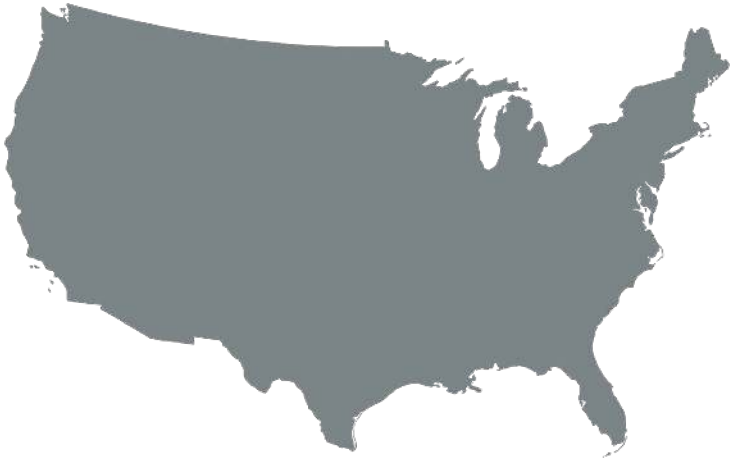
The Tao of Hardware & The Te of Implants

whoami?

- EE w/ some CS and Infosec
- 10 years of fun with hardware
  - silicon debug
  - security research
  - pen testing of CPUs
- 5 years of Hardware Security Training:
  - “Applied Physical Attacks on x86 Systems”
- Bside PDX!!!

Joe FitzPatrick
@securelyfitz
joefitz@securinghardware.com

SECURING HARDWARE.COM



OR

Not this TAO

This Tao

Tao

/dou,tou/

noun

(in Chinese philosophy) the absolute principle underlying the universe, combining within itself the principles of yin and yang and signifying the way, or code of behavior, that is in harmony with the natural order.

Tao

/dou,tou/

noun

(in Chinese philosophy) the absolute principle underlying the universe, combining within itself the principles of yin and yang and signifying the way, or code of behavior, that is in harmony with the natural order.

Tao of Hardware:
Hardware is the absolute that underlies the world of computing

Te

/də/

noun

(in Chinese philosophy) A key concept in Chinese philosophy, usually translated "inherent character; inner power; integrity" in Taoism, "moral character; virtue; morality" in Confucianism and other contexts, and "quality; virtue" (guna) or "merit; virtuous deeds" (punya) in Chinese Buddhism.

Te

/də/

noun

(in Chinese philosophy) A key concept in Chinese philosophy, usually translated "inherent character; inner power; integrity" in Taoism, "moral character; virtue; morality" in Confucianism and other contexts, and "quality; virtue" (guna) or "merit; virtuous deeds" (punya) in Chinese Buddhism.

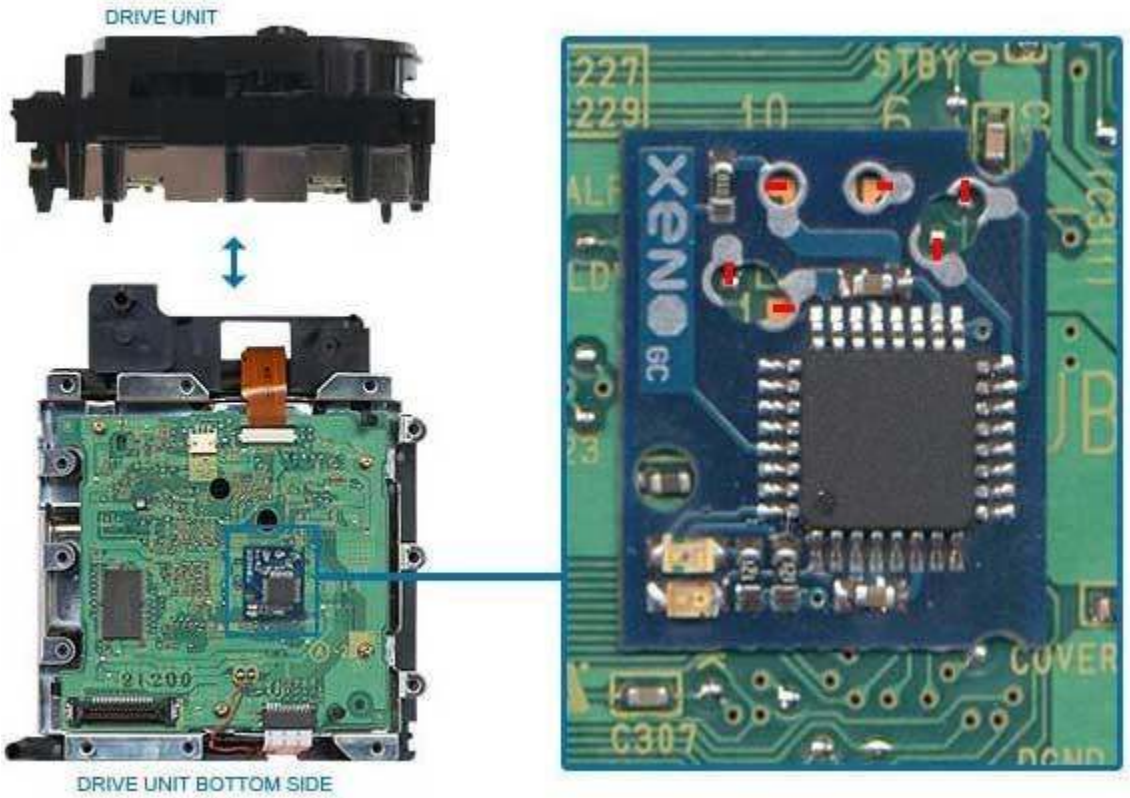
Te of Implants:
Understanding and harnessing the inner strength
of minor additions to hardware systems

Do the difficult things while they are simple and do the great things while they are small

-- Lao-Tze

“Hardware Implants” pre-2013

“Hardware Implants” pre-2013

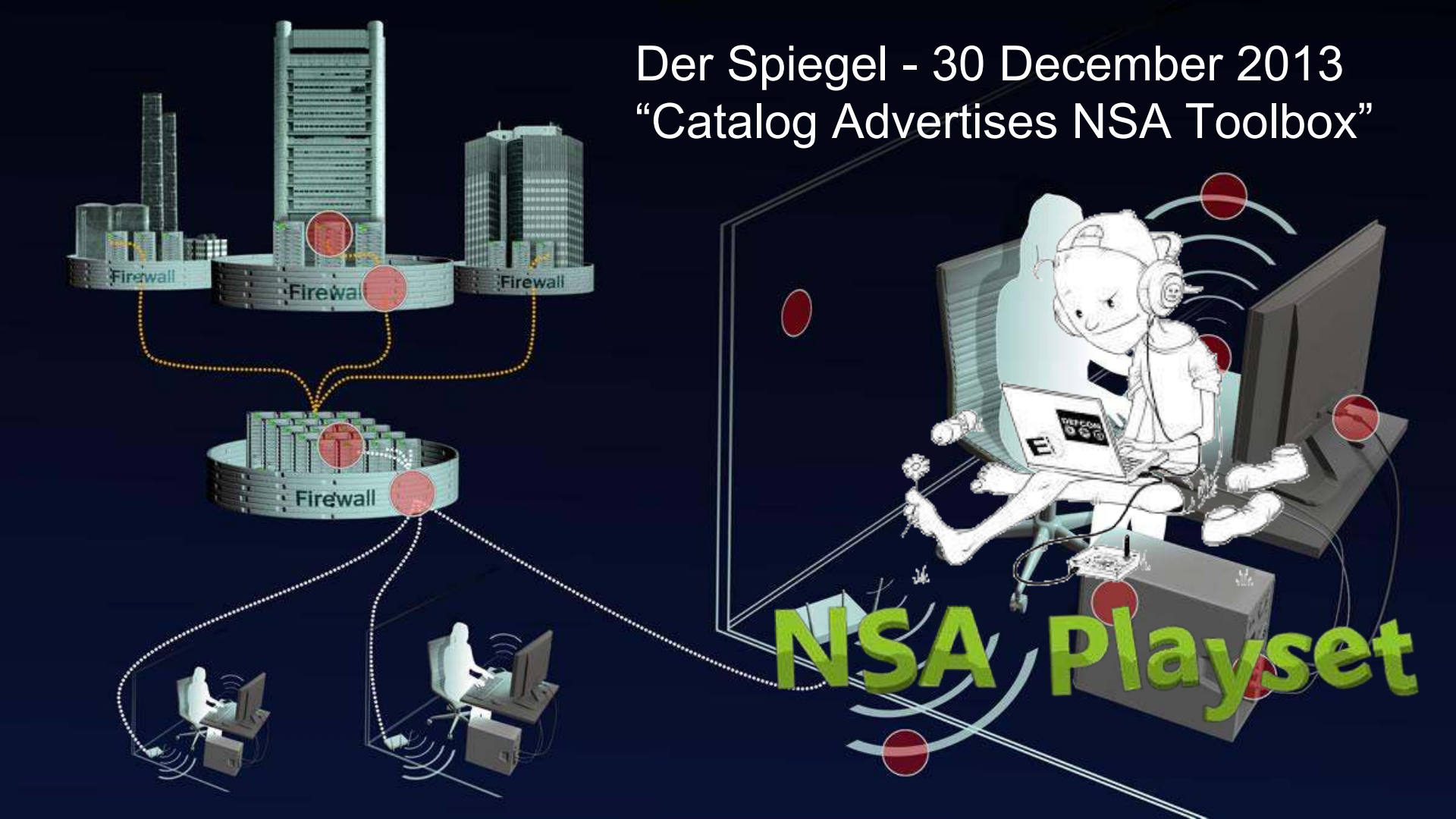


curinghardware.com

https://www.modchipcentral.com/store/images/xeno-gc-install-diagram.jpg

Der Spiegel - 30 December 2013
“Catalog Advertises NSA Toolbox”

Der Spiegel - 30 December 2013
“Catalog Advertises NSA Toolbox”



So What?

- Malicious hardware implants are real
- Hardware implants don’t live in a vacuum
- Use hardware to give software access, then back off

Today’s Implants:

1. Blindly escalate privilege using JTAG
2. Patch kernels via DMA on an embedded device
3. Enable wireless control of an off-the-shelf PLC
4. Hot-plug a malicious PLC expansion
5. BadUSB-style display adapters

Today's Implants:

1. Blindly escalate privilege using JTAG
2. Pa
3. En
4. Hot-plug a malicious PLC expansion

[Dailydave] Junk Hacking Must Stop!

Dave Aitel dave at immunityinc.com
Mon Sep 22 14:53:47 EDT 2014

- Previous message: [Dailydave] Protecting your code versions.
- Next message: [Dailydave] Junk Hacking Must Stop!
- **Messages sorted by:** [date] [thread] [subject] [author]

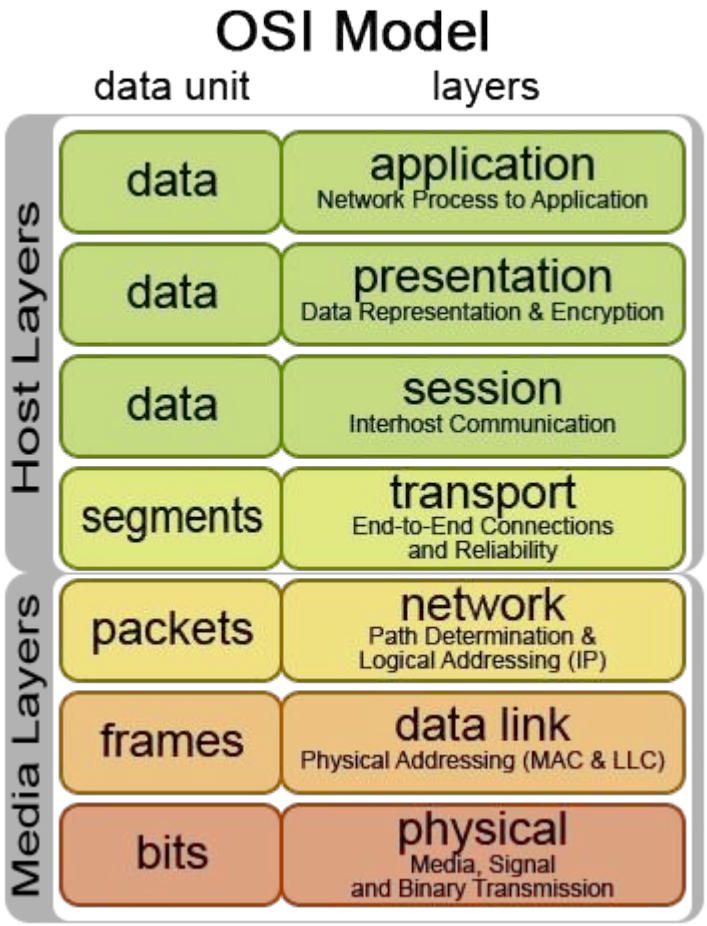
Junk Hacking

1. Blindly escalate privilege using JTAG
2. Patch kernels via DMA on an embedded device
3. Enable wireless control of an off-the-shelf PLC
4. Hot-plug a malicious PLC expansion

Critical Infrastructure

Today’s Implants:

1. Blindly escalate privilege using JTAG
2. Patch kernels via DMA on an embedded device
3. Enable wireless control of an off-the-shelf PLC
4. Hot-plug a malicious PLC expansion
5. BadUSB-style display adapters



JTAG Model

Boundary scan, Halt/Resume, Memory access

--- (no one uses this crap)

--- N/A - sessionless...

Target-specific configuration

IR/DR access

TAP FSM

TDI, TDO, TMS, TCK, TRST

Jtagsploitation: JTAG to Root, 5 ways

1. Modify Non-Volatile Storage via Boundary Scan
2. Scrape memory for offline analysis
3. Patching bootargs
4. Live Kernel Patching
5. Finding & Patching a Process

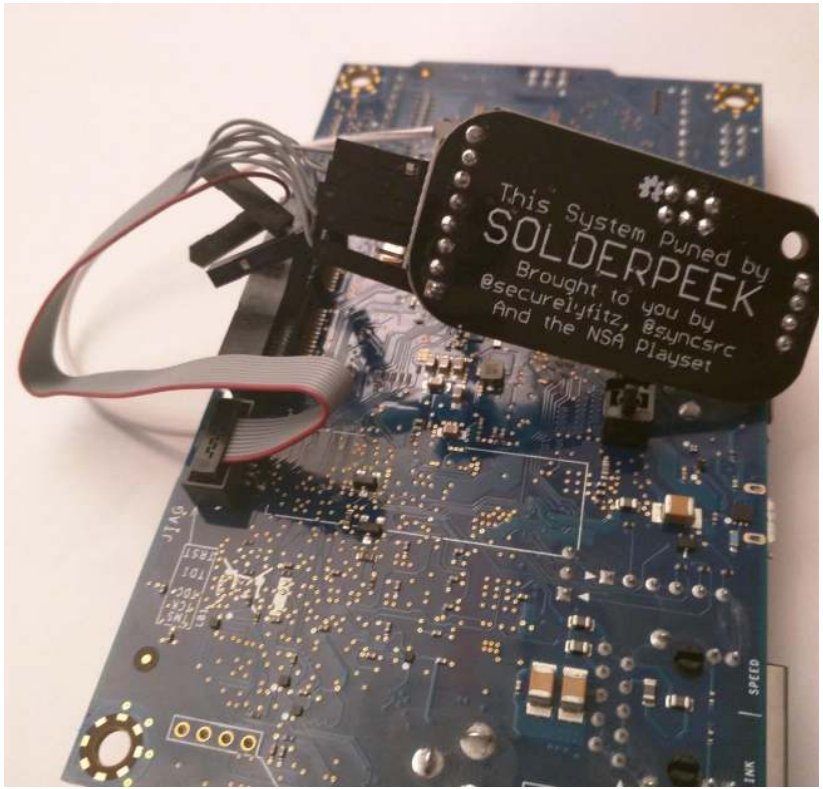
44CON, Sept 2015 & Bsides Portland, Oct 2015 https://github.com/syncsrc/jtagsploitation

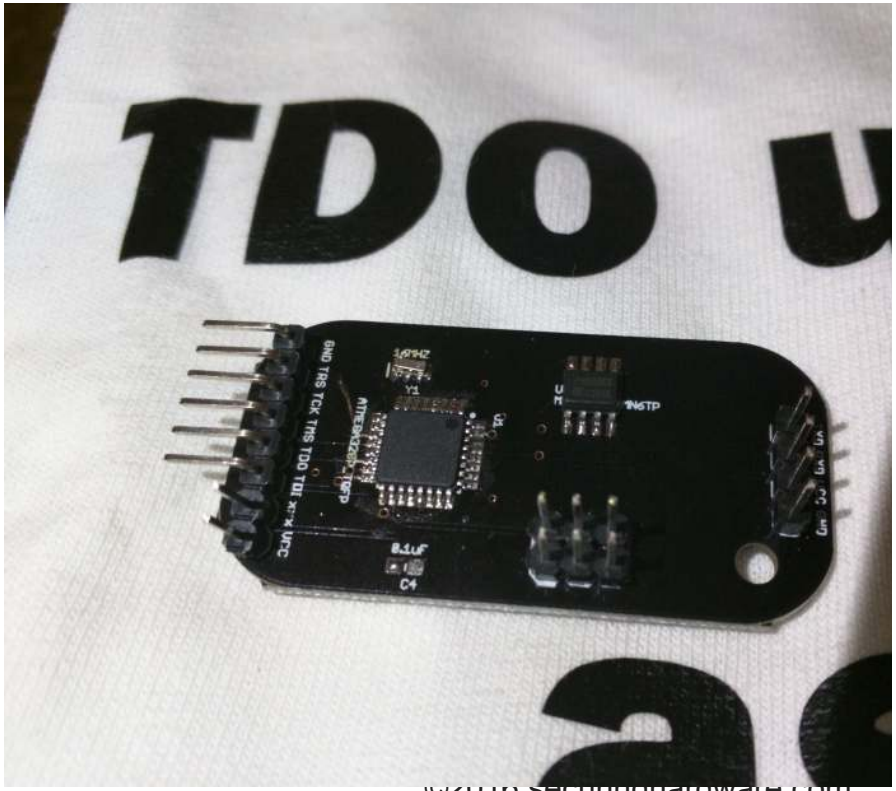
getty Parameters

```
$ xxd /sbin/getty
...
00006770  1b 5b 48 1b 5b 4a 00 25  73 25 73 20 28 61 75 74  |.[H.[J.%s%s (aut|
00006780  6f 6d 61 74 69 63 20 6c  6f 67 69 6e 29 0a 00 25  |omatic login)..%|
00006790  73 3a 20 72 65 61 64 3a  20 25 6d 00 25 73 3a 20  |s: read: %m.%s: |
000067a0  69 6e 70 75 74 20 6f 76  65 72 72 75 6e 00 63 68  |input overrun.ch|
000067b0  65 63 6b 6e 61 6d 65 20  66 61 69 6c 65 64 3a 20  |eckname failed: |
000067c0  25 6d 00 2d 68 00 2d 66  00 2d 2d 00 25 73 3a 20  |%m.-h.-f.--.%s: |
000067d0  63 61 6e 27 74 20 65 78  65 63 20 25 73 3a 20 25  |can't exec %s: %|
000067e0  6d 00 38 62 69 74 73 00  61 75 74 6f 6c 6f 67 69  |m.8bits.autologi|
```

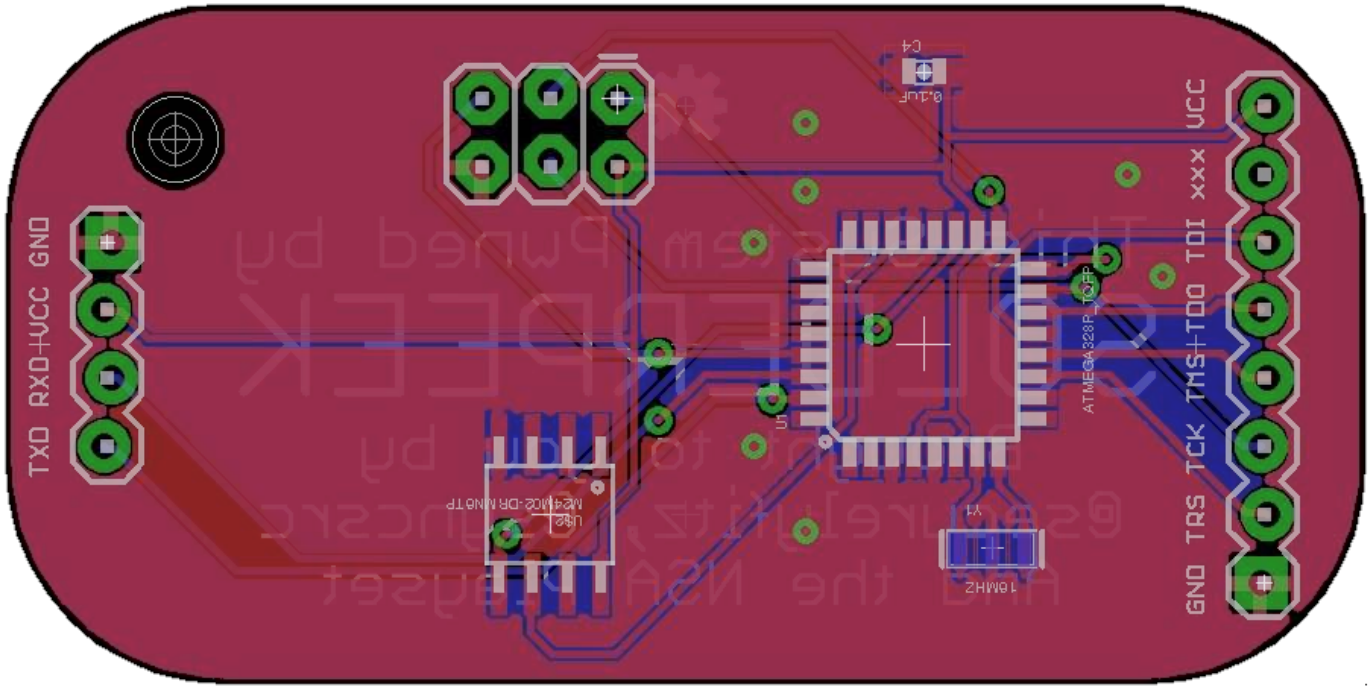
Changing ‘--’ to ‘-f’ results in user being pre-authenticated

NSA Playset: SOLDERPEEK





NSA Playset: SOLDERPEEK



Generating an SVF from OpenOCD

~/openocd-0.9.0/contrib

File Edit View Search Terminal Help

```
[matt@         contrib]$ ./log2svf.py -h
usage: log2svf.py [-h] [-v] [-w] [-j JTAG] [-s SVF] -l LOG [LOG ...]

This script will take an OpenOCD log file and generate an SVF from it.

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit
  -w, --warn            Issue warnings for non-fatal errors instead of exiting
  -j JTAG, --jtag JTAG  file to read DEBUG_JTAG parameters from (defaults to
                        src/jtag/jtag.h)
  -s SVF, --svf SVF     Output file
  -l LOG [LOG ...], --log LOG [LOG ...]
                        OpenOCD log file(s) to parse. If multiple files are
                        specified commands from all log files will be
                        concatenated into a single SVF

This script parses debug messages from the jtag_build_buffer() function. To
enable JTAG debug printing, OpenOCD must be configured with the '--enable-
verbose-jtag-io' option and run with the '-d 1' switch. It is also recommended
to run with '-c init -c "poll off"' to disable target status polling.
[matt@         contrib]$ ./log2svf.py -l ../bin/openocd.log -w
!Begin Test Program
TRST OFF;
ENDIR IDLE;
ENDDR IDLE;
!WARNING: Scan length of 672 exceeded length of data from log file: 64
!WARNING: Scan length of 672 exceeded length of data from log file: 64
SDR 672 TDI (0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000FFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF);
SIR 7 TDI (7F);
SIR 5 TDI (1E);
```


intel Galileo
Galileo
#00

Today’s Implants:

1. Blindly escalate privilege using JTAG
2. Patch kernels via DMA on an embedded device
3. Enable wireless control of an off-the-shelf PLC
4. Hot-plug a malicious PLC expansion
5. BadUSB-style display adapters

So What?

