Implementing a usrsock based Wi-Fi driver on NuttX

Masayuki Ishikawa

Sony Home Entertainment & Sound Products, Inc

August 15-16 2020

NuttX Online Workshop

About Me

SONY

Masayuki Ishikawa

Senior Software Engineer
at Sony Home Entertainment & Sound Products Inc.

NW-A800

NW-ZX1

Technical background

- 3D Graphics, Home Networking, Internet-to-Home, Embedded Systems

Product development

- Portable Media Player (Linux/Android)
- Digital Voice Recorder, Music Player, Headphone (NuttX)

Public talks

- Arm Techcon 2016, ELC2017NA, OpenIoT2018NA, NuttX2019, ELC2019NA/E

ICD-UX560

WF-SP900

Agenda

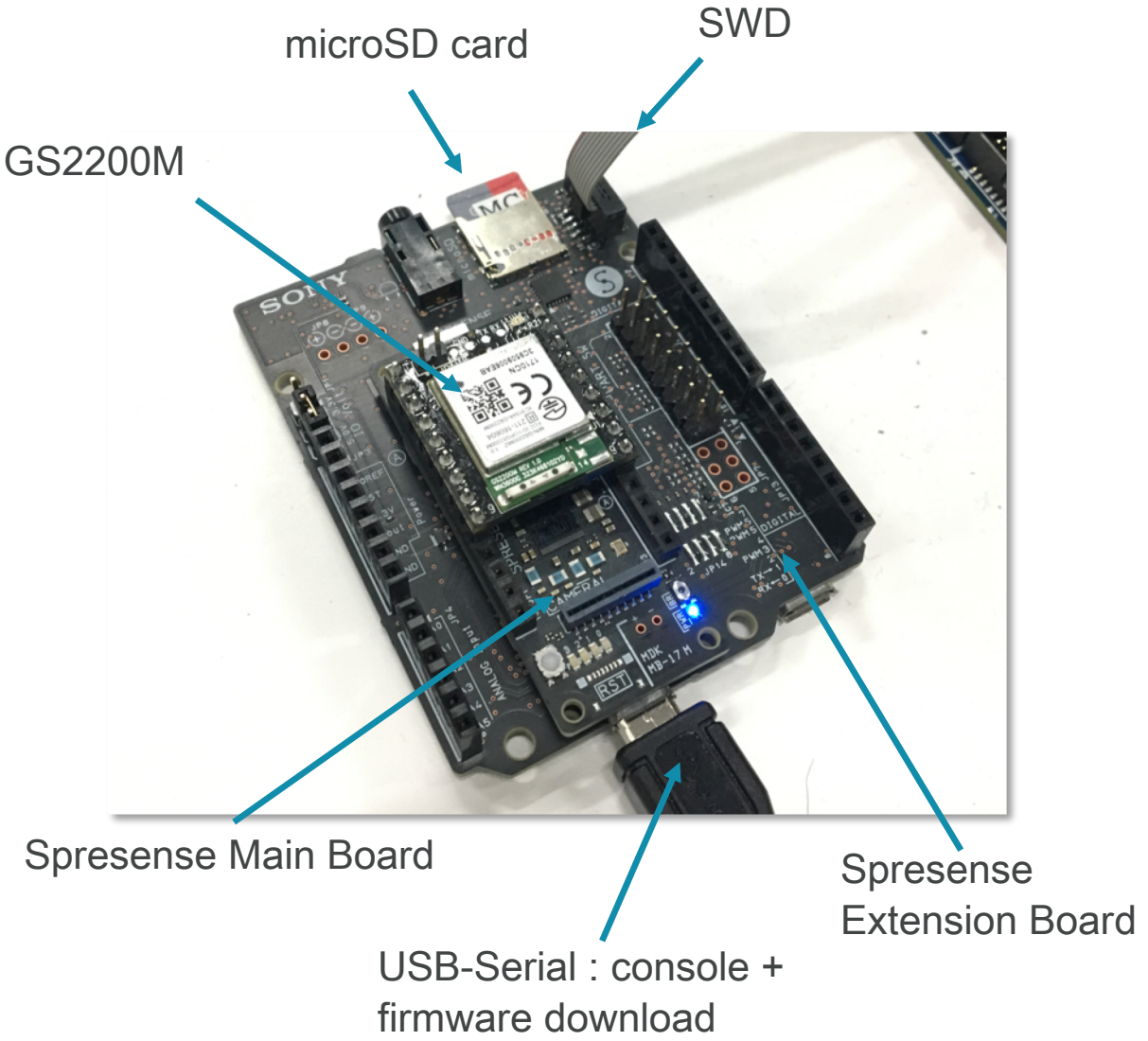
SONY

- Hardware
  - Spresense + Telit GS2200M
  - STM32F4Discovery + Telit GS2200M
  - Read & Write transaction on SPI
- Software
  - Architecture
  - What is usrsock?
  - Serial-To-Wi-Fi
- Actual use-cases
- Demo video

Hardware : Spresense + GS2200M

SONY

- Spresense (main + extension)
  - Arm Cortex-M4F x 6 (up to 160MHz)
  - SRAM 1.5MB
  - microSDHC
  - SPI
    - High performance mode: up to 13.00 Mbps
- Telit GS2200M *
  - Radio: 802.11b/g/n (2.4GHz only)
  - Voltage: VDDIO 1.8-3.3V, VIN3.3V
  - Interface**: UART/SPI (up to 10Mbps)/SDIO
  - Embedded TCP/IP stack



