Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Bezpečnostná analýza vybraných hardvérových zariadení

Diplomová práca

2017 Bc. Tomáš Paulík

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Bezpečnostná analýza vybraných hardvérových zariadení

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	RNDr. Richard Ostertág PhD.

Bratislava, 2017 Bc. Tomáš Paulík

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Tomáš Paulík
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Bezpečnostná analýza vybraných hardvérových zariadení
Security analysis of hardware devices

Cieľ: Práca bude zameraná na posúdenie bezpečnosti operačnej jednotky UNISIEŤ. A to najmä z nasledovných pohľadov:

- aplikovateľnosť časového útoku, identifikovaného pri operačnej jednotke RAK DEK

- možnosť útoku pomocou pamäťového kľúča prepisujúceho internú databázu kľúčov

- možnosť klonovania kľúčov napriek kombinácii operačnej jednotky so zariadením antiklon

Vedúci: RNDr. Richard Ostertág, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 14.12.2015

Dátum schválenia: 16.12.2015 prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Ďakujem vedúcemu diplomovej práce RNDr. Richardovi Ostertágovi PhD. za odbornú pomoc pri jej vypracovávaní a pánovi Petrovi Grečkovi zo spoločnosti RYS za ochotnú spoluprácu.

Abstrakt

Práca sa venuje zariadeniam, ktoré slúžia na zvýšenie bezpečnosti obytných priestorov. Zamerali sme sa na zariadenia, ktoré využívajú dotykové elektronické kľúče (DEK) založené na iButtonoch. V práci uvádzame fungovanie 1-Wire protokolu, ktorý využívajú iButtony pri komunikácii s operačnými jednotkami. Najskôr popisujeme fungovanie základného typu iButtonu, ktorý sa využíva na automatickú identifikáciu a následne sa venujeme verzii, ktorá je rozšírená o pamäť. Prvým hardvérovým zariadením, s ktorým sme sa zaoberali je antiklon. Antiklon je schopný rozpoznať kópiu iButtonu a zablokovať jej komunikáciu s operačnou jednotkou. V práci ukazujeme slabinu zariadenia aj s konkrétnou implementáciou, ktorá zariadenie antiklon obchádza. V ďalšej časti práce sa venujeme operačnej jednotke UNISIEŤ. Preverovali sme jej zraniteľnosť na rôzne typy útokov. Zaujala nás funkcionalita, ktorá umožňuje prepisovanie databázy a aktualizáciu parametrov s využitím pamäťového iButtonu. Ukázali sme, že je možné získať potrebné heslo a s jeho využitím prepísať databázu identifikátorov operačnej jednotky.

Kľúčové slová: bezpečnosť, 1-Wire, Antiklon, UNISIEŤ

Abstract

In this thesis we study the security of physical security access control systems. Specifically we looked at the security of touch-based electronic keys (DEK) based on the iButton standard. In the first part we describe the 1-Wire protocol used by iButtons to communicate with their operating units. We describe unlocking a gate with a basic iButton key (which authenticates with its unique hardware ID) and then with an iButton key extended with memory. Then we research the security of anti-cloning hardware devices which are supposedly able to recognize a copy of a basic iButton key and block its communication with the operating unit. We show a weakness in a specific anti-cloning device and hardware implementation of an iButton copy, which can circumvent this anti-cloning device. In the next part of thesis we look at a more advanced operating unit called UNISIET. We have tested multiple attacks and found a vulnerability during key database update using a memory iButton. We are able to obtain a password and overwrite the key database.

Keywords: security, 1-Wire, Antiklon, UNISIEŤ

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Práca sa venuje hardvérovým zariadeniam, ktoré slúžia na zvýšenie bezpečnosti obytných priestorov. Zamerali sme sa na systémy využívajúce dotykové elektronické kľúče (DEK). Primárnym podnetom pre vznik týchto systémov bola snaha výrobcu systému zabrániť vstupu nepovolaným osobám do chráneného objektu. DEK, ktorému sa v práci venujeme je založený na iButton technológii. Technológia iButton využíva na komunikáciu protokol 1-Wire, ktorý opisujeme v prvej kapitole práce. Je nutné pochopiť štandardné príkazy a spôsob komunikácie, aby sme vedeli popísať analýzu nameraných dát alebo testovať zariadenia využívajúce tento protokol. iButton je tvorený čipom malej veľkosti. Má vysokú tepelnú a mechanickú odolnosť. Čip môže byť osadený v plastovom prívesku, ktorý je možné pripevniť ku kľúčenke. Každý čip by mal obsahovať globálne unikátne identifikačné číslo, ktoré slúži k presnej identifikácii majiteľa čipu. Podrobný popis iButtonov so zameraním na druhy, ktoré využívame v ďalšej časti práce je spracovaný v druhej kapitole. Systém DEK nahrádza používanie klasických kľúčov a kódových zámkov. Dotyková plocha a operačno-pamäťová jednotka sa inštalujú v blízkosti dverí s elektrickým zámkom. Pre vstup do objektu je nevyhnutný fyzický kontakt medzi dotykovým elektronickým kľúčom a dotykovou plochou. Výhodou DEK kľúčov je, že stratený alebo odcudzený DEK sa vyradí z databázy, čím sa znemožní jeho ďalšie používanie. Pokiaľ by ho majiteľ znovu našiel, je ho možné opätovne zaregistrovať a používať ho ako predtým. Rovnako praktické je aj to, že jeden DEK môže byť zaregistrovaný vo viacerých „dverách“, čo znamená, že s jedným DEK kľúčom je možné vstupovať do viacerých budov alebo miestností. To môže byť riešením pre vstup do budov pre doručovateľov, údržbárov, upratovačky a pod. Táto technológia už síce má mnohých nástupcov, ale naďalej sa v praxi vyskytuje vo veľkom množstve kvôli cenovej dostupnosti. Práve kvôli veľkému využívaniu začali na trhu vznikať kópie originálnych iButtonov. Kópie sú ešte lacnejšie ako originál a umožňujú prepisovať svoje identifikačné číslo. Tým vzniká niekoľko problémov. Jedným z nich je, že celá technológia iButtonov a nimi používaného 1-Wire protokolu zakladá na unikátnosti identifikačných čísel. Ďalším problémom je určitá anonymita používateľa. Nad systémom využívajúcim DEK sú postavené rôzne bezpečnostné projekty. Jedným z nich je projekt bezpečného bývania od spoločnosti RYS. Táto spoločnosť s nami ochotne spolupracovala a poskytla nám zariadenia, ktoré sme mohli v práci testovať. Projekt bezpečné bývanie sa reali-

zuje od roku 1999 a jeho základným cieľom je vytváranie komplexných riešení ochrany obyvateľov viacbytových domov. Zásadný problém, ktorý vytváranie klonov spôsobuje v takomto bezpečnostnom projekte, je znižovanie bezpečnosti v bytovom dome, nárast neoprávnených vstupov a zvyšovanie neprehľadnosti používaných kľúčov. To všetko postupne vedie k strate kontroly nad systémom. Kopírovanie elektronických kľúčov, rovnako ako kopírovanie bežných kľúčov nie je nelegálne. Preto v snahe zabrániť používaniu vytvorených klonov vzniklo zariadenie antiklon, ktoré je možné doplniť do existujúceho prístupového systému a zároveň je schopné rozpoznať kópiu DEK kľúča. V prípade, že antiklon deteguje okopírovaný DEK, tak neprepustí komunikáciu do operačnej jednotky. Jedným z cieľov našej práce je overiť spoľahlivosť antiklonu. Snažíme sa vytvoriť kópiu iButtonu, ktorú by zariadenie antiklon nevedelo rozpoznať.

V zvyšnej časti práce sa snažíme identifikovať možné zraniteľnosti operačnej jednotky UNISIEŤ. Táto operačná jednotka je bežne používaná v systémoch využívajúcich DEK, alebo bezkontaktných systémoch využívajúcich RFID technológiu. Operačná jednotka poskytuje navyše pokročilejšiu funkcionalitu ako RAK DEK analyzovaný v práci [8]. Spoločnosť RYS ponúka k operačnej jednotke UNISIEŤ softvér BBIQ. S využitím softvéru je možné pridávanie používateľov, ich zadeľovanie do skupín, pridávanie informácií k jednotlivým identifikačným kľúčom v databáze a mnoho ďalších funkcií. Veľkou výhodou je možnosť programovania prenosového média. Táto funkcionalita umožňuje aktualizáciu databázy alebo nastavení operačnej jednotky bez nutnosti online komunikácie s PC. Aktualizácia prebieha pomocou predprogramovaného pamäťového iButtonu, ktorý sa priloží k dotykovej ploche pripojenej k operačnej jednotke UNISIEŤ. Ako bezpečnostný prvok sa využíva heslo, ktoré musí byť nastavené v operačnej jednotke a priloženom iButtone. Cieľom našej práce je aj preveriť možnosť využitia pamäťového iButtonu na získanie hesla. To by umožnilo prepísanie databázy a zmenu nastavení v operačnej jednotke.

Kapitola 1

1-Wire protokol

Všetky zariadenia, s ktorými sme pracovali, využívajú na komunikáciu 1-Wire protokol. 1-Wire je sériový protokol, ktorý využíva na komunikáciu iba jednu dátovú zbernicu a zem. Komunikujúce zariadenia majú svoje role. Jedna strana vystupuje ako master a druhá ako slave. Master iniciuje a riadi komunikáciu s jedným alebo viacerými zariadeniami vystupujúcimi v úlohe slave pripojených na jednej spoločnej dátovej zbernici. Každý originálny 1-Wire slave by mal mať unikátne, nemenné, 64 bitov dlhé, identifikačné číslo, ktoré slúži pre adresáciu pri komunikácii. Skladá sa z 8-bitov identifikujúcich typ zariadenia (zvaných family code), 48-bitového sériového čísla a nakoniec 8-bitový kontrolný súčet (CRC). Bežné zariadenia v pozícii slave fungujú v napäťovom rozsahu 2,8V až 5,25V. Väčšina zariadení čerpá energiu paraziticky zo zbernice. 1-Wire protokol je určený na kontaktnú komunikáciu. Výhodou je, že zariadenia sa po odpojení alebo strate kontaktu vrátia do presne definovaného počiatočného stavu. Je nutné vedieť začať odznova pri zariadeniach, ktoré majú neistý kontakt. iButtony sú navrhnuté tak, aby fungovali aj pri slabej konektivite. Je však nutné zabezpečiť aspoň minimálny čas pre prenos príkazov a dát. Tento čas sa líši v závislosti od príkazu, ktorým sa venujeme neskôr. Podrobné informácie o 1-Wire protokole sú dostupné v oficiálnom štandarde od výrobcu [2].

1.1 Funkcionalita a využitie

V súčasnosti existuje zhruba 40 rôznych druhov 1-Wire zariadení. Všetky 1-Wire zariadenia sa dajú zadeliť do deviatich rôznych tried podľa podporovanej funkcionality. Všetky podporujú identifikáciu svojim unikátnym číslom a následne môžu byť rozšírené o pamäť, teplomer, šifrovanie pamäte a ďalšie možnosti. Podrobnejšie sa venujeme len rozšíreniu o pamäť, ktoré v našej práci využívame. Na autentifikáciu sa využíva základný typ, ktorý sa len identifikuje svojim sériovým číslom. Všetky tie, ktorých trieda začína s DS19 sú iButtony, ktorým sa podrobnejšie venujeme v ďalšej časti práce.

1.2 Komunikácia

Vo 1-Wire protokole sa všetky dáta prenášajú asynchrónne po bitoch cez half-duplex. To znamená, že v jeden čas je možné len vysielať alebo prijímať. Všetku komunikáciu iniciuje zariadenie v pozícii master. Na synchronizáciu pri posielaní bitu (oboma smermi) sa využíva spádová hrana v napätí. Celá komunikácia je riadená v časových úsekoch a dá sa definovať štyrmi funkciami:

- reset pulz
- presence pulz
- zápis do časového úseku
- čítanie z časového úseku

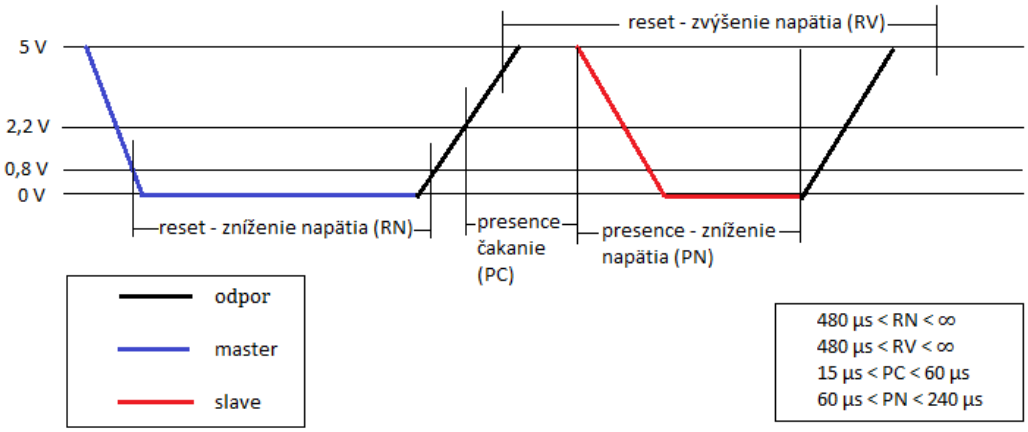
Vo všetkých funkciách je dôležitý pokles napätia na zbernici a jeho trvanie. A práve trvanie uvádzané v štandardoch bolo kľúčovým pre analýzu nameraných dát. Pre najväčšiu toleranciu sa zvykne testovať hodnota na dátovej zbernici v strede časového úseku. Aktívna časť časového úseku pre iButton má dĺžku 60 μs. Čiže iButton sleduje zbernicu 30 μs po zaznamenaní spádovej hrany. Počas aktívnej časti časového úseku je nutné, aby napätie zostalo buď pod 0,8 V (minimálnou hodnotou, ktorá sa považuje za logickú nulu) alebo malo hodnotu väčšiu ako 2,2 V (minimum potrebné pre vyhodnotenie logickej jednotky). Po každom časovom úseku je potrebný čas aspoň 1 μs ako príprava pre nasledujúci bit. Tento čas sa považuje za neaktívnu časť časového úseku aj keď je nutné ho počítať spolu s aktívnou časťou do trvania časového úseku. Vo zvyšnej časti práce budú úlohu slave zariadení vždy zastávať iButtony.

1.2.1 Reset pulz

Každá komunikácia začína tým, že master pošle reset pulz. To v praxi znamená, že sa zníži napätie po dobu aspoň 8 časových úsekov (480 μs). Následne zdvihne napätie a počká ďalších 480 μs. Počas toho sa očakáva, že ak sa na dátovej zbernici nachádza iButton, tak zareaguje poslaním presence pulzu. Počas tohoto čakania nie je povolená žiadna iná komunikácia. Reset pulz zabezpečuje čistý štart komunikácie, ktorý nahrádza ľubovoľnú časovú synchronizáciu.

1.2.2 Presence pulz

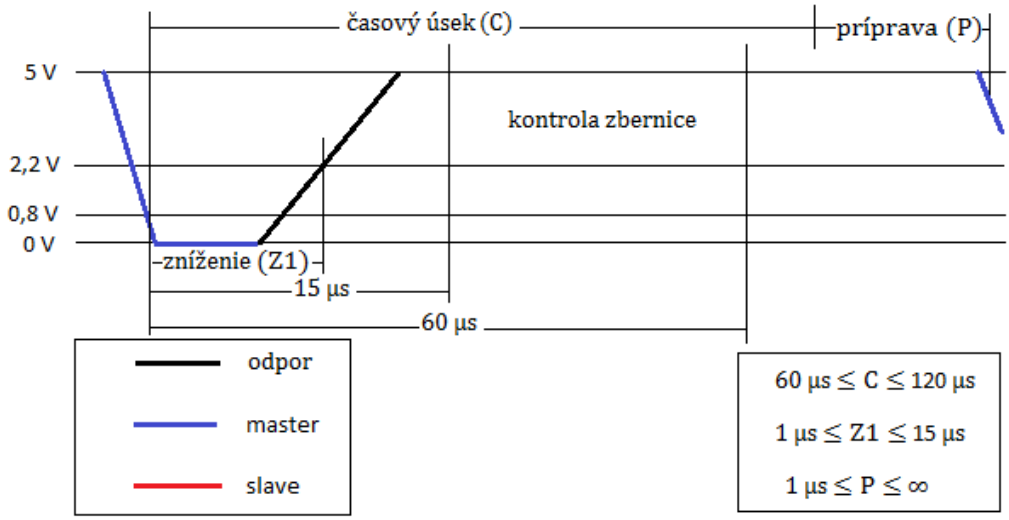
Keď iButton zaznamená reset pulz, je nutné aby počkal 15 μs až 60 μs a následne znížil napätie na nulu po dobu 60 μs až 240 μs. Tým dá masterovi jednoznačne vedieť, že sa nachádza na dátovej zbernici a očakáva príkaz, ktorý má vykonať. Priebeh komunikácie reset a presence pulzu je vidieť na obrázku 1.1.



