

~~maXbox~~



maXbox Starter 35

Start with Web Box Configuration

1.1 A Portable Web Box

Today we go through the steps running a small web server called web box or embedded computing with the internet;

That means you can turn maXbox in a web box.

The second part is about coding ideas, prototyping and new technologies that are in the lab. It's about research papers, and software philosophy, and about researchers worldwide.

So first we need an IP address and the port stored in a ini-file.

Let's jump to the Ini-file. Many applications use ini files to store configuration information. Using ini files has the advantage that they can be used in cross-platform applications and they are easy to read and edit.

This is how we set the web config (and others) in the ini file of maXbox : `maxboxdef.ini`

On subsequent execution of maXbox, the ini values are read in when the form is created and written back out in the `OnClose` and other "in between" events.

You can open the file by the menu `../Help/Config File`
then we jump to the Web section:

[WEB]

IPPORT=8080 //for internal webserver – menu /Options/Add Ons/WebServer2

IPHOST=192.168.1.53

[SSL]

ROOTCERT='filepathY' //for use of HTTPS and certificates...

SCERT='filepathY'

RSAKEY='filepathY'

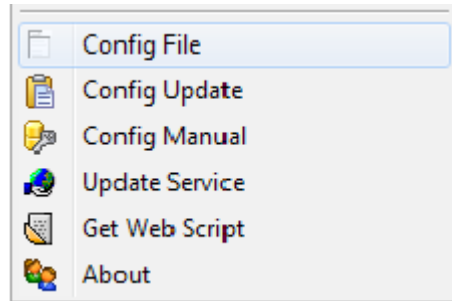
Now you write the **IPHOST** you get by your router or just write 127.0.0.1

If you want to look at your IP you get you can open `../Program/Information`
for example you see:

- Network On COMPort: 5
- Internet On

- Local IP: 192.168.1.53 DNS: 192.168.1.1
- Host Name: MAXBOX8 Win64: True OS:
- User Name: max Is Admin: False

Then you save the configuration and make a `../Help/Config Update`



It's about a web that exist, or at least should exist (or should not!). It's about personal web server marketing, fun (functionality) and new ideas out there.

As more "things" on planet Earth are converted to the inventory set of digitally connected Internet devices, the roles and responsibilities of web developers and technology managers will need to evolve.

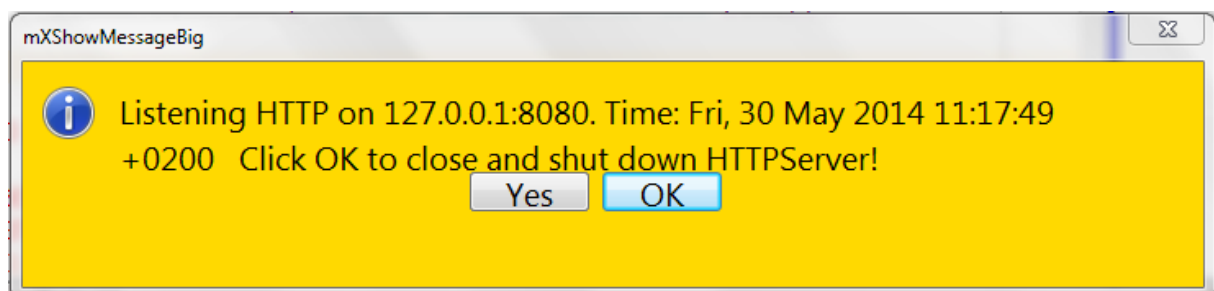
Hope you did already work with the Starter 1 till 29 (especially 5, 17 and the 26) at:

<http://sourceforge.net/apps/mediawiki/maxbox/>

Next we start the web server by

`Options/Add_ons/WebServer2/.`

. Be sure you did `../Help/Config Update` otherwise your changes wont work.



After the start you will see the Listener Box with some config information.
Now you can open a browser and type the URL:

`http://127.0.0.1:8080`

I set the port address to 8080 in case another web server like Apache is running.



The port numbers in the range from 0 to 1023 are the well-known ports or system ports. They are used by system processes that provide widely used types of network services.

The big thing now is to serve and download files from your personal web server like:

http://127.0.0.1:8080/SOA_Delphi.pdf //all files are found in the `../web` directory!

The rest of the magic is done by your browser and java script.
In the maXbox Console you see the protocol of the web server like this:

```
Command GET /SOA_Delphi.pdf at 6/12/2014 10:53:58 AM received from
127.0.0.1:52055
Serving file C:\maXbook\maxbox3\mX3999\maxbox3\web\SOA_Delphi.pdf
(929437 bytes/ 929437 bytes sent) to 127.0.0.1:52055 at 6/12/2014
10:53:59 AM
Command GET /favicon.ico at 6/12/2014 10:54:00 AM received from
127.0.0.1:52056
Serving file C:\maXbook\maxbox3\mX3999\maxbox3\web\index.htm (716
bytes/ 716 bytes sent) to 127.0.0.1:52056 at 6/12/2014 10:54:01 AM
Command GET /favicon.ico at 6/12/2014 10:54:01 AM received from
127.0.0.1:52057
Serving file C:\maXbook\maxbox3\mX3999\maxbox3\web\index.htm (716
bytes/ 716 bytes sent) to 127.0.0.1:52057 at 6/12/2014 10:54:01 AM
HTTP Server closed on: Thu, 12 Jun 2014 11:08:03 +0200.
```

This lesson will introduce you a simple web server to show the code behind and in the end we dive into the world of serial communications and control a lamp from a browser to a web server by sending commands from the PC to the Arduino using the Serial Monitor and Interface.

In our case we explain one example of a HTTP server which is an intermediate to the COM serial communication with the AVR micro controller on Arduino¹.

Another Controller is the Delphi Controller. The Delphi Controller and Delphi DevBoard were designed to help students, Pascal programmers and electronic engineers understand how to program micro controllers and embedded systems especially in programming these devices and targets (see Appendix).

This is achieved by providing hardware (either pre-assembled or as a DIY kit of components), using course material, templates, and a Pascal compatible cross-compiler and using of a standard IDE for development and debugging (Delphi, maXbox, Lazarus or Free Pascal).



Let's begin with HTTP (Hypertext Transfer Protocol) and TCP. TCP/IP stands for Transmission Control Protocol and Internet Protocol. TCP/IP can mean many things, but in most cases, it refers to the network protocol itself.

Each computer on a TCP/IP network has a unique address associated with it, the so called IP-Address. Some computers may have more than one address associated with them. An IP address is a 32-bit number and is usually represented in a dot notation, e.g. 192.168.0.1. Each section represents one byte of the 32-bit address. In maXbox a connection with HTTP represents an Indy object.

¹ An Arduino board consists of an 8-bit Atmel AVR [microcontroller](#), or an ARM cortex on the Due

In our case we will operate with the local host. It is common for computers to refer to themselves with the name local host and the IP number 127.0.0.1.