- Implant hardware is simple, cheap, and readily available
- Payload can be prototyped with standard tools
- When headers exist, installation is quick
- Pinout is adaptable easily enough
- It’s small enough to stick in lots of places
- JTAG is JTAG is JTAG* - tested on ARM, MIPS and x86

*mostly

What is DMA?

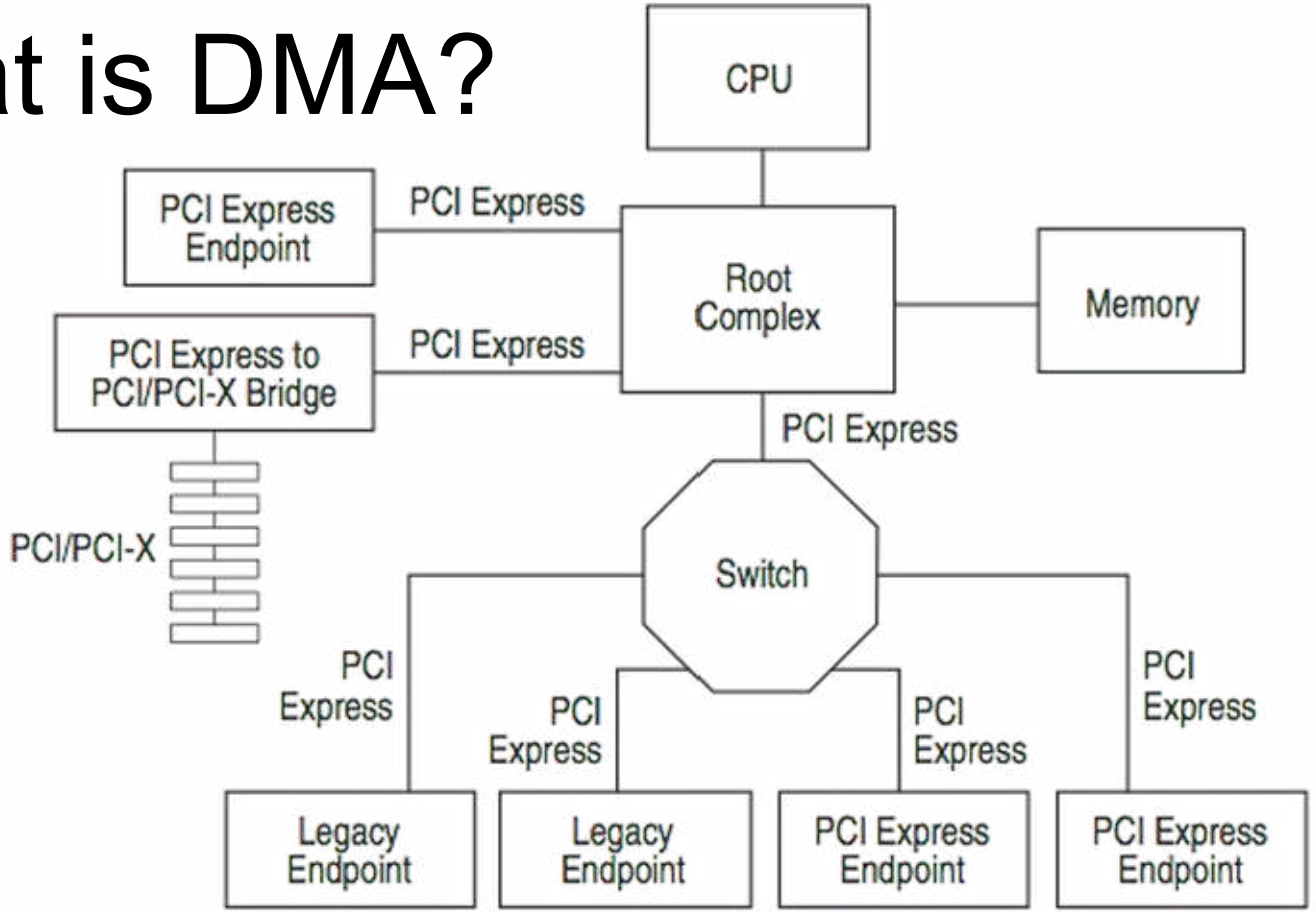


Diagram: PCIe 2.1 specification

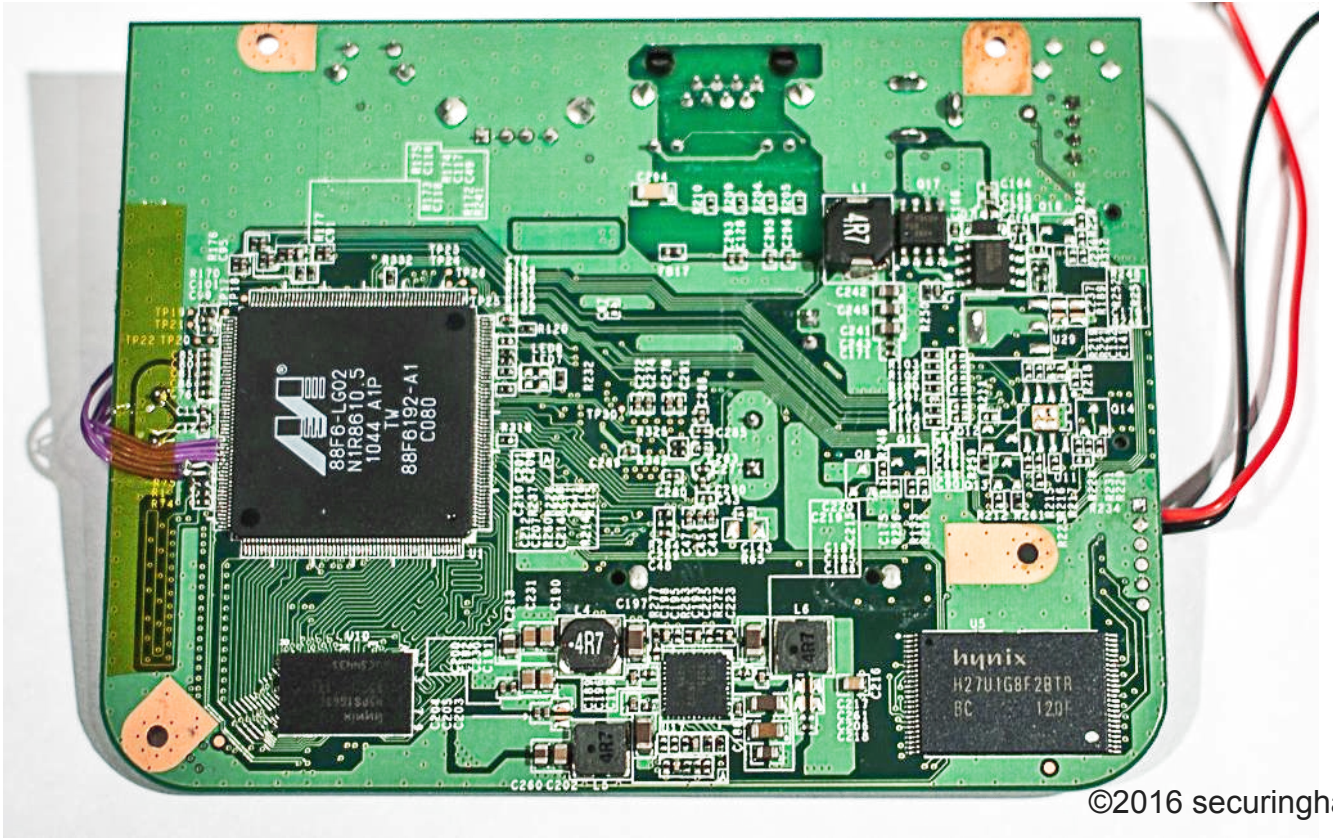
A Brief History of DMA attacks


Ver1.0
4312

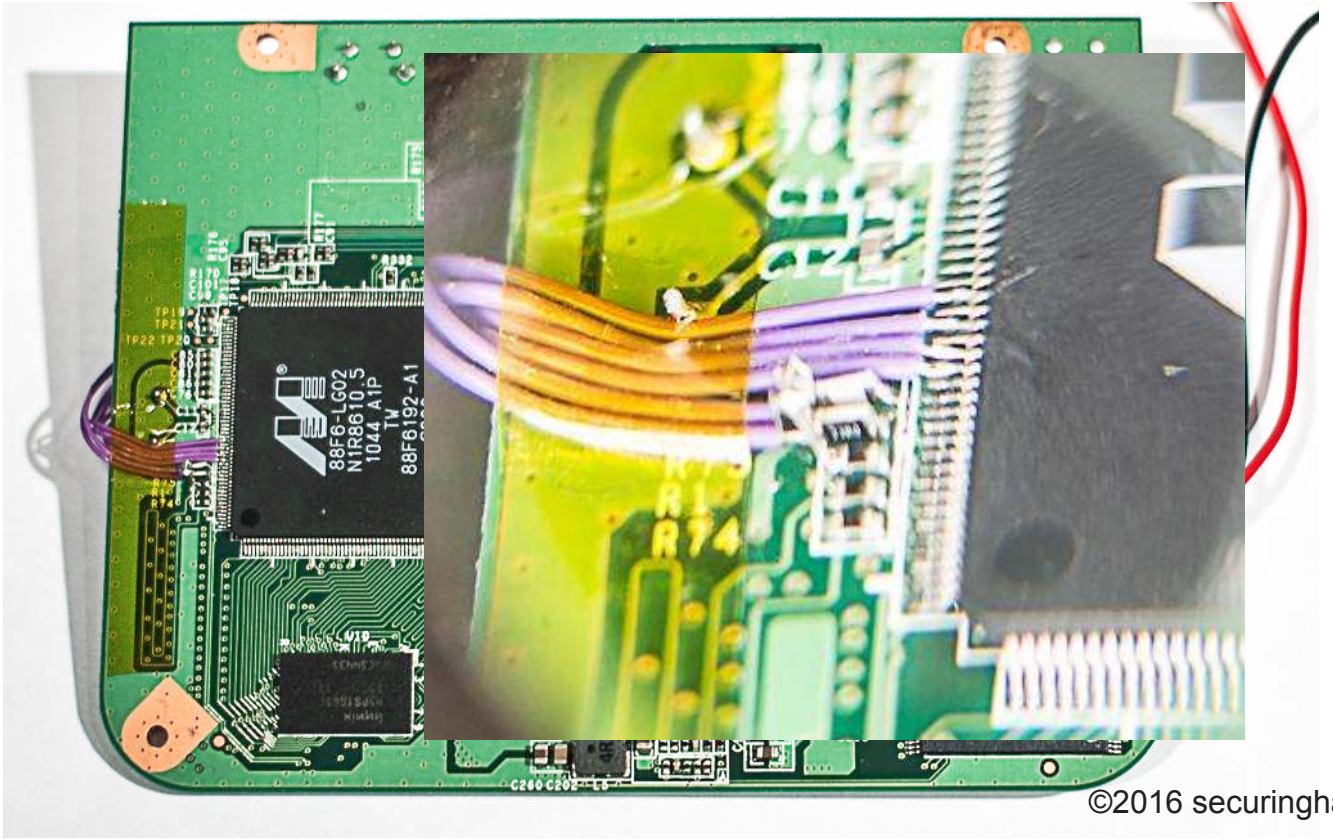
Advancing ARM/MIPS systems

ghardware.com

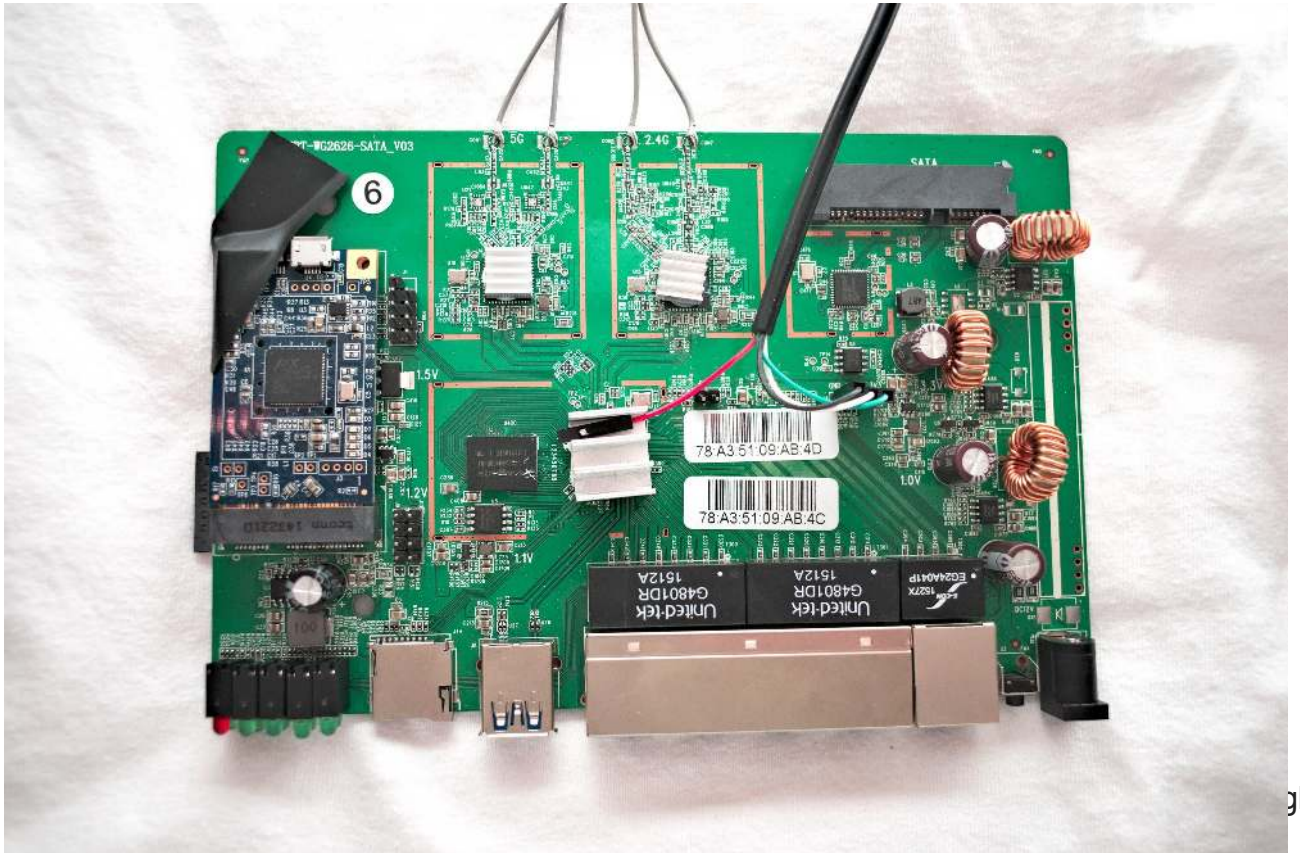
PCIe on embedded hardware



PCIe on embedded hardware

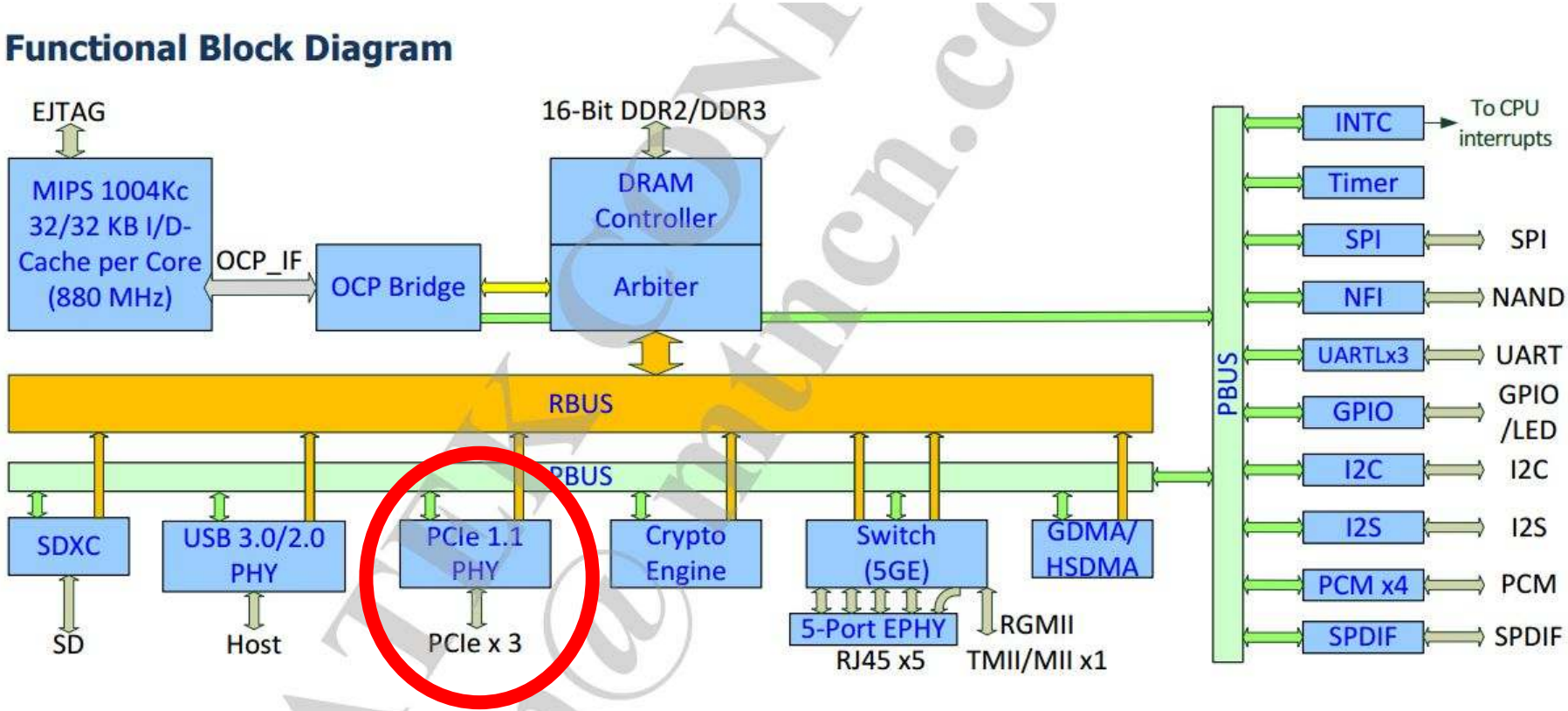


PCIe on embedded hardware



ghardware.com

PCIe on embedded hardware



Attack-side Software

Quick ‘n’ dirty PCIe memory read/write with PyUSB

```
while baseAddress<endAddress:
    print('BBBBI',0xcf,0,0,0x40,baseAddress)
    print("addr",baseAddress)
    pciout.write(struct.pack('BBBBI',0xcf,0,0,0x40
        ,baseAddress))
    cache+=pciin.read(0x100)
    baseAddress+=256
return bytes(cache[offset:offset+byteCount])
```

```
bufferIndex=0
while baseAddress<endAddress:
    subbuf=readbuf[bufferIndex:bufferIndex+128]
    print("addr",baseAddress,'subbuf',len(subbuf))
    pciout.write(struct.pack('BBBBI'+'B'*128,0x4f,
        0,0,0x20,baseAddress,*subbuf))
    baseAddress+=128
    bufferIndex+=128
```

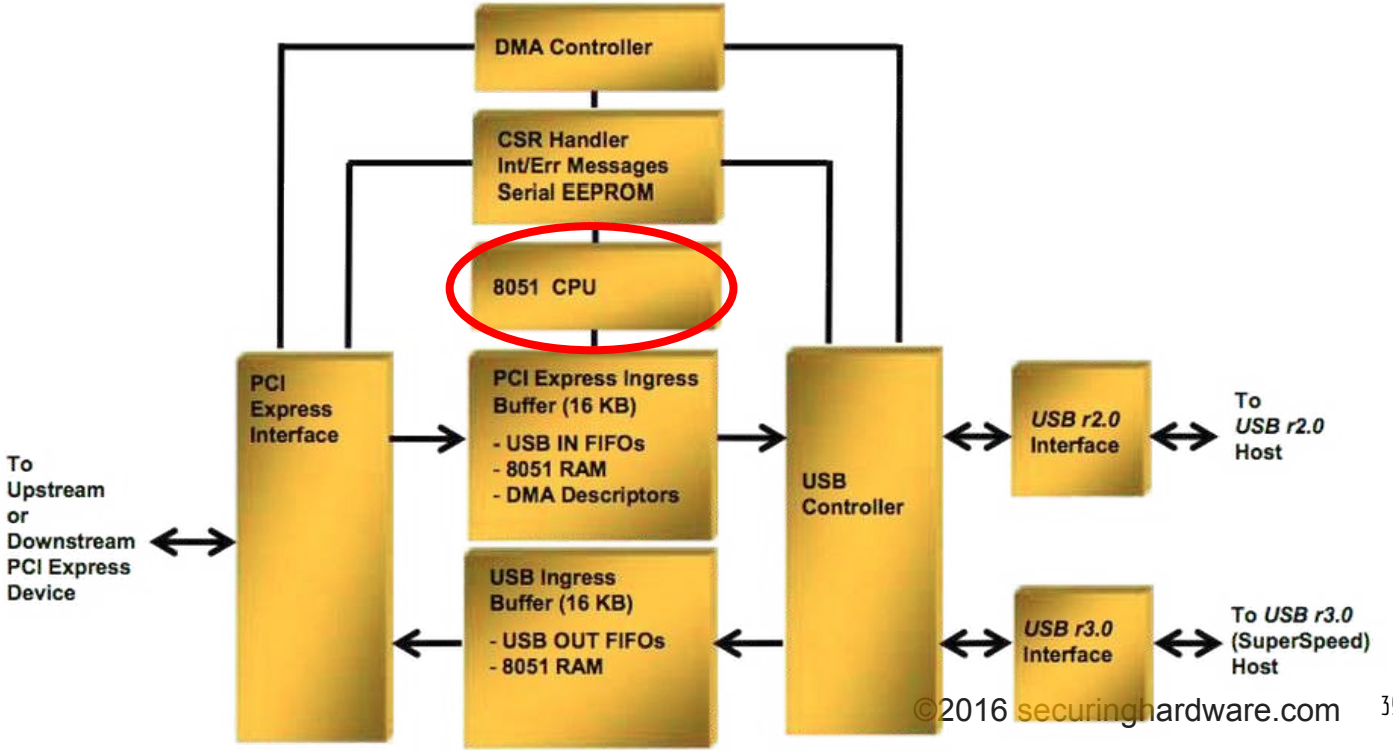
Linux File System ACL Enforcement

```
int generic_permission(struct inode *inode, int mask) {
        int ret;
        /*
         * Do the basic permission checks.
         */
        ret = acl_permission_check(inode, mask);
        if (ret != -EACCES)
                return ret;
.....
        /*
         * Searching includes executable on directories, else just read.
         */
        mask &= MAY_READ | MAY_WRITE | MAY_EXEC;
        if (mask == MAY_READ)
                if (capable_wrt_inode_uidgid(inode, CAP_DAC_READ_SEARCH))
                        return 0;

        return -EACCES;
}
```

