From black to white (box) attacks on secure systems: or why do your light bulbs need a firmware update

Eyal Ronen

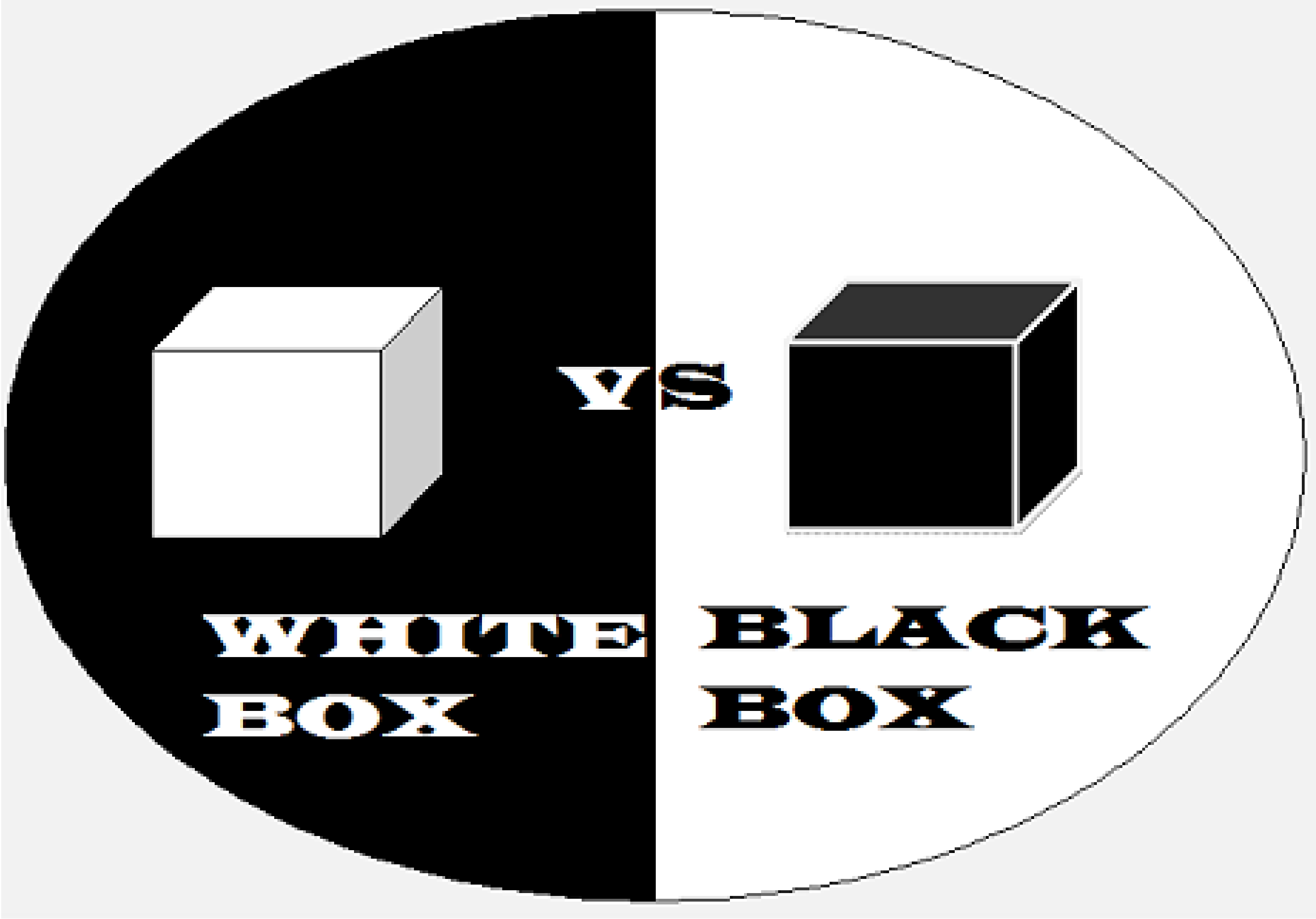
מכון ויצמן למדע

WEIZMANN INSTITUTE OF SCIENCE

What is this talk about

- Example of a hardware attack process

- Focus on what didn’t work and the hard labor
  - You can read about the other stuff in the paper


VS
WHITE BOX
BLACK BOX

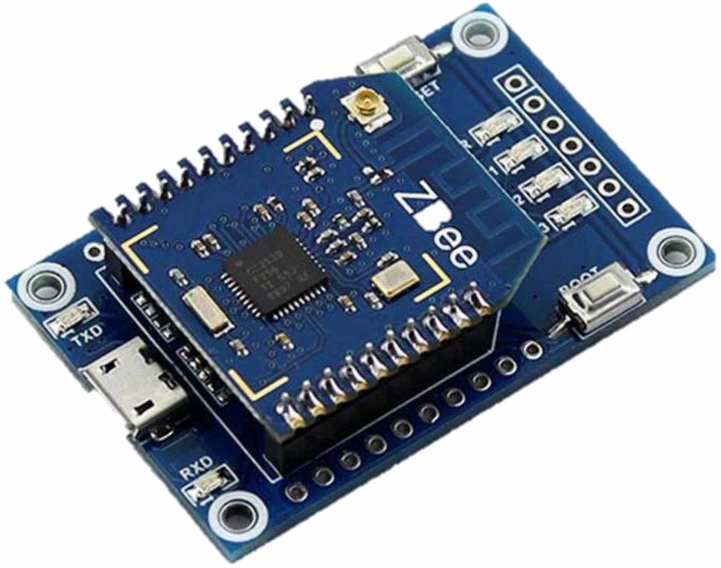
Black to White Using RE

What can we use?

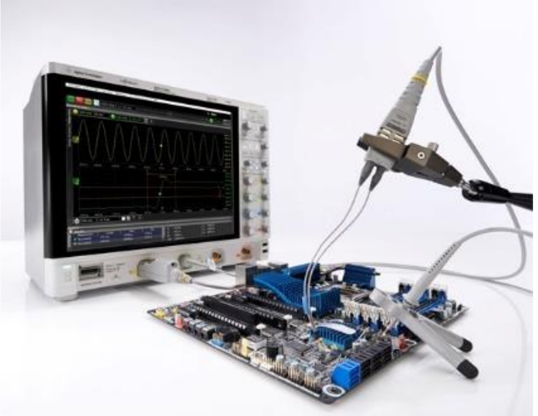
What can we use?

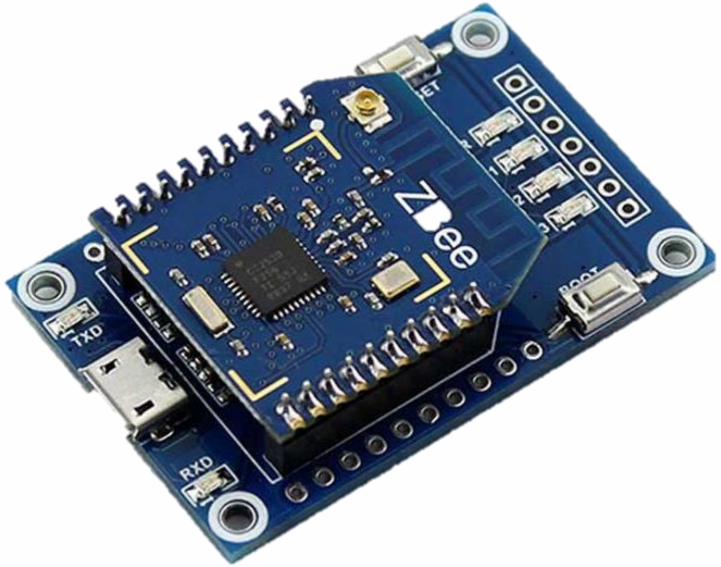
What can we use?

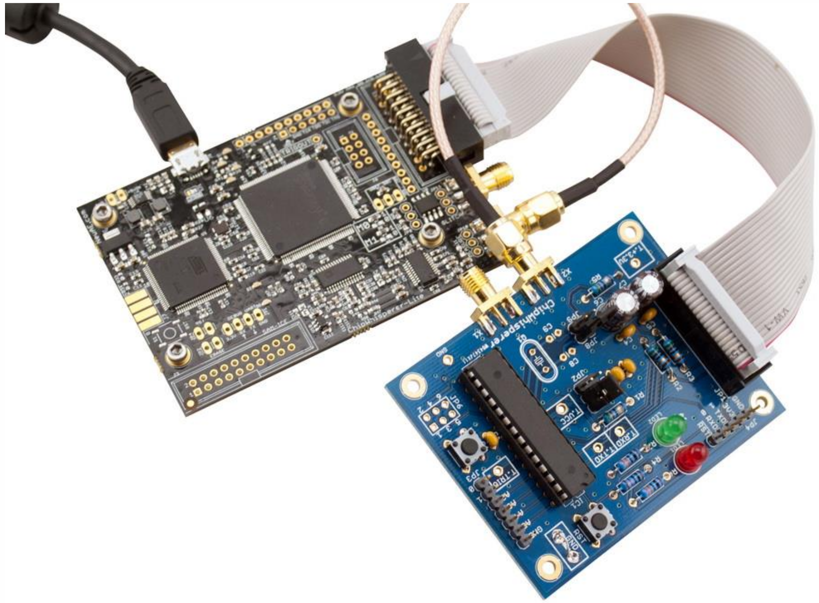
What can we use?



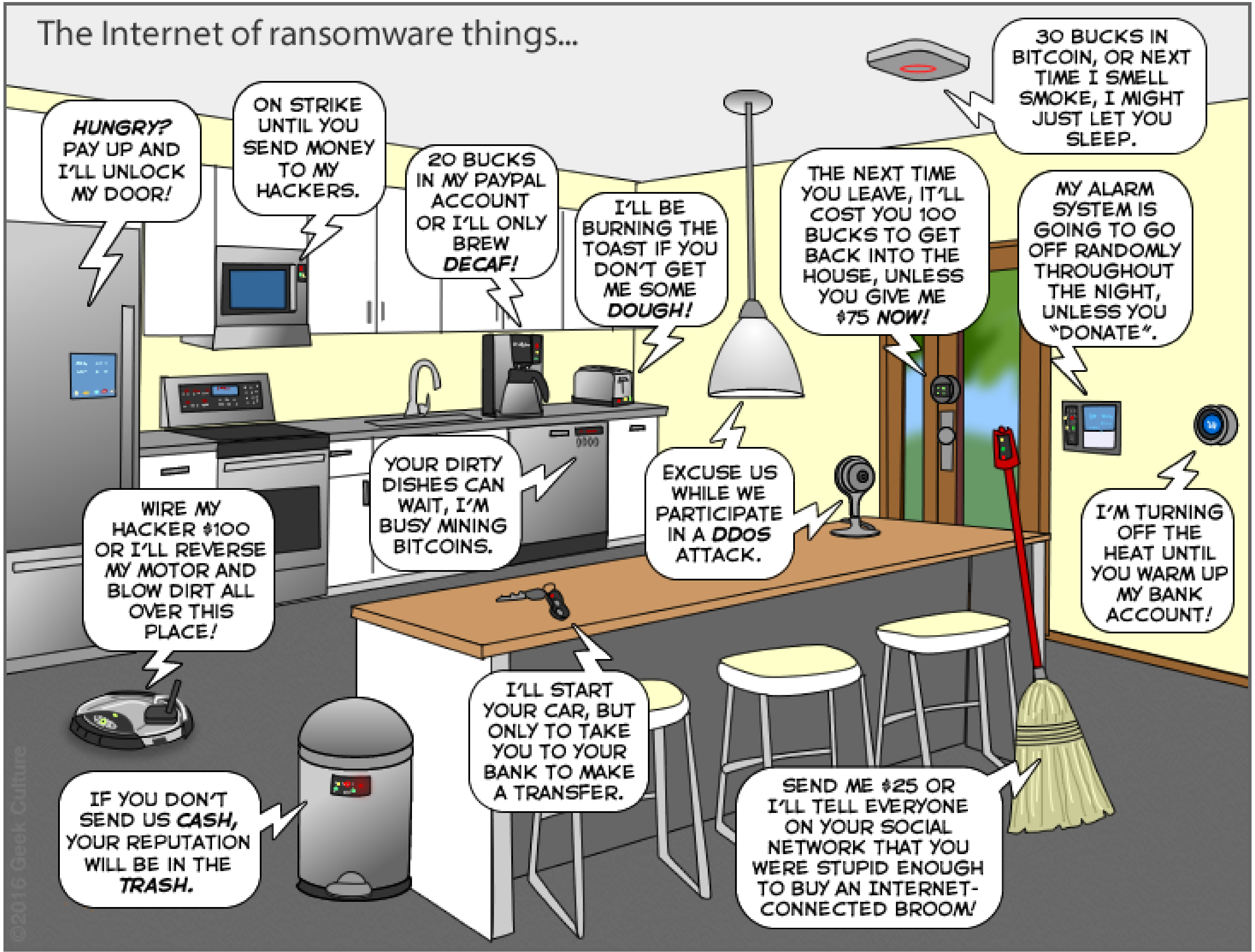
What can we use?







The Joy of Tech™ by Nitrozac & Snaggy


The Internet of ransomware things...
HUNGRY? PAY UP AND I'LL UNLOCK MY DOOR!
ON STRIKE UNTIL YOU SEND MONEY TO MY HACKERS.
20 BUCKS IN MY PAYPAL ACCOUNT OR I'LL ONLY BREW DECAF!
I'LL BE BURNING THE TOAST IF YOU DON'T GET ME SOME DOUGH!
THE NEXT TIME YOU LEAVE, IT'LL COST YOU 100 BUCKS TO GET BACK INTO THE HOUSE, UNLESS YOU GIVE ME $75 NOW!
30 BUCKS IN BITCOIN, OR NEXT TIME I SMELL SMOKE, I MIGHT JUST LET YOU SLEEP.
MY ALARM SYSTEM IS GOING TO GO OFF RANDOMLY THROUGHOUT THE NIGHT, UNLESS YOU "DONATE".
YOUR DIRTY DISHES CAN WAIT, I'M BUSY MINING BITCOINS.
EXCUSE US WHILE WE PARTICIPATE IN A DDoS ATTACK.
WIRE MY HACKER $100 OR I'LL REVERSE MY MOTOR AND BLOW DIRT ALL OVER THIS PLACE!
I'M TURNING OFF THE HEAT UNTIL YOU WARM UP MY BANK ACCOUNT!
I'LL START YOUR CAR, BUT ONLY TO TAKE YOU TO YOUR BANK TO MAKE A TRANSFER.
IF YOU DON'T SEND US CASH, YOUR REPUTATION WILL BE IN THE TRASH.
SEND ME $25 OR I'LL TELL EVERYONE ON YOUR SOCIAL NETWORK THAT YOU WERE STUPID ENOUGH TO BUY AN INTERNET-CONNECTED BROOM!

IoT Goes Nuclear: Creating a ZigBee Chain Reaction

Eyal Ronen, Colin O'Flynn,
Adi Shamir, Achi-Or Weingarten

מכון ויצמן למדע
WEIZMANN INSTITUTE OF SCIENCE

DALHOUSIE UNIVERSITY

Typical IoT devices: Philips Hue Smart Lights

Typical IoT devices: Philips Hue Smart Lights

- Mature technology and standards, a relatively simple system

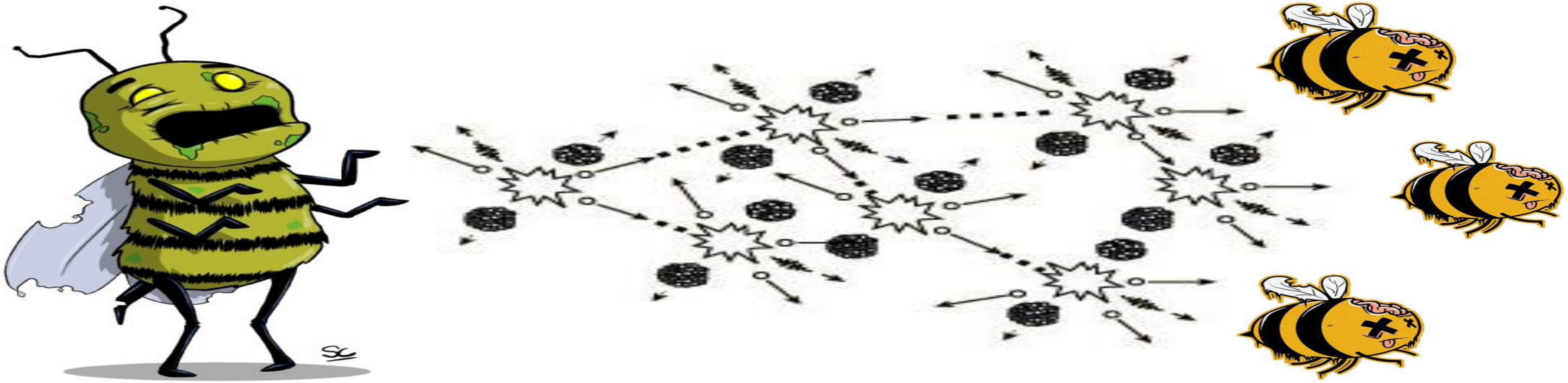
Typical IoT devices: Philips Hue Smart Lights

- Mature technology and standards, a relatively simple system

- A high end product with high end security, but…

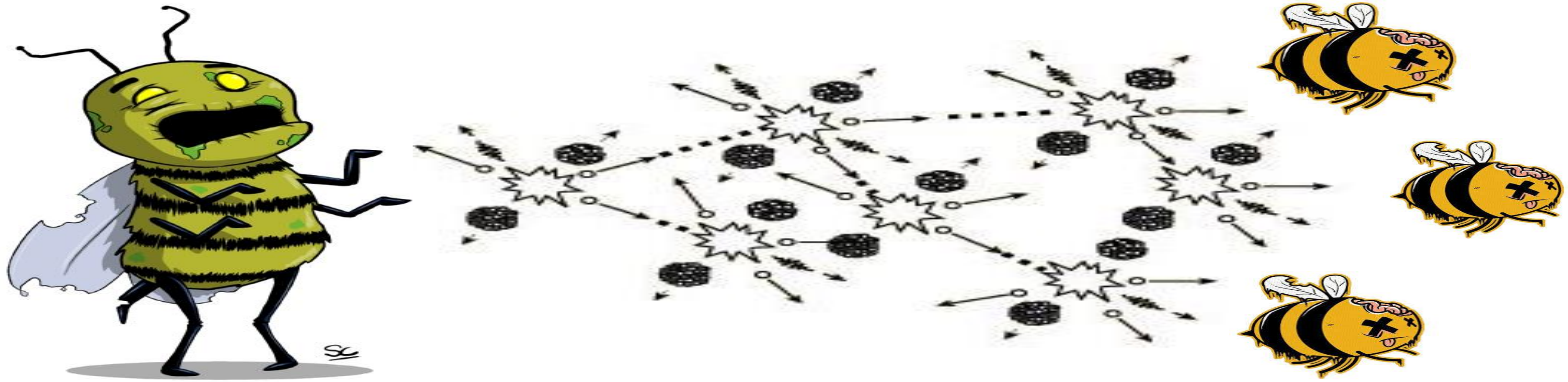
Creating a lightbulb worm

- We have proven the possibility of creating a worm which spreads using only the standard ZigBee wireless interface



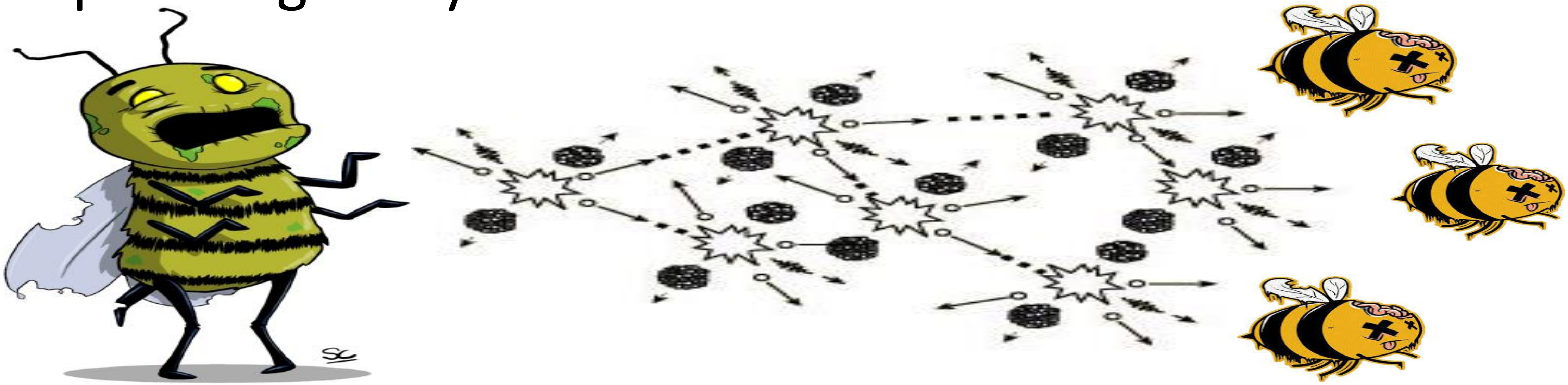
Creating a lightbulb worm

- We have proven the possibility of creating a worm which spreads using only the standard ZigBee wireless interface
  - Taking over a preinstalled smart light

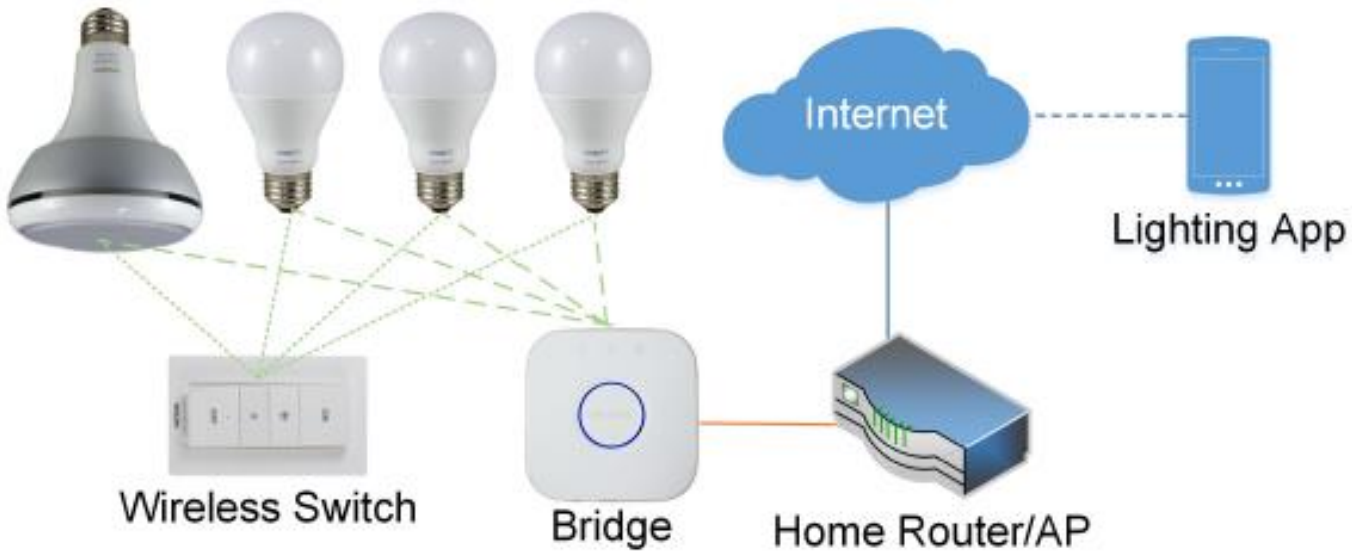


Creating a lightbulb worm

- We have proven the possibility of creating a worm which spreads using only the standard ZigBee wireless interface
  - Taking over a preinstalled smart light
  - Spreading everywhere

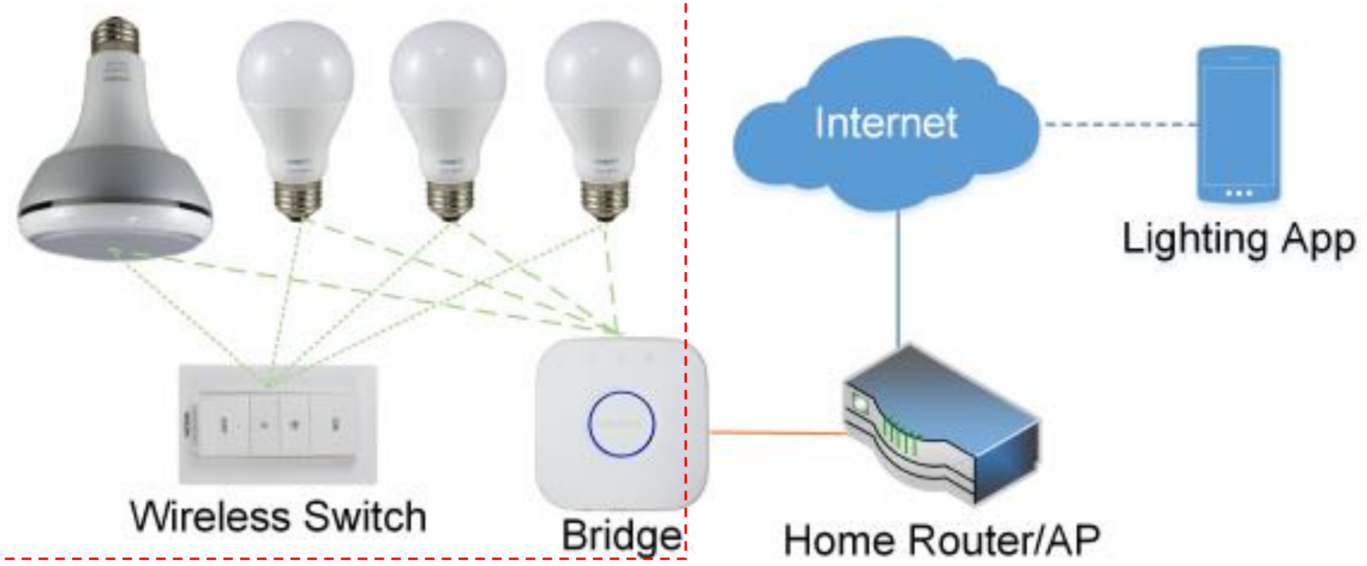


The underlying ZLL protocol



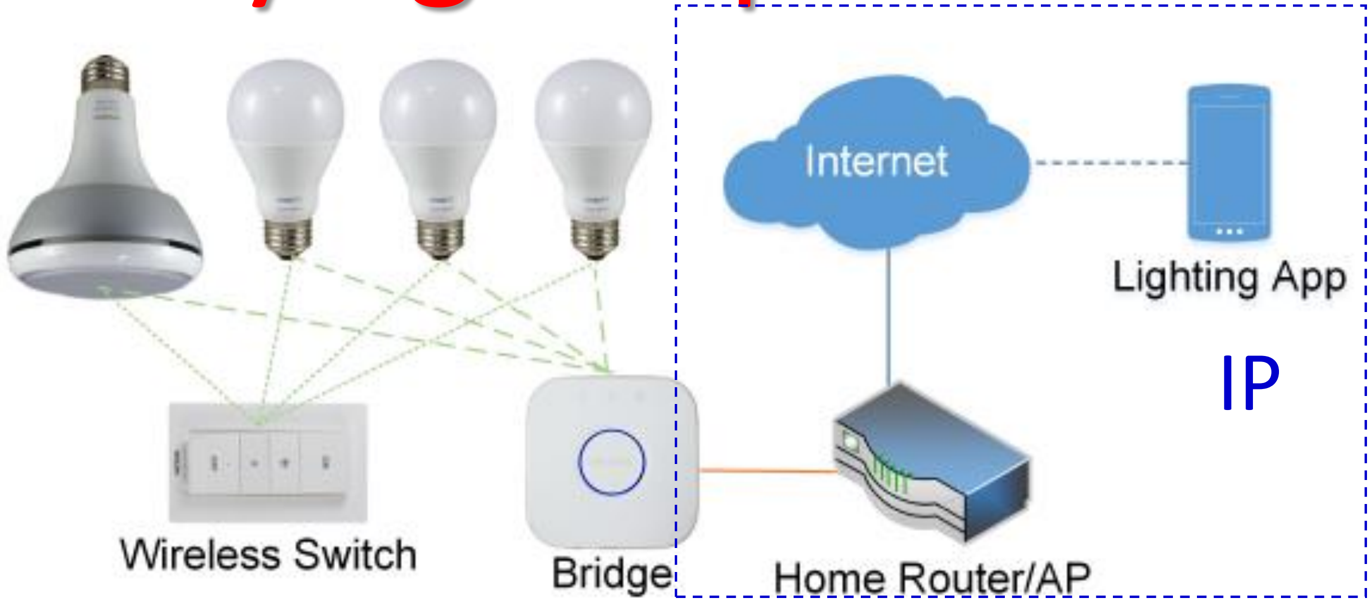
The underlying ZLL protocol

Zigbee
Personal
Area
Network



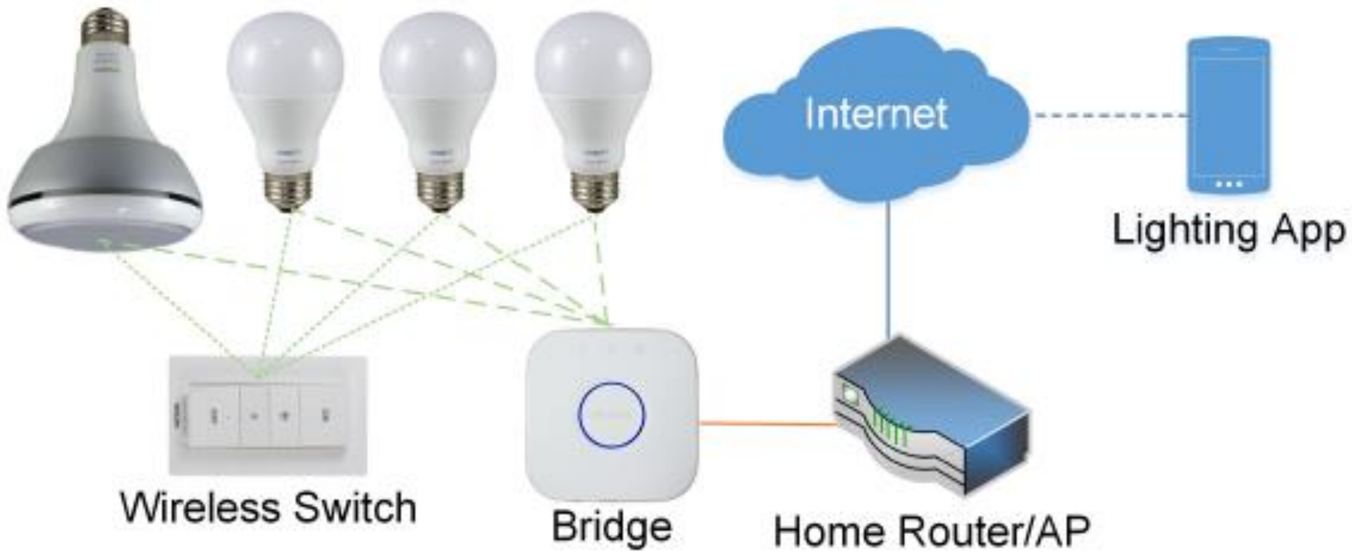
- Each installed light is connected to a central controller using the ZigBee Light Link (ZLL) wireless protocol in a Personal Area Network (PAN)

