Arduino Basics

An Arduino tutorial blog. Free Arduino tutorials for everyone !

4 July 2012

Simple Arduino Serial Communication

This Tutorial is progressive and will be updated from time to time. The goal is to start from a very basic form of Arduino Serial communication, and progressively add or improve components so that we can ultimately transmit data from one computer to another using an XBee.
Please note: I am not an expert, but am happy to share what I have learned. The Arduino forums are a great place to ask questions, feel free to link to this blog (if required).

Let us begin.

Stage 1: ECHO ECHO

Parts Required:

- Computer
- USB cable
- Arduino UNO (or equivalent)
- Arduino IDE

The following code will make the Arduino ECHO anything you send to it. Therefore, if you type a 3, the Arduino will send back a 3. If you type a letter F, the Arduino will send back a letter F.
Enter the following code into your Arduino IDE and upload it to your Arduino.

Arduino Sketch

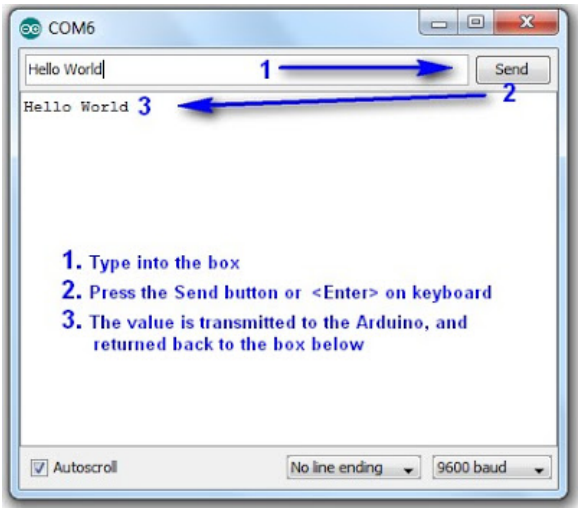
```
/* Simple Serial ECHO script : Written by ScottC 03/07/2012 */

/* Use a variable called byteRead to temporarily store
   the data coming from the computer */
byte byteRead;

void setup() {
// Turn the Serial Protocol ON
  Serial.begin(9600);
}

void loop() {
   /*  check if data has been sent from the computer: */
  if (Serial.available()) {
    /* read the most recent byte */
    byteRead = Serial.read();
    /*ECHO the value that was read, back to the serial port. */
    Serial.write(byteRead);
  }
}
```

The above code was formatted using this site

Instructions

1. Once the Arduino sketch has been uploaded to the Arduino. Open the Serial monitor, which looks like a magnifying glass at the top right section of the Arduino IDE. Please note, that you need to keep the USB connected to the Arduino during this process, as the USB cable is your communication link between your computer and the Arduino.

2. Type anything into the top box of the Serial Monitor and press <Enter> on your keyboard. This will send a series of bytes to the Arduino. The Arduino will respond by sending back your typed message in the larger textbox.



3. Please note that we are using **Serial.write(byteRead);** on line 18 to get the Arduino to ECHO the message back to you on your computer.

Things to Try

1. Delete lines 16 to 18, and replace them with the following line :

```
Serial.write(Serial.read());
```

This essentially eliminates the byteRead variable in the sketch above. But we will be using it later on, so once you have tested it out, put the code back together as originally displayed.

---

2. Replace line 18 with a **Serial.println** instead of **Serial.write**

```
Serial.println(byteRead);
```

Once uploaded, type **1** <enter> **2** <enter> **3** <enter> into the Serial Monitor.
You should see:

49
50
51

Serial.print and Serial.println will send back the actual ASCII code, whereas Serial.write will send back the actual text. See ASCII codes for more information.

---

3. Try typing in numbers like 1.5 or 2.003 or -15.6 into the Serial Monitor using Serial.write and Serial.print or Serial.println commands as described before.

You will notice that the decimal point transmits as a number using Serial.print or Serial.println, and will transmit as a decimal point when using Serial.write

Let us say that you have number pairs that you want the Arduino to interpret. How do you separate the numbers? The answer is Delimiters.
You may be familiar with CSV (comma separated value) files, where each field is separated by a comma (,). The comma is a useful way of separating or grouping information.

Lets say you have the following stream of numbers:
12345678910

How will your Arduino know if this is a single number, or a series of numbers?
Eg:

12,34,56,78,91,0
123,456,78,910
1,2,3,4,5,6,7,8,9,10
12345678910

The comma **delimiters** help to identify how the numbers should be interpreted.

In the echo example in **Stage 1** above, you would have noticed that when we used **Serial.print(byteRead);** that the values displayed one after another in a similar fashion to 12345678910.

You would have also noticed that **Serial.println(byteRead);** provided a line break between each value sent. And depending on the numbers sent, it could have looked like this:
12
34
56
78
91
0

The Serial.println() function essentially uses a line feed to separate the values being sent. This line break can be used as a delimiter, but we will look at that later on. For now we will concentrate on using a comma (,).

We will now get the Arduino to "listen" for the comma to help it identify a new number.
According to the ASCII code site, a comma is equal to byte code **44**. So when the Arduino reads a byte code that is equal to 44, we will get it to print a line feed.

Enter the following sketch into your Arduino IDE and upload it to your Arduino.

