

Next instead of reading the files and writing the results in dwt.bin The goal is use as serial tx & rx to send the data to the program. Then transmit the result over the serial tx.

The folder testfiles/2048/ has the files to perform a 2048 lifting step.
The folder testfiles/256/ has the files to perform a 256 lifting step.

The following command compiles the code ./buildpi_lift.sh

There is a define in pi_jpeg.c that turns off the debug

```
rm -f dwt.bin ; ./pi_jpeg 0 1
0x0 0x22048 0x1022048
ptrs.fwd_inv = 0x2022060
reading r.bin
fwd lifting step only
w = 0x800 ptrs.inp_buf wptr = 0x22048 alt = 0x1022048 ptrs.fwd_inverse = 0x2022060
ptrs.fwd_inverse = 0x1
starting red dwt
finished ted dwt
octave
GNU Octave, version 4.4.1
Copyright (C) 2018 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.
```

Octave was configured for "arm-unknown-linux-gnueabi".

Additional information about Octave is available at <https://www.octave.org>.

Please contribute if you find this software useful.
For more information, visit <https://www.octave.org/get-involved.html>

Read <https://www.octave.org/bugs.html> to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

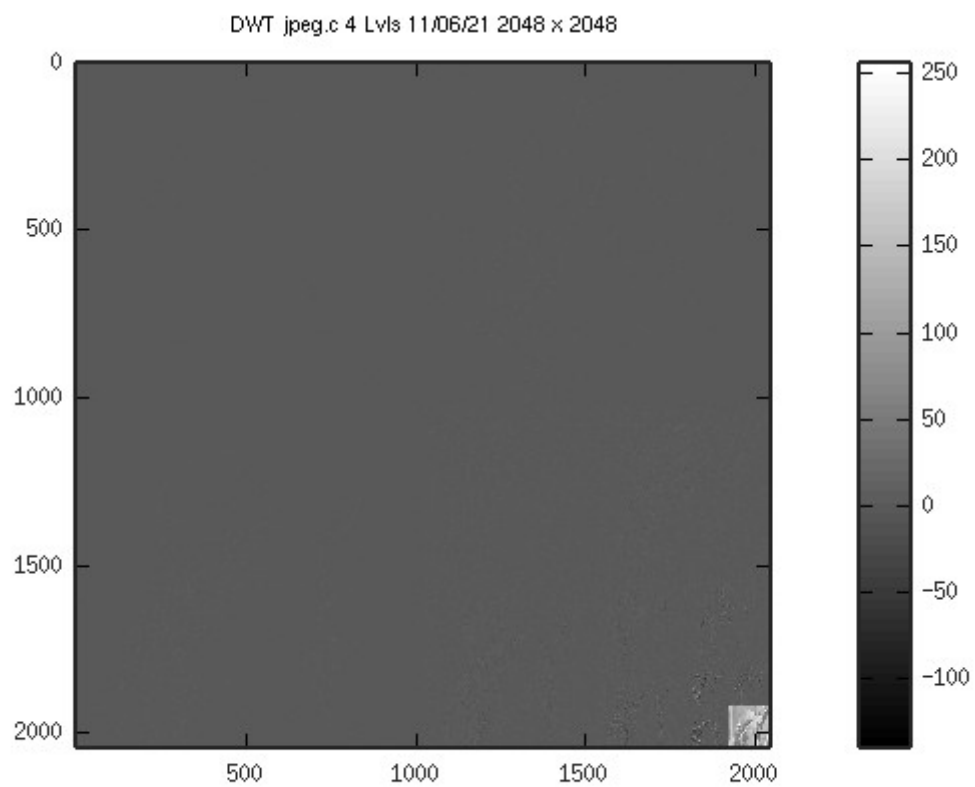
octave:1> rgb

The input was a pgm file 2048 x 2048



$\psi_2 = -4.11346$

4 lvs 2048 x 2048 lifting step.



