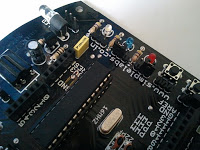
[**Simple Labs' Induino User Guide**](http://induino.blogspot.in/)

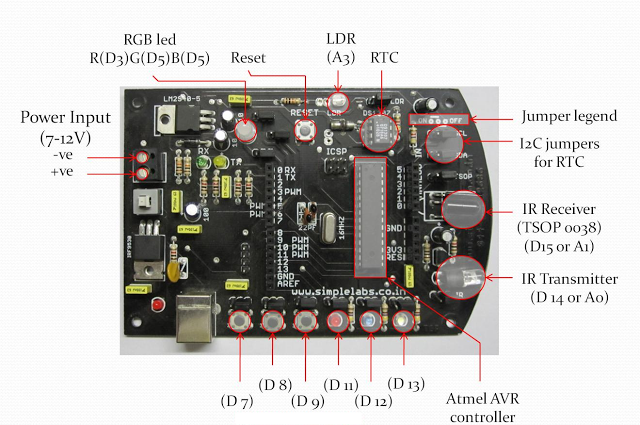
**InduinoX - An Overview**

**InduinoX**

Induino is an electronics prototyping board, developed by Simple Labs, based on the open-source Arduino platform. Induino-X, the latest version of the Induino boards, contains a few sensors, components and devices on-board for easy prototyping. The hardware is programmed using a Wiring-based language (syntax + libraries), similar to C++ with some simplifications and modifications, and a Processing-based IDE. The term Induino is a simple combination of words, India + Arduino = Induino.. ;)

[](http://lh5.googleusercontent.com/-OwsKMC0rGPA/TssFPa4g3pI/AAAAAAAAAaE/ooKZ1mMtTlg/s1600/Induino.jpg)

**Pin Mappings**

[](http://3.bp.blogspot.com/-Uopowmls9Y8/TuY37Yw52RI/AAAAAAAAAb0/olnmtpHRzUQ/s1600/Pin_Mappings4.png)

**Platform**

**Hardware**

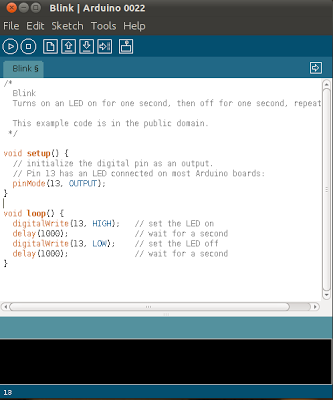
The Induino board consists of an 8-bit Atmel AVR micro-controller with complementary components to facilitate programming and incorporation into other circuits. An important aspect is the standard way that connectors are exposed, allowing the CPU board to be connected to any external devices and a variety of interchangeable add-on modules (known as shields). Also, the on-board components on Induino-X can be connected/disconnected to/from the i/o pins by using jumpers.

The Induino can be connected to a computer and programmed using a standard USB. A FTDI FT232 is used for USB to serial conversion. In-spite of an internal oscillator in the micro-controller, the boot-loader is configured such that it works with a 16MHz external clock oscillator.

**Shields**

Shields are the add-on circuits which are mounted on the Induino board, which gives extended functionality from the available functionality of the board.

Induino Motor Shield  
**Software**

[](http://lh5.googleusercontent.com/-NKcVy4b5PKQ/TssE6l9_1jI/AAAAAAAAAYY/IsN-7sTz4aM/s1600/Arduino022.png)

**Arduino IDE**

The Arduino IDE is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring project. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit makefiles or run programs on the command line. The ArduinoIDE comes with a C/C++ library called "Wiring" (from the project of the same name), which makes many common input/output operations much easier. Arduino programs are written in C/C++, although users only need define two functions to make a runnable program:

* setup() – a function run once at the start of a program that can initialize settings
* loop() – a function called repeatedly until the board powers off

The IDE is available forWindows,MAC & Linux.   
  
**The Induino-X board description**

**Microcontrollers**

[](http://lh5.googleusercontent.com/-1lxskJF9F2M/TssE6mW341I/AAAAAAAAAYU/VjrC_DInoA8/s1600/Atmega168.jpg)

ATmega 168

The Induino-X is compatible with the following micro-controllers...

|  |  |  |  |
| --- | --- | --- | --- |
| Micro-controllers with their memories: | | | |
|  | **FLASH** | **SRAM** | **EEPROM** |
| **ATmega 328** | 32 KB | 2 KB | 1 KB |
| **ATmega 168** | 16 KB | 1 KB | 512 B |
| **ATmega 88** | 8 KB | 512 B | 256 B |

2 KB of the flash memory is consumed by the bootloader. All the above controllers have 8-bit CPUs and 14 digital input/output pins, out of which 6 can be used as analog inputs and 6 as PWM outputs.

**Power Supply**

The Induino-X operates at 5V. It can be powered via the USB or with an external power supply. A 5V linear voltage regulator is employed on-board to support non-USB power supplies. The power source is selected automatically between the USB(IRF 9530) or the external supply(voltage regulator). External supply can be given via the screw terminals (7-12V DC recommended) In the screw terminals the one towards the edge of the board is negative and the other one is positive.

**Hardware on-board**

The following is a list of hardware on-board with their respective pin numbers on the Induino-X. Each lead of the Interface-able Hardware, which is connected to the micro-controller, is provided with a jumper to facilitate the isolation of micro-controller and the hardware when not in use.

1. RGB Tri-colour LED

Red    : Pin 5 (PWM)

Green : Pin 6 (PWM)

Blue   : Pin 3 (PWM)

2. Tactile switches

Switch 1 : Pin 7

Switch 2 : Pin 8

Switch 3 : Pin 9

3. LEDs (Colours of the LEDs may vary depending upon the version you have)

LED 1 (red)    : Pin 11 (PWM)

LED 2 (blue)   : Pin 12

LED 3 (white) : Pin 13

4. IR LED : Pin 14 (Analog 0)

5. TSOP   : Pin 15 (Analog 1)

6. LDR     : Pin 17 (Analog 3)

7. DS1307 (RTC)

SDA : Pin 18 (Analog 4)

SCL : Pin 19 (Analog 5)

**LEDs**

[](http://lh5.googleusercontent.com/-ZHM6gLoSDfE/TssFEC-OSVI/AAAAAAAAAZE/tWtMWfQIukQ/s1600/LED.jpg)

LED

There are three LEDs on the InduinoX (red, blue and white) connected to pins 11, 12 and 13 respectively. Every LED is connected to the micro-controller with a 470 ohm resistor in series, to limit the amount of current flowing through it.  All the LEDs have their jumpers right above them.

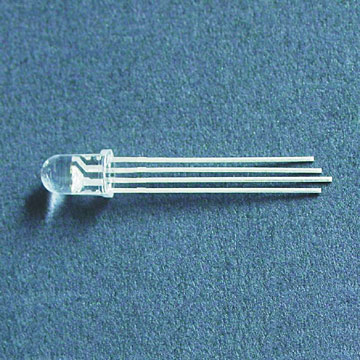
**Tactile Switches**

[](http://lh5.googleusercontent.com/-pH99ADEpft4/TssFF2_67tI/AAAAAAAAAZM/cxvtqawp1HM/s1600/Push_button.jpg)

Tactile Switch

There are three switches on the InduinoX, connected to pins 7, 8 and 9. The tactile switches are connected in active-low mode[They give a HIGH logic when not being pressed and a LOW logic when pressed]. The Internal [Pull-up resistors](http://en.wikipedia.org/wiki/Pull-up_resistor) of the micro-controller should be enabled to use the switches. In the  To enable the internal pull-up resistors, first set a pin input mode and write a digital HIGH to that particular pin.

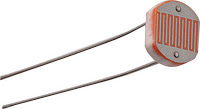
**RGB LED**

[](http://lh5.googleusercontent.com/-TArMtUTeMhA/TssFKjBEgII/AAAAAAAAAZc/AG6BChECBVg/s1600/RGB_LED.jpg)

RGB LED

The RGB LED on the Induino-X is of common-cathode type. It has one common cathode and 3 anodes - one each for blue , red and green connected to pins 3, 5 and 6 respectively on the Induino-X. the Pins 3,5 & 6 are PWM pins and can be controlled from the program by giving 8-bit values. The RGB led is equivalent to one pixel of an LED TV.. ;)

**LDR**

[](http://lh5.googleusercontent.com/-FRlcooywOl4/TssE_0E-IfI/AAAAAAAAAY8/QZruLP1uOBY/s1600/LDR.jpg)

LDR

The LDR is a light dependent resistor. The resistance of the LDR is inversely proportional to the intensity of light incident on it. The LDR on the InduinoX board is to help you learn to work with Analog Inputs from Sensors. the LDR is connected to the lower half of a potential divider configuration with a 10K ohm resistor, the output of the potential divider is connected on analog pin 3. The jumper for the LDR is on the right side of the LDR.

**TSOP**

[](http://lh5.googleusercontent.com/-2qr-q6jMk9I/TssFNcb2YBI/AAAAAAAAAZ4/WXGGt1hizRY/s1600/TSOP.jpg)

TSOP

The TSOP SM0038 is an IR receiver on the InduinoX. The TSOP will help you to interface your TV remote with the InduinoX and in the Process learn the basics of Wireless Communication. The TSOP is connected to pin digital 15(Analog 1). The Jumper for the TSOP is right above it.

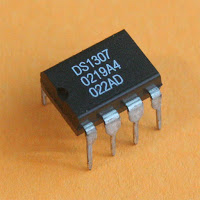
**IR LED**

[](http://lh5.googleusercontent.com/-fj2U3kt1s08/TssE9A3MU7I/AAAAAAAAAY0/bItHOkTc2SA/s1600/IRLED.jpg)

IR LED

The InduinoX has an IR LED connected to pin 14 (Analog 0). Using this, you can simulate remote control signals for various devices. [You can build universal remotes using the InduinoX!] You can also use this along with the TSOP and build a simple low-cost wireless communication system. An NPN transistor(IN2222A) switches the IR LED through a 10 ohm resistor. The jumper of the IR LED is right below it.

