Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Útok na digitálne elektronické kľúče pomocou časového postranného kanálu

bakalárska práca

2015 Tomáš Paulík

Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Útok na digitálne elektronické kľúče pomocou časového postranného kanálu

bakalárska práca

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

Bratislava, 2015 Tomáš Paulík

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Tomáš Paulík
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Útok na digitálne elektronické kľúče pomocou časového postranného kanálu.
Timing Attack on iButton Digital Electronic Keys.

Cieľ: Cieľom práce je preskúmať bezpečnosť brán používajúcich digitálne elektronické kľúče na báze iButton čipov so zameraním na analýzu možností poskytovaných časovým postranným kanálom. Pre reálne testovanie zraniteľnosti bude použitá platforma Arduino (alebo podobná, ale s výkonnejším CPU).

Vedúci: RNDr. Richard Ostertág, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: doc. RNDr. Daniel Olejár, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 04.11.2014

Dátum schválenia: 25.11.2014 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie:

Ďakujem vedúcemu bakalárskej práce RNDr. Richardovi Ostertágovi PhD. za odbornú pomoc pri jej vypracovávaní.

Abstrakt

Práca preveruje bezpečnosť prístupových systémov, ktoré využívajú elektronické kľúče iButton. Konkrétne sa venujeme prístupovému systému RAK DEK. Pomocou zariadenia Arduino UNO simulujeme iButton pri komunikácii s RAK DEK. V práci poukazujeme na slabinu v bezpečnosti zariadenia RAK DEK. Zaoberáme sa spôsobom objavenia slabiny a realizáciou úspešného útoku s využitím časového postranného kanálu. Uvádzame využité zariadenia a program, ktorým útok realizujeme.

Kľúčové slová: iButton, Arduino, RAK DEK, časový postranný kanál

Abstract

This thesis assesses the security of electronic access systems based on iButton keys. Specifically we target the RAK DEK access system. We use the Arduino UNO device to simulate an iButton key during communication with the RAK DEK device. We point out a weakness found in the security of the RAK DEK device and address the methods we used to find it and the realization of a successful timing attack.

Keywords: iButton, Arduino, RAK DEK, timming attack

Obsah

Zoznam obrázkov

Zoznam funkcií

Úvod

Čoraz viac obytných domov využíva elektronické kľúče ako alternatívu k bežnému kľúču. Elektronické kľúče majú výhody nielen pre obyvateľov budovy, ale aj pre tých, ktorí prístupové systémy využívajúce elektronické kľúče obstarávajú. Pre používateľov elektronických kľúčov je výhodné nahratie jedného kľúča do viacerých prístupových systémov. Tým získajú jeden kľúč k viacerým prístupovým systémom naraz. Ďalšou z výhod je úspora času pri otváraní prístupových systémov. Jednou z veľkých výhod pre obstarávateľov prístupových systémov je zabezpečovanie nových kľúčov alebo riešenie straty kľúča. Pri bežných zámkoch je pri strate kľúča zvyčajne nutná výmena zámku a rozmnoženie potrebného množstva kľúčov. Pri elektronickom zámku stačí stratený kľúč vymazať z pamäti prístupového systému a majiteľovi priradiť nové unikátne číslo. Ako elektronické kľúče sa využívajú iButtony a jedným z prístupových systémov je zariadenie RAK DEK.

iButtony ako také majú množstvo rôznych využití. Napríklad meranie teploty, vlhkosti alebo času. Vedia slúžiť aj na prenos dát. iButtony slúžiace na meranie teploty sa využívajú často v potravinárskom priemysle. Konkrétne pri potravinách chúlostivých na teploty pri preprave alebo skladovaní. Stačí nastaviť, ako často má iButton snímať teplotu a pribaliť ho k prepravovaným potravinám. Pri prevzatí potravín treba načítať namerané dáta a pozrieť, či teploty neprekročili teplotný interval potrebný na zachovanie potravín bez poškodenia. V našej práci sa orientujeme výhradne na jednu podtriedu iButtonov a to DS1990A, ktorá je najjednoduchšia a slúži v tomto prípade na verifikáciu unikátnym identifikátorom.

V prvej kapitole práce si vysvetlíme, čo je to iButton. Vysvetlíme fungovanie iButtonu a protokol, ktorý využíva na komunikáciu s prístupovým systémom. Podrobne rozoberieme celý priebeh komunikácie zariadení aj z pohľadu oficiálneho štandardu aj s uvedením nami nameraných hodnôt. Všetky schématické obrázky a

texty uvedené v prvej kapitole vychádzajú z oficiálneho štandardu [5].

V druhej kapitole práce uvedieme a vysvetlíme jednotlivé možnosti útoku na zariadenia. Cieľom kapitoly je vytvoriť prehľad možností a odôvodniť, prečo sme sa rozhodli venovať práve útoku s využitím časového postranného kanálu.

V tretej kapitole si podrobnejšie popíšeme zariadenia, ktoré sme využívali. Popíšeme ich vlastnosti a vysvetlíme, kde konkrétne sme ich pri praktickej časti práce potrebovali použiť. V závere tejto časti popisujeme spôsob, akým sme zariadenia pri praktickej časti práce zapojili. Uvádzame schému zapojenia s jej slovným popisom. Obrázky využité v tejto časti práce sú prevzaté z oficiálnych stránok zariadení a schéma zapojenia zariadení je vyhotovená pomocou voľne dostupného softvéru Fritzing [4].

Štvrtá kapitola sa venuje hodnotám a zisteniam, ktoré sme zistili pri meraniach. Podrobnejšie sa venujeme nameraným hodnotám z osciloskopu, ktoré nám umožnili skúmať priebeh komunikácie iButtonu s prístupovým systémom RAK DEK. V podkapitole sa venujeme potrebnému nastaveniu fotoodporu.

Piata kapitola práce sa zaoberá implementáciou kódu, ktorým simulujeme iButton. Popisujeme generovanie jednotlivých identifikátorov a spôsob, akým postupujeme v závislosti od času stráveného vyhodnocovaním identifikátora.

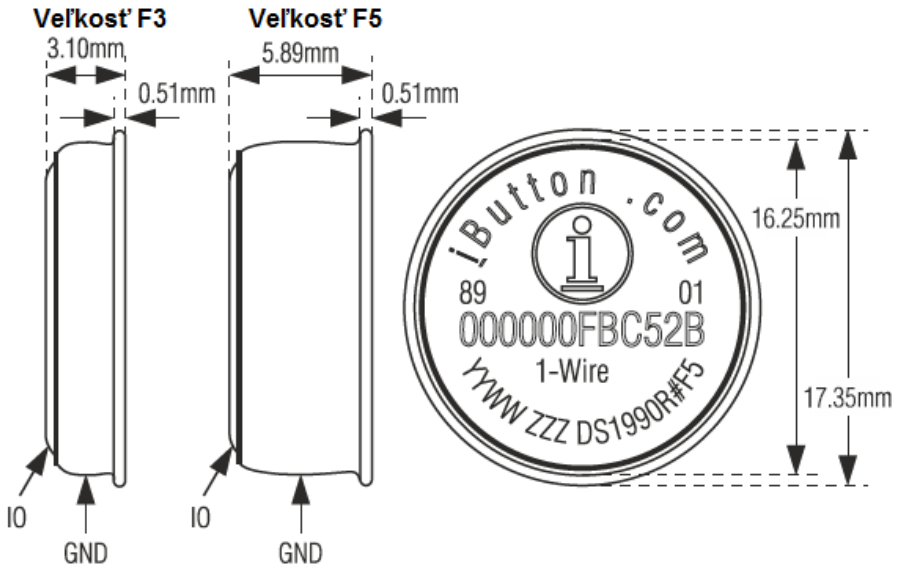
Kapitola 1

iButton

Zariadenie iButton je počítačový čip uzatvorený v 16 mm hrubej nehrdzavejúcej oceli v tvare gombíka. Vďaka svojmu unikátnemu a odolnému obalu je možné iButton prenášať kdekoľvek. Naviac, vydrží teploty v rozmedzí -40 °C až +85 °C. Kedže je rozmerovo malý, je možné ho osadiť do predmetov bežného použitia ako je prsteň alebo prívesok na kľúče.

1.1 Základné informácie

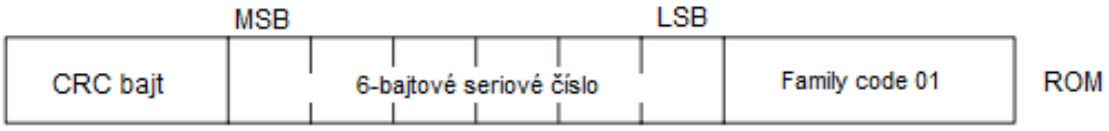
Každý oficiálny iButton má na obale unikátnu, laserom vypálenú, 8 bajtov dlhú adresu, ktorá sa v našom prípade využíva ako kľúč. V ďalšej časti textu sa budeme na unikátnych 6 bajtov adresy odkazovať ako na identifikátor kľúča. Znázornenie iButtonu je možné vidieť na obrázku 1.1. Obal iButtonu sa využíva ako elektronické komunikačné rozhranie. Konkrétne obal obsahuje dátový konektor a uzemnenie. Obe časti sú prepojené s čipom, ktorý sa nachádza vnútri zariadenia. Dátový konektor (IO) je na vrchnej časti obalu a uzemnenie (GND) sa nachádza po okrajoch. Tieto dva kontakty sú oddelené polypropylénovou priehradkou. Spotreba energie je minimálna. Energiu potrebnú na komunikáciu iButton paraziticky čerpá z dátovej zbernice zariadenia, ku ktorému sa pripojí. Na komunikáciu s prístupovým systémom iButton využíva 1-Wire protokol.



