

Arduino Programming :

Serial Communication

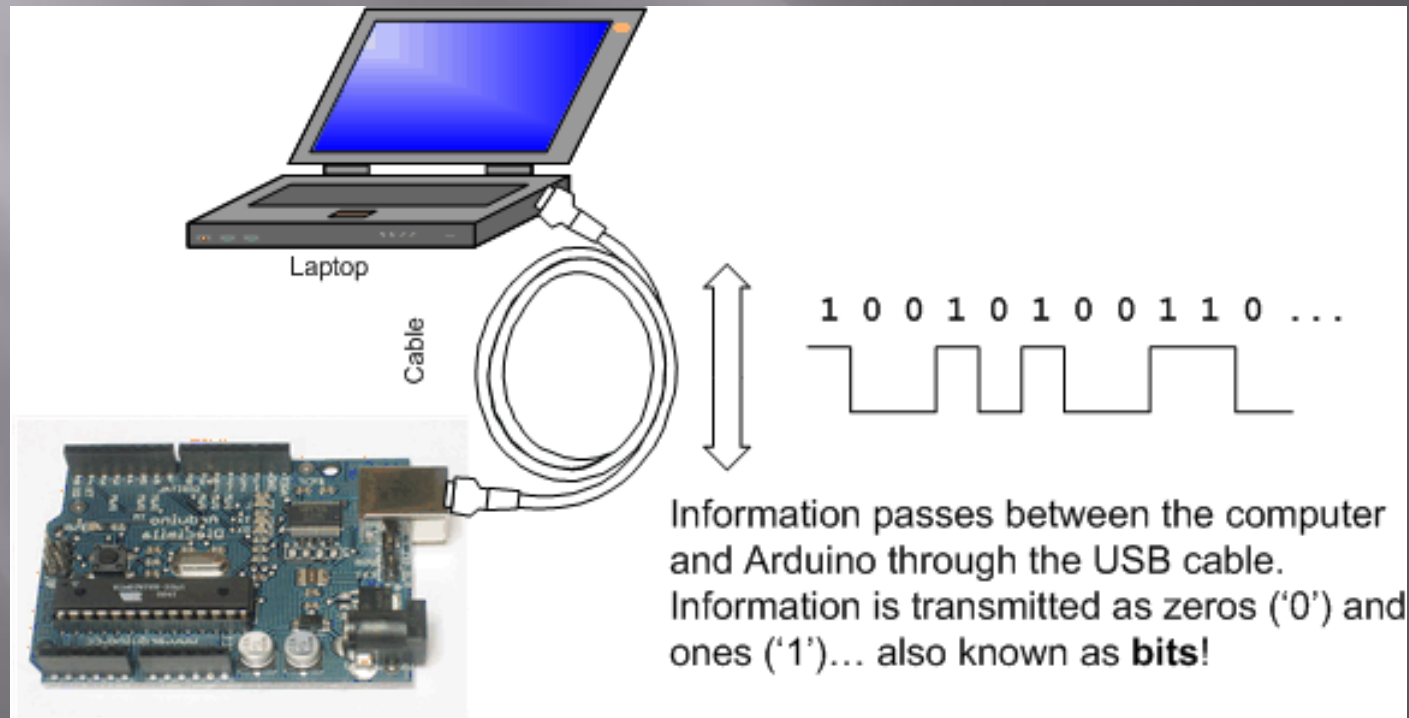
Objectives

- ▣ Understand how to send and receive data between the Arduino and computer
- ▣ Understand how to format characters to numbers

Serial communication

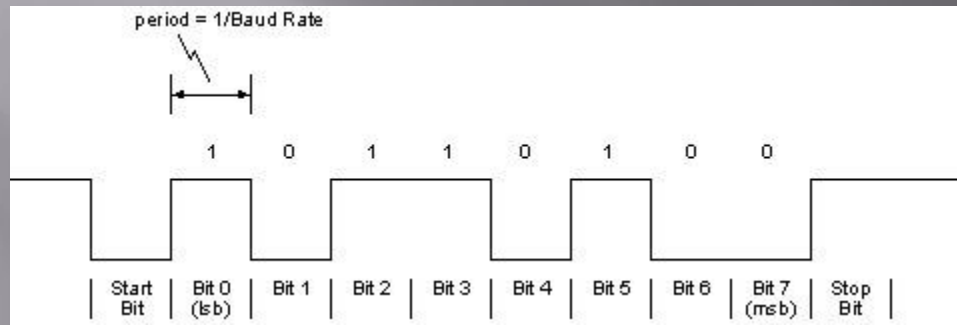
Communication to the Arduino Uses the Rx and Tx pins

- Rx = Receive, Tx = Transmit
- The Arduino uses a UART to transfer data
 - (Universal Asynchronous Receiver / Transmitter)



Data is sent at a specific rate

The speed that data is transmitted is often called the 'baud rate' and is in units of bits per second




