INTRODUCTION AU REVERSE HARDWARE

Damien CAUQUIL – @virtualabs – dcauquil@quarkslab.com

Philippe TEUWEN – @doegox – pteuwen@quarkslab.com

Ingénieurs R&D Sécurité chez Quarkslab

Nous vous proposons un aperçu de la rétro-ingénierie matérielle, principalement à un niveau « amateur éclairé » avec un matériel de hobbyiste accessible, ainsi que les références utiles pour approfondir les domaines couverts.

Photo d'illustration.

Effectuer des manipulations sur le hardware effraie souvent les reversers les plus chevronnés et pourtant, vous allez voir, ce n'est pas si difficile de s'y mettre. Nous ne couvrirons pas la partie installation d'un espace de travail sécurisé et choix de l'outillage, car cela nous emmènerait bien au-delà du sujet.

Tout comme pour la rétro-ingénierie logicielle, nous aborderons la rétro-ingénierie matérielle selon deux grands axes : statique et dynamique. À la différence du logiciel, ces approches peuvent être plus ou moins destructives et il faudra prendre ce facteur en compte lors de l'acquisition d'exemplaires de test et dans l'ordonnancement des activités de reverse.

Suivant le besoin, il n'est pas nécessaire de comprendre le rôle de chaque composant d'un produit pour en saisir le comportement global et cibler par exemple les parties les plus prometteuses à auditer.

1. ANALYSE STATIQUE

OSINT

La première étape sera de trouver en ligne un maximum d'informations sur le produit ciblé : site du fabricant, démontages (*teardowns*), notamment ceux d'iFixit **[1]**, etc. Parfois, il peut être utile de se rabattre sur des informations d'autres produits de la même famille mieux documentés.

Si le produit a reçu une certification FCC (*Federal Communications Commission*, États-Unis), on trouvera un code alphanumérique inscrit physiquement sur le produit, code qui servira à retrouver le dossier correspondant sur le site de la FCC **[2]**. Notons qu'il est souvent plus aisé de passer par le portail alternatif de Dominic Spill **[3]**. L'intérêt principal est d'accéder aux photos internes de l'appareil (ce qui peut faciliter le démontage) et au manuel du constructeur et d'avoir une idée des interfaces radio existantes.

Analyse macroscopique

Une fois le produit ouvert, on se retrouve typiquement devant un ou des PCB (*Printed Circuit Board*) et des périphériques, généralement aisés à identifier (bloc d'alimentation, capteur, interface standard, écran, boutons, micro, speaker, actuateur...).

Côté PCB, selon le cas, on peut avoir une partie analogique, facilement identifiable par le type de composants utilisés, souvent difficilement compréhensible dans le détail, mais rarement utile à reverser, ouf ! Savoir distinguer les grands blocs fonctionnels comme la régulation de tension, l'émission ou la réception de signaux, l'amplification, les convertisseurs analogiques-numériques (ADC/DAC), etc. devrait largement suffire, une fois qu'on a pu établir les liens avec la partie numérique du PCB en charge de l'intelligence et du contrôle. La figure 1 (page suivante) montre une analyse macroscopique d'un PCB extrait d'un routeur Wi-Fi.

