**MATURITNÍ PRÁCE**

Obohacené zapínací tlačítko se zabezpečovací schopností

Studijní obor: IT

Třída: C4

Školní rok: 2022/2023

Jméno a příjmení: Vít Bezouška

Vedoucí práce: Radek Lampíř

**Abstrakt**

**Abstract**

**Klíčová slova**

Prohlašuji, že jsem tuto práci vypracoval(a) samostatně a použil(a) jsem literární prameny a informace, které cituji a uvádím v seznamu použité literatury a zdrojů informací.

V Praze dne …………. ……..……………………

jméno a příjmení

Obsah

[Úvod 6](#_Toc118229301)

[Nápad a první krůčky 7](#_Toc118229302)

[První použití Arduina 7](#_Toc118229303)

[Nedostatek RAM 8](#_Toc118229304)

[Komunikace mezi kalkulačkami Texas Instruments 9](#_Toc118229305)

[Komunikace mezi Arduiny 10](#_Toc118229306)

[Shrnutí 10](#_Toc118229307)

[Cíle práce 12](#_Toc118229308)

[Zabezpečovací zařízení 12](#_Toc118229309)

[Zamezit zapnutí počítače 12](#_Toc118229310)

[Zamezit přístupu k datům anebo k uživatelskému účtu 12](#_Toc118229311)

[Ukazatel času nebo dnešního data 13](#_Toc118229312)

[Prezentovatelné šasí 14](#_Toc118229313)

[Struktura práce 21](#_Toc118229314)

[Titulní strana 21](#_Toc118229315)

[Abstrakt, Abstract, Klíčová slova 21](#_Toc118229316)

[Prohlášení 21](#_Toc118229317)

[Obsah 22](#_Toc118229318)

[Úvod 22](#_Toc118229319)

[Vlastní text práce 22](#_Toc118229320)

[Závěr 23](#_Toc118229321)

[Seznam použitých zdrojů 23](#_Toc118229322)

[Seznam příloh a přílohy 24](#_Toc118229323)

[Základní formátování 24](#_Toc118229324)

[PowerPointová prezentace (či google prezentace) 25](#_Toc118229325)

[Obhajoba maturitní práce a názorné ukázky praktické části maturitní práce 25](#_Toc118229326)

[Způsob odevzdání maturitní práce 26](#_Toc118229327)

# Úvod

Tato práce obsahuje…

# Nápad a první krůčky

Na tomto projektu jsem pracoval už na začátku roku 2021. Jednalo se sice tou dobou jen o jednoduché spojování drátků, ale už to by se dalo považovat za začátek. Bylo to červené tlačítko s krytkou a spínač na klíček, uchycený ve zmenšené kartonové krabičce od tužek. Dvěma kabely to bylo připojené na PW+ a PW-, které se sepnuly pouze když bylo otočeno klíčkem a zároveň bylo stlačené tlačítko.

1 První prototyp zapínacího tlačítka, únor 2021

2 Zapínání počítače pomocí spojení dvou drátků, srpen 2021 (snímek z videa)

Tenhle systém byl velmi neefektivní a ani jsem se nesnažil o žádný cable management. Navíc jsem tou dobou ještě neměl páječku a všechny spoje byly jen velmi pofidérně přivázané takže se často stalo, že jsem do kabelů kopl a něco vypojil, většinou přímo z modulu. Protože jsem líný a nechtělo se mi kabely přilepovat zespoda ke stolu, někdy začátkem srpna téhož roku jsem tlačítko i zámek odpojil a počítač zapínal manuálním spojením těch dvou drátků. Přestože mi první Arduino dorazilo z Číny jen o několik dní později, nespojil jsem si dvě a dvě dohromady, a ještě začátkem října jsem spojoval drátky.

## První použití Arduina

Nejpozději 15.10.2021 byla v provozu první verze tlačítka s Arduinem, konkrétně nějakou neoficiální verzí varianty Nano z Aliexpressu za 3.35$. Relativně velké červené tlačítko jsem vyměnil za generický push switch, protože jeho piny se akorát vešly do prototypovací destičky, pomocí které jsem všechno propojoval. Tato verze už využívala i malý jednobarevný displej s rozlišením 128x32 pixelů a úhlopříčkou 0.91 palce, připojený pomocí protokolu I2C. Na něm se po stisknutí tlačítka zobrazila uvítací hláška, a po zhruba 30 sekundách nečinnosti jednoduchý spořič obrazovky.

3 Druhý prototyp zapínacího tlačítka, říjen 2021

4 První pokusy s displejem, srpen 2021 (snímek z videa)

Tato verze ale také měla pár vad. Hlavní z nich bylo časté nezapnutí počítače po stisknutí tlačítka. Důvodem bylo moje nepochopení principu fungování PNP tranzistorů. Byl jsem v přesvědčení, že aby v PNP tranzistoru proudila elektřina z emitoru do kolektoru, musí být do báze přivedeno podobné napětí jako do emitoru a do kolektoru pak bude proudit elektřina z emitoru i báze. To ale není pravda. Aby to fungovalo, musíme do báze přivést *nižší* napětí tak, aby část proudu z emitoru šla do báze, čímž se umožní proudu projít i do kolektoru[[1]](#footnote-1). Správně jsem měl mezi Arduino a bázi sériově zapojit ještě odpor, ale to jsem zjistil až později.

## Nedostatek RAM

Tou dobou jsem už začínal rozumět potenciálu, který ta malá věcička měla.   
Objednal jsem si tedy kompatibilní čtečku micro SD karet a pár malých repráčků, abych mohl naimplementovat i přehrávání nějaké znělky při zapnutí počítače. Zde jsem ale narazil na další problém: grafická knihovna, kterou jsem používal, Adafruit GFX se totiž nevešla do operační paměti mého Arduina zároveň s knihovnou na čtení SD karet, a knihovnou na přehrávání audio souborů.

Nějakou dobu jsem si nevěděl rady, ale nakonec jsem se rozhodl si pořídit ještě jedno Arduino s větší RAM. Našel jsem si na Aliexpressu variantu Uno, která je o něco větší než Nano. Bohužel jsem si ale nepřečetl popis výrobku dostatečně pozorně a spletl jsem si velikost programové paměti s operační. Arduino Uno je totiž sice větší než Nano a specifikacemi se opravdu liší. Bohužel ale ne ve velikosti RAM, ta je u obou 2KB.

