

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-mcd-u-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

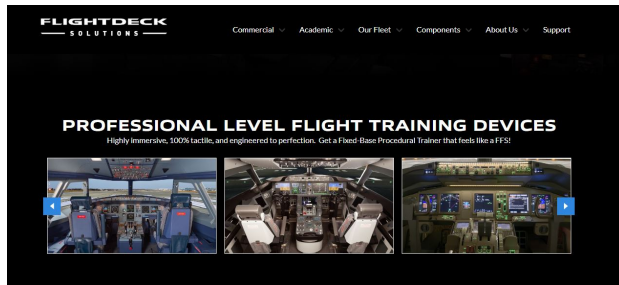
- Coming Soon!! (on our radar!)

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

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

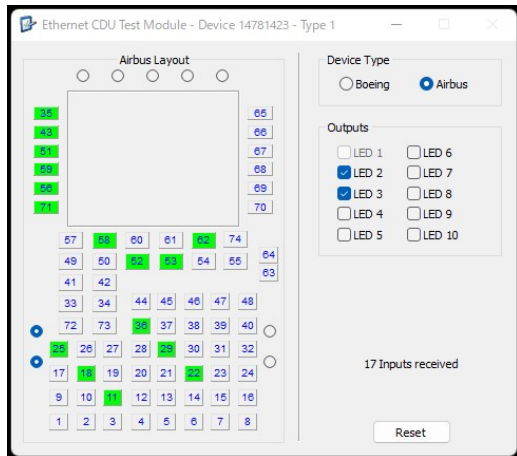
<https://flightdecksolutions.com/components/p/fds-a320-pro-mx-mcd-u-e>

Flight-sim equipment provider for professional flight training use

**Honeywell****BAE SYSTEMS**

<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

The screenshot shows the FlyByWire website. The header includes the FlyByWire logo and navigation links: NOTAMS, Projects, Documentation, Map, and Community. The main content area is titled 'Community Insights' and features four statistics: 251 Live Flights, 3656 Commits, 186 Contributors, and 1M+ Downloads. Below this is a Discord link with a description of the server and a 'Join the Community' button. To the right is a map of Europe with various locations marked. The footer contains logos for Flightsim.to, FSNEWS, YOUR CONTROLS, and Salty Simulations, along with a small green button with an upward arrow.

flybywire

NOTAMS Projects Documentation Map Community

Community Insights

Discover the extensive community behind every FlyByWire Simulations aircraft - a vibrant and active online group that prioritises collaborative work and openness.

251	3656	186	1M+
Live Flights	Commits	Contributors	Downloads

Discord

Our Discord server is where we plan the entirety of our projects and provide most of our support. Join us to chat with other members of the community, get started with contributing, or ask us a question!

[Join the Community →](#)

Flightsim.to FSNEWS YOUR CONTROLS Salty Simulations

<https://flybywiresim.com/>

Fully comply with Open Source development model

flybywiresim / a32nx Public

Product Team Enterprise Explore Marketplace Pricing

Search Sign in Sign up

Code Issues 312 Pull requests 32 Discussions Actions Projects 3 Wiki Security Insights

master 25 branches 85 tags

Go to file Code About

3 authors fix show yellow gs reference line in correct conditions (#7274) ✓ dets8c 15 hours ago 3,656 commits

.ace	feat(efb): flyPadOS 3 (#6528)	10 days ago
.cargo	build: cargo workspace at the project root (#3346)	16 months ago
.github	fix: show yellow gs reference line in correct conditions (#7274)	15 hours ago
.vscode	feat(efb): flyPadOS 3 (#6528)	10 days ago
docs	feat(efb): flyPadOS 3 (#6528)	10 days ago
flybywire-sircraft-a320-neo	refactor(mcd): improved departure/arrival pages (#7259)	yesterday
jest	feat: failures (#5359)	10 months ago
msfs-avionics-mirror	fix(mcd): f-pin page annotations (#7057)	2 months ago
scripts	build: use synchronous copyFile call in build script (#7243)	11 days ago
src	fix: show yellow gs reference line in correct conditions (#7274)	15 hours ago
typings	feat(efb): flyPadOS 3 (#6528)	10 days ago
.clang-format	feat: reworked Fly-By-Wire, Autopilot, Autothrust, Engine Model, PFD ...	14 months ago
.editorconfig	feat: reworked Fly-By-Wire, Autopilot, Autothrust, Engine Model, PFD ...	14 months ago
.eslintignore	fix(build): add missing eslint ignores (#7030)	2 months ago
.gitignore	infotool: add support to RCEC simulator framework (RCECF3)	3 months ago

