STREDNÁ PRIEMYSELNÁ ŠKOLA ELEKTROTECHNICKÁ
PREŠOV

IV.B - 27

ŠK. R. 2021 – 2022

PČOZ – ZÁZNAMNÍK CAN KOMUNIKAČNEJ LINKY

DANIEL ŠLOSÁR

Konzultant: Ing. Ondrej Kontura

Anotácia v slovenskom jazyku

Cieľom tejto práce je navrhnúť a skonštruovať zariadenie – autonómny záznamník komunikácie na zbernici CAN. Úlohou zariadenia bude záznam dátovej komunikácie na linke do internej pamäte/pripojiteľného USB zariadenia. Zariadenie bude mať svoj vlastný obal, ktorý bude navrhnutý špeciálne pre toto zariadenie. Zariadenie bude obsahovať HMI pre ovládanie spustenia a ukončenia záznamu, indikáciu prebiehajúceho záznamu a prípadných chybových stavov (tlačidlá, displej). Zariadenie bude napájane externou batériou (powerbank) alebo zo siete. Súčasťou zadania je tvorba webového rozhrania pre filtráciu a vizualizáciu zozbieraných dát.

Anotácia v anglickom jazyku

This work aims to design and construct a device. An autonomous communication recorder on the CAN/Modbus bus. The task of this device will be to record data communication on the line into the internal memory/connectable USB device. Device will have it’s own case that will be designed and constructed specially for this device. The device will contain HMI for controlling the start and end of the recordings, indication of recordings and possible error states (buttons, display). This device will take power from an external battery (power bank) or the main supply. Part of the assignment is to create a computer app that will visualize the collected data.

Čestné vyhlásenie

Vyhlasujem, že som praktickú časť odbornej zložky maturitnej skúšky na tému „Záznamník CAN komunikačnej linky“ vypracoval samostatne, s použitím uvedenej literatúry.

Som si vedomý zákonných dôsledkov, ak v nej uvedené údaje nie sú pravdivé.

Prešov, 10. máj 2022 .....

vlastnoručný podpis

Poďakovanie

Chceli by sme sa poďakovať celej firme ComAp s ktorou sme tento projekt riešili. Za ich dodané materiály či skúsenosti, ktoré nám pomohli pri vypracovaní tohto projektu. Najmä Ing. Štefanovi Hedvigovi, Ing. Mariánovi Šilonovi a Mgr. Lenke Mrázovej za skvelú podporu a konzultácie, ktoré nám pomohli vyriešiť problémy na ktoré sme narazili počas vývoju tohto produktu. Ďalej by sme sa chceli poďakovať Ing. Ondrejovi Konturovi za konzultácie a sprostredkovanie spolupráce so spoločnosťou ComAp.

Obsah

Úvod

CAN je zbernica, využívaná najčastejšie na vnútornú komunikačnú sieť senzorov a funkčných jednotiek v automobile. Pomocou CAN vieme cenovo a efektívne ovládať početné elektronické zariadenia automobilu ako sú napr. svetlá alebo sťahovacie okná, ktoré by inak vyžadovali extenzívnu kabeláž. To ale neznamená že ho nemôžeme použiť napr. na pásovej linke. Firma ComAp sa zaoberá riešením takýchto zariadení.

Túto tému sme si zvolili z dôvodu spolupráce s firmou, ktorá nám vie pomôcť s riešením problémov. Zaujala nás možnosť pracovať a vyvíjať reálny produkt, ktorý bude v budúcnosti užitočný. Hotový produkt sa zapojí na linku CAN kde bude snímať komunikáciu prípadne zlyhanie na linke. Po skončení čítania linky sa zariadenie odpojí a užívateľ bude schopný pripojiť toto zariadenie do PC, prípadne stiahnuť údaje na USB zariadenie. Zozbierané dáta budú následne pripravené na poslanie do webovej aplikácie, kde bude možné zozbierané dáta následne filtrovať a prechádzať.

V neposlednom rade každý z nás si zvolil túto tému preto, lebo každý z nás navštevuje iný odbor na tejto škole. Preto sme chceli využiť doterajšie získané znalosti z vyučovacích hodín a preniesť ich na reálny produkt, ktorý bude používaný v praxi. Samozrejme táto téma je zaujímavá a súvisí s odbormi, ktoré študujeme.

1 Cieľ práce

Našim cieľom bolo vybrať vhodné zariadenie a naprogramovať ho. Toto autonómne zariadenie, ktoré sa pripojí na zbernicu CAN bude zaznamenávať prevádzku na linke do internej pamäte / USB zariadenie:

- vyhľadanie a zoznámenie sa s protokolom CAN a spôsobe ich fungovania,
- výber vhodného development kitu,
- výber vhodného prevodníka CAN-u,
- návrh záznamníka,
- návrh a realizácia obalu pre záznamník,
- naprogramovanie mikrokontroléra,
- navrhnutie komunikácie s perifériami,
- podrobná analýza projektu,
- otestovanie vstupných/výstupných dát,
- spracovanie dát na webe,
- naprogramovanie progresívnej webovej aplikácie na zozbieranie a vizualizáciu dát.

2 Metodika práce (Materiál a metodika práce)

V začiatku tohto projektu sme si museli najprv vybrať vhodný mikrokontrolér pre činnosť nášho projektu. Podľa zadaných parametrov sme hľadali vhodné kity dostupné na trhu. Najvhodnejším kitom sa nám zdal Discovery Kit od firmy STM. Túto voľbu sme prekonzultovali s našími mentormi a zhodli sme sa na kite STM32F429. Ďalej sme si rozdelili ciele medzi našími rolami v tíme. Firmware programátor a hardverový dizajnér museli čakať na dlhšiu dobu z dôvodu problémov s nedostatkom kitov na trhu. Softverový porgramátor začal spracovávať webovú aplikáciu s ktorou bude možné spracovávať a vizualizovať nazbierané údaje. Vybral si vhodnú technológiu používateľského rozhrania Blazor Webassembly. Analytik začal spracovávať tému CAN rozhraní a spisovať zadanie do internej wikipédie firmy ComAp. Toto zadanie bude dostupné v prílohe. Následne začal s návrhom vhodných displejov pre kit. Ďalej pokračoval s návrhom vhodného displejového menu pre CAN záznamník. Firmware programátor začal pracovať na komunikácii kitu s CAN prevodníkom. Hardverový dizajner začal navrhovať vhodnú krabičku na umiestnenie kitu.

3 Úvod do problematiky

3.1 CAN

Controller Area Network (CAN) - je multiplexná sériová komunikácia vysokej prenosovej rýchlosti so zaistením vernosti dát. Po zbernici prebieha komunikácia medzi dvoma uzlami pomocou správ (Data/Remote Frame). Každá správa obsahuje špeciálny identifikátor (ID). Pomocou špeciálnych správ (Error/Overload Frame) je zabezpečení management siete. Protokol CAN sa vyznačuje silným mechanizmom zabezpečenia prenášaných dát. Norma definuje dátovú a fyzickú vrstvu zbernice do prenosovej rýchlosti 1Mbit/s.

Filtrovanie správ CAN - komunikácie po sieti CAN sú často schopné filtrovať komunikáciu sieťou CAN a podľa nastavených pravidiel zhromažďovať iba správy s požadovaným identifikátorom a po prijatí tieto správy ukladať do FIFO registru a následne, kde čakajú na spracovanie.

3.1.1 Správy CAN-u

Správy vysielané po zbernici CAN neobsahujú žiadne informácie o cieľovom uzle, ktorému sú určené. Každá správa je určená identifikátorom, ktorý udává význam prenášanej správy a jej prioritu.

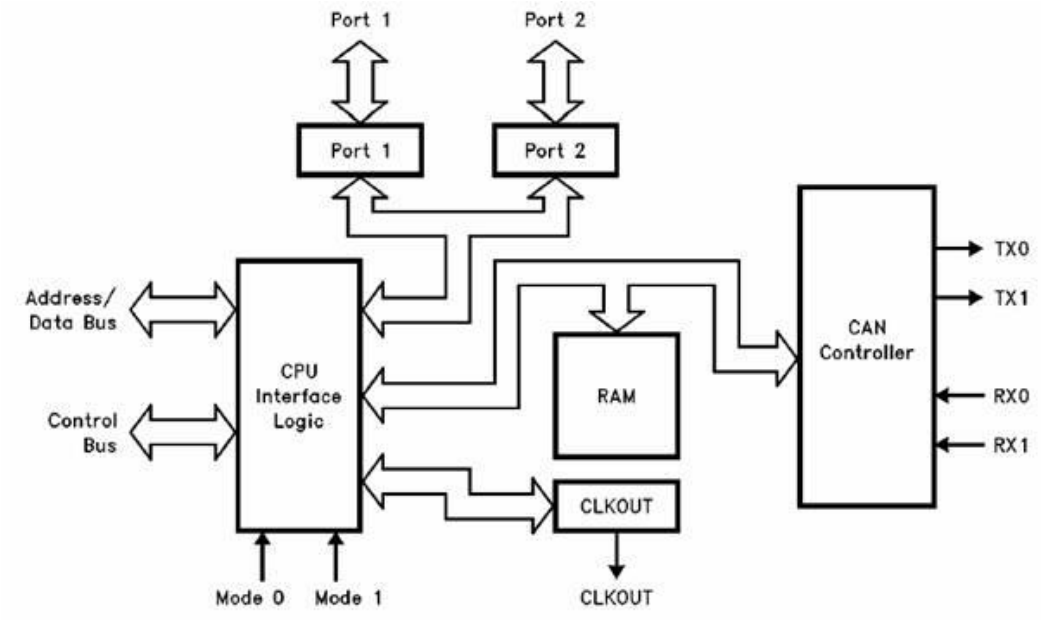
Protokol CAN definuje štyri rôzne typy správ (Frame):

1. **Data Frame** – tvorí základ komunikácie, slúži na prenos dátových informácii až do 8 Bytov
2. **Remote Frame** – predáva žiadosť na prenos dát
3. **Error Frame** – slúži na detekciu chyby prenosu na zbernici a môže byť vyslaný ktoroukoľvek jednotkou v sieti
4. **Overload Frame** – slúži na vyžiadanie spomalenie prenosu dát

3.1.2 Hardwarová štruktúra CAN-u

Štruktúra zariadení, ktoré podporujú komunikáciu CAN-u sa skladá z troch častí.

1. **Prijímač a vysielač dát** – zariadenia obsahujú prijímač a vysielač dát, ktorý komunikuje priamo s fyzickým médiom, ktorým sa prenášajú dáta
2. **Sada registrov** – zariadenia obsahujú sadu registrov, do ktorých sa ukladajú prijaté dáta alebo dáta pripravené na odoslanie
3. **Interface** – zariadenie obsahuje interface, ktorý slúži na komunikáciu s nadradeným CPU



Obr. 1 Bloková Schéma CAN-u (http://noel.feld.cvut.cz/vyu/scs/prezentace2004/CAN/)

3.2 Programovanie mikrokontroléra

V prvom rade základné koncepty rôznych typov komunikácií medzi elektronickými zariadeniami a komunikačnými protokolmi, ktoré budú použité v tomto projekte ako SPI a CAN FD a ich principiálne charakteristiky a rôzne výhody.

Ďalej sú popisované použité hardware komponenty detailnou špecifikáciou každého komponentu. V prípade CAN FD kontroléra, ktorého budeme používať, budeme vysvetľovať aj jeho softvérovú časť, odhaľovaním ako sú ukladané dáta v jednotlivých používaných registroch a ako fungujú a neposlednom rade dáta posielané do kontroléra musia byť posielané aby mohli byť korektne spracované.

Na konci, proces vývoja záznamníka krok za krokom začiatkom vysvetľovania zmien vykonávaných v jednotlivých použitých a už existujúcich STM32 knižniciach pre prispôsobenie nášmu projektu a ako sú vylepšené niektoré aspekty originálneho

CAN FD ovládača. Potom samotné tvorenie main skriptu zodpovedného za chod všetkých knižníc a funkcií potrebných na setup záznamníka.

3.3 Webová aplikácia

Skôr, než sme začali pracovať na vývoji počítačovej aplikácie určenej na zobrazenie a analýzu dát zozbieraných samotným záznamníkom, zistili sme si z dostupnej odbornej literatúry informácie o rôznych použiteľných technológiách, ktoré by vyhovovali našim požiadavkám. Získané informácie, ktoré sa stali základom našej práce, prezentujeme v nasledujúcich podkapitolách.

3.3.1 Progresívna webová aplikácia

Progresívna webová aplikácia (PWA), je typ aplikačného softvéru dodávaného cez web vytvorený pomocou bežných webových technológií vrátane HTML, CSS, JavaScript a WebAssembly. Je určený na prácu na akejkoľvek platforme, ktorá používa prehliadač vyhovujúci štandardom, vrátane stolných aj mobilných zariadení.

Keďže progresívna webová aplikácia je typ webovej stránky alebo webovej lokality známej ako webová aplikácia, nevyžadujú samostatné viazanie ani distribúciu. Vývojári môžu jednoducho zverejniť webovú aplikáciu online, uistiť sa, že spĺňa základné „požiadavky na inštaláciu“ a používatelia si budú môcť aplikáciu pridať na svoju domovskú obrazovku. Publikovanie aplikácie do digitálnych distribučných systémov ako Apple App Store alebo Google Play je voliteľné.

