

쿼드콥터 개발

PM : 이상훈 강사님

황선우 백영민 신영용 이성용 목현호

문제점 황선우

2016.5~ 2016.6.1 Vivado design suite 사용법 서적 없고 온라인상에 자료 부족
->Xilinx 공식 메뉴얼로 사용법 파악.

2016.5~ 2016.6.1 필터 로직 설계를 위한 VHDL 학습

Dropbox ▸ xilinx ▸				
이브러리에 포함 ▾	공유 대상 ▾	급기	새 폴더	
이름	수정된 날짜	유형	크기	
gev	2016-04-18 오후...	파일 폴더		
gvrd-ac701-tpg_Vivado2015.4	2016-04-18 오후...	파일 폴더		
12-Visser.pdf	2016-05-02 오후...	Adobe Acrobat D...	700KB	
AM2302.pdf	2016-07-02 오후...	Adobe Acrobat D...	597KB	
C_Users_Bitcamp_Documents_XilinxDo...	2016-06-03 오후...	Adobe Acrobat D...	2,258KB	
C_Users_Bitcamp_Documents_XilinxDo...	2016-06-03 오후...	Adobe Acrobat D...	8,249KB	
C_Users_Bitcamp_Documents_XilinxDo...	2016-06-03 오후...	Adobe Acrobat D...	2,886KB	
C_Users_Bitcamp_Documents_XilinxDo...	2016-06-03 오후...	Adobe Acrobat D...	2,283KB	
DDI0407F_cortex_a9_r2p2_mpcore_trm...	2016-06-30 오전...	Adobe Acrobat D...	1,127KB	
HCSR04.pdf	2016-06-29 오후...	Adobe Acrobat D...	79KB	
HC-SR04-datasheet-version-2.pdf	2016-06-30 오전...	Adobe Acrobat D...	526KB	
PicoZed 7015_7030_User's_Guide_v1.4...	2016-03-22 오전...	Adobe Acrobat D...	1,498KB	
PicoZed_Petalinux_2015_2_eMMC_boot...	2016-06-13 오후...	Adobe Acrobat D...	2,933KB	
SVDK_1.2_Schematic.pdf	2015-06-02 오후...	Adobe Acrobat D...	8,485KB	
SVDK_Hardware_UserGuide.pdf	2016-06-01 오전...	Adobe Acrobat D...	679KB	
sweden hls opencv.pdf	2016-05-19 오전...	Adobe Acrobat D...	3,379KB	
The_Zynq_Book_ebook.pdf	2016-04-29 오후...	Adobe Acrobat D...	25,338KB	
The_Zynq_Book_Tutorials_Aug_15.pdf	2016-04-29 오후...	Adobe Acrobat D...	11,931KB	
ug585-Zynq-7000-TRM.pdf	2016-06-21 오후...	Adobe Acrobat D...	16,572KB	
ug871-vivado-high-level-synthesis-tutor...	2016-04-28 오전...	Adobe Acrobat D...	12,364KB	
ug888-vivado-design-flows-overview-tu...	2016-03-31 오전...	Adobe Acrobat D...	2,486KB	
ug896-vivado-ip.pdf	2016-05-19 오후...	Adobe Acrobat D...	5,590KB	
ug899-vivado-io-clock-planning.pdf	2016-05-17 오후...	Adobe Acrobat D...	2,381KB	
ug902-vivado-high-level-synthesis.pdf	2016-06-21 오후...	Adobe Acrobat D...	8,333KB	
ug908-vivado-programming-debugging...	2016-04-11 오후...	Adobe Acrobat D...	5,599KB	
ug939-vivado-designing-with-ip-tutoria...	2016-05-17 오후...	Adobe Acrobat D...	2,378KB	
ug998-vivado-intro-fpga-design-hls.pdf	2016-05-30 오전...	Adobe Acrobat D...	2,055KB	
ug1037-vivado-axi-reference-guide.pdf	2016-05-17 오후...	Adobe Acrobat D...	3,048KB	
vivado-ide user_guide.pdf	2016-03-30 오후...	Adobe Acrobat D...	5,816KB	
xapp1168-axi-ip-integrator.pdf	2016-05-17 오후...	Adobe Acrobat D...	1,952KB	
xilinx hls zynq opencv.pdf	2016-05-27 오전...	Adobe Acrobat D...	317KB	
xilinx korea.pdf	2016-05-19 오전...	Adobe Acrobat D...	3,872KB	
한글 zynq 설명서.pdf	2016-06-29 오전...	Adobe Acrobat D...	5,963KB	

영상처리

문제점

2016.6.1-6.6 물체 추적을 위해서 오차보정을 위한 칼만 필터, 트래킹 알고리즘

(**cam, mean shift**, 루카스카나데) 공부하였으나, 해당 트래킹 알고리즘은 **ROI**가 필요함. 그러나 우리는 입력도구 연결이 불가능하고 무선으로 단방향 통신 할 것이므로 해당 알고리즘 적용불가.

->**object detection**으로 선회

2016.6.7 특정 물체를 따라가기 위해서는 학습 시켜야 함(**adaboost**와 **cascade**)

->**harr**관련 논문들을 참조하여 특정물체 학습시키고 결과 **xml**로 출력

2016.6.9 opencv harr detecmultiscale함수 내부 드라이빙 중 영상처리 알고리즘 이해부족으로 난항.

->적분영상을 만드는 방법, 쌓선형 보간법, **false positive**와 **false negative**에 따라 **threshold** 설정 방법 검색, **scanning**

2016.6.10 opencv 내부분석 마무리. 분석결과 내부를 모두 **FPGA**로 구현하는 것은 퍼포먼스 향상을 장담할 수 없고 우리가 가진 보드의 **pl**로직 내에서 가능할지 판단 불가능

-> **soc**의 리눅스에 **opencv**포팅하고 함수 내부의 계산량이 많은 부분들을 **pl**로직으로 구현

FPGA

문제점

2016. 20 GIGE 스트리밍 디자인에 Framebuffer, GIGE core, Ethernet MAC IP locked 되어있음.
-> Image processor 구현(Inverter 및 밝기 변화)

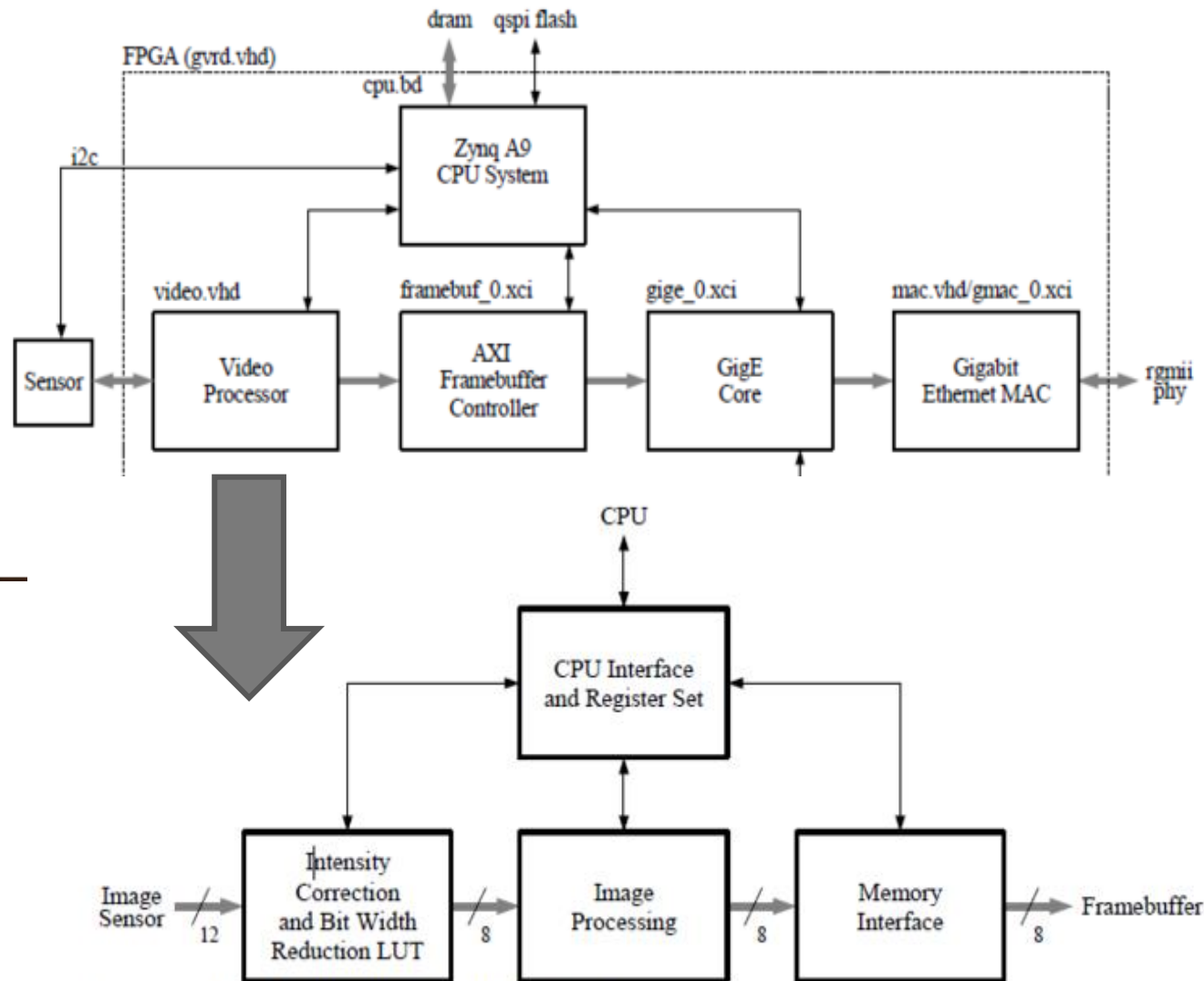


Figure 2: Block diagram of the video processor

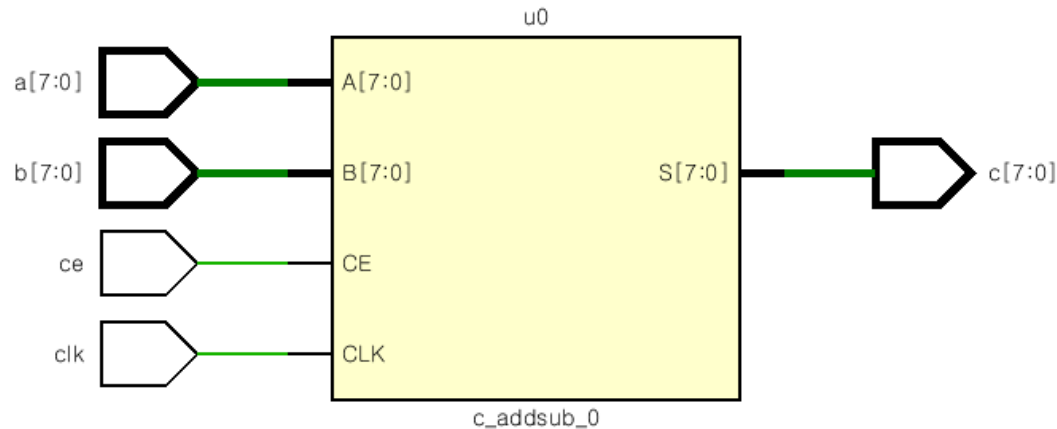
FPGA

문제점

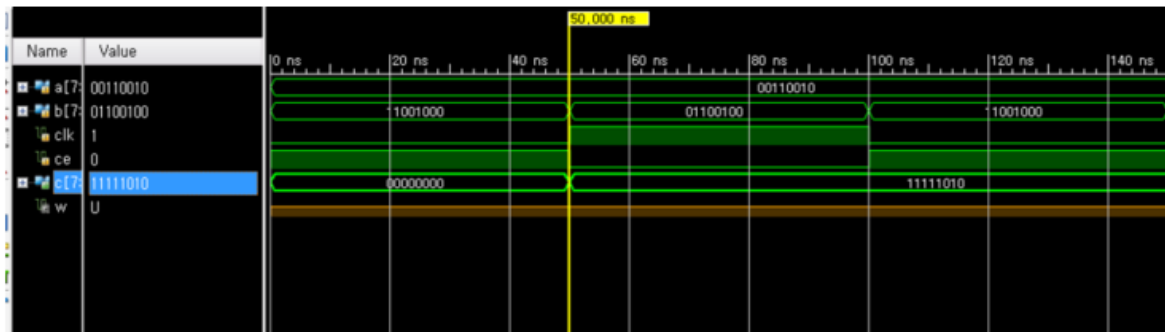
2016. 20

Vhdl에서 픽셀 밝기 값에 일정 값씩 더해 줄 수 없음. ($B=A+10$; 안됨.)

-> n-bit가산기 설계 ip를 사용하여 VHDL에서 component로 사용.



이미지 슬라이드



FPGA

문제점

2016. 20

가산기로 100을 더해주었을 때 기존에 155이상이었던 부분들은 다시 어두워지기 때문에 제외했다. 그렇게 하였을 때 밝기 값이 기존에 155이상이었던 부분들은 경계처리가 된다.



FPGA

Canny Edge detector 동영상

필터 적용전이나 후에나 **50fps**을 유지한다.



FPGA

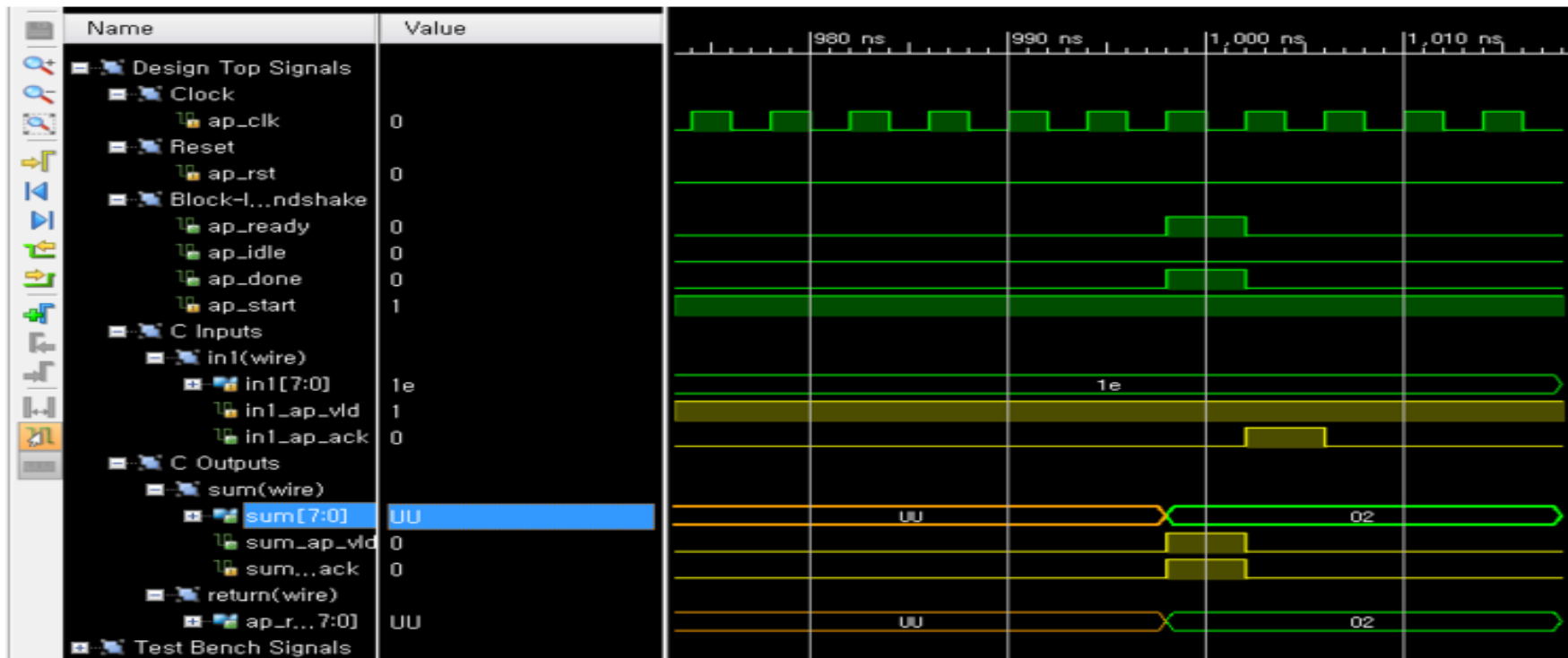
문제점

2016. 20

화질 향상을 위해 HLS를 이용하여 감마 보정 회로 설계.