The underlying ZLL protocol



- Each installed light is connected to a central controller using the ZigBee Light Link (ZLL) wireless protocol in a Personal Area Network (PAN)
- The bridge is connected to a secure home/ office network, and is controlled by a smartphone app via IP

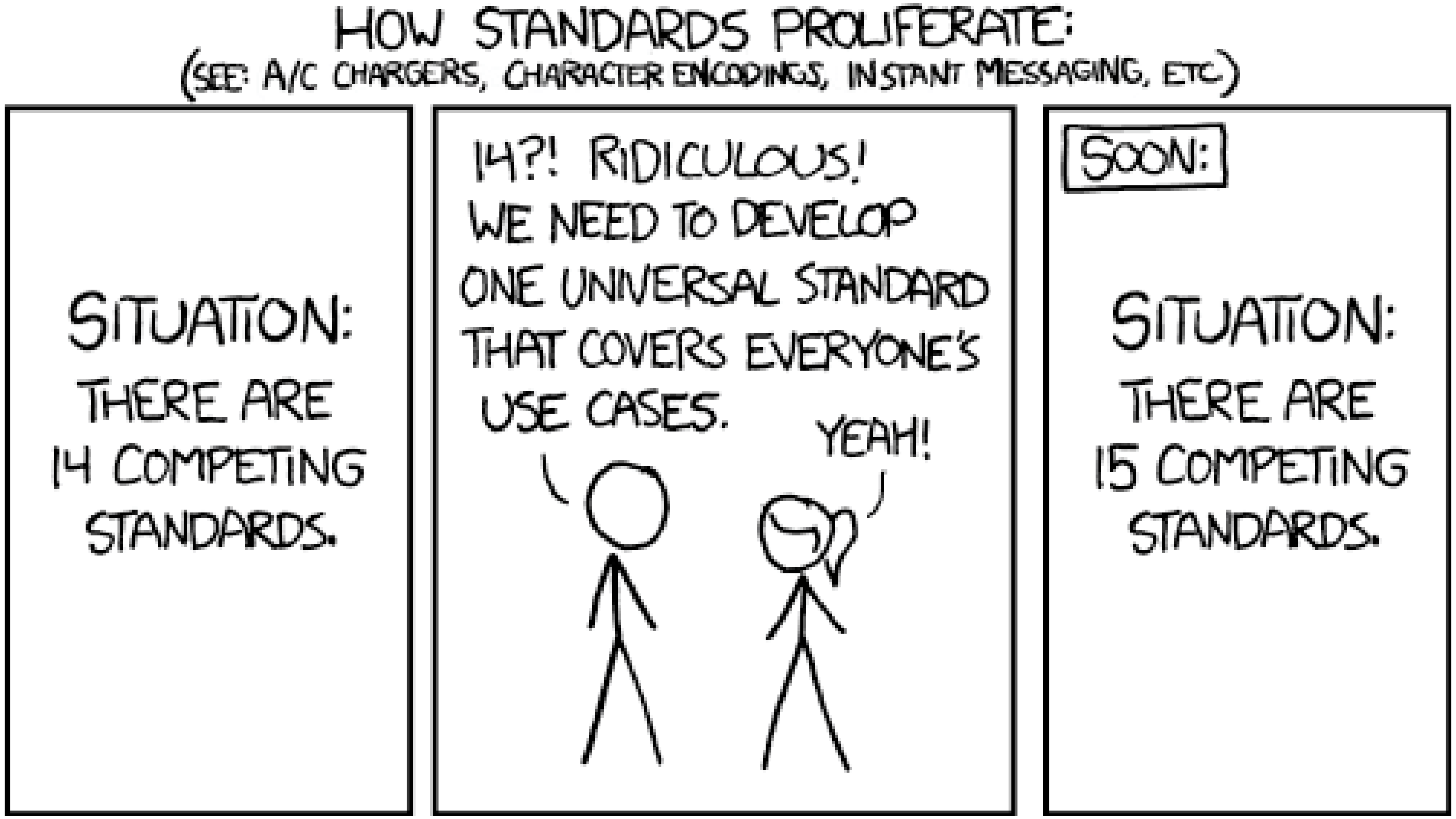
The underlying ZLL protocol



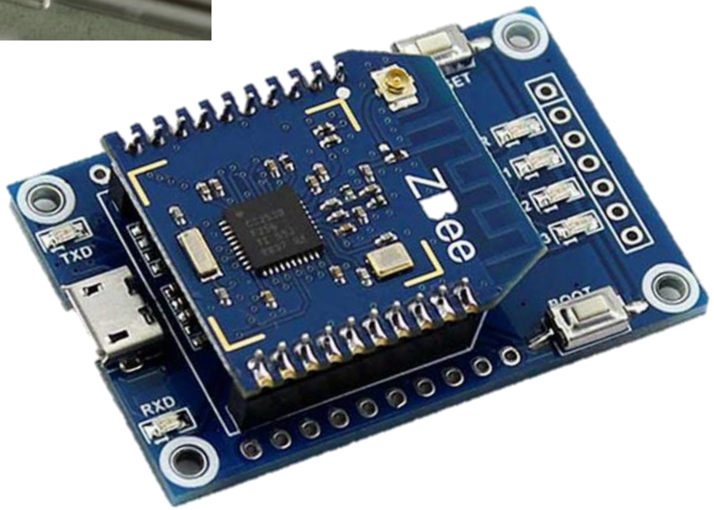
- Each installed light is connected to a central controller using the ZigBee Light Link (ZLL) wireless protocol in a Personal Area Network (PAN)
- The bridge is connected to a secure home/ office network, and is controlled by a smartphone app via IP
- It enables each authorized user to turn each light on or off, to change the light intensity, and to set its color

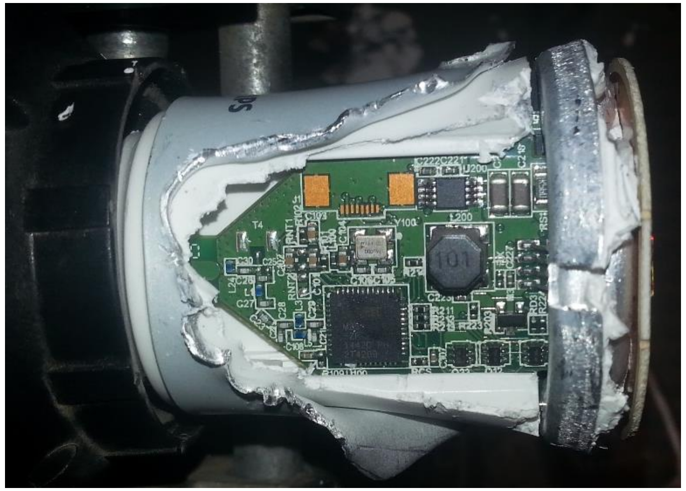
The fun world of standards

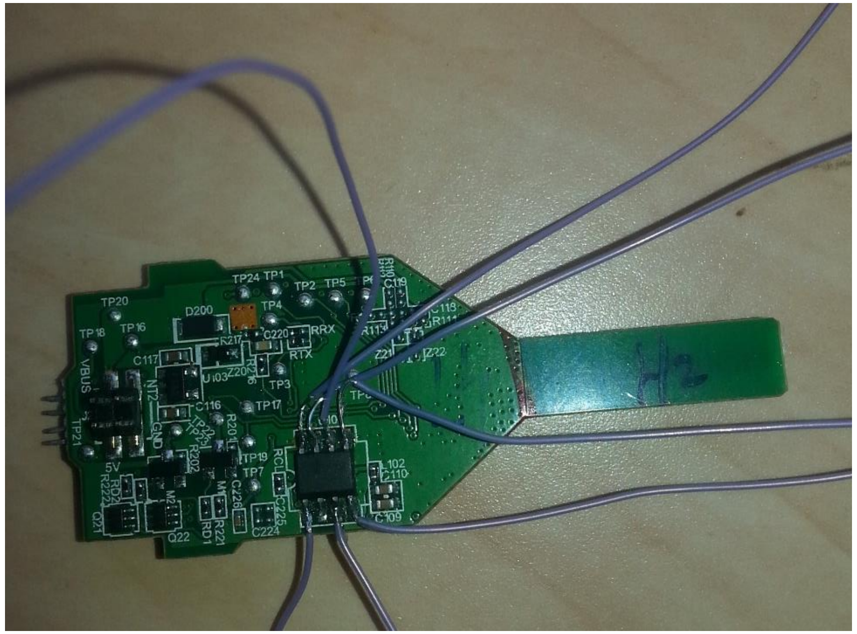
- ZigBee Pro
- ZigBee HA
- ZigBee ZLL
- ZigBee OTA Update
- ....

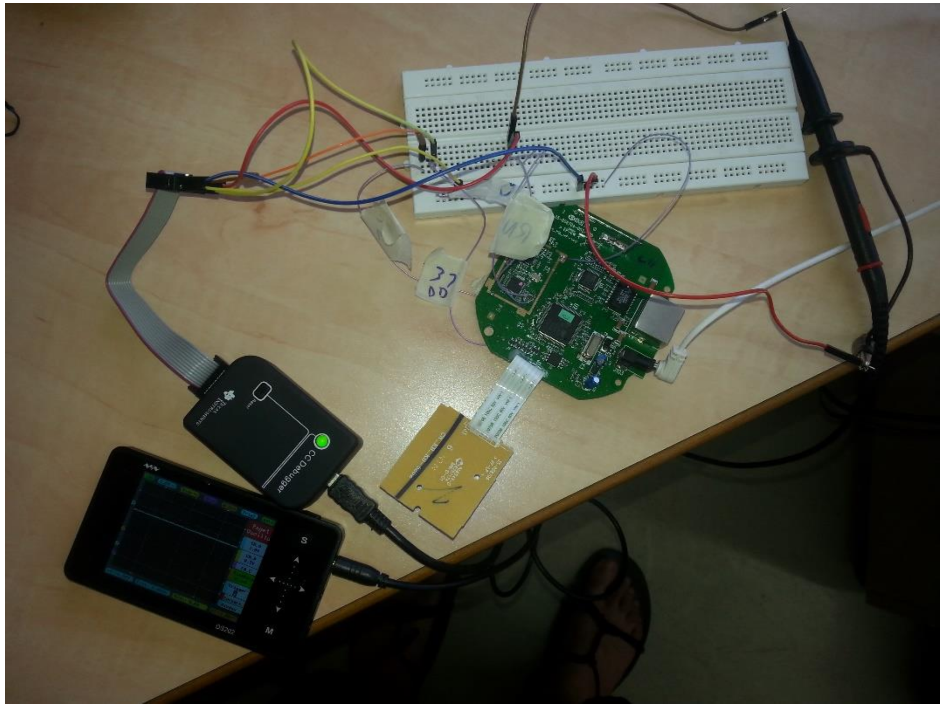

HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

```
class ScanRespPkt(PktParser):
    def _reset(self):
        PktParser.__init__(self, 'ScanRespPkt', [UINT32('Trans ID'), UINT8('RSSI correction'), UINT8('Zigbee Info'),
                                                UINT8('ZLL info'), UINT16('Key bitmask'), UINT32('Resp ID'),
                                                UINT64('Extended PAN identifier IEEE address'),
                                                UINT8('Network update identifier'),UINT8(' Logical channel'),
                                                UINT16('PAN identifier'), UINT16('Network address'),
                                                UINT8('Number of sub-devices'), UINT8('Total group identifier')])
        self._tail = PktParser('', [UINT8('Endpoint identifier'), UINT16('Profile identifier'),
                                    UINT16('Device identifier'), UINT8('Version'), UINT8('Group identifier count')])

    def __init__(self):
        self._reset()

    def unpack(self, raw):
        self._reset()
        raw = PktParser.unpack(self, raw)
        if(self['Number of sub-devices']['val'] == 1):
            raw = self._tail.unpack(raw)
            self.update(self._tail)
        ZLLInterPanState['Cur RespID'] = self['Resp ID']['val']
        return raw

    def pack(self, list):
        self._reset()
        PktParser.pack(self, list[0:13])
        if(self['Number of sub-devices']['val'] == 1):
            self._tail.pack(list[13:])
            self.update(self._tail)
        ZLLInterPanState['Cur RespID'] = self['Resp ID']['val']
```

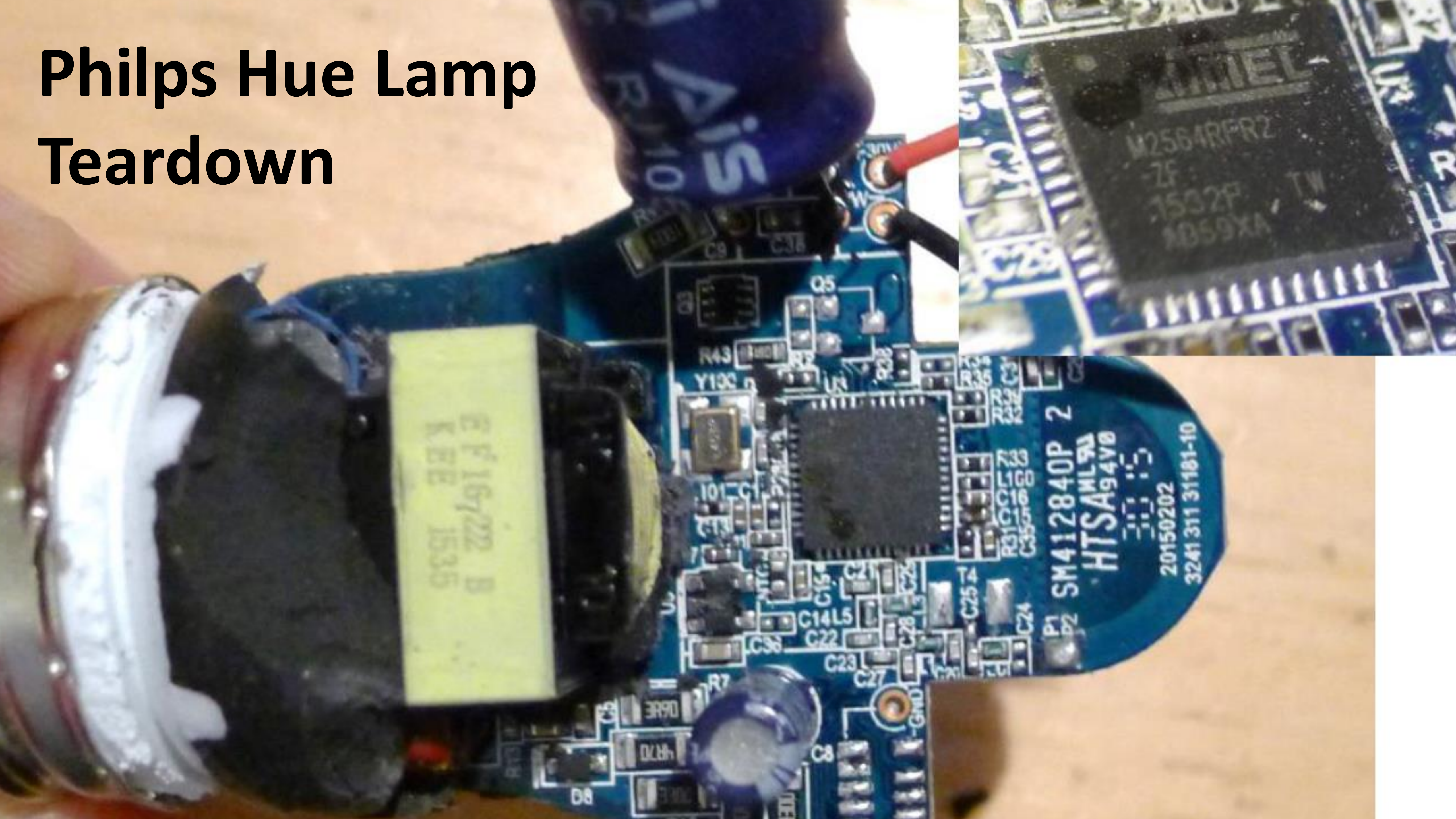


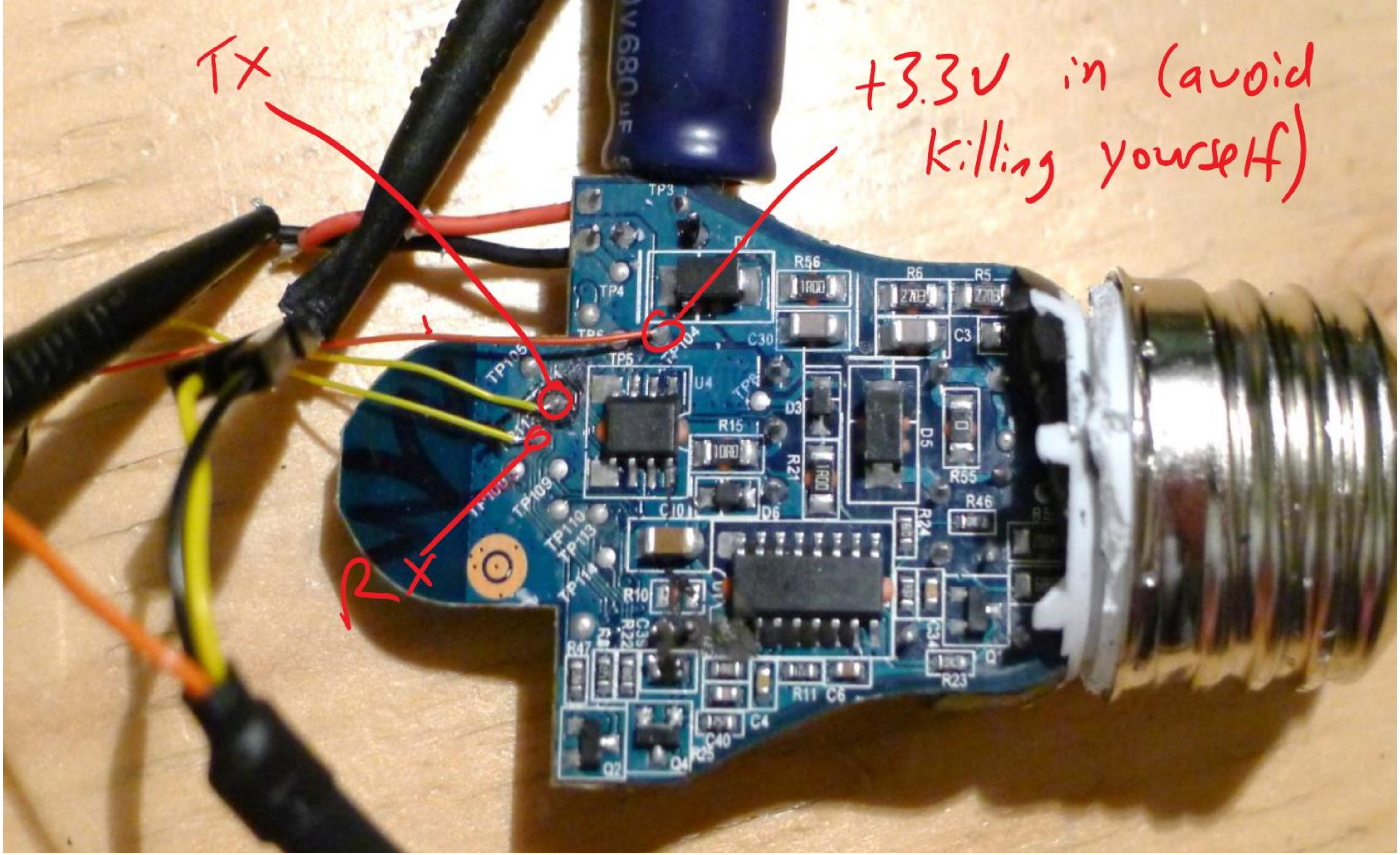






Philps Hue Lamp Teardown




Tx
+3.3V in (avoid killing yourself)
Rx
680uF
TP3
TP4
TP5
TP104
TP105
TP8
TP109
TP110
TP113
TP114
R56
R6
R5
C30
C3
U4
D3
D5
R15
R21
R55
R46
D6
C10
R24
R10
U1
C34
Q
R47
R22
R11
C6
C4
C40
R25
R23
Q2
Q4

Boot sequence debug printout

Locked

```
[Log,Info,ConnectedLamp,MCUCR=0x00,LockBits=0xFC,LowFuse=0xF6,HighFuse=0x9
A,ExtFuse=0xFE]
[Log,Info,ConnectedLamp,devsig=0x1EA803]
[Log,Info,S_DeviceInfo,Booting into normal mode...]
[Log,Info,S_DeviceInfo,DeviceId: Bulb_A19_DimmableWhite_v2]
[Log,Info,N_Security,LIB4.5.75]
[Log,Info,N_Security,KeyBitMask,0x0012]
[Log,Info,ConnectedLamp,Platform version 0.41.0.1,package_ZigBee
117,package_BC_Stack 104,svn 26632]
[Log,Info,ConnectedLamp,Product version WhiteLamp-Atmel 5.38.1.15095,built
by LouvreZLL]
[Log,Info,A_Commissioning,Factory New at Ch: 11]
[TH,Ready,0]
```

Challenges in taking over a preinstalled smart light

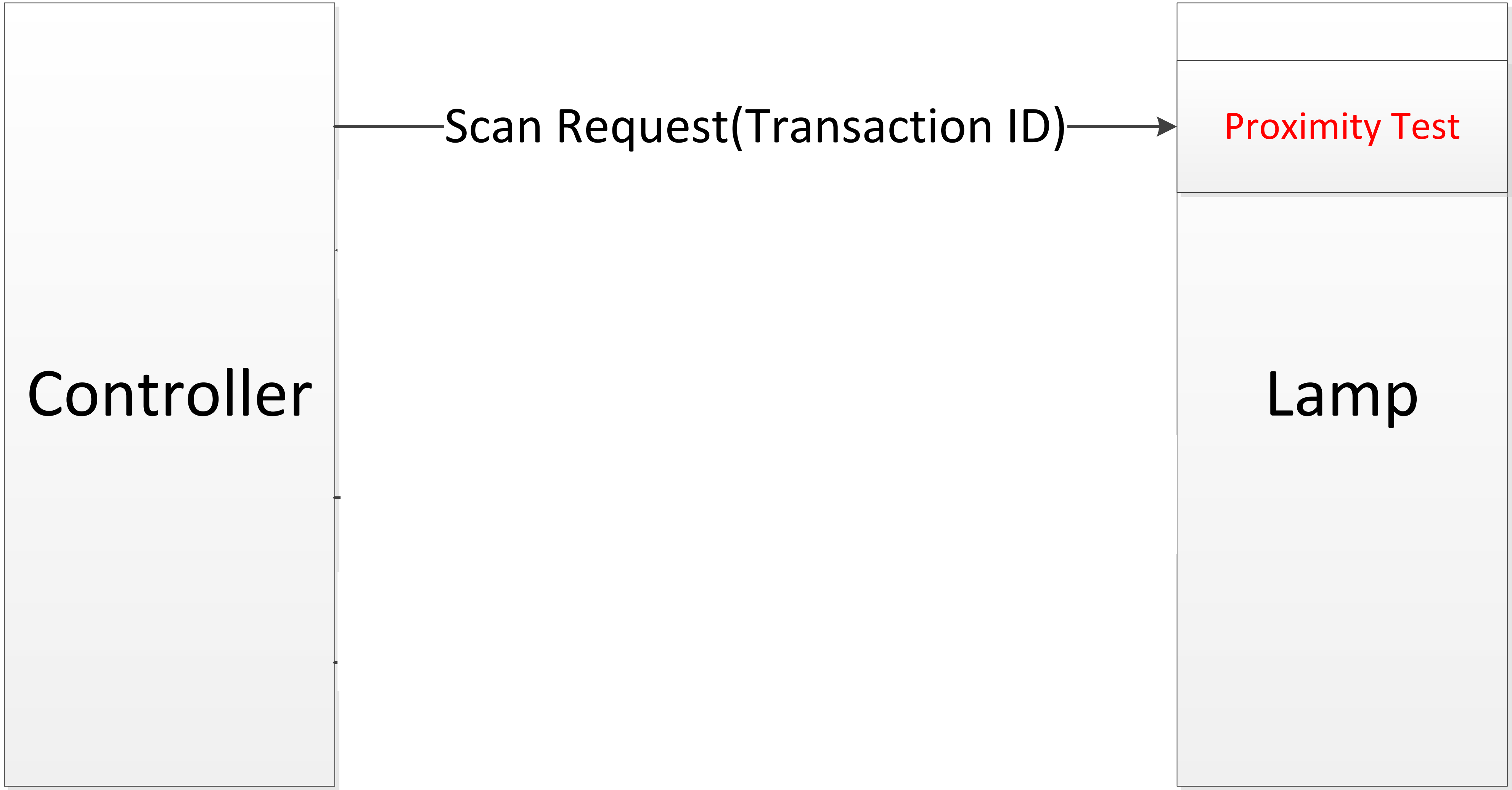
Challenges in taking over a preinstalled smart light

- ZigBee Light Link standard uses multiple cryptographic and security protocols to prevent misuse

Challenges in taking over a preinstalled smart light

- ZigBee Light Link standard uses multiple cryptographic and security protocols to prevent misuse
- In particular, uses a proximity test to make sure that the only way to take control of an already installed Hue lamp is by operating it within 10-20 cm from its new controller

Protocol Session Outline



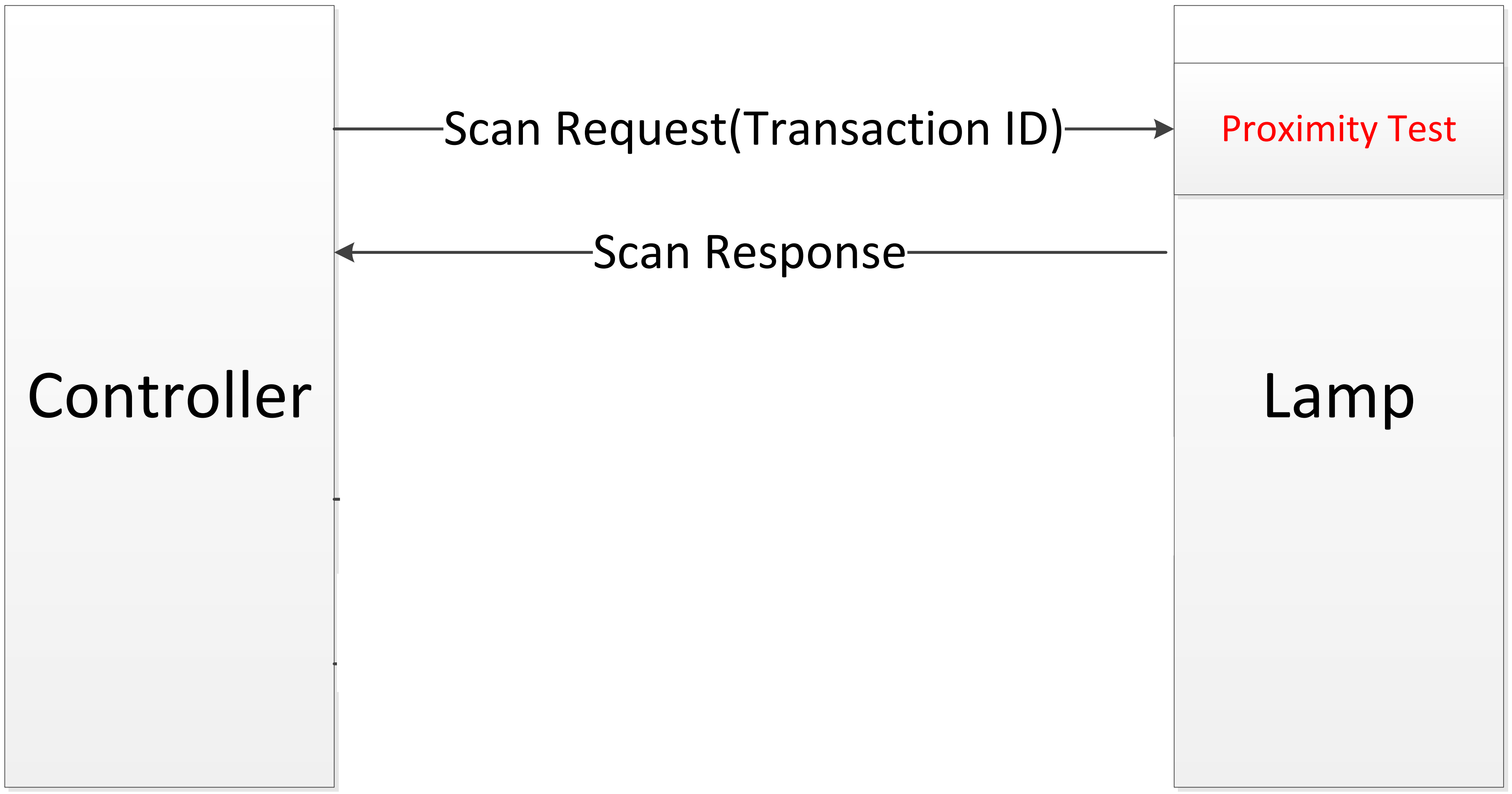
Controller

Scan Request(Transaction ID)

