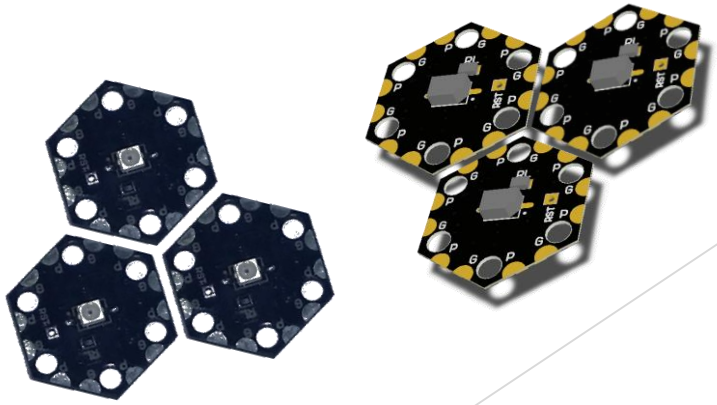
Modular Optical Wireless Elements (MOWE)

Concept & Features

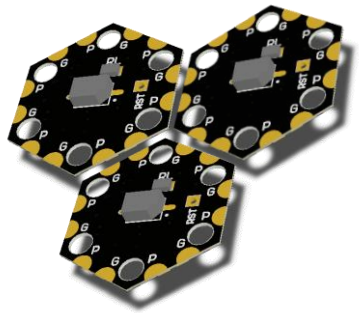


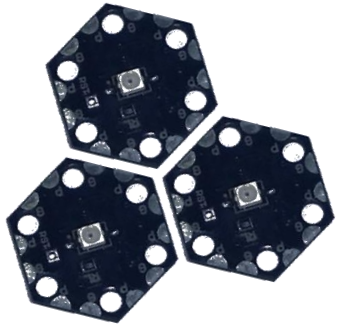
ASAAD KAADAN

June 23, 2014

Modular Optical Wireless Elements

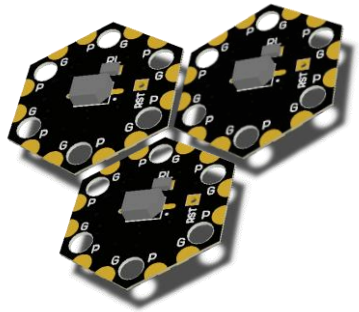
- MOWE concept.
- Hardware design.
- Setup & connection.
- Command line parser.
- Data streaming.
- Firmware update.
- Useful features.
- Useful Tools: Tandy Supercomputer Simulator.
- Useful Tools: Automatic Topology Generator.

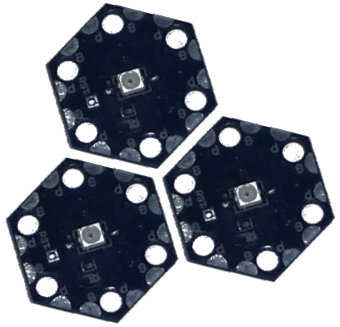




Modular Optical Wireless Elements

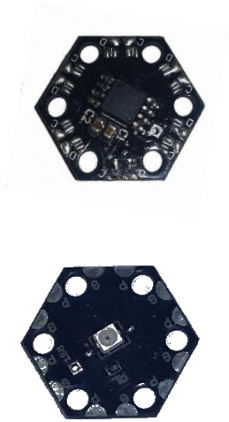
- ✓ Semiconductor-based optical communication modules featuring fully modular and distributed design.
- ✓ Inexpensive, lightweight and easy-to-assemble modules for planar and spherical optical arrays.
- ✓ Modules interconnect with each other without a separate connection apparatus (e.g., cables and connectors), reducing weight, cost and complexity while improving reliability.
- ✓ Intuitive user interface and ready-to-use APIs to facilitate various applications.
- ✓ Fully reusable, upgradable and scalable design.



