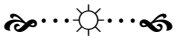
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA



ĐỒ ÁN MÔN HỌC THIẾT KẾ LUẬN LÝ (CO3091)

BÁO CÁO GIAI ĐOẠN 2

Lớp L08 - Nhóm 3 - HK221

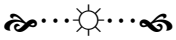
Giảng viên hướng dẫn: Huỳnh Phúc Nghị

STT	Sinh viên thực hiện	Mã số sinh viên
1	Phạm Duy Quang	2011899
2	Hà Vĩnh Nguyên	2011698
3	Đỗ Thành Minh	2011610
4	Phạm Đức Thắng	2012080

Thành phố Hồ Chí Minh, tháng 11 năm 2022

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA



ĐỀ TÀI: ỨNG DỤNG PHÁT HIỆN VẠCH KẺ ĐƯỜNG, VẠCH KẺ Ô ĐỖ XE, VẠCH TRẮNG CHỈ ĐƯỜNG,... THÔNG QUA HÌNH ẢNH THU ĐƯỢC TỪ CAMERA

STT	Sinh viên thực hiện	Phân công
1	Phạm Duy Quang	Đề xuất giải thuật, hiện thực giải thuật trên phần cứng, kiểm thử, viết báo cáo về các bước hiện thực giải thuật trên phần cứng và tổng hợp báo cáo
2	Hà Vĩnh Nguyên	Hỗ trợ chỉnh sửa và hiện thực, kiểm thử giải thuật trên phần cứng
3	Đỗ Thành Minh	Tìm hiểu thiết bị phần cứng, viết báo cáo về cách hiện thực giải thuật trên phần cứng
4	Phạm Đức Thắng	Hỗ trợ tìm hiểu, đề xuất, hiện thực giải thuật trên phần cứng

Thành phố Hồ Chí Minh, tháng 11 năm 2022

MỤC LỤC

I. Báo cáo giai đoạn 2

Link Github về Project của nhóm 3 trong giai đoạn 2: https://github.com/ULTIMATE-Mystery/Logic-Design-Project-Group-3-Semester-221-HCMUT/tree/Phase-2.

1. Nội dung đã làm được trong giai đoạn vừa rồi là gì?

- Chọn ra phần cứng phù hợp với kinh tế và nguồn lực của nhóm để thực hiện đồ án giai đoạn 2, đó là ESP32-CAM:

- ESP32-CAM được đánh giá là module Wifi giá rẻ hoàn toàn phù hợp cho các dự án tự làm trong lĩnh vực Internet of Things (IoT). Các tính năng đi kèm với GPIOs, hỗ trợ cho một loạt các giao thức như SPI, I2C, UART,…

- Giá thành tương đối rẻ, kích thước nhỏ gọn, có camera tích hợp trong module.

- Được sử dụng phổ biến trong nhiều dự án, tạo nên cộng đồng hỗ trợ mạnh và rộng lớn.

- Dòng tiêu thụ ở chế độ deep sleep giúp tiết kiệm điện năng.

- Có thể dễ dàng lắp đặt trên breadboard hoặc tích hợp lên bo mạch sản phẩm một cách chắc chắn.

- Chi tiết đặc tính và thông số kỹ thuật của module cực kỳ tiện dụng này được liệt kê dưới đây:

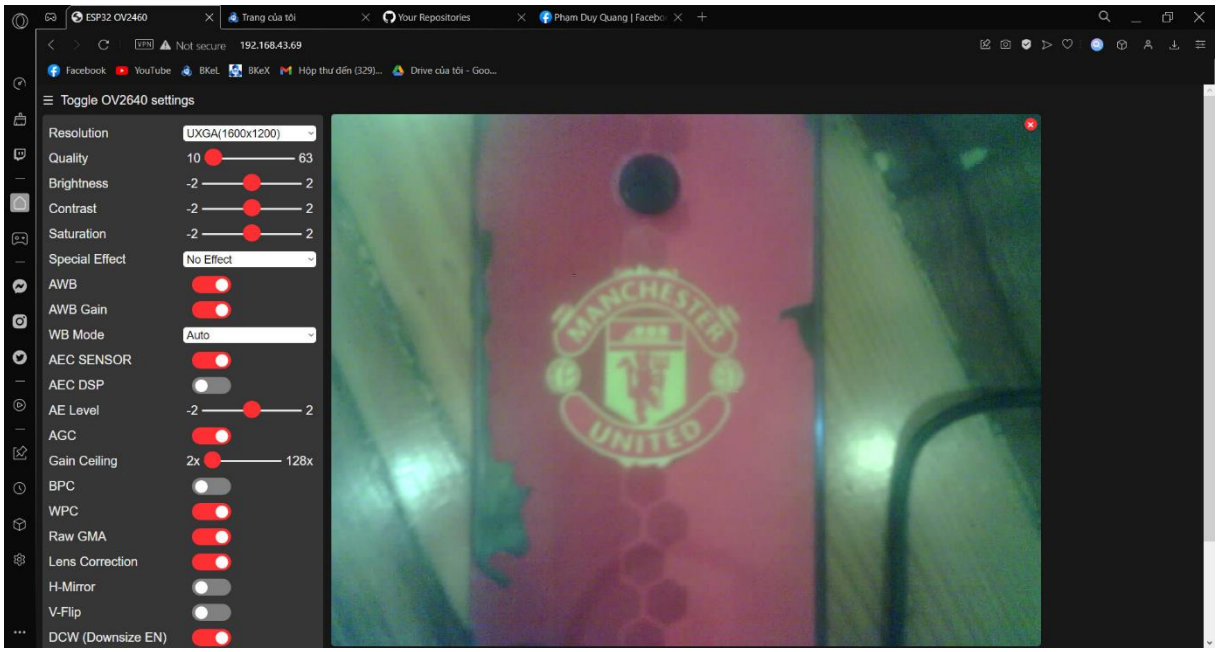
Module	ESP32-CAM AI-Thinker
Bộ xử lý	LX6 32-bit
Kiến trúc vi mạch	Đường dữ liệu 7 lớp
SPI Flash	32-bit
SRAM	520 KB
PSRAM	4 MB

Dãy nhiệt độ hoạt động	$-20℃ - 85℃$
Điện áp hoạt động	5V
UART	1
Tốc độ truyền UART	115200 bps
SPI	Có
I2C	Có
PWM	Có
Wifi	802.11 b / g / n
Bluetooth	Bluetooth 4.2 BR / EDR theo tiêu chuẩn BLE
Hỗ trợ thẻ TF	Hỗ trợ tối đa 4G
Bảo mật dữ liệu	WPA / WPA2 / WPA2-Enterprise / WPS
Định dạng hình ảnh đầu ra	BMP, GRAYSCALE, JPEG (chỉ hỗ trợ OV2640)
Số cổng đầu vào / đầu ra	9
Công suất tiêu thụ	Khi tắt đèn flash tiêu tốn: 180mA ở 5V. Bật đèn flash với độ sáng tối đa: 310mA ở 5V. Chế độ ngủ sâu (sleepmode): có mức tiêu thụ điện năng thấp nhất có thể đạt 6mA ở 5V. Modem-sleep mode: <20mA ở 5V, slight-sleep mode: <6.7mA ở 5V

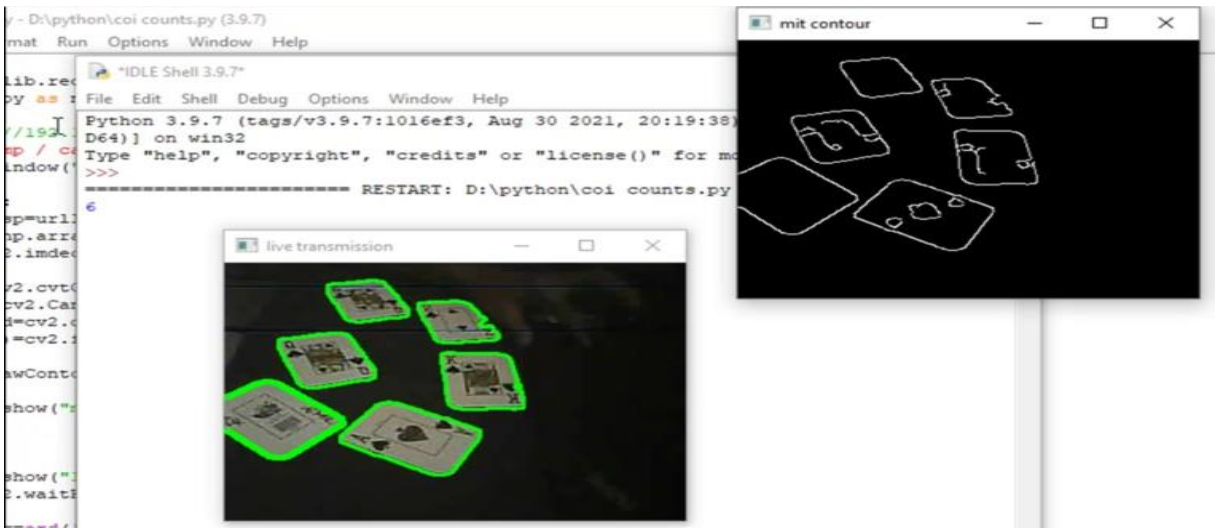
Tần số xung nhịp	240 MHz (tối đa)
Loại package	DIP-16
Kích cỡ package	27 × 40,5 × 4,5 (±0,2) mm

