임베디드컴퓨팅
Embedded Computing
(0009488)
Sensor applications, appendix

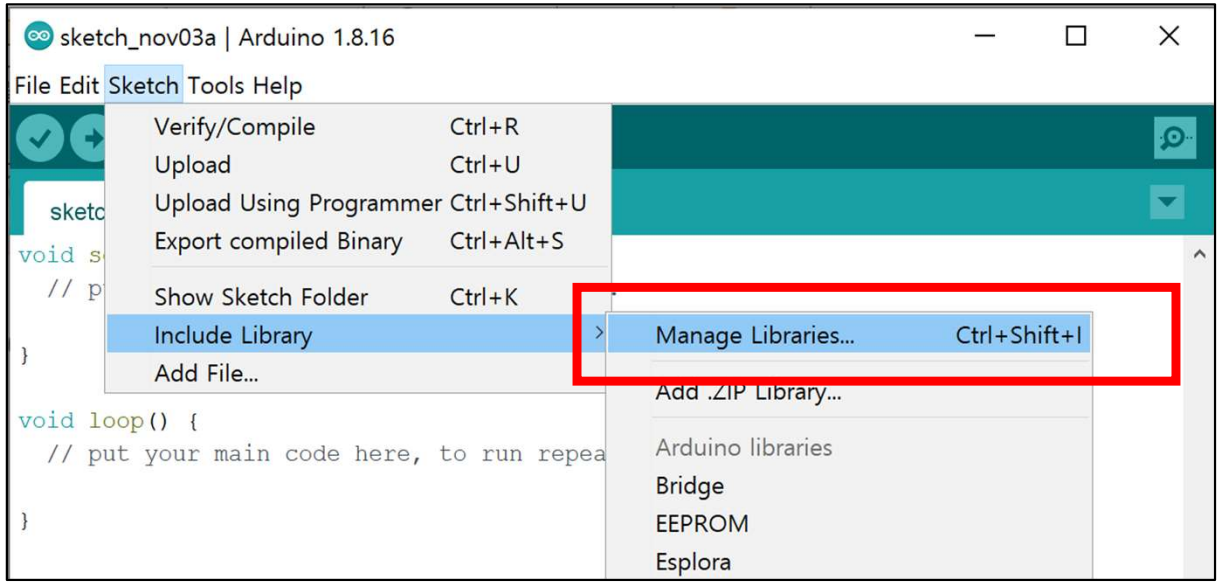
2022년 2학기

정보기술대학 정보통신공학과
김 영 필
ypkim@inu.ac.kr

INU 인천대학교
INCHEON NATIONAL UNIVERSITY

Library in Arduino

- To use a library in a sketch, select it from [Sketch] > [Import Library].



- You can download or create your own library.
  - To write your own libraries, see the tutorial and API Style Guide for making a good Arduino-style API for your library.
    - https://www.arduino.cc/en/Hacking/LibraryTutorial
    - https://www.arduino.cc/en/Reference/APIStyleGuide

Arduino Style Guide for Writing Libraries (1)

- Reference: Arduino API style guide
  - https://www.arduino.cc/en/Reference/APIStyleGuide
- Be kind to the
  - for an intelligent person who has not programmed before.
- Match your to the underlying capabilities.
  - hide unnecessary details; expose exactly needed only.
- Organize your public functions around the data and functionality that the user wants.
  - Think about what the average person thinks the thing does, and try to organise your API functions around that.
- Use
  - Avoid technically specialized terms; instead, use generally accepted ones.
- Avoid words that have to the general public.
  - e.g. Errors

Arduino Style Guide for Writing Libraries (2)

- When you have to use a [redacted], write a sentence or two describing it to the general public FIRST.
- Document and comment as you go.
- Use the established core libraries and styles.
  - e.g. for each usage, use Arduino built-in functions such as digitalRead(), analogWrite()
- Use [redacted] function names, not underscore.
  - e.g. readTemperature(), digitalWrite()
  - LONG_CONSTANT_NAMES_FULL_OF_CAPS are hard to read.
- Try to avoid boolean arguments.
  - Consider providing two different functions with meaningful names
- Don’t assume knowledge of pointers.
  - Provide method to pass/get data

E.g.: Turn your sketch to a library

- From simple morse code to a library
  - https://www.arduino.cc/en/Hacking/LibraryTutorial
- At least two files for a library
  -
    - Definitions for the library
  -
    - Actual code.

```
int pin = 13;

void setup()
{
  pinMode(pin, OUTPUT);
}

void loop()
{
  dot(); dot(); dot();
  dash(); dash(); dash();
  dot(); dot(); dot();
  delay(3000);
}

void dot()
{
  digitalWrite(pin, HIGH);
  delay(250);
  digitalWrite(pin, LOW);
  delay(250);
}

void dash()
{
  digitalWrite(pin, HIGH);
  delay(1000);
  digitalWrite(pin, LOW);
  delay(250);
}
```

In Header file,

- Class definition

```
class Morse
{
  public:
    Morse(int pin);
    void dot();
    void dash();
  private:
    int _pin;
};
```

to use built-in Arduino functions

```
#include "Arduino.h"
```

to avoid including your header twice or more

```
#ifndef Morse_h
#define Morse_h

// the #include statment and code go here...

#endif
```

put comment at the top:
its name, a short description of what it does, who wrote it, the date, and the license.

```
/*
  Morse.h - Library for flashing Morse code.
  Created by David A. Mellis, November 2, 2007.
  Released into the public domain.
*/
```

In Source file,

Write a

```
Morse::Morse(int pin)
{
  pinMode(pin, OUTPUT);
  _pin = pin;
}
```

Include at the top

```
/*
  Morse.cpp - Library for flashing Morse code.
  Created by David A. Mellis, November 2, 2007.
  Released into the public domain.
*/
```

