임베디드컴퓨팅
Embedded Computing
(0009488)
C/C++ in Arduino

2022년 2학기

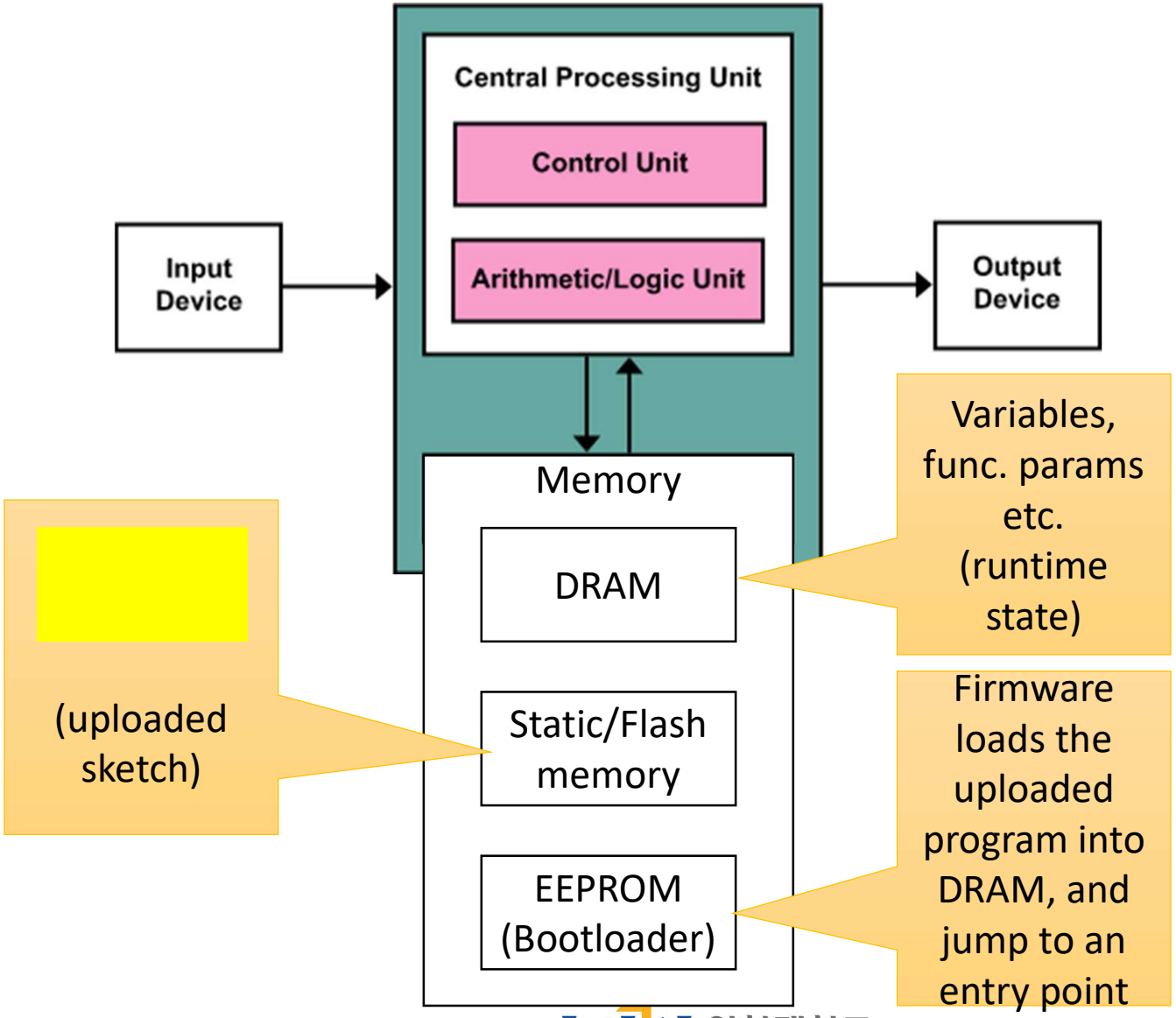
정보기술대학 정보통신공학과
김 영 필
ypkim@inu.ac.kr

인천대학교
INCHEON NATIONAL UNIVERSITY

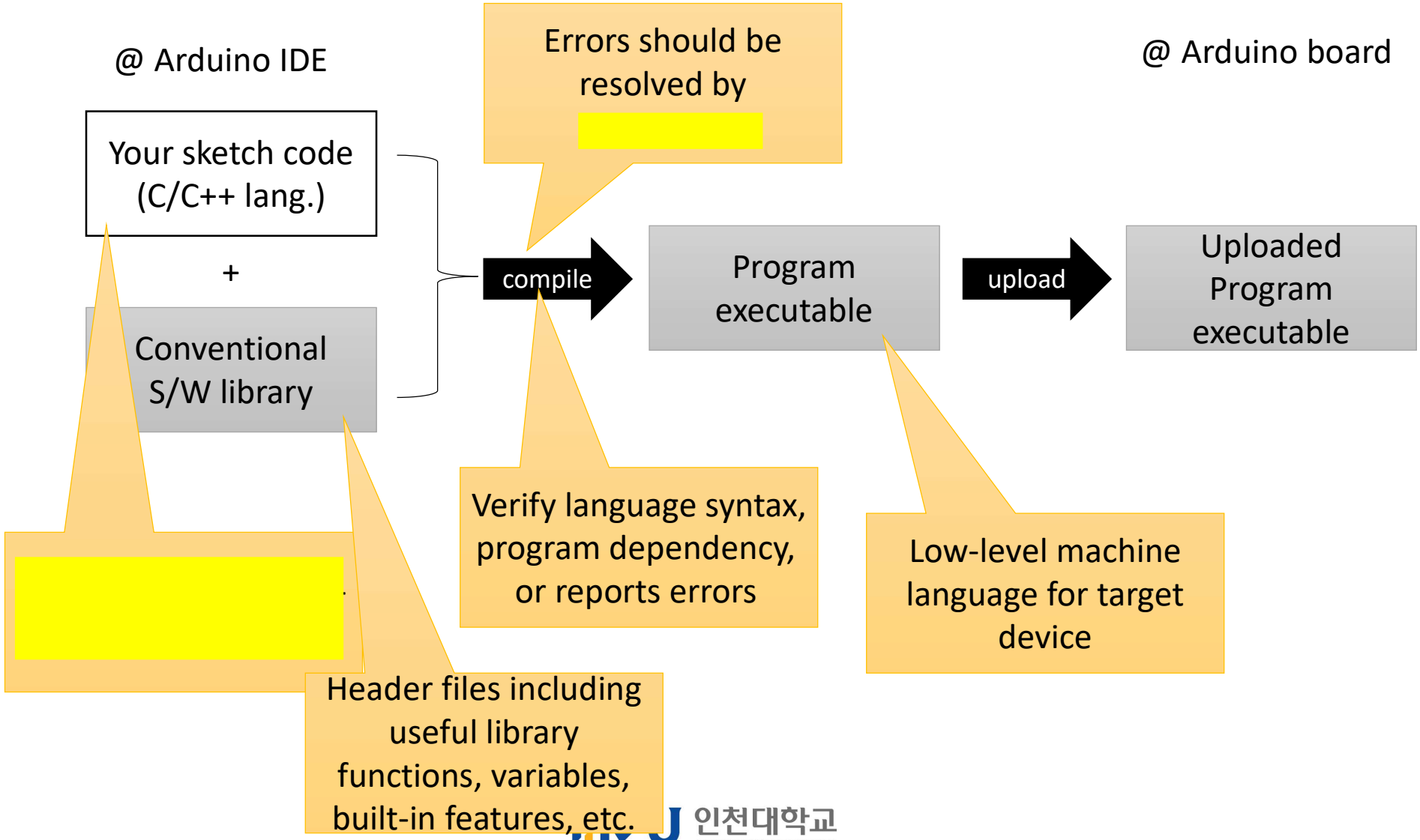
Before you get it started

- [redacted] is essential for controlling/operating Arduino board