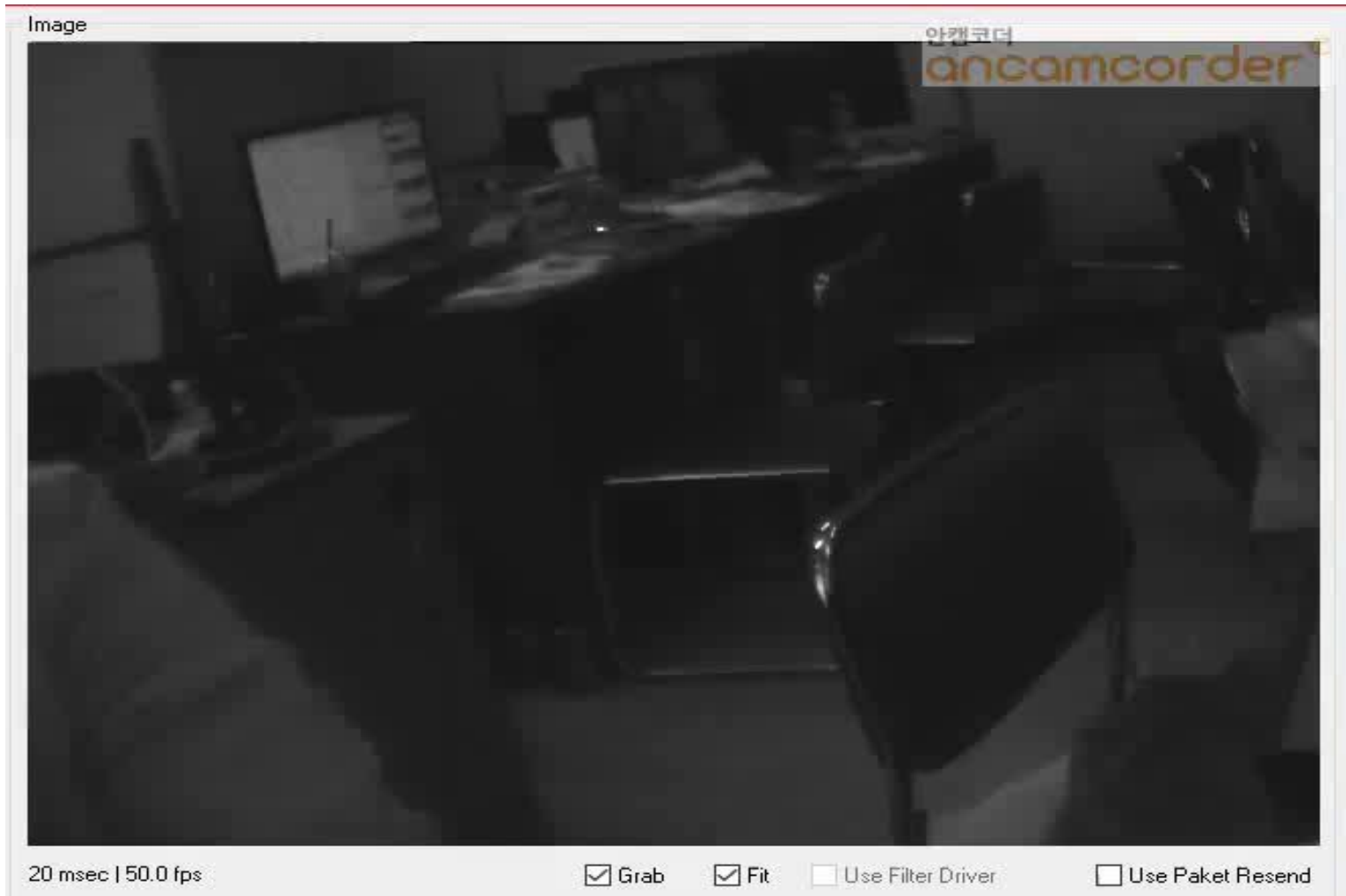
로직상에 문제가 없음을 C-RTL simulation을 통해 확인하였는데도 아웃풋에 신호가 안 나왔음.

-> ns를 넉넉히 주고 돌려보니 뒤에 아웃풋 변화가 생김을 알 수 있었음. 가산기, 인코더와 같은 단순 로직들과 달리 수 클럭 안에 결과가 나오지 않기 때문에 클럭을 기다려야 했었음.



FPGA 문제점

감마보정필터 동영상
필터를 적용하였을 경우 대비가 명확해지는 효과



펌웨어 문제점

~2016.6.28 RF보드로 넘겨줄 데이터를 만들기 위해 PMOD GPIO핀들을 사용하여 초음파 센서 제어. 10us만큼 Trig 핀에 신호를 주었음에도 결과가 나오지 않음.
-> 센서 Datasheet에서 Module로부터 Sonic burst가 발생하기 까지 Latency가 필요하다는 것을 확인하고 Cortex A9의 Global timer를 이용하여 대기하여줌.

~2016.7.3 GPIO를 이용하여 초음파센서 2개 동시 제어.
센서를 2개 이상 연결할 때 결과 값이 하나만 출력되는 문제점 발생
-> LED와 버튼을 연결하여 Xilinx SDK상에서 디버깅하여 문제점 해결.

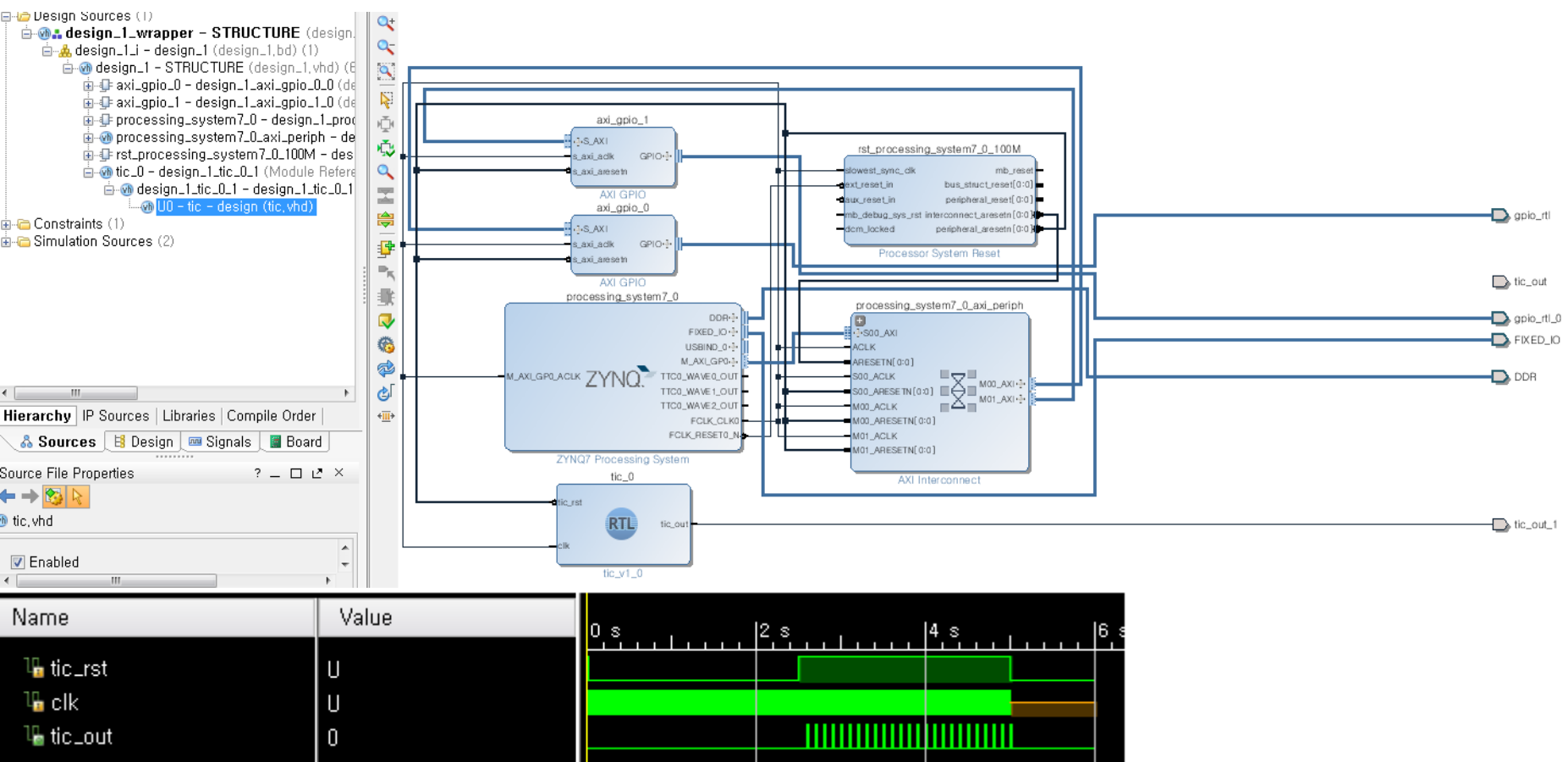


Firmware&&FPGA

문제점

~2016.7.7 Xilinx SDK에서 시간을 측정할 때 S/W적인 latency가 발생.
(명령어 하나를 실행하려면 최소 4사이클이 필요)

->TIC 타이머를 VHDL로 설계하여 시간측정에 사용함.
시스템 클록을 분주하여 카운트



HW

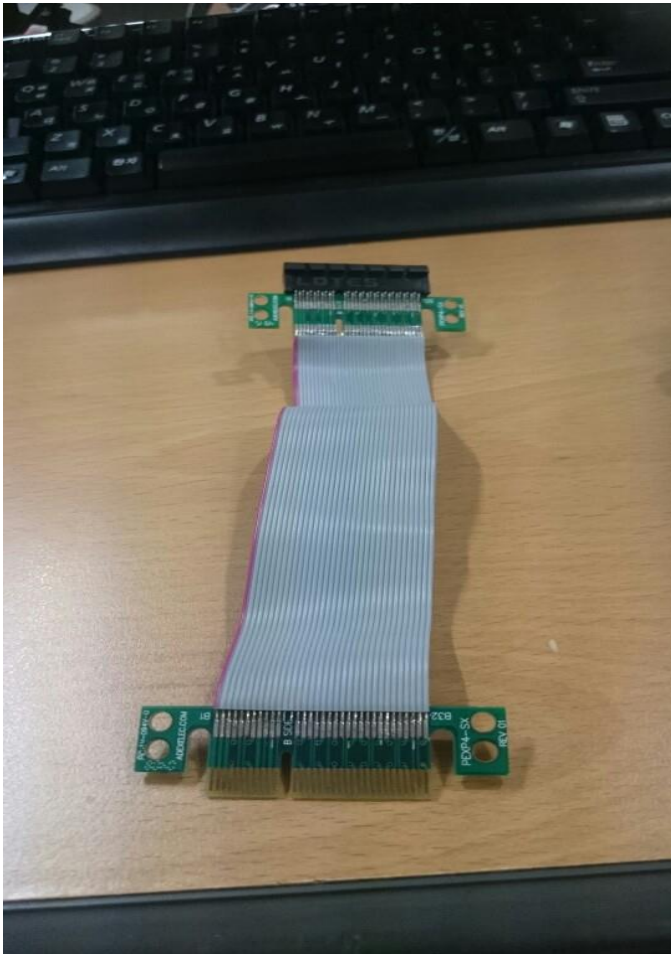
문제점

~2016.7.8

연장 케이블 보드에 연결 시 보드 전원인가 안됨

멀티미터로 점검해본 결과 하나하나 독립적이어야 하는 핀이 다 연결되어있음.

->합선, 구매 한달 만에 배송 온 카메라 연장 케이블 불량판정.



PETALINUX(Yocto)

문제점

~2016.7.12

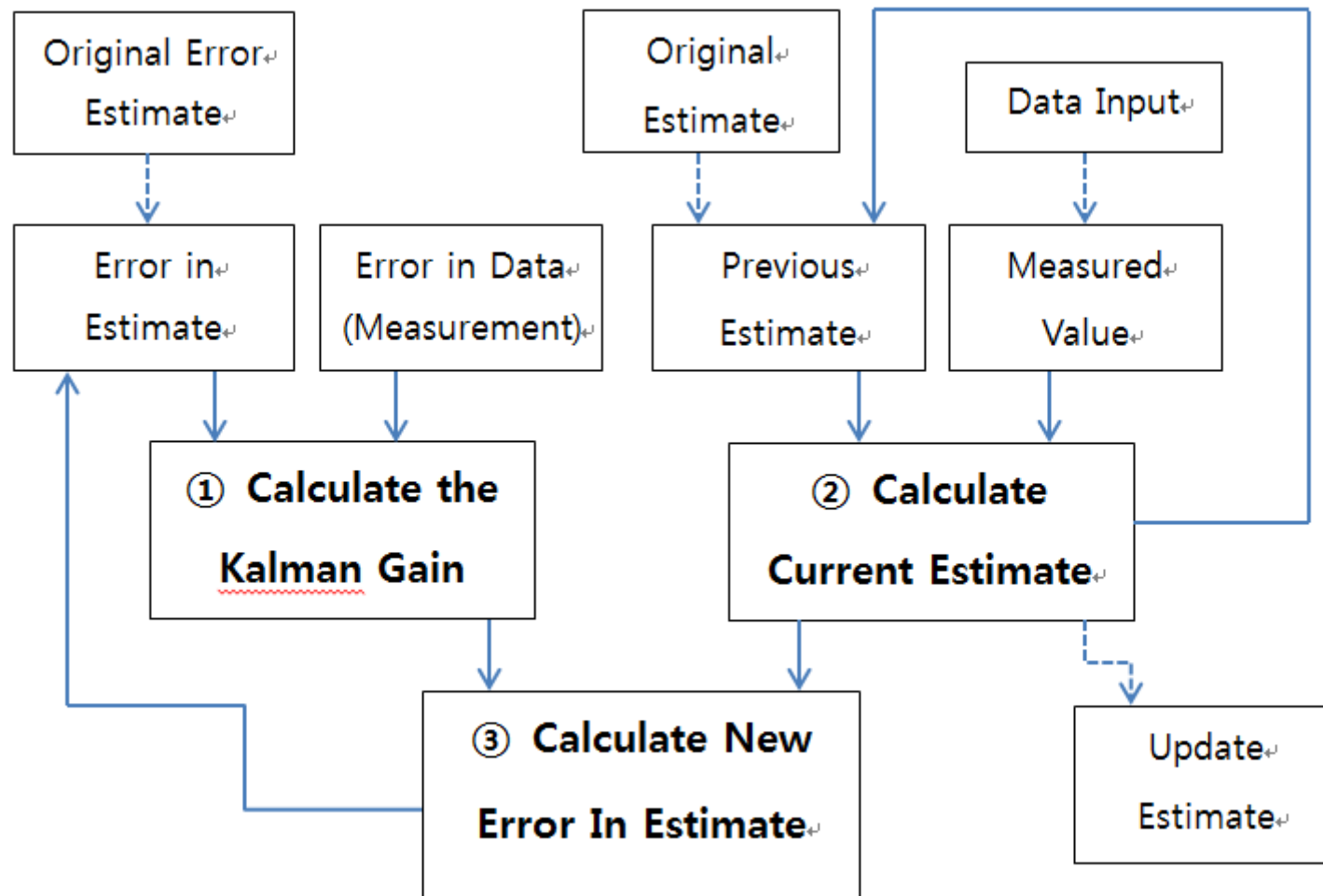
기존 zedboard에서 제공하는 리눅스는 3rd-party app이 내장되어 있음.
->부트로더 수정. Petalinux-build 후에 u-boot로 boot.bin 생성하고
emmc에 리눅스를 저장하여 부팅함.

Root file system이 qspi에 저장되었음에도 불구하고 RAMFS에서 동작.
/etc/fstab에 마운트정보를 저장해도 volatile인 상태.
SD카드나 usb쪽으로 petalinux가 지원. Emmc없음.

쿼드콥터

신영웅

칼만 필터



칼만 필터

The diagram illustrates the Kalman filter update equation: $\hat{X}_k = K_k \cdot Z_k + (1 - K_k) \cdot \hat{X}_{k-1}$. Annotations with arrows point to specific parts of the equation: '현재 추정치' (Current estimate) points to \hat{X}_k ; '측정된 값' (Measured value) points to Z_k ; '이전 추정치' (Previous estimate) points to \hat{X}_{k-1} ; and '칼만 이득' (Kalman gain) points to K_k .

