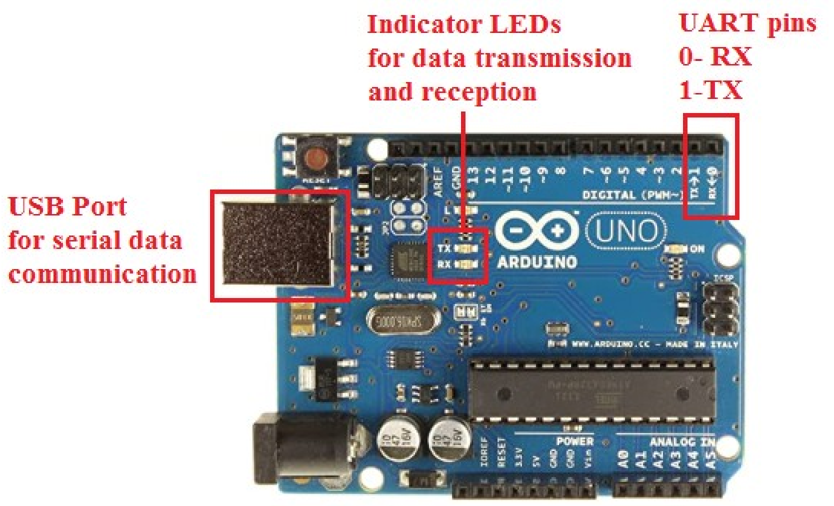
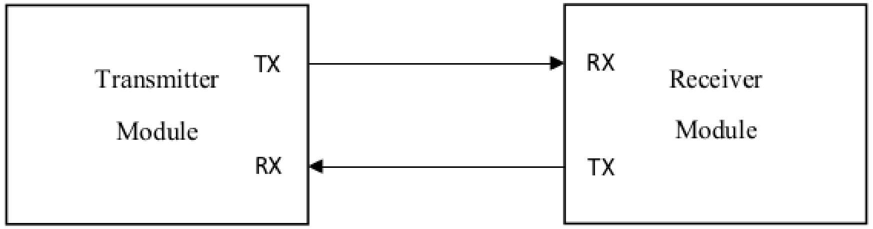
Serial Data Communication in Arduino



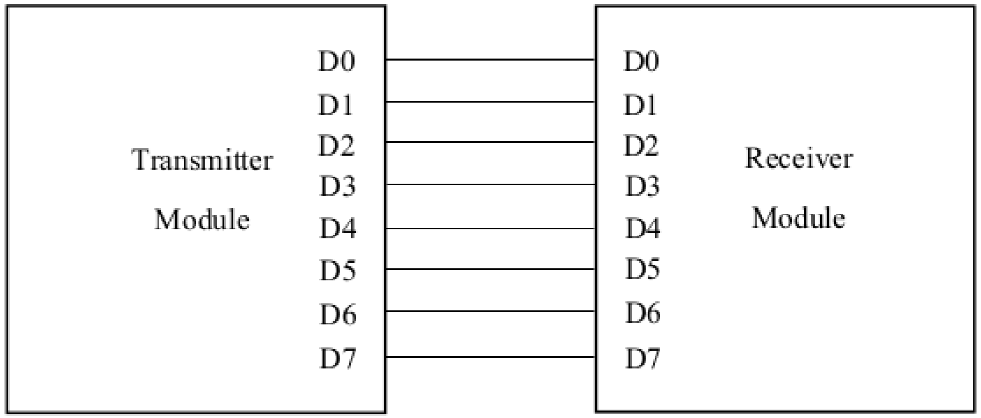
UART (Universal Asynchronous Receiver Transmitter) Module

Serial Vs Parallel

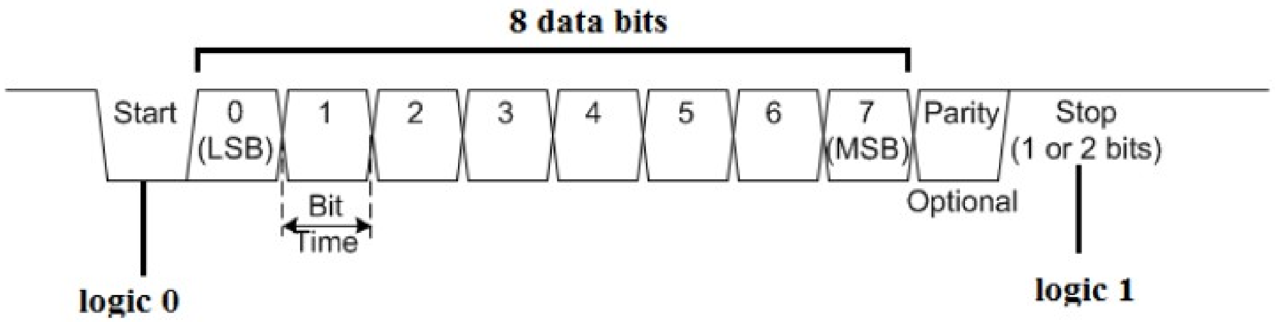
Serial Data Communication



Parallel Data Communication



Data Format



Start bit – Marks the start of data packet (Logic 0)

Stop Bit – Marks the end of data packet (Logic 1)

Parity Bit – Signifies number of 1's in the data field
(1 – odd parity, 0- even parity)

Data Field – 8 bits of data (but can range from 5 to 8 bits)

Important Arduino Functions-Serial Data Communication

- Serial.begin( )
- Serial.print ( )
- Serial.println ( )
- Serial.write ( )
- Serial.available ( )
- Serial.read ( )

Serial.begin ( )

- This command is used to initialize serial communication at a particular baud rate value, specified in terms of bits per second (bps).

  Serial.begin (9600);

  This initializes serial communication at 9600 bps baud rate.
- The other baud rate values that are widely used include: 300 bps, 600 bps, 1200 bps, 2400 bps, 4800 bps, 19200 bps, 38400 bps, 57600 bps, 74880 bps and 115200 bps. The higher the baud rate value, the higher isthe speed of data transmission.
- The two devices which are communicating data serially must be configured at the same baud rate values.
- The default data format includes 8 data bits, No parity bit and 1 stop bit.

Serial.print ( )

- This function is used to print the data on serial port. The data printed on serial port is in ASCII format which is human readable.
- Syntax:

  Serial.print (val)

  Serial.print (val, format)

- This function returns the number of bytes written on the serial port although we will not read the number of bytes that are written but in case you want to then it is possible.

Examples

- Serial.print(“Welcome”); ---This will print a string Welcome on serial port
- Serial.print (75); ---This will print 75 on serial port
- Serial.print (3.14); ---This will print 3.14 on serial port
- Serial.print (3.7521, 2); ---This will print 3.75 on serial port. Here, 2 signifies the number of digits after decimal point that needs to be displayed
- Serial.print (‘A’); --- This will print A on serial port
- Serial.print (85, HEX); --- This will print 55 on serial port
- Serial.print (85, OCT); --- This will print 125 on serial port
- Serial.print (85, BIN); --- This will print 1010101 on serial port

Serial.println ( )

- This is same as Serial.print ( ) function except the fact that using this function the data is printed on serial port and then it is followed by a carriage return and a new line character. Carriage return character is represented by “\r” and new line character is represented by “\n”. the ASCII code for “\r” is 13 and that for “\n” is 10.
- Syntax:

  Serial.println (val);

  Serial.println (val, format)

  format: OCT for octal, DEC for decimal, HEX for hexadecimal, BIN for binary data format

Serial.write ( )

This function writes binary data on the serial port.

Syntax:

Serial.write (val);

This is used for sending the data value (val) on to serial port in binary form

Serial.write (str);

This function can be used for printing string on serial port

- The function also returns the number of bytes that are written on to the serial port and reading it is again an optional thing and is very rarely done

Serial.available ( )

- This function is used to check whether serial data byte is received by Arduino or not and it returns the number of bytes that are available to be read by Arduino.
- while(!Serial.available( ));

  The controller will wait on this statement until any serial data byte is received by Arduino.

```
if (Serial.available( ) > 0)
{
    // statements to execute on reception of serial data
}
```

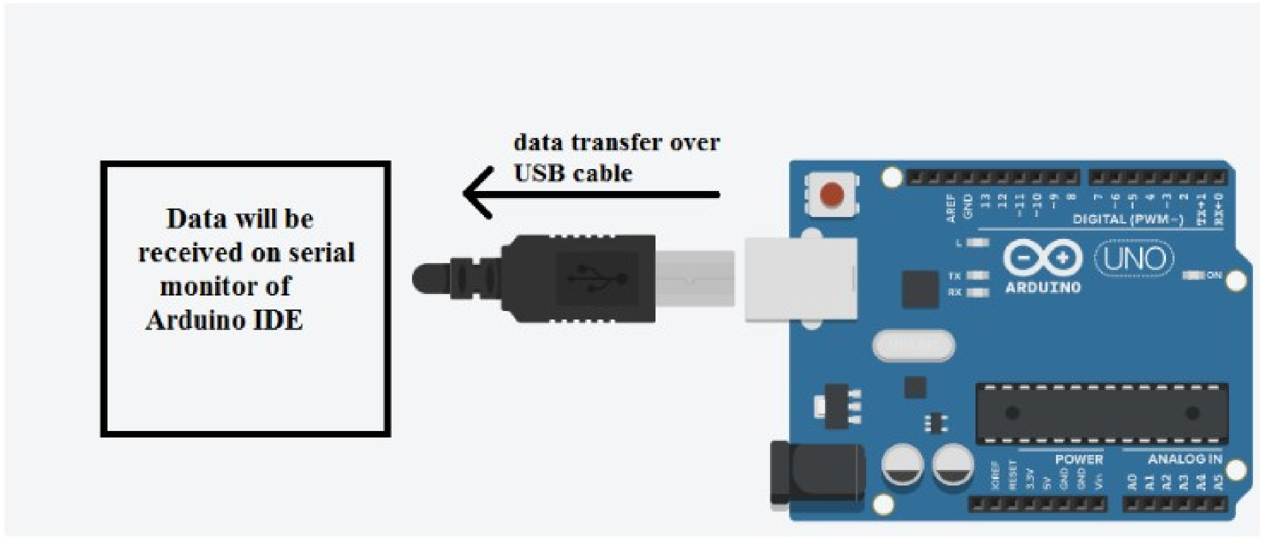
Serial.read ( )

- This function is used to read the serial data byte and it returns the first incoming data byte that is received serially. In the code, this function can be used as follows:

  temp = Serial.read ( );

  Here, temp is a variable that will be defined in the code but let us assume that it is of type char. Then, Serial.read ( ) function will read the first incoming data byte that is received serially and store it in temp variable.

Sending Data from Arduino to Computer



Data will be received on serial monitor of Arduino IDE
data transfer over USB cable
DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN

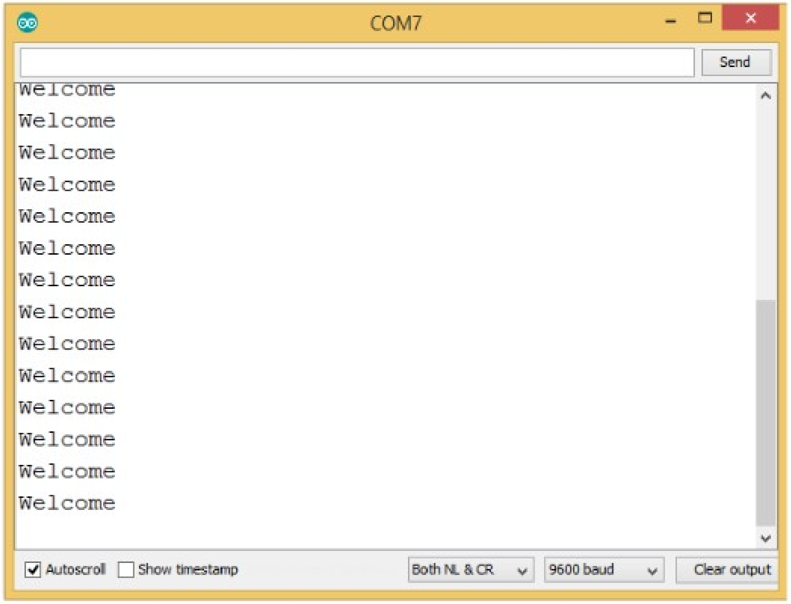
Arduino Program

```
void setup()
{

// put your setup code here, to run once:
Serial.begin(9600);
// initialize serial communication at 9600 bps baud rate
}

void loop()
{
// put your main code here, to run repeatedly:
Serial.println("Welcome"); // print welcome string on serial port
delay(1000); // 1 sec delay
}
```

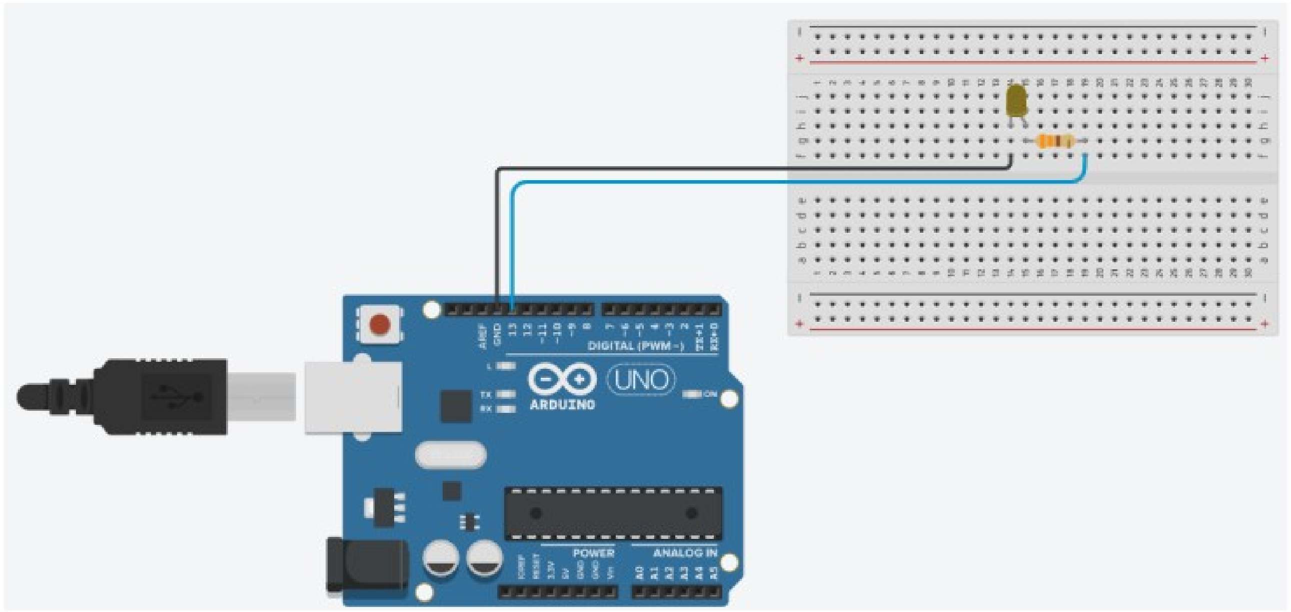
Serial Monitor Output



```
Welcome
Welcome
Welcome
Welcome
Welcome
Welcome
Welcome
Welcome
Welcome
Welcome
Welcome
Welcome
Welcome
Welcome
```

[✓] Autoscroll [ ] Show timestamp Both NL & CR 9600 baud Clear output

Serial Data Reception Example



Coding Logic

To wait for serial data to be received on Arduino board, the following condition can be used:

```
while(!Serial.available( ));
```

To check whether the serial data is received or not, the following condition can be used:

```
if (Serial.available ( ) > 0)
{
// execute the following statements if serial data is received
}
```

To read the serially received data byte, the following statement is used:

```
temp = Serial.read( );
```

Here, temp is a variable that needs to be declared in the code and it will store the serially received data byte.

To compare the serially received data byte with predefined characters ‘A’ and ‘B’, we can use the following logic:

```
if (temp == ’A’)
{
// write the statement to turn ON the LED connected to pin 13
}
if (temp == ‘B’)
{
// write the statement to turn OFF the LED connected to pin 13
}
```