Linux-based Digital Frame System

NuMicro® 32位系列微控制器范例代码介绍

文件信息

应用简述	本范例以电子相框为基础产品，并在 Linux 平台上实现整体应用设计。系统内建简易的 Web Server，使用者可透过浏览器上传或更新照片至装置中，形成一套直觉且方便的图片更新流程，提供产品在操作端的维护与应用方式。
BSP 版本	Linux-5.10.x
开发平台	NuMaker-HMI-N9H30

Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

本范例程序针对海外市场需求，将示范 Linux 环境下，快速完成显屏应用，搭配 NuMaker-HMI-N9H30 开发平台，以电子相框为题，逐步完成接口设计与功能整合。善用操作系统优势，展示 Web Server 应用，用户可以透过网页来更新或上传照片，即使在手机或平板依旧能上传照片。

此范例搭配新唐 GUI 开发平台 NuMaker-HMI-N9H30 开发板，其带有尺寸为 7 吋的 TFT-LCD 电阻式触控面板，分辨率为 800x480。在开始前，需先架设 NuMaker-HMI-N9H30 Linux 开发环境，请参考 ARM9 系列的 NUC9x0 Linux Environment on VMware User Manual 文件进行开发前配置。

2. 代码介绍

本电子相框系统框架如下所示，可分成两个部分。

Part 1 — 显示功能

核心显示功能，由主程序负责影像轮播，并从 photo 文件夹中依序读取图片档案显示于屏幕上；

Part 2 — 照片更新

新增 Web Server 负责更新影像，用户可透过浏览器或手机端经由 Wi-Fi 上传照片至 photo 文件夹，且在浏览器端会先执行图片缩放与旋转处理，使更新的影像符合显示器分辨率，无需在 Linux 端额外运算即可直接呈现，形成一套可实时更新播放的电子相框系统。

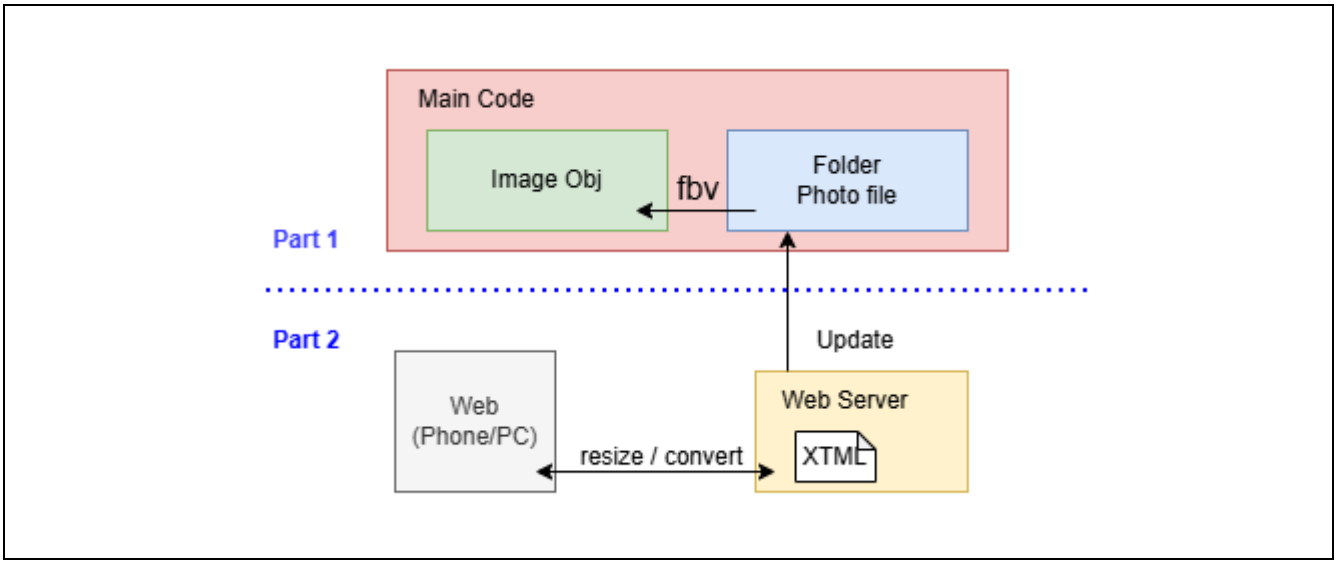

图 2-1 电子相框系统框架

如何建立 Linux 中的 Rootfs，请参阅 *Linux 5.10 RootFS Build Guide*.

本章节说明如何在 Linux/Buildroot 环境中实现「相片更新」功能。在 Buildroot 的 menuconfig 中启用以下必要套件（可使用 “/ 键”搜寻套件名称）：

- **BR2_PACKAGE_DHCPCD**

  用于取得 DHCP 指派的 IP 地址，负责网络接口的自动配置。
  Location :

  Target packages --->

Networking applications --->

[*] dhcpc

- **BR2_PACKAGE_NGINX**
  轻量级 Web 服务器，负责提供相片上传与管理所需的 HTTP 服务。

  Location:

  Target packages --->

  Networking applications --->

  [*] nginx

- **BR2_PACKAGE_FBV**

  简易的 Framebuffer 图片浏览工具，可直接在 Linux framebuffer 上显示 JPG/PNG/BMP 图片。

  Location:

  Target packages --->

  Graphic libraries and applications (graphic/text) --->

  [*] fbv

- **BR2_PACKAGE_LIGHTTPD**

  轻量化 Web Server，与 Nginx 类似但资源占用更低，适合嵌入式系统使用。可作为 HTTP 服务器提供档案上传、静态网页、REST API 等功能。

  Location:

  Target packages --->

  Networking applications --->

  [*] lighttpd

Part 1: 主程序系统

主程序持续扫描指定文件夹*/opt/photo* 文件夹，搜寻扩展名为 *.png* 的图片，并使用 **fbv** 指令在显示器上依序显示，形成轮播效果。程序亦会释放 **scandir()** 所配置的内存，避免内存泄漏。此程序可用于电子相框或简易显示系统。

注意，程序中只判断扩展名为 *.png* 的图片并显示，原因是在上传上来的照片会限制储存成*.png* 的格式，因此仅需要判断.png 的图片。

```
int main(void)
{
    struct dirent **namelist;
    int count;
    char fullpath[CMD_MAX_LEN];
    char cmd[CMD_MAX_LEN];

    while (1) {
        // Scan the photo directory and filter only *.png files.
        // 'filter_png' will return only files ending with .png
        // 'cmp_png' will sort them by filename or timestamp depending on your
implementation.
        count = scandir(PHOTO_DIR, &namelist, filter_png, cmp_png);
        if (count < 0) {
            perror("scandir failed");   // Print error if directory scan fails
            return -1;
        }

        // If no PNG files are found, wait 1 second and try again.
        if (count == 0) {
            printf("No PNG files found in %s\n", PHOTO_DIR);
            sleep(1);
            continue;
        }

        // Display each PNG file sequentially in sorted order.
        for (int i = 0; i < count; i++) {

            // Build the full file path: PHOTO_DIR + "/" + filename
            snprintf(fullpath, sizeof(fullpath), "%s/%s",
                     PHOTO_DIR, namelist[i]->d_name);

            // Use fbv (framebuffer viewer) to display the image.
            // Options:
            //   -a : automatically zoom to best fit
            //   -i : auto exit after display
            // Redirect all output to /dev/null to keep console clean.
            snprintf(cmd, sizeof(cmd), "fbv -a -i %s > /dev/null 2>&1", fullpath);
            system(cmd);   // Execute the command

            // Free each allocated dirent entry returned by scandir()
            free(namelist[i]);


            usleep(mseconds * 100);
        }


        sleep(3);

        // Free the namelist array allocated by scandir()
        free(namelist);
```

```
    }

    return 0;
}
```

Part 2: Web Server 处理流程

为 Web Server 处理程序，其中设置 *index.html* 来呈现使用者透过浏览器所呈现的 layout 布局，使用者可透过浏览器或手机端经由 Wi-Fi/Ethernet 上传照片至 *photo* 文件夹，且在浏览器端会先执行图片缩放与旋转处理。

