drwxrwxr-x. 5 aluno aluno 4096 Mar 13 10:12 pkg\_resources

drwxrwxr-x. 6 aluno aluno 4096 Mar 13 10:12 setuptools

drwxrwxr-x. 2 aluno aluno 4096 Mar 13 10:12 setuptools-40.6.2.dist-info

drwxrwxr-x. 5 aluno aluno 4096 Mar 13 10:12 pip

drwxrwxr-x. 2 aluno aluno 4096 Mar 13 10:12 pip-18.1.dist-info

-rw-rw-r--. 1 aluno aluno 34074 Mar 13 10:16 six.py

drwxrwxr-x. 6 aluno aluno 4096 Mar 13 10:16 serial

drwxrwxr-x. 2 aluno aluno 4096 Mar 13 10:16 pyserial-3.4.dist-info

drwxrwxr-x. 3 aluno aluno 4096 Mar 13 10:16 pyaes

drwxrwxr-x. 2 aluno aluno 4096 Mar 13 10:16 pyaes-1.6.1-py3.6.egg-info

drwxrwxr-x. 2 aluno aluno 4096 Mar 13 10:16 six-1.14.0.dist-info

drwxrwxr-x. 3 aluno aluno 4096 Mar 13 10:16 ecdsa

drwxrwxr-x. 2 aluno aluno 4096 Mar 13 10:16 ecdsa-0.15.dist-info

drwxrwxr-x. 2 aluno aluno 4096 Mar 13 10:16 \_\_pycache\_\_

drwxrwxr-x. 2 aluno aluno 4096 Mar 13 10:16 esptool-2.8-py3.6.egg-info

(virtu) [aluno@fleln static]$ cp ./Projeto\_ESP8266/virtualPython/virtu/python3.6/site-packages/esptool.py ./Projeto\_ESP8266/esp-open-sdk/xtensa-lx106-elf/bin/

cp: impossível obter estado de “./Projeto\_ESP8266/virtualPython/virtu/python3.6/site-packages/esptool.py”: Arquivo ou diretório não encontrado

(virtu) [aluno@fleln static]$ cp ./Projeto\_ESP8266/virtualPython/virtu/python3.6/site-packages/esptool.py ./Projeto\_ESP8266/esp\_open\_sdk/xtensa-lx106-elf/bin/

cp: impossível obter estado de “./Projeto\_ESP8266/virtualPython/virtu/python3.6/site-packages/esptool.py”: Arquivo ou diretório não encontrado

(virtu) [aluno@fleln static]$ cp ./Projeto\_ESP8266/virtualPython/virtu/python3.6/site-packages/esptool.py ./Projeto\_ESP8266/esp-open-sdk/xtensa-lx106-elf/bin/^C

(virtu) [aluno@fleln static]$ sudo pip install esptool

Presumimos que você recebeu as instruções de sempre do administrador

de sistema local. Basicamente, resume-se a estas três coisas:

#1) Respeite a privacidade dos outros.

#2) Pense antes de digitar.

#3) Com grandes poderes vêm grandes responsabilidades.

aluno não está no arquivo sudoers. Este incidente será relatado.

(virtu) [aluno@fleln static]$ pip install esptool

Requirement already satisfied: esptool in ./Projeto\_ESP8266/virtualPython/virtu/lib/python3.6/site-packages (2.8)

Requirement already satisfied: pyserial>=3.0 in ./Projeto\_ESP8266/virtualPython/virtu/lib/python3.6/site-packages (from esptool) (3.4)

Requirement already satisfied: pyaes in ./Projeto\_ESP8266/virtualPython/virtu/lib/python3.6/site-packages (from esptool) (1.6.1)

Requirement already satisfied: ecdsa in ./Projeto\_ESP8266/virtualPython/virtu/lib/python3.6/site-packages (from esptool) (0.15)

Requirement already satisfied: six>=1.9.0 in ./Projeto\_ESP8266/virtualPython/virtu/lib/python3.6/site-packages (from ecdsa->esptool) (1.14.0)

You are using pip version 18.1, however version 20.0.2 is available.

You should consider upgrading via the 'pip install --upgrade pip' command.

(virtu) [aluno@fleln static]$ ls

config\_esp8266sdk.sh lost+found mcuxpresso Projeto\_ESP8266 toolchain.sh

(virtu) [aluno@fleln static]$ clear

]

(virtu) [aluno@fleln static]$ sudo pip install esptool

aluno não está no arquivo sudoers. Este incidente será relatado.

(virtu) [aluno@fleln static]$ cp ./Projeto\_ESP8266/virtualPython/virtu/python3.6/site-packages/esptool.py ./Projeto\_ESP8266/esp-open-sdk/xtensa-lx106-elf/bl^C/

(virtu) [aluno@fleln static]$ ^C

(virtu) [aluno@fleln static]$ ls

config\_esp8266sdk.sh lost+found mcuxpresso Projeto\_ESP8266 toolchain.sh

](virtu) [aluno@fleln static]$ cat Projeto\_ESP8266/virtualPython/virtu/python3.6/site-packages/esptool.py

cat: Projeto\_ESP8266/virtualPython/virtu/python3.6/site-packages/esptool.py: Arquivo ou diretório não encontrado

(virtu) [aluno@fleln static]$ cat Projeto\_ESP8266/

esp-open-rtos/ esp-open-sdk/ virtualPython/

(virtu) [aluno@fleln static]$ cat Projeto\_ESP8266/esp-open-

esp-open-rtos/ esp-open-sdk/

(virtu) [aluno@fleln static]$ cat Projeto\_ESP8266/esp-open-sdk/

1000-mforce-l32.patch empty\_user\_rf\_pre\_init.c .gitignore .sdk\_patch\_2.1.0-18-g61248df

crosstool-config-overrides ESP8266\_NONOS\_SDK-2.1.0-18-g61248df/ .gitmodules user\_rf\_cal\_sector\_set.c

crosstool-NG/ esp-open-lwip/ lx106-hal/ user\_rf\_cal\_sector\_set.o

c\_types-c99.patch esptool/ Makefile xtensa-lx106-elf/

c\_types-c99\_sdk\_2.patch examples/ README.md

dhcps\_lease.patch .git/ sdk/

(virtu) [aluno@fleln static]$ cat Projeto\_ESP8266/esp-open-sdk/xtensa-lx106-elf/

bin/ build.log.bz2 include/ lib/ libexec/ share/ xtensa-lx106-elf/

(virtu) [aluno@fleln static]$ cat Projeto\_ESP8266/esp-open-sdk/xtensa-lx106-elf/bin/

esptool.py xtensa-lx106-elf-c++filt xtensa-lx106-elf-gcc-4.8.5 xtensa-lx106-elf-gprof xtensa-lx106-elf-ranlib

xtensa-lx106-elf-addr2line xtensa-lx106-elf-cpp xtensa-lx106-elf-gcc-ar xtensa-lx106-elf-ld xtensa-lx106-elf-readelf

xtensa-lx106-elf-ar xtensa-lx106-elf-ct-ng.config xtensa-lx106-elf-gcc-nm xtensa-lx106-elf-ld.bfd xtensa-lx106-elf-size

xtensa-lx106-elf-as xtensa-lx106-elf-elfedit xtensa-lx106-elf-gcc-ranlib xtensa-lx106-elf-nm xtensa-lx106-elf-strings

xtensa-lx106-elf-c++ xtensa-lx106-elf-g++ xtensa-lx106-elf-gcov xtensa-lx106-elf-objcopy xtensa-lx106-elf-strip

xtensa-lx106-elf-cc xtensa-lx106-elf-gcc xtensa-lx106-elf-gdb xtensa-lx106-elf-objdump

(virtu) [aluno@fleln static]$ cat Projeto\_ESP8266/esp-open-sdk/xtensa-lx106-elf/bin/esptool.py

#!/usr/bin/env python

# NB: Before sending a PR to change the above line to '#!/usr/bin/env python2', please read https://github.com/themadinventor/esptool/issues/21

#

# ESP8266 ROM Bootloader Utility

# https://github.com/themadinventor/esptool

#

# Copyright (C) 2014-2016 Fredrik Ahlberg, Angus Gratton, other contributors as noted.

#

# This program is free software; you can redistribute it and/or modify it under

# the terms of the GNU General Public License as published by the Free Software

# Foundation; either version 2 of the License, or (at your option) any later version.

#

# This program is distributed in the hope that it will be useful, but WITHOUT

# ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS

# FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

#

# You should have received a copy of the GNU General Public License along with

# this program; if not, write to the Free Software Foundation, Inc., 51 Franklin

# Street, Fifth Floor, Boston, MA 02110-1301 USA.

import argparse

import hashlib

import inspect

import json

import os

import serial

import struct

import subprocess

import sys

import tempfile

import time

\_\_version\_\_ = "1.2"

class ESPROM(object):

# These are the currently known commands supported by the ROM

ESP\_FLASH\_BEGIN = 0x02

ESP\_FLASH\_DATA = 0x03

ESP\_FLASH\_END = 0x04

ESP\_MEM\_BEGIN = 0x05

ESP\_MEM\_END = 0x06

ESP\_MEM\_DATA = 0x07

ESP\_SYNC = 0x08

ESP\_WRITE\_REG = 0x09

ESP\_READ\_REG = 0x0a

# Maximum block sized for RAM and Flash writes, respectively.

ESP\_RAM\_BLOCK = 0x1800

ESP\_FLASH\_BLOCK = 0x400

# Default baudrate. The ROM auto-bauds, so we can use more or less whatever we want.

ESP\_ROM\_BAUD = 115200

# First byte of the application image

ESP\_IMAGE\_MAGIC = 0xe9

# Initial state for the checksum routine

ESP\_CHECKSUM\_MAGIC = 0xef

# OTP ROM addresses

ESP\_OTP\_MAC0 = 0x3ff00050

ESP\_OTP\_MAC1 = 0x3ff00054

ESP\_OTP\_MAC3 = 0x3ff0005c

# Flash sector size, minimum unit of erase.

ESP\_FLASH\_SECTOR = 0x1000

def \_\_init\_\_(self, port=0, baud=ESP\_ROM\_BAUD):

self.\_port = serial.serial\_for\_url(port)

self.\_slip\_reader = slip\_reader(self.\_port)

# setting baud rate in a separate step is a workaround for

# CH341 driver on some Linux versions (this opens at 9600 then

# sets), shouldn't matter for other platforms/drivers. See

# https://github.com/themadinventor/esptool/issues/44#issuecomment-107094446

self.\_port.baudrate = baud

""" Read a SLIP packet from the serial port """

def read(self):

return self.\_slip\_reader.next()

""" Write bytes to the serial port while performing SLIP escaping """

def write(self, packet):

buf = '\xc0' \

+ (packet.replace('\xdb','\xdb\xdd').replace('\xc0','\xdb\xdc')) \

+ '\xc0'

self.\_port.write(buf)

""" Calculate checksum of a blob, as it is defined by the ROM """

