



Application Note

BRT_AN_032

FT90X Ethernet LED Control

Version 1.0

Issue Date: 2018-03-27

In an increasingly connected world, more and more devices are going online. To enable online connectivity typically requires an MCU with Ethernet or Wi-Fi capabilities. To demonstrate the principle, this Application Note describes a network device which allows control of 2 WS2812 RGB addressable LEDs. The FT90X device provides an HTTP interface with web-page control over the colour and brightness of 2 LED devices. An UPnP device is implemented to facilitate discovery.

Use of Bridgetek devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify and hold Bridgetek harmless from any and all damages, claims, suits or expense resulting from such use.

Bridgetek Pte Ltd (BRTChip)

178 Paya Lebar Road, #07-03, Singapore 409030

Tel: +65 6547 4827 Fax: +65 6841 6071

Web Site: <http://www.brtchip.com>

Copyright © Bridgetek Pte Ltd

Table of Contents

1 Introduction	4
1.1 Overview	5
1.2 Scope	5
1.2.1 Features	5
1.2.2 Possible Enhancements.....	5
2 Project Overview	6
2.1 Sources Folder.....	6
2.2 Iwip Folder	6
2.3 upnp Folder	6
2.4 ws28xx Folder	7
2.1 Tools Folder.....	7
2.2 httpdfs Folder.....	7
3 Software Implementation.....	9
3.1 HTTPD Server-Side-Includes	9
3.2 HTTPD CGI Handling.....	10
3.3 UPnP Device Callbacks	10
4 Operation	12
4.1 Discovery	12
4.2 Colour Picker Page	12
4.3 Slider Page	13
4.4 Network Page.....	13
5 Importing into the FT9XX Toolchain	15
6 Contact Information	16
Appendix A– References	17
Document References	17
Acronyms and Abbreviations.....	17
Appendix B – List of Tables & Figures	18

List of Tables.....	18
List of Figures	18
Appendix C– Revision History	19

1 Introduction

In an increasingly connected world, more and more devices are going online. To enable online connectivity typically requires an MCU with Ethernet or Wi-Fi capabilities. To demonstrate the principle, this Application Note describes an implementation of a web server which allows for control of 2 WS2812 serial addressable LEDs. A web server is implemented on an FT90X device which when connected to a Local Area Network (LAN) allows a web browser to control the LEDs on an MM900EVxA board from a graphical web page.

The LEDs have individual settings for Red, Green and Blue segments so can show a full range of colours. Two methods of changing the colours are given: a simple slider for each of red, green and blue; or a colour map where the desired colour can be selected by clicking and intensity changed with a slider control.

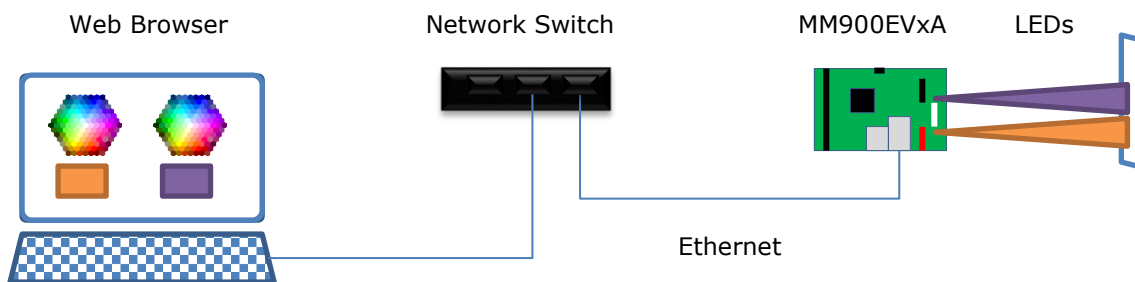


Figure 1 Block Diagram

The web server code running on the FT90X on the MM900EVxA board will serve a web page with a method of setting the colour displayed by the 2 LEDs on the board. The browser will send back an HTTP GET request with LED intensity levels which are sent to the LEDs via a GPIO pin.