- Arduino uses a high-level programming language following **C/C++ style**.
- **A source code** is written by [redacted]
  - [redacted] verifies the source code and builds a binary program by translating it to [redacted]
  - **A valid sketch program** can be **uploaded** into Arduino.
- Syntax of PL determines correctness and validity of source code in terms of structural point.
- In this class, we focus on basic syntax, variables, functions, and control statements

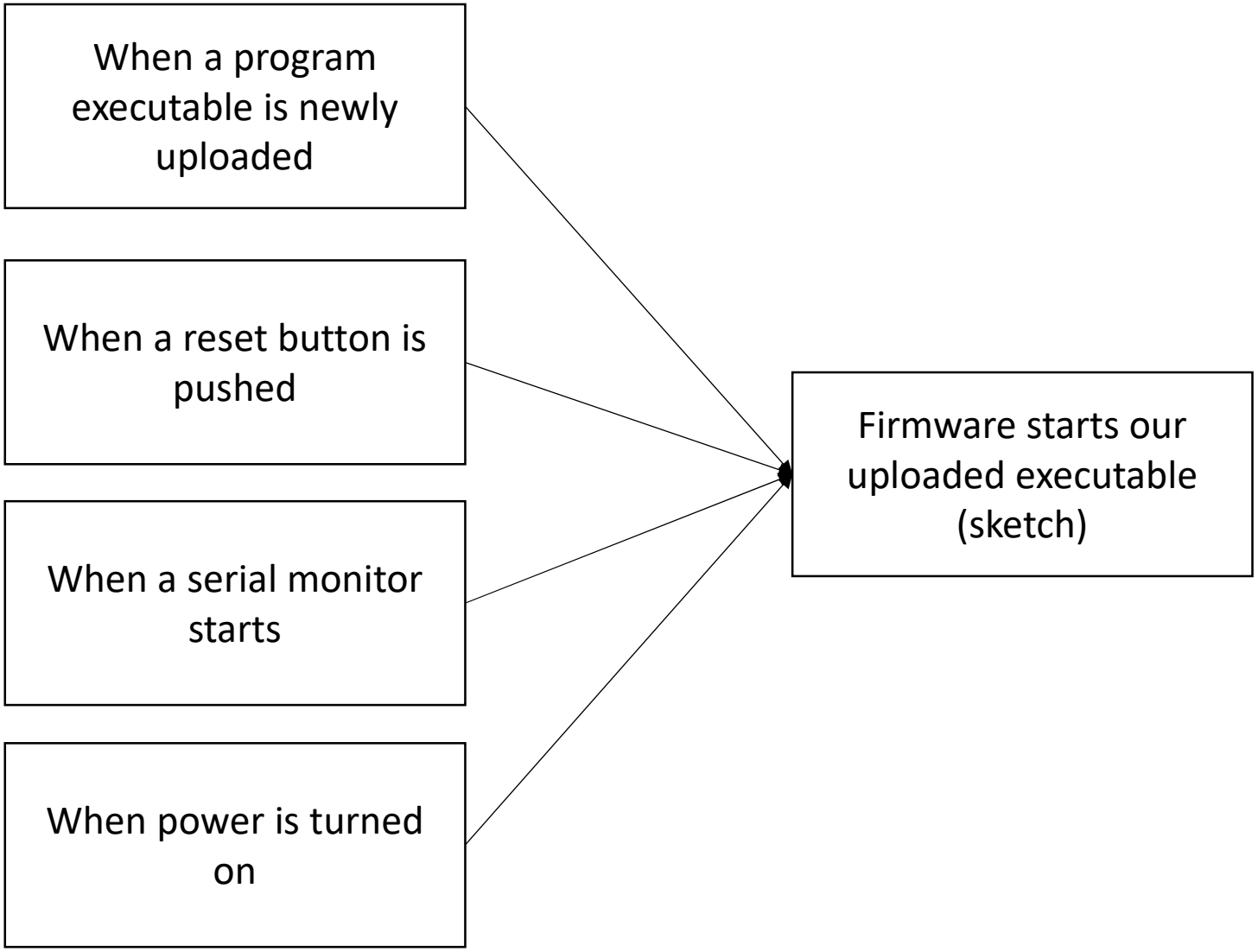
Where is our program stored?



