-box comparison apparatus was built and verified.

The extent to which each of these contributions was validated is assessed in Chapter 8.

1.7 Outline of the thesis

Chapter 2 sets out what is known about how songbirds learn their song and derives from it the requirements that shape a robot intended to act as a tutor.

Chapter 3 describes the Canaribot as a system: the experimental scenario it supports, its two-unit architecture, its degrees of freedom, and its sensing, audio, and control subsystems, together with the state of the platform at the beginning of the internship.

Chapter 4 presents the mechanical work: the constraints the design must satisfy, the geometry inherited at the beginning, the revision carried out region by region, the additive manufacturing parameters, and the problems that remain open.

Chapter 5 describes the three actuators developed during this work.

Chapter 6 describes the electronic system and the commissioning of the assembled boards, including the faults found and how they were diagnosed.

Chapter 7 describes the embedded software and the client application, and the changes made to them.

Chapter 8 assesses the work against the objectives and states its limitations, and Chapter 9 concludes and proposes further work.

Chapter 2

Background

A robot built to act as a song tutor has to satisfy requirements that come from the biology of the animal it addresses rather than from any general engineering specification. This chapter sets out those requirements and where they come from, describes the zebra finch variant of the same platform on which the present work builds, and states what follows for the design. The engineering consequences are the subject of the rest of the thesis.

More broadly, the Canaribot belongs to a class of interactive biomimetic robots used as controlled social stimuli in animal-behaviour research. Such systems have been used with several species to replace or complement a living interaction partner while retaining control over the behaviour presented to the subject [7, 8]. The value of the approach lies in that combination: the stimulus remains physically present and able to respond to the animal, while what it does can be specified and repeated by the experimenter.

2.1 What the robot has to do, and why

The requirements that shape the Canaribot come from a small number of findings about how song learning works. They are stated briefly here; the engineering consequences are what the rest of this thesis is about.

The first is that learning depends on a model heard during a limited period. A young songbird deprived of a model does not sing normally as an adult, and species differ in when they are able to learn: some are limited to a sensitive period during development and can barely modify their song afterwards, while others, described as *open-ended learners*, retain the ability as adults and experience a reopening of the sensitive period each year. The two species used in the nIrVAna project fall on opposite sides of this division. The canary is an open-ended learner: its song becomes plastic again each autumn and recrystallises in spring, so that the repertoire changes between breeding seasons [1]. The zebra finch is not: its song crystallises at around 90 days after hatching and remains fixed for the rest of its life [4]. This is the origin of the age window given in Section 1.4, and therefore of the deadlines that governed the work described here.

The second is that how the model is delivered matters as much as what is delivered, which is the finding on which the whole platform rests and which is set out in

Section 1.2. What should be added is that both visual and vocal interaction with the tutor contribute to the accuracy of learning [4]: it is not enough for the young bird to hear the model, it also matters that it can see the tutor producing it. A robot that only played sound would therefore miss an important part of what makes a live tutor effective.

The third is that more is not better. Exposure to a song model does not improve imitation indefinitely: in an operant setting, forty playbacks of the song motif per day are sufficient for significant learning, and greater exposure produces *less* complete imitation [4]. A robot tutor therefore has to limit what it delivers rather than merely deliver it, which is the reason for the refractory interval described in Section 3.1.

One qualification should be recorded alongside these, in the interest of accuracy. The canary is among the species reported to learn readily from loudspeaker playback, unlike the zebra finch [4]. The case for a robot is therefore not identical for the two species. Whether a moving robot outperforms playback for canaries has not been established, and the apparatus described in Section 3.8 was built to establish it.

2.2 The zebra finch variant

The Canaribot is one of two variants of the same platform. The other is built around a zebra finch rather than a canary, and is described by Araguas et al. [5] under the name MANDABOT.

The two robots are closely comparable. As described by Araguas et al., MANDABOT rotates its body about a vertical axis and its head independently, opens its beak and moves its tail, and carries a loudspeaker inside the body. The head is driven by a stepper motor controlled by a TMC2130, and the beak and tail by coils and magnets driven by a TB6612FNG — the same devices that appear on the Canaribot board described in Chapter 6. Two perches on either side of the robot rest on load cells. The behaviour is the same one described in Section 3.1: when the bird lands on a perch the robot turns towards it and either moves its head or plays a song, and returns to a neutral position when the bird leaves.

What that paper established is the premise on which the work in this thesis rests. Twelve young zebra finches were tutored by the robot and twelve by a live adult male, one hour a day, five days a week, for five weeks. There was no significant difference between the two groups in how well they imitated the song model, and the robot group did better than the artificial methods used previously — passive playback, operant playback, and video. A robot of this kind is therefore accepted by a young bird as a valid song tutor.

Two of its findings bear directly on the chapters that follow. The young birds pecked the robot, and the pecks were counted: they increased over the first two weeks and then declined, and roughly 57% of them were on the body rather than on the head and beak. The failure mode described in Section 4.3.2 is therefore not particular to the Canaribot but a known property of this class of instrument. And the paper closes by listing the questions its own results leave open: whether a motionless robot would work as well, whether the beak needs to open in time with the sound, and whether a robot painted as a female could be used in a courtship context.

Those three questions correspond, respectively, to the paired-box configuration of Section 3.8, the beak actuator of Section 5.2, and the female variant of Section 5.4.

2.3 What follows for the design

Four requirements recur throughout this thesis, and it is worth stating them together before the chapters that respond to them.

The robot has to be **physically present and accepted as a bird**, which is what distinguishes it from playback and from video. In animal–robot interaction studies, morphological features, colour, shape and motion patterns have all been shown to affect whether living animals accept a robotic replica [9]. This is why the external appearance is treated in Chapter 4 as a functional specification rather than a matter of finish, and why nothing may protrude from the body of the robot.

It has to **move in time with the sound**, since what the subject sees should correspond to what it hears. This is why the beak is driven from the playback path in the firmware rather than being commanded separately, as described in Section 7.6.

It has to **react to the subject**, since this is what distinguishes a robot tutor from an animated loudspeaker. Hence the instrumented perches and the microphone described in Section 3.4.

Finally, it has to be **controllable and repeatable**, which is the point of using a machine rather than a live bird. Since the laboratory needs several units running in parallel, this means repeatable across units as well as across sessions. This extends to manufacture: additive manufacturing gives a direct route from digital geometry to repeated production, but the design has to account for the constraints and variability of the process rather than treating the printed part as an exact reproduction of the nominal geometry [10]. This need for repeatability shaped much of the work described in this thesis.

Chapter 3

The Canaribot System

This chapter describes the Canaribot as a system. It begins with the experimental session the robot is built to support, since the architecture and the design constraints of the chapters that follow are all responses to what that session requires, and then works through the physical architecture, the degrees of freedom, and the sensing, audio and control subsystems. It closes by setting out what already existed at the beginning of the internship, and by describing a comparison setup built during it. Figure 3.1 shows a complete unit as it appears in use.

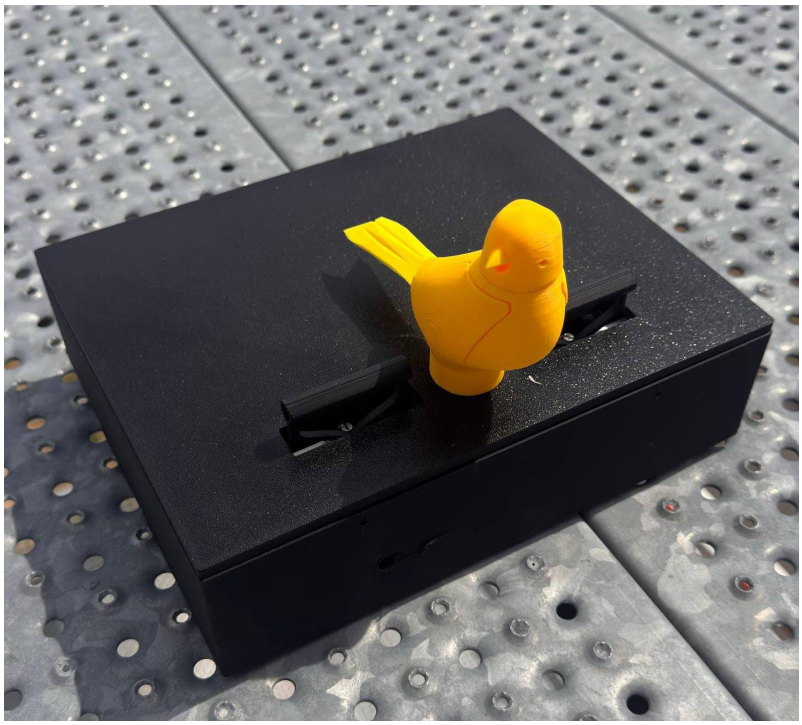


Figure 3.1: A complete Canaribot unit: the bird mounted on the lid of the electronics box, with the two instrumented perches on either side of it.

3.1 The experimental scenario

The subject of an experiment is a young canary that has not yet learned to sing. It is housed in an enclosure isolated from outside stimuli, so that the only song it hears is the one the robot produces, and the whole session is filmed so that its behaviour can be analysed afterwards. The Canaribot is placed inside the enclosure with it. Figure 3.2 shows the arrangement.

The electronics are housed in a box closed by a lid, and both the robot and the two perches are mounted on that lid: the bird sits at the centre with a perch on either side of it, each resting on a load cell. When idle, the robot stays still in a neutral position while the control software monitors the perches and the microphone.

An interaction begins when the subject lands on one of the perches. The load cell registers the added weight and the software detects the event once the measured value passes a fixed threshold. Only the transition from empty to occupied is treated as an event, so a bird that remains on the perch does not trigger the robot repeatedly.

Once the event is detected, the robot turns towards the bird: the body rotates about its vertical axis to face the occupied perch, and the head can be rotated separately to refine the orientation. The robot then plays a recorded canary call or song, selected by the experimenter beforehand. In the complete behaviour the beak opens together with the vocalisation and the tail moves at the same time, so that what the subject sees matches what it hears, as it would with a live tutor.

After a stimulus has been delivered the system waits before allowing another one, an interval of three seconds enforced in the command interpreter. This prevents the robot from singing continuously if the bird lands repeatedly in a short time, and it is not merely a practical precaution: as noted in Section 2.1, more exposure to a song model does not produce better imitation, and beyond a certain quantity it produces worse.

Over the course of the experiment the subject gradually begins to produce song, imitating what it has heard. The robot is therefore the tutor from which the bird learns, and the purpose of the setup is to observe how that learning develops.

Alongside the perch-triggered behaviour, the system also listens. A microphone in the electronics box feeds an audio processing thread that computes the dominant frequency of the incoming signal and applies a threshold and a temporal rule to decide whether the bird has vocalised, as described in Section 7.7. When it has, the event is reported and logged, and it can be used to trigger a response from the robot — which allows experiments in which the robot reacts to the song of the subject and not only to where it is standing.

Every event, whether a perch trigger, a detected vocalisation, a command issued by the experimenter or a stimulus delivered, is timestamped and recorded, so that the full history of a session is available for later analysis.

The configuration described above is the standard one, but it also serves as a basis for variations. The appearance of the robot can be changed to make it a less realistic tutor, for instance by printing it in a colour other than the natural yellow of the canary. The set of songs presented can be varied. The robot can be made to move or to remain still, which is the comparison described in Section 3.8. And since the bird and the box are separate units, a subject of one species can be presented

with the robot of another, as in Figure 3.3b: the appearance of the tutor and the song it delivers can then be varied independently, which allows the relative weight of the two to be examined. In each case the rest of the setup is unchanged, so that the effect of the modified element can be observed on its own.

This tutoring paradigm is the main use of the system and the one for which it has been used so far. Other directions have been considered at ETIS, including experiments in which several robots interact with one another rather than with a live bird. These have not been carried out, but they are part of the reason why the system is built around a general command protocol and a scriptable client rather than around a single fixed behaviour: new experimental designs should be implementable without modifying the robot itself.

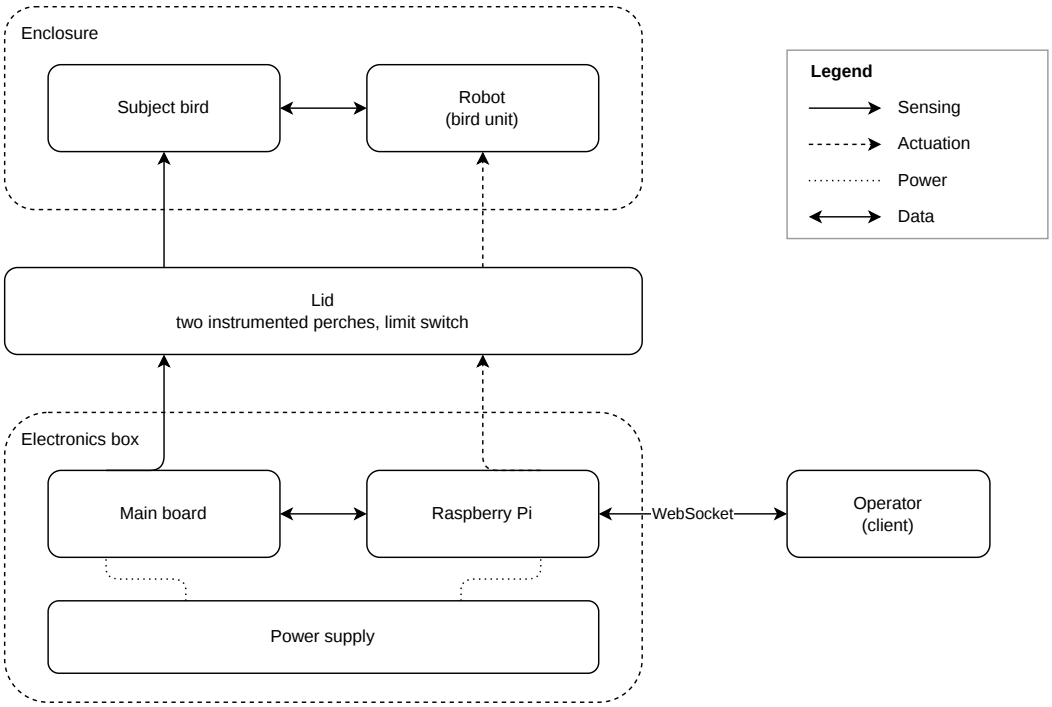
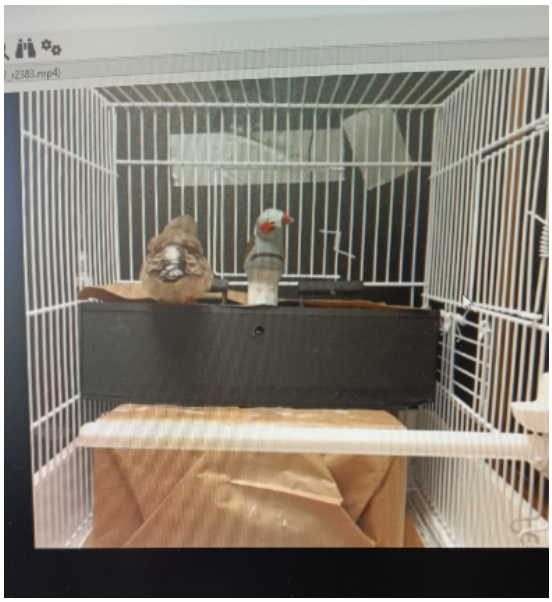


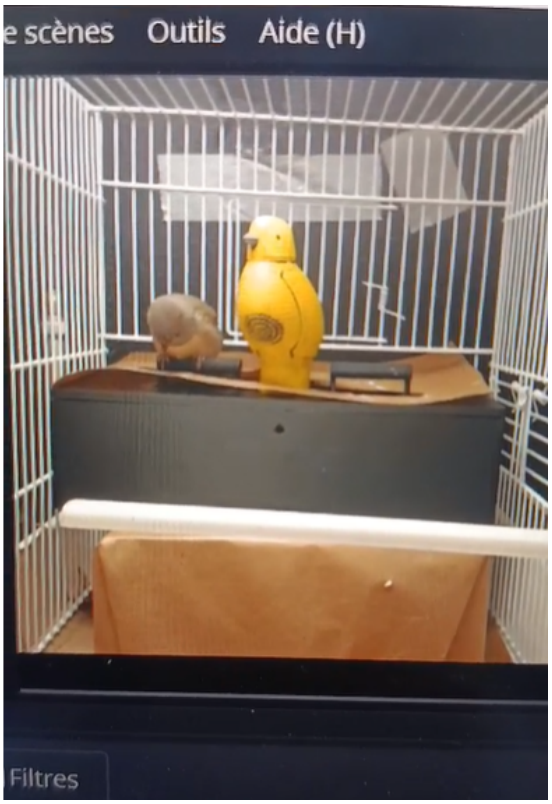
Figure 3.2: System-level block diagram. The enclosure holds the subject bird and the robot, which is mounted on the lid of the electronics box together with the two instrumented perches and the limit switch. Inside the box are the Raspberry Pi, the main board and the power supply. Sensing paths run from the perches and the microphone into the box, actuation paths run from the board to the robot, and a WebSocket link connects the Raspberry Pi to the laptop running the client application.

3.2 Physical architecture

The system is divided into two units: the *bird*, which is the visible robot, and the *electronics box*, which sits beneath it and contains the computer, the power supply



(a) A zebra finch with the robot of its own species.



(b) A zebra finch with the canary robot.

Figure 3.3: Two experimental sessions, seen from the monitoring station. Because the bird and the electronics box are separate units, the robot presented to a subject need not be of the subject's own species, which allows the appearance of the tutor and the song it delivers to be varied independently of one another.

and most of the electronics. The bird is mounted on the lid of the box, so the two are physically joined but functionally distinct. Figure 3.6 shows how they fit together.

Several considerations led to this separation.

The most restrictive is **volume**. The bird has to keep the outline of a real canary, which leaves very little space inside it, so anything that does not need to be there is placed in the box instead. Only the head motor, the loudspeaker, the beak and tail actuators and their wiring remain inside the bird.

Mass matters for the same reason. The bird is carried and rotated by a shaft driven by a stepper motor in the box, so any mass added to it increases the inertia that motor has to move; and the motors that drive the head and the tail are necessarily small, since they have to fit inside the body, so they can only move those parts if the parts are light. Every component of the bird is therefore kept as light as the structure allows.

Acoustic noise is a concern specific to this application. The experiment is about vocal behaviour, and any sound the apparatus makes that is not part of the intended stimulus is a possible source of interference. Keeping the power supply, the motor drivers and the body motor inside a closed box reduces the noise they produce and separates it from the microphone and from the bird. The same concern determined the choice of motor driver, as described in Section 6.3.

Safety follows from the presence of a live animal moving freely around the apparatus: exposed electronics and moving parts would be a hazard for the bird and a source of damage and contamination for the equipment.

Finally, the arrangement makes **assembly, replacement and maintenance** straightforward. The bird is fitted by pushing it onto the drive shaft, after which the lid is closed, and removing it is equally quick. A damaged bird can therefore be replaced, or one bird exchanged for another — a different individual, a different colour, or a different species — without touching the electronics. Given how often units are returned for repair, and given that the same box serves both canary and zebra finch units, this is a practical requirement rather than a convenience. The box, in turn, can be opened and serviced without disturbing the bird, and a faulty box can be swapped for another.

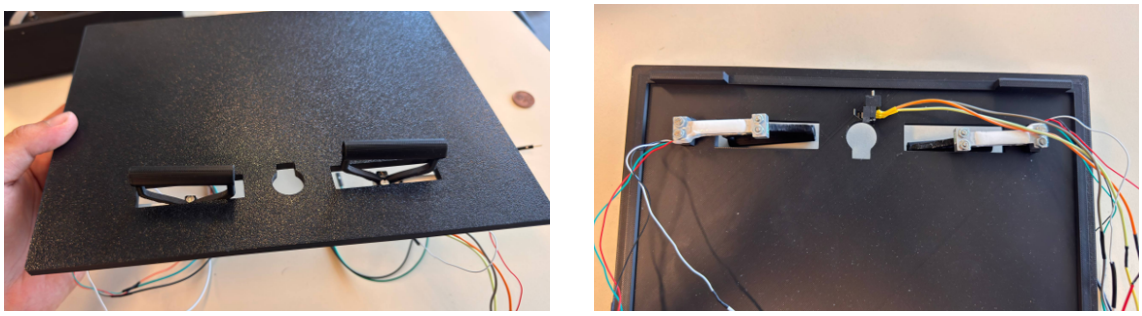
Only the body of the bird is visible above the lid, as Figure 3.1 shows. Below it pass the drive shaft that rotates the bird and the wiring serving the head motor, the loudspeaker and the actuators, which leaves the body through a small opening underneath; when the bird is mounted, that opening sits inside the box and is hidden from view. These conductors connect to a ribbon harness which in turn connects to the main board, and the harness ends in individual connectors so that a bird can be connected or disconnected quickly.

