Embedded Systems (7)

- Will start at 15:10
- PDF of this slide is available via ScombZ

Hiroki Sato <i048219@shibaura-it.ac.jp>

15:10-16:50 on Wednesday

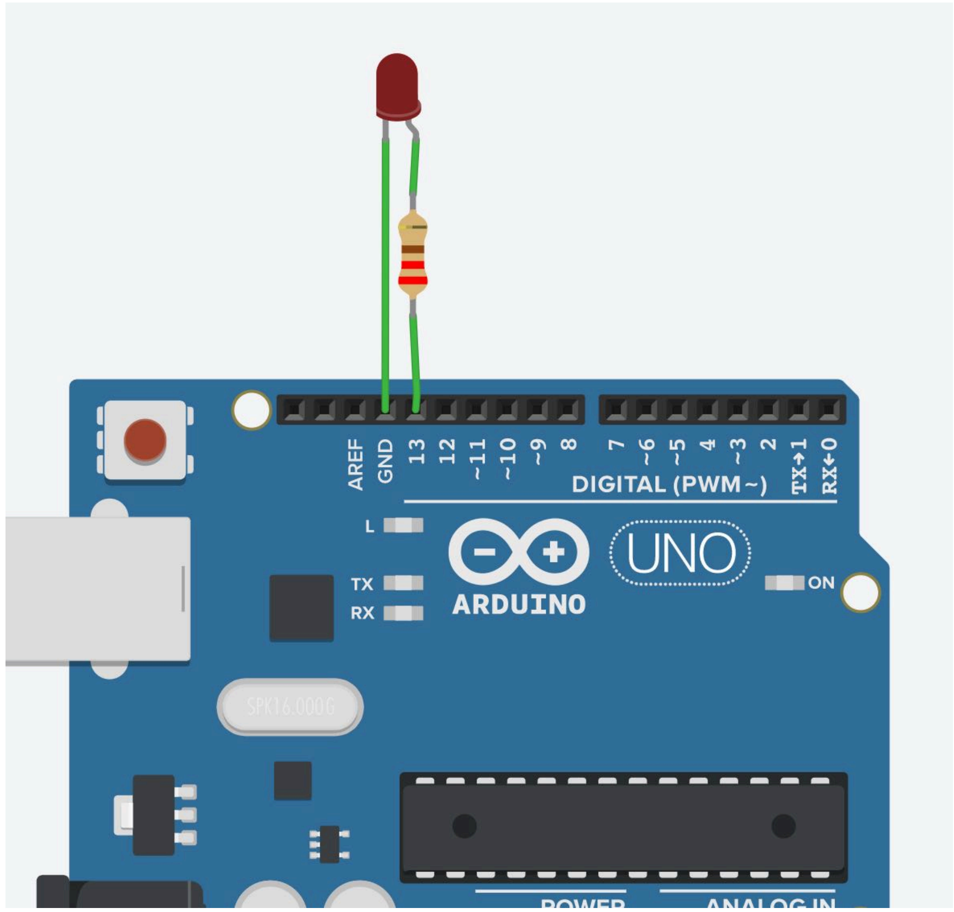
Targets At a Glance

- **What you will learn today**
  - Recap: project (b), (c) in the last week
  - Communication (in two weeks)
    - Key concepts: *parallel/serial, synchronous/asynchronous, full/half-duplex, hardware-assisted/software-based, wired/wireless, protocol stack, ...*
    - Examples: *parallel port, SPI, UART, Ethernet*

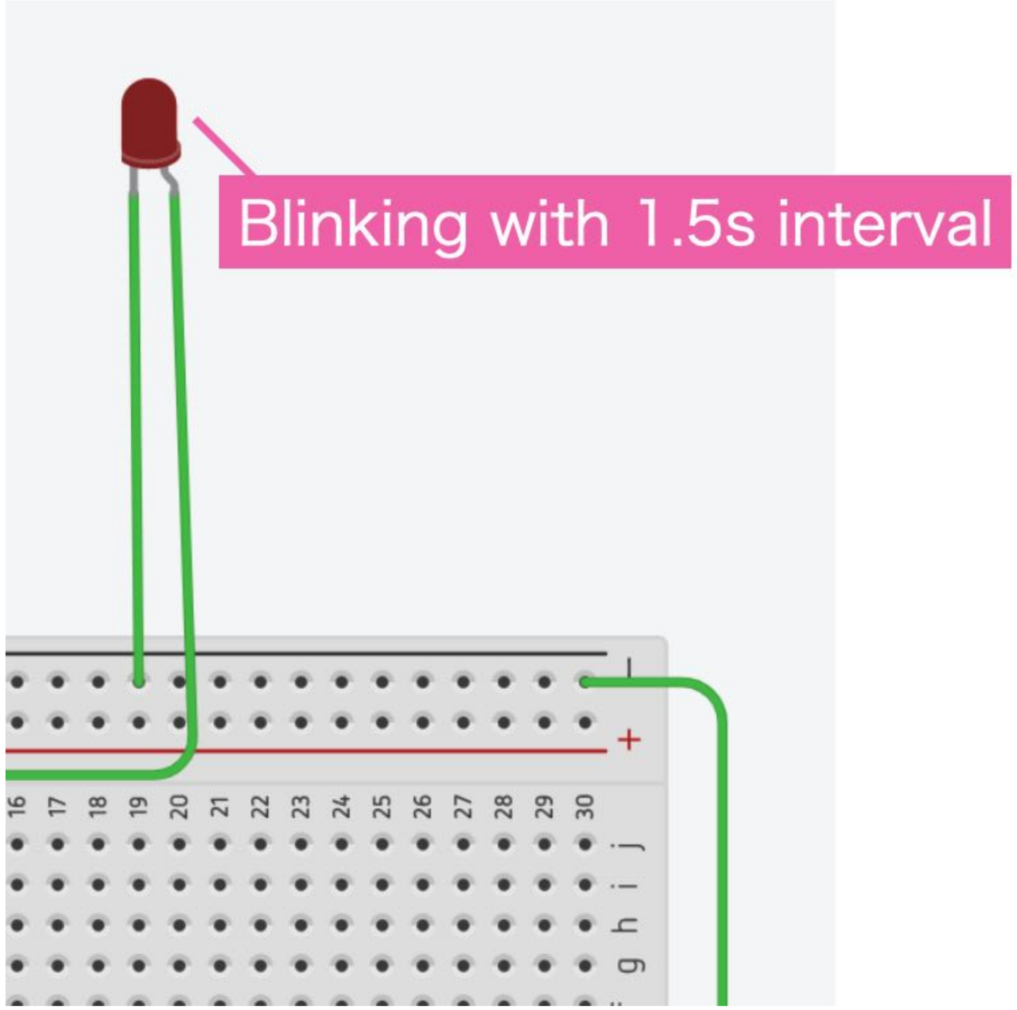
  - Today's Project
    - Simple 2-wire/3-wire communication by timer and external interrupt

(b), (c)

Blinker with 0.5s interval by timer interrupts.



Temp. sensing system with a blinker with 1.5s interval by timer interrupts.



(b)

```
void setup()
{
  TCCR1A  = 0;
  TCCR1B = (1 << WGM12) | (1 << CS12);
  OCR1A   = 31250 - 1;
  TIMSK1 |= (1 << OCIE1A);

  pinMode(13, OUTPUT);
}

ISR (TIMER1_COMPA_vect) {
  digitalWrite(13, !digitalRead(13));
}

void loop()
{
  delay(1000);
}
```

1/256

0.0625us x 256 = 16us

16us x 31250 = 0.5s

Note: OCR1A is 16-bit long

How to Use a Timer

- Pre-scaler configuration: TCCR1B

16.11.2 TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 16-5. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

(c)

1/1024

```
void setup()
{
  TCCR1A  = 0;
  TCCR1B = (1 << WGM12) | (1 << CS12) | (1 << CS10);
  OCR1A   = 23437 - 1;
  TIMSK1 |= (1 << OCIE1A);

  pinMode(13, OUTPUT);
}

ISR (TIMER1_COMPA_vect) {
  digitalWrite(13, !digitalRead(13));
}

void loop()
{
  delay(1000);
}
```

0.0625us x 1024 = 64us

64us x 23437 ≒ 1.5s

(c), alternative

```
void setup()
{
  TCCR1A  = 0;
  TCCR1B = (1 << WGM12) | (1 << CS12);
  OCR1A   = 31250 - 1;
  TIMSK1 |= (1 << OCIE1A);

  pinMode(13, OUTPUT);
}

int ic;
ISR (TIMER1_COMPA_vect) {
  if (ic++ % 3 == 0)
    digitalWrite(13, !digitalRead(13));
}

void loop()
{
  delay(1000);
}
```

