List of Arduino APIs:

I2C

#include <Wire.h>

Wire.beginTransmission(addr); // Begin a transmission to the I2C servant device with the given address.
// Subsequently, queue bytes for transmission with the write() function
// and transmit them by calling endTransmission().

Wire.write(val); // Writes data from a servant device in response to a request from a
// master, or queues bytes for transmission from a master to servant
// device (in-between calls to beginTransmission() and
// endTransmission()).

Wire.endTransmission(); // Ends a transmission to a servant device that was begun by
// beginTransmission() and transmits the bytes that were queued by
// write().

Wire.requestFrom(addr, num) //Send read request for 'num' bytes to device with I2C address 'addr'

Wire.available() //Is data we've asked to read available on the I2C bus? Returns
//how many bytes are available.

Wire.read() // Reads a byte that was transmitted from a servant device to a master
// after a call to requestFrom() or was transmitted from a master to a
// servant. **This is a blocking transaction**. If a NACK is received, function returns 0.

Note: *Only the pins **A4 and A5** can be used as I2C pins. It is set automatically by the Wire library. **A4 is SDA and A5 is SCL**.*

I2C example

```
#include <Wire.h>
void setup()
{
  Wire.begin();         // join i2c bus (address optional for master)
  Serial.begin(9600);   // start serial for output
}

void loop()
{
  Wire.requestFrom(2, 6);    // request 6 bytes from servant device #2
  while(Wire.available())    // servant may send less than requested
  {
    char c = Wire.read();    // receive a byte as character
    Serial.print(c);         // print the character
  }

  delay(500);
}
```

Analog Read

Uno: operating voltage: 5V, usable pins: A0-A5, bits 10

analogRead(pin) //input is pin number (A0 to A5 on most boards), output is analog value on pin.

Analog Write

Uno: PWM pins 3, 5, 6, 9, 10, 11. PWM frequency 490 Hz (pins 5 and 6: 980 Hz)

analogWrite(pin, value) // pin to write to. value is the duty cycle: between 0 (always off) and 255 (always on)

Digital I/O

pinMode(pin, mode) //mode is INPUT, OUTPUT or INPUT_PULLUP

digitalWrite(pin, value) //Write value HIGH/LOW at GPIO 'pin'

digitalRead(pin) // Reads the value from a specified digital pin, either HIGH or LOW.

UART/Serial

serial.begin(speed) //initializes the UART to "speed" baud.

serial.read() // returns the first byte of incoming serial data (or -1 if not data is available)

serial.write(buf, len) // buf is an array of characters you wish to send. Len is how many bytes to send

Serial.print(78) gives "78" **Serial.print(1.23456)** gives "1.23"

Serial.print('N') gives "N" **Serial.print("Hello world.")** gives "Hello world."

Servo

servo.attach(pin) // Attach the Servo variable to a pin. Note that in Arduino 0016 and earlier,
// the Servo library supports servos on only two pins: 9 and 10.

servo.write(angle) // specifies an angle to write from 0 to 180.

Servo example

```
#include <Servo.h>

Servo myservo;

void setup()
{
  myservo.attach(9);
  myservo.write(90);  // set servo to mid-point
}

void loop() {}
```

SPI

Default SPI Pins on Arduino UNO: MOSI: GPIO 11; MISO: GPIO 12; CLK: GPIO 13; SS: GPIO 10

SPI.begin() : Initializes the SPI pins to SS = 1, SCLK = 0 , MOSI = 0;

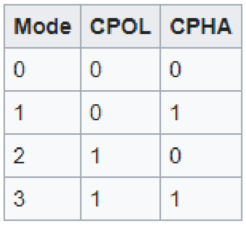
SPISettings my_spi_setting(speed, data order, mode):

my_spi_setting is global that contains the following after execution

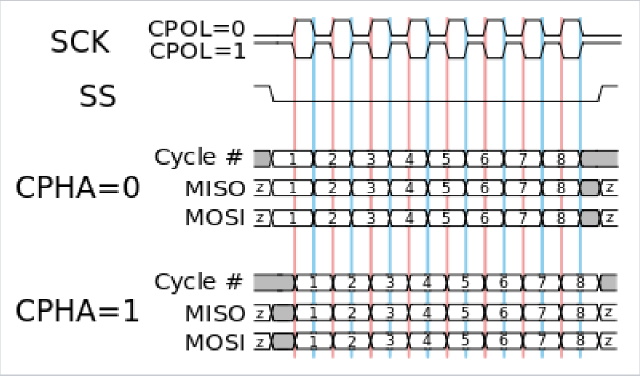
speed: integer expressed in Hz

data order: MSBFIRST or LSBFIRST

mode: SPI_MODE0, SPI_MODE1, SPI_MODE2, and SPI_MODE3



Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1



SPI.beginTransaction(SPI_settings):
Initializes the SPI bus with the settings in SPI_settings

SPI.endTransaction(): Ends a SPI transaction

receivedVal = **SPI.transfer**(val): Sends an 8-bit value on the SPI bus. At the same time it reads the value from the servant and returns the value.

SPI sample code:

```
#include <SPI.h>

       // Example with incompatible SPI devices (i.e they need different SPI_MODE
       const int servantAPin = 20;
       const int servantBPin = 21;

       // set up the speed, data order and data mode
       SPISettings settingsA(2000000, MSBFIRST, SPI_MODE1);
       SPISettings settingsB(16000000, LSBFIRST, SPI_MODE3);

       void setup() {
         // set the Servant Select Pins as outputs and drive them high.
         pinMode (servantAPin, OUTPUT); digitalWrite (servantAPin, HIGH);
         pinMode (servantBPin, OUTPUT); digitalWrite (servantBPin, HIGH);
         SPI.begin();
       }

       uint8_t stat, val1, val2, result;

       void loop() {
         // read three bytes from device A
         SPI.beginTransaction(settingsA);   digitalWrite (servantAPin, LOW);
         // reading only, so data sent does not matter
         stat = SPI.transfer(0);   val1 = SPI.transfer(0);   val2 = SPI.transfer(0);
         digitalWrite (servantAPin, HIGH);
         SPI.endTransaction();
         // if stat is 1 or 2, send val1 or val2 else zero
         if (stat == 1) {
          result = val1;
         } else if (stat == 2) {
          result = val2;
         } else {
          result = 0;
         }
         // send result to device B
         SPI.beginTransaction(settingsB);
         digitalWrite (servantBPin, LOW);
         SPI.transfer(result);
         digitalWrite (servantBPin, HIGH);
         SPI.endTransaction();
       }
```