Hardware i testy penetracyjne

Przewodnik po metodach ataku i obrony

Jean-Georges Valle

Helion

Tytuł oryginału: Practical Hardware Pentesting: A guide to attacking embedded systems and protecting them against the most common hardware

Tłumaczenie: Andrzej Watrak

ISBN: 978-83-283-8792-8

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/harite
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

6

ROZDZIAŁ

Podsłuchiwanie i atakowanie najpopularniejszych protokołów

Teraz, gdy wiesz już, jak programuje się układy, wykorzystaj swoją wiedzę do przeprowadzenia prawdziwego ataku na system. W tym rozdziale poznasz kilka standardowych protokołów wykorzystywanych w komunikacji pomiędzy układem a otaczającym go światem. Protokoły te definiują zazwyczaj komunikację pomiędzy układami jedynie w warstwie fizycznej i prawie nigdy nie sięgają wyższych warstw.

W tym rozdziale opisane są następujące zagadnienia:

- stosowanie, podsłuchiwanie i atakowanie protokołu I^2C,
- stosowanie, podsłuchiwanie i atakowanie protokołu SPI,
- stosowanie, podsłuchiwanie i atakowanie protokołu UART,
- stosowanie, podsłuchiwanie i atakowanie protokołu D1W.

Wymagania techniczne

W tym rozdziale poznasz najczęściej stosowane protokoły komunikacyjne, nauczysz się je podsłuchiwać, wstrzykiwać przy ich użyciu dane i atakować metodą „człowiek pośrodku". Aby wykonać opisane przykłady, musisz przygotować kilka rzeczy. Nie jest to wprawdzie absolutnie konieczne, ale niniejszy rozdział zawiera zarówno teoretyczną, jak i praktyczną wiedzę. Dlatego bardzo zachęcam Cię do wykonania opisanych ćwiczeń.

Sprzęt

Aby wykonać ćwiczenia opisane w tym rozdziale, przygotuj następujące elementy:

- płytę prototypową,
- dwie bardzo tanie, chińskie płytki bluepill (patrz *https://stm32duinoforum.com/forum/wiki_subdomain/index_title_Blue_Pill.html*),
- klucz ST-LINK do programowania płytek,
- przewody,
- analizator stanów logicznych (w tym rozdziale wykorzystany jest OpenBench).

Dodatkowo potrzebne będą następujące urządzenia peryferyjne:

- I^2C: układ PDIP 24LC I2C EEPROM.
- SPI: układ MX25L8008 w obudowie DIP.
- UART: dowolny adapter szeregowy USB. Doskonale nadaje się do tego celu niedrogi adapter oparty na układzie CP2102. Dobrym pomysłem jest nabycie kilku sztuk adapterów.
- D1W: czujnik temperatury MAX31820.

Niezbędne będzie następujące oprogramowanie dla systemu Linux:

- `arm-gcc-eabi-none`
- `stlink`
- `sigrok`
- Fritzing (do planowania rozmieszczenia komponentów na płycie prototypowej).

W tym rozdziale wykorzystanych jest kilka schematów ilustrujących rozmieszczenie komponentów na płycie prototypowej i połączenia między nimi. Ponieważ schematy te w książce są wydrukowane w odcieniach szarości, czasami trudno jest rozróżnić poszczególne przewody. Jeżeli nie są wystarczająco czytelne, zainstaluj oprogramowanie Fritzing i otwórz w nim kolorowe, załączone do książki pliki, znajdujące się w katalogu *bluepill\r06\schematy*. W katalogu *r06* są również zapisane wszystkie przykładowe kody.

Zacznijmy od protokołu I^2C.

Protokół I^2C

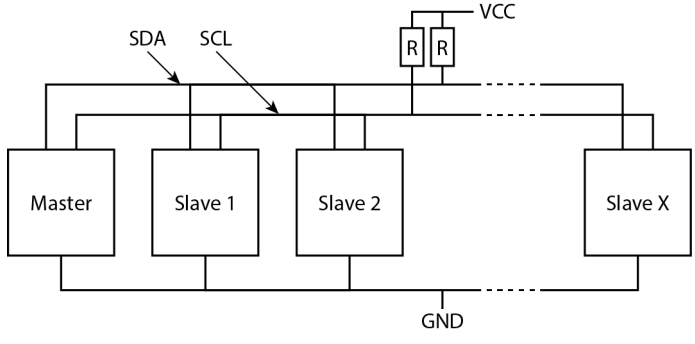
Protokół I^2C (ang. *Inter-Integrated Circuit*, pośrednik pomiędzy układami scalonymi) został opracowany w latach 80-tych ubiegłego wieku w firmie Phillips w celu wykorzystania w telewizorach. Dzięki swej prostej architekturze i niewielkiej liczbie niezbędnych komponentów jest do dziś szeroko stosowany.

Tryby pracy

Protokół I²C służy do łączenia kilku układów za pomocą dwóch linii: danych (dwukierunkowej) oraz zegara. Obie linie mają wspólną masę. Jeden z układów działa w trybie nadrzędnym (ang. *master*), a pozostałe w podrzędnym (ang. *slave*). W razie potrzeby układy mogą zmieniać tryby pracy.

Warstwa fizyczna

Protokół I²C ma tę ważną cechę, że zarówno linia danych, nazywana SDA (ang. *Serial Data*, dane szeregowe), oraz linia zegara, nazywana SCL (ang. *Serial Clock*, zegar szeregowy), są podniesione. Obie linie są poprzez rezystor dołączone do linii zasilania (VCC lub VDD), dzięki czemu ich stan jest wysoki, jeżeli układ nie wymusza stanu niskiego. Układy tworzą zazwyczaj topologię szyny, ale jeżeli są stosowane niewielkie prędkości transmisji można je łączyć w topologię gwiazdy. Zarówno układ nadrzędny, jak i podrzędny, może wysyłać sygnał taktujący. Rysunek 6.1 przedstawia typową szynę.



Rysunek 6.1. Typowa architektura szyny I²C

Bardzo ważną rolę w warstwie fizycznej odgrywa prędkość transmisji. Ma ona na przykład znaczenie podczas podszywania się pod układ. W poniższej tabeli są opisane różne prędkości transmisji.

Tryb	Skrót	Częstotliwość	Min. prąd (Imin)
Standardowy	Sm	Poniżej 100 kHz	2 mA
Szybki	Fm	Od 100 kHz do 400 kHz	2 mA
Szybki+	Fm+	Od 400 kHz do 1 MHz	3 mA
Bardzo szybki	HSm	Od 1 MHz do 3,4 MHz	6 mA
Ultra szybki	UFm	Od 3,4 MHz do 5 MHz	20 mA

Prędkość transmisji zależy głównie od wartości rezystora podnoszącego z następujących powodów:

- Wszystko, co znajduje się na płytce (ścieżki, komponenty itp.), wprowadza zakłócenia, których źródłem jest otoczenie, obudowa i wiele innych czynników.

- Ścieżki posiadają pasożytniczą pojemność, rezystancję, indukcyjność i inne parametry. Razem z rezystorami podnoszącymi wpływają na natężenie prądu ładującego kondensator.

W przypadku małych prędkości transmisji wpływ powyższych zakłóceń jest niewielki, natomiast przy większych jest już znaczący i szkodliwy. Dlatego w rozdziale 1., „Przygotowanie laboratorium do testów penetracyjnych i zasady bezpieczeństwa", wspomniałem, aby do budowania układów wykorzystujących duże prędkości transmisji nie używać płyt prototypowych.

> Zgodnie z niepisaną zasadą: im mniejsza rezystancja, tym większa prędkość transmisji. Mała rezystancja skutkuje dużym natężeniem prądu, a co za tym idzie, szybszą reakcją układu. Znaczenie ma również pojemność obwodu, ale nie mamy na nią wpływu.

Do wyliczenia wartości rezystora użyj poniższej, uproszczonej formuły uwzględniającej pojemności ścieżek, którą trudno zmierzyć:

$$R_{pull} = \frac{VCC - \min(0{,}4; (0{,}1 \cdot VCC))}{I_{min}}$$

Poniższa tabela zawiera wartości rezystancji dla typowych poziomów napięć logicznych. Na jej podstawie wybierz rezystor o wartości najbliższej wyliczonej za pomocą powyższej formuły.

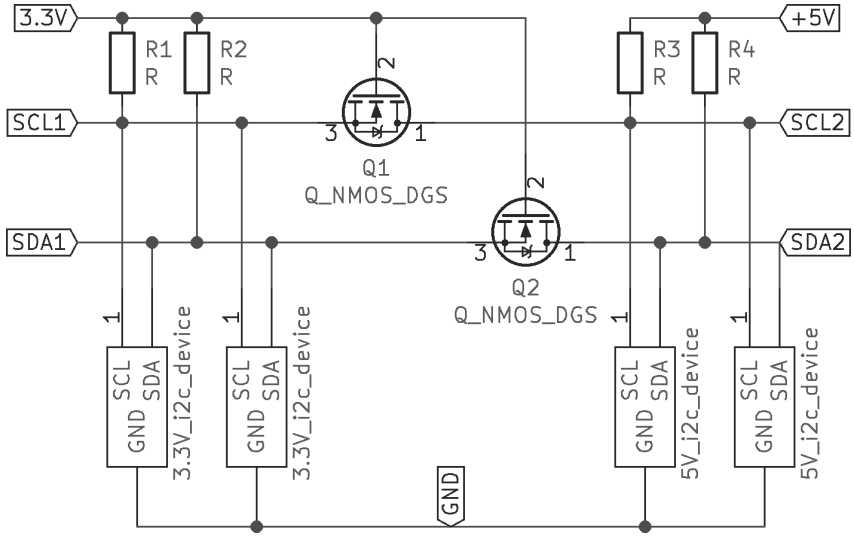
Tryb	Poziom logiczny VCC						
	5 V	3 V	2,5 V	1,8 V	1,5 V	1,2 V	1 V
Standardowy	2300	1350	1125	810	675	540	450
Szybki	2300	1350	1125	810	675	540	450
Szybki+	1533	900	750	540	450	360	300
Bardzo szybki	766	450	375	270	225	180	150
Ultraszybki	230	135	112,5	81	67,5	54	45

> Mikrokontroler może nie być przystosowany do uzyskanego natężenia prądu. Jeżeli o tym zapomnisz, możesz go spalić. Problem możesz rozwiązać, stosując tranzystory. Im mniejsza wartość rezystora podnoszącego, tym krótszy czas zmiany stanu, ale minimalne napięcie może nie być odpowiednio niskie, co potencjalnie uniemożliwi działanie całego systemu.

