PROJEKT-DOSSIER · DIY-HARDWARE · HOME-ASSISTANT-INTEGRATION

Pigeon Sentry *v4*

ESPHOME-HYBRID MIT PAN/TILT & HA-INTEGRATION

STANDORT · FÜRTH 90762
REV. 4.0 · 2026-05
ZIEL · STADTTAUBE

WAS SICH GEGENÜBER V3 ÄNDERT

Zwei-Chip-Architektur mit Home-Assistant-Anbindung

Audio-Erkennung bleibt auf einem ESP32-S3 mit Edge Impulse (wie v3 bewährt). Servos, Ventil und Home-Assistant-Anbindung wandern auf einen zweiten, billigen ESP32 mit ESPHome. Verbindung zwischen beiden: MQTT über deinen bestehenden HA-Stack. Damit bekommst du OTA-Updates, HA-Dashboard, Logbuch und Automatisierungen — bei nur ~10 € Mehrkosten.

- Audio-ESP32 (S3): minimaler Sketch, sendet MQTT bei Erkennung
- Actor-ESP32 (regulär, ~5 €): komplett über ESPHome-YAML konfiguriert
- Home Assistant: 6 Sensoren, 4 Buttons, Lovelace-Karte, Nacht-Automation
- OTA-Updates für beide Chips über WLAN
- Mehrkosten gegenüber v3: ~10 € · neue Gesamtsumme ~135 €

Zwei ESP32s, eine Aufgabe: das eine hört, das andere handelt. Verbunden über deinen *Home-Assistant-MQTT-Broker*. Du bekommst Echtzeit-Sensoren im Dashboard, kannst manuell auslösen, automatisiert nachts deaktivieren und das ganze System per WLAN aktualisieren — ohne USB-Kabel.

MATERIALKOSTEN
~135 €

BAUZEIT
2 Wochenenden

HA-ENTITIES
10+

OTA-UPDATES

Beide Chips

BAUPLAN / INHALT

01	Hybrid-Architektur im Überblick	10 min
	neu	
02	Stückliste & Bestellung	~135 €
	neu	
03	Hardware verkabeln	90 min
04	Mechanik & Düsenhalterung	45 min
05	Edge Impulse — Modell trainieren	90 min
06	Audio-ESP32 Firmware (MQTT-Sender)	30 min
	neu	
07	Actor-ESP32 mit ESPHome	40 min
	home assistant	
08	Home Assistant Integration	30 min
	home assistant	
09	Zonen kalibrieren	30 min
10	Wasser anschließen	30 min
11	Automationen & Dashboard	20 min
	home assistant	
12	Optional: Kamera nachrüsten	später
	neu	
13	Tuning, Tricks, Realismus	laufend

01

Hybrid-Architektur

NEU

Zwei spezialisierte Chips statt einer Eierlegenden-Wollmilchsau. Begründung: Edge Impulse (Arduino-Library mit TFLite-Micro) und ESPHome (ESP-IDF mit eigener YAML-Toolchain) lassen sich theoretisch in eine Custom Component zusammenfügen, aber das ist sehr fragil. Mit zwei Chips ist alles klar getrennt — und der zweite Chip kostet weniger als zwei Brötchen.

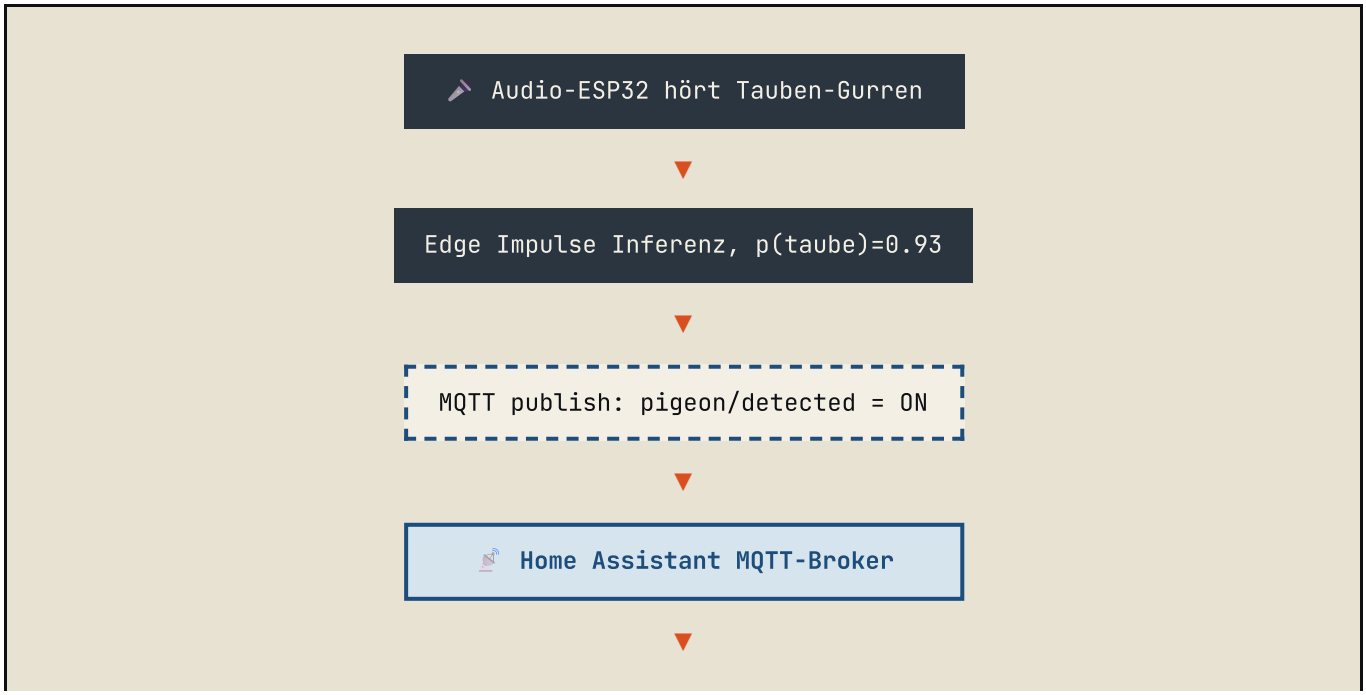
AUDIO-ESP32 (S3)

