Introduction to Arduino

Mahesh Neelakanta
Director of ITS
FAU Libraries

July 2020

credit where credit is due..

Most material in this training program is based on the Sparkfun Activity Guide for SparkFun Tinker Kit and is licensed similarly Creative Commons Attribution Share-Alike 4.0 International License.

Who are you and why are you doing this?

- Mahesh Neelakanta
  - Director of ITS
  - IT Director at Engineering for 13 years and recently moved to Libraries in Fall 2019

- Library technology outreach initiatives to bring various areas of technology to the general student, faculty and staff population
  - 3D Printing (ITS Webinar by Hansy, Crystal and Yom)
  - Cloud Computing
  - Artificial Intelligence and Cloud Computing

Housekeeping

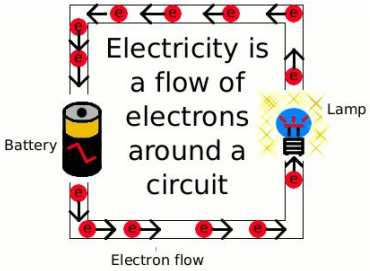
- Meeting is in webinar mode
  - Attendee mics will be muted (I can unmute if needed)
  - No video of attendees (only panelists)
  - Please ask your questions in Q&A and/or Chat
  - You can raise your hand as needed
- Polls
  - https://meet.ps/mahesh
- Slides and Documentation
  - https://github.com/faumahesh/arduino

Module 1 - Goals

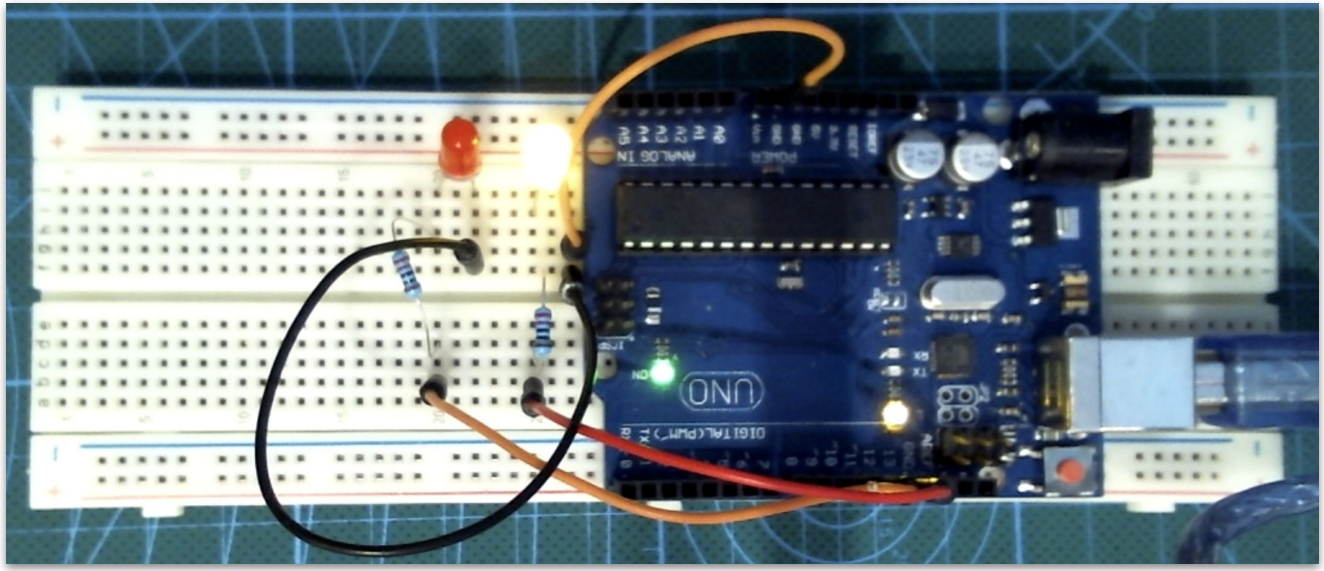
- Let's start with a demo!
- How is this series organized?
- What is Arduino and how to get started?
- What can you do with Arduino?
- The Sparkfun Tinkerkit Unboxing (https://www.sparkfun.com/tinkerkit)
- Let's Get Started!
  - Install Arduino Software
  - Circuit 1 - Blink an LED
- Module 2 Goals
- Additional Reading , Q&A

Let's start with a demo!

- Blinking LED

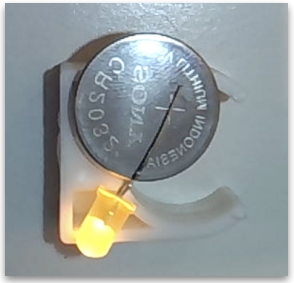



```
1
2 void setup() {
3   pinMode(13, OUTPUT);
4   pinMode(12, OUTPUT);
5 }
6
7
8 void loop() {
9   digitalWrite(13, HIGH);   // Turn on the LED
10  digitalWrite(12, LOW);    // Turn on the LED
11  delay(1000);              // Wait for two seconds
12  digitalWrite(13, LOW);    // Turn off the LED
13  digitalWrite(12, HIGH);   // Turn off the LED
14  delay(1000);              // Wait for two seconds
15 }
16
```



Fancy Version

Cheap Version

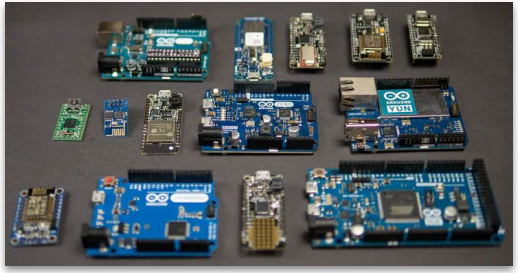


How is this series organized?

- We will be covering the Sparkfun Activity Guide for the Tinker Kit over a period of 5-6 sessions
- Each session (except #2) will have 1-2 circuits from guide, ad-hoc circuit followed by a Q&A period
- Session 2 (next one) will focus on Arduino "C" language introduction
- This is a beginner course. At the end of the series, we will gauge interest for an intermediate class (WiFi, Shields) and an advanced class (IoT, Cloud)
- Sessions will be recorded for replay at FAU Libraries Youtube or Facebook
- Survey at the end of this session on what schedule works best for everyone
    - Weekly, twice a month, monthly?