The A32NX Project is a community driven open source project to create a free Airbus A320neo in Microsoft Flight Simulator that is as close to reality as possible.

flybywiresim.com

react javascript rust typescript reactjs matlab a320 msfs2020

Readme
GPL-3.0 license
Code of conduct
4.3k stars
162 watching
848 forks

Releases 63

v0.8.1 (Latest)
24 days ago

+ 62 releases

<https://github.com/flybywiresim/a32nx>

Adopts rolling release model (Stable / Development / Experimental)

READY TO FLY?

Download

We have included many options to download our addons, you can use our custom and simple installer to always keep your products up to date, or you can download using standalone installations.

- Integrates seamlessly into Microsoft Flight Simulator - no external programs required.
- Safe, trustworthy, and constantly updated to assure nothing is broken.
- One click install, neatly organized into one compact folder.

Installer

Our easy-to-use installer is the easiest way to get started with our addons. Simply launch and install any addon you want, with only two clicks.

[Download](#)

Direct Download

If you prefer a direct download, the following links are available.

Stable Release	Download
Development Build	Download
Experimental Build	Download

<https://flybywiresim.com/a32nx/>

A32NX MOD aims to reproduce precise reality

FlyByWire A32NX

- Overview
- Common Questions
- Versions and Features
- Installation Guide
- Recommended Settings
- Liveries Guide
- Support
- Feature Guides
- API and Hardware

FlyByWire A32NX Overview

This section of the FlyByWire Documentation is dedicated to the A32NX add-on itself. It covers the software and more technical aspects of the FlyByWire add-on.

The A32NX Project is a community-driven open source project to create a free Airbus A320neo in Microsoft Flight Simulator that is as close to reality as possible. It started out as an enhancement project to the default A320neo and is now proceeding as an independent add-on project aiming to bring the FlyByWire A32NX up to payware-level systems depth and functionality, all for free.

Aircraft Configuration(s)

The following aircraft configurations are currently simulated:

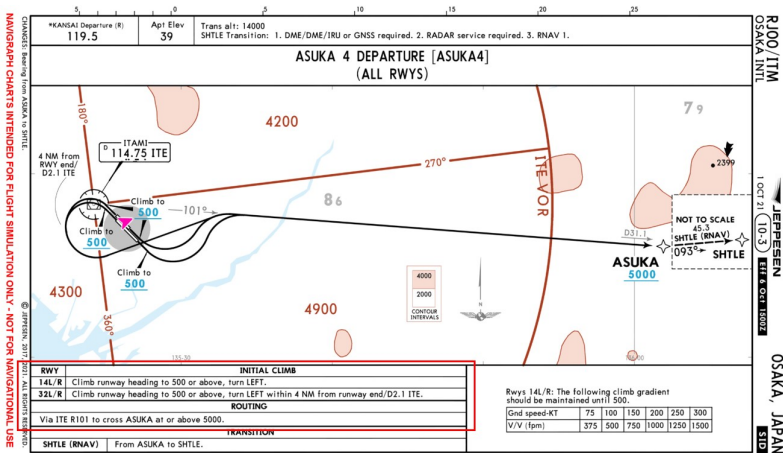
Simulated Hardware	
Model	A320-251N
Engine	CFM LEAP 1A-26
APU	APS3200
FMS	Honeywell Release H3
FWC Std.	H2F9C
TAWS	Honeywell EGPWS
ACAS	Honeywell TPA-100B
ATC	Honeywell TRA-100B
MMR	Honeywell 1MMR
WXR	Honeywell RDR-4000

<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

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.



