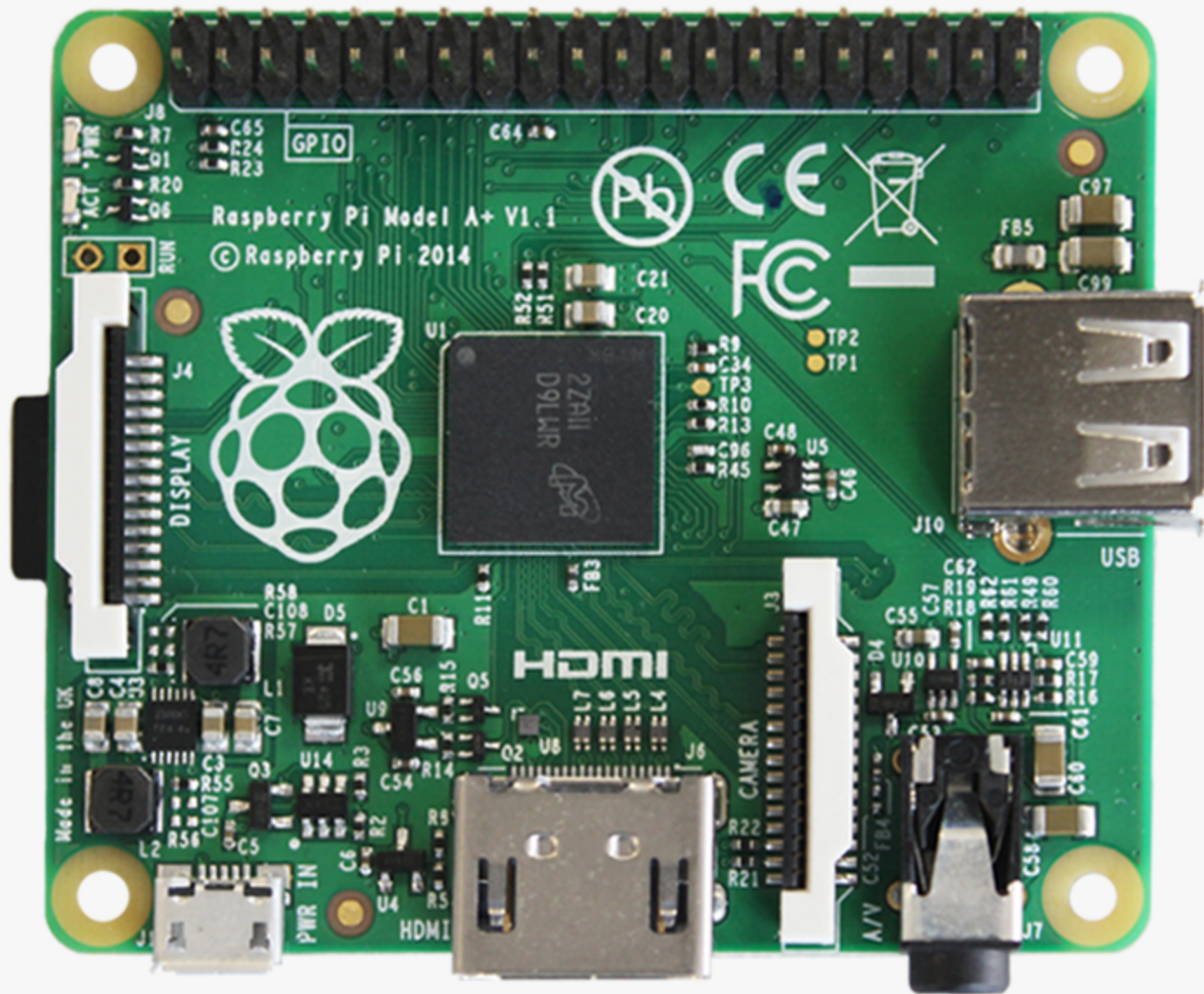




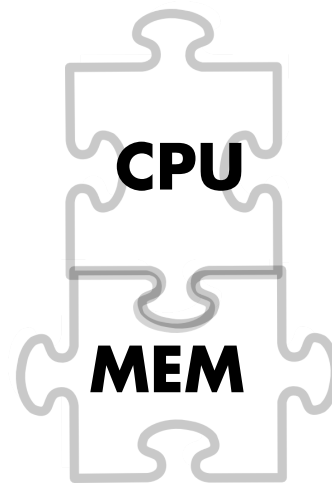
Goal I

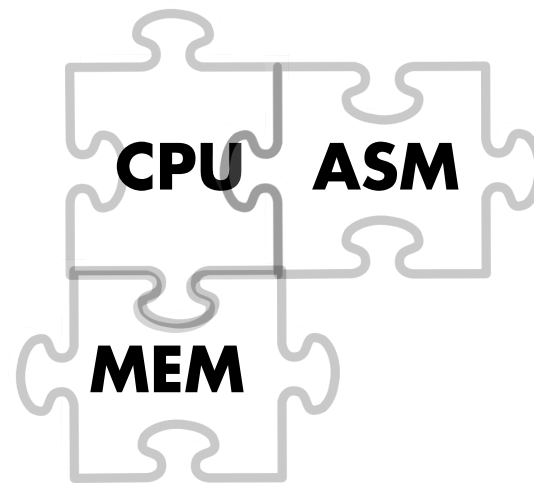
**Understand
How Computers Work**

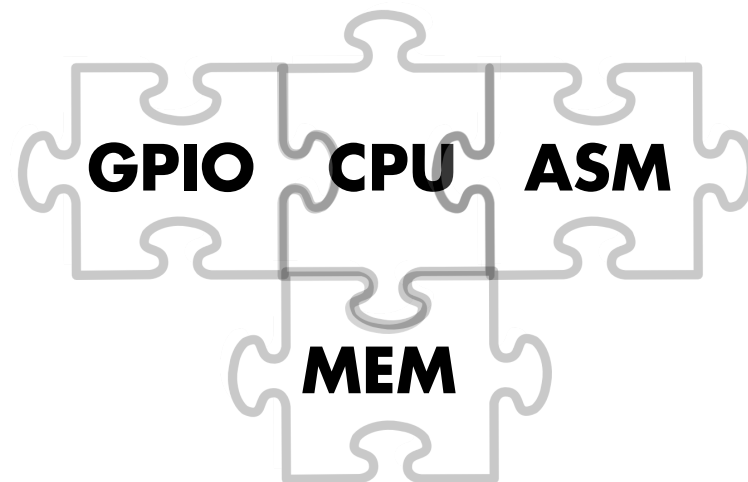
Computer Systems from the Ground Up

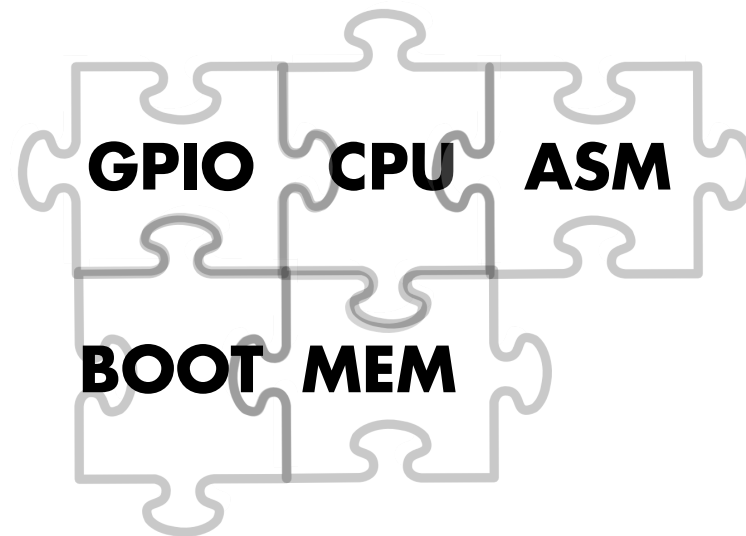


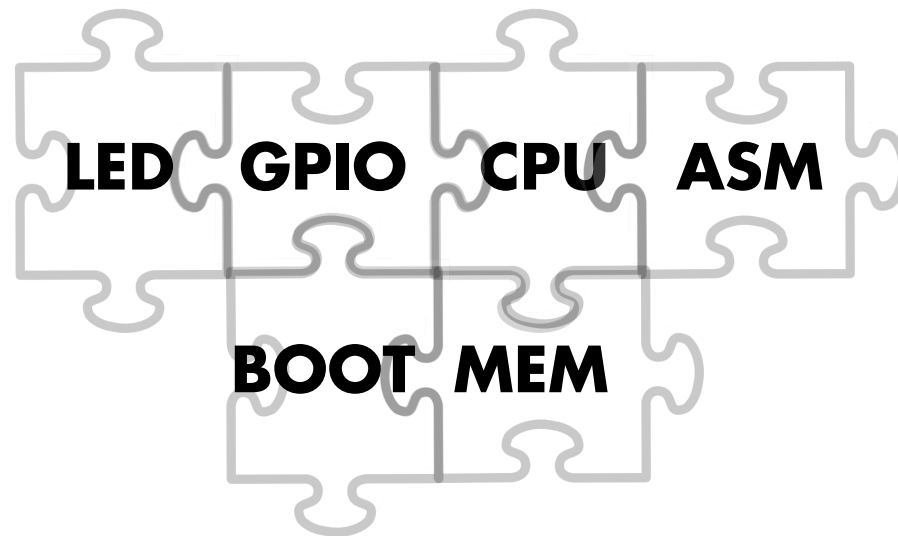
Bare Metal on the Raspberry Pi

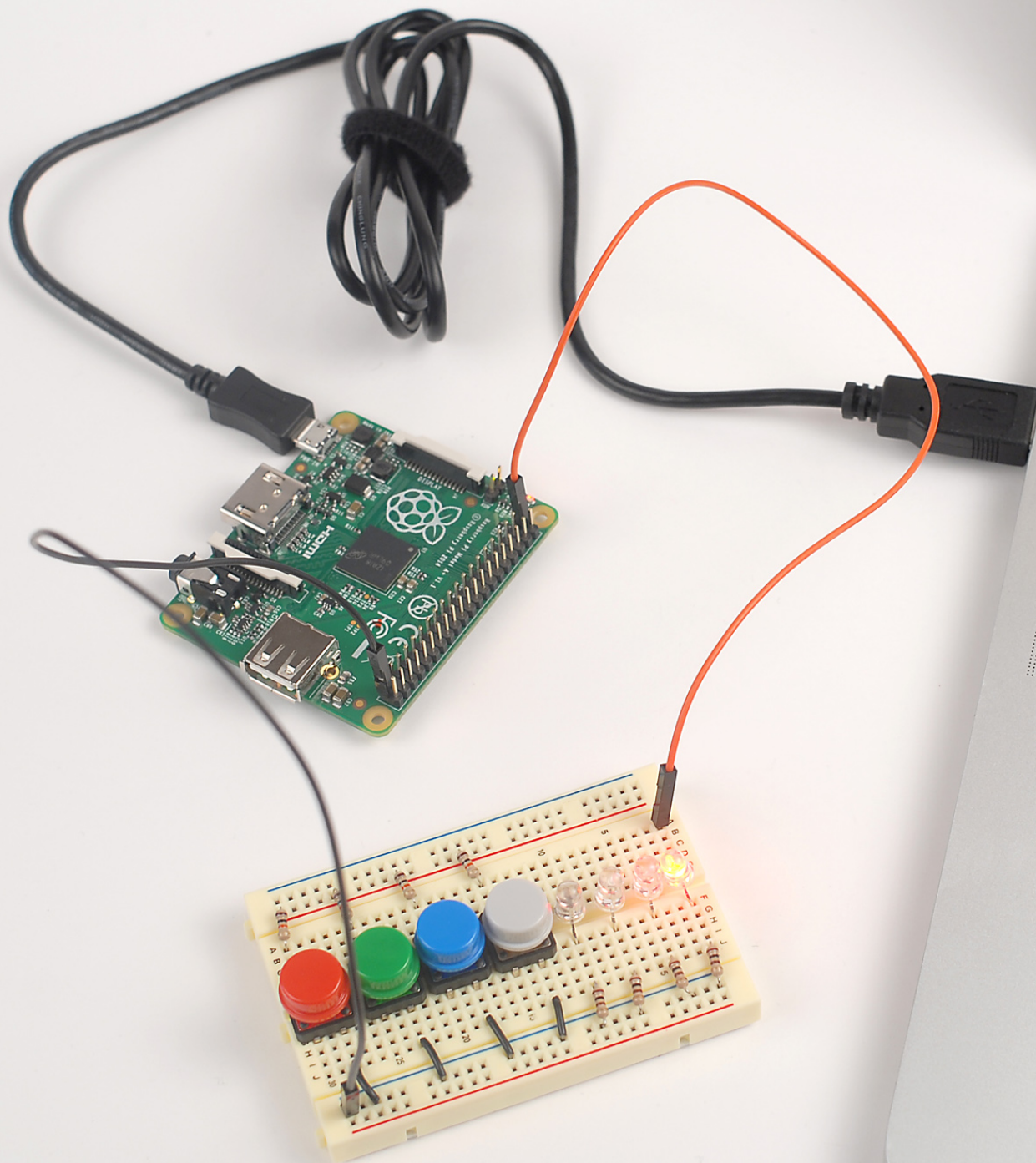


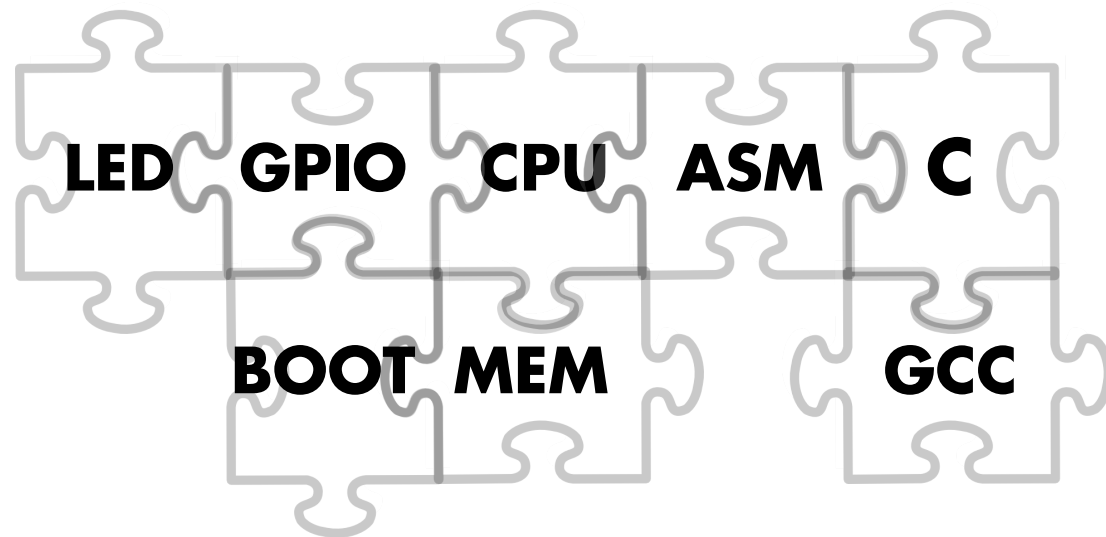


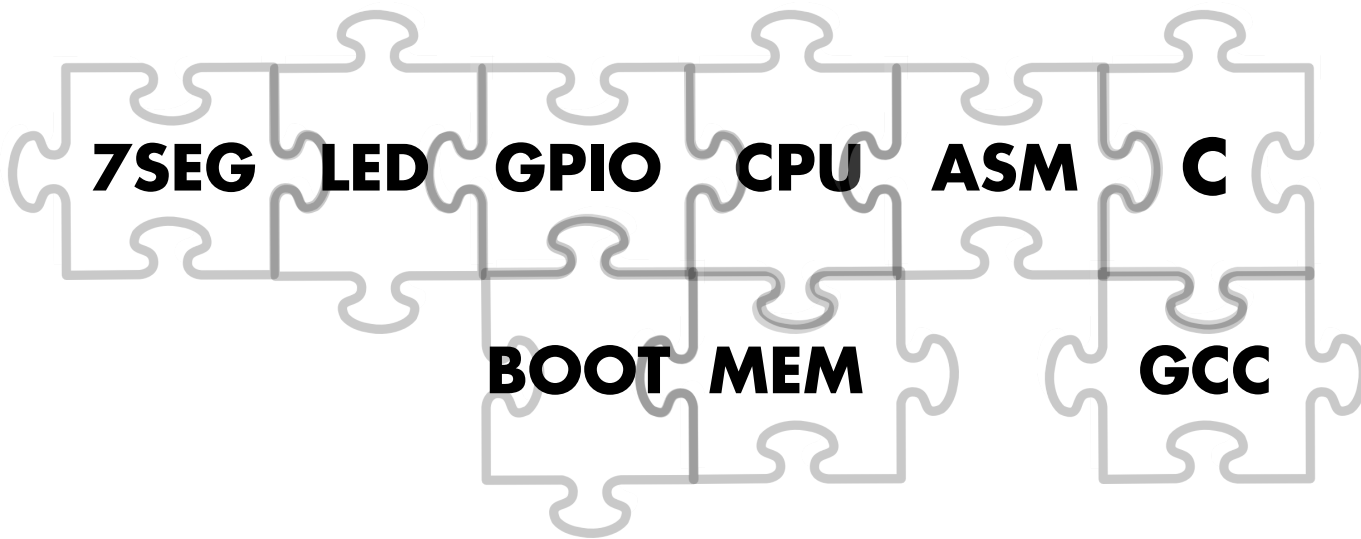


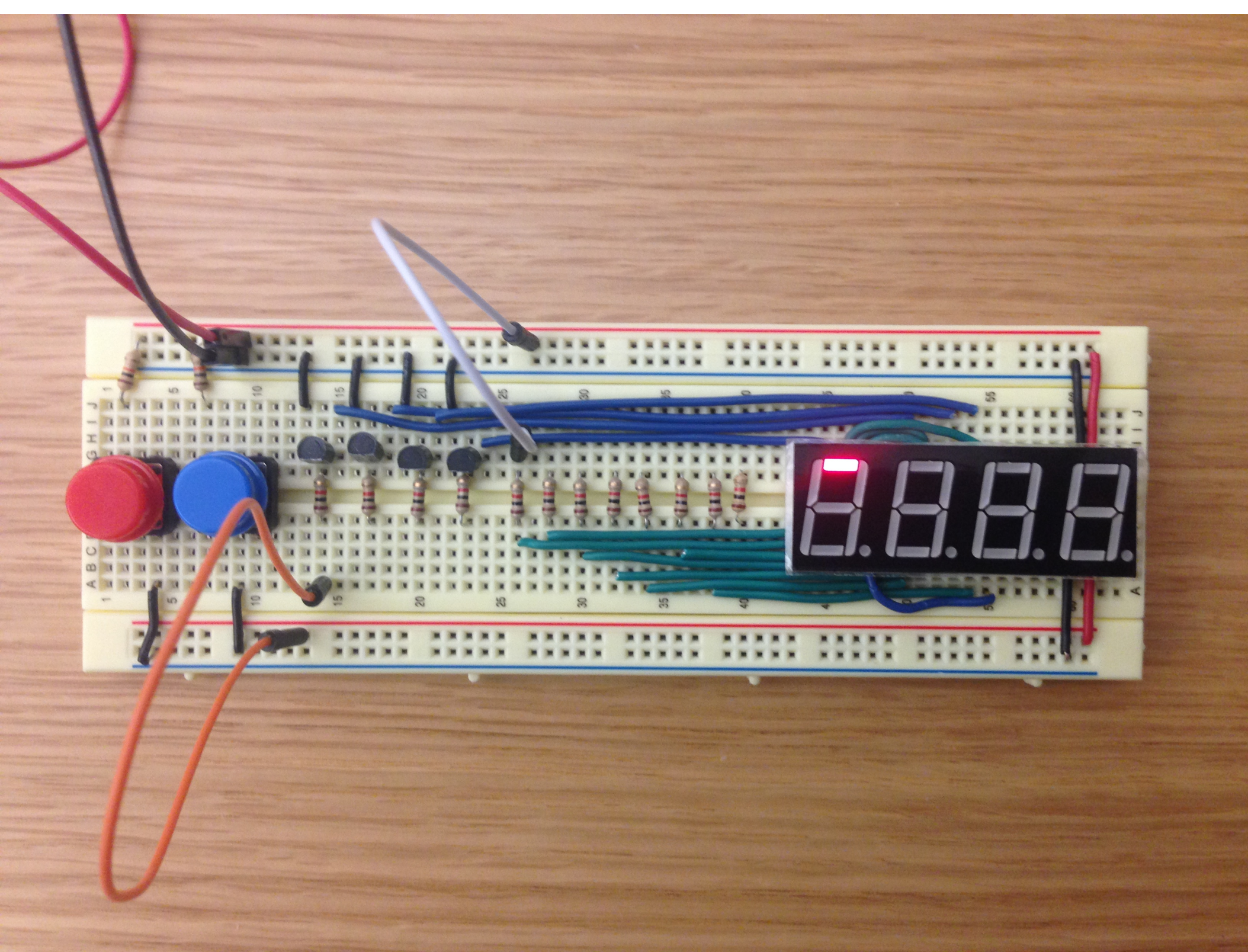






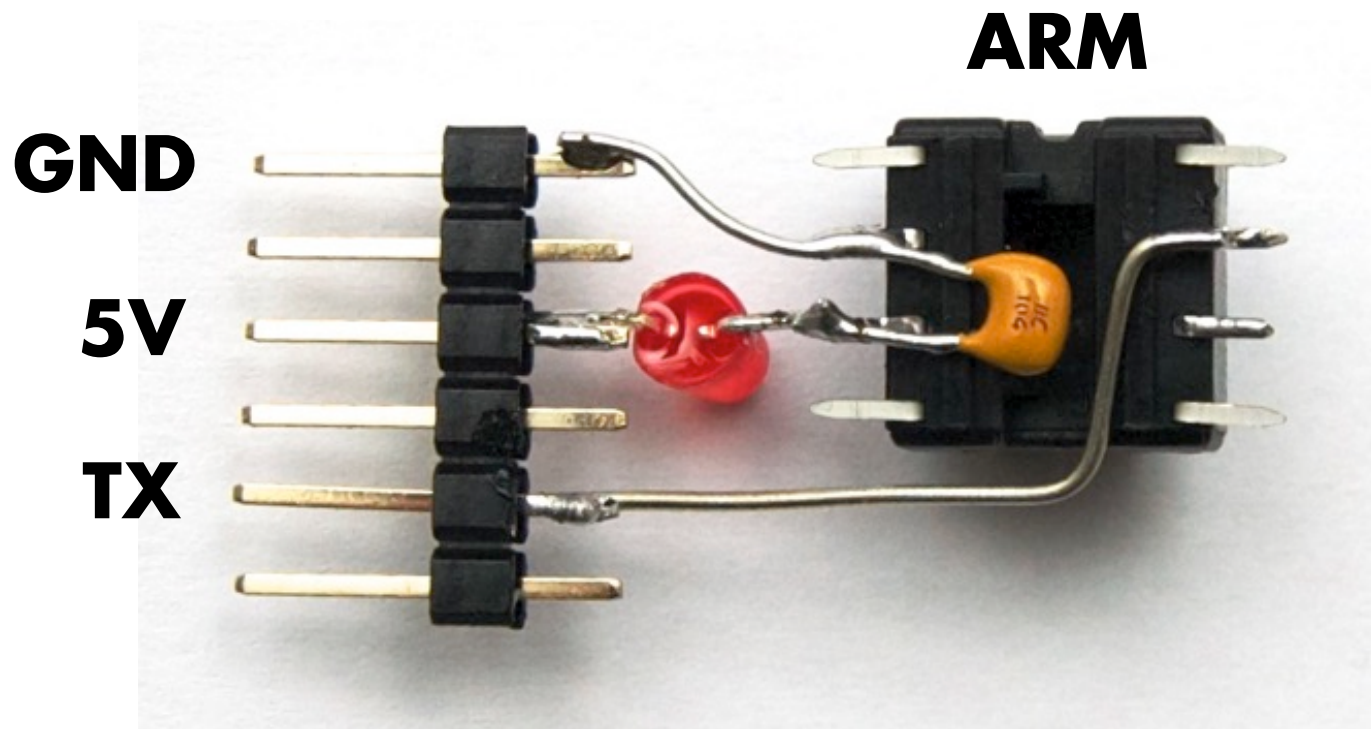






Simple and Clear
Functionality
is
Beautiful

Jean-Claude Wippler



www.jeelabs.org

```

/* match: search for regexp anywhere in text */
int match(char *regexp, char *text) {
    if (regexp[0] == '^')
        return matchhere(regexp+1, text);
    do { /* must look even if string is empty */