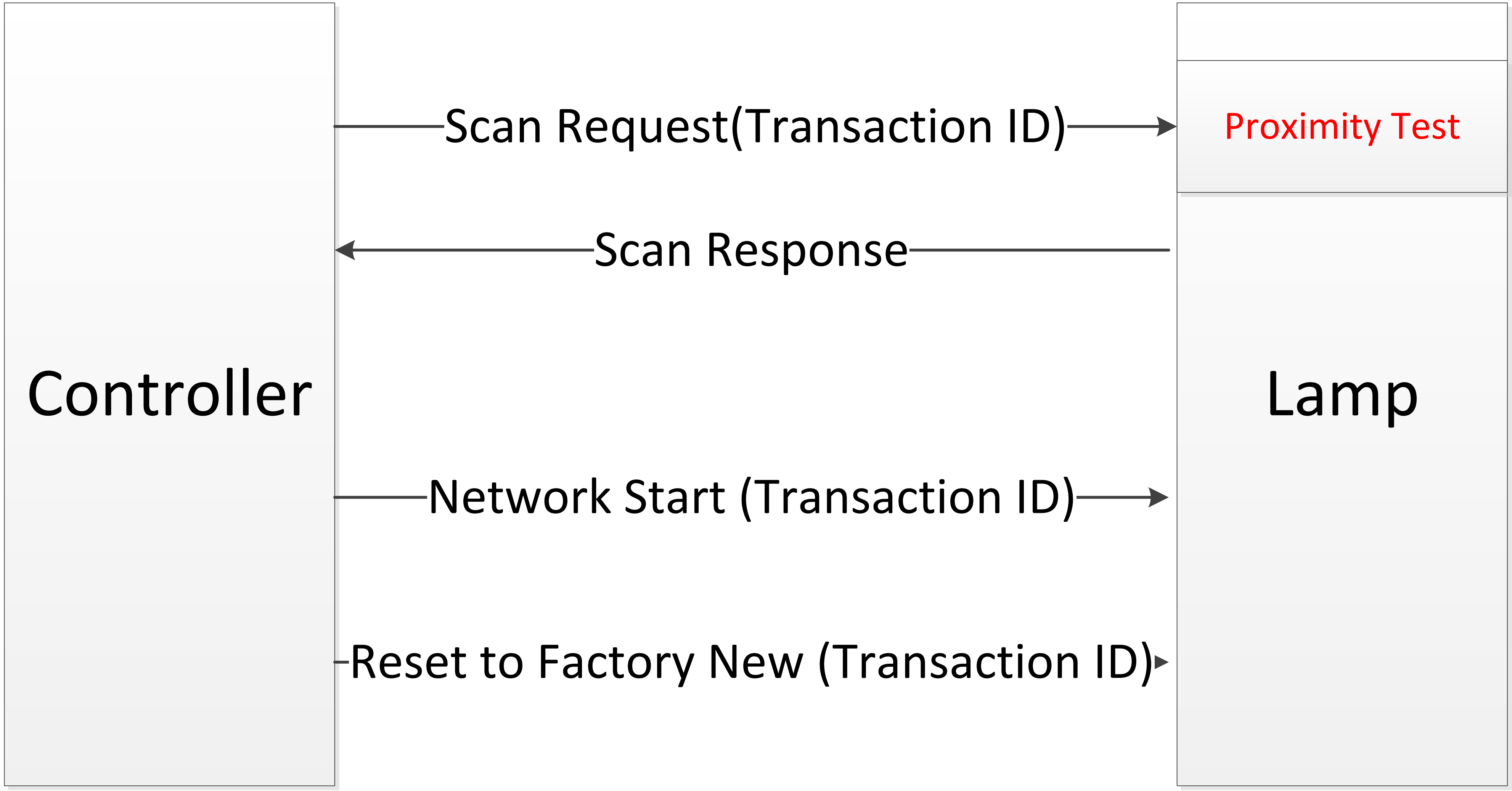
Proximity Test

Lamp

Protocol Session Outline



Protocol Session Outline



Controller

Scan Request(Transaction ID)

Proximity Test

Scan Response

Lamp

Network Start (Transaction ID)

Reset to Factory New (Transaction ID)

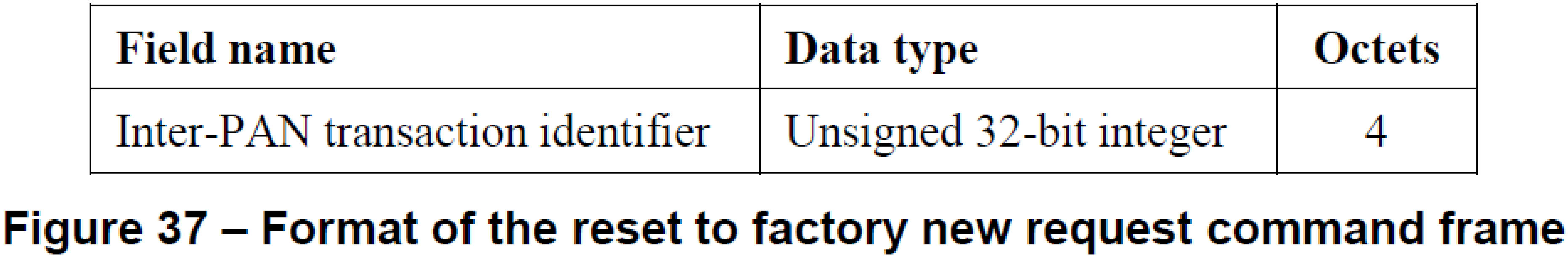
Protocol Implementation Bug

Protocol Implementation Bug

- We want to cause the light to Reset to Factory New

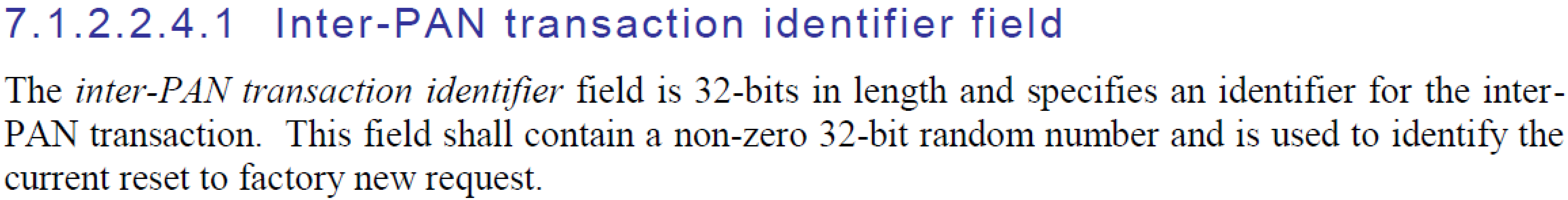
Protocol Implementation Bug

- We want to cause the light to Reset to Factory New



Field name	Data type	Octets
Inter-PAN transaction identifier	Unsigned 32-bit integer	4

Figure 37 – Format of the reset to factory new request command frame

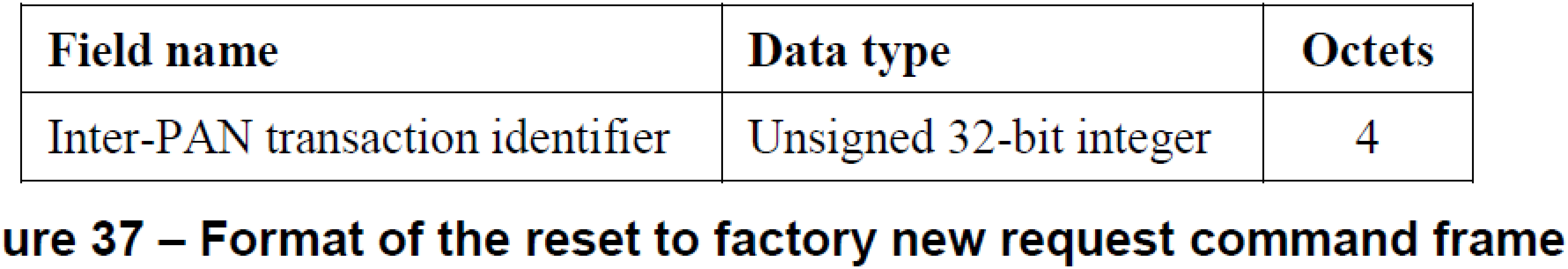


7.1.2.2.4.1 Inter-PAN transaction identifier field

The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction. This field shall contain a non-zero 32-bit random number and is used to identify the current reset to factory new request.

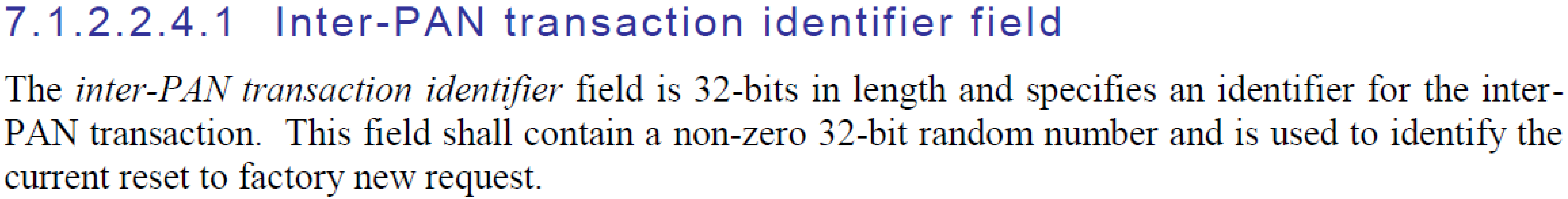
Protocol Implementation Bug

- We want to cause the light to Reset to Factory New



Field name	Data type	Octets
Inter-PAN transaction identifier	Unsigned 32-bit integer	4

Figure 37 – Format of the reset to factory new request command frame



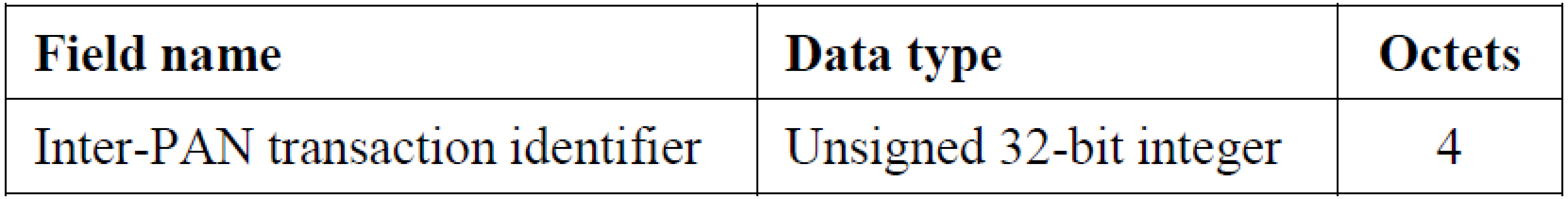
7.1.2.2.4.1 Inter-PAN transaction identifier field

The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction. This field shall contain a non-zero 32-bit random number and is used to identify the current reset to factory new request.

- Can’t set a valid Transaction ID due to proximity test

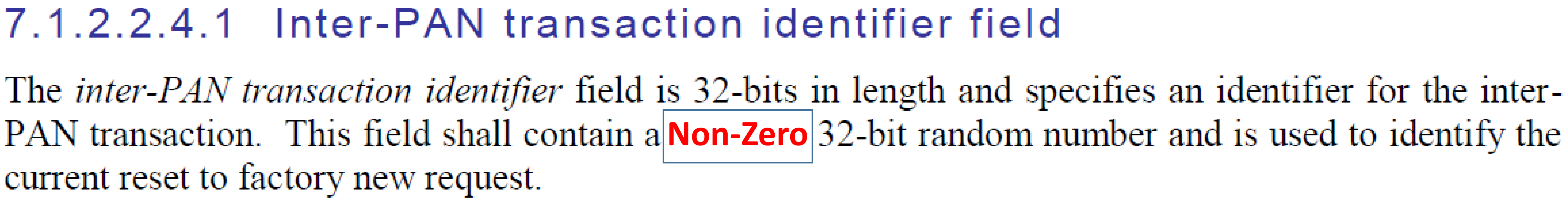
Protocol Implementation Bug

- We want to cause the light to Reset to Factory New



Field name	Data type	Octets
Inter-PAN transaction identifier	Unsigned 32-bit integer	4

Figure 37 – Format of the reset to factory new request command frame



7.1.2.2.4.1 Inter-PAN transaction identifier field

The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction. This field shall contain a **Non-Zero** 32-bit random number and is used to identify the current reset to factory new request.

- Can’t set a valid Transaction ID due to proximity test

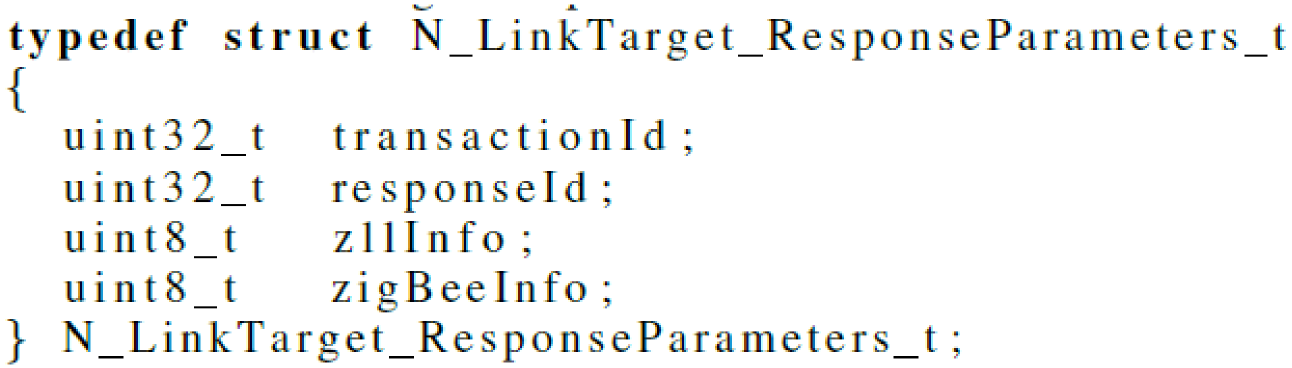
The case of ZERO (day)

The case of ZERO (day)

- How is the Session data is saved in memory?

The case of ZERO (day)

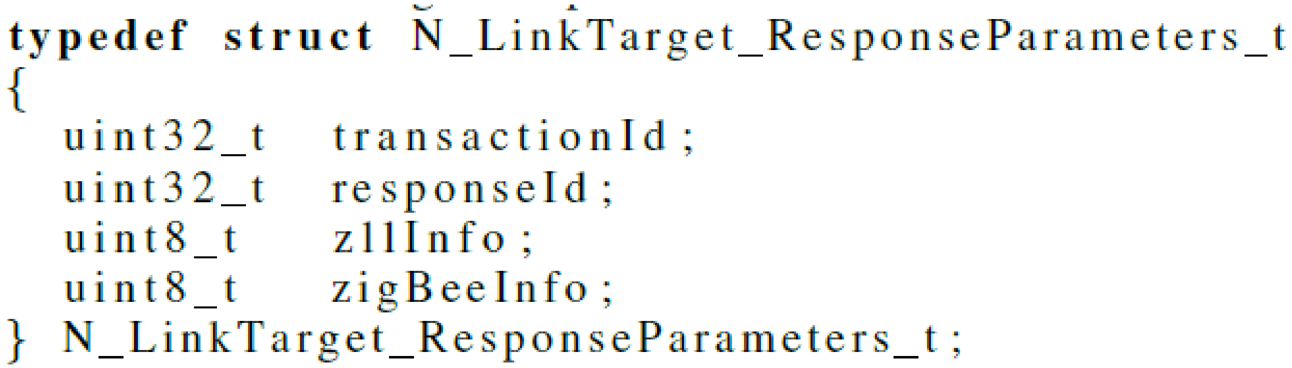
- How is the Session data is saved in memory?



```
typedef struct N̆_Link̂Target_ResponseParameters_t
{
  uint32_t   transactionId;
  uint32_t   responseId;
  uint8_t    zllInfo;
  uint8_t    zigBeeInfo;
} N_LinkTarget_ResponseParameters_t;
```

The case of ZERO (day)

- How is the Session data is saved in memory?

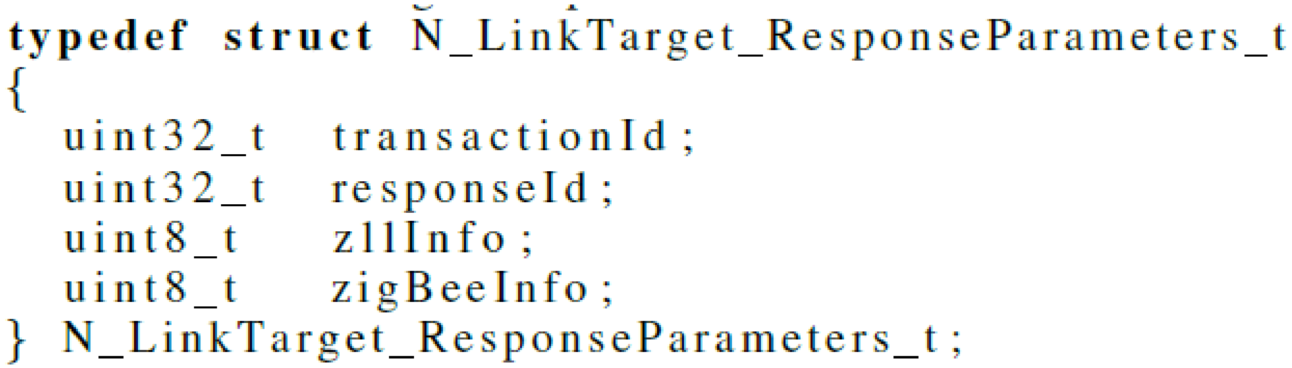


```
typedef struct Ň_LinkTarget_ResponseParameters_t
{
    uint32_t    transactionId;
    uint32_t    responseId;
    uint8_t     zllInfo;
    uint8_t     zigBeeInfo;
} N_LinkTarget_ResponseParameters_t;
```

- What is default values in the struct?

The case of ZERO (day)

- How is the Session data is saved in memory?

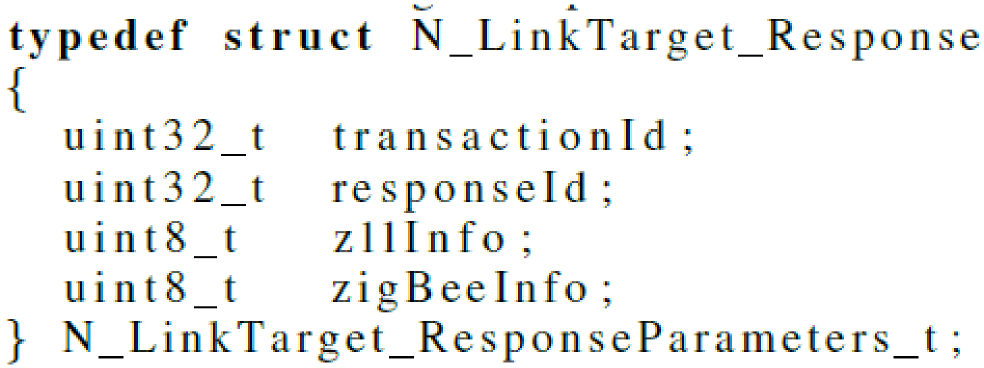


```
typedef struct Ň_LinkTarget_ResponseParameters_t
{
  uint32_t   transactionId;
  uint32_t   responseId;
  uint8_t    zllInfo;
  uint8_t    zigBeeInfo;
} N_LinkTarget_ResponseParameters_t;
```

- What is default values in the struct?
- Well surely it is
  checked on access…

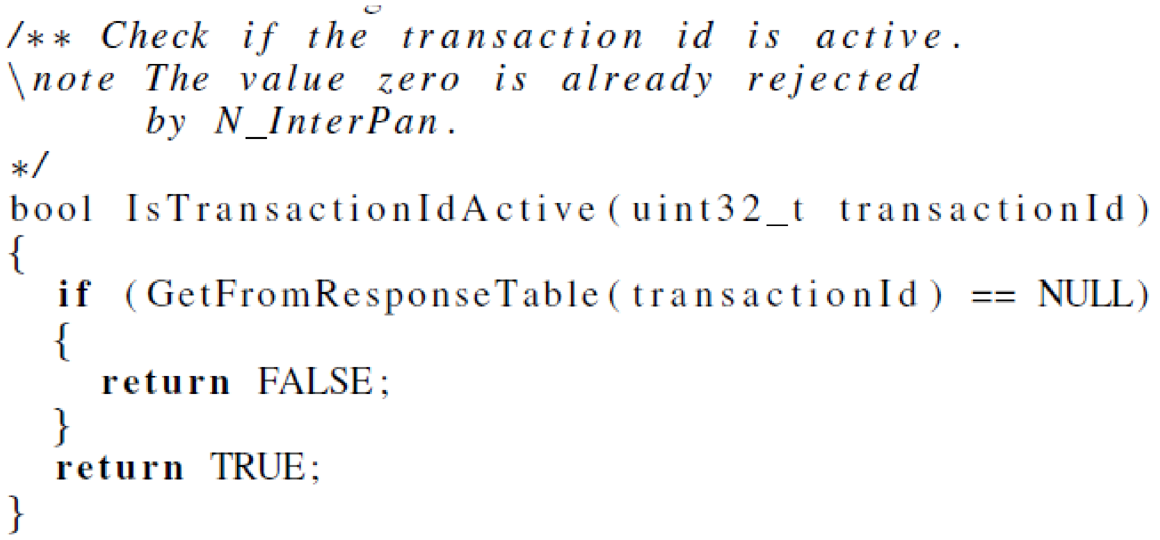
The case of ZERO (day)

- How is the Session data is saved in memory?



```
typedef struct Ň_LinkTarget_ResponseParameters_t
{
   uint32_t   transactionId;
   uint32_t   responseId;
   uint8_t    zllInfo;
   uint8_t    zigBeeInfo;
} N_LinkTarget_ResponseParameters_t;
```

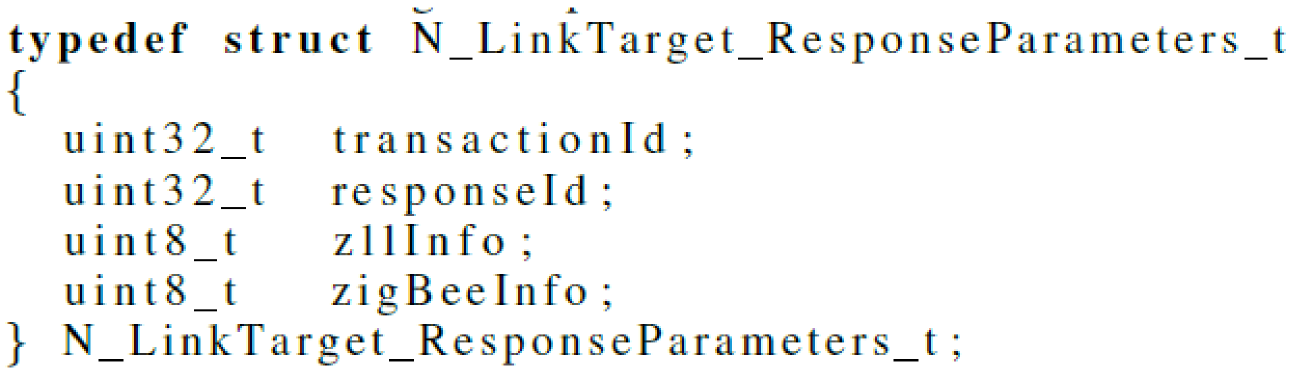
- What is default values in the struct?
- Well surely it is checked on access…



```
/** Check if the transaction id is active.
\note The value zero is already rejected
      by N_InterPan.
*/
bool IsTransactionIdActive(uint32_t transactionId)
{
  if (GetFromResponseTable(transactionId) == NULL)
  {
    return FALSE;
  }
  return TRUE;
}
```

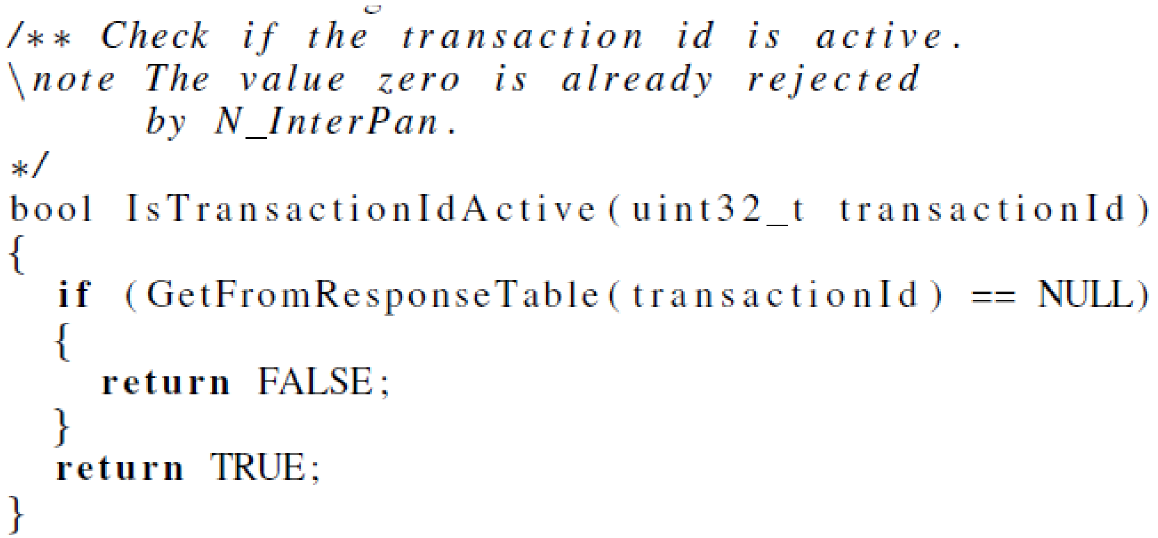
The case of ZERO (day)

- How is the Session data is saved in memory?



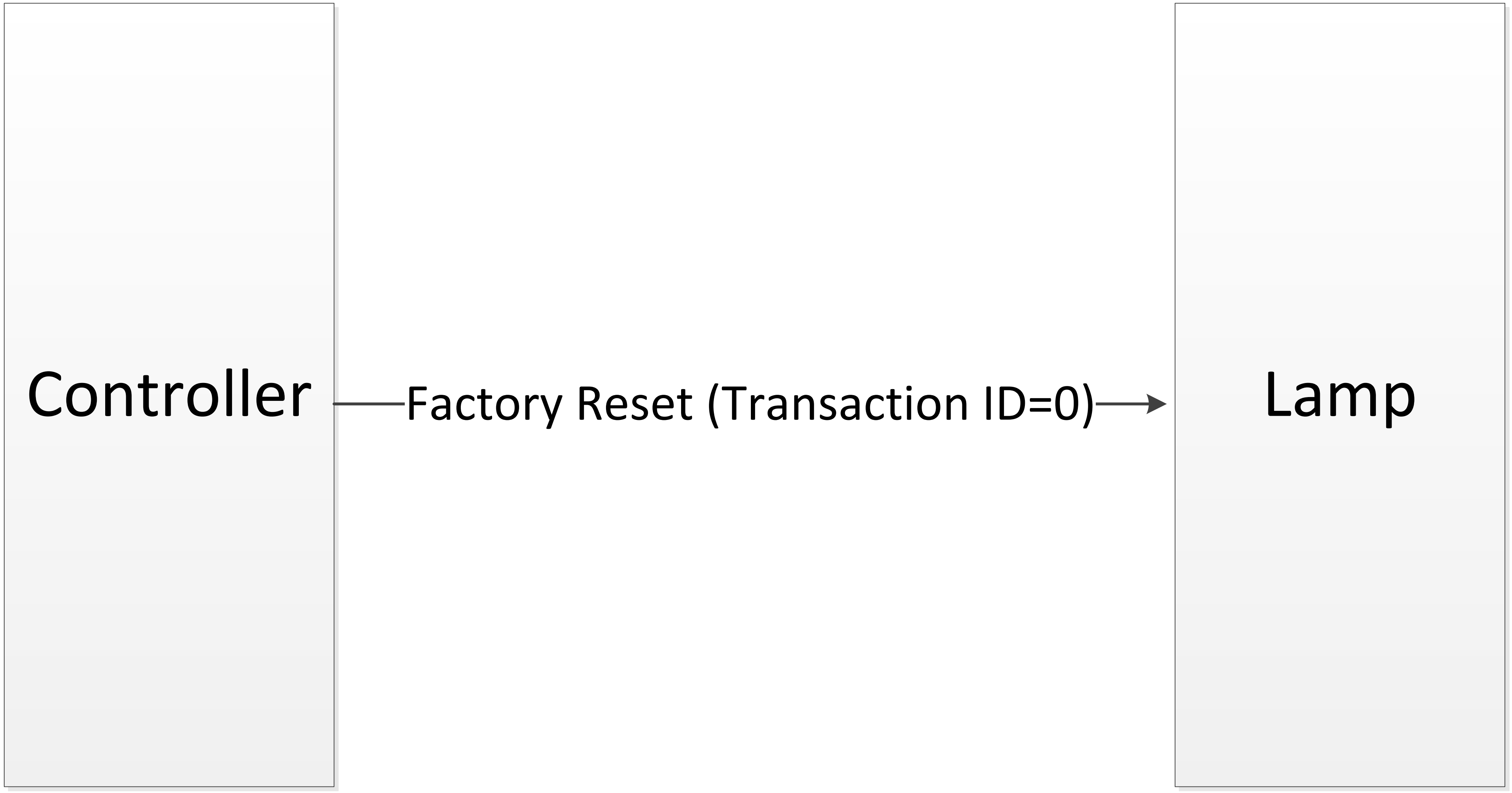
```
typedef struct Ň_LinkTarget_ResponseParameters_t
{
  uint32_t  transactionId;
  uint32_t  responseId;
  uint8_t   zllInfo;
  uint8_t   zigBeeInfo;
} N_LinkTarget_ResponseParameters_t;
```

- What is default values in the struct?
- Well surely it is checked on access…
- Just on Scan Request message



```
/** Check if the transaction id is active.
\note The value zero is already rejected
      by N_InterPan.
*/
bool IsTransactionIdActive(uint32_t transactionId)
{
  if (GetFromResponseTable(transactionId) == NULL)
  {
    return FALSE;
  }
  return TRUE;
}
```

Protocol Attack Outline

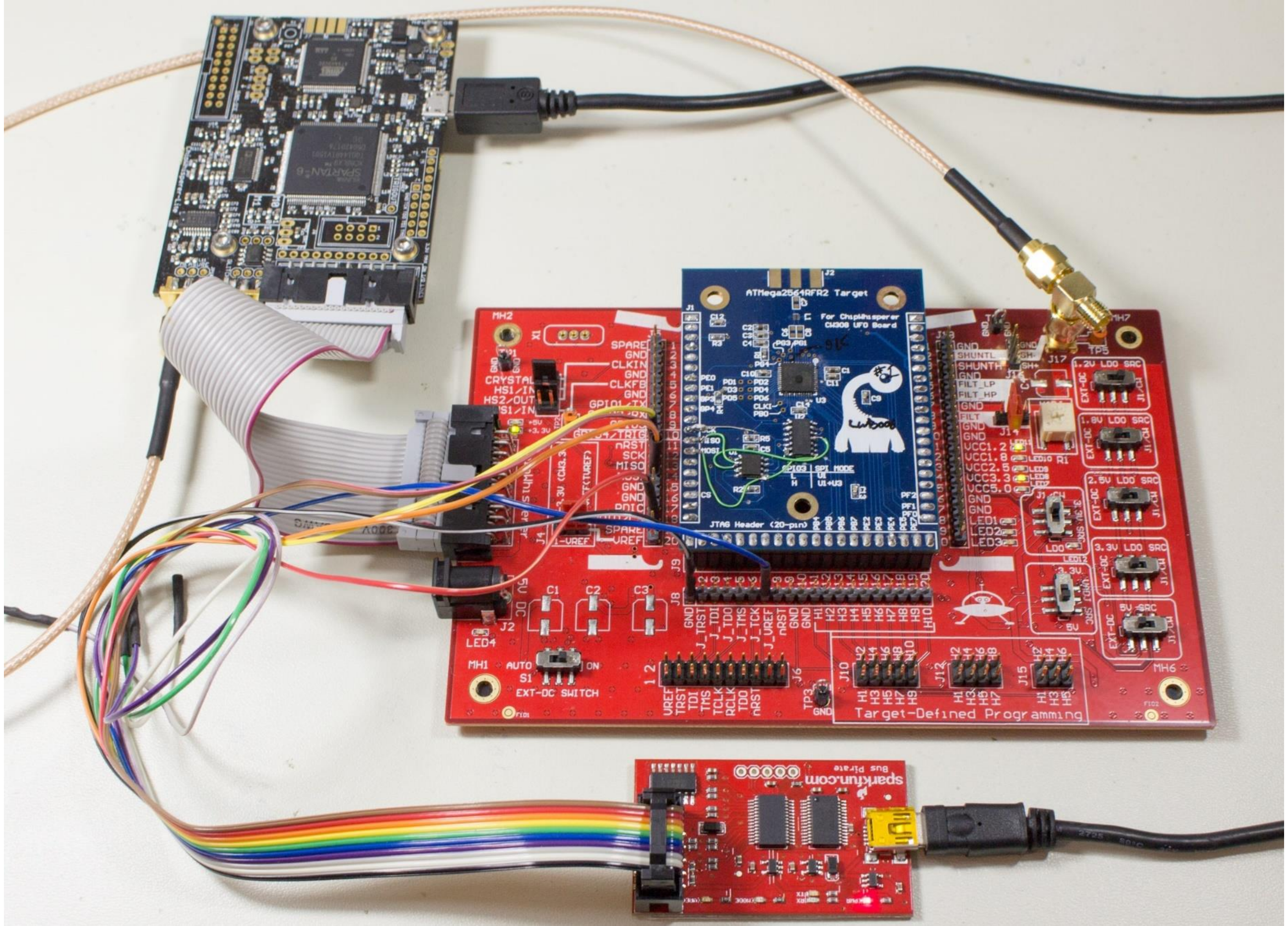


We bought a cheap and lightweight commercial Zigbee evaluation kit:

ZigBee WarFlying -
Taking over a building’s lights

By launching a drone carrying a fully automated attack equipment 400 meters away

Spreading everywhere



Getting inside the SoC

Getting inside the SoC

- SoC with Harvard architecture

Getting inside the SoC

- SoC with Harvard architecture
- “Open source” stack, but no binaries, and relatively good code

Getting inside the SoC

- SoC with Harvard architecture
- “Open source” stack, but no binaries, and relatively good code
- Lets use the software update

Getting inside the SoC

- SoC with Harvard architecture

- “Open source” stack, but no binaries, and relatively good code

- Lets use the software update
  - No software updates for my lights

Getting inside the SoC

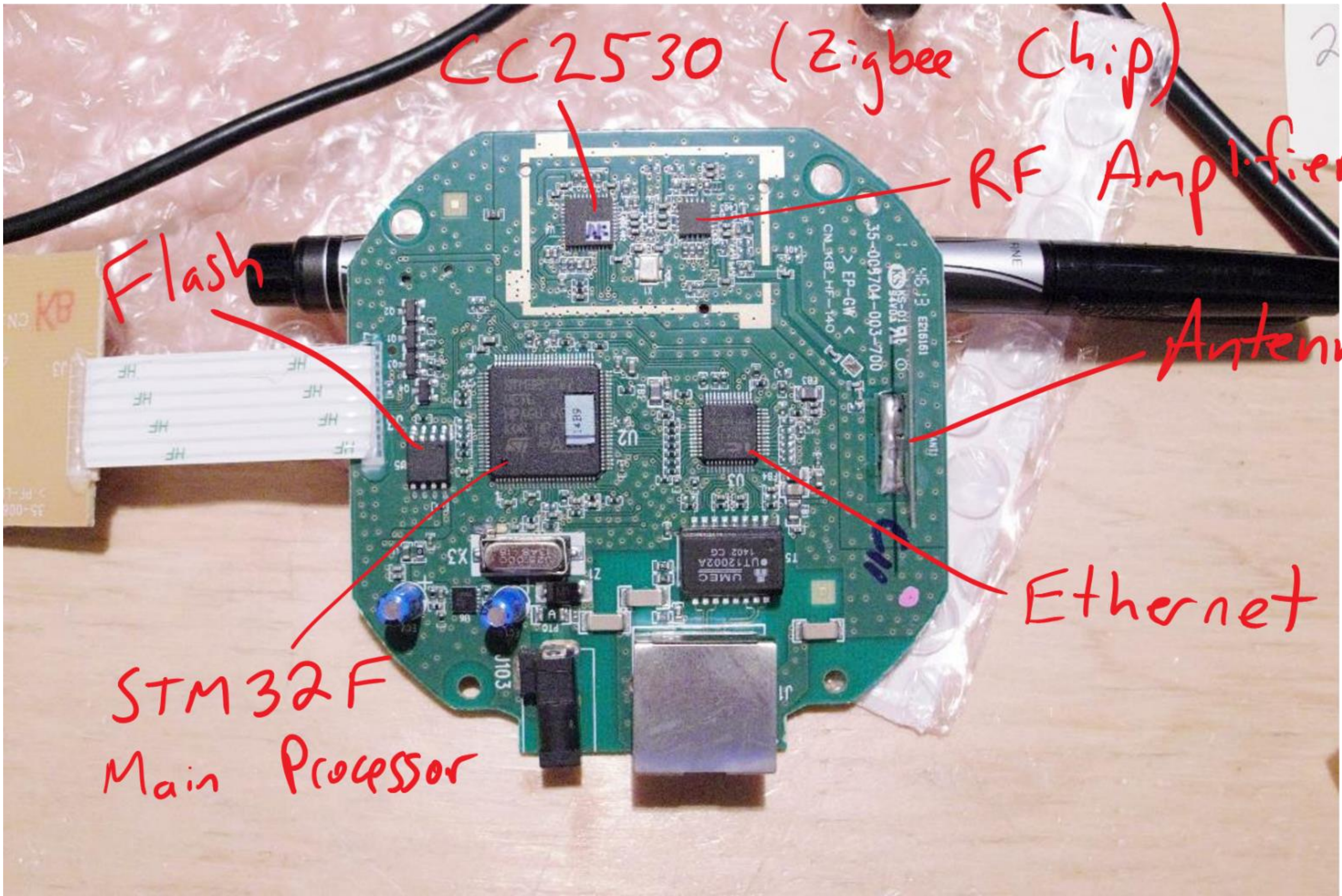
- SoC with Harvard architecture
- “Open source” stack, but no binaries, and relatively good code
- Lets use the software update
  - No software updates for my lights
  - Can’t buy the older models

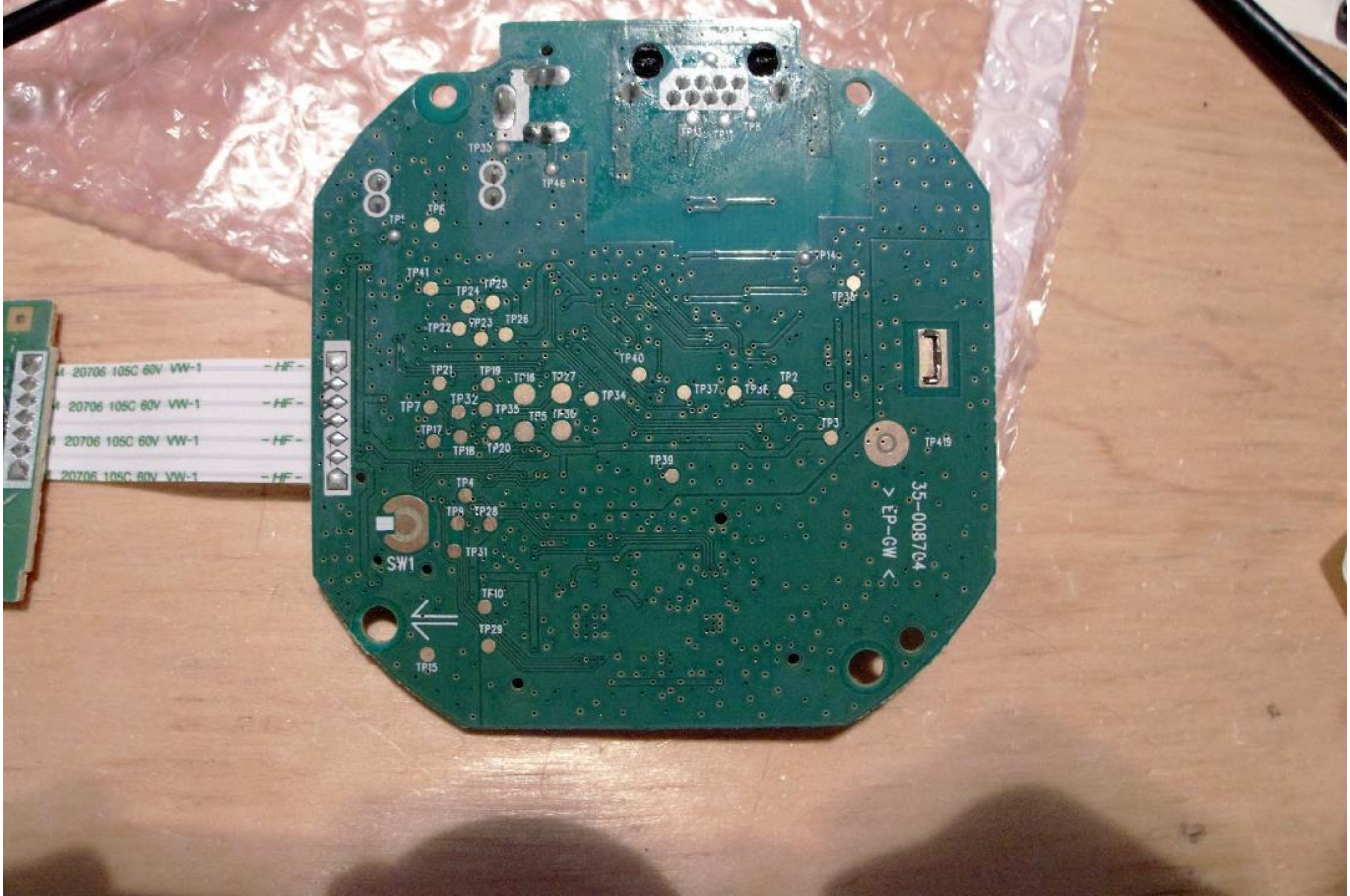
Getting inside the SoC

- SoC with Harvard architecture
- “Open source” stack, but no binaries, and relatively good code
- Lets use the software update
  - No software updates for my lights
  - Can’t buy the older models
  - Start with the bridge

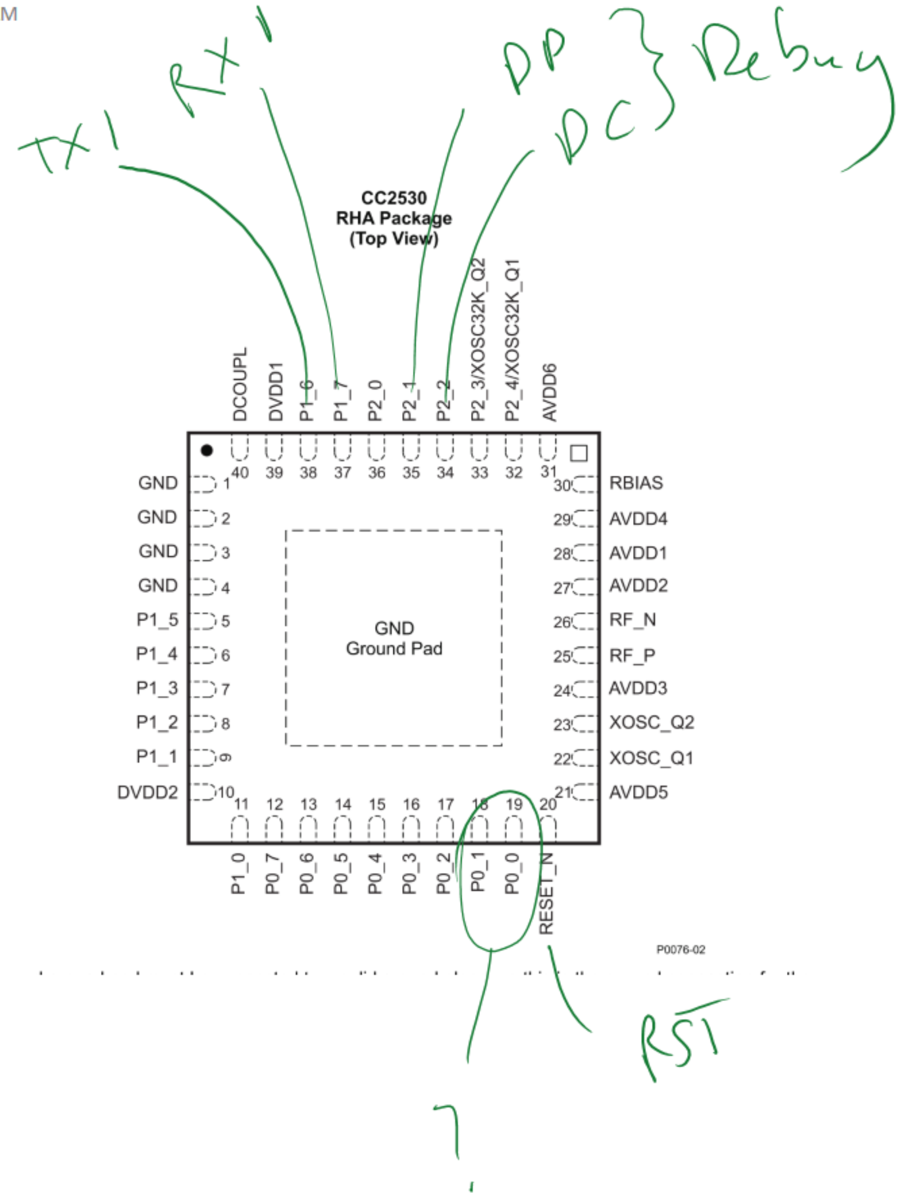
First try – older TI based model

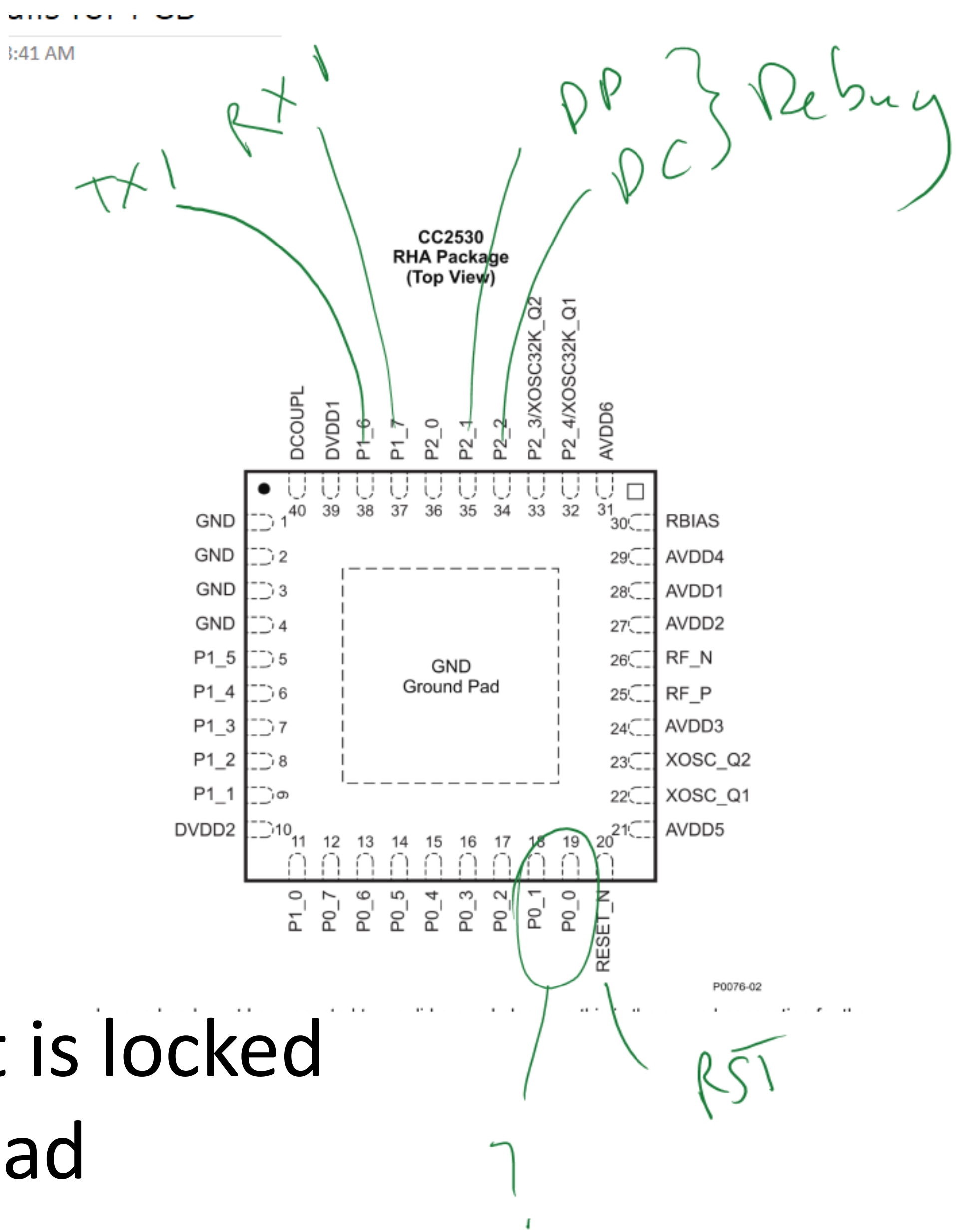
The one that got away


CC2530 (Zigbee Chip)
RF Amplifier
Flash
Antenna
Ethernet
STM32F
Main Processor


TP33
TP46
TP13
TP11
TP8
TP1
TP6
TP14
TP41
TP24
TP25
TP22
TP23
TP26
TP38
TP40
TP21
TP19
TP16
TP27
TP34
TP37
TP36
TP2
TP7
TP32
TP35
TP5
TP30
TP17
TP18
TP20
TP3
TP419
TP39
TP4
TP9
TP28
TP31
SW1
TP10
TP29
TP15
35-0087C4
> EP-GW <
20706 105C 60V VW-1
-HF-

:41 AM


TX1
RX1
DP
DC
Debug
RST
?
CC2530
RHA Package
(Top View)
DCOUPL
DVDD1
P1_6
P1_7
P2_0
P2_1
P2_2
P2_3/XOSC32K_Q2
P2_4/XOSC32K_Q1
AVDD6
40
39
38
37
36
35
34
33
32
31
GND 1
GND 2
GND 3
GND 4
P1_5 5
P1_4 6
P1_3 7
P1_2 8
P1_1 9
DVDD2 10
30 RBIAS
29 AVDD4
28 AVDD1
27 AVDD2
26 RF_N
25 RF_P
24 AVDD3
23 XOSC_Q2
22 XOSC_Q1
21 AVDD5
GND
Ground Pad
11
12
13
14
15
16
17
18
19
20
P1_0
P0_7
P0_6
P0_5
P0_4
P0_3
P0_2
P0_1
P0_0
RESET_N
P0076-02



- Try to connect, it is locked for debug and read

Extracting Keys from Second Generation Zigbee Chips

Travis Goodspeed
1933 Black Oak Street
Jefferson City, TN, USA
travis@radiantmachines.com

ABSTRACT

First generation Zigbee chips were SPI slaves with no internal processing beyond cryptographic acceleration. Extracting a key was as simple as spying on the SPI transactions. The second generation chips, typified by the CC2430 from Texas Instruments and the EM250 from Ember, contain both a microcontroller and a radio, making the SPI sniffing attack all but irrelevant. Nevertheless, both chips are vulnerable to local key extraction. This paper describes techniques for doing so, focusing on the CC2430 as the EM250 has no protection against outside access. Recommendations are made for defending CC2430 firmware by using compiler directives to place sensitive information in flash memory, rather than in RAM. All Chipcon radios with 8051 cores released prior to the publication of this paper are expected to be vulnerable.

Keywords

Zigbee, CC2430, EM250, System on a Chip (SoC)

1. GENERATIONS

First generation Zigbee chips, such as the CC2420, were simply digital radios with SPI interfaces and a bit of hardware-accelerated cryptography. They could not run a Zigbee stack themselves, but rather relied upon an external microcontroller cores were added for convenience, not security, as will be explained below.

The third generation of chips will include more powerful microprocessors and–hopefully–a lot more security. The offering from Texas Instruments is the CC430 family, based upon the MSP430X2 processor. Ember will be using the Arm Cortex M3 in its EM300 series. These chips are out of scope for this paper, as they are not yet commercially available. Also, Freescale's line of radios have not yet been examined by the author, but they will be in the near future.

2. CONCERNING THE EM250

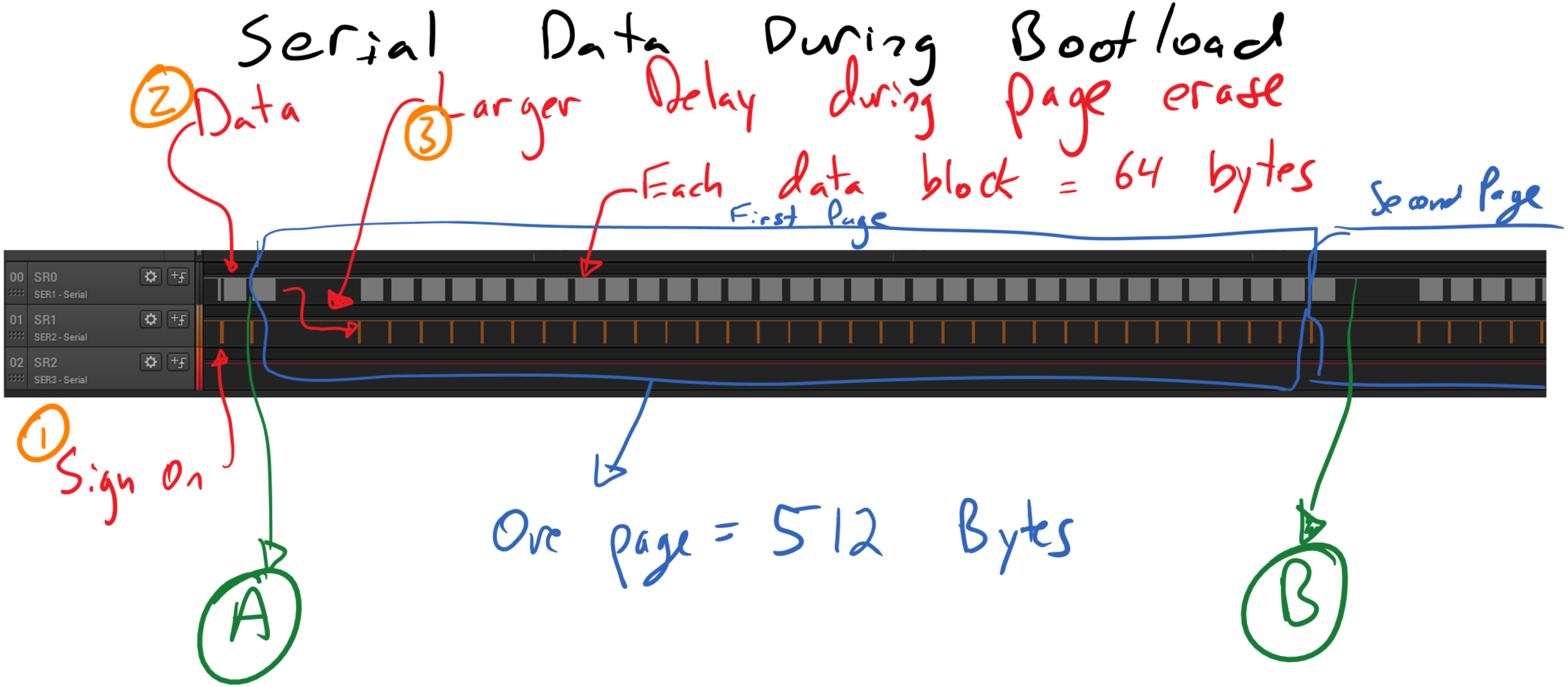
The Ember EM250 contains a 16-bit XAP2b microprocessor from Cambridge Consultants Ltd.[3] Debugging support is provided by that firm's proprietary SIF protocol, with hardware and software available only through Ember. SIF itself is a variant of JTAG.

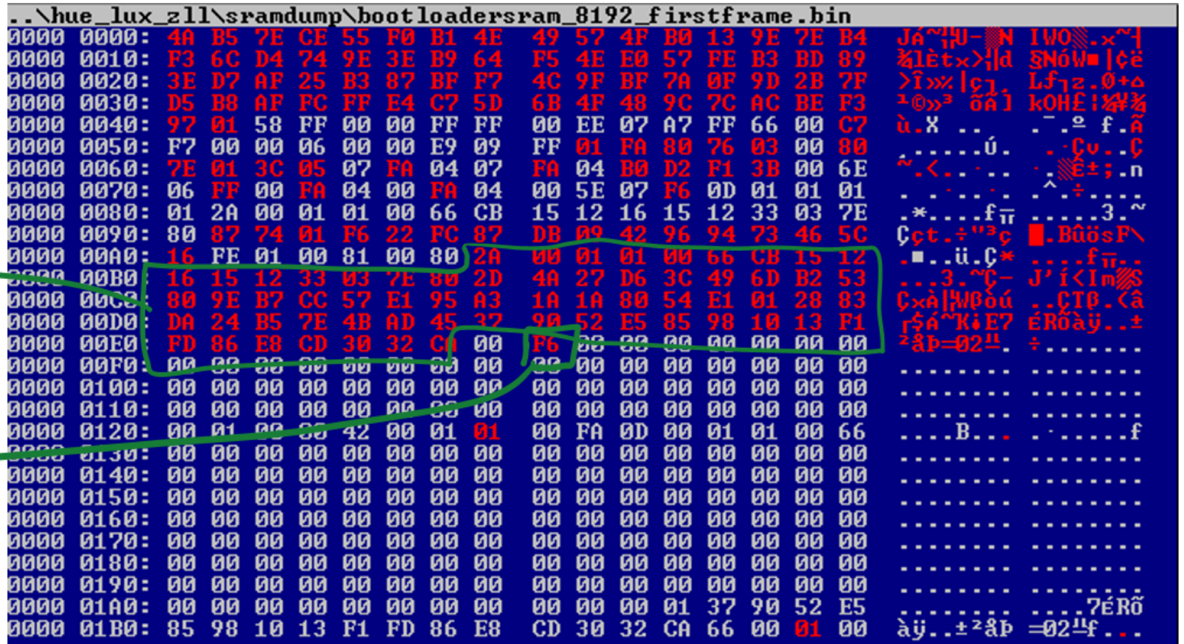
While the datasheet and various piece of marketing literature claim "The EM250 employs a configurable memory protection scheme usually found on larger microcontrollers.", this refers not to a debugging fuse or bootloader password, but rather to protection from accidental self-corruption of memory. This is in the form of Application/System separation, allowing the EmberZNet stack to defend certain regions

Looking at EBL source: https://github.com/lee-wei/CC2540/blob/master/Projects/ble/util/EBL/app/sbl_exec.c

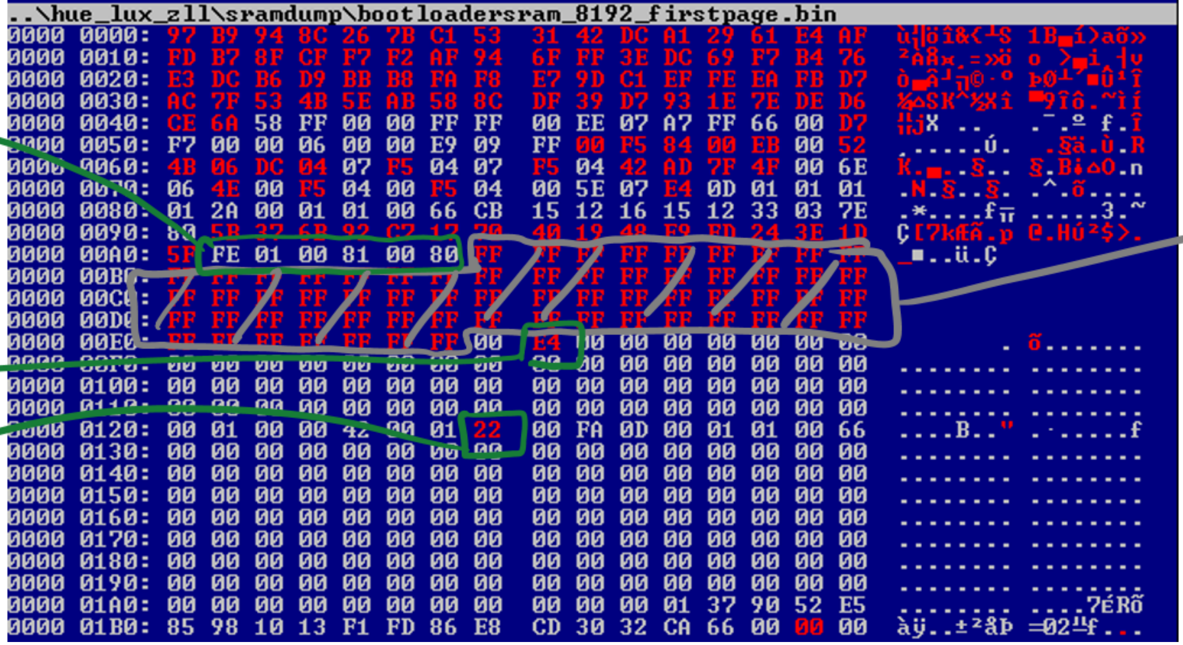
```
static void aesLoadKey(void)
{
    // Read the security key from flash 1 byte at a time to thwart an interrupt & read XDATA attack.
    uint8 *keyPtr = (uint8 *)aesKey;

    ENCCS = ECB | AES_LOAD_KEY | 0x01;

    // 'while ((ENCCS & BV(3)) == 0)' was seen to hang without #pragma optimize=none.
    // So proactively adding this wait after every 'ENCCS = ' which empirically seems to work.
    ASM_NOP; ASM_NOP; ASM_NOP; ASM_NOP; ASM_NOP; ASM_NOP; ASM_NOP; ASM_NOP;

    for (uint8 cnt = 0; cnt < KEY_BLENGTH; cnt++)
    {
      ENCDI = *keyPtr++;
    }
}
```