Arduino Sketch

```
/* Simple Serial ECHO script : Written by ScottC 04/07/2012 */
/* Stage 2 : Delimiters */

/* Use a variable called byteRead to temporarily store
   the data coming from the computer */
byte byteRead;

void setup() {
// Turn the Serial Protocol ON
  Serial.begin(9600);
}

void loop() {
   /*  check if data has been sent from the computer: */
  if (Serial.available()) {
    /* read the most recent byte */
    byteRead = Serial.read();

    /*Listen for a comma which equals byte code # 44 */
    if(byteRead==44){
      Serial.println();
    }else{
      /*ECHO the value that was read, back to the serial port. */
      Serial.write(byteRead);
    }
  }
}
```

The above code was formatted using this site

Instructions

1. Once the code has been uploaded to the Arduino, open the Serial Monitor once again and type the following sequence of numbers:

1 <enter> 2 <enter> 3 <enter>

You should see the Serial monitor display the following number: 123

---

2. While still in the serial monitor, type the following:

, <enter> 1 <enter> , <enter> 2 <enter> , <enter> 3 <enter> ,

Please note the commas between each numerical entry. You should now see a pattern like this:
1
2
3

---

3. While still in the serial monitor, type the following:

12,23,34,45, <enter>

Please note the commas between each numerical entry. You should now see a pattern like this:
12
23
34
45

You will notice that the commas have been replaced by line feeds, and each number should display on a new line.

---

4. While still in the serial monitor, type the following:

1,,2,,3, <enter>

You should see the following pattern:
1

2

3

So hopefully that explains the concept of delimiters and how they can be used to separate a stream of numbers, no matter how long it takes to get to the Arduino. We used an IF-statement to listen for the comma, but we could have used any other delimiter provided we knew the byte code.

We did not identify how to send delimiters FROM the Arduino, but we will get to that I promise. It is not that hard, and uses a similar principle. I am sure you can work it out, if not, stay tuned.

STAGE 3: Arduino Maths: Simple addition

In this stage, we are going to get the Arduino to do simple maths. We will send it two integers (or whole numbers), and the Arduino will do the hard work and send us the answer in no time at all.
This might seem like a simple task, but when you send a number like 27 to the Arduino, it does not receive the number 27. It receives 2 and then 7 in byte form. In other words, the Arduino will see the byte codes 50 and then 55 as per the ASCII table on this page.

One way to convert this byte code back to a 2 and a 7 is to subtract 48 from each byte received, providing the byte is in the range 48 to 57 inclusive (which equates to the numbers 0-9).
We are not done yet. We then need to join these numbers to make 27.

Step1: Subtract 48 from the bytes received, only if the bytes are in the range 48 to 57.
Example: 50 - 48 = 2
55- 48 = 7

Step2: Multiply the previous number by 10, before adding the most recent byte received.
Example: (2 x 10) + 7 = 27

If we have a number like 1928, then we would create this number using the following calculation
1 = 1
(1 x 10) + 9 = 10 + 9 = 19
(19 x 10) + 2 = 190 + 2 = 192

(192 x 10) + 8 = 1920 + 8 = 1928

Step3: Use a "+" sign as a delimiter so that the Arduino can move onto the Second number

Step4: Capture the second number as per Step2. An "=" sign will tell the Arduino that it has reached the end of the second number, and to proceed to step 5.

Step5: Add the 2 numbers together and send back the answer.

The following code will carry out the 5 steps above.
Enter the following sketch into your Arduino IDE and upload it to your Arduino.

Arduino Sketch

```
/* Simple Serial ECHO script : Written by ScottC 05/07/2012 */
/* Stage 3: Arduino Maths: Simple Addition */

/* Global variables needed for programming workflow
   byteRead: holds the value being read from the COM port
   num1: holds the entire first number
   num2: holds the entire second number
   answer: holds the sum of num1 and num2
   mySwitch: enables the switch between num1 and num2  */

byte byteRead;
long num1, num2,answer;
boolean mySwitch=false;

void setup() {
/* Turn the Serial Protocol ON and
   initialise num1 and num2 variables.*/
  Serial.begin(9600);
  num1=0;
  num2=0;
}

void loop() {
   /*  check if data has been sent from the computer: */
  while (Serial.available()) {
    /* read the most recent byte */
    byteRead = Serial.read();

    //listen for numbers between 0-9
    if(byteRead>47 && byteRead<58){
       //number found

       /* If mySwitch is true, then populate the num1 variable
          otherwise populate the num2 variable*/
       if(!mySwitch){
         num1=(num1*10)+(byteRead-48);
       }else{
         num2=(num2*10)+(byteRead-48);
       }
    }

    /*Listen for an equal sign (byte code 61)
      to calculate the answer and send it back to the
      serial monitor screen*/
    if(byteRead==61){
      answer=num1+num2;
      Serial.print(num1);
      Serial.print("+");
      Serial.print(num2);
      Serial.print("=");
      Serial.println(answer);

      /* Reset the variables for the next round */
      num1=0;
      num2=0;
      mySwitch=false;

    /* Listen for the addition sign (byte code 43). This is
       used as a delimiter to help define num1 from num2 */
    }else if (byteRead==43){
      mySwitch=true;
    }
  }
}
```

The above code was formatted using this site

Instructions

1. Once the code has been uploaded to the Arduino, open the Serial Monitor once again and type the following sequence:

1+2= <enter>

You should get the following message sent back to Serial Monitor

1+2=3

Things to Try

1. Enter this sequence:

10 <enter>
\+ <enter>
10 <enter>
= <enter>

Result: 10+10=20

---

2. Enter this sequence:

10 <enter>
20 <enter>
+5= <enter>

Result: 1020+5=1025

---

3. Enter this sequence:

10+20+30= <enter>

Result: 10+2030=2040

I have specifically written this script to add **two** whole numbers together. If you start to introduce more complicated calculations, the results become unpredictable.

---

4. Enter this sequence:

1.2+1.0= <enter>

Result: 12+10=22

Once again, I have only designed this script to handle whole numbers. Therefore, decimal points are ignored.

---

5. Enter this sequence:

-5 + 10= <enter>

Result: 5+10=15

This script ignores the negative sign, and treats the -5 as a positive 5.

