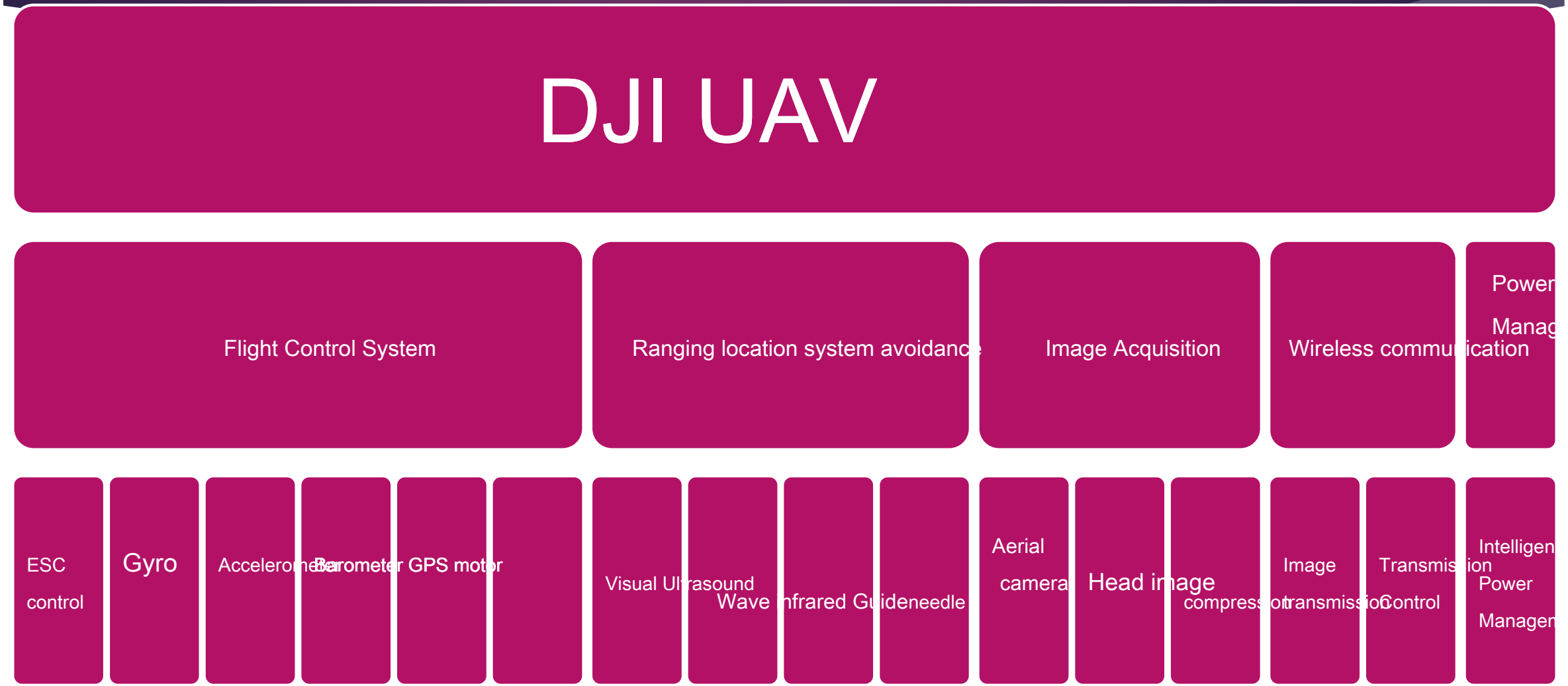
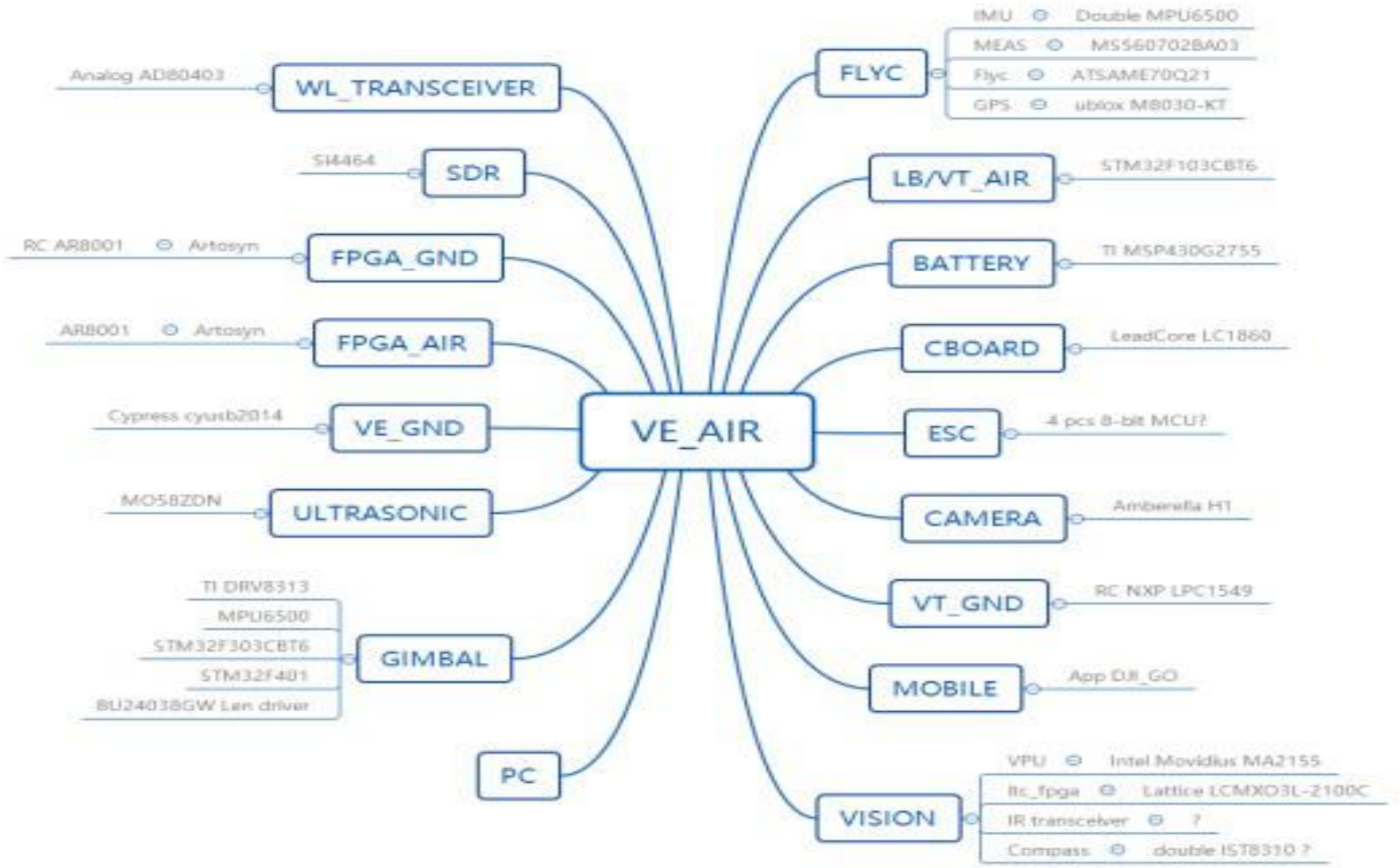
How to hack the drone