**DS1307 (RTC)**

[](http://lh5.googleusercontent.com/-Qp1fcihlOhE/TssE7IXyIlI/AAAAAAAAAYc/4ywYyCk5JYw/s1600/Ds1307.jpg)

DS1307

The DS1307 is a Real Time Clock IC [An IC dedicated to the task of keeping track of Time]. This IC helps learn and work with I2C Communication using the Arduino. The IC is connected to the I2C pins of the Arduino (SDA - Analog 4 & SCL - Analog 5). You can program this IC using eith Wire.H or using the DS1307 Library.

**Getting started**

**We have a separate post on this. Check it here =>**[**InduinoX - Getting Started Guide**](http://induino.blogspot.com/2011/12/induinox-driver-installation-guide.html)

**Overview of the Blink Program...**

We hope you are reading this after having gone through the [InduinoX - Getting Started Guide](http://induino.blogspot.com/2011/12/induinox-driver-installation-guide.html) In the program we use pin 13 to blink LED. As blinking an LED is an output operation pin 13 is set as an output pin. The LED is then turned ON and OFF at an interval of 1000 milliseconds (or 1 second)

//A program to Blink the LED connected to the pin number 13

void setup()

{

pinMode(13,OUTPUT);

//Now we are initializing the pin13 as output pin.

}

// a function which executes again and again

void loop()

{

digitalWrite(13,HIGH);

// now we are turning the LED ON

delay(1000);

//delay in Milli seconds 1000 will produce a delay of one second

digitalWrite(13,LOW);

//Turning the LED OFF

delay(1000);

//Delay again

}

Note: Arduino is case sensitive so the upper case is not equal to the lower case. And the "//" provided here are called the comments that means the compiler ignores the text after it encounters "//".

[**Click Here to Download InduinoX Sample Codes & Required Libraries[Right Click & use Save As]**](http://downloads.simplelabs.co.in/induinox_samples.zip)

**Simple Labs' InduinoX and Arduino Uno Comparison**

**What is InduinoX?**

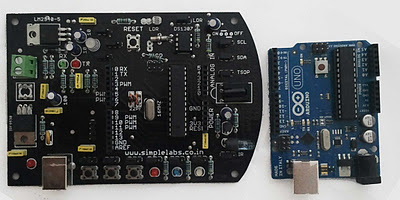
InduinoX is an indian made Arduino clone board. InduinoX is a clone of the popluar Arduino Duemilanove, the predecessor of the current Arduino UNO. It is functionally and mechanically very similar to the Arduino Duemilanove and will work with all types of Shields made for the Arduino Uno / Duemilanove.

About 3 years back, we designed our first Arduino Clone the Induino. We named it Induino to add to our pride of having made it in India. The Induino was an affordable arduino clone that was priced at 800/- [The cheapest to be sold on some scale at that time!] however the Induino evolved over a period of time to include many on-board peripherals increasing the experimentation capabilities and became the InduinoX.

**Comparison Table**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **InduinoX** | **Arduino UNO** |
| Microcontroller | Atmega328 | Atmega328 |
| USB Serial Converter | FT232RL | Atmega8u2 |
| ICSP Programming Headers | Yes | Yes |
| Arduino As a Programmer | Yes | Yes |
| On-Board Rx Tx LEDs | Yes | Yes |
| On-Board IO LEDs | 3 LEDS [PIN 11,12 & 13] | 1 LED [PIN 13] |
| On-Board Push Buttons | 3 [PIN 7,8 & 9] | None |
| On-Board Resettable Fuse | Yes | Yes |
| On-Board RTC | Yes | No |
| On-Board TSOP  (Remote Cotnrol Receiver) | Yes [PIN 15 a.k.a. A1] | No |
| On-Board RGB LED | Yes [PWM 3,5&6] | No |
| On-Board Sensor – LDR | Yes [Analog IN 3] | No |
| On-Board IR LED  (Generates Remote Control Signals) | Yes [PIN 14 a.k.a A0] | No |
| Shield Compatible | Yes | Yes |
| USB Connector Type | USB B | USB B |
| Programming IDE | Arduino | Arduino |
| Works on USB Power | Yes | Yes |
| External Power Supply Connector | 2 Pin Screw Terminal [Easy for Battery] | DC Adaptor Jack |
| Bit Bang Programming Headers | Yes | No |

Here are both the boards side by side. InduinoX on the Left Side and UNO on the right side.

[](http://4.bp.blogspot.com/-VrvY81BbnfY/Tu2PvMO4m-I/AAAAAAAAAcE/APOAxIqQJwU/s1600/Induino_uno_comp1.jpg)

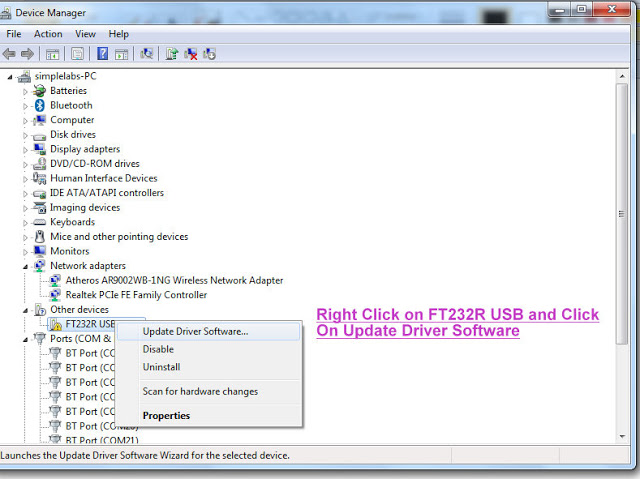
**InduinoX - Getting Started Guide**

**InduinoX Getting Started – For Windows XP / Windows 7**

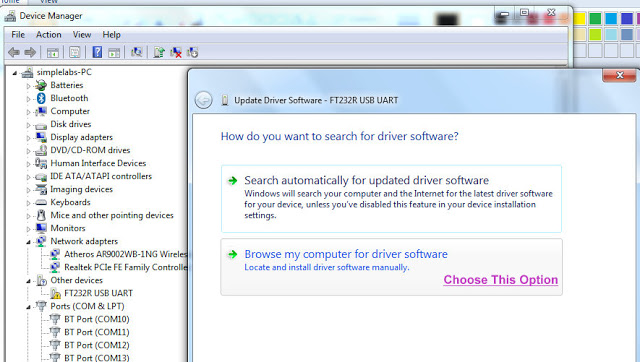
1. **Download the latest version of Arduino for Windows from here =>** [**http://arduino.cc/en/Main/Software**](http://arduino.cc/en/Main/Software)
2. **Unzip the downloaded file and remember its location**
3. **Connect the InduinoX board to your PC using the usb cable provided. You should see the power (red) led go on on the InduinoX board.**
4. **Windows will detect a new usb device. Ignore any options to install drivers and Do not let windows install any drivers automatically.**
5. **Open windows device manager**

**You can open windows device manager by right clicking on 'My Computer' and choosing Properties. In the new Window, Click on Device Manager [In Windows XP, this can be found under 'Hardware Tab' of the Properties Window]**

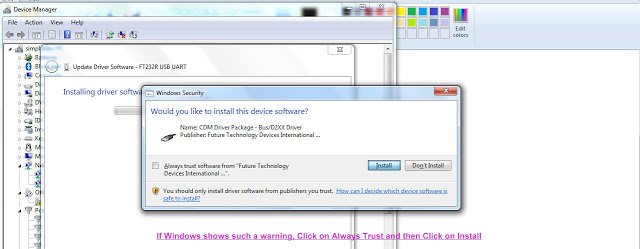
1. **Now Scroll Down the List of Devices in the Device Manager till you come to a Device 'FT232R USB UART'. You should see a warning icon next to it as the drives have not been installed. Right Click on this Device and Choose 'Update Driver Software...' option.**

[](http://3.bp.blogspot.com/-4XWpEyQbCog/Tu21koo_8WI/AAAAAAAAAdE/bRaiXqlfUpM/s1600/step2.jpg)

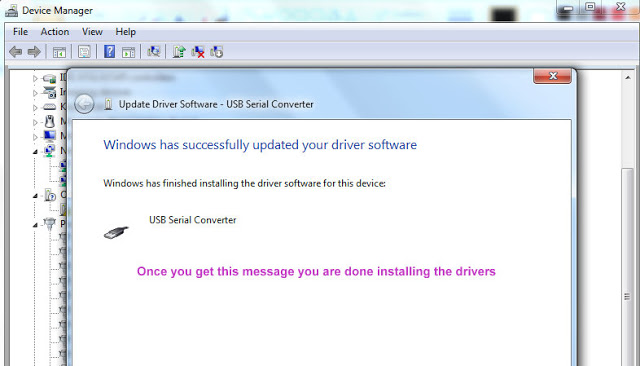
1. **Now, Windows will provide you two options, Choose the one that says “Locate and install drivers manually from a specific Location.”**

[](http://2.bp.blogspot.com/-imNsNNEDU58/Tu21swaKpjI/AAAAAAAAAdM/2mNJCzSX0gM/s1600/step3.jpg)

1. **In the next screen, you will see a browse button, click on the button and browse to the location of the drivers folder[You will find the drivers folder inside the unzipped arduino folder] and then click 'OK'. Then Click 'Next'**
2. **Now, Windows might throw a Security  warning, ignore it and click on next.**