Obr. 1.1: DS1990A iButton so sériovým číslom

1.2 DS1990A iButton so sériovým číslom

DS1990A je najjednoduchším typom iButtonu. Priamo z výroby má predprogramovanú ROM pamäť. Nepotrebuje žiadnu energiu na udržanie dát a skoro žiadnu energiu na činnosť. Na obrázku 1.2 je vidieť usporiadanie dát na iButtone typu DS1990A. Prvý bajt, ktorý sa posiela z ROM, je family code. Pre celú triedu DS1990A je to číslo 01. Po family code nasleduje jedinečné sériové číslo posielané od najmenej významných bitov. Posledný bajt obsahuje kontrolu cyklickým kódom (CRC). Ak sa CRC vypočítané masterom pri čítaní zhoduje s CRC načítaným z iButtonu, tak komunikácia prebehla v poriadku.

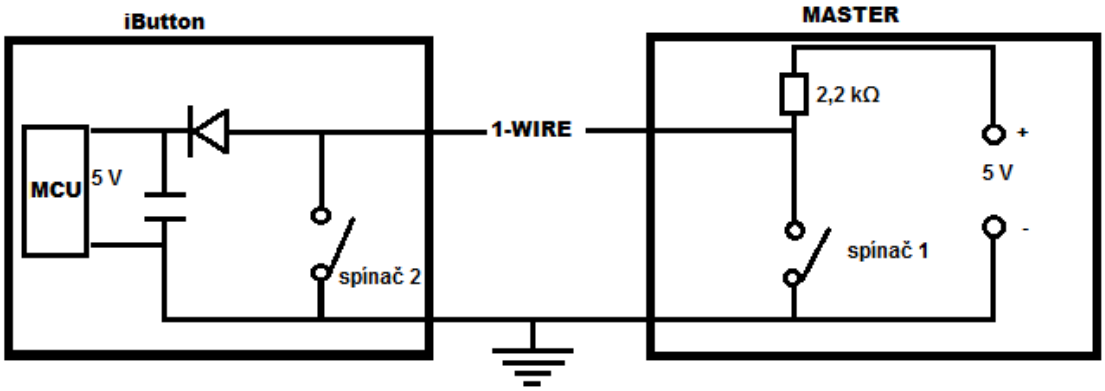


Obr. 1.2: Usporiadanie dát iButtonu typu DS1990A

1.3 1-Wire protokol

Rozhranie 1-Wire protokolu má dve rýchlosti komunikácie: štandardný mód s rýchlosťou 16 kbps a overdrive mód s rýchlosťou 142 kbps. Sériový prenos sa uskutočňuje cez half-duplex (obojsmerne, ale naraz iba jedným smerom) cez diskrétne definované

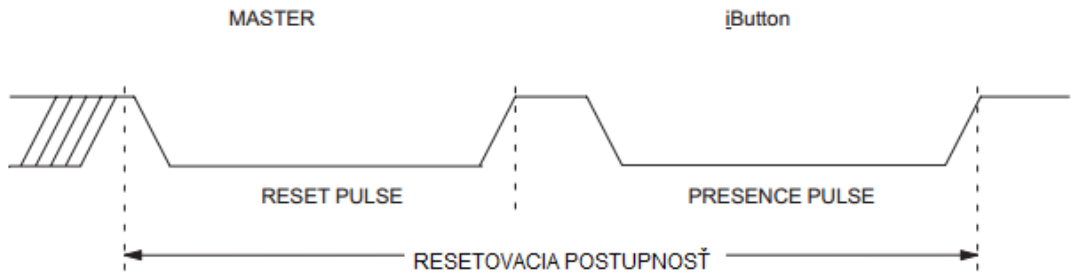
časové úseky. Komunikácia prebieha formou master-slave. Zariadenie, ku ktorému sa iButton pripojí, sa správa ako master. Master iniciuje komunikáciu poslaním príkazu zariadeniu, ktoré vystupuje ako slave (v tomto prípade iButtonu). Príkazy a dáta sa posielajú bit po bite začínajúc od najmenej významných. Synchronizácia medzi zariadeniami sa uskutočnuje tým, že master spojí dátový vodič so zemou na určitý čas. Na obrázku 1.3 je načrtnutá schéma zapojenia zariadení. Spojenie dátového vodiča so zemou sa dá v schéme predstaviť ako spojenie spínača 1. V tom okamihu sa prúd nevedie cez 1-wire spojenie, ale je priamo uzemnený. To sa v komunikácii prejaví znížením napätia na 0 V. Ak potrebuje iButton stiahnuť napätie na 0 V, tak zopne spínač 2. Kondenzátor udrží potrebnú eneriu na fungovanie MCU počas zníženého napätia. Táto metóda komunikácie v časových úsekoch sa dá prerušiť, keďže každý úsek je časovaný nezávisle. Prenos údajov nemôže prebiehať bez spojenia zariadení. Už niekoľko milisekúnd po spojení iButton zaregistruje znížené napätie a dá masterovi vedieť o svojej prítomnosti. Master vždy začne komunikáciu poslaním reset pulse. Následne očakáva odpoveď v podobne presence pulse.



Obr. 1.3: Schéma zapojenia iButtonu a mastera

Master si kvôli synchronizácii môže presence pulz vyžiadať poslaním reset pulzu aj mimo začiatku komunikácie. Komunikáciu pomocou reset a presence pulzu môžeme vidiet na obrázku 1.4 a podrobnejšie sa jej budeme venovať v ďalšej časti tejto kapitoly.

Po presence pulze iButton očakáva príkaz. Ľubovoľný príkaz sa skladá zo zreťazenia 8 časových úsekov na prenos bitu s hodnotou jedna alebo nula. Kedže rodina DS1990 iButtonov je jedna z najjednoduchších, tak podporuje iba ROM príkazy ako



Obr. 1.4: Komunikácia pomocou presence a reset pulse

sú read, skip, match alebo search. Iné zariadenia podporujú aj príkazy s prístupom k pamäti ako Write alebo Copy. Po prečítaní a vyhodnotení príkazu iButton začne posielať odpoveď na príkaz vo vymedzených časových úsekoch. iButton vystupuje vždy ako slave, takže necháva začiatok časového úseku na masterovi. Master iniciuje začiatok časového úseku na prečítanie dát a iButton zareaguje podľa toho, aký bit potrebuje poslať.

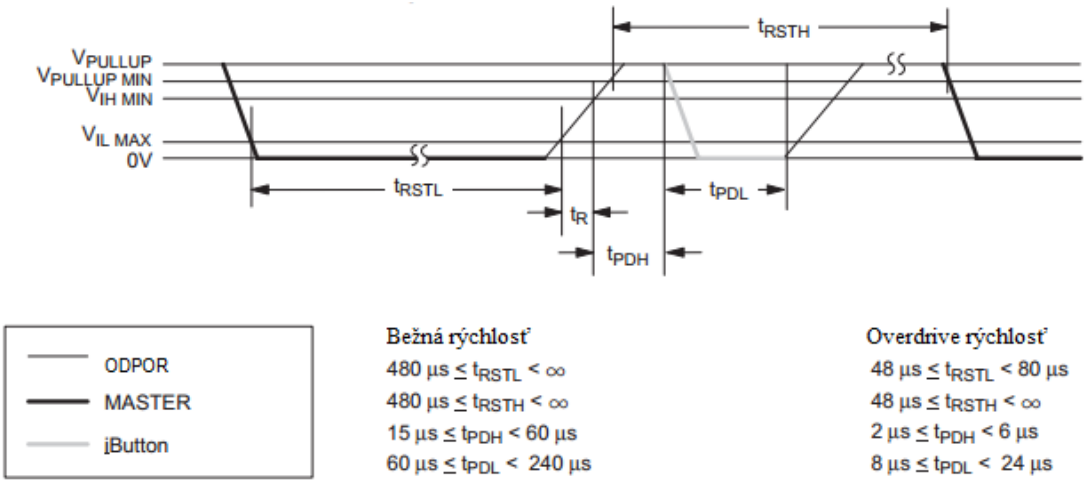
1.4 Časovanie úsekov

Logika za načasovaním umožňuje meranie a generovanie digitálnych impulzov rôznej dĺžky. Dáta sa pri komunikácii prenášajú asynchrónne po bitoch cez half-duplex. Môžu byť rôzne interpretované. Napríklad ako príkazy, ktoré sa porovnávajú s informáciami uloženými v iButtone a na ich základe sa rozhodne o ďalšej činnosti. iButton využíva spádovú hranu v napätí na synchronizáciu vnútorných obvodov. Načasovanie pri jednotlivých operáciách je robené s ohľadom na časové úseky.

