

XiLokar & Virtu

CHS present CHS

'FUN' WITH Watches

LeHAEK 2025

Quarkslab

# /WHO



Xilokar

- > Embedded Software engineer
- > Hobbyist reverser

Virtu

- > Reverse-engineer at Quarkslab
- > Love hacking stuff !
- > Spend too much time (and money) on *AliExpress* and *Amazon* ...

IT ALL STARTS WITH ... A SMARTWATCH

- CAPTN OBVIOUS WAS HERE -



Qu'est-ce qui vous ferait plaisir aujourd'hui ?

Gifi Mon compte Panier

Plein air

Jardin & Extérieur

Offres du moment

Nos produits

Nos petits prix

Nos nouveautés

Nos catalogues

Accueil > Loisirs > Multimédia et divertissement > Informatique > Montre connectée Homday Xpert bluetooth étanche écran tactile 1,83"

Indiquez votre magasin préféré pour voir la disponibilité des produits

Code postal ou ville



Vendu par Gifi

## Montre connectée Homday Xpert bluetooth étanche écran tactile 1,83"

11<sup>€</sup>  
90

Dans 0,15 s à 0,20 s

### Caractéristiques du produit

- Compatibilité : android/iOS
- Dimensions cadran : 3,5 x 4 cm
- Écran : 1,83 pouces
- Entrée : 5V [DC] 0,2A
- Batterie : 3,7V [DC] 200mAh
- Matière(s) : plastique ABS, alliage de zinc, silicone

### Description du produit

Découvrez la montre connectée Homday Xpert, un compagnon idéal pour un mode de vie actif et connecté !

Avec son écran tactile de 1,83 pouces, cette montre vous offre une interface intuitive et facile à naviguer.

Sa conception étanche vous permet de la porter sans souci lors de vos activités quotidiennes, qu'il pleuve ou que vous soyez en pleine séance d'entraînement. Grâce à



# LOOKS AWESOME !



## Description du produit

Découvrez la montre connectée Homday Xpert, un compagnon idéal pour vos activités sportives. Avec son écran tactile de 1,83 pouces, cette montre vous offre une interface intuitive et facile à utiliser. Sa conception étanche vous permet de la porter sans souci lors de vos activités sportives, que ce soit pendant une pleine séance d'entraînement. Grâce à sa connectivité Bluetooth, vous pouvez la synchroniser avec votre smartphone pour un suivi précis de vos activités. Restez stylé et connecté avec la montre Homday Xpert.

Cet article rentre dans le cadre de la réglementation DEEE. Si vous souhaitez le renvoyer, veuillez suivre les instructions indiquées sur la page de retour du magasin GiFi. Cliquez pour plus d'informations, sur les conditions de reprise et de remboursement.

[Voir moins](#)

REF. 000000000000630391

(EN) Features: Waterproof, health monitoring, Bluetooth

# OH, WAIT ...



Source: Underscore, feat. *Stéphane Marty* (a.k.a Deus Ex Silicium)

IS IT ... A SCAM ?



# HARDWARE ANALYSIS

# THE THING THAT SHOULD NOT BE

# WATERPROOF, YOU SAY ?



C'mon, no silicon/epoxy, really ?

# LOOK AT THESE SENSORS LEDS !

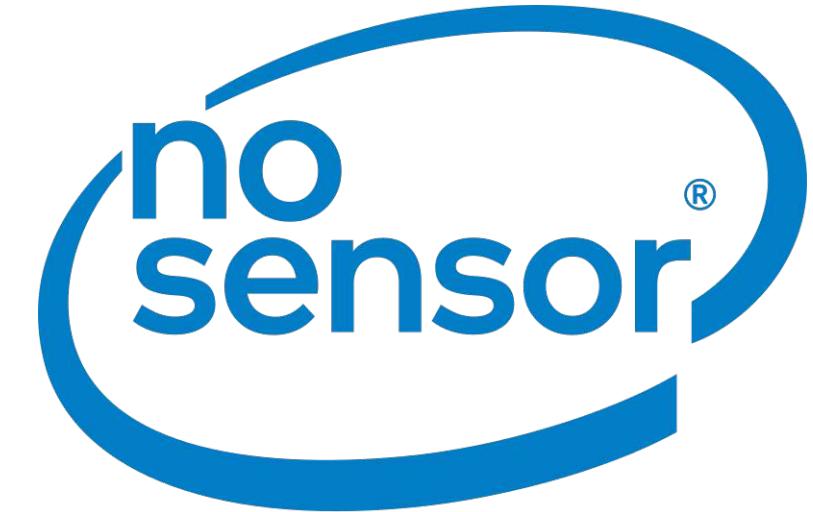


and still no glue/epoxy/silicon ...

# LOOK AT THESE SENSORS LEDS !



and still no glue/epoxy/silicon ...



inside™



# WEIRD SYSTEM-ON-CHIP



Is it a  $\pi$  ? Is it "JC" ? No, it's "JL" !

# PEOPLE ARE STRUGGLING WITH JL CHIPS

The screenshot shows a GitHub repository page for 'kagaimiq/jielie'. The page title is 'JieLi STUFF'. It features the JieLi Technology logo, which includes a stylized red 'JL' monogram and the text '杰理科技' (JieLi Technology) and '用芯美好世界' (Chips Delight the World). Below the logo, there is a subtitle 'JieLi Technology - Chips Delight the World.' followed by a horizontal line. The main content area contains several paragraphs of text. The first paragraph describes JieLi Technology as a company based in Zhuhai, Guangdong, China, in 2010. The second paragraph discusses the company's chip usage across various devices like MP3 players and speakers. The third paragraph provides information on identifying JieLi chips by their markings, mentioning models like 'AC1082', 'AC1187', 'AC1631D97294-04A', 'AC20BP05193-65A4', and 'AC6965A'. The fourth paragraph notes that some chips are rebadged, such as 'TD5161A' being AC6965A or 'MH-M18' being AC6925C. The fifth paragraph states that recent chip series like 'JL697N' and 'JL700N' now have their actual chip names printed on them. At the bottom, a link 'Here' is provided for more information on chip markings.

github.com/kagaimiq/jielie

README

## JieLi STUFF

**JieLi TECHNOLOGY** 用芯美好世界  
Chips Delight the World.

JieLi Technology - Chips Delight the World.

---

JieLi Technology ('杰理' aka 'Jerry', 'n' // 'jelly', 光) is a company that was based in year 2010 in the Zhuhai city of Guangdong province, China.

Their chips can be found in various cheap MP3 players, Bluetooth speakers, Bluetooth/MP3 modules, dash cams and so on. (along with other companies like "Jianrong" Appotech/Buildwin, "AB" Bluetrum, MVSilicon, Beken, RDA, etc.)

Chips can be identified as JieLi ones by them having a slanted "JL"/"n" logo, and a misleading/nonsense marking that doesn't tell the real chip name right away, like "AB1526CG3X1F.1-82E" (AC1082), "AB1819CK1Y11.1-87" (AC1187), "AC1631D97294-04A" (AC5204A), "AC20BP05193-65A4" (AC6965A), to name a few.

Sometimes their chips are rebadged, for instance: "TD5161A" is the AC6965A, "MH-M18" is AC6925C (recently it's AC6925D).

Although it seems like they now do print the actual chip names on some recent chip series like JL697N, JL700N, etc.

[Here](#) is some information on what the markings actually mean.

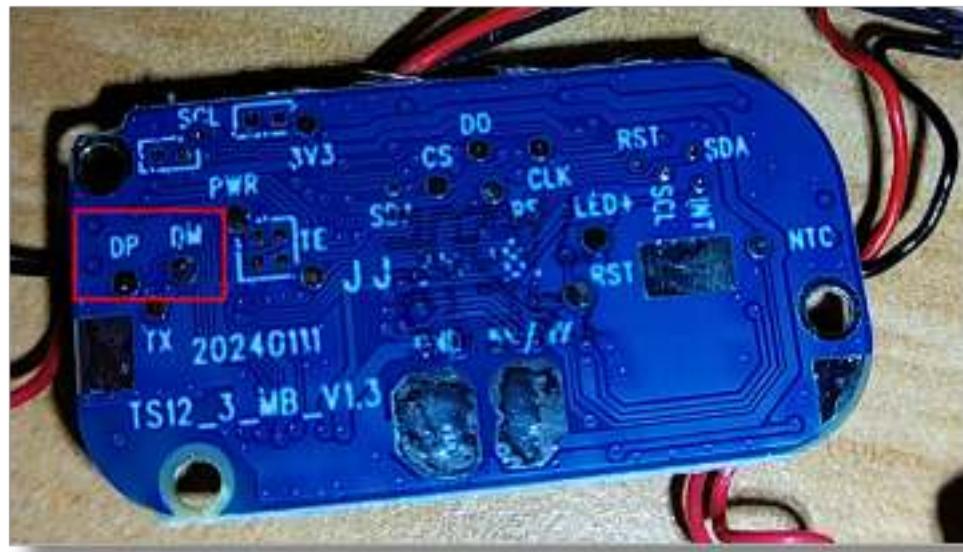
<https://github.com/kagaimiq/jielie/>

# LETS EXTRACT ITS FIRMWARE !



# FIRMWARE EXTRACTION / PART #1

# USB TEST PADS



- > Soldered DM and DP to a USB connector
- > Nope.

# JIELI PROPRIETARY PROGRAMMER



€30, 4 weeks to wait for delivery

```
#define DATA_PIN 8
#define CLOCK_PIN 7

void clockOut(int value) {
    digitalWrite(DATA_PIN, value);
    digitalWrite(CLOCK_PIN, 0);
    delayMicroseconds(7);
    digitalWrite(CLOCK_PIN, 1);
    delayMicroseconds(3);
}

void setup() {
    pinMode(CLOCK_PIN, OUTPUT);
    pinMode(DATA_PIN, OUTPUT);
}

void loop() {
    clockOut(0); clockOut(0); clockOut(0); clockOut(1);
    clockOut(0); clockOut(1); clockOut(1); clockOut(0);
    clockOut(1); clockOut(1); clockOut(1); clockOut(0);
    clockOut(1); clockOut(1); clockOut(1); clockOut(1);

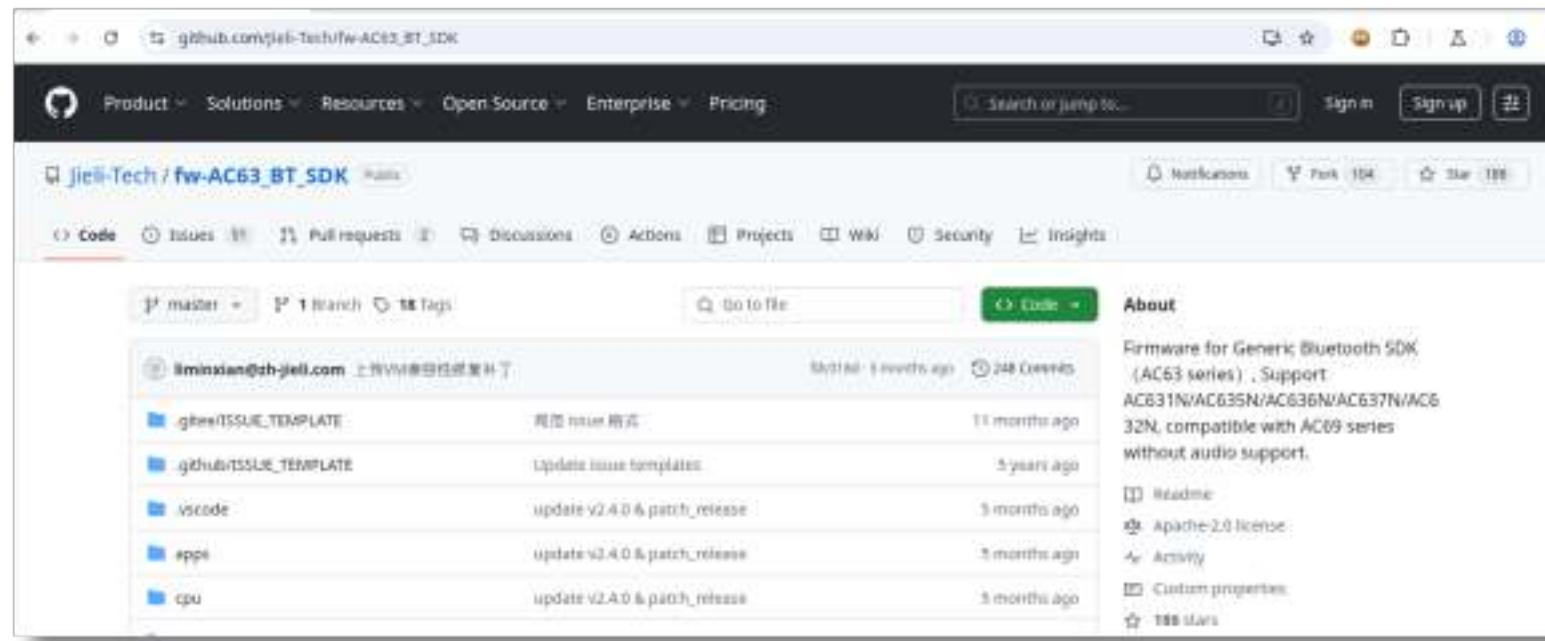
    delayMicroseconds(2);
    digitalWrite(DATA_PIN, 0);
    digitalWrite(CLOCK_PIN, 0);
}
```

Cool hack discovered weeks later



# SEARCHING THE (DEEP) WEB

# SDK AVAILABLE ON GITHUB



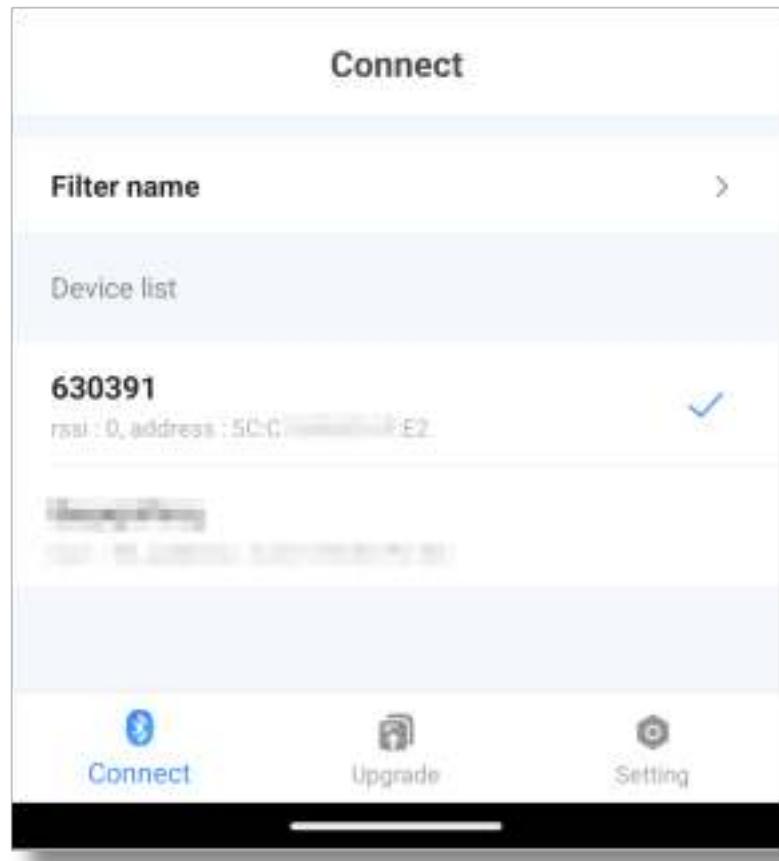
including a working toolchain

# GITHUB REPO WITH OTA ?



and there's a release !