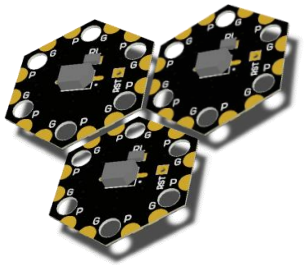


MOWE System Concept

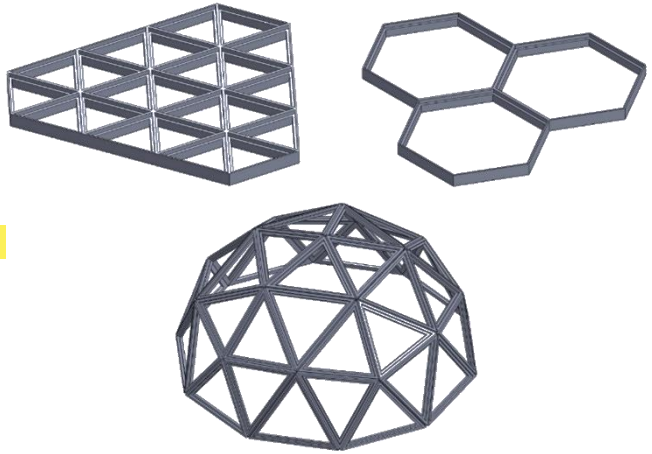
Modules



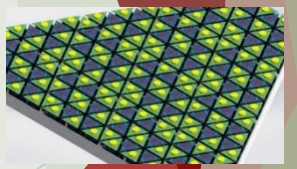
Array



Frame

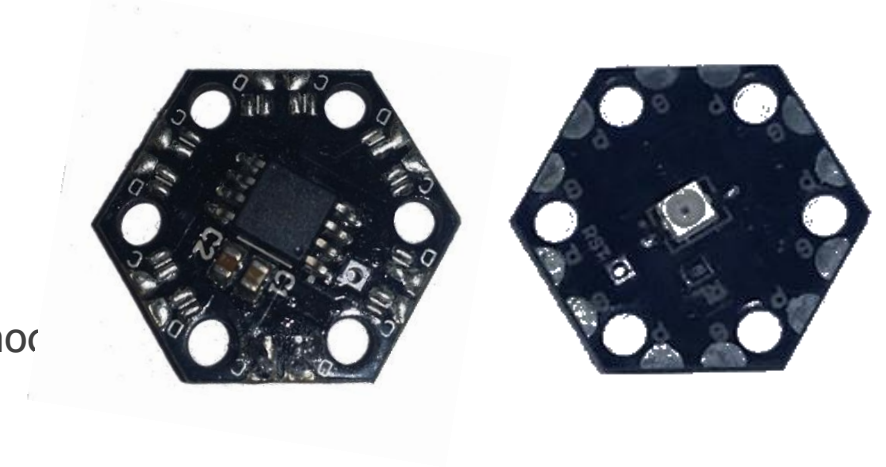


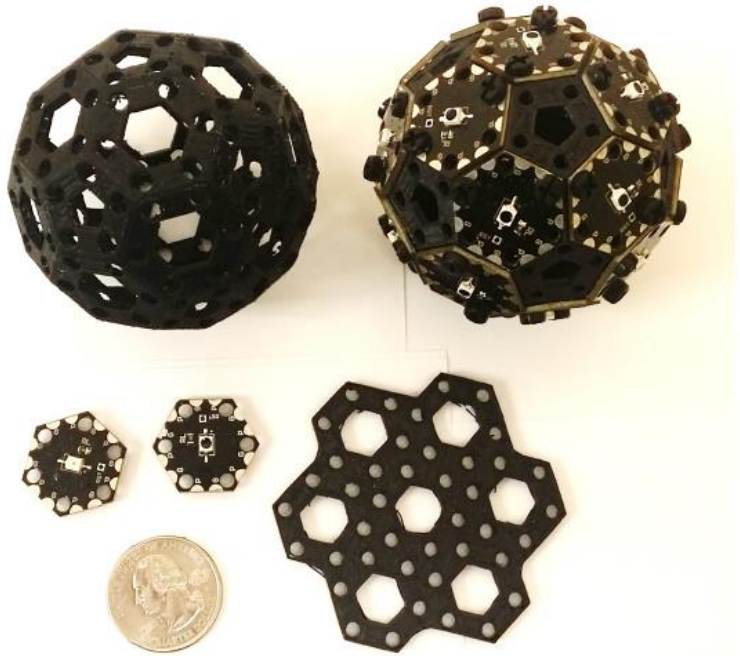
Terminal



Hardware Design - 1st Generation

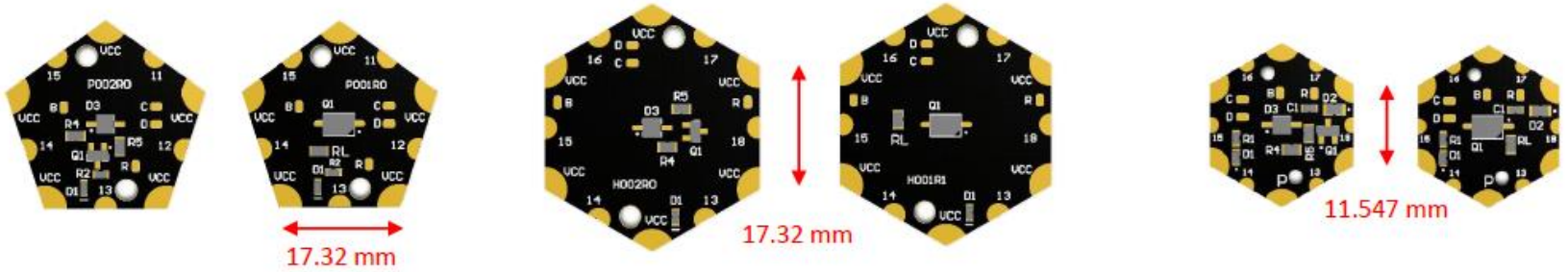
- Hexagonal receivers
- 20mm hexagons
- Height: 1.7mm PCB, 5.3mm max
- Weight: 0.03 ounce = 0.85 gram (per mod
- Atmel 8-bit AVR Microcontroller
- Photodiode (**VEMT3700**):
  - Angle of half sensitivity: φ = ± 60°
  - $\lambda_{0.1}$ **(nm):** 450 to 1080
  - Peak wavelength response at **850nm**
- Photodiode (**VEMT3700F**):
  - Angle of half sensitivity: φ = ± 60°
  - $\lambda_{0.1}$ **(nm):** 870 to 1050
  - Peak wavelength response at **940nm**