I have done this on purpose. I wanted to show you how the Arduino reads numbers from the com port, and how easy it is to exclude vital functionality in your code. I have kept this script simple, however, if you wanted to, you could make the Arduino deal with each of the above situations and more. Multiplication, division and subtraction is handled in the same way.

This is the last thing I want you to try before we go to the next stage:

6. Enter this sequence:

2147483646+1= <enter> Result: 2147483646+1=2147483647
2147483647+1= <enter> Result: 2147483647+1=-2147483648

Note that the maximum size of a "long" number is 2147483647. If you add one to this number, the result is equal to the minimum size of a "long" which is -2147483648.

STAGE 4: Sending doubles to Arduino : The double doubler

Now we get to some tricky business. Sending and receiving Doubles (to and from) the Arduino.

Up until now, I have tried to keep it simple using whole numbers, but there will come a time when you will want to send a fraction of a number through the Serial line.
To test our program, we will want to send a very small number to the Arduino, multiply the number by 2, and return it back.

Our final test is to try a number like : **0.000001**
and then a number like: **123.321**

IMPORTANT NOTE: When the Arduino sends a float or a double through the COM port using Serial.print() or Serial.println(), it will automatically send the number to 2 decimal places.
A number like 1.2345 will appear as 1.23, and a number like 1.9996 will appear as 2.00
To demonstrate this, we will get the Arduino to send these floats/doubles to the Serial Monitor.

Enter the following sketch into your Arduino IDE and upload it to your Arduino.

Arduino Sketch

```
1 /* Stage 4: Simple Transmission of a Double
2    Written by ScottC on 7/7/2012 */
3
4 /* Declare the doubles that will be sent to the Serial Monitor */
5    double myDub1, myDub2;
6
7 /* This part of the program only runs ONCE */
8
9    void setup(){
10
11   /* Turn ON Serial Communication */
12      Serial.begin(9600);
13
14   /* Assign a value 1.2345 and 1.9996 to the Doubles being sent */
15      myDub1=1.2345;
16      myDub2=1.9996;
17
18   /*Send the values to the Serial Monitor */
19      Serial.print("myDub1 (1.2345) : ");
20      Serial.println(myDub1);
21      Serial.print("myDub2 (1.9996) : ");
22      Serial.println(myDub2);
23   }
24
25
26   void loop(){
27  //Loop does nothing
28   }
```

The above code was formatted using this site

When you open the Serial monitor (after you have uploaded the sketch above), you will notice the following output:

```
myDub1 (1.2345) : 1.23
myDub2 (1.9996) : 2.00
```

The blue text represents the string (or array of characters) being sent using lines 19 and 21.
The red text represents the actual double being sent using lines 20 and 22.

You will notice that myDub2 rounds to 2.00. This may or may not be what you want.
If you wish to increase the number of decimal places, then you will need to change lines 20 and 22 to the following:

```
20      Serial.println(myDub1,4);
22      Serial.println(myDub2,4);
```

The number 4 highlighted in red, indicates the number of decimal places you wish to send.
Try it ! And try changing this number to something bigger or smaller.

Ok - now that we understand this little Serial.print(double,decimals) trick, we will now get the Arduino to echo back a Double.

Before we jump in, perhaps we should try and map out our strategy. For this we will choose a simple decimal to make it easier. So in this example, we will choose **0.1**
Once we get this working, we can then do our final test (as mentioned above).

If we send 0.1 to the Arduino, it will read the following byte code

48	0
46	.
49	1

We can use the decimal point as a delimiter.
We will use the following 5 steps to echo the double back to the Serial Monitor:

Step1: Arduino collects all numbers before the decimal point using the same technique as in Stage3.

Step2: When the Arduino receives byte code 46, it will go into decimal mode.

Step3: The Arduino will collect numbers after the decimal point using a similar technique to step1.

Step4: Use maths to create the double, and then multiply it by 2

Step5: Display the doubled Double value in the Serial monitor.

Enter the following sketch into your Arduino IDE and upload it to your Arduino.

Arduino Sketch

```
/* Simple Serial ECHO script : Written by ScottC 06/07/2012 */
/* Stage 4: Double doubler */

/* Global variables needed for programming workflow
   ------------------------------------------------------
   byteRead: holds the value being read from the COM port
   num1: holds the number before the decimal point
   num2: holds the number after the decimal point
   complNum: holds the complete number (before multiplation)
   answer: holds the final value after multiplication
   counter: is used to convert num2 to the number after the decimal
   numOfDec: counts the numbers after the decimal point
   mySwitch: enables the switch between num1 and num2  */

   byte byteRead;
   double num1, num2;
   double complNum,answer,counter;
   int numOfDec;
   boolean mySwitch=false;


   void setup() {
/* Turn the Serial Protocol ON and
   initialise num1 and num2 variables.*/
     Serial.begin(9600);
     num1=0;
     num2=0;
     complNum=0;
     counter=1;
     numOfDec=0;
   }

   void loop() {
/*  check if data has been sent from the computer: */
     while (Serial.available()) {
     /* read the most recent byte */
        byteRead = Serial.read();

       //listen for numbers between 0-9
       if(byteRead>47 && byteRead<58){
          //number found

          /* If mySwitch is true, then populate the num1 variable
          otherwise populate the num2 variable*/
```

```
45          if(!mySwitch){
46            num1=(num1*10)+(byteRead-48);
47          }else{
48            num2=(num2*10)+(byteRead-48);
49
50         /* These counters are important */
51            counter=counter*10;
52            numOfDec++;
53          }
54       }
55
56    /*Listen for an equal sign (byte code 61)
57      to calculate the answer and send it back to the
58      serial monitor screen*/
59      if(byteRead==61){
60   /* Create the double from num1 and num2 */
61      complNum=num1+(num2/(counter));
62
63   /* Multiply the double by 2 */
64      answer=complNum*2;
65
66   /* Send the result to the Serial Monitor */
67      Serial.print(complNum,numOfDec);
68      Serial.print(" x 2 = ");
69      Serial.println(answer,numOfDec);
70
71   /* Reset the variables for the next round */
72      num1=0;
73      num2=0;
74      complNum=0;
75      counter=1;
76      mySwitch=false;
77      numOfDec=0;
78
79  /* Listen for the decimal point (byte code 46). This is
80     used as a delimiter to help define num1 from num2 */
81     }else if (byteRead==46){
82        mySwitch=true;
83     }
84   }
85 }
```

The above code was formatted using this site

Things to Try

1. Type the following into the serial monitor:

1.2= <enter> Result: 1.2 x 2 = 2.4

Make sure that you type the equal sign (=) before you press enter, otherwise the Arduino will not know that you are finished, and will not send anything back.

