Univerzita Komenského

Fakulta matematiky, fyziky a informatiky

C++ framework pre embedded zariadenia

Bakalárska práca

Meno študenta: Daniel Grohoľ

Študijný odbor: Aplikovaná informatika

Pracovisko: Katedra aplikovanej informatiky

Školiteľ: RNDr. Jozef Šiška

Bratislava

18.2.2019

1 Teoretické východiská

1 Terminológia

1.1 Embedded devices

Kľúčovým pojmom tejto práce je, respektíve sú Embedded devices, čo môžeme preložiť ako vstavané (alebo zabudované) zariadenia. Sú to zariadenia určené predovšetkým pre jednu konkrétnu činnosť, ktoré sú zabudované do väčšieho systému. Príkladom z bežnej praxe je auto so zabudovaným antiblokovacím brzdným systémom (ABS). V tejto práci nás však bude zaujímať hlavne oblasť IOT (Internet of things), kde vecou "thing" sa myslí práve embedded zariadenie. Cena takýchto zariadení na trhu je často veľmi nízka (rádovo v desiatkach eur, často aj do 10 eur) takisto jeho zaobstaranie je nenáročné v bežne známych e-shopoch. Nízka cena umožňuje sériovú výrobu a masové používanie. Tieto zariadenia sú často programované v jazykoch C, C++ alebo aj v JAVE, avšak Java Virtual Machine vyžaduje pre svoje spustenie FreeRTOS (k nemu sa dostaneme neskôr).

1.2 C++

V našej práci budeme používať hlavne programovací jazyk C++, ktorý je jedným z najpoužívanejších a najširšie uplatniteľných programovacích jazykov. Oproti jazyku C poskytuje výhody objektového programovania a s nimi aj možnosť použitia rozsiahlych knižníc, ako napríkad STL (Standard Template Library), ktoré značne uľahčujú a zrýchľujú programovanie. Používanie tejto knižnice je bezproblémové na výkonných počítačoch s veľkým množstvom operačnej pamäte, avšak čo sa týka embedded zariadení, tuna to má značnú nevýhodu, a to v tom, že pamäť je častokrát alokovaná dynamicky, čo znamená, že programátor si dopredu nemôže byť istý, že jeho program ju počas behu nevyčerpá celú. Toto by viedlo k neželanému prerušeniu, a pokiaľ by dané zariadenie malo vykonávať nejakú kritickú úlohu, napríklad ovládanie senzora v automobile , mohlo by to ohroziť aj ľudské životy. Preto sa v oblasti embedded zariadení stalo dobrou praxou nepoužívať STL knižnicu,

prípadne používať iba jej vybrané časti. V niektorých frameworkoch pre embedded zariadenia sa dá používanie tejto knižnice jednoducho vypnúť.

1.3 FreeRTOS

FreeRTOS je malý real-time operačný systém určený pre embedded zariadenia. Jeho nevýhodou je väčšia hardverová náročnosť, preto sa nehodí pre každé použitie. Keďže využíva thready, programátor sa pri ich použití dostáva do rizík spojených s konkurenčnými procesmi. V našej práci sa však cheme aj kvôli tomuto riziku venovať asynchrónnemu programovaniu, kde tieto problémy odpadajú. Väčšina dnešných frameworkov pre embedded zariadenia využíva práve FreeRTOS.

1.4 Statická alokácia pamäte

Ak hovoríme, že program alokuje pamäť staticky, znamená to, že je veľkosť pamäte v čase kompilácie už známa a fixná (nemení sa teda počas behu samotného programu), a že sa pamäť alokuje na stack-u (zásobníku). Tento typ alokácie je dôležitý hlavne pre zariadenia, ktoré majú obmedzenú veľkosť pamäte, kedy si chceme byť dopredu istí, že sa nám počas behu programu neminie celá pamäť. Príkladom takýchto zariadení sú práve embedded zariadenia.