Poziomy logiczne i translacja napięcia

Translacja napięcia dotyczy głównie protokołu I2C, ale innych również. W protokole I^2C napięcia poziomów logicznych nie są odgórnie narzucone. Zarówno układ nadrzędny, jak i podrzędny może zmienić stan linii SDA na niski. Niezbędne jest jednak zastosowanie w tym celu translacji

napięcia. Służą do tego specjalne, ale dość drogie układy. Na szczęście inżynier z firmy Philips, Herman Schutte, wymyślił pewną sztuczkę, która pozwala osiągnąć zamierzony efekt za pomocą dwóch tranzystorów MOSFET. Ilustruje ją schemat pokazany na rysunku 6.2.

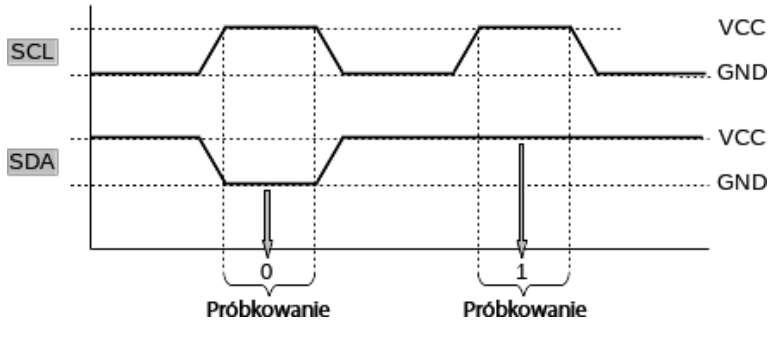


Rysunek 6.2. Dwukierunkowa translacja napięcia

Pomysłodawca proponuje, aby do translacji napięcia 5 V ↔ 3,3 V użyć tranzystorów MOSFET BSS138. Zgodnie z kartą katalogową napięcie bramki tego tranzystora jest równe 1,3 V. Jeżeli więc napięcie poziomów logicznych będzie niższe od tej wartości, trzeba znaleźć inny model tranzystora, z niższym napięciem bramki. Niektórzy producenci oferują model BSS138 z bardzo niskim napięciem. Kup kilka rodzajów tranzystorów, wybierz pracujące w niższym zakresie napięć, odpowiednim do uzyskania żądanej translacji. Na stronie Diodes Incorporated znalazłem model BSS138 z napięciem bramki równym zaledwie 700 mV.

Fizyczna forma bitów

Bity danych są kodowane jako różnice napięć na liniach SDA i SCL. Ilustruje to rysunek 6.3.

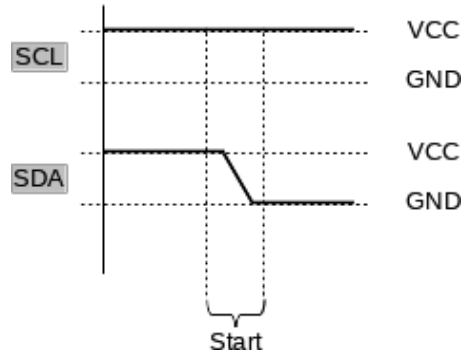


Rysunek 6.3. Próbkowanie sygnału w protokole I²C

Próbkowanie ma zazwyczaj miejsce mniej więcej w środku cyklu zegara, tj. gdy linia SCL jest w stanie wysokim, jak na rysunku 6.3, lub w chwili zmiany stanu z niskiego na wysoki. Wynikiem próbkowania jest liczba 0 lub 1 w zależności od stanu linii SDA.

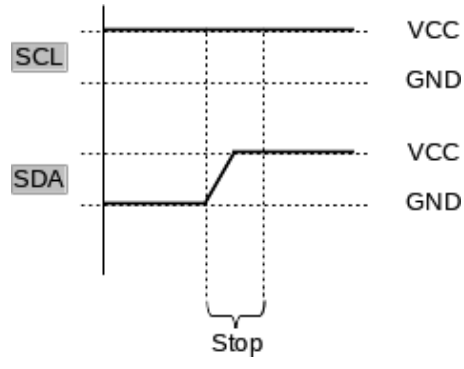
Poza zwykłym kodowaniem bitów sygnalizowanych jest kilka przypadków szczególnych:

- *Rozpoczęcie transmisji*: układ nadrzędny zmienia stan linii SDA na niski w chwili, gdy linia SCL ma stan wysoki, jak na rysunku 6.4.



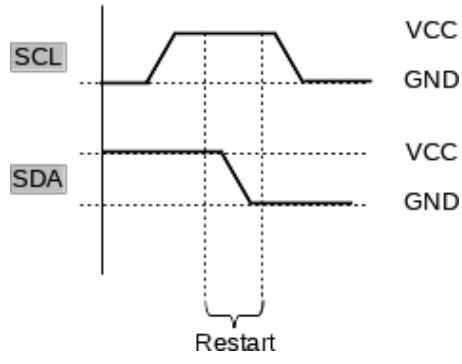
Rysunek 6.4. Rozpoczęcie transmisji w protokole I²C

- *Zakończenie transmisji*: układ nadrzędny zmienia stan linii SDA na wysoki w chwili, gdy linia SCL ma stan wysoki, jak na rysunku 6.5.

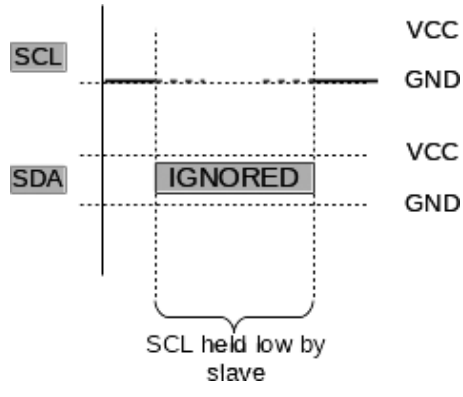


Rysunek 6.5. Zakończenie transmisji w protokole I²C

- *Restart transmisji*: przypadek podobny do rozpoczęcia transmisji. Linia SDA zmienia stan na niski w chwili, gdy linia SCL ma stan wysoki, jak na rysunku 6.6.
- *Wstrzymanie transmisji*: układ podrzędny zmienia stan linii SCL na niski, uniemożliwiając w ten sposób układowi nadrzędnemu taktowanie transmisji. Układ nadrzędny wykrywa zmianę stanu i wstrzymuje transmisję danych. Na koniec układ podrzędny zwalnia linię SCL. Ilustruje to rysunek 6.7.



Rysunek 6.6. Restart transmisji w protokole I²C



Rysunek 6.7. Wstrzymanie transmisji w protokole I²C

Poznałeś warstwę fizyczną protokołu, więc możemy teraz przejść do warstwy logicznej.

Warstwa logiczna

W protokole stosowana jest 7-bitowa adresacja urządzeń (lub rozszerzona, 10-bitowa, jednak nie wszystkie urządzenia ją obsługują). Oznacza to, że teoretycznie protokół może obsłużyć do 126 urządzeń (adres zerowy jest adresem rozgłoszeniowym i nie wszystkie układy go obsługują).

Każde urządzenie ma 7-bitowy adres. Ostatni bit bajtu oznacza żądanie odczytu lub zapisu danych zgodnie z poniższą tabelą:

Bit, odczyt	Emiter, zapis	Bit znaczący, zapis	Bit, zapis	Emiter, odczyt	Bit znaczący, odczyt
	Nadrz.	Start		Nadrz.	(Re)start
0	Nadrz.	Bit adresu 1	0	Nadrz.	Bit adresu 1
1	Nadrz.	Bit adresu 2	1	Nadrz.	Bit adresu 2
2	Nadrz.	Bit adresu 3	2	Nadrz.	Bit adresu 3

Bit, odczyt	Emiter, zapis	Bit znaczący, zapis		Bit, zapis	Emiter, odczyt	Bit znaczący, odczyt	
3	Nadrz.	Bit adresu 4		3	Nadrz.	Bit adresu 4	
4	Nadrz.	Bit adresu 5		4	Nadrz.	Bit adresu 5	
5	Nadrz.	Bit adresu 6		5	Nadrz.	Bit adresu 6	
6	Nadrz.	Bit adresu 7		6	Nadrz.	Bit adresu 7	
7	Nadrz.	Zapis: 0		7	Nadrz.	Odczyt: 1	
8	Podrz.	Potwierdzenie		8	Podrz.	Bit danych 1	
9	Podrz.	Pauza (opcja)		9	Podrz.	Bit danych 2	
10	Nadrz.	Bit polecenia 1		10	Podrz.	Bit danych 3	
11	Nadrz.	Bit polecenia 2		11	Podrz.	Bit danych 4	
12	Nadrz.	Bit polecenia 3		12	Podrz.	Bit danych 5	
13	Nadrz.	Bit polecenia 4		13	Podrz.	Bit danych 6	
14	Nadrz.	Bit polecenia 5		14	Podrz.	Bit danych 7	
15	Nadrz.	Bit polecenia 6		15	Podrz.	Bit danych 8	
16	Nadrz.	Bit polecenia 7		16	Nadrz.	Potwierdzenie	
17	Nadrz.	Bit polecenia 8			Podrz.	Pauza (opcja)	
18	Podrz.	Potwierdzenie		17	Podrz.	Bit danych 1	Powt. w razie potrzeby
	Podrz.	Pauza (opcja)	Powt. w razie potrzeby	18	Podrz.	Bit danych 2	
	Nadrz.	(Re)start		19	Podrz.	Bit danych 3	
0	Nadrz.	Bit adresu 1		20	Podrz.	Bit danych 4	
1	Nadrz.	Bit adresu 2		21	Podrz.	Bit danych 5	
2	Nadrz.	Bit adresu 3		22	Podrz.	Bit danych 6	
3	Nadrz.	Bit adresu 4		23	Podrz.	Bit danych 7	
4	Nadrz.	Bit adresu 5		24	Podrz.	Bit danych 8	
5	Nadrz.	Bit adresu 6		25	Nadrz.	Potwierdzenie	
6	Nadrz.	Bit adresu 7			Podrz.	Pauza (opcja)	

