Arduino S/W

Lecture 6

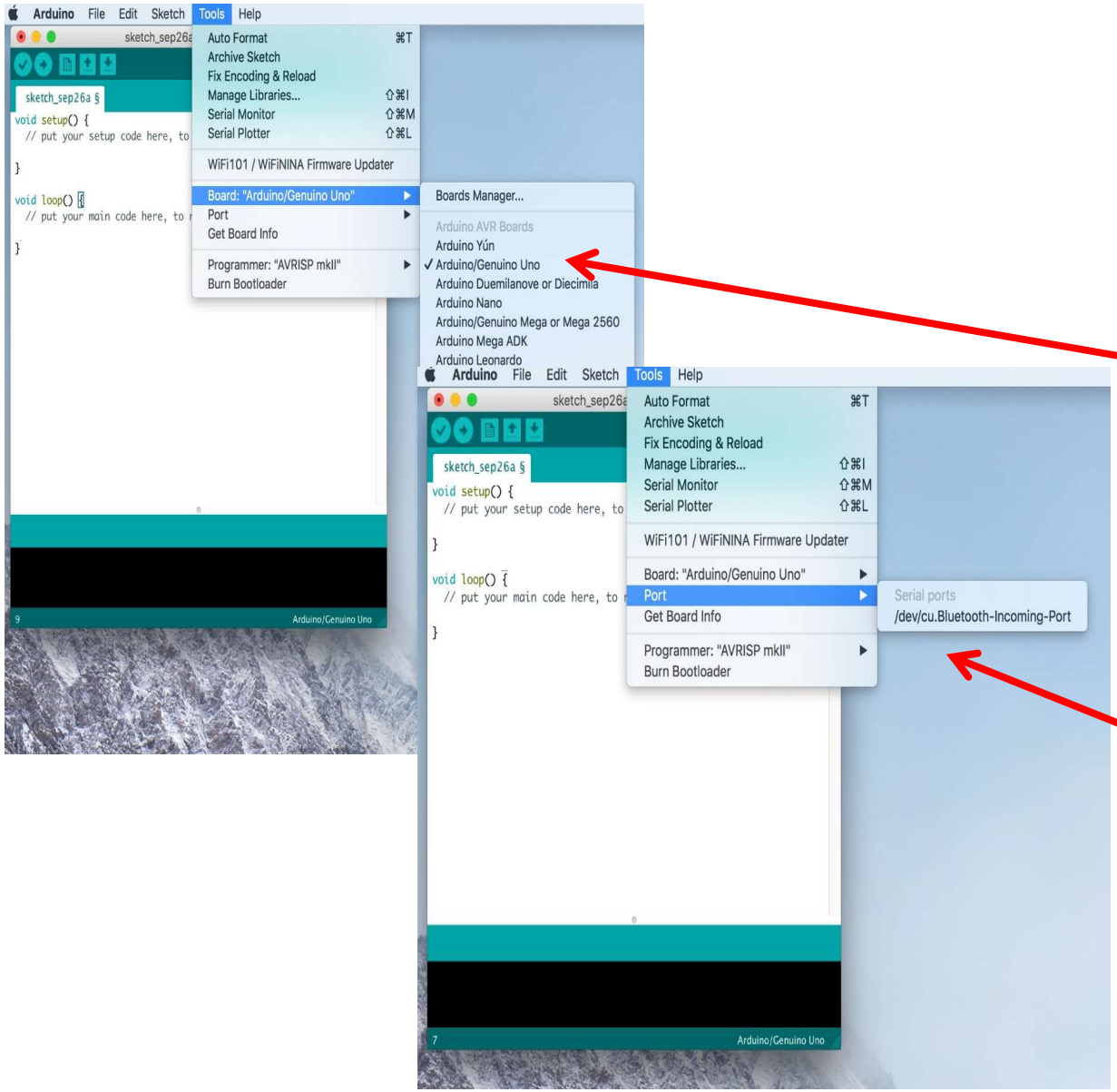
Arduino S/W

- Arduino provides a open-source development interface know as the Integrated Development Environment (IDE).
- IDE supports program facility for program development, verification, and downloading to the Arduino board.
- IDE support a serial monitor to allow users to interface with the program running on the Arduino board.
- IDE provides some basic programs for the user to learn how to program the Arduino board.
- The IDE runs on PCs and Macs and can be downloaded at www.arduino.cc/en/main/software.
- An Arduino program is called a **Sketch** and is written in **C/C++**.