1.5 Dynamická alokácia pamäte

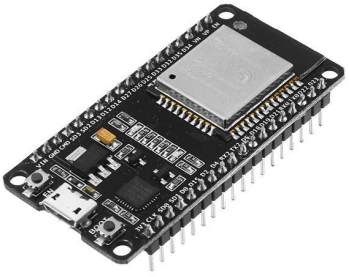
Opakom statickej alokácie je dynamická alokácia, kedy sa pamäť alokuje počas behu programu, a alokácia prebieha na heap-e (halda). V embedded zariadeniach sa tomuto spôsobu chceme vyhnúť. Väčšina kontajnerov v STL knižnici jazyka C++, ako napríklad **std::list**, alokuje pamäť dynamicky, čo je z hľadiska embedded systémov neprijateľné.

V našej práci sa budeme snažiť vytvoriť framework, ktorý bude tento problém riešiť. Každý z frameworkov v tejto práci rieši tento problém.

1.6 ESP 32

ESP32 je séria čipov s nízkymi nárokmi na energiu vyrábaných čínskou firmou Espressif. V našej práci sa sústredíme práve na tento mikročip. Jeho výhodou je nízka cena, už spomínaná nízka energetická spotreba a zabudovaný WI-FI a Bluetooth modul, čím sú práve tak populárne v IOT oblasti.

Technická špecifikácia	
Hlavný procesor	Tensilica Xtensa 32-bit LX6 microprocessor
Počet jadier procesora	2 alebo 1 (závisí od modelu)
Takt procesora	do 240 MHz
ROM	448 KiB
SRAM	8 KiB
Embedded flash	0, 2 alebo 4 MiB (závisí od modelu)
Externá flash pamäť a SRAM	až do 16 MiB
Bezdrôtové pripojenie	Wi-Fi 802.11 b/g/n/e/i (802.11n @ 2.4 GHz až do 150 Mbit/s)
	Bluetooth v4.2 BR/EDR a Bluetooth Low Energy (BLE)
Zabezpečenie	IEEE 802.11 štandardné zabezpečovacie prvky vrátane WFA, WPA/WPA2 a WAPI
	Secure boot
	Šifrovanie flash pamäte
	Hardverová podpora šifrovania (AES, SHA-2, RSA,...)



Obr. 1.1: ESP32

1.7 Asynchrónne programovanie

Pojem asynchrónne programovanie sa v poslednej dobe dostáva čím ďalej tým viac do popredia, a to hlavne vďaka obľúbenosti jazyka JavaScript, ktorý ho používa v značnej miere aj vďaka snahe zlepšiť pužívateľské prostredie webových stránok, hlavne aby miera odozvy bola čo najmenšia, a užívateľa to neodradilo.

Uveďme si príklad asynchrónnej operácie. Predstavme si, že v našom zdrojovom kóde programu máme za sebou dve operácie, ktoré po sebe bezprostredne nadväzujú. Prvou, operáciou A, je sťahovanie súboru, druhou, operáciou B je vypísanie nejakého textu na obrazovku. Pokiaľ by tieto operácie boli synchrónne (čo je opakom asynchrónneho prístupu), musel by používateľ po príchode na stránku čakať, kým sa operácia A, teda stiahnutie súboru dokončí, čo by mohlo trvať minúty. Toto čakanie by väčšinu užívateľov odradilo.

1.8 Promises

V programovacom jazyku JavaScript predstavuje objekt Promise (môžeme preložiť ako prísľub) udalosť, ktorá môže (ale nemusí) nastať v budúcnosti. Táto udalosť, teda Promise môže mať 3 stavy:

- čakajúci (počiatočný stav)
- splnený (reprezentuje úspešnú operáciu)
- zamietnutý (reprezentuje neúspešnú operáciu)

Pokiaľ Promise zmení svoj stav, pomocou notifikácií dá o tom vedieť všetkým objektom, ktoré sa o túto informáciu zaujímajú. Ak sa Promise dostane do stavu splnený alebo zamietnutý, už sa viac tento stav nedá zmeniť.

