WE-I Plus Basic Tutorial

May, 2022

Himax Technologies, Inc.
奇景光電股份有限公司

Outline

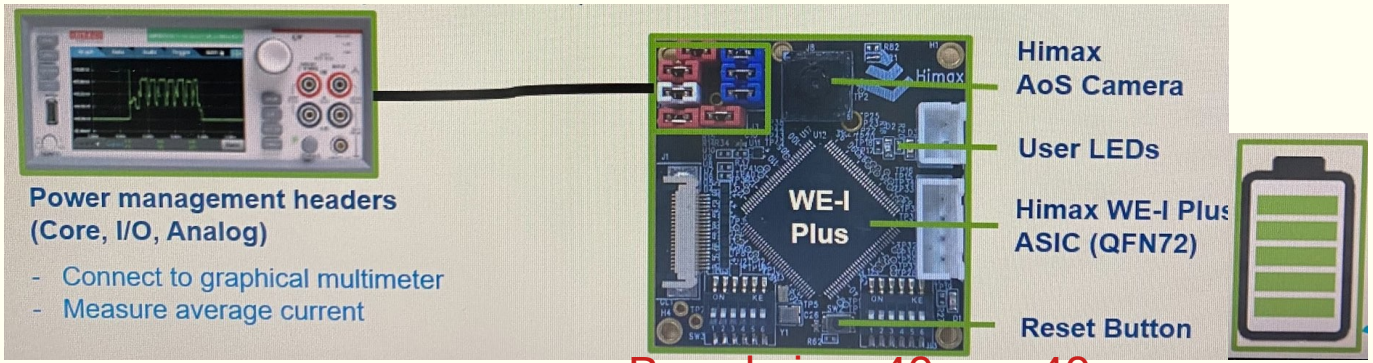
- WE-I Plus Turnkey Solutions
- WE-I Plus AI Processor Block Diagram
- Hardware Interface
- Built with example
- Deploy to WE-I
- SDK API
- Hand-on example
- Q&A
- Appendix

WE-I Plus Turnkey Solutions

- HIMAX website:
  - https://www.himax.com.tw/zh/products/intelligent-sensing/always-on-smart-sensing/application-solutions/
- Automatic Meter Reading
- People Counter
- Smart Tripod Head
- Smart Door bell
- Battery Camera
- Dashcam Parking mode

WE-I Plus - Ultra low power tiny ML inference

- Power measurement use case – 1FPS TFLM “Person detection”
  - “Person detection” NN model
  - Image size: 96x96
  - Weight size:250KB
  - OPs/Inference: 60M
- Use case scenario
  - 1FPS, periodic wakeup
- Measured WE-I average power
  - 2.2mW (on-board DCDC)
  - 3.7mW (internal LDO)



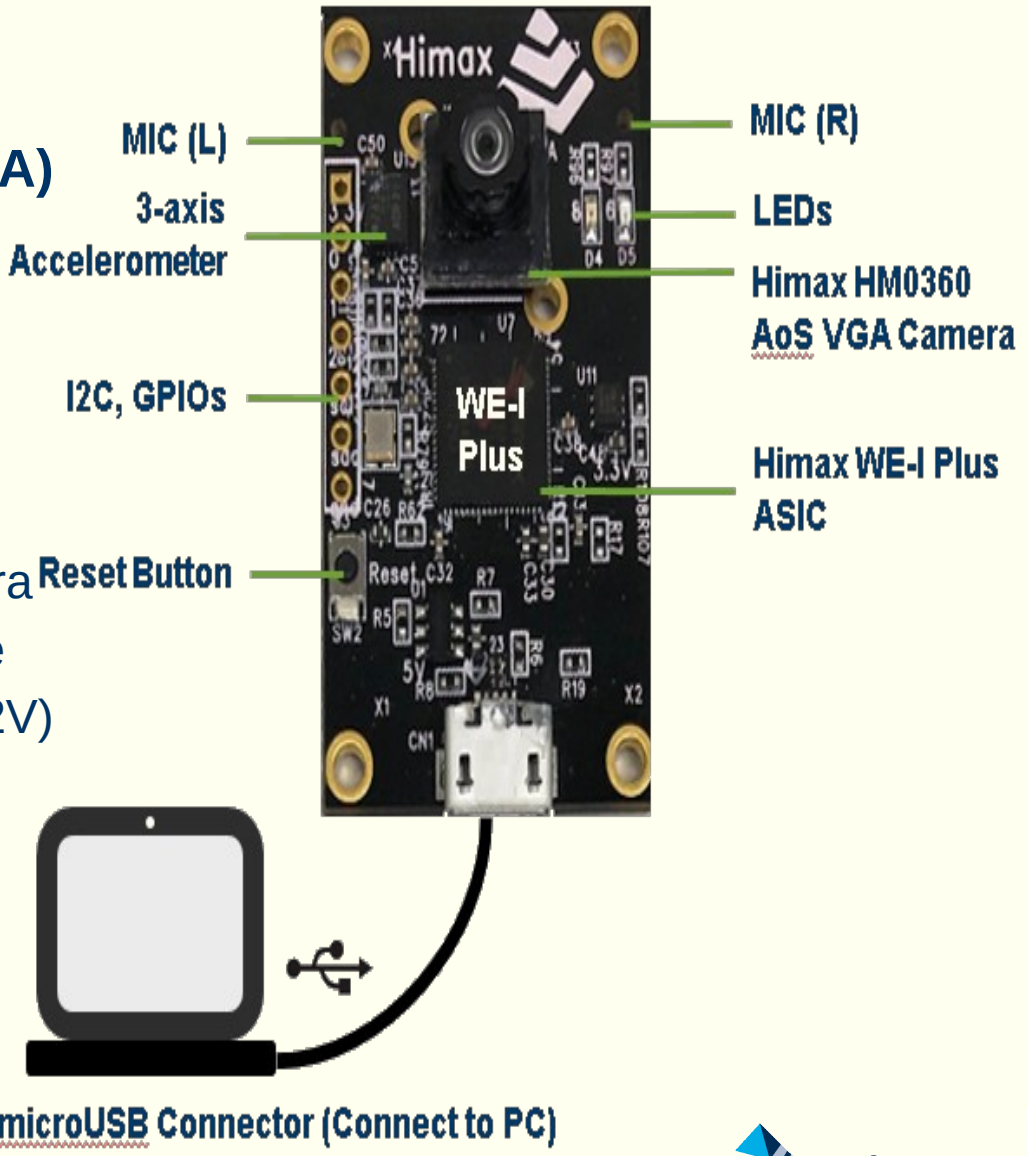
Battery Life

Broad size: 40mmx40mm

Hardware Interface

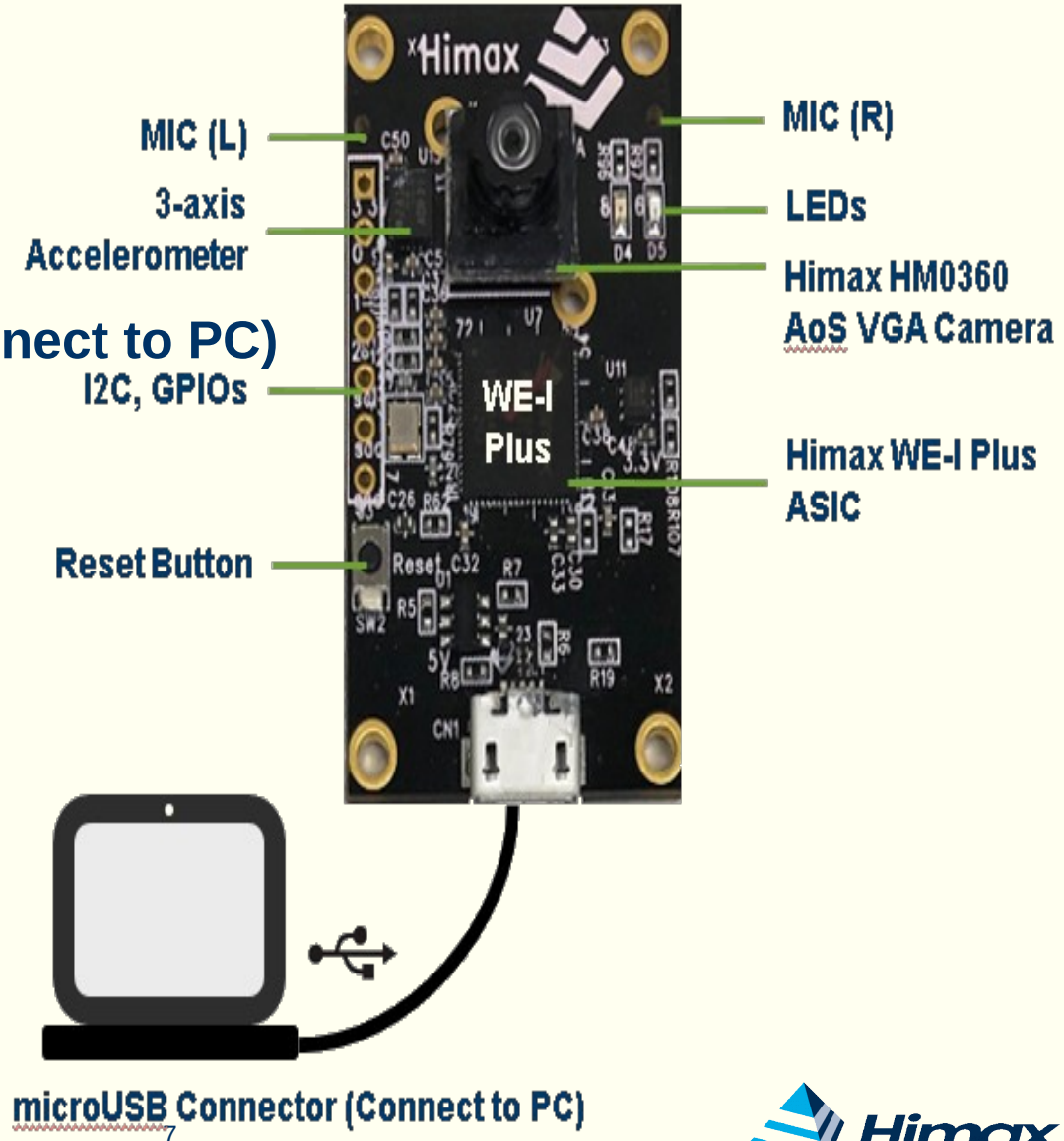
Hardware Interface

- **Board size**: 40mm x 27mm
- **Himax WE-I Plus ASIC (HX6537-A)**
  - ARC 32-bit EM9D DSP with FPU
  - Max. 400MHz clock frequency
  - 2MB SRAM
  - 2MB Flash
- **On Board**
  - Himax HM0360 AoS™ VGA camera
  - FTDI USB to SPI/I²C/UART bridge
  - LDO power supply (3.3/2.8/1.8/1.2V)
  - 3-axis accelerometer (LSM9DS1)
  - 1x reset button
  - 2x microphones (L/R)
  - 2x user LEDs (RED/GREEN)
  - microUSB connector



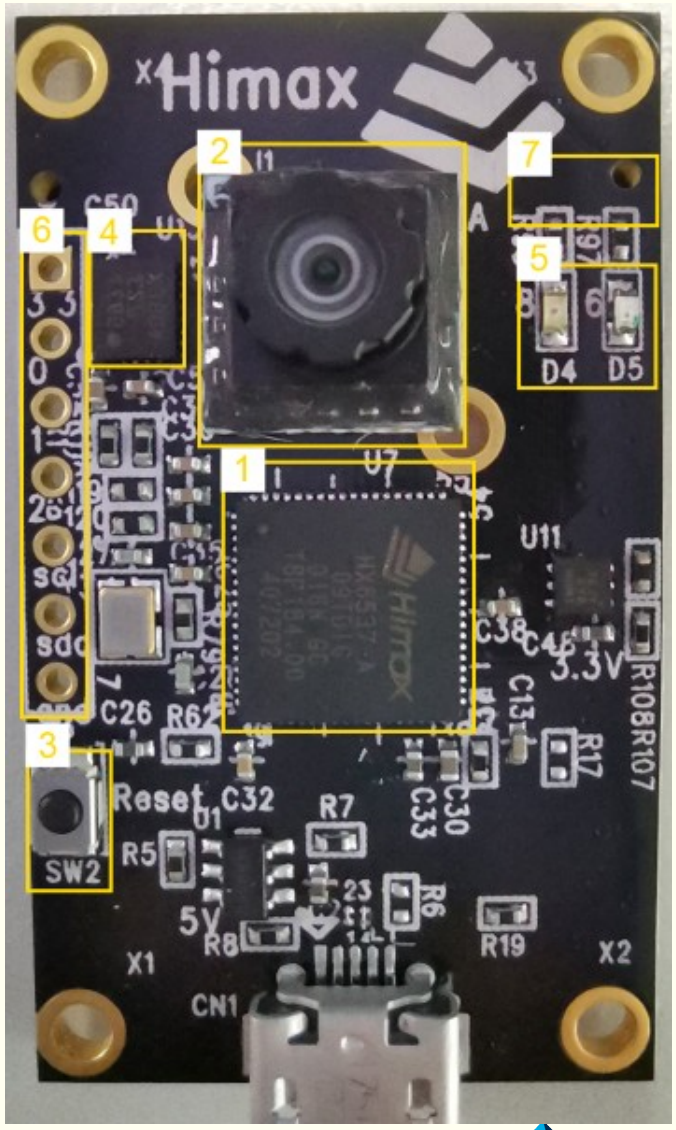
Hardware Interface

- **Expansion Header**
  - 1x I²C
  - 3x GPIOs
  - Power/Ground
- **microUSB Connector (Connect to PC)**
  - Flash programming
  - Output message



Hardware Interface

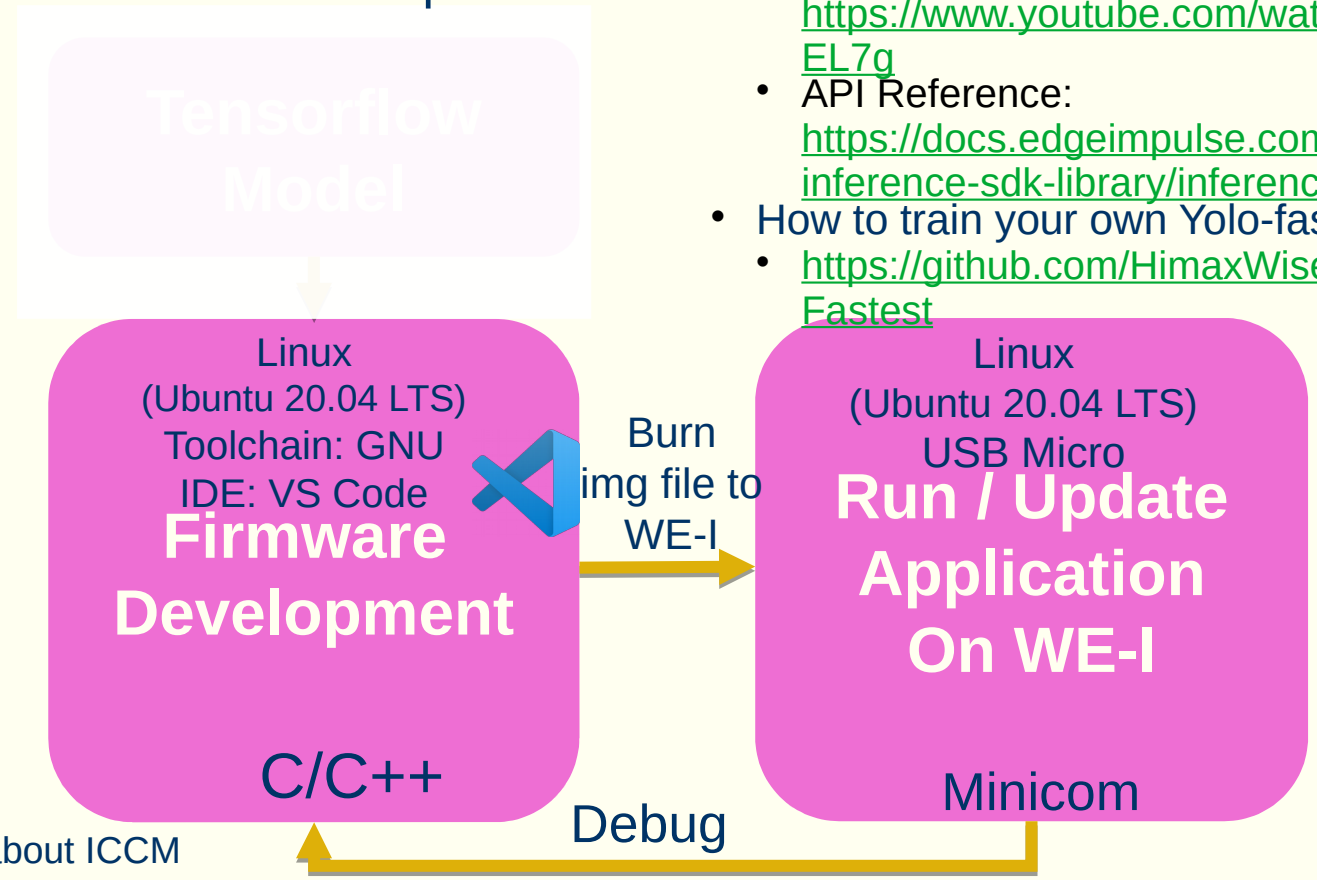
1. CPU HX6537-A
2. HM0360 AoS™ VGA camera
   - Active Pixel Array: 640 x 480
   - Frame Rate: 60FPS
3. Reset Button
4. LSM9DS1 IMU sensor
   - 3 acceleration: ±2 / ±4 / ±8 / ±16 g
5. Green and Red LED
6. Expansion Header
   - Pin1: 3V3
   - Pin2: GPIO0
   - Pin3: GPIO1
   - Pin4: GPIO2
   - Pin5: I2C_M1_SCL
   - Pin6: I2C_M1_SDA
   - Pin7: GND
7. Microphones (L/R) at back side



