Professional MCDU box meets Open Source MOD

基調講演 @ Osaka NDS Embedded Cross Online Forum #15

Hisao Munakata (magu775<at>gmail.com)

Hobbyist (Not representing any organization)

2022-7-7

Disclaimer

I will introduce various SW/HW components in this material, however...

- I am not representing any party who develops these components.
- Things introduced here is just a quick hack, and cannot be the reference.
- Implementation highly depends on my environment, may not portable.
- Of course, no guarantee, no support commitment, play at your own risk
- Component (product) names are registered trade mark of each vendors

Microsoft Flight Simulator 2020 (a.k.a. MSFS or MSFS2020)

Product brief summary

- Runs on Windows PC and Xbox
- Highly utilize resources on cloud
- Revived by Asobo Studio
- Includes various planes includes flight systems
  - Garmin G1000, G3000, Boeing, Airbus,..
- Can use 3rd party HW (joy-stick, rudder,..)
- Also, accept 3rd party plane model (as MOD)

Now MSFS becomes entire flight simulator execution platform

3rd party gaming device (common products)

Standard Windows game device

- USB HID class device
- Windows OS support natively
- Test program integrated to Control Panel
- No extra driver needed
- MSFS detects and works
- Mostly right out of the box experience

A320 PRO-MX MCDU

Flightdeck Solutions E-Series A320 PRO-MX MCDU

Multi-Function Control Display Unit

- Emulate Airbus MCDU HW
- Identical size, material, color
- HDMI interface (video input)
- Ethernet port (event input/output)
- Speaks standard TCP/IP protocol
- Does not contain Flight Management and Guidance SW

https://flightdecksolutions.com/components/p/fds-a320-pro-mx-mcdu-e

Mostly for semi-professional use, MSFS support is not ready yet

A CRUCIAL COMPONENT

The Control Display Unit (CDU) is the gateway to the heart and soul of any modern airliner. With its key role as a human-machine interface, it plays a critical role in the operation of today's aircraft. Pilots interact with these devices from the time they board to the time they leave the cockpit.

With over a decade in production and thousands of units in circulation, FDS has long been the go-to CDU supplier for major airlines, airframers, aviation authorities, flight schools, R&D firms, and enthusiasts. The new E-Series PRO-MX MCDUs are designed to cater to each and every one of those diverse market segments.

Compatible with numerous major high level A320 avionics software suites and add-ons, and equipped with a built-in desk stand, FDS E-Series PRO-MX MCDUs can be easily used in a classroom setting or dropped into virtually any A320 trainer.

COMPATIBILITY

PROSIM

- A320

Jeehell

- TBD

FS2020

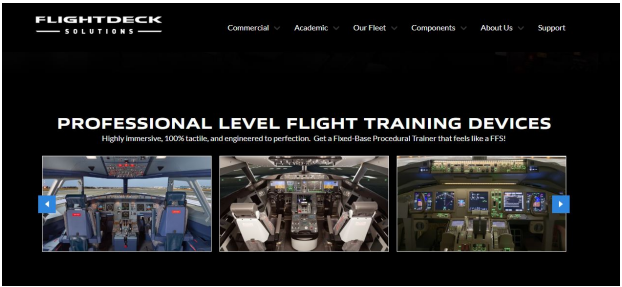
- Coming Soon!! (on our radar!)

X-PLANE 11 (unofficial, user-created plugins)

- ToLiss A319 / A321*
- Flight Factor A320 Ultimate*

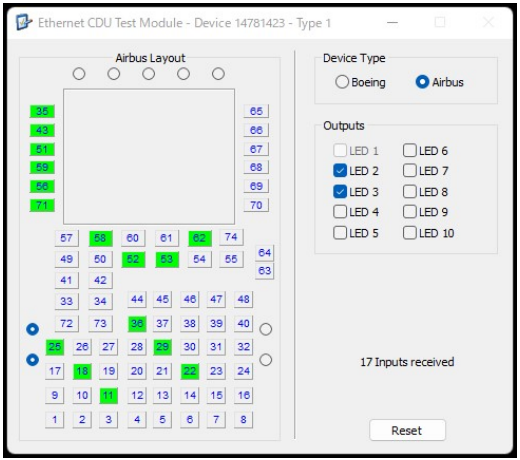
https://flightdecksolutions.com/components/p/fds-a320-pro-mx-mcdu-e