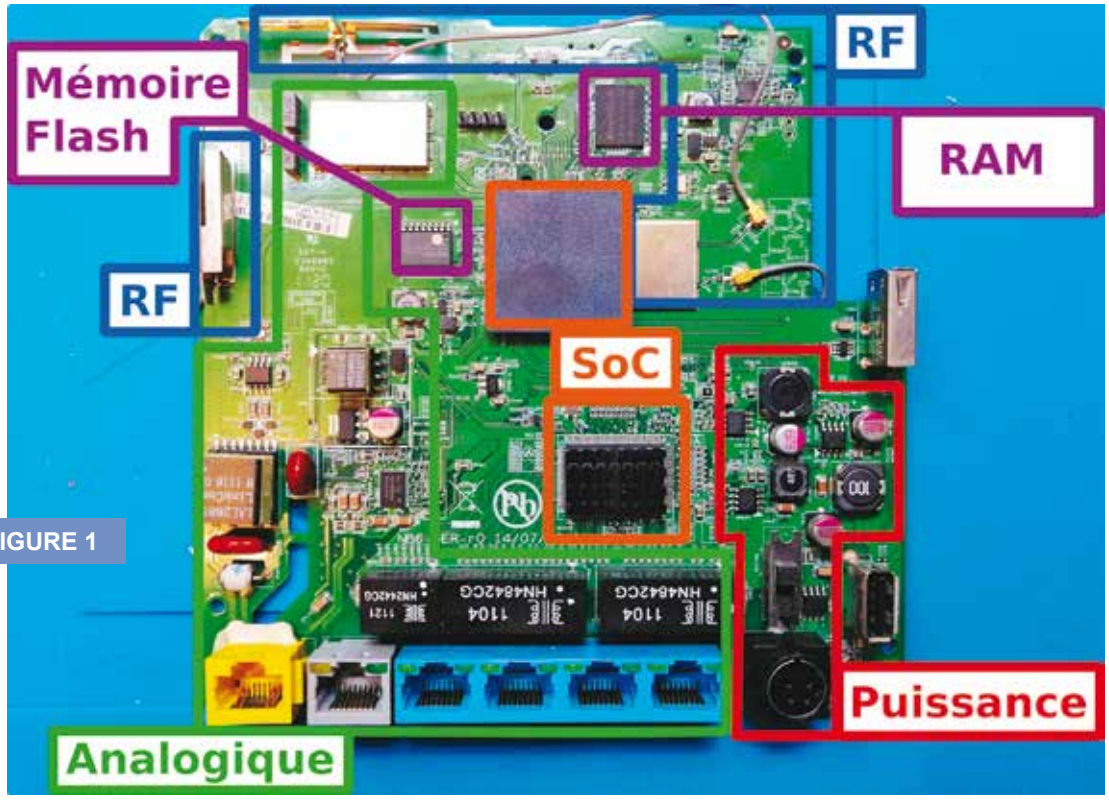

FIGURE 1

Blocs fonctionnels d'un PCB de routeur Wi-Fi.

Identification des composants

Avant tout, il est bon de savoir reconnaître les différents types d'encapsulation aux noms obscurs tels que TO220, SOT23, QFP, TSOP, BGA, etc. Le tutoriel de Sparkfun **[4]** en donnera une introduction et la page Wikipédia dédiée **[5]**, plus exhaustive, sera utile pour les modèles plus rares.

Pour les CMS (composants montés en surface, SMD en anglais) comportant des références laconiques telles que « XYZ », les sites Alltransistors **[6]** et Smdmark **[7]** peuvent faire des miracles.

Pour le reste, on s'attellera principalement à l'identification des puces présentes sur le circuit. Si le composant comporte une inscription et que la référence est connue, on retrouvera sans grand-peine sa *datasheet*, toujours très utile, voire indispensable. Les documents annexes tels que des *Application Notes* peuvent se révéler utiles également.

Si la référence est inconnue au bataillon, on tentera de retrouver le constructeur si un logo est présent, en le comparant si nécessaire aux bases de données en ligne telles que celle de Fandom **[8]** puis de chercher des composants avec une référence similaire dans le catalogue du constructeur.

Si aucune référence n'est visible ou si elle a été volontairement griffée, on tentera d'identifier le *pinout*, c.-à-d. la fonctionnalité de chaque broche du composant, et peut-être ainsi reconnaître une mémoire EEPROM ou un microcontrôleur classique. L'approche dynamique de l'observation des signaux à l'oscilloscope facilitera cette tâche.

On peut également dessouder le circuit, le remettre sur un PCB d'adaptation vers un brochage DIP (*Dual In-line Package*), ou l'insérer dans un adaptateur ZIF (*Zero Insertion Force*) idoine et tenter de l'identifier automatiquement au moyen d'un programmateur universel tel que le TNM5000.

Pour réaliser cette opération sur des formats de puce plus exotiques, notamment certains BGA (*Ball Grid Array*), cela nécessitera un PCB réalisé sur mesure ou la soudure manuelle de micro-fils **[9]**. Pour rendre le produit à nouveau opérationnel, il faut soit remettre la puce en place, soit, si on prévoit de devoir répéter l'opération, souder un adaptateur DIP à même le PCB à l'aide de micro-fils **[10]**. Il sera alors très aisé de passer du produit au programmateur et inversement afin d'observer d'éventuels changements au cours de la vie du produit, voire de changer le contenu de la puce et d'en observer l'impact.