Built with example

Built with example

- WE-I Firmware Development Flow



- Train model Resource
  - Edge impulse: https://www.edgeimpulse.com
  - Tutorial: Getting Started with the Himax WE-I Plus and Edge Impulse
  - Object detection video: https://www.youtube.com/watch?v=iazSrguEL7g
  - API Reference: https://docs.edgeimpulse.com/reference/c++-inference-sdk-library/inferencing-sdk
- How to train your own Yolo-fastest model:
  - https://github.com/HimaxWiseEyePlus/Yolo-Fastest

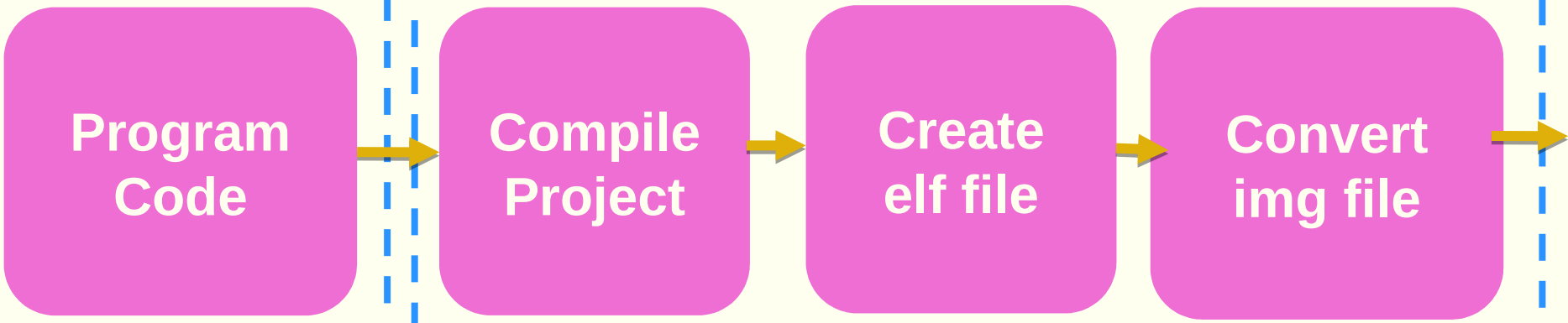
- Be careful about ICCM size.
- Don't use big memory size library such as iostream.

Built with example

- WE-I Firmware Development Flow – Linux environment

Use Himax SDK API for C or C++ program
IDE: VS Code

Make sure $make tool version >= 3.82
Toolchain: GNU

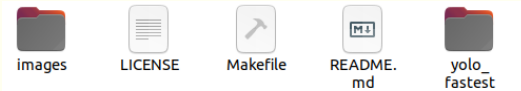


Build with example – Himax Resource

- Train model Resource
  - Edge impulse: https://www.edgeimpulse.com
  - Tutorial: Getting Started with the Himax WE-I Plus and Edge Impulse
  - Object detection video: https://www.youtube.com/watch?v=iazSrguEL7g
  - API Reference: https://docs.edgeimpulse.com/reference/c++-inference-sdk-library/inferencing-sdk
- How to train your own Yolo-fastest model:
  - https://github.com/HimaxWiseEyePlus/Yolo-Fastest
- Himax yolo-fastest person detection example:
  - https://github.com/HimaxWiseEyePlus/WE_I_Plus_User_Examples/tree/main/HIMAX_Yolo_Fastest_Person_Detection_Example
- SDK API 3 examples:
  - https://github.com/HimaxWiseEyePlus/bsp_tflu/tree/master/HIMAX_WE1_EVB_example/scenario_app/workshop_example
- Himax google tflm 4 examples:
  - https://github.com/HimaxWiseEyePlus/himax_tflm
- WE-I Plus Expansion Qwiic sensor (Sparkfun) example:
  - https://github.com/HimaxWiseEyePlus/bsp_tflu/tree/master/HIMAX_WE1_EVB_example/scenario_app/co2_person_lora_example

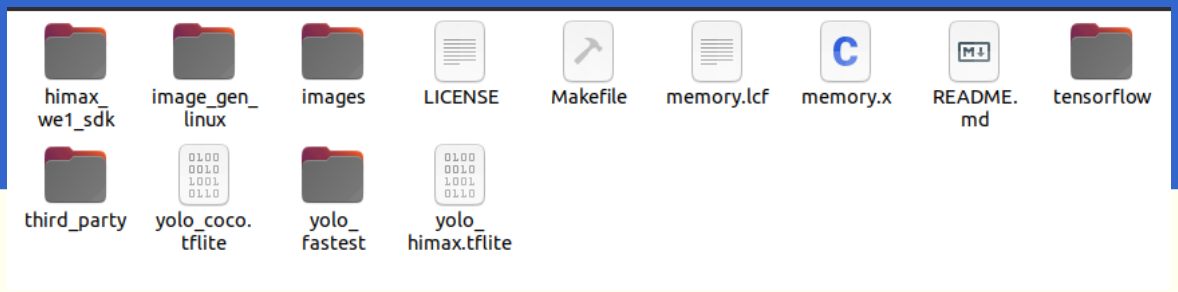
Built with example

- Download the yolo fastest examples from Himax github.
  - $ git clone https://github.com/HimaxWiseEyePlus/WE_I_Plus_User_Examples
- Check ‘make’ tool version >= 3.82 and install ‘curl’ command.
  - $ sudo apt update
  - $ sudo apt upgrade
  - $ sudo apt install curl
- Install the development toolkit:
  - ARC GNU Development Toolkit https://github.com/foss-for-synopsys-dwc-arc-processors/toolchain/releases/download/arc-2020.09-release/arc_gnu_2020.09_prebuilt_elf32_le_linux_install.tar.gz
  - After download it, please remember to add it to environment PATH.
    - (export PATH=[location of your ARC_GNU_ROOT]/bin:$PATH)
    - Example:
      - gedit ~/.bashrc
      - export PATH=/home/kris/ARC/MetaWare/arc/bin:$PATH
      - source ~/.bashrc



- Go under the folder of yolo fastest person detection example
  - $ cd WE_I_Plus_User_Examples/HIMAX_Yolo_Fastest_Person_Detection_Example

Built with example



- Download related third party, tflite model and library data (only need to download once)
  - $ make download
- If you are trying to build example with GNU toolkit. please check the ARC_TOOLCHAIN defined in Makefile.
  - #ARC_TOOLCHAIN ?= mwdt
  - ARC_TOOLCHAIN ?= gnu
- Build the example with

▪ Model trained by COCO dataset:

We default use the model , `yolo_coco.tflite` , trained by COCO dataset, just key-in following command on the console. Flash image name will be `yolo_coco*.img` .

```
make yolo_coco
make flash example=yolo_coco
```

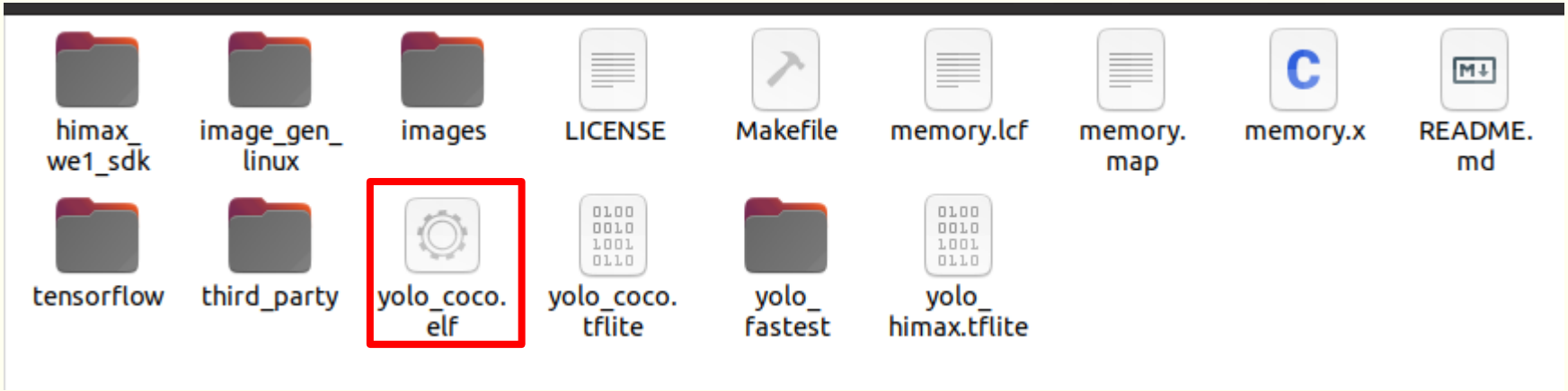
▪ Model trained by HIMAX dataset:

▪ If you want to build the example with `yolo_himax.tflite` . Just key-in following command on the console. Flash image name will be `yolo_himax*.img` .

```
make yolo_himax
make flash example=yolo_himax
```

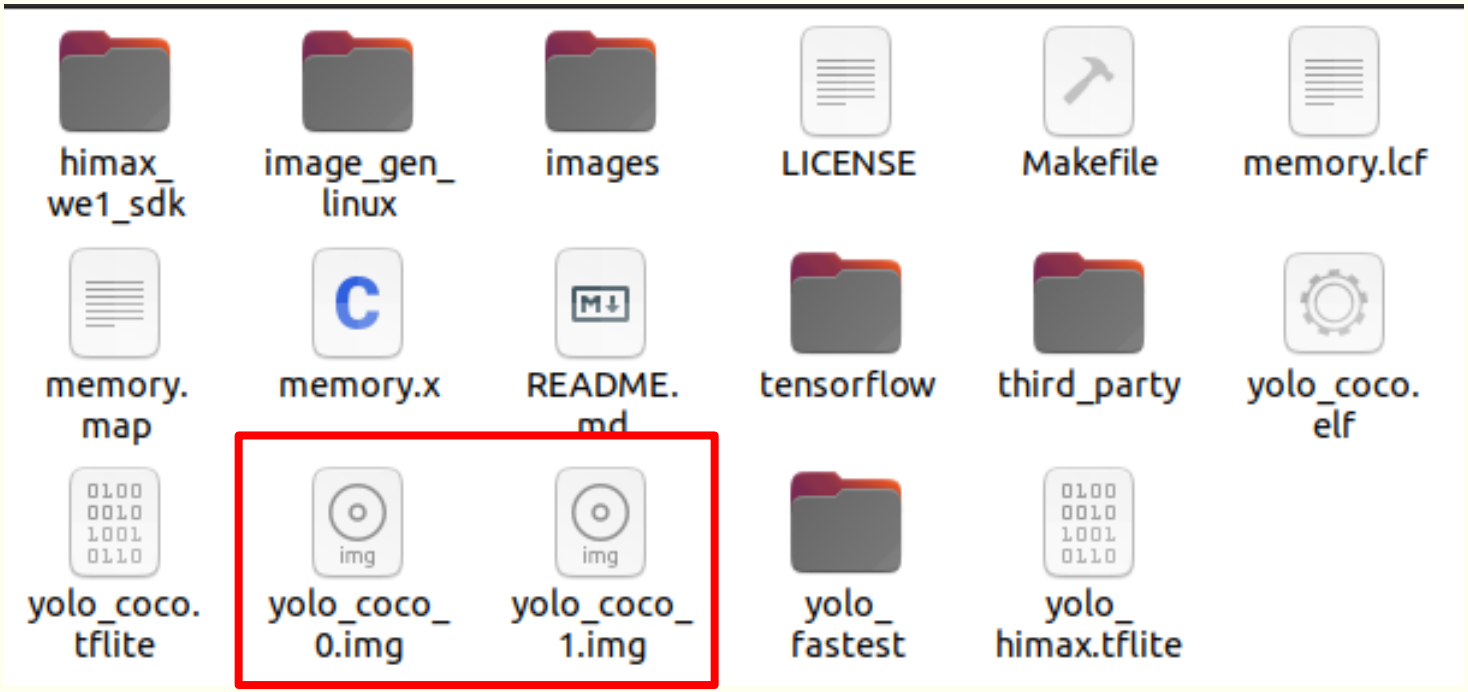
Built with example

- If you success to built with example, you will get the elf file.
  - $ make yolo_coco
  - output: yolo_coco.elf



Built with example

- If you success to generate flash image file, you will get the img file.
  - $ make flash example=yolo_coco
  - output: yolo_coco_0.img and yolo_coco_1.img



Deploy to WE-I

Deploy to WE-I

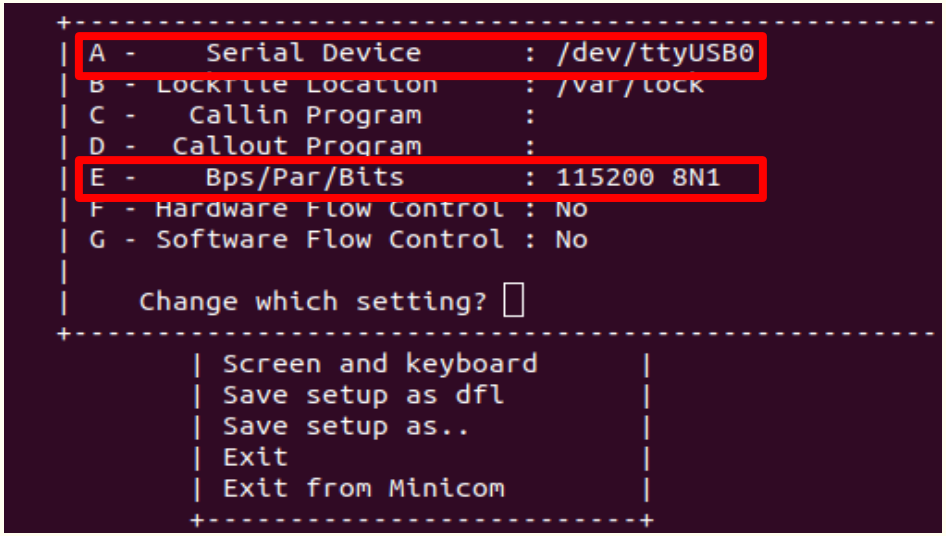
- Install “minicom” which is a serial terminal emulation application for two main purposes for HIMAX WE1 EVB Debug UART port.
  - $ sudo apt-get install minicom
  - Print application output.
  - Burn application to flash by using xmodem send application binary.
    - Minicom will extra install "lrzsz" package to support xmodem protocol

```
bigcat@bigcat-HP-Compaq-Pro-6300-MT:~$ sudo apt-get install minicom
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  lrzsz
The following NEW packages will be installed:
  lrzsz minicom
0 upgraded, 2 newly installed, 0 to remove and 216 not upgraded.
```

    - If you did not have “lrzsz” instruction, please install by following instruction.
      - $ sudo apt-get install lrzsz (to support xmodem protocol)

