Ticket Display System with Arduino ⊕ HDMI Shield - Part I

Introduction

Ticket Display System is a common device you see it everywhere. From supermarkets, groceries, clinics, museums, whenever people need to queue for services such ticket display system shall exist. Usually it is designed with a personal computer (PC) or single board computer (SBC) running Windows. Queue management software is preloaded in hard-drive and launched from PC startup.

Photo below shows similar system from a local supermarket.

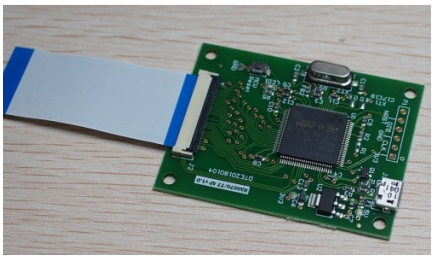
But there are always alternatives to PC with less power consumption, shorter boot up time, and much less complexity. In this document I am going to show you how a simple ticket display system is finished with a HDMI Shield-Arduino combo. Few lines of code in Arduino IDE is enough to do it.

Materials

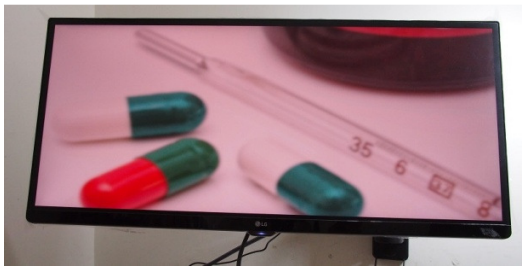
The following materials are required to run the example in this document.

- HDMI Shield version 2/1 (mandatory)

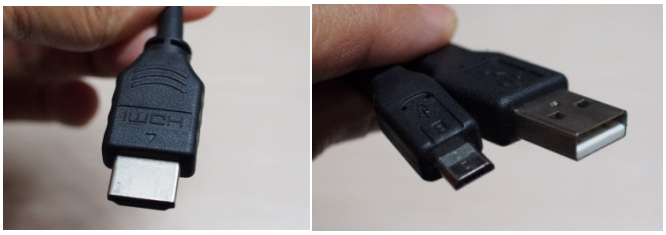
- STM32F103 driver board (mandatory)



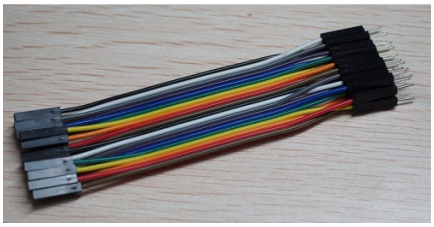
- A 1080p compatible HDTV/Monitor with HDMI/DVI input (mandatory)



- HDMI cable & a MicroUSB cable (both mandatory)

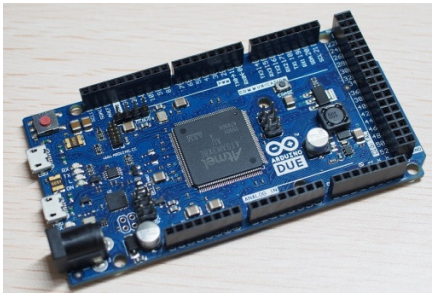


- Jumper cables(optional)



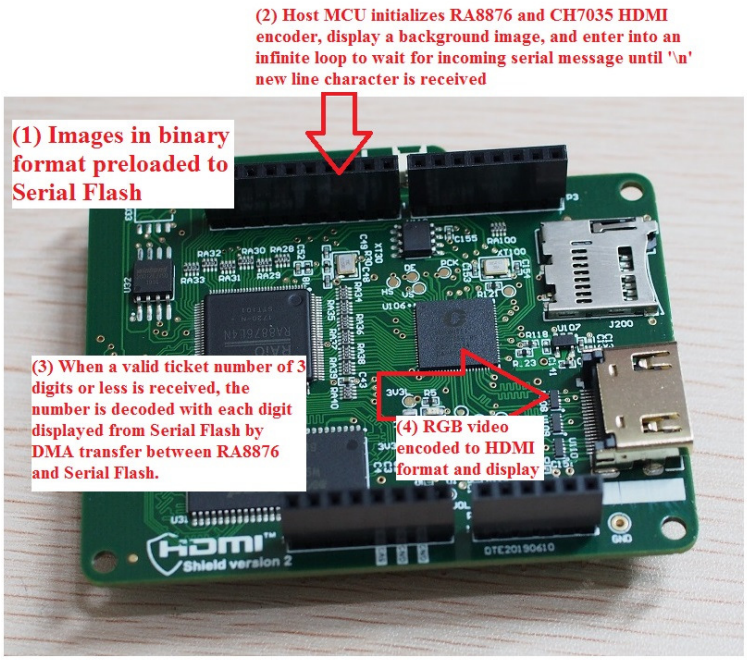
- DUE Zipper board with ESP8266 as MCU host or other Arduino boards we have tested so far (Arduino DUE as an optional item in this document)

