Arduino Board Syntax[1,2]

Lecture 7

1. M. Banzi & M. Shiloh, Getting Started with Arduino, 3rd Edition, MakerMedia, 2014
2. https://www.arduino.cc/reference/en/

Basic Structure

- Declare Variables

```
int led = 9;    //Declares the variable led as one that stores integers
and is set to the value 13
```

- Setup where code will only be executed once.

```
void setup() {
/* Add statements to set up sketch operation */
}
```

- Loop where code will be repeatedly executed until the board power is turned off.

```
void loop() {
/* Add statements to perform sketch operation */
}
```

Special Symbols

- Semicolon - ;

  Every line of code must end with a semicolon.

  ```
  delay (1000);  //delay the sketch by 1 second
  ```

- Parenthesis - (), square brackets - [] and Curly brackets – {}

  Data is passed/received to/from a function which is placed between parenthesis. ()
  Arrays are defined using square brackets. []
  Blocks of code appear between curly brackets.{}

  ```
  int array1[4]={1, 3, 4, 7};

  void loop() {
        Serial.println("ciao");  //The word ciao is sent to the serial monitor
  }
  ```

- Comments

  Single line comments have the characters // at the start of the comments

  Multi-line comments appear between the two sets of characters /* and */

Constants

- The Arduino supports key word constants. Here are ones used often. See https://www.arduino.cc/reference/en/ for more.

  HIGH | LOW Used for pin states:

  1. if the pin is used as an input then it's a HIGH when a voltage of 3.3 volts or greater appears at the pin or it's a LOW when a voltage of 1.5 volts or less appears at the pin.
  2. if the pin is used as an output then the board places a voltage of 5 volts when the pin is declared HIGH and places a voltage of 0 volts when the pin is declared LOW.

  INPUT | OUTPUT Used to set a pin to either an input or output port.

  true | false Used to test a comparison.

  LED_BUILTIN Set to 13 for the on-board LED that is connected to pin 13.

Data Types

- The Arduino supports various types of data types. Here are ones used often. See https://www.arduino.cc/reference/en/ for more.

1. Integer data type

int – Uses 2 bytes and declares a variable as an integer of value -32,768 to 32,768.

unsigned int – Uses 2 bytes and declares a variable as an integer of value 0 to 65,535.

long – Uses 4 bytes and declares a variable as an integer of value -2,147,483,648 to 2,147,483,648

unsigned long– Uses 4 bytes and declares a variable as an integer of value 0 to 4,294,967,295.

Data Types (cont'd)

2. Floating point data types

float – Uses 4 bytes (1 bit for sign, 8 bits for the exponent, and 23 for the value) and declares a variable as floating point number of value -3.4028235E38 to 3.4028235E38.

double - Uses 8 bytes (1 bit for sign, 11 bits for the exponent, and 52 for the value) and declares a variable as floating point number of value -1.79766931348623157E308 to 1.79766931348623157E308.

2. 3. Array data types

array – Arrays are defined using square brackets, curly brackets, and the following format.

```
int load[5]={1, 2, 3, 4, 5};
```

Data Types (cont'd)

4. Character data types

Characters use 1 byte per character and are defined formats similar to arrays.

char – Uses 1 byte per character

char string1[3]="BME";

String – similar to char

String string1[3]="BME";

Data Types (cont'd)

5. Conversion of values to a data type

   float(x) - converts the value of x into a floating point data type

   (float)x – alternative form

Using this syntax, conversion of a value to a data type can be used for the other data types: int, unsigned, long, unsigned long, double, etc.

Arithmetic Operators

Arithmetic

Operator	Test	Example
+	Addition	
-	Subtraction	
*	Multiplication	
/	Division	
=	Assignment	
%	Remainder	r=7%5; //r=2

Compound Operators

Symbol	Function	Example
++	Increment	y=x++; // y=x+1
--	Decrement	y=x--; // y=x-1

Other Operators and Symbols

Relationship and equality operators used for number and strings

Operator	Test	Example
==	Equal to	If(val == HIGH){
!=	Not equal to	If(val != HIGH){
>	Greater than	If(val > HIGH){
<	Less than	If(val < HIGH){
>=	Greater than or equal to	If(val >= HIGH){
<=	Less than or equal to	If(val <= HIGH){

Other Operators and Symbols

Logical Operators

Symbol	Function
&&	Logical And
||	Logical Or
!	Not

Bit Operators

Operator	Test
&	Bitwise And
|	Bitwise Or
^	Bitwise Exclusive Or
~	Bitwise Negation

Control Statements

1. IF Statement: if condition is true, take action and continue.

   IF Statement: if condition is true, take action. Otherwise take another action.

```
if (condition){
    //If true take
    action
    val=10;
}
```

```
if (condition){
    //If true take
    action
}
else {
    //If false take
    another action
    val=20;
}
```

Control Statements Continued

2. For Statement: defines a loop with a counter initialization, condition for looping, counter increment

   Initialization occurs once

   Condition: each time the loop is executed the condition is tested.

