MaixPy Turtorial

CTI One Corporation

Version: x0.1 (Alpha)
Date: Jan 22, 2020
Project Lead: Harry Li, Ph.D.

Team members:
Minh Duc Ong

RISC-V Boards for MaixPy
Hardware for GPP Interface

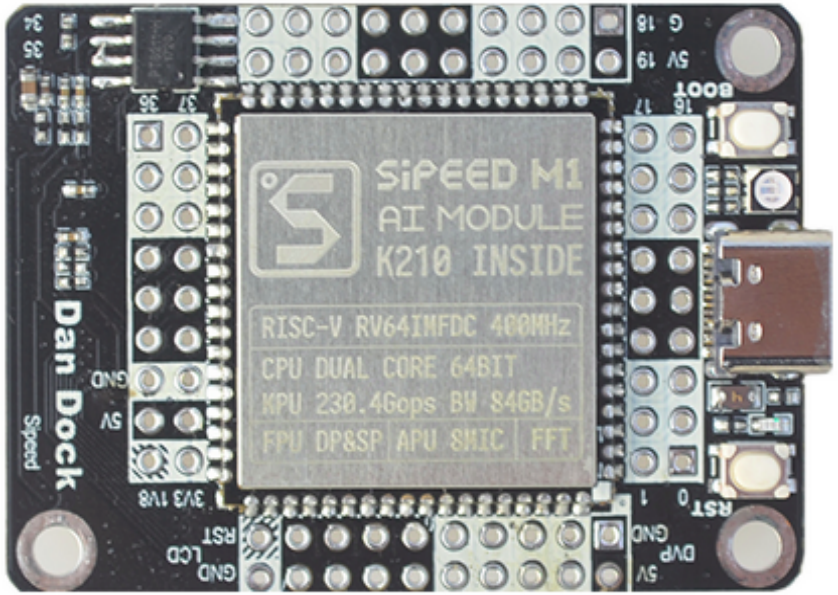
Reference (documents) https://www.seeedstudio.com/Sipeed-M1-dock-suit-M1-dock-2-4-inch-LCD-OV2640-K210-Dev-Board-1st-RV64-AI-board-for-Edge-Computing.html

RISC-V Boards

References: https://maixpy.sipeed.com/en/(https:/github.com/sipeed/(https:/github.com/sipeed/(https:/github.com/sipeed/MaixPy

Sipeed M1 dock suit (M1 dock + 2.4 inch LCD + OV2640) K210 Dev. Board 1st RV64 AI board

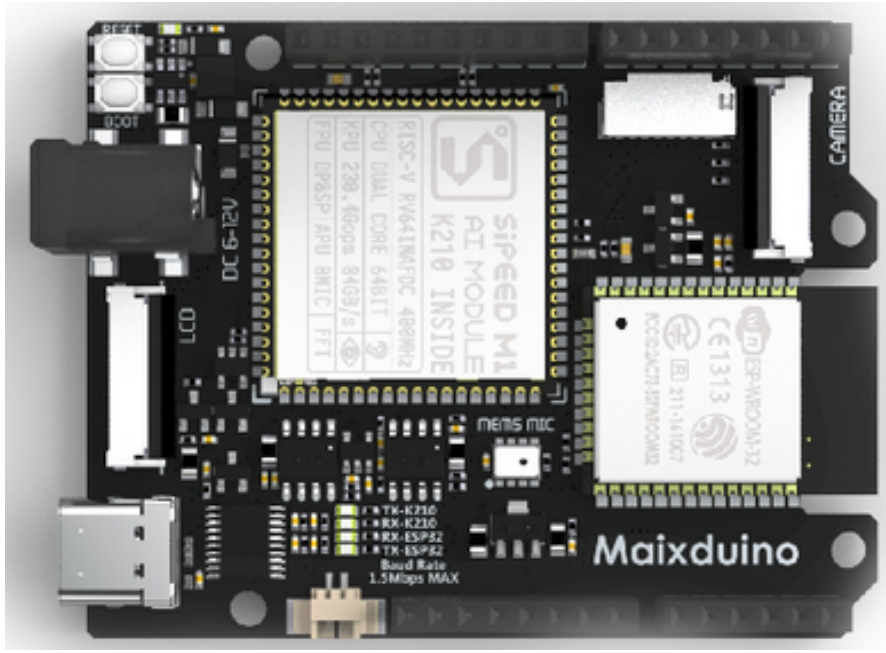
SKU 110991189



To buy https://www.seeedstudio.com/Sipeed-M1-dock-suit-M1-dock-2-4-inch-LCD-OV2640-K210-Dev-Board-1st-RV64-AI-board-for-Edge-Computing.html

Dan dock with Sipeed M1(Dan) module

Sipeed Maixduino



https://www.amazon.com/Seeed-Studio-Sipeed-Maixduino-RISC-V/dp/B07SW9ZWQQ/ref=asc_df_B07SW9ZWQQ/?tag=hyprod-20&linkCode=df0&hvadid=366315306136&hvpos=1o3&hvnetw=g&hvrand=6703665758521683456&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9031967&hvtargid=aud-829758849484:pla-790796503425&psc=1&tag=&ref=&adgrpid=75347436639&hvpone=&hvptwo=&hvadid=366315306136&hvpos=1o3&hvnetw=g&hvrand=6703665758521683456&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9031967&hvtargid=aud-829758849484:pla-790796503425

RISC-V Boards Accessory

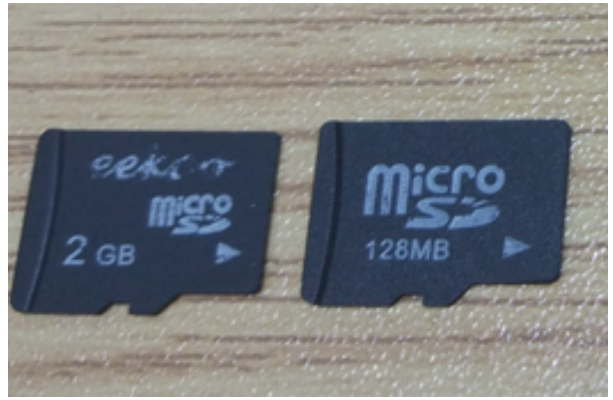
https://maixpy.sipeed.com/en/hardware/lcd.md

USB type C cable

Micro SD card: For quicker operation and additional storage, you can insert a Micro SD card or a TF card into the card slot available on most Maix devices. When purchasing a memory card, try to choose a new fast Micro SD card, such as a SD 2 generation protocol, Class10 memory card.

CAM: ov2640 cameras bundled with Maix device are typically offered with two different lens options; a larger focusable fisheye lens, or a smaller fixed-focus lens

LCD Display: The LCD (24-pin interface) of the st7789 driver chip is used with a resolution of 320x240.