When HTTP is used on the Internet, browsers like Firefox on Android act as clients and the application that is hosting the website like softwareschule.ch acts as the server.

1.1.1 Get the Code

As you already know the tool is split up into the toolbar across the top, the editor or code part in the centre and the output window at the bottom. Change that in the menu `/view` at our own style.



In maXbox you will **start the web server as a script**, so the web server IS the script that starts the Indy objects, configuration and a browser too on board:

`Options/Add_ons/Easy_Browser/.`



Before this starter code will work you will need to download maXbox from the website. It can be down-loaded from <http://www.softwareschule.ch/maxbox.htm> (you'll find the download maxbox3.zip on the top left of the page). Once the download has finished, unzip the file, making sure that you preserve the folder structure as it is. If you double-click `maXbox3.exe` the box opens a default demo program. Test it with F9 / F2 or press **Compile** and you should hear a sound. So far so good now we'll open the examples:

```
443_webserver_arduino_rgb_light4.txt
102_pas_http_download.txt //if you don't use a browser
```

If you can't find the two files try also the zip-file loaded from:

http://www.softwareschule.ch/download/maxbox_internet.zip or direct as a file
http://www.softwareschule.ch/examples/443_webserver_arduino_rgb_light4.txt

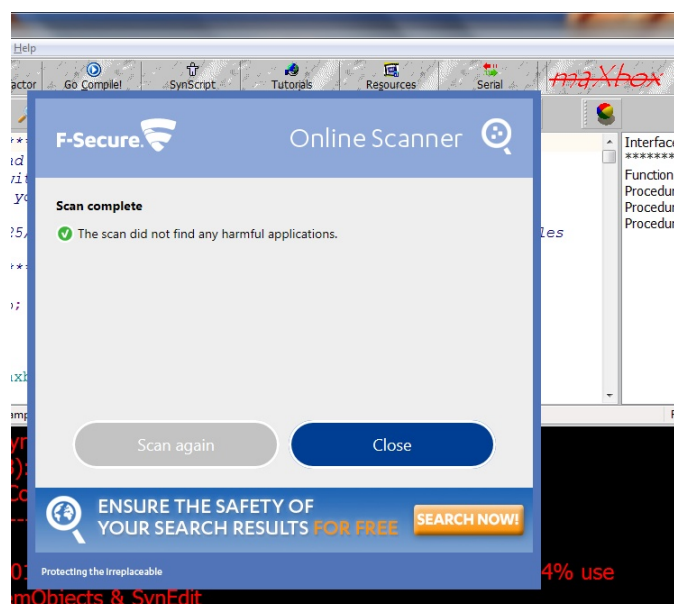
Now let's take a look at the code of this project. Our first line is

```
01 program Motion_HTTPServer_Arduino42_RGB_LED_Light;
```

We have to name the play, means the program's name is above.



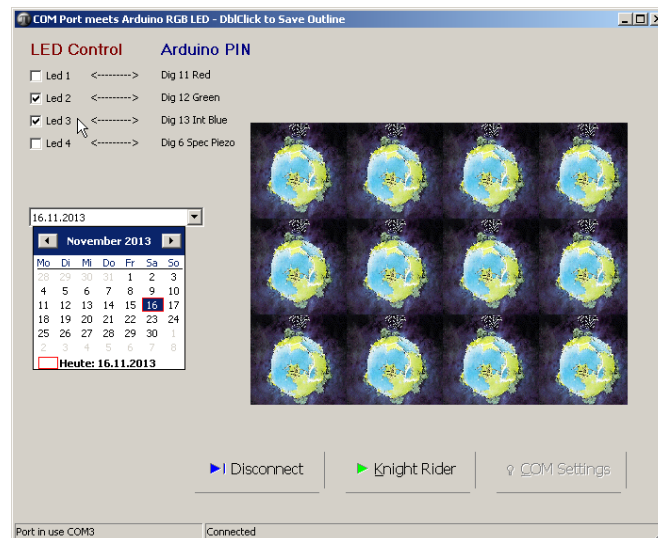
This example requires two objects from the classes: `TIdCustomHTTPServer` and `TComPort` so the second one is a package to connect and transport with the COM Ports to Arduino (see UML below).



It includes 5 components: TComPort, TComDataPacket, TComComboBox, TComRadioGroup and TComLed. With these tools you can build serial communication apps easier and faster than ever. First we start with the web server and second we explain the COM port. After creating the object in line 122 we use the first methods to configure our server calling Port and IP. The object makes a bind connection with the `Active` method by passing a web server configuration.

```
122 HTTPServer:= TIdCustomHTTPServer.Create(self);
```

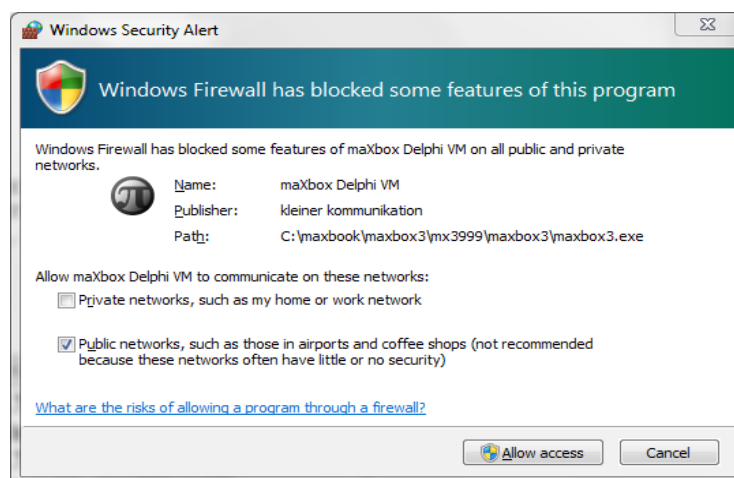
So the object `HTTPServer` has some methods and properties like `Active` you can find in the `TIdCustomHTTPServer.pas` unit or `IdHTTPServer` library. A library is a collection of code or classes, which you can include in your program.