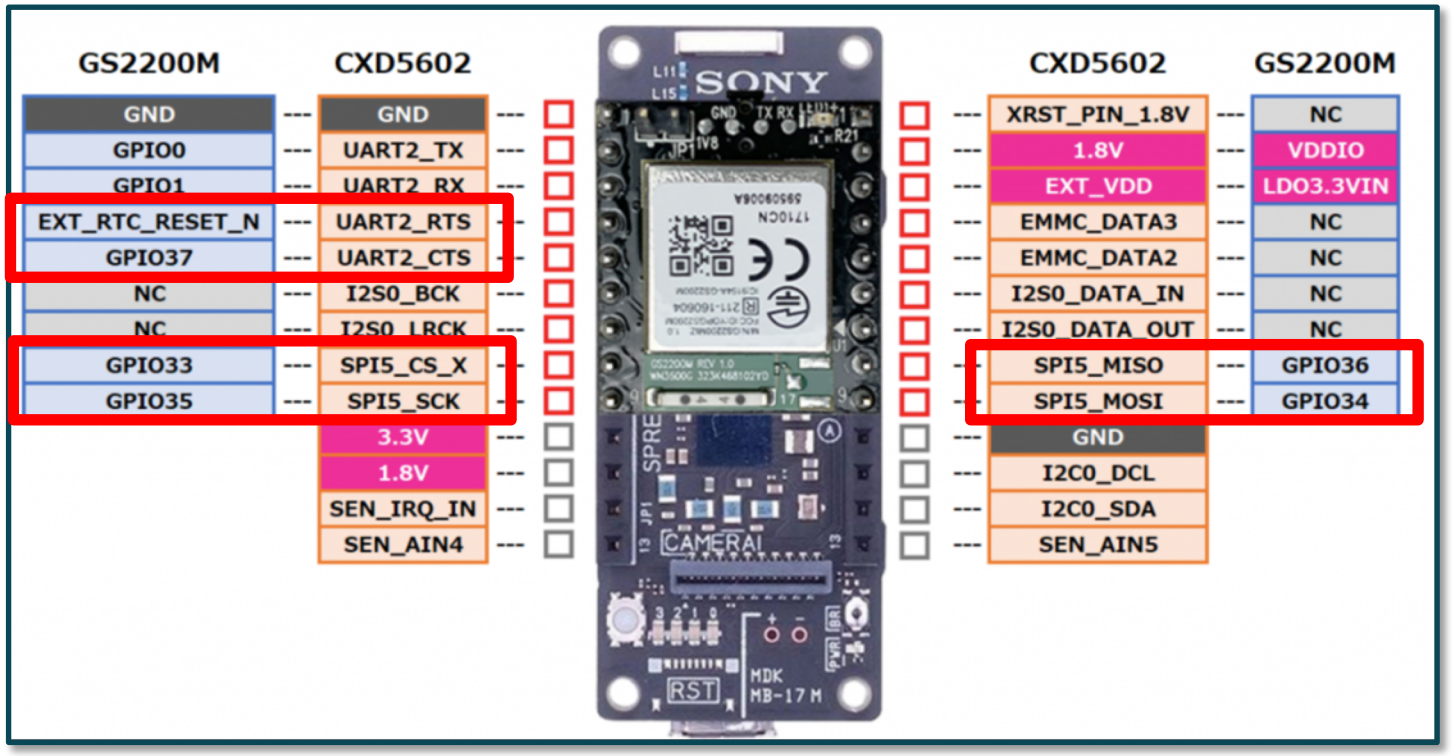
THE APACHE SOFTWARE FOUNDATION

* You can buy the Wi-Fi board from chip1stop

**Default interface is configured by firmware

Pin assignments

- I/O voltage: 1.8V
- Reset pin : UART2_RTS
- IRQ pin : UART2_CTS *
- Interface : SPI5



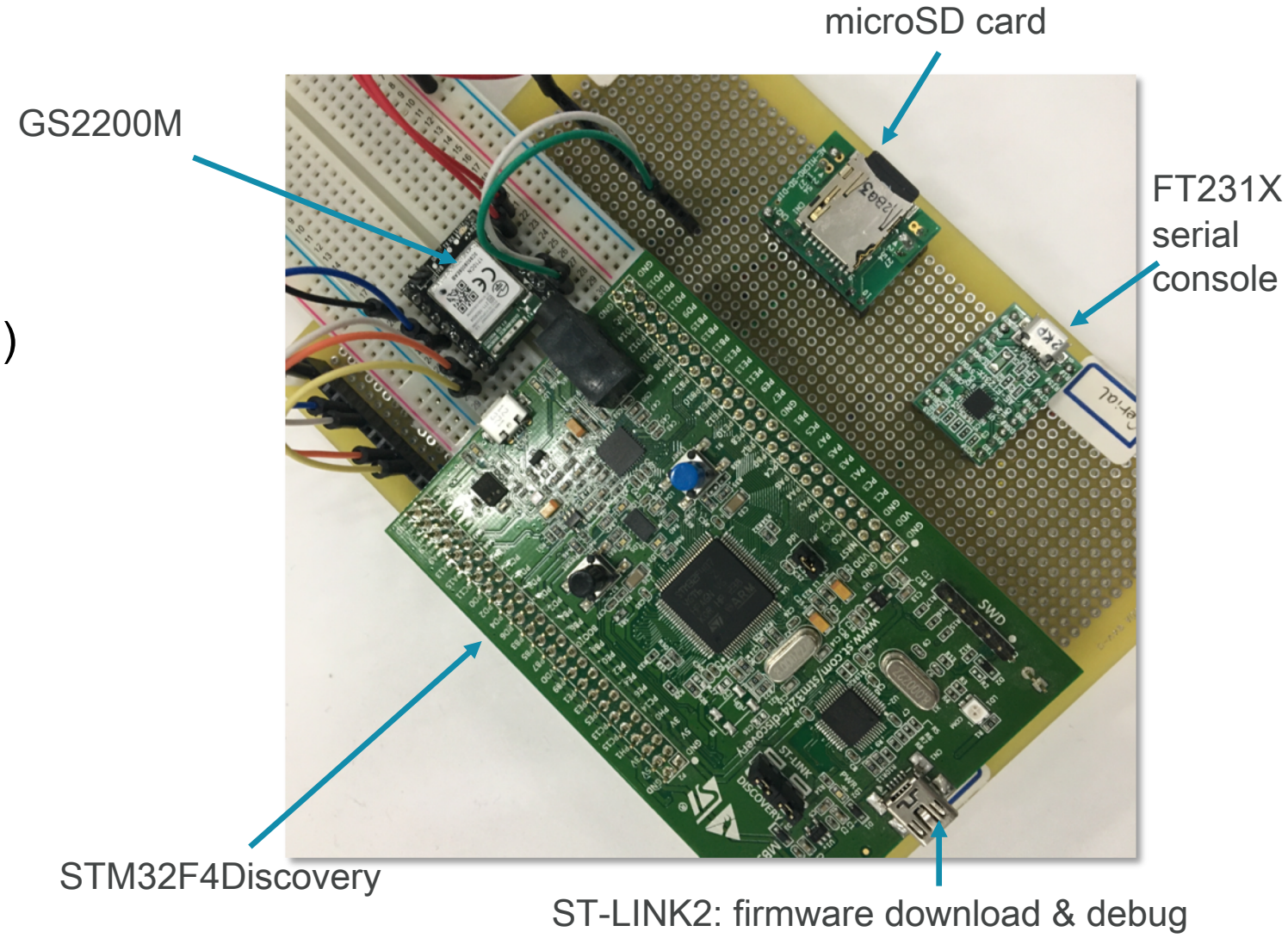
https://idy-design.com/product/is110b.html

*GPIO37 in GS2200M is assigned for IRQ. See also : 1VV0301396_GS2200M_HW_User_Guide_r0.pdf

Hardware : STM32F4Discovery + GS2200M

SONY

- STM32F4Discovery
  - Arm Cortex-M4F (168MHz)
  - SRAM 128KB + 64KB (CCM)
  - microSD over SPI (SPI2 is assigned)
    - SDHCI can be used
- Telit GS2200M
  - Radio: 802.11b/g/n (2.4GHz only)
  - Voltage: VDDIO3.3V, VIN3.3V
  - Interface: UART/SPI/SDIO
  - Embedded TCP/IP stack



Pin assignments

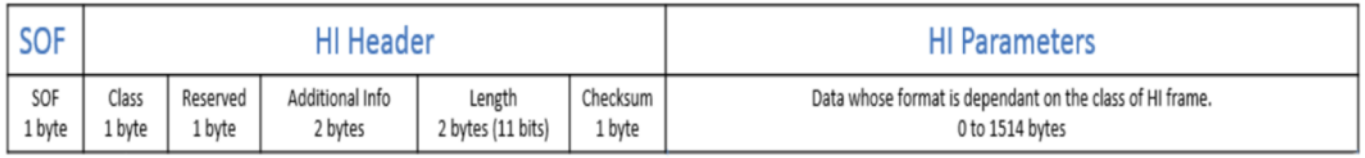
SONY

- FT231X to STM32F407
  - RXD : PA2 (USART2_TX)
  - TXD : PA3 (USART2_RX)
- microSD to STM32F407
  - PIN2 (CD/D3) : PB12 (for chip select)
  - PIN5 (CLK) : PB13 (SPI2_SCK)
  - PIN3 (CMD) : PB15 (SPI2_MOSI)
  - PIN7 (D0) : PB14 (SPI2_MISO)
- GS2200M to STM32F407
  - GPIO33 : PE5 (for chip select)
  - GPIO35 : PB3 (SPI3_SCK)
  - GPIO34 : PB5 (SPI3_MOSI)
  - GPIO36 : PB4 (SPI3_MISO)
  - GPIO37 : PD2 (for interrupt)
  - EXT_RTC_RESET_N : PE4 (for reset)
  - VDDIO : 3.3V

HI frame format on SPI (from MCU)*

SONY

Figure 19 HI Frame Format (From Host Side)



SOF	HI Header					HI Parameters
SOF 1 byte	Class 1 byte	Reserved 1 byte	Additional Info 2 bytes	Length 2 bytes (11 bits)	Checksum 1 byte	Data whose format is dependant on the class of HI frame. 0 to 1514 bytes

Table 20 HI Parameters Service Class Identifiers

Identifiers	Description
Start of frame	0xA5
Class	0x01 - WRITE_REQUEST from MCU side 0x02 - READ_REQUEST from MCU side 0x03 - DATA from MCU side
Reserved	0x00
Additional Info	0x00,0x00
Length	Maximum 2032
CheckSum	A single checksum byte is used, computed as the 1's complement of the 8-bit long (modulo-256) sum of all the bytes of the HI HEADER (not including the Start delimiter).