M1 Dock Suit (Dan) Module Connectors

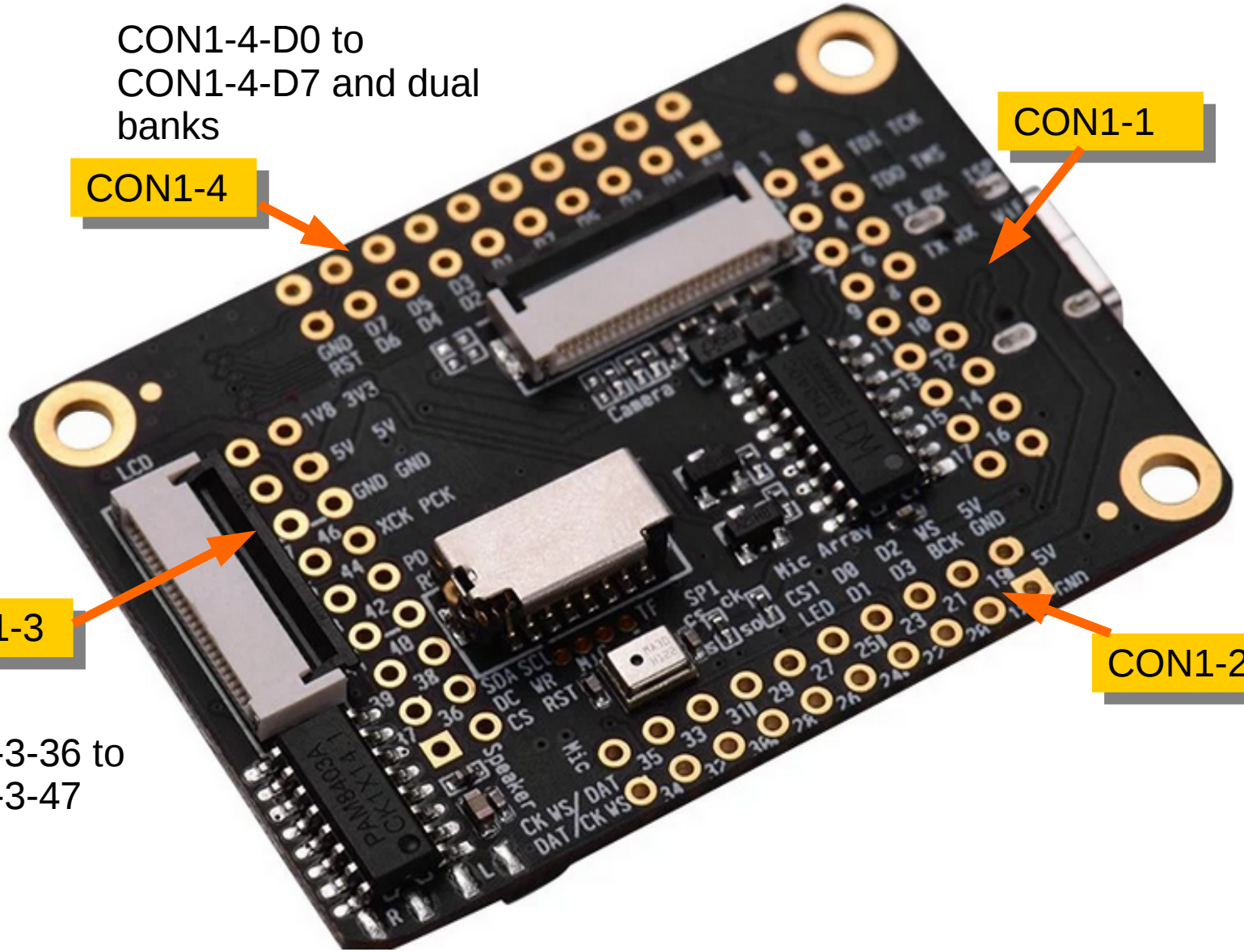
This module is also called MAIX

CON1-4-D0 to CON1-4-D7 and dual banks

CON1-4

CON1-1

CON1-1-0 to CON1-1-17



CON1-3-36 to CON1-3-47

CON1-2

GND, CON1-19 to CON1-35

K210/M1 CPU for M1 Dock Suit

https://www.seeedstudio.com/Sipeed-M1-dock-suit-M1-dock-2-4-inch-LCD-OV2640-K210-Dev-Board-1st-RV64-AI-board-for-Edge-Computing.html

1st competitive RISC-V chip, also 1st competitive AI chip, newly release in Sep. 2018

28nm process, dual-core RISC-V 64bit IMAFDC, on-chip 8MB high-speed SRAM (not for XMR :D), 400MHz frequency (up to 800MHz)

KPU (Neural Network Processor), 64 KPU which is 576bit width, support convolution kernels, any form of activation function. It offers 0.25TOPS@0.3W, 400MHz, when overclock to 800MHz, it offers 0.5TOPS. Object recognition 60fps@VGA

APU (Audio Processor), support 8 mics, up to 192KHz sample rate, hardcore FFT unit, easy for Mic Array (MAIX offer it too)

Flexible FPIOA (Field Programmable IO Array), can map 255 functions to all 48 GPIOs on the chip

DVP camera and MCU LCD interface

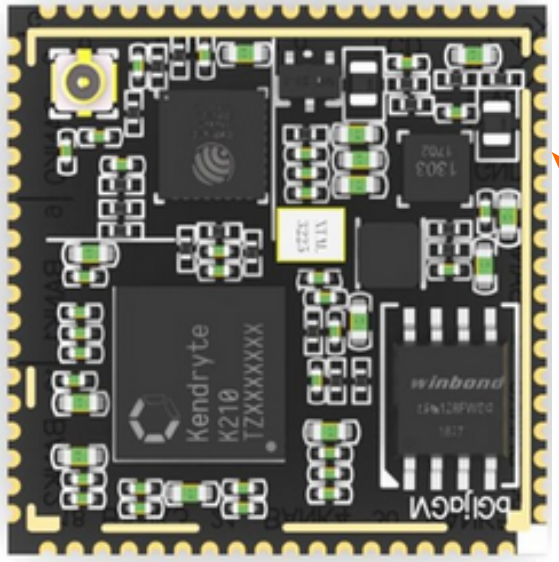
Peripherals: AES Accelerator, SHA256 Accelerator, FFT Accelerator (not APU's one), OTP, UART, WDT, IIC, SPI, I2S, TIMER, RTC, PWM, etc.

K210/M1 CPU and 50 mil Connector

https://www.seeedstudio.com/Sipeed-M1-dock-suit-M1-dock-2-4-inch-LCD-OV2640-K210-Dev-Board-1st-RV64-AI-board-for-Edge-Computing.html

K210 CPU → M1 CPU

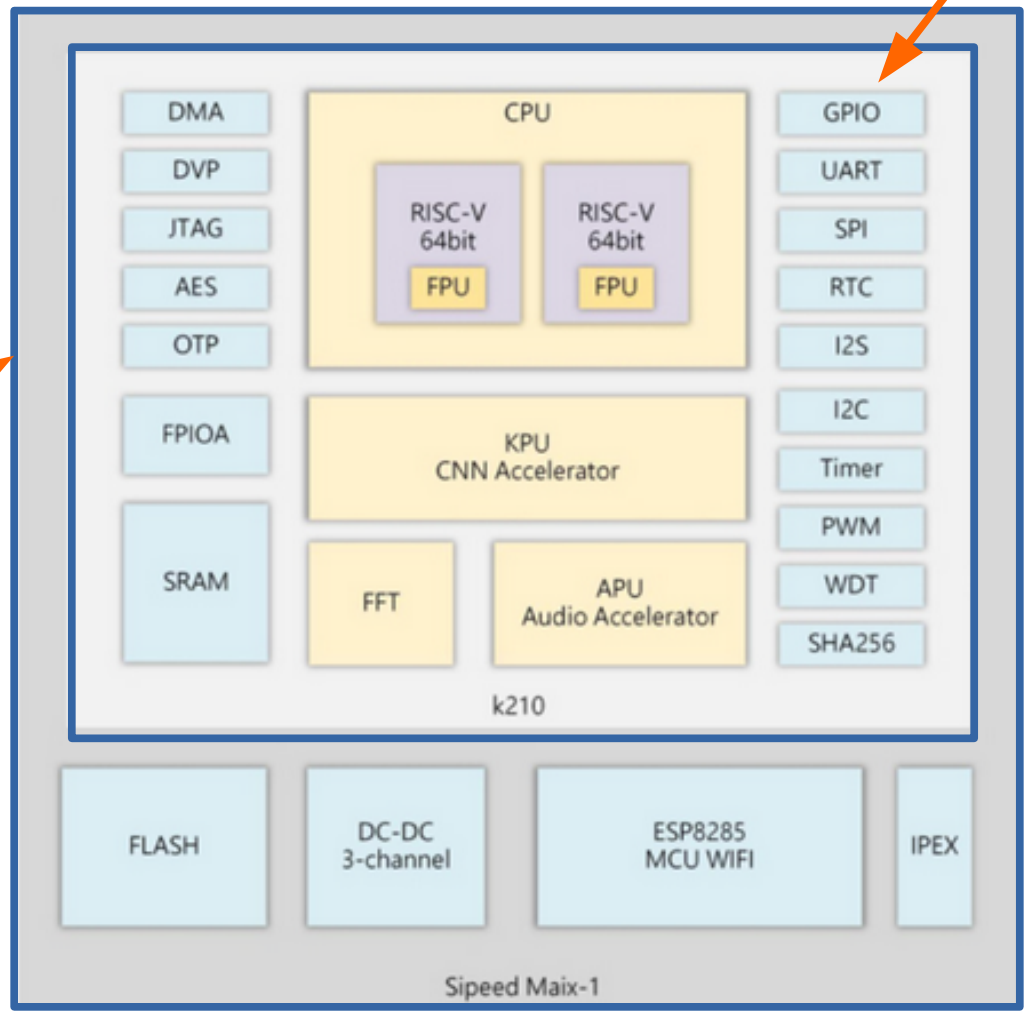
1. Sipeed MAIX-I module, e.g., M1, added (1) 3-channel DC-DC power; (2) 8MB/16MB/128MB Flash; (3) if M1w module, then add wifi chip esp8285; (4) Square Inch Module.