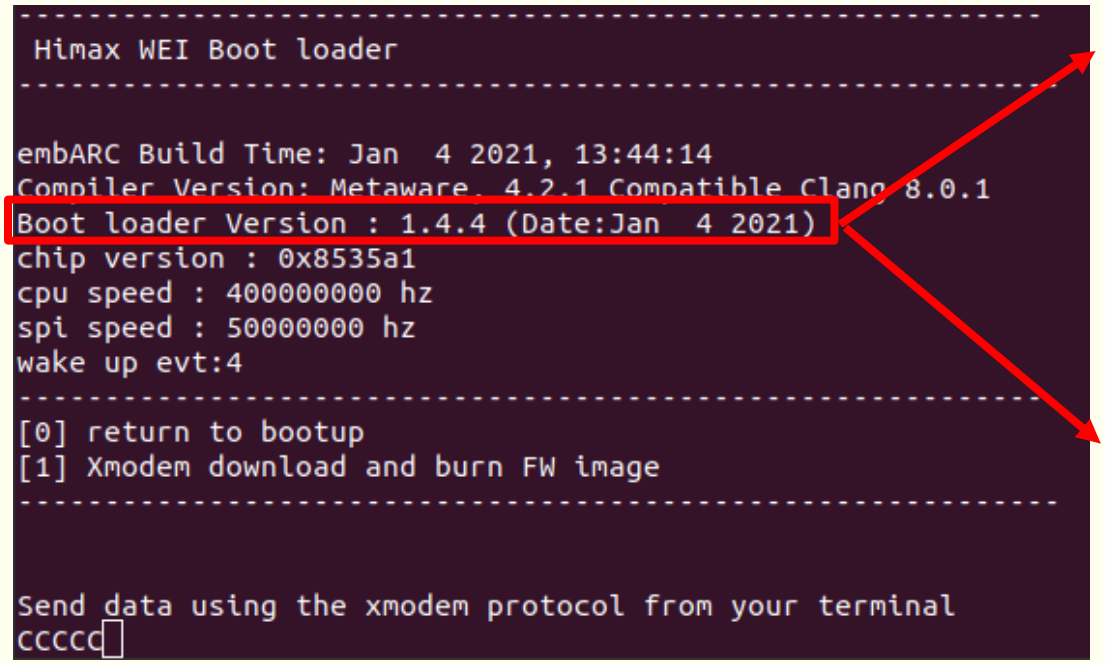
Deploy to WE-I

- Following steps are the way that how to update application in the flash.
  1) Connect HIMAX WE-I EVB with micro usb cable, check device ID by typing
     - $ ls /sys/bus/usb-serial/devices/ -ltrah
  2) Open “minicom” by typing
     - $ sudo minicom -s
  3) Select “Serial port Setup”, make sure “A - Serial Device” select correct device and “E - Bps/Par/Bits” set to correct value.



Deploy to WE-I

4) After finish all setting, press Esc button at configuration menu will return to minicom.
5) Reset WE-I EVB by press “reset” button, then press any keyboard key (except enter key) in 0.3 sec. boot option will be displayed.

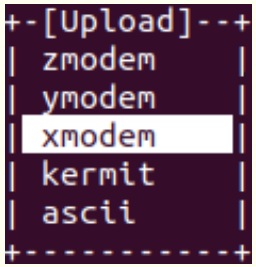


- Please check that your Boot loader version is 1.4.4 or not.
- If your version is lower than 1.4.4 will not support multiple images update.
- Please reference **Appendix** to update.

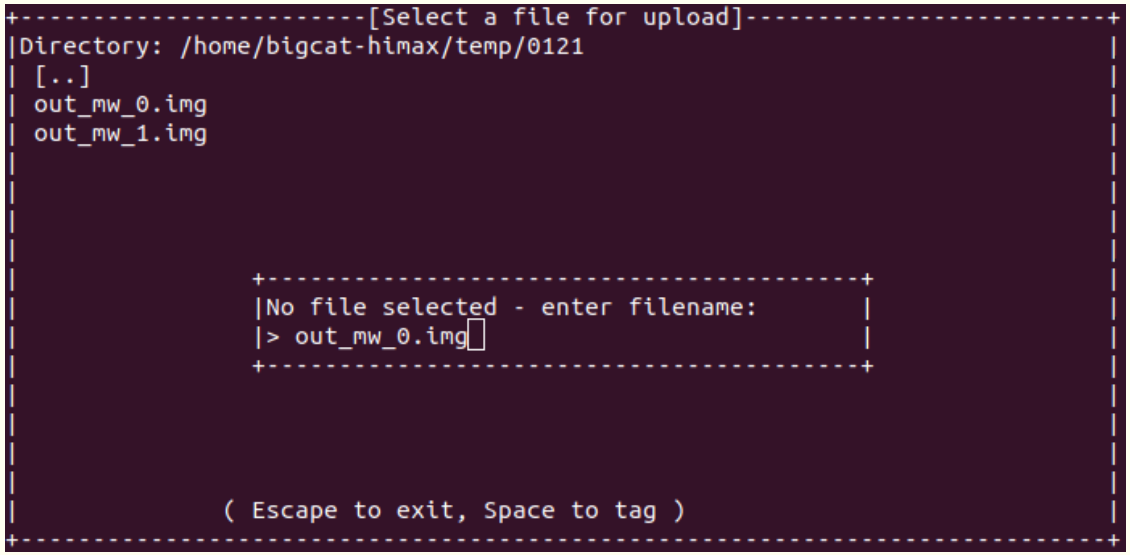
6) Press button 1 and WE1 EVB will enter receiving mode after then. Press Ctrl+A to enter minicom menu.

Deploy to WE-I

7) Press s button to upload file and select "xmodem".



8) Fill target flash image path and image name. Please start from file name with suffix "_0.img" and then "_1.img" if multiple images.



Deploy to WE-I

9) Press any key after transfer done.

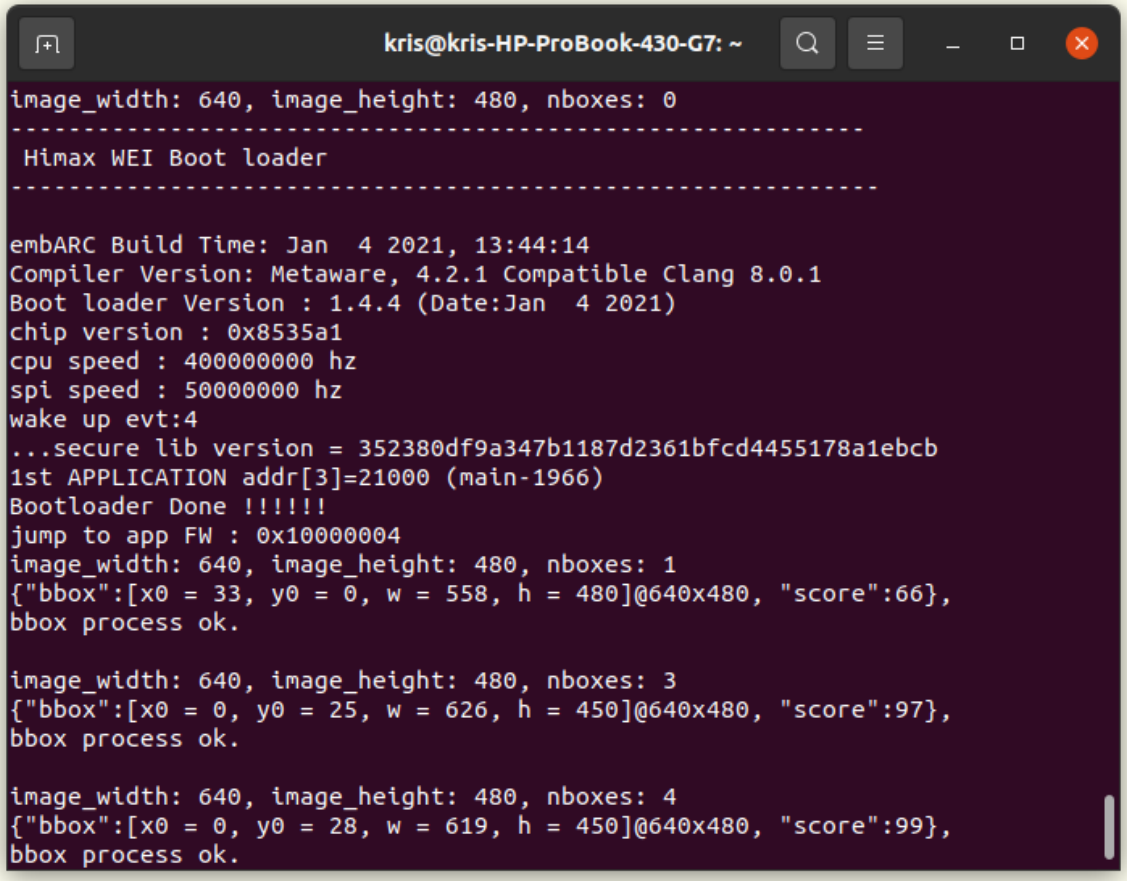
```
+-----------[xmodem upload - Press CTRL-C to quit]------------+
|Xmodem sectors/kbytes sent: 1164/145kRetry 0: NAK on sector  |
|Xmodem sectors/kbytes sent: 2506/313kRetry 0: NAK on sector  |
|Bytes Sent: 368640   BPS:7909                                |
|                                                             |
|Transfer complete                                           |
|                                                             |
| READY: press any key to continue...                        |
+-------------------------------------------------------------+
```

10) After "burn application done" message displayed on the console, back to step 6) and select another flash image or press reset button to restart.

```
Send data using the xmodem protocol from your terminal
CCCCCburn application done
```

Deploy to WE-I

- After above steps, update `yolo_coco*.img` or `yolo_himax*.img` to HIMAX WE1 EVB. After get data from sensor, we can display them on the console and see the images with bounding box on the PC TOOL.
  - You can see your detect results on the console .
    - Console (minicom)



Deploy to WE-I

PC Tool download link:
https://github.com/HimaxWiseEyePlus/WE_I_Plus_User_Examples/releases/download/v1.0/PC_TOOL

- Show yolo fastest result by PC TOOL
  - Download and install FT4222 Linux driver here
    - https://www.ftdichip.com/Support/SoftwareExamples/libft4222-linux-1.4.4.9.tgz
    - $tar xfvz libft4222-1.4.4.9.tgz
    - $sudo ./install4222.sh
  - Building FT4222 Linux driver (reference to ReadMe.txt)
    - $cd examples
    - $cc get-version.c -lft4222 -Wl,-rpath,/usr/local/lib
    - $sudo ./a.out
    - You should see a message similar to this:
      - Chip version: 42220100, LibFT4222 version: 010200E5
    - If you see a message such as "No devices connected" or "No FT4222H detected", this may indicate that:
      - There is no FT4222H connected. Check by running 'lsusb'.
      - Your program did not run with sufficient privileges to access USB. Use 'sudo', or 'su', or run as root.

Deploy to WE-I

- Show yolo fastest result by PC TOOL
  - Assign access right to usb device, go to deirectory
    - $cd /etc/udev/rules.d/
  - create file with naming 99-ftdi.rules, fill following data in file
    - $sudo apt install vim
    - $vim 99-ftdi.rules
    - # FTDI's ft4222 USB-I2C AdapterSUBSYSTEM=="usb", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="601c", GROUP="plugdev", MODE="0666"

Deploy to WE-I

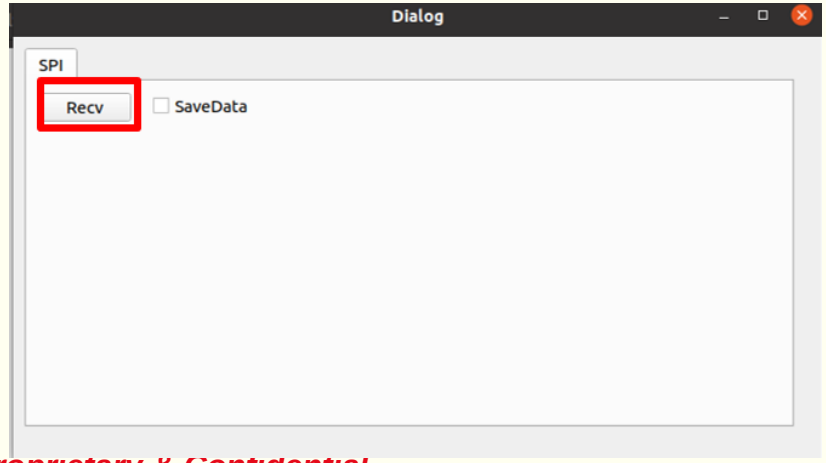
PC Tool download link:
https://github.com/HimaxWiseEyePlus/WE_I_Plus_User_Examples/releases/download/v1.0/PC_TOOL

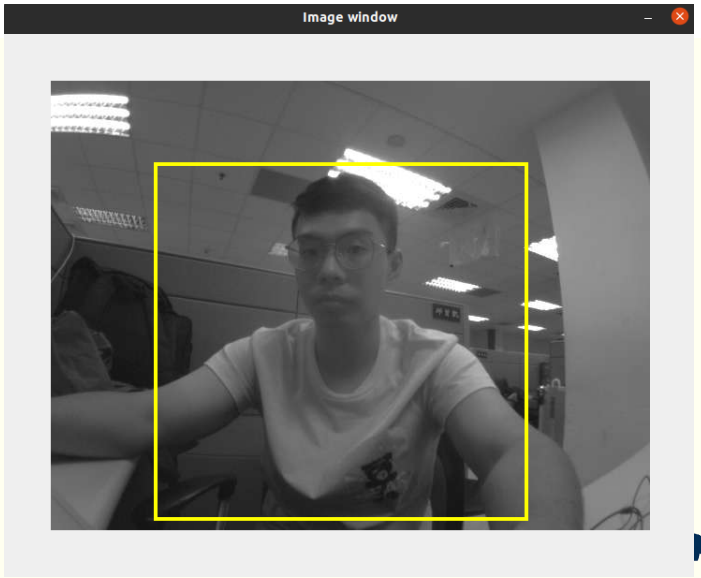
- Show yolo fastest result by PC TOOL
  - You can see your detect results with bounding box on the PC TOOL.
    - Before you click `PC_TOOL`, you can download here, please key-in following command on console.

```
chmod 777 PC_TOOL
```

    - Click `PC_TOOL` under your folder.

- Click Recv and you will get the result





Deploy to WE-I

Four examples download link:
https://github.com/HimaxWiseEyePlus/himax_tflm

- Deploy flow

Download model/SDK/third Party lib

↓

Select toolchain

↓

Select example and build

↓

Build flash

↓

flashing to Himax WE-I Plus

Deploy to Himax WE1 EVB

The example project for HIMAX WE1 EVB platform can be generated with following command:

Download related third party data and model setting (only need to download once)

```
make download
```

Default building toolchain in makefile is Metaware Development toolkit, if you are trying to build with GNU toolkit. please change the `ARC_TOOLCHAIN` define in `Makefile` like this

```
#ARC_TOOLCHAIN ?= mwdt
ARC_TOOLCHAIN ?= gnu
```

Build magic wand example and flash image, flash image name will be `magic_wand.img`

```
make magic_wand
make flash example=magic_wand
```

Build micro speech example and flash image, flash image name will be `micro_speech.img`

```
make micro_speech
make flash example=micro_speech
```

Build person detection INT8 example and flash image, flash image name will be `person_detection_int8.img`

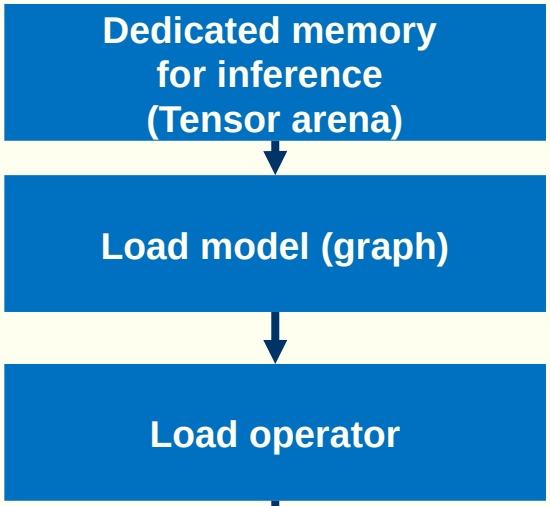
```
make person_detection_int8
make flash example=person_detection_int8
```

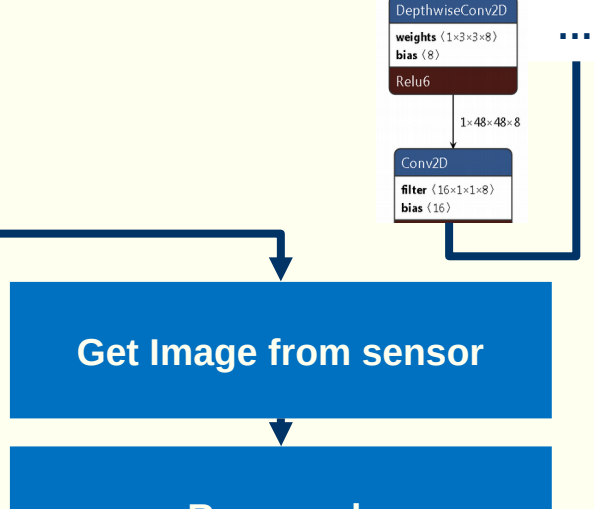
Build handwriting example and flash image, flash image name will be `handwriting.img`. please check here to know more about handwriting detail.