How to store our program?



When our program begin to run?



Common mistakes on developing Arduino

Connection error between a computer and Arduino board

Electronic component wiring errors

Programming compiling errors
(sytax errors, mis-configuration, etc.)

Programming logic errors (semantic errors, design fault, etc.)

Communication control error, Analog/Digital serial communication error

Basic syntax for Arduino programming language

How to put comment?

- Comment
  - Code lines in the program that are used to inform yourself or others about the way the program works
  - Ignored by the compiler

- comment (/* … */)
  - Multiple lines comment

- comment (//)

Example Code

```
/* This is a valid comment */

/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
  (Another valid comment)
*/

/*
  if (gwb == 0) { // single line comment is OK inside a multi-line comment
    x = 3;          /* but not another multi-line comment - this is invalid */
  }
// don't forget the "closing" comment - they have to be balanced!
*/
```

https://www.arduino.cc/reference/en/language/structure/further-syntax/blockcomment/

How to express integer constant?

- Integer can be expressed in various ways
  -
- It can also be expressed as an unsigned integer or a 4-byte integer (long).

Integer constant

https://www.arduino.cc/reference/en/language/variables/constants/integerconstants/

BASE	EXAMPLE	FORMATTER	COMMENT
1	123	none	
2	0b1111011	leading "0b"	characters 0&1 valid
8	0173	leading "0"	characters 0-7 valid
16	0x7B	leading "0x"	characters 0-9, A-F, a-f valid

How to express floating point constant?

- Floating point numbers are written as 1.23
- Expressed in a variety of scientific notation.
  - [redacted] are both accepted as valid exponent indicators.

Floating Point constant

https://www.arduino.cc/reference/en/language/variables/constants/floatingpointconstants/

FLOATING-POINT CONSTANT	EVALUATES TO:	ALSO EVALUATES TO:
10.0	10	
2.34E5	2.34 * 10^5	234000
67e-12	67.0 * 10^-12	0.000000000067

How to declare data type?

- Conceptually, data type is about
  - String, Character, Numeric values (integer, floating point, etc.), Boolean, Void, Memory address, or newly defined data
  - Signed numeric value or Unsigned numeric value
  - Variables, function arguments, function return value, etc.
- Data type also determines the to store the data value
- Data type should be declared before using it
  - E.g. int datatype syntax ➔

Syntax

```
int var = val;
```

Parameters

var: variable name.

val: the value you assign to that variable.

https://www.arduino.cc/reference/en/language/variables/data-types/int/

Basic datatypes and capacity

- **boolean (8 bit) -** simple logical true/false
- **byte (8 bit) or unsigned char (8 bit) -** unsigned number from 0-255
- **char (8 bit) -** signed number from -128 to 127.
- **word (16 bit) or unsigned int (16 bit) -** unsigned number from 0-65535
- **int** - signed number from -32768 to 32767
- **unsigned long (32 bit) -** unsigned number from 0-4,294,967,295. The result of the millis() is stored this type
- **long (32 bit) -** signed number from -2,147,483,648 to 2,147,483,647
- **float** - signed number from -3.4028235E38 to 3.4028235E38. Floating point on the Arduino is not native; the compiler has to treat additionally

String and character

- Text strings can be represented in two ways.
  - String data type (String object)
  - and null-terminate it (C style)
- Syntax
  - char Str1[15];
  - char Str2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
  - char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '₩0'};
  - char Str4[] = "arduino";
  - char Str5[8] = "arduino";
  - char Str6[15] = "arduino";

String object

- An instance of **String class**
  - Support various methods, operators related to String
- **Syntax**
  - String(val)
  - String(val, base)
  - String(val, decimalPlaces)
- **Parameters**
  - **val:** a variable to format as a String. Allowed data types: string, char, byte, int, long, unsigned int, unsigned long, float, double.
  - **base:** (optional) the base in which to format an integral value.
  - **decimalPlaces:** only if val is float or double. The desired decimal places.
- **Returns**
  - [redacted] of the String class.

```
String stringOne = "Hello String";
String stringOne = String('a');
String stringTwo = String("This is a string");
String stringOne = String(stringTwo + " with more");
String stringOne = String(13);
String stringOne = String(analogRead(0), DEC);
String stringOne = String(45, HEX);
String stringOne = String(255, BIN);
String stringOne = String(millis(), DEC);
String stringOne = String(5.698, 3);
```

https://www.arduino.cc/reference/en/language/variables/data-types/stringobject/

Identifier and keyword

- **Identifiers**
  - user-written variable or function **names**.
  - **Rule of the first character of the identifier**
    - must be an
  - Valid e.g. : zzzz, x200, x_2, _3
  - Invalid e.g. : ‘x02 (begin with special char.), 0xyz (begin with number), x*2d (including special char.), int (keyword)
-
  - a reserved string, a string declared in advance
  - cannot be used as identifiers
  - written in the keywords.txt file in the lib folder of the Arduino folder.
  - In addition, declared in additional libraries that are called when using LCD or SD memory cards.

Eg. Control structure keyword

- break
- continue
- do…while
- else
- for
- goto
- if
- return
- switch…case
- while

These are fundamental keywords of C lang.

See reference doc whenever you confuse it

https://www.arduino.cc/reference/en/

Eg. Math function name keyword

- abs()
- constrain()
- map()
- max()
- min()
- pow()
- sq()
- sqrt()

These are built-in library function names.

See reference doc whenever you need it

https://www.arduino.cc/reference/en/

E.g. Datatype keyword

- bool
- boolean
- byte
- char
- double
- float
- int
- long
- short
- size_t
- string
- String()
- unsigned char
- unsigned int
- unsigned long
- void
- word

These are fundamental keywords of C lang.

sizeof() is useful to check its capacity (byte length).

Arithmetic expression

- Numeric operations requires expression
- Expression =
- Arduino supports various operators
  - Arithmetic operators, relational operators, logical operators, assignment operators, bitwise operators, square operators, etc.

Arithmetic Operators

- % (remainder)
- * (multiplication)
- + (addition)
- - (subtraction)
- / (division)
- = (assignment operator)

Bitwise Operators

- & (bitwise and)
- << (bitshift left)
- >> (bitshift right)
- ^ (bitwise xor)
- | (bitwise or)
- ~ (bitwise not)

How can we check the bit result?

One of the easiest ways is

Compound Operators

- %= (compound remainder)
- &= (compound bitwise and)
- *= (compound multiplication)
- += (compound addition)
- -= (compound subtraction)
- /= (compound division)
- ^= (compound bitwise xor)
- |= (compound bitwise or)
- ++ (increment)
- -- (decrement)

X = X <opr> Y

X <opr>= Y

Z = X++ vs. Z = X--

Z = ++X vs. Z = --X

Comparison Operators

- != (not equal to)
- < (less than)
- <= (less than or equal to)
- == (equal to)
- > (greater than)
- >= (greater than or equal to)

What is the result of these operations?

can evaluate this!

Boolean Operators

- ! (logical not)
- && (logical and)
- || (logical or)

What is the result of these operations?

can evaluate this!