지능형 자율주행 RC Tank

- 재미난 사람들-

P M	이상훈
P L	강삼민
팀원	서영일
	이건수
	이　찬
	이충환
	장세진

목차

- 프로젝트 개요
- 팀원 소개 & 역할
- 적용 기술
- 진행 계획
- 진행 내용
- 문제점
- 시연 영상 & 시연
- Q & A

프로젝트 개요

간단한 프로젝트 소개 - 나중에

역할

강삼민	서영일	이건수	이 찬	이충환	장세진
Device Driver (V4L2)	Firmware	Firmware	Firmware	Image Processing	Image Processing
CAN	Encoder	Circuit Design	CAN	Object Detection	Lane Detection
AirGun Design	Motor Control	RTOS	Motor Control	Parallel Processing	Kalman Filter
				AirGun Design	CAN

적용 기술

Language	C, C++
OS	Linux, RTOS
Board	TI AM5728(Cortex-A + C66 DSP + SGX) Heterogeneous DSP, TI MCU Cortex-R5F
Etc	CCS, OpenCV, OpenCL

계획

	1주	2주	3주	4주	5주	6주	7주	8주
강삼민	Device Driver							
			CAN					
	AirGun Design							
서영일	ePWM	ESC Calibration	Servo Motor	Encoder				
		Motor Control						
이건수	Circuit Design							
			RTOS					
이 찬	CAN							
		Motor Modeling		PI Control		Motor Control		
이충환	Object Detection							
				Parallel Processing				
	AirGun Design							
장세진	Lane Detection							
				Kalman Filter				
						CAN		

진행 내용

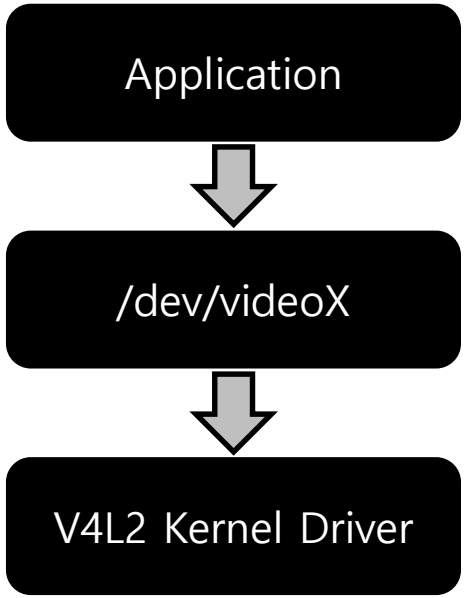
Device Driver
CAN

강삼민

진행 내용 – V4L2

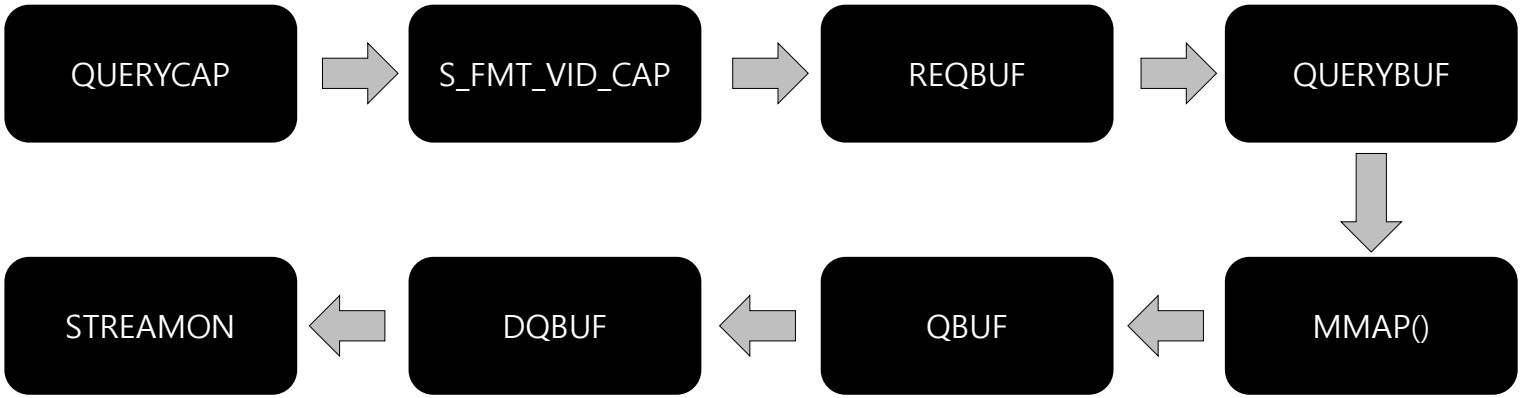
- **Video For Linux Ver. 2(V4L2)**
  - PCI Camera, USB Camera 등 다양한 비디오 디바이스를 위한 Kernel API
  - 사용자 프로그램에서 사용하는 API와 디바이스 드라이버를 위한 API가 존재

- **V4L2 드라이버 동작 과정**
  - 사용자 S/W에서 카메라 디바이스 접근
  - 접근 시 요청한 명령에 따라 V4L2 동작



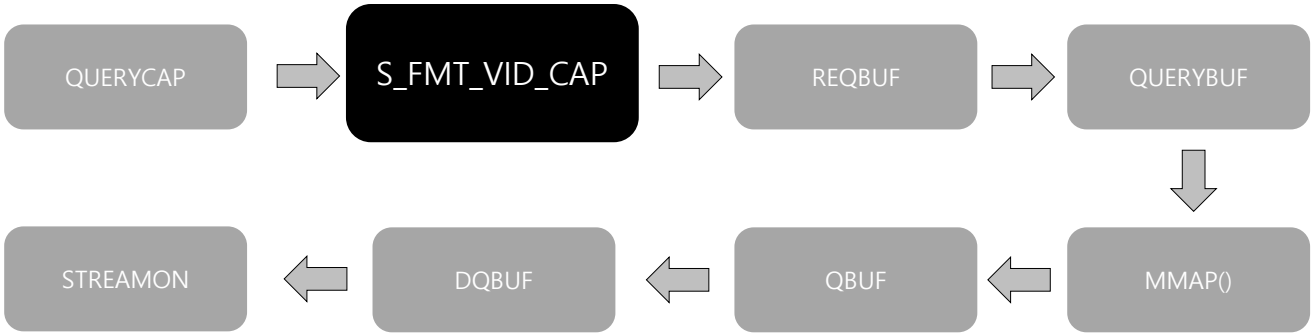
진행 내용 – V4L2

- **V4L2 드라이버 동작 과정**
  - 사용자가 아래 순서로 비디오 디바이스 드라이버에 요청
  - V4L2 Driver는 사용자가 보내온 명령을 차례로 수행
  - 해당 동작 순서 및 함수들은 비디오 디바이스를 동작시키기 위한 최소한의 필요 요소



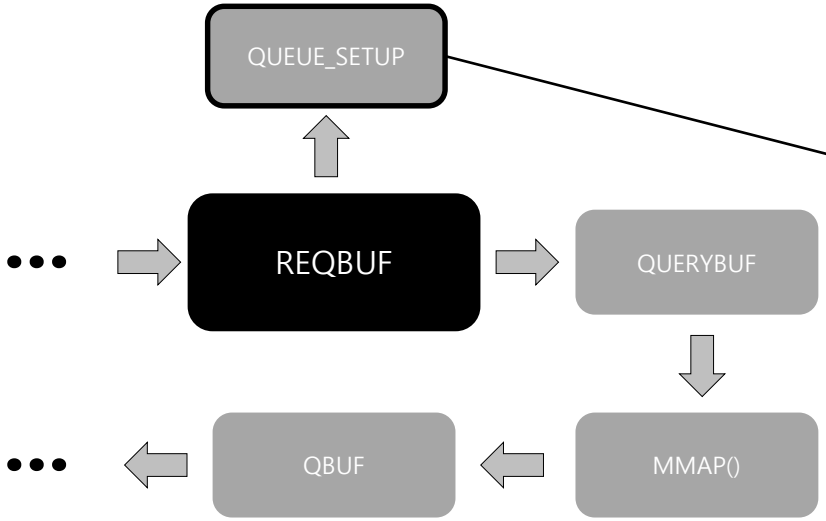
진행 내용 – V4L2

- **V4L2 드라이버 동작 과정 - S_FMT_VID_CAP**
  - 사용자가 요청한 포맷을 기반으로 비디오 디바이스를 설정
  - 해상도, 색상, 이미지 포맷 등 비디오 디바이스 자체에 값을 설정
  - 현재 미구현



진행 내용 – V4L2

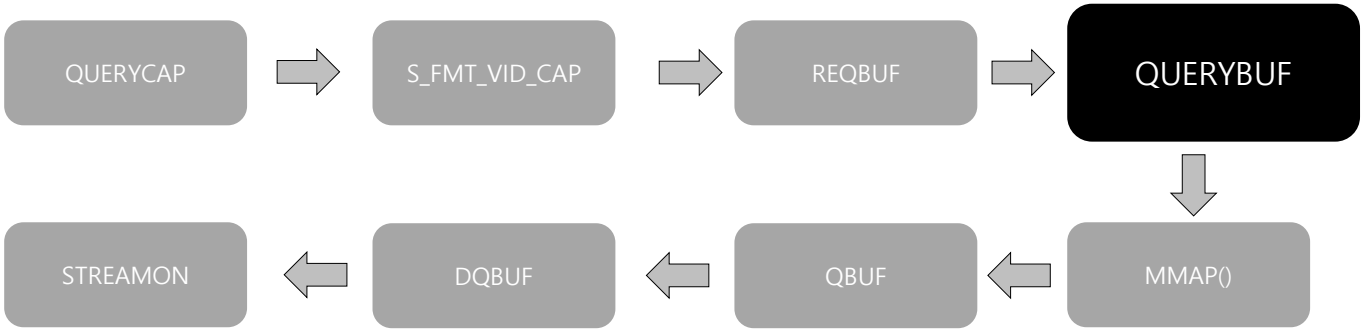
- **V4L2 드라이버 동작 과정 – REQBUF**
  - 아직 정확한 동작 원리 및 역할 분석 미비
  - REQBUF 함수가 실행되기 전 QUEUE_SETUP 함수를 호출하여
    비디오 디바이스로부터 받은 정보를 저장할 QUEUE에 대한 설정 진행




```
static int t_queue_setup(struct vb2_queue *vq, const struct v4l2_format *f
{

        struct vb2_queue *v_q = vb2_get_drv_priv(vq);
        thirdy_t *t_struct = container_of(v_q, thirdy_t,t_viq);
        fmt = t_struct->v4l2_fmt;

        if(fmt && fmt->fmt.pix.sizeimage < t_struct->t_data->dwMaxVideoFra
                return -EINVAL;

        *nplanes = 1;

        sizes[0] = fmt ? fmt->fmt.pix.sizeimage : t_struct->t_data->dwMaxV
        printk("Thirdy_Driver - t_queue_setup OK / %d\n", *nbuffers);

        return 0;
}
```

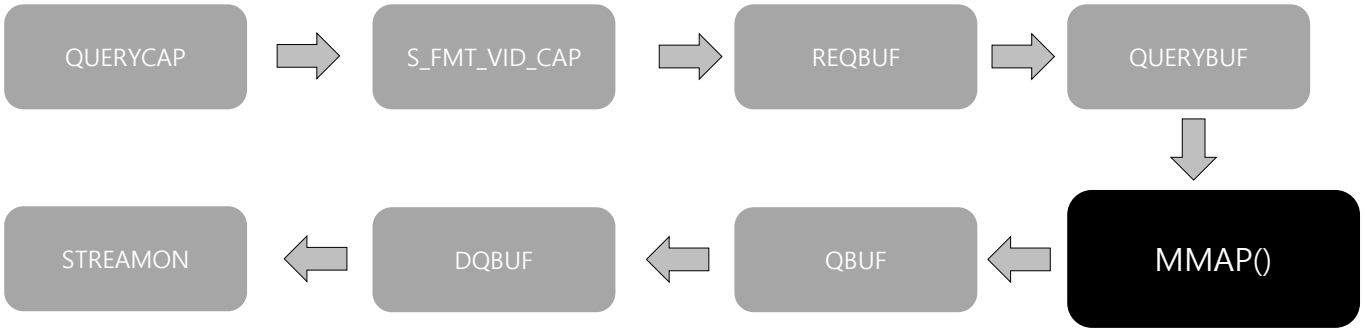
진행 내용 – V4L2

- **V4L2 드라이버 동작 과정 – QUERYBUF**
  - 아직 정확한 동작 원리 및 역할 분석 부족



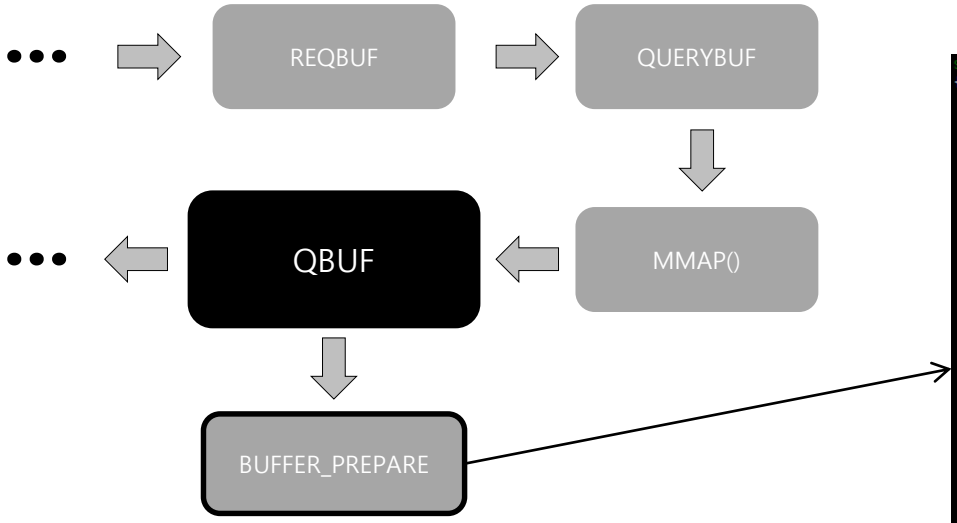
진행 내용 – V4L2

- **V4L2 드라이버 동작 과정 – MMAP()**
  - MMAP 설명 쓰기



진행 내용 – V4L2

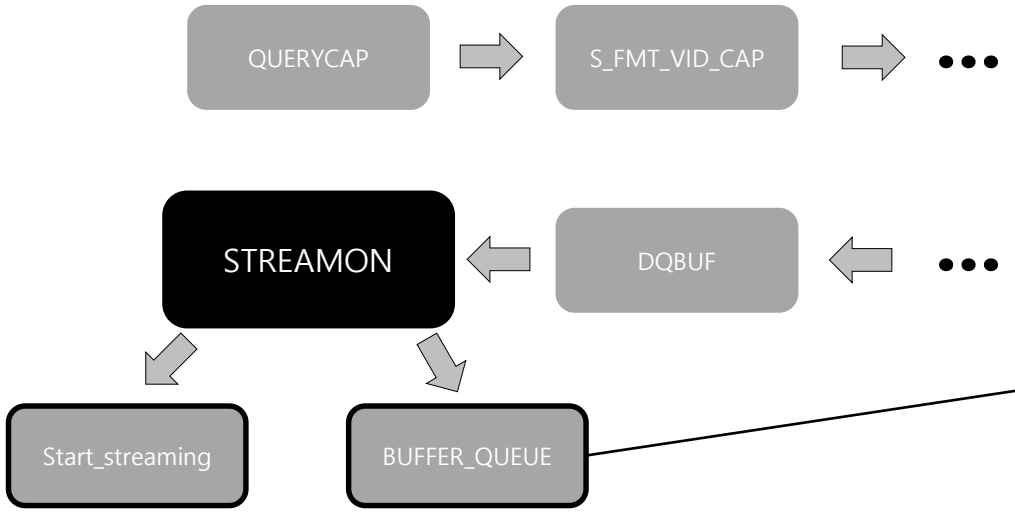
- **V4L2 드라이버 동작 과정 – QBUF**
  - 아직 정확한 동작 원리 및 역할 분석 부족
  - QBUF의 경우 함수 동작 전 BUFFER_PREPARE 함수를 호출하여 데이터를 저장하고 QUEUE에 쌓일 BUFFER를 준비




```
static int t_buffer_prepare(struct vb2_buffer *vb)
{
        struct vb2_queue *v_q = vb2_get_drv_priv(vb->vb2_queue);
        thirdy_t *t_struct = container_of(v_q, thirdy_t, t_vig);
        struct t_buffer *v_bf = container_of(vb, struct t_buffer, buf);


        if(vb->v4l2_buf.type == V4L2_BUF_TYPE_VIDEO_OUTPUT && vb2_get_plane_
                return -EINVAL;

        v_bf->state = 1;
        v_bf->error = 0;
        v_bf->mem = vb2_plane_vaddr(vb, 0);
        v_bf->length = vb2_plane_size(vb, 0);

        if(vb->v4l2_buf.type == V4L2_BUF_TYPE_VIDEO_CAPTURE)
                v_bf->bytesused = 0;
        else
                v_bf->bytesused = vb2_get_plane_payload(vb, 0);


        t_struct->buf = v_bf;

        printk("Thirdy_Driver - t_buffer_prepare OK\n");
        return 0;
```

진행 내용 – V4L2

- **V4L2 드라이버 동작 과정 – STREAMON**
  - 실질적으로 비디오 디바이스를 동작시키는 함수
  - STREAMON 함수 동작 전 BUFFER_QUEUE를 호출하여 앞서 설정한 QUEUE에 BUFFER를 Linked List로 연결
  - Start_streaming 함수를 호출하여 디바이스에서 데이터를 얻어 지속적으로 BUFFER에 저장




```
static void t_buffer_queue(struct vb2_buffer *vb)
{
      struct vb2_queue *v_q = vb2_get_drv_priv(vb->vb2_queue);
        thirdy_t *t_struct = container_of(v_q, thirdy_t, t_viq);
        struct t_buffer *buf = container_of(vb, struct t_buffer, buf);
        unsigned long flags;


        spin_lock_irqsave(&t_struct->irqlock, flags);
        if(likely(!(t_struct->viq_flags & (1 << 0))))
                list_add_tail(&buf->queue, &t_struct->irqqueue);
        else
        {
                buf->state = 5;
                vb2_buffer_done(&buf->buf, VB2_BUF_STATE_ERROR);
        }
        spin_unlock_irqrestore(&t_struct->irqlock, flags);

        printk("Thirdy_Driver - t_buffer_queue OK\n");
}
```