Flight-sim equipment provider for professional flight training use



https://flightdecksolutions.com/

It come with HW function test tool

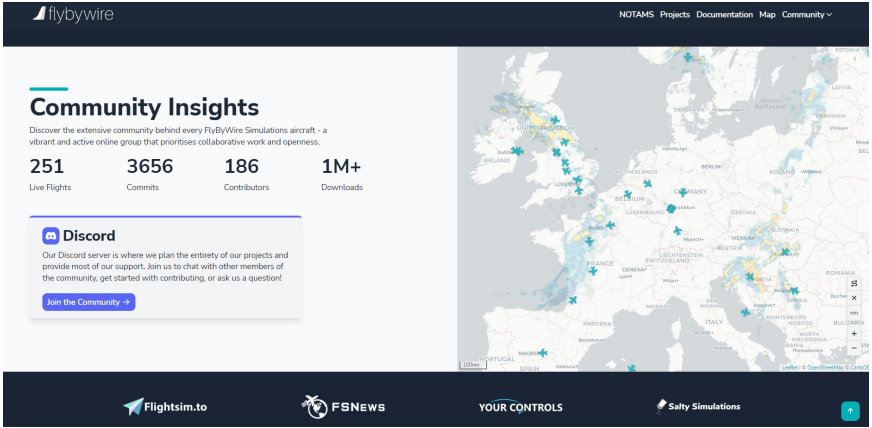


Tool named as interface IT module

- HW validation tool
- Developed by TEKWork Limited
- Generate TCP/IP packet
- Key input event
  - MCDU to PC (tool)
- Indicator on/off control
  - PC (tool) to MCDU

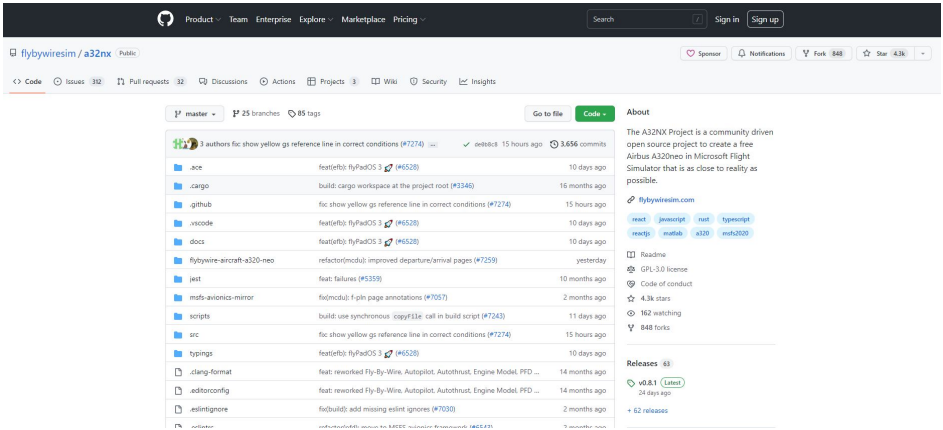
FryByWire Simulations A32NX project

Open Source Project who develops MOD for MSFS



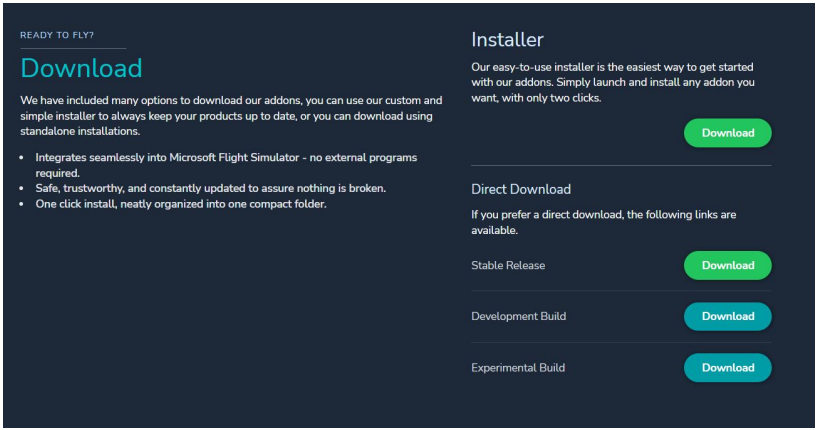
https://flybywiresim.com/

Fully comply with Open Source development model



https://github.com/flybywiresim/a32nx

Adopts rolling release model (Stable / Development / Experimental)