- Truyền được hình ảnh trực tiếp từ camera của ESP32-CAM lên local server, sau đó lấy hình ảnh từ local server về máy để xử lý ảnh thời gian thực bằng Python (tuy nhiên điều này chưa đúng với yêu cầu của đồ án là phải xử lý ảnh trực tiếp trên vi xử lý).

Đưa hình ảnh trực tiếp từ camera ESP32-CAM lên local server:



Dùng Python lấy những hình ảnh trực tiếp từ local server về xử lý ảnh trên thời gian thực:



- Nhận biết rằng việc xử lý hình ảnh trực tiếp trên những vi xử lý là rất khó khăn vì sự giới hạn về thư viện (không có thư viện OpenCV) cùng với tốc độ xử lý và bộ nhớ giới hạn.
- Nhóm chuyển sang thực hiện các thao tác xử lý ảnh trên nền tảng FPGA, mặc dù đã gặp không ít khó khăn nhưng nhóm đã đạt được một số kết quả nhất định.
- Nhận biết rằng với xử lý hình ảnh thì thư viện OpenCV là khó có thể thiếu cũng như những thiết bị phần cứng phù hợp cho các tác vụ xử lý ảnh (Raspberry Pi – một máy vi tính rất nhỏ gọn, FPGA, các vi xử lý hỗ trợ thư viện OpenMV,…).

2. Nội dung dự định làm trong giai đoạn tiếp theo là gì?

- Hoàn thiện hơn nữa các thao tác xử lý ảnh trên nền tảng FPGA để thực nghiệm trên thực tế.
- Đánh giá thực nghiệm và so sánh hiệu năng trên các nền tảng phần cứng khác nhau.
- Hoàn thiện báo cáo tổng kết về đồ án.

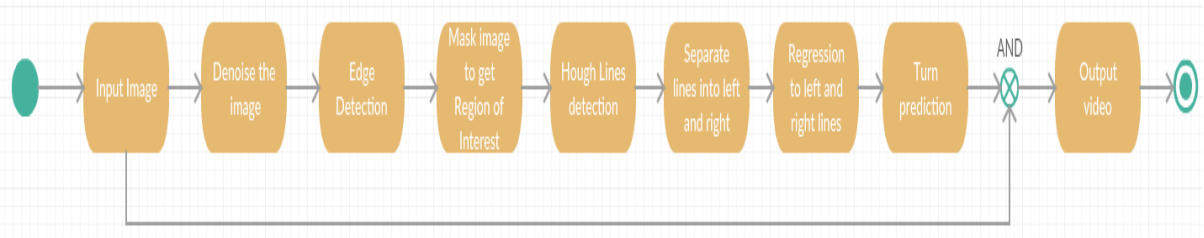
3. Khó khăn nào gặp phải trong giai đoạn vừa rồi?

- Với nguồn lực, kinh tế và thời gian giới hạn, nhóm khó tiếp cận được các thiết bị phần cứng với hiệu năng xử lý cao như FPGA mà chỉ có thể chọn những thiết bị vi xử lý với hiệu năng, tốc độ xử lý và bộ nhớ lưu trữ thấp.
- Việc hiện thực các tác vụ xử lý hình ảnh mà không có thư viện OpenCV (ESP32-CAM không hỗ trợ) khiến cho nhóm gặp rất nhiều khó khăn. Mặc dù đã rất cố gắng, nhóm chỉ có thể thực hiện một vài tác vụ xử lý hình ảnh mà không thể xử lý toàn bộ trực tiếp trên vi xử lý ESP32-CAM mà nhóm đã chọn.