What is Arduino and how to get started?

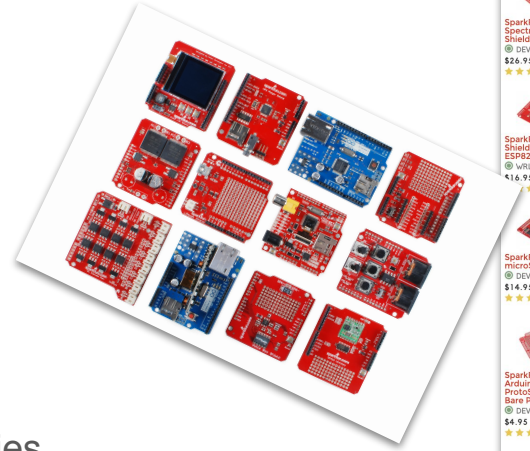
- Easy to use hardware framework to learn (hands-on) electronics and coding
- Cheap enough to experiment-destroy-restart ($10 - $30)
- Open source hardware and lots of software to get started quickly
- Rich ecosystem of "shields" for easy extensibility - https://store.arduino.cc/usa/arduino/shields
- Common platform and jumping point to more advanced devices
- Program in "C" or Visually using Blockly (or Python with higher end units)
    - Arduino IDE (what we will be using) - https://www.arduino.cc/en/main/software
    - Otto Blockly - https://github.com/OttoDIY/blockly



https://www.hackster.io/news/picking-the-right-arduino-341a0a9550c7

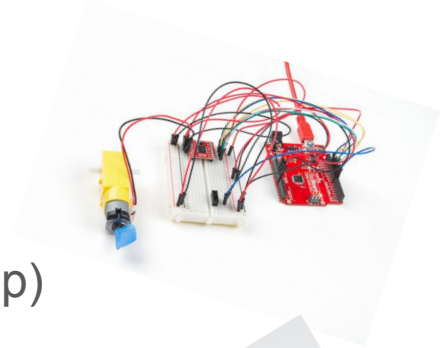
Arduino Boards (microcontrollers)

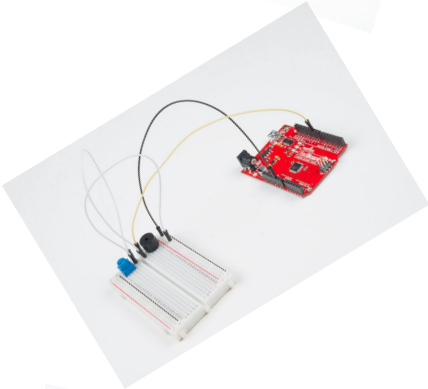
- Open Source hardware by
- Uno for entry level playground
  - Arduino Uno - $23
  - SparkFun RedBoard - $20
  - AdaFruit Metro - $18
  - Amazon Generic Uno - $13
  - Generic from China - $4
- Due, Mega, Nano, Zero
  - Different form factors and capacities
  - https://www.sparkfun.com/standard_arduino_comparison_guide
  - https://electropeak.com/learn/arduino-buying-guide-how-to-choose-right-arduino-board/



What can you do with Arduino?

- Learn to code with an physical device (vs on desktop/web/app)
- Learn the basics of electronics and circuits by experimenting
- Build devices with sensors and displays
  - Check temperature, humidity, barometric pressure, light
  - Turn a switch on/off
  - Play a tune
  - Make a motor rotate
- Bond with your kids with hands on activities
- Enter into a much wider world of connected devices such as home automation and Internet-of-Things (IOT)
- What about you? https://meet.ps/mahesh





Sparkfun Tinker Kit Unboxing

- https://www.sparkfun.com/products/14556
- SparkFun RedBoard
- SparkFun Motor Driver (with Headers)
- Breadboard - Self-Adhesive (White)
- Servo -- Sub-Micro Size
- Hobby Gearmotor -- 200 RPM (Pair)
- Temperature Sensor -- TMP36
- Mini Photocell
- Piezoelectric Speaker
- SparkFun USB Mini-B Cable -- 6 Foot
- Jumper Wires -- 7" M/M 30 AWG (30 Pack)
- LED - RGB Diffused Common Cathode
- Red, Blue, Yellow, and Green LEDs
- Red, Blue, Yellow, and Green Buttons
- Mini Power Switch
- 10K Trimpot
- Battery Holder - 4xAA to Barrel Jack Connector
- 330 and 10K Resistors

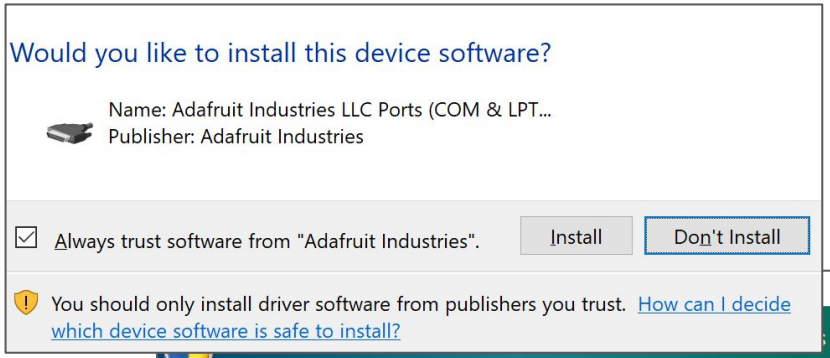
Let's Get Started!