---

2. Type the following into the serial monitor:

100.001= <enter> Result: 100.001 x 2 = 200.002

You will notice that the Arduino is formatting the decimal to the SAME number of decimals as that entered. This is controlled by the variable: numOfDec.

---

3. Now for our final test: Type the following into the serial monitor:

0.000001= <enter> Result: 0.000001 x 2 = 0.000002

First test: PASSED

---

4. Type the following into the Serial monitor for our last test:

123.321= <enter> Result: 123.321 x 2 = 246.642

Second test: PASSED

BEWARE: While everything looks perfect, let me tell you that it isn't. But hopefully this code will help you get on the right track. If you decide to type in a number like 123123.111222, you will not get the answer you expected.
I have found that this program will work if the amount of numbers before and after the decimal point are less

than about 9. Eg. 1234.1234 will produce the right result.
However, 11111.2222 will NOT, because there are 9 numbers represented.

I think this has something to do with the memory allocated to a double, but I am not sure.
I don't know if people work with these types of numbers, but I am sure there is a workaround, and I am sure someone out there can work it out. I don't personally need this kind of precision, but thought to mention it just in case you do.

STAGE 5: Sending sensor data to the Serial Monitor

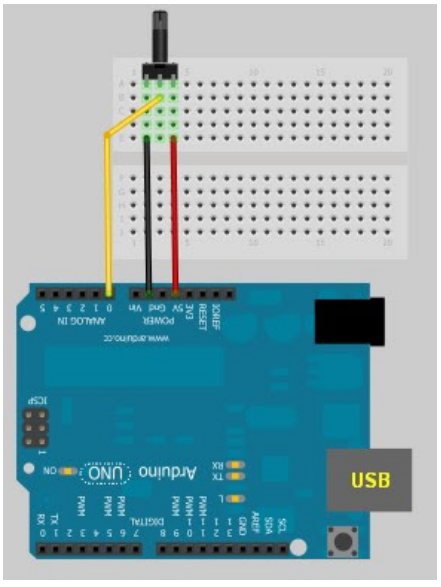
We know the Arduino is very good at copy-Cat games, how about getting the Arduino to send us some data from one of our sensors. We will use the Serial Monitor to view the sensor data.

Disconnect the USB cable, and hook up one of your favourite analog sensors to your Arduino. For simplicity, I am going to hook up a potentiometer as per the Fritzing sketch below.

Parts Required

- Arduino UNO (or equivalent)
- Computer with USB cable
- Breadboard
- Potentiometer
- 3 Wires

Arduino Fritzing Sketch



Once you have attached your sensor to the board, plug your USB cable into the Arduino, and upload the following sketch.

Arduino Sketch

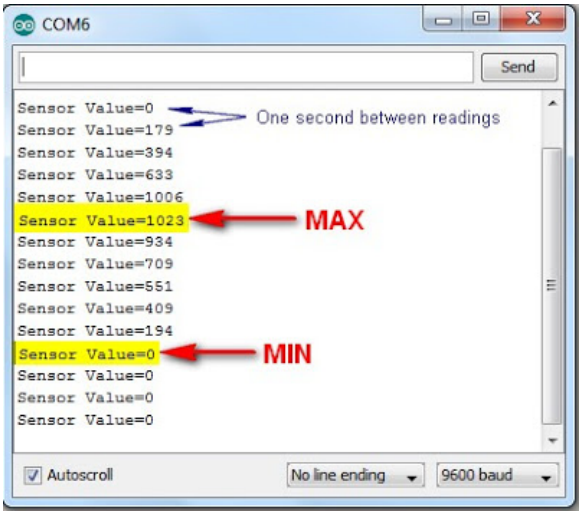
```
1  /* Stage 5: Send Sensor Value to Serial Monitor
2     Written by ScottC on 7/7/2012 */
3
4  int sensorVal = 0;
5
6  void setup() {
7 // Setup Serial communication with computer
8    Serial.begin(9600);
9  }
10
11 void loop() {
12 // Read the value from the sensor:
13    sensorVal = analogRead(A0);
```

```
14
15  // Send the value to the Serial Monitor
16     Serial.print("Sensor Value=");
17     Serial.println(sensorVal);
18
19  // Interval between readings = 1 second
20     delay(1000);
21   }
```

The above code was formatted using this site

Instructions

1. Open the Serial monitor and watch the readings change depending on the input conditions. In my case, by turning the potentiometer from left to right, I get an output similar to the picture below.



As per the Arduino reference site, AnalogRead returns an integer between 0 and 1023. You can see this is true based on the picture above. But what if we do not want a value between 0 and 1023. Let us say we want a value between 0 and 100?

You would have to use the map function. We will do it by changing line 13 to this:

```
13  sensorVal = map(analogRead(A0),0,1023,0,100);
```

The map function is quite a cool function, and good fun to play around with. So here are some things to try.