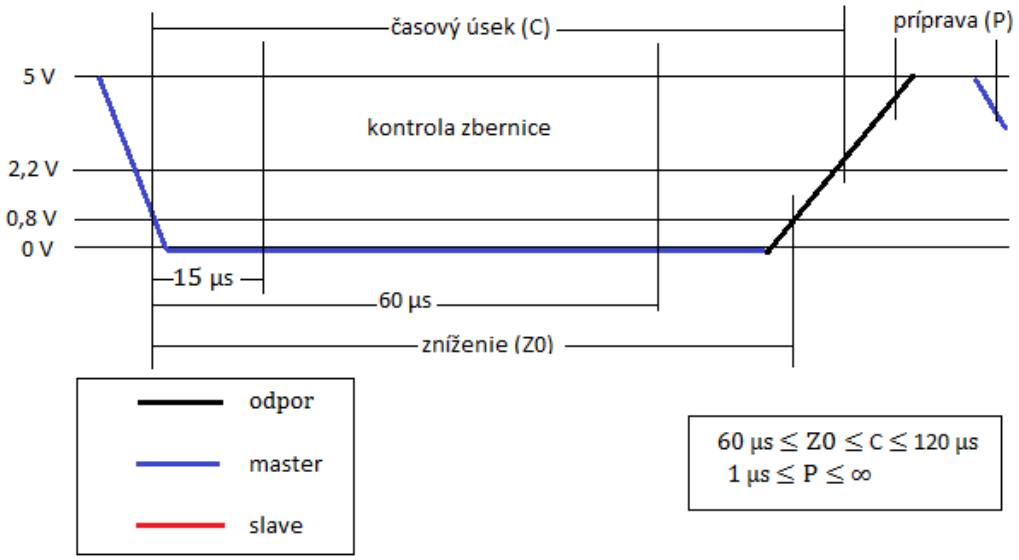
Obr. 1.1: Reset a presence pulz

1.2.3 Zápis do časového úseku

Posielanie bitov sa realizuje posielaním časových úsekov, ktoré sa vyhodnotia na logickú nulu alebo jednotku. Pre poslanie logickej jednotky je nutné aby bolo napätie na zbernici znížené po dobru kratšiu ako 15 μs. Pre poslanie logickej nuly je naopak potrebné držať napätie znížené po dobu aspoň 60 μs. Trvanie aktívnej časti časového úseku sa môže aj predĺžiť, ale nesmie presiahnuť trvanie ôsmich časových úsekov, ktoré predstavuje reset pulz. Zápis logickej jednotky do časového rámca je na obrázku 1.2 a zápis logickej nuly na obrázku 1.3.



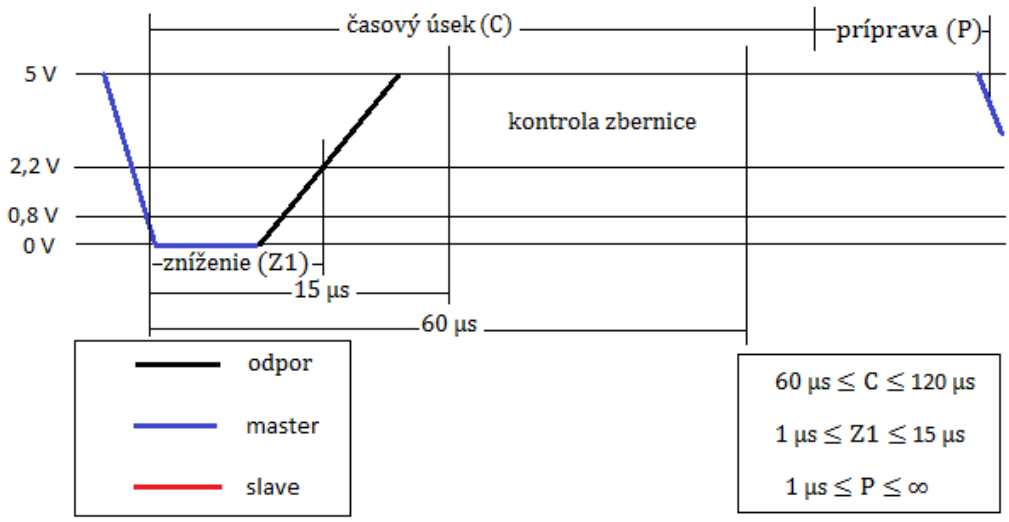
Obr. 1.2: Zápis logickej jednotky do časového úseku



Obr. 1.3: Zápis logickej nuly do časového úseku

1.2.4 Čítanie z časového úseku

Pre čítanie dát musí master vygenerovať časový úsek slúžiaci na čítanie, aby definoval začiatok posielania každého bitu. Z pohľadu mastera vyzerá priebeh čítania rovnako ako zápis logickej jednotky. Ak iButton potrebuje poslať bit s hodnotou jedna, tak len nechá zbernicu nedotknutú. Tá po spádovej hrane určujúcej začiatok časového úseku sama stúpne späť (vďaka pull-up rezistoru) na napäťové maximum a tým vygeneruje stav zodpovedajúci posielaniu logickej jednotky. Ak iButton potrebuje poslať bit s hodnotou nula, podrží napätie znížené po dobu aspoň 15 μs. Master musí po spádovej hrane pridržať napätie znížené aspoň 1 μs. Snaží sa však držať napätie znížené čo najkratšie, aby mohol čo najskôr sledovať hodnotu posielanú na zbernici. Čas sledovania sa snaží čo najviac priblížiť k 15 μs. Po prečítaní posielanej hodnoty je potrebné ešte nechať určitý čas, ktorý iButton využíva na uvoľnenie dátovej zbernice, aby sa napätie vrátilo späť na maximum. Táto doba sa môže líšiť od 0 μs po 45 μs. Bežná hodnota je však 15 μs. Priebeh časového úseku určeného na čítanie je vidieť na obrázku 1.4.



Obr. 1.4: Čítanie časového úseku

1.3 Príkazy

Ako sme už spomenuli, komunikácia vždy začína poslaním reset pulzu od mastera, pokračuje odpoveďou iButtonu v podobe presence pulzu a následne iButton očakáva príkaz, ktorý má vykonať. Všetky iButtony podporujú štyri príkazy, ktorých kód v hexadecimálne sústave uvádzame v zátvorke:

- read ROM (0x33)
- skip ROM (0xCC)
- match ROM (0x55)
- search ROM (0xF0)

Jednotlivé triedy iButtonov podporujú navyše špecifické príkazy pre prístup k pamäti, jej úpravu a mnohé ďalšie. Zoznam všetkých oficiálne podporovaných príkazov pre jednotlivé triedy iButtonov je dostupný na `http://owfs.sourceforge.net/family.html`. Podrobnejšie si popíšeme príkazy, ktoré podporujú všetky iButtony a zvlášť sa budeme venovať len tým, ktoré zohrali dôležitú úlohu pri našej analýze dát. Na konci kapitoly je vidieť prechody cez všetky ROM príkazy iButtonov.

1.3.1 read ROM

Základný príkaz, ktorý umožňuje masterovi prečítať postupne 8-bitový family code, 48-bitové sériové číslo a nakoniec 8-bitový kontrolný súčet. Tento príkaz je použiteľný iba ak sa na dátovej zbernici nachádza len jeden iButton. Ak by ich bolo viac, tak

posielané bity by kolidovali a výsledok by vyzeral ako ich konjunkcia. Výsledok takejto kolízie by nemal mať platný kontrolný súčet a preto sa vyhodnotí čítanie za chybné.

1.3.2 skip ROM

Slúži pre rýchlejší prístup k pamäti. Ak si master vie byť istý, že na zbernici je len jedno zariadenie, tak tento príkaz umožňuje preskočiť čítanie identifikačného čísla a rovno posielať príkazy na prácu s pamäťou.

1.3.3 match ROM

Umožňuje adresáciu konkrétneho zariadenia na zbernici. Keďže oficiálne by nemali existovať dva iButtony s rovnakým identifikačným číslom, tak je takáto adresácia jednoznačná a ostatné iButtony na zbernici sa do komunikácie nezapájajú, až kým nenastane nový reset pulz.

1.3.4 search ROM

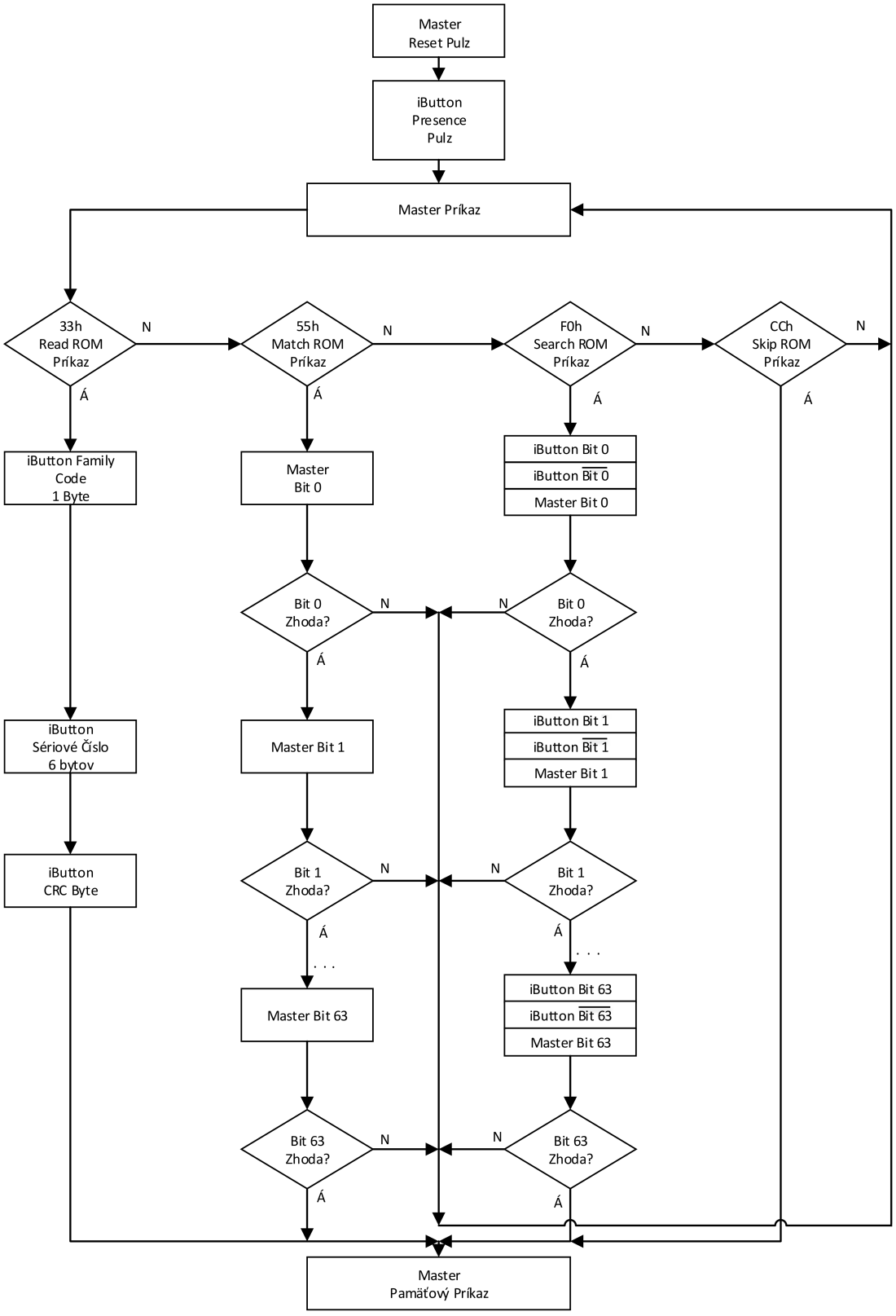
Ak sa na zbernici nachádza viacero zariadení a master nepozná ich identifikačné čísla, tak pomocou search ROM algoritmu vie nájsť a adresovať jedno konkrétne zariadenie. Search ROM algoritmus je v podstate prehľadávanie do hĺbky, v ktorom chodíme po vetvách kým nenájdeme adresu pripojeného zariadenia. Ďalšie prehľadávania idú po zvyšných vetvách, kým neprejdú všetky listy.

Po poslaní search ROM príkazu sa prehľadávanie začína tým, že všetky pripojené zariadenia pošlú prvý bit (začína sa od najmenej významných rovnako ako pri čítaní) ich identifikačného čísla. Vzhľadom ku charakteristike 1-Wire sa toto posielanie prejaví ako konjunkcia všetkých prvých bitov, ktoré zariadenia poslali. Keď zariadenia pošlú svoj prvý bit, master začne nový časový úsek, v ktorom majú všetky zariadenia poslať negáciu svojho prvého bitu. Z týchto dvoch prečítaných bitov vie master určiť, aké zariadenia sa nachádzajú na zbernici. Všetky možnosti sú popísané v tabuľke 4.4.1.

Tabuľka 1.1: Vyhodnocovanie odpovedí v search ROM algoritme.

Pôvodný bit	Negovaný bit	Získaná informácia
0	0	Zariadenia majú hodnotu bitu 1 aj 0
0	1	Všetky zariadenia majú daný bit 0
1	0	Všetky zariadenia majú daný bit 1
1	1	Na zbernici nie sú žiadne komunikujúce zariadenia

Po obdržaní dvoch bitov master musí poslať jeden bit späť všetkým zariadeniam. Ak zúčastnené zariadenie má bit s hodnotou, akú poslal master, tak sa naďalej zúčastňuje komunikácie. Ak nie, tak zostane nečinné až do ďalšieho reset pulzu. Tento postup prečítania dvoch bitov a poslanie jedného sa opakuje so zostávajúcimi 63-mi bitmi identifikačného čísla. Tým algoritmus prinúti všetky zariadenia okrem jedného aby zostali nečinné. Ak sa na zbernici nachádzali len zariadenia s rovnakou hodnotou bitu, tak algoritmus pokračuje vetvou daného bitu. Ak sa nachádzajú obe hodnoty, tak v prvom behu sa algoritmus rozhodne pre vetvu s hodnotou 0 (alebo 1 je to vec implementácie) a zapamätá si túto hodnotu pre ďalšie opakovania. Algoritmus si môže zvlášť evidovať prvých osem bitov určujúcich typ zariadení, aby ich mohol zadeliť do skupín a prípadne ďalej vynechať celú triedu naraz. Rovnako sa overuje platnosť kontrolného súčtu aby boli vybraté len validné adresy. Celý proces zistenia a adresovania konkrétneho zariadenia trvá zhruba tri krát dĺžku čítania identifikačného čísla, ale umožňuje sekvenčné vyberanie pripojených zariadení bez znalosti ich adries.



Obr. 1.5: Prechod iButtonov ROM príkazmi

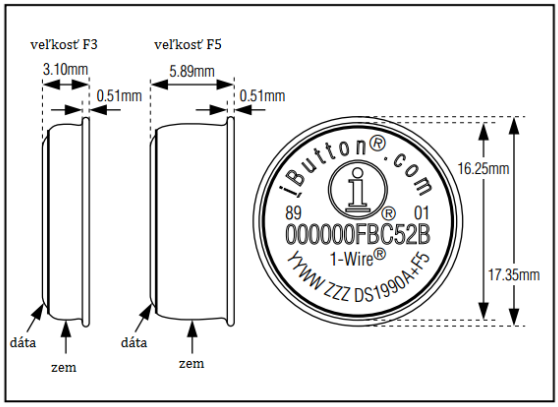
Kapitola 2

iButton

iButton je počítačový čip vložený do 16 *mm* hrubej nehrdzavejúcej ocele. Vďaka rozmerom je možné ho osadiť do bežných predmetov, napríklad kľúčenky alebo prsteňu. Pevný obal zaručuje odolnosť a možnosť použitia aj v nepriaznivých podmienkach. Zariadenie je testované na 10 ročnú výdrž. Je použiteľný denne na rôzne účely, ako sú kontrolovaný prístup do budovy alebo k počítaču, ukladanie dát, aktualizovanie databázy či meranie teploty.

2.1 Komponenty iButtonu

Zariadenie iButton využíva svoj obal ako elektronické komunikačné rozhranie. Každý obal sa skladá z dátového kontaktu a uzemnenia. Oba kontakty sú spojené so silikónovým čipom vnútri. Dátový kontakt je na vrchu (predná časť s laserom výpaleným sériovým číslom) a uzemňovací kontakt tvorí boky obalu. Pre jednoduchšie osadenie iButtonu uzemnenie tvorí lem okolo iButtonu. Kontakty sú od seba navzájom oddelené pomocou polypropylénovej priehradky. Jednoduchým dotykom dvoch kontaktov vie iButton komunikovať. Na komunikáciu využíva 1-Wire protokol, ktorý sme popísali v predchádzajúcej časti práce. Každý originálny iButton má vlastnú, globálne unikátnu nemennú adresu, ktorú má laserom vypálenú priamo na obale a uloženú v čipe. Túto adresu je možné použiť ako kľúč alebo identifikátor daného zariadenia.



Obr. 2.1: iButton a jeho rozmery

2.2 Typy iButtonov

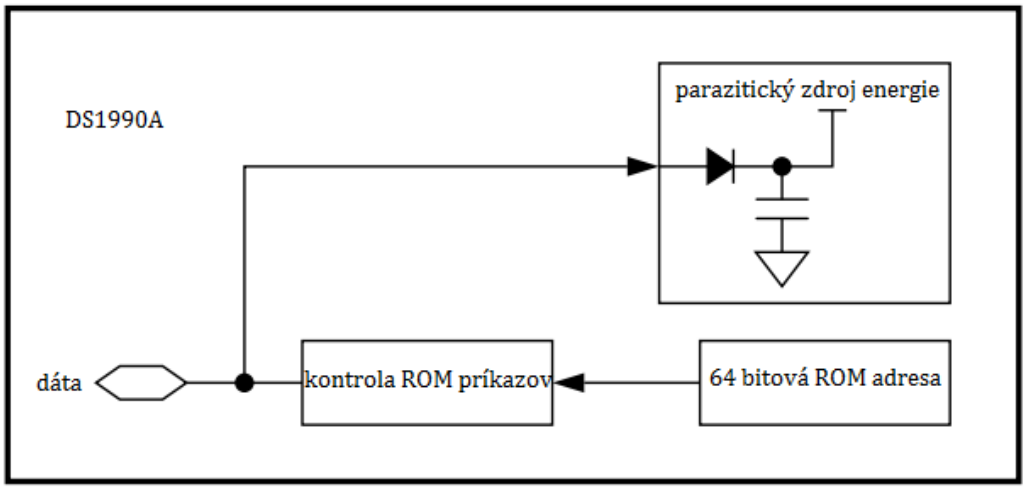
Rad produktov iButton zahŕňa viac ako 20 odlišných typov s rôznou funkcionalitou pridanou nad základný iButton. Produkty sa dajú zadeliť podľa funkcií do nasledovných kategórii:

- iba adresa
- s pamäťou
- s reálnym časom
- bezpečnostné
- ukladanie logov

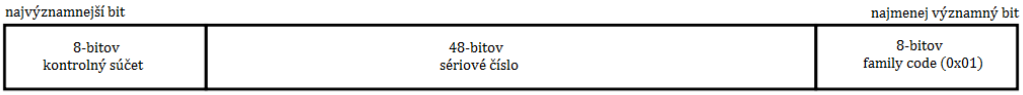
V našej práci sa podrobnejšie venujeme typu, ktorý využíva iba svoju adresu ako kľúč a typu s pamäťou, ktorý umožňuje aktualizáciu databázy operačnej jednotky UNISIEŤ.

