AI-IOT Vision Framework Design, Architecture and Implementation

Oct. 2020

Jianfeng Qin

Dongsheng Zhang

Lin Tao

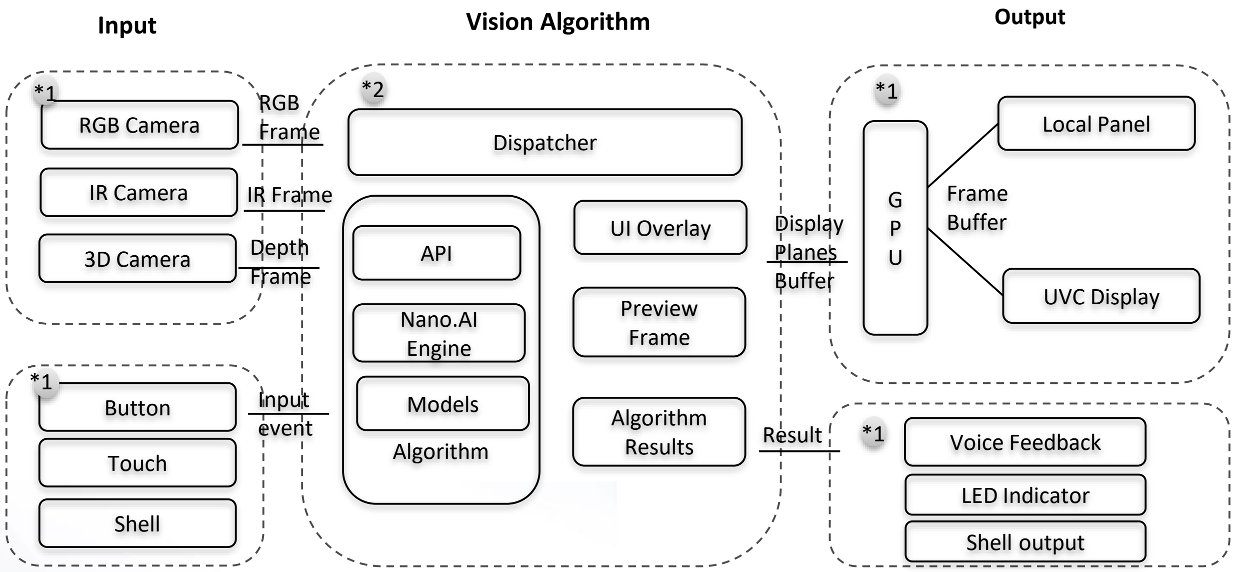
V1.1

NXP SOLUTIONS

AI-IOT Vision Framework

- **Contents**
  - AI-IOT Vision Solution Introduction
  - Customized modules
  - Vision framework architecture
  - Camera Dev
  - Display Dev
  - Input Dev

AI-IOT Vision Framework – AI-IOT Vision Solution Introduction



*1 : Hardware dependent *2 : Hardware independent

- ***Projects:***
  - *Vision 1.0 Face Rec*
  - *Vision 2.0 Face Rec*
  - *Vision 3.0 Face Rec*
  - *…*
- ***POC in pipeline:***
  - *Face Rec Tomy*
  - *Fisheye Face Rec*
  - *Gesture*
  - *Human Pose*
  - *…*