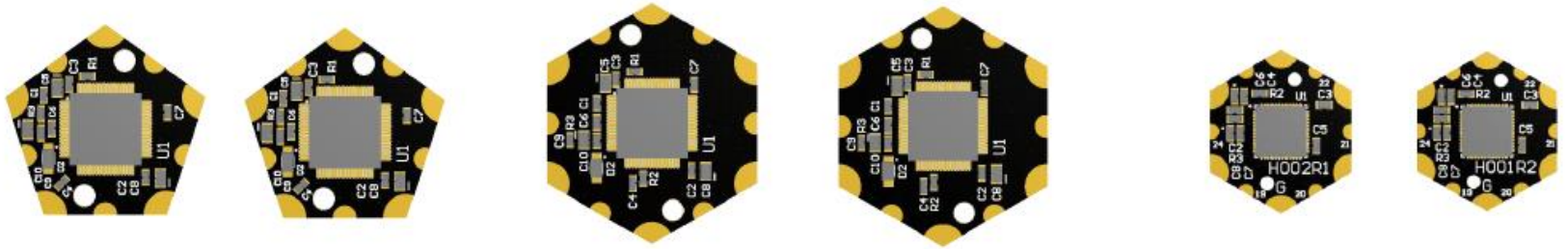


Hardware Design – 2nd Generation

PO02R0	PO01R0	HO02R0	HO01R1	HO02R1	HO01R2
Pentagon Transmitter	Pentagon Receiver	Hexagon Transmitter	Hexagon Receiver	Hexagon Transmitter	Hexagon Receiver



Top	Top	Top	Top	Top	Top

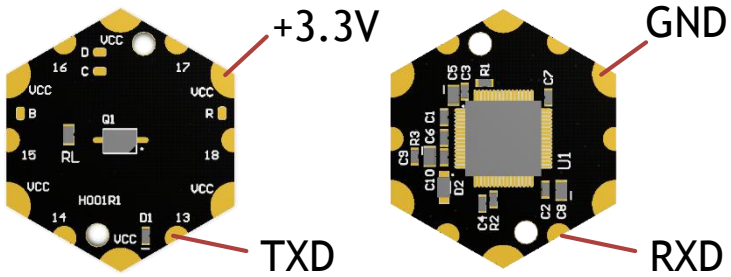


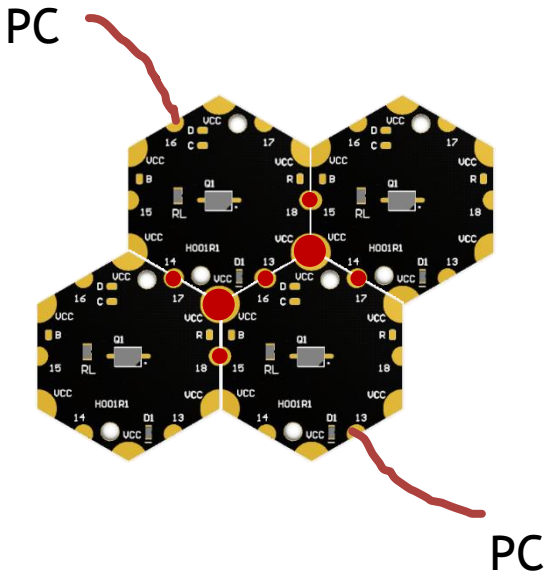
Bottom	Bottom	Bottom	Bottom	Bottom	Bottom

1st Generation vs. 2nd Generation

	1st Generation	2nd Generation
Architecture	I2C Bus	Peer-to-peer
Streaming speed	~100 bps	~ 1 Mbps
MCU	8-bit, 24 MHz	32-bit, 48 MHz, DMAs
ADC	10-bit	12-bit
Module height	20mm	20mm, 30mm
Module cost	~ $1	~ $4
Module weight	~ 0.85 g	~ 2 g
Assembly time		50-80% less
Firmware upgrade	1 option	3 options
Extra features		- Indictor LED - Boot and SWD pins