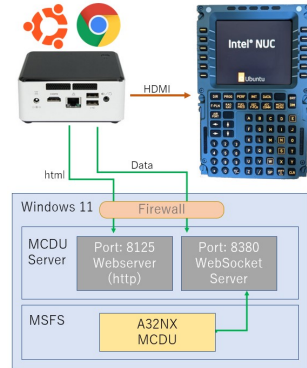
We recommend to **not** change the default port for the websocket if not required. If you have to change this port because it is already occupied on your machine you need to change it as well in the EFB Sim Options Settings page.

<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



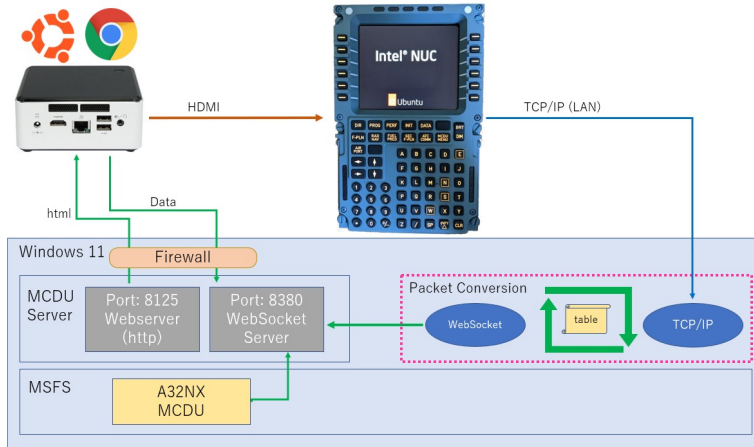
Automatically MCDU screen appears, when MSFS and MCDU Web are up

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)



About Wireshark

NEWS

Get Acquainted ▾

Get Help ▾

Develop ▾

About

Awards and Accolades

Authors

Wireshark Sponsors

Wireshark is the world's foremost and widely-used network protocol analyzer. It lets you see what's happening on your network at a microscopic level and is the de facto (and often de jure) standard across many commercial and non-profit enterprises, government agencies, and educational institutions. Wireshark development thrives thanks to the volunteer contributions of networking experts around the globe and is the continuation of a project started by Gerald Combs in 1998.

Wireshark has a rich feature set which includes the following:

- Deep inspection of hundreds of protocols, with more being added all the time
- Live capture and offline analysis
- Standard three-pane packet browser
- Multi-platform: Runs on Windows, Linux, macOS, Solaris, FreeBSD, NetBSD, and many others
- Captured network data can be browsed via a GUI, or via the TTY-mode TShark utility
- The most powerful display filters in the industry
- Rich VoIP analysis
- Read/write many different capture file formats: tcpdump (libpcap), Pcap NG, Catapult DCT2000, Cisco Secure IDS iplog, Microsoft Network Monitor, Network General Sniffer* (compressed and uncompressed), Sniffer* Pro, and NetXray*, Network Instruments Observer, NetScreen snoop, Novell LANalyzer, RADCOM WAN/LAN Analyzer, Shomiti/Finisar Surveyor, Tektronix K12xx, Visual Networks Visual UpTime, WildPackets EtherPeek/TokenPeek/AiroPeek, and many others
- Capture files compressed with gzip can be decompressed on the fly
- Live data can be read from Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI, and others (depending on your platform)
- Decryption support for many protocols, including IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, and WPA/WPA2
- Coloring rules can be applied to the packet list for quick, intuitive analysis
- Output can be exported to XML, PostScript*, CSV, or plain text

<https://www.wireshark.org/>

Capture TCP/IP traffic (using Wireshark)

(((host == 192.168.11.12) && tcp && !(tcp.analysis.keep_alive)))