진행 내용 – V4L2

- **V4L2 드라이버 동작 과정**
  - QBUF, DQBUF, STREAMON 함수가 반복하며 디바이스로부터 영상 정보를 받아옴
  - Start_streaming 함수 내용 중 디바이스로부터 받은 정보를 Codec을 통해 Decode 하는 부분이 존재
  - Decode된 데이터는 OpenCV의 ShowImage 함수 등을 이용해 화면에 출력

```
while(1)
{
    STREAMON ⇐ DQBUF ⇐ QBUF ⇐ ...
}
```

```
] Thirdy_Driver - thirdy_dqbuf OK
] Thirdy_Driver - t_buffer_prepare OK
] Thirdy_Driver - t_buffer_queue OK
] Thirdy_Driver - thirdy_qbuf OK
] Thirdy_Driver - thirdy_streamon OK
] Frame complete (EOF found).
] thirdy_buf_finish Start
] Thirdy_Driver - thirdy_dqbuf OK
] Thirdy_Driver - t_buffer_prepare OK
] Thirdy_Driver - t_buffer_queue OK
] Thirdy_Driver - thirdy_qbuf OK
] Thirdy_Driver - thirdy_streamon OK
] Frame complete (EOF found).
] thirdy_buf_finish Start
] Thirdy_Driver - thirdy_dqbuf OK
] Thirdy_Driver - t_buffer_prepare OK
```

문제점

2016. 06. 10.
- vidioc_reqbufs 함수가 동작 하지 않음
  → 해당 함수가 동작하기 위해선 vidioc_g_fmt_vid_cap 함수 혹은 vidioc_g_fmt_vid_cap_mplane 함수가 선언되있어야 하는 것을 발견하고 빈 함수 생성 후 해결

2016. 06. 11.
- vidioc_reqbufs 함수 다시 에러
  → vb2_queue 구조체에 값이 없었던 문제로 커널에서 제공되는 uvc 코드와 skel 코드를 참고하여 해결

2016. 06. 18.
- USB와 통신이 안되는 문제
  → 연결되는 USB장치를 usbcore에 등록하는 부분에서 문제가 발생한 것이었고 해당 부분에서 기존 매크로를 사용하지 않았던 것을 매크로를 사용함으로 해결

2016. 06. 20.
- VIDIOC_S_FMT_VID_CAP 함수 에러(Setting Pixel Format: Device or resource busy)
  → 해당 함수를 호출하는 부분을 분석해본 결과 코드 진행에 필요한 vfh와 vfd의 값이 제대로 저장되어있지 않고 있던 문제로 file->private_data에 필요한 데이터를 저장하는 것으로 해결

문제점

2016. 06. 30. ~ 2016. 07. 06.
- 카메라를 동작시키기 위한 함수 중 하나인 start_streaming 함수를 구현하고 있으나 드라이버 동작 시 무한루프에 빠지면서 OS가 뻗는 현상 발생
  → Queue_Setup()과 Buffer_queue(), Buffer_prepare()에서 서로 다른 Queue와 Buffer를 사용하는 문제점을 발견하고 동일한 Queue와 Buffer를 사용하도록 수정 후 OS 다운 현상 해결

2016. 07. 07.
- Dmesg를 통해 확인해본 결과 UVCVideo 드라이버와 동작 과정이 거의 동일하나 디바이스로부터 읽어온 데이터를 Buffer에 저장하지 못하는 현상 발생
  → UVCVideo 소스와 비교하여 일치 하지 않는 부분 발견 후 해당 함수 분석
  → 분석 결과 현재 개발한 드라이버에서는 불필요한 기능으로 판단 후 기능 제거
  → 이후 드라이버 정상 작동

```
static void t_video_validate_buffer(thirdy_t *t_struct, struct t_buffer *buf)
{
        if (t_struct->t_data->dwMaxVideoFrameSize != buf->bytesused)
                buf->error = 1;
}
```

≠

```
static void uvc_video_validate_buffer(const struct uvc_streaming *stream,
                                      struct uvc_buffer *buf)
{
        if (stream->ctrl.dwMaxVideoFrameSize != buf->bytesused &&
            !(stream->cur_format->flags & UVC_FMT_FLAG_COMPRESSED))
                buf->error = 1;
}
```

→

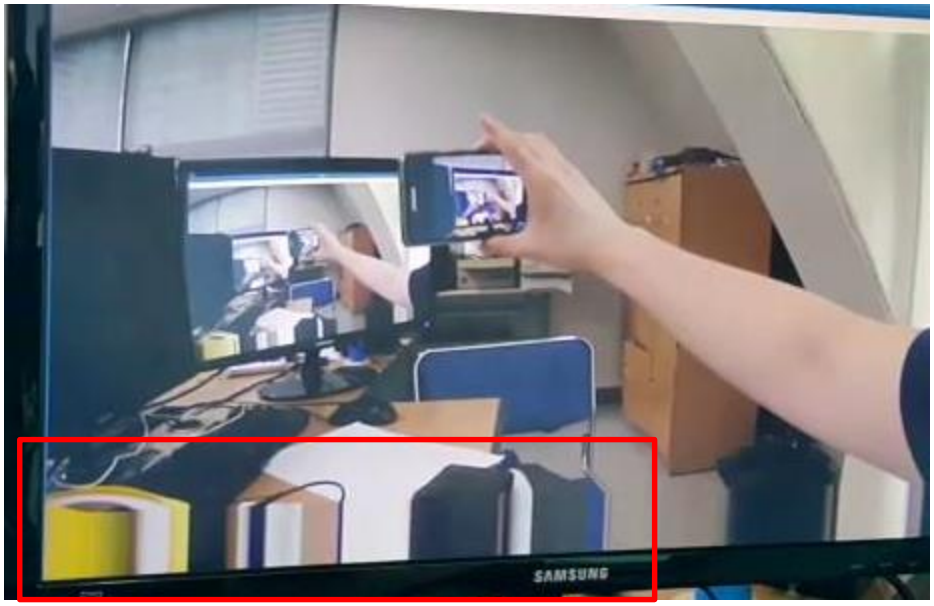
```
static void t_video_validate_buffer(thirdy_t *t_struct, struct t_buffer *buf)
{
        if (t_struct->t_data->dwMaxVideoFrameSize != buf->bytesused && 0)
                buf->error = 1;
}
```

문제점

2016. 07. 10. ~ 현재

- 드라이버 정상 구동 후 S_FMT_VID_CAP 함수를 통해 데이터 포맷을 설정하려고 시도 했으나 아무런 설정이 되지 않고 기본값으로 동작하는 문제 발견
  -> UVCVideo 코드와 비교해본 결과 S_FMT_VID_CAP 내용 중 구현되지 않은 내용 발견
  -> 직접적인 설정 부분 내용이 굉장히 방대하기 때문에 시간을 갖고 구현하기로 결정

- 화면 출력 도중 중간중간 화면 아랫부분이 번짐 현상이 있는 것을 발견
  -> 아직 특별히 문제되는 구간을 발견하지 못한 상태 ..



진행 내용

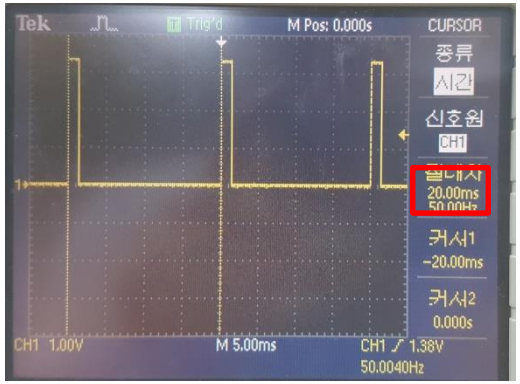
Encoder
Motor Control

서영일

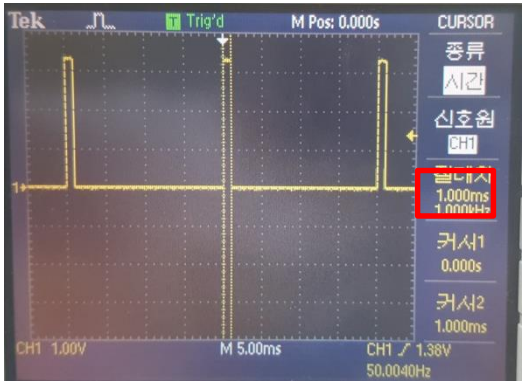
ePWM Module

B5	GIOA[5]	NONE	NONE	EXTCLKIN	NONE	eTPWM1A

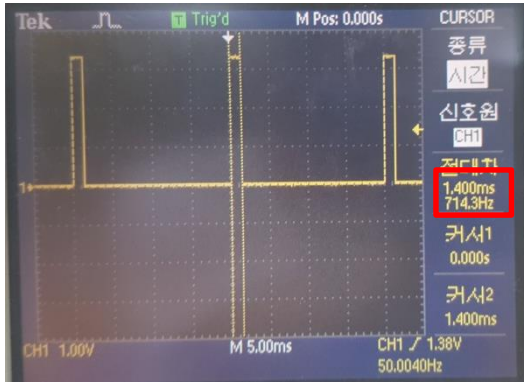
PWM Signal Outpin -> B5



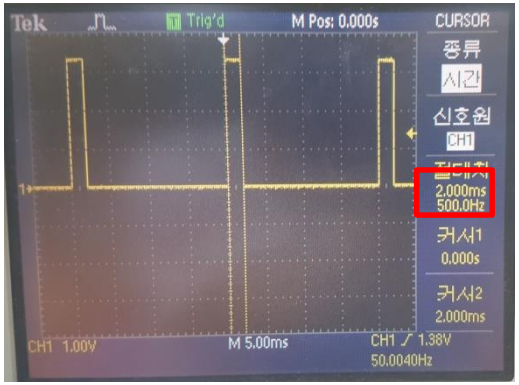
< 20ms 주기 >



< 5% duty
(1ms) >

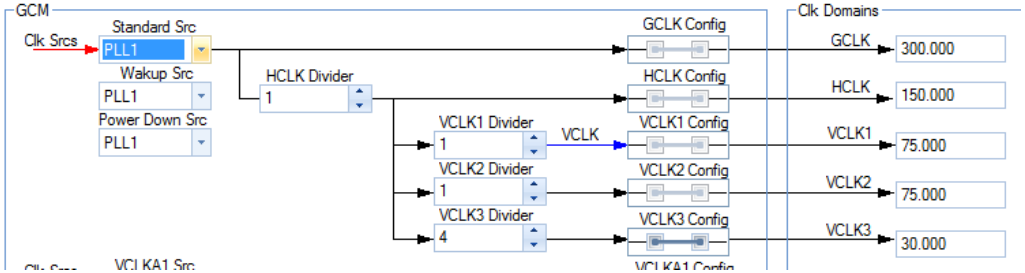


< 7.5% duty
(1.5ms) >

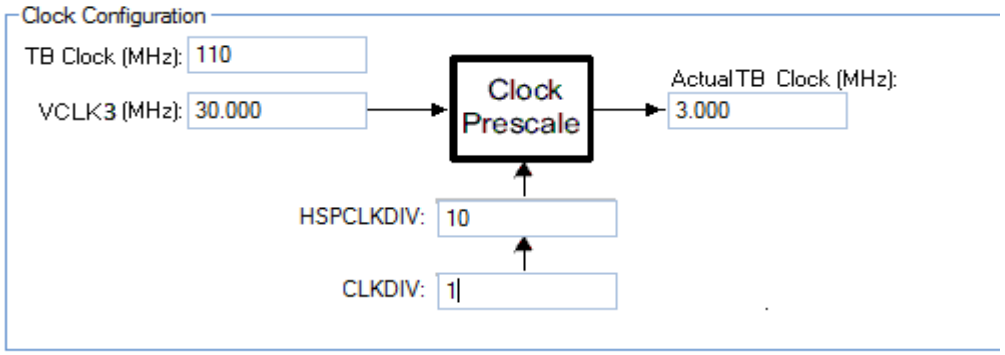


< 10% duty
(2ms) >

ePWM Module



< Global Clock Module을 이용하여
VCLK3를 30MHz로 분주 >



< 30MHz가 된 VCLK3를 1/10하여
ePWM 모듈의 동작 클럭을 3MHz로 분주 >

$$\frac{60000}{30000000} = 20m$$

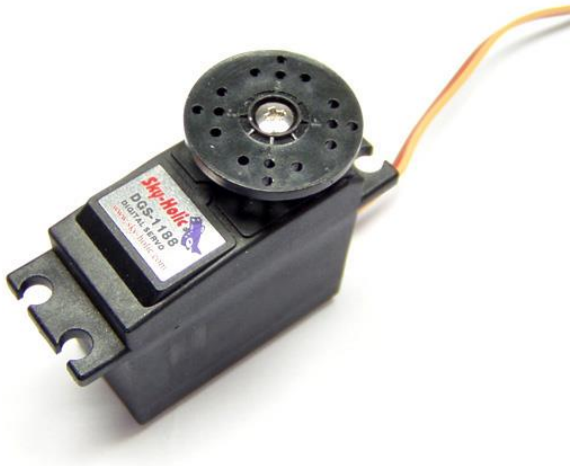
```
/** - Sets time period or frequency for ETP
etpwmREG1->TBPRD = 59999U;

/** - Setup the duty cycle for PWMA */
etpwmREG1->CMPA = 0U;
```

ePWM Module

```
switch(cnt)
{
case 0:
    etpwmSetCmpA(etpwmREG1, 6000);  // 2ms (10% duty)
    break;
case 1:
    etpwmSetCmpA(etpwmREG1, 3000);  // 1ms (5% duty)
    break;
case 2:
    etpwmSetCmpA(etpwmREG1, 4500);  // 1.5ms (7.5% duty)
    break;
}
```

CMPA Register 값으로 Duty Ratio를 설정

Servo Motor



1.5ms = 4500
/150
10us = 30 = 1°

		DGS-288	DGS-1188
Type of Gear		Metal gear	
Motor type		Ferrite brush motor	
Operating Voltage		4.8v ~6.0v	
Neutral pulse		1.5 ㎳(Adjustabla)	
Travel angle		40~45° at ±400㎲(Adjuastable)	
Dead Band		1~2 ㎲ (Adjustable)	
Speed	(sec/60° at4.8V)	0.12	0.15
	(sec/60° at6.0V)	0.11	0.12
Torque	(kg/cm at 4.8V)	6.0	10.0
	(kg/cm at 6.0V)	7.0	12.0
Weight(g)		46	51
Dimensions(mm)		40x20x34	40x20x38
Ball bearing		Basic(2BB)	Basic(2BB)

eQEP Module - Encoder

모델 : E50S8-2000-3-N-5

타입 : 인크리멘탈

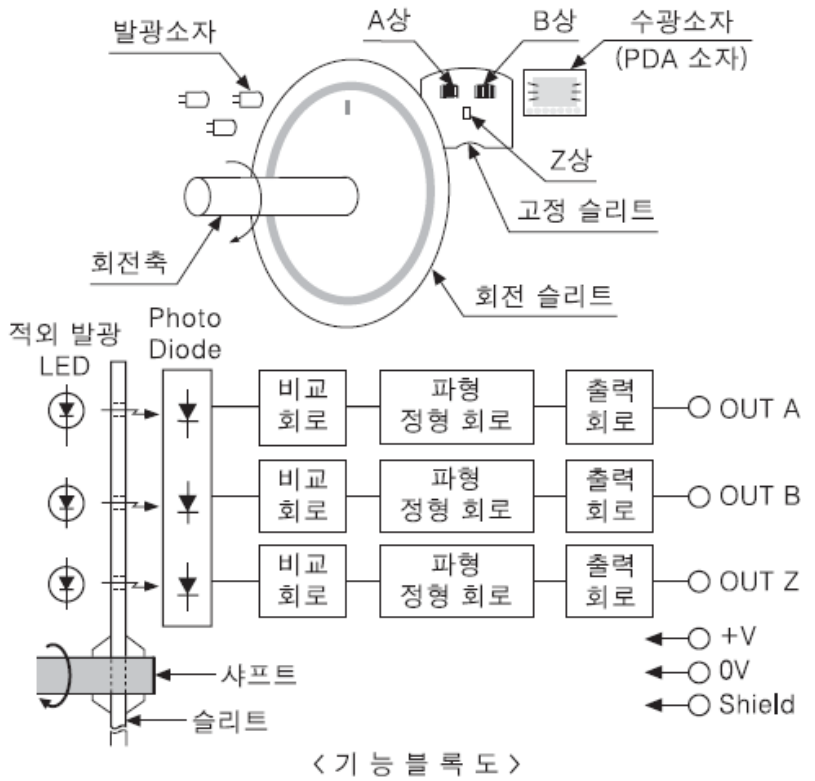
분해능 : 2000 Pulse/Rev

출력상 : A, B, Z(I)

제어출력 : NPN 오픈 콜렉터 출력

전원전압 : 5V

최대허용회전수 : 5000rpm



〈 기 능 블 록 도 〉

eQEP Module

4.2.1.5 Enhanced Quadrature Encoder Pulse Modules (eQEP)

Table 4-5. ZWT Enhanced Quadrature Encoder Pulse Modules (eQEP)(1)

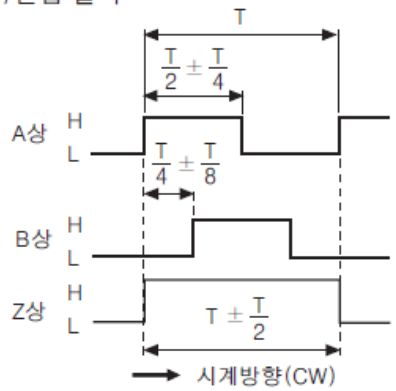
Terminal		Signal Type	Default Pull State	Pull Type	Output Buffer Drive Strength	Description
Signal Name	337 ZWT					
MIBSPI3CLK/AD1EXT_SEL[1]/**eQEP1A**	V9	Input	Pullup	Fixed, 20 µA	-	Enhanced QEP1 Input A
MIBSPI3NENA/MIBSPI3NCS[5]/N2HET1[31]/**eQEP1B**	W9	Input	Pullup	Fixed, 20 µA	-	Enhanced QEP1 Input B
MIBSPI3NCS[0]/AD2EVT/**eQEP1I**	V10	I/O	Pullup	Fixed, 20 µA	8mA	Enhanced QEP1 Index
MIBSPI1NCS[1]/MII_COL/N2HET1[17]/**eQEP1S**	F3	I/O	Pullup	Fixed, 20 µA	8mA	Enhanced QEP1 Strobe
N2HET1[1]/MIBSPI4NENA/N2HET2[8]/**eQEP2A**	V2	Input	Pullup	Fixed, 20 µA	-	Enhanced QEP2 Input A
N2HET1[3]/MIBSPI4NCS[0]/N2HET2[10]/**eQEP2B**	U1	Input	Pullup	Fixed, 20 µA	-	Enhanced QEP2 Input B
GIOA[2]/N2HET2[0]/**eQEP2I**	C1	I/O	Pullup	Fixed, 20 µA	8mA	Enhanced QEP2 Index
N2HET1[30]/MII_RX_DV/**eQEP2S**	B11	I/O	Pullup	Fixed, 20 µA	8mA	Enhanced QEP2 Strobe

(1) These signals are double-synchronized and then optionally filtered with a 6-cycle VCLK4-based counter.