To ale nevadí, protože se mi podařilo propojit ty dvě Arduina tak, aby Uno ovládalo displej a při stisku tlačítka spustilo počítač, a Nano aby jen četlo audiosoubory z SD karty a pak je přehrávalo. Dokonce to ani nebylo tak těžké, jak by se mohlo zdát. Inspiroval jsem se komunikačním protokolem kalkulaček firmy Texas Instruments.

### Komunikace mezi kalkulačkami Texas Instruments

Společnost Texas Instruments nabízí mnoho různých kalkulaček s různými funkcemi a schopnostmi. Rozhodli se ale nechat tyto kalkulačky komunikovat jak mezi sebou, tak i s počítačem pomocí jen tří drátků. Aby spolu mohly komunikovat všechny jejich kalkulačky, bylo potřeba vyřešit problém synchronizace komunikace, aby obě kalkulačky stíhaly přijímat poslaná data a zároveň se zbytečně neomezovaly v přenosové rychlosti, i když jedna z nich může pracovat na nižší frekvenci. Řešením bylo nedovolit odesílateli poslat další bit, dokud příjemce nepotvrdí příjem.

V praxi to funguje tak, že jeden drátek je společná zem, druhý je signál 1 a třetí je signál 0. Zezačátku obě kalkulačky posílají do obou signálových pinů elektřinu, a jakmile se jedna z nich rozhodne posílat data, přestane posílat elektřinu do jednoho ze spojů podle toho, kterou hodnotu chce odeslat. Jakmile druhá kalkulačka zaregistruje tuto změnu, odpojí druhý signálový pin, čímž dává první kalkulačce signál, že zprávu přečetla. Obě potom opět připojí oba piny a odesílající kalkulačka může poslat další signál. Když jsou obě kalkulačky rychlé, komunikace probíhá rychle, pokud ne, rychlost komunikace se přizpůsobí té pomalejší[[2]](#footnote-2).

5 Příklad komunikace mezi dvěma kalkulačkami TI

### Komunikace mezi Arduiny

6 Demonstrace mého komunikačního protokolu

Přestože jsem se u protokolu TI inspiroval, svůj „protokol“ jsem naimplementoval po svém. Protože mi stačila opravdu velmi jednoduchá komunikace, vystačil jsem si se dvěma spoji. Podařilo se mi také zajistit funkčnost hlavního Arduina, kterým se ovládá zapínání počítače, i v případě, že to druhé, které jen spouští zvuk, neodpovídá, je rozbité, nebo třeba jen odpojené.

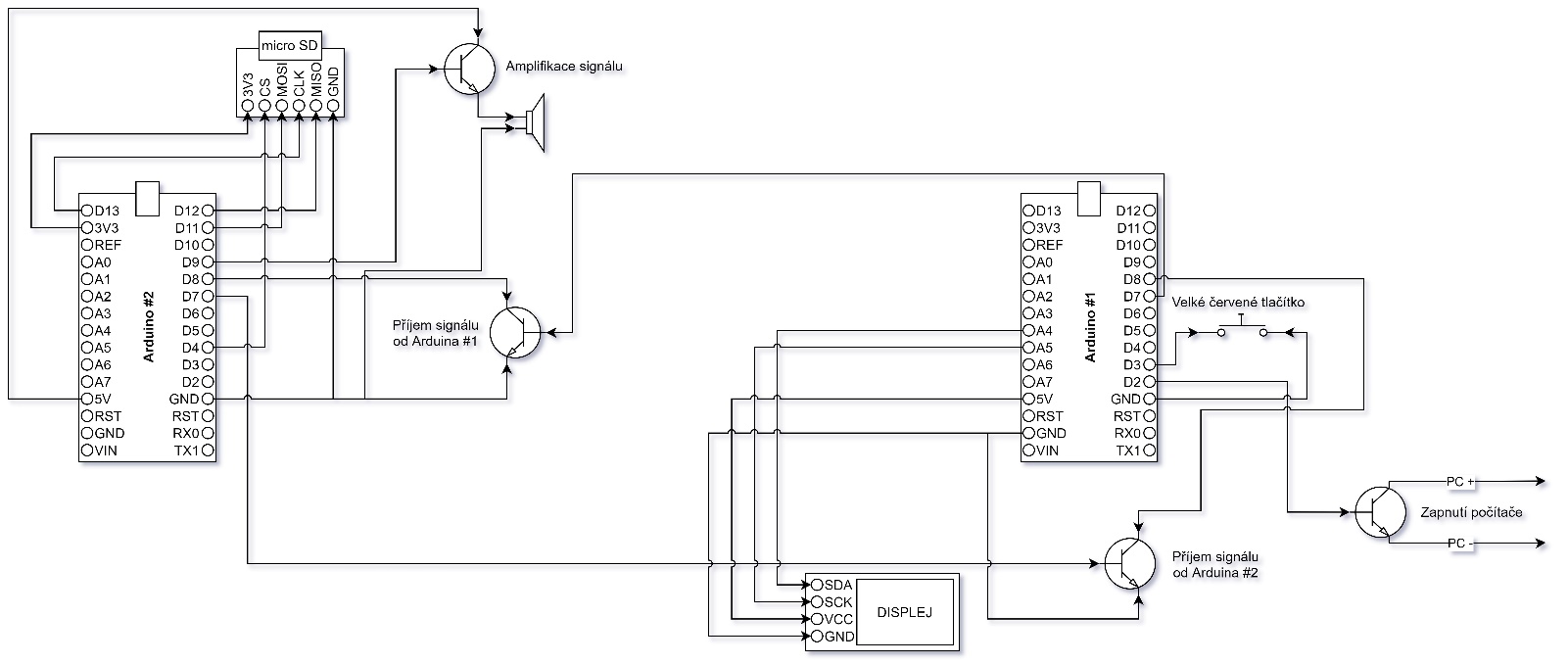
Zvukové Arduino se okamžitě po zapnutí totiž ohlásí posláním stálého signálu do hlavního Arduina, čímž dává najevo, že je připraveno. Jakmile dostane signál zpátky, znamená to, že má přehrát náhodný soubor z SD karty. Jakmile požadavek zpracuje a začne hrát, informuje o tom to hlavní Arduino zastavením odchozího signálu, načež hlavní Arduino také přestane posílat signál a de facto tím komunikaci restartuje, protože když zvukové Ino zpracuje tento signál, započne komunikaci znovu.