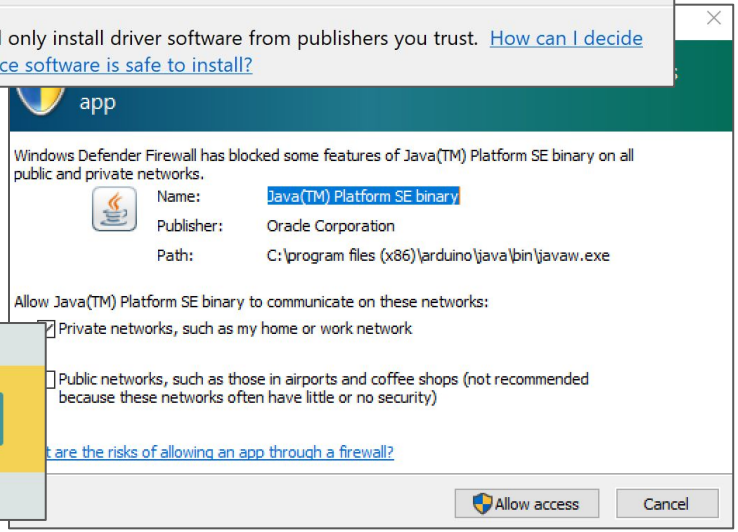
Arduino Software

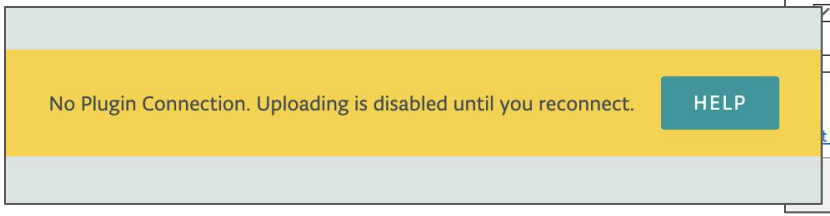
1. Web based interface on https://create.arduino.cc/
2. Software simulator at https://www.tinkercad.com/
3. Download and install desktop client for Mac, Windows or Linux
    a. https://www.arduino.cc/en/Main/Software
4. For this class, we will be using the desktop client

Install Arduino Software

https://learn.sparkfun.com/tutorials/installing-arduino-ide

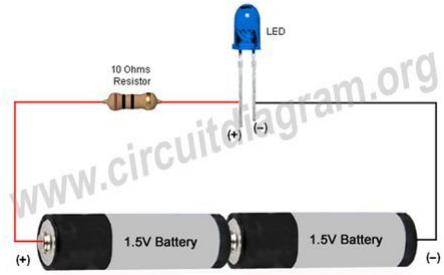






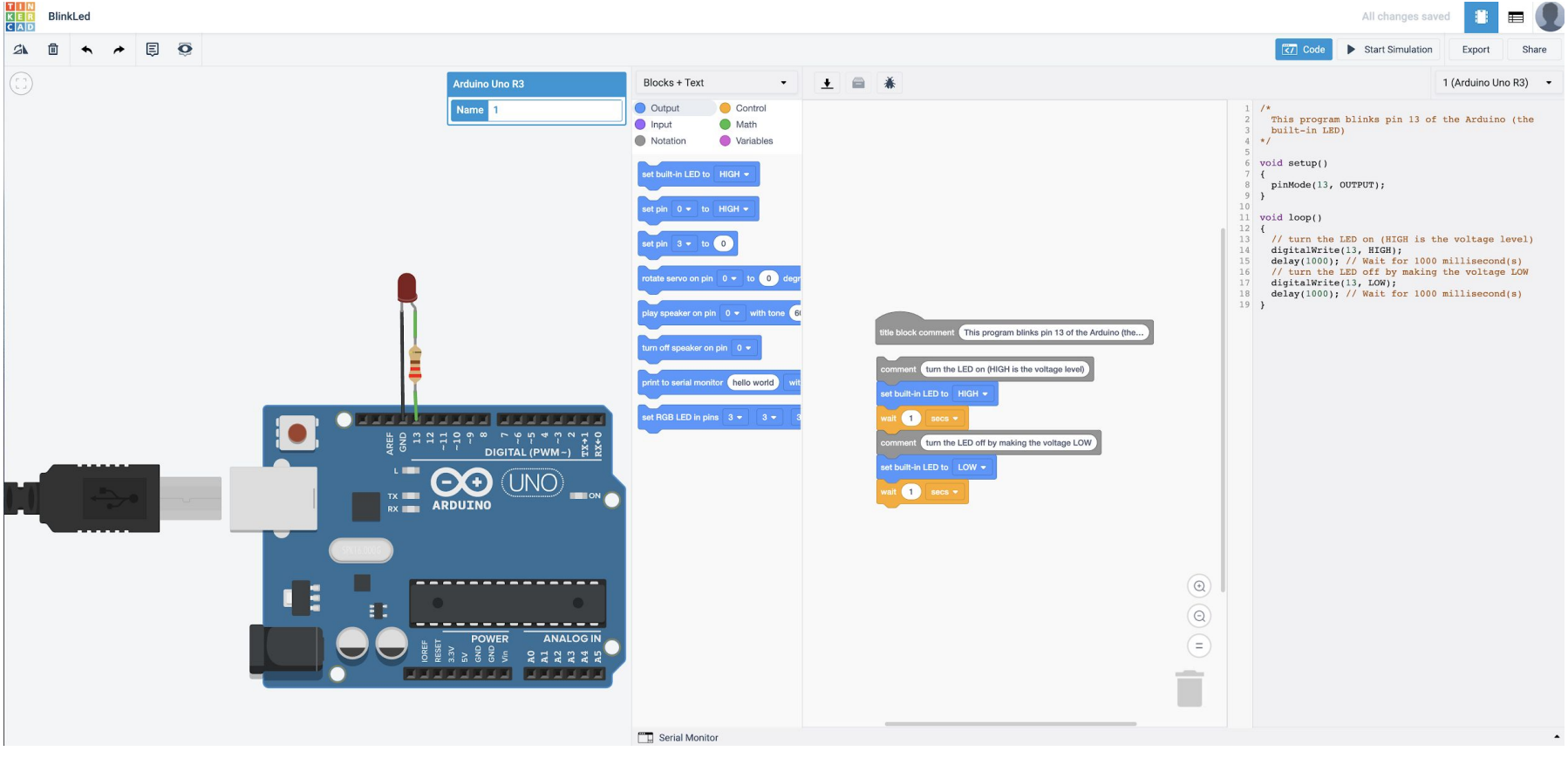
Circuit 1 - Blink a LED

- https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard
- https://learn.sparkfun.com/tutorials/redboard-hookup-guide
- https://learn.sparkfun.com/tutorials/activity-guide-for-sparkfun-tinker-kit/circuit-1-blink-an-led
  - Build the circuit
  - Copy the code
  - Test the circuit
  - Modify the circuit to add a second LED
  - Modify the code and circuit to control the two LEDs separately
  - Modify the code to blink the two LEDS opposite of each other