The load cells, the limit switch used to home the body axis, and the microphone are mounted on the lid rather than on the bird, and they too reach the board through connectors rather than being permanently wired to it. The lid is therefore removable: opening the box means unplugging the sensor connectors and lifting it off, and closing it means plugging them back in. This is what allows the electronics to be serviced without disturbing the bird, and a lid to be exchanged between boxes.

The size of the box is set by two things: it has to hold all the electronics, and it has to fit on the print bed of the printer used to produce it.



Figure 3.4: Interior of the electronics box, showing the Raspberry Pi, the main board, the body motor, the USB sound card and the various connecting cables.



(a) (b)

Figure 3.5: Lid of the box. (a) Seen from above: the two perches on either side of the central opening through which the drive shaft and the wiring pass. (b) Seen from below: the load cells supporting the perches, and the limit switch against which the locking block of the bird comes to rest.

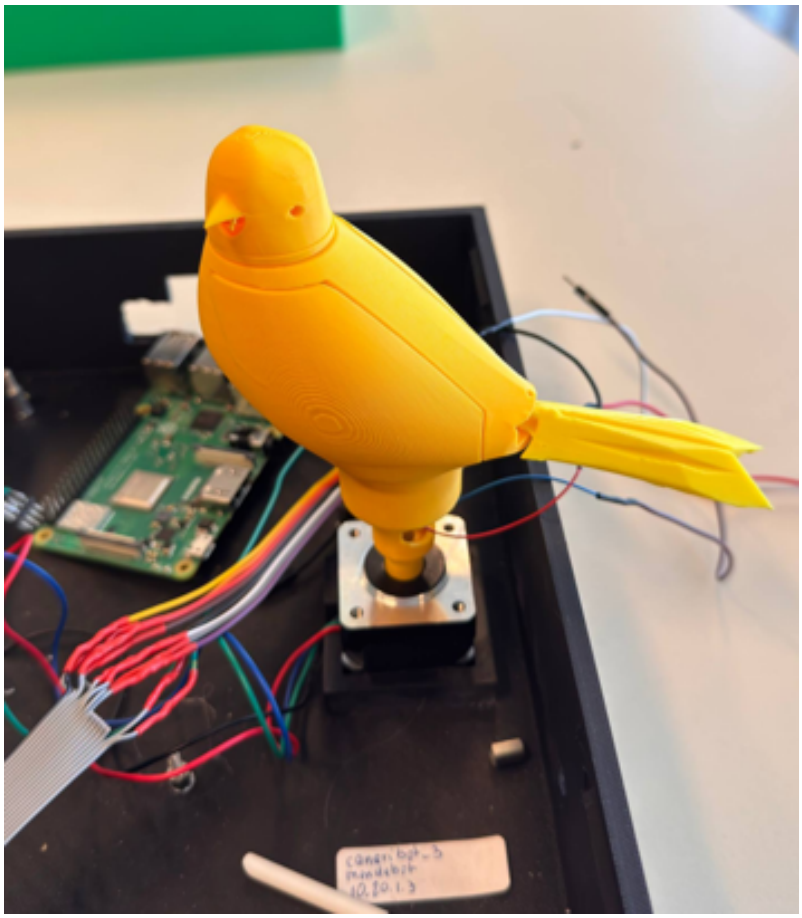


Figure 3.6: Cross-section of the two-unit architecture.

3.3 Degrees of freedom of the bird

The bird has four degrees of freedom, listed in Table 3.1, and a fifth in the female variant. Two of the four were already present at the start of the internship; the other two, and the fifth, were developed during this work and are described in Chapter 5.

The **body** rotates about a vertical axis, to turn towards the perch on which the subject has landed. It is driven by a stepper motor in the box — an SY35ST28-0504A, rated 0.5 A with a holding torque of 1 kg cm [11] — whose shaft passes through the lid and carries the bird directly. The travel is limited rather than continuous, since the wiring running into the bird would otherwise be twisted, and a homing procedure at start-up establishes the reference position.

The **head** rotates about its own vertical axis, independently of the body, giving a finer and faster orientation cue. It is driven by a compact 8 mm two-phase, four-wire stepper motor, mounted on a printed holder inside the body, with the head fixed to its shaft.

The **beak** opens and closes in time with the vocalisation. A coil in the head attracts a magnet embedded in the beak, and the beak returns to the closed position by the elasticity of the structure when the coil is de-energised.

The **tail** moves relative to the body, adding a further visual cue alongside the vocalisation. A coil drives a lever in a straight line, and a transmission converts this into rotation of the tail.

In the female variant, the upper **body** tilts forward about a horizontal axis, reproducing the bowing posture of the female canary. Unlike the beak and the tail, this posture has to be held rather than performed as a brief gesture, which calls for a servo rather than a coil. The split body about which the movement takes place was built; the transmission driving it was specified but not implemented, as described in Section 5.4.

Three different actuator technologies are therefore used on the same robot. Steppers are fitted on the two orientation axes, which need precise and quiet positioning; coils drive the beak and the tail, which need fast, brief movements in very little space; and a servo is used for the bowing posture, which has to be held. The reasons are developed in Section 5.5.

3.4 Sensor system setup

The system senses the subject in two ways: the perches detect where the bird is, and the microphone detects when it vocalises. A third sensor, the limit switch, concerns the machine rather than the animal.

The perches are mounted on the lid, one on either side of the robot, each resting on a load cell. The load cells are read by two HX711 modules, which amplify the signal and convert it to a 24-bit digital value. A tare is performed at start-up to remove the standing offset, so the perches must be empty when the system is switched on. A landing is detected by applying a fixed threshold to the tared value: what matters is whether a bird is present and on which side, not how much it weighs.

The limit switch is a microswitch on the lid that gives the body axis a mechanical reference. A stepper motor driven in open loop has no absolute position

Table 3.1: Degrees of freedom of the bird, with the actuator and driver used for each, its location, and its state at the end of the work. The first four belong to the standard configuration; the fifth exists only in the female variant.

Axis	Actuator	Driver	Location	Status
Body yaw	SY35ST28-0504A stepper motor	TMC2130 (U4)	Box	Inherited, working
Head yaw	8 mm two-phase, four-wire micro stepper	TMC2130 (U3)	Bird	Inherited, working
Beak gape	Coil and magnet, 69 Ω	TB6612FNG ch. A	Bird, head	Built and verified in place; not driven from the box
Tail	Coil and lever	TB6612FNG ch. B	Bird	Built and verified in place; not driven from the box
Body pitch	GWS PICO BB micro servo	Hardware PWM	Bird, base	Split body built; actuation proposed but not implemented

feedback, so this reference has to be re-established each time the system is powered up. The homing procedure drives the body until the switch is triggered, then returns by a fixed number of steps to the neutral position.

The microphone, connected through the USB sound card, allows the system to detect the subject's vocalisations. The signal is captured through Advanced Linux Sound Architecture (ALSA) and analysed in a dedicated thread, described in Section 7.7. This is what allows the robot to respond to what the bird does, and not only to where it is.

3.5 Audio subsystem

The acoustic stimulus is the main output of the system, so the audio chain is the subsystem whose quality matters most for the experiment. Its implementation on the board is described in Section 6.6; what concerns the experiment is described here.

Playback is performed by the Raspberry Pi from a library of pre-recorded Waveform Audio File Format (WAV) files, organised by individual bird and by stimulus type, with separate directories for songs and for calls. The audio is sent to an external USB sound card rather than to the audio output integrated in the Raspberry Pi, whose analogue output is derived from a Pulse-Width Modulation (PWM) signal and is not of sufficient quality for reproducing birdsong. The same USB device also provides the microphone input. From there, the signal reaches an amplifier on the

main board and, through the harness, a loudspeaker housed inside the body of the bird.

The choice of loudspeaker was made from those available in the laboratory and is constrained from both sides. It has to be small enough to fit inside the body alongside the head motor and the wiring, and at the same time loud enough to reproduce a canary call at a realistic level. Impedance also matters, since it determines the power the amplifier delivers. Several of the available units proved too weak once installed, so we selected them by testing them in place rather than relying on specifications alone.

3.6 Control and communication

The robot is controlled by a Raspberry Pi 3B+ [12] running a C++ application, which handles the hardware, the behaviour of the robot and the network connection. The experimenter does not interact with the robot directly but through a web client running in a browser on a separate machine.

This arrangement follows from how the experiments are run. During a session, the enclosure has to be left undisturbed, since approaching it or handling the apparatus would interfere with the behaviour being observed. A network interface allows the experimenter to control the robot and follow what is happening from another room while the session continues. It also separates the user interface from the software running on the robot: the two communicate only through the message protocol, so the client can be changed or replaced as the experimental protocols evolve without touching the embedded code, and a new capability added to the robot becomes available to the client as soon as it has a command.

The application runs a WebSocket server [13] on port 8080. A WebSocket connection is persistent and bidirectional, which is what this system needs: the client sends commands to the robot, and the robot reports events — a perch being occupied, a vocalisation detected — as they happen, without being asked. Only one client is allowed to connect at a time, so that a session cannot receive commands from two operators at once. Messages are JavaScript Object Notation (JSON) objects in which the key is the action and the value its argument, covering movement, stimulus selection and playback, and session management. The software is described in detail in Chapter 7, and the full command set in Section 7.4.

3.7 Starting point

Because this work was carried out on a system that had been under development for several years and by several people, it is worth stating clearly what already existed at the beginning of the internship and what was developed during it. This distinction is used throughout the thesis, and the assessment of the objectives in Chapter 8 refers back to it.

Everything needed to operate a robot already existed. The electronics box was built and functioning, with its board, its power supply and its motor drivers. The embedded software handled the two rotation axes, with the homing procedure and

the discrete positioning used to turn the robot towards a perch; the acquisition from the load cells, with tare and threshold detection; the playback of the stimuli; the microphone acquisition and vocalisation detection; and the WebSocket server and command interpreter. The client application, with its manual controls, its experiment planner and its logging, was in use. Part of the initial work therefore consisted simply in understanding how these components worked and how they fitted together, since much of this was not documented and the people who had written it were not always available.

The beak and the tail had been planned but not realised. The beak had a firmware implementation and a coil could be fitted in the head, but the electronics of the existing box did not allow either actuator to be driven, so bringing them into service required the revised board described in Chapter 6. The tail had nothing on the robot itself: no mechanism, and no firmware — no GPIO assignment, no initialisation, no control function and no entry in the command table. The client, however, already provided the control for it, which is a further sign that the capability was planned rather than improvised. The female variant did not exist in any form.

What was missing above all, however, was an assembled bird to work on. Models existed in Blender and units had been built from them earlier, but those units were in use at Nanterre, and the models themselves were imprecise: the printed parts did not fit together properly, so every unit had to be adjusted by hand during assembly, and the birds in service broke easily. Since the laboratory needed several identical units rather than one, this was the principal obstacle, because a model that requires manual correction each time it is printed cannot be produced in numbers.

The first task was therefore to produce a bird locally that was genuinely usable, in the circumstances described in Section 1.4: a loudspeaker, a head motor and a body motor, and nothing more, but assembled correctly from the printed parts and more robust than the earlier ones. Adding the beak and tail actuators became possible only once this was in place, since those mechanisms have to fit inside a body whose dimensions are known and repeatable. That work is described in Chapter 4.

3.8 The paired-box configuration

An early request, listed in Section 1.4, was for a setup allowing a comparison between two conditions: a robot that moves and sings, and a robot that only sings. The question matters for the project as a whole. If a subject responds in the same way to both, the movement of the robot adds nothing for that paradigm and a loudspeaker would be enough; if the responses differ, the movement is an active part of the stimulus, and the effort spent on the mechanical behaviour of the robot is justified.

The configuration was requested by Nanterre — two units presenting the same stimulus, one moving and one not — and how to realise it was decided at ETIS. The first is a complete unit, with its full electronics box. The second is a bird carrying a loudspeaker only, with no motors, in a box containing nothing but the audio amplifier and its supply.

The audio source is shared between them, and shared upstream of any amplification: the line output of the sound card is split by a jack splitter, one branch entering

the board of the complete unit and the other running to the second box through a two-metre jack cable. Each amplifier then drives its own loudspeaker from the same input signal. The model of the electronics box was modified to add an opening for the cable, and the two units were assembled and tested.

Splitting the source at that point means the two conditions present the same stimulus, generated once and at the same moment, rather than two similar ones. The two reproduction chains then have to be equivalent as well, which is why the loudspeakers were selected by testing them in place: a difference in level between the two conditions would itself have been a variable in an experiment meant to isolate the effect of movement. It is worth noting too that the reduced unit is a bird containing a loudspeaker and not a bare loudspeaker, so what differs between the two conditions is the movement while the visual presence of a bird is the same in both.

The setup was built and delivered but has not been used so far. The work moved on to the extended version of the robot with beak and tail actuation, and the comparison was left for a later date. It is described here because the apparatus remains available, and the question it addresses remains open.

Chapter 4

Mechanical Design of the Bird Unit

The bird is the part of the system the subject actually sees, and the part that had to be rebuilt almost entirely during this work. This chapter sets out the constraints the design has to satisfy, describes the geometry inherited at the start of the internship, and then works through the revision carried out on it, region by region. It closes with the manufacturing parameters and with the two problems that remain open.

4.1 Design constraints

Every design decision described later in this chapter is a response to one or more of the constraints set out here. Some come from the experimental context, others from the division of the system into a bird and a box described in Section 3.2.

The most restrictive of them is the external appearance. The robot has to be treated by a live canary as another bird, or at least as something bird-like enough to produce a social response, so the outer shape has to stay within the proportions of a canary and nothing may protrude from it in a way that is visually obvious. Everything else has to fit inside what this leaves, and what it leaves is very little. Into that volume have to go the head motor and its holder, the loudspeaker, the coil and magnet of the beak, the tail actuator, and the wiring connecting all of them to the box below. The head motor sits in the upper part of the body with its shaft protruding so that the head can be fitted onto it, the coil and magnet of the beak are housed in the head itself, and the wiring runs down through the body and leaves through an opening underneath, which sits inside the box once the bird is mounted. Space is scarce everywhere and every addition competes for it; the design of the female variant in Section 5.4 is dominated by this.

It follows that the bird cannot be a single printed body, and this is the architectural decision underlying everything else in the chapter. A closed shell would be simpler to print and stronger, but there would be no way to get the motor, the loudspeaker and the coils inside it, and no way to reach them afterwards. The bird is therefore made up of separate parts that assemble around its contents, as described in Section 4.2, and the mating features between them have to do two things at once: locate the parts accurately enough that the assembled bird holds the shape of a canary, and come apart and go back together repeatedly without damage. Every component inside the bird is reachable by removing parts in sequence, and none of

them requires breaking anything.

Mass is constrained for the reasons given in Section 3.2: the body is carried and rotated by a motor in the box, and the parts inside the bird are moved by actuators small enough to fit in it, so mass is kept as low as the structure allows.

The mounting interface imposes several dimensional constraints at once. The body is fitted directly onto the vertical shaft of the body motor, which passes through the lid of the box, an arrangement that is mechanically simple and free of backlash but demanding to dimension. The base of the body has to match the shaft it fits onto; its height has to be correct relative to the box, since the motor sits at a fixed level inside and only the part above the lid should be visible, with the opening for the wiring hidden below it; and the same region has to leave room for the conductors to pass from the body into the box. All of this occupies the lower part of the body, which is then unavailable for anything else. The wiring itself, running from the box through the base and up towards the head, must not obstruct the moving parts, must not be pinched during assembly, and must tolerate the rotation of the body — which is what limits the body axis to a bounded travel rather than continuous rotation.

Two further constraints come from how the bird is made and how it is kept working. All structural parts are 3D printed, so the geometry has to be printable: overhangs must either support themselves or be placed where support material can be removed without marking a visible surface, thin features must be thick enough to print reliably, and mating features must allow for the tolerances of the process. And the unit has to be serviceable in the field. Actuators fail, wires break, and the robot is maintained over years by successive students whose expertise is not in mechanics, so the assembly has to be one that can be taken apart and reassembled by someone who did not design it. This is consistent with design-for-additive-manufacturing approaches in which assembly and disassembly are treated explicitly as design considerations, and joining features are adapted to the constraints of the additive process rather than transferred directly from conventional manufacturing [14].

4.2 Baseline geometry

The bird is made up of eight printed parts, shown individually in Figure 4.1 and assembled in Figure 4.2: the body, the head, the beak, a disc acting as the neck, the two wings, the tail, a holder for the head motor, and a pin joining the wings.

The **body** is the main structural element. It carries the interface to the drive shaft at its base, houses the head motor holder at its upper end, and provides the internal space through which the wiring runs.

The **head motor** sits inside the body with its shaft protruding at the top. The **neck disc** fits onto this shaft and the **head** is mounted on the disc.

The **beak** is a separate part, inserted into the head through the mouth opening, with only its front portion visible from outside. It is not fixed rigidly but pivots about a wire passing through the head from one eye to the other, so that it rocks about this axis and its visible portion moves up and down. The portion inside the head carries a recess housing the magnet, so that the coil described in Section 5.2 can attract it and produce this motion.

The **wings** are joined by a pin passing through matching holes in each of them, inside the body, which holds them together and gives them a stable position relative to the body. The **tail** is fitted at the rear and carries no load, contributing only to the appearance of the bird.

4.3 Redesign of the bird

The models inherited at the beginning of the internship were dimensionally inaccurate, and the parts printed from them did not fit together or into the box as intended. There was no assembled bird at ETIS to start from either: units built from those models earlier were in use at Nanterre, so the first task was to produce one locally, and doing so showed how much the geometry needed.

That unit was needed quickly, which determined how it was made. Correcting a printed part properly means identifying what is wrong with it, changing the model, printing it again and checking the result, and repeating that until it fits — and each iteration costs hours of printing. With a working bird required in short order, the parts were instead corrected as they came off the printer: the wings were filed and sanded until they entered their mating features, the opening in the box was enlarged with a drill and a rotary tool so that the bird could rotate freely, and the recess for the head motor was opened out by hand where it left insufficient space. The box being used had no harness either, so one was made for it, following the arrangement already in use on the units at Nanterre and described in Section 6.7.

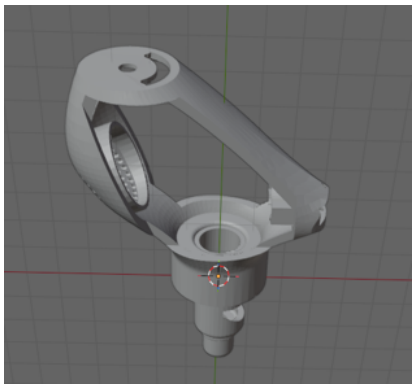
The result was rough but it worked, and it is what made everything described in this chapter possible. It is also the opposite of what the laboratory needed, since a unit that has to be adjusted by hand cannot be reproduced. Each defect corrected by hand on that first unit was therefore traced back to the model that produced it, and the geometry was revised so that the printed parts fit as produced; no manual adjustment is required on units built from the current models.

Each region of the bird raised its own problems, and the revision is described below in those terms.