Inner bridge software update





```
..\hue_lux_zll\sramdump\bootloadersram_8192_firstframe.bin
0000 0000: 4A B5 7E CE 55 F0 B1 4E  49 57 4F B0 13 9E 7E B4   JÁ~╬U-▓N IWO░.x~╢
0000 0010: F3 6C D4 74 9E 3E B9 64  F5 4E E0 57 FE B3 BD 89   ¾lÈtx>╣d §NóW■│¢ë
0000 0020: 3E D7 AF 25 B3 87 BF F7  4C 9F BF 7A 0F 9D 2B 7F   >Î»%│ç┐. Lƒ┐z.Ø+⌂
0000 0030: D5 B8 AF FC FF E4 C7 5D  6B 4F 48 9C 7C AC BE F3   ¹©»³ õÃ] kOH£¦¼╛¾
0000 0040: 97 01 58 FF 00 00 FF FF  00 EE 07 A7 FF 66 00 C7   ù.X ..   .¯.º f.Ã
0000 0050: F7 00 00 06 00 00 E9 09  FF 01 FA 80 76 03 00 80   ......Ú.  ..·Çv..Ç
0000 0060: 7E 01 3C 05 07 FA 04 07  FA 04 B0 D2 F1 3B 00 6E   ~.<...·.. ·.░Ê±;.n
0000 0070: 06 FF 00 FA 04 00 FA 04  00 5E 07 F6 0D 01 01 01   . .·..·. .^.÷....
0000 0080: 01 2A 00 01 01 00 66 CB  15 12 16 15 12 33 03 7E   .*....f╦ .....3.~
0000 0090: 80 87 74 01 F6 22 FC 87  DB 09 42 96 94 73 46 5C   Ççt.÷"³ç █.BûösF\
0000 00A0: 16 FE 01 00 81 00 80 2A  00 01 01 00 66 CB 15 12   .■..ü.Ç* ....f╦..
0000 00B0: 16 15 12 33 03 7E 80 2D  4A 27 D6 3C 49 6D B2 53   ...3.~Ç- J'Í<Im▓S
0000 00C0: 80 9E B7 CC 57 E1 95 A3  1A 1A 80 54 E1 01 28 83   Ç×À╠WßòúÇ ..ÇTß.(â
0000 00D0: DA 24 B5 7E 4B AD 45 37  90 52 E5 85 98 10 13 F1   ┌$Á~K¡E7 ÉRÕàÿ..±
0000 00E0: FD 86 E8 CD 30 32 CA 00  F6 00 00 00 00 00 00 00   ²åÞ═02╩. ÷.......
0000 00F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0100: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0110: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0120: 00 01 00 00 42 00 01 01  00 FA 0D 00 01 01 00 66   ....B... .·.....f
0000 0130: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0140: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0150: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0160: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0170: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0180: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0190: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 01A0: 00 00 00 00 00 00 00 00  00 00 00 01 37 90 52 E5   ........ ....7ÉRÕ
0000 01B0: 85 98 10 13 F1 FD 86 E8  CD 30 32 CA 66 00 01 00   àÿ..±²åÞ ═02╩f...
```

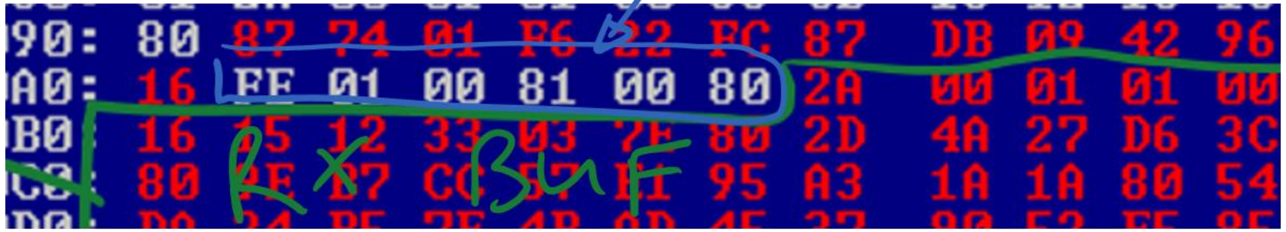


```
..\hue_lux_zll\sramdump\bootloadersram_8192_firstpage.bin
0000 0000: 97 B9 94 8C 26 7B C1 53  31 42 DC A1 29 61 E4 AF   ù╣öî&{┴S 1B▄í)aõ»
0000 0010: FD B7 8F CF F7 F2 AF 94  6F FF 3E DC 69 F7 B4 76   ²ÀÅ¤.=»ö o >▄i.┤v
0000 0020: E3 DC B6 D9 BB B8 FA F8  E7 9D C1 EF FE EA FB D7   Ò▄Â┘╗©·° þØ┴´■Û¹Î
0000 0030: AC 7F 53 4B 5E AB 58 8C  DF 39 D7 93 1E 7E DE D6   ¼⌂SK^½Xî ▀9Îô.~Ìí
0000 0040: CE 6A 58 FF 00 00 FF FF  00 EE 07 A7 FF 66 00 D7   ╬jX ..   .¯.º f.Î
0000 0050: F7 00 00 06 00 00 E9 09  FF 00 F5 84 00 EB 00 52   ......Ú.  .§ä.Ù.R
0000 0060: 4B 06 DC 04 07 F5 04 07  F5 04 42 AD 7F 4F 00 6E   K.▄..§.. §.B¡⌂O.n
0000 0070: 06 4E 00 F5 04 00 F5 04  00 5E 07 E4 0D 01 01 01   .N.§..§. .^.õ....
0000 0080: 01 2A 00 01 01 00 66 CB  15 12 16 15 12 33 03 7E   .*....f╦ .....3.~
0000 0090: 80 5B 37 6B 92 C7 17 70  40 19 48 E9 FD 24 3E 1D   Ç[7kÆÃ.p @.Hú²$>.
0000 00A0: 5F FE 01 00 81 00 80 FF  FF FF FF FF FF FF FF FF   _■..ü.Ç
0000 00B0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 00C0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 00D0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 00E0: FF FF FF FF FF FF FF 00  E4 00 00 00 00 00 00 00          . õ.......
0000 00F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0100: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0110: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0120: 00 01 00 00 42 00 01 22  00 FA 0D 00 01 01 00 66   ....B.." .·.....f
0000 0130: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0140: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0150: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0160: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0170: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0180: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 0190: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0000 01A0: 00 00 00 00 00 00 00 00  00 00 00 01 37 90 52 E5   ........ ....7ÉRÕ
0000 01B0: 85 98 10 13 F1 FD 86 E8  CD 30 32 CA 66 00 00 00   àÿ..±²åÞ ═02╩f...
```

TX BUFFER ATTACK

```
for(uint8 i=0; i < data_to_send; i++){
    uart_write(tx_buf[i]);
}
```

Tx Buffer

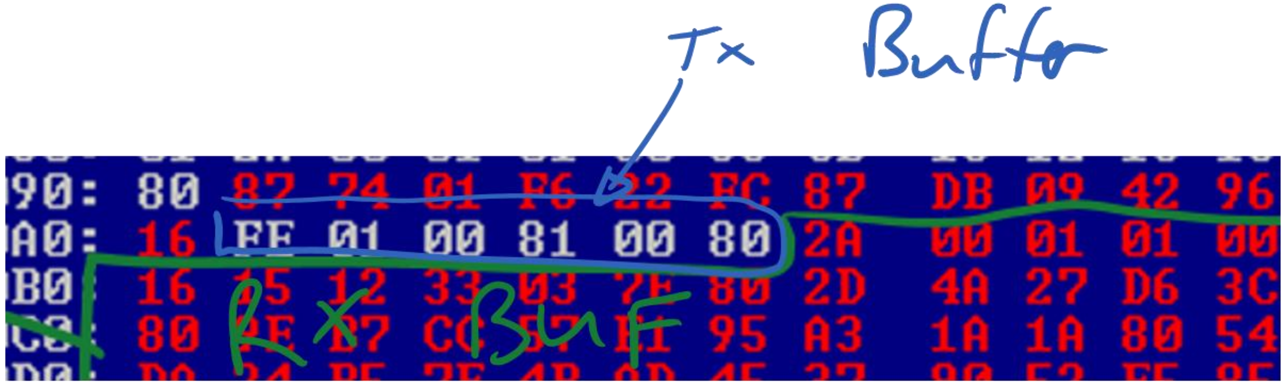


```
90: 80 87 74 01 F6 22 FC 87  DB 09 42 96
A0: 16 FF 01 00 81 00 80 2A  00 01 01 00
B0: 16 15 12 33 03 7E 80 2D  4A 27 D6 3C
C0: 80 AE B7 C0 5[illegible] E1 95 A3  1A 1A 80 54
```

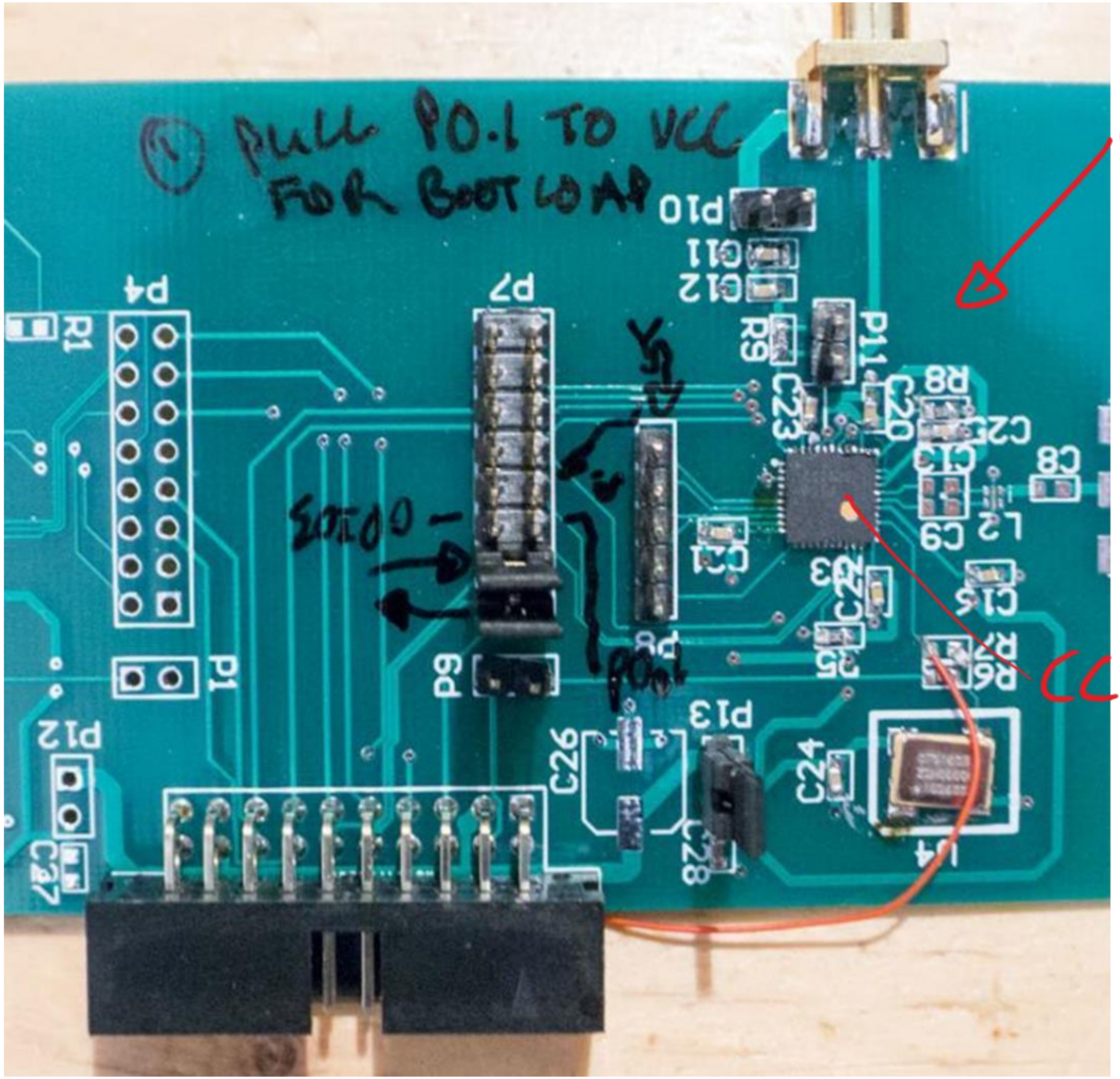
TX BUFFER ATTACK

```
for(uint8 i=0; i < data_to_send; i++){
    uart_write(tx_buf[i]);
}
```

Glitch Attack!



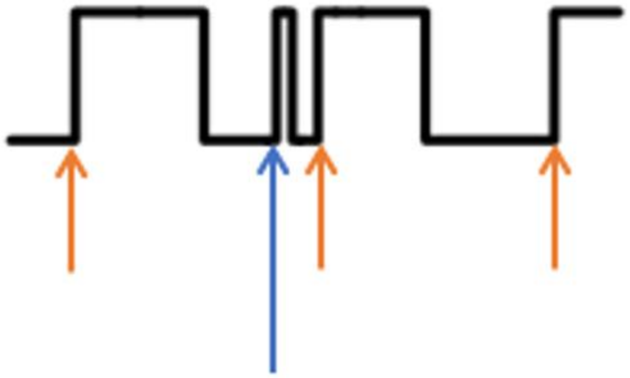
```
90: 80 87 74 01 F6 22 FC 87 DB 09 42 96
A0: 16 FE 01 00 81 00 80 2A 00 01 01 00
B0: 16 [illegible] 12 33 03 7E 80 2D 4A 27 D6 3C
C0: 80 [illegible] B7 [illegible] [illegible] E1 95 A3 1A 1A 80 54
```

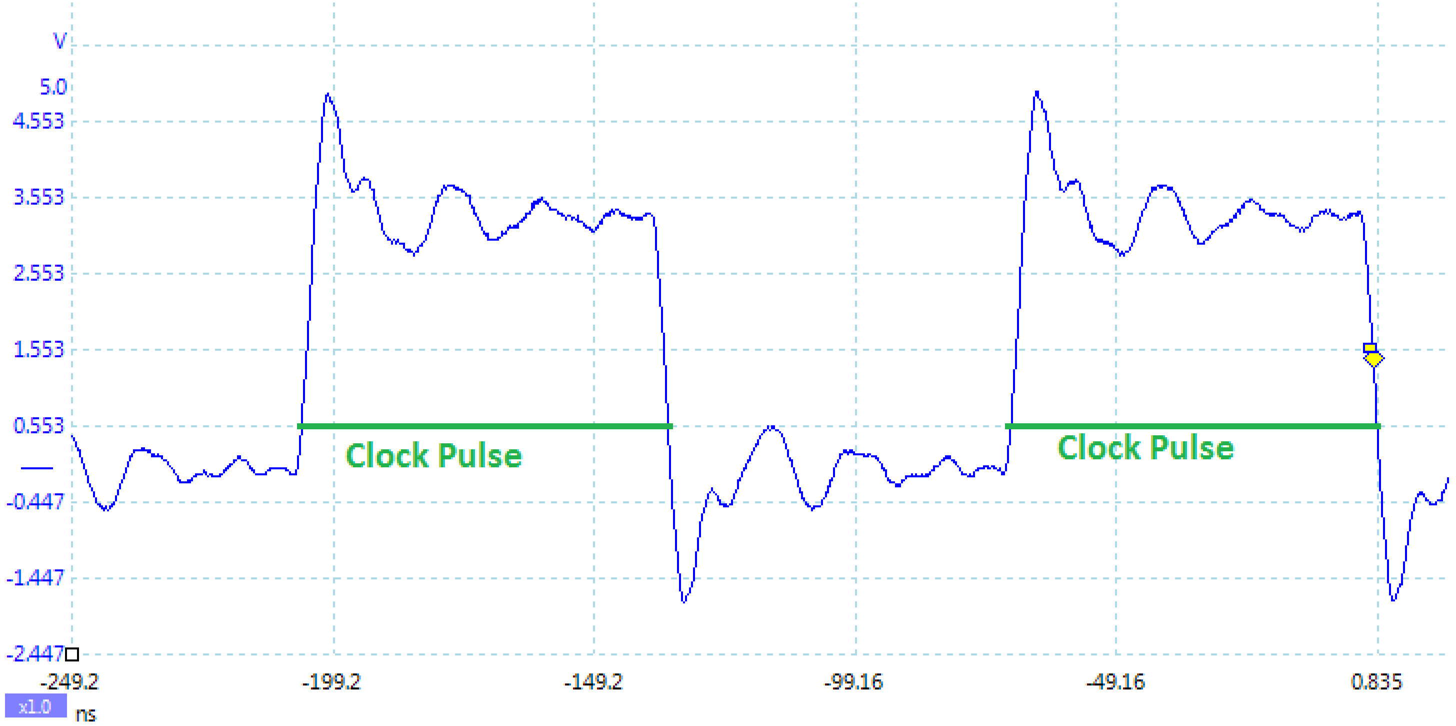


Custom PCB

CC2530 from Bridge

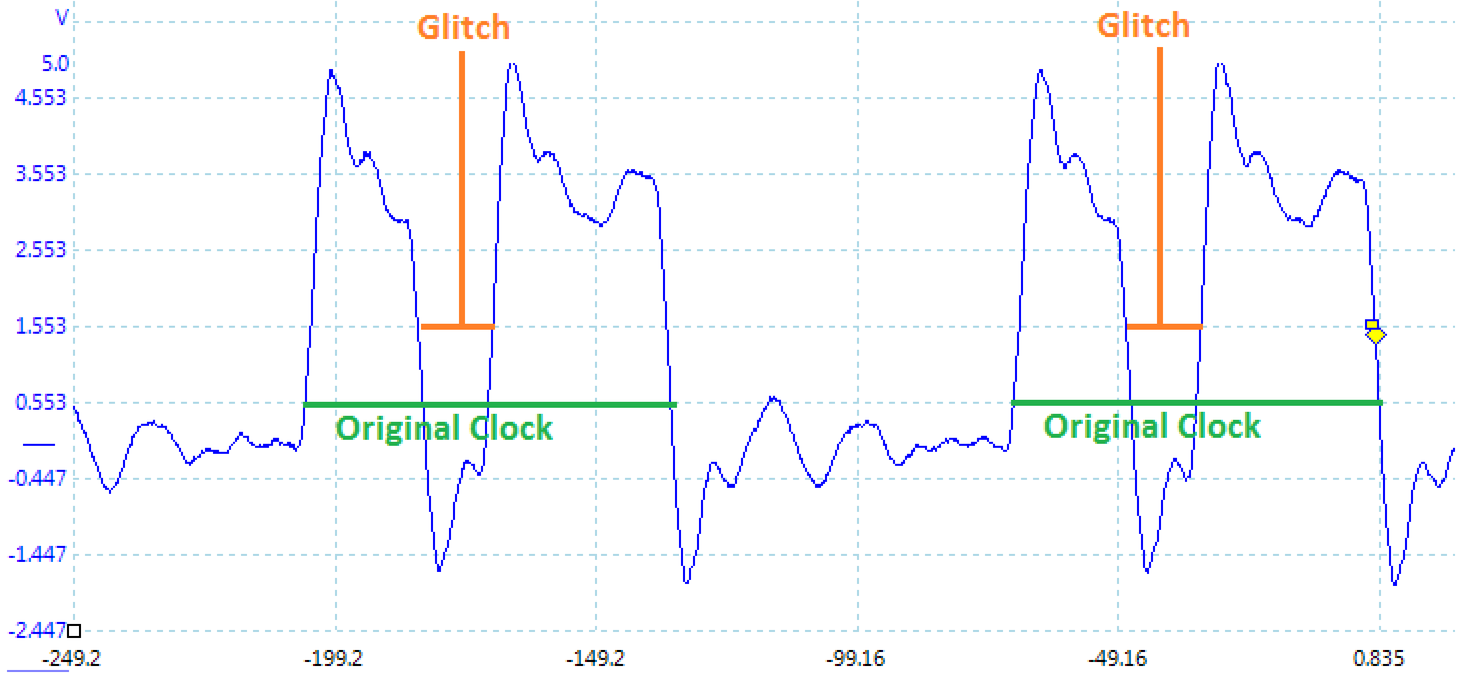
Clock Glitching





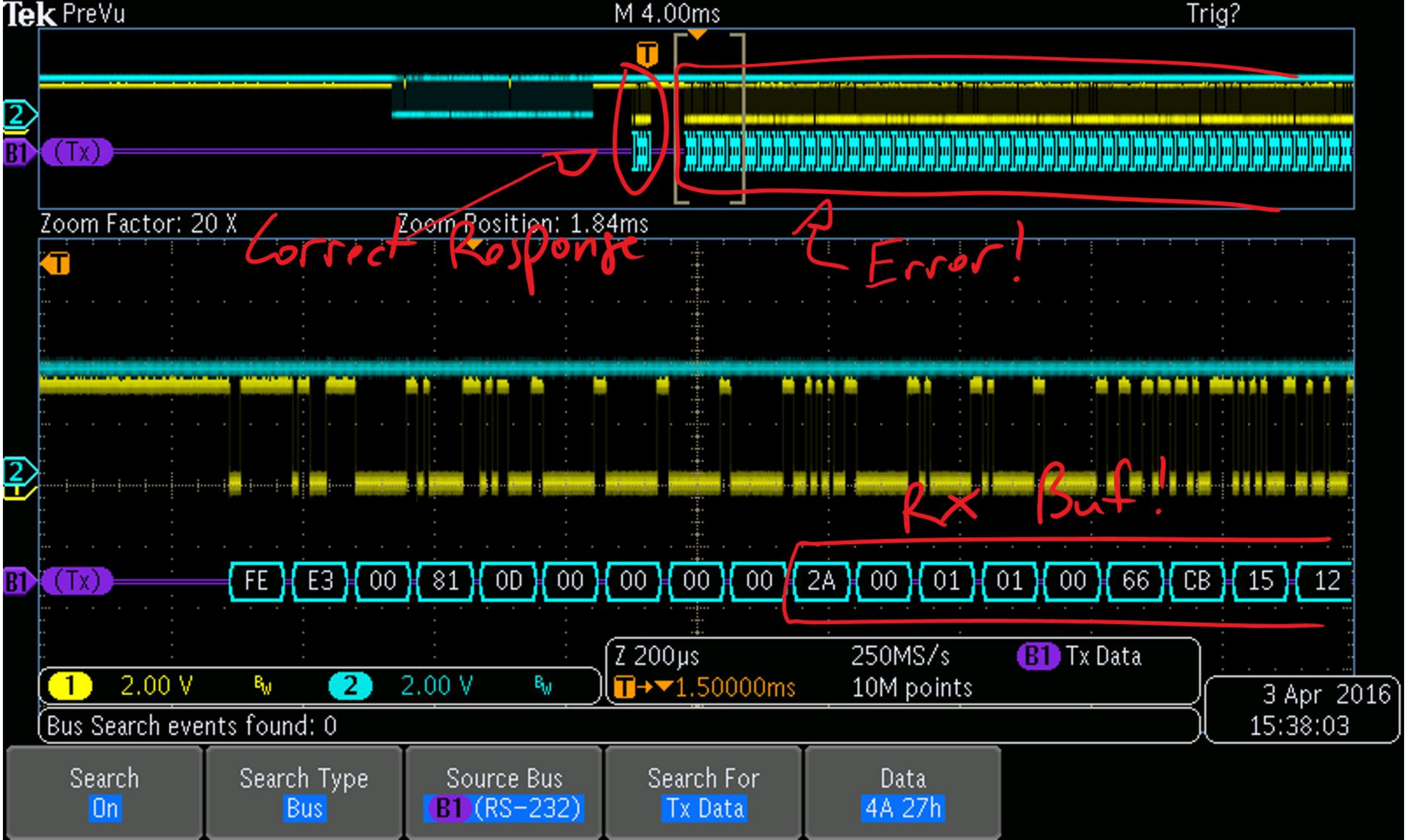
Original Clock

7.37 MHz



Width = 10%
Offset = +15%

Clock XORd with Glitch



Correct Response

Error!

Rx Buf!

Z 200µs
T→▼1.50000ms
250MS/s
10M points
B1 Tx Data

3 Apr 2016
15:38:03

Bus Search events found: 0

Search: On
Search Type: Bus
Source Bus: B1 (RS-232)
Search For: Tx Data
Data: 4A 27h

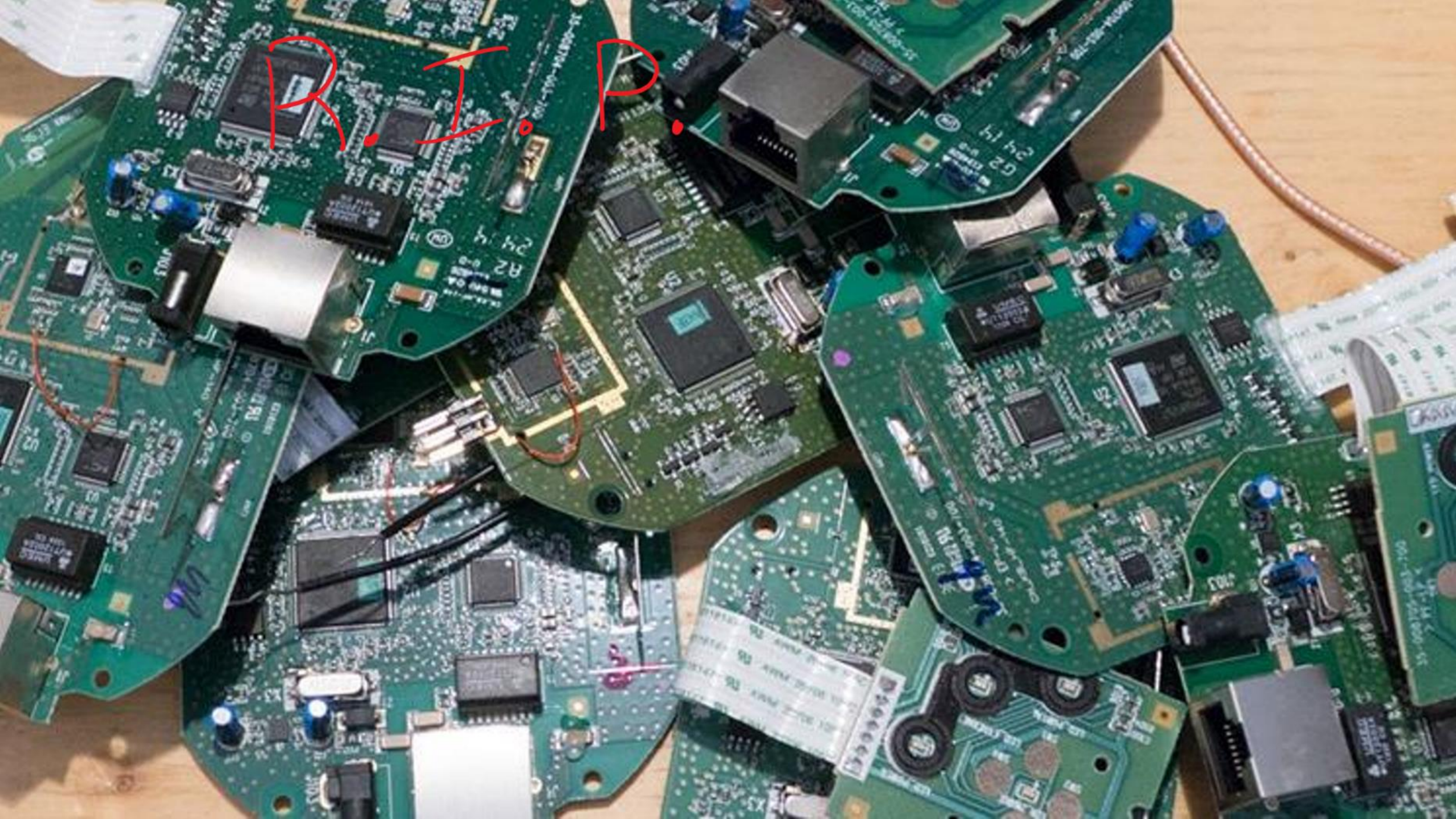
Appears section of SRAM is erased after use.

↳ This is good practice!

↳ May be possible with more glitches.

Glitch Attacks To Firmware

- Appears we can use glitching to dump SRAM.
- Careful timing required to get decrypted data.


R.I.P.

Trying to break the read protect

Trying to break the read protect

- The protect bit is saved as the last bit in memory

Trying to break the read protect

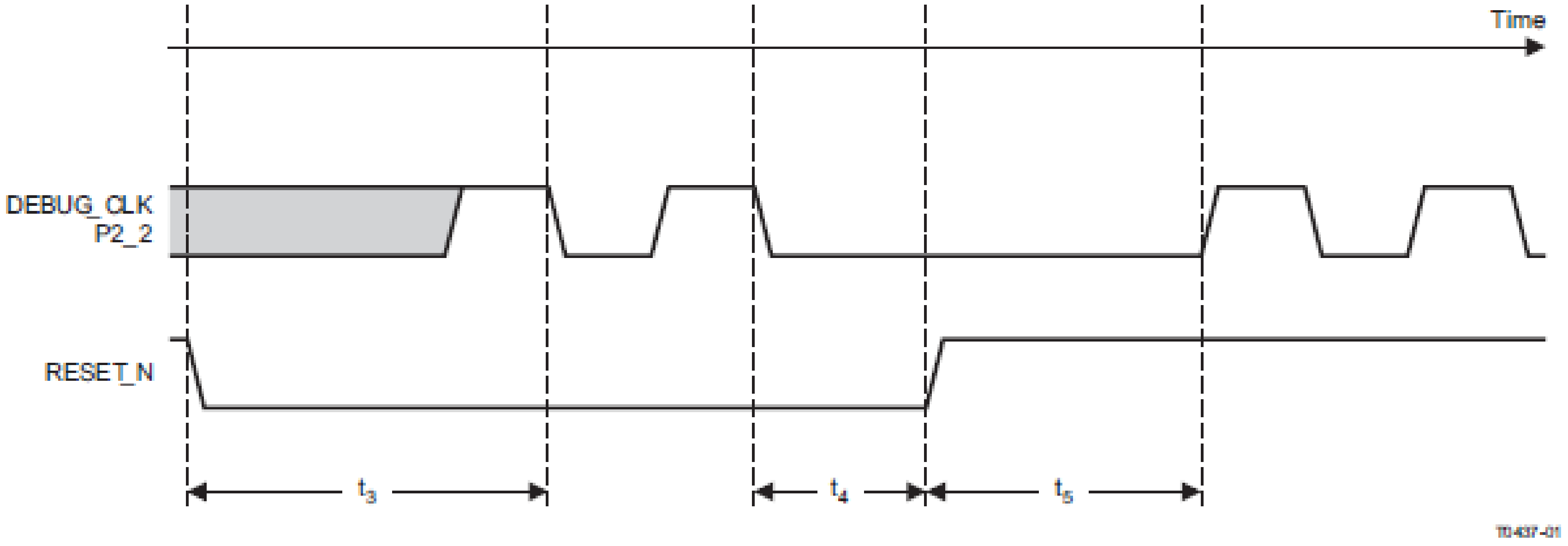
- The protect bit is saved as the last bit in memory
- We don’t care about any other bits around it, so we can corrupt the data around it

Trying to break the read protect

- The protect bit is saved as the last bit in memory
- We don’t care about any other bits around it, so we can corrupt the data around it
- We assume it is read when entering debug

Trying to break the read protect

- The protect bit is saved as the last bit in memory
- We don’t care about any other bits around it, so we can corrupt the data around it
- We assume it is read when entering debug



Glitch attack

Glitch attack

- Lets glitch the clock!