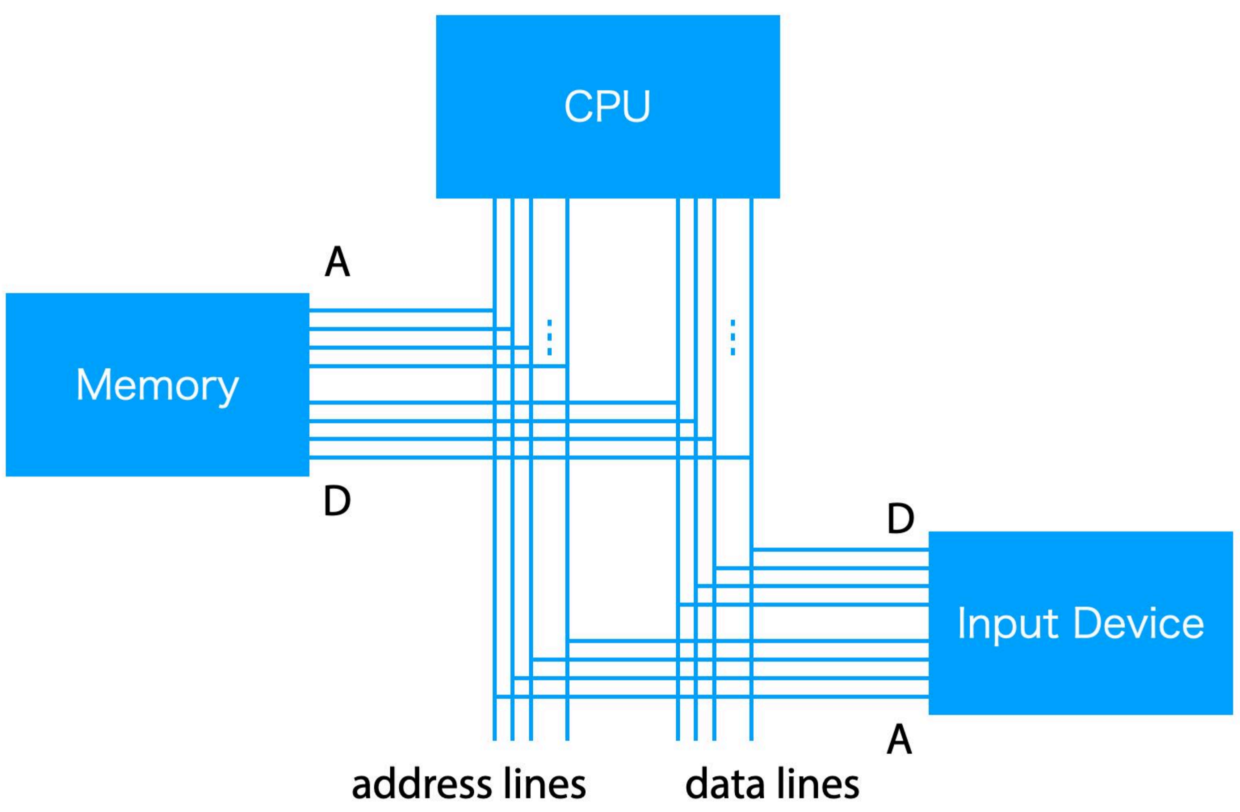
1/256

0.0625us x 256 = 16us

16us x 31250 = 0.5s

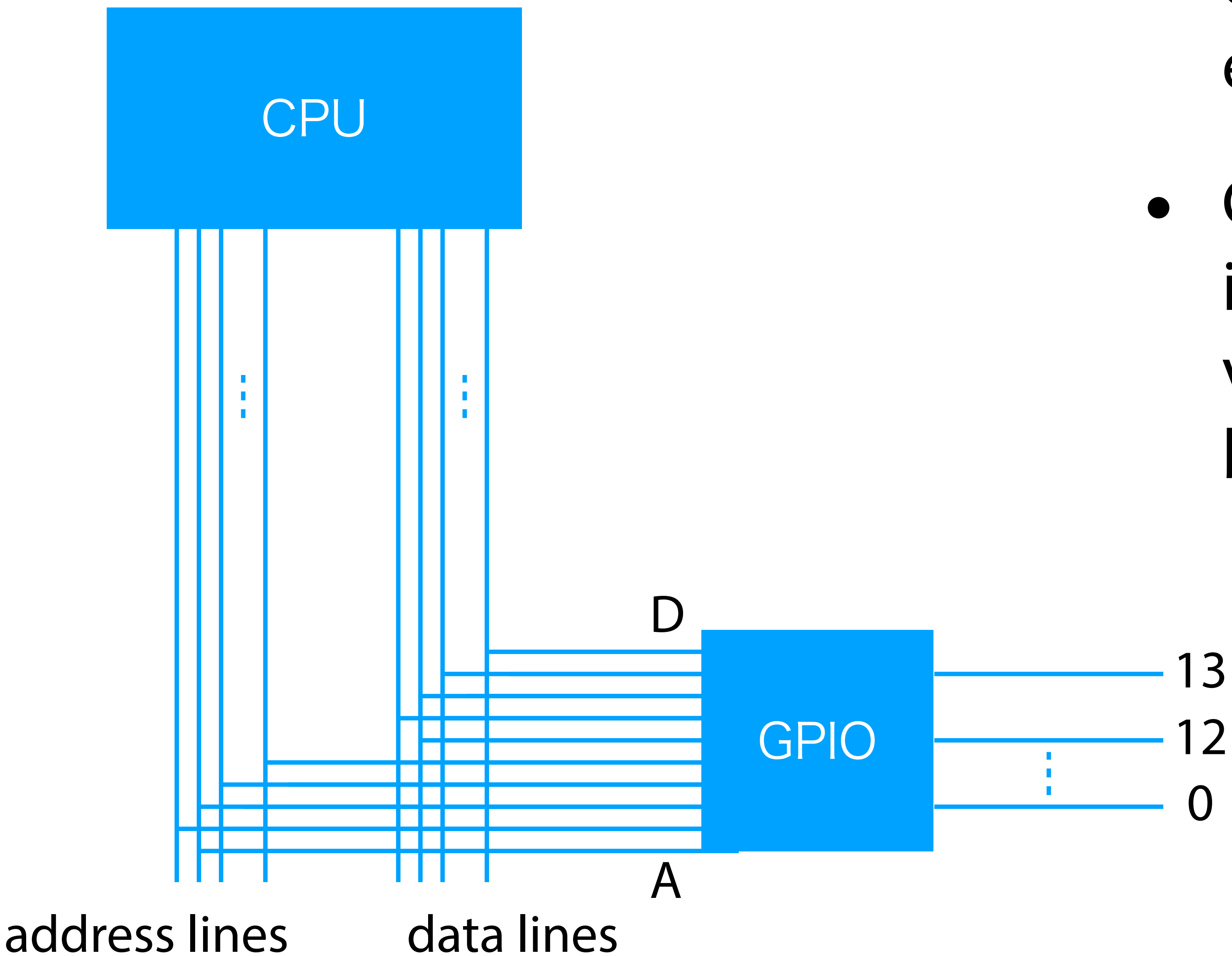
Communication

Communication



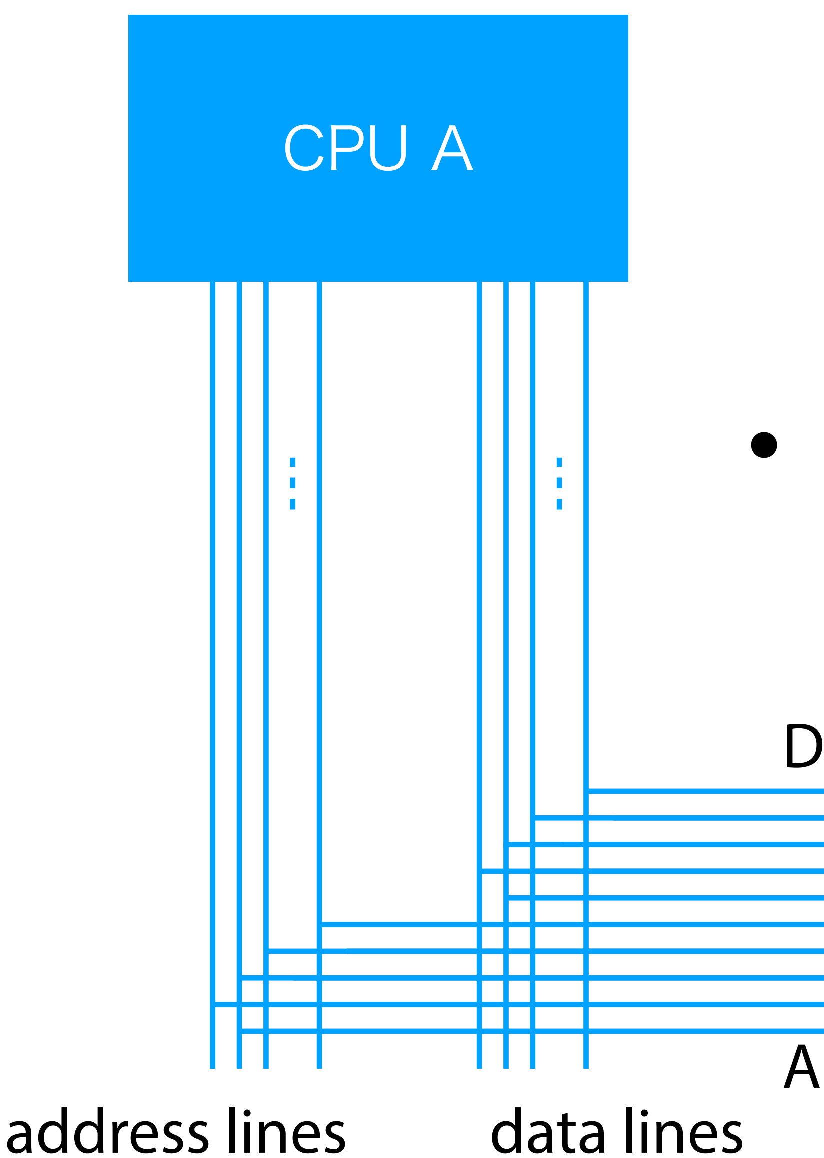
- Processor and devices attached to the bus can be communicated by memory access (including register access)
- How to realize a data transfer from a computer system to another?

I/O Interface

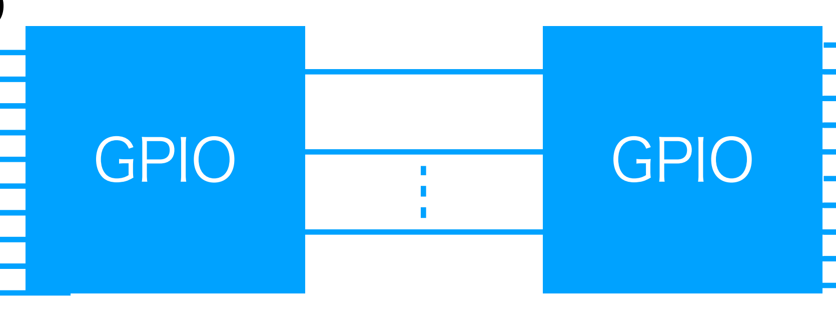


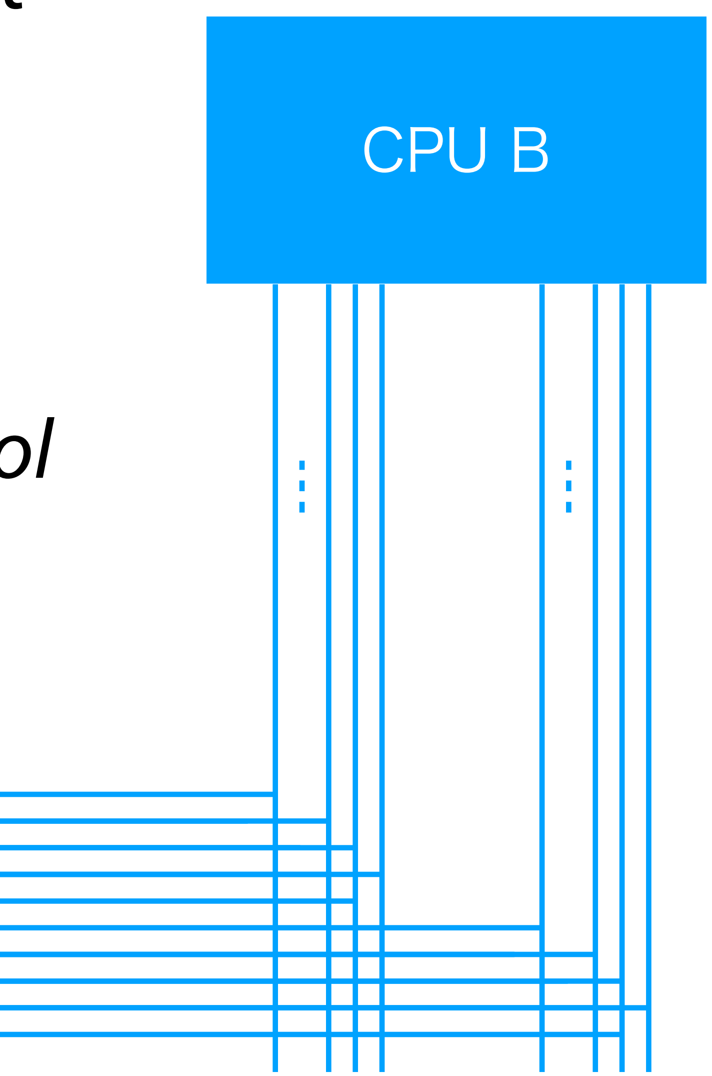
- A simple way to communicate with an external device
- GPIO (general-purpose input-output interface): voltage on the wires can be controlled.