7 První podoba propojených Arduin, leden 2022

Protože hlavní Ino ví, kdy to zvukové je a není připraveno přehrát zvuk, může o tom informovat uživatele pomocí svého displeje. Už při první implementaci tohoto protokolu jsem připravil i chybovou hlášku, informující uživatele o nepřipravenosti druhého Ina, a i načítací obrazovku pro zobrazení v době mezi odesláním signálu *hraj* a přijetím signálu *hraju.*

8 Chybová hláška Arduina

## Shrnutí

V téhle konfiguraci Arduina zůstala zhruba tři čtvrtě roku až do doby, kdy jsem z projektu udělal maturitní práci. V mezičase jsem sice měnil menší grafické aspekty a přidával jsem soubory znělek, ze kterých mohl audio modul vybírat, ale funkční aspekty zůstaly stejné.

9 Schéma projektu ve fázi obhájení návrhu na maturitní práci, září 2022

# Cíle práce

V rámci tohoto projektu jsem si vytyčil několik cílů. Tyto cíle se můžou lišit od těch, které jsou uvedeny v návrhu na maturitní práci, protože se mi zdálo, že by mohly být až moc ambiciózní a nemusel bych je stihnout všechny. Nechtěl jsem ale riskovat neúspěch u maturitní komise nesplněním cílů a tak jsem si oficiálně vytyčil trochu vágnější cíle, abych mohl v průběhu práce svůj projekt pozměnit.

## Zabezpečovací zařízení

První a hlavní cíl byl z projektu udělat zabezpečovací zařízení, bez kterého by případný zločinec nemohl proniknout do softwarového nitra počítače. Jsem si jist, že existují i jiné způsoby, než ty dva, které napadly zrovna mě, ale tyto mi přišly nejreálnější.

### Zamezit zapnutí počítače

Nejjednodušší způsob zabezpečení by bylo zapínání počítače na klíč. Když zločinec nemá klíč, teoreticky nemůže počítač zapnout. Prakticky by ale mohl buď otevřít PC skříň, vyndat systémový disk a nabootovat na jiném systému, anebo prostě zkratovat PW+ a PW-, buď naříznutím kabelů, nebo šroubovákem přímo na základní desce.

Aby zařízení odolalo těmto útokům, musely by počítačová bedna a zařízení být zabezpečeny proti vniknutí, a drátky přenášející PW+ a PW- nepřestřihnutelné. To mi přišlo jako nedosažitelné s mými schopnostmi a zdroji.

Nevýhodou by také bylo, že by nebylo jak jinak počítač spustit. Modul by tím byl jediným bodem selhání.

### Zamezit přístupu k datům anebo k uživatelskému účtu



Fikanější způsob je, nějakým způsobem uložit do Arduina heslo a po úspěšném ověření uživatele předat heslo počítači jako keyboard input. Arduino by se tím dalo použít třeba pro dešifrování disků pomocí BitLockeru nebo pro přihlášení do uživatelského účtu v OS.

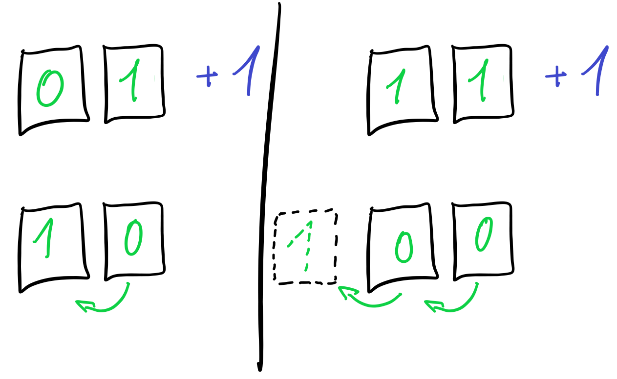
Přestože tohle řešení neochrání přihlašovací údaje proti keyloggerům, čili programům, které monitorují veškeré uživatelské vstupy a zpřístupňují je nepovolaným osobám, přišlo mi tohle řešení jako jedno z nejlepších. Proti keyloggerům se, stejně jako proti jakémukoliv jinému malwaru, dá chránit jednoduše použitím antivirového programu. Tohle je tedy přístup, který jsem si vybral.

## Ukazatel času nebo dnešního data

Vycházím z vlastní zkušenosti, když tvrdím, že toto zařízení zůstane trvale zapojené do počítače. Pokud je ten počítač dostatečně moderní a správně nastaven, dodává přes své USB porty elektřinu i když je vypnutý. Toho se dá využít a zobrazovat na displeji Arduina užitečné informace, třeba čas nebo datum. Arduino je bohužel jednoduché zařízení a po ztrátě napájení, nebo restartu, všechno zapomene a nemá způsob jak bez externích vychytávek uchovávat čas. Naštěstí umí komunikovat s počítačem pomocí sériové komunikace přes simulovaný COM port.

Chtěl jsem tedy vytvořit i jednoduchý pythonový script, který by po úspěšném nabootování do OS dal Arduinu s obrazovkou vědět kolik je hodin a kolikátého je. Tyto informace si Arduino ponechá a bude k nim přidávat čas od svého spuštění i po vypnutí počítače, a může tak nepřetržitě zobrazovat čas.

Tento přístup má dva problémy. Prvním je, že krystalový oscilátor, pomocí kterého Arduino ví, jak dlouho už je zapnuté, se může odchylovat od reality o 70s[[3]](#footnote-3) nebo v některých případech i 300s za den.[[4]](#footnote-4)

Druhý problém je, že Arduino má velmi jednoduchý procesor a nedokáže správně pracovat s velkými čísly. Po zhruba 50 dnech mu dojde místo na počítání doby od zapnutí a jemu nezbyde nic jiného, než začít počítat zase od nuly. Tomuhle se říká integer overflow. Naštěstí to není zas tak velký problém, protože se Arduino restartuje pokaždé, když se na něj pokusí pythonový script připojit. To znamená, že datum a čas budou relativně přesné po 50 dní od posledního zapnutí počítače.

10 Znázornění integer overflow

## Prezentovatelné šasí

Ne, že by se mi nelíbilo neustále mít na stole porci barevných makaronů s měděnou náplní, ale přeci jen by bylo hezčí mít místo toho úhlednou plastovou krabičku. Plánoval jsem tedy pomocí školní 3D tiskárny vyrobit nějakou krabičku do které bych všechno hezky naskládal a ze které by vedl jen jeden až dva kabely. Na tohle by se byla hodila i praxe u Prusa Research, která ale nakonec nevyšla z kapacitních důvodů poskytovatele.

