VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

# DIPLOMOVÁ PRÁCE

Brno, 2019 Bc. Matej Semančík


# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

## ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS


# ZABEZPEČENÍ BEZKLÍČOVÝCH SYSTÉMŮ U AUTOMOBILŮ
SECURITY OF KEYLESS SYSTEMS IN CARS

**DIPLOMOVÁ PRÁCE**
MASTER'S THESIS


**AUTOR PRÁCE** — **Bc. Matej Semančík**
AUTHOR

**VEDOUCÍ PRÁCE** — **Ing. Ondřej Krajsa, Ph.D.**
SUPERVISOR


**BRNO 2019**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

# Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

***Student:*** Bc. Matej Semančík ***ID:*** 164394
***Ročník:*** 2 ***Akademický rok:*** 2018/19

**NÁZEV TÉMATU:**

## Zabezpečení bezklíčových systémů u automobilů

**POKYNY PRO VYPRACOVÁNÍ:**

Navrhněte a realizujte systém zabezpečení automobilů využívajících bezklíčový, bezdrátový, způsob ověření. Proveďte analýzu stávajících systémů, jejich kritických míst a porovnejte s navrženým systémem. Pomocí softwarově definovaného rádia návrh realizujte.

**DOPORUČENÁ LITERATURA:**

[1] BOWNE, Samuel. Hands-On Cryptography with Python: Leverage the Power of Python to Encrypt and Decrypt Data. Birmingham: Packt Publishing, 2018. ISBN 9781789534443

[2] SWEIGART, Al. Cracking Codes with Python: An Introduction to Building and Breaking Ciphers. San Francisco: No Starch Press, Incorporated, 2018. ISBN 9781593278229

***Termín zadání:*** 1.2.2019 ***Termín odevzdání:*** 16.5.2019

***Vedoucí práce:*** Ing. Ondřej Krajsa, Ph.D.
***Konzultant:***

**prof. Ing. Jiří Mišurec, CSc.**
*předseda oborové rady*

## ABSTRAKT

Táto práca sa zaoberá bezdrôtovými bezkľúčovými systémami u automobilov, analýzou týchto systémov a najčastejších typov útokov na tieto systémy. Ďalej sa zaoberá návrhom a realizáciou bezkľúčového automobilového systému, ktorý by mal čo najlepšie odolávať spomínaným typom útokov a návrhom a realizáciou útoku na existujúci bezkľúčový automobilový systém.



## KĽÚČOVÉ SLOVÁ

automobil, AVR, bezkľúčové systémy, CC1101, mikrokontrolér, softvérovo definované rádio, zabezpečenie



## ABSTRACT

This thesis deals with car remote keyless systems, analysis of these systems and most common attacks on these systems. It also deals with design and realisation of secure car remote keyless system, which should be resistant to said attacks, and with design and realisation of attack against existing car remote keyless system.



## KEYWORDS

AVR, car, CC1101, keyless systems, microcontroller, security, software-defined radio

SEMANČÍK, Matej. *Zabezpečení bezklíčových systémů u automobilů*. Brno, 2018, 60 s. Diplomová práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: Ing. Ondřej Krajsa, Ph.D.

# VYHLÁSENIE

Vyhlasujem, že som svoju diplomovú prácu na tému „Zabezpečení bezklíčových systémů u automobilů“ vypracoval samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákoníka Českej republiky č. 40/2009 Sb.

Brno ..... .....

podpis autora

## POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomovej práce, pánovi Ing. Ondřejovi Krajsovi, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.


Brno ..... .....

podpis autora

# Obsah

**Úvod** **11**

**1 Teoretická časť práce** **12**
1.1 Bezdrôtové bezkľúčové systémy . . . . . 12
1.1.1 RKS z pohľadu užívateľskej interakcie . . . . . 12
1.2 Autentifikačné techniky . . . . . 14
1.2.1 Fixed Code – technika s fixným kódom . . . . . 14
1.2.2 Rolling Code – technika s postupným kódom . . . . . 14
1.2.3 Challenge-Response technika . . . . . 15
1.3 Typy útokov . . . . . 16
1.3.1 Útok spätným prehraním (Playback attack) . . . . . 16
1.3.2 Skenovací útok (Scan attack) . . . . . 16
1.3.3 Útok zosilnením signálu (Two-Thief attack) . . . . . 16
1.3.4 Útok predikciou náhodnej výzvy (Challenge-forward prediction attack) . . . . . 17
1.3.5 Slovníkový útok (Dictionary attack) . . . . . 17
1.3.6 RollJam . . . . . 18

**2 Testovanie a návrh riešenia** **19**
2.1 Najčastejšie implementácie SDR . . . . . 19
2.1.1 Texas Instruments CC1101 . . . . . 20
2.1.2 HackRF One . . . . . 21
2.1.3 RTL-SDR . . . . . 22
2.1.4 USRP1 . . . . . 23
2.1.5 GNU Radio . . . . . 23
2.1.6 Gqrx . . . . . 24
2.1.7 Inspectum . . . . . 25
2.1.8 UHD . . . . . 25
2.2 Voľba sady nástrojov . . . . . 25
2.3 Analýza signálu kľúčenky . . . . . 25
2.4 Návrh útoku . . . . . 27
2.5 Návrh bezkľúčového systému . . . . . 29

**3 Praktická časť - bezkľúčový systém** **31**
3.1 Schéma zapojenia . . . . . 31
3.2 PlatformIO . . . . . 32
3.2.1 Nastavenie projektu . . . . . 33

3.3 CC1101 . . . 35
3.3.1 Konfigurácia čipu . . . 36
3.3.2 Prehľad konfiguračných registrov . . . 38
3.4 Firmvér zariadení . . . 38
3.4.1 Konfiguračný hlavičkový súbor . . . 39
3.4.2 Komunikačný protokol . . . 41
3.4.3 Rozdiely medzi aktívnym a pasívnym režimom . . . 44
3.4.4 Generovanie náhodných výziev . . . 44
3.4.5 Šifrovanie náhodných výziev . . . 47
3.5 Odolnosť voči útokom . . . 48
3.5.1 Útok spätným prehraním (Playback attack) . . . 48
3.5.2 Skenovací útok (Scan attack) . . . 48
3.5.3 Útok zosilnením signálu . . . 49
3.5.4 Útok predikciou náhodnej výzvy . . . 49
3.5.5 Slovníkový útok . . . 49
3.5.6 RollJam . . . 49

**4 Praktická časť - implementácia útoku** **50**
4.1 Analýza signálu . . . 50
4.2 Nastavenie CC1101 . . . 53
4.3 Program . . . 53

**5 Záver** **55**

**Literatúra** **56**

**Zoznam symbolov, veličín a skratiek** **58**

**Zoznam príloh** **59**

**A Obsah priloženého CD** **60**

# Zoznam obrázkov


2.1 CC1101 moduly . . . . . 21
2.2 HackRF One, zdroj: https://greatscottgadgets.com/hackrf/ . . 22
2.3 RTL-SDR usb adaptér, zdroj: https://www.rtl-sdr.com/ . . . . . 22
2.4 USRP1, zdroj: https://www.ettus.com/ . . . . . 23
2.5 Grafické prostredie Gqrx . . . . . 24
2.6 Vodopádový graf signálu kľúčenky pri zamykaní a odomykaní . . . . 27
2.7 Časový priebeh signálu kľúčenky pri odomykaní . . . . . 28
3.1 Schéma zapojenia zariadení . . . . . 32
3.2 Export hodnôt registrov z programu SmartRF™ Studio . . . . . 37
3.3 CC1101 paket . . . . . 37
3.4 Štruktúra paketu komunikačného protokolu . . . . . 42
3.5 Priebeh výmeny správ medzi kľúčenkou a automobilom . . . . . 43
3.6 Blokovanie komunikácie kľúčenky s potencionálnym útočníkom . . . . 44
3.7 Blokovanie komunikácie automobilu s potencionálnym útočníkom . . 44
4.1 Kód kľúčenky zobrazený v programe Inspectrum . . . . . 51
4.2 Výpočet modulačnej rýchlosti v programe Inspectrum . . . . . 52
4.3 Odvodený amplitúdový graf signálu v programe Inspectrum . . . . . 52

# Zoznam tabuliek

3.1 Konfigurácia čipu CC1101 . . . . . 36
3.2 Prehľad konfiguračných registrov CC1101 . . . . . 38

# Zoznam výpisov


3.1 Konfiguračný súbor `platformio.ini` . . . . . 33
3.2 Hlavičkový súbor `constants.h` . . . . . 39
3.3 Výpis typov paketu . . . . . 42
3.4 Definícia štruktúry paketu . . . . . 42
3.5 Implementácia generátora náhodných čísel . . . . . 45
3.6 Generovanie náhodnej výzvy . . . . . 47
3.7 Príklad použitia knižnice `tiny-AES-c` . . . . . 48
4.1 Ukážka extrahovaných symbolov z programu Inspectrum . . . . . 51
4.2 Python skript extrahujúci bitovú sekvenciu ASK signálu . . . . . 51
4.3 Kódy kľúčenky zachytené pomocou CC1101 . . . . . 53

# Úvod

V dnešnej modernej spoločnosti sa komfort začína stávať už viacmenej štandardom. To platí aj v oblasti automobilov – len ťažko by sme si dnes vedeli predstaviť nové auto bez diaľkového uzamykania.

Automobilová bezpečnosť si prešla dlhú cestu vývojom, od klasického prístupu na fyzický kľúč, cez odomknutie pomocou PIN kódu na klávesnici umiestnenej priamo na aute [1], až po bezdrôtové systémy, ktoré sa používajú práve dnes. Takéto systémy sa nazývajú bezdrôtové bezkľúčové systémy - RKS (Remote Keyless System). Tieto systémy so sebou ale z pochopiteľných dôvodov prinášajú radu rizík – komunikujú totiž bezdrôtovo. Takáto komunikácia sa dá ľahko odchytiť útočníkom, ktorý vie zneužiť chyby v komunikačných protokoloch a odcudziť vozidlo.

Táto práca sa najprv v teoretickej časti venuje práve RKS, analýze bezpečnostných rizík týchto systémov, analýze najčastejších typov útokov na tieto systémy a návrhu bezpečného RKS schopného odolávať týmto útokom postaveného na AVR architektúre s bezdrôtovými modulmi Texas Instruments CC1101 za použitia AES šifrovania. Taktiež v teoretickej časti navrhneme útok na existujúci RKS ktorým poukážeme na slabiny existujúcich systémov. V praktickej časti navrhnutý RKS a útok implementujeme vo forme hardvérového prototypu.

# 1 Teoretická časť práce

V tejto časti sa bližšie pozrieme na špecifikáciu RKS u automobilov, čo tieto systémy obsahujú a na akých princípoch fungujú. Opíšeme si typy útokov a neskôr sa pokúsime navrhnúť taký RKS, ktorý by mal odolávať čo najlepšie všetkým typov útokov.

## 1.1 Bezdrôtové bezkľúčové systémy

Bezkľúčové systémy sú forma elektronického zámku, ktorý slúži na prístup do budov, otváranie brán, alebo prístup do automobilov. Bezkľúčové systémy u automobilov neslúžia nutne len na odomykanie dvier, ale používajú sa aj na ovládanie rôznych funkcií vozidla, ako napr. otvorenie kufra, alebo aktivácia/deaktivácia alarmu. U automobilov sa tieto systémy skladajú z palubného počítača umiestneného vo vozidle a kľúčenky, ktorú nosí užívateľ (vlastník automobilu) pri sebe. Táto kľúčenka sa často označuje aj ako CID (Customer identification device), pretože má vo svojej pamäti uložený unikátny kód, ktorý identifikuje pár kľúčenka-automobil (nie nutne sú CID a kľúčenka jedno rovnaké zariadenie, vo väčšine prípadov to ale tak je). O tomto kóde vie taktiež palubný počítač automobilu – týmto spôsobom vie automobil určiť, či sa s ním snaží komunikovať jemu priradená kľúčenka. Obe tieto zariadenia obsahujú rádiový vysielač pomocu ktorého medzi sebou komunikujú. Zväčša sa jedná o komunikáciu v nelicencovanom UHF frekvenčnom pásme (Ultra high frequency – Ultra krátke vlny). V európe sa používa pre účely tejto komunikácie pásmo nazývané LPD433 – jedná sa o nelicencované pásmo vyhradené pre tzv. low-power zariadenia (zariadenia s nízkou energetickou náročnosťou) so šírkou pásma 433.050 MHz až 434.790 MHz.

### 1.1.1 RKS z pohľadu užívateľskej interakcie

Z pohľadu užívateľskej interakcie existujú dva typy bezkľúčových bezdrôtových systémov – aktívne a pasívne.

#### Aktívne systémy

U prvého typu je vyžadovaná akcia od užívateľa, tj. interakcia s kľúčenkou. Ak sa užívateľ rozhodne odomknúť, alebo uzamknúť automobil, stlačí tlačidlo na kľúčenke, ktorá vyšle automobilu autorizačný kód a ten vykoná požadovanú akciu. U aktívnych systémov ešte delíme komunikáciu na jednosmernú a obojsmernú. U jednosmernej vyšle kľúčenka kód a nečaká na odpoveď od automobilu. Tento spôsob komunikácie je všeobecne náchylnejší na útoky, pretože medzi kľúčenkou a automobilom neprebieha

žiadna forma potvrdenia. U obojsmerných systémov prebieha medzi automobilom a kľúčenkou obojsmerná výmena správ a je tu už možnosť implementácie zabezpečeného protokolu alebo potvrdenia kľúčenke o vykonaní, či nevykonaní požadovanej akcie automobilom.

Riziká tohto typu systému spočívajú v tom, že útočník vie ľahko odchytiť autorizačný kód prenášaný medzi kľúčenkou a automobilom (za predpokladu že je použitý fixný autorizačný kód – vysvetlené v ďalších častiach práce) a neskôr ho použiť na prístup do vozidla.

### Pasívne systémy

U pasívnych systémov komunikuje automobil a CID bez potreby užívateľskej interakcie s CID (u pasívnych systémov sa väčšinou používa terminológia CID, vo väčšine prípadov CID = kľúčenka, CID ale môže mať aj inú formu, ako napr. imobilizér). Cieľom je odomknúť automobil bez toho aby musel užívateľ použiť kľúčenku – to znamená, že automobil alebo CID musia sami zahájiť vzájomnú komunikáciu keď sa užívateľ priblíži ak automobilu. Vieme o dvoch takýchto spôsobch.

Prvý spočíva v tom, že CID obsahuje akcelerometer, alebo iný pohybový senzor, vďaka ktorému sa aktivuje prenos autorizačného kódu – reálne použitie je také, že užívateľ nosí CID pri sebe a pri jeho pohybe CID vysiela autorizačný kód, ktorý odomnke vozidlo, ak sa jeho užívateľ nachádza v blízkosti (tzn. signál je dostatočne silný na to, aby ho zachytilo vozidlo). Ak sa užívateľ vzdiali od vozidla a tým pádom vozidlo prestane prijímať autorizačný kód z CID, automaticky sa uzamkne. Tento návrh systému má ale niekoľko nevýhod:

- Útočník má uľahčené odchytenie komunikácie medzi CID a vozidlom, pretože CID nepretržite vysiela signál pri pohybe – tzn. že útočník nemusí čakať na užívateľa, kým stlačí tlačidlo na kľúčenke
- Keďže CID vysiela nepretržite kým je užívateľ v pohybe, je jeho energetická náročnosť oveľa vyššia ako pri kľúčenkách u aktívnych systémov

Druhý spôsob spočíva v automatickom odomknutí až keď sa užívateľ pokúsi otvoriť dvere automobilu kľučkou. Hneď po zatiahnutí za kľučku automobil začne vysielať nízkofrekvenčný signál, ktorý zobudí CID z režimu s nízkou spotrebou energie, po zobudení tento signál CID dekóduje a ak zistí, že obsahuje validný kód, tak vyšle svoj autorizačný kód, ktorý odomkne vozidlo. Takýto systém vyžaduje obzvlášť rýchly komunikačný protokol, pretože vozidlo sa musí odomknúť ešte predtým ako kľučka dosiahne pri otváraní svoju finálnu pozíciu, ináč hrozí že nastane mechanické zablokovanie [2] a otvorenie dverí sa nepodarí na prvýkrát. Tento návrh pasívneho

systému je obzvlášť nachylný na populárny útok zosilnením signálu (Two-Thief attack), kedy dvaja útočníci (prvý pri vozidle a druhý blízko užívateľa) zosilnia signál medzi vozidlom a užívateľovou kľúčenkou ktorá sa nachádza mimo dosahu vozidla, za účelom získania prístupu do vozidla.

## 1.2 Autentifikačné techniky

U RKS sa používa viacero autentifikačných techník na validáciu správ (autorizačných kódov) medzi dvoma zariadeniami – kľúčenkou/CID a automobilom. Tie najznámejšie sú technika s fixným kódom (Fixed Code Technique), technika s postupným kódom (Rolling Code Technique) a tzv. Challenge-Response (výzva-odpoveď) technika. Nižšie si jednotlivo popíšeme tieto techniky.

