Le génie pour l'industrie

Rapport d'étape

PAR
JORA, Alexandru (JORA09019100)

Sigle du cours: ELE792
Professeur responsable: Thibeault, Claude
SESSION: Hiver 2016

MONTRÉAL, le 19 février 2016

1. Rappel du contexte, de la problématique et des objectifs visés

1.1 Contexte

L'objectif de ce projet est de concevoir et fabriquer un gestionnaire de mots de passe matériel à interface USB nommé StickPass. Ce projet est développé dans le cadre du cours ELE792 Projet de fin d'études en génie électrique durant la session d'hiver 2016 à l'École de Technologie Supérieure de Montréal.

1.2 Problématique

La grande majorité des applications informatiques sécurisées utilisent d'une façon ou d'une autre la protection par mots de passe. Cette méthode comporte quelques défaillances qui peuvent facilement être évitées par un utilisateur expérimenté, cependant des études prouvent que la grande majorité des usagers utilisent les mots de passe d'une façon qui les laisse vulnérables à des attaques potentielles. Pour que les mots de passe soient efficaces, il faut que l'utilisateur suivent quelques règles simples: les mots de passe doivent être différents pour chaque application, les mots de passe doivent être longs, les mots de passe doivent être complexes et finalement ils doivent être difficiles à deviner. Bien que ces règles soient simples, lorsque combinées ensembles elles rendent le processus de gestion des mots de passe embêtant et laborieux.

Il existe des logiciels de gestion de mots de passe qui permettent aux utilisateurs d'emmagasiner leur identités et de simplement les utiliser sans avoir à se rappeler de leur mots de passe. Ces solutions sont efficaces, mais malheureusement comportent quelques défauts. Premièrement, il y a le fait que tous les mots de passe de l'usager sont localisés chez un même vendeur (possiblement sur un même serveur), ce qui peut effrayer certains individus. Deuxièmement, ce genre de logiciel requiert l'installation d'une application usager, souvent sous la forme de «add-on» pour le navigateur web, sur chaque ordinateur où l'utilisateur prévoit utiliser le logiciel. Ceci peut parfois causer des problème sur des ordinateurs d'entreprise où les usagers ont des accès limités. Troisièmement, la plupart de ces applications nécessitent l'accès à l'internet pour fonctionner. Si l'usager utilise un logiciel qui ne nécessite pas l'accès à l'internet, il doit alors le configurer sur toutes ses machines et les synchroniser ce qui ramène le problème initial de complexité d'utilisation.

1.3 Objectifs visés

Dans le cadre de ce projet, l'objectif principal est de concevoir un gestionnaire de mots de passe matériel à interface USB. Les principaux objectif sont les suivants:

1. L'appareil permet à l'usager d'utiliser des mots de passe complexes et différents pour ses applications.
2. L'appareil ne nécessite pas que l'usager mémorise les mots de passe.
3. L'appareil devra être facile à configurer et à utiliser.
4. L'appareil ne devra pas comporter de failles de sécurité sévères qui pourraient compromettre l'information confidentielle de l'usager.

2. Revue de la documentation

Cette section a pour but d'identifier et de faire la synthèse des concepts théoriques pertinents au projet.

2.1 Systèmes embarqués et microcontrôleurs

Le projet StickPass est un système embarqué, c'est à dire un système électronique et informatique dédié à un usage spécifique. Contrairement à un système informatique à usage plus général tel qu'un ordinateur, un système embarqué doit respecter beaucoup de contraintes notamment en ce qui à trait aux ressources disponibles. Le projet StickPass est implémenté sur un microcontrôleur 8bits de type AVR ATTINY85 de la compagnie Atmel ce qui fait en sorte que les ressources disponibles sont très limitées. Ce microcontrôleur en particulier possède 8K de mémoire flash, 512 bytes de mémoire SRAM et 512 bytes de mémoire EEPROM. Le microcontrôleur possède un oscillateur interne, 6 GPIO, un ADC à 10 bits de résolution, 2 timers à résolution de 8bits et finalement fonctionne avec une alimentation de 2.7V à 5.5V[1]. Le logiciel sera implémenté en langage C.

