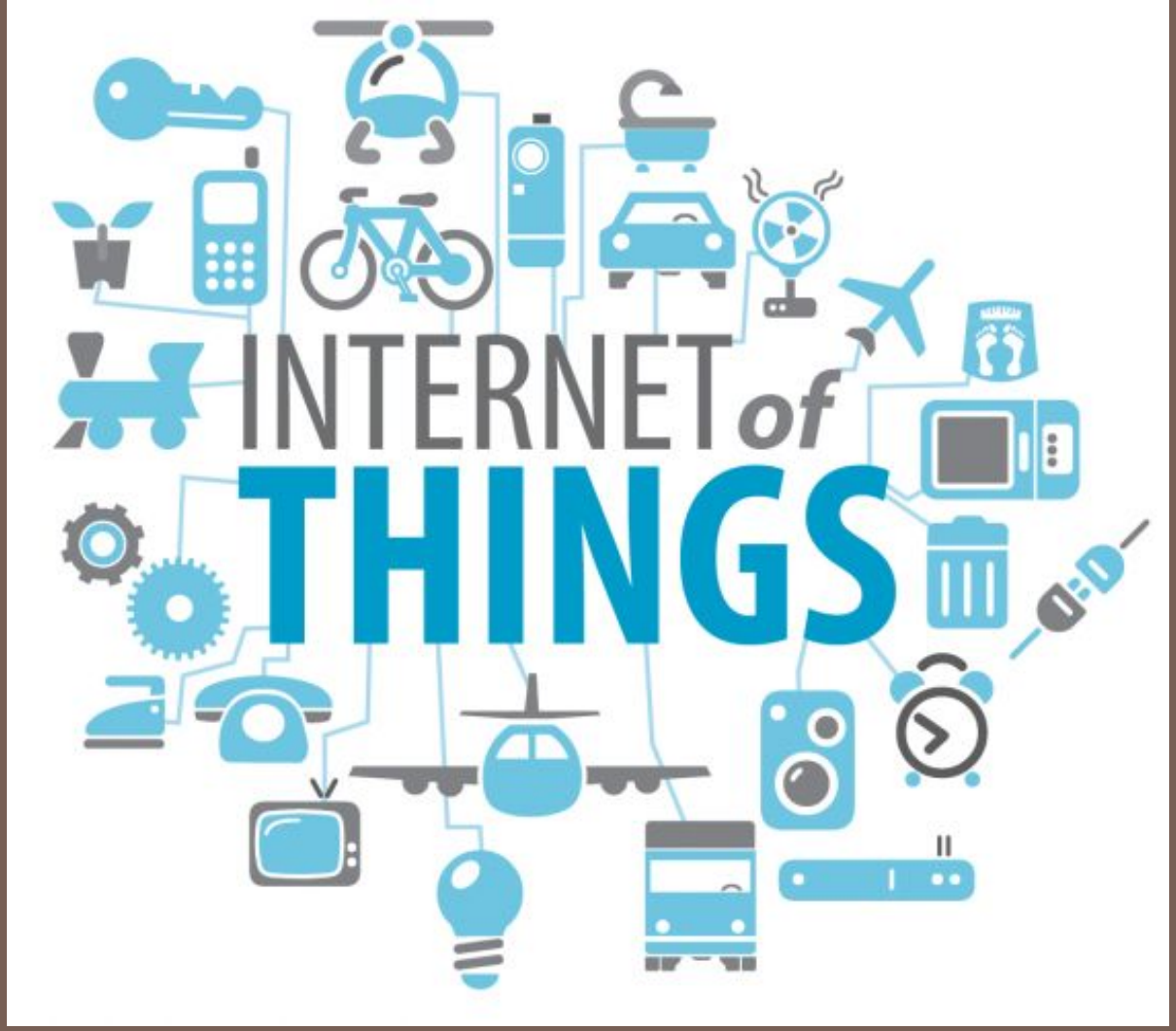


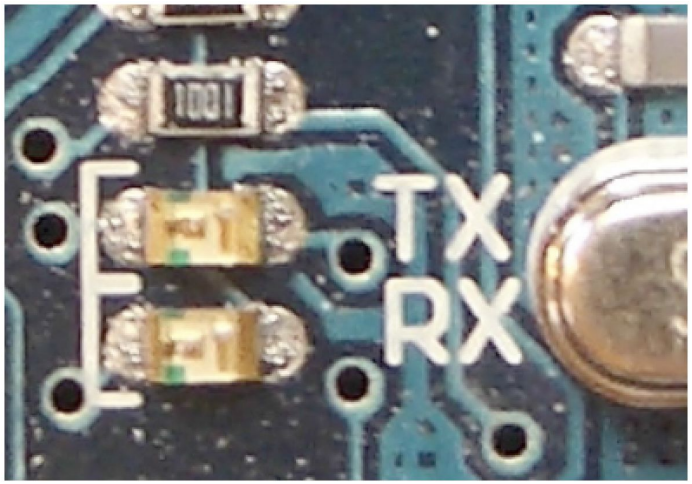
IMPLEMENTING IOE

Week 3

Assist. Prof. Rassim Suliyev - SDU 2017

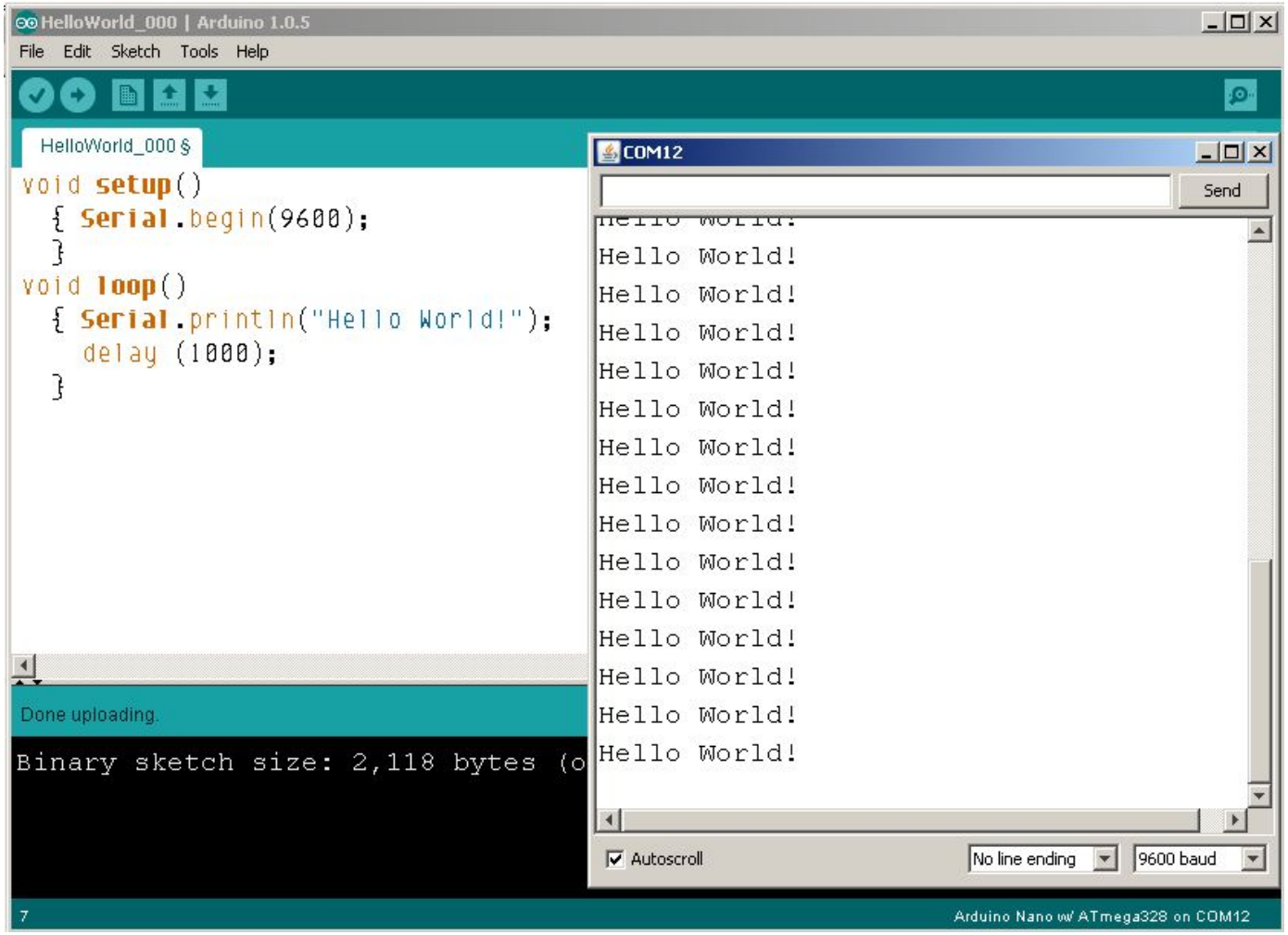
Communicating with Others

- Arduino can use same USB cable for programming and to talk with computers
- Talking to other devices uses the “Serial” commands
- TX – sending to PC
- RX – receiving from PC



Serial Communications

- Sends “Hello world!” to your computer
- Click on “Serial Monitor” button to see output