   If true looping continues. If false looping ends and execution proceeds to the statement following the For Statement

   Increment: each time the through the loop, the counter is incremented and the condition is retested.

```
for (initialization; condition; increment) {
```

```
for (int i = 0; i <= 100; i++ ) {      // for i starting at 0, is it <= 100, run loop and increment I

        delay (1000);                  //delay a second
                                        // when i > 100 end loop
}
```

Control Statements Continued

3. While Statement: defines a loop with a condition for running the loop. When the condition becomes false the loop ends.

```
while (condition) {

while (var <= 100) {     // while a variable named var less than or equal to 100, run loop
        delay (500);              //delay 1/2 second
                                  // when var > 100 end loop
}
```

Control Statements Continued

4. Switch Case Statement: defines a switch and a series of case statements. When there is a case statement matching the switch, the statements following the case statements are executed. The default case is executed if no match is found.

```
switch (var) {
        case label1;
        //statements
        break;
        case label2;
        //statements
        break;
        default;
        //statements
        break;
}
```

```
switch (var) {
        case 1;
        delay (1000);    //delay 1 second
        break;
        case 2;
        delay (500);    //delay 1/2 second
        break;
        default;
        delay (2000);    //delay 2 seconds
        break;
}
```

Mathematical Operations

1. Absolute value: finds the absolute value of a number, x.

   abs(x)

   y=abs(-3); //y=3

2. Map: maps a value from range of numbers to another range.

   map(value, fromlow, fromhigh, tolow, tohigh)

   y=map (20, 1, 50, 1, 100); //y=40

3. Minimum/Maximum: finds the minimum/maximum of 2 values.

   min(val1, val2) max(val1,val2)

   y=min(3,4); //y=3

   y=max(3,4); //y=4

Mathematical Operations Continued

4. Power: calculates the value of a number raised to a power.

   pow(base, exponent)

   y=pow(3, 2); //y=9

5. Square root: calculates the square root of a number.

   y=sqrt(value)

   Y=sqrt(64); //y=8

6. Trigonometric operations: calculates the sine, cosine and tangent of a number in radians

   sin(x) cos(x) tan(x)

   y=sin(3.14); //y=0

   y=cos(1.57); //y=0

   y=tan(0.78); //y=1

Input and Output Functions
Digital Pins

1. Configuring a digital pin as an input or output

   pinmode(pin,mode);

   pinmode(13,OUTPUT); //Turns pin 13 into an output

2. Turns a digital pin HIGH or LOW

   digitalWrite(pin,value);

   digitalWrite(13,LOW); //Turns pin 13 LOW

3. Controls the PWM signal at certain digital pins; a value of 0 turns the pin off and a values of 255 turns pin fully on.

   analogwrite(pin,value);

   analogwrite(13, 127); //Turns on a PWM with 50% duty cycle

4. Reads the state of an input digital pin and returns HIGH or LOW

   digitalRead(pin);

   val=digitalRead(13); //Reads pin 13 and returns the value into variable named val

Input and Output Functions
Analog Pins

1. Reads the state of an input Analog pin and return a number from 0 to 1023 which corresponds to a voltage between 0 and 5 volts.

   analogRead(pin);

   val=analogRead(0); //Reads analog pin 0 and returns the value into variable named val

2. You can not write to an analog pin

Time Functions

1. Returns the number of milliseconds (microseconds) that have passed since the sketch started

```
millis();
duration=millis()-lasttime; //computes time since lasttime in milli-seconds

micros();
duration=micros()-lasttime; //computes time since lasttime in micro-seconds
```

2. Pauses sketch for the number of milliseconds specified

```
delay(ms);
delay(500); //delay for ½ second
```

3. Pauses sketch for the number of microseconds specified

```
delayMicroseconds(µs);
delayMicroseconds(500); //delay for ½ millisecond
```

Serial Monitor

1. Prepares the Arduino to send/receive data.

   Serial.begin(speed);

   Serial.begin(9600); // Typical setting for the Arduino

2. Sends data to the serial port.

   Serial.print(data); //prints data

   Serial.println(data); //prints data with line feed

   Serial.print(75); //Prints the characters “75”

   Serial.print(75,HEX); //Prints “4B” (75 in hexadecimal)

   Serial.print(75,BIN); //Prints 1001011 (75 in binary)

   Serial.println(75);//Prints”75” with a carriage return and linefeed i.e., “75\r\n”

3. Reads 1 byte of incoming serial data

   Serial.read();

   data=Serial.read(); //Put a byte of data into the variable data

Serial Plotter

1. Same operation at the serial monitor example when serial monitor is selected a waveform plot of the data is produced.
2. Prepares the Arduino to send/receive data.

   Serial.begin(speed);

   Serial.begin(9600); // Typical setting for the Arduino
3. Sends data to the serial port.

   Serial.print(data);

Libraries

1. In order to extend the Arduino, libraries may uses. Libraries contain functions which aid the extension of the Arduino.
2. To use a library, it must be importing or included into the sketch.

   #include <library.h>
3. Examples
   a. LiquidCrystal – setCursor(), blink(), scrollDisplayLeft(), scrollDisplayRight()
   b. Servo – attach(pin), *servo*.write(angle), *servo*.read()
   c. Stepper – setSpeed(rpms), step(steps)
   d. WiFi – WiFi.config(ip), WiFi.status()