# A WARRIOR HAS NO FEAR



# ANDROID LOGCAT (EXTRACT)

```
> -startAuth- device = name : 630391
> -sendAuthDataToDevice- authData : 00E93EA141E1FC673E017E97EADC6B968F
> -handleAuthData- data : 015B196811ACDC6C1A8F454232D652A9CC
> -sendAuthDataToDevice- authData : 0270617373
> -handleAuthData- data : 00F3F1310CD723FF8290A1F513B7049B3B
> -sendAuthDataToDevice- authData : 01CC1C01B262AEBB4F340F74774FA8A713
> -handleAuthData- data : 0270617373
> -onAuthSuccess- auth ok.
> -handleAuthData- auth ok.
```

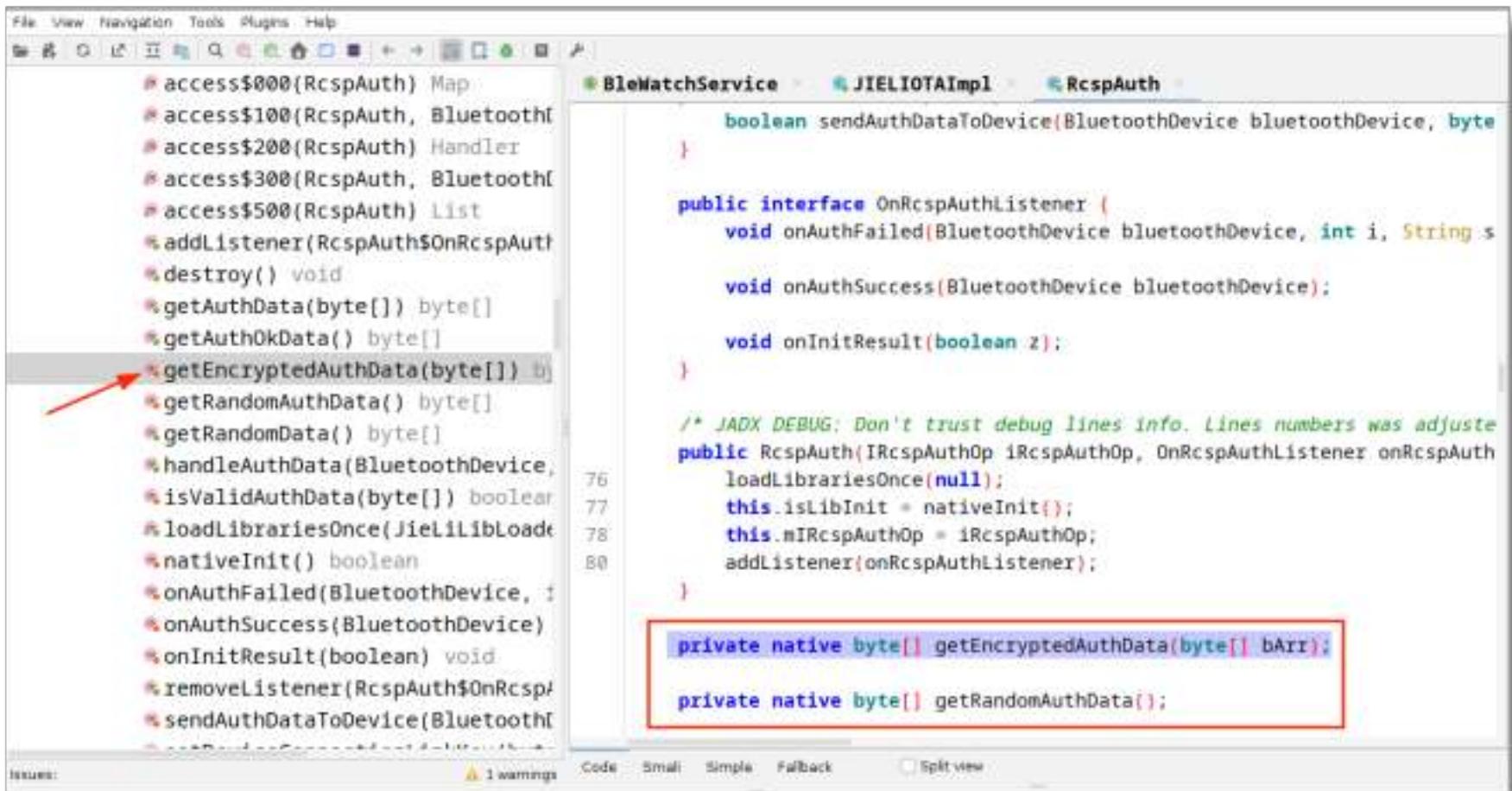
Authentication ongoing ...

# ANDROID LOGCAT (CONTD)

```
D ota:OTAManager: receiveDataFromDevice :: CommandBase{  
    opCodeSn=24, opCode=3, name='GetTargetInfoCmd', type=2, status=0,  
    param=GetTargetInfoParam{mask=-1}, response=TargetInfoResponse{  
        versionName='V_0.0.0.0', versionCode=0, protocolVersion='V_2.0',  
        edrAddr='5C:C1:B9:4F:09:7A', edrStatus=0, edrProfile=-114,  
        bleAddr='5C:C1:B9:BC:0F:E2', volume=0, maxVol=0, quantity=100,  
        lowPowerLimit=30, functionMask=0, curFunction=0, sdkType=8,  
        name='null', pid=95, vid=2, uid=0, mandatoryUpgradeFlag=0,  
        requestOtaFlag=0, ubootVersionCode=58396,  
        ubootVersionName='14.4.1.12', isSupportDoubleBackup=false,  
        isNeedBootLoader=true, singleBackupOtaWay=1, allowConnectFlag=0,  
        authKey='null', projectCode='null', customVersionMsg='null',  
        bleOnly=false, isSupportMD5=false, isGameMode=false, expandMode=0,  
        communicationMtu=540, receiveMtu=272  
    }  
}
```

UBoot version ? At least it speaks over OTA !

# DIGGING DEEPER INTO THE APK



```
File View Navigation Tools Plugins Help
B G O L I Q E D S + - M F P A
RrspAuth
    ● access$000(RrspAuth) Map
    ● access$100(RrspAuth, BluetoothDevice)
    ● access$200(RrspAuth) Handler
    ● access$300(RrspAuth, BluetoothDevice)
    ● access$500(RrspAuth) List
    ● addListener(RrspAuth$OnRrspAuthListener)
    ● destroy() void
    ● getAuthData(byte[]) byte[]
    ● getAuthOkData() byte[]
    ● getEncryptedAuthData(byte[]) byte[] *
    ● getRandomAuthData() byte[]
    ● getRandomData() byte[]
    ● handleAuthData(BluetoothDevice, byte[])
    ● isValidAuthData(byte[]) boolean
    ● loadLibrariesOnce(JieLiliLibLoader)
    ● nativeInit() boolean
    ● onAuthFailed(BluetoothDevice, int, String)
    ● onAuthSuccess(BluetoothDevice)
    ● onInitResult(boolean) void
    ● removeListener(RrspAuth$OnRrspAuthListener)
    ● sendAuthDataToDevice(BluetoothDevice, byte[])
    ● startAuth(BluetoothDevice, byte[])
    ● stopAuth()

    ● BleWatchService
    ● JIELIOTAIImpl
    ● RrspAuth

        boolean sendAuthDataToDevice(BluetoothDevice bluetoothDevice, byte[] bArr);
    }

    public interface OnRrspAuthListener {
        void onAuthFailed(BluetoothDevice bluetoothDevice, int i, String s);
        void onAuthSuccess(BluetoothDevice bluetoothDevice);
        void onInitResult(boolean z);
    }

    /* JADX DEBUG: Don't trust debug lines info. Lines numbers was adjusted */
    public RrspAuth(IRrspAuthOp iRrspAuthOp, OnRrspAuthListener onRrspAuthListener) {
        76        loadLibrariesOnce(null);
        77        this.isLibInit = nativeInit();
        78        this.mIRrspAuthOp = iRrspAuthOp;
        79        addListener(onRrspAuthListener);
    }

    private native byte[] getEncryptedAuthData(byte[] bArr);

    private native byte[] getRandomAuthData();
}
```

The screenshot shows the JADX decompiler interface with the RrspAuth class loaded. The left pane lists various methods, and the right pane shows the corresponding Java code. A red arrow points to the `getEncryptedAuthData` method in the list of methods on the left. In the code editor, two native methods are highlighted with a red border: `private native byte[] getEncryptedAuthData(byte[] bArr);` and `private native byte[] getRandomAuthData();`.

# DIGGING DEEPER INTO THE APK

The screenshot shows the Immunity Debugger interface with two panes: 'Listing' on the left and 'Decompiler' on the right.

**Listing Tab:**

- Shows assembly code for the `libota_auth.so` library.
- Registers and memory dump sections are visible.
- A red box highlights the `default_sec` section, which contains several `??` entries.
- A pink box highlights the `s_w__#_00014a36` section, containing assembly code for writing to memory.

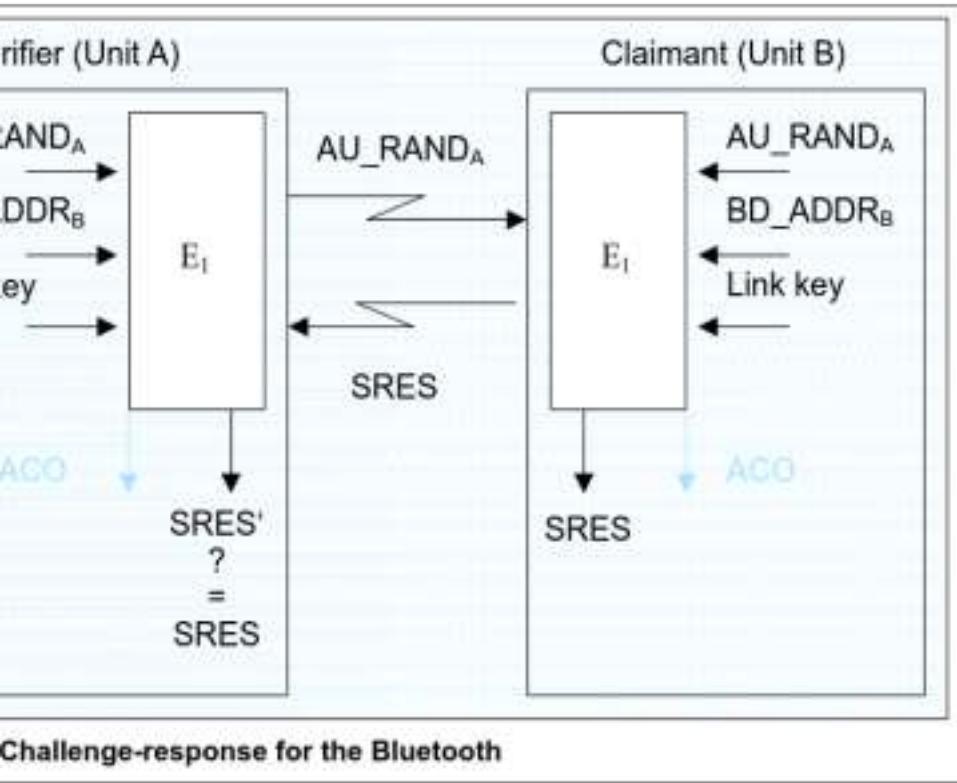
**Decompiler Tab:**

- Shows the decompiled C-like code for the `getEncryptedAuthData` function.
- The highlighted `default_sec` section corresponds to the `function_Eltest` call at line 14.
- The highlighted `s_w__#_00014a36` section corresponds to the `writ1` and `writ2` assignments at lines 21 and 26.
- Annotations with arrows point from the assembly highlights to the corresponding decompiled code lines.

```
Decompiled code for getEncryptedAuthData (libota_auth.so)
```

```
1 void getEncryptedAuthData(int *param_1, undefined4 param_2, int param_3)
2 {
3     int iVar1;
4     undefined4 uVar2;
5     undefined1 uStack_2d;
6     byte abStack_2c [16];
7     int iStack_1c;
8
9     iStack_1c = __stack_chk_guard;
10    if (param_3 == 0) {
11        uStack_2d = 1;
12        function_Eltest(&default_sec,(byte *)0x1,(undefined (*) [16])&_e__#_00014a36,abStack_2c);
13    }
14    else {
15        (**(code **)*(param_1 + 0x20c))(param_1,param_3);
16        iVar1 = (**(code **)*(param_1 + 0x208))(param_1,param_3,0);
17        iVar1 = (**(code **)*(param_1 + 0x200))(param_1,param_3,0);
18        iVar1 = (**(code **)*(param_1 + 0x1f))(iVar1 + 1),(undefined (*) [16])&_e__#_00014a36,abStack_2c);
19    }
20    if (iVar1 != 0) {
21        (**(code **)*(param_1 + 0x100))(param_1,param_3,iStack_1c,0);
22    }
23    iVar2 = (**(code **)*(param_1 + 0x208))(param_1,0x11);
24    (**(code **)*(param_1 + 0x300))(param_1,iVar2,0,0x11,&uStack_2d);
25    if (__stack_chk_guard != iStack_1c) {
26        /* WARNING: Subroutine does not return */
27        __stack_chk_fail(iStack_1c);
28    }
29 }
30 }
```

# E<sub>1</sub> FUNCTION ? KNOB HAS IT COVERED.



The screenshot shows a GitHub repository named `francozappa/knob` with a file named `e1/e1.py`.

**Code Snippet:**

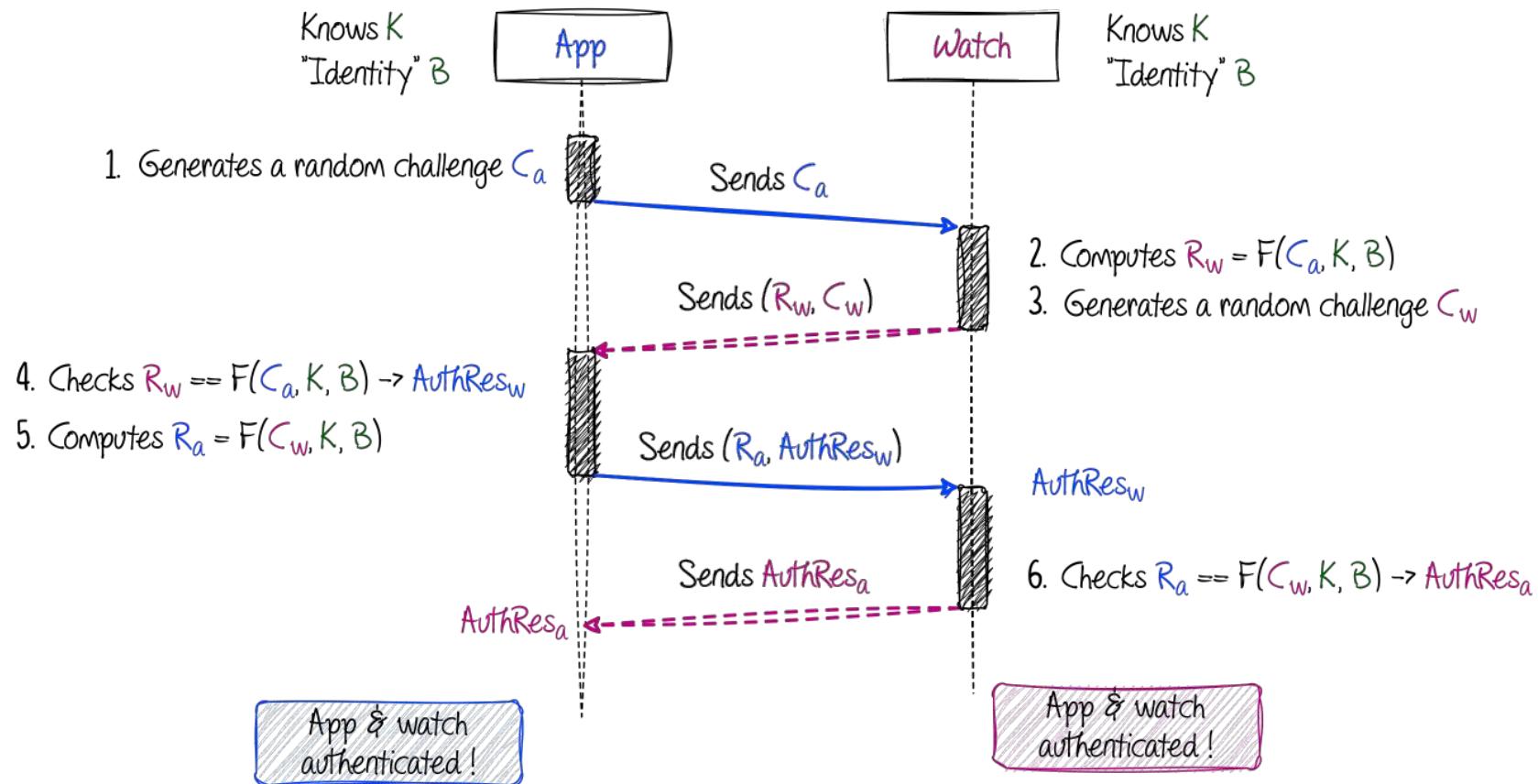
```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
from e1 import *
from R import *
log.setLevel(logging.DEBUG)

def e1(AU_RAND, BDADDR):
    """Generate SRES and ACO, Eq 12, pag 36/51.

    BDADDR is slave address (format byte 1 is RSB)
    R = Rand(16 - RAND, address, 8)
    """
    R = Rand(16 - RAND, address, 8)
```

Thank you, Daniele Antonioli !  
(a.k.a Franco Zappa)

# JIELI OTA (MUTUAL) AUTH IN A NUTSHELL



Pre-shared constant values  $K$  and  $B$  used.

Btw, this scheme is prone to a reflective attack (no need to know  $K$  and  $B$ )



# THIS IS THE MOMENT I KNEW ... I F\*CKED UP

- > JieLi OTA cannot download firmware from device 
- > Lost days on this for nothing ... or maybe not ?

# THIS IS THE MOMENT I KNEW ...

## I F\*CKED UP

- > JieLi OTA cannot download firmware from device 😡
- > Lost days on this for nothing ... or maybe not ?
- > Wait, I know how authentication works !

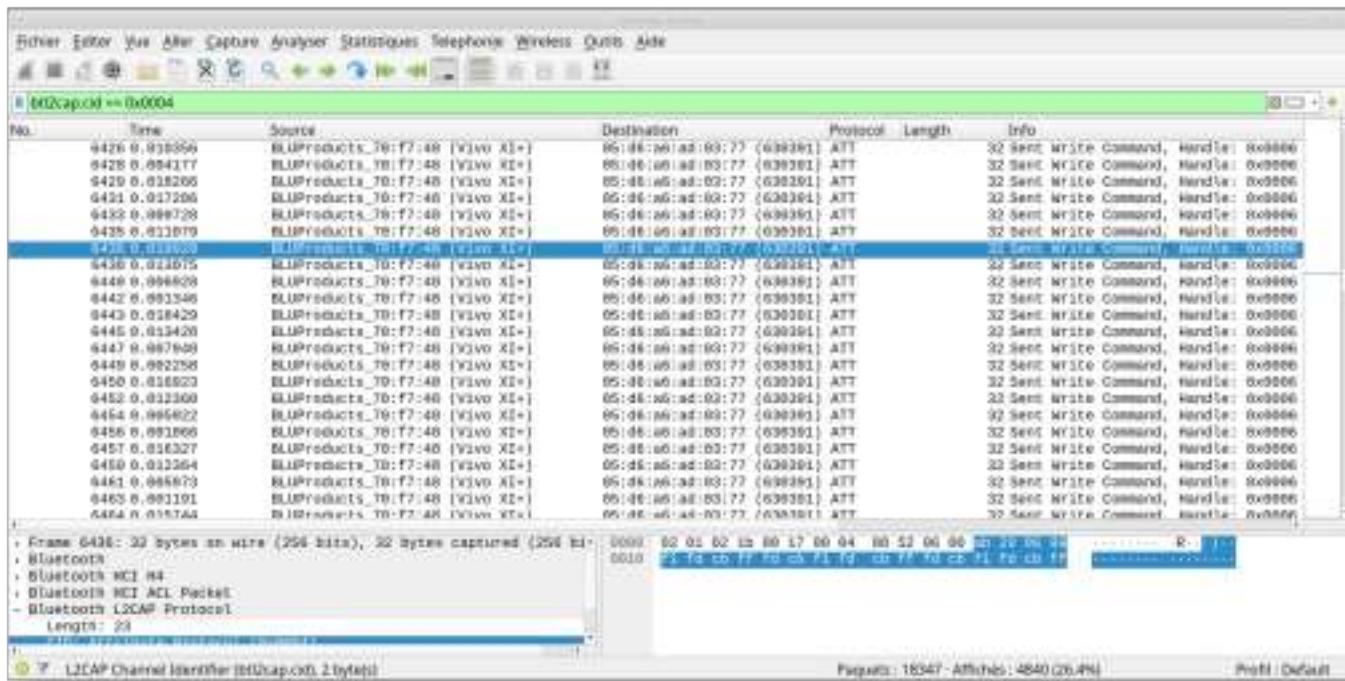


IN YOUR FACE ?

SO DYNAMIC

There must be some code here

# SNIFFING THE DIALS



BLE OTA authentication first, then dial file is sent

# SNIFFING THE DIALS

```
xiLukar@xiLukar:~/tmp/dials$ ls  
sniffed_dial.bin  
xiLukar@xiLukar:~/tmp/dials$ █
```

WE NEED MORE FILES...

# WE NEED MORE FILES...



# ANDROID LOGCAT (ONE MORE TIME)

```
I OKHTTP : --->Date: Mon, 24 Feb 2025 21:59:00 GMT
I OKHTTP : --->Content-Type: application/json
I OKHTTP : --->Transfer-Encoding: chunked
I OKHTTP : --->Connection: keep-alive
I OKHTTP : --->Vary: Origin
I OKHTTP : ---><-- 200 OK "https://app-overseas.sz-tjd.com/overseas/api/
    app/dialsmall/dialSmall/getTagDialCollection" (183ms)
I OKHTTP : --->
I OKHTTP : --->
I OKHTTP : --->{ "code":200, "msg":"æ<9f>¥è¯¢æ<88><90>å<8a><9f>" , "data": [...]
I OKHTTP : ---><-- END HTTP (715-byte body)
```

## getTagDialCollection

```
{  
  "16023": {  
    "dialName": "JLC1198",  
    "prepicUrl": "/statics/dial/2022/11/26/1669446584711_JLC1198.png",  
    "price": 0.99  
  },  
  "16017": {  
    "dialName": "JLC1192",  
    "prepicUrl": "/statics/dial/2022/11/26/1669446429808_JLC1192.png",  
    "price": 0.99  
  },  
  ...  
}
```

## getDialDetails

```
{  
  "data" : {  
    "id":16023,  
    "price": "0.99",  
    "dwCount": 8,  
    "virtuallyDwCount": 6994,  
    "url":"https://cdn.sz-tjd.com/pro/.../1669446584711_JLC1198.bin"  
  }  
}
```

## getDialDetails

```
{  
  "data" : {  
    "id":16023,  
    "price": "0.99",  
    "dwCount": 8,  
    "virtuallyDwCount": 6994,  
    "url":"https://cdn.sz-tjd.com/pro/.../1669446584711_JLC1198.bin"  
  }  
}
```

Oh, thanks!

# SO MUCH DIALS

```
nickat@nickat-OptiPlex-5090:~/tmp/dials$ ls  
1668663382373 JLC1004.bin 1668750176491 JLC1016.bin 1668840703021 JLC1076.bin 1669433212835 JLC1170.bin  
1668663469345 JLC1005.bin 1668751053088 JLC1014.bin 1668848301526 JLC1061.bin 1669433267575 JLC1171.bin  
1668667414320 JLC1042.bin 1668753131397 JLC1015.bin 1668848309526 JLC1062.bin 1669433297777 JLC1172.bin  
1668667515468 JLC1047.bin 1668756348483 JLC1027.bin 1668848317388 JLC1069.bin 1669445941871 JLC1174.bin  
1668667542873 JLC1048.bin 1668756786184 JLC1031.bin 1668995560995 JLC1215.bin 1669445974365 JLC1175.bin  
1668667682597 JLC1053.bin 1668759748634 JLC1030.bin 1669103267962 JLC1058.bin 1669446166631 JLC1182.bin  
1668667783879 JLC1059.bin 1668761038604 JLC1032.bin 1669103277866 JLC1074.bin 1669446194815 JLC1182.bin  
1668668027738 JLC1006.bin 1668763254947 JLC1029.bin 1669103289668 JLC1075.bin 1669446226874 JLC1184.bin  
1668670563889 JLC1007.bin 1668824115387 JLC1056.bin 1669103298237 JLC1077.bin 1669446252498 JLC1185.bin  
1668670667342 JLC1016.bin 1668830359866 JLC1056.bin 1669103310647 JLC1078.bin 1669446314628 JLC1187.bin  
1668734654741 JLC1021.bin 1668830367432 JLC1049.bin 1669103320270 JLC1079.bin 1669446338695 JLC1189.bin  
1668734675000 JLC1024.bin 1668830383527 JLC1043.bin 1669103330700 JLC1080.bin 1669446429808 JLC1192.bin  
1668734783725 JLC1025.bin 1668840374338 JLC1063.bin 1669171893118 JLC1044.bin 1669446454298 JLC1193.bin  
1668734728641 JLC1033.bin 1668840404953 JLC1064.bin 1669190372418 JLC1009.bin 1669446484313 JLC1194.bin  
1668734739456 JLC1035.bin 1668840427488 JLC1065.bin 1669432250265 JLC1152.bin 1669446510231 JLC1195.bin  
1668734747826 JLC1036.bin 1668840451924 JLC1066.bin 1669432336966 JLC1156.bin 1669446538781 JLC1196.bin  
1668734754539 JLC1037.bin 1668840475477 JLC1067.bin 1669432416127 JLC1159.bin 1669446562616 JLC1197.bin  
1668734761327 JLC1038.bin 1668840498598 JLC1068.bin 1669432439880 JLC1160.bin 1669446584711 JLC1198.bin  
1668734843981 JLC1088.bin 1668840583939 JLC1071.bin 16694330806286 JLC1165.bin 1669446975725 JLC1181.bin  
1668734869423 JLC1011.bin 1668840606398 JLC1072.bin 1669433078382 JLC1167.bin 1670290515032 JLC1041.bin  
1668734878896 JLC1013.bin 1668840632368 JLC1073.bin 1669433104782 JLC1168.bin 1679277875597 JLF13.bin  
nickat@nickat-OptiPlex-5090:~/tmp/dials$
```

# DIAL FORMAT

	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0x0	04 00 02 00 01 00 04 00 01 FE 01 00 F0 00 1E 01
0x10	03 00 40 00 00 00 0A 04 F0 00 02 00 01 00 01 00
0x20	58 00 00 00 0A 04 F0 00 01 00 01 00 01 00 28 04
0x30	00 00 00 04 F0 00 07 00 01 00 01 00 18 06 00 00
0x40	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x50	00 00 00 00 00 00 00 00 FF FF 01 83 00 00 78 00
0x60	68 00 00 00 C0 03 00 00 A5 A5 A5 A5 FF FF FF FF
0x70	FF