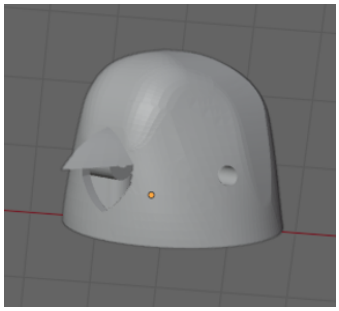
4.3.1 Base and interface to the body motor

The base was too large: it did not pass through the opening in the lid, and once forced through it did not seat on the shaft properly. The whole lower region had to be redimensioned before anything else could work, and correcting it meant revising several features together — the bore, the boss, the locking block, the opening in the lid and the position of the limit switch all have to agree for the bird to fit, to lock and to home correctly. The height of the base relative to the lid belongs to the same set, since the vertical position of the bird is fixed by where the motor sits inside the box.

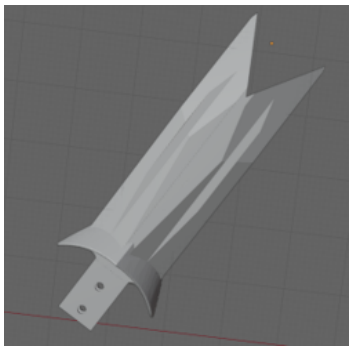
The bore that engages the shaft had a second problem, and one that came from how the part was produced rather than from its geometry. The shaft is a D-shaft, a cylinder with one side flattened so that its cross-section is a circle with a chord across it, and the bore was shaped to match, so that the two flats engage and the parts cannot rotate relative to one another — something a plain cylindrical fit cannot



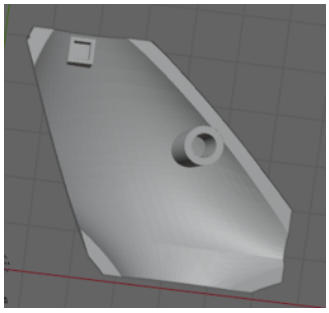
(a) Body



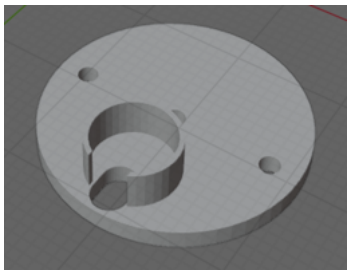
(b) Head



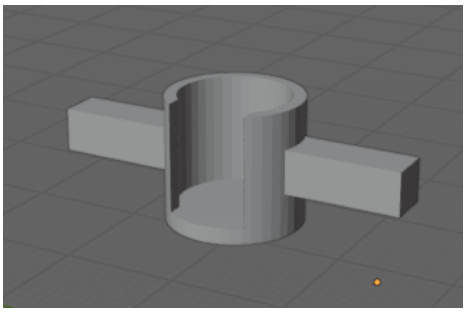
(c) Tail



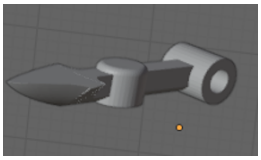
(d) Wing



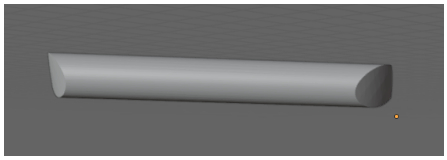
(e) Neck disc



(f) Head motor holder



(g) Beak



(h) Wing fixation pin

Figure 4.1: The parts of the bird, shown as CAD models. The body carries the interface to the drive shaft at its base and houses the head motor holder at its upper end; the neck disc fits onto the motor shaft and carries the head, into which the beak is inserted through the mouth opening; the two wings are held by the fixation pin passing through them inside the body; and the tail is fitted at the rear. The parts are not shown to the same scale.



Figure 4.2: The assembled bird.

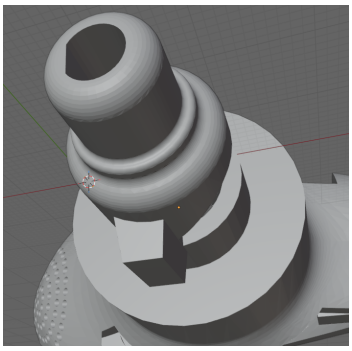
achieve without relying on friction alone. The feature existed in the inherited model but did not survive into the printed part. The bore was dimensioned too tightly, so it had to be opened out by hand before the shaft would enter, and since it prints with support material inside, clearing that material took the edge of the flat with it and left a bore closer to round than to the intended profile. Either way the engagement was lost and the shaft could turn without the bird following it. The bore was therefore redimensioned to receive the shaft as printed, and the orientation and support settings were reworked in the slicer so that clearing the support no longer damages the flat.

The base ends in a cylindrical boss with a rectangular block projecting from its side, visible in Figure 4.3, and the opening in the lid has a matching shape: a circular hole with a rectangular slot at one point of its circumference. The bird can therefore be inserted in one angular position only, and rotating it once through locks it in place. This bayonet fit means the bird cannot come off during operation while remaining removable in a few seconds when needed.

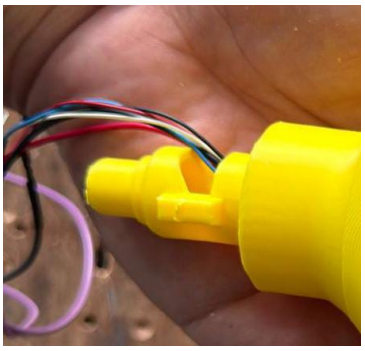
The same block serves a second purpose. As the body rotates below the lid, the block reaches the limit switch and provides the angular reference used by the homing procedure described in Section 3.4. Figure 4.4 shows this interface on an assembled unit, with the lid raised to expose the block and the switch. Alongside the boss is the opening through which the wiring leaves the body and enters the box.

4.3.2 Head and neck

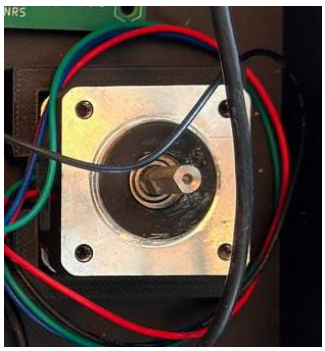
This is the most constrained region of the bird: within the shape of a canary head and neck it has to accommodate the head motor and its holder, the beak coil, and the wiring for both. Three changes were made to it, and they are described in the order in which they became necessary.



(a) The base seen from below in the CAD model.



(b) The same region on a printed unit, with the wiring leaving the body.



(c) The drive shaft of the body motor, seen from above.

Figure 4.3: The interface between the bird and the body motor. The base of the bird carries a D-shaped bore that engages the flat of the drive shaft, a rectangular block projecting from the side of the cylindrical boss, which locks the bird under the lid and reaches the limit switch as the body rotates, and an opening through which the wiring leaves the body and enters the box. The flattened profile of the shaft is what prevents the two from rotating relative to one another.

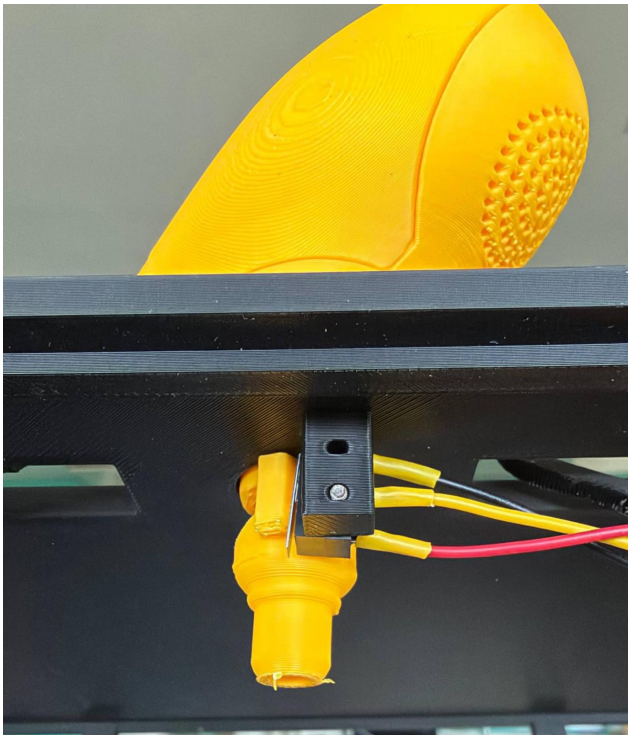


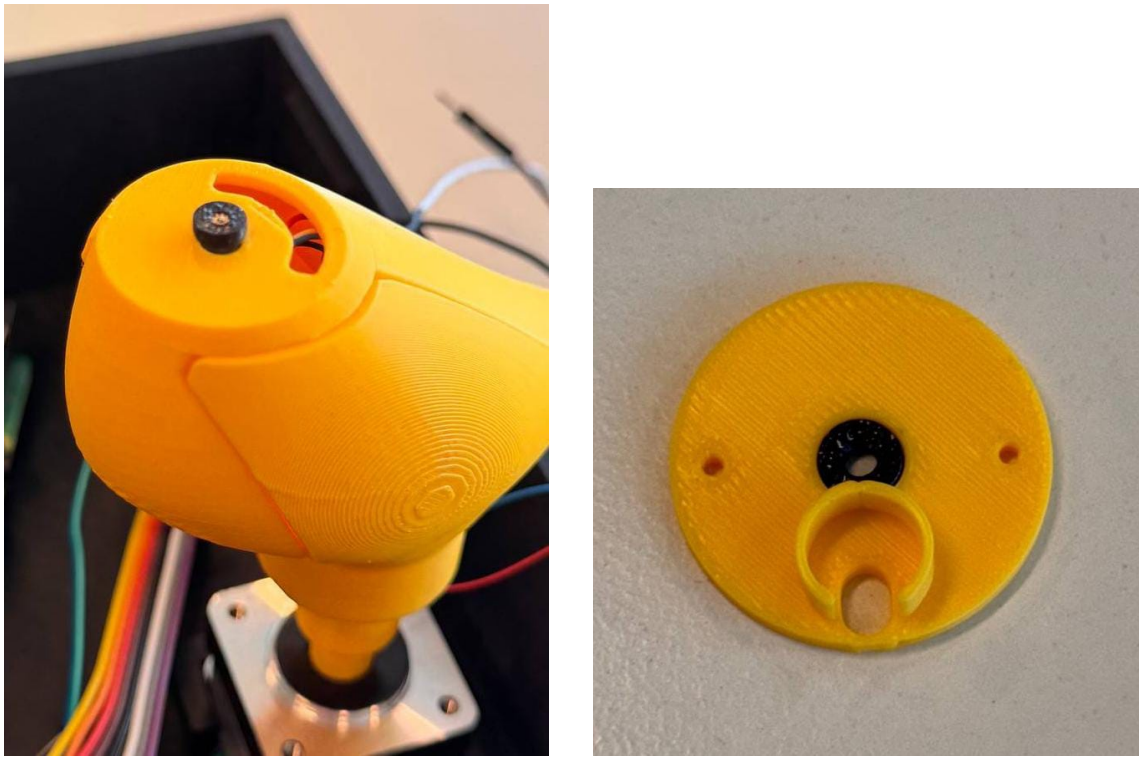
Figure 4.4: Bird mounted on the lid with the lid raised to expose the underside of the mounting interface. The rectangular block on the base is shown in contact with the limit switch at the homing position of the body axis.

The first concerned the shaft of the head motor, which did not protrude far enough from the body. This made the head difficult to fit and required the first units to be aligned by hand with a rotary tool; the model was revised so that the shaft protrudes further, and the head assembly now seats correctly as printed.

The second, and the most consequential, was the fixation of the head. The head is the part the subjects peck at, and its coming off the shaft was the most frequent reason for units being returned for repair. It had previously been bonded with adhesive, which works poorly here: adhesion to printed Polylactic Acid (PLA) is weak, because the material has a low surface energy and the printed surface offers little for a bond to key into, and the joint degrades under repeated impact, each strike extending the debond slightly with no visible change until the remaining bonded area gives way. Repairing it with the same method restores the original strength and no more, so the failure recurs.

The method adopted instead melts the plastic locally onto the shaft with a soldering iron, so that on cooling it forms a mechanical interlock rather than a surface bond. The joint then relies on the strength of the plastic itself rather than on adhesion to it, and does not depend on how the surface was prepared. This is not the best method in principle — an insert moulded into the part, or a mechanical clamp, would be better — but it was the most effective of those that could be carried out with the equipment available in the laboratory. It required the neck disc to be reshaped around the shaft, and on the first units, which had no moving beak, the holder for the beak coil was removed from the disc altogether so that the region

could be shaped for the joint alone. In the final days of the internship, discs printed in Thermoplastic Polyurethane (TPU) were added between the shaft and the neck disc: TPU is flexible, so the disc takes up small misalignments and absorbs part of an impact rather than transmitting all of it to the joint. Figure 4.5 shows the interface with the head removed.



(a) (b)

Figure 4.5: The head-to-body interface with the head removed. (a) The shaft of the head motor protruding from the upper part of the body, with the TPU disc fitted over it. (b) The neck disc, showing the TPU insert around the opening that receives the shaft.

The third change reinstated that holder, once the beak actuator had to be tested. It was reshaped to fit the coil selected for the purpose, positioned above the neck disc and facing the magnet housed in the beak, and given an opening for the coil leads. Those leads pass through an existing opening in the body, join the rest of the wiring inside it, and leave under the bird with everything else. The space available is very small, which limited the size of coil that could be used and was one of the constraints on the actuator design described in Section 5.2.

4.3.3 Tail

In the inherited design the tail was decorative: it was fitted at the rear, carried no load, and nothing moved it. Making it an actuated part took several iterations, each prompted by what the previous one did badly.

The first change was mass. The coil that drives the tail has very little force available, so anything the tail carries beyond what the structure requires comes directly out of the movement obtained. The part was reworked to be as light as it could be made while remaining printable.

The second was the engagement with the actuator, and this went through two arrangements. The first followed the slot-and-pin transmission described in Section 5.3: a hole was added through the body to carry a steel wire acting as the pin, with the corresponding feature on the tail. It was built and tried, but the simpler arrangement turned out to work as well, so it was set aside. What replaced it is a recess in the tail into which the lever of the coil seats directly, so that the movement of one carries the other with nothing in between. A holder for the coil was added inside the body at the same time, positioned so that the lever meets the recess when the tail is fitted.

The third came from watching the assembled mechanism work. The tail is a push fit into the body, and the two surfaces bore against each other over a broad area, so the friction between them absorbed a significant part of the force the coil could produce. The tail was therefore thinned and given a small raised point on the bearing surface, so that it rests on that point rather than across the whole face. The contact area falls, the friction falls with it, and more of the available force reaches the movement.

The transmission itself, and the alternatives considered for it, are described in Section 5.3.

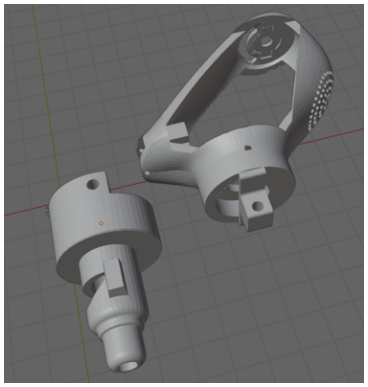
4.3.4 Female variant model

The bowing posture requires the upper body to rotate relative to the base, so the body has to be divided and a pivot introduced between the two halves. Where to place the division is the decision that shapes everything else.

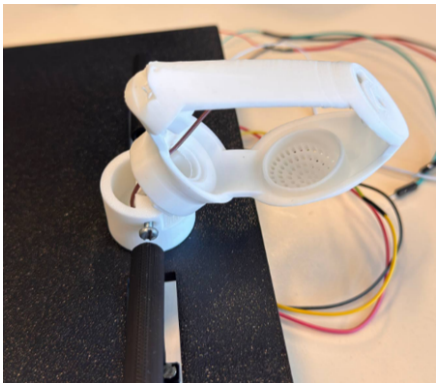
It was made horizontally, just below the point where the belly begins to swell outwards, in the widest part of the cylindrical section that forms the base of the bird and sits immediately above the lid. Three considerations converge on that position. It has to be above the lid, since anything below it is inside the box. It keeps the whole of the mounting interface described in Section 4.3.1 in the fixed lower part, so the bird still engages the drive shaft and locks into the lid exactly as the standard variant does, while the rotating group above it is body, head and wings together — which move as one in the live bird as well. And it corresponds, in the proportions of the robot, to where a real bird articulates on its legs: the movement produced is the body tilting forward over its own stance, which is what the display consists of, rather than a rotation about some arbitrary point of the body.

The section is still cylindrical at that height, which also makes the joint straightforward to dimension, since the two mating faces are flat and circular.

Cavities were added on both sides of the split to let the wiring pass through it, since the conductors for the head motor, the loudspeaker and the beak still have to reach the upper body across the joint. Lugs with matching holes were added to the upper and lower parts, so that a screw passing through them acts as the pivot. The mechanism intended to drive the rotation is described in Section 5.4.



(a) The two halves of the body in the CAD model.



(b) The printed body mounted on the box, shown tilted forward.

Figure 4.6: The split body of the female variant. The body is divided horizontally between the base and the lower belly, with lugs on either side of the split carrying the screw that forms the pivot, and cavities that allow the wiring to cross the joint. The upper body rotates forward about this pivot while the base stays fixed to the drive shaft.

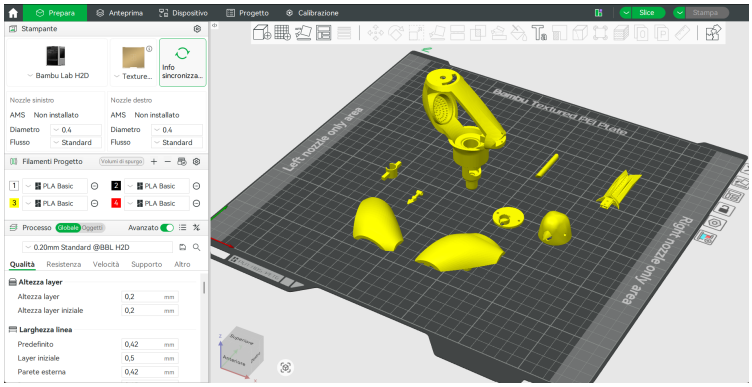
4.4 Additive manufacturing

All structural parts of the bird are printed in PLA on Bambu Lab printers — a P1S, a P2S and an H2D — all equipped with a 0.4 mm nozzle [15, 16, 17]. A layer height of 0.20 mm was used for the prints. The filament is Bambu Lab PLA Basic, of 1.75 mm nominal diameter with a tolerance of ± 0.03 mm, printed with a nozzle temperature between 190 °C and 230 °C [18]. PLA is stiff, prints accurately enough to hold the tolerances needed at the mating features, and reproduces fine detail reliably; its weaknesses, a low softening temperature and brittleness, are not limiting here, since the bird operates at room temperature and carries only small loads. TPU is used for the discs interposed in the head joint described above, where compliance is needed rather than stiffness, in a variant formulated to feed through the automatic material system of the printer, since standard TPU is too compliant to be handled by it.

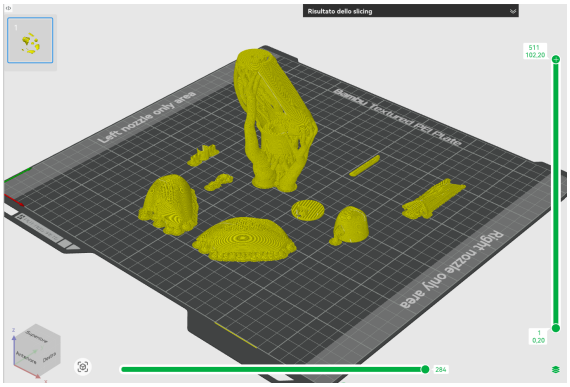
Print orientation and support settings were established part by part, by trial: each part was printed, inspected and reprinted with different settings until it came out accurate and with clean surfaces. Two considerations guided this. The first is that the layer interfaces are the weakest planes of a printed part, so features loaded transversely — such as the pivot of the female variant — have to be oriented so that the load is not carried across them. The second is that removing support material leaves a mark on the surface it was attached to, which on a robot whose appearance is part of the stimulus is a defect. Parts were therefore oriented, where possible, so that supported surfaces end up inside the assembly rather than on the outside of the bird. The slicing was done in Bambu Studio, version 2.5.0.66 [19], and Figure 4.7 shows a plate prepared for printing, before and after slicing.