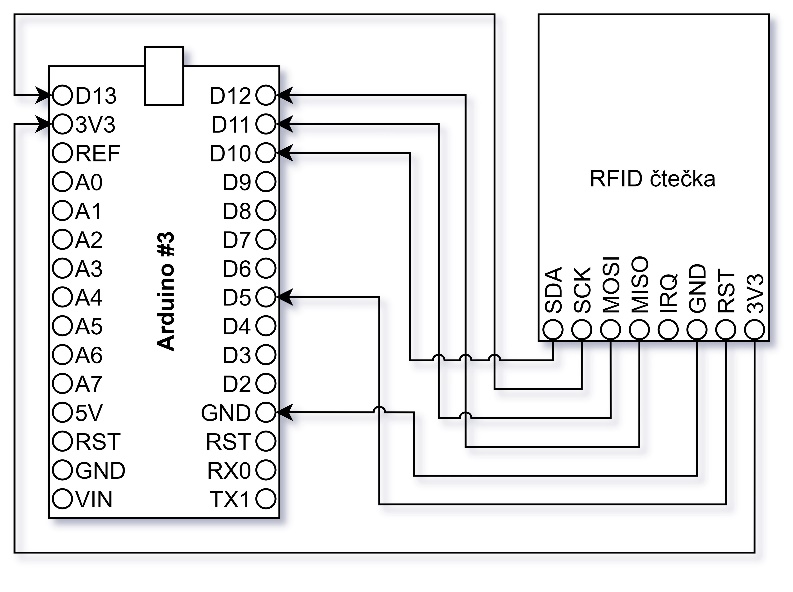
11 Concept art šasí vytištěného 3D tiskárnou, září 2022

# Průběh práce

Hned jak jsem dostal nápad na toto téma maturitní práce, ještě předtím než jsem věděl kdo mi bude dělat vedoucího, nebo jestli mi téma vůbec bude přijato, začal jsem pracovat na komunikaci mezi Arduinem a počítačem.

Protože byla komunikace potřeba víceméně jen kvůli získávání dat o času, které pak hlavní Arduino označí na displeji, nedávalo by smysl, aby komunikovalo jakékoliv jiné Arduino, než to, ke kterému je připojen displej.

## Začátek zabezpečování

Po nějaké době jsem se rozhodl, že Arduino, které bude posílat heslo do počítače, nebude ukládat heslo v plaintextu, ani v jednoduše dešifrovatelném formátu, aby bylo těžší jej prolomit. Rozhodl jsem se použít čtečku čipových karet a to heslo uložit tam, v zašifrované podobě.

Mělo to ale jeden háček. Čtečka čipových karet se k Arduinu připojuje k pinům D5, D10, D11, D12 a D13. Tři z těchto pinů ale byly už na audio modulu použity ke komunikaci se čtečkou micro SD karet, a displejové Arduino potřebovalo komunikovat s počítačem pomocí sériové komunikace, aby získalo aktuální datum a čas. Tou dobou jsem jako displejové Arduino používal moje jediné Uno, ale Uno se, jako jediný z modelů, které vlastním, dá přeprogramovat tak, aby se tvářilo jako klávesnice podle standardu HID (Human Interface Device).

12 Schéma připojení čtečky RFID čipů k Arduinu

Tento problém ale nebyl neřešitelný. Stačilo to Uno nahradit dalším Nanem, a k tomu, teď už volnému, Unu připojit čtečku čipů, a nahrát na něj klávesnicový firmware. Alespoň to jsem si myslel, že to vyřeší.

## Problémy s upraveným firmwarem Una

Běžné Arduino se po připojení k počítači tváří jako že je připojené přes sériový COM port. To dovoluje počítači i Arduinu posílat si jak příkazy, tak i běžný text. Arduino k tomuto účelu používá *Serial.write()*, pokud posílá byty anebo *Serial.print()*, popřípadě *.println()*, chce-li poslat text, nebo čísla.

Na takto poslané příkazy by mohl na počítači čekat nějaký program, který by je zpracoval a poslal operačnímu systému odpovídající vstupy. Tento program se ale nemusí spustit například v nouzovém režimu, což by komplikovalo případné servisní zásahy.

Když na Uno nahrajeme speciální firmware, pořád se k počítači připojí přes virtuální sériový port, ale místo plaintextových komunikací posílá buffer ve kterém je uvedeno která klávesa je právě stisknuta. A právě v tom jednotném tvaru slova *klávesa* je problém. Nepodařilo se mi totiž zjistit, jak najednou poslat informaci o stisknutí více než jedné klávesy. To je potřeba hlavně pro zadávání speciálních symbolů pomocí pravého altu nebo jednoho ze shiftů.

Jak to, že toto použití Arduina není popsané? To je velmi jednoduché. Existují totiž jiné modely Arduina, které podporují komunikaci podle standardu HID bez dalších úprav firmwaru. Zákazníci si tedy radši koupí o trochu dražší model Arduina, než aby se hmoždili s tou levnější.

Rozhodl jsem se tedy kapitulovat a objednal jsem si z Číny imitaci Arduina Leonardo, což je víceméně to samé jako Uno, ale má o několik pinů víc, o zhruba 500B větší RAM a, jak už jsem zmiňoval, jednodušeji se z něj dělá klávesnice.



## Haló, tady Windowsovi!

Když jsem čekal na příchod Leonarda, neměl jsem šanci se nudit. Byla ještě spousta věcí, které jsem neměl hotové. Například jsem naimplementoval komunikaci Arduina s počítačem pomocí jednoduchého pythonového scriptu.

Tento pythonový script se aktivuje po přihlášení do Windows, detekuje všechny dostupné COM porty a připojí se na ně. Arduina, ke kterým se script připojí se automaticky restartují a ohlásí, které Arduino jsou. Pythonový script právě na toto čeká, a jakmile detekuje, který port patří Arduinu s obrazovkou, pošle mu zprávu, pomocí které mu nejen sdělí, že spojení bylo navázáno, ale také mu řekne, který spořič obrazovky má použít podle toho, jak si to uživatel nakonfiguroval (k tomu se dostaneme později). Pokud si uživatel zvolil, že Ino bude zobrazovat jen jednoduchou animaci připomínající spořič starého DVD přehrávače, skript pošle jen informaci o tom, co má Ino zobrazovat.