Arduino IDE

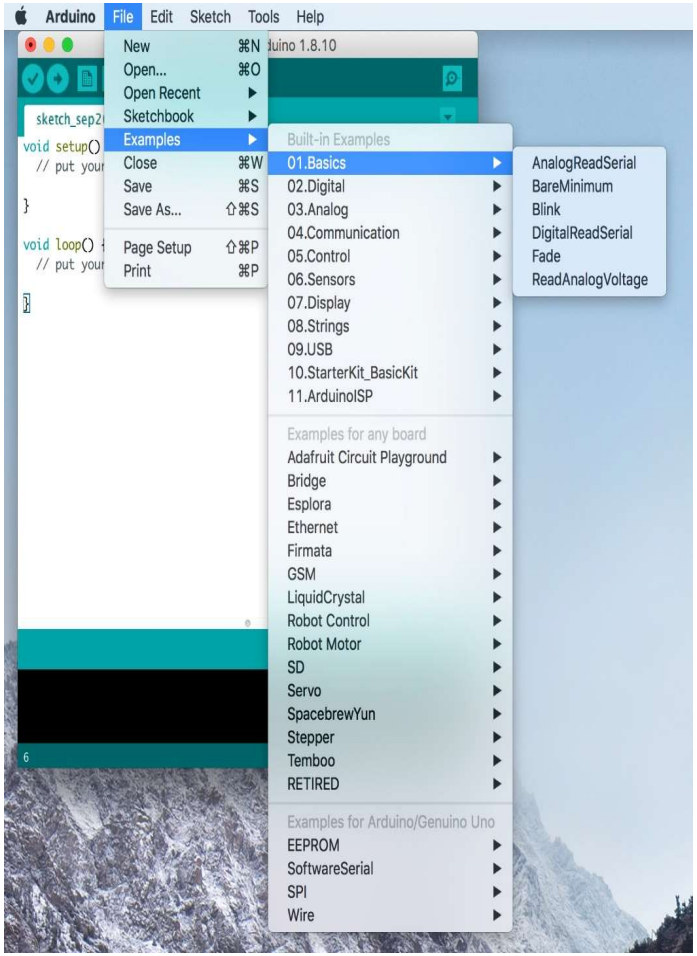
- Editor: Where sketches are written
- Verify: Tests whether the sketch mets the C/C++ syntax rules
- Upload: When the verify shows no errors, send the sketch to the Arduino board
- Message Area and Console: Where messages are output when verifying and uploading

First steps for running a sketch



1. Connect the Arduino board to a USB port on your computer.
2. On the Toolbar, select Tools>Board and click on your Arduino board
3. On the Toolbar, select Tools>Port and click on the port your Arduino is connected. (com1, com2 or com3 for PCs or /dev/cu.usbmodem-XXXX for Mac.

Where Example Sketches are found



On the toolbar, select File>Examples and chose the desired sketch.

Arduino Sketch Structure

- Sketch executable statements end with the character ;

- Comments are used to document the sketch
  - Multiple line comments begin with the characters /* and end with the characters */
  - Single line comments begin with the characters // and can be placed in a line by itself or within the same line as an executable statement.

- Declare global variables area (not needed always) is a place to define variable names used in the sketch and their values. This appears as the first executable statements (i.e., before the Setup and Loop areas).
- Setup area (always needed) is where statements are kept that only run once in the sketch.

- Loop area (always needed) is where statements are which are continuously executed in loop or repetitive manner.

An Example Sketch

```
/*
  Fade

  This example shows how to fade an LED on pin 9 using the analogWrite()
  function.

  The analogWrite() function uses PWM, so if you want to change the pin you're
  using, be sure to use another PWM capable pin. On most Arduino, the PWM pins
  are identified with a "~" sign, like ~3, ~5, ~6, ~9, ~10 and ~11.

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Fade
*/

int led = 9;           // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

Sketch Areas

Preliminary
Comments

```
/*
  Fade

  This example shows how to fade an LED on pin 9 using the analogWrite()
  function.

  The analogWrite() function uses PWM, so if you want to change the pin you're
  using, be sure to use another PWM capable pin. On most Arduino, the PWM pins
  are identified with a "~" sign, like ~3, ~5, ~6, ~9, ~10 and ~11.