Od roku 2021 sú funkcie PWA v rôznej miere podporované prehliadačmi Google Chrome, Apple Safari, Firefox pre Android a Microsoft Edge, ale nie Firefoxom pre stolné počítače.

Všetky progresívne webové aplikácie sú navrhnuté tak, aby fungovali v akomkoľvek prehliadači, ktorý je v súlade s príslušnými webovými štandardmi. Rovnako ako pri iných multiplatformových riešeniach je cieľom pomôcť vývojárom vytvárať multiplatformové

aplikácie jednoduchšie, ako by to robili s natívnymi aplikáciami. Progresívne webové aplikácie využívajú stratégiu progresívneho vylepšenia vývoja webu.

Niektoré progresívne webové aplikácie využívajú architektonický prístup nazývaný App Shell Model. V tomto modeli pracovníci servisu ukladajú základné používateľské rozhranie alebo „shell“ webovej aplikácie s citlivým webovým dizajnom do offline vyrovnávacej pamäte prehliadača. Tento model umožňuje PWA udržiavať natívne používanie s webovým pripojením alebo bez neho. To môže zlepšiť čas načítania poskytnutím počiatočného statického rámca, rozloženia alebo architektúry, do ktorej je možné obsah načítať postupne aj dynamicky.

3.3.2 Technológie PWA

Na vytváranie progresívnych webových aplikácií sa bežne používa mnoho technológií. Webová aplikácia sa považuje za PWA, ak spĺňa „kritériá inštalovateľnosti“, a teda môže pracovať offline a možno ju pridať na domovskú obrazovku zariadenia. Na splnenie tejto definície všetky PWA vyžadujú minimálne service worker a manifest.

3.3.3 Manifest

Manifest webovej aplikácie je špecifikácia W3C, ktorá definuje manifest založený na JSON (zvyčajne označený manifest.json), aby vývojárom poskytol centralizované miesto na vkladanie metadát spojených s webovou aplikáciou vrátane:

- názov webovej aplikácie,
- odkazy na ikony webových aplikácií alebo obrázkové objekty,
- preferovaná adresa URL na spustenie alebo otvorenie webovej aplikácie,
- konfiguračné údaje webovej aplikácie,
- predvolená orientácia webovej aplikácie,
- možnosť nastavenia režimu zobrazenia napr. celá obrazovka,
- tieto metadáta sú kľúčové pre aplikáciu, ktorá sa má pridať na domovskú obrazovku alebo inak uvedená vedľa natívnych aplikácií.

3.3.4 Podpora iOS

iOS Safari čiastočne implementuje manifesty, zatiaľ čo väčšinu metadát PWA je možné definovať prostredníctvom rozšírení metaznačiek špecifických pre Apple. Tieto značky umožňujú vývojárom povoliť zobrazenie na celú obrazovku, definovať ikony a úvodné obrazovky a určiť názov aplikácie.

3.3.5 Úložisko dát

Kontexty spúšťania progresívnych webových aplikácií sa uvoľňujú vždy, keď je to možné, takže progresívne webové aplikácie musia uchovávať väčšinu dlhodobého interného stavu (údaje používateľa, dynamicky načítané prostriedky aplikácie) jedným z nasledujúcich spôsobov.

3.3.6 Webové úložisko

Web Storage je štandardné API W3C, ktoré umožňuje ukladanie hodnôt kľúča v moderných prehliadačoch. Rozhranie API sa skladá z dvoch objektov, sessionStorage (ktorý umožňuje ukladanie len pre reláciu, ktoré sa vymaže po skončení relácie prehliadača) a localStorage (ktoré umožňuje ukladanie, ktoré pretrváva počas relácií).

3.3.7 Service worker

Servisný pracovník je webový worker, ktorý implementuje programovateľný sieťový proxy, ktorý dokáže reagovať na webové/HTTP požiadavky hlavného dokumentu. Je schopný skontrolovať dostupnosť vzdialeného servera a uložiť obsah do vyrovnávacej pamäte, keď je tento server dostupný, a poskytnúť tento obsah neskôr do dokumentu. Service worker, rovnako ako všetky ostatné webové workery, pracujú oddelene od kontextu hlavného dokumentu. Servisní pracovníci môžu spracovávať oznámenia push a synchronizovať údaje na pozadí, ukladať do vyrovnávacej pamäte alebo získavať požiadavky na zdroje, zachytávať sieťové požiadavky a prijímať centralizované aktualizácie nezávisle od dokumentu, ktorý ich zaregistroval, aj keď tento dokument nie je načítaný.

Servisní pracovníci prechádzajú životným cyklom v troch krokoch: Registrácia, Inštalácia a Aktivácia. Registrácia zahŕňa informovanie prehliadača o polohe servisného pracovníka pri príprave na inštaláciu. Inštalácia nastane, keď v prehliadači webovej aplikácie nie je nainštalovaný žiadny servisný pracovník alebo ak existuje aktualizácia pre servisného pracovníka. K aktivácii dôjde, keď sa zatvoria všetky stránky PWA, takže medzi predchádzajúcou verziou a aktualizovanou verziou nedôjde ku konfliktu. Životný cyklus tiež pomáha udržiavať konzistentnosť pri prepínaní medzi verziami servisného pracovníka, pretože pre doménu môže byť aktívny iba jeden servisný pracovník.

3.3.8 API indexovanej databázy

Indexed Database API je štandardné databázové API W3C dostupné vo všetkých hlavných prehliadačoch. Rozhranie API je podporované modernými prehliadačmi a umožňuje ukladanie objektov JSON a akýchkoľvek štruktúr reprezentovateľných ako reťazec. Rozhranie API indexovanej databázy je možné použiť s obaľujúcou knižnicou, ktorá okolo nej poskytuje ďalšie konštrukcie.

3.3.9 Blazor Webassembly

Blazor WebAssembly je nová technológia používateľského rozhrania od spoločnosti Microsoft, oficiálne vydaná s .NET Core 3.1 a aktualizovaná v .NET 5. Blazor umožňuje vývojárom vytvárať jednostránkové aplikácie (SPA) pomocou C# a .NET s využitím architektúry založenej na komponentoch. Blazor WebAssembly je implementácia Blazor na strane klienta v prehliadači, ktorá zahŕňa .NET runtime implementované vo WebAssembly. Predtým bolo možné vytvárať webové stránky pomocou ASP.NET Core MVC a Blazor Server, pričom každá z týchto ponúk bola riešením na strane servera. Výhodou inštrukčného súboru WebAssembly je relatívne vysoká rýchlosť oproti JavaScriptu. Nevýhodou je dlhšie prvotné načítavanie stránky, pretože do prehliadača sa musí nainštalovať .NET Runtime, avšak deje sa tak bez zásahu užívateľa.

WebAssembly je inštrukčný súbor so špecifickým binárnym formátom. Akýkoľvek hostiteľ (hardvér alebo softvér), ktorý spĺňa špecifikáciu, je preto schopný čítať binárne súbory a spúšťať ich – buď interpretované, alebo priamo kompiláciou do strojového jazyka špecifického pre zariadenie.

Wasm je podobný bežnej inštrukčnej sade (Common Intermediate Language), do ktorej sa kompiluje zdrojový kód .NET. Rovnako ako .NET, aj Wasm môže byť generovaný z vyšších jazykov, ako je C#.

3.4 Desktop Aplikácia

Keďže sme pri testovaní webovej aplikácie zistili výrazne zhoršený výkon pri čítaní a spracovávaní veľkých súborov, rozhodli sme sa vytvoriť aj desktopovú aplikáciu pre operačný systém windows, pri ktorej je čítanie a zapisovanie súborov omnoho rýchlejšie. Na tento účel sme zvolili programovací jazyk c#, aby sme mohli využiť zdrojový kód z našej webovej aplikácie. Pre vytvorenie používateľského rozhrania sme použili framework WPF.

3.4.1 Windows Presentation Foundation (WPF)

Windows Presentation Foundation je UI framework, ktorý vytvára klientske aplikácie pre stolné počítače. Vývojová platforma WPF podporuje širokú škálu funkcií vývoja aplikácií vrátane aplikačného modelu, zdrojov, ovládacích prvkov, grafiky, rozloženia, dátových väzieb, dokumentov a zabezpečenia. WPF používa jazyk XAML (Extensible Application Markup Language) na poskytovanie deklaratívneho modelu pre programovanie aplikácií.

3.5 GTKWave

Súčasťou webovej aj desktopovej aplikácie je aj prevodník, ktorý vygeneruje zo vstupného CSV súboru súbor typu VCD, ktorý je možné otvoriť v aplikácii GTKWave. GTKWave je plne funkčný open-source nástroj na zobrazovanie časového priebehu analógových a digitálnych veličín založený na GTK+ pre Unix, Win32 a Mac OSX, ktorý číta súbory LXT, LXT2, VZT, FST a GHW, ako aj štandardné súbory Verilog VCD/EVCD a umožňuje ich prezeranie.

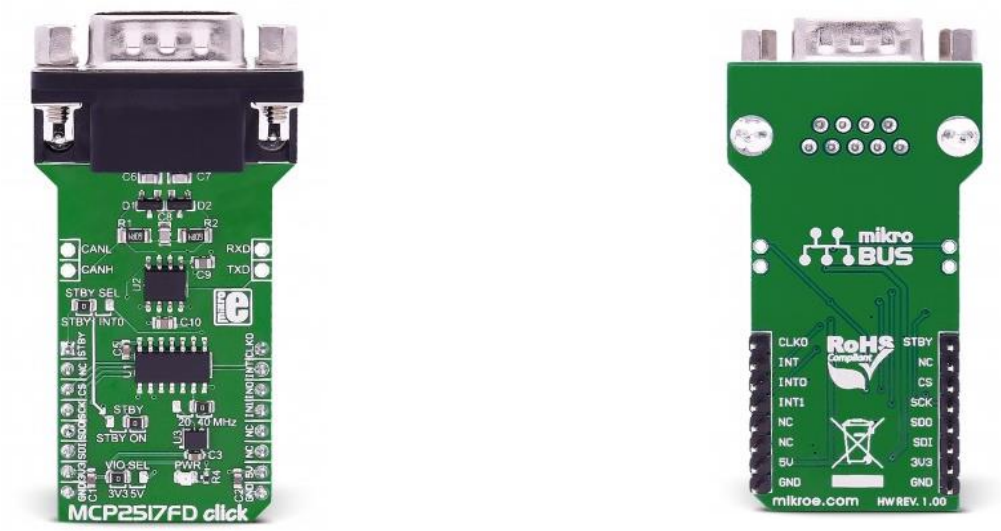
3.6 Hardwarová časť

3.6.1 Výber hardwaru

Hardvér nášho projektu sa skladá z viacerých častí a to hlavne z vývojovej dosky (Obr. 2), prevodníka, ktorý zabezpečuje komunikáciu medzi vývojovou doskou a výrobnou linkou (Obr. 3), tlačidlá na zabezpečenie komunikácie užívateľa s vývojovou doskou (Obr. 4), powerbanka na napájanie bez zbytočných káblov (Obrázok PWB1 ešte nevybraná), krabička respektíve obal celého zariadenia, ktorú si sami navrhneme a vytlačíme (Obrázok).

Vývojovú dosku sme vyberali kvôli špecifikáciám, ktoré je nám potrebné na tento projekt. Doska musí mať procesor, s ktorým môžeme pracovať a konfigurovať, USB port, displej pre nastavovanie rýchlosti prenosu dát a iných premenných, ktoré je potrebné nastavovať počas prevádzky. Existujú dosky, ktoré majú CAN alebo CAN FD port už rovno vstavané ale problém takýchto dosiek je veľkosť a cenová kategória, keďže môžu v niektorých prípadoch dosiahnuť ceny aj desaťkrát dosky ktorá nám postačí na spracovanie a ukladanie dát aj keď nemá vstavaný port CAN FD. Takýto port si môžeme nie až tak zložito pripojiť na dosku externe cez nami vybratý CAN FD click (Obr. 3) celým menom MCP2517FD CLICK

Obr. 2 STM32F429 Discovery Kit (https://hallroad.org/stm32f429-439-arm-cortex-m4-development-kit-in-pakistan.html)



Obr. 3 MCP2517FD Click (https://www.mikroe.com/mcp2517fd-click)

SPI

Kedže tento CAN FD click funguje že prijíma dáta typu CAN a odosiela typ dát SPI tak potrebujeme vedieť čo SPI je ako také. SPI (Serial Peripheral Interface) patrí k najviac používaným rozhraniam pre komunikáciu vnútri vstavaných systémov. Slúži na pripojenie rôznych druhov periférií k riadiacej jednotke. Rozhranie bolo vyvinuté firmou Motorola a stalo sa de facto štandardom. Rozhranie SPI podporujú rôzne druhy snímačov (teplomery, akcelerometre, magnetometre atď.), prevodníky, audio kodeky, pamäte (EEPROM, Flash, SD karty), displeje a pod.