Glitch attack

- Lets glitch the clock!
  - Probably boot on internal clock ☹

Glitch attack

- Lets glitch the clock!
  - Probably boot on internal clock ☹
- Let’s try Voltage glitch!

Glitch attack

- Lets glitch the clock!
  - Probably boot on internal clock ☹
- Let’s try Voltage glitch!
- Need to find the sweet spot - Low enough to corrupt data, high enough to not reset:

Glitch attack

- Lets glitch the clock!
  - Probably boot on internal clock ☹
- Let’s try Voltage glitch!
- Need to find the sweet spot - Low enough to corrupt data, high enough to not reset:
  - External Capacitors

Glitch attack

- Lets glitch the clock!
  - Probably boot on internal clock ☹
- Let’s try Voltage glitch!
- Need to find the sweet spot - Low enough to corrupt data, high enough to not reset:
  - External Capacitors
  - Internal capacity

Glitch attack

- Lets glitch the clock!
  - Probably boot on internal clock ☹
- Let’s try Voltage glitch!
- Need to find the sweet spot - Low enough to corrupt data, high enough to not reset:
  - External Capacitors
  - Internal capacity
  - Brownout detector

Glitch attack

Glitch attack

- Use Arduino PWM output – semi success

Glitch attack

- Use Arduino PWM output – semi success
  - Iterate over offset, frequency and duty cycle

Glitch attack

- Use Arduino PWM output – semi success
  - Iterate over offset, frequency and duty cycle
  - Results

Glitch attack

- Use Arduino PWM output – semi success
  - Iterate over offset, frequency and duty cycle
  - Results
    - Normal debug

Glitch attack

- Use Arduino PWM output – semi success
  - Iterate over offset, frequency and duty cycle
  - Results
    - Normal debug
    - Reset

Glitch attack

- Use Arduino PWM output – semi success
  - Iterate over offset, frequency and duty cycle
  - Results
    - Normal debug
    - Reset
    - Chip erased

Glitch attack

- Use Arduino PWM output – semi success
  - Iterate over offset, frequency and duty cycle
  - Results
    - Normal debug
    - Reset
    - Chip erased
    - A new undocumented state

Glitch attack

- Use Arduino PWM output – semi success
  - Iterate over offset, frequency and duty cycle
  - Results
    - Normal debug
    - Reset
    - Chip erased
    - A new undocumented state
  - Could try fuzzing, or use better glitching source

Second try - Atmel

Getting software updates

- No software update for Atmel based lamps

Getting software updates

- No software update for Atmel based lamps
- So lets impersonate as an older model and version

Getting software updates

- No software update for Atmel based lamps
- So lets impersonate as an older model and version
- Looked for posting on upgrades on the Internet (mainly Reddit)

Getting software updates

- No software update for Atmel based lamps
- So lets impersonate as an older model and version
- Looked for posting on upgrades on the Internet (mainly Reddit)

Known upgrades (From Internet Posts)

66009663 -> 66013452

65003148 -> 66013452 (recorded with type 100)

66010820 -> 66012457 (recorded with type 104) (GU10)

65003148 -> 66012457 (recorded with type 104) (GU10)

65003148 -> 66013452 (recorded with type 103)

Human to machine translation

- We sniff normal communication, version is encoded differently

Human to machine translation

- We sniff normal communication, version is encoded differently
- Record all version we bought

Human to machine translation

- We sniff normal communication, version is encoded differently
- Record all version we bought

Model	number	string	hex	Ar|Ab|Sr|Sb	hex	Led	Image Type in Req
LCT001	66013452	5.23.1.13452					
LCT001	66010820	5.8.1.10820		66 0 2a 44			
LCT001	66013187	5.23.1.13187	5.23.1.0x3383	66 0 51 131	42 0 33 83	Old Hue	0x0104
LWB004 Lux	66012040	5.17.1.12040	5.17.1.0x2f08	66 0 47 8	42 0 2f 08		0x0105
LWB006 HUE WHITE	66015095	5.38.1.15095	5.38.1.0x3AF7	66 0 58 247	42 0 3A F7		
LCT001	66009663	5.8.1.9663 Or 5.23.1.9663	.0x25bf		42 0 25 bf		
LCT001	65003148				41 0 0c 4c		
HML004	66014169						
LCT007	66014919	5.38.1.14919	5.38.1.0x3A47	66 0 58 71	42 0 3A 47		New Controller 2NG

Human to machine translation

- We sniff normal communication, version is encoded differently
- Record all version we bought

Model	number	string	hex	Ar|Ab|Sr|Sb	hex	Led	Image Type in Req
LCT001	66013452	5.23.1.13452					
LCT001	66010820	5.8.1.10820		66 0 2a 44			
LCT001	66013187	5.23.1.13187	5.23.1.0x3383	66 0 51 131	42 0 33 83	Old Hue	0x0104
LWB004 Lux	66012040	5.17.1.12040	5.17.1.0x2f08	66 0 47 8	42 0 2f 08		0x0105
LWB006 HUE WHITE	66015095	5.38.1.15095	5.38.1.0x3AF7	66 0 58 247	42 0 3A F7		
LCT001	66009663	5.8.1.9663 Or 5.23.1.9663	.0x25bf		42 0 25 bf		
LCT001	65003148				41 0 0c 4c		
HML004	66014169						
LCT007	66014919	5.38.1.14919	5.38.1.0x3A47	66 0 58 71	42 0 3A 47		New Controller 2NG

- 66012040 – 66 0 12040 – 0x42 0x00 0x2f08 – 0x42 0x00 0x2f 0x08

Light impersonating

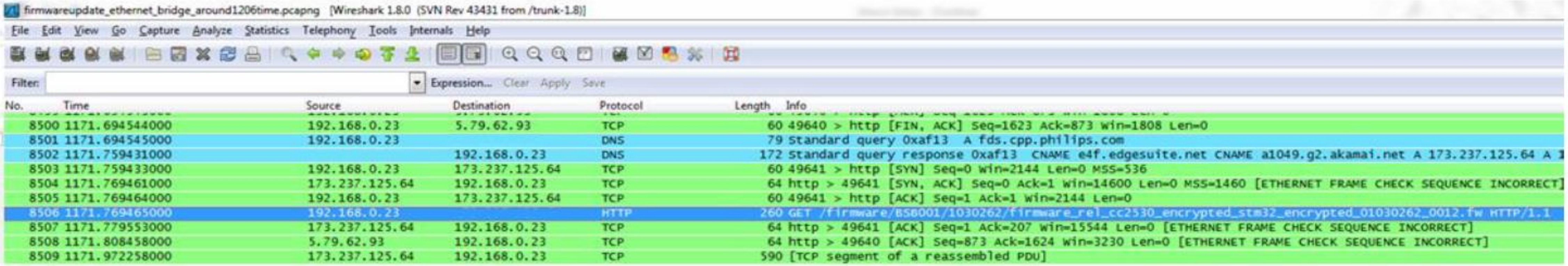
- Write impersonating code, to identify as old models

Light impersonating

- Write impersonating code, to identify as old models
- Sniff OTA updates on Zigbee and on bridge

Light impersonating

- Write impersonating code, to identify as old models
- Sniff OTA updates on Zigbee and on bridge


firmwareupdate_ethernet_bridge_around1206time.pcapng [Wireshark 1.8.0 (SVN Rev 43431 from /trunk-1.8)]
File
Edit
View
Go
Capture
Analyze
Statistics
Telephony
Tools
Internals
Help
Filter:
Expression...
Clear
Apply
Save

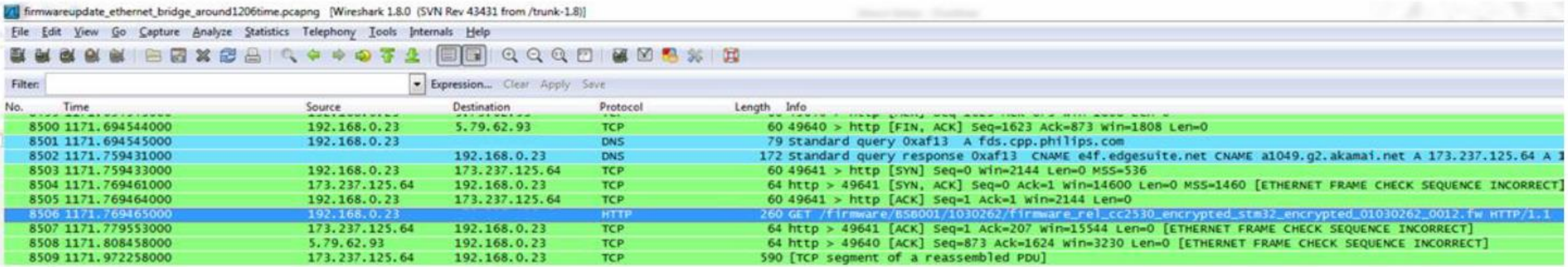
```
No.   Time              Source           Destination      Protocol  Length  Info
8500 1171.694544000    192.168.0.23     5.79.62.93       TCP         60 49640 > http [FIN, ACK] Seq=1623 Ack=873 Win=1808 Len=0
8501 1171.694545000    192.168.0.23                      DNS         79 Standard query 0xaf13  A fds.cpp.philips.com
8502 1171.759431000                     192.168.0.23     DNS        172 Standard query response 0xaf13  CNAME e4f.edgesuite.net CNAME a1049.g2.akamai.net A 173.237.125.64 A 1
8503 1171.759433000    192.168.0.23     173.237.125.64   TCP         60 49641 > http [SYN] Seq=0 Win=2144 Len=0 MSS=536
8504 1171.769461000    173.237.125.64   192.168.0.23     TCP         64 http > 49641 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
8505 1171.769464000    192.168.0.23     173.237.125.64   TCP         60 49641 > http [ACK] Seq=1 Ack=1 Win=2144 Len=0
8506 1171.769465000    192.168.0.23                      HTTP       260 GET /firmware/BSB001/1030262/firmware_rel_cc2530_encrypted_stm32_encrypted_01030262_0012.fw HTTP/1.1
8507 1171.779553000    173.237.125.64   192.168.0.23     TCP         64 http > 49641 [ACK] Seq=1 Ack=207 Win=15544 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
8508 1171.808458000    5.79.62.93       192.168.0.23     TCP         64 http > 49640 [ACK] Seq=873 Ack=1624 Win=3230 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
8509 1171.972258000    173.237.125.64   192.168.0.23     TCP        590 [TCP segment of a reassembled PDU]
```

http://xxx/firmware/HUE0100/66013452/ConnectedLamp-Target_0012_13452_8D.sbl-ota

http://xxx/firmware/BSB001/1030262/firmware_rel_cc2530_encrypted_stm32_encrypted_01030262_0012.fw

Light impersonating

- Write impersonating code, to identify as old models
- Sniff OTA updates on Zigbee and on bridge


firmwareupdate_ethernet_bridge_around1206time.pcapng [Wireshark 1.8.0 (SVN Rev 43431 from /trunk-1.8)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.   Time            Source          Destination     Protocol  Length Info
 8500 1171.694544000  192.168.0.23    5.79.62.93      TCP           60 49640 > http [FIN, ACK] Seq=1623 Ack=873 Win=1808 Len=0
 8501 1171.694545000  192.168.0.23                    DNS           79 Standard query 0xaf13  A fds.cpp.philips.com
 8502 1171.759431000                  192.168.0.23    DNS          172 Standard query response 0xaf13  CNAME e4f.edgesuite.net CNAME a1049.g2.akamai.net A 173.237.125.64 A 1
 8503 1171.759433000  192.168.0.23    173.237.125.64  TCP           60 49641 > http [SYN] Seq=0 Win=2144 Len=0 MSS=536
 8504 1171.769461000  173.237.125.64  192.168.0.23    TCP           64 http > 49641 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
 8505 1171.769464000  192.168.0.23    173.237.125.64  TCP           60 49641 > http [ACK] Seq=1 Ack=1 Win=2144 Len=0
 8506 1171.769465000  192.168.0.23                    HTTP         260 GET /firmware/BSB001/1030262/firmware_rel_cc2530_encrypted_stm32_encrypted_01030262_0012.fw HTTP/1.1
 8507 1171.779553000  173.237.125.64  192.168.0.23    TCP           64 http > 49641 [ACK] Seq=1 Ack=207 Win=15544 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
 8508 1171.808458000  5.79.62.93      192.168.0.23    TCP           64 http > 49640 [ACK] Seq=873 Ack=1624 Win=3230 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
 8509 1171.972258000  173.237.125.64  192.168.0.23    TCP          590 [TCP segment of a reassembled PDU]
```

http://xxx/firmware/HUE0100/66013452/ConnectedLamp-Target_0012_13452_8D.sbl-ota

http://xxx/firmware/BSB001/1030262/firmware_rel_cc2530_encrypted_stm32_encrypted_01030262_0012.fw

- They are encrypted

Start OTA attack

Start OTA attack

- Try to load old firmware to new bulb using OTA protocol

Start OTA attack

- Try to load old firmware to new bulb using OTA protocol
  - Failed on file Type – fix

Start OTA attack

- Try to load old firmware to new bulb using OTA protocol
  - Failed on file Type – fix
  - Failed on file Size – fix

Start OTA attack

- Try to load old firmware to new bulb using OTA protocol
  - Failed on file Type – fix
  - Failed on file Size – fix
- Start OTA – get invalid version msg after first block

Start OTA attack

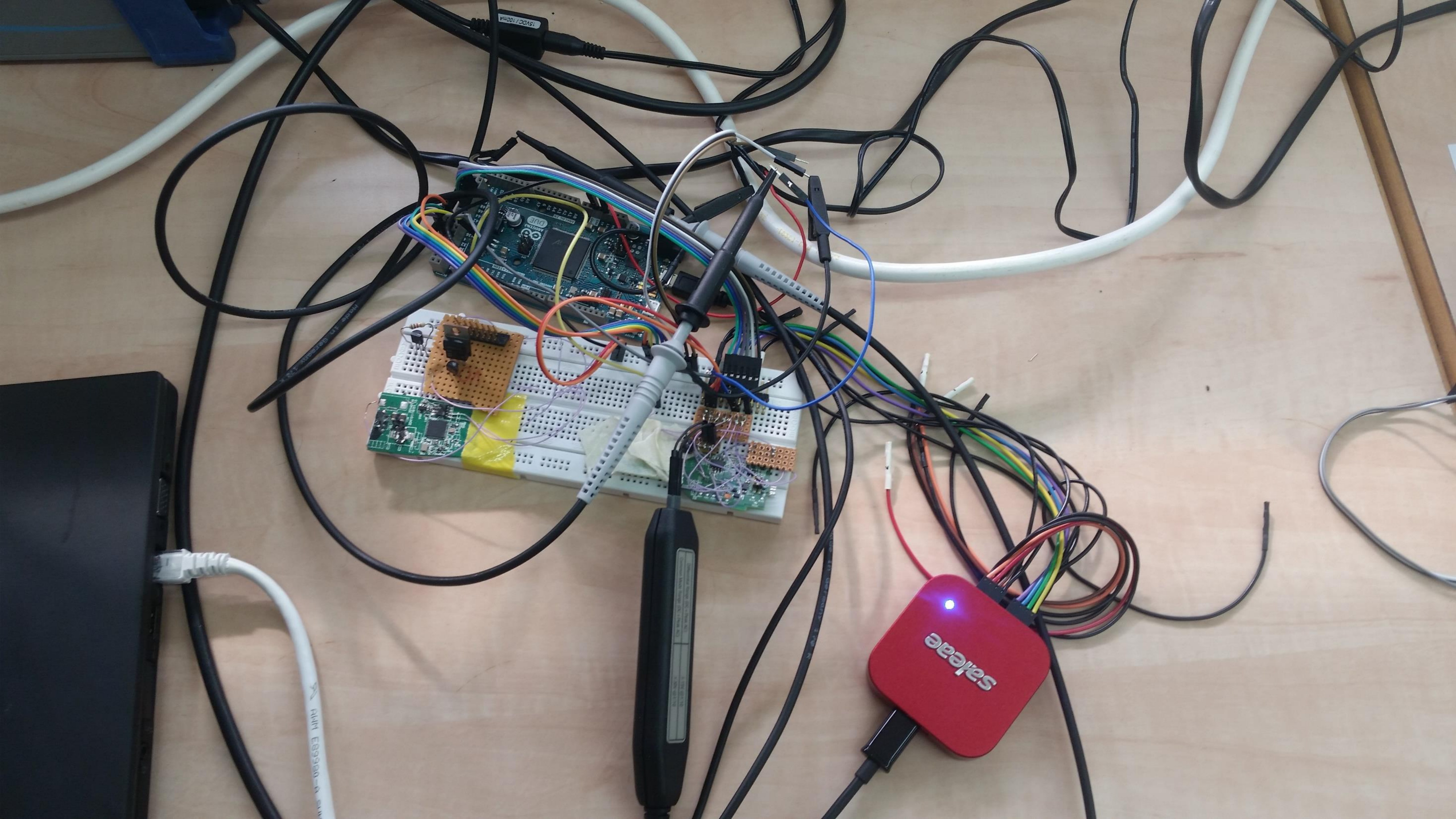
- Try to load old firmware to new bulb using OTA protocol
  - Failed on file Type – fix
  - Failed on file Size – fix
- Start OTA – get invalid version msg after first block
  - Change block size to one

Start OTA attack

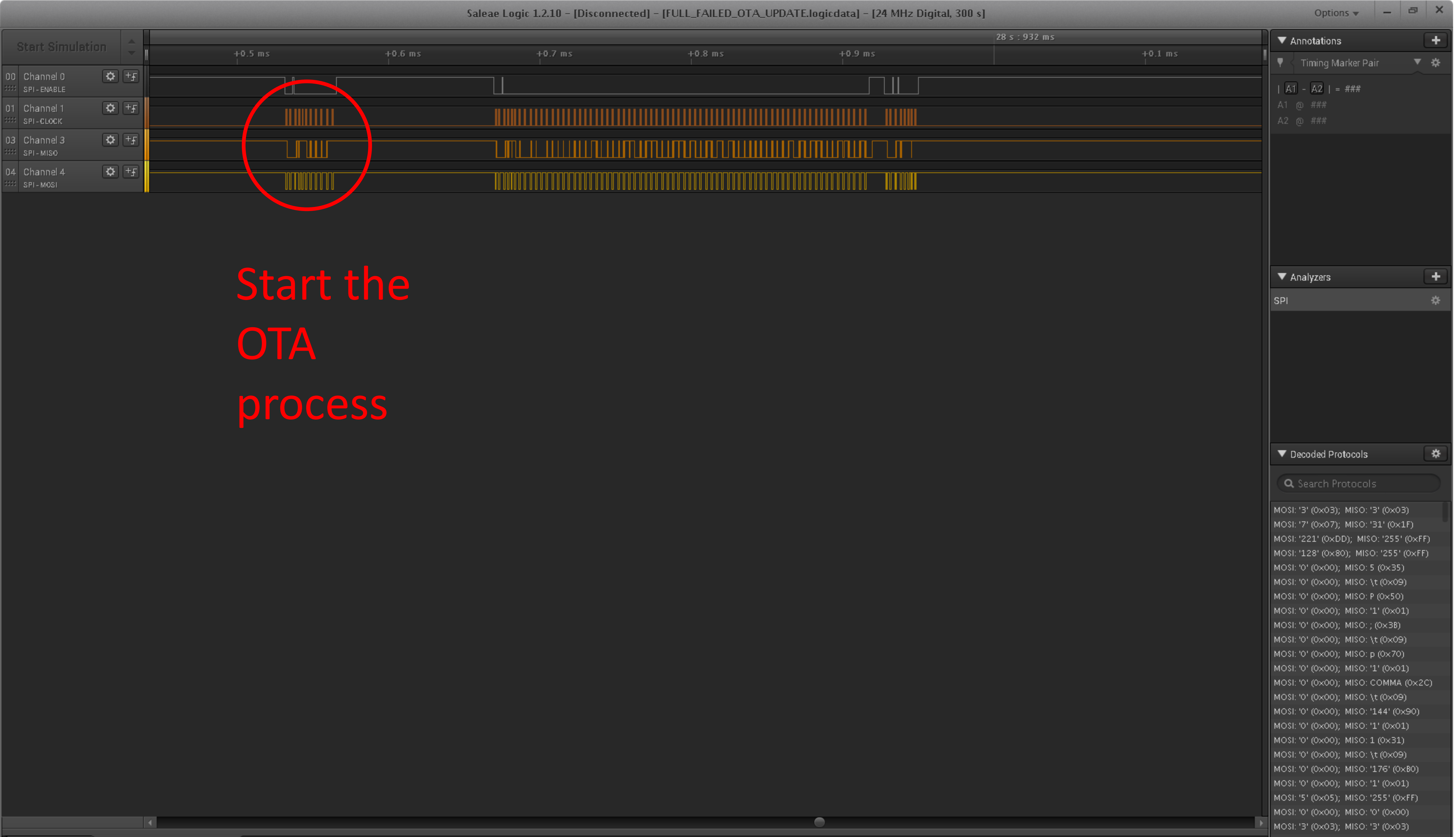
- Try to load old firmware to new bulb using OTA protocol
  - Failed on file Type – fix
  - Failed on file Size – fix
- Start OTA – get invalid version msg after first block
  - Change block size to one
  - Failed after 56 bytes – Zigbee OTA header size

Start OTA attack

- Try to load old firmware to new bulb using OTA protocol
  - Failed on file Type – fix
  - Failed on file Size – fix
- Start OTA – get invalid version msg after first block
  - Change block size to one
  - Failed after 56 bytes – Zigbee OTA header size
  - Fix type and size in header – OTA started and failed



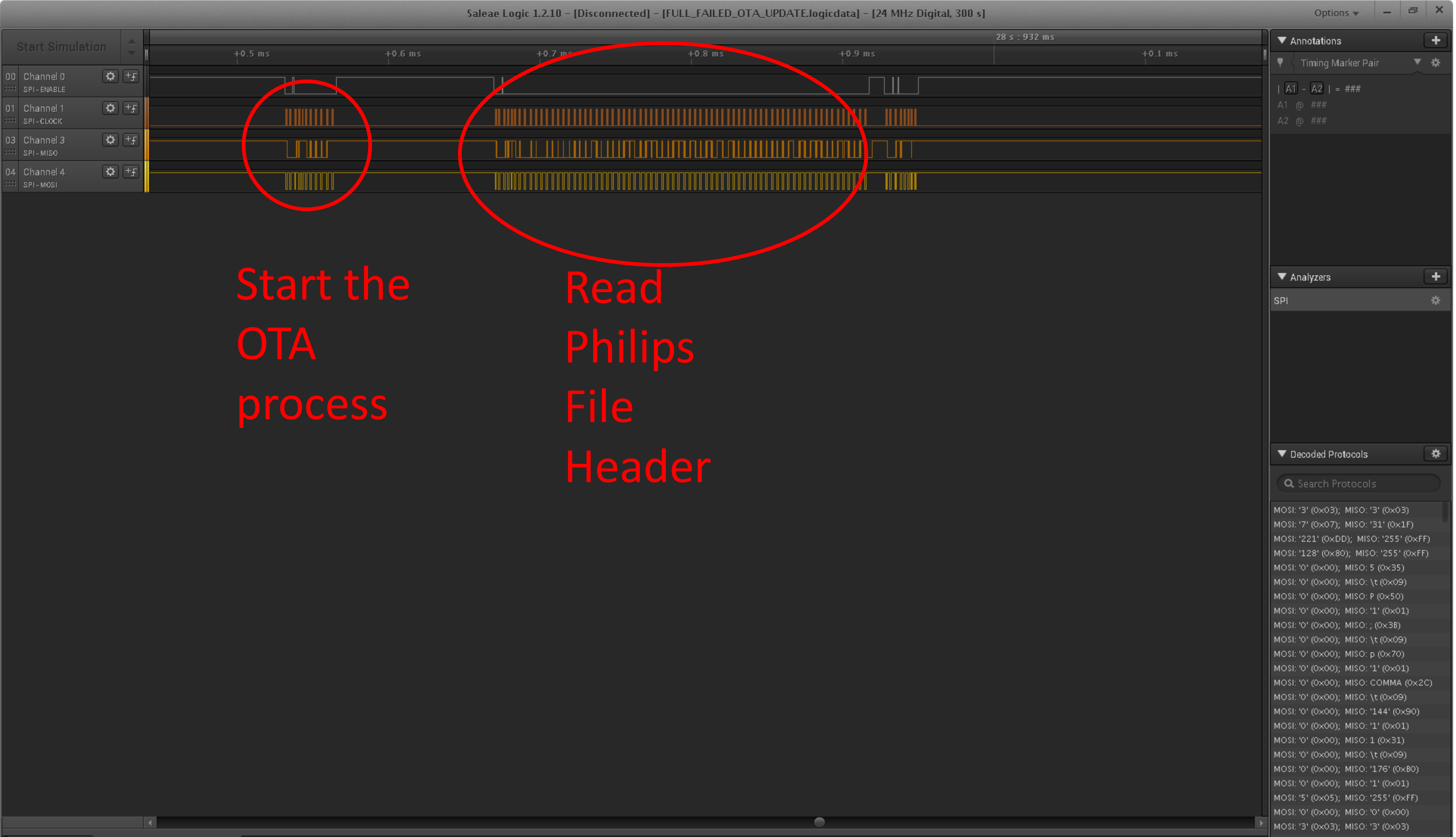
saleae



Saleae Logic 1.2.10 - [Disconnected] - [FULL_FAILED_OTA_UPDATE.logicdata] - [24 MHz Digital, 300 s]

Start the OTA process

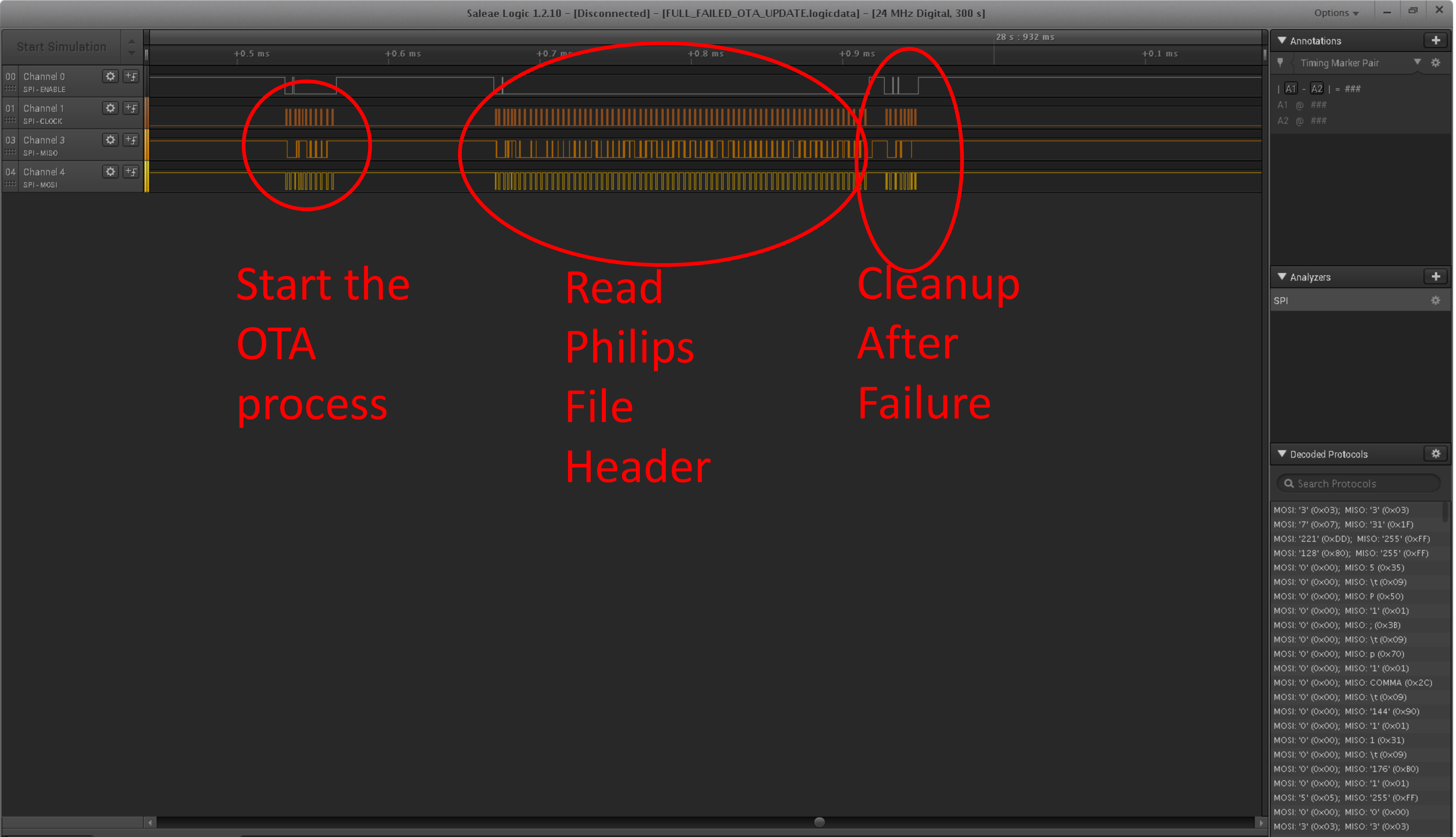
```
MOSI: '3' (0x03);  MISO: '3' (0x03)
MOSI: '7' (0x07);  MISO: '31' (0x1F)
MOSI: '221' (0xDD);  MISO: '255' (0xFF)
MOSI: '128' (0x80);  MISO: '255' (0xFF)
MOSI: '0' (0x00);  MISO: 5 (0x35)
MOSI: '0' (0x00);  MISO: \t (0x09)
MOSI: '0' (0x00);  MISO: P (0x50)
MOSI: '0' (0x00);  MISO: '1' (0x01)
MOSI: '0' (0x00);  MISO: ; (0x3B)
MOSI: '0' (0x00);  MISO: \t (0x09)
MOSI: '0' (0x00);  MISO: p (0x70)
MOSI: '0' (0x00);  MISO: '1' (0x01)
MOSI: '0' (0x00);  MISO: COMMA (0x2C)
MOSI: '0' (0x00);  MISO: \t (0x09)
MOSI: '0' (0x00);  MISO: '144' (0x90)
MOSI: '0' (0x00);  MISO: '1' (0x01)
MOSI: '0' (0x00);  MISO: 1 (0x31)
MOSI: '0' (0x00);  MISO: \t (0x09)
MOSI: '0' (0x00);  MISO: '176' (0xB0)
MOSI: '0' (0x00);  MISO: '1' (0x01)
MOSI: '5' (0x05);  MISO: '255' (0xFF)
MOSI: '0' (0x00);  MISO: '0' (0x00)
MOSI: '3' (0x03);  MISO: '3' (0x03)
```



Start the OTA process

Read Philips File Header

```
MOSI: '3' (0x03);  MISO: '3' (0x03)
MOSI: '7' (0x07);  MISO: '31' (0x1F)
MOSI: '221' (0xDD);  MISO: '255' (0xFF)
MOSI: '128' (0x80);  MISO: '255' (0xFF)
MOSI: '0' (0x00);  MISO: 5 (0x35)
MOSI: '0' (0x00);  MISO: \t (0x09)
MOSI: '0' (0x00);  MISO: P (0x50)
MOSI: '0' (0x00);  MISO: '1' (0x01)
MOSI: '0' (0x00);  MISO: ; (0x3B)
MOSI: '0' (0x00);  MISO: \t (0x09)
MOSI: '0' (0x00);  MISO: p (0x70)
MOSI: '0' (0x00);  MISO: '1' (0x01)
MOSI: '0' (0x00);  MISO: COMMA (0x2C)
MOSI: '0' (0x00);  MISO: \t (0x09)
MOSI: '0' (0x00);  MISO: '144' (0x90)
MOSI: '0' (0x00);  MISO: '1' (0x01)
MOSI: '0' (0x00);  MISO: 1 (0x31)
MOSI: '0' (0x00);  MISO: \t (0x09)
MOSI: '0' (0x00);  MISO: '176' (0xB0)
MOSI: '0' (0x00);  MISO: '1' (0x01)
MOSI: '5' (0x05);  MISO: '255' (0xFF)
MOSI: '0' (0x00);  MISO: '0' (0x00)
MOSI: '3' (0x03);  MISO: '3' (0x03)
```



Start the OTA process

Read Philips File Header

Cleanup After Failure

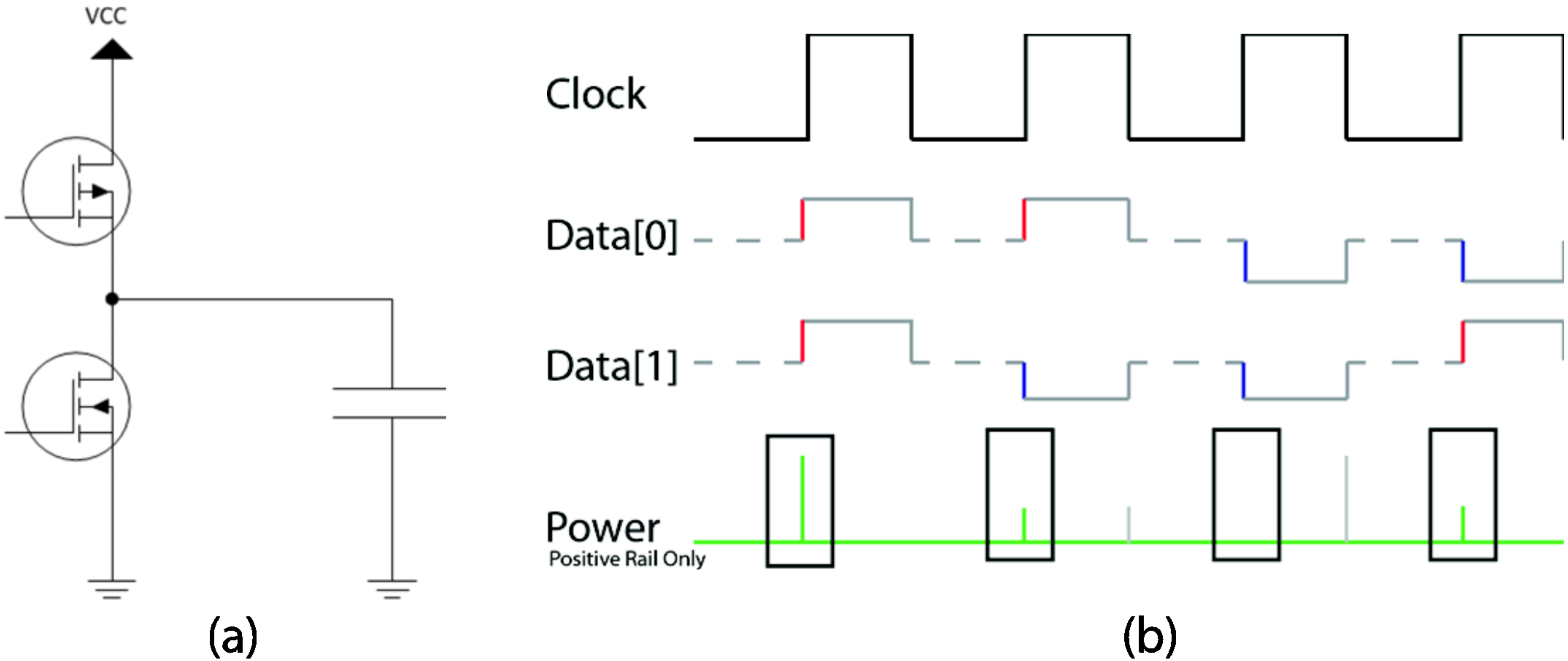
```
MOSI: '3' (0x03);  MISO: '3' (0x03)
MOSI: '7' (0x07);  MISO: '31' (0x1F)
MOSI: '221' (0xDD);  MISO: '255' (0xFF)
MOSI: '128' (0x80);  MISO: '255' (0xFF)
MOSI: '0' (0x00);  MISO: 5 (0x35)
MOSI: '0' (0x00);  MISO: \t (0x09)
MOSI: '0' (0x00);  MISO: P (0x50)
MOSI: '0' (0x00);  MISO: '1' (0x01)
MOSI: '0' (0x00);  MISO: ; (0x3B)
MOSI: '0' (0x00);  MISO: \t (0x09)
MOSI: '0' (0x00);  MISO: p (0x70)
MOSI: '0' (0x00);  MISO: '1' (0x01)
MOSI: '0' (0x00);  MISO: COMMA (0x2C)
MOSI: '0' (0x00);  MISO: \t (0x09)
MOSI: '0' (0x00);  MISO: '144' (0x90)
MOSI: '0' (0x00);  MISO: '1' (0x01)
MOSI: '0' (0x00);  MISO: 1 (0x31)
MOSI: '0' (0x00);  MISO: \t (0x09)
MOSI: '0' (0x00);  MISO: '176' (0xB0)
MOSI: '0' (0x00);  MISO: '1' (0x01)
MOSI: '5' (0x05);  MISO: '255' (0xFF)
MOSI: '0' (0x00);  MISO: '0' (0x00)
MOSI: '3' (0x03);  MISO: '3' (0x03)
```

Downloaded firmwares

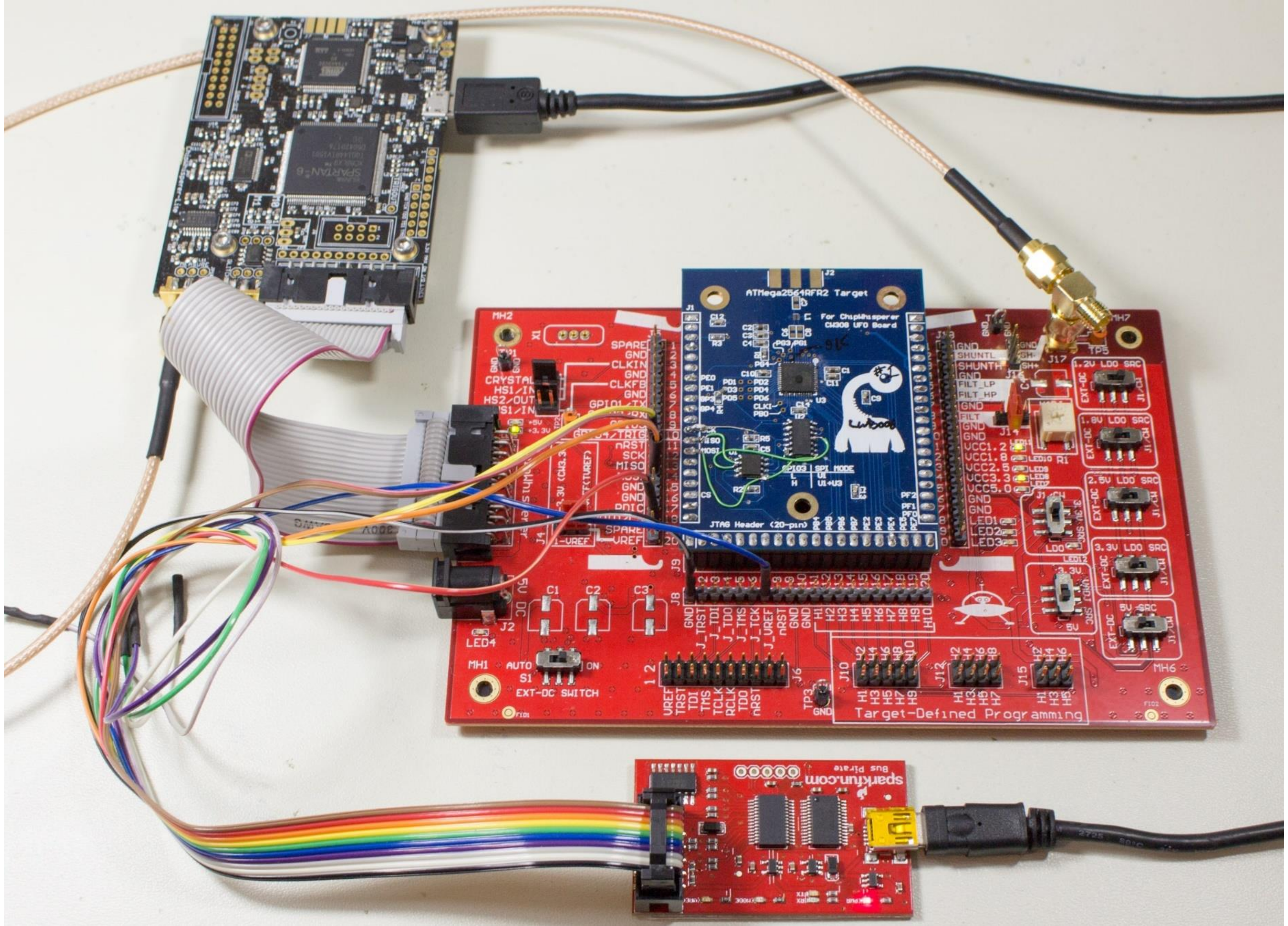
```
GU10=           '2A000100006652141002173039O3EF402E370B25ECC04765CBE11E0E74F7A114EE6B58B52FF30D83681267714C7A
Connected=      '2A000100006652141002173039O3EF402E370B25ECC04765CBE11E0E74F7A114EE6B58B52FF30D83681267714C7A
ConnectedFixed= '2A0001040066521410021730390002002E370B25ECC04765CBE11E0E74F7A114EE6B58B52FF30D83681267714C7A
LivingColors=   '2A0001030069611411031638O703E88011B6996E648CE50CF315CBC2A810C5D26301FD1E5E1E201005823C9AEFAC
fw01016441STM='2A00010201F43E1408281633310618809DB509B3F6E9326D6F8FD2089ECB375D47A6654262B77352C33AAAFD2DB492
fw01018228STM='2A00010201F426141015115924O5EFC0EC546C95824A01524E08D5B3D8CCDAA293C7BC8ECC28087059D6D621CFD801
fw01024156STM='2A00010201F4D51504041310560608 40A84D129BFF0172734E64CD06CC0D0D37507B920B5B7FF6957584CD077111BD
fw01029624STM='2A00010201F44D1511172220440 6A3402EECED20A08438712C2BEF1C815DC534819CB82B3067AAA555E575DF9203B3
fw01029624='2A00010100668C15111722203803 7E8013CE617BE6A3732061E15FDEDC6B0BBF5F165BF1238F173894AF1AFE3DB8A0274
fw01024156='2A0001010066A5150404131053036EC025C053D8B1D93161F218DDE77DF30570EA03C753D16EA8A7DEA13F7F82370F78E
fw01030262='2A0001010066CB151216151233037E802D4A27D63C496DB253809EB7CC57E195A31A1A8054E1012883DA24B57E4BAD453
fw01018228='2A0001010066F5141015115920036BC0290FE89BA8EE70D3C0AF5324306D168C8BA71810EFFD738723B41E12B252C2A2D
fw01016441='2A0001010066FD140828163329036BC00A2CDADABFD5C4DBCBE11EBE0066012F4667D2327D2915DE9F852559 9793F2065

fw01016441STM='2A00010201F43E1408281633310618809DB509B3F6E9326D6F8FD2089ECB375D47A6654262B77352C33AAAFD2DB492
fw01016441='2A0001010066FD140828163329036BC00A2CDADABFD5C4DBCBE11EBE0066012F4667D2327D2915DE9F8525599793F2065
fw01018228STM='2A00010201F426141015115924O5EFC0EC546C95824A01524E08D5B3D8CCDAA293C7BC8ECC28087059D6D621CFD801
fw01018228='2A0001010066F5141015115920036BC0290FE89BA8EE70D3C0AF5324306D168C8BA71810EFFD738723B41E12B252C2A2D
fw01024156STM='2A00010201F4D5150404131056060840A84D129BFF0172734E64CD06CC0D0D37507B920B5B7FF6957584CD077111BD
fw01024156='2A0001010066A5150404131053036EC025C053D8B1D93161F218DDE77DF30570EA03C753D16EA8A7DEA13F7F82370F78E
fw01029624STM='2A00010201F44D15111722204406A3402EECED20A08438712C2BEF1C815DC534819CB82B3067AAA555E575DF9203B3
fw01029624='2A00010100668C151117222038037E8013CE617BE6A3732061E15FDEDC6B0BBF5F165BF1238F173894AF1AFE3DB8A0274

fw01030262='2A0001010066CB151216151233037E802D4A27D63C496DB253809EB7CC57E195A31A1A8054E1012883DA24B57E4BAD453
```

```
GU10=           '2A00 0100 00 6652 141002 173039 03EF40
LivingColors=   '2A00 0103 00 6961 141103 163807 03E880
fw01016441STM=  '2A00 0102 01 F43E 140828 163331 061880
fw01018228STM=  '2A00 0102 01 F426 141015 115924 05EFC0
fw01024156STM=  '2A00 0102 01 F4D5 150404 131056 060840
```

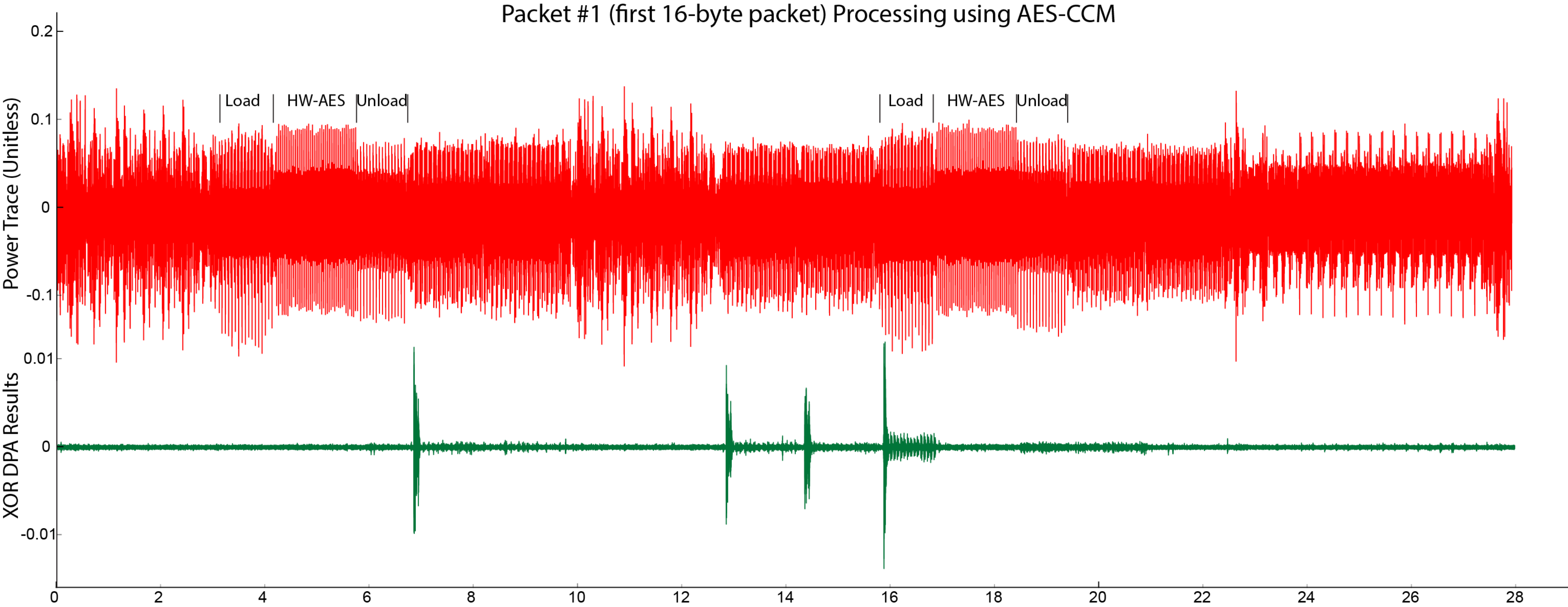
Correlation power analysis



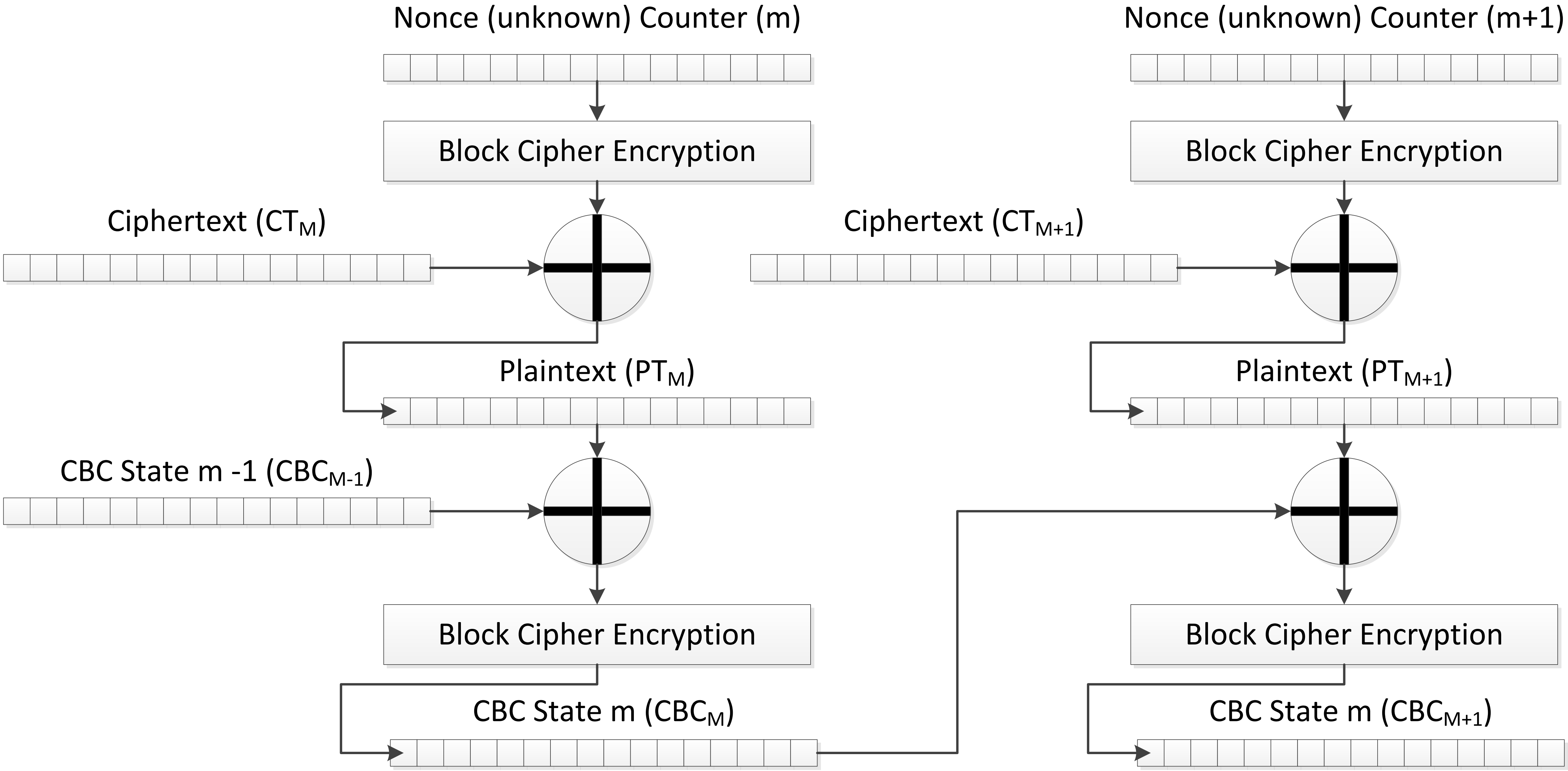
Power Analysis Example Setup



CPA for RE


Packet #1 (first 16-byte packet) Processing using AES-CCM
Power Trace (Unitless)
XOR DPA Results
0.2
0.1
0
-0.1
0.01
0
-0.01
Load
HW-AES
Unload
Load
HW-AES
Unload
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28

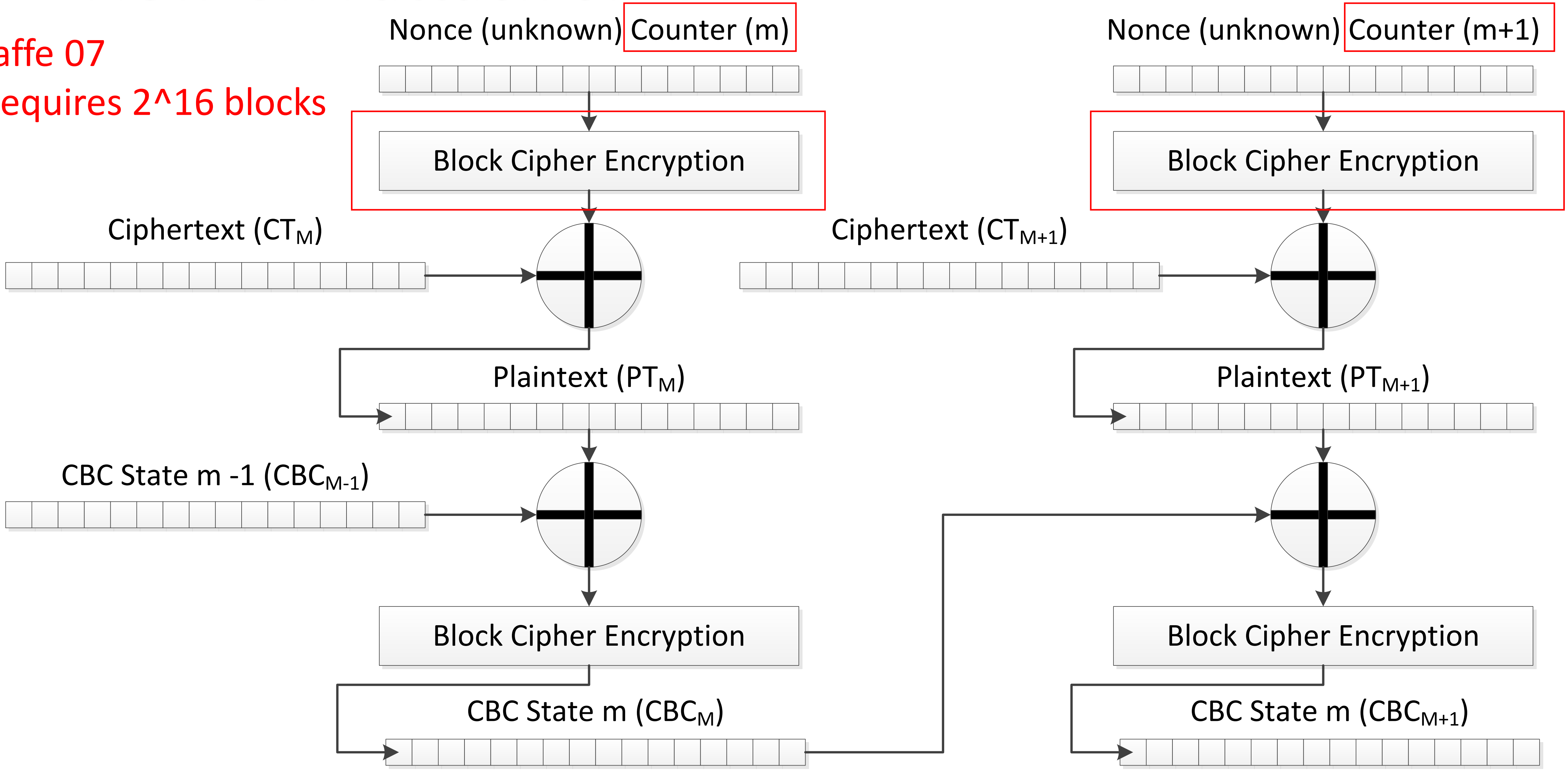
New CPA attack on CCM



New CPA attack on CCM

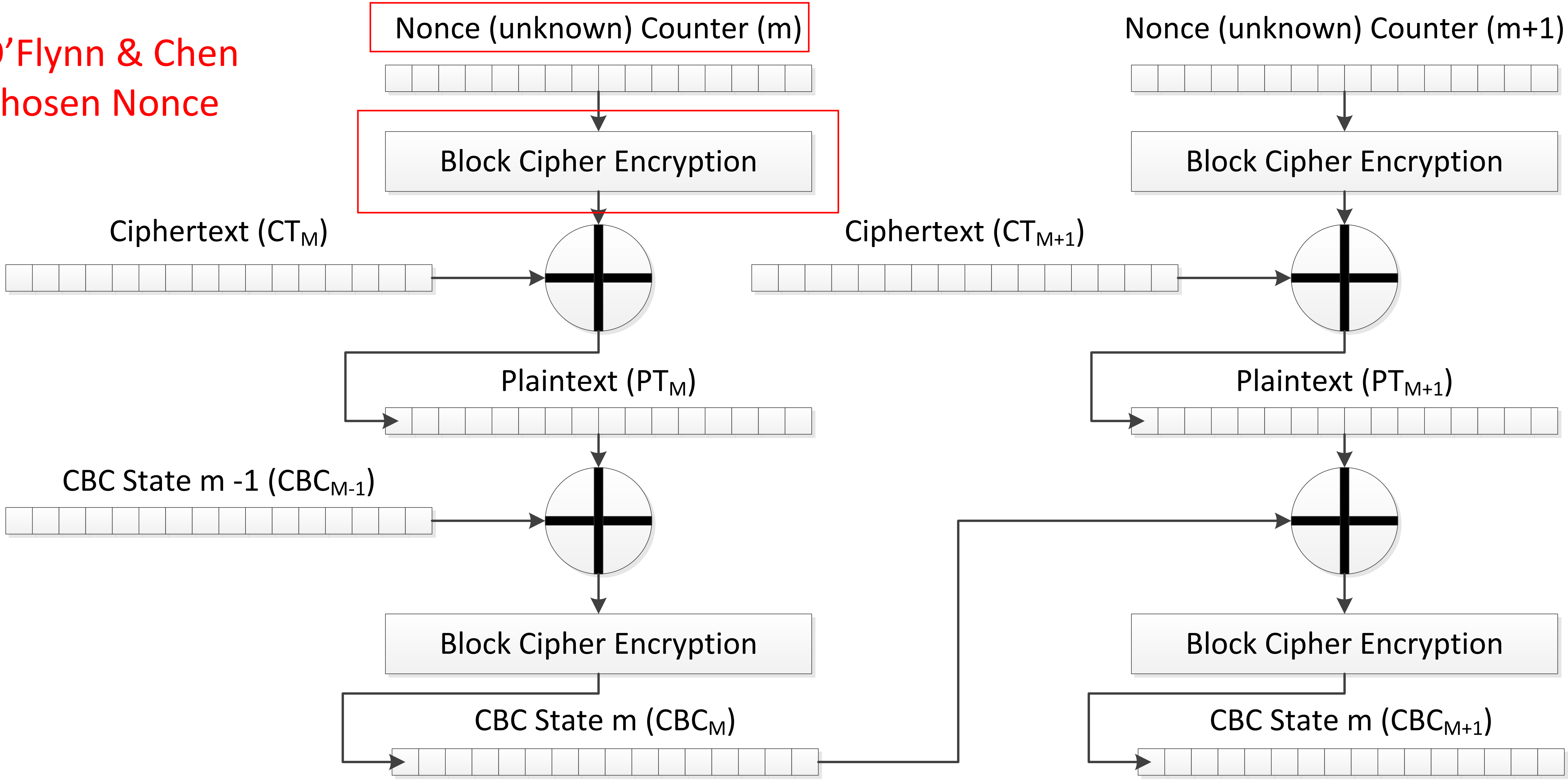
Jaffe 07

Requires 2^16 blocks

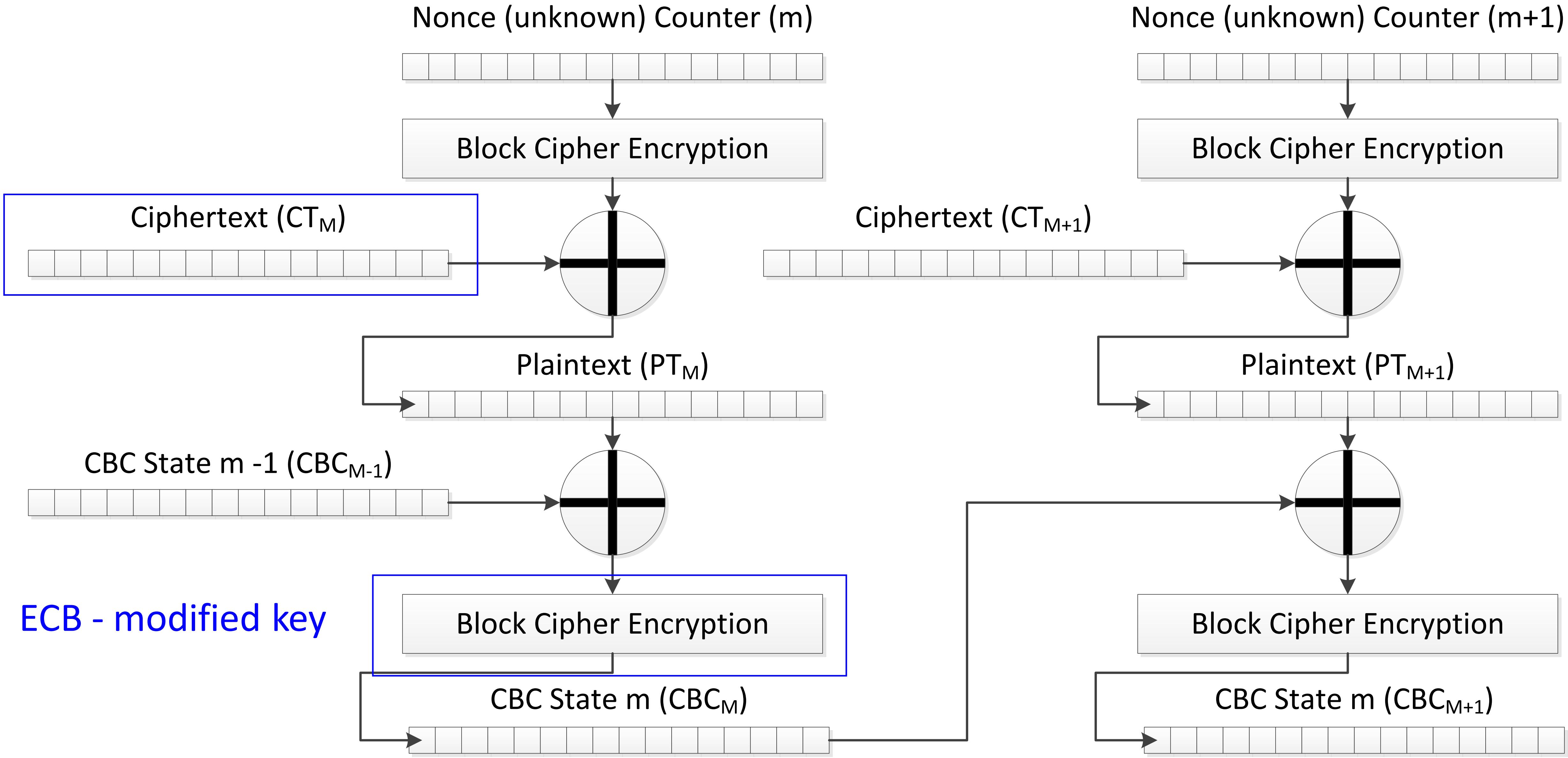


New CPA attack on CCM

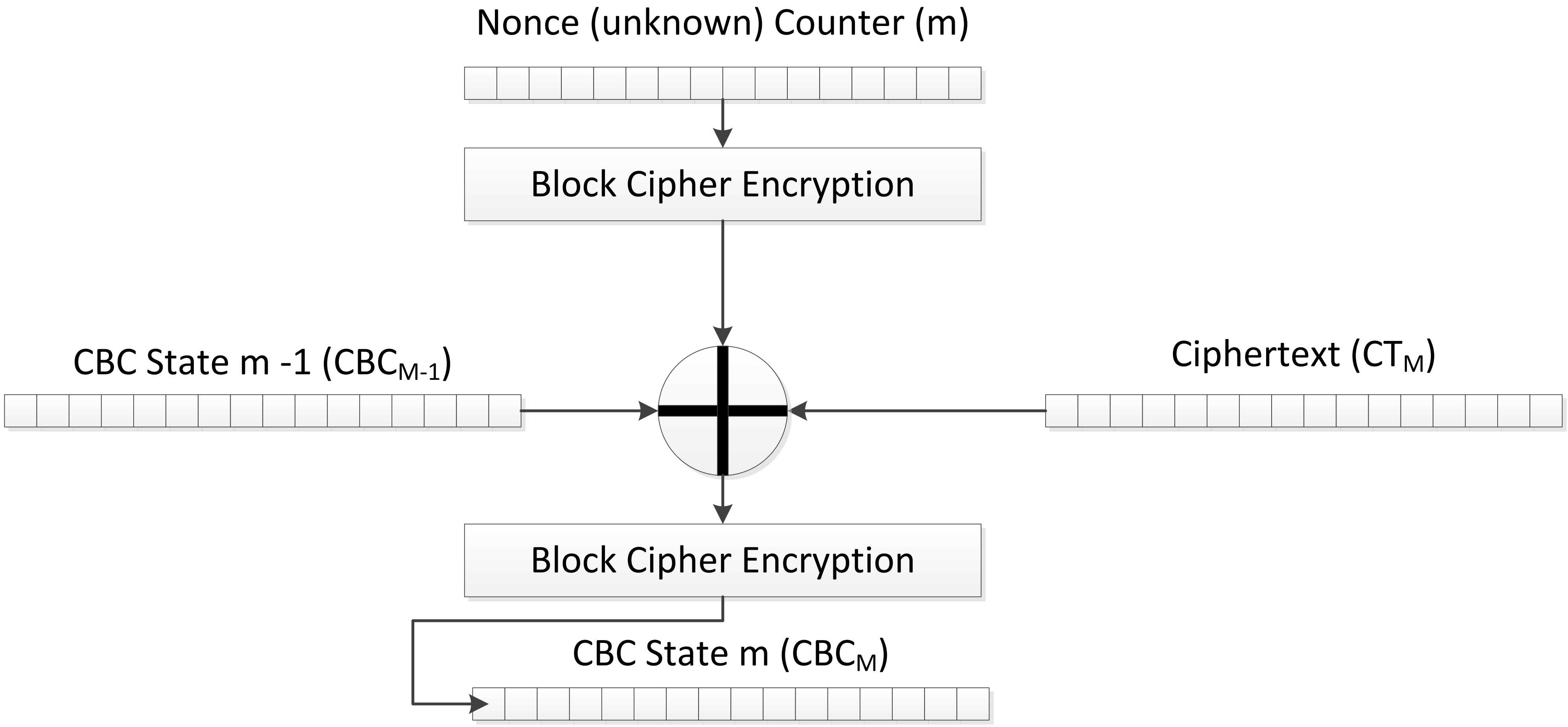
O'Flynn & Chen
Chosen Nonce



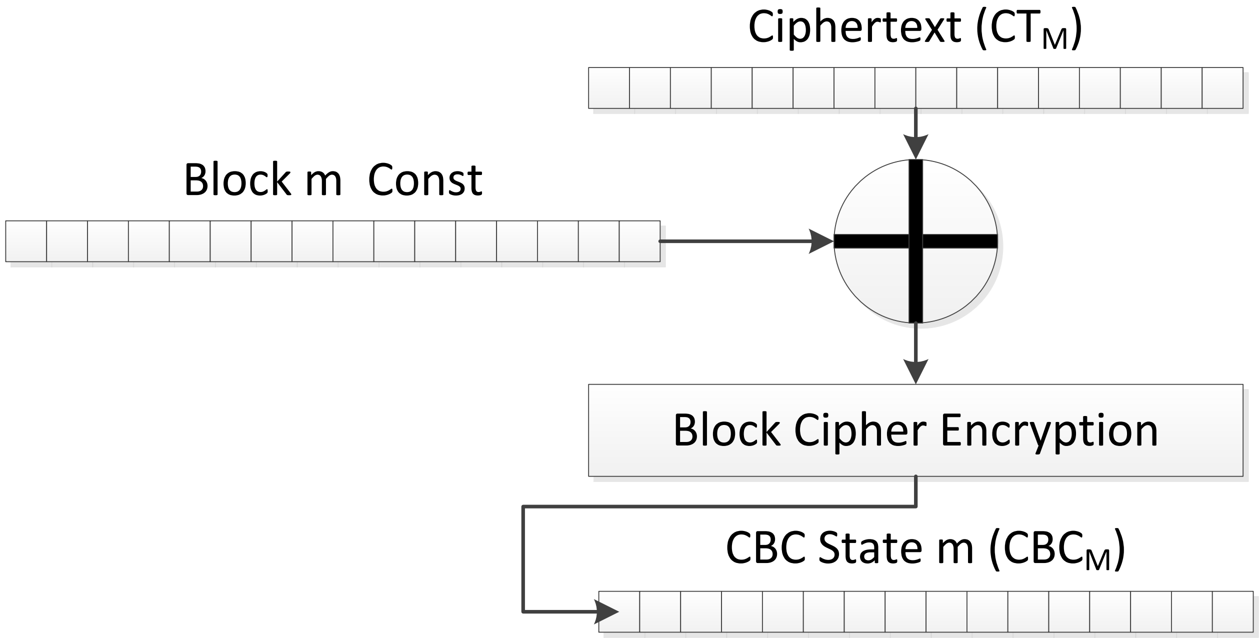
New CPA attack on CCM



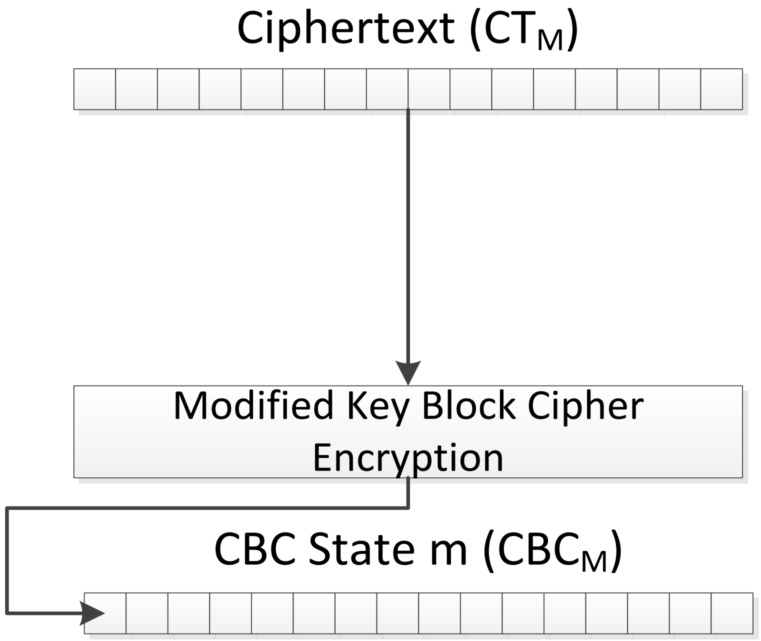
New CPA attack on CCM



New CPA attack on CCM

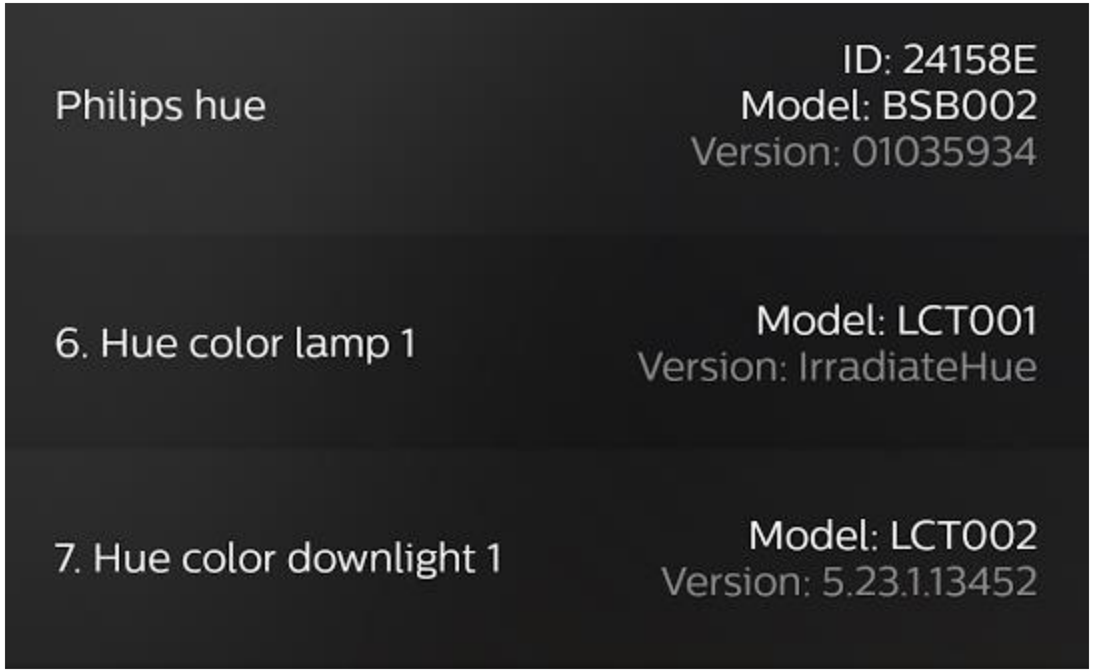


New CPA attack on CCM