2.3 iButton s adresou (DS1990)

Trieda DS1990A iButtonov slúži výhradne ako elektronické registračné číslo pre automatickú identifikáciu. Vyhotovuje sa v dvoch prevedeniach F3 a F5, ktoré líšia sa šírkou obalu ako je vidieť na obrázku 2.1. Všetky dáta sa prenášajú 1-Wire protokolom po jedinej dátovej zbernici. Každý originálny iButton triedy DS1990A má jedinečné 64-bitov dlhé registračné číslo. Využíva sa prevažne v prístupových systémoch, na sledovanie postupu prác alebo kontrolu inventárov. Výhodou je, že celá komunikácia trvá kratšie ako 5 ms. Ako je vidieť na obrázku 2.2, DS1990 berie energiu paraziticky cez dátový kontakt. Funkčná jednotka ROM obsahuje 1-Wire rozhranie a logiku potrebnú



Obr. 2.2: Schéma iButtonu

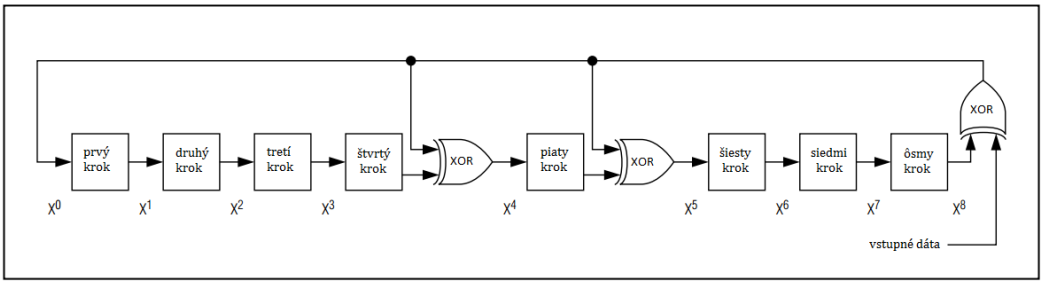


Obr. 2.3: Sériové číslo iButtonu

pre zvládanie príkazov, ktoré pristupujú k 64 bitovej adrese. Podrobnejšie informácie o triede DS1990A sú dostupné v technickom liste (datasheet) [3].

2.3.1 64 bitová adresa

Prvých 8 bitov tvorí takzvaný 1-Wire family code, ktorý určuje o aký typ iButtonu sa jedná. Ako vidieť aj na obrázku 2.3, tak Trieda DS1990 má family code s hexadecimálnou hodnotou 0x01. Nasledujúcich 48 bitov predstavuje unikátne sériové číslo. Posledných 8 bitov tvorí kontrolný súčet predchádzajúcich 56-tich bitov. Kontrolný súčet sa počíta cez lineárny register so spätnou väzbou. Generujúcim polynómom registra je $x^8 + x^5 + x^4 + 1$. Bity registra sú inicializované nulami. Proces generovania kontrolného súčtu začína od najmenej významných bitov family code a následne postupuje cez 48 bitové sériové číslo. Podrobný popis generovania kontrolného súčtu aj s príkladom je uvedený na[5]. Lineárny register je vidieť na obrázku 2.4.



Obr. 2.4: Lineárny register kontrolného súčtu

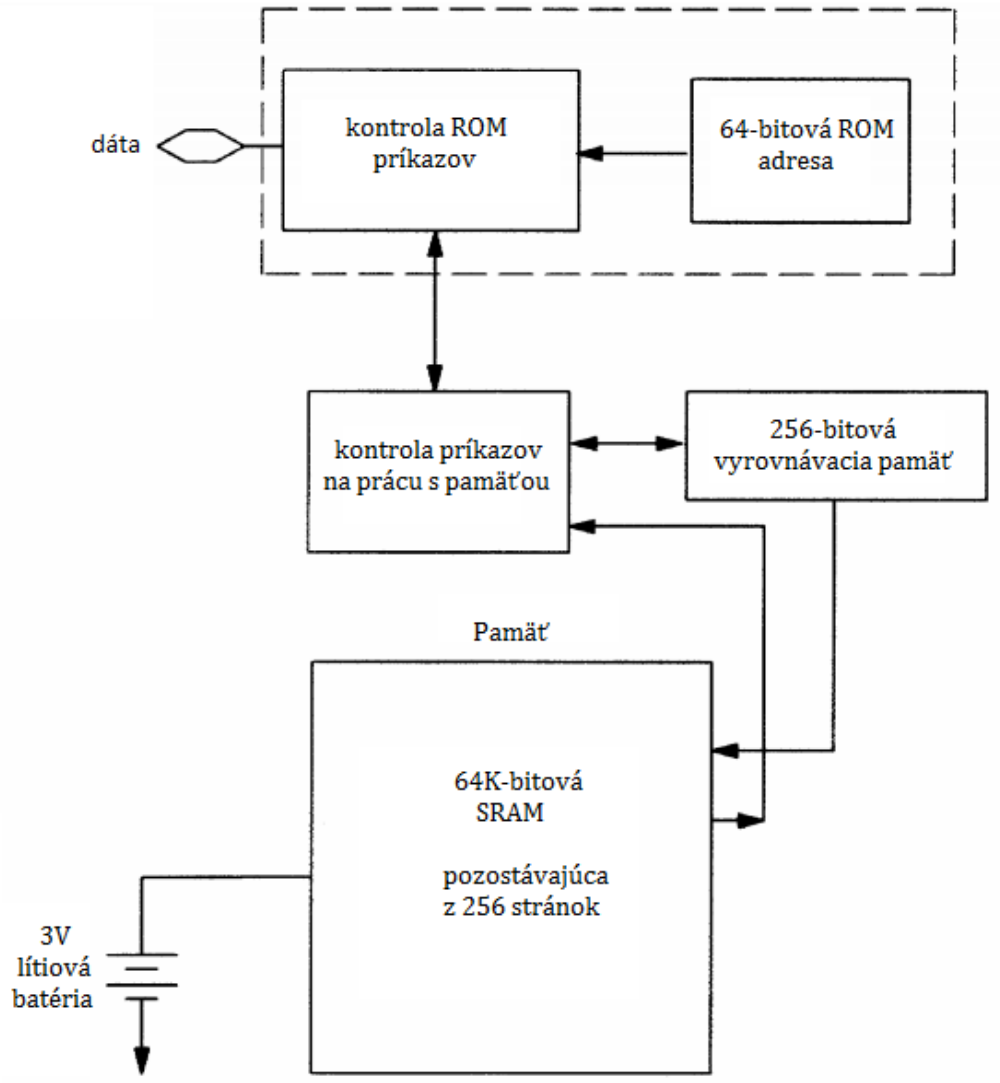
2.3.2 Podporované príkazy

Každá typ iButtonov podporuje nejakú podmnožinu príkazov. Všetky príkazy sú 8 bitov dlhé a základné z nich sme popísali v kapitole venovanej 1-Wire protokolu. Trieda DS1990A podporuje len základné príkazy Read ROM (0x33), Search ROM (0xF0), Match ROM (0x55) a Skip ROM (0xCC).

2.4 iButton s pamäťou (DS1996)

Trieda DS1996 sú pamäťové iButtony, ktoré slúžia na čítanie alebo zápis dát. Energeticky nezávislá pamäť poskytuje jednoduché riešenie pre ukladanie a získavanie podstatných informácií vzťahujúcich sa k zariadeniam, ku ktorým sa iButton pripojí. Dáta sú posielané sériovo cez 1-Wire protokol. Okrem toho obsahujú stránku pamäte, ktorá slúži výlučne ako vyrovnávacia pamäť (scratchpad) pri zápise. Dáta sa najskôr zapíšu do vyrovnávacej pamäte a po kontrole sa prekopírujú do skutočnej pamäte zariadenia. Tento proces zaručuje integritu dát pri modifikácii pamäte. Rozmery a vyhotovenie iButtonu poskytujú vysokú odolnosť a ľahkú prepravu. Používa sa napríklad na kalibráciu zariadení alebo úpravu databázy prístupových systémov. Na obrazku 2.5 je diagram popisujúci vzťahy medzi časťou pre interpretáciu ROM príkazov a časťou pre spracovanie príkazov na prácu s internou SRAM pamäťou. Trieda DS1996 sa skladá z troch hlavných komponentov:

- 64-bitová ROM adresa
- 256-bitová vyrovnávacia pamäť
- 65536-bitová SRAM



Obr. 2.5: Diagram popisujúci vzťahy hlavných komponentov DS1996 iButtonov

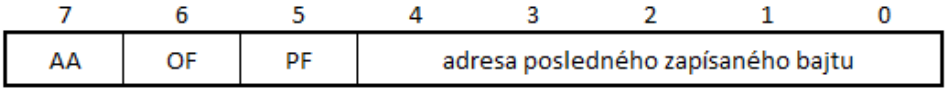
Trieda DS1996 tiež paraziticky čerpá energiu zo zbernice. Takto získavaná energia má v tomto prípade dve využitia:

- vykonávanie ROM príkazov (tým nevyčerpáva vlastnú batériu)
- umožňuje používanie ROM príkazov aj v prípade, že batéria je vybitá

Zvyšné obvody čerpajú energiu výlučne z vlastnej batérie. Sériové číslo je uložené rovnako ako pre triedu s DS1990. Líšia sa iba v hodnote family code, ktorý má pre triedu DS1996 hodnotu 0x0C. Kontrolný súčet sa počíta rovnakým lineárnych registrom ako pri DS1990 (obrázok 2.4).

2.4.1 Adresové registre a status prenosu

Kvôli sériovému prenosu dát používa trieda DS1996 tri registre nazývané TA1, TA2 a E/S. Registre TA1 a TA2 musia obsahovať adresu, do ktorej sa bude zapisovať, alebo z ktorej sa bude čítať. Register E/S slúži ako bajtové počítadlo a status prenosu. Slúži na overenie integrity dát po Write príkazoch. Master má teda možnosť tento register iba čítať. Spodných päť bitov E/S registra obsahuje adresu posledného zapísaného bajtu do vyrovnávacej pamäte. Piaty bit zvaný PF (partial byte flag) obsahuje informáciu či posielané dáta mali dĺžku, ktorá je celočíselným násobkom ôsmych. Šiesty bit OF (Overflow) nesie informáciu či zapísané dáta nepresiahli veľkosť vyrovnávacej pamäte. Najvyšší bit AA (Authorization accepted) nesie informáciu, či dáta vo vyrovnávacej pamäti už boli prekopírované do pamäte. Zápis do vyrovnávacej pamäte vynuluje tento bit. Registere E/S je vidieť na obrázku 2.6.



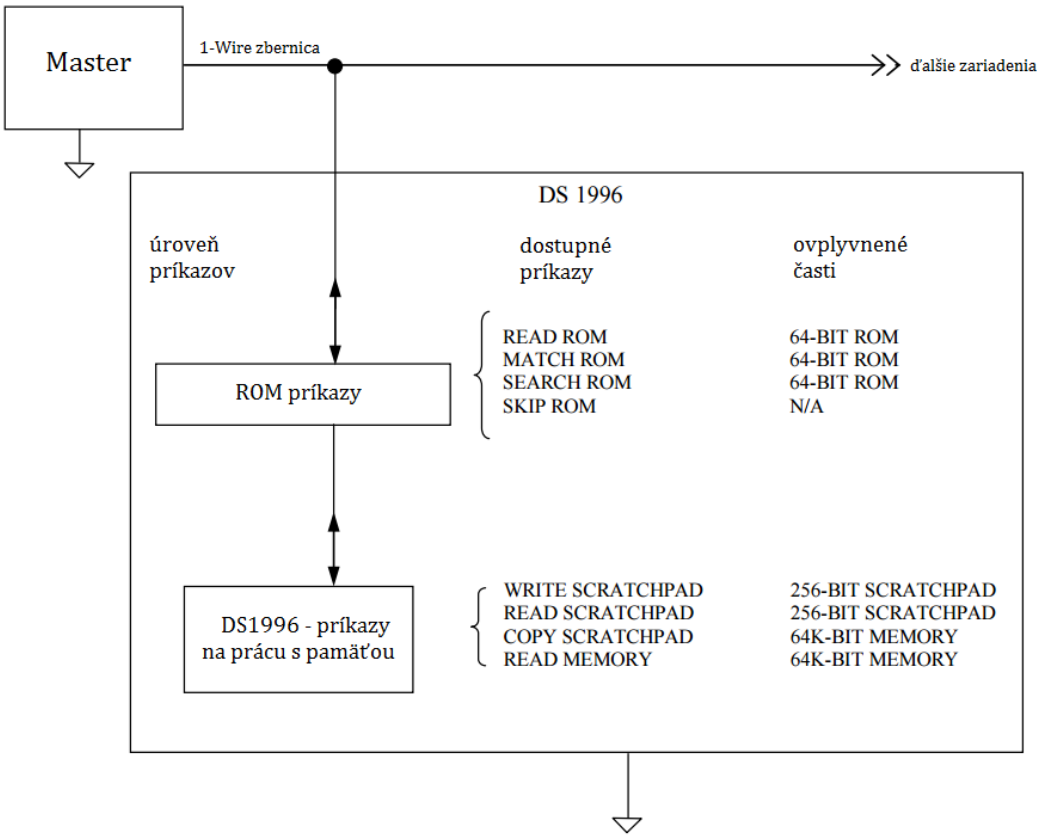
Obr. 2.6: Register E/S

2.4.2 Podporované príkazy

Hierarchickú štruktúru príkazov je vidieť na obrázku 2.7. Master musí poslať jeden zo základných ROM príkazov: read ROM, match ROM, search ROM, skip ROM. Po ROM príkaze a jeho úspešnom vykonaním sú prístupné aj príkazy na prácu s pamäťou. Master môže poslať ľubovoľný zo štyroch príkazov:

- Write Scratchpad (0x0F)
- Read Scratchpad (0xAA)
- Copy Scratchpad (0x55)
- Read Memory (0xF0)

Všetky dáta, ktoré sa čítajú alebo zapisujú začínajú od najmenej významných bitov.



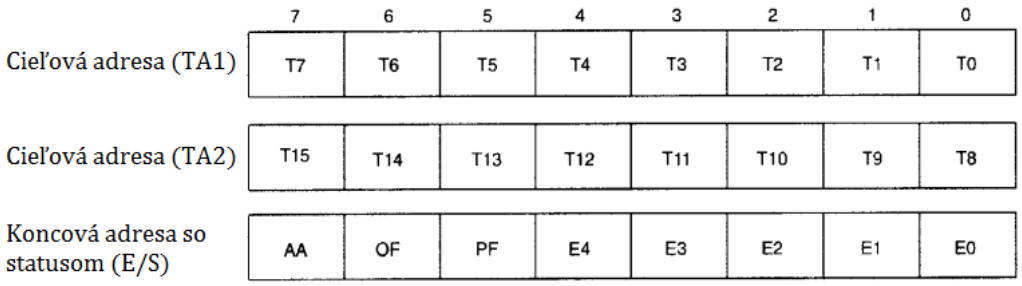
Obr. 2.7: Hierarchická štruktúra príkazov pre triedu DS1996

2.4.3 Príkazy na prácu s pamäťou

Ako bolo popísane vyššie a ukázané na obrázku 2.7 je nutné prejsť základnými časťami protokolu pred prístupom k pamäti. Pre analýzu a simuláciu dátového iButtonu triedy DS1996 je nutné pochopiť jeho správanie. Preto v tejto časti popíšeme správanie jednotlivých príkazov.

2.4.3.1 Write scratchpad

Master po poslaní príkazu write scratchpad musí poslať dva bajty obsahujúce cieľovú adresu. Následne posiela dáta, ktoré majú byť zapísané do vyrovnávacej pamäte. Značenie v registroch je vidieť na obrázku 2.8. Dáta vo vyrovnávacej pamäti budú začínať na bajtovom offsete T4:T0. Do pamäte budú uložené na adresu získanú z registrov TA2 a TA1. Koniec zápisu bude na E4:E0, ktorý si bude môcť master prečítať pri čítaní vyrovnávacej pamäte.



Obr. 2.8: Adresové registre

2.4.3.2 Read scratchpad

Tento príkaz slúži na overovanie integrity dát zapísaných vo vyrovnávacej pamäti. Po poslaní príkazu začne master čítať posielané dáta, ktoré v prvých dvoch bajtoch obsahujú cieľovú adresu aká bola uvedená v registroch TA1 a TA2. Ďalší bajt je E/S register, v ktorom sú už nastavené bity PF, OF, AA. Ďalej nasledujú dáta uložené vo vyrovnávacej pamäti začínajúc offsetom T4:T0 (keďže vyrovnávacia pamäť má len 32 bajtov, časť adresy uloženej v registri TA2 nemá vplyv a vyhodnotí sa len offset uložený v prvých piatich bitoch TA1).

2.4.3.3 Copy scratchpad

Príkaz slúži na prekopírovanie dát z vyrovnávacej pamäti do finálnej pamäti na správnu adresu. Je možné ho vykonať až po read scratchpad príkaze. Po poslaní copy scratchpad príkazu musí master poslať tri bajty, ktoré mal obdržať pri čítaní a kontrole dát. Dáta sa musia zhodovať s dátami uloženými v troch registroch TA1, TA2 a E/S presne v tomto poradí. Ak sú správne, nastaví sa AA flag v E/S registri a začne sa s kopírovaním do pamäte. Po prekopírovaní dát sa posiela logická nula až po najbližší reset pulz. Všetky pokusy o prerušenie kopírovania sú ignorované. Vďaka vlastnému napájaniu je možné vykonávať proces kopírovania do pamäte nehľadiac na prebiehajúcu komunikáciu na 1-Wire zbernici. Kopírovanie zvyčajne trvá 30 μs.

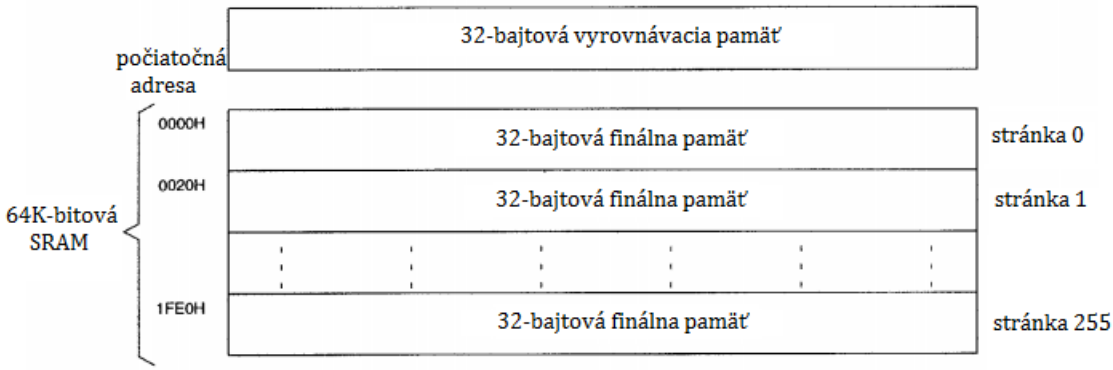
2.4.3.4 Read memory

Tento príkaz sa môže použiť na čítanie celej uloženej pamäti. Po poslaní príkazu master posiela dva bajty TA1 a TA2, ktoré určujú adresu na čítanie. Čítanie sa začne na danej adrese a môže pokračovať až po koniec pamäte. Za koncom sa posielajú už len logické jednotky. Register E/S sa pri tom nijak nemení. Proces čítania dát má urýchlený prenos dát. To je možné vidieť aj v ďalších častiach práce. Pri analýze dát je jednoznačne vidieť, že posielané bity pri čítaní z iButtonu DS1996 sú výrazne rýchlejšie ako bežná komunikácia. V štandarde sa odporúča dáta zabalit do packetov, ktoré sú veľkosti

jednej stránky. Takýto packet by mal obsahovat aj 16-bitový kontrolný súčet pre každú stránku pamäte. Umožňuje to rýchlejsie overenie integrity čítaných dát bez nutnosti opakovaného čítania.

2.4.4 Zápis s verifikáciou

Pri zápise dát na iButton je nutné použiť vyrovnávaciu pamäť. Najskôr master pošle príkaz write scratchpad, následne pošle registre obsahujúce adresu kam chce zapisovať. Potom posiela dáta, ktoré sa zapisujú do vyrovnávacej pamäte. V ďalšom kroku pošle príkaz read scratchpad, aby overil integritu dát, ktoré sa do vyrovnávacej pamäte zapísali. iButton pošle najskôr registre TA1, TA2 a E/S, z ktorých sa overí platnosť adresy aj možné chyby. Ak je všetko v poriadku, iButton začne posielať dáta z vyrovnávacej pamäte, ktoré master kontroluje. Keď master skontroluje dáta musí poslať príkaz copy scratchpad, aby sa dáta prekopírovali do pamäte na požadovanú adresu zapísanú v registroch TA1, TA2. Rozloženie pamäte je vidiet na obrázku 2.9. Podrobnejšie informácie o triede DS1996 sú dostupné v technickom liste [4].



Obr. 2.9: Rozloženie pamäte

Kapitola 3

Antiklon

V našej práci sme testovali antiklon RAK DEK pre jednu dotykovú plochu. Zariadenie sa dá zakúpiť u špecializovaných predajcov a podľa oficiálneho popisu je učené pre operačnú jednotku RAK DEK na zistenie a blokovanie klonov (kópií elektronických identifikátorov typu iButton) DEK kľúčov. Je schopný zistiť, či 48-bitové sériové číslo iButton čipu priloženého k dotykovej ploche je originálne alebo skopírované.

Zapája sa medzi operačnú jednotku a dotykovú plochu. Antiklon zablokuje skopírované identifikačné číslo a nepošle ho ďalej do operačnej jednotky. Zistenie neoriginálneho identifikačného čísla signalizuje krátkym tónom. Ak je k dotykovej ploche priložený originál iButton čip, antiklon pošle identifikačné číslo ďalej do operačnej jednotky, ktorá preverí jeho existenciu v databáze užívateľov. Ak sa tam číslo nachádza, aktivuje relé pre odblokovanie zámkov.

Problém s neoriginálnymi iButtonmi je hlavne v tom, že väčšina príkazov, ktoré sme si popísali v úvodnej časti, vychádza z predpokladu, že sériové číslo je unikátne a nemenné. Ďalší problém vzniká s praktickým využitím iButtonov ako kľúčov pre prístupové systémy. Pokročilejšie operačné jednotky, ako napríklad UNISIEŤ, podporujú rozšírenú funkcionalitu na sledovanie a logovanie prístupov. To vie byť užitočné pre zaradenie používateľov do rôznych skupín, odlíšenie návštev, pozorovanie podozrivého správania. Táto funkcionalita však stráca význam, ak môže byť naraz niekoľko kľúčov s rovnakým sériovým číslom. Nevýhodou je aj strata iButtonu. Za bežných okolností stačí vymazať daný kľúč z databázy a nahrať nový pre majiteľa. Ak pripustíme klony, treba ich vymeniť všetky alebo prepísať na novú hodnotu uloženú v databáze.

Nás zaujímalo, či vieme simulovať iButton, ktorý nie je originálny a stále by bol akceptovaný aj antiklonom. Úspešná simulácia by znamenala, že útok popísaný v práci Útok na digitálne elektronické kľúče pomocou časového postranného kanálu[8], by potenciálne mohol byť realizovateľný aj za prítomnosti antiklonu.

3.1 Klon iButtonu

Klon je neoriginálny elektronický čip, ktorý má prepisovateľnú pamäť pre 48-bitové sériové číslo. Pri zakúpení je táto pamäť prázdna a pre sfunkčnenie čipu je nutné do neho nakopírovať sériové číslo už existujúceho čipu. Pri používaní klonov sa môže aj bez antiklonu vyskytovať nespoľahlivé odblokovanie dverí, tzn. po priložení k dotykovej ploche nie je identifikačné číslo prečítané a čip je potrebné prikladať opakovane.

Neoriginálne čipy s prepisovateľnou pamäťou sú označované rôzne. Môžu byť bez gravírovania alebo s gravírovaním, ktoré obsahuje označenie výrobcu, typ prepisovateľného čipu apod. Klony nikdy nemajú vygravírované identifikačné číslo.

Klony sú voľne predajné a ich cena sa pohybuje približne v polovici až tretine ceny originálu. Označujú sa typicky RW1990. Dá sa nájsť niekoľko rôznych prístupov ako takýto klon prepisovať. Niektorí predajcovia priamo ponúkajú jednoduchý hardvér a softvér, ktorý slúži na prepisovanie zakúpených klonov. Iný ponúkajú len klony s popisom, aký príkaz akceptujú na prepisovanie pamäti. Využívajú hexadecimálne číslo, ktoré nie je obsadené žiadnym oficiálnym príkazom. Najčastejšie príkazy na prepis pamäte, ktoré sme našli sú 0xD5 a 0x3C. Vyskytujú sa v návodoch mnohých zahraničných používateľov.

3.2 RAK DEK

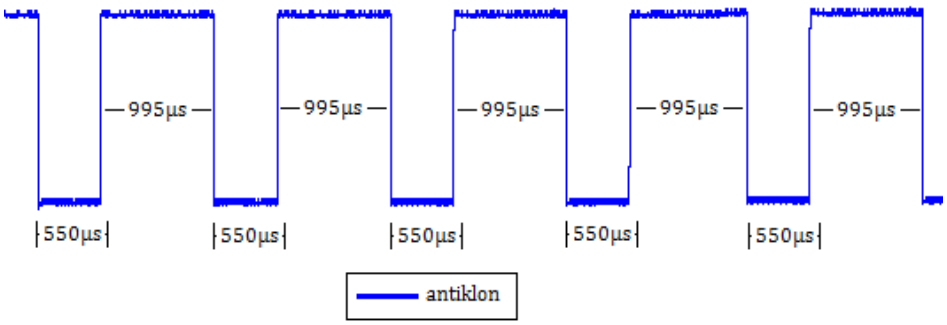
Operačná jednotka RAK DEK je jednoduchá operačno-pamäťová jednotka, ktorá s použitím softvéru RAK umožňuje vytvorenie jednoduchej databázy, definovanie vlastností zariadenia a identifikátorov. Pre programovanie je nevyhnutné mať operačnú jednotku pripojenú k PC pomocou USB kábla. Na doske plošných spojov sa nachádzajú tlačidlá pre programovanie a mazanie databázy. Tlačidlá umožňujú vkladanie identifikátorov do databázy priamo cez dotykovú plochu pripojenú k operačnej jednotke alebo zmazanie celej aktuálnej databázy.

3.3 Hypotéza a spôsob merania

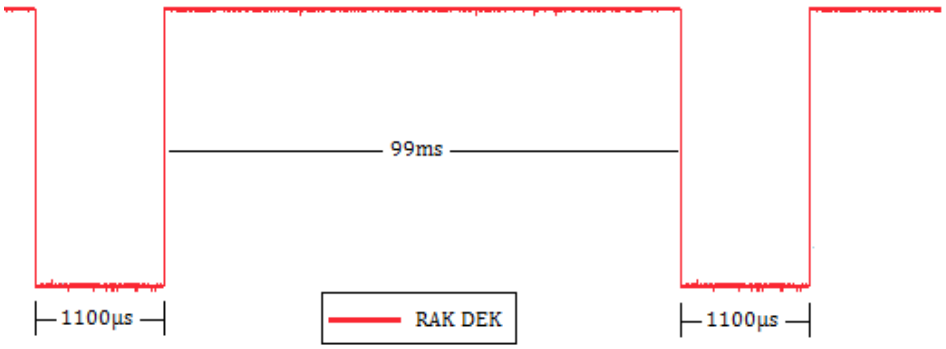
Našim prvým predpokladom bolo, že zariadenie antiklon bude postupne testovať rôzne známe príkazy, ktoré slúžia na zápis nového sériového čísla. Ak by zistil, že priložený iButton poslaný príkaz pozná, vie ho označiť za neoriginálny. Na potvrdenie takéhoto predpokladu sme potrebovali namerať prebiehajúcu komunikáciu medzi iButtom priložením k dotykovej ploche a antiklonom pripojením k operačnej jednotke RAK DEK. Na meranie komunikácie sme využívali osciloskop Picoscope 6403D. Na spracovanie nameraných dát sme použili oficiálny softvér PicoScope 6. Veľkou výhodou bolo, že softvér podporuje sériové dekódovanie komunikácie využívajúcej 1-Wire protokol.

3.4 Výsledky merania

V prvom meraní sme pripojili antiklon medzi operačnú jednotku RAK DEK a dotykovú plochu. Z nameranej komunikácie na obrázkoch 3.1 a 3.2 je vidieť, že antiklon posiela reset pulz oveľa častejšie ako operačná jednotka RAK DEK. Antiklon ho generuje pravidelne každých 995 μs. Operačná jednotka RAK DEK posiela reset pulz pravidelne každých 99 ms. Pozorovateľný rozdiel je aj v dĺžke reset pulzu. Reset pulz generovaný antiklonom trvá okolo 550 μs , zatiaľ čo RAK DEK má trvanie reset pulzu až 1100 μs.



Obr. 3.1: Perióda reset pulzov antiklonu

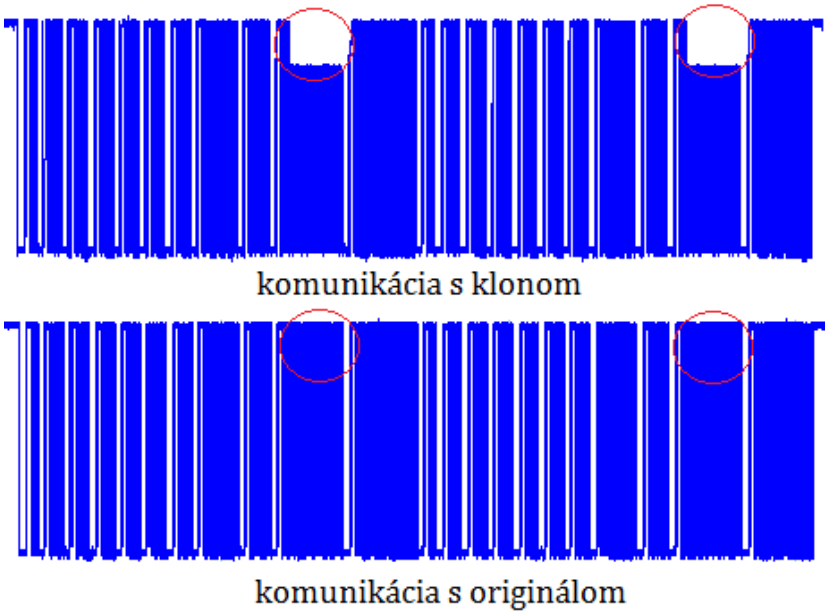


Obr. 3.2: Perióda reset pulzov operačnej jednotky RAK DEK

V ďalších dvoch meraniach sme chceli porovnať komunikáciu originálneho iButtonu a klonovaného. Antiklon testuje dátovú zbernicu reset pulzom častejšie, aby stihol vykonať všetky svoje príkazy čo najskôr po priložení iButtonu. Z meraní sme totiž zistili, že antiklon posiela spolu až 22 príkazov. Niektoré sa však opakujú. Zoznam unikátnych príkazov, ktoré posiela antiklon je 0xB5, 0xE3, 0x0F, 0x33, 0xCE, 0xCA, 0xF0.

Je vidieť, že v testovaných príkazoch sa nachádzajú základné ROM príkazy ako read ROM (0x33, 0x0F) a search ROM (0xF0). Nachádzajú sa tam príkazy, ktoré teoreticky podporujú iné triedy iButtonov ako napríklad príkaz 0xB5. A príkazy, ktoré oficiálne nie sú podporované žiadnou triedou iButtonov. Prehľad oficiálnych príkazov

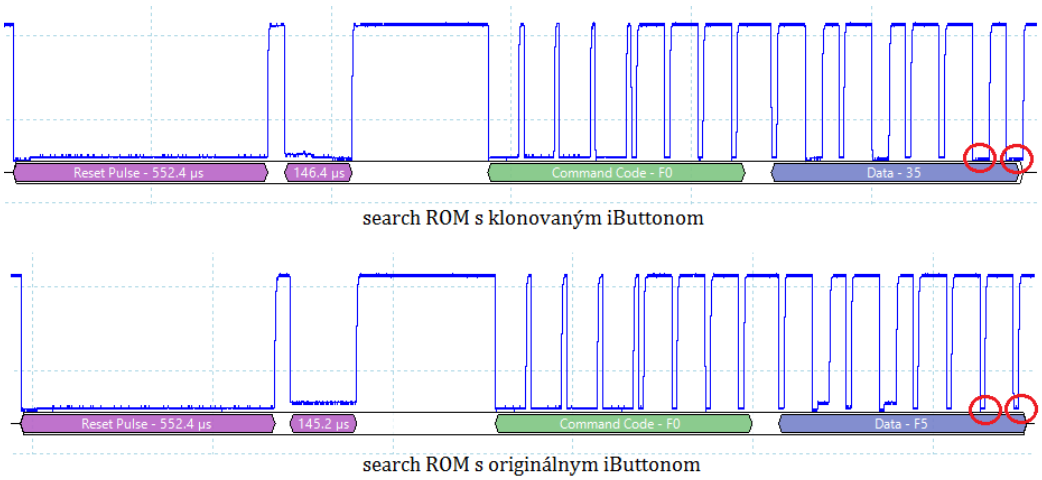
pre iButtony je spracovaný do tabuľky na `http://owfs.sourceforge.net/family.html`. Namerané komunikácie sa nakoniec líšili len v dvoch podstatných veciach. Jednou je pokles napätia na zbernici po príkaze 0xCA od klonovaného iButtonu a druhou je odpoveď na oficiálny príkaz search ROM. Porovnanie celkového priebehu komunikácie a pozorovateľný pokles napätia je vidieť v zväčšenej verzii nameranej komunikácie na obrázku 3.3. V oboch prípadoch sa ako dáta po príkaze prenášajú čisté jednotky. Čiže je možné, že antiklon len generuje časové okná a očakáva odpoveď od iButtonu, alebo naopak priamo posiela bity s hodnotou jedna. Tieto dve možnosti sa nedajú odlíšiť z priebehu komunikácie. Isté však je, že originálny iButton by takýto príkaz nemal poznať a podporovať. Rovnako zvláštny je aj fakt, že niektoré príkazy antiklon pošle no následne hneď spustí nový reset pulz. Iným príkazom vytvára časové úseky ďalej, či už na posielanie dát alebo čítanie. Nezávisí to však od posielaného príkazu. Napríklad základný príkaz read ROM sa celý vykoná len dva krát z ôsmych poslaní príkazu.



Obr. 3.3: Porovnanie priebehu komunikácie s pozorovateľným poklesom napätia

Priebeh komunikácie a rozdiel v odpovedi na príkaz search ROM je vidieť na obrázku 3.4. Pred search algoritmom sa niekoľko krát vyskytne aj príkaz read ROM. Takže antiklon má mať informáciu o sériovom čísle práve priloženého iButtonu. Rozdiel pri príkaze search ROM nastáva v tom, že originálny iButton odpovie správne podľa algoritmu, ktorý sme popísali v úvodnej časti práce. To sa prejaví ako bajt s hexadecimálnou hodnotou 0xF5. Klonovaný iButton z neznámych dôvodov odpovie zle a prejaví sa to ako bajt s hodnotou 0x35. Presnejšie priebeh úvodného bajtu po príkaze search ROM by mal vyzerať tak, že zariadenie pošle najmenej významný bit. Ten je v tomto prípade s hodnotou jedna kvôli family code (0x01). Následne sa očakáva jeho negovaná hodnota, čiže nula. Master, v tomto prípade antiklon, odpovie hodnotou bitu,

ktorý si vybral. Master sa rozhodne pre bit s hodnotou jedna. Pokračuje sa posielaním druhého bitu, ktorý je nula. Nasleduje negácia a master znovu vyberie hodnotu jedna. Tu má nastať situácia, že priložený iButton zistí, že jeho bit má odlišnú hodnotu ako je tá, ktorú vybral master a do zbytku komunikácie nemá zasahovať. To sa prejaví v komunikácii ako bity s logickou hodnotou jedna. Čo presne zodpovedá reakcii originálu. Klon sa snaží komunikovať ďalej, ale ani to nie v súlade s algoritmom. Celkový priebeh tohoto algoritmu nie je vidieť, pretože anitklon zastaví komunikáciu už po prvom takto vygenerovanom bajte (overia sa len prvé 3 bity adresy v behu algoritmu search ROM). Antiklon sa vo všetkých nameraných komunikáciách správal rovnako, takže si neudržuje žiadny vnútorný stav pre search algoritmus mimo jedného overovania.



Obr. 3.4: Rozdielne odpovede na príkaz search ROM

Z týchto meraní sme usúdili, že buď by mohol byť antiklon schopný rozpoznať pokles napätia, ktorý by mohol predstavovať pokus o zápis do pamäte alebo inú výpočtovo náročnejšiu činnosť, alebo vie na základe prečítaného sériového čísla vyhodnotiť správnosť behu algoritmu search ROM.

3.5 Arduino

Ďalším krokom bolo skúsiť simulovať iButton a zistiť, či je možné obísť antiklon. Na simuláciu iButtonu sme sa rozhodli používať Arduino UNO. Arduino je open-source platforma založená na mikrokontroléroch firmy Atmel a grafickom vývojovom prostredí. Samotná doska obsahuje 14 digitálnych vstupov/výstupov a 6 analógových. Ďalej konektor na pripojenie externého napájania a resetovacie tlačidlo. Rovnako je na doske pripojená jedna LED dióda, ktorú je možné ovládať. Ostatné voliteľné periférie je nutné pripojiť zvlášť. Výhodou zariadenia Arduino je jednoduché pripojenie k počítaču a programovanie. Arduino je vybavené micro USB konektorom a po pripojení

k PC sa hlási ako sériový port. Vlastné programovanie prebieha v jednoduchom prostredí Arduino IDE. Vývojové prostredie je napísané v Jave a samotné mikrokontroléry sa programujú v jazyku C++. My sme však pre nutnú časovú presnosť jednotlivých funkcií využívali aj písanie častí kódu v assembleri. Podrobný popis a informácie o zariadení sú dostupné z oficiálnej stránky projektu [1].

3.6 Simulácia iButtonu

Arduino poskytuje oficiálnu 1-Wire knižnicu, ktorú sme chceli pri simulácii využiť. Ukázalo sa však, že knižnica poskytuje rozhranie len z pohľadu mastera. Knižnica je dostupná online [6] spolu s dokumentáciou. Je pochopiteľné, že základné využitie zakladá na tom, že máte iButton k dispozícii a programujete rozhranie, ktoré s ním má komunikovať. Nám však takéto riešenie nijak nepomohlo. Tak sme hľadali dostupnosť hotových riešení, ktoré by boli schopné cez arduino simulovať iButton. Dá sa nájsť viacero neoficiálnych open-source projektov, ktoré sa o takúto implementáciu pokúšajú. Vyskúšali sme niekoľko rôznych. Všetky sú dostupné na stránke `https://github.com`. Konkrétne sa jednalo napríklad o OneWireHub od orguala, OneWireArduinoSlave od neuoy alebo OneWireSlave od Marcusa Lange. Žiaľ, žiadne z nich nefungovalo úplne. Najväčší problém s takýmito riešeniami je, že autori sa nesnažili implementovať 1-Wire protokol vo všeobecnosti, ale skôr ho využiť na jeden svoj konkrétny prípad. Tým vznikajú problémy ako zafixovaná dĺžka reset pulzov, problémy so synchronizáciou jednotlivých bitov, predpokladanie fixnej postupnosti príkazov. Tieto veci pri jednom konkrétnom riešení a odladení nie sú problém, ale pre všeobecné riešenie sú neakceptovateľné. Napríklad v našom prípade je vidieť, že operačná jednotka RAK DEK posiela reset pulz dva krát dlhší ako antiklon. Čiže fixný predpoklad o dĺžke reset pulzu je neprípustný. Ďalší problém je očakávanie ideálnej postupnosti reset pulz, presence pulz, príkaz, dáta. Môže dôjsť k problému s kontaktom zariadení alebo iným dôvodom, s ktorými tieto riešenia nepočítajú. Už priamo operačné jednotky, ktoré sme mali k dispozícii aj za bežných a ideálnych podmienok pošlú v úvode komunikácie z neznámych dôvodov reset pulz dva krát po sebe. A synchronizácia by celkovo mala byť zabezpečovaná čakaním na spádovú hranu generovanú masterom a nie počítaním očakávanej dĺžky časových úsekov. Keďže žiadne z nájdených riešení v našom prípade nefungovalo, tak sme museli všetky zavrhnúť a rozhodli sa implementovať svoju vlastnú verziou 1-Wire protokolu z pohľadu iButtonu. Úpravy existujúcich riešení by boli len nepríjemným variantom vlastnej implementácie. Naša implementácia sa opiera o kód použitý pre Arduino v práci „Útok na digitálne elektronické kľúče pomocou časového postranného kanálu“ [8]. Použili sme základné funkcie a upravili ich tak, aby neboli špecifické pre konkrétne zariadenie, na ktorom bol útok realizovaný. Uvádzame popis

jednotlivých funkcii s ukážkami pseudokódu pre hlavné časti protokolu.

3.6.1 Implementácia 1-Wire protokolu z pohľadu iButtonu

Celá implementácia zakladá na rozbití komunikácie do nezávislých funkcií. Funkcie by mali zodpovedať jednotlivým častiam 1-Wire protokolu, ktoré boli popísané v prvej kapitole práce. Následne sa pospájajú jednotlivé funkcie v správnom poradí s prihliadaním na aktuálnu komunikáciu. Spoločný základ pre všetky funkcie tvorí schopnosť prepočítavať operácie procesora na reálny čas, vedieť invertovať čas na operácie procesora a sledovanie zmeny stavu na zbernici (pri arduine je to zmena stavu digitálneho pinu). Na výpočet času slúži assemblerová funkcia, ktorá na základe počtu vykonaných operácii vie vrátiť čas v mikrosekundách. Na výpočet počtu operácii z času slúži jej inverzná funkcia. Na sledovanie zmeny stavu pinu sa využívajú assemblerové inštrukcie sbic a sbis. Inštrukcia sbic testuje jeden bit vstupno-výstupného registra a preskočí nasledujúcu inštrukciu ak má bit hodnotu nula. Inštrukcia sbis naopak preskočí nasledujúcu inštrukciu v prípade, že bit vstupno-výstupného registra je jedna.

Prvou funkciou 1-Wire protokolu na implementovanie ja zaregistrovanie reset pulzu na dátovej zbernici. To sa podľa štandardu prejaví zníženým napätím po dobu aspoň 480 μs. To znamená, že sledujeme pin, na ktorom je pripojená 1-Wire zbernica a čakáme na pokles v napätí (kým hodnota na pine nie je nula). Keď nastane pokles napätia, začneme počítať čas kým napätie stúpne späť (kým hodnota na pine nie je jedna). Ak je tento čas aspoň 480 μs, tak sme detegovali reset pulz a následne je potrebné potvrdiť prítomnosť na zbernici presence pulzom. Počítanie času sa robí súčtom operácii, ktoré procesor stihne vykonať, kým nastane zmena hodnoty na pine. Pseudokód reset pulzu je popísaný nižšie na 3.5. Funkcie dostávajú ako argumenty vstupno-výstupný register a bit, ktorý určuje sledovaný bit registra. To spolu v kombinácii určuje, ktorý pin arduina máme pripojený a sledujeme.

```
void cakaj_na_reset_pulz()
{
  uint16_t doba;
  do {
    cakaj_kym_bit_nieje_nula(Register, Bit);
    pocitaj_pokial_bit_nieje_jedna(Register, Bit, &doba);
  }
  while(doba < mikrosek_do_cyklov(480));
}
```

Obr. 3.5: Pseudokód reset pulzu

Po obdržaní reset pulzu máme podľa štandardu 480 μs na odpoveď. Zmenu stavu zbernice z logickej jednotky na logickú nulu zabezpečíme využitím arduino funkcie pinModeOutput, ktorý zmení pin na výstupný. To spojí dátový vodič so zemou, čím sa vytvorí pokles v napätí. V tomto stave počkáme 60 μs až 240 μs (tu na čase, ktorý nastavíme nezáleží ak spĺňa rozsah zo štandardu), zmeníme pin späť na vstupný funkciou pinModeInput a počkáme, kým napätie stúpne späť na pôvodnú hodnotu (vďaka pull-up odporu). Na 3.6 je uvedený pseudokód presence pulzu.

```
void posli_presence_pulz()
{
  pinModeOutput();
  cakaj_mikrosekund(150);
  pinModeInput();
  cakaj_kym_bit_je_jedna(Register, Bit);
}
```

Obr. 3.6: Pseudokód presence pulzu

Po poslaní presence pulzu sa očakáva od mastera poslanie príkazu. Príkaz pozostáva z ôsmych bitov. Pre každý bit čakáme na spádovú hranu vygenerovanú masterom a počítame dobu počas ktorej master drží napätie znížené. Následne podľa trvania zníženého napätia vyhodnotíme či sa jednalo o logickú jednotku alebo nulu. Ak je napätie znížené dlhšie ako 480 μs jedná sa o reset pulz a komunikáciu začíname od začiatku. Pseudokód takéhoto testovania bitov je vidieť na 3.7. Ak úspešne prečítame všetkých osem bitov, tak ich vyhodnotíme a porovnáme s podporovanými príkazmi. V prvej verzii sme sa rozhodli podporovať len príkaz read ROM, ktorý používa operačná jednotka RAK DEK pri komunikácii s iButtonom na overovanie sériového čísla. Neznáme príkazy ignorujeme a čakáme na príchod nového reset pulzu. Vyhodnocovanie príkazov je vidieť na pseudokóde 3.8. Vidieť v ňom, že sme využívali inline funkcie. Dôvodom na vytvorenie inline funkcie, je využitie direktívy pre kompilátor, aby urýchlil jej vykonávanie. Zvyčajne sa to rieši tým, že kompilátor vloží telo funkcie priamo na miesta, kde je funkcia zavolaná. Pre nutnú presnosť počas počítania uplynutého času na základe počtu vykonaných inštrukcií procesora sme využívali inštrukcie cli a sei. Inštrukcia cli zakáže všetky prerušovania, ktoré by bežne mohli nastať a tým prerušiť nami počítané a vykonávané inštrukcie. Inštrukcia sei opať povolí prerušenia. Návratová hodnota dva našej funkcie slúži na signalizáciu, že nastal reset pulz.

```
inline uint8_t citaj_bit() {
  uint16_t trvanie;
  cli();
  cakaj_kym_bit_nieje_nula(Register,Bit);
  pocitaj_kym_bit_nieje_jedna(Register,Bit,&trvanie);
  sei();
  if (trvanie >= mikrosek_na_cykly(480))
    return 2;
  if(trvanie < mikrosec_na_cykly(40))
    return 1;
  else
    return 0;
}
```

Obr. 3.7: Pseudokód testovania bitu pri čítaní príkazu

```
inline uint8_t vyhodnot_prikaz() {
  uint16_t prikaz = 0;
  uint16_t pocetBitov = 8;
  uint16_t hodnota;
  do {
    prikaz >>= 1;
    cakaj_kym_bit_je_jedna(Register,Bit);
    hodnota = citaj_bit();
    if (hodnota == 2)
      return 0xFF
    if (hodnota == 1)
      prikaz |= 1 << 7;
    pocetBitov--;
  } while (pocetBitov != 0);
  return prikaz;
}
```

Obr. 3.8: Pseudokód vyhodnocovania príkazov

Ak obdržíme príkaz read ROM očakávame, že master bude postupne generovať časové úseky, v ktorých postupne po bitoch pošleme sériové číslo. Sériové číslo si môžeme zvoliť ľubovoľné, ktoré má platný family code 0x01 a platný kontrolný súčet. Môžeme simulovať aj iButtony, ktoré toto nespĺňajú, ale operačná jednotka aj antiklon ich auto-

maticky ignorujú. Ak master miesto časového úseku vygeneruje reset pulz, komunikácia sa začne od začiatku. Posielanie sériové čísla sa skladá z čakania na spádovú hranu generovanú masterom a následne odpovede podľa logickej hodnoty posielaného bitu. Na poslanie jednotky len počkáme kým sa napätie vráti na pôvodnú hodnotu vďaka pull-up odporu. Pri posielaní nuly podržíme napätie znížené, rovnako ako pri presence pulze, po dobu 40 μs. Pseudokód posielania jednotky je vidieť na 3.9 a posielanie nuly na 3.10.

```
void posli_jedna() {
    cli();
    cakaj_kym_bit_nieje_nula(Register, Bit);
    sei();
    cli();
    cakaj_kym_bit_nieje_jedna(Register, Bit);
    sei();
 }
```

Obr. 3.9: Pseudokód vyhodnocovania príkazov

```
void posli_nulu() {
    cli();
    cakaj_kym_bit_nieje_nula(Register, Bit);
    sei();
    pinModeOutput();
    cakaj_mikrosekund(40);
    pinModeInput();
    cli();
    cakaj_kym_bit_nieje_jedna(Register, Bit);
    sei();
 }
```

Obr. 3.10: Pseudokód vyhodnocovania príkazov

3.7 Výsledný program

Náš výsledný program počká na reset pulz, odošle presence pulz a očakáva príkaz. Ak pri čítaní niektorého bitu zistí, že nastal reset pulz ide od začiatku. Ak obdrží neznámy príkaz, tak počká na ďalší reset pulz. V prípade známeho príkazu reaguje v závislosti

od štandardu. Ďalším krokom implementácie by mohlo byť doimplementovanie ostatných štandardných ROM príkazov pre triedu DS1990. Pre naše účely však stačil príkaz read ROM. V takto pripravenom programe sme si pripravili posielanie identifikačného čísla, ktorého originálny aj klonovaný iButton sme mali k dispozícii. Naprogramované arduino simulujúce iButton sme otestovali najskôr s operačnou jednotkou RAK DEK. Priebeh komunikácie bol v poriadku a kľúč nahraný v databáze bol akceptovaný. Pokračovali sme zapojením antiklonu, ktorý úspešne odlišoval klon od originálu iButtonu, ale nami simulovaný kľúč akceptoval tiež. To znamená, že stačilo ignorovať všetky príkazy okrem read ROM a výsledný program úspešne odsimuloval originál.

3.8 Výsledok simulácie

Po zachytení a analýze komunikácie prebiehajúcej medzi antiklonom a iButtonom sme boli schopný simulovať posielanie identifikačného čísla iButtonu, ktoré neodlíšilo ani zariadenie antiklon. To znamená, že s úpravami kódu ktoré sme zrealizovali pri simulácii by bolo možné uskutočniť útok popisovaný v práci [8]. Bolo by však nutné overiť či svietenie LED diódy generuje antiklon, alebo len prenáša svietenie generované operačnou jednotkou RAK DEK. Jediné čo sa v prípade, že je útok naďalej realizovateľný, mierne zhorší je čas potrebný na realizáciu útoku. Kým antiklon vykoná všetky svoje testy niekedy nestihne pravidelný reset pulz operačnej jednotky RAK DEK. Takže uskutoční poslanie identifikačného čísla operačnej jednotke až pri jej ďalšom reset pulze. Lenže pravidelné reset pulzy mimo komunikácie chodia každých 99 μs, takže predĺženie nie je veľké. Všetky merania, z ktorých sme vychádzali sú dostupné v prílohe práce.

3.8.1 Možné zlepšenia antiklonu

Základný problém zariadenia antiklon je, že teoreticky sa vždy dá spraviť simulácia existujúceho originálneho iButtonu. Simulovaný originál je v podstate jeho klon, ktorý je v celej komunikácii identický. Vzhľadom na aktuálne používanie neštandardných príkazov na zápis, je možné najbežnejšie z nich testovať a overiť či ich priložené zariadenie podporuje alebo nie. To má výhodu v tom, že klony sa predávajú a vyrábajú vo veľkom. Takže sa dá naraz odstaviť všetky, ktoré podporujú zapisovanie do pamäte určitým príkazom. No celkovo zabrániť vzniku klonov je zrejme nereálne. Jednou z chýb aktuálneho zariadenia je aj to, že neoveruje štandardné príkazy dôkladne. Zariadenie testuje príkazy ako sú read ROM dokonca v dvoch variantoch 0x33 a 0x0F pre triedu DS1990, ale nevadí ak náš klon príkaz 0x0F úplne ignoruje. Rovnako príkaz search ROM, ktorý musí podporovať každý oficiálny iButton.

