Building and Flashing a Custom *firmware.bin* for Ringo/Makerphone (Version 1.0 - 19 Sep 2020)

1. Obtain the latest Ringo firmware from github and arrange into the appropriate directory structure (for this example, we'll use release 1.0.5):

    a. Browse to https://github.com/CircuitMess/CircuitMess-Ringo-firmware/tree/v1.0.5 and Download ZIP. The file will be named ***CircuitMess-Ringo-firmware-1.0.5.zip***.

    b. Browse to https://github.com/CircuitMess/CircuitMess-Ringo/tree/v1.0.5 and Download ZIP. The file will be named ***CircuitMess-Ringo-1.0.5.zip***.

    c. Make a new empty directory somewhere to hold the firmware directory structure; for purposes of this document, I'll be a bit disorganized and just put it on my desktop and name it ***CustomRingoFirmware***. Wherever you put this directory, plan to leave it there so that VS Code can find it in the future.

    d. Open ***CircuitMess-Ringo-firmware-1.0.5.zip***, and within that open directory ***CircuitMess-Ringo-firmware-1.0.5.*** You will see a number of files and two directories, ***lib*** and ***src***. Select and copy all those files and directories, and paste them into your empty ***CustomRingoFirmware*** directory.

    e. Open the ***CustomRingoFirmware*** directory, and then the ***lib*** directory, and then the ***MAKERphone*** directory. You will see that it is empty, so let's fill it... Open ***CircuitMess-Ringo-1.0.5.zip***, and within that open directory ***CircuitMess-Ringo-1.0.5***.You will see a number of files and several directories. Select and copy all those files and directories, and paste them into that empty ***CustomRingoFirmware\lib\MAKERphone*** directory.

    f. You may now delete both .zip files, as we have copied all the files into the appropriate directory structure for building the firmware.

    g. OPTIONAL: Make any custom modifications you might want to the various firmware and library source files. I might suggest finding the line: `uint16_t firmware_version = 105;` in the source file ***CustomRingoFirmware\lib\MAKERphone\src\MAKERphone.h*** and change the version number from 105 to 9905. This way you will be able to see that you are running custom firmware once you have flashed it to your Ringo. Use Notepad, Wordpad, or your favorite text editor to make the change(s).

2. Obtain Microsoft VS Code and launch it:

    a. Download the 32-bit portable .zip file from here: https://code.visualstudio.com/Download. Unzip it putting the unzipped directory anywhere you like; in my case, I put it on my desktop. Since I downloaded version 1.49.0, the unzipped directory is named ***VSCode-win32-ia32-1.49.0***. Run VS Code by opening that directory and launching the application ***Code.exe***. The dark IDE window should be displayed.

3. Install the PlatformIO extension to VS Code:

    a. From the VS Code menu, open **View/Extensions**. In the search box, type **PlatformIO**. Under **PlatformIO IDE**, click on **Install**. It will take quite some time for this to complete; watch in the lower right of the dark window for various progress bars. When installation is complete, it will say ***Please restart VSCode***.

    b. Close the dark VS Code IDE window and launch ***Code.exe*** again. Wait until the **PIO Home** tab opens (orange bug looking at you).

4. Create a new project in PlatformIO incorporating the code in your ***CustomRingoFirmware*** directory:

    a. Click on **+ New Project** under **Quck Access**.

    b. Give it a Name: **CustomRingoFirmware** (must be same as the name of the directory).

    c. For Board: type in **WEMOS LOLIN32**.

    d. Framework: should autofill with **Arduino**.

    e. For Location:, uncheck the box **Use default location**.

    f. Under Choose a location where we will create a project folder, navigate to the directory containing the directory **CustomRingoFirmware** (in my case, this was my **Desktop**). Do **not** navigate all the way into the directory **CustomRingoFirmware**.

    g. Click **Finish** and wait until it finishes creating the project. After it has finished, you should see **Desktop\CustomRingoFirmware** (in my case) listed under **Recent Projects**.

5. Build firmware.bin:

    a. Close the **VS Code** window and launch it again.

    b. Wait until the **PI Home** tab opens (orange bug looking at you).

    c. From the VS Code menu, open **Terminal/Run Task**.

    d. Select **PlatformIO**, and then **PlatformIO: Clean CustomRingoFirmware**. This clears any prior build out of the directory structure. Wait until it says **Done cleaning** in the terminal window at the bottom of the dark VS Code IDE window. Ignore the non-zero number listed to the right of Problems; those problems don't apply to this project.

    e. Again from the VS Code menu, open **Terminal/Run Task**.

    f. Select **PlatformIO**, and then **PlatformIO: Build CustomRingoFirmware**. Compiling will commence. This may take a while. Watch the terminal window at the bottom of the dark VS

Code IDE window to see when the build is complete. Hopefully it will say **SUCCESS** at the end.

g. Outside of VS Code, navigate in Windows to directory **CustomRingoFirmware\.pio\build\lolin32**. There you will see the newly generated **firmware.bin** file. This is the end of the process to build **firmware.bin**. You have done it!

h. Repeat steps 5e thru 5g each time you modify any firmware file(s); Repeat steps 5c thru 5g to re-compile all files, even those not dependent on ones you've modified.

i. Instructions to flash firmware.bin will follow. If you are already familiar with flashing firmware.bin using esptool.py, you need go no further in this document.

6. Install Python 3.8 if some version of Python is not already installed on your Windows system (if you already have Python installed, you only need to perform step 6c):

   a. Get the Python 3.8 install file from python.org: https://www.python.org/downloads/release/python-385/. Choose your poison, 32-bit or 64-bit installer.

   b. Install it like installing anything else in Windows but be sure to run the install file as administrator. Choose **Customize installation**, make sure all check boxes are checked under **Optional features**, make sure all check boxes except the last 2 are checked under **Advanced options** and make the install directory be ***C:\Program Files\Python38-32***. Continue with installation and finish it.

   c. Run a DOS box (Command Prompt) as administrator. In this elevated DOS box, enter the command **pip install pyserial**. That should install successfully (though it may complain about an old version of pip). Or it may tell you that pyserial is already installed (*Requirement already satisfied*).

7. Manually flash **firmware.bin** to the Ringo:

   a. Make a new empty directory somewhere to contain the files necessary to flash firmware.bin to the Ringo. I chose to create this on my Desktop and named the directory **FlashCustomRingoFirmware**.

   b. Copy the **firmware.bin** and **partitions.bin** files from **CustomRingoFirmware\.pio\build\lolin32** to your directory **FlashCustomRingoFirmware**.

   c. Browse to this page: https://github.com/espressif/arduino-esp32/releases/tag/1.0.4 and download the file **esp32-1.0.4.zip**. Open that zip file and then the **esp32-1.0.4\tools\sdk\bin** directory and copy the file **bootloader_dio_80m.bin** to your **FlashCustomRingoFirmware** directory. From that same zip file, go to the **esp32-1.0.4\tools\partitions** directory and copy the file **boot_app0.bin** to your **FlashCustomRingoFirmware** directory.

   d. Copy the file ***C:\Users\<your-user-name>\.platformio\packages\tool-esptoolpy\esptool.py*** to your **FlashCustomRingoFirmware** directory.

e. Create a new text file in your **FlashCustomRingoFirmware** directory named **flash.txt**. Open the file with Notepad or your favorite text editor and add the following two lines *(note that due to variations in the way your Python is installed/configured, you may need to change the **py** at the beginning of the first line to one of **python**, **python3**, or **python2**; or you may need to remove the **py** completely; use whatever works for you in step 7h below.)*:

```
py .\esptool.py --chip esp32 --port COM4 --baud 921600
--before default_reset --after hard_reset write_flash
-z --flash_mode dio --flash_freq 80m --flash_size
detect 0xe000 boot_app0.bin 0x1000
bootloader_dio_80m.bin 0x10000 firmware.bin 0x8000
partitions.bin
pause
```

f. Change **COM4** to the COM port that comes up in Device Manager for your Ringo when you plug its USB cable into your computer (that COM port will usually be called ***Silicon Labs CP210x USB to UART Bridge***.

g. Rename **flash.txt** to **flash.bat**.

h. To flash your Ringo, plug its USB cable into your computer and launch **flash.bat**. A DOS box will open and list the progress of the flash. If the flash fails to connect to your Ringo, make sure you've specified the correct COM port in **flash.bat**. If it continues to fail, your computer's USB port may not supply enough voltage/current to do the flash operation; in this case, use a powered USB hub between your computer and the Ringo. If the DOS box won't even run python, see 7e above for possible variations in the batch file command to fix this.

i. That's it, you're done. The Ringo will reboot running the custom firmware.