```
[Log,Info,LOOK AT ME. ,I'm the captain now........]
[Log,Info,LOOK AT ME. ,DeviceId: Bulb_A19_v1]
[Log,Info,N_Security,LIB4.5.70]
[Log,Info,N_Security,KeyBitMask,0x0012]
[Log,Info,S_OTA,Bootloader: Upgrade succeeded.]
[Log,Info,ConnectedLamp,errs=0,lastErr=NULL@0]
[Log,Info,ConnectedLamp,Platform version 0.43.0,package_Z_Stack 11155,built by LouvreZLL]
[Log,Info,ConnectedLamp,Product version InfectedLamp-TI 0.0.1, broken by Eyal & Colin   ]
[Log,Info,ConnectedLamp,PowerGlitchCount=0]
[Log,Info,A_Commissioning,Factory New at Ch: 11]
[TH,Ready,0]
[Log,Info,TH,ISTACK free: 82]
[Log,Info,TH,XSTACK free: 664]
[Log,Info,S_ThermalShutdown,Shutdown]
[Log,Info,S_XNv,CompactSector,s=4]
[Log,Info,OSAL,Task took too long: id=10, elapsed=1042848]
[Log,Info,TH,ISTACK free: 76]
[Log,Info,TH,ISTACK free: 75]
```

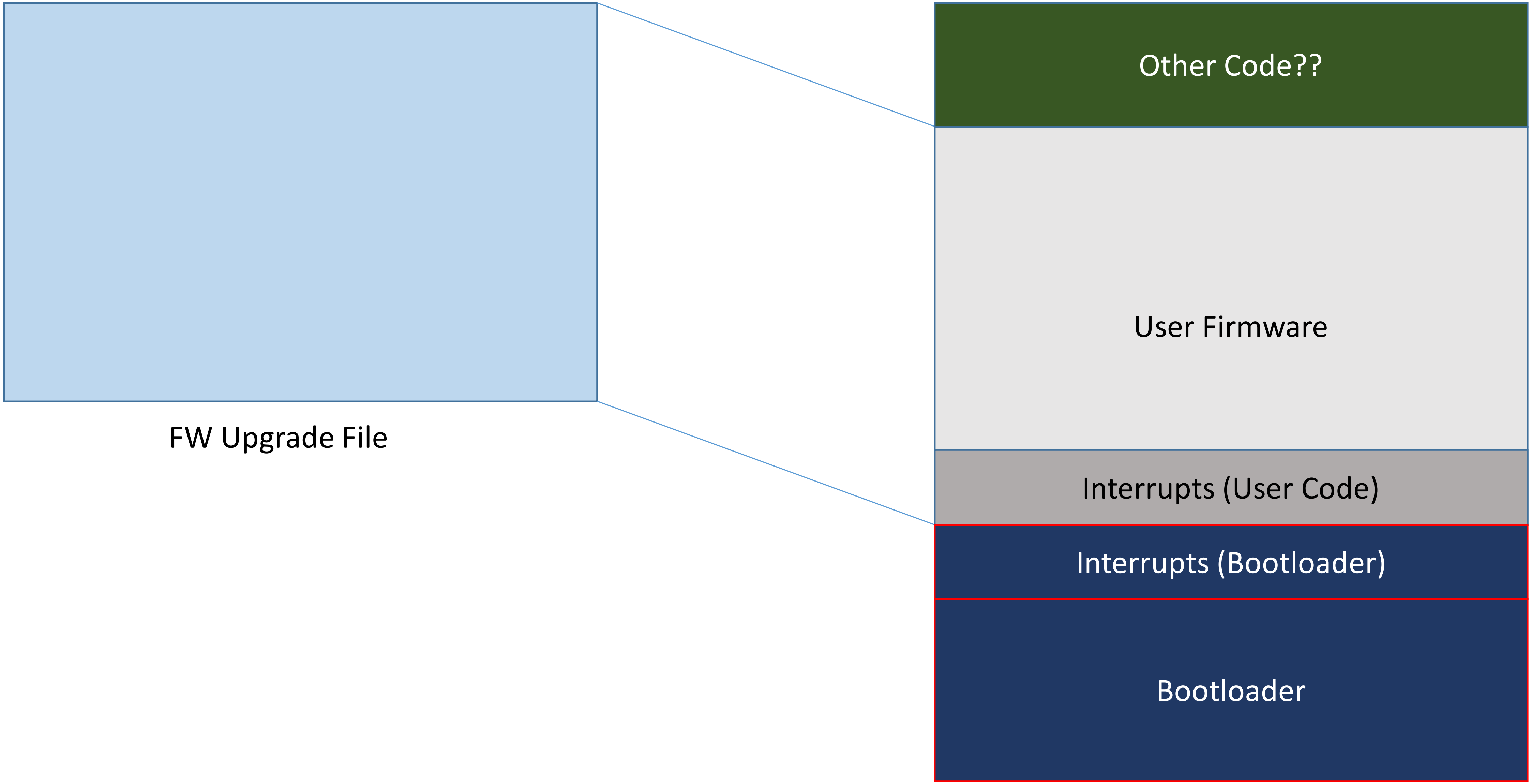
Reflashing Even Older TI-Based Bulbs (initial work)



Philips hue	ID: 24158E Model: BSB002 Version: 01035934
6. Hue color lamp 1	Model: LCT001 Version: IrradiateHue
7. Hue color downlight 1	Model: LCT002 Version: 5.23.1.13452

Key are not enough

- The bootloader is not part of the update code
  - Without it we don’t know the address space, interrupts, etc.
- So we write a dumper code
- Dumper code is patched into binary near the expected start point
- Code can’t use stack & have only relative calls



Other Code??

User Firmware

Interrupts (User Code)

Interrupts (Bootloader)

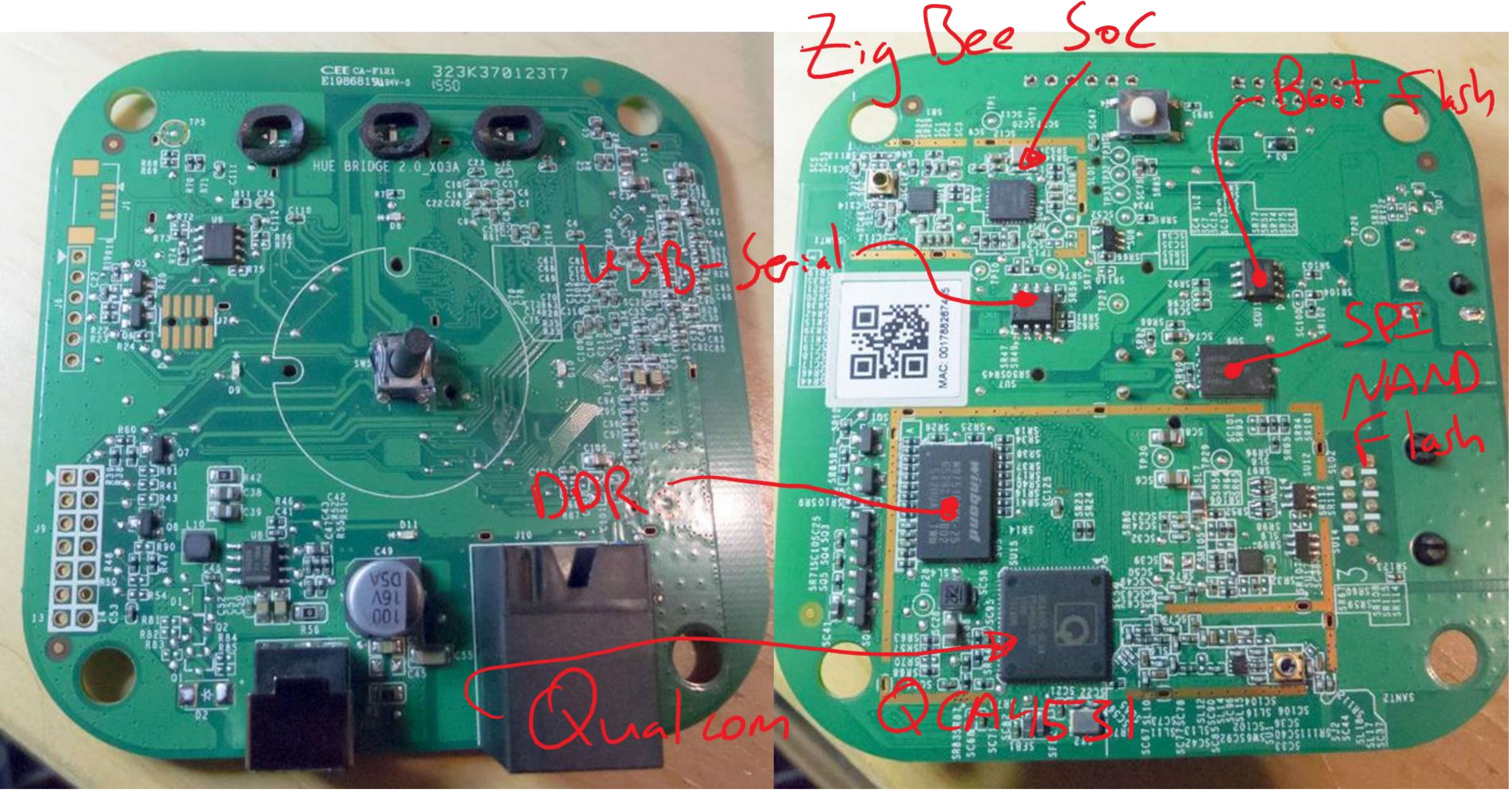
Bootloader

FW Upgrade File

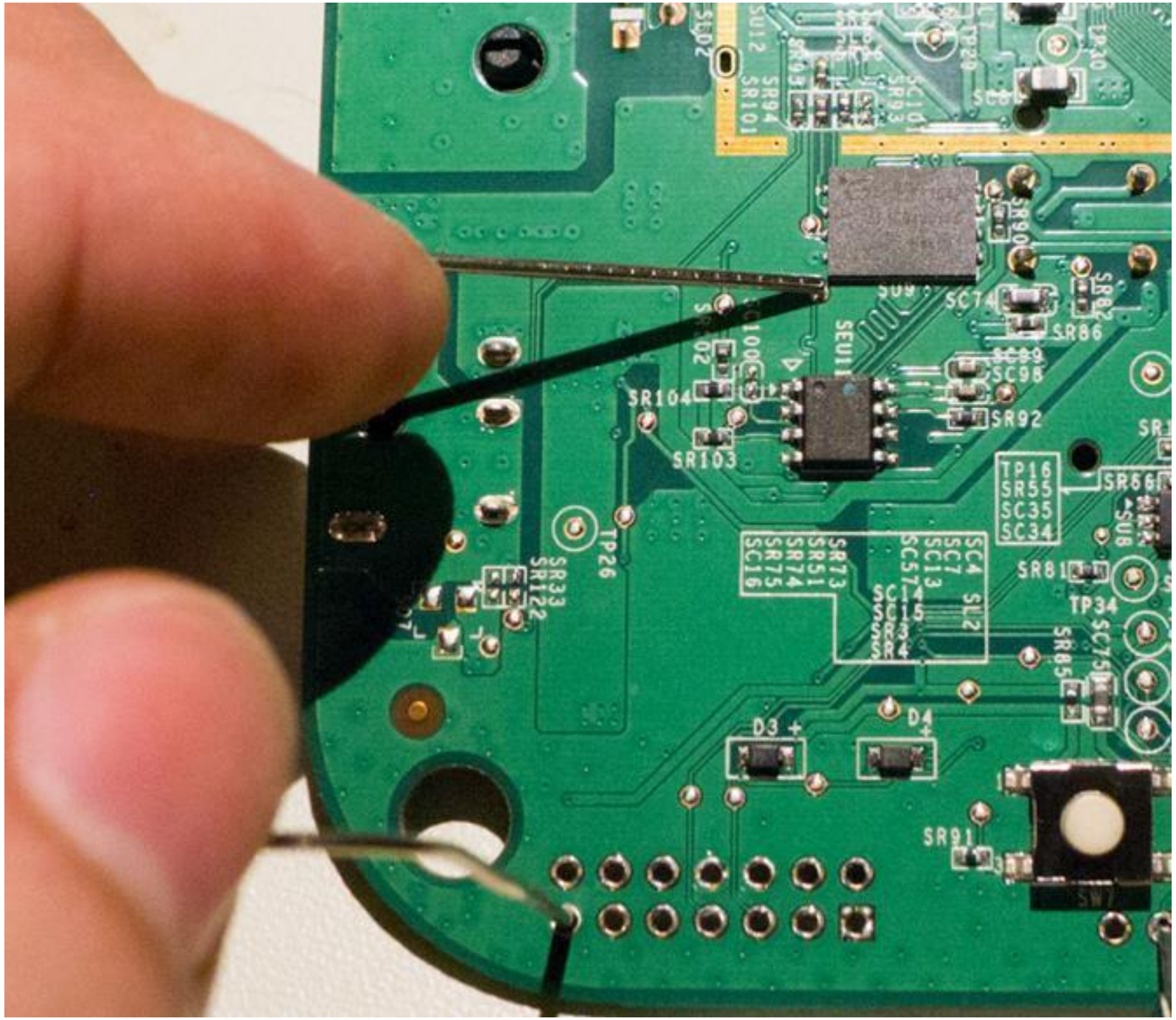
FW Upgrade File

Test Dumper Image

```
## 2a6:      80 e1         ldi    r24, 0x10    ; 16
## 2a8:      84 b9         out    0x04, r24    ; 4
##    while(1){
##    PORTB = 0x01;
## 2aa:      91 e0         ldi    r25, 0x01    ; 1
##    PORTB = 0xFF;
## 2ac:      8f ef         ldi    r24, 0xFF    ; 255
##    DDRB = (1<<4);
##    while(1){
##    PORTB = 0x01;
## 2ae:      95 b9         out    0x05, r25    ; 5
##    PORTB = 0xFF;
## 2b0:      85 b9         out    0x05, r24    ; 5
## 2b2:      fd cf         rjmp   .-6          ; 0x2ae <main+0x40>

patch_togglepins = [0x80, 0xE1, 0x84, 0xB9, 0x91, 0xE0, 0x8F, 0xEF,
0x95, 0xB9, 0x85, 0xB9, 0xFD, 0xCF]
```

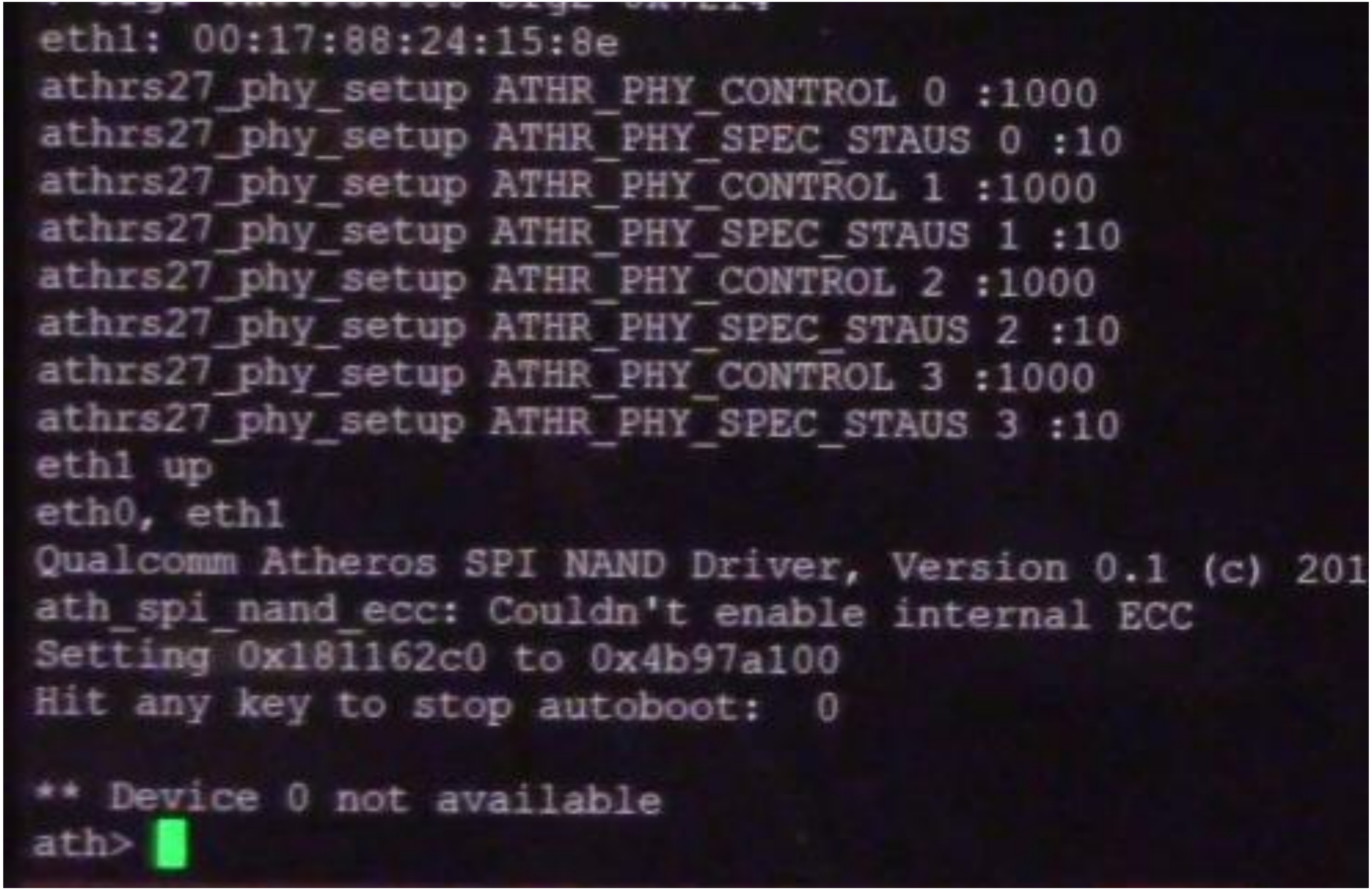

Zig Bee SoC
Boot Flash
USB-Serial
SPI NAND Flash
DDR
Qualcom QCA4531
HUE BRIDGE 2.0_X03A

HACKING
TOOLS



https://www.youtube.com/watch?v=hi2D2MnwiGM

Or: http://www.oflynn.com



```
eth1: 00:17:88:24:15:8e
athrs27_phy_setup ATHR_PHY_CONTROL 0 :1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 0 :10
athrs27_phy_setup ATHR_PHY_CONTROL 1 :1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 1 :10
athrs27_phy_setup ATHR_PHY_CONTROL 2 :1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 2 :10
athrs27_phy_setup ATHR_PHY_CONTROL 3 :1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 3 :10
eth1 up
eth0, eth1
Qualcomm Atheros SPI NAND Driver, Version 0.1 (c) 201
ath_spi_nand_ecc: Couldn't enable internal ECC
Setting 0x181162c0 to 0x4b97a100
Hit any key to stop autoboot:  0

** Device 0 not available
ath>
```

Want to know more?

Want to know more?

- The paper and videos are at

Want to know more?

- The paper and videos are at

  iotworm.eyalro.net

Want to know more?

- The paper and videos are at
  iotworm.eyalro.net
- A great source for tutorials on hardware attacks

Want to know more?

- The paper and videos are at
  iotworm.eyalro.net
- A great source for tutorials on hardware attacks
  wiki.newae.com

Want to know more?

- The paper and videos are at
  iotworm.eyalro.net
- A great source for tutorials on hardware attacks
  wiki.newae.com