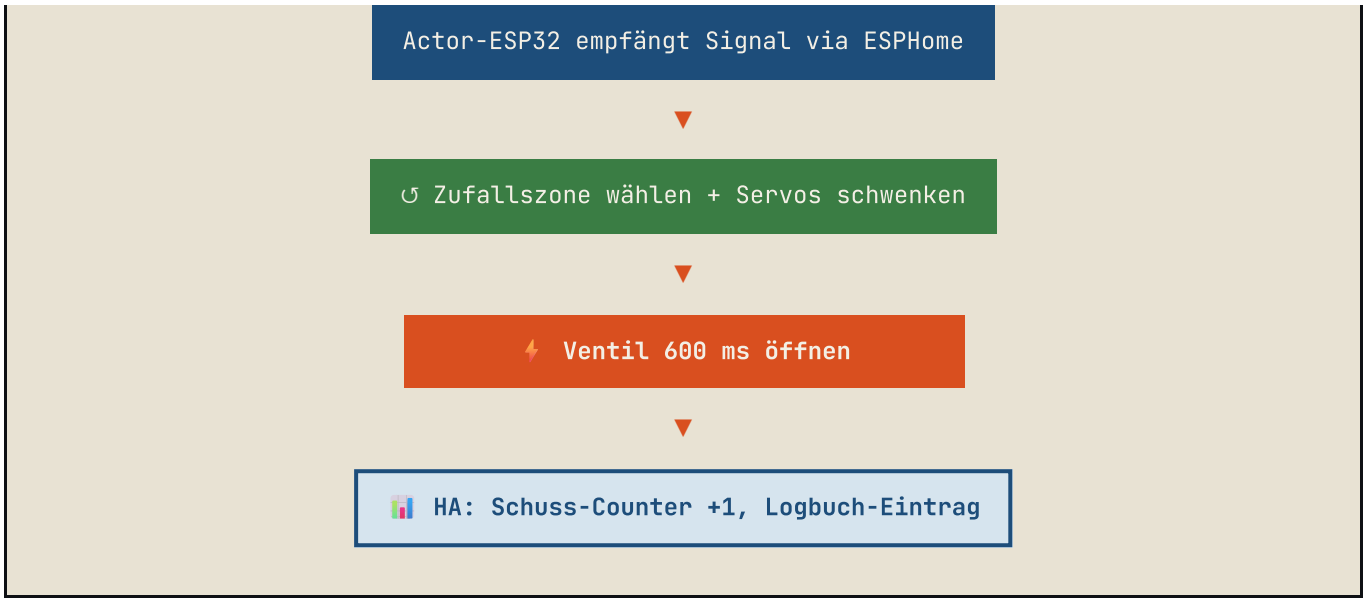
- I2S-Mikrofon INMP441
- Edge Impulse Inferenz
- WLAN + MQTT-Client
- Sendet: `pigeon/detected` + `pigeon/probability`
- Firmware: Arduino-Sketch
- OTA: ArduinoOTA-Library

ACTOR-ESP32 (REGULÄR)

- 2× Servos (Pan/Tilt)
- 1× Relais für Magnetventil
- WLAN + ESPHome
- Hört auf MQTT-Topics
- Firmware: ESPHome-YAML
- OTA: ESPHome nativ

DATENFLUSS





VORAUSSETZUNG · MQTT-BROKER Du brauchst einen MQTT-Broker in deinem HA. Wenn nicht vorhanden: HA-Settings → Add-Ons → „Mosquitto broker" installieren, einen User anlegen, MQTT-Integration unter Settings → Devices → Integrations hinzufügen. Dauert 5 Minuten. Ohne Broker funktioniert v4 nicht — fallback wäre dann v3 ohne HA-Integration.

02

Stückliste

GEÄNDERT

Ein einziger Teil kommt gegenüber v3 dazu: ein zweiter, regulärer ESP32 (kein S3 nötig, der billige tut's). Rest identisch.

ELEKTRONIK / STEUERUNG

ST.	BAUTEIL	≈ PREIS
1×	**ESP32-S3 DevKit (mit PSRAM) · Audio-Chip** Für Edge-Impulse-Inferenz. PSRAM zwingend.	15 €
1×	**ESP32 DevKit · Actor-Chip** NEU Regulärer ESP32-WROOM-32, z. B. AZ-Delivery oder NodeMCU-32S. Kein S3, kein PSRAM nötig.	8 €
1×	**INMP441 I2S MEMS-Mikrofon** Suchwort: „INMP441 omnidirectional I2S".	6 €
1×	**1-Kanal-Relaismodul 5 V mit Optokoppler** Aktiv-LOW, ≥10 A Schaltleistung.	4 €
2×	**MG996R Servo (Metallgetriebe)** Pan + Tilt. ~9 kg·cm Stallkraft.	14 €
1×	**Pan/Tilt-Halterung MG996R** Fertiger Kit oder Thingiverse-3D-Druck.	10 €
1×	**Netzteil 12 V / 2 A** Versorgt Magnetventil.	10 €
1×	**Netzteil 5 V / 3 A** Versorgt Servos + beide ESP32s.	12 €
1×	**Elektrolytkondensator 1000 µF / 16 V** Stützkondensator gegen Servo-Spannungseinbrüche.	1 €
2×	**DC-Hohlstecker-Schraubadapter** Eine pro Netzteil.	5 €
~20	**Jumper-Kabel** 40er-Pack.	4 €

