초음파, 부저 제어 및 후방주차센서 프로젝트

경남대학교 김진호

RTES
Real Time Embedded System

2023.11.07

목 차

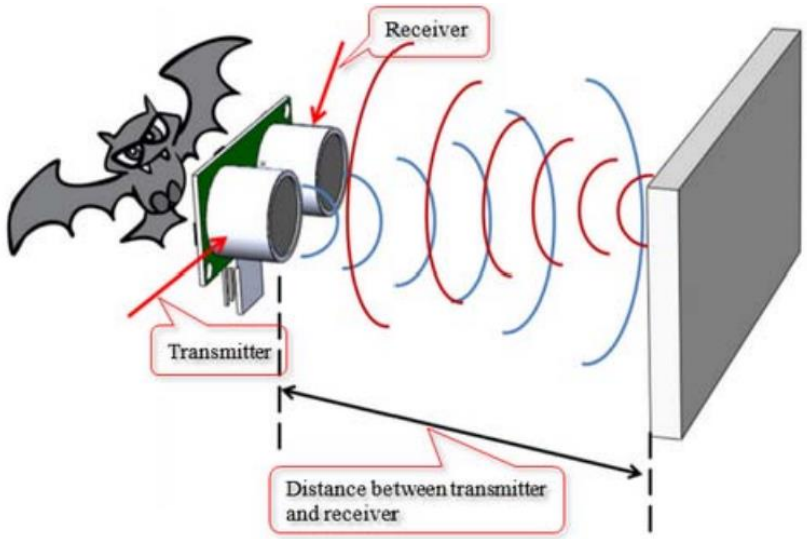
목 차

초음파 센서 개요

- 초음파 센서 모듈
  - 모델명 : HC-SR04
  - 초음파 발신 후 수신까지 소요된 시간을 측정하여 거리 계산 가능
  - 동작 전압 : DC +3.3 ~ 5V
  - 측정 가능 거리 : 2 ~ 400cm
  - Resolution(해상도) : 0.3cm

초음파 센서 개요

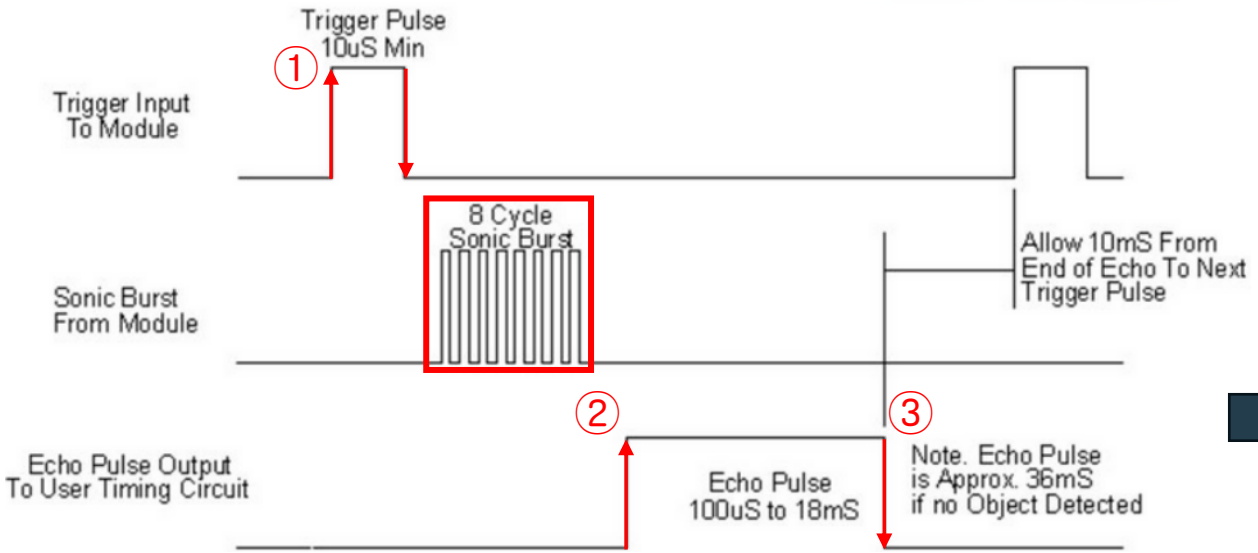
- **TOF (Time of Flight) 기법**
  - 센서가 광선 또는 전파를 방출하고,
  - 물체에 의해 반사되어 수신되기까지의 시간 또는 위상차를 이용하여
  - 물체까지의 거리를 계산하는 방식



- **TOF 기법의 활용**
  - 사람 근접 감지, 거리 계산, 카메라 자동 초점 등에 활용 가능

초음파 센서 개요

- 초음파 센서 동작 원리
  - ① Trig pin에 Trigger Pulse(>10us) 신호(High&Low)를 주면,
  - ② 센서가 초음파 펄스 방출하며 방출이 완료될 때, Echo pin을 High
  - ③ 대상물에 부딪혀 돌아온 신호를 수신했을 때, Echo pin을 Low
  - ④ 이 때 Echo pin에서 발생하는 Rising/Falling Edge를 이용하여 두 Edge 간 시간 간격을 측정한다.
  - ⑤ 측정된 시간과 초음파의 속도를 통해 거리를 측정한다
    - 초음파의 속도는 일반 공기 중에서 약 340m/s



④ 시간 간격 측정
⑤ 거리 계산

RTES
Real Time Embedded System

초음파 센서 – 거리 계산 예시 1

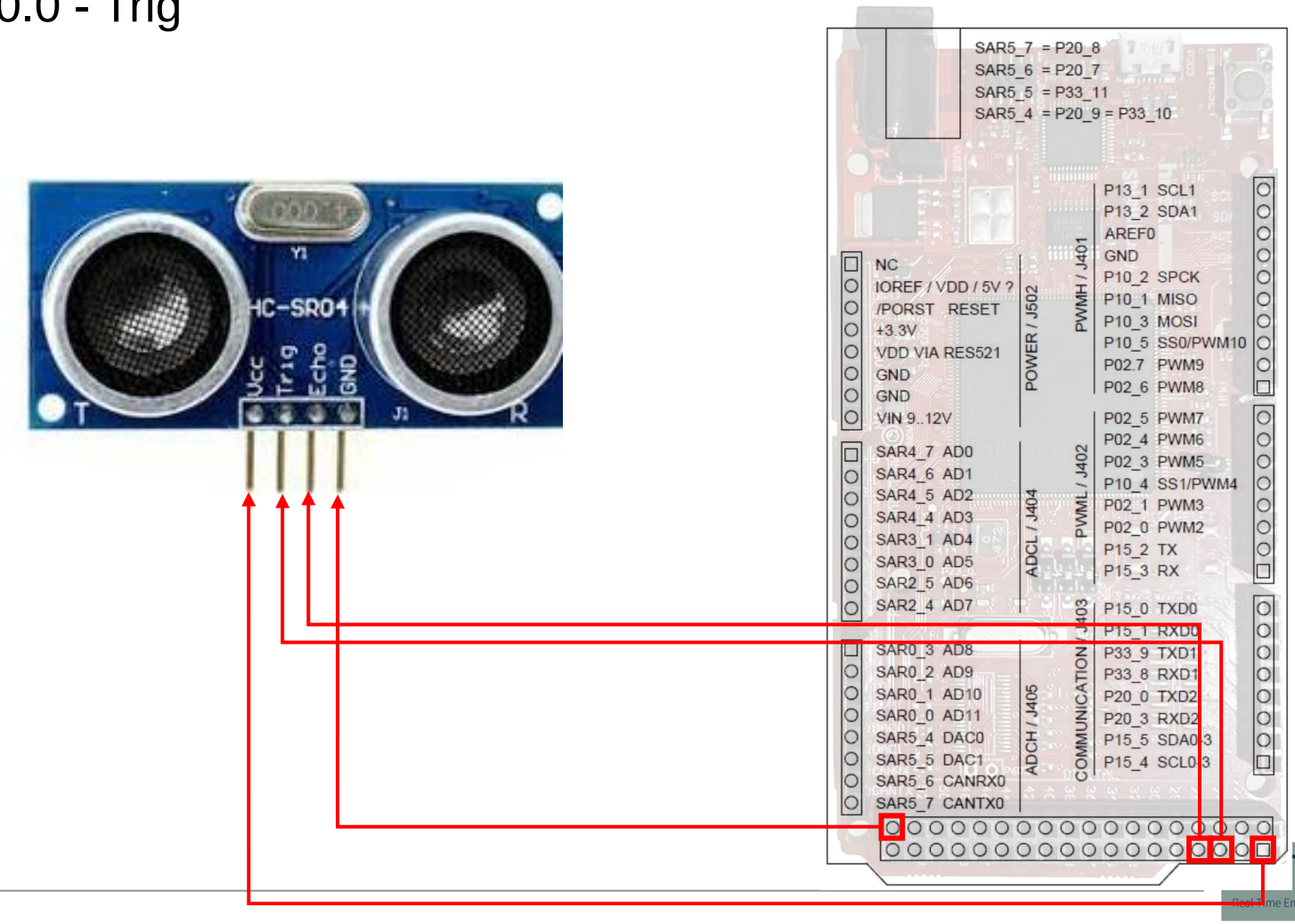
- **ex) 측정된 시간이 0.1초 일 때, 거리 계산**
  - 속도 = 340m/s(일반적인 초음파 속도)
  - 시간 = 0.1초
  - 신호가 왕복한 거리 = 속도 x 시간
    = 340m/s x 0.1초
    = 34m
  - 센서에서 물체까지의 거리 = (신호가 왕복한 거리) / 2
    = 34 / 2
    = 17m
  - 즉, 센서에서 물체까지의 거리는 약 17 m

초음파 센서 – 거리 계산

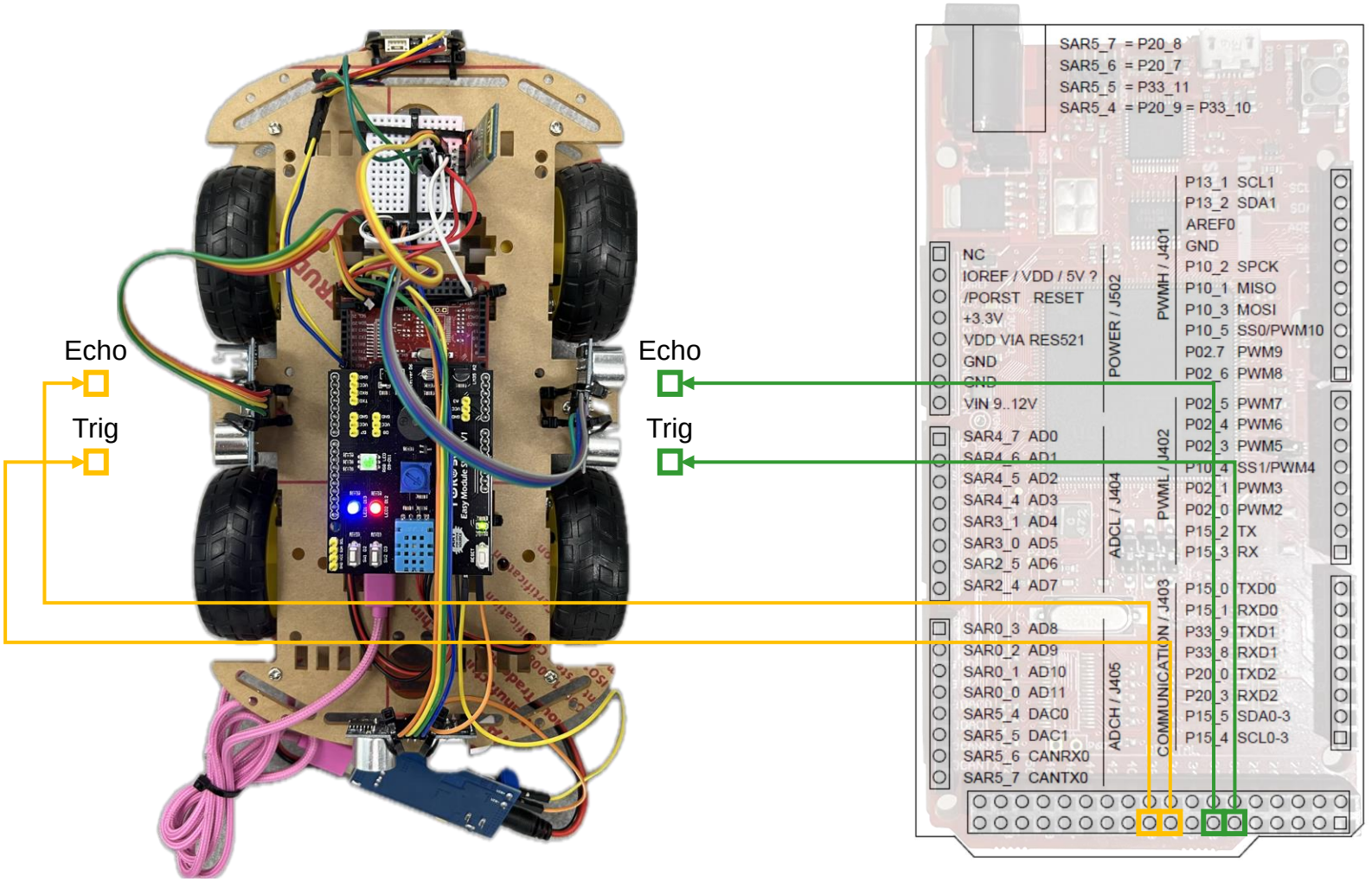
- **Prescaler = 1024로 설정되었을 때 Echo duration이 57 Tick이라면 물체와의 거리는?**
  - 모듈로 공급되는 소스 클럭의 속도 : 100Mhz
  - 100Mhz / 1024 = 97,656.25 이므로 1초에 약 97,656 tick 발생
  - 1 tick = 10.24us (마이크로초)
  - 57 tick = 10.24us * 57 = 583.68us
  - 물체와의 거리 = 음속(0.0343cm/us) * Echo duration / 2
    = 0.0343cm/us * 583.68 / 2
    = 10.01cm

초음파 센서 거리 측정 예제

- **TC275 ShieldBuddy와 결합**
  - P01.1 - Echo
  - P00.0 - Trig



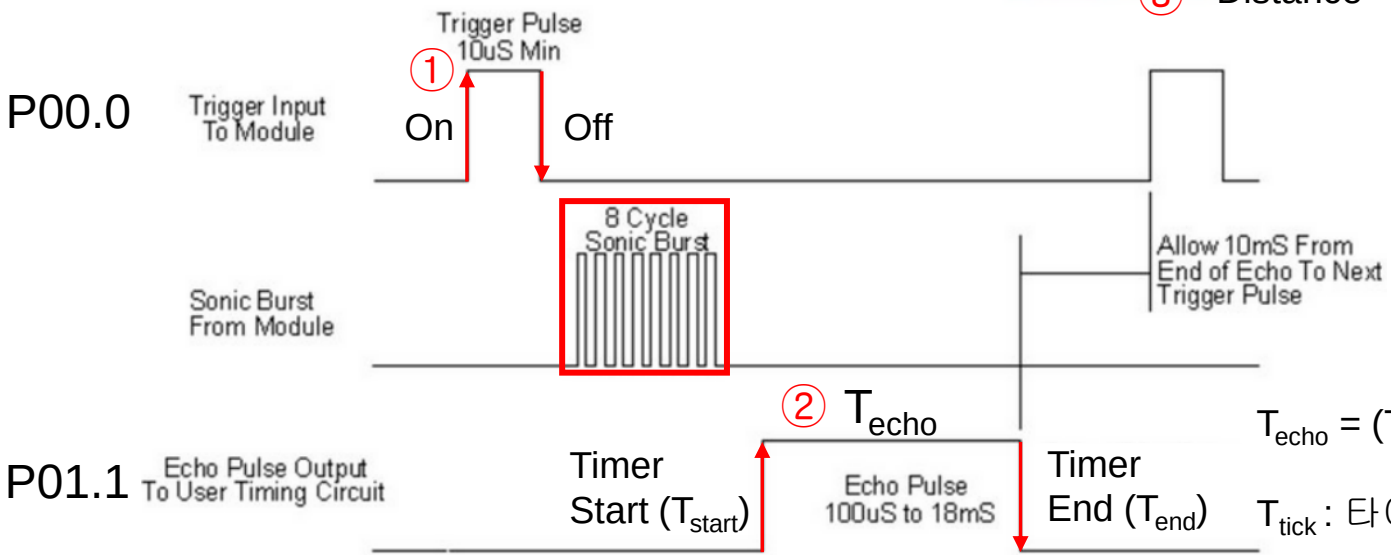
초음파 센서 연결 (좌,우)


Echo
Trig
Echo
Trig
SAR5_7 = P20_8
SAR5_6 = P20_7
SAR5_5 = P33_11
SAR5_4 = P20_9 = P33_10
NC
IOREF / VDD / 5V ?
/PORST RESET
+3.3V
VDD VIA RES521
GND
GND
VIN 9..12V
POWER / J502
SAR4_7 AD0
SAR4_6 AD1
SAR4_5 AD2
SAR4_4 AD3
SAR3_1 AD4
SAR3_0 AD5
SAR2_5 AD6
SAR2_4 AD7
ADCL / J404
SAR0_3 AD8
SAR0_2 AD9
SAR0_1 AD10
SAR0_0 AD11
SAR5_4 DAC0
SAR5_5 DAC1
SAR5_6 CANRX0
SAR5_7 CANTX0
ADCH / J405
P13_1 SCL1
P13_2 SDA1
AREF0
GND
P10_2 SPCK
P10_1 MISO
P10_3 MOSI
P10_5 SS0/PWM10
P02.7 PWM9
P02_6 PWM8
PWMH / J401
P02_5 PWM7
P02_4 PWM6
P02_3 PWM5
P10_4 SS1/PWM4
P02_1 PWM3
P02_0 PWM2
P15_2 TX
P15_3 RX
PWML / J402
P15_0 TXD0
P15_1 RXD0
P33_9 TXD1
P33_8 RXD1
P20_0 TXD2
P20_3 RXD2
P15_5 SDA0-3
P15_4 SCL0-3
COMMUNICATION / J403

초음파 센서 거리 측정 예제

- 초음파 센서 측정 방법
  - ① Trig pin에 Trigger Pulse(>10us) 신호(High&Low)를 출력
  - ② Echo 신호의 High 시간 측정 (T_{echo})
    - 타이머 모듈을 이용하여 시간 측정
  - ③ T_{echo} 를 활용하여 장애물과의 거리 계산
    - 초음파의 속도는 일반 공기 중에서 약 340m/s
    - 거리 = 속도 x 시간
    - 장애물 거리 = 0.0343cm/us * T_{echo} / 2

③ Distance = 0.0343 cm/us * T_{echo} / 2



$T_{echo} = (T_{end} - T_{start}) * T_{tick}$

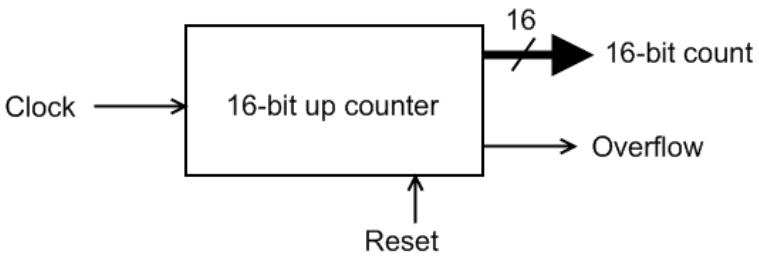
T_{tick} : 타이머 모듈 1 Tick 의 시간 (us)

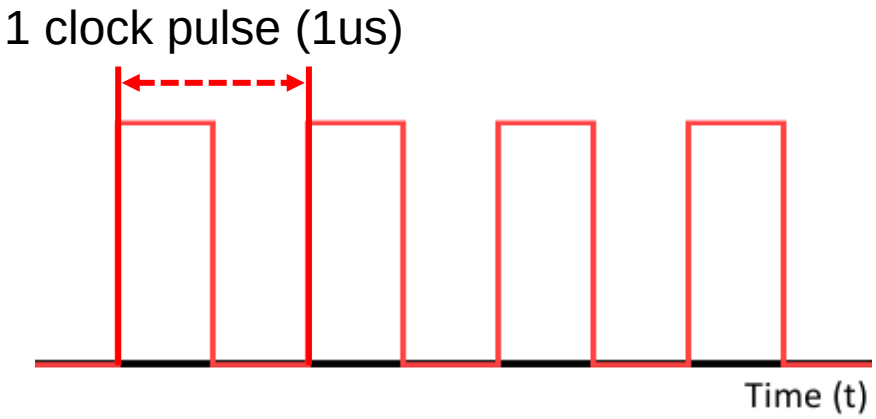
목 차

카운터 & 타이머 기본 동작 원리

- 카운터 동작 원리
  - 입력 클럭 신호의 주파수에 따라 16bit 카운터를 1씩 증가(업카운트)
  - 카운터는 1클럭 펄스마다 1씩 증가함
  - 입력 클럭이 1Mhz인 경우, 각 클럭 주기는 1us임
    - 즉, 카운터 1은 1us를 뜻함
    - 2^{16}=65536-1(0~65535)이므로, 최대 65535 카운트 가능