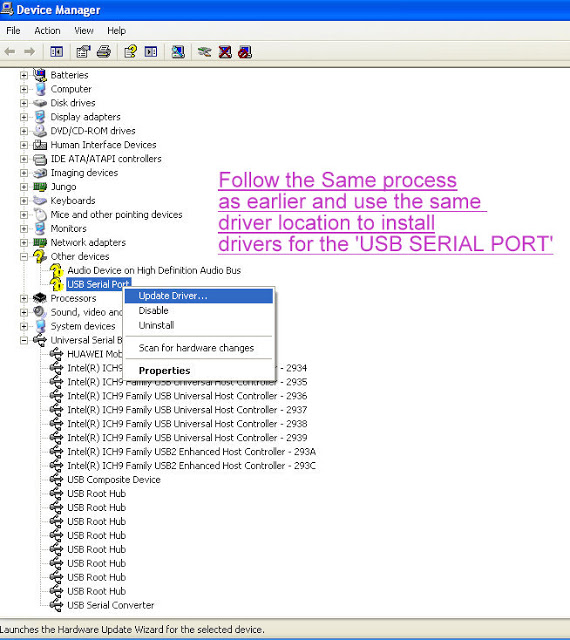
[](http://4.bp.blogspot.com/-rZwigRVlEgw/Tu219YE-H8I/AAAAAAAAAdc/SvsOh_LQrJQ/s1600/step11.jpg)

1. **The Device will be installed as a USB Serial Converter**

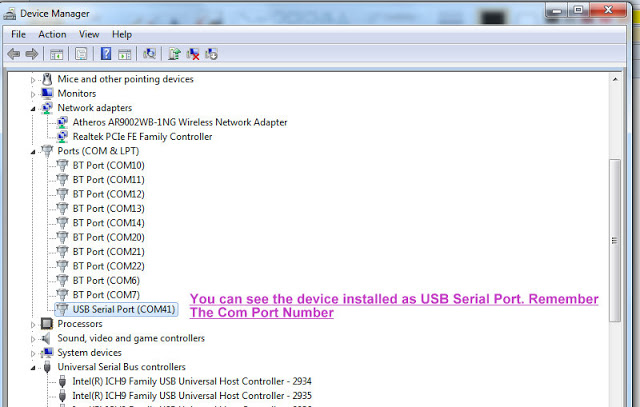
[](http://1.bp.blogspot.com/-sd7KGP6zloA/Tu22Kys0O9I/AAAAAAAAAdk/kjr9zmBqBog/s1600/step12.jpg)

1. **Now Windows will detect a USB Serial Port and ask you for drivers. Follow the exact same process as we did earlier, choosing the same driver location as earlier.**

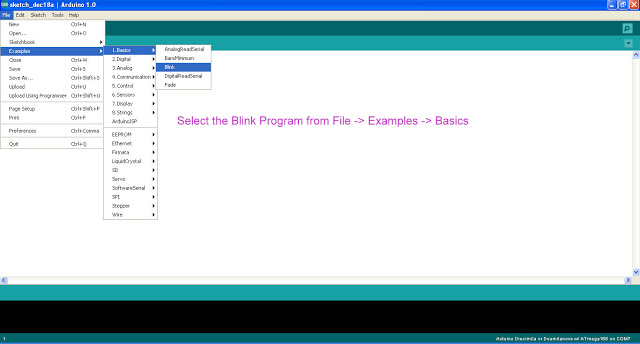
**In case Windows doesn't ask you, scroll through the Device manager and find the listing called 'USB Serial Port' and follow the above steps.**

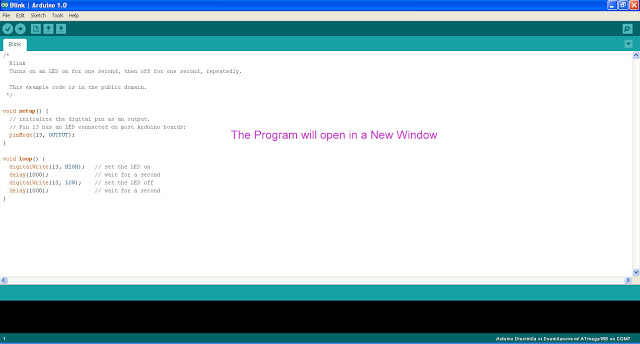
[](http://2.bp.blogspot.com/-IO0IGH2XEBw/Tu22Te6v8yI/AAAAAAAAAds/WGV7jPU27ys/s1600/new_step.jpg)

1. **Once everything is finished, You should be able to see a new device 'USB Serial Port (COMXX)'. Make a note of this Number.**

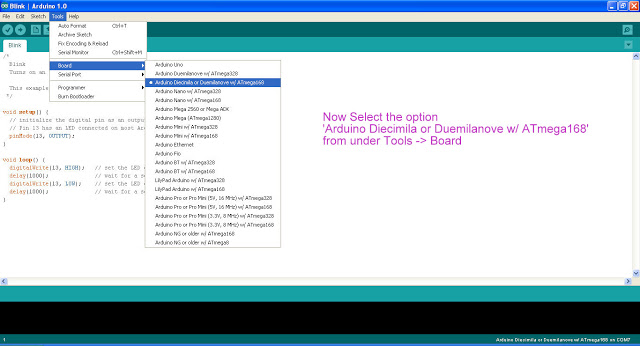
[](http://4.bp.blogspot.com/-lefeLI_ixuw/Tu22aYhnUOI/AAAAAAAAAd0/sHffpeR8vBM/s1600/step14.jpg)

1. **Now go the unzipped Arduino folder and run the arduino.exe file**
2. **The Arduino IDE will open up**
3. **In the Arduino IDE open the blink example program by clicking on it from under File -> Examples -> Basics**

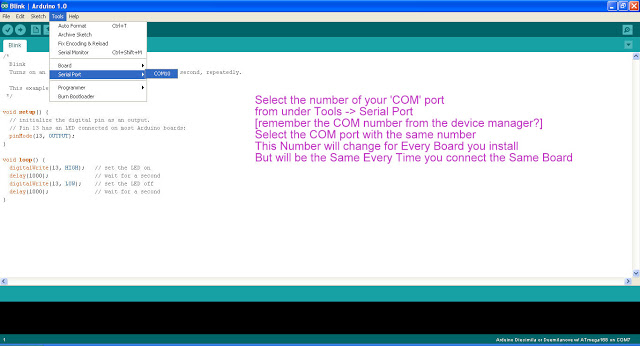
[](http://1.bp.blogspot.com/-o65vcb-aXI8/Tu22oNt3YNI/AAAAAAAAAd8/iQiWP3PTp6E/s1600/new_step1.jpg)

[](http://3.bp.blogspot.com/--6D8ldFu9wY/Tu22w-yhH_I/AAAAAAAAAeE/09I33Gk-3YE/s1600/new_step2.jpg)

1. **Now select the board by going to Tools -> Boards -> and selecting ~~'Arduino Diecimila or Arduino Duemilanove w/ Atmega168'~~ ARDUINO UNO**  
   **If you have one our recent board InduinoX with ATmega328, Select Arduino UNO as the board**

[](http://4.bp.blogspot.com/-S_5pRv6B7Ck/Tu23XOuh0iI/AAAAAAAAAec/wmaHtjOuldE/s1600/new_step3.jpg)

1. **Now select the Serial Port by going to Tools -> Serial Port -> and selecting the 'COM port that was earlier shown in the device manager'**

[](http://4.bp.blogspot.com/-czsIBUj-Ako/Tu22_Mrq7eI/AAAAAAAAAeM/qBltMMQN-AE/s1600/new_step4.jpg)

1. **Now Click on the Upload button. In a moment, you should see 2 LEDs Rx & Tx blink on your InduinoX indicating the program being uploaded. Once the program is uploaded, you will see the White LED[13th Pin] on the InduinoX Blink**

**InduinoX User Guide - Programming the Push Buttons**

In this tutorial, We'll see how to make use of the push buttons on the InduinoX board.

The push-buttons on the InduinoX are designed to work with the internal pull-up resistor on the microcontroller. For your understanding, They will give a low signal when the button is pressed. Enabling the internal pull up on the microcontroller will keep the corresponding pin HIGH unless the button is being pressed. When the button is being pressed, the corresponding pin will go LOW.

The three push-buttons are connected to Digital Pins 7, 8 & 9

So lets try to write a Simple Program to glow an LED while a button is being pressed. Then we will improvise our Binary Counter Program by adding a button to it.

Heres the code, the comments are self-explanatory

void setup()

{

pinMode(7,INPUT); // Declare the 7th pin as a input pin. We will use the button on the 7th pin

digitalWrite(7,HIGH); // enable the internal pullup resistor - Everytime you use a switch on the InduinoX, do this

pinMode(13,OUTPUT); // Our LED

}

void loop()

{

while(digitalRead(7)==0) // digitalRead(7) will read the current state of pin number 7 and give an output of '0' or '1'.

//In our case, the digitalRead() funciton will return a '0' when the button is being pressed and '1' when the button is not being pressed

// The Control will stay inside the while loop till the button is released

{

digitalWrite(13,HIGH); // Turn the LED ON

}

digitalWrite(13,LOW); // Turn the LED OFF when the control exits the While loop

}

Now Here's the switch added to the Binary Counter



and Here's the Code

/\*

This sketch increases a 3 bit number every time a button is pressed by the user and shows the output on 3 LEDs

\*/

int i = 0;

void setup()

{

pinMode(11,OUTPUT); // declare LED pins as output pins

pinMode(12,OUTPUT);

pinMode(13,OUTPUT);

pinMode(7,INPUT);// Declare the 7th pin as a input pin. We will use the button on the 7th pin

digitalWrite(7,HIGH);

}

void loop()

{

if(digitalRead(7)==0) // if the button is pressed

{

if(i<7) // if counter value is less than 7 or 3 bits

i++; // increment counter value

else

i=0; // reset counter to 0

int a=i%2; // calculate LSB

int b=i/2 %2; // calculate middle bit

int c=i/4 %2; // calculate MSB

digitalWrite(11,c); // write MSB

digitalWrite(12,b); // write middle bit

digitalWrite(13,a); // write LSB

while(digitalRead(7)==0); // wait till button is released to avoid incrementing the counter again

delay(100); // small delay to avoid debounce

}

}

**InduinoX User Guide - Programming the RGB LED**

In this edition we will see how to use the RGB LED on the InduinoX Board.  The RGB LED was included as part of the InduinoX board for the user to be able to experiment with analog outputs / PWM.