Les spécifications ci-dessus sont les principales contraintes qu'il faudra respecter lors de l'implémentation. Il est impératif que le logiciel ne dépasse pas 8K de mémoire en taille, ce qui veut dire que le code devra être bien optimisé autant au niveau d'utilisation de mémoire qu'au niveau utilisation du CPU. USB

2.2 USB

L'aspect le plus important du projet est qu'il utilise l'interface USB. Cela veut dire que le matériel et le logiciel doivent respecter les contraintes relatives à la spécification USB. Sans entrer dans les détails

1 http://www.atmel.com/devices/attiny85.aspx

complexes, la section qui suit identifie et explique les concepts principaux du protocole USB qui sont utilisés dans ce projet.

2.2.1 Caractéristiques électriques

À la base, la communication USB consiste de 4 lignes dont deux servent à l'alimentation (+5V et GND). Les deux autres sont une paire différentielle (D+ et D-) servant à transférer des données encodées en NRZI à des fins de synchronisation entre l'hôte et le périphérique. Une valeur différentielle '1' est transmise lorsque D + est supérieur à 2.8V et D- inférieure à 0.3V. Une valeur différentielle de '0' est transmise lorsque D- est supérieure à 2.8V et un D + inférieure à 0.3V.

La version 1.1 du protocole USB fonctionne sur deux vitesses différentes soit: Full Speed (12Mbits/s) ou Low Speed (1.5Mbits/s). Le StickPass supporte seulement le mode Low Speed pour des raisons de simplicité. Pour que l'hôte reconnaisse que l'appareil fonction en mode Low Speed il faut connecter une résistance de 1.5K entre le signal D- et 5V.

Puisque le microcontrôleur sélectionné ne possède pas d'interface USB physique, le protocole USB 1.1 sera implémenté dans le logiciel en utilisant la librairie V-USB (virtual USB)[2]. La communication USB est donc abstraite par le logiciel et la seule implémentation matérielle nécessaire est l'utilisation de deux pins du microcontrôleur pour les signaux D+ et D-.

2.2.2 Modes de transfert

Dans la communication USB, c'est l'hôte qui contrôle les opérations et qui gère la bande passante. Pour ce faire, le protocole USB offre 4 types de transferts de données disponibles[3]:

1. Transferts de type contrôle: Ces transferts sont généralement utilisés pour des opérations de commande et d'état. Ils sont initiés par l'hôte et envoyés en rafales afin de bénéficier d'une qualité de service «best effort». La taille du transfert est de taille 8 bytes maximum pour les appareils fonctionnant a 1.5Mbits.
2. Transferts de type interrupt: Ces transferts permettent à l'appareil de générer une interruption sur le bus USB et ainsi signaler à l'hôte qu'il y a des données à envoyer. L'hôte effectue un «polling» constant sur le bus, donc si l'interruption est présente les données seront envoyées. La taille maximale par paquet est de 8 bytes. Contrairement au transfert de type contrôle, la qualité de service est garantie et d'autres essais peuvent avoir lieu à la prochaine période si des erreurs sont détectées.
3. Transferts de type isochronous: Ces transferts servent à envoyer des données continuellement et périodiquement. Ce mode est utilisé pour les données contenant de l'information sensible au temps, donc l'accès à la bande passante sur le bus USB est garantie et il y a présence de détection d'erreur avec le mécanisme CRC.

2 https://www.obdev.at/products/vusb/index.html
3 http://www.beyondlogic.org/usbnutshell/usb4.shtml

4. Transferts de type bulk: Ces transferts servent à envoyer de grandes quantités de données en rafale; par exemple un travail d'impression pour une imprimante. Les données en bulk sont transférées que lorsque le bus USB n'effectue pas d'autres transactions, donc ce transfert est dédié aux données non-sensibles au temps et au délais de livraison.

On note que les mode isochronous et bulk ne sont pas disponibles pour les appareils fonctionnant en mode Low Speed, donc seuls les transferts de type contrôle et interrupt sont utilisés dans ce projet.

2.2.3 USB-HID

Un dispositif d'interface humaine ou HID est un type de d'appareil informatique qui interagit directement avec l'usager. Ces dispositifs respectent la classe HID[4] de la spécification USB afin de garantir un bon fonctionnement avec n'importe quel ordinateur (qui possèdent généralement des pilotes HID préchargés). Des bons exemples de dispositifs HID sont les claviers et les souris.

Le StickPass implémente la classe HID dans son microprogramme, car à la base il agit comme un clavier. Cela va également permettre d'utiliser l'appareil sur n'importe quel ordinateur.

Une particularité du protocole USB est qu'il supporte le «plug and play», c'est à dire que le chargement dynamique de pilote. Lorsqu'un appareil est branché, l'hôte détecte cet événement et interroge l'appareil afin de charger le bon pilote dans le noyau. Cet à ce moment que le StickPass s'énumère en tant que dispositif à interface humaine.

2.2.4 libusb

Du coté de l'hôte, une application usager sera implémenté afin de permettre la configuration de l'appareil. Cette application sera codée en langage C et utilise la librairie libusb pour envoyer des requêtes USB à l'appareil. Cette librairie présente un API du usbcore (pilote du côté hôte qui communique avec tous les appareils USB).

3. Méthodologie de travail

Cette section décrit les choix de conception qui ont été faits et les étapes à réaliser en fonction de ces choix.

3.1 Circuit électronique

Pour des raisons de simplicité, le choix a été fait d'utiliser un microcontrôleur sans interface USB physique et donc d'utiliser une implémentation logiciel de la communication USB. Cela simplifie le circuit électronique en permettant l'utilisation d'un microcontrôleur plus petit, moins puissant et moins dépendant de composantes externes. Puisque la librairie V-USB ne supporte que les appareils en mode

4 http://www.usb.org/developers/hidpage/Hut1_12v2.pdf

Low Speed, la conception électronique est simplifiée également, car il n'est pas requis de respecter les contraintes d'une conception à haute vitesse.

3.1.1 Étapes à réaliser

1. Conception du circuit électronique
   a) Choix d'un microcontrôleur.
      i. Le choix s'est fait sur le microcontrôleur ATTINY85 de ATMEL, principalement du à sa petite taille et la compatibilité avec la librairie V-USB.
   b) Utiliser la documentation de V-USB, plus spécifiquement la section «contraintes matérielles» afin de concevoir le circuit de base.
      i. Consulter des circuits exemples pour mieux comprendre la documentation.
   c) Rajouter les composantes spécifiques au projet, c'est à dire les boutons et LED de débogage.
   d) Assembler le circuit sur une plaque de prototypage et tester le fonctionnement de base.
      i. Utiliser des outils tels que oscilloscope au besoin.
      ii. Implémenter un logiciel de test de base qui fait clignoter une LED.
      iii. Implémenter un logiciel de base qui envoie un message à l'hôte.
   e) Réviser le schéma électronique au besoin.
   f) Concevoir le PCB.
      i. Utilisation du logiciel Kicad.
   g) Fabriquer et assembler le PCB final.

3.2 Conception du microprogramme (firmware)

Le microprogramme est lourdement dépendant des contraintes de la librairie V-USB. L'implémentation doit répondre à des contraintes de timing précises afin de fonctionner correctement.

3.2.1 Étapes à réaliser

1. Configuration de l'environnement de travail
   a) Installer le compilateur AVR et les librairies de développement.
   b) Choisir un environnement de développement pour écrire le logiciel.
      i. L'éditeur de texte VIM a été choisi, car c'est un outil de développement minimaliste et versatile que le programmeur connaît bien.
   c) Écrire un Makefile avec les routines pour compiler, assembler, produire le fichier hex et flasher le microcontrôleur.

2. Implémentation du microprogramme
   a) Incorporer la librairie V-USB.
      i. Lire la documentation.

ii. Configurer les paramètres relatifs au microcontrôleur utilisé.

iii. Choisir une fréquence de CPU compatible.

iv. Calibrer l'oscillateur interne.

v. Tester et valider le fonctionnement de base.

b) Implémenter la classe HID.

i. Lire la documentation.

ii. Implémenter le usb_descriptor.

iii. Implémenter la fonction usbFunctionSetup.

iv. Implémenter la fonction usbFunctionWrite.

v. Implémenter la fonction qui construit les rapports contenant l'information sur les caractères à injecter dans l'hôte.

c) Implémenter le module d'écriture et de lecture dans la EEPROM.

i. Selon l'architecture de mémoire choisie, implémenter des fonctions pour la lecture, l'écriture et la mise à jour de données dans la mémoire EEPROM.

d) Implémenter la protection d'accès aux données.

e) Implémenter la machine à états finis qui représente la logique du microprogramme.

i. Permet l'itération parmi les identités.

ii. Permet d'injecter une identité choisie.

iii. Permet d'interagir avec l'application usager.

iv. Permet de mettre à jour la mémoire avec de nouvelles identités.

3. Implémentation de l'application usager

a) Utiliser la librairie libusb afin d'envoyer des requêtes USB à l'appareil.

b) Implémenter la protection de l'accès à l'appareil.

c) Implémenter la génération de mots de passe complexes.

d) Implémenter un mode de lecture de toutes les identités sur l'appareil afin de créer une sauvegarde locale.

4. Processus de conception

Cette section décrit le processus de conception utilisé pour les tâches déjà réalisées.

4.1 Développement du microprogramme

L'approche générale de conception du microprogramme est de diviser l'effort de développement en blocs distincts afin de faciliter l'intégration par la suite. Ces blocs sont identifiés ci-dessous.

4.1.1 Implémentation des fonctionnalités de base relatives à la communication

La première étape est d'établir que la communication avec l'hôte fonctionne correctement. La librairie V-USB a été intégrée au projet et configurée pour que le microcontrôleur puisse s'énumérer correctement. Par la suite, quelques routines de débogage ont été implémentées afin de faciliter le processus de développement. Ces routines sont: un module pour contrôler la diode électroluminescente ainsi qu'un module pour simuler le fonctionnement d'un port série afin d'envoyer des données de débogage à l'hôte. Le module servant à émuler le port série utilise les deux modes de transferts de données disponibles c'est à dire transferts de type contrôle pour les petits messages et transferts de type interrupt pour un flux de données plus large.

4.1.2 Architecture de la mémoire

Une fois les outils de débogage validés, l'étape suivante a été de concevoir une architecture pour la façon que les identités sont manipulées et organisées en mémoire. Le microcontrôleur possède une mémoire flash de 8k ainsi qu'une mémoire EEPROM de 512 Bytes. La mémoire flash ne supporte que 10 000 cycles d'écriture max alors que la mémoire EEPROM en supporte 100 000. Étant donné que l'encryptage des données sensibles est un requis et assumant une utilisation moyenne de 6 fois par jour, la mémoire flash aurait grossièrement une durée de vie de 4.5 années alors que la mémoire EEPROM aurait une durée de vie de 45 années. Il faut également considérer le fait que 8K de mémoire pour le microprogramme entier est déjà un défi en tant que tel. Pour ces deux raisons le choix a été fait d'emmagasiner les données de l'usager dans la mémoire EEPROM.

Par la suite, il a fallu concevoir un système d'encodage des données dans l'application usager et de décodage dans le microprogramme. Afin de simplifier le processus, la décision a été prise de créer un agencement de grandeur statique dans la mémoire EEPROM qui possède la forme suivante:

```
-------------------------------------------------------------------------------
| idName (10 bytes) | idUsername (32 bytes) | idPassword (22 bytes) |
-------------------------------------------------------------------------------
```

Ce bloc statique de 64 bytes contient un identifiant pour l'identité (idName), le nom d'usager (idUsername) et le mot de passe (idPassword). Les contraintes de longueur pour chaque éléments seront implémentées au niveau de l'application usager. La mémoire EEPROM peut donc contenir 8 identités. Cette architecture n'est pas la plus efficace, mais afin de diminuer le temps de développement elle a été

privilégié à la gestion de mémoire dynamique dans la EEPROM. Les routines d'encodage et de décodage sont donc simplifiées, car il ne suffit que d'envoyer les données dans un tampon de 64 bytes avec du «stuffing» pour les bytes non-utilisés. Finalement les routines d'écriture, de lecture et de mise à jour de données dans l'EEPROM sont implémentées, testées et validées.