用户的窗口布局如下图所示：

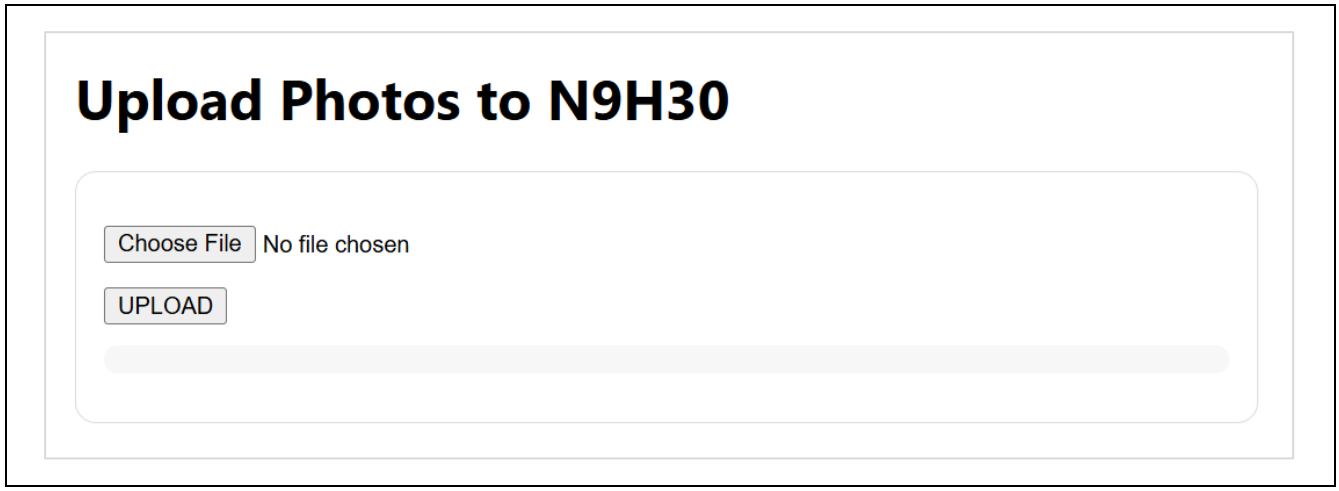

图 2-2 网页显示页面

网页 **HTML** 档案排版规划如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">

<!--
  Make layout responsive on mobile devices.
  Without this line, the UI will appear zoomed-in on smartphones.
-->
<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Photo Upload</title>

<style>
  /*
   * ===== Global Page Layout =====
   * Use system fonts for better performance.
   * Max width is set so content remains readable on large displays.
   */
  body {
```

```
    font-family: system-ui, -apple-system, Segoe UI, Roboto, Arial;
    max-width: 680px;           /* Limit UI width for better readability */
    margin: 40px auto;          /* Center vertically on desktop */
    padding: 0 16px;
  }

  /* Simple card-style container with soft rounded borders */
  .card {
    border: 1px solid #ddd;
    border-radius: 12px;
    padding: 16px;
  }

  /* Maintain consistent vertical spacing between each UI section */
  .row { margin: 12px 0; }

  /* Upload progress bar (hidden before upload starts) */
  progress {
    width: 100%;
    height: 16px;
  }

  /*
   * Status display area (upload result / error message).
   * `white-space: pre-wrap` keeps line breaks in log output.
   */
  #log {
    white-space: pre-wrap;
    background: #f7f7f7;
    padding: 8px;
    border-radius: 8px;
  }
</style>
</head>

<body>
<h1>Upload Photos to N9H30</h1>

<div class="card">
  <!-- File picker: only accept image formats -->
  <div class="row">
    <input id="file" type="file" accept="image/*">
  </div>

  <!-- Upload trigger button -->
  <div class="row">
    <button id="btn">UPLOAD</button>
  </div>

  <!-- Progress bar: hidden until upload begins -->
  <div class="row">
    <progress id="bar" max="100" value="0" hidden></progress>
  </div>

  <!-- Upload result / status message -->
  <div class="row" id="log"></div>
```

```
</div>
```

在网页中的 l**oadBitmap()**功能：

loadBitmap() 函数负责从 **<input type="file">** 取得的图片档案中加载影像，并自动套用 EXIF 的旋转信息，以确保手机拍摄的照片不会出现上下颠倒或横向显示的问题。

```
/* ① Load image and apply EXIF orientation (if supported by browser) */
async function loadBitmap(file) {
  if ('createImageBitmap' in window) {
    try {
      return await createImageBitmap(file, { imageOrientation: 'from-image' });
    } catch (e) { /* Fallback if failed */ }
  }
  // Alternative method using Object URL
  const url = URL.createObjectURL(file);
  try {
    const img = new Image();
    img.decoding = 'async';
    img.src = url;
    await img.decode();
    return img;
  } finally {
    // URL will be revoked after drawing on canvas
  }
}
```

resizeToCanvas()函数：

resizeToCanvas()用途在于将加载的图片依照屏幕分辨率进行等比例缩放，并绘制至固定大小，同时保留原始比例以避免形变；在缩放后若仍有空白区域，会以指定的背景色填满并将影像置中显示，使输出的影像完全符合 800×480 显示尺寸，减少嵌入式系统的图像处理负担。

```
/* ② Resize image to 800×480 (centered, keep aspect ratio, no distortion) */
function resizeToCanvas(bitmap, { maxW, maxH, background = '#000', upsize = true, mode =
'contain' }) {
  const iw = bitmap.width, ih = bitmap.height;
  let scale = mode === 'cover' ? Math.max(maxW/iw, maxH/ih) : Math.min(maxW/iw, maxH/ih);
  if (!upsize) scale = Math.min(scale, 1);
  const dw = Math.round(iw * scale);
  const dh = Math.round(ih * scale);

  // Create canvas buffer
  const canvas = document.createElement('canvas');
  canvas.width = maxW;
  canvas.height = maxH;

  const ctx = canvas.getContext('2d');
  ctx.fillStyle = background;
  ctx.fillRect(0, 0, maxW, maxH);
  ctx.imageSmoothingQuality = 'high';
```

```
  // Calculate offset (center image)
  const dx = Math.floor((maxW - dw) / 2);
  const dy = Math.floor((maxH - dh) / 2);
  ctx.drawImage(bitmap, dx, dy, dw, dh);

  return canvas;
}
```

canvasToBlob() 函数:

canvasToBlob() 负责将绘制完成的内容转换成 Blob 格式的图片文件，并限制输出为 PNG 并适应压缩质量；此方式可直接搭配 **XMLHttpRequest** 进行 HTTP PUT/POST 上传到 N9H30，不需要再进行额外的转档或压缩。

```
/* ③ Convert canvas to Blob (PNG or JPEG) */
function canvasToBlob(canvas, { type = 'image/png', quality = 0.85 } = {}) {
  // `quality` parameter is ignored by PNG but kept for compatibility
  return new Promise(resolve => canvas.toBlob(b => resolve(b), type, quality));
}
```

最后，程序设计完成后，需要将程序代码进行编译。待编译完成后，将生成执行档案。该档案可在 NuMaker-HMI-N9H30 开发平台上运行。以下说明 Linux 环境编译流程:

Linux 环境中的交叉编译流程:

1. 将 EC_N9H30_Linux-based Digital Frame System_V1.00 下的 BuildCode 移动到 N9H30 Linux 开发环境中**(Ubuntu)**。

2. 在进行交叉编译时，需要先确认工具链（toolchain）是否设定正确。**Makefile** 中会使用 **CC** 指定的交叉编译程序来将 **SRCS** 里的来源档案编译成 **TARGET** 的可执行文件，并可透过 **STRIP** 对产出的执行档进行处理。

```
CC := arm-linux-gcc
STRIP := arm-linux-strip

TARGET = DigitalPhotoFrame
SRCS := DigitalPhotoFrame.c
```

3. 执行下列指令即可加载 Buildroot 所产生的交叉编译程序设定，让之后的 **gcc**、**make** 等指令自动使用正确的 toolchain：

```
~/shared/N9H30/buildroot_2024$ source output/host/environment-setup
```

4. 移动到 **BuildCode** 文件夹，透过 **make** 指令进行编译:

```
~/shared/N9H30/buildroot_2024/BuildCode$ make
```

若要清除编译档案或重新编译，皆需执行以下指令:

```
~/shared/N9H30/buildroot_2024/BuildCode$ make clean
```

5. 编译完成后，将生成执行档 **DigitalPhotoFrame** 放置到*~output/target/opt* ，并重新 **pack**。该档案可在 NuMaker-HMI-N9H30 开发平台上运行。

```
~/shared/N9H30/buildroot_2024$ make linux-rebuild && make
```

- N9H30 系统配置说明:
- 在启动 **nginx** 之前，系统需先取得网络 IP。透过 **udhcpc** 从 **eth0** 网络接口向 **DHCP** 服务器请求并取得 IP 地址，之后才能启动 Web Server。

```
#!/bin/sh

echo q | fbv -i /opt/logo_800x480.png
# 1. Check DHCP (try up to 5 times)
if udhcpc -i eth0 -n 5; then
    echo "DHCP OK – IP acquired."

    # 2. Create required folders for nginx (to avoid 'No such file or directory' errors)
    mkdir -p /var/log/nginx
    mkdir -p /var/cache/nginx/client-body
    mkdir -p /run/nginx

    # 3. Start nginx
    echo "Starting nginx..."
    nginx -t
    nginx &

    # 4. Show current IP address
    IP=$(ifconfig eth0 | awk '/inet /{print $2}')
    echo "Server running at: http://$IP/"

    # 5. Start the Digital Photo Frame application
    ./DigitalPhotoFrame &
else
    #  DHCP failed → show no-network warning image
    echo "DHCP FAILED – showing no network image..."
    fbv /opt/no_network.png
    exit 1
fi
```

- Nginx 配置说明:

```
http {
    include       mime.types;
    default_type  application/octet-stream;

    sendfile        on;
    keepalive_timeout  65;
```

```
    server {
        listen       80 default_server;
        server_name  _;

        # Web root for static files (HTML, JS, CSS, etc.)
        root  /opt/www;
        index index.html;

        # Home page: always serve index.html
        location = / {
            try_files /index.html =404;
        }

        # SPA-style routing:
        location / {
            try_files $uri $uri/ /index.html;
        }

        # === Upload folder: support HTTP PUT ================================
        # URL:  /upload/<filename>
        # Path: /opt/photo/<filename>
        location ^~ /photo/ {
        root /opt;

        client_max_body_size 10M;
        dav_methods PUT;
        create_full_put_path on;
        dav_access group:rw all:r;

        autoindex on;
        autoindex_localtime on;
    }
        # ====================================================================

        # Captive portal / OS connectivity checks -> redirect to main page
        location = /hotspot-detect.html       { return 302 /index.html; }
        location = /library/test/success.html { return 302 /index.html; }
        location = /generate_204              { return 302 /index.html; }
        location = /gen_204                   { return 302 /index.html; }
        location = /clients3/generate_204     { return 302 /index.html; }
        location = /connecttest.txt           { return 302 /index.html; }
        location = /ncsi.txt                  { return 302 /index.html; }

        # Simple error page
        error_page 500 502 503 504 /50x.html;
        location = /50x.html { }
    }
}
```

- 在首次设定或建立编译环境时，系统于开机后会自动执行 **emWin AppWizard** 的范例程序。因此，若需改为启动自定义应用程序，需先修改以下档案后重新 **pack**：

```
~buildroot_2024/board/nuvoton/nuc9x0/rootfs-emwin/etc/init.d
```

- 新增启动的程序名称或脚本:

```
#!/bin/sh
export TSLIB_CONFFILE=/etc/ts.conf
export TSLIB_TSDEVICE=/dev/input/event0
export TSLIB_CALIBFILE=/etc/pointercal
export LD_LIBRARY_PATH=/lib/ts/
/sbin/mdev -s
cd opt
./init.sh  &
```

3. 软件与硬件需求

3.1 软件需求

- Embedded OS: Linux 5.10 版本

3.2 硬件需求

- 电路组件
  - NuMaker-HMI-N9H30

4. 目录信息

目录结构如下图所示。

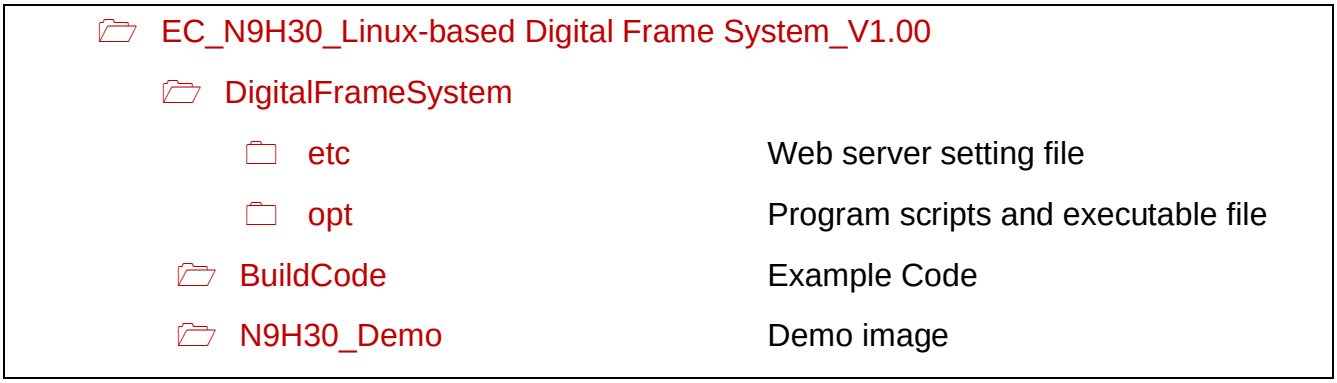

图 4-1 目录信息

5. 范例程序执行

1. 透过 NuWriter 工具，将 *N9H30_Demo* 目录中的 *n9h30image* 档案烧录到开发平台

   可以透过烧录到 DDR 来测试本范例功能。如果是烧录 *n9h30image*，本范例程序会在开机完成后自动执行。

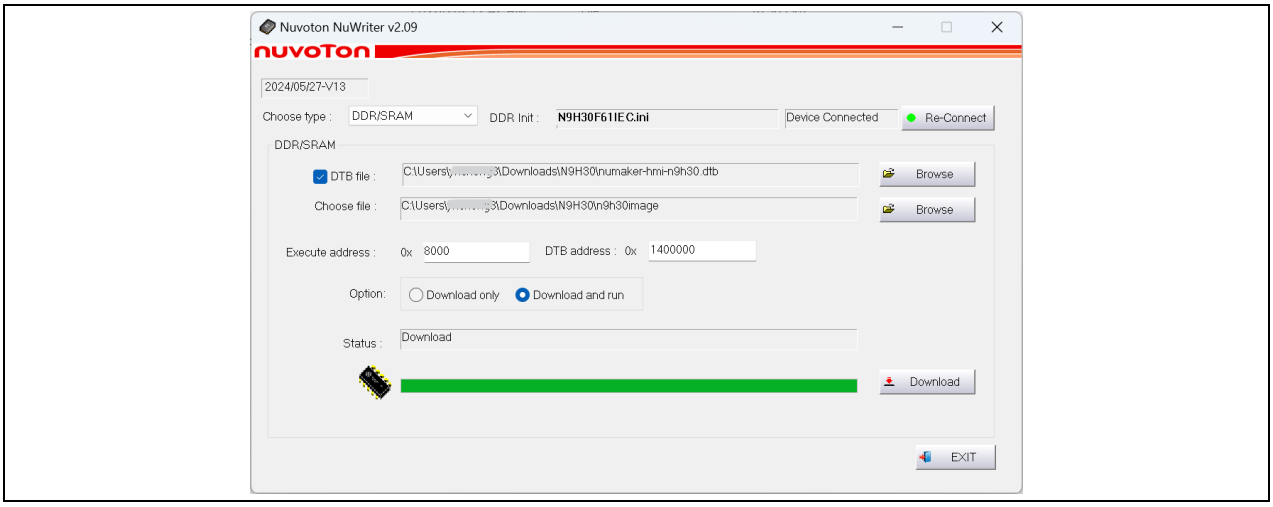

图 5-1 NuWriter 烧录工具

2. 在启动 NuMaker-HMI-N9H30 前，需先连接以太网络至 **eth0**。

3. 将手机与 NuMaker-HMI-N9H30 连接同一个网络，用户可以上传更新照片。

4. 用户可以透过 **ifconfig** 查看当前 IP。

```
~ # ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:00:01:92
          inet addr:192.168.0.32  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe00:192/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1912 errors:0 dropped:0 overruns:0 frame:0
          TX packets:290 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:32
          RX bytes:375753 (366.9 KiB)  TX bytes:26060 (25.4 KiB)
```

图 5-2 IP 查询

5. 透过浏览器开启 192.168.0.32 网址。

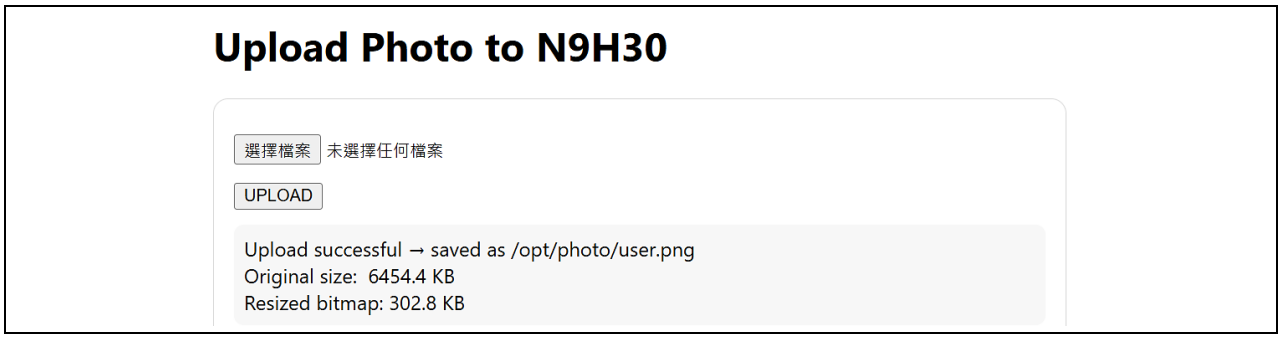

图 5-3 网页更新画面

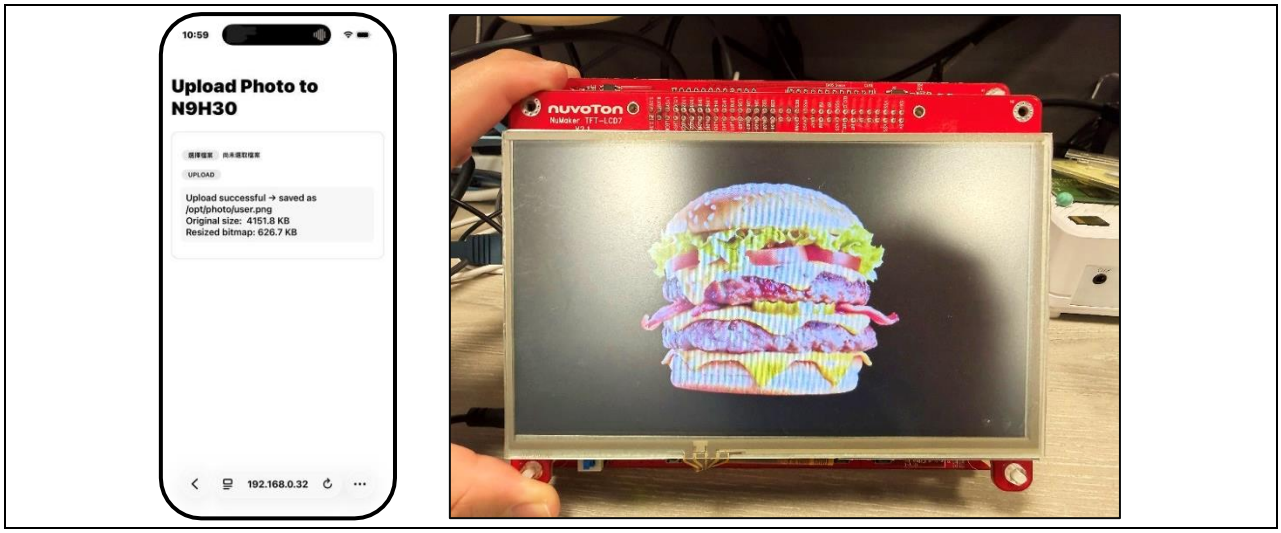

图 5-4 Digital Frame System 执行结果及手机上传更新页面

用户可依照下列步骤自行编译与部署系统：

1. 开启 N9H30 Linux 编译环境，并将 **DigitalFrameSystem** 中的 *etc* 及 *opt* 文件夹复制到 *~output/target* 下的对应文件夹。重新 pack 映像档后烧录到 NuMaker-HMI-N9H30。此步骤流程可参考第二章节。

2. 进入 NuMaker-HMI-N9H30，输入以下指令：

   移动到 *opt* 文件夹

```
cd opt
```

   透过以下指令执行档案：

```
./init.sh  &
```

6. 修订纪录

Date	Revision	Description
2025.09.10	1.00	初始发布。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.