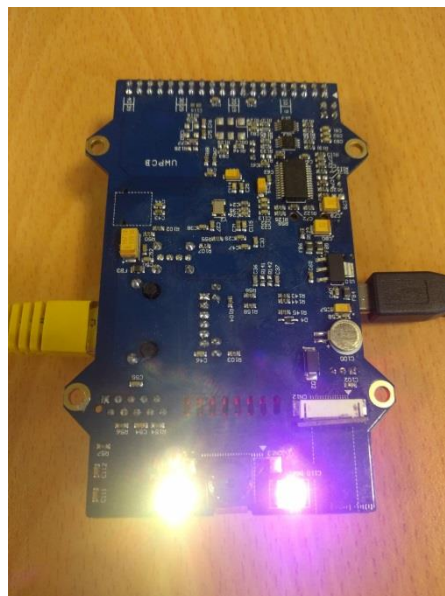


Figure 2 LEDs on Yellow and Fuschia

UPnP is implemented on the device as a method of discovering the device on a Local Area Network. This allows for the device to be listed under "Other Devices" within the "Network" section of Windows Explorer and similar programs.

The document should be read in association with the example code provided in the references section.

1.1 Overview

This document describes the design and implementation of the FT90X Ethernet LED Control code. The FT90X Ethernet LED Control allows users to:

- Implement a web server on an FT90X device.
- Connect the device to a Local Area Network.
- Interact with the web server to control 2 LEDs on the MM900EVxA.
- Update networking settings stored in non-volatile memory on the MM900EVxA.

This document is intended to demonstrate the capabilities of the FT90X family of microcontrollers by utilising the Ethernet port and GPIO.

Third-party open source code is used to implement this application note:

- Printf – tinyprintf.
- lwIP –Light Weight Internet Protocol

Links to resources for these libraries are in Appendix A – References.

1.2 Scope

The application and lwIP implementation are limited to Local Area Networking only and to the HTTP protocol. UPnP discovery implements a minimum device only.

1.2.1 Features

The application note shows a simple implementation of the lwIP HTTPD application and architecture port of lwIP for the FT90X device. HTTP GET requests are supported as are server-side includes.

The file system code for lwIP is modified to load files for the web server from program memory rather the data segment of RAM.

1.2.2 Possible Enhancements

This application note can be seen as a start for customisation or extension. Some example enhancements could be:

- Support for more LEDs or LED strips.
- More options for animation or fine grained control of LEDs.
- HTTPS requests and responses.

2 Project Overview

The project files for the application are divided into the following folders.

Folder	Description
Source	Application source code and abstraction files.
Includes	Application specific header files.
httpdfs	Files to be compiled into lwIP HTTPD application file system.
lib	Library files.
lib\ws28xx	WS2812 support files.
lib\lwip	lwIP library.
lib\upnp	Bridgetek UPnP library for lwIP.
lib\tinyprintf	tinyprintf library.
Tools	Source code for makefsdata tool.

Table 1 - Project Files Overview

2.1 Sources Folder

The main part of the application is found in the "Sources" folder. This is split into two sections and has 2 source code files.

- The "main.c" file is generally responsible for setting up the FT90X device and handling application specific callbacks for the lwIP library. New settings for the LEDs are received in these callbacks and then sent to the LEDs.
- The second file "net.c" implements an interface between the lwIP library and the main code.

The other file in this folder is:

- "crt0.S" a modified startup file (in FT9XX assembly language) to allow the application to write to a protected section of FlashROM on the device.

Files in these folders uses the "Includes" folder for application specific header files.

2.2 lwip Folder

This folder holds the lwIP library which handles networking for the application note. The code has been ported to FT90X and supports reading the file system from program memory rather than RAM.

2.3 upnp Folder

The UPnP library adds automatic discovery to the lwIP networking library. This allows devices to be found by UPnP Control Points on the Local Area Network. The lwIP library contains an implementation of an HTTPD server which is required by the UPnP library.

The library supports both UPnP Device or UPnP Control Point profiles and all configuration data is provided by the user application via callbacks.

2.4 ws28xx Folder

A small driver file for the WS28xx series of devices is included. It allows clocking out an arbitrary length array of data to a chain of WS28xx devices.

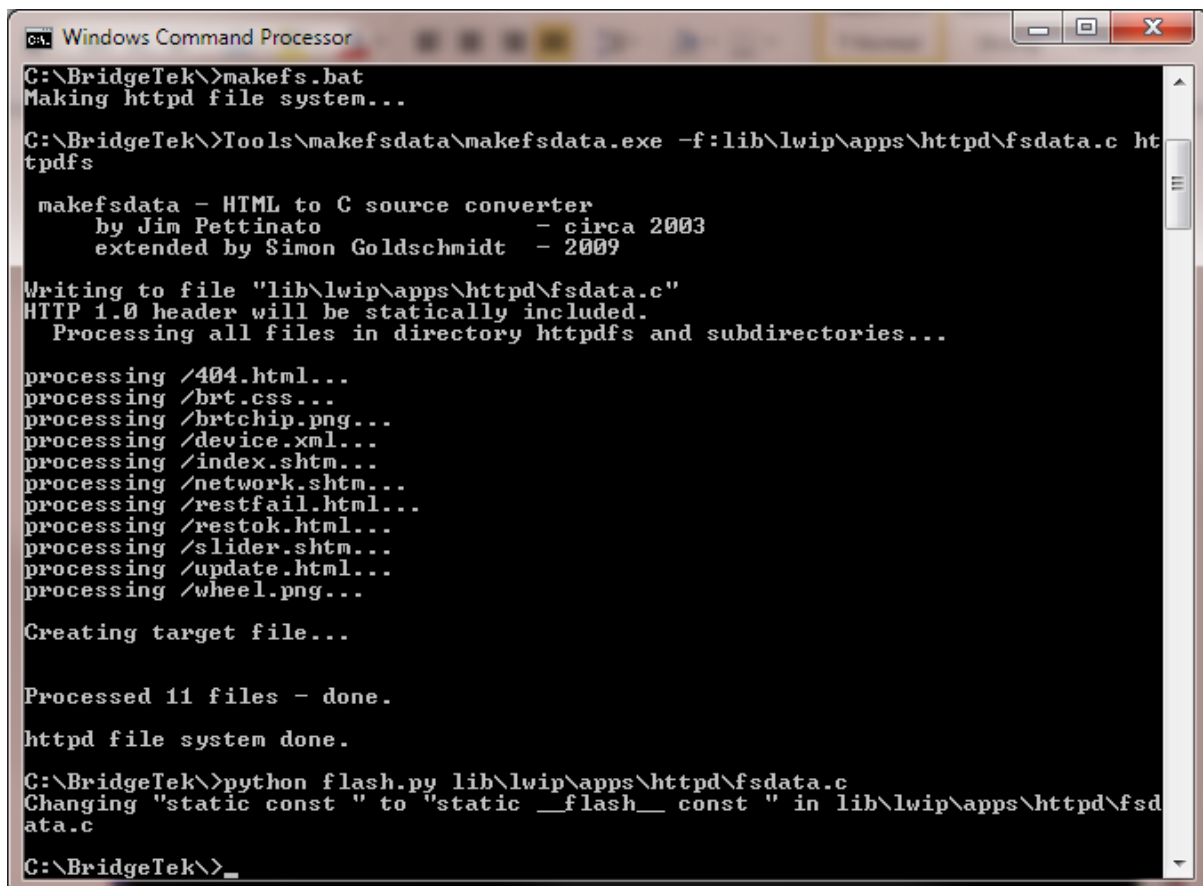
2.1 Tools Folder

This contains source code and a Windows executable for the "makefsdata" utility.

2.2 httpdfs Folder

The lwIP HTTPD application uses a virtual file system compiled into the application ROM image to allow it to serve web pages. The files used in the file system are in this folder.

To compile these into a format ready for the HTTPD application the "makefsdata" tool is used. This is supplied with the lwIP HTTPD application. The standard declaration of the data in the output file "fsdata.c" is "**static const**". This results in the data being placed in the RAM (data section). Preferable on an embedded system is that it should reside in Flash until required. A python script "flash.py" resides in the top level directory of the application note and converts necessary declarations into "**static __flash__ const**" so that file system data is not copied into RAM.



```
C:\BridgeTek>makefs.bat
Making httpd file system...

C:\BridgeTek>Tools\makefsdata\makefsdata.exe -f:lib\lwip\apps\httpd\fsdata.c httpdfs

makefsdata - HTML to C source converter
  by Jim Pettinato           - circa 2003
  extended by Simon Goldschmidt - 2009

Writing to file "lib\lwip\apps\httpd\fsdata.c"
HTTP 1.0 header will be statically included.
  Processing all files in directory httpdfs and subdirectories...

processing /404.html...
processing /brt.css...
processing /brtchip.png...
processing /device.xml...
processing /index.shtm...
processing /network.shtm...
processing /restfail.html...
processing /restok.html...
processing /slider.shtm...
processing /update.html...
processing /wheel.png...

Creating target file...

Processed 11 files - done.

httpd file system done.

C:\BridgeTek>python flash.py lib\lwip\apps\httpd\fsdata.c
Changing "static const " to "static __flash__ const " in lib\lwip\apps\httpd\fsdata.c

C:\BridgeTek>_
```

Figure 3 Running the makefs.bat script

The resulting C source code file is placed in the "lib\lwip\apps\httpd" folder as "fsdata.c". It is not compiled directly but included (as source code) by the file "fs.c". Modifying the "fsdata.c" file is recognised by the Eclipse build environment and the relevant files are rebuilt.

3 Software Implementation

The lwIP HTTPD application provides the main part of the application note code. This resides in the lwIP library and callback functions are found in "main.c" to provide the desired functionality.

The HTTPD application reads static and server-side-include files from the virtual file system. Server-side-include files are processed in the HTTPD application and include data added by an application callback function.

Form results using the GET method are passed to the application note code for processing. Within this processing the desired LED settings are received and the WS28xx devices updated.

The UPnP library separately responds to SSDP protocol requests and sends appropriate broadcast messages to support UPnP discovery. The responses are made up from information garnered from callbacks to application callback functions. The HTTPD server is required to have an XML configuration file within its file system containing UPnP data.

3.1 HTTPD Server-Side-Includes

Within the HTTPD application the server-side-include (SSI) code is enabled by defining the label LWIP_HTTPD_SSI in "lwipopts.h". The label LWIP_HTTPD_SSI_INCLUDE_TAG is set to zero to ensure that the tag names do not appear within the output of the SSI handler. This ensures that any values added by the application can appear within HTML tags, scripts or values.

The function which is called when an SSI appears in an html file, is configured by the following line in the led_bridge function:

```
http_set_ssi_handler(app_httpd_ssi_cb, ssiTags, ssiTag_total);
```

The callback function is set as app_httpd_ssi_cb and a list of strings containing valid tags (as an array of char *) is passed along with the number of entries in the list.

When the callback function is reached lwIP passes the index of the matched SSI tag in the array ssiTags, along with a pointer to receive the included data and the maximum length in chars of the included data.

```
u16_t app_httpd_ssi_cb(int iIndex, char *pcInsert, int iInsertLen)
```

When processing the received tags, the result is copied into the location pointed to by pcInsert as a NULL-terminated string. The sprintf or strncpy functions are used to ensure that the size of the result is no longer than iInsertLen. The actual length of the data written is returned by the function.

The supported SSI tags are populated in the array ssiTags using macros to both create a NULL-terminated string and populate an enum (enumeration) with the index of the tags. C pre-processor macros are used to make this automatic and reduce the possibility of mismatches between the contents of the array and the index values.

There are several network related SSI tags and a set of tags to report the current state of the LEDs.

```
<!--#mac--> - MAC Address of device  
<!--#dhcp--> - Returns "enabled" or "disabled". Used for text.  
<!--#dhcp_en--> - Returns "" if DHCP is enabled or "disabled". Used for forms.  
<!--#dhcp_ch--> - Returns "checked" if DHCP is enabled or "". Used for forms.  
<!--#ip--> - Returns the IP address assigned to the device.
```

<!--#gw--> - Returns the IP gateway address assigned to the device.
<!--#nm--> - Returns the network mask assigned to the device.