Promise je v JavaScripte trieda, ktorá má metódy:

- then()
- catch()

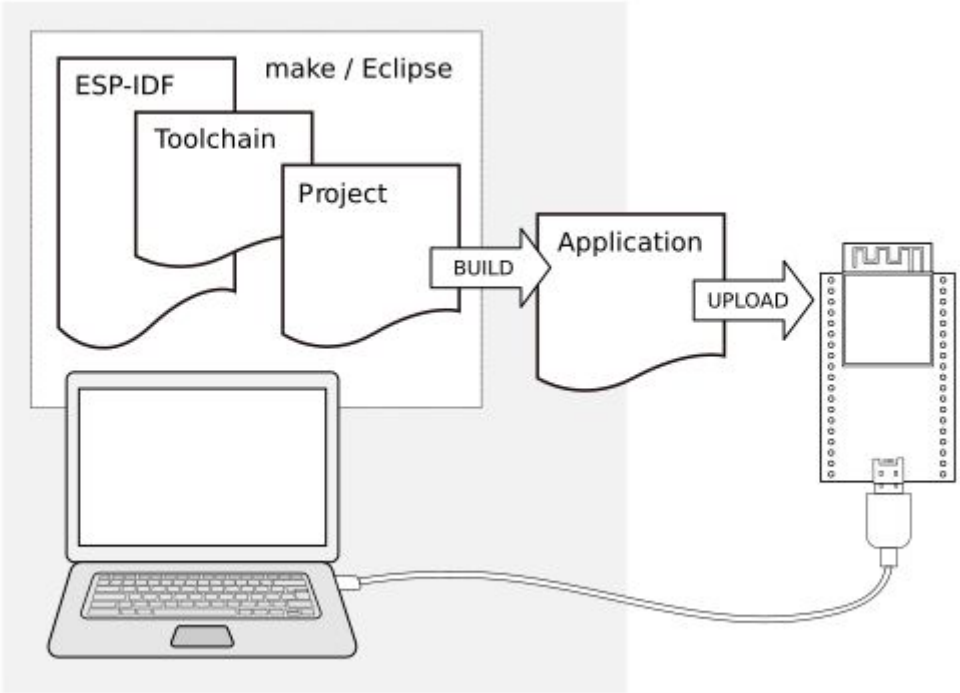
1.9 Observable

2 1.2. Existujúce riešenia

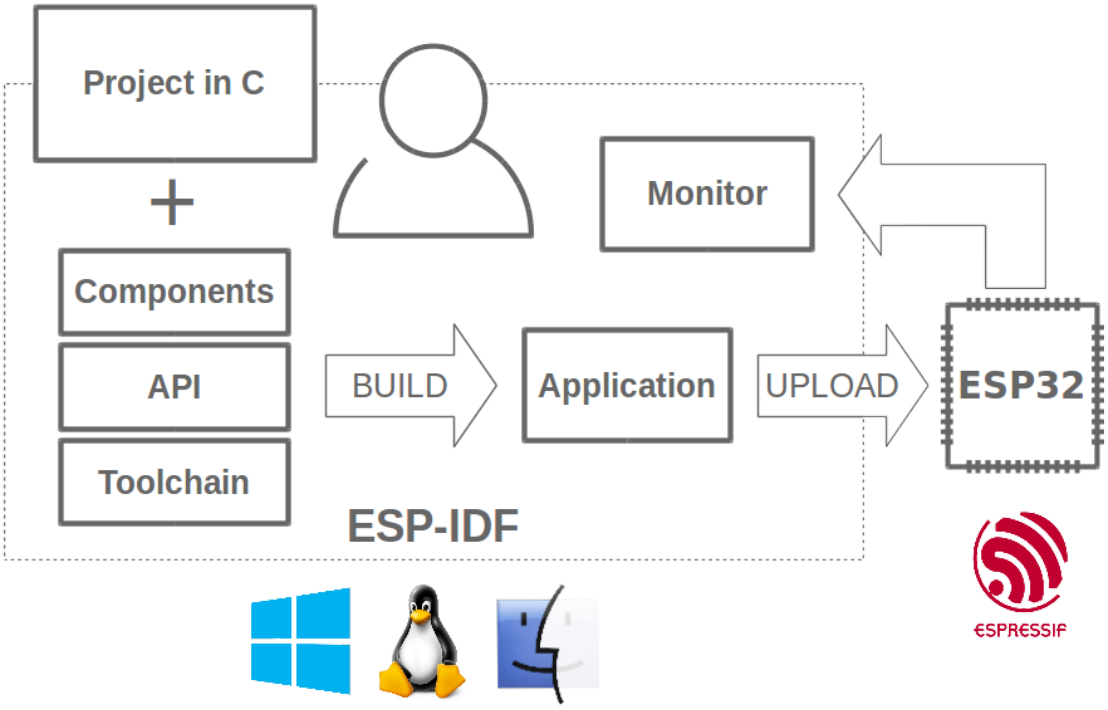
V tejto podkapitole predstavíme a popíšeme už existujúce C++ frameworky a knižnice pre embedded zariadenia.