@staticmethod

def checksum(data, state=ESP\_CHECKSUM\_MAGIC):

for b in data:

state ^= ord(b)

return state

""" Send a request and read the response """

def command(self, op=None, data=None, chk=0):

if op is not None:

pkt = struct.pack('<BBHI', 0x00, op, len(data), chk) + data

self.write(pkt)

# tries to get a response until that response has the

# same operation as the request or a retries limit has

# exceeded. This is needed for some esp8266s that

# reply with more sync responses than expected.

for retry in xrange(100):

p = self.read()

if len(p) < 8:

continue

(resp, op\_ret, len\_ret, val) = struct.unpack('<BBHI', p[:8])

if resp != 1:

continue

body = p[8:]

if op is None or op\_ret == op:

return val, body # valid response received

raise FatalError("Response doesn't match request")

""" Perform a connection test """

def sync(self):

self.command(ESPROM.ESP\_SYNC, '\x07\x07\x12\x20' + 32 \* '\x55')

for i in xrange(7):

self.command()

""" Try connecting repeatedly until successful, or giving up """

def connect(self):

print 'Connecting...'

for \_ in xrange(4):

# issue reset-to-bootloader:

# RTS = either CH\_PD or nRESET (both active low = chip in reset)

# DTR = GPIO0 (active low = boot to flasher)

self.\_port.setDTR(False)

self.\_port.setRTS(True)

time.sleep(0.05)

self.\_port.setDTR(True)

self.\_port.setRTS(False)

time.sleep(0.05)

self.\_port.setDTR(False)

# worst-case latency timer should be 255ms (probably <20ms)

self.\_port.timeout = 0.3

for \_ in xrange(4):

try:

self.\_port.flushInput()

self.\_slip\_reader = slip\_reader(self.\_port)

self.\_port.flushOutput()

self.sync()

self.\_port.timeout = 5

return

except:

time.sleep(0.05)

raise FatalError('Failed to connect to ESP8266')

""" Read memory address in target """

def read\_reg(self, addr):

res = self.command(ESPROM.ESP\_READ\_REG, struct.pack('<I', addr))

if res[1] != "\0\0":

raise FatalError('Failed to read target memory')

return res[0]

""" Write to memory address in target """

def write\_reg(self, addr, value, mask, delay\_us=0):

if self.command(ESPROM.ESP\_WRITE\_REG,

struct.pack('<IIII', addr, value, mask, delay\_us))[1] != "\0\0":

raise FatalError('Failed to write target memory')

""" Start downloading an application image to RAM """

def mem\_begin(self, size, blocks, blocksize, offset):

if self.command(ESPROM.ESP\_MEM\_BEGIN,

struct.pack('<IIII', size, blocks, blocksize, offset))[1] != "\0\0":

raise FatalError('Failed to enter RAM download mode')

""" Send a block of an image to RAM """

def mem\_block(self, data, seq):

if self.command(ESPROM.ESP\_MEM\_DATA,

struct.pack('<IIII', len(data), seq, 0, 0) + data,

ESPROM.checksum(data))[1] != "\0\0":

raise FatalError('Failed to write to target RAM')

""" Leave download mode and run the application """

def mem\_finish(self, entrypoint=0):

if self.command(ESPROM.ESP\_MEM\_END,

struct.pack('<II', int(entrypoint == 0), entrypoint))[1] != "\0\0":

raise FatalError('Failed to leave RAM download mode')

""" Start downloading to Flash (performs an erase) """

def flash\_begin(self, size, offset):

old\_tmo = self.\_port.timeout

num\_blocks = (size + ESPROM.ESP\_FLASH\_BLOCK - 1) / ESPROM.ESP\_FLASH\_BLOCK

sectors\_per\_block = 16

sector\_size = self.ESP\_FLASH\_SECTOR

num\_sectors = (size + sector\_size - 1) / sector\_size

start\_sector = offset / sector\_size

head\_sectors = sectors\_per\_block - (start\_sector % sectors\_per\_block)

if num\_sectors < head\_sectors:

head\_sectors = num\_sectors

if num\_sectors < 2 \* head\_sectors:

erase\_size = (num\_sectors + 1) / 2 \* sector\_size

else:

erase\_size = (num\_sectors - head\_sectors) \* sector\_size

self.\_port.timeout = 20

t = time.time()

result = self.command(ESPROM.ESP\_FLASH\_BEGIN,

struct.pack('<IIII', erase\_size, num\_blocks, ESPROM.ESP\_FLASH\_BLOCK, offset))[1]

if size != 0:

print "Took %.2fs to erase flash block" % (time.time() - t)

if result != "\0\0":

raise FatalError.WithResult('Failed to enter Flash download mode (result "%s")', result)

self.\_port.timeout = old\_tmo

""" Write block to flash """

def flash\_block(self, data, seq):

result = self.command(ESPROM.ESP\_FLASH\_DATA,

struct.pack('<IIII', len(data), seq, 0, 0) + data,

ESPROM.checksum(data))[1]

if result != "\0\0":

raise FatalError.WithResult('Failed to write to target Flash after seq %d (got result %%s)' % seq, result)

""" Leave flash mode and run/reboot """

def flash\_finish(self, reboot=False):

pkt = struct.pack('<I', int(not reboot))

if self.command(ESPROM.ESP\_FLASH\_END, pkt)[1] != "\0\0":

raise FatalError('Failed to leave Flash mode')

""" Run application code in flash """

def run(self, reboot=False):

# Fake flash begin immediately followed by flash end

self.flash\_begin(0, 0)

self.flash\_finish(reboot)

""" Read MAC from OTP ROM """

def read\_mac(self):

mac0 = self.read\_reg(self.ESP\_OTP\_MAC0)

mac1 = self.read\_reg(self.ESP\_OTP\_MAC1)

mac3 = self.read\_reg(self.ESP\_OTP\_MAC3)

if (mac3 != 0):

oui = ((mac3 >> 16) & 0xff, (mac3 >> 8) & 0xff, mac3 & 0xff)

elif ((mac1 >> 16) & 0xff) == 0:

oui = (0x18, 0xfe, 0x34)

elif ((mac1 >> 16) & 0xff) == 1:

oui = (0xac, 0xd0, 0x74)

else:

raise FatalError("Unknown OUI")

return oui + ((mac1 >> 8) & 0xff, mac1 & 0xff, (mac0 >> 24) & 0xff)

""" Read Chip ID from OTP ROM - see http://esp8266-re.foogod.com/wiki/System\_get\_chip\_id\_%28IoT\_RTOS\_SDK\_0.9.9%29 """

def chip\_id(self):

id0 = self.read\_reg(self.ESP\_OTP\_MAC0)

id1 = self.read\_reg(self.ESP\_OTP\_MAC1)

return (id0 >> 24) | ((id1 & 0xffffff) << 8)

""" Read SPI flash manufacturer and device id """

def flash\_id(self):

self.flash\_begin(0, 0)

self.write\_reg(0x60000240, 0x0, 0xffffffff)

self.write\_reg(0x60000200, 0x10000000, 0xffffffff)

flash\_id = self.read\_reg(0x60000240)

return flash\_id

""" Abuse the loader protocol to force flash to be left in write mode """

def flash\_unlock\_dio(self):

# Enable flash write mode

self.flash\_begin(0, 0)

# Reset the chip rather than call flash\_finish(), which would have

# write protected the chip again (why oh why does it do that?!)

self.mem\_begin(0,0,0,0x40100000)

self.mem\_finish(0x40000080)

""" Perform a chip erase of SPI flash """

def flash\_erase(self):

# Trick ROM to initialize SFlash

self.flash\_begin(0, 0)

# This is hacky: we don't have a custom stub, instead we trick

# the bootloader to jump to the SPIEraseChip() routine and then halt/crash

# when it tries to boot an unconfigured system.

self.mem\_begin(0,0,0,0x40100000)

self.mem\_finish(0x40004984)

# Yup - there's no good way to detect if we succeeded.

# It it on the other hand unlikely to fail.

def run\_stub(self, stub, params, read\_output=True):

stub = dict(stub)

stub['code'] = unhexify(stub['code'])

if 'data' in stub:

stub['data'] = unhexify(stub['data'])

if stub['num\_params'] != len(params):

raise FatalError('Stub requires %d params, %d provided'

% (stub['num\_params'], len(params)))

params = struct.pack('<' + ('I' \* stub['num\_params']), \*params)

pc = params + stub['code']

# Upload

self.mem\_begin(len(pc), 1, len(pc), stub['params\_start'])

self.mem\_block(pc, 0)

if 'data' in stub:

self.mem\_begin(len(stub['data']), 1, len(stub['data']), stub['data\_start'])

self.mem\_block(stub['data'], 0)

self.mem\_finish(stub['entry'])

if read\_output:

print 'Stub executed, reading response:'

while True:

p = self.read()

print hexify(p)

if p == '':

return

class ESPBOOTLOADER(object):

""" These are constants related to software ESP bootloader, working with 'v2' image files """

# First byte of the "v2" application image

IMAGE\_V2\_MAGIC = 0xea

# First 'segment' value in a "v2" application image, appears to be a constant version value?

IMAGE\_V2\_SEGMENT = 4

def LoadFirmwareImage(filename):

""" Load a firmware image, without knowing what kind of file (v1 or v2) it is.

Returns a BaseFirmwareImage subclass, either ESPFirmwareImage (v1) or OTAFirmwareImage (v2).

"""

with open(filename, 'rb') as f:

magic = ord(f.read(1))

f.seek(0)

if magic == ESPROM.ESP\_IMAGE\_MAGIC:

return ESPFirmwareImage(f)

elif magic == ESPBOOTLOADER.IMAGE\_V2\_MAGIC:

return OTAFirmwareImage(f)

else:

raise FatalError("Invalid image magic number: %d" % magic)

class BaseFirmwareImage(object):

""" Base class with common firmware image functions """

def \_\_init\_\_(self):

self.segments = []

self.entrypoint = 0

def add\_segment(self, addr, data, pad\_to=4):

""" Add a segment to the image, with specified address & data

(padded to a boundary of pad\_to size) """

# Data should be aligned on word boundary

l = len(data)

if l % pad\_to:

data += b"\x00" \* (pad\_to - l % pad\_to)

if l > 0:

self.segments.append((addr, len(data), data))

def load\_segment(self, f, is\_irom\_segment=False):

""" Load the next segment from the image file """

(offset, size) = struct.unpack('<II', f.read(8))

if not is\_irom\_segment:

if offset > 0x40200000 or offset < 0x3ffe0000 or size > 65536:

raise FatalError('Suspicious segment 0x%x, length %d' % (offset, size))

segment\_data = f.read(size)

if len(segment\_data) < size:

raise FatalError('End of file reading segment 0x%x, length %d (actual length %d)' % (offset, size, len(segment\_data)))

segment = (offset, size, segment\_data)

self.segments.append(segment)

return segment

def save\_segment(self, f, segment, checksum=None):

""" Save the next segment to the image file, return next checksum value if provided """

(offset, size, data) = segment

f.write(struct.pack('<II', offset, size))

f.write(data)

if checksum is not None:

return ESPROM.checksum(data, checksum)

def read\_checksum(self, f):

""" Return ESPROM checksum from end of just-read image """

# Skip the padding. The checksum is stored in the last byte so that the

# file is a multiple of 16 bytes.

align\_file\_position(f, 16)

return ord(f.read(1))

def append\_checksum(self, f, checksum):

""" Append ESPROM checksum to the just-written image """

align\_file\_position(f, 16)

f.write(struct.pack('B', checksum))

def write\_v1\_header(self, f, segments):

f.write(struct.pack('<BBBBI', ESPROM.ESP\_IMAGE\_MAGIC, len(segments),

self.flash\_mode, self.flash\_size\_freq, self.entrypoint))

class ESPFirmwareImage(BaseFirmwareImage):

""" 'Version 1' firmware image, segments loaded directly by the ROM bootloader. """

def \_\_init\_\_(self, load\_file=None):

super(ESPFirmwareImage, self).\_\_init\_\_()

self.flash\_mode = 0

self.flash\_size\_freq = 0

self.version = 1

if load\_file is not None:

(magic, segments, self.flash\_mode, self.flash\_size\_freq, self.entrypoint) = struct.unpack('<BBBBI', load\_file.read(8))

# some sanity check

if magic != ESPROM.ESP\_IMAGE\_MAGIC or segments > 16:

raise FatalError('Invalid firmware image magic=%d segments=%d' % (magic, segments))

for i in xrange(segments):

self.load\_segment(load\_file)

self.checksum = self.read\_checksum(load\_file)

def save(self, filename):

with open(filename, 'wb') as f:

self.write\_v1\_header(f, self.segments)

checksum = ESPROM.ESP\_CHECKSUM\_MAGIC

for segment in self.segments:

checksum = self.save\_segment(f, segment, checksum)

self.append\_checksum(f, checksum)

class OTAFirmwareImage(BaseFirmwareImage):

""" 'Version 2' firmware image, segments loaded by software bootloader stub

(ie Espressif bootloader or rboot)

"""

def \_\_init\_\_(self, load\_file=None):

super(OTAFirmwareImage, self).\_\_init\_\_()

self.version = 2

if load\_file is not None:

(magic, segments, first\_flash\_mode, first\_flash\_size\_freq, first\_entrypoint) = struct.unpack('<BBBBI', load\_file.read(8))

# some sanity check

if magic != ESPBOOTLOADER.IMAGE\_V2\_MAGIC:

raise FatalError('Invalid V2 image magic=%d' % (magic))

if segments != 4:

# segment count is not really segment count here, but we expect to see '4'

print 'Warning: V2 header has unexpected "segment" count %d (usually 4)' % segments

# irom segment comes before the second header

self.load\_segment(load\_file, True)

(magic, segments, self.flash\_mode, self.flash\_size\_freq, self.entrypoint) = struct.unpack('<BBBBI', load\_file.read(8))

if first\_flash\_mode != self.flash\_mode:

print('WARNING: Flash mode value in first header (0x%02x) disagrees with second (0x%02x). Using second value.'

% (first\_flash\_mode, self.flash\_mode))

if first\_flash\_size\_freq != self.flash\_size\_freq:

print('WARNING: Flash size/freq value in first header (0x%02x) disagrees with second (0x%02x). Using second value.'

% (first\_flash\_size\_freq, self.flash\_size\_freq))

if first\_entrypoint != self.entrypoint:

print('WARNING: Enterypoint address in first header (0x%08x) disagrees with second header (0x%08x). Using second value.'

% (first\_entrypoint, self.entrypoint))

if magic != ESPROM.ESP\_IMAGE\_MAGIC or segments > 16:

raise FatalError('Invalid V2 second header magic=%d segments=%d' % (magic, segments))

# load all the usual segments

for \_ in xrange(segments):

self.load\_segment(load\_file)

self.checksum = self.read\_checksum(load\_file)

def save(self, filename):

with open(filename, 'wb') as f:

# Save first header for irom0 segment

f.write(struct.pack('<BBBBI', ESPBOOTLOADER.IMAGE\_V2\_MAGIC, ESPBOOTLOADER.IMAGE\_V2\_SEGMENT,

self.flash\_mode, self.flash\_size\_freq, self.entrypoint))

# irom0 segment identified by load address zero

irom\_segments = [segment for segment in self.segments if segment[0] == 0]

if len(irom\_segments) != 1:

raise FatalError('Found %d segments that could be irom0. Bad ELF file?' % len(irom\_segments))

# save irom0 segment

irom\_segment = irom\_segments[0]

self.save\_segment(f, irom\_segment)

# second header, matches V1 header and contains loadable segments

normal\_segments = [s for s in self.segments if s != irom\_segment]

self.write\_v1\_header(f, normal\_segments)

checksum = ESPROM.ESP\_CHECKSUM\_MAGIC

for segment in normal\_segments:

checksum = self.save\_segment(f, segment, checksum)

self.append\_checksum(f, checksum)

class ELFFile(object):

def \_\_init\_\_(self, name):

self.name = binutils\_safe\_path(name)

self.symbols = None

def \_fetch\_symbols(self):

if self.symbols is not None:

return

self.symbols = {}

try:

tool\_nm = "xtensa-lx106-elf-nm"

if os.getenv('XTENSA\_CORE') == 'lx106':

tool\_nm = "xt-nm"

proc = subprocess.Popen([tool\_nm, self.name], stdout=subprocess.PIPE)

except OSError:

print "Error calling %s, do you have Xtensa toolchain in PATH?" % tool\_nm

sys.exit(1)

for l in proc.stdout:

fields = l.strip().split()

try:

if fields[0] == "U":

print "Warning: ELF binary has undefined symbol %s" % fields[1]

continue

if fields[0] == "w":

continue # can skip weak symbols

self.symbols[fields[2]] = int(fields[0], 16)

except ValueError:

raise FatalError("Failed to strip symbol output from nm: %s" % fields)

def get\_symbol\_addr(self, sym):

self.\_fetch\_symbols()

return self.symbols[sym]

def get\_entry\_point(self):

tool\_readelf = "xtensa-lx106-elf-readelf"

if os.getenv('XTENSA\_CORE') == 'lx106':

tool\_readelf = "xt-readelf"

try:

proc = subprocess.Popen([tool\_readelf, "-h", self.name], stdout=subprocess.PIPE)

except OSError:

print "Error calling %s, do you have Xtensa toolchain in PATH?" % tool\_readelf

sys.exit(1)

for l in proc.stdout:

fields = l.strip().split()

if fields[0] == "Entry":

return int(fields[3], 0)

def load\_section(self, section):

tool\_objcopy = "xtensa-lx106-elf-objcopy"

if os.getenv('XTENSA\_CORE') == 'lx106':

tool\_objcopy = "xt-objcopy"

tmpsection = binutils\_safe\_path(tempfile.mktemp(suffix=".section"))

try:

subprocess.check\_call([tool\_objcopy, "--only-section", section, "-Obinary", self.name, tmpsection])

with open(tmpsection, "rb") as f:

data = f.read()

finally:

os.remove(tmpsection)

return data

class CesantaFlasher(object):

# From stub\_flasher.h

CMD\_FLASH\_WRITE = 1

CMD\_FLASH\_READ = 2

CMD\_FLASH\_DIGEST = 3

CMD\_FLASH\_ERASE\_CHIP = 5

CMD\_BOOT\_FW = 6

def \_\_init\_\_(self, esp, baud\_rate=0):

print 'Running Cesanta flasher stub...'

if baud\_rate <= ESPROM.ESP\_ROM\_BAUD: # don't change baud rates if we already synced at that rate

baud\_rate = 0

self.\_esp = esp

esp.run\_stub(json.loads(\_CESANTA\_FLASHER\_STUB), [baud\_rate], read\_output=False)

if baud\_rate > 0:

esp.\_port.baudrate = baud\_rate

# Read the greeting.

p = esp.read()

if p != 'OHAI':

raise FatalError('Failed to connect to the flasher (got %s)' % hexify(p))

def flash\_write(self, addr, data, show\_progress=False):

assert addr % self.\_esp.ESP\_FLASH\_SECTOR == 0, 'Address must be sector-aligned'

assert len(data) % self.\_esp.ESP\_FLASH\_SECTOR == 0, 'Length must be sector-aligned'

sys.stdout.write('Writing %d @ 0x%x... ' % (len(data), addr))

sys.stdout.flush()

self.\_esp.write(struct.pack('<B', self.CMD\_FLASH\_WRITE))

self.\_esp.write(struct.pack('<III', addr, len(data), 1))

num\_sent, num\_written = 0, 0

while num\_written < len(data):

p = self.\_esp.read()

if len(p) == 4:

num\_written = struct.unpack('<I', p)[0]

elif len(p) == 1:

status\_code = struct.unpack('<B', p)[0]

raise FatalError('Write failure, status: %x' % status\_code)

else:

raise FatalError('Unexpected packet while writing: %s' % hexify(p))

if show\_progress:

progress = '%d (%d %%)' % (num\_written, num\_written \* 100.0 / len(data))

sys.stdout.write(progress + '\b' \* len(progress))

sys.stdout.flush()

while num\_sent - num\_written < 5120:

self.\_esp.\_port.write(data[num\_sent:num\_sent + 1024])

num\_sent += 1024

p = self.\_esp.read()

if len(p) != 16:

raise FatalError('Expected digest, got: %s' % hexify(p))

digest = hexify(p).upper()

expected\_digest = hashlib.md5(data).hexdigest().upper()

print

if digest != expected\_digest:

raise FatalError('Digest mismatch: expected %s, got %s' % (expected\_digest, digest))

p = self.\_esp.read()

if len(p) != 1:

raise FatalError('Expected status, got: %s' % hexify(p))

status\_code = struct.unpack('<B', p)[0]

if status\_code != 0:

raise FatalError('Write failure, status: %x' % status\_code)

def flash\_read(self, addr, length, show\_progress=False):

sys.stdout.write('Reading %d @ 0x%x... ' % (length, addr))

sys.stdout.flush()

self.\_esp.write(struct.pack('<B', self.CMD\_FLASH\_READ))

# USB may not be able to keep up with the read rate, especially at

# higher speeds. Since we don't have flow control, this will result in

# data loss. Hence, we use small packet size and only allow small

# number of bytes in flight, which we can reasonably expect to fit in

# the on-chip FIFO. max\_in\_flight = 64 works for CH340G, other chips may

# have longer FIFOs and could benefit from increasing max\_in\_flight.

self.\_esp.write(struct.pack('<IIII', addr, length, 32, 64))

data = ''

while True:

p = self.\_esp.read()

data += p

self.\_esp.write(struct.pack('<I', len(data)))

if show\_progress and (len(data) % 1024 == 0 or len(data) == length):

progress = '%d (%d %%)' % (len(data), len(data) \* 100.0 / length)

sys.stdout.write(progress + '\b' \* len(progress))

sys.stdout.flush()

if len(data) == length:

break

if len(data) > length:

raise FatalError('Read more than expected')

p = self.\_esp.read()

if len(p) != 16:

raise FatalError('Expected digest, got: %s' % hexify(p))

expected\_digest = hexify(p).upper()

digest = hashlib.md5(data).hexdigest().upper()

print

if digest != expected\_digest:

raise FatalError('Digest mismatch: expected %s, got %s' % (expected\_digest, digest))

p = self.\_esp.read()

if len(p) != 1:

raise FatalError('Expected status, got: %s' % hexify(p))

status\_code = struct.unpack('<B', p)[0]

if status\_code != 0:

raise FatalError('Write failure, status: %x' % status\_code)

return data

def flash\_digest(self, addr, length, digest\_block\_size=0):

self.\_esp.write(struct.pack('<B', self.CMD\_FLASH\_DIGEST))

self.\_esp.write(struct.pack('<III', addr, length, digest\_block\_size))

digests = []

while True:

p = self.\_esp.read()

if len(p) == 16:

digests.append(p)

elif len(p) == 1:

status\_code = struct.unpack('<B', p)[0]

if status\_code != 0:

raise FatalError('Write failure, status: %x' % status\_code)

break

else:

raise FatalError('Unexpected packet: %s' % hexify(p))

return digests[-1], digests[:-1]

def boot\_fw(self):

self.\_esp.write(struct.pack('<B', self.CMD\_BOOT\_FW))

p = self.\_esp.read()

if len(p) != 1:

raise FatalError('Expected status, got: %s' % hexify(p))

status\_code = struct.unpack('<B', p)[0]

if status\_code != 0:

raise FatalError('Boot failure, status: %x' % status\_code)

def flash\_erase\_chip(self):

self.\_esp.write(struct.pack('<B', self.CMD\_FLASH\_ERASE\_CHIP))

otimeout = self.\_esp.\_port.timeout

self.\_esp.\_port.timeout = 60

p = self.\_esp.read()

self.\_esp.\_port.timeout = otimeout

if len(p) != 1:

raise FatalError('Expected status, got: %s' % hexify(p))

status\_code = struct.unpack('<B', p)[0]

if status\_code != 0:

raise FatalError('Erase chip failure, status: %x' % status\_code)

def slip\_reader(port):

"""Generator to read SLIP packets from a serial port.

Yields one full SLIP packet at a time, raises exception on timeout or invalid data.

Designed to avoid too many calls to serial.read(1), which can bog

down on slow systems.

"""

partial\_packet = None

in\_escape = False

while True:

waiting = port.inWaiting()

read\_bytes = port.read(1 if waiting == 0 else waiting)

if read\_bytes == '':

raise FatalError("Timed out waiting for packet %s" % ("header" if partial\_packet is None else "content"))

for b in read\_bytes:

if partial\_packet is None: # waiting for packet header

if b == '\xc0':

partial\_packet = ""

else:

raise FatalError('Invalid head of packet (%r)' % b)

elif in\_escape: # part-way through escape sequence

in\_escape = False

if b == '\xdc':

partial\_packet += '\xc0'

elif b == '\xdd':

partial\_packet += '\xdb'

else:

raise FatalError('Invalid SLIP escape (%r%r)' % ('\xdb', b))

elif b == '\xdb': # start of escape sequence

in\_escape = True

elif b == '\xc0': # end of packet

yield partial\_packet

partial\_packet = None

else: # normal byte in packet

partial\_packet += b

def arg\_auto\_int(x):

return int(x, 0)

def div\_roundup(a, b):

""" Return a/b rounded up to nearest integer,

equivalent result to int(math.ceil(float(int(a)) / float(int(b))), only

without possible floating point accuracy errors.

"""

return (int(a) + int(b) - 1) / int(b)

def binutils\_safe\_path(p):

"""Returns a 'safe' version of path 'p' to pass to binutils

Only does anything under Cygwin Python, where cygwin paths need to

be translated to Windows paths if the binutils wasn't compiled

using Cygwin (should also work with binutils compiled using

Cygwin, see #73.)

"""

if sys.platform == "cygwin":

try:

return subprocess.check\_output(["cygpath", "-w", p]).rstrip('\n')

except subprocess.CalledProcessError:

print "WARNING: Failed to call cygpath to sanitise Cygwin path."

return p

def align\_file\_position(f, size):

""" Align the position in the file to the next block of specified size """

align = (size - 1) - (f.tell() % size)

f.seek(align, 1)

def hexify(s):

return ''.join('%02X' % ord(c) for c in s)

def unhexify(hs):

s = ''

for i in range(0, len(hs) - 1, 2):

s += chr(int(hs[i] + hs[i + 1], 16))

return s

class FatalError(RuntimeError):

"""

Wrapper class for runtime errors that aren't caused by internal bugs, but by

ESP8266 responses or input content.

"""

def \_\_init\_\_(self, message):

RuntimeError.\_\_init\_\_(self, message)

@staticmethod

def WithResult(message, result):

"""

Return a fatal error object that includes the hex values of

'result' as a string formatted argument.

"""

return FatalError(message % ", ".join(hex(ord(x)) for x in result))

# "Operation" commands, executable at command line. One function each

#

# Each function takes either two args (<ESPROM instance>, <args>) or a single <args>

# argument.

def load\_ram(esp, args):

image = LoadFirmwareImage(args.filename)

print 'RAM boot...'

for (offset, size, data) in image.segments:

print 'Downloading %d bytes at %08x...' % (size, offset),

sys.stdout.flush()

esp.mem\_begin(size, div\_roundup(size, esp.ESP\_RAM\_BLOCK), esp.ESP\_RAM\_BLOCK, offset)

seq = 0

while len(data) > 0:

esp.mem\_block(data[0:esp.ESP\_RAM\_BLOCK], seq)

data = data[esp.ESP\_RAM\_BLOCK:]

seq += 1

print 'done!'

print 'All segments done, executing at %08x' % image.entrypoint

esp.mem\_finish(image.entrypoint)

def read\_mem(esp, args):

print '0x%08x = 0x%08x' % (args.address, esp.read\_reg(args.address))

def write\_mem(esp, args):

esp.write\_reg(args.address, args.value, args.mask, 0)

print 'Wrote %08x, mask %08x to %08x' % (args.value, args.mask, args.address)

def dump\_mem(esp, args):

f = file(args.filename, 'wb')

for i in xrange(args.size / 4):

d = esp.read\_reg(args.address + (i \* 4))

f.write(struct.pack('<I', d))

if f.tell() % 1024 == 0:

print '\r%d bytes read... (%d %%)' % (f.tell(),

f.tell() \* 100 / args.size),

sys.stdout.flush()

print 'Done!'

def detect\_flash\_size(esp, args):

if args.flash\_size == 'detect':

flash\_id = esp.flash\_id()

size\_id = flash\_id >> 16

args.flash\_size = {18: '2m', 19: '4m', 20: '8m', 21: '16m', 22: '32m'}.get(size\_id)

if args.flash\_size is None:

print 'Warning: Could not auto-detect Flash size (FlashID=0x%x, SizeID=0x%x), defaulting to 4m' % (flash\_id, size\_id)

args.flash\_size = '4m'

else:

print 'Auto-detected Flash size:', args.flash\_size

def write\_flash(esp, args):

detect\_flash\_size(esp, args)

flash\_mode = {'qio':0, 'qout':1, 'dio':2, 'dout': 3}[args.flash\_mode]

flash\_size\_freq = {'4m':0x00, '2m':0x10, '8m':0x20, '16m':0x30, '32m':0x40, '16m-c1': 0x50, '32m-c1':0x60, '32m-c2':0x70}[args.flash\_size]

flash\_size\_freq += {'40m':0, '26m':1, '20m':2, '80m': 0xf}[args.flash\_freq]

flash\_params = struct.pack('BB', flash\_mode, flash\_size\_freq)

flasher = CesantaFlasher(esp, args.baud)

for address, argfile in args.addr\_filename:

image = argfile.read()

argfile.seek(0) # rewind in case we need it again

if address + len(image) > int(args.flash\_size.split('m')[0]) \* (1 << 17):

print 'WARNING: Unlikely to work as data goes beyond end of flash. Hint: Use --flash\_size'

# Fix sflash config data.

if address == 0 and image[0] == '\xe9':

print 'Flash params set to 0x%02x%02x' % (flash\_mode, flash\_size\_freq)

image = image[0:2] + flash\_params + image[4:]

# Pad to sector size, which is the minimum unit of writing (erasing really).

if len(image) % esp.ESP\_FLASH\_SECTOR != 0:

image += '\xff' \* (esp.ESP\_FLASH\_SECTOR - (len(image) % esp.ESP\_FLASH\_SECTOR))

t = time.time()

flasher.flash\_write(address, image, not args.no\_progress)

t = time.time() - t

print ('\rWrote %d bytes at 0x%x in %.1f seconds (%.1f kbit/s)...'

% (len(image), address, t, len(image) / t \* 8 / 1000))

print 'Leaving...'

if args.verify:

print 'Verifying just-written flash...'

\_verify\_flash(flasher, args, flash\_params)

flasher.boot\_fw()

def image\_info(args):

image = LoadFirmwareImage(args.filename)

print('Image version: %d' % image.version)

print('Entry point: %08x' % image.entrypoint) if image.entrypoint != 0 else 'Entry point not set'

print '%d segments' % len(image.segments)

print

checksum = ESPROM.ESP\_CHECKSUM\_MAGIC

for (idx, (offset, size, data)) in enumerate(image.segments):

if image.version == 2 and idx == 0:

print 'Segment 1: %d bytes IROM0 (no load address)' % size

else:

print 'Segment %d: %5d bytes at %08x' % (idx + 1, size, offset)

checksum = ESPROM.checksum(data, checksum)

print

print 'Checksum: %02x (%s)' % (image.checksum, 'valid' if image.checksum == checksum else 'invalid!')

def make\_image(args):

image = ESPFirmwareImage()

if len(args.segfile) == 0:

raise FatalError('No segments specified')

if len(args.segfile) != len(args.segaddr):

raise FatalError('Number of specified files does not match number of specified addresses')

for (seg, addr) in zip(args.segfile, args.segaddr):

data = file(seg, 'rb').read()

image.add\_segment(addr, data)

image.entrypoint = args.entrypoint

image.save(args.output)

def elf2image(args):

e = ELFFile(args.input)

if args.version == '1':

image = ESPFirmwareImage()

else:

image = OTAFirmwareImage()

irom\_data = e.load\_section('.irom0.text')

if len(irom\_data) == 0:

raise FatalError(".irom0.text section not found in ELF file - can't create V2 image.")

image.add\_segment(0, irom\_data, 16)

image.entrypoint = e.get\_entry\_point()

for section, start in ((".text", "\_text\_start"), (".data", "\_data\_start"), (".rodata", "\_rodata\_start")):

data = e.load\_section(section)

image.add\_segment(e.get\_symbol\_addr(start), data)

image.flash\_mode = {'qio':0, 'qout':1, 'dio':2, 'dout': 3}[args.flash\_mode]

image.flash\_size\_freq = {'4m':0x00, '2m':0x10, '8m':0x20, '16m':0x30, '32m':0x40, '16m-c1': 0x50, '32m-c1':0x60, '32m-c2':0x70}[args.flash\_size]

image.flash\_size\_freq += {'40m':0, '26m':1, '20m':2, '80m': 0xf}[args.flash\_freq]

irom\_offs = e.get\_symbol\_addr("\_irom0\_text\_start") - 0x40200000

if args.version == '1':

if args.output is None:

args.output = args.input + '-'

image.save(args.output + "0x00000.bin")

data = e.load\_section(".irom0.text")

if irom\_offs < 0:

raise FatalError('Address of symbol \_irom0\_text\_start in ELF is located before flash mapping address. Bad linker script?')

if (irom\_offs & 0xFFF) != 0: # irom0 isn't flash sector aligned

print "WARNING: irom0 section offset is 0x%08x. ELF is probably linked for 'elf2image --version=2'" % irom\_offs

with open(args.output + "0x%05x.bin" % irom\_offs, "wb") as f:

f.write(data)

f.close()

else: # V2 OTA image

if args.output is None:

args.output = "%s-0x%05x.bin" % (os.path.splitext(args.input)[0], irom\_offs & ~(ESPROM.ESP\_FLASH\_SECTOR - 1))

image.save(args.output)

def read\_mac(esp, args):

mac = esp.read\_mac()

print 'MAC: %s' % ':'.join(map(lambda x: '%02x' % x, mac))

def chip\_id(esp, args):

chipid = esp.chip\_id()

print 'Chip ID: 0x%08x' % chipid

def erase\_flash(esp, args):

flasher = CesantaFlasher(esp, args.baud)

print 'Erasing flash (this may take a while)...'

t = time.time()

flasher.flash\_erase\_chip()

t = time.time() - t

print 'Erase took %.1f seconds' % t

def run(esp, args):

esp.run()

def flash\_id(esp, args):

flash\_id = esp.flash\_id()

esp.flash\_finish(False)

print 'Manufacturer: %02x' % (flash\_id & 0xff)

print 'Device: %02x%02x' % ((flash\_id >> 8) & 0xff, (flash\_id >> 16) & 0xff)

def read\_flash(esp, args):

flasher = CesantaFlasher(esp, args.baud)

t = time.time()

data = flasher.flash\_read(args.address, args.size, not args.no\_progress)

t = time.time() - t

print ('\rRead %d bytes at 0x%x in %.1f seconds (%.1f kbit/s)...'

% (len(data), args.address, t, len(data) / t \* 8 / 1000))

file(args.filename, 'wb').write(data)

def \_verify\_flash(flasher, args, flash\_params=None):

differences = False

for address, argfile in args.addr\_filename:

image = argfile.read()

argfile.seek(0) # rewind in case we need it again

if address == 0 and image[0] == '\xe9' and flash\_params is not None:

image = image[0:2] + flash\_params + image[4:]

image\_size = len(image)

print 'Verifying 0x%x (%d) bytes @ 0x%08x in flash against %s...' % (image\_size, image\_size, address, argfile.name)

# Try digest first, only read if there are differences.

digest, \_ = flasher.flash\_digest(address, image\_size)

digest = hexify(digest).upper()

expected\_digest = hashlib.md5(image).hexdigest().upper()

if digest == expected\_digest:

print '-- verify OK (digest matched)'

continue

else:

differences = True

if getattr(args, 'diff', 'no') != 'yes':

print '-- verify FAILED (digest mismatch)'

continue

flash = flasher.flash\_read(address, image\_size)

assert flash != image

diff = [i for i in xrange(image\_size) if flash[i] != image[i]]

print '-- verify FAILED: %d differences, first @ 0x%08x' % (len(diff), address + diff[0])

for d in diff:

print ' %08x %02x %02x' % (address + d, ord(flash[d]), ord(image[d]))

if differences:

raise FatalError("Verify failed.")

def verify\_flash(esp, args, flash\_params=None):

flasher = CesantaFlasher(esp)

\_verify\_flash(flasher, args, flash\_params)

def version(args):

print \_\_version\_\_

#

# End of operations functions

#

def main():

parser = argparse.ArgumentParser(description='esptool.py v%s - ESP8266 ROM Bootloader Utility' % \_\_version\_\_, prog='esptool')

parser.add\_argument(

'--port', '-p',

help='Serial port device',

default=os.environ.get('ESPTOOL\_PORT', '/dev/ttyUSB0'))

parser.add\_argument(

'--baud', '-b',

help='Serial port baud rate used when flashing/reading',

type=arg\_auto\_int,

default=os.environ.get('ESPTOOL\_BAUD', ESPROM.ESP\_ROM\_BAUD))

subparsers = parser.add\_subparsers(

dest='operation',

help='Run esptool {command} -h for additional help')

parser\_load\_ram = subparsers.add\_parser(

'load\_ram',

help='Download an image to RAM and execute')

parser\_load\_ram.add\_argument('filename', help='Firmware image')

parser\_dump\_mem = subparsers.add\_parser(

'dump\_mem',

help='Dump arbitrary memory to disk')

parser\_dump\_mem.add\_argument('address', help='Base address', type=arg\_auto\_int)

parser\_dump\_mem.add\_argument('size', help='Size of region to dump', type=arg\_auto\_int)

parser\_dump\_mem.add\_argument('filename', help='Name of binary dump')

parser\_read\_mem = subparsers.add\_parser(

'read\_mem',

help='Read arbitrary memory location')

parser\_read\_mem.add\_argument('address', help='Address to read', type=arg\_auto\_int)

parser\_write\_mem = subparsers.add\_parser(

'write\_mem',

help='Read-modify-write to arbitrary memory location')

parser\_write\_mem.add\_argument('address', help='Address to write', type=arg\_auto\_int)

parser\_write\_mem.add\_argument('value', help='Value', type=arg\_auto\_int)

parser\_write\_mem.add\_argument('mask', help='Mask of bits to write', type=arg\_auto\_int)

def add\_spi\_flash\_subparsers(parent, auto\_detect=False):

""" Add common parser arguments for SPI flash properties """

parent.add\_argument('--flash\_freq', '-ff', help='SPI Flash frequency',

choices=['40m', '26m', '20m', '80m'],

default=os.environ.get('ESPTOOL\_FF', '40m'))

parent.add\_argument('--flash\_mode', '-fm', help='SPI Flash mode',

choices=['qio', 'qout', 'dio', 'dout'],

default=os.environ.get('ESPTOOL\_FM', 'qio'))

choices = ['4m', '2m', '8m', '16m', '32m', '16m-c1', '32m-c1', '32m-c2']

default = '4m'

if auto\_detect:

default = 'detect'

choices.insert(0, 'detect')

parent.add\_argument('--flash\_size', '-fs', help='SPI Flash size in Mbit', type=str.lower,

choices=choices,

default=os.environ.get('ESPTOOL\_FS', default))

parser\_write\_flash = subparsers.add\_parser(

'write\_flash',

help='Write a binary blob to flash')

parser\_write\_flash.add\_argument('addr\_filename', metavar='<address> <filename>', help='Address followed by binary filename, separated by space',

action=AddrFilenamePairAction)

add\_spi\_flash\_subparsers(parser\_write\_flash, auto\_detect=True)

parser\_write\_flash.add\_argument('--no-progress', '-p', help='Suppress progress output', action="store\_true")

parser\_write\_flash.add\_argument('--verify', help='Verify just-written data (only necessary if very cautious, data is already CRCed', action='store\_true')

subparsers.add\_parser(

'run',

help='Run application code in flash')

parser\_image\_info = subparsers.add\_parser(

'image\_info',

help='Dump headers from an application image')

parser\_image\_info.add\_argument('filename', help='Image file to parse')

parser\_make\_image = subparsers.add\_parser(

'make\_image',

help='Create an application image from binary files')

parser\_make\_image.add\_argument('output', help='Output image file')

parser\_make\_image.add\_argument('--segfile', '-f', action='append', help='Segment input file')

parser\_make\_image.add\_argument('--segaddr', '-a', action='append', help='Segment base address', type=arg\_auto\_int)

parser\_make\_image.add\_argument('--entrypoint', '-e', help='Address of entry point', type=arg\_auto\_int, default=0)

parser\_elf2image = subparsers.add\_parser(

'elf2image',

help='Create an application image from ELF file')

parser\_elf2image.add\_argument('input', help='Input ELF file')

parser\_elf2image.add\_argument('--output', '-o', help='Output filename prefix (for version 1 image), or filename (for version 2 single image)', type=str)

parser\_elf2image.add\_argument('--version', '-e', help='Output image version', choices=['1','2'], default='1')

add\_spi\_flash\_subparsers(parser\_elf2image)

subparsers.add\_parser(

'read\_mac',

help='Read MAC address from OTP ROM')

subparsers.add\_parser(

'chip\_id',

help='Read Chip ID from OTP ROM')

subparsers.add\_parser(

'flash\_id',

help='Read SPI flash manufacturer and device ID')

parser\_read\_flash = subparsers.add\_parser(

'read\_flash',

help='Read SPI flash content')

parser\_read\_flash.add\_argument('address', help='Start address', type=arg\_auto\_int)

parser\_read\_flash.add\_argument('size', help='Size of region to dump', type=arg\_auto\_int)

parser\_read\_flash.add\_argument('filename', help='Name of binary dump')

parser\_read\_flash.add\_argument('--no-progress', '-p', help='Suppress progress output', action="store\_true")

parser\_verify\_flash = subparsers.add\_parser(

'verify\_flash',

help='Verify a binary blob against flash')

parser\_verify\_flash.add\_argument('addr\_filename', help='Address and binary file to verify there, separated by space',

action=AddrFilenamePairAction)

parser\_verify\_flash.add\_argument('--diff', '-d', help='Show differences',

choices=['no', 'yes'], default='no')

subparsers.add\_parser(

'erase\_flash',

help='Perform Chip Erase on SPI flash')

subparsers.add\_parser(

'version', help='Print esptool version')

# internal sanity check - every operation matches a module function of the same name

for operation in subparsers.choices.keys():

assert operation in globals(), "%s should be a module function" % operation

args = parser.parse\_args()

print 'esptool.py v%s' % \_\_version\_\_

# operation function can take 1 arg (args), 2 args (esp, arg)

# or be a member function of the ESPROM class.

operation\_func = globals()[args.operation]

operation\_args,\_,\_,\_ = inspect.getargspec(operation\_func)

if operation\_args[0] == 'esp': # operation function takes an ESPROM connection object

initial\_baud = min(ESPROM.ESP\_ROM\_BAUD, args.baud) # don't sync faster than the default baud rate

esp = ESPROM(args.port, initial\_baud)

esp.connect()

operation\_func(esp, args)

else:

operation\_func(args)

class AddrFilenamePairAction(argparse.Action):

""" Custom parser class for the address/filename pairs passed as arguments """

def \_\_init\_\_(self, option\_strings, dest, nargs='+', \*\*kwargs):

super(AddrFilenamePairAction, self).\_\_init\_\_(option\_strings, dest, nargs, \*\*kwargs)

def \_\_call\_\_(self, parser, namespace, values, option\_string=None):

# validate pair arguments

pairs = []

for i in range(0,len(values),2):

try:

address = int(values[i],0)

except ValueError as e:

raise argparse.ArgumentError(self,'Address "%s" must be a number' % values[i])

try:

argfile = open(values[i + 1], 'rb')

except IOError as e:

raise argparse.ArgumentError(self, e)

except IndexError:

raise argparse.ArgumentError(self,'Must be pairs of an address and the binary filename to write there')

pairs.append((address, argfile))

setattr(namespace, self.dest, pairs)

# This is "wrapped" stub\_flasher.c, to be loaded using run\_stub.

\_CESANTA\_FLASHER\_STUB = """\

{"code\_start": 1074790404, "code": "080000601C000060000000601000006031FCFF71FCFF\

81FCFFC02000680332D218C020004807404074DCC48608005823C0200098081BA5A9239245005803\

1B555903582337350129230B446604DFC6F3FF21EEFFC0200069020DF0000000010078480040004A\

0040B449004012C1F0C921D911E901DD0209312020B4ED033C2C56C2073020B43C3C56420701F5FF\

C000003C4C569206CD0EEADD860300202C4101F1FFC0000056A204C2DCF0C02DC0CC6CCAE2D1EAFF\

0606002030F456D3FD86FBFF00002020F501E8FFC00000EC82D0CCC0C02EC0C73DEB2ADC46030020\

2C4101E1FFC00000DC42C2DCF0C02DC056BCFEC602003C5C8601003C6C4600003C7C08312D0CD811\

C821E80112C1100DF0000C180000140010400C0000607418000064180000801800008C1800008418\

0000881800009018000018980040880F0040A80F0040349800404C4A0040740F0040800F0040980F\

00400099004012C1E091F5FFC961CD0221EFFFE941F9310971D9519011C01A223902E2D1180C0222\

6E1D21E4FF31E9FF2AF11A332D0F42630001EAFFC00000C030B43C2256A31621E1FF1A2228022030\

B43C3256B31501ADFFC00000DD023C4256ED1431D6FF4D010C52D90E192E126E0101DDFFC0000021\

D2FF32A101C020004802303420C0200039022C0201D7FFC00000463300000031CDFF1A333803D023\

C03199FF27B31ADC7F31CBFF1A3328030198FFC0000056C20E2193FF2ADD060E000031C6FF1A3328\

030191FFC0000056820DD2DD10460800000021BEFF1A2228029CE231BCFFC020F51A33290331BBFF\

C02C411A332903C0F0F4222E1D22D204273D9332A3FFC02000280E27B3F721ABFF381E1A2242A400\

01B5FFC00000381E2D0C42A40001B3FFC0000056120801B2FFC00000C02000280EC2DC0422D2FCC0\

2000290E01ADFFC00000222E1D22D204226E1D281E22D204E7B204291E860000126E012198FF32A0\

042A21C54C003198FF222E1D1A33380337B202C6D6FF2C02019FFFC000002191FF318CFF1A223A31\

019CFFC00000218DFF1C031A22C549000C02060300003C528601003C624600003C72918BFF9A1108\

71C861D851E841F83112C1200DF00010000068100000581000007010000074100000781000007C10\

0000801000001C4B0040803C004091FDFF12C1E061F7FFC961E941F9310971D9519011C01A662906\

21F3FFC2D1101A22390231F2FF0C0F1A33590331EAFFF26C1AED045C2247B3028636002D0C016DFF\

C0000021E5FF41EAFF2A611A4469040622000021E4FF1A222802F0D2C0D7BE01DD0E31E0FF4D0D1A\

3328033D0101E2FFC00000561209D03D2010212001DFFFC000004D0D2D0C3D01015DFFC0000041D5\

FFDAFF1A444804D0648041D2FF1A4462640061D1FF106680622600673F1331D0FF10338028030C43\

853A002642164613000041CAFF222C1A1A444804202FC047328006F6FF222C1A273F3861C2FF222C\

1A1A6668066732B921BDFF3D0C1022800148FFC0000021BAFF1C031A2201BFFFC000000C02460300\

5C3206020000005C424600005C5291B7FF9A110871C861D851E841F83112C1200DF0B0100000C010\

0000D010000012C1E091FEFFC961D951E9410971F931CD039011C0ED02DD0431A1FF9C1422A06247\

B302062D0021F4FF1A22490286010021F1FF1A223902219CFF2AF12D0F011FFFC00000461C0022D1\

10011CFFC0000021E9FFFD0C1A222802C7B20621E6FF1A22F8022D0E3D014D0F0195FFC000008C52\

22A063C6180000218BFF3D01102280F04F200111FFC00000AC7D22D1103D014D0F010DFFC0000021\

D6FF32D110102280010EFFC0000021D3FF1C031A220185FFC00000FAEEF0CCC056ACF821CDFF317A\

FF1A223A310105FFC0000021C9FF1C031A22017CFFC000002D0C91C8FF9A110871C861D851E841F8\

3112C1200DF0000200600000001040020060FFFFFF0012C1E00C02290131FAFF21FAFF026107C961\

C02000226300C02000C80320CC10564CFF21F5FFC02000380221F4FF20231029010C432D010163FF\

C0000008712D0CC86112C1200DF00080FE3F8449004012C1D0C9A109B17CFC22C1110C13C51C0026\

1202463000220111C24110B68202462B0031F5FF3022A02802A002002D011C03851A0066820A2801\

32210105A6FF0607003C12C60500000010212032A01085180066A20F2221003811482105B3FF2241\

10861A004C1206FDFF2D011C03C5160066B20E280138114821583185CFFF06F7FF005C1286F5FF00\

10212032A01085140066A20D2221003811482105E1FF06EFFF0022A06146EDFF45F0FFC6EBFF0000\

01D2FFC0000006E9FF000C022241100C1322C110C50F00220111060600000022C1100C13C50E0022\

011132C2FA303074B6230206C8FF08B1C8A112C1300DF0000000000010404F484149007519031027\

000000110040A8100040BC0F0040583F0040CC2E00401CE20040D83900408000004021F4FF12C1E0\

C961C80221F2FF097129010C02D951C91101F4FFC0000001F3FFC00000AC2C22A3E801F2FFC00000\

21EAFFC031412A233D0C01EFFFC000003D0222A00001EDFFC00000C1E4FF2D0C01E8FFC000002D01\

32A004450400C5E7FFDD022D0C01E3FFC00000666D1F4B2131DCFF4600004B22C0200048023794F5\

31D9FFC0200039023DF08601000001DCFFC000000871C861D85112C1200DF000000012C1F0026103\

01EAFEC00000083112C1100DF000643B004012C1D0E98109B1C9A1D991F97129013911E2A0C001FA\

FFC00000CD02E792F40C0DE2A0C0F2A0DB860D00000001F4FFC00000204220E71240F7921C226102\

01EFFFC0000052A0DC482157120952A0DD571205460500004D0C3801DA234242001BDD3811379DC5\

C6000000000C0DC2A0C001E3FFC00000C792F608B12D0DC8A1D891E881F87112C1300DF00000", "\

entry": 1074792180, "num\_params": 1, "params\_start": 1074790400, "data": "FE0510\

401A0610403B0610405A0610407A061040820610408C0610408C061040", "data\_start": 10736\

43520}

"""

if \_\_name\_\_ == '\_\_main\_\_':

try:

main()

except FatalError as e:

print '\nA fatal error occurred: %s' % e

sys.exit(2)

(virtu) [aluno@fleln static]$ cp ./Projeto\_ESP8266/virtualPython/virtu/python3.6/site-packages/esptool.py ./Projeto\_ESP8266/esp\_open\_sdk/xtensa-lx106-elf/bin/

cp: impossível obter estado de “./Projeto\_ESP8266/virtualPython/virtu/python3.6/site-packages/esptool.py”: Arquivo ou diretório não encontrado

(virtu) [aluno@fleln static]$ cp ./Projeto\_ESP8266/virtualPython/virtu/lib/python3.6/site-packages/esptool.py ./Projeto\_ESP8266/esp\_open\_sdk/xtensa-lx106-elf/bin/

cp: não foi possível criar arquivo comum “./Projeto\_ESP8266/esp\_open\_sdk/xtensa-lx106-elf/bin/”: Arquivo ou diretório não encontrado

(virtu) [aluno@fleln static]$ cp ./Projeto\_ESP8266/virtualPython/virtu/lib/python3.6/site-packages/esptool.py ./Projeto\_ESP8266/esp-open-rtos/

bootloader/ core/ extras/ .gitignore ld/ LICENSE open\_esplibs/ tests/

code\_of\_conduct.md .dir-locals.el FreeRTOS/ .gitmodules lib/ lvgl/ parameters.mk .travis.yml

common.mk examples/ .git/ include/ libc/ lwip/ README.md utils/

(virtu) [aluno@fleln static]$ cp ./Projeto\_ESP8266/virtualPython/virtu/lib/python3.6/site-packages/esptool.py ./Projeto\_ESP8266/esp-open-rtos/

bootloader/ core/ extras/ .gitignore ld/ LICENSE open\_esplibs/ tests/

code\_of\_conduct.md .dir-locals.el FreeRTOS/ .gitmodules lib/ lvgl/ parameters.mk .travis.yml

common.mk examples/ .git/ include/ libc/ lwip/ README.md utils/

(virtu) [aluno@fleln static]$ cp ./Projeto\_ESP8266/virtualPython/virtu/lib/python3.6/site-packages/esptool.py ./Projeto\_ESP8266/esp-open-sdk/xtensa-lx106-elf/bin

(virtu) [aluno@fleln static]$ make -j4

make: \*\*\* Nenhum alvo indicado e nenhum arquivo make encontrado. Pare.

(virtu) [aluno@fleln static]$ make

make: \*\*\* Nenhum alvo indicado e nenhum arquivo make encontrado. Pare.

(virtu) [aluno@fleln static]$ source toolchain.sh

/home/static/Projeto\_ESP8266/virtualPython/virtu/bin:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/opt/imgtec/Toolchains/mips-mti-elf/2017.10-05/bin:/opt/imgtec/Toolchains/mips-img-elf/2017.10-05/bin:/opt/intelFPGA/18.1/quartus/bin:/opt/intelFPGA/18.1/modelsim\_ae/bin:/opt/intelFPGA/18.1/modelsim\_ae/linuxaloem:/opt/intelFPGA/18.1/nios2eds/bin:/home/aluno/.local/bin:/home/aluno/bin:/opt/imgtec/Toolchains/mips-mti-elf/2017.10-05/bin:/opt/imgtec/Toolchains/mips-img-elf/2017.10-05/bin:/opt/intelFPGA/18.1/quartus/bin:/opt/intelFPGA/18.1/modelsim\_ae/bin:/opt/intelFPGA/18.1/modelsim\_ae/linuxaloem:/opt/intelFPGA/18.1/nios2eds/bin:/home/static/Projeto\_ESP8266/esp-open-sdk/xtensa-lx106-elf/bin

(virtu) [aluno@fleln static]$ cd

lost+found/ mcuxpresso/ Projeto\_ESP8266/ .Trash-1001/

(virtu) [aluno@fleln static]$ cd Projeto\_ESP8266/esp-open-

esp-open-rtos/ esp-open-sdk/

(virtu) [aluno@fleln static]$ cd Projeto\_ESP8266/esp-open-rtos/examples/

(virtu) [aluno@fleln examples]$

(virtu) [aluno@fleln examples]$ make flash -4 -C blink

make: invalid option -- '4'

Uso: make [opções] [alvo] ...

Opções:

-b, -m Ignorado para compatibilidade.

-B, --always-make Processa todos os alvos incondicionalmente.

-C DIRETÓRIO, --directory= DIRETÓRIO

muda para o DIRETÓRIO antes de fazer algo.

-d Imprime muita informação de depuração.

--debug[=OPÇÕES] Imprime vários tipos de informações de depuração.

-e,--envrionment-overrides

As variáveis de ambiente sobrescrevem os arquivos make.

--eval=STRING Evaluate STRING as a makefile statement.

-f ARQUIVO, --file=ARQUIVO --makefile=ARQUIVO

Lê o ARQUIVO com um arquivo make.

-h, --help Imprime esta mensagem e sai.

-i, --ignore-errors Ignore errors from recipes.

.. -l DIRETÓRIO, --include-dir= DIRETÓRIO

Procura no DIRETÓRIO por arquivos make.

-j [N], --jobs[=N] Permite N tarefas de uma vez ou várias sem o argumento.

-k, --keep-going Continua mesmo que alguns alvos não possam ser processados.

-l [N], --load-average[=N], --max-load[=N]

Não inicia múltiplas tarefas a menos que a carga seja menor que N.

-L, --check-symlink-times Use the latest mtime between symlinks and target.

-n, --just-print, --dry-run, --recon

Don't actually run any recipe; just print them.

-o ARQUIVO, --old-file= ARQUIVO, --assume-old ARQUIVO

Considera o ARQUIVO muito velho e não o refaz.

-p, --print-data-base Imprime o banco de dados interno do make.

-q, --question Run no recipe; exit status says if up to date.

-r, --no-builtin-rules Desabilita as regras implícitas embutidas.

-R, --no-builtin-variables Desabilita as configurações das variávies embutidas.

-s, --silent, --quiet Don't echo recipes.

-S, --no-keep-going, --stop

Desativa a opção -k.

-t, --touch Executa um `touch' nos alvos ao invés de reprocessá-los.

-v, --version Imprime o número de versão do make e sai.

-w, --print-directory Imprime o diretório atual.

--no-print-directory Desativa a opção -w, mesmo que tenha sido ativada implicitamente.

-W ARQUIVO, --what-if= ARQUIVO, --new-file=ARQUIVO, --assume-new=ARQUIVO

Considera o ARQUIVO muito novo.

--warn-undefined-variables Avisa quando um variável não definida for referenciada.

--warn-undefined-functions Warn when an undefined user function is called.

Este programa foi compilado para x86\_64-redhat-linux-gnu

Informe os problemas para <bug-make@gnu.org>.

(virtu) [aluno@fleln examples]$ make flash -j4 -C blink

make: Entrando no diretório `/home/static/Projeto\_ESP8266/esp-open-rtos/examples/blink'

esptool.py -p /dev/ttyUSB0 --baud 115200 write\_flash -fs 16m -fm qio -ff 40m \

0x0 ../../bootloader/firmware\_prebuilt/rboot.bin 0x1000 ../../bootloader/firmware\_prebuilt/blank\_config.bin 0x2000 ./firmware/blink.bin

WARNING: Flash size arguments in megabits like '16m' are deprecated.

Please use the equivalent size '2MB'.

Megabit arguments may be removed in a future release.

esptool.py v2.8

Serial port /dev/ttyUSB0

Traceback (most recent call last):

File "/home/static/Projeto\_ESP8266/virtualPython/virtu/lib64/python3.6/site-packages/serial/serialposix.py", line 265, in open

self.fd = os.open(self.portstr, os.O\_RDWR | os.O\_NOCTTY | os.O\_NONBLOCK)

FileNotFoundError: [Errno 2] No such file or directory: '/dev/ttyUSB0'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):

File "/home/static/Projeto\_ESP8266/virtualPython/virtu/bin/esptool.py", line 3201, in <module>

\_main()

File "/home/static/Projeto\_ESP8266/virtualPython/virtu/bin/esptool.py", line 3194, in \_main

main()

File "/home/static/Projeto\_ESP8266/virtualPython/virtu/bin/esptool.py", line 2883, in main

esp = ESPLoader.detect\_chip(each\_port, initial\_baud, args.before, args.trace)

File "/home/static/Projeto\_ESP8266/virtualPython/virtu/bin/esptool.py", line 273, in detect\_chip

detect\_port = ESPLoader(port, baud, trace\_enabled=trace\_enabled)

File "/home/static/Projeto\_ESP8266/virtualPython/virtu/bin/esptool.py", line 237, in \_\_init\_\_

self.\_port = serial.serial\_for\_url(port)

File "/home/static/Projeto\_ESP8266/virtualPython/virtu/lib64/python3.6/site-packages/serial/\_\_init\_\_.py", line 88, in serial\_for\_url

instance.open()

File "/home/static/Projeto\_ESP8266/virtualPython/virtu/lib64/python3.6/site-packages/serial/serialposix.py", line 268, in open

raise SerialException(msg.errno, "could not open port {}: {}".format(self.\_port, msg))

serial.serialutil.SerialException: [Errno 2] could not open port /dev/ttyUSB0: [Errno 2] No such file or directory: '/dev/ttyUSB0'

make: \*\* [flash] Erro 1

make: Saindo do diretório `/home/static/Projeto\_ESP8266/esp-open-rtos/examples/blink'

(virtu) [aluno@fleln examples]$ make -j4 -C blink

make: Entrando no diretório `/home/static/Projeto\_ESP8266/esp-open-rtos/examples/blink'

make: Nada a ser feito para `all'.

make: Saindo do diretório `/home/static/Projeto\_ESP8266/esp-open-rtos/examples/blink'

(virtu) [aluno@fleln examples]$ make flash -j4 -C blink

make: Entrando no diretório `/home/static/Projeto\_ESP8266/esp-open-rtos/examples/blink'

esptool.py -p /dev/ttyUSB0 --baud 115200 write\_flash -fs 16m -fm qio -ff 40m \

0x0 ../../bootloader/firmware\_prebuilt/rboot.bin 0x1000 ../../bootloader/firmware\_prebuilt/blank\_config.bin 0x2000 ./firmware/blink.bin

WARNING: Flash size arguments in megabits like '16m' are deprecated.

Please use the equivalent size '2MB'.

Megabit arguments may be removed in a future release.

esptool.py v2.8

Serial port /dev/ttyUSB0

Connecting....

Detecting chip type... ESP8266

Chip is ESP8266EX

Features: WiFi

Crystal is 26MHz

MAC: 18:fe:34:d6:7c:ad

Uploading stub...

Running stub...

Stub running...

Configuring flash size...

Compressed 3104 bytes to 2169...

Wrote 3104 bytes (2169 compressed) at 0x00000000 in 0.2 seconds (effective 126.0 kbit/s)...

Hash of data verified.

Compressed 2048 bytes to 23...

Wrote 2048 bytes (23 compressed) at 0x00001000 in 0.0 seconds (effective 1830.6 kbit/s)...

Hash of data verified.

Compressed 254368 bytes to 186445...

Wrote 254368 bytes (186445 compressed) at 0x00002000 in 16.4 seconds (effective 124.0 kbit/s)...

Hash of data verified.

Leaving...

Hard resetting via RTS pin...

make: Saindo do diretório `/home/static/Projeto\_ESP8266/esp-open-rtos/examples/blink'

(virtu) [aluno@fleln examples]$ gedit .blink .c

(virtu) [aluno@fleln examples]$

(virtu) [aluno@fleln examples]$ gedit .blink.c

(virtu) [aluno@fleln examples]$ gedit blink.c

(virtu) [aluno@fleln examples]$ gedit blçi

(virtu) [aluno@fleln examples]$ gedit blink/blink.c

^C^C

(virtu) [aluno@fleln examples]$ ^C

(virtu) [aluno@fleln examples]$ pip --version

pip 18.1 from /home/static/Projeto\_ESP8266/virtualPython/virtu/lib64/python3.6/site-packages/pip (python 3.6)

(virtu) [aluno@fleln examples]$ make blink

make: Nada a ser feito para `blink'.

(virtu) [aluno@fleln examples]$ make clean blink

make: \*\*\* Sem regra para processar o alvo `clean'. Pare.

(virtu) [aluno@fleln examples]$ cd blink

(virtu) [aluno@fleln blink]$ make clean

(virtu) [aluno@fleln blink]$ cd ..

(virtu) [aluno@fleln examples]$ make blink

make: Nada a ser feito para `blink'.

(virtu) [aluno@fleln examples]$ make -C blink

make: Entrando no diretório `/home/static/Projeto\_ESP8266/esp-open-rtos/examples/blink'

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libmain/timers.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libmain/ets\_timer.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libmain/spi\_flash.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libmain/os\_cpu\_a.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libmain/uart.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libmain/misc.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libmain/user\_interface.c

AS /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libmain/xtensa\_context.S

AR build/open\_esplibs\_libmain.a

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libnet80211/ieee80211\_hostap.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libnet80211/wl\_cnx.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libnet80211/ieee80211\_ets.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libnet80211/ieee80211\_input.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libnet80211/ieee80211\_sta.c

AR build/open\_esplibs\_libnet80211.a

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libphy/phy\_chip\_v6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libphy/phy.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libphy/phy\_sleep.c

AR build/open\_esplibs\_libphy.a

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libpp/pm.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libpp/if\_hwctrl.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libpp/pp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libpp/esf\_buf.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libpp/lmac.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libpp/wdev.c

AR build/open\_esplibs\_libpp.a

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libwpa/os\_xtensa.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libwpa/wpa\_main.c

AR build/open\_esplibs\_libwpa.a

CC /home/static/Projeto\_ESP8266/esp-open-rtos/examples/blink/blink.c

AR build/program.a

CC /home/static/Projeto\_ESP8266/esp-open-rtos/FreeRTOS/Source/stream\_buffer.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/FreeRTOS/Source/queue.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/FreeRTOS/Source/croutine.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/FreeRTOS/Source/list.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/FreeRTOS/Source/event\_groups.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/FreeRTOS/Source/tasks.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/FreeRTOS/Source/timers.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/FreeRTOS/Source/portable/esp8266/port.c

AR build/freertos.a

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/sys\_arch.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/esp\_interface.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/netbuf.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/api\_lib.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/netifapi.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/tcpip.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/netdb.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/if\_api.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/err.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/api\_msg.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/sockets.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/mem.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/init.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/memp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/sys.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/tcp\_in.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/dns.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/timeouts.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ip.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/def.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/raw.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/tcp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/pbuf.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/netif.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/stats.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/altcp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/inet\_chksum.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/altcp\_tcp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/udp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/tcp\_out.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv4/ip4\_frag.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv4/dhcp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv4/etharp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv4/ip4\_addr.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv4/igmp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv4/ip4.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv4/autoip.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv4/icmp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/ip6\_frag.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/dhcp6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/inet6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/ip6\_addr.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/ip6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/nd6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/mld6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/ethip6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/icmp6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/netif/slipif.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/netif/ethernet.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/netif/bridgeif\_fdb.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/netif/lowpan6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/netif/bridgeif.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/netif/ethernetif.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/lwiperf/lwiperf.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/sntp/sntp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/tftp/tftp\_server.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/netbiosns/netbiosns.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_threadsync.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_scalar.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmpv3.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_mib2.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_asn1.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_pbuf\_stream.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_mib2\_tcp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_mib2\_system.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_mib2\_ip.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmpv3\_mbedtls.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_netconn.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_mib2\_snmp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_traps.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_mib2\_interfaces.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_core.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_msg.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_mib2\_icmp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_raw.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_table.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_snmpv2\_framework.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_snmpv2\_usm.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_mib2\_udp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/smtp/smtp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/mdns/mdns.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/http/http\_client.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/http/fsdata.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/http/httpd.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/http/fs.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/mqtt/mqtt.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/altcp\_tls/altcp\_tls\_mbedtls\_mem.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/altcp\_tls/altcp\_tls\_mbedtls.c

AR build/lwip.a

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/esp\_interrupts.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/app\_main.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/spiflash.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/esp\_spi.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/sdk\_compat.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/sysparam.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/esp\_gpio.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/phy\_info.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/esp\_phy.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/newlib\_syscalls.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/esp\_iomux.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/esp\_timer.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/esp\_hwrand.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/debug\_dumps.c

AS /home/static/Projeto\_ESP8266/esp-open-rtos/core/exception\_vectors.S

AS /home/static/Projeto\_ESP8266/esp-open-rtos/core/spiflash-cache-enable.S

C++ /home/static/Projeto\_ESP8266/esp-open-rtos/core/cplusplus\_operators.cpp

AR build/core.a

AR build/open\_esplibs.a

Removing unwanted objects from ../../lib/libgcc.a

Removing unwanted objects from ../../libc/xtensa-lx106-elf/lib/libc.a

SDK processing stage 1: Removing unwanted objects from ../../lib/libmain.a

SDK processing stage 2: Renaming symbols in SDK library build/sdklib/libmain\_stage1.a -> build/sdklib/libmain.a

SDK processing stage 1: Removing unwanted objects from ../../lib/libnet80211.a

SDK processing stage 2: Renaming symbols in SDK library build/sdklib/libnet80211\_stage1.a -> build/sdklib/libnet80211.a

SDK processing stage 1: Removing unwanted objects from ../../lib/libphy.a

SDK processing stage 2: Renaming symbols in SDK library build/sdklib/libphy\_stage1.a -> build/sdklib/libphy.a

SDK processing stage 1: Removing unwanted objects from ../../lib/libpp.a

SDK processing stage 2: Renaming symbols in SDK library build/sdklib/libpp\_stage1.a -> build/sdklib/libpp.a

SDK processing stage 1: Removing unwanted objects from ../../lib/libwpa.a

SDK processing stage 2: Renaming symbols in SDK library build/sdklib/libwpa\_stage1.a -> build/sdklib/libwpa.a

LD build/blink.out

FW firmware/blink.bin

WARNING: Flash size arguments in megabits like '16m' are deprecated.

Please use the equivalent size '2MB'.

Megabit arguments may be removed in a future release.

esptool.py v2.8

Creating image for ESP8266...

make: Saindo do diretório `/home/static/Projeto\_ESP8266/esp-open-rtos/examples/blink'

(virtu) [aluno@fleln examples]$ ls -ltr blink/firmware/

total 252

-rw-rw-r--. 1 aluno aluno 254132 Mar 13 10:53 blink.bin

(virtu) [aluno@fleln examples]$ cd blink

(virtu) [aluno@fleln blink]$ make clean

(virtu) [aluno@fleln blink]$ cd ..

(virtu) [aluno@fleln examples]$ pwd

/home/static/Projeto\_ESP8266/esp-open-rtos/examples

(virtu) [aluno@fleln examples]$ make -C blink

make: Entrando no diretório `/home/static/Projeto\_ESP8266/esp-open-rtos/examples/blink'

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libmain/timers.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libmain/ets\_timer.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libmain/spi\_flash.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libmain/os\_cpu\_a.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libmain/uart.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libmain/misc.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libmain/user\_interface.c

AS /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libmain/xtensa\_context.S

AR build/open\_esplibs\_libmain.a

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libnet80211/ieee80211\_hostap.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libnet80211/wl\_cnx.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libnet80211/ieee80211\_ets.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libnet80211/ieee80211\_input.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libnet80211/ieee80211\_sta.c

AR build/open\_esplibs\_libnet80211.a

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libphy/phy\_chip\_v6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libphy/phy.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libphy/phy\_sleep.c

AR build/open\_esplibs\_libphy.a

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libpp/pm.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libpp/if\_hwctrl.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libpp/pp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libpp/esf\_buf.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libpp/lmac.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libpp/wdev.c

AR build/open\_esplibs\_libpp.a

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libwpa/os\_xtensa.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/open\_esplibs/libwpa/wpa\_main.c

AR build/open\_esplibs\_libwpa.a

CC /home/static/Projeto\_ESP8266/esp-open-rtos/examples/blink/blink.c

AR build/program.a

CC /home/static/Projeto\_ESP8266/esp-open-rtos/FreeRTOS/Source/stream\_buffer.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/FreeRTOS/Source/queue.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/FreeRTOS/Source/croutine.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/FreeRTOS/Source/list.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/FreeRTOS/Source/event\_groups.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/FreeRTOS/Source/tasks.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/FreeRTOS/Source/timers.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/FreeRTOS/Source/portable/esp8266/port.c

AR build/freertos.a

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/sys\_arch.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/esp\_interface.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/netbuf.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/api\_lib.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/netifapi.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/tcpip.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/netdb.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/if\_api.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/err.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/api\_msg.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/api/sockets.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/mem.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/init.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/memp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/sys.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/tcp\_in.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/dns.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/timeouts.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ip.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/def.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/raw.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/tcp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/pbuf.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/netif.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/stats.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/altcp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/inet\_chksum.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/altcp\_tcp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/udp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/tcp\_out.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv4/ip4\_frag.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv4/dhcp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv4/etharp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv4/ip4\_addr.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv4/igmp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv4/ip4.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv4/autoip.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv4/icmp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/ip6\_frag.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/dhcp6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/inet6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/ip6\_addr.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/ip6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/nd6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/mld6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/ethip6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/core/ipv6/icmp6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/netif/slipif.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/netif/ethernet.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/netif/bridgeif\_fdb.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/netif/lowpan6.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/netif/bridgeif.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/netif/ethernetif.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/lwiperf/lwiperf.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/sntp/sntp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/tftp/tftp\_server.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/netbiosns/netbiosns.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_threadsync.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_scalar.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmpv3.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_mib2.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_asn1.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_pbuf\_stream.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_mib2\_tcp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_mib2\_system.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_mib2\_ip.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmpv3\_mbedtls.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_netconn.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_mib2\_snmp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_traps.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_mib2\_interfaces.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_core.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_msg.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_mib2\_icmp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_raw.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_table.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_snmpv2\_framework.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_snmpv2\_usm.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/snmp/snmp\_mib2\_udp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/smtp/smtp.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/mdns/mdns.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/http/http\_client.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/http/fsdata.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/http/httpd.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/http/fs.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/mqtt/mqtt.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/altcp\_tls/altcp\_tls\_mbedtls\_mem.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/lwip/lwip/src/apps/altcp\_tls/altcp\_tls\_mbedtls.c

AR build/lwip.a

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/esp\_interrupts.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/app\_main.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/spiflash.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/esp\_spi.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/sdk\_compat.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/sysparam.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/esp\_gpio.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/phy\_info.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/esp\_phy.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/newlib\_syscalls.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/esp\_iomux.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/esp\_timer.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/esp\_hwrand.c

CC /home/static/Projeto\_ESP8266/esp-open-rtos/core/debug\_dumps.c

AS /home/static/Projeto\_ESP8266/esp-open-rtos/core/exception\_vectors.S

AS /home/static/Projeto\_ESP8266/esp-open-rtos/core/spiflash-cache-enable.S

C++ /home/static/Projeto\_ESP8266/esp-open-rtos/core/cplusplus\_operators.cpp

AR build/core.a

AR build/open\_esplibs.a

Removing unwanted objects from ../../lib/libgcc.a

Removing unwanted objects from ../../libc/xtensa-lx106-elf/lib/libc.a

SDK processing stage 1: Removing unwanted objects from ../../lib/libmain.a

SDK processing stage 2: Renaming symbols in SDK library build/sdklib/libmain\_stage1.a -> build/sdklib/libmain.a

SDK processing stage 1: Removing unwanted objects from ../../lib/libnet80211.a

SDK processing stage 2: Renaming symbols in SDK library build/sdklib/libnet80211\_stage1.a -> build/sdklib/libnet80211.a

SDK processing stage 1: Removing unwanted objects from ../../lib/libphy.a

SDK processing stage 2: Renaming symbols in SDK library build/sdklib/libphy\_stage1.a -> build/sdklib/libphy.a

SDK processing stage 1: Removing unwanted objects from ../../lib/libpp.a

SDK processing stage 2: Renaming symbols in SDK library build/sdklib/libpp\_stage1.a -> build/sdklib/libpp.a

SDK processing stage 1: Removing unwanted objects from ../../lib/libwpa.a

SDK processing stage 2: Renaming symbols in SDK library build/sdklib/libwpa\_stage1.a -> build/sdklib/libwpa.a

LD build/blink.out

FW firmware/blink.bin

WARNING: Flash size arguments in megabits like '16m' are deprecated.

Please use the equivalent size '2MB'.

Megabit arguments may be removed in a future release.

esptool.py v2.8

Creating image for ESP8266...

make: Saindo do diretório `/home/static/Projeto\_ESP8266/esp-open-rtos/examples/blink'

(virtu) [aluno@fleln examples]$ ^C

(virtu) [aluno@fleln examples]$ ^C

(virtu) [aluno@fleln examples]$ ^C

(virtu) [aluno@fleln examples]$