### 1.2.1 Fixed Code – technika s fixným kódom

Táto technika je zo všetkých naprimitívnejšia. Používa sa u niektorých garážových brán a v dnešnej dobe sa ich bezpečnosť dá prelomiť do pár sekúnd, resp. minút, podľa dlžky kódu.

Funguje tak, že zariadenia majú vo svojej pamäti uložený rovnaký autorizačný kód ktorý im bol pridelený počas výroby, prípadne si tento kód dokáže užívateľ nakonfigurovať sám. V momente keď užívateľ vyvolá akciu na kľúčenke, kľúčenka tento kód vyšle druhému zariadeniu a ak sa druhé zariadenie (automobil, brána) nachádza v blízkosti, príjime tento kód a overí si či je validný. Kód je validný vtedy ak sú autorizačné kódy zariadení rovnaké. Ak je kód validný, druhé zariadenie vykoná požadovanú operáciu.

Keďže v tejto technike sa zariadenia autorizujú stále rovnakým kódom, je táto technika obzvlášť náchylná na útoky, pretože v dobe odosielania kódu ho môže útočník ľahko odchytiť a neskôr použíť na neautorizovaný prístup, prípadne môže útočník použiť skenovací, resp. bruteforce útok kde skúša posielať všetky možné kódy, až kým neuhádne ten správny.

### 1.2.2 Rolling Code – technika s postupným kódom

Pri tejto technike každá prenesená správa obsahuje rozdielny autorizačný kód. Používa sa často u prístupových systémov a garážových brán [3] a je celkom rozšírená u automobilových RKS.

Autorizačný kód je generovaný odosielajúcim zariadením, pričom sa na generovanie používa pseudo-náhodný generátor. Každé zariadenie obsahuje aj sekvenčný

čítač ktorého hodnota (v kombinácii s unikátnym kódom uloženým v oboch zariadeniach) slúži ako tzv. seed (semeno) pre tento generátor. Pri odosielaní prvé zariadenie (kľúčenka) vygeneruje pomocu generátora náhodný autorizačný kód a inkrementuje hodnotu svojho čítača. Potom tento kód odošle druhému zariadeniu (vozidlu). Po prijatí na druhej strane zariadenie tiež inkrementuje hodnotu svojho čítača a vygeneruje kód, ktorý porovná s prijatým. Ak sa kódy zhodujú, tak sú validné a druhé zariadenie (vozidlo) môže vykonať požadovanú akciu. Môže nastať aj situácia, že hodnoty čítačov u zariadení nie sú synchronizované (napr. aktivácia kľúčenky mimo dosah vozidla) – to znamená že kód by nebol nikdy validný, preto zariadenia počítajú s určitým rozmedzím hodnôt čítača (semena generátora), napr. 64 hodnôt – ak zariadenie prijíme autorizačný kód ktorý sa nezhoduje s jeho vygenerovaným kódom, skúša inkrementovať hodnotu čítača v dovolenom rozmedí, až kým sa mu nepodarí vygenerovať rovnaký kód a takto ho validovať. Takto si zariadenia synchronizujú sekvenčný čítač. Aby sme pridali na bezpečnosti tejto techniky, zvykne sa ešte prenášaný autorizačný kód šifrovať kľúčom, ktorý je u oboch zariadení rovnaký.

Táto autentifikačná technika eleminuje možnosť útoku opakovaním (tj. keď útočník skúša posielať kód, ktorý už v minulosti odchytil) Tiež stojí za zmienku, že v tejto technike sa zvyknú používať kryptograficky bezpečné pseudo-náhodné generátory (CSPRNG – Cryptographically Secure Pseudo-Random Number Generator) ktoré generujú bezpečné a dlhé autorizačné kódy, ktoré sa nedajú dopredu predpovedať.

### 1.2.3 Challenge-Response technika

Táto technika sa často používa u imobilizérov [4] a využíva obojsmernú komunikáciu medzi zariadeniami. Používa sa často u pasívnych systémov. Obe zariadenia (automobil a CID) majú vo svojej pamäti uložený rovnaký šifrovací kľúč. Ak užívateľ vyvolá proces autentifikácie (napr. zatiahne za kľučku), prvé zariadenie (vozidlo) vygeneruje náhodné číslo – tzv. random challenge (náhodná výzva) a odošle ho druhému zariadeniu. Druhé zariadenie (CID) toto číslo prijíme a následne ho zašifruje svojim kľúčom. Tým vznikne odpoveď (response), ktorú odošle späť prvému zariadeniu. Po prijatí (alebo počas čakania na odpoveď) prvé zariadenie vypočíta očakávanú odpoveď z odoslanej výzvy a svojho šifrovacieho kľúča. Ak sú po prijatí a porovnaní obe odpovede rovnaké, tak prvé zariadenie považuje druhé za verifikované a tým pádom môže vykonať požadovanú akciu. Ak je táto technika použitá u pasívnych RKS, je obzvlášť náchylná na útok zosilnením signálu ktorý si popíšeme v nasledujúcej sekcii.

## 1.3 Typy útokov

Teraz si popíšeme rôzne typy útokov, ktoré sú najčastejšie používané na bezkľúčové systémy. Každý z útokov sa špecializuje na iný typ systému a využíva rôzne vady týchto systémov za účelom získania prístupu, resp. odcudzenia vozidla.

### 1.3.1 Útok spätným prehraním (Playback attack)

Toto je asi najprimitívnejší útok na RKS. U tohto útoku sa útočník dostane do blízkosti auta v momente kedy ho užívateľ odomyká. Odchytí kód prenášaný medzi kľúčenkou a vozidlom, tento kód si uloží a neskôr sa ním pokúsi odomknúť vozidlo. Tento typ útoku je efektívny voči systémom ktoré používajú autentifikačnú techniku s fixným kódom, pretože kód sa nikdy nemení.

### 1.3.2 Skenovací útok (Scan attack)

U tohto útoku útočník len háda autorizačný kód vozidla, tzv. „skenovaním“ (preto scan attack), v podstate sa jedná o bruteforce útok – útočník skúša generovať náhodné autorizačné kódy až kým neuhádne ten správny a automobil sa odomkne. Tento útok je z časového hľadiska náročnejší ako útok spätným prehraním (uhádnutie kódu niečo trvá) a hlavné je, že útočník nemusí čakať na užívateľa a signál z jeho kľúčenky. Takto útok funguje u aktívnych systémov.

U pasívnych systémov útočník svoj kód nemení, skúša len ťahať za kľučku vozidla, pričom vozidlo stále vyšle inú náhodnú výzvu. Útočník na každú výzvu odpovedá rovnakým kódom až kým nie je úspešný.

Keďže útočník nemusí vedieť podrobnosti o technických detailoch použitého komunikačného protokolu medzi vozidlom a kľúčenkou, je tento útok ľahký na realizáciu – útočníkovi stačí vedieť iba dĺžku autorizačného kódu ktorým má vykonávať útok. Časová náročnosť tohto útoku teda stúpa s dĺžkou použitého autorizačného kódu.

### 1.3.3 Útok zosilnením signálu (Two-Thief attack)

Tento útok vyžaduje participáciu dvoch útočníkov a je použiteľný iba u pasívnych systémov.

Obaja útočníci pri sebe nosia zariadenie na zosilnenie signálu medzi CID a automobilom – tzv. opakovač (angl. repeater), čím sa snažia zúžiť komunikačnú vzidalenosť medzi vozidlom a CID. Keď sa užívateľ vzdiali od vozidla na vzdialenosť, pri ktorej už vozidlo nedokáže komunikovať s CID, útočníci sa strategicky rozmiestnia – prvý k vozidlu a druhý na miesto v blízkosti CID. Jeden z možných príkladov je, keď

útočníci počkajú kým užívateľ zaparkuje vozidlo pred budovou, zamkne vozidlo a vstúpi do budovy, kde nemá na vozidlo priamy výhľad. V tom momente začne prvý útočník ťahať za kľučku vozidla, ktoré začne vysielať signál obsahujúci náhodnú výzvu určenú pre CID. Prvý útočník tento signál prijíme a zosilní ho natoľko, aby ho dokázal prijať CID. Druhý útočník ktorý je umiestnený v blízkosti CID čaká na jeho odpoveď, ktorú po prijatí tiež zosilní a pošle späť vozidlu, ktoré po jej prijatí odomkne dvere.

Možnosť realizácie tohto útoku poukazuje na vážnu vadu pasívnych systémov, keďže útočník nepotrebuje vôbec poznať komunikačný protokol medzi zariadeniami, stačia mu len správne nástroje na zosilnenie signálu.

Jeden z možných spôsobov ochrany voči tomuto útoku je meranie času od začiatku prenosu náhodnej výzvy až do prijatia odpovede od CID [5]. Ak je čas dlhší ako daný limit (tzn. že signál bol pravdepodobne prenášaný na väčšiu vzdialenosť), tak vozidlo jednoducho zamietne autorizáciu aj v prípade že prijíme validnú odpoveď.

### 1.3.4 Útok predikciou náhodnej výzvy (Challenge-forward prediction attack)

Tento útok sa špecializuje na analýzu generátora náhodnej výzvy, ktorú vysiela automobil po zatiahnutí za kľučku dverí u pasívnych systémov.

Útočník najprv zozbiera niekoľko náhodných výziev od automobilu, z ktorých sa snaží prísť na to, aká výzva bude vygenerovaná ako ďalšia. Potom sa presunie do blízkosti CID, ktorému vyšle výzvu u ktorej predpokladá, že bude vygenerovaná automobilom ako ďalšia. Odpoveď na túto výzvu si uloží a následne sa potom presunie k automobilu, kde skúsi znova zatiahnúť za kľučku. Za predpokladu že automobil vygeneruje takú výzvu, ktorú útočník predpokladal, môže útočník odpovedať vozidlu odpoveďou ktorú zaznamenal z CID a tak získať prístup do vozidla.

### 1.3.5 Slovníkový útok (Dictionary attack)

Jedná sa tiež o útok na pasivné systémy. U tohto útoku si útočník vytvára slovník, do ktorého si ukladá validné páry výzva-odpoveď. Docieli toho tak, že v blízkosti užívateľa s CID vygeneruje a odošle mnoho náhodných výziev, na ktoré CID odpovedá. Tieto páry výzva-odpoveď si ukladá do slovníka. Neskôr keď si vytvorí dostatočne veľký slovník, sa presunie k automobilu, kde skúša ťahať za kľučku a dúfa, že automobil vytvorí jednu z ním vygenerovaných výziev, na ktorú vie odpovedať korešpondujúcou odpoveďou zo slovníka.

### 1.3.6 RollJam

Na hackerskej konferencii DEF CON v roku 2015 predviedol bezpečnostný výskumník Samy Kamkar svoje zariadenie nazvané RollJam [6] ktoré cieli na systémy s autentifikačnou technikou postupného kódu.

Princíp spočíva v tom, že útočník umiestní toto zariadenie na automobil, resp. do blízkosti automobilu a čaká na užívateľovu akciu (odomknutie kľúčenkou). V momente kedy kľúčenka vyšle signál s kódom, RollJam tento kód zaznamená a zároveň zaruší prostredie silnejším signálom, čo znemožní automobilu prijať tento kód. Užívateľ to po prvom neúspešnom pokuse samozrejme skúsi znova – tentokrát ale útočník takto zaznamená aj druhý kód ktorý si uloží a vozidlu prepošle ten prvý, čo má za následok, že užívateľ sa už síce dostane do vozidla, ale útočník má k dispozícii ešte druhý kód, ktorý vie použiť neskôr. Týmto útokom Kamkar poukázal na zraniteľnosť vtedajších automobilových RKS, resp. použitých bezpečnostných čipov ktoré sa vo veľkom používali na zabezpečenie vozidiel (bol napr. zasiahnutý aj známy generátor KeeLoq), ktorá spočíva v tom že generovaným autorizačným kódom nikdy nevypršala platnosť. Tvrdí, že o zraniteľnosti sa vedelo, len tento útok dovtedy ešte nikto nedemonštroval a výrobcovia tomu taktiež neprikladali veľkú váhu [6]. Dnes je otázne koľko výrobcov už túto vadu odstránilo.

# 2 Testovanie a návrh riešenia

V tejto kapitole sa bližšie pozrieme na návrh riešenia tejto práce a zoznámime sa s nástrojmi, ktoré použijeme pri testovaní. Najprv budeme analyzovať signál prenášaný medzi kľúčenkou a automobilom u už existujúceho RKS, potom navrhneme možnú implementáciu útoku na takýto systém a neskôr navrhneme nový bezkľúčový systém ktorý bude lepšie odolávať útokom.

V prvom rade musíme vyriešiť, akým spôsobom budeme analyzovať rádiový signál. Na tento účel nám poslúži softvérovo definované rádio (SDR), resp. sada nástrov ktorá implementuje SDR (popíšeme si ju neskôr). Softvérovo definované rádio je v podstate softvérová implementácia hardvéru ktorý slúži na rádiovú komunikáciu, tzn. sa jedná o rôzne frekvenčné filtre, modulátory, demodulátory, atp. implementované na hardvéri v spojení s počítačom, resp. mikrokontrolérom na ktorom beží program spracovávajúci dáta. Toto riešenie má hlavnú výhodu v tom, že môže bežať na klasickom procesore v osobnom počítači, mikrokontroléri, alebo na FPGA, čo znamená že sa dá ľahko programovať, čím vieme vytvoriť vlastné rádiové zariadenia. To z SDR robí ideálny nástroj na vývoj rádiových zariadení, ktoré budú neskôr implementované priamo na samostatnom čipe.

V nasledujúcej sekcii sa pozrieme na rôzne možnosti použitia SDR.

## 2.1 Najčastejšie implementácie SDR

Softvérovo definované rádio je systém, ktorý sa typicky skladá z hardvéru slúžiaceho na príjem/vysielanie rádiového signálu a procesora (počítača) na ktorom beží program spracovávajúci tento signál. Pre naše potreby si preto budeme musieť vybrať takú sadu nástrojov, ktorá spĺňa podmienky pre realizáciu tejto práce. Nie nutne musíme použiť tú istú sadu nástrojov na analýzu signálu a vytvorenie bezkľúčového systému. Každá z ponúkaných má svoje výhody a nevyhody.

Základná podmienka pre vytvorenie bezkľúčového systému je možnosť vysielať a prijímať rádiový signál z dvoch nezávislých zariadení a teda, mať dva nezávislé RF moduly.

Máme viacero možností. Prvou je, že si môžeme poskladať vlastné SDR z mikrokontroléra a prídavneho RF modulu. Mikrokontrolér by v tomto prípade slúžil na spracovanie a generovanie signálu za pomoci podporných analógových obvodov, pričom RF modul na jeho vysielanie a prijímanie. Za predpokladu že bude mikrokontrolér využívaný aj na samotné spracovanie RF signálu, tkvie nevýhoda tohto riešenia v tom, že mikrokontrolér musí obsahovať výkonný A/D a D/A prevodník na prevod RF signálu do, resp. z digitálnej pododoby a v neposlednom rade by sme

museli navrhnúť aj podporné obvody ktoré by nám umožnili tento signál modulovať a vysielať na určitej frekvencii.

Existuje ale riešenie, ktoré nám umožní použiť bežne dostupný mikrokontrolér založený na AVR architektúre bez podporných rádiových obvodov – existujú totiž RF moduly ktoré obsahujú všetok potrebný hardvér (modulátory a demodulatóry, frekvenčné filtre, prevodníky, atp.) na jednom čipe. Príkladom je napríklad rádiový čip CC1101 od firmy Texas Instruments, ktorý obsahuje hardvér schopný spracovávať a generovať rádiový signál. Tento čip komunikuje s mikrokontrolérom pomocou SPI zbernice a ponúka množstvo funkcií už na hardvérovej úrovni, ktorými vie odprostiť mikrokontrolér od potreby spracovávať samotný signál. Nie je tak flexibilný ako drahšie SDR kity ponukájúce výkonný hardvér a možnosť komunikovať na širokom rozsahu frekvencií, no je to ideálne riešenie pre implementáciu menšieho rádiového zariadenia s iba jednou funkciou (napríklad RKS).

K dispozícii máme dva RF moduly CC1101 a dva mikrokontroléry ATmega328 osadené na doske Arduino Nano.

Ďalšia možnosť je použitie už hotového SDR kitu, ktorý obsahuje antény a výkonné A/D a D/A prevodníky. Dnes existuje viacero takýchto zariadení. V komunite rádioamatérov sú obzvlášť populárne zariadenia HackRF One a RTL-SDR, ktoré sú schopné komunikovať na širokom rozsahu frekvencií. My máme k dispozícii zariadenie USRP1 ktoré je podobné HackRF One, no výkonnejšie, avšak ťažšie ovládateľné a bez extenzívnej podpory komunity. Vieme ho však pomocou základných ovládačov použiť na odchytenie signálu za účelom neskoršej analýzy.

Pre všetky tieto SDR kity existuje aj open-source softvérová sada nástrojov nazývaná GNU Radio, ktorá umožňuje komunikáciu s SDR pomocou USB zbernice, vizualizáciu a spracovanie signálu a mnoho ďalšieho.