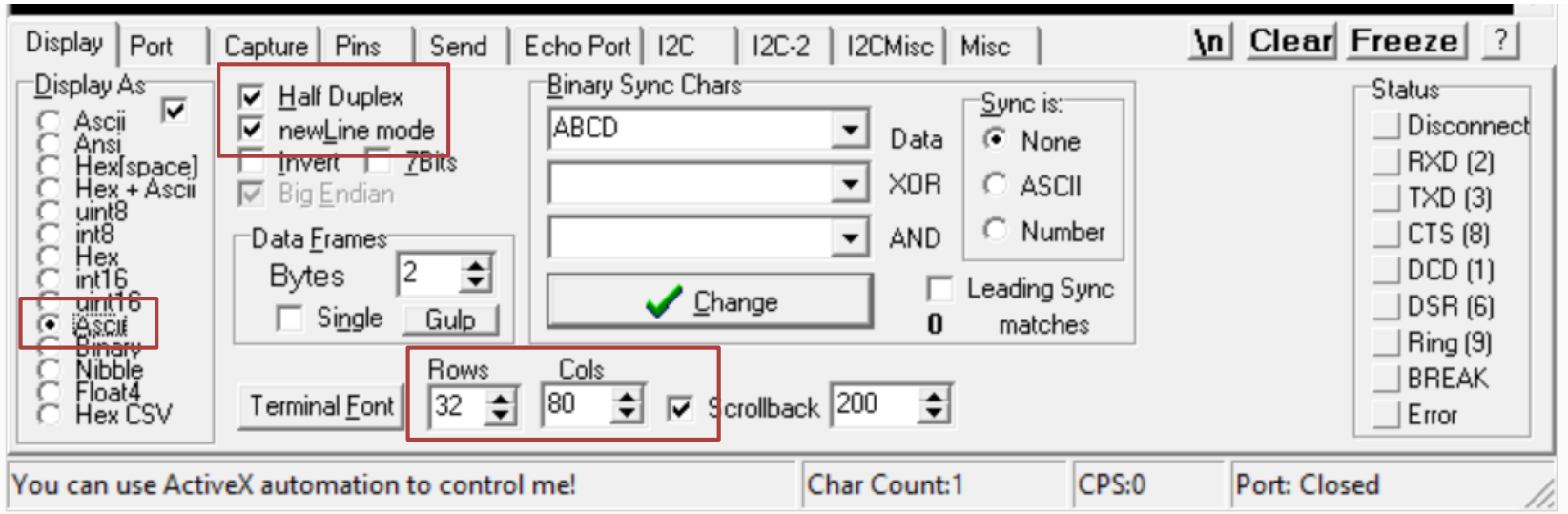
Setup & Connection

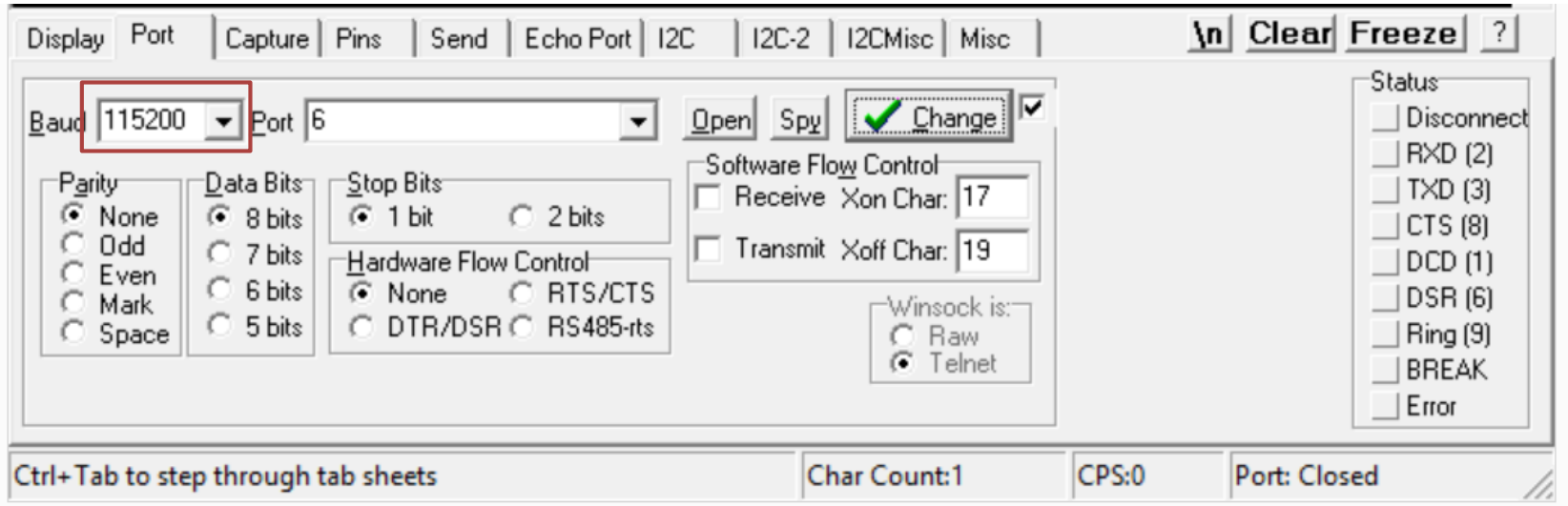




- Use any serial terminal software (RealTerm).
- 115200 baudrate for command line parser (CLP), 921600 for data streaming.
- ASCII display mode.
- Might need to check ‘echo’ commands.