0+6	Header	24	Descriptor 3
6+2	num_descriptor	58	Entry
8	Descriptor 1	58+2	bg_color
16	Descriptor 2	5A+2	type
16+2	type	5C+2	x
18+2	width	5E+2	y
1A+2	height	60+4	offset
20+4	offset	0x58	64+4 size
			0x64
			0x30C
		68+960	payload

## DIAL FORMAT - DESCRIPTOR

- > type (hour, minutes, picture)
- > number of values
- > entry offset in file

# DIAL FORMAT - ENTRY

- > background color
- > pixel format
- > position on screen
- > payload offset in file

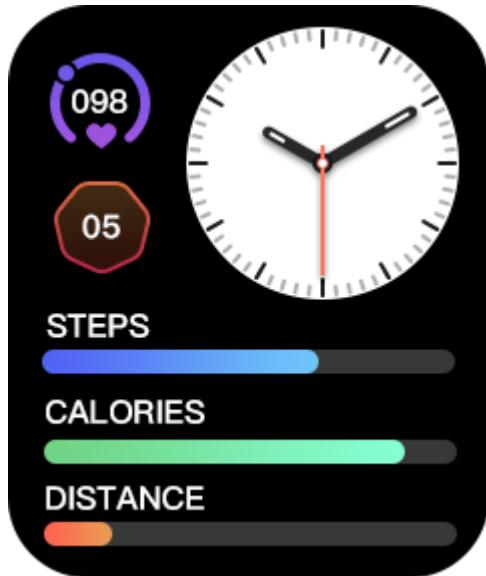
# DIAL FORMAT - PAYLOAD

- > raw RGB565
- > RLE compressed
- > mask with background color

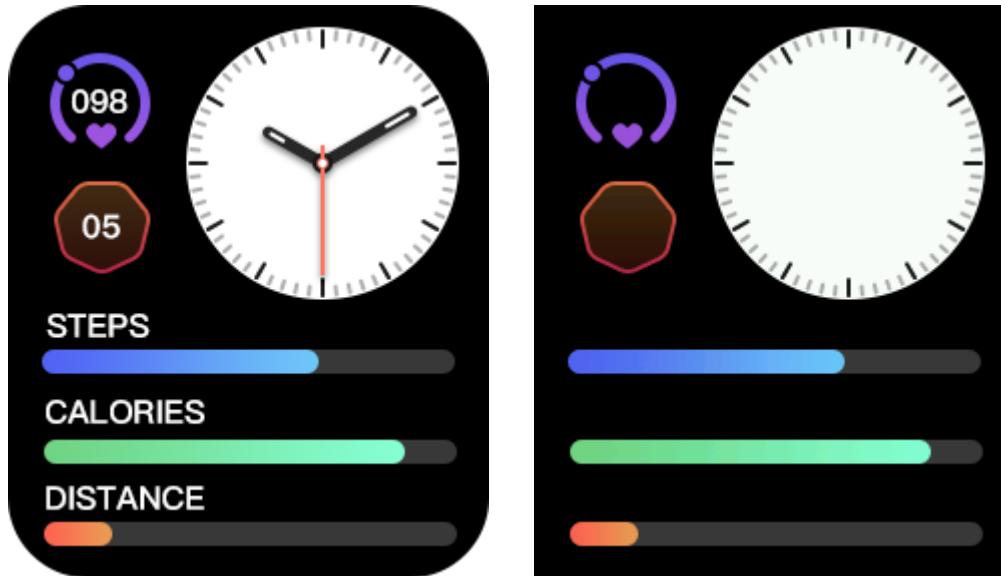
# DIAL FORMAT – STRUCTURE

- > small preview
- > background
- > fonts
- > pictures

# DIAL FORMAT - ODDITIES



# DIAL FORMAT - ODDITIES



# DIAL UPLOAD

- > python dial generator
- > whad script to upload (implements JieLi's OTA mutual Auth)

# DIAL UPLOAD



COOL BUT

COOL BUT  
WHERE IS THE CODE ?



# FUN WITH OFFSETS



# FUN WITH OFFSETS



# FUN WITH OFFSETS

# FUN WITH OFFSETS

- > Find base address

# FUN WITH OFFSETS

- > Find base address
- > Sometimes, it crashes

# FUN WITH OFFSETS

- > Find base address
- > Sometimes, it crashes
- > Recovering is hard

# FUN WITH OFFSETS

- > Find base address
- > Sometimes, it crashes
- > Recovering is hard
- > file loaded at 0xf9b000 ?

# FUN WITH OFFSETS

- > Find base address
- > Sometimes, it crashes
- > Recovering is hard
- > file loaded at 0xf9b000 ?

AND NOW, LET'S EXTRACT THIS FIRMWARE !

# FROM PIXELS TO FIRMWARE



~~50~~ 16 SHADES OF GREY

# 50 16 SHADES OF GREY

0	█	A	█	2	█	0
1	█	A	█	2	█	1
2	█	A	█	2	█	2
3	█	A	█	2	█	3
4	█	A	█	2	█	4
5	█	A	█	2	█	5
6	█	A	█	2	█	6
7	█	A	█	2	█	7
8	█	A	█	2	█	8
9	█	A	█	2	█	9
A	█	A	█	2	█	A
B	█	A	█	2	█	B
C	█	A	█	2	█	C
D	█	A	█	2	█	D
E	█	A	█	2	█	E
F	█	A	█	2	█	F

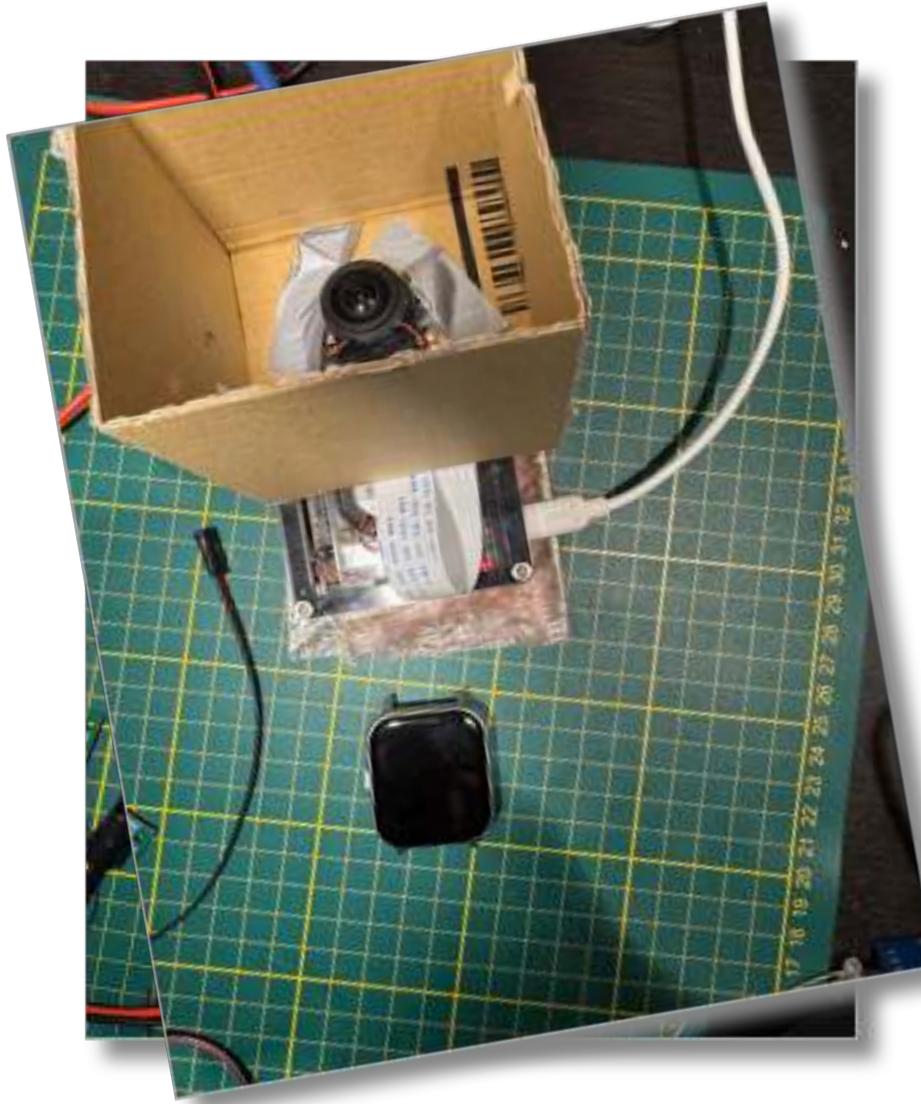
Theory for 0xA2

# PROFESSIONAL SETUP





# PROFESSIONAL SETUP



52.1

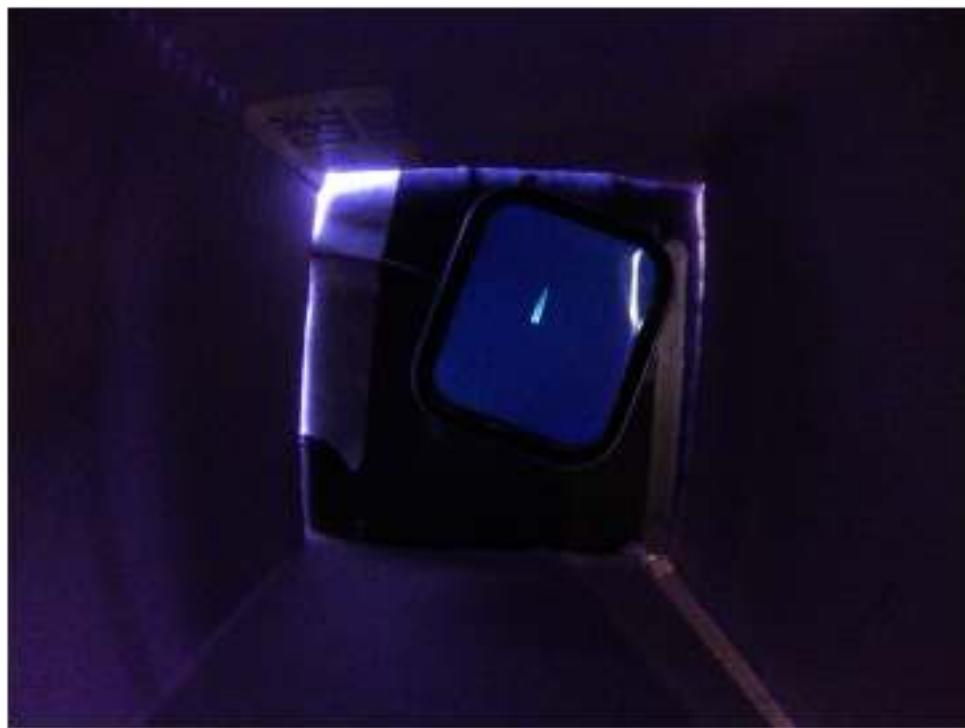


# PROFESSIONAL SETUP

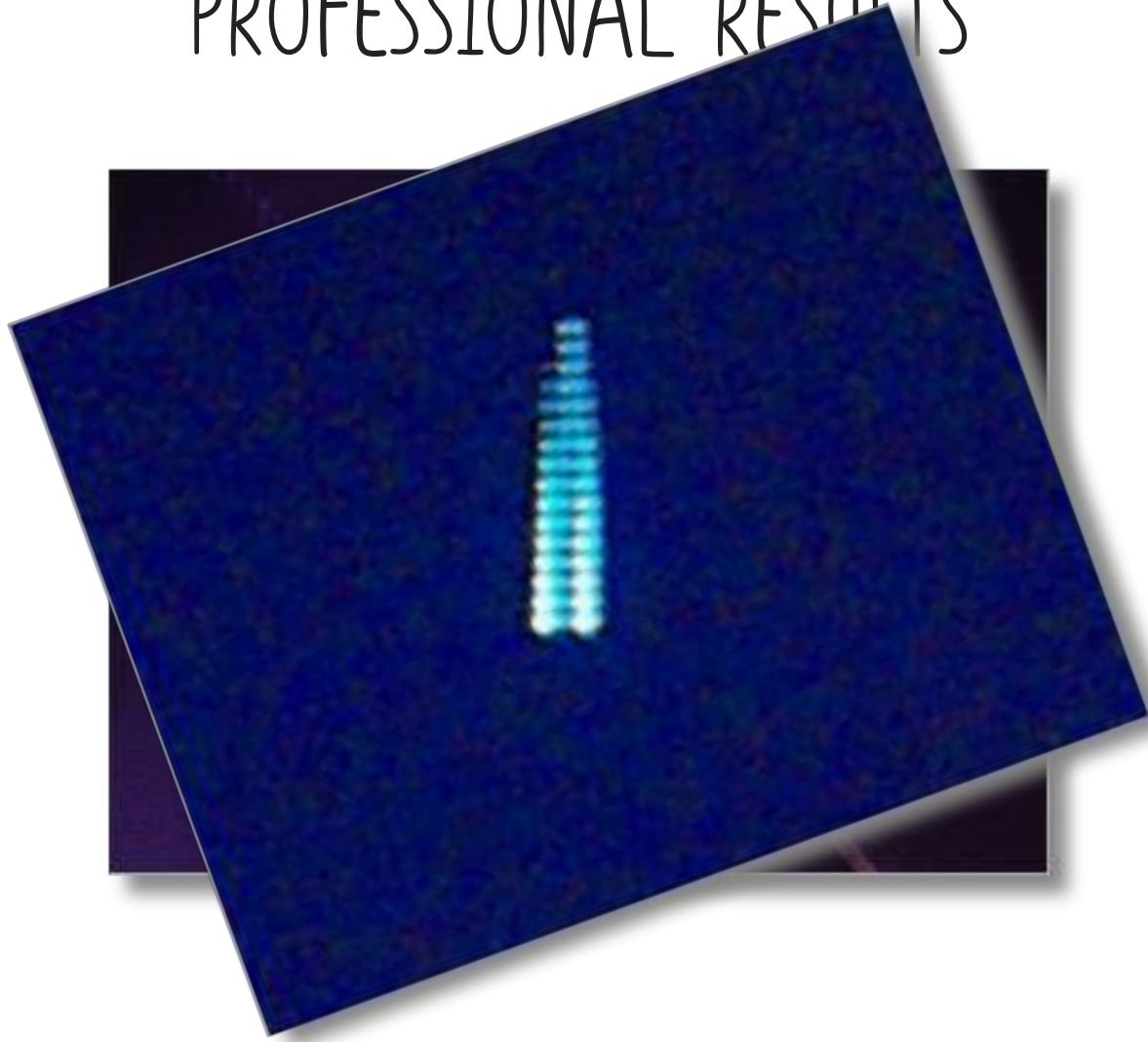




# PROFESSIONAL RESULTS



# PROFESSIONAL RESULTS



# FIRMWARE EXTRACTION / PART #2

HERE WE GO AGAIN, M\*THERF\*CK3R !