or



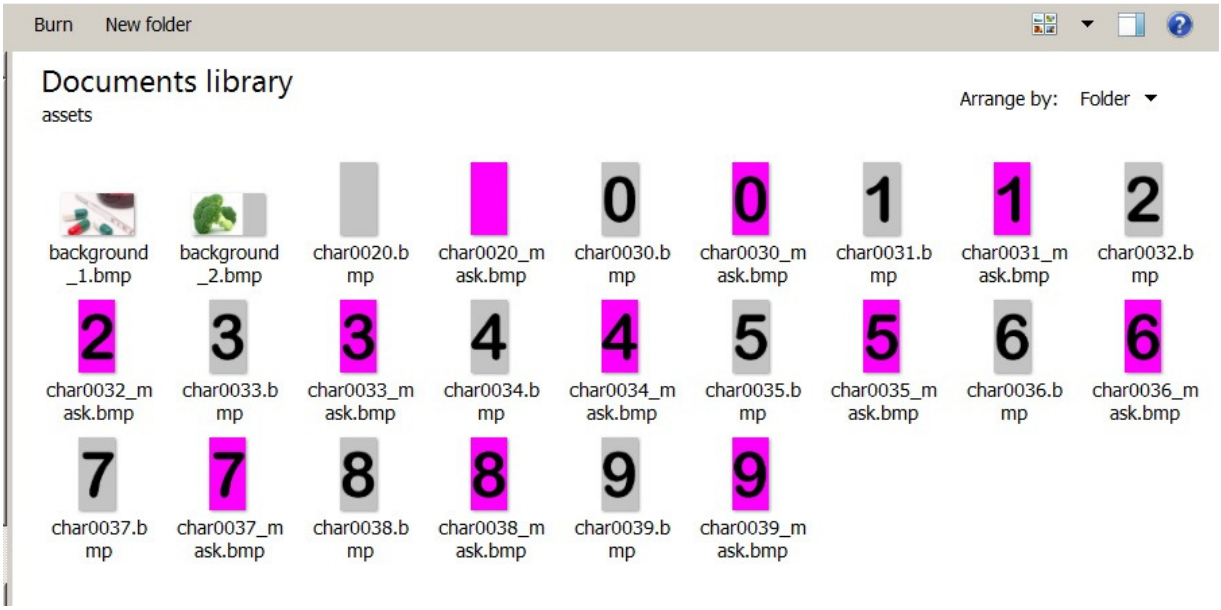
Working principle

is illustrated with annotations below:

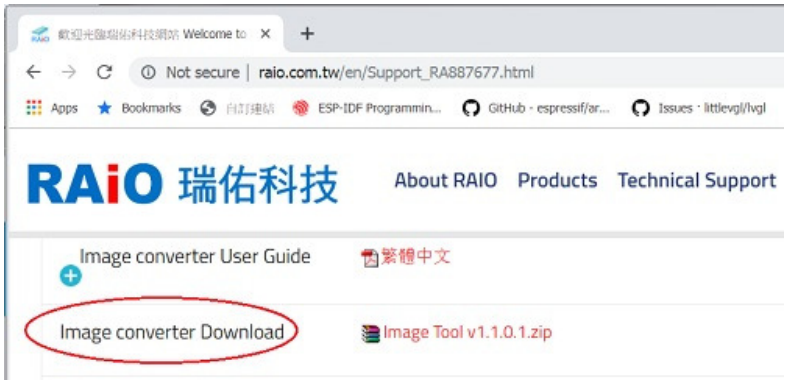


Procedures

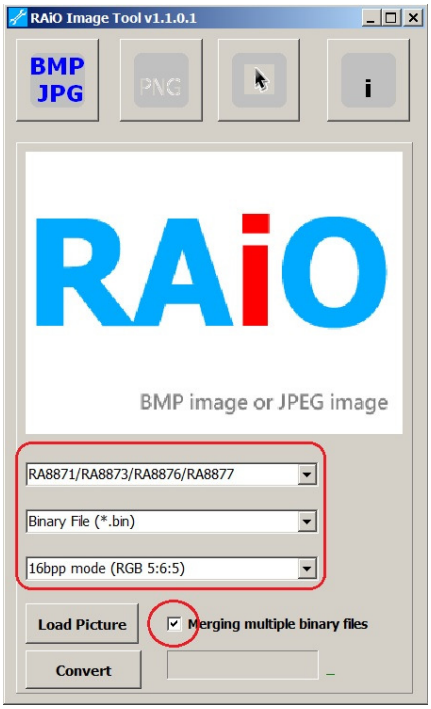
1. First we need to prepare the background image and digits in bitmap format. Digits from '0'-'9' and two files "background_1.bmp[i]" and "background_2.bmp[ii]" have been prepared for you. They are included in the **\assets** folder. Digits in bitmap format with naming convention char00xx.bmp denote ticket number with xx=hex code of digit. Files with _mask suffixes are used for sprite version which will be explained in Part II of this document.



2. Now we need to use an Image Converter software to convert BMP/JPEG to binary. It is available from RAiO's web site at http://www.raio.com.tw/en/Support_RA887677.html. Its user guide is written in Chinese but we will walk through its operation in this document.



3. Download and install. Launch it with pull-down menu options as screen shot below. There are only three pull-down menus and one checkbox to use.



Please notice that our RA8876 firmware works with RGB5:6:5 only.