No.	Time	Source	Destination	Protocol	Length	Info
20	2.188188	192.168.11.12	192.168.11.37	TCP	67	10346 → 61510 [PSH, ACK] Seq=1 Ack=1 Win=1500 Len=13
21	2.228294	192.168.11.37	192.168.11.12	TCP	54	61510 → 10346 [ACK] Seq=1 Ack=14 Win=65231 Len=0
27	2.417388	192.168.11.12	192.168.11.37	TCP	68	10346 → 61510 [PSH, ACK] Seq=14 Ack=1 Win=1500 Len=14
28	2.463092	192.168.11.37	192.168.11.12	TCP	54	61510 → 10346 [ACK] Seq=1 Ack=28 Win=65217 Len=0
39	3.092811	192.168.11.12	192.168.11.37	TCP	67	10346 → 61510 [PSH, ACK] Seq=28 Ack=1 Win=1500 Len=13
40	3.133247	192.168.11.37	192.168.11.12	TCP	54	61510 → 10346 [ACK] Seq=1 Ack=41 Win=65204 Len=0
41	3.269194	192.168.11.12	192.168.11.37	TCP	68	10346 → 61510 [PSH, ACK] Seq=41 Ack=1 Win=1500 Len=14
42	3.319752	192.168.11.37	192.168.11.12	TCP	54	61510 → 10346 [ACK] Seq=1 Ack=55 Win=65190 Len=0
68	3.931465	192.168.11.12	192.168.11.37	TCP	67	10346 → 61510 [PSH, ACK] Seq=55 Ack=1 Win=1500 Len=13
69	3.977029	192.168.11.37	192.168.11.12	TCP	54	61510 → 10346 [ACK] Seq=1 Ack=68 Win=65177 Len=0
72	4.108003	192.168.11.12	192.168.11.37	TCP	68	10346 → 61510 [PSH, ACK] Seq=68 Ack=1 Win=1500 Len=14
73	4.149377	192.168.11.37	192.168.11.12	TCP	54	61510 → 10346 [ACK] Seq=1 Ack=82 Win=65163 Len=0

> Frame 27: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface \Device\NPF_{45FBDC29-4493-46F5-8485-9827478B502D}, id 0
 > Ethernet II, Src: Microchi_e1:8b:ef (68:27:19:e1:8b:ef), Dst: WistronI_d6:11:7a (98:ee:cb:d6:11:7a)
 > Internet Protocol Version 4, Src: 192.168.11.12, Dst: 192.168.11.37
 > Transmission Control Protocol, Src Port: 10346, Dst Port: 61510, Seq: 14, Ack: 1, Len: 14
 > Data (14 bytes)

```

0000  98 ee cb d6 11 7a 68 27 19 e1 8b ef 08 00 45 00  ....zh' ....E.
0010  00 36 06 f4 00 00 64 06 b8 4c c0 a8 0b 0c c0 a8  -6....d. .L.....
0020  0b 25 28 6a f0 46 05 9e ca ce 6a 26 0a dc 50 18  -(j.-F.. ..j&..P-
0030  05 dc 1b ae 00 00 42 31 3d 53 57 3a 34 34 3a 4f  ....B1 -SW:44:0
0040  46 46 0d 0a                                     FF-
  
```

Capture TCP/IP traffic (using Wireshark)

Wireshark packet capture showing TCP traffic from 192.168.11.12 to 192.168.11.37. The filter is `(iphost == 192.168.11.12) && tcp && !tcp.analysis.keep_alive`.