II. Nội dung báo cáo

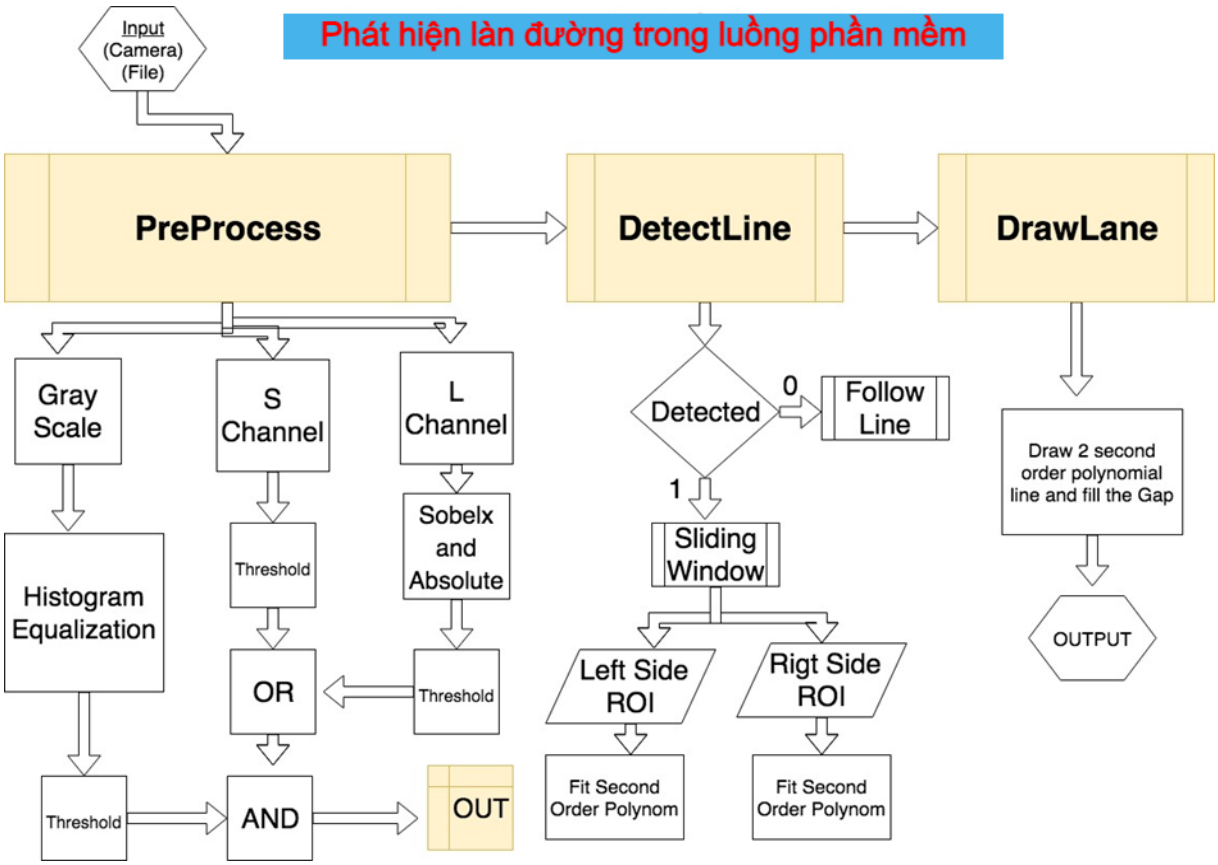
1. Thiết kế sơ đồ khối

1.1. Xử lý trực tiếp trên vi xử lý ESP32-CAM

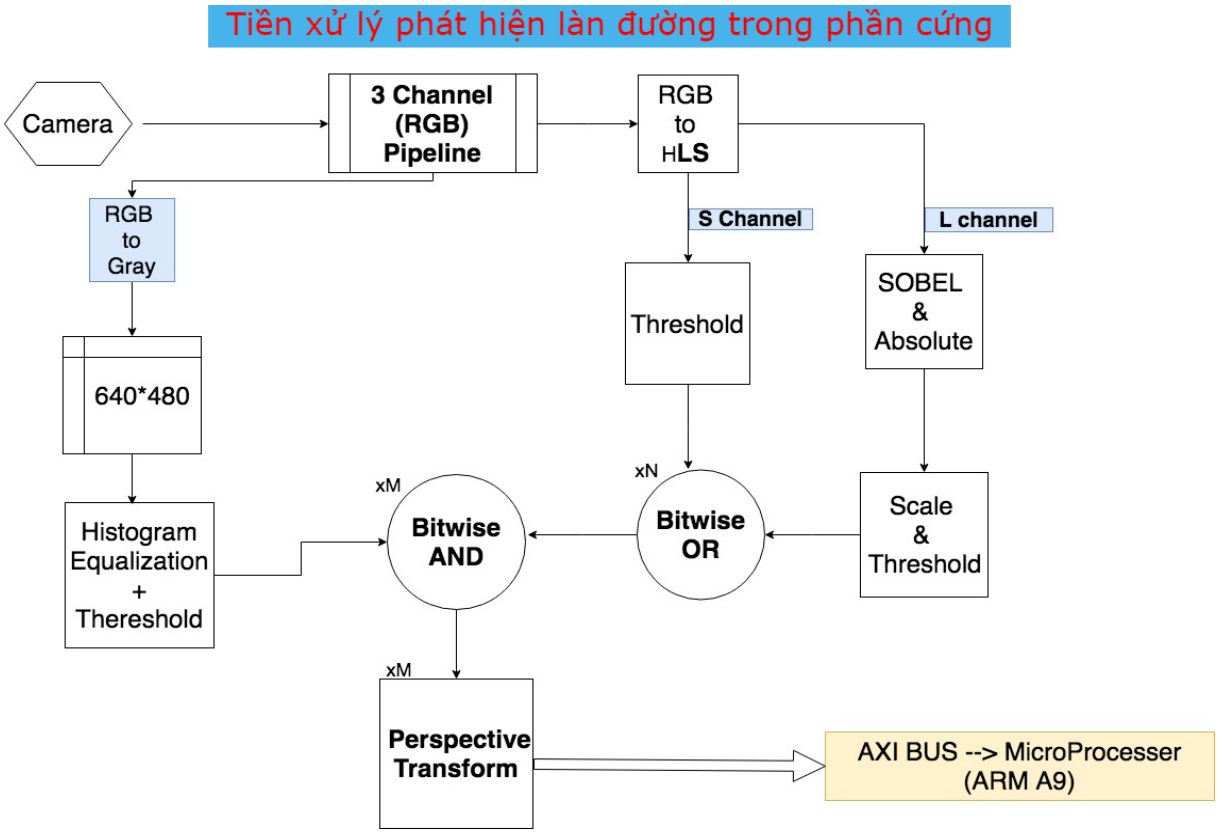


1.2. Xử lý trên nền tảng FPGA

1.2.1. Xử lý trong luồng phần mềm



1.2.2. Tiền xử lý trên phần cứng



2. Hiện thực từng khối xử lý

2.1. Hiện thực trên ESP32-CAM

Dù rất cố gắng nhưng nhóm đã không thể xử lý ảnh toàn bộ trực tiếp trên ESP32-CAM. Những gì nhóm đã thực hiện đó là:

- Tìm cách thêm thư viện OpenCV vào lập trình trên ESP32-CAM nhưng thất bại.

- Tìm cách thực hiện từng bước xử lý ảnh mà không dùng đến thư viện OpenCV nhưng vẫn thất bại, nhóm chỉ có thể thực hiện thao tác xử lý ảnh đơn giản đó là làm xám ảnh (grayscale) với đoạn code sau trên Arduino IDE:

```
#include "esp_camera.h"
#include "soc/soc.h"           // Disable brownour problems
#include "soc/rtc_cntl_reg.h"  // Disable brownour problems
#include "driver/rtc_io.h"
#include "img_converters.h"

// Pin definition for CAMERA_MODEL_AI_THINKER
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
```

```
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22



// our call back to dump whatever we got in binary format
size_t jpgCallBack(void * arg, size_t index, const void* data, size_t len)
{
  uint8_t* basePtr = (uint8_t*) data;
  for (size_t i = 0; i < len; i++) {
    Serial.write(basePtr[i]);
  }
  return 0;