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Fade
*/

int led = 9;           // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

Sketch Areas

Declare Global variables

```
/*
  Fade

  This example shows how to fade an LED on pin 9 using the analogWrite()
  function.

  The analogWrite() function uses PWM, so if you want to change the pin you're
  using, be sure to use another PWM capable pin. On most Arduino, the PWM pins
  are identified with a "~" sign, like ~3, ~5, ~6, ~9, ~10 and ~11.

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Fade
*/

int led = 9;           // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

Sketch Areas

Setup

```
/*
  Fade

  This example shows how to fade an LED on pin 9 using the analogWrite()
  function.

  The analogWrite() function uses PWM, so if you want to change the pin you're
  using, be sure to use another PWM capable pin. On most Arduino, the PWM pins
  are identified with a "~" sign, like ~3, ~5, ~6, ~9, ~10 and ~11.

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Fade
*/

int led = 9;           // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

Sketch Areas

Loop

```
/*
  Fade

  This example shows how to fade an LED on pin 9 using the analogWrite()
  function.

  The analogWrite() function uses PWM, so if you want to change the pin you're
  using, be sure to use another PWM capable pin. On most Arduino, the PWM pins
  are identified with a "~" sign, like ~3, ~5, ~6, ~9, ~10 and ~11.

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Fade
*/

int led = 9;           // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

Setup and Loop Statements/Functions

- Setup and loop are C/C++ functions and follow the C/C++ syntax for functions.
- Note that both the setup and loop statements begin with the word void.
  - ❑ This tells the C/C++ complier that the setup and loop do not produce any results to be used by other parts of the program.

```
void setup() {
  // declare pin 9 to be an
output:
  pinMode(led, OUTPUT);
}
```

```
void loop() {
  // set the brightness of pin 9:
// statements were removed for
convenience
  analogWrite(led, brightness);
delay(30);
}
```

Setup and Loop Statements/Functions Cont'd

- Note that the word setup and loop there are left and right parentheses with nothing between them.
  - ❑ This tells the C/C++ complier that the setup and loop do not have any special inputs to run them.
- Note that the statements within the setup and loop areas begin with the character { and end with the character }.

```
void setup() {
  // declare pin 9 to be an
output:
  pinMode(led, OUTPUT);
}
```

```
void loop() {
  // set the brightness of pin 9:
// statements were removed for
convenience
  analogWrite(led, brightness);
delay(30);
}
```

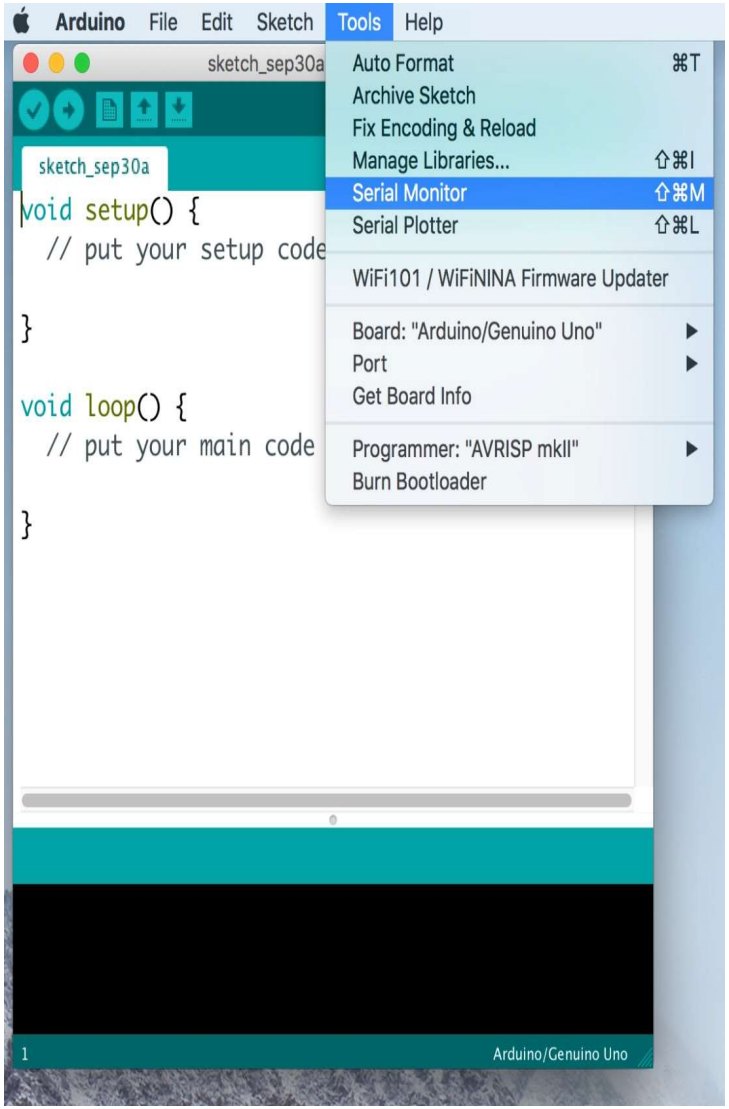
Declare Variables Area

- In the Declare Variable area, variables may be defined with their value.