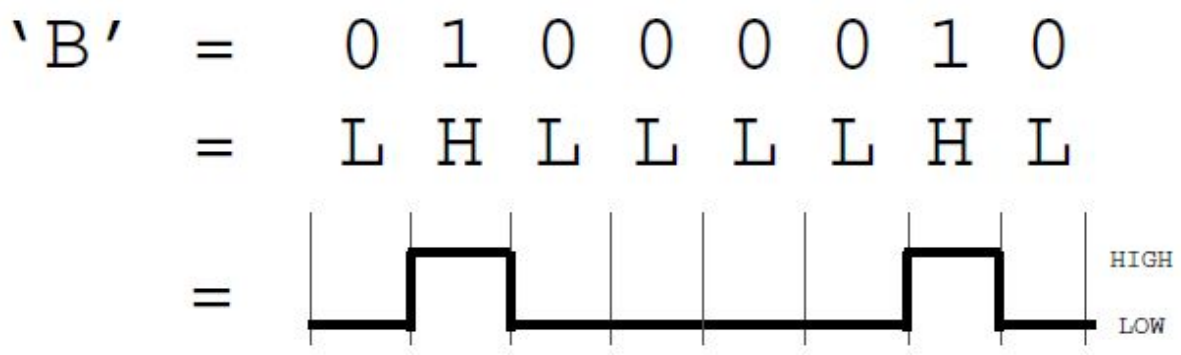
Arduino Communications

- Is just serial communications
- Arduino doesn't really do USB
- It really is "serial", like old RS-232 serial
- All microcontrollers can do serial
- Not many can do USB
- Serial is easy, USB is hard

serial terminal from the old days

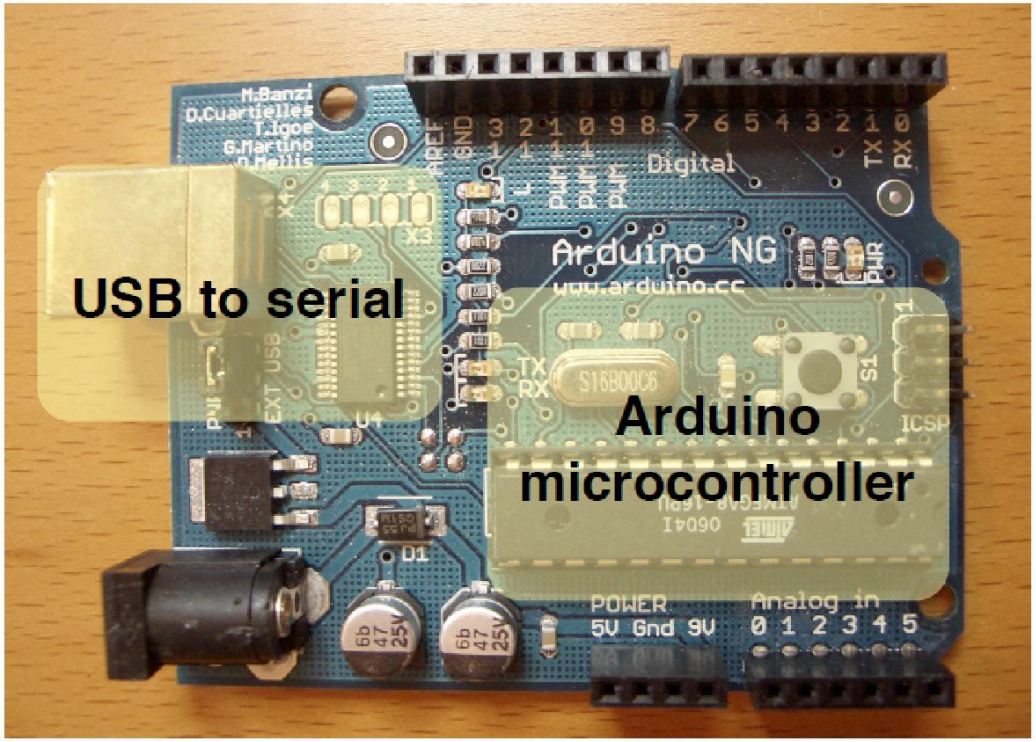
Serial Communications

- "Serial" because data is broken down into bits, each sent one after the other down a single wire.
- The single ASCII character 'B' is sent as:



- Toggle a pin to send data, just like blinking an LED
- You could implement sending serial data with `digitalWrite()` and `delay()`
- A single data wire needed to send data. One other to receive.