You also need to have Microsoft Excel installed in your PC for this tool to work because a .xls file will be generated to show the address map of binary files created.

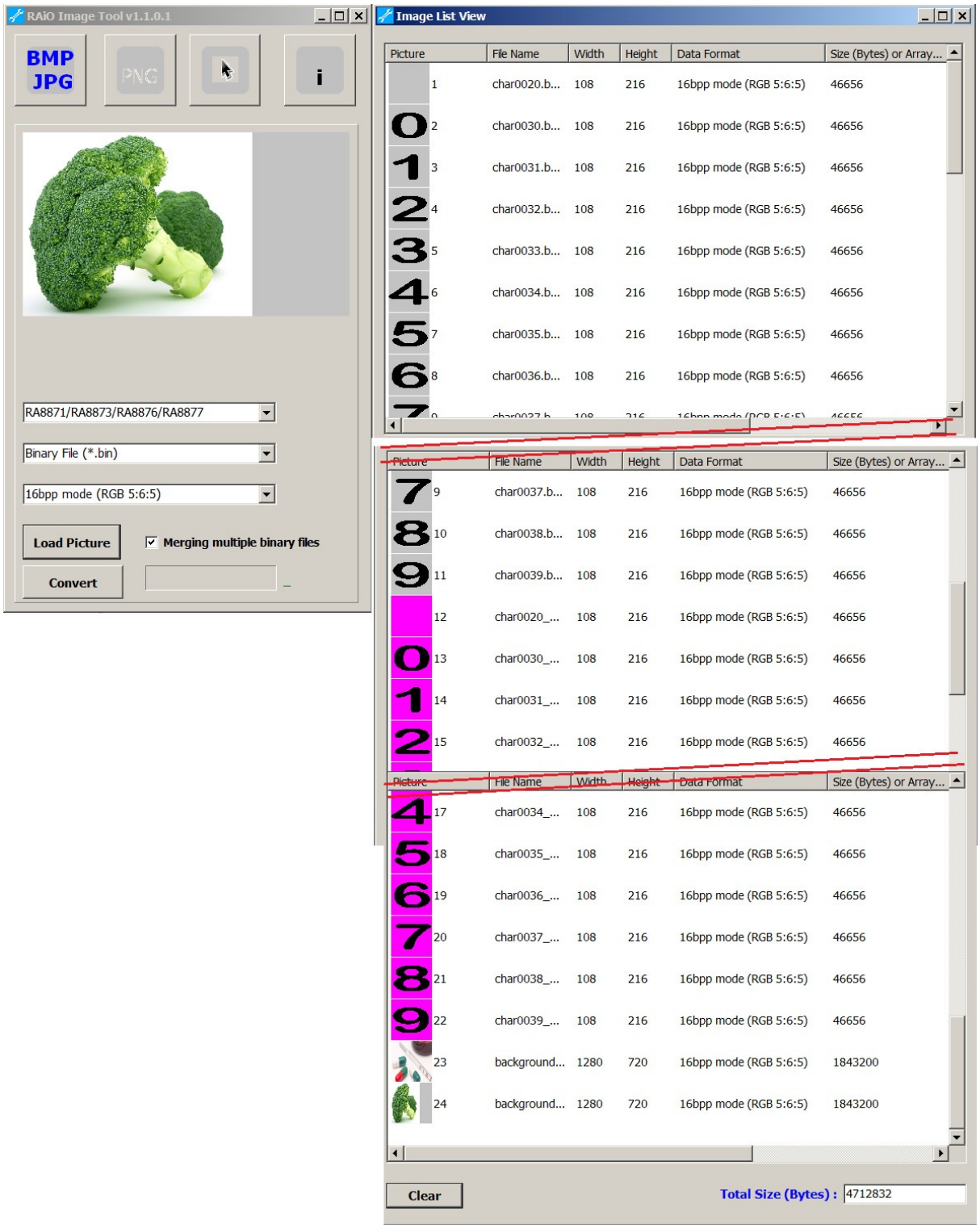
Now click the **Load Picture** button, and browse to bitmap files one-by-one from **\assets** folder in the order

- char0020.bmp > char0030.bmp > ... > char0039.bmp
- char0020_mask.bmp > char0030_mask.bmp ...> char0039_mask.bmp
- background_1.bmp > background_2.bmp.

The order is important because bitmap files are converted to binary data with addresses in the same order of a structure declared in the Arduino Sketch (Ra8876_ticketdisplay_part1.ino).

When more images are added to the RAiO Image Tool you will see the byte size become larger. At the end we are consuming 4,712,832 bytes for the ticket display system. We have no problem with this consumption because the Serial Flash onboard of our HDMI Shield is a 128Mbit model which is equivalent to a 16 Mbytes data capacity.

At the end you will see user interface like this :



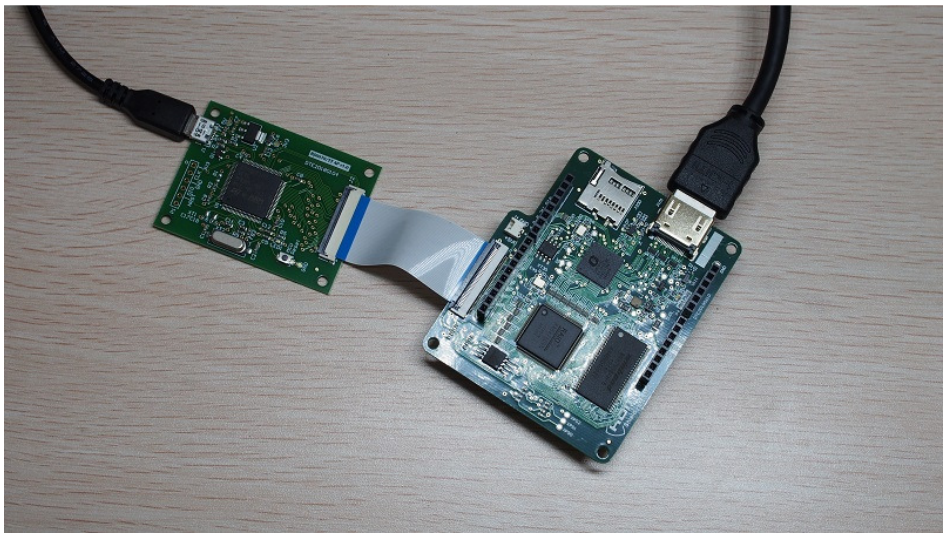
4. Click **Convert**. Inside the **/assets** folder you will see three files generated. They are
    - All_76_Pic_65K.bin (binary file to be preloaded to Serial Flash)
    - All_76_Pic_65K.h (a brief guide in .h file, but we won't use it here)
    - All_76_Pic_65K.xls. (Excel file containing image parameters and addresses that we will use it in source code)

    You may close the RAiO Image Tool now. Its task is finished.

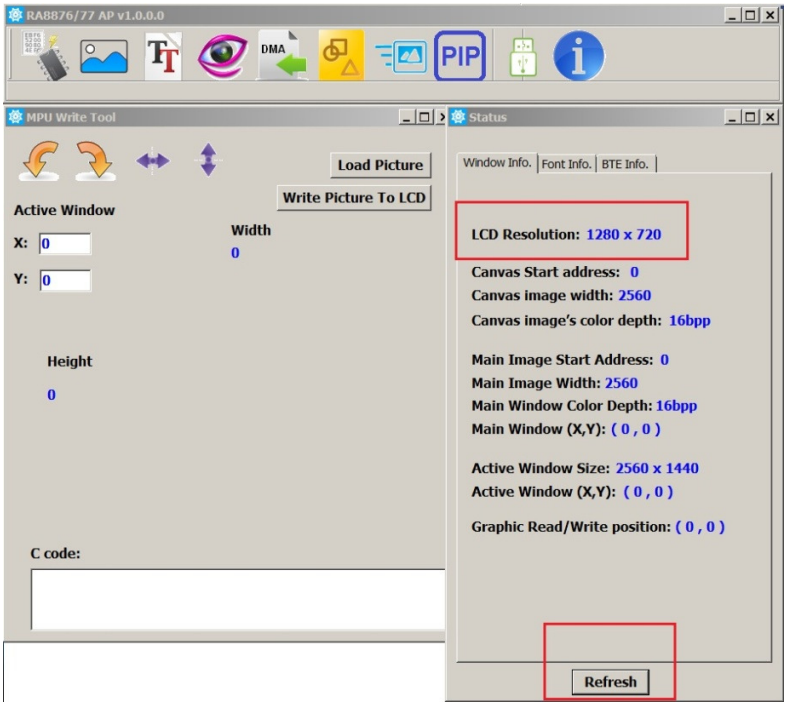
5. Launch RA8876/77 AP Software. This program can be downloaded from http://www.raio.com.tw/en/Support_RA887677.html.

By this time I have assumed the device driver for STM32F103 has been properly installed. Its procedure is described in the Getting Started Guide[iii] from page 17.