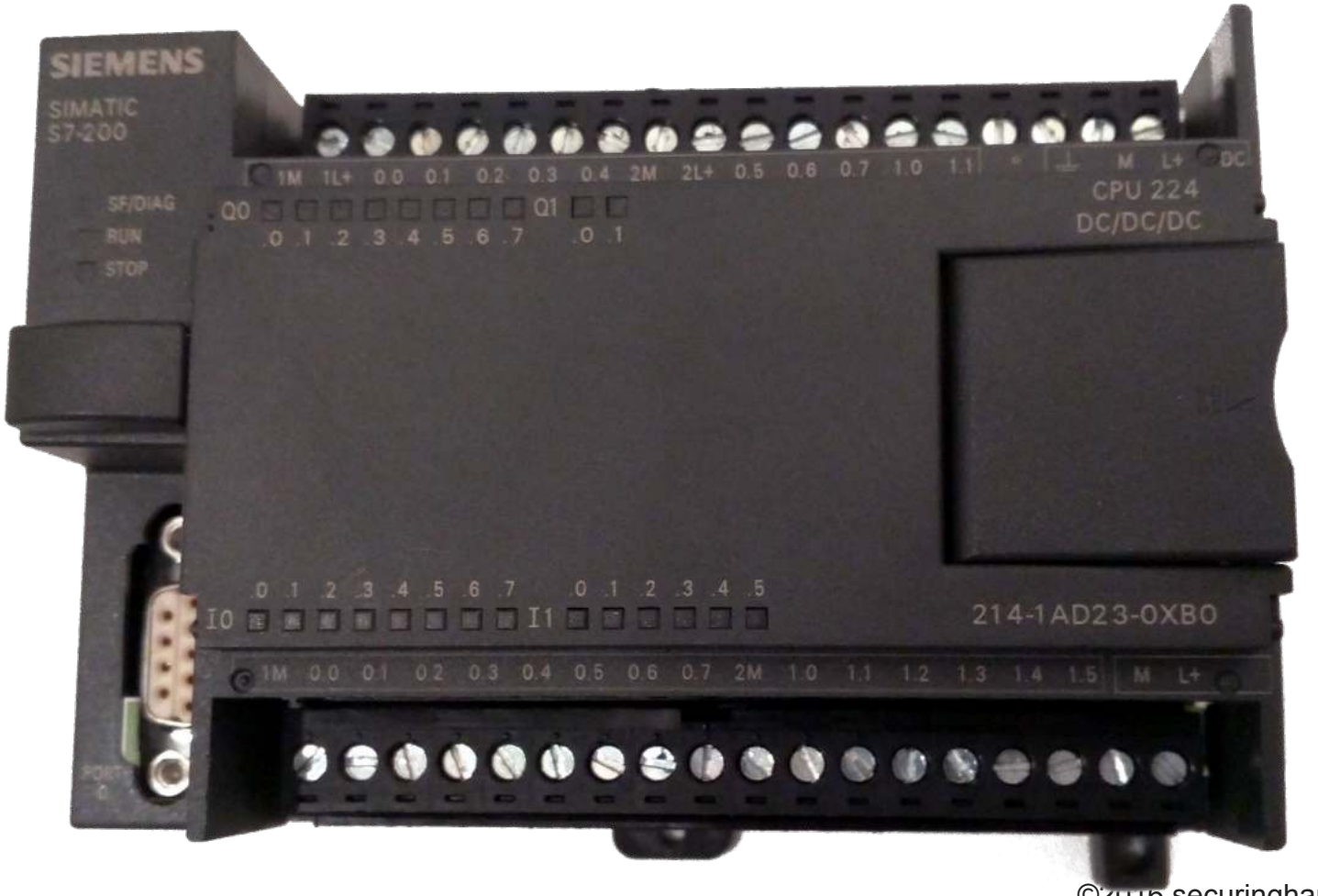
Implant-izing with 8051 code

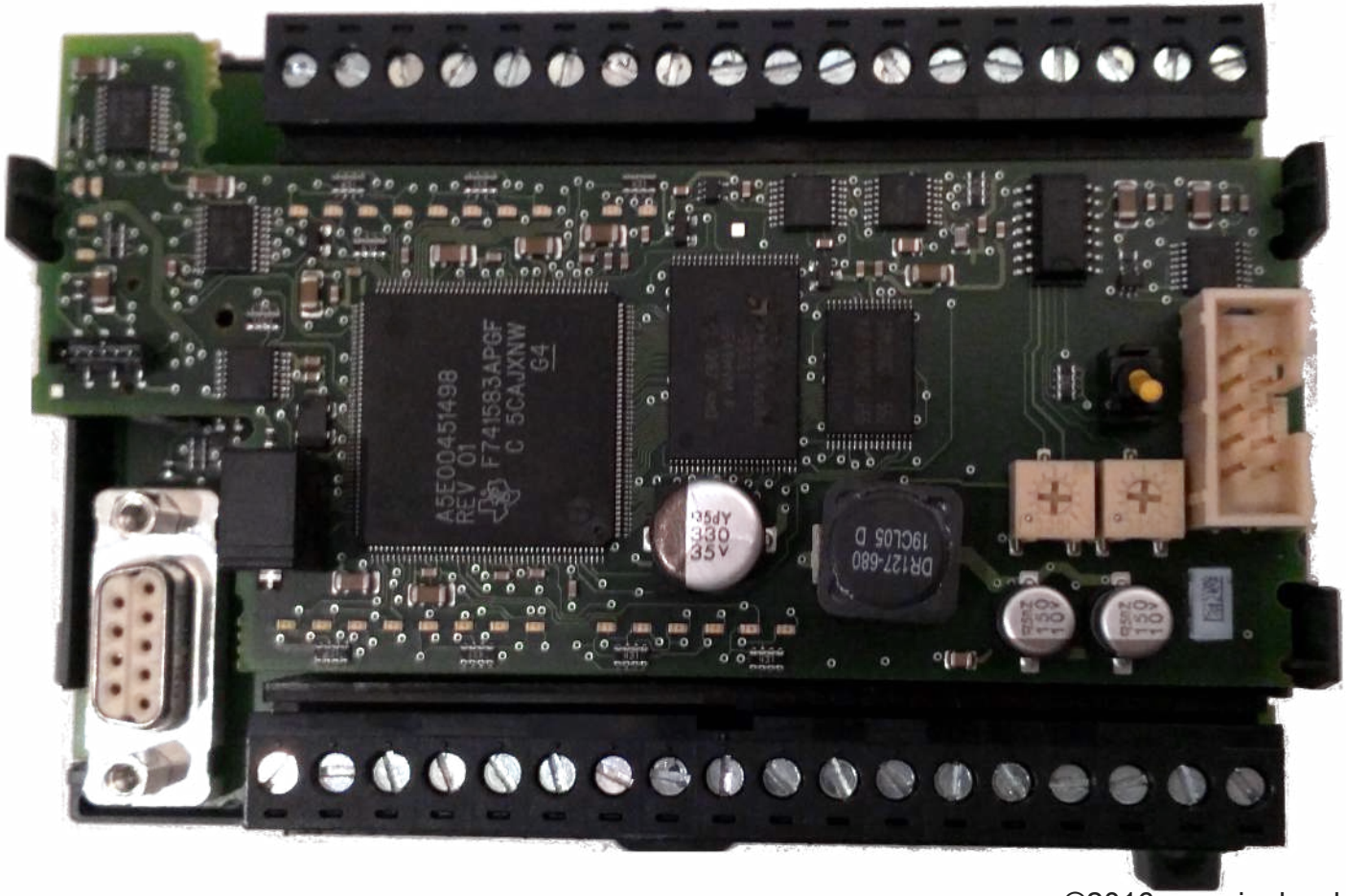
Figure 1-1. USB 3380 Block Diagram

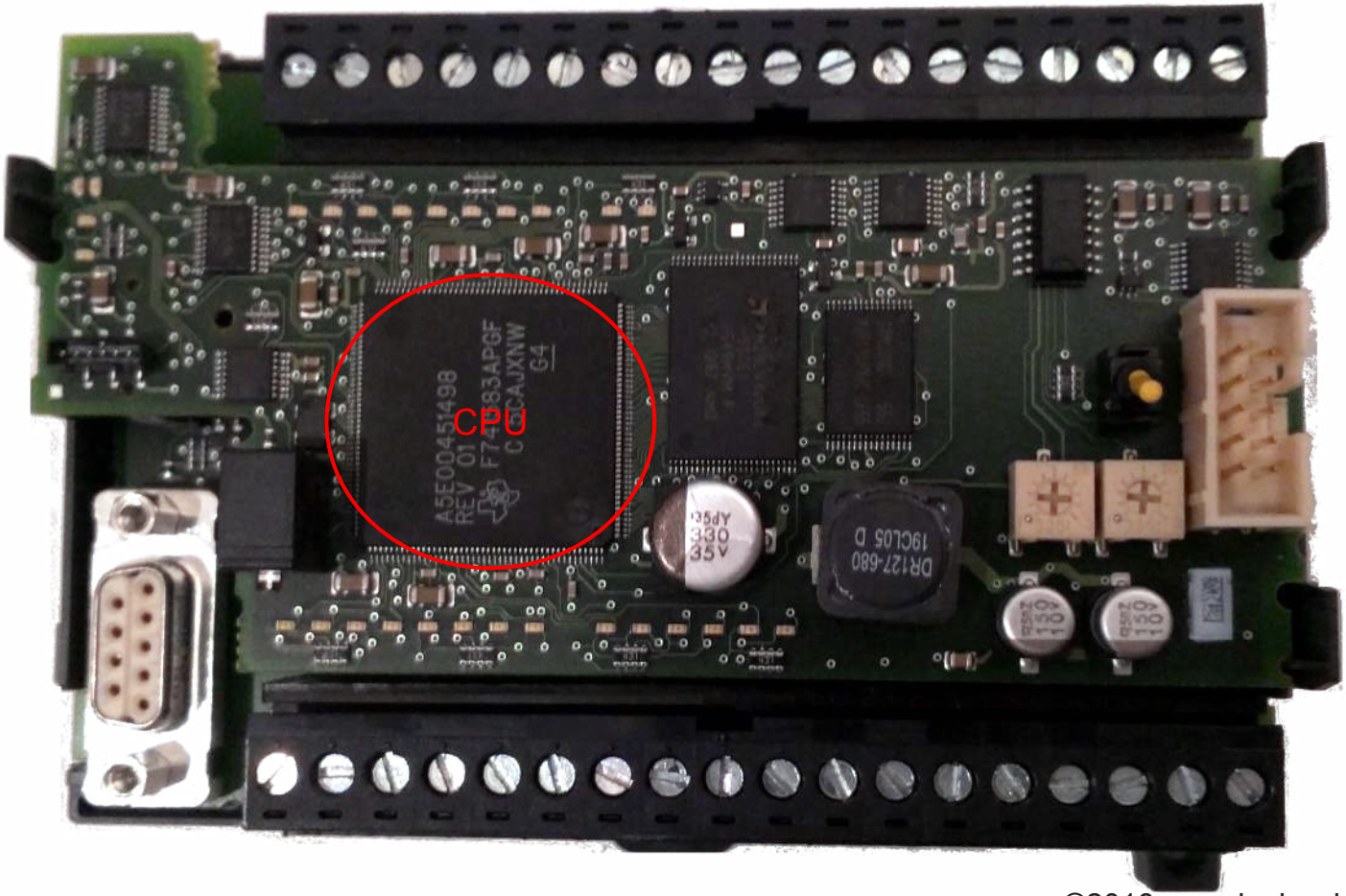


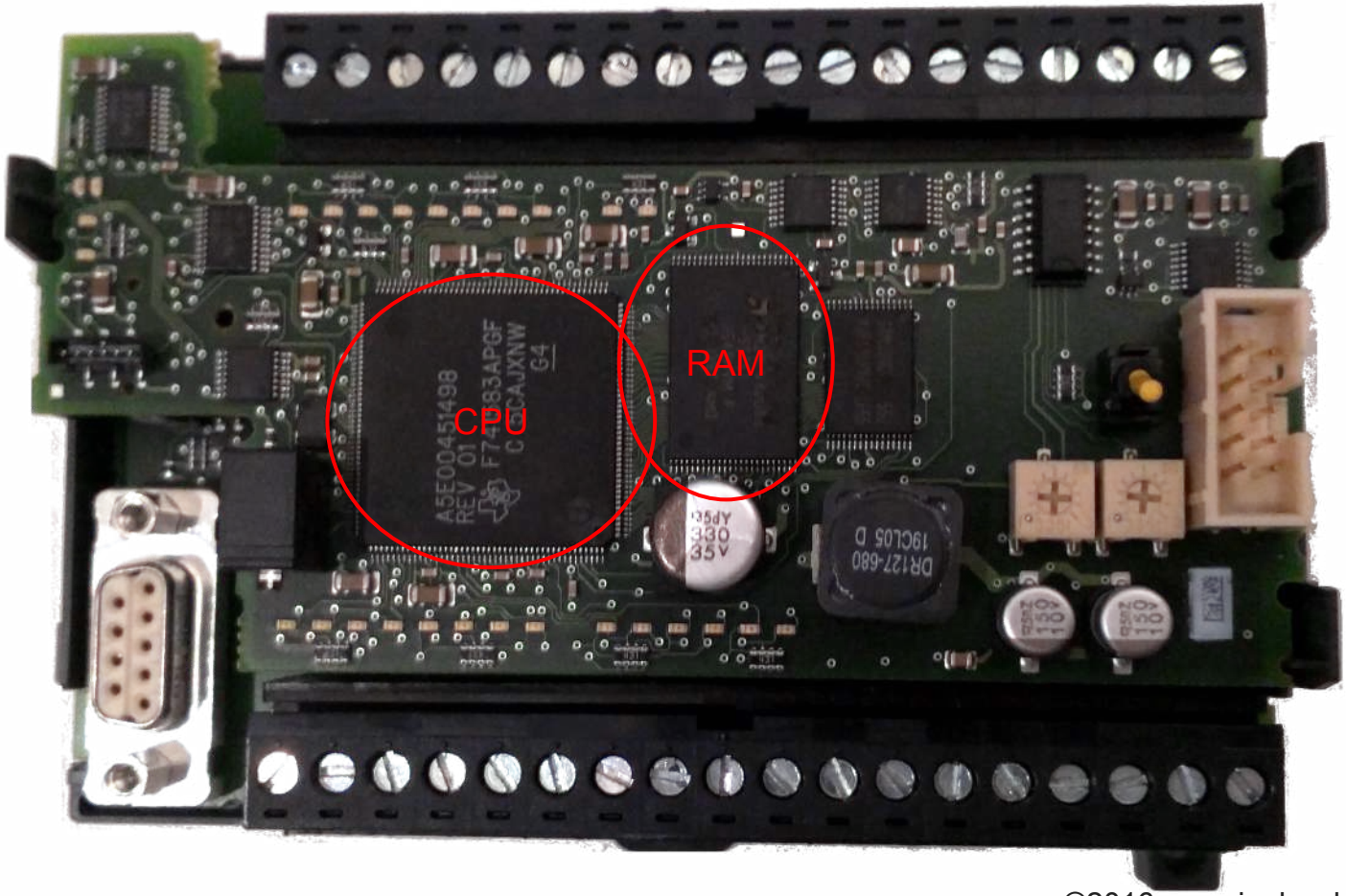
Today’s Implants:

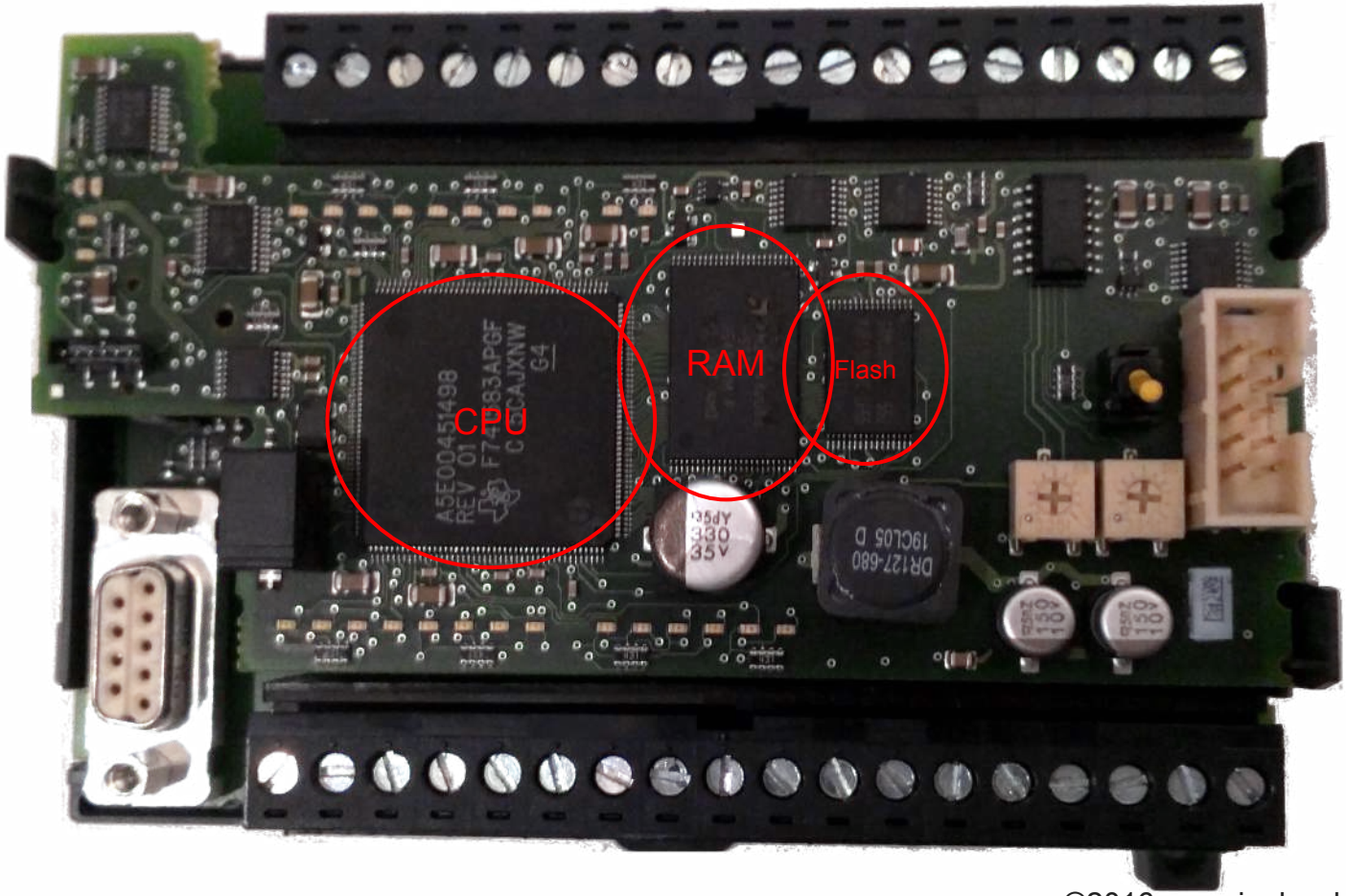
1. Blindly escalate privilege using JTAG
2. Patch kernels via DMA on an embedded device
3. Enable wireless control of an off-the-shelf PLC
4. Hot-plug a malicious PLC expansion
5. BadUSB-style display adapters