WASSERTECHNIK & MECHANIK

ST.	BAUTEIL	≈ PREIS
1×	**12 V Magnetventil G½″, NC, Messing** Stromlos geschlossen.	12 €
2×	**Gardena-Schnellkupplung G½″**	8 €
1×	**Verstellbare Hartstrahldüse, Messing**	12 €
1×	**Spiralschlauch / Flex-Schlauch ½″, 2 m** Bewegungsspielraum für Pan/Tilt-Düse.	8 €
2×	**Schlauchschelle 12–22 mm**	2 €
1×	**IP65-Verteilergehäuse ~200×120×80 mm** Beide ESP32s, Relais, Klemmen.	12 €
3×	**Kabelverschraubung M16, IP68**	4 €
1×	**Schaumstoff-Windschutz Mikrofon**	3 €
1×	**Schutzgehäuse für Pan/Tilt-Einheit** Plastikbox oder Acryldome.	5 €
	Gesamt – Hybrid-Setup mit HA-Integration	~135 €

03

Hardware verkabeln

Die Verkabelung folgt der logischen Trennung der zwei Chips. Beide teilen sich Netzteile und Massen, aber kommunizieren nicht direkt miteinander — die Verbindung läuft über WLAN/MQTT.

1

Audio-ESP32 (S3) verkabeln

~15 MIN

Bekommt nur das Mikrofon und Strom. Sonst nichts.

INMP441 · VDD	Audio-ESP32 · 3V3
INMP441 · GND	Audio-ESP32 · GND
INMP441 · L/R	Audio-ESP32 · GND
INMP441 · WS	Audio-ESP32 · GPIO 15
INMP441 · SCK	Audio-ESP32 · GPIO 14
INMP441 · SD	Audio-ESP32 · GPIO 32
5-V-Netzteil (+)	Audio-ESP32 · 5V (VBUS)
5-V-Netzteil (-)	Audio-ESP32 · GND

2

Actor-ESP32 verkabeln

~25 MIN

Bekommt alle „Hände" — Servos und Ventil-Relais.

PAN-SERVO

Pan · Rot (V+)
5-V-Netzteil (+)

Pan · Braun (GND)
5-V-Netzteil (-) UND Actor-ESP32 · GND

Pan · Orange (Signal)
Actor-ESP32 · GPIO 18

TILT-SERVO

Tilt · Rot (V+)
5-V-Netzteil (+)

Tilt · Braun (GND)
5-V-Netzteil (-)

Tilt · Orange (Signal)
Actor-ESP32 · GPIO 19

STÜTZKONDENSATOR

Elko 1000 µF · (+)
5-V-Schiene

Elko 1000 µF · (-)
GND-Schiene

RELAIS & MAGNETVENTIL

Relais · VCC
Actor-ESP32 · 5V (VBUS)

Relais · GND
Actor-ESP32 · GND

Relais · IN
Actor-ESP32 · GPIO 5

12-V-Netzteil (+)

Relais · COM

Relais · NO

Magnetventil Pin 1

Magnetventil Pin 2

12-V-Netzteil (-)

STROMVERSORGUNG

5-V-Netzteil (+)

Actor-ESP32 · 5V (VBUS)

5-V-Netzteil (-)

Actor-ESP32 · GND

3

Gemeinsame Masse zwischen den Chips

~2 MIN

PFLICHT-VERBINDUNG Audio-ESP32 GND und Actor-ESP32 GND müssen mit der GND-Schiene des 5-V-Netzteils verbunden sein. Sonst sind die zwei Chips zwar via WLAN „verbunden", aber elektrisch entkoppelt — was bei gleichem Netzteil zu instabilem Verhalten führt.

Mechanik & Düsenhalterung

Unverändert gegenüber v3 — kurzfassung:

- [ ] Pan/Tilt-Halterung zusammenbauen, dabei Servos vorher per Test-Sketch zentrieren (90°/90°)
- [ ] Hartstrahldüse am Tilt-Horn fixieren, Schlauchanschluss zeigt nach hinten
- [ ] Schlauchführung: 2-m-Spiralschlauch oder PVC-Schleife für Bewegungsspielraum
- [ ] Servos in einem regenfesten Schutzgehäuse (Plastikdose oder Acryldome) unterbringen

TEST-SKETCH FÜR SERVO-ZENTRIERUNG Vor dem Aufschrauben der Servo-Hörner: kurzen Sketch laden, der beide Servos auf 90° fährt. Dann erst Hörner aufschrauben — sonst hast du asymmetrische Endlagen.

SERVO_CENTER.INO

```
#include <ESP32Servo.h>
Servo pan, tilt;
void setup() {
  pan.attach(18);
  tilt.attach(19);
  pan.write(90);
  tilt.write(90);
}
void loop() {}
```

05

Modell trainieren

Identisch zu v2/v3. Wenn du das Modell schon hast: überspringen.

1

Edge Impulse Studio einrichten

~10 MIN

- [ ] edgeimpulse.com, Projekt „Pigeon Detector", Target ESP-EYE

2

Daten sammeln, trainieren, exportieren

~75 MIN

- [ ] Klasse `pigeon_coo`: Handyaufnahmen + xeno-canto „Columba livia", 5–10 Min
- [ ] Klasse `background`: Wind, Verkehr, andere Vögel, tiefe Stimmen, 5–10 Min
- [ ] Impulse: Window 1000 ms, Increase 250 ms, MFCC, Classification Keras
- [ ] 50–100 Epochen, Ziel-Accuracy > 92 %
- [ ] Deployment → Arduino Library, Quantized int8 → ZIP herunterladen

06

Audio-ESP32 Firmware

NEU

Minimaler Sketch: lauscht, klassifiziert, sendet bei Erkennung MQTT. Keine Servos, kein Ventil — das macht der Actor. Wesentliche Pakete: **PubSubClient** (MQTT), **WiFi.h**, **ArduinoOTA** (für WLAN-Updates).

1

Libraries installieren

~10 MIN

- [ ] Arduino IDE 2.x · esp32-Board-Manager 3.x · ESP32S3 Dev Module
- [ ] PSRAM: OPI PSRAM aktivieren
- [ ] Library Manager: **PubSubClient** (von Nick O'Leary)
- [ ] Library Manager: **ArduinoOTA** (in esp32-Core enthalten)
- [ ] Sketch > Bibliothek einbinden > .ZIP-Bibliothek → Edge-Impulse-ZIP

2

Sketch einfügen, WLAN- und MQTT-Daten anpassen

~10 MIN

PIGEON_AUDIO_V4.INO

```
// Pigeon Sentry v4 - Audio-ESP32
// Hoert mit, klassifiziert, sendet MQTT an Home Assistant

#include <PigeonDetector_inferencing.h>
#include <driver/i2s.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <ArduinoOTA.h>

// ====== ZUGANGSDATEN ANPASSEN ======
const char* WIFI_SSID     = "DEIN_WLAN";
const char* WIFI_PASS     = "DEIN_PASSWORT";
const char* MQTT_HOST     = "192.168.1.10";  // IP deines HA
const int   MQTT_PORT     = 1883;
const char* MQTT_USER     = "mqttuser";
const char* MQTT_PASS     = "mqttpass";
const char* DEVICE_ID     = "pigeon_audio";
const char* OTA_PASS      = "otapass";

// MQTT-Topics
const char* T_DETECTED    = "pigeon/detected";
const char* T_PROBABILITY = "pigeon/probability";
const char* T_STATUS      = "pigeon/audio_status";
const char* T_ENABLED     = "pigeon/enabled";  // von HA gesteuert

// ====== HARDWARE ======
#define MIC_BCK_PIN   14
#define MIC_WS_PIN    15
#define MIC_DATA_PIN  32
#define LED_PIN       2

const float TRIGGER_THRESHOLD   = 0.85;
const int   CONSECUTIVE_WINDOWS = 2;
const unsigned long SEND_COOLDOWN = 3000; // nicht oefter als alle 3s

// ====== STATE ======
WiFiClient wifi;
PubSubClient mqtt(wifi);
int positiveStreak = 0;
unsigned long lastSend = 0;
bool systemEnabled = true;
static signed short sampleBuffer[EI_CLASSIFIER_RAW_SAMPLE_COUNT];

// ====== I2S ======
void initMic() {
  const i2s_config_t cfg = {
    .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX),
    .sample_rate = EI_CLASSIFIER_FREQUENCY,
    .bits_per_sample = I2S_BITS_PER_SAMPLE_16BIT,
    .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
    .communication_format = I2S_COMM_FORMAT_STAND_I2S,
    .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
    .dma_buf_count = 8, .dma_buf_len = 512,
    .use_apll = false, .tx_desc_auto_clear = false, .fixed_mclk = 0
  };
```

```
  const i2s_pin_config_t pins = {
    .bck_io_num = MIC_BCK_PIN, .ws_io_num = MIC_WS_PIN,
    .data_out_num = I2S_PIN_NO_CHANGE, .data_in_num = MIC_DATA_PIN
  };
  i2s_driver_install(I2S_NUM_0, &cfg, 0, NULL);
  i2s_set_pin(I2S_NUM_0, &pins);
  i2s_zero_dma_buffer(I2S_NUM_0);
}

int get_audio_signal(size_t offset, size_t length, float *out) {
  for (size_t i = 0; i < length; i++)
    out[i] = (float)sampleBuffer[offset + i] / 32768.0;
  return 0;
}

void readBlock() {
  size_t br;
  i2s_read(I2S_NUM_0, sampleBuffer,
           EI_CLASSIFIER_RAW_SAMPLE_COUNT * sizeof(int16_t),
           &br, portMAX_DELAY);
}

// ====== MQTT ======
void onMqtt(char* topic, byte* payload, unsigned int len) {
  String msg;
  for (unsigned int i = 0; i < len; i++) msg += (char)payload[i];
  if (String(topic) == T_ENABLED) {
    systemEnabled = (msg == "ON" || msg == "1" || msg == "true");
    Serial.print("Enabled: "); Serial.println(systemEnabled);
  }
}

void reconnectMqtt() {
  while (!mqtt.connected()) {
    if (mqtt.connect(DEVICE_ID, MQTT_USER, MQTT_PASS,
                     T_STATUS, 1, true, "offline")) {
      mqtt.publish(T_STATUS, "online", true);
      mqtt.subscribe(T_ENABLED);
    } else delay(2000);
  }
}

// ====== SETUP / LOOP ======
void setup() {
  Serial.begin(115200);
  pinMode(LED_PIN, OUTPUT);

  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() ≠ WL_CONNECTED) { delay(300); Serial.print("."); }
  Serial.println(WiFi.localIP());

  ArduinoOTA.setHostname(DEVICE_ID);
  ArduinoOTA.setPassword(OTA_PASS);
  ArduinoOTA.begin();

  mqtt.setServer(MQTT_HOST, MQTT_PORT);
```

```
  mqtt.setCallback(onMqtt);
  reconnectMqtt();

  initMic();
  Serial.println("Audio-ESP32 Armed.");
}

void loop() {
  ArduinoOTA.handle();
  if (!mqtt.connected()) reconnectMqtt();
  mqtt.loop();

  readBlock();

  signal_t signal;
  signal.total_length = EI_CLASSIFIER_RAW_SAMPLE_COUNT;
  signal.get_data = &get_audio_signal;
  ei_impulse_result_t result = { 0 };
  if (run_classifier(&signal, &result, false) ≠ EI_IMPULSE_OK) return;

  float p = 0.0;
  for (size_t i = 0; i < EI_CLASSIFIER_LABEL_COUNT; i++) {
    if (strcmp(result.classification[i].label, "pigeon_coo") == 0)
      p = result.classification[i].value;
  }

  // Wahrscheinlichkeit ueber MQTT publishen (alle 2s, retained)
  static unsigned long lastPub = 0;
  if (millis() - lastPub > 2000) {
    char buf[16];
    snprintf(buf, sizeof(buf), "%.3f", p);
    mqtt.publish(T_PROBABILITY, buf, true);
    lastPub = millis();
  }

  // Trigger-Logik
  if (p > TRIGGER_THRESHOLD && systemEnabled) {
    positiveStreak++;
    if (positiveStreak ⩾ CONSECUTIVE_WINDOWS &&
        (millis() - lastSend) > SEND_COOLDOWN) {
      mqtt.publish(T_DETECTED, "ON");
      digitalWrite(LED_PIN, HIGH);
      delay(100);
      digitalWrite(LED_PIN, LOW);
      lastSend = millis();
      positiveStreak = 0;
      Serial.println(">>> Pigeon detected, MQTT sent.");
    }
  } else {
    positiveStreak = 0;
  }
}
```