Setup & Connection





Command Line Parser

- You can use any port and any module. Just connect to your serial terminal software and hit ‘Enter’.
- `help` display available commands.
- Modules are given unique IDs in the form #xxx.
- Each module can be named with an alias. You can use the alias wherever the module ID is used.
- Most commands can be targeted to: `me` (the module itself) - ID - alias - `all`
- Consult the ‘Module Firmware Description’ document for more details.

Command Line Parser

- Array management:
  - `name #xxx CoolName` — Name a module with an alias.
  - `ping #xxx` — Ping a module.
  - `update #XXX` — Update firmware.
- Receivers:
  - `sample #XXX` — Read one sample
  - `read #XXX rate file/port [--noled]`

    Read continuously from a module to any port at a given rate.
  - `set pdmode #XXX digital/analog` — Set PD data mode.
  - `set onelevel/zerolevel #XXX level` — Set PD discretization level.

Command Line Parser

- Transmitters:
  - `on/off/toggle #XXX` — Send one signal.
  - `write #XXX rate file/port [--noled]`

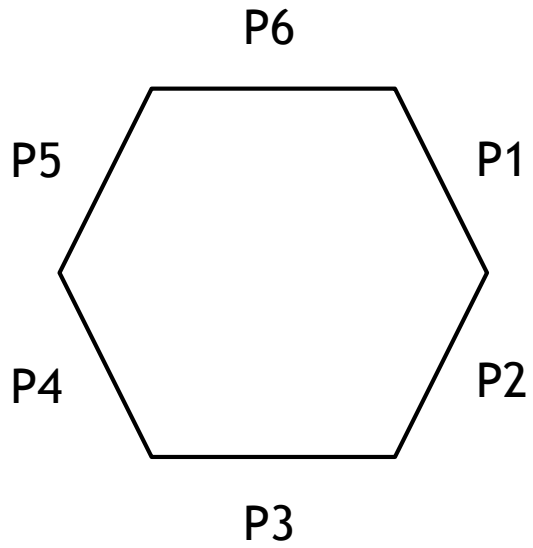
    Write continuously from any port to a module at a given rate.
  - `pulse #XXX width` — Send one pulse.
- Both
  - `stream #XXX` — Stream in/out a module.
  - `stop/pause #XXX` — Control reads/writes/streams.
  - `link #RXX #TXX #TXX` — Link modules in/out.

Data Streaming

- Two backbone communication modes:
  - Routing (Single-cast, multicast and broadcast).
    - Does not block ports. It transmits and routes data via short packets.
    - Not very reliable. Speed is usually in Kbps because of message processing delay.
    - Broadcast commands employ random backoff to avoid collision.
  - Streaming (SISO and SIMO).
    - Dedicated DMA streams with very high speeds (~1-2 Mbps).
    - Ports are blocked and must be restored using other ports.
    - Very reliable but speed is not controllable and it is difficult to process data.

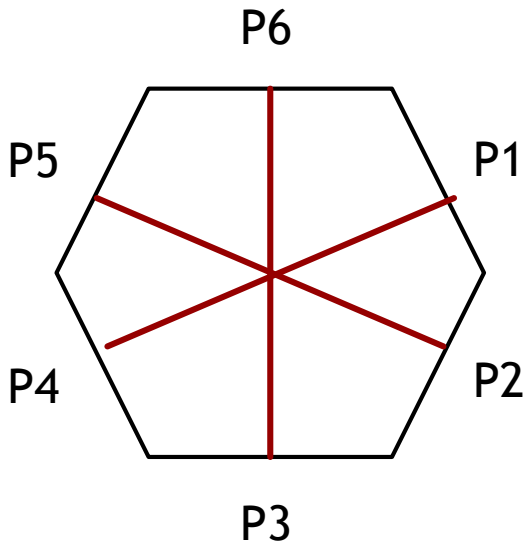
Data Streaming

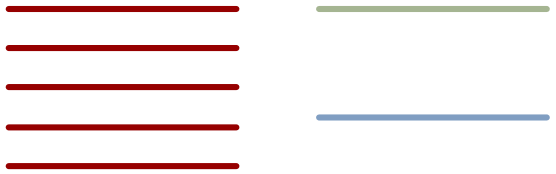
- How to build streams?
  - Each module has 7 available DMA channels (for now):
    - 5 for port-to-port streams.
    - 1 for port-to-front-end and vice versa streams.
    - 1 for memory-to-front-end and vice versa streams.



Data Streaming

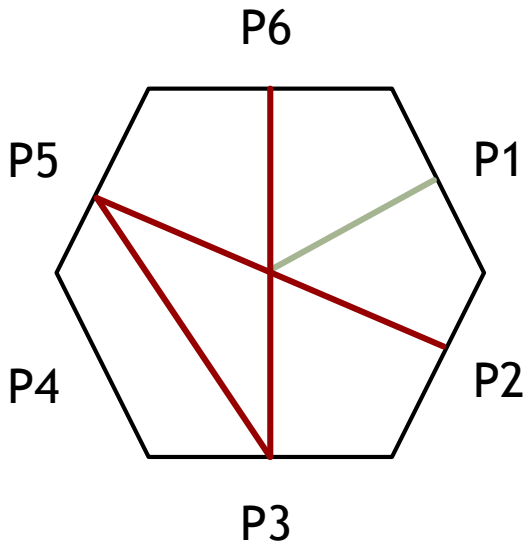
- How to build streams?
  - Each module has 7 available DMA channels (for now):
    - 5 for port-to-port streams.
    - 1 for port-to-front-end and vice versa streams.
    - 1 for memory-to-front-end and vice versa streams.