Production was also limited by what was available in the laboratory. Printing stopped when a filament ran out — yellow PLA in particular, the colour used for

the canary bodies — and resumed when it was resupplied.



(a) Parts arranged on the build plate.



(b) The same plate after slicing, with the support material generated.

Figure 4.7: Preparation of a print in Bambu Studio [19]. The parts are positioned and oriented on the build plate so that supported surfaces end up inside the assembly rather than on the visible outside of the bird, and so that features loaded transversely are not stressed across the layer interfaces. The slicer then generates the support material and the tool path.

4.5 Open problems

The head axis is the region of the bird that has given the most trouble, and one problem there remains open at the end of the work. Possible remedies are given in Section 9.2.

The original failure was straightforward: the birds were returned with the head detached. The head came off its shaft under the pecking the subjects direct at it, and this was the most frequent reason for units coming back for repair. The change of fixation method described in Section 4.3.2 addressed that directly, and heads no longer come off, or do so far less often.

What the pecking does instead is displace the head gradually on its shaft. Each impact turns it a little, the offsets accumulate, and after some time in service the head comes to rest at an angle other than the one the firmware assumes. Nothing breaks, which is precisely what makes it awkward: no fault is reported, the software

runs normally, and the only sign is the appearance of the robot, which an experimenter following the session from another room does not see. In the worse cases the head is left in a position no live bird would adopt — and since the robot works by being accepted as another bird, a posture that is visibly wrong undermines the stimulus without anything indicating that it has.

The same displacement threatens the wiring. The conductors serving the beak coil cross the neck joint, and rotation beyond the intended range twists them against one another until they eventually break. The head axis was therefore restricted in software to a limited range of travel, which bounds the problem as long as the head stays where the firmware believes it to be — but that is exactly the assumption the accumulated offset breaks.

The underlying reason for all of this is that the head axis has no absolute reference. The body axis is homed against the limit switch at start-up, so any error in its position is cleared at every power cycle; the head axis has nothing equivalent, and no means of detecting or correcting an offset once it has accumulated. The heat-staked joint and the TPU disc slow the rate at which the offset builds up, but they do not remove it, and a mechanical limit or a reference feature on that axis is what the problem calls for.

Chapter 5

Actuator Development

Two actuators were added to the bird during this work: one to open the beak and one to move the tail. A third, intended to produce the bowing posture of the female variant, was designed but not built, and is described at the end of the chapter on that basis.

The chapter begins with the requirements that governed the choice of actuation technology, since the robot ends up using three different technologies on the same machine and the reasons are best set out together, and closes with a comparison of the three.

5.1 What each actuator has to do

The choice of actuator followed in each case from the movement it had to produce, and the three movements differ in a way that matters.

The **beak** opens and closes together with the vocalisation, so that what the subject sees matches what it hears. The movement is short, fast and repeated, and no intermediate position needs to be held.

The **tail** moves relative to the body, adding a further visual cue alongside the song. Like the beak, it is a brief gesture rather than a position to be maintained.

The **body pitch of the female variant** is different in kind. The bowing posture is assumed, held for a period, and released, so it requires a controlled angle that stays where it is put, over a travel of a few tens of degrees.

Beyond the movement itself, three constraints applied to all three. **Space**, since whatever is chosen has to fit inside a body of realistic proportions, alongside the head motor, the loudspeaker and the wiring. **Weight and force together**, since a heavier actuator needs a stronger one to move it, and the parts being moved have to stay light enough for the small actuators available to move them at all. And the **number of conductors**: the wiring between the box and the bird was already in place, with two conductors reserved for the beak and two for the tail. Any actuator needing more than two — a servo needs three, a stepper four — was therefore ruled out from the start for these two functions, independently of whether it would have fitted.

Brief gestures, in very little space, with two conductors, point to a coil and magnet. Electromagnetic coil–magnet actuation has been used in compact robotic

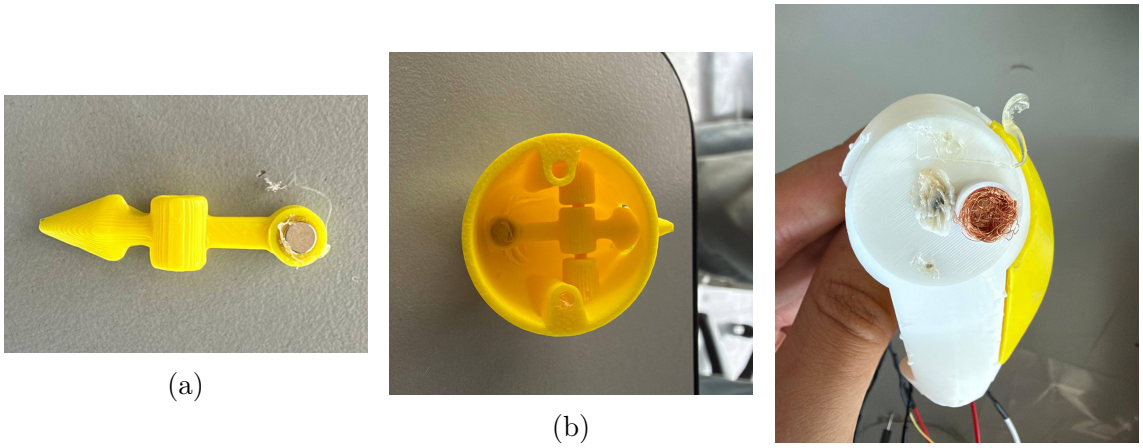
systems where small packaging and simple mechanical integration matter: Nemitz et al. integrated voice-coil actuators with 3D-printed structural components in a modular soft robot [20]. A held posture points to a servo. The female variant therefore uses a different type of actuator from the other two, and required a different route for its wiring.

5.2 Beak

5.2.1 Working principle

The beak is a lever. It pivots about a wire passing through the head from one eye to the other, as described in Section 4.2, with the visible part on one side of the pivot and, on the other, an arm that remains inside the head and carries a small bonded magnet. The visible part is the lower mandible, the upper one being fixed to the head as it is in a real bird, so the beak opens when that part moves down. A coil faces the magnet, carried in a holder on the neck disc and therefore enclosed within the head once the head is fitted.

The beak rests closed, and energising the coil opens it: the magnetic force displaces the inner arm, the lever rotates about the wire, and the mandible drops. Removing the current returns the lever to rest under the elasticity of the structure, closing the beak again. Whether the force is one of attraction or of repulsion depends on the polarity applied to the coil and on the orientation of the magnet, and the direction inputs of the driver are set once at initialisation to whichever produces the opening movement. Figure 5.1 shows the main components of the mechanism and their arrangement inside the head.



(a) (b) (c)

Figure 5.1: Components of the beak actuation mechanism: (a) the lower mandible carrying the bonded magnet, (b) the lever mechanism mounted inside the head, and (c) the driving coil mounted in its holder on the neck disc (the coil shown is a worn specimen used during various tests).

5.2.2 Drive electronics

The coil is driven by channel A of the TB6612FNG described in Section 6.4. The direction inputs are set once at initialisation to the polarity that produces the opening movement, and the actuation is performed by gating the PWM input. Table 5.1 gives the signals involved, as implemented on the board.

Table 5.1: Control signals of the beak channel and the GPIO lines assigned to them on the board.

Signal	GPIO	Function
PWMA	13	Output gating
AIN1	24	Direction
AIN2	23	Direction
STBY	17	Standby, shared with the tail channel

5.2.3 Current and dissipation

The resistance of the coil was measured with a digital multimeter, disconnected from the circuit, giving $R_{\text{coil}} = 69\,\Omega$. The board provides a series resistor $R_1 = 16\,\Omega$ on this channel, and the supply is 5 V, so the steady-state current is

$$I = \frac{V}{R_1 + R_{\text{coil}}} = \frac{5}{16 + 69} \approx 58.8\,\text{mA} \tag{5.1}$$

The coil then dissipates about 0.24 W and the resistor about 0.06 W. These figures correspond to continuous energisation, which does not occur in operation: the firmware energises the coil for a set interval and then switches it off, so the average over a session is much lower. The distinction is not academic, since the coil sits inside a printed PLA head where heat has nowhere to go, and PLA softens at a temperature well below that of most engineering plastics. The pulsed actuation is therefore a thermal requirement as much as a behavioural one.

The current drawn is under 6% of the roughly 1 A the carrier can supply per channel, so the driver operates with a very large margin.

5.2.4 Status

The mechanism was built and tested in place. With the coil fitted in its holder and the magnet bonded to the beak, connecting the two conductors directly to a 5 V supply opens the beak as intended, so the coil, the magnet, the pivot and the clearances all work as designed on the assembled bird.

What was not done is driving it from the box. The firmware for the beak existed and its pin assignment was amended during this work, as described in Section 7.9, but the beak was never commanded through the software on a complete unit, for the reasons given in Section 6.9.

5.3 Tail

5.3.1 Transmission

The coil drives a small lever in a straight line, whereas the tail has to rotate, so something has to connect the two and convert one movement into the other.

Three mechanisms were considered. A *simple lever* pivoting about a fixed point is the easiest to make, but the contact between the rod and the arm describes an arc while the rod moves in a straight line, so the geometry is satisfied exactly at one point of the travel only. A *pin engaging a slot* resolves this discrepancy kinematically rather than by accommodation, and tolerates small misalignments between the parts. A *flexible tendon with a return spring* tolerates misalignment best of all, but adds a component subject to fatigue and gives less control over the resulting position, which is then determined by the equilibrium between tension and return force rather than by a constraint.

The slot-and-pin arrangement was built and tried first, on the reasoning that the dominant uncertainty in an assembly of printed parts at this scale is the alignment between features produced in separate operations, so the mechanism most tolerant of misalignment is the one most likely to work on the first assembly and after repeated disassembly for maintenance.

What was adopted in the end is simpler than any of the three. The lever of the coil seats directly in a recess in the tail, as described in Section 4.3.3, so the two move together with nothing in between. The travel obtained is smaller than a proper transmission would give, and the geometry of the contact is not exact, but it works, it adds no parts to a region where space is scarce, and it avoids the friction and play that every additional joint introduces. Figure 5.2 shows the arrangement as built. It should be read as a working arrangement rather than a finished one: whether the movement it produces is large enough to serve as a visual cue is among the things the verification described in Section 9.2 would establish.

As with the beak, the actuator is a coil rather than a servo, for the reasons given in Section 5.2. The tail itself was made lighter, as described in Section 4.3.3, since the coil has very little force available and any excess mass directly reduces the movement obtained.

5.3.2 Drive electronics

The tail coil uses channel B of the same driver, with a series resistor $R_2 = 22\,\Omega$. Table 5.2 gives the signals.

5.3.3 Current and dissipation

The resistance of the tail coil was measured in the same way, giving $R_{\text{coil}} = 25\,\Omega$. With the series resistor $R_2 = 22\,\Omega$ and a 5 V supply, the steady-state current is

$$I = \frac{V}{R_2 + R_{\text{coil}}} = \frac{5}{22 + 25} \approx 106\,\text{mA} \tag{5.2}$$



Figure 5.2: Tail actuation mechanism as built, showing the coil mounted inside the 3D-printed body shell and the actuator lever directly engaging the recess in the tail.

Table 5.2: Control signals of the tail channel and the GPIO lines assigned to them on the board.

Signal	**GPIO**	**Function**
PWMB	12	Output gating
BIN1	27	Direction
BIN2	22	Direction
STBY	17	Standby, shared with the beak channel

The coil then dissipates about 0.28 W and the resistor about 0.25 W, again for continuous energisation, which does not occur in operation. The current is just over 10% of the roughly 1 A the carrier can supply per channel, so this channel too operates with a wide margin.

5.3.4 Status

As with the beak, the mechanism was built and tested in place: with the coil fitted in the body and its lever seated in the tail, connecting the conductors directly to a 5 V supply moves the tail as intended, confirming the engagement and the clearances on the assembled bird.

No implementation of the tail existed in the firmware, and the one added follows the same pattern as the beak, as described in Section 7.9. The client already provided a control for it, so once the command existed on the robot the function became available from the interface without any change on that side. As for the beak, however, the tail was never commanded through the software on a complete unit.

5.4 Female variant: bowing mechanism

5.4.1 The display and what it requires

The female canary responds to male song with a copulation solicitation display: the body is lowered and tilted forward, the tail brought forward and the head back, with the wings held away from the body and vibrated [21]. Reproducing it would allow experiments in which the robot provides female feedback to a live male rather than only presenting to him, and the possibility was taken up during a wait for materials, in the circumstances described in Section 1.5.

Two things follow for the design. The movement is a posture that is assumed and held rather than a brief gesture, which determines the choice of actuator. And it is a compound movement, of which the mechanism considered here reproduces only the lowering and forward tilt of the body — a simplification whose behavioural adequacy has not been tested.

This section is divided accordingly. What was built and verified is described first; the actuation, which was worked out but not implemented, follows separately.

5.4.2 What was built

The body was split horizontally between the base and the lower belly, as described in Section 4.3.4. Lugs were added on either side of the split, and a screw passing through them forms the pivot about which the upper body — body, head and wings as a rigid group — rotates while the base stays fixed to the drive shaft. Cavities on both sides of the split allow the conductors serving the head motor, the loudspeaker and the beak to cross the joint, since they still have to reach the upper body.

The parts were printed and assembled, and the resulting body tilts about the pivot as intended, which establishes that the kinematic arrangement works and that the split does not compromise the routing of the wiring. Figure 4.6 shows the result.

The actuator was also validated, independently of any mechanism. A GWS PICO BB micro servo was driven from a separate Raspberry Pi, without the electronics box, in order to confirm the control chain before committing to an integration. GPIO 13 was used for the control signal, on the grounds that it supports hardware PWM generation on this processor, which produces a cleaner pulse train than a software-timed signal. A short program generated a 50 Hz signal with pulse widths between 1000 µs and 2000 µs, and the servo followed the commanded positions over the full range. Table 5.3 gives its specification.

That pin is not free on the board, however: as Table 6.3 shows, GPIO 13 carries the PWM signal of the channel serving the beak, so the integration will have to allocate another line with hardware PWM capability.

Table 5.3: Specification of the micro servo used for the bowing mechanism.

Quantity	**Value**
Type	GWS PICO BB micro servo
Stall torque	0.7 kg cm
Transit speed	0.12 s per 60°
Mass	5.7 g
Control signal	50 Hz PWM, 1000 µs to 2000 µs

5.4.3 Proposed actuation

What remains is the transmission between the servo and the upper body. It was worked out during the development but not built, and is set out here as a proposal rather than as a result. Figure 5.3 shows the arrangement.

Arrangement

A servo is used rather than a coil because the posture has to be held: a coil holds nothing once de-energised, and holding it energised would mean a continuous current with the resulting position determined by the load rather than by the command, and with the thermal consequences discussed in Section 5.2.

There is no room inside the upper body at the height where the force has to be applied, and nothing may be visible on the outside of the robot. The servo would therefore sit in the base of the bird, mounted on it once the base is fitted to the drive shaft, so that it turns with the bird rather than staying fixed while the body rotates above it, and a steel wire would carry the force upward. This is a standard resolution of a packaging constraint, and the same principle by which tendon-driven robotic hands place their motors in the forearm. A wire of that length buckles in compression well below the load it sustains in tension, so it can pull but not push, and the return has to come from elsewhere.

The division of labour between the two is therefore asymmetric. The servo produces the movement in one direction only: it draws the wire down, and since the wire attaches ahead of the pivot, the upper body rotates forward into the bowed

position. Releasing the wire does not restore it — once the body is tilted, its centre of mass lies ahead of the pivot and gravity acts to keep it there — so a spring provides the restoring moment, anchored at the rear of the base and attached to the upper body so that tilting forward extends it. When the servo releases, the spring returns the body upright.

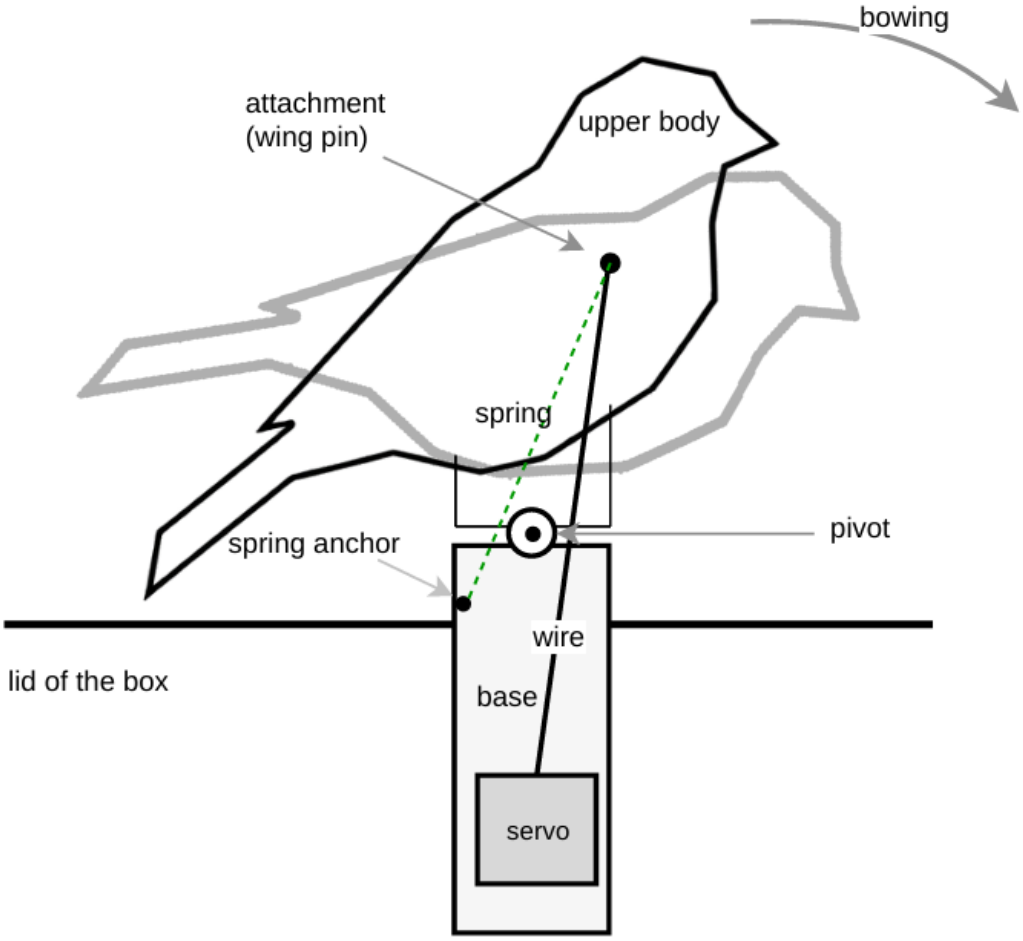


Figure 5.3: Kinematic scheme of the proposed bowing mechanism. The base stays fixed to the drive shaft and the upper body rotates about the pivot at the split, into the position shown dashed. The servo is mounted on the base below the lid of the box and draws a steel wire upward to the pin that joins the wings; since that point lies ahead of the pivot, the body tilts forward. The spring runs between the same point and an anchorage on the base above the lid, so that tilting extends it and releasing the wire lets it return the body upright.

Where the spring and the wire attach

Both act on the upper body, and both have to be inside it: nothing may move on the outside of the robot, for the reason given in Section 4.1, and the spring has to

run within the body in any case.

Two attachment points were considered. The first is a hook added to a wing for the purpose. The second is the pin that already joins the two wings, described in Section 4.2: it passes transversely through the body at approximately the right height, it is retained on both sides, and using it would save a component and a set of mating features in a region where space is scarce. Whether it is stiff enough to carry the load, and whether attaching to it would interfere with the wings, were not established.

The lower end of the spring anchors to the base, towards the rear of the bird so that the spring is extended as the body tilts forward. That anchorage does not exist yet either, and would have to be added to the base in the same operation — a small feature to retain the end of the spring, positioned so that it does not interfere with the servo or with the wiring crossing the joint.

Forces

Two conditions have to be satisfied, and they pull in opposite directions.

The spring has to be stiff enough that its moment about the pivot exceeds the gravitational moment of the tilted body at the end of the stroke, since that is the condition for the body to return. This sets a lower bound on its stiffness, determined by the mass of the rotating group and by how far its centre of mass moves ahead of the pivot.

