VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Mikroprocesorové a vestavěné systémy

M - ESP32: Přístupový terminál

14. prosince 2022 Simona Češková (xcesko00)

Obsah

1 Úvod do problému

1.1 ESP32

Mikroprocesor ESP32, konkrétně wemos d1 r32 je deska s integrovaným WiFi modulem s čipem ESP-32. Obsahuje USB rozhraní pro jeho programování i napájení. Komunikaci zajišťuje řadič CH340. ESP-32 je 32 bitovým mikroprocesorem a GPIO piny, které mohou vykonávat činnost podle naprogramovaného kódu. [1]

Mikroprocesory jsou zařízení, která slouží k více účelům a dají se naprogramovat, díky tomu, že na vstupu akceptují digitální data, je poté dokážou spracovat a z nich zobrazit výstup, který zobrazuje výsledek. Spracování je díky instrukcím, které mají uložené v paměti.[2]

1.1.1 Input a output rozhraní

Pro spuštění programu je nutné mít na mikroprocesor ESP32 připojenou klávesnici 3x4 a 2 LED diody. To znamená, že z ESP32 bude využito 9 input/output portů + port GND.

1.2 Použité knihovny

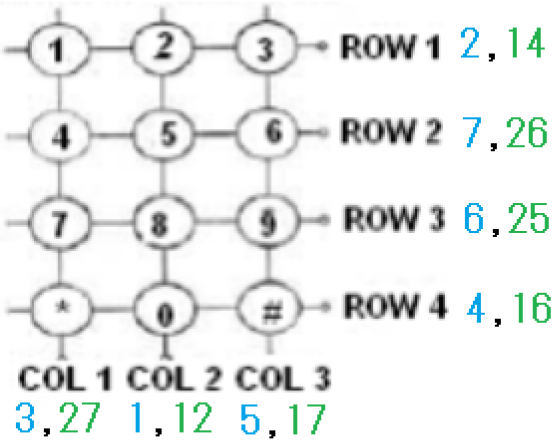
- freertos/FreeRTOS.h - práce s edf prostředím
- freertos/task.h - vytváření tasků - souběžně probíhajících procesů
- freertos/queue.h - využívání čekání
- driver/gpio.h - nastavování pinů - obsahuje většinu použitých funkcí

1.3 Použité rozhraní

Program byly spoušteň a překládán pomocí aplikace ESP-IDF 5.0 CMD a tudíž napsán v nejnovější verzi 5.0.

1.4 Zapojení pinů z 3x4 klávesnice do ESP32

Na obrázků níze jsou zaznačeny modrou barvu čísla pinů, které se nachází na klávesnici. Tyto piny jsou zapojeny do GIO portů na ESP32, jejichž číslo je označeno barvou zelenou.



Obrázek 1: Rozložení pinů na klávesnici

Dvě LED diody - červená a zelená - jsou zapojeny v tomto pořadí do GIOP 13 a GIOP 5. Pro ně je dále nutné k nim přidat rezistor, který bude mezi Vcc a anodou. Katoda bude vést do GND. Pro správný chod programu je nutné takto mikro procesor s klávesnicí a diodami zapojit.

2 Popis řešení

2.1 Rozdělení na podproblémy při řešení

2.1.1 Detekce stisknutého tlačítka

Detekce tlačítla funguje tak, že se do řádků a nebo sloupců postupně pouští Vcc napětí a kontroluje se, zda do konkrétního tlačítla (kombinace jednoho sloupce a jednoho řádku) se napětí dostalo. V případě řešení mého projektu jsem zvolila procházení přes sloupce. Do prvního sloupce nastavím pomocí funkce `gpio_set_level()` `OUTPUT` level na 1 (`high`). Protože jsou všechny sloupce nastaveny díky `gpio_set_direction()` na `OUTPUT`, tak v tomto momentě srkze něj může procházet napětí. Poté je nutné zkontrolovat, zda prvním řádkem prošlo napětí - port, který odpovídá danému řádku, bude mít nastavený level na 1. To lze zjistit pomocí funkce `gpio_get_level()`. Pokud ano, tak kombinace sloupce a řádku odpovídá stisknutému tlačítku, jinak se zkontrolují zbyté tři řádky. Na konci toho úseku kontroly se nastaví level sloupce zpět na 0 pomocí `gpio_set_level()`, aby se tento proces mohl zopakovat pro další sloupec.

2.1.2 Použití přerušení

Přerušení je ansynchroní operací, při které procesor přeruší vykonávanou instrukci, obslouží zdroj vyvolaného přerušení a poté pokračuje v plnění dalších instrukcí. Zdroj přerušení je jakákoliv změna, která se stane během běhu programu. Během obsluhy přerušení není možné provádět jiné změny, než ty které obsahuje, proto je to bezpečná cesta pro provádění změn. U nastavení řádků `gpio_cfg.intr_type` je použila typ `GPIO_INTR_POSEDGE`, který značí `rising edge`, proto při kontrole řádků se kontroluje zda je jeho level nastaven v 1. V ten moment se zvedá hrana napětí a vyvolá se přerušení.

2.1.3 Zkombinování ověřování na klávesnici s prací s LED diodami

Při výkonu nějaké činnosti ve nějaké funkci, v které je třeba iniciovat změnu stavu diody a jejího svícení, se zavolá daná funkce z hlavního `while` cyklu. V tomto případě nejde o žádné složité výpočty, proto není třeba používat druhého procesu, který by obsluhoval zároveň diody a práci s heslem.

2.1.4 Použití funkce `vTaskDelay()`

Tato funkce slouží k vyvování zpoždění/počkání a je nutné ji použít za část kódu v kterých se nastavují důležité změny, aby se nestalo, že program pojede v procesu dál, ale změny se stihnou zapsat až za běhu.

2.2 Kostra programu

2.2.1 Inicializace portů a proměnných

První sekci programu tvoří funkce pro inicializaci:

- `disable_interrupt()` vypne možnost dostávat přerušení, protože se nachází v přerušení. `Enable_interrupt()` jej zase zapne.
- `init_ports()` - nastaví porty
  - nejdříve vyresetuje porty a jejich původní nastavení pomocí funkce `gpio_reset_pin()`
  - pomocí struktury `gpio_config_t`, na kterou jsem přišla díky demu cvičení v přednášce "Platforma ESP32 a její programování"na stránce Elearning. tato struktura umožňuje nastavit pro více portů vlastnosti současně. Po nastavení všech parametrů se konfigurace díky funkci `gpio_config` uloží do registrů.

  - nastaví řádům `pulldown` registry funkcí a zároveň vypne `gpio_pullup` registry, aby nedošlo k případným chybám.
  - jako poslední přiředí portům správnou funkci. Všechny sloupce jsou pouze v `OUTPUT` módu, protože z nich napětí může pouze odcházet dál a naopak všechny řádky jsou v `INPUT` módu, protože u nich chceme zpětnou vazbu, jestli jimi teče napětí. Diody jsou stejně jako sloupce v `OUTPUT` módu.

- `init_gpio()` jsem taktéž použila díky demo cvičení. Nejprve povolí přerušení pro všechny sloupce, poté vytvoří `xQueueCreate`, která je dále volaná ve `while_loop()` funkci jako `xQueueReceive`. Ta čeká až přijde nějaký požadavek, jakmile se tak stane, tak vrátí `TRUE` a program se v tomto případě dostane dál k testování jaké tlačítko na klávesnici bylo zmáčknuto. Mimo toho ještě nastavuje pomocí `gpio_isr_handler_add() ISR handler` = obsluhu přerušení pro odpovídající GPIO piny.

2.2.2 Hlavní běh programu

- `while_loop()` označuje cyklus v kterém se nekonečně dokola kontroluje, jestli nebyla nějaká klávesa zmáčknutá.

2.2.3 Funkce pro práci s diodami

- díky `led_flash()` zabliká dioda, protože obsahuje funkci `gpio_set_level`, která nastaví pomocí globálních proměnných (`R_LED_state` a `G_LED_state`) diodám 1 pro svícení, nebo 0 pro zhasnutí.
- `good_G_LED()` pro signalizaci, že bylo heslo zadáno správně
- `wrong_R_LED()` pro situaci, při které bylo zle zadáno heslo
- `both_LED()` umožňuje uživateli vidět kdy zadal správně heslo při pokusu o změnu hesla. V tomto módu se čeká až zadá nové heslo. Poté se uloží.

2.2.4 Pomocné funkce pro ověřování hesel

- `compare()` porovnává uložené heslo s nově zadaným heslem
- `add_char()` přidá znak do proměnného pole, dokud nedojde k poslednímu ze čtyř znaků a poté je díky `compare()` porovná.
- `store_new_password()` uloží nové heslo do pole `code_array`, v kterém se uchová
- `before_changing_password()` kontrola jestli bylo správně zadné původní heslo, pro púmožnění jeho změny na nové

3 Zhodnocení

3.1 Úskalí řešení projektu

Nejtěžší na projektu bylo na začátku správně nastavit prostředí pro běh programu. Měla jsem problém jak s instalací rozšíření Platformia a IDF do Visual Studio Code, tak i s instalací konzole .ESP-IDF pro Windows. Naštěstí se to později podařilo a již poté se psal projekt dobře.

3.2 Ověření vlastností řešení

Pro ověření správné činnosti je nutné se podívat na problém i jako uživatel a podle toho vytvořit prostředí, které bude především uživatelsky přívětivé. Uživatel očekává zpětnou vazbu, které mu dodávají v tomto případě pouze LED diody. Proto je vhodné zpětně signalizovat podařené i nepodařené úspěchy přihlášení takovou cestou, kterou by uživatel jako chování čekal. Červená LED dioda slouží pro negativní signalizaci, zelená LED dioda pro positivní signalizaci a pokud jsou obě použity zároveň, tak uživatel tuší, že došlo k speciálnímu stavu (vybrání nového hesla). Pro použití v reálném životě by bylo lepší vzhledem k bezpečnosti využít možnost zhasnutí obou LED diod, aby signalizace nepřilákala pozornost okolí. Pro účely jasné zpěné vazby v tomto projektu jsem ponechala signalizaci za použití obou LED diod.

3.3 Výsledek

Výsledkem projektu je program pro mikrokontroler ESP32, který dokáže zpracovávat data, která dostal pomocí externí klávesnice. Zpracováním se myslí ošetření neplatných vstupů. U platných vstupů podle jejich sekvence určit zda se jedná o požadované heslo, nebo ne a podle toho se zachovat. Umožňuje i změnu hesel i když vždy po restartování u sobě nechá uložené základní heslo (1234).

3.4 Možné rozšíření

K řešení by se daly přidat i další inovace, například podpora více znakového hesla, dále povolení využít maximálně n pokusů ($n \in \mathbb{N}$) do zablokování a následné odblokování jednotným speciálním kódem.