A -> V9
B -> W9
Z -> V10

▣ 출력파형

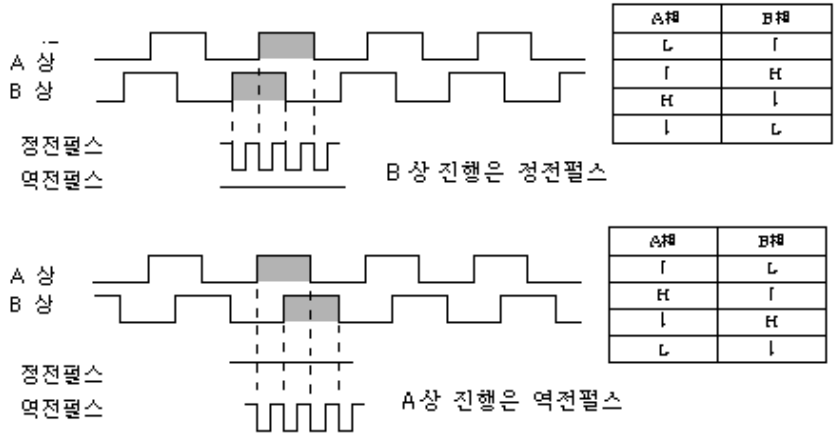
●Totem Pole 출력/NPN 오픈콜렉터 출력
/전압 출력



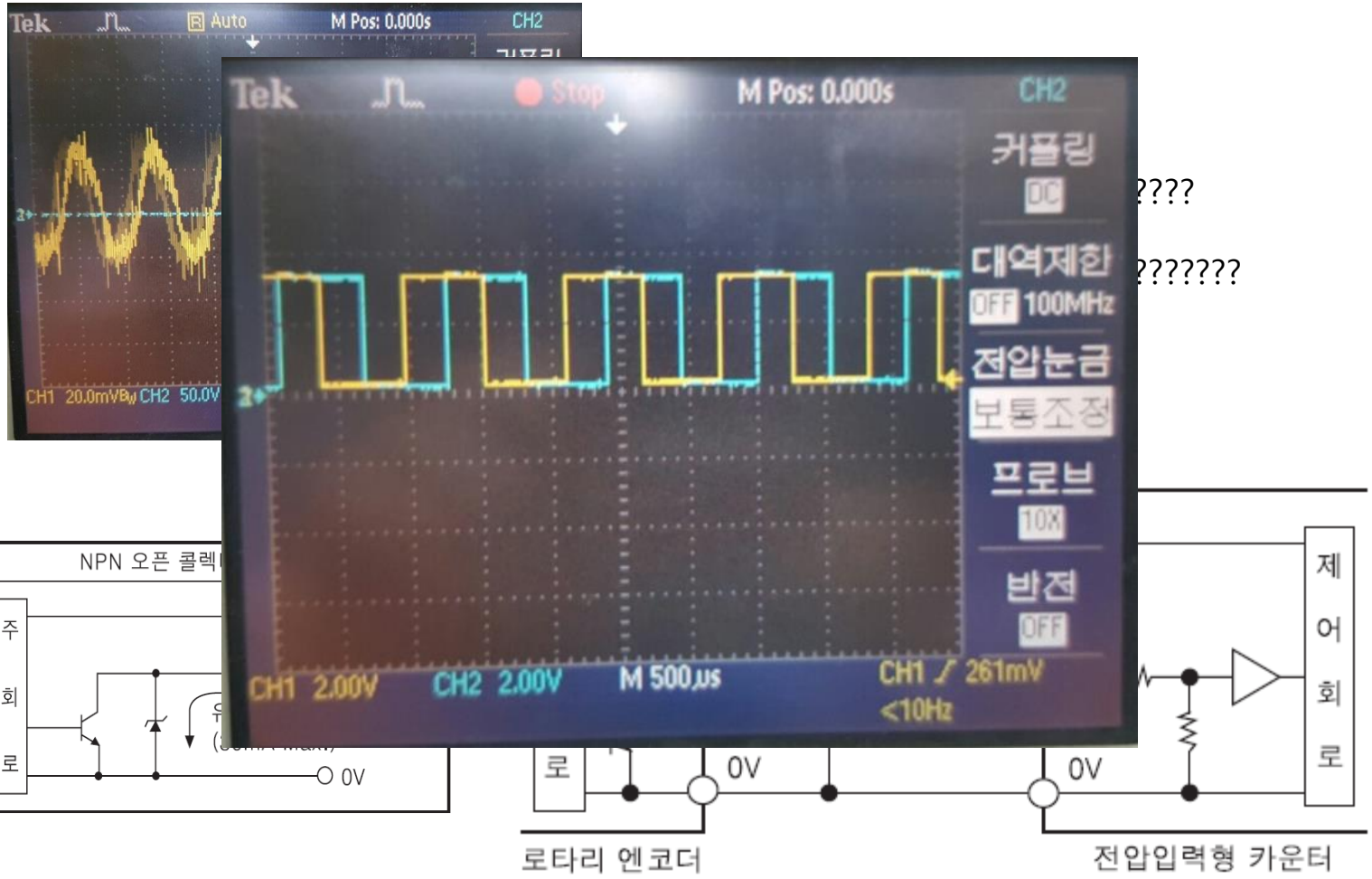
※시계방향(CW):Shaft에서 볼 때 우회전입니다.

A상과 B상의
위상차 90°

·회전방향의 검출



eQEP Module


Tek
Auto
M Pos: 0.000s
CH2
CH1 20.0mV CH2 50.0V
Tek
Stop
M Pos: 0.000s
CH2
커플링
DC
대역제한
OFF 100MHz
전압눈금
보통조정
프로브
10X
반전
OFF
CH1 2.00V
CH2 2.00V
M 500µs
CH1 ∫ 261mV
<10Hz
????
???????
NPN 오픈 콜렉
주
회
로
0V
로
0V
0V
제
어
회
로
로타리 엔코더
전압입력형 카운터

eQEP Module

Table 34-4. eQEP Position Counter Register (QPOSCNT) Field Descriptions

Bits	Name	Description
31-0	QPOSCNT	This 32-bit position counter register counts up/down on every eQEP pulse based on direction input. This counter acts as a position integrator whose count value is proportional to position from a give reference point.

Pulse 개수를 카운트 (Pulse 수 x 4의 값을 가짐)

Table 34-11. eQEP Unit Timer Register (QUTMR) Field Descriptions

Bits	Name	Description
31-0	QUTMR	This register acts as time base for unit time event generation. When this timer value matches with unit time period value, unit time event is generated.

VCLK3(기본 75Mhz) 를 이용하는 Unit Timer

Table 34-12. eQEP Unit Period Register (QUPRD) Field Descriptions

Bits	Name	Description
31-0	QUPRD	This register contains the period count for unit timer to generate periodic unit time events to latch the eQEP position information at periodic interval and optionally to generate interrupt.

QUTMR의 주기(MAX 값)을 정해줌 QUTMR=QUPRD가 되면 Unit time event 발생

Table 34-10. eQEP Position Counter Latch Register (QPOSLAT) Field Descriptions

Bits	Name	Description
31-0	QPOSLAT	The position-counter value is latched into this register on unit time out event.

Unit time event가 발생하면 QPOSCNT의 값을 저장

eQEP Module

```
72 #define PI 3.141592
73 #define wheel_r 0.000075 // km
74 #define gear_ratio 18
75 #define resolution 2000
76 #define pulse_per_sec 250000
77
78 double velo = 0U;
```

km/h 단위의 값으로
계산하여
velo 변수에 저장

```
void eqepNotification(eqepBASE_t *eqep,uint16 flags)

{
/*  enter user code between the USER CODE BEGIN and USER CODE END. */
/* USER CODE BEGIN (51) */
    double cnt = eqepREG1->QPOSLAT;
    double enc_rps = pulse_per_sec / resolution * cnt;
    double wheel_rps = enc_rps / gear_ratio;
    double velocity = 2 * PI * wheel_r * 3600 * wheel_rps;

    velo = velocity;
/* USER CODE END */
}
```

eQEP Module

GRP(eQEP1)			
GRP(eQEP1).REG(QUTMR)	Unsigned / Readable,Writeable	0x000072ED	
cnt	double	0x416FFFFFE0000000 (Hex)	0x08001278
GRP(eQEP1).REG(QFLG)	Unsigned / Readable,Writeable	0x0869	
GRP(eQEP1).REG(QCLR)	Unsigned / Readable,Writeable	0x0000	
velo	double	197651866.59854999	0x08001518
GRP(eQEP1).REG(QPOSLAT)	Unsigned / Readable,Writeable	0x00FFFFFF	
GRP(eQEP1).REG(QPOSCNT)	Unsigned / Readable,Writeable	0x00FFFFFF	
Add new expression			

Underflow ??????????????????????????????

197651866.59854999 Km/h ??????????????????????????????

??
?????????????????????? ??

Encoder, Motor Control
서영일

문제점

2016. 06. 07
TMS570보드의 N2HET 모듈을 이용하여 모터 구동을 위한 PWM 신호를 출력함.
ESC 켈리브레이션을 위해서는 20ms 주기에 1ms, 1.5ms, 2ms의 듀티를 갖는 PWM 신호가 필요한데 N2HET 모듈의 PWM 신호 듀티를 정해주는 함수인 pwmSetDuty는 uint32형 인자에 원하는 듀티비를 정수로 받아 출력하는 함수여서 7.5%에 해당하는 1.5ms를 출력 할 수 없었음

2016. 06. 10
1.5ms 듀티의 신호 출력을 위해 ePWM 모듈을 사용하였다. 이 모듈의 동작 클럭인 VCLK3는 기본 75MHz로 20ms 주기를 만드려면 20 x 10^3 x 75 x 10^6 = 1500000 이라는 값을 넣어 줘야 했는데 이 값이 들어갈 TBPRD 레지스터는 16비트짜리로 1500000를 넣을 수 없는 문제가 있었음

2016. 06. 13
앞의 문제를 해결하기 위해 보드의 클럭들을 제어하는 GCM을 이용해 VCLK3를 30MHz까지 낮추고 ePWM 모듈의 디바이더를 이용해 1/10을 하여 3Mhz까지 동작 스피드를 낮춤으로써 계산되는 값을 작게 만들었고 20 x 10^3 x 3 x 10^6 = 60000 이라는 값을 얻어 레지스터에 넣을 수 있게 되어 듀티비를 정해 줄 수 있는 CMPA 레지스터에 각각 5%, 7.5%, 10%에 해당하는 3000, 4500, 6000 이라는 값을 넣어 1, 1.5, 2ms의 듀티비를 갖는 PWM 신호를 만들 수 있었음

2016. 06. 15
차체의 조향을 위해 전륜에 Servo motor를 연결하고, BLDC모터와 마찬가지로 PWM 신호를 이용하여 원하는 각도로 움직이도록 제어하였음

Encoder, Motor Control
서영일

문제점

2016. 06. 20
모터의 현재 속도를 파악하기 위해 로터리 엔코더를 구매하여 출력 신호를 오실로스코프로 테스트하였으나 원하는 클럭 파형이 나오지 않고 이상한 사인파형만 측정됨.

2016. 06. 23
출력방식이 NPN 오픈 콜렉터 출력인 것을 파악하고 엔코더에서 출력되는 클럭을 카운트 할 카운터에 해당하는 R5의 입력핀과 엔코더의 출력핀 사이에 pull-up저항을 연결하여 R5 자신의 입력 전압을 이용하여 입력받을 수 있게 회로를 구성하여 원하는 신호를 확인함.

2016. 07. 04 ~ 현재
R5에서 제공되는 eQEP 모듈을 이용하여 엔코더를 제어하고자 하였음
Halcogen의 예제 코드를 이용하여 엔코더 출력 값을 받아보려 했으나 어떤 이유에서인지 아무런 변화를 확인하지 못함.. 현재 해당 모듈의 데이터시트를 자세히 읽으면서 동작 방식이나 제어 방법을 학습 중

진행 내용

RTOS
Circuit Design

이건수

솔레노이드 밸브

$I_E = I_B + I_C$ — ①

$I_C = \beta I_B$ — ②

$I_{E1} = I_{B1} + I_{C1}$

$I_{C1} = I_{B1} + \beta I_{B1} = (1+\beta) I_{B1}$

$I_E = I_{B2} + I_{C2}$

$I_{E2} = I_{B2} + \beta I_{B2}$

$= (1+\beta) I_{B2}$

$I_{E1} = I_{B2}$

$I_{E2} = I_{B2} + \beta I_{B2}$

$= (1+\beta) I_{B2}$

$= (1+\beta)[(1+\beta) I_{B1}]$

$I_{E2} = I_{B2} + \beta I_{B2}$

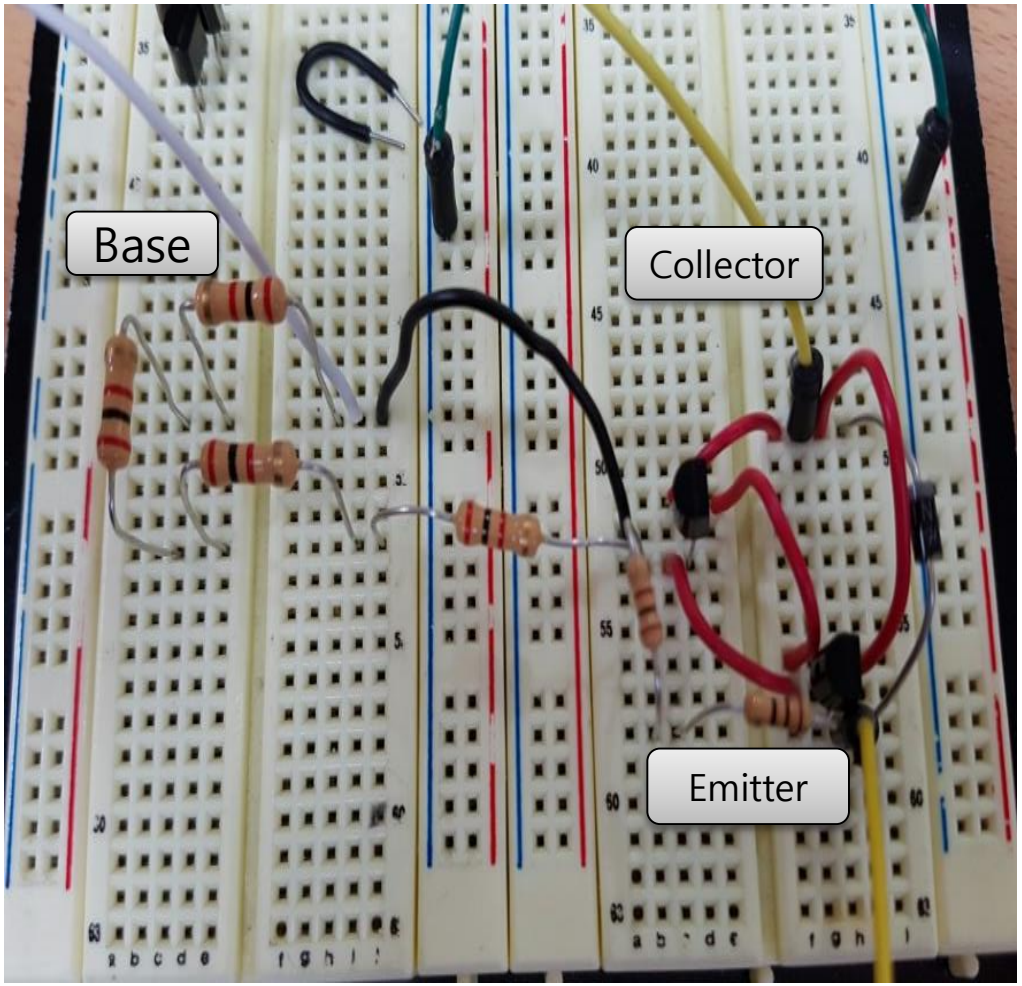
$= (1+\beta) I_{B2}$

$= (1+\beta)[(1+\beta) I_{B1}]$

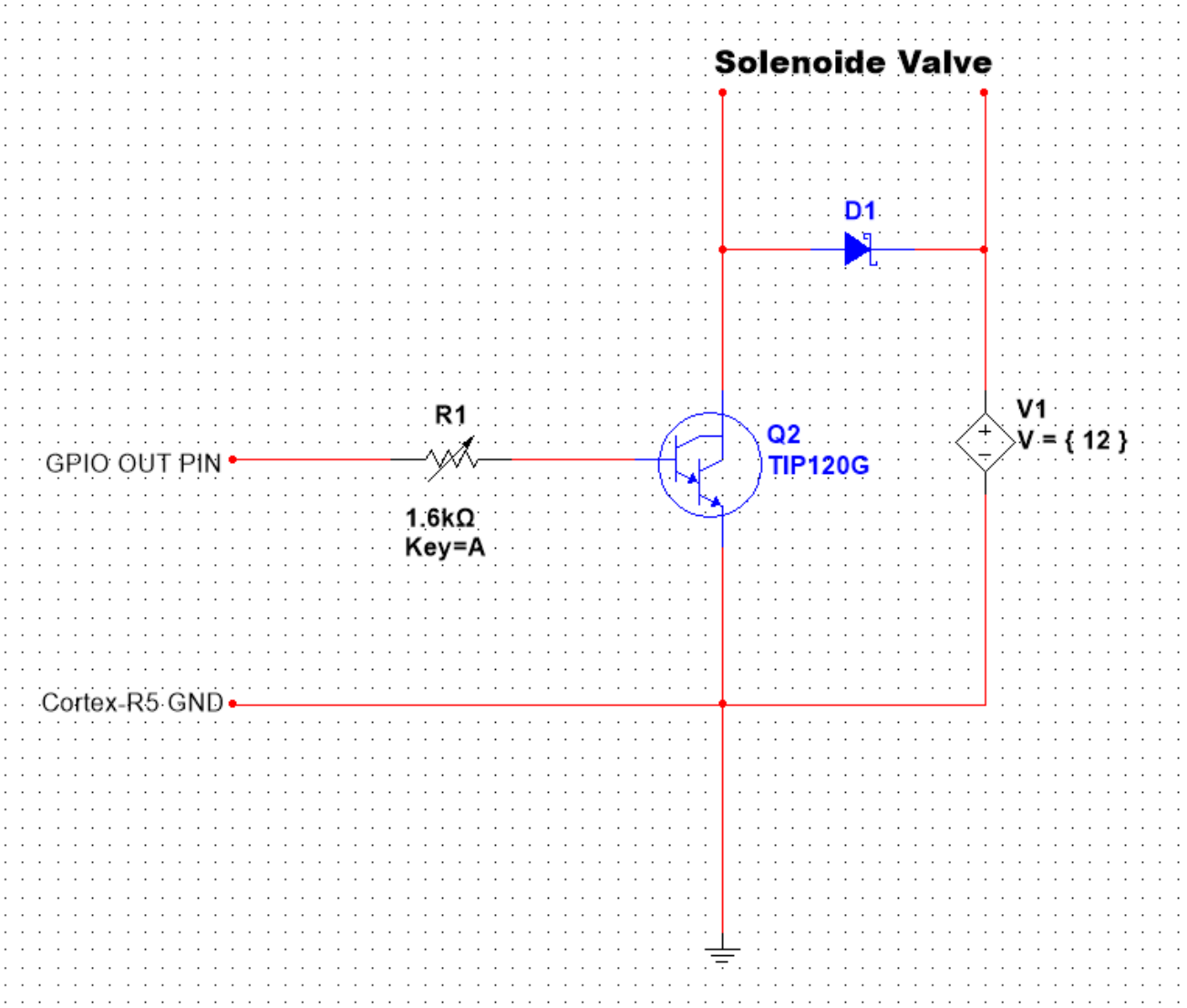
$\therefore \beta \times \beta I_{B1} =$