3

Erstes Flashen per USB, danach OTA

~10 MIN

- [ ] WLAN-SSID/Passwort und MQTT-Host/User/Passwort eintragen
- [ ] Per USB einmal flashen — danach ist OTA verfügbar
- [ ] Serieller Monitor: WLAN-IP, dann „Audio-ESP32 Armed."
- [ ] Taubengurren am Laptop abspielen → in HA siehst du im MQTT-Tab das Topic `pigeon/detected` auf ON springen

07

Actor-ESP32 mit ESPHome

NEU

Hier kommt der eigentliche Komfortgewinn: alles in einer YAML-Datei, OTA-Updates über das ESPHome-Dashboard, automatische Entity-Erstellung in HA.

1

ESPHome installieren (falls noch nicht)

~10 MIN

- [ ] HA-Settings → Add-Ons → ESPHome installieren
- [ ] Add-On starten, „Open Web UI"
- [ ] Neues Gerät: „Pigeon Actor" → Plattform „ESP32" → kein konkretes Board nötig
- [ ] WLAN-Zugangsdaten eintragen (für OTA)

2

YAML-Konfiguration

~15 MIN

Folgende YAML in das ESPHome-Gerät einfügen (ersetzt die Default-Konfig):

PIGEON_ACTOR.YAML

```yaml
esphome:
  name: pigeon-actor
  friendly_name: Pigeon Actor

esp32:
  board: esp32dev
  framework:
    type: arduino

# Logging und API ---------------------------------------------
logger:
  level: INFO

api:
  encryption:
    key: "GENERATE_ME_32_BYTE_BASE64_KEY"

ota:
  - platform: esphome
    password: "otapass"

wifi:
  ssid: !secret wifi_ssid
  password: !secret wifi_password
  fast_connect: true

# MQTT als zweiter Kommunikationsweg -------------------------
mqtt:
  broker: 192.168.1.10
  username: mqttuser
  password: mqttpass
  discovery: false

# Globale Variablen fuer Zustand und Counter -----------------
globals:
  - id: shot_count
    type: int
    restore_value: yes
    initial_value: '0'
  - id: current_zone
    type: int
    initial_value: '0'

# Servos -----------------------------------------------------
servo:
  - id: pan_servo
    output: pan_output
    restore: false
    auto_detach_time: 2s
  - id: tilt_servo
    output: tilt_output
    restore: false
    auto_detach_time: 2s

output:
```

```yaml
  - platform: ledc
    id: pan_output
    pin: GPIO18
    frequency: 50Hz
  - platform: ledc
    id: tilt_output
    pin: GPIO19
    frequency: 50Hz

# Ventil-Relais ------------------------------------------------
switch:
  - platform: gpio
    id: valve_relay
    pin:
      number: GPIO5
      inverted: true   # Aktiv-LOW Relais
    name: "Pigeon Valve"
    restore_mode: ALWAYS_OFF
    internal: false

# Sensoren: aktuelles p(taube) vom Audio-Chip -------------------
sensor:
  - platform: mqtt_subscribe
    id: pigeon_probability
    name: "Pigeon Probability"
    topic: pigeon/probability
    accuracy_decimals: 3

  - platform: template
    name: "Pigeon Shot Count"
    id: shot_count_sensor
    lambda: |-
      return id(shot_count);
    update_interval: 5s
    accuracy_decimals: 0

# Binary Sensor: Detected-Trigger vom Audio-Chip ----------------
binary_sensor:
  - platform: mqtt_subscribe
    name: "Pigeon Detected"
    topic: pigeon/detected
    payload_on: "ON"
    payload_off: "OFF"
    on_press:
      then:
        - script.execute: fire_water

# Schalter "System aktiviert" - in HA toggelbar -----------------
  - platform: homeassistant
    id: system_enabled
    entity_id: input_boolean.pigeon_enabled

# Buttons: Manuell auf Zone schwenken & spritzen ----------------
button:
  - platform: template
    name: "Pigeon · Fire Zone 1"
```

```yaml
    on_press:
      then:
        - lambda: 'id(current_zone) = 0;'
        - script.execute: fire_water_forced

  - platform: template
    name: "Pigeon · Fire Zone 2"
    on_press:
      then:
        - lambda: 'id(current_zone) = 1;'
        - script.execute: fire_water_forced

  - platform: template
    name: "Pigeon · Center Servos"
    on_press:
      then:
        - servo.write:
            id: pan_servo
            level: 0.0
        - servo.write:
            id: tilt_servo
            level: 0.0

  - platform: template
    name: "Pigeon · Reset Counter"
    on_press:
      then:
        - lambda: 'id(shot_count) = 0;'

# Skripte -------------------------------------------------------
script:
  # Wird vom Audio-Trigger aufgerufen, prueft System-Status
  - id: fire_water
    then:
      - if:
          condition:
            binary_sensor.is_on: system_enabled
          then:
            - script.execute: pick_zone_and_fire

  # Zonen-Auswahl per Zufall (gewichtet) + Schwenk + Schuss
  - id: pick_zone_and_fire
    then:
      - lambda: |-
          // Gewichtete Zufallsauswahl: 3,3,2,1
          int weights[] = {3, 3, 2, 1};
          int total = 9;
          int r = random(0, total);
          int acc = 0;
          for (int i = 0; i < 4; i++) {
            acc += weights[i];
            if (r < acc) { id(current_zone) = i; break; }
          }
      - script.execute: fire_water_forced

  # Schwenkt auf current_zone und gibt Wasserstoss ab
```

```
  - id: fire_water_forced
    then:
      - lambda: |-
          // Zonen-Definition (pan/tilt in Servo-Level -1.0 .. +1.0)
          // entspricht Winkel: -90 (-1.0)  ..  +90 (+1.0)
          float pan_targets[]  = {-0.33,  0.33,  0.66, 0.00};
          float tilt_targets[] = { 0.44,  0.44,  0.22, -0.06};
          int z = id(current_zone);
          if (z < 0 || z > 3) z = 0;
          id(pan_servo).write(pan_targets[z]);
          id(tilt_servo).write(tilt_targets[z]);
      - delay: 400ms
      - switch.turn_on: valve_relay
      - delay: 600ms
      - switch.turn_off: valve_relay
      - delay: 200ms
      - servo.write:
          id: pan_servo
          level: 0.0
      - servo.write:
          id: tilt_servo
          level: 0.0
      - lambda: 'id(shot_count) += 1;'
```

3

Erstes Flashen per USB

~10 MIN

- [ ] ESPHome-Dashboard → bei „pigeon-actor" auf „Install"
- [ ] „Plug into the computer running ESPHome Dashboard" wählen
- [ ] USB-Kabel an HA-Host (oder ein Rechner, der ESPHome läuft) anschließen
- [ ] Flash läuft. Nach Erfolg: WLAN-Verbindung, automatisches Erscheinen in HA

HINWEIS · ENCRYPTION-KEY GENERIEREN ESPHome zeigt beim Neuanlegen eines Gerätes automatisch einen Vorschlag für den 32-Byte-Base64-Key an. Den unter `api: encryption: key:` in der YAML einsetzen und auch in HA bei der ESPHome-Integration eintragen.

Erste Tests

~5 MIN

- [ ] HA → Devices & Services → ESPHome → pigeon-actor → Entities
- [ ] „Pigeon · Center Servos" drücken → Servos fahren auf Null
- [ ] „Pigeon · Fire Zone 1" drücken → Servo schwenkt, Ventil schaltet, Schuss zählt hoch
- [ ] Im MQTT-Tab manuell `pigeon/detected = ON` publishen → automatischer Schuss

08

Home Assistant Integration

HOME ASSISTANT

Der Actor-ESP32 erscheint automatisch in HA als ESPHome-Gerät — alle Buttons, Sensoren und Schalter werden ohne Zusatzkonfiguration angelegt. Ergänzend brauchst du nur zwei Helper.

1

Helper anlegen

~5 MIN

HA → Settings → Devices & Services → Helpers → Create Helper:

- [ ] Toggle (Input Boolean) · Name `Pigeon Enabled` · Entity-ID `input_boolean.pigeon_enabled`
- [ ] (Optional) Datetime · Name `Pigeon Last Trigger` · Entity-ID `input_datetime.pigeon_last_trigger`

2

MQTT-Sensor in configuration.yaml

~5 MIN

Damit der Online-Status des Audio-ESP32 sauber in HA erscheint, in der HA-Konfiguration ergänzen:

CONFIGURATION.YAML

```yaml
mqtt:
  binary_sensor:
    - name: "Pigeon Audio Online"
      state_topic: pigeon/audio_status
      payload_on: "online"
      payload_off: "offline"
      device_class: connectivity
```

Nach Konfig-Reload (oder HA-Neustart) erscheint die Entity `binary_sensor.pigeon_audio_online`.

3

Übersicht aller Entities

~2 MIN

ENTITY	QUELLE	BEDEUTUNG
binary_sensor.pigeon_detected	ESPHome	Aktive Erkennung
binary_sensor.pigeon_audio_online	MQTT	Audio-Chip lebt
sensor.pigeon_probability	ESPHome	Live p(taube)-Wert
sensor.pigeon_shot_count	ESPHome	Schüsse seit Reset
switch.pigeon_valve	ESPHome	Ventil-Status (read)
button.pigeon_fire_zone_1..2	ESPHome	Manueller Schuss
button.pigeon_center_servos	ESPHome	Servos auf 0/0
button.pigeon_reset_counter	ESPHome	Counter zurücksetzen
input_boolean.pigeon_enabled	Helper	System an/aus

09

Zonen kalibrieren

Die Servo-Level-Werte in der YAML (Bereich -1.0 bis +1.0 entspricht etwa -90° bis +90°) müssen für deinen Hof angepasst werden.

1

Servo-Levels finden — über HA Developer Tools

~20 MIN

HA → Developer Tools → Services → `esphome.pigeon_actor_servo_write` existiert nicht direkt, aber du kannst die ESPHome-Web-Console des Geräts öffnen und im Lambda-Feld manuell setzen. Einfacher: vorübergehend zwei **Number**-Entities in die YAML einbauen, die Pan und Tilt direkt steuern:

KALIBRIER_HELPER.YAML (TEMPORAER)

```yaml
number:
  - platform: template
    name: "Pan Calibration"
    min_value: -1.0
    max_value: 1.0
    step: 0.01
    optimistic: true
    on_value:
      then:
        - servo.write:
            id: pan_servo
            level: !lambda "return x;"

  - platform: template
    name: "Tilt Calibration"
    min_value: -1.0
    max_value: 1.0
    step: 0.01
    optimistic: true
    on_value:
      then:
        - servo.write:
            id: tilt_servo
            level: !lambda "return x;"
```

Nach Reload erscheinen zwei Slider in HA. Damit Pan/Tilt auf jede Zielzone fahren, Werte notieren, danach in das `fire_water_forced`-Lambda übertragen. Den Kalibrier-Block hinterher wieder rauskommentieren oder belassen — er stört nicht.

2

Werte in die Zone-Definition übertragen

~5 MIN

Im `fire_water_forced` -Skript die Arrays anpassen:

ZONEN_FINALISIEREN

```
float pan_targets[]  = {-0.30,  0.35,  0.65, 0.00};  // deine gemessenen Werte!
float tilt_targets[] = { 0.42,  0.45,  0.20, -0.08};
```

Die Indizes 0..3 entsprechen den Zonen 1..4. Wenn du nur 2 Zonen brauchst (gegenüber + Anbau), kürze beide Arrays auf zwei Werte und passe die `weights[]` entsprechend an.

10

Wasser anschließen

Wie in v3 — Außenhahn → Schlauch → Magnetventil (Pfeilrichtung beachten) → Flexschlauch → Pan/Tilt-Düse.

Außenhahn G¾"	Gardena-Adapter
Schlauch 1 (fest)	Magnetventil EIN (Pfeil!)
Ventil AUS	Schlauch 2 zur Pan/Tilt-Basis
Flexschlauch	Hartstrahldüse auf Tilt-Achse

TIPP · HA-GESTEUERTER TROCKENTEST Bei den ersten Tests Außenhahn zu lassen und nur den Mechanik-/MQTT-Fluss prüfen. Erst wenn HA-Buttons sauber zum Ventil-Klick und Servo-Schwenk führen: Wasser dazu schalten.

Automationen & Dashboard

HOME ASSISTANT

BEISPIEL-AUTOMATION: NACHTS DEAKTIVIEREN

Tauben gurren tagsüber. Nachts kommen falsche Geräusche durch Marder, Katzen oder Wind. Automatisch deaktivieren:

AUTOMATION.YAML

```yaml
alias: Pigeon · Nachts deaktivieren
trigger:
  - platform: sun
    event: sunset
    offset: "+00:30:00"
action:
  - service: input_boolean.turn_off
    target:
      entity_id: input_boolean.pigeon_enabled
mode: single

---

alias: Pigeon · Morgens aktivieren
trigger:
  - platform: sun
    event: sunrise
    offset: "-00:30:00"
action:
  - service: input_boolean.turn_on
    target:
      entity_id: input_boolean.pigeon_enabled
mode: single
```

BENACHRICHTIGUNG BEI VIELEN AUSLÖSUNGEN

Wenn an einem Tag mehr als z. B. 50 Schüsse fallen, ist evtl. ein False-Positive-Problem entstanden:

ALARM.YAML

```yaml
alias: Pigeon · False-Positive-Verdacht
trigger:
  - platform: numeric_state
    entity_id: sensor.pigeon_shot_count
    above: 50
condition:
  - condition: time
    after: "06:00:00"
    before: "22:00:00"
action:
  - service: notify.mobile_app_DEIN_HANDY
    data:
      title: "Pigeon Sentry"
      message: "Heute schon 50 Ausloesungen — bitte False-Positives pruefen."
```

LOVELACE-KARTE

Eine kompakte Übersicht im Dashboard:

DASHBOARD.YAML

```yaml
type: entities
title: "🕊️  Pigeon Sentry"
entities:
  - entity: input_boolean.pigeon_enabled
    name: System aktiv
  - entity: binary_sensor.pigeon_audio_online
    name: Audio-Chip
  - entity: binary_sensor.pigeon_detected
    name: Gerade erkannt?
  - entity: sensor.pigeon_probability
    name: "p(Taube)"
  - entity: sensor.pigeon_shot_count
    name: Schuss-Counter
  - type: divider
  - entity: button.pigeon_fire_zone_1
    name: Manuell Zone 1
  - entity: button.pigeon_fire_zone_2
    name: Manuell Zone 2
  - entity: button.pigeon_center_servos
    name: Servos zentrieren
  - entity: button.pigeon_reset_counter
    name: Counter reset
```

✓ BONUS · DATEN-TRENDING Mit dem History-Plot von HA oder Grafana siehst du nach 2–3 Wochen sehr schön, ob die Schuss-Frequenz abnimmt. Das ist der härteste Indikator, ob die Tauben tatsächlich seltener werden.