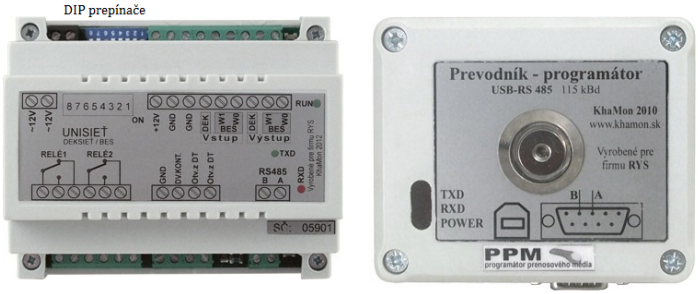
Na druhú stranu klony by pre ich cenovú dostupnosť mohli kľudne existovať aj vo verzii, ktorá podporuje zápis do pamäte len jednorázovo. Po zápise by ďalej vystupovali

ako originálny iButton s daným sériovým číslom. Ďalšou možnosťou je mať hardvérový prepínač, ktorý by umožňoval zápis do pamäte. Ak by nebol zapnutý, tak by sa všetky príkazy na zápis ignorovali. Iste existujú aj ďalšie riešenia, ale klony majú naďalej problém s tým, že s nimi oficiálny štandard nepočíta. Zároveň kazia myšlienky, ktoré by sa inak na tejto technológii dali realizovať.

Kapitola 4

UNISIEŤ

Univerzálna operačno-pamäťová jednotka, podporuje obe technológie RFID aj iButton. Takže k operačnej jednotke sa môže pripájať bezkontaktná čítačka i dotyková plocha súčasne. Sieťové vyhotovenie umožňuje komunikáciu so vzdialeným PC, ktorý má prístup k obojsmernej kontrole jedných dverí s nezávislým záznamom o vstupe a výstupe. Pred samotným softvérovým nastavovaním vlastností je potrebné nastaviť sieťové adresy na operačno-pamäťových jednotkách pomocou DIP prepínačov. Tým sa zabezpečí správna komunikácia a aktualizácia údajov v systéme. Pomocou DIP prepínačov je možné nastaviť aj dĺžku otvárania dverí a aktivovať funkciu časový snímač, ktorá slúži na kontrolu stavu a dĺžky otvorenia dverí. Ďalej podporuje funkciu zapínací kontakt a programátor prenosového média. Pripája sa k PC pomocou programátora-prevodníka. Programátor sa využíva na programovanie operačnej jednotky UNISIEŤ. Zabezpečuje prenos identifikačných čísel jednotlivých DEK kľúčov do zoznamu kľúčov operačnej jednotky cez osadenú dotykovú plochu. Obsahuje tiež prevodník USB-RS485, čím umožňuje online prepojenie medzi PC a operačnou jednotkou UNISIEŤ na väčšie vzdialenosti. Podporuje tiež funkciu programovanie prenosového média. Obe zariadenia je vidieť na obrázku 4.1.



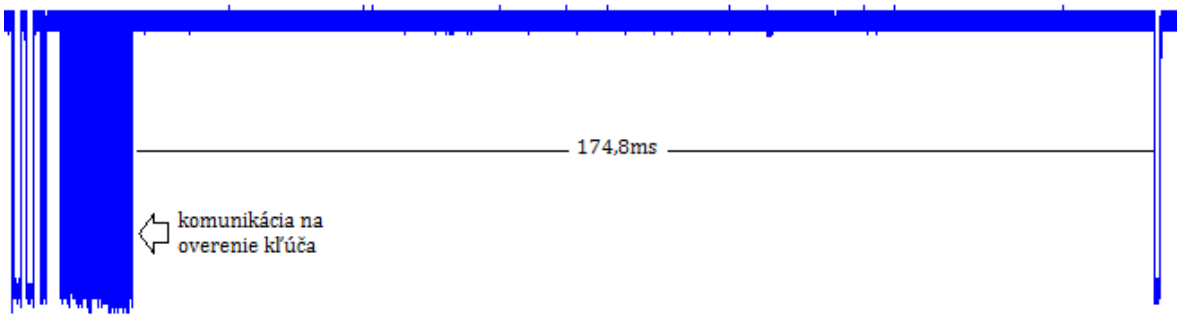
Obr. 4.1: Náhľad na zariadenia UNISIEŤ a programátor-prevodník

Softvérové nastavenia sa robia pomocou programu, ktorý poskytuje predajca. V našom prípade to bola spoločnosť RYS, ktorá poskytuje voľne stiahnuteľný softvér

BBIQ. Softvér aj popis dostupných zariadení je uvedený online [7]. Pri našej práci sme používali BBIQ vo verzii 3.136. Program poskytuje rôznu funkcionalitu, ako napríklad vytváranie novej databázy zariadení, online komunikácia s pripojeným zariadením (cez programátor-prevodník), pridávanie používateľov (a ich DEK kľúčov), zadeľovanie používateľov do skupín a veľa ďalšieho. My sme sa zamerali na aktualizáciu pomocou prenosového čipu, ktorá využíva pamäťový iButton na prepísanie aktuálnej databázy DEK kľúčov v zariadení UNISIEŤ. Podrobný popis inštalácie, vytvorenia databázy a jednotlivých funkcii je dostupný v manuálne na `http://rys.sk/manual/bbiq_user.pdf`.

4.1 Možnosti útoku na operačnú jednotku UNISIEŤ

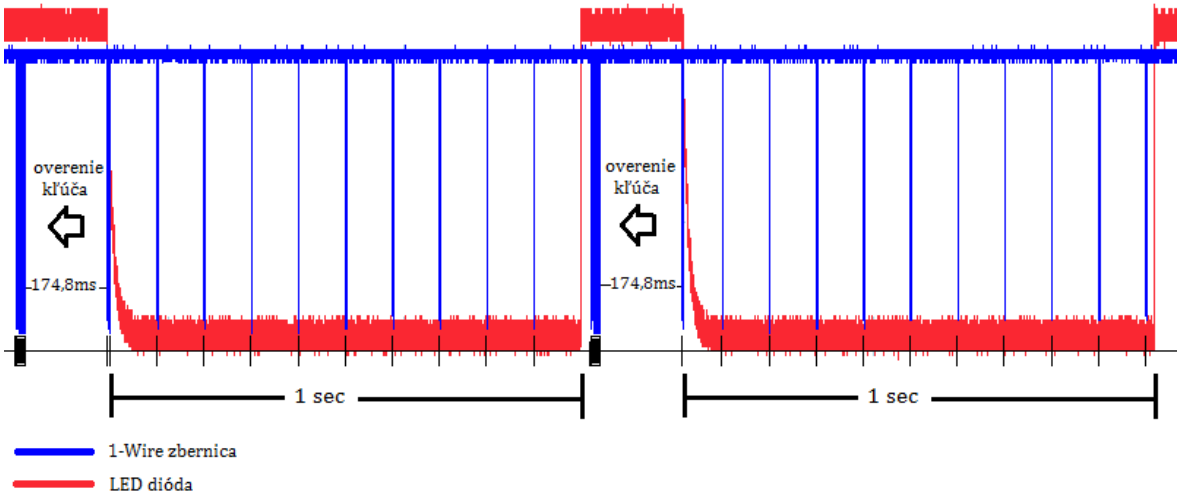
V prvom kroku sme chceli overiť odolnosť operačnej jednotky voči útokom postrannými kanálmi so základným typom iButtonu. Na to sme potrebovali namerať komunikáciu prebiehajúci pri overovaní sériového čísla iButtonu. Najskôr sme chceli overiť, či táto operačná jednotka nebude mať možnosť využitia postranného časového kanálu tvoreného posielaním reset pulzov. Presnejšie sme chceli zistiť, či je rozdiel medzi posielaním reset pulzu po testovaní kľúča uloženého v databáze operačnej jednotky a posielaním kľúča, ktorý sa tam nenachádza. Náš predpoklad bol, že ak by sa po prečítaní posielaný kľúč vyhodnocoval postupne po bajtoch, tak operačná jednotka by pri vyhodnocovaní jednotlivých záznamov strávila menej času, ak by sa daný bajt nenachádzal v žiadnom identifikátore. Ak by sa nachádzal v niektorých identifikátoroch, tak by to malo čas strávený overovaním záznamov predĺžiť. To by malo oddialiť aj posielanie nasledujúceho reset pulzu, ktorý vieme namerať. Ak by sme vedeli odlíšiť čas, pri ktorom bajte bol kľúč zamietnutý a naopak kedy sa tam nejaký s danou hodnotou bajtu nachádzal, tak by sme vedeli zrealizovať útok. Na overenie tejto hypotézy sme vložili do brány kľuč a namerali prebiehajúcu komunikáciu pri overovaní tohoto kľúča. Na zachytenie prebiehajúcej komunikácie sme rovnako ako pri antiklone využili osciloskop Picoscope 6403D. Vždy sme priložili k dotykovej ploche rovnaký iButton s fixným sériovým číslom a postupne sme menili sériové číslo uložené v databáze. V našom prípade sme do operačnej jednotky na začiatku vložili sériové číslo 017C74F016000040 a namerali takto prebiehajúcu komunikáciu. Ukážku takto nameranej komunikácie je vidieť na obrázku 4.2. Čas do nasledujúceho reset pulzu bol 174,8 ms.



Obr. 4.2: Nameraná komunikácia s iButtonom

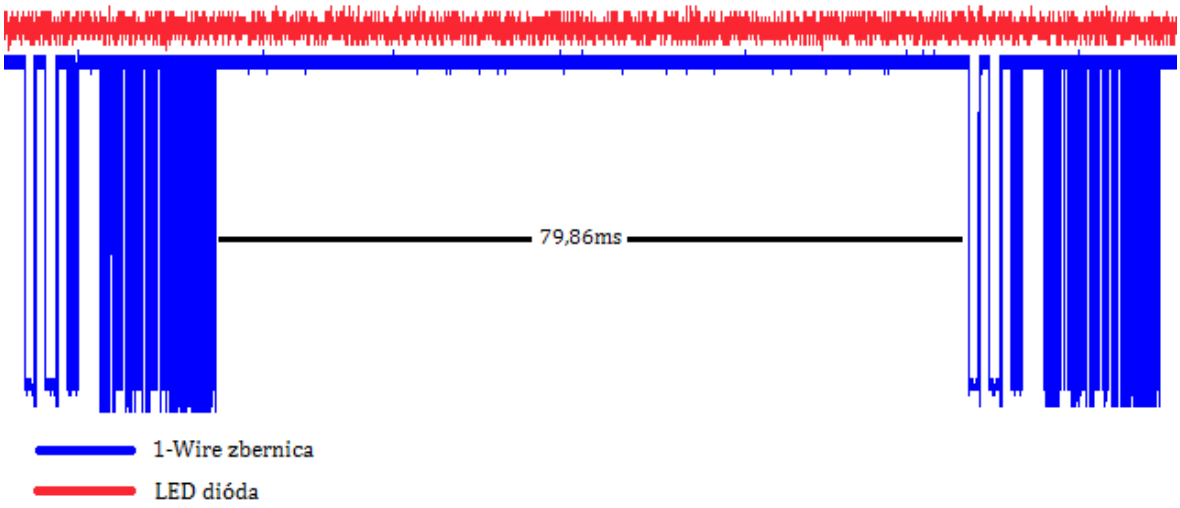
V ďalšom kroku sme vložili do databázy iba kľuč, ktorý sa líšil hneď v prvom posielanom bajte (až po family code, ktorý je nutné dodržať). To musel byť kvôli rovnakému kontrolnému súčtu kľúč 01002FF016000040. Kontrolný súčet sme chceli dodržať pre prípad, že by sa overoval v databáze skôr (takúto skúsenosť sme mali z operačnej jednotky RAK DEK). Čas do najbližšieho reset pulzu pri kľúči líšiacom sa hneď v prvom bajte bol 79,86 ms. Tak sme skúsili opačný extrém a to kľuč , ktorý sa líšil až v poslednom bajte. Do databázy sme tento krát vložili len kľuč 017C74F016005E40. Tu sme zistili, že čas do najbližšieho reset pulzu bol opäť 79,86 ms. Pre istotu sme ešte otestovali aj kľúče, ktoré sa líšili na druhom a treťom bajte, ale dostali sme rovnaký výsledok. Na koniec sme vyskúšali aj kľuč, ktorý mal odlišný posledný bajt a kontrolný súčet pre prípad, že by sa overoval kontrolný súčet v databáze kľúčov skôr. Aj tento pokus dopadol s rovnakým výsledkom. Tým sme uzavreli hypotézu o postrannom časovom kanále využívajúci reset pulz ako nezrealizovateľnú.

Ďalším nápadom bolo inšpirovať sa prácou „Útok na digitálne elektronické kľúče pomocou časového postranného kanálu“ [8] a otestovať svietenie LED diódy počas komunikácie. Využívali sme ten istý iButton aj rovnaké testovacie kľúče ako pri hypotéze o reset pulze. Počas testovania kľúča uloženého aktuálne v databáze prebehla najskôr komunikácia pre čítanie priloženého sériového čísla. Následne, po už uvedených 174,8 ms, prišiel reset pulz , s ktorým začala svietiť aj LED dióda. Tá svietila pri trvale priloženom kľúči 1s aj počas reset pulzov od operačnej jednotky. Tieto reset pulzy sa ignorovali. Po tomto čase zhasla, prišiel nový reset pulz a prebehla celá komunikácia od začiatku. Priebeh komunikácie a svietenia LED diódy je vidieť na obrázku 4.3. Na obrázku z meraní vyzerá svietenie LED diódy ako pokles napätia. V skutočnosti je to však zmena stavu diódy.



Obr. 4.3: Nameraná komunikácia a svietenie LED diódy pri komunikácii s iButtonom uloženým v databáze

Ďalej sme postupovali rovnako ako pri testovaní reset pulzu. Vložili sme do databázy kľúč, ktorý sa líšil v prvom posielanom bajte a namerali sme takto prebiehajúcu komunikáciu aj s LED diódou. Dióda v tomto prípade vôbec nezasvietila. Ako sme už predtým zistili, ak je kľúč zlý, tak operačná jednotka v pravidelných intervaloch 79,86 *ms* skúša novú komunikáciu. Aj pri správnom kľúči sa LED dióda rozsvieti až s ďalším reset pulzom, počas ktorého však ešte nová komunikácia neprebieha. To znamená, že ak pri zlom kľúči testuje ďalším reset pulzom novú hodnotu tak nemá ani priestor kde svietiť. Otestovali sme pre istotu aj ďalšie kľúče, ale pri žiadnom, ktorý sa nenachádzal v databáze, dióda nezasvietila. Ukážku takejto komunikácie je vidieť na obrázku 4.4.

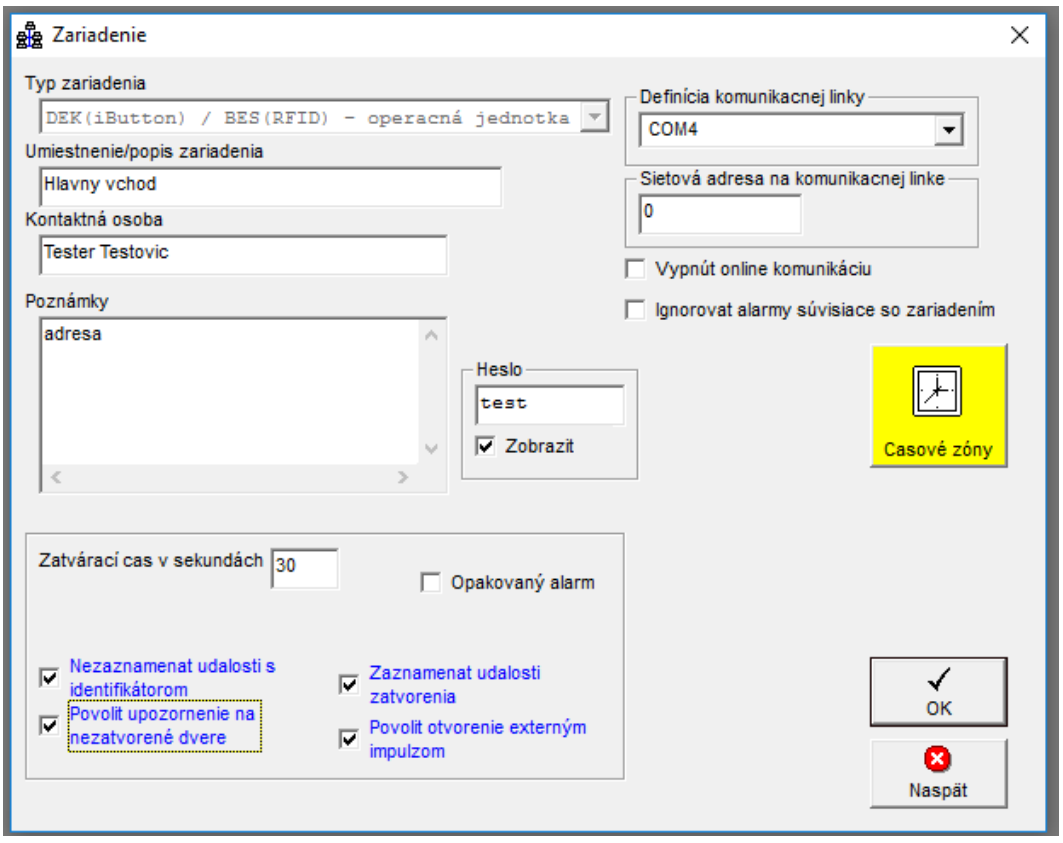


Obr. 4.4: Nameraná komunikácia a svietenie LED diódy pri komunikácii s iButtonom, ktorý nie je v databáze

Tým sme uzavreli naše hypotézy a pokusy o využitie dostupných postranných kanálov ako nezrealizovateľné a začali sme sa sústrediť na softvérovú funkcionalitu, ktorú poskytuje operačná jednotka UNISIEŤ. Rozhodli sme sa zamerať na funkciu programovania prepisovateľného média, ktoré umožňuje aktualizáciu uložených kľúčov v databáze operačnej jednotky.