현재 추정치

측정된 값

$$\hat{X}_k = K_k \cdot Z_k + (1 - K_k) \cdot \hat{X}_{k-1}$$

이전 추정치

칼만 이득

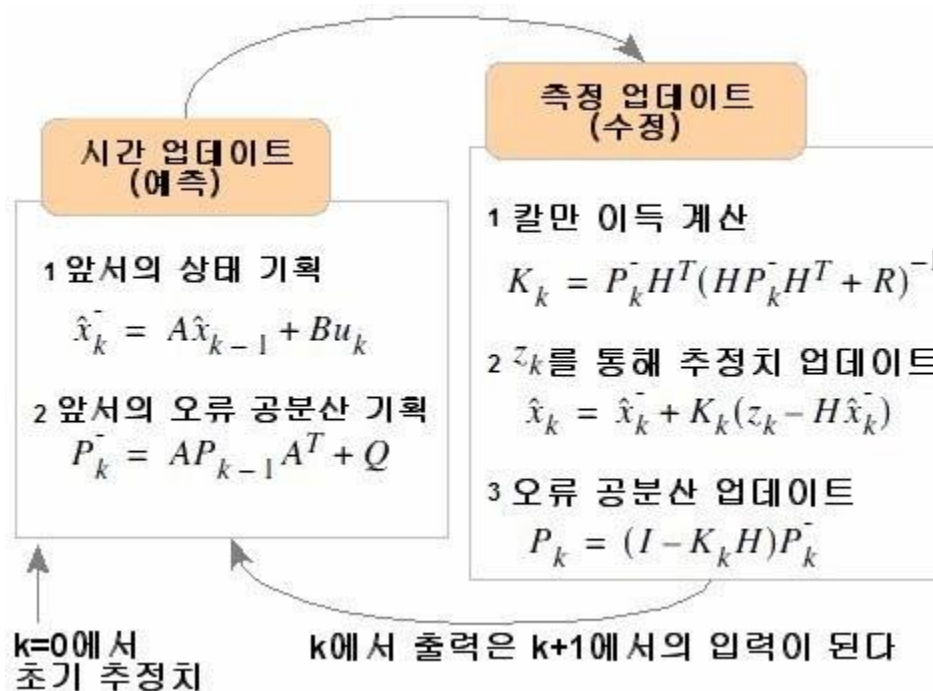
칼만 필터

$$x_k = Ax_{k-1} + Bu_k + w_{k-1}$$

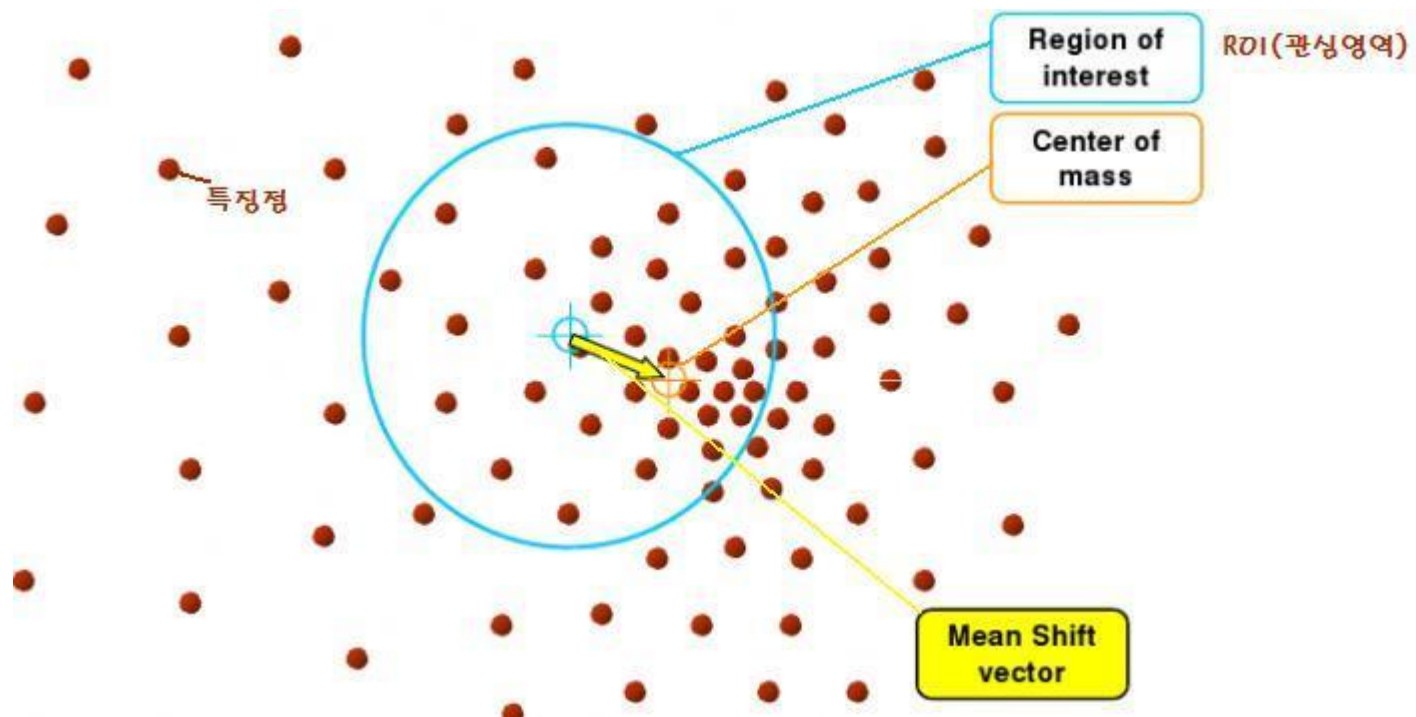
$$z_k = Hx_k + v_k$$

시간 업데이트 (예측)	측정 업데이트 (수정)
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$ $P_k^- = AP_{k-1}A^T + Q$	$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$ $\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$ $P_k = (I - K_k H)P_k^-$

칼만 필터

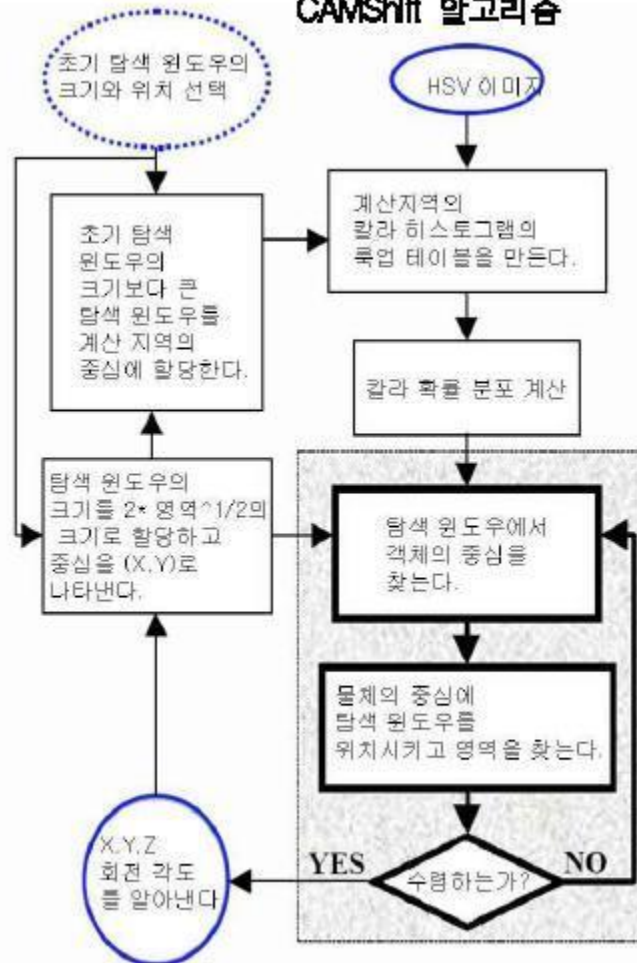


MEAN-SHIFT

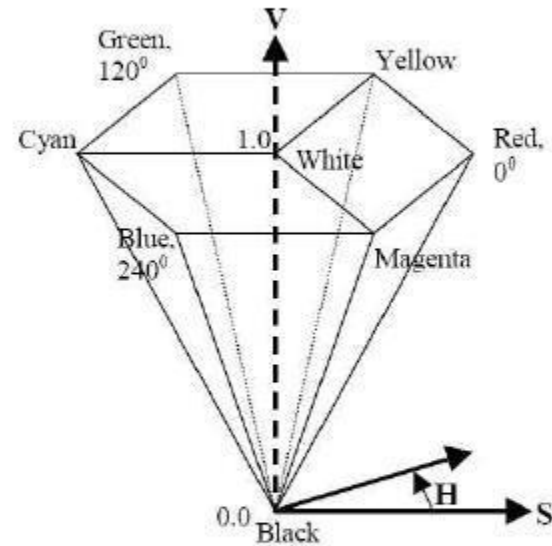


CAMSHIFT

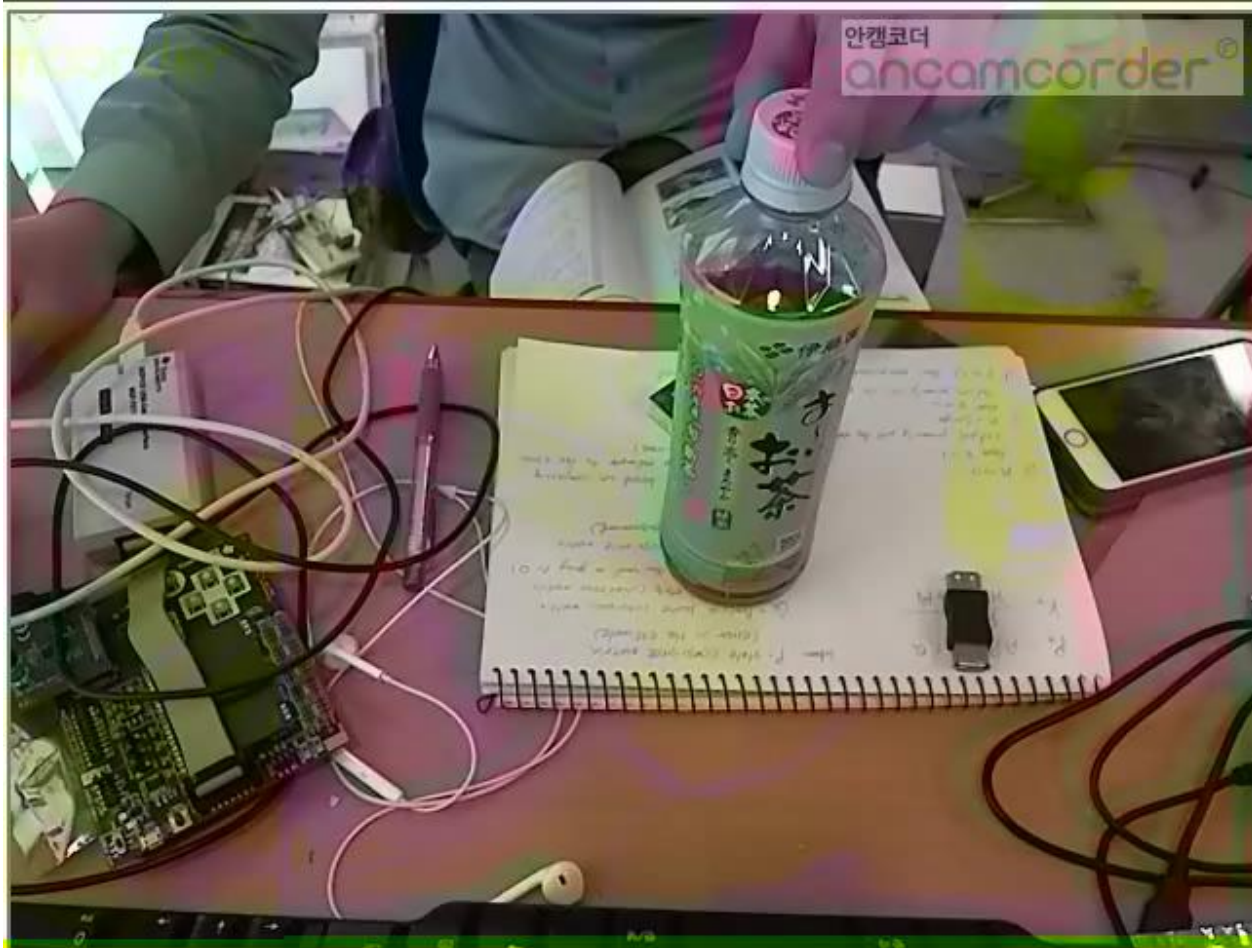
CAMShift 알고리즘



HSI 모델



CAM-SHIFT



OPTICAL FLOW

Optical flow

- Definition: optical flow is the *apparent* motion of brightness patterns in the image
- Ideally, optical flow would be the same as the motion field
- Have to be careful: apparent motion can be caused by lighting changes without any actual motion
 - Think of a uniform rotating sphere under fixed lighting vs. a stationary sphere under moving illumination

루카스-카나데

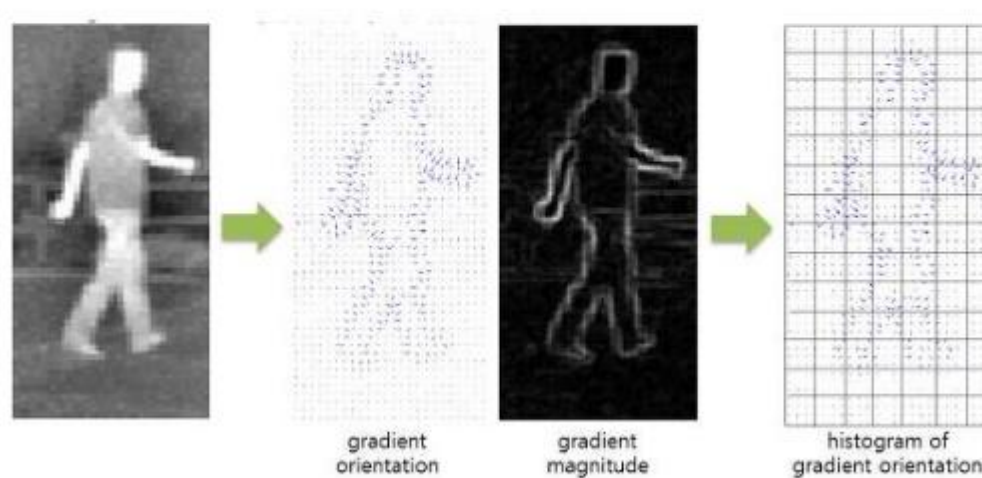
안캠코더
ancamcorder[®]

학습(HOG, LBP, HAAR)

2. HOG(Histogram of Oriented Gradient)

[Dalal05] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," CVPR 2005.

HOG는 대상 영역을 일정 크기의 셀로 분할하고, 각 셀마다 edge 픽셀(gradient magnitude가 일정 값 이상인 픽셀)들의 방향에 대한 히스토그램을 구한 후 이들 히스토그램 bin 값들을 일렬로 연결한 벡터이다. 즉, HOG는 edge의 방향 히스토그램 템플릿으로 볼 수 있다.



<그림 2>

학습(HOG, LBP, HAAR)

5. LBP(Local Binary Pattern)

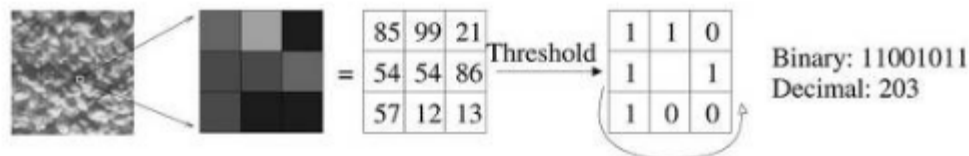
[Ojala96] Ojala, T., Pietikäinen, M. and Harwood, D., A Comparative Study of Texture Measures with Classification Based on Feature Distributions, Pattern Recognition 29(1):51-59, 1996.

[Ahonen06] T. Ahonen, A. Hadid, and M. Pietikinen, "Face description with local binary patterns: Application to face recognition," PAMI 2006.

[Liao07] S. Liao, X. Zhu, Z. Lei, L. Zhang and S. Z. Li, "Learning Multi-scale Block Local Binary Patterns for Face Recognition," ICB 2007.

LBP는 원래 영상의 텍스처(texture)를 분류하기 위한 용도로 개발된 feature인데 이후 얼굴인식(face recognition)과 같은 다른 영상인식 응용에도 활용되고 있다.

LBP(Local Binary Pattern)는 영상의 모든 픽셀에 대해 계산되는 값으로서 각 픽셀의 주변 3×3 영역의 상대적인 밝기 변화를 2진수로 코딩한 인덱스 값이다 (중앙 픽셀보다 밝으면 1, 어두우면 0으로 코딩한 후 이 값들을 연결한 이진수를 로컬 텍스처에 대한 인덱스로 활용)



<그림 5> 출처: [Ahonen06]

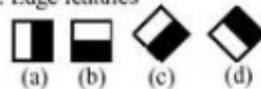
학습(HOG, LBP, HAAR)

3. Haar feature

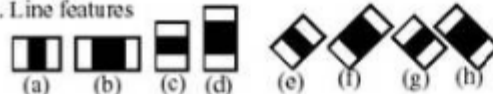
[Viola01] P. Viola and M. J. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," CVPR 2001.

Haar feature는 기본적으로 영상에서의 영역과 영역의 밝기차를 이용한 feature로서 아래 왼쪽 그림과 같이 다양한 형태의 elementary feature들이 존재하며 이들 elementary feature들을 다수(수백, 수천개) 조합하여(다양한 위치와 크기로) 물체에 대한 특징을 추출하는 방법이다.

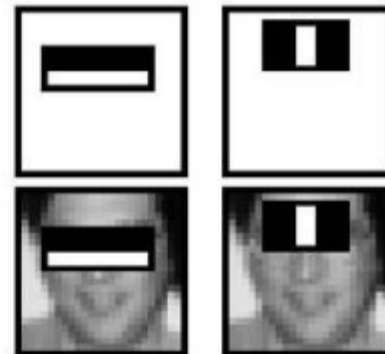
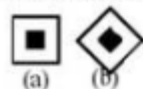
1. Edge features



2. Line features



3. Center-surround features



<그림 3>

HAAR

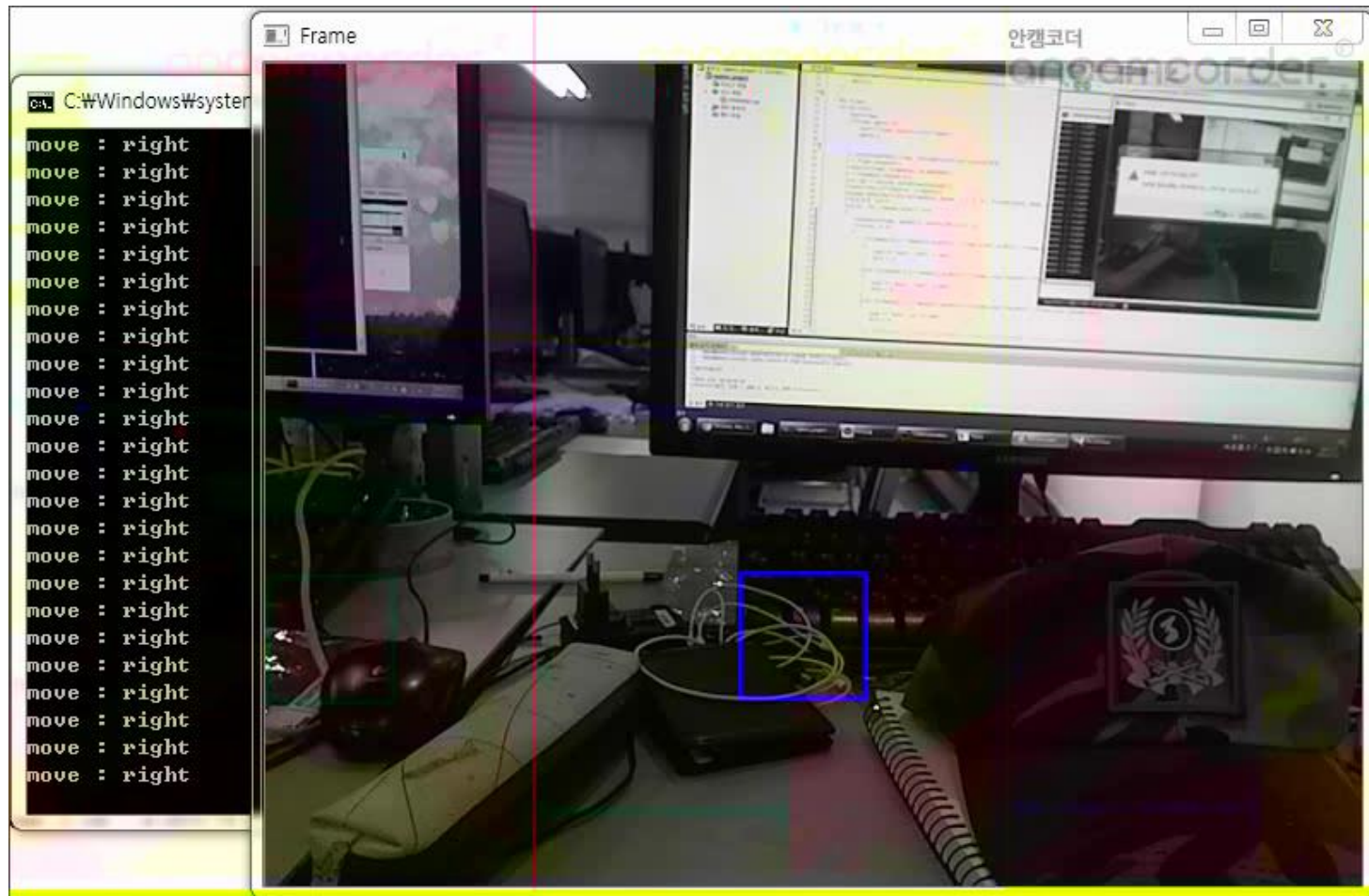


OPENCV – HAAR CODE

```
a = frame.channels();
cvtColor(frame, frameGray, CV_BGR2GRAY);
a = frameGray.channels();
bool abc = cascade.isOldFormatCascade();
cascade.detectMultiScale(frameGray, hands, 1.1, 5, 0 | CV_HAAR_SCALE_IMAGE, Size(10,10));
//관심영역 그리기
for(int i=0; i<hands.size(); i++)
{
    rectangle(frame, hands[i], Scalar(255,0,0), 2);
}
```

```
void CascadeClassifier::detectMultiScale( const Mat& image, vector<Rect>& objects,
vector<int>& rejectLevels,
vector<double>& levelWeights,
double scaleFactor, int minNeighbors,
int flags, Size minObjectSize, Size maxObjectSize,
bool outputRejectLevels )
{
```

HAAR를 통한 물체 중심 찾기



- 신영용

- 프로젝트 시작 ~ 6.9

동영상 스트리밍을 통한 물체 트래킹 시스템

절차 : 칼만 필터 (물체 인식 불가) -> Min_Shift, Cam_Shift (색 인식 불가)
-> Optical_flow[루카스 카나데] (장치 인식 문제) - 학습[Haar]

- 6.10 ~ 12

학습을 위한 .xml 파일 생성

절차 : 예비군 모자 (모자의 인식률이 떨어짐) - 붉은 별[북한군 마크]
(단순한 패턴으로 인식이 어려움) - 예비군 마크

- 6.13 ~

petalinux 명령어

보드 위에 리눅스가 sudo, apt-get, make, install, gcc 등 우리가
알고있는 명령 사용 불가

해결 방안 : opencv를 사용하기 위한cross-comfile 필요

6.11 ~

petaLinux 설정

: SVDK 전원 토글시 OS의 설정 초기화

해결 방안 : 진행중 ~

6.13 ~

Linux Device Driver & Smart RF Kit

RF Kit로 다른 보드에서 Data를 받아오기 위해 Data Sheet를 읽고 각 Register가 어떻게 연결 되어 있는지 확인. 내부 제어는 가능하나 외부에서 데이터를 받아오거나 외부로 데이터를 보내는 방법이 쉽지않음

6.18 ~

SVDK & R5 통신

ethernet2can (모듈 사용)

SVDK에서 ethernet 예제 실행시 패킷이 보내지는지 확인 불가. SVDK와 PC를 같은 방법으로 연결 시키고 모듈 프로그램으로도 확인 불가.

6.29 ~

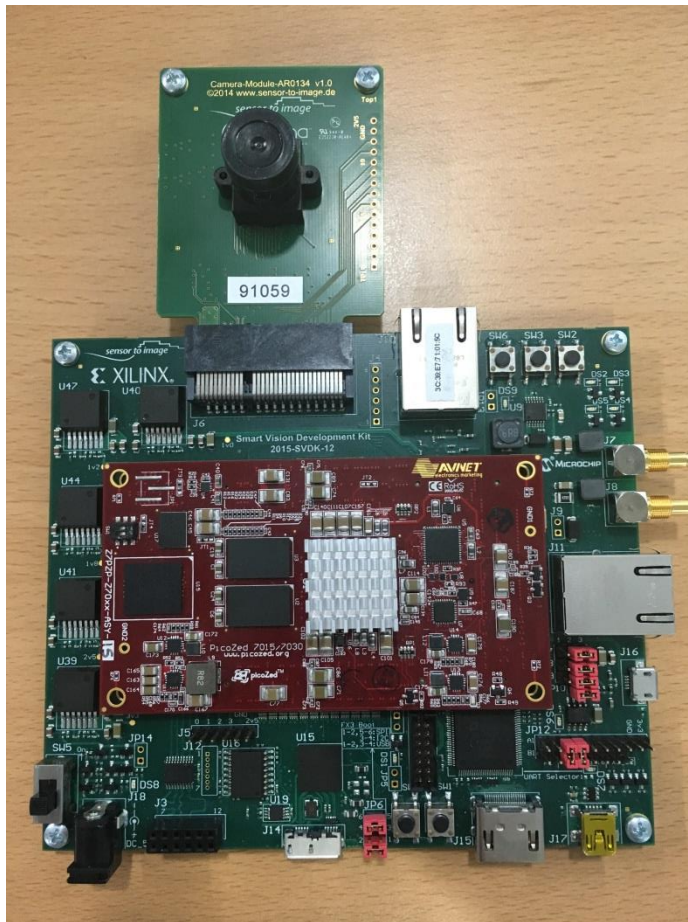
SVDK & SmartRF 통신

절차 : ethernet2ethernet (SmartRF ethernet register 존재 하지 않음) ->
ethernet2uart (같은 이유) -> uart2uart (진행중)

-> SmartRF에서 Uart통신 함수가 존재하나 main 함수 및 사용하고자 하는 곳에 Uart
초기화 함수를 사용하면 Board 디버그시 작동하지 않음



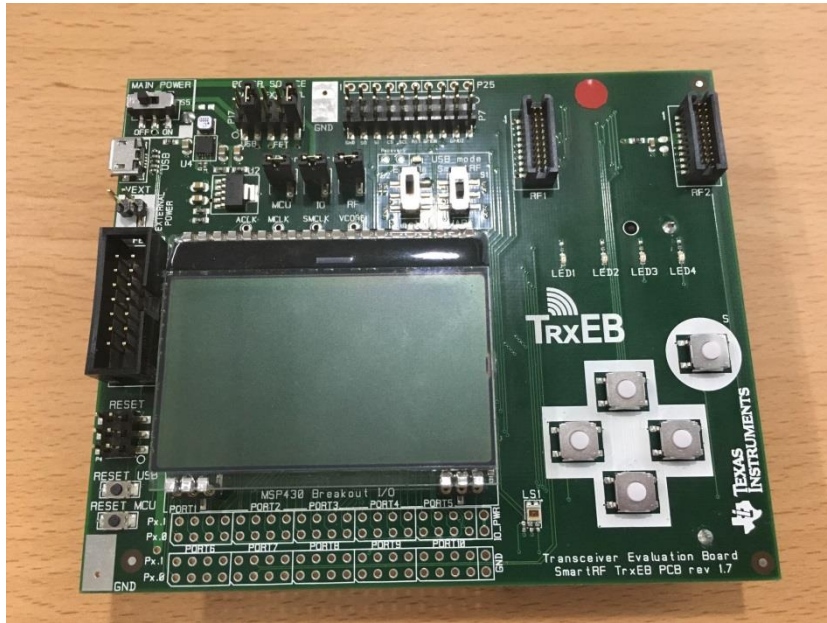
UART - SVDK



UART – SVDK & PC



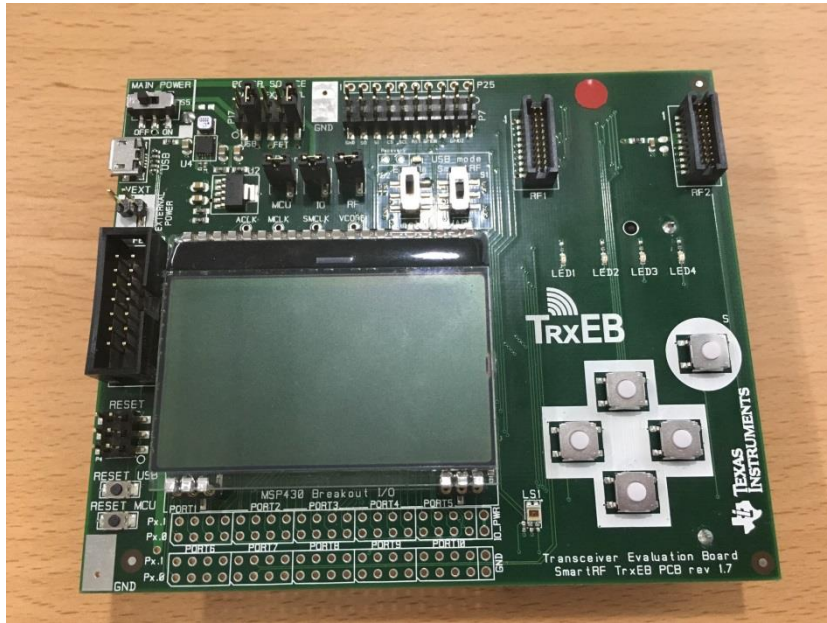
UART - RF



UART – RF & PC



UART - RF



UART – SVDK & RF



UART – 아두이노 & 아두이노

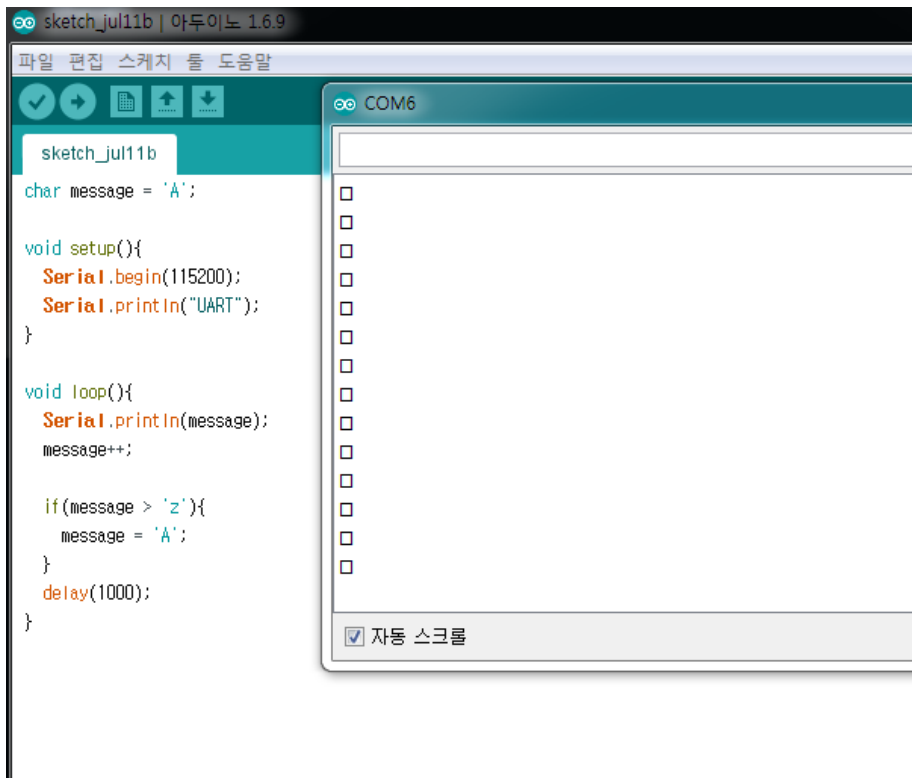


UART – 아두이노 & 아두이노



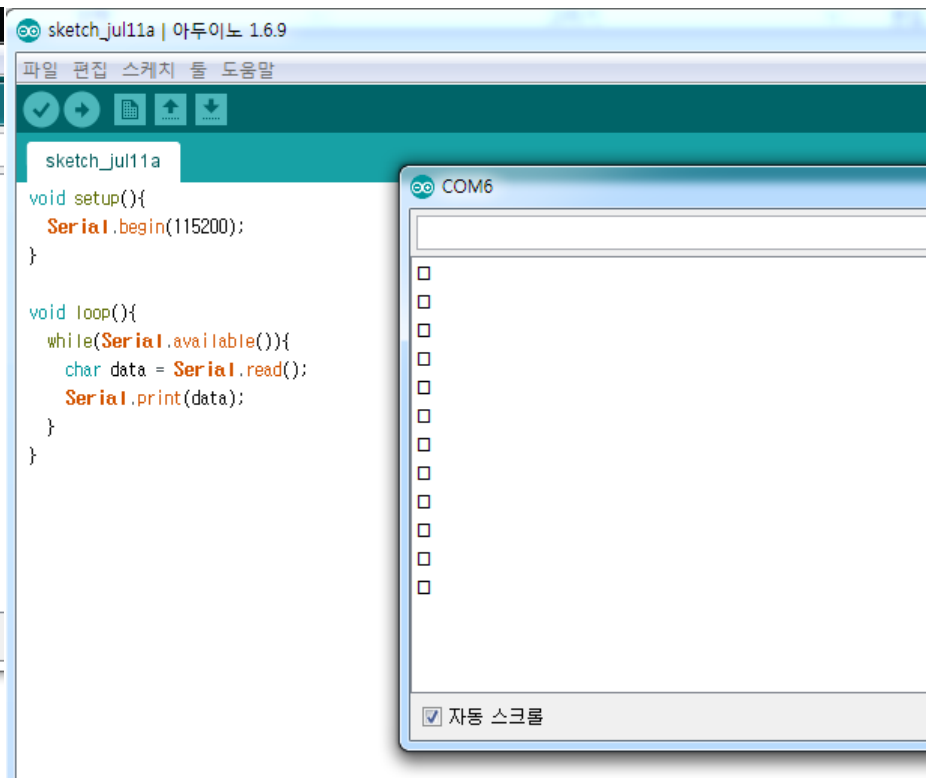
UART – 아두이노 & PC





업로딩...

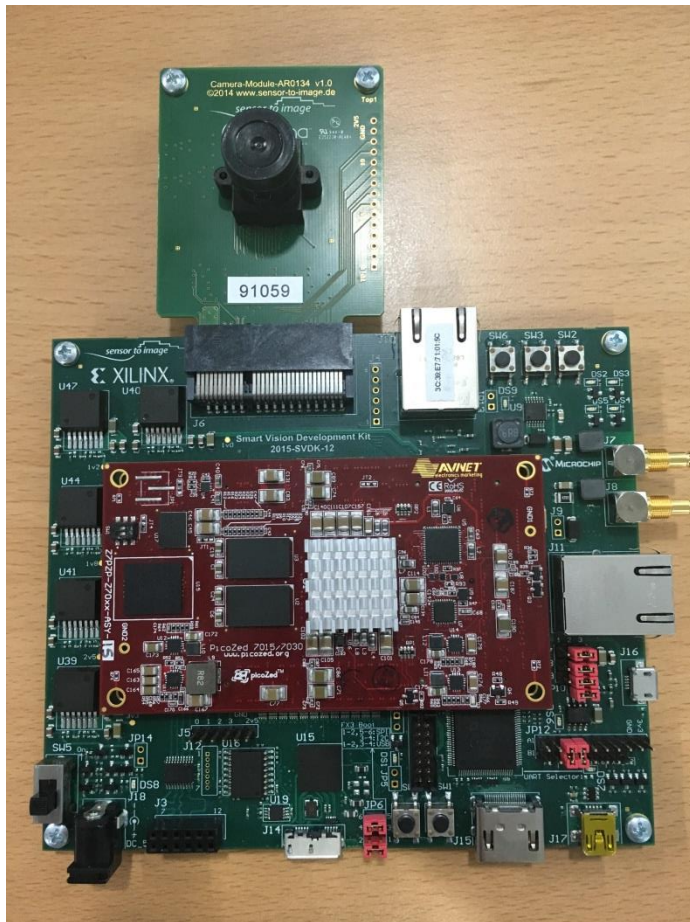
스케치는 프로그램 저장 공간 2,068 바이트(6%)를 사용, 최대 32,256 바이트.
전역 변수는 동적 메모리 206바이트(10%)를 사용, 1,842바이트의 지역변수가 남음, 최대는 2,048 바이트.
avrdude: stk500_getsync() attempt 1 of 10: not in sync: resp=0x87
avrdude: stk500_getsync() attempt 2 of 10: not in sync: resp=0x0d
avrdude: stk500_getsync() attempt 3 of 10: not in sync: resp=0x0a
avrdude: stk500_getsync() attempt 4 of 10: not in sync: resp=0x88
avrdude: stk500_getsync() attempt 5 of 10: not in sync: resp=0x0d
avrdude: stk500_getsync() attempt 6 of 10: not in sync: resp=0x0a
avrdude: stk500_getsync() attempt 7 of 10: not in sync: resp=0x89
avrdude: stk500_getsync() attempt 8 of 10: not in sync: resp=0x0d
avrdude: stk500_getsync() attempt 9 of 10: not in sync: resp=0x0a
avrdude: stk500_getsync() attempt 10 of 10: not in sync: resp=0x8a



업로딩...

스케치는 프로그램 저장 공간 1,714 바이트(5%)를 사용, 최대 32,256 바이트.
전역 변수는 동적 메모리 182바이트(8%)를 사용, 1,866바이트의 지역변수가 남음, 최대는 2,048 바이트.
avrdude: stk500_getsync() attempt 1 of 10: not in sync: resp=0xdf
avrdude: stk500_getsync() attempt 2 of 10: not in sync: resp=0x0d
avrdude: stk500_getsync() attempt 3 of 10: not in sync: resp=0x0a
avrdude: stk500_getsync() attempt 4 of 10: not in sync: resp=0xe0
avrdude: stk500_getsync() attempt 5 of 10: not in sync: resp=0x0d
avrdude: stk500_getsync() attempt 6 of 10: not in sync: resp=0x0a
avrdude: stk500_getsync() attempt 7 of 10: not in sync: resp=0xe1
avrdude: stk500_getsync() attempt 8 of 10: not in sync: resp=0x0d
avrdude: stk500_getsync() attempt 9 of 10: not in sync: resp=0x0a
avrdude: stk500_getsync() attempt 10 of 10: not in sync: resp=0xe2

SVDK & 아두이노



TMS570LC4357 (CORTEX R5) HDK 보드

개발 진행 사항

백영민

I. 시스템 클록을 이용해 ePWM 제어.

- 개발 초기단계 : HalCoGen의 Default 설정인 **Up Counter**로 75MHz 시스템 클록 환경에서 **PWM** 제어를 실시함.
- 개발 중/후기 단계 : 실제 모터제어에 필요한 **ESC**가 **PWM** 제어를 위해 요구되는 최소 주기가 2ms인 것을 고려해 시스템 클록을 50MHz(=파장 1ms)로 낮추고, Time Base Counter를 **Up-Down Counter**로 설정해 주기를 2배로 늘려 전체 주기를 2ms로 설정함.

(feat with 고추잠자리)

개발 진행사항 (계속)

II. MSP6050 센서와 R5F 보드간 통신

- I2C 통신 프로토콜을 이용해 외부 센서모듈을 **Slave**로, **Data**를 받을 R5F 보드를 **Master**로 설정해서 통신환경을 구성함.
- **Master**쪽인 R5F 보드는 센서 값을 받는 입장이기 때문에 **Packet Sequence**를 **Burst Read Sequence**로 구성함. (Data Sheet 참고)
- 한번에 받을 **Data** 개수를 14개(가속도 3축, 자이로 3축, 온도)로 설정해 **Burst Read** 하도록 설정함. (각 값마다 레지스터가 2개씩 필요)
- 초기 연결 **Sequence**와 **Burst Read** 부분을 각각 별도의 함수형태로 만들어 **Task**를 구성함.

개발 진행사항 (계속)

III. ESP8266-12E WIFI 모듈과 R5F 보드 연동

- 모듈을 사용하기 위해 최신 Firmware(v1.5, baud rate 115200)를 모듈 내 Flash memory에 설치함.
- WIFI 모듈에 불안정한 전압 인가를 방지하기 위해 330uF Capacitor를 회로에 같이 구성함.
- AT 명령어로 WIFI 모듈이 중간 공유기 역할(AP mode)을 하도록 설정하고, 무선 연결된 station과 보드간 1:1 연결(\approx WIFI Direct)이 되도록 구성. 모듈과 R5F보드는 UART로 연결해 정보를 주고 받을 수 있도록 함.
- WIFI 모듈이 외부 장치로부터 정보를 받을 경우 vTaskSuspendAll 함수를 호출해 Critical Section과 유사하게 동작하도록 구성함.

개발 진행사항 (계속)

IV. RTOS Task 처리

- 개발 초기 단계 : 모든 R5F에 연결돼 동작하는 것들을 전부 Task로 만듦.
(외부로 LED를 연결해 ePWM을 시각적으로 제어하는 Task1, MPU6050에서 센서 값을 받는 Task2, 센서 값을 이용해 Kalman 연산을 수행하는 내부 연산 Task3, PID 제어를 하는 연산 Task4, WIFI 모듈로 무선 통신을 하는 Task5로 구성.)
- 개발 중/후기 단계 : 내부적으로 연산을 하는 Task는 Task가 아니라 함수 형태로 만들고(Kalman 연산, PID 연산), 서로 종속적인 관계가 있는 Task들을 하나의 Task로 묶어 순차적으로 실행하도록 함.
(센서값+Kalman필터+PID제어 & WIFI 통신+ePWM)

Task1

Task2

개발 진행사항 (계속)

V. ESC Calibration

- 보유하고 있는 BLCeli 60A ESC 모델은 PWM을 위한 최소 throttle 값이 1ms였기 때문에 Min throttle을 설정할 때 50% 이상 Duty Cycle을 설정해야 정상적으로 PWM 제어를 할 수 있음.
(Up-Down Counter로 Full Duty가 2ms이기 때문, 54%로 설정해줌)
- 현재 Calibration() 함수로 만들어 R5F 보드가 reset 될 때마다 함수호출이 되는데 Data Sheet를 참고해 Max, Min값을 Latency를 주어 자동적으로 들어가게 하거나 혹은 기존 설정이 저장되면 단순히 연결만 시켜 사용하도록 구상 중.

TMS570LC4357 (CORTEX R5) HDK 보드

- 센서 제어 및 통신, **PWM** 제어, **RTOS** 환경 실시간 태스크 구현 및 스케줄링, **WIFI** 무선 연결(안드로이드-**WIFI**모듈-HDK 보드) 담당

백영민

-> 원인 ==> 해결

문제점

5월 26일 ~ 5월 31일 : 하드웨어 환경설정 및 제공된 소프트웨어 사용법 미숙

==> Technical Reference, User Guide Data sheet로 준비

6월 2일 : 가속도-자이로 센서(MPU6050)의 동작을 위한 정격 전원을 HDK 보드의

GPIO핀 중 하나를 선택해 개발보드와 연결을 했지만 센서가 동작하지 않음

-> 센서의 전압 규격이 3.3V ~ 5V 내에서 동작을 하지만 HDK 보드의 **GPIO** 핀은 3.3V, 이마저도 정상출력이 되지않고 3.2V 중반으로 전압이 출력
센서가 동작하기 위한 전압이 부족했음.

==> 임시적인 방법으로 아두이노 우노(UNO) 보드의 5V DC 전원 출력을
전원으로 사용함(4V 후반의 전원을 유지해 동작이 가능하게 됨)

문제점

6월 6일 : MPU6050 센서와 개발 보드간 통신을 위해 I2C 프로토콜을 이용해야 했는데

I2C의 통신 매커니즘대로 구현을 했으나 개발 보드쪽에서 센서값 확인 불가

-> 외부 센서 모듈이 통신을 하기 전에 충분한 전원과 장치가 동작할 시간이 필요했음.

==> 전원 인가 외에도 센서와 통신 연결을 하기 전, 고의적인 Latency 시간을 두어 센서의 부팅과 같은 하드웨어 준비를 명확하게 함.

6월 11일 : 개발 보드의 ePWM 제어를 통해 외부 LED의 밝기를 조절하는 실험을 하는

단계에서 HalCoGen 프로그램의 Duty 비율 지정에 대한 궁금증이 생김.

HalCoGen 프로그램을 통해 ePWM의 Default Duty 비율을 50%로 설정할

경우, 75MHz 시스템 클럭에 1us의 주기를 가지면 '38'이라는 값이 레지스터에 저장됨. '38'이라는 값이 실제 Duty Cycle의 [비율]을 나타내는 것인지, 50%일 때의 LED 밝기값을 나타내는 것인지 의문을 가짐

==> 시스템 클럭과 Period(주기)의 변화를 통해 실험한 결과 50%일 때의 LED 밝기값, PWM제어의 결과로 출력되는 전압의 크기였다. (즉, 후자)

문제점

6월 15일 : WIFI 모듈(ESP8266-12E)을 동작시키기 위해서 특정한 펌웨어를 모듈의 플래시 메모리 영역에 설치해줘야 하는데 문제점이 발생함.

-> 펌웨어를 설치하기위한 다양한 전용 툴과 방법이 있었지만 제대로 동작해서 설치가 이루어진 툴을 찾는데 시간을 많이 소비함.

==> ESP flash 프로그램과 아두이노 우노 보드를 이용해 1.5v firmware 설치를 완료함.

6월 17일 ~ 현재 : 개발 보드와 WIFI 모듈과의 연결 이전에 아두이노 우노(UNO) 보드를 이용해 시리얼 통신을 하는 실험 중 문제점 발견됨.

-> ESP8266 모듈은 모듈에 인가되는 전원이나 혹은 전류가 부족하게 인가되면 RST(restart) 신호를 보내는데, 특이할 정도로 매우 반복적으로 RST 응답신호를 보냈다.

(시리얼 통신으로 확인 가능)

==> '전원부' 설계가 되어 해결. 하지만 배터리 용량이 턱없이 부족함

문제점

6월 23일 ~ 현재 : 가속도-자이로 센서와 ePWM제어, 센서의 입력값을 가지고 계산되는 Kalman 필터의 연산, WIFI 모듈 통신 작업 모두를 RTOS의 환경에서 별도의 Task 형태로 구현을 했으나 도중 스케줄링 문제가 발생함.

-> 각각의 Task가 동일한 Priority로 설정되고(각각의 작업들이 무한루프를 돌면서 작업을 해야하기 때문에 다른 우선순위로 설정을 하게되면 스케줄링을 위한 Context Switching이 발생하지 않음), 이로인해 Round Robin 형태의 스케줄링이 진행된다. 그러나 Task들 중에는 '종속적'인 관계를 가진 Task가 있었기 때문에 이상한 값이 출력되거나 연산이 전혀 되지않아 Task가 동작을 안 하는 것처럼 보여지기도 함. 추가로, 지속적인 전원의 불안정으로 WIFI 모듈의 동작이 수시로 RST되는 현상이 발생함.

==> 종속적인 관계를 가진 Task를 찾아 병합을 해주는 작업을 실시

(가속도-자이로 센서 <-> Kalman 필터 <-> PID 제어,

ePWM제어 <-> WIFI 연결을 통한 안드로이드 명령)

진행상황 - 1주차 이성용

2016/5/30 : 칼만필터의 개념 파악 및 자료수집

2016/5/31 : '칼만필터의 이해' 책으로 처음 칼만 필터가 나오게 된 배경을 파악하였고 필요한 것은 통계학. 평균을 재귀함수로 구현하는 방법으로 시작한 칼만필터

2016/6/01 : 시스템모델변수가 필요하다고함. 자료수집한 파일들을 바탕으로 시스템 모델 변수에 대해 파악하는 과정에서 통계에 깊은 지식이 필요하단 것을 알

2016/6/02 : 제대로 이해하기 위해 MATLAB을 이용하여 칼만필터를 구현하여 보았다.

2016/6/03 : C언어로 칼만필터를 구현한 코드를 구해서 코드분석

1주차 문제점 : 칼만필터에 대한 정확한 기본지식이 부족하였고 통계학에 대한 지식이 필요함
칼만필터에 대한 C 프로그램 분석자료 중...

```
// Constants.
static float A[n][n] = { { 1.0, 0.019968 }, { 0.0, 1.0 } }; /* 시스템 모델 A, 현 단계에서의 상태변수와 다음단계의
                                                             상태변수를 연결하는 변환계수 */

static float B[n][m] = { { 0.00019968 }, { 0 } };
static float C[r][n] = { { 1.0, 0.0 } };                // 시스템모델 H time step에 따른 변화 가능성
static float R[r][r] = { { 0.0004 } };                  // 시스템모델 R 측정 노이즈 분산
static float Q[n][n] = { { 1.0, 0.0 }, { 0.0, 1000.0 } }; // 시스템모델 Q 프로그램 노이즈 분산

// Persistent states.
static float xhat[n][1] = { { 0.0 }, { 0.0 } };          // 초기 상태변수값 아마 자이로 가속도
static float P[n][n] = { { 3.0, 0.0 }, { 0.0, 3.0 } };   // 초기 오차 공분산 아마 예측값
```

진행상황 - 2주차

2016/6/06 : 코드분석 60%정도..