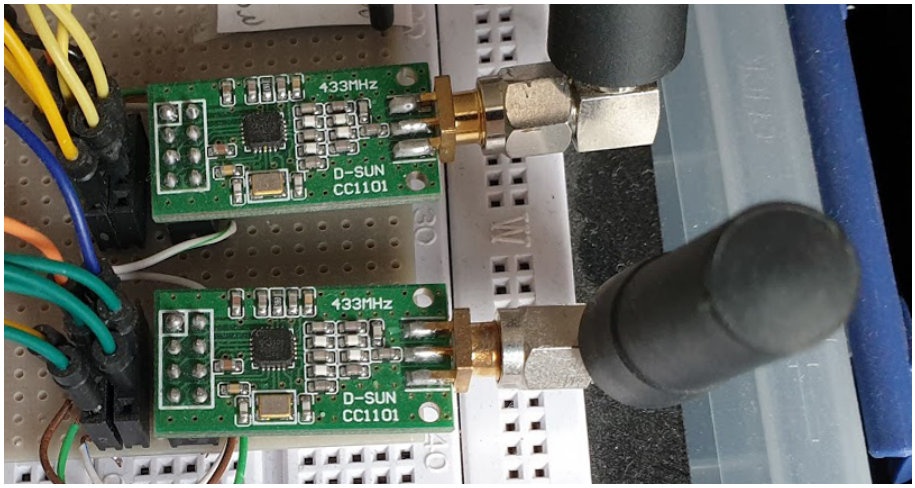
V ďalších častiach si bližšie popíšeme spomínané nástroje, ktorými sa dá spracovávať RF signál a implementovať bezkľúčový systém a zvolíme si architektúru, na ktorej budeme implementovať túto záverečnú prácu.

### 2.1.1 Texas Instruments CC1101

CC1101 je cenovo dostupný RF vysielač a prijímač operujúci na nelicencovaných frekvenčných pásmach pod 1 GHz. Bol navrhnutý za účelom použitia v priemyselnej, vedeckej a lekárskej sfére [7] a mimo iných, je schopný operácie aj v nelicencovanom 433 MHz pásme. Medzi kľúčové vlastnosti tohto čipu patrí podpora rôznych typov modulácií (napr. 2-FSK, 4-FSK alebo ASK, resp. OOK), konfigurovateľná prenosová rýchlosť, voľba prenosového kanálu, konfigurácia odstupu jednotlivých kanálov, filtrácia paketov na hardvérovej úrovni, automatická detekcia preambuly

(konfigurovateľnej dĺžky) a synchronizačného slova (samozrejme tiež konfigurovateľného), výpočet CRC, indikácia RSSI a mnoho ďalšieho.

Samotný čip sa dá pre prototypovacie účely kúpiť na doske plošného spoja obsahujúcej podporné pasívne súčiastky potrebné pre správnu funkcionalitu, hlavičku na prepojenie s mikrokontrolérom a konektor na anténu. Ako sme už spomínali, máme k dispozícii dva takéto moduly, ktoré môžeme vidieť na obrázku 2.1




Obr. 2.1: CC1101 moduly

CC1101 komunikuje s mikrokontrolérom pomocou SPI zbernice, ktorá slúži na konfiguráciu samotného čipu a zároveň na výmenu dát medzi mikrokontrolérom. CC1101 sa pri SPI komunikácii chová ako tzv. slave zariadenie, takže pripojený mikrokontrolér je v roli tzv. master zariadenia. Konfigurácia čipu je realizovaná zápisom do jeho 47 konfiguračných registrov, ktorých hodnoty ovplyvňujú vlastnosti a spôsob chovania čipu.

### 2.1.2 HackRF One

HackRF One je samostatné zariadenie schopné komunikácie v rozsahu 1 MHz – 6 GHz v half duplex móde. Pripája sa k počítaču pomocou USB zbernice a má open-source design hardvéru. Obsahuje mimo iného rôzne konektory na pripojenie antén, softvérovo konfigurovateľné filtre, prevodníky a vlastný procesor umožňujúci samostatnú operáciu [8]. Je to lacnejšia, no stále dosť výkonná alternatíva k USRP1, ktorá je dobrou voľbou pre začatiatok vývoja na SDR. Zariadenie môžeme vidieť na obrázku 2.2.


Obr. 2.2: HackRF One, zdroj: `https://greatscottgadgets.com/hackrf/`

### 2.1.3 RTL-SDR

RTL-SDR je komunitný projekt, ktorý využíva ako základ DVB-T tuner s RTL2832U čipsetom [9]. Tento čipset sa stal populárnym práve preto, že komunita prišla na spôsob ako vyrobiť s pomerne lacného DVB-T tunera softvérovo definované rádio pomocou vlastného USB ovládača. Je veľmi rozšírený pre svoju nízku cenu (okolo $25) a v súčasnej dobe má väčšinou podobu malého USB adaptéra s konektorom na pripojenie antény. Vzhľad záleží ale na výrobcovi, zariadenie môže mať totiž mnoho podôb, základ je iba RTL2832U čipset ktový tvorí jadro zariadenia.


Obr. 2.3: RTL-SDR usb adaptér, zdroj: `https://www.rtl-sdr.com/`

RTL-SDR má aktívnu komunitu ktorá, preň vytvorila množstvo skriptov a programov na jeho ovládanie a príjem dát. Dôležitá vlastnosť je, že toto SDR, keďže je založené na TV tuneri, dokáže iba prijímať signál. Jeho frekvenčný rozsah sa pohybuje v rozmedzí od 500 kHz do 1.75 GHz. To z neho robí ideálny nástroj len na odposluch a analýzu dát v tomto pásme.

### 2.1.4 USRP1

USRP1 je zariadenie z rodiny Universal Software Radio Peripheral$^{TM}$ [10]. Tieto zariadenia vyrába spoločnosť Ettus Research ktorá spadá pod National Instruments. Používajú sa hlavne vo výskumnej sfére a nie sú až tak obľúbené aj u rádioamatérov. Väčšina USRP zariadení komunikuje s hosťovským počítačom pomocou USB zbernice alebo po sieti, niektoré zo zariadení obsahujú výkonné FPGA ktoré dokáže nahradiť hosťovský počítač v spracovaní signálu, čo im umožňuje samostatnú operáciu. USRP zariadenia sa štandardne používajú so sadou nástrojov GNU Radio.


Obr. 2.4: USRP1, zdroj: `https://www.ettus.com/`

USRP1 obsahuje výkonné Altera Cyclone FPGA, A/D prevodník s rýchlosťou 64 MS/s, D/A prevodník s rýchlosťou 128 MS/s a má možnosť inštalovania prídavných RF modulov, tzv. „daughterboards“. Naše USRP1 obsahuje modul WBX schopný komunikovať v rozmedzí 50 MHz až 2,2 GHz, má konektory na dve antény a umožňuje plne duplexnú komunikáciu.

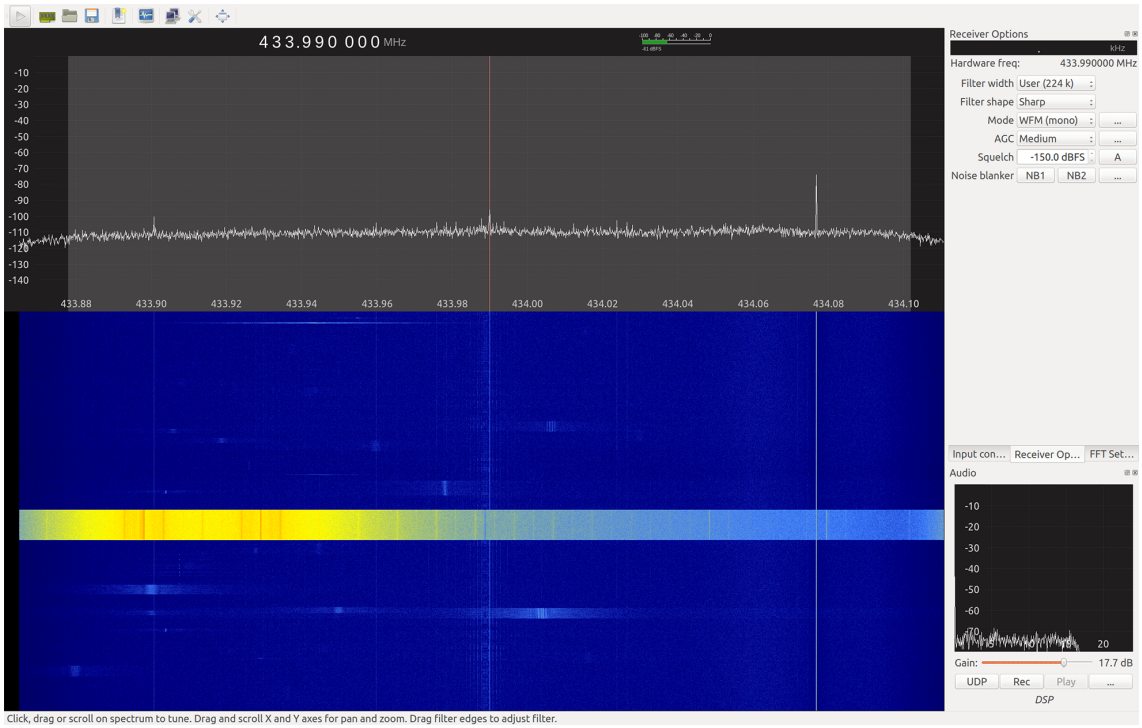
### 2.1.5 GNU Radio

GNU Radio je sada nástrovoj pre SDR umožňujúca spracovanie a analýzu signálu. Je to open-source softvér s aktívnou komunitou užívateľov, čo z neho robí silný nástroj, ktorý je veľmi modulárny. Poskytuje programy spustiteľné z príkazového riadku, ale aj grafickú nadstavbu, ktorá sa ovláda spájaním funkčných blokov, ktoré ako celok vedia vytvoriť komplexné rádiove aplikácie. Jednotlivé bloky slúžia stále iba na jednen účel (napr. blok slúžiaci na moduláciu, filtráciu signálu a podobne), pričom

GNU Radio umožňuje aj vytváranie nových blokov s vlastnou logikou v programovacom jazyku Python. Ak sme náročnejší a musíme klásť dôraz dôraz na efektivitu a rýchlosť spracovania signálu, umožňuje GNU Radio vytváranie blokov aj v jazyku C++ ktorý je viac nízkoúrovňový.

### 2.1.6 Gqrx

Gqrx je grafický SDR prijímač, ktorým budeme vedieť vizualizovať prijímaný rádiový signál. Je založený na sade nástrojov GRU Radio a je tiež open-source. Budeme ho používať na analýzu rádiovej komunikácie, podporuje FFT zobrazenie frekvenčného spektra a tzv. „waterfall“ – vodopádový graf, ktorý zobrazuje priebeh rádiového signálu v čase na určitom frekvenčnom pásme, ukážku môžeme vidieť na obrázku 2.5, kde vidno FFT analýzu a vodopádový graf pri príjímaní signáu v nelicencovanom rádiovom pásme. Gqrx podporuje ukladanie rádiového signálu do audio súboru, ktorý môžeme neskôr analyzovať v jednom zo softvérov na editáciu audia (napr. Audacity). Gqrx nám hlavne poslúži ako frekvenčný skener, ak nebudeme dopredu vedieť na ktorých frekvenciách komunikujú zariadenia.




Obr. 2.5: Grafické prostredie Gqrx

### 2.1.7 Inspectum

Inspectrum je jednoduchý program, ktorý umožňuje detailné prezeranie I/Q vzoriek. Ponúka nám detailný pohľad na časový pribeh signálu vo frekvenčnom spektre a umožňuje aj jednoduchú analýzu tohto signálu. Ukážku je možno vidieť na obrázku 2.7.

### 2.1.8 UHD

UHD (USRP Hardware Driver) je hardvérový ovládač pre USRP zariadenia, ktorý umožňuje detekciu, získanie informácií a komunikáciu s týmito zariadeniami a jeho inštalácia je nutná na spojazdnenie USRP1. Jeho nainštalovaním sa nám sprístupnia rôzne utility slúžiace na príjem a vysielanie signálu, konfiguráciu USRP zariadení, demoduláciu signálu a mnoho ďalšieho. Jednou z týchto utilít je program `uhd_rx_cfile` ktorý naladí USRP anténu na určitú frekvenciu, nastaví vzorkovaciu frekvenciu a uloží získané vzorky do súboru. Tieto vzorky sa nazývajú I/Q dáta, reprezentujú zmenu v amplitúde a fázovom posune sínusiody a dajú sa analyzovať v množstve programov. I/Q dáta sa často používajú v SDR sfére. Na analýzu I/Q vzoriek môžeme použiť už spomínaný Gqrx ktorý ich dokáže prehrať v reálnom čase, alebo môžeme použiť tiež voľne dostupný program `inspectrum` ktorý vie tieto vzorky vykresliť ako priebeh signálu a tak ich vieme bližšie analyzovať.

## 2.2 Voľba sady nástrojov

Na analýzu rádiového signálu použijeme zariadenie USRP1 ktoré nám bolo poskytnuté školou v kombinácii s programom Gqrx a Inspectum. Jeho ovládanie pomocou UHD ovládačov je pomerne ľahké a priamočiare.

Čo sa týka samotnej implementácie bezkľúčového systému a útoku, volíme si implementáciu na AVR mikrokontroléri s Arduino frameworkom a RF modulom CC1101. Dôvodov je hneď niekoľko – autorovi práce je príjemnejšia práca s mikrokontrolérmi a programovanie, ako práca s GNU Radiom v grafickom rozhraní, ktoré má strmú učiacu krivku. Implementácia prototypu na mikrokontroléri je zároveň flexibilnejšia z hľadiska portovania na iné platformy, keďže program pre kľúčenku a palubný počítač automobilu budeme písať v jazyku C++.

## 2.3 Analýza signálu kľúčenky

V tejto sekcii si bližšie ukážeme postup, akým sme dokázali prijať a vizualizovať signál z kľúčenky.

Najprv potrebujeme nainštalovať UHD ovládač pre USRP, GNU Radio a všetky ostatné utility na vizualizáciu signálu. To sa dá docieliť dvoma spôsobmi – keďže je GNU Radio modulárne a v základe neobsahuje kompatibilitu s UHD zariadeniami, prvý spôsob by bol ho manuálne skompilovať zo zdrojových kódov a vybrať komponenty ktoré chceme do inštalacie pridať. To isté platí pre Gqrx ktoré tiež v základe nepodporuje UHD zariadenia. To je zdĺhavý a komplikovaný proces, preto môžeme zvoliť duhú variantu, ktorou je použitie Linuxovej distribúcie Ubuntu, obsahujúcej už predinštalované GNU Radio so všetkými podporovanými komponentami. GNU Radio sa totiž distribuuje aj ako Live DVD s Linux Ubuntu, kde nájdeme predinštalované GNU Radio a všetko potrebné na začiatok práce so softvérovo definovanými rádiami. Okrem iného táto distribúcia obsahuje aj Gqrx verziu podporujúcu UHD zariadenia.

Volíme teda druhú voľbu a bootujeme Ubuntu s GNU Radiom. Po prihlásení sa do systému pripojíme USRP1 a z terminálu pomocou programov `uhd_find_devices` a `uhd_usrp_probe` zistíme, či je naše USRP1 funkčné a aké má nainštalované hardvérove moduly.

Pri našom testovaní sme mali na krátku dobu k dispozícii kľúčenku od automobilu Opel Vivaro. Táto kľúčenka obsahuje dve tlačidlá – prvé na odomknutie a druhé na uzamknutie vozidla. Pomocou Gqrx najprv zistíme, na ktorej fekvencii vysiela kľúčenka.

Je dôležité poznamenať, že pri hľadaní tejto frekvencie, ju nesmieme aktivovať zbytočne veľa krát ak je automobil mimo dosah. Môže totiž dôjsť k prekročeniu limitu sekvenčného čítača (kľúčenka inkrementuje svoju hodnotu a automobil nie), čo môže mať za následok že nakoniec nebudeme schopní odomknúť vozidlo pretože hodnoty oboch zariadení nebudú v dovolenom limite.