Communication over GPIO



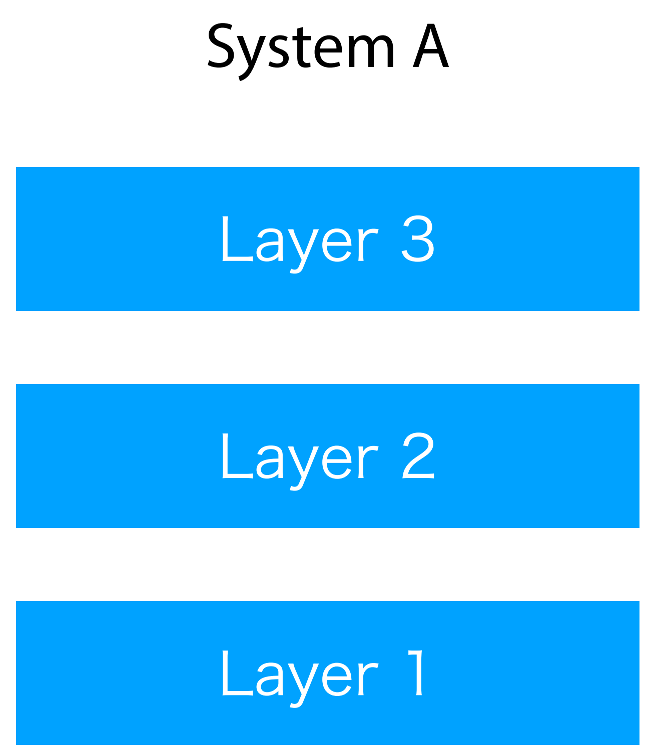
- Processor A and B must know how to communicate in advance.
- *Communication protocol*





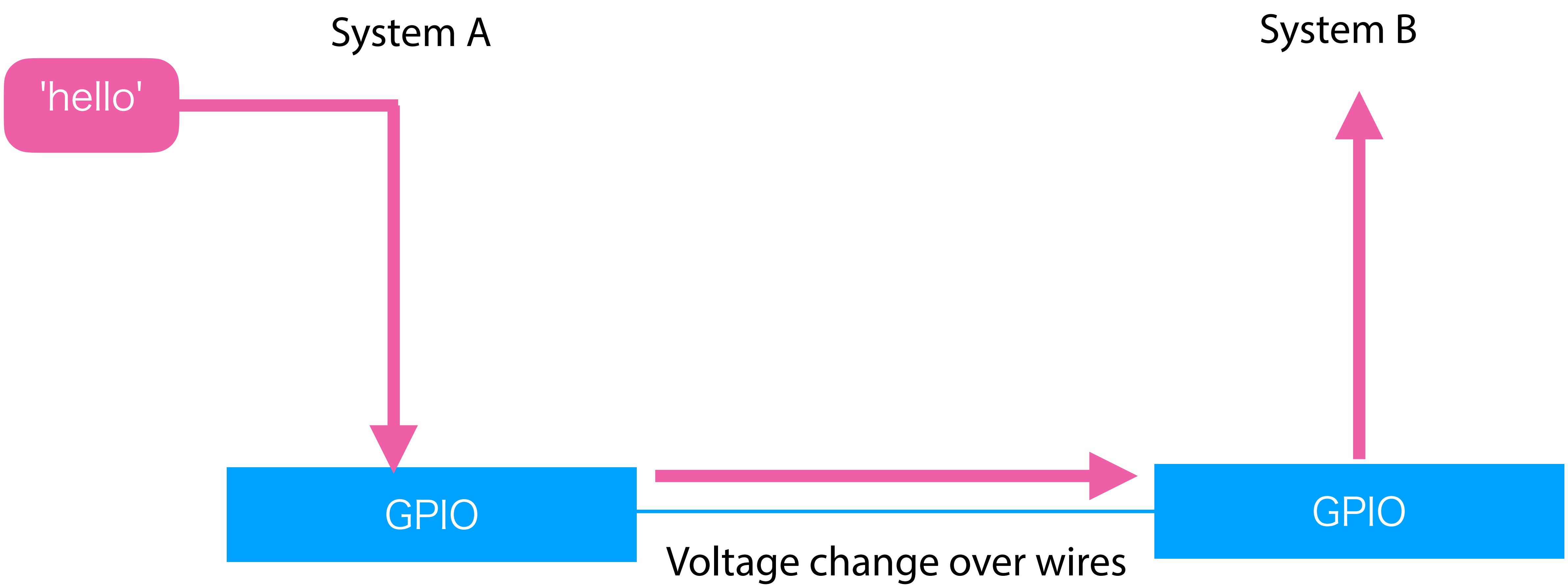
Protocol

- OSI (open systems interconnection) reference model (ISO/IEC 7498)
  - A layered abstraction model (7 layers)



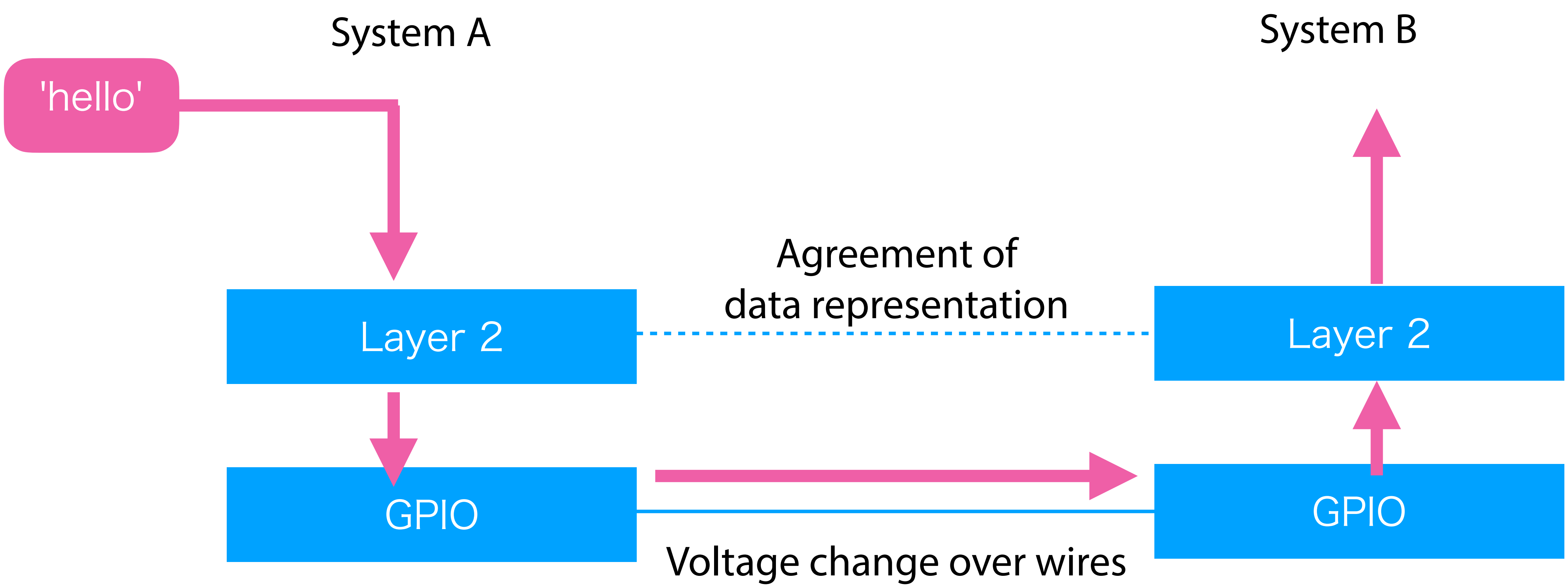
Why Layered Model?

- Let's review communications over GPIO
  - System A wants to send 'hello' to B.
  - How the data are represented on the wires?
  - When the data will arrive? How to know that?



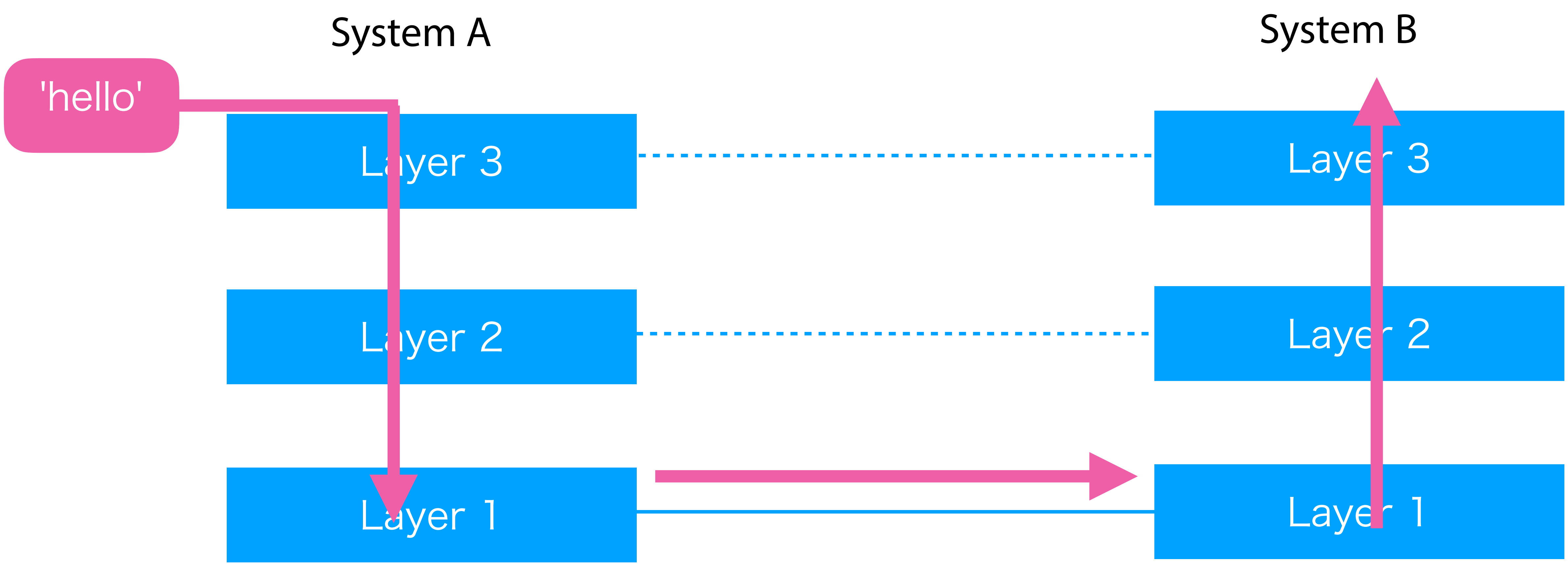
Why Layered Model?

- GPIO layer just knows how to convert the data to voltage changes
- Layer 2 knows how to represent the data: 'hello' is 0x68 0x65 0x6c 0x6c 0x6f, for example.



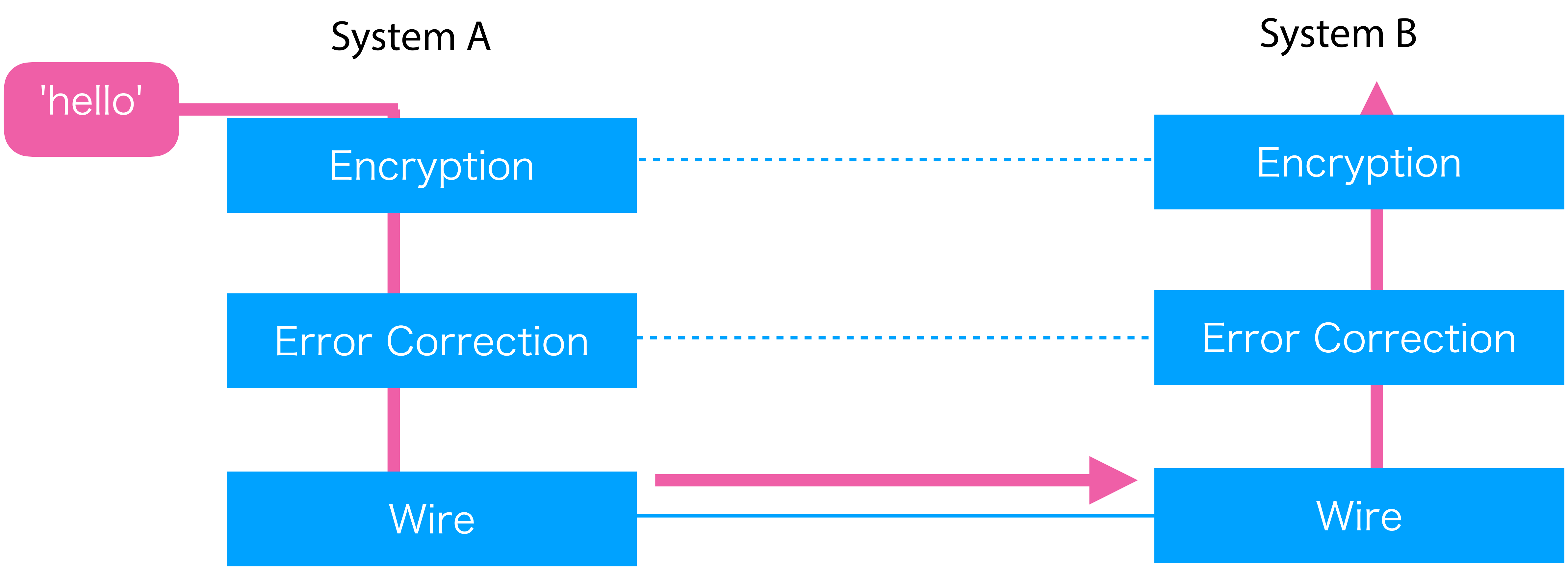
More Layers

- Protocol: "agreement" between each layer
- Protocol stack: structure of the communication system



Practical Examples

- Additional functionality can be represented in layers
- Layers can be constructed



3 Layers in OSI Model

Network

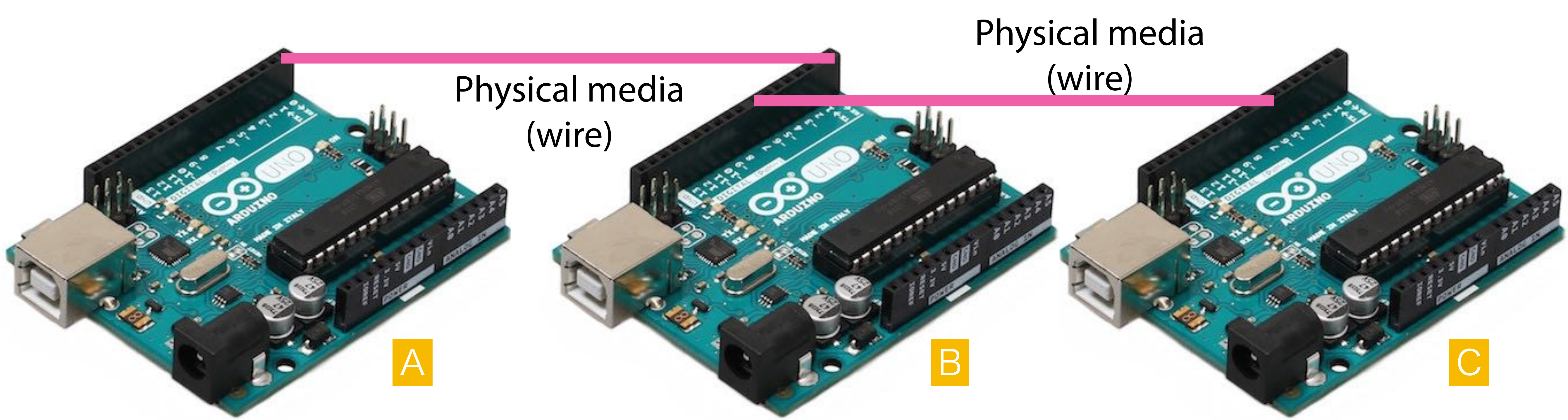
- How to deliver the data between multiple media

Data Link

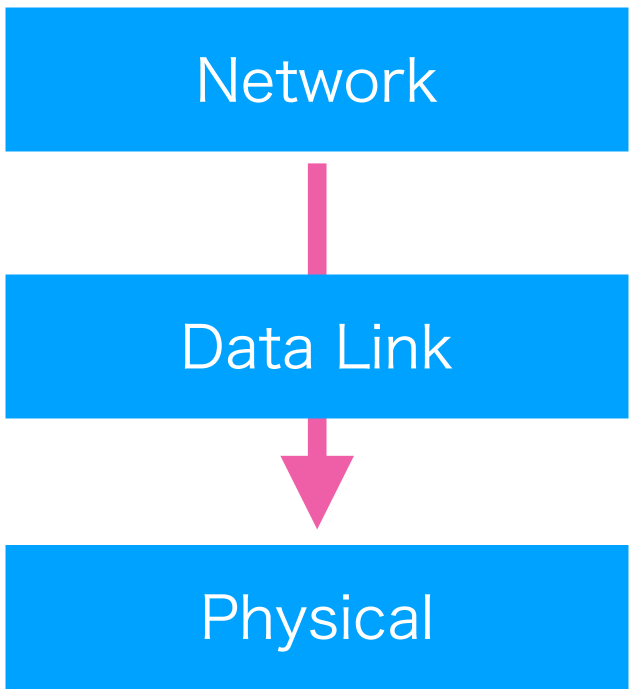
- Defines the "directly-connected" systems, which share the same physical media and how to communicate among them.

Physical

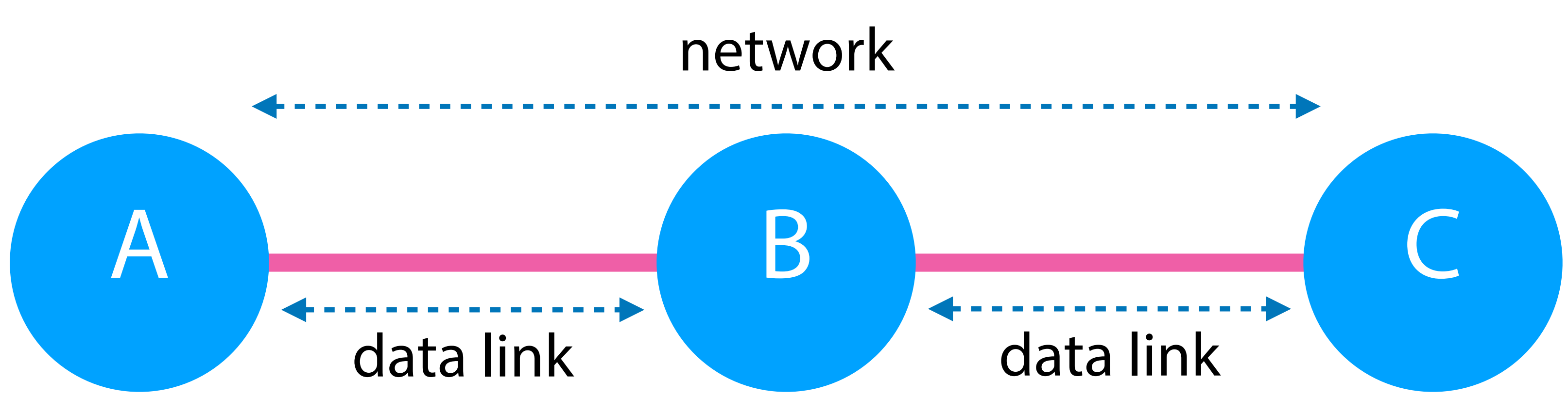
- Defines "media": wire, voltage, how to connect physically



3 Layers in OSI Model

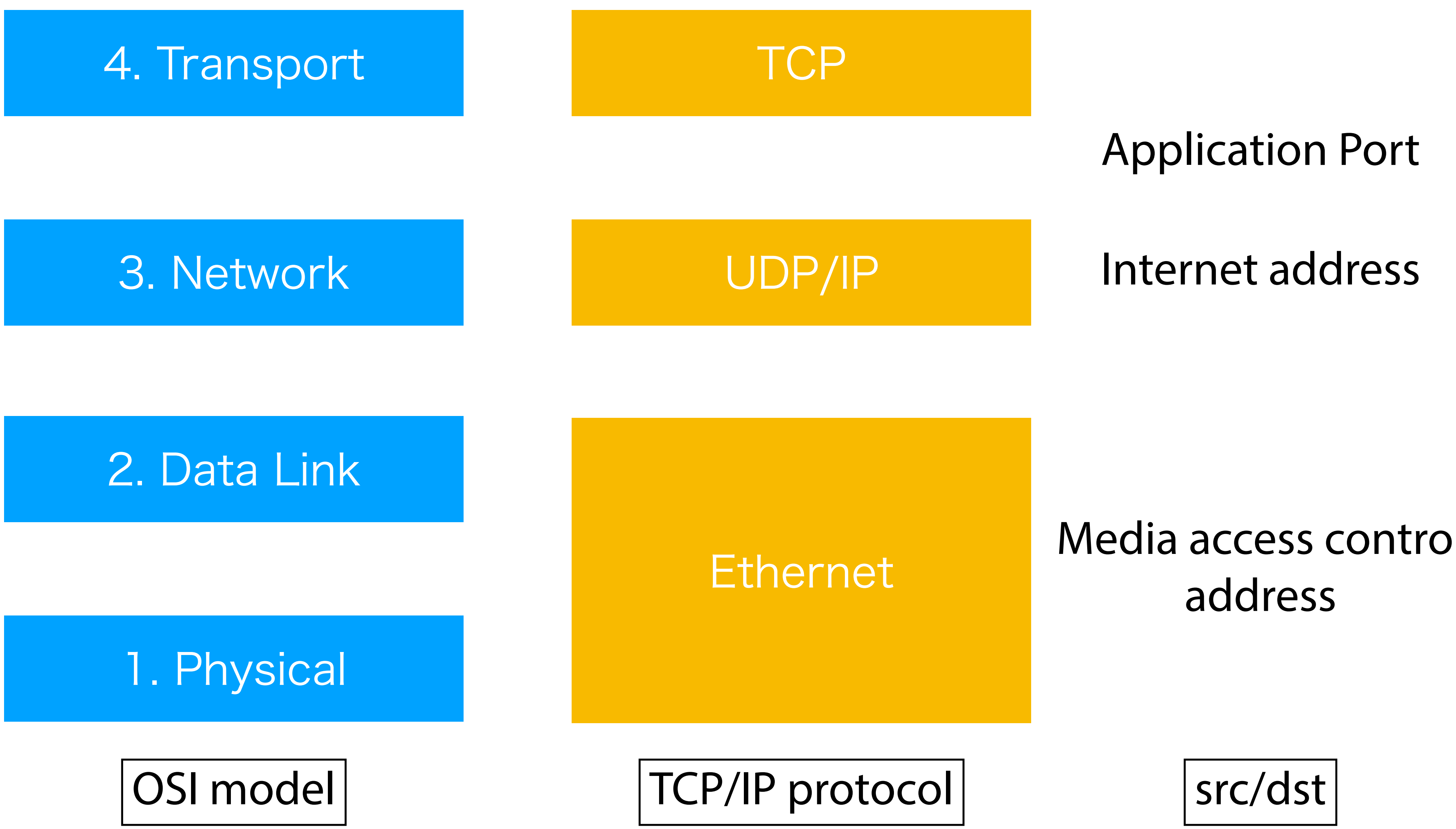


- How to deliver the data between multiple media
- Defines the "directly-connected" systems, which share the same physical media and how to communicate among them.
- Defines "media": wire, voltage, how to connect physically



Internet Protocol

- TCP/IP protocol stack is a foundation of Internet.
- Not strictly layered model, but can be understand in a similar way.

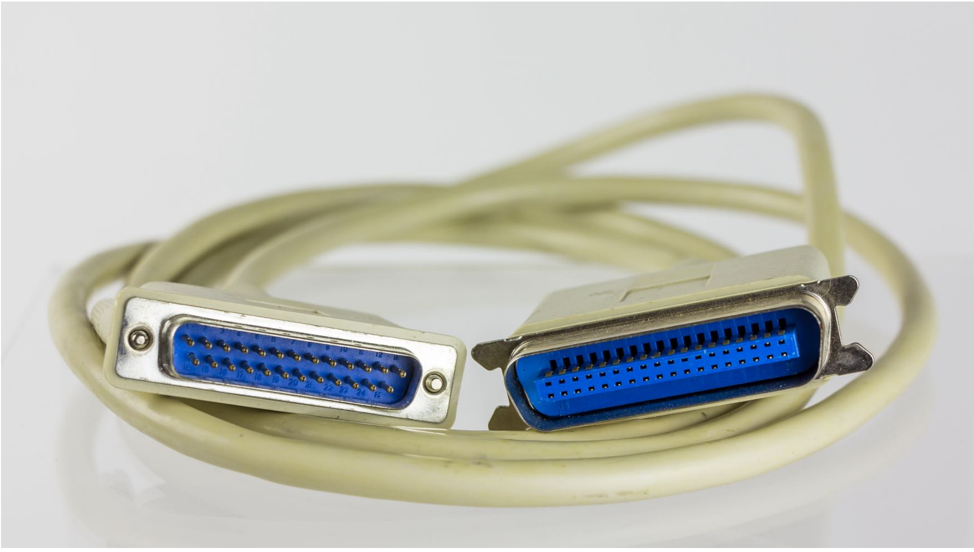


Physical Layer: Serial or Parallel

- Data are often represented in bytes while a single wire can represent only 1 bit at a time.
- How do we deliver multiple bits?
  - By using multiple wires
  - By using a single wire in a time-division manner.
- Parallel vs serial communication

Parallel Communication

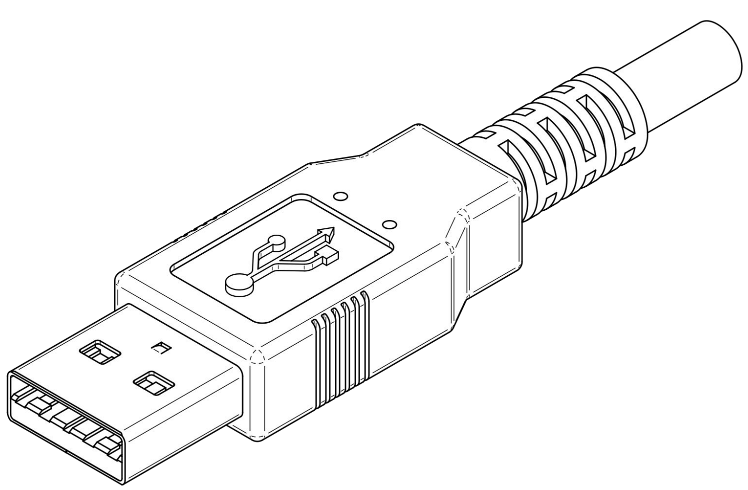
- Pros: easy to use in software
- Cons: difficult in hardware production and high-speed communication. No longer used actively these days.



IEEE 1284 DB-25 to 36-pin connector cable

Serial Communication

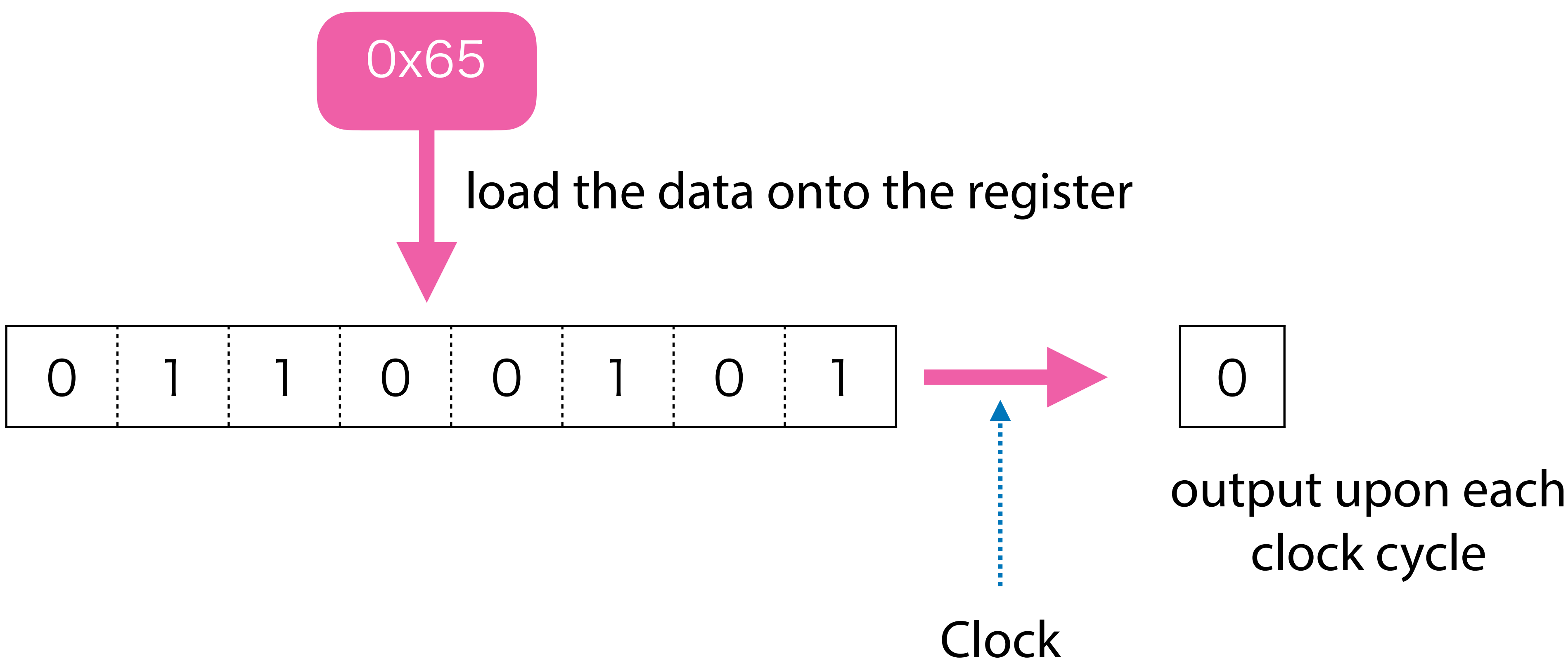
- Pros: easy to implement in hardware
- Cons: was considered difficult to realize high-speed communication



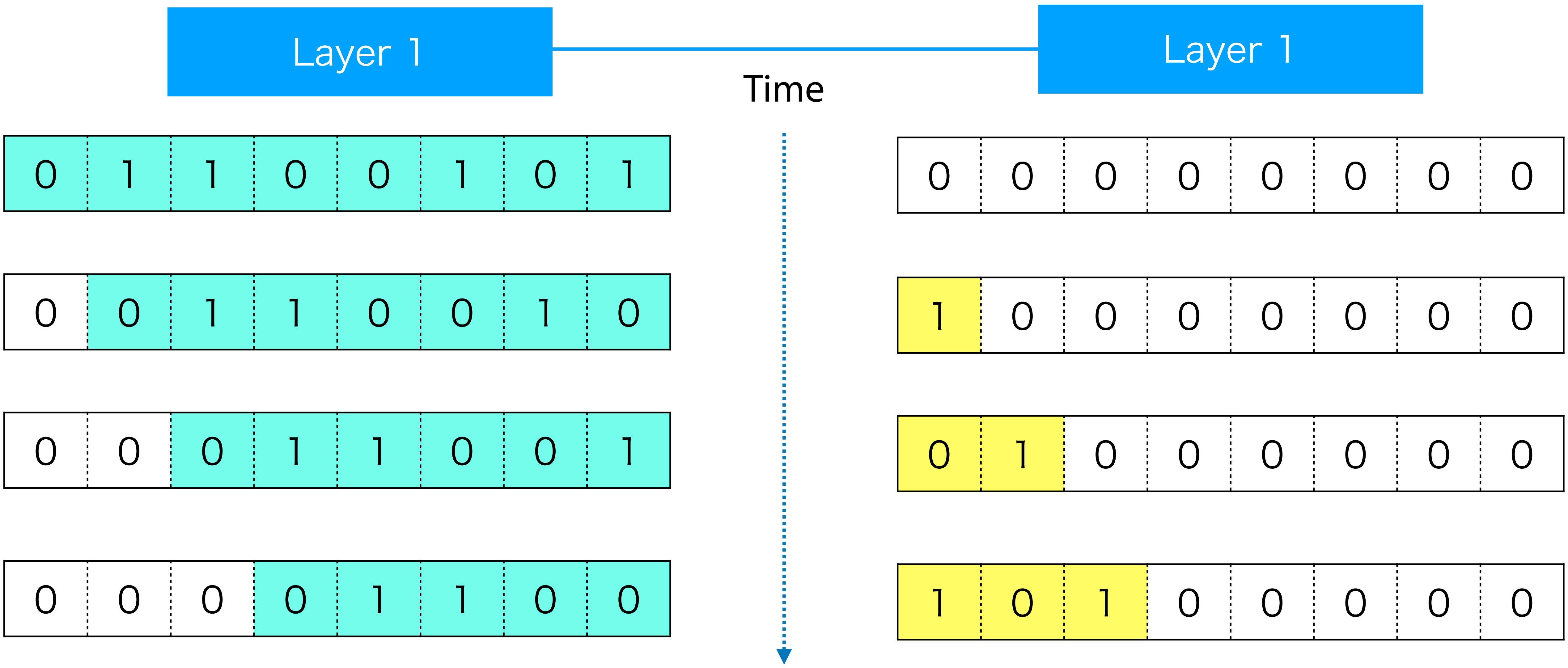
USB 2.0 Type-A, 4-pin cable

Serial Communication

- Shift register to convert data into a pulse sequence

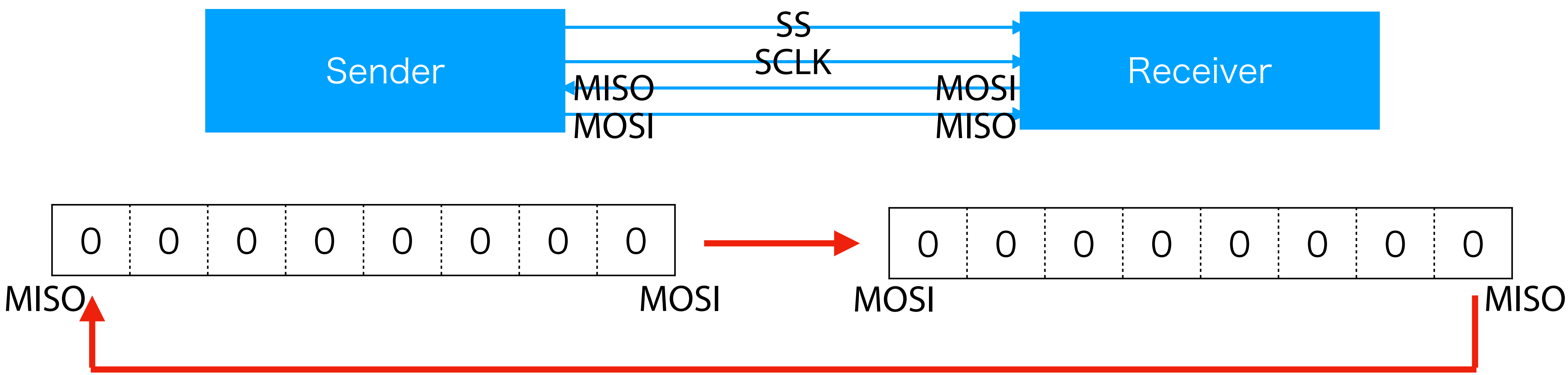


Serial Communication



- It takes 8 clock cycles to transfer 8-bit data. Note that it is not limited to 8 bits.
- The clock must be shared.

SPI (Serial Peripheral Interface)



- What layers SPI covers?
  - Physical layer (4 wires)
  - Data Link layer (wires can be shared, and SS as destination address)

Physical Layer: full-duplex or half-duplex

- **full-duplex**: both sending and receiving can be performed at the same time.
- **half-duplex**: you cannot send during receiving or vice versa.
- A single wire is typically half-duplex while it is possible to make it full-duplex by multiple access technology.
- SPI is full-duplex protocol.

Physical Layer: Synchronous or Asynchronous

- **Synchronous** means that the protocol requires a clock on both sides and the communication is based on the clock. The clock must be shared in some way (over a wire or something).
- **Asynchronous** means no clock is required.

Conclusions

- Communication protocols and OSI layered model are explained. Network, Data Link, and Physical are important for small embedded systems.
- SPI serial communication is introduced briefly.

- **Next week**:
  Communication (continued)
- **Homework**:
  Finish your projects (not for evaluation)

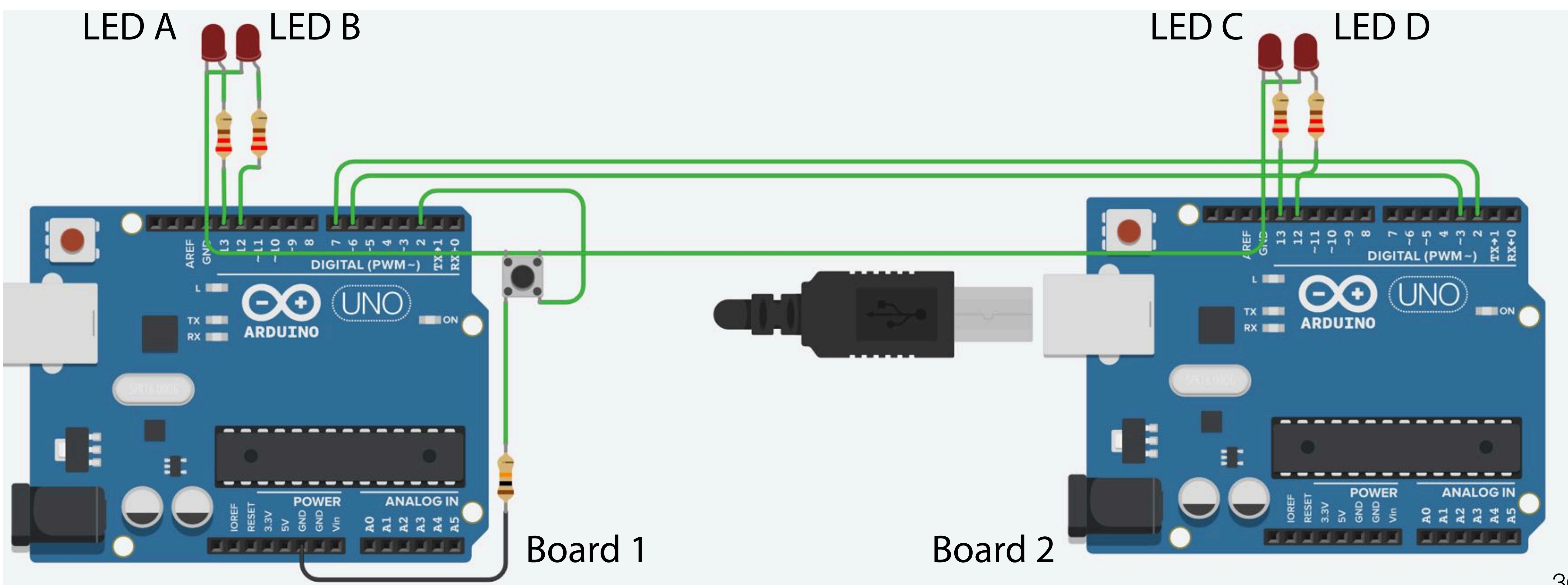
Today's Projects

- a) Simple 2-wire/1-bit unidirectional communication between two boards by using GPIO and external interrupt (will be explained)

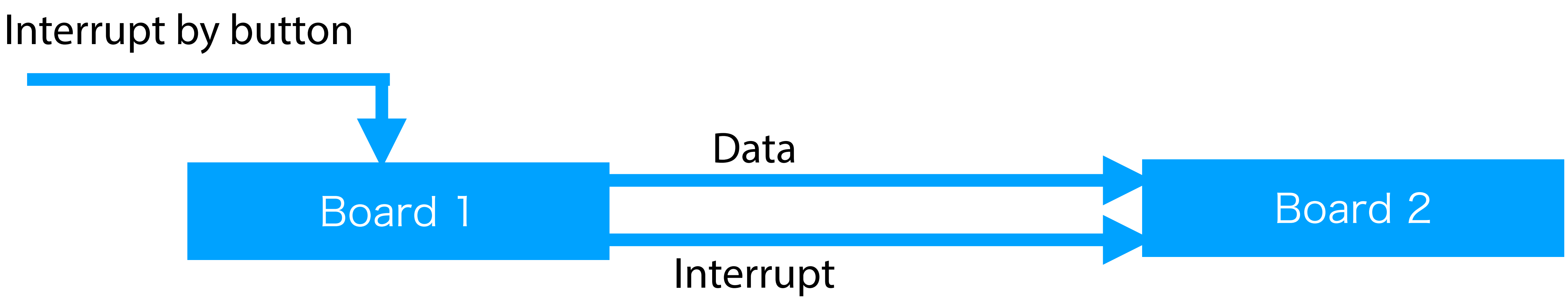
- b) Simple 3-wire bidirectional communication by using GPIO, timer, and external interrupt (do this by yourself)

a) 2-Wire Serial Communication

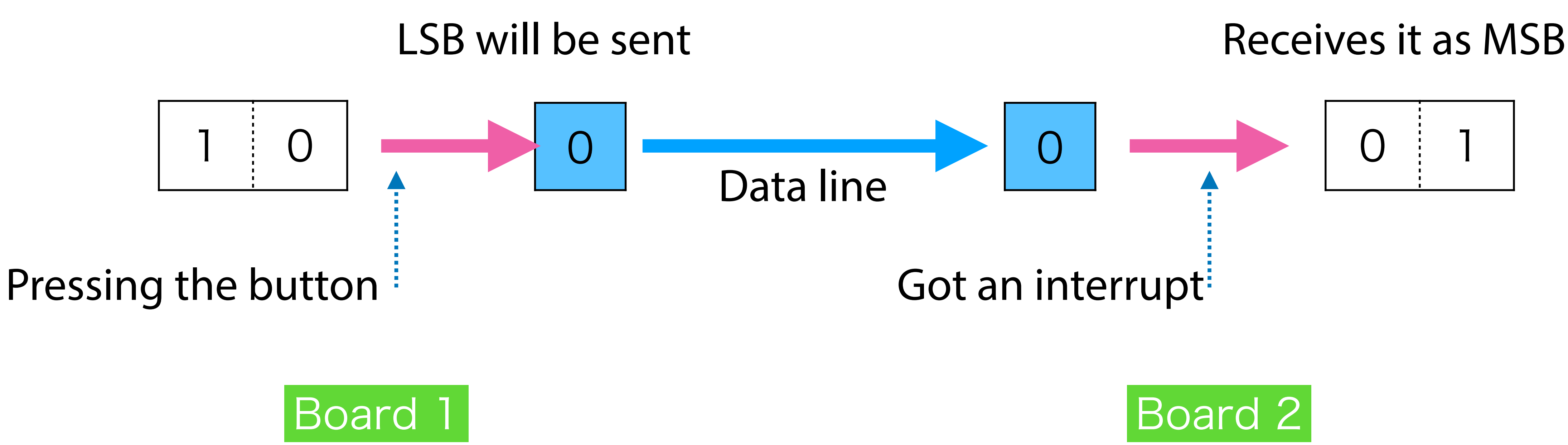
- Use two boards; one is a button + 2 LEDs based on the external interrupt example (a) in the last week, and another is one with pin 2 connected with pin 7, pin 3 connected with pin 6 on another board.
- LED A, B, C, and D must be initialized as ON,OFF,OFF,ON (1001).
- Upon pressing the button, ABCD must be changed to (0100), something like shifting the 4-bit data to the right. This must be done by sending 1-bit data from board 1 over pin6-pin3 connection. Use the pin7-pin2 connection to detect arrival of the data on board 2. After pushing the button 4 times, ABCD will be (0000).
- Do not forget to connect GND between the two boards.



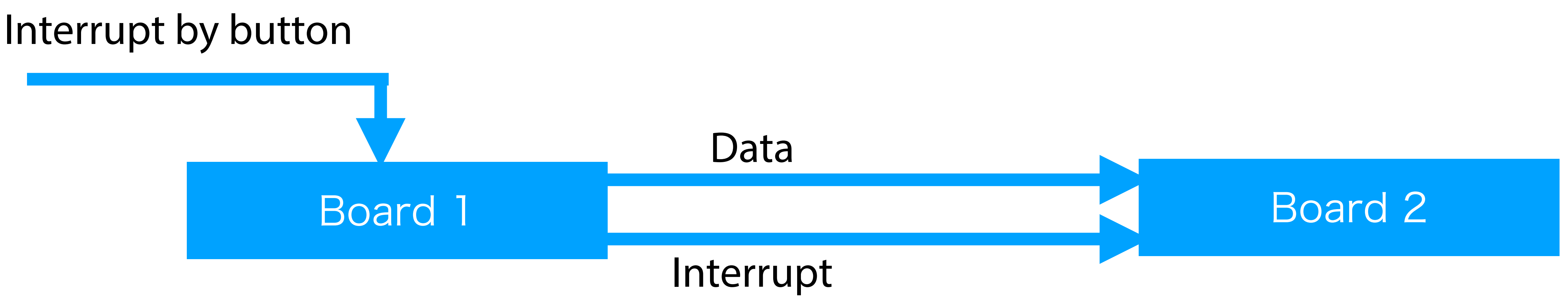
a) 2-Wire Serial Communication

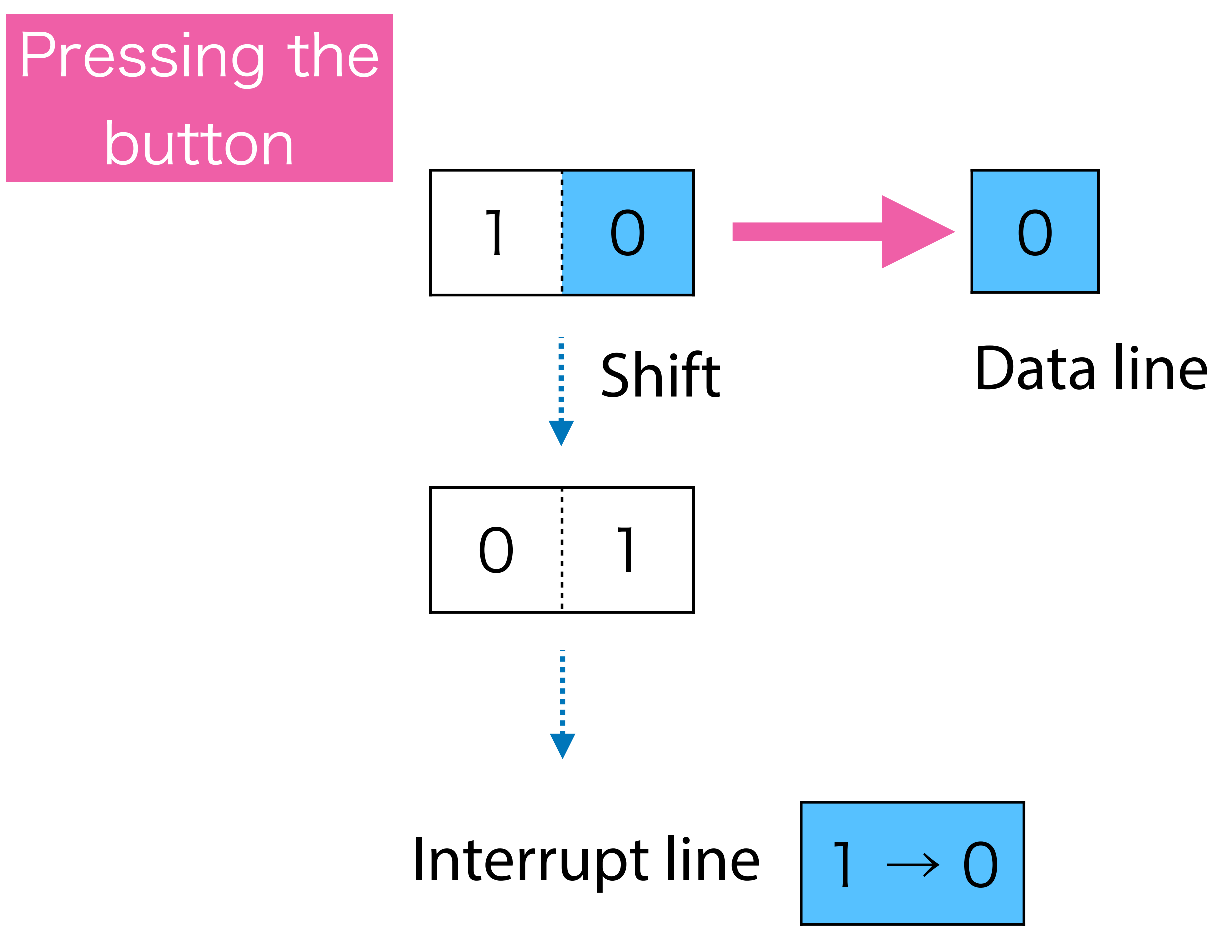


- Unidirectional 1-bit data transfer from Board 1 to 2.
- A wire for data sends the LSB. Board 2 receives it upon voltage of the interrupt line falling down.



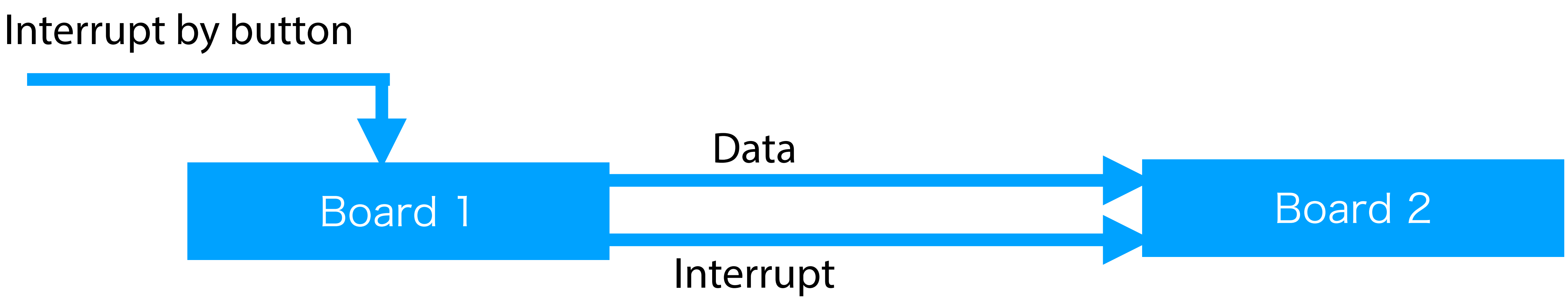
Board 1



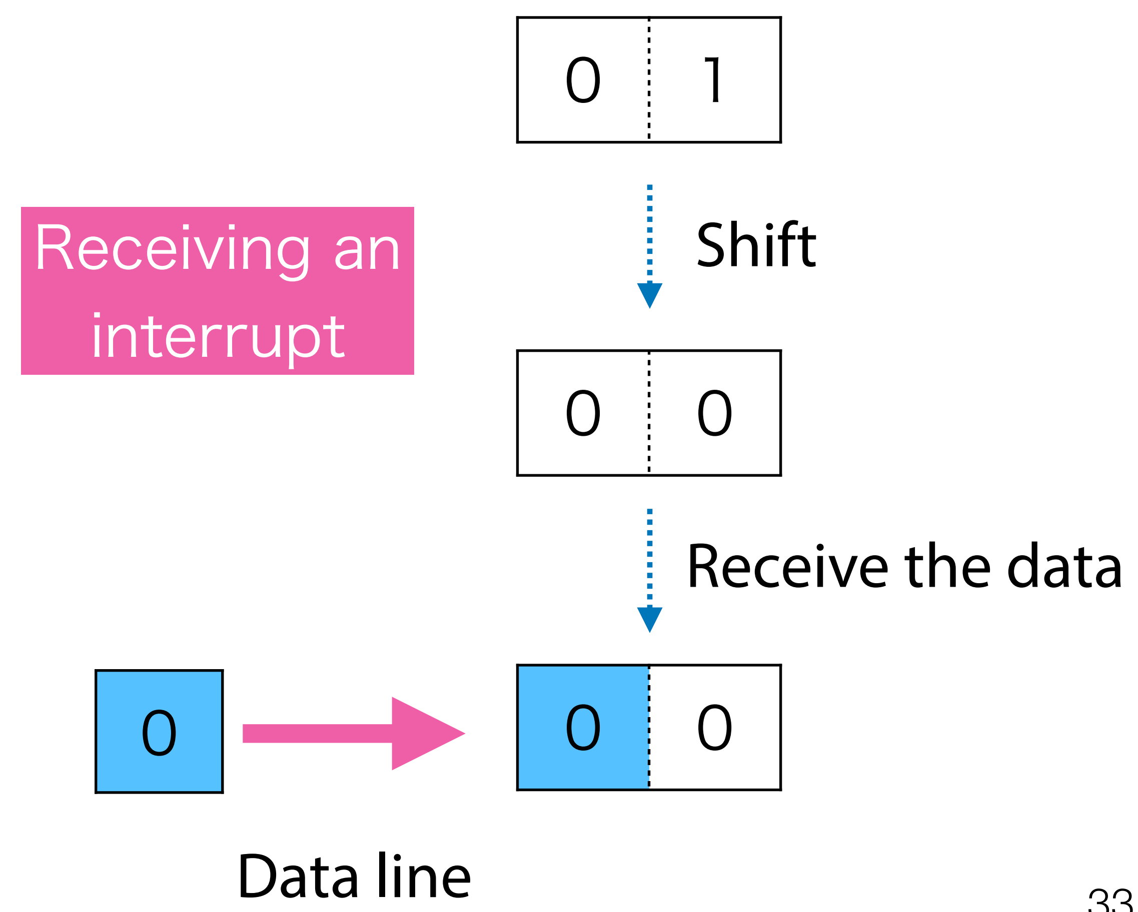


- 2-bit status must be maintained as the LED blinking pattern.
- Accepts interrupts from the button. Upon each interrupt, the status is shifted and the data line is the LSB of the previous state.
- The interrupt line must be changed from high to low when sending the data.

Board 2



- 2-bit status must be maintained as the LED blinking pattern (same as 1).
- Accepts interrupts **from the line**. Upon each interrupt, the status is shifted, the data line is read and the MSB is updated based on that.



Board 1

```
#define   A 13
#define   ASHIFT    1
#define   B 12
#define   BSHIFT    0
#define   INT_IN    2
#define   INT_OUT   7
#define   DATA_OUT  6

int led = 2; // 10b

void update_led()
{
  digitalWrite(A, (led >> ASHIFT) & 1 ? HIGH : LOW);
  digitalWrite(B, (led >> BSHIFT) & 1 ? HIGH : LOW);
}

void setup()
{
  pinMode(A, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(INT_OUT, OUTPUT); // Interrupt to board 2
  pinMode(DATA_OUT, OUTPUT); // Data to board 2
  pinMode(INT_IN, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(INT_IN),
                  on_push, FALLING);

  digitalWrite(INT_OUT, HIGH);
  digitalWrite(DATA_OUT, LOW);
  update_led();
}
```

```
void on_push()
{
  digitalWrite(DATA_OUT, (led & 1) ? HIGH : LOW);
  digitalWrite(INT_OUT, LOW);
  digitalWrite(INT_OUT, LOW); // small delay
  digitalWrite(INT_OUT, LOW); // small delay
  digitalWrite(INT_OUT, HIGH);
  led >>= 1;
  led &= 0x03;
  update_led();
}

void loop()
{
  delay(1000);
}
```

Board 1

```
#define   A 13
#define   ASHIFT    1
#define   B 12
#define   BSHIFT    0
#define   INT_IN    2
#define   INT_OUT   7
#define   DATA_OUT  6

int led = 2; // 10b
```

2-bit status

```
void update_led()
{
  digitalWrite(A, (led >> ASHIFT) & 1 ? HIGH : LOW);
  digitalWrite(B, (led >> BSHIFT) & 1 ? HIGH : LOW);
}
```

LED output function

```
void setup()
{
  pinMode(A, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(INT_OUT, OUTPUT); // Interrupt to board 2
  pinMode(DATA_OUT, OUTPUT); // Data to board 2
  pinMode(INT_IN, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(INT_IN),
                  on_push, FALLING);

  digitalWrite(INT_OUT, HIGH);
  digitalWrite(DATA_OUT, LOW);
  update_led();
}
```

```
void on_push()
{
  digitalWrite(DATA_OUT, (led & 1) ? HIGH : LOW);
  digitalWrite(INT_OUT, LOW);
  digitalWrite(INT_OUT, LOW); // small delay
  digitalWrite(INT_OUT, LOW); // small delay
  digitalWrite(INT_OUT, HIGH);
  led >>= 1;
  led &= 0x03;
  update_led();
}

void loop()
{
  delay(1000);
}
```

Board 1

```
#define  A 13
#define  ASHIFT   1
#define  B 12
#define  BSHIFT   0
#define  INT_IN   2
#define  INT_OUT  7
#define  DATA_OUT 6

int led = 2; // 10b

void update_led()
{
  digitalWrite(A, (led >> ASHIFT) & 1 ? HIGH : LOW);
  digitalWrite(B, (led >> BSHIFT) & 1 ? HIGH : LOW);
}

void setup()
{
  pinMode(A, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(INT_OUT, OUTPUT); // Interrupt to board 2
  pinMode(DATA_OUT, OUTPUT); // Data to board 2
  pinMode(INT_IN, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(INT_IN),
                  on_push, FALLING);

  digitalWrite(INT_OUT, HIGH);
  digitalWrite(DATA_OUT, LOW);
  update_led();
}
```

```
void on_push()
{
  digitalWrite(DATA_OUT, (led & 1) ? HIGH : LOW);
  digitalWrite(INT_OUT, LOW);
  digitalWrite(INT_OUT, LOW); // small delay
  digitalWrite(INT_OUT, LOW); // small delay
  digitalWrite(INT_OUT, HIGH);
  led >>= 1;
  led &= 0x03;
  update_led();
}

void loop()
{
  delay(1000);
}
```

INT_OUT and DATA_OUT are wires for communication

attachInterrupt() is used for interrupt by the button

INT_OUT should be HIGH at first

Board 1

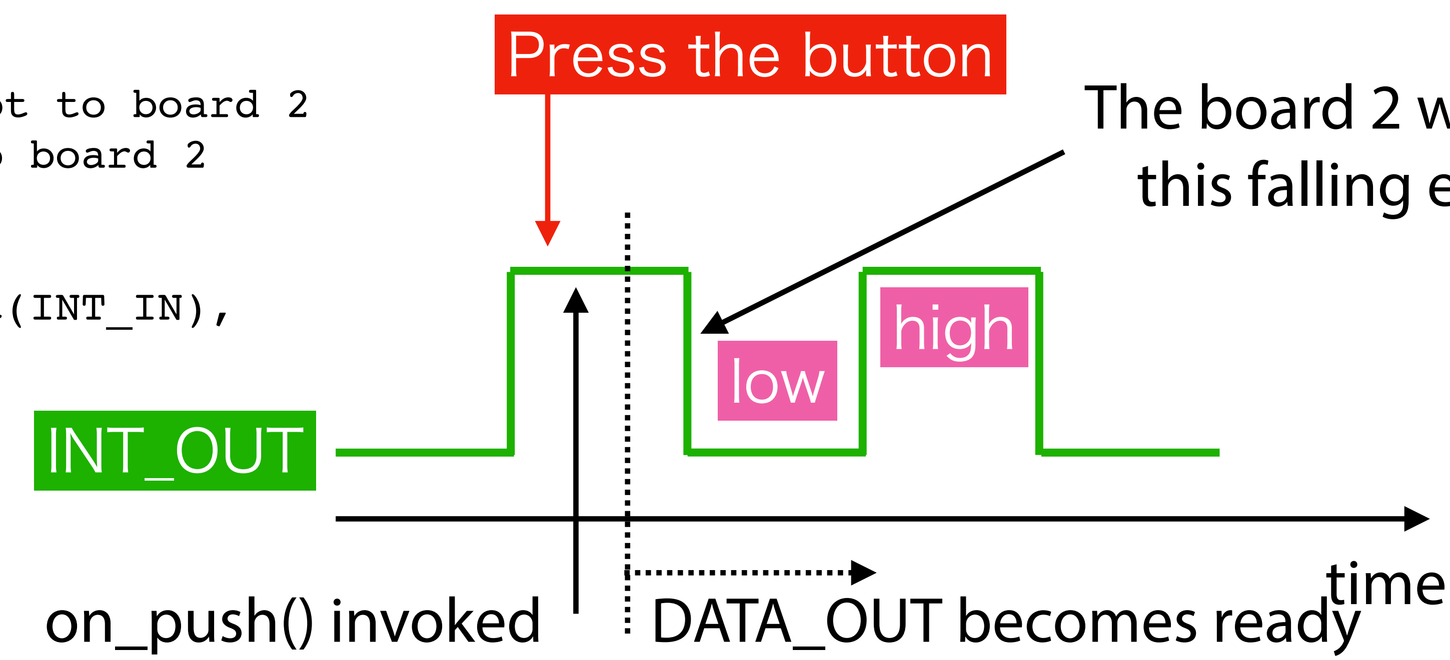
```
#define   A 13
#define   ASHIFT    1
#define   B
#define   B
#define   I
int led = 2; // 10b

void update_led()
{
  digitalWrite(A, (led >> ASHIFT)
  digitalWrite(B, (led >> BSHIFT) & 1 ? HIGH : LOW);
}

void setup()
{
  pinMode(A, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(INT_OUT, OUTPUT); // Interrupt to board 2
  pinMode(DATA_OUT, OUTPUT); // Data to board 2
  pinMode(INT_IN, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(INT_IN),
                  on_push, FALLING);

  digitalWrite(INT_OUT, HIGH);
  digitalWrite(DATA_OUT, LOW);
  update_led();
}
```

Upon pressing the button, on_push() function will be invoked

DATA_OUT is updated, and then INT_OUT will be LOW and then HIGH in a 10ms.

```
void on_push()
{
  digitalWrite(DATA_OUT, (led & 1) ? HIGH : LOW);
  digitalWrite(INT_OUT, LOW);
  digitalWrite(INT_OUT, LOW); // small delay
  digitalWrite(INT_OUT, LOW); // small delay
  digitalWrite(INT_OUT, HIGH);
  led >>= 1;
  led &= 0x03;
  update_led();
}

{
  delay(1000);
}
```

The status (led) will be updated by shifting.
Logical AND with 0x03 is required to make it in 2-bit width.



Board 2

```
#define   C 13
#define   CSHIFT    1
#define   D 12
#define   DSHIFT    0
#define   INT_IN    2
#define   DATA_IN   3

int led = 1; // 01b

void update_led()
{
  digitalWrite(C, (led >> CSHIFT) & 1 ? HIGH : LOW);
  digitalWrite(D, (led >> DSHIFT) & 1 ? HIGH : LOW);
}

void setup()
{
  pinMode(C, OUTPUT);
  pinMode(D, OUTPUT);
  pinMode(DATA_IN, INPUT_PULLUP); // Data from board 1
  pinMode(INT_IN, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(INT_IN),
                  on_data, FALLING);

  update_led();
}
```

```
void on_data()
{
  int s;

  led >>= 1;
  led &= 0x03;
  s = digitalRead(DATA_IN);
  if (s == HIGH) {
     led |= 2;
  }
  update_led();
}

void loop()
{
  delay(1000);
}
```

Board 2

```
#define   C 13
#define   CSHIFT    1
#define   D 12
#define   DSHIFT    0
#define   INT_IN    2
#define   DATA_IN   3

int led = 1; // 01b
```

2-bit status (same)

```
void update_led()
{
  digitalWrite(C, (led >> CSHIFT) & 1 ? HIGH : LOW);
  digitalWrite(D, (led >> DSHIFT) & 1 ? HIGH : LOW);
}
```

LED output function (same)

```
void setup()
{
  pinMode(C, OUTPUT);
  pinMode(D, OUTPUT);
  pinMode(DATA_IN, INPUT_PULLUP); // Data from board 1
  pinMode(INT_IN, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(INT_IN),
                  on_data, FALLING);

  update_led();
}

void on_data()
{
  int s;

  led >>= 1;
  led &= 0x03;
  s = digitalRead(DATA_IN);
  if (s == HIGH) {
     led |= 2;
  }
  update_led();
}

void loop()
{
  delay(1000);
}
```

Board 2

```
#define   C 13
#define   CSHIFT    1
#define   D 12
#define   DSHIFT    0
#define   INT_IN    2
#define   DATA_IN   3

int led = 1; // 01b

void update_led()
{
  digitalWrite(C, (led >> CSHIFT) & 1 ? HIGH : LOW);
  digitalWrite(D, (led >> DSHIFT) & 1 ? HIGH : LOW);
}

void setup()
{
  pinMode(C, OUTPUT);
  pinMode(D, OUTPUT);
  pinMode(DATA_IN, INPUT_PULLUP); // Data from board 1
  pinMode(INT_IN, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(INT_IN),
                  on_data, FALLING);

  update_led();
}
```

```
void on_data()
{
  int s;

  led >>= 1;
  led &= 0x03;
  s = digitalRead(DATA_IN);
  if (s == HIGH) {
     led |= 2;
  }
  update_led();
}

void loop()
{
  delay(1000);
}
```

INT_IN and DATA_IN are wires for communication

attachInterrupt() is used for interrupt **by the int line**

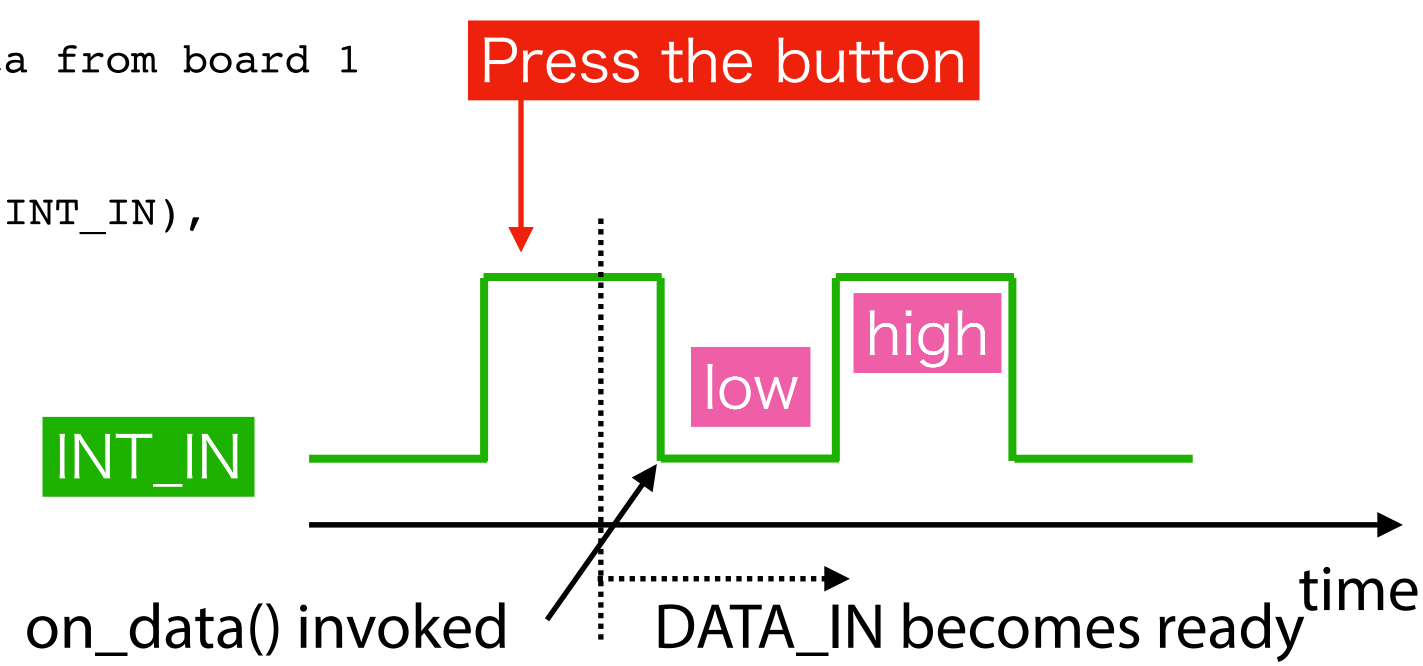
Board 2

```
#define   C 13
#define   C
#define   D
#define   D
#define   INT_IN     2
#define   DATA_IN    3

int le

void update_led()
{
  digitalWrite(C, (led >> CSHIFT) & 1 ? HIGH : LOW);
  digitalWrite(D, (led >> DSHIFT) & 1 ? HIGH : LOW);
}

void setup()
{
  pinMode(C, OUTPUT);
  pinMode(D, OUTPUT);
  pinMode(DATA_IN, INPUT_PULLUP); // Data from board 1
  pinMode(INT_IN, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(INT_IN),
                  on_data, FALLING);

  update_led();
}
```

Upon arrival of an interrupt, on_data() function will be invoked

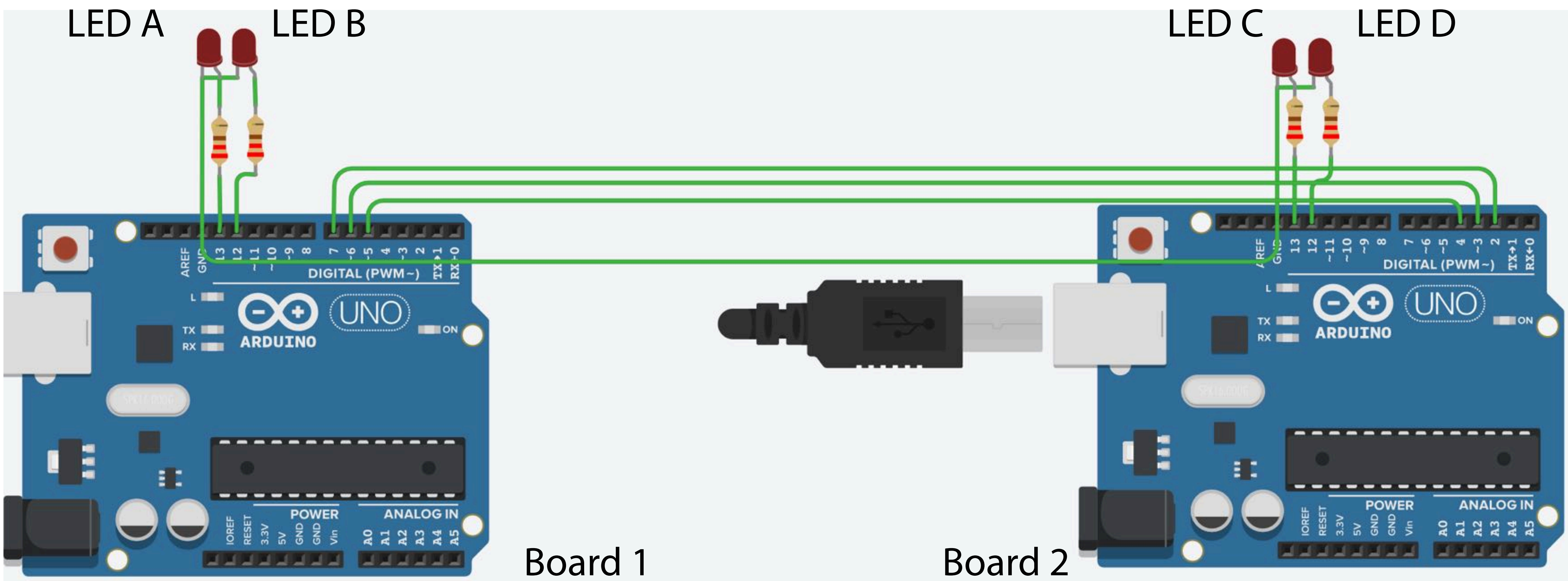
Read DATA_IN, and if it is HIGH, MSB of led is updated.

```
void on_data()
{
  int s;

  led >>= 1;
  led &= 0x03;
  s = digitalRead(DATA_IN);
  if (s == HIGH) {
     led |= 2;
  }
  update_led();
}

void loop()
{
  delay(1000);
}
```



b)

- Based on a), make 4-bit LED blinking pattern be shifted with 1s interval. Remove the button. The cycle is (1001), (1100), (0110), (0011).
- This must be done by bidirectional serial communication from board 1 to board 2 and vice versa. Use pin7-pin2 to send a clock from board 1 to board 2, and make the two boards to communicate. Clock can be generated by timer.
- Use pin6-pin3 for data transfer from board 1 to board 2, and pin5-pin4 for transfer from board 2 to board 1.



Conclusions and Time for Your Project

- Feel free to discuss with your friends
- If you have a question, ask the teaching assistant or just speak up.

- **Next week**:
  Communication between two processors (or more)
  Do not forget to submit your design before the deadline