*See GS2200M-S2W-Adapter-Command-Reference-Guide_r3.0.pdf

HI frame format on SPI (from GS node)

SONY

Figure 20 HI Frame Response (from GS Node)

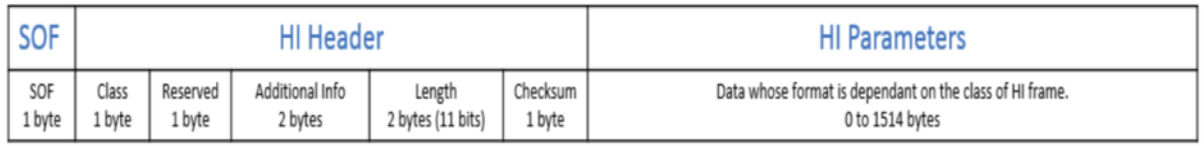


Table 21 HI Frame Response (from GS Node)

Identifier	Description
Start of frame	0xA5
Class	0x11 - WRITE_RESPONSE_OK to MCU side 0x12 - READ_RESPONSE_OK to MCU side 0x13 - WRITE_RESPONSE_NOK to MCU side 0x14 - READ_RESPONSE_NOK to MCU side 0x15 - DATA to MCU side
Reserved	0x00
Additional Info	0x00,0x00 0x00, 0x01 - Pending Data for transfer from GS2000 to MCU
Length	0 (No Data)
CheckSum	A single checksum byte is used, computed as the 1's complement of the 8-bit long (modulo-256) sum of all the bytes of the HI HEADER (not including the Start delimiter).

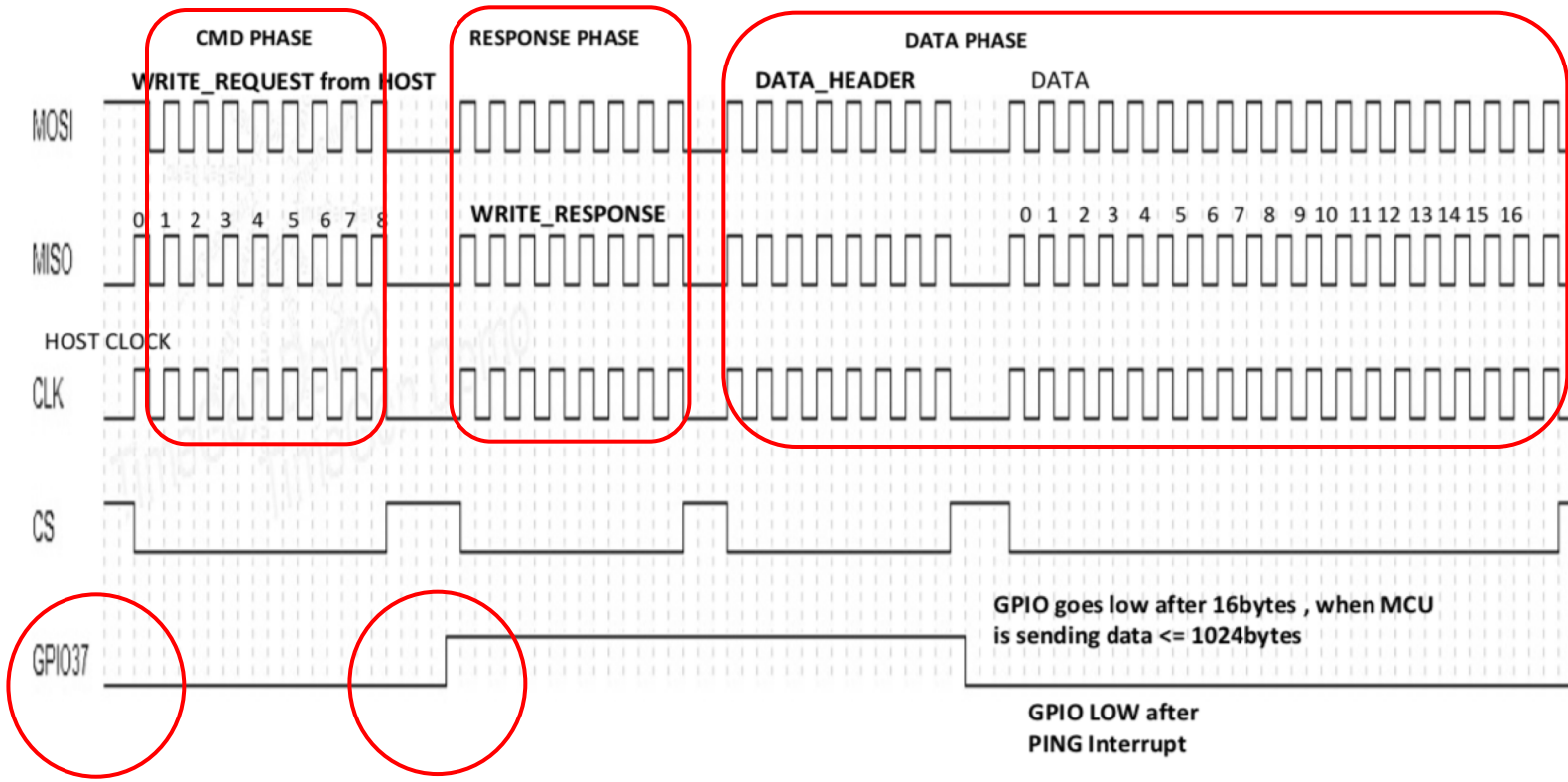
Actual data length is set when the class is READ_RESPONSE_OK