4.2 Programovanie prenosového čipu

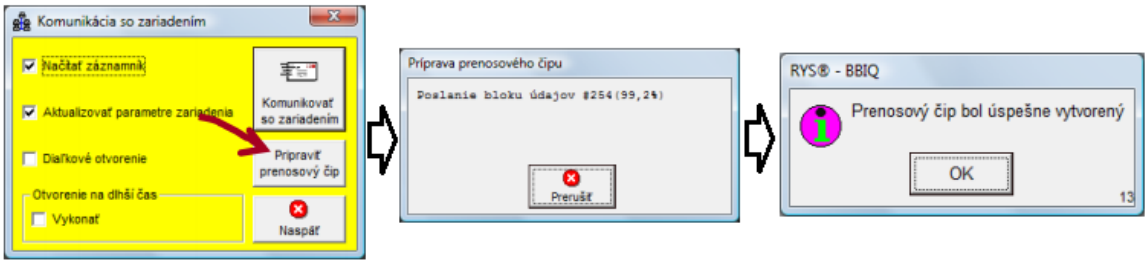
Hlavnou výhodou tejto funkcie je možnosť pridania alebo vymazania identifikátora bez nutnosti online komunikácie s PC na mieste, kde je operačná jednotka osadená. Umožňuje aj aktualizáciu softvérovo nastaviteľných parametrov operačnej jednotky pomocou prenosového čipu. V tomto prípade pomocou iButtonu triedy DS1996. Po úspešnej inštalácii, vytvorení databázy a pridaní operačnej jednotky UNISIEŤ je možné využiť funkciu programovania prenosového čipu. Dôležitá informácia je, že prvé naprogramovanie operačnej jednotky sa musí uskutočniť formou online komunikácie s PC. Súčasťou programovania musí byť zadanie hesla. Heslo môže byť dlhé maximálne desať znakov. Ukážku takéhoto nastavenia priamo v programe BBIQ je vidieť na obrázku 4.5.



Obr. 4.5: Ukážka prvotného nastavenia operačnej jednotky UNISIET s heslom test

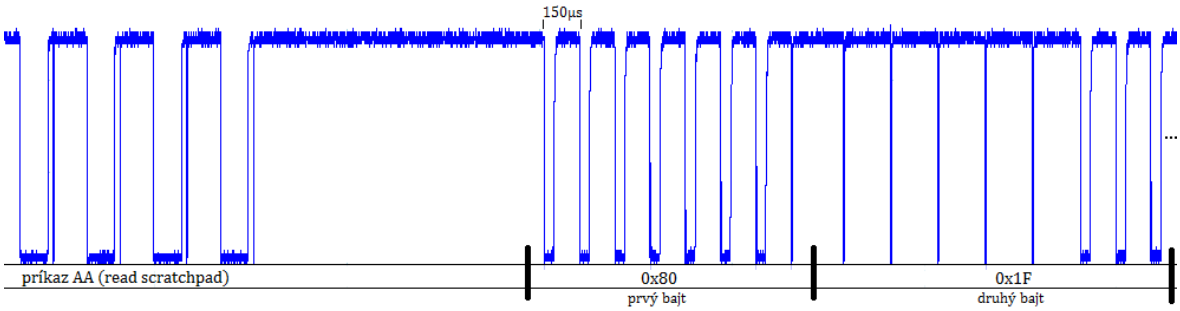
Po úspešnom nastavení máme možnosť všetky ďalšie zmeny nahrať do operačnej

jednotky pomocou prenosového čipu. K tomu stačí mať priložený iButton DS1996 k dotykovej ploche programátora-prevodníka a v programe BBIQ zapnúť funkciu „Pripraviť prenosový čip“. Priebeh prípravy prenosového čipu je vidieť na obrázku 4.6. Takto naprogramovaný iButton je možné zobrať na miesto, kde je operačno-pamäťová jednotka nainštalovaná. Prenosový čip je potrebné priložiť na vstupnú dotykovú plochu brány. Operačná jednotka na základe správneho hesla začne prijímať databázu. Prenos je sprevádzaný zvukovou signalizáciou piezomeniča v operačnej jednotke. Prenos signalizuje prerušovaný tón a ukončenie prenosu trvalý tón.



Obr. 4.6: Príprava prenosového čipu

Nás zaujímal priebeh programovania prenosového čipu a následne jeho čítanie operačnou jednotkou. Programovanie prenosového čipu bolo zaujímavé pre ľahšie zistenie, čo všetko sa nahráva na pamäťový iButton. Analýza nameranej komunikácie však mala niekoľko problémov. Prvým veľkým problémom je, že sme nemali žiadnu informáciu o prenášaných dátach. Napríklad pri skúmaní komunikácie antiklonu sme vedeli, že po príkaze read ROM očakávame sériové číslo iButtonu. V tomto prípade sme vedeli v akom poradí majú prichádzať príkazy na prácu s pamäťou, ale o obsahu dát sme nevedeli nič. Druhý problém bola zložitosť analýzy takto nameraných dát. Ako sme spomenuli v kapitole 2 komunikácia pri čítaní a overovaní dát je urýchlená pre skrátenie komunikácie. To skrátenie je na úkor medzier medzi jednotlivými bajtmi. Pri bežnej komunikácii alebo zápise do vyrovnávacej pamäte je medzi každým posielaným bajtom pauza zhruba 1,1 ms. Pri čítaní z vyrovnávacej alebo reálnej pamäte sa posielajú bity nepretržite každých 150 μs. Náš softvér picoscope nebol schopný spojiť posielané bity do jednotlivých bajtov a teda ani určiť ich hexadecimálnu hodnotu. V tomto prípade sme namerali len komunikáciu ako je vidieť na obrázku 4.7 a bolo nutné vyhodnocovať každý bit na logickú jednotku alebo nulu. Našťastie vyhodnocovanie logických hodnôt sme boli schopný realizovať priamo pozorovaním nameranej komunikácie. Následne sme ich pospájali do bajtov a určili hexadecimálnu hodnotu. V neposlednom rade sme sa snažili identifikovať aké dáta sa posielajú.



Obr. 4.7: Ukážka posielania príkazu a prvých dvoch bajtov pri komunikácii programátora-prevodníka s iButtonom

4.3 Analýza komunikácie pri programovaní prenosového čipu

Pre analýzu komunikácie sme najskôr vypli voliteľné funkcie pre operačnú jednotku UNISIEŤ. Heslo sme zvolili „test“. Následne sme zrušili online komunikáciu, odpojili operačnú jednotku od PC a cez softvér nahrali do databázy tri nami vytvorené sériové čísla iButtonov. Takto upravenú databázu sme chceli aktualizovať pomocou pamäťového iButtonu. Zachytili sme celú komunikáciu, ktorá prebieha medzi programátorom-prevodníkom a priloženým iButtonom. Pri postupnej analýze nameraných dát sme zistili, že komunikácia sa drží štandardu popísaného v prvej kapitole práce. Programátor-prevodník v tejto komunikácii vystupuje ako master. Počas komunikácie takto pripraveného obsahu sa pošlú štyri 32-bajtové bloky dát. Všetky prejdú fázami protokolu reset a presence pulz, príkaz read ROM, poslaním sériového čísla, príkaz write scratchpad s adresovými registrami a dátami, príkaz read scratchpad s adresovými registrami, E/S registrom a dátami, príkaz copy scratchpad s adresovými registrami a E/S registrom. Analýzou jednotlivých dátových blokov sme zistil, že prvý blok obsahuje kľúče uložené v databáze. Kľúče sú oddelené nulovým bajtom. Ukážku takto analyzovanej komunikácie pre posielanie bloku s identifikátormi je vidieť na 4.8. Rozpísané sú len časti týkajúce sa príkazov na prístup k pamäti. Začiatok kľúča je označený jeho poradovým číslom a písmenom Z. Koniec kľúča ohraničuje poradové číslo so znakom K. V databáze sa náchádzajú postupne kľúče: 0000000000000000, 010000000000003D a 017C74F016000040.

- master pošle reset pulzu a obdrží presence pulzu dva krát po sebe
- master pošle príkaz read ROM (0x33)
- iButton pošle svoje identifikačné číslo
- master pošle príkaz na prácu s pamäťou a registre obsahujúce adresu, kam sa majú dáta v pamäti zapísať. Príkaz a registre vyzerajú nasledovne :

```
master: 11110000 .. 0x0F - write scratchpad
master: 00000000 .. register T1  0x00
master: 00000000 .. register T2  0x00
```

- následne master posiela 32-bajtový blok, ktorý sa má zapísať do vyrovnávacej pamäte. Ten vyzerá nasledovne:

```
00000000 .. 0x00   1Z
00000000 .. 0x00
00000000 .. 0x00
00000000 .. 0x00
00000000 .. 0x00
00000000 .. 0x00
00000000 .. 0x00
00000000 .. 0x00   1K
00000000 .. 0x00
10000000 .. 0x01   2Z
00000000 .. 0x00
00000000 .. 0x00
00000000 .. 0x00
00000000 .. 0x00
00000000 .. 0x00
00000000 .. 0x00
00000000 .. 0x00
```

```
10111100 .. 0x3D   2K
00000000 .. 0x00
10000000 .. 0x01   3Z
00111110 .. 0x7C
00101110 .. 0x74
00001111 .. 0xF0
01101000 .. 0x16
00000000 .. 0x00
00000000 .. 0x00
00000010 .. 0x40   3K
00000000 .. 0x00
00000000 .. 0x00
00000000 .. 0x00
00000000 .. 0x00
00000000 .. 0x00
00000000 .. 0x00
```

Obr. 4.8: Zapisovanie bloku obsahujúceho prvé tri identifikátori na pamäťový iButton

Po zápise bloku do vyrovnávacej pamäte nasleduje proces čítania zapísaných dát, ktoré postupuje nasledovne:

- master pošle reset pulzu a obdrží presence pulzu dva krát po sebe
- master pošle príkaz read ROM (0x33)

- iButton pošle svoje identifikačné číslo a očakáva príkaz na prácu s pamäťou a registre obsahujúce adresu, kam sa mali dáta zapísať. Príkaz a registre vyzerali nasledovne:

```
master:   01010101 .. 0xAA - read scratchpad
master:   00000000 .. register T1  0x00
master:   00000000 .. register T2  0x00
iButton:  11111000 .. register E/S 0x1F
```

- iButton postupne posiela 32-bajtový blok obsahujúci dáta zapísane vo vyrovnávacej pamäti
- master opäť pošle reset pulzu a obdrží presence pulzu dva krát po sebe
- iButton opäť pošle svoje identifikačné číslo a očakáva príkaz na prácu s pamäťou, ktorý je popísaný nižšie

```
master:   10101010 .. 0x55 - copy scratchpad
master:   00000000 .. register T1  0x00
master:   00000000 .. register T2  0x00
master:   11111000 .. register E/S 0x1F
```

- na záver master pošle reset pulz a obdrží presence pulz

Ďalší posielaný blok obsahoval 32 nulových bajtov. Tretí blok obsahoval všetky bajty nulové okrem troch. Prvý posielaný bajt mal hodnotu 0x11, dvanásty s hodnotou 0x3C a úplne posledný s hodnotou 0xF0. Posledný posielaný blok je v skrátenej podobe vypísaný v 4.9. Medzi koncom komunikácie pre zapísanie jedného bloku a reset pulzu označujúceho začiatok nového je pauza 230 ms.

```
master: 11110000 .. 0x0F - write scratchpad
master: 00000011 .. register T1 0xC0
master: 11111000 .. register T2 0x1F

00101110 .. 0x74    01111000 .. 0x1E
10100110 .. 0x65    00000000 .. 0x00
11001110 .. 0x73    00000000 .. 0x00
00101110 .. 0x74    00000000 .. 0x00
00000100 .. 0x20    00000000 .. 0x00
00000100 .. 0x20    11111000 .. 0x1F
00000100 .. 0x20    00000000 .. 0x00
00000100 .. 0x20    00000000 .. 0x00
00000100 .. 0x20    00000000 .. 0x00
00000100 .. 0x20    00000000 .. 0x00
00000000 .. 0x00    00000000 .. 0x00
00000000 .. 0x00    00000000 .. 0x00
11000000 .. 0x03    00000000 .. 0x00
00000000 .. 0x00    00000000 .. 0x00
11000000 .. 0x03    00000000 .. 0x00
00000000 .. 0x00    00000000 .. 0x00
```

Obr. 4.9: Blok obsahujúci heslo

Dôležité na tomto bloku je všimnúť si, že sa zapisuje až na koniec pamäte. Presnejšie za adresu 1FC0, ktorá zodpovedá koncu 254-tej stránky v pamäti. Po hlbšej analýze sme zistili, že práve tento blok obsahuje heslo. Ako je vidieť na 4.9 prvých desať posielaných bajtov po prepise z hexadecimálnej sústavy na znaky predstavuje postupnosť „test“ (0x74, 0x65, 0x73, 0x74) doplnenú o šesť medzier (0x20). Ako sme povedali v úvode tejto kapitoly desať znakov je maximálna dĺžka hesla. Nasleduje oddelovač nulovým bajtom a zvyšné bajty predstavujú jednotlivé nastavenia pre UNISIEŤ.

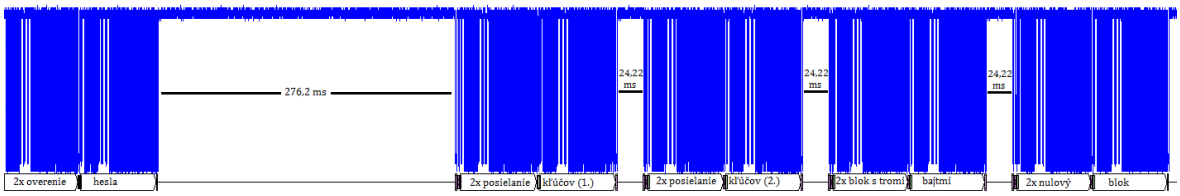
4.3.1 Testovanie softvérových nastavení

Z toho ako vyzeral formát posielania jednotlivých sériových čísel iButtonov nás zaujímali hlavne dve otázky. Či nulový bajt po každom identifikátore slúži pre nastavenia, ktoré sa v programe BBIQ dajú zapnúť pre jednotlivé kľúče, alebo má úlohu len oddeľovača jednotlivých záznamov. Softvér poskytuje možnosť označiť niektoré kľúče za stratené alebo pre návštevy. Nakoniec sme však zistili, že aj keď sme poskúšali zapnúť všetky možnosti, ktoré softvér ponúka, nulový bajt medzi identifikátormi sa nezmenil.

Druhá otázka bola ako sa posielanie vysporiada s tým, že dĺžka kľúča s oddeľovačom nie je deliteľom čísla 32. Ako sme uviedli do vyrovnávacej pamäte je možné zapísať maximálne 32 bajtov na jeden zápis. Keď sme do databázy vložili viac ako tri kľúče zistili sme, že kľúče sa posielajú postupne za sebou bajt po bajte koľko sa zmesti. Ak sa nejaký kľúč nezmestí celý v jednom zápise, jeho identifikačné číslo sa rozdelí a zvyšná časť sa pošle v ďalšom zápise. Žiadne z testovaných nastavení, ktoré sa dajú zapnúť pomocou programu BBIQ neovplyvnilo nulový blok ani blok s obsahom v troch bajtoch, ktoré sa posielali vždy nezmenené. Pri skúšaní rôznych nastavení sme zistili, že bajty v bloku s heslom (rozpísanom na 4.9) medzi bajtmi s hexadecimálnou hodnotou 0x1E a 0x1F zodpovedajú práve týmto voliteľným nastaveniam ako sú nezaznamenávanie identifikátorov, alebo povolenie otvorenia dverí externým impulzom.

4.4 Analýza komunikácie prenosového čipu a operačnej jednotky UNISIEŤ

Po nameraní a analyzovaní dát uložených na pamäťový iButton sme sa zamerali na komunikáciu, ktorá prebieha pri aktualizácii databázy z takto pripraveného pamäťového čipu. Celá komunikácia využíva príkazy read memory. Takže sa jednotlivé bity posielajú nepretržite za sebou a bolo nutné ich vyhodnocovať opäť manuálne. Z nameranej komunikácie medzi operačnou jednotkou a pamäťovým iButtonom sa nám potvrdil predpoklad, že prvý posielaný blok obsahuje heslo. Keďže sa heslo vždy ukladá na koniec pamäte vie ho operačná jednotka adresovať a overiť jeho pravosť. Po prečítaní bloku obsahujúceho heslo nasleduje pauza 276,2 ms. Medzi zvyšnými blokmi čítanými z pamäte iButtonu je pauza 24,22 ms. Čítanie každého bloku z pamäte pozostáva z postupnosti reset pulz, presence pulz, príkazu read ROM, prečítania identifikačného čísla, poslanie príkazu na prácu s pamäťou (v tomto prípade read memory), adresový register TA1, adresový register TA2, čítanie 32 bajtov z pamäte iButtonu. Každý blok sa číta dva krát po sebe. Keďže napriek odporúčaniam zo štandardu sa neukladá žiaden kontrolný súčet pre jednotlivé bloky predpokladáme, že to slúži na overenie integrity prečítaných dát. Po akceptovaní hesla sa posielajú postupne kľúče uložené v databáze, blok obsahujúci tri nenulové bajty (spomenuté pri nahrávaní dát na iButton) a blok obsahujúci samé nuly. Namerané dáta z priebehu aktualizácie operačnej jednotky je vidieť na obrázku 4.10. Po dvojitom prečítaní každého bloku sa vždy pošle ešte jeden reset a presence pulz.

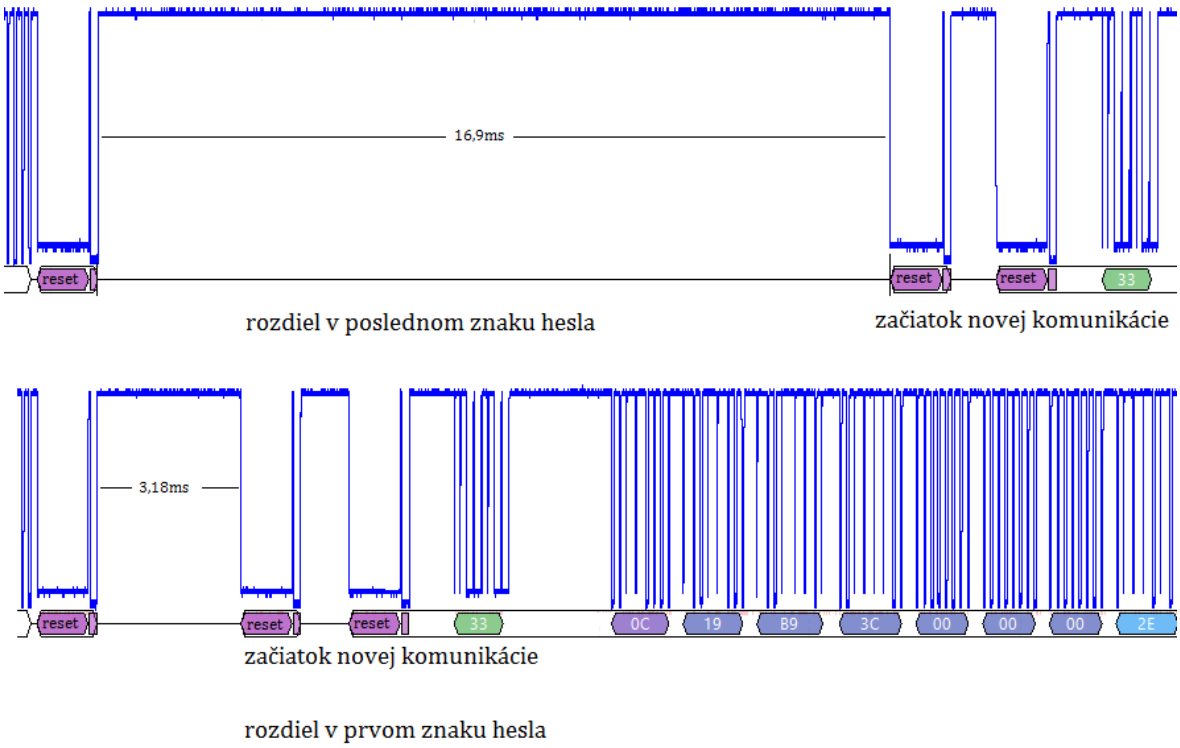


Obr. 4.10: Oddialený pohľad na priebeh aktualizácie operačnej jednotky

Po nameraní platnej aktualizácie nás zaujímalo ako operačná jednotka reaguje v prípade, že pamäťový iButton pošle neplatné heslo. Preto sme skúsili v softvéri nechať všetky nastavenia rovnaké, ale zmenili heslo na „zle“. S takto upravenými heslom sme prepísali pamäťový iButton. Zápis na pamäťový iButton vyzeral identicky okrem bloku obsahujúceho heslo kde sa zapísali bajty 0x7A (’z’), 0x6C (’l’), 0x65 (’e’), sedem krát 0x20 (medzera) a zvyšných 22 bajtov vyzeralo rovnako ako predtým. Namerali sme prebiehajúcu komunikáciu medzi operačnou jednotkou a takto upraveným iButtonom. Ukázalo sa, že pri trvalo priloženom iButtone, operačná jednotka naďalej prečíta blok obsahujúci heslo dva krát po sebe a zistí, že heslo je zlé. To sa prejaví tým, že miesto čakania 276,2 ms (počas ktorých už zrejme prebieha aktualizácia nastavení uvedených v bajtoch za heslom) posiela nový reset pulz hneď po 3,18 ms a proces čítania s overovaním hesla prebieha od začiatku. Keďže čas, po ktorom brána skúša nové heslo je dosť krátky, zaujímalo nás, či je to spôsobené dĺžkou hesla. Ak by bolo heslo overované po znakoch a menil by sa čas, po ktorom operačná jednotka pošle nový reset pulz, tak by sme vedeli takýmto skúšaním zistiť heslo dostatočne rýchlo.

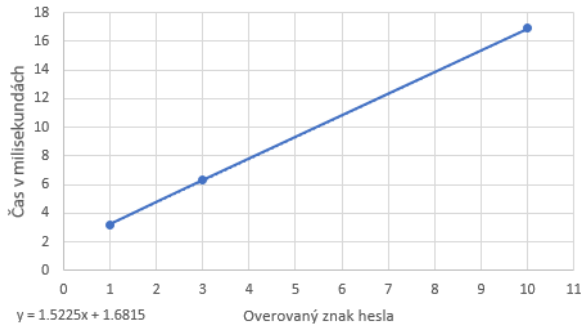
4.4.1 Testovanie závilosti času overovania hesla od dĺžky jeho zhody

Pripojili sme bránu online cez programátor-prevodník (pretože zmena hesla v operačnej jednotky je možná len počas online komunikácie) a nastavili jej heslo maximálnej dĺžky. Následnej sme bránu odpojili a pripravili pamäťový iButton, ktorý však bol aktualizovaný s heslom líšiacim sa v prvom znaku hesla. Po nameraní prebiehajúcej komunikácie sme zistili, že operačná jednotka posiela reset pulz a skúša nové heslo opäť po 3,18 ms. To znamená, že buď posiela operačná jednotka reset pulz konštante po tomto čase, alebo rovnako ako pri predchádzajúcom meraní bol čas ovplyvnený rozdielom v prvom znaku hesla. Preto sme vyskúšali pripraviť pamäťový iButton s heslom, ktoré sa líšilo až v poslednom znaku. Očakávaný reset pulz na testovanie nového hesla v tomto prípade prišiel až po 16,9 ms. Obe merania je vidieť na obrázku 4.11.



Obr. 4.11: Porovnanie času do nasledujúceho reset pulzu v závislosti od znaku v hesle

Z toho sme začali predpokladať časovú závilosť nasledujúceho reset pulzu voči znaku, ktorý sa líši medzi posielaným heslom a tým uloženým v operačnej jednotke. Otestovali sme závilosť aj pri kratších heslách. Konkrétne sme v operačnej jednotke mali nastavené pôvodné heslo „test“ a na pamäťovom iButtone heslo „temp“. Čiže rozdiel nastal v treťom znaku. Čas do reset pulzu pri takejto komunikácii prišiel po 6,28 *ms*. Z toho sme usúdili, že môže existovať lineárna závilosť medzi odlišným znakom hesla a časom do nasledujúceho reset pulzu, ktorý predstavuje začiatok nového overovania hesla. Závilosť je znázornená na obrázku 4.12. Predpokladáme, že čas do nasledujúceho reset pulzu narastá po zhruba 1,52 *ms* za každý správny znak hesla.



Poradie znaku v hesle	Čas v ms
1.	3,18
3.	6,28
10.	16,9

Obr. 4.12: Závsilosť znaku hesla a času do nasledujúceho reset pulzu

4.5 Odhad útoku na operačnú jednotku UNISIEŤ

Operačná jednotka UNISIEŤ testuje heslo nepretržite. Čas, po ktorom pošle reset pulz pre novú komunikáciu prezrádza koľko znakov posielaného hesla je správnych. Tým sa nám naskytuje možnosť realizácie útoku využívajúci reset pulz ako postranný časový kanál. Komunikácia, ktorá dva krát prečíta blok pamäte iButtonu obsahujúci heslo trvá 150 ms. Zistili sme, že aj keď sa heslo líši až na poslednom desiatom znaku, tak čas do novej komunikácie je len zhruba 17 ms. Program BBIQ, ktorým sa realizuje online komunikácia s PC, podporuje pri zadávaní hesla skoro všetky tlačiteľné symboly. Keby sme uvažovali všetky, dostaneme 224 rôznych možností pre jeden znak hesla. Prehľad všetkých tlačiteľných znakov aj s ich hexadecimálnou hodnotou je uvedený napríklad na `http://www.ascii-code.com/`. Priebeh celej komunikácie a overenie hesla v najhoršom prípade trvá 167 ms. Keby sme vyskúšali všetky znaky na poslednom mieste trvá nám to 37,41 s. Z odhadu nárastu času do nasledujúceho reset pulzu v závilosti od správneho znaku by sa priebeh celej komunikácie dal v najhoršom prípade odhadnúť na:

$$\sum_{n=1}^{10}(150 + (n * 1{,}52 + 1{,}68)) * 223 = 356\,889{,}2 \text{ ms} = 5{,}945 \text{ min.}$$

Zistenie hesla nám umožňuje prepísať databázu identifikačných čísel aj zmenu nastavení operačnej jednotky. Jedným z nich je napríklad zakázanie logovania záznamov o použitých identifikátoroch.

4.5.1 Spôsob realizácie útoku

Z praktického hľadiska by programovanie pamäťového iButtonu s rôznym heslom a jeho následné testovanie bolo veľmi neefektívne. Preto by bolo výhodné využiť Arduino alebo podobné zariadenie na simulovanie pamäťového iButtonu. Stačí si vybrať identifikačné číslo iButtonu, ktorý musí mať family code 0x0C a platný kontrolný súčet. Podľa zisteného priebehu komunikácie vieme, že potrebujeme po reakcii na prichádzajúci reset pulz najskôr odsimulovať proces posielania identifikačného čísla po príkaze read ROM, ktorý sme v práci podrobne popísali. Následne obdržíme príkaz na čítanie pamäťového bloku obsahujúceho heslo. Na začiatku by mohlo byť optimálne testovať heslo obsahujúceho desať medzier (keďže vieme, že reálne heslo je doplnené do maximálnej dĺžky medzerami). Za heslom by v bloku mohli nasledovať bajty so základnými nastaveniami ako sme uviedli v analýze nameraných dát na 4.9. Po poslaní celého bloku dva krát očakávame reset pulz. Odpovieme presence pulzom a začneme počítať čas, ktorý uplynie po nasledujúci reset pulz. Podľa uplynutého času by sme vedeli odlíšiť či je zlý už prvý znak alebo až niektorí nasledujúci. Následne inkrementovať bajt

obsahujúci nesprávny znak a testovať nové heslo. Ak čas do nasledujúceho reset pulzu prekročí čas 200 ms, tak to môžeme považovať za zistenie hesla a operačná jednotka by mala chcieť čítať prvý blok pamäte, ktorý obsahuje identifikačné čísla iButtonov v databáze. Ako sme podrobne popísali v analýze bloku s identifikátormi na 4.8, môžeme uviesť postupne jednotlivé identifikátory, ktoré chceme vložiť do databázy, a vzájomne ich oddeliť nulovým bajtom. Po poslaní blokov s identifikátormi brána očakáva dva bloky, ktoré sme popísali a sú nemenné. Na tento útok postačujú funkcie, ktoré sme popísali v práci pri simulácii základného iButtonu pre zariadenie antiklon. Jediné čo je nutné doplniť je akceptovanie príkazu na čítanie pamäte, predpríprava blokov, ktoré sa posielajú a postupné aktualizovanie hesla. Všetky merania, z ktorých sme vychádzali sú dostupné v prílohe práce.

Takýto útok by sa ešte dal vylepšiť o využitie určitého slovníku možných hesiel. Heslá pre operačnú jednotku zadávajú používateľia alebo správcovia prístupových systémov. Keďže program neposkytuje generátor náhodného hesla (ktorý by bol veľmi vhodný, pretože používateľ si heslo nepotrebuje pamätať ani ho nikde viac nezadáva), tak je rozumným predpokladom, že heslá budú bežné slová neobsahujúce žiadne špecialne znaky. Nami popísaný útok predpokladá všetky znaky aj možnosti hesiel. Zmenšenie testovanej abecedy aj využitie pravdepodobných hesiel môžu útok jedine výrazne urýchliť.

Operačná jednotka by mala mať možnosť zapamätať si, že testované heslo bolo nesprávne. Mohla by oddialiť možnosť testovania ďalšieho. Po prečítaní identifikačného čísla vie, či sa jedná o bežný iButton s adresou, alebo pamäťový s možnou aktualizáciou. Po nesprávnom hesle by mohla na nejaký čas postupne odďaľovať testovanie ďalšieho pamäťového iButtonu. Tým by operačná jednotka zabránila alebo aspoň značne predĺžila čas potrebný pre útok.

Záver

V prvej kapitole sme popísali fungovanie 1-Wire protokolu, ktorý využívajú dotykové elektronické kľúče iButtony. Následne sme uviedli všeobecný popis iButtonov a zamerali sme sa na dve triedy, ktoré využívame v ďalších častiach práce. Jedným z prínosov práce má byť poukázanie na zraniteľnosti spôsobené implementáciou protokolu alebo fungovaním zariadení.

Jedným z cieľov práce bolo preverenie spoľahlivosti zariadenia antiklon pri odhaľovaní kópii iButtonov. V práci uvádzame nameranú komunikáciu, príkazy testované antiklonom a porovnanie správania originálu a klonu. Popisujeme spôsob implementácie programu, ktorý simuluje iButton a zariadenie antiklon ho nie je schopné odlíšiť. Spôsob simulácie je popísaný v práci a samotný program je uvedený v prílohe práce. Simuláciou originálneho iButtonu poukazujeme na slabiny antiklonu, ktoré by bolo vhodné odstrániť. Zároveň rozoberáme problém klonovania iButtonov, ktorý prakticky nemôže mať 100%-tné riešenie.

V ďalšej časti práce sa venujeme operačnej jednotke UNISIEŤ, ktorá podporuje pokročilejšiu funkcionalitu ako RAK DEK analyzovaný v práci [8]. Zamerali sme sa na hľadanie zraniteľností využívajúcich postranné kanály, ktoré sú dostupné cez dotykovú plochu zariadenia. Najskôr sme v snahe o nadviazanie na prácu „Útok na digitálne elektronické kľúče pomocou časového postranného kanálu“ [8] testovali komunikáciu so základnou triedou iButtonov. Otestovali sme možnosť využitia časového postranného kanálu, ktorý využíval prichádzajúci reset pulz po priložení identifikátora nenachádzajúceho sa v databáze. Keď sme zistili, že takýto útok nie je realizovateľný preverili sme aj svietenie LED diódy, ktorá bola v spomínanej práci kľúčová. Operačná jednotka UNISIEŤ má správanie diódy ošetrené a útok nebol realizovateľný. Následne sme sa venovali pokročilejšej funkcionalite, ktorú táto operačná jednotka poskytuje. Zamerali sme sa na možnosť aktualizácie databázy a softvérových nastavení pomocou pamäťového iButtonu. Po nameraní a analýze komunikácie sme prišli na to, že jediným bezpečnostným prvkom takejto aktualizácie je prednastaviteľné heslo. Zistili sme, že operačná jednotka pri komunikácii s pamäťovým iButtonom, ktorý obsahuje neplatné heslo, testuje nové heslo po čase závislom od znaku kde sa heslá líšili. To sa dá využiť ako postranný časový kanál na zistenie správneho hesla. Po získaní správneho hesla sme schopní prepísať celú databázu identifikátorov v operačnej jednotke.

Vieme tiež zmeniť niektoré softvérové nastavenia, ako napríklad zrušenie logovania prístupov k operačnej jednotke. Dôležitá je doba trvania takéhoto útoku. Podľa horného odhadu vieme takýmto spôsobom zistiť heslo operačnej jednotky za menej ako šesť minút. Toto považujeme za vážny nedostatok a naplnenie nášho cieľa identifikovať zraniteľnosť operačnej jednotky UNISIEŤ. Popisujeme možné optimalizácie na urýchlenie útoku a zároveň možnosti ako opraviť operačnú jednotku, aby zabraňovala takýmto zraniteľnostiam.

Práca by mala motivovať k preverovaniu ďalších bežne používaných zariadení, ktoré slúžia pre zvýšenie bezpečnosti. Poskytuje prehľad slabín, ktoré sa oplatí overovať a možnosť rozšírenia na ďalšie zariadenia využívajúce tieto technológie. Program uvedený v prílohe je základom pre programovanie ľubovoľnej 1-Wire funkcionality. To je možné využiť na doprogramovanie oficiálnej knižnice pre zariadenie Arduino, ktorá momentálne nie je dostupná.

Literatúra

[1] Arduino. Arduino uno & genuino uno, 2017. https://www.arduino.cc/en/main/arduinoBoardUno.

[2] Brian Hindman. *Book of ds19xx ibutton standards*. Maxim Integrated Products, Inc., 2006.

[3] Maxim Integrated. DS1990A, serial number iButton datasheet, 2008. https://datasheets.maximintegrated.com/en/ds/DS1990A.pdf.

[4] Maxim Integrated. DS1996, 64kb memory iButton datasheet, 2009. https://datasheets.maximintegrated.com/en/ds/DS1996.pdf.

[5] Maxim Integrated Products, Inc. Understanding and using cyclic redundancy checks with maxim 1-Wire and iButton products, Mar 29, 2001. https://www.maximintegrated.com/en/app-notes/index.mvp/id/27.

[6] Paul Stoffregen. Dallas semiconductor's 1-wire protocol, 2017. https://www.pjrc.com/teensy/td_libs_OneWire.html.

[7] RYS. BBIQ a zariadenia od spoločnosti RYS, Citované 10.2.2017. http://rys.sk/.

[8] Tomáš Paulík. Útok na digitálne elektronické kľúče pomocou časového postranného kanálu, 2015. http://www.dcs.fmph.uniba.sk/bakalarky/obhajene/getfile.php/tomas_paulik_bc.pdf?id=280&fid=536&type=application%2Fpdf.