Xie Jun 2017.11.18

UAV is a complicated systematic project



DJI P4P Architecture



DJI Architecture - Communication between modules

- Two bytes of the communication source or destination address

Subsystem Number	functions Number

E.g 03 He represents flight control, 06 Show 6 Chant function

➢ A total of no more than 32 Sub-systems

➢ Channel mode

  ➢ local, IP, WL, UART, I2C, SPI, CAN, ADB

  ➢ HPI, USB, IAP2

➢ letter of agreement

  ➢ Logic

  ➢ V0

  ➢ V1

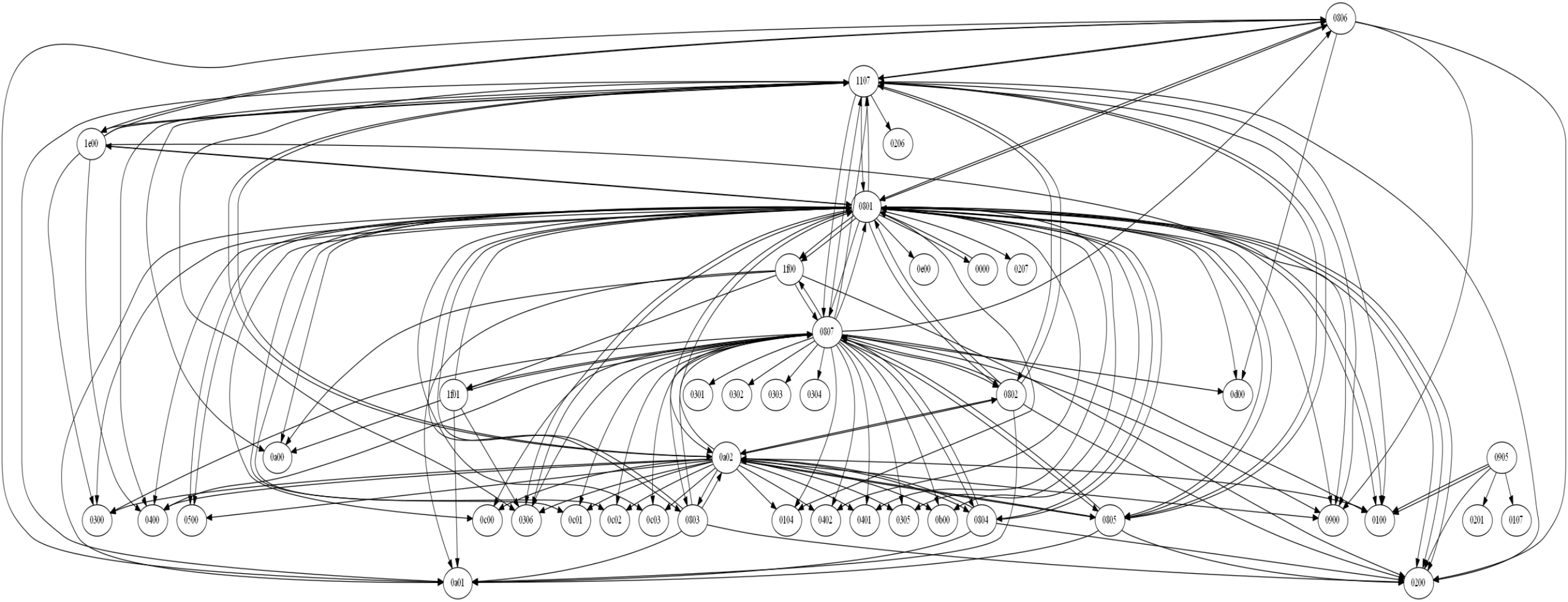
  ➢ NAL

  ➢ MAVLink

```
; "whoami"   00
; "camera"   01
; "mobile"   02
; "flight"   03
; "gimbal"   04
; "cboard"   05
; "rc"       06
; "network"  07
; "ve_air"   08
; "vt_air"   09
; "pc"       0a
; "battery"  0b
; "esc"      0c
; "ve_gnd"   0d
; "vt_gnd"   0e
; "s_to_p_air" 0f
; "s_to_p_gnd" 10
; "mvision" 0x11
; "bvision" 0x12
; "fpga_air" 0x13
; "fpga_gnd"  0x14
; "simulator" 0x15
; "null"
; "null"
; "null"
; "null"
; "mavlink"   0x1a
; "null"
; "glass"     0x1c
; "blackbox"  0x1d
; "test"      0x1e
; "all"       0x1f
```

```
0100 camera
0200 mobile
0400 gimbal
0500 cboard
0801 dji_sys
0802 ma2155
0803 ltc_fpga
0804 ultrasonic
0805 dji_vision
0806 dji_decoding
0807 test_diag
0a00 pc con
0a02 firmware update from pc
1107 ma2155 vision
1105 dji_flight for NFZ
0300 fly control
0301
0302
0303
0304
0305
0306
0900 LightBridge
0905 download service
0b00 battery
0c00 ESC 1
0c01 ESC 2
0c02 ESC 3
0c03 ESC 4
0d00 RC cyusb
0e00 RC LPC1549
1f00 all
```

DJI Architecture - Communication between modules



DJI Architecture - Communication between modules

- letter of agreement Logic & V1

  message v1 format

  Magic Header 1byte • 0x55

  Packet Len 10bits -> less than 0x400

  version 1byte & 4 -> 4 in P4 & P4P

  CRC8 cksum 1 byte -> crc8 (header [0: 3])

  Seq 2bytes -> random

  src ID **1byte -> 0A • 0A00**

  dst ID 1byte -> 0E -> 0E00

  Attr ID 1byte ->

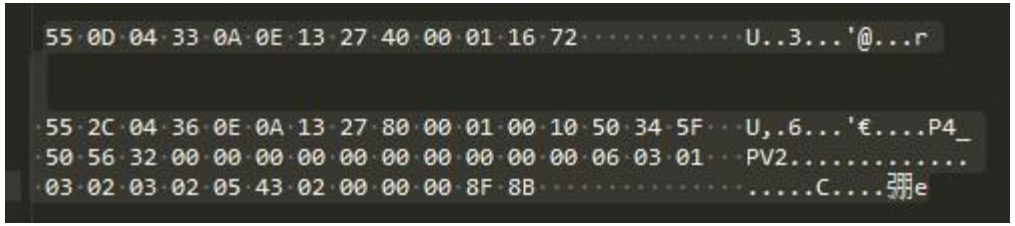
  cmd ID 2bytes -> 00 01 version request

  sub_cmd ID 1byte optional

  body len 2bytes optional

  body few bytes optional

  crc16 cksum 2bytes -> crc16 (pkt [: - 2])



DJI Architecture - Communication between modules

- letter of agreement Logic & V1

message Logic Format

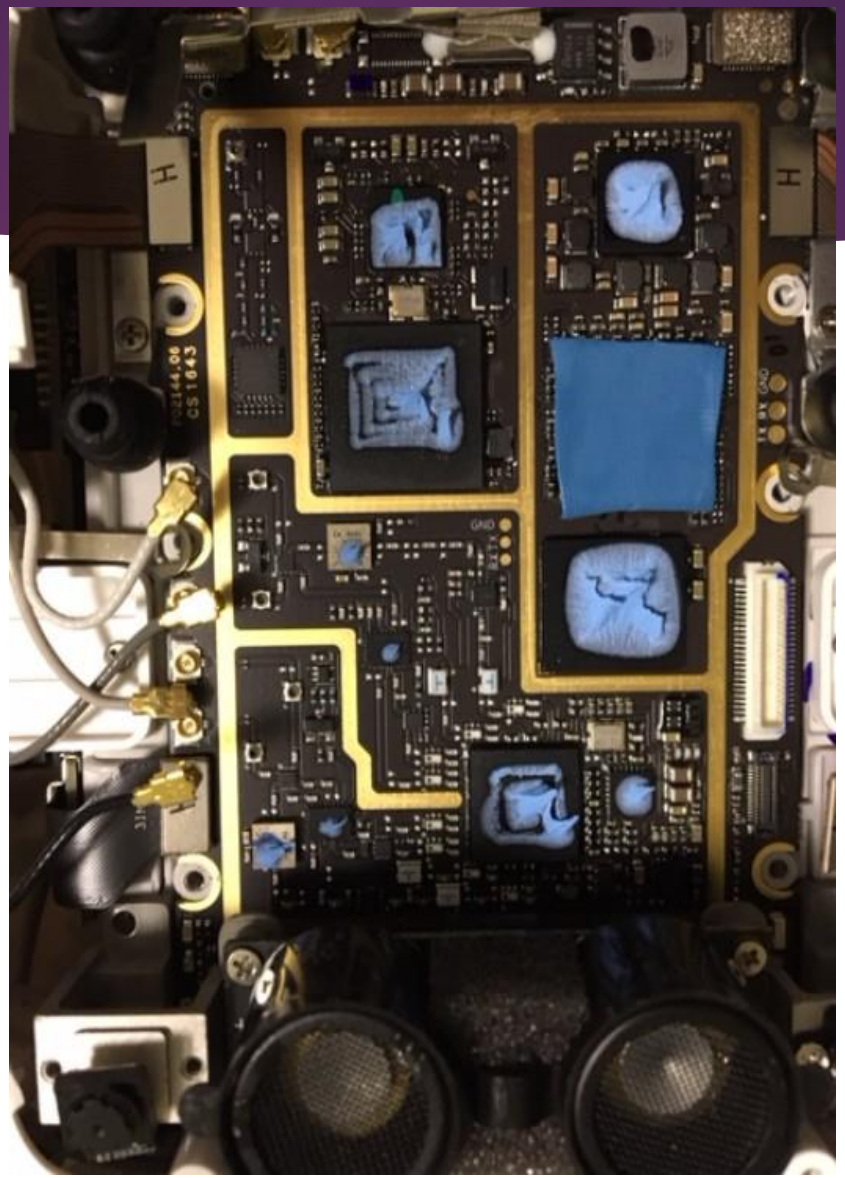
Magic header	4bytes	-> 21 12 ad de
src ID	2bytes	->
dst ID	2bytes	->
Seq	2bytes	->
Attr ID	1byte ->	
CMD ID	2bytes	->
body	few bytes ->	
pkt len	2bytes -> packet len-6	