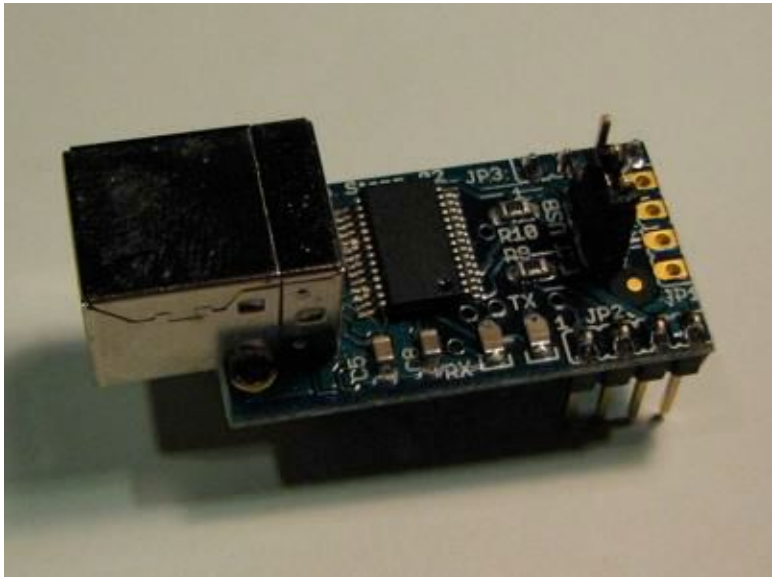
Arduino & USB-to-serial



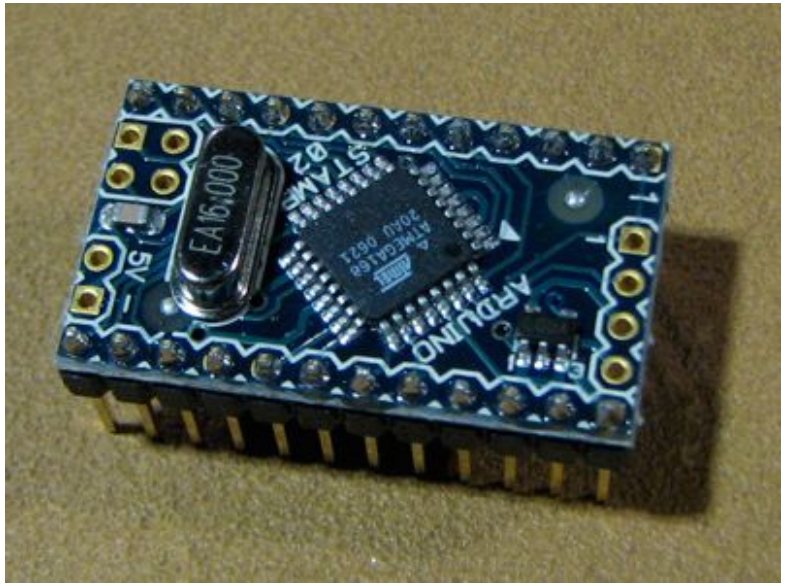
- A standard Arduino has a single hardware serial port
- But serial communication is also possible using software libraries to emulate additional ports