2016/6/07 : 더 깊은 이해를 위해 통계에 대한 지식이 필요하다 판단하여 통계 공부를 하고 있다.. 공분산.. 가우스분포.. 조건부확률밀도함수 등등...

2016/6/08 : 코드분석 100% 완료!! 하지만 이해는 50프로..... 아직은 어떻게 시스템모델변수를 설정해야할지 감이 안옴....

2016/6/09 : 코드를 새로 짜면서 나머지를 이해하기로 하며 코드 만들기 시작!!

2016/6/10 : 코드작성 50% 완료

2주차 문제점 : 확률과 통계를 바탕으로 설계되는 칼만필터에서 통계에 대한 지식이 너무 없었다.
또한 새롭게 칼만필터를 만들면서 지속적인 오류 발생
-> 최적화 과정을 거치지 않아 오차가 지속적으로 누적 되는것으로 판명

```
/* Compute the innovation.
Inn = y - c * xhat;
c = H(시스템모델)
yCchat(Inn) = z(k)(y -> 가속도 각도값) - H*x^(k) -> z(k)를 통해 추정치 업데이트
*/

matrix_multiply((float*)C, (float*)AxhatBu,
matrix_subtraction((float*)y, (float*)Cchat,

/*
Update the state estimate.
z(k)값을 통해 추정치 업데이트
여기서 z(k)는 가속도 각도값
xhat = xhat_est + K * Inn;
x^(k) = {x^(k)} = xhat_est + [(K(k))*z(k)]
*/
matrix_multiply((float*)K, (float*)yCchat, n,
matrix_addition((float*)AxhatBu, (float*)KycC,

/*
예리공분산 업데이트
P(k) = (I-K(k)H)P^(k)
*/
matrix_multiply((float*)K, (float*)C, n, r,
matrix_multiply((float*)KC, (float*)P, n, n,
matrix_subtraction((float*)P, (float*)KCP, n, n, (float*)P);
// Return the estimate.
```

진행상황 - 3주차

2016/6/13 : 코드 반복을 통한 코드 마무리 작업

2016/6/14 : 선형칼만필터를 마무리 하고 비선형칼만필터를 위한 시스템모델변수를 함수화 하려고 시도하다가 포기

포기한 이유는 선형만으로도 일단 충분하고 칼만하나에 많은 시간을 투자할 수 없기때문에 쿼드로터를 안정적으로 띄우고 난후에 시도할 예정

```
self->x_00 = (self->A_01 * self->x_10) + (self->B_00 * u);
```

```
/* Compute the innovation -- error between measured value and state.
```

```
inn = y - c * x
```

```
예측값에서 예측한 x_00가 측정 업데이트로 넘어가면서  
*/
```

```
inn_00 = y - self->x_00;
```

```
// Compute the covariance of the innovation.
```

```
// s = C * P * C' + Sz
```

```
s_00 = self->P_00 + self->Sz_00;
```

```
// Compute AP matrix for use below.
```

```
// AP = A * P
```

```
AP_00 = self->P_00 + self->A_01 * self->P_10;
```

```
AP_01 = self->P_01 + self->A_01 * self->P_11;
```

```
AP_10 = self->P_10;
```

```
AP_11 = self->P_11;
```

```
// Compute the kalman gain matrix.
```

```
// K = A * P * C' * inv(s)
```

```
K_00 = AP_00 / s_00;
```

```
K_10 = AP_10 / s_00;
```

```
// Update the state estimate.
```

```
// x = x + K * inn
```

```
self->x_00 += K_00 * inn_00;
```

```
self->x_10 += K_10 * inn_00;
```

```
// printf("k_00 = %f k_10 = %f\n ", K_00, K_10);
```

```
// Compute the new covariance of the estimation error.
```

```
// P = A * P * A' - K * C * P * A' + Sw
```

```
APAT_00 = AP_00 + (AP_01 * self->A_01);
```

```
APAT_01 = AP_01;
```

선택 C:\Users\wmmlik\Documents\Visual Studio 2015\Projects\Project1\Debug\Project1.exe

roll_gyro(x) = 64.000 deg/sec	roll_accel(x) = -9.000 deg	roll_kalman_output(x) = -8.9675 deg
pitch_gyro(x) = 23.000 deg/sec	pitch_accel(x) = -15.000 deg	pitch_kalman_output(x) = -14.9451 deg
roll_gyro(x) = 45.000 deg/sec	roll_accel(x) = 4.000 deg	roll_kalman_output(x) = 3.9833 deg
pitch_gyro(x) = 9.000 deg/sec	pitch_accel(x) = 10.000 deg	pitch_kalman_output(x) = 9.9604 deg
roll_gyro(x) = 47.000 deg/sec	roll_accel(x) = 10.000 deg	roll_kalman_output(x) = 9.9607 deg
pitch_gyro(x) = 6.000 deg/sec	pitch_accel(x) = -7.000 deg	pitch_kalman_output(x) = -6.9755 deg
roll_gyro(x) = 90.000 deg/sec	roll_accel(x) = 0.000 deg	roll_kalman_output(x) = -0.0013 deg
pitch_gyro(x) = 64.000 deg/sec	pitch_accel(x) = 0.000 deg	pitch_kalman_output(x) = -0.0015 deg
roll_gyro(x) = 70.000 deg/sec	roll_accel(x) = 6.000 deg	roll_kalman_output(x) = 5.9760 deg
pitch_gyro(x) = 63.000 deg/sec	pitch_accel(x) = 5.000 deg	pitch_kalman_output(x) = 4.9797 deg
roll_gyro(x) = 26.000 deg/sec	roll_accel(x) = -6.000 deg	roll_kalman_output(x) = -5.9792 deg
pitch_gyro(x) = 78.000 deg/sec	pitch_accel(x) = -8.000 deg	pitch_kalman_output(x) = -7.9713 deg
roll_gyro(x) = 3.000 deg/sec	roll_accel(x) = 7.000 deg	roll_kalman_output(x) = 6.9718 deg
pitch_gyro(x) = 66.000 deg/sec	pitch_accel(x) = 6.000 deg	pitch_kalman_output(x) = 5.9760 deg
roll_gyro(x) = 38.000 deg/sec	roll_accel(x) = -1.000 deg	roll_kalman_output(x) = -0.9979 deg
pitch_gyro(x) = 75.000 deg/sec	pitch_accel(x) = 12.000 deg	pitch_kalman_output(x) = 11.9536 deg
roll_gyro(x) = 44.000 deg/sec	roll_accel(x) = 5.000 deg	roll_kalman_output(x) = 4.9797 deg
pitch_gyro(x) = 48.000 deg/sec	pitch_accel(x) = -13.000 deg	pitch_kalman_output(x) = -12.9528 deg
roll_gyro(x) = 41.000 deg/sec	roll_accel(x) = 10.000 deg	roll_kalman_output(x) = 9.9609 deg
pitch_gyro(x) = 37.000 deg/sec	pitch_accel(x) = -9.000 deg	pitch_kalman_output(x) = -8.9679 deg
roll_gyro(x) = 72.000 deg/sec	roll_accel(x) = -1.000 deg	roll_kalman_output(x) = -0.9976 deg
pitch_gyro(x) = 82.000 deg/sec	pitch_accel(x) = 5.000 deg	pitch_kalman_output(x) = 4.9800 deg
roll_gyro(x) = 48.000 deg/sec	roll_accel(x) = -15.000 deg	roll_kalman_output(x) = -14.9454 deg
pitch_gyro(x) = 23.000 deg/sec	pitch_accel(x) = -7.000 deg	pitch_kalman_output(x) = -6.9756 deg
roll_gyro(x) = 81.000 deg/sec	roll_accel(x) = 14.000 deg	roll_kalman_output(x) = 13.9464 deg
pitch_gyro(x) = 58.000 deg/sec	pitch_accel(x) = 6.000 deg	pitch_kalman_output(x) = 5.9761 deg
roll_gyro(x) = 28.000 deg/sec	roll_accel(x) = -9.000 deg	roll_kalman_output(x) = -8.9681 deg
pitch_gyro(x) = 85.000 deg/sec	pitch_accel(x) = 11.000 deg	pitch_kalman_output(x) = 10.9576 deg
roll_gyro(x) = 73.000 deg/sec	roll_accel(x) = -5.000 deg	roll_kalman_output(x) = -4.9827 deg

진행상황 - 4주차

2016/6/20 : 쿼드로터를 처음 접하면서 필요한 지식 조사. 필요한 내용을 파악해보니 물체의 동역학이 필수 뉴턴의 운동법칙에 대한 자료 정리 및 공부

2016/6/21 : 동역학에 대한 공부를 시작. 일단 좌표계에 대해 공부, 관성좌표계

2016/6/22 ~ 24: 구면좌표계 공부. Roll, pitch, yaw -> 항공학적으로 표현방법 파악
오일러 각을 이용한 관성좌표계와 구면좌표계사이에 회전행렬과 변환행렬을 이용하여 변환하는 메커니즘을 공부..

1주차 문제점 : 좌표계 변환 과정이 가장 힘들었고 이해도 하기 힘들었다. 반복학습으로 인해 이해하는 방법밖에 없었다.

\mathbf{C} 는 관성 좌표계의 오일러 각의 속도성분과 기체 좌표계의 각속도 벡터간의 관계를 표현하는 행렬로 다음 관계식으로부터 유도할 수 있다.

$$\mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi)\begin{bmatrix}\omega_x \\ \omega_y \\ \omega_z\end{bmatrix} = \begin{bmatrix}0 \\ 0 \\ \dot{\psi}\end{bmatrix} + \mathbf{R}_z(\psi)\begin{bmatrix}0 \\ \dot{\theta} \\ 0\end{bmatrix} + \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\begin{bmatrix}\dot{\phi} \\ 0 \\ 0\end{bmatrix}$$

$$\begin{bmatrix}\omega_x \\ \omega_y \\ \omega_z\end{bmatrix} = \begin{bmatrix}\dot{\phi} \\ 0 \\ 0\end{bmatrix} + \mathbf{R}_x^T(\phi)\begin{bmatrix}0 \\ \dot{\theta} \\ 0\end{bmatrix} + \mathbf{R}_x^T(\phi)\mathbf{R}_y^T(\theta)\begin{bmatrix}0 \\ 0 \\ \dot{\psi}\end{bmatrix}$$

$$\begin{bmatrix}\omega_x \\ \omega_y \\ \omega_z\end{bmatrix} = \mathbf{C}\begin{bmatrix}\dot{\phi} \\ \dot{\theta} \\ \dot{\psi}\end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix}1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta\end{bmatrix}$$

\mathbf{R} 은 관성 좌표계에 대하여 기체 좌표계를 회전 변환하는 행렬로 다음과 같이 정의된다.

$$\mathbf{R} = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) \\ = \begin{bmatrix}\cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta\end{bmatrix}$$

$$\mathbf{R}_x(\phi) = \begin{bmatrix}1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi\end{bmatrix}, \quad \mathbf{R}_y(\theta) = \begin{bmatrix}\cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta\end{bmatrix},$$

$$\mathbf{R}_z(\psi) = \begin{bmatrix}\cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1\end{bmatrix}$$

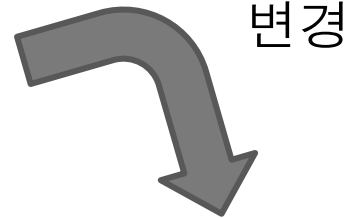
진행상황 - 5주차

2016/6/27 ~ 28 : 관성좌표계와 기체좌표계에 대한 x, y, z, roll, pitch, yaw, 각각의 축으로 이동하는 속도를 수식으로 계산하여 병합시키는 과정

2016/6/29 ~ 7/01: C 코드로 변환하며 오차 수정, PID 설계 후 MATLAB으로 확인 후 C코드 PID 제어 추가

2주차 문제점 : 모델링에 대한 자료는 수집하는 과정에서 많이 나와있었지만 이것을 직접 증명하는 과정에서 계산 오류, C 코드로 변경하는 과정에서 오류, 적분오류가 발생하였다. 작은 오차에 대한 무관심으로 인한 결과였다.

```
// Body angular Velocity Intergral _ make p,q,r //  
if(_body.p == 0 && _body.q == 0 && _body.r == 0)  
{  
    _body.p = _body.pDot*dt/2;  
    _body.q = _body.qDot*dt/2;  
    _body.r = _body.rDot*dt/2;  
    printf("pqr_in\n");  
}  
else  
{  
    _body.p = _body.p + _body.pDot*dt;  
    _body.q = _body.q + _body.qDot*dt;  
    _body.r = _body.r + _body.vDot*dt;  
}
```



```
// Body angular Velocity Intergral _ make p,q,r //  
if (_body.pDot > _body.Old_pDot)  
    _body.p = _body.p + _body.Old_pDot*dt + (_body.pDot - _body.Old_pDot)*dt / 2;  
else  
    _body.p = _body.p + _body.Old_pDot*dt - (_body.Old_pDot - _body.pDot)*dt / 2;  
  
if (_body.qDot > _body.Old_qDot)  
    _body.q = _body.q + _body.Old_qDot*dt + (_body.qDot - _body.Old_qDot)*dt / 2;  
else  
    _body.q = _body.q + _body.Old_qDot*dt - (_body.Old_qDot - _body.qDot)*dt / 2;  
  
if (_body.rDot > _body.Old_rDot)  
    _body.r = _body.r + _body.Old_rDot*dt + (_body.rDot - _body.Old_rDot)*dt / 2;  
else  
    _body.r = _body.r + _body.Old_rDot*dt - (_body.Old_rDot - _body.rDot)*dt / 2;  
  
_body.Old_pDot = _body.pDot;  
_body.Old_qDot = _body.qDot;  
_body.Old_rDot = _body.rDot;
```

진행상황 - 6주차

2016/6/27 ~ 28 : PID 설계 하면서 같이 시작했던 부분인데 생각보다 어렵지 않았고 강사의 수업 중에 설명 해주셨던 부분을 따라가면서 다시 정리. 모터의 스펙을 구할 수 없어서 모델링까지만 하고 C 코드로 변환하지 못함

motor

$$T = k_t \dot{\theta} \quad C = k_e \dot{\theta}$$

$$P = \frac{dW}{dt} = \frac{d(\tau \cdot \theta)}{dt} = \tau \cdot \frac{d\theta}{dt} = \tau \cdot \omega \quad \tau = m a = m \frac{dv}{dt}$$

$$E = \int \tau \omega dt = k_t \int \omega dt = \frac{1}{2} k_t \omega^2 \quad \phi = \frac{q}{C}, V = \frac{q}{C} \quad E = \int V d\phi = \frac{1}{C} \int q d\phi = \frac{1}{2C} \frac{q^2}{C} = \frac{1}{2} C V^2$$

$$T = 2\tau \tau = 2\tau \tau_a L B$$

$$T = \frac{2}{\pi} \phi \tau$$

$$T = k_e \dot{\theta}$$

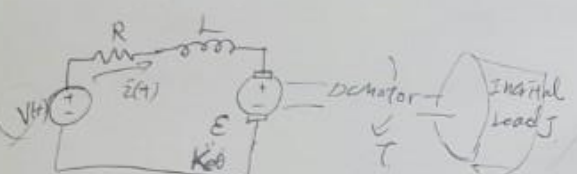
$$E_a = 2B l v \quad \phi = (v \times B) \cdot l$$

$$E = - \frac{d\phi}{dt} = \frac{B l v}{dt} = \frac{B l dv}{dt} = B l v$$

$$v = r \omega \quad \phi = \pi r l B$$

$$B = \frac{\phi}{\pi r l}$$

$$E_a = 2B l v = 2 \pi l r \omega = 2 \pi l r \frac{\phi}{\pi r l} = \frac{2 \phi \omega}{\pi}$$

$$E_a = k_e \omega \quad k_e = \frac{2 \phi}{\pi}$$


$$V(t) = R i(t) + L \frac{di(t)}{dt} + k_e \dot{\theta}(t)$$

$$V(t) = L \ddot{\theta}(t) + R \dot{\theta}(t) + k_e \omega(t)$$

$$V(s) = L s I(s) + R I(s) + k_e \omega(s)$$

$$-T(s) + J s \omega(s) + k_f \omega(s) = 0$$

$$\Rightarrow J s \omega(s) + k_f \omega(s) - k_e I(s) = 0$$

$$J \ddot{\theta}(t) + k_f \dot{\theta}(t) = T(t)$$

$$J \dot{\omega}(t) + k_f \omega(t) = T(t)$$

$$T = k_e \dot{\theta}$$

$$\begin{bmatrix} Ls + R & k_e \\ -k_e & Js + k_f \end{bmatrix} \begin{bmatrix} I(s) \\ \omega(s) \end{bmatrix} = \begin{bmatrix} V(s) \\ 0 \end{bmatrix}$$

$$\omega(s) = \frac{\det \begin{bmatrix} Ls + R & V(s) \\ -k_e & 0 \end{bmatrix}}{\det \begin{bmatrix} Ls + R & k_e \\ -k_e & Js + k_f \end{bmatrix}} = \frac{+ V(s) k_e}{(Ls + R)(Js + k_f) + k_e k_e + k_f R}$$

$$\frac{\omega(s)}{V(s)} = \frac{k_e}{(Ls + R)(Js + k_f) + k_e k_e + k_f R}$$

Motor
Voltage - Angle velocity

진행상황 - 7주차

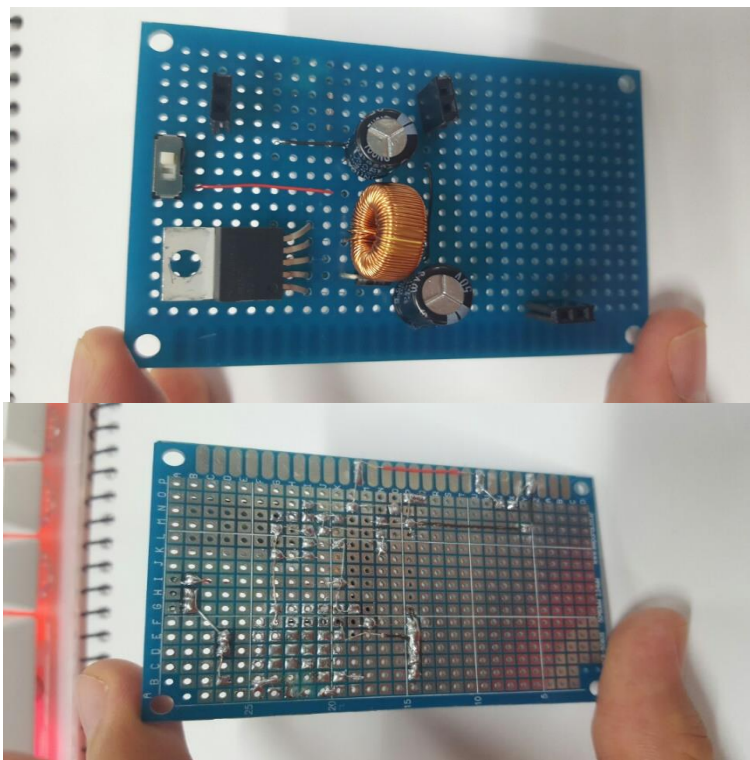
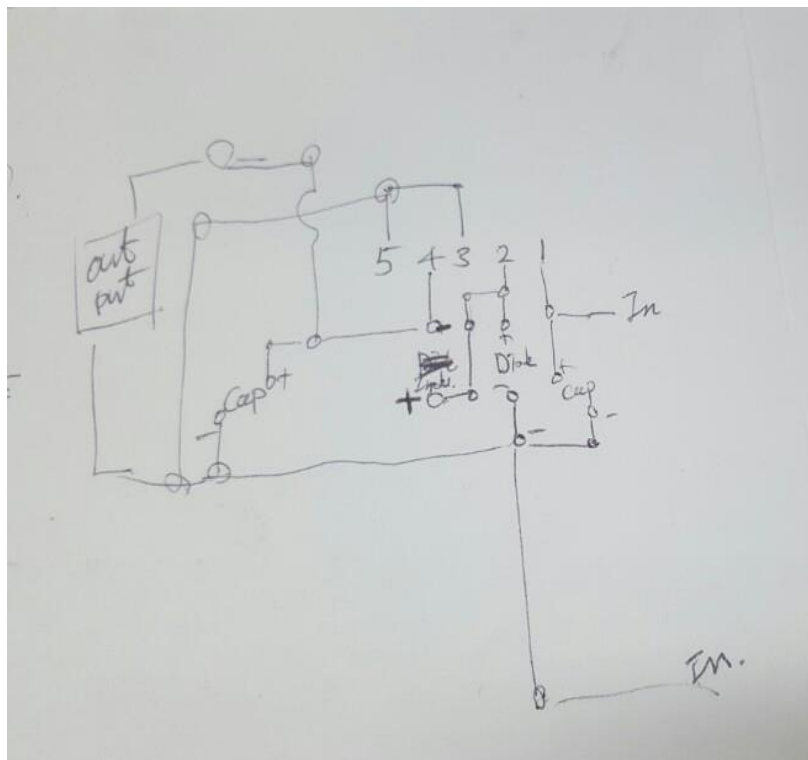
2016/6/29 ~ : Altium Design을 활용하여 만들어 보았으나 PCB 작업에 많은 돈이 들어가기에 Tool 사용하는 방법만 익히는 것으로 마무리 하였고 브레드보드에 3.3, 5, 12V 전원 설계를 위한 첫 실험에 성공한 후 현재 만능기판에 설계중, 3.3V 전원설계완료(7/5)

2016/07/06 : 아두이노코드를 이용하여 BLDC모터테스트 성공, ESC Calibration 함수 코드 분석 및 CCS 로 코드작성

2016/07/07 : R5로 ESC Calibration 코드 작성중...

2016/07/08 : 12V 전원회로설계, R5보드로 BLDC모터제어 테스트 성공

문제점 : ESC calibration이 제대로 되지 않아 아두이노로 밖에 제대로된 테스트를 할 수 없었다. ESC마다 설정이 다른 이유 때문이기도 하므로 조금 더 삽질을 해봐야 할 것 같다.



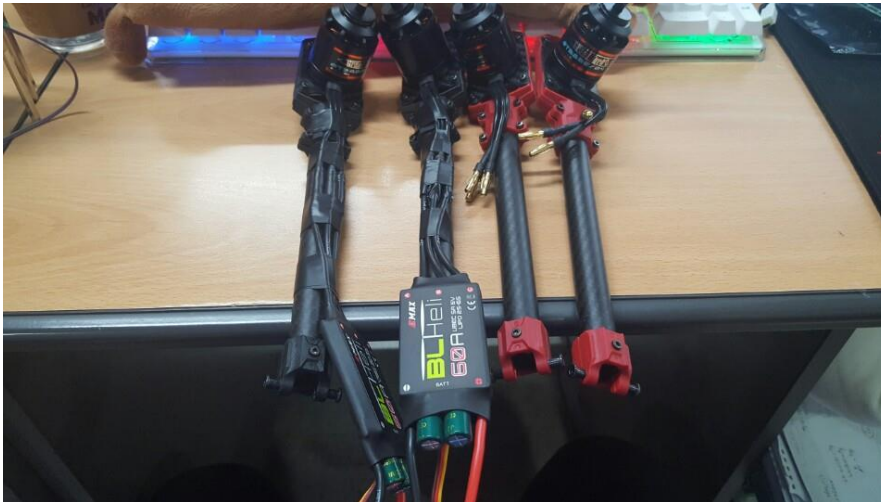
진행상황 - 8주차

2016/07/11 : 5V 1A,3A 전원회로설계, R5로 ESC calibration 및 모터제어완료

현재까지 문제점 : 전원회로는 모두 완성되었으나 9V 배터리로는 전류가 금방 부족해져 센서들이 10분이상 지속되다 보면 리셋이 되어버림... 고민해야댐....

2016/07/12 ~ : 프레임 구상 및 조립 중..., 전원회로문제는 쿼드콥터 프레임에 달려있는 전원분배회로를 통해 4Cell짜리 배터리를 하나더 구입해서 만들어 놓은 전원회로에 연결 할 예정.

쿼드콥터의 x,y,z 모멘트를 계산하여 소스에 적용, PD제어기 값 변경



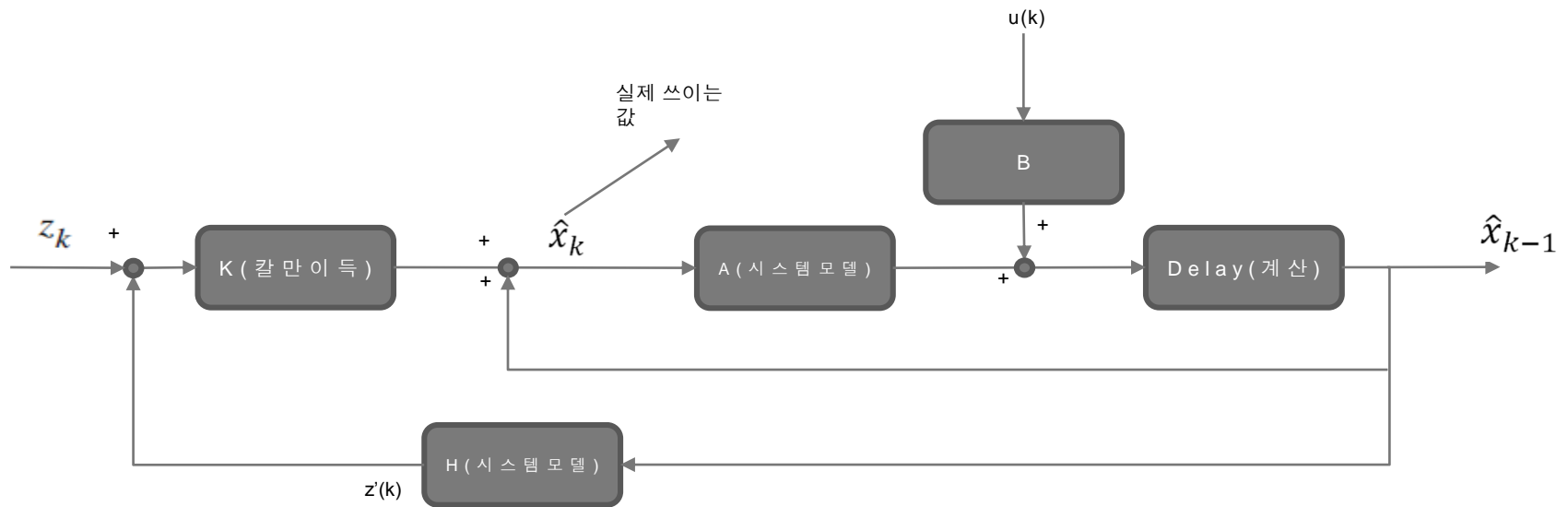
일단 두개의 ESC를 연결하였고 나머지 두개는 호버링 시험 후 연결할 예정

#define I_xx	0.0086	// 쿼드콥터 x의 관성모멘트	kg m^2	#define I_xx	0.030392303	// 쿼드콥터 x의 관성모멘트	kg m^2
#define I_yy	0.0086	// 쿼드콥터 y의 관성모멘트	kg m^2	#define I_yy	0.031480833	// 쿼드콥터 y의 관성모멘트	kg m^2
#define I_zz	0.0172	// 쿼드콥터 z의 관성모멘트	kg m^2	#define I_zz	0.057060711	// 쿼드콥터 z의 관성모멘트	kg m^2
#define L	0.225	// 쿼드콥터의 한 쪽 길이	m	#define L	0.33	// 쿼드콥터의 한 쪽 길이	m

변경되지 않은 모멘트 값

변경된 모멘트 값

칼만필터 Block diagram



-칼만필터의 개념

칼만필터는 대상 시스템의 확률적인 모델과 측정값을 이용하여 시스템의 상태변수를 찾아내는 최적 추정기법이다.

즉, 상태식에 의한 시간전파와 측정식에 의한 개선을 통하여 상태값을 추정하게 되며, 실제 상태값과 추정된 상태값의 오차 공분산을 최소화하는 알고리즘이다.

시스템이 선형이고 정규분포를 갖는 백색잡음에 의해 구동되는 칼만필터는

최소 공분산을 갖는 편향되지 않는 최적의 추정기로 알려져 있다.

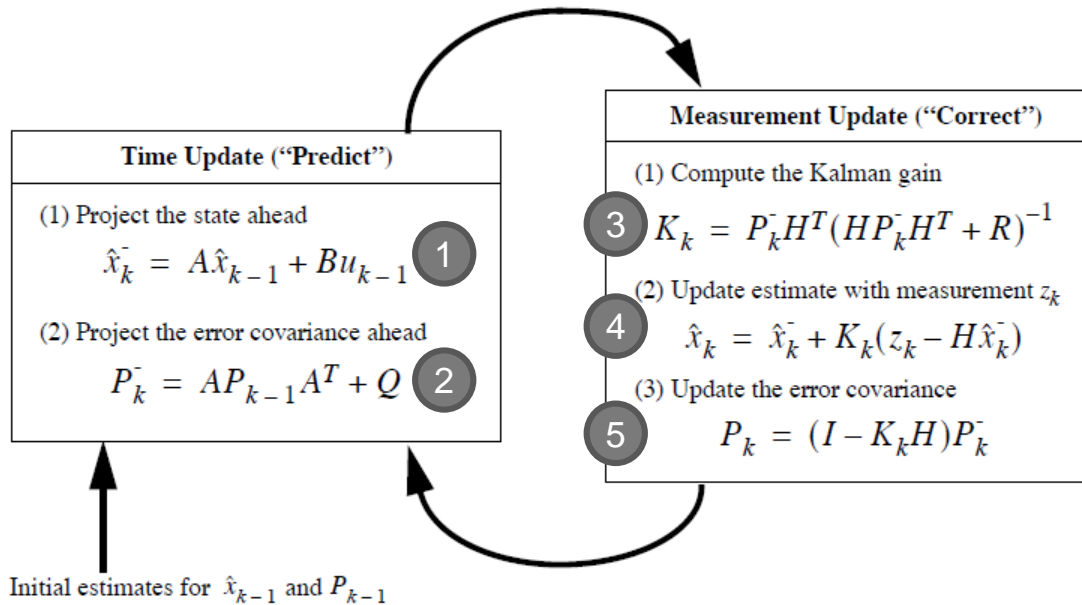
칼만필터는 추정값에 대한 확률분포를 따져서 가장 확률이 높은 값을 추정값으로 선택하는 방법으로 측정값의 예측 오차로 예측 값을 적절히 보정해서 최종 추정값을 계산 칼만이득을 통한 각각의 결과로 나타나는 상태를 위한 가장 최적의 평균 인자를 찾는 것이 칼만필터의 목표이다.

