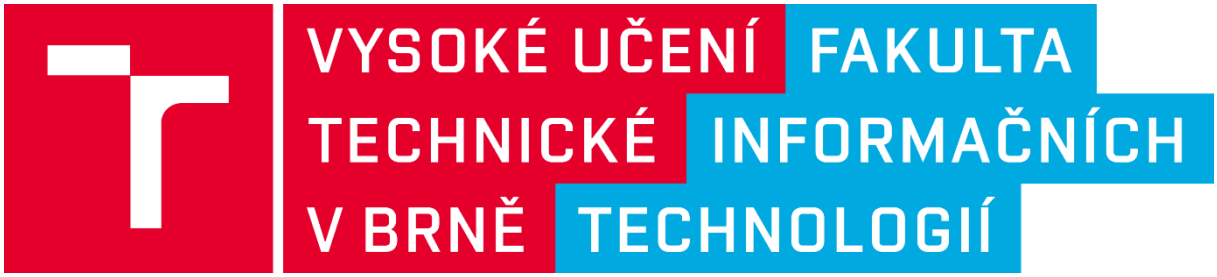
VYSOKÉ UČENÍ TECHNICKÉ V BRNE
FAKULTA INFORMAČNÝCH TECHNOLÓGIÍ



IMP - Mikroprocesorové a vestavěné systémy

ESP32: Přístupový terminál

Matej Horník (xhorni20)

28.11.2021

Obsah

1. Motivace

Cieľom projektu bolo vytvoriť prístupový terminál s maticou klávesnicou 4x3 a dvomi led diódami. Led diódy indikujú v akom stave je terminál. Pri zadaní správneho štvôr miestneho kódu, zelená led dióda sa rozsvieti na 1 sekundu, v inom prípade svieti červená led dióda. Prístupový terminál umožňuje zmenu prednastaveného prístupového kódu (‘1234‘) na iný.

2. Implementácia

2.1. Základné informácie

Prístupový terminál bol implementovaný v jazyku C a vyvíjaný pomocou doplnku do VS Code, ktorý sa volá PlatformIO. Prístupový terminal bol otestovaný na mikrokontroléri ESP32. Celý program je implementovaný v súbore *src/main.c*.

2.2. Využité SW a HW prostriedky

Pri vypracovávaní projektu bol použitý ESP32 mikrokontrolér s frekvenciu 240 Mhz, vnútornou nevolatilnou pamäťou veľkosti 4MB a RAM pamäťou 320 KB. Taktiež bola použitá štandardná maticová klávesnica 4x3 so 7 pinmi, dvomi led diódami, jedna červená druhá zelená a 2 rezistormi pre led diódy. Mikrokontrolér ESP32 sa pripája k počítaču pomocou micro-USB kábla. Medzi softwarovými prostriedkami boli použité interupty na GPIO vstupoch, ktoré sa vyvolali pri stisnutí a pustení tlačítka. Taktiež bola využitá fronta pre ukladania znakov z klávesnice poskytnutá od operačného systému, ktorí beží na mikrokontroléri ESP32. Fronta bola inicializovaná pomocou funkcie **xQueueCreate(),** z knižnice *freertos/queue.h.* Ako posledná softwarová utilita bol použitý binárny semafor na detekovanie pustenia tlačidla aby skener stisnutých tlačidiel vedel, že má čakať až dokým stlačené tlačidlo nebolo pustené. Semafor bol inicializovaný pomocou **xSemaphoreCreateBinary()**, implementácia API pre semafory sa nachádza v knižnici *freertos/semphr.h*.

2.3. Popis implementácie

Pri spustení sa hlavnej funkcie **app_main()** sa ako prvé nastavia GPIO vstupy a výstupy pomocou funkcie **gpio_config()**. Piny klávesnice, ktoré sú pre stĺpce su nastavené ako vstupy a piny riadkov sú nastavené ako výstupy. Ďalej sa nakonfigurovali interupt funkcie pre vstupné piny, ktoré budú detekovať stisk tlačidla. Inicializovala sa fronta a semafor, ktoré budú použité. Nastavili sa aj výstupné piny pre led diódy a vytvorili sa úlohy pomocou **xTaskCreate()**. Hlavná logika programu je implementovaná v úlohe **main_task().** V úlohe **check_task()** je implementované postupné zapínanie jednotlivých riadkov a skenovanie či nejaké tlačidlo bolo stisnuté. Každý riadok prejde na 10 milisekúnd do zapnutého stavu a ak sa stane že tlačidlo z toho riadku bolo stlačené tak čaká na signál pustenie tlačidla pomocou semaforu, ktorý mu bude poskytnutý interupt funkciou daného stĺpca. Úloha bude čakať na semafor pomocou funkcie **xSemaphoreTake()** až dokým mu nebude poskytnutý. Po získaní semaforu úloha bude pokračovať v zapínaní riadkov a testovaní stisnutia tlačidla.

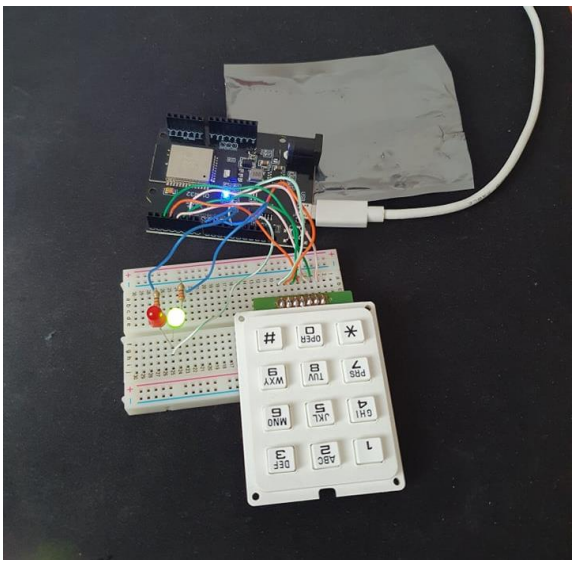
Ďalšou časťou projektu sú interupt funkcie, ktoré sú vyvolané pri zmene hodnoty na vstupných pinoch, čo sú stĺpce maticovej klávesnice. Interupt funkcie sú tri, jedna pre každý stĺpec. V interupt funkcií je nastavený "debounce" čas na 20 milisekúnd. To ošetruje to, aby sa pri stisnutí a pustení tlačidla vygeneroval len jeden signál, ktorý sa má spracovať. Tento efekt sa rieši pomocou funkcie **esp_log_timestamp()** z knižnici *esp_log.h*. Funkcia získa čas od začiatku štartu mikrokontroléru a porovná z časom minulého vyvolania interupt funkcie. Rozdiel v časoch musí byť väčší ako 20. Po vygenerovaní interupt signálu sa zmení stav pravdivostnej hodnoty *pressed* aby sa vedelo v akom stave sa nachádza tlačidlo. Pri púšťaní tlačidla sa indikuje semaforom úlohe **check_task()**, že tlačidlo bolo pustené. Signál sa posiela pomocou funkcie **xSemaphoreGiveFromISR()**. Posledná vec, ktorú interupt rieši je uloženie stisnutého tlačidla do fronty znakov. Získanie aktuálne aktivované riadku sa zistí pomocou globálnej premeny *row*, ktorú nastavuje funkcia **check_task().** Uloženie znaku do fronty zaisťuje funkcia **xQueueSendFromISR()**.

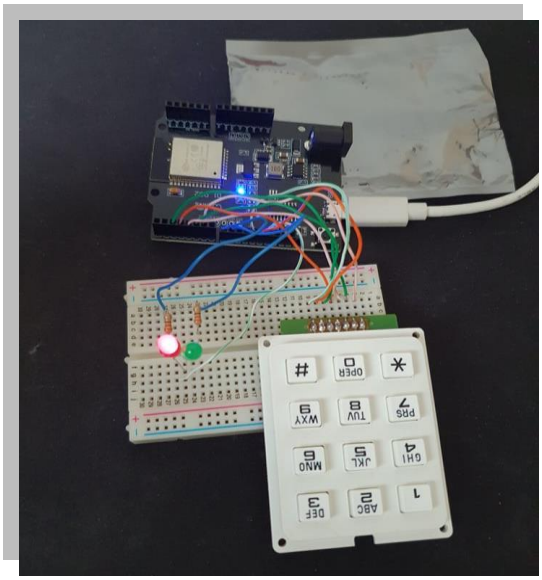
Poslednou časťou prístupného terminálu je hlavná úloha **main_logic()**, ktorá má na starosti hlavnú logiku programu. Hlavná časť úlohy je cyklus, ktorý čaká na znaky uložené do fronty. Po prijatí znaku z fronty pomocou funkcie **xQueueReceive()** sa znak spracuje. Po stisnutí tlačidla '#' sa prechádza do módu zmeny hesla. Táto zmena prechodu sa indikuje blikaním červenej led diódy každých 250 milisekúnd. V tomto móde je možné zmeniť prístupové heslo. Zmena funguje tak, že sa najprv zadá aktuálne heslo a hneď za ním nové

heslo, mód prijme osem znakov a prvé štyri porovná s aktuálnym heslom a ak sa zhodujú, nové heslo, teda posledné 4 znaky sa nastavia ako nové heslo. Správna zmena hesla sa indikuje dvojitým zablikaním zelenej diódy. Pri nesprávnom hesle sa prejde naspäť do normálneho režimu a červená dióda prestane blikať. Blikanie červenej diódy má na starosti úloha **red_blink_task()**. Túto úlohu ovláda hlavná úloha pomocou funkcií **vTaskSuspend()** a **vTaskResume()**. Do normálneho módu sa dá opäť prejsť po stisnutí tlačidla '#'. V normálnom móde aplikácie, to identifikuje červená led dióda, ktorá svieti, sa heslo zadáva pomocou kláves na klávesnici. Po zadaní 4 znakov, program porovná správnosť zadaného hesla a ak sa zhodujú tak zelená led dióda sa rozsvieti na 1 sekundu. Po nesprávnom zadaní hesla sa nič nedeje len sa vymaže počítadlo zadaných znakov. Znak '*' slúži na vymazanie počtu zadaných znakov a užívateľ môže heslo zadávať znovu. Vymazanie sa indikuje bliknutím červenej led diódy.

3. Záver

Aplikácia bola otestovaná pomocou ESP32 a maticovej klávesnice, ktorá nám bola poskytnutá. Aplikácia spĺňa zadanie a všetky časti fungujú tak ako boli zadané. Program umožňuje zmeniť predvolené heslo ('1234') na iné. Pri testovaní aplikácie neboli detekované žiadne chyby.





Hodnotenie: $\sum = (K_1 + K_2*F/5) * (E + F + Q + P + D) =$

$= (0.25 + 0.75*1) * (1 + 5 + 2.8 + 1 + 3.7) =$ 13.5 bodu.

4. Zdroje

Espressif Systems (Shanghai) Co., Ltd.: Esspresif API Reference. 2021, [Online]
URL https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/index.html
FreeRTOS API Reference. 2021, [Online]
URL https://www.freertos.org/a00106.html
Espressif IoT Development Framework. 2021, [Online]
URL https://github.com/espressif/esp-idf
Keypad Datasheet, [Online]
URL https://www.farnell.com/datasheets/1662617.pdf