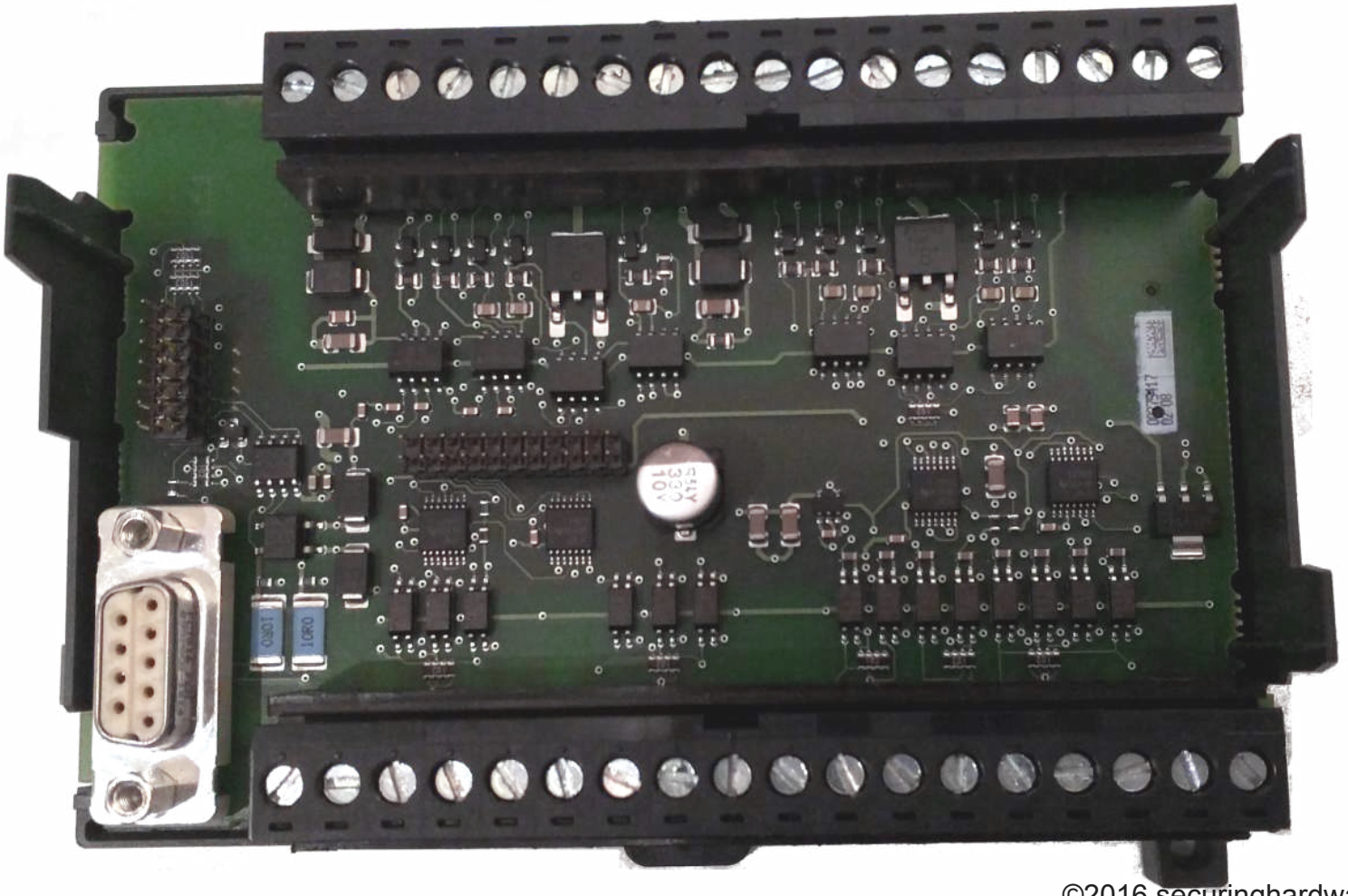


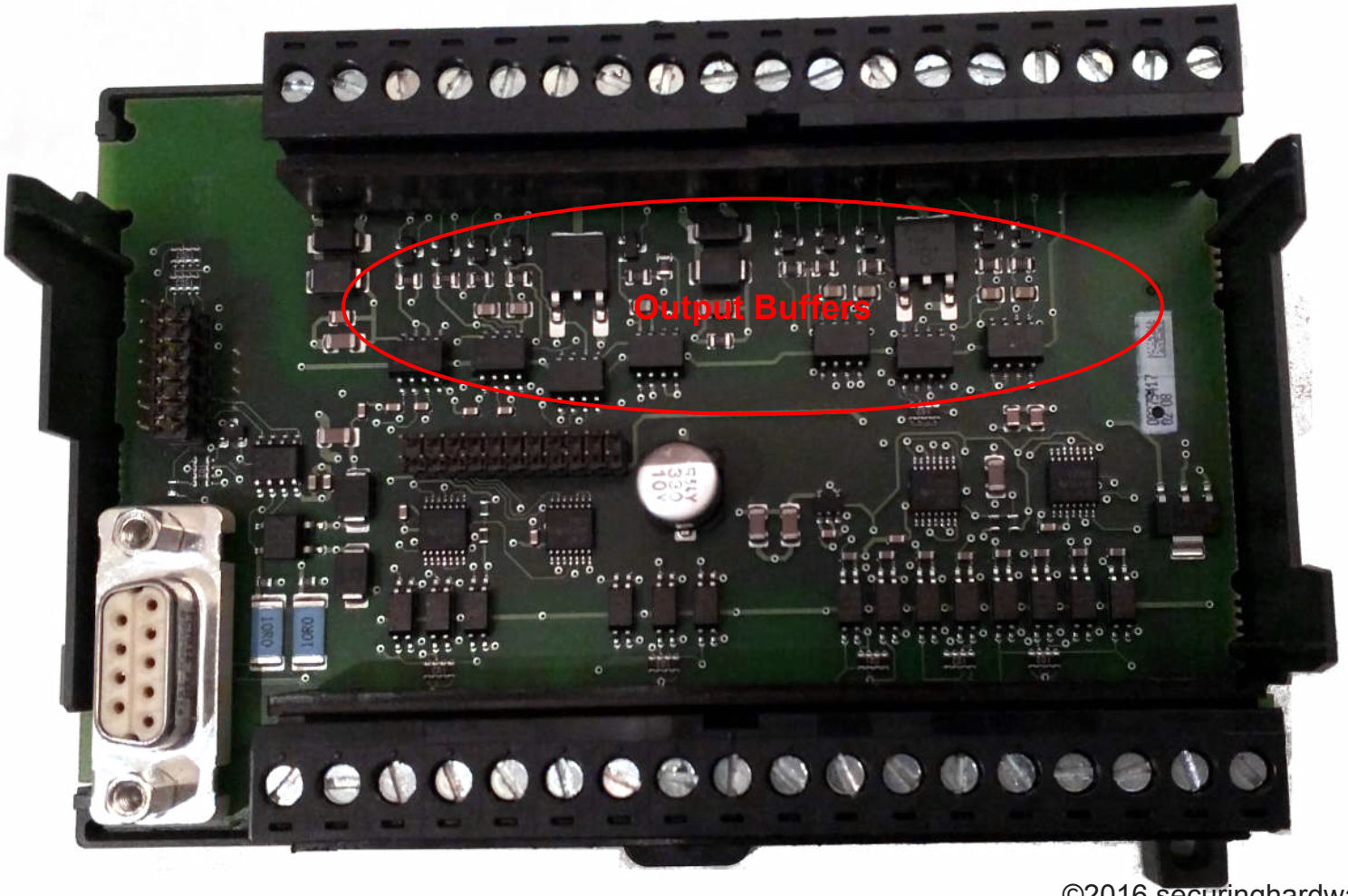


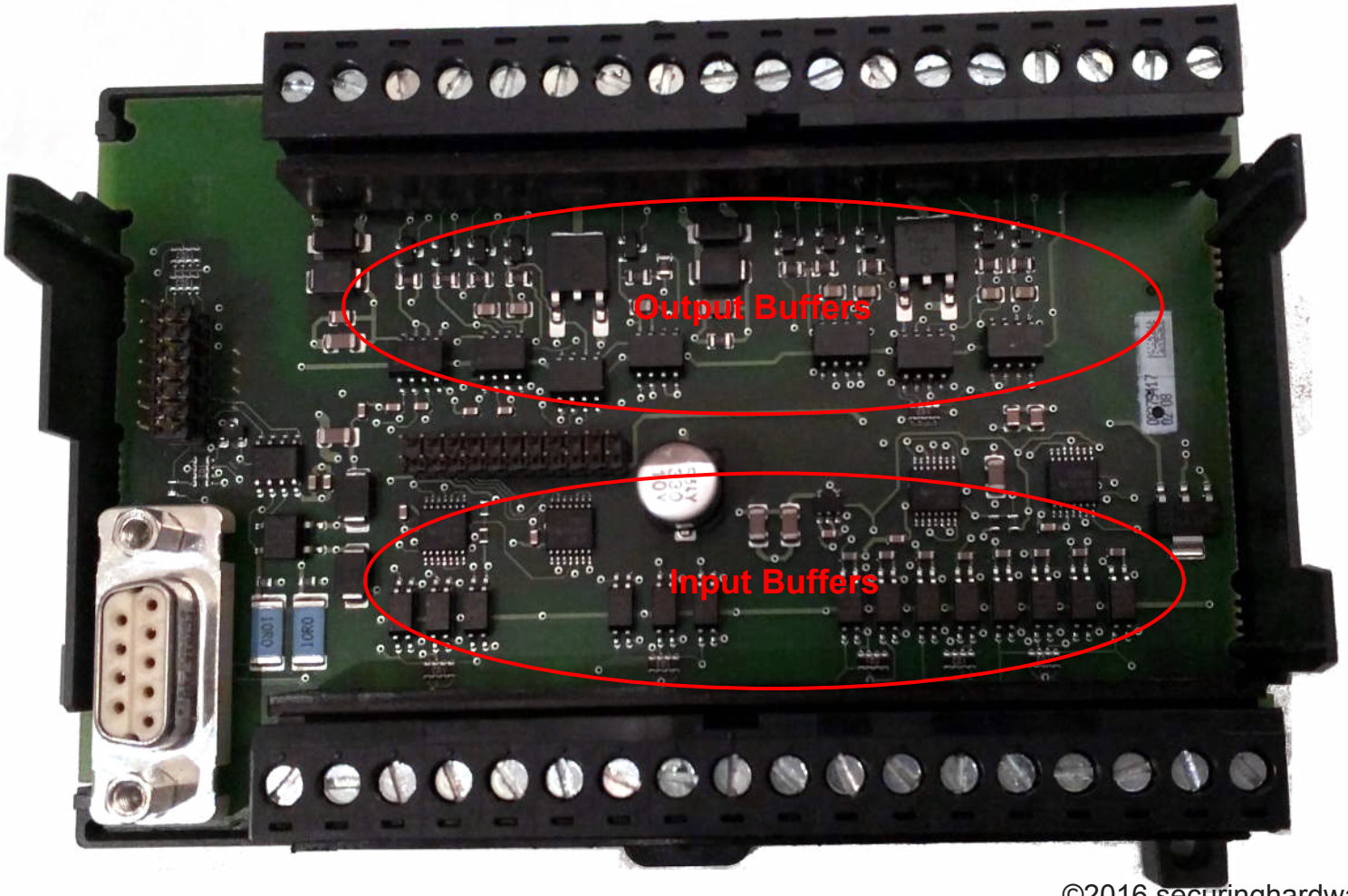


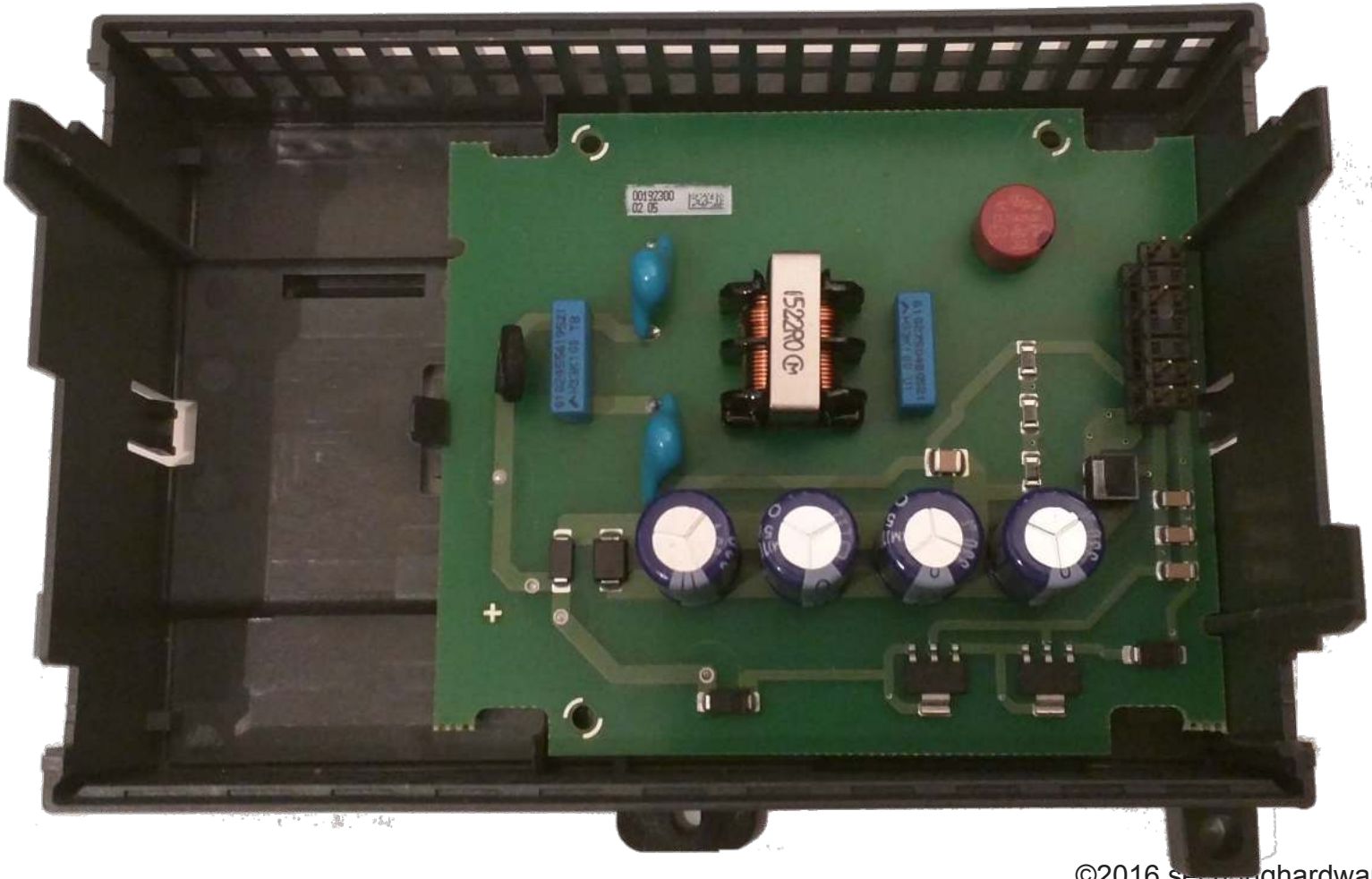


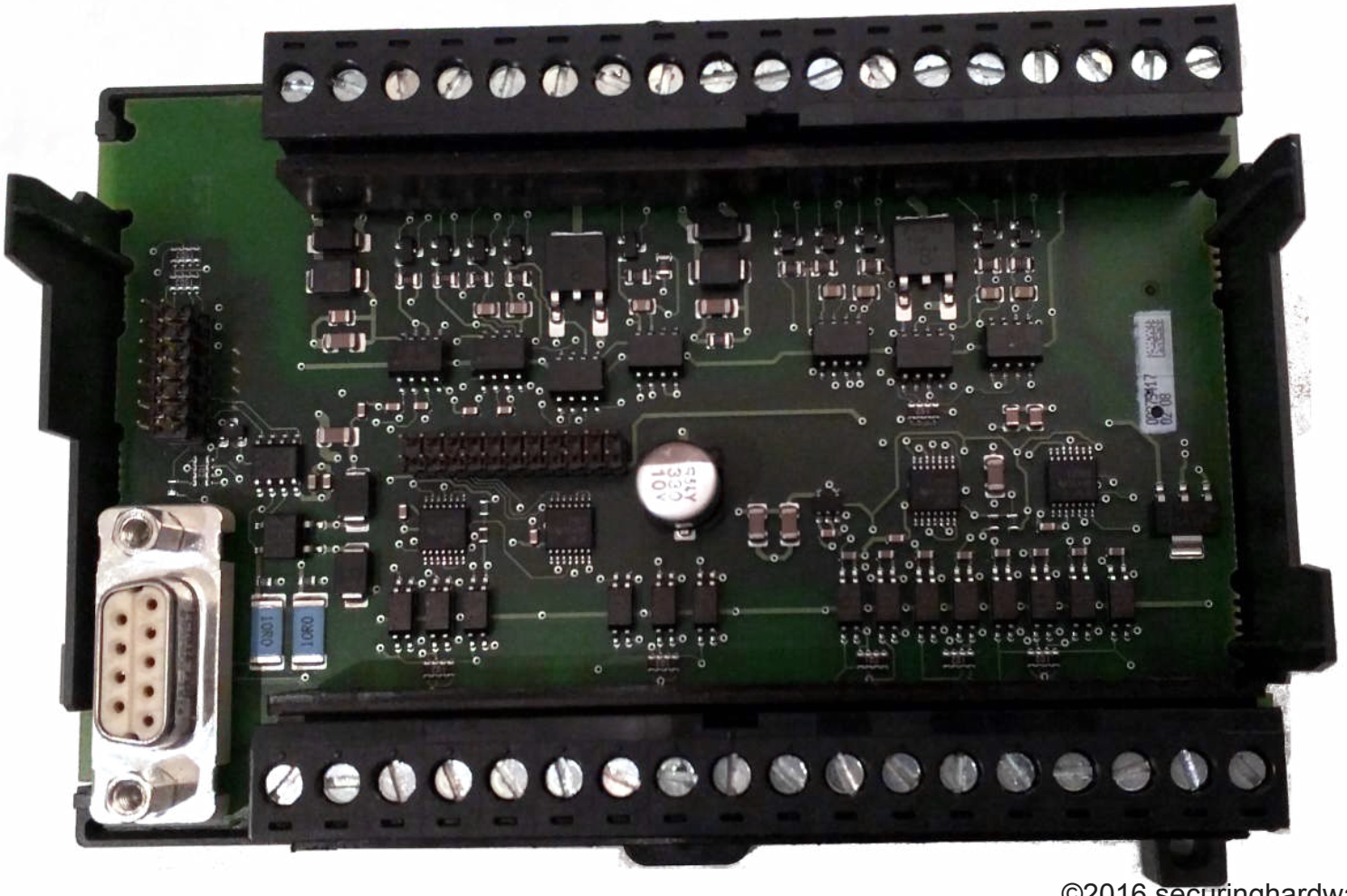


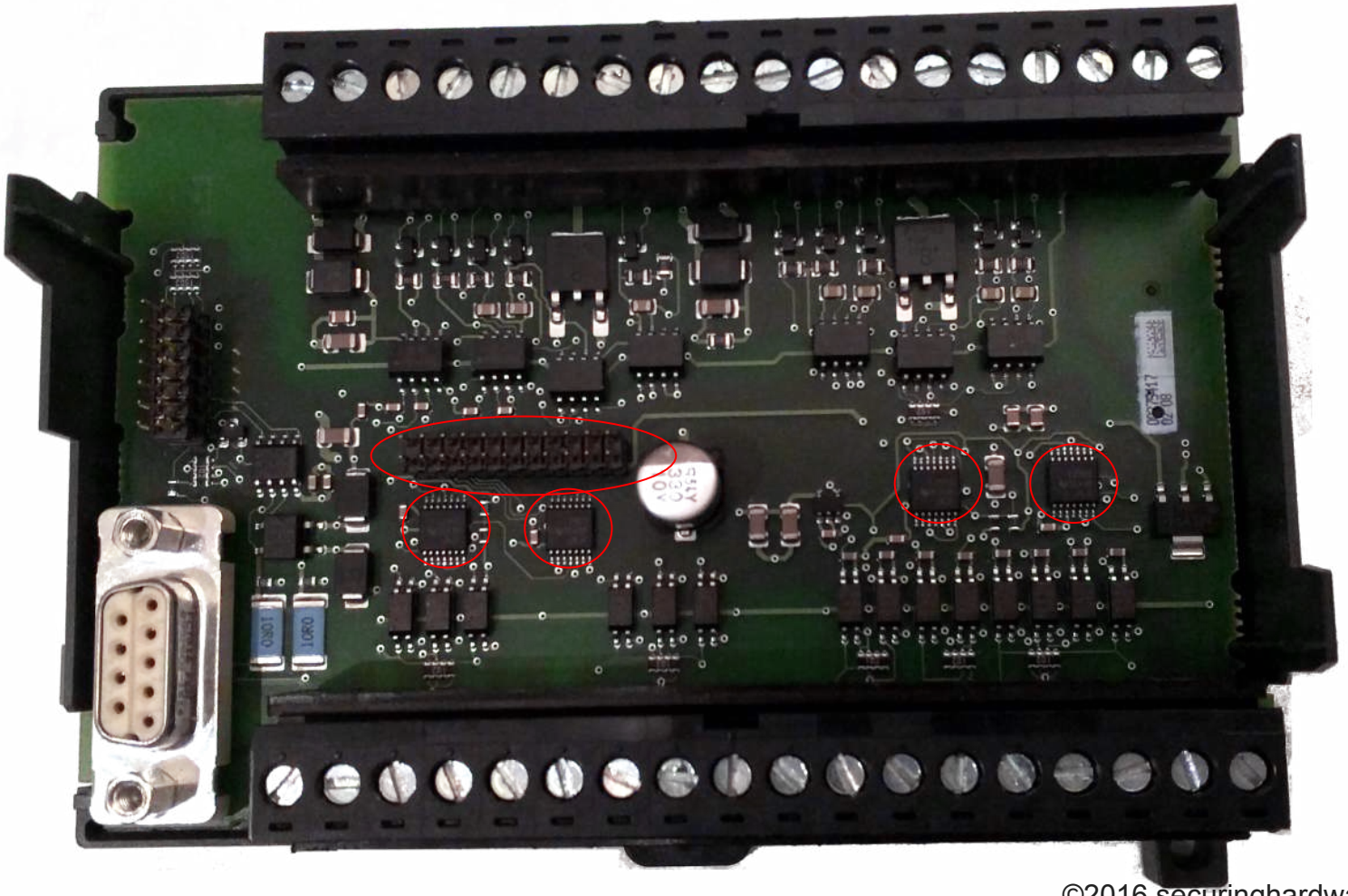


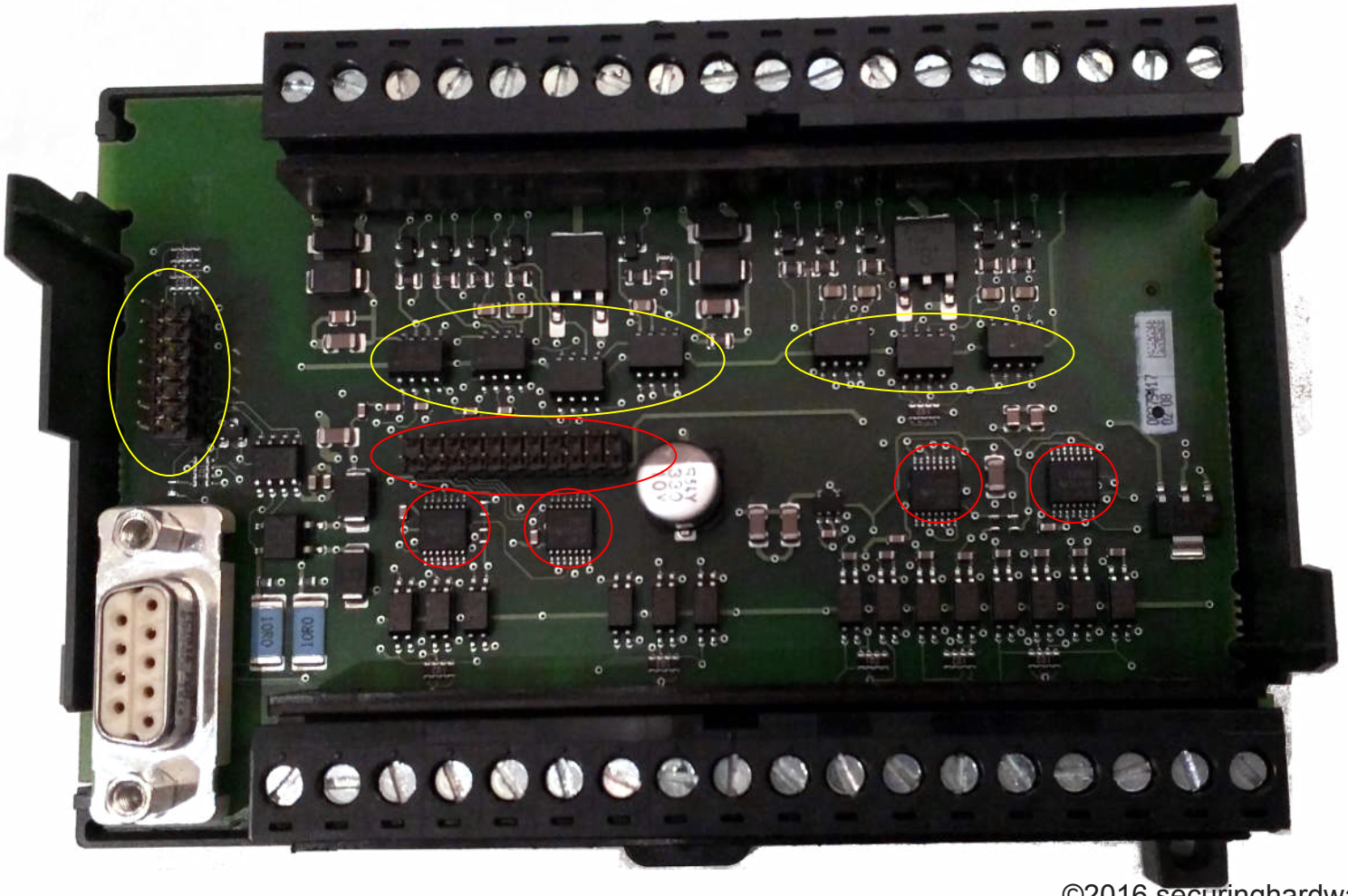


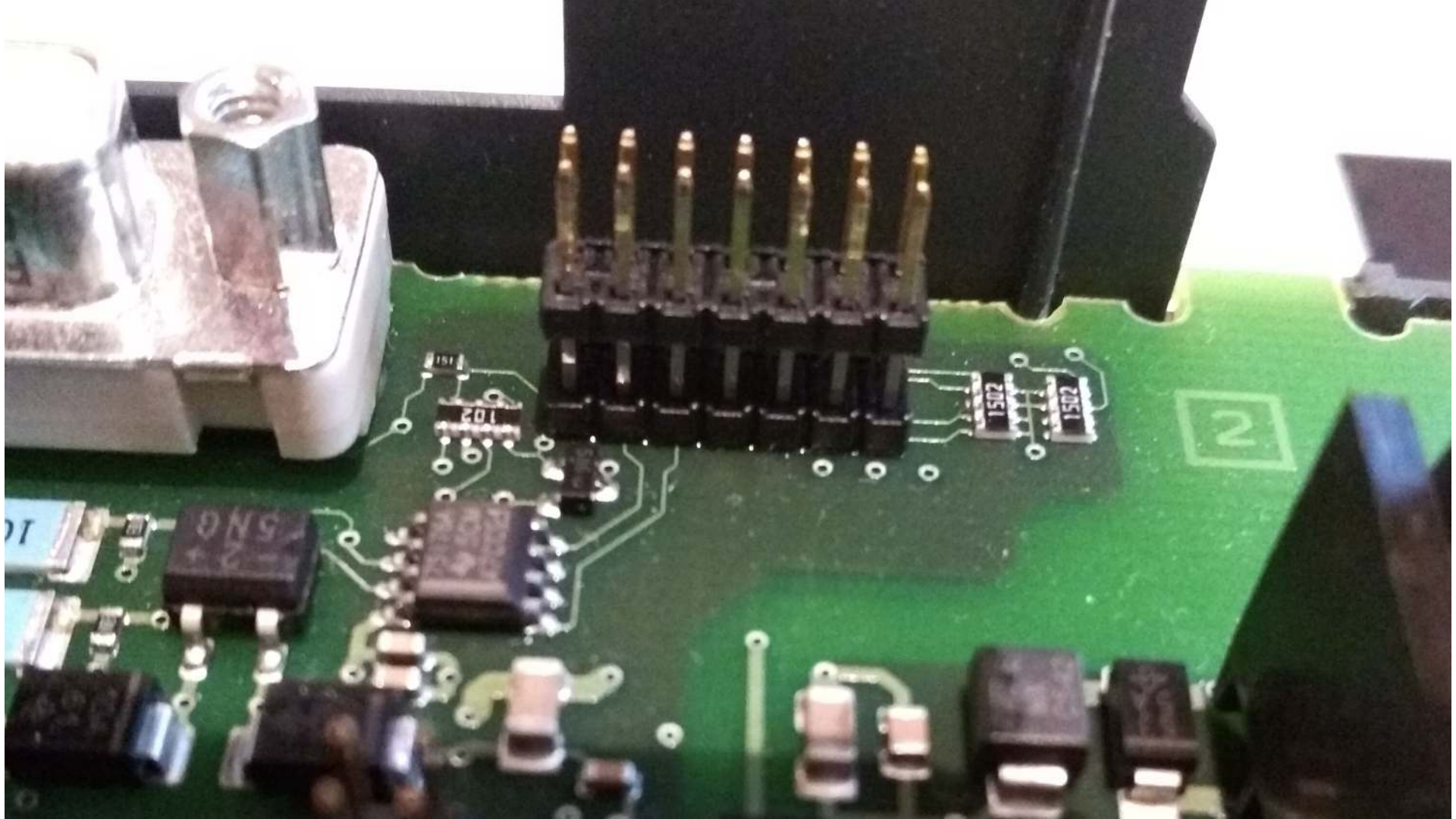


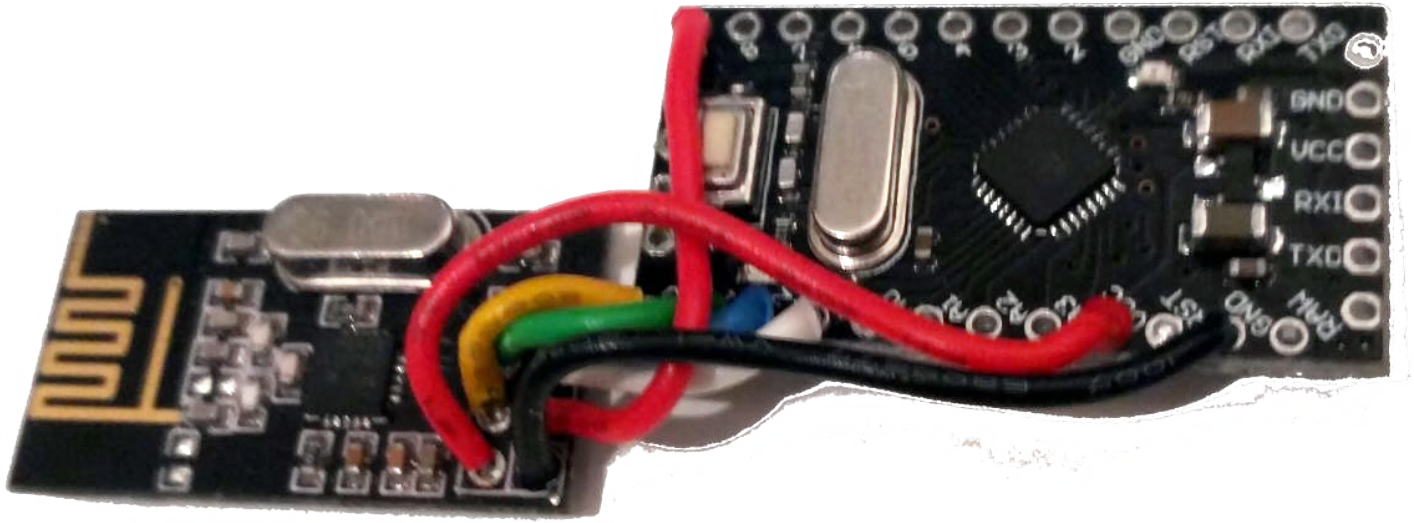


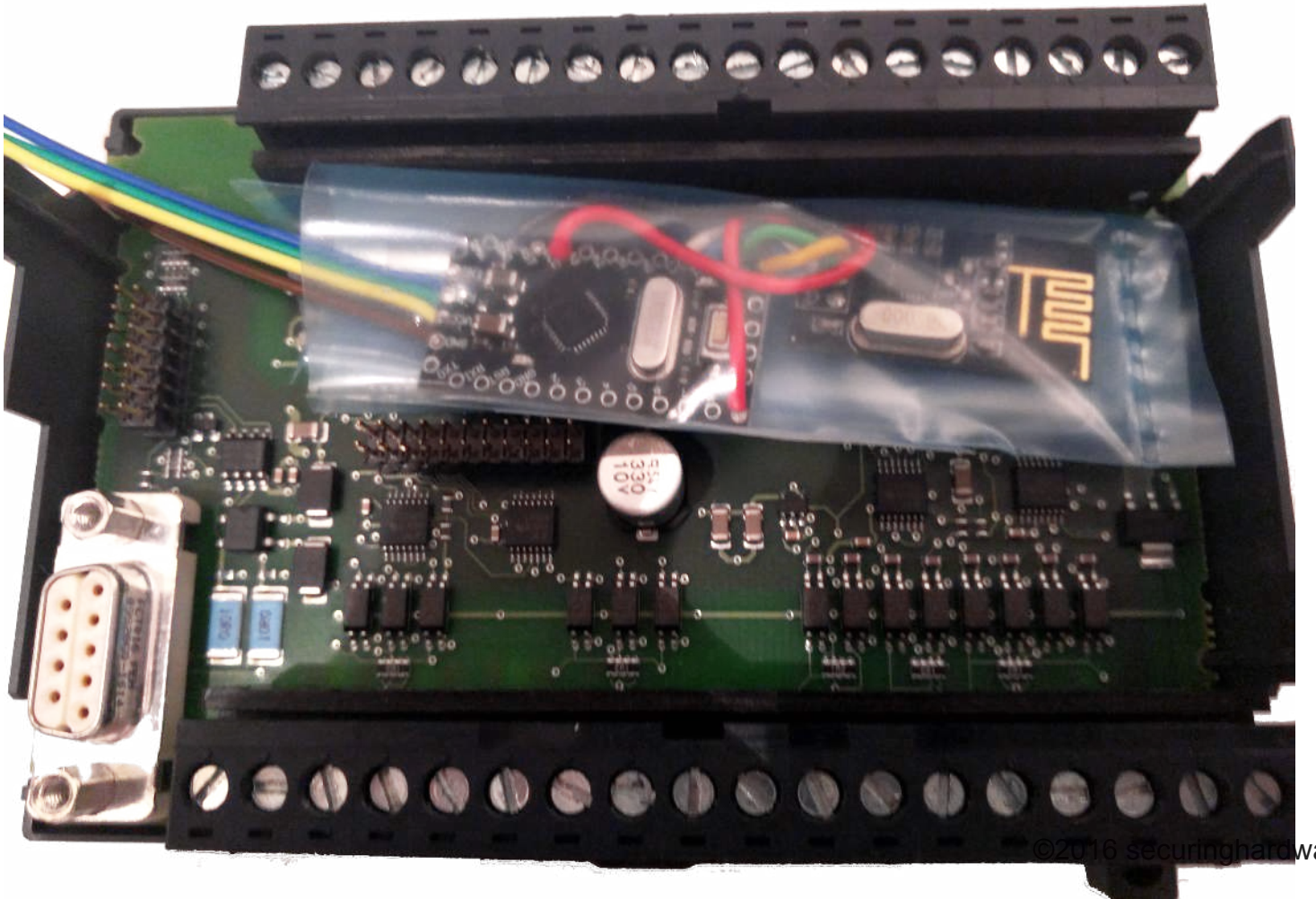