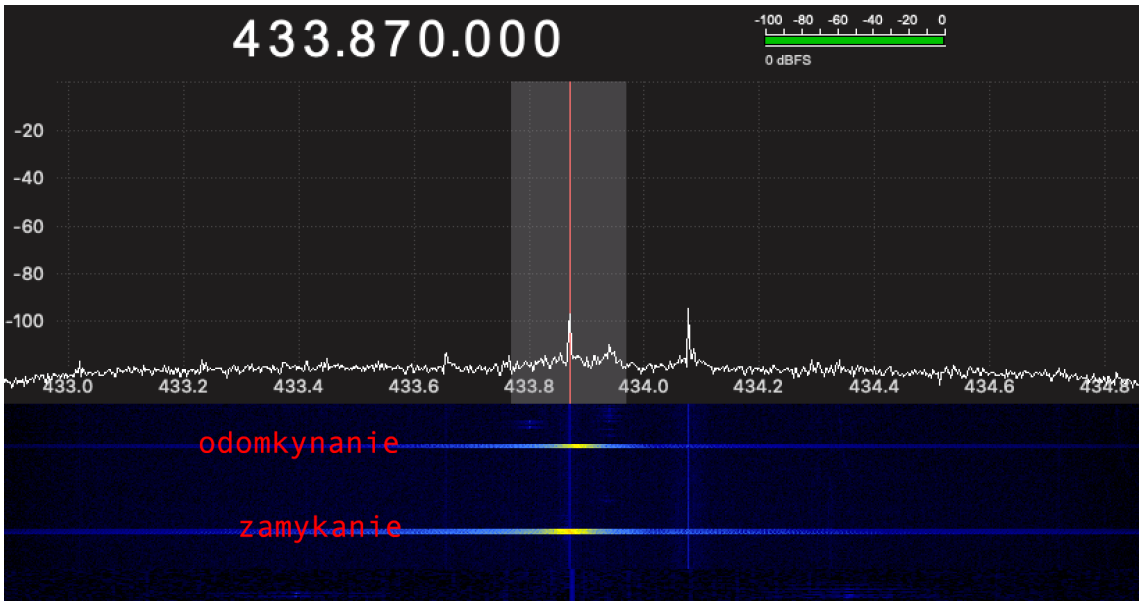
Zistili sme že kľúčenka vysiela signál v okolí 433,87 MHz. Zaujímavé zistenie je, že odomykacia správa bola vysielaná na trochu vyššej frekvencii, ako zamykacia, čo môžeme vidieť na snímke obrazovky z Gqrx na obrázku 2.6.

Oba tieto signály sme si uložili do súboru s I/Q vzorkami pomocou programu `uhd_rx_cfile` na neskoršie spracovanie a analýzu. Použili sme vzorkovaciu prekvenciu 2 MS/s. Príklad použitia programu je nasledovný.

```
uhd_rx_cfile -a type=usrp1 -f 433.87M -r 2000000 fob.raw
```

Prvý parameter špecifikuje použitý typ USRP, druhým parametrom volíme frekvenciu na ktorej má USRP vzorkovať (433,87 MHz), tretí paramater špecifikuje vzorkovaciu frekvenciu (2 MS/s) a posledný parameter špecifikuje výstupný súbor.

Poďme sa pozrieť na pribeh signáu. Na zobrazenie priebehu použijeme program `inspectrum` ktorý si ako vstup berie súbor s I/Q vzorkami. Na obrázku 2.7 môžeme vidieť kód ktorý vysielala kľúčenka pri stlačení tlačidla na odomknutie. Na vrchnej



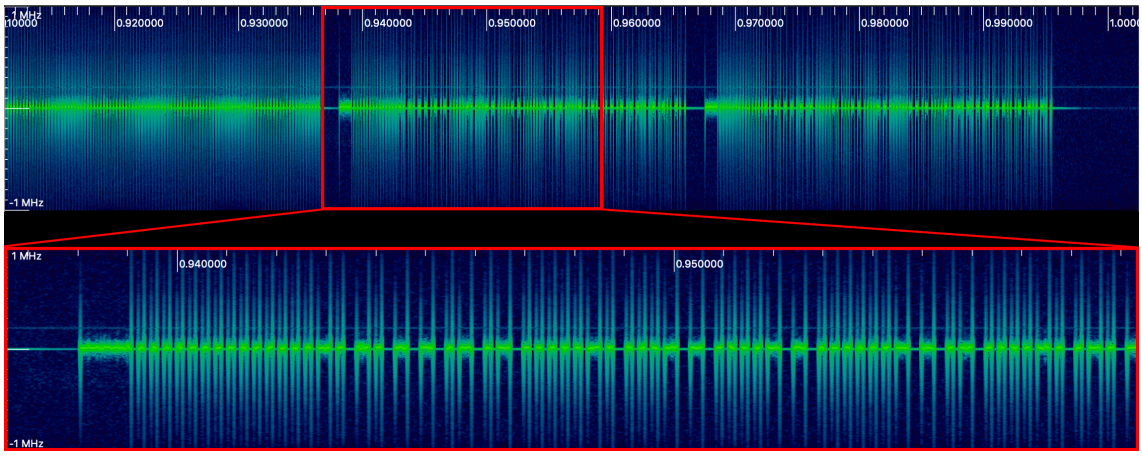

Obr. 2.6: Vodopádový graf signálu kľúčenky pri zamykaní a odomykaní

časti obrázku vidíme celý pribeh signálu. Vyzerá to tak, že kľúčenka používa ASK moduláciu. Môžeme si všimnúť že správa sa skladá z troch častí. Najprv kľúčenka vyšle preambulu zloženú z opakovaných 1 a 0, ktorá označuje začiatok správy. Za ňou nasleduje dvakrát ten istý autorizačný kód. Kľúčenka tento kód odosiela dvakrát aby zvýšila šancu, že kód automobil prijíme bezchybne. Pri opätovnom stláčaní odomykacieho tlačidla na kľúčenke sme pozorovali prenášaný kód a zistili sme, že sa jedná o postupný kód, pretože každý z odoslaných kódov sa líšil. Taktiež sme pozorovali zmenu iba v druhej časti kódu. To znamená že prenášaný autorizačný kód na začiatku pravdepodobne obsahuje identifikátor automobilu a prípadne metadáta obsahujúce identifikátor akcie ktorú chcela kľúčenka vykonať (odomknutie, zamknutie).

## 2.4 Návrh útoku

Predpokladajme že pri riešení práce budeme mať k dispozícii automobil s aktívnym RKS. Väčšina dnešných automobilov s aktívnymi RKS používa autentifikáciu s postupným kódom (rolling code), čo nám dáva možnosť realizovať viacero typov útokov. My sme si k demoštrácii vybrali práve RollJam útok, pretože je zo všetkých spomínaných najefektívnejší.

Podmienka realizácie tohto útoku je, že musíme vedieť prijímať signál z kľúčenky a zároveň vysielať rušiaci signál, čo vyžaduje dve antény naladené na odlišné frekvencie.




Obr. 2.7: Časový priebeh signálu kľúčenky pri odomykaní

Predpokladajme, že sa budeme zameriavať len na jeden určitý model vozidla ktorý používa autentifikáciu s postupným kódom. Najprv zistíme, na akých frekvenciách a akou rýchlosťou komunikujú kľúčenka a automobil. To docielíme zobrazením frekvenčného spektra pomocou jedného z nástrojov poskytovaných sadou nástrojov GNU Radio, alebo programom Inspectrum a použitím zariadenia USRP1. Pri tomto útoku sa využíva vlastnosť napadnutého RKS, ktorá spočíva v tom, že frekvenčné tolerančné okno na ktorom počúva automobil je spravidla širšie ako to v ktorom bude vysielať kľúčenka. Nedá sa totiž predpokladať, že každá kľúčenka bude vysielať na presne danej frekvencii, preto sa takto musí zohľadniť tolerancia použitých súčiastok RF modulu kľúčenky, ktoré slúžia na naladenie jej frekvencie. My túto vlastnosť vieme pri útoku využiť vo svoj prospech. Povedzme, že automobil počúva na frekvenčnom rozsahu 434,0 – 434.6 MHz a kľúčenka vysiela na 434,5 MHz.

Máme k dispozícii dva samostatné mikrokontroléry s CC1101 modulom. Prvý naladíme presne na frekvenciu kľúčenky, pričom je dôležité aby bol dostatočne selektívny iba v okolí tejto frekvencie, a nekryl sa s naším druhým modulom, pretože by sa rušili. Druhý modul naladíme na frekvenciu ktorá je stále vo frekvenčnom tolerančnom okne vozidla, no zároveň sa nekryje s frekvenciou na ktorej vysiela kľúčenka, povedzme 434,2 MHz.

CC1101 moduly sa pokúsime nakonfigurovať tak, aby boli schopné komunikovať s kľúčenkou a automobilom. Konkrétne budeme musieť nastaviť presne frekvenciu na ktorej komunikujú, komunikačnú rýchlosť a typ modulácie.

V momente kedy aktivujeme kľúčenku, detekujeme prichádzajúci signál vo frekvenčnom rozmedzí automobilu a pomocou prvého modulu tento signál (kód) zaznamenáme a uložíme si ho. Zároveň pomocou druhého modulu zarušíme prostredie, čo zabráni automobilu prijať a rozoznať signál od kľúčenky, pretože sa v jeho frekvenčnom okne objavia dva signály, pričom ten rušiaci bude dokonca silnejší. Automobil

sa neodomkne a pokus o odomknutie opakujeme. Tentokrát rovnakým spôsobom zaznamenáme aj druhú kód, no po skončení prenosu odošleme automobilu zaznamenaný kód z prvého pokusu, čo by malo mať za následok, že automobil sa už tentokrát odomkne. Neskôr skúsime poslať automobilu aj náš druhý zaznamenaný kód, ktorý by mal fungovať na odomknutie.

V rámci tohto testovania útoku sa môžeme ešte zamerať na štruktúru prenášaného kódu – zvykne totiž obsahovať viacero informácii, než len autentifikačný postupný kód – identifikátor vozidla a dátový blok bitov, ktoré špecifikujú požadovanú akciu ktorá sa má vykonať (napríklad odomknutie, zamknutie, prípadne otvorenie kufra). Modifikáciou týchto bitov by sme mali vedieť zneužiť zaznamenaný kód na rôzne typy akcií.

## 2.5 Návrh bezkľúčového systému

V tejto sekcii si priblížime možné návrhy nového bezkľúčového systému, ktorý budeme neskôr implementovať.

Prenášaný kód medzi zariadeniami (kľúčenka a automobil) obsahuje okrem postupného autorizačného kódu aj rôzne metadáta ako identifikačné číslo vozidla a identifikátor akcie ktorá sa má vyvolať. Ak útočník dokáže rozlíšiť ktorá časť kódu sú tieto metadáta, môže ich neskôr po zachytení kódu zmeniť a využiť vo svoj prospech. Navrhnuté zlepšenie spočíva v zašifrovaní prenášanej správy kľúčom ktorý by bol unikátny pre každý pár vozidlo-kľúčenka. Použili by sme symetrickú kryptografiu a to z dôvodu, že pri kľúčenke musíme hľadieť na jej energetickú náročnosť – symetrické šifrovanie je energeticky výhodnejšie ako asymetrická šifra a zároveň tento stupeň bezpečnosti nie je pre nás až tak významný, pretože by sme týmto len sťažili útočníkovi analyzovať prenášanú správu medzi zariadeniami, no útočník by bol stále schopný realizovať RollJam útok.

Ďalší stupeň bezpečnosti by spočíval v istom spôsobe detekcie rušenia signálu. Ak by automobil zaznamenal v jeho frekvenčom pásme rušiaci signál, aktivoval by určitý typ výstražnej signalizácie (svetelná alebo zvuková) čím by upozornil užívateľa, že niečo nie je v poriadku a mal by skontrolovať okolie automobilu, resp. karosériu, na ktorej môže byť umiestnené RollJam zariadenie. Ďalej by sme implementovali invalidáciu platnosti kódov. Spočíva v tom, že po použití kódu vygenerovaného pomocou určitej hodnoty sekvenčného čítača by kódy vygenerované s nižšími hodnotami automaticky prestali byť platné – to znamená že aj keď útočník ukradne kód pomocou RollJam techniky, užívateľ neskôr môže vykonať validnú akciu pomocou kľúčenky keď nebude útočník poblízku a tým invalidovať všetky ostatné predošlé kódy, tzn. že útočník nebude schopný použiť svoj ukradnutý kód.

Ďalší návrh na zlepšenie je skombinovať autentifikáciu s postupným kódom s challenge-reponse technikou. Tu by už zariadenia komunikovali obojsmerne a vedeli by sme navrhnúť sofistikovanejší komunikačný protokol. Komunikáciu by stále iniciovala kľúčenka pomocou postupného kódu. Ak bude prvotná autentifikácia postupným kódom validná, automobil odpovie náhodnou výzvou na ktorú znova kľúčenka odpovedá. Výhoda oproti stávajúcim systémom by bola v tom, že by sa nejednalo o pasívny systém, ale aktívny, tzn. že útočník by nevedel vyvolať náhodnú výzvu sám napr. zatiahnutím za kľučku, ale musel by si počkať na aktiváciu kľúčenky užívateľom a najprv prelomiť ochranu postupným kódom, aby vôbec vedel iniciovať komunikáciu s vozidlom. Až potom by riešil problém náhodnej vyzvy. To by mu jednak významne sťažilo zbieranie náhodných výziev za účelom použitia útoku predikciou náhodnej výzvy alebo slovníkového útoku a samozrejme by sme úplne eliminovali možnosť použitia útoku zosilnením signálu.

Zároveň by sme použili dostatočne dlhé autorizačné kódy, aby sme útočníkovi sťažili možnosť použitia skenovacieho útoku a naviac môžeme implementovať zablokovanie systému na nejaký čas (rádovo desiatky sekúnd) po prijatí určitého počtu nesprávnych autorizačných kódov, čo ešte viac zvýši časovú náročnosť útokov.

Posledný a v finálny (v diplomovej práci realizovaný) návrh je RKS s obojsmerným komunikačným protokolom založeným na challenge-response technike. Základom pre tento systém je šifrovanie 128 bitov dlhých náhodných výziev použitím AES blokovej šifry so 128 bitov dlhým šifrovacím kľúčom. Aby sme predišli útokom s predikciou náhodnej výzvy, bude náhodná výzva generovaná nepredvidateľným zdrojom entropie. Pre väčšiu odolnosť voči útokom bude automobil blokovať všetky pokusy o nadviazanie komunikácie na určitý čas v prípade, že dostane nevalidnú odpoveď od kľúčenky. Systém bude vedieť fungovať v aktívnom a pasívnom móde. Implementačné detaily si popíšeme v ďalších kapitolách.

# 3 Praktická časť - bezkľúčový systém

V tejto kapitole si popíšeme implementáciu nášho bezkľúčového systému – od použitého hardvéru až po detaily softvérovej implementácie a komunikačného protokolu zariadení. Začneme od hardvéru a postupne prejdeme k softvérovej implementácii.

Ako základ pre náš systém sme si vybrali dva AVR mikrokontroléry ATmega328 osadené na doske Arduino Nano. Na rádiovú komunikáciu použijeme bezdrôtové moduly CC1101. Keďže Arduino Nano používa 5V logickú úroveň a CC1101 používa 3,3V úroveň, prepojili sme obe zariadenia pomocou konvertora logických úrovní.

Prvý mikrokontrolér má úlohu automobilu (ďalej len automobil) a druhý má úlohu kľúčenky (ďalej len kľúčenka).

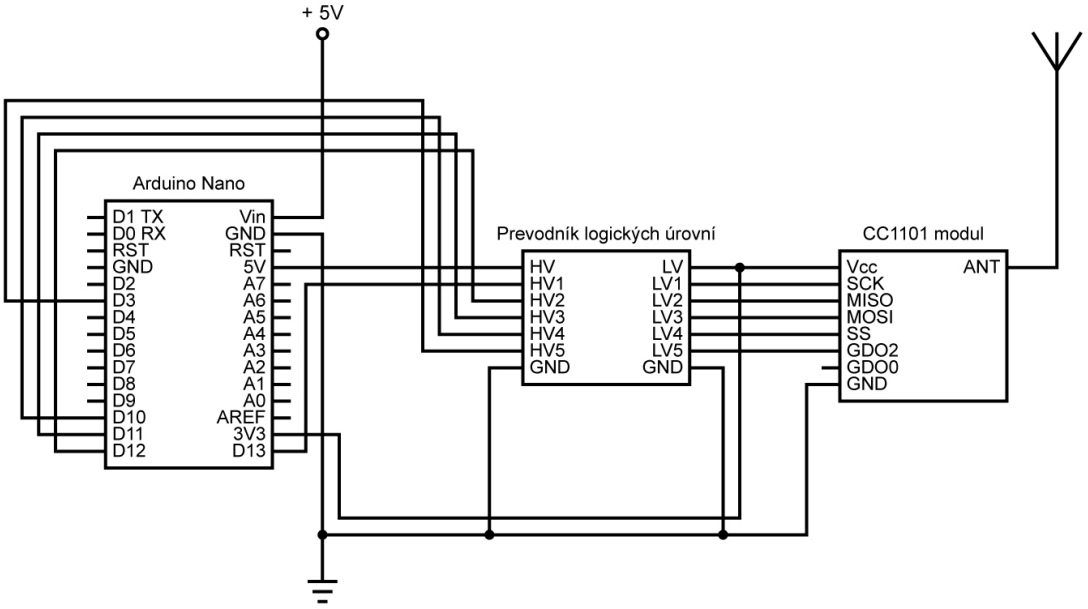
Na kompiláciu kódu oboch zariadení sme použili open-source kompilačný systém PlatformIO, ktorý sa dá ovládať z príkazového riadku. PlatformIO sa stará o kompiláciu nášho zdrojového kódu a následné nahranie strojového kódu do mikrokontroléra. Riadi sa konfiguračným súborom, ktorý obsahuje definície použitého procesora (ATmega328), použitý framework (Arduino), použité knižnice, sériový port pripojeného zariadenia, atp. Podľa toho sa rozhodne, aký kompilátor použiť a akým spôsobom má program do mikrokontroléra nahrať.