}


void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout
detector


  Serial.begin(115200);


  camera_config_t config;
```

```
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_GRAYSCALE;

  if (psramFound()) {
    config.frame_size = FRAMESIZE_QQVGA; // FRAMESIZE_ +
QVGA|CIF|VGA|SVGA|XGA|SXGA|UXGA
    config.fb_count = 2;
```

```
  } else {
    Serial.println(F("ps RAM not found"));
    return;
  }

  // Init Camera
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

// disable white balance and white balance gain
  sensor_t * sensor = esp_camera_sensor_get();
  sensor->set_whitebal(sensor, 0);       // 0 = disable , 1 = enable
  sensor->set_awb_gain(sensor, 0);       // 0 = disable , 1 = enable

  camera_fb_t * fb = NULL;

  // Take Picture with Camera
  fb = esp_camera_fb_get();

  if (!fb) {
    Serial.println("Camera capture failed");
    return;
```

```
  }


  // DUMP THE PIXELS AS ASCII TO SERIAL
  Serial.printf("\n\nWidth = %u, Height=%u\n", fb->width, fb->height);
  for (size_t i = 0; i < fb->len; i++) {
    if (i % 16 == 0) Serial.printf("\n%06u\t", i);
    if (fb->buf[i] < 0x10) Serial.write('0');
    Serial.print(fb->buf[i], HEX);
  }


  Serial.println(F("\n\n----------------------\nPREPARE TO CAPTURE TO FILE\n"));
  delay(6000);


  frame2jpg_cb(fb, 10, jpgCallBack, NULL);


  esp_camera_fb_return(fb);
}


void loop() {}
```