THE APACHE SOFTWARE FOUNDATION

Write transaction on SPI

SONY

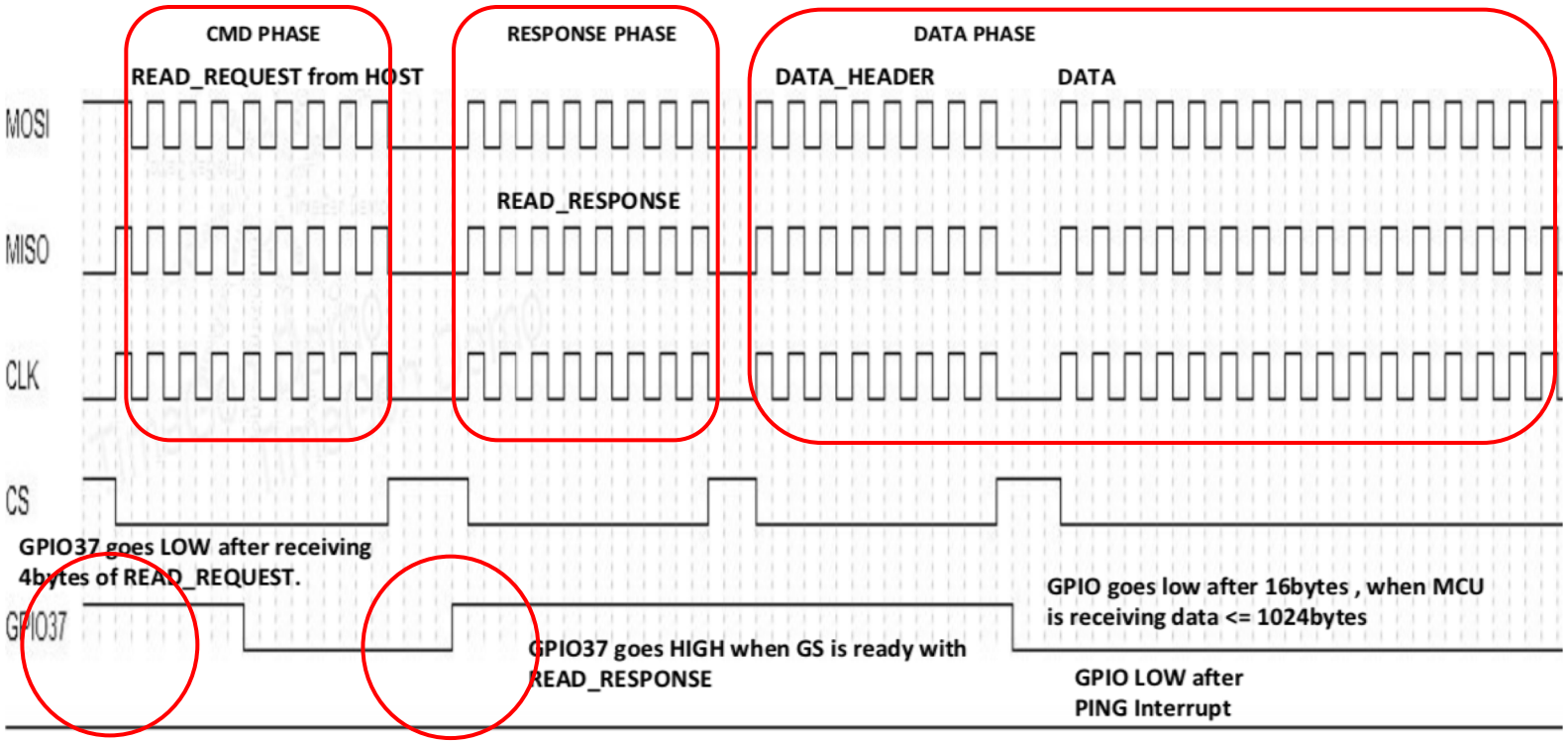
- Start with GPIO37=L
- Send WRITE_REQUEST
- Wait for GPIO37=H
- Receive WRITE_RESPONSE
- Send DATA_HEADER and DATA



Read transaction on SPI

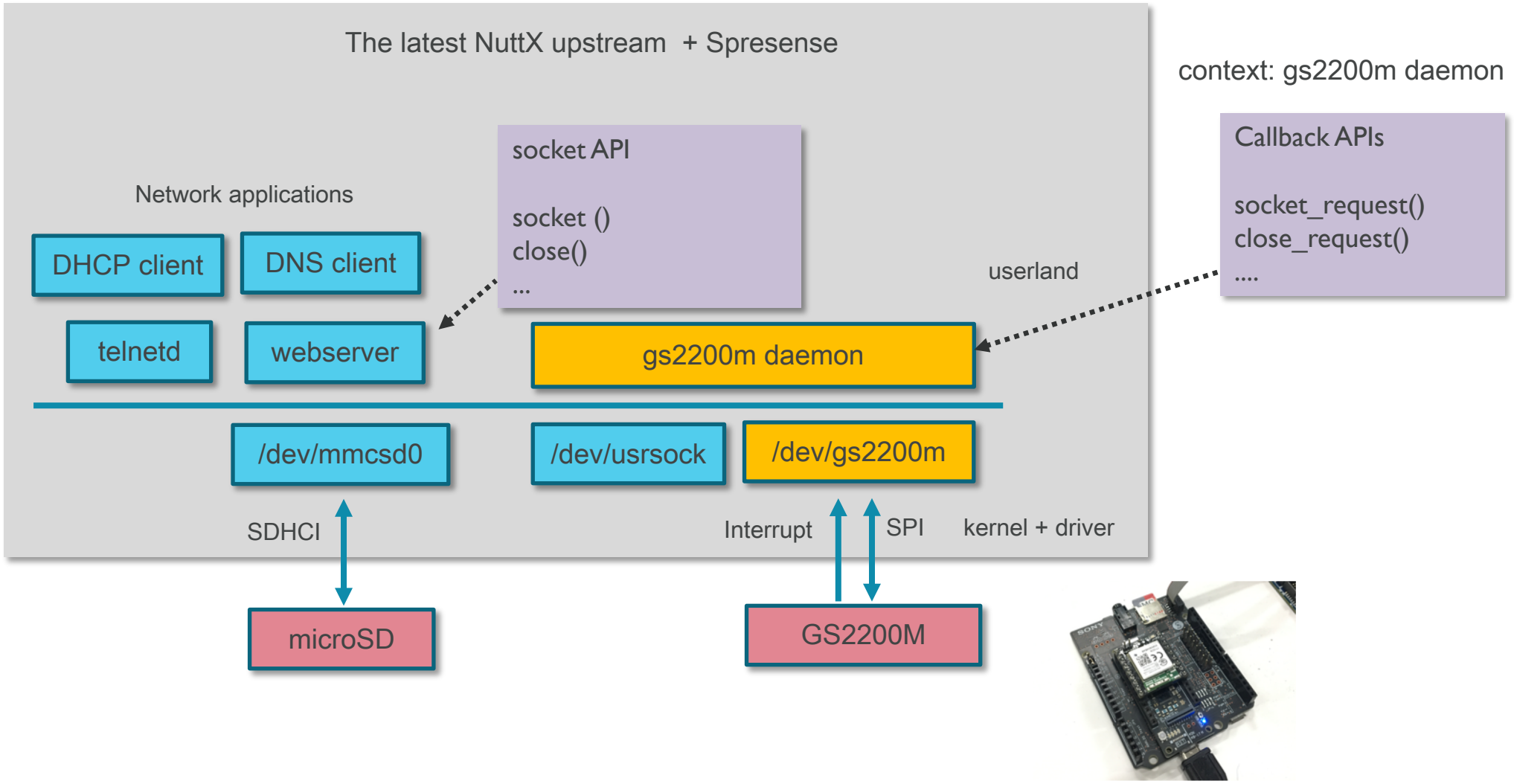
SONY

- Start with GPIO37=H
- Send READ_REQUEST
- Wait for GPIO37=H
- Receive READ_REPONSE
- Receive DATA_HEADER and DATA



Software architecture

SONY

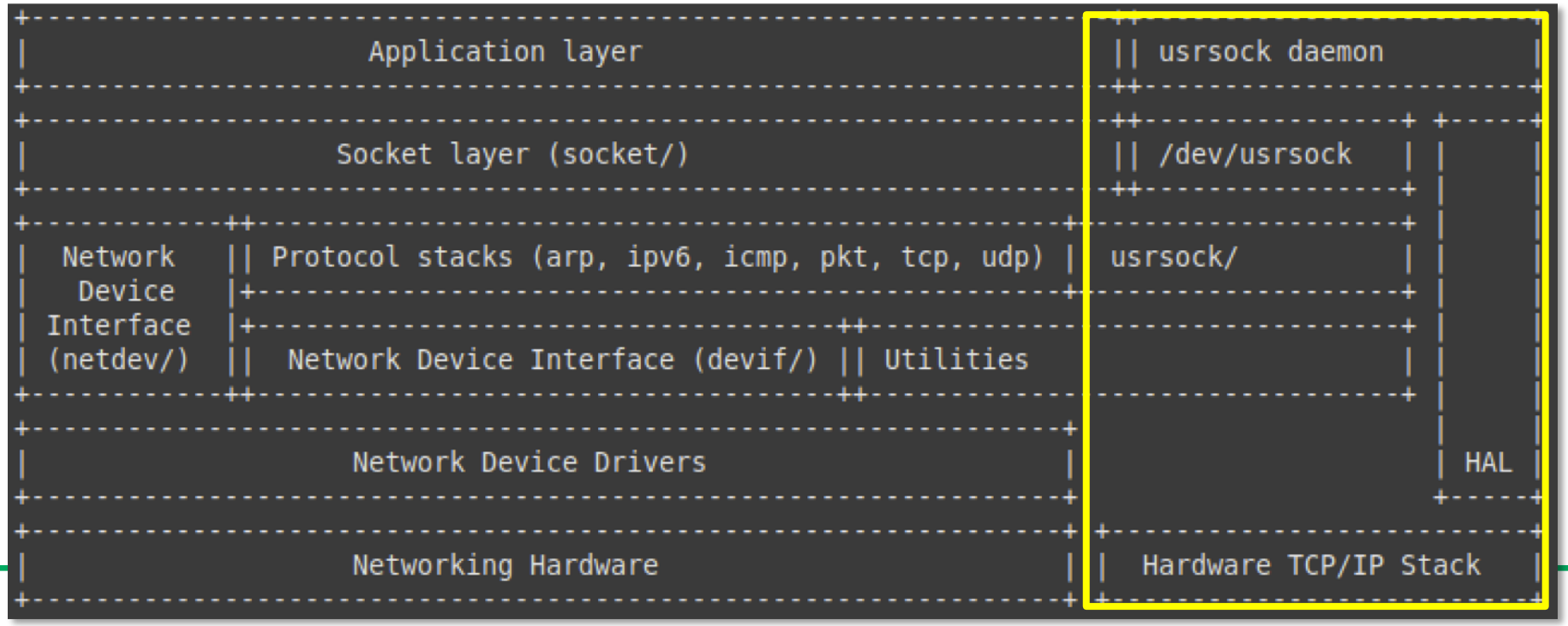


NOTE: Though GS2200M supports high-level protocols such as HTTP(S)/MQTT, in this driver we only use low level.

What is the usrsock ?

SONY

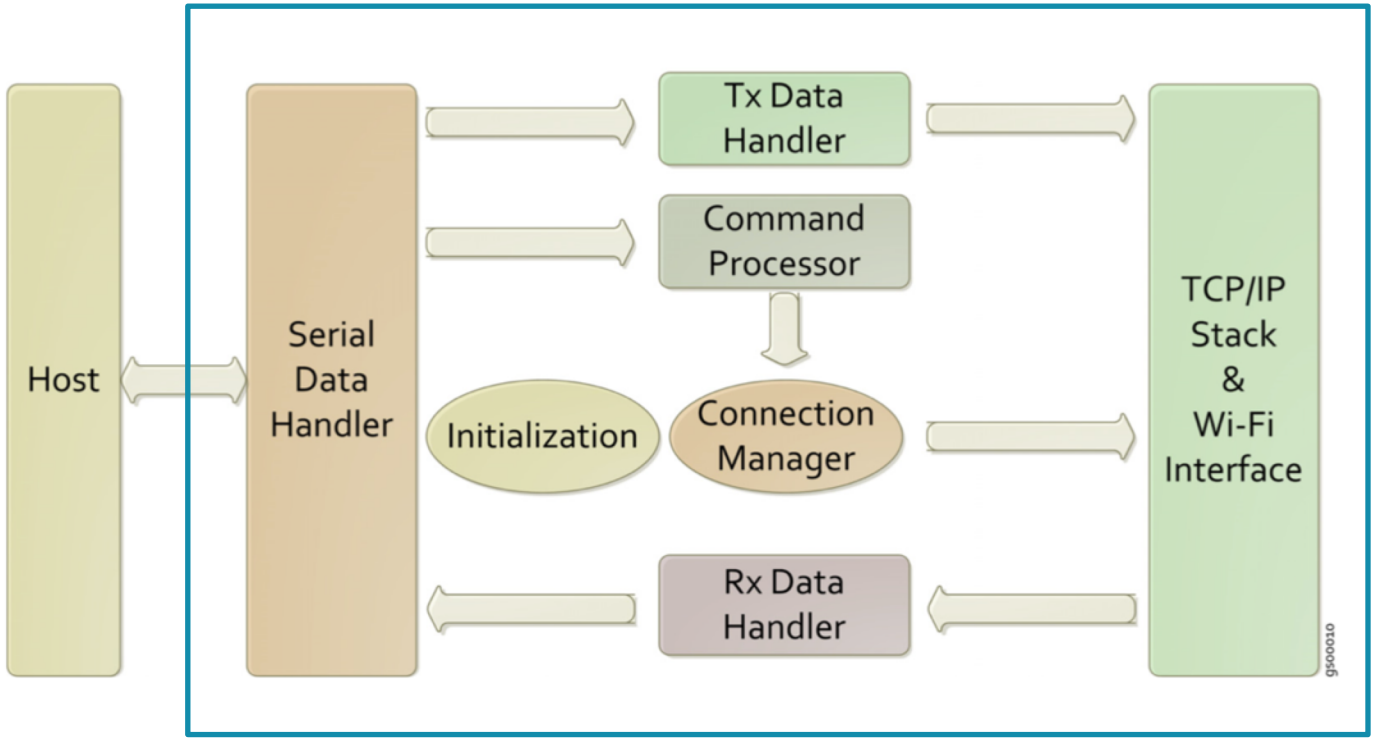
- User-space networking stack API
- User-space daemon and HAL provide NuttX networking features
- This allows seamless integration of HW-provided TCP/IP stacks to NuttX



Serial-to-WiFi on GS2200M

SONY

- The Serial-to-WiFi stack is used to provide WiFi capability to any device having a serial interface.
- This approach offloads WLAN, TCP/IP stack and network management overhead to the WiFi chip, allowing a small embedded host (for example an MCU) to communicate with other hosts on the network using a WiFi wireless link.
- The host processor can use serial commands to configure the Serial-to-WiFi Application and to create wireless and network connections.



Serial-to-WiFi

NOTE: IP-to-Wi-Fi is also possible, if you build with SDK builder.

AT command examples (1/2)

SONY

Table 497 Network AT Supported Commands

Command	Parameters	Response / Effect
AT+NDHCP	n[,<hostname>,<radio mode>,<lease period>,<retry interval>]	Enable or disable DHCP client support for IPV4 parameters.
AT+NSET	<Src Address>,<Net-mask>,<Gateway>	Static network parameters overrides previous values.
AT+WA	<SSID>,[,[<BSSID>][,<Ch>],[Rssi Flag],[WPS Registrar],[Unscheduled automatic power save delivery configuration]]	Associate to specified SSID, BSSID, and channel. RSSI is an optional parameter with values: • 1 - associate to the AP specified by SSID with highest RSSI value. • 0 - associate to the AP specified by SSID without considering RSSI value. This is the default settings.

AT command examples (2/2)