Things to Try

1. Change line 13 to the following, upload to the Arduino and then open the Serial Monitor to see the effect.

Trial 1:

```
13  sensorVal = map(analogRead(A0),0,1023,100,0);
```

Trial 2:

```
13  sensorVal = map(analogRead(A0),0,1023,0,1000);
```

Trial 3:

```
13  sensorVal = map(analogRead(A0),200,800,0,100);
```

In **Trial 1**: We see that the values have been inverted. Instead of ranging from 0 up to100, they now go from 100 down to 0.

In **Trial 2**: The analog readings are now mapped to a range of 0 up to 1000.

In **Trial 3**: The analog readings that range from 200 to 800 are mapped to a range of 0 to 100. Therefore if the analog readings drop below 200, we will end up with a negative value for sensorVal.
If the analog readings go above 800, we will end up with a value greater than 100. For this particular example, my readings actually range from -33 to 137.

Therefore an Analog reading of 0 = -33
Analog reading of 200 = 0
Analog reading of 800 = 100
Analog reading of 1023 = 137

What if we don't want the output to go beyond our intended limits of 0 to 100?
Then you would have to use the constrain function. This essentially trims the reading range of the sensor, and sets a minimum and maximum value.

Replace line 13 with the following code:

```
13  sensorVal = constrain(map(analogRead(A0),200,800,0,100),0,100);
```

Therefore an Analog reading of 0 = 0
Analog reading of 100 = 0
Analog reading of 200 = 0
Analog reading of 800 = 100
Analog reading of 955 = 100
Analog reading of 1023 = 100
Analog values between 200 and 800 will produce a result between 0 and 100.

If you wish to continue with this tutorial (stage 6 and above), please follow this link:
Serial Communication Stage 6 and above

Visit my ArduinoBasics Google + page.
Follow me on Twitter by looking for ScottC @ArduinoBasics.
I can also be found on Pinterest and Instagram.
Have a look at my videos on my YouTube channel.

However, if you do not have a google profile...
Feel free to share this page with your friends in any way you see fit.

Posted by Scott C at 00:40

Labels: addition, AnalogRead, Arduino, ArduinoBasics, best arduino blog, code, comma, communication, constrain, delimiters, double, echo, floats, map, precision, Serial, Serial Monitor, USB

72 comments:

Anonymous 18 February 2013 at 17:47

Extremely Ossam explanation.The best of my findings.

Reply

Replies

Scott C 20 February 2013 at 19:25

Thanks :)

Reply

Mat (from France) 21 February 2013 at 01:51

Hi Scott !
what a wonderful job you did !
I really like to read this Serial tutorial : the best I found
Thanks for sharing !

I'd love to read more explanation from you about bufferUntil :-)

yours
Mat

Reply

Replies

Scott C 22 February 2013 at 21:11

Thanks Mat,
I will take your suggestion and write about BufferUntil in the next session, but due to my limited time at the moment - will have to wait a little bit before I can get on this. But thank you for the suggestion.

Scott

Reply

Ichimon 23 February 2013 at 19:38

Hello wow this is one very nice tutorial to understand...

But, I have some problem on Stage 1.
My output is very different , when is type 0 it come out

Output:
48
10

OK that is my output..not similar to your:

Output:
48

I am hoping for your answer..Thank You..

Reply

Replies

Scott C 23 February 2013 at 23:25

Hi Ichimon,
Try typing 0, and then press the "send" button without pressing Enter. And see if that helps.
48 is equivalent the number 0,
and 10 is equivalent to a line feed.
You are somehow transmitting a line feed. Not sure if it is because of a specific operating system or keyboard, but interesting nonetheless.

Regards
Scott

Reply

Anonymous 5 May 2013 at 13:09

Great tutorial! Many Thanks.

Edgar

Reply

Replies

Scott C 7 May 2013 at 01:23

No problem Edgar. Glad you like it.
Its good to get feedback every now and then :)

Reply

Anonymous 6 May 2013 at 11:14

I have a question:

When you use Serial.write(1.456), you are sending data as "bytes", right?
Once you receive that data at the other side of the serial (for example simulink), is this possible to obtain 1.456? Or are you receiving 1.45 or just 1?
Thanks!

Reply

Replies

Scott C 6 May 2013 at 17:52

Hi Anonymous,

Please note the difference between Serial.write() and Serial.print().
See stage 4 of this tutorial on how to send a value such as 1.456 from the Arduino to the Serial Monitor. Once you get this part working, then you can work on the other half of your project, which would be the task of getting Simulink to listen for Serial data.

If you wanted to send 1.456 through using Serial.write, then I would write the code like this:

```
Serial.write(49);
Serial.write(46);
Serial.write(52);
Serial.write(53);
Serial.write(54);
Serial.println();
```

You may want to look at the ASCII table as to how I got those numbers.
I'll give you a hint: When you write 49, it sends through a 1

If you choose to send through the value using Serial.print():
I would send it through like this:

```
Serial.println(1.456, 3);
```

If you write it like this: Serial.println(1.456); then it will only send to the default 2 decimal places which is 1.46

Hope this helps
Scott

Scott C 6 May 2013 at 18:18

Sorry, there is one other option:

```
Serial.write("1.456");
```

And an ASCII table can be found here.

Reply

Anonymous 6 May 2013 at 19:48

Thanks for your answer, Scott C!

I understand that I can send different types of data depending on using .write or print. On simulink I

have a block that strictly needs his input data to be bytes, that is the reason I am using Serial.write.

In that Simulink block, I can choose how the data can be shown (double, single, int8, unint8,...). My problem is that just int8 or unint8 works. Once I select "double" or "single", the data obtained is not correct.

I assume that the problem remains on the sender, the arduino, and consequently Serial.write.

Then, once I use Serial.write(1.456),theorically the receiver should or should not be able to obtain 1.456 again?

Thanks for your effort!

Reply

Replies

Scott C 7 May 2013 at 00:59

This is not as easy as one may think, because you are trying to send a float/double. You may want to have a read of the following Arduino Forum discussion which looks similar to what you are trying to achieve.

You will also find that if you try to compile the following program on the Arduino, that you will get an error, because there is no function call representing Serial.write(double);

```
//Arduino Example Program
void setup(){
Serial.begin(9600);
Serial.write(1.456);
}

void loop(){
}
```

My other suggestion would be to create an interface between the Arduino and Simulink, so that you could successfully read the values from the Arduino, and could then convert the data into the correct format for Simulink, but I have no experience with Simulink, so would not know if that suggestion is feasible.

Anonymous 7 May 2013 at 05:56

Thanks Scott C, this is interesting.

Maybe I shoud just multiply my float for 100, send this by serial.write(), (I assume this will send the integer at least) and then in Simulink multiply for 1/100.

Thank you

Scott C 7 May 2013 at 07:59

I would be interested to know what happens with multiplied values above 255. Let me know how you go.

Reply

Anonymous 7 May 2013 at 09:02

Multiplying by 100 does not work.

I ask the simulink block to obtain int8. May values can go arround from +125 to -125. This might be the 255 you are suspecting.

So when this is reading, for example: 120,124,122,123,124,125 then goes to -125,-124,-123.. instead of going to 126,127,128.

That means that there is not possible to send the number 1000 via Serial.write()?
I am not sure what this is happening.

Thank you

Reply

Replies

Scott C 8 May 2013 at 00:36

The problem is that Serial.write() sends a single byte. If you go beyond the byte limits, then it will do as you described in your comment above.
A number such as 1.456, is a float which can be stored within 4 bytes. A number such as 1456 can be stored as an integer which is 2 bytes.

I have written an example below which will hopefully help you get onto the right track. Upload this sketch onto the Arduino and then open the Serial monitor to see the outcome. This program makes use of "union" to provide the functionality you may need to get this to work. However, it depends on whether or not you can re-assemble the 4 bytes on Simulink side, which is something that you will have to test.

Most of this code was sourced from the Arduino Forum.

Here is the code:

```
/* ============================================ */
union{byte uBytes[4];float uFloat;} uData;
union{byte xBytes[4];float xFloat;} xData;

void setup(){
Serial.begin(9600);
float myFloat = 1.456;

uData.uFloat=myFloat;

Serial.write(uData.uBytes[0]);
Serial.write(uData.uBytes[1]);
Serial.write(uData.uBytes[2]);
Serial.write(uData.uBytes[3]);
Serial.println();
xData.xBytes[0]=uData.uBytes[0];
xData.xBytes[1]=uData.uBytes[1];
xData.xBytes[2]=uData.uBytes[2];
xData.xBytes[3]=uData.uBytes[3];

float yourFloat=xData.xFloat;
Serial.println(yourFloat, 3);

}

void loop(){
}

/* ============================================ */
```

Scott C 8 May 2013 at 00:38

The code above converts a Float to an array of 4 bytes, and then copies it to another array of 4 bytes which is converted back to a Float.

Reply

Anonymous 9 May 2013 at 09:17

Thanks for your help Scott,

The block I am using does not understand about strings I guess.

In simulink you are able to select the data that is receiving, that is: int8,int16,int32, single and double.

You can select the "parity" too, which I selected "none" (from: none, even, odd, mark or space), is that correct?

And, I also can select "byte order" between bigEndian or LittleEndian (I have BigEndian but I do not know the difference).

I am thinking about sending a byte of the integer of the float and a byte of the 2 decimals of the float. I will read this in simulink as two different numbers but once read I can play with them. I will try this.

Keep you update, thanks for all your help!

Reply

Anonymous 1 November 2013 at 00:50

Really good beginners tutorial

Reply

Alejandro Varela 30 December 2013 at 00:48

when you say...
/* read the most recent byte */
byteRead = Serial.read();

Serial.read() does not read the "most recent" byte, the bytes are pushed onto a queue (FIFO).
Example:
(time n) a device push the byte 0x01 in the arduino
(time n+1) a device push the byte 0x02 in the arduino
(time n+2) a device push the byte 0x03 in the arduino <-- most recent
(time n+3) arduino call Serial.read()
in this case Serial.read will return 0x01

Reply

Replies

Scott C 30 December 2013 at 08:20

Alejandro, you are correct. I should have written /* read next available byte */
Well picked up.

Reply

Anonymous 12 January 2014 at 22:26

Good tutorial, I tried it with lcd connected to my microcontroller(a clone of arduino), can you show me how to input characters via universal 4x4 keypad connected to the microcontroller?

Reply

Replies

Scott C 12 January 2014 at 22:30

have a look at this tutorial here

Reply

patrick villacorta 17 January 2014 at 16:46

how to clear all text on the serial screen??

Reply

Replies

Scott C 6 February 2014 at 18:11

I don't think there is a way to do this through code. But you could do it by closing and opening the serial monitor manually - not sure if this is what you mean?

Reply

Joseph Adjei 22 February 2014 at 01:00

I am trying to do the same with the 2 entry numbers but except I want 5 entry numbers but I cant seem to get it right.. this is the code that I have for right now. int enternumber;
int num1, num2, num3, num4, num5;
long answer=0;
boolean myswitch = false;

void setup(){
Serial.begin(9600);
num1 = 0;
num2 = 0;
num3 = 0;

```
num4 = 0;
num5 = 0;

}

void loop () {
while (Serial.available()){
enternumber = Serial.read();
Serial.write(enternumber);
if (enternumber > '0' && enternumber < '9'){
if (!myswitch){
num1=(num1*10)+(enternumber-48);
num2=(num2*10)+(enternumber-48);
num3=(num3*10)+(enternumber-48);
num4=(num4*10)+(enternumber-48);
}else{
num5=(num5*10)+(enternumber-48);
}
}

if (enternumber == 61){
answer=num1+num2+num3+num4+num5;
Serial.print(num1);
Serial.print("+");
Serial.print(num2);
Serial.print("+");
Serial.print(num3);
Serial.print("+");
Serial.print(num4);
Serial.print("+");
Serial.print(num5);
Serial.print("=");
Serial.println(answer);

num1=0;
num2=0;
num3=0;
num4=0;
num5=0;
myswitch=false;

}else if (enternumber ==43){
myswitch=true;
}
}
}
```

Reply

Joseph Adjei 22 February 2014 at 01:01

with 1+1+1+1+1=5...I keep getting 1+1+1+1+1=1+1+1+1+1111=1115. please help!

Reply

Replies

Scott C 22 February 2014 at 08:02

Hi Joseph,
You have confused me a little, but that's ok.
If you only want to have 2 inputs - with the first input replicated 4 times, but with the last input independent from the first 4 numbers, then you have coded this properly.
I would eliminate the Serial.write(enternumber); line. And just type 1+1= into the serial monitor.
This will give you the answer you want. If you want the output of 3+3+3+3+5=17 as the ouput, then you would enter this into the Serial monitor: 3+5=
However, if you actually want 5 independent inputs and get the arduino to calculate, then you have done it all wrong. Pay attention to "myswitch" and what myswitch is actually there for.
What is "myswitch" actually doing?, and why do you need it in my original code?
There lies your problem.
Regards
ScottC

Reply

Anonymous 4 March 2014 at 22:18

Hi Scott
i have a text file containing data as a string on separate rows like :
string1
string2
string3
ect..

so when i'm trying to send the data line by line with processing to arduino .. the first string recieved with arduino is the line 3 not line 1 and i cannot recieved the rest of string because the arduino always concatenate the strings comming from processing the '\n' dont work with me
help plz

Reply

Replies

Scott C 6 March 2014 at 21:01

Have you seen this post
http://arduinobasics.blogspot.com.au/2011/06/reading-text-or-csv-file-using.html ?

Reply

Meaghan Bond 16 April 2014 at 06:19

Thanks for the help! Very simple.

Do you know if you can trigger an interrupt on the Arduino when there's data available on the Serial monitor?

Reply

Replies

Scott C 16 April 2014 at 08:07

Hi Meaghan,
That is a good question. I have never tried to do this myself.
However, from what I can see at the Arduino website (here) - there seems to be a possibility of Serial data loss while running the Interrupt function.

Meaghan Bond 17 April 2014 at 04:40

If you keep your interrupt function short and aren't expecting Serial data at a high rate, it seems worthwhile, but I can't figure out how to attach an interrupt to something that's not a pin. If I'm able to find information elsewhere, I'll report back.

Scott C 17 April 2014 at 06:51

Hi Meaghan,
What are you trying to do?

Reply

Meaghan Bond 17 April 2014 at 06:56

I want to have a computer send commands to my Arduino to start certain sequences - i.e., send a '1' via serial to start sequence A, send a '2' via serial to start sequence B, etc. For this device, the serial commands are replacing button presses, which I previously handled using interrupts.
There should be at least a 1 second delay between each serial command, so using a short interrupt function (set buttonPressed = true, for example) shouldn't be a problem - i.e., I shouldn't be missing the next serial byte. I just can't figure out how to get a byte available on serial to trigger the interrupt routine.

Reply

Replies

Scott C 17 April 2014 at 07:32

I would have thought you could have done this without interrupts. Have a look at my Bluetooth tutorials which may be similar to what you are trying to achieve.

Scott C 17 April 2014 at 15:04

Actually, just continue on with this tutorial. And I cover something similar to what you are trying to do in stage 9 . I forgot that I had this in there. It was a while ago that I wrote this tutorial. Let me know if this is suitable.

Reply

Hashim Al Sakkaf 28 April 2014 at 06:41

hi and thanks for this nice tutorial, i have a question which i did not find an answer to it and i need your help on it, how can i send int values as bytes using serial.write, then how can i read those values at the receiver side (other arduino)?

Reply

Replies

Scott C 29 April 2014 at 19:25

Continue the tutorial to stage 11.

Reply

Anonymous 19 June 2014 at 18:49

nice tutorial...

Reply

Mohamed Fahmy 18 October 2014 at 00:33

Thank you for an answer to the question: What is the purpose of using double addition sign ++ in some program lines such as the following one?

{
for(int lcd_cursor=0; lcd_cursor<32; lcd_cursor++)
{

Reply

Replies

Anonymous 18 October 2014 at 02:01

lcd_cursor++
Increments that variable by one

Reply

Scott C 23 October 2014 at 23:16

Attention tsr:
You posted a question here. I have removed it.
Questions related to your specific project are best served within the Arduino forum.

Reply

Anonymous 25 November 2014 at 03:18

Thank you so much this wunderful tutorial

Reply

Replies

Scott C 25 November 2014 at 08:10

I am glad you like it

Reply

jackeychua 30 November 2014 at 01:47

hi can i ask for some question about arduino coding?
i want ask for the get value from the BTLE serial
how can i do a program that
1) ask user choose 1 or 2 for switch case
2) if 1 go for open lock function
3) when go to the function ask user to key in password

problem when user key in 1 it go to the function but in the function it wont let me key in other value from the btle serial

Reply

Replies

Scott C 1 December 2014 at 07:23

Questions relating to your own projects are best served on a forum.
May I suggest that you post your project and project related questions on the Arduino Forum ?
Thanks.

Reply

Anonymous 7 March 2015 at 18:04

Hi Scott C,

I was made the stage 1 (echo echo), but when I type 'h' the result is 'n' or type 'hello' the result is 'nßűűű'. What am I doing wrong?

Reply

Replies

Anonymous 7 March 2015 at 21:09

I found it. The baud rate is different in monitor and the code.

Reply

Hammad 22 March 2015 at 15:32

hi every one I need some help about matrix addressing and logging sensor value to serial.
for example i have 10*10 matrix when i send row and column address from pc arduino turns on respective row and column until next address is sent from PC.
and return value of ADC value.

Reply

Replies

Scott C 22 March 2015 at 22:25

Hammad - please post your project related query to the ArduinoBasics forum.
Alternatively, the Arduino forum is likely to be even better.

Reply

michael 21 May 2015 at 03:05

Hi every one,
great explanation. But why don't you mention the loop back test? It is a lot easier then writing a programm. You should try it. http://forum.arduino.cc/index.php?topic=73748.0

Reply

Replies

Scott C 26 May 2015 at 17:36

Not sure if it is easier, and I find it hard to believe that someone will get an Arduino without programming it in some way or another. But thank you for mentioning the loop back test, I had not come across this before, and worthy of a mention.

Reply

Anonymous 21 May 2015 at 12:03

Thanks for telling us about the map function. Cool.

Reply

Replies

Scott C 26 May 2015 at 17:38

No worries -I would also recommend checking out the "constrain" function.

Reply

Anonymous 13 June 2015 at 15:05

Hello,

i have a question. I have a program which encodes a Number into serial commands and recieve serial commands to handle it as a input.
Now i want to emulate the Hardware and sending a command to the software. But i have no idea how it is coded (i am really a beginner). If i use your Echo programm, the software work well but if i send just the bits to the software i get errors.
If i type in these to send:

30 00 01 02 00 07 00

I get this on the serial communication:

02 30 0A 20 0A 21 0A 22 0A 20 0A 27 0A 20 AE 03

So what do i have to send from Arduino to get these communication back?
If i use Serial.write("02300A200A210A220A200A270A20AE03"); it dont work.

Hope anybody can help me

Reply

Replies

Scott C 15 June 2015 at 12:21

Please post your code to the ArduinoBasics forum,
however, someone there is already asking a similar question.
So maybe read that first, and if that does not help you, then post your own query.
It is much easier to deal with these types of questions on the forum.

http://arduinobasics-forum.1116184.n5.nabble.com/Help-me-with-my-project-f2.html

Reply

salim said 17 July 2015 at 17:23

Very nice introduction to arduino serial communication.
I like your post.

Reply

Replies

Scott C 18 July 2015 at 10:34

Thanks Salim - I appreciate your feedback,

Reply

charan naidu 20 July 2015 at 21:55

Is there any other way to read a serial data except serial.available function?

Reply

Replies

Scott C 21 July 2015 at 09:20

The Serial.available function is the most common. What are you trying to do ?

Reply

Gabriel Alejandro 10 August 2015 at 11:49

Perfectamente explicado. Un verdadero profesor! Gracias!

Reply

Replies

Scott C 10 August 2015 at 12:41

De Nada ! Thanks for the feedback Gabriel.

Reply

Biruntha Gnaneswaran 26 November 2015 at 19:07

Hi,
I want to transfer mp3/wav files via bluetooth from raspberry pi to arduino. how can i receive mp3/wav files in arduino?

Reply

Anonymous 6 December 2015 at 22:42

Best Tutorial.. But what is the my switch? I cant see it going True..

Reply

Replies

Scott C 7 December 2015 at 09:02

It goes true at the end of the sketch - (depending on the sketch you are looking at).
I use it above to let the Arduino know that I have encountered a specific value such as a decimal point or an addition symbol...

Reply

rahul lakkisetty 29 January 2016 at 02:08

Hiii nice tutorial,,i have a small doubt.
i am making a project to control prosthetic arm by emg sensor,i think a way to read values from emg sensor by analogread and to print these values in serial monitor by serial.printin the i wanna write a programm depending on the values generated on the serial monnitor to turn the servo motor is it possible.Thankyou in advance

Reply

Replies

Scott C 29 January 2016 at 09:06

Hi Rahul,
You can print values to the serial monitor and use the same values to control the position of the servo. You will probably need to use the "map" and "constrain" commands. Have a look at line #68 within this tutorial's arduino code. This tutorial is somewhat similar to what you are trying to achieve. Yes - I use a different sensor, but the concept is similar. I use temperature values and convert them to servo position values

rahul lakkisetty 29 January 2016 at 20:07

Thanq very much bro

Reply

Sami 20 April 2016 at 12:14

First of all, thanks a lot to the writer for this very useful post.

In my project, I'm trying to send some float type data with upto 2 digits after the decimal to a MYSQL server. I'm using Arduino Uno & Arduino Ethernet Shield, and I need to send the data with upto 4 digits after the decimal.

I was really frustrated, but my the mercy of God, I came to read this post & found out that it's Arduino's nature.

Most probably I shouldn't ask it here. But still, could the writer help me with a code solving my problem?

Thanks in advance.

Reply

Replies

Scott C 20 April 2016 at 16:58

I think I show you how to do it in the tutorial above ??

Reply

Enter your comment...

Comment as: Sebastiano Me ▾ Sign out

Publish Preview Notify me

Feel free to leave a comment about this tutorial below.
Any questions about your particular project should be asked in the ArduinoBasics forum.

Comments are moderated due to large amount of spam.

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

This Week's Most Popular Posts

433 MHz RF module with Arduino Tutorial 1

HC-SR04 Ultrasonic Sensor

Simple Arduino Serial Communication

Relay Module

433 MHz RF module with Arduino Tutorial 2

Most Recent Posts

Arduino Stroboscope Animation

Circuit Diagram

Get Arduino Data over the internet using jQuery and AJAX

Two Million Views

Generosity Campaign Update - Day 3

Recent Posts Widget by Helplogger