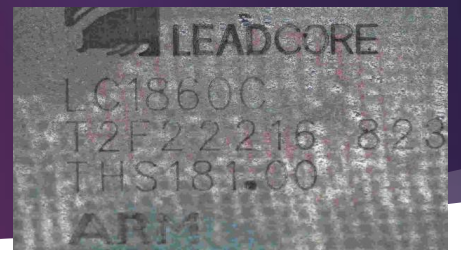
```
def gen_host(d):
    c=(d>>5)|((d&0x1f)<<8)
    return c

gen_host(0xf1)==0x1107
gen_host(0x09)==0x0900
```

CenterBoard

- CenterBoard MCU LC1860
- Vision MA2155 + Lattice LCMXO3L2100C
- Memory & Storeage
  Samsung
  eMCP (eMMc + DRAM)
  KMFJ20005A 4GB
- SiliCon LABS Si4464





Root UAV

why would root UAV?

UAV research more convenient, easy to study other target modules

CenterBoard MCU LC1860 Embedded Systems linux , With adb The

default is to open the key scripts Start_dji_system.sh , adb_en.sh

```
debug=false
grep production /proc/cmdline >> /dev/null
if [ $? != 0 ]; then
    debug=true  # engineering version, enable adb by default
else
    cmdline=`cat /proc/cmdline`
    temp=${cmdline##*board_sn=}
    board=${temp%% *}
    in_whitelist.sh $board
    if [ $? == 0 ]; then
        debug=true
    fi
fi

if $debug; then
    /system/bin/adb_en.sh
else
    setprop sys.usb.config rndis,mass_storage,bulk,acm
fi
```

```
#!/system/bin/sh

level=$1
: ${level:=NonSecurePrivilege}

# mark debug enable
mkdir -p /tmp/dji
echo $level > /tmp/dji/secure_debug

# init adb device serial
if [ -f /data/dji/cfg/adb_serial ]; then
serial=`cat /data/dji/cfg/adb_serial`
busybox printf "$serial" > /sys/class/android_usb/android0/iSerial
fi

setprop service.adb.root 1
setprop service.adb.tcp.port -1
setprop sys.usb.config rndis,mass_storage,bulk,acm,adb
busybox devmem 0xe10093d0 8 0x40    #enable uart
sleep 1
busybox udhcpd
```

how root ?

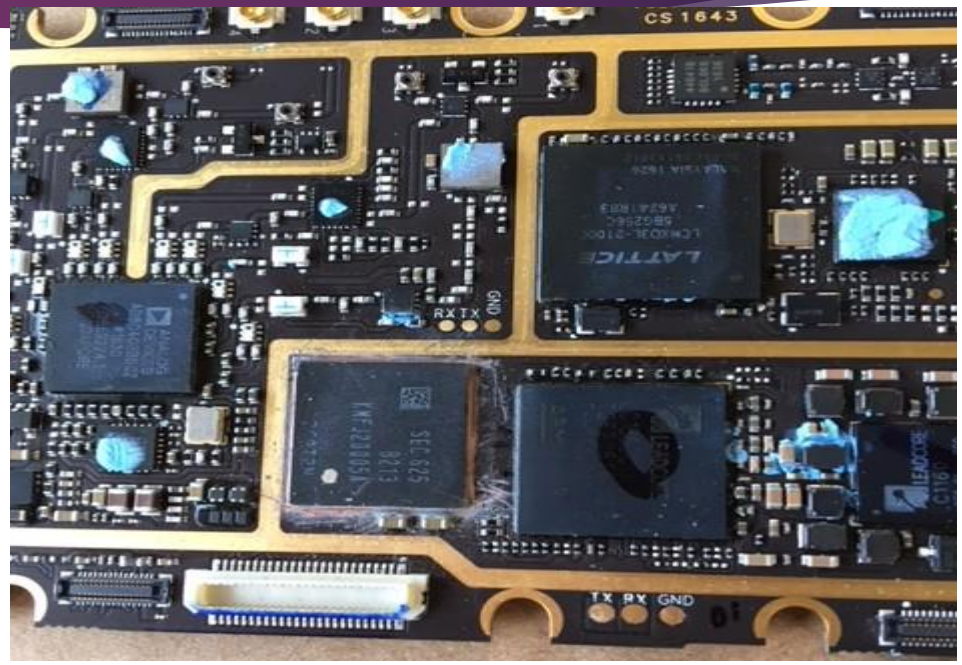
1. Software vulnerability to do adb_en.sh script

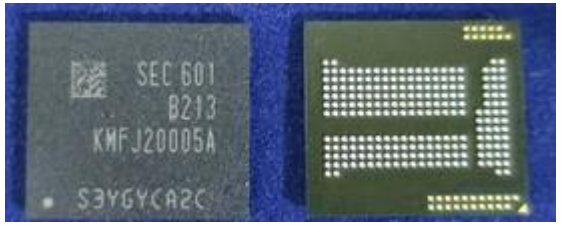
2. If you can modify start_dji_system.sh script

"Debug = true" Can be reached root purpose

Root UAV

- By modifying start_dji_system.sh Script reach root effect
- More generic root The method requires rigid-flex
- The goal is to modify the contents of memory inside

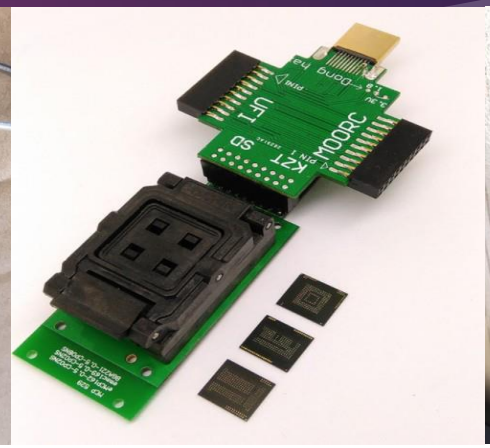


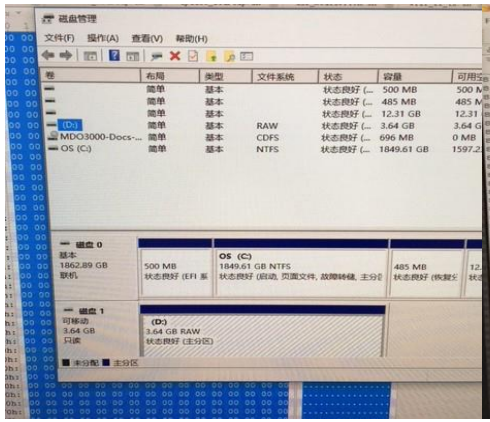
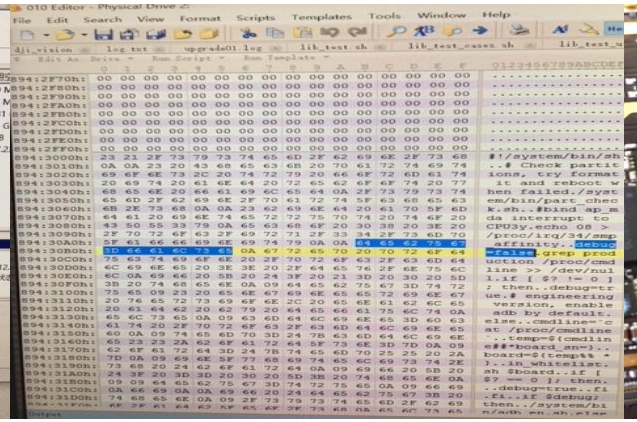
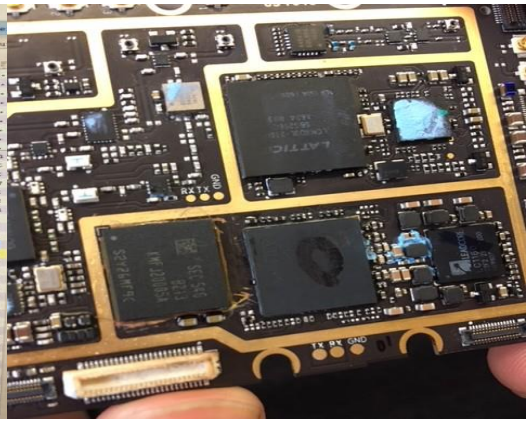


Root UAV

- The blowing air gun eMCP
- eMMc The card reader reads Raw data
- Find the right partition file storage
- Ext4 Mounting system Partition
- modify start_dji_system.sh file
- System img Write back Raw Data file
- eMMc Readers write back eMCP
- eMCP Chip soldered back seat

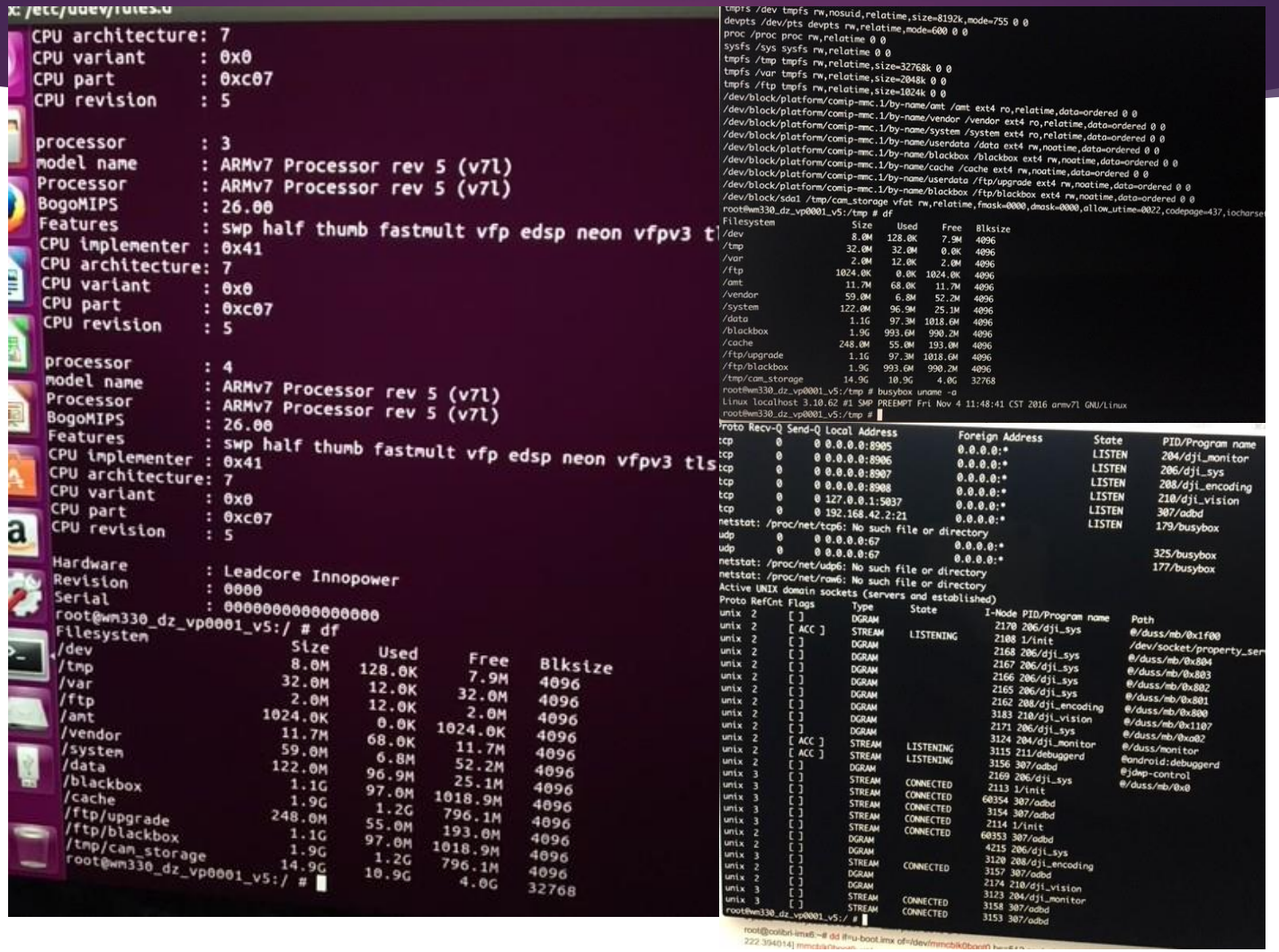




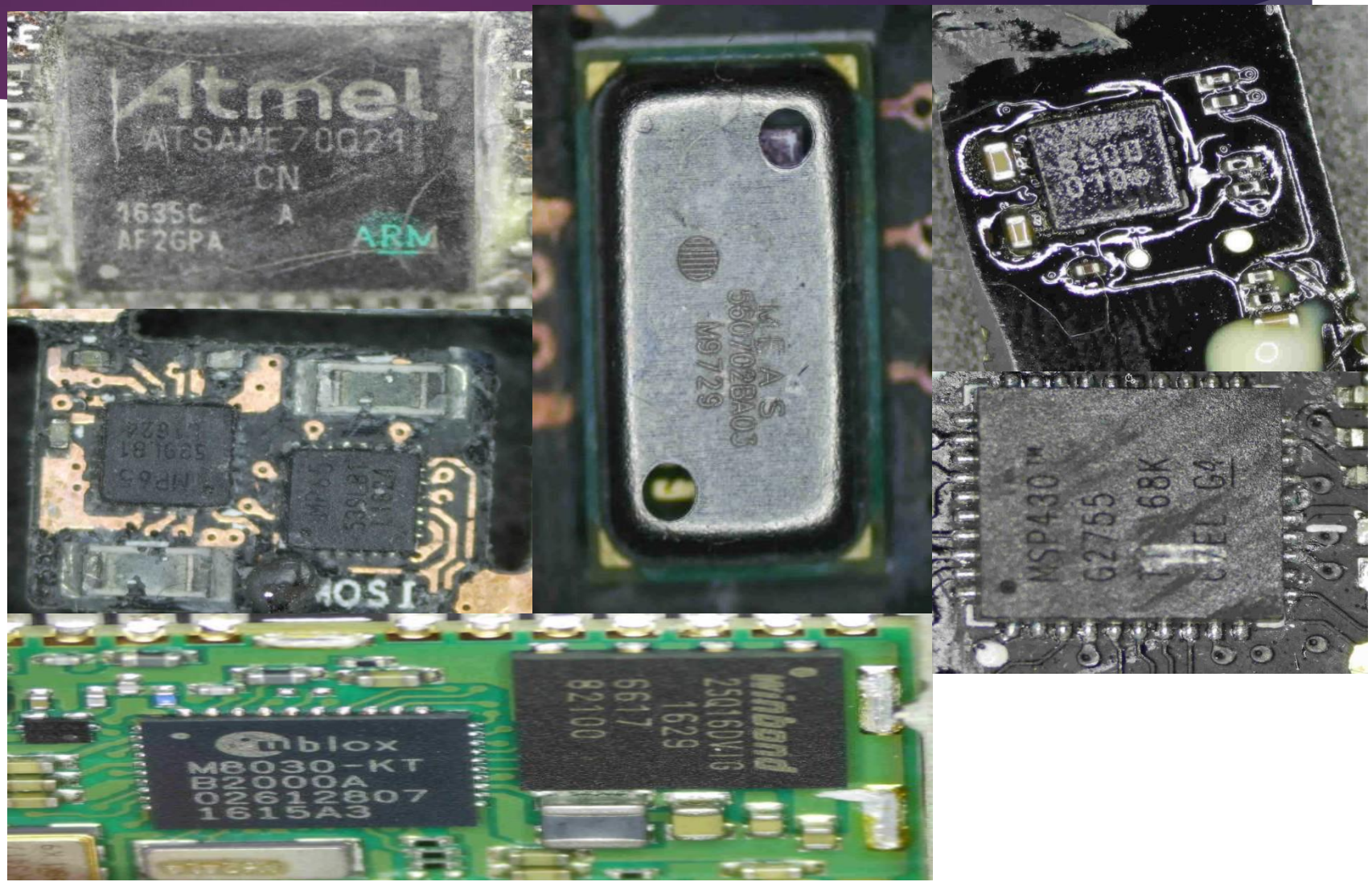
Root UAV

- finally



Flight Control System

- Flight Control MCU Atmel ATSAME 70Q21
- IMU MP6500 (Gyroscope + accelerometer)
- Barometer MEAS
  MS560702BA03
- Ublox GPS Module
- Double IST8310 Compass Module
- Smart Power
- ESC ESC

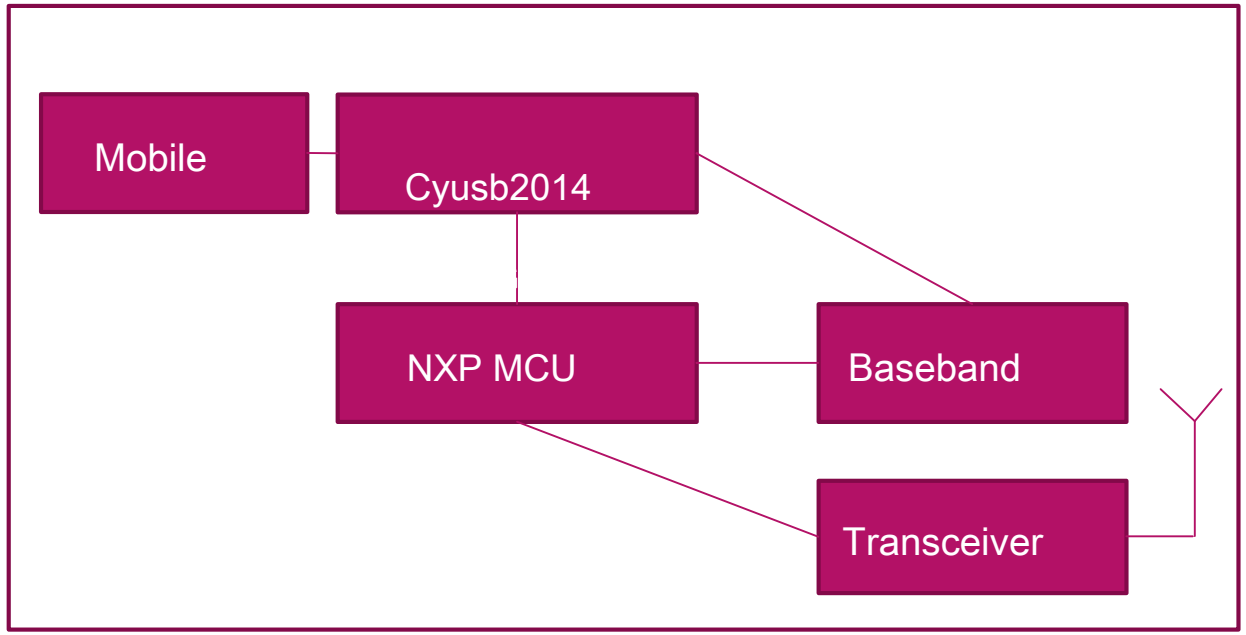


visual system

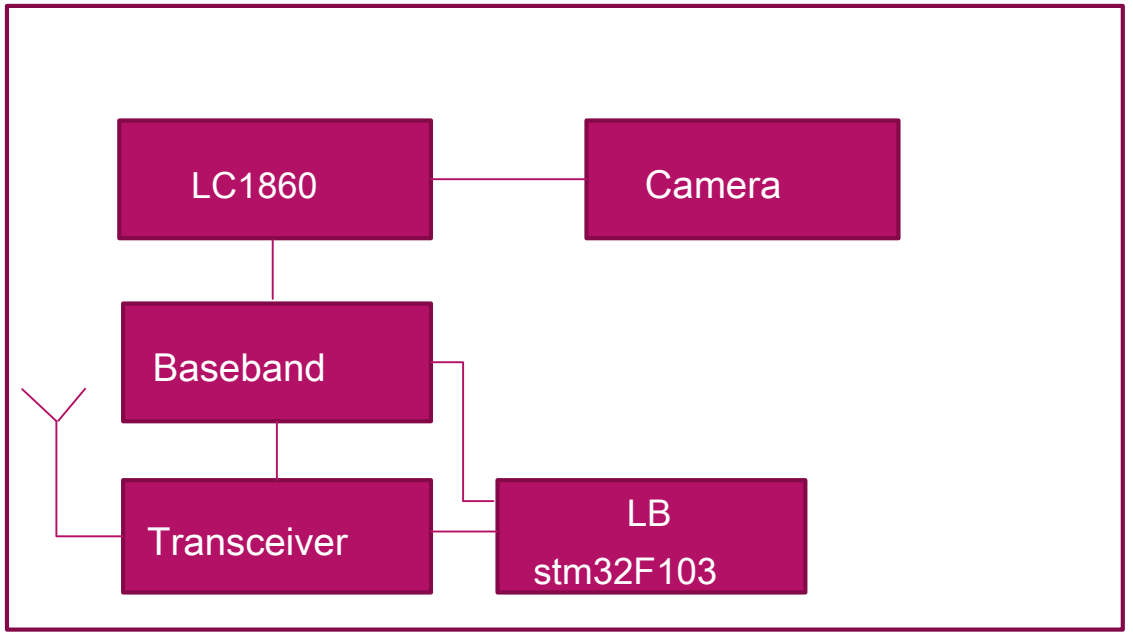
- MA2155 Depth visual processing
  - And the front obstacle recognition ranging
  - Character, attitude, water, ROI Recognition
- Binocular infrared obstacle avoidance
  - Infrared about recognition
- Ultrasonic wave
  - Below the obstacle detection

Wireless communication system architecture

RC



Aircraft



Wireless communication system architecture

- FIG NC & passage downstream channel
- OFDM Broadband transmission
- 1Mhz & 10Mhz bandwidth
- RC 1Tx & 2Rx signal path channel
- Air 1Tx & 2Rx signal path channel
- Transceiver AD80403 SPI config same as Ad9361
- CNC channel 2.4G 24 Hopping sequence, 5.8G 12 Hopping sequence
- CNC channel 2.4G 45 Frequency points, 5.8G 42 Frequency points
- Each switching frequency interval 14ms , 1s Switch 71 Frequency points

Operating Frequencies

Device operates within approved frequencies overlapping with the following cellular bands: LTE 255,Unlicensed NII-3 DOWN | LTE 46,TD Unlicensed DOWN |

Frequency Range	Power Output	Rule Parts
2.404-2.4788 GHz	566 mW	15C
5.727-5.8213 GHz	175 mW	15E

Wireless communication system architecture

- Pair & link
- The pairing code RC MCU ID generate
- by RC ID Hopping sequence generation

RC id generation algorithm

IAP_chip_ReadUID get 16 bytes

b1=crc8(uid,0x77)

b2=crc16(uid,0x3692)

b3=xor(uid,)

RC_id=[b2[0:1],b1,b3]

```
freq hop seed generation
4pro type==3

a=crc8(a1,4,0x33)
a1=[0xa7,0x9f,0x4e,0x09] ,rc id gened

table_58[1,5,0x0b,0x0d,0x11,0x13,0x17,0x19,0x1d,0x1f,0x25,0x29]
table_24[1,2,4,7,8,0x0b,0x0d,0x0e,0x10,0x11,0x13,0x16,0x17,0x1a,0x1c,0x1d,0x1f,0x20,0x22,0x25,0x26,0x29,0x2b,0x2c]

if band==2
   b=a%0x0c
   c=table_58[b]
if band==0
   b=a%0x18
   c=table_24[b]
init_seed=c



if band==2 and type==3
    hops=0x2a
if band==0 and type==3
    hops=0x2d
```

all of 5.8ghz freqz tx hop list

1
['5732.0', '5734.0', '5736.0', '5738.0', '5741.0', '5743.0', '5745.0', '5748.0', '5750.0', '5752.0', '5755.0', '5757.0', '5759.0', '5761.0', '5764.0', '5766.0', '5768.0', '5771.0', '5773.0', '5775.0', '5778.0',
'5780.0', '5782.0', '5784.0', '5787.0', '5789.0', '5791.0', '5794.0', '5796.0', '5798.0', '5801.0', '5803.0', '5805.0', '5807.0', '5810.0', '5812.0', '5814.0', '5817.0', '5819.0', '5821.0', '5727.0', '5729.0']
5
['5750.0', '5761.0', '5773.0', '5784.0', '5796.0', '5807.0', '5819.0', '5734.0', '5745.0', '5757.0', '5768.0', '5780.0', '5791.0', '5803.0', '5814.0', '5729.0', '5741.0', '5752.0', '5764.0', '5775.0', '5787.0',
'5798.0', '5810.0', '5821.0', '5736.0', '5748.0', '5759.0', '5771.0', '5782.0', '5794.0', '5805.0', '5817.0', '5732.0', '5743.0', '5755.0', '5766.0', '5778.0', '5789.0', '5801.0', '5812.0', '5727.0', '5738.0']
11
['5778.0', '5803.0', '5732.0', '5757.0', '5782.0', '5807.0', '5736.0', '5761.0', '5787.0', '5812.0', '5741.0', '5766.0', '5791.0', '5817.0', '5745.0', '5771.0', '5796.0', '5821.0', '5750.0', '5775.0', '5801.0',
'5729.0', '5755.0', '5780.0', '5805.0', '5734.0', '5759.0', '5784.0', '5810.0', '5738.0', '5764.0', '5789.0', '5814.0', '5743.0', '5768.0', '5794.0', '5819.0', '5748.0', '5773.0', '5798.0', '5727.0', '5752.0']
13
['5787.0', '5817.0', '5750.0', '5780.0', '5810.0', '5743.0', '5773.0', '5803.0', '5736.0', '5766.0', '5796.0', '5729.0', '5759.0', '5789.0', '5819.0', '5752.0', '5782.0', '5812.0', '5745.0', '5775.0', '5805.0',
'5738.0', '5768.0', '5798.0', '5732.0', '5761.0', '5791.0', '5821.0', '5755.0', '5784.0', '5814.0', '5748.0', '5778.0', '5807.0', '5741.0', '5771.0', '5801.0', '5734.0', '5764.0', '5794.0', '5727.0', '5757.0']
17
['5805.0', '5748.0', '5787.0', '5729.0', '5768.0', '5807.0', '5750.0', '5789.0', '5732.0', '5771.0', '5810.0', '5752.0', '5791.0', '5734.0', '5773.0', '5812.0', '5755.0', '5794.0', '5736.0', '5775.0', '5814.0',
'5757.0', '5796.0', '5738.0', '5778.0', '5817.0', '5759.0', '5798.0', '5741.0', '5780.0', '5819.0', '5761.0', '5801.0', '5743.0', '5782.0', '5821.0', '5764.0', '5803.0', '5745.0', '5784.0', '5727.0', '5766.0']
19
['5814.0', '5761.0', '5805.0', '5752.0', '5796.0', '5743.0', '5787.0', '5734.0', '5778.0', '5821.0', '5768.0', '5812.0', '5759.0', '5803.0', '5750.0', '5794.0', '5741.0', '5784.0', '5732.0', '5775.0', '5819.0',
'5766.0', '5810.0', '5757.0', '5801.0', '5748.0', '5791.0', '5738.0', '5782.0', '5729.0', '5773.0', '5817.0', '5764.0', '5807.0', '5755.0', '5798.0', '5745.0', '5789.0', '5736.0', '5780.0', '5727.0', '5771.0']
23
['5736.0', '5789.0', '5745.0', '5798.0', '5755.0', '5807.0', '5764.0', '5817.0', '5773.0', '5729.0', '5782.0', '5738.0', '5791.0', '5748.0', '5801.0', '5757.0', '5810.0', '5766.0', '5819.0', '5775.0', '5732.0',
'5784.0', '5741.0', '5794.0', '5750.0', '5803.0', '5759.0', '5812.0', '5768.0', '5821.0', '5778.0', '5734.0', '5787.0', '5743.0', '5796.0', '5752.0', '5805.0', '5761.0', '5814.0', '5771.0', '5727.0', '5780.0']
25
['5745.0', '5803.0', '5764.0', '5821.0', '5782.0', '5743.0', '5801.0', '5761.0', '5819.0', '5780.0', '5741.0', '5798.0', '5759.0', '5817.0', '5778.0', '5738.0', '5796.0', '5757.0', '5814.0', '5775.0', '5736.0',
'5794.0', '5755.0', '5812.0', '5773.0', '5734.0', '5791.0', '5752.0', '5810.0', '5771.0', '5732.0', '5789.0', '5750.0', '5807.0', '5768.0', '5729.0', '5787.0', '5748.0', '5805.0', '5766.0', '5727.0', '5784.0']

Wireless communication system architecture

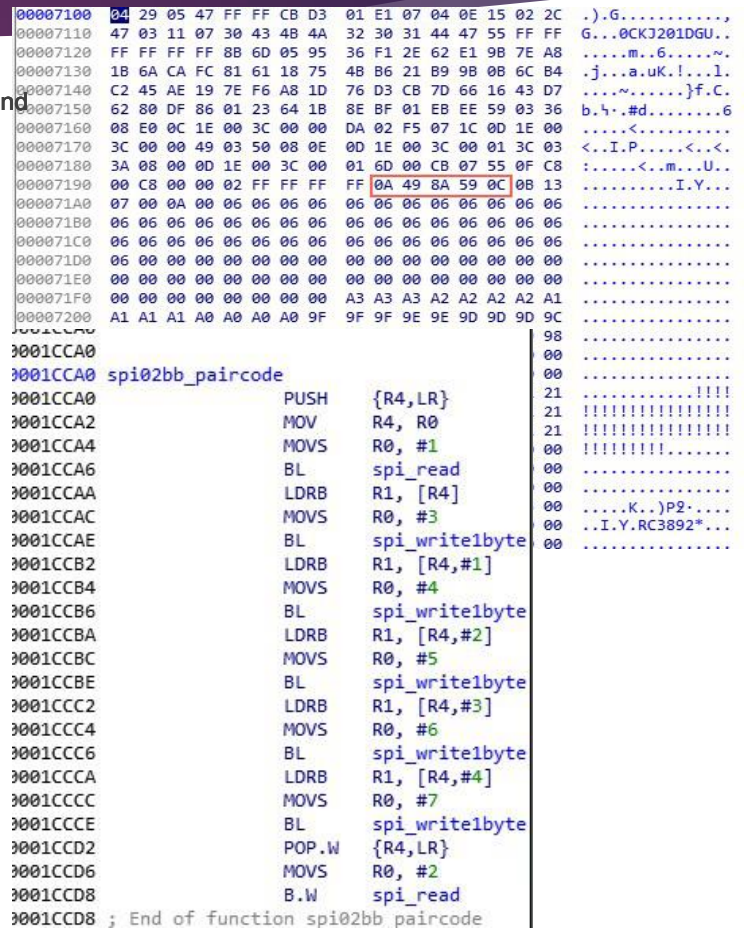
- RC Initialization -> NXP LPC1549
- RC initialization baseband of SPI register
- RC initialization Transceiver of SPI register
- RC Fixed frequency hopping radio
- Air Initialization -> STM32F103
- Air initialization baseband of SPI register
- Air initialization Transceiver of SPI Register
- Air Pick a high signal to noise ratio of the frequency monitor
- Air Roger that RC Synchronization signals and hopping sequence

- Write 5 Byte to the pairing code baseband

SPI address

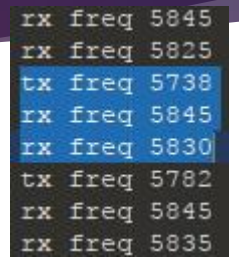
3 , 4 , 5 , 6 , 7

- Write 5 Byte passkey



Wireless communication system architecture

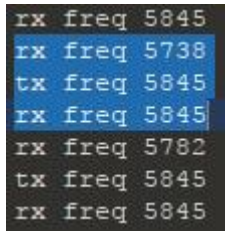
- RC versus Air Synchronization principle
- RC Hopping cycle is set

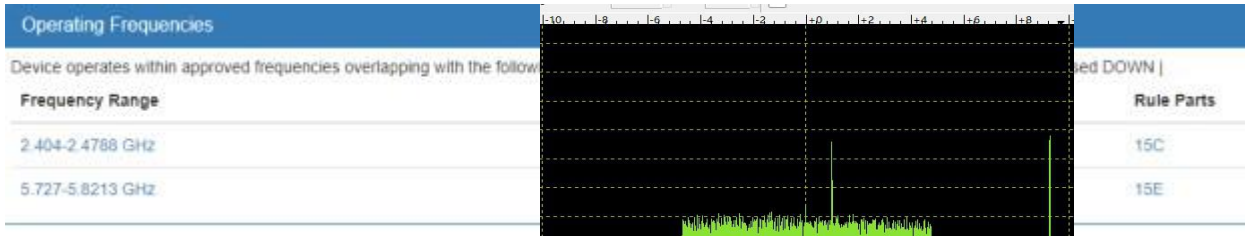


rx Hopping setting ----- 0.3ms --- tx Hopping setting ---- 3.075ms ----- rx Figure pass setting ---- 13.69ms --- rx Set the next frequency hopping

- Air Hopping cycle is set

rx Image transmission settings ----- 0.3ms --- rx Hopping setting ---- 2.91ms ---- tx Image transmission data transmission setting ---- 10.78ms --- rx Image transmission settings



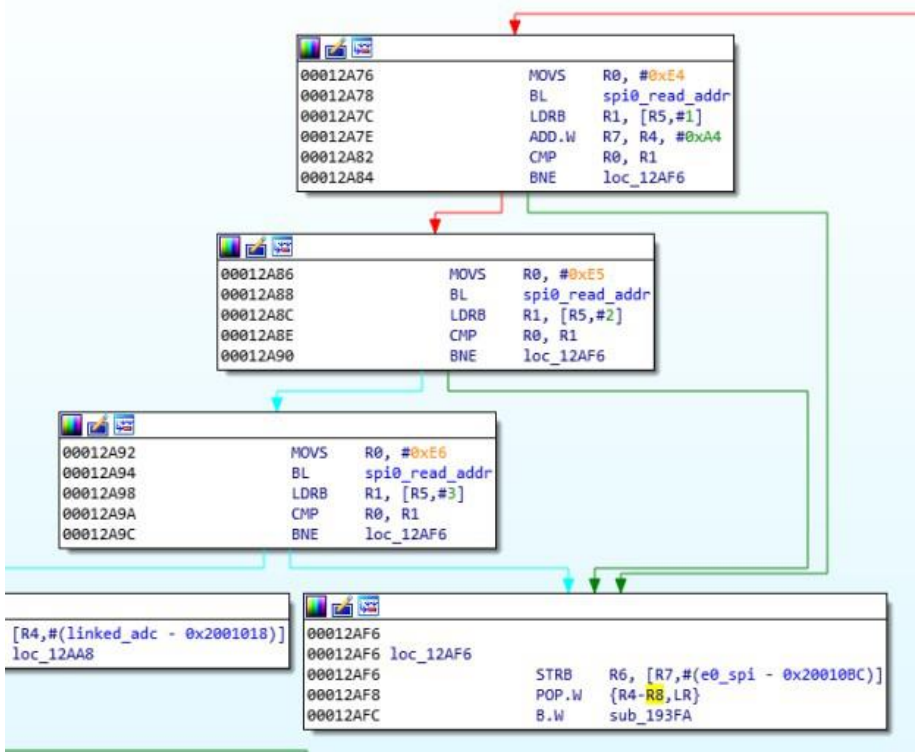


2.400 - 2.483 GHz和5.725 - 5.825 GHz

2.400 - 2.483 GHz（无干扰、无遮挡）
FCC：7000 m
CE：3500 m
SRRC：4000 m
5.725 - 5.825 GHz（无干扰、无遮挡）
FCC：7000 m
CE：2000 m
SRRC：5000 m

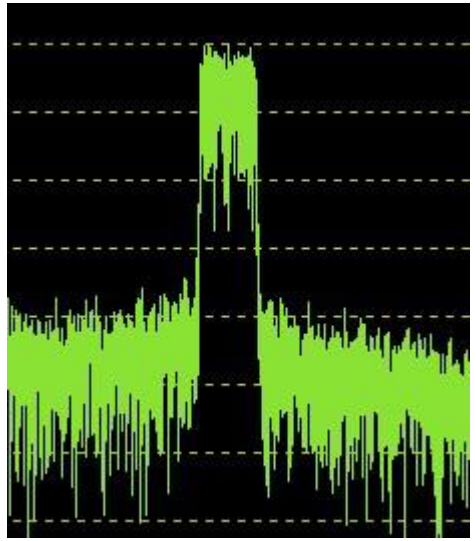
Wireless communication system architecture

- RC verification Air pair
- RC Read baseband SPI address(0xe4,0xe5,0xe6) Values are compared



How to hijack a UAV

- By pairing crack the code, the code word space 0x1000000
- Reform RC Firmware, fixed frequency intervals 10ms initialization baseband SPI Pairing code register
- At intervals of 10ms by SDR Fixed frequency reproduction Air signal
- Read baseband SPI register(0xe4,0xe5,0xe6) Values are compared
- The result is written to the serial port
- Crack need offline operation, not in real time
- Theoretically break up just a conservative estimate 46 hour
- Successful break can remotely modify the aircraft pairing code and take complete control of UAV



Todo list:

Successfully injected code into RC Firmware, crack the

algorithm by writing test code constructs RC Firmware

verification test

Anti-drone technology transfer confrontation

- Stm32 IO remap defeat SWD debugging
- Peripheral hardware detection, ADC Particle Size Determination
- Atmel chip Security Bit
- Switch protection circuit against SWD reset signal

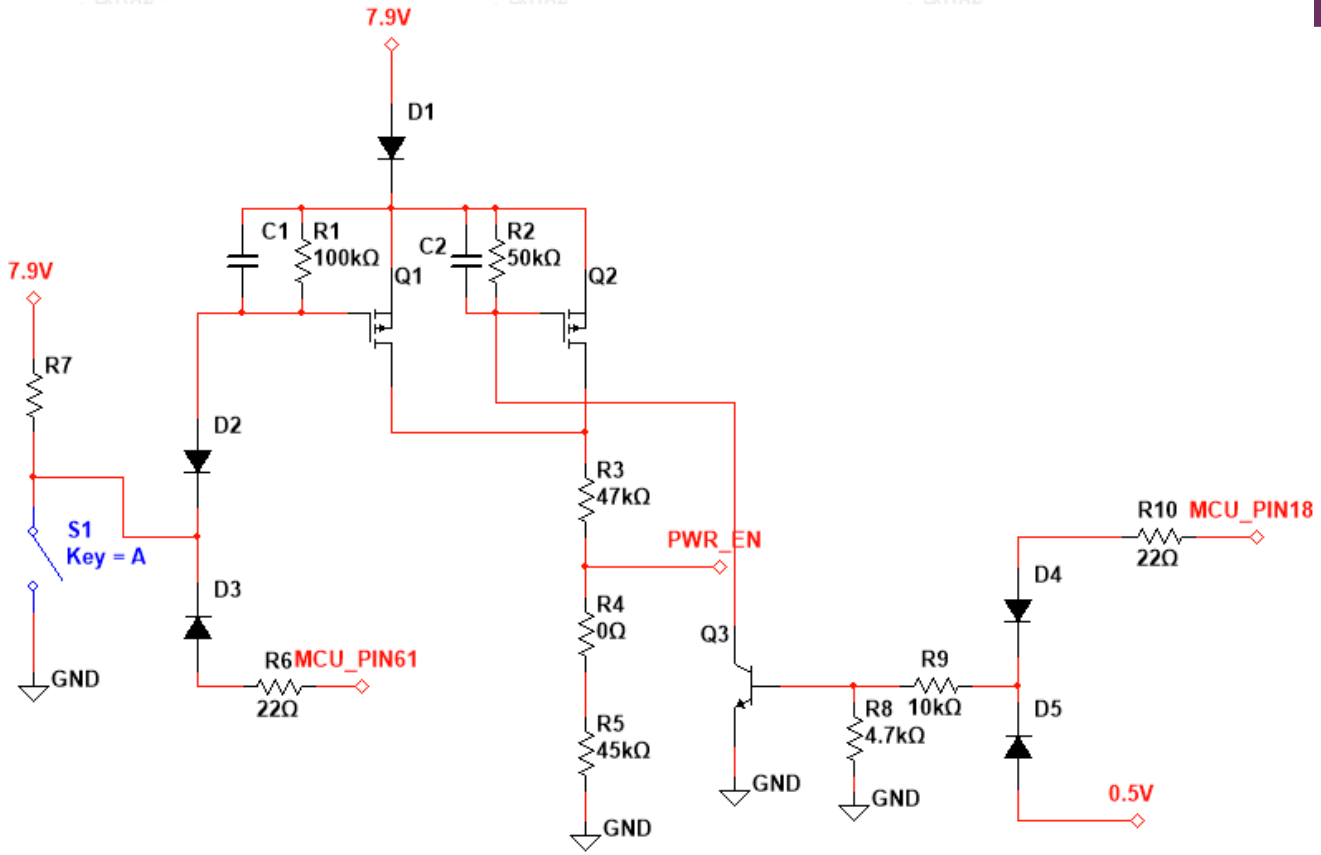
```
SUB             SP, SP, #0x20
BL              sub_800D7B8
MOVS            R1, #1
MOV             R0, R1
BL              RCC_APB2PeriphClockCmd
MOVS            R1, #1
LDR             R0, =0x300400 ; Full SWJ Disabled (JTAG-DP + SW-DP)
BL              GPIO_PinRemapConfig
ADD             R0, SP, #0x20+var_14
```

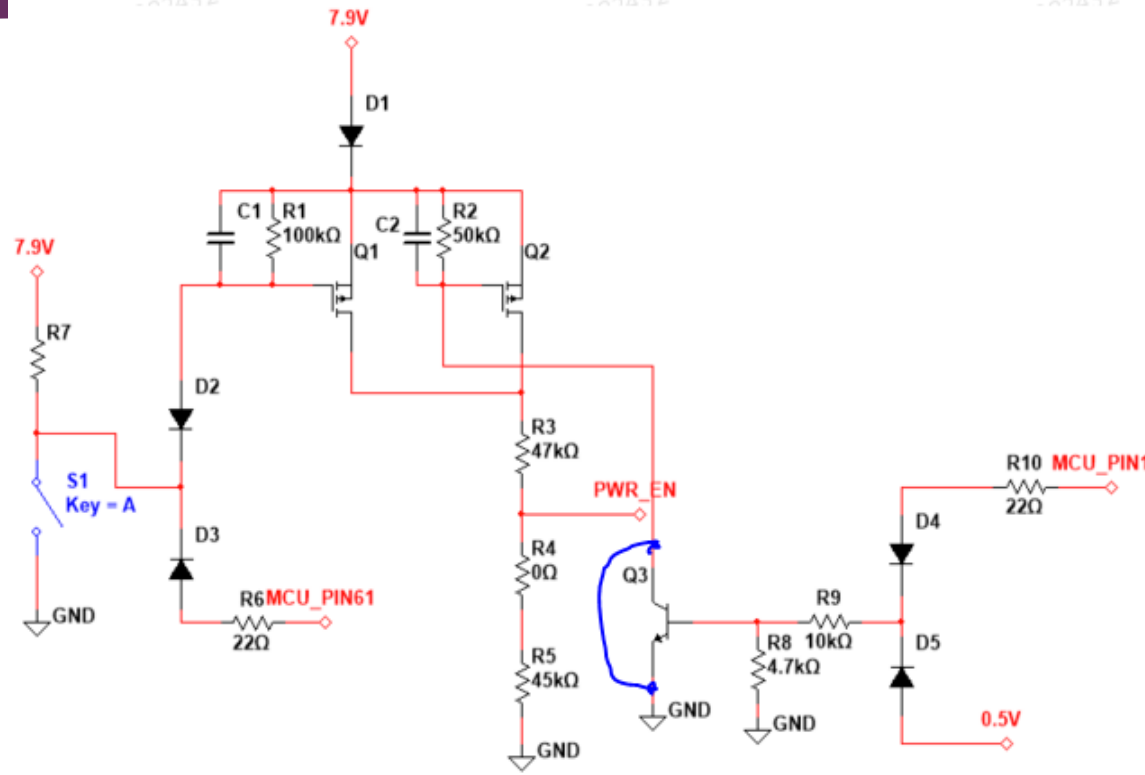
```
00BE14 set_jreset_io_disable                ; CODE XREF: sub_800D588+52↓p
00BE14
00BE14 var_8            = -8
00BE14 var_6            = -6
00BE14 var_5            = -5
```

```
PUSH            {R3,LR}
MOVS            R1, #1
MOVS            R0, #4
BL              RCC_APB2PeriphClockCmd
MOV.W           R0, #0x8000
STRH.W          R0, [SP,#8+var_8]
MOVS            R0, #3
STRB.W          R0, [SP,#8+var_6]
MOVS            R0, #4
STRB.W          R0, [SP,#8+var_5]
MOV             R1, SP
LDR             R0, =0x40010800
BL              GPIO_Init ; PA15 disable JTDI
POP             {R3,PC}
```

```
PUSH            {R3,LR}
MOVS            R1, #1
MOVS            R0, #8
BL              RCC_APB2PeriphClockCmd
MOVS            R0, #0x10
STRH.W          R0, [SP,#8+var_8]
MOVS            R1, #3
STRB.W          R1, [SP,#8+var_6]
STRB.W          R0, [SP,#8+var_5]
LDR             R0, =0x40010C0C
MOV             R1, SP
SUBS            R0, #0xC
BL              GPIO_Init ; PB4
```

Anti-drone technology transfer

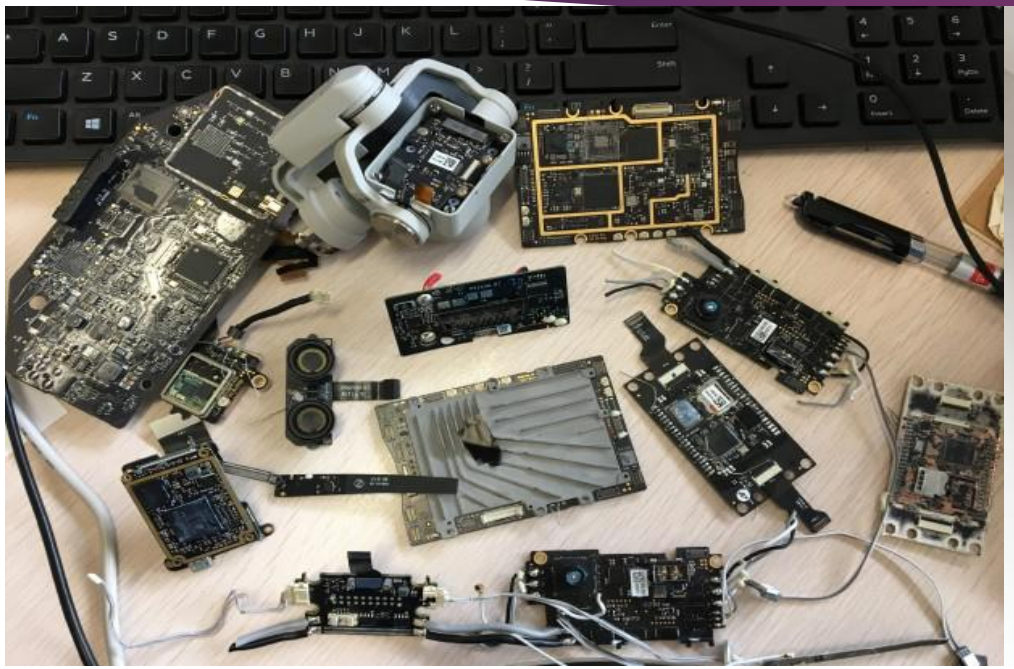




follow up research

- OTA Upgrade to crack agreement
- Constructed some hardware modules and firmware upgrade (Smart Battery, LightBridge)
- Fuzzing Flight control protocol interface
- SDR Analog wireless communication

Q & A



Thanks

vessialq@gmail.com