1.4.1 Reset a Presence pulz

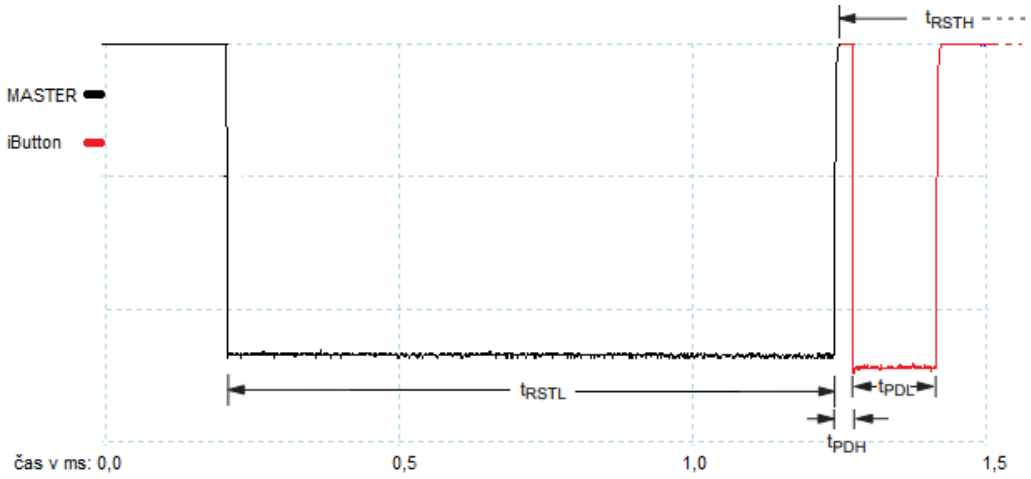
Ako sme si už spomenuli, 1-Wire protokol využíva reset pulz. Ten je definovaný ako zníženie napätia (na obrázku 1.5 značený ako t_{RSTL}) po dobu aspoň 480 μs. Hneď po znížení nasleduje zvýšenie napätia (na obrázku 1.5 značený ako t_{RSTH}) na pôvodnú hodnotu, ktoré musí z pohľadu mastera trvať opäť aspoň 480 μs. Takto dlhý čas je potrebný pre iButton, aby mohol potvrdiť svoju prítomnosť poslaním presence pulzu. Úlohou reset pulzu je detegovať prítomnosť zariadenia. Ak master pošle iButtonu reset pulz, iButton počká 15 μs až 60 μs (na obrázku 1.5 značený ako t_{PDH}) a pošle presence pulz. Presence pulz musí mať trvanie 60 μs až 240 μs

(na obrázku 1.5 značený ako t_{PDL}).



Obr. 1.5: Reset a Presence pulz

Pri našich meraniach sme namerali trvanie zníženého napätia reset pulzu 1100 μs a 150 μs pri presence pulze. Nameranú komunikáciu môžeme vidieť na obrázku 1.6 s vyznačenými časovými intervalmi podľa štandardu.

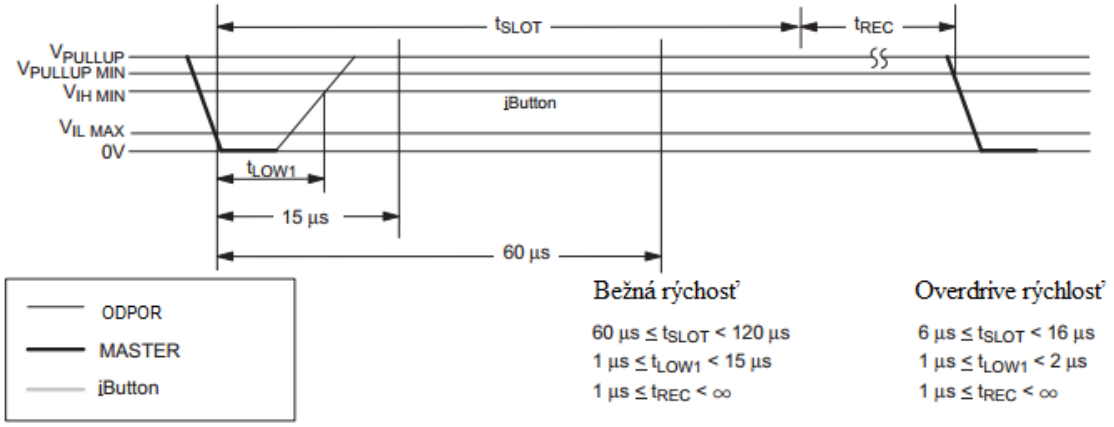


Obr. 1.6: Nameraný Reset a Presence pulz

1.4.2 Zápis do časového úseku zo strany mastera

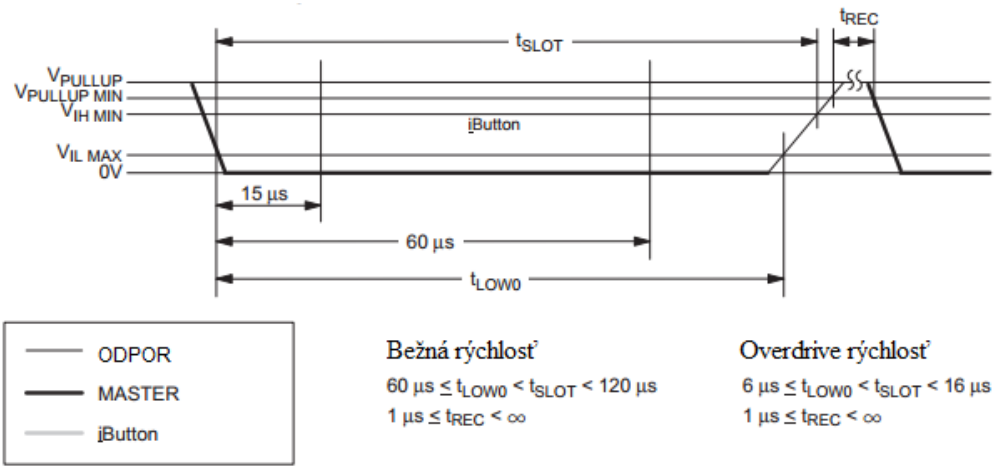
Aktívna časť časového úseku je vo 1-Wire protokole definovaná na 60 μs (na obrázku 1.7 označovaný ako t_{SLOT}). Za bežných okolností začne iButton testovať spojenie 30 μs po spáde v napätí. Povolené tolerančné pásmo je medzi 15 μs až 60 μs. To znamená, že snímanie iButtonu sa môže nachádzať kdekoľvek v tomto intervale. Počas tohto časového úseku musí napätie v spojení zostať buď pod hodnotou $V_{IL\,MAX}$,

alebo nad hodnotou $V_{IH\,MIN}$ pre jednoznačnosť rozoznania jednotky a nuly. Trvanie zníženého napätia musí pri zápise jednotky trvať maximálne 15 μs (na obrázku 1.7 označený ako t_{LOW1}).



Obr. 1.7: Časový úsek popisujúci zápis jednotky

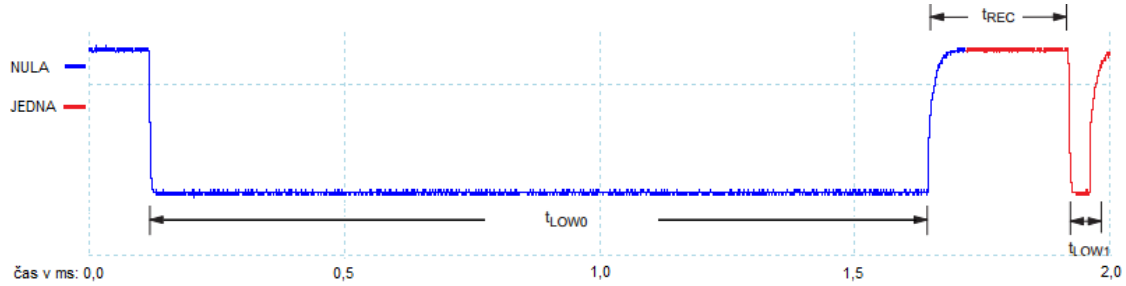
Pri zápise nuly musí zotrvať znížené aspoň 60 μs (na obrázku 1.8 označený ako t_{LOW0}). Trvanie aktívnej časti časového úseku sa môže predĺžiť nad 60 μs. Toto predĺženie je limitované len tým, že doba zníženého napätia, trvajúca aspoň 8 aktívnych časových úsekov (480 μs), je definovaná ako reset pulz. Na konci každej aktívnej časti časového úseku potrebuje iButton čas na zotavenie (na obrázkoch 1.7 a 1.8 označený ako t_{REC}) trvajúci minimálne 1 μs, ktorý slúži ako príprava na nasledujúci bit a dobitie kondenzátora. Čas na zotavenie sa dá považovať za neaktívnu časť časového úseku.



Obr. 1.8: Časový úsek popisujúci zápis nuly

Pri zápise jednotky sme namerali znížené napätie 3-4 μs. Pri zápise nuly 120 μs.

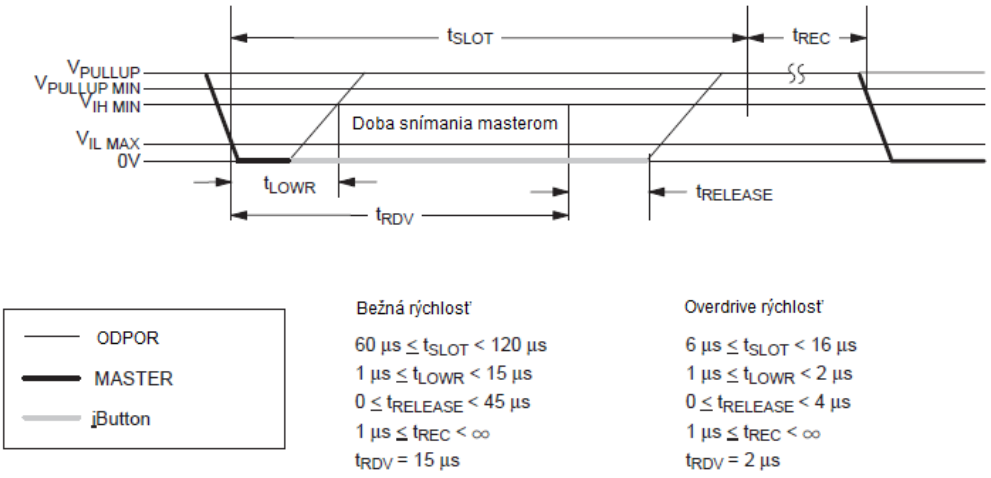
Čas na zotavenie bol zhruba 20 μs. Namerané hodnoty pre zápis jednotky a nuly s časom na zotavenie medzitým môžeme vidieť na obrázku 1.9.



Obr. 1.9: Nameraný zápis nuly a jednotky

1.4.3 Čítanie z časového úseku masterom

Kedže po poslaní príkazu master očakáva dáta, ktoré si od iButtonu vyžiadal, potrebuje ich vedieť čítať. Dáta sú posielané kombinovaním časových úsekov na poslanie jednotky a na poslanie nuly. Pre čítanie dát musí master generovať časový úsek na čítanie s počiatočnou spádovou hranou pre každý posielaný bit. Z pohľadu mastera vyzerá časový úsek na čítanie identicky s príkazom na posielanie jednotky (Obrázok 1.7).



Obr. 1.10: Časový úsek slúžiaci na čítanie dát

Po tom, čo iButton deteguje zníženie napätia, posiela jeden bit svojej adresy. Ak potrebuje poslať bit s hodnotou jedna, iButton nechá napätie nezmenené. V prípade, že iButton potrebuje poslať bit s hodnotou nula, pridrží napätie znížené po dobu označenú na obrázku 1.10 ako t_{RDV}. Úsek t_{LOWR}, ktorý posiela master, by

mal trvať aspoň 1 μs, aby iButton stihol zmenu zaznamenať. Po úseku označenom t_{RDV} nasleduje časť $t_{RELEASE}$, ktorá slúži na zvýšenie napätia na pôvodnú hodnotu. Časť $t_{RELEASE}$ môže trvať v intervale od 0 μs po 45 μs avšak jej číselná hodnota býva 15 μs.

Kapitola 2

Možnosti útoku

V oblasti kryptológie sú za útoky postrannými kanálmi považované tie, ktoré sa nesústredia na hľadanie slabiny samotného algoritmu, ale využívajú informácie získané z fyzickej implementácie systému pri vykonávaní operácií algoritmu. Predstavíme si niekoľko najbežnejších, vrátane nami využívaného útoku postranným časovým kanálom.

2.1 Klasifikácia útokov postrannými kanálmi

V tejto kapitole sa zameriame na základné rozdelenie útokov podľa druhu analyzovaného postranného kanálu. Každý druh útoku je založený na jednej konkrétnej merateľnej veličine. Väčšinou sa jedná o fyzikálne veličiny, ktoré má útočník možnosť namerať. Aby bol útok možný, musia byť získané hodnoty závislé na priebehu výpočtu, ktoré zariadenie vykonáva. Útoky sú pomenované práve po veličinách, ktoré sa pri útoku využívajú. Dôležitou súčasťou útoku je analýza získaných dát. Touto analýzou je možné zistiť niektoré informácie o systéme alebo druh algoritmu, ktorý je využívaný. Analýzou je možné docieliť zjednodušenie útoku hrubou silou.

2.2 Prúdový postranný kanál

Tento druh kanálu sa dá nájsť v každom zariadení obsahujúcom elektronickú časť. Základným princípom tohto útoku je závislosť množstva energie, ktoré sa spotrebuje, od typu prebiehajúcej operácie. Väčšina elektronických zariadení, ktoré nevykoná-

vajú žiadnu zložitú operáciu, má mnohonásobne menší odber ako pri ich vykonávaní. Týmto druhom útoku sú najčastejšie napadnuté čipové karty, pretože nemajú vlastný zdroj. To umožnuje ľahké meranie ich spotreby.

2.3 Elektromagnetický postranný kanál

Útok využívajúci elektromagnetický postranný kanál priamo závisí na skutočnosti, že priebeh niektorých operácií vo fyzickej implementácii algoritmu je sprevádzaný elektromagnetickým žiarením. Všetky zariadenia z elektronických častí využívajú elektrický prúd a preto vyžarujú elektromagnetické vlny do svojho blízkeho okolia. Útočník, ktorý sa vie dostať do dostatočnej blízkosti komponentov a má potrebné vybavenie, vie tieto vlny zachytávať a analýzou dát odchytávať informácie. Niekedy sú útoky využívajúce elektromagnetický postranný kanál zahrnuté pod útoky prúdovým postranným kanálom. Je tomu tak práve pre úzky súvis väčšieho odberu prúdu a generovania elektromagnetických vĺn. Základné elektronické súčiastky nadobúdajú pri činnosti na svojich vstupoch a výstupoch logickú nulu alebo logickú jednotku. Elektromagnetický postranný kanál využíva práve prechody medzi tymito dvoma logickými stavmi, ktoré sa prejavia zmenou intenzity elektromagnetického poľa v okolí zariadenia.

2.4 Chybový postranný kanál

Pri útokoch chybovým postranným kanálom sa útočník snaží zaviesť do priebehu výpočtu chyby tak, aby mu ich výskyt prezradil niečo bližšie o systéme. Chyby sa môže pokúsiť vyvolať napríklad krátkodobým zvýšením (resp. znížením) napätia zariadenia, extrémnymi teplotami alebo ožiarením intenzívnym svetlom.

2.5 Časový postranný kanál

Využitím časového postranného kanálu útočník sleduje a analyzuje čas potrebný na vyhodnotenie vstupu. Je to príklad útoku, ktorý využíva správanie sa implementácie algoritmu a nie matematické vlastnosti algoritmu. Keďže každá logická operácia trvá určitý čas, ktorý sa môže líšiť v závislosti od vstupu, je útočník schopný presným

meraním dĺžky spracovania pripravených vstupov získať potrebné informácie. Preto útok postranným časovým kanálom je použiteľný na zariadenia, ktoré majú priamu súvislosť medzi vstupom a časom potrebným na spracovanie vstupu. Takýmto spôsobom útoku vie byť odhalenie informácie rýchlejšie než hrubou silu.

V našej práci ďalej využívame práve časový postranný kanál z dôvodu, že sme odhalili závislosť medzi vstupom a dĺžkou času stráveného jeho vyhodnotením. Ďalším faktorom je, že čas potrebný na vyhodnotenie získaných dát a generovanie nového potenciálne lepšieho vstupu, ktorý sa blíži k niektorému z akceptovaných vstupov, je výrazne kratší ako čas potrebný na útok hrubou silou.

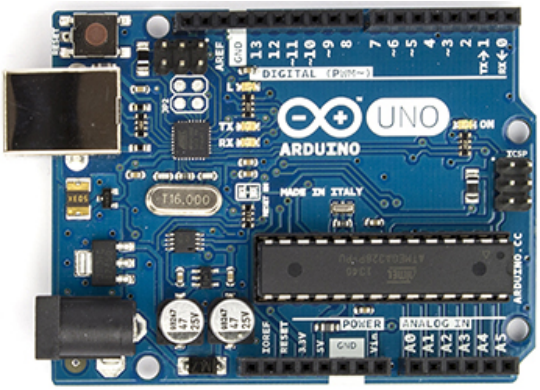
Kapitola 3

Implementácia

Pri našej práci sme využívali zariadenia Arduino UNO, Picoscope 6403D a plošnú dosku RAK DEK, ktoré si bližšie popíšeme.

3.1 Arduino UNO

Arduino UNO je mikrokontrolérová doska s mikrokontrolérom ATmega328P. Obsahuje 14 digitálnych pinov, 6 analógových pinov, 16 MHz keramický rezonátor a USB konektor. Pracuje pod napätím 5 V. Rozmery zariadenia sú 68,6 mm x 53,4 mm a váha 25 g. Zariadenie môžeme vidieť na obrázku 3.1. Arduino UNO stačí pripojiť pomocou USB káblu k počítaču alebo externému zdroju. Schéma zariadenia je dostupná na oficiálnej stránke [2]. ATmega328P má FLASH pamäť veľkosti 32 kB, z čoho je 0,5 kB vyčlenených pre bootloader. Zbytok pamäti je k dispozícii pre program. Arduino obsahuje 2 kB SRAM, ktorá je použiteľná programom pre dáta a 1 kB EEPROM. Každý jeden zo 14 pinov môže byť použitý ako vstupný alebo výstupný. Jednotlivé piny je možné nastavovať s využitím funkcií pinMode(), digitalWrite() a digitalRead(). Arduino UNO vie komunikovať s počítačom, iným Arduinom alebo čiste s mikrokontrolérom. Pri našej práci sme využívali oficiálny softvér [3], ktorý nám umožňuje jednoduché nahrávanie skompilovaného programu cez USB port do Arduina. Taktiež poskytuje jednoduché sledovanie našich textových výpisov prostredníctvom sériového terminálu. V našom riešení Arduino UNO slúži na generovanie kľúčov a komunikáciu s plošnou doskou RAK DEK, v ktorej vystupovalo ako iButton.



Obr. 3.1: Pohľad na Arduino UNO zhora

3.2 Picoscope 6403D

Osciloskop je elektronické meracie zariadenie, ktoré umožňuje pozorovať neustále meniace sa napätie signálu a zobrazovať ho dvojrozmerne ako funkciu v závislosti od času. Umožňuje aj súbežné snímanie viacerých signálov. Na zobrazenej krivke sa dajú sledovať vlastnosti ako je amplitúda, frekvencia a iné. Zariadenie Picoscope 6403D, ktoré sme využívali, vie priamo pomocou USB prepojenia s počítačom zobrazovať potrebné údaje. Dajú sa ním pozorovať 4 rôzne signály, má šírku pásma 500 MHz, USB 2.0 a vlastný softvér pre rôzne operačné systémy [8]. Uvedený softvér sme využívali pri pozorovaní komunikácie plošnej dosky RAK DEK s Arduino UNO alebo iButtonom. Na meranie signálov LED diódy ako aj samotnej komunikácie sme využívali testovacie sondy pripojené k osciloskopu. Picoscope 6403D môžeme vidieť na obrázku 3.2.

Obr. 3.2: Pohľad na Picoscope 6403D

3.3 RAK DEK

Plošná doska RAK DEK je plnohodnotná operačno-pamäťová jednotka, ktorá je voľne predajná [7]. Rozšírené programovanie operačno-pamäťovej jednotky poskytuje softvér RAK, ktorý umožňuje vytvorenie jednoduchej databázy, definovanie vlastností identifikátorov, vlastnosti zariadenia, nastavenie časovej dĺžky zopnutia relé, hľadanie identifikátora a načítavanie identifikátorov z operačno-pamäťovej jednotky. Pre programovanie je nevyhnutné pripojiť operačno-pamäťovú jednotku k PC pomocou kábla USB. RAK DEK poskytuje jednosmernú kontrolu dverí bez ukladania záznamov. Vyžaduje napájanie 8-15 V pri striedavom napätí alebo 9-24 V pri jednosmernom prúde. Podporuje identifikátor iButton DS1990. Programovať sa dá priamo s využitím tlačidiel na doske alebo za pomoci softvéru. Poskytuje konektor na pripojenie dotykovej plochy, relé a USB. Rozmery plošnej dosky sú 85 mm x 85 mm. Zariadenie môžeme vidieť na obrázku 3.3. V našej práci sme k plošnej doske RAK DEK pripojili zariadenie Arduino UNO a dotykovú plochu s LED diódou, na ktorú sme pripevnili fotorezistor. Vďaka tomuto spojeniu sme mohli pomocou naprogramovaného Arduina generovať kľúče a skúmať správanie plošnej dosky RAK DEK. Súčasne sme sledovali dobu svietenia LED diódy pomocou fotorezistoru.

Obr. 3.3: Pohľad na plošnú dosku RAK DEK zhora

3.4 Schéma zapojenia zariadení

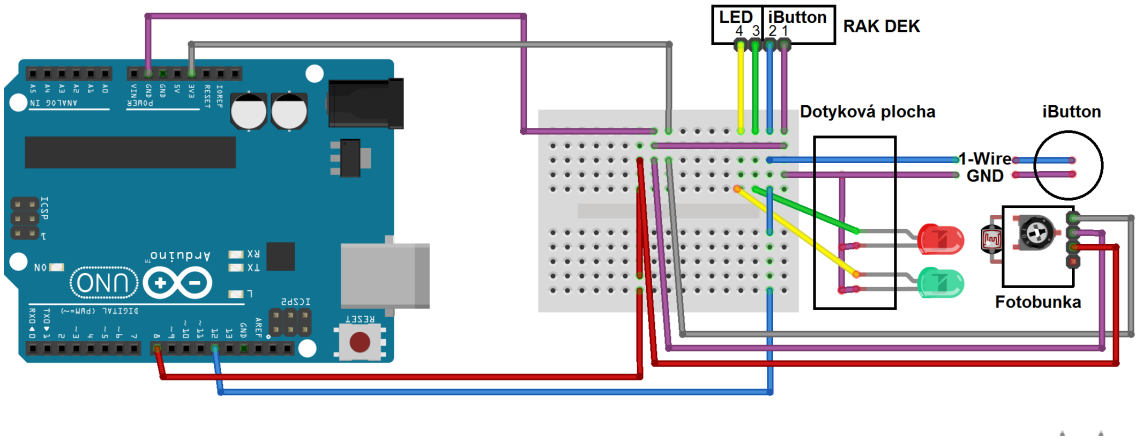
V praktickej časti práce sme potrebovali zariadenia vhodne prepojiť. K Arduinu sme pridali Arduino Proto Shield [1], ktorý obsahuje aj bezspájkové kontaktné pole, čo nám umožnilo prepájať jednotlivé komponenty bez spájkovania. V shielde sú jednotlivé piny prepojené po stĺpcoch (z pohľadu znázorneného na obrázku 3.4). Ak napríklad spojíme jeden pin v stĺpci so vstupným pinom číslo 12 na Arduine, tak celý stĺpec (až po oddeľovač) bude komunikovať s týmto vstupným pinom. Na obrázku 3.4 nám tento shield reprezentuje stredná sivá časť, do ktorej sú zapojené zvyšné zariadenia. Na obrázku sú rovnakou farbou označené prepojenia s jedným pinom na Arduine.

3.4.1 Zapojenie RAK DEK

K zariadeniu RAK DEK sa pri bežnom využití pripája priamo dotyková plocha, magnetický zámok a zdroj. Dotyková plocha so štyrmi vodičmi sa pripája nasledovne: na konektor označený číslom jedna sa pripojí uzemnenie (GND), na konektor číslo dva sa pripája vodič slúžiaci na komunikáciu cez 1-Wire protokol, číslo tri a číslo štyri sú LED diódy. Pri našom riešení a meraniach sme dodržali toto zapojenie s tým, že RAK DEK aj dotykovú plochu sme prepojili cez Proto Shield k Arduino UNO. Na obrázku 3.4 je zariadenie RAK DEK zaznačené len symbolicky štyrmi vychádzajúcimi konektormi. RAK DEK sme s dotykovou plochou a Arduinom prepojili tak, že LED dióda dotykovej plochy je prepojená priamo s RAK DEK. Uzemnenie je spoločné pre dotykovú plochu aj pre RAK DEK cez Arduino UNO. Komunikáciu 1-Wire protokolom máme prepojenú so vstupno-výstupným pinom číslo 12. Cez tento pin čítame príkazy od RAK DEK a posielame naše odpovede simulujúce iButton. Na obrázku je naznačený iButton, ktorý je tam len kvôli znázorneniu komunikácie s dotykovou plochou. V skutočnosti v našom riešení máme dotykovú plochu pripojenú priamo k fotobunke a celú komunikáciu iButtonu nahrazduje Arduino.

3.4.2 Zapojenie fotobunky

Dôležitou súčasťou nášho riešenia je fotobunka, ktorou meriame dobu svietenia LED diódy na dotykovej ploche. Použitá fotobunka je na obrázku 3.4 symbolicky reprezen-



Obr. 3.4: Schéma zapojenia zariadení

tovaná fotoodporom a štyrmi výstupmi. Pri zapojení sme využívali len tri výstupy. Zapojili sme uzemnenie, ktoré spolu s RAK DEK a dotykovou plochou je riešené cez Arduino. Ďalej sme potrebovali zapojiť napájanie a digitálny výstup. Digitálny výstup je v našom riešení spojený s Arduinom cez vstupno-výstupný pin číslo 8. Analógový výstup sme nechali nezapojený.

Kapitola 4

Pozorovania a merania

V tejto kapitole uvádzame zistenia, na ktoré sme prišli počas testov a meraní s využitím osciloskopu. Podrobnejšie sa budeme venovať tým nameraným hodnotám, ktoré poukazujú na slabinu systému.