Bonus! Arduino Simulator

https://www.tinkercad.com/



Additional Reading

- https://learn.sparkfun.com/tutorials/what-is-an-arduino
- https://learn.sparkfun.com/tutorials/what-is-a-circuit
- https://www.arduino.cc/en/Guide/Introduction
- https://files.seeedstudio.com/wiki/Book_and_stickers/A_Brief_Intro_to_Electronics.pdf
- https://learn.sparkfun.com/tutorials/redboard-hookup-guide

Q&A

Contact information : mahesh@fau.edu

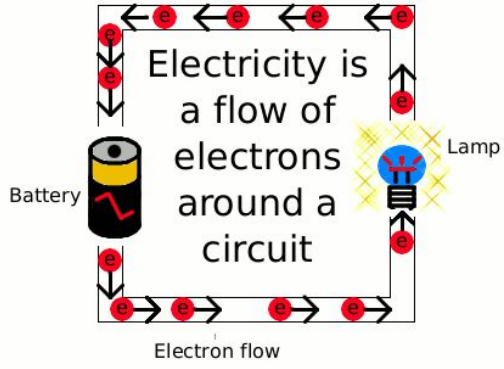
Module 3 - Goals

- Questions from previous session?
- What are some other embedded devices?
- Basic electronics and circuits
- Circuit 2 - Potentiometer
- Circuit 3 - Photoresistor
- Q&A

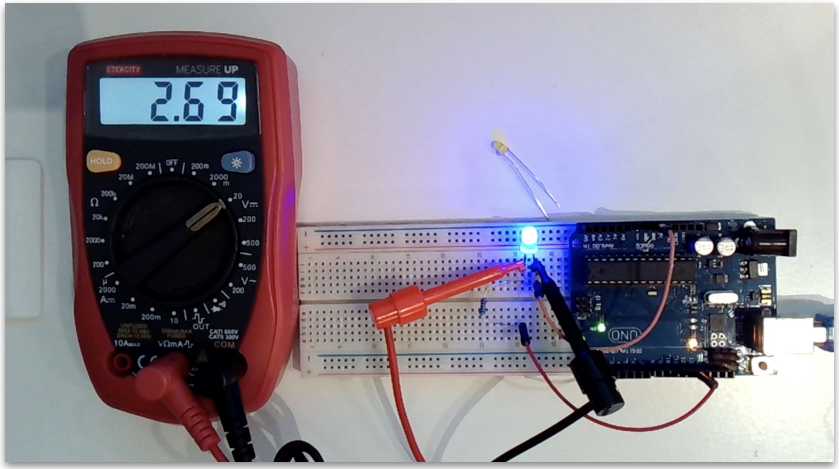
Module 2 - Goals

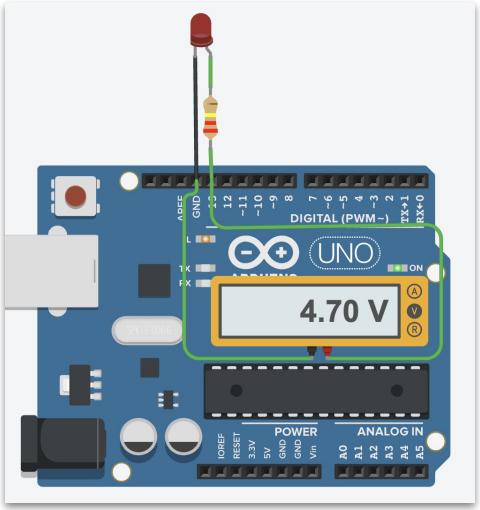
- Questions from previous session?
- Blinking LED with a multimeter to measure the electric voltage
- Morse Code SOS example
- Arduino Language Tutorial
- Extending Blinking LEDs with some logic

