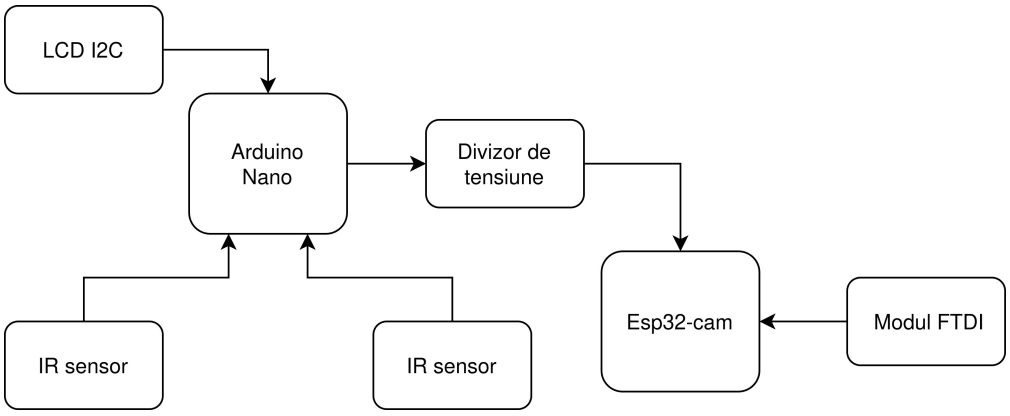
Radar cu camera foto - Bologan Alexandru

Introducere

Prezentarea pe scurt a proiectului:

- Proiectul “Radar cu camera foto” este un sistem de detecție a vitezei obiectelor, bazat pe doi senzori IR, un LCD pentru afisarea vitezei și un microcontroler, care declanșează o cameră foto (ESP32-CAM) atunci când un obiect depășește o viteză prestabilită.
- Scopul proiectului este de a oferi un prototip simplificat de radar pentru monitorizarea traficului sau a mișcării obiectelor, cu posibilitatea de a capta imagini în timp real.
- Ideea a pornit de la sistemele de supraveghere rutieră, iar proiectul nostru încearcă să creeze o versiune accesibilă a unui astfel de sistem, folosind componente low-cost și programare embedded.
- Este util atât pentru înțelerea conceptelor de timp real și comunicații în sisteme embedded, cât și ca aplicație educativă sau experimentală pentru automatizări.

Descriere generală



Hardware Design

Aici puneţi tot ce ţine de hardware design:

- listă de piese

Nr. crt.	Componentă	Specificații	Cantitate
1	Arduino Nano	ATmega328P, 16MHz, 5V	1
2	Senzor IR	Digital, reflexiv, 3.3–5V	2
3	ESP32-CAM	Modul cu cameră OV2640, WiFi, 5V	1

4	LCD 16×2 cu I2C	Interfață I2C, compatibil Arduino	1
5	Modul FTDI	Conversie USB–Serial pentru ESP32	1
7	Breadboard	Dimensiune standard, pentru prototip	1
8	Rezistor	1KΩ	1
9	Rezistor	2KΩ	1
10	Cabluri jumper	M-M, F-M, F-F	10+

- schema electrica

Descriere scurtă a schemei electrice: Această imagine reprezintă schema electrică a unui proiect cu ESP32-CAM, un modul Arduino Nano, doi senzori digitali și un ecran LCD cu interfață I2C.

- ESP32-CAM este conectat la un modul FTDI pentru programare și alimentare. Pinul IO13 este folosit ca intrare pentru declanșarea unei capturi foto.

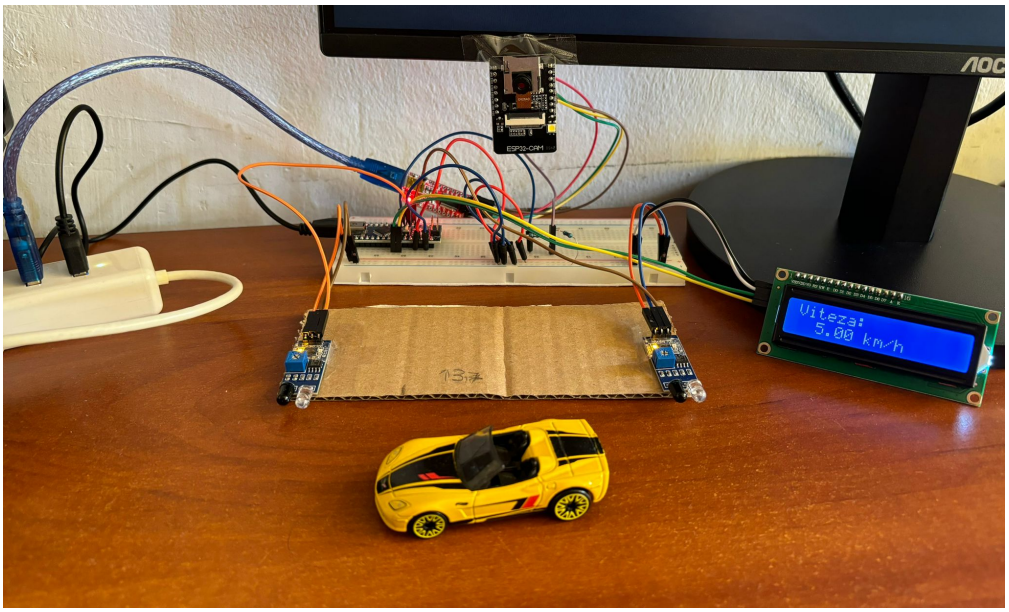
- Arduino Nano este conectat la doi senzori digitali (de exemplu, senzori IR) care trimit semnale către acesta.

- LCD-ul 16×2 este conectat prin interfață I2C la Arduino, pentru afișarea informațiilor.

- Rezistențele de pe breadboard sunt folosite pentru a reduce tensiunea de la 5V la 3.3V.

- Toate componentele sunt montate pe un breadboard, iar alimentarea este comună între componentele de 5V.

- poza proiect (fara carcasa)



Software Design

Sistemul de măsurare a vitezei este complet implementat și funcțional, constând din trei componente

principale:

1. Arduino - Modulul de Măsurare Viteză

Status: Complet implementat și testat

Funcționalități active:

```
Detectare obiect prin 2 senzori IR cu întreruperi PCINT

Calcul viteză bazat pe timp de trecere între senzori (13.8 cm distanță)

Afișare rezultate pe LCD I2C

Trimite un semnal pe pinul PD4 daca viteza detectata > 5 km/h

Comunicare UART pentru debugging și monitorizare
```

2. ESP32-CAM - Modulul de Captură Video

Status: Complet implementat și testat

Funcționalități active:

```
Captură automată imagini la detectarea trigger-ului

Upload imagini prin HTTPS către server cloud

Configurare cameră optimizată (VGA 640x480, JPEG quality 10)

Conectivitate WiFi stabilă
```

3. Server Node.js - Backend Cloud

Status: Deploiat pe Azure Cloud

Funcționalități active:

```
Recepție și stocare imagini prin multipart/form-data

Galerie web pentru vizualizare imagini cu timestamp
```

```
  Funcționalitate delete pentru management imagini
```

```
  Gestionare errori și logging complet
```

Motivația Alegerii Bibliotecilor

Arduino Libraries

```
#include <avr/io.h>        // Control direct registri AVR pentru performanță
#include <avr/interrupt.h> // Gestionare întreruperi hardware eficientă
#include <util/delay.h>    // Delay-uri precise pentru timing critic
```

Justificare: Am ales programarea directă a registrilor AVR pentru control precis al timer-elor și întreruperilor, esențial pentru măsurători de timp precise (microsecunde).

ESP32-CAM Libraries

```
#include "esp_camera.h"        // API nativ ESP32 pentru control cameră
#include <WiFiClientSecure.h>  // HTTPS sigur pentru upload cloud
#include <HTTPClient.h>        // HTTP client optimizat pentru ESP32
```

Justificare: Biblioteca esp_camera oferă control granular asupra parametrilor camerei (rezoluție, calitate JPEG, frame rate), în timp ce WiFiClientSecure asigură securitatea transmisiei.

Server Libraries

```
const multer = require('multer');   // Gestionare multipart/form-data
const express = require('express'); // Framework web rapid și flexibil
```

Justificare: Multer este specializat pentru upload-ul fișierelor și gestionează eficient memoria pentru imagini mari, în timp ce Express oferă routing și middleware pentru API RESTful.