1: The GUI of the Win App

Indy is designed to provide a very high level of abstraction. Much more stuff or intricacies and details of the TCP/IP stack are hidden from the Indy programmer. A Indy session looks like this:

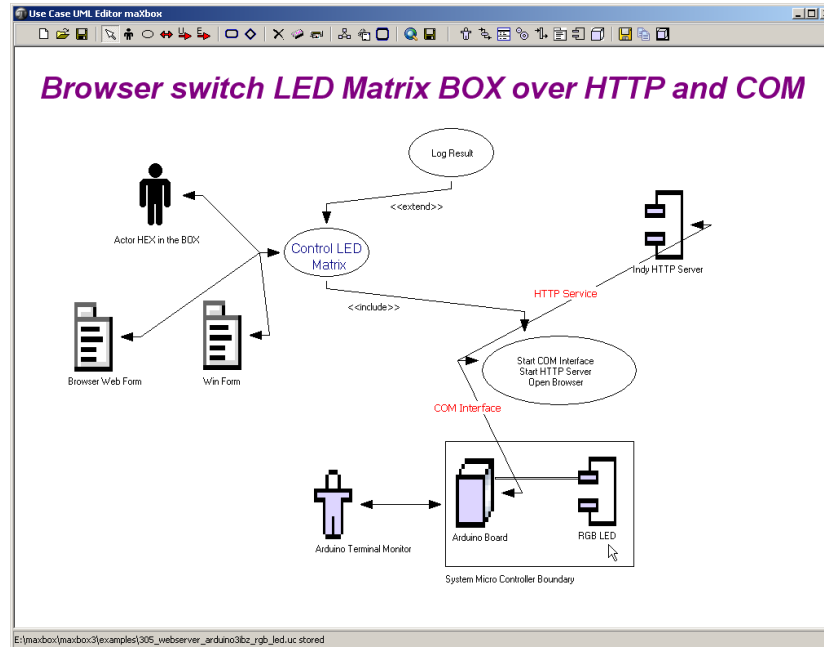
```
with IndyClient do begin
  Host:= 'zip.pbe.com'; // Host to call
  Port:= 6000; // Port to call the server on
  Connect; // get something to do
end;
```



Indy is different than other so called Winsock components you may be familiar with. If you've worked with other components, the best approach for you may be to test Indy. Nearly all other

components use non-blocking (asynchronous) calls and act asynchronously. They require you to respond to events, set up state machines, and often perform wait loops.

👉 In facts there are 2 programming models used in TCP/IP applications. Non blocking means that the application will not be blocked when the app socket read/write data. This is efficient, because your app don't have to wait for connections. Unfortunately, it is complicated.



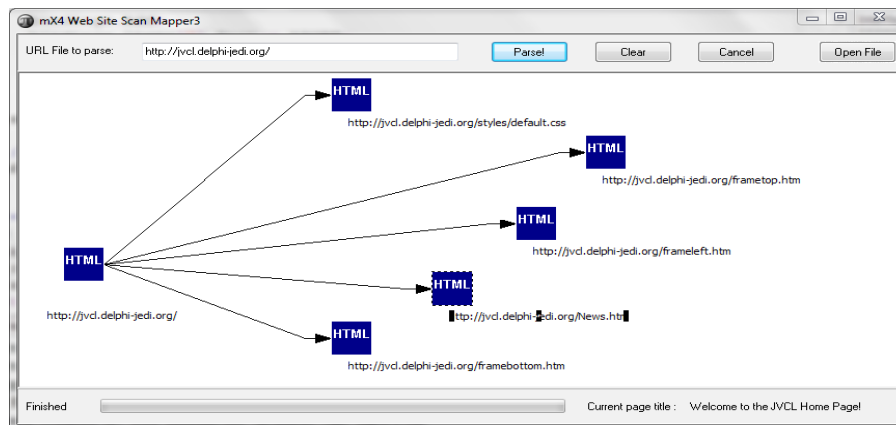
2: The Use Case of the App

Let's get back to our Create in line 125. In line 131 and 132 you see port and IP address config of a `const` in line 14, instead of IP you can set host name as a parameter.

```
126 with HTTPServer do begin
127     if Active then Free;
128     if not Active then begin
129         Bindings.Clear;
130         bindings.Add;
131         bindings.items[0].Port:= APORT;
132         bindings.items[0].IP:= GetHostIP; //IPADDR; '127.0.0.1' or 192.168.1.53'
133         Active:= true;
134         onCommandGet:= @HTTPServerGet;
135         Printf('Listening HTTP on %s:%d.', [Bindings[0].IP, Bindings[0].Port]);
136     end;
```

⚙️ Host Names as GetHostIP function

👉 Host names are "human-readable" names for IP addresses.

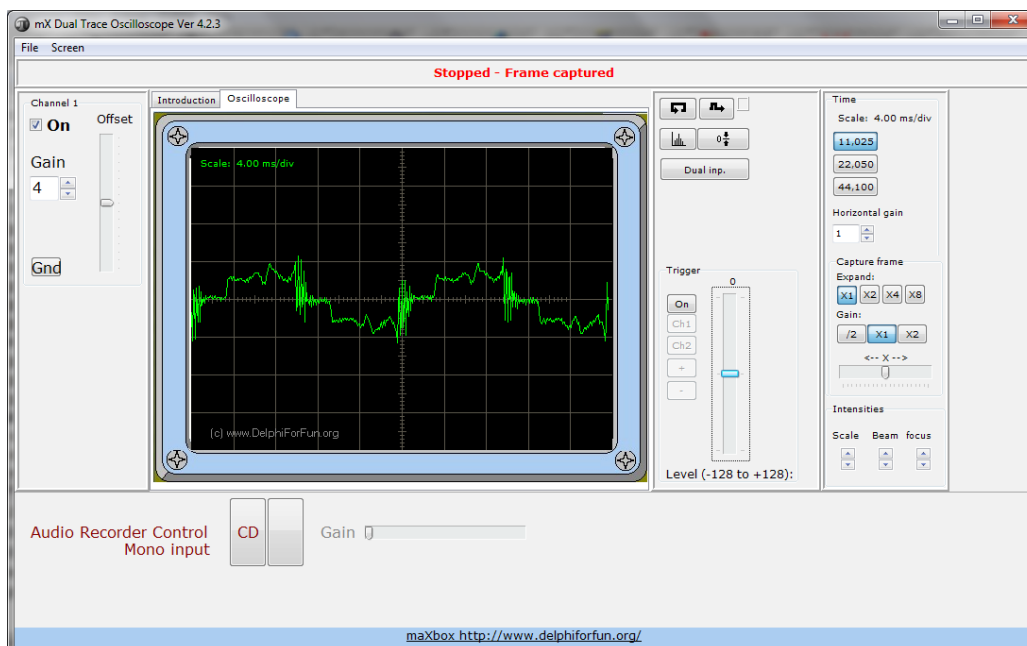


```

1 //maXbox is a scr
2 //runs under Win,
3 //micro is the wo
4 //*****
5
6
7 program Softpedia
8
9 function PlaySound
10 external 'PlaySound'
11
12 const ASIGN = 'CODESIGN with HEX in the BOX is up!';
13       AFILE = 'maxbox3.exe';
14       MP3SONG = 'maxbox.mp3';
15
16 var
17   inFrm: TForm;
18   mp3path: string;
19   i: byte;

```

firstdemo3.txt File loaded
 Welcode: memo2 is output window
 firstdemo3.txt File loaded
 Version on Internet Checked!
 Ver: 3.9.8.9 (398). Work Dir: D:\Desktop\maxbox3



With this oscilloscope you can measure traffic or micro controllers in /Options/..

Host names are used both to make it easier on us humans, and to allow a computer to change its IP address without causing all of its potential clients (callers) to lose track of it.

Often called “URL or links” because a host address is normally inserted at the top of the browser as one of the first items to look at for searching the web.

The full target of the request message is given by the URL property. Usually, this is a URL that can be broken down into the protocol (HTTP), Host (server system), script name (server application), path info (location on the host), and a query.



So far we have learned little about HTTP and host names. Now it's time to run your program at first with F9 (if you haven't done yet) and learn something about GET and HTML. The program (server) generates a standard HTML output or other formats (depending on the MIME type) after downloading with GET or HEAD.

So our command to shine on a LED is ../LED and F5 to switch (127.0.0.1:8080/LED).

Those are GET commands send with the browser, or /R for Red or /G for Green.

The first line identifies the request as a GET. A GET request message asks the Web server application to return the content associated with the URI that follows the word GET.

The following shows the magic behind in the method `HTTPServerGet()`:

```
43 procedure HTTPServerGet(aThr: TIdPeerThread; reqInf: TIdHTTPRequestInfo;  
44                               respInf: TIdHTTPResponseInfo);
```

One word concerning the thread: In the internal architecture there are 2 threads categories. First is a listener thread that “listens” and waits for a connection. So we don't have to worry about threads, the built in thread `TIdPeerThread` will be served by Indy through a parameter:

```
54 if uppercase(localcom) = uppercase('../LED') then begin  
55     cPort.WriteString('1')  
56     writeln(localcom+ ': LED on');  
57     RespInfo.ContentText:= getHTMLContentString('LED is:  ON');  
58 end else  
59 if uppercase(localcom) = uppercase('../DEL') then begin  
60     cPort.WriteString('A');  
61     writeln(localcom+ ': LED off');  
62     RespInfo.ContentText:= getHTMLContentString('LED is:  OFF')  
63 end;
```

HTTP request messages contain many headers that describe information about the client, the target of the request, the way the request should be handled, and any content sent with the request. Each header is identified by a name, such as "Host" followed by a string value.

When an HTML hypertext link is selected (or the user otherwise specifies a URL), the browser collects information about the protocol, the specified domain, path to the information, date and time, the operating environment, the browser itself, and other content information. It then does a request.



You can also switch with F5 in a browser to switch LEDs on and off:

```
69     webswitch:= NOT webswitch;  
70     if webswitch then begin  
71         cPort.WriteString('1') //goes to Arduino for Red  
72         RespInfo.ContentText:= getHTMLContentString('LED is:  ON Switch');  
73     end else begin  
74         cPort.WriteString('A');  
75         RespInfo.ContentText:= getHTMLContentString('LED is:  OFF Switch')
```



```

76     end
77     end

```

One of a practical way to learn much more about actually writing HTML is to get in the maXbox editor and load or open a web-file with extension html. Or you copy the output and paste it in a new maXbox instance. Then you click on the context menu and change to `HTML Syntax`! In this mode the PC is the master and executes the control code while the Arduino or Delphi Controller acts as an interface slave and follows the commands coming from the PC or Browser through its RS232 port. Each RGB field in these records reflects the state of the sensors and actuators of the RGB LED in those sense only actors as LED light are in use. The running Arduino or M485A monitor server will accept read and write commands on the input through the RS232 port as follows:

```

if (val=='1'){
    digitalWrite(ledPin11,HIGH); }
else if (val=='A'){
    digitalWrite(ledPin11,LOW);
}
if (val=='2'){
    digitalWrite(ledPin12,HIGH); }

```



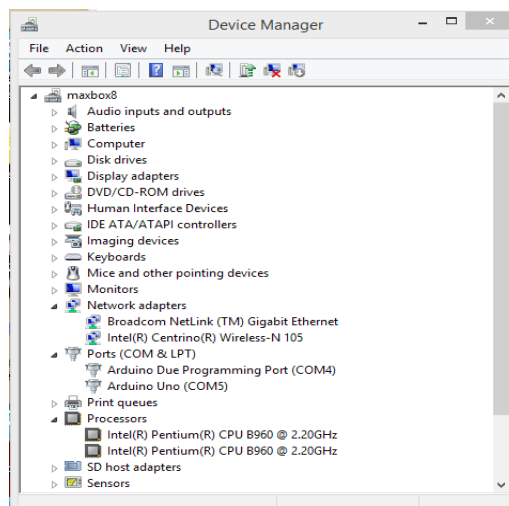
3: A Browser set

When the browser starts from script the server is ready for commands to pass as chars in line 59 or 60 to serial communication. When a the server application finishes with our client request, it lights the LED and constructs a page of HTML code or other MIME content, and passes the result back (via server in `TIidHTTPResponseInfo`) to the client for display.

```


61     writeln(localcom+ ' : LED on');
62     RespInfo.ContentText:= getHTMLContentString('LED is:  ON');

```



Have you tried the program, it's also possible to test the server without Arduino or a browser. The **Compile** button is also used to check that your code is correct, by verifying the syntax before the program starts. Another way to check syntax before run is F2 or the **Syntax Check** in menu Program. When you run this code from the script `102_pas_http_download.txt` you will see a content (first 10 lines) of the site in HTML format with a help of method `memo2.lines.add`:

```
begin
  idHTTP:= TIdHTTP.Create(NIL)
  try
    memo2.lines.text:= idHTTP.Get2('http://127.0.0.1')
    for i:= 1 to 10 do
      memo2.lines.add(IntToStr(i)+' :'+memo2.lines[i])
    finally
      idHTTP.Free
    end
end
```

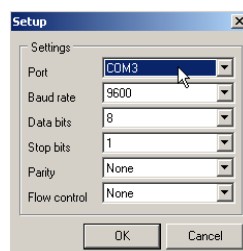
 The Object `TIdHTTP` is a dynamically allocated block of memory whose structure is determined by its class type. Each object has a unique copy of every field defined in the class, but all instances of a class share the same methods. With the method `Get1` you can download files.

```
11 begin
12   myURL:= 'http://www.softwareschule.ch/download/maxbox_examples.zip';
13   zipStream:= TFileStream.Create('myexamples2.zip', fmCreate)
14   idHTTP:= TIdHTTP.Create(NIL)
15   try
16     idHTTP.Get1(myURL, zipStream)
```

Of course a lot of lines to get a file from the web try it shorter with the magic function `wGet()`:

```
wGet('http://www.softwareschule.ch/download/maxbox_starter17.pdf', 'mytestpdf.pdf');
```

It downloads the entire file into memory if the data is compressed (Indy does not support streaming decompression for HTTP yet). Next we come closer to the main event of our web server, it's the event `onCommandGet` with the corresponding event handler method `@HTTPServerGet()` and one object of `TIdPeerThread`. You can use them as server to serve files of many kinds!