Khi chạy đoạn code trên, nó sẽ thiết lập ESP32-CAM và chụp ảnh (thang độ xám 160x120) và in ra Serial output với tốc độ 115200 baud tất cả các pixel (ta nhận được 19200 byte với giá trị chính xác cho 160x120 pixel thang độ xám trên 8 bit). Sau đó nó sẽ in ra Serial Monitor một dòng là “PREPARE TO CAPTURE TO FILE” và đợi khoảng 6 giây.

Và sau đó ta gọi một hàm để tạo tệp jpg từ bộ đệm khung (**frame2jpg_cb**). Bằng cách này, ta có thể xem dữ liệu pixel và cũng có thể tạo một tệp mà ta có thể xem trên máy tính của mình.

Mở một Serial software có thể xử lý dữ liệu nhị phân và khi thấy dòng "PREPARE TO CAPTURE TO FILE", kích hoạt tính năng chụp tệp của software đó để lấy luồng byte vào một tập tin. Sau khi hoàn thành, dừng chụp và đổi tên tệp thành định dạng **.jpg**.

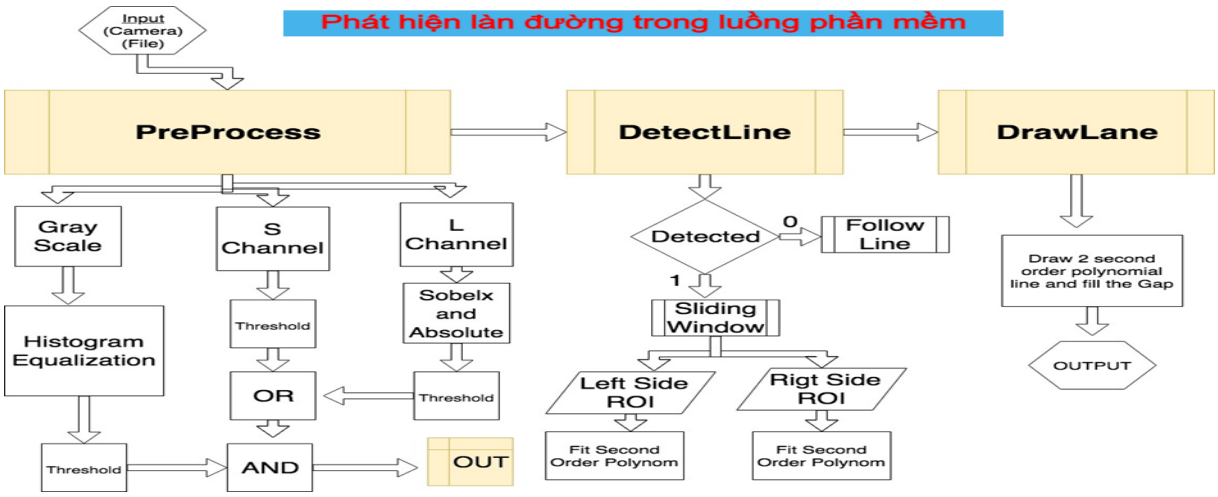
Đây là một ví dụ về hình ảnh nhận được:

2.2. Hiện thực trên nền tảng FPGA (phần cứng kết hợp với phần mềm)

Thuật toán phát hiện làn đường bao gồm ba phần chính: Tiền xử lý, phát hiện làn đường, vẽ làn đường. FPGA Zedboard dòng ZYNQ-7000 của Công ty Xilinx được sử dụng làm “Hệ thống trên Chip” (SoC) để giải quyết vấn đề này. Thuật toán phát hiện làn đường được triển khai trong FPGA sau khi thuật toán được chia ra thực hiện trên phần mềm và phần cứng.