No.	Time	Source	Destination	Protocol	Length	Info
20	2.188188	192.168.11.12	192.168.11.37	TCP	67	10346 → 61510 [PSH, ACK] Seq=1 Ack=1 Win=1500 Len=13
21	2.228294	192.168.11.37	192.168.11.12	TCP	54	61510 → 10346 [ACK] Seq=1 Ack=1 Win=1500 Len=0
27	2.417388	192.168.11.12	192.168.11.37	TCP	68	10346 → 61510 [PSH, ACK] Seq=1 Ack=1 Win=1500 Len=13
28	2.463092	192.168.11.37	192.168.11.12	TCP	54	61510 → 10346 [ACK] Seq=1 Ack=1 Win=1500 Len=0
39	3.092811	192.168.11.12	192.168.11.37	TCP	67	10346 → 61510 [PSH, ACK] Seq=1 Ack=1 Win=1500 Len=13
40	3.133247	192.168.11.37	192.168.11.12	TCP	54	61510 → 10346 [ACK] Seq=1 Ack=1 Win=1500 Len=0
41	3.269194	192.168.11.12	192.168.11.37	TCP	68	10346 → 61510 [PSH, ACK] Seq=1 Ack=1 Win=1500 Len=13
42	3.319752	192.168.11.37	192.168.11.12	TCP	54	61510 → 10346 [ACK] Seq=1 Ack=1 Win=1500 Len=0
68	3.931465	192.168.11.12	192.168.11.37	TCP	67	10346 → 61510 [PSH, ACK] Seq=1 Ack=1 Win=1500 Len=13
69	3.977029	192.168.11.37	192.168.11.12	TCP	54	61510 → 10346 [ACK] Seq=1 Ack=1 Win=1500 Len=0
72	4.108003	192.168.11.12	192.168.11.37	TCP	68	10346 → 61510 [PSH, ACK] Seq=1 Ack=1 Win=1500 Len=13
73	4.149377	192.168.11.37	192.168.11.12	TCP	54	61510 → 10346 [ACK] Seq=1 Ack=1 Win=1500 Len=0

Frame 27: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
 Ethernet II, Src: Microchi_e1:8b:ef (68:27:19:e1:8b:ef), Dst: WistronI_d6:11:71:05:28:6a
 Internet Protocol Version 4, Src: 192.168.11.12, Dst: 192.168.11.37
 Transmission Control Protocol, Src Port: 10346, Dst Port: 61510, Seq: 14, Ack: 1, Win: 1500, Len: 13
 Data (14 bytes):
 0000 98 ee cb d6 11 7a 68 27 19 e1 8b ef 08 00 45 00zh'.....E-
 0010 00 36 06 f4 00 00 64 06 b8 4c c0 a8 0b 0c c0 a8 -6....d...L.....
 0020 0b 25 28 6a f0 46 05 9e ca ce 6a 26 0a dc 50 18 -%(j-F...j&-P...
 0030 05 dc 1b ae 00 00 42 31 3d 53 57 3a 34 34 3a 4fB1-SN:44:0
 0040 46 46 0d 0a FF

Device Type: ☐ Boeing ☒ Airbus

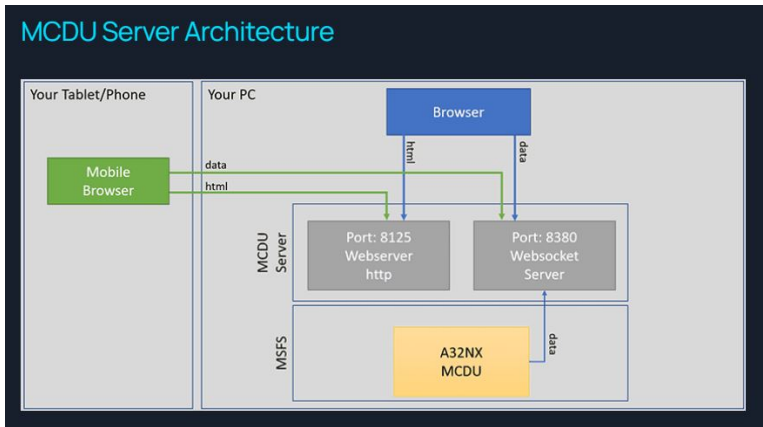
Outputs:

- ☐ LED 1
- ☒ LED 2
- ☒ LED 3
- ☐ LED 4
- ☐ LED 5
- ☐ LED 6
- ☐ LED 7
- ☐ LED 8
- ☐ LED 9
- ☐ LED 10

17 Inputs received

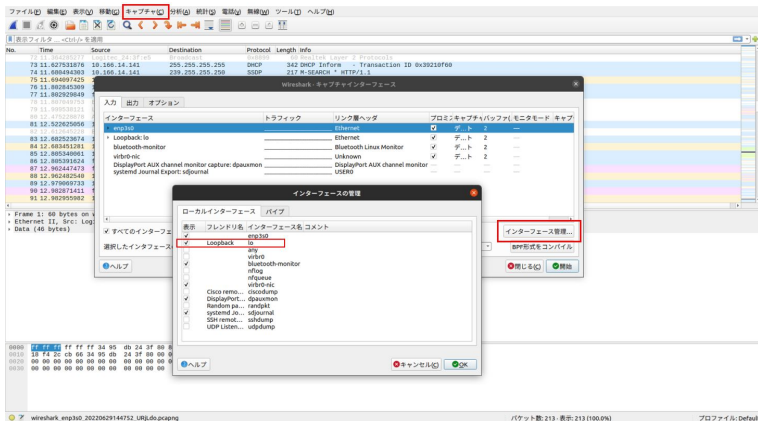
Reset

MCDU server architecture (Web socket interface)



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

Wireshark Adapter selection (to capture local socket)



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

Capture the web socket data (using Wireshark)

tcpstream 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	1.994849	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_{45F8DC29-4A93-46F5-8485-98274788502D}, id 0

> Ethernet II, Src: WistronI_d6:11:7a (98:ee:c8: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

▼ 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 01 07 65 76 65 6e 74 3a 33  ....even t:3

```

Command translation table (TCP <-> WebSocket)

```

C MCDU_comm2.h
C:\Users> a5082794 > Downloads > C MCDU_comm2.h
1 1: "DOT",
2 2: "event:0",
3 3: "event:PLUSMINUS",
4 4: "event:2",
5 5: "event:DIV",
6 6: "event:SP",
7 7: "event:OVFY",
8 8: "event:CLR",
9 9: "event:7",
10 10: "event:8",
11 11: "event:9",
12 12: "event:U",
13 13: "event:V",
14 14: "event:W",
15 15: "event:X",
16 16: "event:Y",
17 17: "event:4",
18 18: "event:5",
19 19: "event:6",
20 20: "event:P",
21 21: "event:Q",
22 22: "event:R",
23 23: "event:S",
24 24: "event:T",
25 25: "event:1",
26 26: "event:2",
27 27: "event:3",
28 28: "event:K",
29 29: "event:L",
30 30: "event:M",
31 31: "event:N",
32 32: "event:0",

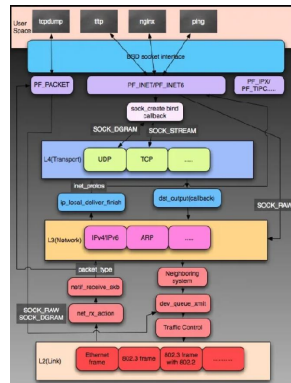
C MCDU_comm2.h
C:\Users> a5082794 > Downloads > C MCDU_comm2.h
30 29: "event:L",
31 30: "event:M",
32 31: "event:N",
33 32: "event:0",
34 33: "event:PREVPAGE",
35 34: "event:UP",
36 35: "event:L1",
37 36: "event:F",
38 37: "event:G",
39 38: "event:H",
40 39: "event:I",
41 40: "event:J",
42 41: "event:AIRPORT",
43 43: "event:L2",
44 44: "event:A",
45 45: "event:B",
46 46: "event:C",
47 47: "event:D",
48 48: "event:E",
49 49: "event:FPLN",
50 50: "event:RAD",
51 51: "event:L3",
52 52: "event:FUEL",
53 53: "event:SEC",
54 54: "event:ATC",
55 55: "event:MENU",
56 56: "event:L5",
57 57: "event:DIR",
58 58: "event:PROG",
59 59: "event:L4",
60 60: "event:PERF",
61 61: "event:INIT",
62 62: "event:DATA",

