Cat Feeder - FÎNTÎNĂ Ştefania-Maria

Introducere

- Acest proiect este un dispenser automat de mancare pentru pisici. El elibereaza mancare la ore prestabilite, dar poate permite si hranirea manuala prin butoane, analizează comportamentul pisicii, daca aceasta mananca, si afiseaza informatii despre comportamentul ei. In plus, indica vizual starea rezervorului de mancare prin LED-uri: verde, portocaliu, rosu (gol).
- Scopul proiectului este de a automatiza procesul de hranire al unei pisici si de a monitoriza consumul de mancare, oferind informatii clare despre obiceiurile alimentare ale pisicii. Datele colectate (ora, prezenta pisicii) sunt salvate pe un card SD, iar ulterior pot fi accesate si afisate intr-o pagina web locala, pentru o vizualizare usoara si centralizata a istoricului alimentar.
- Ideea a pornit de la nevoia de a hrani pisica chiar si atunci cand nu este nimeni acasa, dar si de a intelege mai bine daca aceasta mananca regulat sau nu.
- Este util pentru toti cei care au animale de companie si vor sa se asigure ca acestea sunt hranite corect, mai ales cand nu sunt acasa. Un astfel de dispozitiv ii poate ajuta in situatii in care trebuie sa plece si nu gasesc pe nimeni disponibil sa aiba grija de pisica. Acelasi lucru se aplica si in cazul meu, pentru ca am doua pisici si, de fiecare data cand plec de acasa, trebuie sa le duc la parintii mei.

Descriere generală

Placa de dezvoltare ESP32 acționează ca unitate de control principală. Aceasta gestionează toate componentele periferice, inclusiv servo motorul pentru distribuirea hranei, afișajul OLED, buzzer-ul, LED-urile de stare și butoanele de control. De asemenea, comunică cu un modul RTC (ceas în timp real) pentru programarea automată a hrănirii și poate salva evenimentele pe un card SD pentru jurnalizare.

Comunicarea cu afișajul OLED și modulul RTC se face prin protocolul I²C, utilizând pinii hardware ai ESP32. În plus, un senzor de vibrații este utilizat pentru a detecta eventualele mișcări sau lovituri ale hrănitorului, ca formă de interacțiune sau alertă.

Funcționalitate generală:

- Scopul principal al sistemului este distribuirea automată a hranei la intervale programate, cu posibilitatea hrănirii manuale la cerere. Hrănirea este realizată prin acționarea unui servo motor SG90, care eliberează o cantitate de hrană prestabilită.

Componente de control și interacțiune:

- Butonul 1 (Hrănire manuală): La apăsare, declanșează imediat ciclul de hrănire. Este utilizat și pentru testare sau completarea unei mese în afara programului.
- Butonul 2 (Reset counter): Resetează contorul care numără de câte ori a fost activat servo-ul. Acest contor este folosit pentru estimarea nivelului de hrană rămas.

- 3 LED-uri (Verde, Galben, Roșu): Indicarea nivelului de hrană pe baza contorului:
  - Verde: recipientul este plin (0–2 hrăniri).
  - Galben: nivel mediu (3–5 hrăniri).
  - Roșu: nivel scăzut (peste 6 hrăniri).
- Buzzer: Oferă feedback sonor pentru hrănire reușită.
  - Senzor de vibrații (SW-420): Detectează lovituri sau mișcări ale dispozitivului.

Alimentare:

Dispozitivul este alimentat dintr-un modul cu 2 baterii Li-Ion 18650 conectate în paralel, printr-un modul TP4056 pentru încărcare și protecție. Tensiunea este stabilizată la 5V cu ajutorul unui modul step-up/down YL-46, care alimentează ESP32 și restul componentelor.

Funcționare:

La pornire, ESP32 inițializează ceasul RTC. La fiecare interval prestabilit sau la apăsarea butonului de hrănire, servo-ul distribuie hrană. După fiecare activare, contorul se incrementează și LED-urile se actualizează. Feedback-ul este oferit atât vizual (LED), cât și auditiv (buzzer). Afișajul OLED arată starea curentă a sistemului și istoricul recent al hrănirilor.

Totodată, interfața web integrată permite monitorizarea obiceiurilor alimentare ale pisicii, oferind o experiență completă și accesibilă de la distanță.

Hardware Design

1. **Servomotor SG90**

```
 Legături:
    * Semnal (portocaliu) → GPIO 26, pin PWM hardware, potrivit pentru
controlul unui servo, nu interferează cu alte periferice
    * VCC (roșu) → VIN (5V), servo-ul necesită alimentare de 5V
    * GND (negru/maro) → GND
```

2. **Afișaj OLED 0.96” (I2C)**

```
 Legături:
    * SDA → GPIO 21, pin I2C hardware standard
    * SCL → GPIO 22, pin I2C hardware standard
    * VCC → 3.3V, tensiunea recomandată pentru modulul OLED
    * GND → GND
```

3. **RTC DS3231**

```
 Legături:
    * SDA → GPIO 21, partajat cu OLED, I2C permite mai multe dispozitive pe
```

```
aceiași pini
    * SCL → GPIO 22
    * VCC → 3.3V, compatibil cu ESP32
    * GND → GND
```

4. **Modul microSD (SPI)**

```
 Legături:
   * CS → GPIO 5, pin digital liber utilizat ca chip select
   * MOSI → GPIO 23, pin SPI hardware standard
   * MISO → GPIO 19, pin SPI hardware standard
   * SCK → GPIO 18, pin SPI hardware standard
   * VCC → 3.3V, unele module SD nu au regulator intern
   * GND → GND
```

5. **Senzor de vibrații SW-420**

```
 Legături:
   * DO → GPIO 32, pin digital liber, bun pentru citiri simple
   * VCC → 3.3V, compatibil cu ESP32
   * GND → GND
```

6. **Buzzer activ**

```
 Legături:
    * + → GPIO 27, pin digital folosit pentru semnal simplu HIGH
    * – → GND
```

7. **Butoane**

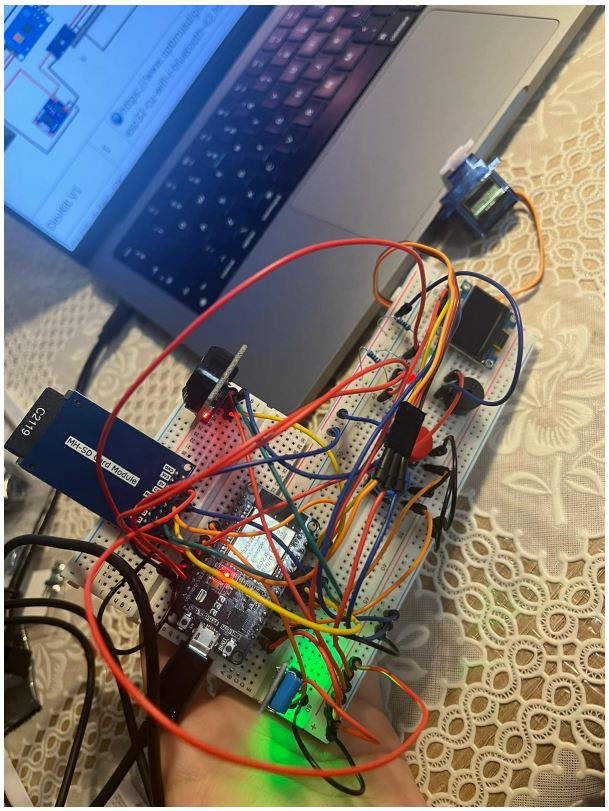
```
 Buton hrănire:
    * Un capăt → GND
    * Celălalt → GPIO 4, pin digital liber, configurat cu INPUT_PULLUP
 Buton resetare:
    * Un capăt → GND
    * Celălalt → GPIO 33, pin digital cu INPUT_PULLUP, stabil în această
configurație
 Buton log:
    * Un capăt → GND
    * Celălalt → GPIO 34, pin digital citit direct (nu suportă
INPUT_PULLUP), verificat cu gpio_get_level()
```

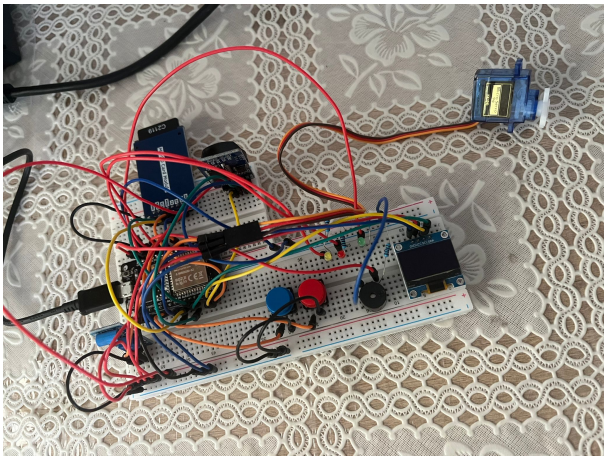
8. **LED-uri (indicatori)**

```
 Verde:
    * Anod → GPIO 12, pin digital simplu
    * Catod → GND prin rezistor
 Galben:
    * Anod → GPIO 13, pin digital simplu
    * Catod → GND prin rezistor
```

```
Roșu:
   * Anod → GPIO 14, pin digital simplu
   * Catod → GND prin rezistor
```

ESP32 DevKit V1	1	https://www.optimusdigital.ro/ro/placi-cu-bluetooth/4371-placa-de-dezvoltare-esp32-cu-wifi-i-bluetooth-42.html
Ecran OLED 0.96” I2C (128×64)	1	https://sigmanortec.ro/Display-OLED-0-96-I2C-IIC-Albastru-p135055705
Modul RTC DS3231	1	https://www.optimusdigital.ro/ro/altele/12402-modul-cu-ceas-in-timp-real-ds3231.html
Servo motor SG90	1	https://www.optimusdigital.ro/en/servomotors/2261-micro-servo-motor-sg90-180.html
Modul microSD card	1	[https://sigmanortec.ro/Modul-card-SD-p137611367]]
Buzzer activ	1	https://sigmanortec.ro/buzzer-activ-5v-arduino
LED verde 5mm	1	https://sigmanortec.ro/led-verde-5mm
LED galben 5mm	1	https://sigmanortec.ro/led-galben-5mm
LED roșu 5mm	1	https://sigmanortec.ro/led-rosu-5mm
Rezistor 220Ω	3	https://sigmanortec.ro/rezistor-220-ohmi
Rezistor 10KΩ	2	https://sigmanortec.ro/rezistor-10k-ohmi
Buton tactil (4 pini, 6x6mm)	2	https://sigmanortec.ro/buton-mini-6x6x5-4-pini
Modul senzor de vibrații SW-420	1	https://www.robofun.ro/senzor-de-vibratii-sw-420
Modul TP4056 cu protecție	1	TP4056 charger] | | Suport baterii 18650 (2x) | 1 | [[https://sigmanortec.ro/suport-2x-18650
Baterii 18650 Li-Ion	2	https://sigmanortec.ro/baterie-lithium-18650-3-7v-2600mah
Modul step-down	1	DC-DC 1.5-5V to 3.3V
Breadboard	1	https://sigmanortec.ro/breadboard-830-puncte
Fire DuPont M-M	20+	





Software Design

1. Mediul de dezvoltare

Aplicatia este dezvoltata in Arduino IDE , compatibil cu platforma ESP32 DevKit v1, utilizata in proiect. Codul este scris in limbajul C++, iar incarcarea pe placa se realizeaza prin port USB. Serial Monitor-ul este folosit pentru debug si afisarea mesajelor in timpul executiei.

2. Librarii si surse third-party utilizate

Biblioteca	Scop
Wire.h	Comunicare I2C cu RTC si display OLED
ESP32Servo.h	Controlul servo motorului prin PWM specific ESP32
RTClib.h	Gestionarea modulului RTC DS3231 (ora si data)
Adafruit_SSD1306.h	Afisare text pe display OLED monocrom
SPI.h	Comunicare SPI cu modulul microSD
SD.h	Scriere si citire fisiere pe cardul microSD
driver/gpio.h	Acces direct la registrele GPIO pentru citire/scriere eficienta

3. Algoritmi si structuri implementate

Codul foloseste o structura bazata pe evenimente (interuperi hardware) si verificari ciclice in bucla principala loop(). Bucla loop() verifica constant:

- ora curenta obtinuta de la RTC (DS3231),

- starea butoanelor (hranire manuala, reset, afisare log),
- semnalul de la senzorul de vibratii (conectat pe GPIO32).

Hrana este eliberata automat la orele 08:00, 14:00 si 18:00, pe baza valorii returnate de modulul RTC.

Hrana poate fi eliberata si manual, prin apasarea butonului conectat pe GPIO4. Acesta genereaza o intrerupere externa configurata cu attachInterrupt(), care seteaza un flag ce este tratat in loop().

Evenimentele importante sunt inregistrate pe cardul microSD, in fisierul log.txt. Acestea includ:

- hraniri automate si manuale,
- resetarea contorului de hraniri,
- detectia vibratiilor (cand pisica interactioneaza cu bolul).

Display-ul OLED conectat prin I2C afiseaza ora curenta si, temporar, mesaje precum “Contor resetat” sau ultimul eveniment logat.

LED-urile indica nivelul estimativ al rezervorului de mancare:

- Verde: 0–3 hraniri
- Galben: 4–6 hraniri
- Rosu: peste 6 hraniri

Servo motorul este controlat prin PWM pentru a elibera mancarea. Buzzer-ul este controlat prin registre (GPIO.out_w1ts / GPIO.out_w1tc) si emite un sunet scurt la fiecare hranire.

Senzorul de vibratii este verificat periodic. Daca se detecteaza vibratii (nivel LOW) si a trecut timpul de ignorare (10 secunde dupa hranire), se considera ca pisica a mancat si se logheaza evenimentul.

