FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Dokumentácia

ESP32: Zabezpečovací systém s detekcí pohybu

Mikroprocesorové a vestavěné systémy

20. decembra 2020 Adam Múdry (xmudry01)

Obsah

1 Úvod do problému

1.1 Zadanie

Úlohou projektu bolo zostrojiť riešenie na tému *Zabezpečovací systém s detekciou pohybu* s využitím vývojového kitu na báze **SoC** *(system-on-chip)* **ESP32** od firmy Espressif, konkrétne fotobunku aktivovanú pohybom.

1.2 Hardvér

Na riešenie som mal možnosť použiť vývojové kity **ESP-EYE**, **ESP32-S2 Saola** a **ESP32-Buddy**. Pohybový senzor je **PIR Sensor** *(passive infrared sensor)* **Parallax #555-28027**.

1.3 Softvér

Pri výbere softvéru som mal voľnú ruku a rozhodoval som sa medzi knižnicou **Nesper**, **Arduino core for the ESP32** a **ESP-IDF**.

- Knižnica Nesper je wrapper pre ESP-IDF a je napísaná v jazyku Nim, čo je staticky typovaný prekladaný programovací jazyk so syntaxom podobným Pythonu a rýchlosťou porovnateľnou s jazykom C. Výsledok jeho prekladu je v našom prípade zdrojový kód v jazyku C alebo v jazyku C++, ktorý je následne preložený zvoleným prekladačom. Toto ho robí vhodný pre programovanie aj na embedded platformách.
- Arduino core for the ESP32 je knižnica pre známe Arduino IDE, ktoré mnohí využívajú pre svoje embedded projekty. Jazyk pre Arduino projekty je C++.
- ESP-IDF je oficiálny framework od firmy Espressif písaný v jazyku C.

Na riešenie som si vybral ESP-IDF, pretože je to oficiálny framework s najlepšou podporou a najväčšou funkcionalitou. Nesper je v momente písania tejto dokumentácie stále *work-in-progress* a mnohá funkcionalita chýba alebo je pokazená a riešenie projektu pomocou Arduino core by bolo v C-style jazyku tak isto ako pri ESP-IDF.

2 Popis riešenia

Pôvodne som plánoval vytvoriť svoje riešenie pomocou jedného modulu **ESP-EYE** (ESP32 dev-board so vstavanou kamerou), ktorý by slúžil zároveň ako **senzor**, **fotobunka** a **HTTP server**, pomocou ktorého sa budú dať snímky zobraziť vo webovom prehliadači - avšak zistil som, že moje riešenie iba s jedným modulom nie je možné, pretože daný modul nemá rozhranie **GPIO** *(general-purpose input/output)*, na ktoré by som mohol PIR senzor pripojiť.

Preto som sa rozhodol na riešenie použiť 2 moduly ESP32:

- **ESP-EYE**, ktorý slúži ako fotoaparát + HTTP server a
- **ESP32-S2 Saola**, ktorý má rozhranie GPIO, na ktoré je pripojený samotný PIR senzor na detekciu pohybu.

Na týchto zariadeniach bežia 2 rôzne programy a komunikujú pomocou **MQTT** protokolu vhodného pre **IoT** zariadenia.

2.1 Využité prostriedky

Na riešenie projektu som použil už vyššie spomínaný hardvér:

- **ESP-EYE** (ESP32)
- **ESP32-S2 Saola**
- **Parallax PIR Sensor** (#555-28027)

Softvér je písaný v jazyku **C**, popričom som využil:

- **ESP-IDF** - oficiálny development framework pre ESP32 and ESP32-S série SoC zariadení
- **ESP32 Camera Driver** - knižnica pre prácu s rôznymi obrazovými snímačmi
- **ESP32 Addressable LED Strip Library** - knižnica pre prácu s RGB LED nachádzajúcou sa na ESP32-S2 Saola

Ďalší využitý softvér je **MQTT Broker** server bežiaci na PC, ktorý umožnuje komunikáciu medzi modulmi.

2.2 Detaily implementácie

2.2.1 Senzor pohybu

PIR senzor vie pracovať na širšom rozpätí napätia, tj. od 3V do 6V. Ja som využil 5V a GND (ground) GPIO pin na ESP32. Out/data pin z PIR senzora je zapojený na GPIO pine 17.