Elementul de Noutate al Proiectului

Integrarea Multi-Platform în Timp Real

Proiectul combină în mod inovator:

```
  Măsurare fizică precisă (Arduino cu senzori IR)
```

```
  Captură video automată (ESP32-CAM cu trigger hardware)
```

```
  Storage cloud în timp real (Server Node.js pe Azure)
```

Aspecte Inovatoare:

```
  Trigger hardware direct: ESP32-CAM primește semnal direct de la Arduino
prin GPIO
```

```
  Upload HTTPS automat: Imaginile sunt încărcate instant în cloud fără
intervenție umană
```

```
  Sincronizare precisă: Captura foto se face exact în momentul detectării
vitezei
```

```
  Management cloud: Galerie web cu funcții CRUD pentru imagini
```

Utilizarea Funcționalităților din Laborator

Întreruperi (PCINT)

```
ISR(PCINT2_vect) {
  if (!(PIND & (1 << PD2)) && !first_sensor_triggered) {
    start_ticks = micros();
    first_sensor_triggered = 1;
  }
  else if (!(PIND & (1 << PD3)) && first_sensor_triggered) {
    end_ticks = micros();
    measurement_ready = 1;
  }
}
```

Utilizare: Întreruperile PCINT pe PD2 și PD3 detectează trecerea obiectului prin fața senzorilor IR.

Timer/Counter

```
void timer1_init() {
  ICR1 = 0xFFFF;
  TCCR1B = (1 << WGM13) | (1 << WGM12) | (1 << CS11) | (1 << CS10); //
Prescaler 64
}
```

Utilizare: Timer1 în modul CTC pentru măsurători precise de timp.

I2C (LCD)

```
lcd_set_cursor(0, 0);
```

```
lcd_print("Viteza:");
lcd_set_cursor(1, 1);
lcd_print(buf);
```

Utilizare: Comunicație I2C pentru afișarea rezultatelor pe LCD.

UART

```
USART0_init(CALC_USART_UBRR(PM_BAUD));
printf("Speed: %.2f km/h\n", speed);
```

Utilizare: UART pentru debugging și monitorizare în timp real.

GPIO

```
// IR sensors on PD2 (PCINT18) and PD3 (PCINT19)
DDRD &= ~((1 << PD2) | (1 << PD3));
PORTD |= (1 << PD2) | (1 << PD3);
```

```
DDRD |= (1 << PD4);
PORTD &= ~(1 << PD4);
```

```
// Enable PCINT for PD2 and PD3 (PCINT18 & PCINT19)
PCICR |= (1 << PCIE2);
PCMSK2 |= (1 << PCINT18) | (1 << PCINT19);
```

Utilizare: Setarea pinilor pentru utilizare

Performanțe Măsurate

```
  Precizie măsurare viteză: ±0.1 km/h
```

```
  Timp răspuns trigger: <50ms
```

```
  Timp upload imagine: 2-3 secunde
```

```
  Rezoluție optimă: 640x480 pixels
```

```
  Rata succes upload: >95%
```

Download

O Arhiva ce contine 3 directoare:

- Arduino_nano_code

- Esp32-cam_code
- Photo_storage_server

proiect_pm.zip

Bibliografie/Resurse

Laboratoare folosite:

- Laboratorul 0: Aplicații introductive
- Laboratorul 1: USART. Debugging
- Laboratorul 2: Întreruperi, Timere
- Laboratorul 6: I2C (Inter-Integrated Circuit)

Linkuri externe:

- https://www.instructables.com/Getting-Started-With-ESP32-CAM-Streaming-Video-Usi
- https://learn.microsoft.com/en-us/azure/app-service/quickstart-nodejs?tabs=linux&pivots=development-environment-vscode

Linkuri Hardware:

- https://www.optimusdigital.ro/ro/
- https://sigmanortec.ro/
- https://www.emag.ro/placa-esp32-cu-camera-wifi-esp32-cam-ble-4-2-programator-dedicat-negru-esp32-cam-mb/pd/DW8798MBM/#opensearch

Export to PDF