Motion Tracking Device

Marin Radu - 333CA

Introducere

Indicații pentru asistenți

Prezentarea pe scurt a proiectului vostru:

- ce face
- care este scopul lui
- care a fost ideea de la care aţi pornit
- de ce credeţi că este util pentru alţii şi pentru voi

MotionSentry

MotionSentry este un dispozitiv integrat pentru detectarea și urmărirea mișcării, bazat pe senzori infraroșu. Acesta scanează mediul înconjurător, iar la detectarea unei mișcări, orientează automat dispozitivul în direcția respectivă. Reprezintă o soluție *lightweight* de simulare a unui modul de tip *motion tracker*, utilizând un singur senzor.

Scopul proiectului este de a oferi o metodă eficientă și accesibilă pentru integrarea funcționalității de urmărire a mișcării în diverse proiecte sau dispozitive, cu un consum minim de resurse. Rezultatul constă într-un suport capabil de scanare și urmărire în timp real, ușor de integrat în alte sisteme prin simplul principiu de *clip-on*.

Ideea centrală a proiectului este construirea unei baze pentru o cameră video inteligentă, care, pe lângă o simplă mișcare de tip *sweep* pe un unghi de 180°, are și capacitatea de a se orienta automat către sursele de mișcare detectate, oferind astfel un plus de securitate. Acest concept este scalabil și poate fi aplicat oricărui dispozitiv care beneficiază de urmărirea mișcării, fără a necesita o precizie extremă.

Utilitatea sistemului derivă din simplitatea și versatilitatea sa, putând fi folosit într-o varietate de contexte. Proiectul în sine este relativ ușor de realizat din punct de vedere al resurselor și conceptelor utilizate, fiind o soluție de tip *DIY* pentru problema propusă. Exemplele prezentate sunt doar o parte din scenariile în care acest dispozitiv își poate dovedi valoarea.

Descriere generală

Indicații pentru asistenți

O schemă bloc cu toate modulele proiectului vostru, atât software cât şi hardware însoţită de o descriere a acestora precum şi a modului în care interacţionează.

Exemplu de schemă bloc: http://www.robs-projects.com/mp3proj/newplayer.html

Sistemul este compus din următoarele componente hardware și software:

Componente hardware principale:

- ESP32-WROOM - microcontroller central
- HC-SR501 PIR Sensor - senzor de detectare a mișcării
- 2x Servomotoare SG90 - pentru scanare și urmărire
- Buzzer pasiv 5V - pentru alarmă la detectarea mișcării
- 2 surse de alimentare separate (3xAA și 4xAA)

Module software:

- Scanner - controlează mișcarea de scanare a senzorului PIR
- Sentry - responsabil pentru urmărirea mișcării detectate
- PIR - gestionează detectarea mișcării prin senzorul infraroșu
- Buzzer - controlează semnalele audio
- WiFi - creează și menține serverul web pentru monitorizare

Interacțiunea componentelor:

- Senzorul PIR este montat pe primul servomotor și scanează zona
- La detectarea mișcării, al doilea servomotor orientează dispozitivul către sursa mișcării
- ESP32 gestionează comunicația între module și expune datele prin serverul web
- Buzzer-ul emite semnale sonore la detectarea mișcării

Element de noutate: implementarea unui design *lightweight*, usor de folosit si integrat in alte circuite sau proiecte hardware, functionalitatea fiind similara unui *framework* sau API din contexte orientate-software.

Hardware Design

Indicații pentru asistenți

Aici puneţi tot ce ţine de hardware design:

- listă de piese
- scheme electrice (se pot lua şi de pe Internet şi din datasheet-uri, e.g. http://www.captain.at/electronic-atmega16-mmc-schematic.png)
- diagrame de semnal
- rezultatele simulării

Componente și alimentare

Sistemul folosește două surse separate de alimentare pentru o funcționare optimă:

- 3xAA + regulator step-up 5V pentru alimentarea ESP32 și a buzzer-ului
- 4xAA direct pentru alimentarea servomotoarelor și a senzorului PIR

Această separare este esențială pentru a evita interferențele și căderile de tensiune cauzate de consumul variabil al servomotoarelor, care ar putea afecta funcționarea stabilă a microcontroller-ului.

Conectarea pinilor

Componenta	Pin ESP32	Motivație
Buzzer	D18	Pin cu capacitate PWM pentru control precis al tonurilor
Servomotor Scanner	D19	Pin cu capacitate PWM, necesar pentru controlul servomotorului
Servomotor Sentry	D22	Pin dedicat controlului mișcărilor de urmărire
Senzor PIR	D23	Pin cu suport pentru întreruperi externe, esențial pentru detectarea promptă a mișcării

Alegerea acestor pini a fost făcută ținând cont de cerințele specifice ale fiecărei componente:

- Pinii D18 și D19 sunt utilizați pentru componente care necesită semnale PWM
- Pinul D23 pentru PIR suportă întreruperi, ceea ce permite reacții rapide la mișcare
- Pinii sunt grupați pentru o organizare mai bună a cablajului

Date consum de energie

Componentă	Consum în standby (mA)	Consum în funcționare (mA)
ESP32 (WiFi activ)	80	120-150
Servomotor Scanner (în mișcare)	5	100-120
Servomotor Sentry (în mișcare)	5	100-120
Senzor PIR	50	65
Buzzer	0	30

Total	**140 mA**	**415-485 mA**

Autonomie estimată:

- Cu bateriile 3xAA (2000mAh) pentru ESP32: ~16 ore
- Cu bateriile 4xAA (2000mAh) pentru servomotoare și PIR: ~8 ore în utilizare intensivă

Software Design

Codul, împreună cu toate resursele folosite, se pot găsi pe pagina de GitHub.

Indicații pentru asistenți

Descrierea codului aplicaţiei (firmware):

- mediu de dezvoltare (if any) (e.g. AVR Studio, CodeVisionAVR)
- librării şi surse 3rd-party (e.g. Procyon AVRlib)
- algoritmi şi structuri pe care plănuiţi să le implementaţi
- (etapa 3) surse şi funcţii implementate

Mediu de dezvoltare: Visual Studio Code, folosind extensia oficială PlatformIO

Dependențe externe:

- framework-ul Arduino din PlatformIO
- madhephaestus/ESP32Servo - pentru controlul servomotoarelor prin PWM

Structura codului:

- main: punctul de pornire, inițializare și legătură între toate componentele
- buzzer/: funcții legate de acțiunea buzzer-ului
- PIR/: logica din spatele detecției mișcării
- scanner/: configurează mișcarea treptată a PIR-ului (și a servomotorului aferent)
- sentry/: implementarea urmăririi mișcării efective detectate de senzor
- wifi/: parte pseudo-izolată de restul sistemului, în care se pornește wifi-ul și se creează serverul web
- firmware_compress.py: script de compresie gzip a fișierelor firmware

Arhitectura software:

- Conform diagramei de semnal, proiectul conține 3 componente majore care activează simultan:
  - Primul servo, cel care mișcă PIR-ul
  - Al doilea servo, care urmărește mișcarea efectivă
  - Serverul web
- Toate componentele rulează pe același core FreeRTOS, fiind utilizată noțiunea de *“concurrency”* (+ sincronizările asociate)
- Ele interacționează prin intermediul unor variabile volatile de tip bool, care marchează schimbarea

stării sistemului și determină un feedback din partea componentei afectate (pe serial și fizic)

Concepte folosite:

1. GPIO - pentru controlul componentelor conectate
2. UART - pentru afisare de mesaje pe serial
3. Întreruperi - detectarea mișcării generează o întrerupere
4. Timere - mișcarea PIR-ului se face pe baza unui timer la intervale și unghiuri predefinite
5. PWM - necesar în mișcarea servomotoarelor și în acționarea asupra buzzer-ului
6. Wi-Fi - ESP32 creează un server web unde afișează momentul de timp al ultimei mișcări detectate
7. FreeRTOS - folosit pentru a separa cele 3 procese descrise anterior

Calibrarea senzorului PIR:

- Aceasta a fost realizată luând în considerare delay-ul dintre detecții
- Valoarea acestui delay a fost aleasă astfel încât să fie minimală, dar să evite totuși fals-pozitive (de exemplu, detectarea aceleiași mișcări de 2 ori, sau pornirea alarmei atunci când senzorul în sine se mișcă)
- Această calibrare se poate observa și în partea software, prin noțiunea de **debounce**

Optimizări:

Au fost realizate 3 tipuri de optimizări: ale dimensiunii firmware-ului, ale eficienței de execuție a codului și optimizări legate de *power management*.

1. Dimensiune Firmware
   - Optimizări de genul opțiunii *-Os* la compilare
   - Fișierul rezultat se trece printr-un script Python de compresare *gzip*
   - Aceste optimizări s-au realizat după sesizarea utilizării excesive ale memoriei flash de pe ESP32

1. Eficiență cod:
   - API-ul FreeRTOS pentru crearea de task-uri pe același core împarte codul în funcție de partea implementată
   - Evitarea oricăror algoritmi / structuri de date care ar putea încetini execuția (analiză tradeoff funcționalitate-overhead)

1. Power management:
   - Orice componente ESP32 care nu sunt folosite au fost dezactivate (ex: bluetooth, partiții de fișiere *SPIFFS*)
   - WiFi **nu** poate fi pus pe sleep, deoarece ar contrazice funcționalitatea proiectului în sine (trebuie să fie mereu activ log-ul)

Rezultate Obținute

Indicații pentru asistenți

Care au fost rezultatele obţinute în urma realizării proiectului vostru.