2.1 ESP-IDF

Výrobca čipov ESP32, čínska firma Espressif, ponúka zdarma dostupný framework pod názvom Espressif IOT Development Framework, ktorý je oficiálny framework pre mikrokontróler ESP32, popísaný vyššie. Je postavený na operačnom systéme FreeRTOS, ktorému by sme sa však chceli v našej práci vyhnúť, pretože s ním je spojené známe riziko viac-vláknového programovania.
Samotný framework tvorí Toolchain, Na stránke frameworku sa nachádza podrobný návod na inštaláciu, ktorý zahŕňa aj inštaláciu potrebnej Toolchain pre daný operačný systém (Podporovaný je Windows, MacOS aj Linux).



Obr.: Vývoj aplikácií pre ESP32[1]



Obr. 1.2: Schéma frameworku ESP-IDF [1]

[1] https://docs.espressif.com/projects/esp-idf/en/latest/get-started/

2.2 Embedded Template Library (ETL)

Embedded Template Library je knižnica (alebo inak aj framework) navrhnutá pôvodne pre embedded zariadenia, takže jej cieľom je riešiť problém dynamickej alokácie pamäte na tomto (pamäťovo) obmedzenom hardvéri. Jej cieľom nie je úplne nahradiť STL, skôr sa ju snaží doplniť a prispôsobiť pre embedded zariadenia.

Výhody tejto knižnice sú:

- nezávislosť od STL
- pamäť alokovaná iba staticky
- nepoužíva virtuálne metódy (virtual methods)
- všetky zdrojové súbory sú hlavičkového formátu (header)
- jednoduchosť inštalácie

2.2.1 Nezávislosť od STL

Pri používaní knižnice ETL sa môžeme výhradne obísť bez použitia štandardnej C++ knižnice STL. Táto vlastnosť sa dá zabezpečiť použitím makra **ETL_NO_STL**. Ak to chceme nastaviť, teda ak chceme zakázať STL knižnicu, musíme toto makro zadefinovať v súbore **etl_profile.h**, ktorý je potrebné ešte predtým vytvoriť.

2.3 Smooth

Posledným príkladom je Smooth, je C++ framework pre aplikácie využíva operačný systém FreeRTOS, avšak, ako autor sám uvádza, práca s ním je úplne založená na event-driven architektúre a tým pádom je thread-safe (bezpečná z hľadiska vlákien). Tento framework stavia na štandardnej knižnici, ktorá je nevhodná pre embedded riešenia, keďže pri jej používaní hrozí vyčerpanie pamäte, avšak Smooth poskytuje viaceré funkcie, ako napríklad inicializácia aplikácií, časové a systémové udalosti, či plnenie úloh.

Obsah

Literatúra

https://www.banggood.com/ESP32-Development-Board-WiFiBluetooth-Ultra-Low-Power-Consumption-Dual-Cores-ESP-32-ESP-32S-Board-p-1109512.html?cur_warehouse=CN

https://www.etlcpp.com/setup.html

https://github.com/PerMalmberg/Smooth

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using_promises

https://eloquentjavascript.net/11_async.html

https://www.promisejs.org/

https://www.banggood.com/ESP32-Development-Board-WiFiBluetooth-Ultra-Low-Power-Consumption-Dual-Cores-ESP-32-ESP-32S-Board-p-1109512.html?cur_warehouse=CN