Bit, odczyt	Emiter, zapis	Bit znaczący, zapis	Bit, zapis	Emiter, odczyt	Bit znaczący, odczyt
7	Nadrz.	Zapis: 0			
8	Podrz.	Potwierdzenie			
9	Podrz.	Pauza (opcja)			
10	Nadrz.	Bit polecenia 1			
11	Nadrz.	Bit polecenia 2			
12	Nadrz.	Bit polecenia 3			
13	Nadrz.	Bit polecenia 4			
14	Nadrz.	Bit polecenia 5			
15	Nadrz.	Bit polecenia 6			
16	Nadrz.	Bit polecenia 7			
17	Nadrz.	Bit polecenia 8			
18	Podrz.	Potwierdzenie			
	Podrz.	Pauza (opcja)			

To jest pełny opis komunikacji pomiędzy układami nadrzędnym i podrzędnym z wykorzystaniem protokołu I^2C. Teraz zajmijmy się jego podsłuchiwaniem.

Podsłuchiwanie transmisji

Istnieją przynajmniej dwie metody podsłuchiwania protokołu I^2C: ogólna, tj. przy użyciu analizatora stanów logicznych, oraz z wykorzystaniem płytki Bus Pirate.

Badanym obwodem może być dowolny mikrokontroler dołączony do układu wykorzystującego protokół I^2C. W tym przykładzie jest to płytka bluepill dołączona do układu PCF8574P.

> Ponieważ prawdopodobnie będziesz pierwszy raz korzystał z urządzeń wyposażonych w interfejs USB, nie zapomnij skonfigurować w systemie Linux reguł udev, dzięki którym nie będziesz potrzebował uprawnień administratora.

Po dołączeniu urządzenia do komputera przejrzyj dziennik `dmesg` i poszukaj w nim identyfikatorów producenta (`idVendor`) i produktu (`idProduct`). Po dołączeniu analizatora stanów logicznych w dzienniku znalazłem następujące wpisy:

```
[xxx.xx] usb xxx: New USB device found, idVendor=04d8, idProduct=fc92, bcdDevice= 1.00
[xxx.xx] usb xxx: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[xxx.xx] usb xxx: Product: Logic Sniffer CDC-232
```

W katalogu */etc/udev/rules.d* utwórz plik zawierający następującą regułę:

```
SUBSYSTEMS=="usb", ATTRS{idVendor}=="identyfikator_producenta",
ATTRS{idProduct}=="identyfikator_produktu", MODE:="0666"
```

Następnie przeładuj reguły za pomocą następującego polecenia (będziesz potrzebował uprawnień administratora):

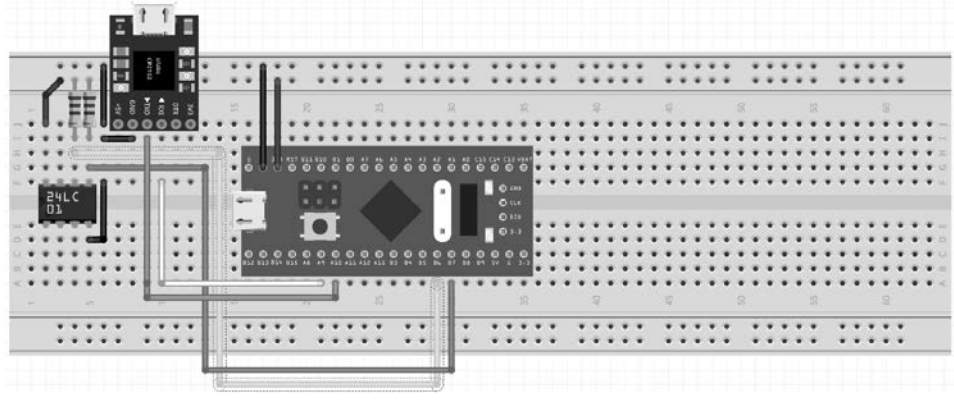
```
# udevadm control --reload-rules
```

Teraz możesz zacząć podsłuchiwać protokół.

Podsłuchiwanie przy użyciu analizatora stanów logicznych

W opisany w tym podrozdziale sposób można podsłuchiwać również inne protokoły komunikacyjne.

Użyj otwartego analizatora sprzętowego OpenBench i otwartego oprogramowania sigrok. Połącz płytkę bluepill, pamięć EEPROM, dwa rezystory podnoszące i adapter szeregowy w sposób pokazany na rysunku 6.8. Schemat znajduje się w załączonym do książki pliku *bluepill\r06\schematy\i2c.fzz*.

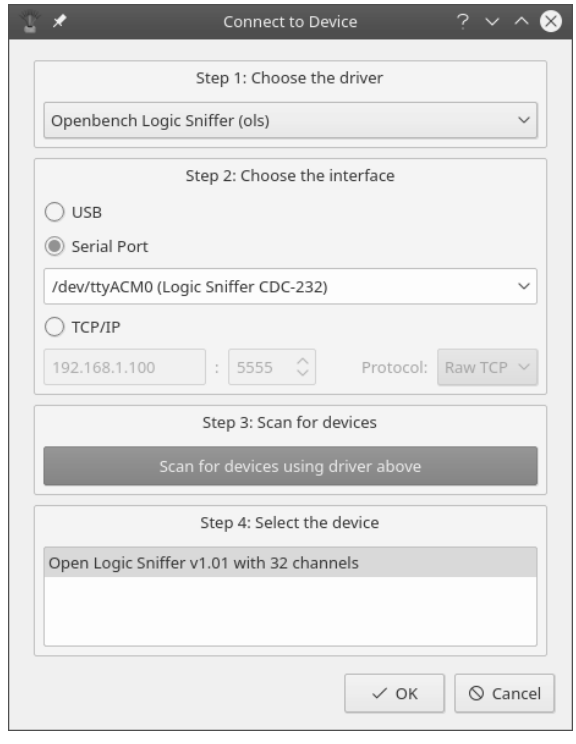


Rysunek 6.8. Środowisko do podsłuchiwania protokołu I^2C

Przygotuj środowisko w następujący sposób:

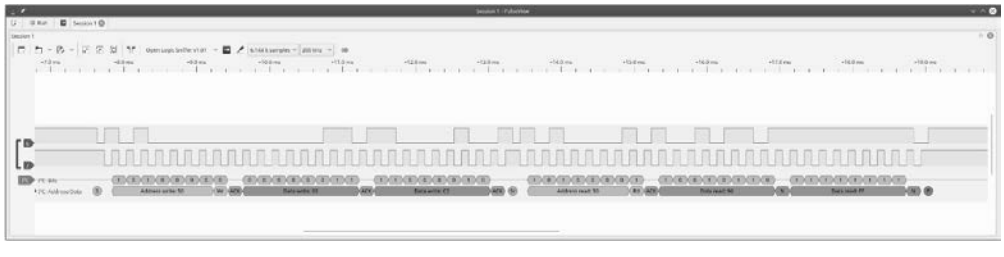
1. Połącz ze sobą masy wszystkich komponentów.
2. Połącz pin nr 0 analizatora z linią SCL.
3. Połącz pin nr 1 analizatora z linią SDA.

4. Podłącz analizator do portu USB komputera.
5. Uruchom program PulseView (graficzną nakładkę na program sigrok) i połącz się z analizatorem.
6. Kliknij polecenie *Connect to Device* (podłącz urządzenie) i wybierz opcje jak na rysunku 6.9.



Rysunek 6.9. Podłączenie analizatora stanów logicznych

7. Kliknij ikonę  , a następnie wybierz piny analizatora, których będziesz używał. Kliknij przycisk rozpoczynający podsłuchiwanie i zaczekaj, aż pojawi się obraz sygnałów jak na rysunku 6.10.



Rysunek 6.10. Obraz sygnałów w analizatorze

8. Dodaj dekoder sygnału. W tym celu kliknij żółto-zieloną ikonę w pasku narzędzi.
9. Wybierz dekoder protokołu I^2C, a następnie wskaż piny odpowiadające liniom SCL i SDA. W moim przypadku są to odpowiednio piny nr 6 i 7.

Pojawi się wykres dekodera prezentujący wartości przesyłane za pomocą protokołu I^2C. Od tej chwili możesz podsłuchiwać komunikację za pomocą analizatora stanów logicznych.

Podsłuchiwanie przy użyciu płytki Bus Pirate

W opisany w tym podrozdziale sposób można podsłuchiwać również inne protokoły.