4: COM Port Settings

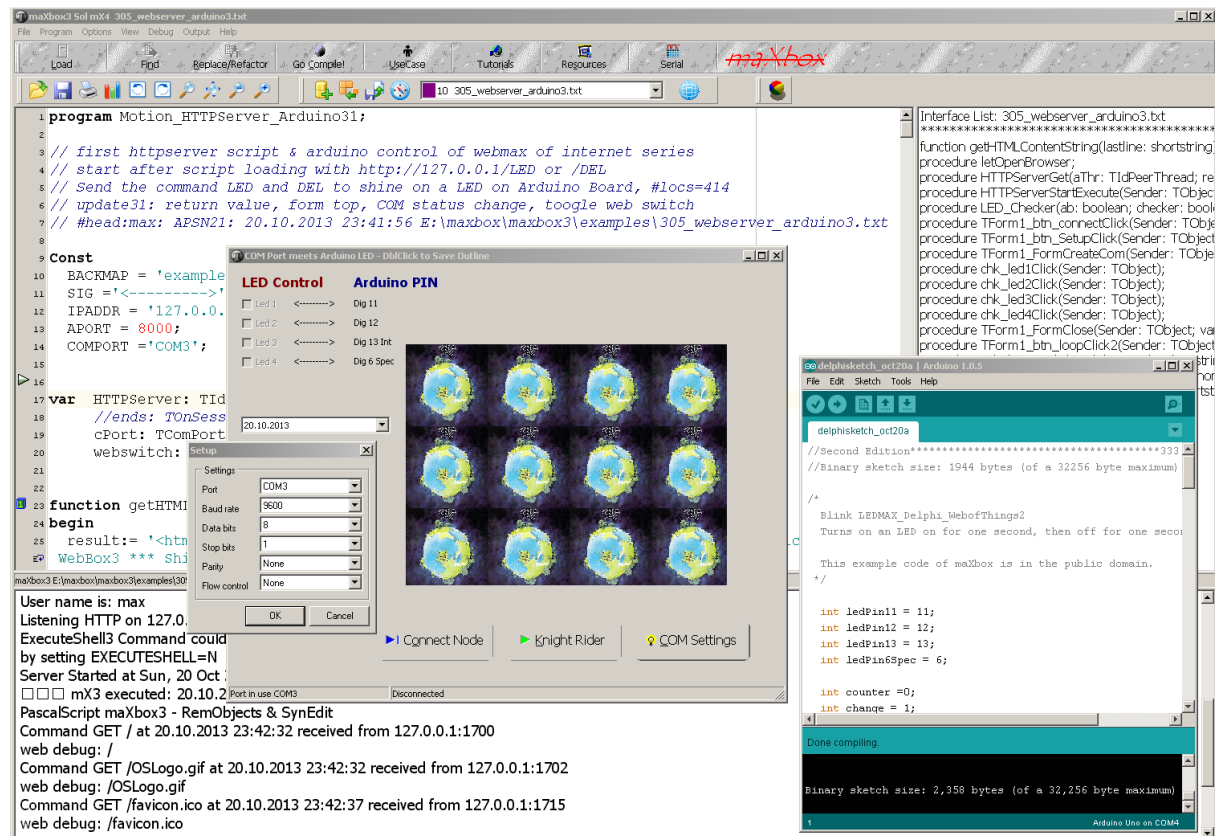
1.1.2 Sockets Check

Close to the end of the first part some knowledge about sockets. Socket connections can be divided into three basic types, which reflect how a connection was initiated and what a local socket is connected to. These are

- Client connections.
- Listening connections.
- Server connections.

Once the connection to a client socket is completed, the server connection is indistinguishable from a client connection. Both end points have the same capabilities and receive the same types of events. Only the listening connection is fundamentally different, as it has only a single endpoint.

Sockets provide an interface between your network server or client application and a networking software. You must provide an interface between your application and clients that use it.



6: the BOX, Arduino and the GUI

Sockets let your network application communicate with other systems over the network. Each socket can be viewed as an endpoint in a network connection. It has an address that specifies:

- The system on which it is running.
- The types of interfaces it understands.
- The port it is using for the connection.

A full description of a socket connection includes the addresses of the sockets on both ends of the connection. You can describe the address of each socket endpoint by supplying both the IP address or host and the port number.

Many of the protocols that control activity on the Internet are defined in Request for Comment (RFC) documents that are created, updated, and maintained by the Internet Engineering Task Force (IETF), the protocol engineering and development arm of the Internet. There are several important RFCs that you will find useful when writing Internet applications:

- RFC822, "Standard for the format of ARPA Internet text messages," describes the structure and content of message headers.

- RFC1521, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies," describes the method used to encapsulate and transport multipart and multiformat messages.
- RFC1945, "Hypertext Transfer Protocol—HTTP/1.0," describes a transfer mechanism used to distribute collaborative hypermedia documents.

In the next line we just start a browser to test our server in a so called frame work flow ☺

```
34 procedure letOpenBrowser;
35 // TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
36 begin
37 //ShellAPI.ShellExecute(Handle,PChar('open'),'http://127.0.0.1:80/',Nil,Nil,0);
38 S_ShellExecute('http:'+IPADDR+':'+IntToStr(APORT)+'/', '', seCmdOpen)
39 end;
```



Try to change the IP address in line 132 of `IP:= IPADDR` with a DHCP or dynDNS address, so you can reach Arduino from an Android, but change also the const in line 13.



Try to get data back from the new function `getWebScript()`:

```
getScriptandRun('http://www.softwareschule.ch/examples/demoscript.txt');
//getScriptandRunAsk;
https://github.com/maxkleiner/maXbox3/blob/masterbox2/examples/476\_getscripttest.txt
```