$\psi_2 = -194.610$

```
cp testfiles/256/* .  
./buildpi_lift.sh  
devel@mypi3-20:~/pico-lifting $ rm -f dwt.bin ; ./pi_jpeg 0 1  
0x0 0x22048 0x62048  
ptrs.fwd_inv = 0xa2060  
reading r.bin  
fwd lifting step only  
w = 0x100 ptrs.inp_buf wptr = 0x22048 alt = 0x62048 ptrs.fwd_inverse = 0xa2060  
ptrs.fwd_inverse = 0x1  
starting red dwt  
finished ted dwt
```

octave
GNU Octave, version 4.4.1
Copyright (C) 2018 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.

Octave was configured for "arm-unknown-linux-gnueabi".

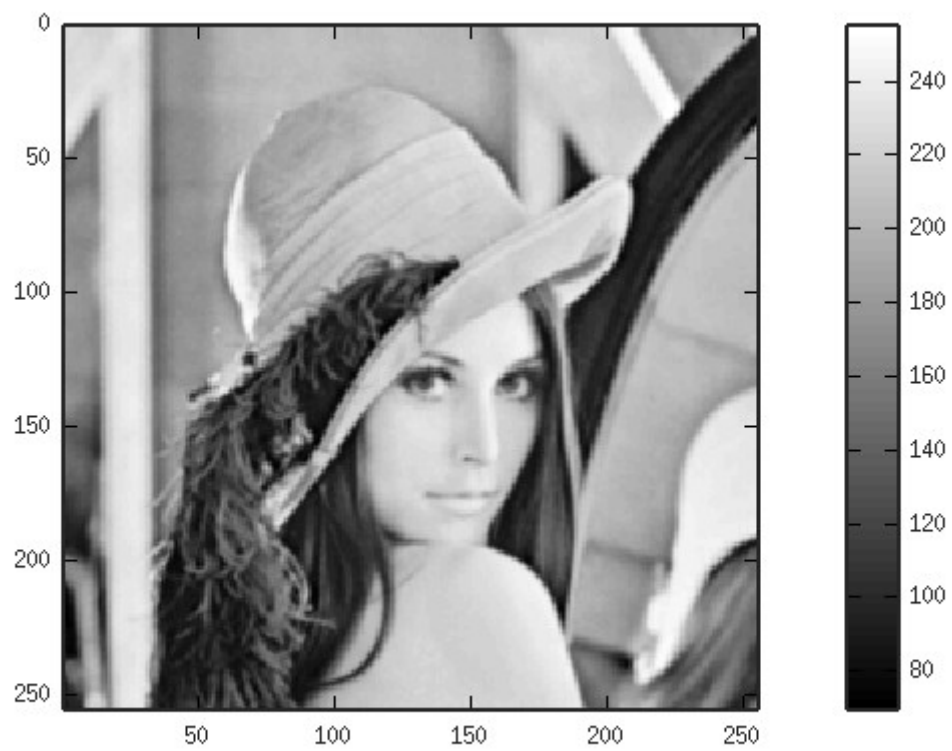
Additional information about Octave is available at <https://www.octave.org>.

Please contribute if you find this software useful.
For more information, visit <https://www.octave.org/get-involved.html>

Read <https://www.octave.org/bugs.html> to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