Pri štarte programu sa inicializuje NVS flash pamäť, network interface a základný event loop. Ďalej sa nastaví RGB LED a GPIO pin, na ktorom je zapojený PIR senzor. Následne sa vytvorí sa vytvorí interrupt handler pre funkciu, ktorá zaobstaráva obsluhu interruptu na nami určenom GPIO pine. Ako posledné sa modul pripojí na sieť a inicializuje MQTT funkcionalitu.

Ak PIR senzor zaznamená pohyb, spustí sa interrupt a následne naša funkcia, ktorá vyšle MQTT správu pre modul fotobuky, aby vytvorila snímku.

2.2.2 Fotobunka s HTTP serverom

Pri štarte programu sa inicializuje NVS flash pamäť, network interface a základný event loop. Ďalej sa nastaví GPIO pin pre LED a inicializuje sa kamerový systém modulu. Modul sa pripojí na sieť, inicializuje MQTT funkcionalitu a následne spustí HTTP server, ktorý servíruje JPEG snímky odfotené fotobunkou.

Ak modul dostane MQTT správu od prvého modulu so senzorom pohybu, tak modul zahodí predošlý buffer s fotografiou a vytvorí novú, ktorú následne môžeme zobraziť napr. vo webovom prehliadači, ak doň zadáme IP adresu daného modulu. Alternatívne vieme snímku získať napr. pomocou programu CURL a GET requestom na danú IP.

3 Spustenie

3.1 Potrebný softvér

- **ESP-IDF** v minimálnej verzii 4.2 a správne nastavené: návod

3.2 Postup

- V správne nastavenom termináli pre ESP-IDF sa dostaneme do zložky s programom, ktorý chceme postavať (**motion** pre pohybový senzor, **camera** pre fotobunku)
- Použijeme príkaz **idf.py menuconfig**, kde nastavíme WiFi prihlasovacie údaje a adresu MQTT brokera
- Použijeme príkaz **idf.py build**, ktorý spustí preklad a linkovanie a vytvorí binárny súbor
- Použijeme príkaz **idf.py -p PORT flash**, ktorý zapíše binárny súbor do pamäte ESP32
- Ak chceme vidieť debugovacie správy na štandardnom výstupe v termináli, použijeme príkaz **idf.py -p PORT monitor**
- Zadáme IP adresu fotobunky do prehliadača a zobrazí sa nám posledná odfotená snímka
- Nová snímka vznikne vtedy, ak spustíme PIR senzor na druhom module a ten vyšle MQTT príkaz pre fotobunku. Zobrazí sa po novom načítaní web stránky.

4 Záver

4.1 Nedostatky

Program pre pohybový senzor obsahuje bug, ktorý je zrejme v samotnom ESP-IDF. Ak sa odosiela MQTT správa mimo MQTT handlera, tak jej odoslanie trvá medzi instantným odoslaním až pár sekundovým spomalením. Pri monitorovaní programu pomocou **idf.py monitor** môžeme vidieť túto chybu ako opakované výpisy na štandardnom výstupe:

```
...
I (13540) phy: pll_cap_ext 10
I (13550) phy: pll_cap_ext 10
I (13560) phy: pll_cap_ext 10
I (13560) phy: pll_cap_ext 10
I (13570) phy: pll_cap_ext 10
I (13570) phy: pll_cap_ext 10
...
I (15260) phy: pll_cap_ext 10
I (15320) Motion sensor: MQTT_EVENT_PUBLISHED, msg_id=58815
```

4.2 Zhodnotenie

Riešenie projektu hodnotím ako úspešné. Trochu som sa oddialil od pôvodného zadania, avšak mal som na to valídny dôvod. Podarilo sa mi spraviť všetko tak, ako som chcel a aj keď som narazil na chybu, je to chyba ktorú nemôžem vyriešiť a k tomu funkcionalitu projektu obmedzuje minimálne.

Odkaz na video s demonštráciou (obhajoba): FIT Nextcloud

5 Bibliografia

Literatúra

[1] ESP-IDF Programming Guide. [online], [vid. 2020-12-15].
URL https://docs.espressif.com/projects/esp-idf/en/latest/esp32

[2] ESP-IDF Github + examples. [online], [vid. 2020-12-15].
URL https://github.com/espressif/esp-idf

[3] ESP32 Camera driver. [online], [vid. 2020-12-15].
URL https://github.com/espressif/esp32-camera

[4] RGB LED library. [online], [vid. 2020-12-15].
URL https://github.com/Lucas-Bruder/ESP32_LED_STRIP