Theo phân tích thời gian của thuật toán phát hiện làn đường, phần tiền xử lý đã được thực hiện trong phần cứng. Các nền tảng SDSoC và Vivado đã được sử dụng để thực hiện thuật toán phát hiện làn đường lập trình trên FPGA. Ngôn ngữ mô tả phần cứng (Verilog) được sử dụng trên nền tảng Vivado trong khi ngôn ngữ lập trình bậc cao (C++) được sử dụng trên nền tảng SDSoC. Trên nền tảng SDSoC, thư viện "xfOpenCV" cũng được sử dụng. Thư viện “xfOpenCV” là một bộ gồm hơn 50 nhân (kernels), được tối ưu hóa cho FPGA và SoC của Xilinx, dựa trên thư viện thị giác máy tính OpenCV. Các nhân trong thư viện xfOpenCV được tối ưu hóa và hỗ trợ trong Bộ công cụ Xilinx SDSoC.

2.2.1. Các bước xử lý giải thuật trên phần mềm



Đây là một số đầu ra trên các bước khác nhau của thuật toán:

a-) Original Frame (RGB)

b-) "L" Channel (HLS Color Space)

e-) "S" Channel (HLS Color Space)

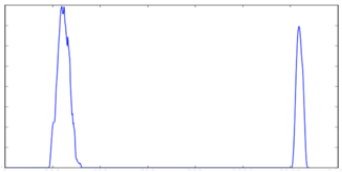
c-) Horizontal derivation

f-) Threshold

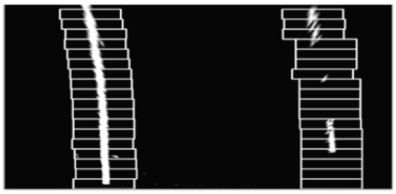
d-) Threshold

g-) Bitwise OR

h-) After Perspective Transform



i-) Histogram

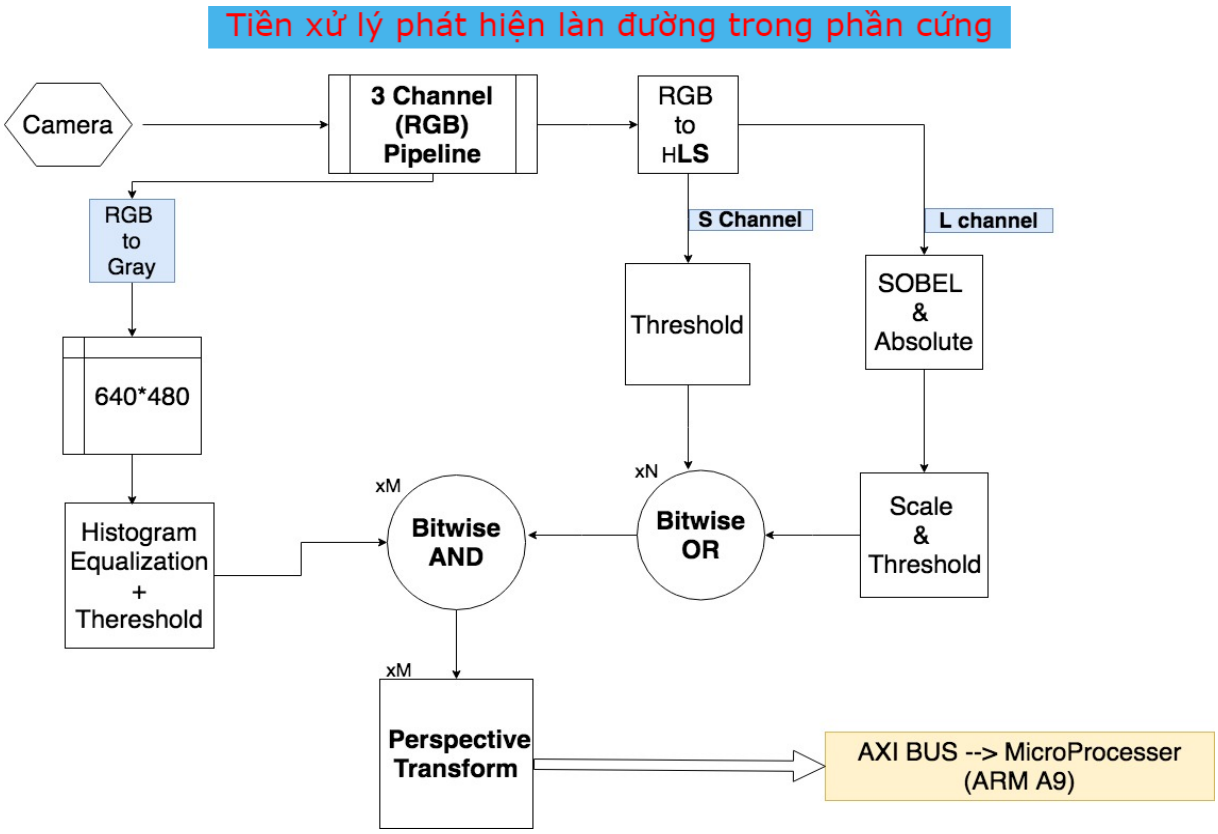


j-) Sliding Windows

2.2.2. Khối phần cứng và phần mềm trên FPGA

Một số phần của thuật toán thuộc hệ thống được triển khai trong phần cứng để hiện thực "Hệ thống phát hiện làn đường" nhanh hơn. Zedboard FPGA dòng ZYNQ-7000 của công ty Xilinx phù hợp với hệ thống vì thuật toán phát hiện làn đường có thể được tách biệt trên đó dưới dạng khối phần cứng và phần mềm. Khối phần mềm có sẵn trên bộ xử lý ARM Cortex-A9 trong Zedboard; khối phần cứng sẽ được triển khai trong các ô logic của Zedboard.