Samotné CC1101 moduly uvedieme do prevádzky nakonfigurovaním ich 47 konfiguračných registrov, lepšie povedané zápisom do ich registrov pomocou SPI zbernice z mikrokontroléra. Keďže je náročné vygenerovať ručne všetky hodnoty konfiguračných registrov tak, aby CC1101 fungovalo správne, poskytuje Texas Instuments program SmartRF™ Studio, pomocou ktorého sme vygenerovali počiatočnú konfiguráciu registrov, ktorú sme neskôr upravili podľa našich požiadaviek.

## 3.1 Schéma zapojenia

Obe zariadenia (automobil a kľúčenku) sme zapojili podľa schémy na obrázku 3.1. Arduino piny 13 (SCK), 12 (MISO), 11 (MOSI) a 10 (SS) sú určené na komunikáciu s CC1101 po SPI zbernici.

CC1101 modul má k dispozícii aj dva GPIO piny (GDO0 a GDO2) ktoré vedia signalizovať rôzne stavy čipu. Ich chovanie je konfigurovateľné zápisom do konfiguračného registra určeného pre daný pin. My používame pin GDO2 nakonfigurovaný tak, že stále po prijatí paketu s validným CRC sa jeho hodnota zdvihne na logickú 1 a po prečítaní prvého prijatého bajtu z RX FIFO fronty CC1101 modulu sa jeho hodnota vráti späť na logickú 0. Na tento pin máme v programe nastavené prerušenie, ktorým si signalizujeme, že boli prijaté dáta.




Obr. 3.1: Schéma zapojenia zariadení

## 3.2 PlatformIO

PlatformIO tvorí open-source ekosystém pre vývoj na IoT platformách. Jeho cieľom je uľahčiť vývoj tým, že sa snaží spájať pod jednou strechou väčšinu dostupných „embedded“ (doslovný preklad vstavaných) platforiem a vytvoriť pre nich jednotný ekosystém, pomocou ktorého je pre ne možné vyvíjať software bez potreby udržiavať si na svojom PC všetky potrebné toolchainy (sady nástrojov) potrebné pre vývoj. Navyše má aktívnu komunitu, ktorá neustále prispieva a udržiava dostupné platformy, pridáva nové definície dosiek, knižnice, atp.

Skladá sa z dvoch hlavných častí – prvú tvorí rozšírenie do vývojového prostredia nazvané PlatformIO IDE (sú podporované všetky populárne IDE, od Visual Studio Code až po textové editory ako Sublime alebo Atom). Pomocou IDE rozšírenia môžeme v jeho grafickom rozhraní zakladať nové projekty, vyhľadávať nové knižnice, inštalovať nové sady nástrojov pre rozličné platformy, frameworky, atp. Vo svojom základe ale toto rozšírenie pracuje so svojim jadrom – druhou, základnou komponentou zvanou PlatformIO Core.

PlatformIO Core je sada utilít príkazového riadku, ktorá implementuje celú funkcionalitu tohto ekosystému. Core je multiplatformné, čo znamená že ho vieme využívať na všetkých operačných systémoch. Základom pre PlatformIO je jeho konfiguračný súbor `platformio.ini` ktorý musí byť uložený v koreňovom adresári dokumentu.

Ak potrebujeme vytvoriť nový prázdny PlatformIO projekt, môžeme tak ľahko urobiť príkazom `platformio init` alebo `pio init` (príkaz `platformio` má aj alias `pio` ktorý budeme používať, pretože sa píše rýchlejšie). Tento príkaz vytvorí prázdny projekt s nasledujúcou adresárovou štruktúrou:

```
sampleproject
├── include/
├── lib/
├── platformio.ini
├── src/
└── test/
```

Pre nás je podstatný súbor `platformio.ini` a priečinky `include/`, `lib/` a `src/`. Priečinok `src/` obsahuje zdrojové kódy programu. Priečinok `include/` môže obsahovať hlavičkové súbory zdieľané medzi jednotlivými prostrediami projektu a do priečinku `lib/` môžeme umiestniť knižnice ktoré budú taktiež zdieľané medzi jednotlivými prostrediami.

### 3.2.1 Nastavenie projektu

Náš konfiguračný súbor `platformio.ini` môžeme vidieť na výpise 3.1

Výpis 3.1: Konfiguračný súbor `platformio.ini`

```
; Car computer board environment
[env:car]
platform = atmelavr
board = nanoatmega328
framework = arduino
upload_port = /dev/cu.usbserial-14142312
monitor_speed = 9600
src_filter = +<car/>

; Key fob board environment
[env:fob]
platform = atmelavr
board = nanoatmega328
framework = arduino
upload_port = /dev/cu.usbserial-14142311
monitor_speed = 9600
src_filter = +<fob/>
```

Všimnime si, že súbor obsahuje dve sekcie začínajúce kľúčovým slovom `env:` a končiace názvom prostredia. Každá táto sekcia definuje už spomínané prostredie. Každé prostredie môžeme chápať ako samostatnú variantu nášho programu. My sme náš projekt rozdelili na dve prostredia – jedno pre automobil a druhé pre kľúčenku. Každé prostredie má definovanú platformu pre ktorú sa program kompiluje, typ použitej dosky, použitý framework, port na ktorom je zariadenie pripojené, dokonca si môžeme špecifikovať aj rýchlosť sériovej linky ak by sme chceli využiť serial monitor z PlatformIO Core. PlatformIO ponúka veľa možností ako nakonfigurovať projekt a vyššie uvedená konfigurácia je len zlomok toho, čo sa dá cez konfiguračný súbor špecifikovať.

Zdrojové kódy pre automobil a kľúčenku sme rozdelili do podpriečinkov takto:

```
src
├── car
│   └── main.cpp
└── fob
    └── main.cpp
```

To nám umožnilo vyfiltrovať pomocou premennej `src_filter` v konfiguračnom súbore zdrojové kódy, ktoré sú určené iba pre dané prostredie.

Následne môžeme pracovať s príkazom `pio` nasledovne:

- Príkaz `pio run` skompiluje všetky zdrojové kódy v priečinku `src/`
- Argumentom `-t` môžeme bližšie špecifikovať čo má `pio run` urobiť. Napríklad `pio run -t upload` skompiluje zdrojové kódy a nahraje ich do pripojených zariadení na portoch špecifikovaných v konfiguračnom súbore
- Argumentom `-e` môžeme obmedziť vykonávanie príkazu `pio run` iba na jedno prostredie. Napríklad `pio run -t upload -e fob` skompiluje a nahraje program len do kľúčenky

Výhoda PlatformIO teda spočíva v tom, že je flexibilné a veľmi dobre konfigurovateľné. Ak by sme chceli napríklad portovať program automobilu na inú platformu (povedzme ESP32), PlatformIO nám to ľahko umožní tým, že pre ňu vytvoríme samostatné prostredie, nadefinujeme typ použitej dosky a prípadne portujeme zdrojové kódy, pričom môžeme stále využívať zdieľané knižnice ako u ostatných prostredí, pretože tie by mali byť v dobre nastavenom projekte nezávislé na platforme.

## 3.3 CC1101

Tento čip obsahuje radu registrov, z ktorých môžeme čítať, alebo do nich zapisovať. Väčšina registrov je konfiguračných (je povolený zápis, aj čítanie), potom existujú registre stavové (je povolené iba čítanie) z ktorých sa vieme dozvedieť rôzne informácie o stave čipu (verziu čipu, stav interného stavového automatu, počet bajtov v RX fronte, atp.) a nakoniec je tu tretí typ registrov ktoré sú príkazové (tzv. Command Strobe registers), ktorých adresáciou vieme čipu poslať inštrukciu na zmenu stavu (napríklad vstup do vysielacieho módu, vymazanie TX fronty, atp.). Každá adresa registra má veľkosť 1 B. Každý stavový alebo konfiguračný register môže mať hodnotu max. 1 B.

Existujú 3 formy komunikácie s čipom:

1. Command Strobe zápis – funguje tak, že po SPI zbernici pošleme čipu iba jeden bajt (adresu príkazového registra), čo má za následok že čip vykoná požadovanú akciu.
2. Zápis a čítanie z jedného registra – pri zápise do registra najprv pošleme po SPI adresu daného registra a za ňou hodnotu ktorú doň chceme zapísať. Pri čítaní musíme poslať po SPI adresu registra z ktorého chceme čítať, sčítanú logickým OR-om hodnotou 0x80 (Read Offset). Následne nám čip pošle po SPI hodnotu tohto registra.
3. Zápis a čítanie z viacerých registrov naraz, tzv. Burst mode – Burst mód nám umožňuje zapisovať a čítať z viacerých vedľa seba ležiacich registrov (tzn. že ich adresy nadväzujú na seba, napr. 0x00, 0x01, 0x02, 0x03, ...) v jednej SPI transakcii. Pri zápise posielame adresu prvého registra logicky sčítanú s hodnotou 0x40 (Burst Write Offset) a pri čítaní hodnotou 0xC0 (Burst Read Offset). Následne začne čip pri posielaní hodinového signálu vraciať na MISO pine postupne hodnoty všetkých registrov ktoré nadväzujú na seba (interne inkrementuje čítač ktorý ukazuje po každých 8 prečítaných bitoch na ďalší register v poradí). Pomocou burst módu vieme efektívne zapisovať všetky konfiguračné registre naraz, alebo zapisovať do FIFO front.

Čítanie prijatých dát a zápis dát určených na odoslanie funguje na koncepte FIFO fronty. Čip obsahuje dve FIFO fronty, každú o dĺžke 64 bajtov. Jedna je určená pre dáta čakajúce na odoslanie (TX FIFO) a jedna pre prijaté dáta (RX FIFO). Ak čip začne prijímať dáta, začne ich plniť do RX FIFO fronty, ktorú môžeme pomocou Burst módu prečítať. Podobne to funguje aj pri odosielaní – najprv zapíšeme pomocou Burst módu dáta do TX FIFO fronty a potom pomocou Command Strobe príkazu nastavíme čip do vysielacieho módu. Vtedy čip automaticky vyšle von všetky dáta vo fronte.

### 3.3.1 Konfigurácia čipu

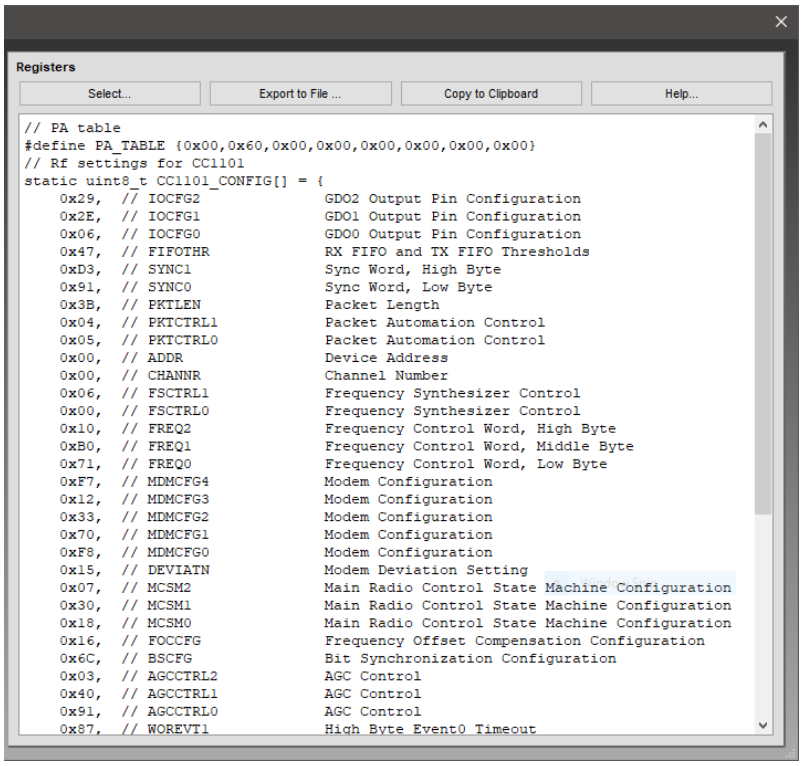
Pre náš projekt sme zvolili konfiguráciu čipu zhrnutú v tabuľke 3.1 ktorá obsahuje najdôležitejšie vlastnosti. Konfiguračné registre sme vygenerovali pomocou programu SmartRF™ Studio. Na obrázku 3.2 vidíme snímku z obrazovky tohto programu pri exporte hodnôt konfiguračných registrov.

Tab. 3.1: Konfigurácia čipu CC1101

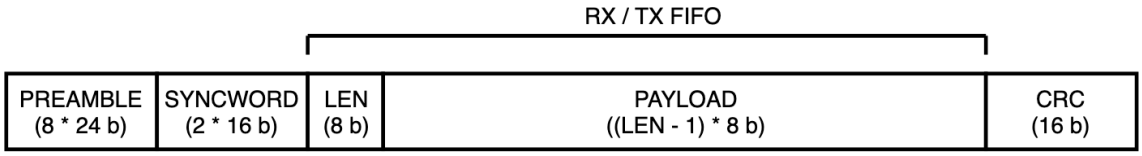
| | |
|---|---|
| Nosná frekvencia | 433,9 MHz |
| Odstup kanálov | 50 kHz |
| Číslo kanálu | konfigurovateľné pri kompilácii |
| Modulačná rýchlosť | 50 kBd |
| Výkon antény | 0 dBm / -20 dBm |
| Modulácia | ASK |
| Dĺžka preambuly | 24 B |
| Synchronizačné slovo | konfigurovateľné pri kompilácii |
| Dĺžka synchronizačného slova | 2 * 16 b |
| Dĺžka paketu | variabilná |
| Adresácia paketov | vypnutá |
| Skrambler (Data whitening) | zapnutý |
| Výpočet CRC | zapnutý |

Vysielame na nelicencovanej frekvencii okolo 434 MHz. Použili sme základnú ASK moduláciu. Číslo kanálu a synchronizačné slovo vieme konfigurovať priamo vo firmvéri. Synchronizačné slovo má dĺžku 2 B, no čip je nastavený tak, aby ho posielal 2-krát. Taktiež sme zapli automatické skramblovanie dát (skramblujú sa len prenášané dáta, preambula a synchronizačné slovo nie), aby mal prenášaný signál čo najmenšiu jednosmernú zložku. V aktívnom režime je nastavený čip na výkon 0 dBm, v pasívnom režime na -20 dBm kvoli tomu, aby vedela kľúčenka s automobilom komunikovať iba na malú vzdialenosť.

Na obrázku 3.3 vidíme štruktúru CC1101 rámca (oficiálna dokumentácia pracuje s pojmom paket) vzhľadom na nami použitú konfiguráciu. Používame variabilnú dĺžku paketu, čo znamená, že prvý bajt (LEN) nasledujúci za synchronizačným slovom označuje celkový počet bajtov vo FIFO fronte. CC1101 podporuje aj filtráciu paketov na hardvérovej úrovni tak, že za LEN bajt by sme ešte vložili bajt s adresou príjemcu a v konfiguračných registroch nastavili adresu zariadenia a zapli samotnú filtráciu. Túto funkciu ale nepoužívame z dôvodu, že adresa o veľkosti 1 B je pre naše účely malá. Preto filtrujeme pakety na softvérovej úrovni vo firmvéri, kde na to používame 32 bitov dlhý identifikátor systému.




Obr. 3.2: Export hodnôt registrov z programu SmartRF™ Studio




Obr. 3.3: CC1101 paket

Pri odosielaní a prijímaní dát pracujeme len s FIFO frontou (vyznačená na obrázku 3.3) ktorú pred odoslaním naplníme. Následne CC1101 dáta vo fronte enkapsuluje medzi preambulu, synchronizačné slovo a vypočítaný kontrolný súčet. Po prijatí sa pozrie, či je synchronizačné slovo rovnaké ako to nakonfigurované a či sedí CRC súčet. Ak áno, jedná sa o validný paket – dáta rozbalí (zahodí preambulu, synchronizačné slovo a CRC) a sprístupní ich nám v RX FIFO fronte. Ak nie, tak dáta zahodí. CC1101 ponúka nastavenie kvalifikátora synchronizačného slova, čo v postate znamená, že môžeme špecifikovať, koľko bitov zo synchronizačného slova musí byť rovnakých, aby bol paket validný. My používame nastavenie 30/32, čo znamená že dovolená chybovosť môže byť 2 bity.

Čip je nakonfigurovaný tak, aby po prijatí validného paketu nastavil svoj GDO2 GPIO pin na logickú 1. Na tento pin máme nastavené v mikrokontroléri prerušenie, ktorým zistíme že je vo FIFO fronte k dispozícii paket.

### 3.3.2 Prehľad konfiguračných registrov

V tabuľke 3.2 nájdeme stručný prehľad použitých konfiguračných registrov. Väčšina registrov slúži na nastavenie viacerých funkcií zároveň, my uvádzame len tie, s ktorými sme pracovali.

Tab. 3.2: Prehľad konfiguračných registrov CC1101

facebook

| Adresa | Názov | Popis |
|---|---|---|
| 0x00 | IOCFG2 | Funkcia GDO2 pinu |
| 0x04 | SYNC1 | Prvý (vyšší) bajt synchronizačného slova |
| 0x05 | SYNC2 | Druhý (nižší) bajt synchronizačného slova |
| 0x07 | PKTCTRL1 | Obsahuje nastavenia filtrácie paketov |
| 0x08 | PKTCTRL0 | Zapína sa tu skrambler a CRC výpočet |
| 0x0A | CHANNR | Číslo prenosového kanálu |
| 0x0D | FREQ2 | Nastavenie nosnej frekvencie |
| 0x0E | FREQ1 | Nastavenie nosnej frekvencie |
| 0x0F | FREQ0 | Nastavenie nosnej frekvencie |
| 0x11 | MDMCFG1 | Nastavenie prenosovej rýchlosti |
| 0x12 | MDMCFG2 | Nastavenie modulácie, kvalifikátor synchronizačného slova |

## 3.4 Firmvér zariadení

Program, bežiaci na oboch zariadeniach je napísaný v jazyku C++. Na komunikáciu s čipom CC1101 sme si vytvorili vlastnú knižnicu, ktorá má na starosť zápis prvotnej konfigurácie čipu, nastavenie jednotlivých konfiguračných registrov a samozrejme tiež posielanie a prijímanie dát. Projekt obsahuje hlavičkový súbor `constants.h` ktorý je zdieľaný medzi zdrojovým kódom kľúčenky a automobilu. Tento súbor je určený na konfiguráciu celého systému kľúčenky a automobilu. Súbor možno vidieť na výpise 3.2.

Na začiatku programu sa pripojíme pomocou knižnice k čipu CC1101 a zapíšeme do jeho registrov celú konfiguráciu. Potom nastavíme prerušenie na pin 3 ktorý je pripojený k CC1101 pinu GDO2 a začneme počúvať na prichádzajúce pakety.

V aktívnom režime kľúčenka podporuje dve tlačidlá – jedno na odomknutie a jedno na uzamknutie vozidla. Po stlačení tlačidla sa kľúčenka autorizuje. V pasívnom režime beží na kľúčenke časovač, ktorý každých pár sekúnd odošle automobilu signál o tom, že je poblízku a autorizuje sa.

### 3.4.1 Konfiguračný hlavičkový súbor

Tu si popíšeme jednotlivé konštanty uvedené v súbore `constants.h`

Výpis 3.2: Hlavičkový súbor `constants.h`

```
#define SYNC_WORD 0xD391
#define CHANNEL_NUMBER 0x00
#define DATA_WHITENING_ENABLED true

#define SYSTEM_ID 0x60a3d109
#define ENCRYPTION_KEY \
  { \
    0x2b, 0x7e, 0x15, 0x16, \
    0x28, 0xae, 0xd2, 0xa6, \
    0xab, 0xf7, 0x15, 0x88, \
    0x09, 0xcf, 0x4f, 0x3c  \
  }

#define SYSTEM_FLAG_ACTIVE 1
#define SYSTEM_FLAG_PASSIVE 0

#define CAR_SAFETY_TIMER_MS 5000
#define FOB_ACTION_TIMEOUT_TIME_MS 2000

#define CAR_PASSIVE_UNLOCK_TIMEOUT_MS 5000
#define PASSIVE_RESPONSE_TIME_THRESHOLD_MICROS 16000
```

#### Synchronizačné slovo (`SYNC_WORD`)

Synchronizačné slovo má dĺžku 16 bitov a môžeme ho chápať ako unikátny identifikátor výrobcu automobilu, resp. akúsi prvotnú filtráciu paketov už na hardvérovej úrovni. Automobil, ani kľúčenka nebudú prijímať pakety ktoré sa začínajú synchronizačným slovom iným ako to, ktoré majú nakonfigurované.

#### Číslo kanálu (`CHANNEL_NUMBER`)

Číslo kanálu určuje frekvenčný posun signálu kľúčenky a automobilu. Týmto číslom sa násobí kanálový odstup (50 kHz) a pridáva sa k nosnej frekvencii. Napríklad, ak používame nosnú frekvenciu 433,90 MHz a kanál číslo 1, budú zariadenia vysielať na frekvencii 433,95 MHz, pre kanál číslo 2 je to frekvencia 434,00 MHz atď.

#### Skrambler (DATA_WHITENING_ENABLED)

Touto konštantou vieme zapínať alebo vypínať použitie skramblera.

#### ID systému (SYSTEM_ID)

ID systému je 32 bitov dlhé číslo, ktoré unikátne identifikuje skupinu automobilu a jeho kľúčeniek. Toto číslo slúži na filtráciu paketov na softvérovej úrovni. Iba automobil a kľúčenka s rovnakým systémovým ID vedia spolu komunikovať.

#### Šifrovací kľúč (ENCRYPTION_KEY)

Toto je 128 bitov dlhý šifrovací kľúč určený k šifrovaniu náhodných výziev pomocou AES blokovej šifry. Tento kľúč musí byť taktiež unikátny pre každú skupinu automobilu a jeho kľúčeniek.

#### Príznak pre aktívny režim (SYSTEM_FLAG_ACTIVE)

Zapnutím tohoto príznaku nastavíme celý systém do aktívneho módu, kedy musí autorizáciu vyvolať užívateľ stlačením tlačidla na kľúčenke.

#### Príznak pre pasívny režim (SYSTEM_FLAG_PASSIVE)

Zapnutím tohoto príznaku nastavíme celý systém do pasívneho režimu, kedy kľúčenka vysiela periodicky požiadavky na autorizáciu.

#### Bezpečnostný časovač automobilu (CAR_SAFETY_TIMER_MS)

Ak si automobil vyžiada od kľúčenky zašifrovanú náhodnú výzvu a kľúčenka mu ju vráti nesprávne zašifrovanú, môžeme sa domnievať, že sa s automobilom snaží komunikovať útočník. V takom prípade prestane automobil odpovedať na všetky pokusy o komunikáciu po dobu definovanú v tejto konštante v milisekundách.

#### Čas návratu kľúčenky do nečinného módu (FOB_ACTION_TIMEOUT_TIME_MS)

Po stlačení tlačidla na kľúčenke, kľúčenka vyšle automobilu žiadosť vykonať jednu z akcií (odomknúť, uzamknúť) a zmení svoj interný stav z nečinného, na aktívny. Potom kľúčenka čaká na odpoveď od automobilu po maximálnu dobu špecifikovanú v tejto konštante v milisekundách. Potom sa vráti späť do nečinného stavu. Implementáciou týchto stavov chceme zabrániť možnosti slovníkového útoku, kde útočník posiela kľúčenke náhodné výzvy a kľúčenka odpovedá šifrovanými výzvami. Takto kľúčenka odpovedá iba v aktívnom stave ktorý trvá veľmi krátko.

#### Čas automatického uzamknutia vozidla (`CAR_PASSIVE_UNLOCK_TIMEOUT_MS`)

Ak pracuje systém v pasívnom režime a v blízkosti nie je žiadna kľúčenka, automobil sa uzamkne po čase definovanom v tejto konštante v milisekundách.

#### Najdlhší povolený čas odozvy (`PASSIVE_RESPONSE_TIME_THRESHOLD_MICROS`)

Ak pracuje systém v pasívnom režime, musíme sa brániť voči útoku zosilnením signálu (útok dvoch zlodejov) meraním odozvy kľúčenky. Ak stúpne dĺžka odozvy kľúčenky nad túto hranicu v mikrosekundách, môžeme predpokladať že sa medzi kľúčenkou a automobilom nachádza útočník, ktorý zosilňuje signál a tým pádom zväčšuje odozvu kľúčenky. Vtedy je autorizácia hoci aj správnou zašifrovanou výzvou neplatná.

### 3.4.2 Komunikačný protokol

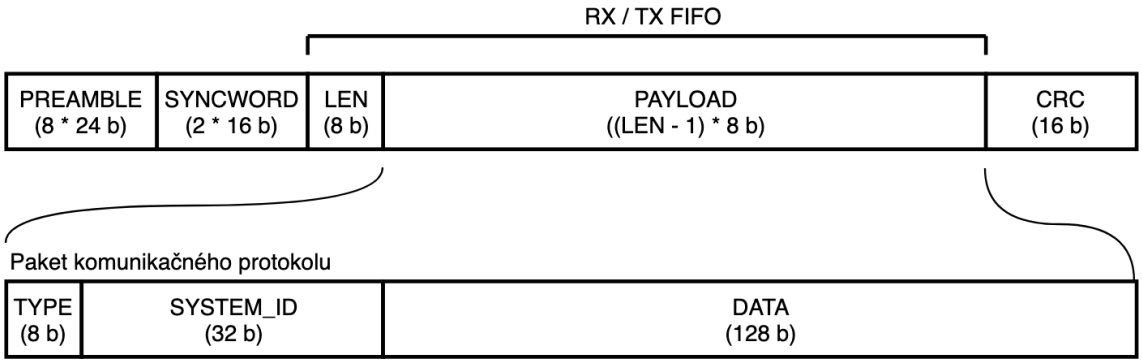
Komunikačný protokol zariadení je obojsmerný a je založený na Challenge-Response technike spomínanej už v teoretickej časti práce. Náhodné výzvy majú dĺžku 128 bitov a sú na strane kľúčenky šifrované symetrickou AES šifrou so 128 bitov dlhým šifrovacím kľúčom. Výzvy generujeme v automobile hardvérovým generátorom náhodných čísel za pomoci tzv. extraktora náhodnosti, čím sme zaručili že vygenerované čísla sú dokonale náhodné, tzn. nedajú sa predikovať a neopakujú sa.

Náš protokol je stavový, čo znamená že obe zariadenia si držia stav o tom, na ktorom kroku sa nachádza priebeh výmeny správ medzi nimi. Tým sme zaručili, že so zariadeniami sa nedá nadviazať komunikácia len poslaním náhodnej výzvy kľúčenke (ak by si chcel útočník vytvoriť slovník), alebo len poslaním odpovede automobilu s náhodnou zašifrovanou výzvou bez toho aby si ju automobil predtým nevyžiadal.

#### Štruktúra paketu

Na obrázku 3.4 je znázornená štruktúra paketu komunikačného protokolu a taktiež vizualizácia toho, ako je tento paket zasadený do FIFO fronty v CC1101. Na výpise 3.4 je definícia paketu z jeho hlavičkového súboru.

Paket sa začína 8-bitovým číslom označujúcim typ paketu. Je to jedno z čísel definovaných vo výpise 3.3.




Obr. 3.4: Štruktúra paketu komunikačného protokolu

Výpis 3.3: Výpis typov paketu

```
#define PACKET_REQUEST_LOCK 0x01
#define PACKET_REQUEST_UNLOCK 0x02
#define PACKET_REQUEST_PROXIMITY_UNLOCK 0x03
#define PACKET_RESPONSE_RANDOM_CHALLENGE 0x04
#define PACKET_RESPONSE_ENCRYPTED_CHALLENGE 0x05
#define PACKET_RESPONSE_OK 0x06
```

Za ním nasledude identifikátor systému, čo je unikátne 32-bitové číslo spoločné pre každú skupinu automobilu a jeho kľúčeniek. Za týmto číslom je alokovaných 128 bitov určených na prenos dát. Dátový blok sa využíva len pri odoslaní náhodnej výzvy automobilom, alebo pri odpovedi z kľúčenky, kde kľúčenka posiela späť automobilu zašifrovanú náhodnú výzvu.

Výpis 3.4: Definícia štruktúry paketu

```
#define PACKET_CHALLENGE_LEN 0x10 // 16 bytes

typedef struct {
  uint8_t type;
  uint32_t system_id;
  uint8_t payload[PACKET_CHALLENGE_LEN];
} PacketData;
```

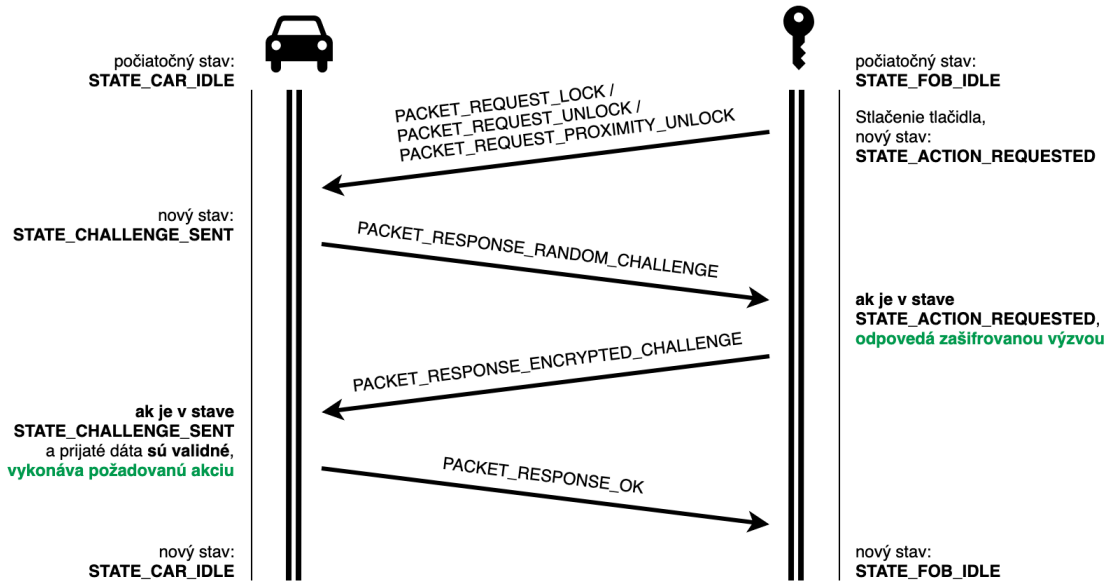
### Výmena správ

Komunikáciu stále nadväzuje kľúčenka. V aktívnom režime musí užívateľ stlačiť tlačidlo na kľúčenke, v pasívnom režime kľúčenka sama vysiela požiadavky. Pozrime sa na obrázok 3.5.

V aktívnom režime kľúčenka po stlačení tlačidla odošle automobilu žiadosť o odomknutie, alebo uzamknutie (typ paketu `PACKET_REQUEST_LOCK`

alebo `PACKET_REQUEST_UNLOCK`). V pasívnom režime sa používa typ paketu `PACKET_REQUEST_PROXIMITY_UNLOCK`.

Automobil následne vygeneruje 128 bitov dlhú náhodnú výzvu a pošle ju kľúčenke. Kľúčenka túto náhodnú výzvu zašifruje svojim šifrovacím kľúčom a pošle späť automobilu. Po prijatí ju automobil dešifruje svojim šifrovacím kľúčom a obe výzvy (prijatú a odoslanú) porovná. Ak sa výzvy zhodujú, automobil vykoná kľúčenkou požadovanú akciu a odošle kľúčenke potvrdenie o vykonaní akcie.



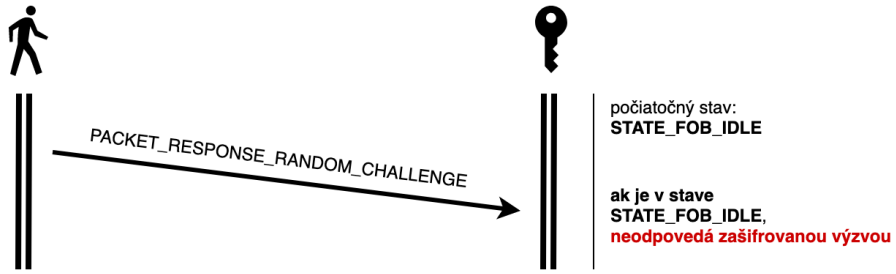

Obr. 3.5: Priebeh výmeny správ medzi kľúčenkou a automobilom

V prípade, že sa výzvy nezhodujú, automobil zablokuje na 5 sekúnd všetky pokusy o nadviazanie komunikácie. Vtedy predpokladáme, že existuje útočník, ktorý takto skúša odomknúť vozidlo bez znalosti šifrovacieo kľúča. Týmto spôsobom mu z časového hľadiska sťažíme realizovať útok na automobil.

Na obrázku 3.5 si taktiež všimnime, že automobil, aj kľúčenka si držia určitý stav. Tento stav slúži na to, aby sme eliminovali dve možné bezpečnostné chyby v systéme.

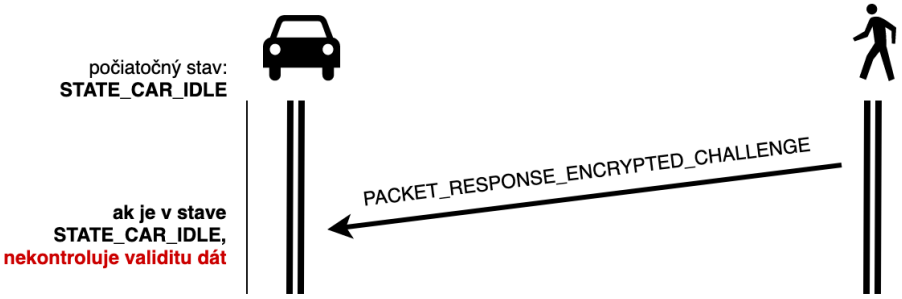
Prvou chybou by bolo, ak by mohol útočník posielať kľúčenke náhodné výzvy a kľúčenka by odpovedala zašifrovanými výzvami bez toho, aby si ich predtým sama vyžiadala. Útočník tak môže robiť za účelom zozbierania dát k slovníkovému útoku. V našom systéme môže kľúčenka odpovedať zašifrovanou výzvou iba vtedy, ak si ju prvotne vyžiada (užívateľ stlačí tlačidlo na kľúčenke) a to po dobu maximálne dvoch sekúnd.

Druhou chybou by bolo, ak by mohol útočník posielať automobilu „zašifrované“ výzvy bez toho, aby si ich predtým automobil vyžiadal. Automobil totiž stále po kontrole správnosti prijatej výzvy vyresetuje svoju generovanú výzvu na nulovú




Obr. 3.6: Blokovanie komunikácie kľúčenky s potencionálnym útočníkom

hodnotu, aby nebola znova použiteľná (výzvu takzvane invalidujeme). Útočník by takto mohol prísť na túto vlastnosť a poslať automobilu rovno odpoveď typu `PACKET_RESPONSE_ENCRYPTED_CHALLENGE` bez dát, čo by automobil vyhodnotil ako validnú odpoveď. Toto za použitia stavov nie je možné, viď. obrázok 3.7.




Obr. 3.7: Blokovanie komunikácie automobilu s potencionálnym útočníkom

### 3.4.3 Rozdiely medzi aktívnym a pasívnym režimom

Ako sme už spomínali, v aktívnom režime vyvoláva užívateľ autorizáciu sám, stlačením tlačidla na kľúčenke a v pasívnom režime vysiela kľúčenka požiadavku na autorizáciu periodicky každé dve sekundy. V pasívnom režime používame znížený výkon antény, aby sa automobil odomkol iba vtedy, ak je užívateľ poblízku (cca do 10 metrov).

V pasívnom režime automobil implementuje časovač, ktorým automaticky uzamkne vozidlo do 5 sekúnd od momentu, kedy sa prestane kľúčenka autorizovať – tzn. že sa vzdiali.

### 3.4.4 Generovanie náhodných výziev

Aby sme zaručili, že automobilom generované výzvy budú vhodné na použitie v systéme ktorý využíva kryptografické zabezpečenie, sme museli vymyslieť spôsob, ako generovať čísla ktoré spĺňajú tieto dve podmienky:

- Čísla sa nebudú opakovať
- Nebude možné predpovedať vygenerované čísla

Použitie pseudo-náhodného generovania čísel teda nepripadalo v úvahu a museli sme implementovať lepší spôsob. Zvolili sme cestu extraktora náhodnosti, konkrétne implementáciu Von Neumannovho extraktora.

Extraktor náhodnosti je funkcia, ktorá po aplikácii na zdroj s nízkou entropiou generuje vysoko náhodný výstup, ktorý sa zdá byť nezávislý od tohto zdroja a zároveň má rovnomerné rozdelenie. [11]

Ako nepredvídateľný zdroj entropie sme zvolili jeden z nepripojených pinov na Arduino doske, využívajúci A/D prevodník (tzv. analog pin). Výstup z tohto prevodníka tvorí náš zdroj.

Von Neumannov extraktor funguje nasledovne. Vstupom do extraktora je sekvencia bitov, kde je pravdepodobnosť výskytu 1 alebo 0 rovnaká. Extraktor si vezme z tejto sekvencie dva bity a porovná ich. Ak sú rovnaké, nevyprodukuje žiaden výstup. Ak sa bity líšia, je výstupom extraktora prvý z bitov.

Našu implementáciu vidíme na výpise 3.5.

Výpis 3.5: Implementácia generátora náhodných čísel

```
uint8_t generate_random_byte() {
  uint8_t bits_stored = 0;
  uint8_t result = 0x00;

  while (bits_stored < 8) {
    uint8_t bit1 = ((uint8_t)analogRead(A0)) & 0x01;
    delayMicroseconds(64);
    uint8_t bit2 = ((uint8_t)analogRead(A0)) & 0x01;

    if (bit1 != bit2) {
      result |= (bit1 << bits_stored);
      bits_stored++;
    }

    delayMicroseconds(64);
  }

  return result;
}
```

Najprv prečítame dva najmenej významné bity z A/D prevodníka s odstupom 64 mikrosekúnd. Odstup 64 mikrosekúnd preto, že A/D prevodník z nepripojeného pinu nemá dostatočne vysoké rozlíšenie a nemení hodnotu svojho výstupu tak rýchlo, aby

sme mohli tvrdiť že pravdepodobnosť výskytu 1 a 0 medzi dvoma čítaniami bude rovnaká. Ak sú bity rozdielne, uložíme si prvý bit a bitovým posunom ho pridáme do výsledného bajtu. Toto opakujeme 8-krát, až kým nevytvoríme jeden bajt.

Aby sme vytvorili celú náhodnú výzvu o dĺžke 128 bitov, tento proces opakujeme 16-krát, ako na výpise 3.6.

Výpis 3.6: Generovanie náhodnej výzvy

```
#define PACKET_CHALLENGE_LEN 0x10 // 16 bytes

void generate_random_challenge(uint8_t* challenge_buffer) {
  for (uint8_t i = 0; i < PACKET_CHALLENGE_LEN; i++) {
    challenge_buffer[i] = generate_random_byte();
  }
}
```

Tento spôsob generovania náhodných čísel sa ukázal ako veľmi efektívny. Pri testovaní pseudo-náhodného generátora čísel zo štandardnej C knižnice bolo z 1000 vygenerovaných náhodných výziev až 167 rovnakých. Pri testovaní našej implementácie sme vygenerovali 10000 náhodných výziev, pričom medzi nimi nebol jediný duplikát. Nemôžeme ale tvrdiť, že vygenerované čísla sa nikdy opakovať nebudú. Možno budú, ale s veľmi malou pravdepodobnosťou. Vygenerované dáta nájdeme v prílohe na priloženom CD.

Jedinou nevýhodou tejto metódy je dĺžka trvania generovania náhodnej výzvy, ktorá je omnoho väčšia ako u iných metód. Namerali sme 87 milisekúnd pre 128 bitov dlhé číslo. Táto doba je ale pre nás znesiteľný kompromis vzhľadom na efektivitu metódy.

### 3.4.5 Šifrovanie náhodných výziev

Na šifrovanie náhodných výziev používame AES blokovú šifru. Výber padol na AES šifru z viacerých dôvodov. Bezpečnosť – táto šifra sa v dnešnej dobe nedá prelomiť v rozumnom časovom okne. Rýchlosť – je symetrická, čo znamená že je aj rýchla a výpočtovo a energeticky nenáročná, a teda ideálna na implementáciu v mikrokontroléroch.

Šifrovací a dešifrovací kľúč je u symetrických šifier rovnaký. To v našom prípade nevadí, pretože kľúč je súčasťou firmvéru a nikde sa neprenáša.

Keďže je dĺžka náhodnej výzvy 128 bitov a minimálna veľkosť šifrovaného bloku u AES 128 bitov, môžeme použiť iba operačný mód ECB, u ktorého sa nenadväzuje na predchádzajúco-zašifrované bloky. Kľúč ma takisto dĺžku 128 bitov.

Keďže už existuje množstvo voľne dostupných implementácií AES šifry, použili sme už hotovú knižnicu, a to z jediného dôvodu. Nepotrebujeme totiž „znovu vynaliezať koleso“. Väčšina existujúcich implementácií je už otestovaná voči oficiálnym testovacím dátam z NIST (Americký národný inštitút pre štandardy a technológie).

Použili sme knižnicu `tiny-AES-c` <`https://github.com/kokke/tiny-AES-c`>.

Jedná sa o implementáciu AES v jazyku C. Táto knižnica sa orientuje na portabilitu, čo znamená že jej implementácia zaberá po kompilácii programu len málo

miesta. Taktiež je nenáročná na pamäť RAM. Podporuje všetky operačné módy šifry, ktoré sa dajú v jej hlavičkovom súbore zapínať a vypínať, čím vieme ešte viac optimalizovať jej veľkosť po kompilácii programu. Ukážka použitia je vo výpise 3.7.

Výpis 3.7: Príklad použitia knižnice `tiny-AES-c`

```
#include <constants.h>
#include <packet.h>
#include <aes.hpp>

uint8_t challenge_buffer[PACKET_CHALLENGE_LEN];
uint8_t encryption_key[] = ENCRYPTION_KEY;
AES_ctx aes_context;

void example() {
    // Inicializacia kontextu
    AES_init_ctx(&aes_context, encryption_key);

    // Zasifrovanie obsahu challenge_buffer
    AES_ECB_encrypt(&aes_context, challenge_buffer);

    // Desifrovanie obsahu challenge_buffer
    AES_ECB_decrypt(&aes_context, challenge_buffer);
}
```

## 3.5 Odolnosť voči útokom

Pozrime sa na odolnosť nášho systému voči rôznym typom útokov.

### 3.5.1 Útok spätným prehraním (Playback attack)

Tento útok je u nášho systému nemožný, keďže náhodná výzva je platná iba raz. Ak útočník zachytí zašifrovanú výzvu kľúčenky, nemá možnosť ju použiť druhýkrát.

### 3.5.2 Skenovací útok (Scan attack)

Tento útok by bol veľmi neefektívny a z časového hľadiska nemožný. Útočník sa síce môže tváriť ako kľúčenka a skúšať sa autentifikovať náhodnými kódmi, no automobil zablokuje na určitý čas komunikáciu ak autentifikácia nie je úspešná. Naviac používame 128 bitov dlhé autorizačné kódy, čo znamená že uhádnúť správny kód (ktorý je platný iba raz) by bolo len o náhode.

### 3.5.3 Útok zosilnením signálu

Tento útok je aplikovateľný len v pasívnom prežime. Voči tomuto útoku sa bránime meraním času odozvy kľúčenky. Po testovaní sme zistili, že od odoslania náhodnej výzvy kľúčenke, až po prijatie zašifrovanej výzvy ubehne cca 15000 mikrosekúnd. Následne sme zvolili toleranciu 1 ms, čo znamená, že ak od odoslania náhodnej výzvy kľúčenke ubehne viac ako 16000 mikrosekúnd, bude autorizácia neplatná. Predpokladáme, že ak útočník použije útok zosilnením signálu, pridá to na odozve minimálne 1 ms.

### 3.5.4 Útok predikciou náhodnej výzvy

Útok predikciou náhodnej výzvy je založený na analýze pseudo-náhodného generátora. Keďže používame hardvérový generátor náhodných výziev, nie je možné predikovať vygenerovanú náhodnú výzvu. Jediný spôsob, akým sa dá ovplyvniť generovanie náhodnej výzvy je manipulácia s nepripojeným analógovým pinom ktorý je zdrojom entropie generátora.

### 3.5.5 Slovníkový útok

Pri tomto útoku si útočník zbiera páry vygenerovaných a zašifrovaných výziev do slovníka. To môže robiť tak, že odpočúva komunikáciu medzi kľúčenkou a automobilom, alebo posiela kľúčenke náhodné výzvy, ktoré by musela kľúčenka zašifrovať a poslať späť. Druhý spôsob nie je v aktívnom režime možný, pretože kľúčenka v našom systéme odpovedá zašifrovanými výzvami len vtedy, ak na nej užívateľ stlačí tlačidlo. V pasívnom režime, kedy kľúčenka vysiela autorizačné požiadavky sama je náš systém viac náchylný na tento typ útoku.

Ak si už útočník ale vytvorí slovník, je stále veľmi malá šanca, že automobil vygeneruje rovnakú výzvu dvakrát, vďaka spôsobu ktorým výzvy generujeme.

### 3.5.6 RollJam

Tento útok je určený len pre systémy s postupným autorizačným kódom, a teda neaplikovateľný na náš systém.

# 4 Praktická časť - implementácia útoku

V tejto kapitole si ukážeme implementáciu RollJam útoku podľa predchádzajúceho návrhu.

K dispozícii sme mali automobil Škoda Octavia (ročník 2005) s jednou kľúčenkou. Najprv sme pomocou programu `uhd_fft` zistili, na ktorej frekvencii kľúčenka vysiela. Tento program patrí do balíku programov určených na prácu s UHD zariadeniami (USRP1) a zobrazuje FFT analýzu a vodopádový graf signálu v reálnom čase. Potom sme pomocou USRP1 zaznamenali niekoľko kódov z kľúčenky a následne ich v programe Inspectrum analyzovali.

Podľa zistených vlastností signálu sme neskôr nakonfigurovali CC1101 čipy tak, aby sme pomocou nich vedeli zaznamenávať kódy z kľúčenky v reálnom čase a zároveň komunikovať s automobilom.

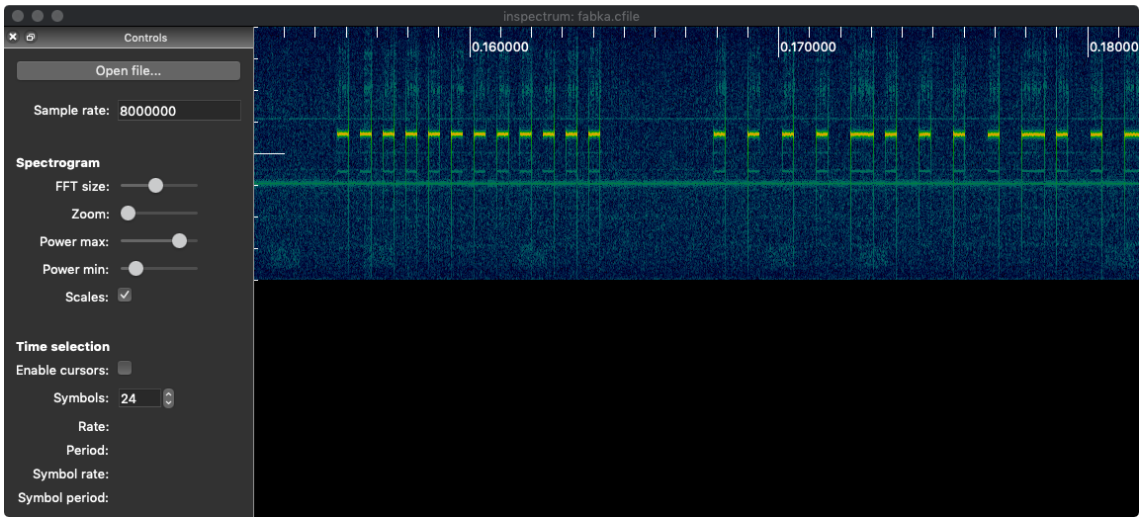
K implementácii sme použili rovnaké zariadenia ako v predchádzajúcej časti venujúcej sa implementácii bezkľúčového systému. Prvý mikrokontrolér slúži na zaznamenávanie signálu kľúčenky a komunikáciu s automobilom a zároveň na ovládanie druhého mikrokontroléra, ktorý má za úlohu vysielať rušiaci signál.

## 4.1 Analýza signálu

Prvým krokom bolo zistiť na akej frekvencii vysiela kľúčenka. Použili sme na to zariadenie USRP1 v kombinácii s programom `uhd_fft`. Zistili sme, že kľúčenka vysiela na frekvenčnom rozmedzí 433,6 - 433,8 MHz. Následne sme pomocou príkazu `uhd_rx_cfile` nahrali do I/Q súboru niekoľko kódov, ktoré sme následne analyzovali pomocou programu Inspectrum. Na obrázku 4.1 vidíme začiatok zaznamenaného kódu s preambulou. Na prvý pohľad je zrejmé, že kľúčenka používa ASK moduláciu.

Inspectrum ponúka aj základnú analýzu modulovaného signálu. Po zapnutí voľby „Enable cursors“ môžeme označiť viacero symbolov v signále a Inspectrum vypočíta, akú modulačnú rýchlosť mal prijatý signál, viď. obrázok 4.2. Zistili sme že kľúčenka komunikuje s automobilom rýchlosťou 2,70 kBd.

Ďalším krokom bola samotná demodulácia signálu na jednotlivé bity. Inspectrum má aj na toto nástroj. Po označení požadovaného rozsahu signálu kurzorom môžeme pridať pod zaznamenaný signál amplitúdový priebeh signálu („Druhý klik > Add derived plot > Add sample plot“), viď obr. 4.3 a následne z neho extrahovať pole číslených hodnôt reprezentujúcich jednotlivé symboly („Druhý klik > Extract symbols > Copy to clipboard“). Úkážka týchto hodnôt je na výpise 4.1.