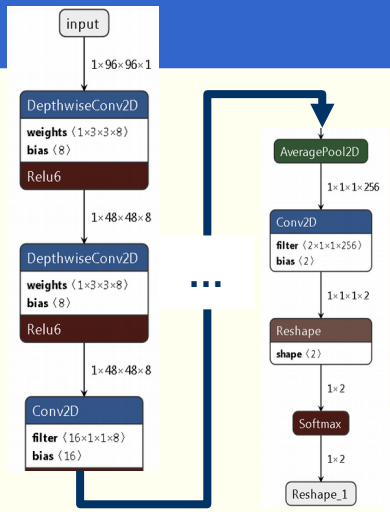
```
make handwriting
make flash example=handwriting
```

Inference on WE-I Plus EVB

- Flow to inference







Inference on WE-I Plus EVB

- Change to your model

```
constexpr int kTensorArenaSize = 136 * 1024;
static uint8_t tensor_arena[kTensorArenaSize];
```

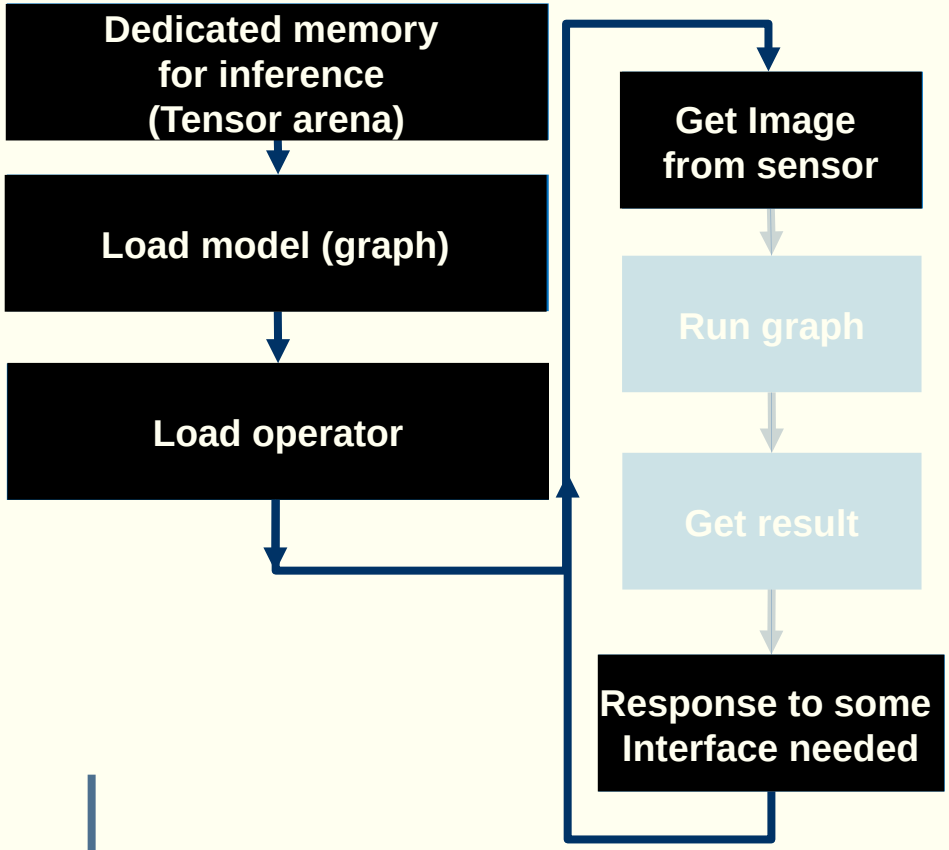
```
// Map the model into a usable data structure. This doesn't involve any
// copying or parsing, it's a very lightweight operation.
model = tflite::GetModel(g_person_detect_model_data);
```

```
// Keep model aligned to 8 bytes to guarantee aligned 64-bit accesses.
alignas(8) const unsigned char g_person_detect_model_data[] = {
    0x1c, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x0e, 0x00, 0x18, 0x00, 0x04, 0x00, 0x08, 0x00, 0x0c, 0x00, 0x10, 0x00,
    ...
    0x00, 0x0c, 0x00, 0x07, 0x00, 0x00, 0x00, 0x08, 0x00, 0x0a, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x01, 0x02, 0x00, 0x00, 0x00,
};
const int g_person_detect_model_data_len = 300568;
```

```
static tflite::MicroMutableOpResolver<5> micro_op_resolver;
micro_op_resolver.AddAveragePool2D();
micro_op_resolver.AddConv2D();
micro_op_resolver.AddDepthwiseConv2D();
micro_op_resolver.AddReshape();
micro_op_resolver.AddSoftmax();
```

```
hx_drv_sensor_capture(&g_pimg_config);

hx_drv_image_rescale((uint8_t*)g_pimg_config.raw_address,
                     g_pimg_config.img_width, g_pimg_config.img_height,
                     image_data, image_width, image_height);
```




```
// Process the inference results.
int8_t person_score = output->data.uint8[kPersonIndex];
int8_t no_person_score = output->data.uint8[kNotAPersonIndex];
```

Inference on WE-I Plus EVB

- Change to your model- all depend on your own model
  - Change tensor arena size
  - Change model
  - Change OP
  - Change post-processing

Inference on WE-I Plus EVB

- Change to your model
  - Convert .tflite to .cc file
    - $ xxd -i [your model].tflite [c file name].cc


```
kris@kris-HP-ProBook-430-G7: ~/Desktop/test
kris@kris-HP-ProBook-430-G7:~/Desktop/test$ xxd -i yolo_coco.tflite yolo_coco.cc
```

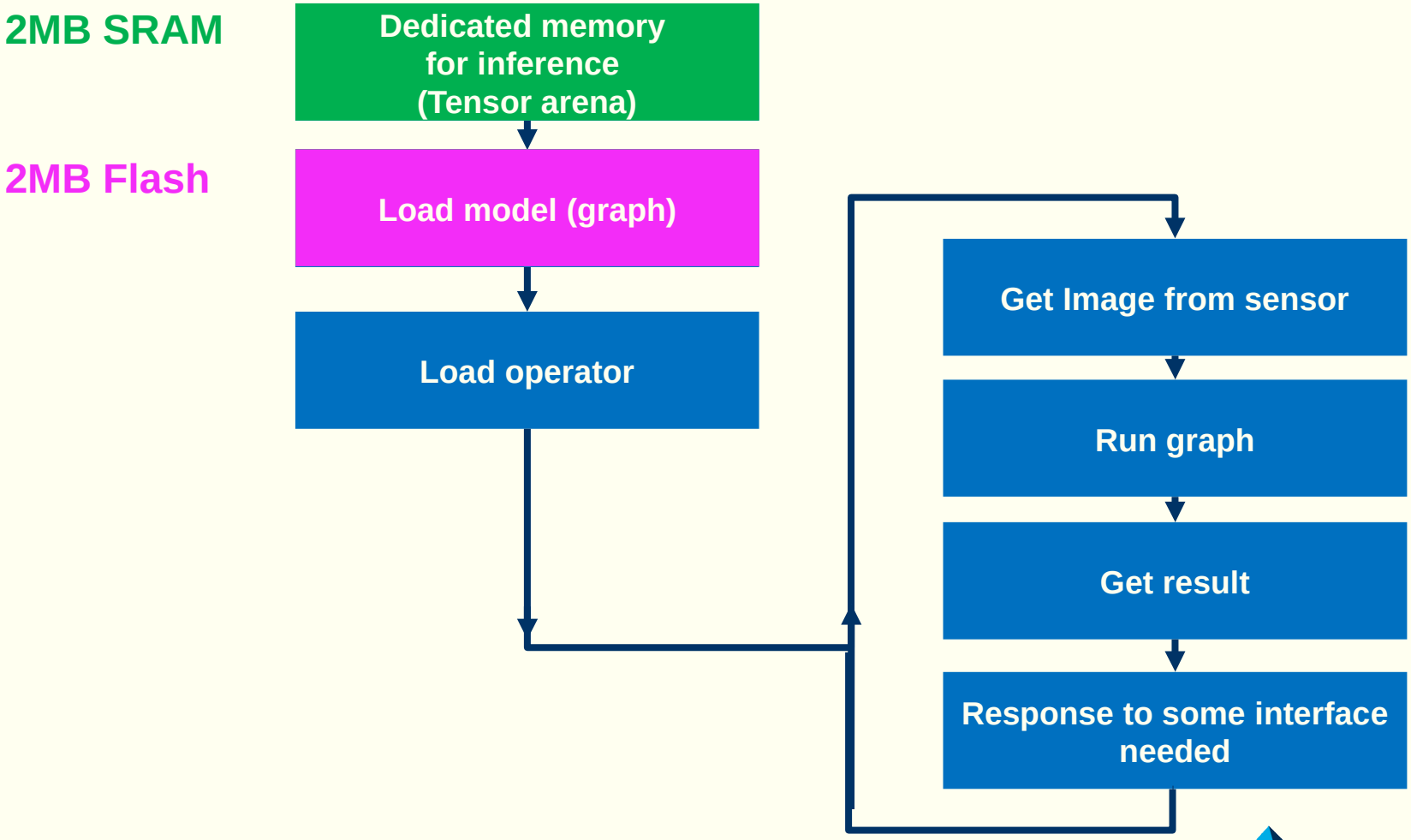
  - .cc file result

yolo_coco.
cc

```
unsigned char yolo_coco_tflite[] = {
  0x20, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x12, 0x00, 0x1c, 0x00, 0x18, 0x00, 0x14, 0x00, 0x10, 0x00,
  0x0c, 0x00, 0x08, 0x00, 0x00, 0x00, 0x04, 0x00, 0x12, 0x00, 0x00, 0x00,
...
  0x0c, 0x00, 0x10, 0x00, 0x0f, 0x00, 0x00, 0x00, 0x08, 0x00, 0x04, 0x00,
  0x0c, 0x00, 0x00, 0x00, 0x22, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x22
};
unsigned int yolo_coco_tflite_len = 570568;
```

Inference on WE-I Plus EVB

- Change to your model - Flow to inference



Inference on WE-I Plus EVB

- Change to your model
  - Model will be loaded in the SRAM during inference. Depend on your use case, model data can be pulled out of the SRAM if memory area is not enough for use.
  - We have the HIMAX SDK (v19+) about reading model from flash, detail device initialization will be done by hx_drv_flash_init() and hx_drv_flash_get_Model_address()
    - You should initial flash first. Then you can get the right model address.
    - You can simply call them to initial flash and get model addess to retrieve the model data from flash.

```
void setup() {
    hx_drv_uart_initial(UART_BR_115200);

    //flash init
    //must do before hx_drv_flash_get_Model_address()
    if (hx_drv_flash_init() != HX_DRV_LIB_PASS)
    {
        hx_drv_uart_print("Spi 0 for flash initial fail.");
        return ;
    }
```

```
    //get model (.tflite) from flash
    model = tflite::GetModel((unsigned char*)hx_drv_flash_get_Model_address());
    if (model->version() != TFLITE_SCHEMA_VERSION) {
        TF_LITE_REPORT_ERROR(error_reporter,
                             "Model provided is schema version %d not equal "
                             "to supported version %d.",
                             model->version(), TFLITE_SCHEMA_VERSION);
    return;
  }
```

Inference on WE-I Plus EVB

- Change to your model – image gen tool v2_1_11+
  - image_gen_gnu parameter (gnu)
    - -e: generated .elf file name (input)
    - -s: slice to 1024KB
    - -o: .img file name (output)
    - -t: .tflite file name (option: if you want to put your tflite file at flash)
    - Example:
      ./image_gen_gnu -e yolo.elf -t himax_dataset_yolo.tflite -s 1024 -o yolo_coco.img

Inference on WE-I Plus EVB

- Change to your model – Makefile explain:
  - If want to add your code should add code here.

```
164  yolo_SRCS := \
165  tensorflow/lite/micro/kernels/arc_mli/scratch_buffers.cc \
166  tensorflow/lite/micro/kernels/arc_mli/scratch_buf_mgr.cc \
167  tensorflow/lite/micro/kernels/arc_mli/mli_slicers.cc \
168  yolo_fastest/main.cc \
169  yolo_fastest/main_functions.cc
170
171
172  OBJS := \
173  $(patsubst %.cc,%.o,$(patsubst %.c,%.o,$(SRCS)))
174
175  yolo_OBJS:= \
176  $(patsubst %.cc,%.o,$(patsubst %.c,%.o,$(yolo_SRCS)))
```

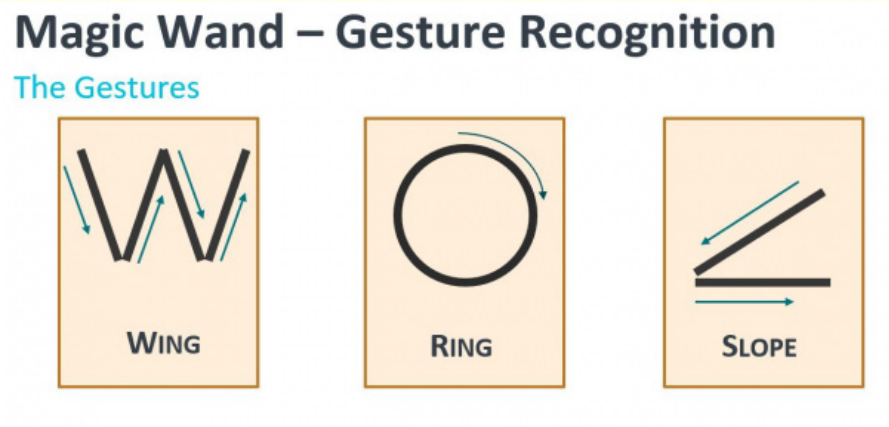
```
406  yolo_coco: MAP_NAME = yolo_coco
407  yolo_coco: yolo_coco.elf
408
409  yolo_himax: MAP_NAME = yolo_himax
410  yolo_himax: yolo_himax.elf
411
412  yolo_coco.elf : $(OBJS) $(yolo_OBJS)
413          $(CXX) $(CXXFLAGS) -o $@ $(OBJS) $(yolo_OBJS) $(LDFLAGS)
414
415  yolo_himax.elf : $(OBJS) $(yolo_OBJS)
416          $(CXX) $(CXXFLAGS) -o $@ $(OBJS) $(yolo_OBJS) $(LDFLAGS)
417
```

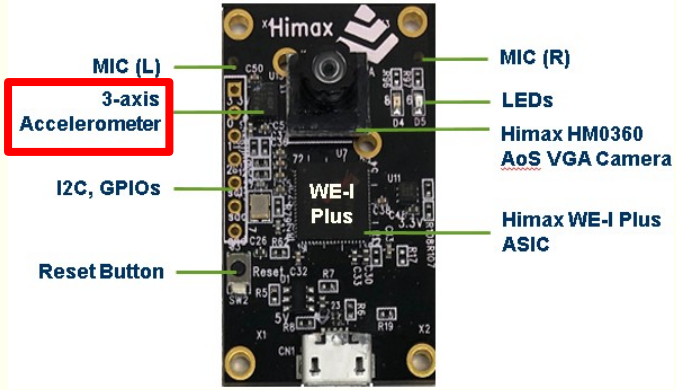
Inference on WE-I Plus EVB

- Change to your model – Makefile explain:
  - If don't want to put model on flash, you should change command here (gnu or metaware).