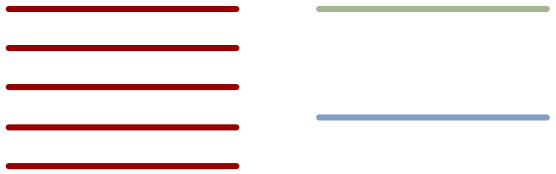




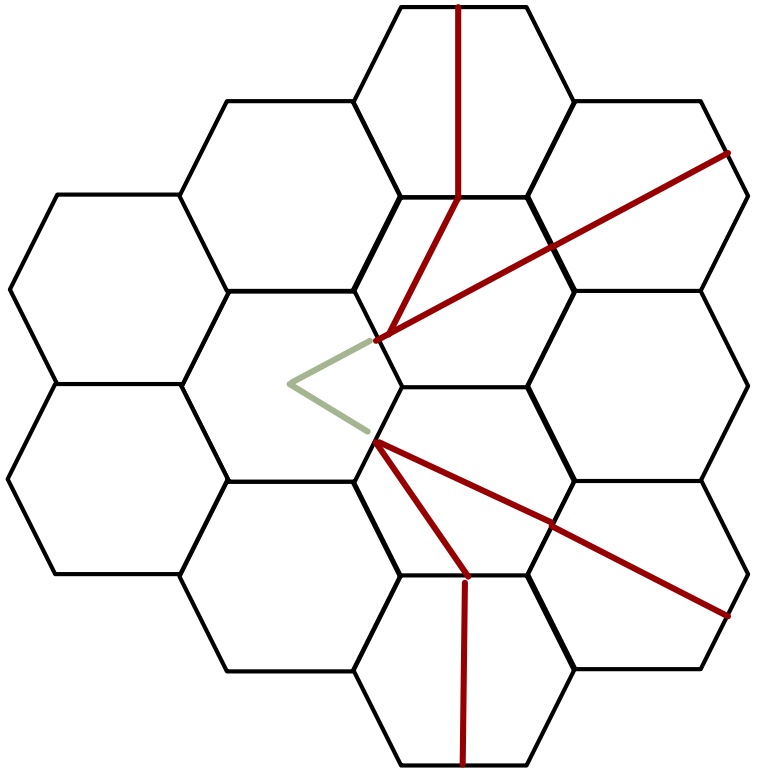
Data Streaming

- How to build streams?
  - Each module has 7 available DMA channels (for now):
    - 5 for port-to-port streams.
    - 1 for port-to-front-end and vice versa streams.
    - 1 for memory-to-front-end and vice versa streams.

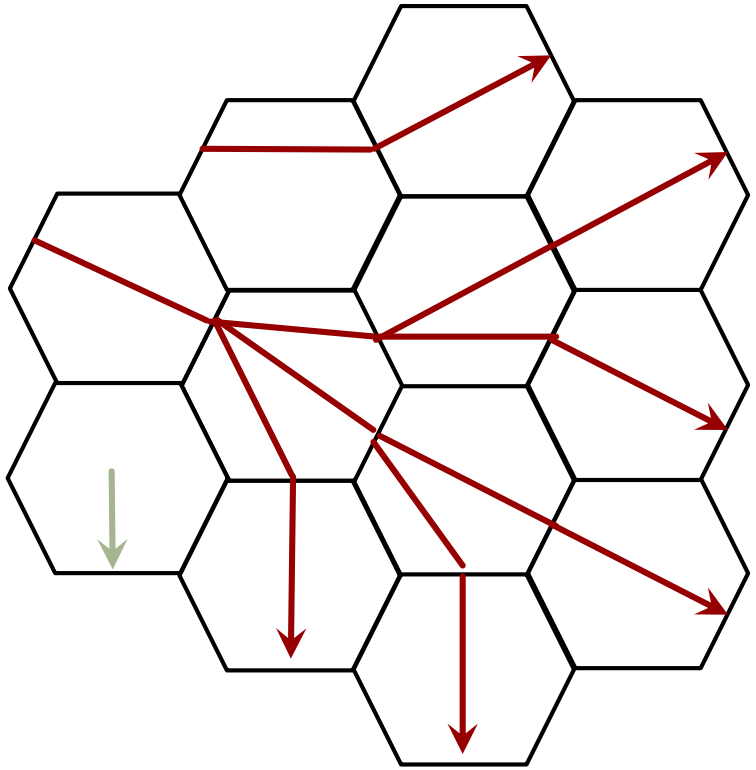




Data Streaming



Data Streaming

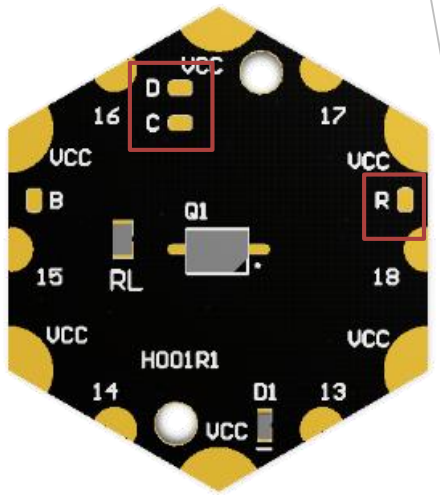


Firmware Update

- Three options:
  - Use SWD pins to individually program/debug a module.
  - Use bootloader to individually program a module.
  - Use bootloader to program the entire array.

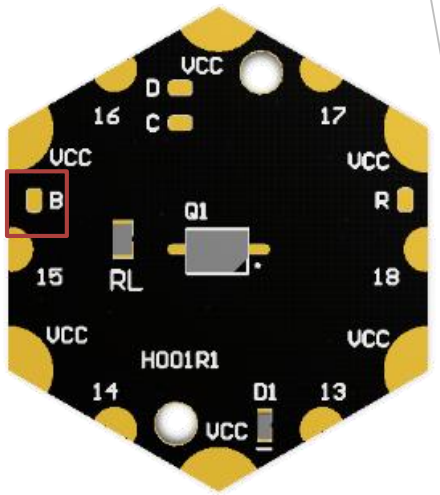
Firmware Update

- Three options:
  - Use SWD pins to individually program/debug a module.
  - Use bootloader to individually program a module.
  - Use bootloader to program the entire array.
    - You need an external programmer/debugger.



Firmware Update

- Three options:
  - Use SWD pins to individually program/debug a module.
  - Use bootloader to individually program a module.
  - Use bootloader to program the entire array.
    - You need to put the module in a bootloader mode:
      - Apply +3.3V to the boot pin and reset.
      - Or, use the firmware update command `update`
      - When modules arrive from factory, they will be already in bootloader mode.
    - Use ‘ST Flash Loader Demonstrator’ tool.
    - IMPORTANT: It only works using ‘P1’ (13) port (P2 in the pentagons).