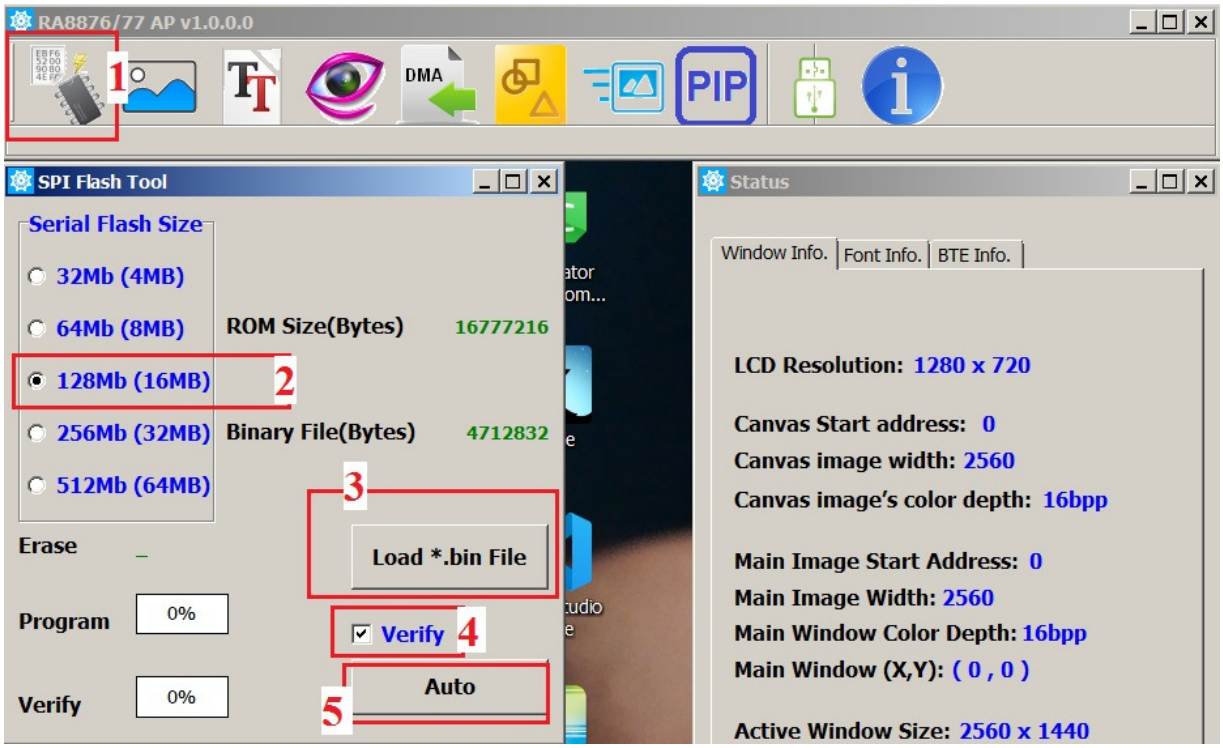
6. Now we need to connect the STM32F103 driver board with HDMI Shield. Connect MicroUSB port of STM32F103 to your PC.



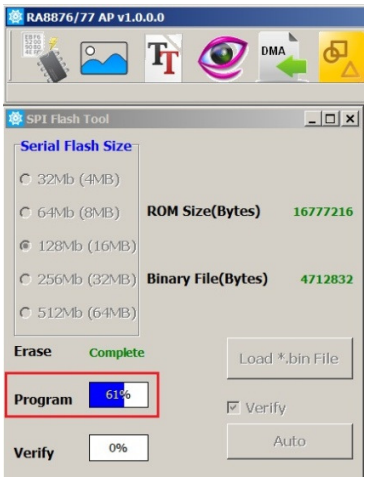
7. With the correct driver installed you will see all buttons enabled. Make sure the **Status Window** shows 1280x720 as the LCD Resolution. If not, Click **Refresh** button.



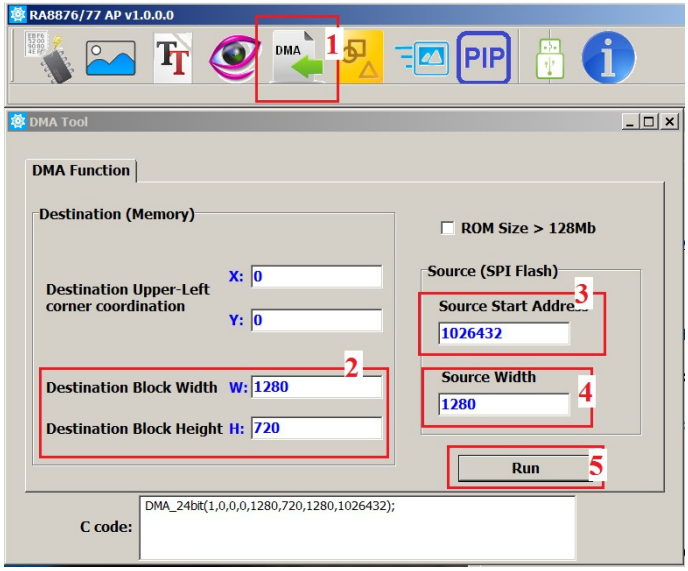
8. Click **SPI Flash Tool** button at the left. You will see a dialog box for **SPI Flash Tool** with UI as below. Follow these procedures:
    - Click checkbox **128Mb (16MB)** for HDMI Shield board version 2. If you have a board version 1 (with 256Mbit Serial Flash), check **256Mb (32MB)**
    - Click **Load *.bin File** button to browse to the file 'All_76_Pic_65K.bin' we have created in step 4
    - Check **Verify** checkbox
    - Click **Auto** button



Downloading progress will start with **Erasing...** and followed by **Program** until finish.

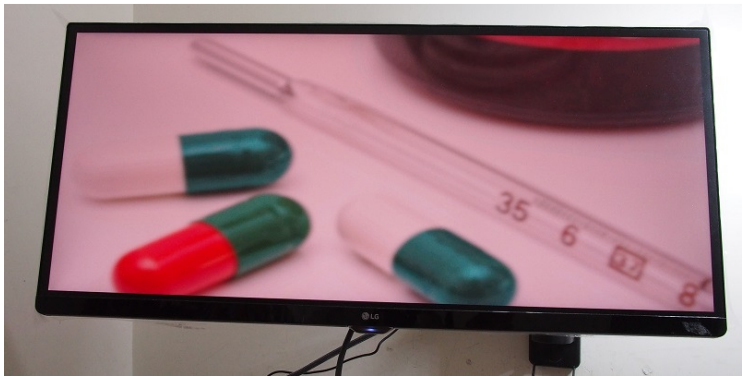


9. After data download you may test it with DMA button from the menu bar with procedures as follows:
    - Click **DMA** button
    - Input 1280 as **W**, 720 as **H**.
    - **Source Start Address** is 1026432. This data is available from the Excel file All_76_Pic_65K.xls.
    - **Source Width** is 1280.
    - Click **Run**



Remarks: If you are using HDMI Shield board version 1, click **ROM Size > 128Mb** because it is a 256Mb Serial Flash onboard.

Now you will see the monitor connected displaying the Pharmacy background image in no time.



Picture below shows the broccoli background with **Starting Address** changed to 2869632.

Feel free to try this software with different parameters, for example, a different **Destination Upper-Left coordinates** will display an image at another locations.

Now we have finished with the SPI Flash downloading task. Disconnect MicroUSB and HDMI cables, and disconnect STM32f103 driver board from HDMI Shield. Its task has finished.

Let's turn to the Arduino application (Ra8876_ticketdisplay_part1.ino).

Caution: To use this code on HDMI Shield Version 2 please update the Ra8876_Lite Arduino Library. Changes are:

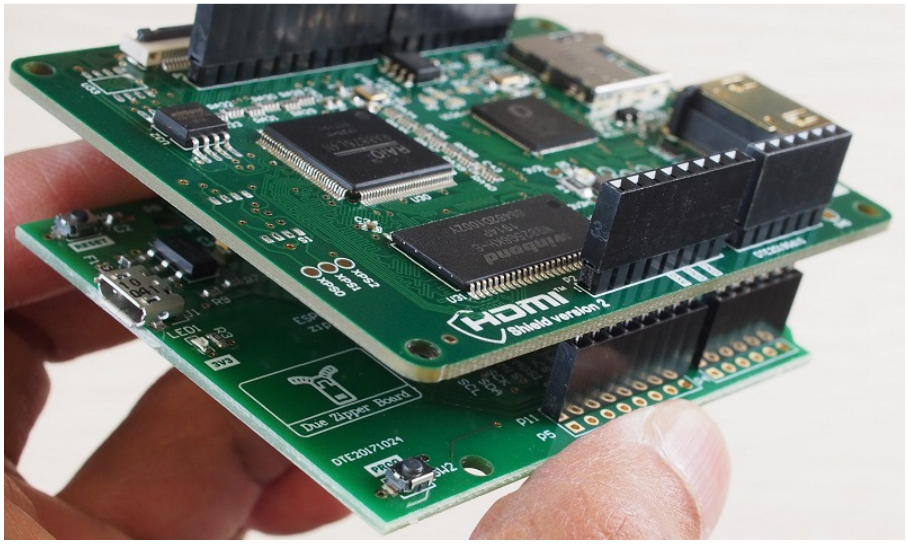
(1) File UserConfig.h : Add support for BOARD_VERSION_2 with 128Mbit Serial Flash

(2) File Ra8876_Lite.cpp : Add support for BOARD_VERSION_2 with 128Mbit Serial Flash

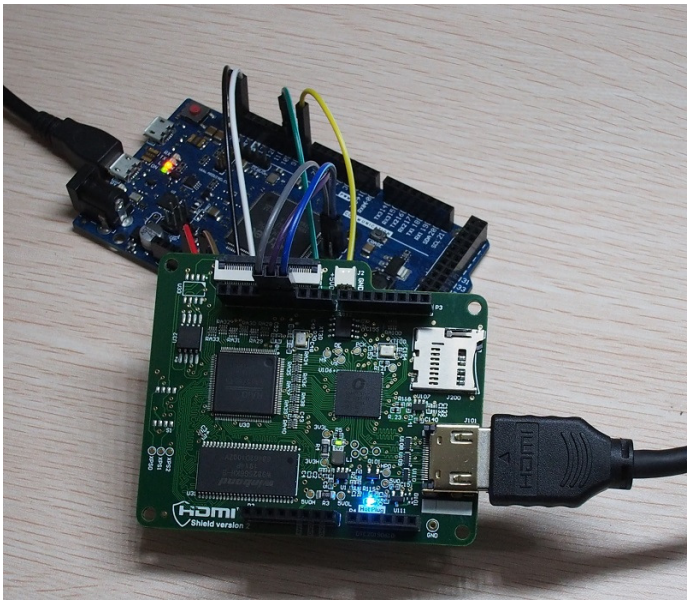
If you are using HDMI Shield Version 1, open UserConfig.h and comment #define BOARD_VERSION_2.

We will use **Due Zipper Board** with ESP8266 Wifi SoC as the host board. Stack HDMI Shield on it with microUSB cable connected to Due Zipper, HDMI cable to a monitor.

Make sure 2.00mm jumpers all set to P12 for ESP8266.

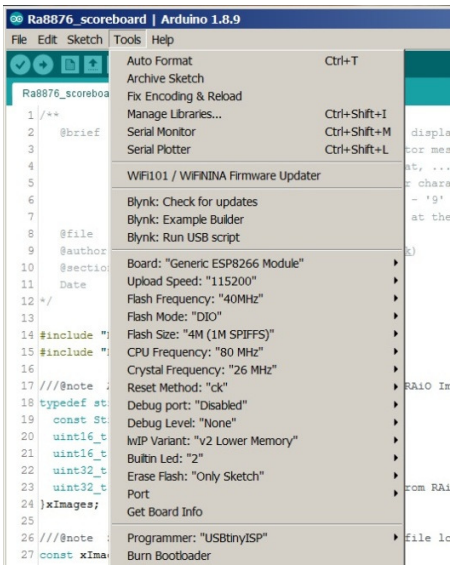


If you don't like to use our **Due Zipper Board**, it is okay to use jumper cables to connect it to your MCU board. Example below shows an Arduino DUE board connecting with jumper cables.



Open the sketch file 'Ra8876_ticketdisplay_part1.ino'.

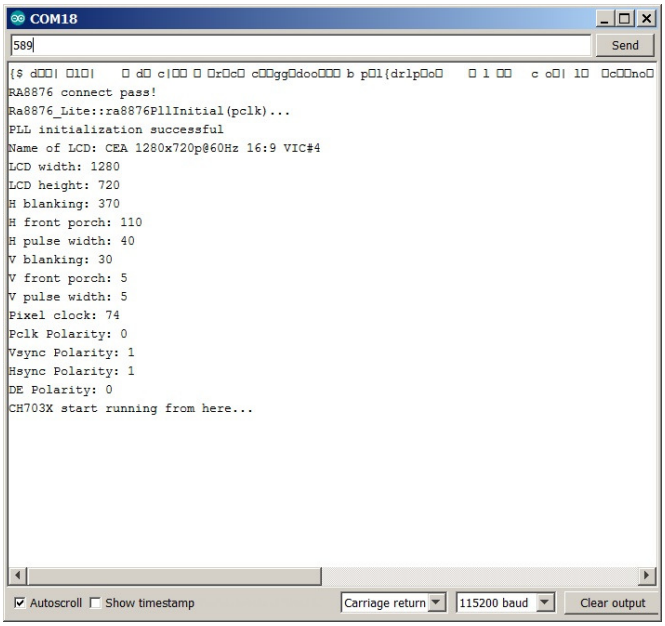
From [Tools] > [Board] select [Generic ESP8266 Module]. The other parameters are illustrated below:



On **Due Zipper board**, press and hold **PROG** button, and single click on **RESET** button to bring ESP8266 to bootloader mode.

Click [Sketch] > [Upload].

Open Serial Monitor from Arduino IDE, enter any number in 3 digits you like to display, e.g. **589<Send>**.



The monitor will show the queue number you just entered.

Try other numbers, say 12324545<Send>. The first 3 digits will be displayed with the remaining digits truncated.

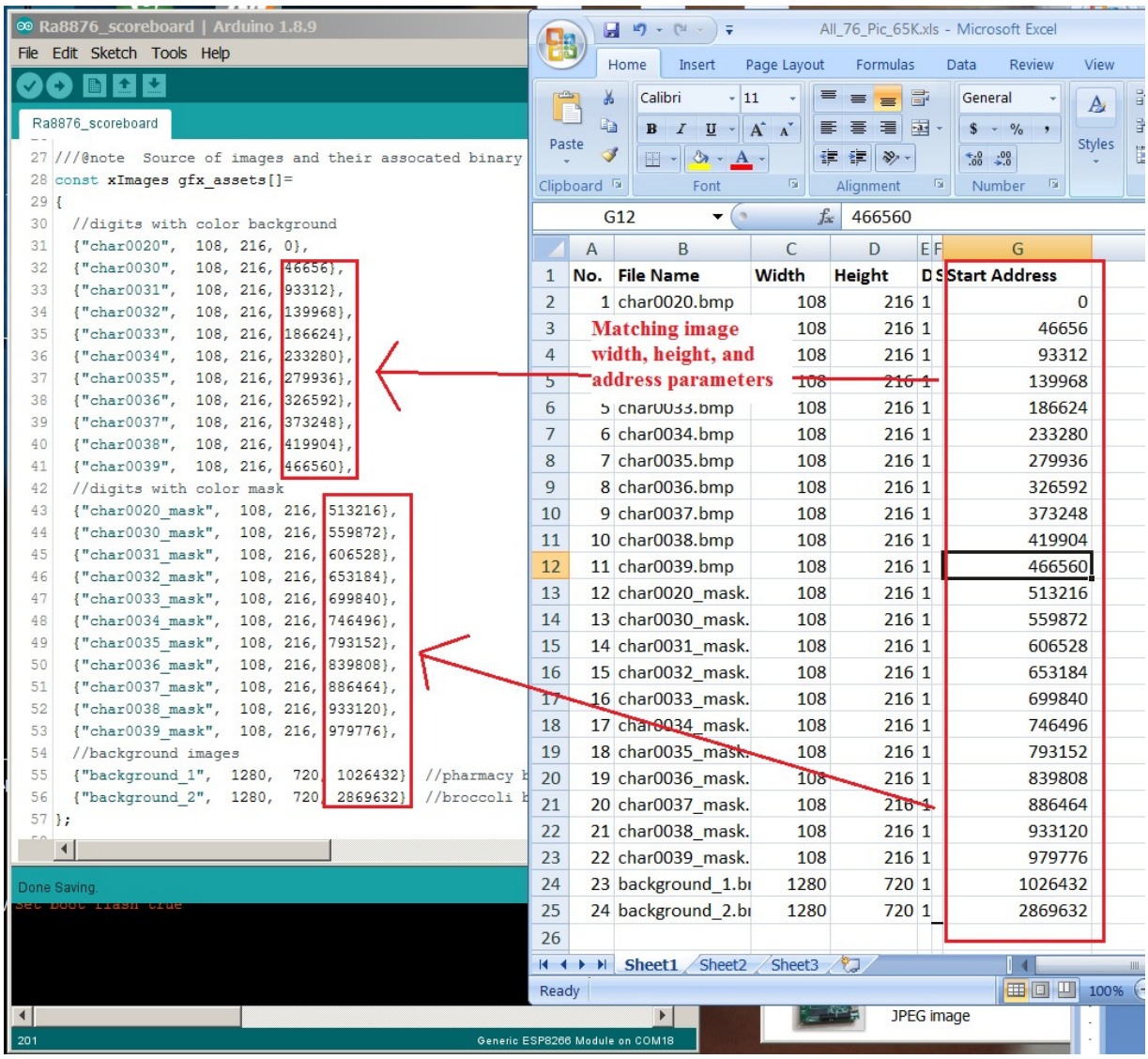
Also try changing the source code with another background