        if (matchhere(regexp, text))
            return 1;
    } while (*text++ != '\0');
    return 0;
}

/* matchhere: search for regexp at beginning of text */
int matchhere(char *regexp, char *text) {
    if (regexp[0] == '\0')
        return 1;
    if (regexp[1] == '*')
        return matchstar(regexp[0], regexp+2, text);
    if (regexp[0] == '$' && regexp[1] == '\0')
        return *text == '\0';
    if (*text != '\0' && (regexp[0] == '.' || regexp[0] == *text))
        return matchhere(regexp+1, text+1);
    return 0;
}

/* matchstar: search for c*regexp at beginning of text*/
int matchstar(int c, char *regexp, char *text) {
    do { /* a * matches zero or more instances */
        if (matchhere(regexp, text))
            return 1;
    } while (*text != '\0' && (*text++ == c || c == '.'));
    return 0;
}

```

Ken Thompson

Regular Expression

ch

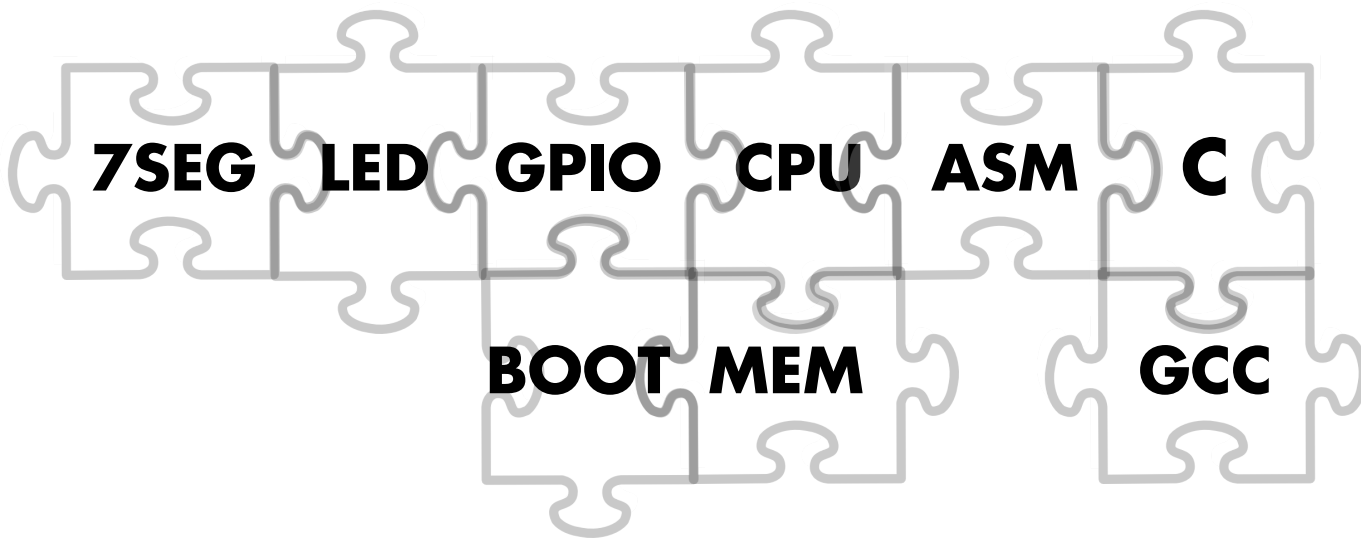
.

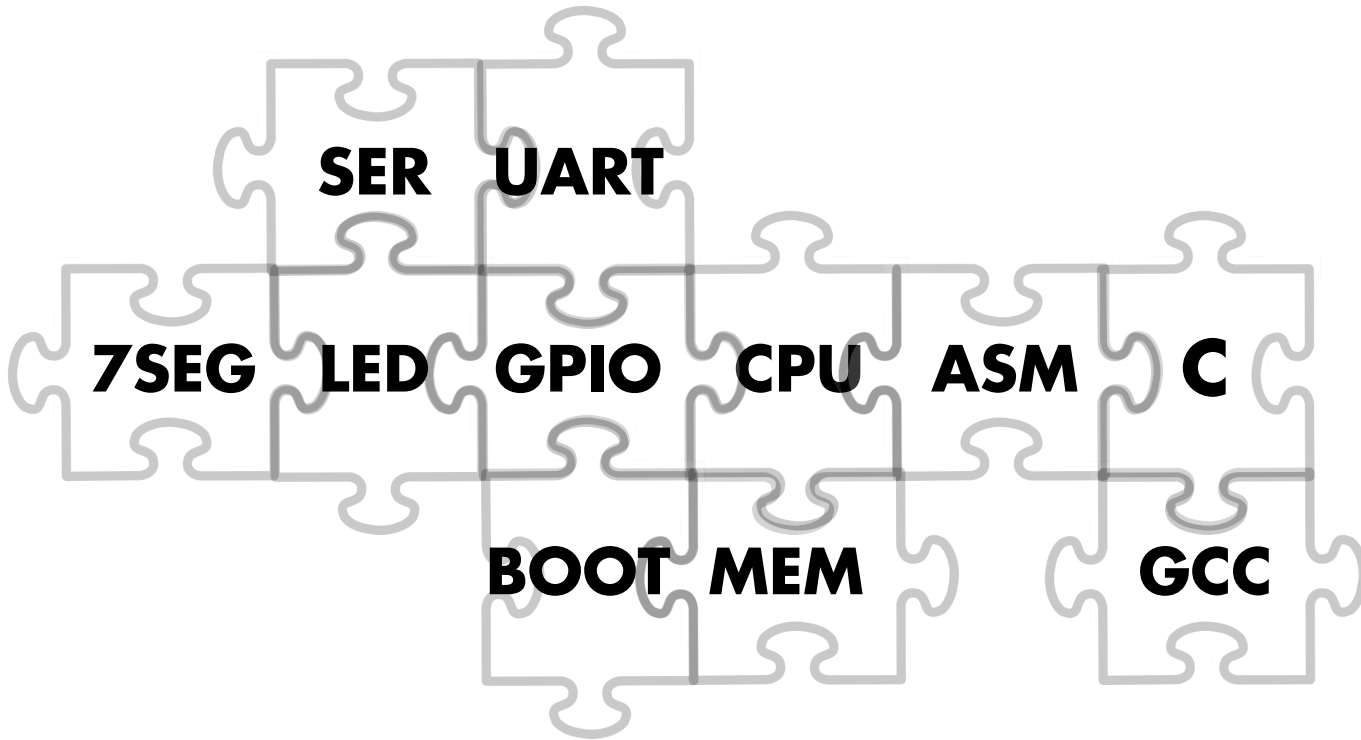
*

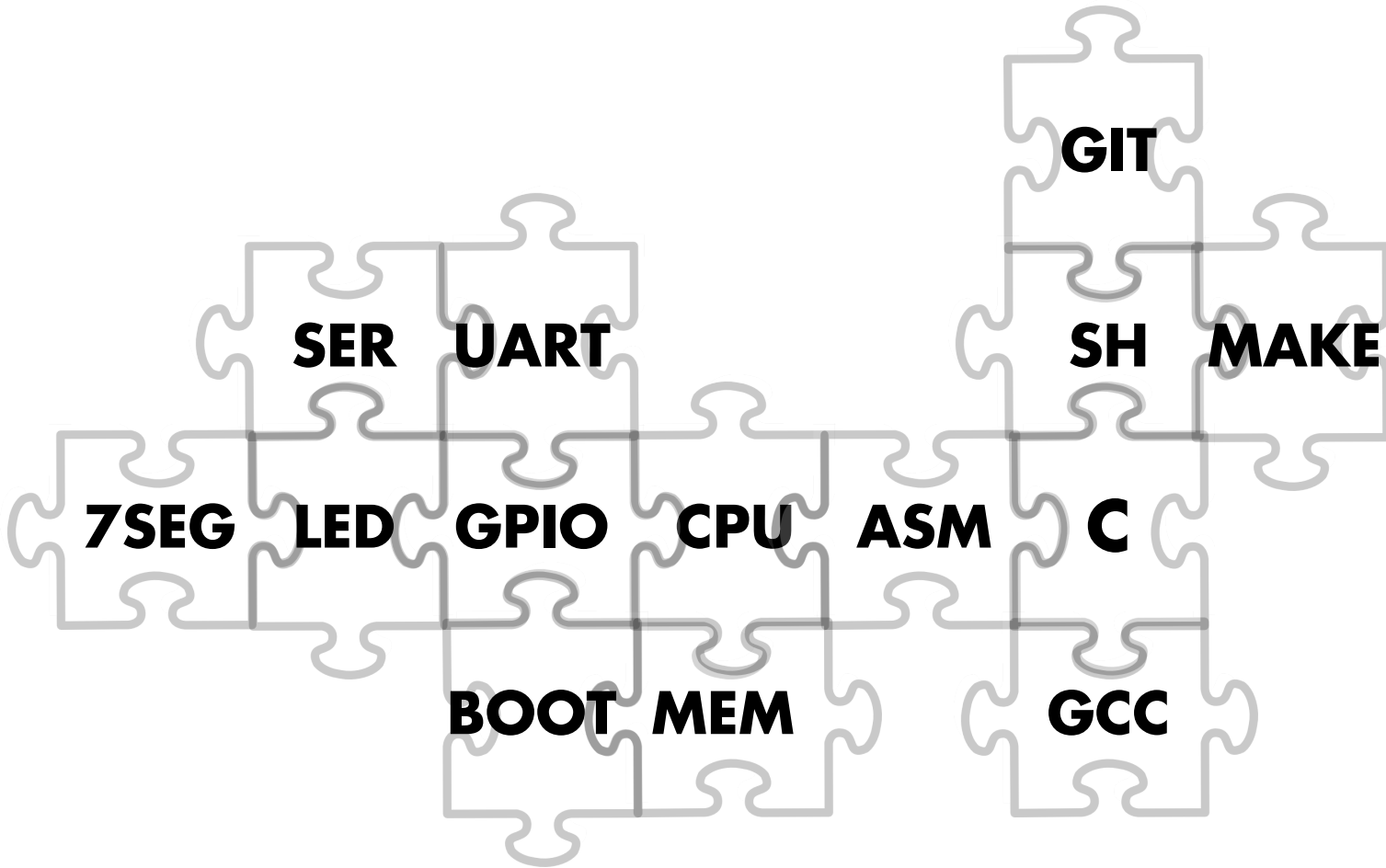
^

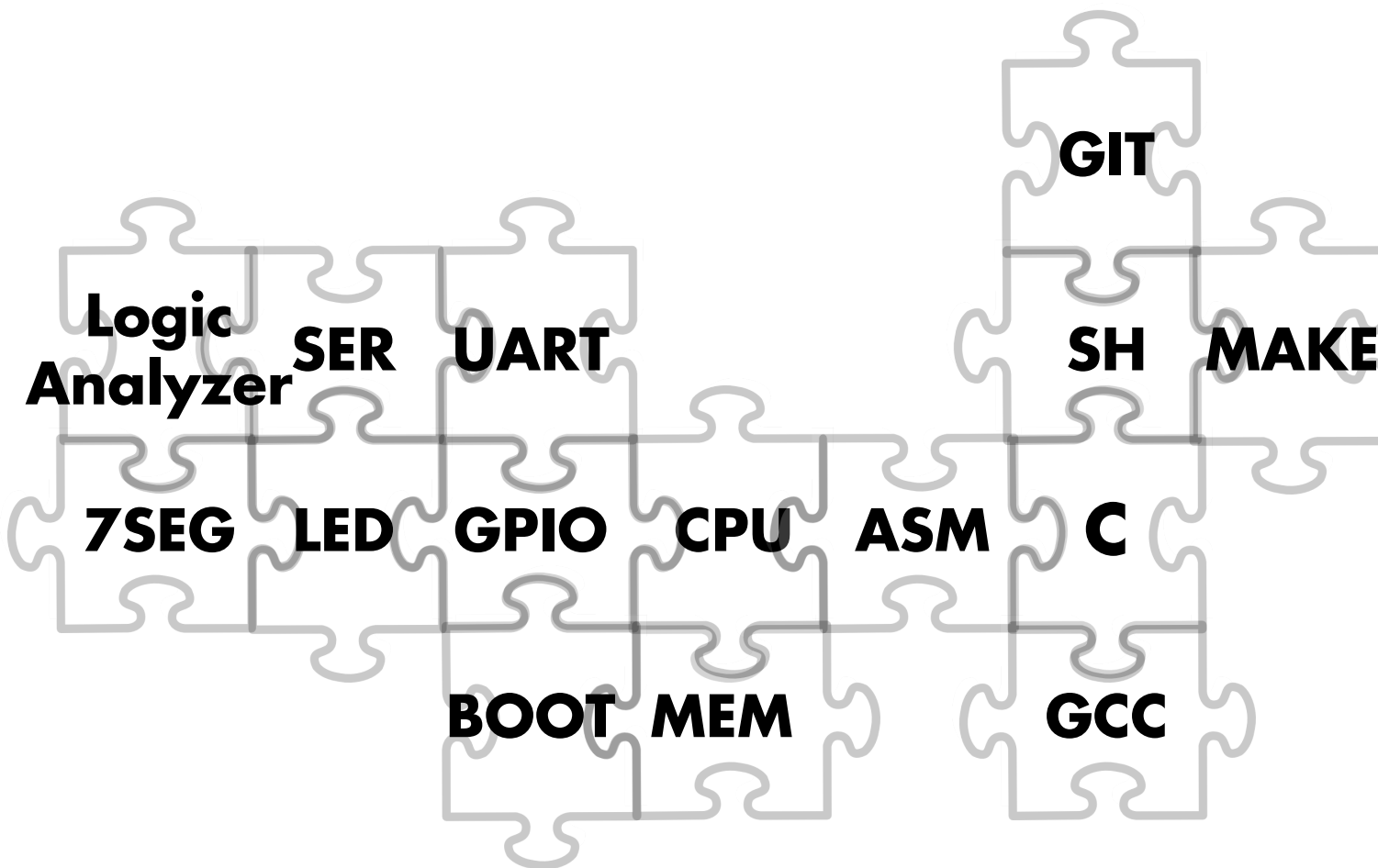
\$

Read Beautiful Code









**Logic
Analyzer**

SER

UART

GIT

SH

MAKE

7SEG

LED

GPIO

CPU

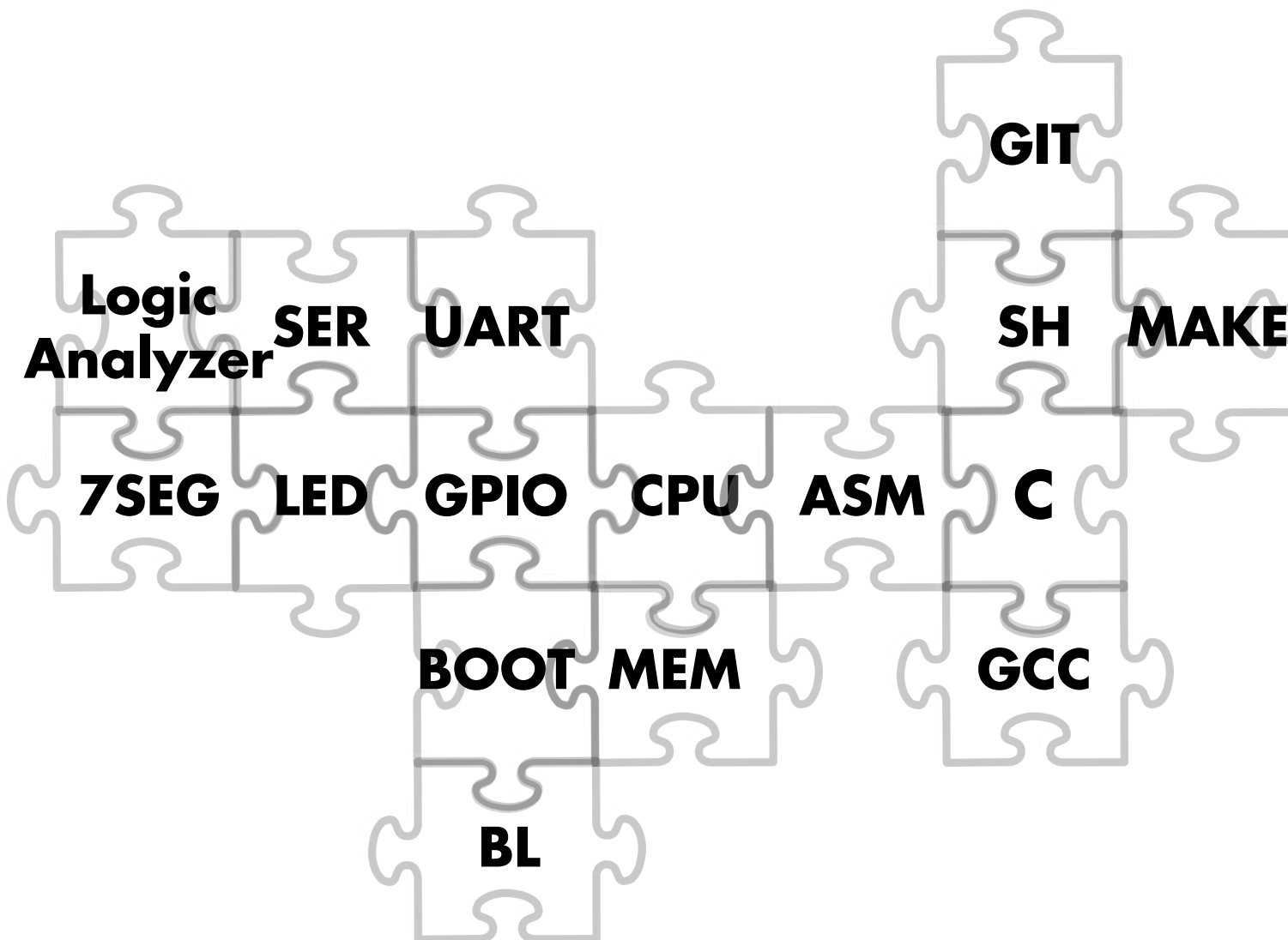
ASM

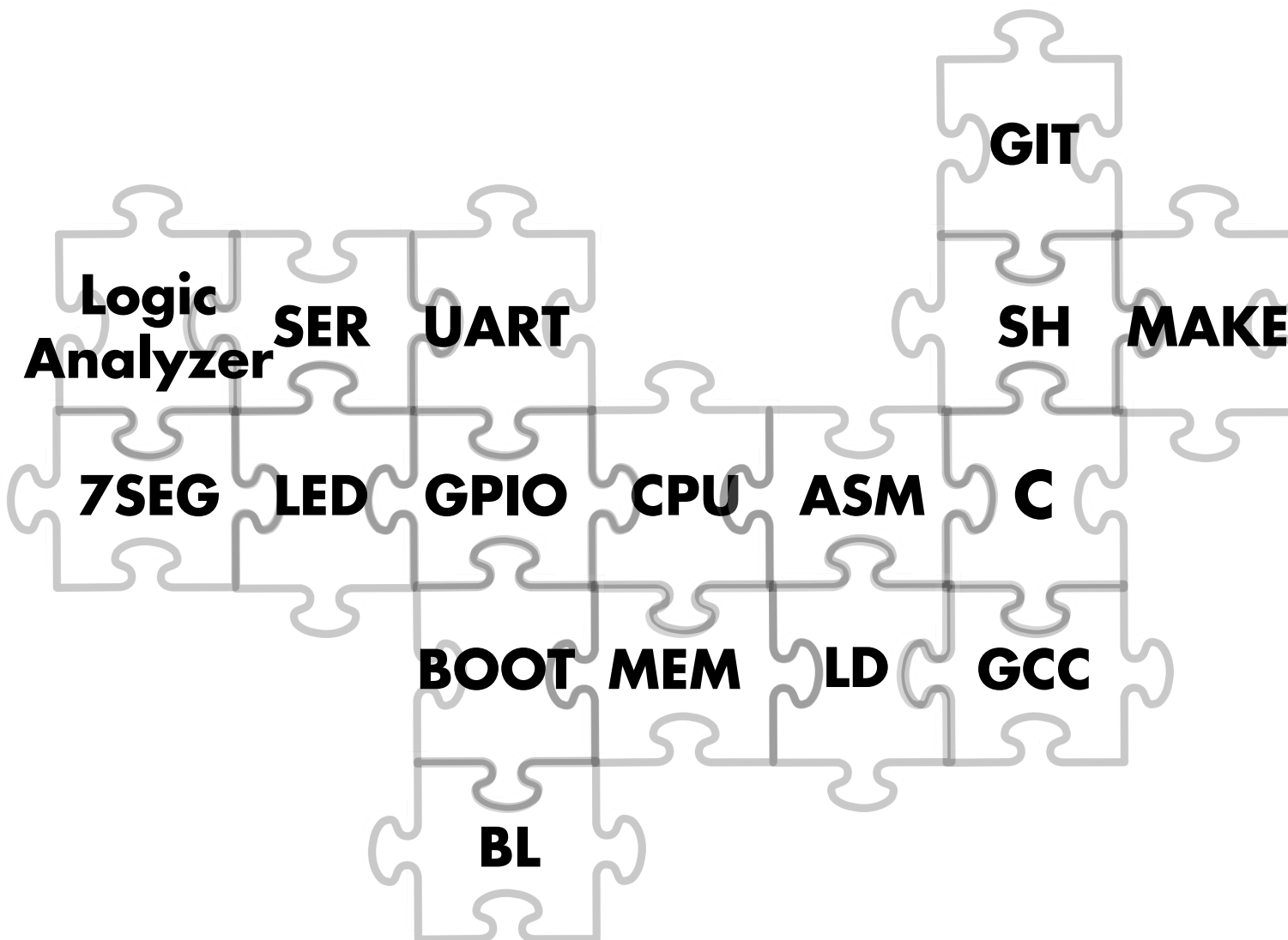
C

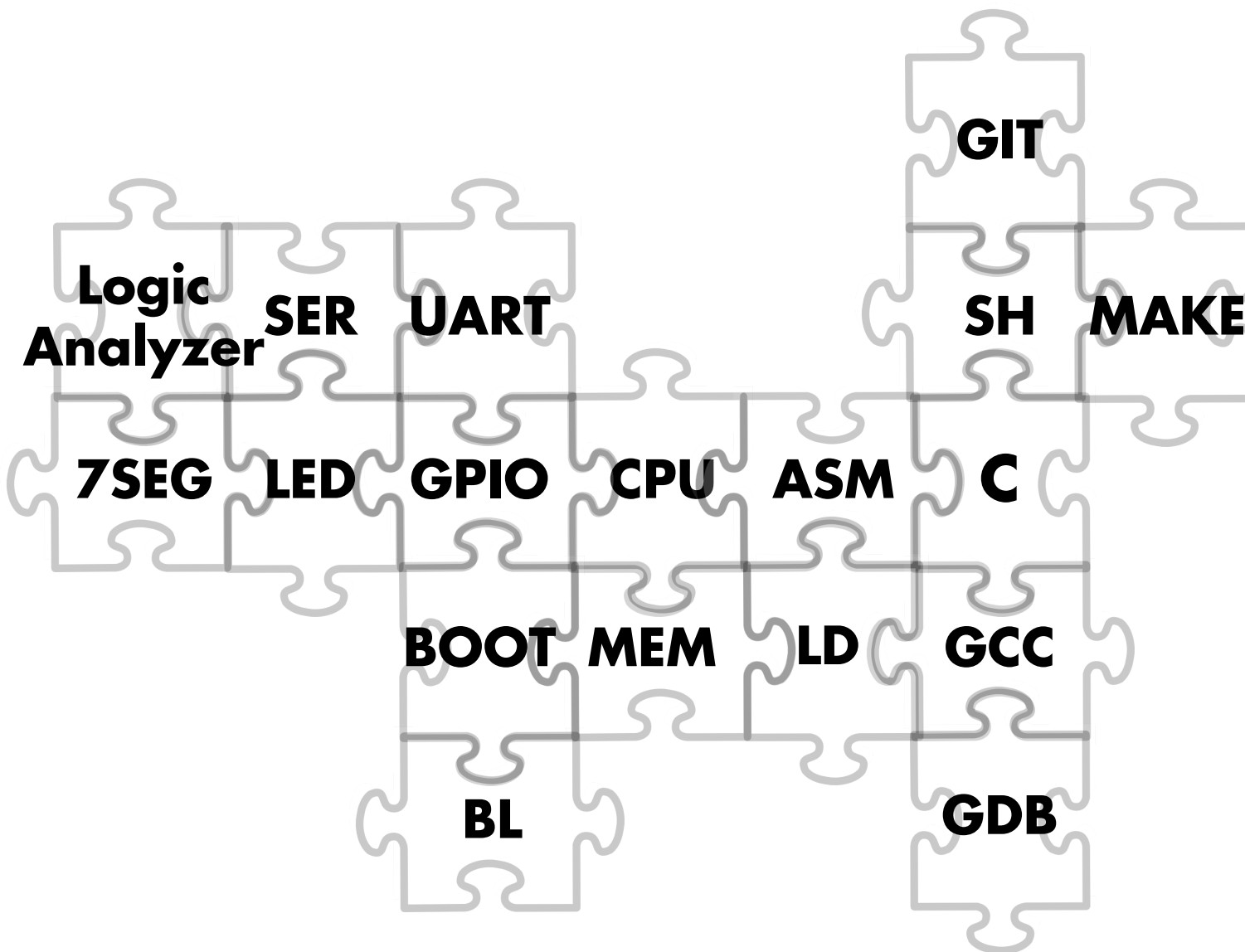
BOOT

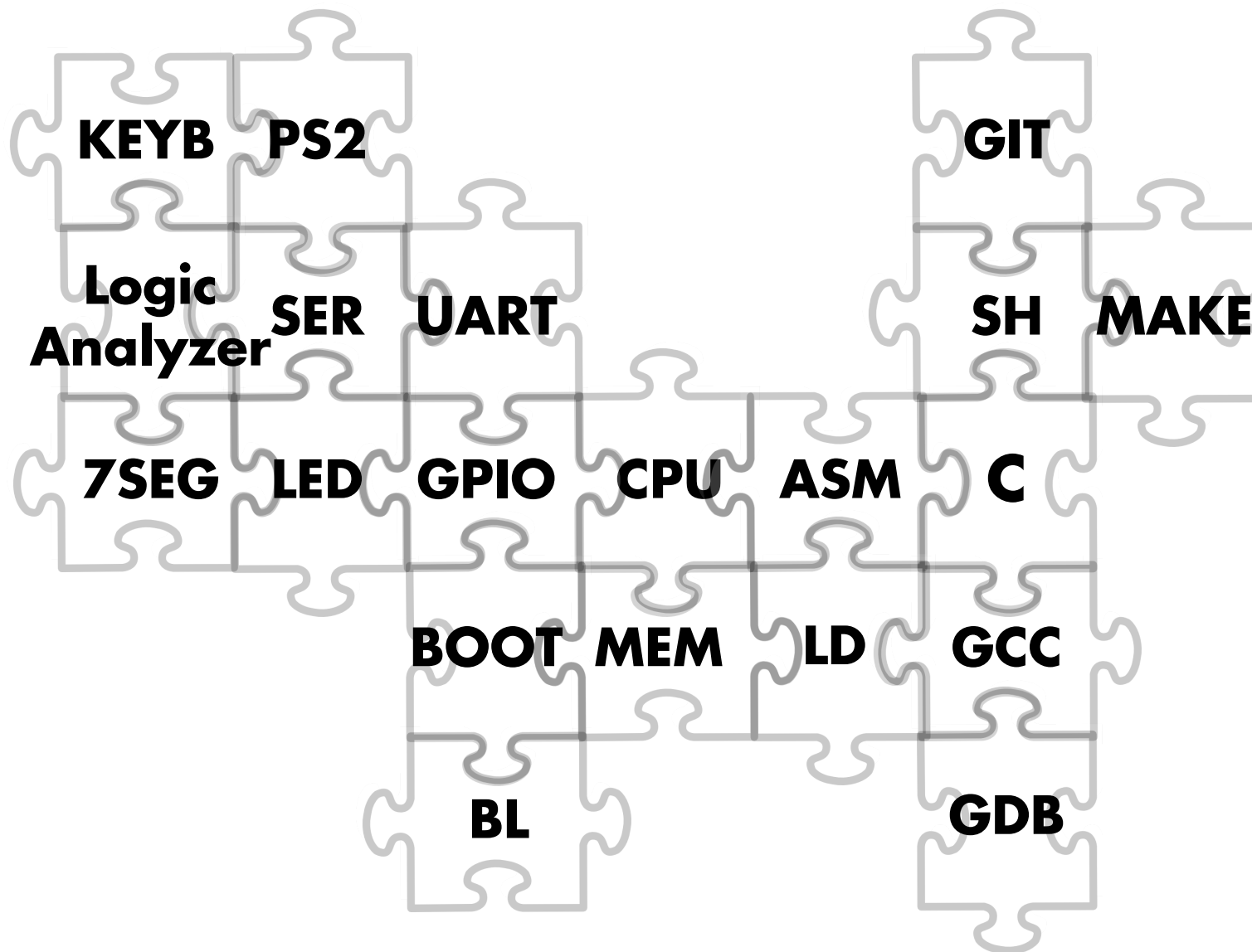
MEM

GCC









KEYB

PS2

GIT

**Logic
Analyzer**

SER

UART

SH

MAKE

7SEG

LED

GPIO

CPU

ASM

C

BOOT

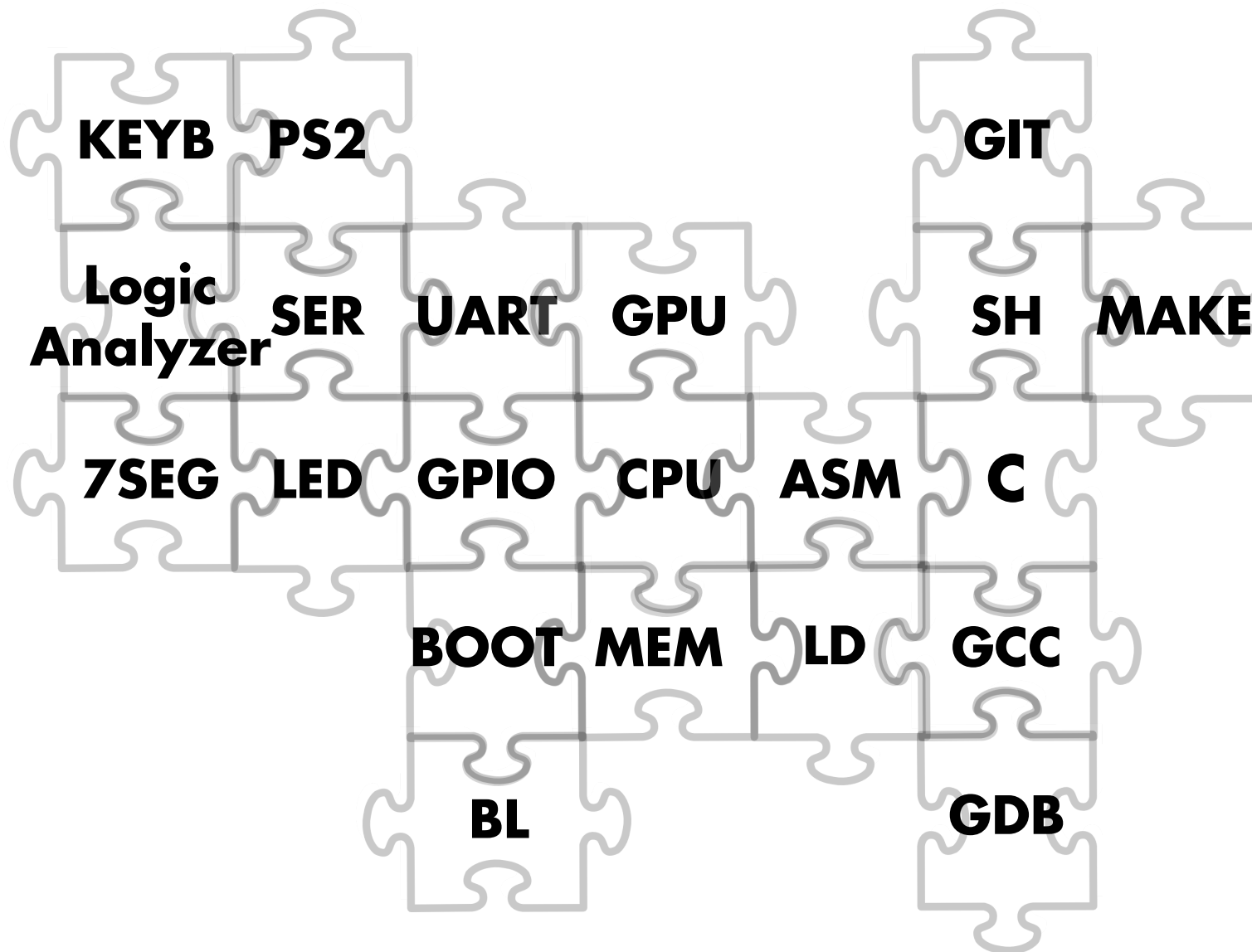
MEM

LD

GCC

BL

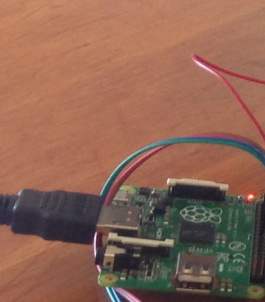
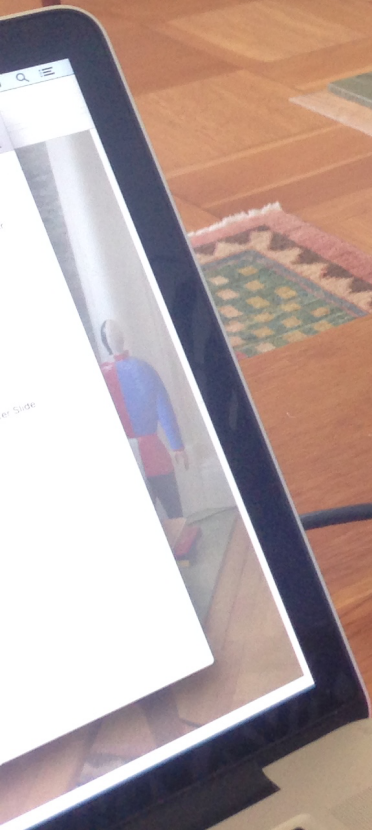
GDB



Raspberry Pi System Monitor

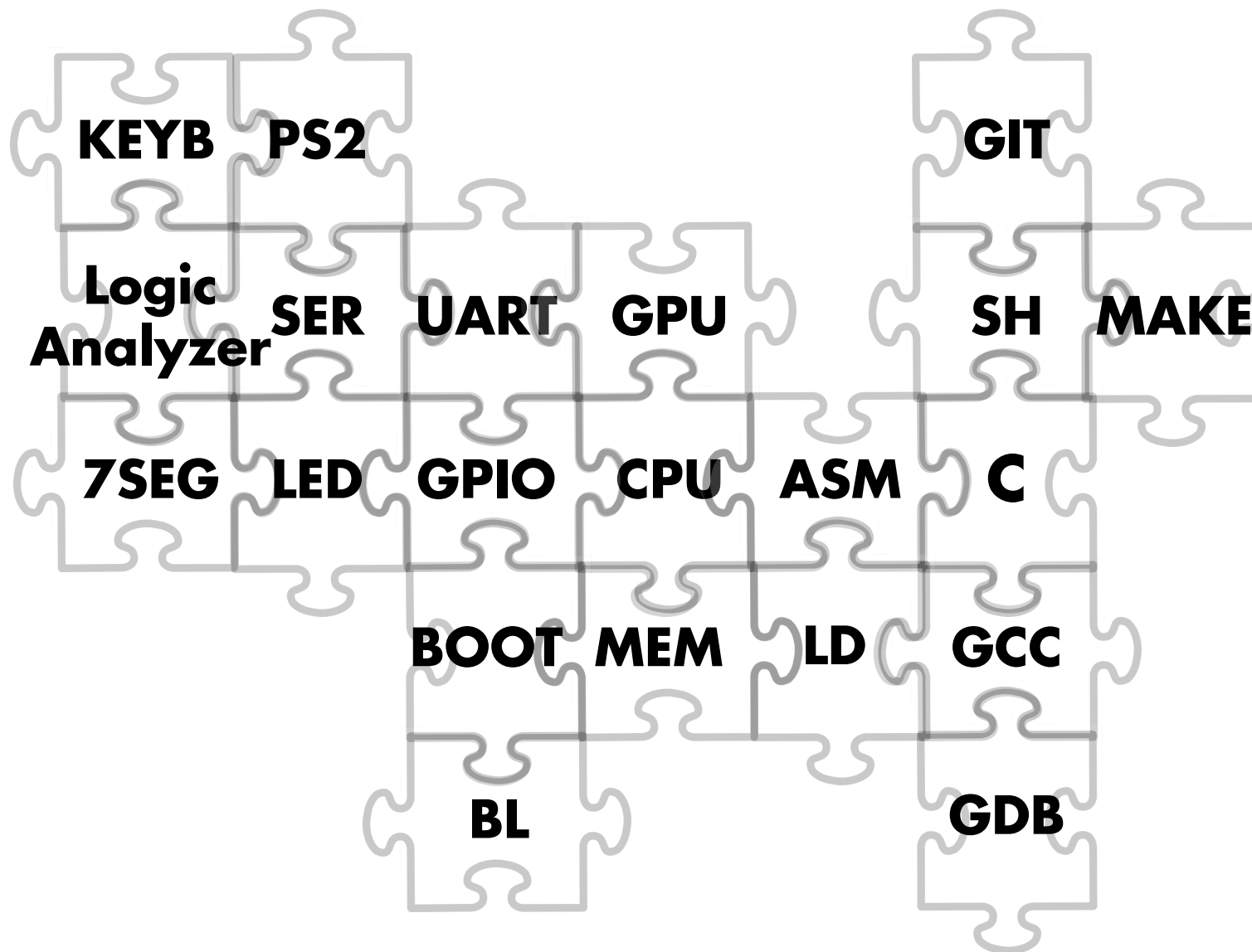
```
*
00008000- 07 DA A0 E3 1B 00 00 EB
*
00008008- B9 1B 00 EA 18 F0 9F E5
*
00008010- 18 F0 9F E5 18 F0 9F E5
*7000
00007000- 00
*
00007008- 00 00 00 00 00 00 00 00
*
00007010- 00 00 00 00 00 00 00 00
*
00007018- 00 00 00 00 00 00 00 00
*7000:01 00 a0 e3 1e ff 2f e1
*7000,7007
00007000- 01 00 A0 E3 1E FF 2F E1
*70000
1
*7000/7007
e3a00001: mov r0, #1
e12ffff1e: bx lr
*
```

SONY

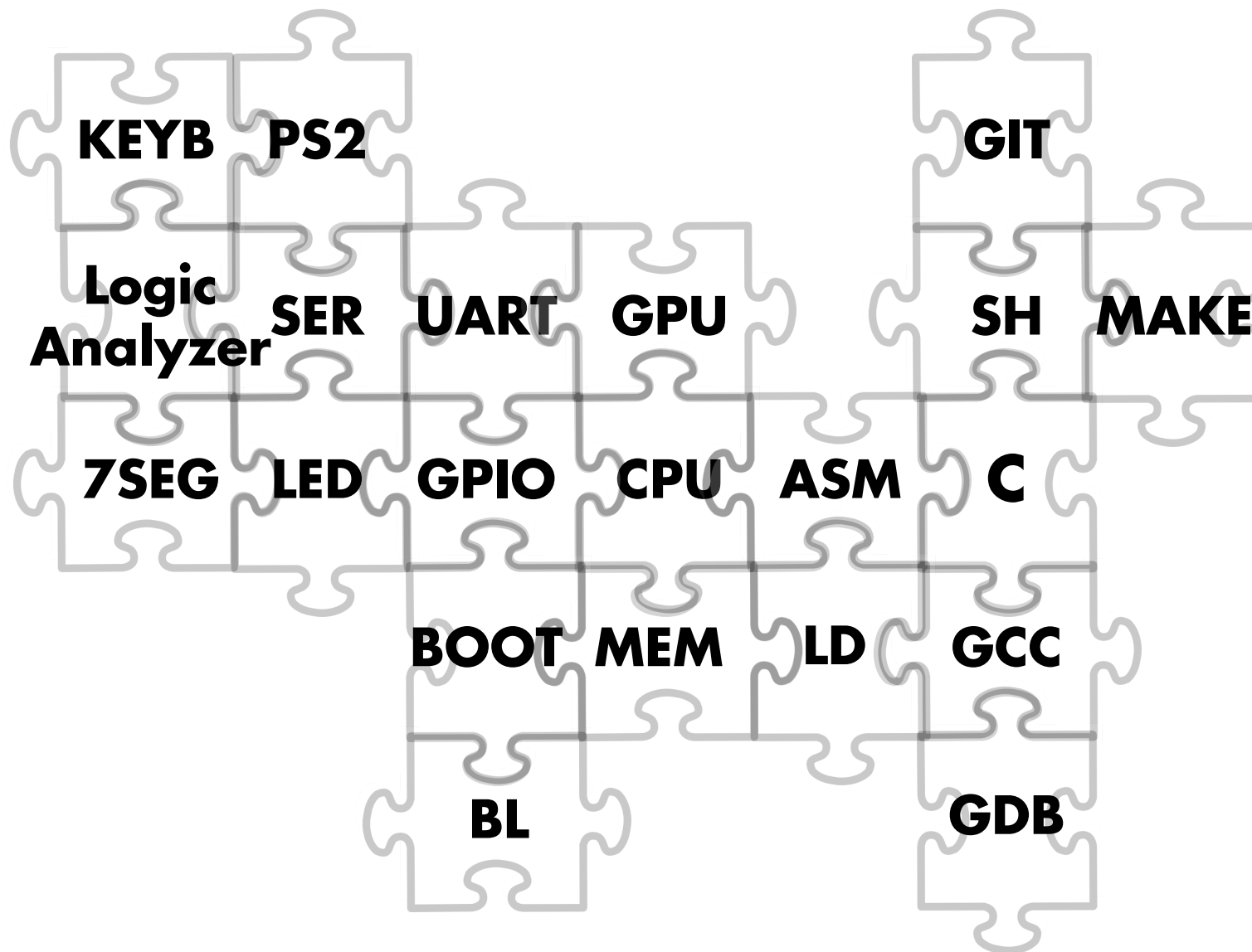


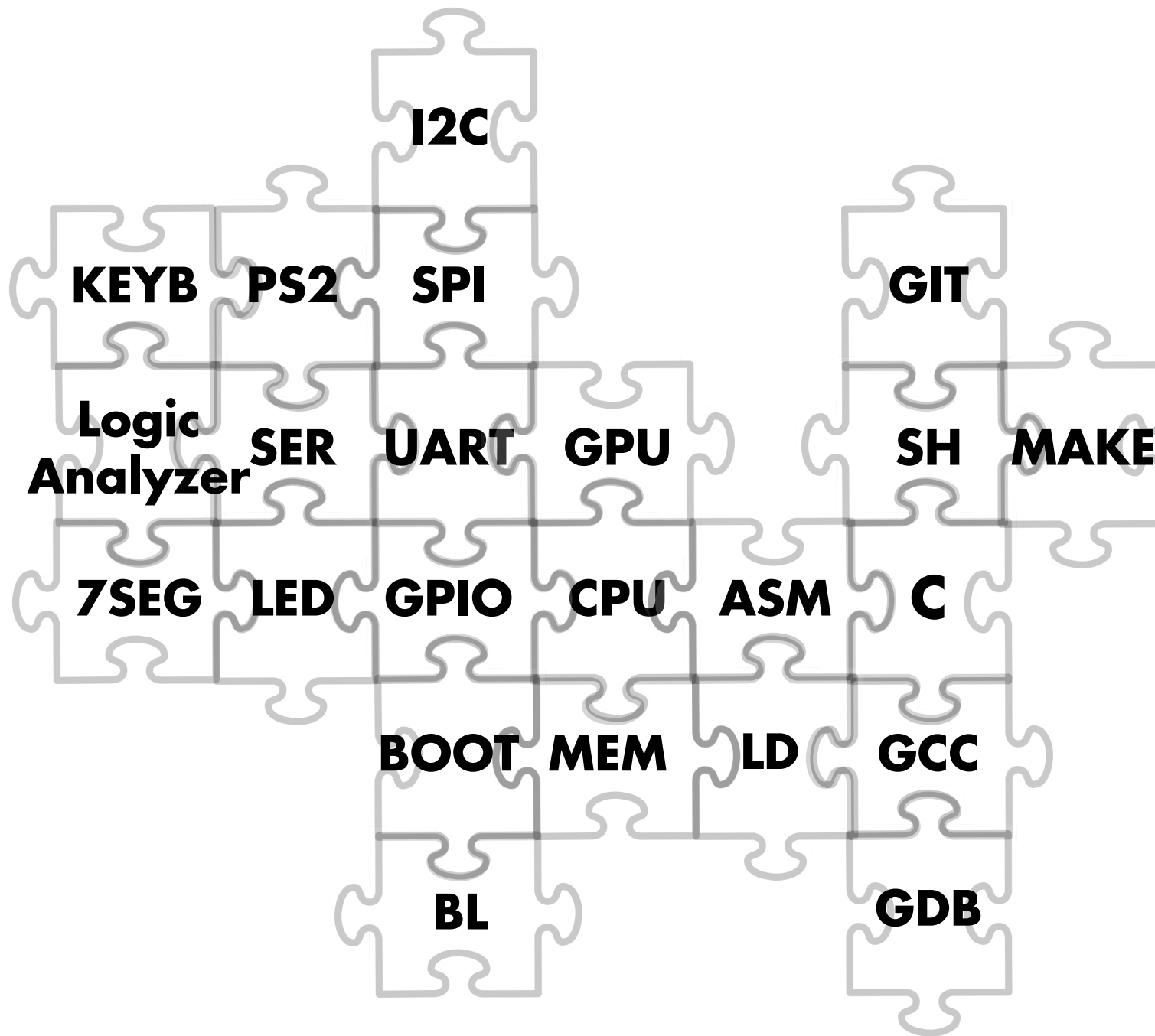
```
*
*
*      APPLE II
*      SYSTEM MONITOR
*
*      COPYRIGHT 1977 BY
*      APPLE COMPUTER, INC.
*
*      ALL RIGHTS RESERVED
*
*      S. WOZNIAK
*      A. BAUM
*
```

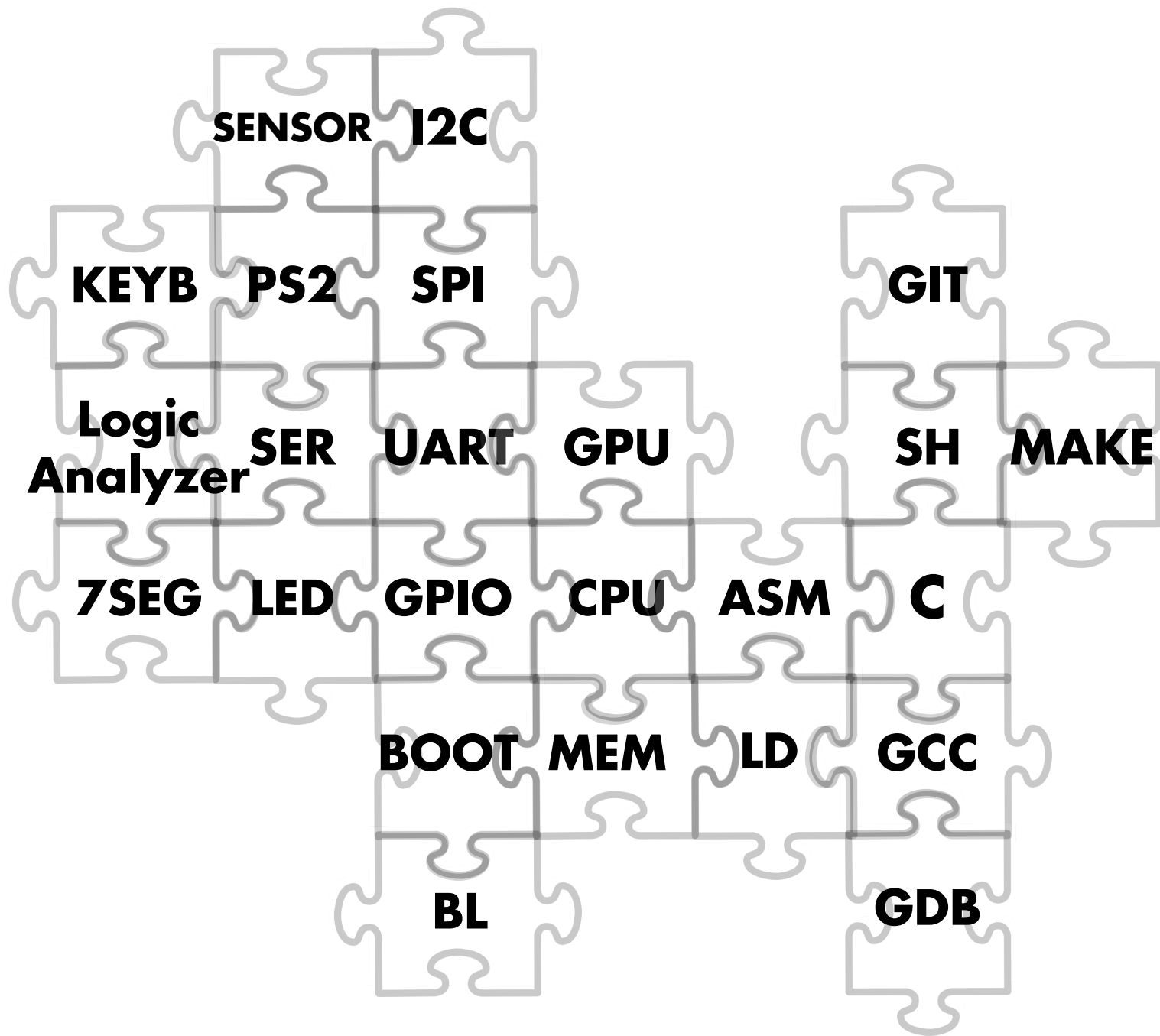
	TITLE	"APPLE II SYSTEM MONITOR"
LOC0	EPZ	\$00
LOC1	EPZ	\$01
WNDLFT	EPZ	\$20
WNDWDTH	EPZ	\$21
WNDTOP	EPZ	\$22
WNCDBTM	EPZ	\$23
CH	EPZ	\$24
CV	EPZ	\$25
GBASL	EPZ	\$26
GBASH	EPZ	\$27
BASL	EPZ	\$28
BASH	EPZ	\$29

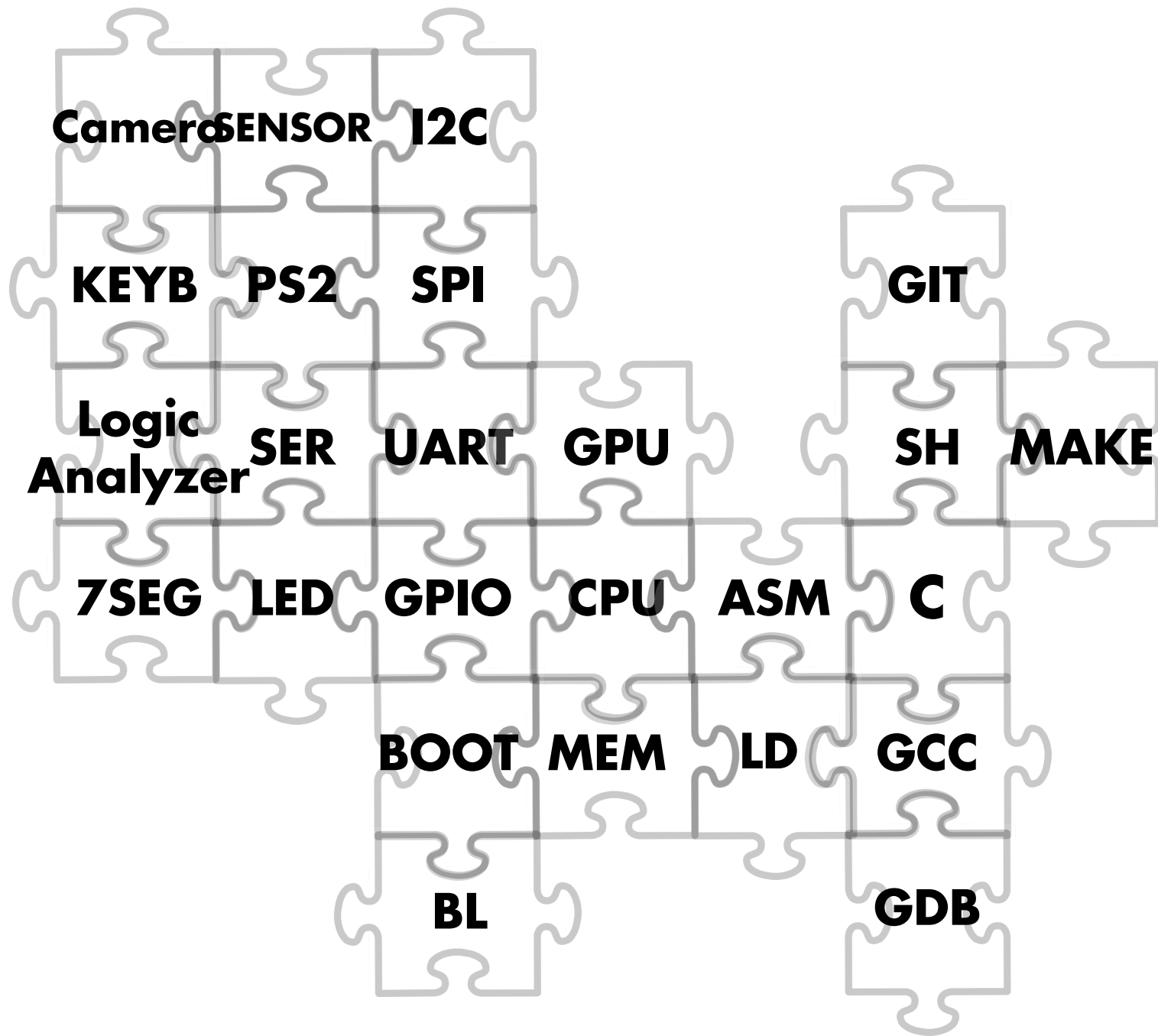


Your Journey Isn't Over









Camera

SENSOR

I2C

KEYB

PS2

SPI

GIT

Logic
Analyzer

SER

UART

GPU

SH

MAKE

7SEG

LED

GPIO

CPU

ASM

C

BOOT

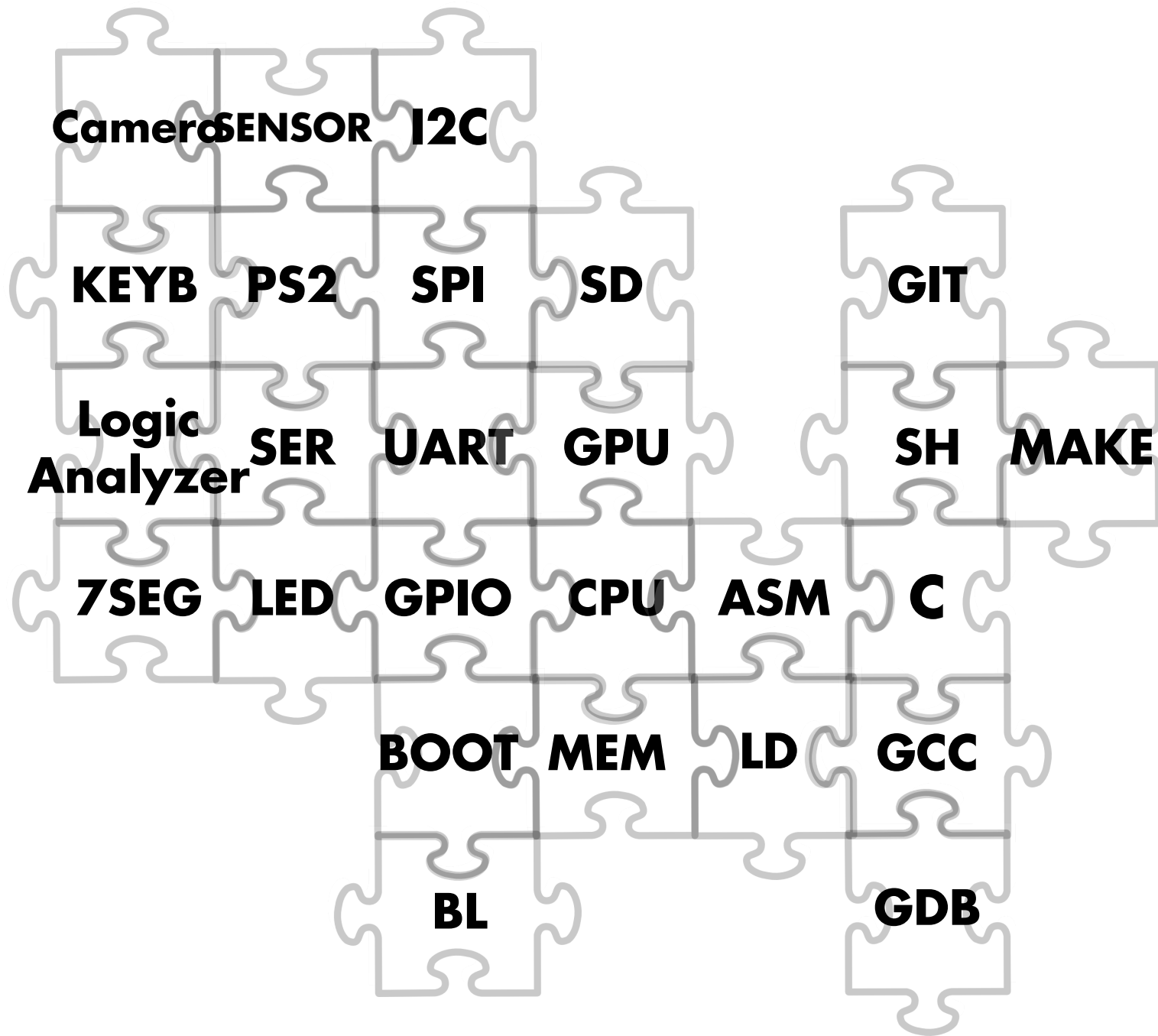
MEM

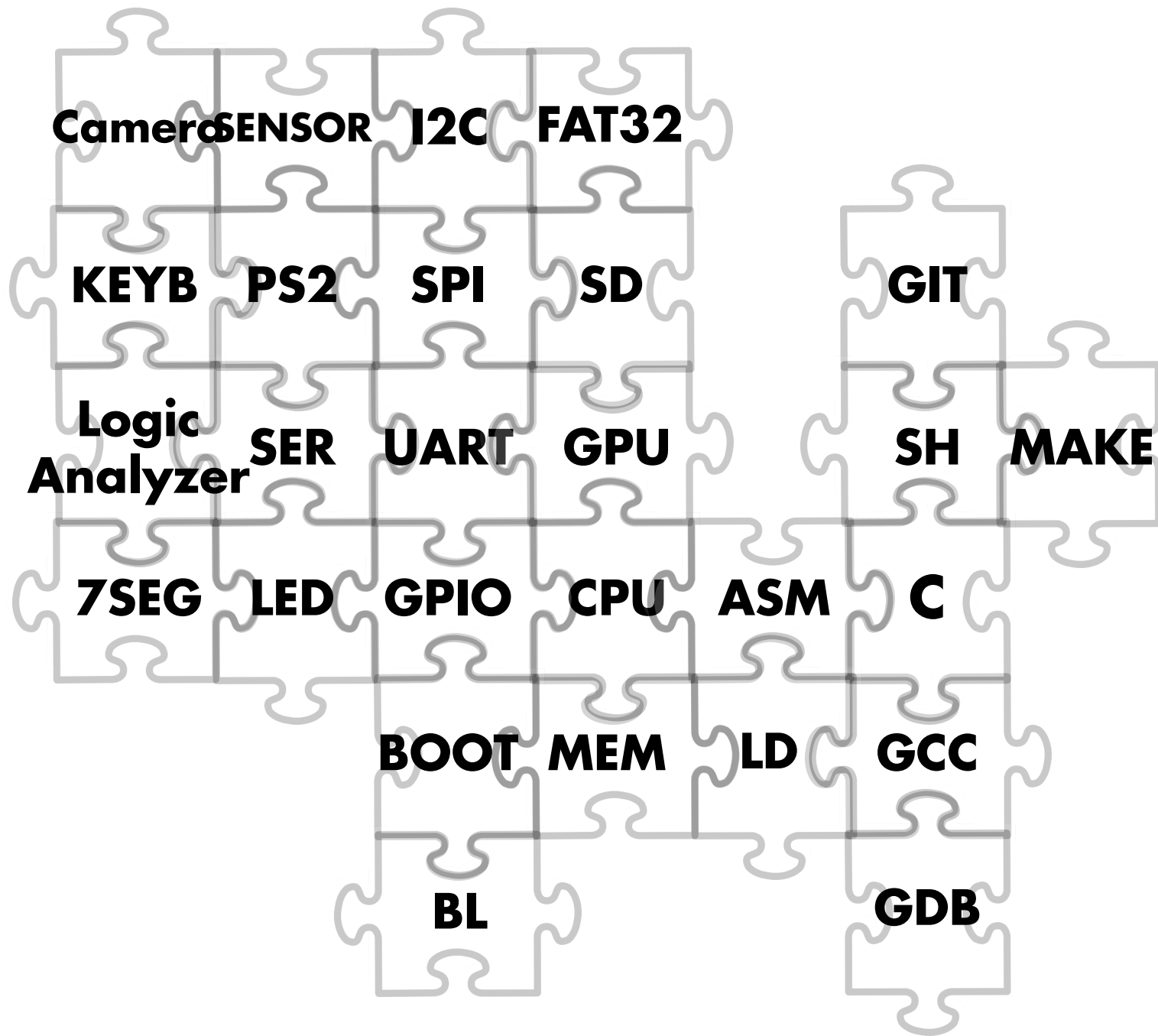
LD

GCC

BL

GDB





Camera

SENSOR

I2C

FAT32

KEYB

PS2

SPI

SD

GIT

Logic
Analyzer

SER

UART

GPU

SH

MAKE

7SEG

LED

GPIO

CPU

ASM

C

BOOT

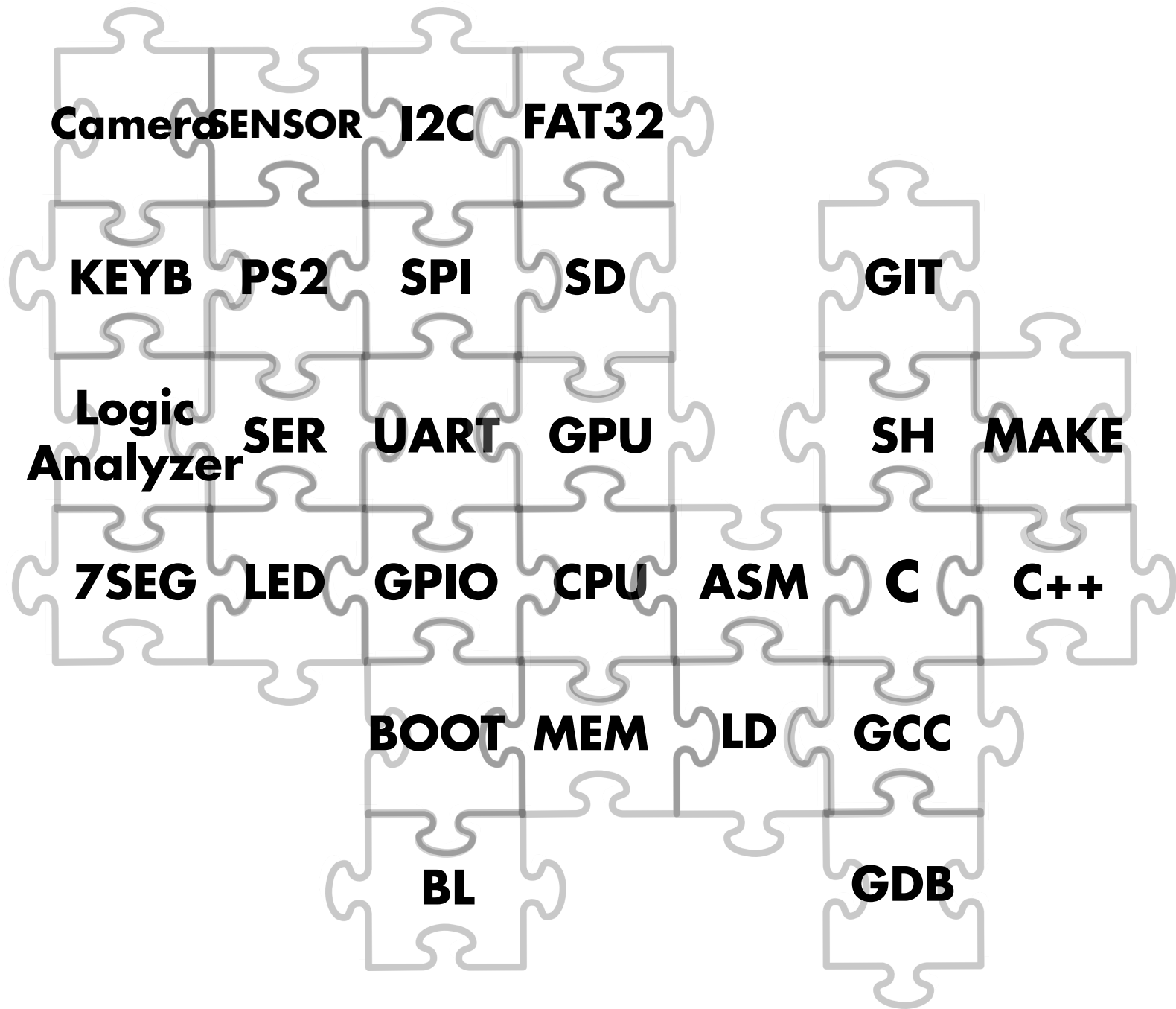
MEM

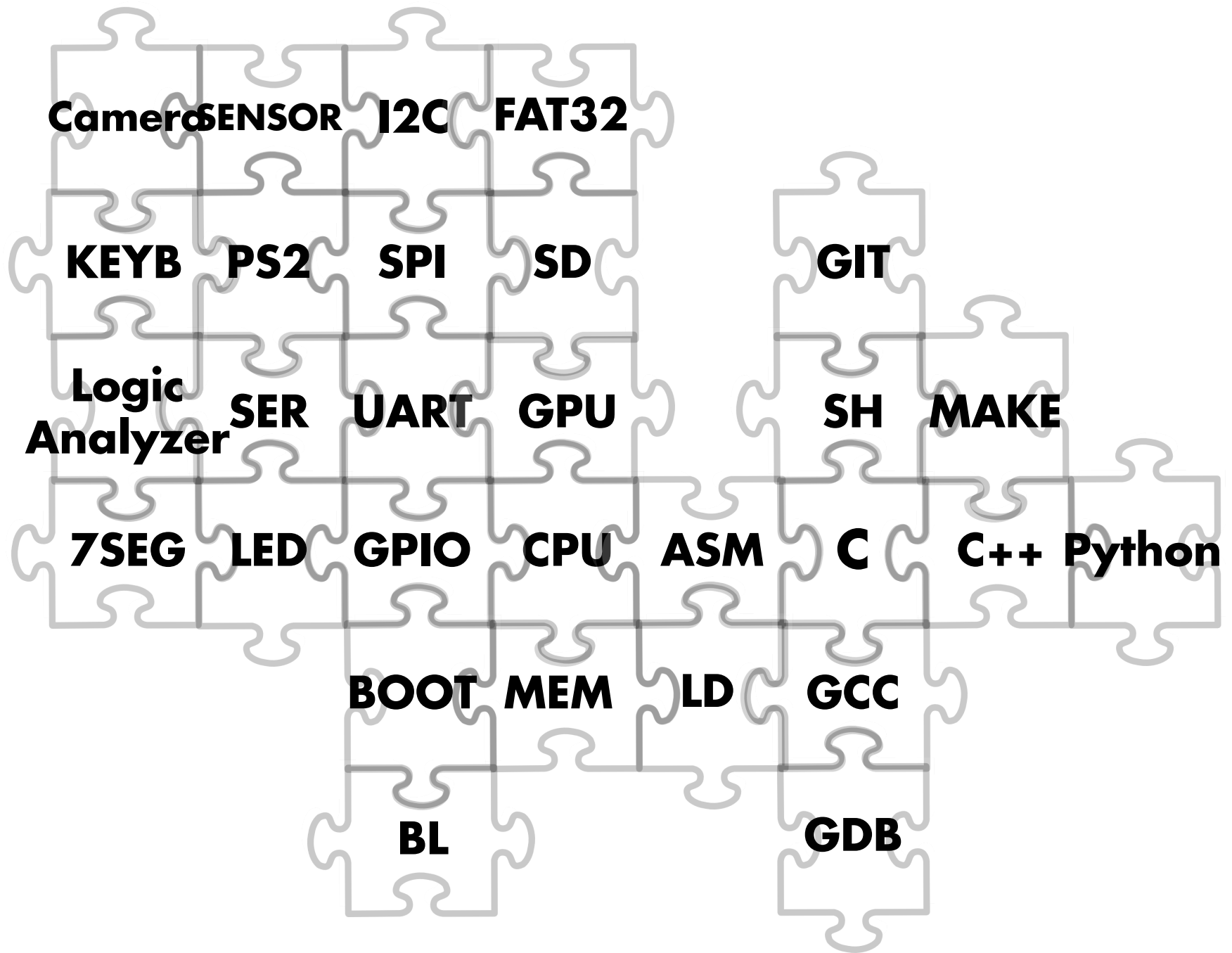
LD

GCC

BL

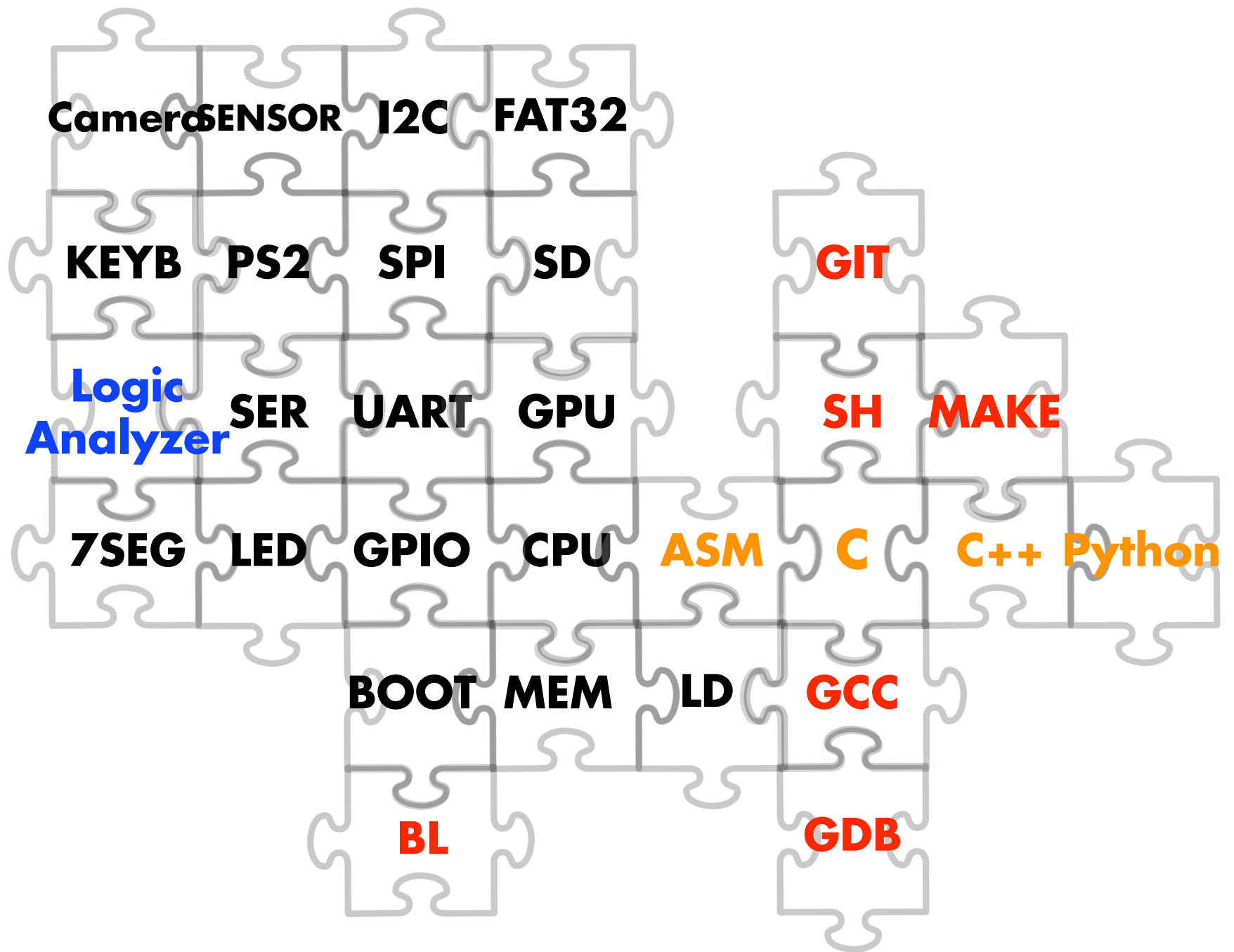
GDB

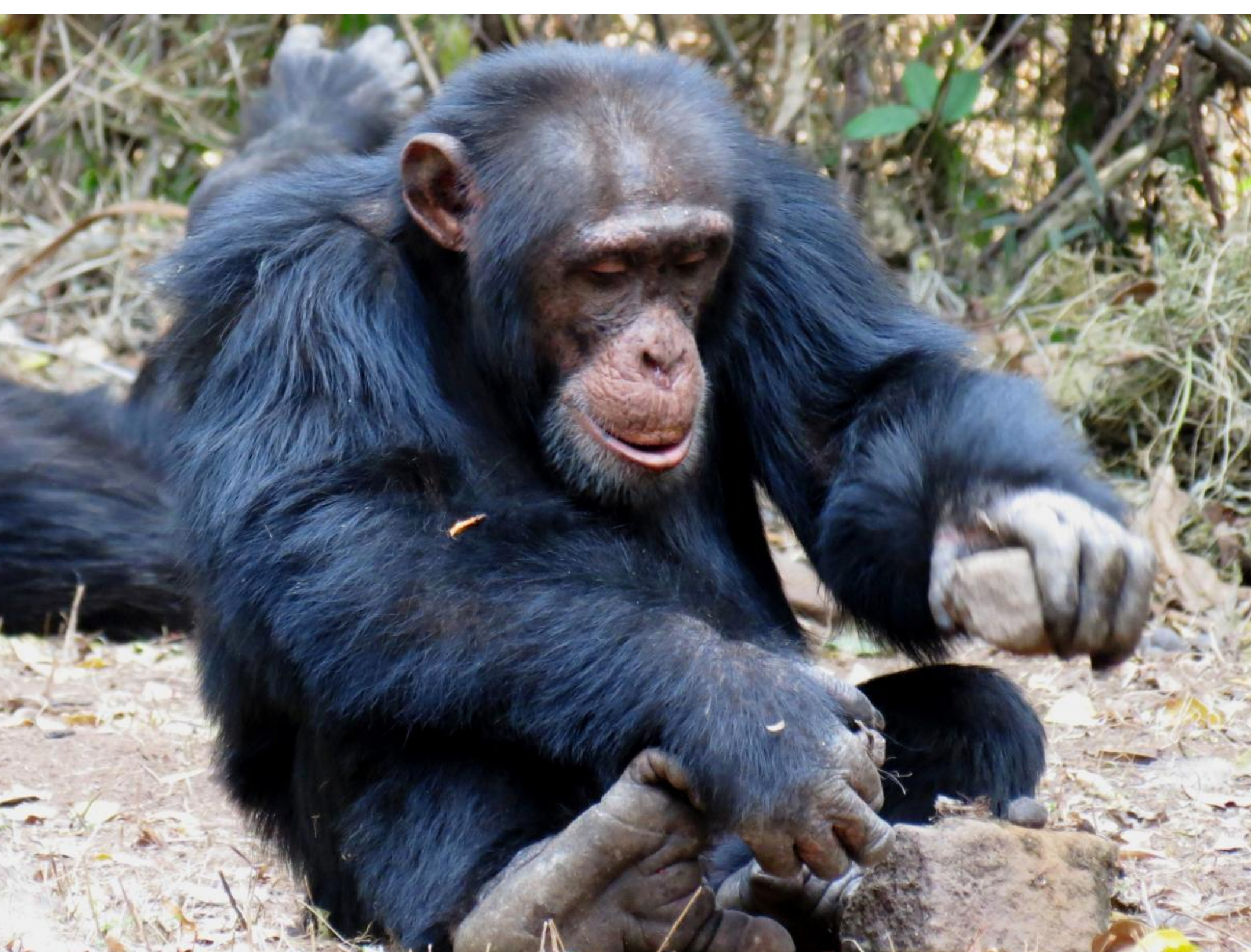




Goal 2

Master Your Tools









The Essential Set of Skills

Debugging, Testing, and Troubleshooting

Engineering Habits

Always start from a known working state, stop in a working state. If it breaks, what changed?

Epsilon-steps! Make simple small change, test carefully, then take another small step. Programming is much simpler if you figure out a good order to write the code!

Test, test, test, and test some more; Test as you go

Make things visible (printf, logic analyzer, gdb)

Systematic (D&C), not random, search for bug. Form hypotheses and perform experiments. Ask: why did my code do what I observed?

Don't be frustrated by bugs, relish the challenge, take frequent breaks

Copyrighted Material

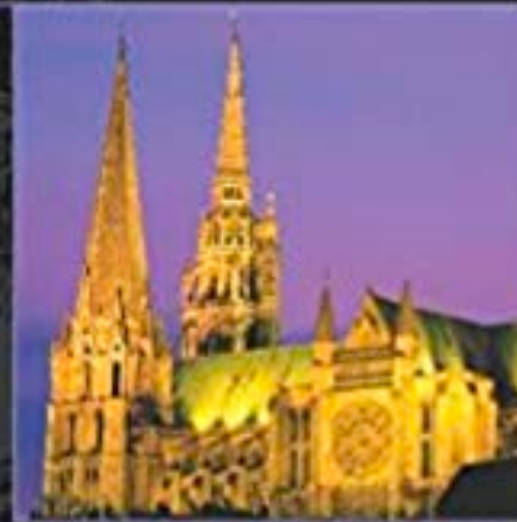
The Addison-Wesley Signature Series



TEST-DRIVEN DEVELOPMENT

BY EXAMPLE

KENT BECK



Copyrighted Material

2002

Principles

Test the interface not the implementation

Formulate tests as assertions on properties/invariants of the interface

Organize tests around each module/class (units)

Refactoring is iso-functionality; tests should stay green

Small steps, with continuous integration

Lower costs of developing tests using unit testing framework

Tests enable maintenance/evolution

Manual testing is not a *test*

Computer = Workspace

Organized Development Environment



<http://amhistory.si.edu/juliachild/>



To invent, you need a good imagination and a pile of junk
Thomas Edison

Be a Maker and a Doer

Ren Ng invents the Light Field Camera



(A)



(B)



(C)

Figure 3.10: The medium format digital camera used in our prototype.

Linux and Beyond

Operating Systems

Multiple processes (scheduling)

Memory management (virtual memory)

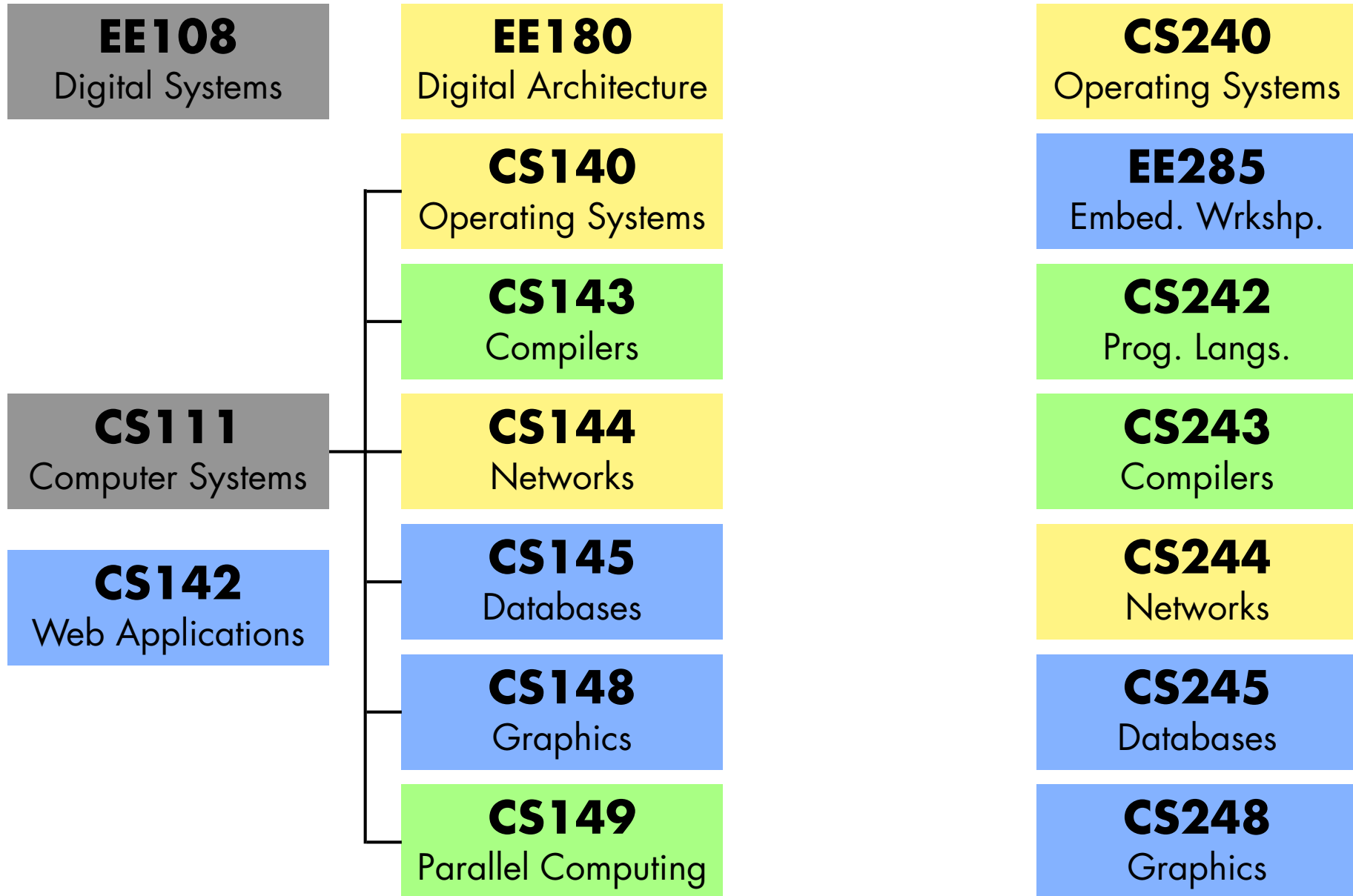
Share peripherals (devices)

File systems and storage

Networking

Protect/secure processes from each other

Next Steps



Demonstrations

- **Fri Mar 24th from 9:00 to 11:00 am in Gates B02**
- **Start with fast forward : 2 minute project pitch**
- **Setup project and demo to students and instructors**

Submission

- **Due Fri Mar 24h at midnight**
- **Final github commit of code**
- **Include README.md describing your project (pictures!!, attributions)**

Logistics

- **Return pi and keyboards (the rest of the stuff is yours)**
- **Submit reimbursements**

Comments and Suggestions?

**Please remember
to fill out course evals!**

Thank you!