IMP Projekt: Dekodér Morseovej abecedy*

Dokumentácia

Jakub Duda, xdudaj02

* Pôvodné zadanie je *M - ESP32: Měření teploty*, ale z dôvodu problému súvisiaceho s potrebnými súčiastkami nastala zmena zadania.

1. Úvod

Pôvodne som sa mal venovať inému zadaniu, no z dôvodu stratenia dôležitej súčiastky, som po konzultácii s vedúcim dostal za úlohu nasledovné improvizované zadanie.

Mojou úlohou bolo vytvoriť dekodér Morseovej abecedy, ktorý zaznamenáva stlačenia tlačidla predstavujúce symboly Morseovej abecedy, vyhodnocuje ich na základe ich dĺžky, dekóduje ich do klasickej abecedy a vypisuje.

2. Použité nástroje

2.1. Hardwarové nástroje

Medzi využité hardwarové nástroje patrí: programovateľná doska Wemos D1 R32 s procesorom ESP32, ktorá je pomocou USB kábla pripojená do počítača a dva káblіky.

Na doske je jeden káblik pripojený na zem (GND) a druhý je pripojený na port (GPIO5). Spájaním týchto káblikov je simulovaná funkcionalita tlačidla.

2.2. Softwarové nástroje

Medzi využité softwarové nástroje patrí: editor Visual Studio Code a jeho rozšírenie PlatformIO. Program je napísaný v jazyku C s využitím frameworku ESP-IDF. Program používa základné knižnicu stdio.h a sys/time.h, a tiež nasledujúce knižnice potrebné pre prácu s ESP32: freertos/FreeRTOS.h, freertos/task.h, freertos/queue.h, driver/gpio.h, esp_system.h, nvs_flash.h.

3. Implementácia

3.1. Konfigurácia

Hlavná konfigurácia sa nachádza v súbore platformio.ini. V tomto súbore sa nachádzajú nastavenia týkajúce sa typu pripojeného zariadenia, použitého frameworku a tzv. baud rate.

Zvyšná konfigurácia je riešená vo funkcii gpio_init(). V tejto funkcií sa nachádzajú nastavenia všetkých používaných komponentov a dátových štruktúr.

Najprv je nastavený výstupný port pre LED-ku, ktorá signalizuje akcie tlačidla. Pre tieto účely je nastavený GPIO pin číslo 2, určený v konštante GPIO_LED_RED. Tento pin je nastavený ako výstupný.

Nasleduje nastavenie portu pre prijímanie symbolov Morseovej abecedy. Pre tieto účely je nastavený GPIO pin číslo 5, určený v konštante GPIO_INPUT. Tento pin je nastavený ako vstupný pin a je doňho pripojený jeden zo spomínaných káblikov. Keďže druhý z týchto káblikov je pripojený na

zem, pri spojení káblikov bude na vstupnom porte 0. Aby bolo možné rozlíšiť medzi spojenými a rozpojenými káblikmi, potrebujem, aby pri rozpojení káblikov bola na tomto pine logická 1. Toto ide jednoducho dosiahnuť, pripojením tzv. pull up rezistoru na tento pin. Ďalej potrebujem byť schopný reagovať na zmenu hodnoty na tomto pine, preto na ňom potrebujem povoliť prerušenia a keďže ma zaujíma zmena každým smerom, prerušenia sú nastavené na hocijakú hranu (nástupná alebo klesajúca).

Ďalej nasledujú nastavenia prerušení, inicializácia flash pamäte a inicializovanie fronty udalostí. Fronta udalostí je použitá na pohodlné zabezpečenie ošetrenia všetkých prerušení, ktoré sú do tejto fronty postupne pridávané.

Konfigurácia je zakončená prepnutím LED-ky do správneho stavu a inicializáciu bufferu na ukladanie dekódovaných písmen.

3.2. Implementácia

V hlavnej funkcii app_main() sa nenachádza veľa funckionality, obsahuje len volanie inicializačnej funkcie gpio_init a čakajúcu slučku. Veľká väčšina funkcionality je implementovaná vo funkcií gpio_task(). Táto funkcia je volaná z obsluhy prerušenia.

Funkcia gpio_task() obsahuje nekonečnú slučku, v ktorej sa pomocou funkcie xQueueReceive() kontroluje existencia udalosti (prerušenia) vo fronte. Táto funkcia má nastavený 1500 ms timeout, po ktorého vypršaní program ukončí čítanie symbolov Morseovej abecedy tvoriaci daný znak klasickej abecedy a začne čakať na symboly tvoriace ďalší znak klasickej abecedy.