4.1.3 Implémentation de la classe USB-HID

Jusqu'à maintenant l'appareil est capable de communiquer, mais n'agit pas en tant que dispositif à interface humaine. Afin d'accomplir cette étape les paramètres de la librairie V-USB sont modifiés afin de permettre à l'appareil de s'énumérer comme HID. Par la suite il faut implémenter un usbHidDescriptor au niveau de l'appareil afin d'indiquer à l'hôte de quel type d'appareil il s'agit, comment ses données sont envoyés à l'hôte et comment celui-ci devrait les interpréter. Puisque le StickPass permet d'injecter des caractère dans des champs de texte, c'est à la base un clavier. Le descripteur pour clavier disponible dans la spécification USB 1.1 est donc utilisé tel quel dans le microprogramme. Puisque ce descripteur est en fait un large tableau statique et constant, il est déclaré avec l'attribut PROGMEM afin que le compilateur le place dans la mémoire flash. Cela économise de la RAM.

Après avoir implémenté le descripteur, il reste à développer la logique d'encapsulation des données dans le format définit dans le descripteur. Cela est effectué à l'aide d'une machine à états finis qui prend un byte à la fois les caractères à injecter et les convertit selon la valeur hexadécimale définie par la spécification HID. Malheureusement la spécification HID définit des codes différents de la table ASCII ce qui rend la routine un peu plus longue.

Rendu à cette étape, l'appareil est apte à injecter des caractères dans les champs de texte du côté de l'hôte.

5. Bilan de l'état d'avancement du projet et mise à jour du plan de travail

Cette section a pour but de donner une mise à jour sur le déroulement du projet.

5.1 Avancement

Les principaux avancements sont énumérés ci-dessous:

1. Conception du circuit électronique complétée.
2. Assemblage du circuit sur une plaque de prototypage complété.
3. Communication USB intégrée et testée.
4. Classe HID implémentée et testée partiellement.
5. Gestion de la mémoire implémentée et testée.

6. Début d'implémentation des requis fonctionnels.
7. Conception du PCB complétée.

5.2 Mise à jour du plan de travail

Planification des tâches	Temps estimé (h)	Temps dépensé (h)	Temps restant (h)
Documentation	**21**	**7**	**14**
Rédaction du rapport d'étape	5	7	0
Rédaction du rapport technique final	8	0	8
Préparation de la présentation orale	8	0	8
Fabrication du circuit électronique (prototype)	**25**	**25**	**0**
Sélection du microcontrolleur	4	4	0
Conception du circuit	6	6	0
Approvisionnement des composantes	3	3	0
Assemblage du circuit	2	2	0
Test de la programmation du microprogramme	6	6	0
Sanity check du circuit	4	4	0
Conception logiciel	**60**	**18**	**42**
Implémentation du pilote HID sur le microcontrolleur	4	4	0
Implémentation du protocole USB virtuel sur le microcontrolleur	8	8	0
Implémentation des requis fonctionnels du système	24	4	20
Implémentation des requis de sécurité du système	12	0	12
Implémentation de l'application usager	12	2	10
Test du logiciel	**44**	**10**	**34**
Test des communications	6	3	3
Test du protocol USB	12	4	8
Test du pilote HID	6	2	4
Test des requis fonctionnels	8	1	7
Test des requis de sécurité	4	0	4
Test de l'application usager	6	0	6
Test de performance	2	0	2
Fabriquation du prototype final	**29**	**12**	**17**
Conception du PCB	8	7	1
Révision du PCB	2	2	0
Approvisionnement des composantes	3	3	0
Fabrication du PCB	4	0	4
Assemblage PCB	4	0	4
Test PCB	8	0	8
TOTAL	**179**	**72**	**107**

Avec l'avancement effectué, il est fort probable que la section *Conception logiciel* ait été mal estimée. Le nouveau temps estimé est de 45 heures au lieu de 60 heures. Les autres estimations ne changent pas.