Winter School

How to Handle Multiple Camera in Computer vision

김태주

목차

1. 자율 주행 자동차를 위한 센서
2. 자율 주행 자동차를 위한 영상 센서
    1. 영상 센서란?
        1. CCD/CMOS + Global shutter, Rolling Shutter
    2. 적외선 카메라
    3. RGB 카메라 (Bayer filter)
    4. 조리개와 포컬
    5. 카메라 인터페이스
3. Multiple Camera
    1. 동기화
        1. S/W
        2. H/W

목차

4. Embedded

1. Embedded, Edge Computer란?
2. Nvidia Jetson (실습)
    1. 리눅스 (Ubuntu 18.04) 개발 환경 – Ubuntu에 대한 간략한 설명
    2. Qt, GUI 프로그래밍
        1. Pytorch 설치, lib과 include에 대해
        2. qmake와 .pro
        3. 버튼을 이용한 Camera Connect
    3. Grabber 제작
        1. Thread, Thread 프로그래밍
        2. Image Aquisition
    4. 대용량 시퀀스 영상 저장
        1. Recoding 실습

1. 자율 주행 자동차를 위한 센서

미국 자동차기술 학회(SAE)의 자율주행기술 발전 6단계

자동화단계	특징	내용
사람이 주행환경을 모니터링 함		
Level 0	비자동 (No Automation)	운전자가 전적으로 모든 조작을 제어하고, 모든 동적 주행을 조장하는 단계
Level 1	운전자 지원 (Driver Assistance)	자동차가 조향 지원시스템 또는 가속/감속 지원시스템에 의해 실행되지만 사람이 자동차의 동적 주행에 대한 모든 기능을 수행하는 단계
Level 2	부분 자동화 (Partial Automation)	자동차가 조향 지원시스템 또는 가속/감속 지원시스템에 의해 실행되지만 주행환경의 모니터링은 사람이 하며 안전운전 책임도 운전자가 부담
자율주행 시스템이 주행환경을 모니터링 함		
Level 3	조건부자동화 (Conditional Automation)	시스템이 운전 조작의 모든 측면을 제어하지만, 시스템이 운전자의 개입을 요청하면 운전자가 적절하게 자동차를 제어해야하며, 그에 따른 책임도 운전자가 보유
Level 4	고도 자동화 (High Automation)	주행에 대한 핵심제어, 주행환경 모니터링 및 비상시의 대처 등을 모두 시스템이 수행하지만 시스템이 전적으로 항상 제어하는 것은 아님
Level 5	완전 자동화 (Full Automation)	모든 도로조건과 환경에서 시스템이 항상 주행 담당

※ 자료 : 자율주행기술동향-기술수준 구분, 한국교통연구원, 2016.04.

그림 1. Level 3에 해당 하는 Uber 자율주행차 개요

1. 자율 주행 자동차를 위한 센서

GNSS

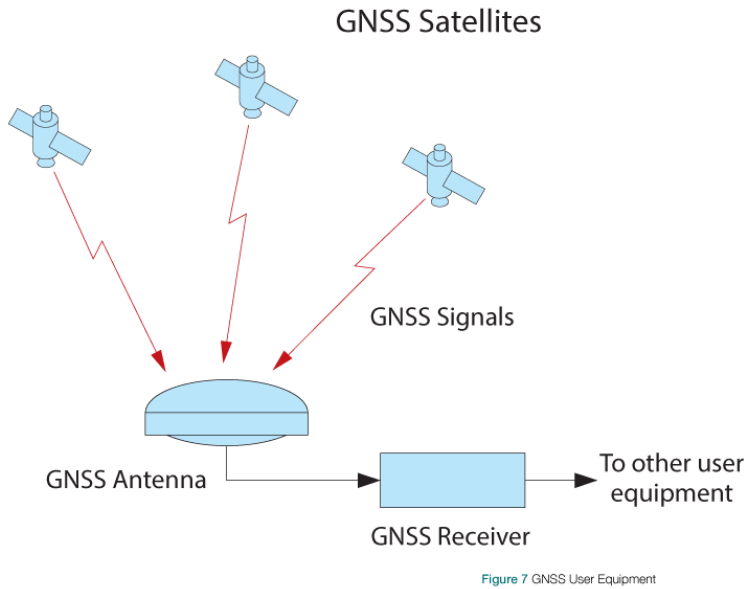
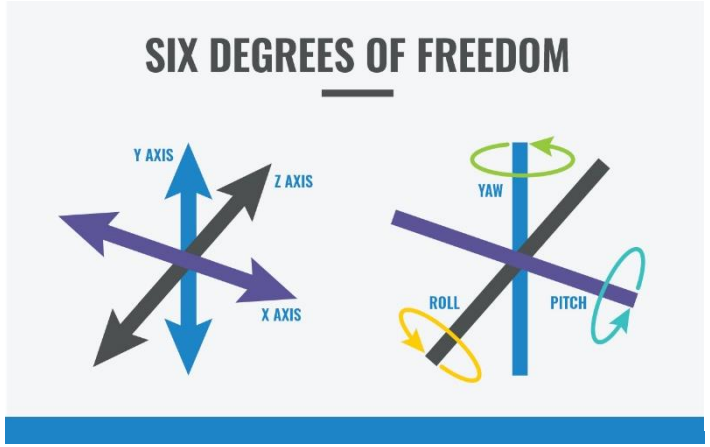


Figure 7 GNSS User Equipment

IMU



1. 자율 주행 자동차를 위한 센서

LiDAR

그림 1. Velodyne Puck

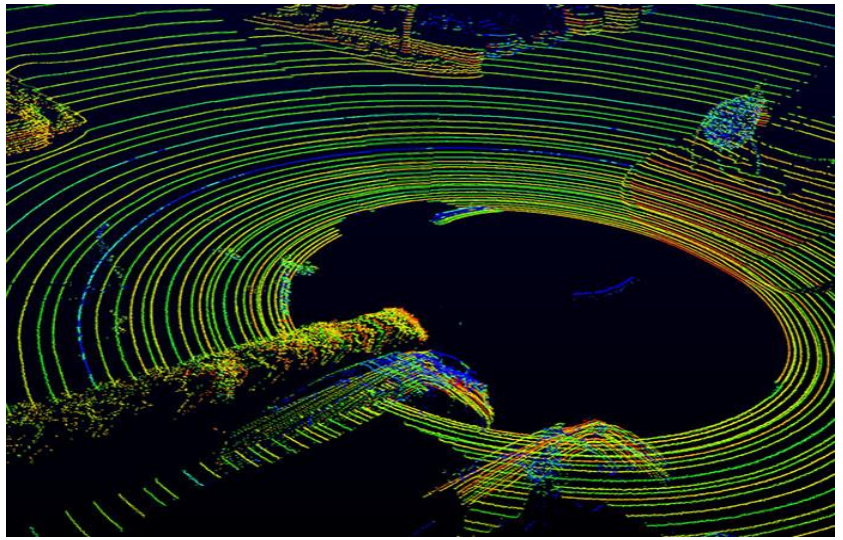


그림 2. LiDAR로 얻은 영상

1. 자율 주행 자동차를 위한 센서

rader

그림 1. 현대모비스에서 개발한 후측방 단거리 레이더

sonar

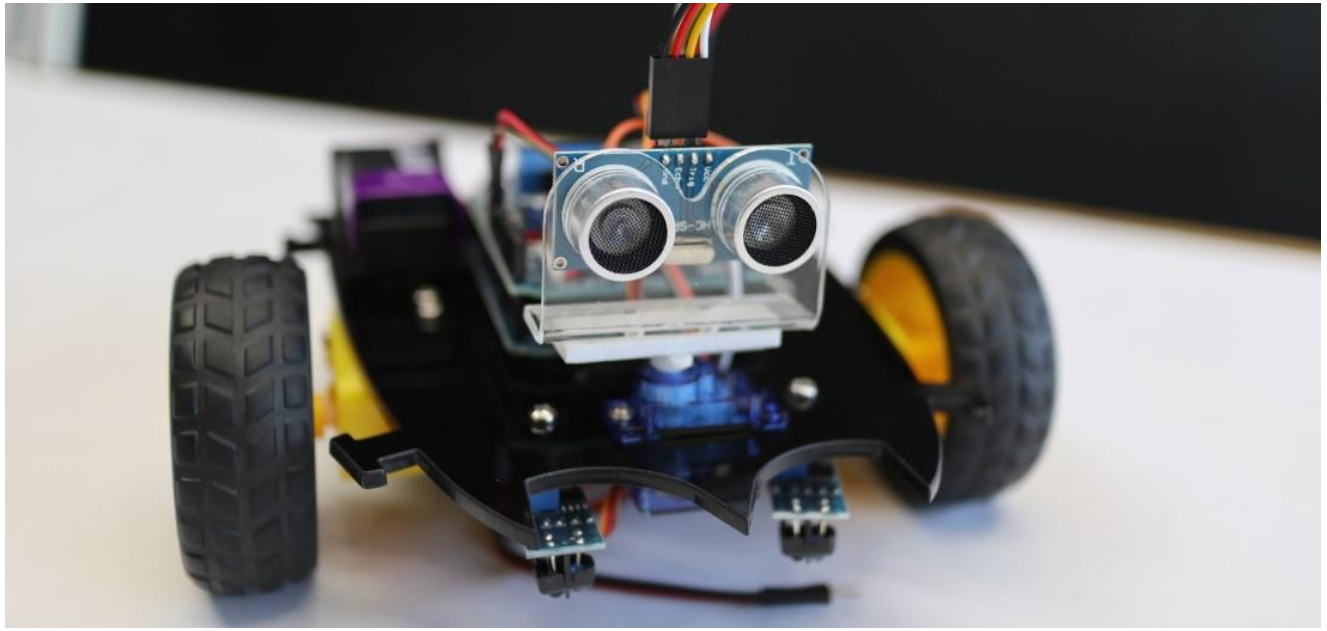
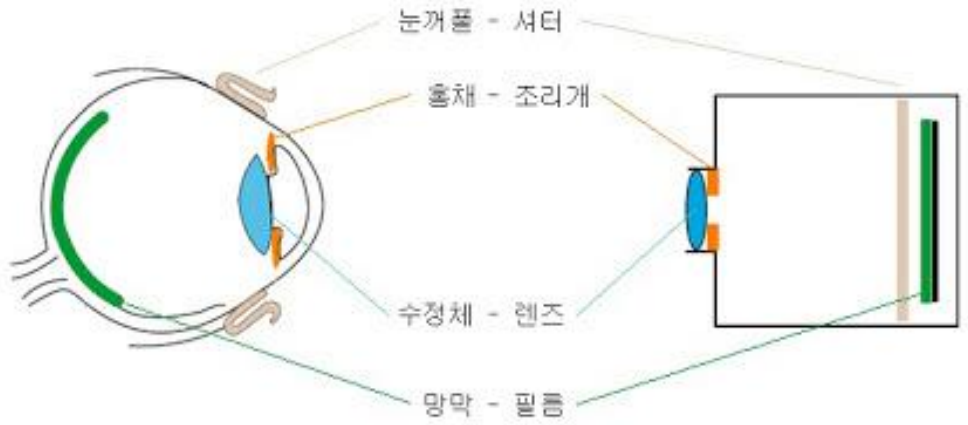


그림 2. 아두이노 초음파 센서

자율 주행 자동차를 위한 “영상” 센서

1. 영상 센서란? 렌즈를 통해 들어오는 빛을 전기적인 영상 신호(광다이오드)로 바꿔주는 센서



	Shutter	CCD vs. CMOS
CCD(Charged Couple Device)	Global Shutter	Horizontal shift register
CMOS(Complementary Metal-Oxide Semiconductor)	Rolling Shutter / Global Shutter	On-Chip

자율 주행 자동차를 위한 "영상" 센서

	장점	단점
CCD(Charged Couple Device)	- Block 단위로 셀을 처리하기 때문에 노이즈가 적음 - 글로벌 셔터 - 미세한 표현과 섬세한 색상구분이 가능	- Blooming effect - 가격이 상대적으로 비쌈 - 주변 회로가 복잡하다 - 원칩화가 불가능 - Smearing-effect
CMOS(Complementary Metal-Oxide Semiconductor)	- 단위 셀마다 증폭기 보유 - 범용적인 반도체 공정으로 제작이 가능하여 비교적 저비용 - 소비 전력이 적어 발열이 적음	- 셀마다 고정된 증폭기를 할당하기 때문에 증폭기 특성 차로 인한 노이즈 발생 - 고속 촬영시 롤링 셔터가 발생함

자율 주행 자동차를 위한 “영상” 센서

2. 적외선 카메라?

열(적외선 에너지)에서 발생되는 파장을 비접촉식으로 감지하여 전기 신호로 변환 후, 영상화하는 장치

그림 1. Night Vision

그림 2. Infrared Thermal Camera

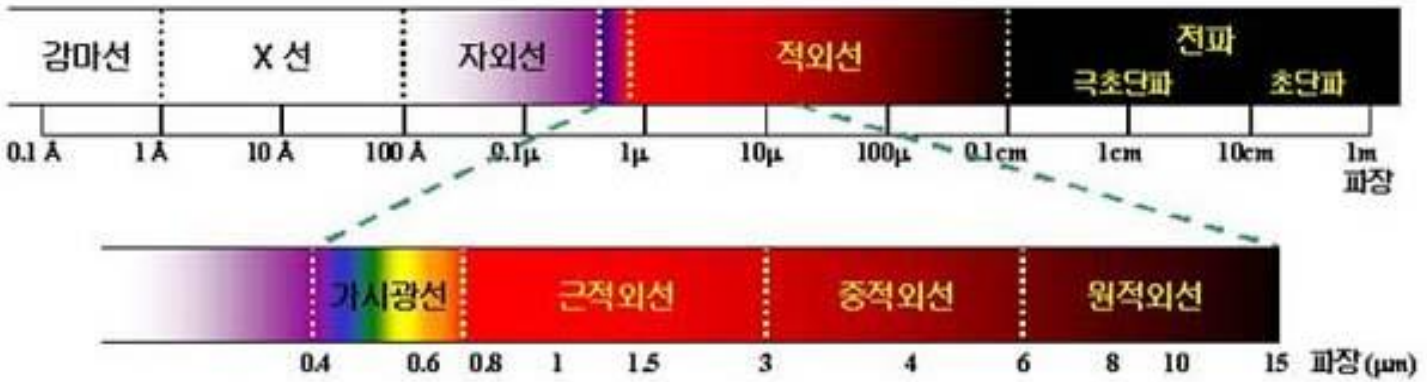


그림 4. 가시광선 및 적외선 영역의 스펙트럼 분포

그림 3. IR 조명을 이용한 Night vision

자율 주행 자동차를 위한 "영상" 센서

적외선 파장 종류

- Near infrared, NIR, IR DIN
  - 0.75~1.4um
  - 3,591 ~ 1,797 C
  - Using "Night vision camera"
- Mid-wavelength infrared, MidIR, IIR
  - 3 ~ 8um
  - 693 ~ 89 C
- Long-wavelength infrared, LWIR, IR-C
  - 8 ~ 15um
  - 89 ~ -80 C
  - "Thermal imaging" region
- Far-infrared, FIR
  - 15 ~ 1,000um
  - -80.15 ~ -270.15 C

The International Commission on Illumination (CIE) recommended the division of infrared radiation into the following three bands:

Abbreviation	Wavelength	Frequency
IR-A	700 nm ~ 1400 nm (0.7 µm ~1.4 µm)	215 THz ~ 430 THz
IR-B	1400 nm ~ 3000 nm (1.4 µm ~ 3 µm)	100 THz ~ 215 THz
IR-C	3000 nm ~ 1 mm (3 µm ~ 1000 µm)	300 GHz ~ 100 THz

그림 1. Night Vision

그림 2. Infrared Thermal Camera

참고 블로그

자율 주행 자동차를 위한 “영상” 센서

- Night vision
  - 가시광선 0.7 um 바로 위의 파장 LIR, SWIR(0.7 ~ 3)을 CCD로 측정
  - 0.7 ~ 3um는 열복사 스펙트럼
  - 그림 3. 처럼 IR 조명으로 적외선을 반사 받아 야간 투시가 가능하다.
- Infrared Thermal Camera
  - LWIR(8 ~ 15um)을 측정 (혹은 0.9~14um까지)
  - LWIR은 89 ~ -80 C, 즉 실생활에서 유용한 온도 범위를 가지고 있기에 LWIR을 주로 이용하여 측정

그림 1. Night Vision

그림 2. Infrared Thermal Camera

그림 3. IR 조명을 이용한 Night vision

자율 주행 자동차를 위한 "영상" 센서

3. Bayer filter

RGB 카메라의 이미지 센서(CCD, CMOS)는 화수 수만큼의 mono 셀들에
R, G, B 색상 필터들이 특정한 패턴을 가지고 배치되어
Raw Bayer data를 만들어 냅니다.

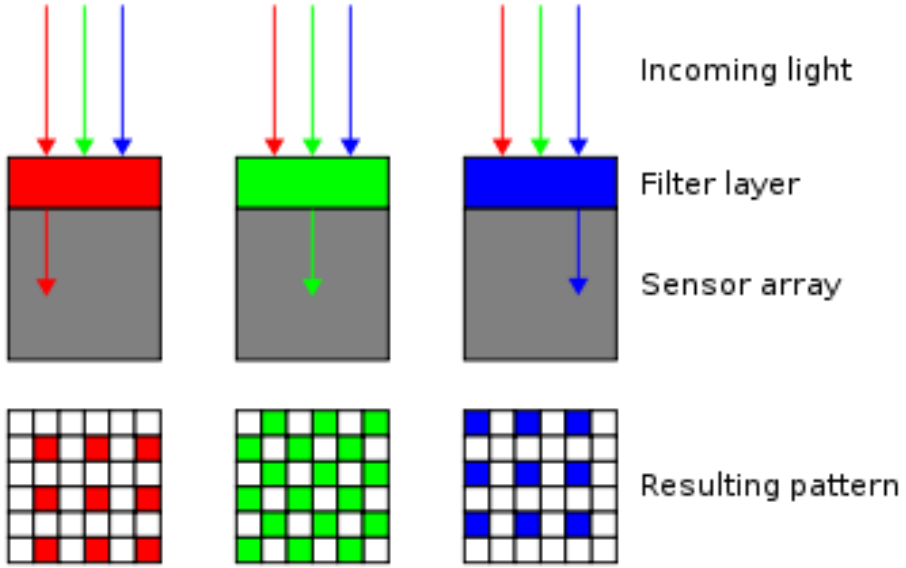


그림 2. 이미지 센서의 필터 구조

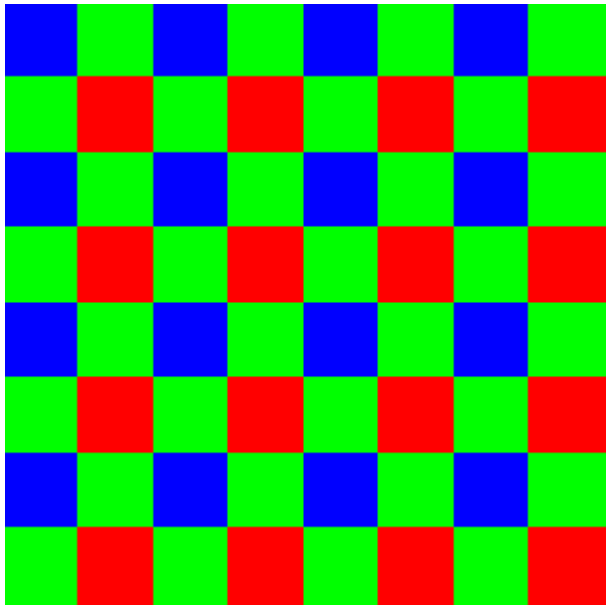


그림 3. 베이어 필터 패턴

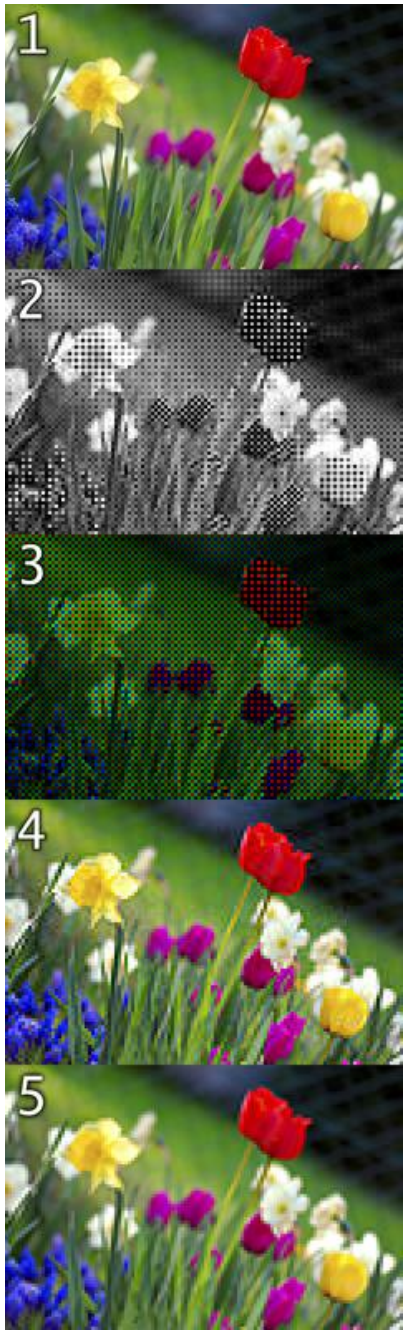


그림 1.
1. 원본 장면
2. 베이어 필터가 있는 120 × 80 픽셀 센서의 출력
3. Bayer 필터 색상으로 색상으로 구분 된 출력
4. 누락 된 색상 정보를 보간 한 후 재구성 된 이미지
5. 비교를 위해 120 × 80 픽셀의 풀 RGB 버전

출처

자율 주행 자동차를 위한 “영상” 센서

4. 조리개와 focal

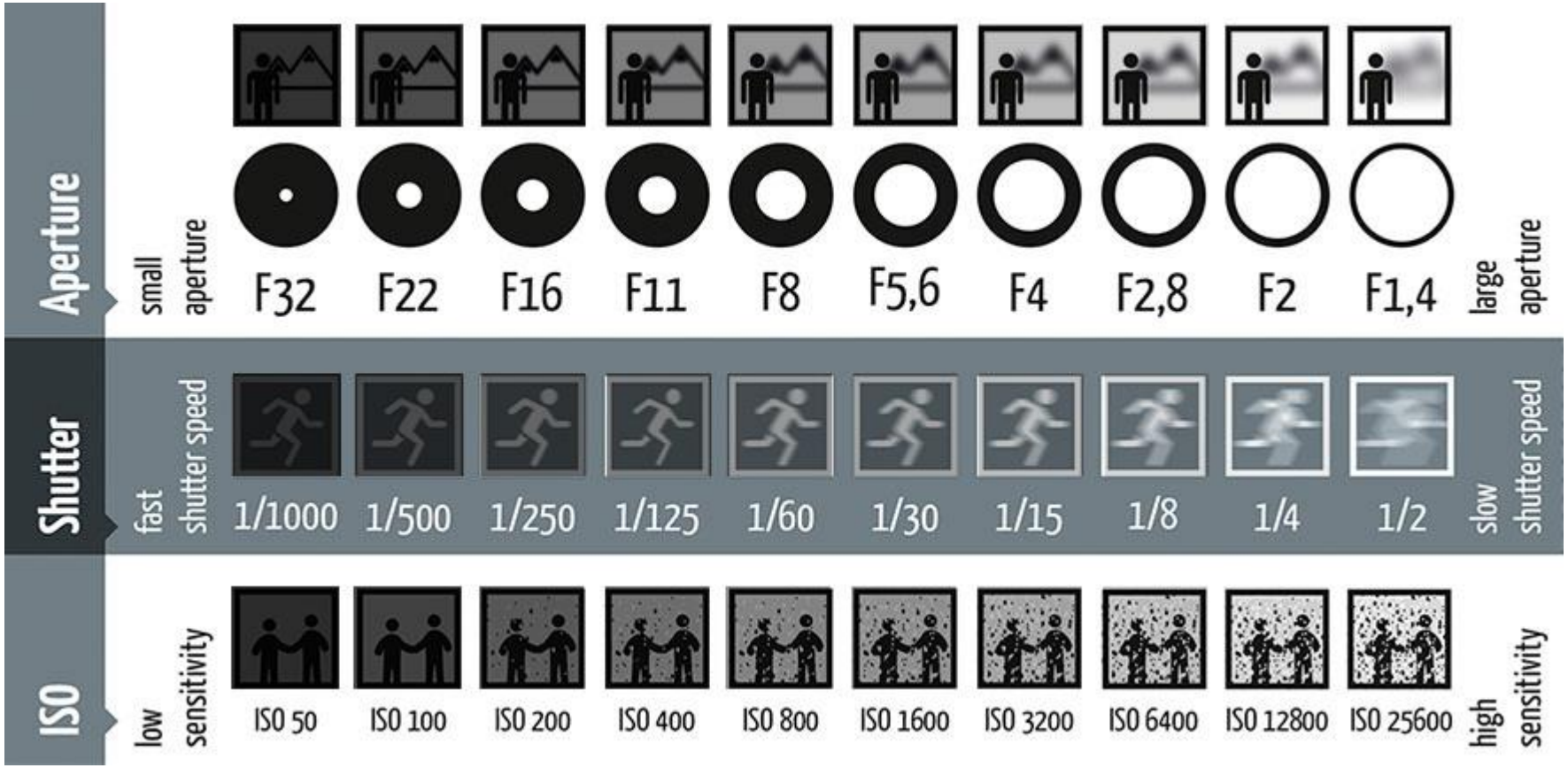


그림. 조리개, Shutter speed, ISO

자율 주행 자동차를 위한 “영상” 센서

4. 조리개와 focal

렌즈의 종류 (DSLR 기준)