Vlastnosti:

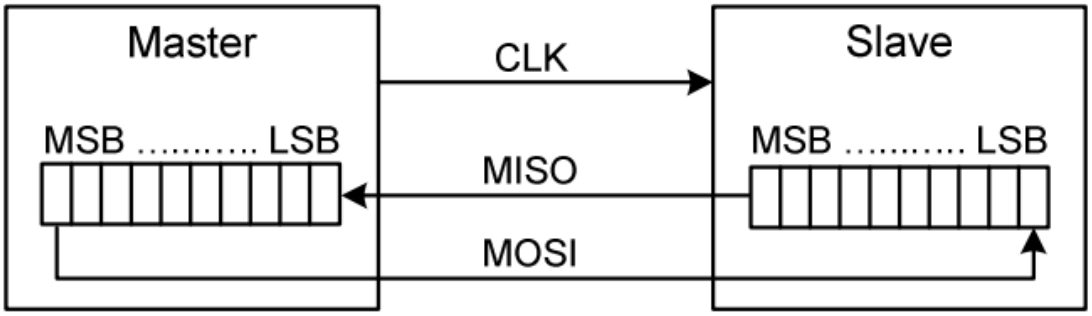
- synchrónna komunikácia (so samostatnými hodinami)
- Master-Slave (jeden Master, viac Slave-ov)
- plný duplex
- rýchlosť nie je definovaná, je možná aj viac ako 100 Mb/s
- rôzna šírka slova (zvyčajne 8 až 16 bitov)
- zvyčajne MSB first

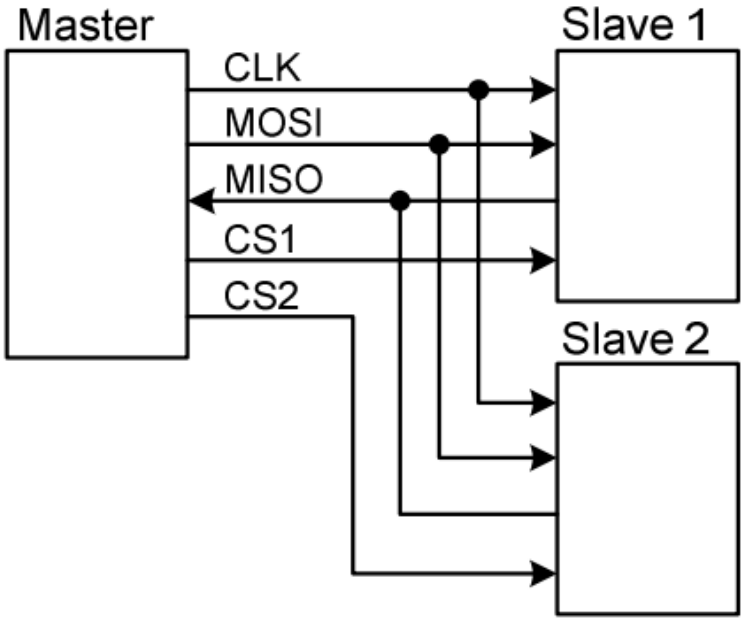
Na komunikáciu sa používajú 4 signály:

- MISO (Master In Slave Out) – vstup dát do Master-a
- MOSI (Master Out Slave In) – výstup dát z Master-a
- CS (Chip Select, SS – Slave Select) – výber aktívneho Slave-a. Tento signál je aktívny v nule.

- SCLK (Serial Clock) – hodinový signál

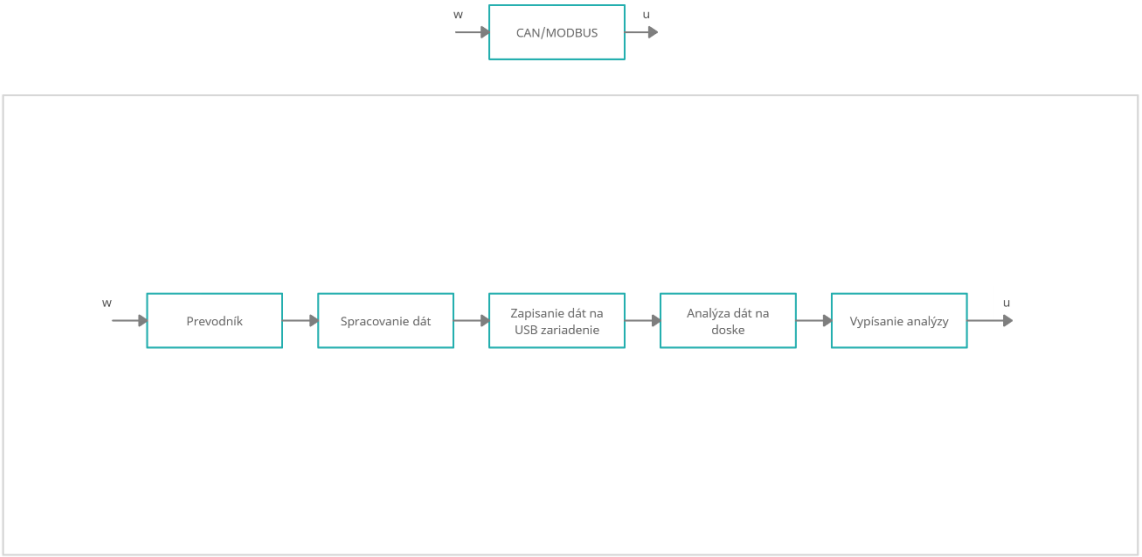
Princíp synchrónnej sériovej komunikácie je znázornený na obrázku. Na realizáciu takejto komunikácie stačia len posuvné registre. Prenos dát je vždy obojsmerný, t.j. počas vysielania dát sa vždy aj prijímajú dáta.





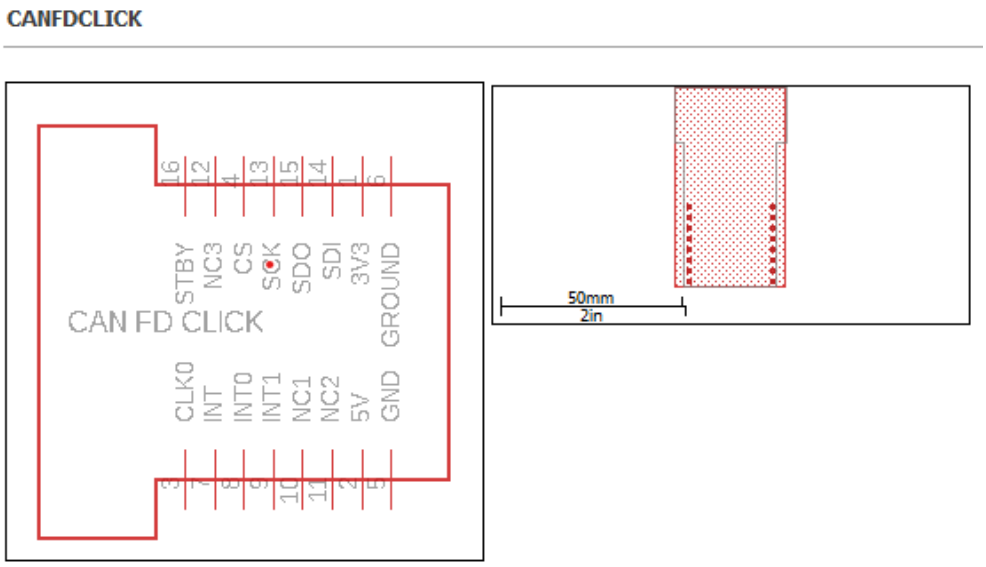
Obr. 4 Tlačidlá (https://ema-elektro.sk/elektromaterial/spinace-prepinace-tlacidla/tlacidlove-spinace/spinac-stlacaci-10mm--stvorec-cierny-250v-1a)

3.6.2 Bloková schéma



3.6.3 Schéma zapojenia

Kedže nám treba schému zapojenia je potrebné si nakresliť súčiastky v príslušnom programe (Eagle). Súčiastky musíme presne nakresliť aby sme vedeli aké rozmery nám bude treba neskôr pre krabičku a preto si musíme odmerať súčiastky alebo najlepšie ak si nájdeme dokumentáciu ku jednotlivým súčiastkám kde sú presne napísané rozmery. Súčiastky je potrebné vytvoriť ako knižnice a postupne zakresľovať symboly, a odtlačky (footprint). Tutoriálov na tvorbu vlastných súčiastok nájdeme na internete veľa. Samostná krabička sa bude kresliť v 3D programe Autodesk Fusion.



Mikrokontrolér STM32F429ZIT6 s 2 MB pamäte Flash, 256 kB RAM v balení LQFP144

• 2,4" QVGA TFT LCD

• USB OTG s konektorom Micro-AB

• I3G4250D, snímač pohybu ST MEMS 3-osový digitálny výstupný gyroskop

• Šesť LED diód:

– LD1 (červená/zelená) pre USB komunikáciu

– LD2 (červená) pre zapnutie 3,3 V

– Dve používateľské LED diódy: LD3 (zelená), LD4 (červená)

– Dve LED diódy USB OTG: LD5 (zelená) VBUS a LD6 (červená) OC (nadprúd)

• Dve tlačidlá (používateľské a resetovacie)

• 64-Mbit SDRAM

• Rozširujúca hlavička pre I/O LQFP144 pre rýchle pripojenie k prototypovej doske a ľahké sondovanie

• Palubný ST-LINK/V2-B

• Funkcie USB:

– Ladiaci port

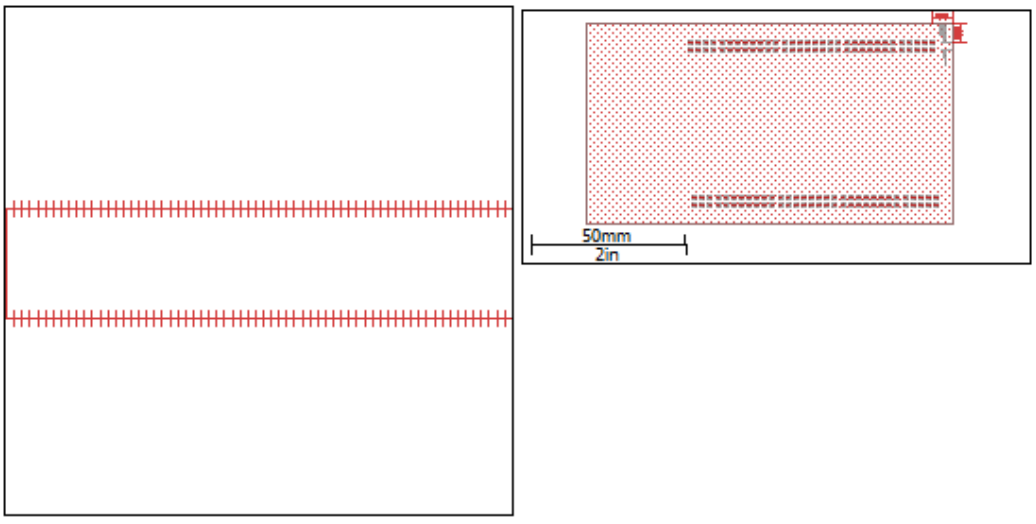
– Virtuálny COM port

- Velke ulozisko

• Mbed Enabled™ (pozri http://mbed.org)

• Napájanie dosky: cez USB zbernicu alebo z externého 5V napájacieho napätia

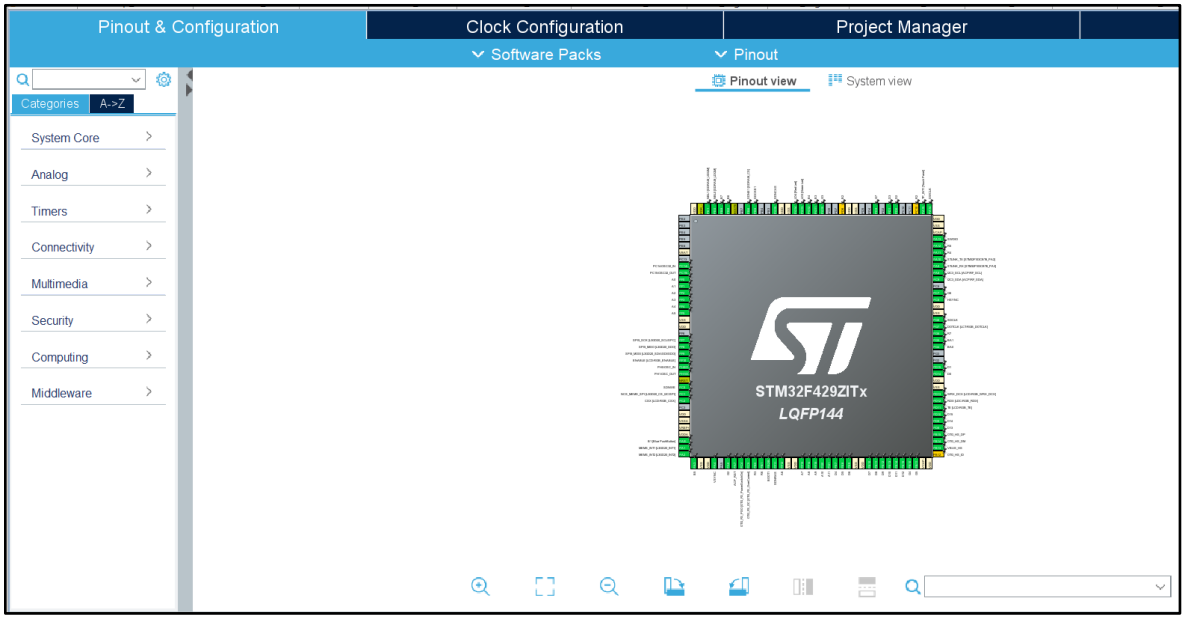
STM32F429I



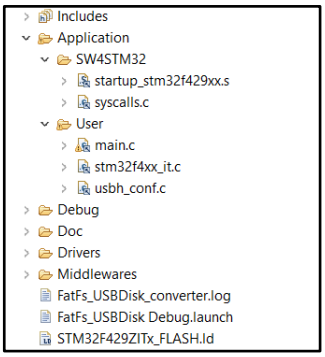
4 Praktická časť práce

4.1 Programovanie mikrokontroléra

Na písanie kódu sme používali software STM32CubeIDE, ktorý nám pomohol a uľahčil celý vývoj. Pomocou grafického konfigurátora sme si nakonfigurovali všetky piny, ktoré používame a nastavili sme systémový clock. Keď sme mali všetko správne nakonfigurované IDE nám vygenerovalo programovú štruktúru do ktorej sme mohli písať kód.



Obr. 3 Grafický konfigurátor STM32CubeIDE



Obr. 4 Programová štruktúra

4.1.1 Počiatočný stav

Východiskovým bodom tohto projektu bol už vytvorený súbor knižníc, o ktorých sa bude diskutovať neskôr, pričom najdôležitejšou z nich je “drc_candfspi_api.c”,ktorá bola vytvorená výrobcom CAN/SPI prevodníka Microchip Technology Inc. Táto knižnica je zodpovedná za konfiguráciu prevodníka a zhromažďovanie všetkých potrebných funkcií na prácu s ním.

Preto, keďže *drc_candfspi_api* bol navrhnutý na iný účel, ďalším krokom bolo prerobiť ho tak, aby bol funkčný na komunikáciu s STM32 board namiesto Raspberry Pi. Na to bolo potrebné odstrániť všetky funkcie a časti kódu súvisiace s USB pripojením a znova ich vytvoriť, aby fungovali s protokolom SPI.

Pred vykonaním akejkoľvek úpravy knižnice bolo potrebné dobre porozumieť spôsobu, akým funguje v každej časti, aby sme vedeli, ktoré prvky sa musia zmeniť a ako, a preto to bude vysvetlené.

Pred vykonaním akejkoľvek úpravy knižnice bolo potrebné dobre porozumieť spôsobu, akým funguje v každej časti, aby ste vedeli, ktoré prvky sa musia zmeniť a ako, a preto to bude vysvetlené.

drc_candfspi_api obsahuje triedu CANFD_SPI. Táto trieda definuje objekt, ktorý bude reprezentovať samotnú komunikáciu SPI a bude to kontajner všetkých nasledujúcich funkcií, nazývaných metódy triedy. Nakoniec, tieto funkcie možno klasifikovať do rôznych skupín alebo sekcií, kde každá z nich vykonáva špecifickú úlohu, a budú diskutované nižšie.

Na prvých riadkoch pôvodného drc_candfspi_api.c sú zavolané všetky potrebné knižnice (konštanty, triedy, binascii,...). Tieto knižnice obsahujú konštantné parametre a základné funkcie, ktoré sa budú opakovane používať v drc_candfspi_api.c. Potom sa zavolá metóda APP_CANFDSPI_INIT na vytvorenie objektu CANFD_SPI, pričom sa nastavia všetky potrebné parametre na jeho definovanie.

Ďalším krokom je vytvorenie všetkých premenných, ktoré budú používať nasledujúce funkcie (vysielacie a prijímacie vyrovnávacie pamäte, chybové príznaky, prevádzkový režim... atď.) a ich nastavenie na predvolené hodnoty, ktoré sa importujú z knižnice konštánt.

Prvá funkcia nazývaná „INITIALIZE“ sa používa na prípravu prevodníka CAN FD na použitie jeho resetovaním do default stavu a nastavením konfiguračných registrov. Následne sú definované základné funkcie, ktoré budú použité na nadviazanie komunikácie cez SPI, ako writeByte a readByte.

Ďalšia skupina funkcií je väčšia a preberá úlohu konfigurácie ovládača (nastavenie registrov FIFO, nastavenie operačného módu, inicializázia RAM). Nakoniec posledná skupina funkcií, o ktorých sa bude dôslednejšie diskutovať v 5.4.2, sú tie, ktoré budú použité v hlavnom skripte dataloggera.

Skladá sa zo zložitejších komunikačných funkcií, ako sú transmitMessageTasks, transmitChannelEventGet a TransmitChannelLoad. Úloha, ktorú vykonávajú pri práci spolu je skontrolovať, či je FIFO queue prázdny pred odoslaním správy, a ak je prázdny, správa sa odošle do kontroléra. Rovnaká operácia sa stane, keď je potrebné prijať správu z ovládača do akéhokoľvek zariadenia.

V našom projekte sme použili aj operačný systém FreeRTOS ktorý nám umožňuje robiť viacero operácií paralelne.

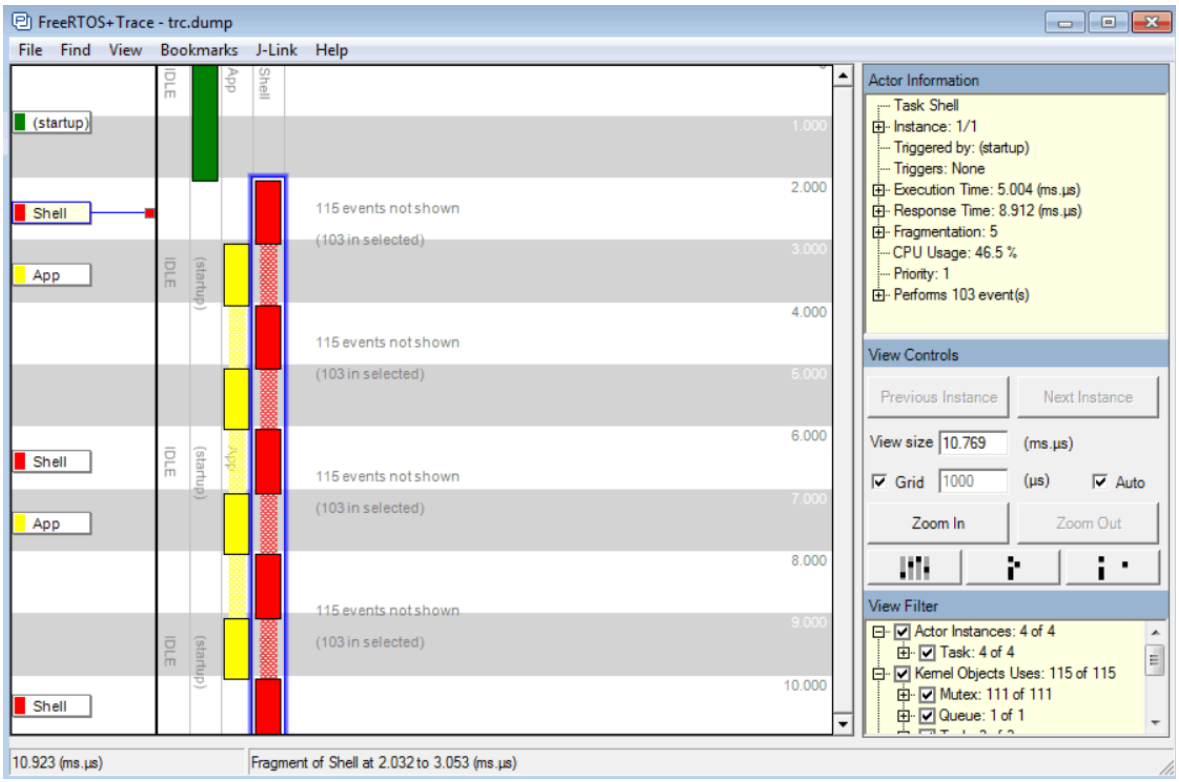
4.1.2 FreeRTOS

FreeRTOS (Free Real-time Operating System) je typ operačného systému, ktorý poskytuje možnosť reagovať na události v okolí počítača priebežne, čiže v realnom čase.

Operačný systém FreeRTOS dokáže počas behu zaznamenávať rôzne systémové udalosti ako napr. presné časovanie prepínania úloh.

Väčšina operačných systémov umožňuje spustenie viacerých programov súčasne. Toto sa nazýva multitasking. V skutočnosti môže každé jadro procesora bežať iba v jednom vlákne vykonávania v akomkoľvek danom časovom bode. Časť operačného systému nazývaná plánovač je zodpovedná za rozhodnutie, ktorý program sa kedy spustí, a poskytuje ilúziu súčasného vykonávania rýchlym prepínaním medzi jednotlivými programami.

Všetky funkcie potrebné pre beh FreeRTOSu nám vygenerovalo a implementovalo do main skriptu STM32CubeIDE.



Obr. 5 Vizualizácia úloh FreeRTOS na jednotlivých vláknach

4.1.3 Záznam správ

Na prvom vlákne procesora nám beží záznam CAN správ. Proces záznamu správ sa delí na tri časti.

4.1.3.1 Prenos správ

Prvá časť slučky má za cieľ poslať správu CAN FD do riadiacej jednotky. Funkcia zodpovedná za túto akciu sa nazýva transmitMessageTasks a berie ako parametre predtým vytvorený identifikátor, txd a DLC. TransmitMessageTasks je rozdelené do troch krokov, ktoré budú vysvetlené nižšie. Po prvé, transmitMessageTasks definuje hodnoty všetkých parametrov, ktoré tvoria správu CAN FD (SID, EID, DLC, FDF, IDE…).

Druhou úlohou, ktorú táto funkcia vykonáva, je skontrolovať register, ktorý zobrazuje stav TX flags, volaním funkcie transmitChannelEventGet. Flags TX sú zodpovedné za hlásenie stavu kanála FIFO nakonfigurovaného ako vysielací kanál a používajú sa na kontrolu, či je plný alebo nie. Ak TX FIFO kanál nie je plný, znamená to, že správu možno odoslať a ak sa zistí, že kanál je plný, prenos sa preruší.

Poslednou úlohou je potom pripraviť správu CAN FD a odoslať ju. Táto operácia je vyvinutá funkciou transmitChannelLoad, ktorá vytvára vyrovnávaciu pamäť dát, ktorú vyplní každá časť celej správy CANFD v správnom poradí a nakoniec ju odošle do kontroléra s funkciou writeByteArray.

4.1.3.2 Príjem správ

Keď prevodník prijme správu CAN, STM32 ju prečíta. Funkcia, ktorá sa na to volá, je receiveMessageTasks, ktorá má podobnú štruktúru ako transmitMessageTasks, ale bez prvého kroku. Flags RX sa čítajú volaním funkcie receiveChannelEventGet. Ak kanál RX FIFO nie je prázdny, zavolá sa funkcia receiveMessageGet na prečítanie správy v kanáli RX FIFO. Ak je prázdny, úloha je dokončená

Nakoniec, keď bola správa CAN FD prečítaná z prevodníka CAN FD, posledná časť cyklu má za cieľ vytvoriť textový súbor a potom doň uložiť každú správu prijatú z ovládača, ktorá zobrazuje históriu všetkých získaných správ.

4.1.4 Zapisovanie správ na USB disk

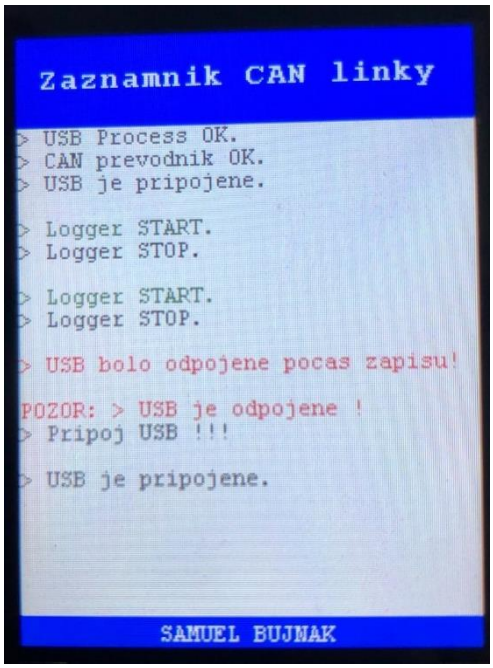
Knižnica FATFS (HAL LIB 20) je „všeobecná“ knižnica pre všetky implementácie súvisiace s FAT, ako sú SDCARD, USB FLASH, SPI FLASH a tiež SDRAM sa dá použiť so správnou inicializáciou FAT.