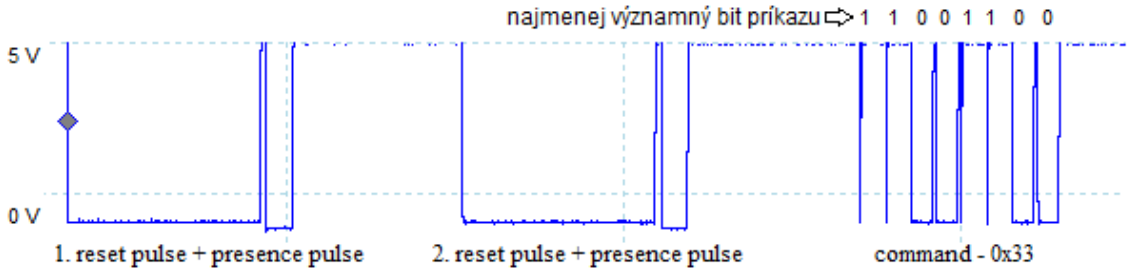
4.1 Pozorovania na RAK DEK

Na začiatku bolo veľmi dôležité zistiť, čo všetko zariadenie RAK DEK overuje. Pripravili sme si program, ktorý simuluje komunikáciu iButtonu na základe 1-Wire protokolu. To nám dalo možnosť testovať aj vstupy, ktoré reálne nenastanú. Pri testovaní týchto vstupov sme zistili, že RAK DEK overuje platnosť CRC každého kľúča a vyžaduje family code 01 (ostatné ignoruje). Z pozorovania identifikátorov reálnych iButtonov sme zistili, že ich počet nie je tak veľký, aby identifikátory využívali všetkých 6 bajtov. V praxi to znamená, že kľúč má dva najľavejšie bajty identifikátora nulové. To však z nášho pohľadu iba urýchľuje čas strávený hľadaním správneho kľúča.

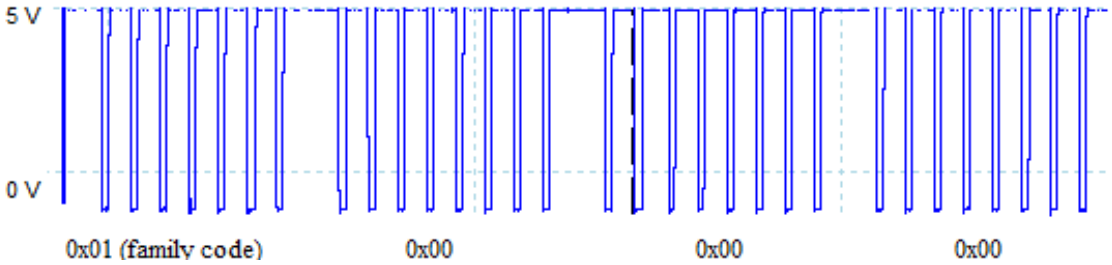
4.2 Merania s osciloskopom

Vďaka meraniam na osciloskope sme mohli pozorovať presný priebeh komunikácie medzi RAK DEK a Arduino UNO simulujúcim iButton. Zistili sme, že RAK DEK posiela iButtonu command 0x33, čo predstavuje príkaz Read ROM. Prvým neočakavaným zistením bolo, že zariadenie RAK DEK posiela reset pulz dvakrát po sebe.

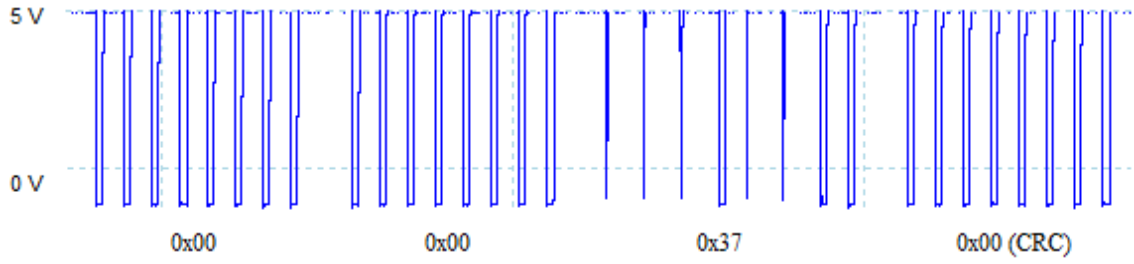
Prvý reset pulz je pravidelný pulz, ktorý slúži na zaregistrovanie prítomnosti zariadenia a druhý pravdepodobne iniciuje ďalšiu komunikáciu a jej synchronizáciu. Priebeh nameranej komunikácie je vidieť na obrázkoch 4.1, 4.2 a 4.3. Na obrázkoch je viditeľne odlíšiteľné posielanie jednotky oproti príkazu na poslanie nuly.



Obr. 4.1: Reset pulz + command



Obr. 4.2: Prvé 4 bajty kľúča 00 37 00 00 00 00 00 01



Obr. 4.3: Druhé 4 bajty kľúča 00 37 00 00 00 00 00 01

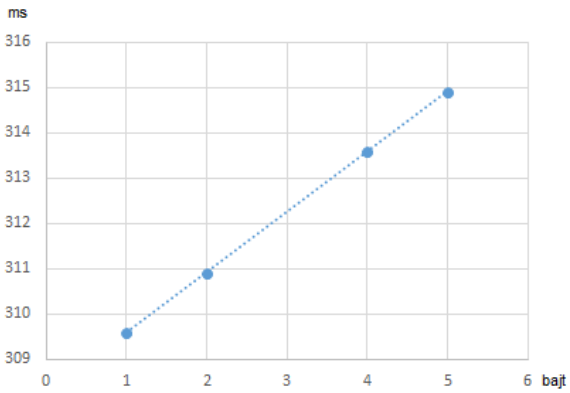
Dôležitým zistením bolo, že zariadenie RAK DEK posiela reset pulz pravidelne. Bez priloženého kľúča je reset pulz posielaný pravidelne každých 100 ms. S priloženým kľúčom je reset pulz posielaný každých 700 ms. Tento fakt neumožňoval využitie reset pulzu ako časového postranného kanálu. Veľmi podstatným bolo zistenie, že LED dióda pripojená k plošnej doske RAK DEK aj LED dióda na pripevnenej dotykovej ploche sa správa rovnako. Dĺžka svietenia je závislá od spracovávaného kľúča. Preto sme sa upriamili na využitie časového postranného kanálu využívajúc svietenie LED diódy.

Pri pozorovaní správania sa LED diódy počas komunikácie sme narazili na ďalší neočakávaný problém, ktorý bolo treba riešiť. Zistili sme, že LED dióda pravidelne svieti 100 ms a následne 1000 ms nesvieti aj počas komunikácie. To sa pri meraniach, kde sa svietenie diódy vyskytlo počas priebehu komunikácie, neprejavilo vôbec pretože LED dióda svieti vždy počas komunikácie s iButtonom. Ak však svietenie vyšlo tesne pred koncom alebo na konci komunikácie, podstatne to predĺžilo čas svietenia LED diódy. Opakovaným testovaním sme zistili, že v priebehu 11 testovaní kľúčov po sebe nastane pravidelné svietenie diódy aj na konci niektorej komunikácie.

Ďalej bolo potrebné zistiť, v akom poradí sú overované jednotlivé bajty. Pre zistenie sme potrebovali vygenerovať kľúče s čo najväčším rozdielom, vzhľadom na počet správnych bitov. Preto sme sa rozhodli do zariadenia RAK DEK vložiť kľúč 0x86, 0x80, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01 a testovať správanie kľúčov s najväčšou odchýlkou voči vloženému. Zachovali sme hodnotu CRC, aby sme neovplyvnili čas overovania bajtov. Preto sme na testovanie vybrali ako prvý 0x86, 0x80, 0x00, 0x00, 0x00, 0x5e, 0x00, 0x01, ktorý sa odlišuje už v prvom (najmenej významnom) bajte identifikátora. Ako druhý 0x86, 0x00, 0x2c, 0x00, 0x00, 0x00, 0x01, 0x01, ktorý sa odlišuje až na predposlednom bajte identifikátora. Doba svietenia LED diódy pri prvom testovanom kľúči bola 309,6 ms, zatiaľ čo pri druhom až 314,9 ms. Z toho je zrejmé, že bajty sa overujú od najmenej významného.

S touto informáciou sme sa rozhodli ísť skúmať rozdiel voči jednotlivým bajtom postupne. Analogicky sme si vygenerovali kľúče s rovnakým CRC, ktoré sa líšili v druhom a štvrtom bajte. Pri porovnávaní času sme našli lineárnu závislosť času stráveného porovnávaním s kľúčmi v pamäti voči poradiu porovnávaného bajtu. V tabuľke 4.4 je uvedené poradové číslo prvého rozdielneho bajtu, testovaný kľúč a nameraný čas. V grafe je vidieť skoro lineárnu závislosť nameraných hodnôt rastúcu po približne 1,3 ms za 1 bajt.

Jedno z posledných zistení bolo poradie porovnávania údajov v pamäti RAK DEK. Zistili sme, že ako prvé sa porovnávajú CRC. Ak sa niektoré v pamäti zhoduje s načítaným, začne sa kontrolou family kódu a následne porovnávaním identifikátoru bajt po bajte od najmenej významných bajtov.