# READING PIXELS FROM SCREEN

- > Using a *camera* does not seem to work
- > Why don't we simply sniff pixels from the screen connector?
- > We just need to solder some wires and see what's going on

# SCREEN CONTROLLER ?

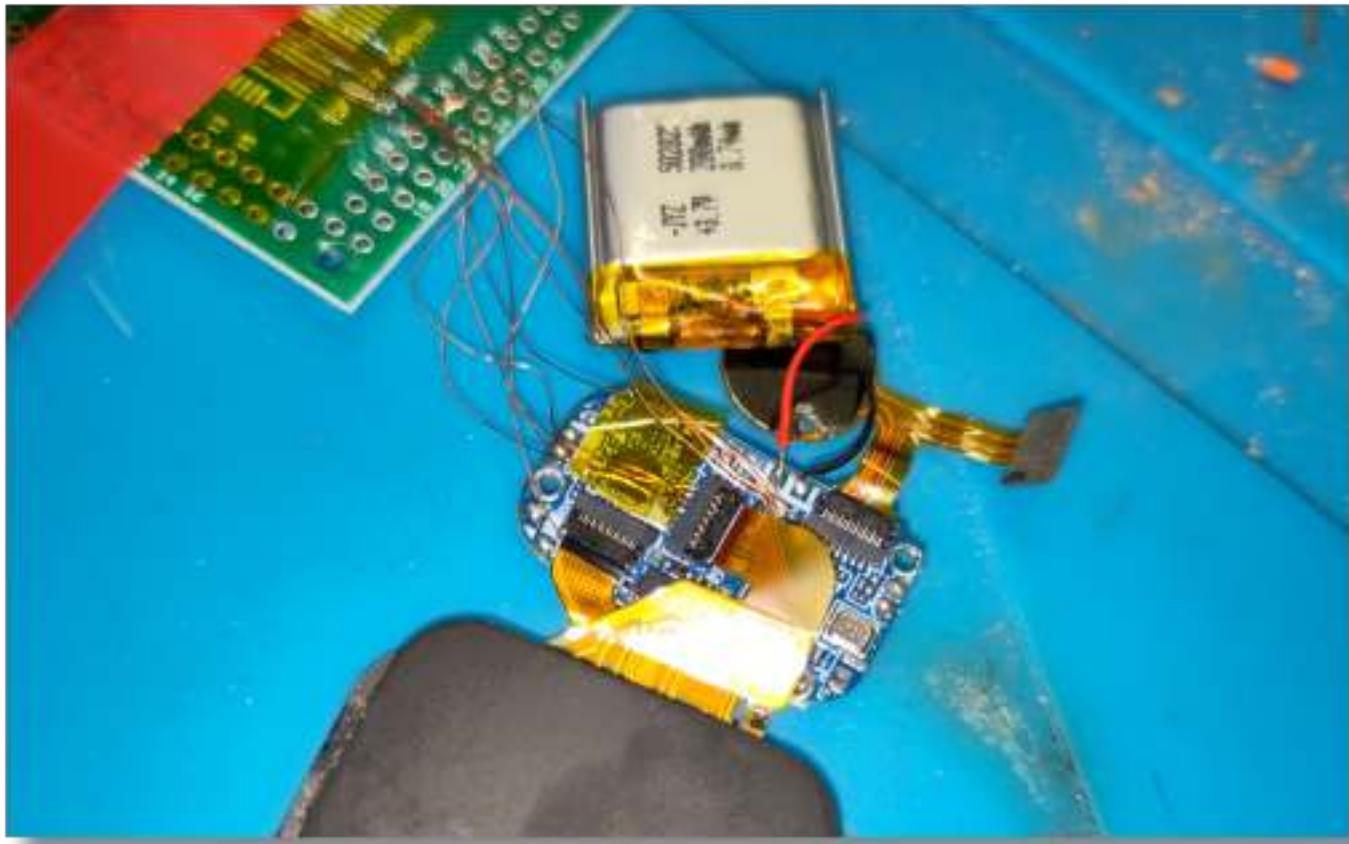


OEM	Place of Origin	Guangdong, C
ENH	Resolution	240×284
1.83	Viewing Angle	ALL
TFT	Driving IC	NV3030B
4 white LEDs	Operating temp	-20°C To 70°C



Resolution	240×280
interface	4 Line SPI
Drive IC	ST7789V2
Top	-20°C~+70°C

# WIRES, WIRES EVERYWHERE !



# ITS FAAAAAST !



# 400 MHZ LOGIC ANALYZER ON A BUDGET



# HACKADAY

[HOME](#)[BLOG](#)[HACKADAY.IO](#)[TINDIE](#)[CONTESTS](#)[SUBMIT](#)[ABOUT](#)

June 19, 2025

## BUILD A 400 MHZ LOGIC ANALYZER FOR \$35

by: [John Elliot V.](#)

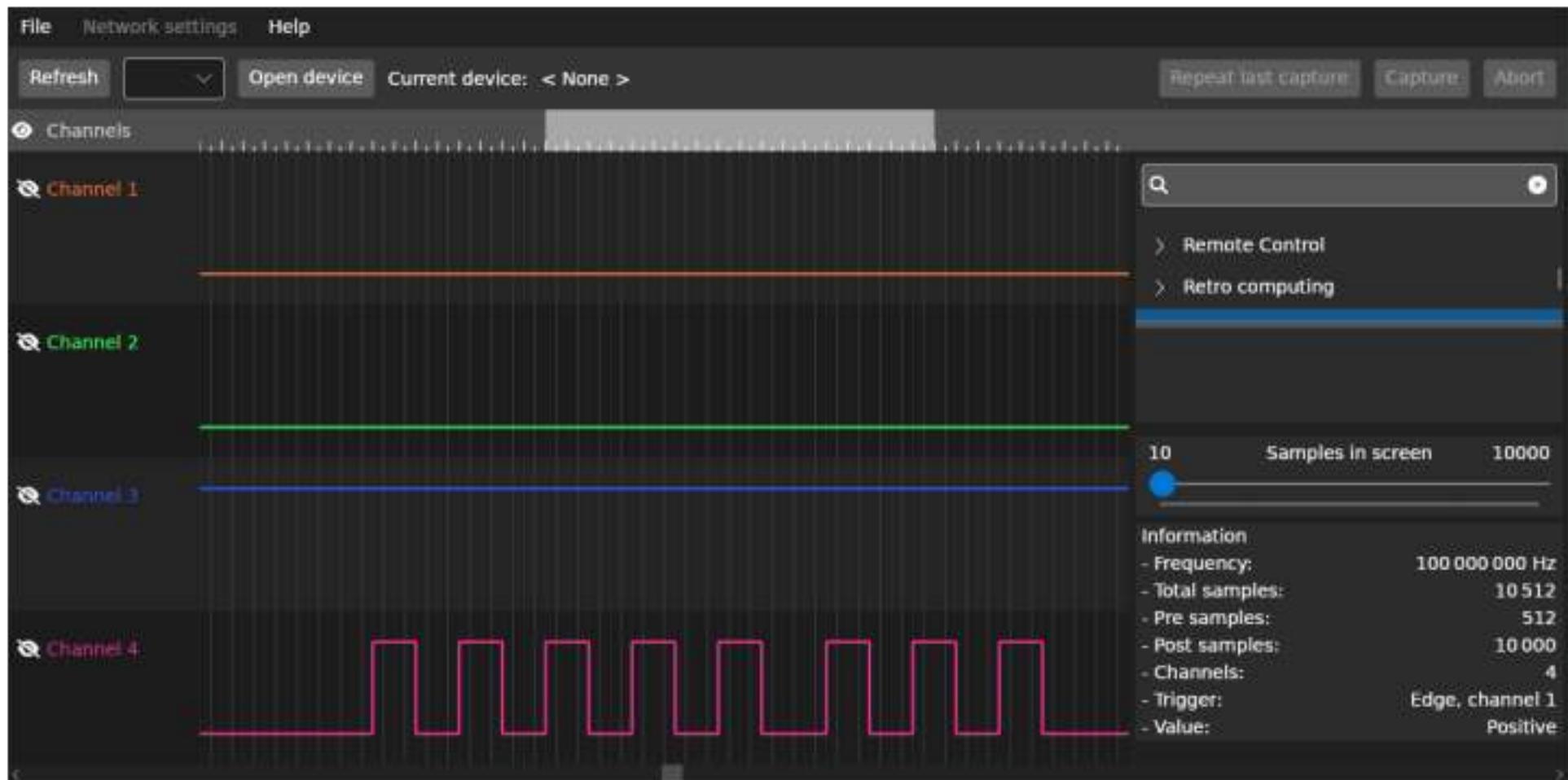
14 Comments



June 12, 2025

[SEARCH](#)

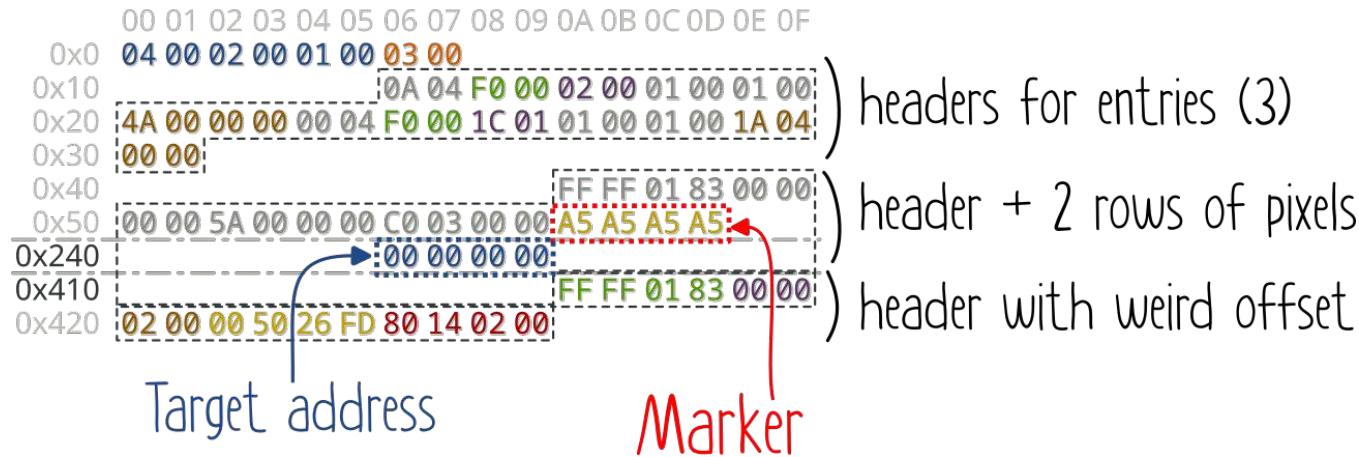
# LOGIC ANALYZER FTW: 25 MHz SIGNAL



# PROGRESS SO FAR ...

- > Screen communication uses a *3-wire interface*
- > Clock line is about 25 MHz 
- > We found the *MOSI* line that ships pixels data
- > Confirmed screen IC is NV3030B

# CRAFTING A DIAL WITH MARKER AND OUT-OF-BOUND OFFSET



Weird offset is 0xfd265000

Data size is  $240 \times 284 \times 2 = 136320$  (0x21480)

# UPLOADING DIAL WITH BLE



# OVERCLOCKED RP2040 + PIO

```
; Fetch bits when raising edge on clock
.define gpio_clock 3

.program sniff

    wait 1 gpio gpio_clock ; wait clock to go high (raising edge)
    in pins, 1              ; read 1 bit from mapped GPIOs and feed
                            ; ISR (autopush set to 8), bytes read
                            ; will be automatically sent to RX FIFO
```

```
while(1) {
    index = 0; // Acquire data from PIO

    pd = (uint32_t *)buffer;
    while (pd < (uint32_t *)(buffer + BUFFER_SIZE))
        *(pd++) = pio_sm_get_blocking (pio, 0); // Wait for PIO to send a byte

    for (int j = 0; j < (BUFFER_SIZE/4); j++)
        printf("%02x%02x%02x%02x", buffer[j*4+3], buffer[j*4+2], buffer[j*4+1], buffer[j*4]);
}
```

# PYTHON: READING BYTES FROM SERIAL

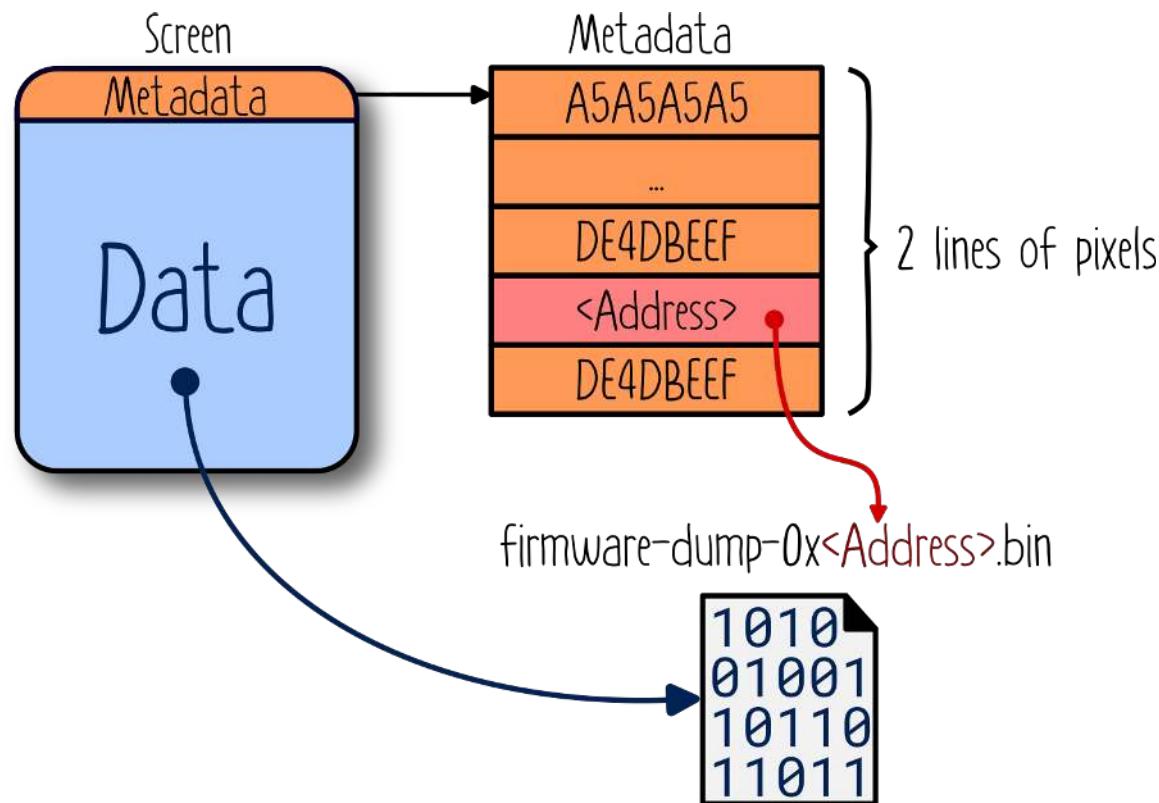
```
# Open serial port and collect data. press CTL-C to process.
sniffer = Serial("/dev/ttyACM0", 115200)
content = b''
while True:
    try:
        buf = sniffer.read(4096)
        if len(buf) > 0:
            content += buf
            sys.stdout.write(f"[{i}] Collected {len(content)} bytes\r")
    except KeyboardInterrupt:
        break
```