bgIndex = putSerialFlashImage(0, 0, "background_2") > putSerialFlashImage(0, 0, "background_1").

You will see the background changed to a with pharmacy theme.

Code description

The code starts from a description of images (**struct xImages**) that we have preloaded to Serial Flash in previous procedures. Screenshot below shows the map with reference to the All_76_Pic_65K.xls Excel file. If you change the position or content of images this map will change. You will have to change the parameters in source code to match the image parameters and addresses of the Excel file created after the change.



From setup() RA8876 is initialized with 1280*720 (720p) in 16-bit color mode. HDMI encoder is configured with a boosting factor from 720p to 1080p with DVI output mode.

Finally, a background image is displayed on screen with DMA function putSerialFlashImage().

```
172 void setup() {
173   Serial.begin(115200);
174   pinMode(RA8876_XNINTR, INPUT_PULLUP);
175   attachInterrupt(digitalPinToInterrupt(RA8876_XNINTR), isr, FALLING); //pending for Vsync hardware is
176
177   ///@note  Initialize RA8876 starts from here
178   if (!ra8876lite.begin(&CEA_1280x720p_60Hz))  //init RA8876 with a video resolution of 1280*720@60Hz
179   {
180 #ifdef DEBUG_LLD_RA8876
181     printf("RA8876 or RA8877 Fail\n");
182 #endif
183   }
184   else
185   {
186     ra8876lite.canvasImageBuffer(1280,720);     //Canvas set to the same size of 1280*720 in 16 bit-pe
187     ra8876lite.displayMainWindow();             //align display window to the same canvas starting po:
188     ra8876lite.canvasClear(color.White);        //set whole screen to white on startup
189     ra8876lite.graphicMode(true);
190     ra8876lite.displayOn(true);
191
192     HDMI_Tx.begin();                    //Init I2C
193     HDMI_Tx.init(VIDEO_in_1280x720_out_DVI_1080p_60Hz); //map defined in .\HDMI\videoInOutMap.h
194     HDMI_Tx.setI2SAudio(0, 0, 0);       //disable audio
195     delay(1000);
196   }
197
198   bgIndex = putSerialFlashImage(0, 0, "background_1");
199 }
```

(1) Initialize RA8876 with resolution 1280*720 in 16-bit color

(2) Initialize CH7035 HDMI encoder to boost 1280*720 to 1080p (DVI)

(3) Display a background image with DMA

Actually this is almost the end of this program. What we need to do is to put the host MCU to an infinite loop to listen for incoming messages. Once a valid ticket number of 3 digits or less is received with a '\n' new line character, it is stored in a buffer and the function putJumboScore() is called to display each digit decoded by DMA transfer from Serial Flash to RA8876. Video in RGB is automatically converted to HDMI by CH7035 with no software intervention once it is configured in initialization phase. Screen shot below shows the main loop of this program, which takes only 5 lines of code!

```
201 void loop() {
202     char char_buffer[4];
203 #ifdef ESP8266
204   yield();  //this is to avoid wdt reset in ESP8266
205 #endif
206     String inString = messageParser(); //blocking function here until \n received
207     sprintf(char_buffer, "%3d", inString.toInt()); //format to 3 digits
208     char_buffer[3]='\0'; //null terminate
209     putJumboScore(930,128, char_buffer); //print digits on screen
210 }
```

This is it! We have finished the first part of Ticket Display System with Arduino.

[i] Image Source : https://pixabay.com/photos/thermometer-temperature-fever-flu-833085/

[ii] Image Source : https://pixabay.com/photos/broccoli-vegetable-diet-food-fresh-1238250/

[iii] http://www.techtoys.com.hk/BoardsKits/HDMIshield/pdf/Getting_Started_Guide_DTE20190711_update20190804.pdf