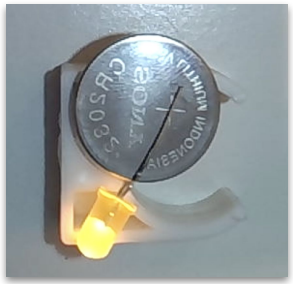
A quick detour on electricity



https://rme.sharylandisd.org/cms/One.aspx?portalId=418090&pageId=2408938

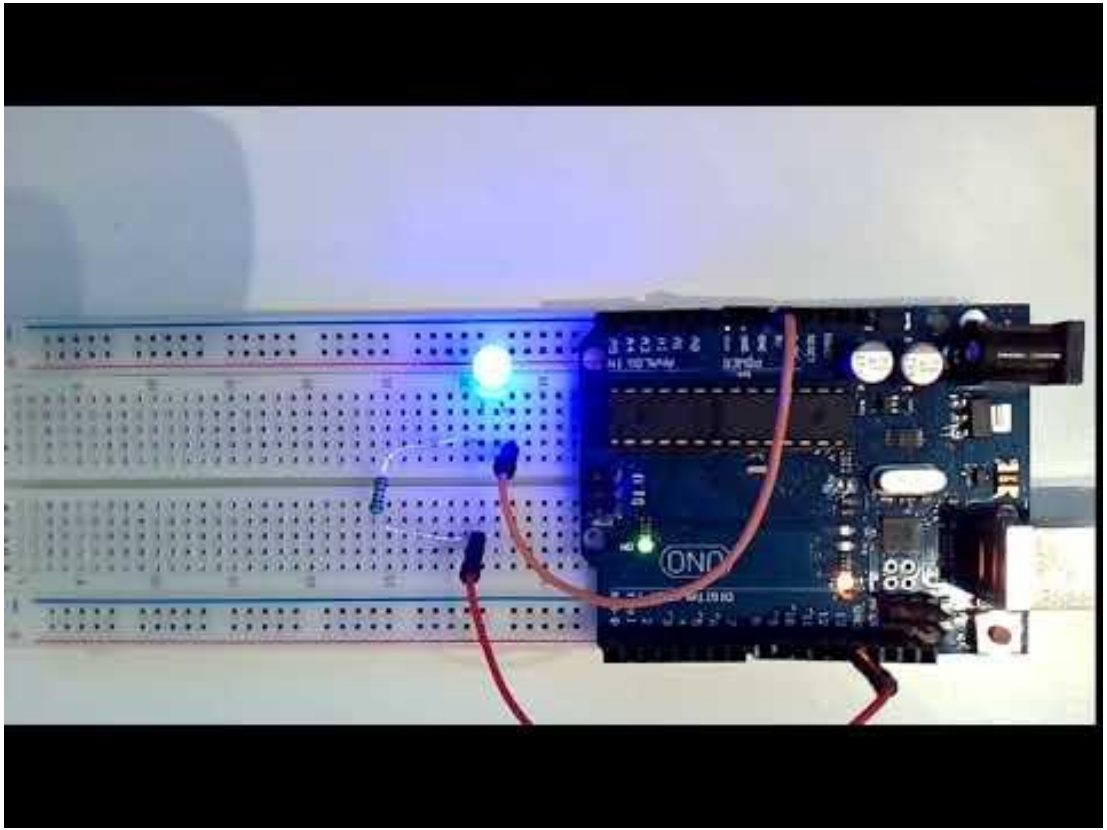






Blinking LED with a multimeter to measure the electric voltage

Morse Code SOS Example



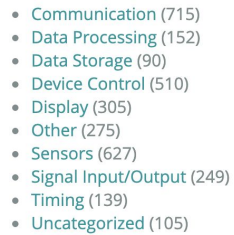
https://github.com/faumahesh/arduino/tree/master/morse_simple

morse_simple

```
//
// Simple Morse SOS demo
// 7/2020 - Mahesh Neelakanta
//

#define LEDPIN 13
#define DOTDELAY 200   // How long to wait per dot
#define DASHDELAY 350  // How long to wait per dash
#define CHARDELAY 500  // How long to wait after end of each character
#define EOMDELAY  1000 // How long to wait after end of message

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

void loop()
{
 // SOS = ... --- ...

 dotBlink();
 dotBlink();
 dotBlink();
 delay(CHARDELAY);

 dashBlink();
 dashBlink();
 dashBlink();
 delay(CHARDELAY);

 dotBlink();
 dotBlink();
 dotBlink();
 delay(EOMDELAY);

}

void dotBlink()
{
  digitalWrite(LEDPIN, HIGH);  // Turn on the LED
  delay(DOTDELAY);             // Leave LED on for a short (DOT) delay
  digitalWrite(LEDPIN, LOW);   // Turn off the LED
  delay(DOTDELAY);             // Leave LED on for a short (DOT) delay

}

void dashBlink()
{
  digitalWrite(LEDPIN, HIGH);  // Turn on the LED
  delay(DASHDELAY);            // Leave LED on for a long (DASH) delay)
  digitalWrite(LEDPIN, LOW);   // Turn off the LED
  delay(DASHDELAY);            // Leave LED on for a long (DASH) delay)
}
```

Arduino Language

- Variation of the C/C++ Language.
- Core Language
  - Structure
  - Data Types, Variables and Constants
  - Functions
  - https://www.arduino.cc/reference
- Libraries (Extensions)
  - https://www.arduino.cc/reference/en/libraries/



STRUCTURE

The elements of Arduino (C++) code.

Sketch

loop()
setup()

Control Structure

break
continue
do...while
else
for
goto
if
return
switch...case
while

Further Syntax

#define (define)
#include (include)
/* */ (block comment)
// (single line comment)
; (semicolon)
{} (curly braces)

Arithmetic Operators

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

Comparison Operators

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

Boolean Operators

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

Pointer Access Operators

& (reference operator)
* (dereference operator)

Bitwise Operators

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

Compound Operators

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

VARIABLES

Arduino data types and constants.

Constants

HIGH | LOW
INPUT | OUTPUT | INPUT_PULLUP
LED_BUILTIN
true | false
Floating Point Constants
Integer Constants

Conversion

(unsigned int)
(unsigned long)
byte()
char()
float()
int()
long()
word()

Data Types

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

Variable Scope & Qualifiers

const
scope
static
volatile

Utilities

PROGMEM
sizeof()

FUNCTIONS

For controlling the Arduino board and performing computations.

Digital I/O

digitalRead()
digitalWrite()
pinMode()

Analog I/O

analogRead()
analogReference()
analogWrite()

Zero, Due & MKR Family

analogReadResolution()
analogWriteResolution()

Advanced I/O

noTone()
pulseIn()
pulseInLong()
shiftIn()
shiftOut()
tone()

Time

delay()
delayMicroseconds()
micros()
millis()

Math

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

Trigonometry

cos()
sin()
tan()

Characters

isAlpha()
isAlphaNumeric()
isAscii()
isControl()
isDigit()
isGraph()
isHexadecimalDigit()
isLowerCase()
isPrintable()
isPunct()
isSpace()
isUpperCase()
isWhitespace()

Random Numbers

random()
randomSeed()

Bits and Bytes

bit()
bitClear()
bitRead()
bitSet()
bitWrite()
highByte()
lowByte()

External Interrupts

attachInterrupt()
detachInterrupt()

Interrupts

interrupts()
noInterrupts()

Communication

Serial
Stream

USB

Keyboard
Mouse

The next set of slides have significant content taken from the arduino language reference guide:

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

Control Structure

Comments, Curly Braces and Semicolons

```
/* 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!
*/
```

```
// pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;
digitalWrite(led, HIGH);  // turn the LED on (HIGH is the voltage level)
```

```
int a = 13;
```

Functions

```
void myfunction(datatype argument) {
  // any statement(s)
}
```

Loops

```
while (boolean expression) {
  // any statement(s)
}

do {
  // any statement(s)
} while (boolean expression);

for (initialisation; termination condition; incrementing expr) {
  // any statement(s)
}
```

Conditional Statements

```
if (boolean expression) {
  // any statement(s)
}

else if (boolean expression) {
  // any statement(s)
}
else {
  // any statement(s)
}
```

if

[Control Structure]

Description

The `if` statement checks for a condition and executes the proceeding statement or set of statements if the condition is 'true'.