- Some common “slow” baud rates:
 - 9600 19.2KB 38.4KB

Sending data from the Arduino to the computer screen:

Pretty easy to do!


```
void setup()
{
  Serial.begin(9600);
}
```

Initialize the serial communication at 9600 baud in setup()

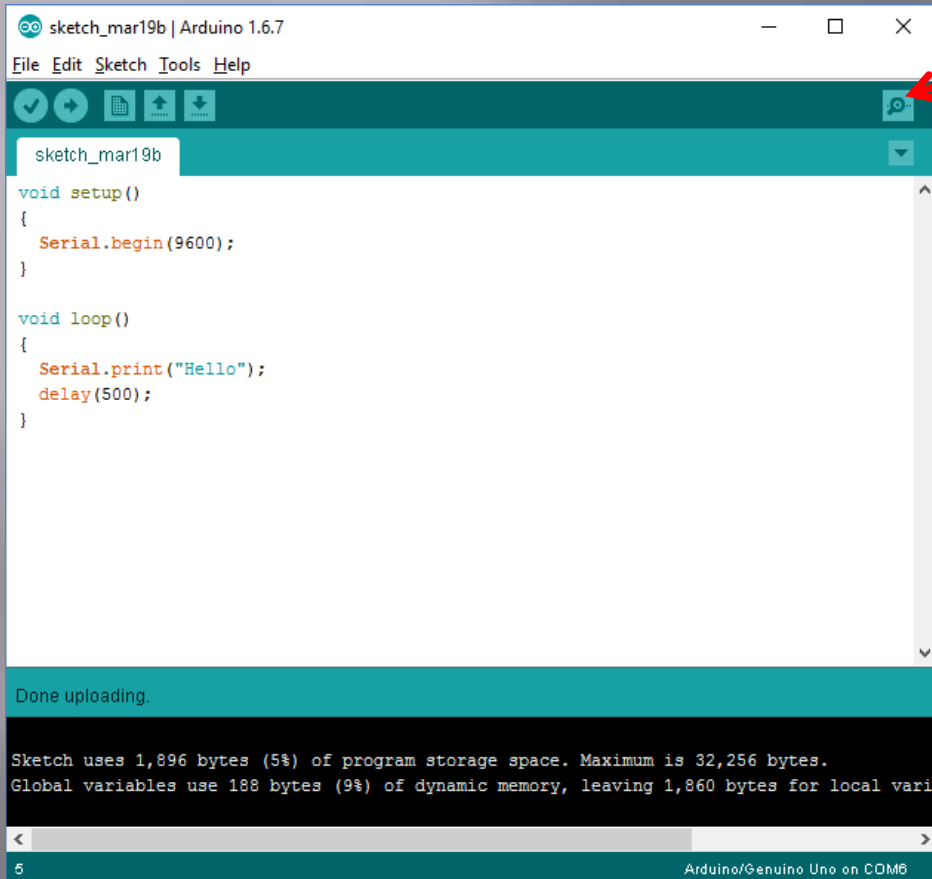


```
void loop()
{
  Serial.print("Hello");
  delay(1000);
}
```

Serial.print sends a "string" (set of characters) to the Arduino to the computer screen



To see what's on the screen click on the serial monitor button

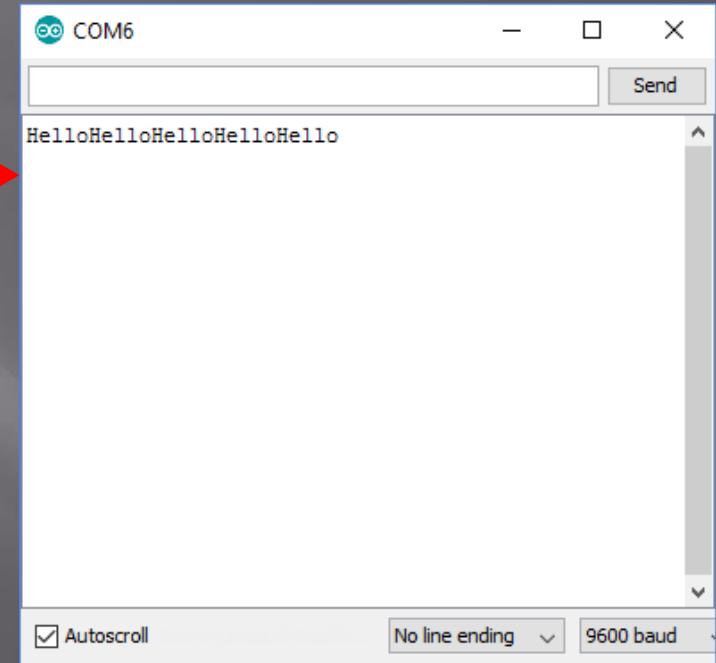


The screenshot shows the Arduino IDE interface. The main window is titled "sketch_mar19b | Arduino 1.6.7". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for saving, running, uploading, and serial monitor. The serial monitor icon, which looks like a speech bubble with a magnifying glass, is highlighted by a red arrow. The sketch editor shows the following code:

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.print("Hello");
  delay(500);
}
```

At the bottom of the IDE, there is a status bar that says "Done uploading." and a console window showing memory usage information: "Sketch uses 1,896 bytes (5%) of program storage space. Maximum is 32,256 bytes. Global variables use 188 bytes (9%) of dynamic memory, leaving 1,860 bytes for local variables." The status bar also indicates "5" and "Arduino/Genuino Uno on COM6".



The screenshot shows the serial monitor window titled "COM6". It has a "Send" button at the top right. The main area displays the output "HelloHelloHelloHelloHello". At the bottom, there are settings: "Autoscroll" is checked, "No line ending" is selected, and the baud rate is set to "9600 baud". A red arrow points from the serial monitor icon in the IDE to this window.

Formatting a new line:

By changing the `Serial.print()` command to `Serial.println()` a carriage return is inserted:

```
HelloHelloHelloHello // using Serial.print
```

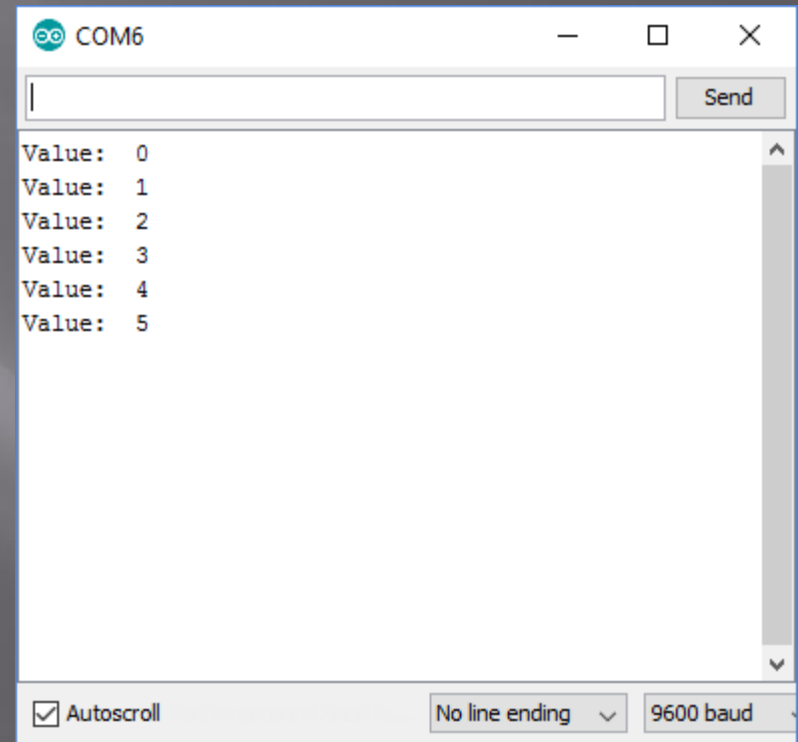
```
Hello // Using Serial.println  
Hello  
Hello
```

Tabs are a great way to separate text from data

```
int count;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  for (count = 0; count < 10; count++)
  {
    Serial.print("Value:");
    Serial.print("\t");
    Serial.println(count);
    delay(500);
  }
}
```



Uses of Serial.print()

- We can send data to the screen which can be used as a simple interface.
- We can also use it to “debug” (analyze) variables in our program.
- For example, if we are trying to read an analog input we can send that value to the screen to see the values.

Sending Characters to the Arduino

- We can also send characters (letters, numbers, symbols) to the Arduino from the computer keyboard.
- This is done using the `Serial.read()` function.
- The data is received when the Enter button is pressed on the keyboard.
- However, since the Arduino doesn't know when data is coming, we need to setup a loop using a function called `Serial.available()`
- This allows us to wait until data has been received.

Simple receive character code

```
if (Serial.available() > 0 ) // Wait for input...
{
    charIn = Serial.read(); // Read a character
    if (charIn == 'F')
        digitalWrite (LED, OFF);
}
```

Sending Data to the Arduino

- One issue is that the Arduino only receives text characters from the computer. Often we want to read a numerical value.
- To do this, we need to convert between characters and numbers.
- This is done using the `Serial.parseInt()` function.
- Let's put this all together...

```
int value;
```

```
void setup()
```

```
{  
  Serial.begin(9600);  
}
```

```
void loop()
```

```
{  
  if (Serial.available() > 0) // \n  
  {  
    value = Serial.parseInt(); // \n  
    Serial.print("First Value: ");  
    Serial.print("\t");  
    Serial.println(value);  
    Serial.print("Doubled Value: ");  
    Serial.print("\t");  
    Serial.println(value * 2);  
  }  
}
```