Pokud si ale uživatel zvolil zobrazování času nebo data, musí skript poslat Inu i čas, popřípadě datum. Aby bylo zpracovávání těchto informací pro Arduino co nejjednodušší, posílám mu jen počet sekund od minulé půlnoci. U data je to složitější, protože počty dnů se v jednotlivých měsících mění. Tady raději posílám celé datum ve formátu ANSI, tj. YYYY-MM-DD.

### Pčtrosí řešení na přestupný problém

Pokud vás zajímá, jak řeším přestupné roky, tak na to mám jednoduchou odpověď: neřeším. Roky jsou přestupné jen jednou za čtyři až osm let, a implementace systému, který by bral v potaz a počítal s devětadvacátým únorem mi přišla jako zbytečně vynaložené úsilí vzhledem k výsledku. Uložené datum na Arduinu se stejně aktualizuje při příštím restartu počítače, přinejhorším bude Arduino ukazovat jeden den špatné datum.

Rád bych podotkl, že tento způsob řešení vzácných problémů se využívá i v seriózních situacích. Moderní operační systémy běžně řeší deadlocky tím, že je prostě neřeší. Pokud si například dva programy chtějí přečíst ten samý soubor, není v tom problém, ale pokud pak budou chtít do toho samého souboru zapisovat, OS jim to nedovolí, protože by měnili data, která ten druhý program pořád ještě čte. Čeká se tedy buď, že programy budou navrženy tak, aby k této situaci vůbec nedošlo, anebo když už k ní dojde, tak že jeden z programů ustoupí dobrovolně, anebo zasáhne uživatel a problém vyřeší.[[5]](#footnote-5)[[6]](#footnote-6)[[7]](#footnote-7)[[8]](#footnote-8)

13 Někdy je prostě výhodnější předstírat, že všechno je v pořádku.

### Failsafe design komunikace

Stejně jako při implementaci komunikace mezi Arduiny, i tady jsem se snažil, aby když se něco pokazí, tak aby zbytek pořád fungoval. U počítače to bylo jednoduché: když nenajde obrazovkové Arduino ani při třetím pokusu, usoudí, že není zapojeno, a nemá tedy cenu se ho snažit kontaktovat a skript se ukončí.

U Arduina to také bylo jednoduché. Vyšli zprávu, které Ino jsi, a pokud se ti nikdo neozve, asi není zapnutý počítač a nemá cenu na zprávu čekat. Jako spořič obrazovky se použije odrážející se ikonka, protože ta jako jediná nepotřebuje k fungování žádnou informaci z vnějšího světa.

### Ostatní vlastnosti skriptu

Aby bylo jednodušší detekovat a řešit problémy v kódu, skript si v uživatelově složce v *C:/Users/* vede log do souboru *enhanced-power-button-log.txt*. Do tohoto souboru se mimo jiné ukládají také veškeré příchozí komunikace od všech připojených Arduin.

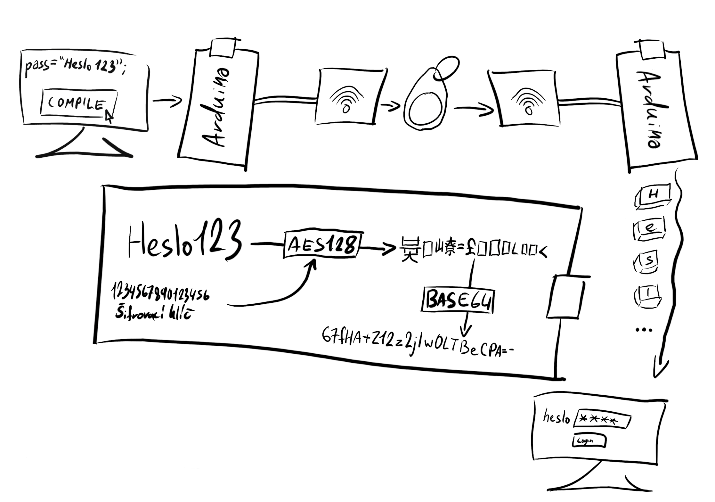
V té samé složce si uživatel může vytvořit svůj enhanced-power-button-config.txt, kam uloží svou preferenci spořiče obrazovky. Pythonový script tento soubor vždy zkontroluje a pošle do Arduina odpovídající instrukce.

14 Náhled do logu scriptu

## Relativně Frantické Inkódovací Dilema

Leonardo bylo ještě stále na cestě, a tak jsem se pustil do něčeho, co můžu testovat na normálním Nanu: program na zašifrování hesla a jeho následné zapsání na RFID čip.

### Jak zapsat zašifrované heslo na čip?

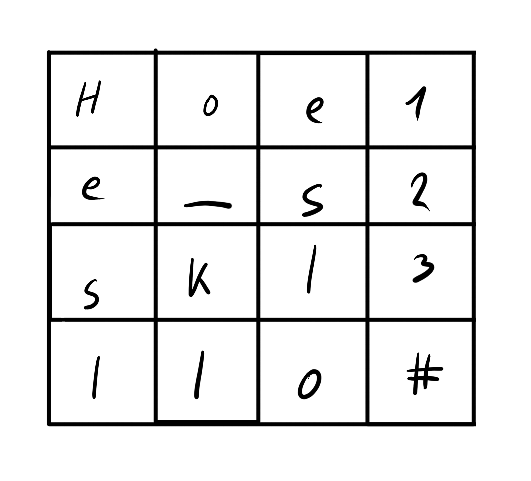
Když se zašifruje nějaký kus textu, výsledek bývá člověkem nečitelný, a to je dobře! Dělá to ale problémy při ukládání na čip. Některé znaky mají totiž speciální účel. Například znak s hodnotou nula označuje konec Stringu, což je kus textu, který může mít libovolnou délku. Když má program vypsat nějaký kus textu, je mu řečeno „tady začni, a přestaň až když narazíš na ␀“.

Výsledek šifry ale může obsahovat i právě tento null terminator, takže aby to program neinterpretoval jako předčasný konec textu, zakódoval jsem celý výstup ze šifrování do Base64 a teprve zakódovaný výsledek teprve zapisuji na čip.