Décapsulation de puce

Si malgré tout on se retrouve devant une puce inconnue à ce stade, il est temps d'envisager des méthodes plus destructives : la décapsulation afin d'observer au microscope le circuit intégré. La méthode la plus rapide et la préférée de Stéphane Marty (auteur de la chaîne YouTube *Deus Ex Silicium* **[11]**, un incontournable si vous vous intéressez au reverse hardware) est la calcination du boîtier au chalumeau de poche, cf. figure 2. Le butane ne chauffe qu'à 1300°C tandis que le silicium fond à 1414°C, tout va bien... Un examen au microscope révélera peut-être une référence, le nom du fabricant, voire celui du commanditaire comme l'illustre la figure 3.

FIGURE 2

Chaud devant ! Source : Stéphane Marty, CC BY-SA.

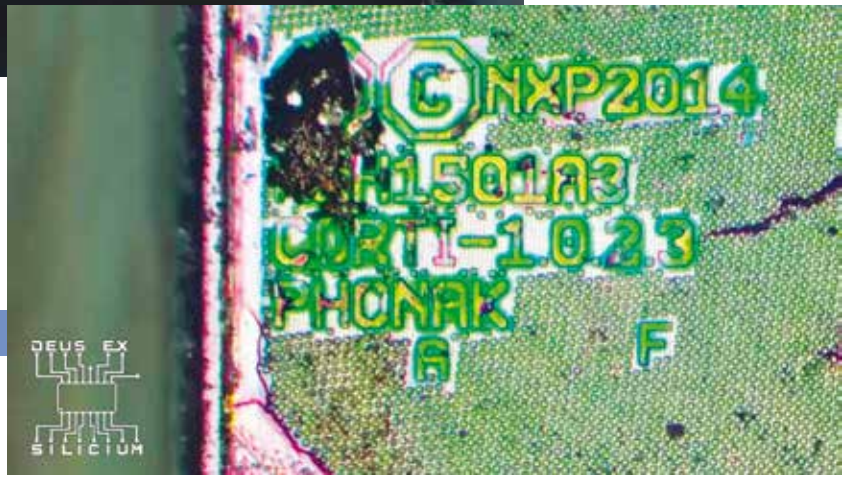

FIGURE 3

Exemple de marquage sur circuit intégré. Source : Stéphane Marty, CC BY-SA.

La méthode laisse néanmoins quelques scories et si un rendu plus propre est nécessaire, on optera pour une dégradation chimique du boîtier à l'acide nitrique ou sulfurique chauffé. Vous l'aurez compris, on entre dans une catégorie de manipulations extrêmement dangereuses, à n'entreprendre que dans un environnement sûr et correctement équipé et en toute connaissance de cause. Et n'envisagez même pas l'acide hydrofluorique si vous tenez à vos os, littéralement.

Certains décapsulent au laser, mais cela nécessite davantage de réglages et de manipulations. Notons que sur certaines encapsulations moins robustes, du type goutte d'époxy, une décapsulation mécanique par grattage couplée à un apport de chaleur grâce à une buse à air chaud sera suffisante.

Si le chip utilise une technologie de gravure suffisamment large pour être observée au microscope métallurgique et comporte une ROM, il est envisageable de la décoder à partir de l'assemblage de clichés à l'aide de logiciels tels que *bitract* **[12]**.

Pour aller plus loin, *Siliconpr0n* de John McMaster **[13]** est une référence sur la décapsulation et l'analyse de circuits intégrés et comporte une large galerie de photos de puces de silicium à très haute résolution.

Le stade ultime est de parvenir à reverser l'intégralité du circuit intégré et à l'émuler sur un FPGA. C'est une des spécialités de Furrtek **[14]** qui participe ainsi à la préservation des anciens jeux d'arcade et de console.

Analyse du tracé du PCB

On repérera les différents éléments destinés à faciliter le débogage du PCB : connecteurs, rangées de points de test, voire points de test isolés, qui donneront peut-être accès à un USART, un port JTAG ou un équivalent propriétaire (par ex. SWD chez ARM). Une analyse au microscope de la surface des points de test révélera des griffures sur les points réellement utilisés lors de l'usinage et de la programmation ou du contrôle du produit.

L'ensemble des lignes constituant un port JTAG est notoirement compliqué à retrouver si le PCB ne donne aucun indice et que de nombreux points de test doivent être testés. On se tournera alors vers des outils tels que le *JTAGulator* **[15]** ou *JTAGenum* **[16]**, plus abordables, mais plus lents.