Syntax

```
if (condition) {
  //statement(s)
}
```

Parameters

`condition`: a boolean expression (i.e., can be `true` or `false`).

Example Code

The brackets may be omitted after an if statement. If this is done, the next line (defined by the semicolon) becomes the only conditional statement.

```
if (x > 120) digitalWrite(LEDpin, HIGH);

if (x > 120)
digitalWrite(LEDpin, HIGH);

if (x > 120) {digitalWrite(LEDpin, HIGH);}

if (x > 120) {
  digitalWrite(LEDpin1, HIGH);
  digitalWrite(LEDpin2, HIGH);
}
// all are correct
```

else

[Control Structure]

Description

The **`if…else`** allows greater control over the flow of code than the basic `if` statement, by allowing multiple tests to be grouped. An **`else`** clause (if at all exists) will be executed if the condition in the **`if`** statement results in **`false`**. The **`else`** can proceed another **`if`** test, so that multiple, mutually exclusive tests can be run at the same time.

Each test will proceed to the next one until a true test is encountered. When a true test is found, its associated block of code is run, and the program then skips to the line following the entire if/else construction. If no test proves to be true, the default **`else`** block is executed, if one is present, and sets the default behavior.

Note that an **`else if`** block may be used with or without a terminating **`else`** block and vice versa. An unlimited number of such **`else if`** branches are allowed.

Syntax

```
if (condition1) {
  // do Thing A
}
else if (condition2) {
  // do Thing B
}
else {
  // do Thing C
}
```

Example Code

Below is an extract from a code for temperature sensor system

```
if (temperature >= 70) {
  // Danger! Shut down the system.
}
else if (temperature >= 60) { // 60 <= temperature < 70
  // Warning! User attention required.
}
else { // temperature < 60
  // Safe! Continue usual tasks.
}
```

while

[Control Structure]

Description

A `while` loop will loop continuously, and infinitely, until the expression inside the parenthesis, () becomes false. Something must change the tested variable, or the while loop will never exit. This could be in your code, such as an incremented variable, or an external condition, such as testing a sensor.

Syntax

```
while (condition) {
  // statement(s)
}
```

Parameters

`condition`: a boolean expression that evaluates to `true` or `false`.

Example Code

```
var = 0;
while (var < 200) {
  // do something repetitive 200 times
  var++;
}
```

do...while

[Control Structure]

Description

The `do…while` loop works in the same manner as the `while` loop, with the exception that the condition is tested at the end of the loop, so the do loop will always run at least once.

Syntax

```
do {
  // statement block
} while (condition);
```

Parameters

`condition`: a boolean expression that evaluates to `true` or `false`.

Example Code

```
int x = 0;
do {
  delay(50);          // wait for sensors to stabilize
  x = readSensors();  // check the sensors
} while (x < 100);
```

for

[Control Structure]

Description

The `for` statement is used to repeat a block of statements enclosed in curly braces. An increment counter is usually used to increment and terminate the loop. The `for` statement is useful for any repetitive operation, and is often used in combination with arrays to operate on collections of data/pins.

Syntax

```
for (initialization; condition; increment) {
  // statement(s);
}
```

Parameters

`initialization`: happens first and exactly once.
`condition`: each time through the loop, `condition` is tested; if it's `true`, the statement block, and the **increment** is executed, then the **condition** is tested again. When the **condition** becomes `false`, the loop ends.
`increment`: executed each time through the loop when `condition` is `true`.

Example Code

```
// Dim an LED using a PWM pin
int PWMpin = 10;  // LED in series with 470 ohm resistor on pin 10

void setup() {
  // no setup needed
}

void loop() {
  for (int i = 0; i <= 255; i++) {
    analogWrite(PWMpin, i);
    delay(10);
  }
}
```

break

[Control Structure]

Description

`break` is used to exit from a `for`, `while` or `do…while` loop, bypassing the normal loop condition. It is also used to exit from a `switch case` statement.

Example Code

In the following code, the control exits the `for` loop when the sensor value exceeds the threshold.

```
int threshold = 40;
for (int x = 0; x < 255; x++) {
  analogWrite(PWMpin, x);
  sens = analogRead(sensorPin);
  if (sens > threshold) {     // bail out on sensor detect
    x = 0;
    break;
  }
  delay(50);
}
```

continue

[Control Structure]

Description

The `continue` statement skips the rest of the current iteration of a loop (`for`, `while`, or `do…while`). It continues by checking the conditional expression of the loop, and proceeding with any subsequent iterations.

Example Code

The following code writes the value of 0 to 255 to the `PWMpin`, but skips the values in the range of 41 to 119.

```
for (int x = 0; x <= 255; x ++) {
  if (x > 40 && x < 120) {  // create jump in values
    continue;
  }

  analogWrite(PWMpin, x);
  delay(50);
}
```

Comparison Operators

Comparison Operators:

```
x == y (x is equal to y)
x != y (x is not equal to y)
x <  y (x is less than y)
x >  y (x is greater than y)
x <= y (x is less than or equal to y)
x >= y (x is greater than or equal to y)
```

```
if (x > y) {  // tests if x is greater (bigger) than y

      // do something only if the comparison result is true

}

if (x <= y) { // tests if x is less (smaller) than or equal to y

      // do something only if the comparison result is true

}

if (x == y) { // tests if x is equal to y

       // do something only if the comparison result is true

}

if (x != y) { // tests if x is not equal to y

       // do something only if the comparison result is true

}
```

Math Operators

Operator Symbol	Description	Example
=	Store the value of the right hand side to the "variable" on the left hand side	`int temp;` `temp = analogRead(A1);`