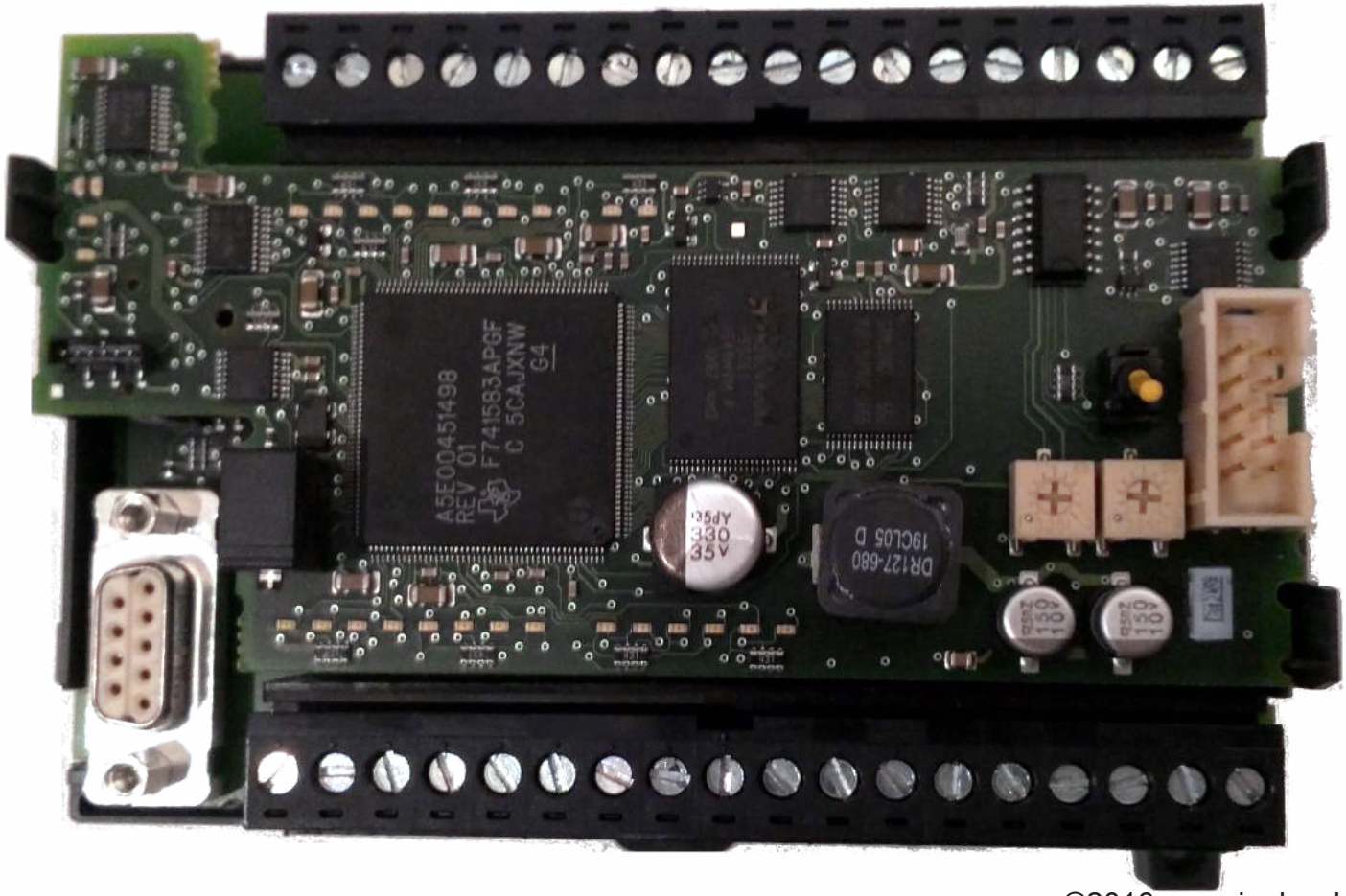


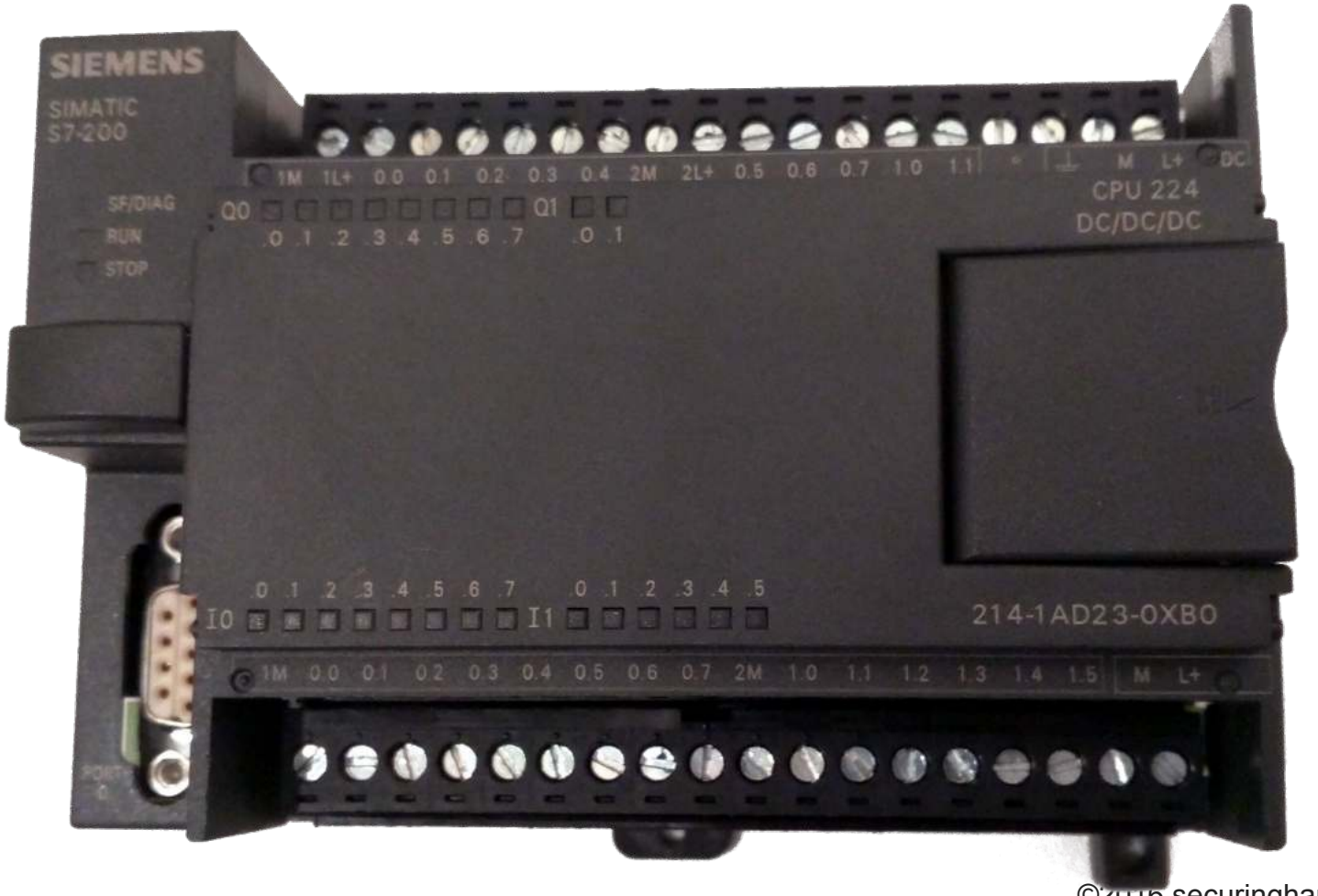



102
150Ω
150Ω
2









So What?

- No software changes
- All the original hardware is in there
- No modification apparent from the outside
- One signal can be wirelessly set/unset

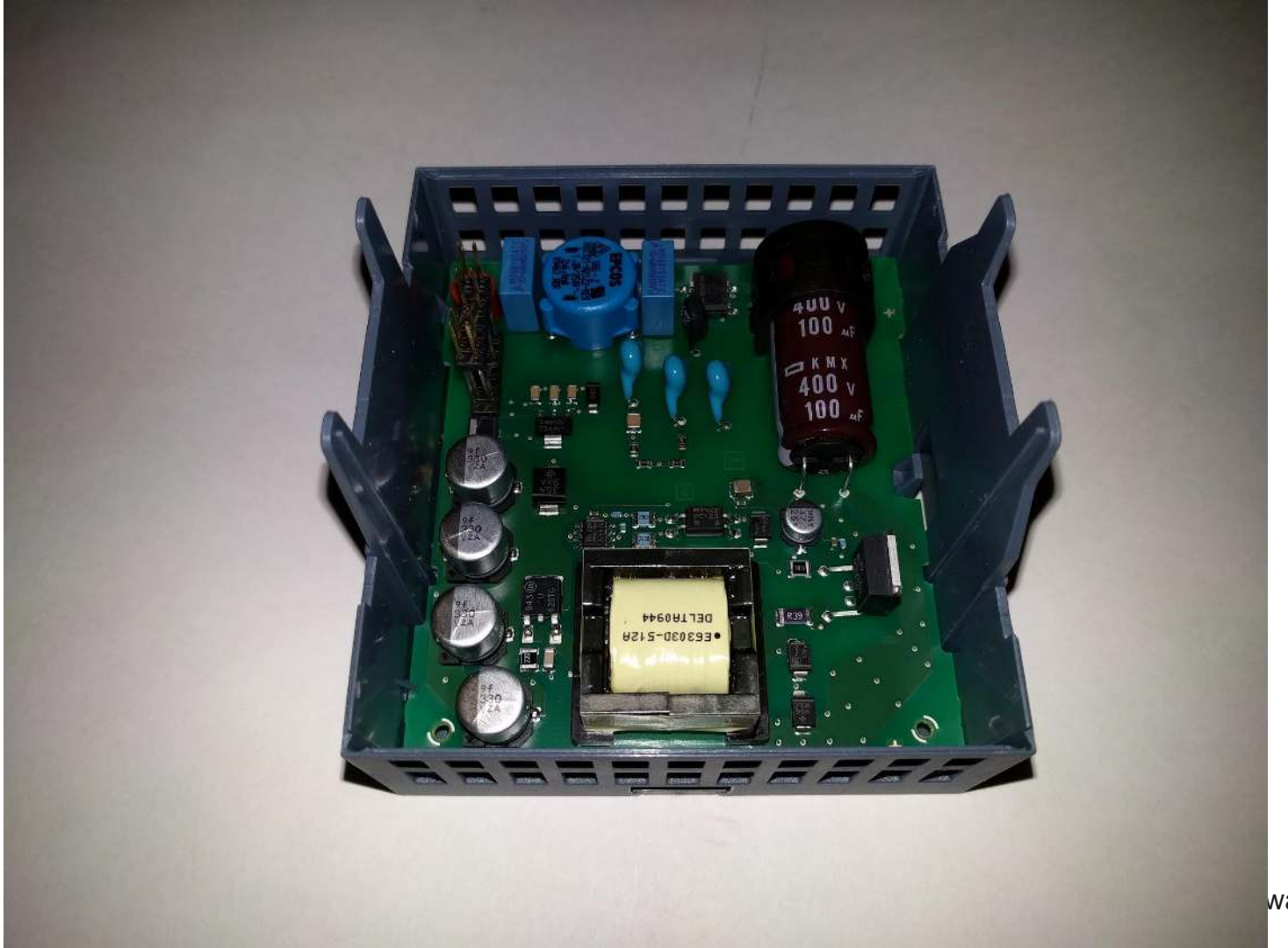
That same header also connects the DB-9 programming port...

we could MITM that too if we wanted....