The servo then has to overcome that spring, plus the gravitational moment at the start of the stroke and the friction at the pivot. The force available at the wire is the servo torque divided by the radius at which it attaches to the horn,

$$F = \frac{\tau}{r} \tag{5.3}$$

so with a stall torque of 0.7 kg cm and a wire attached at, say, 1 cm from the axis, about 6.9 N would be available. The attachment radius has not been chosen, since the servo was never fitted, and it is the parameter that decides how the fixed product of force and stroke is divided between the two.

Two conclusions follow even so. The attachment should be as far above the pivot as the geometry allows, since the force required falls as that distance grows — an argument in favour of the wing pin, which sits higher than most alternatives. But the stroke required grows with the same distance, so the horn radius has to be chosen accordingly: for a given servo the product of force and stroke is fixed, and the geometry only decides how it is apportioned.

In practice the spring would be selected by trying those available against the assembled mechanism, as the other components of the robot were. What the relations above give is the range to look in, and the assurance that a spring chosen only for being stiff enough to return the body may be too stiff for the servo to overcome. The measurements that would close the analysis are described in Section 9.2.

What integration would require

Beyond building the mechanism, three things would be needed. Three conductors — supply, ground and control — have to reach the servo from the box, and the unused contacts of J2 can carry them as they are. A free line with hardware PWM capability has to be allocated on the board, for the reason given above. And the firmware needs a control path for it, which does not exist: a servo is driven by a pulse width rather than by gating an output, so it does not follow the pattern established for the beak and the tail, and it would need its own initialisation, its own method on the `Canari` class and its own entry in the command table described in Section 7.4.

Status

The split body was designed, printed and assembled, and the servo was validated on the bench. The transmission between the two was specified but not built, and the integration was not carried out.

5.5 Comparison of the three actuator types

Ending up with three different actuation technologies on one small robot is not an accident of availability: each answers to a movement the others cannot produce, and the criteria that decided each case are summarised in Table 5.4.

Table 5.4: Comparison of the three actuation technologies used on the robot, against the criteria that governed the choice in each case.

	Coil and magnet	**Micro servo**	**Stepper**
Space needed	Very little	Little	Moderate
Connections	2	3	4
Linkage	None	Horn and linkage	Mount only
Control	Effectively on/off	Proportional, held	Proportional
Holds position	Only while energised	Yes, at low current	Yes
Acoustic noise	Low	Gear noise	Low with StealthChop

A brief gesture in very little space is best served by a coil: it is the smallest and simplest option, it needs only two conductors, and its inability to hold a position does not matter when none is required. A held posture needs a servo, and with it a third conductor. Precise, repeatable orientation needs a stepper, with four — the body and head motors were already specified when the work began and were not selected here.

The number of connections matters more in this robot than it would elsewhere, since every additional conductor has to cross from the box into the bird and, in the case of the head, across a rotating joint where it is subject to the twisting described in Section 4.5.

The last row of the table would not appear in a general robotics context. In an enclosure containing a microphone, any sound an actuator makes falls on the signal of interest, and at exactly the moments that matter. The consequences of this for the stepper drivers are described in Section 6.3; for the servo, whether its gear noise is audible in the enclosure has not been measured, and this is proposed as future work in Section 9.3.

Chapter 6

Electronic System

Everything the robot does electrically passes through a single custom board. This chapter describes that board, working outwards from its architecture through the power distribution, the drivers for the motors and the coils, the sensing front end and the audio chain, to the connectors and the harness. It ends with the assembly of the boards produced during this work and with the commissioning that followed, since finding out why particular units did not work occupied a significant part of the available time and determined what could and could not be verified before the end.

6.1 Overview

The electronics are built around a single custom board, *Canaribot V1.0*, designed at ETIS by Fred de Melo. It sits inside the box together with the Raspberry Pi and connects to it through a 40-pin ribbon cable carrying the whole GPIO header; everything else — motors, sensors, loudspeaker — connects to this board. Figures 6.1 and 6.2 show the schematic and the assembled board.

The board does not implement the driver and amplifier circuits itself. Instead it provides sockets for commercially available modules: two SilentStepStick carriers for the stepper drivers, two SparkFun HX711 boards for the load cells, a Pololu carrier for the TB6612FNG dual H-bridge, an Adafruit carrier for the PAM8302A audio amplifier, and an encapsulated Traco Power Direct Current (DC)/DC converter. What the board itself provides is the wiring between these modules and the Raspberry Pi, the power distribution, the connectors towards the bird and the sensors, and a small number of discrete components. Table 6.1 lists them.

Two consequences of this approach recur throughout the chapter. A failed module can be replaced without any rework at component level, which matters in a laboratory where the boards are assembled and repaired by students rather than by technicians. But part of the electrical behaviour of the system is then determined on the modules rather than on this board, and one consequence of that kind is described in Section 6.6.

The discrete resistors are carbon film parts of the CFR25 type, in the 0207 package, with a tolerance of $\pm 5\,\%$.

Table 6.1: Principal components of the Canaribot V1.0 board, with the function of each.

Ref.	**Component**	**Function**
PS1	Traco Power TEN 25-1211	DC/DC conversion, 9 V to 18 V in to 5 V / 5 A out
U3	Watterott SilentStepStick TMC2130 v2.0	Head stepper driver
U4	Watterott SilentStepStick TMC2130 v2.0	Body stepper driver
TB1	TB6612FNG (Pololu carrier)	Dual H-bridge, beak and tail
U1, U2	HX711 (SparkFun SEN-13879)	Load cell amplifier and 24-bit Analog-to-Digital Converter (ADC)
U6	PAM8302A (Adafruit carrier)	2.5 W class-D audio amplifier
J1	IDC 2×20	Raspberry Pi GPIO header
J2	Hirose DF11-16DP-2DSA	Harness to the bird
J3	Barrel jack	External power input
J4, J5	Pin header 1×5	Load cells
J7, J8	Pin header 1×3	Limit switches
J9	Pin header 1×4	Body stepper motor
J11, J12	Screw terminal	5 V distribution
J6, J10	Screw terminal	12 V distribution
J14–J17	Jumper	Motor supply voltage selection
J18	Audio jack 3.5 mm	Audio input from the USB sound card
J19	Screw terminal	Loudspeaker output
R1, R2	16 Ω, 22 Ω, ±5 %	Current limiting, beak and tail
R3–R6	10 kΩ, ±5 %	Pull resistors on host control lines
C1, C2	Rubycon, 100 µF, 16 V	Decoupling, motor supply rails

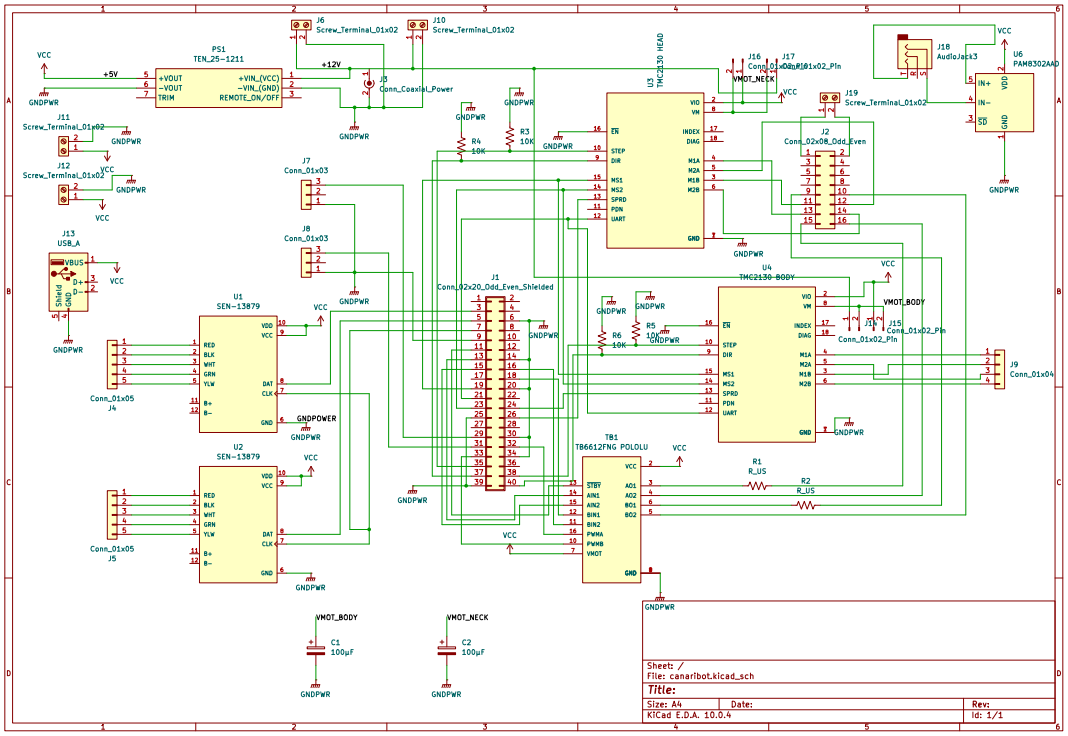


Figure 6.1: Complete schematic of the Canaribot V1.0 board, showing the module sockets, the power distribution, and the connectors towards the Raspberry Pi, the bird and the sensors.

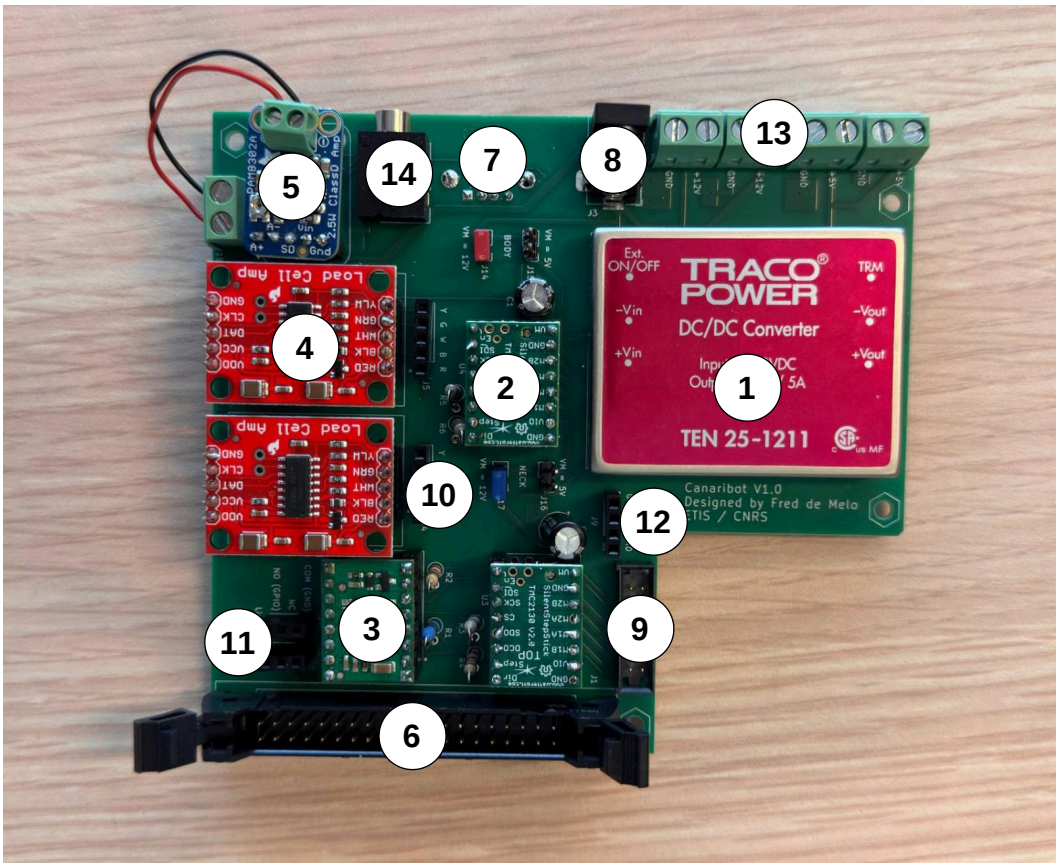


Figure 6.2: The assembled Canaribot V1.0 board. The modules are: (1) the Traco Power TEN 25-1211 converter, which derives the 5 V rail from the external supply; (2) the two SilentStepStick TMC2130 drivers for the head and body axes; (3) the TB6612FNG dual H-bridge on its Pololu carrier, which drives the beak and tail coils; (4) the two HX711 modules amplifying and digitising the load cell signals; and (5) the PAM8302A audio amplifier. The interfaces towards the Raspberry Pi are (6) the 40-pin IDC header carrying the whole GPIO and (7) the USB connector supplying it. Power enters through (8) the barrel jack and is distributed by (13) the screw terminals for the 5 V and 12 V rails. The remaining connectors serve the robot and the sensors: (9) the Hirose DF11 connector towards the bird, (10) the load cell headers, (11) the limit switch headers, (12) the body motor header, and (14) the audio input jack from the USB sound card.

6.2 Power distribution

Power enters the box through a barrel jack (J3) from an external mains adapter. The adapter is not a specified part of the design: any unit providing 12 V at 2.5 A serves, and several different ones were used interchangeably over the course of the work. From the jack the supply divides immediately. One branch goes to the Traco Power TEN 25-1211 converter (PS1) [22], which accepts 9 V to 18 V at its input and provides 5 V at up to 5 A; the other is distributed as it arrives, and serves as the supply available to the stepper motors. Only the 5 V rail is therefore produced on the board.

An encapsulated commercial converter was used rather than a discrete regulator design. It provides galvanic isolation between input and output, is specified as a complete unit, and removes the switching regulator design from the scope of the board. Its optional remote-enable and output-trim inputs are left unconnected, and the converter operates at its nominal output whenever power is applied. The wide input range is also convenient in a laboratory, since the apparatus can be powered from different 12 V supplies without modification.

One property of an isolated converter has to be kept in mind. Its input and output grounds are not connected internally, so a connection has to be made externally if the two sides are to share a common reference. On one of the boards assembled during this work this connection was not present, with the consequences described in Section 6.9.

The two rails are supplied by the same 30 W adapter. The converter is rated for 25 W, while the logic, sensing and audio circuits draw considerably less than this in normal operation. The actual consumption of the complete system was not measured.

The 5 V rail supplies the Raspberry Pi, the logic, the load cell modules, the audio amplifier and the coil driver. Both rails are brought out to screw terminals, which provide convenient connection points for auxiliary devices and for voltage measurements during commissioning. Table 6.2 lists the two rails and their main consumers.

Motor supply selection

Each stepper motor can be supplied from either rail, selected by jumpers: J14 and J15 for the body, J16 and J17 for the head, with the corresponding silkscreen markings.

The possibility of choosing between 5 V and 12 V allows the motor supply to be adjusted according to the requirements of each axis. The higher voltage can improve motor performance at higher speeds, while the lower voltage can reduce electrical and acoustic effects. Since the two axes have different mechanical characteristics, the supply can therefore be selected independently for each one.

Decoupling

Two 100 µF electrolytic capacitors rated for 16 V, C1 and C2, are placed on the motor supply nets. They provide local energy storage close to the motor drivers and

Table 6.2: Supply rails distributed on the board and the loads each serves.

Rail	Source	Nominal	Consumers
12 V	External adapter at J3	12 V, 2.5 A	Input to the Traco converter; motor supply distributed through J6 and J10
5 V	Traco converter output	5 V, 5 A	Raspberry Pi, logic, load-cell modules, audio amplifier and coil driver; distributed through J11 and J12
VMOT_BODY	12 V or 5 V, selected by jumper	—	Body stepper power stage, U4
VMOT_NECK	12 V or 5 V, selected by jumper	—	Head stepper power stage, U3

help reduce voltage fluctuations caused by the rapid changes in motor current.

6.3 Stepper drivers

The two rotation axes are driven by TMC2130 drivers [23] mounted on Watterott SilentStepStick carrier boards, revision 2.0 [24], U3 for the head and U4 for the body. The choice of this driver is mainly related to the acoustic requirements of the robot.

Stepper motors can generate audible noise during operation, which is undesirable in this application because the robot is used to study vocal behaviour and its movements may occur at the same time as acoustic stimuli are played. The TMC2130 provides a *StealthChop* operating mode designed to reduce the audible noise produced by the motor, particularly at low and moderate speeds. This makes it suitable for the two axes of the robot, although the actual reduction in acoustic emission was not measured during this work.

Configuration and open-loop operation

The drivers are configured by the Raspberry Pi over Serial Peripheral Interface (SPI) at start-up, and the movements are then commanded through the STEP and DIR signals. One of the parameters set in this way is the microstepping resolution, which determines how many step pulses correspond to a given rotation — so the step counts used in the positioning routines depend on it, and the firmware archive contains a variant where the ratio was changed and the calibration counts divided accordingly. For the head axis the standstill current is set to zero, to avoid unnecessary heating inside the enclosed bird body.

Both axes operate in open loop: the firmware infers the position from the number of pulses it has sent rather than measuring it. A limit switch does not change that — it is read only during the homing procedure at start-up, to establish where the axis is before counting begins, and is not consulted again while the robot is running. So a step lost during a session goes undetected and persists until the next power cycle, when homing clears it. The head axis, which has no limit switch, has no such opportunity, and that is the subject of the open problem stated in Section 4.5.

6.4 Beak and tail driver

The beak and tail coils are driven by a TB6612FNG dual H-bridge [25] mounted on a Pololu carrier [26], designated TB1 on the schematic. It provides two independent full-bridge output stages sharing a common supply and a common standby input. The device itself is rated for 1.2 A of average output current per channel, while the carrier is specified for approximately 1 A continuous per channel.

The device is sold as a driver for small brushed DC motors, and it suits a coil equally well: both present the same kind of electrical load, an inductance in series with a resistance. The bridge applies a voltage of the chosen polarity across the coil, while the output stage provides a recirculation path for the current when the inductive load is switched off.

Control

Each channel is controlled by two direction inputs, which determine the polarity applied across the load, and a PWM input which gates the output. A single standby input, STBY, is shared between the two channels and must be high for the device to operate at all; when it is low, both outputs go to a high-impedance state regardless of the other inputs.

The firmware sets the direction inputs once at initialisation and then actuates by gating the PWM input alone, since the movement required is a single gesture rather than a bidirectional motion, and the return to rest comes from the compliance of the structure rather than from reversing the current.

The shared STBY input has one consequence worth making explicit: the beak and the tail are not electrically independent. Anything holding STBY low disables both at once, so a fault on this single net presents as the simultaneous failure of two apparently unrelated subsystems — which is what was observed on the reference unit, as described in Section 6.9.

Current limiting

Two resistors are placed in series between the driver and the harness: R1, of 16 Ω, on output AO1 of channel A, and R2, of 22 Ω, on output BO1 of channel B. Together with the resistance of the coils they determine the current that flows when a channel is energised, as computed in Sections 5.2 and 5.3, and they also bound the current the driver can deliver in the event of a short circuit in a coil or in the harness.

GPIO mapping

Writing firmware for this driver requires knowing which GPIO of the Raspberry Pi corresponds to which control input of the driver, and that correspondence was not written down anywhere available at the time. It was therefore reconstructed from the material that was: the pinout of the Pololu module, which relates each signal name to a pin number on the module; the design files of the board, which relate that pin to a pin of connector J1; and the pinout of the Raspberry Pi, which relates the pins of its header to the GPIO numbers used in software. The three do not use the same numbering — the thirty-third pin of the header is GPIO 13 — so the mapping only emerges once they are put together.

The middle step came from the IPC-D-356 netlist exported from the board design, which lists for every pad of every component the electrical net it belongs to; pads appearing on the same net are connected, so extracting the entries for the driver and for J1 gives their correspondence. Table 6.3 is the result.

Table 6.3: Mapping between the control signals of the TB6612FNG, the pins of connector J1, and the GPIO lines of the Raspberry Pi, as implemented on the board.