Pri zapisovaní správ postupujeme nasledovane:

1. f_mount – funkcia zaregistruje pracovnú oblasť zväzku
2. f_open – funkcia otvorí súbor, ak nie je vytvorený tak ho vytvorí
3. f_putc a f_write – tieto funkcie používame na samotný zápis do súboru
4. f_close – funkcia na zatvorenie súboru po zápise
5. f_mount – funkcia odpojí prac. oblasť ale musí mať parameter NULL

4.1.5 Displej

Displej nás informuje aká úloha sa práve vykonáva. Na displeji vieme vypisovať chybné hlásenia napr. ak sa odpojí USB počas logovania, displej nás ihneď informuje o odpojení USB. Displej nás informuje aj o tom či prijímame správy a aj o tom či boli všetky nastavenia a periférie pripojené.



Obr. 6 Výpis na displej

4.2 Softvérová aplikácia

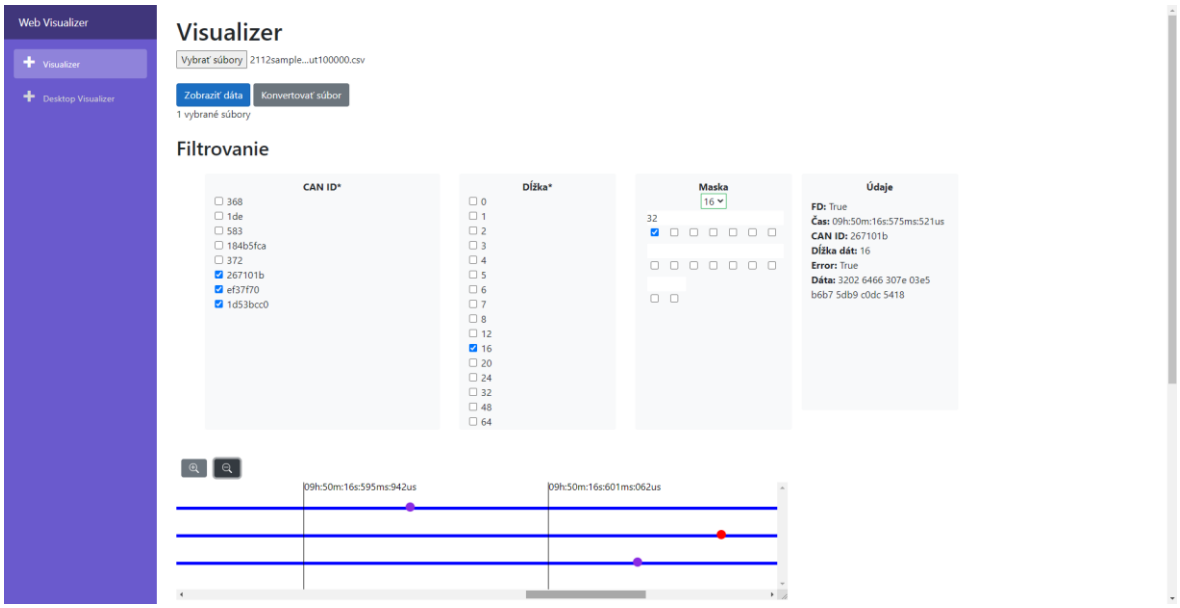
4.2.1 Webová aplikácia

Táto aplikácia slúži na zobrazenie dát, ktoré boli zozbierané záznamníkom. Celá aplikácia beží na strane klienta, aby sa predišlo prenášaniu pomerne veľkého množstva dát cez sieť.

Po nahratí súboru CSV máme možnosť zobraziť dáta priamo v aplikácii alebo ak to používateľ preferuje, môže si tieto dáta konvertovať na súbor formátu VCD, ktorý si následne môže stiahnuť a otvoriť v aplikácii GTKWave. Aplikácia ponúka široké možnosti filtrovania, vďaka čomu sa zobrazia iba dáta, ktoré sú pre používateľa zaujímavé. Konkrétne spôsoby filtrovania sme vyvíjali na základe požiadaviek našich mentorov. Výber filtrov sa prejaví ako na priamej vizualizácii, tak aj v súbore VCD. Filtrovať je možné podľa konkrétnych CAN ID, používateľ si takto vie vybrať správy pochádzajúce len z relevantných kontrolérov. Filter podľa dátovej dĺžky je vhodný pre distingvovanie medzi jednotlivými typmi správ, napríklad tzv. heartbeat (periodicky opakujúce sa správy, ktoré vyjadrujú, že daný kontrolér je práve činný). Zadefinovaním masky je možné rozlíšiť správy ešte presnejšie, a to filtráciou podľa hodnôt konkrétnych bytov v dátovej časti CAN rámca. Maska sa zadáva používateľsky prívetivým spôsobom, a to graficky najprv výberom

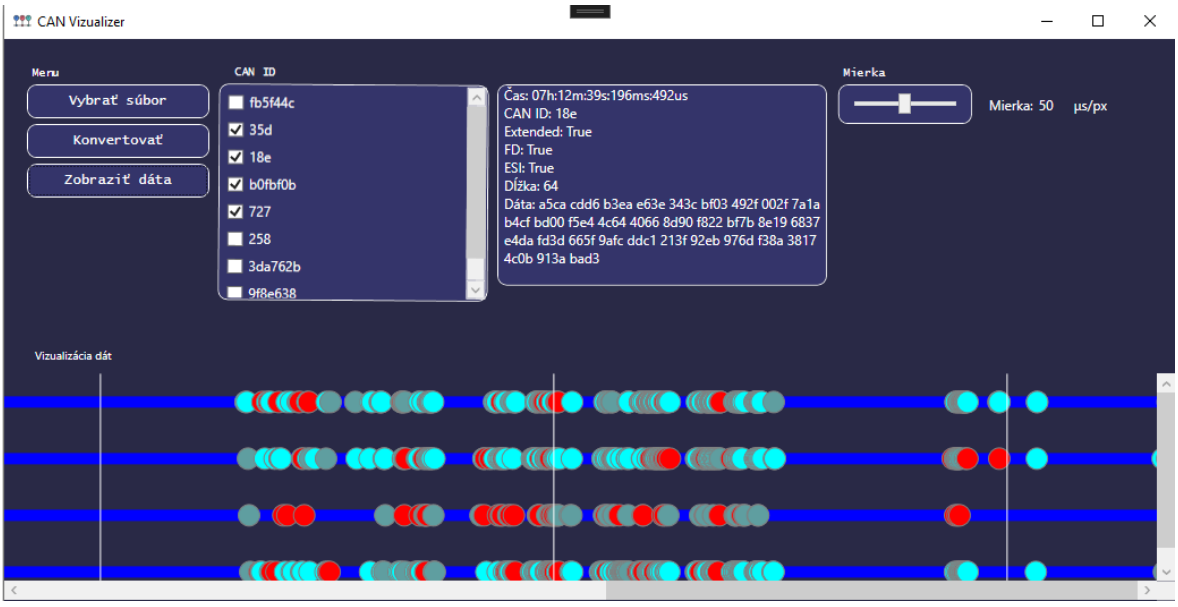
požadovanej dátovej dĺžky a následne zadania hodnôt vybraných bytov v hexadecimálnom tvare. Pomocou checkboxov pod okienkami vyberieme byty, ktoré majú byť programom kontrolované. V sekcii s názvom dáta sa po kliknutí na konkrétny rámec na časovej línii zobrazí jeho obsah.

Vizualizácia je uskutočnená prostredníctvom časových línií. Každá línia predstavuje jedno CAN ID. Na každej línii sú v podobe krúžkov vizualizované jednotlivé rámce, ktoré majú pozíciu presne podľa času, kedy boli zaznamenané.



4.2.2 Desktopová aplikácia

Pri zázname CAN komunikácie pri plnej rýchlosti môže veľkosť výsledného súboru narásť veľmi rýchlo, avšak čítanie takéhoto súboru prostredníctvom prehliadača trvá veľmi dlho. Preto tieto príležitosti sme sa rozhodli vytvoriť desktopovú aplikáciu pre OS Windows. Táto aplikácia má oproti webovej verzii mierne obmedzenú funkcionalitu, no mnohonásobne lepší výkon. Vďaka priamemu prístupu k súborovému systému je možné pri konverzii na súbor VCD spracovať prakticky ľubovoľne veľký súbor, pretože dáta môžeme priamo zapisovať do súboru nemusia ukladať do operačnej pamäte a teda ich objem nie je limitovaný veľkosťou pamäte RAM.



4.3 Testovanie softvérovej aplikácie

4.3.1 Testovanie aplikácií

Názorne testovanie aplikácii nám bolo ukázane pri návšteve ComAp pobočky v Košiciach. Boli nám vysvetlené jednotlivé test cases, ktoré boli použité pri testovaní reálnych produktov v praxi. Príklad jednoduchého test casu by mal vyzerať takto:

Test Case ID	Popis testu	Testovacie kroky	Testovacie dáta	Očakávaný výsledok	Reálny výsledok	Pass/Fail
TU01	Overenie zapnutia/vypnutia aplikácie	1. Zapnutie aplikácie 2. Vypnutie aplikácie	CAN Visualizer.exe	Funkčné zapnutie a vypnutie aplikácie	Ako očakávaný	Pass

Tieto test cases sú používane aj pri reálnom testovaní softvérových aplikácii. Pri testovaní aplikácii som potreboval vygenerované vstupné dáta. Preto som si vytvoril jednoduchý generátor CAN správ, ktorý mi vygeneruje náhodne správy do CSV súboru s ktorým potom môžem testovať softvérové aplikácie.

4.3.2 Testovanie desktopovej aplikácie

Proces testovania desktopovej aplikácie začal zoznámením s aplikáciou a následným testovaním. Zo začiatku som sa snažil desktopovú aplikáciu takzvane „pokaziť“. Skúšal som zapnúť všetky rôzne tlačidlá, vyklikávanie údajov, minimalizácia, maximalizácia atď.

Potom som prešiel na vkladanie vhodných aj nevhodných súborov, stlačenie iných tlačidiel akých by bolo potrebné, odpojenie od internetu, vkladanie viacerých súborov a vkladanie veľkých súborov. Všetky test case boli odovzdané softvérovému vývojárovi.

Test Case ID	Popis testu	Testovacie kroky	Testovacie dáta	Očakávaný výsledok	Reálny výsledok	Pass/ Fail
TU01	Overenie funkčnosti desktopovej aplikácie	1.Zapnutie/Vypnutie aplikácie 2.Vloženie csv súboru do aplikácie 3.Uloženie vcd výstupu 4.Kontrola dát	CanInput1.csv	Funkčnosť aplikácie	Ako očakávaný	Pass
TU02	Vloženie vhodného vstupného súboru	1.Zapnutie aplikácie 2.Vloženie vhodného csv súboru 3.Kontrola dát	CanInput1.csv	Správne fungovanie programu pri vhodne vloženom vstupnom súbore	Ako očakávaný	Pass
TU03	Vloženie nevhodného vstupného súboru	1.Zapnutie aplikácie 2.Vloženie nevhodného súboru 3. Zobrazenie dát	Doc.docx Note.txt	Vypísanie chyby pri nevhodne vloženom súbore	Program spadne pri vložení nevhodného súboru	Fail

TU04	Vloženie poškodeného csv súboru	1.Zapnutie aplikácie 2.Vloženie poškodeného csv súboru 3.Zobrazenie chybných dát	CanInputC 1.csv	Vypísanie chyby pri poškodenom súbore	Program nezobrazí časovú os ani údaje	Fail
TU05	Vybranie jedného CAN ID	1.Zapnutie aplikácie 2.Vloženie vhodného csv súboru 3.Vybranie jedného ID z CAN Ids 4.Zobrazenie vizualizácie dát	CanInputC 1.csv	Správne fungovanie programu a zodpovedajúca vizualizácia na základe vybraného CAN ID	Ako očakávaný	Pass
TU06	Vybranie viacerých CAN Ids	1.Zapnutie aplikácie 2.Vloženie vhodného súboru 3.Vybranie viacerých Ids z CAN Ids 4.Zobrazenie vizualizácie dát	CanInputC 100.csv	Správne fungovanie aplikácie a zodpovedajúce vizualizácie na základe vybraných CAN Ids	Ako očakávaný	Pass
TU07	Nevybratie žiadného CAN ID	1.Zapnutie aplikácie 2.Vloženie testovacieho súboru 3.Žiaden výber CAN ID 4.Zobrazenie dát	CanInputC 1.csv	Plynulé pokračovanie aplikácie bez ďalších problémov	Ako očakávaný	Pass
TU08	Vybranie všetkých CAN Ids	1.Zapnutie aplikácie 2.Vloženie testovacieho csv súboru 3.Vybranie všetkých CAN Ids 4.Zobrazenie vizualizácii dát	CanInputC 100.csv	Plynulé pokračovanie aplikácie	Ako očakávaný	Pass
TU09	Vloženie veľkého objemu nasnímaných dát	1.Zapnutie aplikácie 2.Vloženie veľkého csv súboru 3.Zobrazenie dát	CanInputC 1000000.csv	Bezproblémové načítanie a zobrazenie dát pri veľkom súbore	Padnutie programu pri vložení veľkého množstva dát	Fail

TU10	Zobrazenie dát pri vložení csv súboru	1.Zapnutie aplikácie 2.Vloženie správneho csv súboru 3.Vizualizácia vložených dát	CanInputC 1.csv	Správna vizualizácia vložených dát	Ako očakávaný	Pass
TU11	Zobrazenie potrebných údajov pri vizualizácii dát	1.Zapnutie aplikácie 2.Vloženie správneho csv súboru 3.Vizualizácia csv vložených dát 4.Overenie údajov pre vizualizáciu	CanInputC 1.csv	Správne zobrazenie všetkých údajov v tabuľke údajov	Nesprávne zobrazenie niektorých údajov z tabuľky údajov	Fail
TU12	Nastavenie mierky	1.Zapnutie aplikácie 2.Vloženie csv súboru 3.Zmena mierky	CanInputC 100.csv	Správne zobrazenie dát po zmene mierky	Ako očakávaný	Pass

4.3.3 Testovanie webovej aplikácie

Proces testovania webovej aplikácie začal zoznámením s webovou aplikáciou a následným testovaním. Zo začiatku som sa snažil webovú aplikáciu takzvane „pokaziť“. Skúšal som zapnúť všetky rôzne tlačidlá, obnovenie stránky, vyklikávanie údajov atď.