The InduinoX board has 6 PWM Pins [Pins 3,5,6,9,10,11] that can generate a PWM signal of 8-bit resolution.[8-bits can represent a maximum value of 255, and a 8-bit resolution here means that 5 volts is represented by 255 divisions. So if you want to generate 1 volt, you would use the value 51]

The RGB LED is connected to PWM Pins 3,5 & 6. [Blue to 3, Red to 5 & Green to 6] The RGB LED is a common Cathode Type LED

.

The Intensity of each of the Color Spectrum [Red, Green & Blue] can be varied by controlling the voltage applied on the individual pins. The PWM pins in addition to generating digital HIGH / LOW signals can generate analog voltages between 0 & 5. The analogWrite function will help us use these pins.  
  
The analogWrite function will take a 8-bit numerical value as a parameter [called duty cycle] and produce an output voltage corresponding to this value. It will set the pin to generate a steady square wave of the specified duty cycle at roughly 490Hz frequency.

So Lets try to experiment with the RGB LED. Try the following code

void setup()

{

}

void loop()

{

analogWrite(3,153);// Setting the voltage for Blue to around 3 Volts

analogWrite(5,51);// Setting the voltage for Red to around 1 Volt

analogWrite(6,51);// Setting the voltage for Green to around 1 Volt

}

Now you will see that the RGB LED glows with more of Blue color. Try to change the Values for the other Colors & experiment.  
  
Here's a small project implementing the LEDs, Buttons & the RGB LED.  
  
2 Buttons - 7 & 9 are used to increment and decrement the brightness of a chosen color in the RGB LED. A color can be chosen by using the Button on the 8th pin. The 3 regular LEDs are used to display the chosen color. When you choose Blue, the Blue LED will glow. In addition to this, pressing the Buttons 7 & 9 at the same time will cause the RGB LED to be reset to its initial state - all colors become zero. Note that the increment and decrement happen in a cyclic fashion ( 255 ++ will become 0 and 0 -- will become 255) Enjoy!  
  
**#### Video Errata - The RGB LED is a Common Cathode Type & Not Common Anode as mentioned in the Video####**

  
  
  
Here's the Source Code

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*### RGB LED COLOR CONTROL USING BUTTONS ###\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

int RGB[3]; // Variable to store individual Color Value of Different Colors of the RGB LED

int arr\_ind=0; // Variable for Navigating through the Above Array - An Aray Index Variable

void setup()

{

pinMode(7,INPUT); // The Increment Button

pinMode(8,INPUT); // The Color Select Button

pinMode(9,INPUT); // The Decrement Button

digitalWrite(7,HIGH); // Enabling the Internal Pull-Up Resistor for the Button

digitalWrite(8,HIGH);// Enabling the Internal Pull-Up Resistor for the Button

digitalWrite(9,HIGH);// Enabling the Internal Pull-Up Resistor for the Button

pinMode(11,OUTPUT);//LED for Current Selected Color Indication for RED Color - Lights up When the User Selects RED Color

pinMode(12,OUTPUT);//LED for Current Selected Color Indication for BLUE Color - Lights up When the User Selects BLUE Color

pinMode(13,OUTPUT);//LED for Current Selected Color Indication for GREEN Color - Lights up When the User Selects GREEN Color

RGB[0] = 0; // RGB[0] will store the value for the BLUE Color

RGB[1] = 0; // RGB[1] will store the value for the RED Color

RGB[2] = 0; // RGB[2] will store the value for the GREEN Color

applyColor(); // Calling a Function that will handle the AnalogWrite functions for the RGB LED

}

void loop()

{

if(digitalRead(7)==0) // Checking if the Increment button is Being Pressed, If True, the value of the currently selected color's value is incremented

{

if(RGB[arr\_ind]<255) // Checks if the currently selected color value is lesser than 255 before incrementing. So when it reaches 255, the value is reset to 0.

RGB[arr\_ind]++;

else

RGB[arr\_ind]=0;

delay(100);

}

if(digitalRead(9)==0)// Checking if the Decrement button is Being Pressed, If True, the value of the currently selected color's value is decremented

{

if(RGB[arr\_ind]>0)// Checks if the currently selected color value is greater than 0 before decrementing. So when it reaches 0, the value is reset to 255.

RGB[arr\_ind]--;

else

RGB[arr\_ind]=255;

delay(100);

}

if(digitalRead(8)==0)// Checking if the color button is Being Pressed, If True, the value of the array index is incremented to the next value

{

if(arr\_ind<2)

arr\_ind++;

else

arr\_ind=0;

while(digitalRead(8)==0); // This while is used to debounce the button press or in other words, wait for the user to release the button

delay(50);

}

if((digitalRead(7)==0)&&(digitalRead(9)==0))// Checking if both the increment & decrement buttons are being pressed at the same time. If so, all color values are reset to zero

{

RGB[0] = 0; // RGB[0] will store the value for the BLUE Color

RGB[1] = 0; // RGB[1] will store the value for the RED Color

RGB[2] = 0; // RGB[2] will store the value for the GREEN Color

digitalWrite(11,HIGH);digitalWrite(12,HIGH);digitalWrite(13,HIGH);// This to indicate a reset in progress. All three LEDS GLOW for 200 milliseconds and go Off

delay(200);

digitalWrite(11,LOW);digitalWrite(12,LOW);digitalWrite(13,LOW);

}

switch(arr\_ind) // The switch is used to indicate the current color selection through the corresponding LED based on the current value of the Array Index

{

case 0: digitalWrite(11,LOW);digitalWrite(12,HIGH);digitalWrite(13,LOW);break;

case 1: digitalWrite(11,HIGH);digitalWrite(12,LOW);digitalWrite(13,LOW);break;

case 2: digitalWrite(11,LOW);digitalWrite(12,LOW);digitalWrite(13,HIGH);break;

}

applyColor();// Calling a Function that will handle the AnalogWrite functions for the RGB LED

}

// The function applyColor() will apply the RGB array variable's current value to the Analog Pins 3,5 & 6 which control the RGB LED

void applyColor()

{

analogWrite(3,RGB[0]);

analogWrite(5,RGB[1]);

analogWrite(6,RGB[2]);

}

**InduinoX User Guide - Programming the LDR**

The LDR on the InduinoX is connected to Analog Pin 3 through a Voltage Divider configuration. The Voltage divider is in such a configuration that the voltage sensed by the Analog Pin3 would be inversely proportional to the light incident on the LDR. The voltage would vary in the range of 0-5 Volts  
  
The Analog Input on Arduino is of 10-bit resolution. 10-bit resolution means that the Voltage range of 0-5 Volts is represented in 1024 steps from 0-1023. So you would be reading an input value in the range of 0-1023 where 1023 would correspond to 5 Volts(or very very less light!)  
  
Arduino[the IDE!] comes with a serial library that can be used to transmit data serially to a computer. We shall make use of this library to transmit our LDR value to a Computer every 1 second.  
  
Here's a Video of the Sample Program



Here's the Source Code.

int val = 0; // A Variable to Store the Light Value from the LDR

void setup()

{

Serial.begin(9600);// Start a Serial Connection

}

void loop()

{

val = analogRead(3);// Reads a 10-bit value corresponding to the voltage applied on analog input pin 3.

Serial.print("Light Intensity is : ");// Prints the given string / value to the serial monitor

Serial.println(val);// prints the value of the variable val to the serial monitor and moves the cursor to the next line (the ln part of println does this

delay(1000);

}

Now Lets build a Simple Lighting Level Controller using the LDR and the 3 on-board LEDS[11,12 & 13]. We will try and increase / decrease the number of lights that are ON based on changing Light Intensity Levels Measured by the LDR.  
  
Make note of the values shown in the serial monitor for various levels of light intensity. Use these values as threshold and every time the LDR value increases above one of the thresholds, switch On a LED and everytime it decreases below the threshold, switch Off the LED.  
  
Here's a Video of the project in Action



Here's the source code

/\*

### LDR Based Lighting Level Controller ###

This is a simple program where 3 LEDS are switched ON / OFF one by one as the Light Intensity Sensed by the LDR Decreases / Increases

\*/

#define threshold1 650 // First Threshold Value of Darkness above which the first LED is switched ON

#define threshold2 750 // Second Threshold Value of Darkness above which the second LED is switched ON

#define threshold3 950 // Third Threshold Value of Darkness above which the third LED is switched ON

int val = 0; // A Variable to Store the Light Value from the LDR

void setup()

{

pinMode(11,OUTPUT); // LED 1

pinMode(12,OUTPUT); // LED 2

pinMode(13,OUTPUT); // LED 3

Serial.begin(9600);// Start a Serial Connection

}

void loop()

{

val = analogRead(3);// Reads a 10-bit value corresponding to the voltage applied on analog input pin 3.

