Little "Nothing" suitable as "Dust catcher on the shelf"

As I wrote here, company Giga Devices launched RISC-V microcontroller GD32VF103. Nothing groundbreaking, but I've found mini-development kit at Aliexpress for this MCU.
Still nothing special...

But, this kit has 0.96" 160x80 IPS RGB LCD display - and this caught me.

So, for the display, I bought the whole kit....

The kit was made by the China manufacturer Sipeed and it selling by Seeed Studio at the price 4.90 USD, but the delivery cost 5.- USD, so I used Aliexpress. It   was on offer (the seller no longer offers it) just for 7.- USD with free delivery.
Just because I wanted to save some money, I bought it.
You can still found it at Ali, but you just need to be careful - some sellers offers it without LCD display.

So anyway, kit arrived succesfully, and I was full of expectations of display quality, and I decided to test it. I hoped that some demo app will be on it...

But the USB connector grounded me... Of course i didn't check what kind of connector the manufacturer used. It is USB-C.
And guess what happened! I didn't have a single USB-C at my workshop!

The problem was solved by a local smartphone holders shop, whera I bought cable for ridiculous 7 euros.
At least It was in a nice box... (save money... 😵)

Next shock was after I plugged it in. Although the demo app was loaded in the kit, it read graphic data from SD cards (not included of course), and the demo data was in black and white only... 😩

So, the process of generating your own graphic data started.

1. **DEMO firmware**
   Kit **Longan nano** contains an demo application, that after run looking for two binary files on the card:
   - logo.bin
   - bmp.bin

   The logo.bin file contains the Sipeed logo, although in red, but it ain't much.
   Upon startup, this logo will apear on the display for a short time and then the sequence of images from the bmp.bin file will start playing (showing). They are really bmp images that are   packed in one file.
   Actually I'ts playing animation named bad_apple - I dont know whether the name have something to suggest, or if it's their traditional tale.

   Firmware is written in kind of platformio enviroment (using fatfs by ChaN), which I can't run, although they wrote that it's just few click on the mouse. Maybe I have different mouse.

   So It's time for plan B - try to generate your own bmp.bin file.

2. **Documentation**
   Horrible
   I don't know read "dumped tea" language yet, so the google translator got very hard work. There isn't much informations in english. Of course there are wiki pages with descriptions, but the link to description of the graphic binary files doesn't work right here.
   You need to switch to the East Asian dialect.

   I extracted procedure out of this.

3. **Creating a sequential file from the video**
   Please note that PC (I using windows) have to a [python](#) interpreter installed.

   As I mentioned later, **Logan nano** can display bmp files. Changing them quickly create a feeling of video (no audio). It's possible to generate such a   sequence of images by simply capturing the screen during playback in movie player and then packing it into one binary file.

   How to do it?.

   1. [PotPlayer](#) - first off all, you need to download and start this media player and open your favourite video, which have to be   displayed on the 0.96 inch display with the resolution 160x80 px (so small for XXX)
      - After launching player, you need to set **CTRL+G** - "Consecutive Image Capturer" - to save captured video images to files.
      Required settings: BMP format, size: 160x80. Maximum playback time: according to the capture rate, the maximum time can be 219 sec (2190 frames every 100 ms)
      - Start playing the video and then start capturer.
      After you reach the set number of captured images, needs to adjust them the desired color depth.

   2. Program [XNViewMP](#) is needed for changing properties of images
      - So, first you need to point XnViewMP to the folder with captured images.
      - Press **CTR+U** to open bulk conversion
      - Set INPUT: change the sorting so that the first frame is on top
      - OPERATIONS: add operation "change color depth" and set to **24 bits**
      - OUTPUT: set the output directory. Enter the file name as # (numbered files)
      - Run it.
      Captured screens with the names 1.bmp to xx.bmp will saved in the output directory (depending on how many were input).

   Capture in linux

      1.
         ```
         ffmpeg -i inputvideo.mkv -s 160x80 -pix_fmt bgr24 thumb%04d.bmp -hide_banner
         ```

   3. Next step is combining them all into the one binary file. There is a packaged archive with [files](#) available (ha - you need python on a computer for this!).
      Extract these scripts and copy them to a directory with saved generated and converted bmp files.
      - In file genhex.py, change the nuber of repetitions for the cycle in the "while num <= xxxx:" line to the nuber that corresponds to the last image name (change xxx to a number)
      - run **genhex.py** via command line
      - the result bmp.bin file to be copied into the SD card into the root folder.

      And linux version
      Edit genhex.py and change to:

      1.
         ```
         os.system("python bmp2hex.py -kbin thumb{0:04}.bmp".format(num))
         ```

4. The root folder have to also contain a logo file (logo.bin). This [original](#) can be used.

Insert SD Card into kit, connect USB cable to PC (needs for power only) and you can see result (1100 captured screen in this case).
https://youtu.be/1aanJGTd3_Y

4. **Creating a sequence file from animation**
   I was interested how the moving images made by me will look like.

   1.

   2. I used program [Synfyg studio](#) (open source) for create animation. After I downloaded and   ran it, I made a short animation. Mainly I wanted to display the color spectrum on LCD, so I animate change the background color of the text

   3. I exported the created animation to bmp (the program directly save the sequence of bmp images - yipeee)

   4. At script **genhex.py**, I changed the line:

      1.
         os.system("python.exe bmp2hex.py -kbin {0}.bmp".format(num))

      to

      1.
         os.system("python.exe bmp2hex.py -kbin   {0}{1:04d}.bmp".format("picturename.",num))

      The change was required, because the input files had a slightly unexpected numbering format (0001.bmp vs 1.bmp).

   5. Generated bmp.bin was trasnfered to SD card.

      Result:
      https://youtu.be/EVIbLVxEF9Y

Of course, thanks to the poor quality of my capturing device, the quality of the screen is not very visible on the YouTube.

But in eyes observation, the color quality and viewing angles of this tiny display are excelent.

By the way, I got an unbelievable idea to with this kit...

[Download materials for Longan nano.](#)

Edit 16.11.2019

Andy provided me the procedure tested for Linux. I added to the article. And of course a video of his work (see below).
Thank you Andy.