Potom som prešiel na vkladanie vhodných aj nevhodných súborov, stlačenie iných tlačidiel akých by bolo potrebné, odpojenie od internetu, vkladanie viacerých súborov a vkladanie veľkých súborov. Všetky test case boli odovzdané softvérovému vývojárovi.

Test Case ID	Popis testu	Testovacie kroky	Testovacie dáta	Očakávaný výsledok	Reálny výsledok	Pass/ Fail
TU01	Overenie funkčnosti webovej aplikácie	1.Načítanie aplikácie 2.Vloženie csv súboru do aplikácie 3.Uloženie vcd výstupu 4.Kontrola dát	CanInput1. csv	Funkčnosť aplikácie	Ako očakávaný	Pass

TU02	Vloženie vhodného vstupného súboru	1. Načítanie aplikácie 2.Vloženie vhodného csv súboru 3.Kontrola dát	CanInput1.csv	Správne fungovanie webovej aplikácie pri vhodne vloženom vstupnom súbore	Ako očakávaný	Pass
TU03	Vloženie nevhodného vstupného súboru	1. Načítanie aplikácie 2.Vloženie nevhodného súboru 3. Zobrazenie dát	Doc.docx Note.txt	Vypísanie chyby pri nevhodne vloženom súbore	Ako očakávaný	Pass
TU04	Vloženie poškodeného csv súboru	1. Načítanie aplikácie 2.Vloženie poškodeného csv súboru 3.Zobrazenie chyby dát	CanInput1.csv	Vypísanie chyby pri poškodenom súbore	Aplikácia spadne	Fail
TU05	Vybranie jedného CAN ID	1. Načítanie aplikácie 2.Vloženie vhodného csv súboru 3.Vybranie jedného ID z CAN Ids 4.Zobrazenie vizualizácie dát	CanInput1.csv	Správne fungovanie programu a zodpovedajúca vizualizácia na základe vybraného CAN ID	Ako očakávaný	Pass
TU06	Vybranie viacerých CAN Ids	1. Načítanie aplikácie 2.Vloženie vhodného súboru 3.Vybranie viacerých Ids z CAN Ids 4.Zobrazenie vizualizácie dát	CanInput100.csv	Správne fungovanie aplikácie a zodpovedajúce vizualizácie na základe vybraných CAN Ids	Ako očakávaný	Pass

TU07	Nevybratie žiadneho CAN ID	1. Načítanie aplikácie 2.Vloženie testovacieho súboru 3.Žiaden výber CAN ID 4.Zobrazenie dát	CanInput1.csv	Plynulé pokračovanie aplikácie bez ďalších problémov	Ako očakávaný	Pass
TU08	Vybranie všetkých CAN Ids	1. Načítanie aplikácie 2.Vloženie testovacieho csv súboru 3.Vybranie všetkých CAN Ids 4.Zobrazenie vizualizácii dát	CanInput100.csv	Plynulé pokračovanie aplikácie	Ako očakávaný	Pass
TU09	Vloženie veľkého objemu nasnímaných dát	1. Načítanie aplikácie 2.Vloženie veľkého csv súboru 3.Zobrazenie dát	CanInput1000000.csv	Rýchle načítanie a zobrazenie dát pri veľkom súbore	Dlhé načítavanie aplikácie pri vložení veľkého množstva dát	Fail
TU10	Zobrazenie dát pri vložení csv súboru	1. Načítanie aplikácie 2.Vloženie správneho csv súboru 3.Vizualizácia vložených dát	CanInput1.csv	Správna vizualizácia vložených dát	Ako očakávaný	Pass
TU11	Zobrazenie potrebných údajov pri vizualizácii dát	1. Načítanie aplikácie 2.Vloženie správneho csv súboru 3.Vizualizácia csv vložených dát 4.Overenie údajov pre vizualizáciu	CanInput1.csv	Správne zobrazenie všetkých údajov v tabuľke údajov	Nesprávne zobrazenie niektorých údajov z tabuľky údajov	Fail
TU12	Nastavenie mierky	1. Načítanie aplikácie 2.Vloženie csv súboru 3.Zmena mierky	CanInput100.csv	Správne zobrazenie dát po zmene mierky	Ako očakávaný	Pass
TU13	Správanie aplikácie pri výpade internetového pripojenia	1.Načítanie aplikácie 2.Vloženie csv súboru 3.Vypadnutie internetu	CanInput1.csv	Ponechanie súborov aj po výpadku internetu	Ako očakávaný	Pass

4.3.4 Testovanie CAN záznamníka

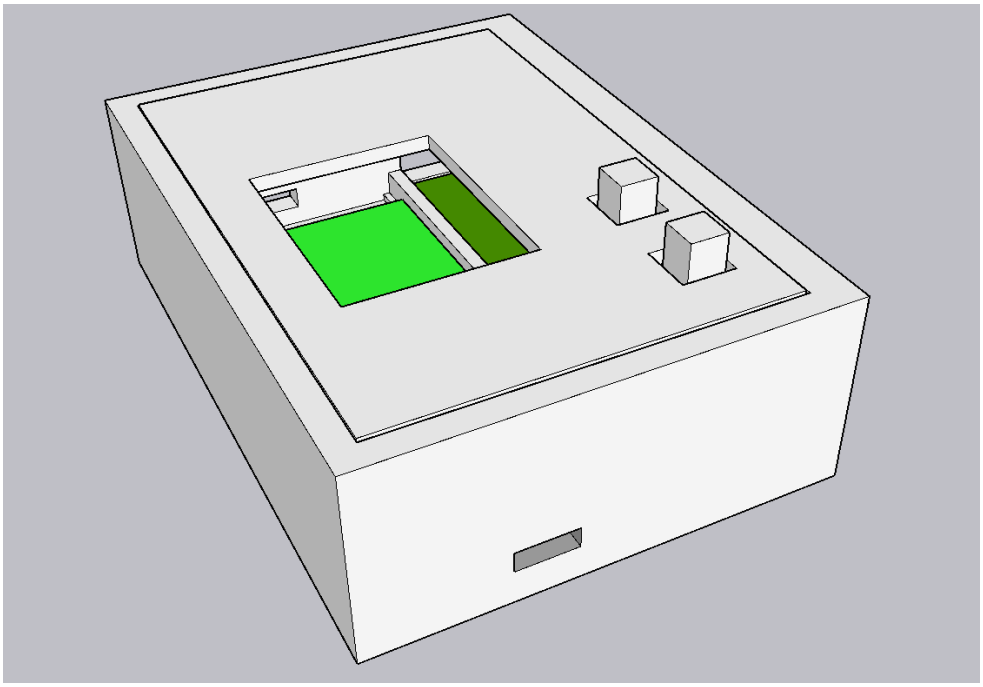
Test Case ID	Popis testu	Testovacie kroky	Testovacie dáta	Očakávaný výsledok	Reálny výsledok	Pass/ Fail
TU01	Overenie funkčnosti CAN záznamníka	1.Zapnutie/vypnutie záznamníka 2.Začatie logovania 3.Skončenie logovania 4.Zapisanie dát na USB disk	CAN záznamník Log.csv	Funkčnosť prevodníka	Ako očakávaný	Pass
TU02	Logovanie dát na záznamníku	1.Zapnutie záznamníka 2.Začatie logovania 3.Skončenie logovania 4.Kontrola logovaných dát	CAN záznamník Log.csv	Správne fungovanie záznamníka pri logovaní dát	Ako očakávaný	Pass
TU03	Držanie tlačidla logovania	1.Zapnutie záznamníka 2.Držanie tlačidla prevodníka 3.Začatie logovania	CAN záznamník	Začatie logovania bez problémov	Záznamník začne a prestane cyklicky logovať	Fail
TU04	Odpojenie USB počas logovania	1.Zapnutie záznamníka 2.Začatie logovania 3.Vytiahnutie USB počas logovania 4.Zastavenie logovania	CAN záznamník	Vypísanie chyby pri vybranom USB	CAN záznamník prijíma správy ale nevypíše chybu na displeji	Fail
TU05	Zapísanie CAN logovaných dát	1.Zapnutie záznamníka 2.Začatie logovania poškodených správ 3.Zastavenie logovania	CAN záznamník	Nezapísanie nalogovaných dát	Ako očakávaný	Pass

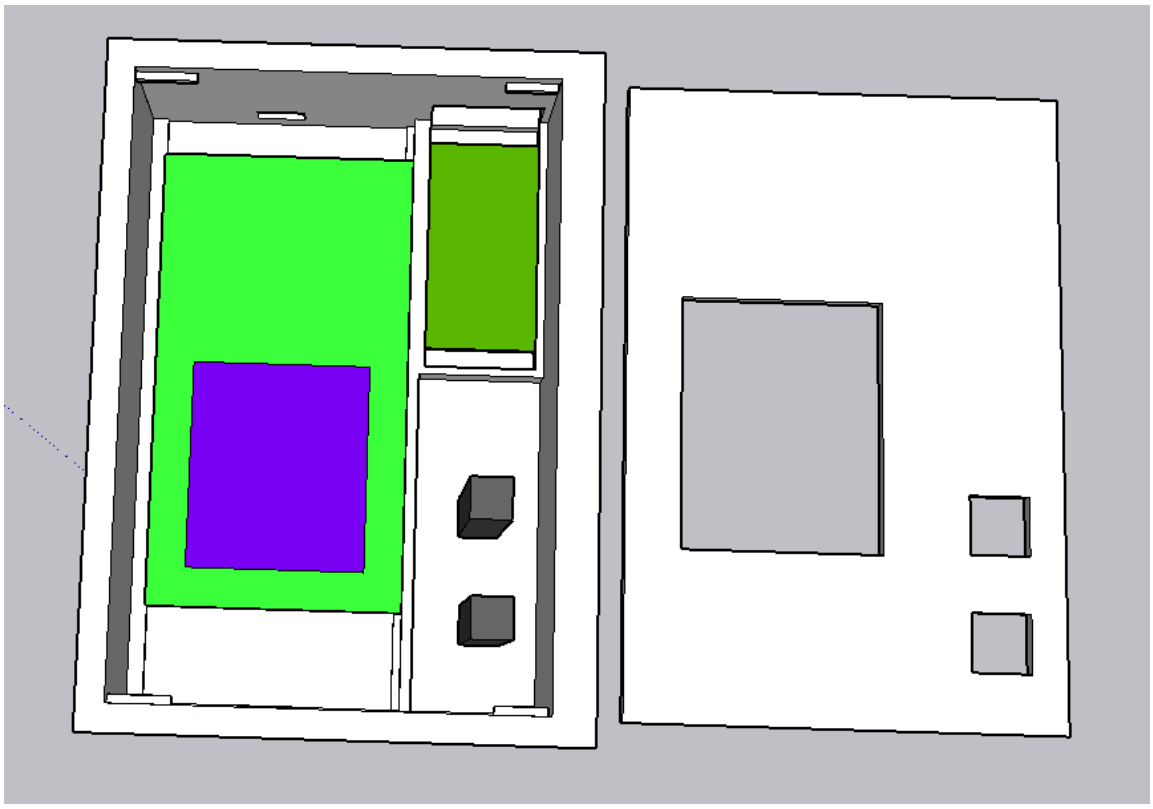
4.4 Hardware

Návrh krabičky v 3D priestore začal oboznámením sa z problematikou krabičky ako takej. Krabičku môže urobiť hocikto ale krabička pre nejaké zariadenie musí mať nejaké paramentre respektíve musí spĺňať požiadavky vopred dané. Požiadavky našej krabičky sú aby sa jednotlivé súčiastky do nej vmestili ale aby takzvane „nelietali“ v krabičke. Malo by tam byť miesto na dve tlačidlá, vývojovú dosku (a s ňou spojený dongle) a CAN prevodník. Obrázok ukazuje 3D model s odnímateľným krytom a dosadenými všetkými súčiastkami pod určitým uhlom (tlačidlá sú ešte biele nie).

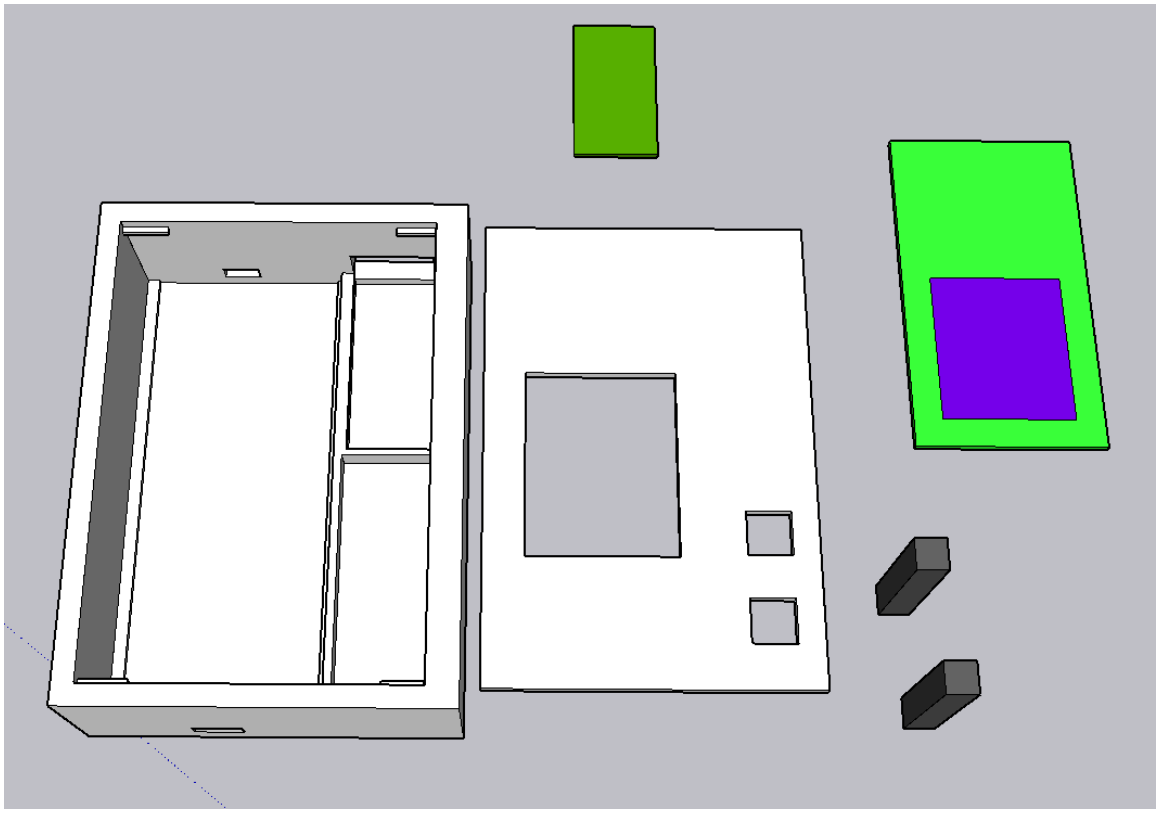
Urobiť kocku alebo kváder problém nerobilo ale urobiť všetky malé aj veľké zmeny programu takzvane vadili. Pri každej zmene sa chceli meniť aj veci spolu nesúvisiace a preto som prešiel z programu Autodesk Fusion 360 na program SketchUp kde sa takéto problémy diali tiež ale menej a bola tam možnosť jednoduchej opravy alebo upravy.

Na presné vyrábanie krabičky som si musel namodelovať aj jednotlivé súčiastky aby som náhodou neurobil chybu a aby sa nestalo niečo také že by to bolo príliš malé alebo veľké a preto je to presné na milimetre. Tmavo zelený kváder je CAN prevodník, svetlozelený kváder s fialovou plôškou je vývojová doska pri čom tá fialová plôška je displej, na ktorý je treba zhora pozerať a čierné kvádre so štvorcovou podstavou sú spomínané tlačidlá. Tento model krabičky je neposledná verzia ale je to verzia, v ktorej sa lepšie orientuje, ale oproti poslednej (reálnej) verzii sú rozdiely v hrúbke strán a vo výške celej krabičky. Kedže krabička je predmet, ktorý má za úlohu chrániť nielen výrobok alebo jednotlivé súčiastky, ale aj človeka tak je vyrobená z pevného nevodivého materiálu.