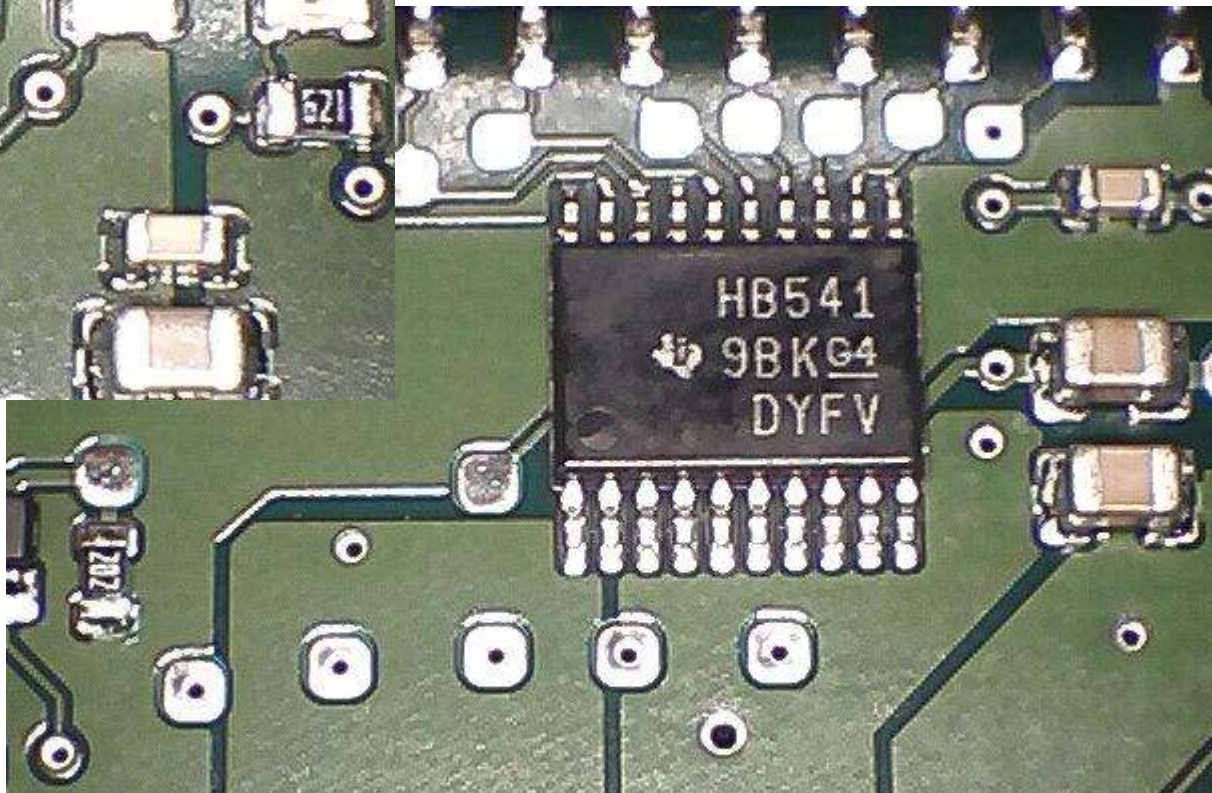
BUT! it proved to be nearly impossible to get a copy of the S7-200 software to get this up and running. It was far cheaper to target a different system:

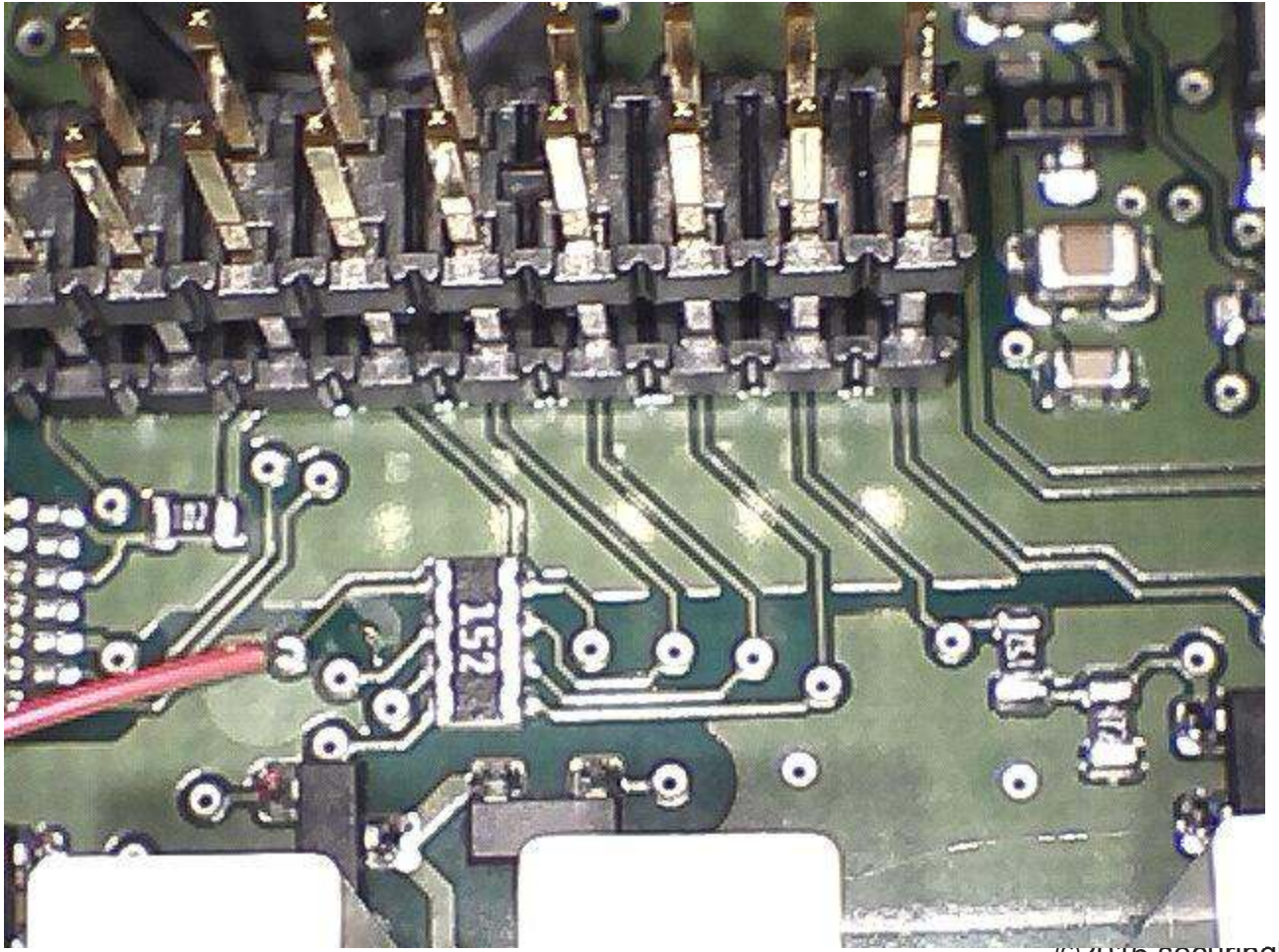

SIEMENS
SIMATIC
S7-1200

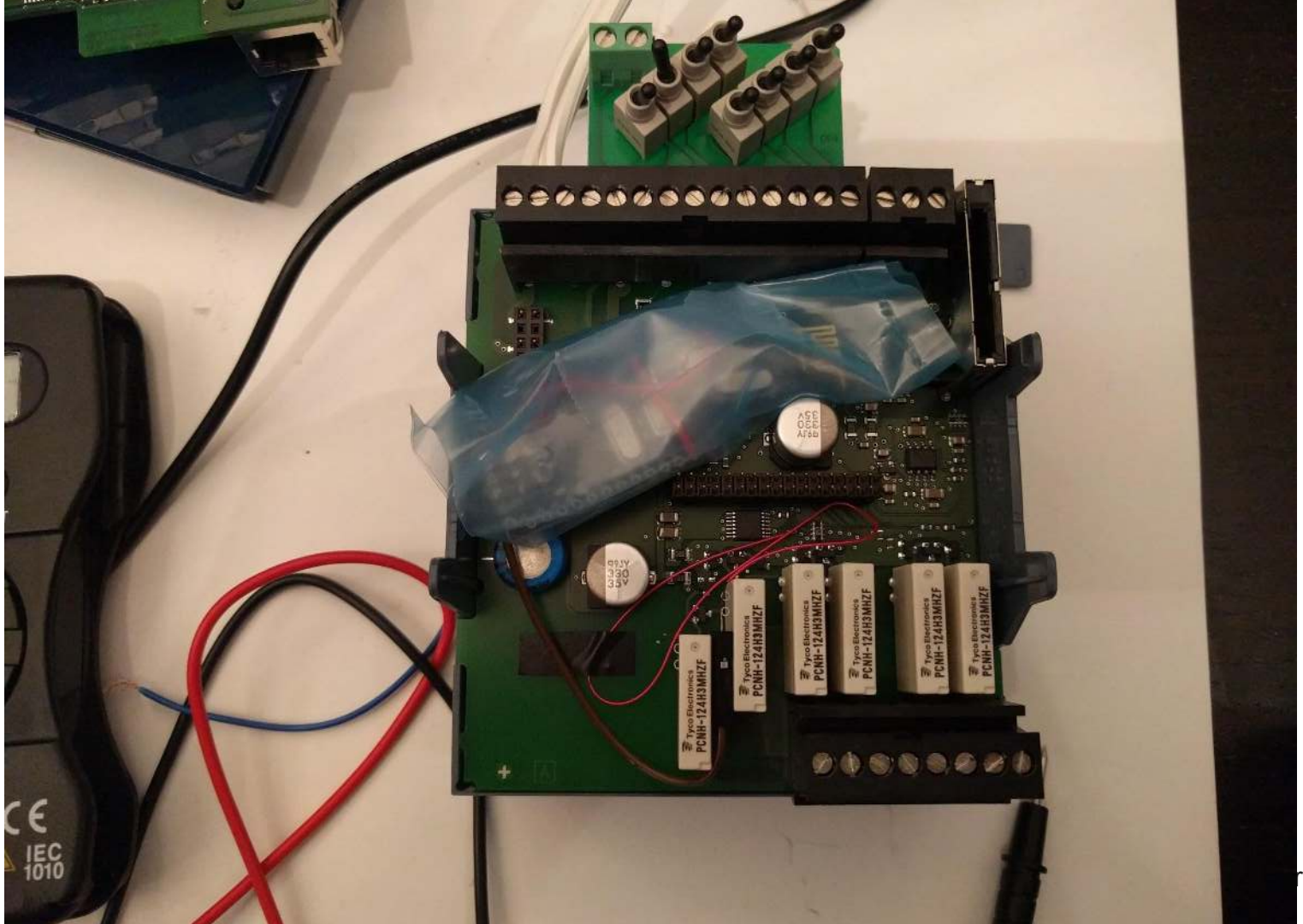

Tyco Electronics
PCNH-124H3MHZF

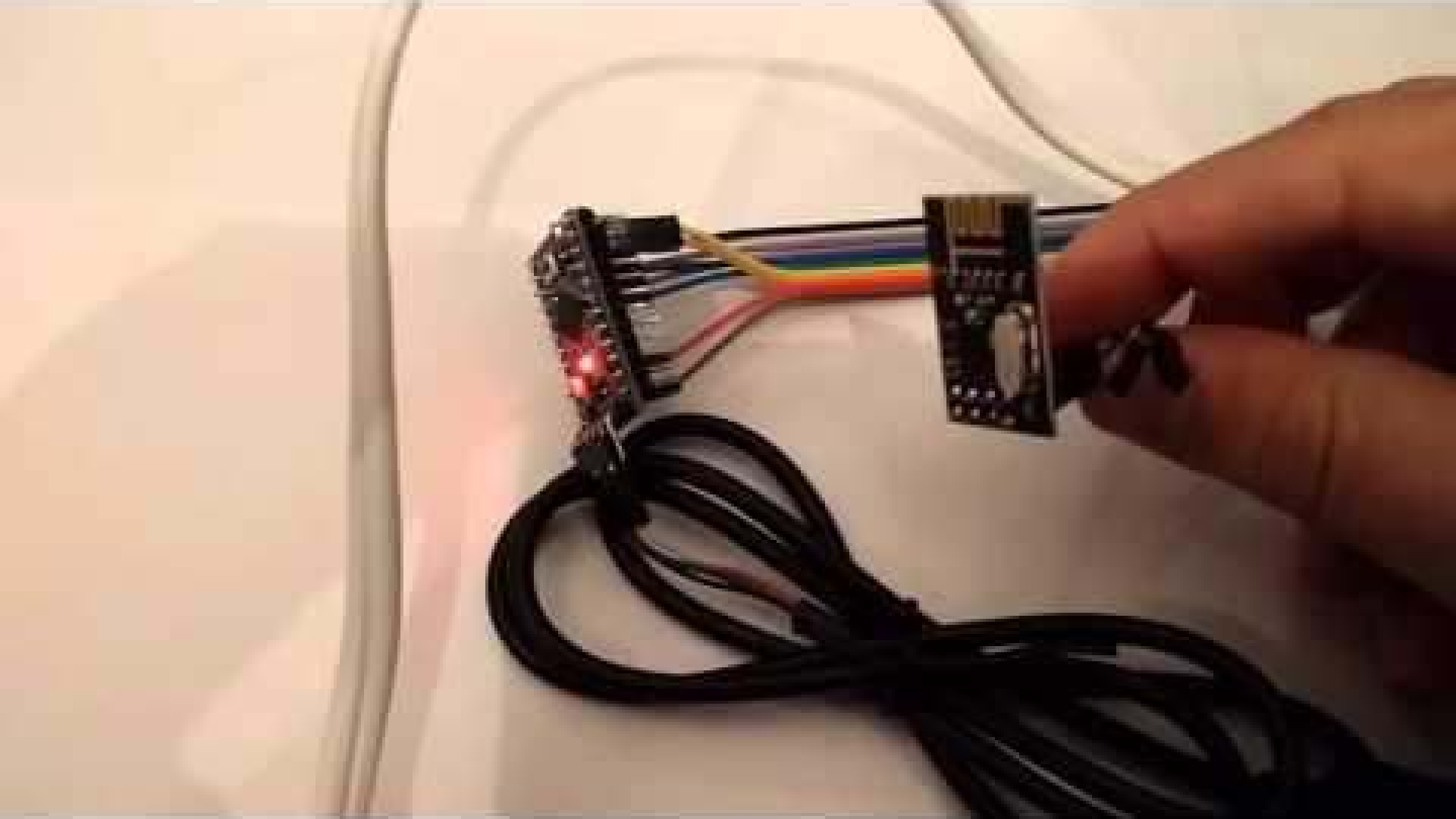

400 V
100 µF
K M X
400 V
100 µF
DELTA0944
•E63030-S12R


621
621


HB541
9BKG4
DYFV
621




Tyco Electronics
PCNR-124H3MHZF
IEC
1010

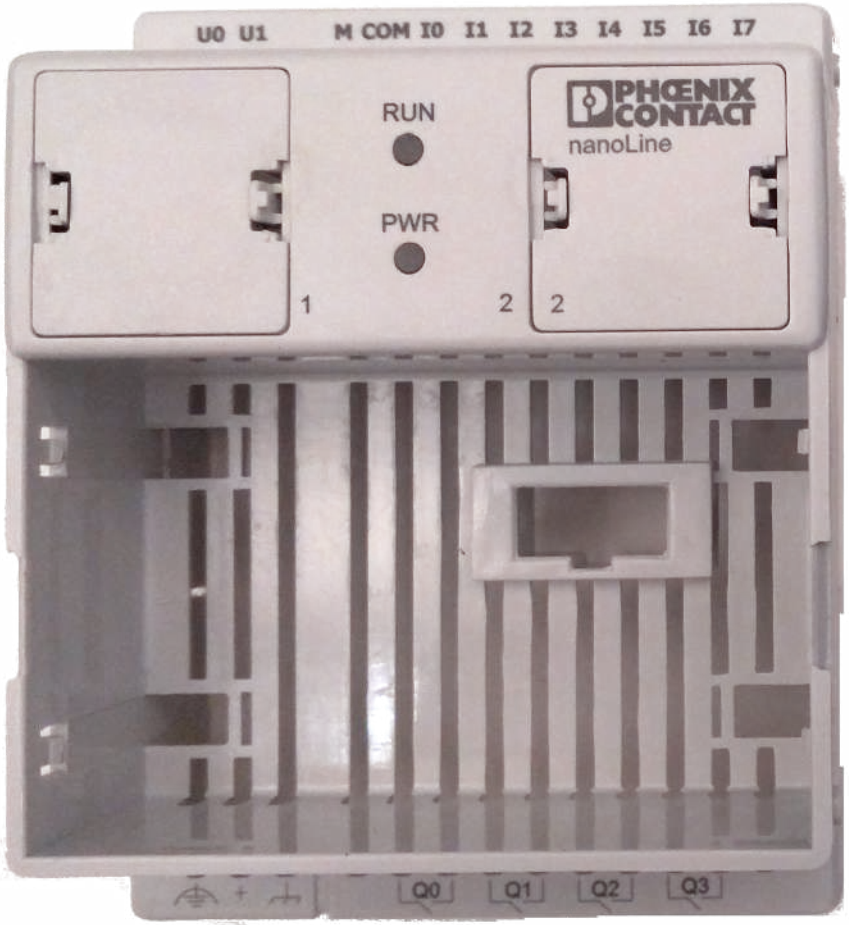


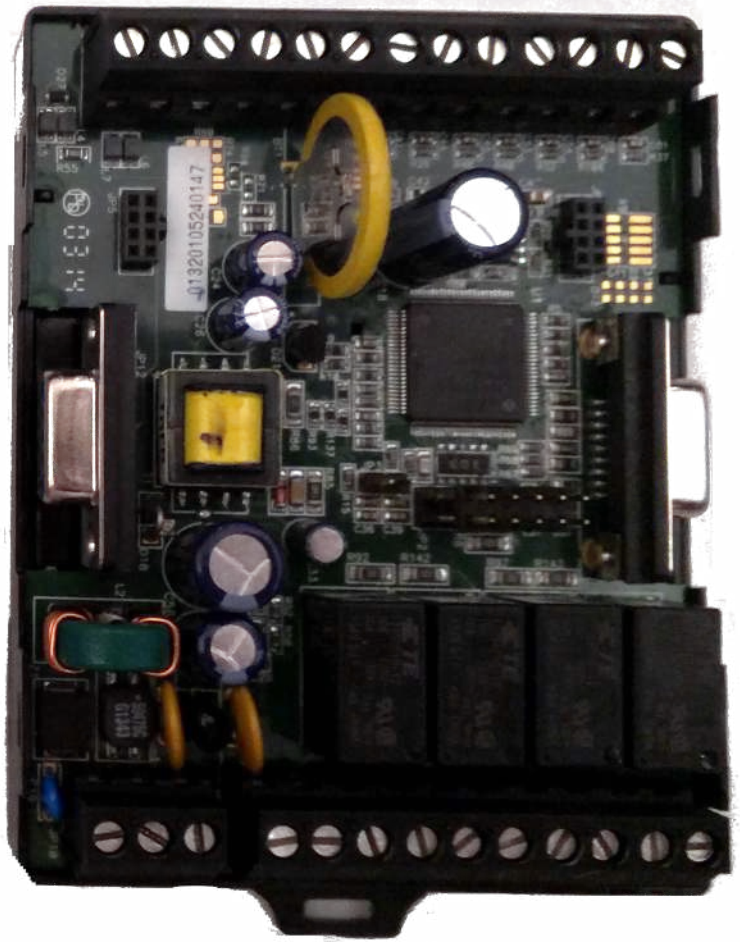
So What?

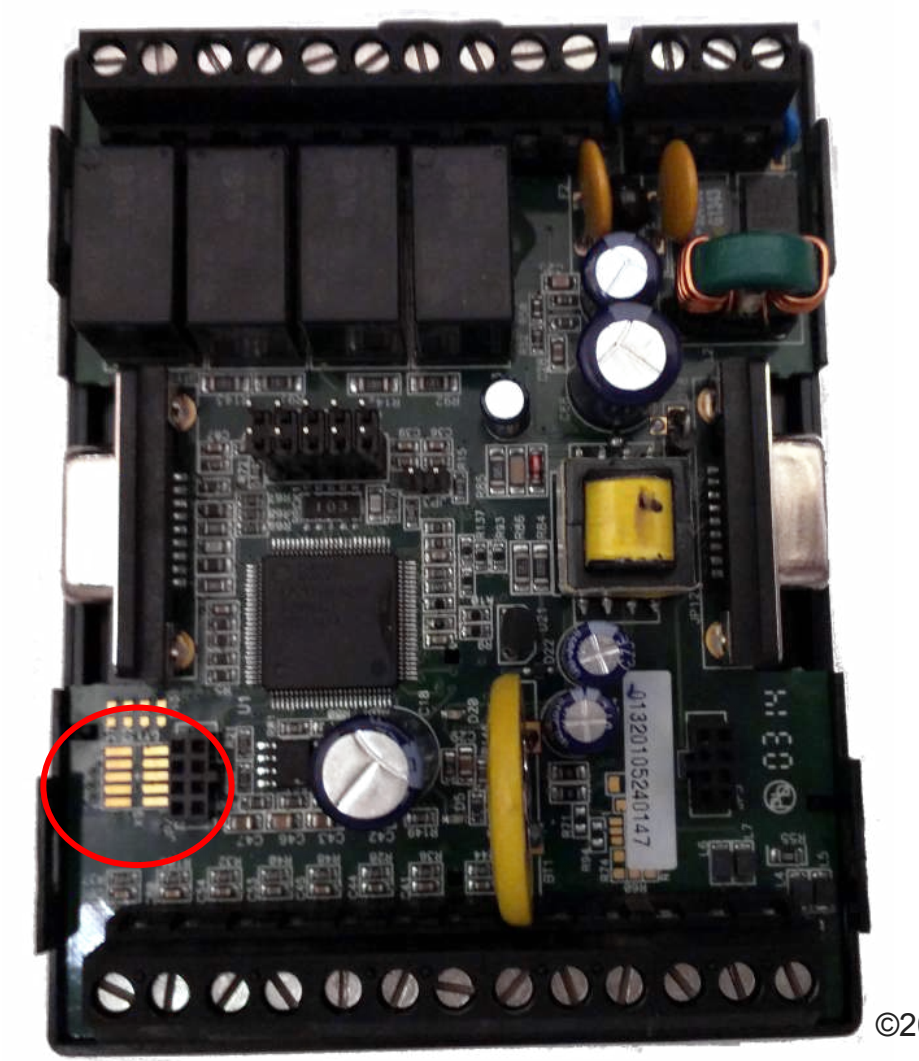
- Again - a very cheap implant
- Mostly off-the-shelf open source code
- Can be retrofitted to most I/O types - if you know where

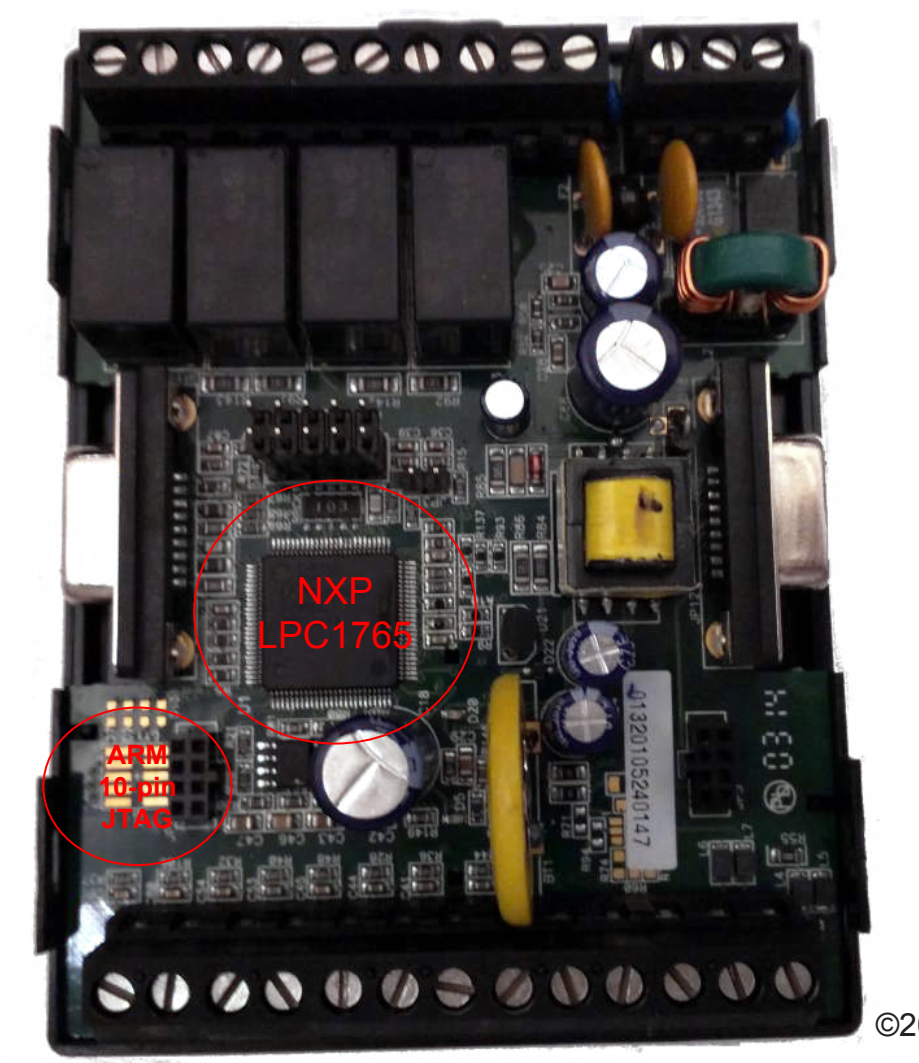
Today’s Implants:

1. Blindly escalate privilege using JTAG
2. Patch kernels via DMA on an embedded device
3. Enable wireless control of an off-the-shelf PLC
4. Hot-plug a malicious PLC expansion
5. BadUSB-style display adapters









8.30.3 Code security (Code Read Protection - CRP)

This feature of the LPC17xx allows user to enable different levels of security in the system so that access to the on-chip flash and use of the JTAG and ISP can be restricted. When needed, CRP is invoked by programming a specific pattern into a dedicated flash location. IAP commands are not affected by the CRP.