# RETRIEVING PIXELS FROM SNIFFED DATA

```
0005F240: 24 50 16 BF EA 79 C7 44 20 14 95 40 28 D8 4F 47 $P..y.D..@(.OG  
0005F250: 29 04 91 2A 00 00 00 EF 2B 00 00 00 13 2C A5 A5 ).*...+...  
0005F260: A5 A5 FF  
0005F270: FF  
0005F280: FF  
0005F290: FF  
0005F2A0: FF  
0005F2B0: FF  
0005F2C0: FF FF
```

Sequence of *0x2A*, *0x2B*, and *0x2C* commands

# SAVING LEAKED MEMORY INTO FILE



# REPEAT UNTIL YOU GET THE WHOLE MEMORY

```
virtualabs@virtubox:/tmp/dumps$ ls -ahl
total 3,2M
drwxr-xr-x 2 virtualabs virtualabs 32K 19 juin 17:49 .
drwxr-xr-x 7 virtualabs virtualabs 32K 20 juin 11:11 ..
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 08:37 firmware-dump-0x00000000.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 08:40 firmware-dump-0x00021480.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 08:42 firmware-dump-0x00042900.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 08:44 firmware-dump-0x00063d80.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 08:46 firmware-dump-0x00085200.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 08:47 firmware-dump-0x000a6680.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 08:52 firmware-dump-0x000c7b00.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 08:53 firmware-dump-0x000e8f80.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 08:56 firmware-dump-0x0010a400.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 09:02 firmware-dump-0x0012b880.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 09:03 firmware-dump-0x0014cd00.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 09:04 firmware-dump-0x0016e180.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 09:05 firmware-dump-0x0018f600.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 09:06 firmware-dump-0x001b0a80.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 09:07 firmware-dump-0x001d1f00.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 09:08 firmware-dump-0x001f3380.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 09:09 firmware-dump-0x00214800.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 09:10 firmware-dump-0x00235c80.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 09:21 firmware-dump-0x00257100.bin
-rw-r--r-- 1 virtualabs virtualabs 134K 17 mars 09:22 firmware-dump-0x00278580.bin
```

# LAST STEP : MERGE DUMPS INTO A SINGLE FILE

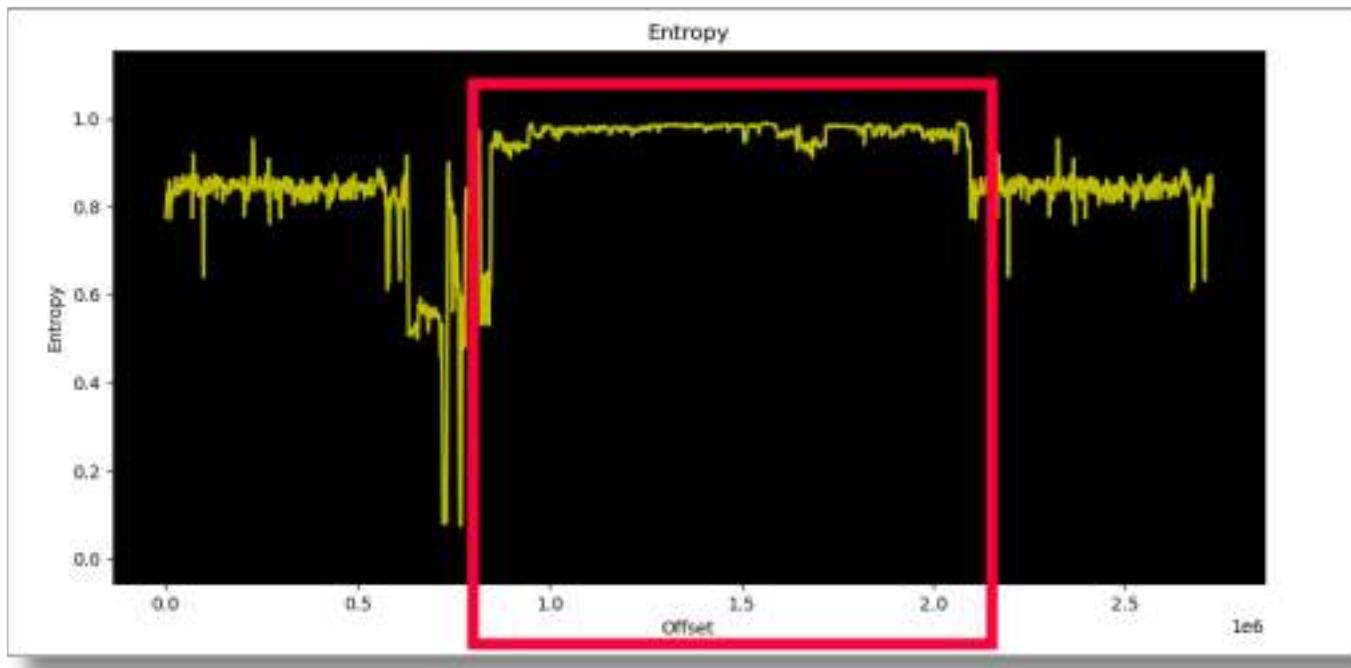
```
virtualabs@virtubox:/tmp/dumps$ xxd -l 300 final-flash-image.bin
00000000: 69f8 b6b1 2001 e001 ac4f 0c00 83ff 0000 i.... ....0.....
00000010: 6170 705f 6172 6561 5f68 6561 6400 ffff app_area_head...
00000020: c320 892d 2001 0000 fc4a 0c00 82ff 0000 . .- ....J.....
00000030: 6170 702e 6269 6e00 ffff ffff ffff ffff app.bin.....
00000040: c7b1 d316 1c4c 0c00 9003 0000 82ff 0000 .....L.....
00000050: 6366 675f 746f 6f6c 2e62 696e 00ff ffff cfg_tool.bin....
00000060: c9f9 01ff 0070 0d00 0020 0100 1281 0000 .....p.... ...
00000070: 564d 00ff ffff ffff ffff 0000 0100 VM.....
00000080: 2168 ffff 0000 0000 0070 0d00 9282 0000 !h.....p.....
00000090: 5052 4354 00ff ffff ffff 434f 4445 PRCT.....CODE
000000a0: 8c6e ffff 0090 0e00 0010 0000 9281 0000 .n.....
000000b0: 4254 4946 00ff ffff ffff 4155 544f BTIF.....AUTO
```

# HURRAY, WE HAVE A FIRMWARE !

- > Successfully leaked through unexpected data displayed on screen 
- > Required some soldering, programming and BLE-fu
- > That's 2 MiB of data !

LONG IS THE ROAD ...

# WHAT HAVE WE JUST DUMPED ?



That's never a good sign when entropy is  $\sim= 1.0$

# NOR FLASH STRUCTURE

## General layout

Example of a layout for bluetooth audio chips e.g. BR23/BR25/BR28/etc:



JieLi's new firmware file structure

# WHAT DO WE HAVE HERE ?

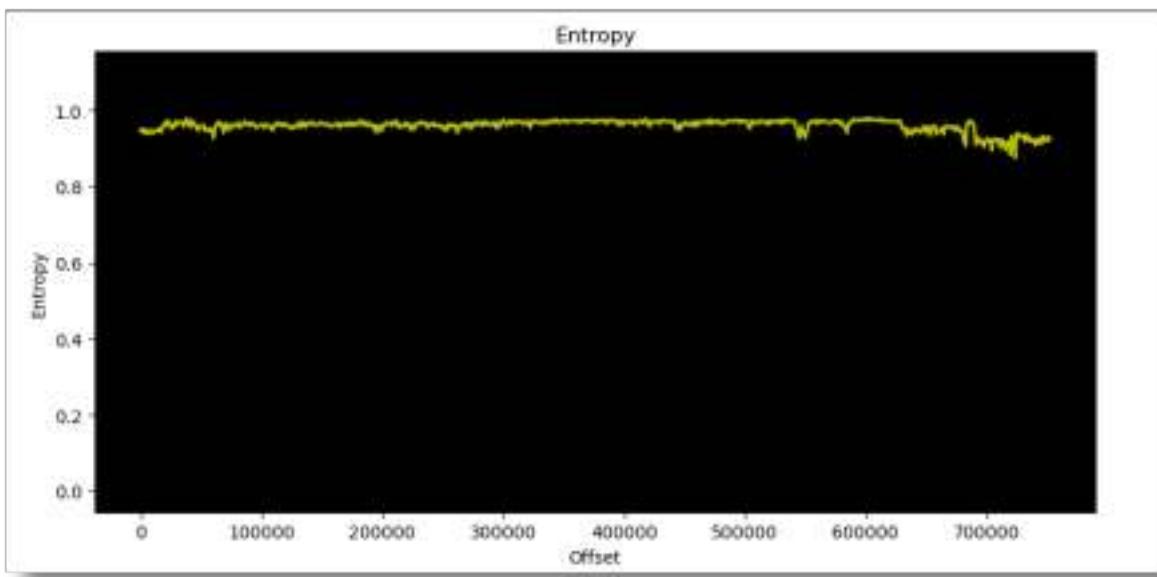
	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F		
0x0	69 F8 B6 B1 20 01 E0 01 AC 4F 0C 00 83 FF 00 00	0	JieLi New Filesystem
0x10	a p p _ a r e a _ h e a d 00 FF FF	0+32	App. area Header
0x20	C3 20 89 2D 20 01 00 00 FC 4A 0C 00 82 FF 00 00	20	File #0
0x30	a p p . b i n 00 FF FF FF FF FF FF FF FF FF	20+2	File header CRC
0x40	C7 B1 D3 16 1C 4C 0C 00 90 03 00 00 82 FF 00 00	22+2	Content CRC
0x50	c f g _ t o o l . b i n 00 FF FF FF	24+4	Content offset
0x120	80 FF 06 0E 03 00 EE FF B0 53 00 00 ED FF B0 53	28+4	Content size
0x130	00 00 D8 E8 07 00 C0 FF F4 4B EC 01 C1 FF 00 00	2C+1	Flags
0x140	80 00 C2 FF 00 00 00 00 A2 A2 12 03 03 05 93 05	2D+1	Reserved
0x150	02 FC FB 01 D4 E8 07 00 C3 FF 00 54 00 00 41 20	2E+2	Index
0x160	C2 FF B8 5B 01 00 A2 A2 02 03 B1 05 F2 5D C3 FF	30+12	File name
0x170	00 CC 02 00 41 20 C2 FF 00 00 00 00 A2 A2 02 03	3C+4	Mode
0x180	B1 05 F2 5D C4 FF 00 00 00 00 C1 FF 94 00 EC 01		
0x190	C2 FF 60 4B 00 00 A2 A2 12 03 13 05 C3 05 F2 5C		
0x1A0	64 E0 00 2B 72 E1 40 2F 64 E0 80 2B 80 FF C8 33		
0x1B0	08 00 80 FF 62 2E 05 00 C0 FF BA 3A E5 01 D0 00		
0x1C0	64 E0 00 0B 00 17 80 00 64 E0 00 0B 61 E1 40 0F		

Generated with Corkami's SBud v2

# FILES !

```
virtualabs@virtubox:/tmp/rootfs$ ls -l
total 2836
-rw-r--r-- 1 virtualabs virtualabs 805628 20 juin 11:10 app.bin
-rw-r--r-- 1 virtualabs virtualabs 4096 20 juin 11:10 BTIF
-rw-r--r-- 1 virtualabs virtualabs 912 20 juin 11:10 cfg_tool.bin
-rw-r--r-- 1 virtualabs virtualabs 4096 20 juin 11:10 EXIF
-rw-r--r-- 1 virtualabs virtualabs 753664 20 juin 11:10 FATFSI
-rw-r--r-- 1 virtualabs virtualabs 880640 20 juin 11:10 PRCT
-rw-r--r-- 1 virtualabs virtualabs 375068 20 juin 11:10 USERIF
-rw-r--r-- 1 virtualabs virtualabs 73728 20 juin 11:10 VM
virtualabs@virtubox:/tmp/rootfs$ 
```

# STILL SOMETHING WRONG WITH SOME FILES ...



Entropy of *FATFSI*

# JIELI'S "CRYPTO"

## "ENC" chiper

The cipher used by the [ENC](#) peripheral is of a simple stream chiper type, that basically uses the CRC16-CCITT shift register logic (with a polynomial  $x^{16}+x^{12}+x^5+1$  aka 0x1021) as the LFSR logic.

Each byte is XOR-ed with the low bits of the register (initialized with the "key" value, which is 16 bits long), and then the register is updated with the aforementioned LFSR logic.

Here's an example C snippet:

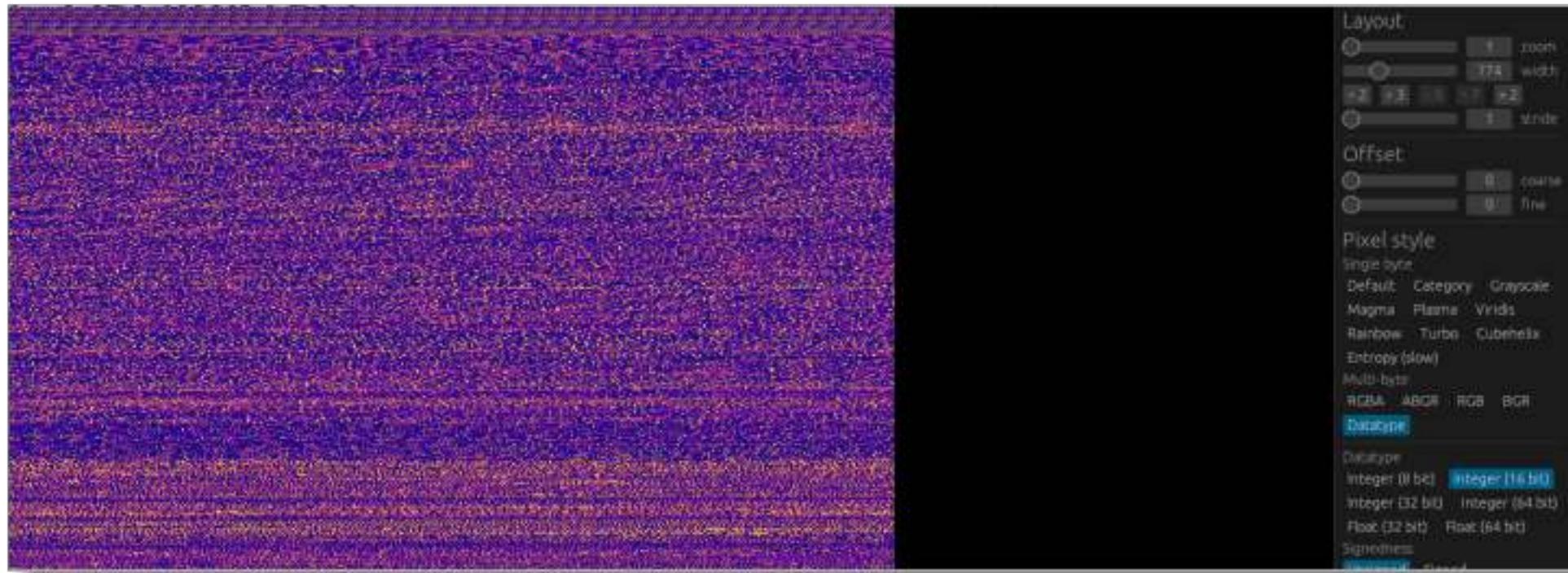
```
uint16_t jl_enc_cipher(uint8_t *data, int len, uint16_t key) {
    while (len--) {
        *data++ ^= key;
        key = (key << 1) ^ (key >> 15 ? 0x1021 : 0);
    }
    return key;
}
```



# FATFSI GOT DECRYPTED

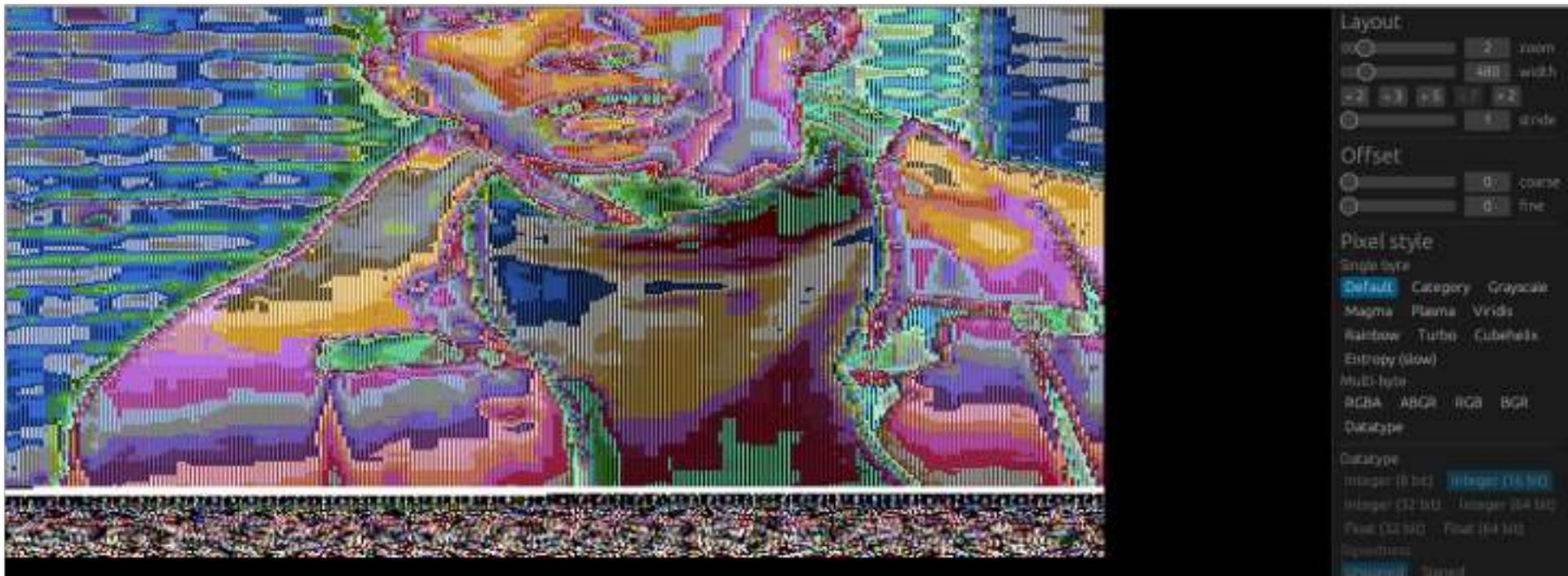
```
virtualabs@virtubox:/tmp/rootfs$ xxd -l 256 FATFSI.dec.bin
00000000: 7cf0 0000 b9f0 0000 f6f0 0000 33f1 0000 | .....3...
00000010: 6df1 0000 aaf1 0000 e7f1 0000 27f2 0000 m.....'...
00000020: 67f2 0000 a7f2 0000 e4f2 0000 24f3 0000 g.....$...
00000030: 61f3 0000 a1f3 0000 e1f3 0000 21f4 0000 a.....!...
00000040: 61f4 0000 9ef4 0000 def4 0000 1ef5 0000 a......
00000050: 5ef5 0000 89f5 0000 b4f5 0000 f4f5 0000 ^......
00000060: 31f6 0000 71f6 0000 b1f6 0000 f1f6 0000 1...q.....
00000070: 2bf7 0000 6bf7 0000 abf7 0000 ebf7 0000 +...k.....
00000080: 19f8 0000 59f8 0000 96f8 0000 d0f8 0000 ....Y.....
00000090: 0df9 0000 4af9 0000 8af9 0000 c4f9 0000 ....J.....
000000a0: 01fa 0000 41fa 0000 81fa 0000 befa 0000 ....A.....
000000b0: fefa 0000 3efb 0000 7efb 0000 befb 0000 ....>...~.....
000000c0: ffbfb 0000 3bfc 0000 7bfc 0000 bbfc 0000 ....;...{.....
000000d0: fbfc 0000 3bfd 0000 78fd 0000 b8fd 0000 ....;...x.....
000000e0: f8fd 0000 38fe 0000 78fe 0000 b8fe 0000 ....8...x.....
000000f0: f8fe 0000 38ff 0000 78ff 0000 b8ff 0000 ....8...x.....
virtualabs@virtubox:/tmp/rootfs$ 
```

# STILL SOME GARBAGE

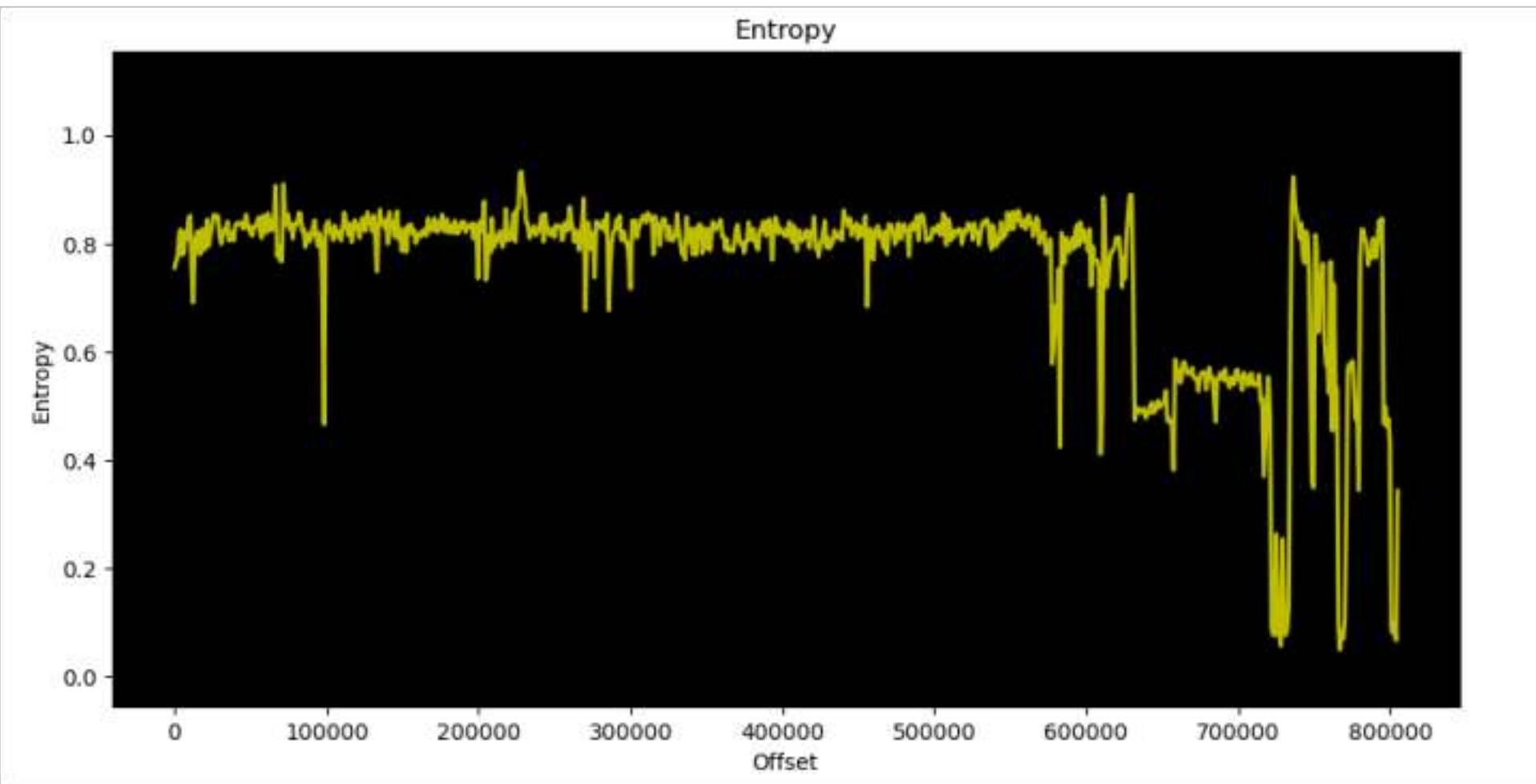


Data viz with *Binocle*

# FOUND A FUNNY LOST SECTOR



# GOOD NEWS: APPLICATION IS NOT ENCRYPTED !



# CODE REVERSE-ENGINEERING

IS IT ARM ? MIPS ?  
NO, ITS PI32 (V2) !

## pi32

JieLi's own custom architecture that is heavily based off the Analog Devices' Blackfin core.

- ELF machine ID: 0xF0 (240)
- Little endian
- 32-bit architecture
- DSP-like instruction set
- 16 general purpose registers
- 64-bit multiply accumulator
- ...



# DISASSEMBLING APP WITH GHIDRA

```
*****  
undefined FUN_01e17186()  
undefined    r0:1      <RETURN>  
FUN_01e17186                                XREF[1] :  FUN_01e0ac94:01e0ac9c(c)  
01e17186 78 04      push    {0x8}  
01e17188 0c 9c      add     r4,r0,#0x1c  
01e1718a 46 e0 9c 01    movz    r6,#0x19c  
01e1718e c7 ff 20      mov     r7,#0x11320  
    13 01 00  
01e17194 c8 ff a8      mov     r8,#0x10f1a8  
    f1 10 00  
01e1719a d8          ??      D8h  
01e1719b ec          ??      ECh  
01e1719c 7a          ??      7Ah    z  
01e1719d 06          ??      06h  
01e1719e 41          ??      41h    A  
01e1719f 16          ??      16h  
01e171a0 80          ??      80h  
01e171a1 ea          ??      EAh  
01e171a2 6b          ??      68h    k  
01e171a3 cc          ??      CCh
```



# WAIT, WE HAVE AN SDK WITH OBJDUMP



```
sdk.elf: file format ELF32-pi32v2
```

```
Disassembly of section .text:
```

```
text_code_begin:
```

```
1e00100: 81 ea 29 bd      call 227922 <boot_info_init : 1e37b56 >
1e00104: ee ff 10 a0 00 00  sp = 40976
1e0010a: ed ff 10 a0 00 00  ssp = 40976
1e00110: d8 e8 07 00      [ --sp ] = {r2-r0}
1e00114: c0 ff f0 cb eb 01  r0 = 32230384 <psram_laddr : 1ebcbf0 >
1e0011a: c1 ff 00 00 80 00  r1 = 8388608 <psram_vaddr : 800000 >
1e00120: c2 ff 00 00 00 00  r2 = 0 <test_encode_main.c : 0 >
1e00126: a2 a2           r2 = r2 >> 2
1e00128: 12 03           rep 4 r2 {
1e0012a: 03 05           r3 = [ r0++=4 ]
1e0012c: 93 05           [ r1++=4 ] = r3
                           }
1e0012e: 02 fc fb 01      if (r2 > 0) goto -10 <text_code_begin+0x28 : 1e00128 >
1e00132: d4 e8 07 00      {r2-r0} = [sp++]
```

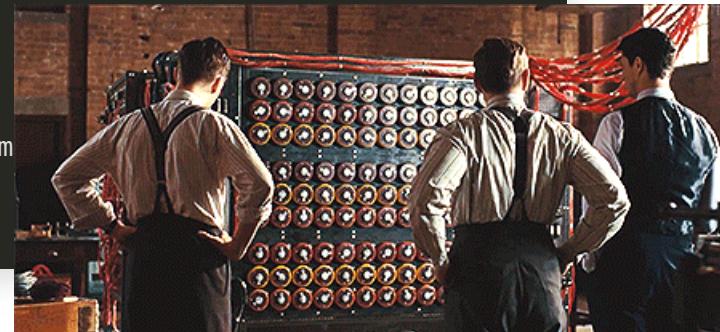
Not sure if it really helps ...

# SLEIGH OF HAND

```
# Recursively generate our push registers
pshmapregs: pshmapregs^pshmreg^msep is counter<15 & mread & msep & pshmreg & next & pshmapregs
  { build pshmapregs; build pshmreg; }
pshmapregs: pshmreg is counter=15 & mread & pshmreg {}
pshmap: pshmapregs is pshmapregs [bitset=0; counter=0; sep=0;] {}

# Recursively generate our pop registers
popmapregs: popmapregs^popmreg^msep is counter<15 & mread & msep & popmreg & next & popmapregs
  { build popmreg; build popmapregs; }
popmapregs: popmreg is counter=15 & mread & popmreg {}
popmap: popmapregs is popmapregs [bitset=0; counter=0; sep=0;] {}

# Recursively generate our pop special registers
popmapsregs: popmapsregs^popmsreg^msep is counter<3 & msread & msep & popm
  { build popmsreg; build popmapsregs; }
popmapsregs: popmsreg^msep is counter=3 & msep & msread & popmsreg {}
popsrmap: popmapsregs is popmapsregs [bitset=0; counter=0; sep=0;] {}
```



# THATS WAY BETTER



```
----- FUNCTION -----
undefined FUN_01e17186()
assume elsebranch = 0x8
assume group = 0x7
assume ifblock = 0x8
assume thenbranch = 0x8
undefined    r0:1      <RETURN>
FUN_01e17186

#ie17186 78 04      push    {r0,r8,r7,r6,r5,r4}
#ie17188 9c 9c      add     r4,r0,#0x1c
#ie1718a 46 e0 9c 81 movz    r6,#0x19c
#ie1718c c7 ff 28      mov     r7,#0x11328
        13 01 00
#ie17194 c8 ff a8      mov     r8,#0x10f1a8
        f1 10 00

LAB_01e1719a:
#ie1719a d8 ec 7a 06    lw      r0,[r7+r0=>DAT_000452f0<>2]
#ie1719e 41 10      mov     r1,r4
#ie171a0 80 ea 6b cc    call    FUN_01e38a7a
#ie171a4 05 16      mov     r5,r0
#ie171a6 85 58      jnz    r5,LAB_01e171d8
#ie171a8 64 e0 00 0b      mov     r0,icfg
#ie171ac 60 e1 40 8f      and    r8,r0,#0x300
#ie171b0 45 20      mov     r5,#0x0
#ie171b2 01 ff 00      jne    r0,#0x300,LAB_01e171e6
        03 17 00
#ie171b8 d8 ec 80 00    ldw    r0,r8=>DAT_0010f1a8,#0x0
#ie171bc 00 f8 13 0c      je     r0,0x6,LAB_01e171e6
#ie171c0 64 e0 00 0b      mov     r0,icfg

----- XREF -----
2 /* WARNING: Globals starting with '_' overlap smaller symbols at the same address */
3
4 undefined4 * FUN_01e17186(int param_1)
5
6 {
7     undefined4 *puVar1;
8     int iVar2;
9     undefined4 *puVar3;
10    uint in_icfg;
11
12    while( true ) {
13        puVar1 = (undefined4 *)FUN_01e38a7a(_DAT_000452f0,param_1 + 0x1c);
14        if (puVar1 != (undefined4 *)0x0) {
15            iVar2 = 7;
16            puVar3 = puVar1;
17            do {
18                *puVar3 = 0;
19                puVar3 = puVar3 + 1;
20                iVar2 = iVar2 + -1;
21            } while (iVar2 != 0);
22            puVar1[4] = puVar1 + 4;
23            puVar1[5] = puVar1 + 4;
24            return puVar1;
25        }
26        if ((in_icfg & 0x300) != 0x300) {
27            return (undefined4 *)0x0;
28        }
29        if (_DAT_0010f1a8 == 6) {
30            return (undefined4 *)0x0;
31        }
32        if ((in_icfg & 0xff) != 0) break;
33        iVar2 = FUN_01e8137e(0x12824,0);
34        if (iVar2 != 0) {
            return _main_Conditional_8();
        }
    }
}
```

C:\Decompile\FUN\_01e17186 \* Bytes: app.bin \* Defined Strings \*

# IS IT REALLY FAKE ?

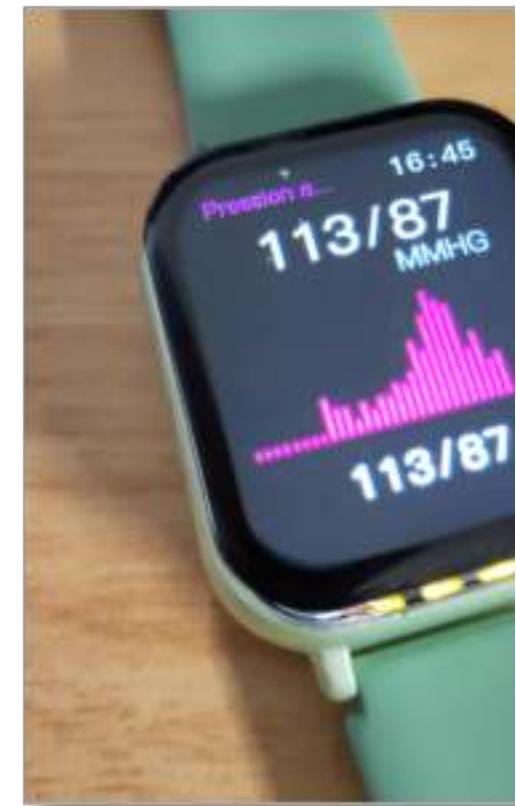
```
01e02b40 73 63 37      ds      "sc7a20"  
                      61 32 30 00  
01e02b47 00 00 00 00      addr     00000000  
01e02b4b 00 00 00 00      addr     00000000  
01e02b4f 00 00 00 00      addr     00000000  
01e02b53 00 00 00 00      addr     00000000  
01e02b57 00              ??      00h  
01e02b58 3c 6a e4 01      addr     FUN_01e46a3c  
01e02b5c 98 6a e4 01      addr     LAB_01e46a98  
01e02b60 e4 8f e4 01      addr     LAB_01e48fe4  
01e02b64 04 90 e4 01      addr     LAB_01e49004  
01e02b68 44 90 e4 01      addr     LAB_01e49044  
01e02b6c 92 90 e4 01      addr     LAB_01e49092  
01e02b70 68 72 5f      ds      "hr_fake"  
                      66 61 6b  
                      65 00  
01e02b78 00 00 00 00      addr     00000000  
01e02b7c 00 00 00 00      addr     00000000  
01e02b80 00 00 00 00      addr     00000000  
01e02b84 ea 90 e4 01      addr     LAB_01e490ea  
01e02b88 00 00 00 00      addr     00000000  
01e02b8c 24 91 e4 01      addr     FUN_01e49124  
01e02b90 d6 91 e4 01      addr     FUN_01e491d6
```

HR for Heart Rate, and fake says it all 

# FOUND SOME INTERESTING STRINGS

```
if ((DAT_00015a0d == '\x01') || (DAT_00015a0e == '\x01')) {  
    _sprintf(&s_bp_systolic, s_%03d/%02d_@1eb8740, (uint)blood_pressure_systolic,  
            (uint)blood_pressure_diastolic);  
    puVar1[2] = 0xffff;  
    puVar1[1] = &s_bp_systolic;  
    *puVar1 = 0x23;  
    draw_text(param_1, 0x193, 0x30, 0x31);  
    _sprintf(&s_bp_diastolic, s_%03d/%02d_@1eb8740, (uint)blood_pressure_systolic,  
            (uint)blood_pressure_diastolic_);  
    *(undefined8 *) (puVar1 + 1) = 0xffff0000c2a0;  
    *puVar1 = 0x23;  
    draw_text(param_1, 0x1b9, 0x42, 0xe4);
```

AAA/BB



# FOLLOWING THE WHITE RABBIT ...

```
if ((CONCAT22(uVar15, CONCAT11(uVar14, uVar10)) == 2) &&
    (4 < CONCAT22(uVar11, CONCAT11(uVar6, uVar8)))) {
    uVar2 = randint();
    blood_pressure_systolic = ((char)uVar2 - (char)((int)uVar2 / 0x10 & 0xffU) << 4) + 110;
    uVar2 = randint();
    blood_pressure_diastolic = (char)uVar2 + (char)((int)uVar2 / 0x14) * -0x14 + 70;
```

```
uint8_t seed = randint() & 0xff;
blood_pressure_systolic = (seed - seed/16)*16 + 110;
```

$$110 \leq \text{blood\_pressure\_systolic} \leq 125$$

# PRNG

```
5 uint32_t randint(void)
6
7 {
8     int rand_state;
9     longlong lVar1;
10    uint uVar2;
11
12    rand_state = (int)*(undefined8 *)(_random_state + 0xa04);
13    lVar1 = (longlong)rand_state * 0x4c957f2d;
14    uVar2 = rand_state + 0x5851f42d + (int)((ulonglong)lVar1 >> 0x20) +
15        (int)((ulonglong)*(undefined8 *)(_random_state + 0xa04) >> 0x20) * 0x4c957f2d;
16    *(ulonglong *)(_random_state + 0xa4) = CONCAT44(uVar2,(int)lVar1 + 1);
17    return uVar2 & 0x7fffffff;
18 }
```

# SAME FOR HEART RATE !

```
if ((4 < DAT_00011548) && (DAT_00054834 == 1)) {
    uVar2 = randint();
    heart_rate_current = ((char)uVar2 - (char)((int)uVar2 / 0x10 & 0xFFU) << 4)) + 0x41;
    if (heart_rate_max <= heart_rate_current) {
        heart_rate_max = heart_rate_current;
    }
    uVar9 = 0;
    uVar17 = 0;
    bVar16 = heart_rate_min;
    if (heart_rate_current <= heart_rate_min) {
        bVar16 = heart_rate_current;
    }
    bVar12 = heart_rate_current;
    if (heart_rate_min != 0) {
        bVar12 = bVar16;
    }
}
```

65 <= heart\_rate\_current <= 80

OH NO, THIS SH\*T IS REAL !

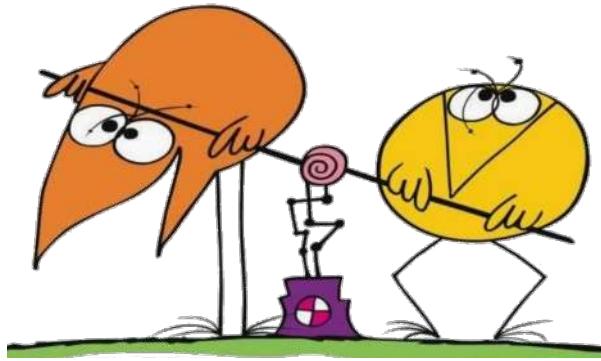


# FAKE VS. "REAL" MEASURES

- > Randomly generated:
  - Heart rate
  - Blood pressure
  - SP02
- > No evidence of random:
  - Footsteps counter
  - Sleep monitoring

# CONCLUSION

WELL, THERE'S SO MUCH TO SAY ABOUT THIS...



"TRYING AGAIN AND AGAIN EVENTUALLY LEADS TO SUCCESS.  
THE MORE WE FAIL THE GREATEST CHANCE WE HAVE TO SUCCEED."  
- SHADOK PROVERB

# SUMMARY OF OUR JOURNEY

# SUMMARY OF OUR JOURNEY

That's too much work to make sure it's a scam !

# SUMMARY OF OUR JOURNEY

That's too much work to make sure it's a scam !

Full stop.

# KEY TAKEAWAYS

# KEY TAKEAWAYS

- > Bluetooth addresses not randomized: tracking is possible

# KEY TAKEAWAYS

- > Bluetooth addresses not randomized: tracking is possible
- > No *real* authentication: anybody can upload a rogue firmware or dial

# KEY TAKEAWAYS

- > Bluetooth addresses not randomized: tracking is possible
  - > No *real* authentication: anybody can upload a rogue firmware or dial
  - > No *sensor*. measured values are randomly generated\*
- \* Except maybe for footsteps and sleep monitoring

# KEY TAKEAWAYS

- > Bluetooth addresses not randomized: tracking is possible
  - > No *real* authentication: anybody can upload a rogue firmware or dial
  - > No sensor. measured values are randomly generated\*
  - > Bad input validation: easy to remotely DoS any watch
- \* Except maybe for footsteps and sleep monitoring

# JIELI'S USB PROGRAMMER ?

- > Received weeks after we got the firmware out of the watch
- > Too much work, did not have time to test it 
- > Also got a devboard, but that's tricky to use

# SO, WHAT'S NEXT ?

- > Firmware modification and upload through OTA
- > 100% of Pi32v2 instructions supported in Ghidra
- > Asking Gifi for a refund ?

## ARTICLE 8 - DROIT APPLICABLE - RÉCLAMATIONS - LITIGES - MEDIATION

Le droit français s'applique aux présentes conditions de vente.

Pour toute réclamation vous pouvez vous adresser au service clients à l'adresse postale,  
**GIGAMARKET Zone Industrielle la Barbière - 47300 Villeneuve sur Lot** ; nous contacter au :  
05 53 40 54 68 ou par mail à l'adresse [serviceclient@gifi.fr](mailto:serviceclient@gifi.fr).

Conformément à l'article L. 612-1 du Code de la consommation, vous pouvez, si votre réclamation n'a pas abouti, recourir gratuitement au service de médiation SAS MEDIATION SOLUTION dont nous relevons :

Email : [contact@sasmediationsolution-conso.fr](mailto:contact@sasmediationsolution-conso.fr)

# THANK YOU, Q/A TIME !

Xilokar

 [@xilokar@mamot.fr](mailto:@xilokar@mamot.fr)

 <https://blog.xilokar.info>

 [xilokar@xilokar.info](mailto:xilokar@xilokar.info)

Virtu

 [@virtualabs@mamot.fr](mailto:@virtualabs@mamot.fr)

 [dcauquil@quarkslab.com](mailto:dcauquil@quarkslab.com)