Optional: Kamera nachrüsten

SPÄTER

Falls du irgendwann „live sehen" willst, was im Hof passiert — z. B. um Auslösungen visuell zu verifizieren oder einfach, weil es schön ist, sich an HA-Stream zu erfreuen — gibt's zwei pragmatische Wege:

VARIANTE A · ESP32-CAM

- [ ] ESP32-CAM-Modul mit OV2640 (~8 €)
- [ ] ESPHome-Component `esp32_camera` nativ unterstützt
- [ ] Bildqualität: ausreichend für Übersicht, nicht für Detailerkennung
- [ ] Auflösung typischerweise 800×600, 5–15 fps
- [ ] Stromhungrig und WLAN-empfindlich

ESP32_CAM.YAML

```yaml
# Auf einem dritten ESP32-CAM, separate ESPHome-Konfig
esp32_camera:
  external_clock:
    pin: GPIO0
    frequency: 20MHz
  i2c_pins:
    sda: GPIO26
    scl: GPIO27
  data_pins: [GPIO5, GPIO18, GPIO19, GPIO21, GPIO36, GPIO39, GPIO34, GPIO35]
  vsync_pin: GPIO25
  href_pin: GPIO23
  pixel_clock_pin: GPIO22
  power_down_pin: GPIO32
  name: "Pigeon Cam"
  resolution: 800x600
  jpeg_quality: 10
```

VARIANTE B · ECHTE IP-KAMERA

- [ ] Reolink, Tapo, Eufy etc. — gibt's ab ~40 € mit echtem Outdoor-Schutz und vernünftiger Auflösung
- [ ] In HA via ONVIF- oder generischen RTSP-Integration einbinden
- [ ] Frigate-Add-On kann zusätzlich KI-Objekterkennung machen (Person, Vogel, Katze)
- [ ] Klar besser als ESP32-CAM, aber teurer

PRO-TIPP · TRIGGER-GETRIGGERTES SNAPSHOT Wenn du eine Kamera hast (egal welche), kann HA bei jedem `binary_sensor.pigeon_detected = on` ein Snapshot speichern. So baust du dir nach 2 Wochen ein Tauben-Foto-Archiv und kannst False Positives visuell überprüfen. Service: `camera.snapshot` in der Trigger-Action.

Tuning & Realismus

STELLSCHRAUBEN — IM CODE UND IN HA

- [ ] **TRIGGER_THRESHOLD** (Audio-Sketch): Erkennungs-Schwelle, default 0.85
- [ ] **CONSECUTIVE_WINDOWS** (Audio-Sketch): Anzahl positiver Fenster, default 2
- [ ] **SEND_COOLDOWN** (Audio-Sketch): kürzeste Pause zwischen MQTT-Sends, default 3 s
- [ ] **pan_targets / tilt_targets** (YAML): die Zielzonen — hier passt du nach Beobachtung an
- [ ] **weights[]** (YAML): wie häufig jede Zone gewählt wird
- [ ] **Burst- und Pause-Dauer** (YAML): in `fire_water_forced` die `delay`-Werte
- [ ] **System aktiv/inaktiv**: HA-Toggle ohne Code-Änderung

OTA-WORKFLOW

Beide Chips können jetzt über WLAN aktualisiert werden:

- [ ] **Audio-ESP32**: Arduino IDE → Sketch → Werkzeuge → Port → „pigeon_audio at 192.168.x.y" → Hochladen. Passwort: das in `OTA_PASS`
- [ ] **Actor-ESP32**: ESPHome-Dashboard → Install → Wireless. Geht in Sekunden

WAS DU IM HA-VERLAUF SEHEN WIRST

✓ REALISTISCH IN 4 WOCHEN Tagebuch im Schuss-Counter: Tag 1–3 vielleicht 80–150 Schüsse/Tag (alle Tauben werden überrascht). Tag 7–14 absinkend auf 30–60 (Stammgäste meiden den Hof). Ab Tag 21 idealerweise < 20/Tag (Restpopulation oder False Positives). Wenn du nach 4 Wochen immer noch 100+ pro Tag siehst: Mikrofon-Position prüfen, False-Positive-Trainingsdaten ergänzen.

⚠ KLASSISCHES PROBLEM · SONNEN-AUFHEIZUNG DES GEHÄUSES Beide ESP32s in einem schwarzen IP65-Kasten in praller Sonne werden im Sommer 70 °C+. ESP32 throttelt ab 80°C. Lösungen: helle Box, schattige Montage, oder kleines 5V-Lüfter (3 €) mit Temperatur-Sensor in HA.

WINTER-MODUS

❄ FROST-ROUTINE Vor dem ersten Frost: Außenhahn zu, HA-Button „Pigeon · Fire Zone 1" mehrmals drücken um Wasser aus Ventil und Schlauch zu entfernen, danach `input_boolean.pigeon_enabled` auf OFF, Pan/Tilt-Einheit demontieren und ins Trockene. Im Frühling: zurückbauen, Zonen evtl. neu kalibrieren.

WENN ALLES NICHT REICHT

Akustische Vergrämung mit Pan/Tilt und HA-Integration ist das technisch maximale, was du auf deinem Gebiet realistisch tun kannst. Ergänzend bleibt nur: **passive Maßnahmen direkt an deinen eigenen Bauten** (Pergola, CUBE, Brüstung) — Drähte und Spikes auf allen Sitzkandidaten. Dann hast du den Hof als Durchflug-Korridor, aber nicht als Sitzplatz.