Obr. 4.1: Kód kľúčenky zobrazený v programe Inspectrum

Výpis 4.1: Ukážka extrahovaných symbolov z programu Inspectrum

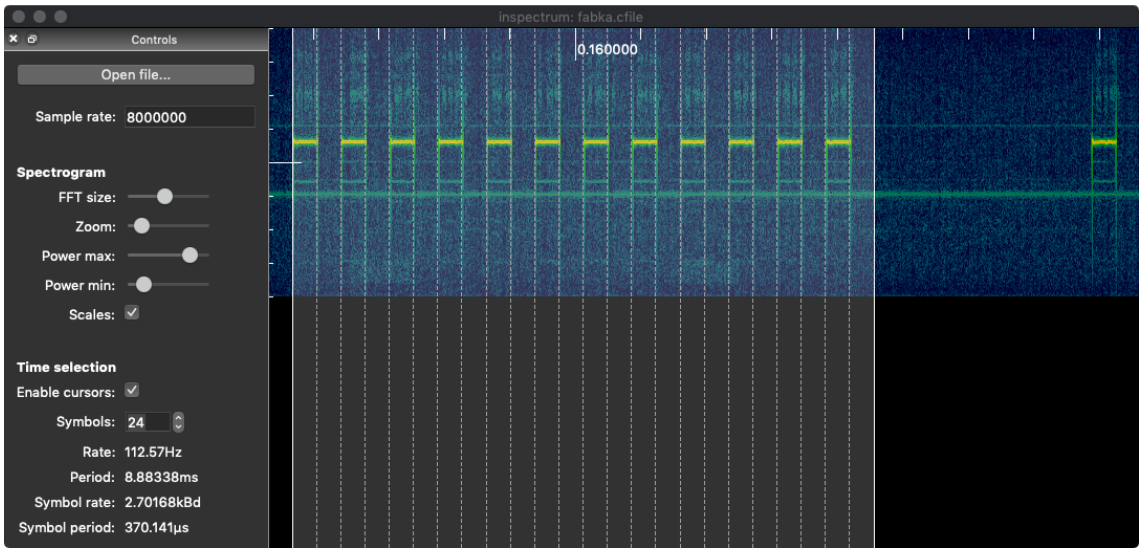
```
15.9435, -0.999957, 15.8023, -0.991139, 15.059, -0.999298,
15.1458, -0.997509, 16.1663, -0.979753, 15.286, -0.976355, ...
```

Tieto hodnoty sa potom dajú ľahko spracovať. Vidíme že logické 1 majú hodnotú väčšiu ako 0. Na extrahovanie bitovej sekvencie z tohto poľa hodnôt sme si napísali ľahký skript v programovacom jazyku Python. Na výpise 4.2 vidíme tento skript s orezanou vstupnou sekvenciou.

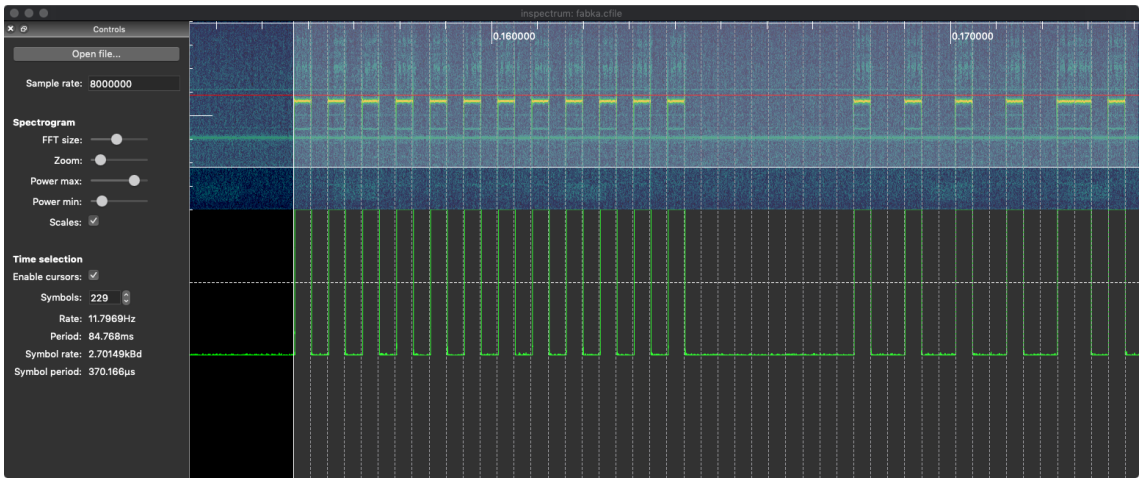
Výpis 4.2: Python skript extrahujúci bitovú sekvenciu ASK signálu

```
values = [
    -0.996353, 15.9541, -0.998837, -0.984955,
    15.6251, 13.9069, -0.998868, 14.1764
]
sequence = ''

for val in values:
    if val > 0:
        sequence += '1'
    else:
        sequence += '0'

print sequence
```

Tento skript vypíše svoj výstup na štandardný výstup. Výstup z vyššie zmieneného skriptu je takýto: `01001101`.




Obr. 4.2: Výpočet modulačnej rýchlosti v programe Inspectrum




Obr. 4.3: Odvodený amplitúdový graf signálu v programe Inspectrum

Skript sme aplikovali na viacero zaznamenaných kódov, z čoho sme vyvodili nasledujúce poznatky:

- Začiatok správy tvorila preambula o dĺžke 3 bajty, ktorú nasledoval 1 bajt vyplnený nulami.
- Keďže nepárny počet bajtov v preambule je neštandardný, môžeme za preambulu považovať len prvé dva bajty a ostatné dva za synchronizačné slovo
- Za synchronizačným slovom nasledovalo 200 dátových bitov
- Približne polovicu dát tvoril stále meniaci sa kód, čo môžeme považovať za techniku s postupným autorizačným kódom (Rolling Code)
- Druhá polovica dát sa nemenila, čo pravdepodobne znamená, že ide o unikátny identifikátor vozidla

## 4.2 Nastavenie CC1101

Z analýzy sme sa dozvedeli čo potrebujeme nakonfigurovať pre správne „naladenie“ CC1101 čipov tak, aby dokázali komunikovať s kľúčenkou a automobilom. Nastavili sme ASK moduláciu a základnú frekvenciu 433 MHz. Odstup jednotlivých kanálov je nastavený na 50 kHz. Keďže kľúčenka vysiela v rozmedzí 433,6 - 433,8 MHz, nastavili sme kanál číslo 12, ktorý odpovedá frekvencii 433,60 MHz. Symbolovú rýchlosť sme nastavili na 2,7 kBd.

Dĺžku preambuly sme nakonfigurovali na 2 bajty a zapli sme detekciu synchronizačného slova vo formáte `10101010 00000000` čo podľa zaznamenaného signálu z kľúčenky odpovedá posledným dvom bajtom pred dátovou časťou. Ďalej sme nakonfigurovali GPIO pin GDO2 tak, aby zdvihol svoju hodnotu na logickú 1 hneď vtedy, keď čip prijíme synchronizačné slovo. Pomocou toho budeme vedieť, kedy začať s rušením signálu.

Dĺžku paketu sme nastavili na fixnú hodnotu 25 bajtov (200 bitov).

Týmto spôsobom sa nám podarilo odchytiť kódy priamo v reálnom čase pomocou mikrokontroléra. Na výpise 4.3 je niekoľko prijatých kódov v hexadecimálnom formáte, kde sme medzerou oddelili časť kódu ktorá sa opakuje.

Výpis 4.3: Kódy kľúčenky zachytené pomocou CC1101

```
1269A49B49A4934DB4DB49A6D2 6D34D349B4DB6D36DB6DA6DA
1369249A4D34D349B4DB6D34D2 6D34D349B4DB6D36DB6DA6DA
124D269269B4926DB49A6DB692 6D34D349B4DB6D36DB6DA6DA
1B6D26D26D34D24DB6934D26D2 6D34D349B4DB6D36DB6DA6DA
```

## 4.3 Program

K realizácii útoku sme použili dva mikrokontroléry s CC1101 čipmi. Na prvom mikrokontroléri beží program, ktorý detekuje výskyt signálu kľúčenky. Volajme ho detektor. Detekujeme konkrétne, či sa obsah jeho RX fronty po prijatí kódu končí nemennou sekvenciou `6D34D349B4DB6D36DB6DA6DA`, pretože sme pri testovaní zistili, že kľúčenka niekedy pri krátkom stlačení tlačidla nestihne odoslať celý autorizačný kód a obsah RX fronty môže byť neúplný.

Druhý mikrokontrolér slúži ako rušička signálu – je pripojený k prvému mikrokontroléru signálnou linkou, ktorou mu signalizuje, kedy má začať rušiť signál. Rušička signálu operuje na vedľajšom kanále (kanál 14 – 433,70 MHz) a pri rušení vlastne len vysiela opakovane sekvenciu jednotiek a núl. Keďže oba CC1101 čipy majú frekvenčný filter oveľa selektívnejší ako filter automobilu, nemali by sa pri svojej operácii rušiť.

V momente, keď prvý kontrolér detekuje prichádzajúci signál z kľúčenky (synchronizačné slovo), pošle po káblovej linke signál rušičke, ktorá začne na vedľajšom kanále rušiť signál automobilu, čím mu znemožní úspešný príjem autorizačného kódu. Medzitým detektor prijíme prvý kód a uloží si ho do zásobníka. Po rovnakom prijatí a uložení druhého kódu, odošleme automobilu ten prvý. K detektoru sme pripojili LED diódu, ktorou signalizujeme, kedy je pripravený odoslať „ukradnutý“ kód automobilu. Taktiež sme pripojili mikrospínač, ktorým vieme kód odoslať.

Na otestovanie útoku sme mali len obmedzený čas (keďže autor práce nie je vlastník spomínaného automobilu) a dospeli sme k nasledujúcim záverom. Počas hodinového testovania sme zistili, že rušička a detektor nemôžu operovať na veľmi blízkych kanáloch, z dôvodu vzájomného rušenia. Ideálny bol odstup dva kanály. Prvotný nápad na riešenie tohto problému bol zmenšiť šírku pásma vstupného signálneho filtra, no podľa dokumentácie CC1101 je najmenšia možná šírka 58 kHz, ktorú sme aj používali). Testovanie sťažilo to, že automobil má už svoj vek a mal problémy s prijímaním signálu kľúčenky, pretože aj bez rušenia signálu nie vždy prijal kód správne a neodomkol sa. Kľúčenka bola taktiež vadná a buď po stlačení tlačidla signál vôbec nevysielala, alebo nestihla vyslať celý kód (rovnaké problémy popísal aj majiteľ vozidla). Aj napriek tomu sme dosiahli približne polovičnú úspešnosť v prijatí kódu a následnom preposlaní kódu automobilu po stlačení tlačidla na detektore, čo môžeme považovať za úspešný útok.

# 5 Záver

Táto práca sa zaoberala bezdrôtovými bezkľúčovými systémami. Rozobrali sme jednotlivé kategórie týchto systémov a spracovali sme najčastejšie typy útokov na tieto systémy.

Potom sme si ukázali rôzne nástroje použiteľné na implementáciu bezdrôtového bezkľúčového systému a analyzovali sme signál z reálnej kľúčenky.

Cieľom tejto práce bolo navrhnúť a implementovať bezdrôtový bezkľúčový systém, preto sme najprv navrhli útok, ktorým sme chceli poukázať na slabiny súčasných systémov použitých v automobilovom priemysle. Potom sme navrhli možné vylepšenia terajších bezkľúčových sytémov a nový bezkľúčový systém, ktorý je bezpečnejší ako tie terajšie.

V praktickej časti sme navrhnutý systém realizovali pomocou AVR mikrokontrolérov a rádiových modulov Texas Instruments CC1101. Náš systém vie pracovať v aktívnom, aj pasívnom režime a používa na zabezpečenie hardvérový generátor náhodných autorizačných kódov v kombinácii s AES šifrovaním so 128 bitov dlhým autorizačným kódom.

V neposlednom rade sme aj úspešne realizovali útok na terajšie automobilové bezkľúčové systémy, ktorým sme poukázali na ich slabé miesta.

# Literatúra


[1] Remote keyless system. Wikipedia [online]. [cit. 2018-11-08].
Dostupné z URL:
<https://en.wikipedia.org/wiki/Remote_keyless_system#History>

[2] ALRABADY, A.I. a S.M. MAHMUD. Analysis of Attacks Against the Security of Keyless-Entry Systems for Vehicles and Suggestions for Improved Designs. *IEEE Transactions on Vehicular Technology* [online]. 2005, 54(1), 41-50 [cit. 2018-11-23]. DOI: 10.1109/TVT.2004.838829. ISSN 0018-9545.
Dostupné z URL:
<http://ieeexplore.ieee.org/document/1386610/>

[3] K. Marneweck, *An Introduction to Keeloq Code Hopping*, AZ, Chandler:TB003 Applicat. Notes, Microchip Technol., Inc., 1996.

[4] J. Gordon, U. Kaiser, T. Sabetti, *A low cost transponder for high security vehicle immobilizers*, Proc. ISATA'96, 1996.

[5] ALRABADY, A.I. a S.M. MAHMUD. Some attacks against vehicles' passive entry security systems and their solutions. *IEEE Transactions on Vehicular Technology* [online]. 2003, 52(2), 431-439 [cit. 2018-11-28]. DOI: 10.1109/TVT.2003.808759. ISSN 0018-9545.
Dostupné z URL:
<http://ieeexplore.ieee.org/document/1198589/>

[6] A hacker made a $30 gadget that can unlock many cars that have keyless entry. *Business Insider* [online]. 2015, Aug. 6, 2015 [cit. 2018-11-28].
Dostupné z URL: <https://www.businessinsider.com/samy-kamkar-keyless-entry-car-hack-2015-8>

[7] CC1101: Low-Power Sub-1 GHz RF Transceiver. *Texas Instruments* [online]. 2013 [cit. 2019-05-03].
Dostupné z URL:
<http://www.ti.com/lit/ds/symlink/cc1101.pdf>

[8] HackRF One. *Great Scott Gadgets* [online]. [cit. 2018-12-09].
Dostupné z URL:
<https://greatscottgadgets.com/hackrf/>

[9] About RTL-SDR. *RTL-SDR.COM* [online]. [cit. 2018-12-09].
Dostupné z URL:
<https://www.rtl-sdr.com/about-rtl-sdr/>

[10] USRP1. *Ettus Research* [online]. [cit. 2018-12-06]. Dostupné z: <https://www.ettus.com/product/details/USRPPKG>

[11] Randomness extractor. *Wikipedia* [online]. 16 March 2019 [cit. 2019-05-06]. Dostupné z URL: <https://en.wikipedia.org/wiki/Randomness_extractor>

# Zoznam symbolov, veličín a skratiek

| | |
|---|---|
| **2-FSK** | Binary Frequency Shift Keying |
| **4-FSK** | Quaternary Frequency Shift Keying |
| **ASK** | Amplitude Shift Keying |
| **CID** | Customer identification device |
| **CRC** | Cyclic Redundancy Check |
| **CSPRNG** | Cryptographically Secure Pseudo-Random Number Generator |
| **FIFO** | First In, First Out |
| **FPGA** | Field-programmable gate array |
| **GPIO** | General-purpose input/output |
| **IDE** | Integrated development environment |
| **NIST** | National Institute of Standards and Technology |
| **OOK** | On-Off Keying |
| **RAM** | Random Access Memory |
| **RF** | Radio Frequency |
| **RKS** | Remote Keyless System |
| **RSSI** | Received Signal Strength Indicator |
| **SDR** | Software Defined Radio |
| **SPI** | Serial Peripheral Interface |
| **UHD** | USRP Hardware Driver |
| **UHF** | Ultra high frequency |
| **USRP** | Universal Software Radio Peripheral |

# Zoznam príloh

**A Obsah priloženého CD** **60**

# A Obsah priloženého CD

Na priloženom CD nájdeme elektronickú verziu tejto práce, zdrojové kódy nášho bezkľúčového systému, zdrojové kódy útoku a výstup z testovania generátorov náhodných výziev. K správnej kompilácii zdrojových kódov je potrebné mať nainštalované PlatformIO Core, ktoré je možné získať na adrese <`https://platformio.org/install/cli`>. Ak chceme nahrať kód do zariadení, je potrebné upraviť sériové porty v konfiguračných súboroch `platformio.ini`.

Kompiláciu a nahranie do zariadení vykonáme príkazom `pio run -t upload` v pracovnom priečinku daného PlatformIO projektu.

```
/ ..... koreňový priečinok priloženého CD
├── praca.pdf ..... elektronická verzia záverečnej práce
└── sources
    ├── cc1101-attack ..... PlatformIO projekt RollJam útoku
    │   ├── platformio.ini . konfiguračný súbor PlatformIO pre implementáciu útoku
    │   ├── analysis
    │   │   ├── extract_bits.py ..... skript extrahujúci bitovú sekvenciu signálu
    │   │   ├── readme.txt
    │   │   └── fabka.cfile ..... I/Q súbor so zaznamenaným signálom kľúčenky
    │   ├── lib
    │   │   └── cc1101/
    │   └── src ..... zdrojové kódy detektora a rušičky signálu
    │       ├── detector/
    │       └── jammer/
    └── cc1101-car-fob ..... PlatformIO projekt bezkľúčového systému
        ├── platformio.ini ... konfiguračný súbor PlatformIO pre bezkľúčový systém
        ├── include
        │   ├── SystemMode.h
        │   ├── constants.h ..... hlavičkový konfiguračný súbor RKS
        │   └── states.h
        ├── generator_tests ..... testovacie výstupy generátorov náhodných výziev
        │   ├── analogread_seed_generator.txt
        │   └── analogread_von_neumann_generator.txt
        ├── lib
        │   ├── tiny-AES-c/
        │   ├── cc1101/
        │   └── packet/
        └── src ..... zdrojové kódy automobilu a kľúčenky
            ├── car/
            └── fob/
```