Suivant le besoin, il peut être nécessaire de reverser le PCB lui-même, partiellement ou en totalité, afin de découvrir comment l'ensemble des éléments sont interconnectés. Une approche non invasive consiste à utiliser un multimètre avec un testeur de continuité, bref si le signal passe, l'appareil émet un signal sonore. Préférer les modèles qui ont une réaction instantanée aux modèles qui mettent ne serait-ce qu'une fraction de seconde à réagir. Cela fait gagner un temps précieux quand on balaye des rangées de pins et cela évite les faux négatifs. Équiper les sondes d'aiguilles pour une plus grande précision si nécessaire. Toutes les pistes ne seront pas forcément accessibles, notamment en présence de BGA. Pour avoir plus d'informations, il faut passer à des méthodes de plus en plus invasives : dessouder les BGA (opération réversible après rebillage du composant), voire dessouder la totalité des composants si on dispose d'un exemplaire à sacrifier.

Si le PCB est réalisé en une ou deux couches, une inspection visuelle suffira à en reconstruire le tracé. S'il comporte des couches internes, un test de continuité entre pads accessibles en surface permettra de progresser, en troquant dans un premier temps les aiguilles contre des surfaces nettement plus larges, par exemple constituées de tresse à dessouder, afin de balayer un grand nombre de contacts à la fois.

Si ces approches sont insuffisantes ou trop laborieuses, envisager le délaminage du PCB couche par couche par ponçage. Mais l'opération, présentée dans *PoC||GTFO 0x06:8 Introduction to Delayering and Reversing PCBs* par Joe Grand **[17]**, reste délicate.

Pour aller plus avant dans le domaine, le lecteur pourra consulter les ouvrages de Ng Keng Tiong : *The Art of PCB Reverse Engineering* pour un niveau de reverse amateur et *PCB-RE: Tools & Techniques* pour un niveau de reverse industriel.

2. ANALYSE DYNAMIQUE

Une fois l'analyse statique d'un circuit imprimé effectuée, on connaît les références et brochages des composants intégrés ainsi que les différents connecteurs et points de test accessibles et leur brochage partiel ou complet. Ces connecteurs, pistes et points de test constituent la surface d'exposition du circuit imprimé. Cette surface d'exposition peut être exploitée afin de capter des informations, de s'interfacer avec un ou plusieurs composants ou encore de réaliser des attaques de l'homme du milieu, à l'instar de ce qu'il est possible de faire au sein d'un réseau local par exemple.

Interfaces de communication

Les communications entre circuits intégrés reposent généralement sur des interfaces standardisées. Ces interfaces permettent, via un ensemble de connexions entre deux ou plusieurs circuits intégrés, de transmettre et recevoir des informations. Celles-ci sont très utiles à la compréhension du fonctionnement d'un système, voire à la mise en place d'attaques sur ce dernier. Il est donc important d'être en mesure de les identifier et de déterminer leurs caractéristiques et celles des composants intégrés qui les emploient.

Il existe trois interfaces de communication majeures entre composants :

- USART (*Universal Synchronous and Asynchronous Receiver Transmitter*) ;
- SPI (*Serial Peripheral Interface*) ;
- IIC ou I²C (*Inter-Integrated Circuit*), qui est d'ailleurs considérée comme un bus plutôt qu'une interface.

Pour identifier ces différentes interfaces, on emploiera un analyseur logique. Cet équipement spécialisé, une fois connecté aux différents points du circuit à tester, en mesure les niveaux logiques, et ce, quelques millions de fois par seconde, afin de tracer un chronogramme affichant le niveau de chaque ligne en fonction du temps. Nous verrons comment repérer visuellement ce qui semble constituer un ensemble de lignes d'interface, pour y appliquer et configurer le décodeur adéquat et reconstituer les données échangées.

Il existe des analyseurs logiques à des prix très raisonnables, mais qui ont une vitesse d'échantillonnage (le nombre de mesures qu'ils sont capables d'effectuer par seconde) limitée (24 Ms/s et 100 Ms/s sur les modèles de la figure 4, page suivante). Ces analyseurs ne peuvent donc pas fonctionner sur des protocoles trop rapides, mais cela suffit amplement dans la plupart des cas. Il faudra par ailleurs faire attention aux niveaux des signaux utilisés sur le circuit ciblé (1,8 V, 3,3 V, 5 V, etc.) et utiliser des outils tels les *level shifters* pour les adapter si nécessaire.

FIGURE 4 *Analyseurs logiques à bas coût compatibles avec Pulseview.*

USART

L'interface USART assure la transmission bidirectionnelle simultanée d'informations entre deux composants, selon des caractéristiques connues de ces deux derniers dont notamment la vitesse de transmission (*baudrate*), le nombre de bits par symbole, le bit de parité et la durée du stop bit. Les équipements sont supposés être configurés tous deux avec les mêmes paramètres pour que la communication soit possible. Les vitesses de transmission sont standardisées de façon à faciliter la compatibilité avec les oscillateurs externes couramment employés en électronique. Ces caractéristiques sont généralement notées sous la forme **`<baudrate> <bits par symbole><parité><stop bit>`**, comme **`115200 8N1`** qui est la configuration la plus répandue.

Une interface USART utilise une ligne RX pour la réception de données, et une ligne TX pour la transmission, cela du point de vue de chaque partie. On aura donc le TX de l'un connecté au RX de l'autre. À ces lignes de communication peuvent s'ajouter des lignes de *signaling*, telles que DTR (*Data Ready*) et CTS (*Clear To Send*), mais elles sont plus rarement employées.

La figure 5 présente une capture réalisée via un analyseur logique et le logiciel open source *Pulseview* **[18]**, dans laquelle on distingue la ligne TX. On déterminera la vitesse de transfert en mesurant la durée de transmission d'un bit à l'aide des marqueurs, comme le montre la capture.

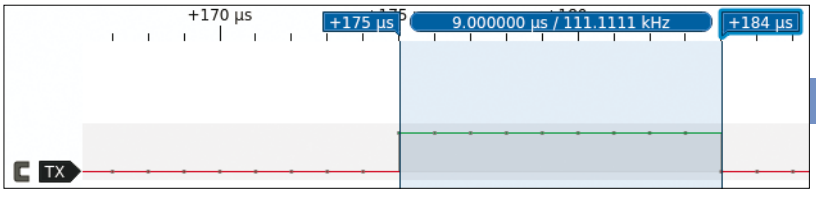

FIGURE 5 *Capture d'une communication USART.*

Le calcul du *baudrate* par Pulseview donne 111 111 bits/s, ce qui correspond à un *baudrate* standard de 115 200 (la différence est due à la vitesse d'échantillonnage de l'analyseur logique ainsi qu'à une possible légère dérive de l'émetteur, tolérée à hauteur de 5 % maximum). Les autres paramètres se déduisent par observation des trames.

Pulseview fournit un décodeur de protocole adapté qui une fois correctement configuré, décode les octets transmis. La figure 6 montre qu'il identifie ainsi le *start bit* (1), l'interprétation de l'octet (2) et le *stop bit* (3) en regard du chronogramme.

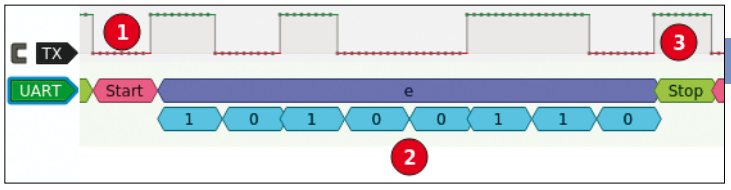

FIGURE 6

Décodage d'une communication USART.

SPI

L'interface SPI assure une communication bidirectionnelle simultanée entre un composant intégré et un ou plusieurs autres composants. Un contrôleur dirige ainsi un ou plusieurs périphériques afin d'échanger des informations via trois lignes de données communes (MOSI, MISO et CLK) et une ligne de sélection (CS) par périphérique, comme le montre la figure 7. Ce type d'interface est utilisé par exemple par des mémoires de stockage Flash de faible capacité (quelques Mo) utilisant de petites empreintes (SOIC8 ou similaire).