+	Add the left and right hand side values	`int a = 10;`
		`int b = 20;`
		`int c;`
		`c = a + b ;`

-	Subtract the right hand side value from left hand side value	`int a = 10;`
		`int b = 20;`
		`int c;`
		`c = a - b ;`

*	Multiply the left and right hand side values	`float tF = 75.0;`
		`float tC;`
		`tC = tF * 1.8 + 32.0;`

/	Divide the right hand side value from left hand side value	`float tC = 23.0;`
		`float tF;`
		`tC = (tF - 32.0) / 1.8;`

Boolean Operators

Boolean Operators

```
if (!x) { // if x is not true
  // statements
}
```

Boolean Operators

! (logical not)

&& (logical and)

|| (logical or)

```
if (digitalRead(2) == HIGH && digitalRead(3) == HIGH) { // if BOTH the switches read HIGH
  // statements
}
```

```
if (x > 0 || y > 0) { // if either x or y is greater than zero
  // statements
}
```

Compound Operators

Compound Operators

%= (compound remainder)

&= (compound bitwise and)

*= (compound multiplication)

++ (increment)

+= (compound addition)

-- (decrement)

-= (compound subtraction)

/= (compound division)

^= (compound bitwise xor)

|= (compound bitwise or)

```
x = 1; x++; // x will be 2

y = 7 ; y--; // y will be 6

x += 5; // x will be 7

y -= 3; // y will be 3

z = 10; z /= 5; // z will be 2

if (x == 7) { x++; } // x will be 8

if (x == 6 || y == 3) { y += 7 } // y will be 10

if (x == 6 && y == 3) { y += 7 } // y will remain 10
```

Data Types, Variables and Constants

Data Types

Data types are used to define the type of information (data) that is to be computed, acquired, modified, stored, sent, etc. Different data types use up different amounts of storage (memory). The most common data types include:

- byte : integer values between 0 and 255 (unsigned)
- char : ASCII character values. Ex: 'a', 'A', '1', '2', 'Z', '&'...
- int : integer values between -32768 to +32767 (signed)
- long : integer values between -2,147,483,648 to 2,147,483,647 (signed)
- boolean : true or false values
- float : decimal values between 3.4028235E+38 and -3.4028235E+38
- string : a list (array) of characters
- unsigned : prefix to make a int, long, etc. become just 0 and positive values

Variables

Variables are values of a certain data type that is used to store values that can change over time. In Arduino C, the variables can be defined using a data type prefix and can be initialized to a value or assigned a value later in the program. Examples:

```
int a = 10;

float b = 3.14159;

string name[] = "fau";

long bignum = 0;
bignum = 32000 * 32000;

ledON = false;

char sos[] = "SOS";
```

```
int a = 5;
int b = 10;
int c = a * b;

int r;
float pi = 3.14159;

r = a + b + c;

area = pi * r * r;
```

Constants

Constants are just like variables except that their values are fixed (constant) and cannot be changed. Constant

```
int r = 10;
const float pi = 3.14159;

area = pi * r * r;

pi = 2.7182 ; // this will give a compiler error
```

```
// Pre-defined constants

pinMode(13, OUTPUT);
digitalWrite(13, HIGH);

boolean ledOn = true;
boolean motorOn = false;
```

Constants take up memory space (ram). If you are just trying to have a word representation for a value, you can also use **#define**

```
#define ledPin 3
// The compiler will replace any mention of ledPin with the value 3 at compile time.
```

Array Variables

Arrays are variables that have a list of values where each value is accessed using an index (normally an integer)

```
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};
char message[6] = "hello";
```

```
int myArray[10]={9, 3, 2, 4, 3, 2, 7, 8, 9, 11};
// myArray[9]    contains 11
// myArray[10]   is invalid and contains random information (other memory address)
```

```
mySensVals[0] = 10;
```

```
x = mySensVals[4];
```

```
for (byte i = 0; i < 5; i = i + 1) {
  Serial.println(myPins[i]);
}
```

Functions

What are functions?

Functions are used to improve programming style by segmenting different tasks (or tasks that need to be repeated) into their own sections. This allows for breaking a larger problem down into sub-problems. Functions are used in arduino in a variety of ways including system functions like ***setup()*** and ***loop()*** for initialization and processing, user-written functions to split a larger task into subtasks and libraries of 3rd party written functions (methods) that extend the arduino ecosystem.

setup() and loop()

blink_two_leds

```
1
2  void setup() {
3    pinMode(13, OUTPUT);
4    pinMode(12, OUTPUT);
5  }
6
7
8  void loop() {
9    digitalWrite(13, HIGH);   // Turn on the LED
10   digitalWrite(12, LOW);   // Turn on the LED
11   delay(1000);              // Wait for two seconds
12   digitalWrite(13, LOW);    // Turn off the LED
13   digitalWrite(12, HIGH);    // Turn off the LED
14   delay(1000);              // Wait for two seconds
15 }
```

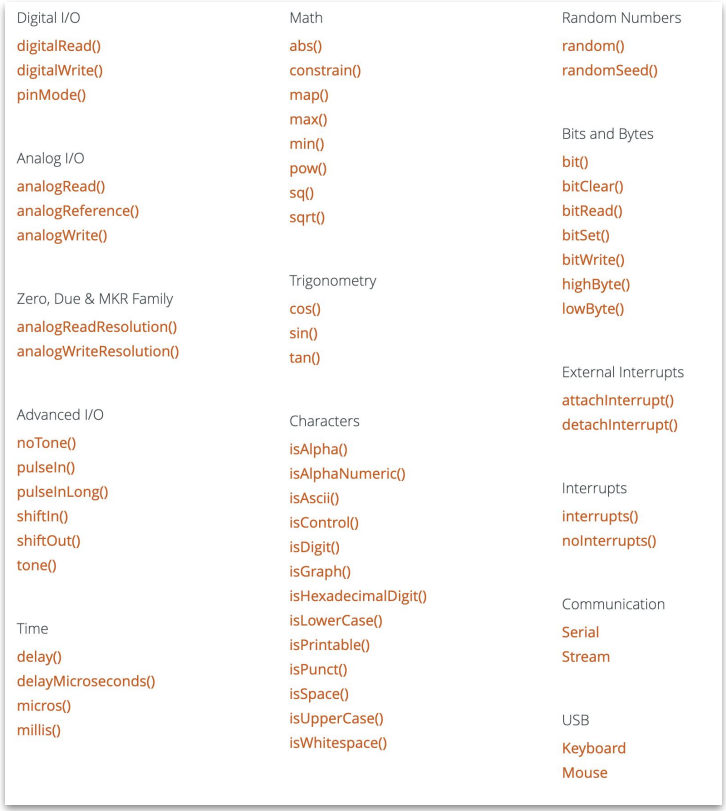
setup() is a function which is run (called) when the arduino first starts up or is reset (via reset button). It is run only once for each startup or reset.

loop() is a function which is run (called), It is called repeatedly forever, until the arduino is turned off or an exception (crash) occurs.

User functions

```
17 void loop()
18 {
19  // SOS = ... --- ...
20
21  dotBlink();
22  dotBlink();
23  dotBlink();
24  delay(CHARDELAY);
25
26  dashBlink();
27  dashBlink();
28  dashBlink();
29  delay(CHARDELAY);
30
31  dotBlink();
32  dotBlink();
33  dotBlink();
34  delay(EOMDELAY);
35
36 }
37
38 void dotBlink()
39 {
40   digitalWrite(LEDPIN, HIGH);  // Turn on the LED
41   delay(DOTDELAY);             // Leave LED on for a short (DOT) delay
42   digitalWrite(LEDPIN, LOW);   // Turn off the LED
43   delay(DOTDELAY);             // Leave LED on for a short (DOT) delay
44
45 }
46
47 void dashBlink()
48 {
49   digitalWrite(LEDPIN, HIGH);  // Turn on the LED
50   delay(DASHDELAY);            // Leave LED on for a long (DASH) delay)
51   digitalWrite(LEDPIN, LOW);   // Turn off the LED
52   delay(DASHDELAY);            // Leave LED on for a long (DASH) delay)
53 }
```

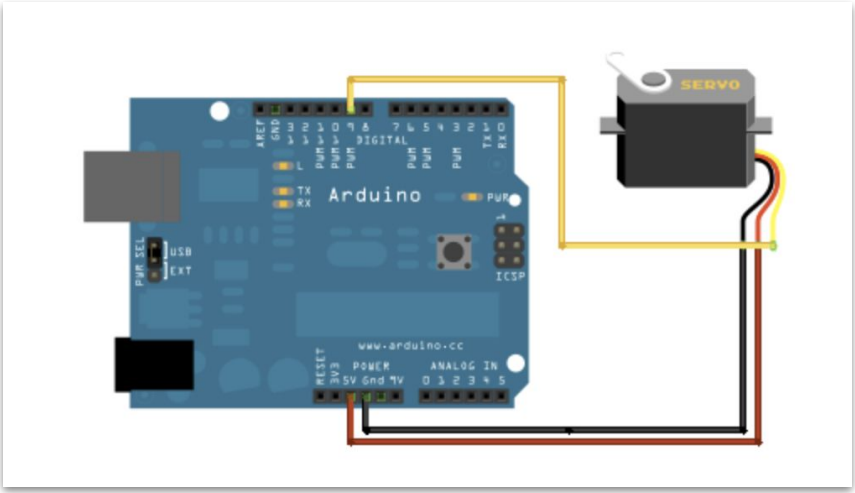
Arduino Core Library



Libraries

Sweep.ino

```
/* Sweep
 by BARRAGAN <http://barraganstudio.com>
 This example code is in the public domain.

 modified 8 Nov 2013
 by Scott Fitzgerald
 http://www.arduino.cc/en/Tutorial/Sweep
*/

#include <Servo.h>

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```



Methods

- attach()
- write()
- writeMicroseconds()
- read()
- attached()
- detach()

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

Let's put it all together!

Extending Blinking LEDs with some logic

```
//
// Medium Morse SOS demo
// 7/2020 - Mahesh Neelakanta
//

/*
   Added const, loops, if-else, comparison operators
   math operators, compound operators, data types
*/

#define DOTDELAY 200   // How long to wait per dot
#define DASHDELAY 350  // How long to wait per dash
#define CHARDELAY 500  // How long to wait after end of each character
#define EOMDELAY  1000 // How long to wait after end of message

const int LEDPIN = 13;
const char msg[] = "AB";


void dot()
{
  digitalWrite(LEDPIN, HIGH);  // Turn on the LED
  delay(DOTDELAY);             // Leave LED on for a short (DOT) delay
  digitalWrite(LEDPIN, LOW);   // Turn off the LED
  delay(DOTDELAY);             // Leave LED on for a short (DOT) delay
}

void dash()
{
  digitalWrite(LEDPIN, HIGH);  // Turn on the LED
  delay(DASHDELAY);            // Leave LED on for a long (DASH) delay)
  digitalWrite(LEDPIN, LOW);   // Turn off the LED
  delay(DASHDELAY);            // Leave LED on for a long (DASH) delay)
}
```

```
void setup()
{
  Serial.begin(115200);

  pinMode(LEDPIN, OUTPUT);
}

void loop()
{
  int index;
  char letter;

  Serial.print("Printing: ");
  Serial.print(msg);
  Serial.print(" Length=");
  Serial.println(sizeof(msg)-1);

  for(index = 0; index < sizeof(msg)-1 ; index++) {
    letter = msg[index];
    Serial.println(letter);

    if (letter == 'A' || letter == 'a') {
      dot(); dash();
    }
    else if (letter == 'B' || letter == 'b') {
      dash(); dot(); dot(); dot();
    }
    else if (letter == 'S' || letter == 's') {
       dot(); dot() ; dot();
    }
    else if (letter = 'O' || letter == 's') {
       dash(); dash(); dash();
    }
    else {
      // Unknown value so skip it
    }
    delay(CHARDELAY);
  }
  delay(EOMDELAY);
}
```

Q&A

Contact information : mahesh@fau.edu