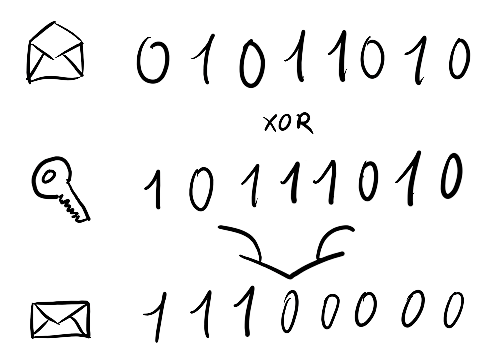
15 Grafické znázornění šifrovacího koloběhu s detailním znázorněním samotného šifrování

Base64 je způsob zapisování jakýhkoli dat jako sérii písmen, číslic a znaků +, / a =. Hlavní účel kódování do b64 je zajištění jednoduchého ukládání. Můžeme si to představit, jakože se vstup rozepíše na dlouhý řádek jako jedničky a nuly, a každých 24 bitů, tzn. 24 jedniček nebo nul zapíšeme jako čtyři znaky, podle speciální tabulky tak, aby každé 3 znaky dekódovaného textu byly zapsané jako 4 znaky enkódovaného textu. Tímto sice plýtváme úložiště, ale zajistíme, že data budou přečtena správně a to i člověkem.[[9]](#footnote-9)

### AES aneb 16B k bezpečnosti

K šifrování hesla jsem si vybral Advanced Encryption Standard, což je velmi široce používaná symetrická bloková šifra. Symetrická je proto, že používá stejný klíč na šifrování i dešifrování, na rozdíl od třeba RSA, která používá dva, jeden veřejný na zašifrovávání a druhý, tajný, k dešifrovávání. Bloková je šifra AES proto, že velké kusy dat nešifruje celé, ale rozdělí si je na 16-ti bytové bloky, které šifruje jeden po druhém.

#### Princip šifrování

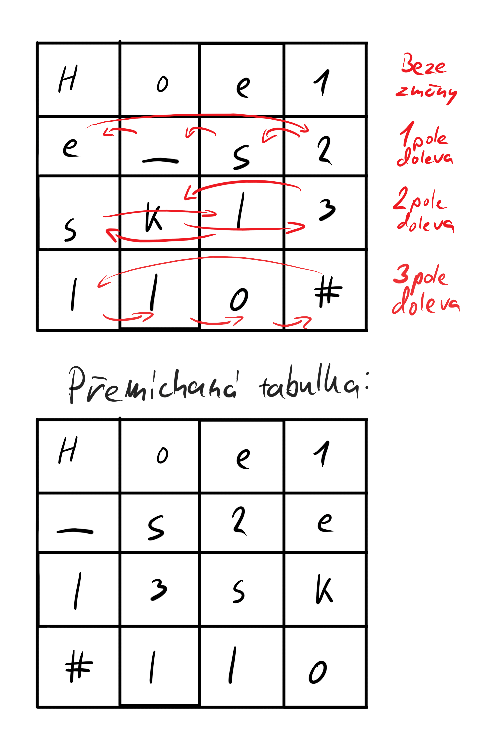
AES je relativně jednoduchá na pochopení. Jak už jsem zmiňoval, rozdělí si zprávu na bloky o 16-ti bytech, to je 128 jedniček a nul, chytrým způsobem změní jejich hodnoty a vyplivne nám stejně velkou, ale bez klíče nečitelnou zprávu.

16 Znázornění rozložení zprávy „Heslo\_Kleslo123#“ do tabulky

Jak to ale dělá? AES si nejprve vytvoří několik podklíčů z našeho hlavního klíče. Poté seřadí tuto zprávu do tabulky po sloupcích shora dolů, zleva doprava. S tímto rozvržením bude pracovat po celou dobu šifrování. Poté vezme náš první podklíč a na zprávě s ním udělá operaci XOR, tzn. porovná všechny jedničky a nuly bloku s jedničkami a nulami klíče, a pokud jsou stejné, zapíše nulu, pokud nikoliv, zapíše jedničku.

18 Příklad provedení operace XOR na 1 bytu zprávy a klíče

Dále nahradí každý z těchto bytů jiným podle speciálně navržené tabulky. Můžeme si to představit jako kdybychom místo každého písmena *A* psali písmeno *F* a místo každého *F* třeba *R* a tak dále.

Další krok je promíchání v rámci řádků. Každý byte se posune vlevo o takové množství políček, jako je číslo řádku ve kterém je. Protože jsme v počítačovém světě, nesmíme zapomenout, že začínáme na nule, tzn. první řádek se vůbec nepohne, druhý se pohne o jedno políčko, třetí o dvě a čtvrtý o tři.

17 Znázornění promíchání řádků. Nutno poznamenat, že zpráva by už touhle dobou byla pozměněna předchozími kroky.

Poslední dva kroky jsou vynásobit každý sloupec speciální maticí a přidá další klíč operací XOR. Nyní ale přichází důvod, proč je tahle šifra tak bezpečná: všechny tyhle kroky se několikrát opakují. Kolikrát záleží na velikosti klíče. Klíč velký 16 bytů, čili 128 bitů, znamená 10 iterací, 24 bytový (192 bitový) iteruje 12x a 32 bytový (256 bitový) iteruje celkem čtrnáctkrát.[[10]](#footnote-10)[[11]](#footnote-11)[[12]](#footnote-12)

#### A co když budu prostě zkoušet všechny možné kombinace?

I kdybyste dedikovali celý svůj život údržbě počítače, který by dokázal vyzkoušet jeden klíč každou nanosekundu a využili všechny známé zranitelnosti této šifry, pořád by vám to trvalo 2 891 184 772 957 889 100 001 let[[13]](#footnote-13). A to je jenom v případě, že používáme AES128. Kdyby se nám i to zdálo nedostatečné, můžeme použít AES256, který nabízí tak absurdně vysokou úroveň bezpečnosti, že i kdyby existoval počítač, který by dokázal vyzkoušet jeden klíč za jedinou Planckovu jednotku času, což je nejmenší možný časový úsek, který je fyzicky možné změřit, trvalo by to přibližně 65 299 601 948 325 717 200 000 000 let[[14]](#footnote-14). Jen tak pro představu, slunce vyhasne už za nějakých 5 miliard, čili 5 000 000 let.[[15]](#footnote-15)

## Struktura práce

1. Titulní strana (strana 1 - započítává se do číslování stran, ale číslo strany se neuvádí)
2. Abstrakt, Abstract, Klíčová slova (nečísluje se, ale započítává se do číslování stran)
3. Prohlášení (strana 3 - započítává se do číslování stran, ale číslo strany se neuvádí)
4. Obsah
5. Úvod
6. Vlastní text
7. Závěr
8. Seznam použitých zdrojů
9. Seznam příloh
10. Přílohy

## Titulní strana

Píše se na samostatný list (viz vzor). Čísluje se, ale číslování se neuvádí .