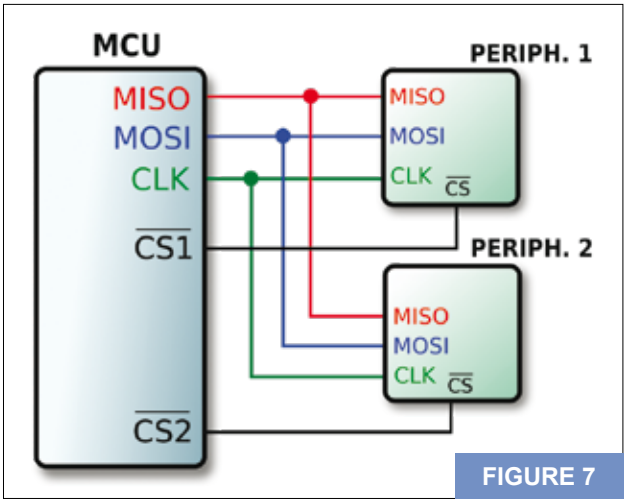

FIGURE 7

Connexion d'un MCU à deux périphériques SPI.

Sur le chronogramme de la figure 8, la ligne CLK est facilement identifiable, car elle transmet des séries de 8 impulsions (3), correspondant à un signal d'horloge indiquant au périphérique le moment où un bit de donnée en provenance du contrôleur est disponible sur la ligne MOSI. Le périphérique transmet simultanément sur la ligne MISO un bit de donnée à destination du contrôleur. La ligne CS d'un périphérique permet au contrôleur de lui notifier le début (1) et la fin (2) d'une transaction. Attention toutefois à la phase et la polarité du signal d'horloge (appelée communément le *mode*), qui conditionnent le fonctionnement de l'interface. Pulseview possède bien sûr un décodeur adapté, qui affiche les différentes transactions et les données échangées.

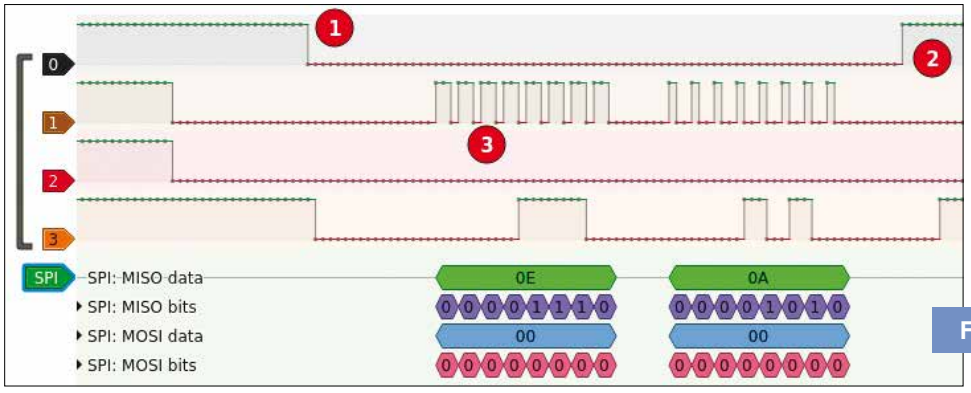

FIGURE 8

Décodage d'une communication SPI sous Pulseview.

Il est par ailleurs possible de reconstruire le contenu d'une mémoire de stockage Flash en interceptant et décodant les communications avec cette dernière lors du démarrage d'un système par exemple.

I²C

À la différence de SPI, I²C définit un bus de communication bidirectionnel *half-duplex*, c'est-à-dire un ensemble de lignes de données interconnectant un ou plusieurs périphériques à un contrôleur et lui permettant de communiquer avec ces derniers au travers dudit bus. Chaque composant présent sur ce bus, à l'exception du contrôleur, possède une adresse propre qui permet de l'identifier, comme l'illustre la figure 9.

Le bus I²C utilise deux lignes SDA et SCL, respectivement la ligne de donnée et celle d'horloge, qui permettent de transmettre les informations d'une manière similaire à l'interface SPI.