TB6612 pin	Signal	J1 pin	GPIO
10	PWMA	33	13
11	AIN2	16	23
12	AIN1	18	24
13	STBY	11	17
14	BIN1	13	27
15	BIN2	15	22
16	PWMB	32	12

The firmware inherited with the platform declares its own pin assignment for this driver, and the two do not match. Of the four definitions grouped under the beak, the standby input corresponds, while the other three match the signals of the opposite driver channel from the one whose outputs reach the beak coil. The three are consistent with one another, so the firmware describes a channel correctly; what differs is which channel the names refer to.

Neither actuator had ever been driven from the box, so this part of the firmware had never been exercised, and nothing in the behaviour of the platform would have shown the difference. The platform has also been developed by several people over several years, as described in Section 1.3, with both the software and the board revised at different times, so a divergence of this kind between two documents that nothing keeps in step is not surprising.

The mapping can be checked against the parts of the firmware that were exercised. The step and direction signals of both stepper axes, the SPI channel assignments and both limit switch inputs are declared in the firmware and agree with the board in every case — which is expected, since those subsystems worked, and which also confirms that the file examined describes the board in use.

6.5 Load cell front end

Each of the two instrumented perches rests on a SparkFun straight-bar load cell of 100 g capacity [27], read by an HX711 module [28, 29], U1 and U2 on the schematic, with the load cells connected through the pin headers J4 and J5. The silkscreen carries the conventional colour code of load cell wiring, so the connection can be made correctly without reference to a document.

A load cell uses strain gauges to convert the deformation caused by a load into a small electrical signal. This signal is only of the order of a few millivolts and therefore requires amplification before it can be measured reliably. The HX711 combines the required amplifier and a 24-bit analogue-to-digital converter in a single module, making it suitable for directly reading the load cells.

The device communicates with the Raspberry Pi through two digital lines using a simple synchronous serial protocol. The tare and threshold logic built on top of this is described in Section 3.4.

6.6 Audio

The signal from the USB sound card enters the board through J18, a 3.5 mm jack, and reaches the inputs of the PAM8302A amplifier (U6). Figure 6.3 shows this section of the schematic. Since the source is monophonic, the jack is wired accordingly: the tip and the sleeve drive the two signal inputs of the amplifier, and the ring is left unconnected.

The PAM8302A [30] is a class-D amplifier capable of delivering 2.5 W into a 4 Ω load, and correspondingly less into the 8 Ω loudspeaker used here. A class-D stage is suitable for this application because it is efficient and compact, and the amplifier is available as a module that mounts directly on the board. Its gain is set by resistors on the carrier, which also provides a potentiometer allowing it to be adjusted.

Its output is bridge-tied, meaning that neither terminal is at ground potential and the loudspeaker is driven differentially between the two. This matters for measurement: the signal has to be observed between the two output terminals and not between either of them and ground.

The output is not routed on the board

One aspect of the audio path is worth setting out explicitly, because it is invisible in the fabrication data. The amplifier module carries a five-way header along one edge, which provides the supply, the shutdown input and the two signal inputs, and it is through this header that it plugs into the board. Its output, however, leaves through a two-way screw terminal on the opposite edge of the module, and that terminal is connected to J19 with wires during assembly. The connection is therefore made manually during assembly and does not appear in the fabrication files.

A point of nomenclature belongs here as well, because it is a plausible source of error. The two signal pins on the header are labelled `A+` and `A-` on the silkscreen of the module, whereas the schematic designates them `IN+` and `IN-`. They are inputs in both cases; the outputs are the screw terminal on the opposite edge. The labelling

nonetheless invites the opposite reading, and a measurement taken between `A+` and `A-` in the belief that they are the outputs would characterise the wrong end of the amplifier.

The shutdown input

The amplifier provides an active-low shutdown input, SD, which disables the output stage when held low. On the Adafruit carrier this input is pulled high [31], so leaving it unconnected on the Canaribot board leaves the amplifier enabled by default. It is not driven by the host, and nothing on the board acts on it.

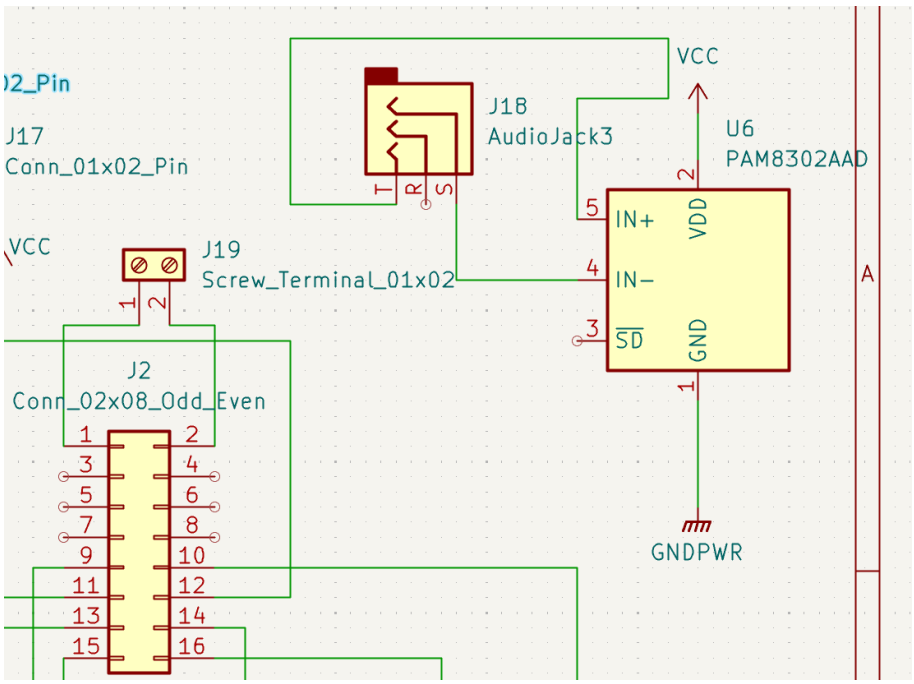


Figure 6.3: Audio section of the schematic. The signal from the USB sound card enters at the jack connector J18, whose tip and sleeve drive the two signal inputs of the PAM8302A amplifier (U6) while the ring is left unconnected, the source being monophonic. The screw terminal J19 carries the loudspeaker connection towards J2 and the harness; the output stage of the amplifier is not routed on the board and reaches J19 through wires fitted during assembly. The shutdown input SD is not driven by the board, the carrier providing its own pull-up.

6.7 Connectors and harness

The board carries three categories of connector: the interface to the Raspberry Pi, the interface to the bird, and a set of local connectors for the sensors and the body motor.

To the Raspberry Pi

J1 is a 40-pin Insulation-Displacement Connector (IDC) header carrying the complete GPIO header of the Raspberry Pi, connected by a ribbon cable. A single mechanical operation therefore connects every signal, and the Raspberry Pi requires no modification.

One consequence is worth recording, because it shaped how the diagnostic work of Section 6.9 was carried out. Once the cable is fitted, the pins of the Raspberry Pi header are physically inaccessible, so any measurement on a GPIO line has to be made on the board side of the connector — and a continuity measurement between the host and a driver input means disconnecting the ribbon cable first, so that what is measured is the board alone.

To the bird

J2 is a Hirose DF11 header with sixteen contacts [32], and is the only electrical interface between the box and the bird. It carries four conductors for the head stepper motor, two for the loudspeaker, and two pairs for the beak and tail actuators, with the remaining contacts unused.

Harness pinout

The harness follows an arrangement that has to be reproduced on every unit, and one was built during this work for the box that lacked it. On the box side, female jumper terminations are soldered by hand to each conductor of the ribbon cable, one per colour. On the bird side, the leads of the head motor, the loudspeaker and the coils are fitted with male terminations, soldered one unit at a time as each bird is assembled. Connecting a bird is then a matter of mating the two sets colour by colour, and disconnecting it of pulling them apart — which is what makes a bird removable in the few seconds described in Section 3.2.

Since the conductors are distinguished only by the colour of their insulation, which colour goes to which contact is a convention rather than a specification, and the one adopted is recorded in Table 6.4. Building a harness means following it: each conductor is soldered to the contact its colour is assigned to, and an error there is not visible once the connector is assembled, since the bird simply does not behave as commanded and the fault looks electrical rather than manual.

Six of the sixteen contacts are unused. Three of them would carry the servo of the female variant, as described in Section 5.4, which is why that addition requires no modification to the board or to the harness.

Local connectors

The remaining connectors serve components mounted inside the box or on its lid: J4 and J5 for the load cells, J7 and J8 for the limit switches, J9 for the four conductors of the body motor, the screw terminals for the supply rails, and J19 for the loudspeaker. All of these are labelled on the silkscreen with the function of each contact, which is a good feature of the board and should be preserved in future revisions. The

Table 6.4: Correspondence between the contacts of connector J2, the conductors of the harness and their destinations inside the bird, established by continuity testing. Contacts are numbered as in the board design, starting from the end at which the green conductor sits.

J2 contacts	Wire colours	Destination
1, 2	Green, blue	Loudspeaker
3–8	—	Unused
9, 10	Violet, grey	Tail coil
11–14	White, black, brown, red	Head motor
15, 16	Orange, yellow	Beak coil

one connector not labelled in this way is J2, the interface to the bird — and it is precisely the connector whose pinout had to be reconstructed by measurement.

6.8 Assembly

Ten boards were assembled in the course of this work. The board is populated entirely with through-hole components and with modules mounted on pin headers, so it is assembled by hand.

The parts are fitted lowest first, so that the board can be laid face down on a flat surface at each stage: the discrete resistors, then the pin header sockets for the modules, then the connectors, the electrolytic capacitors, and finally the encapsulated converter. The modules themselves go in last, once every soldered joint has been made and inspected, so that a fault in the soldering does not put a module at risk and so that the modules can be removed later without desoldering.

The audio amplifier output has to be connected to J19 with wires from the screw terminal on the module itself, as described in Section 6.6. This connection is made during assembly and does not appear in the fabrication data, so it has to be known in advance by whoever assembles a board.

Not all ten boards were completed. Component availability was the binding constraint throughout: assembly stopped when a part ran out and resumed when it was resupplied, so the boards are at different levels of completion. This is a normal situation in a laboratory working from a shared stock of components, but it means the number of boards assembled and the number of working units available at a given moment are not the same thing.

Each completed board was checked once assembled, to confirm that the supply rails were correct and that the subsystems responded as expected. Where something did not work, the board had to be reworked: desoldering a component to replace it, correcting a joint, or adding a connection. This rework accounted for a significant part of the time spent on the boards, more than the original assembly in some cases, and what it uncovered is the subject of the next section.

6.9 Commissioning the assembled boards

A board that has been assembled correctly does not necessarily work, and a significant part of the time available went into finding out why particular units did not. This section describes how that was done and what was found.

It also accounts for a limitation that recurs throughout this thesis. Driving the beak and the tail requires a working board, and until one was available neither actuator could be commanded from the box at all — so the mechanisms described in Chapter 5 were developed and verified by energising them directly from a supply, and the software path was never exercised. The first board to work reliably was ready only in the closing stages of the work.

6.9.1 Method

Finding out why a subsystem does not respond is largely a matter of working backwards along the path the signal is supposed to take. Behind any actuator there is a chain — the software, the GPIO, the ribbon cable, the board, the driver, the harness, and the component itself — and the fault is somewhere along it. Testing at a point in the middle indicates which half to examine next, which is faster than checking every link in order.

Two rules were followed throughout. Continuity is checked before voltage, since it is done with the board unpowered and establishes whether a connection exists at all. And where a working unit is available, comparing the two is often faster than reasoning from the schematic, because any difference between them is a candidate cause. The same reasoning applies to the software: a short program that drives one driver channel and nothing else, with none of the rest of the application running, removes most of what could be wrong before a measurement is taken.

Three boards are referred to below. **Unit A** is the reference unit already in service at the beginning, built to the older revision of the board. **Unit B** is a board of the new revision assembled during this work and commissioned in detail. **Unit C** is a further board of the same revision assembled afterwards, and is the one that entered service.

6.9.2 Faults found

Table 6.5 summarises the seven faults identified.

The second entry is not a fault in the ordinary sense. On Unit A the enable input of the driver never went high whatever the software did, and continuity testing found no connection between that pin and the host header, so the beak and the tail could not be driven on that unit at all regardless of the firmware. This explains why two apparently independent actuators had never worked on the reference unit despite the beak having a firmware implementation behind it, and it is the reason the work moved to the newer revision.

Three of the others are worth a closer look.

Table 6.5: Faults identified during the commissioning of the assembled boards, with the symptom observed, the cause where one was established, and the outcome.

Unit	Subsystem	Symptom	Cause	Outcome
All	Beak	No response to commands, though the coil worked when driven directly	Pin assignment in the firmware did not correspond to the driver channel serving the beak	Discrepancy identified; correction not verified
A	Beak, tail	Driver outputs floating; STBY low regardless of command	Enable line of the driver not reaching the host on that board	Work moved to the new revision
B	Power	Raspberry Pi would not start; 5 V rail at 2.26 V; supply cable warm	Converter return path not closed on that board	Link added; not required on Unit C
B	Audio	Amplifier ground reading 5 V	Ground reference not established at the module	Connection made
B	Audio	Signal at the amplifier input, nothing at the output	Shutdown input observed going low during playback; cause not determined	Resolved by rebuilding
B	Head axis	Rotation in one direction only	Not determined	Resolved by rebuilding
All	Playback	Channel count error on the direct hardware device	Format mismatch that `hw` does not resolve; `plughw` performs the conversion	Characterised

6.9.3 The beak pin assignment

The beak did not respond to any command, but connecting the coil directly to 5 V produced the expected movement, so the mechanism was sound and the fault lay in the drive chain.

The first thing to check is whether the GPIO the firmware drives is the one the board expects, since a mismatch there produces exactly that symptom: a chain intact everywhere except that its two halves are attached to different pins. The schematic was not available at the time, so the mapping was reconstructed from the fabrication data by the procedure described in Section 6.4; it did not agree with the firmware, and the definitions were amended accordingly.

Whether the amendment was correct could not be established, since the beak was never subsequently driven from the box. A later comparison against the project files showed that the reconstruction had itself been imperfect, and that the assignment in Table 6.3 is the one the board implements. It is worth recording as it stands: a real discrepancy was identified, and the correction was made against a mapping that had not itself been checked against the source.

6.9.4 The converter return path

The converter return path

Unit B did not power up: the Raspberry Pi showed no sign of life and the cable supplying it grew warm. Power was removed before investigating further, since continued operation under these conditions could have damaged the converter or the connected electronics.

The Raspberry Pi itself was undamaged and started normally from a separate mains adapter. With the board powered, the nominal 5 V rail measured only 2.26 V, indicating that the supply was not regulating correctly under load. Continuity measurements did not reveal a direct short circuit.

For comparison, Unit A was examined physically. This unit belongs to an older PCB revision and carried a hand-added wire link on its underside between the $-V_{\mathrm{in}}$ and $-V_{\mathrm{out}}$ terminals of the converter. Unit B, which belongs to the newer revision, did not include this connection. Adding a similar link to Unit B restored normal operation.

This observation should not be interpreted as evidence that the same connection is required by the newer PCB revision. Unit C, assembled later from the same revision as Unit B, operated correctly without it. The modification therefore served as an effective diagnostic and temporary corrective measure on Unit B, but the underlying cause of the fault was not established.

6.9.5 The amplifier shutdown input

On Unit B a signal was present at the input of the audio amplifier but nothing at its output. Replacing the module with another of the same type changed nothing, which put the cause on the board rather than in the device. Monitoring the shutdown input showed it sitting high while the system was idle and going low during playback,

disabling the amplifier at exactly the moment it was needed. That accounts for every observation, including the very faint reproduction audible on headphones connected across the output terminals, which is the residual coupling through a disabled device rather than an amplified signal.

Why it went low was not established, and the design offers no obvious explanation: the pin is not connected on the Canaribot board, and the carrier pulls it high, so under normal conditions nothing should have been driving it down. Unit C, built to the same design, did not show the fault, which suggests the behaviour was specific to that board rather than inherent to the design — but the cause was not identified, and the board was set aside rather than understood.

6.9.6 Rebuilding rather than continuing

Unit B was not brought into service. After three faults on it, plus the head axis rotating in one direction only, a further board was assembled from new; Unit C worked on commissioning and is the unit that entered service.

Rebuilding restored function but established nothing about the head axis, so nothing can be said about whether that fault will appear again. The decision was reasonable nonetheless: a board that has produced three faults is likelier to produce a fourth than a fresh one, a bare board costs little, and the remaining hypotheses would have needed an oscilloscope, which was not available. Unit B was kept rather than discarded, so a comparison remains available if the behaviour recurs.

One conclusion follows from this work. Several of the problems encountered involved undocumented differences between assembled units, including hand-added connections that were not represented in the available design files. In one case, the relevant difference was found only by physically comparing boards from different revisions. Any modification made during assembly or repair should therefore be documented when it is introduced, so that future users can distinguish an intentional change from a fault or from a revision-specific feature.

Chapter 7

Firmware and Control Software

The embedded software and the client application were already written when the internship began, by Arnaud Blanchard and Léo Coquet as described in Section 1.3. This chapter describes how they work, since the diagnostic work of Section 6.9 depends on understanding them and since several of the design decisions described in earlier chapters are visible only in the software. It ends with the changes made during this work.

7.1 Environment

The application runs on the Raspberry Pi 3B+ [12] under Raspberry Pi Operating System (OS) [33], kernel 6.1.21-v7+, and is written in C++. Access to the GPIO, SPI and PWM peripherals is provided by the pigpio library [34]. In this application the library is used through its direct C interface, whose low-level access to the Raspberry Pi peripherals requires the program to run with elevated privileges. The WebSocket server uses `libwebsockets` [35], messages are encoded with `nlohmann/json` [36], audio capture uses ALSA [37], and the spectral analysis of the microphone signal uses FFTW3 [38].

C++ is an appropriate choice here for two reasons. The step pulses driving the stepper motors, and the bit-banged protocol reading the load cell converters, are generated in software and require timing determinism that an interpreted language on a general-purpose operating system does not reliably provide. And the libraries needed for the peripherals, for the network interface and for the spectral analysis are all natively available in C or C++.

One limitation of this arrangement is worth noting. The step pulses are generated by a loop that sets a pin, waits, and clears it, asking the operating system for each interval — and a general-purpose operating system does not guarantee those intervals exactly, so the pulses are not perfectly regular. This places a practical ceiling on how fast the motors can be driven, though for the slow and short movements this robot makes it is not restrictive.

7.2 Structure

The application is organised around a small number of components with separated responsibilities, shown in Figure 7.1.

The `Canari` class owns the hardware. Its constructor initialises everything at start-up and its destructor releases it at the end, and in between it exposes what the robot can do as methods — `bodyMove`, `headMove`, `beak_open`, `beak_close`, `tail_open`, `tail_close`. It also holds the two `HX711` objects that read the load cells, the mutexes guarding the actuators, and the thread that monitors the microphone.

The `Server` class implements the WebSocket server. It holds a reference to the `Canari` object, manages the connected client, runs the monitoring thread that polls the sensors and reports events, and dispatches incoming messages to the command interpreter. The `Client` class represents a connected client and owns its outgoing message queue together with the mutex protecting it. The `HX711` class encapsulates the load cell interface, implementing the protocol, the tare procedure and the scaling.

The stepper motor control is implemented not as classes but as two modules of free functions, `Stepper_Sensor` for the body axis and `Stepper_Tete` for the head, each holding its state in file-scope variables. The two are near-duplicates of one another, differing in the pin assignments, the SPI channel, the register values and the calibration constants, with the head module distinguishing itself by appending `2` to every identifier.

7.3 Concurrency model

The application is multi-threaded, and the division of work between threads follows from the different timing of the tasks involved.

The **main thread** runs the WebSocket service loop, processing network events and dispatching commands as they arrive.

The **monitoring thread**, launched when a client connects, polls the two load cells and the vocalisation flag about every 100 ms and pushes events to the client as they occur. It runs separately because reading a load cell blocks until the converter signals that a sample is ready, and because the polling has to continue whether or not a command is being processed.

The **microphone thread**, launched when the `Canari` object is constructed, performs the acquisition and analysis described in Section 7.7 and runs for the lifetime of the application.