M1 CPU packaged into 1 square inch module

M1 CPU

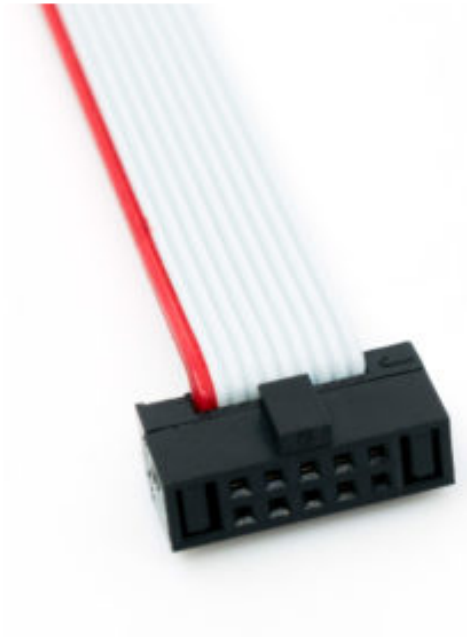
K210 CPU

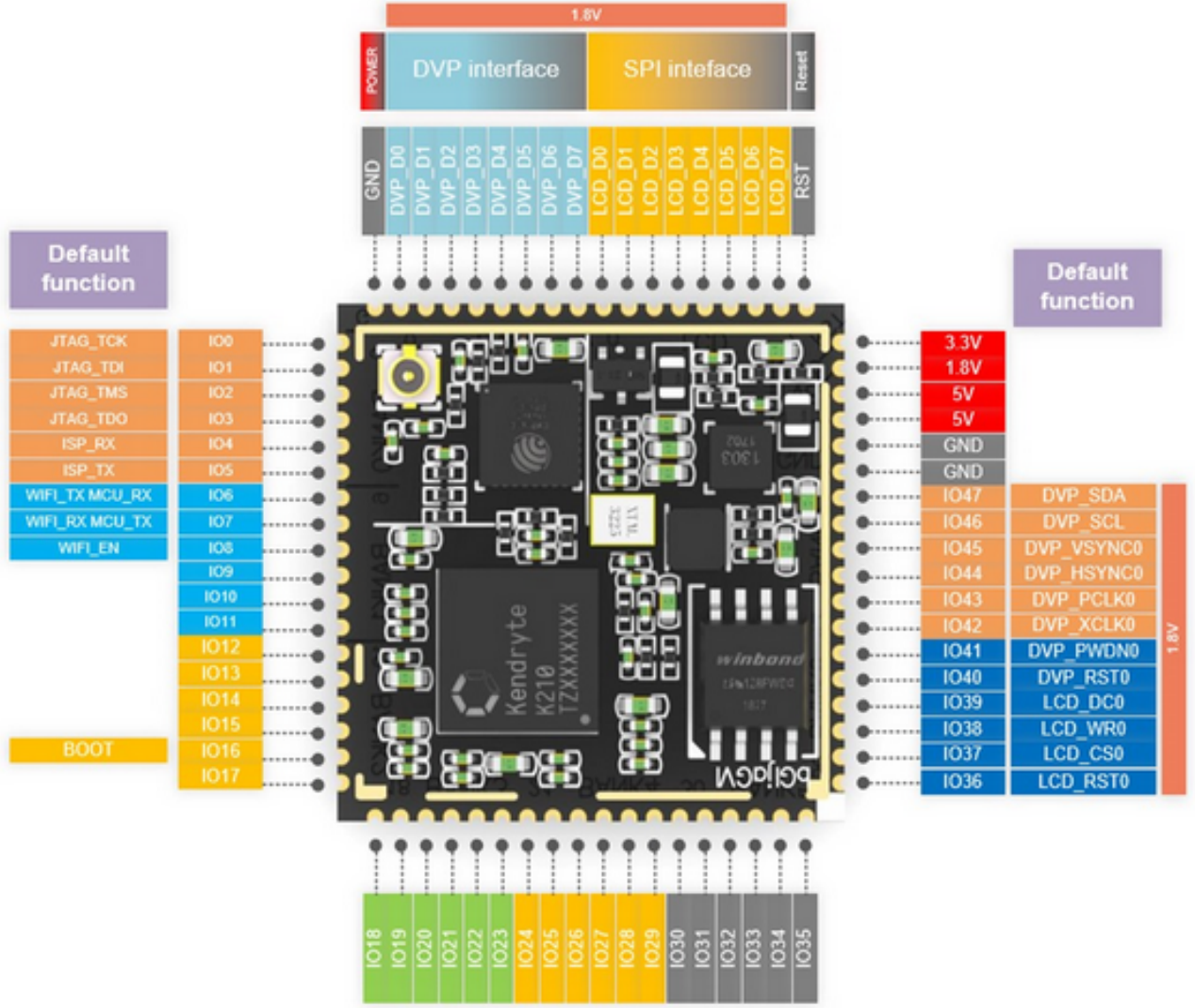


M1 CPU and Its 50 mil Pin Break Outs

https://www.seeedstudio.com/Sipeed-M1-dock-suit-M1-dock-2-4-inch-LCD-OV2640-K210-Dev-Board-1st-RV64-AI-board-for-Edge-Computing.html

M1 CPU IO breaks out as 1.27mm (50mil) pins, and pin's voltage is selectable from 3.3V and 1.8V.





Maxiduino Module Connectors

Kendryte: C/C++ Standalone SDK for K210

https://wiki.sipeed.com/en/maix/sdk/kendryte_sdk.html

Original standalone SDK base on C/C++ (NO RTOS)

kendryte / **kendryte-standalone-sdk**

https://github.com/kendryte/kendryte-standalone-sdk/commit/d1f33dd8bed7d4c81ef3c4bd153a4007e50bc375

Modify system default print uart

Modify system default print uart from github sample repo

```
/* Initialize bss data to 0 */
init_bss();
/* Init UART */
uart_init(UART_DEVICE_3);
uart_configure(UART_DEVICE_3, 115200, 8, UART_STOP_1, UART_PARITY_NONE);
uart_set_receive_trigger(UART_DEVICE_3, UART_RECEIVE_FIFO_1);
fpioa_set_function(4, FUNC_UART3_RX);
fpioa_set_function(5, FUNC_UART3_TX);
sys_register_getchar(uart3_getchar);
sys_register_putchar(uart3_putchar);
uart_debug_init(UART_DEVICE_3);
/* Init FPIOA */
fpioa_init();
/* Register finalization function */
```

RTOS (RT-Thread) for K210/M1

https://wiki.sipeed.com/en/maix/sdk/rtt.html

https://github.com/RT-Thread/rt-thread

RT-Thread RTOS like a traditional real-time operating system. The kernel has real-time multi-task scheduling, semaphore, mutex, mail box, message queue, signal etc. However, it has three different things: Device Drivers; Component; Dynamic Module.

1. The device driver is more like a driver framework, UART, IIC, SPI, SDIO, USB device/host, EMAC, MTD NAND etc. The developer can easily add low level driver and board configuration;

2. The Component is a software concept upon RT-Thread kernel, for example a shell (finsh/msh shell), virtual file system (FAT, YAFFS, UFFS, ROM/RAM file system etc), TCP/IP protocol stack (lwIP), POSIX (thread) interface etc. One component must be a directory under RT-Thread/Components and one component can be descripted by a SConscript file (then be compiled and linked into the system).

3. The Dynamic Module, formerly named as User Applicaion (UA) is a dynamic loaded module or library, it can be compiled standalone without Kernel. Each Dynamic Module has its own object list to manage thread/semaphore/kernel object which was created or initialized inside this UA. More information about UA, please visit another git repo.