AI-IOT Vision Framework – Various Customized modules

- **Camera Interface**

	FlexIO	CSI	MIPI	UVC	MIPI 2 CSI
RT106F	Y	Y			Y
RT117F	Y	Y	Y		Y
Simulator				Y	

- **Camera sensor**

	MT9M114	GC0308	HiMax 3D	ORBBEC 3D	OV5640
RT106F	Y	Y	Y	Y	
RT117F	Y	Y	Y	Y	Y

- **Display Interface**

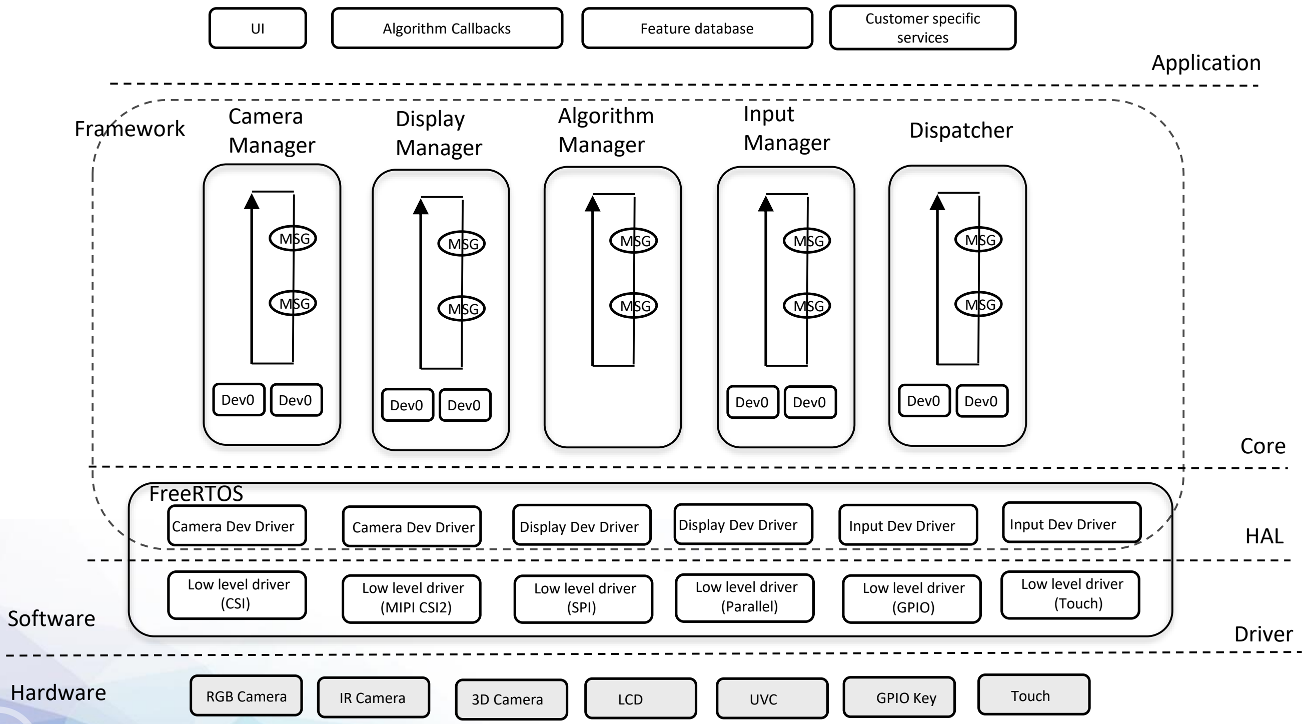
	Parallel	SPI	MIPI
RT106F	Y	Y	
RT117F	Y	Y	Y

- **Display Panel**

	Riverdi(spi,320x240)	RK024HH243-T(Parallel 240x320)	RK043FN02H-CT(Parallel 480x272)	RK055AHD091-CTG(MIPI 720p)	Others
RT106F	Y	Y			
RT117F		Y	Y	Y	
Customer					Y(each customer had their own panel)

- ***Requirements***
  - *Multiple platforms*
  - *Multiple interfaces*
  - *Multiple devices*
- ***Goal***
  - *Easy Customize*
    - *Modulable*
    - *Scalable*
    - *Flexible*
  - ***Agile development***
    - *Based on Ubuntu Simulator for fast dev*
    - *Build reusable component cross platform/product*

AI-IOT Vision Framework - Architecture



- ***Key concept***
  - *Core*
    - *Manage the input, algorithm, output*
    - *Only algorithm, workflow dependent*
    - *Only dependent on abstract hardware(HAL)*
  - *HAL(Hardware Abstract Layer)*
    - *Hide the low level hardware difference*
    - *Abstract the unified interface to Core*
  - ***Goal***
    - *Centralized and minimized software change*
      - *Hardware customization*