완벽한 이해를 위해 필요한 개념

기댓값, 평균, 표준편차, 분산, 공분산, 확률, 확률분포, 분포함수,
베이지의 정리, 누적분포함수, 확률밀도함수, 조건부 확률밀도함수

https://ko.wikipedia.org/wiki/%EC%B9%BC%EB%A7%8C_%ED%95%84%ED%84%B0#.EC.B9.BC.EB.A7.8C_.ED.95.84.ED.84.B0.EC.9D.98_.EC.A0.81.EC.9A.A9_.EB.B6.84.EC.95.BC
참조문헌-위키

1,2,3,4,5는 실제 코드상에서의 흐름



변환과정

$$\hat{x}_{k-1}^- \Rightarrow \hat{x}_{k-1} \Rightarrow \hat{x}_k^- \Rightarrow \hat{x}_k$$

보정되지않은 k-1 보정된 k-1 보정되지않은 k 보정된 k

A,B,H,R : system matrix

R : 측정 노이즈 공분산

Q : 예측 노이즈 공분산(프로세스 노이즈 예측)

H : time step에 따른 변화가능한 매트릭스

u_k : 측정값(자이로 각속도)

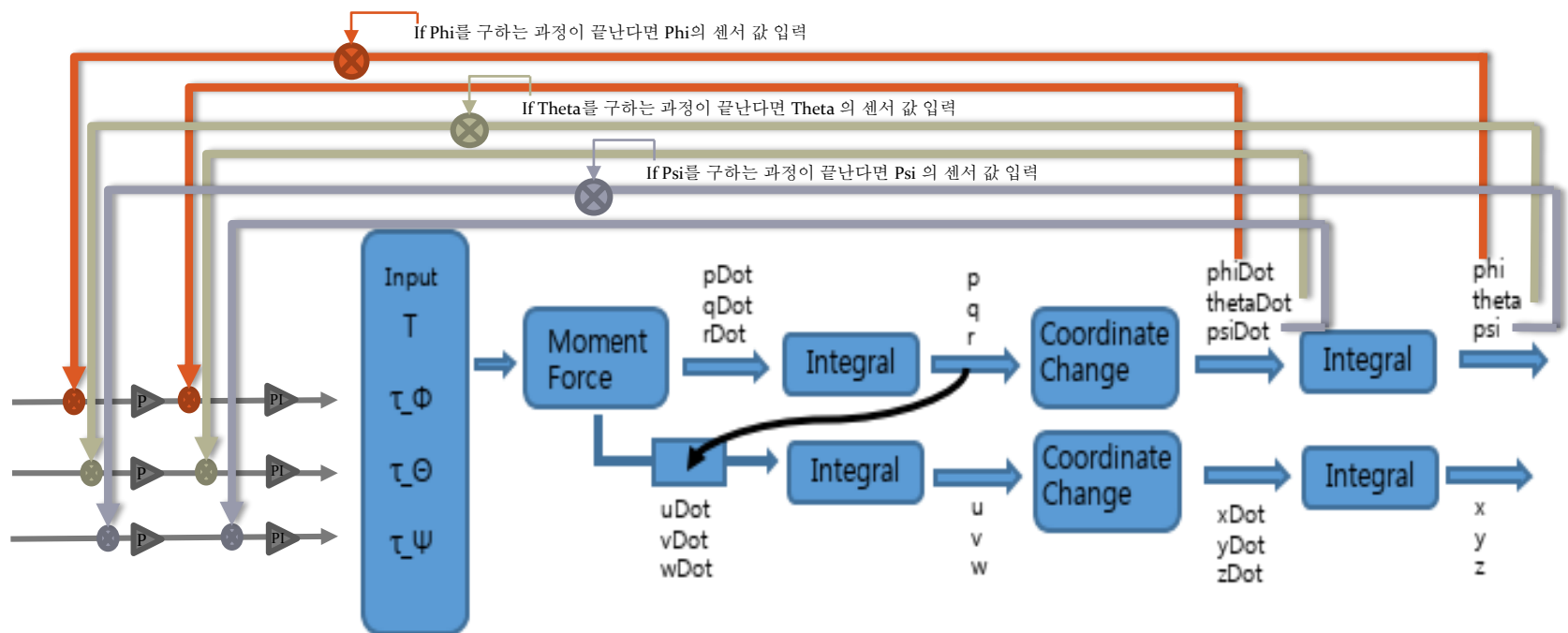
z_k : 측정값(가속도 각도)

P_k : 예측 에러 공분산

A : 해당 시간에서 이전 상태에 기반한 상태 전이 행렬

B : 사용자 입력에 의한 상태전이행렬

자세제어 모델



Total thrust $\vec{F}_T = k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2)$, $k = 3.3 \times 10^{-4}$ N/rpm²
 roll moment $M_x = Lk(\omega_3^2 - \omega_1^2)$
 pitch $M_y = Lk(\omega_2^2 - \omega_4^2)$
 yaw $M_z = Lk(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2)$

Roll $\ddot{\phi} = \frac{1}{I_x} \left[\frac{1}{2} \frac{Lk}{m} u_1 \right]$, Pitch $\ddot{\theta} = \frac{1}{I_y} \left[\frac{1}{2} \frac{Lk}{m} u_2 \right]$, Yaw $\ddot{\psi} = \frac{1}{I_z} \left[\frac{1}{2} \frac{Lk}{m} u_3 \right]$

Motion equation

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\theta & 0 \\ s_\psi c_\theta & c_\psi c_\theta & 0 \\ s_\theta & c_\theta & 1 \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} -\dot{\psi} s_\theta & -\dot{\psi} c_\theta & 0 \\ \dot{\phi} c_\psi c_\theta + \dot{\theta} s_\psi c_\theta & -\dot{\phi} s_\psi c_\theta + \dot{\theta} c_\psi c_\theta & 0 \\ \dot{\phi} s_\psi c_\theta + \dot{\theta} c_\psi c_\theta & -\dot{\phi} c_\psi c_\theta + \dot{\theta} s_\psi c_\theta & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} + \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\theta & 0 \\ s_\psi c_\theta & c_\psi c_\theta & 0 \\ s_\theta & c_\theta & 1 \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix}$$

$\vec{v} = \vec{v}_0 + \vec{\omega} \times \vec{r}$

$\frac{d(I\vec{\omega})}{dt} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = I \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} + \vec{\omega} \times (I\vec{\omega})$, $\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = I^{-1} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + I^{-1} (I\vec{\omega}) \times \vec{\omega}$

$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$, $I_{xx} = I_{zz}$

$\ddot{\phi} = -\dot{\psi} \dot{\theta} c_\theta + \frac{c_\psi}{I_{xx}} u_1 - \frac{s_\psi}{I_{zz}} u_3 + \frac{I_{yy} I_{zz}}{I_{xx}} (\dot{\psi} - \dot{\theta} s_\theta) \dot{\theta} c_\theta$

$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = I^{-1} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + I^{-1} (I\vec{\omega}) \times \vec{\omega} + I^{-1} \vec{G}_a$

$\ddot{\theta} = \frac{s_\psi}{I_{yy}} \dot{\phi} + \dot{\phi} \dot{\theta} c_\theta + \frac{s_\psi}{I_{zz}} u_2 + \frac{c_\psi}{I_{xx}} u_1 - \frac{I_{yy} I_{zz}}{I_{xx}} (\dot{\psi} - \dot{\theta} s_\theta) \dot{\theta} c_\theta$

$\vec{G}_a = I_r(\vec{\omega} \times \vec{e}_z) \omega_{m-d}$

$\omega_{m-d} = \omega_{m-2} + \omega_{m-1} - \omega_{m-1} - \omega_{m-2}$

$\ddot{\psi} = \dot{\phi} \dot{\theta} s_\theta + \frac{\dot{\phi} \dot{\theta}}{I_{xx}} + \frac{s_\psi c_\theta}{I_{yy}} u_2 + \frac{c_\psi c_\theta}{I_{zz}} u_1 + \frac{1}{I_{zz}} u_3 - \frac{I_{yy} I_{zz}}{I_{xx}} (\dot{\psi} - \dot{\theta} s_\theta) \dot{\theta} s_\theta$

$\dot{\omega}_m = -\frac{1}{T} \omega_m - \frac{d}{\eta r^3 J} \omega_m^2 + \frac{1}{k_m T} u$, $\frac{1}{T} = \frac{k_m^2}{RJ}$

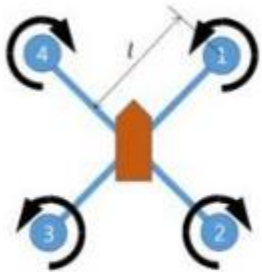
ω_m = 모터 각속도, u = 모터 입력, T = 모터 시정수, k_m = 코크 상수
 d = 항력계수, η = 기어비 손실 계수, r = 기어비, J = 프로그래머 관성
 R = 모터 전압 저항

Torque: $\tau_i = \frac{T_i (V_c + V_i) + 0.125 \rho b C_p^4 \omega_m^3}{\omega_{m-1}}$

T_i = i 블레이드 thrust acting, V_c = forward velocity, V_i = induced velocity, ρ = 공기 밀도
 b = 프로펠러 폭, C = 프로펠러 chord, R_p = 프로펠러 반경, C_d = 항력 계수

Angle $\begin{pmatrix} u_2 \\ u_3 \\ u_4 \end{pmatrix} \rightarrow \begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ Translation

X형 모델로 바꾸기 위한 모델링 변환



(x type)

$$\begin{bmatrix} T \\ L \\ M \\ N \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ -l_1 & -l_1 & l_1 & l_1 \\ l_1 & -l_1 & -l_1 & l_1 \\ K_Q & -K_Q & K_Q & -K_Q \end{bmatrix}}_{Q_{eq}} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix}$$

T : 전체 추력 $l_1 = l \sin 45$

L : Roll 모멘트

M : Pitch 모멘트

N : Yaw 모멘트

T1 : 모터 1

T2 : 모터 2

T3 : 모터 3

T4 : 모터 4

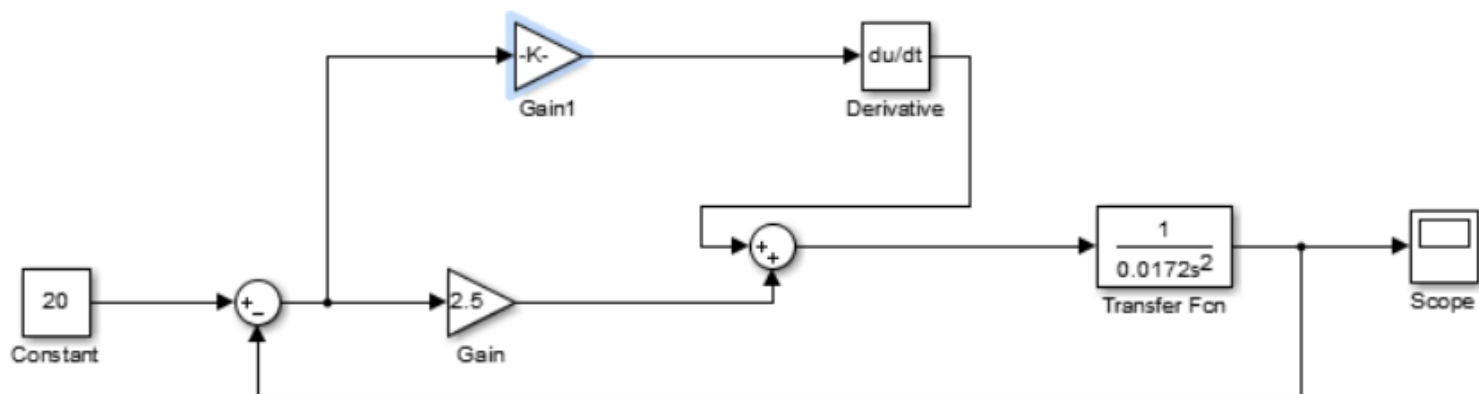
K_Q : 추력상수

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & -\frac{1}{2l \sin 45} & \frac{1}{2l \sin 45} & \frac{1}{4Kq} \\ \frac{1}{4} & -\frac{1}{2l \sin 45} & -\frac{1}{2l \sin 45} & -\frac{1}{4Kq} \\ \frac{1}{4} & \frac{1}{2l \sin 45} & \frac{1}{2l \sin 45} & \frac{1}{4Kq} \\ \frac{1}{4} & \frac{1}{2l \sin 45} & -\frac{1}{2l \sin 45} & -\frac{1}{4Kq} \end{bmatrix} \begin{bmatrix} T \\ L \\ M \\ N \end{bmatrix}$$

PD 제어

$$G(s) = \frac{G_{loop}(s)}{1 + G_{loop}(s)}$$

MATLAB Simulink PD 제어도



계산 후 넣은 PD제어상수 및 코드

```
double KpP = 36, KdP = 1.563;  
double KpR = 37, KdR = 1.62;  
double KpY = 55, KdY = 2.5;
```

```
if (flag_Theta)  
{  
    errP = _inertial.theta + _sensor.Theta_rad;  
    KpP_term = KpP * errP;  
  
    D_errP = (errP - prev_errP) / dt;  
    KdP_term = KdP * D_errP;  
  
    prev_errP = errP;  
    enterTauTheta = -(KpP_term + KdP_term);  
  
}  
  
if (flag_Phi)  
{  
    errR = _inertial.phi + _sensor.Phi_rad;  
    KpR_term = KpR * errR;  
  
    D_errR = (errR - prev_errR) / dt;  
    KdR_term = KdR * D_errR;  
  
    prev_errR = errR;  
    enterTauPhi = -(KpR_term + KdR_term);  
  
}  
  
if (flag_Psi)  
{  
    errY = _inertial.psi + _sensor.Psi_rad;  
    KpY_term = KpY * errY;  
  
    D_errY = (errY - prev_errY) / dt;  
    KdY_term = KdY * D_errY;  
  
    prev_errY = errY;  
    enterTauPsi = -(KpY_term + KdY_term);  
  
}
```

목번호

CAN 통신

문제점

**2016/05/29 ~ 2016/06/15 : Ethernet to CAN 관련하여 R5쪽
CAN통신 데이터 시트 공부 및 모듈 테스트 실험 및 구현**

2016/06/16 : Ethernet to CAN 컨버터-R5간의 연결 문제 발생
컨버터 **MFC** 프로그램의 데이터 오고 가는 부분에서 데이터의 확인을 못하여 다른 컨버터를 구매

**2016/06/17 : 다른 Ethernet to CAN 컨버터를 이용하여 R5에서
컨버터의 연결 확인 R5쪽에서 컨버터를 통해 데이터를 보내는 것을
확인하였으나, PC(이더넷)쪽에서 보내는 데이터를 R5쪽에서 확인이 어려웠음.**

2016/06/20 : PC(이더넷)쪽에서 보내는 데이터를 R5에서 확인.

R5-CAN 컨버터-PC 이더넷 통신 확인

다음은 모범 사례다.

CAN 통신

문제점

2016/06/25 : R5-SVDK CAN 통신 구현중 SVDK쪽에 이더넷 연결은 되는데 컨버터쪽으로 연결하는 방법이 해결 안됨.

2012/06/29 : SV아 쪽에서 지속적으로 확인중 있지만 컨버터로 연결을 하고 데이터를 어떤식으로 보내는지 확인을 해야하는데 방법을 찾아야함.

2016/07/04 : R5-SVDK CAN 통신을 할 수 있도록 SVDK 이더넷 부분 해결하려고 하는중.

다음은 모범 사례다.

RF

목현호

문제점

2016/07/05 : RF(CC1200)에 HC-SR501(인체감지센서), AM2302(온습도 센서) 연결 시도 중이며, CC1200에서 MSP430 Breakout I/O 이 부분에 센서들을 어떻게 연결할지 구상중이며

현재 MSP430 breakout I/O 핀 중에서 사용하지 않는 GPIO 핀들을 통해 센서를 제어할 생각이며 센서 VCC / GND은 breakout I/O 부분에서 제공되는 포트를 사용할 계획.'

2016/07/07 : RF(CC1200) MSP430 Breakout I/O 포트 핀 헤더를 이용해 핀 활성화 및 RF(CC1200) LCD 디스플레이 부분 센서 값 출력 관련 공부

2016/07/08 : RF(CC1200) EasyLink Test 부분을 이용하여 MSP430 Breakout I/O Port 사용 관련 공부.

다음은 모범 사례다.

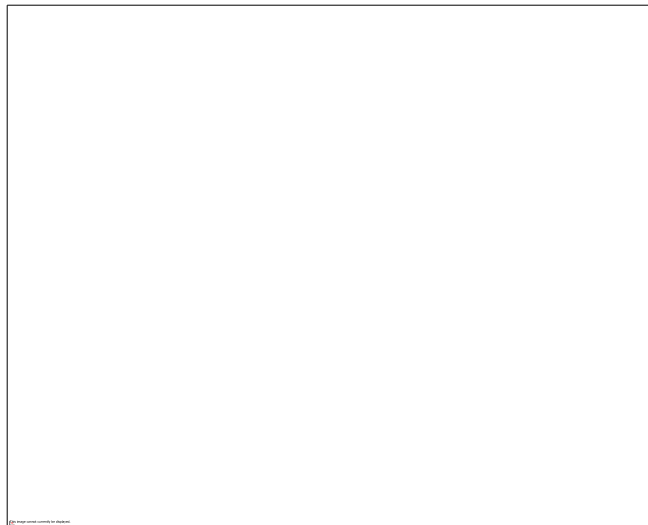
RF

목현호

문제점

2016/07/11 : RF(CC1200) EasyLink Test 부분을 이용하여 MSP430 Breakout I/O Port 사용 관련 공부.

문제점 Breakout I/O에 센서를 연결 후 사용하고자 하는 Port를 설정한 뒤에 센서가 작동을 하고 있는데, 감지를 하게 된다면 이 값을 패킷을 통해 전달을 하고 CC1200의 LCD에 출력을 하고 싶지만 제대로 되고 있지가 않고 있음. 해결방안은 센서 output 관련 부분과 이 output value을 담은 패킷이 출력되는 디버깅하여 수정할 계획입니다.



다음은 모범 사례다.

RF

목현호

문제점

2016/07/11 : RF(CC1200) EasyLink Test 부분을 이용하여 MSP430 Breakout I/O Port 사용 관련 공부.

Port 연결 후 패킷이 전송되면 LED On 전송이 안되면 Off
PIR 센서를 연결 후 감지가 되면 패킷 값이 +1 or detected
PIR 센서를 연결 후 감지가 안되면 패킷 값 0 or ended
LCD 출력 부분은 해결 / PIR output 관련 수정이 필요