<!--#led1_r--> - Returns the decimal intensity of LED1 Red component.
<!--#led1_g--> - Returns the decimal intensity of LED1 Green component.
<!--#led1_b--> - Returns the decimal intensity of LED1 Blue component.
<!--#led2_r--> - Returns the decimal intensity of LED2 Red component.
<!--#led2_g--> - Returns the decimal intensity of LED2 Green component.
<!--#led2_b--> - Returns the decimal intensity of LED2 Blue component.

The included text from these can be used in text, forms or JavaScript in SSI pages.

3.2 HTTPD CGI Handling

To enable CGI support in lwIP, define the label LWIP_HTTPD_CGI in lwipopts.h.

There are 3 handlers added to the array of CGI callback functions:

```
const tCGI app_httpd_cgi_cb [] = {  
    {"/net", app_httpd_cgi_net},  
    {"/leds", app_httpd_cgi_leds_get},  
    {"/pick", app_httpd_cgi_colour_get},  
};
```

The handlers are set in lwIP by this call in the led_bridge function:

```
http_set_cgi_handlers(app_httpd_cgi_cb, sizeof(app_httpd_cgi_cb)/sizeof(tCGI));
```

These will allow forms to be submitted via the HTTP GET method to the device. Sample URLs will be of the format:

```
http://x.x.x.x/net?ip=192.168.1.35?gw=192.168.1.1
```

```
http://x.x.x.x/leds?led1r=255?led1g=0?led1b=255
```

```
http://x.x.x.x/pick?led1=ffaa22
```

The callback functions are entered each time the GET is received and are given an array of parameters parsed from the URL to process. Once all the parameters are processed (unknown or invalid parameters are ignored) the actions required are performed. For "leds" and "pick" methods the LEDs are updated, for "net" methods the non-volatile storage of network parameters are changed and the device is restarted.

3.3 UPnP Device Callbacks

To enable UPnP Device support, define the label UPNP_ENABLE_DEVICE in upnp_opts.h.

The UPnP Device code requires certain information from the application to describe itself in adverts and as response to searches. There are two handler callback functions required.

- 1) For responding to M-SEARCH requests. The Search Type from the SSDP request is sent along with an action code. The function can decide to respond to the request if the Search Type matches some application-specific criteria and the data requested by the action code are correct. If it does not wish to respond then the M-SEARCH will be ignored.
- 2) For NOTIFY requests. When a NOTIFY is scheduled to be sent, the callback will ask for information for a particular device. This allows multiple UPnP devices to be implemented.

The handlers are set in the led_bridge function:

```
upnp_set_dev_handler(app_upnp_dev_msghdr_cb, app_upnp_dev_nothdr_cb, 1);
```

This implements a single UPnP device with search and notify functions.

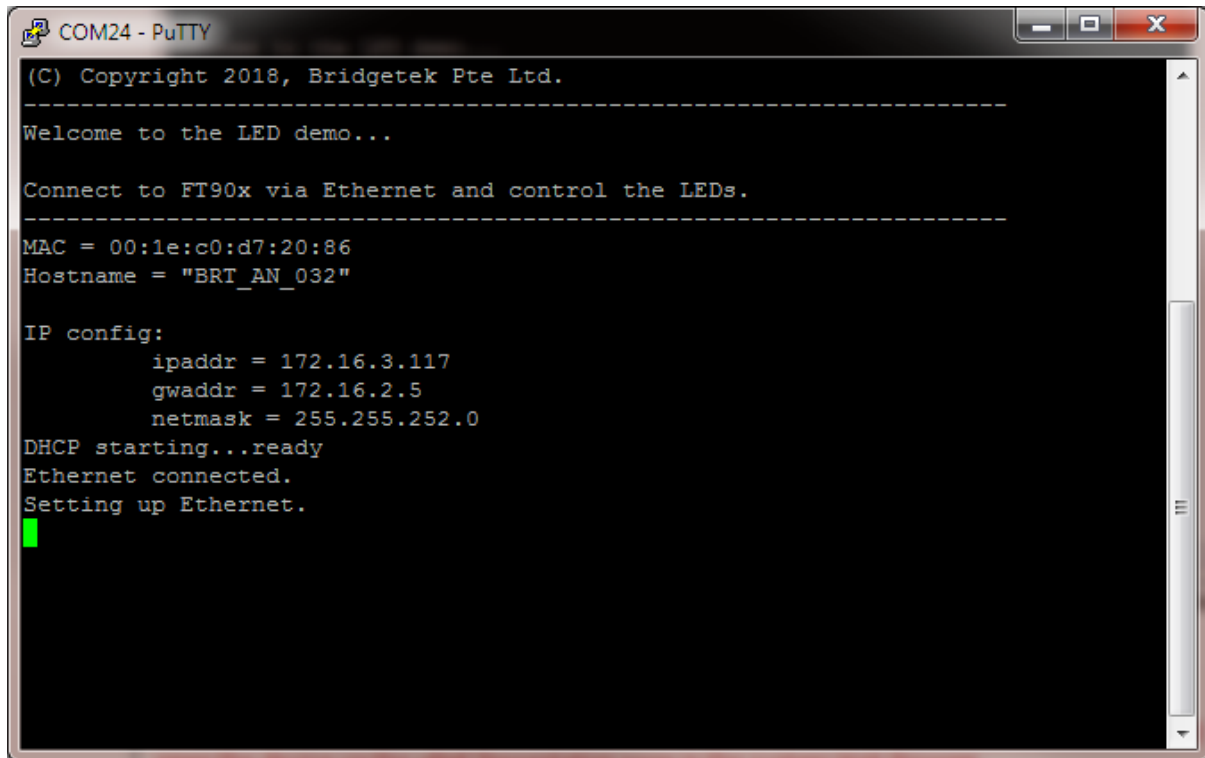
The supported device is defined elsewhere in the code as "upnp:rootdevice". This allows a simple, minimum function UPnP device to show a "PresentationURL" on an UPnP Control Point. In Windows, double-clicking on the device in Windows Explorer will open a web browser with a link to the main page of the device.

4 Operation

To control the LEDs a connection must be made from a web browser to the device.

4.1 Discovery

The IP address of the device can be found by monitoring the UART connection to the FT90X.



```
COM24 - PuTTY
(C) Copyright 2018, Bridgetek Pte Ltd.
-----
Welcome to the LED demo...

Connect to FT90x via Ethernet and control the LEDs.
-----
MAC = 00:1e:c0:d7:20:86
Hostname = "BRT_AN_032"

IP config:
  ipaddr = 172.16.3.117
  gwaddr = 172.16.2.5
  netmask = 255.255.252.0
DHCP starting...ready
Ethernet connected.
Setting up Ethernet.
█
```

Figure 4 Serial Port Output showing Network Configuration

It is also possible to discover the device through uPnP. On Windows this can be found on the Windows Explorer "Network" section in the "Other Devices" part. Double clicking on this icon will open the device *via* the uPnP Presentation page in the system Web Browser.

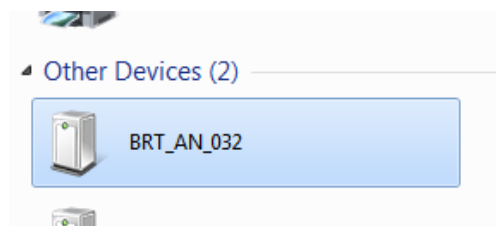


Figure 5 Windows Explorer "Network" Discovery

4.2 Colour Picker Page

The main page is a colour picker which has 2 independent colour maps and a slider for brightness. There are preview bars for the chosen colour below.

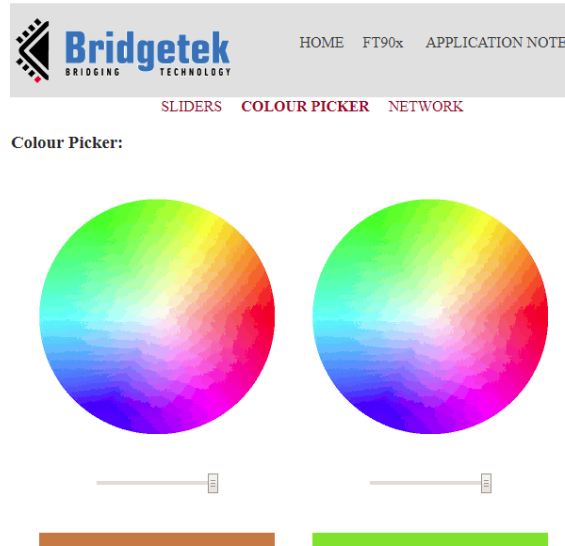


Figure 6 Colour Picker Page

When the user clicks the colour selector or changes the brightness value slider a GET request is sent to the device *via* JavaScript. This does not result in the page being reloaded from the device.

4.3 Slider Page

The second page is one with separate sliders for the Red, Green and Blue levels on each LED. Again there are preview bars for the chosen colour.

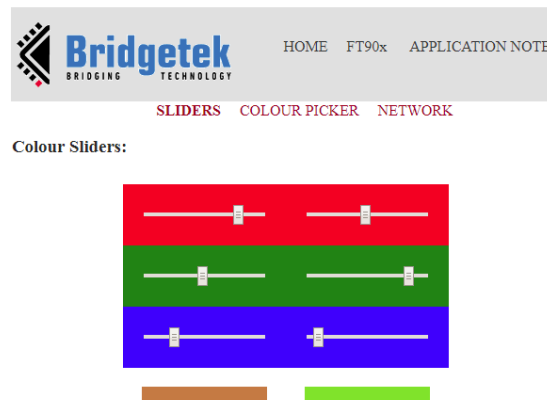
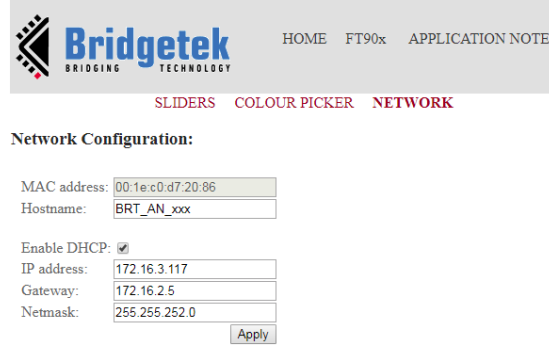


Figure 7 Slider Page

When a slider value is changed then a GET request is sent to the device *via* JavaScript. This does not result in the page being reloaded from the device.

4.4 Network Page

The configuration page is used to change the network settings of the device.



Bridgetek BRIDGING TECHNOLOGY

HOME FT90x APPLICATION NOTE

SLIDERS COLOUR PICKER **NETWORK**

Network Configuration:

MAC address: 00:1e:c0:d7:20:86

Hostname: BRT_AN_000

Enable DHCP:

IP address: 172.16.3.117

Gateway: 172.16.2.5

Netmask: 255.255.252.0

Apply

Figure 8 Network Configuration Page

To commit changes to the network configuration the “Apply” button will send the updated information to the device and the application will change the non-volatile settings. The device will be reset via the watchdog timer to enable the new settings to be used.

5 Importing into the FT9XX Toolchain

The firmware found at the following link can be easily imported into the [FT9XX Toolchain](#):

[http://brtchip.com/ft90x/#FT90x UART to USB BOMs Memory Bridge](http://brtchip.com/ft90x/#FT90x_UART_to_USB_BOMs_Memory_Bridge)

Once installed, select File --> Import --> General --> Existing Projects into Eclipse, and point to the downloaded and extracted project directory.

The project will appear in Eclipse Project Explorer as shown in Figure 5.1.

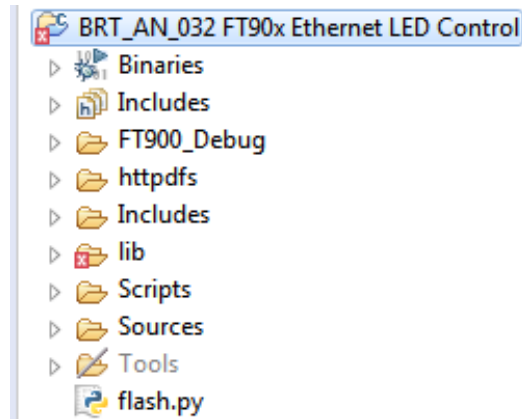


Figure 5.1 Eclipse Project Structure

6 Contact Information

Head Quarters – Singapore

Bridgetek Pte Ltd
178 Paya Lebar Road, #07-03
Singapore 409030
Tel: +65 6547 4827
Fax: +65 6841 6071

E-mail (Sales) sales.apac@brtchip.com
E-mail (Support) support.apac@brtchip.com

Branch Office – Taipei, Taiwan

Bridgetek Pte Ltd, Taiwan Branch
2 Floor, No. 516, Sec. 1, Nei Hu Road, Nei Hu District
Taipei 114
Taiwan, R.O.C.
Tel: +886 (2) 8797 5691
Fax: +886 (2) 8751 9737

E-mail (Sales) sales.apac@brtchip.com
E-mail (Support) support.apac@brtchip.com

Branch Office - Glasgow, United Kingdom

Bridgetek Pte. Ltd.
Unit 1, 2 Seaward Place, Centurion Business Park
Glasgow G41 1HH
United Kingdom
Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

E-mail (Sales) sales.emea@brtchip.com
E-mail (Support) support.emea@brtchip.com

Branch Office – Vietnam

Bridgetek VietNam Company Limited
Lutaco Tower Building, 5th Floor, 173A Nguyen Van
Troj,
Ward 11, Phu Nhuan District,
Ho Chi Minh City, Vietnam
Tel : 08 38453222
Fax : 08 38455222

E-mail (Sales) sales.apac@brtchip.com
E-mail (Support) support.apac@brtchip.com

Web Site

<http://brtchip.com/>

Distributor and Sales Representatives

Please visit the Sales Network page of the [Bridgetek Web site](#) for the contact details of our distributor(s) and sales representative(s) in your country.

System and equipment manufacturers and designers are responsible to ensure that their systems, and any Bridgetek Pte Ltd (BRTChip) devices incorporated in their systems, meet all applicable safety, regulatory and system-level performance requirements. All application-related information in this document (including application descriptions, suggested Bridgetek devices and other materials) is provided for reference only. While Bridgetek has taken care to assure it is accurate, this information is subject to customer confirmation, and Bridgetek disclaims all liability for system designs and for any applications assistance provided by Bridgetek. Use of Bridgetek devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify and hold harmless Bridgetek from any and all damages, claims, suits or expense resulting from such use. This document is subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Neither the whole nor any part of the information contained in, or the product described in this document, may be adapted or reproduced in any material or electronic form without the prior written consent of the copyright holder. Bridgetek Pte Ltd, 178 Paya Lebar Road, #07-03, Singapore 409030. Singapore Registered Company Number: 201542387H.

Appendix A– References

Document References

[FT900/901/902/903 Datasheet](#)

[FT905/906/907/908 Datasheet](#)

[MM900EVxA datasheet](#)

[AN_324 FT9XX User Manual](#)

<http://brtchip.com/SoftwareExamples-ft90x/> - Source Code V1.0

Acronyms and Abbreviations

Terms	Description
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
LAN	Local Area Network
IC	Integrated Circuit
JPEG	Joint Photographic Experts Group
MCU	Microcontroller Unit
RAM	Random Access Memory
RGB	Red Green Blue (Color Model)
UPnP	Universal Plug and Play
SSDP	Simple Service Discovery Protocol (used by UPnP)

Appendix B – List of Tables & Figures

List of Tables

Table 1 - Project Files Overview	6
--	---

List of Figures

Figure 1 Block Diagram.....	4
Figure 2 LEDs on Yellow and Fuschia	4
Figure 3 Running the makefs.bat script.....	7
Figure 4 Serial Port Output showing Network Configuration	12
Figure 5 Windows Explorer "Network" Discovery	12
Figure 6 Colour Picker Page	13
Figure 7 Slider Page	13
Figure 8 Network Configuration Page.....	14
Figure 5.1 Eclipse Project Structure	15

Appendix C– Revision History

Document Title: BRT_AN_032 FT90X Ethernet LED Control
Document Reference No.: BRT_000224
Clearance No.: BRT#127
Product Page: <http://brtchip.com/ft900/>
Document Feedback: [Send Feedback](#)

Revision	Changes	Date
1.0	Initial version	2018-03-27