## Abstrakt, Abstract, Klíčová slova

Píše se na samostatný list (viz vzor). Čísluje se, ale číslování se neuvádí.

Abstraktem se rozumí stručný obsah práce prezentovaný v několika větách. Bývá zpracován v jazyce práce a v anglickém jazyce. Pod abstraktem jsou uvedena klíčová slova, která jednoznačně charakterizují náplň práce. Klíčových slov by mělo být 4 - 6.

## Prohlášení

Text prohlášení se píše na samostatný list, umisťuje se v dolní části stránky (viz vzor).

## Obsah

Maturitní práce má svou strukturu (viz výše). Vlastní text je pak členěn do kapitol, podkapitol. V obsahu jsou uváděny názvy jednotlivých částí maturitní práce, názvy kapitol a podkapitol a číslo stránky, na které se nacházejí. Čísluje se, ale číslování se neuvádí.

## Úvod

V úvodu se přesně vymezí problém, kterým se práce zabývá, vysvětlí se, k čemu má práce sloužit, proč je napsána. Dále má být uvedena stručná charakteristika zkoumaného problému, má být formulován cíl a metody práce. Úvod by měl obsahovat i informaci o struktuře práce.

Délka úvodu bývá zhruba 200 slov.

## Vlastní text práce

Podrobně charakterizuje řešenou problematiku.

Obvykle se skládá z teoretické a praktické části. Teoretická část by měla tvořit pouze stručný vhled do problematiky a neměla by přesáhnout jednu polovinu vlastního textu práce.

V praktické části student uvede podrobný postup a výsledky vlastní práce. Popíše jednotlivé kroky řešení a výsledky, kterých dosáhl.

Délka práce je minimálně 25 stran (započítává se obsah, úvod, vlastní text práce a závěr).

Obsah se neoznačuje číslem, ale počítá se.

Na každé straně má být asi 33 řádků textu a cca 66 znaků na řádek.

Tabulky, grafy a obrázky menší než půl stránky, mohou být zařazeny do textu. Pokud jsou větší, budou zařazeny do příloh. Je-li tabulek, grafů, obrázků v textu více číslujeme je průběžně např. Obr. 1, Tabulka 4, Graf 2.

## Závěr

V závěru může autor rekapitulovat výsledky praktické části maturitní práce, vyhodnotit naplnění cílů a poukázat na jejich využitelnost v praxi, případně navrhnout možnost dalšího pokračování práce. Může též stručně rekapitulovat, co se během práce naučil, s čím se potýkal a jaký přínos pro něj psaní práce mělo. Závěr má mít spíše zevšeobecňující charakter.

## Seznam použitých zdrojů

Seznam použité literatury a zdrojů informací je zařazen na konci práce. Je rozdělen na dvě části. Na seznam použitých literárních publikací (knihy,časopisy a podobně) a na jiné zdroje informací (internet, CD média a podobně). Uvádíme údaje zpravidla v tomto sledu:

* Příjmení a jméno autora, autorů
* Název publikace
* Pořadí vydání
* Místo vydání
* Nakladatel
* Rok vydání
* ISBN

Např.

NOVÁKOVÁ Jana. Formální úprava diplomové práce. 1. vyd., dotisk. Ostrava : Ostravská Univerzita,1998. ISBN 80-7042-141-X.

Příklad uvedení online zdroje (stránka webového portálu):

WESTCOM. O nás. Webnode.cz [online]. ©2008-2011 [cit. 2011-04-26]. Dostupné z: <http://www.webnode.cz/o-nas/>

Formát citací a uvedení použitých zdrojů vychází z ČSN ISO 690.

## Seznam příloh a přílohy

Přílohy vkládáme na konec práce a v textu na ně odkazujeme (např. na přílohu 1). Přílohy číslujeme za sebou. Do příloh zařazujeme takový materiál, který je důležitý pro pochopení celé práce, jehož uvedení v textu by však čtenáře rušilo. Jedná se o pomocné tabulky, grafy, obrázky, texty použitých dotazníků, fotografický materiál, fotokopie archiválií apod. Přílohy musí být okomentovány.

## Základní formátování

* Tisk: jednostranný
* Formát stránky: A4
* Orientace stránky: na výšku
* Řádkování: 1,5
* Typ písma: Arial, Times New Roman
* Velikost základního písma: 12
* Mezera mezi odstavci: 6b
* Zarovnání: do bloku
* Hlavní kapitoly začínají na nové stránce.
* Nadpisy mohou být číslovány podle zásad desetinného třídění. Desetinné třídění se vkládá automaticky, jako formátovací prvek stylu nadpisu. Za tečkami v desetinném třídění se nepíše tečka, za poslední číslicí není tečka, mezi poslední číslicí a vlastním nadpisem jsou dvě mezery.
* Poznámky pod čarou se číslují v textu průběžně výše položenými arabskými číslicemi.
* Tabulky a obrázky se číslují průběžně v celém dokumentu.
* Stránky se průběžně číslují v zápatí stránky, a to ve středu stránky. Titulní stránka, abstrakt a stránka s čestným prohlášením se číslují, ale číslování se neuvádí (číslování stránky se tedy uvádí od úvodu po seznam příloh).
* Okraje: levý okraj: 3 cm, pravý okraj: 2,5 cm, horní okraj: 2,5 cm, dolní okraj 2,5 cm
* Text se formátuje pomocí stylů.
* Nepoužívají se více než tři různé velikostí písma. Zdůrazňování částí textu se provádí změnou sklonu písma. Podtrhávání, prokládání nebo tučné písmo používáme jen ve zvlášť odůvodněných případech.
* Tabulky, grafy a obrázky menší než půl stránky mohou být zařazeny do textu. Pokud jsou větší, budou zařazeny do příloh. Je-li tabulek, grafů, obrázků v textu více, číslují se průběžně, např. Obr. 1, Tabulka 4, Graf 2.

