ZÁPADOČESKÁ UNIVERZITRA V PLZNI

Fakulta aplikovaných věd

Základy programování pro IoT

Semestrální práce na téma:

Piškvorky

Osobní číslo: A23B0406P

Autor: Michal Malík

Datum: 22. prosince 2024

Obsah

1 Úvod

Tato dokumentace popisuje realizaci semestrální práce v rámci předmětu *Základy programování pro IoT*. Projekt se zaměřuje na vývoj aplikace pro **Internet věcí (IoT)**, která umožňuje monitorování a ovládání zařízení. Tento dokument shrnuje kroky návrhu, implementace, testování a nasazení projektu.
Téma práce se soustředí na známou hru piškvorky a možnost hrát "po síti"mezi dvěma hráči. Aplikace dokonce obsahuje i možnost nastavit si herní pole, podle čeho se mění i pravidla konkrétní hry.

2 Analýza

2.1 Kód pro jedno zařízení

Předtím, než jsem se pustil do programování aplikace, která mi dovolí hrát s někým přes internet, rozhodl jsem se, že vytvořím program, který mi dovolí hrát "couch-multiplayer"na jednom zařízení.
Rozhodl jsem se, že budu postupovat tímto způsobem:

1. Možnost nastavení herní plochy
2. Vytvoření herního pole
3. Průběh hrou
4. Vypsání vítěze či remízi

2.1.1 Možnost nastavení herní plochy

Pro práci s uživatelem jsem využil fyzická tlačítka, které zařízení nabízelo. Abych uživatele informoval jaká velikost je nastavená, umístil jsem doprostřed obrazovky textové pole, které ve formátu <velikost>X<velikost> mění svojí hodnotu, podle funkcní **add_one** a **drop_one**.
Tyto funkce byly vnořeny do tlačítek A a C, prostřední tlačítko B sloužilo k potvrzení hodnoty pole. A k výskoku z jinak nekonečné smyčky, jenž reprezentovala výběr velikosti herní plochy.

2.1.2 Vytvoření herní plochy

Pro správu hry využívám dvě pole polí (matice). **State_matrix**, kde se ukládá na hodnoty i, j symbol hráče, který zrovna hraje. Buď "X"nebo "O", pokud se na daném políčku ještě nehrálo, základní hodnotou je znak "-".
Druhá matice, s názvem **button_matrix** má v sobě uložená tlačítka s naimplementovanou funkcí **change_button**, jenž přebírá parametry i, j, které reprezentují adresu v matici. Funkce po zmáčknutí tlačítka změní hodnotu ve **state_matrix** a i vnitřní text tlačítka, který se nachází na pozici i, j v matici **button_matrix**. Při definici všech tlačítek, se zároveň vytváří jejich podoba na obrazu našeho zařízení.

2.1.3 Průběh hrou

Po nastavení herního pole program vklouzne do další smyčky, která neskonční dokud funkce **check_game_state** nevyhodnotí hru jako dokončenou. Pomocí globální proměnné **playing_player** algoritmus ví, který hráč zrovna hraje, a pomocí této informace vkládá na správnou pozici správný znak. Vždy po nějakém čase, proběhne právě kontrola stavu hry. To je trochu nešťastný způsob ověřování, ale jelikož program fungoval i přesto správně, tak jsem už způsob, kdy se pole zkontroluje, neměnil.

2.1.4 Vypsání vítěze či remízi

Pokud funkce **check _game_state** vyhodnotí hru jako dohranou, vyskočí z cyklu a připraví výpis vítěze. Samotná funkce i samo zjistí kdo vyhrál, a podle toho nastaví globální proměnnou **player_won**.

2.2 Hra pro více hráčů

Teď už jen sem musel přijít na to, jak hru budu moci ovládat přes dvě zařízení najednou, bez toho aniž by vznikali nějaké kolize. Proto jsem vytvořil druhý program, jenž byl skoro stejný jako původní pro jednoho hráče. Ten byl redukovaný o možnost nastavení velikosti pole a jediná jeho funkčnost byla v odehrávání.
Pro propojení mezi dvěma zařízení jsem se rozhodl použít MQTT server a k němu populární broker Mosquitto, který je k dispozici zdarma. Vytvořil jsem si 3 různé topici:

- create_field
- player_one_played
- player_two_played

2.2.1 create_field

Na tento topic posílá data uživatel z prvního zařízení, říkejme mu hráč "X". Druhý hráč, který tento topic odebírá převezme hodnotu, převede ji na datový typ integer a s její pomocí sám sobě vytvoří pole. Druhý hráč, hráč "O"ale zatím stále čeká, než odehraje hráč "X", který je podle obecných pravidel piškvorek vždy na řadě první.

2.2.2 player_one_played & player_two_played

Hráč "X"publikuje na **player_one_played** a hráč "O"z tohoto topicu odebírá a naopak. Hodnoty, které se na tyto topicy posílají, reprezentují souřadnice, na kterých daný hráč odehrál a podle toho přizpůsobí herní plochu na obou zařízeních.
Zároveň se zde nachází pojistka, která omezí možnost uživatele hrát, i když není na řadě s tahem.