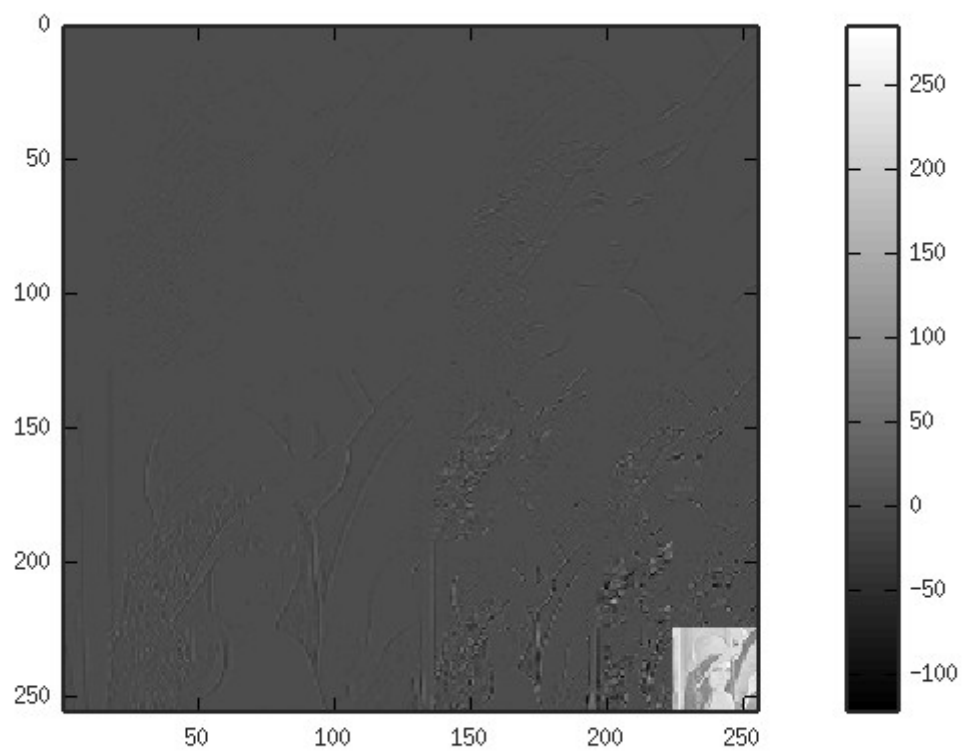
octave:1> rgb

The input was a pgm file 256 x 256



$\psi_2 = 174.001$

3 lvs 256 x 256 lifting step.



$\psi_2 = -178.095$

SWDIO----18 red-----black-----black-----
 grd-----20 yellow----red-----blue-----
 SWCLK---22 orange----orange-----green-----

```
blue----swclk
green---grd
black---swdio
```

Raspberry Pi Pico C/C++

Electronics Hub

LEARN TUTORIALS DIY PRODUCT REVIEWS GUIDES & HOWTO ABOUT

The following table shows all the necessary connections between Raspberry Pi and Raspberry Pi Pico that you need to make.

Raspberry Pi Pico	Raspberry Pi
SWDIO	GPIO 24 (PIN 18)
SWD GND	GND (PIN 20)
SWCLK	GPIO 25 (PIN 22)

How to Program and Debug x Raspberry Pi Pico C/C++ x Getting started with Raspb x +

https://www.electronicshub.org/programming-raspberry-pi-pico-with-swd/


Electronics Hub

LEARN TUTORIALS DIY PRODUCT REVIEWS GUIDES & HOWTO ABOUT

Raspberry Pi Pico SWD Programming and Debug

Like all ARM Cortex processors, the Raspberry Pi Pico also has dedicated hardware for debugging via the SWD Interface. The two wires required for SWD Debugging are called SWDIO (bidirectional SWD Data) and SWCLK (SWD Clock).

On the Raspberry Pi Pico, the SWD Pins are separated from the rest of the GPIO Pins and are placed at the bottom of the Board.



The 2-wire SWD Interface of RP2040 on the Raspberry Pi Pico board allows you to do the following:

- Upload program into External Flash or Internal SRAM.
- Control the state of execution of the processor i.e., run, halt, step, set breakpoints, etc.
- Access processors memory and IO peripherals (which are memory mapped) through the system bus

pico-pi-sdk-build.txt has the instruction from the video below.

Revisit the video below

<https://www.youtube.com/watch?v=UZwq3eb5My0>

Downloaded the getting-started

<https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf>