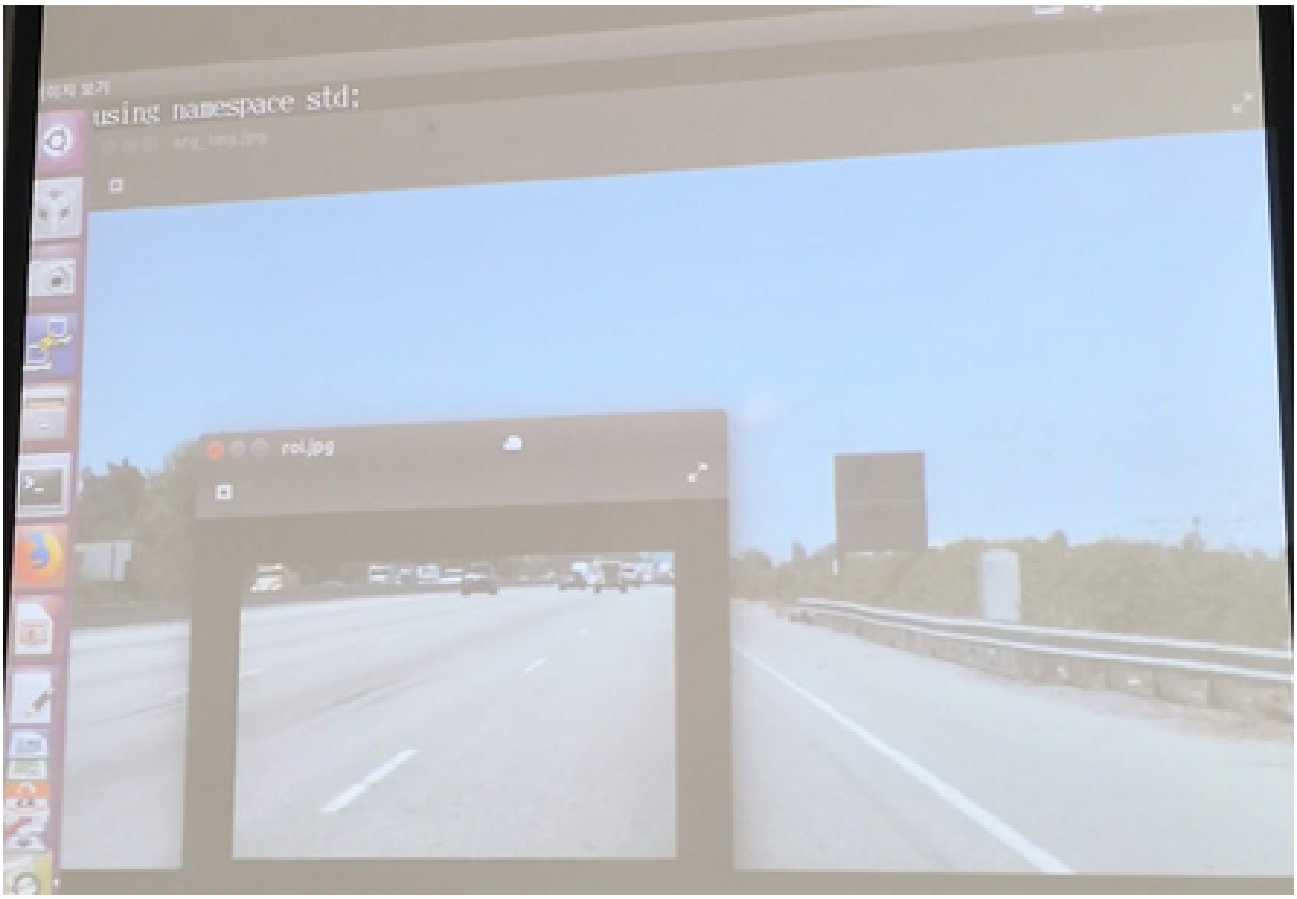
[2 일차]

[ROI (Region of Interesting) : fixed_roi.cpp]

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/opencv.hpp>
#include <iostream>
#include <stdio.h>

using namespace cv;
using namespace std;

int main(int argc, char **argv)
{
	Mat img = imread("sample.jpg", -1);
	int h=img.rows;
	int w=img.cols;
	printf("h=%d, w=%d\n", h, w);

	if(img.empty())
		return -1;

	printf("h-100 = %d, w-200 = %d, h-50=%d, w-100=%d\n", h-100, w-200, h-50, w-100);

	Mat roi = img(Range(100, 200), Range(50, 100));

	imwrite("org_img.jpg", img);
	imwrite("roi.jpg",roi);

	imshow("Origin Image", img);
	imshow("ROI Image", roi);

	waitKey(0);

	destroyWindow("Origin Image");
	destroyWindow("ROI Image");
	return 0;
}
```

```
g++ -I/usr/local/include/opencv -I/usr/local/include/opencv2 -L/usr/local/lib fixed_roi.cpp
-lrt -lopencv_core -lopencv_imgproc -lopencv_features2d -lopencv_imgcodecs -lopencv_highgui
```

```
root@am57xx-evm:~/gihwahong# ./a.out
h=360, w=480
h-100 = 260, w-200 = 280, h-50=310, w-100=380
init done
Using Wayland-EGL
wlpvr: PVR Services Initialised
```



<값 변경 테스트>

테스트 1)

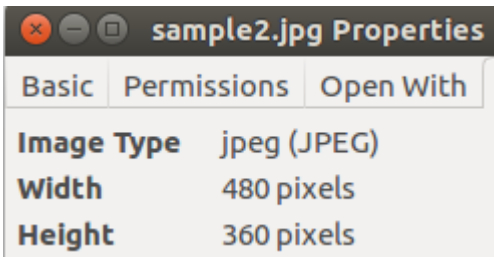
```
printf("h-50 = %d, w-200 = %d, h-50=%d, w-100=%d\n", h-50, w-200, h-50, w-100);
Mat roi = img(Range(50, 200), Range(50, 100));
```

테스트 1 결과 : 같은 너비에 세로 길이가 늘어났다.

전체 결과 : 일단 range(a, b), range(c, d) 에서 (b-a, d-c)값이 짤라내는 사진의 가로 세로를 결정

range(a,b)의 값이 자르는 세로 위치를 결정. 위(0) 아래(최대 360). a:시작. b:끝

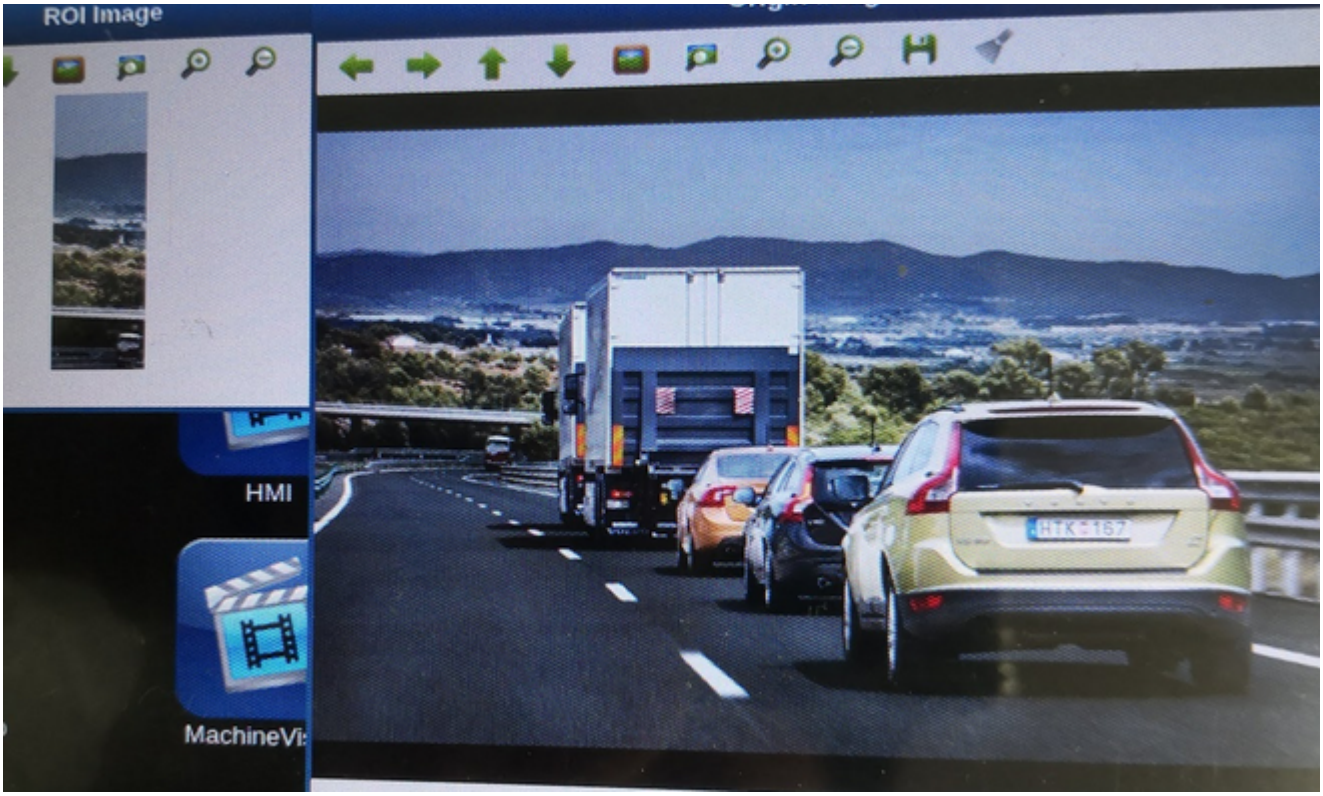
range(c,d)의 값이 자르는 가로 위치를 결정. 왼쪽(0) 오른쪽(480) c:시작. d:끝



전체 테스트)

Mat roi = img(Range(100, 150), Range(50, 75));

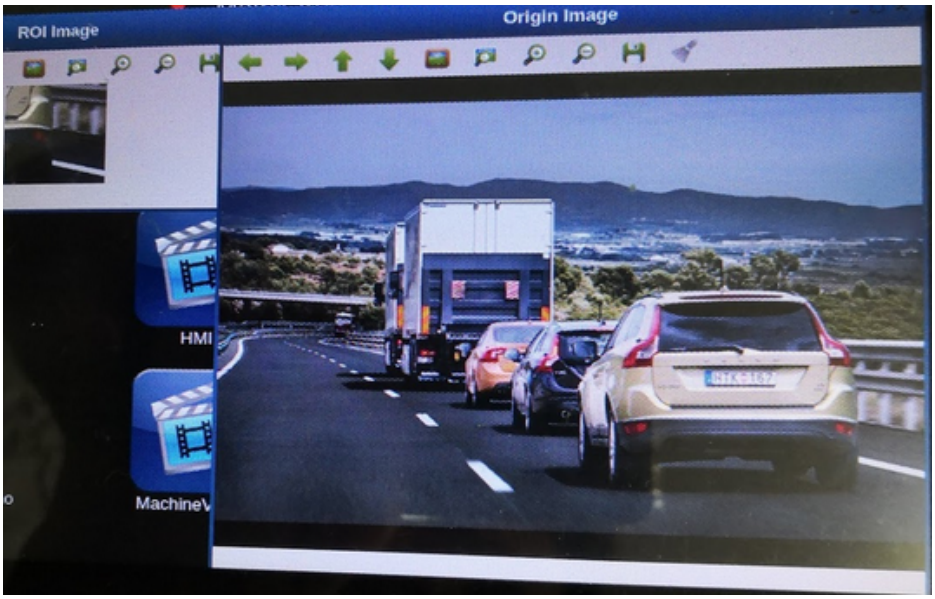
Mat roi = img(Range(50, 200), Range(50, 75));



Mat roi = img(Range(50, 200), Range(50, 200));

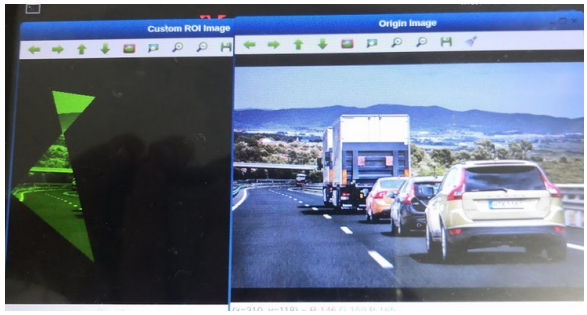
Mat roi = img(Range(200, 350), Range(200, 350));

Mat roi = img(Range(220, 300), Range(400, 480));



[CUSTOM ROI : 다각형 ROI custom_roi.cpp]

//ROI 는 사각만 자를 수 있는데, 이제 커스텀 ROI 로 다른 형태로 짤라본다.




```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/opencv.hpp>
#include <iostream>
#include <stdio.h>

using namespace cv;
using namespace std;

Mat custom_roi(Mat img)
{
	Mat output;
	Mat mask=Mat::zeros(img.size(), img.type());

	Point pts[4]={
		Point(50, 50),  //(x 좌표,y 좌표) 왼쪽위가 (0,0)임
		Point(110, 60),
		Point(0, 200),  // 점찍는 순서대로 다각형 그린다.
		Point(120, 300)
	};

	fillConvexPoly(mask, pts, 4, Scalar(255,0,0));
	// pts 에저장된 4 개 좌표로 블록다각형구성하고, bgr 세팅된 색 스케일로 채운다.->mask 로 넘김
	bitwise_and(img, mask, output);
	// img, mask 의 픽셀 AND 하여, output 으로 넘김.
	// https://stackoverflow.com/questions/44333605/what-does-bitwise-and-operator-exactly-do-in-opencv
	return output;
}

int main(int argc, char **argv)
{
	Mat img = imread("sample.jpg", -1);

	int h=img.rows;
	int w=img.cols;

	printf("h=%d, w=%d\n", h, w);

	if(img.empty())
		return -1;

	Mat croi=custom_roi(img);

	imwrite("org_img.jpg", img);
	imwrite("croi.jpg", croi);

	//namedWindow("Origin Image", cv::WINDOW_AUTOSIZE); 해도 이미지사이즈오버됨
	imshow("Origin Image", img);
	imshow("Custom ROI Image", croi);

	waitKey(0);

	destroyWindow("Origin Image");
	destroyWindow("Custom ROI Image");

	return 0;
}
```

[gaussian_blur : gaussian_blur.cpp]

<code>

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/opencv.hpp>
#include <iostream>
#include <stdio.h>

using namespace cv;
using namespace std;

int main(int argc, char **argv)
{
	Mat img = imread("sample.jpg", -1);
	if(img.empty())
		return -1;

	Mat blur;
	GaussianBlur(img, blur, Size(3,3),0,0);
		//gaussian(통계) 중간값, 감마잇을때 감마키움->중간값을 좀더 넓게 퍼지게 함. blur 효과

	imwrite("org_img.jpg", img);
	imwrite("blur.jpg", blur);

	imshow("Origin Image", img);
	imshow("Blur Image", blur);

	waitKey(0);

	destroyWindow("Origin Image");
	destroyWindow("Blur Image");

	return 0;
}
```

3. **Gaussian Filter:**

It is performed by the function GaussianBlur :

```
for ( int i = 1; i < MAX_KERNEL_LENGTH; i = i + 2 )
    { GaussianBlur( src, dst, Size( i, i ), 0, 0 );
      if( display_dst( DELAY_BLUR ) != 0 ) { return 0; } }
```

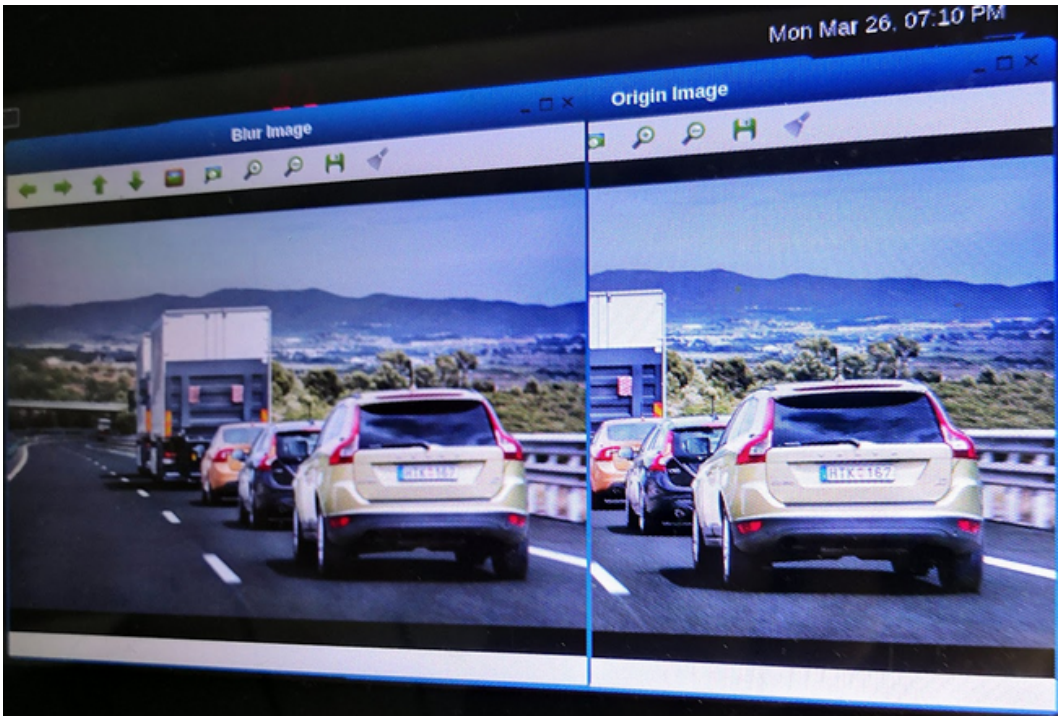
Here we use 4 arguments (more details, check the OpenCV reference):

- *src*: Source image
- *dst*: Destination image
- *Size(w, h)*: The size of the kernel to be used (the neighbors to be considered). w and h have to be odd and positive numbers otherwise thi size will be calculated using the σ_x and σ_y arguments.
- σ_x: The standard deviation in x. Writing 0 implies that σ_x is calculated using kernel size.
- σ_y: The standard deviation in y. Writing 0 implies that σ_y is calculated using kernel size.

gaussian blur 용도 : 연산 양 줄이기 위한 용도

< 결과사진 >

```
GaussianBlur(img, blur, Size(3,3),0,0);
```



왼쪽 사진의 결과 화면이 blur 되어있음을 확인 할 수 있다.

<테스트>

```
GaussianBlur(img, blur, Size(25,25),0,0);
```

[과제 :신호등 인식 mcu 모터 구동] traffic_light_motor.cpp

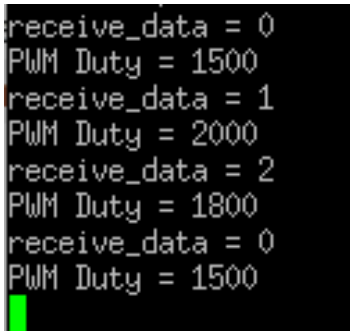
< 과제 구현 목표 >

신호등 빨 노 초 이미지구한다. → roi 해서 포커싱하고 빨강인지 노랑인지 캐치하고 → 모터 굴리기(노랑 : 풀쓰로틀 빨강: 정지, 초록: 보통)

어제는 rgb 입력했지만, 이제는 signal 이용해서 특정 시간마다 사진을 바꾼다.

<내 노트>

사진 랜덤 선택 → 가우시안 → 색추출 → UART 로 MCU 에 송신 → MCU 에서 받아서 조건 분기 모터 PWM 구동



쓰인 색 r g b 각각 모서리 버려버리고, 안에 픽셀들 다 더해 평균하면 평균값 rgb 어디에 근접하나 해서 판별할 수 있다.
→ 이미지 데이터가 클 경우 int 는 문제 생기니까 double 이나 float 써야한다.
DSP Instruction 쓰지 않으면, 처리가 느리다.