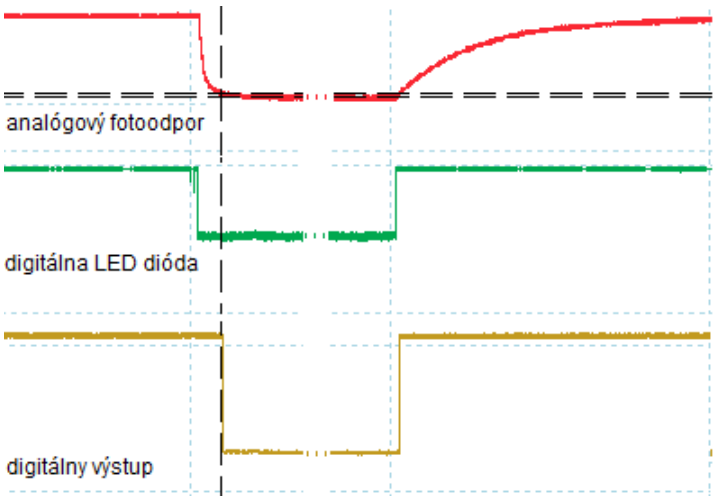


bajt	kľúč	čas v ms
1.	86800000005E0001	309,6
2.	8600000000D50101	310,9
4.	860000A700000101	313,6
5.	86002C0000000101	314,9

Obr. 4.4: Tabuľka a graf znázorňujúce závislosť času voči porovnávanému bajtu

4.2.1 Nastavenie fotorezistoru

Ďalšou dôležitou úlohou bolo nastavenie fotorezistoru, aby posielal digitálne signály o zmene stavu LED diódy čo najpresnejšie. Na to sme opäť využili osciloskop a nakalibrovali hraničné pásmo, od ktorého zaznamenáva zmenu.



Obr. 4.5: Zmena stavu fotorezistoru v závislosti od LED diódy

Na obrázku 4.5 je možné pozorovať červenou analógový signál, ktorý klesá aj stúpa postupne pri zmene stavu diódy. Zelenou digitálny výstup LED diódy fotorezistora, ktorý signalizuje zmenu okamžite po zmene stavu analógového signálu. A žltou nastavený digitálny výstup, ktorý zareaguje až v okamihu, keď analógový signál klesne (resp. stúpne) pod (nad) hranicu 620 mV. Na obrázkoch je hranica zmeny znázornená čiernou vertikálnou prerušovanou čiarou.

Kapitola 5

Spôsob generovania kľúčov

V tejto časti práce popíšeme spôsob, akým realizujeme útok s využitím časového postranného kanálu LED diódy. Generovanie vstupných kľúčov riešime pomocou Arduino UNO a pripraveného programu, ktorý je kombináciou assembleru a jazyka C. V kapitole si pre jednoduchosť uvedieme pseudokód s popisom funkčnosti. Samotný zdrojový kód je dostupný v prílohe práce.

5.1 Implementácia 1-Wire protokolu

Aby náš program bol schopný komunikovať so zariadením RAK DEK a vystupovať ako iButton, bolo nutné implementovať základné funkcie 1-Wire protokolu. Ako sme si spomenuli v prvej kapitole, kľúčovým je práve načasovanie a meranie trvania poklesov v napätí. Základom načasovania bol prepočet z operácií vykonaných procesorom na mikrosekundy. Tento prepočet sme realizovali jednoduchou funkciou, ktorá dostala na vstupe počet cyklov operácii a na základe frekvencie procesora vrátila dobu trvania v mikrosekundách. Rovnako sme implementovali inverznú funkciu, ktorá dostala na vstupe čas v mikrosekundách a vrátila počet cyklov. Na využívanie týchto prepočtov bolo lepšie mať základné funkcie priamo v assembleri. Vďaka tomu sme boli schopní presne určiť počet cyklov procesora vykonávaných operácii. Funkcie, ktoré sme takto implementovali, slúžia na počítanie uplynutého času do zmeny v stave na sledovanom pine. To využívame napríklad pri čakaní na reset pulz a posielaní presence pulzu na začiatku komunikácie. Pri týchto funkciách využívame aj pomocné funkcie slúžiace na sledovanie zmeny stavu pinu. Pomocné funkcie sledujú,

kedy sa bit na vstupno-výstupnom pine zmení na 0 alebo 1. Potrebujeme ich napríklad na to, aby sme počkali, kým napätie stúpne, než opätovne začneme kontrolovať jeho pokles. Funkcie môžeme vidieť naznačené nižšie na výpise 5.1 a 5.2.

```
void cakaj_na_reset_pulz()
{
  uint16_t count;
  do {
    cakaj_pokial_bit_nieje_nula(Pin, Bit);
    pocitaj_pokial_bit_nieje_jedna(Pin, Bit, &count);
  }
  while ( count < mikrosec_do_cyklov( 480 ) );
}
```

Popis funkcie v pseudokóde 5.1: Reset Pulz

Funkcia čaká spádovú hranu na pripojenom vstupnom pine a následne začne počítať uplynutý čas do opätovného zvýšenia napätia. Výsledok uloží do premennej count. Toto opakuje v cykle, pokiaľ trvanie poklesu napätia nie je aspoň 480 μs. Ak je trvanie dostatočne dlhé, jedná sa podľa špecifikácie o reset pulz a program zareaguje poslaním presence pulzu.

```
void posli_presence_pulse()
{
  pinModeOutput();
  cakajMicrosekund(60 az 240 mikrosec);
  pinModeInput();
  cakaj_pokial_bit_je_jedna(Pin, Bit);
}
```

Popis funkcie v pseudokóde 5.2: Presence Pulse

Program využitím funkcie pinModeOutput() zmení nastavenie pinu na výstupný. To spojí dátový vodič so zemou, čím vytvorí pokles v napätí. Takto znížené napätie pridrží aspoň 60 μs podľa štandardu a následne nechá zvýšiť napätie späť s využitím funkcie pinModeInput(). I/O pin sa v našom nastavení pri prepnutí na vstup stane vstupom s vysokou impendaciou, čím sa efektívne odpojí od zbernice. Pre istotu ešte počkáme, pokiaľ sa napätie zvýši na pôvodnú hodnotu, aby sme inou funkciou

nezaznamenali pokles v napätí skôr, než reálne nastal. Po poslaní presence pulzu sme podľa špecifikácie mali očakávať obdržanie príkazu (commandu). Takže sme implementovali funkciu, ktorá na základe doby zníženého napätia určila či nám RAK DEK poslal bit s hodnotou jedna alebo nula. Z ôsmych takto získaných bitov sme vyskladali výsledný bajt príkazu. Pseudokód funkcie môžeme vidieť na 5.3.

```
uint8_t obdrz_command()
{
  cmd = 0;
  zostavajuce_bity = 8;
  do {
       cmd >>= 1;
       val = zisti_bit();
       if ( val == 1 )
         cmd |= 1 << 7;
       zostavajuce_bity--;
  } while ( zostavajuce_bity != 0 );
  return cmd;
}
```

Popis funkcie v pseudokóde 5.3: Obdrž Command

Získaný bajt sme vyhodnotili a ak sa jednalo o príkaz 0x33 Read ROM tak sme príkaz akceptovali. V opačnom prípade musela nastať niekde chyba a tým pádom je nutné iniciovať celú komunikáciu od začiatku. Priebeh úvodu komunikácie sa dá popísať ako je naznačené na výpise 5.4, kde 0xFF je náš pomocný príkaz pre druhý reset pulz. Oneskorenie vychádza zo špecifikácie.

```
do {
    cakaj_na_reset_pulse();
    do {
      cakajMicrosekund( 15 );
      posli_presence_pulse();
      buf = obdrz_command();
    } while (buf == 0xFF);
  } while ( buf != 0x33 );
```

Popis funkcie v pseudokóde 5.4: Úvod komunikácie

Po obdržaní platného príkazu je rad na nás, aby sme postupne poslali pripravený identifikátor. Ten posielame kombinovaním príkazov na poslanie jednotky a poslanie nuly. Počas týchto funkcií zabránime prerušeniam, aby nedošlo k posunu alebo chybe v komunikácii. Na zamedzenie prerušení nám slúži funkcia cli() a na ich opätovné povolenie funkcia sei(). Spôsob posielania jednotky a nuly môžeme vidieť na výpise 5.5 a 5.6.

```
void posli_jednotku()
{
  cli();
  cakaj_pokial_bit_nieje_nula(Pin, Bit);
  sei();
  cli();
  cakaj_pokial_bit_nieje_jedna(Pin, Bit);
  sei();
}
```

Popis funkcie v pseudokóde 5.5: Posielanie jednotky

```
void posli_nulu()
{
  cli();
  cakaj_pokial_bit_nieje_nula(Pin, Bit);
  sei();
  pinModeOutput();
  cakajMicrosekund( 60 );
  pinModeInput();
  cli();
  cakaj_pokial_bit_nieje_jedna(Pin, Bit);
  sei();
}
```

Popis funkcie v pseudokóde 5.6: Posielanie nuly

Ako sme si uviedli v prvej kapitole, príkaz na posielanie jednotky obsahuje počkanie na spádovú hranu v napätí a následne počká, kým sa napätie nezvýši na pôvodnú hodnotu. Príkaz na poslanie nuly musí podržať napätie znížené dosť dlho na to, aby zariadenie RAK DEK zaznamenalo tento bit ako nulu.