Arduino Mini

- Arduino Mini separates the two circuits



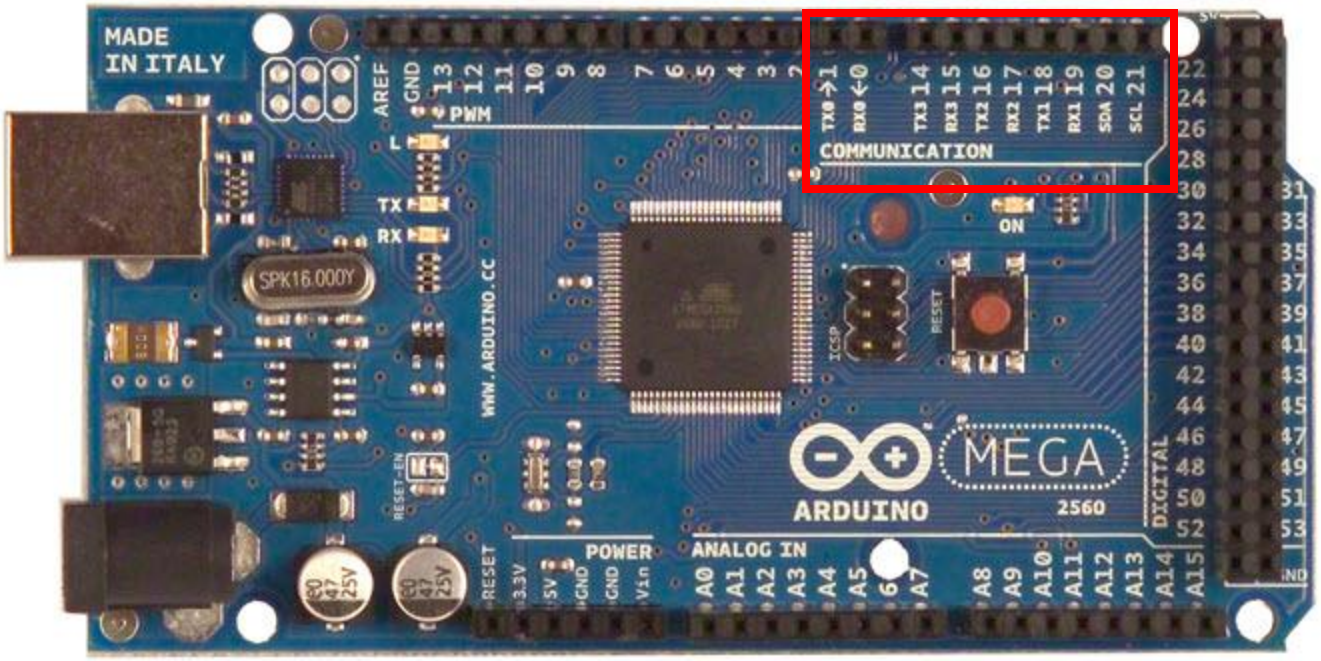
Arduino Mini USB adapter



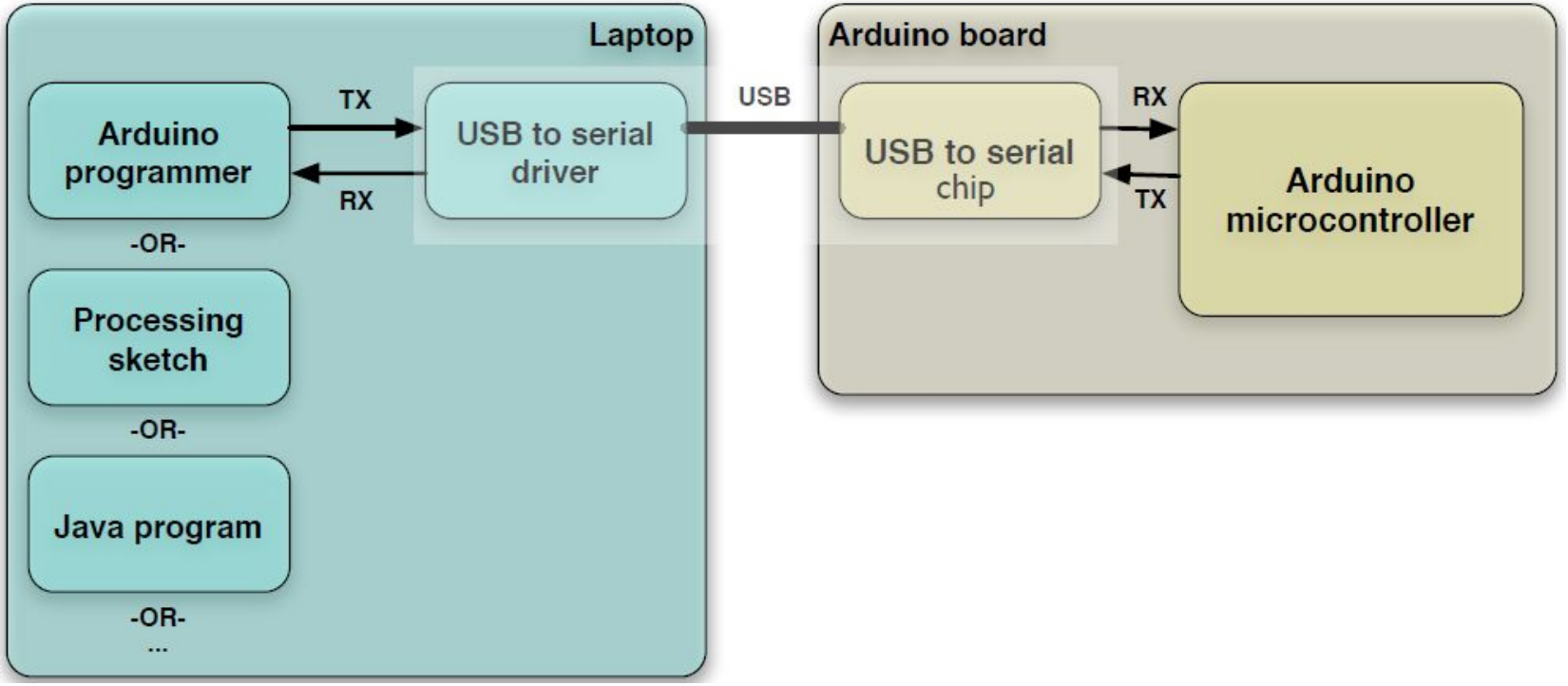
Arduino Mini

Arduino Mega

- The Arduino Mega has four hardware serial ports
- Only one of these has a USB adapter built in



Arduino to Computer



- USB is totally optional for Arduino, but it makes things easier
- Original Arduino boards were RS-232 serial, not USB
- All programs that talk to Arduino (even the Arduino IDE) think that they're talking via a serial port

Arduino & USB

- Since Arduino is all about serial, and not USB, Interfacing to things like USB flash drives, USB hard disks, USB webcams, etc. is not possible
- Also, USB is a host/peripheral protocol. Being a USB "host" means needing a lot of processing power and software, not something for a tiny 8kB microcontroller. It can be a peripheral. In fact, there is an open project called "AVR-USB" that allows AVR chips like used in Arduino to be proper USB peripherals