Pro oba hráče zde probíhá už zmiňovaná funkce **check_game_state**, která i na obou zařízeních zařídí vypsání správného výsledku potom, co hra skončí.

3 Implementace

3.1 Použité technologie

- Programovací jazyk: Python (redukovaná verze MicroPython)
- Komunikační protokol: MQTT
- Platforma: M5Stack Core 2

3.2 Ukázka kódu

3.2.1 Nastavení herního pole

Následuje ukázka kódu pro nastavení herního pole uživatelem:

```
# Funkce pro fyzická tlačítka
def add_one():
    global field_size
    global field_size_text
    if(field_size == 10):
        field_size = 3
    else:
        field_size += 1
    field_size_text.set_text(str(field_size) + "x" + str(field_size))

def drop_one():
    global field_size
    global field_size_text
    if(field_size == 3):
        field_size = 10
    else:
        field_size -= 1
    field_size_text.set_text(str(field_size) + "x" + str(field_size))

# Průběh
while not btnB.wasPressed():
        btnA.wasPressed(drop_one)
        btnC.wasPressed(add_one)
        wait(0.2)

    screen.clean_screen()
    btnA.wasPressed()
    btnC.wasPressed()

    # Vytvoření matice a tlačítek
    state_matrix = create_square_matrix(field_size)
    button_matrix = create_square_matrix(field_size)

    create_field_by_size(field_size)
    m5mqtt.publish(str('create_field'), str(field_size), 0)
```

3.2.2 Připojení k Wifi a nastavení MQTT serveru

Následuje ukázka kódu, kde je ukázan způsob, jakým se zařízení připojí na MQTT server:

```
# Připojení k Wi-Fi
label0 = M5Label('Not Connected', x=110, y=120, color=0x000, font=
    FONT_MONT_14, parent=None)
wifiCfg.doConnect('SSID', 'Password')

if not wifiCfg.wlan_sta.isconnected():
    for _ in range(5):  # Zkus připojit 5x
        wifiCfg.reconnect()
        if wifiCfg.wlan_sta.isconnected():
            break

if wifiCfg.wlan_sta.isconnected():
    # Připojení k serveru
    screen.set_screen_bg_color(0x9999ff)
    m5mqtt = M5mqtt('', 'test.mosquitto.org', 1883, '', '', 300)
    m5mqtt.subscribe(str('player_two_played'), player_two_played)
    m5mqtt.start()
    screen.clean_screen()
else:
    label0.set_text('Connection Failed')
```

3.2.3 Odehrání

Následuje ukázka kódu pro zpracování tahu hráče:

```
def change_button(row, col):
    global playing_player
    global state_matrix
    global button_matrix
    global m5mqtt

    # Pokud tlačítko není označené
    if state_matrix[row][col] == "-":
        if playing_player == 2:
            return
        else:
            button_matrix[row][col].set_btn_text(STR_X)
            state_matrix[row][col] = STR_X
            playing_player = 2  # Přepnutí na hráče 2
            m5mqtt.publish(str('player_one_played'), str(str(row)+
                str(col)), 0)
```

3.2.4 Konečný výpis hry

Na závěr se podíváme na kód, který má nastarost informovat hráče o výsledku:

```
screen.clean_screen()
END = M5Label('THE END', x=83, y=69, color=0x000, font=FONT_MONT_34
    , parent=None)
winner = M5Label('Winner:', x=66, y=109, color=0x000, font=
    FONT_MONT_22, parent=None)
if(player_won == 1):
    player_who_won = M5Label('Player 1', x=163, y=109, color=0
        xfe0000, font=FONT_MONT_22, parent=None)
elif(player_won == 2):
    player_who_won = M5Label('Player 2', x=163, y=109, color=0
        x18ad03, font=FONT_MONT_22, parent=None)
elif(player_won == 0):
    player_who_won = M5Label('Draw', x=163, y=109, color=0x000,
        font=FONT_MONT_22, parent=None)
```

4 Závěr

Projekt úspěšně demonstruje základní koncepty IoT, který jsme se v předmětu KIV/ZPI naučili, včetně komunikace přes MQTT, MicroPython a ovládání zařízení. Dokumentace poskytuje popis návrhu a kódovou realizaci projektu.
Přišlo mi zajímé programovat poprvé něco na kus hardwaru a jsem rád, že to bylo v přívětivém programovacím jazyce jako je Python. Jediná škoda, která mě potrápila, byl nedostatek dokumentace k různým funkcím M5Stacku a trošku horší IDE UIFlow, které toho moc nenabízelo, až na možnost si pomoct Blocky pseudokódem.
I přesto jsem si nakonec programování M5Stacku užil a doufám, že si to někdy budu moci znova vyzkoušet.