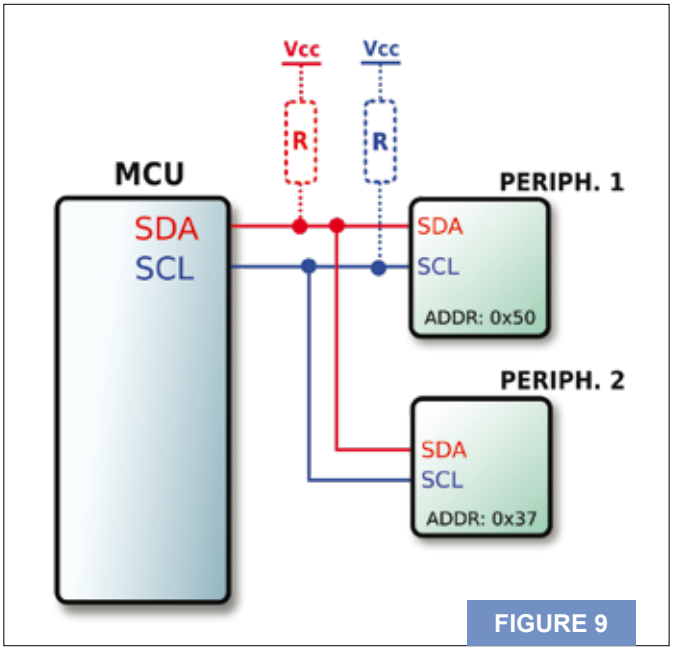

FIGURE 9

Connexion d'un MCU à deux périphériques I²C.

Une seule ligne de données étant utilisée, l'émission et la réception de données se font de manière alternée. Le chronogramme de la figure 10 montre les différentes étapes de la communication. Le contrôleur initie la communication (1) au travers d'une séquence spéciale sur les lignes SDA et SCL puis indique l'adresse du périphérique (2) et le type d'opération souhaitée (3). Ce dernier acquitte bonne réception (4), et le contrôleur envoie sa donnée (5).

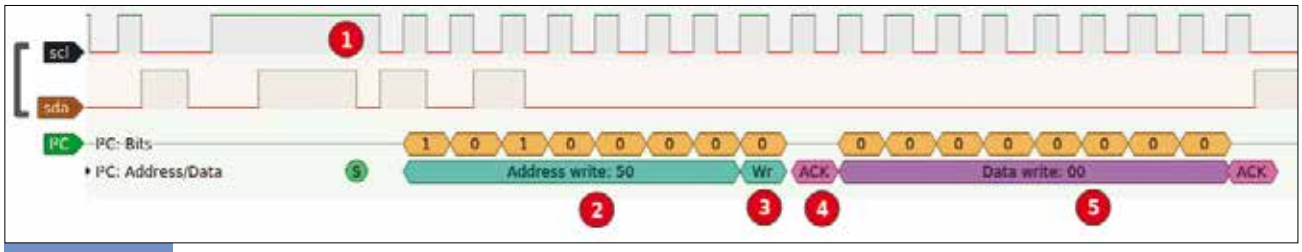

FIGURE 10 *Décodage d'une communication I²C sous Pulseview.*

Décodage des communications

Les interfaces USART, I²C et SPI établissent l'échange des données entre différents composants, néanmoins le format des données transmises varie selon les spécifications de ces derniers. Ceux-ci peuvent en effet implémenter leur propre format et convention de communication, encapsulés dans les protocoles précédemment détaillés. Seule la lecture de la spécification technique des composants en question permettra d'interpréter les données échangées et les actions réalisées par un composant sur un autre.

Les logiciels d'analyse logique comme Pulseview autorisent généralement l'enchaînement de décodeurs et possèdent de nombreux décodeurs spécifiques à certains composants pour en interpréter les échanges et les rendre plus lisibles. Si un composant n'est pas encore supporté, il faudra implémenter un nouveau décodeur, ce qui est possible avec Pulseview.

Extraction de mémoire

Mais finalement, à quelles puces va-t-on s'intéresser ? Les cibles de choix sont les mémoires non volatiles comme les EEPROM, les Flash (qui ne sont jamais que des EEPROM plus évoluées), et les MCU (microcontrôleurs), car... ils contiennent aussi une mémoire Flash et/ou une EEPROM. Et qui dit mémoire, dit données et code, de quoi alimenter une phase de rétro-ingénierie logicielle, mais avec ses spécificités comme nous l'expliquerons plus bas.

Les petites mémoires externes communiquent avec le MCU selon divers bus, notamment en SPI, et il est possible, une fois ces interfaces et leurs caractéristiques identifiées, d'extraire le contenu de ces mémoires, par exemple grâce à un Raspberry Pi et à *flashrom* **[19]**.

Nous avons mentionné au début l'usage de programmateurs universels. Selon les mémoires et les microcontrôleurs et la disposition du PCB, la lecture du composant peut parfois se faire *in-situ*, sans dessouder la puce, à l'aide de points de test ou d'un port dédié sur le PCB ou de pinces se mettant par-dessus le composant afin d'accéder à son interface de programmation et de debug (JTAG, SWD, ISP, PDI, UPDI, etc.).