Serial Message Protocol

- Where each message begins and ends?
- Sides must agree how information is organized in the message (*communications protocol*)
- Header - one or more special characters that identify the start of message
- Footer - one or more special characters that identify the end of message

Sending Debug Information from Arduino to Your Computer

- This sketch prints sequential numbers on the Serial Monitor:

```
void setup(){
  Serial.begin(9600); // send and receive at 9600 baud
}
int number = 0;
void loop(){
  Serial.print("The number is ");
  Serial.println(number); // print the number
  delay(500); // delay half second between numbers
  number++; // to the next number
}
```

- Output is:

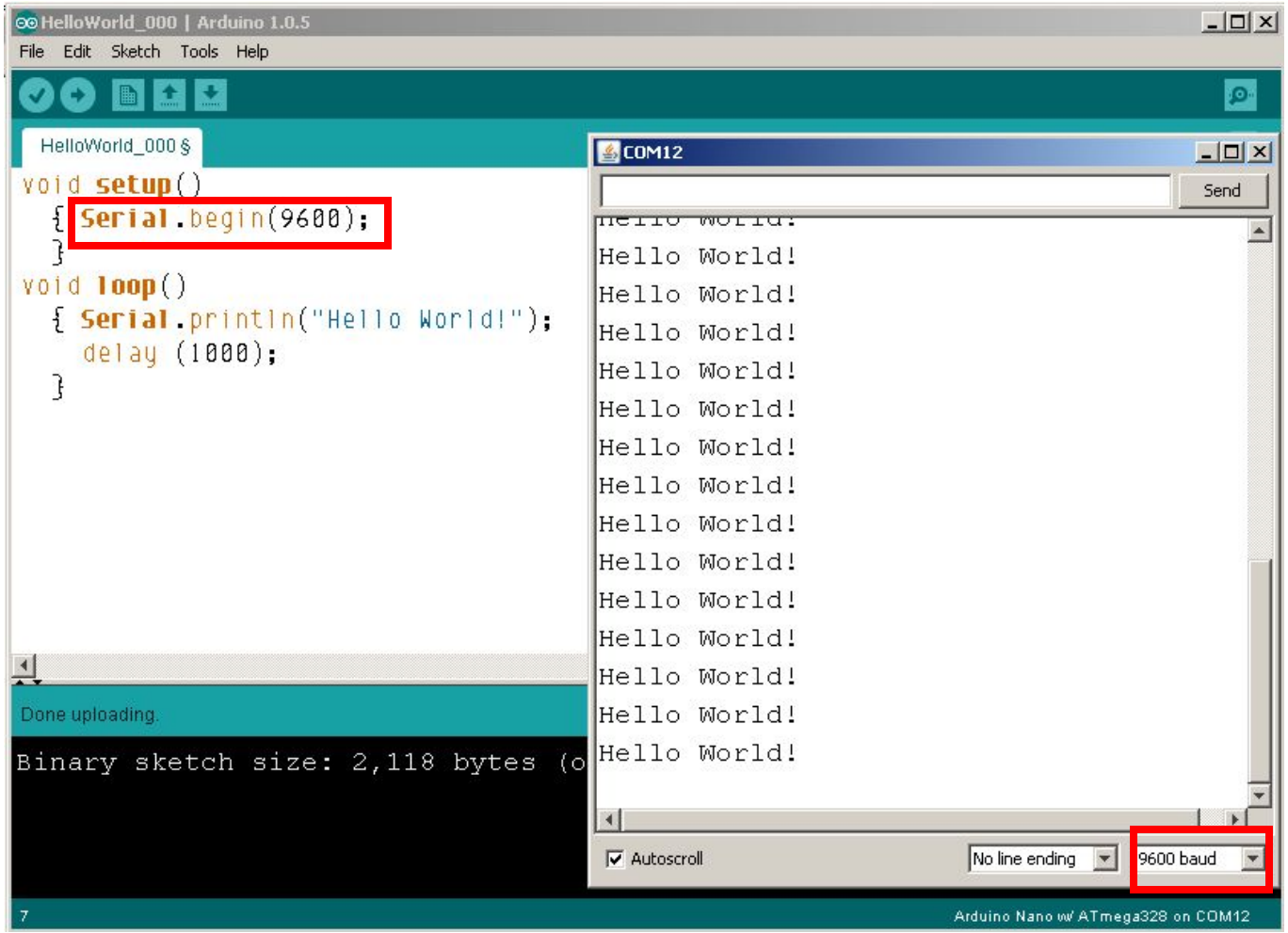
The number is 0
The number is 1
The number is 2

Baud rate

First call the `Serial.begin()`

The function takes a single parameter: the desired communication speed (baud).

You must use the same speed for the sending side and the receiving side.



baud is a measure of the number of bits transmitted per second

Sending information

- You can display text using the `Serial.print()` or `Serial.println()` function

```
int val = 123;
Serial.print(val);      // sends 3 ASCII chars "123"
Serial.print(val,DEC);  // same as above
Serial.print(val,HEX);  // sends 2 ASCII chars "7B"
Serial.print(val,BIN);  // sends 8 ASCII chars "01111011"
Serial.print(val,BYTE); // sends 1 byte, the verbatim value
```

- `println()` – prints the data followed by a carriage return character and a newline character
- These commands can take many forms
- Numbers are printed using an ASCII character for each digit
- Floats are similarly printed as ASCII digits, defaulting to two decimal places
- Bytes are sent as a single character
- Characters and strings are sent as is

Strings

- **String message = "This string"; //C++ type strings**
  - **message.length()**
    //provides thenumber of characters) in the string
  - **message.concat(anotherMessage)**
    //appends the contents of anotheMessage to message (also + operator)
  - **message.substring(s, e);**
    //returns a substring starting from s till e
  - You can use the **indexOf** and **lastIndexOf** functions to find an instance of a particular character in a string

- **char message[8] = "Arduino"; //C type string**
  - **int length = strlen(message);**
    // return the number of characters in the string
  - **strcpy(destination, source);**
    // copy string source to destination
  - **strcat(destination, source);**
    // append source string to the end of the destination string
  - **if(strcmp(str, "Arduino") == 0)**
    // do something if the variable str is equal to "Arduino"

Comparing C type Strings

```
char str1[ ] = "left";
char str2[ ] = "right";
if(strcmp(str1, str2) == 0)
  Serial.print("strings are equal)


strcmp("left", "leftcenter") == 0)
// this will evaluate to false


strncmp("left", "leftcenter", 4) == 0)
// this will evaluate to true
```

String Object

- charAt(n) or [n] - Access a particular character of the String
- concat(parameter) or + - Appends the parameter to a String
- endsWith(string2) - Tests whether or not a String ends with string2
- equals(string2) or == - Compares two strings for equality (case sensitive)
- indexOf(val, [strt]) – locates val in a String by searching forward starting from strt index. To search backward use lastIndexOf(val,[strt])
- length() - Returns the length of the String, in characters
- remove(index,[count]) – remove all characters (or count caracters if given) from a String starting from index
- replace(substring1, substring2) – Replace all instances of substring1 in a String to substring2
- setCharAt(index, c) - Sets a character to c at index of the String
- startsWith(string2) - Tests whether or not a String starts with the string2
- substring(from, [to]) - Get a substring of a String, from - inclusive, to – exclusive
- toInt() or toFloat() - Converts a valid String to an integer or float
- toLowerCase() or toUpperCase() - Get a lower-case or upper-case version of a String
- trim() - Get a version of the String with any leading and trailing whitespace removed

Mathematical Operators

```
int myValue;
myValue = 1 + 2;  // addition
myValue = 3 - 2;  // subtraction
myValue = 3 * 2;  // multiplication
myValue = 3 / 2;  // division (the result is 1)

int value =   1 + 2 * 3 + 4;


int myValue0 =  20 % 10;  // get the modulus(remainder) of 20 divided by 10
int myValue1 =  21 % 10;  // get the modulus(remainder) of 21 divided by 10


int x = analogRead(0);
int y = analogRead(1);

if(abs(x-y) > 10)
{
  Serial.println("The analog values differ by more than 10");
}
```

Comparing Character and Numeric Values

Operator	Test for	Example
==	Equal to	`2 == 3 // evaluates to false`
!=	Not equal to	`2 != 3 // evaluates to true`
>	Greater than	`2 > 3 // evaluates to false`
<	Less than	`2 < 3 // evaluates to true`
>=	Greater than or equal to	`2 >= 3 // evaluates to false`
<=	Less than or equal to	`2 <= 3 // evaluates to true`

Logical and Bitwise operators

- **Logical operators**

Symbol	Function	Comments
&&	Logical And	Evaluates as `true` if the condition on both sides of the && operator are true
||	Logical Or	Evaluates as `true` if the condition on at least one side of the || operator is true
!	Not	Evaluates as `true` if the expression is false, and `false` if the expression is true

- **Bitwise operators**

Symbol	Function	Comment	Example
&	Bitwise And	Sets bits in each place to 1 if both bits are 1; otherwise, bits are set to 0.	`3 & 1` equals `1` (`11 & 01` equals `01`)
|	Bitwise Or	Sets bits in each place to 1 if either bit is 1.	`3 | 1` equals `3` (`11 | 01` equals `11`)
^	Bitwise Exclusive Or	Sets bits in each place to 1 only if one of the two bits is 1.	`3 ^ 1` equals `2` (`11 ^ 01` equals `10`)
~	Bitwise Negation	Inverts the value of each bit. The result depends on the number of bits in the data type.	`~1` equals `254` (`~00000001` equals `11111110`)

Combining Operations and Assignment

Operator	Example	Equivalent expression
+=	`Value += 5;`	`Value = Value + 5; // add 5 to Value`
-=	`Value -= 4;`	`Value = Value - 4; // subtract 4 from Value`
*=	`Value *= 3;`	`Value = Value * 3; // multiply Value by 3`
/=	`Value /= 2;`	`Value = Value / 2; // divide Value by 2`
>>=	`Value >>= 2;`	`Value = Value >> 2; // shift Value right two places`
<<=	`Value <<= 2;`	`Value = Value << 2; // shift Value left two places`
&=	`Mask &= 2;`	`Mask = Mask & 2; // binary and Mask with 2`
|=	`Mask |= 2;`	`Mask = Mask | 2; // binary or Mask with 2`

Advanced Mathematical Operators

```
myConstrainedValue  = constrain(myValue, 100, 200);
```

`min(x,y)` returns the smaller of two numbers. `max(x,y)` returns the larger

`pow(x, y)` returns the value of `x` raised to the power of `y`

```
myValue =  pow(3,2);      float s = pow(2, 1.0 / 12); // the twelfth root of two
```

The `sqrt(x)` function returns the square root of `x`

```
float deg = 30;
float rad  = deg * PI / 180;
Serial.println(rad);
Serial.println sin(rad));
Serial.println (cos(rad));
```

Other Useful Operators

```
randomSeed(1234);  // change the starting sequence of random numbers.
random(max);       // returns a random number between 0 and max -1
random(min, max); // returns a random number between min and max -1
```

`bitSet(x, bitPosition)` sets (writes a 1 to) the given `bitPosition` of variable `x`.

`bitClear(x, bitPosition)` clears (writes a 0 to) the given `bitPosition` of variable `x`.

`bitRead(x, bitPosition)` returns the value (as 0 or 1) of the bit at the given `bitPosition` of variable `x`.

`bitWrite(x, bitPosition, value)` sets the given value (as 0 or 1) of the bit at the given `bitPosition` of variable `x`.

`bit(bitPosition)` returns the value of the given bit position: `bit(0)` is 1, `bit(1)` is 2, `bit(2)` is 4, and so on.

<< (bit-shift left) and >> (bit-shift right) operators

Functions

```
type functionName(parameters)
{
        statements;
}
```

Functions are declared by first declaring the function type. This is the type of value to be returned by the function such as 'int' for an integer type function. If no value is to be returned the function type would be void. After type, declare the name given to the function and in parenthesis any parameters being passed to the function.

```
int delayVal()
{
      int v; // create temporary variable 'v'
      v = analogRead(pot); // read potentiometer value
      v /= 4; // converts 0-1023 to 0-255
      return v; // return final value
}
```

Receiving Serial Data in Arduino

- Serial.available() - Get the number of bytes (characters) available for reading from the serial port.
- This is data that's already arrived and stored in the serial receive buffer (which holds 64 bytes)
- Serial.read() - Reads incoming serial data
- Serial.readBytes(buffer, length) - reads characters from the serial port into a buffer. The function terminates if the determined length has been read, or it times out

Controlling Arduino

```
int ledPin = 13;   // choose a pin for LED
int val = 0;       // variable to store the data received via Serial port
void setup() {
  pinMode(ledPin,OUTPUT);    // make ledPin an output
  Serial.begin(9600);        // initialize the Serial port
}
void loop () {
  // Serial.available() – is a method to see whether something is
  // received or not via Serial port without pausing the main program
  if( Serial.available() ) {
    val = Serial.read();      // read the value received via Serial port
    if( val == 'H' ) {        // if ‘H’, then blink
      digitalWrite(ledPin, HIGH);
      delay(1000);
      digitalWrite(ledPin, LOW);
    }
  }
}
```

Tasks Part 1

- Concatenate two strings (Name, Surname) with space between them and output to serial monitor
- Find a number of spaces in a given text
- Given a string “Name Surname Age”, divide it to 3 strings
- Convert a String containing a number to a number
- Write function that compares 2 numbers and returns -1 if a<b, 0 if a=b, 1 if a>b
- Write function that returns minimum number from an array of integers

Tasks Part 2

- Write function that sorts array of integers
- Read the number N from Serial port and make LED blink N times
- Read numbers N and M and return N to the power of M

Home Work

- Given 2 strings A and B. A contains some text and B contains a number. Print B times A.
- Write function that compares two c type strings
- Write function that returns both minimum and maximum number from an array of integers
- Read N from Serial port and return N'th Fibonacci number
- Read N from Serial port then read N numbers into array, print sorted array
- Read a character from Serial port and print its ASCII value
- Read a String from Serial port then append "-OK" to it and print the resulting string