$\therefore \beta^2 I_{B1}$

$\therefore I_{E2} = \beta^2 I_{B1}$ //



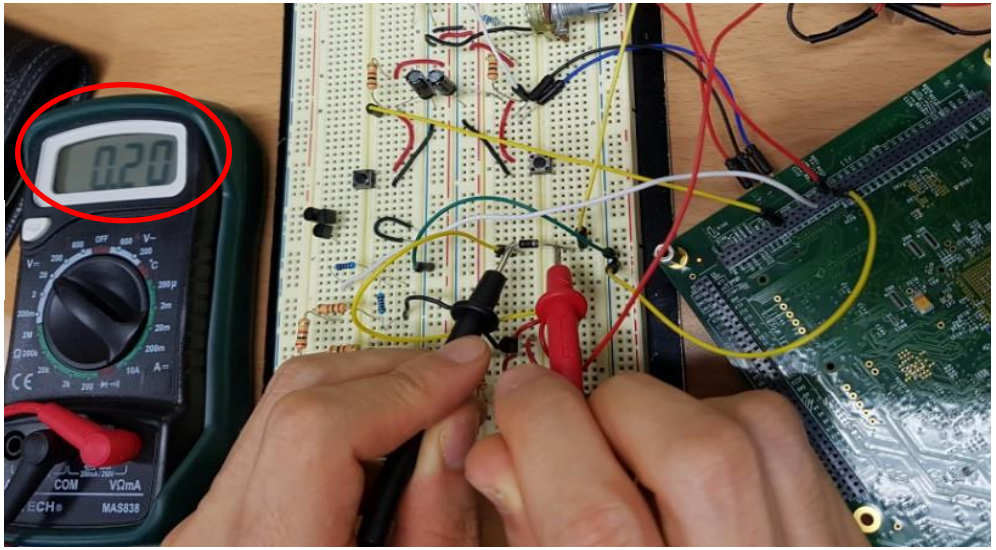
솔레노이드 밸브

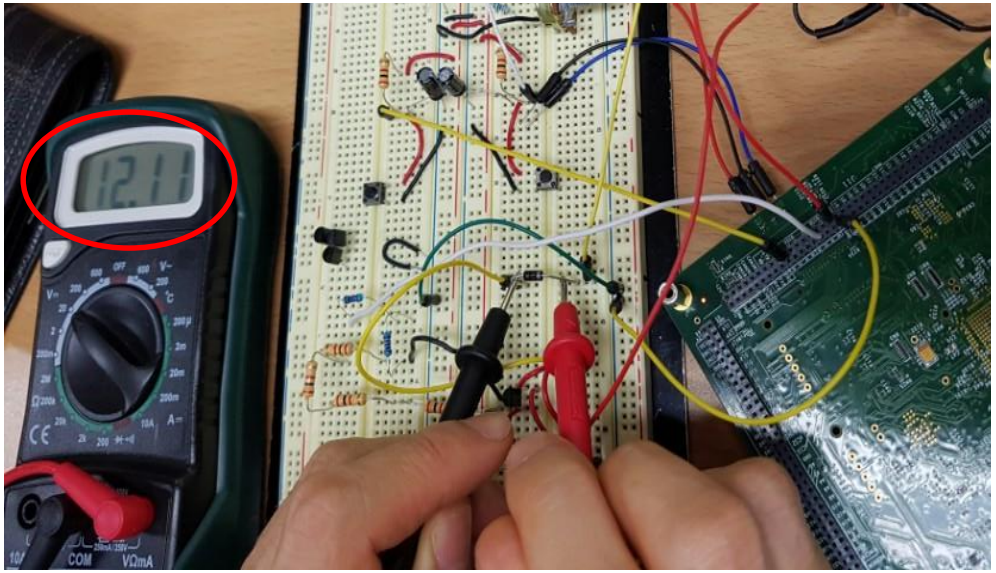

Solenoide Valve
D1
R1
GPIO OUT PIN
1.6kΩ
Key=A
Q2
TIP120G
V1
V = { 12 }
Cortex-R5 GND

솔레노이드 밸브

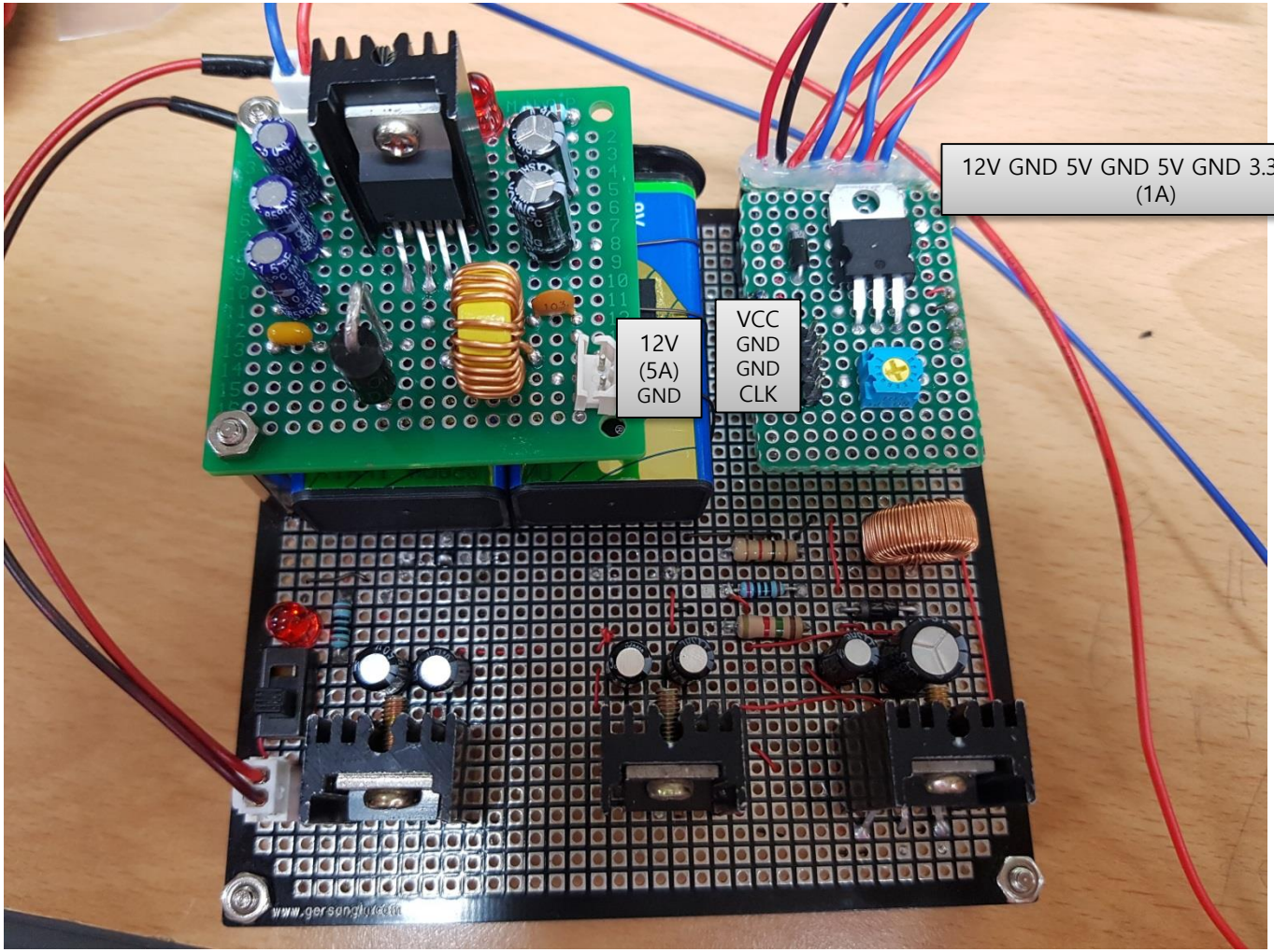
```
#define SOLENOIDE_VALVE_ON      gioPORTB->DOUT |=  0x04
#define SOLENOIDE_VALVE_OFF     gioPORTB->DOUT &= ~0x04
```

```
void Valve(void *pvParameters)
{
    for(;;)
    {
        if( val )           // solenoide on
        {
            vTaskSuspendAll();
            SOLENOIDE_VALVE_ON;
            xTaskResumeAll();
        }
        else if( !val )
        {
            vTaskSuspendAll();
            SOLENOIDE_VALVE_OFF;
            xTaskResumeAll();
        }
    }
}
```

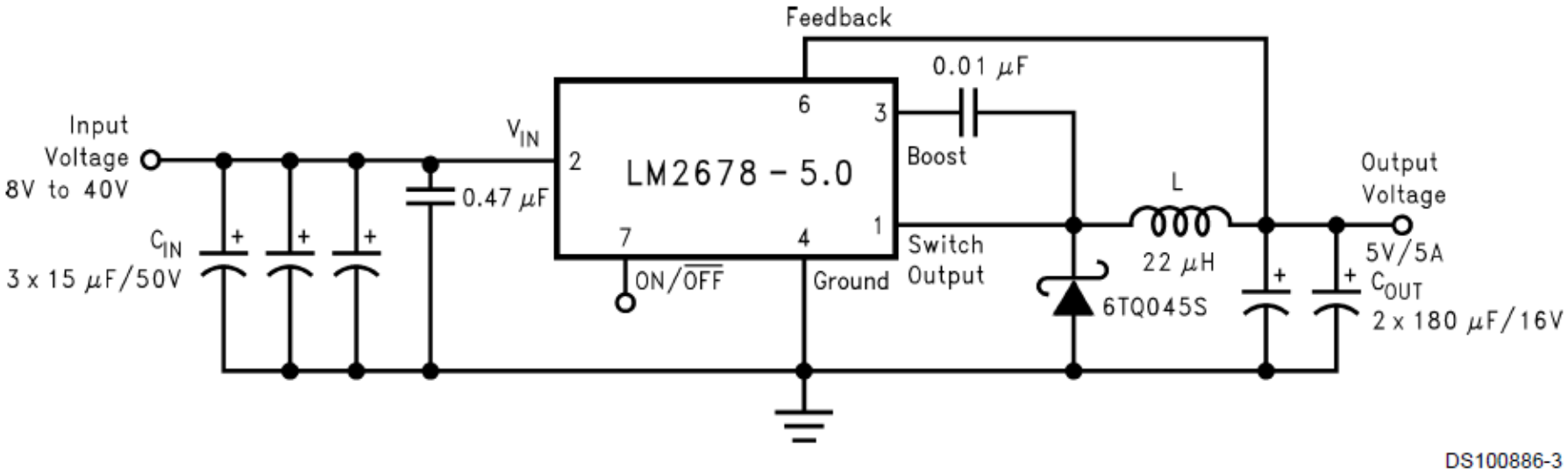




전원부

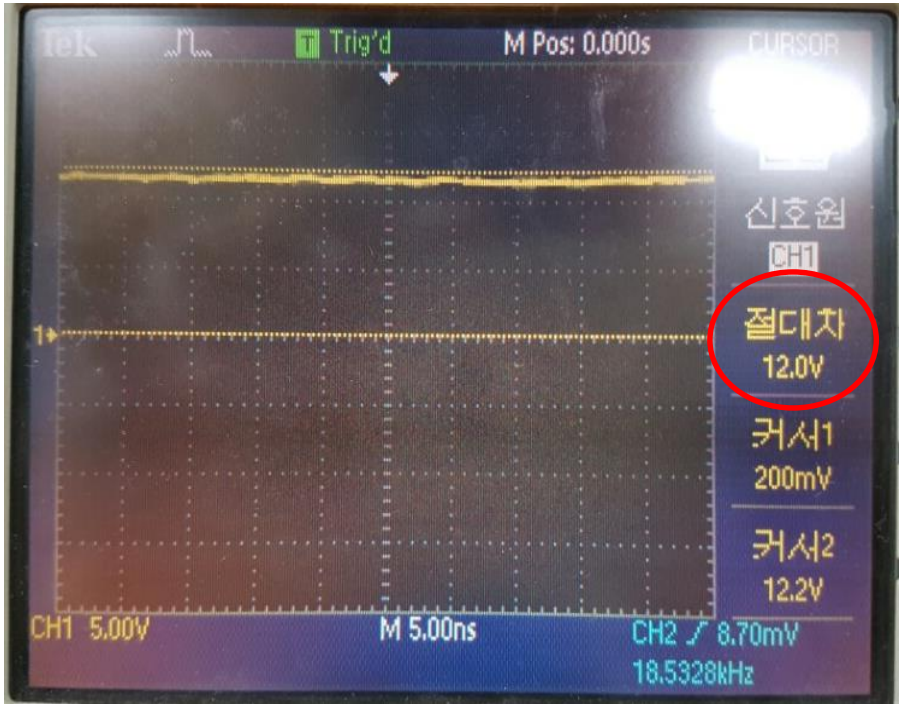


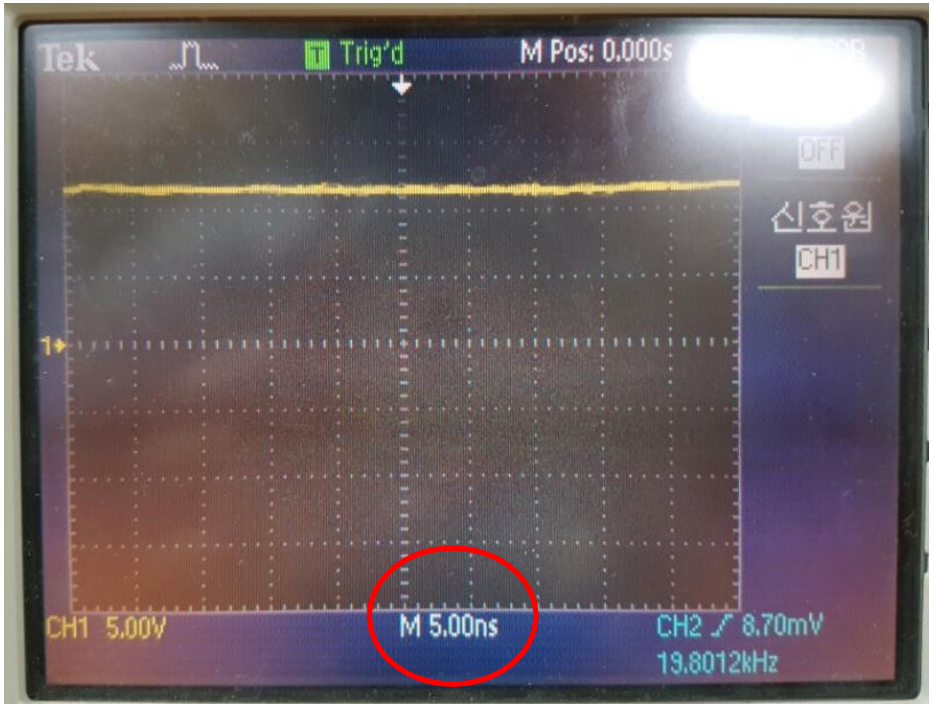
전원부


Feedback
0.01 µF
Input
Voltage
8V to 40V
V_{IN}
6
3
2
LM2678 - 5.0
Boost
L
Output
Voltage
0.47 µF
C_{IN}
3 x 15 µF/50V
7
4
1
Switch
Output
22 µH
5V/5A
ON/OFF
Ground
6TQ045S
C_{OUT}
2 x 180 µF/16V
DS100886-3

전원부 구동문제

Symbol	Parameter	Conditions	Typical (Note 3)	Min (Note 4)	Max (Note 4)	Units
V_{OUT}	Output Voltage	V_{IN} = 15V to 40V, 100mA ≤ I_{OUT} ≤ 5A	12	11.76/**11.64**	12.24/**12.36**	V
η	Efficiency	V_{IN} = 24V, I_{LOAD} = 5A	92			%





RTOS

```
void canMessageNotification(canBASE_t *node, uint32 messageBox)
{
    if(node==canREG1)
    {
        uint8 rx_data1[D_COUNT] = {0};
        canGetData(canREG1, canMESSAGE_BOX2, (uint8 * )&rx_data1); /* copy to RAM */
        sciDisplayText(sciREG1,"rx_data1\n\r",10);
        Int_CharDisplay(sciREG1, &rx_data1, 2);

        dir = rx_data1[DIR_INDEX] & 0xFF;
        ang = rx_data1[ANG_INDEX] & 0xFF;
        val = rx_data1[VAL_INDEX] & 0xFF;
    }
}
```

00 00 00 01 00 01 00 05

총 발사 차 방향 차 각도

```
void Servo(void *pvParameters)
{

    for(;;)
    {
        if( ang > 40 )  continue;
        else if( dir == RIGHT )
        {
            vTaskSuspendAll();
            etpwmSetCmpA(etpwmREG1, SERVO_N - ang * 50);
            xTaskResumeAll();
        }
        else if( dir == LEFT )
        {
            vTaskSuspendAll();
            etpwmSetCmpA(etpwmREG1, SERVO_N + ang * 50);
            xTaskResumeAll();
        }
    }
}
```

RTOS
이건수

문제점

2016. 06. 13.
- SafeRTOS 데모버전 함수 내부를 볼 수 없어 스케쥴링 오류시 원인파악 어려움
  => Cotex R5보드에 맞게 포팅된 FreeRTOS를 사용(함수 내부를 볼 수 있음)

2016. 06. 16.
- 같은 우선순위 태스크끼리 스케쥴링이 되지만 우선순위가 다르면 스케쥴링이 되지않음
  => 태스크간 딜레이를 줌으로 해결
- 같은 우선순위 태스크간 딜레이나 다른 태스크 일시정지를 안시키면 현재 태스크 실행중 선점당함
  => vTaskSuspendAll()함수와 vTaskResume()함수를 이용하여 선점당하는 문제 해결

2016. 06. 20. ~ 현재
- 기본 태스크 2개(LED제어)를 실행시키면서 CAN통신 인터럽트로 부터 데이터 값 받아오기 완료
- LED가 아닌 BLDC모터, 서보모터, 솔레노이드 밸브를 태스크로 만들어 실행시키면서 CAN통신을 이용한 DSP로부터 데이터값 받아오기!!

Circuit Design
이건수

문제점

2016. 06. 01.
- 12V, 5V, 3.3V 전원회로 완성

2016. 06. 11.
- 12V를 사용하는 솔레노이드 밸브를 R5 GPIO핀(3.3V)으로 제어에 전압부족 발생
  => 별도의 외부전원 12V 전원을 트랜지스터 스위칭 특성을 이용하여 ON/OFF 제어

- 하나의 트랜지스터로 포화영역에서도 솔레노이드 모터에 전류부족 발생
  => 두개의 트랜지스터를 하나의 트랜지스터(Dalington Transister)회로로 구성
  => Dalington 회로를 하나의 반도체 소자로 되어있는 tip120 칩을 구매

2016. 06. 17.
- 밸브가 기압에 따라 버틸 수 있게 베이스에 가변저항을 사용(베이스의 전류조절 / 포화전류 2mA)
- 전원이 Off될 때 발생하는 역전압을 정류다이오드로 방지
- 솔레노이드 밸브 제어 완료!!

2016. 06. 30.
- DSP가 12V 레귤레이터로 동작하지 않는 문제발생(DSP에서 최대 5A 사용, 전류부족)
  => 5A를 출력하는 레귤레이터 및 주변 소자 주문 완료

2016. 07. 04. ~ 현재
- 알카라인 전지로 DSP구동 테스트. But, 레귤레이터 없이 테스트 불가
  => 레귤레이터 도착 즉시 확인 예정

RTOS
이건수

문제점

2016. 07. 12.
- 12V 5A 출력의 전원 회로로 DSP 구동하지 못함
- 콘덴서 값 바꿔봐도 전압은 잘 나오지만 DSP 구동시키지 못함
- 오실로스코프로 찍어봐도 12V 파형은 정말 깔끔하게 나옴
- 원인은 전류 부족으로 생각됨
- 하지만 빵판테스트는 DSP구동이 되었지만,
  만능기판에 납땜 후에는 DSP 구동이 되지 않은 것이 의문

2016. 07. 13
- 메뉴얼내 전형적인 회로도에 소자들을 똑같이 구입

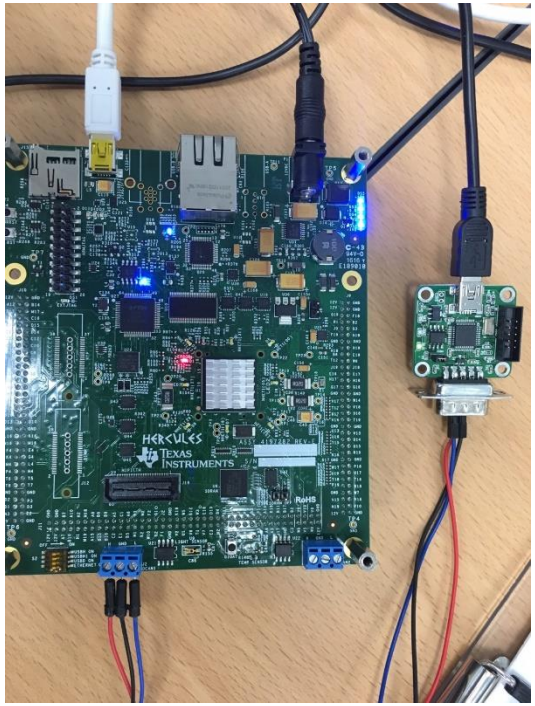
진행 내용

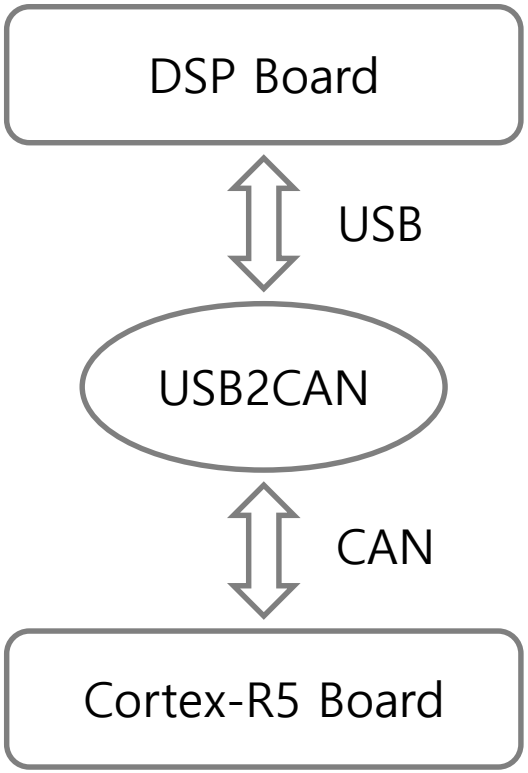
CAN
Motor Control

이찬

CAN Communication
이찬

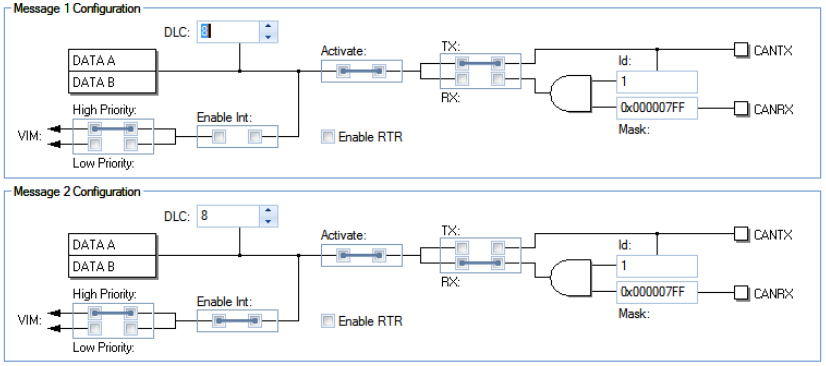
진행사항 및 공부내용





CAN Communication
이찬

진행사항 및 공부내용



MCU

Bit Rate, ID, Mask Setting
Enable Interrupt(RX)

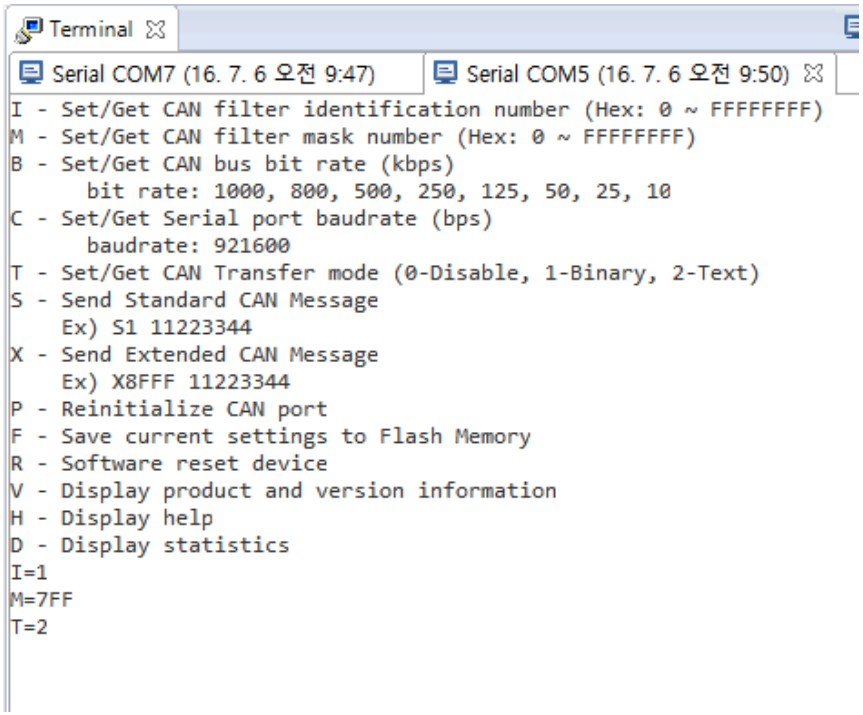
```
/** @def canMESSAGE_BOX1
*   @brief Alias name for CAN message box 1
*
*   @note This value should be used for API argument @a messageBox
*/
#define canMESSAGE_BOX1 1U
```

```
/** - Initialize message 1
*     - Wait until IF1 is ready for use
*     - Set message mask
*     - Set message control word
*     - Set message arbitration
*     - Set IF1 control byte
*     - Set IF1 message number
*/
/*SAFETYMCUSW 28 D MR:NA <APPROVED> "Potentially infinite loop found - Hardware Status check for execution seque
while ((canREG1->IF1STAT & 0x80U) ==0x80U)
{
} /* Wait */


    canREG1->IF1MSK  = 0xC0000000U | (uint32)((uint32)((uint32)0x000007FFU & (uint32)0x1FFFFFFFU) << (uint32)0U);
    canREG1->IF1ARB  = (uint32)0x80000000U | (uint32)0x40000000U | (uint32)0x20000000U | (uint32)((uint32)((uint32)1
    canREG1->IF1MCTL = 0x00001080U | (uint32)0x00000000U | (uint32)0x00000000U | (uint32)8U;
    canREG1->IF1CMD  = (uint8) 0xF8U;
    canREG1->IF1NO   = 1U;
```

```
/** @fn void can1HighLevelInterrupt(void)
*   @brief CAN1 Level 0 Interrupt Handler
*/
#pragma CODE_STATE(can1HighLevelInterrupt, 32)
#pragma INTERRUPT(can1HighLevelInterrupt, IRQ)

/* SourceId : CAN_SourceId_020 */
/* DesignId : CAN_DesignId_018 */
/* Requirements : HL_SR221, HL_SR222, HL_SR223 */
void can1HighLevelInterrupt(void)
{
    uint32 value = canREG1->INT;
    uint32 ES value:
```

CAN Communication
이찬

진행사항 및 공부내용

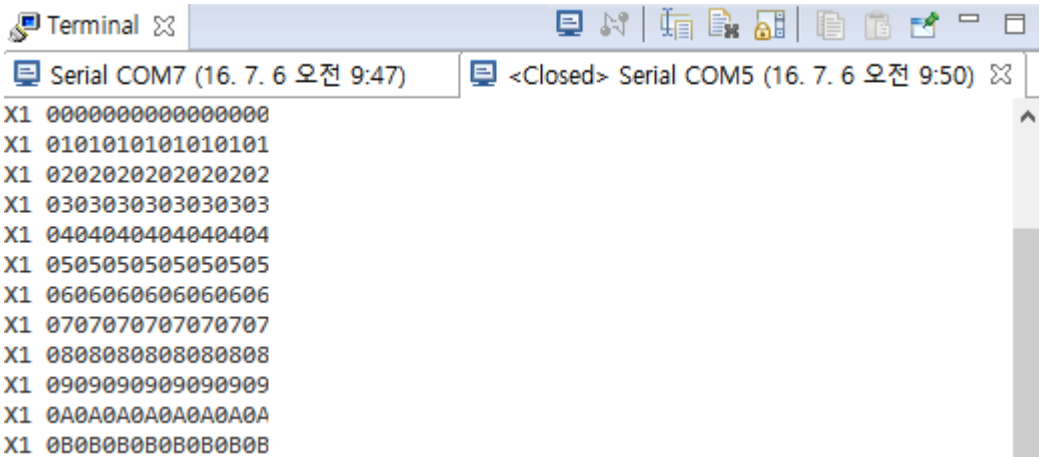


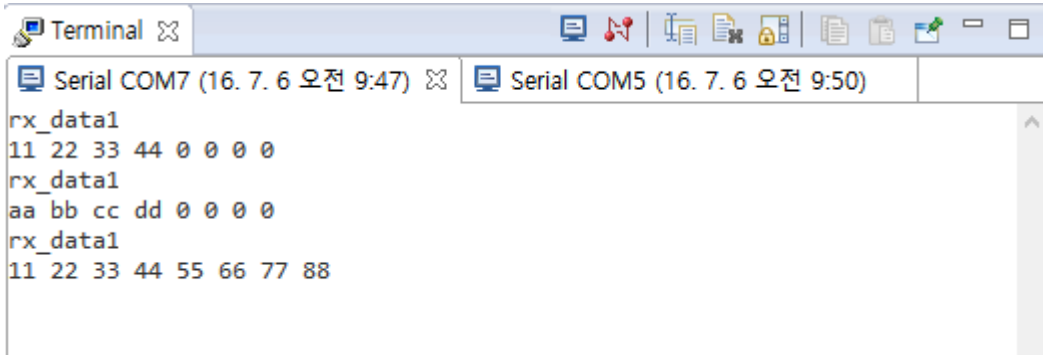
CAN2USB Module

ID, Mask
Baudrate
Transfer mode Setting

CAN Communication
이찬

진행사항 및 공부내용





COM5 : USB2CAN
COM7 : Cortex-R5

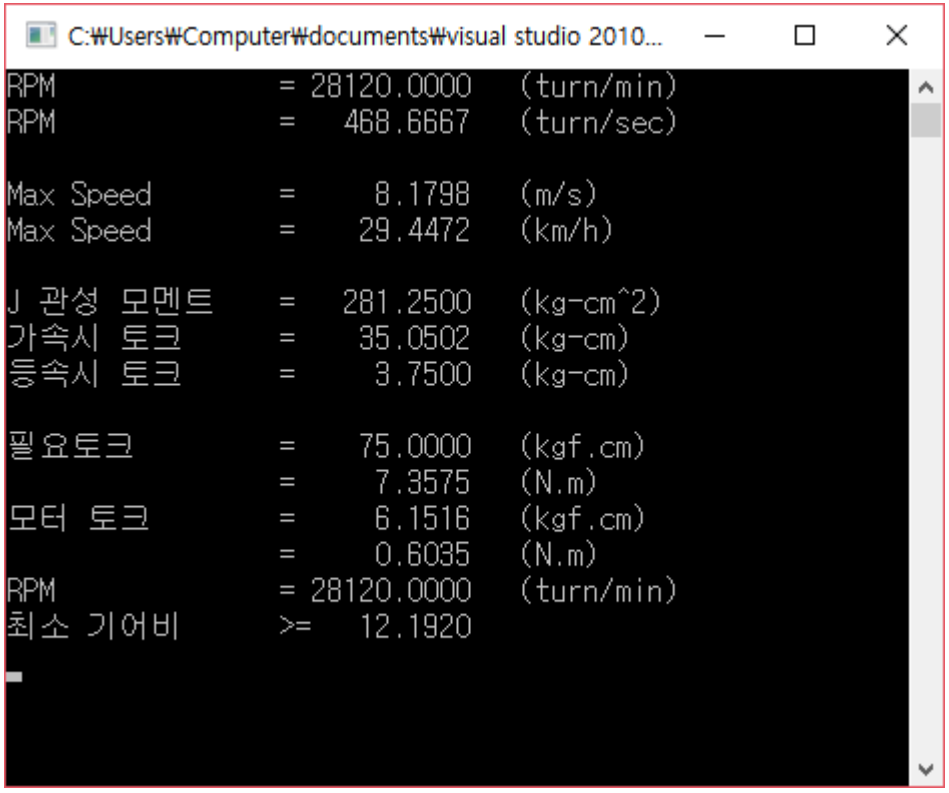
TX-RX

Motor Control
이찬

진행사항 및 공부내용

```
PI = 3.141592;          // 원주율
r = 7.5;                // 바퀴 반지름
gear = 27.0;            // 기어비
kv = 1900.0;            // 모터
current = 120.0;        // 최대 전류
volt = 14.8;            // 입력 전압
J = 0.0;                // 관성 모멘트
W = 10.0;               // 중량
B = 0.1;                // 마찰계수
g = 980.0;              // 중력 가속도 980cm/s^2
t = 1.0;                // 가속 샘플링 시간
N_m = 9.81*0.01;        // 중력가속도 Nuton meter

rpm = kv*volt;
printf("RPM₩t₩t = %10.4f₩t(turn/min)₩n", rpm);
printf("RPM₩t₩t = %10.4f₩t(turn/sec)₩n₩n", rpm/60);

max_speed = (2*PI*r*(rpm/(60*gear)))/1;
printf("Max Speed₩t = %10.4f₩t(m/s)₩n", max_speed/100);
printf("Max Speed₩t = %10.4f₩t(km/h)₩n₩n", max_speed*3600/100000);

J = W * (r*2)*(r*2) / 8;
Ta = J/g*2*PI*rpm/60/gear/t;
Tm = B * W * 2*r / 4;
printf("J 관성 모멘트₩t = %10.4f₩t(kg-cm^2)₩n", J);
printf("가속시 토크₩t = %10.4f₩t(kg-cm)₩n", Ta+Tm);
printf("등속시 토크₩t = %10.4f₩t(kg-cm)₩n", Tm);


/*******************************************************/
need_T = r*W;
motor_T = 97400*volt*current/1000/(kv*volt);
min_gear = need_T/motor_T;

printf("₩n");
printf("필요토크₩t = %10.4f₩t(kgf.cm)₩n", need_T);
printf("₩t₩t = %10.4f₩t(N.m)₩n", need_T*N_m);
printf("모터 토크₩t = %10.4f₩t(kgf.cm)₩n", motor_T);
printf("₩t₩t = %10.4f₩t(N.m)₩n", motor_T*N_m);
```

Motor Spec
RC car Modeling

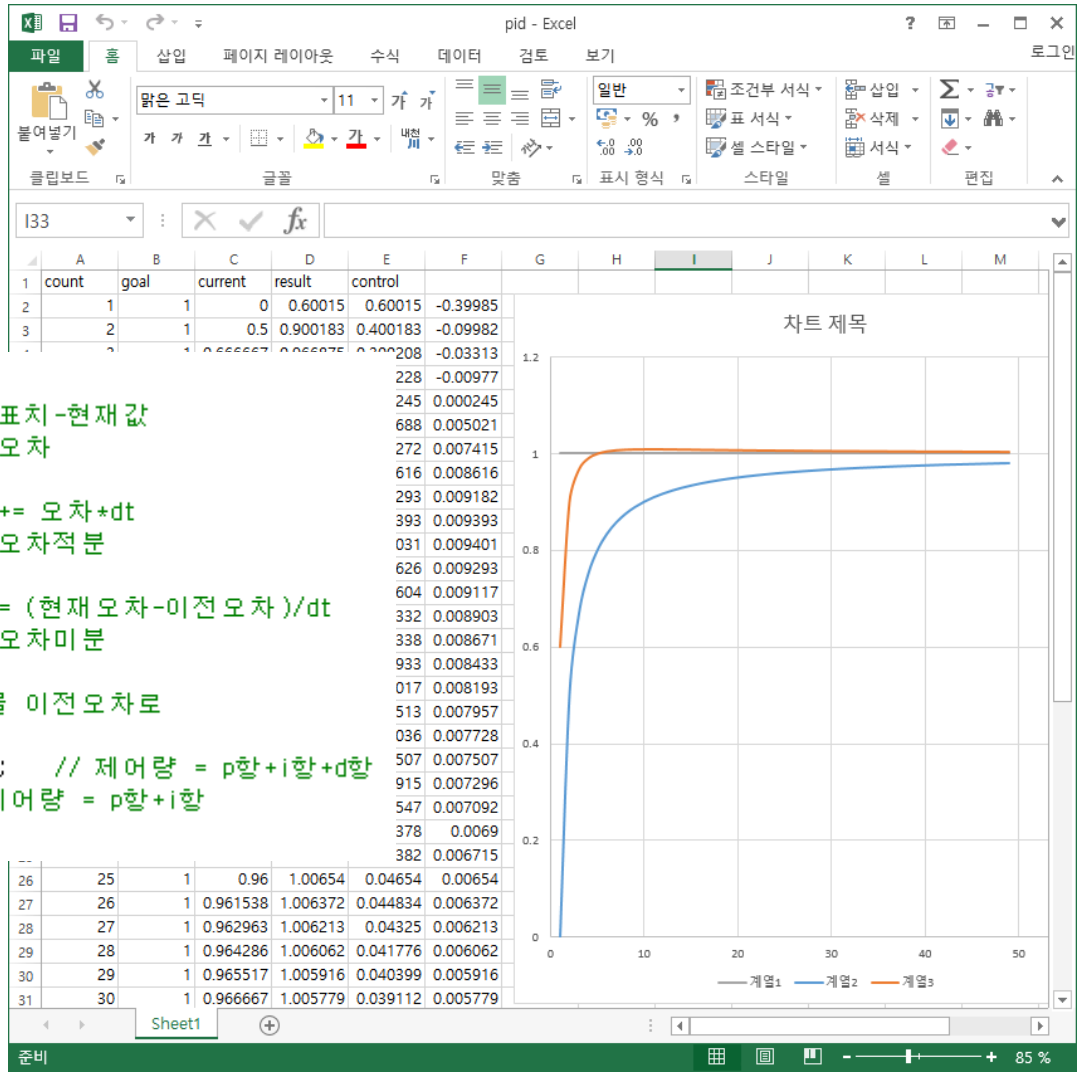


Motor Control
이찬

진행사항 및 공부내용

PID Control Visualization (Excel)

```
for(i=0;i<50;i++){
    err = AIM - tmp[i];          // 오차 = 목표치-현재값
    Kp_term = Kp * err;          // p항 = Kp*오차

    I_err += err * dt;           // 오차적분 += 오차*dt
    Ki_term = Ki * I_err;        // i항 = Ki*오차적분

    D_err = (err-prev_err)/dt;   // 오차미분 = (현재오차-이전오차)/dt
    Kd_term = Kd * D_err;        // d항 = Kd*오차미분

    prev_err = err;              // 현재오차를 이전오차로

    control[i] = Kp_term + Ki_term + Kd_term;   // 제어량 = p항+i항+d항
    //control[i] = Kp_term + Ki_term;   // 제어량 = p항+i항
    //control[i] = Kp_term; // 제어량 = p항
```



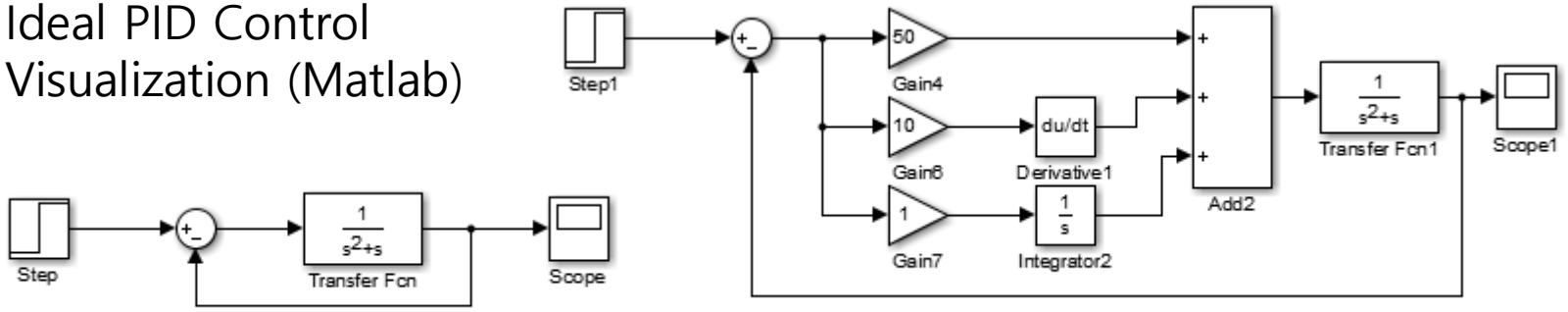
	A	B	C	D	E	F
1	count	goal	current	result	control	
2	1	1	0	0.60015	0.60015	-0.39985
3	2	1	0.5	0.900183	0.400183	-0.09982
26	25	1	0.96	1.00654	0.04654	0.00654
27	26	1	0.961538	1.006372	0.044834	0.006372
28	27	1	0.962963	1.006213	0.04325	0.006213
29	28	1	0.964286	1.006062	0.041776	0.006062
30	29	1	0.965517	1.005916	0.040399	0.005916
31	30	1	0.966667	1.005779	0.039112	0.005779

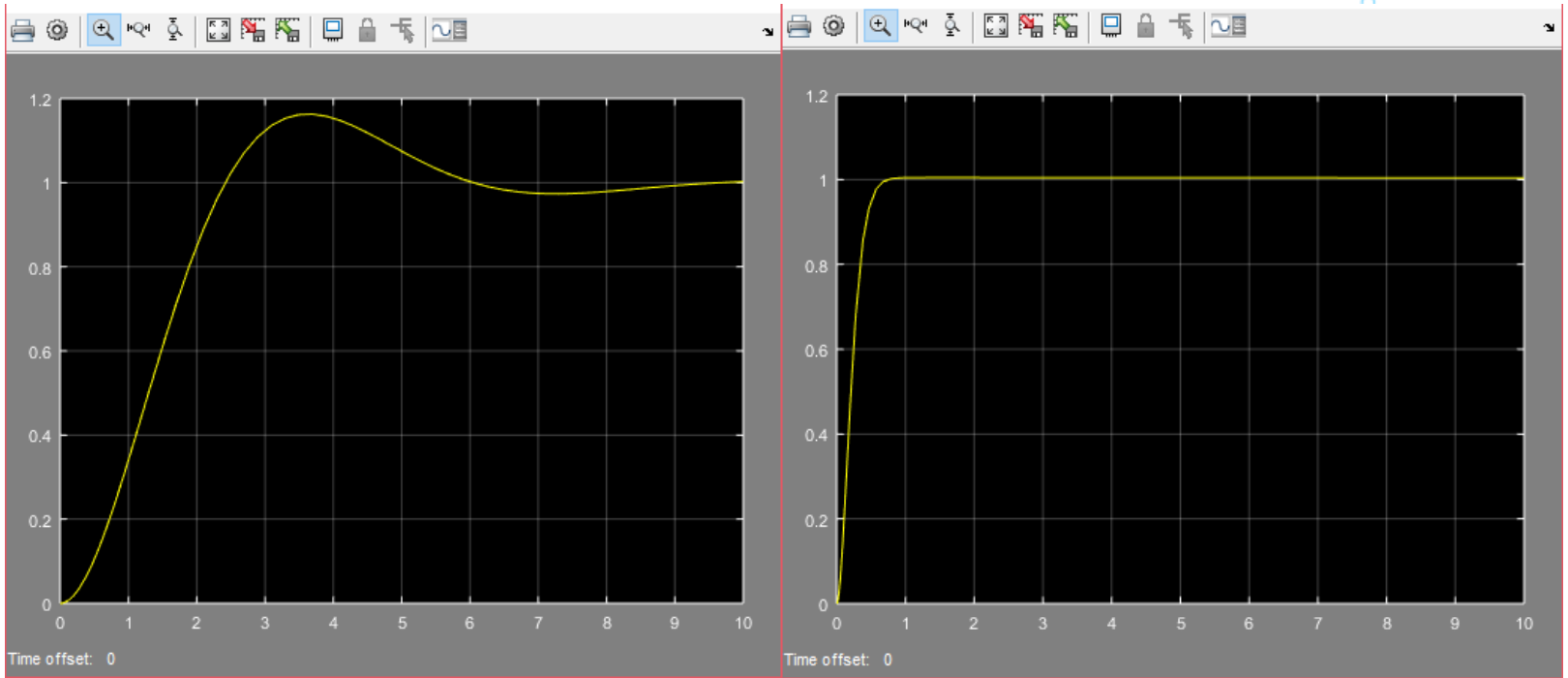
208 -0.03313
228 -0.00977
245 0.000245
688 0.005021
272 0.007415
616 0.008616
293 0.009182
393 0.009393
031 0.009401
626 0.009293
604 0.009117
332 0.008903
338 0.008671
933 0.008433
017 0.008193
513 0.007957
036 0.007728
507 0.007507
915 0.007296
547 0.007092
378 0.0069
382 0.006715

Motor Control
이찬

진행사항 및 공부내용

Ideal PID Control Visualization (Matlab)

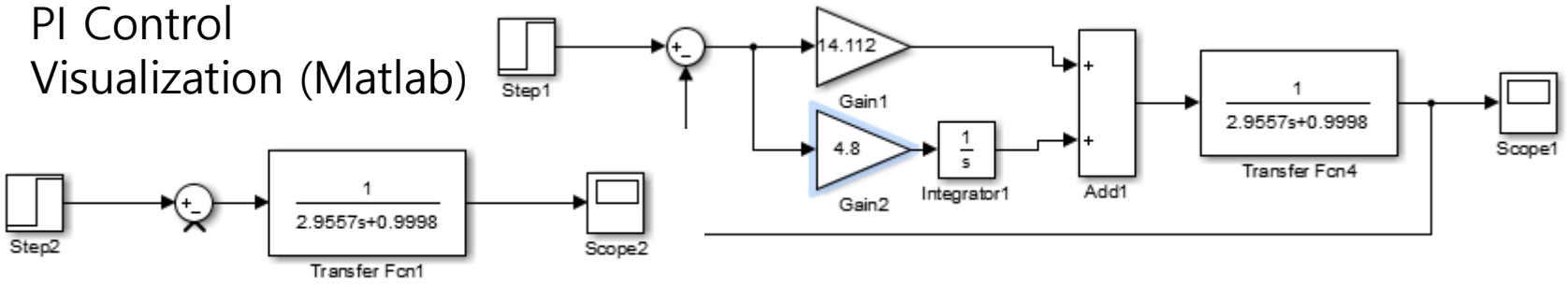


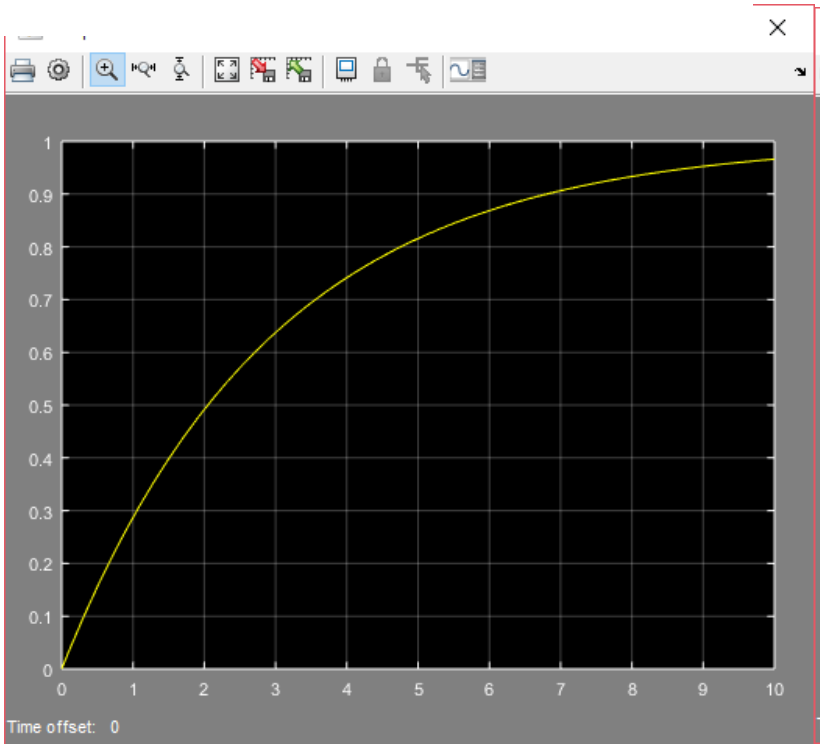


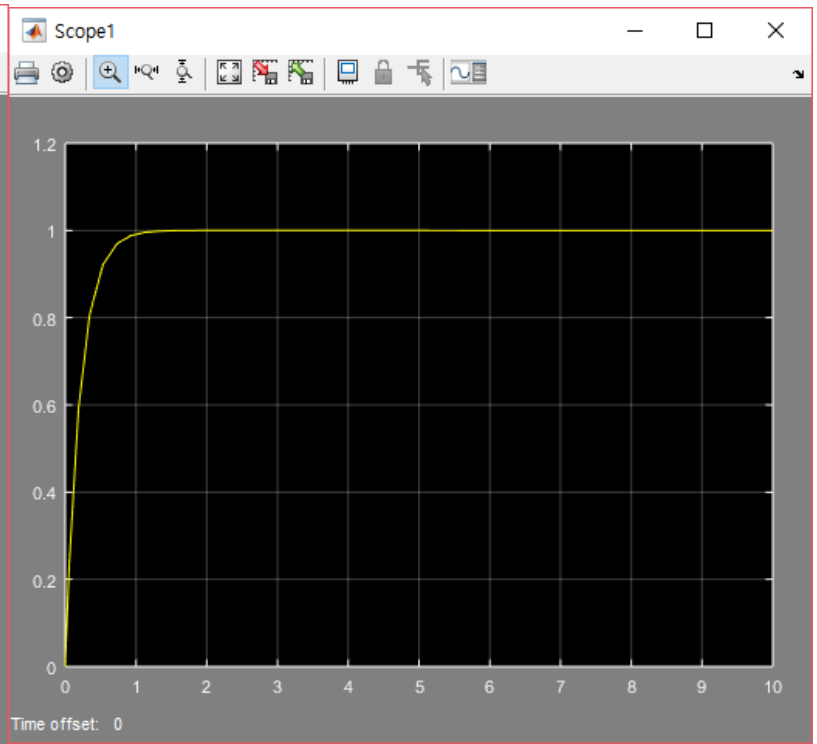
Motor Control
이찬

진행사항 및 공부내용

PI Control
Visualization (Matlab)







CAN, Motor Control
이찬

문제점

2016. 06. 08.
- cortex r5 - USB2CAN 모듈 송수신 실패
  => ID, MASK, BitRate 세가지 값을 송수신 모두 같게 맞춤으로써 해결.
  또한, 모듈에서 요구하는 전송형태를 유지하여야 전송 가능.

2016. 06. 10.
- DSP - cortex r5 수신 값 확인 실패
  => binary값을 text형식으로 변환하여 송신.
  binary를 그대로 출력할 경우 값이 깨졌으나, text로 출력하여 식별 가능하도록 하였음.

2016. 06. 14.
- 보드간 CAN통신 완료
  => 전송값 총 32bit 중 각각 8bit씩 나누어 동시 네 종류의 값 처리.

2016. 06. 20
- PID 기본 개념 이해 및 도식화
  => 실제 모터구동과 유사한 임의의 값을 생성하여
  PID 각각의 제어가 어떤 역할을 하는지 그래프를 통해 확인.

2016. 06. 27
- BLDC 모터 스펙 이해, 모델링 이해 및 적용 어려움
  => 전달함수(Gain) 도출에 있어 변수 각각의 의미와 그 변수 값을 구하는 과정에서
  이해하느라 시간이 많이 소모됨. 적당한 타협으로 근사치의 식을 구할 수 있었음.

CAN, Motor Control
이찬

문제점

2016. 07. 04 ~ 현재 (잠시 보류)
- 모터 정밀제어를 위한 PI제어 적용 시도중...
  => 엔코더를 통해 들어오는 모터 실제 속력을 받아서 처리해야 함.

2016. 07. 07.
- CAN통신을 통해 들어온 값 중 조향에 관여하는 각도 값을 추출하여 서보 모터(조향) 제어 확인
  => 만약 차선 각도가 1도 틀어졌을 때 서보 모터도 1도만 틀 경우
  실제 주행에 문제가 없는지는 추후 실험을 통해 다시 보아야 함.

2016. 07. 12. ~ 현재
- Matlab을 통해 전달함수와 제어기를 거친 후의 그래프 확인 결과 모터 모델링에 이상을 감지하여 다른 모델링 기법 도입 시도 중.
- 또한 PI 계수를 도출함에 있어 기존에 지글러-니콜스(Ziegler-Nichols) 동조법에 기초하여 도출하였지만, 도출 후 조정 값이 훨씬 더 커지는 상황이 발생하여 다른 방법 모색 중.

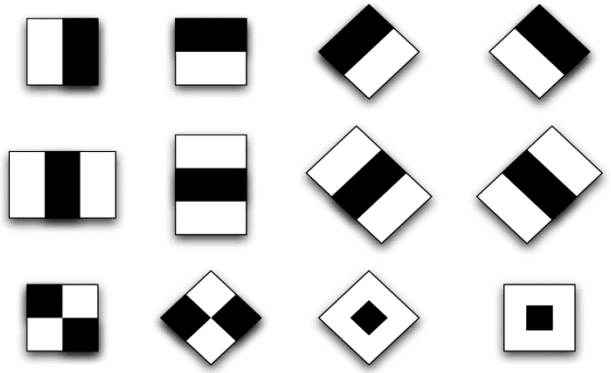
진행 내용

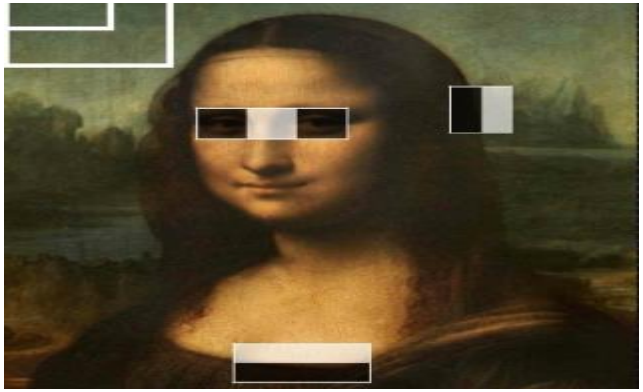
Object Detection
Parallel Processing

이충환

Object Detection, Parallel Processing
이충환

– Haar like feature





– Integral Image

10	15	7	9	5	6	4	2
9	32	65	45	12	7	5	2
7	24	66	65	41	34	4	7
9	11	70	89	44	37	42	11
32	78	91	78	48	65	15	24
64	12	89	58	65	45	37	9
1	1	96	89	56	48	59	3
2	3	45	65	44	71	57	4

Original image

10	25	32	41	46	52	56	58
19	66	138	192	209	222	231	235
26	97	235	354	412	459	472	483
35	117	325	533	635	719	774	796
67	227	526	812	962	1111	1181	1227
131	303	691	1035	1250	1444	1551	1606
132	305	789	1222	1493	1735	1901	1959
134	310	839	1337	1652	1965	2188	2250

Integral image

Original image :

89+44+27+78+48+65+58+65+45 = 529

Intergral image :

1444-459-691+235 = 529

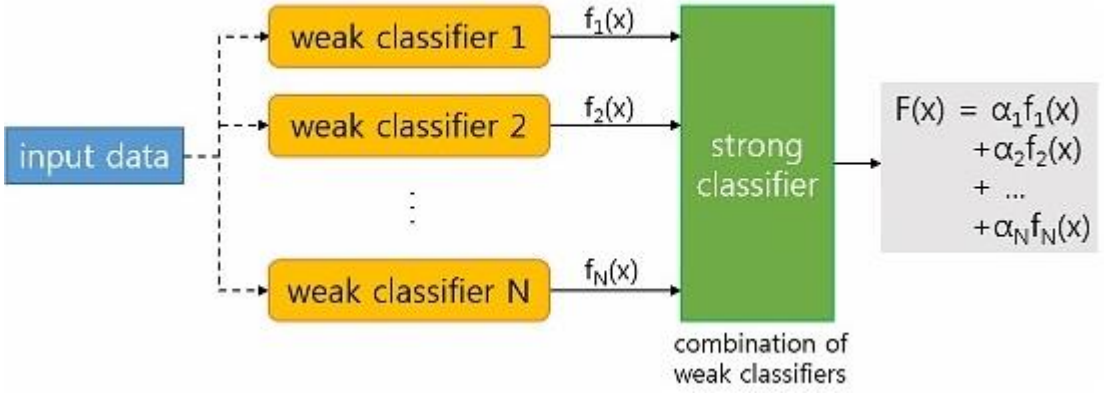
– Adaboost

Boosting Algorithm

약한 분류기를 반복적으로 이용, 강한 분류기를 만드는 알고리즘

Adaptive Boosting Algorithm (AdaBoost)

적응형 부스팅 알고리즘. 약 분류기에 가중치를 주어 정확도를 높이는 부스팅 알고리즘

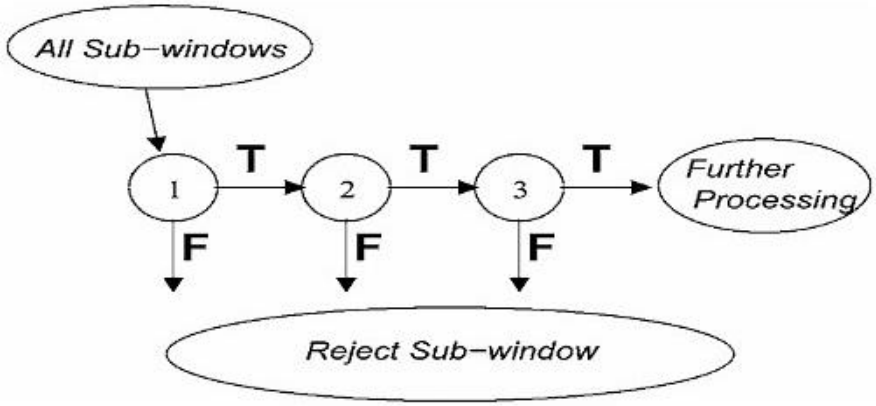


Strong classifier

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

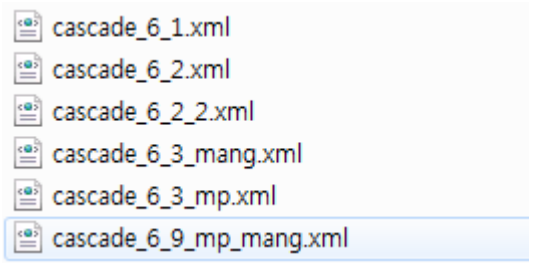
Object Detection, Parallel Processing
이충환

– Cascade Methode



– Machine Learning

Positive & Negative image

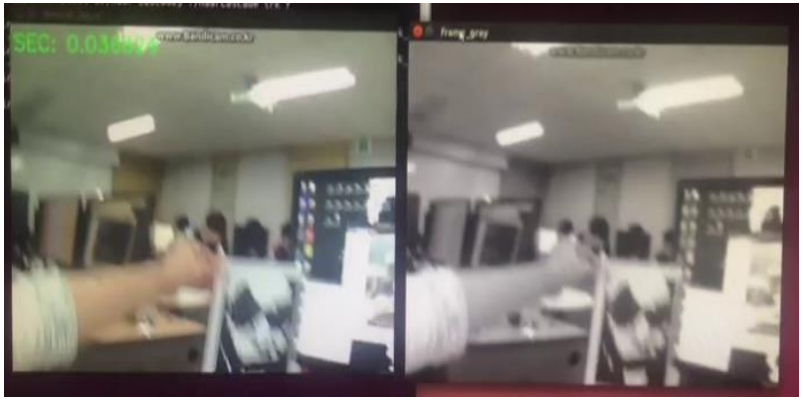


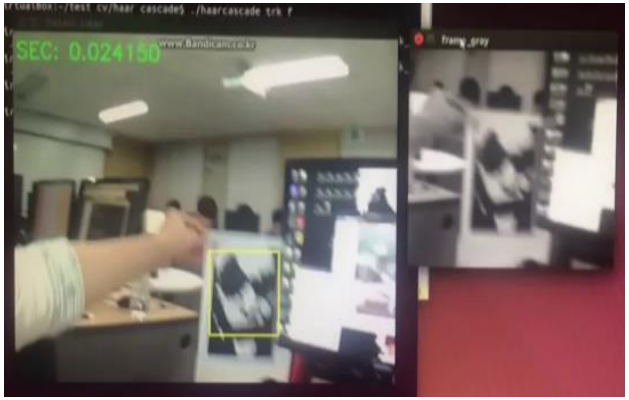
XML file

Object Detection, Parallel Processing
이충환

– Detection

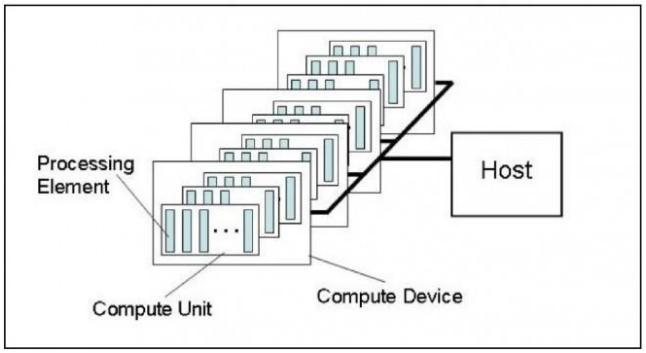
– Tracking



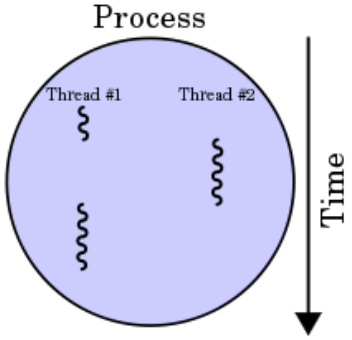


Object Detection, Parallel Processing
이충환

– Parallel Processing



Opencl



Thread

– Noise 제거

Gaussian Blur

Object Detection, 병렬 처리
이충환

- 문제점(1/2)

~ 2016. 06. 4.
- 영상처리 알고리즘 결정. Color로 할지, feature로 할지. Color는 빛에 영향을 너무 많이 받음.

~ 2016. 06. 10.
- 연산양은 많지만 보다 높은 정확도를 얻을 수 있는 Feature detecting으로 결정. Haar Cascade Classifier를 사용 하기로 함. 하지만 연산이 너무 많아 동영상 FPS값이 매우 낮아짐. 정확도 이외에 해결 해야 할 문제임.

~ 2016. 06. 19.
- Harr feature 학습을 위해, 약 17,000장의 사진 샘플을 이용함. 하지만 학습하는데 시간이 너무 오래 걸림. 학습 결과 xml파일이 생성되는데, 거의 일주일이 걸림. 그래서 PC로는 학습을 하고, 따로 Haar Cascade Classifier 관한 논문 분석을 시작. 이후 Opencv에 구현돼 있는 코드 분석을 할 예정.

~ 2016. 06. 26.
- 논문 분석 후, Opencv에 구현되어있는 코드를 분석함. 이거 생각보다 꾀나 오래 걸림. 그리고 생각 보다 매우 복잡함. 처음에는 이 코드를 분석하여 Opencl로 재 코딩 할 생각 이였으나, 그러기에는 시간이 너무 오래 걸릴 것 같음. 그리고 이미 Opencl로 구현이 되어있음.. 코드 분석한 걸로 만족.

~ 2016. 07. 2.
- 본격적인 코드 작성. Cascade Classifier를 두 번 사용하여, 정확도를 높임. 또한 tracking의 개념을 적용하여, 이전 프레임에서 물체를 발견한 부분에 Roi를 설정하여 다음 부분에 넘겨줌. 계산할 부분을 줄여 속도를 높임. 정확도는 괜찮은데 FPS가 15~20정도 밖에 안나옴. 아직 갈 길이 멀음.

~ 2016. 07. 6
- FPS를 높이기 위해, 여러 시도를 함. OPENCL를 이용해, 보드에 있는 DSP칩을 사용해 봤지만 더 느려짐. GPU를 사용하려면 Opengl를 사용해야됨. Tbb는 인텔 cpu 전용임... 속도를 높이기 위한 다른 방법이 필요함.

Object Detection, 병렬 처리
이충환

- 문제점(2/2)

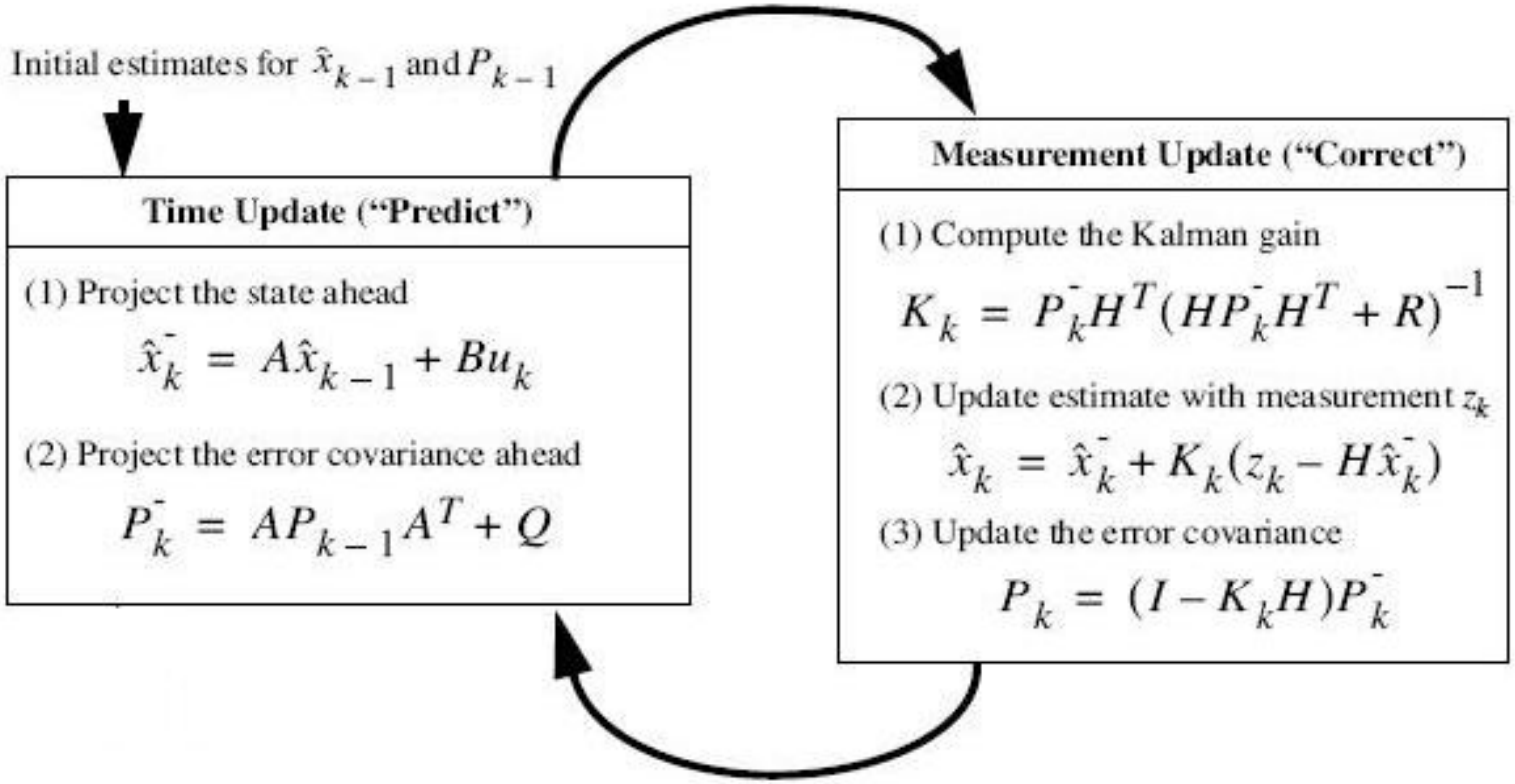
~ 2016. 07. 13
- 3개의 thread를 사용해 coding 단에서 처리를 해봄. Intel CPU상에서는 더 빨랐지만, DSP보드에 올리니 더 느려심.
  그리고 직접 강당에서 동영상을 찍어 처리해봤는데, 바닥에 빛이 반사돼서 오검출이 좀 있음.

진행 내용

Lane Detection
Kalman Filter

장세진

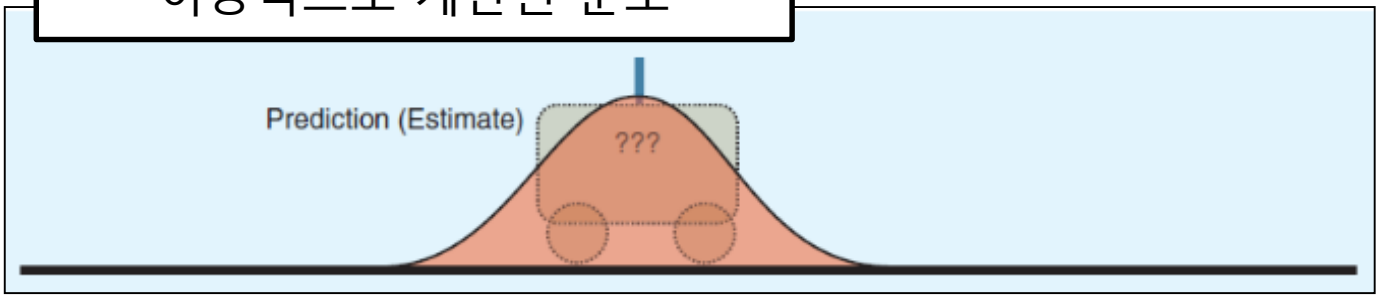
Kalman Filter - 기본식



< 칼만 필터의 기본식 >

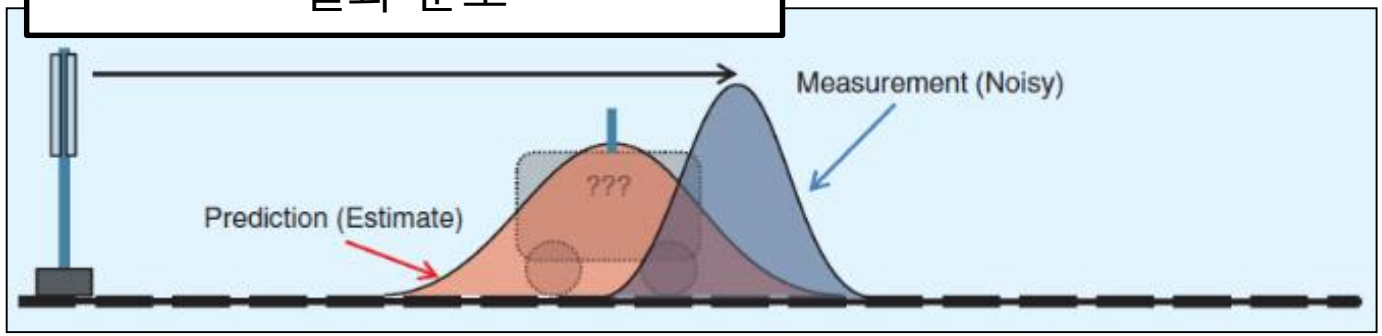
Kalman Filter - 모델링

이상적으로 계산된 분포



초기 위치의 분포

결과 분포



Kalman Filter - 모델링

$$x_{k+1} = Ax_k + Bu_k + w_k$$ Process Noise

빨간색 종 (Q)