1.2 Android with Arduino

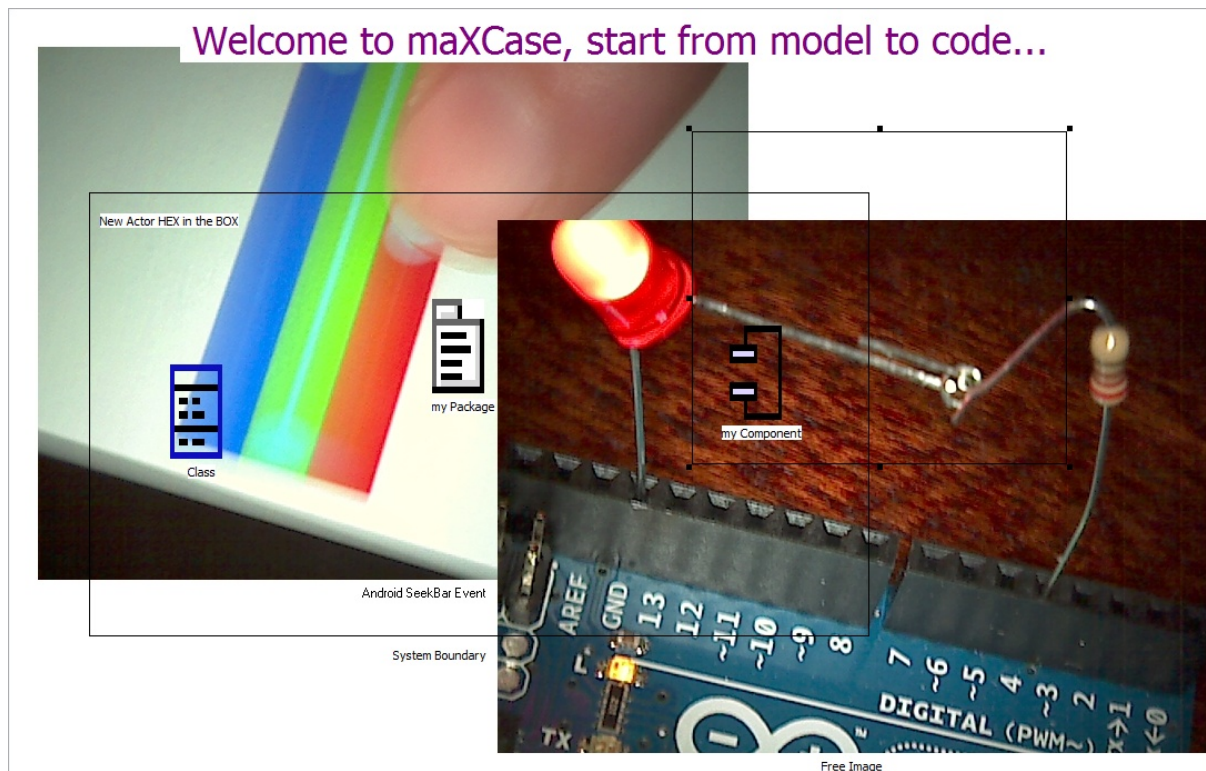
Now at last the dream team with this USB host (I did a lot with my students).

The Arduino ADK is a combination of hardware and software designed to help people interested in designing accessories for Android devices.

During Google's 2011 IO keynote, Google introduced a board based on the Arduino Mega 2560 which includes this USB host. A new USB library [3] was introduced, enabling data to be sent and received from any external devices. This library is included with the latest version of the Android OS. Any devices with OS 3.1 and later can use the USB port to connect to accessories.

<http://labs.arduino.cc/ADK/Index>

App makers can freely use it in their development. This library has been included in 2.3.4, and some devices have been reported to work with 2.3.3. I work with Andro 4.2.2 and Arduino 1.5.



8: A seek bar on Android dims the Red LED on Arduino

The Mega ADK board is a derivative of the Arduino Mega 2560. The modified Mega 2560 board includes a USB host chip. This host chip allows any USB device to connect to the Arduino.

The USB host is not part of the original core of Arduino. To use the new features on this board you will need to include some libraries in your sketches.

There are three libraries needed to make the system RGB SeekBar work:

- MAX3421e: handles the USB host chip
- USB: handles the USB communication
- Android Accessory: checks if the device connecting is one of the available accessory-enabled phones

http://www.softwareschule.ch/examples/B004_RGB_Led.ino.htm

```
#include "variant.h"
#include <stdio.h>
#include <adk.h>
#include <Scheduler.h>

#define LED 13
#define RED 2
#define GREEN 3
#define BLUE 4

/* Aware: Serial has to be right below on newline! */
// Accessory descriptor. It's how Arduino identifies itself to Android.
char model[] = "HelloWorldModel"; // model in xml
char displayName[] = "A Hello World Accessory"; // your Arduino board
char companyName[] = "Hello maXbox Inc";
```

```
// Make up anything you want for these
char versionNumber[] = "1.2";
char serialNumber[] = "1";
char url[] = "http://www.osciptime.com/repo/OsciPrimeICS.apk";

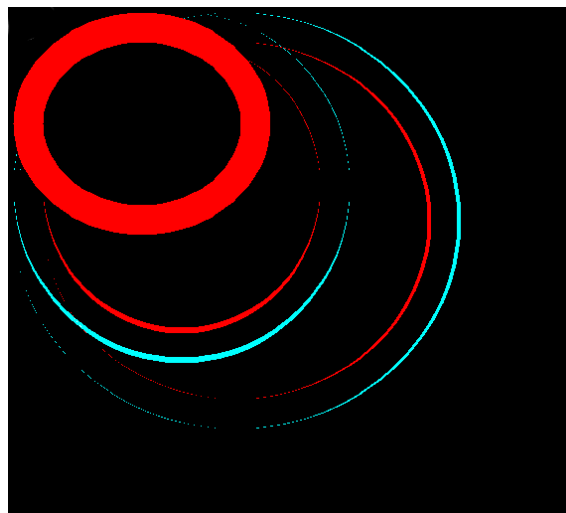
volatile uint8_t pause = 255;
USBHost Usb;
ADK adk(&Usb, companyName, model,
displayName,versionNumber,url,serialNumber);
```

Arduino is an amazing device and will enable you to make anything from interactive works of art to robots. With a little enthusiasm to learn how to program the Arduino and make it interact with other components as well as a bit of imagination, you can build anything you want.

Arduino can also be extended with the use of Shields which circuit boards are containing other devices (e.g. GPS receivers, LED Cubes, LCD Displays, Drones, MIDI Synthesizers, Ethernet connections, etc.) that you can simply slot into the top of your Arduino to get extra functionality.

Arduino is an open-source single-board micro-controller, descendant of the open-source Wiring platform designed to make the process of using electronics in multitude projects more accessible. The Arduino can be connected to Dot matrix displays, glasses, switches, motors, temperature sensors, pressure sensors, distance sensors, web-cams, printers, GPS receivers, Ethernet modules and so on.

The Arduino board is made of an Atmel AVR microprocessor, a crystal or oscillator (basically a crude clock that sends time pulses to the micro-controller to enable it to operate at the correct what type of Arduino you have, you may also have a USB connector to enable it to be connected to a PC or Linux to upload or retrieve data. The board exposes the micro-controllers I/O (Input/Output) pins to enable you to connect those pins to other circuits, buses or sensors.



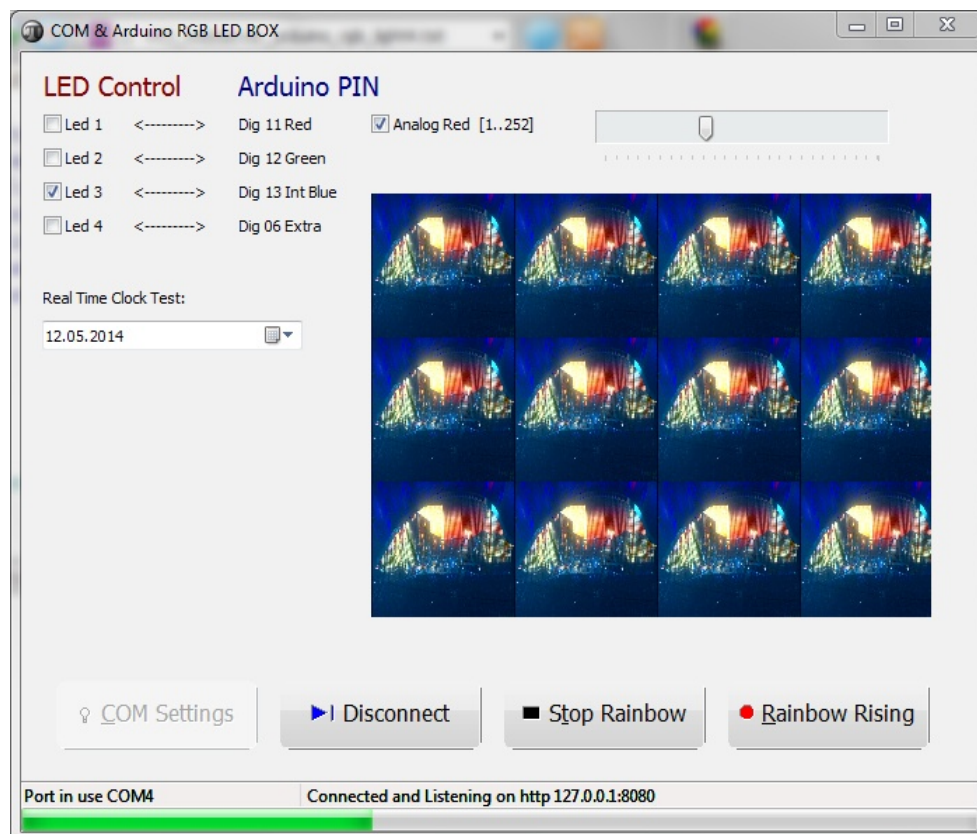
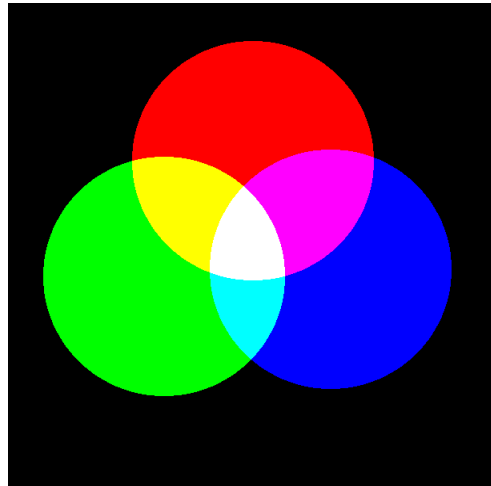
Feedback @
max@kleiner.com

Literature:
 Kleiner et al., Patterns konkret, 2003, Software & Support
 Links of maXbox, Web of Things, Arduino and Indy:

<http://www.softwareschule.ch/download/webofthings2013.pdf>
http://www.softwareschule.ch/download/Arduino_C_2014_6_basta_box.pdf

<http://www.softwareschule.ch/maxbox.htm>
<http://www.indyproject.org/Socket/index.EN.aspx>
http://www.softwareschule.ch/examples/443_webserver_arduino_rgb_light4.htm
<http://en.wikipedia.org/wiki/Arduino>
<http://fritzing.org/>

<http://sourceforge.net/projects/maxbox>
<http://sourceforge.net/apps/mediawiki/maxbox/>
<http://sourceforge.net/projects/delphiwebstart>
http://www.blaisepascal.eu/subscribers/vogelaar_elctronics_info_english.php



[maXbox Arduino Framework](#)

1.3 Appendix Arduino LED Rainbow

```
procedure TF_Rainbowloop(Sender: TObject);
var mtime, multiple: integer;
begin
    LED_Cheker(false, true);
    {Color Spectrum from Red to White code (r,y,g,c,b,m,w...}
    mtime:= 500; //1000;
    multiple:= 2;
    statBar.Panels[1].Text:='Rainbow - Click LED4 to end loop!';
    try
        with cPort do begin //using
            repeat
                //WriteStr('1'); Sleep(mtime);
                digitalWrite(red, AHIGH); // red
                delay(mtime);
                digitalWrite(green, AHIGH); // yellow
                delay(mtime);
                digitalWrite(red, ALOW); // green
                delay(mtime);
                digitalWrite(blue, AHIGH); // cyan
                delay(mtime);
                digitalWrite(green, ALOW); // blue
                delay(mtime);
                digitalWrite(red, AHIGH); // magenta
                delay(mtime);
                digitalWrite(green, AHIGH); // white
                mtime:= mtime * multiple;
                delay(mtime);
                digitalWrite(blue, ALOW); // reset
                digitalWrite(green, ALOW);
                digitalWrite(red, ALOW);
                mtime:= mtime div multiple; //time/=multiple;
            until chk_led4.Checked=true;
            chk_led4.Checked:= false;
        end;
    except
        Showmessage(R_EXCEPTMESS);
    end;
end;
```

1.4 Appendix Arduino Base Code

```
//*****Arduino Code*****
* Turns on and off 5 light emitting diodes (LED) connected to digital pins 2 to 6. The LEDs will be
controlled using check boxes on maXbox that sends serial data to Arduino. */
int val = 0; // variable to store the data from the serial port
int ledPin1 = 2; // LED connected to digital pin 2
int ledPin2 = 3; // LED connected to digital pin 3
int ledPin3 = 4; // LED connected to digital pin 4
```