V prípade existencie udalosti vo fronte, program vstúpi do tejto vetvy a získa čas údaj v danej chvíli. Následne pomocou konštrukcie if odfiltruje udalosti, ktoré nastali do 200 ms od poslednej akcie. Týmto spôsobom je riešený problém kmitania hodnoty na pine pri zmene stavu. Čas poslednej akcie je uložený v špeciálnej premennej, ktorej je na adekvátnych miestach programu aktualizovaná. Nasleduje menší delay, ktorý zabezpečí aby kvôli kmitaniu hodnôt bezprostredne po stlačení tlačidla nebol načítaný nesprávny aktuálny stav.

Následne je nastavená hodnota na výstupnom pine určujúcom stav LED-ky. Táto hodnota je nastavená ako opačná k hodnote na vstupnom pine, do ktorého je zapojené tlačidlo.

Ďalej nasledujú činnosti spojené s dekódovaním vstupného kódu. V prípade, že bola zaznamenaná stúpajúca hrana, a teda nastalo uvoľnenie tlačidla, program kontroluje ubehnutý čas od poslednej akcie, a teda stlačenia tlačidla. Na základe trvania stlačenia tlačidla, program rozlíši dva rôzne znaky Morseovej abecedy, krátky a dlhý. Minimálna dĺžka stlačenia pre dlhý znak je 700 ms. Program pomocou UART-u vypíše daný znak do konzoly a taktiež ho uloží do pripraveného bufferu a upraví pomocné premenné.

V prípade klesajúcej hrany, ktorá značí stlačenie tlačidla, prebehne kontrola stavu programu. Ak sa nachádza v stave „medzi písmenami“, teda dokončil čítanie jedného znaku klasickej abecedy a čaká na symboly pre ďalší znak klasickej abecedy, skontroluje sa vypršanie timeoutu nastaveného na 4000 ms. Tento timeout značí koniec slova, a teda program uloží do bufferu znak medzery.

Ak sa stane, že vyprší timeout značiaci koniec čítania symbolov tvoriacich jedno písmeno, program skontroluje, či boli načítané nejaké symboly a v kladnom prípade vypíše a uloží do bufferu dekódované písmeno. Ak sa stane, že kombinácia prijatých symbolov nie je validná pre žiadne písmeno klasickej abecedy, program vypíše informáciu o nevalidnosti danej kombinácii symbolov a do bufferu neuloží

nič. Dekódovaný znak je získaný pomocou adekvátnej zmeny indexu do preddefinovaného poľa obsahujúceho znaky klasickej abecedy na istých indexoch, vždy pri prečítaní symbolu. Index sa zvýši o jedna pri prečítaní krátkeho znaku a predom definovanú hodnotu v prípade dlhého znaku. Táto hodnota je uložená v poli a závisí na počte prečítaných symbolov pre daný znak klasickej abecedy. Môže nadobudnúť hodnotu 16 pre prvý načítaný symbol, 8 pre druhý, respektíve 4 alebo 2 pre tretí alebo štvrtý.

Ak vypršal timeout značiaci koniec čítania symbolov tvoriacich jedno písmeno a taktiež vypršal timeout značiaci koniec čítania symbolov tvoriacich jednu sekvenciu, ktorý je nastavený na 6000 ms, táto sekvencia sa z bufferu vypíše na konzolu. Program pokračuje čakaním na ďalšie vstupy.

4. Záver

Program funguje spoľahlivo a ponúka užívateľsky príjemný výstup. Nedostatočná funkcionalita programu môže byť pre používateľa citeľná pri záujme o zmenu použitých hodnôt timeoutov. Túto funkcionalitu som neimplementoval, pretože sa mi nepodarilo spojazdniť odosielanie dát do zariadenia.

5. Video-dokumentácia

Kratšie video, obsahuje chybný znak: https://www.youtube.com/watch?v=aUSh880a0ok
Dlhšie video, obsahuje iba platné znaky: https://www.youtube.com/watch?v=rPoshDfK-_k

6. Bibliografia

1. ESP-IDF Programming Guide. (19.12.2021). docs.espressif.com. *https://docs.espressif.com/projects/esp-idf/en/latest/esp32/index.html*
2. Example ESP-IDF projects. (19.12.2021). www.github.com/espressif. *https://github.com/espressif/esp-idf/tree/master/examples*