Serial.print("Light Intensity is : ");// Prints the given string / value to the serial monitor

Serial.println(val);// prints the value of the variable val to the serial monitor and moves the cursor to the next line (the ln part of println does this

if(val > threshold1) // Checks & Turns the First LED ON / OFF based on Light Intensity

digitalWrite(11,HIGH);

else

digitalWrite(11,LOW);

if(val > threshold2) // Checks & Turns the Second LED ON / OFF based on Light Intensity

digitalWrite(12,HIGH);

else

digitalWrite(12,LOW);

if(val > threshold3) // Checks & Turns the Thirdf LED ON / OFF based on Light Intensity

digitalWrite(13,HIGH);

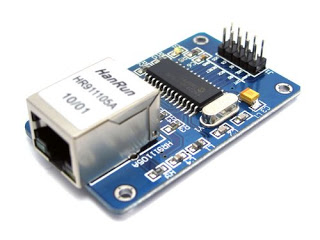
else

digitalWrite(13,LOW);

}

**InduinoX User Guide - Interfacing with the ENC28J60 Ethernet Module**

**The ENC28J60 Module**

[](http://3.bp.blogspot.com/-98a1NBXhuEs/UBY5KkPNO0I/AAAAAAAAAFs/B9WvTUappCM/s1600/Ethernetmodule1.jpg)

The [ENC28J60 Ethernet Module](http://www.simplelabs.co.in/content/enc28j60-ethernet-module) is a low-cost ethernet module that offers ethernet connectivity over SPI. This module is different from the ENC28J60 Ethernet Shield only in its form factor and the fact that there is no SD card slot. This module can be interfaced with any microcontroller over SPI.  We will see how to use this module with Arduino.

Its been use with the arduino for sometime and there are a number of libraries available to work with the module.

You can download the following zip file and extract its contents to the libraries folder of your arduino installation.

There are 2 folders in this library - etherShield and ETHER\_28j60. The etherShield is a base library and the ETHER\_28j60 is a wrapper around this based library that simplifies the functionalities of the etherShield library for the beginner. You can program your Ethernet Shield directly using the etherShield Library however, this library requires you have a finer understanding of networking and how it works. We suggest you stick to the ETHER\_28J60. You can find documentation on this [here](http://www.doctormonk.com/2010/03/simplified-ethernet-library-for-28j60.html).

[Download Library](http://downloads.simplelabs.co.in/ENC28j60.zip)

**Module Overview**

There are 10 Pins on the Module that bring out SPI interface and additional functionalities of the ENC28J60 IC. The Module is to be powered from 3.3Volts supply.

**Connections Overview**

Connect the pins of the module to pins of your Arduino as in the table

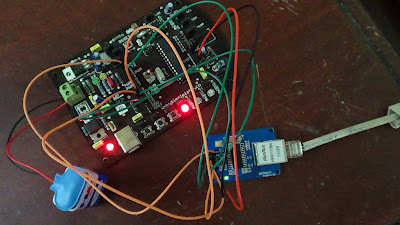
|  |  |  |
| --- | --- | --- |
| MODULE | InduinoX / Arduino UNO (ATMEGA328) | Arduino MEGA |
| VCC | 3.3V Pin | 3.3V Pin |
| GND | GND | GND |
| CS | SS - PIN 10 | SS - PIN 53 |
| SI | MOSI - PIN 11 | MOSI - PIN 51 |
| SCK | SCK - PIN 13 | SCK - PIN 52 |
| WOL | PIN 5 | PIN 5 |
| SO | MISO - PIN 12 | MISO - PIN 50 |
| INT | Interrupt 0 - PIN 2 | Interrupt 0 - PIN 2 |

**A Simple Test Program**

Once you install the libraries, you can try the 'HelloWorld' program from ETHER\_28j60 library. Upload the program, disconnect the board (make it standalone powered externally!), open your browser and type in the ip address of the board. (http://192.168.1.15).. Voila! now your arduino is on the network!

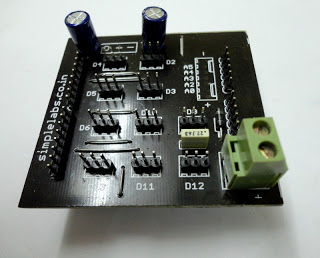
**Controlling the InduinoX**

The 'HelloWorld' is a simple program now to try the 'WebRemote' example. This will let you control the 6th pin on the InduinoX. You can modify this to achieve your web control needs.

[](http://1.bp.blogspot.com/-M9dni-PZBr0/UBY6D2Vk2AI/AAAAAAAAAF0/yzPORyYkdDc/s1600/DSC00012.JPG)

**InduinoX User Guide - Interfacing with the Simple Labs' Servo Shield**

[The Simple Labs' Servo Shield](http://www.simplelabs.co.in/content/simple-labs-servo-shield) is a custom designed Shield to drive Servo Motors. The Shield Can drive upto 10 servo motors at a time. It was originally designed for our [Quadbot Robot Kit](http://www.simplelabs.co.in/content/quad-bot-robot-kit-servo-shield) (8 Servos) + [Mini Pan and Tilt Kit](http://www.simplelabs.co.in/content/mini-pan-and-tilt-kit) (2 Servos). In addition to this, the servo shield has suitable pin outs to be able to Connect the [Compound Eye IR Sensor](http://www.simplelabs.co.in/content/compound-infrared-sensor).

[](http://3.bp.blogspot.com/-GotToNJ2FTY/UA_fjW0b8fI/AAAAAAAAAFM/qNzJxruq3iU/s1600/Servo_shield1.JPG)

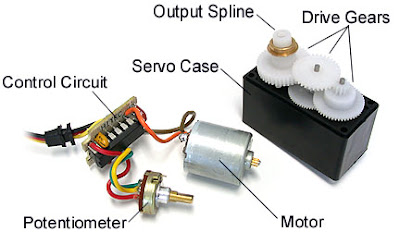
**Servo Motors - What are they?**

Servo motors are dc motors with a servo mechanism that lets us control the precise position of the Shaft.  
  
A servo mechanism is a error correction mechanism that senses the negative error and applies a correction accordingly.  
  
There are many types of Servo Motors available - depending upon applications.  
They are used in areas requiring position control. While the basic working of all the servo motors is almost similar, for the scope of this tutorial we shall stick to discussing RC Servo Motors or the type of Servo motors used in Robotics / Radio Control Projects.

[](http://3.bp.blogspot.com/-6z5LFKYX08g/UA0INugdVWI/AAAAAAAAAEU/whie0UFbXjA/s1600/hitec-hs-5745mg.jpg)

A Servo motor constitutes of 4 parts - a DC motor, a Gear system, A Potentiometer(for sensing feedback) and a Control Circuit. Here the Control Circuit and the Potentiometer together form the Servo Mechanism.

[](http://3.bp.blogspot.com/-WY2pV0Y7tDI/UA0IOXsIQLI/AAAAAAAAAEc/RHgI5m6VXj8/s1600/hitec_servo_b.jpg)

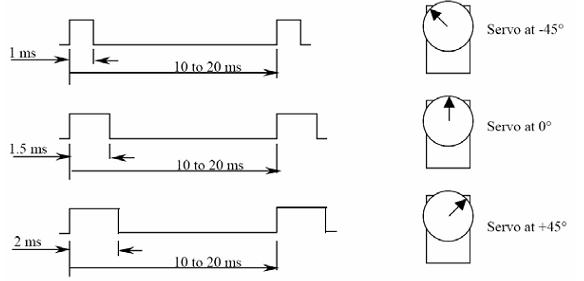
[[](http://2.bp.blogspot.com/-50KtwAX6Fvc/UA0EWjKchHI/AAAAAAAAAEI/i3CdGsLPb0g/s1600/Servo_Breakdown.jpg)](http://2.bp.blogspot.com/-50KtwAX6Fvc/UA0EWjKchHI/AAAAAAAAAEI/i3CdGsLPb0g/s1600/Servo_Breakdown.jpg)

**So How do Servo Motors Work?**

The Control Circuitry on the Servo motor drives the DC motor based on input signals it receives, The output shaft of the DC motor is connected to the gear assembly and the gear assembly is such that it moves the potentiometer when it rotates. The Control Circuitry gets feedback from this potentiometer and spins the DC motor to achieve the desired angle between 0 & 180 degrees. The range of movement of the potentiometer is mechanically restrained.

**And How do we Control these Servo Motors?**

To Control a servo we need to give it a pulse once every 20 milliseconds. The Duration of this pulse will determine the Servo Angle which we need to achieve. For most Servo's a Pulse duration of 1millisecond will set the shaft position to 0 degrees and a pulse duration of 2 milliseconds will set the shaft position to 180 degrees.

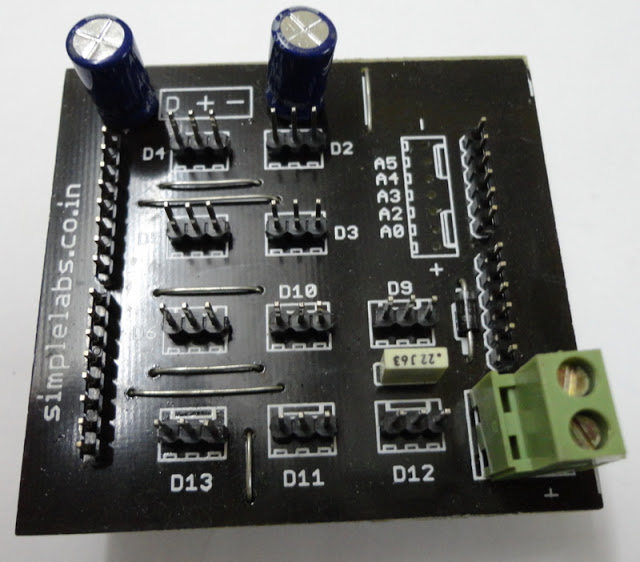
[](http://3.bp.blogspot.com/-JWQvR1bRyVc/UA9-HOMjmOI/AAAAAAAAAEo/2bkJUAUD45I/s1600/servo2.jpg)

**Controlling Servos with an Arduino**

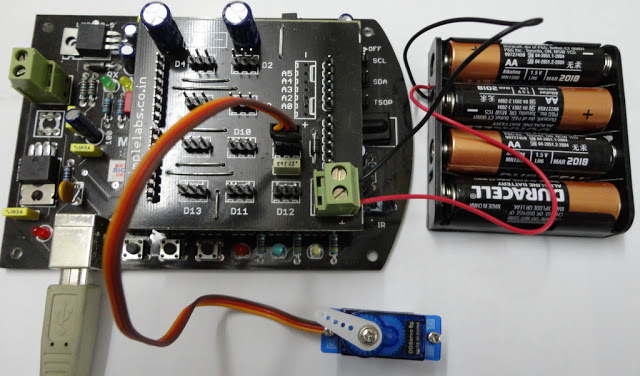
Arduino provides us with a servo control library that lets us control servos with ease. The library provides us with an option of writing both microseconds value and angle values. You can find more information on the library here - > [Arduino Servo Library](http://arduino.cc/it/Reference/Servo)

**Overview of the Servo Shield**

[The Servo Shield](http://www.simplelabs.co.in/content/simple-labs-servo-shield) has clear pin mappings and polarity mappings for the Servos. It has a screw terminal for connecting an external power supply for the Servos. Servos require 4.8-6Volts and consume around 300-500milliamps. The Servo shield requires to be powered from an External Source for it to drive your servos.