```
int  led = 0; //assign the value of zero to an integer variable named led.
```

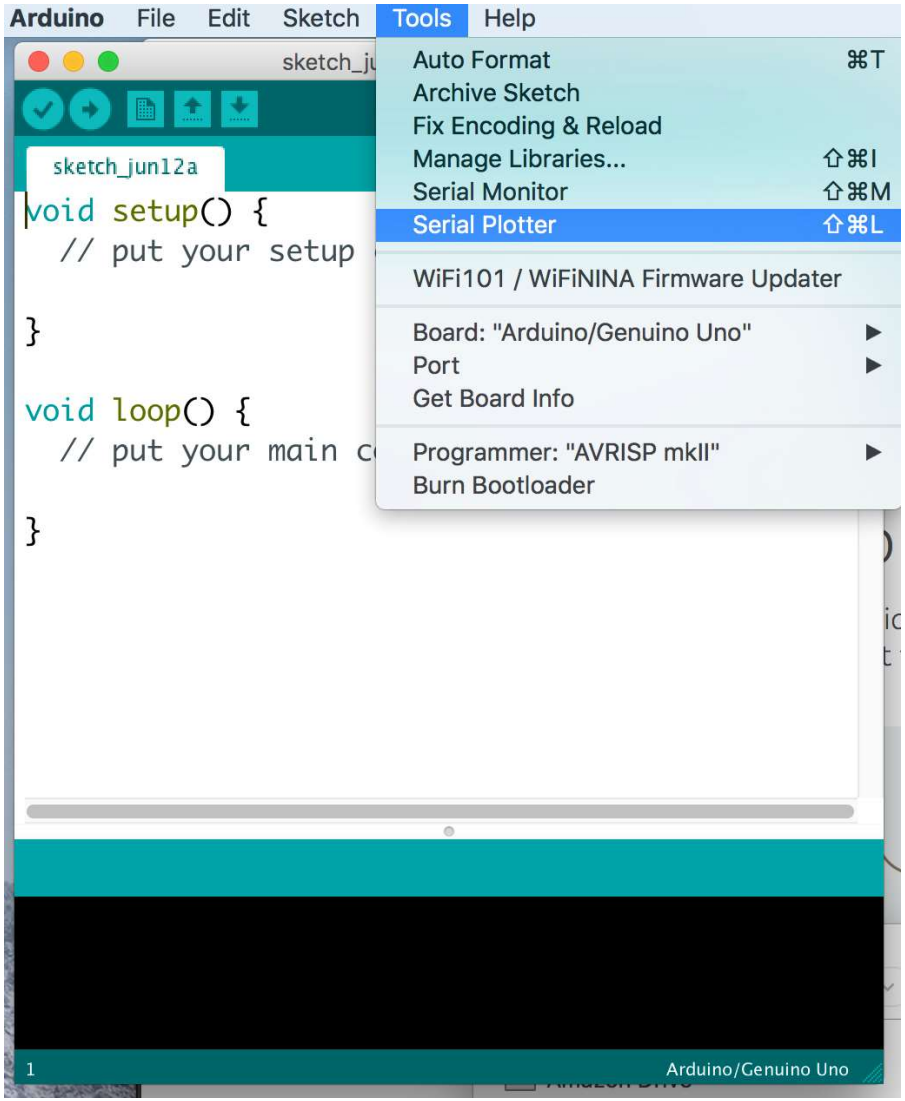
- These variables may be used throughout the sketch by both the setup and loop functions and are called global variables.
- Note variables may also be define within functions like setup() and loop(). They can only be used within the function there are declared and are call local variables.

Serial Monitor



- To access the Serial Monitor click on Tools>Serial Monitor.
- The Serial Monitor supports writing/reading data to/from the Arduino board.

Serial Plotter



- To access the Serial Plotter click on Tools>Serial Monitor.
- The Serial Plotter supports plotting waveforms in real time.