        Only add the HAL(dev driver)
      - *Software customization*

        Adjust the UI/Algorithm Callbacks

AI-IOT Vision Framework - Architecture

- **Core**
  - Dedicate manager to manage the multiple input devices
  - Fully leverage the FreeRTOS task and messaging mechanism
  - Define the unified message for the cross task communication
- **HAL**
  - Design the unified data structure and interface for Camera, display, input device
  - Provide the hardware independent interface to core
  - Build the dev database to support multiple devices, use the build time config to select different device for different platform(106F vs 117F, EVK vs Customer Kit)

AI-IOT Vision Framework- Camera Dev

- **Camera_dev**
  - id
  - camera dev operator
  - camera dev capability
- **Camera dev operator**
  - init
  - deinit
  - start
  - enqueue
  - dequeue
- **Camera dev capability**
  - width,
  - height
  - pixel_format
  - callback
  - param
- **Register camera dev to manager**

```
typedef struct _camera_dev camera_dev;

// callback funtion to notify camera manager that one frame is dequeued
typedef  int (*camera_dev_callback_t)(const camera_dev* dev, void* param);

typedef struct {
    // initialize the dev
    int (*init)(camera_dev* dev, int width, int height, camera_dev_callback_t callback,
void* param);
    // deinitialize the dev
    int (*deinit)(camera_dev* dev);
    // start the dev
    int (*start)(const camera_dev* dev);
    // enqueue a buffer to the dev
    int (*enqueue)(const camera_dev* dev, void* data);
    //dequeue a buffer from the dev
    int (*dequeue)(const camera_dev* dev, void** data);
} camera_dev_operator;

typedef struct {
    // resolution
    int width;
    int height;
    // pixel format
    pixel_format format;
    // callback
    camera_dev_callback_t callback;
    // param for the callback
    void* param;
} camera_dev_private_capability;

struct _camera_dev {
   // unique id which is assigned by camera manager during the registration
    int id;
    // operations
    const camera_dev_operator* ops;
   // private capability
    camera_dev_private_capability cap;
};
```

AI-IOT Vision Framework- Display Dev

- **display_dev**
  - id
  - display dev operator
  - display dev capability
- **dispaly dev operator**
  - init
  - deinit
  - start
  - blit
- **display dev capability**
  - width,
  - height
  - pixel_format
  - callback
  - param
- **Register display dev to manager**

```
typedef struct _display_dev display_dev;

// callback funtion to notify display manager that one frame is blited
typedef  int (*display_dev_callback_t)(const display_dev* dev, void* param);

typedef struct {
    // initialize the dev
    int (*init)(const display_dev* dev, int width, int height, display_dev_callback_t
callback, void* param);
    // deinitialize the dev
    int (*deinit)(const display_dev* dev);
    // start the dev
    int (*start)(const display_dev* dev);
    // blit a buffer to the dev
    int (*blit)(const display_dev* dev, void* frame, int width, int height);
} display_dev_operator;

typedef struct {
    // resolution
    int width;
    int height;
    // pixel format
    pixel_format format;
    // callback
    display_dev_callback_t callback;
    // param for the callback
    void* param;
} display_dev_private_capability;

struct _display_dev {
    // unique id which is assigned by camera manager during the registration
    int id;
    // operations
    const display_dev_operator* ops;
    // private capability
    display_dev_private_capability cap;
};
```

AI-IOT Vision Framework- Input Dev (TBD)

- **input_dev**
  - id
  - input dev operator
  - input dev capability
- **input dev operator**
  - init
  - deinit
  - start
  - key
- **input dev capability**
  - callback
  - param
- **Register input dev to manager**

AI-IOT Vision Framework – Resource

- **Repository**

  -Link for the release(TBD)

- **How to**
  - How to add one camera dev?
  - How to add one display dev?
  - How to customize the GPIO key?
  - How to change the UI?
  - How to adjust the database?