Short-lived threads implement the timed actuation of the beak and the tail. When an actuator is commanded to operate for a given duration, the calling thread switches the output on and starts a thread that waits for the interval, switches it off and ends. The command therefore returns immediately, and the network loop is not held up for the duration of the gesture.

Three synchronisation mechanisms are used. A mutex protects the outgoing message queue of each client. A flag named `isPlaying` indicates that audio playback is in progress, which the monitoring thread uses to suppress perch events during a stimulus, so a bird landing while a call is being played does not trigger a second one. And a mutex guards each actuator.

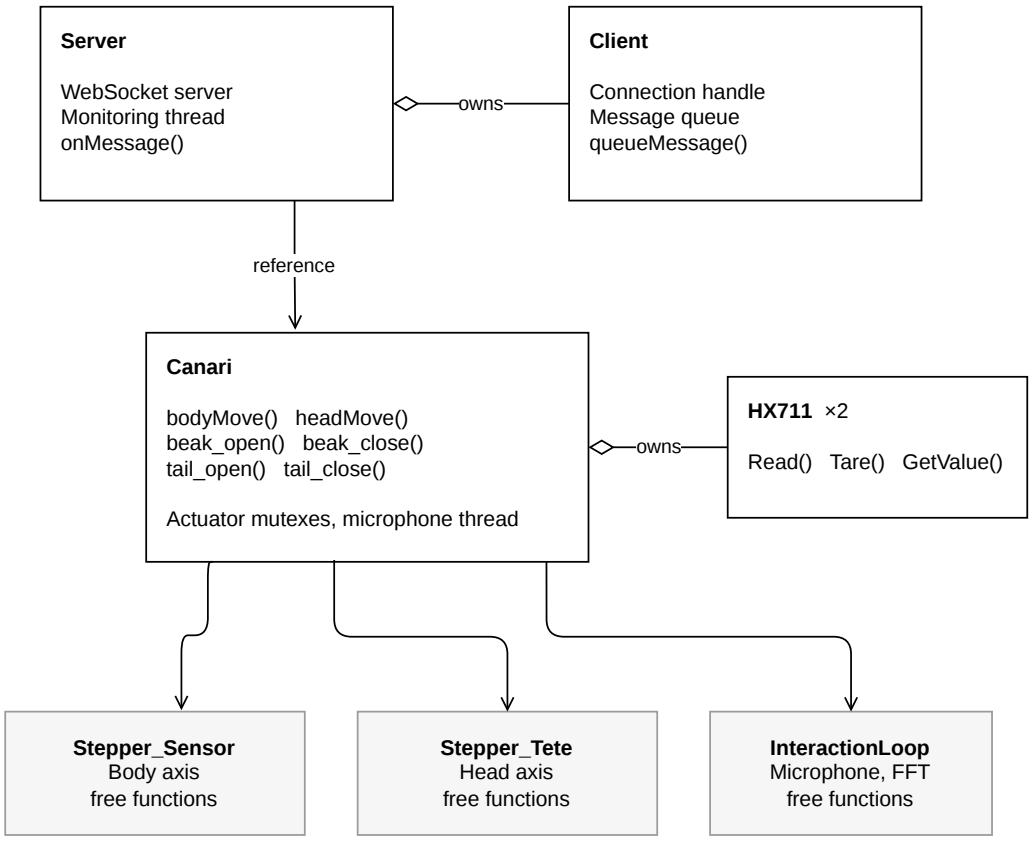


Figure 7.1: Structure of the embedded application. The `Server` owns the `Client` and holds a reference to `Canari`, which owns the two `HX711` objects and calls the two stepper modules and the microphone module, shown in grey as collections of free functions rather than classes.

The actuator mutexes are used in an unusual way, and it is worth saying why. A command arriving while the actuator is already moving is discarded rather than queued: the method checks whether the actuator is busy and returns without acting if it is. This is the right behaviour here, because the movement is tied to a stimulus. A beak that is already open should not accumulate a backlog of requests to open again, and executing one of them later would produce a movement with no relation to the song that prompted it.

7.4 Command protocol

Commands arrive as JSON objects and are dispatched through a table associating each command name with the function that carries it out. Unrecognised commands and malformed messages return an error description. Table 7.1 lists the commands available.

Table 7.1: Commands accepted by the robot over the WebSocket interface, with the argument each takes and its effect.

Command	Argument	Effect
`status`	`"ready"`	Returns the list of available birds
`bird`	Bird identifier	Selects the stimulus library and returns its songs and calls
`body`	Integer position	Turns the body to one of the discrete positions
`head`	Integer position	Turns the head to one of the discrete positions
`beak`	Duration in ms	Opens the beak, closing it after the interval
`tail`	Duration in ms	Actuates the tail for the interval
`sing`	Filename	Plays a song from the selected library
`call`	Filename	Plays a call, subject to the refractory interval
`system`	Shell command	Executes a command on the host

The last of these deserves a word. It runs an arbitrary command on the Raspberry Pi and exists as a maintenance convenience, but it also means that anything able to reach the WebSocket port can run code on the host without authenticating. On an isolated laboratory network, with one client, this is bounded; it should be removed or restricted before the robot is used on a network other systems can reach.

7.5 Motion control

Positioning is discrete. Each axis is commanded to one of a small number of fixed positions — five for the body — and the firmware works out the difference from the current position and issues the corresponding number of steps. How many steps

correspond to one position comes from a calibration constant, kept slightly below the geometric value so that the axis does not run into its mechanical limit.

The reference from which those positions are counted is established at start-up by the homing procedure. Two versions of it exist in the codebase, one of them commented out: one drives the axis against its mechanical limit, the other stops when the limit switch is triggered, requiring several consecutive readings before accepting it so that contact bounce is not mistaken for a trigger. Which of the two is in use is a choice that should be settled and recorded, since both remain in the source.

Head axis

The head is positioned in the same way, by `headTurn`, with its own calibration constant and range. Two things are specific to this axis.

The first is that the current is not constant through a movement: the routine sets a higher value for the first steps and a lower one for the remainder. The second is that the current is reduced to zero when the axis is at rest, as discussed in Section 6.3, which matters because that motor sits enclosed in the printed body where heat has nowhere to go.

Direction convention

The direction of rotation is expressed through two symbolic constants defined in the header of each stepper module, and their numerical values differ between the firmware variants present in the project archive. This is worth flagging, since the direction an axis actually turns depends jointly on those constants, on how the motor phases are wired and on how the motor is mounted — so changing one without the others inverts the behaviour.

7.6 Actuator control

The beak and the tail are controlled by the same pattern. At initialisation the direction inputs of the driver channel are set to the polarity that produces the movement, the standby input is asserted, and the PWM input is left low so the actuator starts at rest.

Actuation is performed by `beak_open(time_ms)` and its tail counterpart. The method returns without acting if the actuator is already moving, as described in Section 7.3; otherwise it switches the coil on. If a duration was given, a thread waits for that interval and then switches it off; if the duration is zero, the coil stays on until an explicit close command arrives.

`beak_open` is also called from the audio playback path, so the beak opens at the start of a stimulus and closes when playback finishes. This is how the visual and acoustic parts of the signal are kept together, and it is worth noting that the coupling is built into the playback function rather than commanded separately by the client: the synchrony is guaranteed, but the relationship between beak and sound is fixed rather than something an experiment could vary.

7.7 Vocalisation detection

The microphone allows the robot to detect when the subject has vocalised, which is what makes it possible to run experiments in which the robot responds to the song of the bird rather than only to where it is standing. The processing is contained in `InteractionLoop.cpp` and was already in place at the start of the internship.

The audio is captured through ALSA in short blocks and analysed one block at a time. For each block the software identifies the dominant frequency, using a Fourier transform, and compares it against a threshold. What distinguishes the method from a simple threshold is what happens next: the results are accumulated in a buffer covering the last few seconds, and the detection fires only when that buffer contains an isolated burst of activity surrounded by silence, which rejects both continuous noise and single transients. When it fires, a flag is raised for the monitoring thread, which reports the event to the client.

Two properties of the implementation are worth recording, since they would matter to anyone extending it. A reference value is measured once at start-up and used thereafter without being updated, so the enclosure has to be quiet at that moment and no motor may be moving. And the threshold and buffer parameters are fixed at compile time, so adapting the detection to the acoustic conditions of a particular enclosure means recompiling the application.

7.8 Client application

The client is a web application built with Next.js, developed at ETIS by Léo Coquet. The operator opens it in a browser and enters the network address of the robot to establish the connection. Figure 7.2 shows the interface.

Its functions divide into four groups. Connection management establishes and monitors the link to the robot. The manual controls expose each motion and each stimulus individually — body, head, beak, tail, and any song or call from the selected library — allowing the platform to be exercised and verified before a session begins; this is also the interface used during development and repair. The experiment planner generates a sequence of timed events, which can be reviewed and regenerated before the session is started, with the next scheduled event highlighted and the remaining time displayed during execution. And the logging records every event with a timestamp and permits export at the end of the session.

Two features of the client are worth noting, since they matter in use. The manual controls are disabled while an experiment is running, so the protocol cannot be disturbed by a command issued by mistake. And the log is backed up locally, separately from the main record, so that a crash or an accidentally closed page does not destroy the data from a session.

7.9 Software modifications

The existing software already provided the control architecture the platform needed, so the modifications made here were confined to two purposes: adding the tail, and

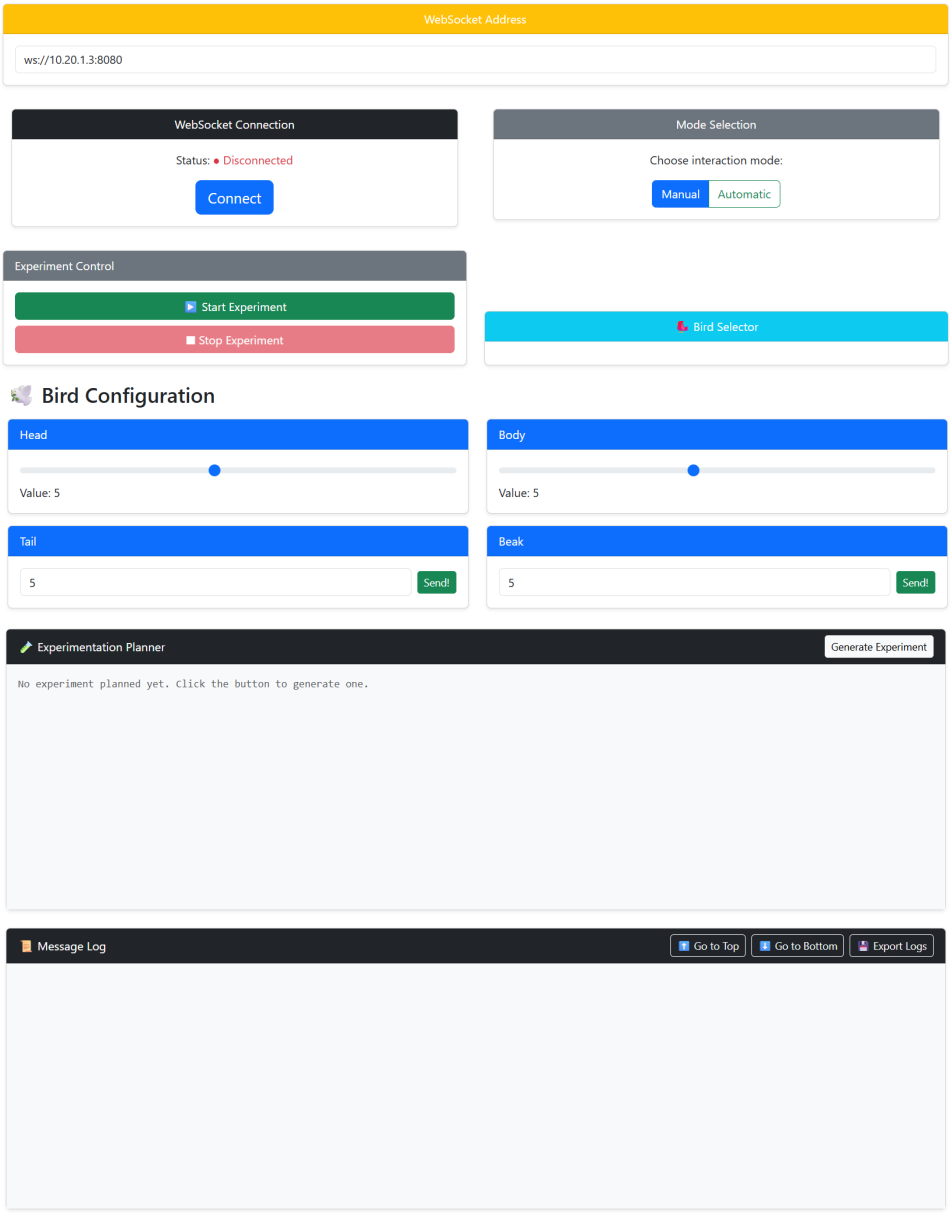


Figure 7.2: The client interface. The operator first enters the network address of the robot and selects the bird whose stimulus library is to be used; the manual controls then allow each function to be exercised individually, which is how a unit is checked before a session and during repair work; and the planner runs the timed sequence of events, with every command, stimulus, perch trigger and detected vocalisation recorded in the log below it.

correcting what came to light during the commissioning described in Section 6.9.

Tail actuation

No implementation of the tail existed. What was added is everything the beak already had: the pin definitions for the three control signals of its driver channel, an initialisation function setting the pins to their starting state, the `tail_open` and `tail_close` methods following the same pattern as the beak, their declarations in the class header, the calls to initialise and release from the constructor and destructor, and the tail entry in the command table.

Following the existing structure rather than adding a separate control path means the tail is commanded through the same protocol as every other function of the robot. The client already provided a control for it, so once the command existed on the robot the function became available from the interface with no change on that side — an illustration of the separation described in Section 3.6.

Beak control path

Two defects affecting the beak were identified and corrected.

The first concerned the pin assignment. The firmware declared the four control signals of the beak as a group, and the mapping reconstructed at the time did not agree with them, so the definitions were amended. A later comparison against the project files showed that the assignment the board implements is the one given in Table 6.3, and that the amendment had been made against an imperfect reconstruction. Since the beak was never driven from the box, neither the original declaration nor the amended one was ever exercised. The diagnosis is described in Section 6.9.

The second was in the command handler, which tested the sign of the requested duration to distinguish an open command from a close command. The condition was written against the identifier `time` rather than the local variable `time_ms`, and since `time` is a function from the standard library and in scope at that point, the expression evaluated its address rather than the intended value. The condition was therefore always satisfied and the close branch could never be reached. The comparison was corrected to test `time_ms`.

The second defect is worth recording beyond its immediate consequence: it compiles without warning and produces no visible symptom until the unreachable branch is needed, which is an argument for enabling and heeding compiler warnings in a project of this kind.

Chapter 8

Results and Discussion

This chapter assesses what the work achieved. It begins with the deliverables and objectives set out in Chapter 1, taken one at a time, then states the improvements to the robot concretely, discusses reproducibility — which was the requirement underlying most of the work — and ends with the limitations of what was done.

8.1 Assessment against the deliverables and objectives

The work is assessed below against the requests listed in Table 1.1 and the objectives of Section 1.5, in the order of their importance to the project rather than the order in which they were listed. Three outcomes are distinguished: *achieved*, where the objective was met; *partially achieved*, where the work was completed but could not be verified in full; and *not achieved*.

Production and repair of units — achieved

This was the reason for the internship and the criterion against which the work was primarily judged, since it is what the laboratory needed in order to run experiments at all.

Units were produced and delivered against the requests listed in Table 1.1, and units returned from Nanterre were repaired and put back into service throughout. Ten electronics boxes were assembled, to differing levels of completion according to what components were available. Two complete canary units were delivered against the September request for six; the three zebra finch units were not started, the internship having ended.

The delivery of the first working bird in April is worth recording separately, since it is not simply one delivery among others. It was a rough unit, assembled only after the printed parts had been corrected by hand, as described in Section 4.3 — but it was the first one built at ETIS, and building it is what revealed which features of the models needed changing. Everything that followed descends from it: the revised geometry came out of the defects it exposed, and the actuator work of Chapter 5 became possible only once the body it produced had dimensions that could be relied on.

Improvement of the bird design — achieved

The second objective, and the one that occupied most of the mechanical work, was to make the bird reproducible and more robust. It was met, and the individual improvements are described in Section 8.2.

The reliability objective, which emerged during the internship rather than being planned at the start, is part of the same result: the fixation of the head was redesigned, addressing the most frequent cause of units being returned for repair. One limitation affecting the head axis remain open and is stated in Section 4.5.

Beak and tail actuators — partially achieved

Both actuators were designed, built, fitted to the bird and shown to work; what was not done is driving them from the electronics box.

For the beak, the coil was characterised by measurement and the drive circuit sized. The coil and magnet were fitted, and connecting them directly to a 5 V supply opens the beak as intended. The firmware existed but its pin assignment did not match the board, and this was identified and amended, though the amendment could not be verified for the reasons given in Section 6.9.

For the tail, nothing existed at the start of the internship, neither mechanism nor software. Three transmission arrangements were considered and one was built and tried before settling on a simpler direct engagement between the lever and the tail, arrived at through successive printed prototypes as described in Section 4.3.3. The firmware was written in full. Connecting the coil directly to 5 V moves the tail as intended.

What remains is to command both through the software on a complete unit. This is a matter of timing rather than of design: driving them requires the revised board, and the first unit of that revision to work reliably was only available in the closing days of the internship. Both are therefore ready to be tested, and what remains to be done is described in Section 9.2.

Female variant — partially achieved

This was not part of the original scope. It was taken up during a wait for materials, as described in Section 1.5, and it is assessed on that basis: as an addition pursued in time that would otherwise have been idle, rather than as an objective with resources allocated to it.

Two things were established. The split body was designed, printed and assembled, and tilts about its pivot as intended, which shows that the kinematic arrangement works and that the division does not compromise the routing of the wiring across the joint. And the servo intended to drive it was validated on the bench, on a separate Raspberry Pi, responding correctly to the commanded positions over the full range.

Between those two there is a mechanism that does not exist. Its arrangement was worked out — the servo in the base, a wire carrying the force upward, a spring providing the return, and the attachment made internally for the reasons given in Section 5.4 — but nothing of it was modelled or built. The return spring in

particular has never been tried: whether one can be found that is stiff enough to bring the body back and soft enough for the servo to overcome is exactly the question the force analysis of that section frames without answering, since the quantities it depends on were never measured. The variant therefore stands as a validated pivot and a validated actuator, with the connection between them proposed rather than demonstrated.

Production of a batch of units — partially achieved

The objective was to renew the equipment available to Nanterre by producing units built around the revised board, and to increase the number of units so that experiments could be run in parallel.

Ten boards were assembled, but not all were completed, since component availability was the binding constraint throughout. The number of boards assembled and the number of working units available at a given moment are therefore not the same, as explained in Section 6.8.

Documentation — achieved

Several interfaces are now written down that were not before. The pinout of the harness between the box and the bird, established by continuity testing, is given in Table 6.4; the correspondence between the control signals of the coil driver and the GPIO lines of the host is given in Table 6.3, arrived at by reconstruction and then checked against the project files; and the command protocol is listed in Table 7.1.

To this can be added the corrected models themselves, which are documentation of a kind. A set of parts that fit as printed carries information that a set requiring manual adjustment does not, and that is the difference between a design which can be reproduced by someone else and one which cannot.

8.2 What was improved

The clearest way to state what this work achieved is to describe the state the bird was in at the beginning and the state it was in at the end, since almost everything else follows from that difference.

At the start, the bird could not be produced. The models were dimensionally wrong, so the first unit had to be corrected by hand during assembly: the wings were filed until they entered their mating features, the opening in the box was enlarged with a drill so that the bird could rotate freely, and the head assembly was aligned with a rotary tool. Parts printed from the current models fit as produced, and none of this is needed any longer. This is the improvement that underlies most of the others, because it is what turns a design that has to be made one unit at a time into one that can be produced in numbers.

Several specific problems were resolved in the course of that revision. The connection to the drive shaft, which previously allowed the shaft to turn without the bird following it, now engages the flat of the D-shaft so that the two are positively locked together. The bayonet fit between the base and the lid holds the bird in

place during operation while still allowing it to be removed in seconds, which matters given how often units are exchanged and repaired. And the shaft of the head motor, which did not protrude far enough for the head to seat and had therefore to be corrected on each unit individually, now allows the head assembly to be fitted directly.

The durability of the bird was addressed at the point where it actually failed. The joint between the neck disc and the motor shaft was the most frequent reason for units being returned for repair, and replacing the adhesive bond with a heat-staked joint and a compliant TPU disc, as described in Section 4.3.2, addresses that failure mode directly.

Two mechanisms were added that the robot did not previously have. The beak is driven by a coil, held in a holder on the neck disc, facing a magnet bonded to the arm of the beak inside the head; the neck disc was reshaped to make room for it. And the tail, which was purely decorative before, now has an actuator: it was lightened so that the small force available could displace it, given a recess that engages the lever of its own coil, and thinned at the bearing surface to reduce the friction absorbing that force. Each was fitted to a bird and shown to move when its coil was energised directly from a supply. Neither has been commanded through the software on a working unit, so what has been demonstrated is that the mechanisms work, not that the robot performs the behaviour.

The female variant is at an earlier stage still. The body is split, the pivot works, and the servo has been tested on its own, but the mechanism connecting them was worked out on paper and never built.

Alongside these changes, the equipment already in service was kept working. Boxes and birds returned from Nanterre during the year were repaired — components replaced, heads refitted, damaged wiring redone — and units were produced around the revised board.

8.3 Reproducibility

The laboratory needed several identical working units rather than one improved prototype, and this requirement shaped much of the work. The outcome differs between the two halves of the system.

On the mechanical side it is clear. The inherited models could not be reproduced: every bird built from them had to be corrected by hand as it was assembled, so each one cost about as much effort as the one before. The revised models produce parts that fit as printed, and this is the single most useful result of the mechanical work — not because any individual bird is better, but because building the next one no longer means solving the same problems again.

On the electronics side the picture is different. The new boards are all built to the same design, so on paper they are identical, and the assembly follows closely what was done on the earlier revision. Reproducibility is not in question there. What has to be verified is each individual board once it is assembled, since a unit that does not work may have a bad joint, a faulty component or a connection that was never made, and nothing about the design tells you which.

The difficulty was in knowing what to compare against. The reference unit belonged to an earlier PCB revision and also contained hand modifications that were not represented in the available documentation. One of these, a wire link added to the working board, was found only by physically comparing the units side by side. Because the boards belonged to different revisions, such differences could not automatically be interpreted as missing connections on the newer design.

8.4 Limitations

The most significant limitation is that the beak and the tail were never driven from the box. Both mechanisms were fitted to a bird and shown to move when energised directly, but neither was commanded through the software on a working unit, and the bowing mechanism of the female variant was never built at all. This is stated first, and separately, so that it is not left to be inferred from the individual sections.

Several limitations belong to the design rather than to what was done to it. Internal volume is the binding constraint of the bird, and every mechanical difficulty encountered reduces to it. There is no position feedback on any of the actuators: the beak and the tail are commanded but not sensed, so a mechanical failure produces no indication at all, and the head axis has no absolute reference, with the consequences described in Section 4.5. And the wiring crossing both rotation joints limits those axes to a bounded travel rather than continuous rotation.

Others belong to the circumstances the work was carried out in. Production depended on what the laboratory had in stock, and stopped when a filament or a component ran out; the schedule was set by the experimental needs of Nanterre rather than by the state of the development, so a unit was needed when it was needed whether or not the models were ready; and the board revision that supports the beak and the tail was available only in the closing stages, which is why those two subsystems reached the point they did and no further. None of these is a shortcoming of the design, but all of them bounded what could be attempted.

Two more concern the diagnostic work. The measurements were made with a hand-held multimeter, and at least one reading in the audio chain was left ambiguous by the limits of that instrument, where an oscilloscope would have settled it. And two faults were resolved without their cause being established: if either turns out to be a property of the design rather than of an individual board, it will appear again on a subsequent unit.

Chapter 9

Conclusions and Future Work

9.1 Summary

This thesis has described the improvement and production of a biomimetic bird robot used as a song tutor in the study of vocal learning in the canary, carried out at the ETIS laboratory within the ANR project nIrVAna and in response to the needs of the ethology laboratory at Université Paris Nanterre.

The main outcome is that the bird can now be produced. The models inherited at the start were dimensionally inaccurate, so the first unit had to be corrected by hand as it was assembled, and the birds in service broke easily in use. The geometry was revised region by region — the base and its interface to the drive shaft, the head and neck, the tail — so that parts printed from the current models fit as produced, and the fixation of the head was reworked to address the failure mode described in Section 4.3.2. Ten electronics boxes were assembled, to differing levels of completion; units were delivered against the requests received during the year; and units returned from Nanterre were repaired and put back into service throughout.

Two new actuator mechanisms were added, one for the beak and one for the tail. The beak is moved by a coil acting on a magnet bonded to the arm of the beak inside the head, with the drive circuit sized and the coil characterised by measurement. The tail, decorative in the inherited design, was lightened and given a recess that the lever of a second coil engages directly; neither the mechanism nor its firmware existed in any form before. Both were fitted to a bird and shown to move when energised directly, but neither was driven from the box under software control, because the board revision that supports them only became available at the very end of the internship.

A female variant was also begun, during a wait for materials rather than as part of the original scope. Its split body was built and its servo validated, but the mechanism connecting the two was worked out and never made.

A further part of the work, unplanned, went into finding out why particular boards did not function. Seven faults were identified, four of them with an established cause, and the working unit that entered service was reached by rebuilding a board rather than by resolving every fault on the one before it.

9.2 Completing the work begun here

The most useful next steps are the ones that finish what was started, and for each of them the groundwork is already in place.

The first is to **drive the beak and the tail from the box**. Both mechanisms are built and have been shown to move when energised directly, the firmware is written, and a board of the right revision is now in service, so what remains is to command them through the client and confirm that they behave as intended over the full range of durations. Two measurements would usefully accompany that verification: the temperature of the beak coil after a hundred actuations at the rate expected during a session, which would test the estimate of Section 5.2 under realistic rather than worst-case conditions, and the resistance of the tail coil, which was never measured and without which the sizing of that channel cannot be completed as it was for the beak.

The second is to **complete the female variant**, of which the split body exists and the servo has been validated, but nothing in between. Building it means deciding first where the wire and the spring attach — whether to add a hook to a wing, or to use the pin that already joins them, which sits at approximately the right height and would save a component — then modelling and printing the parts that choice requires, including the anchorage for the lower end of the spring, and fitting the wire and the spring themselves. Integrating it then needs three conductors from the box to the servo through the spare contacts of the harness connector, a free line with hardware PWM capability allocated on the board, and a control path in the firmware, which does not yet exist.

Once built, the mechanism should be characterised, which would also close the analysis left parametric in Section 5.4. The quantities needed are the mass of the rotating group and the position of its centre of mass relative to the pivot; the tension required at the wire to initiate and to sustain the rotation, which a spring gauge in line with the wire would give; the angle reached as the commanded pulse width is swept across its range; and the angle at the extremes after a hundred actuation cycles, which would reveal wear at the pivot or drift in the spring. The mass of the rotating group, at least, can be obtained from the slicer without physical access to the parts.

The third is simply to **produce the outstanding zebra finch units**, of which three remain from the September request. Models for this species already exist and the electronics are shared with the canary units, so this is a matter of printing, assembling and repairing rather than of design.

The fourth is to **bound the travel of the head axis**, which is what the open problem of Section 4.5 calls for. Pecking displaces the head on its shaft, the offsets accumulate, and the axis ends up rotating beyond the range the firmware assumes — which leaves the head in postures no live bird would adopt and twists the conductors crossing the neck until they eventually break. The body axis does not have this problem because a limit switch gives it an absolute reference at every start-up, and the same principle applied to the head would resolve both consequences at once: either a reference feature allowing the head to be re-homed, or more simply a mechanical stop against which the axis can be driven and its position reset. The

second is probably sufficient, given that what is required is a plausible posture rather than a measured angle.

9.3 Further development

Beyond finishing what was started, three directions are worth pursuing.

Position feedback on the beak and the tail. Both are commanded without being sensed, so a mechanical failure produces no indication at all. A simple contact or optical sensor would allow the firmware to confirm that a commanded gesture actually occurred, which matters in an unattended experimental session where the log is the only record of what the subject was exposed to.

Acoustic characterisation of the robot. Neither the loudspeaker nor the noise of the motors has been characterised in place, and this is worth doing, because the robot moves immediately before it sings: any noise the motors make falls exactly on the events of interest, and the design treats that noise as negligible without the assumption ever having been tested. The measurement would establish the sound pressure level and the frequency response of the loudspeaker at the position the subject occupies, and the spectrum of the noise produced by each axis at the same point. The same applies to the servo of the female variant once it is fitted.

A record of modifications made by hand. The practical conclusion of Section 6.9 belongs here as well: any modification made to a physical unit should be recorded at the time it is made, in a change log associated with the project. It costs a line of text, and it converts a difference that is otherwise undiscoverable into one that a subsequent user can find.

9.4 Experiments the work makes possible

Two experiments are closer than they were.

The paired-box comparison, described in Section 3.8, was built and delivered but has not been run. It asks whether a subject responds differently to a robot that moves and sings than to one that only sings, under acoustically identical stimulation. The question matters for the project as a whole, since it tests the premise the entire apparatus rests on, and it matters particularly for the canary, which as noted in Section 2.1 is among the species reported to learn from playback alone. The apparatus remains available.

The female variant, once built, opens a class of experiment the system could not previously support. Robotic females have been shown to modulate the behaviour of live males in another species: Patricelli et al. [39] found that male bowerbirds adjust the intensity of their courtship display according to the rate at which a robotic female crouches, and reduce it after startling her [40]. Whether something comparable holds for the canary, and whether a robot producing the solicitation display influences how a young male's song develops, are questions the mechanism proposed here would make it possible to ask.

A third direction belongs to the project rather than to this thesis. One of the long-term aims of nIrVAna is to reproduce the song-learning process in a robotic

system [3]. In that perspective, the interaction would no longer be limited to a robot tutoring a live bird: a tutor robot could provide controlled vocal and visual stimuli to a second robot acting as the learner, which would adapt its own vocal output from the interaction. The vocalisation-detection system described in Section 7.7 provides one of the elements required for such a closed-loop robot-to-robot learning experiment.

The work presented here does not complete that programme, but it provides a more reproducible and extensible physical platform on which these experiments can be built. The immediate result is a robot that can be produced and maintained more reliably; the longer-term value is that the same platform can now support increasingly interactive studies of vocal learning.

References

[1] Gilles Cornez et al. "Seasonal changes of perineuronal nets and song learning in adult canaries (*Serinus canaria*)". In: *Behavioural Brain Research* 380 (2020), p. 112437. ISSN: 0166-4328. DOI: 10.1016/j.bbr.2019.112437. URL: https://doi.org/10.1016/j.bbr.2019.112437.

[2] Sébastien Derégnaucourt. "Birdsong learning in the laboratory, with especial reference to the song of the zebra finch (*Taeniopygia guttata*)". In: *Interaction Studies* 12.2 (2011), pp. 324–350. DOI: 10.1075/is.12.2.07der. URL: https://doi.org/10.1075/is.12.2.07der.

[3] Agence Nationale de la Recherche. *Intégration sensorimotrice de la variabilité pendant l'apprentissage du chant — nIrVAna*. Project ANR-23-CE37-0025. 2023. URL: https://anr.fr/Projet-ANR-23-CE37-0025.

[4] Sébastien Derégnaucourt et al. "Comparisons of different methods to train a young zebra finch (*Taeniopygia guttata*) to learn a song". In: *Journal of Physiology-Paris* 107.3 (2013). Special issue: Physiological mechanisms of song learning and production, pp. 210–218. ISSN: 0928-4257. DOI: 10.1016/j.jphysparis.2012.08.003. URL: https://doi.org/10.1016/j.jphysparis.2012.08.003.

[5] Alice Araguas et al. "Design of a robotic zebra finch for experimental studies on developmental song learning". In: *Journal of Experimental Biology* 225.3 (Feb. 2022), jeb242949. ISSN: 0022-0949. DOI: 10.1242/jeb.242949. URL: https://doi.org/10.1242/jeb.242949.

[6] Léo Coquet. *Canaribot client*. Internal software repository, CY Cergy Paris Université. https://git.cyu.fr/canaribot/canaribot_client, access restricted to university accounts.

[7] Tim Landgraf et al. "Animal-in-the-Loop: Using Interactive Robotic Conspecifics to Study Social Behavior in Animal Groups". In: *Annual Review of Control, Robotics, and Autonomous Systems* 4.1 (2021), pp. 487–507. DOI: 10.1146/annurev-control-061920-103228. URL: https://doi.org/10.1146/annurev-control-061920-103228.

[8] Edoardo Datteri. "The Logic of Interactive Biorobotics". In: *Frontiers in Bioengineering and Biotechnology* 8 (2020), p. 637. ISSN: 2296-4185. DOI: 10.3389/fbioe.2020.00637. URL: https://doi.org/10.3389/fbioe.2020.00637.

[9] Donato Romano et al. “A Review on Animal–Robot Interaction: From Bio-Hybrid Organisms to Mixed Societies”. In: *Biological Cybernetics* 113.3 (2019), pp. 201–225. ISSN: 1432-0770. DOI: 10.1007/s00422-018-0787-5. URL: https://doi.org/10.1007/s00422-018-0787-5.

[10] Mary Kathryn Thompson et al. “Design for Additive Manufacturing: Trends, Opportunities, Considerations, and Constraints”. In: *CIRP Annals* 65.2 (2016), pp. 737–760. ISSN: 0007-8506. DOI: 10.1016/j.cirp.2016.05.004. URL: https://doi.org/10.1016/j.cirp.2016.05.004.

[11] Pololu. *SY35ST28-0504A Stepper Motor — Datasheet*. Datasheet. URL: https://www.pololu.com/file/0J688/SY35ST28-0504A.pdf.

[12] Raspberry Pi Foundation. *Raspberry Pi 3 Model B+*. Official product documentation. 2018. URL: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/.

[13] Ian Fette and Alexey Melnikov. *The WebSocket Protocol*. RFC 6455. Internet Engineering Task Force, 2011. DOI: 10.17487/RFC6455. URL: https://doi.org/10.17487/RFC6455.

[14] Christoph Klahn, Daniel Singer, and Mirko Meboldt. “Design Guidelines for Additive Manufactured Snap-Fit Joints”. In: *Procedia CIRP* 50 (2016). 26th CIRP Design Conference, pp. 264–269. ISSN: 2212-8271. DOI: 10.1016/j.procir.2016.04.130. URL: https://doi.org/10.1016/j.procir.2016.04.130.

[15] Bambu Lab. *P1S 3D Printer — Technical Specifications*. Official product documentation. URL: https://eu.store.bambulab.com/products/p1s.

[16] Bambu Lab. *P2S 3D Printer — Technical Specifications*. Official specification sheet. URL: https://store.bblcdn.com/s7/default/2d1a01cd2dca425eb071ccc28c26c9fa/spec.pdf.

[17] Bambu Lab. *H2D 3D Printer — Technical Specifications*. Official product documentation. URL: https://eu.store.bambulab.com/products/h2d.

[18] Bambu Lab. *PLA Basic Filament*. Official product documentation. URL: https://cdn1.bambulab.com/filament/filament-guide/bcv8wbl4hj/filament-guide-en.pdf.

[19] Bambu Lab. *Bambu Studio*. Slicing software, version 2.5.0.66. URL: https://bambulab.com/en/download/studio.

[20] Markus P. Nemitz et al. “Using Voice Coils to Actuate Modular Soft Robots: Wormbot, an Example”. In: *Soft Robotics* 3.4 (2016), pp. 198–204. DOI: 10.1089/soro.2016.0009. URL: https://doi.org/10.1089/soro.2016.0009.

[21] Eric Vallet and Michel Kreutzer. “Female canaries are sexually responsive to special song phrases”. In: *Animal Behaviour* 49.6 (1995), pp. 1603–1610. ISSN: 0003-3472. DOI: 10.1016/0003-3472(95)90082-9. URL: https://doi.org/10.1016/0003-3472(95)90082-9.

[22] Traco Power. *TEN 25 Series 25/30 Watt DC/DC Converters*. Datasheet, model TEN 25-1211. URL: https://www.tracopower.com/products/ten25.pdf.

[23] Trinamic Motion Control GmbH. *TMC2130 Datasheet: Stepper Driver with SPI Interface, StealthChop and SpreadCycle*. Datasheet, revision 1.16. 2023. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/TMC2130_datasheet_rev1.16.pdf.

[24] Watterott electronic GmbH. *SilentStepStick TMC2130 — Stepper Motor Driver Board*. Product documentation, revision 2.0. 2018. URL: https://learn.watterott.com/silentstepstick/.

[25] Toshiba Electronic Devices & Storage Corporation. *TB6612FNG Driver IC for Dual DC Motor*. Datasheet. 2014. URL: https://toshiba.semicon-storage.com/info/TB6612FNG_datasheet_en_20141001.pdf?did=10660&prodName=TB6612FNG.

[26] Pololu Corporation. *TB6612FNG Dual Motor Driver Carrier*. Product 713. URL: https://www.pololu.com/product/713.

[27] SparkFun Electronics. *Mini Load Cell — 100 g, Straight Bar (TAL221)*. Product SEN-14727. URL: https://www.sparkfun.com/mini-load-cell-100g-straight-bar-tal221.html.

[28] Avia Semiconductor. *HX711 24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales*. Datasheet, version 2.0. URL: https://cdn.sparkfun.com/assets/b/f/5/a/e/hx711F_EN.pdf.

[29] SparkFun Electronics. *SparkFun Load Cell Amplifier — HX711*. Product SEN-13879. URL: https://www.sparkfun.com/sparkfun-load-cell-amplifier-hx711.html.

[30] Diodes Incorporated. *PAM8302A* 2.5 W *Filterless Class-D Mono Audio Amplifier*. Datasheet, revision 6. 2021. URL: https://www.diodes.com/datasheet/download/PAM8302A.pdf.

[31] Adafruit Industries. *PAM8302 Mono* 2.5 W *Class-D Audio Amplifier*. Product 2130 and product documentation. URL: https://www.adafruit.com/product/2130.

[32] Hirose Electric. *DF11 Series* 2.0 mm *Pitch, Double-Row Connectors*. Product documentation, part DF11-16DP-2DSA. URL: https://www.hirose.com/en/product/p/CL0543-0523-8-08.

[33] Raspberry Pi Ltd. *Raspberry Pi OS*. Official operating system documentation. URL: https://www.raspberrypi.com/software/.

[34] Joan. *pigpio: a C Library for the Raspberry Pi GPIO*. Software library. URL: https://github.com/joan2937/pigpio.

[35] libwebsockets contributors. *libwebsockets: Lightweight C Networking Library*. Software library and documentation. URL: https://libwebsockets.org.

[36] Niels Lohmann. *JSON for Modern C++*. Software library. URL: https://github.com/nlohmann/json.

[37] ALSA Project. *Advanced Linux Sound Architecture*. Project documentation. URL: https://www.alsa-project.org/alsa-doc/alsa-lib/.

[38] Matteo Frigo and Steven G. Johnson. "The Design and Implementation of FFTW3". In: *Proceedings of the IEEE* 93.2 (2005), pp. 216–231. DOI: 10.1109/JPROC.2004.840301. URL: https://doi.org/10.1109/JPROC.2004.840301.

[39] Gail L. Patricelli et al. "Male displays adjusted to female's response". In: *Nature* 415 (2002), pp. 279–280. DOI: 10.1038/415279a. URL: https://doi.org/10.1038/415279a.

[40] Gail L. Patricelli, Seth W. Coleman, and Gerald Borgia. "Male satin bowerbirds, *Ptilonorhynchus violaceus*, adjust their display intensity in response to female startling: an experiment with robotic females". In: *Animal Behaviour* 71.1 (2006), pp. 49–59. ISSN: 0003-3472. DOI: 10.1016/j.anbehav.2005.03.029. URL: https://doi.org/10.1016/j.anbehav.2005.03.029.