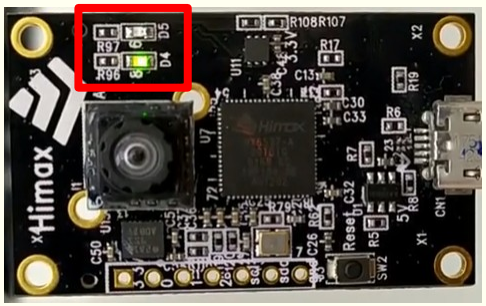
```
else ifeq ($(ARC_TOOLCHAIN), gnu)
flash:
ifdef example
ifeq ($(example), yolo_coco)
        @export PATH=$(shell pwd)/$(GEN_TOOL_DIR)/:$$PATH && \
        cp $(example).elf $(example).tflite $(GEN_TOOL_DIR) && \
        cd $(GEN_TOOL_DIR) && \
        $(GEN_TOOL_NAME) -e $(example).elf -t $(example).tflite -s 1024 -o $(example).img && \
        cp $(example)*.img .. && \
        rm $(example).elf $(example)*.img $(example).tflite
else ifeq ($(example), yolo_himax)
        @export PATH=$(shell pwd)/$(GEN_TOOL_DIR)/:$$PATH && \
        cp $(example).elf $(example).tflite $(GEN_TOOL_DIR) && \
        cd $(GEN_TOOL_DIR) && \
        $(GEN_TOOL_NAME) -e $(example).elf -t $(example).tflite -s 1024 -o $(example).img && \
        cp $(example)*.img .. && \
        rm $(example).elf $(example)*.img $(example).tflite
endif
```

Deploy to WE-I – magic wand

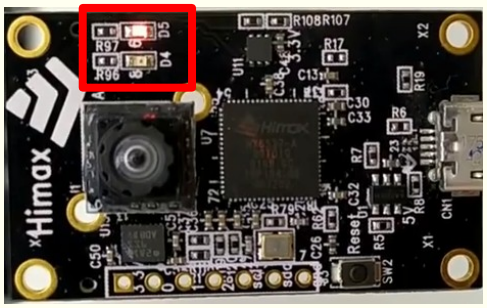




Deploy to WE-I – micro speech

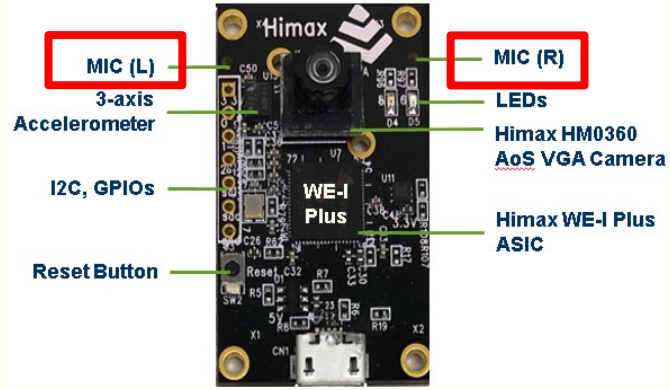


Read ‘Yes’ command



Read ‘No’ command

```
Heard yes (206) @1700ms
Heard no (201) @4800ms
Heard unknown (202) @8200m
s
Heard yes (205) @11600ms
```



SDK API

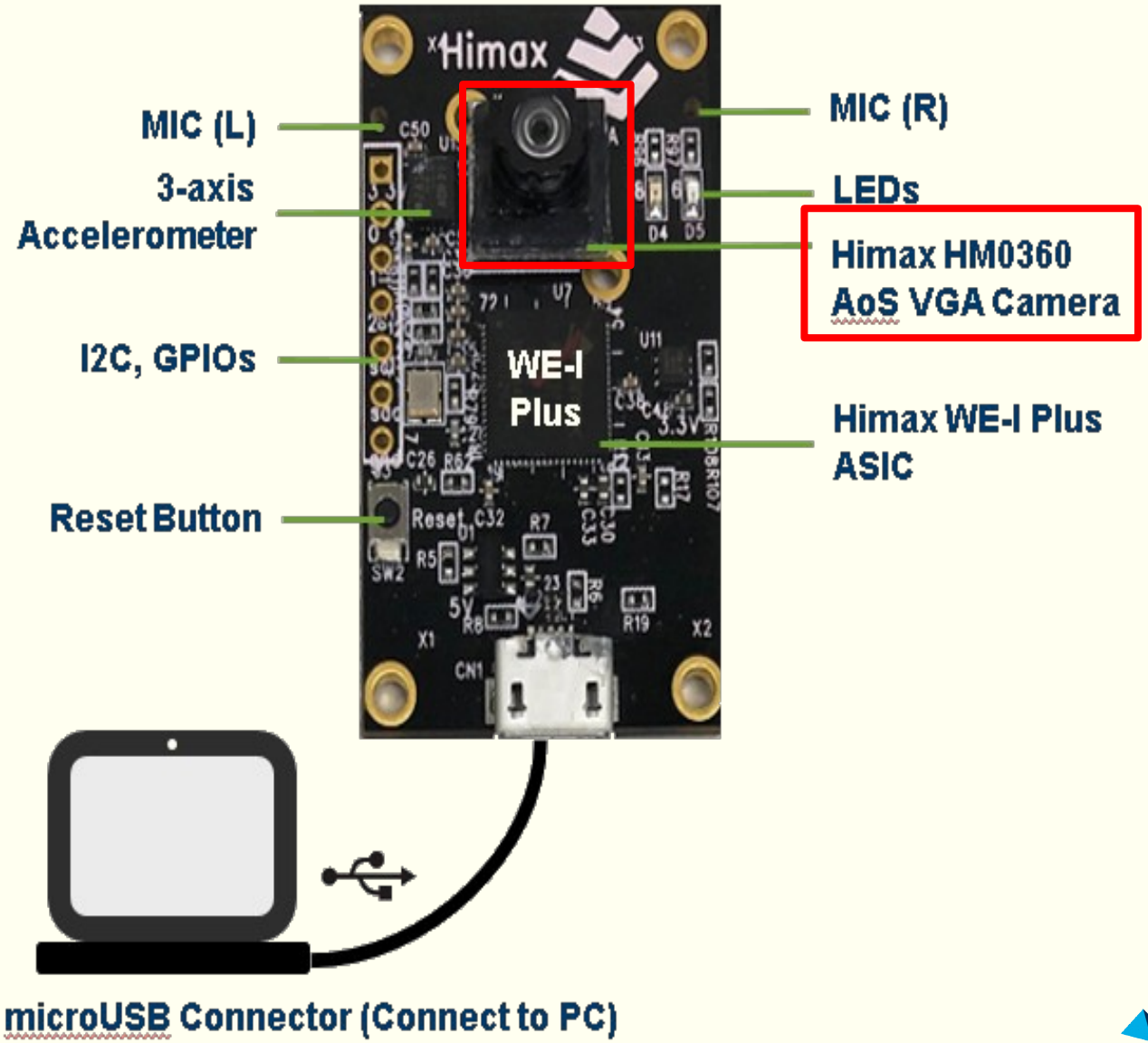
Example & SDK Download

- Download workshop example from Himax Github
  - $ git clone https://github.com/HimaxWiseEyePlus/bsp_tflu
- Go to the file directory of workshop example
  - $ cd bsp_tflu/HIMAX_WE1_EVB_example/scenario_app/workshop_example
- Download third party files setting (only need to download once)
  - $ make download
- SDK function will be under hx_drv_tflm.h.
  - workshop_example/himax_we1_sdk/hx_drv_tflm.h
- There will be three examples about
  - Capture image
  - Accelerometer
  - Microphone

Example & SDK Download

- How to build this three examples
  - Capture image:
    - $ make LAB1_capture_image
    - $ make flash example=LAB1_capture_image
  - Accelerometer
    - $ make LAB2_accelerometer
    - $ make flash example=LAB2_accelerometer
  - Microphone
    - $ make LAB3_microphone
    - $ make flash example=LAB3_microphone

SDK API – LAB1: Capture Images



SDK API – LAB1: Capture Images

- Camera initial
  - ```
    extern HX_DRV_ERROR_E hx_drv_sensor_initial(hx_drv_sensor_image_config_t *pimg_config);
    ```
  - Image sensor initialization, it try to initial sensor and query one JPEG frame + one RAW frame to target address.
  - Current image sensor use is HM0360, image resolution is 640x480.
  - Each pixel data is 8-bit.
- hx_drv_sensor_image_config_t* pimg_config's options are bellow
  - pimg_config.img_width/**< image width, assigned by driver */
  - pimg_config.img_height/**< image height, assigned by driver */
  - pimg_config.jpeg_address/**< JPEG image address, assigned by driver */
  - pimg_config.jpeg_size/**< JPEG image size, assigned by driver */
  - pimg_config.raw_address/**< RAW image address, assigned by driver */
  - pimg_config.raw_size/**< RAW image size, assigned by driver */
    ```

SDK API – LAB1: Capture Images

- Camera initial

```
hx_drv_sensor_image_config_t pimg_config;
  // initial uart
  hx_drv_uart_initial(UART_BR_115200);

  if (hx_drv_sensor_initial(&pimg_config) != HX_DRV_LIB_PASS)
  {
      hx_drv_uart_print("Sensor initial fail.");
      return 0;
  }

  if (hx_drv_spim_init() != HX_DRV_LIB_PASS)
  {
      hx_drv_uart_print("Spi initial fail.");
      return 0;
  }
```

SDK API – LAB1: Capture Images

- Capture images
  - ```
    extern HX_DRV_ERROR_E hx_drv_sensor_capture(hx_drv_sensor_image_config_t *pimg_config);
    ```
  - Query Image sensor and capture one JPEG frame and one RAW frame, sensor back to standby mode then.
  - Both RAW frame and JPEG frame will be provided to target address.
  - ```
    //capture image by sensor
    hx_drv_sensor_capture(&pimg_config);
    ```

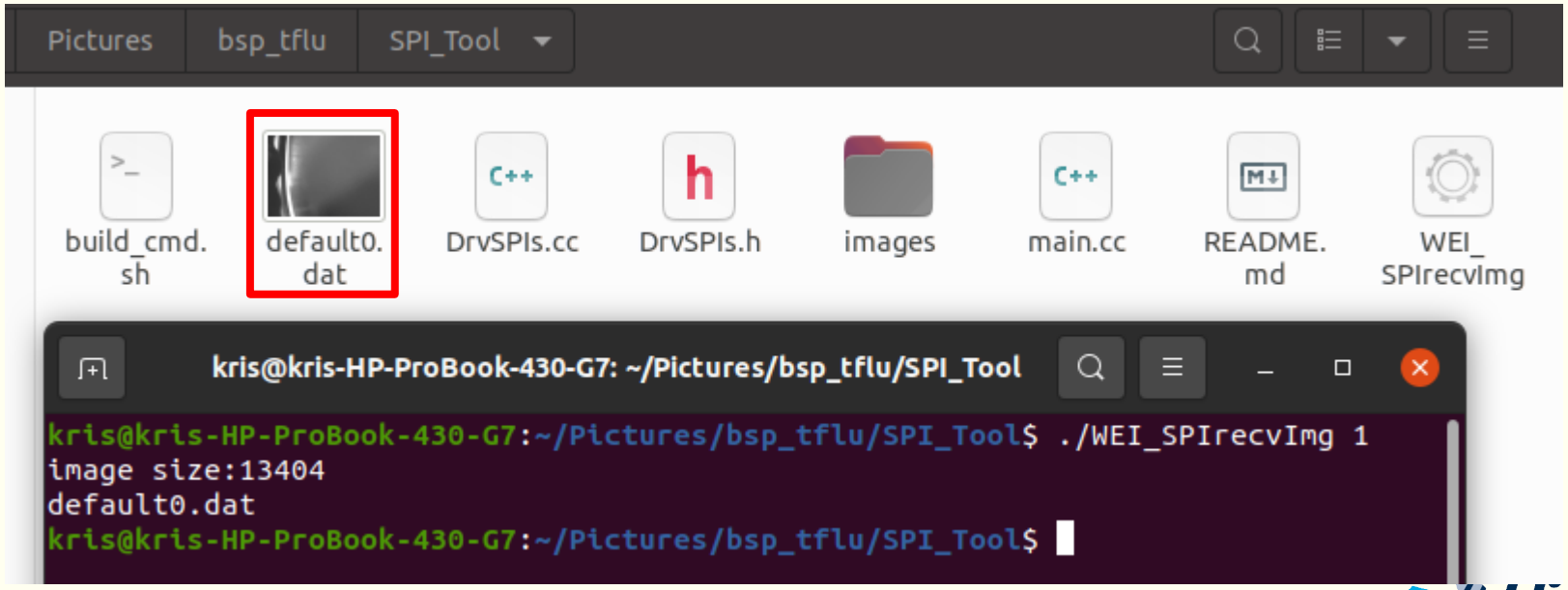
SDK API – LAB1: Capture Images

- Images send by SPI
  - `extern HX_DRV_ERROR_E hx_drv_spim_send(uint32_t addr, uint32_t size, HX_DRV_SPI_TYPE data_type);`
- You need to switch to needed output mode.
- You can send JPG or RAW by this API.
  - data_type = SPI_TYPE_JPG
  - data_type = SPI_TYPE_ RAW

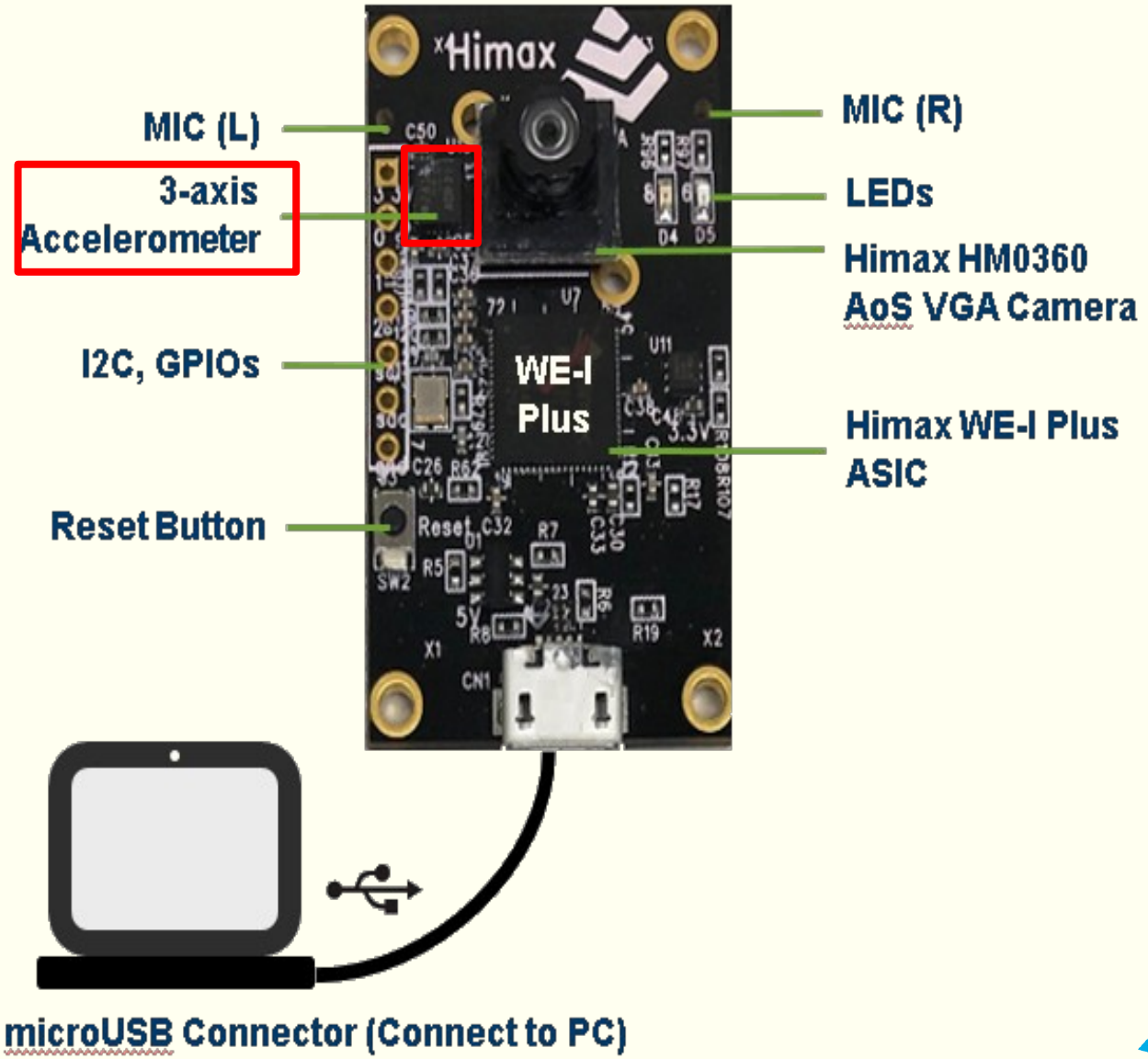
```
//Camera image send by SPI
if(hx_drv_share_switch(SHARE_MODE_SPIM) != HX_DRV_LIB_PASS)
{
    hx_drv_uart_print("Switch to SPI fail.");
    return HX_DRV_LIB_ERROR;
}
if(hx_drv_spim_send(pimg_config.jpeg_address , pimg_config.jpeg_size , SPI_TYPE_JPG) != HX_DRV_LIB_PASS)
{
    hx_drv_uart_print("Image capture failed.");
    return HX_DRV_LIB_ERROR;
}
```

SDK API – LAB1: Capture Images

- Install FT4222 driver:
  - Reference to Deploy to WE-I
- Open terminal and key in the command
  - $ git clone https://github.com/HimaxWiseEyePlus/bsp_tflu
  - Go to the file folder bsp_tflu/SPI_Tool
  - $ ./WEI_SPIrecvImg [num]
  - The [num] is how many images you want to save.



SDK API – LAB2: Accelerometer



SDK API – LAB2: Accelerometer

- Accelerometer initial
  - `extern HX_DRV_ERROR_E hx_drv_accelerometer_initial();`
  - 3-axis accelerometer initialization, it start to retrieve data after initial.
- Receive accelerometer data
  - `extern HX_DRV_ERROR_E hx_drv_accelerometer_receive(float *x, float *y, float *z);`
  - Receive data from 3-axis accelerometer.
- Accelerometer FIFO count get
  - `extern uint8_t hx_drv_accelerometer_available_count();`
  - Check how many data in the accelerometer FIFO.
  - Each count represent 1 set of x-axis, y-axis, z-axis data.

SDK API – LAB2: Accelerometer

- Send data by UART
  - UART should be initial.