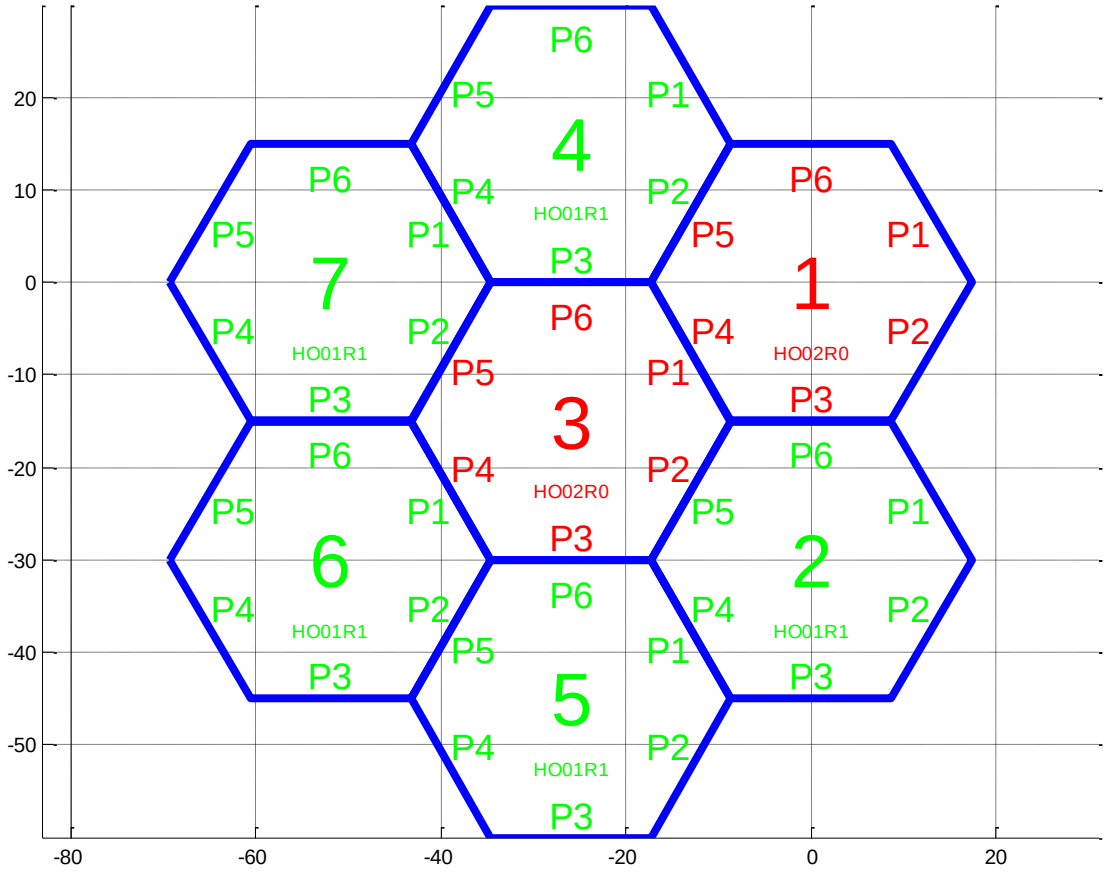
Useful Features

- `CTRL+z` Cycle through the last five commands.
- `route #xxx #xxx` Calculate the shortest route between two modules using Dijkstra's algorithm (source, destination).
- `Group name #xxx #xxx` Group multiple modules.
- `Reset #xxx` Reset a module.
- `info` Useful information about the module and the array plus compile time, date and firmware version.
- All module IDs and aliases are checked against a database.
- Some errors are given for wrong parameters, wrong syntax or wrong commands.

Useful Tools: Tandy Supercomputer Simulator

```
===BEGINNING SIMULATION===
board 1 out port 2 - sending: %HI
board 1 out port 1 - sending: %HI
board 1 received on 2 - %HI
board 1 received on 1 - %HI
board 1 out port 1 - sending: %HI#00000
board 1 out port 2 - sending: %HI#00000
board 1 out port 2 - sending: %ST#00000#00000#00000#65535#00000#00000#00000%
board 1 received on 2 - %ID#00000
board 2 out port 3 - sending: %HI
board 2 out port 4 - sending: %HI
board 2 received on 3 - %HI
board 2 received on 4 - %HI
board 2 out port 3 - sending: %HI#00000
board 2 out port 4 - sending: %HI#00000
board 2 out port 3 - sending: %ST#00000#00000#00000#00000#65535#65535#00000%
board 2 received on 4 - %HI#00000
board 2 received on 3 - %ID#00000
board 0 out port 0 - sending: %HI
board 0 out port 5 - sending: %HI
board 0 received on 0 - %HI
board 0 received on 5 - %HI
board 0 out port 0 - sending: %ID#00000
board 0 received on 0 - %HI#00000
board 0 out port 5 - sending: %ID#00000
board 0 received on 0 - %ST#00000#00000#00000#00000#65535#65535#00000%
board 0 received on 5 - %HI#00000
board 0 received on 5 - %ST#00000#00000#00000#65535#00000#00000#00000%
```

Useful Tools: Automatic Topology Generator



topology.h | clp.c | stm32f0xx_hal_uart.c | stm32f0xx_hal_gpio.c | startup_stm32f091xc.s | ports.c | clp.h | dma.c | stm32f0xx_it.c

```
25 // HO02R0: Hexagon transmiter with VSMY2850G - 30mm height
26 // HO02R1: Hexagon transmiter with VSMY2850G - 20mm height
27 // PO01R0: Pentagon receiver with VEMT3700 compatible with hexagons 30mm height
28 // PO02R0: Pentagon transmiter with VSMY2850G compatible with hexagons 30mm height
29
30 // Enumerations
31 enum  modulePartNumbers{_HO01R1, _HO01R2, _HO02R0, _HO02R1, _PO01R0, _PO02R0};
32 enum  portPol{_normal, _reversed};
33 enum  modulePorts{_P1=1, _P2, _P3, _P4, _P5, _P6};
34
35 #define _N  8          // Number of array modules
36
37 // Array modules
38 #define _mod1   1<<3
39 #define _mod2   2<<3
40 #define _mod3   3<<3
41 #define _mod4   4<<3
42 #define _mod5   5<<3
43 #define _mod6   6<<3
44 #define _mod7   7<<3
45 #define _mod8   8<<3
46
47 // Topology
48 static uint16_t array[_N][7] = {
49 { _HO02R0,          0,          0,  _mod2|_P6,  _mod3|_P1,  _mod4|_P2,          0},   // Module 1
50 { _HO01R1,          0,          0,          0,  _mod5|_P1,  _mod3|_P2,  _mod1|_P3},   // Module 2
51 { _HO02R0,  _mod1|_P4,  _mod2|_P5,  _mod5|_P6,  _mod6|_P1,  _mod7|_P2,  _mod4|_P3},   // Module 3
52 { _HO01R1,          0,  _mod1|_P5,  _mod3|_P6,  _mod7|_P1,  _mod8|_P4,          0},   // Module 4
53 { _HO02R0,  _mod2|_P4,          0,          0,          0,  _mod6|_P2,  _mod3|_P3},   // Module 5
54 { _HO02R0,  _mod3|_P4,  _mod5|_P5,          0,          0,          0,  _mod7|_P3},   // Module 6
55 { _HO02R0,  _mod4|_P4,  _mod3|_P5,  _mod6|_P6,          0,          0,          0},   // Module 7
56 { _PO01R0,          0,          0,          0,  _mod4|_P5,          0,          0}    // Module 8
57 };
58
```

Target stopped. | ST-Link Debugger | L:6 C:81 | CAP NUM SCRL OVR R/W


Thank You