<mcu 코드> // 0 1 2 받으면, 분기 처리 pwm A3

```
#include "HL_sys_common.h"
#include "HL_gio.h"
#include "HL_sci.h"
#include "HL_etpwm.h"
#include "HL_system.h"

#include <string.h>
#include <stdio.h>
/* USER CODE END */

#define UART        sciREG1

void sci_display(sciBASE_t *sci, uint8 *text, uint32 len);
void pwm_init(void);
void wait(int);

uint32 receive_data;
uint32 value;
uint32 tmp;

int main(void)
{
/* USER CODE BEGIN (3) */
    char buf[256];
    unsigned int buf_len;
    gioInit();
    gioSetDirection(gioPORTA,0xffffffff);
    gioSetBit(gioPORTA,7,0);
    gioSetBit(gioPORTA,1,0);
    gioSetBit(gioPORTA,2,0);
    sciInit();
    etpwmInit();
```

```
    etpwmStartTBCLK();
    wait(10000);

    sprintf(buf, "SCI Configure Success!\n\r\0");
    buf_len = strlen(buf);
    sci_display(sciREG1, (uint8 *)buf, buf_len);

    sprintf(buf, "Press Proper Number!\n\r\0");
    buf_len = strlen(buf);
    sci_display(sciREG1, (uint8 *)buf, buf_len);

    for(;;)
    {
        tmp = sciReceiveByte(UART);

        receive_data = tmp - 48;

        sprintf(buf, "receive_data = %d\n\r\0", receive_data);
        buf_len = strlen(buf);
        sci_display(sciREG1, (uint8 *)buf, buf_len);

        pwm_init();

        sprintf(buf, "PWM Duty = %d\n\r\0", value);
        buf_len = strlen(buf);
        sci_display(sciREG1, (uint8 *)buf, buf_len);
    }
/* USER CODE END */

    return 0;
}

/* USER CODE BEGIN (4) */
void pwm_init(void)
{
    if(tmp == 48)  //ascii 0 :  0.4ms  전체주기 20ms    duty : 2%
    {
        gioSetBit(gioPORTA,7,0);
        gioSetBit(gioPORTA,1,0);
        gioSetBit(gioPORTA,2,0);

        gioSetBit(gioPORTA,1,1);

        etpwmSetCmpA(etpwmREG3, 1500);
        value = 1500;
        wait(10000);


    }
    else if(tmp == 49)
    {
        gioSetBit(gioPORTA,7,0);
        gioSetBit(gioPORTA,1,0);
        gioSetBit(gioPORTA,2,0);

        gioSetBit(gioPORTA,2,1);

        etpwmSetCmpA(etpwmREG3, 2000);
        value = 2000;
        wait(10000);
    }
    else if(tmp == 50)  // ascii 2 : 0.8ms  전체주기 20ms   duty : 4%
    {
        gioSetBit(gioPORTA,7,0);
        gioSetBit(gioPORTA,1,0);
        gioSetBit(gioPORTA,2,0);
```

```
        gioSetBit(gioPORTA,1,1);
        gioSetBit(gioPORTA,2,1);

        etpwmSetCmpA(etpwmREG3, 1800);
        value = 1800;
        wait(10000);
    }

}

void sci_display(sciBASE_t *sci, uint8 *text, uint32 len)
{
    while(len--)
    {
        while((UART->FLR & 0x4) == 4)
            ;
        sciSendByte(UART, *text++);
    }
}

void wait(int delay)
{
    int i;

    for(i = 0; i < delay; i++)
        ;
}
```

<**코드**> // 점하나 찍어서 확인. →

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/opencv.hpp>
#include <iostream>

#include <sys/types.h>
#include <sys/poll.h>
#include <termios.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <setjmp.h>
#include <fcntl.h>
#include <stdio.h>
#include <time.h>