```
hx_drv_uart_initial(UART_BR_115200);
```

```
if (hx_drv_accelerometer_initial() != HX_DRV_LIB_PASS)
    hx_drv_uart_print("Accelerometer Initialize Fail\n");
else
    hx_drv_uart_print("Accelerometer Initialize Success\n");
while (1)
{
    int available_count = 0;
    float x, y, z;
    available_count = hx_drv_accelerometer_available_count();
    hx_drv_uart_print("Accel get FIFO: %d\n", available_count);
    for (int i = 0; i < available_count; i++)
    {
        hx_drv_accelerometer_receive(&x, &y, &z);
    }
    uint8_t x_sign = '+';
    int32_t x_int_buf = x * scale_size;
    uint32_t pos_x = 0;
    uint32_t flo_x = 0;
    if(x_int_buf < 0)
    {
        x_sign = '-';
        x_int_buf = x_int_buf * (-1);
    }
    pos_x = x_int_buf / scale_size;
    flo_x = x_int_buf % scale_size;
    uint8_t x_char[11];
    sprintf(x_char, "x= %c%1d.%1d ", x_sign,pos_x,flo_x);
    hx_drv_uart_print(x_char);
}
```

```
Accel get FIFO: 13
x= +0.1 y= +0.5 z= -0.8

Accel get FIFO: 14
x= +0.0 y= +0.5 z= -0.8

Accel get FIFO: 13
x= +0.0 y= +0.5 z= -0.8

Accel get FIFO: 13
x= +0.1 y= +0.5 z= -0.8

Accel get FIFO: 14
x= +0.0 y= +0.5 z= -0.9

Accel get FIFO: 13
x= +0.2 y= +0.5 z= -0.8

Accel get FIFO: 13
x= +0.1 y= +0.5 z= -0.7

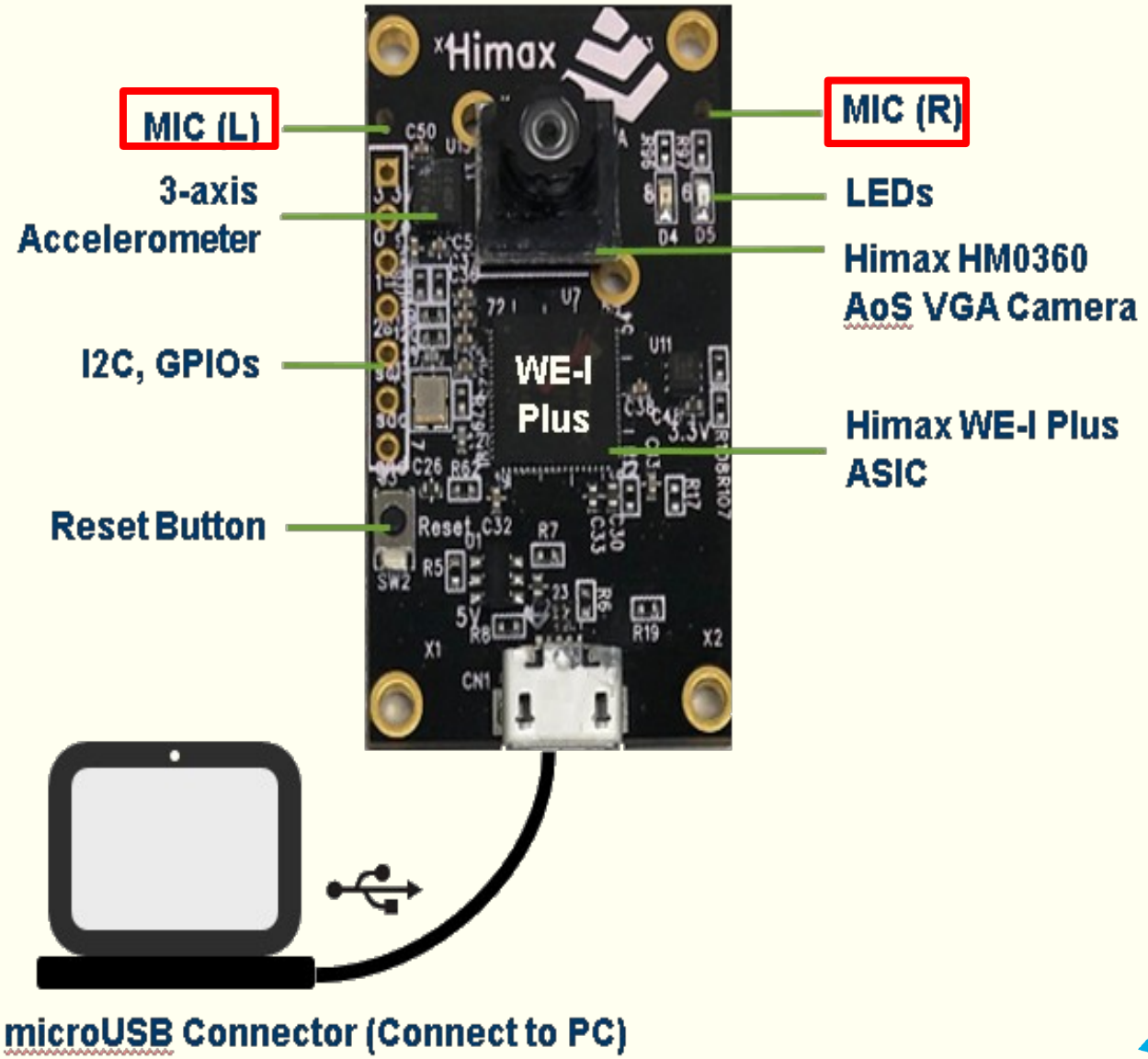
Accel get FIFO: 13
x= +0.0 y= +0.4 z= -0.8

Accel get FIFO: 14
x= +0.0 y= +0.4 z= -0.8

Accel get FIFO: 13
x= +0.0 y= +0.4 z= -0.8

Accel get FIFO: 13
x= +0.0 y= +0.4 z= -0.9
```

SDK API – LAB3: Microphone



SDK API – LAB3: Microphone

- Microphone initial
  - `extern HX_DRV_ERROR_E hx_drv_mic_initial();`
  - Microphone initialization, it will initial microphone setting.
  - Please use hx_drv_mic_on() to start record audio after initial.
  - Receive sampling rate is 16KHz and data format is PDM.
- Microphone enable
  - `extern HX_DRV_ERROR_E hx_drv_mic_on();`
  - Turn on microphone.
- Microphone disable
  - `extern HX_DRV_ERROR_E hx_drv_mic_off();`
  - It will stop receive audio data and time-stamp reset back to zero.
- Get current time-stamp from audio buffer in driver.
  - `extern HX_DRV_ERROR_E hx_drv_mic_timestamp_get(int32_t *time);`
  - For current Himax mic driver, time stamp will updated every 100ms.

SDK API – LAB3: Microphone

- Capture Dual channel audio data from Microphone
  - `extern HX_DRV_ERROR_E hx_drv_mic_capture_dual(hx_drv_mic_data_config_t *pmic_config);`
  - Each sample for dual PDM are 32 bits, includes Left channel 16bits little-endian signed data and right channel 16bits little-endian signed data.
  - During each millisecond, there will be 16 samples (64 bytes) of audio data storage to target address.
    - For example, if data_size is 6400 bytes, that means 1600 samples (100ms) of audio data in address.
  - This API often called when get changes by API (hx_drv_mic_timestamp_get)
  - This API will retrieve latest 100ms audio data from microphone if return HX_DRV_LIB_PASS.
  - Once the API is done, you can get data from target address and wait next time stamp changes (which means wait 100ms).

SDK API – LAB3: Microphone

- Example for capturing dual channel audio data

```
#include "hx_drv_tflm.h"
#include "stdio.h"
#include "string.h"
#define mic_sample_rate 16000
#define AUD_BLK_100MS_SZ (mic_sample_rate / 1000 * 100) //100ms
#define aud_stamp_cnt 30    //Record time: 30 * 100ms

typedef struct {
    int16_t left;
    int16_t right;
} META_AUDIO_t;

hx_drv_mic_data_config_t slt_audio_config;

META_AUDIO_t audio_clip[AUD_BLK_100MS_SZ * aud_stamp_cnt];
```

SDK API – LAB3: Microphone

- Example for capturing dual channel audio data (continue)

```
int main(int argc, char* argv[])
{
    int32_t time_prev = 0, time_cur = 0;
    uint8_t key_data;

    hx_drv_uart_initial(UART_BR_115200);

    if(hx_drv_mic_initial() != HX_DRV_LIB_PASS)
        hx_drv_uart_print("Microphone Initialize Fail\n");
    else
        hx_drv_uart_print("Microphone Initialize Success\n");

    if(hx_drv_mic_on() != HX_DRV_LIB_PASS)
        hx_drv_uart_print("Microphone Enable Fail\n");
    else
        hx_drv_uart_print("Microphone Enable Success\n");

    if(hx_drv_mic_timestamp_get(&time_prev) == HX_DRV_LIB_PASS)
    {
        time_cur = time_prev;
    }
    else
        hx_drv_uart_print("Microphone Timestamp Error\n");

    hx_drv_uart_print("Wait for user press key: [R] \n");
```

SDK API – LAB3: Microphone

- Example for capturing dual channel audio data (continue)

```
while(time_cur == time_prev)
{
    //  Wait for time_stamp change
    //  It changes every 100ms
    hx_drv_mic_timestamp_get(&time_cur);
}
time_prev = time_cur;

if(hx_drv_mic_capture_dual(&slt_audio_config) == HX_DRV_LIB_PASS)
    memcpy((void*)&audio_clip[record_cnt * AUD_BLK_100MS_SZ], (void*)slt_audio_config.data_address, slt_audio_config.data_size*sizeof(uint8_t));
```

```
while (1)
{
    hx_drv_uart_getchar(&key_data);
    if(key_data == 'R')
    {
        hx_drv_uart_print("Start Record Audio\n");
        for(int record_cnt = 0; record_cnt < aud_stamp_cnt; record_cnt ++)...

        hx_drv_uart_print("Microphone Get Data Success\n");

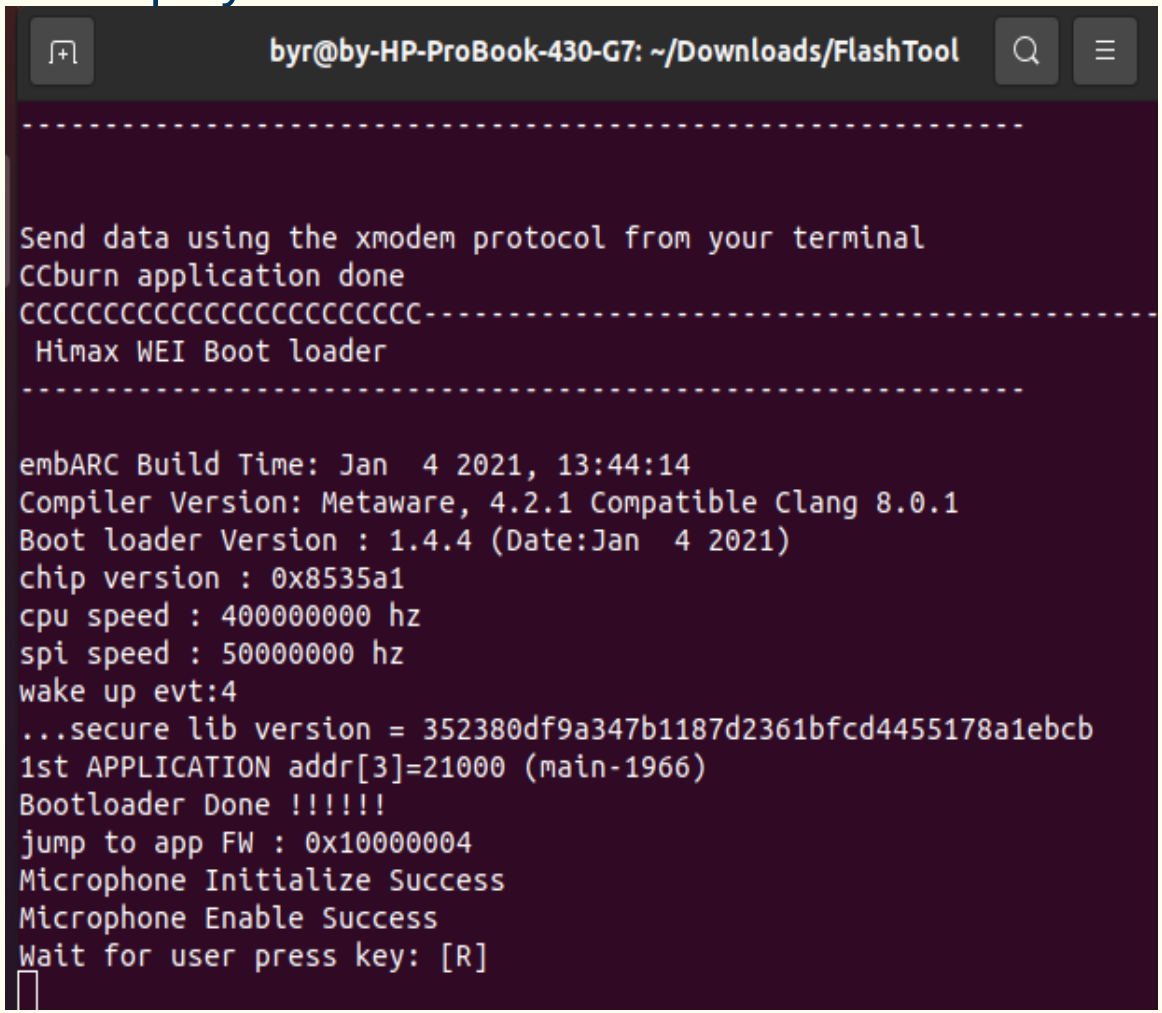
        hx_drv_uart_print("Start to send\n");

        for(int i = 0; i < (AUD_BLK_100MS_SZ * aud_stamp_cnt); i ++)
        {
            hx_drv_uart_print("%5d, %7d, %7d\n", i, audio_clip[i].left,  audio_clip[i].right);
        }
        hx_drv_uart_print("End of send\n");
    }

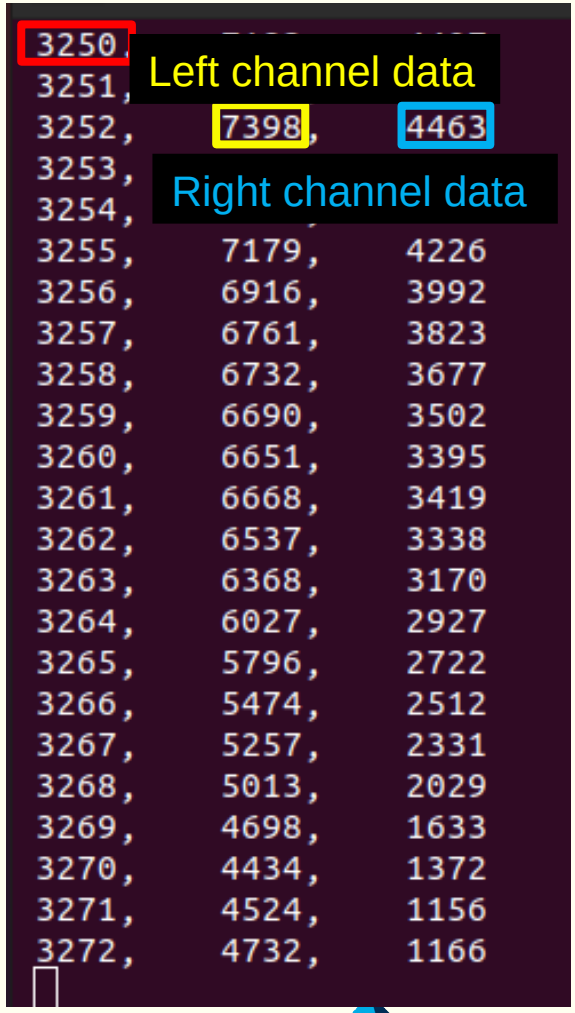
    time_prev = time_cur;
    key_data = '\0';
}
```

SDK API – LAB3: Microphone

- Deploy to WE-I and show at terminal



Data serial number



Demo

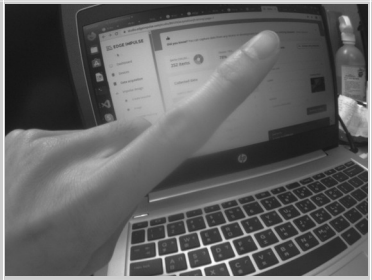
Q&A

Drive for better vision

Hand-on example

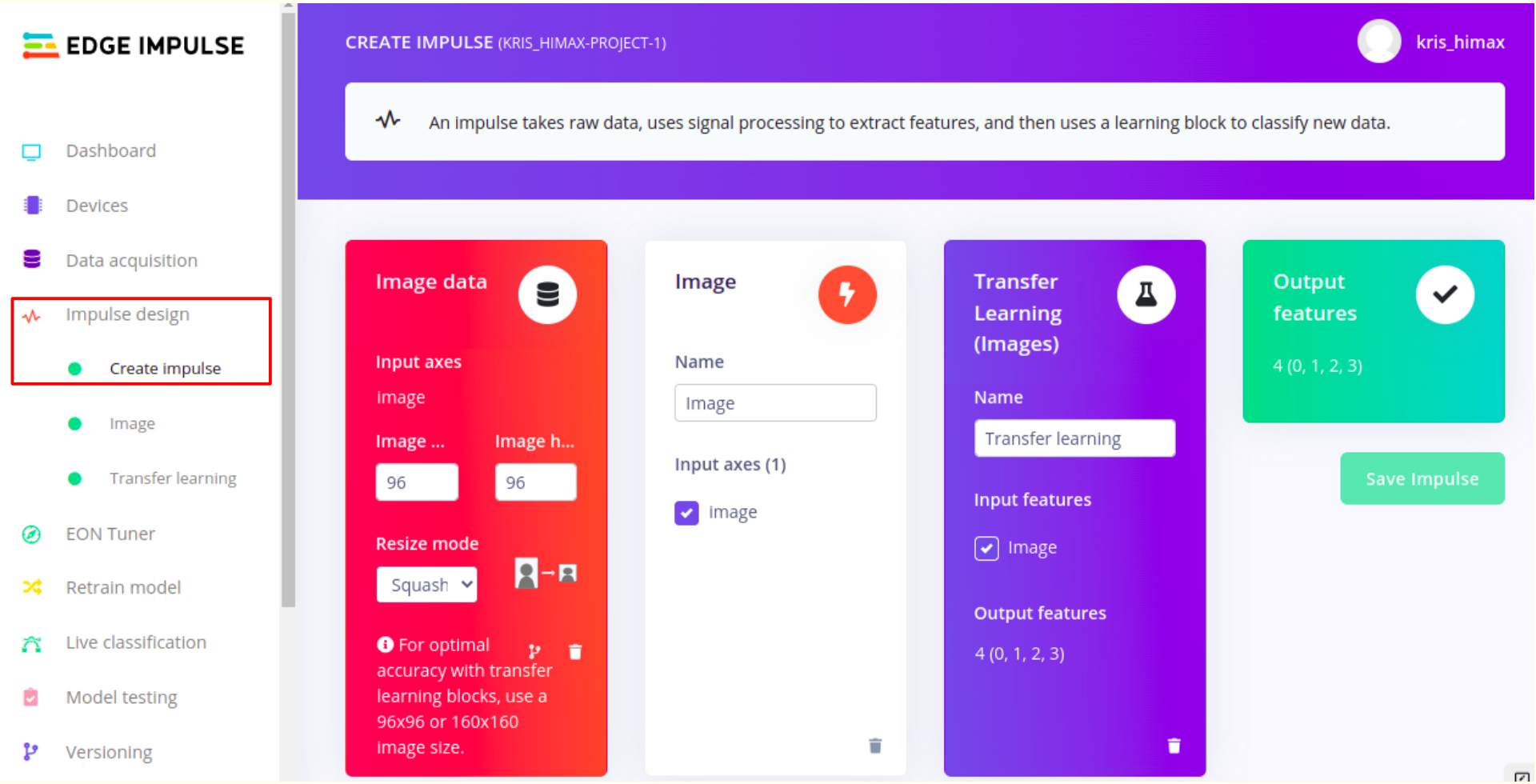
Hand-on example

- Use Edge Impulse to train the Hand Gesture Recognition model and do inference
  - Input Image data:
    - Image width = 96
    - Image height = 96
  - Output features:
    - 4 (0,1,2,3)

label	0	1	2	3
image	unkown			

Hand-on example

- Please login Edge impulse and create your model



Hand-on example

https://docs.edgeimpulse.com/docs/himax-we-i-plus#all-licenses-are-in-use-by-other-developers

- How to use HIMAX WE-I EVB to do data acquisition
  - Installing dependencies
    - Edge Impulse CLI
  - Connecting to Edge Impulse

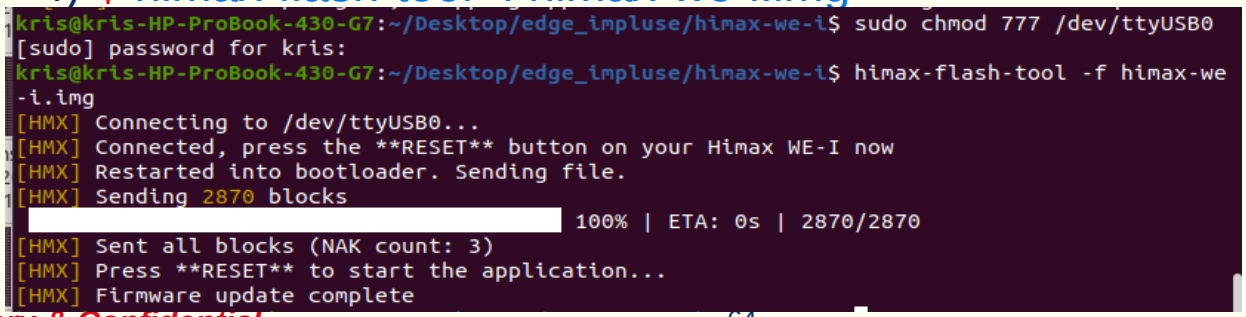
    1. Connect the development board to your computer
    2. Update the firmware
       - The development board does not come with the right firmware yet. To update the firmware:

         1) Download the latest Edge Impulse firmware, and unzip the file.

         2) $ ls /sys/bus/usb-serial/devices/ -ltra

         3) $ sudo chmod 777 /dev/ttyUSB0

         4) $ himax-flash-tool -f himax-we-i.img



Hand-on example

https://docs.edgeimpulse.com/docs/himax-we-i-plus#all-licenses-are-in-use-by-other-developers

- How to use HIMAX WE-I EVB to do data acquisition
  - Connecting to Edge Impulse

    3. Setting keys
      - From a command prompt or terminal, run:
        - $ edge-impulse-daemon

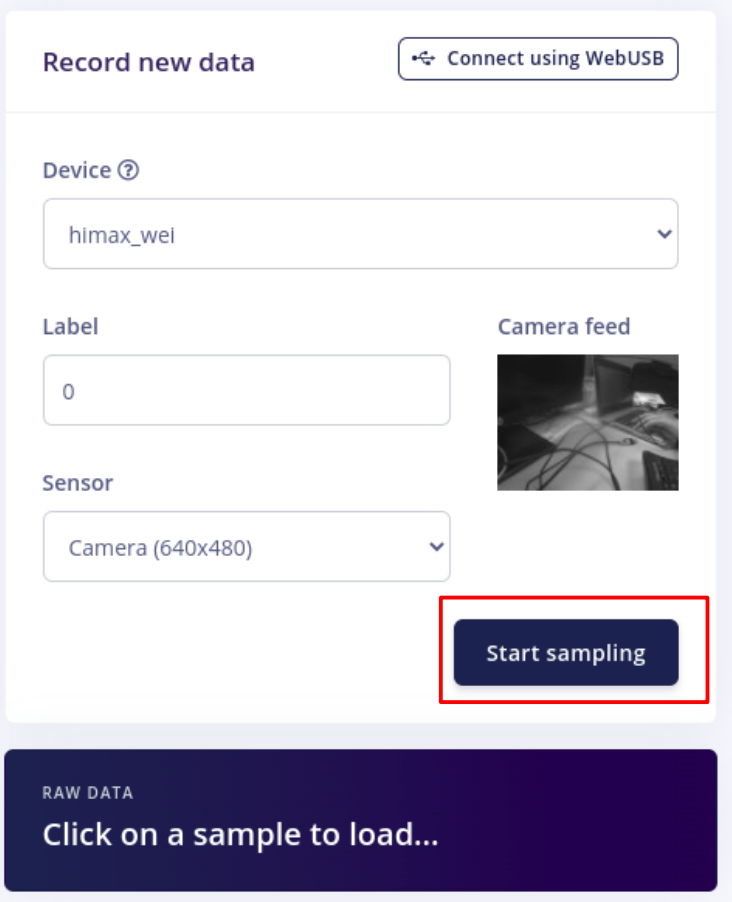
```
kris@kris-HP-ProBook-430-G7:~/Desktop/edge_impluse/himax-we-i$ edge-impulse-daemon
Edge Impulse serial daemon v1.14.7
Endpoints:
    Websocket: wss://remote-mgmt.edgeimpulse.com
    API:       https://studio.edgeimpulse.com/v1
    Ingestion: https://ingestion.edgeimpulse.com

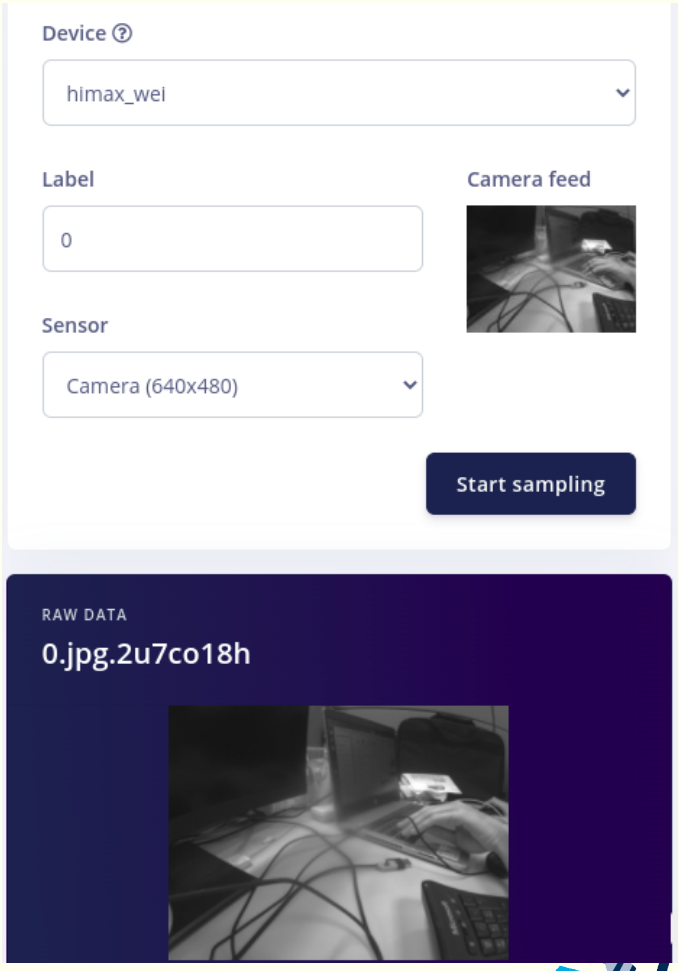
[SER] Connecting to /dev/ttyUSB0
[SER] Serial is connected, trying to read config...
[SER] Retrieved configuration
[SER] Device is running AT command version 1.6.0