- 광각 렌즈 : 15 ~ 40mm
- 표준 렌즈 : 40 ~ 70mm
- 망원 렌즈 : 40mm ~

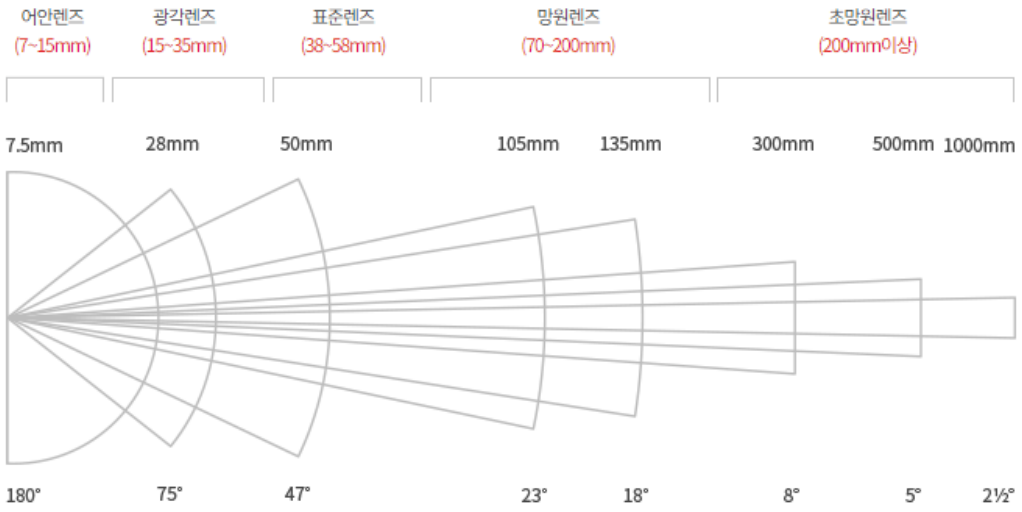


그림 2. 초점 거리와 화각

그림 1. 초점 거리 [출처]

자율 주행 자동차를 위한 “영상” 센서

4. 조리개와 focal

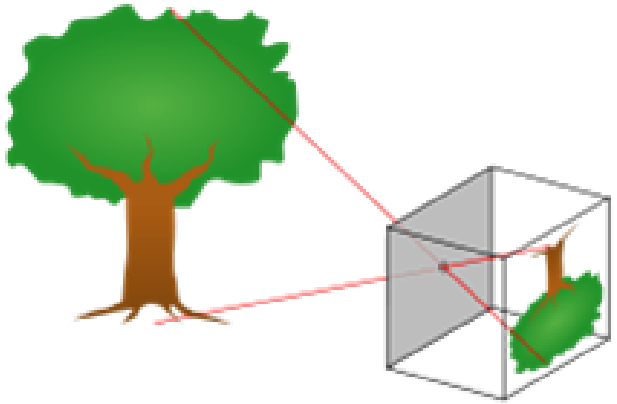


그림 1. Pinhole 카메라 모델

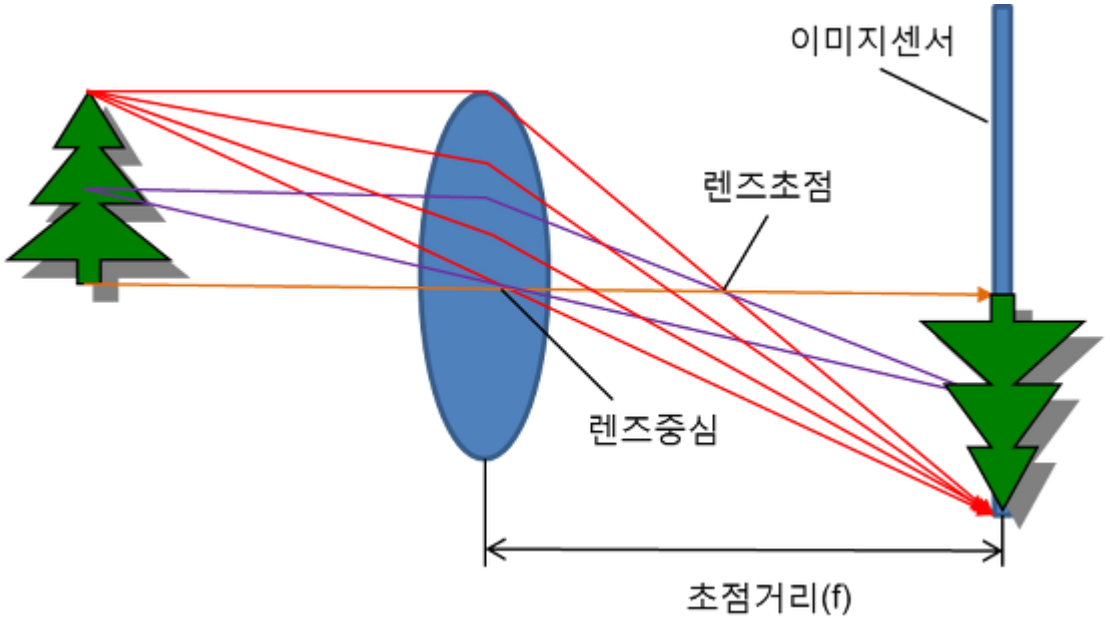


그림 2. 렌즈-이미지 센서 투영

자율 주행 자동차를 위한 "영상" 센서

2. 카메라 인터페이스

IEEE1394 **FireWire 1394.b**	**Camera Link®**	**USB 2.0**	**USB 3.0**	**GigE**
				
800 Mb/s	최대 3.6 Gb/s	480 Mb/s	5Gb/s	1000 Mb/s
100m (GOF 케이블 사용)	10m	5m	3m (recommended)	100m
최대 63대	1	최대 127대	최대 127대	무제한
9핀-9핀	26핀	USB	USB	RJ45/CAT5

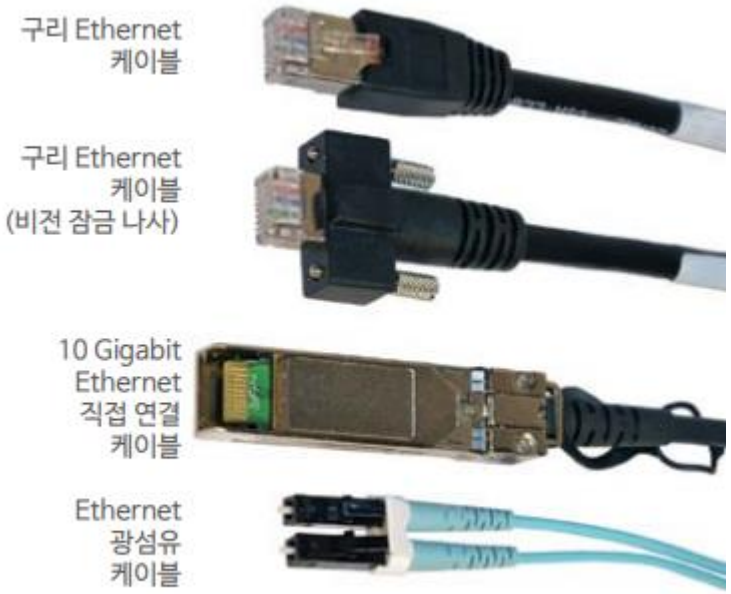
출처

자율 주행 자동차를 위한 “영상” 센서

CoaXPress®

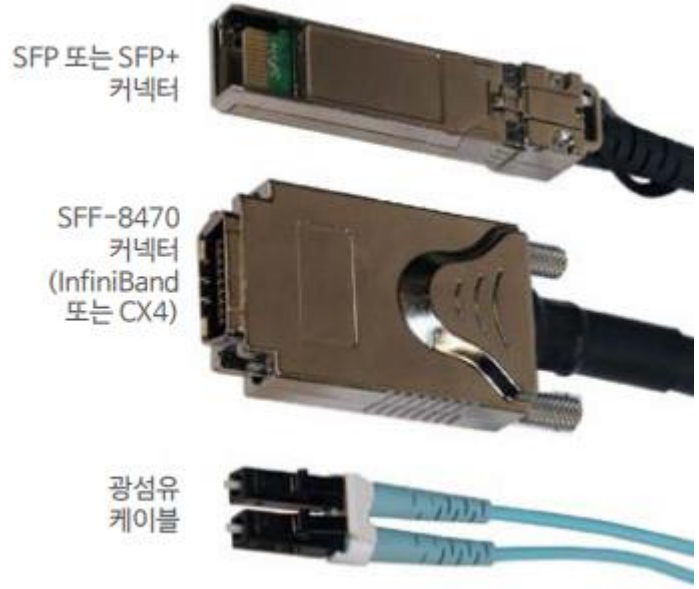
USB® VISION

GiGE® VISION



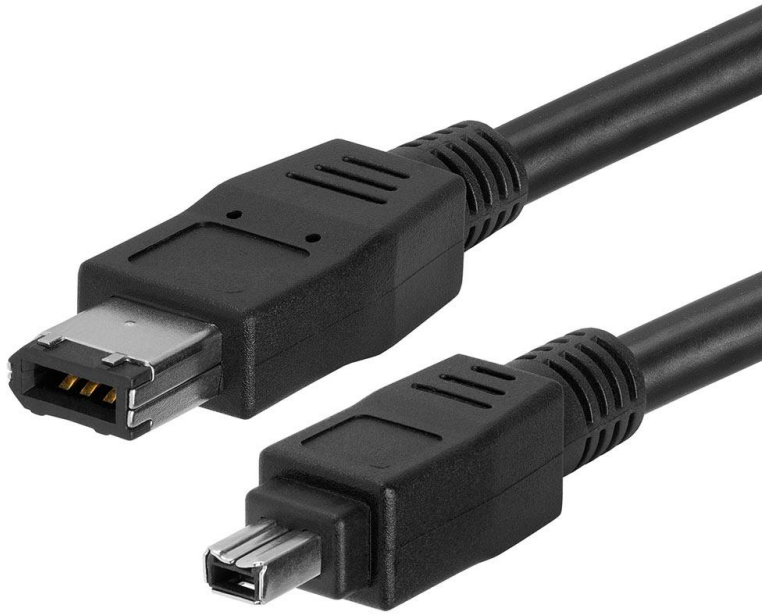
자율 주행 자동차를 위한 “영상” 센서

CAMERA LinkHS™



CAMERA Link®

FireWire, IEEE 1394, i.Link



Multiple Camera

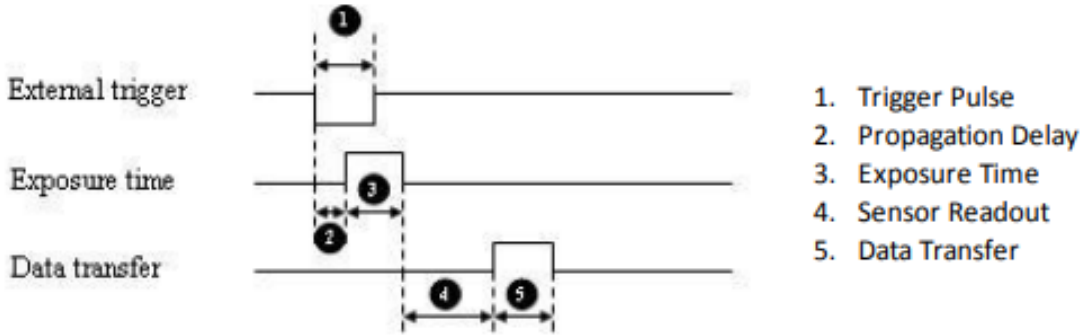
1. 동기화(Synchronization)

동기화란? 동시에 시스템을 작동시키기 위해 사건을 일치 시키는 것!

1-1. S/W Synchronization : 시간 이용하여 사건을 일치 시킴

1-2 H/W Synchronization : 전기적 신호를 이용하여 사건을 일치 시킴

Trigger : 신호 왔으니 찍어!
Strobe : 나 찍었으니 작동해(신호 전송)



1. Trigger Pulse
2. Propagation Delay
3. Exposure Time
4. Sensor Readout
5. Data Transfer

Figure 7.9: External trigger timing characteristics

출처 : Flycapture

General Purpose I/O Connector

The camera has an 8-pin GPIO connector on the back of the case; refer to the diagram for wire color-coding.

Diagram	Color	Pin	Function	Description
	Black	1	I0	Opto-isolated input (default Trigger in)
	White	2	O1	Opto-isolated output
	Red	3	IO2	Input/Output/serial transmit (TX)
	Green	4	IO3	Input/Output/serial receive (RX)
	Brown	5	GND	Ground for bi-directional IO, V_{EXT}, +3.3 V pins
	Blue	6	OPTO_GND	Ground for opto-isolated IO pins
	Orange	7	V_{EXT}	Allows the camera to be powered externally
	Yellow	8	+3.3 V	Power external circuitry up to 150 mA
	To configure the GPIO pins, consult the General Purpose Input/Output section of your camera's Technical Reference Manual.			

그림 1.GPIO (General Purpose I/O Connector)

문제

아래를 보고 카메라 정보를 적으십시오. Ex) Img sensor, interface, 동기화, Shutter 타입, 화각 등

- 2 × PointGrey Flea3 color camera (FL3-GE-13S2C 1288 × 964, 1.3MP, Sony ICX445) GigE 84.9(H) × 68.9(V) with Spacecom HF3.5M-2 Lens 3.5mm
- 2 × FLIR A35 thermal camera (320×256, 7.5~13um) GigE 63(H) × 50(V) with 7.5mm

그림. R2T2 paper에 기재된 카메라 정보

Model Number:	FL3-GE-13S2C-C	**Camera Sensor Format:**	1/3"
Resolution (MegaPixels):	1.2	**Pixels (H x V):**	1,288 x 964
Type:	Color Camera	**Camera Control:**	GigE Vision
Camera Family:	Flea®3	**Connector:**	GigE interface with Screw Locks
Dimensions (mm):	29 x 29 x 30	**Electronic Shutter:**	0.03ms - 32s (Extended Shutter Mode)
Frame Rate (fps):	31	**Image Format:**	Y8, Y16, Mono8, Mono12, Mono16, RGB, YUV411, YUV422, YUV 444, Raw8, Raw12, Raw16
Imaging Device:	Sony ICX445	**Manufacturer:**	FLIR
Mount:	C-Mount	**Note:**	Dimensions exclude ¼-20 Tripod Adapter and Lens Holder
Operating Temperature (°C):	0 to +45	**Pixel Depth:**	8, 12, 16 and 24-bit Digital Data
Pixel Size, H x V (µm):	3.75 x 3.75	**Storage Temperature (°C):**	-30 to +60
Synchronization:	External or Via Software	**Transfer Speed (Mbit/s):**	10/100/1000
Type of Sensor:	Progressive Scan CCD	**Type of Shutter:**	Global
Video Output:	GigE	**Weight (g):**	58.00

표. R2T2 데이터셋에서 사용한 카메라의 Specifications

Embedded, Edge Computer란?

1. Embedded?

- 시스템에 "내장된" 특정 목적의 컴퓨터
- 특정되지 않은 일반적인 목적을 수행하는 개인용 컴퓨터와 대조적

- 근데 Jetson은 컴퓨터처럼 특정되지 않은 목적으로 사용이 가능하네?
  - 우리가 사용하는 그림 1.은 그림 2. 형태의 임베디드 보드에 개인용 컴퓨터로 사용이 가능하도록 기타 I/O, 전원, 네트워크 모듈이 부착이 된 상태이기에, 임베디드 시스템이 아니라 미니 컴퓨터라고 부르는 것이 정확하다.

그림 1. Jetson Nano. Developer Kit

그림 2. Jetson Nano. Module

Nvidia Jetson

1. 리눅스 (Ubuntu 18.04) 개발 환경 – Ubuntu에 대한 간략한 설명

리눅스의 구조

- 커널 : 리눅스의 핵심으로 프로세스 관리, 메모리 관리, 파일 시스템 관리, 장치 관리 등 컴퓨터의 모든 자원을 제어한다.
- 셸 : 사용자와 커널 사이의 중간자 역할을 한다. 사용자가 입력한 명령을 해석하여 커널에 넘겨주고 결과를 받아 화면에 출력한다.
- 응용 프로그램 : 각종 프로그래밍 개발 도구, 문서 편집 도구, 네트워크 관련 도구 등 매우 다양한 응용 프로그램을 제공한다.

리눅스의 특징

- 리눅스는 공개 소프트웨어이며, 무료로 사용 가능하다. (사용자간 피드백과 참여로 빠르게 패치)
- 다중 사용자 및 다중 처리 시스템
- 뛰어난 이식성 (거의 모든 하드웨어를 지원한다.)

Nvidia Jetson

1. 리눅스 (Ubuntu 18.04) 개발 환경 – Ubuntu에 대한 간략한 설명

리눅스 디렉토리 구조 : 트리 형태

/root
- 최상위 디렉토리

/bin
- 시스템을 사용하기 위한 기본 명령어 포함 ex) cp, mv.....

/boot
- 부팅에 필요한 커널 및 관련 파일 위치

/dev
- 장치 파일 위치

/etc
- 리눅스에서 사용되는 응용 프로그램과 서버 프로그램이 환경 설정에 필요한 설정 파일 위치

/home
- 일반 사용자의 홈 디렉토리가 생성되는 곳

/lib
- 시스템 운영 및 프로그램 구동할 때 필요한 공유 라이브러리 위치
- 커널 모듈

/sys
- 리눅스 커널과 관련된 파일이 있는 디렉토리이다

Nvidia Jetson

1. 리눅스 (Ubuntu 18.04) 개발 환경 – Ubuntu에 대한 간략한 설명

리눅스 디렉토리 구조

/proc
- 가상 메모리, 프로세스와 시스템 정보를 제공하기 위한 목적으로 설계된 가상 파일 시스템을 사용하는 디렉토리

/tmp
- 프로세스 생성 과정에서 발생되는 임시 파일 저장
- 시스템이 재 구동하는 경우에는 존재하는 모든 파일 삭제

/usr
- 기본 실행 파일과 라이브러리 파일, 헤더 파일 등 (usr = Unix System Resource)

/opt
- 추가 패키지가 설치되는 디렉토리

/var
- 내용이 자주 변경되는 가변 자료 저장 ex) log

/media
- USB 같은 외부 장치를 연결하는 디렉토리

/run
- 실행 중인 서비스와 관련된 파일이 저장된다.

Nvidia Jetson

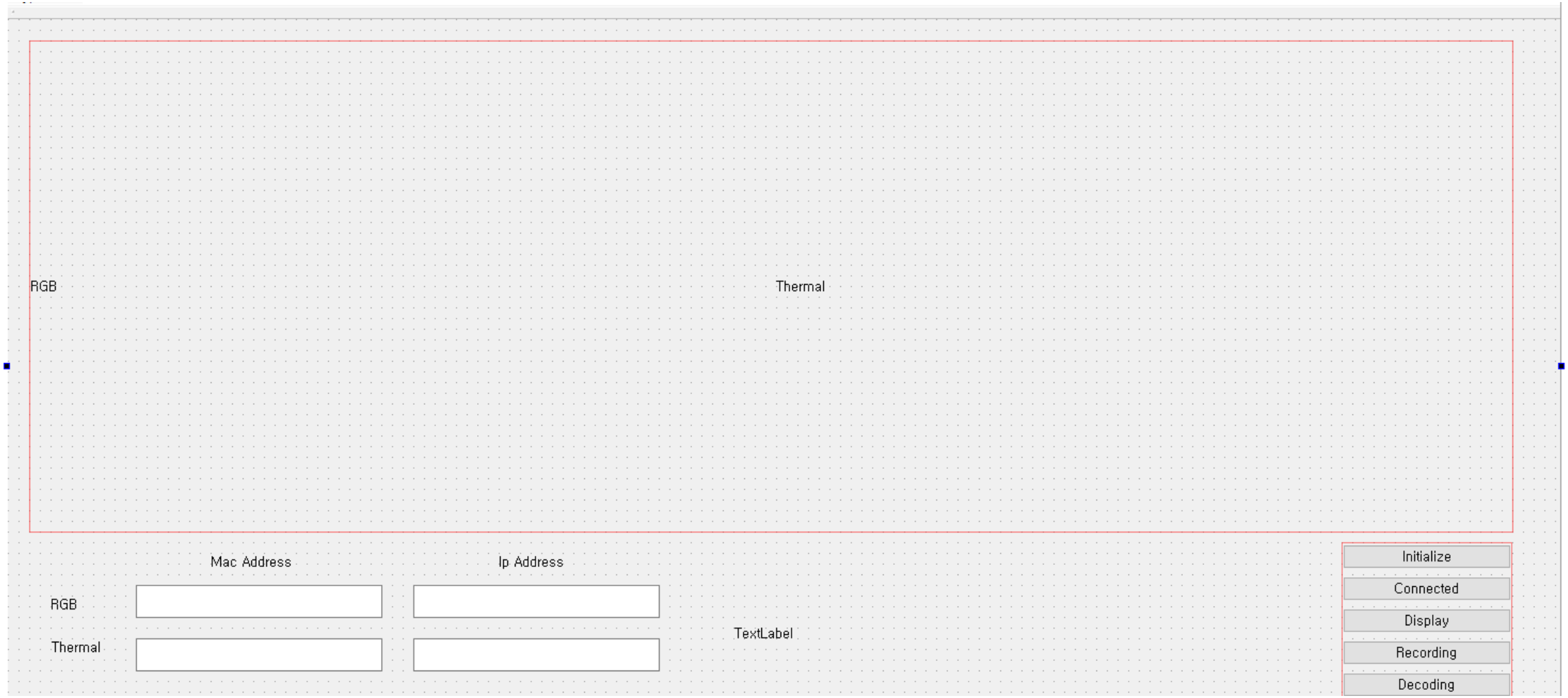
2. QT, GUI 프로그래밍

GUI(Graphic User Interface)

- Component : label, button, check box.....
- Container : Window, Panel, frame.... 등 component를 담는 용기
- Layout manager : 배치 관리자
- Graphics : 이미지, 폰트 등을 componen나 container 추가하는 것
- Event : 특정 사건(버튼 클릭, 마우스 움직임 등) 일어날 때, 어떤 동작이 일어날지 정하는 것

Nvidia Jetson

2. QT, GUI 프로그래밍



1. Pytorch 설치, lib과 include

- https://elinux.org/Jetson_Zoo
- Requirement
- $ pip3 install Cython
- Pytorch-1.1.0 install
- $ sudo pip3 install torch-1.1.0-cp36-cp36m-linux_aarch64.whl
- torchvision - v0.3.0 install
- $ sudo apt-get install libjpeg-dev zlib1g-dev
- $ git clone --branch v0.3.0 https://github.com/pytorch/vision torchvision
- $ cd torchvision
- $ sudo python setup.py install

PyTorch (Caffe2)

- Website: https://pytorch.org/
- Source: https://github.com/pytorch/pytorch
- Version: PyTorch v1.0.0 - v1.4.0
- Packages:

	Python 2.7	Python 3.6
v1.0.0	pip wheel	pip wheel
v1.1.0	pip wheel	pip wheel
v1.2.0	pip wheel	pip wheel
v1.3.0	pip wheel	pip wheel
v1.4.0	pip wheel	pip wheel

1. Pytorch 설치, lib과 include

- Library
  - 공통으로 사용될 수 있는 특정 기능들을 '모듈화' 한 것
  - 컴파일되어 기계어 형태로 존재함
  - *.lib / *.a : 윈도우용 / 리눅스용, 정적 라이브러리
  - *.dll / *.so : 윈도우용 / 리눅스용, 동적 라이브러리
  - 정적, 동적 라이브러리 차이
    - 컴파일시 라이브러리 내용이 실행 파일에 포함되면 정적! NEEDED 따로 라이브러리에서 호출하면 동적!
- 헤더 파일 (include)
  - 기계어 형태의 라이브러리를 사용자/컴파일러가 이해 할 수 있도록 문법에 맞게 작성되어 있는 선언들의 집합

Qmake와 .pro

```
QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = Embedded_Grabber
TEMPLATE = app

TARGET    = QtCuda
CONFIG += c++11 console
CONFIG -= app_bundle

DEFINES += QT_DEPRECATED_WARNINGS

SOURCES += \
        main.cpp \
        mainwindow.cpp \
        Camthread.cpp \
        filethread.cpp \
        afmodel.cpp

HEADERS += \
        mainwindow.h \
        camthread.h \
        filethread.h \
        afmodel.h

FORMS += \
        mainwindow.ui
```

```
INCLUDEPATH += $$PWD/../../../../../opt/pleora/ebus_sdk/linux-aarch64-arm/include
DEPENDPATH += $$PWD/../../../../../opt/pleora/ebus_sdk/linux-aarch64-arm/include

LIBS += \
        -L/opt/pleora/ebus_sdk/linux-aarch64-arm/lib -lEbTransportLayerLib \
        -lEbUtilsLib -lPtConvertersLib -lPtUtilsLib -lPvAppUtils \
        -lPvBase -lPvBuffer -lPvCameraBridge -lPvCoreGEV -lPvDevice -lPvGUI \
        -lPvGenICam -lPvPersistence -lPvSerial -lPvStream \
        -lPvSystem -lPvTransmitter -lPvVirtualDevice -lSimpleImagingLib
```

2. Qmake와 .pro

- qmake는 Makefile generator
- Qt의 프로젝트 파일 .pro에 qmake 내용을 작성합니다.
- 시스템 변수 (qmake)
  - TENPLATE : 프로젝트 타입 정의 ex) TEMPLATE = app
  - HEADERS : 헤더 파일 작성 ex) HEADERS += camthread.h mainwindows.h
  - SOURCES : 소스 파일 작성 ex) SOURCES += CamThread.cpp MainWindows.cpp
  - TARGET : qmake로 생성될 실행 파일 이름, default로 .pro 이름으로 작성된다
  - LIBS : 프로젝트에 링크할 라이브러리 지정한다 ex) LIBS += -L$${LIB_PATH} –lopencv
  - INCLUDEPATH : 전역 헤더 파일의 위치를 찾기 위한 경로 지정
    - Ex) INCLUDEPATH += -I$${INCLUDE_PATH} ₩ -I$${INCLUDE_PATH2}
  - FORMS : Qt 디자이너로 생성된 .ui 파일을 지정한다.
  - QT : 프로젝트에 사용할 QT 모듈을 지정한다. Ex) QT += core gui

2. Qmake와 .pro

- qmake는 Makefile generator
- Qt의 프로젝트 파일 .pro에 qmake 내용을 작성합니다.
- 시스템 변수 (qmake)
  - TENPLATE : 프로젝트 타입 정의 ex) TEMPLATE = app
  - HEADERS : 헤더 파일 작성 ex) HEADERS += camthread.h mainwindows.h
  - SOURCES : 소스 파일 작성 ex) SOURCES += CamThread.cpp MainWindows.cpp
  - TARGET : qmake로 생성될 실행 파일 이름, default로 .pro 이름으로 작성된다
  - LIBS : 프로젝트에 링크할 라이브러리 지정한다 ex) LIBS += -L$${LIB_PATH} –lopencv
  - INCLUDEPATH : 전역 헤더 파일의 위치를 찾기 위한 경로 지정
    - Ex) INCLUDEPATH += -I$${INCLUDE_PATH} ₩ -I$${INCLUDE_PATH2}
  - FORMS : Qt 디자이너로 생성된 .ui 파일을 지정한다.
  - QT : 프로젝트에 사용할 QT 모듈을 지정한다. Ex) QT += core gui

3. 버튼을 이용한 Camera Connect

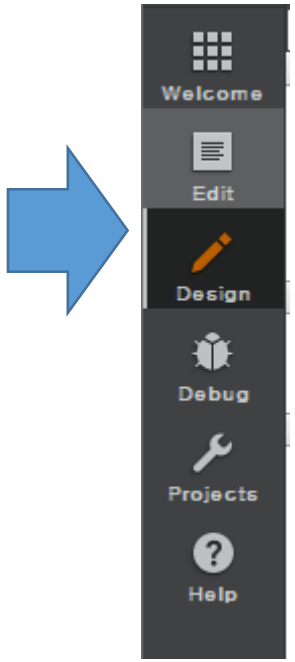


그림 1. Qt Design

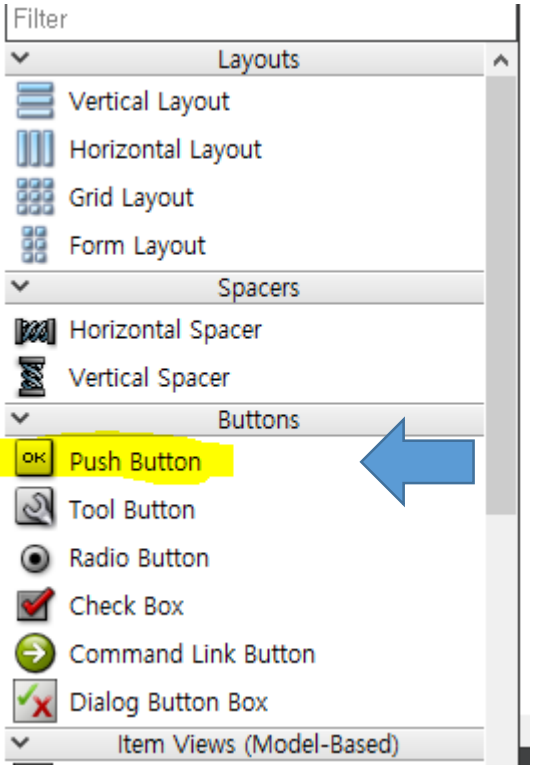


그림 2. Connect 기능을 이벤트로 처리하기 위한 버튼

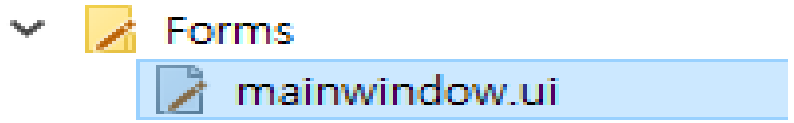


그림 4. GUI 정보가 정의된 파일

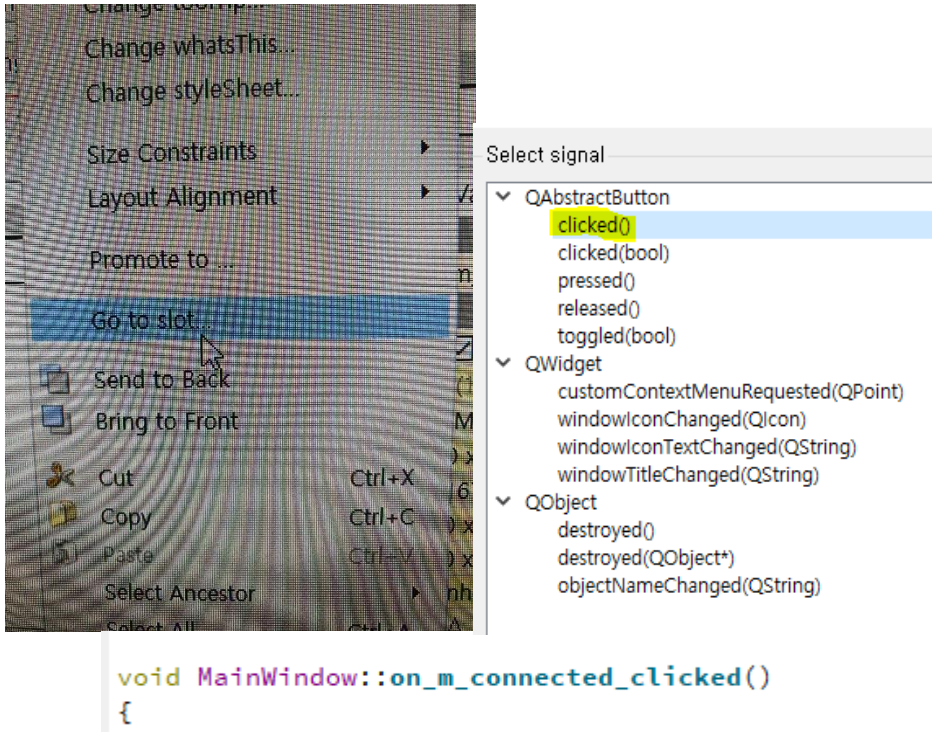


그림 3. Go to slot (Push Button 오르쪽 클릭) Button, Widget간 사용자간 액션을 Go to slot으로 지정 시, 액션에 대한 행위가 emit – Connect – Go to slot으로 만들어진 함수로 들어가집니다. 위의 일련 과정들은 Go to Slot으로 자동으로 작성되어지기 때문에 함수를 똑같이 작성해도 작동하지 않습니다.

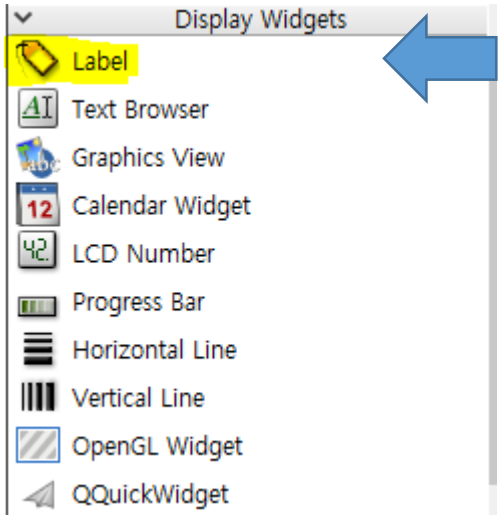


그림 5. 디스플레이에 사용할 Label

3. 버튼을 이용한 Camera Connect

3-1. initialize

```
void MainWindow::on_m_initialize_clicked()
{
    //Run Shell Script
    ShellScript();

    //Camera Ip congigure
    CamInitialize( mThermalMac[0].toStdString().c_str(), mThermalIp[0].toStdString().c_str() );
    CamInitialize( mRGBMac[0].toStdString().c_str(), mRGBIp[0].toStdString().c_str() );

    CamInitialize( mThermalMac[1].toStdString().c_str(), mThermalIp[1].toStdString().c_str() );
    CamInitialize( mRGBMac[1].toStdString().c_str(), mRGBIp[1].toStdString().c_str() );

}
```

3. 버튼을 이용한 Camera Connect

3-1. initialize

```
void MainWindow::ShellScript()
{
    if (system("echo 'v1s10nlab2019' |"
               " sudo -S /opt/pleora/ebus_sdk/linux-aarch64-arm/bin/set_rp_filter.sh"
               " --mode=0 --restartnetworkstack=yes"))
    {
        cout << "set_rp_filter failed" << endl;
    }

    if (system("echo 'v1s10nlab2019' | "
               "sudo -S /opt/pleora/ebus_sdk/linux-aarch64-arm/bin/set_socket_buffer_size.sh"))
    {
        cout << "set_soket_buffer failed" << endl;
    }
    else cout << "set_soket_buffer" << endl;

    if (system("echo 'v1s10nlab2019' | "
               "sudo -S /opt/pleora/ebus_sdk/linux-aarch64-arm/bin/set_usbfs_memory_size.sh"))
    {
        cout << "set_usbfs_memory_size failed" << endl;
    }
    else cout << "set_usbfs_memory_size" << endl;

    system( "echo 'v1s10nlab2019' | sudo -S service eBUSd stop" );

    if (system( "echo 'v1s10nlab2019' | sudo -S service eBUSd start" ))
    {
        cout << "UnValid license" << endl;
    }
    else cout << "Valid license" << endl;
}
```

- set_rp_filter.sh (rp=Reverser Path) : Reverser Path Filtering 값이 기본 1로 설정된 값을 0으로 변경하여, 통신을 위하여 filtering 기능을 꺼둠
- set_soket_buffer_size : 10485760 로 사이즈 증가
- set_usbf_memory_size : usb 통신 제한을 4000으로 확장
- service eBUSd stop/start : eBUS 라이센스 활성화, 작동시 상시 활성화 해줘야함

3. 버튼을 이용한 Camera Connect

3-1. initialize

카메라 IP 설정, CamInitialize

```
void MainWindow::CamInitialize(const char* Mac, const char* IpAdr)
{
    PvDeviceGEV* lDeviceSetIp = NULL;
    PvResult lResult;

    lResult = lDeviceSetIp->SetIPConfiguration( Mac, IpAdr, "255.255.0.0");

    if (!lResult.IsOK()) cout << lResult.GetCodeString().GetAscii() << endl;

}
```

3. 버튼을 이용한 Camera Connect

3-2. Connect

```
lDevice = PvDevice::CreateAndConnect( aConnectionID, &lResult );


lStream = PvStream::CreateAndOpen( aConnectionID, &lResult );

mPipeline = new PvPipeline(aStream);

mPipeline->SetBufferSize(static_cast<uint32_t>(lSize));
mPipeline->SetBufferCount( 16 );
lResult = mPipeline->Start();
```

Grabber 제작

1. Thread, Thread 프로그래밍?

Thread : 프로그램 실행의 단위이며, 하나의 프로세스는
여러 개의 쓰레드로 구성되어 있다.