$$z_k = Hx_k + v_k$$ Measurement Noise

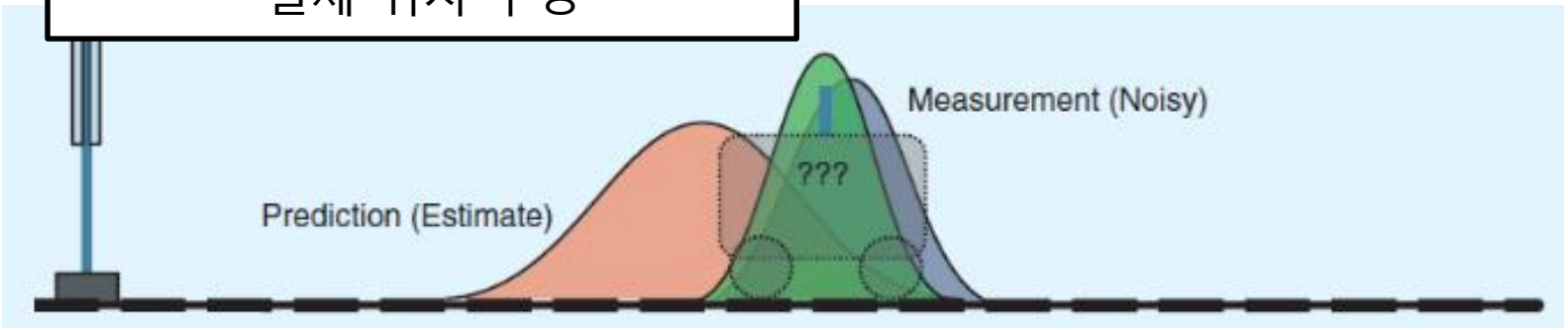
파란색 종 (R)

$$\text{분산} = \sigma^2 = \frac{\sum (X-\bar{X})^2}{n}$$

$$\text{공분산} = Cov[X, Y] = E[(X-\bar{X})(Y-\bar{Y})] = \sigma_{XY}$$

$$= \frac{\sum (X_i-\bar{X})(Y_i-\bar{Y})}{n}$$

실제 위치 추정



Kalman Filter - 모델링

(2) *Update estimate with measurement* z_k

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

$$if\ H = \mathrm{I}$$

모델로부터 계산된 값

$$\hat{x}_k = K_k z_k + (I - K_k)\hat{x}_k^-$$

수정된 값

센서로부터 측정된 값

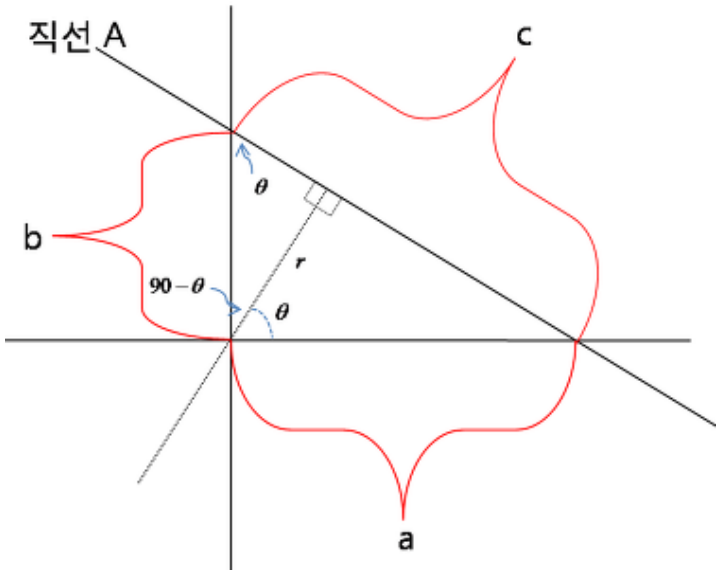
Kalman Gain

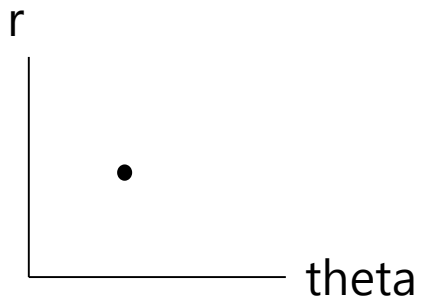
- K가 1에 가까움 : 신뢰도 증가 -> 측정 값을 상태 변수로 간주
- K가 0에 가까움 : 신뢰도 감소 -> 예측 값을 상태 변수로 간주

Kalman Filter - code

나중에

Hough Transform





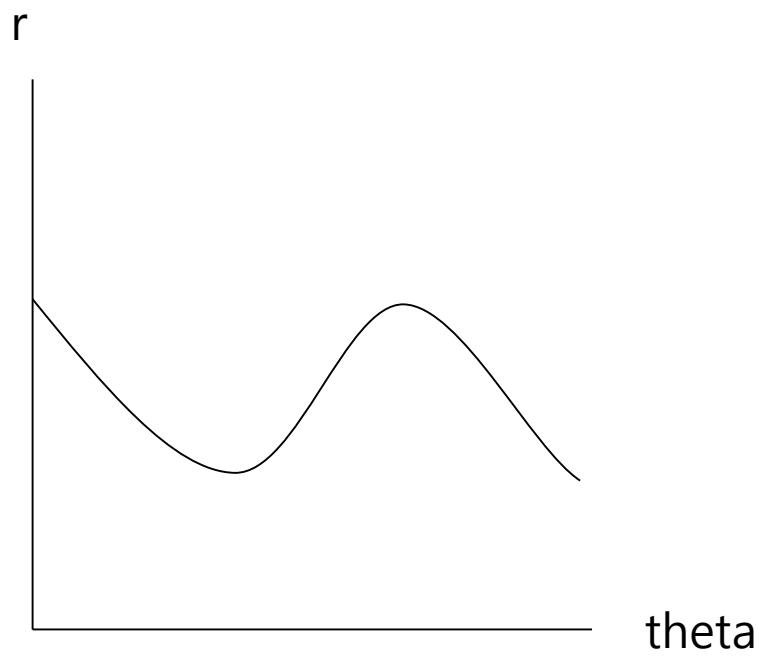
$$xcos\theta + ysin\theta = r$$

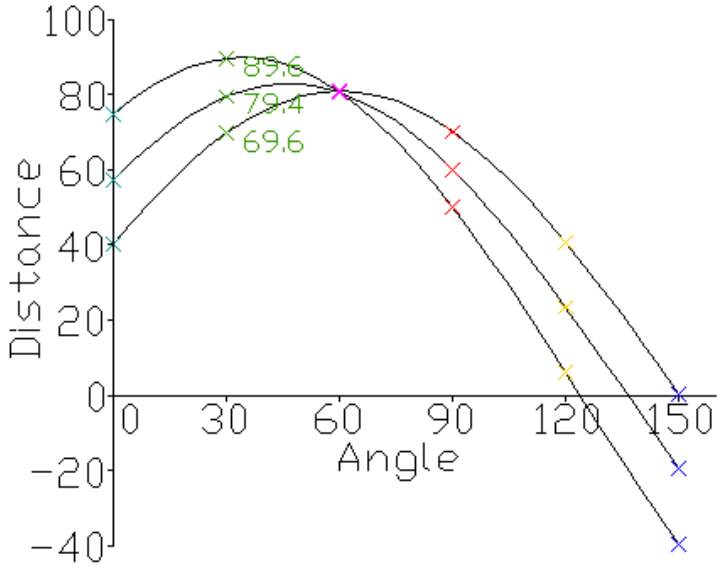
$$a = \frac{A}{B} = -\frac{Cos\theta}{Sin\theta}$$

$$b = \frac{r}{Sin\theta}$$

x, y 좌표 공간에서 같은 직선상에 존재하는 픽셀을 r, theta 매개변수 공간에서 교점을 가지는 특징을 이용하여 직선을 r, theta 좌표 공간으로 mapping 시킨 후 누적된 점을 통해 직선을 검출한다.

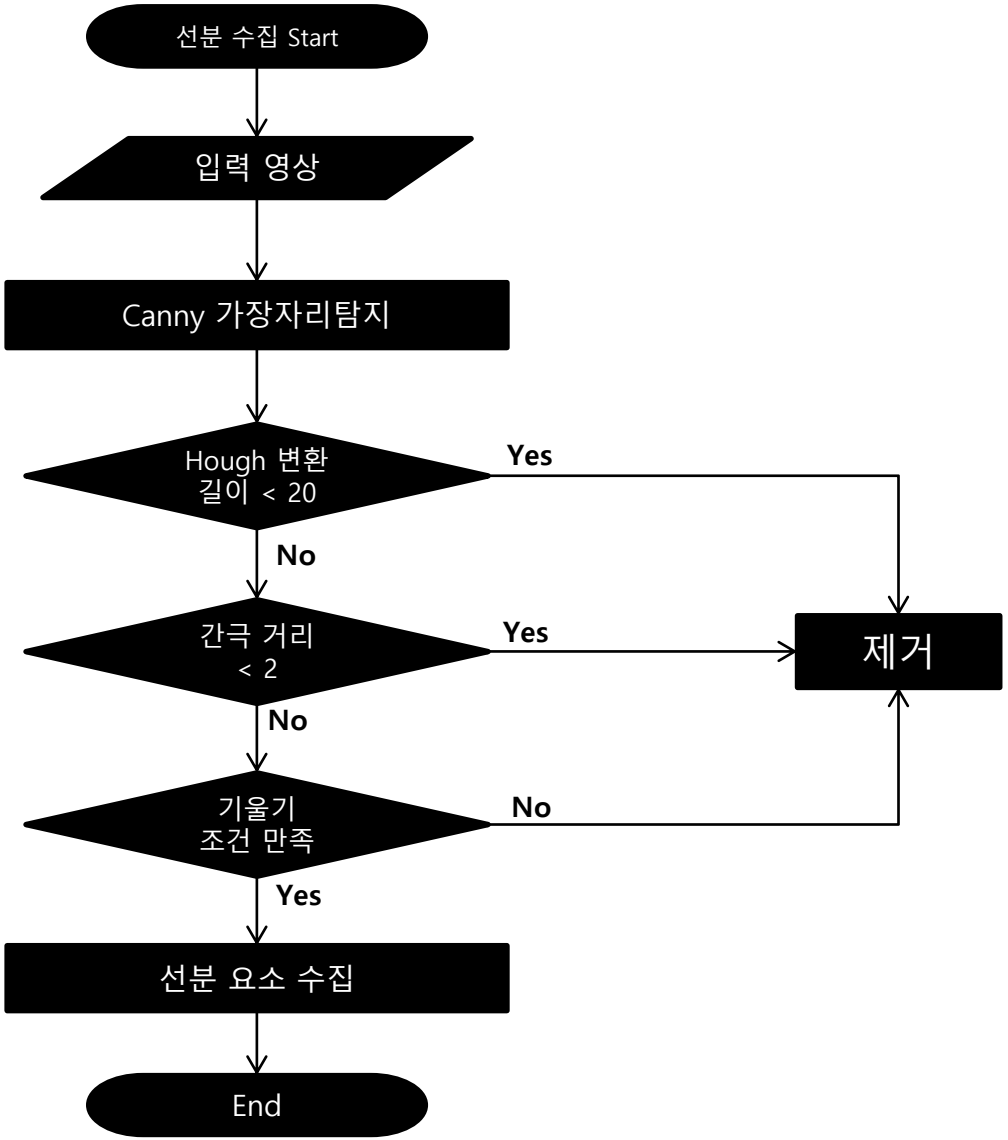
Hough Transform



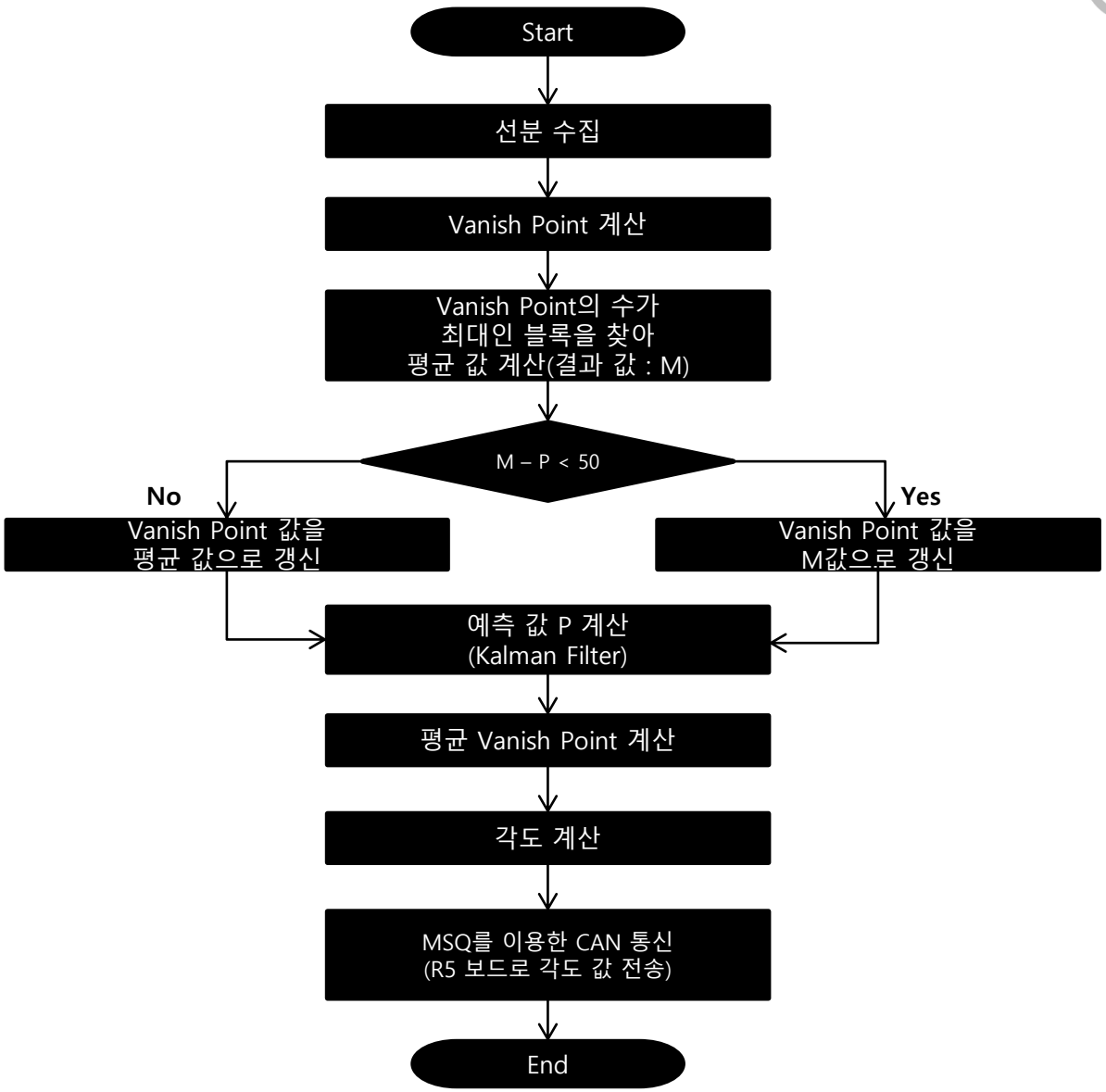


고정된 x, y를 지나는 직선을 theta와 r을 축으로 하는 sin 곡선이 된다.
즉, sin 곡선들의 교점이란 n개의 점이 하나의 직선 위에 있다라는 것을 뜻함.
이를 x, y 평면 위에 직선을 그리면 n개의 점을 가진 직선을 그릴 수 있다.

Flow chart - 선분 수집



Flow chart - 각도 계산



문제점 및 해결방안

나중에

Lane detection, Kalman Filter
장세진

문제점

2016. 06. 10.
- Hough transform 함수의 이해 부족으로 원하는 선만을 추출 불가
  =>곡선 추출 및 속도 개선을 위해 roi를 사용하여 3등분을 한 후 선의 기울기를 계산하여 일정부분 없애고 소멸점을 이용하여 잡음 성분을 추가적으로 없앰.

2016. 06. 17.
- 선들의 기울기 계산과 일정 x값 범위 내에서 선이 있다면 같은 선으로 간주해 제일 오른쪽만을 따오는 과정에서 많은 오차가 생김.
  => 오차를 줄이기 위해 노력해 봤지만 실패, 알고리즘을 바꿈.

2016. 06. 24.
- 가장 큰 문제점인 비슷한 위치의 선들을 하나의 선만 따오지 않고 모든 선의 소멸점을 구한 후 7등분 블록을 나누어 개수를 세어 가장 많은 위치의 평균을 구해 유효한 소멸점을 찾음.
  => 유효한 선보다 잡음이 많으면 안되는 문제점이 있음.

2016. 07. 1.
- 칼만필터 초기값 설정 및 측정오차공분산과 예측오차공분산의 설정에서 이론에서 covariance matrix 라고 되어있으나 배열로 구현하면 다른값 계산에서 문제 발생
  => 배열로 하지 않고 하나의 값으로 놓고 적용, 초기값과 코드를 수정함.

2016. 07. 4.
- 프레임에 따라 값이 튀거나 소멸점이 존재하지 않는 경우가 생기면 예측값 계산에서 에러가 생김.
  => 측정값(유효한 소멸점들의 평균)의 평균을 계산해 두고 이러한 경우 평균값을 대신 측정값으로 줌.

- R5에서 현재 속도를 넘겨주면 계산된 각도를 이용해 얼마의 속도를 줄지 계산해서 넘겨줘야 하는데 선의 곡률을 따온게 아니라 소멸점을 기준으로 각도값만을 계산한거라 어떻게 해야할지 감이 안옴.

현재 진행 상황 - 문제점

현재
 - 실제 트랙을 구성했을 때, 선이 하나만 보이는 문제

현재 진행 상황 - 문제점

현재
- 캠으로 받았을 때, segment fault가 뜸.
  -> 동영상으로 혼자서 돌렸을 때는 잘 돌아갔는데, 합친 후 캠으로 돌려봤을 때 segment fault가 뜨면서 종료가 되는데 아직 원인을 알 수 없음.

```
test
return ang : 34
----------------------------------------

test
return ang : 34
----------------------------------------

test
Segmentation fault
```

- The End -