There are three levels of the Code Read Protection.

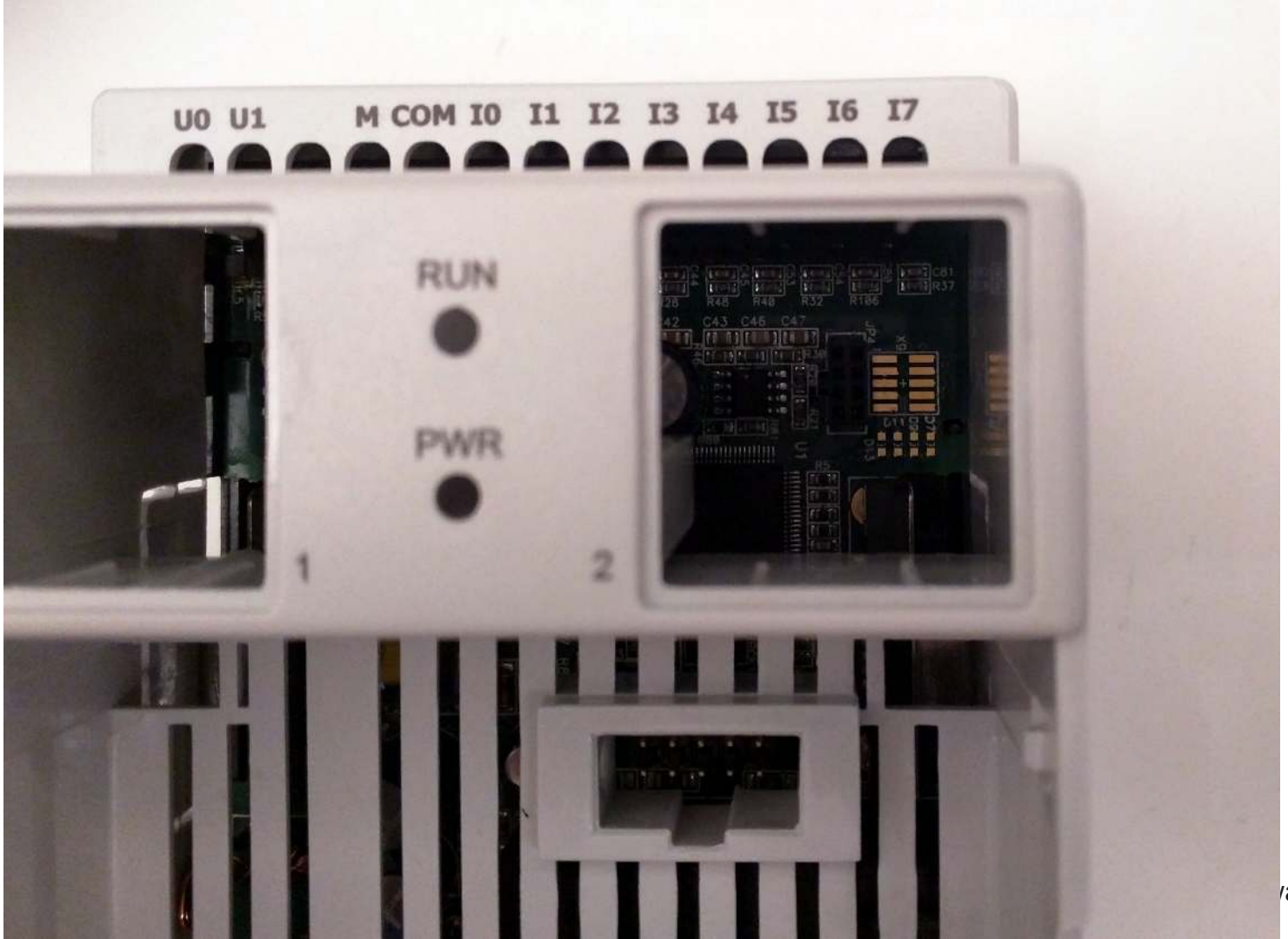
CRP1 disables access to chip via the JTAG and allows partial flash update (excluding flash sector 0) using a limited set of the ISP commands. This mode is useful when CRP is required and flash field updates are needed but all sectors can not be erased.

CRP2 disables access to chip via the JTAG and only allows full flash erase and update using a reduced set of the ISP commands.

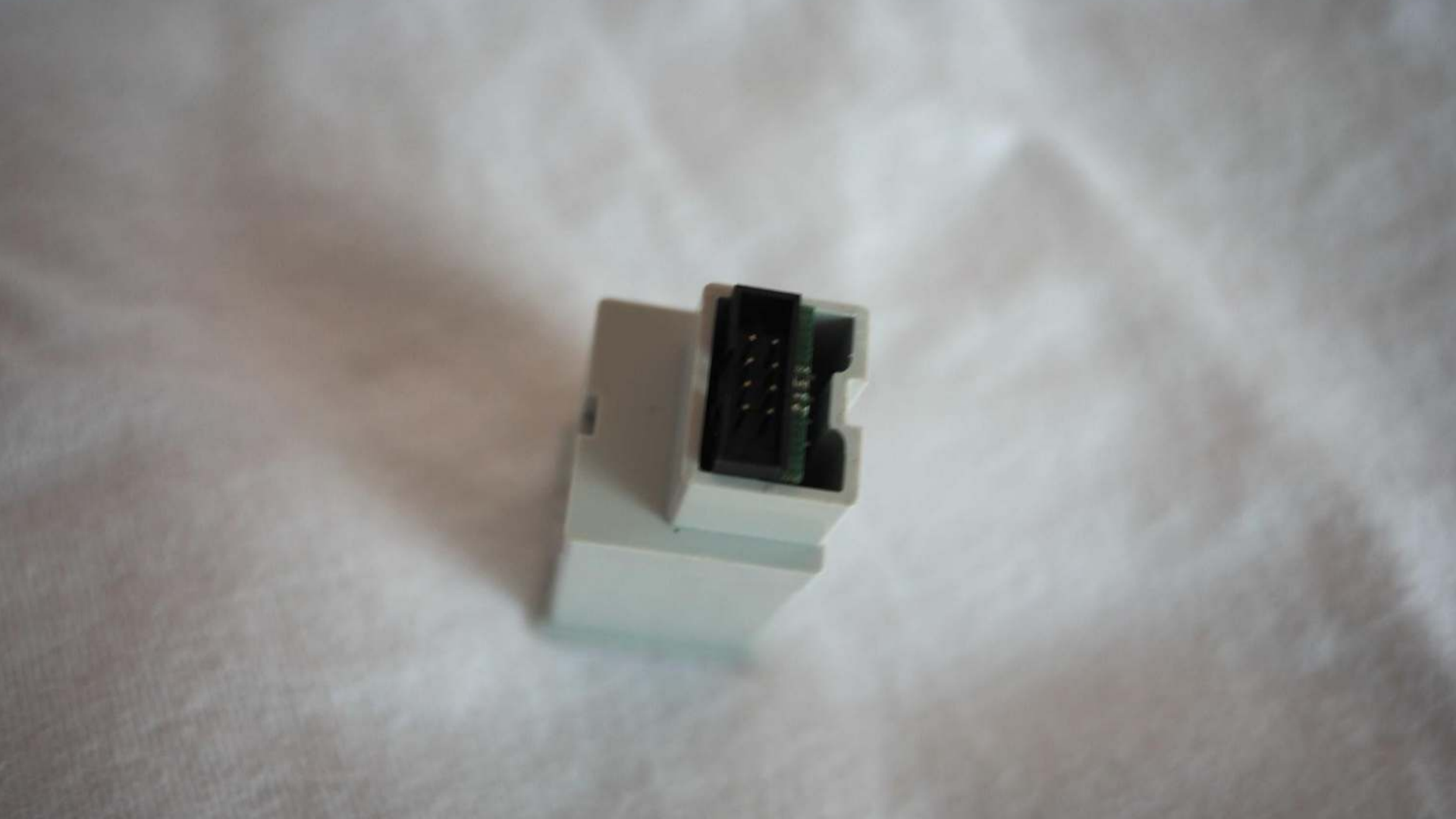
Running an application with level CRP3 selected fully disables any access to chip via the JTAG pins and the ISP. This mode effectively disables ISP override using P2[10] pin, too. It is up to the user's application to provide (if needed) flash update mechanism using IAP calls or call reinvoke ISP command to enable flash update via UART0.

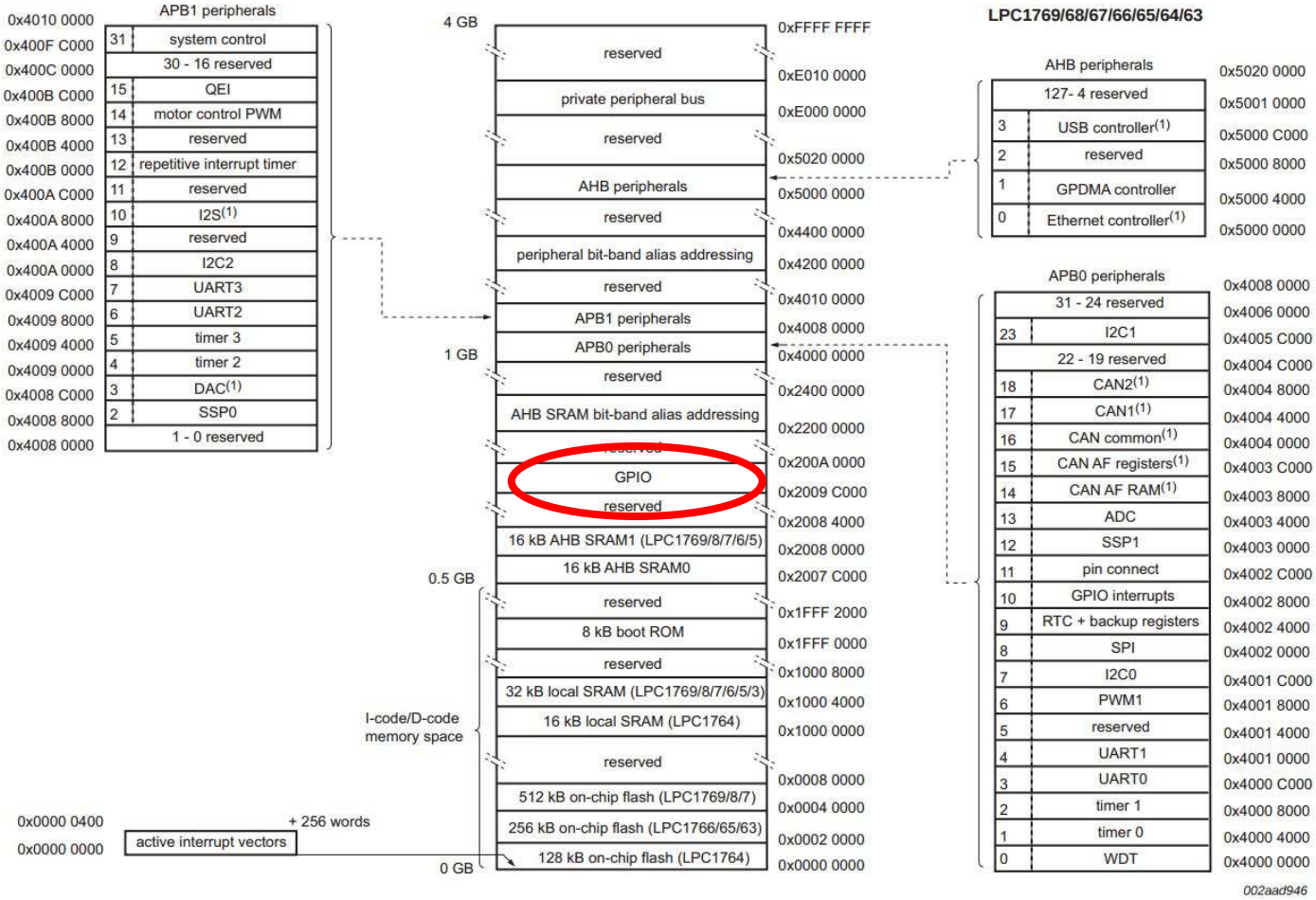
CAUTION

If level three Code Read Protection (CRP3) is selected, no future factory testing can be performed on the device.




USB





(1) Not available on all parts. See Table 2.

Fig 5. LPC17xx memory map

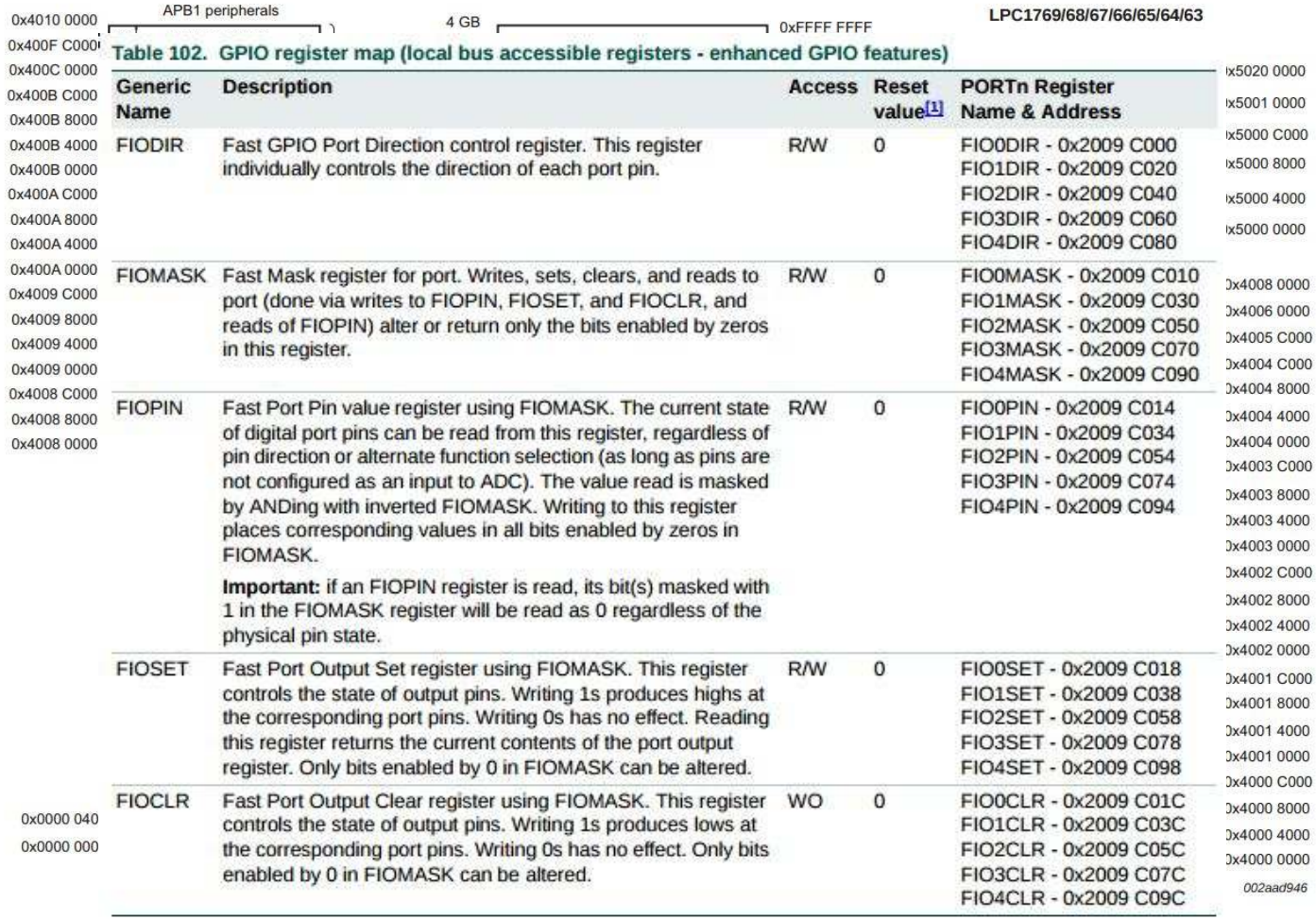


Table 102. GPIO register map (local bus accessible registers - enhanced GPIO features)

Generic Name	Description	Access	Reset value[1]	PORTn Register Name & Address
FIODIR	Fast GPIO Port Direction control register. This register individually controls the direction of each port pin.	R/W	0	FIO0DIR - 0x2009 C000 FIO1DIR - 0x2009 C020 FIO2DIR - 0x2009 C040 FIO3DIR - 0x2009 C060 FIO4DIR - 0x2009 C080
FIOMASK	Fast Mask register for port. Writes, sets, clears, and reads to port (done via writes to FIOPIN, FIOSET, and FIOCLR, and reads of FIOPIN) alter or return only the bits enabled by zeros in this register.	R/W	0	FIO0MASK - 0x2009 C010 FIO1MASK - 0x2009 C030 FIO2MASK - 0x2009 C050 FIO3MASK - 0x2009 C070 FIO4MASK - 0x2009 C090
FIOPIN	Fast Port Pin value register using FIOMASK. The current state of digital port pins can be read from this register, regardless of pin direction or alternate function selection (as long as pins are not configured as an input to ADC). The value read is masked by ANDing with inverted FIOMASK. Writing to this register places corresponding values in all bits enabled by zeros in FIOMASK. **Important:** if an FIOPIN register is read, its bit(s) masked with 1 in the FIOMASK register will be read as 0 regardless of the physical pin state.	R/W	0	FIO0PIN - 0x2009 C014 FIO1PIN - 0x2009 C034 FIO2PIN - 0x2009 C054 FIO3PIN - 0x2009 C074 FIO4PIN - 0x2009 C094
FIOSET	Fast Port Output Set register using FIOMASK. This register controls the state of output pins. Writing 1s produces highs at the corresponding port pins. Writing 0s has no effect. Reading this register returns the current contents of the port output register. Only bits enabled by 0 in FIOMASK can be altered.	R/W	0	FIO0SET - 0x2009 C018 FIO1SET - 0x2009 C038 FIO2SET - 0x2009 C058 FIO3SET - 0x2009 C078 FIO4SET - 0x2009 C098
FIOCLR	Fast Port Output Clear register using FIOMASK. This register controls the state of output pins. Writing 1s produces lows at the corresponding port pins. Writing 0s has no effect. Only bits enabled by 0 in FIOMASK can be altered.	WO	0	FIO0CLR - 0x2009 C01C FIO1CLR - 0x2009 C03C FIO2CLR - 0x2009 C05C FIO3CLR - 0x2009 C07C FIO4CLR - 0x2009 C09C

(1) Not available on all parts. See Table 2.

Fig 5. LPC17xx memory map

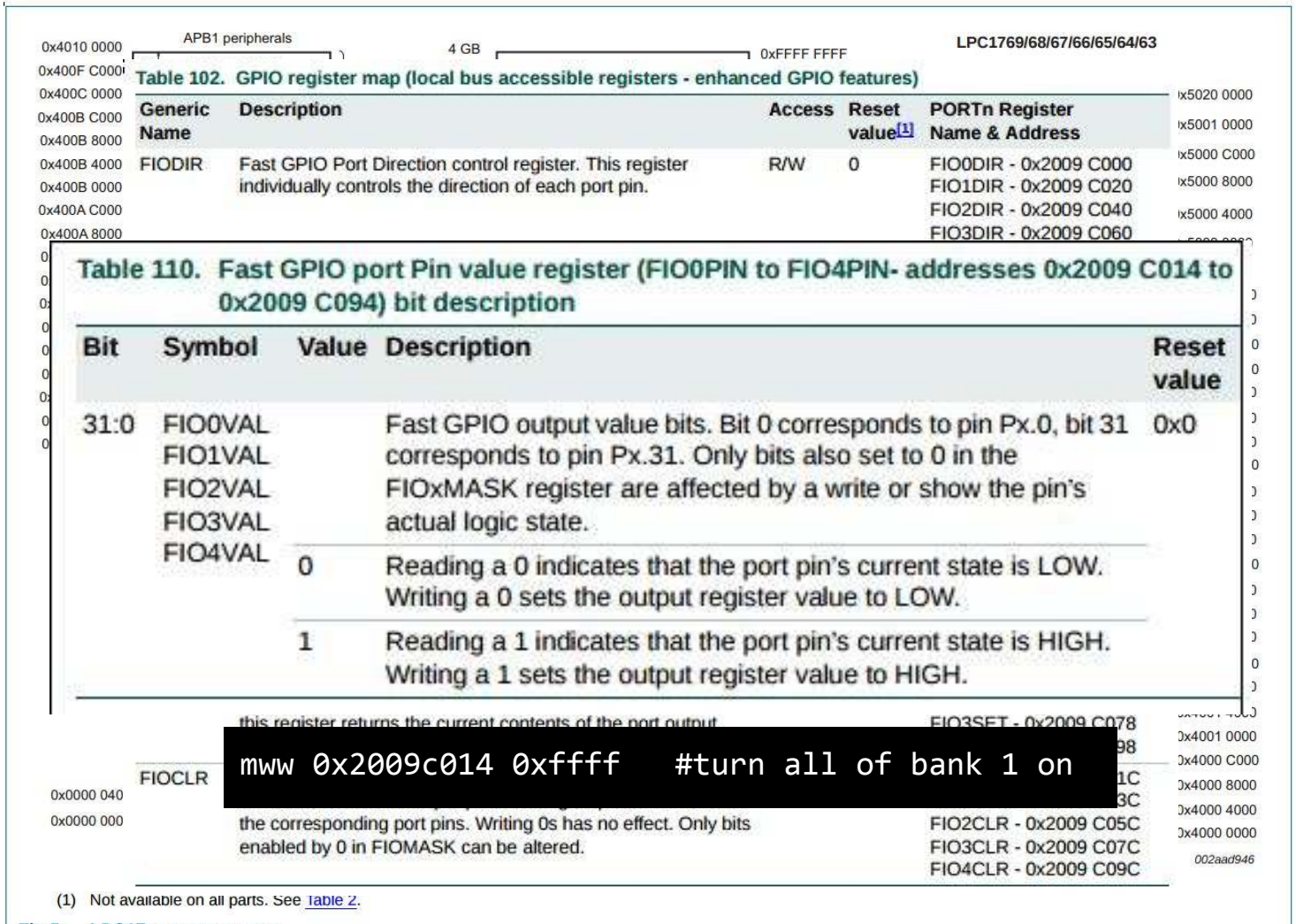
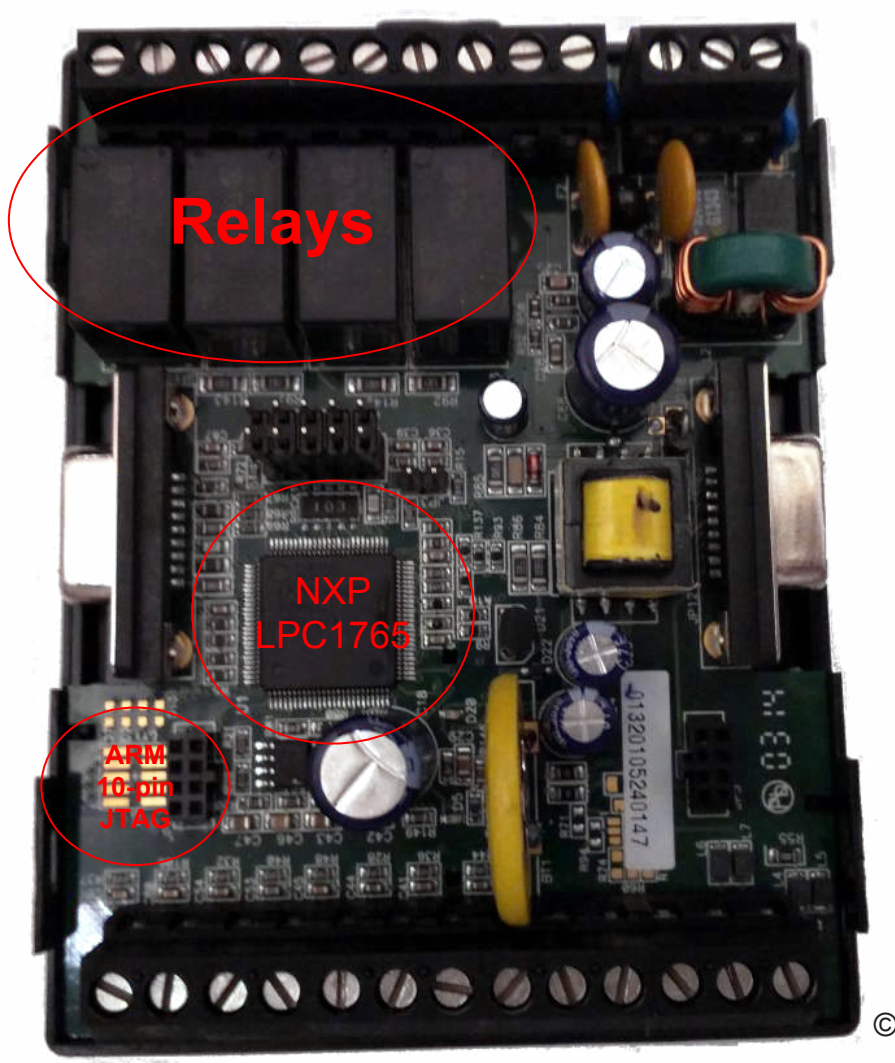


Fig 5. LPC17xx memory map




OL
WAVETEK
25XT
OK

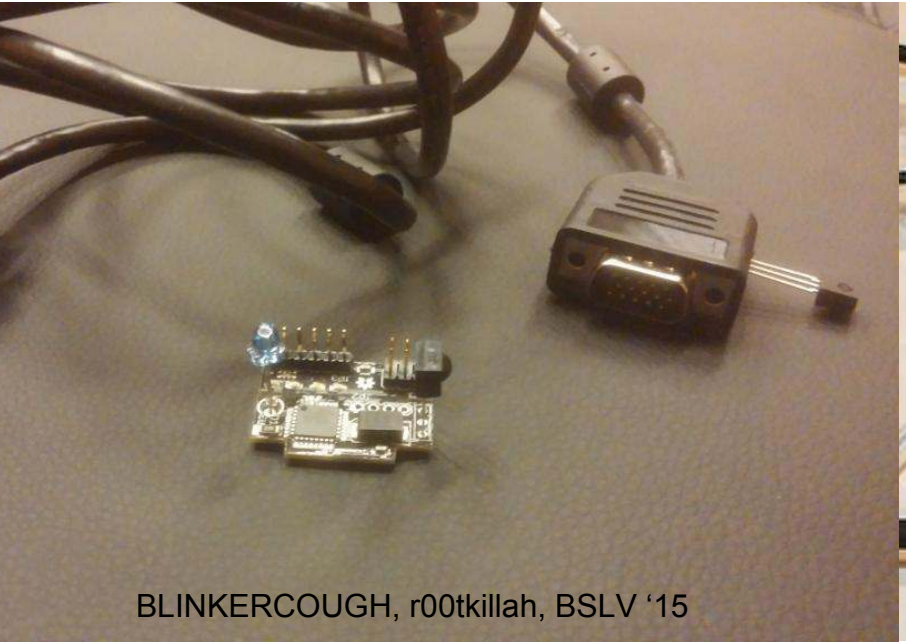
So What?

- JTAG could be disabled (but it’s usually not)
- Test headers are quick and easy to access
- Can be used on a unit IN OPERATION
- When you get JTAG working - you win.

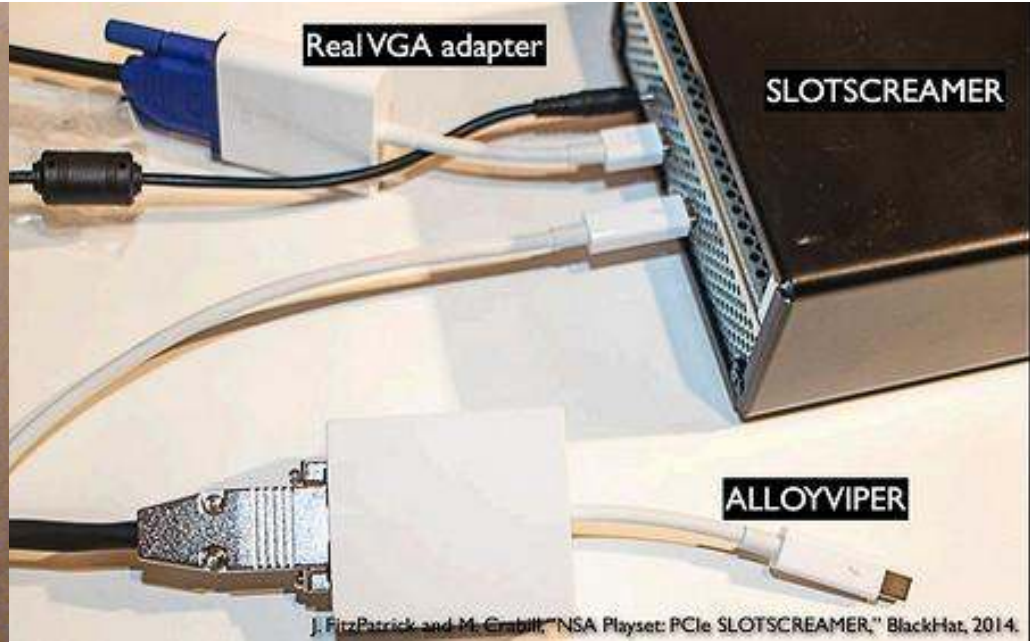
Today’s Implants:

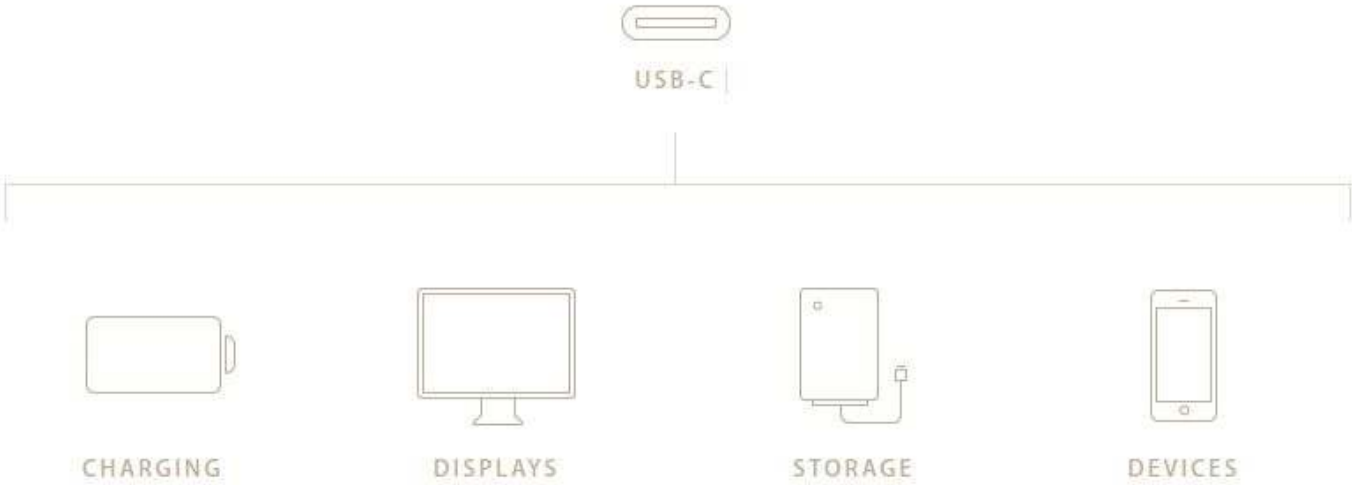
1. Blindly escalate privilege using JTAG
2. Patch kernels via DMA on an embedded device
3. Enable wireless control of an off-the-shelf PLC
4. Hot-plug a malicious PLC expansion
5. BadUSB-style display adapters

Evil Maid --> Evil Conference Projector?



BLINKERCOUGH, r00tkillah, BSLV ‘15





The capability to connect to everything you need.

We gave a lot of consideration to the way MacBook connects to peripherals and power. We chose USB-C for its compact design and versatility. This single port lets you connect your charger; HDMI, DisplayPort, and VGA displays; USB devices like external drives; and your iPhone or iPad. All of which goes to show that sometimes less really is more.

The do-everything adapter

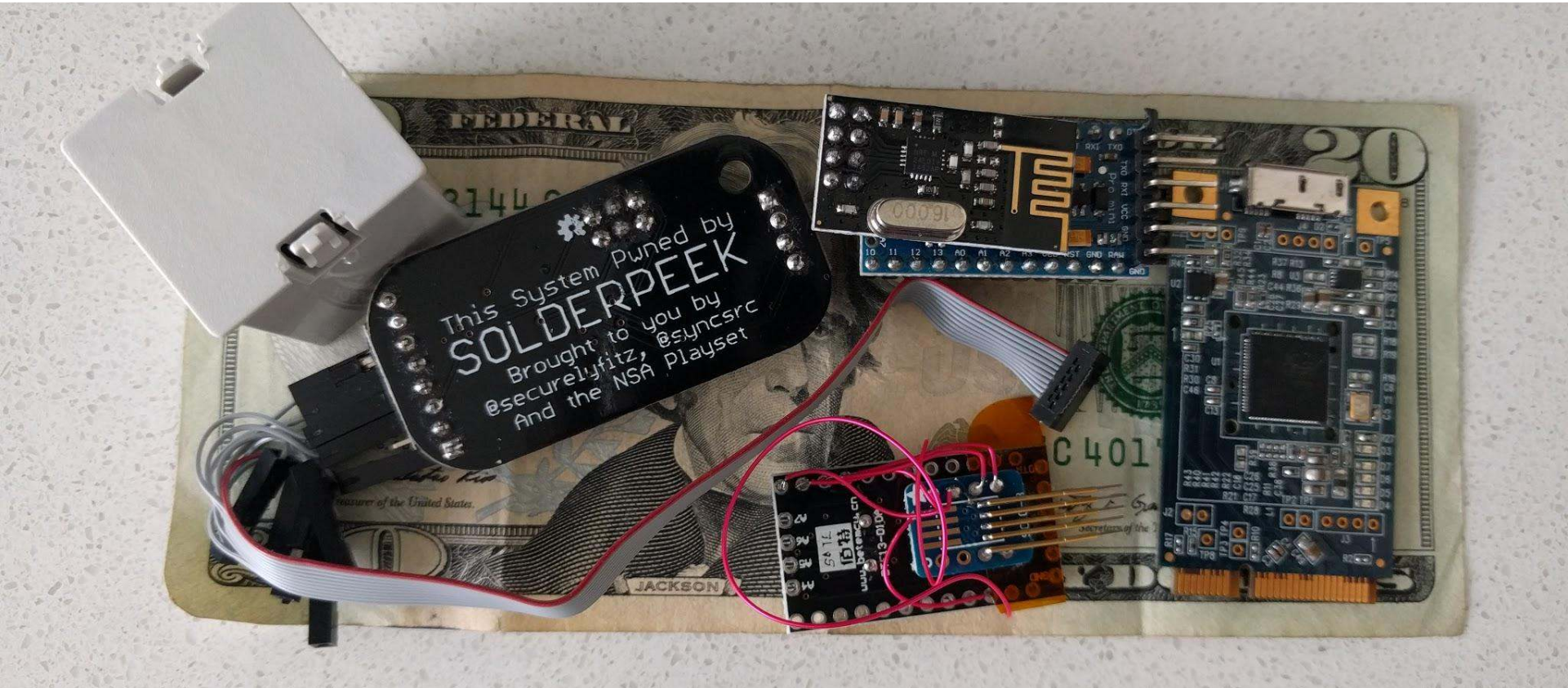
So What?

- When we have only one port - that port can do ANYTHING
- Hardware isn’t just hardware anymore:
  - There are no passive devices
  - There is no reasonable way for a user to know what a device does

Today’s Implants:

1. Blindly escalate privilege using JTAG
2. Patch kernels via DMA on an embedded device
3. Enable wireless control of an off-the-shelf PLC
4. Hot-plug a malicious PLC expansion
5. BadUSB-style display adapters

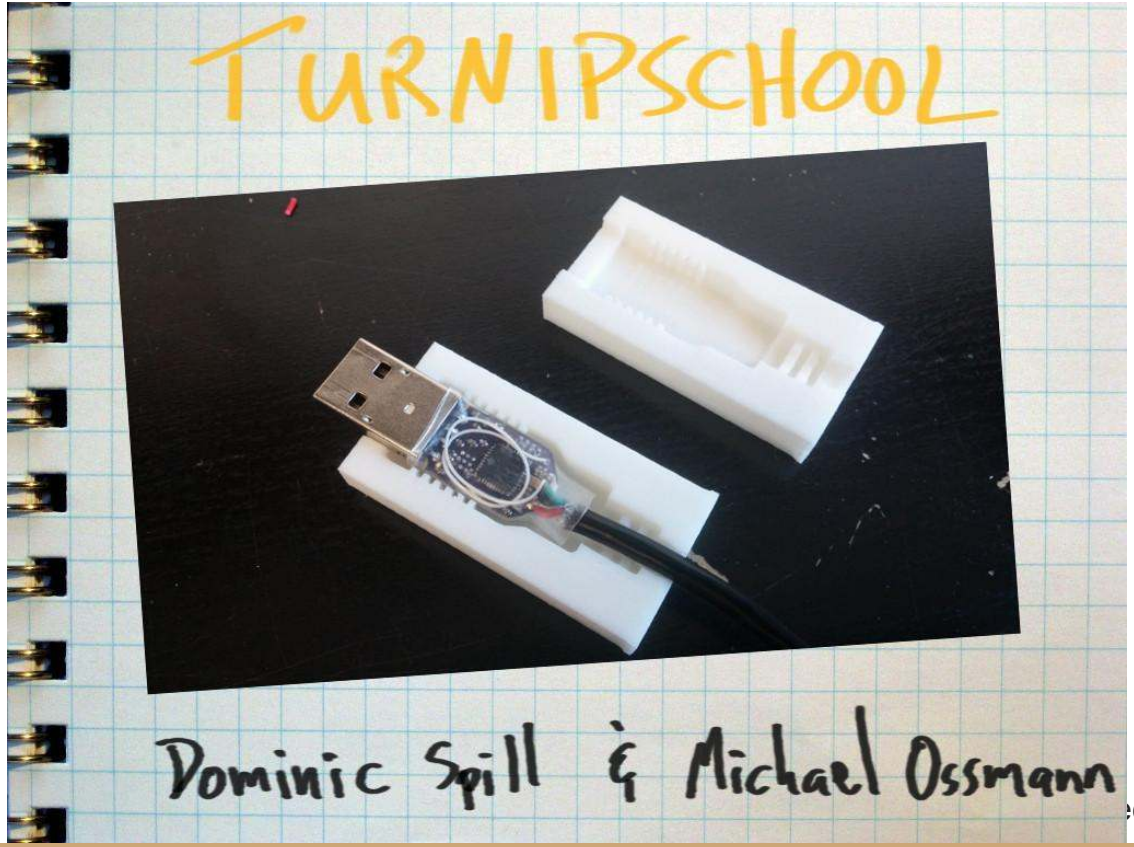
Today’s Implants:



Today’s Implants:

1. Cost between $5 and $75
2. Can be built by hobbyists
3. Weigh less than an ounce
4. Can be customized to many different targets
5. Barely scratch the surface of what’s possible

Bad USB --> Bad USB Cable?



Takeaways

- Hardware is cheap
- Standard interfaces really are standard
- Automating small devices is trivial
- If you can’t trust the hardware, you can’t trust the software
- If you can control the hardware, you can control the software

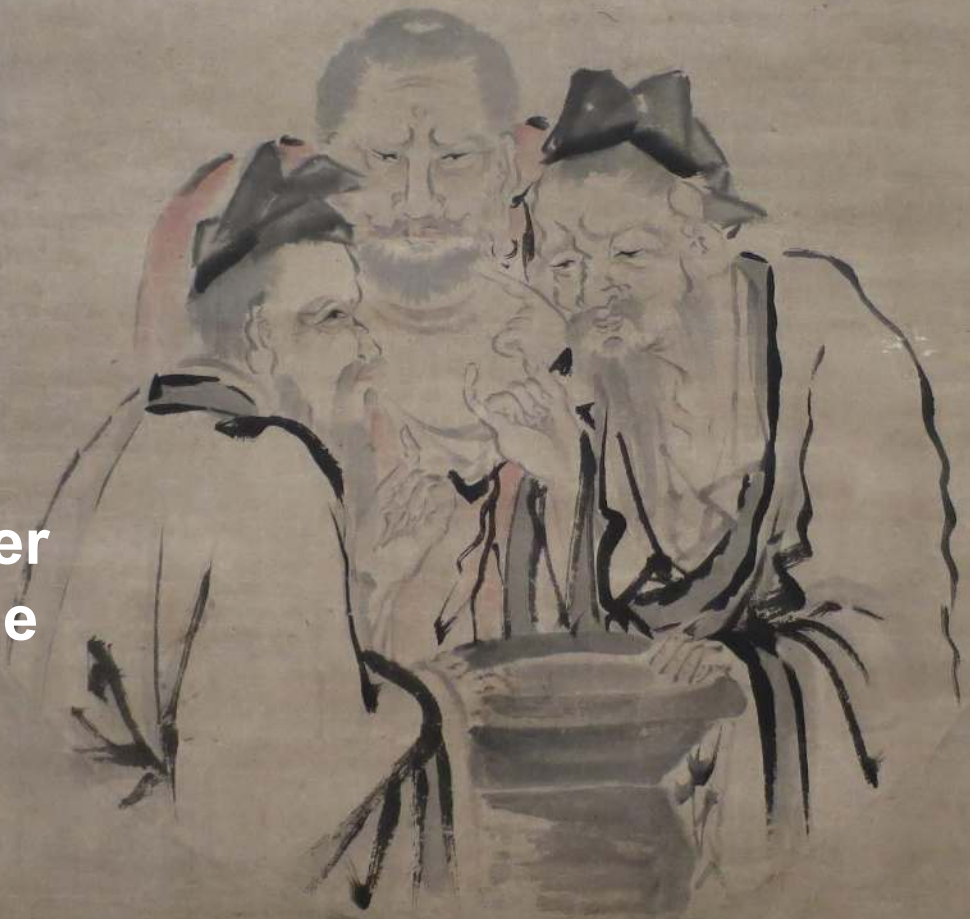
The Three Vinegar Tasters

Confucius

Tastes Sour

Vinegar is Polluted Wine

Conferencegoer who missed the open bar



Buddha

Tastes Bitter

Life is pain and suffering

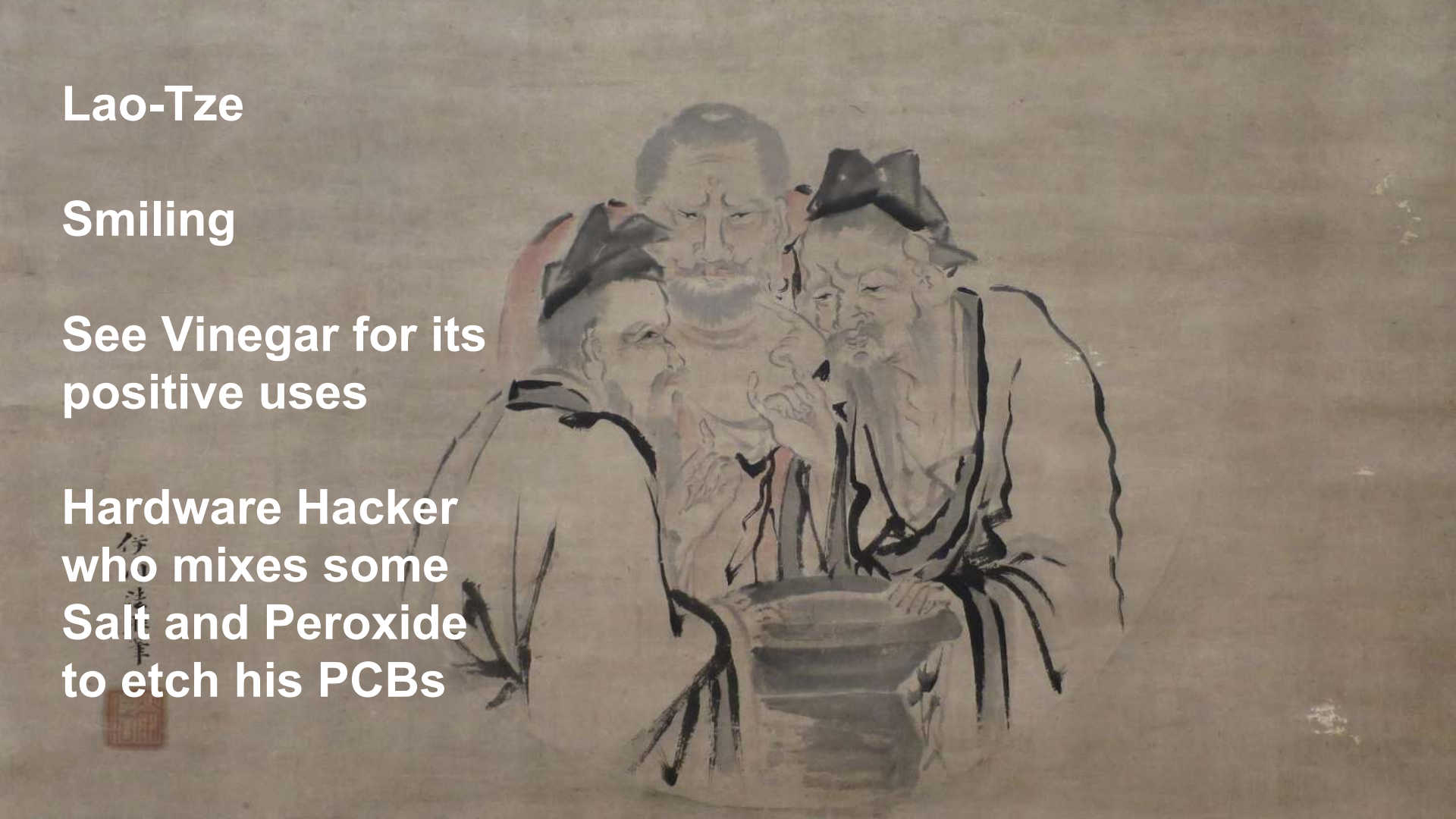
Incident Responder?

Lao-Tze

Smiling

See Vinegar for its positive uses

Hardware Hacker who mixes some Salt and Peroxide to etch his PCBs



The Tao of Hardware
The Te of Implants

Questions?

Joe FitzPatrick - @securelyfitz
joefitz@securinghardware.com