RTOS (RT-Thread) Board Support Packages

https://github.com/RT-Thread/rt-thread

RT-Thread RTOS can support many architectures:

ARM Cortex-M0

ARM Cortex-M3/M4/7

ARM Cortex-R4

ARM Cortex-A8/A9

ARM920T/ARM926 etc

MIPS32

x86

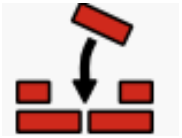
Andes

C-Sky

RISC-V

PowerPC

SCONS Building Tool for RTOS (RT-Thread)



https://www.scons.org/

SCons is an Open Source software construction tool—that is, a next-generation build tool. Think of SCons as an improved, cross-platform substitute for the classic Make utility with integrated functionality similar to autoconf/automake and compiler caches such as ccache. In short, SCons is an easier, more reliable and faster way to build software.

https://github.com/SCons/scons

User Space Code for RTOS on K210/M1

https://github.com/RT-Thread/rtthread-apps

Example 1. Hello, the world

```
#include <stdio.h>
int main(int argc, char **argv)
{
    printf("Hello World!\n");
    return 0;
}
```

Example 2. Create a sub-thread

```
#include <rtthread.h>
void my_thread_entry(void* parameter)
{
    int index;

    while (1)
    {
        rt_kprintf("index => %d\n", index ++);
        rt_thread_delay(RT_TICK_PER_SECOND);
    }
}
int my_thread_init(void)
{
    rt_thread_t tid;

    tid = rt_thread_create("tMyTask', my_thread_entry, RT_NULL,
        2048, 20, 20);
    if (tid != RT_NULL) rt_thread_startup(tid);

    return 0;
}
//This example creates a sub-thread, which named as 'tMyTask'.
```

Micropython

http://docs.micropython.org/en/latest/

GPIO with Micropython

https://maixpy.sipeed.com/en/libs/Maix/gpio.html

Example 1:

```
import utime
from Maix import GPIO
fm.register(board_info.LED_R,fm.fpioa.GPIO0)
led_r=GPIO(GPIO.GPIO0,GPIO.OUT)
utime.sleep_ms(500)
led_r.value()
fm.unregister(board_info.LED_R,fm.fpioa.GPIO0)
```

Example 2:

```
import utime
from Maix import GPIO
fm.register(board_info.LED_R,fm.fpioa.GPIO0)
led_r=GPIO(GPIO.GPIO0,GPIO.IN)
utime.sleep_ms(500)
led_r.value()
fm.unregister(board_info.LED_R,fm.fpioa.GPIO0)
```

Example 3: GPIO with ExINT

```
import utime
from Maix import GPIO
def test_irq(GPIO,pin_num):
    print("key",pin_num,"\n")
fm.register(board_info.BOOT_KEY,fm.fpioa.GPIOHS0)
key=GPIO(GPIO.GPIOHS0,GPIO.IN,GPIO.PULL_NONE)
utime.sleep_ms(500)
key.value()
key.irq(test_irq,GPIO.IRQ_BOTH,GPIO.WAKEUP_NOT_SUPPORT,7)
key.disirq()
fm.unregister(board_info.BOOT_KEY,fm.fpioa.GPIOHS0)
```

MaixPy Set Up

source: https://www.youtube.com/watch?v=KResVuAIMb4
https://maixpy.sipeed.com/en/
http://blog.sipeed.com/p/category/maix-software/maixpy

Step 1: Check the Sipeed board Serial connection, open the Terminal and run this command:

dmesg | grep tty

```
mthndud@mthndud-Blade:~$ dmesg | grep tty
[    0.000000] console [tty0] enabled
[    6.764609] dw-apb-uart.1: ttyS4 at MMIO 0x72001000 (irq = 20, base_baud = 11
5200) is a 16550A
[  178.758789] usb 1-3: ch341-uart converter now attached to ttyUSB0
[  436.374006] ch341-uart ttyUSB0: ch341-uart converter now disconnected from tt
yUSB0
[  440.330562] usb 1-1: ch341-uart converter now attached to ttyUSB0
```

In this case, it is ttyUSB0

Step 2: Install Screen application:

sudo apt-get install screen

Step 3: After Screen is installed, we do access to the Sipeed b

sudo screen /dev/ttyUSB0 115200

Press Ctrl +C, then Ctrl + D to reboot the board

```
 __  __                  _____  __   __  _____   __     __
|  \/  |     /\         |_   _| \ \ / / |  __ \  \ \   / /
| \  / |    /  \          | |    \ V /  | |__) |  \ \_/ /
| |\/| |   / /\ \         | |     > <   |  ___/    \   /
| |  | |  / ____ \       _| |_   / . \  | |         | |
|_|  |_| /_/    \_\     |_____| /_/ \_\ |_|         |_|

Official Site : https://www.sipeed.com
Wiki          : https://maixpy.sipeed.com

Traceback (most recent call last):
  File "_boot.py", line 68, in <module>
OSError: -10002
MicroPython v0.3.2 on 2019-05-15; Sipeed_M1 with kendryte-k210
Type "help()" for more information.
>>>
```

MaixPY example program

Step 4:
Type this program to MaixPy:

```
import sensor
import image
import lcd

lcd.init()
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.run(1)
while True:
        img=sensor.snapshot()
        lcd.display(img)
```

Then press Enter 3 times, and we can see the Videos Streaming from the Camera.

MaixPY Firmware Download

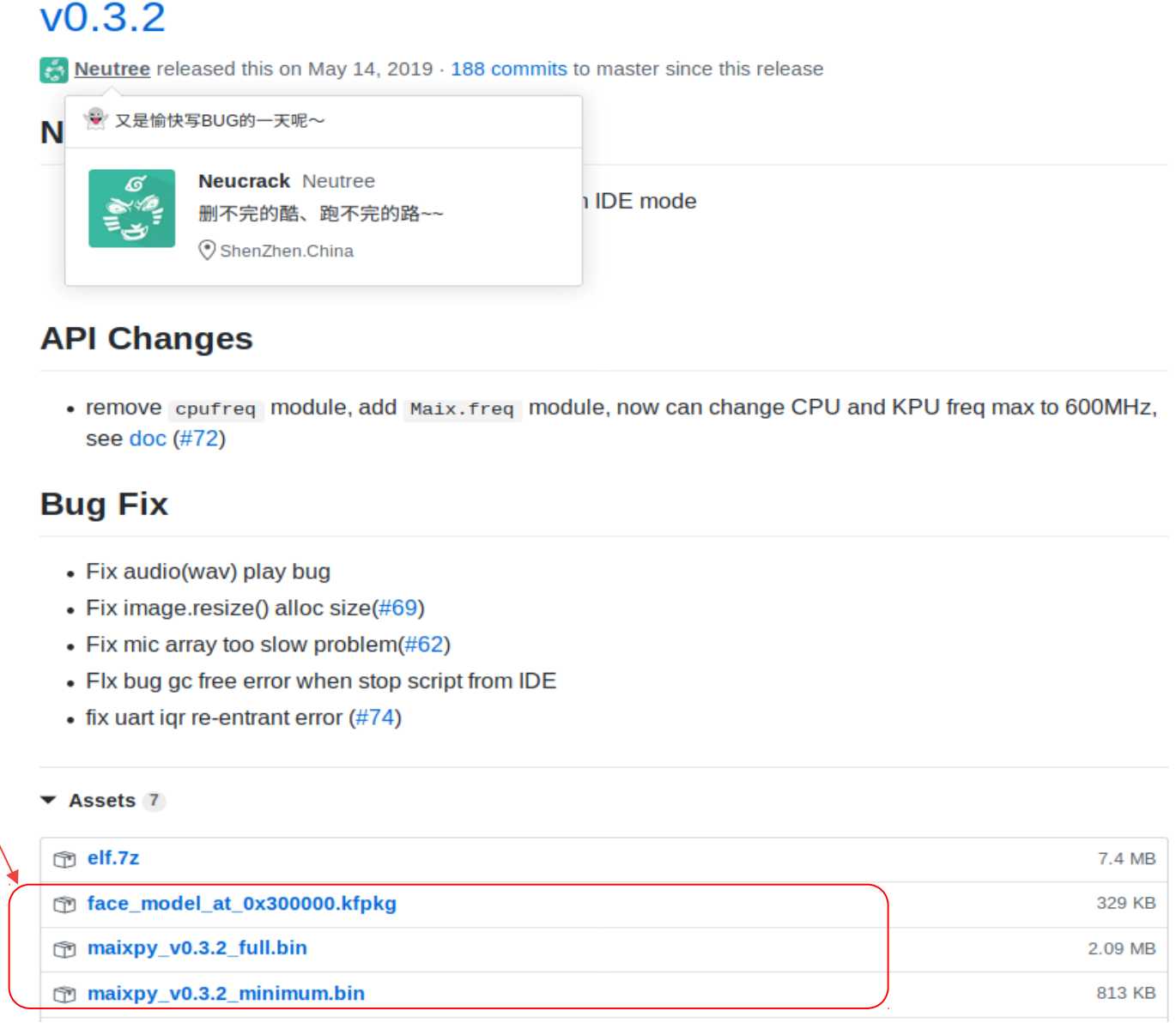
Step 5: Go to https://github.com/sipeed/MaixPY/releases, scroll down to 0.3.2 and download 3 files:

1.face_model_at_0x300000.kfpkg

2.maixpy_v0.3.2_full.bin

3.maixpy_v0.3.2_minimum.bin

HL: 2020-Feb-6, the 3rd one, minimum.bin has bug per Minh's input, so not to down load



KFlash program to flash binary files

Step 6: Go to https://github.com/sipeed/kflash_gui/releases, download kflash v1.5 for Linux:

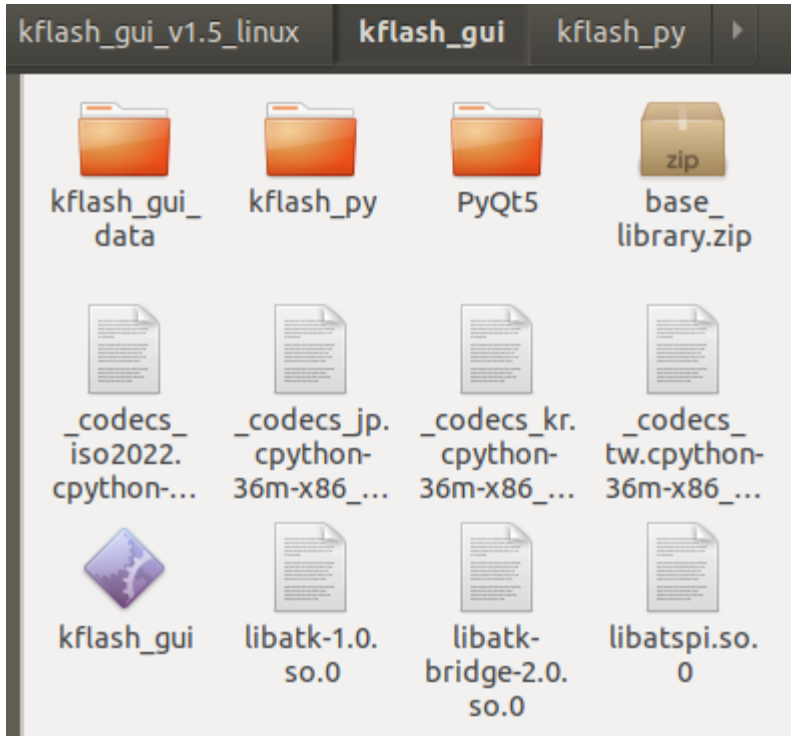
- kflash_gui_v1.5.1_linux.tar.xz
- kflash_gui_v1.5.1_macOS.dmg

HL: Version 1.5.1 has been used as the latest

Download this for Linux

KFlash program to flash binary files

Step 6.1: Decompress the “kflash_gui_v1.5_linux”, and go to kflash_gui.



Step 6.2: Then open a Terminal Window in this folder and run the klash_gui with sudo mode:

sudo ./kflash_gui

MaixPY Firmware Flashing

Step 7: Then open the kflash_gui application and choose options as below

Step 7.1: On “Open File” and select:

maixpy_v0.3.2_full.bin

Step 7.2: On “Board” select:

Sipeed Maix Dock

Step 7.3: On “Burn To” select:

Flash

Step 7.4: On “Port” select:

/dev/ttyUSB0

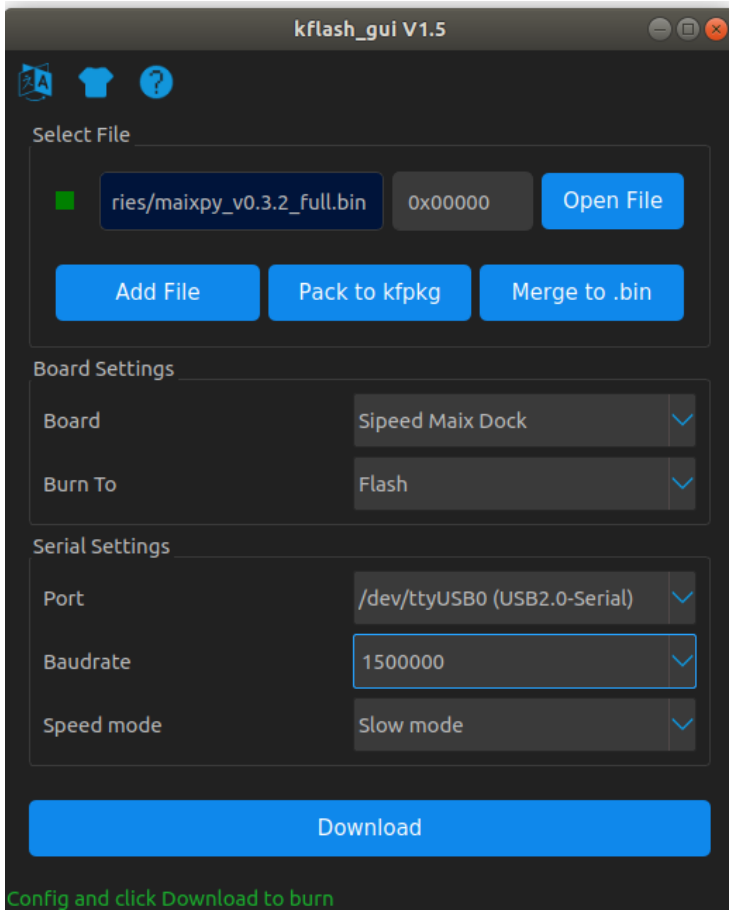
Step 7.5: On “Baud Rate” select:

150000

Step 7.6: On “Speed Mode” select:

Slow Mode

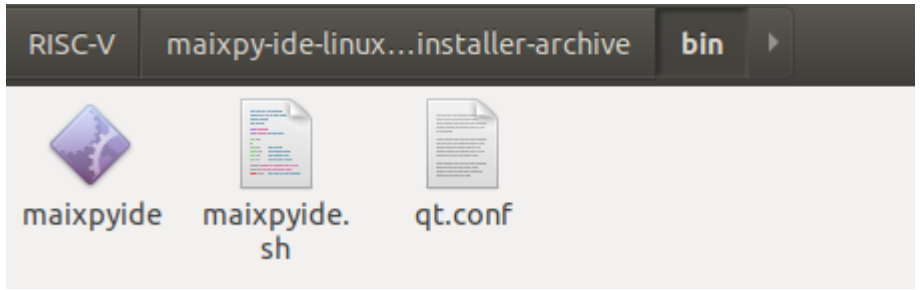
Step 7.7: Click Download to flash the firmware to the board.



MaixPY IDE program

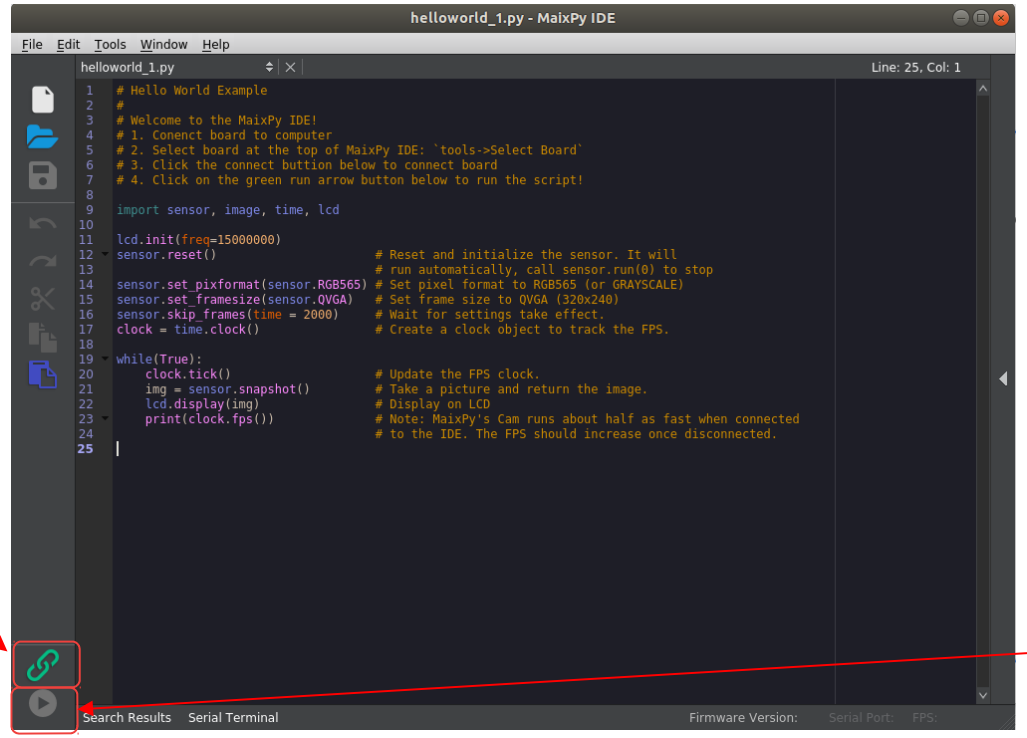
Step 8: Go to http://dl.sipeed.com/MAIX/MaixPy/ide/v0.2.2 and download

maixpy-ide-linux-x86_64-0.2.2-installer-archive.7z



Step 8.1: The decompress, go to ./maixpy-ide-linux-x86_64-0.2.2-installer-archive/bin:

Open Terminal Window and run: sudo ./maixpyide



Click this to link to the board

Click this to run the program

MaixPY Face Detection

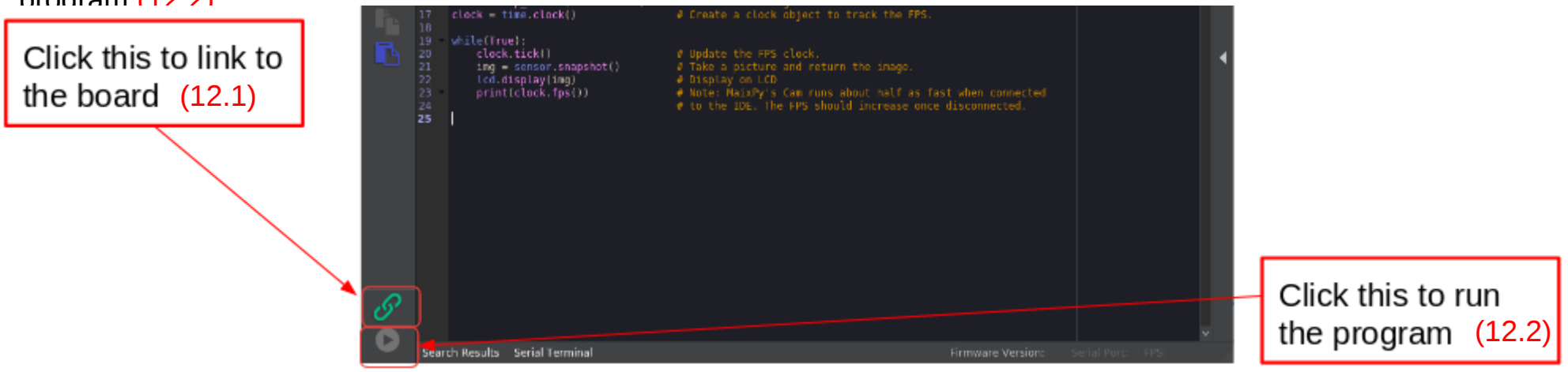
Step 9: Go to https://github.com/sipeed/MaixPY/releases and clone whole folder by these following steps:

9.1 Open Terminal Window
9.2 cd $HOME/Documents
9.3 mkdir MaixPy
9.4 cd MaixPy
9.3 git clone https://github.com/sipeed/MaixPy_scripts.git

Step 10: Then from the MaixPy IDE, on the top left corner, select
File > Document Folder > MaixPy > MaixPy_scripts > machine_vision > demo_find_face

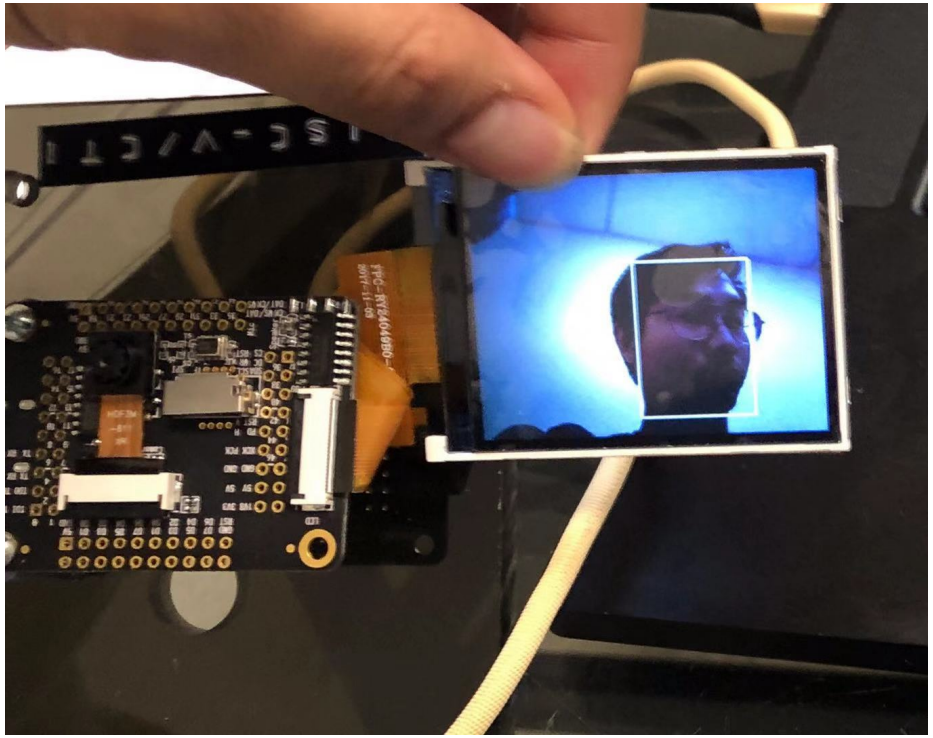
Step 11: Repeat step 7 to flash face_model_at_0x30000.kfpkg to the board

Step 12: On the MaixPy IDE, when the demo_find_face program is opened, click button to link to the board (12.1) and run the program (12.2)



MaixPY Face Detection Result

After flashing the face_model_at_0x30000.kfpkg by the kflash v1.5 program, and run the face detection program , we have the result as shown:



MaixPY Reference

https://maixpy.sipeed.com/en/get_started/edit_file.html

MaixPy file system

Command	Description	Example
`os.chdir()`	changes the current directory	`os.chdir("/flash")`
`os.listdir()`	list the files in the current directory	`os.listdir()`
`os.listdir(path)`	list the files in another directory	`os.listdir("/sd")`
`os.getcwd()`	return the current working directory	`os.getcwd()`
`os.rename(old_path, new_path)`	rename a file	`os.rename("./blue.py", "./aaah.py")`
`os.remove(path)`	remove a file	`os.remove("./herring.py")`

Editing and saving files

Linux users are recommended to use minicom.
Use sudo minicom -s to set the reference to the previous tutorial

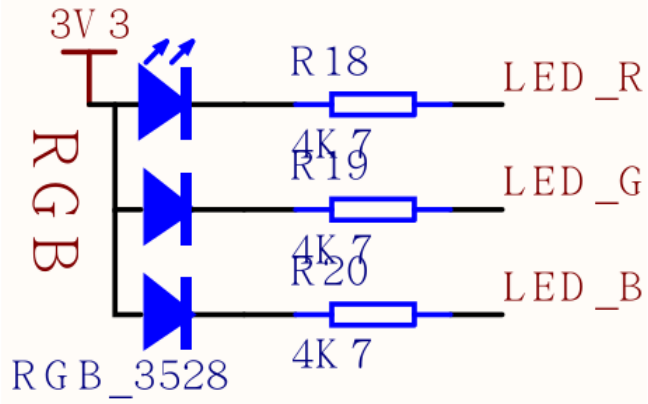
Read/edit file on laptop then load to the board

Execute program/file when loaded to the board

Hello the World

https://maixpy.sipeed.com/en/get_started/edit_file.html

Board K210




```
from Maix import GPIO

fm.register(board_info.LED_R, fm.fpioa.GPIO0)

led_r=GPIO(GPIO.GPIO0,GPIO.OUT)
led_r.value(0)
```

Click the lines of the code one by one from the keyboard inside the terminal and press OK to execute. Start with the package Maix introduced GPIO this class;

Front pin can be set K210, so we use .fm(fpioa manager) correspondence between peripherals and pin registration chip built-in object to this, here fm.fpioa.GPIO0 is a GPIO Peripheral K210's (Note the difference between GPIO (peripheral) and pin (real hardware pin)), so the fm.fpioa.GPIO0 registration to pin board_info.LED_R;

The board_info is a board type information can be entered in serial terminal board_info. then press TAB the button to see all the members, each pin largely value

Then define a GPIO subject, specific parameters to see GPIO the module's documentation, look in the left sidebar.