https://flybywiresim.com/a32nx/

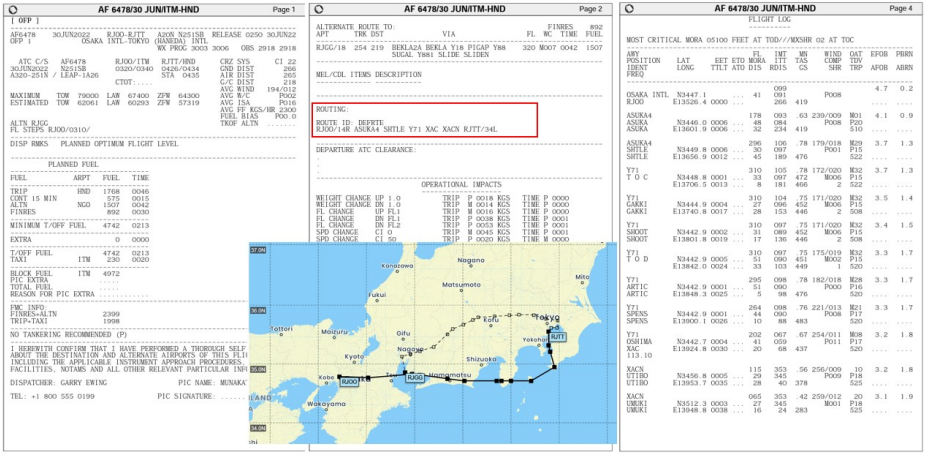
A32NX MOD aims to reproduce precisive reality

https://docs.flybywiresim.com/fbw-a32nx/

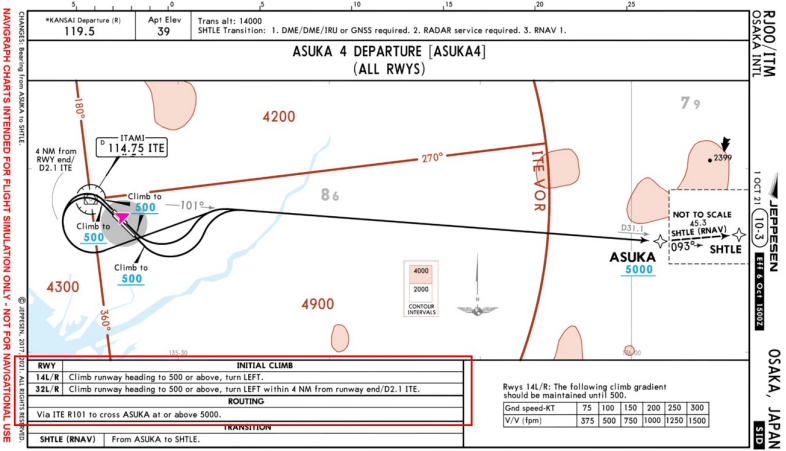
A32NX includes massive reality

- Look & Feel (color, sound,..)
- Physical motion model
- Flight management system
  - MCDU operations
  - FMS (Honeywell) algorithm
  - Flight plan using SimBrief
- Can integrate live data
  - Weather
  - Routing (via Navigraph)
- Follows every MSFS updates

A32NX can import Operational Flight Plan (OFP)



RJOO (Itami) SID (= Standard Instrument Departure) restrictions



A32NX MCDU reflect Flight Plan including restriction

MCDU is HMI for airline pilots, that includes...

- Flight Planning & Verify
  - Waypoints
  - Restriction
  - Backup plan
- Calculate & Set flight profiles
  - Speed / Altitude
  - Fuel consumption
- Manage during the flight
  - Route (waypoint /altitude) change
  - Destination AP change

Newly introduced MCDU web interface

MCDU runs as Web application

- Can run MCDU on separate tablet, smartphone,..
- Can use real printer
- Runs MCDU-server to interface to MSFS/A32NX MOD
- Use WebSocket protocol
- https://docs.flybywiresim.com/fbw-a32nx/feature-guides/web-mcdu/

MCDU Websocket interface (Advanced use case)

Websocket Port

The websocket port is the port where the MCDU Web Interface will communicate with the actual MCDU. It sends and receives data through this connection.

This will allow to have different UI implementations or even hardware MCDUs in the future. We plan to eventually also create separate documentation for using this websocket connection directly.

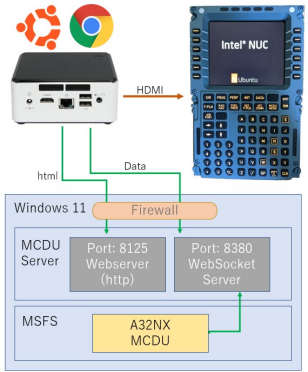
> ⚠ We recommend to **not** change the default port for the websocket if not required. If you have to change this port because it is already occupied on your machine you need to change it as well in the EFB Sim Options Settings page.

https://docs.flybywiresim.com/fbw-a32nx/feature-guides/web-mcdu/#websocket-port

Ubuntu NUC box runs web-browser to feed MCDU display

Ubuntu 22.04 runs Chrome arcade mode

- Use Intel NUC as a stand-alone browser
- Auto boot Ubuntu 22.04 Desktop
- Auto login
- Auto start Chrome
- Use ”Arcade Mode” (for digital signage)
  - contents only (no pain/tab) display
  - Auto connect to MCDU Web server
- Auto shutdown by press power SW

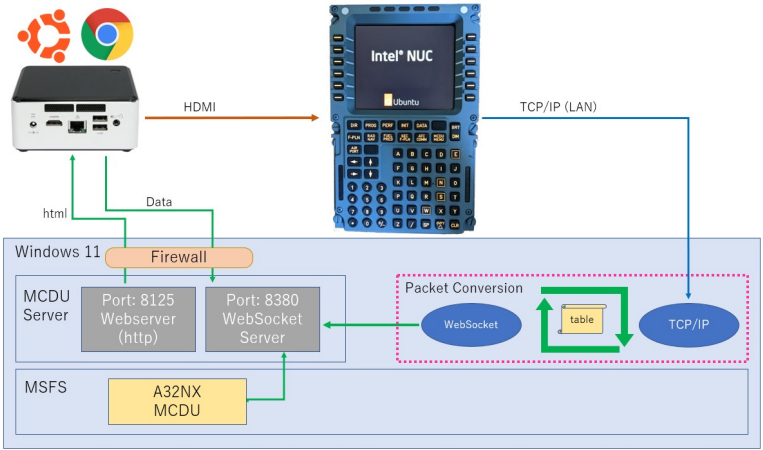


Automatically MCDU screen appears, when MSFS and MCDU Web are up

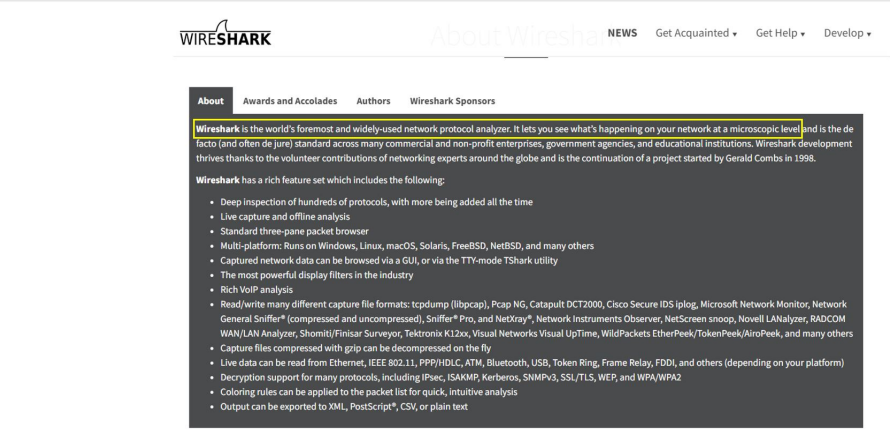
Require Windows firewall setting (depends on installation)

How to integrate HW MCDU box to MSFS

System Architecture (w/MCDU web interface)