Funcționalitate principală:

- Detectarea fiabilă a mișcării pe un unghi de 180° prin scanare continuă

- Orientarea precisă spre sursa mișcării în mai puțin de 300ms
- Interfață web accesibilă pentru monitorizarea evenimentelor de mișcare în timp real
- Autonomie satisfăcătoare pentru un dispozitiv portabil

Performanță:

- Raza de detecție: până la 7 metri (conform specificațiilor senzorului PIR)
- Timp de reacție: 200-300ms de la detectarea mișcării până la orientarea completă
- Consum mediu: sub 200mA în funcționare normală (fără alarme frecvente)
- Fiabilitate crescută prin separarea surselor de alimentare

Limitări identificate:

- Sensibilitatea senzorului PIR variază în funcție de condițiile de mediu
- Servomotoarele consumă energie semnificativă în timpul operațiilor de scanare continuă
- Autonomia poate fi îmbunătățită prin optimizări suplimentare de energie
- Impulsurile primite de la senzor sunt ignorate pe perioada schimbarii unghiului de orientare

Concluzii

Proiectul **MotionSentry** demonstrează posibilitatea implementării unui sistem de urmărire a mișcării eficient și accesibil utilizând componente comune și un microcontroller ESP32. Principalele realizări includ:

- Implementarea unui algoritm de scanare și urmărire cu un singur senzor PIR
- Utilizarea eficientă a resurselor microcontroller-ului prin separarea task-urilor
- Optimizarea consumului de energie prin separarea surselor de alimentare
- Crearea unei interfețe web pentru monitorizare și control

Direcții de dezvoltare ulterioară:

- Adăugarea unui modul de cameră pentru înregistrarea mișcării detectate
- Implementarea unui sistem de notificări push pe telefonul utilizatorului
- Optimizarea suplimentară a consumului de energie pentru autonomie extinsă
- Integrarea cu alte sisteme de automatizare pentru casa inteligentă

Download

Indicații pentru asistenți

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună 😉.

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se

încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Codul sursă și toate fișierele proiectului sunt disponibile pe pagina de GitHub: GitHub Repository.

Proiectul include:

- Cod sursă complet organizat modular
- Fișier README cu instrucțiuni de instalare și utilizare
- Script de compilare și încărcare
- Scheme de conectare a componentelor

Changelog

- Confirmare temă proiect: 30.04.2025
- Comandă piese: 03.05.2025
- Documentație idee generală și componente: 04.05.2025
- Schemă bloc: 04.05.2025
- Ridicare piese: 08.05.2025
- Conectare piese la breadboard + legături între ele: 10.05.2025
- Implementare separare surse de alimentare: 12.05.2025
- Optimizare consum de energie: 14.05.2025
- Finalizare parte software: 15.05.2025
- Finalizare pagina de wiki si github: 16.05.2025

Indicații pentru asistenți

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

Indicații pentru asistenți

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

Componente hardware

Nume	Descriere	Detalii
ESP32-WROOM	Microcontroller ESP32 versiunea 2.4	Datasheet
HC-SR501 PIR Sensor	Senzor de mișcare cu infraroșu	Datasheet
SG90 Micro-Servo	Servomotor	Datasheet
5V Active Buzzer	Buzzer pasiv alimentat la 5V	Site
3xAA Battery Support	Sursă alimentare ESP32	Site

4xAA Battery Support	Sursă alimentare servomotoare și PIR	Site
5V Step-Up	Regulator voltaj	Site
Mini Breadboard	Placă de legare cu 170 de puncte	Site
Other	Condensatoare, rezistențe, fire tată-tată și tată-mamă	-

Resurse software

Nume	Descriere	Link
PlatformIO	Ecosistem open-source de dezvoltare	Site oficial
ESP32Servo	Librărie pentru controlul servomotoarelor	GitHub
FreeRTOS	Sistem de operare în timp real	Site oficial
ESP32 Arduino Core	Core Arduino pentru ESP32	GitHub

Export to PDF

From:
http://ocw.cs.pub.ro/courses/ - **CS Open CourseWare**

Permanent link:
http://ocw.cs.pub.ro/courses/pm/prj2025/cmoarcas/radu.marin0508/motionsentry

Last update: **2025/05/16 21:18**