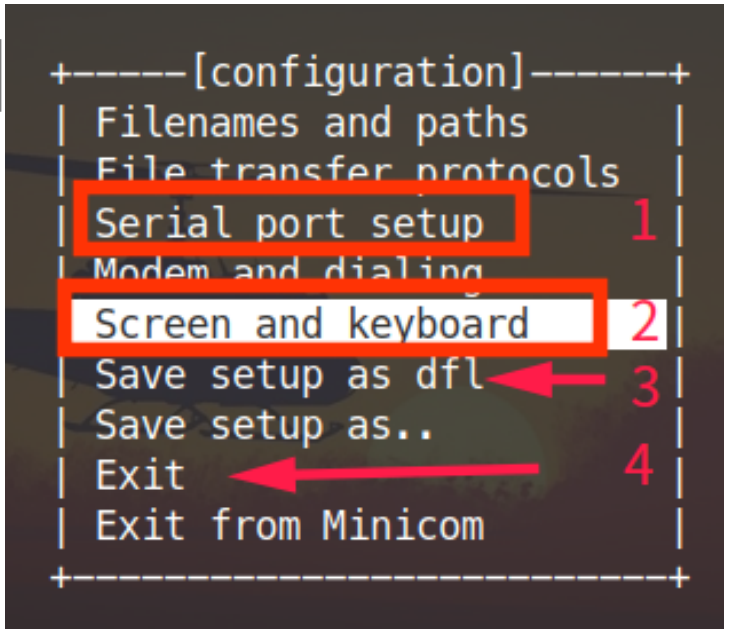
Use led_r.value(0) or led_r.value(1) to set high to low

Communications to The Board

https://maixpy.sipeed.com/en/get_started/power_on.html

```
sudo apt update
sudo apt install minicom
sudo minicom -s
# Then set the serial port no, the baud rate is 115200.
# Set Backspace to DEL function
# Set linewrap to Yes
sudo minicom
```

1

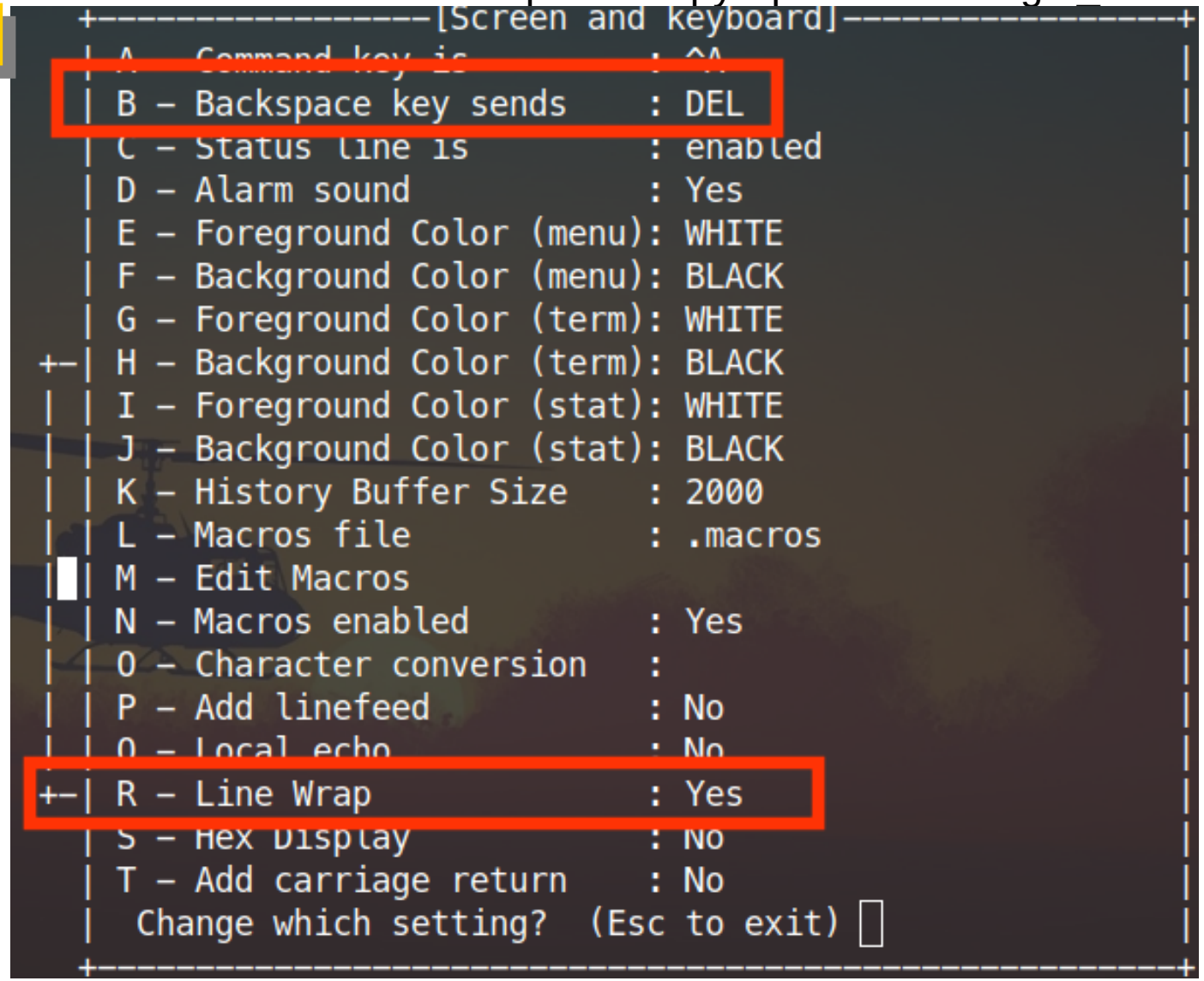


2

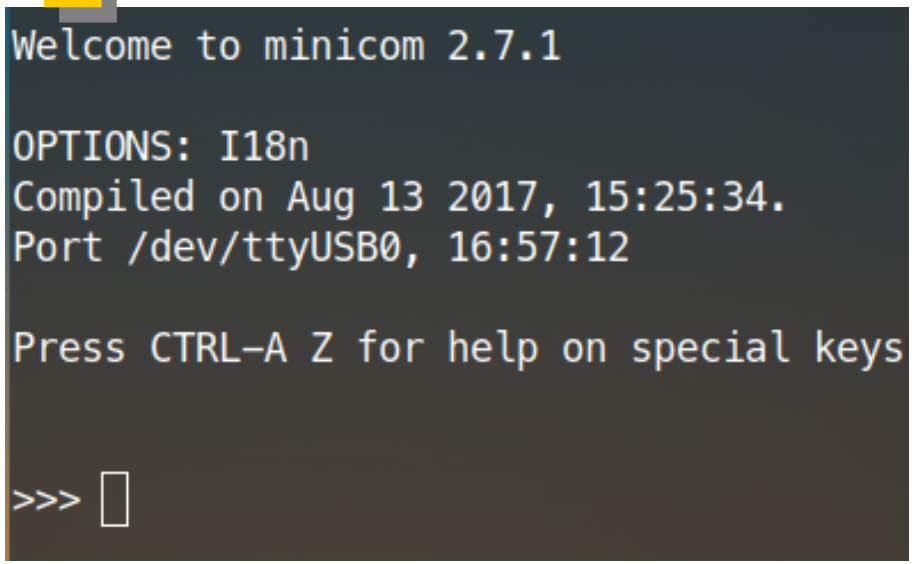
Communications to The Board (2)

https://maixpy.sipeed.com/en/get_started/power_on.html

3



4



Boot Script

https://maixpy.sipeed.com/en/get_started/boot.html

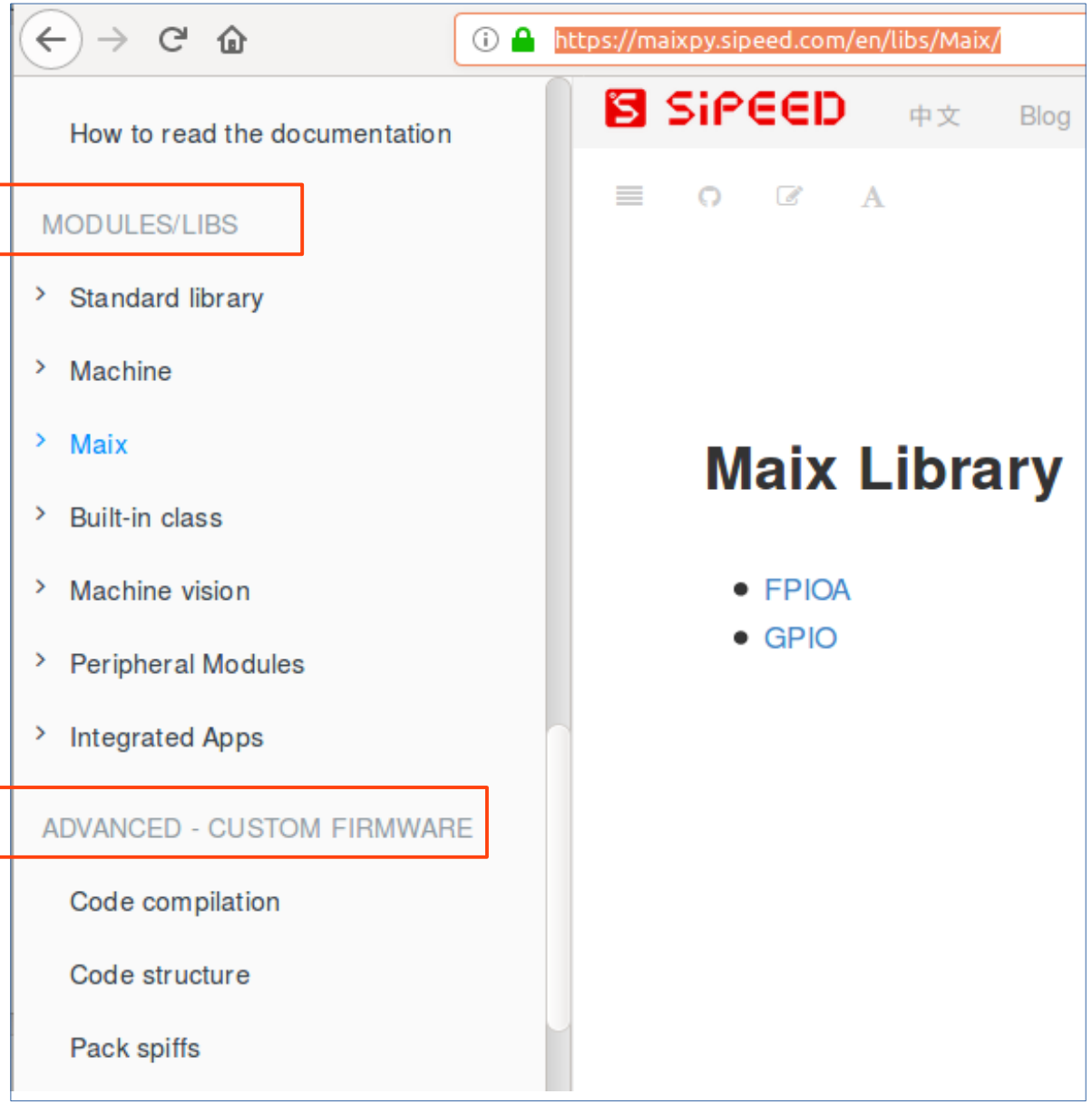
If a memory card formatted with either a SPIFFS or FAT file system is detected at boot, MaixPy will look for a boot.py file in the root directory of that memory card. If /boot.py is found, the system will immediately execute it.

Memory cards must be formatted with a SPIFFS or FAT filesystem, not FAT32.

FAT memory cards will have a mount point of /sd, while SPIFFS cards will appear as /flash.

References

https://maixpy.sipeed.com/en/libs/Maix/



GPIO and Exint

https://maixpy.sipeed.com/en/libs/Maix/gpio.html

FPIOA (Field Programmable Input and Output Array)

GPIOHS	Function
GPIOHS31	LCD_DC
GPIOHS30	LCD_RST
GPIOHS29	SD_CS
GPIOHS28	MIC_LED_CLK
GPIOHS27	MIC_LED_DATA

Do not use these GPIO pins, they have been assigned for the interface functions shown in this table

ExINT: Configure an interrupt handler to call when the pin's trigger source is active. If the pin mode is pin.in, the trigger source is an external value on the pin.

Example: From the website demo1

```
import utime
from Maix import GPIO
fm.register(board_info.LED_R,fm.fpioa.GPIO0)
led_r=GPIO(GPIO.GPIO0,GPIO.OUT)
utime.sleep_ms(500)
led_r.value()
fm.unregister(board_info.LED_R,fm.fpioa.GPIO0)
```

KPU for CNN

KPU is a general-purpose neural network processor, which can do convolutional neural network calculation at low power consumption, for example obtain the size, coordinates and types of detected objects or detect and classify faces and objects.

KPU features:

Supports fixed-point models trained by mainstream framework with some restrictions
There is no direct limit on the number of network layers. It supports separate configuration of each layer of convolutional neural network parameters, including the number of input and output channels, input and output line width and column height.
Supports two convolution kernels 1x1 and 3x3
Support any form of activation function
The maximum supported neural network parameter size in real-time work is 5.5MiB to 5.9MiB
Maximum supported network parameter size when working in non-real time (Flash capacity - software volume

```
import KPU as kpu
task = kpu.load(offset or file_path)
```

Initializing the yolo2 network, by passing initialization parameters to the yolo2 network model

1

```
import KPU as kpu
task = kpu.load(offset or file_path)
anchor = (1.889, 2.5245, 2.9465, 3.94056, 3.99987, 5.3658, 5.155437, 6.92275, 6.718375, 9.01025)
kpu.init_yolo2(task, 0.5, 0.3, 5, anchor)
```

Yolo2

2

```
import KPU as kpu
import image
task = kpu.load(offset or file_path)
anchor = (1.889, 2.5245, 2.9465, 3.94056, 3.99987, 5.3658, 5.155437, 6.92275, 6.718375, 9.01025)
kpu.init_yolo2(task, 0.5, 0.3, 5, anchor)
img = image.Image()
kpu.run_yolo2(task, img) #This is not right, please refer to the routine
```

Face Recognition Model

https://maixpy.sipeed.com/en/libs/Maix/kpu.html

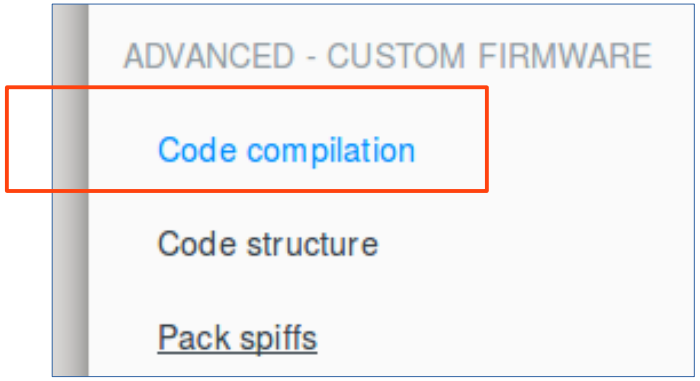
Model download address: http://dl.sipeed.com/MAIX/MaixPy/model/face_model_at_0x300000.kfpkg

```
import sensor
import image
import lcd
import KPU as kpu

lcd.init()
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.run(1)
task = kpu.load(0x300000) #use kfpk to flash kmodel and maixpy
anchor = (1.889, 2.5245, 2.9465, 3.94056, 3.99987, 5.3658, 5.155437, 6.92275, 6.718375, 9.01025)
a = kpu.init_yolo2(task, 0.5, 0.3, 5, anchor)
while(True):
    img = sensor.snapshot()
    code = kpu.run_yolo2(task, img)
    if code:
        for i in code:
            print(i)
            a = img.draw_rectangle(i.rect())
    a = lcd.display(img)
a = kpu.deinit(task)
```

RTOS

https://maixpy.sipeed.com/en/advanced/compile.html



Build Maixpy from Source Code

https://github.com/sipeed/MaixPy/blob/master/build.md

Download source code

the source ports/k210-freertos directory README

Download toolchain

Make

Flash

Spiffs (SPI Flash File System) Tool

https://github.com/pellepl/spiffs

GitHub, Inc. (US)

Spiffs is a file system intended for SPI NOR flash devices on embedded targets. ... It can run on any NOR flash, not only SPI flash - theoretically also on embedded flash of a microprocessor. Multiple spiffs configurations can run on same target - and even on same SPI flash device.

Spiffs is a file system intended for SPI NOR flash devices on embedded targets.

Spiffs is designed with following characteristics in mind:

- Small (embedded) targets, sparse RAM without heap
- Only big areas of data (blocks) can be erased
- An erase will reset all bits in block to ones
- Writing pulls one to zeroes
- Zeroes can only be pulled to ones by erase
- Wear leveling

Posix-like api: open, close, read, write, seek, stat, etc