4. Surse si functii implementate

Functie / Structura	Descriere
`setup()`	Initializeaza componentele hardware: servo, RTC, SD, OLED, GPIO,
	LED-uri, si seteaza intreruperile
`loop()`	Executa periodic verificarile: ora curenta, butoane, senzori, etc.
`onFeedInterrupt()`	Functie ISR pentru tratarea intreruperii de la butonul de hranire
`feed(String source)`	Porneste servo-ul, suna buzzer-ul, logheaza actiunea si actualizeaza
	contorul si LED-urile
`resetFeed()`	Reseteaza contorul de hraniri, actualizeaza LED-urile si logheaza
`updateLEDs()`	Activeaza LED-ul corespunzator in functie de numarul de hraniri
`displayTime(DateTime)`	Afiseaza ora curenta pe OLED sau mesajul temporar de reset
`logSimple(String)`	Scrie o intrare in `log.txt` si verifica scrierea pe cardul SD
`showLastLog()`	Citeste si afiseaza ultima intrare din `log.txt` pe display OLED
`printSDLog()`	Afiseaza in consola toate intrarile din `log.txt`

Notiuni din laborator

1. **GPIO** (driver/gpio.h – pentru control pini digitali)

```
gpio_set_direction((gpio_num_t)BUTTON_FEED, GPIO_MODE_INPUT);
gpio_set_pull_mode((gpio_num_t)BUTTON_FEED, GPIO_PULLUP_ONLY);
```

```
gpio_set_direction((gpio_num_t)BUTTON_RESET, GPIO_MODE_INPUT);
gpio_set_pull_mode((gpio_num_t)BUTTON_RESET, GPIO_PULLUP_ONLY);
gpio_set_direction((gpio_num_t)BUTTON_LOG, GPIO_MODE_INPUT);

gpio_set_direction((gpio_num_t)BUZZER_PIN, GPIO_MODE_OUTPUT);
gpio_set_direction((gpio_num_t)VIBRATION_PIN, GPIO_MODE_INPUT);

if (!gpio_get_level((gpio_num_t)BUTTON_RESET)) {
  resetFeed();
  delay(300);
}

if (gpio_get_level((gpio_num_t)BUTTON_LOG)) {
  showLastLog();
  delay(300);
}

if (millis() > vibrationIgnoreUntil && !gpio_get_level((gpio_num_t)
VIBRATION_PIN)) {
  // log event
}
```

2. **PWM** (ESP32Servo.h – control servo motor)

```
servo.setPeriodHertz(50);
servo.attach(SERVO_PIN, 500, 2400);
servo.write(180);
delay(500);
servo.write(0);
```

3. **SPI** (SPI.h + SD.h – pentru cardul microSD)

```
#include <SPI.h>
#include <SD.h>

#define SD_CS 5

if (!SD.begin(SD_CS)) Serial.println("Eroare card SD!");

File f = SD.open("/log.txt", FILE_APPEND);
if (f) {
  f.println(logEntry);
  f.close();
}

File fRead = SD.open("/log.txt");
if (fRead) {
  while (fRead.available()) {
    lastLine = fRead.readStringUntil('\n');
  }
  fRead.close();
```

```
}
```

4. **I2C** (Wire.h – folosit implicit pentru RTC și OLED)

```
#include <Wire.h>
#include <RTClib.h>
#include <Adafruit_SSD1306.h>

RTC_DS3231 rtc;
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

if (!rtc.begin()) Serial.println("Eroare RTC!");
if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) Serial.println("Eroare
OLED!");

DateTime now = rtc.now();
display.clearDisplay();
display.setCursor(...);
display.println(...);
display.display();
```

5. **Intreruperi** (hardware interrupt pe butonul de hranire)

```
volatile bool feedInterrupt = false;

void IRAM_ATTR onFeedInterrupt() {
  feedInterrupt = true;
}

attachInterrupt(digitalPinToInterrupt(BUTTON_FEED), onFeedInterrupt, FALLING);

// in loop():
noInterrupts();
bool shouldFeed = feedInterrupt;
feedInterrupt = false;
interrupts();

if (shouldFeed) {
  feed("manuala");
}
```

Rezultate Obţinute

https://github.com/stefaniafintina/Cat-Feeder/blob/main/cat_feeder/cat_feeder.ino

Demo

Concluzii

Jurnal

07.05.2025 - Alegerea temei

07.05.2025 - Introducere si Descriere generală si Schema Bloc

15.05.2025 - Completarea secțiunii Hardware Design si Schema electrica

18.05.2025 - Scrierea codului

Bibliografie/Resurse

https://www.youtube.com/watch?v=dqr-AT5HvyM&list=LL&index=8

https://www.youtube.com/watch?v=_sKWP6fl-NU&list=LL&index=1\

https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf