
신호처리를 이용한 원격제어

- Init 0

작성자

팀장 : 이대영

팀원 : 김민중

이윤성

김태훈

정지윤

프로젝트 매니저 : 이상훈 강사님

1. 개요

2. 적용 기술 및 기능 구현

1) 음성 신호 처리

2) 영상 신호 처리

3) 이더넷 통신 및 MCU 제어

4) 모터 제어

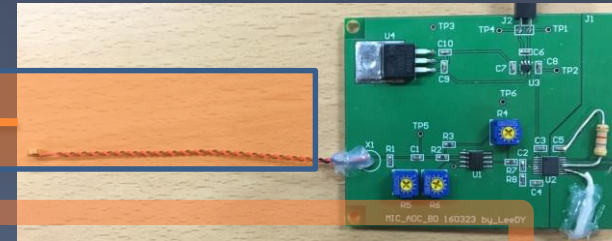
3. 결과

신호처리 네트워크를 활용한 디바이스 제어 시스템

- 음성 및 영상 신호처리를 통해 여러 가지 디바이스를 작동시킬 환경을 구성하고 이 시스템을 가정 및 산업에 적용하여 홈 오토메이션, 공장 자동화를 지원한다.




UDP



66AK2L06
(Keystone II Architecture)

신호처리 네트워크를 활용한 디바이스 제어 시스템

4byte <헤더>	4byte	음성	설명
10001	1	촬영	FPGA B/D로 보냄
10002	2	촬영중, 촬영 중지	FPGA B/D로 보냄
10003	3	불켜	Cortex-M4 B/D로 보냄
10004	4	불꺼	Cortex-M4 B/D로 보냄
10005	5	불업, 블라인드 업	Cortex-M4 B/D로 보냄
10006	6	불따, 블라인드 다운	Cortex-M4 B/D로 보냄
10007	7	구동	Cortex-M4 B/D로 보냄
10008	8	속업, 속력 업	Cortex-M4 B/D로 보냄
10009	9	속따, 속력 다운	Cortex-M4 B/D로 보냄
10010	10	정지	Cortex-M4 B/D로 보냄
10011	11	스냅	FPGA B/D로 보냄  (Ctrl) ▼

이름	포트 번호	IP 주소
상훈	PORT1=7001	IP1=
대영	PORT2=7002	IP2=
민중	PORT3=7003	IP3=
지윤	PORT4=7004	IP4=

모듈별 적용 기술

표 1 - 적용 기술

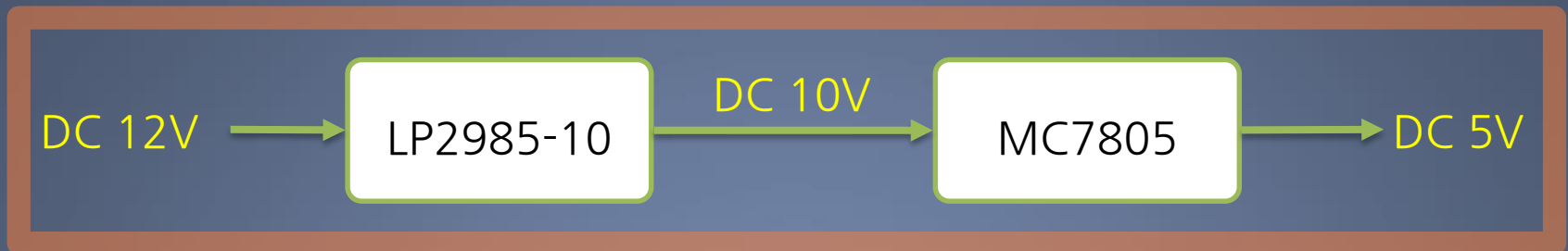
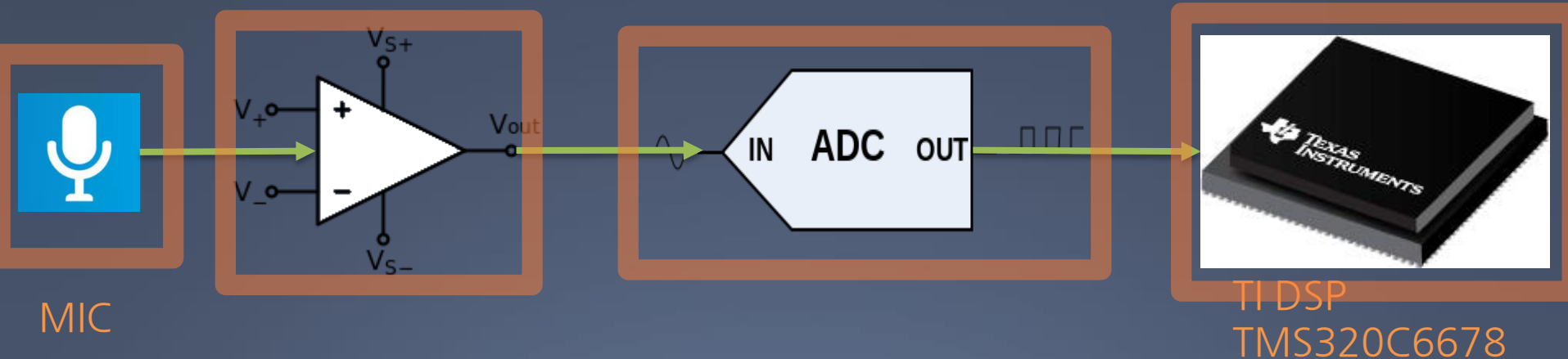
음성 신호 처리	영상 신호 처리	네트워크 구성 및 디바이스 제어	모터 제어
ADC 및 증폭 회로 설계	Visual/IR 카메라	FW 레벨 네트워크 프로그래밍 (UDP 서버 및 클라이언트)	ACI 교류 유도 모터 제어
PCB 설계	모션 인식 영상 프로그래밍	MCU 제어 (STM Cortex-M4)	인버터 설정
TI DSP 활용 (TMS320C6678)	Xilinx FPGA B/D 활용	릴레이 회로 설계	TI 고전압 DSP 활용 (TMS320F2806x)
TI RTOS 구동			타이머와 속도제어
IPC 병렬 프로세싱			PI제어기 이해
FFT 신호 처리			
자연어 해석			

(1) 하드웨어



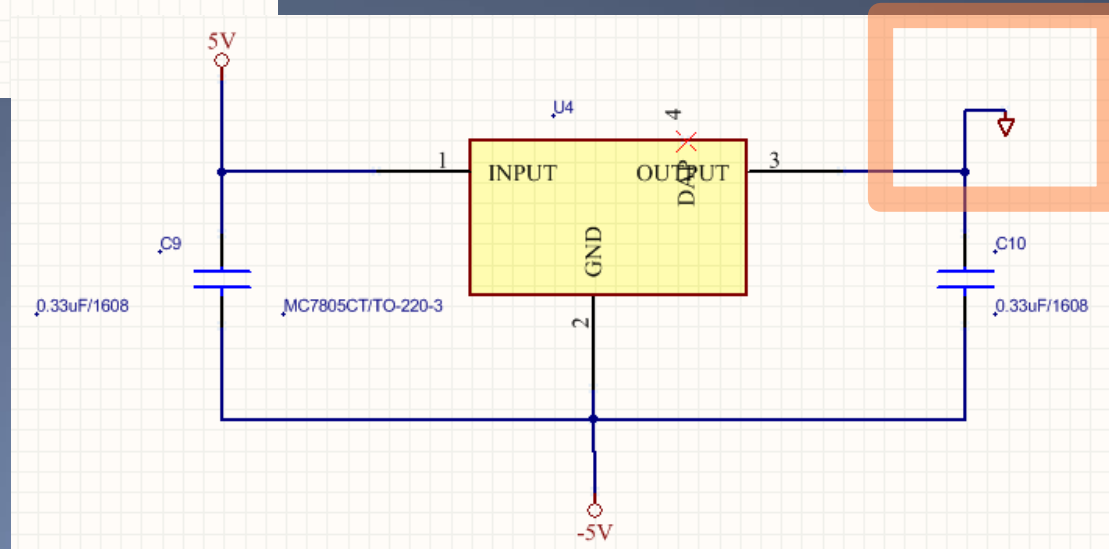
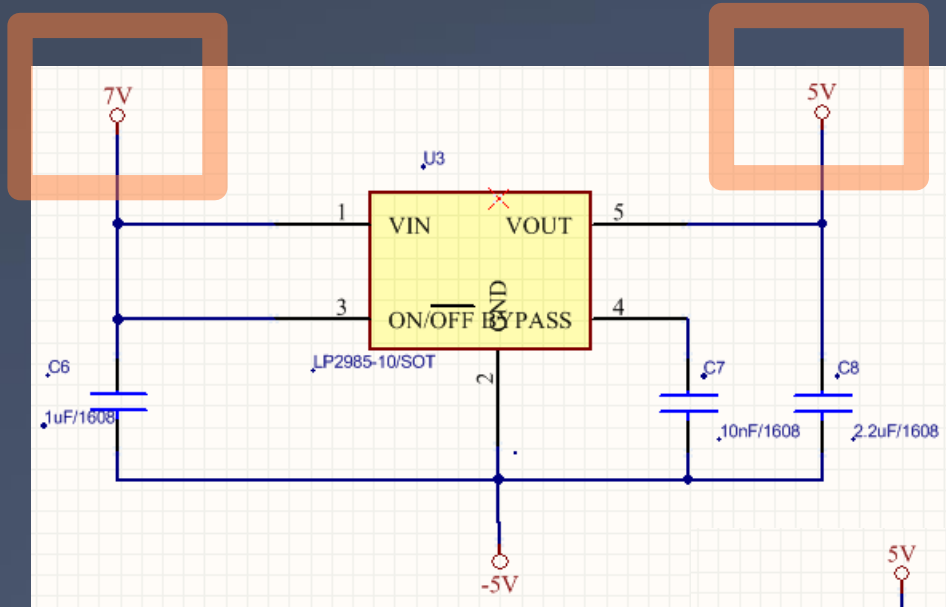
(1) 하드웨어

① 구성도



(1) 하드웨어

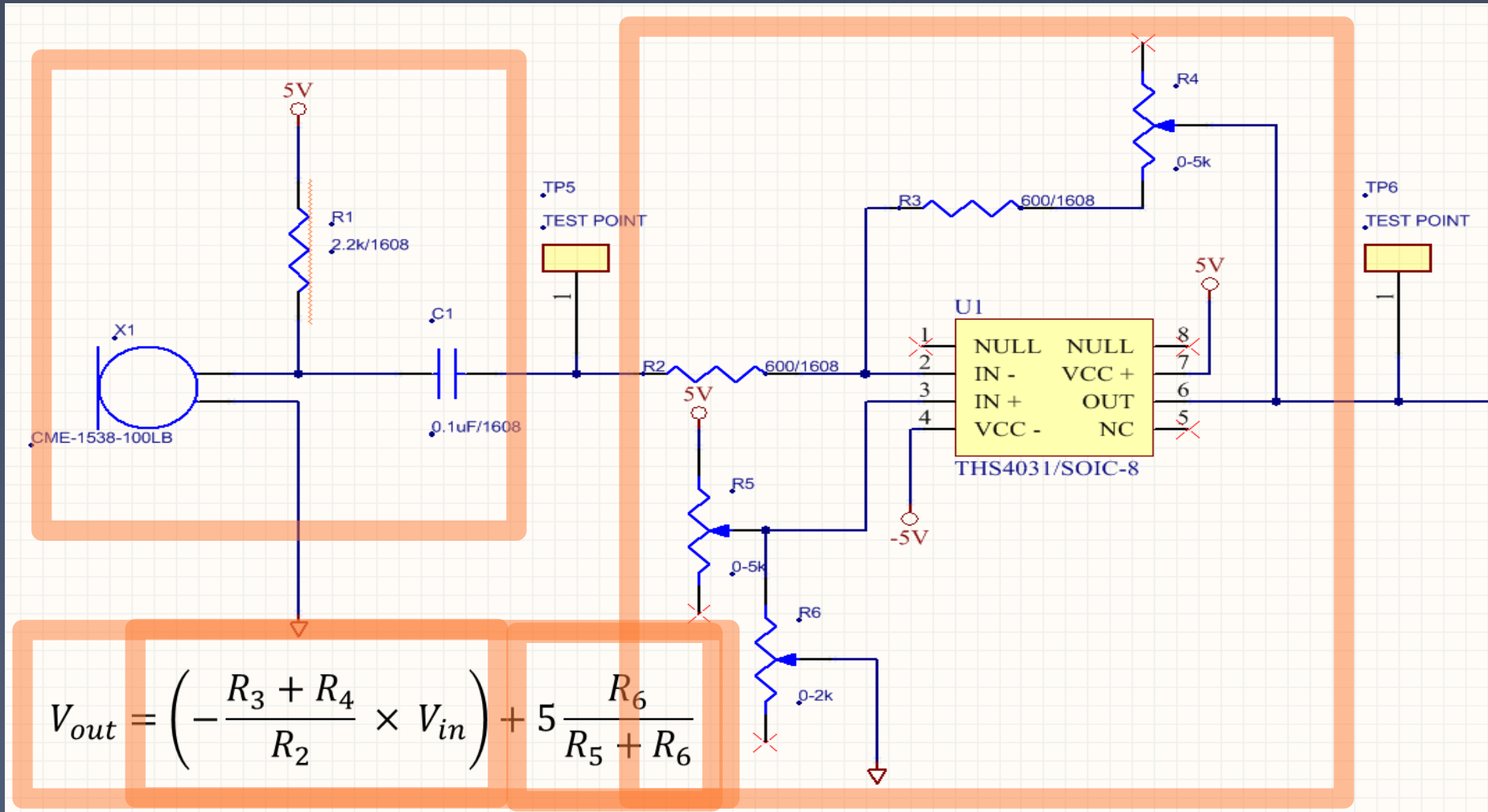
② 회로도 - 전원



(1) 하드웨어

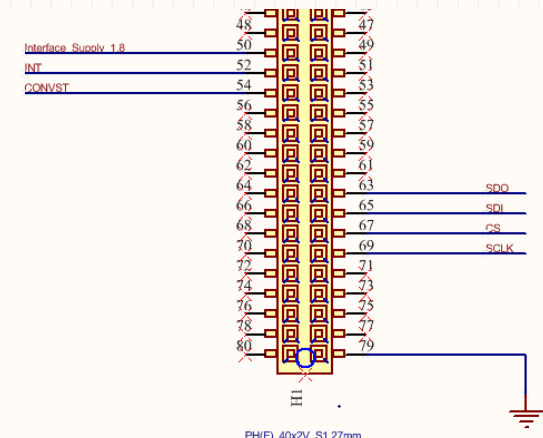
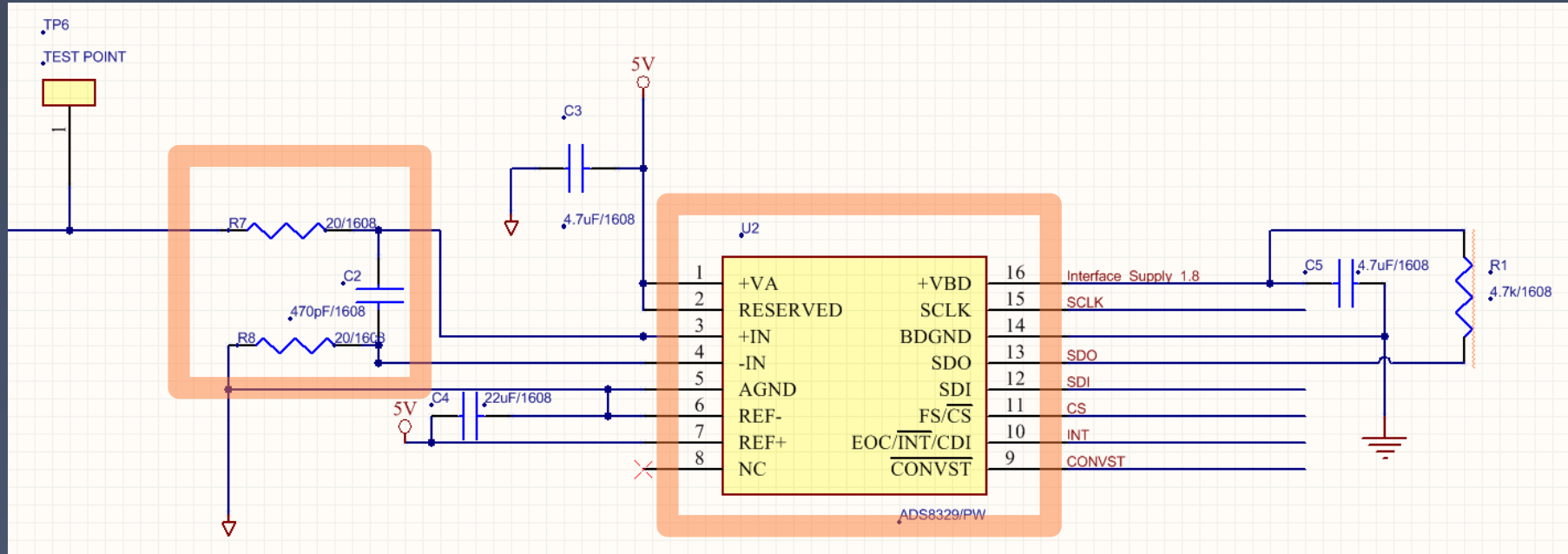
② 회로도 - MIC, 반전 증폭기

- OrCAD, Altium Designer 사용



(1) 하드웨어

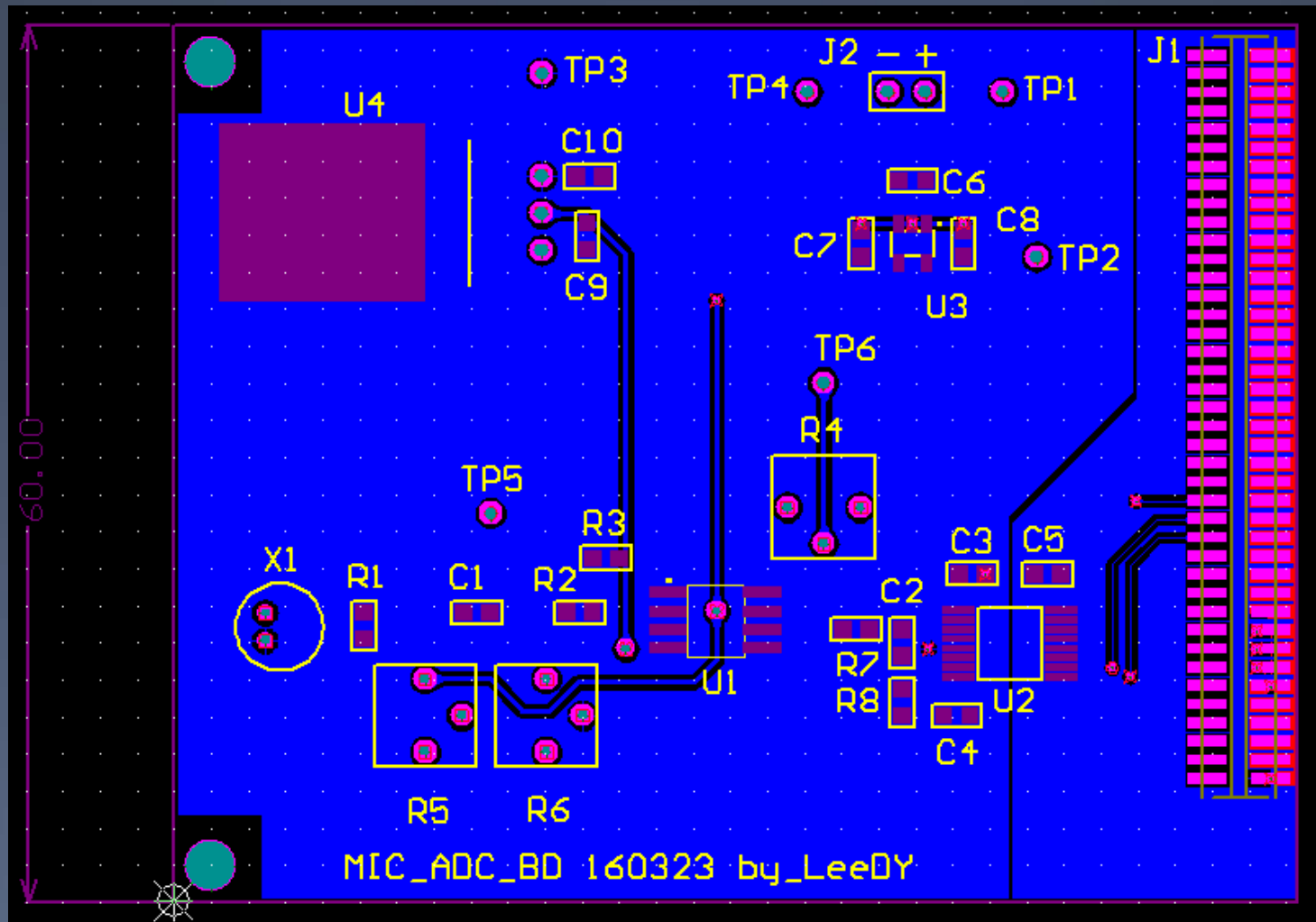
② 회로도 - ADC, SPI



(1) 하드웨어

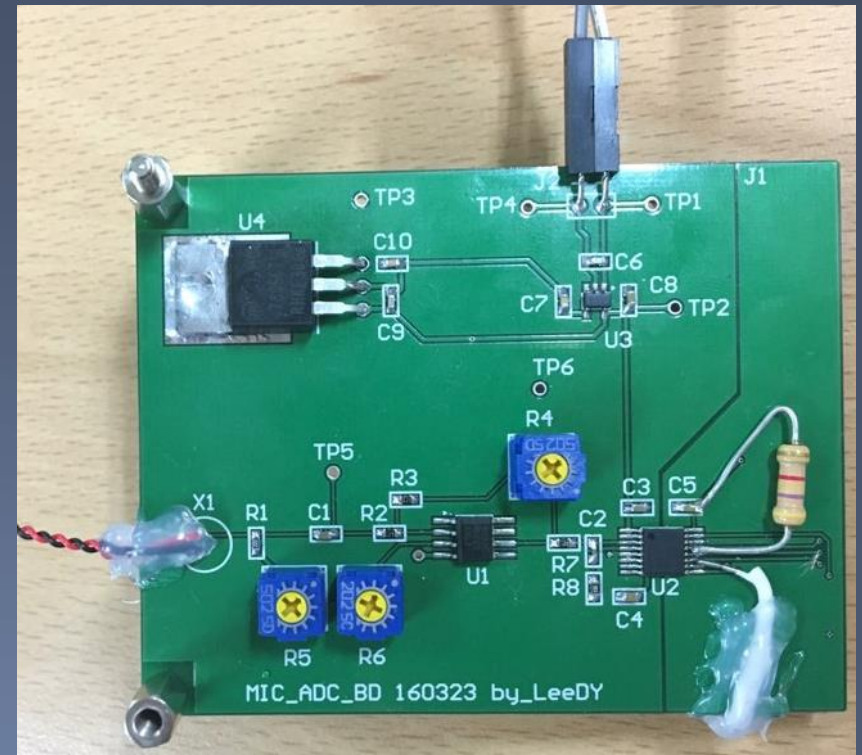
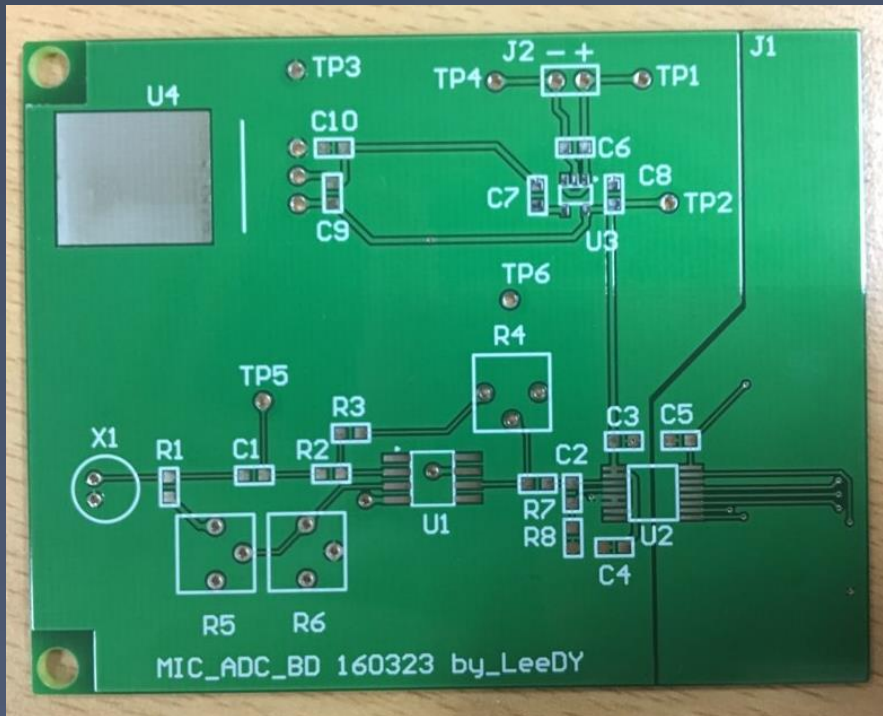
③ PCB

- Altium Designer 사용



(1) 하드웨어

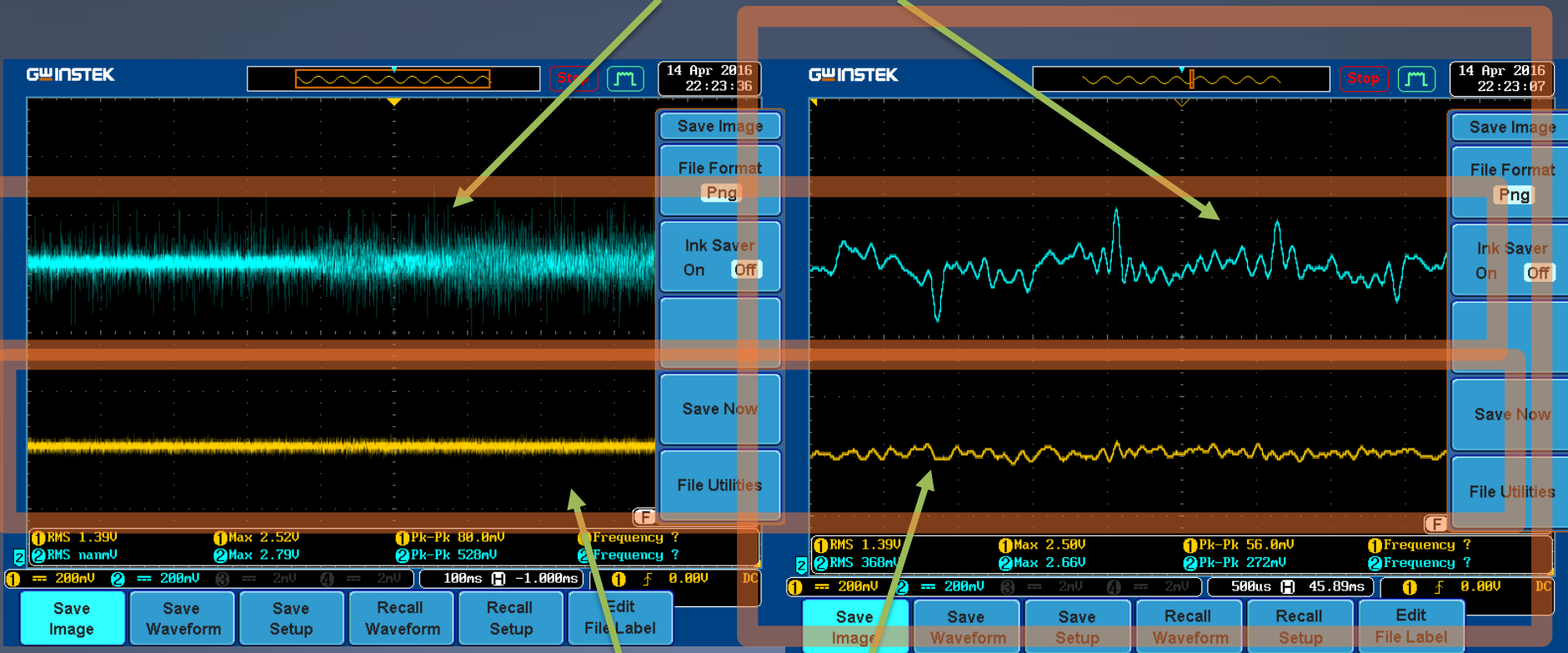
③ PCB



(1) 하드웨어

④ 증폭 파형 측정

증폭기 출력



증폭기 입력(마이크 출력)



(1) 하드웨어

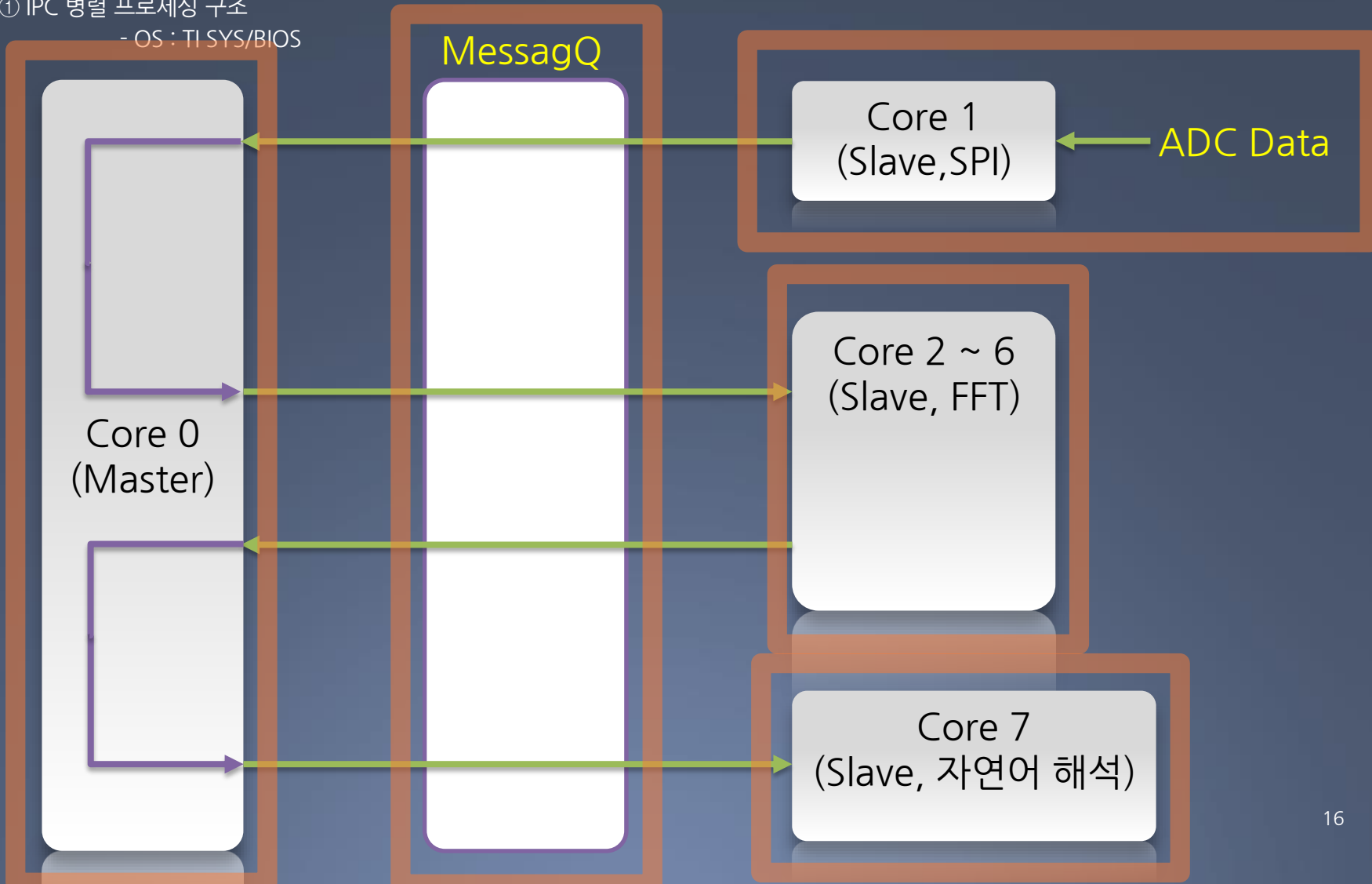
⑤ DSP Board

- 명칭 : TMDXEVM6678LXE
- 코어 : TMS320C6678 (1 GHz, 8 Core)
- RAM : DDR3 512 MB
- NAND, NOR Flash



(2) 소프트웨어

- ① IPC 병렬 프로세싱 구조
- OS : TI SYS/BIOS

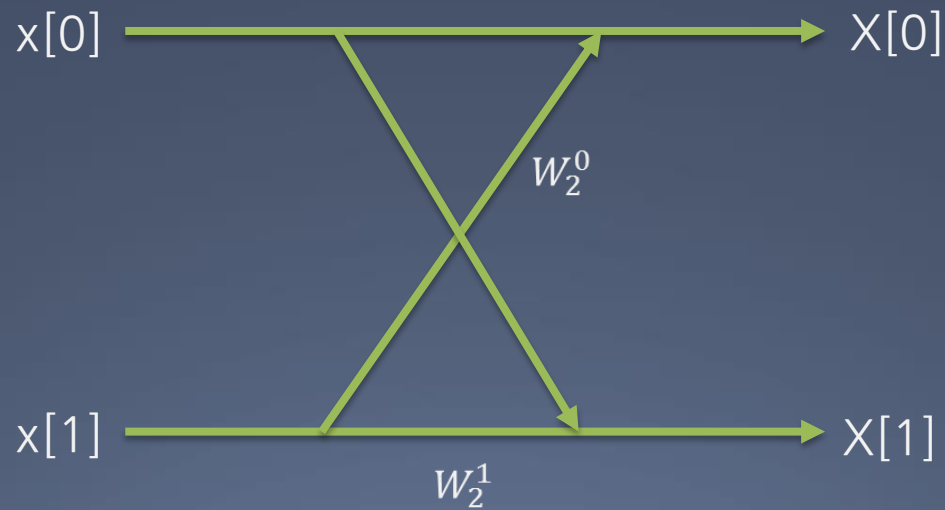


② MessageQ 전달 구조



(2) 소프트웨어

② Fast Fourier Transform (FFT) - Butterfly 연산



$$X[0] = x[0] + W_2^0 * x[1]$$

$$X[1] = x[0] + W_2^1 * x[1]$$

(2) 소프트웨어

② Fast Fourier Transform (FFT) - Butterfly 연산



$$X[0] = G[0] + W_4^0 * H[0]$$

$$X[1] = G[1] + W_4^1 * H[1]$$

$$X[2] = G[0] + W_4^2 * H[0]$$

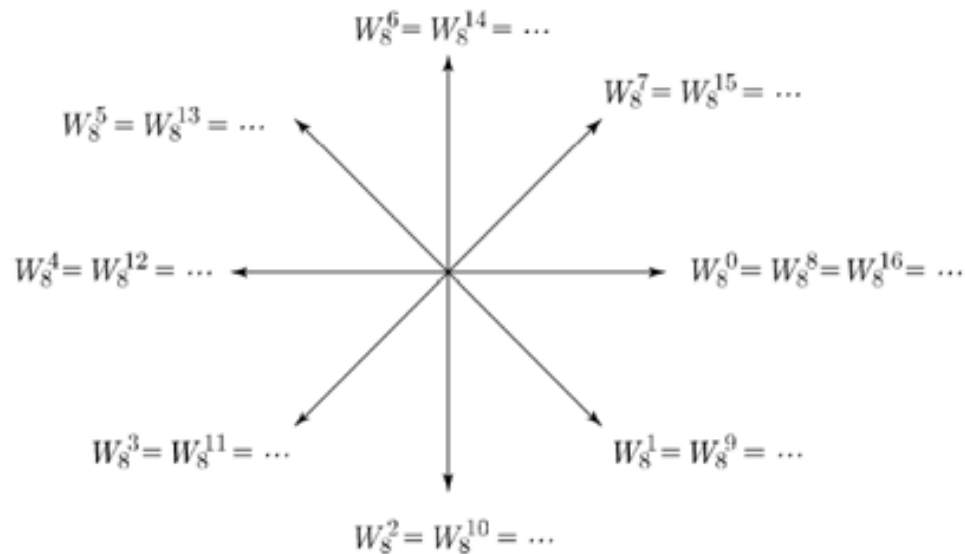
$$X[3] = G[1] + W_4^3 * H[1]$$

(2) 소프트웨어

② Fast Fourier Transform (FFT) - twiddle factor

$$W_N = e^{-j2\pi/N}$$

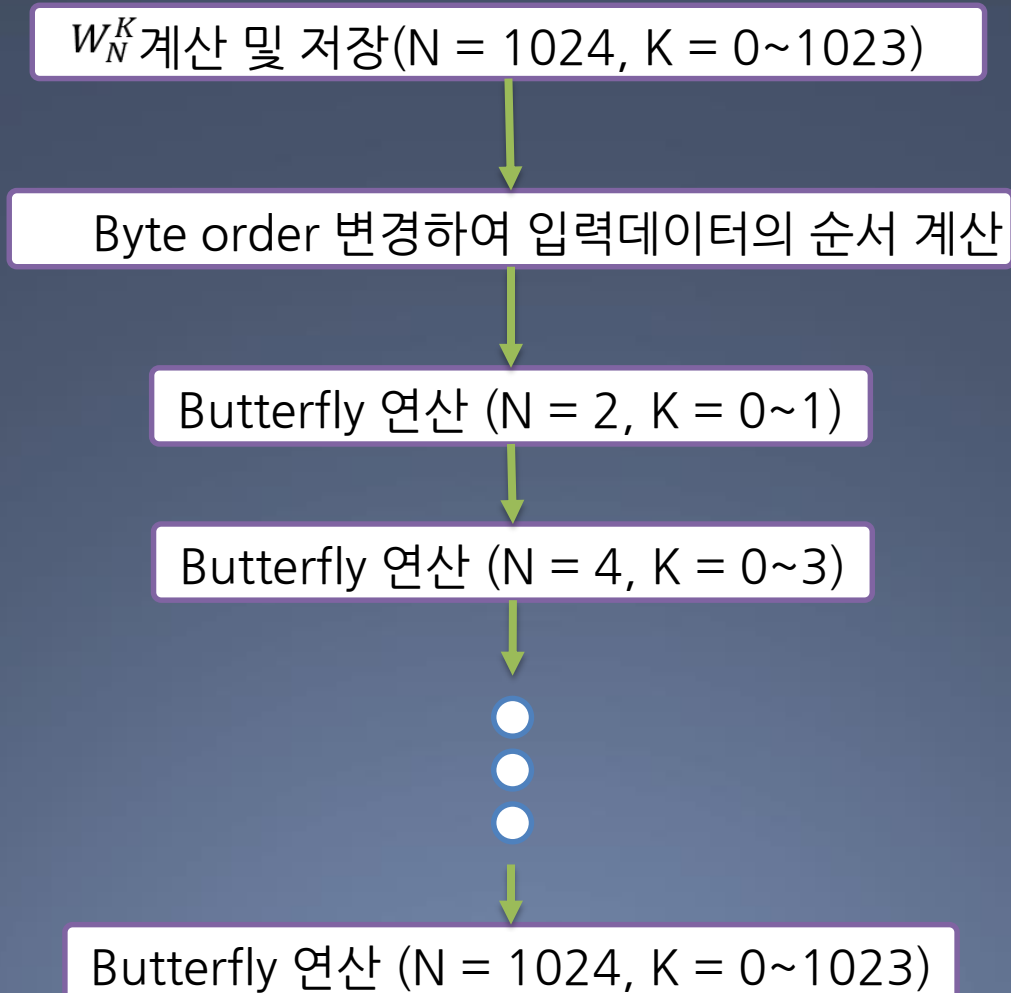
$$X_m = \sum_{n=0}^{N-1} x_n W_N^{mn}, \quad m = 0, 1, \dots, N-1$$



1) 음성 신호 처리

(2) 소프트웨어

③ Fast Fourier Transform (FFT) 흐름도



(3) 신호 처리 과정



(3) 신호 처리 과정

adc : 129	143
adc : 129	171
adc : 129	159
adc : 129	178
adc : 129	155
adc : 129	153
adc : 129	145
adc : 129	206
adc : 129	181
adc : 129	157
adc : 129	143
adc : 129	158
adc : 129	218
adc : 129	169
adc : 129	189
adc : 129	165
adc : 129	165
adc : 129	163
adc : 129	175
adc : 129	159
adc : 129	163
adc : 129	167
adc : 129	92
adc : 129	175
adc : 129	174
...	...

(3) 신호 처리 과정

① 샘플링 데이터

- Sampling Rate : 100kHz

32662

32725

32222

31747

31374

31071

30855

30765

30842

31079

31413

31773

32168

32535

32857

33085

33223

33218

33077

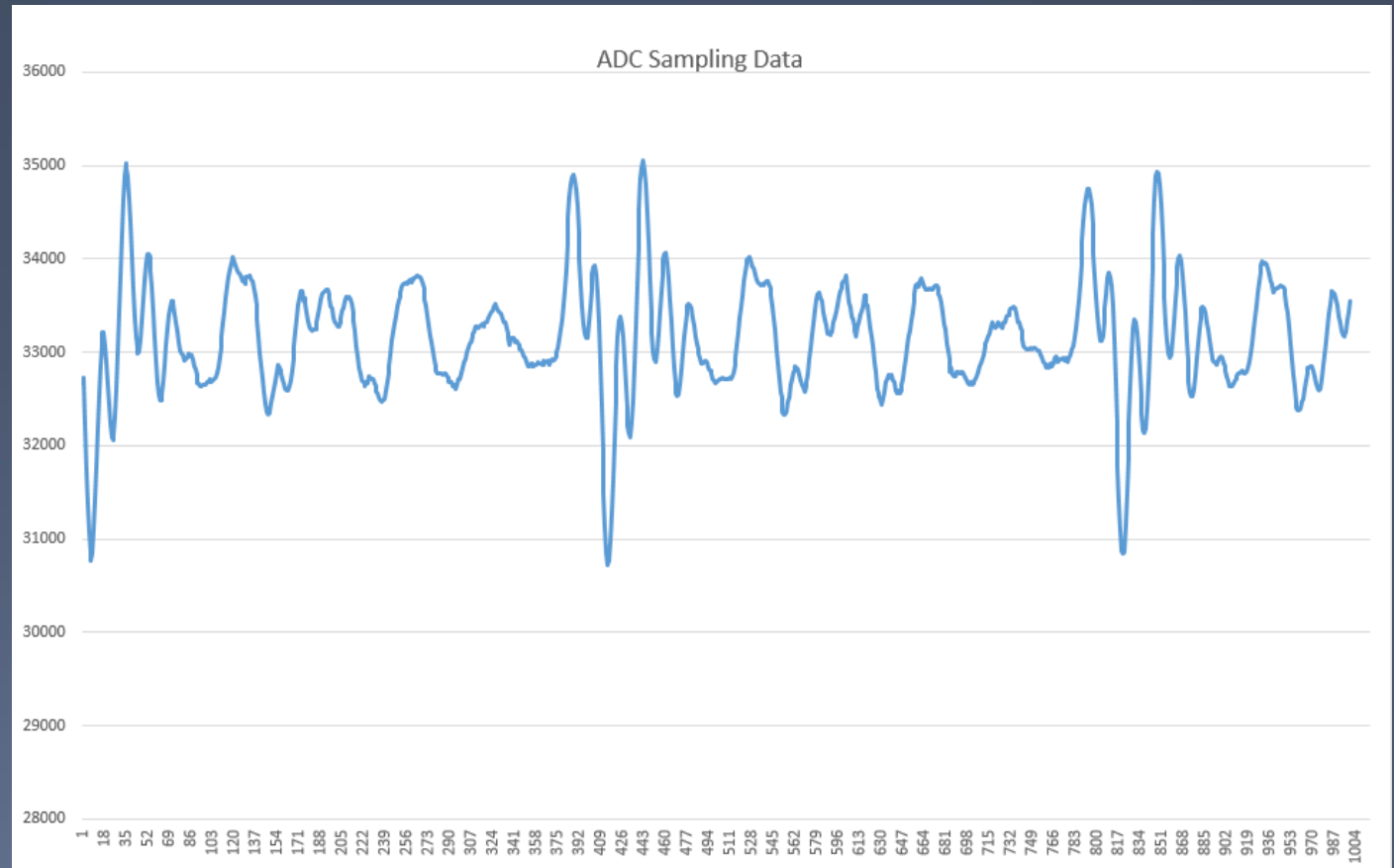
32893

32662

32437

32222

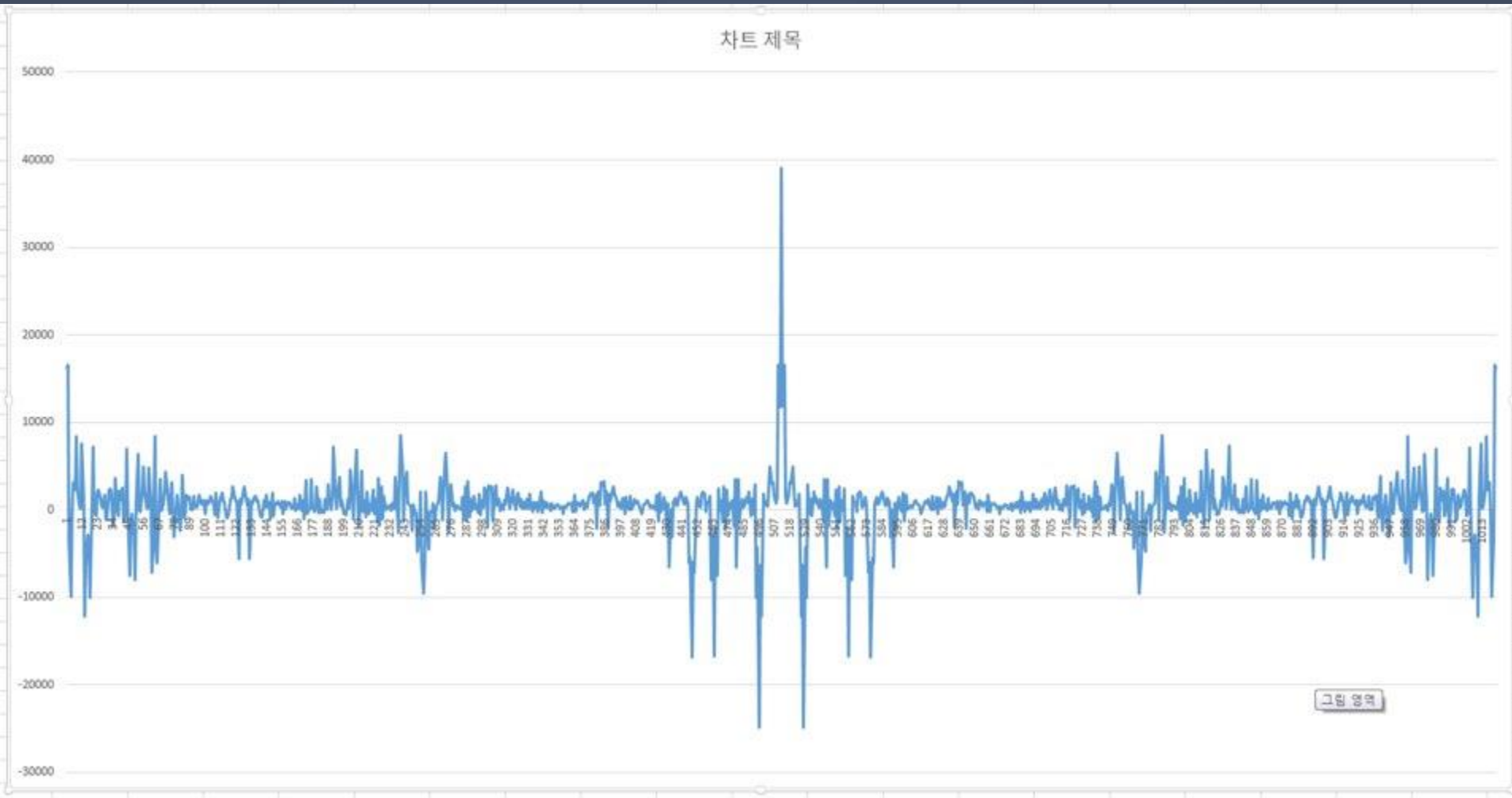
32089



(3) 신호 처리 과정

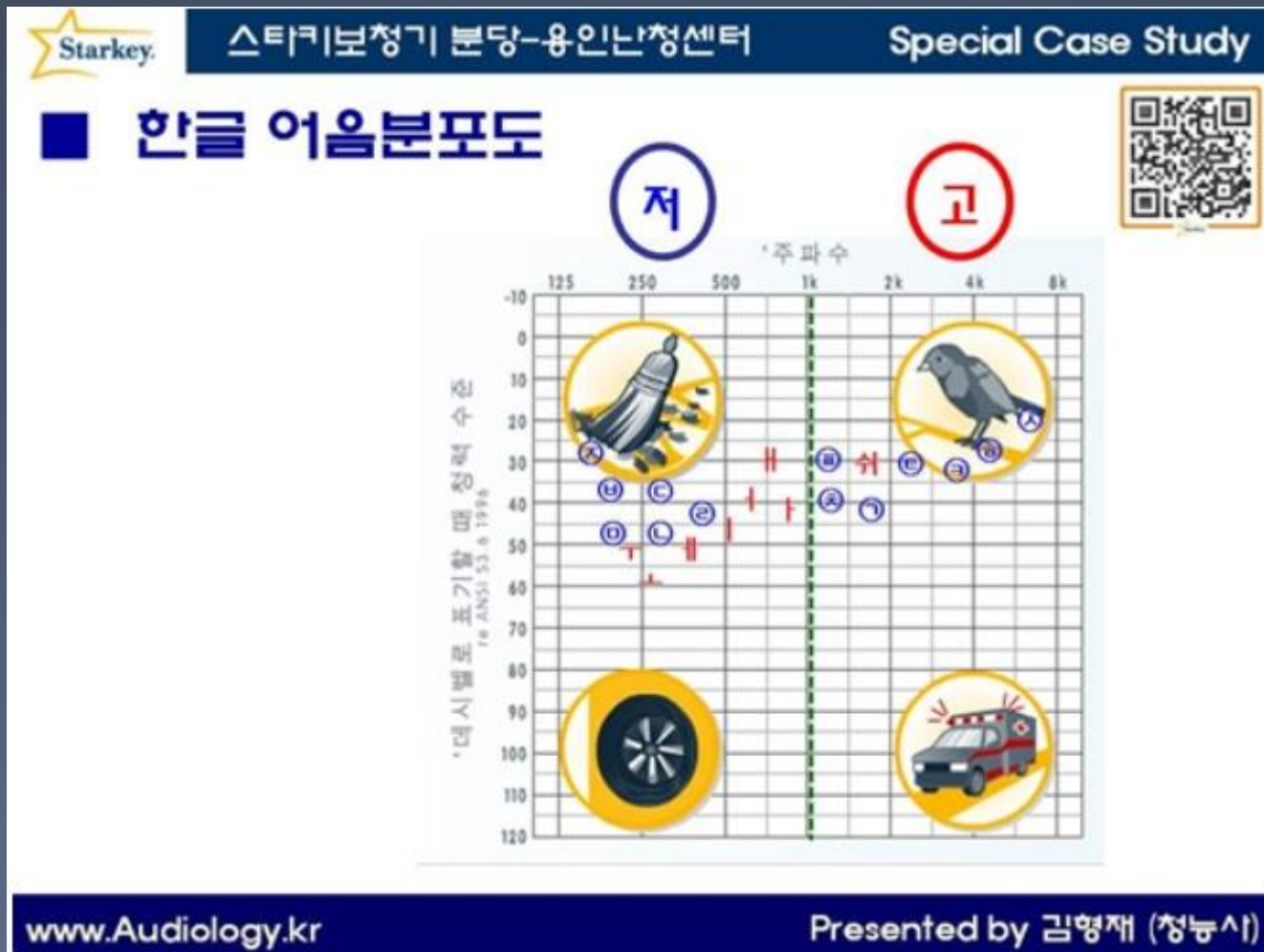
② Fast Fourier Transform (FFT)

- N : 1024



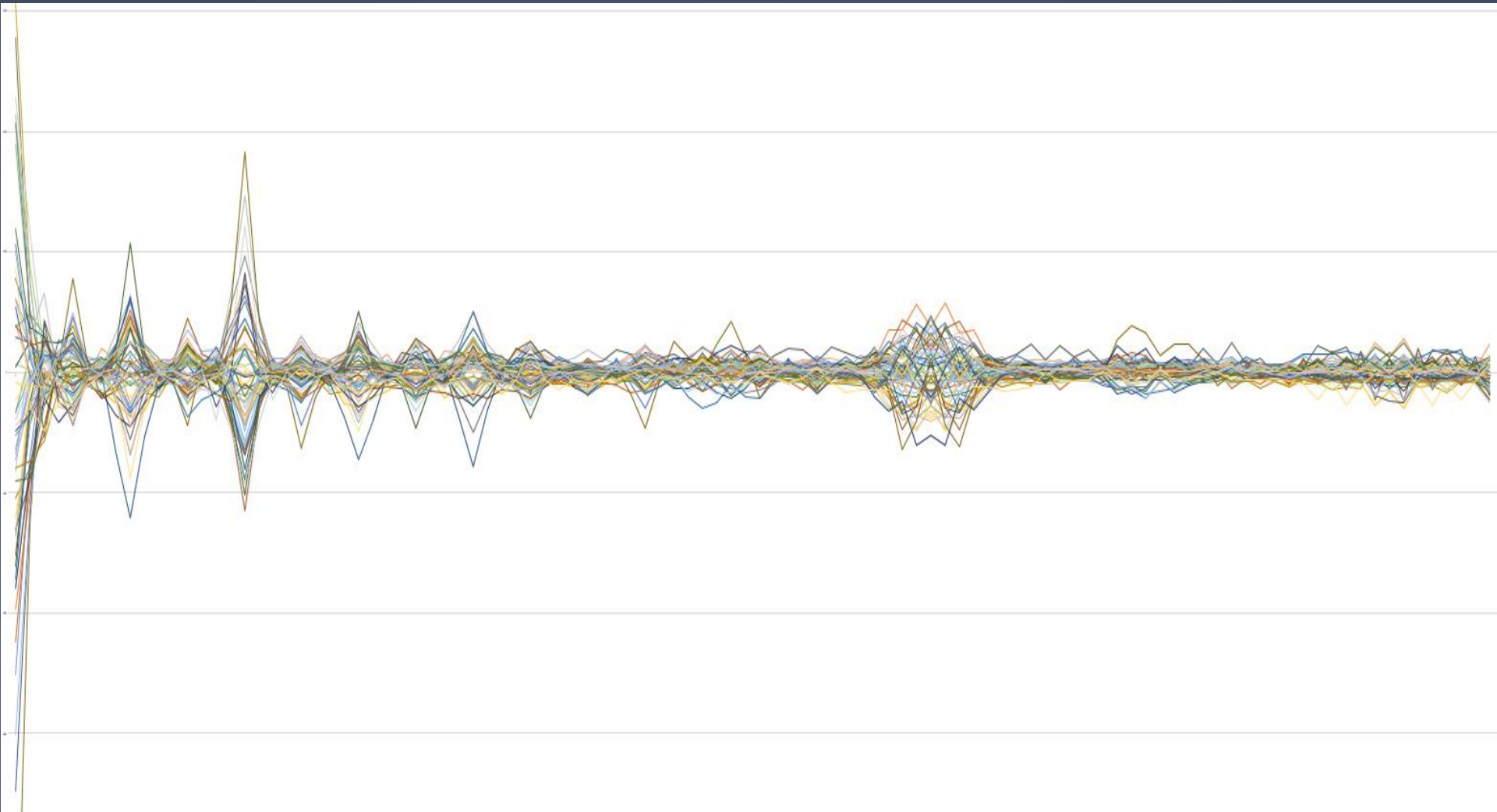
(3) 신호 처리 과정

③ 음성의 주파수 분포



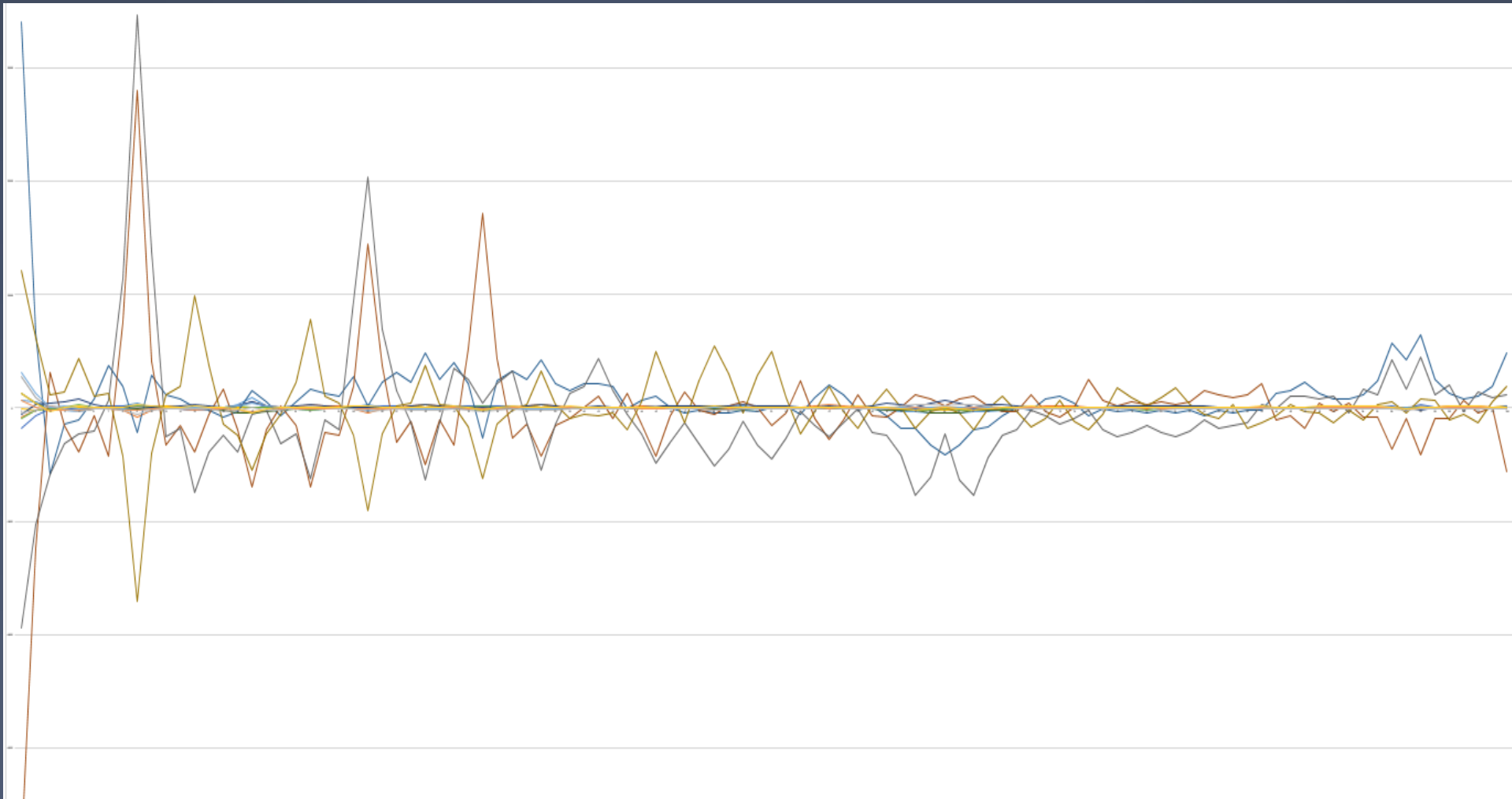
(3) 신호 처리 과정

④ 음성데이터 샘플링



(3) 신호 처리 과정

④ 음성데이터 샘플링



(3) 개발 중 문제점과 해결과정

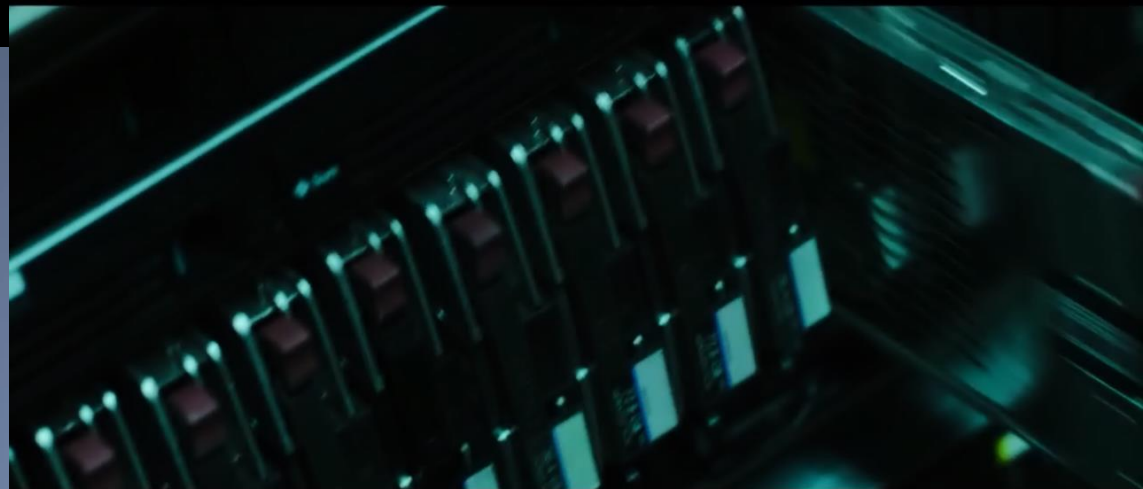
- ① SPI 모듈의 CS(chip select)pin이 보드의 다른 칩과 겹침
→ 다른 GPIO를 Chip Select Pin으로 사용하여 해결
- ② MIC-ADC BD의 가변저항을 0으로 돌려 보드의 오작동 발생
- ③ ADC 칩의 Alalog GND와 Digital GND의 전위차 때문에 과전류가 발생
→ Alalog와 Digital의 전원 소스를 분리하여 해결
- ④ IPC 통신 중 데이터의 손실이 발생
→ MessageQ를 이용하여 전송이 공유메모리에 매번 메모리 할당을 하지 않아 문제발생,
소스를 수정하여 해결

SECURITY CCTV

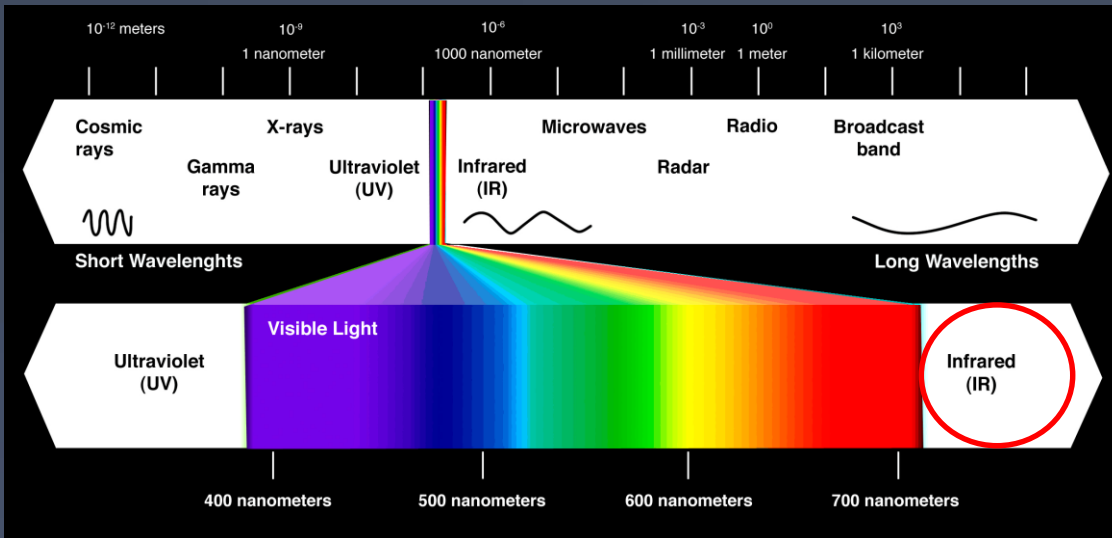
(1) Motivation



(1) Motivation



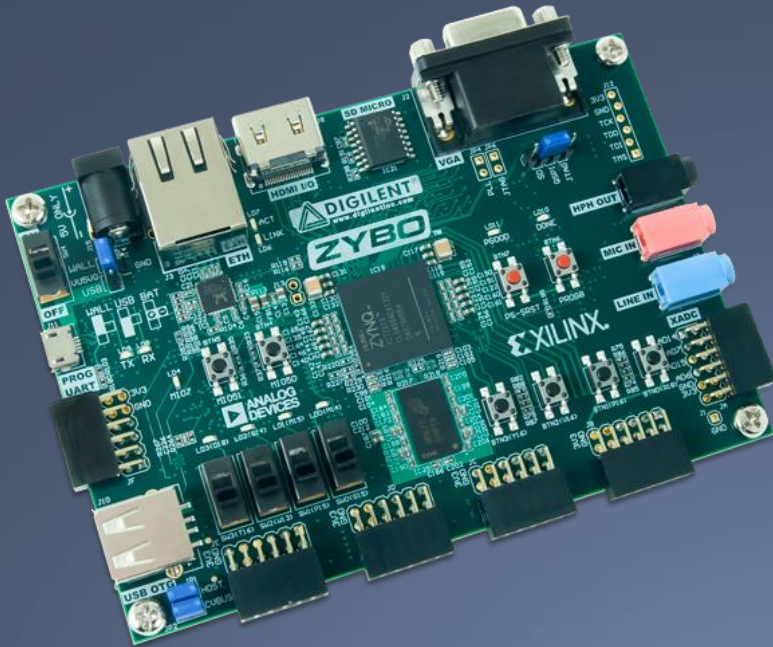
(1) Motivation



(2) 하드웨어

① Xilinx FPGA Board

- 명칭 : Zynq Board (ZYBO)
- 코어 : 650MHz Dual ARM Cortex-A9
- RAM : DDR3 512 MB

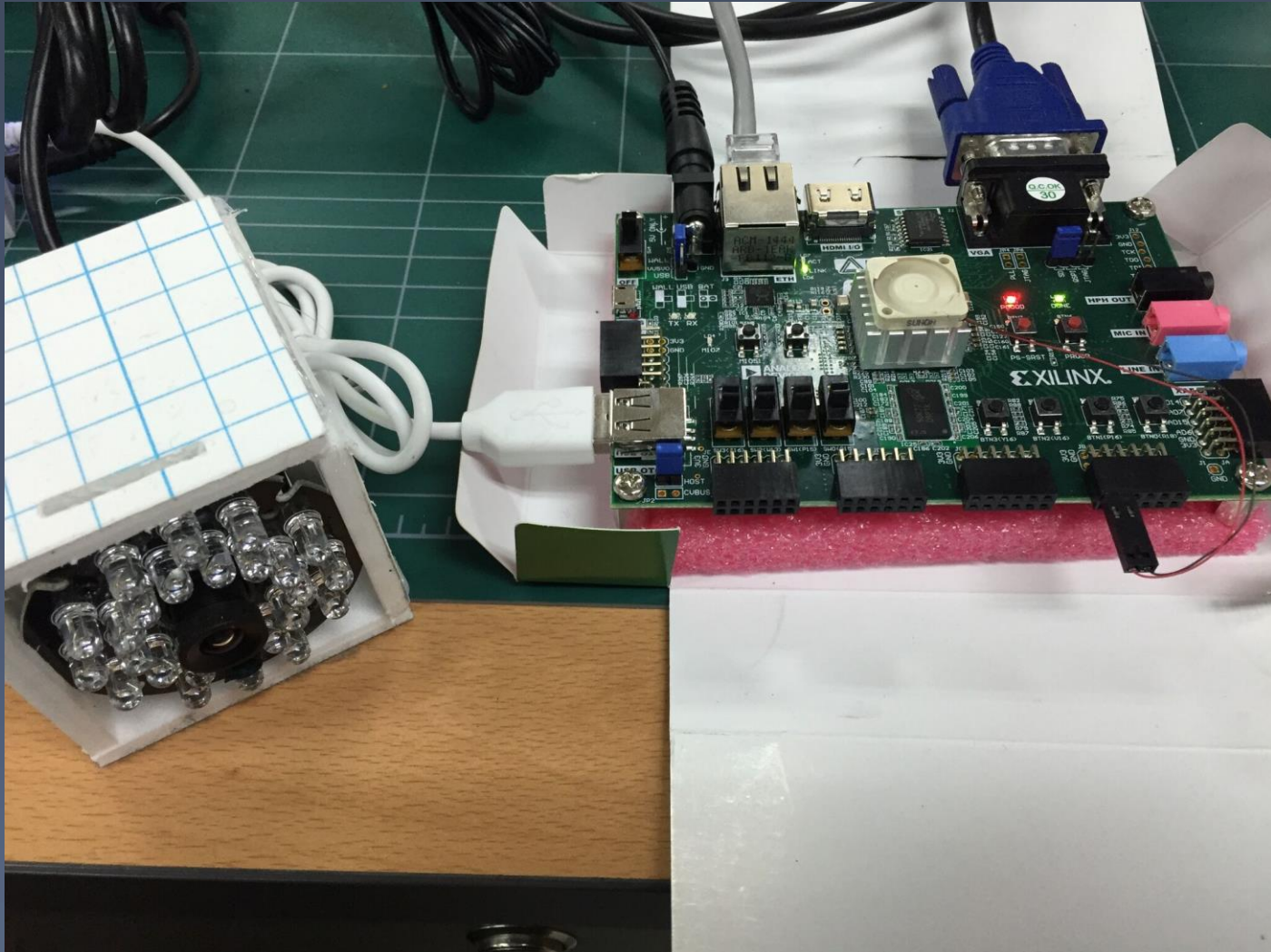


② Visual/IR USB Camera

- Sensor: CMOS
- Frame Rate: 30fps
- Minimum Illumination(Lux): 0.01
- FOV : 105°
- IR leds : 850nm



(2) 하드웨어

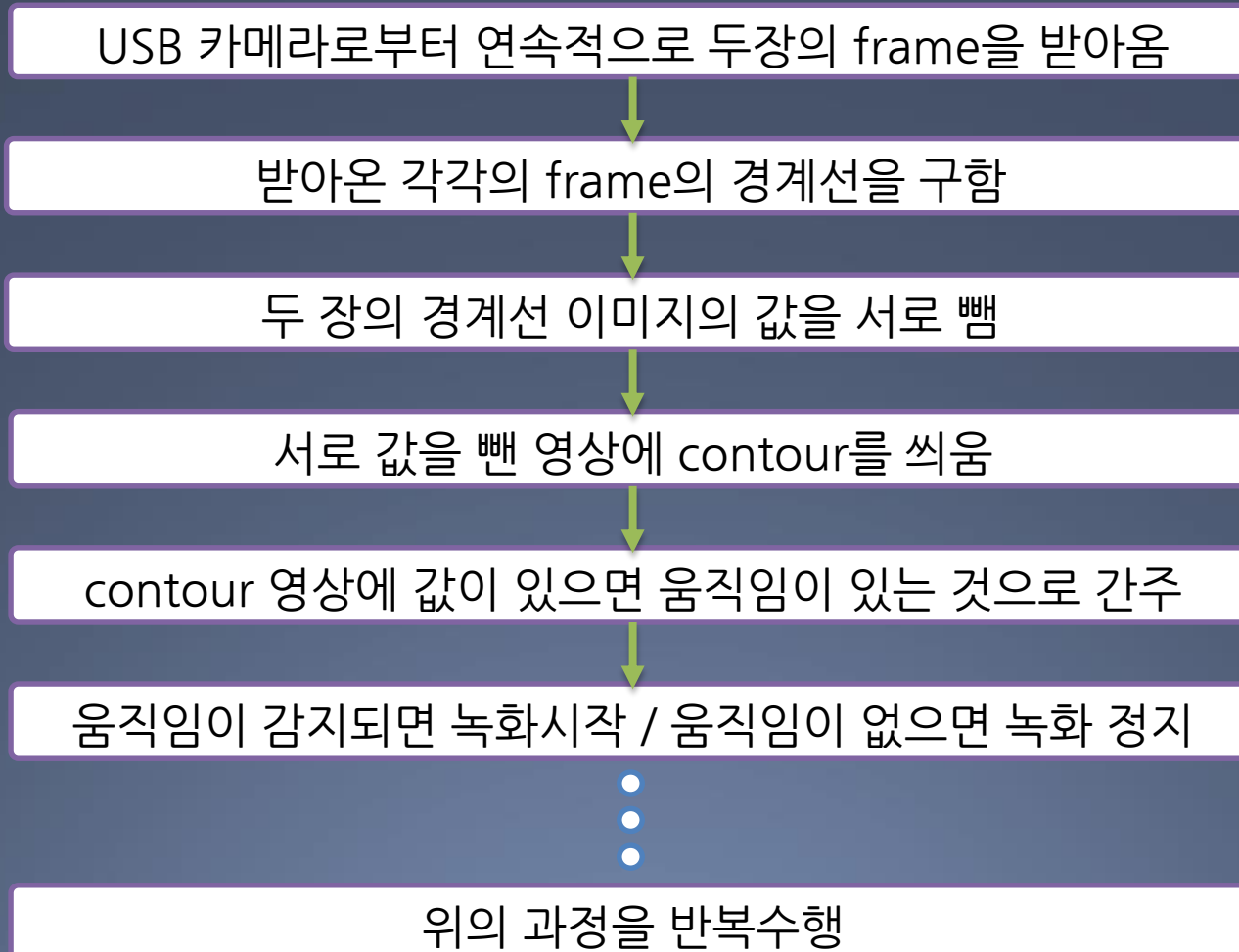


(3) 소프트웨어



(4) 처리 절차

① motion detection 흐름도



(4) 처리 절차

② motion detection 결과



카메라로 부터 영상을 받아옴



경계선 값을 구함



녹화 중 표시

REC

MOTION DETECTED

2016-04-15 15:49:54

모션 인식했을 때 녹화화면

현재 시간 표시

모션 인식 표시

(1) 하드웨어

① Cortex-M4 Board

- 명칭 : STM32F407IGT
- 코어 : Cortex-M4F



(1) 서버Test

192.168.0.191	192.168.0.210	ECHO	77 Request
192.168.0.210	192.168.0.191	ECHO	77 Response
fe80::3c18:660a:f14...	ff02::1:2	DHCPv6	152 Solicit X
fe80::a915:f39c:2ce...	ff02::1:2	DHCPv6	152 Solicit X
192.168.0.199	255.255.255.255	UDP	136 52052 → 1

192.168.0.191	192.168.0.210	ECHO	77 Request
192.168.0.210	192.168.0.191	ECHO	77 Response
fe80::3c18:660a:f14...	ff02::1:2	DHCPv6	152 Solicit X
fe80::a915:f39c:2ce...	ff02::1:2	DHCPv6	152 Solicit X
192.168.0.199	255.255.255.255	UDP	136 52052 → 1

n wire (616 bits), 77 bytes captured (616 bits) on interface 0
 SamsungE_31:12:8b (e8:11:32:31:12:8b), Dst: 02:00:00:00:00:00
 version 4, Src: 192.168.0.191, Dst: 192.168.0.210
 ol, Src Port: 8 (8), Dst Port: 7 (7)

5465666768696a6b6c6d6e6f707172737475767879...

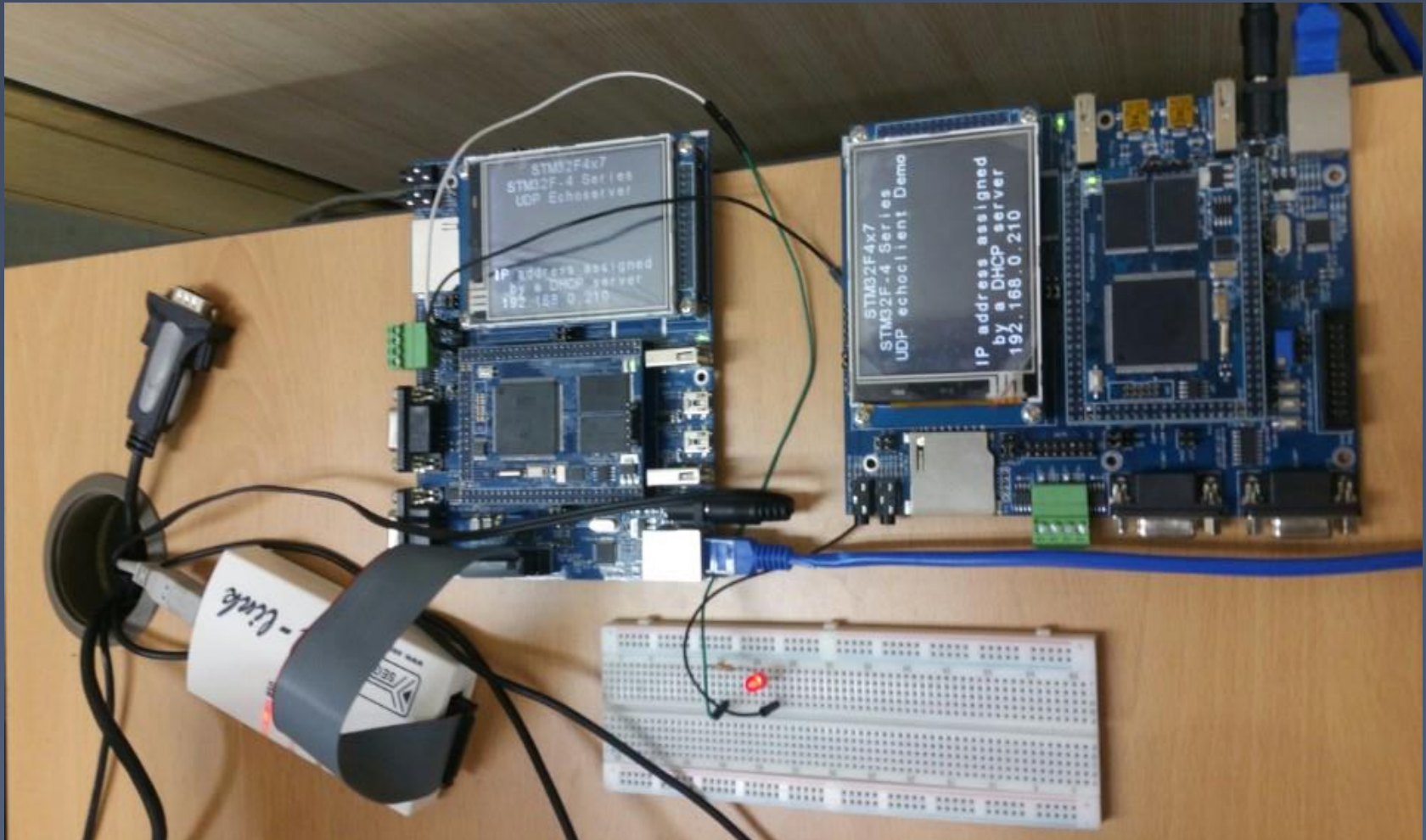
n wire (616 bits), 77 bytes captured (616 bits) on interface 0
 :00:00:00:00:00:00 (02:00:00:00:00:00:00), Dst: SamsungE_31:12:8b (e8:11:32:31:12:8b)
 version 4, Src: 192.168.0.210, Dst: 192.168.0.191
 ol, Src Port: 7 (7), Dst Port: 8 (8)

5465666768696a6b6c6d6e6f707172737475767879...

00 e8 11 32 31 12 8b 08 00 45 00 21....E.
00 80 11 00 00 c0 a8 00 bf c0 a8	.?1.....
07 00 2b 83 1e 61 62 63 64 65 66+ ..abcdef
5c 6d 6e 6f 70 71 72 73 74 75 76	ghijklmn opqrstuv
83 34 35 36 37 38 39 30	xyz12345 67890

8b 02 00 00 00 00 00 08 00 45 00	..21.... ..E.
00 ff 11 38 c2 c0 a8 00 d2 c0 a8	.?..... 8.....
08 00 2b e7 7a 61 62 63 64 65 66+ .zabcdef
5c 6d 6e 6f 70 71 72 73 74 75 76	ghijklmn opqrstuv
83 34 35 36 37 38 39 30	xyz12345 67890

(1) 하드웨어



(2) M4보드간 통신

168.0.191	192.168.0.208	UDP	43 7001 → 7004 Len=1	168.0.191	192.168.0.208	UDP	43 7001 → 7004 Len=1
168.0.208	192.168.0.191	UDP	60 7004 → 7001 Len=1	168.0.208	192.168.0.191	UDP	60 7004 → 7001 Len=1
168.0.199	255.255.255.255	UDP	136 64004 → 10505 Len=	168.0.199	255.255.255.255	UDP	136 64004 → 10505 Len=
0::f87d:4bed:9ff... ff02::1:2		DHCPv6	151 Solicit XID: 0xf0cc	0::f87d:4bed:9ff... ff02::1:2		DHCPv6	151 Solicit XID: 0xf0cc
168.0.199	255.255.255.255	UDP	136 64007 → 10505 Len=	168.0.199	255.255.255.255	UDP	136 64007 → 10505 Len=
0::f87d:4bed:9ff... ff02::1:2		DHCPv6	151 Solicit XID: 0xf0cc	0::f87d:4bed:9ff... ff02::1:2		DHCPv6	151 Solicit XID: 0xf0cc
168.0.185	224.0.0.251	MDNS	83 Standard query 0x00	168.0.185	224.0.0.251	MDNS	83 Standard query 0x00
168.0.199	255.255.255.255	UDP	136 64010 → 10505 Len=	168.0.199	255.255.255.255	UDP	136 64010 → 10505 Len=
0::f87d:4bed:9ff... ff02::1:2		DHCPv6	151 Solicit XID: 0xf0cc	0::f87d:4bed:9ff... ff02::1:2		DHCPv6	151 Solicit XID: 0xf0cc
168.0.199	255.255.255.255	UDP	136 64013 → 10505 Len=	168.0.199	255.255.255.255	UDP	136 64013 → 10505 Len=

e (344 bits), 43 bytes captured (344 bits) on interface 0			
ce\NPF_{3F22FE2C-2DE0-4B22-957E-D1747E1C8ACE})			
chernet (1)			
2016 00:06:51.184659000 00000000 g000			
packet: 0.000000000 seconds]			
.184659000 seconds			
ous captured frame: 0.000024000 seconds]			
ous displayed frame: 0.256896000 seconds]			
or first frame: 3.342721000 seconds]			

11	32	31	12	8b	08	00	45	00	21....E.
0	11	00	00	c0	a8	00	bf	c0	a8	..v.....
0	09	82	fa	33					...Y.\.	..3

00	00	00	00	00	08	00	45	00	..21....E.
11	38	eb	c0	a8	00	d0	c0	a8	8.....
09	13	47	33	00	00	00	00	00	...\.Y..	.G3....

4) 모터 제어



USER_setParams()가 동작한 이후 아래와 같은 결과

```

pUserParams->iqFullScaleCurrent_A = 10.0
pUserParams->iqFullScaleVoltage_V = 450.0
pUserParams->numIsrTicksPerCntrlHz = 800.0
pUserParams->numIsrTicksPerCurrentTick = 1
pUserParams->numCtrlTicksPerSpeedTick = 1
pUserParams->numCtrlTicksPerEstTick = 15
pUserParams->numCtrlTicksPerTrajTick = 15

systemFreq_MHz = 90.0
pwmPeriod_usec = 66.66666666
numVoltageSensors = 3
offsetPole_rps = 20.0
current_sf = 1.989
fluxPole_rps = 100.0
zeroSpeedLimit = 0.000625
forceAngleFreq_Hz = 1.0
maxAccel_Hzps = 20.0
maxAccel_est_Hzps = 2.0
directionPole_rps = 6.0
speedPole_rps = 100.0
dcBusPole_rps = 100.0
fluxFraction = 1.0
indEst_speedMaxFraction = 1.0

```

ACI Motor를 TMS320F28069로 제어 하기 위해서는 ADC_Handle, PLL_Handle, GPIO_Handle, PWM_Handle 등에 대한 레지스터가 등록되어야 합니다. 제어공학을 기반으로 DSP의 버전 및 매뉴얼 데이터 시트를 확인하고 코드를 잘 파악하여 인버터 설정, ADC Mechanism, PI제어기를 배워 익혔습니다.

```

155 TIMER_Handle timerHandle[3]; //!< the timer handles
156
157 WDOG_Handle wdogHandle; //!< the watchdog handle
158
159 HAL_AdcData_t adcBias; //!< the ADC bias
160
161 _iq current_sf; //!< the current scale factor, amps_pu/cnt
162
163 _iq voltage_sf; //!< the voltage scale factor, volts_pu/cnt
164
165 uint_least8_t numCurrentSensors; //!< the number of current sensors
166 uint_least8_t numVoltageSensors; //!< the number of voltage sensors
167
168 #ifdef QEP
169 QEP_Handle qepHandle[1]; //!< the QEP handle
170 #endif
171
172 HAL_Obj;

```

이번에는 FLASH_Handle를 살펴보도록 하자!

```

245 ///! \brief Defines the flash (FLASH) handle
246 ///!
247 typedef struct _FLASH_Obj_ *FLASH_Handle;

```

_FLASH_Obj_는 아래와 같이 구현되어 있다.

```

230 ///! \brief Defines the flash (FLASH) object
231 ///!
232 typedef struct _FLASH_Obj_
233 {
234     volatile uint16_t FOPT; //!< Flash Option Register
235     volatile uint16_t rsvd_1; //!< Reserved
236     volatile uint16_t FPWR; //!< Flash Power Modes Register
237     volatile uint16_t FSTATUS; //!< Status Register
238     volatile uint16_t FSTDYWAIT; //!< Flash Sleep To Standby Wait Register
239     volatile uint16_t FACTIVEWAIT; //!< Flash Standby To Active Wait Register
240     volatile uint16_t FBANKWAIT; //!< Flash Read Access Wait State Register
241     volatile uint16_t FOTPAWAIT; //!< OTP Read Access Wait State Register
242 } FLASH_Obj;

```

```

301 ///! \brief Enumeration to define the pulse width modulation (PWM) sync modes
302 ///!
303 typedef enum
304 {
305     PWM_SyncMode_EPWMxSYNC = (0 << 4),
306     PWM_SyncMode_CounterEqualZero = (1 << 4),
307     PWM_SyncMode_CounterEqualCounterCompareZero = (2 << 4),
308     PWM_SyncMode_Disable = (3 << 4)
309 } PWM_SyncMode_e;

```

```

1435 void PWM_setSyncMode(PWM_Handle pwmHandle, const PWM_SyncMode_e syncMode)
1436 {

```

```

1437     PWM_Obj *pwm = (PWM_Obj *)pwmHandle;
1438
1439     // clear the bits
1440     pwm->TBCTL &= (~PWM_TBCTL_SYNCSEL_BITS);
1441
1442     // set the bits
1443     pwm->TBCTL |= syncMode;
1444
1445     return;
1446 }

```

이번에는 PWM_Handle를 살펴보도록 하자!

```

933 ///! \brief Defines the pulse width modulation (PWM) handle
934 ///!
935 typedef struct _PWM_Obj_ *PWM_Handle;

```

_PWM_Obj_는 아래와 같이 구현되어 있다.

```

871 ///! \brief Defines the pulse width modulation (PWM) object
872 ///!
873 typedef struct _PWM_Obj_
874 {
875     volatile uint16_t TBCTL; //!< Time-Base Control Register
876     volatile uint16_t TBSTS; //!< Time-Base Status Register
877     volatile uint16_t TBPHSHR; //!< Extension for the HRPWM Phase Register
878     volatile uint16_t TBPHS; //!< Time-Base Phase Register
879     volatile uint16_t TBCTR; //!< Time-Base Counter
880     volatile uint16_t TBPRD; //!< Time-Base Period register set
881     volatile uint16_t TBPRDHR; //!< Time-Base Period High Resolution Register
882     volatile uint16_t CMPCTL; //!< Counter-Compare Control Register
883     volatile uint16_t CMPAHR; //!< Extension of HRPWM Counter-Compare A Register
884     volatile uint16_t CMPA; //!< Counter-Compare A Register
885     volatile uint16_t CMPB; //!< Counter-Compare B Register
886     volatile uint16_t AQCTLA; //!< Action-Qualifier Control Register for Output A (EPWMxA)
887     volatile uint16_t AQCTLB; //!< Action-Qualifier Control Register for Output B (EPWMxB)
888     volatile uint16_t AQSPRC; //!< Action qual SW force
889     volatile uint16_t AQCSFRC; //!< Action qualifier continuous SW force
890     volatile uint16_t DBCTL; //!< Dead-band control
891     volatile uint16_t DBRED; //!< Dead-band rising edge delay
892     volatile uint16_t DBFED; //!< Dead-band falling edge delay
893     volatile uint16_t TZSEL; //!< Trip zone select
894     volatile uint16_t TZDCSEL; //!< Trip zone digital comparator select
895     volatile uint16_t TZCTL; //!< Trip zone control
896     volatile uint16_t TZEINT; //!< Trip zone interrupt enable
897     volatile uint16_t TZFLG; //!< Trip zone interrupt flags
898     volatile uint16_t TZCLR; //!< Trip zone clear
899     volatile uint16_t TZFR; //!< Trip zone force interrupt
900     volatile uint16_t ETSEL; //!< Event trigger selection
901     volatile uint16_t ETPS; //!< Event trigger pre-scaler
902     volatile uint16_t ETFLG; //!< Event trigger flags
903     volatile uint16_t ETCLR; //!< Event trigger clear
904     volatile uint16_t ETFRC; //!< Event trigger force
905     volatile uint16_t PCCTL; //!< PWM chopper control
906     volatile uint16_t rsvd_1; //!< Reserved
907     volatile uint16_t HRCNFG; //!< HRPWM Config Reg
908     volatile uint16_t HRPWR; //!< HRPWM Power Register
909     volatile uint16_t rsvd_2[4]; //!< Reserved
910     volatile uint16_t HRMSTEP; //!< HRPWM MEP Step Register
911     volatile uint16_t rsvd_3; //!< Reserved
912     volatile uint16_t HRPCTL; //!< High Resolution Period Control
913     volatile uint16_t rsvd_4; //!< Reserved
914     volatile uint16_t TBPRDHRM; //!< Time-Base Period High Resolution mirror Register
915     volatile uint16_t TBPRDM; //!< Time-Base Period mirror register
916     volatile uint16_t CMPAHRM; //!< Extension of HRPWM Counter-Compare A mirror Register
917     volatile uint16_t CMPAM; //!< Counter-Compare A mirror Register
918     volatile uint16_t rsvd_5[2]; //!< Reserved
919     volatile uint16_t DCTRIPSEL; //!< Digital Compare Trip Select
920     volatile uint16_t DACCTL; //!< Digital Compare A Control
921     volatile uint16_t DCBCTL; //!< Digital Compare B Control
922     volatile uint16_t DCFCTL; //!< Digital Compare Filter Control
923     volatile uint16_t DCCAPCTL; //!< Digital Compare Capture Control
924     volatile uint16_t DCOFFSET; //!< Digital Compare Filter Offset
925     volatile uint16_t DCFWINDCNT; //!< Digital Compare Filter Window Counter
926     volatile uint16_t DCFWINDOWN; //!< Digital Compare Filter Window
927     volatile uint16_t DCFWINDOWNCNT; //!< Digital Compare Filter Window Counter
928     volatile uint16_t DCCAP; //!< Digital Compare Filter Counter Capture
929
930 } PWM_Obj;

```

54	SYNCOSEL	Synchronization Output Select. These bits select the source of the EPWMxSYNCO signal.
00		EPWMxSYNCO
01		CTR = zero: Time-base counter equal to zero (TBCTR = 0x0000)
10		CTR = CMPB: Time-base counter equal to counter-compare B (TBCTR = CMPB)
11		Disable EPWMxSYNCO signal

EPWMxSYNCO Signal을 Synchronization Output으로 선택한다.

InstaSPINTM- MOTION

Motor Type ☒ ACIM ☐ PMSM/BLDC

Identification Method

Identification Settings

Res Est Current(A)

Ind Est Current (A)

Estimation Freq (Hz)

Motor Pole Pairs

Motor Max Current (A)

Motor Max Freq (Hz)

Rated Flux (Vline/Hz)
ACIM Only

Identified Motor Parameters

R_s (Ω)

L_{s_d} (H)
Identifies average L_s and uses for both d and q. If accuracy is known set in user.h

L_{s_q} (H)

Flux (V/Hz)

Magn Current (A)
ACIM Only

R_r (Ω)
ACIM Only

Control Timing

PWM Frequency KHz

ADC Sampling KHz

Current Control KHz

FASTTM KHz

Identify Motor

Status

Motor Not Identified

Identifying Motor

Lock Rotor then press Proceed ACIM Only

Proceed

Motor Identified

Reset System



Version: 2.1.1

Motion Profiler

Choose a curve type and acceleration/jerk limits

Set a target speed and hit Enter

Motion profiles will automatically be generated and sent to the controller

Disable SpinTAC

Reverts to Speed PI and Ramp Reference on the "Speed or Torque" tab

Profile Types

- ☐ Trapezoidal Curve
- ☐ sCurve
- ☒ stCurve (Smoothest) *LineStream Proprietary

Profile Limits

Acceleration Limit: ≤ ≤ [RPM/s]

Jerk Limit: ≤ ≤ [KRPM/s²]

Start Speed [RPM]

Motor Speed [RPM]

Target Speed [RPM]

Profile Time: [Sample Counts]



SpinTAC Profiler + SpinTAC Speed Controller

SpinTAC Profiler + PI Speed Controller

CCS Debug - proj_lab06b/proj_lab06b.c - Code Composer Studio

File Edit View Project Tools Run Scripts Window Help

Debug

proj_lab06b [Code Composer Studio - Device Debugging]

Texas Instruments XDS100V2 USB Emulator_0/C28x (Running)

Texas Instruments XDS100V2 USB Emulator_0/CLA_0 (Disconnected : Unknown)

Variables

Expression	Type	Value	Address
gMotorVars.Flux_VpHz	float	0.2923077	0x00C0B8@P
gMotorVars.Speed_krpm	long	0.09819632769 (Q-Value(24))	0x00C09E@P
gMotorVars.Torque_Nm	long	0.07114565372 (Q-Value(24))	0x00C0AA@P
gMotorVars.SpinTAC.VelldRun	unsigned int	0 (Decimal)	0x00C0DE@P
gMotorVars.SpinTAC.VelldStatus	enum unknown	ST_VEL_ID_IDLE	0x00C0DF@P
gMotorVars.SpinTAC.VelldGoalSpeed_krpm	long	0.0 (Q-Value(24))	0x00C0E0@P
gMotorVars.SpinTAC.VelldTorqueRampTm	long	0.0 (Q-Value(24))	0x00C0E2@P
gMotorVars.SpinTAC.InertiaEstimate_AperK	long	0.01999998093 (Q-Value(24))	0x00C0E4@P
gMotorVars.SpinTAC.FrictionEstimate_AperK	long	0.009999930859 (Q-Value(24))	0x00C0E6@P
gMotorVars.SpinTAC.VelldErrorID	unsigned short	0	0x00C0E8@P
gMotorVars.Kp_Idq	long	0.7053638697 (Q-Value(24))	0x00C0BE@P
gMotorVars.Ki_Idq	long	0.01814192533 (Q-Value(24))	0x00C0C0@P
gMotorVars.VdcBus_KV	long	0.299499929 (Q-Value(24))	0x00C0CA@P

ctrl_states.h est_flux_states.h pll.h spintac_vel_id.h ctrl.c proj_lab05f.c proj_lab06b.c user.c spintac_velocity.h proj_lab06b.out TI Resource Explorer

```

351 // Return the bias value for currents
352 gMotorVars_I_bias.value[0] = HAL_getBias(halHandle,HAL_SensorType_Current,0);
353 gMotorVars_I_bias.value[1] = HAL_getBias(halHandle,HAL_SensorType_Current,1);
354 gMotorVars_I_bias.value[2] = HAL_getBias(halHandle,HAL_SensorType_Current,2);
355
356 // Return the bias value for voltages
357 gMotorVars_V_bias.value[0] = HAL_getBias(halHandle,HAL_SensorType_Voltage,0);
358 gMotorVars_V_bias.value[1] = HAL_getBias(halHandle,HAL_SensorType_Voltage,1);
359 gMotorVars_V_bias.value[2] = HAL_getBias(halHandle,HAL_SensorType_Voltage,2);
  
```

DualTime8-0

Scripting Console

CDT Build Console [proj_lab06b]

```

**** Build of configuration Release for project proj_lab06b ****
"C:\ti\ccsv\utils\bin\make" -k all
make: Nothing to be done for 'all'.
**** Build Finished ****
  
```



SpinTAC is a
High-Performance Controller
that replaces PI control

It uses advanced control techniques and is
Tuned with one parameter
In a matter of minutes

You only have to tune it once and SpinTAC performs across a
Wide Operating Range

To begin, **check the limits and click "Estimate Motor Inertia"**. After SpinTAC is Enabled, move the tab "SpinTAC 2: Tuning".

1 Limits

Motor Ready

Ramp Time: [sec]

$0.1 \leq \text{Ramp Time} \leq 25.0$

Iq Limit: [A]

$0 < \text{Iq Limit} \leq 3$

Speed: [RPM]

$0 < \text{Speed} \leq 8625$

2 Run

Estimate Motor Inertia

Your motor will spin a few times as SpinTAC estimates the motor inertia

3 Status

Zero Speed

Identifying

Not Identified

Identified

4 Results

Inertia:

$10^{-3} \text{ [A}\cdot\text{s/RPM]}$

Friction:

$10^{-3} \text{ [A}\cdot\text{s/RPM]}$

SpinTAC Enabled

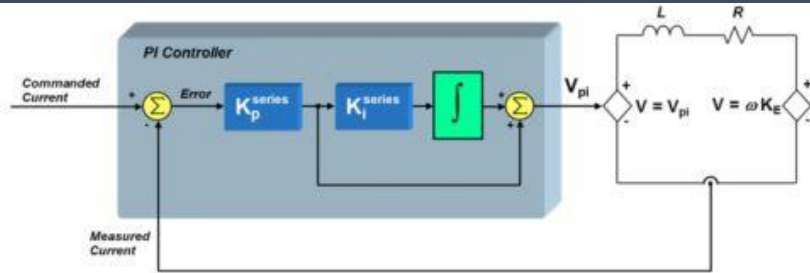


Figure 18: PI current controlled motor system including the stator.

$$Z_{eq} = R + j\omega L \quad i_s Z_{eq} = V \Leftrightarrow i_s = \frac{V}{Z_{eq}} = \frac{V}{R + j\omega L}$$

$$V_{pi} = L \frac{di_s}{dt} + i_s R \Leftrightarrow V_{pi}(s) = L[sI(s) - i(0)] + RI(s) = V(s) = LsI(s) + RI(s) = (Ls + R)I(s)$$

$$V(s) = (Ls + R)I(s), \quad \frac{I(s)}{V(s)} = I(s) \frac{1}{(Ls + R)I(s)} = \frac{1}{(Ls + R)}$$

$$\frac{I(s)}{V(s)} = \frac{1}{(Ls + R)} = \frac{1}{L} \frac{1}{s + \frac{R}{L}}$$

$$G_{loop} = \left(\frac{K_p^{series} \cdot K_i^{series} \left(1 + \frac{s}{K_i^{series}}\right)}{s} \right) \left(\frac{1}{L} \frac{1}{s + \frac{R}{L}} \right) = \frac{K_p^{series} \cdot K_i^{series}}{R} \frac{s + K_i^{series}}{s^2 + s} = \frac{K_p^{series}(s + K_i^{series})}{Ls^2 + Rs}$$

$$G(s) = \frac{G_{loop}}{1 + G_{loop}} = \frac{\frac{K_p^{series}(s + K_i^{series})}{Ls^2 + Rs}}{1 + \frac{K_p^{series}(s + K_i^{series})}{Ls^2 + Rs}} = \frac{K_p^{series}(s + K_i^{series})}{K_p^{series}(s + K_i^{series}) + (Ls^2 + Rs)}$$

$$\frac{1 + \frac{s}{K_i^{series}}}{\left(\frac{L}{K_p^{series} \cdot K_i^{series}} \right) s^2 + \left(\frac{R}{K_p^{series} \cdot K_i^{series}} + \frac{1}{K_i^{series}} \right) s + 1}$$

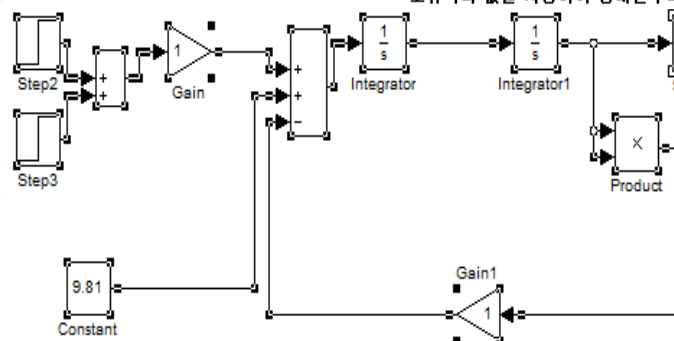
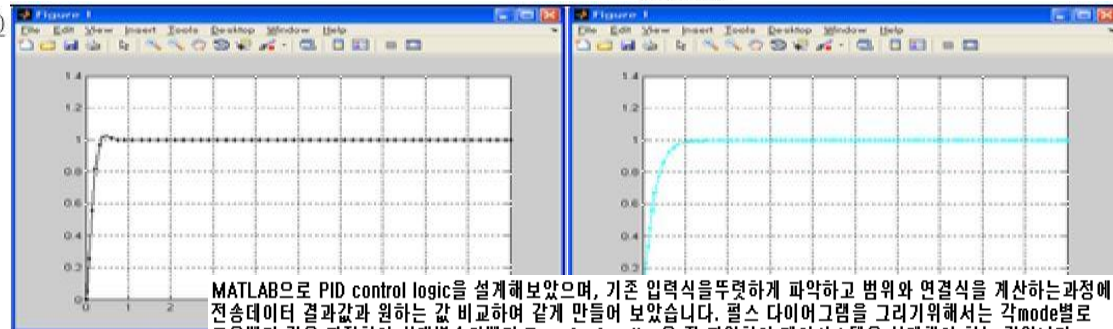
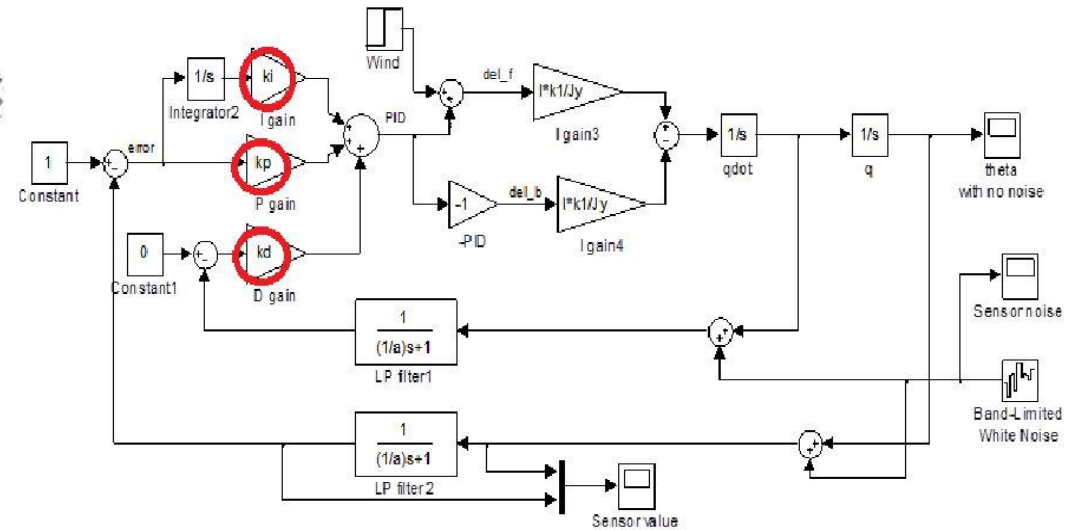
$$G(s) = \frac{G_{loop}}{1 + G_{loop}} = \frac{\frac{K_p^{series}(s + K_i^{series})}{Ls^2 + Rs}}{\frac{K_p^{series}(s + K_i^{series}) + (Ls^2 + Rs)}{Ls^2 + Rs}} = \frac{K_p^{series}(s + K_i^{series})}{K_p^{series}(s + K_i^{series}) + (Ls^2 + Rs)}$$

$$= \frac{K_p^{series} \cdot s + K_p^{series} \cdot K_i^{series}}{K_p^{series} \cdot s + K_p^{series} \cdot K_i^{series} + Ls^2 + Rs} = \frac{1 + \frac{s}{K_i^{series}}}{\frac{Ls^2}{K_p^{series} \cdot K_i^{series}} + 1 + \frac{Rs}{K_p^{series} \cdot K_i^{series}} + \frac{1}{K_i^{series}}}$$

$$\left(\frac{L}{K_p^{series} \cdot K_i^{series}} \right) s^2 + \left(\frac{R}{K_p^{series} \cdot K_i^{series}} + \frac{1}{K_i^{series}} \right) s + 1 = (1 + Cs)(1 + Ds)$$

$$C \cdot D = \frac{L}{K_p^{series} \cdot K_i^{series}}, \quad C + D = \frac{R}{K_p^{series} \cdot K_i^{series}} + \frac{1}{K_i^{series}}$$

$$C = \frac{R}{K_p^{series} \cdot K_i^{series}}, \quad D = \frac{1}{K_i^{series}}$$



비선형 운동방정식

$$\begin{aligned} M\ddot{y} &= -k(y) + M\dot{y} \\ M\ddot{y} &= -v(y) + M\dot{y} \\ M\ddot{y} &= -v^2 + M\dot{y} \end{aligned}$$

$$x_1 = y$$

$$x_2 = \dot{y}$$

$$\dot{x}_1 = \dot{y} = x_2$$

$$\dot{x}_2 = \ddot{y} = -\frac{1}{M}(y^2 + Mg) = -\frac{1}{M}(x_1^2 + Mg)$$

$$\dot{x}_2 = \ddot{y} = -\frac{1}{M}(y^2 + Mg) = -\frac{1}{M}(x_1^2 + Mg)$$

$$\dot{x}_2 = \ddot{y} = -\frac{1}{M}(y^2 + Mg) = -\frac{1}{M}(x_1^2 + Mg)$$

$$\dot{x}_1 = x_2 = 0 \quad \dot{x}_2 = -\frac{1}{M}(x_1^2 + Mg) = 0$$

$$\dot{x}_1 = \dot{y} = x_2 \quad \dot{x}_2 = \ddot{y} = -\frac{1}{M}(y^2 + Mg) = -\frac{1}{M}(x_1^2 + Mg)$$

$$\Delta \dot{x}_1 = x_2 + \Delta x_2 = \Delta x_2$$

$$\Delta \dot{x}_2 = -\frac{1}{M}((x_1 + \Delta x_1)^2 + Mg) = -\frac{1}{M}((\sqrt{Mg} + \Delta x_1)^2 + Mg)$$

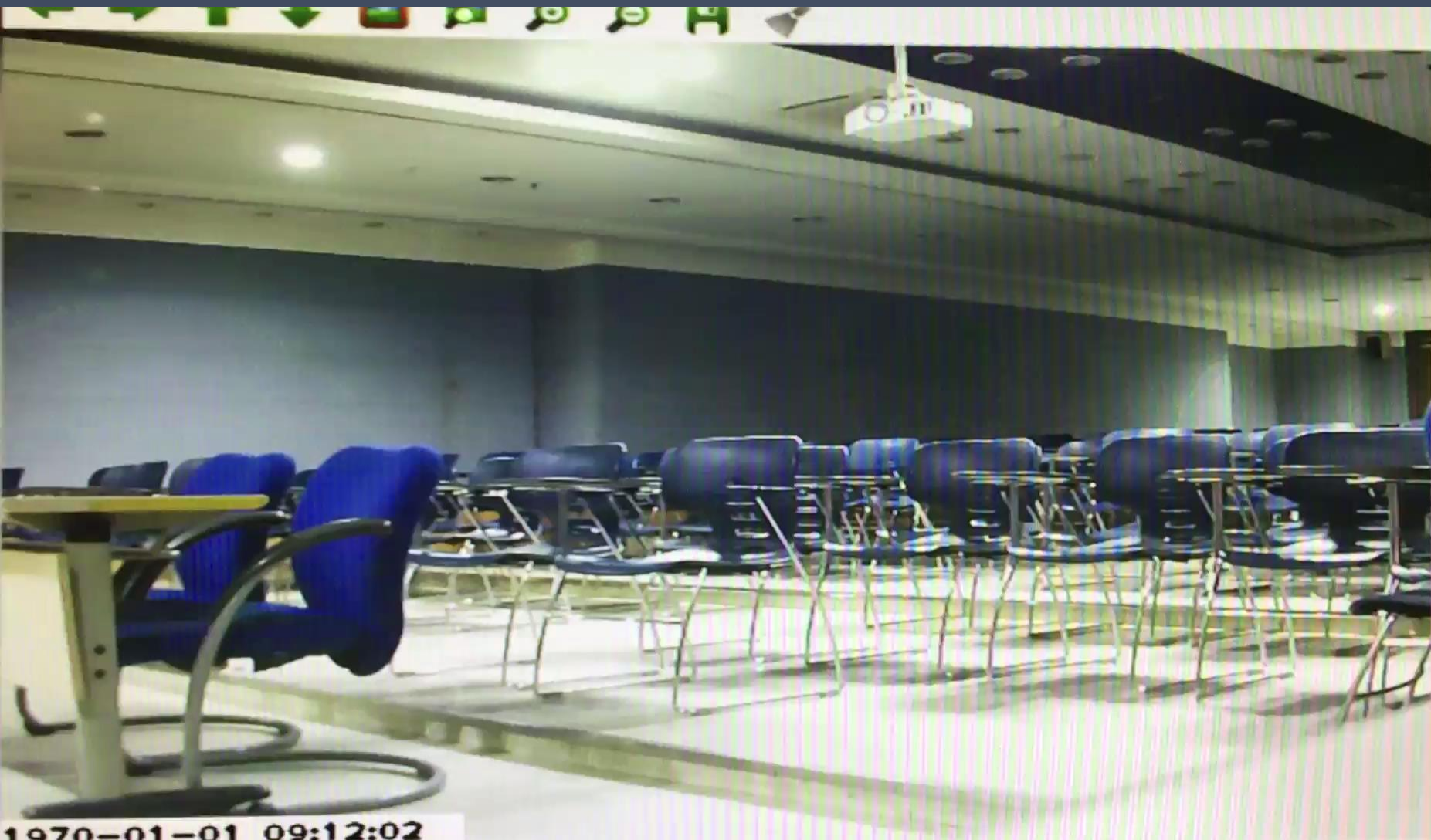
$$\Delta \dot{x}_2 = -\frac{1}{M}((\sqrt{Mg} + \Delta x_1)^2 + Mg) = -\frac{1}{M}((\sqrt{Mg} + \Delta x_1)^2 + Mg)$$

$$\Delta \dot{x}_2 = -\frac{1}{M}((\sqrt{Mg} + \Delta x_1)^2 + Mg) = -\frac{1}{M}((\sqrt{Mg} + \Delta x_1)^2 + Mg)$$

$$\Delta \dot{x}_2 = -\frac{1}{M}((\sqrt{Mg} + \Delta x_1)^2 + Mg) = -\frac{1}{M}((\sqrt{Mg} + \Delta x_1)^2 + Mg)$$

프로젝트 시연







팀원 소개

김민중	이윤성	이대영	정지윤	이상훈 강사님
영상처리	모터제어	음성처리 PCB	네트워크 구성 및 디바이스 제어	서버구축 및 모터제어 PCB

감사합니다