Introduction to Design (2)

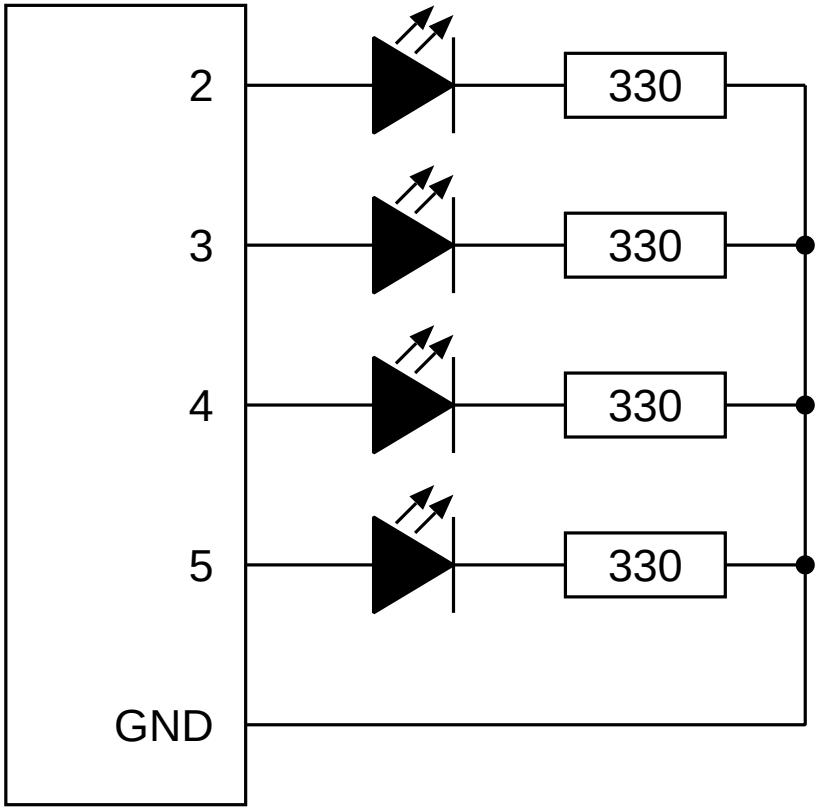
Microcontrollers and Interfacing

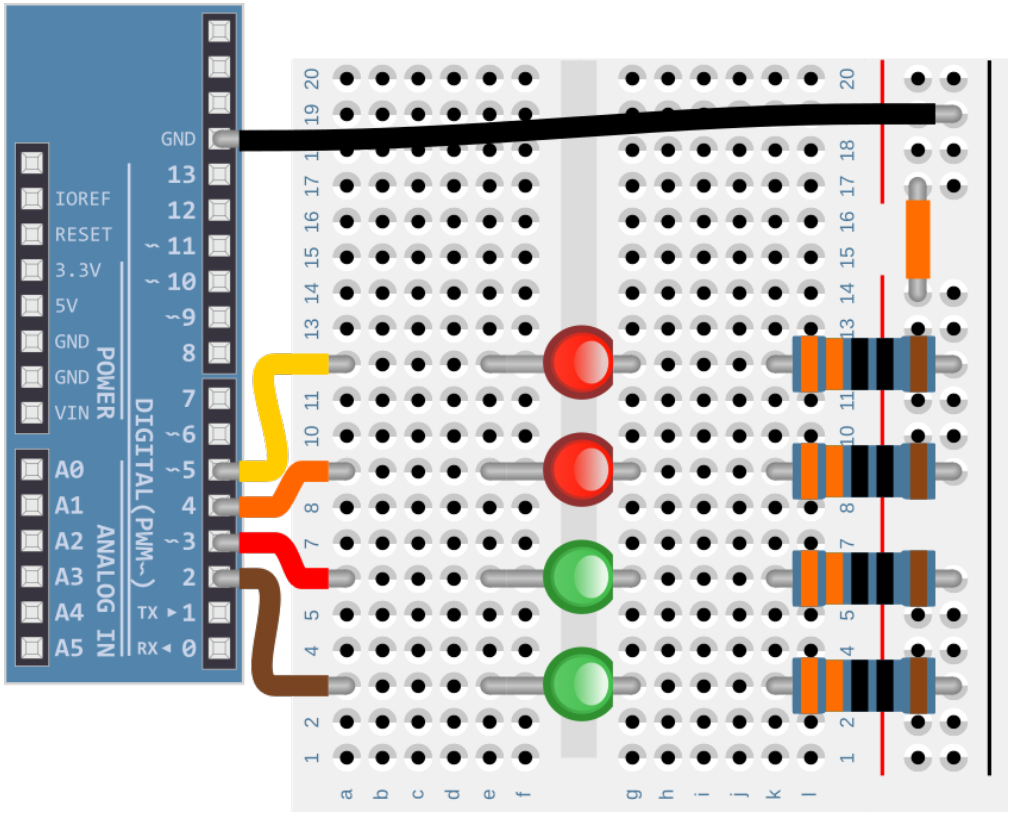
Week 07

Managing multiple outputs

KUAS 京都先端科学大学
KYOTO UNIVERSITY of ADVANCED SCIENCE
Department of Mechanical and Electrical System Engineering

connecting several leds





connect up to 20 LEDs (or other digital devices)

- pins 0 to 13 give 14 outputs, plus six analogue pins can be digital outputs too
- beware of pins 0 and 1, which are used for serial communication
- pins 2–13 are therefore recommended for digital outputs

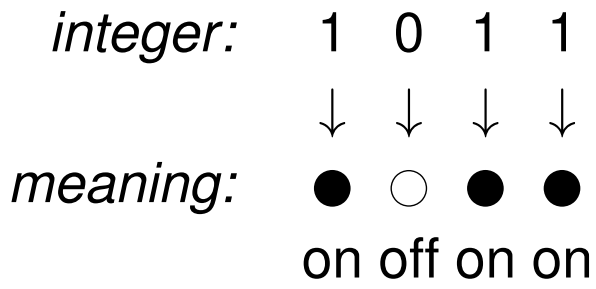
represent all LED states as a single integer

writing one output at a time is tedious

- desired effect is buried inside lots of `digitalWite()`s

why not use an `int` to represent many LED states?

- one integer is made of many digits
- use each digit to store the state of one LED



each LED only has two states: on or off

- only need two digits, e.g., `0` and `1`
- can use binary (base 2) instead of decimal (base 10)
- binary is how the computer stores integers internally

decimal and binary numbers

decimal numbers (base 10)

- the rightmost column has weight $10^0 = 1$ (‘units’ column)
- weight increases $10\times$ per column towards the left

	10^3		10^2		10^1		10^0	
	1000		100		10		1	
	$\times$		$\times$		$\times$		$\times$	
1011_{10} =	1	+	0	+	10	+	1	
↑ ↑	1000	+	0	+	10	+	1	= 1011

digits *base*

binary numbers (base 2)

- the rightmost column has weight $2^0 = 1$ (‘units’ column)
- weight increases $2\times$ per column towards the left

	2^3		2^2		2^1		2^0	
	8		4		2		1	
	$\times$		$\times$		$\times$		$\times$	
1101_2 =	1	+	1	+	0	+	1	
	8	+	4	+	0	+	1	= 13

binary	*decimal*
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

testing the bits in an integer

on our microcontroller, an `int`eger contains 16 binary digits (bits)

- an `int` can represent the state of up to 16 digital outputs

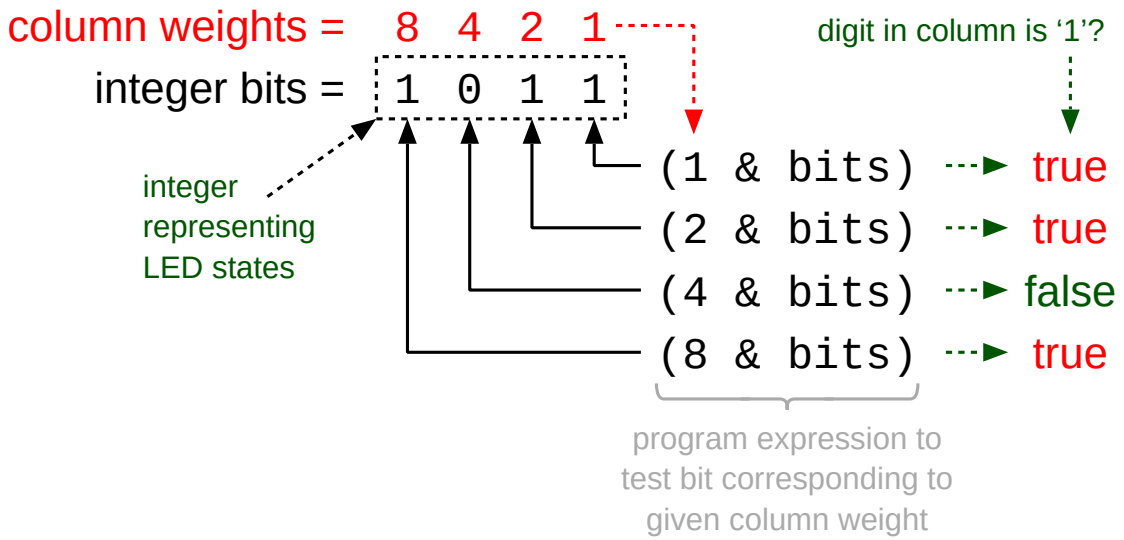
each bit in the integer corresponds to one output pin

- its value is either `0` or `1`

the bits (digit columns) in the integer have power-of-two weights

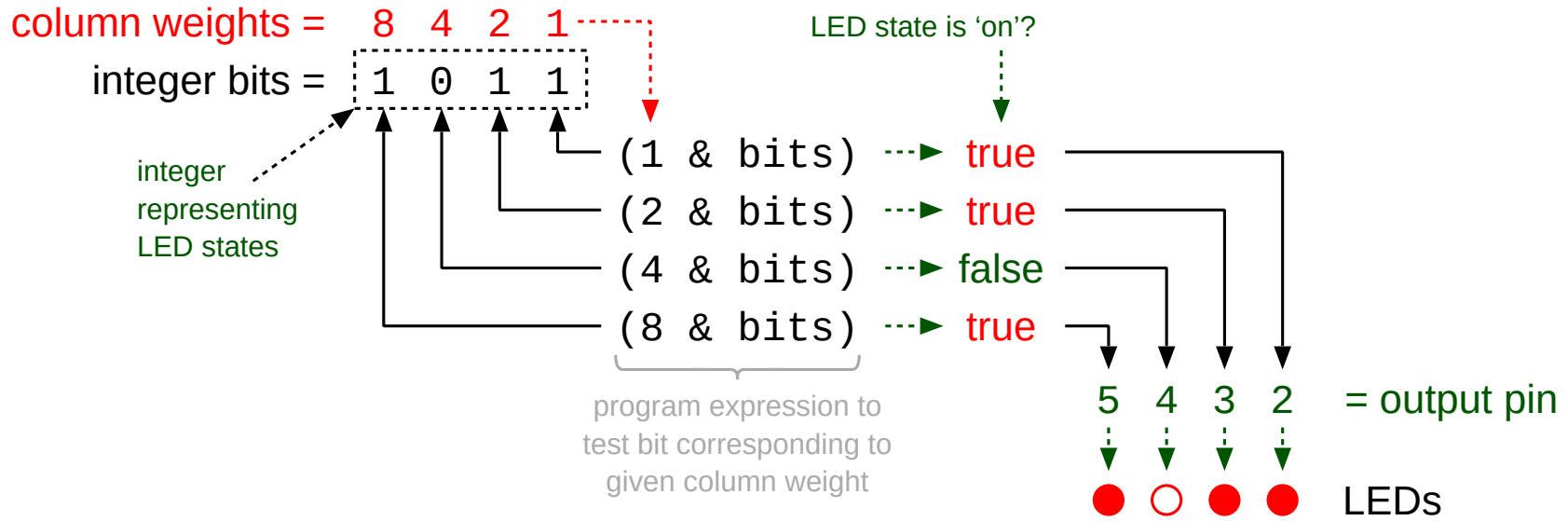
- 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768

use the `&` (‘and’) operator with a column weight to test if the corresponding bit is `1`



setting LEDs according to the bits in an integer

test each bit in an integer, set LED pin to HIGH or LOW accordingly




```
void setLEDs(int bits) {
  //          ⌜ bit weight     ⌜ corresponding output pin ⌝
  if (bits & 1) digitalWrite(2, HIGH); else digitalWrite(2, LOW);
  if (bits & 2) digitalWrite(3, HIGH); else digitalWrite(3, LOW);
  if (bits & 4) digitalWrite(4, HIGH); else digitalWrite(4, LOW);
  if (bits & 8) digitalWrite(5, HIGH); else digitalWrite(5, LOW);
  // similarly for column weights 16, 32, 64, 128, etc.
}
```

e.g: $12_{10} = 1100_2$ represents pins 5 and 4 being `HIGH`, and pins 3 and 2 being `LOW`