Setting device ID... OK
Setting upload host in device... OK
Configuring remote management settings... OK
Configuring API key in device... OK
Configuring HMAC key in device... OK
[SER] Device is not connected to remote management API, will use daemon
[WS ] Connecting to wss://remote-mgmt.edgeimpulse.com
[WS ] Connected to wss://remote-mgmt.edgeimpulse.com
[WS ] Device "himax_wei" is now connected to project "kris_himax-project-1"
[WS ] Go to https://studio.edgeimpulse.com/studio/80339/acquisition/training to build your
machine learning model!
```

Hand-on example

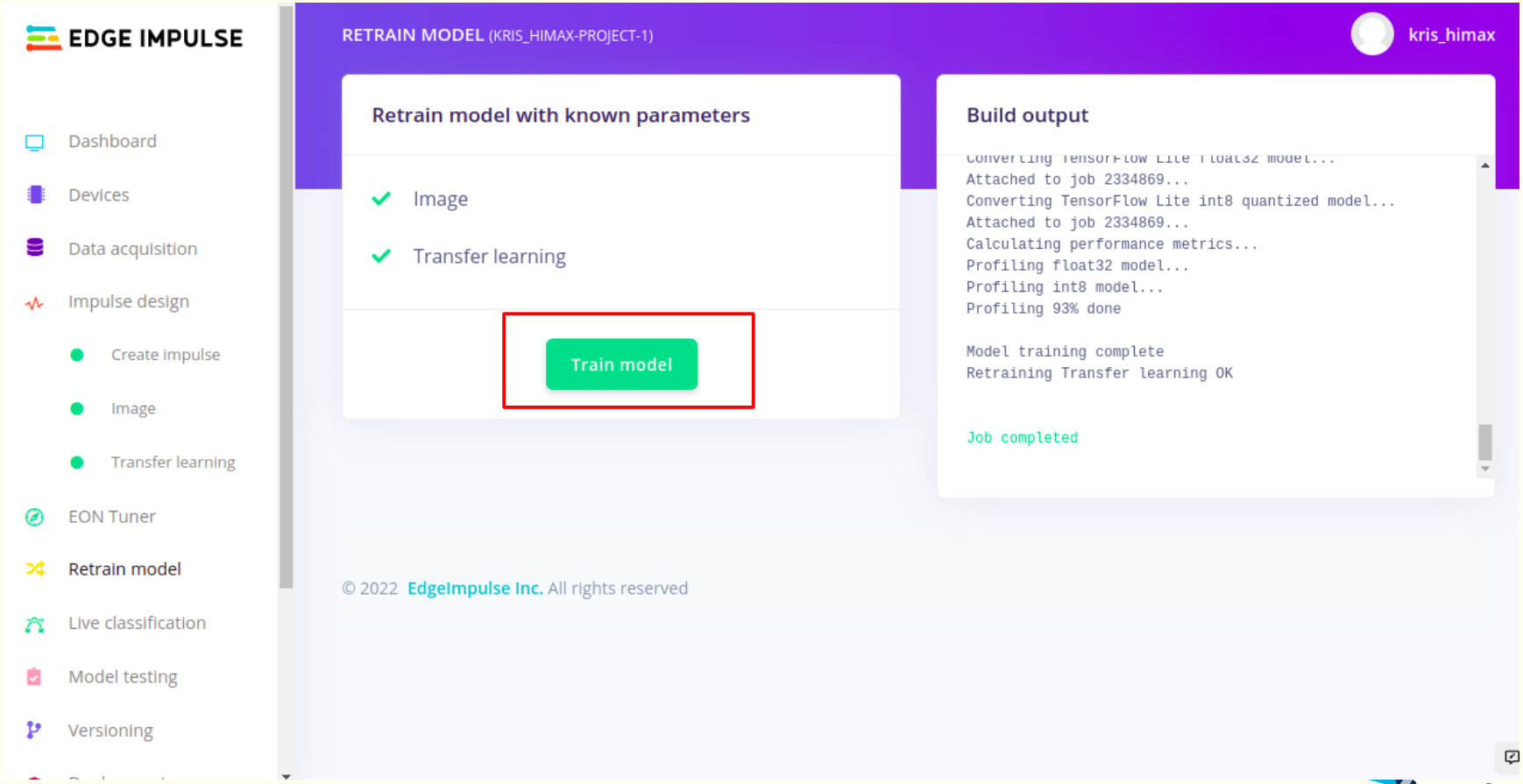
- While you connect to WE-I, please go to Edge Impulse website and start to data acquisition
- Set Label feature (0,1,2,3) and press Start sampling





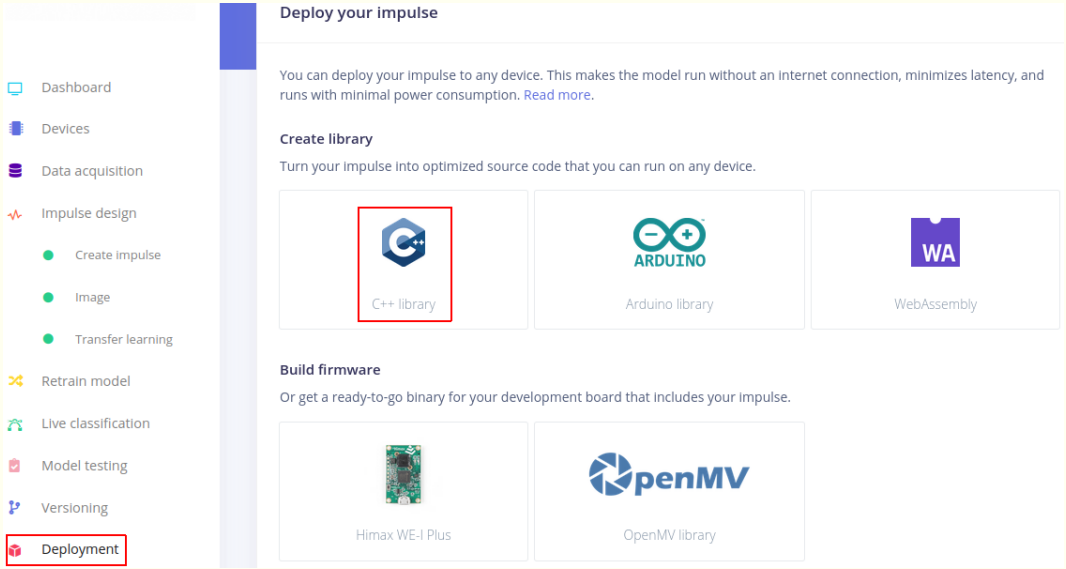
Hand-on example

- Train model
  - Go to Retrain model page to press train model button.
  - Wait until Job completed.
  - Your own model will be ready.



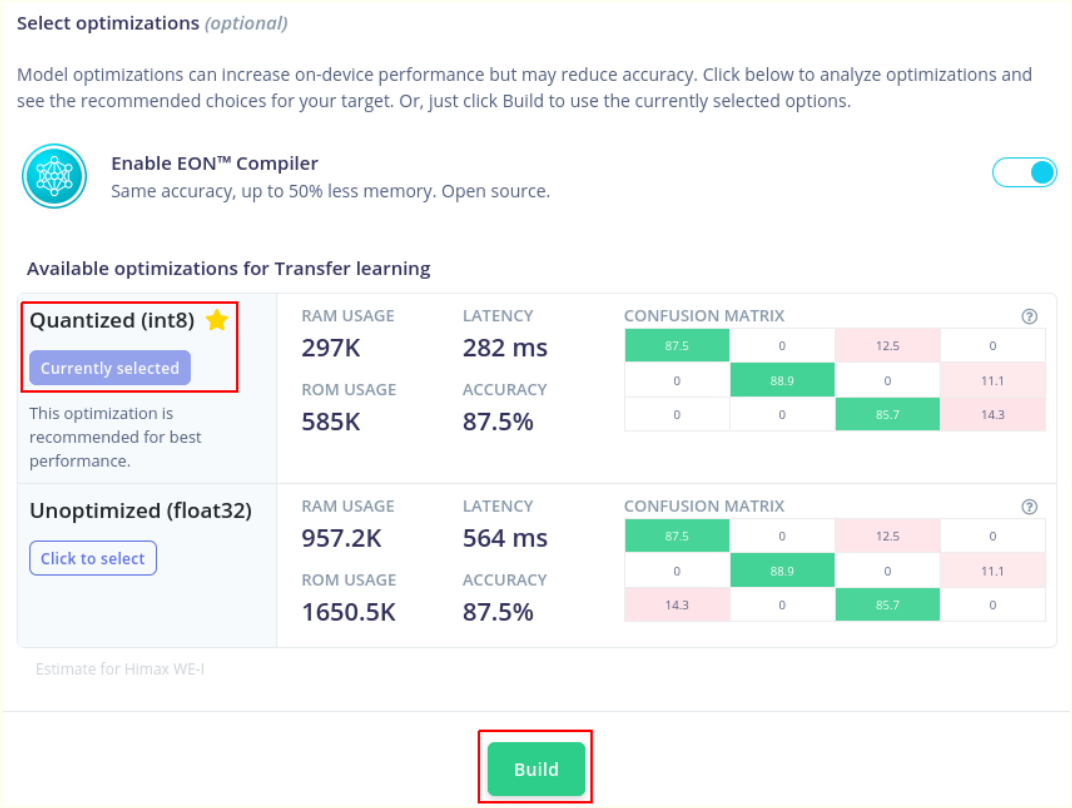
Hand-on example

- Please download Himax GitHub edgeimpulse-example and add your own model to build the image file
  - https://github.com/HimaxWiseEyePlus/bsp_tflu/tree/master/HIMAX_WE1_EVB_example/scenario_app/edgeimpulse-example
  - Download related third party data (only need to download once)
    - $ make -C ../../ download
  - Export C++ library to repository
    - Head over to your Edge Impulse project, go to Deployment page, select C++ library



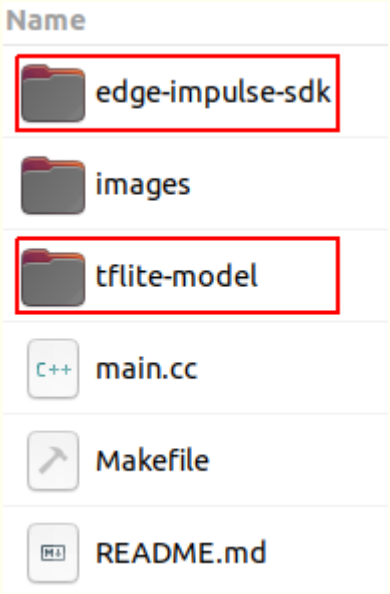
Hand-on example

- Please download Himax GitHub edgeimpulse-example and add your own model to build the image file
  - Export C++ library to repository
    - select Quantized (int8) and click Build to download the .zip file of your project.



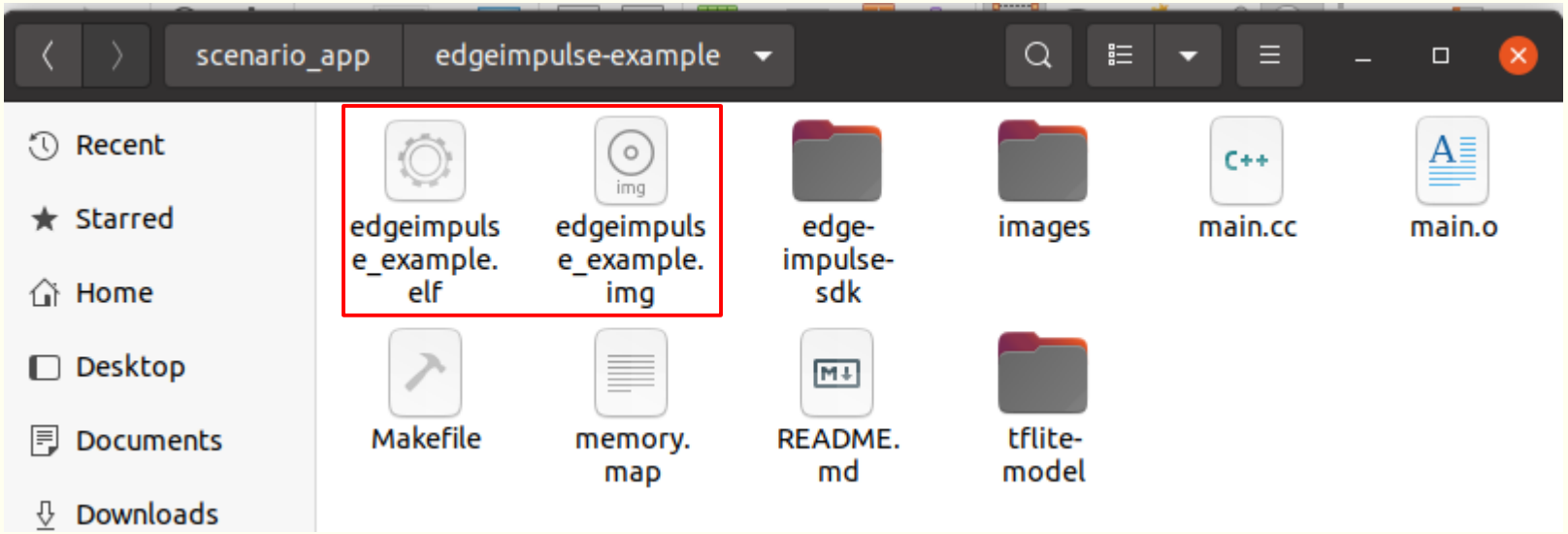
Hand-on example

- Please download Himax GitHub edgeimpulse-example and add your own model to build the image file
  - Export C++ library to repository
    - Extract the .zip file and copy the edge-impulse-sdk and tflite-model folders to this example folder.



Hand-on example

- Build example and flash image
  - Build Edge Impulse example and flash image. Flash image name will be edgeimpulse_example.img.
    - $ make all
    - $ make flash



Hand-on example

- Deploy to HIMAX WE1 EVB
  - After above steps, update edgeimpulse_example.img to HIMAX WE1 EVB
  - In our example, we display the output score of each category on the console.

```
[0] -84, [1] -64 [2] 19 [3] -127
[0] 0, [1] -111 [2] -22 [3] -123
[0] -47, [1] -115 [2] 27 [3] -120
[0] -63, [1] -74 [2] 8 [3] -127
[0] -42, [1] -87 [2] -3 [3] -125
[0] -73, [1] -102 [2] 40 [3] -121
[0] 43, [1] -109 [2] -66 [3] -124
[0] -89, [1] -10 [2] -30 [3] -127
[0] -111, [1] -81 [2] 63 [3] -127
[0] -70, [1] -109 [2] 47 [3] -124
[0] -110, [1] -113 [2] 89 [3] -121
[0] -99, [1] -90 [2] 58 [3] -125
```

  - Please edit main.cc file to show the predict gesture is which category on console.

```
[0] -128, [1] 11 [2] -13 [3] -126
prdeict gesture is 1
[0] -128, [1] 59 [2] -60 [3] -127
prdeict gesture is 1
[0] -127, [1] 34 [2] -36 [3] -127
prdeict gesture is 1
[0] -127, [1] 48 [2] -50 [3] -126
prdeict gesture is 1
[0] -128, [1] 66 [2] -69 [3] -125
prdeict gesture is 1
[0] -127, [1] 18 [2] -21 [3] -125
prdeict gesture is 1
```

Appendix – Himax Resource

- How to train your own Yolo-fastest model:
  - https://github.com/HimaxWiseEyePlus/Yolo-Fastest
- Himax yolo-fastest person dtection example:
  - https://github.com/HimaxWiseEyePlus/WE_I_Plus_User_Examples/tree/main/HIMAX_Yolo_Fastest_Person_Detection_Example
- SDK API 3 examples:
  - https://github.com/HimaxWiseEyePlus/bsp_tflu/tree/master/HIMAX_WE1_EVB_example/scenario_app/workshop_example
- Himax google tflm example
  - https://github.com/HimaxWiseEyePlus/himax_tflm
- PC tool:
  - https://github.com/HimaxWiseEyePlus/WE_I_Plus_User_Examples/releases/download/v1.0/PC_TOOL
- Edgeimpulse example:
  - https://github.com/HimaxWiseEyePlus/bsp_tflu/tree/master/HIMAX_WE1_EVB_example/scenario_app/edgeimpulse-example

Appendix - Update Bootloader Version

1. Please notice that following steps will erase all data (bootloader and application) in the HIMAX WE1 EVB flash.
2. Download FT4222 Linux driver. (https://www.ftdichip.com/Support/SoftwareExamples/libft4222-linux-1.4.4.9.tgz).
3. Install FT4222 driver by typing
   - install4222.sh
4. Assign access right to usb device, go to directory
   - cd /etc/udev/rules.d/
   - create file with naming “99-ftdi.rules”, fill following data in file
   - # FTDI's ft4222 USB-I2C Adapter
     UBSYSTEM=="usb", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="601c", GROUP="plugdev", MODE="0666"
5. Download Flash tooland unzip it. (https://github.com/HimaxWiseEyePlus/bsp_tflu/releases/download/v1.0/FlashTool.zip)

Appendix - Update Bootloader Version

6. Download bootloader, current latest version is v1.4.4 (https://github.com/HimaxWiseEyePlus/bsp_tflu/releases/download/v1.0/himax_we1_bootloader_v1_4_4.bin
7. Copy bootloader to Flash Tool directory.
8. Connect HIMAX WE1 EVB via USB cable.
9. Make sure there is no serial terminal connect to HIMAX WE1 EVB.
10. Type following command to start update flash image
    - $sudo ./WE-I_ISP_Tool [bootloader name]
11. For example, download bootloader “himax_we1_bootloader_v1_4_4.bin” to flash

```
byr@by-HP-ProBook-430-G7:~/Downloads/FlashTool$ sudo ./WE-I_ISP_Tool himax_we1_b
ootloader_v1_4_4.bin
Start to erase.
Erase done.
programming FW image...
Waiting for device ack ...
[==================================================] 100 %

done.
Time:13(sec)
```

Appendix - Update Bootloader Version

12. After bootloader updated, you can double check the version by open terminal at your environment and press reset button on HIMAX WE1 EVB.

```
------------------------------------------------------------
 Himax WEI Boot loader
------------------------------------------------------------

embARC Build Time: Jan  4 2021, 13:44:14
Compiler Version: Metaware, 4.2.1 Compatible Clang 8.0.1
Boot loader Version : 1.4.4 (Date:Jan  4 2021)
chip version : 0x8535a1
cpu speed : 400000000 hz
spi speed : 50000000 hz
wake up evt:4
...secure lib version = 352380df9a347b1187d2361bfcd4455178a1ebcb
1st APPLICATION addr[3]=21000 (main-1966)
```