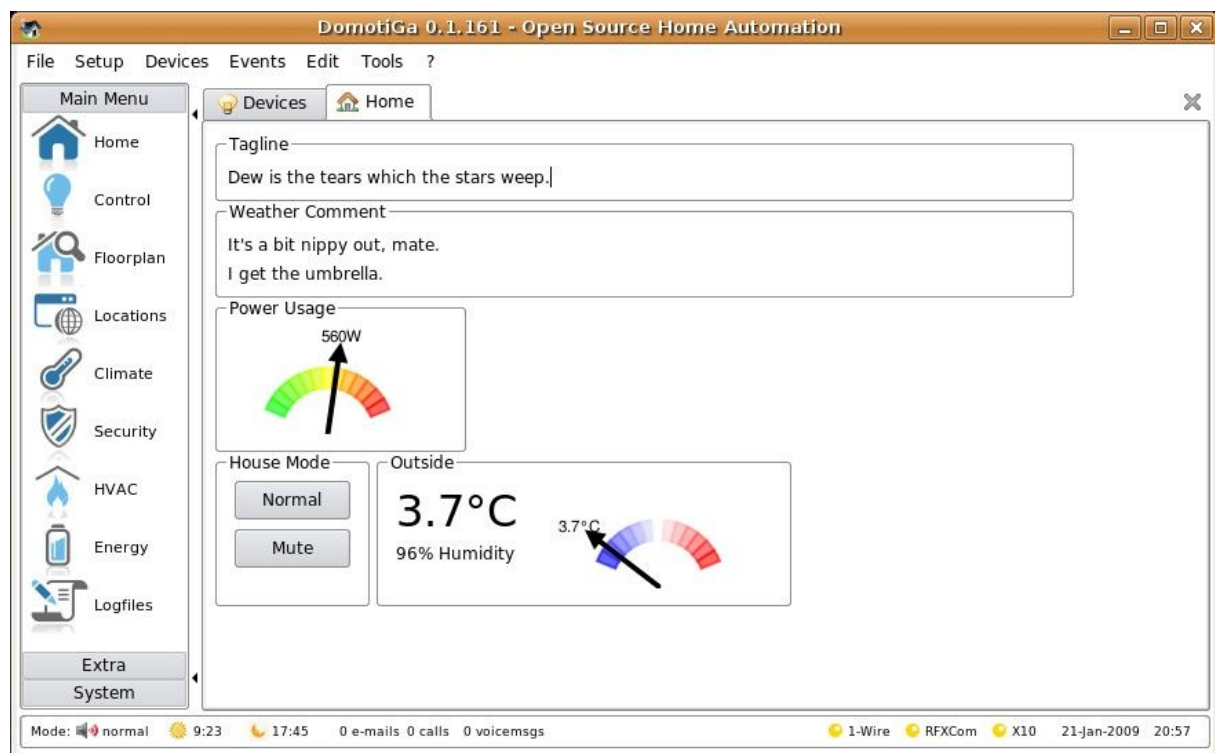
```

void setup() {
  pinMode(ledPin1,OUTPUT); // declare the LED's pin as output
  pinMode(ledPin2,OUTPUT); // declare the LED's pin as output
  pinMode(ledPin3,OUTPUT); // declare the LED's pin as output
  Serial.begin(9600);      // connect to the serial port
}

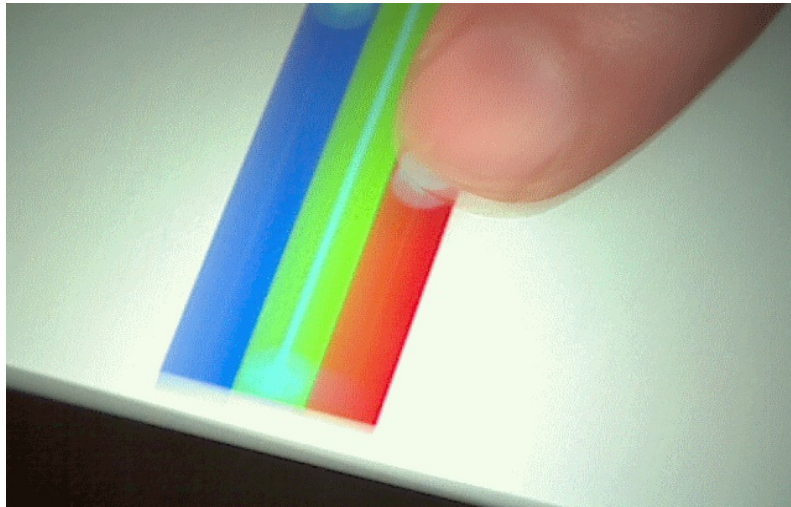
void loop () {
  val = Serial.read();    // read the serial port
  if (val !=-1){
    if (val=='1'){
      digitalWrite(ledPin1,HIGH);
    }
    else if (val=='A'){
      digitalWrite(ledPin1,LOW);    }
    if (val=='2'){
      digitalWrite(ledPin2,HIGH);    }
    else if (val=='B'){
      digitalWrite(ledPin2,LOW);    }
    if (val=='3'){
      digitalWrite(ledPin3,HIGH);    }
    else if (val=='C'){
      digitalWrite(ledPin3,LOW);    }
    //Serial.println();
  }
}

```

1.5 Appendix Arduino App

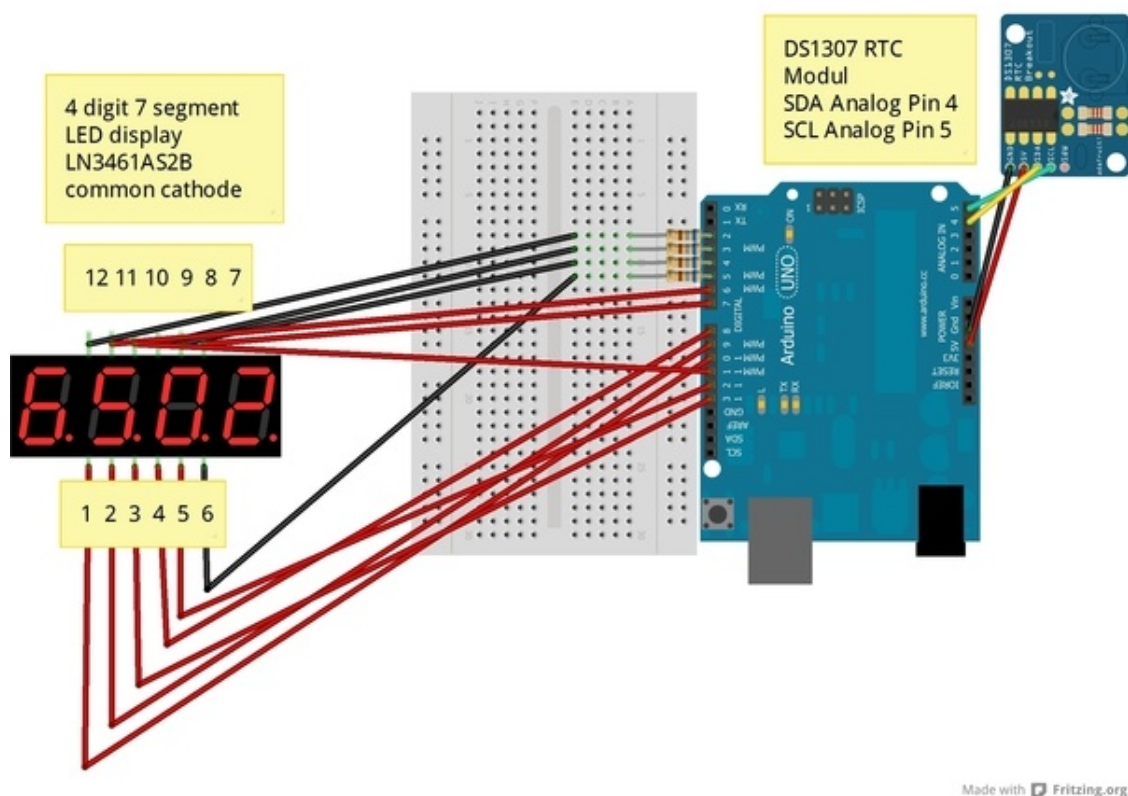


1.6 Android maXbox



9: maXbox Web Cam live in Android on Arduino

1.7 RTC Layout



RTClock: ,Arduino by Silvia Rothen

<http://www.ecotronics.ch/ecotron/arduinocheatsheet.htm>