## PowerPointová prezentace (či google prezentace)

Prezentace musí obsahovat minimálně 10 + 2 počítačové stránky (slide).

10 slides – odborný text, grafy, tabulky a fotografie

2 slides – iniciály (jméno, školní rok, škola, obor) a úvod nebo závěr práce.

Student přinese k maturitní zkoušce na obhajobu danou prezentaci na nosiči flash a zároveň ji odevzdá vedoucímu práce v elektronické podobě 14 dní před termínem obhajoby maturitní práce.

Prezentace neobsahuje teoretickou tématiku obsaženou v maturitní práci, ale je zaměřena představení praktické části maturitní práce. Text se píše stručně, případně formou odrážek.

## Obhajoba maturitní práce a názorné ukázky praktické části maturitní práce

V případě, že student k obhajobě maturitní práce potřebuje názorné ukázky (HW, SW, elektronická zařízení), je povinen si zajistit veškeré potřebné pomůcky a ověřit jejich funkčnost v prostorách školy před samotnou obhajobou maturitní práce, a to v součinnosti s vedoucím práce či pověřeným pracovníkem školy nejpozději 14 dní před termínem obhajoby maturitní práce.

## Způsob odevzdání maturitní práce

Práce je odevzdávána ve dvou výtiscích v kroužkové vazbě. Originál je uložen v archivu školy. Druhý výtisk je uložen u vedoucího práce.

Termín odevzdání - do 24. 3. 2023 vedoucímu maturitní práce nebo na sekretariát školy.

Elektronická verze maturitní práce ve formátu pdf je zaslána na e-mail vedoucího práce či uložena do prostoru úložiště školy k tomuto účelu určenému. Zdrojové kódy či jiné důležité součásti praktické části maturitní práce jsou odevzdány v elektronické podobě do prostoru úložiště do 24. 3. 2023

Tištěná a elektronická verze maturitní práce se musí shodovat.

Prezentace je odevzdána v elektronické formě na e-mail vedoucího práce či do určeného prostoru úložiště nejpozději 14 dní před termínem konání obhajoby maturitní práce.

Vypracoval: Ing. Bc. Jaroslav Solfronk a Ing. Pavla Cidlinská

Schválil: Mgr. Jakub Pour, MBA

ředitel školy

1. FALSTAD, Paul. PNP Transistor (Bipolar). *Falstad* [online]. [San Diego], [cit. 2022-10-21]. Veřejně dostupné z: https://www.falstad.com/circuit/e-pnp.html [↑](#footnote-ref-1)
2. Why Do Calculators have a Headphone Jack?. *YouTube* [online]. [United States], 30.6.2019 [cit. 2022-10-22]. Veřejně dostupné z: https://www.youtube.com/watch?v=W\_mZ7smIz3U [↑](#footnote-ref-2)
3. Joris van Rantwijk. Arduino clock frequency accuracy. *Joris\_VR.* [online]. [Paris]: [Gandi], 2012-06-15, 2012-12-19 [cit. 1.11.2022]. Veřejně dostupné z: http://jorisvr.nl/article/arduino-frequency [↑](#footnote-ref-3)
4. Při vlastním testování jsem zjistil, že některá moje Arduina jsou schopna se odchýlit i o 2 minuty každých 8 hodin. Přisuzoval bych to neoficiálnímu původu mého hardwaru. [↑](#footnote-ref-4)
5. Wikimedia Foundation, Inc. Pštrosí algoritmus – Wikipedie. *Wikipedie: otevřená encyklopedie.* [online]. 15.9.2006, 9.8.2021 [8.11.2022] Veřejně dostupné z <https://cs.wikipedia.org/wiki/P%C5%A1tros%C3%AD_algoritmus> [↑](#footnote-ref-5)
6. Wikimedia Foundation, Inc. Ostrich algorithm – Wikipedia. *Wikipedia: The Free Encyclopedia.* [online]. 8 March 2005, 1 October 2022 [8.11.2022] Veřejně dostupné z <https://en.wikipedia.org/wiki/Ostrich_algorithm> [↑](#footnote-ref-6)
7. Wikimedia Foundation, Inc. Deadlock – Wikipedie. *Wikipedie: otevřená encyklopedie.* [online]. 10.2.2006, 4.11.2022 [8.11.2022] Veřejně dostupné z <https://cs.wikipedia.org/wiki/Deadlock> [↑](#footnote-ref-7)
8. Wikimedia Foundation, Inc. Deadlock – Wikipedia. *Wikipedia: The Free Encyclopedia.* [online]. 8 August 2005, 25 October 2022 [8.11.2022] Veřejně dostupné z <https://en.wikipedia.org/wiki/Deadlock> [↑](#footnote-ref-8)
9. Wikimedia Foundation, Inc. Base64 - Wikipedia. *Wikipedia: The Free Encyclopedia.* [online]. 24 April 2003, 21 November 2022 [21.11.2022] Veřejně dostupné z <https://en.wikipedia.org/wiki/Base64> [↑](#footnote-ref-9)
10. TODO: CITACE <https://www.youtube.com/watch?v=O4xNJsjtN6E> [↑](#footnote-ref-10)
11. TODO: CITACE <https://en.wikipedia.org/wiki/Advanced_Encryption_Standard> [↑](#footnote-ref-11)
12. TODO: CITACE <https://cs.wikipedia.org/wiki/Advanced_Encryption_Standard> [↑](#footnote-ref-12)
13. Podle výše citované stránky <https://en.wikipedia.org/wiki/Advanced_Encryption_Standard> existuje způsob jak získat klíč k šifře AES128 za pouhých 2126.1 pokusů. To je 9.1176403e+37, neboli 91 176 403 a 30 nul. Když tohle číslo budeme považovat za počet nanosekund, můžeme jej převést na roky, pokud jej vydělíme číslem 3.154e+16. [↑](#footnote-ref-13)
14. <https://en.wikipedia.org/wiki/Advanced_Encryption_Standard> zmiňuje i počet pokusů k prolomení šifry AES256: 2254.4, čili 3.8197144e+76, neboli 38 197 144 a 69 nul. Jeden kalendářní rok je 5.85e+50 Planckův jednotek času. Pokud tyto dvě čísla vydělíme, vyjde nám počet let, které bychom museli strávit dešifrováním, abychom AES256 prolomili. [↑](#footnote-ref-14)
15. TODO: CITACE <https://en.wikipedia.org/wiki/Sun> [↑](#footnote-ref-15)