#include "serial.h"

using namespace std;
using namespace cv;

extern char *dev0;

jmp_buf env;
int fd;
int flag = 0;
Mat img;
Mat croi;
int idx;
char name[32] = "";
char *traffic[4] = {"red_traffic.jpg", "yellow_traffic.jpg", "green_traffic.jpg"};

Point pts[3][4] = {
      {
            Point(150, 54),
            Point(183, 53),
            Point(181, 82),
            Point(149, 81)
      },
```

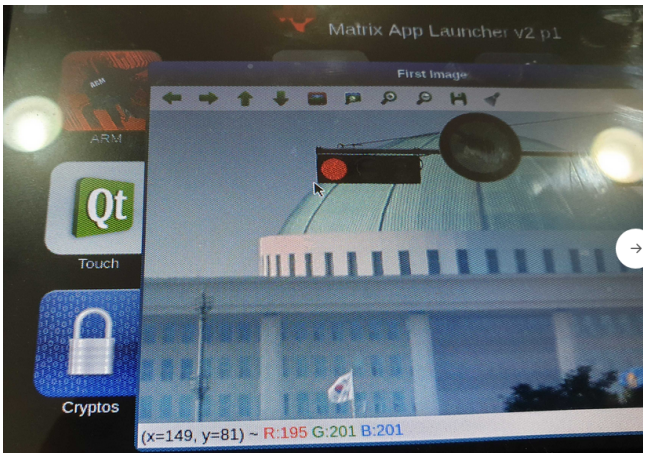
```
	{
		Point(60, 95),
		Point(85, 97),
		Point(86, 125),
		Point(60, 123)
	},
	{
		Point(332, 127),
		Point(352, 130),
		Point(353, 150),
		Point(334, 150)
	}
};

const Point *ppt1[1] = { pts[0] };
const Point *ppt2[1] = { pts[1] };
const Point *ppt3[1] = { pts[2] };

void call_exit(int signo)
{
	longjmp(env, 1);
}

void traffic_chg(int signo)
{
	idx = rand() % 3;
	strcpy(name, traffic[idx]);
	printf("idx = %d, name = %s\n", idx, name);

	img = cv::Mat::zeros(img.size(), img.type());
	img = imread(name, -1);

	flag = 1;
}

Mat custom_roi(Mat img, int idx)
{
	Mat output;
	Mat mask = Mat::zeros(img.size(), img.type());
	//Mat mask(img.size(), CV_8UC3);
	//Mat mask = Mat::zeros(img.size(), img.type());

#if 0
	Point pts[3][4];

	pts[0][0] = Point(328, 175);
	pts[0][1] = Point(384, 173);
	pts[0][2] = Point(385, 111);
	pts[0][3] = Point(330, 116);

	const Point *ppt[1] = { pts[0] };
#endif

	int npt[] = { 4 };

	switch(idx)
	{
		case 0:
			cv::fillPoly(mask, ppt1, npt, 1, cv::Scalar(255, 255, 255));
			   //fillPoly
			      https://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html
			break;
		case 1:
			cv::fillPoly(mask, ppt2, npt, 1, cv::Scalar(255, 255, 255));
			break;
		case 2:
			cv::fillPoly(mask, ppt3, npt, 1, cv::Scalar(255, 255, 255));
			break;
```

```
	}

	bitwise_and(img, mask, output);

	cout << output.type() << endl;

	return output;
}

#if 0
{
    Point(150, 54),
    Point(183, 53),
    Point(181, 82),
    Point(149, 81)
}
#endif

void chk_traffic_color(Mat croi, int idx)
{
    char buf[32] = "";
    printf("croi rows = %d, croi cols = %d\n", croi.rows, croi.cols);

	switch(idx)
	{
		case 0:
			printf("r = %d, g = %d, b = %d\n",
				croi.at<Vec3b>(68, 166)[0],
				croi.at<Vec3b>(68, 166)[1],
				croi.at<Vec3b>(68, 166)[2]);
			       //croi.at<Vec3b>(166, 68)[0],
			       //croi.at<Vec3b>(166, 68)[1],
			       //croi.at<Vec3b>(166, 68)[2]);

			sprintf(buf, "%d", 1);
			printf("buf = %s\n", buf);
			//send_data(fd, buf, 1, 0);

			break;
		case 1:
			printf("r = %d, g = %d, b = %d\n",
				croi.at<Vec3b>(109, 76)[0],
				croi.at<Vec3b>(109, 76)[1],
				croi.at<Vec3b>(109, 76)[2]);

			sprintf(buf, "%d", 2);
			printf("buf = %s\n", buf);
			//send_data(fd, buf, 1, 0);

			break;
		case 2:
			printf("r = %d, g = %d, b = %d\n",
				croi.at<Vec3b>(138, 341)[0],
				croi.at<Vec3b>(138, 341)[1],
				croi.at<Vec3b>(138, 341)[2]);

			sprintf(buf, "%d", 3);
			printf("buf = %s\n", buf);
			//send_data(fd, buf, 1, 0);

			break;
	}
}

int main(int argc, char **argv)
{
	int nr, fd;
	int x, y, w, h;
```

```
	char buf[32] = "";
	char test_img[32] = "green_traffic.jpg";

	int ret;
	int wait_time;

	signal(SIGINT, call_exit);    // Ctrl+C signal 시 call_exit
	signal(SIGALRM, traffic_chg); // alarm 이 SIGALRM 발생 시 traffic_chg() 실행

	srand(time(NULL));
	//fd = serial_config(dev0);
	fd = 1;

	printf("Automatic Traffic Light\n");
	img = imread(test_img, -1);
	//imshow("Green", img);
	//waitKey(0);
	//destroyWindow("Green");

	if(!(ret = setjmp(env)))
	{
		for(;;)
		{
			alarm(0);               // alarm 초기화.
			wait_time = rand() % 1 + 2;   //시간 1~2 초 랜덤
			alarm(wait_time);             //alarm setting

			while(!flag)                  //flag=1 ← traffic_chg()되면,
				;

			waitKey(wait_time * 1000);    //1~2 초 기다림

			flag = 0;                     // flag 초기화

			//cvtColor(img, img, COLOR_BGR2HSV);

			croi = custom_roi(img, idx);  // roi

			//send_data(fd, buf, 1, 0);
			//printf("\n");

			// TODO - Something wrong
			chk_traffic_color(croi, idx); //traffic 신호 색 판별

			imshow("Custom ROI Image", croi);

#if 0
			red.png -> 219, 17, 29
			yellow.jpg -> 251, 246, 84
			green.jpg -> 145, 145
#endif

			//memset(buf, 0x0, 32);
		}
	}
	//close_dev(fd);
	return 0;
}
```

<실행 결과>

```
r = 0, g = 0, b = 0
buf = 1
^Croot@am57xx-evm:~/sanghoon/ff# ./a.out
Automatic Traffic Light
idx = 2, name = green_traffic.jpg
16
croi rows = 393, croi cols = 699
r = 0, g = 0, b = 0
buf = 3
init done
Using Wayland-EGL
wlpvr: PVR Services Initialised
idx = 0, name = red_traffic.jpg
16
croi rows = 302, croi cols = 540
r = 0, g = 0, b = 0
buf = 1
idx = 2, name = green_traffic.jpg
```




```
16
croi rows = 274, croi cols = 399
r = 60, g = 254, b = 237
buf = 2
idx = 2, name = green_traffic.jpg
16
croi rows = 393, croi cols = 699
r = 205, g = 239, b = 108
buf = 3
idx = 0, name = red_traffic.jpg
16
croi rows = 302, croi cols = 540
r = 45, g = 72, b = 246
buf = 1
idx = 0, name = red_traffic.jpg
16
croi rows = 302, croi cols = 540
r = 45, g = 72, b = 246
buf = 1
idx = 0, name = red_traffic.jpg
16
croi rows = 302, croi cols = 540
r = 45, g = 72, b = 246
buf = 1
idx = 1, name = yellow_traffic.jpg
16
croi rows = 274, croi cols = 399
r = 60, g = 254, b = 237
buf = 2
```