SONY

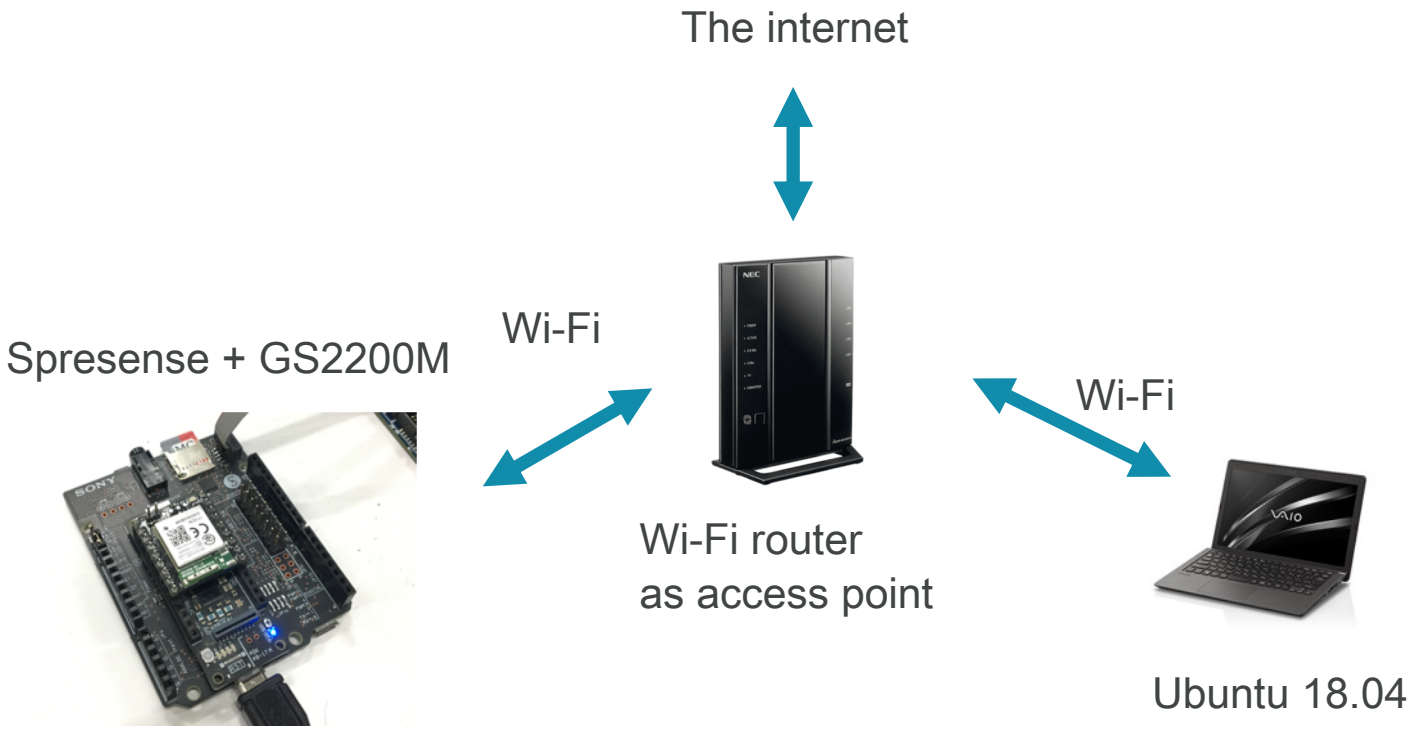
Command	Parameters	Response / Effect
AT+NCTCP	<Dest-Address>,<Port>	Attempts TCP client connection to Destination; CONNECT<CID> if successful.
AT+NCUDP	<Dest-Address>,<Port>[<,Src. Port>]	Open UDP client socket to Destination; CONNECT<CID> if successful. The port range 0xBAC0 to 0XBACF may not be used.
AT+NSTCP	<Port>, [max client connection],<TCP window size>	Start a TCP server on Port; CONNECT<CID> if successful.
AT+NSUDP	<Port>	UDP server on Port; CONNECT<CID> if successful. The port range 0xBAC0 to 0xBACF may not be used.

NOTE: AT+NCUDP is not used in the gs2200m driver

Actual sequence with example apps

SONY

- Initialize GS2200M driver
- Connect to Wi-Fi network
- Run DHCP client *
  - BULK mode in UDP
  - CID (Connection Identifier)
  - Interrupt and work queue
- Run wget command
  - BULK mode in TCP
  - TCP flow control
- Run telnet daemon



*NOTE: Need to disable DHCP client inside GS2200, because it conflicts.

Driver initialization (1/2)

SONY

- board_gs2200m_initialize()
  - Called vial board_app_initialize()
  - Change UART2 pins to GPIO
  - Change eMMC pins to SPI5
  - Call gs2200m_register()
- gs2200m_register()
  - Call gs2200m_initialize()
    - Set SPI to mode1/8bits/10MHz
    - Reset and un-reset the module
  - Call lower->attach(gs2200m_irq, dev) to attach IRQ

```
(gdb) where
#0  gs2200m_initialize (dev=0xd07f140, lower=0xd0708fc <g_wifi_lower>) at wireless/gs2200m.c:3169
#1  0x0d05ad9c in gs2200m_register (devpath=0xd070838 "/dev/gs2200m", spi=0xd0735e4 <g_spi5dev>, lower=0xd0708f\
c <g_wifi_lower>) at wireless/gs2200m.c:3238
#2  0x0d04c50c in board_gs2200m_initialize (devpath=0xd070838 "/dev/gs2200m", bus=5) at src/cxd56_gs2200m.c:280
#3  0x0d04bc40 in cxd56_bringup () at board/cxd56_bringup.c:456
#4  0x0d04b7d4 in board_app_initialize (arg=0) at board/cxd56_appinit.c:92
#5  0x0d032a12 in boardctl (cmd=65281, arg=0) at boardctl.c:326
#6  0x0d01be2a in nsh_initialize () at nsh_init.c:103
#7  0x0d01bde4 in nsh_main (argc=1, argv=0xd07a840) at nsh_main.c:143
#8  0x0d018246 in spresense_main (argc=1, argv=0xd07a840) at board/cxd56_main.c:55
#9  0x0d00a42c in nxtask_startup (entrypt=0xd018235 <spresense_main>, argc=1, argv=0xd07a840) at sched/task_sta\
rtup.c:165
#10 0x0d006746 in nxtask_start () at task/task_start.c:147
```

Driver initialization (2/2)

SONY

- gs2200m_start()
  - Wait for GPIO37 to High
  - Check boot message
    - `Serial2WiFi App`
  - Disable echo
    - `ATE0`
  - Activate RX
    - `AT+WRXACTIVE=1`
  - Set network interface to ‘UP’
  - Enable interrupt

```
NuttShell (NSH) NuttX-9.1.0
nsh> ps
  PID PRI POLICY   TYPE     NPX STATE    EVENT      SIGMASK   STACK COMMAND
    0   0 FIFO     Kthread  N-- Ready               00000000 000000 Idle Task
    1 224 FIFO     Kthread  --- Waiting  Signal     00000000 002052 hpwork
    2  60 FIFO     Kthread  --- Waiting  Signal     00000000 002052 lpwork
    3 100 FIFO     Task     --- Running              00000000 002052 init
    4 200 FIFO     Task     --- Waiting  MQ empty   00000000 001020 cxd56_pm_
nsh> ifconfig
eth0    Link encap:Ethernet HWaddr 00:00:00:00:00:00 at UP
        inet addr:0.0.0.0 DRaddr:0.0.0.0 Mask:0.0.0.0
```

Connect to Wi-Fi network

SONY

- Disassociate
  - `AT+WD`
- Set to STA mode
  - `AT+WM=0`
- Disable DHCP client
  - `AT+NDHCP=0`
- Set address *
  - `AT+NSET=....`
- Get mac address info
  - `AT+NMAC=?`
- Join the network
  - `AT+WA=...`

```
nsh> gs2200m aterm-ac16fc-g wifi-test-24g &
gs2200m [5:50]
nsh> ps
  PID PRI POLICY   TYPE    NPX STATE    EVENT     SIGMASK   STACK COMMAND
    0   0 FIFO     Kthread N-- Ready              00000000 000000 Idle Task
    1 224 FIFO     Kthread --- Waiting  Signal    00000000 002052 hpwork
    2  60 FIFO     Kthread --- Waiting  Signal    00000000 002052 lpwork
    3 100 FIFO     Task    --- Running            00000000 002052 init
    4 200 FIFO     Task    --- Waiting  MQ empty  00000000 001020 cxd56_pm_task
    5  50 RR       Task    --- Waiting  Semaphore 00000000 002012 gs2200m aterm
nsh> ifconfig
eth0    Link encap:Ethernet HWaddr 3c:95:09:00:6e:ab at UP
        inet addr:10.0.0.2 DRaddr:10.0.0.1 Mask:255.255.255.0
```

THE APACHE SOFTWARE FOUNDATION

*NOTE: AT command rejects to set address to “0.0.0.0”. I think this is a bug.

Run DHCP client (1/3)

SONY

- Run DHCP client
  - `nsh> renew eth0`
- Confirm the address with ifconfig
  - `nsh> ifconfig`

```
nsh> renew eth0
nsh> ifconfig
eth0    Link encap:Ethernet HWaddr 3c:95:09:00:6e:ab at UP
        inet addr:192.168.10.21 DRaddr:192.168.10.1 Mask:255.255.255.0
```

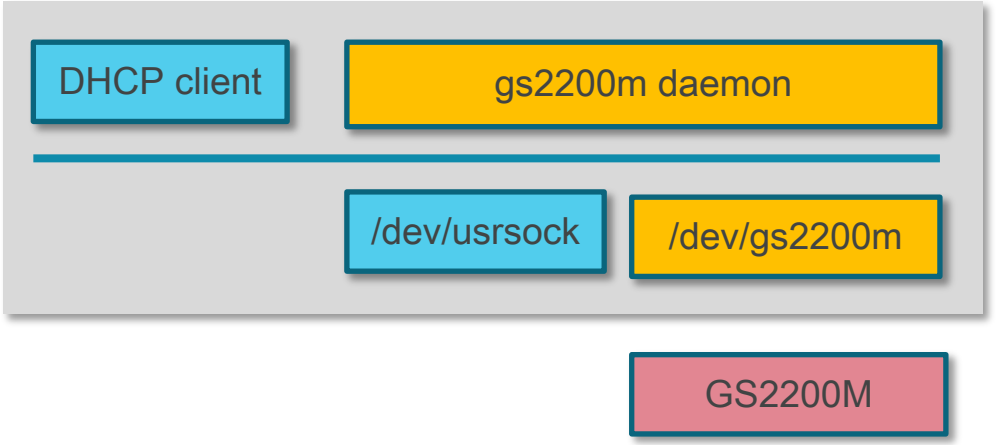
Run DHCP client(2/3)

SONY

- socket() system call
  - socket_request() in gs2200m daemon is called to allocate usockid.
  - however, no driver call happens.
- ioctl() system call
  - ioctl_request() in gs2200m daemon is called
    - This call is used for get/set interface info.
  - then call ioctl(..., GS2200M_IOC_IFREQ, ...)
- sendto() system call
  - sendto_request() in gs2200m daemon is called
  - then call ioctl(..., GS2200M_IOC_SEND, ...)
  - In the gs2200m driver, start udp server to allocate CID. *
  - then send specified data as a bulk packet.

Callback APIs

socket_request()
close_request()
connect_request()
sendto_request()
recvfrom_request()
....



* This might be tricky but in Serial-To-Wi-Fi, UDP client can only allocate a new CID for associated with destination.

Run DHCP client(3/3)

SONY

- recvfrom() system call
  - recvfrom() is blocked until a new packet is received.
  - If a new packet arrives, gs2200m_irq() is called then gs2200m_irq_worker() is called.
  - In gs2200m_irq_worker(), it receive the new packet and post semaphore to notify userland.
  - recvfrom_request() in gs2200m daemon is called then call ioctl(..., GS2200M_IOC_RECV, ...)
  - In the gs2200m driver, copy the packet to caller's buffer.
  - recvfrom() is unblocked and return to caller.
- close() system call
  - close_request() in gs2200m daemon is called.
  - then call ioctl(..., GS2200M_ICO_CLOSE, ...)
  - In the gs2200m driver, issue AT+NCLOSE to deallocate the CID

CID (Connection Identifiers) in GS2200M

SONY

- Once associated, the GS node supports instances of four types of network entities: TCP client, TCP server, UDP client and UDP server.
- Each client, or server, is associated with one or more of a possible 16 Connection Identifiers, where the CID is a single hexadecimal number.
- More than one such entity can exist simultaneously; and a TCP server can have multiple connections, each with its own CID.

BULK mode in UDP

SONY

Table 214 Data Handling Using ESC Sequences on UART Interface (Continued)

Flow Control	Data Mode (Data Type)	Connection Type	Description and Escape <ESC> Command Sequence

HW	Bulk (ASCII Text or Binary)	UDP server	This escape sequence is used when sending and receiving UDP bulk data on a UDP server connection. When this command is used, the remote address and remote port is transmitted.
			
			Module expects to receive the following data sequence from Host:
			
			`<ESC>Y<CID><IPAddress:<port>:<data length><data>`
			
			Module sends the following data sequence to Host:
			
			`<ESC>y<CID><IPAddress><space><client local port><space><data length><data>`

Destination IP address and port

Source IP address and port

Interrupt and work queue

SONY

- gs2200m_irq() : interrupt handler (top half)
  - Disable interrupt
  - Kick the work queue

- gs2200m_irq_worker() : work queue handler (bottom half)
  - Receive a packet and if CID in the packet is valid, add it to the packet queue
  - If CONNECT packet is received (i.e. accept() should be unblocked), then validate the CID
  - If DISCONNECT packet is received (i.e. TCP passive close case), then invalidate the CID
  - If the packet is pushed to the queue, set POLLIN event to unblock poll() in gs2200m daemon
  - Enable interrupt

* Normal AT command and response are handled in the caller's thread without work queue by disabling interrupt

Run wget command (1/2)

SONY

- Run wget command
  - `nsh> wget http://...`
- Check the downloaded file
  - `nsh> cat index.html`

```
nsh> cd /mnt/sd0
nsh> ls -l
/mnt/sd0:
nsh> wget http://example.com/index.html
nsh> ls -l
/mnt/sd0:
 -rw-rw-rw-    1256 index.html
nsh> cat index.html
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-
    <meta name="viewport" content="width=device-width, initial-scale
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "
```

Run wget client (2/2)

SONY

- Daemon & Driver sequence is similar to DHCP client case.
  - However, wget uses TCP (not UDP), so connect() system call and read() system calls are used
- connect() system call
  - connect_request() in gs2200m daemon is called
  - then call ioctl(..., GS2200M_IOC_CONNECT, ...)
  - In the gs2200m driver, start TCP client in GS2200M and obtain a new CID
- read() system call
  - read() is blocked until a new packet is received.
  - If a new packet is received, finally recvfrom_request() in gs2200m daemon is called
  - then call ioctl(..., GS2200M_IOC_RECV, ...)
  - In the gs2200m driver, copy packet (up to the specified length) to caller's buffer.
  - If a packet still exists for the CID, then notify userland

BULK mode in TCP

SONY

Table 214 Data Handling Using ESC Sequences on UART Interface (Continued)

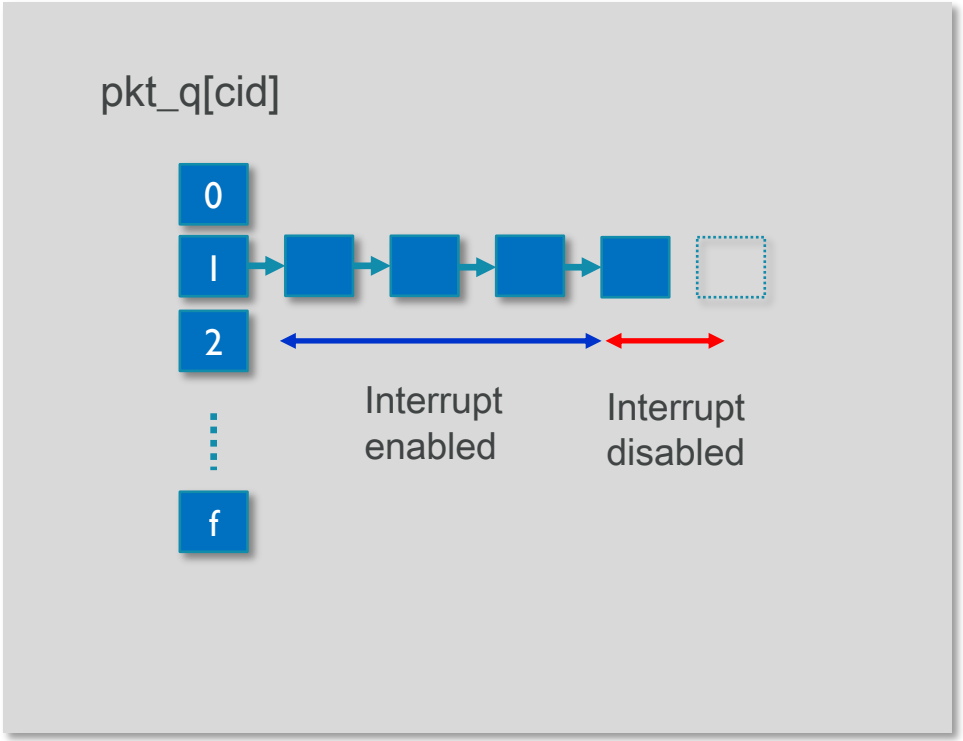
Flow Control	Data Mode (Data Type)	Connection Type	Description and Escape <ESC> Command Sequence
SW or HW	Normal (Binary)	N/A	Binary data transfer with software or hardware flow control are not supported with ESC sequence.

HW	Bulk (ASCII Text or text Binary)	TCP client
		TCP server
			To improve data transfer speed, you can use this bulk data transfer. This sequence is used to send and receive data on TCP client, TCP server, or UDP client connection.
			
			Module send and receive sequence:
			
			`<ESC>Z<CID><data length><data>`
			
			Example: to send a 5 byte user data (e.g., Hello) on CID 1, the format will be:
			
			`<ESC>Z10005Hello`

TCP flow control

SONY

- TCP flow control is needed to avoid out of memory when receiving a large data
  - Currently TCP flow control commands for Serial-To-Wi-Fi are not used
- Instead, TCP flow control is done based on total bulk packet size
  - If the total bulk packet size exceeds a threshold (e.g. 8KB), interrupt for GS2200 is disabled until the size is less than the threshold.



Run telnet daemon (1/2)

SONY

- On NuttX console
  - `nsh> ifconfig`
  - `nsh> telnetd &`
- On Ubuntu
  - `$ telnet 192.168.10.21`
  - After logging into NuttX
  - `nsh> uname –a`
  - `nsh> ps`

```
nsh> ifconfig
eth0    Link encap:Ethernet HWaddr 3c:95:09:00:6e:ab at UP
        inet addr:192.168.10.21 DRaddr:192.168.10.1 Mask:255.255.255.0

nsh> telnetd &
telnetd [7:100]
```

```
ishikawa@mbp-vmw:~$ telnet 192.168.10.21
Trying 192.168.10.21...
Connected to 192.168.10.21.
Escape character is '^]'.

NuttShell (NSH) NuttX-9.1.0
nsh> uname -a
NuttX  9.1.0 2a2dd35339-dirty Aug  2 2020 00:39:25 arm spresense
nsh> ps
  PID PRI POLICY   TYPE    NPX STATE    EVENT     SIGMASK   STACK COMMAND
    0   0 FIFO     Kthread N-- Ready              00000000 000000 Idle Task
    1 224 FIFO     Kthread --- Waiting  Signal    00000000 002052 hpwork
    2  60 FIFO     Kthread --- Waiting  Signal    00000000 002052 lpwork
    3 100 FIFO     Task    --- Waiting  Semaphore 00000000 002052 init
    4 200 FIFO     Task    --- Waiting  MQ empty  00000000 001020 cxd56_pm_
    5  50 RR       Task    --- Ready              00000000 002012 gs2200m a
    8 100 FIFO     Task    --- Waiting  Semaphore 00000000 002028 Telnet da
    9 100 FIFO     Kthread --- Waiting  Semaphore 00000000 001020 telnet_io
   11 100 FIFO     Task    --- Running            00000000 002044 Telnet se
```

Run telnet daemon (2/2)

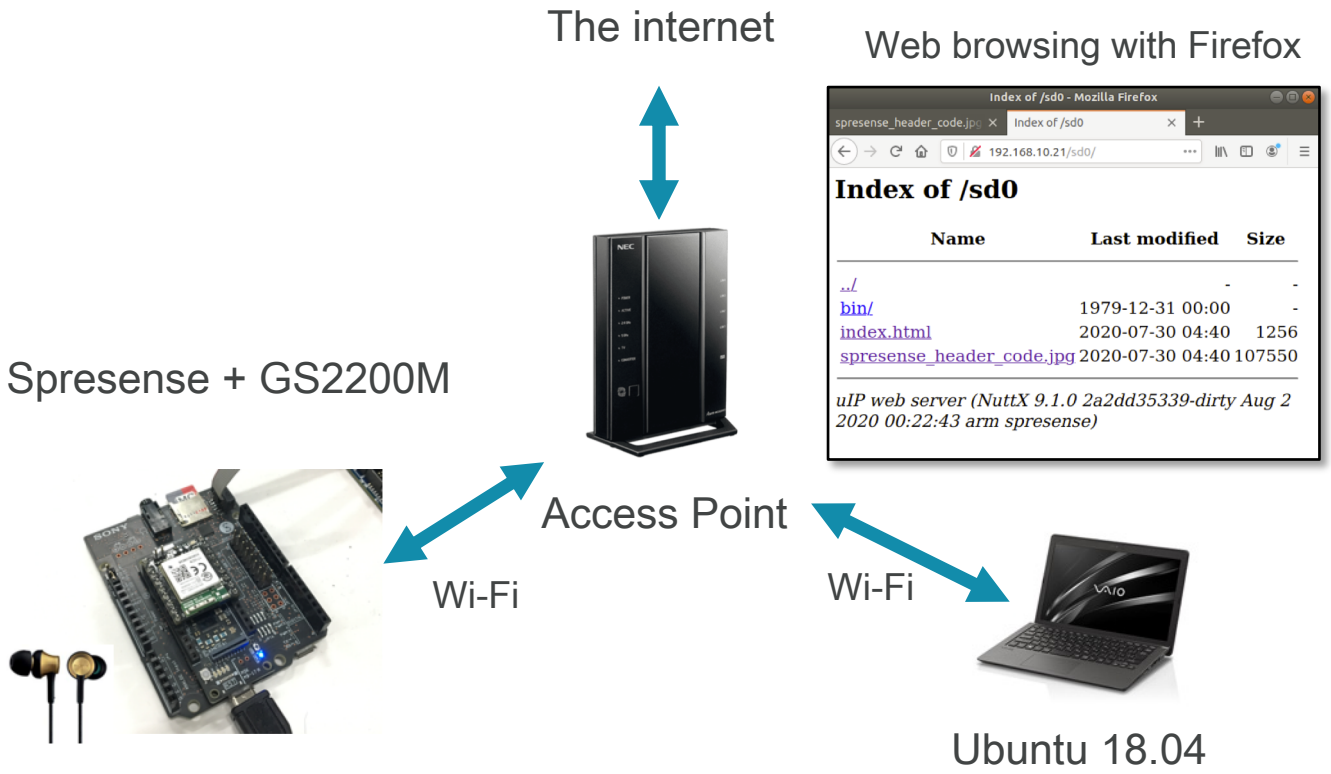
SONY

- Both DHCP client and wget command were UDP client and TCP client respectively.
  - However, telnet daemon is a TCP server program, so following system calls are newly used.
- bind() system call
  - bind_request() in gs2200m daemon is called.
  - Then call ioctl(..., GS2200M_IOC_BIND, ...) to create a TCP server in GS2200M
- listen() system call
  - listen_request() in gs2200m daemon is called but do nothing special.
- accept() system call
  - accept_request() in gs2200m daemon is called with server's CID
  - then call ioctl(..., GS2200M_IOC_ACCEPT,...) to accept connection
  - In the driver, remove the CONNECT packet

Demo videos

SONY

- Spresense + GS2200M
  - Run gs2200m deamon to connect to AP
  - Run telnetd and logging in from PC
  - Run webserver and access from PC
  - Run nxplayer for audio streaming from PC
  - Run wget to receive a file from the Internet
  - Run a downloaded ELF app from PC




NuttX Online Workshop
Thank you!
THE APACHE SOFTWARE FOUNDATION