5.2 Generovanie kľúčov

Na začiatku začíname od kľúča 0x3D, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, ktorý má na všetkých miestach identifikátora 0. Generovanie nového kľúča začína v našom programe kalibráciou. Kalibrácia spraví 11 testov na aktuálne najlepšom kľúči a zaznemaná si namerané časy svietenia LED diódy. Najdlhší čas pokladáme za predĺžený o svietenie LED diódy RAK DEK. Zo zvyšných hodnôt si spravíme priemer a ten pokladáme za čas potrebný na spracovanie daného kľúča. Na začiatku nemáme informáciu, či je kľúč správny alebo nie. Skontrolujeme, či žiadna ďalšia z nameraných hodnôt nemá podobne dlhý čas ako tá najväčšia. To by predstavovalo chybu kalibrácie. Rovnako overíme, či počas komunikácie nenastala nejaká chyba a teda nie je žiadny z časov nulový. V prípade, že hodnota, ktorú pokladáme za predĺženú o svietenie LED diódy a priemerná hodnota boli príliš blízko seba, postupujeme ďalej v pomalšom režime. Ten predstavuje overovanie každého kľúča dvakrát, aby sme určite odlíšili svietenie LED diódy od bežného času pre daný kľúč a vybrali minimum. Tento prípad však nie je bežný. Ak by bol kľúč použitý pri kalibrácii správny, pri ďalšej testovanej hodnote bajtu by sme zistili, že sa nám čas overovania skrátil. Z toho vieme uznať predchádzajúcu hodnotu za správnu pre daný bajt a pokračovať testovanie na ďalšom bajte. Takáto situácia nastane vždy, ak je kľúčov menej ako 256. V prípade, že sú na testovaný bajt všetky hodnoty rovnako dobré, pokladáme za správnu ľubovoľnú a pokračujeme hľadaním ďalšieho bajtu. Pri hľadaní správnych bajtov postupujem podľa zistení z merania osciloskopom. Po kalibrácii sa prvý hľadá CRC kód v pamäti RAK DEK. Po jeho nájdení (zistení, že ich je tak veľa, že existuje ľubovoľný) sa presúvame cez prvý (najmenej významný) bajt identifikátora (keďže family code musí byť 01) až kým neprídeme po posledný. Po prechode na nový bajt vždy necháme program prejsť kalibráciou s momentálne najlepším kľúčom. Na tomto mieste postupne testujeme všetkých 256 možných hodnôt. Pri testovaní každej hodnoty potrebujeme, aby nám CRC zostalo zachované. Preto si posledný (najvýznamnejší) bajt identifikátora modifikujeme tak, aby sa CRC zachovalo a zároveň mohol nami hľadaný bajt obsahovať testovanú hodnotu. Ak čas nameraný s aktuálne testovanou hodnotou bajtu je dlhší o viac ako 1350 μs, pokladáme túto hodnotu za správnu a postupujeme na ďalší bajt. Prípad, že nevyhovuje ani jedna z 256 možností by nemal v praxi nastať. Po každom nájde-

nom bajte testujeme, či je to čiastočné zlepšenie alebo finálny kľúč. Pri zopnutí relé (nájdení správneho kľúča) zariadenie RAK DEK necháva diódu svietiť podstatne dlhšie. Tento čas je dlhší ako čas zvýšený o svietenie LED diódy. Preto, ak na tom istom vstupe zariadenie namerá dvakrát po sebe čas dlhší ako je hodnota predĺžená o svietenie LED diódy, považujeme tento kľúč za správny a program ukončíme.

Ako sme z pozorovaní zistili, reálne iButtony majú na prvých dvoch bajtoch identifikátora nuly. Preto aj v našom programe zohľadňujeme túto možnosť a pri hľadaní v poradí štvrtého bajtu skúsime priamo vložiť hodnotu, ktorá by tam bola, ak by posledné dva (najvýznamnejšie) bajty boli nulové. V prípade, že tento kľúč nevyhovuje (nenastalo predĺženie času), pokračujeme v klasickom prehľadávaní zvyšných možností.

Pre prehľadnosť a lepšie testovanie programu sme do priebehu zakomponovali výpisy, ktoré nám oznamujú namerané hodnoty pri kalibrácii, ukazujú momentálne testovaný bajt, momentálne testovanú hodnotu bajtu a pri nájdení správnej hodnoty ju vyznačia. Výpisy nášho programu môžeme vidieť na obrázku 5.1 a 5.2.

```
Calibrating .......... done. Base: 314548, blink: 349929, blink - base: 35380, max: 383959 us.
  0 1 2 3 4 5 6 7 8 9 A B C D E F
0 * * * * * * * * * * * * * * * *
1 * * * * * * * * * * * * * * * *
2 * * * * * * * * * * * * * * * *
3 * * * * * * * * * * * * * * * *
4 * * * * * * * * * * * * * * * *
5 * * * * * * * * * * * * * * * *
6 * * * * * * * * * * * * * * * *
7 * * * * * * * * * * * * * * * *
8 * * * * * * * * * * * #

25 ?? ?? ?? 8C AF E9 01
```

Obr. 5.1: Výpis hľadania hodnoty bajtu v sériovom terminále

```
Calibrating .......... done. Base: 315849, blink: 350538, blink - base: 34690, max: 383879 us.
  0 1 2 3 4 5 6 7 8 9 A B C D E F
0 * - - - - - - - - - - - - - - -
1 - - - - - - - @

25 00 00 17 8C AF E9 01
```

Obr. 5.2: Výpis nájdenia správneho kľuča v sériovom terminále

Na začiatku je vidieť kalibráciu s výpisom priemerného nameraného času pre testovaný kľúč. Za tým nasleduje čas, ktorý sa už považuje za predĺžený o svietenie LED diódy, rozdiel týchto dvoch hodnôt a maximálny nameraný čas. Všetky

hodnoty sú vypísané v mikrosekundách. Po kalibrácii postupne vypisujeme, akú hodnotu práve testujeme v hľadanom bajte. Na obrázku 5.1 je vidieť vyznačenie správnej hodnoty testovaného bajtu znakom #. Po jeho nájdení vypíšeme kľúč, kde už nájdené bajty majú vypísanú svoju hodnotu a zatiaľ neznáme sú označené znakom ??. Na obrázku 5.2 je vidieť, že ak nájdeme finálny kľúč, vo výpise sa posledná hľadaná hodnota bajtu označí znakom @ a vypíšeme nájdený kľúč.

Záver

Jedným z cieľov práce bolo preverenie bezpečnosti prístupových systémov, ktoré využívajú elektronické kľúče iButton. V obsahu práce sme uviedli slabinu systému, ktorú sme pri meraniach objavili. Uskutočnili sme úspešný útok na zariadenie RAK DEK s využitím časového postranného kanálu svietenia LED diódy. Výsledkom práce je postup, ktorým sme slabinu systému odhalili a program, ktorým útok realizujeme. V práci sú popísané zariadenia, ktoré sme k útoku potrebovali aj s ich konkrétnym zapojením. Výsledný program dokázal nájsť platný kľúč uložený v pamäti prístupového systému RAK DEK v čase 10-15 min. Tento výsledok je veľmi uspokojivý vzhľadom na odhadovaný čas útoku hrubou silou. Ten sa podľa článku [6] odhaduje zhruba na 35,5 týždňa v najhoršom prípade.

Kedže sme nemali čas ani možnosť zakúpiť a testovať viacero druhov prístupových zariadení, práca poskytuje možnosť ďalšieho výskumu. Je možné overiť správanie LED diódy pri iných modeloch. Dá sa očakávať, že iný model zariadenia by mohol dosahovať iné hodnoty pri komunikácii s iButtonom. Časová závislosť overovania jednotlivých bajtov sa môže tiež líšiť. V tom prípade by bolo potrebné zistiť rozdiely a nakalibrovať program na upravené hodnoty.

Literatúra

[1] Arduino. Arduino proto shield, 2014. [Citované 2015-18-05] Dostupné z `http://www.arduino.cc/en/Main/ArduinoProtoShield`.

[2] Arduino. Arduino scheme, 2014. [Citované 2015-18-05] Dostupné z `http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf`.

[3] Arduino. Arduino software, 2014. [Citované 2015-18-05] Dostupné z `http://arduino.cc/en/main/software`.

[4] Fritzing. Fritzing software, 2014. [Citované 2015-18-05] Dostupné z `http://fritzing.org/download/`.

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

[6] Richard Ostertág. The ibutton emulator source code., 2014. [Citované 2015-18-05] Dostupné z `https://micro.dcs.fmph.uniba.sk/iButton.zip`.

[7] RYS. Access control and door entry systems, 2014. [Citované 2015-18-05] Dostupné z `http://rys.sk/html_sk/0032010002.php`.

[8] Pico Technology. Pc oscilloscope and data acquisition software, 2014. [Citované 2015-18-05] Dostupné z `http://www.picotech.com/software.html`.