여러 개의 쓰레드를 구성하고 있는 경우를 멀티 쓰레드라고 하며,
동시에 여러 흐름을 진행할 수 있다.

멀티 쓰레드는 프로세스 내의 메모리를 공유해 사용할 수 있으며,
스레드 간 전환 속도가 빠르다.

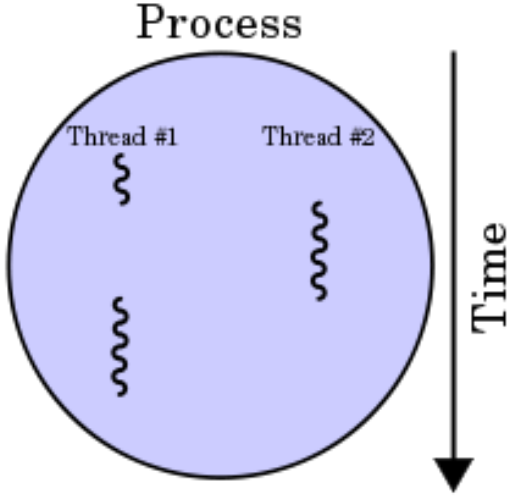
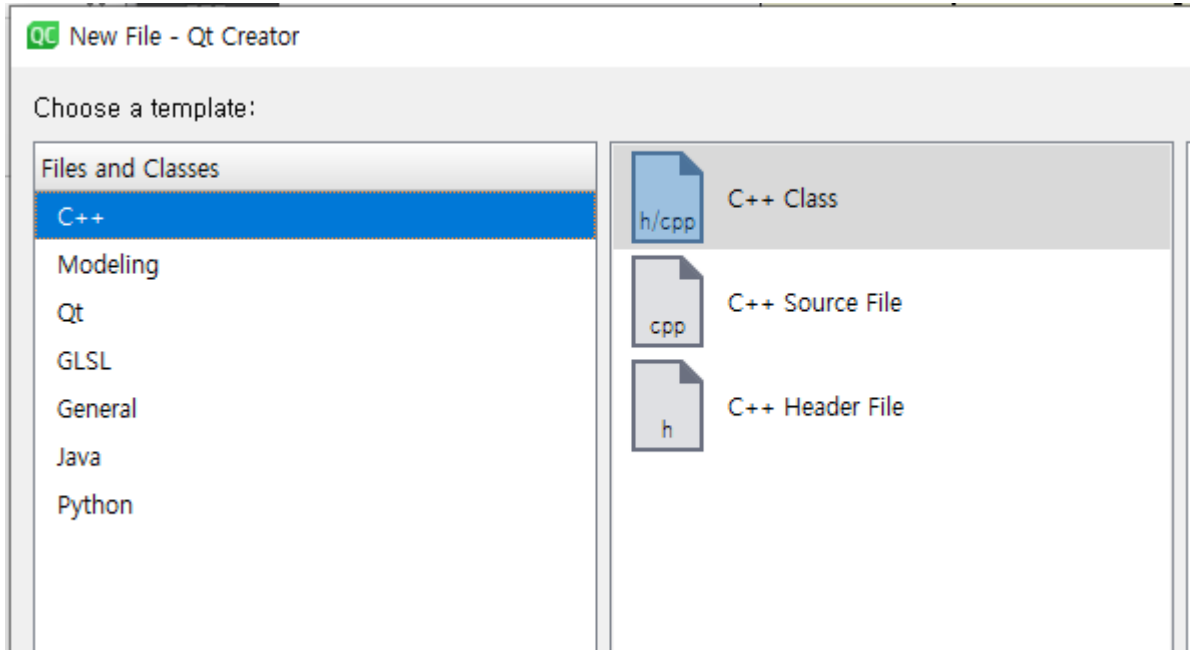


그림 1. 두 개의 스레드를 실행하고 있는 하나의 프로세스

Grabber 제작

1. Thread, Thread 프로그래밍?

해당 실습에서는 Qt 모듈에 있는 Qthread를 사용하여 Thread 프로그래밍 연습해본다.



(1) C++ class를 생성해준다.

생성 시, Qt 모듈 중 QObject를 물려 생성해주시기 바랍니다.

Grabber 제작

1. Thread, Thread 프로그래밍?

해당 실습에서는 Qt 모듈에 있는 Qthread를 사용하여 Thread 프로그래밍 연습해본다.

```
#include <QObject>
#include <QThread>
#include <QMutex>

class FileThread : public QThread
{
    Q_OBJECT
public:
    explicit FileThread(QObject *parent = nullptr);

    void run();
```

(1) C++ class 생성해준다.
(2) Header 파일에 Qthread 모듈을 사용하기 위한 세팅을 해준다.
(3) Thread 활성화

Qthread에서는 void run() 함수가 main()문처럼 작동한다.

MainWindow

```
mCamera[0] = new CamThread(this);

mCamera[0]->start();
```

쓰레드 생성

부모 쓰레드에서 생성한 쓰레드 클래쓰 start() 해주면 void run();이 실행되면서 활성화 됩니다.

Grabber 제작

1. Thread, Thread 프로그래밍?

해당 실습에서는 Qt 모듈에 있는 Qthread를 사용하여 Thread 프로그래밍 연습해본다.

(1) C++ class 생성해준다.
(2) Header 파일에 Qthread 모듈을 사용하기 위한 세팅을 해준다.
(3) Thread 활성화
(4) Thread 종료

MainWindow

```
mCamera[0]->start();

mCamera[0]->quit();
mCamera[0]->wait();
```

- 만약에 쓰레드 안에 루프 문이 존재 한다면, 루프를 멈춥니다.
- quit() 쓰레드를 종료 시킵니다.
- wait() 쓰레드를 대기 시킵니다.

Grabber 제작

2. Image Acquisition 제작

```
// Get device parametets need to control streaming
PvGenParmeterArray* lDeviceParams = Device->GetParameters();
// ////////////////////Thermal Setting
// lDeviceParams->SetEnumValue("SyncMode", 1);
// lDeviceParams->SetEnumValue("NUCMode", 0);
// ////////////////////////////////////////////
// ////////////////////RGB Setting
// lDeviceParams->SetEnumValue("ExposureAuto",0);
// lDeviceParams->SetFloatValue("ExposureTime",1050);
// lDeviceParams->SetEnumValue("TriggerMode",1);
// lDeviceParams->SetEnumValue("TriggerSource",3);
// lDeviceParams->SetIntegerValue("OffsetX",0);
// lDeviceParams->SetIntegerValue("OffsetY",0);
// lDeviceParams->SetIntegerValue("Width",1288);
// lDeviceParams->SetIntegerValue("Height",964);
// ////////////////////////////////////////////

// Map the GenICam Acquisition Start and Acquisition Stop c
PvGenCommand *lStop = dynamic_cast<PvGenCommand *>
// Enable streaming and send the AcquisitionStart command
cout << "Enabling streaming and sending AcquisitionStart co
Device->StreamEnable();
lStart->Execute();
```

```
if (Stream->IsOpen() && Pipeline->IsStarted())
{
Result = Pipeline->RetrieveNextBuffer( &Buffer, 0xFFFFFFFF, &Resul
t);
if( !Result.IsOK() ) continue;}
```

Grabber 제작

2. Image Acquisition

```
if (Stream->IsOpen() && Pipeline->IsStarted())
{
Result = Pipeline->RetrieveNextBuffer( &Buffer, 0xFFFFFFFF, &Result);
if( !Result.IsOK() ) continue;
}
```

Recording

2. Image Acquisition

```
FILE* fileStream;

fileStream = fopen( “Path”, “wb” );

fwrite( data, elementSize, data size, fileStream);
```

현재 프로젝트에서는 원본을 유지하기 위하여, bin 파일에 저장하고 있다.

Decoding

```
fileStream = fopen( “Path”, “rb”);

fread( data, elementSize, dataSize, fileStream );

cv::imwrite ( “img Path”, data );
```

문제

Display 화면을 On/Off 할 수 있는 버튼을 만들고 구현 하십시오.