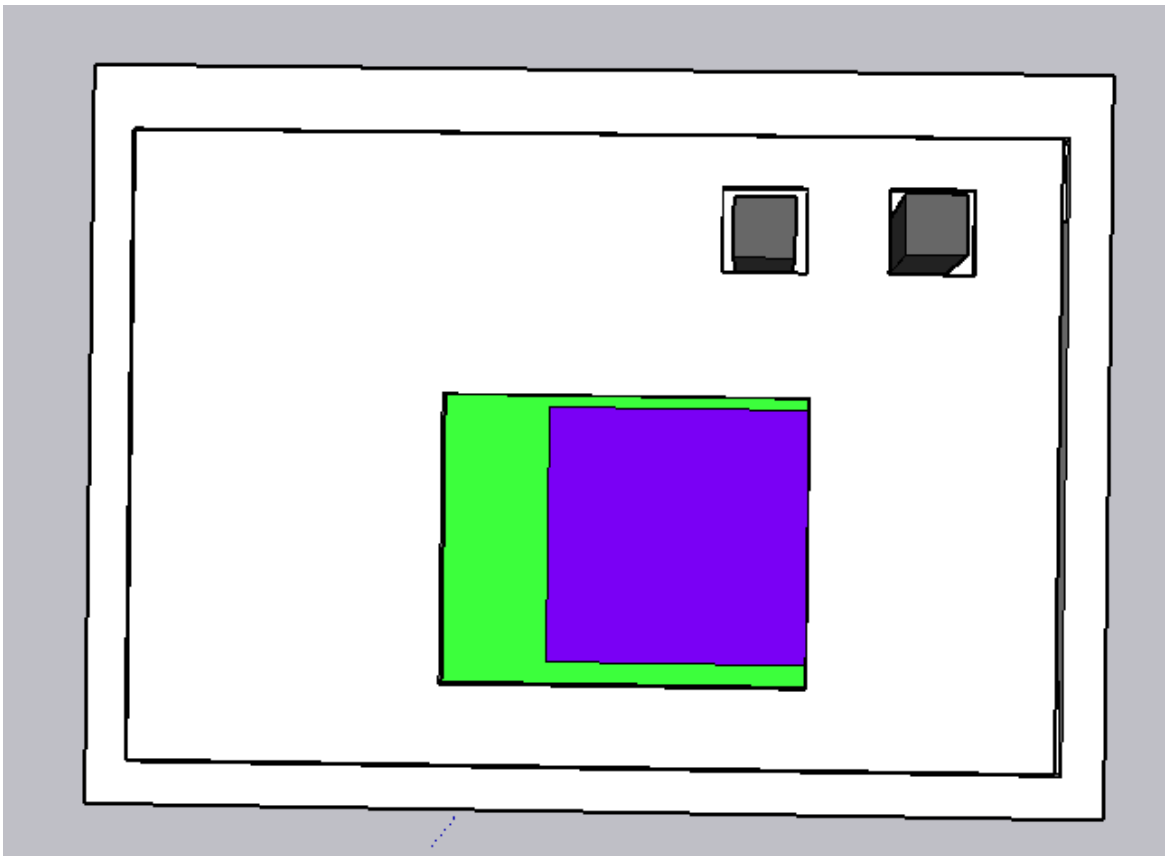




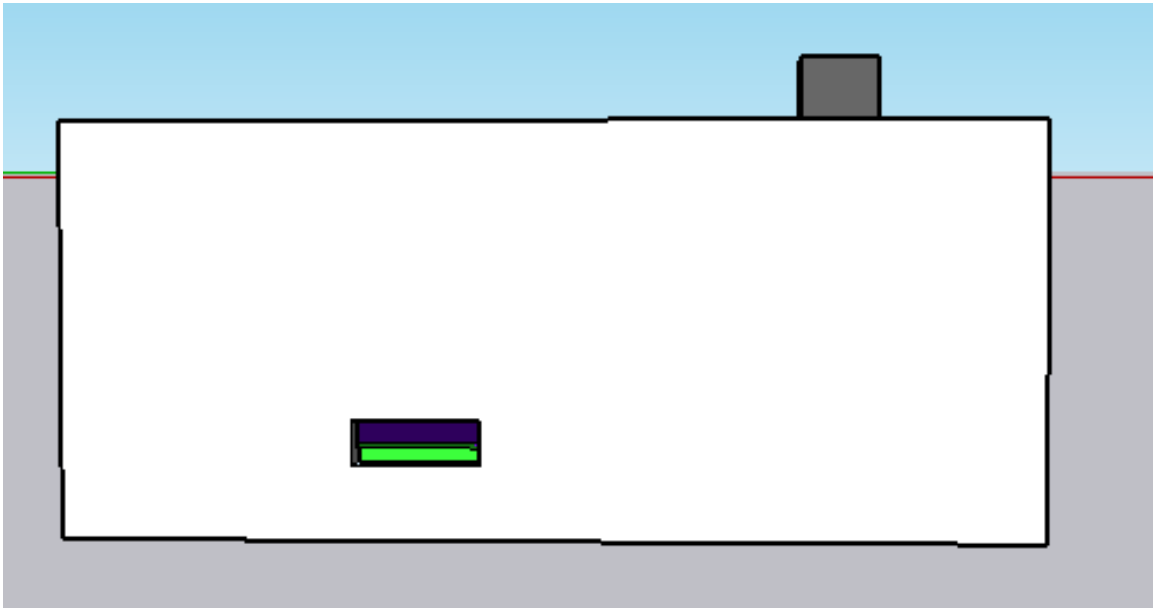
Pohľad s osadenými prvkami



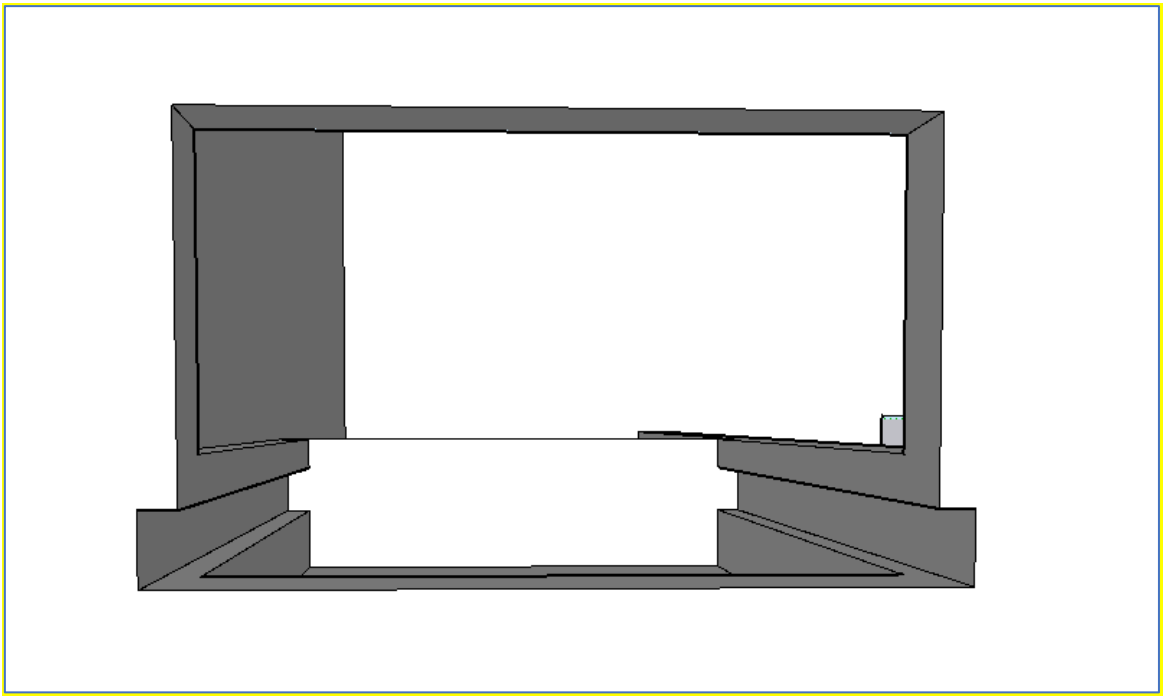
Pohľad s prvkami mimo dosky



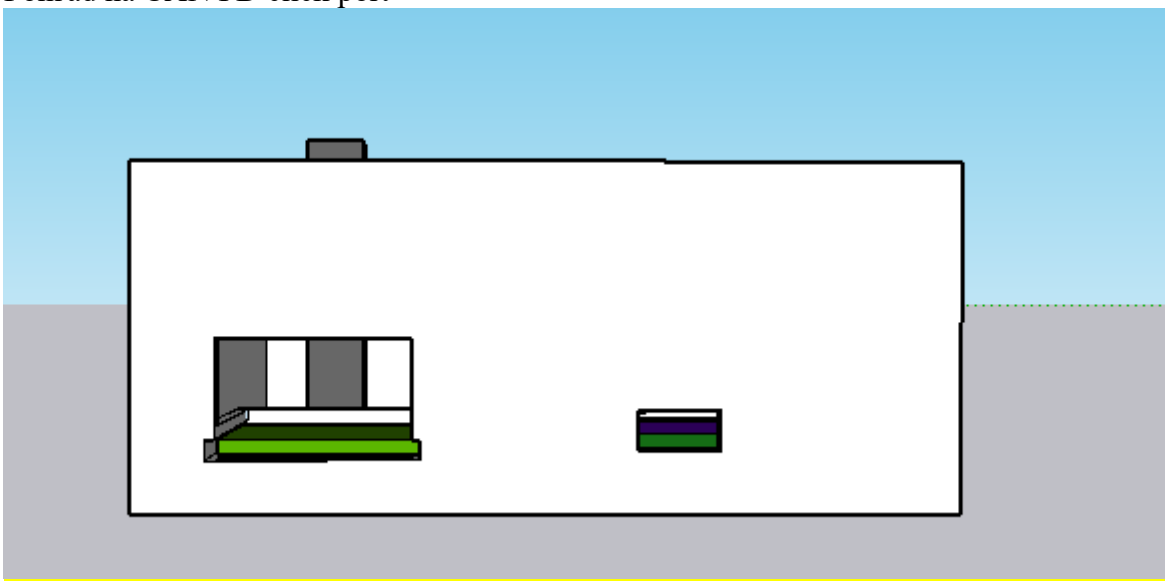
Pohľad zhora



Pohľad spredu

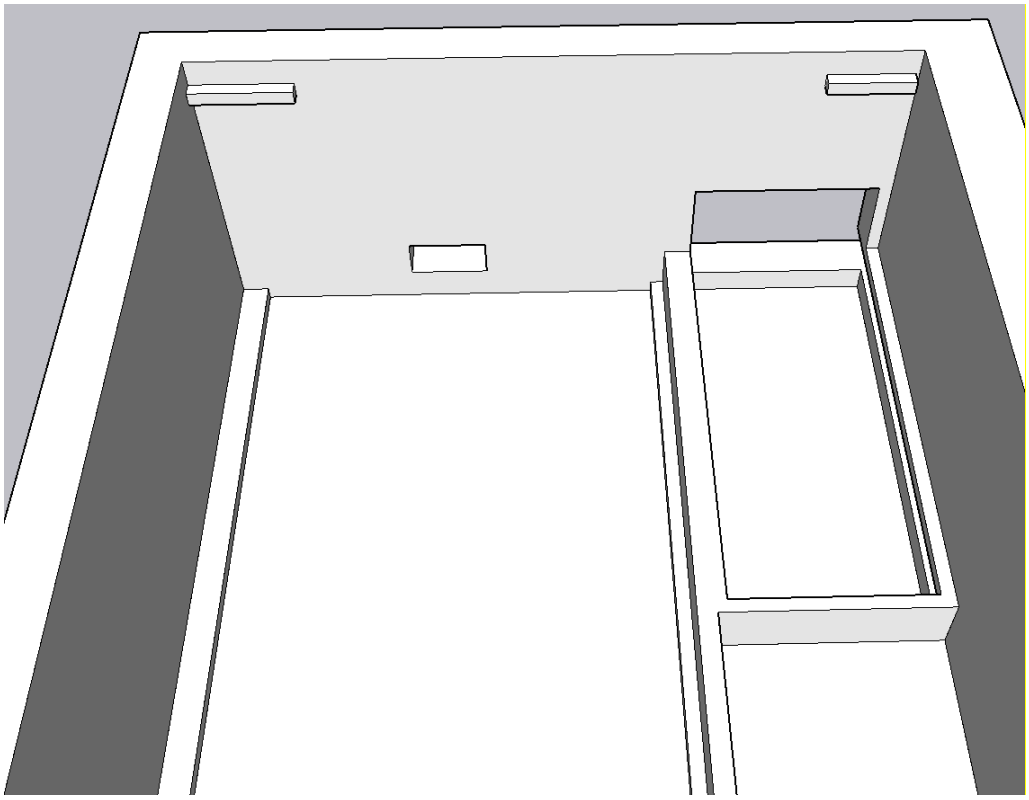


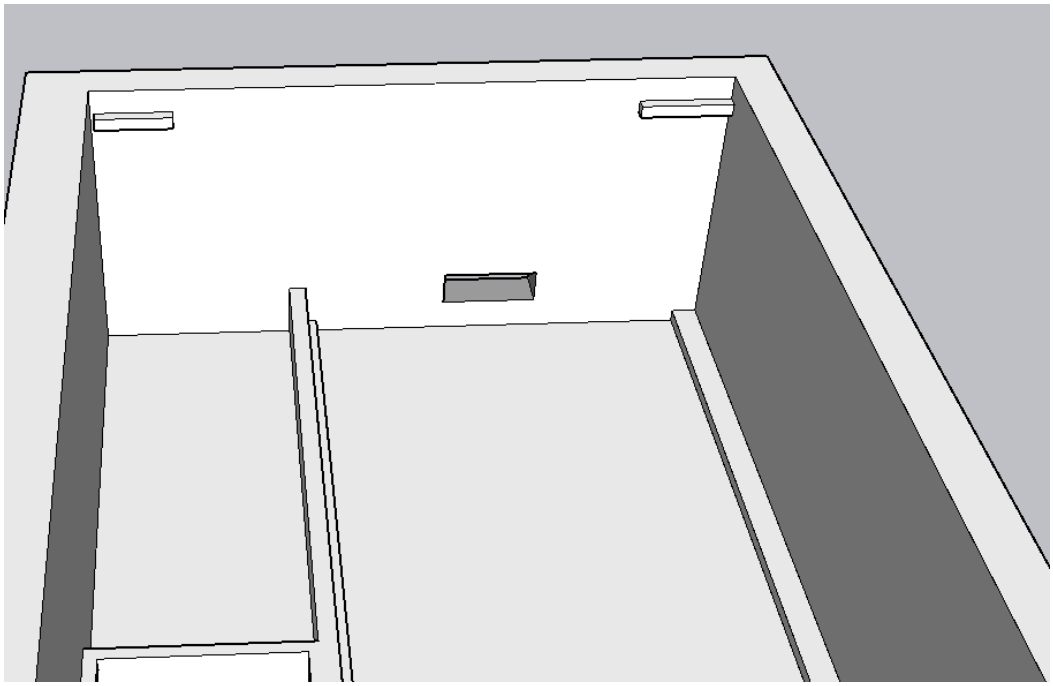
Pohľad na CAN FD click port



Pohľad zozadu

Aby nám pekne sedeli súčiastky tak som vyrobil koľajničky, ktoré nám budú držať súčiastky v jednom smere a v jednej rovine aby neskákali hore a dole. Tieto koľajničky sme urobili iba pomocou schodou hore a dole respektíve 90° ohraničiamy kde bude doska udržiavaná a kde bude mať ešte priestor na pohyb ktorý nám bude treba na zasadenie dosky do krabičky aby správne sedeli USB porty do dierok.





Obrázok pre zaujímavosť: Doska v pomere ku priemerne vysokej žene

5 Výsledky práce

Výsledkom tejto práce bolo zachytávanie dát na CAN linke pomocou CAN záznamníka. Tento CAN záznamník bude umiestnený v špeciálne navrhnutej krabičke pomocou ktorej ho bude ľahko a bez poškodenia možné prenášať na rôzne miesta linky a napájať z akéhokoľvek zdroja s výstupom USB.

Nalogované dáta budú zapísané na USB disku a následne ich môžeme vložiť do počítača, kde ich vložíme do našej vytvorenej webovej aplikácie na vizualizáciu nazbieraných dát. V prípade veľkého množstva nalogovaných dát je možné si stiahnuť desktopovú verziu aplikácie pre lepší výkon. Prípadne si konvertovať výstup nalogovaných dát do vcd súboru, ktorý môžeme otvoriť v aplikácii GTKWave.

Po základnom testovaní záznamníka a softvérovej aplikácie sme opravili chyby, ktoré sme našli pri testovaní.

6 Závery práce

V závere tohto projektu sme došli ku funkčnému CAN záznamníku, ktorý vie logovať správy na CAN linke. Následne dáta uložiť na USB disk a potom ich vložiť do softvérovej aplikácie na vizualizáciu týchto dát.

V hardvérovej časti sme našli vhodné súčiastky, urobili sme blokovú schému. Následne sme vymodelovali krabičku v trojrozmernom prostredí a dosadili jednotlivé prvky do finálneho produktu.

Aplikácia obsahuje funkcie zobrazovania, triedenia, filtrovania podľa CAN ID, dátovej dĺžky alebo masky, taktiež obsahuje funkciu zobrazenia obsahu konkrétnych rámcov. Konvertovanie na VCD taktiež funguje bezproblémovo.

Na hardvéri by sme chceli zlepšiť bezpečnosť krabičky s použitím pántov alebo iných spôsobov na otváranie krabičky a použitie jedného plošného spoja a nie viacero súčiastok.

Práca na projekte nám celkovo trvala 8 mesiacov. Celkové náklady na záznamník dosiahli cenu 80 EUR. Oproti komerčne dostupným CAN záznamníkom je cena približne polovičná.

Zhrnutie

Práca sa zaoberá problematikou logovania správ pomocou CAN záznamníka, ktorý sa pripojí na CAN linku. V práci je opísaná teória CAN-u, vývoj CAN záznamníka, vývoj softvérových aplikácii, testovanie produktu a návrh hardvérovej časti. Výsledok práce je praktický CAN záznamník určený na logovanie CAN správ pri diagnostike CAN linky.

Resumé

The work deals with the issue of logging messages using a CAN logger, which is connected to the CAN bus. The thesis describes CAN theory, development of CAN logger, software application development, product testing and hardware design. The result of the work is a practical CAN logger designed for logging CAN messages for CAN bus diagnostics.

Zoznam použitej literatúry

[1] CAN bus. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-05-08]. Dostupné na: https://cs.wikipedia.org/wiki/CAN_bus

[2] BORTEL, Radoslav. CAN – Controller Area Network [online]. [cit. 2022-05-08]. Dostupné na: http://noel.feld.cvut.cz/vyu/scs/prezentace2004/CAN/#_Toc71182441

[3] POLÁK, Ing. Karel. Sběrnice CAN [online]. 2003/21, 16.6.2003 [cit. 2022-05-08]. Dostupné na: http://www.elektrorevue.cz/clanky/03021/index.html

[4] PAVLIŠIN, Tomáš. Protokol CAN pro řízení [online]. Brno, 2015 [cit. 2022-05-08]. Dostupné na: https://core.ac.uk/download/pdf/30283041.pdf

[5] Know-How CAN. Vector [online]. [cit. 2022-05-08]. Dostupné na: https://www.vector.com/int/en/know-how/can/

[6] JONES, Alex. Rock Solid Knowledge: An introduction to Blazor WebAssembly. Rock Solid Knowledge [online]. Bristol: Rock Solid Knowledge, 2021, 2. 2. 2021 [cit. 2022-02-14]. Dostupné na: https://www.rocksolidknowledge.com/articles/an-introduction-to-blazor-webassembly

[7] MORRIS, Peter. Blazor University: What is WebAssembly?. Blazor University [online].[cit.2022-02-14].Dostupné na: https://blazor-university.com/overview/what-is-webassembly/

[8] Progressive web application. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation,2001-[cit.2022-02-14].Dostupné na: https://en.wikipedia.org/wiki/Progressive_web_application

[9] What is Windows Presentation Foundation (WPF)?. Microsoft [online]. [cit. 2022-05-02]. Dostupné na: https://docs.microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf?view=vs-2022

[10] *Welcome to GTKWave* [online]. [cit. 2022-05-02]. Dostupné na: http://gtkwave.sourceforge.net/

[11] How to Write Test Cases: Sample Template with Examples[online]. [cit. 2022-05-08]. Dostupné na: https://www.guru99.com/test-case.html

[12] Microchip External CAN FD Controller with SPI Interface. Mcp2517Fd datasheet. 2018. Accessed: 2020-02-02

[13] National instruments. CAN FD Explained – A Simple Intro (2020). https://www.csselectronics.com/screen/page/can-fd-flexible-data-rate-intro/language/en Accessed: 2020-02-01

[14] CAN in Automation (CIA). CAN FD – The basic idea. https://www.can-cia.org/canknowledge/can/can-fd/. Accessed: 2020-02-05

[15] Texas Instrumentals. What do CAN bus signals look like? https://e2e.ti.com/blogs_/b/industrial_strength/archive/2015/06/04/what-do-can-bussignals-look-like Accessed: 2020-04-03

[16] Mikroe. MCP2517FD click. https://www.mikroe.com/mcp2517fd-click Accessed: 2020- 03-11

Prílohy

Zoznam príloh:

Príloha A: ZIP súbor so softvérom

Príloha B: 3D model krabičky