Thuật toán được chia thành 2 phần: Chức năng tiền xử lý được thực hiện trong phần cứng (logic lập trình) và các chức năng khác được thực hiện trong phần mềm (phần mềm lập trình – Arm-A9). Chức năng tiền xử lý nên được triển khai trong phần cứng do khả năng xử lý mạnh mẽ của nó. Sơ đồ khối hiện thực phần cứng của chức năng tiền xử lý trông như sau:



3. Kết nối các khối và kiểm tra tính chính xác

3.1. Trên vi xử lý ESP32-CAM

Nhóm đã thất bại trong việc hiện thực các thuật toán phát hiện làn đường trực tiếp trên ESP32-CAM mà chỉ có thể thực hiện các thao tác xử lý ảnh đơn giản như làm xám ảnh (grayscale)…

3.2. Trên nền tảng FPGA

Sử dụng thư viện "XfOpencv" với ngôn ngữ lập trình C++ thông qua nền tảng SDSoC (tổng hợp phần cứng cấp cao) trên Zedboard FPGA đã đạt được tốc độ xử lý khung hình/giây cao hơn cho "chức năng tiền xử lý" (không chuyển đổi phối cảnh) so với khi khối phần cứng không được sử dụng và triển khai trong bộ xử lý. Ta thấy rằng cấu trúc SoC được triển khai trên Zedboard với nền tảng SDSoC và với thư viện "XfOpencv" sử dụng ngôn ngữ lập trình C++ hiệu suất tốt hơn so với giải pháp phần mềm thuần túy trên bộ xử lý.

Trên nền tảng Vivado với ngôn ngữ lập trình mô tả phần cứng Verilog, chức năng "tiền xử lý" của hệ thống phát hiện làn đường đã được triển khai trên card phát triển Zedboard FPGA.

Kết luận: Khi thiết kế phần cứng được thực hiện, ngôn ngữ lập trình bậc cao có thể được sử dụng cho các giải pháp tham chiếu hoặc cho các giải pháp ngắn hạn. Mặt khác, ngôn ngữ lập trình bậc thấp phù hợp với các giải pháp dự kiến sẽ nâng cao hiệu suất sau này.

III. Tài liệu tham khảo

1. *[Tìm hiểu] ESP32-CAM AI-Thinker Board là gì?*, truy cập từ: https://blog.mecsu.vn/esp32-cam-ai-thinker-board-la-gi.

2. *Image processing on FPGA using Verilog HDL*, truy cập từ: https://www.fpga4student.com/2016/11/image-processing-on-fpga-verilog.html.