Za pomocą płytki Bus Pirate można korzystać z protokołu I^2C na wiele sposobów. Jej mankamentem jest jednak brak oprogramowania z graficznym interfejsem użytkownika. Korzysta się z niej za pomocą wiersza poleceń opisanych w dokumentacji na stronie *http://dangerousprototypes.com/docs/Bus_Pirate_I2C* (na osobnych stronach są opisane polecenia właściwe dla innych protokołów).

Płytka jest „widziana" przez system operacyjny komputera jako urządzenie szeregowe. Za jej pomocą można podsłuchiwać powolną transmisję I^2C z taktowaniem do 100 kHz, a następnie wysyłać zarejestrowane dane. Jest to naprawdę bardzo proste. Wystarczy podłączyć piny płytki do płyty prototypowej, którą przygotowałeś wcześniej.

Wykonaj poniższe czynności:

1. Uruchom terminal szeregowy i połącz się z płytką Bus Pirate. Możesz użyć programu screen, minicom lub dowolnego innego terminala (ja używam programu screen). Nie zapomnij zdefiniować reguł udev. Spróbuj to zrobić samodzielnie. Możesz utworzyć czytelny link symboliczny, np. */dev/buspirate*. Być może będzie tego dotyczyło jedno z pytań na końcu rozdziału. Przykładowe polecenie wygląda następująco:

```
$ screen /dev/ttyUSB0 115200
```

2. Przełącz terminal w tryb I^2C (opcja 4), a następnie uruchom makro `snif` (opcja 2).
3. Włącz zasilanie układu. W terminalu powinien pojawić się wynik podobny do poniższego:

```
[0xa0+0x00+0x01+[0xa1+0xXX-0xff]
```

Fragment `0xXX` oznacza zawartość pamięci EEPROM. Przyjmuje losową wartość, jeżeli w pamięci nie były wcześniej zapisane żadne dane.

Na podstawie informacji zawartych w karcie katalogowej pamięci EEPROM można zinterpretować poszczególne fragmenty wyniku:

- `[`: rozpoczęcie transmisji.
- `0xa0`: adres urządzenia. W poprzednim ćwiczeniu analizator pokazał adres `0x50`. Jak pamiętasz, adres składa się z siedmiu bitów, a ósmy oznacza odczyt lub zapis danych. Wynikiem operacji `0x50 << 1` jest liczba `0xa0` zawierająca adres `0x50` urządzenia peryferyjnego.

Operator << oznacza przesunięcie bitów w lewo. Warto znać tę operację. Operatory logiczne i bitowe w języku C są bardzo ważne i powszechnie stosowane w programach dla systemów wbudowanych. Opis operatorów znajdziesz w podręczniku programowania w języku C na stronie *https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html#Bitwise-Logical-Operators*.

Warto również znać sposoby zapisywania w pamięci danych różnych typów, m.in. liczb całkowitych (długich i krótkich) oraz zmiennoprzecinkowych, jak również związane z tym ograniczenia. Znajomość dokładnych szczegółów nie jest niezbędna, wystarczy wiedzieć, jak znaleźć potrzebne informacje. Czasami dzięki temu można dojść do ciekawych odkryć.

- `+`: potwierdzenie adresu przez układ pamięci EEPROM.
- `0x00`: pierwszy bit adresu.
- `+`: potwierdzenie.
- `0x01`: drugi bit adresu.
- `+`: potwierdzenie.
- `[`: restart transmisji.
- `0xa1`: wynik operacji `0xa0 | 1`. Ostatni bit równy 1 oznacza odczyt danych.
- `0xXX`: odczytany bajt danych. Bez potwierdzenia, ponieważ jest to ostatni żądany bajt.

Mikrokontroler odczytuje dane bajt po bajcie, po czym następuje koniec transmisji.

Wstrzykiwanie danych

Wstrzykiwanie danych za pomocą protokołu I^2C jest dość trudne z dwóch powodów:

- Nie każdy układ obsługuje poprawnie protokół wyboru układu nadrzędnego. W takim przypadku, aby wstrzyknąć dane, trzeba użyć podstępu, tj. zrobić to w chwili, gdy układ nadrzędny nie wysyła danych lub przeprowadzić atak typu „człowiek pośrodku".
- Jak wiesz, szyna I^2C wykorzystuje otwarty kolektor i rezystory podnoszące. Oznacza to, że w celu wysłania danych trzeba zmienić stan linii na niski, tak jak to robi układ podrzędny, a następnie pozwolić jej wrócić do stanu wysokiego. Jeżeli układ nadrzędny nie współpracuje z innymi układami nadrzędnymi, transmisja jest silnie zakłócana i może doprowadzić do awarii systemu (co może być interesującym przypadkiem, choć moim zdaniem niezbyt przydatnym).

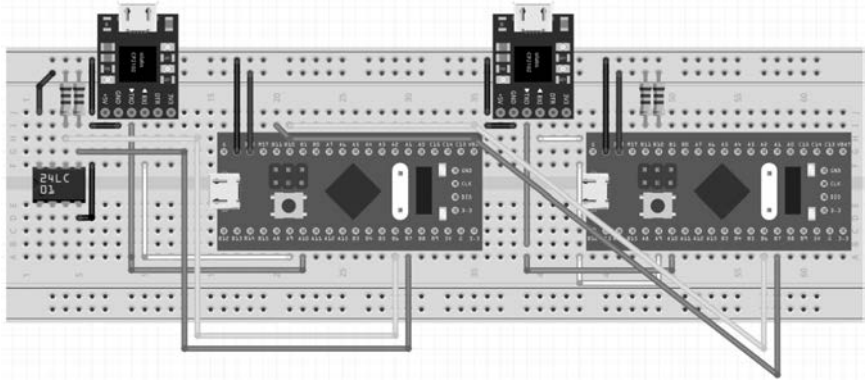
W innych sytuacjach w celu wstrzyknięcia danych wystarczy dołączyć do szyny kolejny układ nadrzędny bez użycia rezystora podnoszącego (wykorzystany będzie istniejący) i starać się uniknąć kolizji danych.

Ćwiczenie

1. Podłącz płytkę bluepill i zapisz w niej kod podsłuchujący komunikację I^2C.
2. Podłącz drugą płytkę bez osobnego rezystora podnoszącego. Kolizje nie powinny wystąpić, chyba że będziesz miał wyjątkowego pecha.
3. Co widać na obu wyjściach szeregowych?
4. Jaki stąd płynie wniosek?

Atak typu „człowiek pośrodku"

Mikrokontroler z dwoma urządzeniami I^2C, na przykład płytkami bluepill, można umieścić pomiędzy układami nadrzędnym i podrzędnym. Mikrokontroler będzie wtedy układem podrzędnym dla nadrzędnego i odwrotnie. Wystarczy jedynie znać adres układu podrzędnego. Rysunek 6.11 przedstawia układ komponentów na płycie prototypowej.

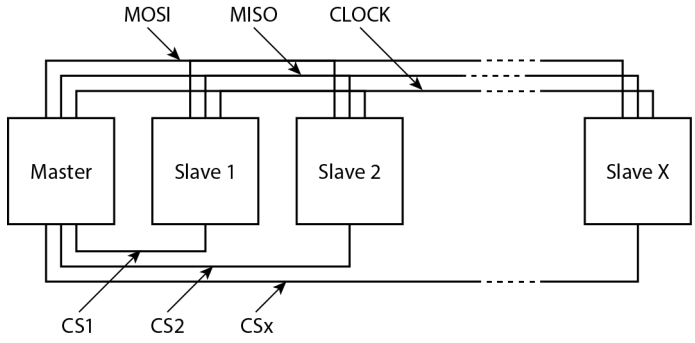


Rysunek 6.11. Środowisko do przeprowadzania ataku typu „człowiek pośrodku"

W załączonych do książki plikach znajduje się kod modyfikujący komunikację między układami, wysyłający co 100 bajtów ciąg `pentest` do układu nadrzędnego. Porównaj go z kodem odczytującym dane, spróbuj go zrozumieć i poeksperymentuj z przerwaniami.

Protokół SPI

Protokół SPI (ang. *Serial Peripheral Interface*, interfejs szeregowy do urządzeń peryferyjnych) wykorzystuje przynajmniej trzy linie: CLK (ang. *clock*, zegar), MOSI (ang. *Master Out Slave In*, wyjście układu nadrzędnego, wejście podrzędnego) i MISO (ang. *Master In Slave Out*, wejście układu nadrzędnego, wyjście podrzędnego). Jeżeli jest kilka układów podrzędnych, każdy z nich wykorzystuje dodatkową linię oznaczaną jako CS (ang. *Chip Select*, wybór układu) lub SS (ang. *Slave Select*, wybór układu podrzędnego), której stan aktywny jest zazwyczaj niski. Rysunek 6.12 przedstawia typową szynę SPI.



Rysunek 6.12. Typowa architektura szyny SPI

Protokół SPI określa jedynie sposób przesyłania bitów. W odróżnieniu od protokołu I^2C nie ma warstwy logicznej.

W systemach, w których ważna jest prędkość transmisji, stosowana jest odmiana protokołu QSPI (ang. *Queued SPI/Quad SPI*, kolejkowany / poczwórny SPI) wykorzystująca cztery linie danych. Niektóre układy obsługują obie odmiany protokołu, które można wybierać za pomocą własnych, niestandardowych poleceń przesyłanych jako dane.

Po zapoznaniu się z połączeniami między układami dowiedz się, jak działa protokół SPI.

Tryby pracy

Zacznijmy od rzeczy najważniejszych. Protokół SPI nie ma określonej częstotliwości taktowania. Narzuca ją układ nadrzędny wysyłający sygnały na linię zegara. Częstotliwość ta nie może być większa od maksymalnej, obsługiwanej przez aktualnie wybrany układ podrzędny (za pomocą linii CS).

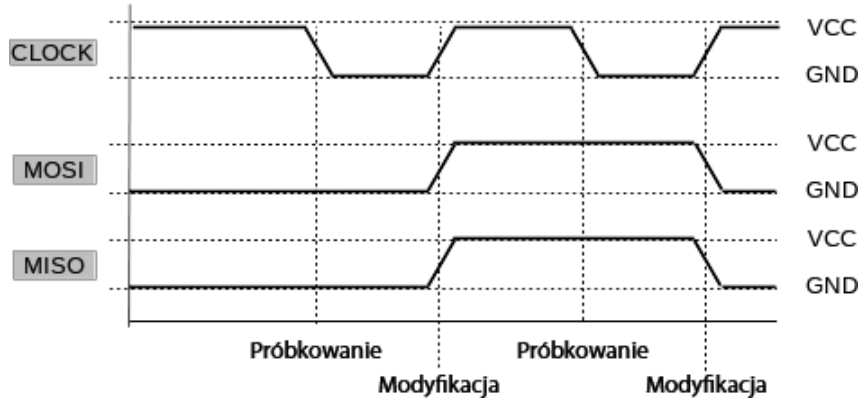
Ponadto należy pamiętać, że protokół SPI ma dwa parametry (których nazwy pochodzą od mikrokontrolera PIC i stały się nieformalnym standardem):

- **CPOL** (ang. *Clock Polarity*, polaryzacja zegara) określający, czy linia zegara jest aktywna w stanie wysokim czy niskim,
- **CPHA** (ang. *Clock Phase*, faza zegara) określający moment próbkowania sygnału na linii danych w trakcie cyklu zegara.

W efekcie protokół SPI może działać w czterech trybach.

Tryb 0

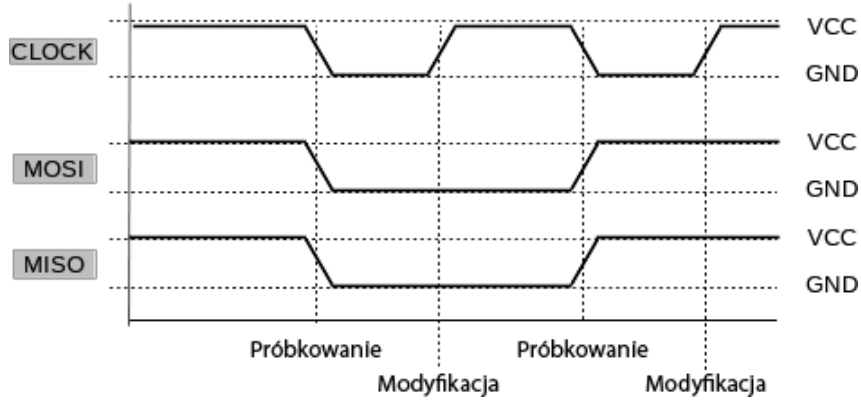
W tym trybie linia zegara jest aktywna w stanie wysokim. Dane są próbkowane przy zmianie stanu na niski i modyfikowane przy zmianie stanu na wysoki. Ilustruje to rysunek 6.13.



Rysunek 6.13. Działanie protokołu SPI w trybie 0

Tryb 1

W tym trybie linia zegara jest aktywna w stanie wysokim. Dane są próbkowane przy zmianie stanu na wysoki i modyfikowane przy zmianie stanu na niski. Ilustruje to rysunek 6.14.



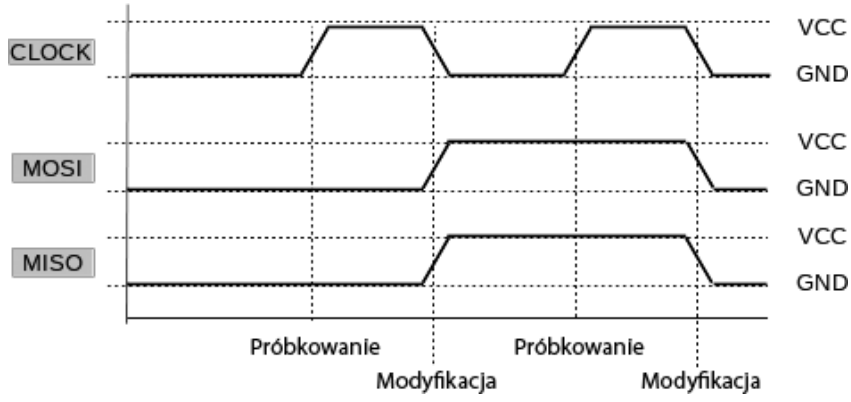
Rysunek 6.14. Działanie protokołu SPI w trybie 1

Tryb 2

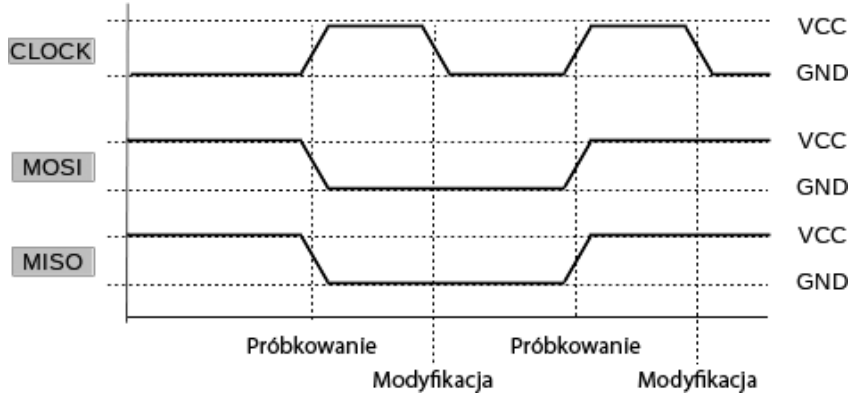
W tym trybie linia zegara jest aktywna w stanie niskim. Dane są próbkowane przy zmianie stanu na wysoki i modyfikowane przy zmianie stanu na niski. Ilustruje to rysunek 6.15.

Tryb 3

W tym trybie linia zegara jest aktywna w stanie niskim. Dane są próbkowane przy zmianie stanu na niski i modyfikowane przy zmianie stanu na wysoki. Ilustruje to rysunek 6.16.



Rysunek 6.15. Działanie protokołu SPI w trybie 2



Rysunek 6.16. Działanie protokołu SPI w trybie 3

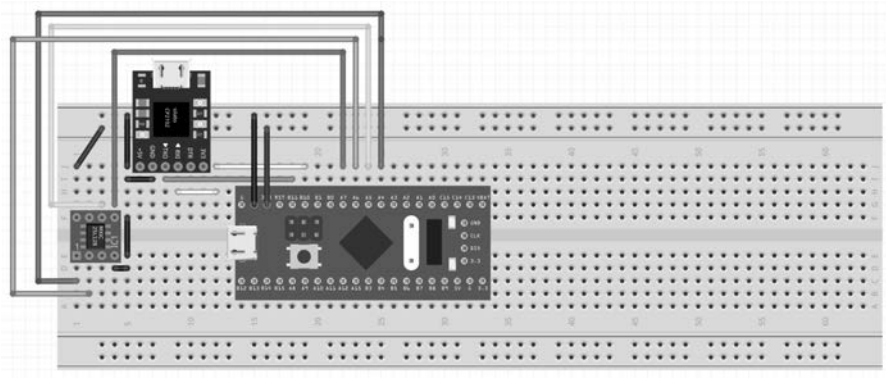
Po zapoznaniu się z działaniem protokołu SPI zajmijmy się podsłuchiwaniem transmisji.

Podsłuchiwanie transmisji

Protokół SPI można podsłuchiwać w podobny sposób jak I^2C. Należy zbudować układ wykorzystujący ten protokół, podłączyć analizator stanów logicznych do odpowiednich pinów i uruchomić program PulseView. Szczegółowe informacje zostały zawarte w podrozdziale „Protokół I2C" w części poświęconej podsłuchiwaniu transmisji.

Zbuduj środowisko pokazane na rysunku 6.17.

Przygotowanie środowiska wygląda niemal tak samo, jak w przypadku protokołu I^2C: połącz ze sobą masy wszystkich komponentów, pin nr 0 analizatora dołącz do linii CLK, pin nr 1 do linii MISO, pin nr 2 do linii MOSI, a pin nr 3 do linii CS. Następnie uruchom program PulseView i dodaj dekoder SPI. Kod dla płytki bluepill znajduje się w załączonym do książki pliku *bluepill\r06\spi_client\spi_client.c*.



Rysunek 6.17. Środowisko do podsłuchiwania protokołu SPI

Wykonaj te same operacje jak w przypadku protokołu I^2C.

Teraz zajmijmy się wstrzykiwaniem danych.

Wstrzykiwanie danych

Aby wstrzyknąć dane za pomocą protokołu SPI, należy dołączyć do szyny układ nadrzędny i dołączyć jego piny MOSI, MISO, CS i CLK do odpowiednich istniejących linii o tych samych nazwach. Następnie należy nasłuchiwać sygnałów na linii CLK, aby rozpoznać ich charakter i uniknąć kolizji danych.

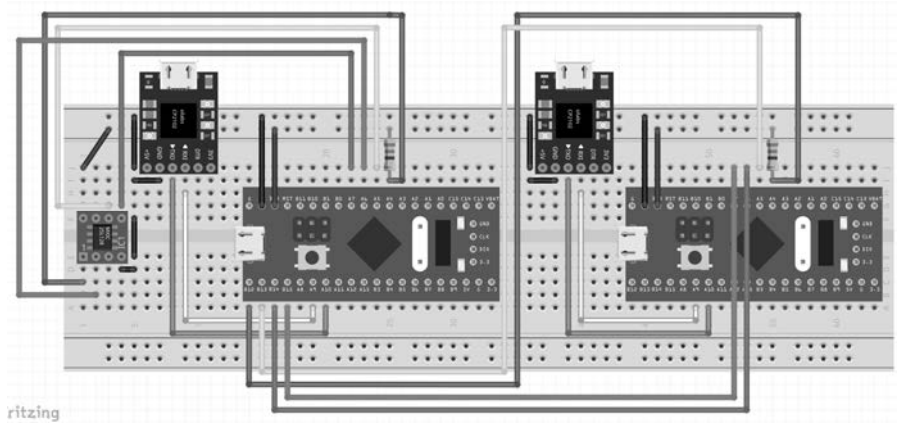
Jeżeli jest dołączonych kilka układów podrzędnych, trzeba również zarządzać linią CS.

Atak typu „człowiek pośrodku"

Umieść sprzęt pomiędzy układami nadrzędnym a podrzędnym. Skonfiguruj go tak, aby był układem podrzędnym dla nadrzędnego i odwrotnie. Jedyny, ale prosty problem polega na tym, że większość urządzeń peryferyjnych obsługiwanych przez mikrokontroler wymaga zewnętrznej linii CS/SS. Podłącz ją do masy, dzięki czemu Twój układ z perspektywy mikrokontrolera będzie zawsze aktywny (jeżeli jest to konieczne).

> Możliwość przeprowadzenia ataku typu „człowiek pośrodku" na mikrokontroler wykorzystujący protokół SPI zależy od faktycznej prędkości transmisji pomiędzy oryginalnymi układami nadrzędnym i podrzędnym. Dotyczy to w szczególności pamięci EEPROM, gdzie komunikacja nie jest transakcyjna, tj. oparta na żądaniach wysyłanych przez mikrokontroler i odpowiedziach pamięci. Niektóre pamięci EEPROM rozpoczynają wysyłanie danych, zanim mikrokontroler zakończy wysyłanie żądania. W takim przypadku należy użyć narzędzia SPI Spy (*https://github.com/osresearch/spispy*) opartego na macierzy FPGA.

Rysunek 6.18 przedstawia schemat połączeń, dostępny również w załączonym do książki pliku *bluepill\r06\schematy\spi_mitm.fzz*.



Rysunek 6.18. Środowisko do przeprowadzania ataku typu „człowiek pośrodku"

Niezbędny kod znajduje się w pliku *bluepill\r06\i2c_mitm\i2c_mitm.c*.

Teraz zajmijmy się protokołem UART.

Protokół UART

Protokół UART, zwany również RS232 lub protokołem szeregowym, opiera swoje działanie na czasie. Do przesyłania danych wykorzystuje dwie linie:

- *RX* (ang. *Receive*): do odbierania danych,
- *TX* (ang. *Transmit*): do wysyłania danych.

Dane mogą być przesyłane na dwa sposoby:

- *Ze sterowaniem sprzętowym*: wykorzystywane są dwie dodatkowe linie. Transmisją może sterować układ nadrzędny, wysyłając sygnał CTS (ang. *Clear To Send*, można wysyłać), lub podrzędny, wysyłając sygnał DTR (ang. *Data Terminal Ready*, terminal danych gotowy).
- *Bez sterowania sprzętowego*: nie jest stosowana warstwa logiczna protokołu, po prostu są przesyłane bity danych.

Poniższa tabela zawiera opisy wykorzystywanych sygnałów.

Nazwa pinu	Opis
TX	Wysyłanie danych.
RX	Odbieranie danych.
CTS/DTR	Sygnał sterujący przepływem danych: *Clear To Send* / *Data Terminal Ready*.
RTS/DSR	*Ready To Send* (gotowy do wysyłania) / *Data Set Read* (zbiór danych gotowy).

Dodatkowo na podstawie bitu parzystości dołączanego do przesyłanych danych można wykrywać błędy transmisji.

Za pomocą protokołu UART może komunikować się ze sobą wiele układów, które jednak nie mają swoich adresów. Informacje o docelowym układzie umieszcza się w przesyłanych danych. Protokół ten stanowi również podstawę dla wielu innych protokołów, m.in. IrDA, smartcard i innych.

Po zapoznaniu się z różnymi sygnałami dowiedz się, jak są wykorzystywane do przesyłania danych.

Tryby pracy

W protokole UART linia nieaktywna ma stan wysoki, a sygnał jest wysyłany poprzez zmianę stanu na niski. Efekt ten osiąga się za pomocą rezystora podnoszącego dołączonego do linii TX. Dlaczego jest to ważne? Ponieważ w końcowym efekcie rezystory mogą być dołączone do wszystkich linii. Wprawdzie zarówno mikrokontroler, jak i urządzenie peryferyjne mogą mieć własne rezystory, ale ponieważ będziemy podłączać własne układy do istniejących linii, musimy uważać, aby nie zakłócać transmisji.

Najważniejszym parametrem, który trzeba znać, jest prędkość transmisji (ang. *baud rate*), czyli liczba bitów przesłanych w ciągu sekundy. Jest to parametr o krytycznym znaczeniu, ponieważ protokół opiera swoje działanie na czasie, tj. moment transmisji pojedynczego bitu jest ściśle zdeterminowany.

Przyjętych jest wiele standardowych prędkości transmisji przedstawionych w poniższej tabeli. W każdej z nich przesyłanie symbolu zajmuje określony czas. Na podstawie tego czasu można określić właściwą prędkość transmisji, którą należy skonfigurować na urządzeniu.

Prędkość transmisji (bit/s)	Czas transmisji symbolu
110	0,009090909
300	0,003333333
600	0,001666667
1200	0,000833333
2400	0,000416667
4800	0,000208333
9600	0,000104167

Prędkość transmisji (bit/s)	Czas transmisji symbolu
14 400	6,94E-05
19 200	5,21E-05
38 400	2,60E-05
57 600	1,74E-05
115 200	8,68E-06
128 000	7,81E-06
256 000	3,91E-06

Protokół UART jest bardzo wrażliwy na dokładność zegara. Staraj się zawsze używać oscylatora kryształowego jako źródła taktowania. Wewnętrzny oscylator zbudowany z rezystora i kondensatora jest mniej dokładny i bardziej dryfuje.

Na początku transmisji jest zawsze przesyłany bit startowy sygnalizujący, że linia już nie jest bezczynna. Na koniec jest przesyłany jeden lub kilka bitów stopu. Można przesyłać dowolną liczbę bitów, ale zazwyczaj jest ich 7 lub 8. Na potrzeby wykrywania błędów można przesyłać bit parzystości.

Najczęściej stosowane parametry transmisji szeregowej są następujące:

- prędkość transmisji (liczba bitów/s),
- bit parzystości (tak/nie),
- liczba bitów zakończenia.

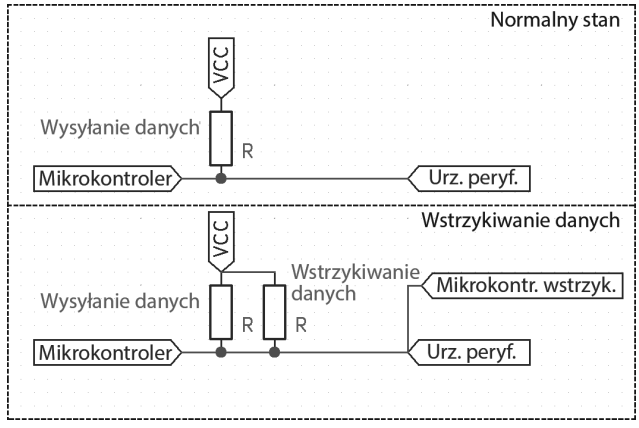
Na przykład bardzo często jest stosowna konfiguracja 9600/8-N-1, czyli 9800 bit/s, serie 8 bitów, brak bitu parzystości, jeden bit zakończenia transmisji.

Podsłuchiwanie transmisji

Podsłuchiwanie protokołu UART za pomocą analizatora stanów logicznych jest bardzo proste. Wystarczy połączyć ze sobą masy wszystkich układów, a piny nr 0 i 1 analizatora podłączyć odpowiednio do linii RX i TX. Następnie należy uruchomić program PulseView i dodać dekoder. Można również podłączyć masę adaptera szeregowego USB do masy obwodu i linii RX. Nie można w ten sposób podłączać linii TX, ponieważ zakłóciłoby to komunikację.

Wstrzykiwanie danych

Najprostszym rozwiązaniem jest podłączenie linii TX adaptera szeregowego USB do linii, w którą mają być wstrzykiwane dane. Nie zawsze jest to jednak dobry sposób, ponieważ można w ten sposób podnieść stan linii zbyt wysoko, wyżej, niż robi to właściwy nadawca. Ilustruje to rysunek 6.19.



Rysunek 6.19. Problem ze wstrzykiwaniem danych

W normalnym stanie mikrokontroler obniża napięcie na pinie do poziomu masy, zazwyczaj bardzo skutecznie, włączając bardzo mały rezystor. Napięcie jest wtedy bardzo niskie, ale niezerowe, ponieważ niewielka rezystancja zawsze pozostaje. Rezystor podnoszący i dodatkowy działają jak dzielnik napięcia. Niemniej jednak napięcie jest mniejsze od wartości granicznej, dzięki czemu urządzenie peryferyjne wykrywa niski stan linii.

Jeżeli do rezystora podnoszącego zostanie równolegle dołączony dodatkowy rezystor, wynikowa rezystancja zostanie obniżona. (W połączeniu szeregowym wynikowa rezystancja jest równa sumie rezystancji składowych, a w równoległym odwrotność wynikowej rezystancji jest równa sumie odwrotności rezystancji składowych). W efekcie mikrokontroler oryginalny lub wstrzykujący dane może nie być w stanie obniżyć poziomu napięcia na tyle, aby urządzenie peryferyjne wykryło niski stan linii. Oznaczałoby to również, że prąd przepływający przez mikrokontroler i adapter szeregowy UART byłby tak duży, że uszkodziłby układ.

W takim wypadku należy:

- Usunąć z adaptera rezystor podnoszący lub zastąpić go innym o większej wartości, o ile jest to możliwe. Istniejący rezystor w urządzeniu peryferyjnym będzie działał jak dzielnik napięcia.
- W przypadku niepowodzenia należy przeprowadzić atak typu „człowiek pośrodku".

Ćwiczenie

Dostosuj kod podsłuchujący protokół I^2C do wstrzykiwania danych za pomocą protokołu UART. Możesz się wzorować na przykładowym kodzie opisanym w następnym podrozdziale.

Atak typu „człowiek pośrodku"

Do przeprowadzenia ataku użyj dwóch adapterów szeregowych USB i prostego programu napisanego w języku Python modyfikującego przesyłane dane. Nie zapomnij przerwać oryginalnego połączenia.

Poniższy przykładowy kod odbiera bajty z urządzenia ttyUSB0, do każdego dodaje liczbę 1, a następnie wysyła do urządzenia ttyUSB1.

```
#! /usr/bin/python
# Import modułu serial.
import serial
# Otwarcie pierwszego adapter szeregowego.
serin = serial.Serial('/dev/ttyUSB0', 115200)
# Otwarcie drugiego adapter szeregowego.
serout = serial.Serial('/dev/ttyUSB1', 115200)
while(True):
    # Odczytanie bajtu z pierwszego adaptera.
    c = serin.read()
    # Dodanie liczby 1.
    c = chr(ord(c)+1)
    # Wysłanie bajtu do drugiego adaptera.
    serout.write(c)
```

W następnym podrozdziale przyjrzymy się protokołowi, w którym nie ma urządzeń peryferyjnych. Protokół D1W jest dość ciekawy, ma proste zastosowania, na przykład w systemie rejestrującym obchód strażnika. Możesz więc zrobić coś, co pozwoli pozostać strażnikowi w dyżurce w mroźną zimową noc.

Protokół D1W

Protokół D1W wykorzystuje jedną linię. Zazwyczaj jest stosowany w prostych czujnikach, np. temperatury lub wilgotności. Jest to ciekawy protokół, ponieważ linia wykorzystywana do przesyłania danych służy również do zasilania urządzenia. Takie rozwiązanie zazwyczaj nie jest stosowane w urządzeniach peryferyjnych obsługiwanych przez mikrokontroler. Dlatego trzeba samodzielnie implementować ten protokół, wykorzystując piny GPIO płytki bluepill.

Protokół D1W wykorzystuje linię z otwarty m kolektorem, podobnie jak I^2C i UART. Dlatego wymaga zastosowania rezystora podnoszącego, o wartości zazwyczaj 5 kΩ, ustawiającego odpowiedni poziom napięcia, gdy mikrokontroler odłączy pin.

Tryby pracy

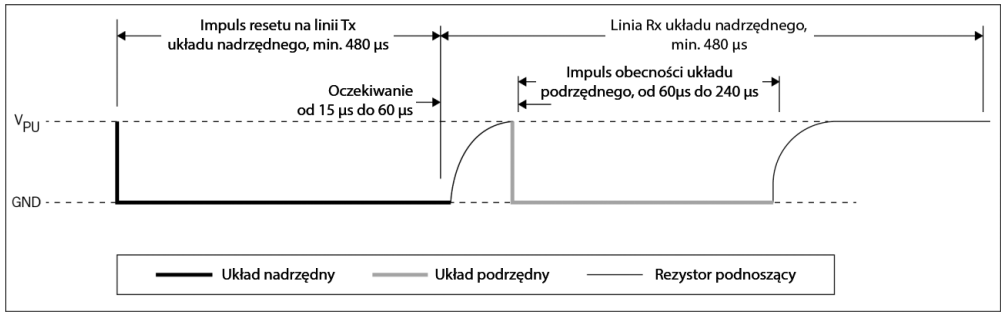
Protokół D1W opiera swoje działanie na czasie. Transmisja jest inicjowana poprzez wysłanie impulsu resetu, na które układ podrzędny odpowiada impulsem obecności.

Impuls resetu

Impuls resetu wysyła układ nadrzędny ustawiając niski stan linii na okres przynajmniej 480 μs. Zatem pierwszym zadaniem jest skonfigurowanie pinu GPIO. Do tego celu jest niezbędny odpowiednio dokładny zegar umożliwiający odmierzanie zadanego przedziału czasu. W załączonym do książki pliku *bluepill\r06\1w_client\1w_client.c* znajduje się odpowiedni *kod kliencki* (atak typu „człowiek pośrodku" opiera się na zdarzeniach, dlatego lepiej jest zacząć od programu sekwencyjnego). Kod inicjujący znajduje się w bibliotece `libLHP_D1W`.

Impuls obecności

Po wysłaniu impulsu resetu układ nadrzędny czeka, aż urządzenie peryferyjne zmieni stan linii na niski na czas od 60 μs do 240 μs. Ilustruje to rysunek 6.20.



Rysunek 6.20. Impulsy w protokole D1W

Po zapoznaniu się z impulsami przyjrzyjmy się realizowanym operacjom.

Odbieranie i wysyłanie danych

Wszystkie operacje odbierania i wysyłania danych są inicjowane przez układ nadrzędny. W celu wysłania bitu 0 układ ten ustawia niski stan linii na okres od 80 μs do 120 μs, a w celu wysłania bitu 1 na bardzo krótki czas 1 μs (wysoki stan linii przywraca rezystor podnoszący).

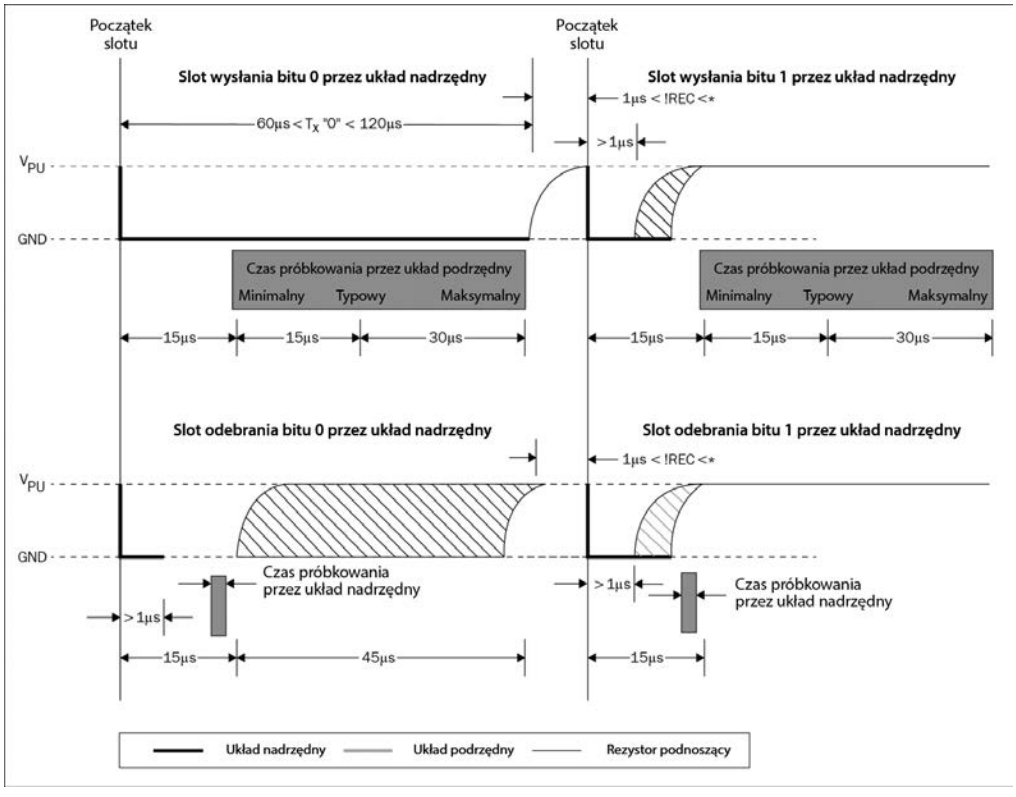
W celu odebrania danych układ nadrzędny ustawia niski stan linii, a następnie przez 15 μs czeka, aż linia zmieni stan na wysoki. Jeżeli tak się stanie, oznacza to, że został wysłany bit równy 1, a w przeciwnym wypadku bit równy 0. Szczegółowe informacje o czasach trwania impulsów są przedstawione na rysunku 6.21 i opisane na str. 17 karty katalogowej czujnika MAX31820, znajdującej się w dołączonym do książki pliku *karty katalogowe\MAX31820.pdf*.

W protokole D1W jest również stosowany kod CRC (ang. *Cyclic Redundancy Check*, cykliczny kod nadmiarowy) umożliwiający wykrywanie błędów transmisji. Informacje o jego wykorzystaniu znajdziesz w dokumentacji. Jest on ważny, ponieważ przeprowadzając atak typu „człowiek pośrodku", trzeba go odpowiednio modyfikować po zmianie danych.

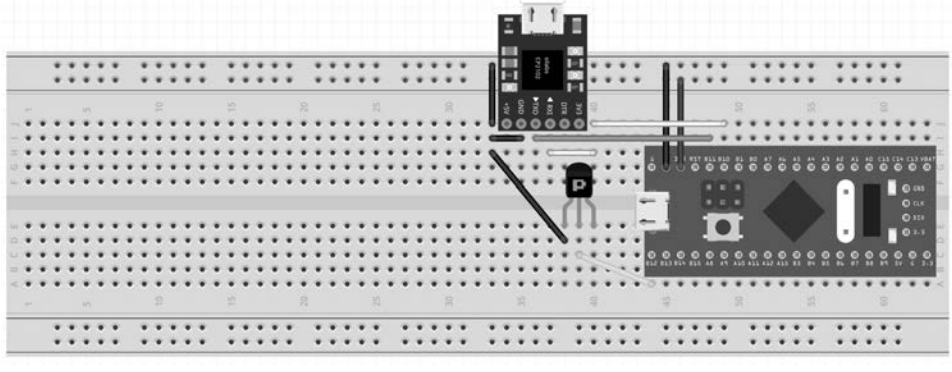
Podsłuchiwanie transmisji

Rysunek 6.22 przedstawia schemat środowiska do podsłuchiwania transmisji. Schemat jest również dostępny w dołączonym do książki pliku *bluepill\r06\schematy\d1w_mitm.fzz*.

Jak poprzednio połącz ze sobą masy wszystkich komponentów, podłącz pin nr 0 analizatora do linii danych, uruchom program PulseView i dodaj dekoder D1W.



Rysunek 6.21. Czasy trwania impulsów odczytu i zapisu danych



Rysunek 6.22. Środowisko do podsłuchiwania protokołu D1W

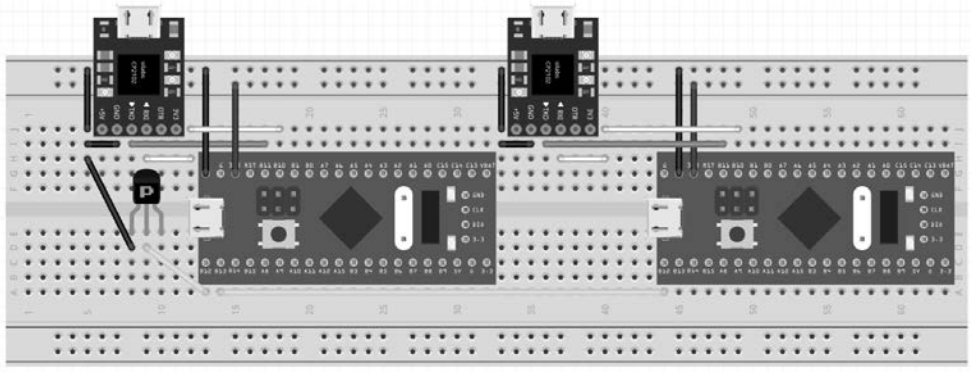
Wstrzykiwanie danych

Podłącz do istniejącej linii danych pin nowego układu nadrzędnego, który będzie wykorzystywany do wysyłania i odbierania danych.

W ogromnej większości przypadków dane są przesyłane w dużych odstępach czasu, więc prawdopodobieństwo kolizji jest bardzo małe. Jeżeli jednak pojawi się błąd CRC, zaczekaj kilka milisekund i ponów próbę.

Atak typu „człowiek pośrodku"

Rysunek 6.23 przedstawia schemat środowiska do przeprowadzania ataku typu „człowiek pośrodku".



Rysunek 6.23. Środowisko do przeprowadzania ataku typu „człowiek pośrodku"

Atak przeprowadza się w taki sam sposób, jak poprzednio: nowy układ podłącza się jako podrzędny dla istniejącego nadrzędnego i odwrotnie. W tym przykładzie modyfikowany jest odczyt czujnika temperatury i odpowiednio kod CRC.

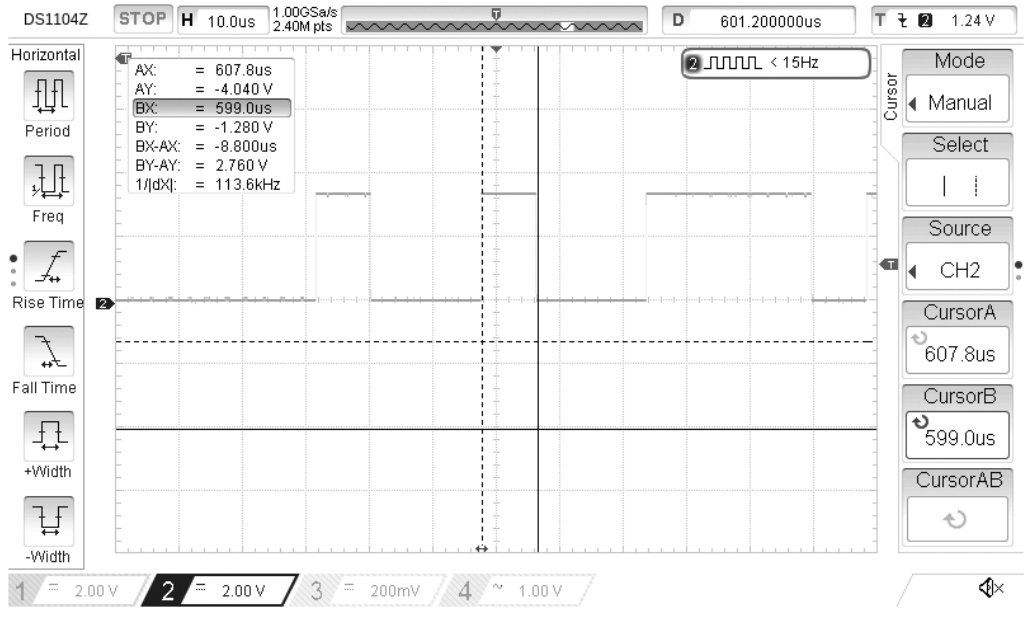
Podsumowanie

W tym rozdziale poznałeś najważniejsze protokoły komunikacyjne, dowiedziałeś się, jak je podsłuchiwać i przeprowadzać ataki typu „człowiek pośrodku". W ten sposób możesz przejmować kontrolę nad większością systemów przesyłających dane z niewielką prędkością. Poznałeś narzędzia i metody niezbędne do modyfikowania działania systemu i podsłuchiwania przesyłanych danych.

W następnym rozdziale dowiesz się, jak wyszukiwać punkty dostępu do opisanych sygnałów i w razie potrzeby tworzyć własne punkty.

Pytania

1. Wyobraź sobie, że wyświetliłeś obraz jakiegoś sygnału, jak na rysunku 6.24, i jesteś niemal pewien, że jest to protokół UART. Jaka jest prędkość transmisji?



Rysunek 6.24. Obraz sygnału na szynie UART

2. Co to jest protokół QSPI?
3. Jakie jest przeznaczenie bitu parzystości w protokole UART?
4. Gdzie został opracowany protokół I2C?
5. W jaki sposób można podłączyć kilka pamięci EEPROM 24LC do jednej szyny I^2C?
6. Wyobraź sobie, że przeprowadzasz atak typu „człowiek pośrodku" na szynę I^2C, do której jest dołączony układ nadrzędny i dwa układy podrzędne. Układ nadrzędny komunikuje się z losowo wybranym układem podrzędnym. Potrzebujesz przełączać się między adresami obu układów podrzędnych, ale płytka bluepill może mieć tylko jeden adres. Co możesz zrobić w takiej sytuacji?
7. Jaki jest wynik poniższej operacji?

```
0x43 0x6f 0x79 0x62 0x65 0x71 0x20 0x7b 0x7c 0x79 0x6a 0x33 0x7e 0x2a 0x6f 0x60
↪0x2a 0x65 0x7b 0x3a 0x65 0x65 0x64 0x65 0x32
^
0x08 0x00 0x1a 0x0a 0x04 0x1c 0x00 0x14 0x0c 0x1c 0x18 0x52 0x0a 0x45 0x1d 0x19
↪0x0a 0x07 0x12 0x54 0x04 0x17 0x0a 0x00 0x13
```

Internet rzeczy również można skutecznie zaatakować

Wraz z rozwojem internetu rzeczy, a także upowszechnianiem się elektronicznego sterowania i kontrolowania różnych procesów przestępcy doskonalą techniki łamania zabezpieczeń systemów wbudowanych. Oznacza to, że testowanie pod kątem bezpieczeństwa powinno dotyczyć sprzętu i systemów wbudowanych. Szczególną rolę w tym procesie odgrywają testy penetracyjne, których celem jest wyszukiwanie luk w zabezpieczeniach.

Oto praktyczny przewodnik po bezpieczeństwie sprzętu. Opisuje podstawy sprzętowej architektury systemów wbudowanych i protokoły komunikacyjne stosowane w urządzeniach elektronicznych. Pokazuje, w jaki sposób można przechwytywać przesyłane dane i jak wykorzystać tę wiedzę do przeprowadzania ataków. W książce zaprezentowano techniki identyfikacji i klasyfikacji zagrożeń systemu. Przeanalizowano także zależności łączące system wbudowany z jego otoczeniem, przy czym zwrócono uwagę na możliwe podatności na ataki i konsekwencje ewentualnego odczytania oprogramowania układowego. W końcowej części natomiast omówiono zasady inżynierii wstecznej oprogramowania, umożliwiającej ataki na urządzenia. Znajdziemy tu również wskazówki dotyczące ochrony urządzeń przed najbardziej typowymi atakami.

Dzięki książce dowiesz się, jak:

- testować systemy wbudowane i rozpoznawać ich najważniejsze funkcjonalności
- identyfikować i atakować krytyczne zabezpieczenia
- odczytywać i modyfikować dane zapisane w systemach wbudowanych
- badać zależności pomiędzy oprogramowaniem układowym a sprzętem
- atakować zabezpieczenia stosowane w różnych blokach funkcjonalnych urządzeń

Jean-Georges Valle jest pentesterem systemów stosowanych między innymi w sterownikach przemysłowych, miejskich urządzeniach IoT, dystrybutorach mediów i miernikach mocy. Opanował sztukę atakowania systemów wbudowanych. Zajmuje się bezpieczeństwem sprzętu i oprogramowania, a także badaniem odporności systemów wbudowanych i oceną podatności sprzętu na ataki.

ISBN 978-83-283-8792-8