C MCDU_comm2.h
C:\Users> a5082794 > Downloads > C MCDU_comm2.h
41 40: "event:J",
42 41: "event:AIRPORT",
43 43: "event:L2",
44 44: "event:A",
45 45: "event:B",
46 46: "event:C",
47 47: "event:D",
48 48: "event:E",
49 49: "event:FPLN",
50 50: "event:RAD",
51 51: "event:L3",
52 52: "event:FUEL",
53 53: "event:SEC",
54 54: "event:ATC",
55 55: "event:MENU",
56 56: "event:L5",
57 57: "event:DIR",
58 58: "event:PROG",
59 59: "event:L4",
60 60: "event:PERF",
61 61: "event:INIT",
62 62: "event:DATA",
73 }
  
```

Network programming (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)

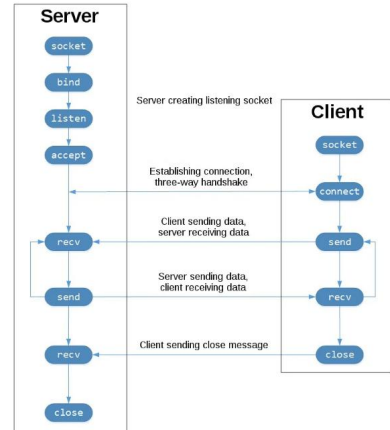


You can easily break the entire system with the bad network code

Network programming (Use Python TCP/IP support)

Python socket library

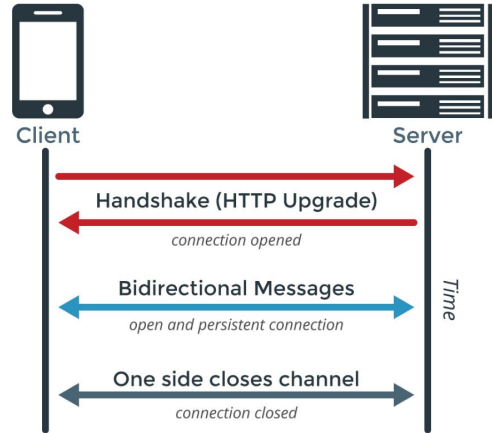
- To send/receive TCP packet to MCDU HW
- 1: create socket
- 2: bind (address, port)
- 3S:listen (remote connection)
- 3C:connect (to server)
- 4: send/rcv
- 5: **close**



Network programming (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)

```

84 class BaseClient:
85     def __init__(self, timeout:int=10, buffer:int=1024):
86         self.__socket = None
87         self.__websocket = None
88         self.__address = None
89         self.__timeout = timeout
90         self.__buffer = buffer
91
92     def connect(self, address, family:int, typ:int, proto:int):
93         self.__address = address
94         self.__socket = socket.socket(family, typ, proto)
95         self.__websocket = create_connection("ws://192.168.11.37:8380/")
96         self.__socket.settimeout(self.__timeout)
97         self.__socket.connect(self.__address)
98
99     def send(self, message:str="") -> None:
100         flag = False
101         while True:
102             try:
103                 message_recv = self.__socket.recv(self.__buffer).decode('utf-8')
104                 self.receive(message_recv)
105                 if flag:
106                     break
107             except KeyboardInterrupt:
108                 print(" Caught CTRL+C")
109                 self.__socket.shutdown(socket.SHUT_RDWR)
110                 self.__socket.close()
111                 sys.exit(0)
112             try:
113                 self.__socket.shutdown(socket.SHUT_RDWR)
114                 self.__socket.close()
115             except:
116                 pass

```

```

117
118     def received(self, message:str):
119         #print(message)
120         if (message.split(':')[1].strip()) == "ON":
121             key_id = message.rsplit(':')[2]
122             if (key_id in MCDU_comm2):
123                 key_server = MCDU_comm2[key_id]
124                 print(key_server)
125                 self.__websocket.send(str(key_server))
126
127 class InetClient(BaseClient):
128     def __init__(self, host:str="192.168.11.12", port:int=10346) -> None:
129         self.server=(host,port)
130         super().__init__(timeout=6000, buffer=1024)
131         super().connect(self.server, socket.AF_INET, socket.SOCK_STREAM, 0)
132
133
134     def signal_handler(signal, frame):
135         # close the socket here
136         self.__socket.shutdown(socket.SHUT_RDWR)
137         self.__socket.close()
138         sys.exit(0)
139         signal.signal(signal.SIGINT, signal_handler)
140
141
142 if __name__=="__main__":
143     cli = InetClient()
144     cli.send()
145

```

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