[](http://2.bp.blogspot.com/-yF_7OFVspVg/UA-LEi8olJI/AAAAAAAAAE0/C5BNChEJhck/s1600/Servo_Shield2.JPG)

The servo shield has Servo control pins on Arduino Digital Pins D2, D3, D4, D5, D6, D9, D10, D11, D12 & D13. Refer to the TOP Legend on the board "D + - " for the Orientation of the Servo Connector Connection.  
  
Connect a Servo to D9 and an external Supply to the Battery Connector as Shown a[4xAA Battery Connector](http://www.simplelabs.co.in/content/battery-holder-4xaa-square) is used here in the image. The Servo shown in the image is part of the Quadbot Kit. If you are looking for a servo, you can check out the [Tower Pro SG-5010](http://www.simplelabs.co.in/content/towerpro-sg-5010-servo)

[](http://3.bp.blogspot.com/--Lvo26lBdqo/UA_cozGW9pI/AAAAAAAAAFA/rmhv9F6E_aM/s1600/Servo_Setup.JPG)

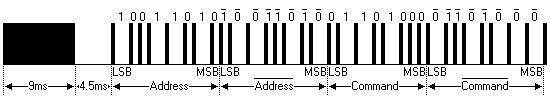
**The Program**

With the above setup you can load the 'Sweep' Program from within the Examples. This program is located at

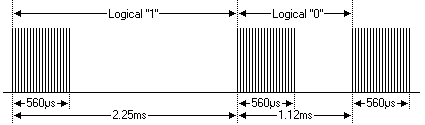
File -> Examples -> Servo -> Sweep (Inside the Arduino IDE!)

**InduinoX User Guide - Working with NEC Remotes**

The Simple Labs InduinoX Arduino Learners' Kit comes with a small remote, the kind of remote you would find with a MP3 Player / Audio Players. Owing to their size these remotes have spread fast and hence this tutorial on how to decode these manually without using the library. If you want to use the library, [check out our previous post on this](http://induino.blogspot.in/2011/12/induinox-user-guide-working-with-tsop.html)  
  
These small remotes use the NEC protocol which is a 32-bit protocol and much different from the SIRC Protocol we saw earlier. Here's the signal train of the NEC protocol

[](http://www.sbprojects.com/knowledge/ir/nectrain.gif)

The Start Bit is a 9 ms long pulse followed by 4.5 ms space. This is  followed by a burst of 32-bit data. The 32-bit data consists of 8-bits of command and 8-bits of address which are transmitted twice for reliability. The first transmission of 8 bits is inverted when it is being transmitted the next time for increased reliability.  
  
  
Heres the Pulse Modulation of the NEC Protocol

[](http://www.sbprojects.com/knowledge/ir/necmodulation.gif)

The Logical '1' is represented by a 560us long 38khz pulse (MARK) followed by a 1690us long LOW signal (SPACE). The Total Duration for the Logical '1' is 2.25us.  
  
The Logical '0' is represented by a 560us long 38khz pulse (MARK) followed by a 560us long LOW signal (SPACE). The Total Duration for the Logical '1' is 1.12us.  
  
So How do we read this?  
  
For our purpose, we can read this as a 16-bit data (8-bit address & 8-bit command)  
  
  
To read this signal here are the things we need to do

* Wait till be get the start bit
* Read the time duration for the subsequent 32bits
* Sort through the time duration in 4 parts of 8-bits each
* Calculate 2 values, one from the bits 0-7 and 16-23 and one from bits 8-15 & 24-31
* Check if these values are equal. If so take the value as valid. if not ignore the value.

Here's the code for a function to read NEC remotes. This function works for the InduinoX board.

int remote()

{

  int val1=0,val2=0, dur[32], i=0, j=0;

  if(pulseIn(15,LOW)>8000)

  {

    for(i=0; i<32;i++)

    {

      dur[i]=pulseIn(15,HIGH);

    }

//Value Calculation

    for(i=0,j=0;i<24;i++,j++)

    {

      if(i==8)

         i=16;

      if(dur[i]>1000)

          val1 = val1+ (1<<j);

    }

    for(i=8,j=0;i<32;i++,j++)

    {

      if(i==16)

        i=24;

      if(dur[i]<1000)

        val2 = val2+ (1<<j);

     }

**Serial**.println(val1);

**Serial**.println(val2);

     if(val1==val2)

       return val1;

     else

       return 0;

  }

  return 0;

}

[http://img1.blogblog.com/img/icon18_email.gif](http://www.blogger.com/email-post.g?blogID=2321591172991593577&postID=2511397916446852289)

**InduinoX User Guide - Interfacing with the Wireless Shield / BTBee**

Bluetooth control seems to be the In-Thing! hook up your InduinoX with your Android phone over Bluetooth and zap you go... With a little bit of Android Expertise, you can double the potential of your Arduino Projects.



**So How does one hook up the InduinoX over Bluetooth?**

For this, we are going to use and [Xbee shield](http://www.simplelabs.co.in/content/simple-labs-wirelss-shield-btbee-xbee-breakout-shield) and a Bluetooth Module - [BTBee](http://www.simplelabs.co.in/content/simple-labs-bluetooth-module-btbee)

The Xbee shield is a low cost arduino shield for plugging in any device with the XBee foot print. The shield offers the flexibility of using any of the digital pins for Rx/Tx through a jumper setting mechanism. In our case, we will stick to use the default Rx/Tx pins of the InduinoX. For those of you willing to explore further, check out [Software Serial using arduino](http://arduino.cc/hu/Tutorial/SoftwareSerial).

The BTBee is a low cost serial bluetooth module with the same footprint as an Xbee. The BTBee has 2 LEDS a Red led for Power and a Green led for status indication. The Green led will be constantly blinking till the device is paired. Beware! the BTBee has a lot of dummy pins just to match the XBee footprint. In effect, it uses only 5 pins

**Pin  Description**

**1 VCC  Power supply**

**2 TX  Data output**

**3 RX  Data input**

**5 Reset reset**

**10 GND  Ground**

Place the BTBee on the Xbee shield as shown in the video. Upload the following test program onto the InduinoX and then place the shield on top of the InduinoX board. Remember place the Xbee shield only after uploading the program as otherwise you might face an error. (Since we use serial communication for programming, if there are other devices attached to the default serial port, programming the microcontroller will throw up an error.)

The test program expects a serial character 'A' to turn ON the led on the 13th pin and another character 'B' to turn it OFF.

Next Download the following App for your android phone [BlueTerm](http://www.appbrain.com/app/blueterm/es.pymasde.blueterm)

The BlueTerm App is a terminal app for Android that can simulate a Serial Terminal over Bluetooth.

Power your InduinoX (you can use USB power), turn on Bluetooth on your phone and scan for devices. If this is the first time you are pairing with the BTBee module you will see a set of numbers (the device id) try to pair with this using the code '1234' once it is paired, it will show up as 'linvor'. Now open the app and connect to linvor. Once connected, the app will show a message as connected and the Green led on the BTBee will stop blinking and remain stable (ON) Now type in the characters and get control of the LED on the InduinoX!

Here's the sample code. Make fun with Bluetooth!

void setup()

{

Serial.begin(9600);

pinMode(13,OUTPUT);

}

void loop()

{

if(Serial.available())

{

int val = Serial.read();

if(val == 65)

digitalWrite(13,HIGH);

if(val==66)

digitalWrite(13,LOW);

Serial.println(val);

}

}

**InduinoX User Guide - Interfacing with the LCD Shield**

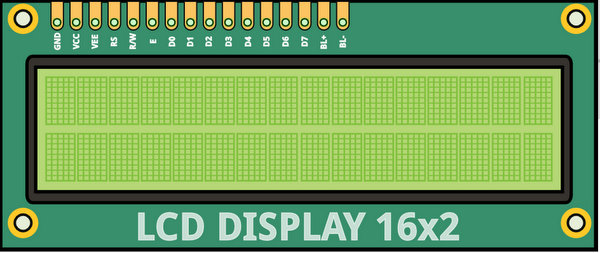
The LCD Shield eases the process of interfacing a LCD display with the Arduino. The shield fits on top of any Arduino and takes power from the Arduino.

[](http://3.bp.blogspot.com/-iO_TYXo7VT4/T1LIhuCMKeI/AAAAAAAAAe8/sauNJ4qxJPE/s1600/LCD_shield.jpg)

**How to place the Shield on top of an Arduino?**  
The shield has one of set 8 pins and 1 set of 6 Pins. The trimpot on the shield is on the top left side. Now place the shield onto your arduino such that the eight pins are placed on top of pins 8,9,10,11,12,13,gnd & aref and the six pins are placed on top of pins Vin, gnd, gnd, +5V, +3.3V & reset



The LCD shield is designed to work with 16 pin LCDs with the following pin configuration.

[](http://4.bp.blogspot.com/-o5itaO6TcvE/T1LcUU3-B3I/AAAAAAAAAfI/YmUH-EZQ_Is/s1600/LCD_Pins.jpg)

**LCD Pin Mappings for the Shield**  
The LCD shields provides a 4-bit mode interface to the Arduino. The Pins are connected as follows.   
RS => 8  
E => 9  
D4 =>10  
D5 =>11  
D6 =>12  
D7 =>13  
  
You can run the Hello World Example under File->Examples->LiquidCrystal->Hello World, with the following changes  
**replace this line  
~~LiquidCrystal lcd(12, 11, 5, 4, 3, 2);~~  
with  
LiquidCrystal lcd(8,9,10,11,12,13);**  
  
**Character LCD - A Quick Overview**  
Here's an interesting Write up to get you understanding the working of the LCD  
http://joshuagalloway.com/lcd.html

|  |  |
| --- | --- |
|  |  |

Simple Labs. Simple template. Template images by [luoman](http://www.istockphoto.com/googleimages.php?id=11394138&platform=blogger&langregion=en_GB). Powered by [Blogger](http://www.blogger.com).

**InduinoX User Guide - Programming the Arduino Booloader**

The InduinoX was designed with FTDI bitbang mode in mind and there are Pinouts from the FTDI for the same. We will here focus only on how to wire the InduinoX to achieve the BitBang Method described at the following link  
  
<http://www.geocities.jp/arduino_diecimila/bootloader/index_old_en.html#mega8_328>

You can download Serjtag from the above link and use the commands from the same.

Check this video for the wiring



**Things to know**

*Where to find Fuse Settings for arduino?*

The fuse settings for different microcontrollers used in different configurations are available in the boards.txt file in your arduino install. You can find this file here

Your Arduino Directory > Hardware > arduino > boards.txt

*Where to find the Bootloader for arduino?*

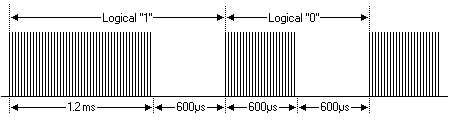
Your Arduino Directory > Hardware > arduino >bootloaders

All the bootloaders have their own directories, so choose the One you want.

Keep us posted on how it goes, mail us on info|at|simplelabs|dot|co|dot|in

**InduinoX User Guide - Working with the IR LED**

The IR Led is a unique idea we had when designing the InduinoX board. We thought if we can put bot the IR LED and TSOP side by side and make 2 boards face each other, we can build a simple basic IR based 2 way wireless communication system. In this edition, we will see how to generate sony signals on the IR led  
  
The IR led is connected to PIN 14 (analog 0) of the InduinoX.  
  
Understanding Frequency  
The Sony SIRC protocol works at 40Khz...most remote controls work at their specific frequencies. TV remotes in India are generally between 32- 40Khz frequecny.  
  
So whats this frequency all about?  
40KHz means 40,000 Cycles in 1 second. Each Cycle represents a ON signal followed by a OFF signal. The signals could be of equal duration.  
  
Heres another image of the Sony signals

[](http://www.sbprojects.com/knowledge/ir/sircmodulation.gif)

If you look at the image, you can see the the 1.2ms high of the Logical '1' has further black lines with spaces in between. These correspond to the ON/OFF cycles. The space between these is what is called the frequency. The frequency of occurrence of a ON/OFF cycle is what it means.  
  
So How do we generate it?  
Lets do some calculation,  
40,000 Cycles in 1 Sec or 1000 millisec or 1000 x 1000 microseconds  
so each cycle if for a duration of 25microseconds.  
  
We can produce this frequency if we can make a pin go high for 13 microseconds followed by low for 12 microseconds.  
  
If we can do this for 2.4 milliseconds then we can generate a Startbit, if we can do this for 1.2 milliseconds then we can generate a Logical '1', and for 0.6 milliseconds, we can generate a Logical '0'.  
  
Now that we know that each cycle will take 25 microseconds lets calcualte the number of cycles required for generating the 2.4 milliseconds start bit  
2.4 milliseconds / 25microseconds => 2400 microseconds / 25microseconds => 96 cycles  
1.2 milliseconds / 25microseconds => 1200 microseconds / 25microseconds => 48 cycles  
0.6 milliseconds / 25microseconds => 600 microseconds / 25microseconds => 24 cycles  
  
the delayMicroseconds() function of arduino can help us required delay in microseconds. So lets first create a pulse function.

void pulseData(int no\_of\_cycles)

{

for(int i=0;i<no\_of\_cycles;i++)

{

digitalWrite(14,HIGH);

delayMicroseconds(13);

digitalWrite(14,LOW);

delayMicroseconds(12);

}

digitalWrite(14,LOW);

delayMicroseconds(600);// This delay is for the space between the signals 0.6 millseconds

}

Now we have a function that can generate 40Khz signal cycles for a specified number of cycles.  
  
Next we need a function that will convert a decimal value to binary, evaluate each bit and then call signal generation function accordingly.  
  
Heres a function that takes any decimal value (upto 12 bits), converts it to binary and transmits the same.

void dec\_to\_bin\_transmit(int val)

{

int cnt = 0;

pulseData(96); // Sending the Startbit

while(cnt<12) // Execute this 12 times to send 12 bits of data

{

if(val>0) // Checking if the decimal value is non-zero

{

if(val%2 == 0) // Binary reminder check, if reminder is 1 then we need to send a logical '1'

{

pulseData(24);

}

else// Binary reminder check, if reminder is not 1 then we need to send a logical '0'

{

pulseData(48);

}

val = val / 2;

}

else // when the decimal value becomes zero, start sending zeroes for the remaining bits

{

pulseData(24);

}

cnt++;

}

}

Note: This function is not optimal ;) there might be data loss. We will leave it to you to figure out how to optimise it.  
  
Here's a simple program where the IR led transmits values from 0 to 1023. It transmits each value thrice.

void setup()

{

pinMode(14,OUTPUT);

}

void loop()

{

for(int count=0; count<1023; count++)

{

for( int j=0;j<3;j++)

{

dec\_to\_bin\_transmit(count);

delay(50);

}

delay(500);

}

}

void dec\_to\_bin\_transmit(int val)

{

int cnt = 0;

pulseData(96); // Sending the Startbit

while(cnt<12) // Execute this 12 times to send 12 bits of data

{

if(val>0) // Checking if the decimal value is non-zero

{

if(val%2 == 0) // Binary reminder check, if reminder is 1 then we need to send a logical '1'

{

pulseData(24);

}

else// Binary reminder check, if reminder is not 1 then we need to send a logical '0'

{

pulseData(48);

}

val = val / 2;

}

else // when the decimal value becomes zero, start sending zeroes for the remaining bits

{

pulseData(24);

}

cnt++;

}

}

void pulseData(int no\_of\_cycles)

{

for(int i=0;i<no\_of\_cycles;i++)

{

digitalWrite(14,HIGH);

delayMicroseconds(13);

digitalWrite(14,LOW);

delayMicroseconds(12);

}

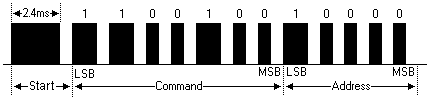
digitalWrite(14,LOW);

delayMicroseconds(600);// This delay is for the space between the signals 0.6 millseconds

}

**InduinoX User Guide - Working with TSOP IR Receiver - Part 2**

We hope you had success with the first part of this write-up. If you have not read it, we suggest you go through it before reading ahead. [InduinoX User Guide - Working with TSOP IR Receiver - Part 1](http://induino.blogspot.com/2011/12/induinox-user-guide-working-with-tsop.html)  
  
Now that we've used the library successfully, we will see how to code manually to receive from a Sony Remote. Later in the next part, we will see how to generate Sony Signals. Its advised that you read this and try this out completely before moving onto signal generation  
  
Lets take a look at the sony signal train

[](http://www.sbprojects.com/knowledge/ir/sirctrain.gif)

Now Here's how we are going to go ahead with decoding

1. Keep checking the TSOP pin (Pin 15) for a LOW pulse of duration in excess of 2ms, the moment you get such a signal proceed on to step 2
2. Run a 'for' loop for 12 counts, during each iteration of the loop, get the current pulse duration using the pulseIn function. Check if the duration is greater than 1000ms (means its a '1') else its a '0'
3. As soon as you detect a '1' or '0' add it to an appropriate binary to decimal conversion logic.

Here's the code, You can plug in the remote function into any of your programs, just remember to declare pin 15 as INPUT.

void setup()

{

pinMode(15,INPUT); // TSOP is connected on the 15ht pin

Serial.begin(9600);

}

void loop()

{

int remote\_val = remote();

if(remote\_val>0)

{

Serial.println(remote\_val);

delay(150); // A remote press will normally generate 3 signal trains. This is to avoid reading duplicates

}

}

int remote()

{

int value = 0;

int time = pulseIn(15,LOW);

if(time>2000) // Checking if the Start Bit has been received. Start Bit Duration is 2.4ms

{

for(int counter1=0;counter1<12;counter1++) // A loop to receive the next 12 bits

{

if(pulseIn(15,LOW)>1000) // checking the duration of each pulse, if it is a '1' then we use it in our binary to decimal conversion, '0's can be ignored.

{

value = value + (1<< counter1);// binary to decimail conversion. 1<< i is nothing but 2 raised to the power of i

}

}

}

return value;

}

**InduinoX User Guide - Working with TSOP IR Receiver**

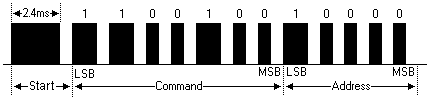
The TSOP SM0038 is an IR receiver on the InduinoX. The TSOP will help you to interface your TV remote with the InduinoX and in the Process learn the basics of Wireless Communication. The TSOP is connected to pin digital 15(Analog 1).

The TSOP outputs a constant HIGH signal when idle and as it receives data, it tends to invert the data. i.e when an IR LED is transmitting data onto the TSOP, everytime the IR led goes high, the TSOP will go LOW and vice versa. Remote control signals are often bytes of data that is encoded and transmitted by pulsing(switching ON & OFF the IR LED at a specific frequency) Most TV remote controls work at 32-40 Khz frequency and most receivers can receive this range.

Heres a link to a nice write up on different remote control protocols. lets first take a look how the Sony Remote Control Protocol Works. We stick to Sony as it is the easiest one to get started with. [Read this before proceeding](http://www.sbprojects.com/knowledge/ir/sirc.php)

Here's a basic outline of how the data is sent. Every time you press a button on a Sony remote control, it sends out a 13Bit data. The first bit is a start bit indicating there are 12 bits of data following it. The next 7 bits are the command bit which will vary depending upon the keys being pressed. The last 5 bits are the address bits which will the same for all buttons but vary for remote controls of different devices.

The black bars in the following image correspond to high signals (called marks) and the white spaces in between correspond to low signals (called spaces). The duration of the 'marks' varies according to the bit being transmitted. It is 2.4ms for the start bit, 1.2ms for HIGH bit and 0.6ms for LOW bit. The duration of the 'spaces' is a constant 0.6ms. Every mark is followed by a space. Any data can be converted to binary format and transmitted in this manner. In fact this is the basic form of all types of serial communication.

[](http://www.sbprojects.com/knowledge/ir/sirctrain.gif)

Technique to decode this signal train, would be to constantly monitor the TSOP pin[Digital 15] for its normal state and the moment it produces a low signal, measure the duration of the low signal. If the measured duration of the low signal is around 2ms then measure and store the duration for the next 12 bits of the incoming data. After storing the data, evaluate the duration and based on the duration convert the data to decimal / hexadecimal and use it in your application.

The duration of a signal on an input pin of the arduino can be measured using the pulseIn function. [Read more about this function here](http://arduino.cc/en/Reference/pulseIn)

There is an interesting IR remote library that can help you read different remotes without any difficulty. It can also generate different remote signals. However currently it can generate these signals only on the 3rd pin of the Arduino / InduinoX (PWM pin). Incase you want to use this library to generate remote control signals, we advise that you put the jumpers of the RGB LED OFF and connect a wire between the 3rd pin and the Analog 0 of the InduinoX. The IR LED on the InduinoX is connected to Analog 0(a.k.a digital 14)

You can download the IR remote library and other libraries, sample codes for the InduinoX here -> [**Click Here to Download InduinoX Sample Codes & Required Libraries[Right Click & use Save As]**](http://downloads.simplelabs.co.in/induinox_samples.zip)

**How to use Libraries in Arduino - An Overview**  
To use any library you download, unzip the downloaded file and copy its contents to the libraries folder inside your arduino directory. You can check the library by opening the arduino ide and going to Sketch -> Import Library Option, if your library is in the proper location, it will show up here. Next if there is an example provided with the library (it will be inside a folder called example inside the base folder of the library) it will show up under the libraries name in the File->Examples Menu. You should reopen Arduino for the library to show up.  
  
  
Once you install the IRremote, You can try the example program, IRrecvDemo. This program will give you a serial output of the HEX code for each value corresponding to each button on a remote. We will be using the decimal value in our next program. To get the decimal value, just do the following modification  
  
replace this line  
*~~Serial.println(results.value, HEX);~~*  
with  
*Serial.println(results.value);*  
  
Ensure the TSOP jumper is ON!

Here's a video of a simple project - A remote control interface for our Binary Counter.



Here's the source code for the same

/\*

This sketch increases a 3 bit number every time '+' button is pressed and decreases the value when '-' button is pressed on the remote.It shows the output on 3 LEDs in Binary Format

\*/

#include <IRremote.h>

int RECV\_PIN = 15;

IRrecv irrecv(RECV\_PIN);

decode\_results results;

int i = 0;

void setup()

{

pinMode(11,OUTPUT); // declare LED pins as output pins

pinMode(12,OUTPUT);

pinMode(13,OUTPUT);

pinMode(7,INPUT);// Declare the 7th pin as a input pin. We will use the button on the 7th pin

digitalWrite(7,HIGH);

irrecv.enableIRIn(); // Start the Remote receiver

Serial.begin(9600);

}

void loop()

{

if (irrecv.decode(&results)) {

Serial.println(results.value);

switch(results.value) // if the '+' button is pressed

{

case 2320:

i=0;

break;// 2320 is the value for '0'

case 16:

i=1;

break;// 16 is the value for '1'

case 2064:

i=2;

break;// 2064 is the value for '2'

case 1040:

i=3;

break;// 1040 is the value for '3'

case 3088:

i=4;

break;// 3088 is the value for '4'

case 528:

i=5;

break;// 528 is the value for '5'

case 2576:

i=6;

break;// 2576 is the value for '6'

case 1552:

i=7;

break;// 1552 is the value for '7'

case 1168: // this is the value for the increment button

if(i<7) // if counter value is less than 7 or 3 bits

i++; // increment counter value

else

i=0;

break;

case 3216: // this is the value for the decrement button

if(i>0) // if counter value is greater than 0 or 3 bits

i--; // decrement counter value

else

i=7; // reset counter to 7

break;

}

int a=i%2; // calculate LSB

int b=i/2 %2; // calculate middle bit

int c=i/4 %2; // calculate MSB

digitalWrite(11,c); // write MSB

digitalWrite(12,b); // write middle bit

digitalWrite(13,a); // write LSB

while(digitalRead(7)==0); // wait till button is released to avoid incrementing the counter again

delay(300); // small delay to avoid debounce

irrecv.resume(); // Receive the next value

}

}

**InduinoX User Guide - Working with RTC DS1307**

The DS1307 on the InduinoX is meant to let you explore I2C communication on the Arduino. We shall see how to work with the DS1307 by using a Library[Part 1] and then by using our code based on the Arduino Wire Library [Part 2]

**I2C Communication**

I2C is short form for 'Inter Integrated Circuit' I2C Communication is Communication Bus standard developed by Phillips for standardising Communication between Integrated Circuits. For Eg. In a circuit, there could be a number of ICs each offering specific functionality[RTC, Temperature Sensor, EEPROM, etc] and they can all communicate on a single I2C Bus and provide combined functionalities. Each device on the I2C Bus would have a unique address by which it can be addressed.

Here's an Interesting Introduction from NXP

**I2C on Arduino**

The I2C Bus uses 2 lines for Communication - SDA(Serial Data) & SCL (Serial Clock). On the InduinoX / Arduino, these are available on  SDA (Analog Input 4) & SCL (Analog Input 5). The I2C bus can be accessed using the 'Wire' Library of Arduino. First, lets try out a Library  
  
 **How to use Libraries in Arduino - An Overview**  
To use any library you download, unzip the downloaded file and copy its contents to the libraries folder inside your arduino directory. You can check the library by opening the arduino ide and going to Sketch -> Import Library Option, if your library is in the proper location, it will show up here. Next if there is an example provided with the library (it will be inside a folder called example inside the base folder of the library) it will show up under the libraries name in the File->Examples Menu.  
  
  
**Using the RTC Library**  
There are a number of libraries available to work with the DS1307, the one from [Ladyada](http://ladyada.net/) is our library of choice for this example. You can download it from here  => [RTC Library](https://github.com/adafruit/RTClib/zipball/master)

Unzip the downloaded file and copy its contents to the libraries folder inside your arduino directory. Open the Arduino IDE [close and reopen it if you had it open!], Open File->Examples->RTClib->ds1307  
  
Compile & Upload the program to your InduinoX and open the Serial monitor. Set the baud rate of the Serial monitor to '57600' and voila you see the time!  
  
Now Lets Modify the code and Build a simple project - Timer Controlled LED. The LED on pin 13 will  switch ON at a specific time and switch OFF at another specific Time.  
  
Here's a Video the project in action



Here's the code for the project

/\*

Switches ON and OFF a LED at a Preset Time of Hour based on RTC

\*/

#include <Wire.h>

#include "RTClib.h"

RTC\_DS1307 RTC;

#define ON\_HOUR 13// Hour of the On Time

#define ON\_MIN 0// Minute of the On Time

#define OFF\_HOUR 13// Hour of the Off Time

#define OFF\_MIN 1// Minute of the Off Time

void setup()

{

Serial.begin(9600);

pinMode(13,OUTPUT); // Pin of the LED to be Switched ON / OFF

Wire.begin();

RTC.begin();

RTC.adjust(DateTime("DEC 31 2011","12:59:45")); // Setting the time to a fixed value. If you want to use the system time comment this line and use the option below

// following line sets the RTC to the date & time this sketch was compiled

// RTC.adjust(DateTime(\_\_DATE\_\_, \_\_TIME\_\_)); //uncomment this line to set time to system time

}

void loop()

{

DateTime now = RTC.now();// Getting the current Time and storing it into a DateTime object

if(now.hour()==ON\_HOUR && now.minute()==ON\_MIN)

digitalWrite(13,HIGH);

if(now.hour()==OFF\_HOUR && now.minute()==OFF\_MIN)

digitalWrite(13,LOW);

Serial.print(now.year(), DEC);

Serial.print('/');

Serial.print(now.month(), DEC);

Serial.print('/');

Serial.print(now.day(), DEC);

Serial.print(' ');

Serial.print(now.hour(), DEC);

Serial.print(':');

Serial.print(now.minute(), DEC);

Serial.print(':');

Serial.print(now.second(), DEC);

Serial.println();

delay(1000);

}