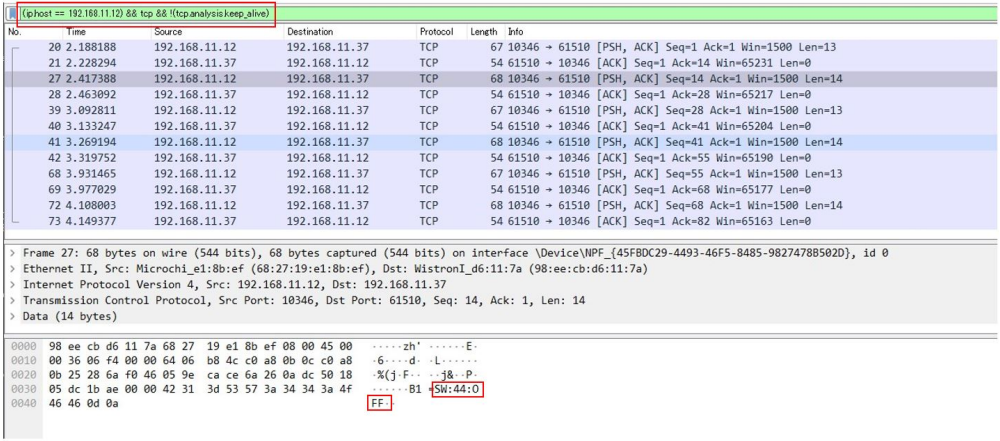


Let's hack how it works ! (using WireShark)

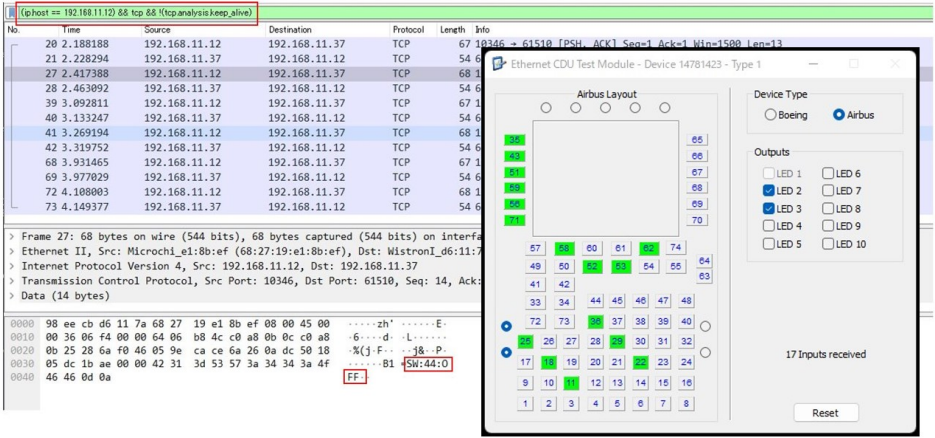


https://www.wireshark.org/

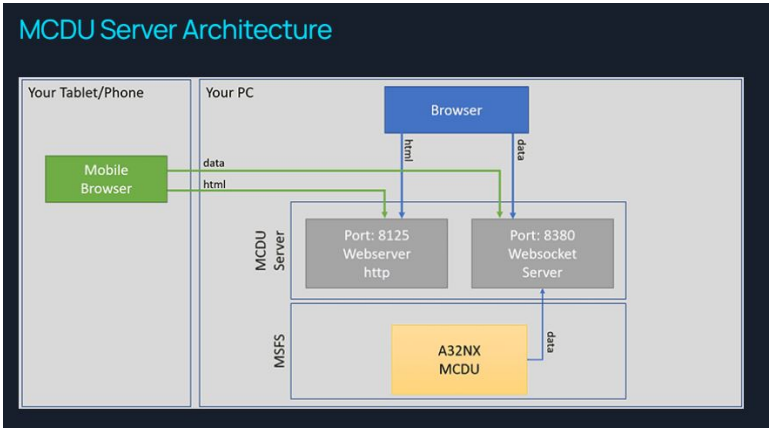
Capture TCP/IP traffic (using Wireshark)



Capture TCP/IP traffic (using Wireshark)

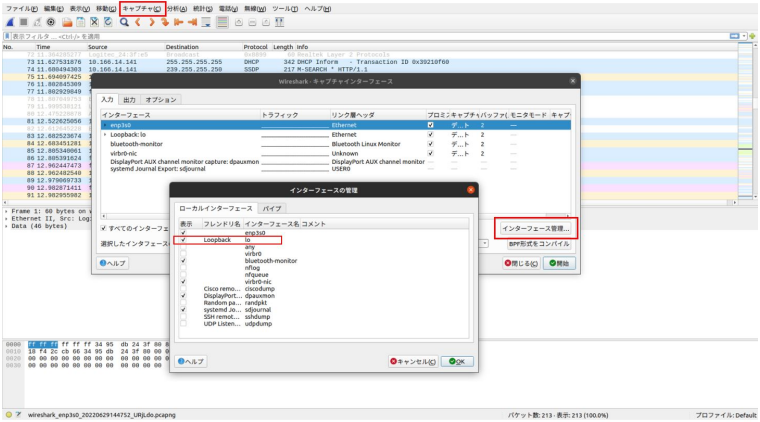


MCDU server architecture (Web socket interface)



https://docs.flybywiresim.com/fbw-a32nx/feature-guides/web-mcdu/#starting-the-mcdu-serve

Wireshark Adapter selection (to capture local socket)



https://www.wireshark.org/docs/wsug_html_chunked/ChManageInterfacesSection.html

Capture the web socket data (using Wireshark)

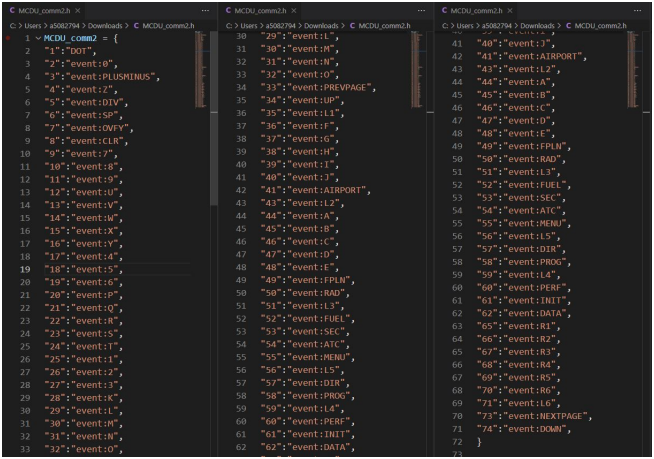
tcp.stream eq 0

No.	Time	Source	Destination	Protocol	Length	Info
74	1.798965	192.168.11.41	192.168.11.37	TCP	78	[TCP Dup ACK 73#1] 54239 → 8380 [ACK] Seq=14 Ack=1675 Win=4043 Len=0 TSval=175558256 TSecr=42423677 SLE=227 SRE=1675
90	1.989937	192.168.11.41	192.168.11.37	TCP	79	54239 → 8380 [PSH, ACK] Seq=14 Ack=1675 Win=4096 Len=13 TSval=175558456 TSecr=42423677
91	1.990191	192.168.11.37	192.168.11.41	TCP	75	8380 → 54239 [PSH, ACK] Seq=1675 Ack=27 Win=8191 Len=9 TSval=42423909 TSecr=175558456
92	1.994049	192.168.11.41	192.168.11.37	TCP	66	54239 → 8380 [ACK] Seq=27 Ack=1684 Win=4095 Len=0 TSval=175558460 TSecr=42423909
129	2.266320	192.168.11.37	192.168.11.41	TCP	1733	8380 → 54239 [PSH, ACK] Seq=1684 Ack=27 Win=8191 Len=1667 TSval=42424185 TSecr=175558460
131	2.297945	192.168.11.37	192.168.11.41	TCP	1514	[TCP Retransmission] 8380 → 54239 [PSH, ACK] Seq=1903 Ack=27 Win=8191 Len=1448 TSval=42424217 TSecr=175558460
132	2.301967	192.168.11.41	192.168.11.37	TCP	66	54239 → 8380 [ACK] Seq=27 Ack=3351 Win=4043 Len=0 TSval=175558768 TSecr=42424185
133	2.301967	192.168.11.41	192.168.11.37	TCP	78	[TCP Dup ACK 132#1] 54239 → 8380 [ACK] Seq=27 Ack=3351 Win=4043 Len=0 TSval=175558768 TSecr=42424217 SLE=1903 SRE=3351
162	2.637012	192.168.11.41	192.168.11.37	TCP	79	54239 → 8380 [PSH, ACK] Seq=27 Ack=3351 Win=4096 Len=13 TSval=175559097 TSecr=42424217
163	2.637294	192.168.11.37	192.168.11.41	TCP	75	8380 → 54239 [PSH, ACK] Seq=3351 Ack=40 Win=8191 Len=9 TSval=42424556 TSecr=175559097
164	2.638894	192.168.11.41	192.168.11.37	TCP	66	54239 → 8380 [ACK] Seq=40 Ack=3360 Win=4095 Len=0 TSval=175559105 TSecr=42424556
165	2.889284	192.168.11.37	192.168.11.41	TCP	1735	8380 → 54239 [PSH, ACK] Seq=3360 Ack=40 Win=8191 Len=1669 TSval=42424808 TSecr=175559105
166	2.917471	192.168.11.41	192.168.11.37	TCP	66	54239 → 8380 [ACK] Seq=40 Ack=5029 Win=4043 Len=0 TSval=175559383 TSecr=42424808
177	3.154683	192.168.11.41	192.168.11.37	TCP	60	[TCP Keep-Alive] 54239 → 8380 [ACK] Seq=39 Ack=5029 Win=4096 Len=0
178	3.154716	192.168.11.37	192.168.11.41	TCP	66	[TCP Keep-Alive ACK] 8380 → 54239 [ACK] Seq=5029 Ack=40 Win=8191 Len=0 TSval=42425074 TSecr=175559383
195	3.571729	192.168.11.41	192.168.11.37	TCP	79	54239 → 8380 [PSH, ACK] Seq=40 Ack=5029 Win=4096 Len=13 TSval=175560032 TSecr=42425074
196	3.572220	192.168.11.37	192.168.11.41	TCP	75	8380 → 54239 [PSH, ACK] Seq=5029 Ack=53 Win=8191 Len=9 TSval=42425491 TSecr=175560032
197	3.573972	192.168.11.41	192.168.11.37	TCP	66	54239 → 8380 [ACK] Seq=53 Ack=5038 Win=4095 Len=0 TSval=175560040 TSecr=42425491

```
> Frame 163: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface \Device\NPF_{45F8DC29-4493-46F5-8485-98274788502D}, id 0
> Ethernet II, Src: WistronI_d6:11:7a (98:ee:cb:d6:11:7a), Dst: 1e:18:c4:b7:28:8e (1e:18:c4:b7:28:8e)
> Internet Protocol Version 4, Src: 192.168.11.37, Dst: 192.168.11.41
> Transmission Control Protocol, Src Port: 8380, Dst Port: 54239, Seq: 3351, Ack: 40, Len: 9
v Data (9 bytes)
    Data: 81076576656e743a33
    [Length: 9]
```

```
0000  1e 18 c4 b7 28 8e 98 ee  cb d6 11 7a 08 00 45 00   ····(··· ···z··E·
0010  00 3d e5 ee 40 00 80 06  00 00 c0 a8 0b 25 c0 a8   ·=··@··· ·····%··
0020  0b 29 20 bc d3 df a7 e8  62 8c ab 48 61 00 80 18   ·) ····· b··Ha···
0030  1f ff 97 ce 00 00 01 01  08 0a 02 87 58 ec 0a 76   ········ ····X··v
0040  d1 b9 81 07 65 76 65 6e  74 3a 33                  ····even t:3
```

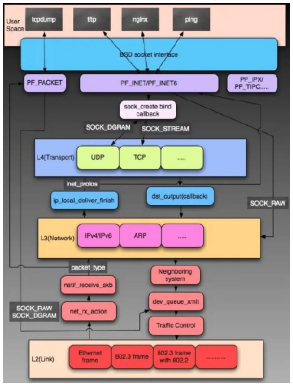
Command translation table (TCP <-> WebSocket)



Network programing (requires OS kernel functions)

network connection is a kernel matters

- kernel support L1 to L5 network features
- Layer5 : Socket interface
- Layer4 : TCP/UDP (Error correct)
- Layer3 : IPv4/IPv6, ARP (Routing)
- Layer2 : Ethernet frame (Data link)
- Layer1 : NIC driver (Card R/W)

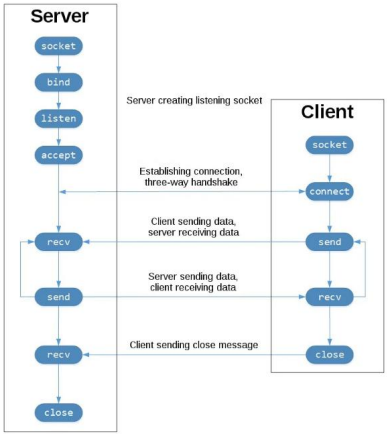


You can easily break the entire system with the bad network code

Network programing (Use Python TCP/IP support)

Python socket library

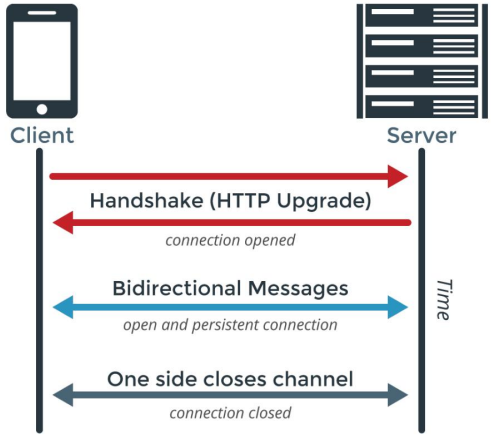
- To send/receive TCP packet to MCDU HW
- 1: create socket
- 2: bind (address, port)
- 3S:listen (remote connection)
- 3C:connect (to server)
- 4: send/recv
- 5: close



Network programing (Also use Python WebSocket support)

Python websockets library

- To communicate with MCDU web-interface
- 1: Establish http connection
- 2: Upgrade to WebSocket
- 3: Keep TCP session
- 4: Bi-directional send/recv
- 5: Session close (at request)



Trial python code (https://github.com/hmunak/MCDU_conn)

```python
class BaseClient:
    def __init__(self, timeout:int=10, buffer:int=1024):
        self.__socket = None
        self.__websocekt = None
        self.__address = None
        self.__timeout = timeout
        self.__buffer = buffer

    def connect(self, address, family:int, typ:int, proto:int):
        self.__address = address
        self.__socket = socket.socket(family, typ, proto)
        self.__websocekt = create_connection("ws://192.168.11.37:8380/")
        self.__socket.settimeout(self.__timeout)
        self.__socket.connect(self.__address)

    def send(self, message:str="") -> None:
        flag = False
        while True:
            try:
                message_recv = self.__socket.recv(self.__buffer).decode('utf-8')
                self.received(message_recv)
                if flag:
                    break
            except KeyboardInterrupt:
                print(" Caught CTRL+C")
                self.__socket.shutdown(socket.SHUT_RDWR)
                self.__socket.close()
                sys.exit(0)
        try:
            self.__socket.shutdown(socket.SHUT_RDWR)
            self.__socket.close()
        except:
            pass
```

```python

    def received(self, message:str):
        #print(message)
        if(message.split(':')[-1].strip()) == "ON":
            key_id = message.rsplit(':')[-2]
            if( key_id in MCDU_comm2):
                key_server = MCDU_comm2[key_id]
                print(key_server)
                self.__websocekt.send(str(key_server))

class InetClient(BaseClient):
    def __init__(self, host:str="192.168.11.12", port:int=10346) -> None:
        self.server=(host,port)
        super().__init__(timeout=6000, buffer=1024)
        super().connect(self.server, socket.AF_INET, socket.SOCK_STREAM, 0)


def signal_handler(signal, frame):
        # close the socket here
        self.__socket.shutdown(socket.SHUT_RDWR)
        self.__socket.close()
        sys.exit(0)
        signal.signal(signal.SIGINT, signal_handler)


if __name__=="__main__":
    cli = InetClient()
    cli.send()

```

Demo

Next step (idea)

FryByWire OSS project

- Share HW MCDU integration status
- Share trial code
- Discuss how to integrate to MCDU server (as a plugin ?)
- Announce to the community (if accepted)

FlightDeck Solutions

- Share MSFS/FrybyWire integration status
- Point OSS discussion space
- Allow them (if they wish) to publish this state
- Discuss possible next step collaborations

Potentially, HW MCDU integration might attract FryByWire Sim users