Implement methods

```
void Morse::dot()
{
  digitalWrite(_pin, HIGH);
  delay(250);
  digitalWrite(_pin, LOW);
  delay(250);
}

void Morse::dash()
{
  digitalWrite(_pin, HIGH);
  delay(1000);
  digitalWrite(_pin, LOW);
  delay(250);
}
```

Simple Morse library by David A. Mellis.

```
/*
  Morse.h - Library for flashing Morse code.
  Created by David A. Mellis, November 2, 2007.
  Released into the public domain.
*/
#ifndef Morse_h
#define Morse_h

#include "Arduino.h"

class Morse
{
  public:
    Morse(int pin);
    void dot();
    void dash();
  private:
    int _pin;
};

#endif
```

```
/*
  Morse.cpp - Library for flashing Morse code.
  Created by David A. Mellis, November 2, 2007.
  Released into the public domain.
*/

#include "Arduino.h"
#include "Morse.h"

Morse::Morse(int pin)
{
  pinMode(pin, OUTPUT);
  _pin = pin;
}

void Morse::dot()
{
  digitalWrite(_pin, HIGH);
  delay(250);
  digitalWrite(_pin, LOW);
  delay(250);
}

void Morse::dash()
{
  digitalWrite(_pin, HIGH);
  delay(1000);
  digitalWrite(_pin, LOW);
  delay(250);
}
```

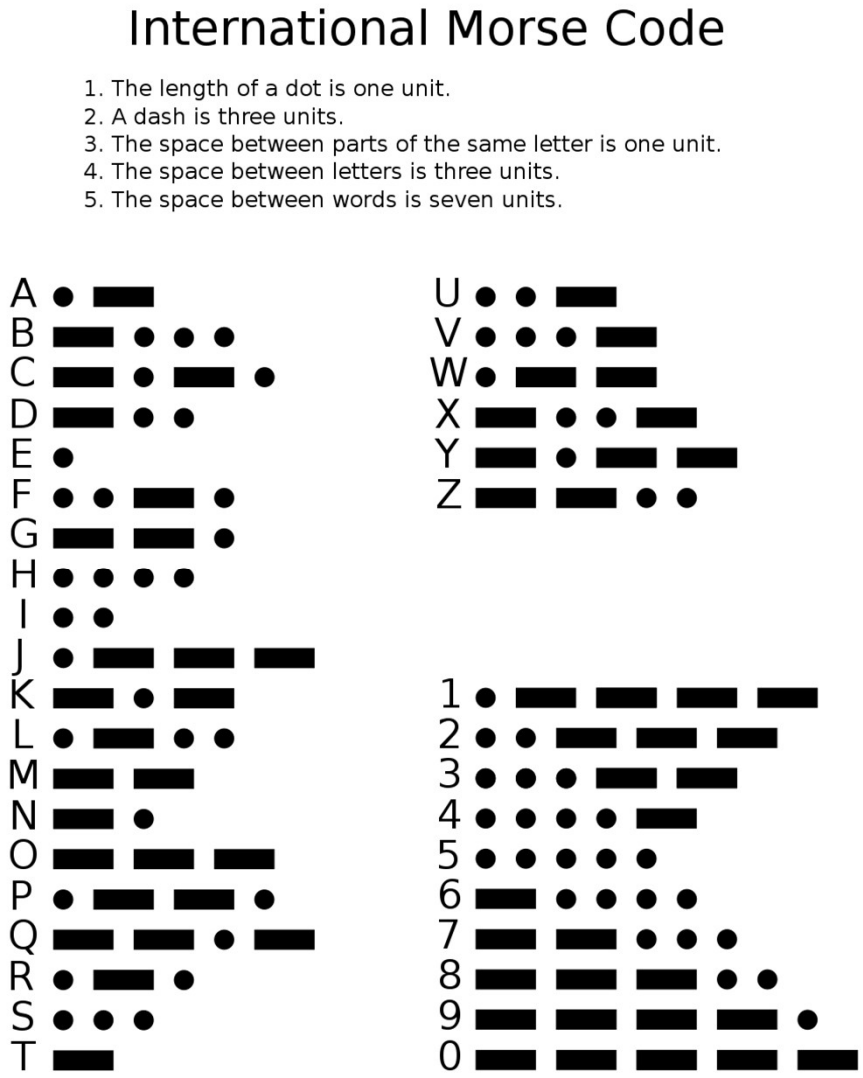
How to use it?

- First, make a Morse directory inside of the [redacted] of your sketchbook directory.
  - Copy or move the Morse.h and Morse.cpp files into that directory.
- Now launch the Arduino environment.
- If you open the Sketch > Import Library menu, you should see Morse inside.
- The library will be compiled with sketches that use it.
- If the library doesn't seem to build, make sure that the files really end in .cpp and .h (with no extra .pde or .txt extension, for example).

```
#include <Morse.h>

Morse morse(13);

void setup()
{
}

void loop()
{
  morse.dot(); morse.dot(); morse.dot();
  morse.dash(); morse.dash(); morse.dash();
  morse.dot(); morse.dot(); morse.dot();
  delay(3000);
}
```

Assignment: Custom Morse lib.

- Requirements
  - Modify Morse library as follows
  - Add **toChar(char ch)**
    - Translate an alphabet code (A-Z) to a Morse code
  - Add **toString(String str)**
    - Translate an alphabet String to a sequence of Morse code
  - Display your alphabet full name in loop ()
    - With printing each alphabet via Serial
- Results
  - (a source code file) sketch source code (***"sketchfilename.ino"***)
  - (a short video) a short demo video capture showing how to setup and what to run (max. 1GB file).



Ref. - https://en.wikipedia.org/wiki/Morse_code