Certains MCU auront des protections activées pour empêcher l'extraction de mémoire, et certaines protections sont contournables par diverses attaques matérielles, donc renseignez-vous avant de jeter l'éponge.

3. RÉTRO-INGÉNIERIE DE MICRO-LOGICIELS

Si vous avez affaire à des microcontrôleurs et au contenu de leurs mémoires Flash, oubliez toute notion de système de fichiers ou encore les outils comme *binwalk*. Votre point de départ sera la spécification technique de l'architecture ciblée, car cette dernière fournit deux informations essentielles : l'emplacement et le format de la table des vecteurs d'interruptions (*Interrupt Vector Table*), mais aussi le mapping mémoire (*Memory Mapping*). Les premiers sont des pointeurs sur des fonctions gérant divers évènements (reset, interruption matérielle, etc.), le second détaille les différents registres et adresses mémoire permettant au CPU de piloter les périphériques internes au MCU et les interfaces (SPI ou USART par exemple).

Côté désassembleurs, *IDA Pro*, *Ghidra* ou *Cutter* supportent bon nombre d'architectures que l'on trouve dans ces MCU : Intel 8051, ARM, MIPS, etc. Ghidra et Cutter sont open source, ont une bonne communauté et possèdent un décompilateur vraiment efficace qui facilite grandement l'analyse, ce qui en fait des outils de choix et gratuits de surcroît.

4. EXPLOITATION

Outre la rétro-ingénierie matérielle, il existe également diverses techniques d'exploitation spécifiquement matérielles, utiles si le but est de réaliser un audit prenant en compte ce type de menaces ou si l'exploit facilite le reverse ou la rétrocompatibilité, par exemple par le contournement de protections.

Citons les attaques de type TOCTOU **[20]** en remplaçant à chaud un périphérique mémoire SPI par un autre, les *pin2pwn* **[21]** qui, en court-circuitant certaines lignes avec une simple aiguille, génèrent des erreurs mémoire afin de basculer le *bootloader* dans un mode de débogage, ou encore le détournement d'une partie du produit, par exemple la prise de contrôle en SPI d'un *transceiver* radio pour capturer ou envoyer des trames sans devoir utiliser un équipement SDR.

CONCLUSION

Nous avons donné un aperçu des différentes étapes et techniques de rétro-ingénierie matérielle. Comme vous le voyez, la plupart sont relativement abordables et suffisantes pour analyser par exemple des appareils IoT. Si le sujet a éveillé votre curiosité, nous vous recommandons l'ouvrage de Jean-Georges Valle publié récemment : *Practical Hardware Pentesting*.

REMERCIEMENTS

Nous remercions chaleureusement nos collègues ainsi qu'Axelle Apvrille pour leur relecture attentive et leurs conseils.

Photo d'illustration.

RÉFÉRENCES

[1] https://fr.ifixit.com/

[2] https://www.fcc.gov/oet/ea/fccid

[3] http://fcc.io/

[4] https://learn.sparkfun.com/tutorials/integrated-circuits/all

[5] https://en.wikipedia.org/wiki/List_of_integrated_circuit_packaging_types

[6] https://alltransistors.com/smd-search.php?search=

[7] http://www.smdmark.com/en-US

[8] https://how-to.fandom.com/wiki/How_to_identify_integrated_circuit_(chip)_manufacturers_by_their_logos

[9] https://blog.quarkslab.com/flash-dumping-part-i.html

[10] https://blog.quarkslab.com/flash-dumping-part-ii.html

[11] https://www.youtube.com/channel/UCH6ppHEvV3_WIXEwmhv9HEg

[12] https://github.com/SiliconAnalysis/bitract

[13] https://siliconpr0n.org/

[14] https://www.patreon.com/furrtek

[15] http://www.jtagulator.com/

[16] https://github.com/cyphunk/JTAGenum

[17] http://www.grandideastudio.com/wp-content/uploads/pcbre_pocorgtfo06_excerpt.pdf

[18] https://sigrok.org/

[19] https://flashrom.org/

[20] https://trmm.net/TOCTOU/

[21] https://carvesystems.com/news/pin2pwn-how-to-root-an-embedded-linux-box-with-a-sewing-needle/