- 타이머 동작 원리
  - 카운터 overflow가 발생하거나 특정 count를 넘어가면 인터럽트가 발생하여 ISR를 호출함





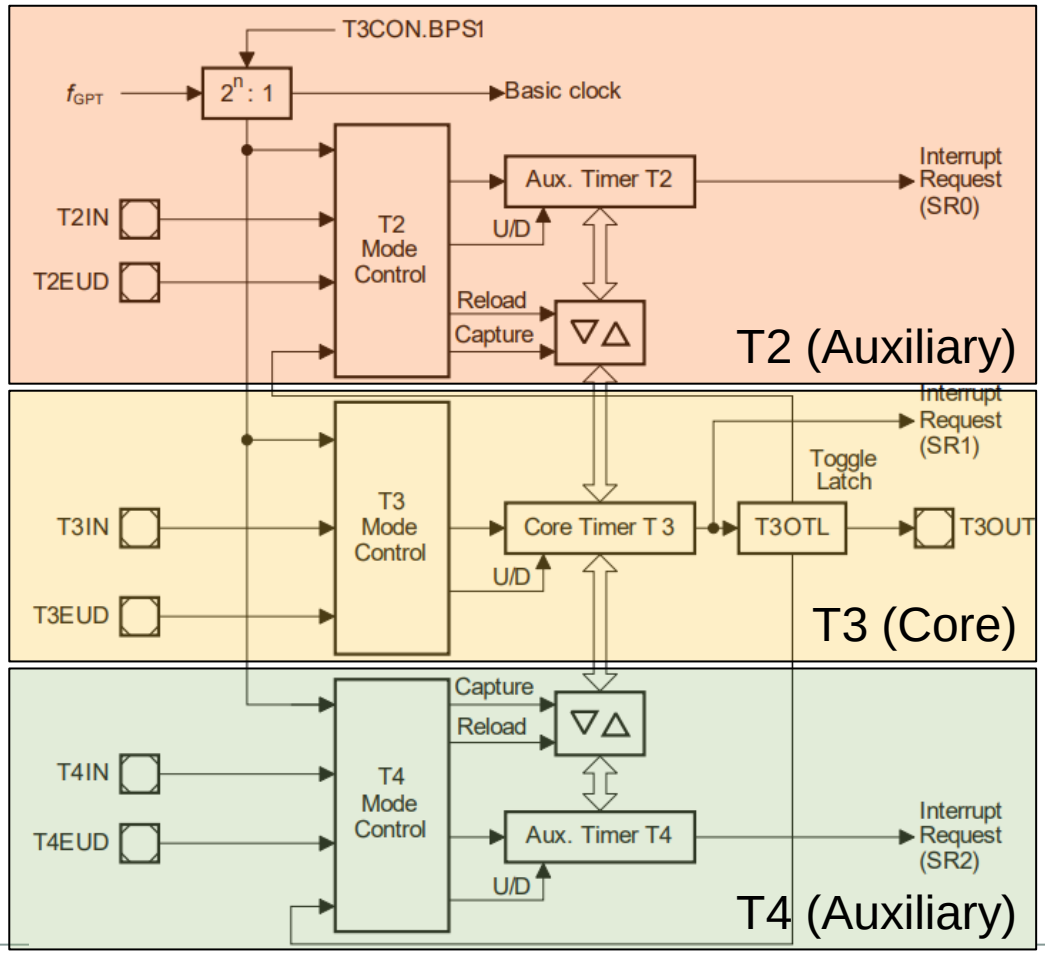
GPT 타이머 모듈 소개

- **GPT(General Purpose Timer) 모듈**
  - GPT12 모듈은 GPT1과 GPT2 유닛 블록으로 구성

- **Block GPT1**
  - 최대 동작 주파수: $f_{SPB}/4$ = 25Mhz
  - 3개의 타이머/카운터 : T2, T3, T4
  - 동작 모드
    - 타이머 모드
    - Gated 타이머 모드
    - 카운터 모드

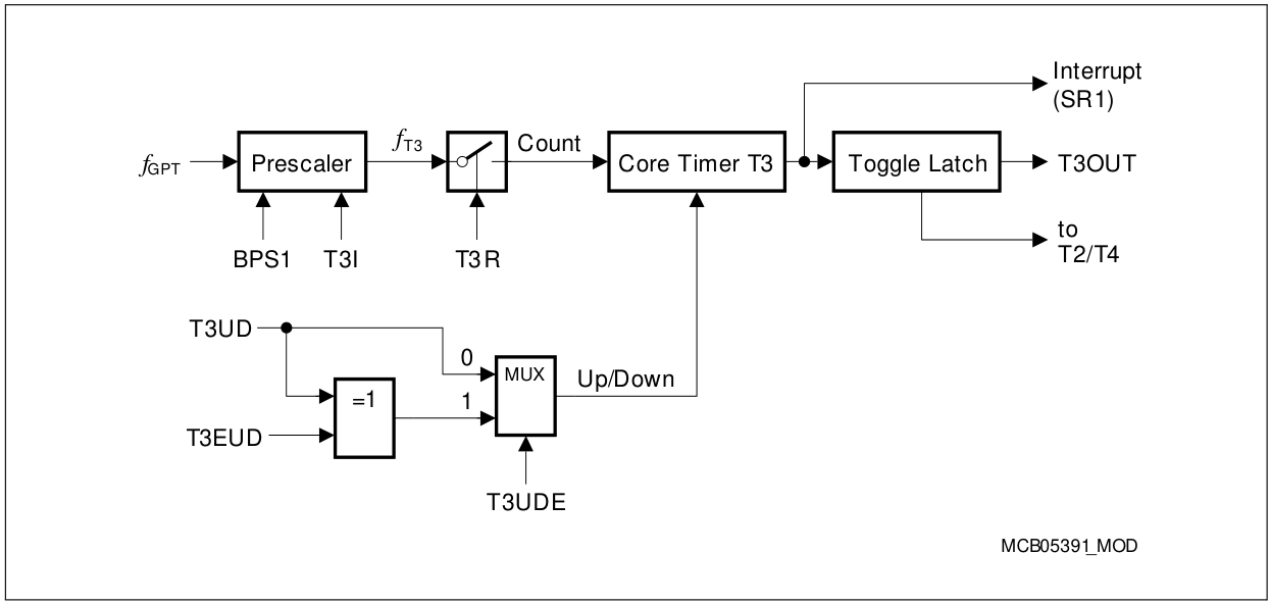
GPT1 블록 다이어그램

- **GPT1 블록 다이어그램**
  - 1개 코어 타이머(T3), 2개 보조 타이머(T2, T4)로 구성
  - 3개 타이머 모두 3가지 모드 지원 (Timer, Gated Timer, Counter)



GPT 타이머 동작 모드

- 타이머 모드: 기본적인 타이머 동작
  - f_{GPT}: GPT 모듈 입력 클럭
  - f_{Tx}: 타이머 입력 클럭
  - TxUD: Up count/ Down count 설정
  - BPS1, TxI: f_{GPT}, f_{Tx} prescaler 설정
  - TxR: 타이머 시작/정지 레지스터

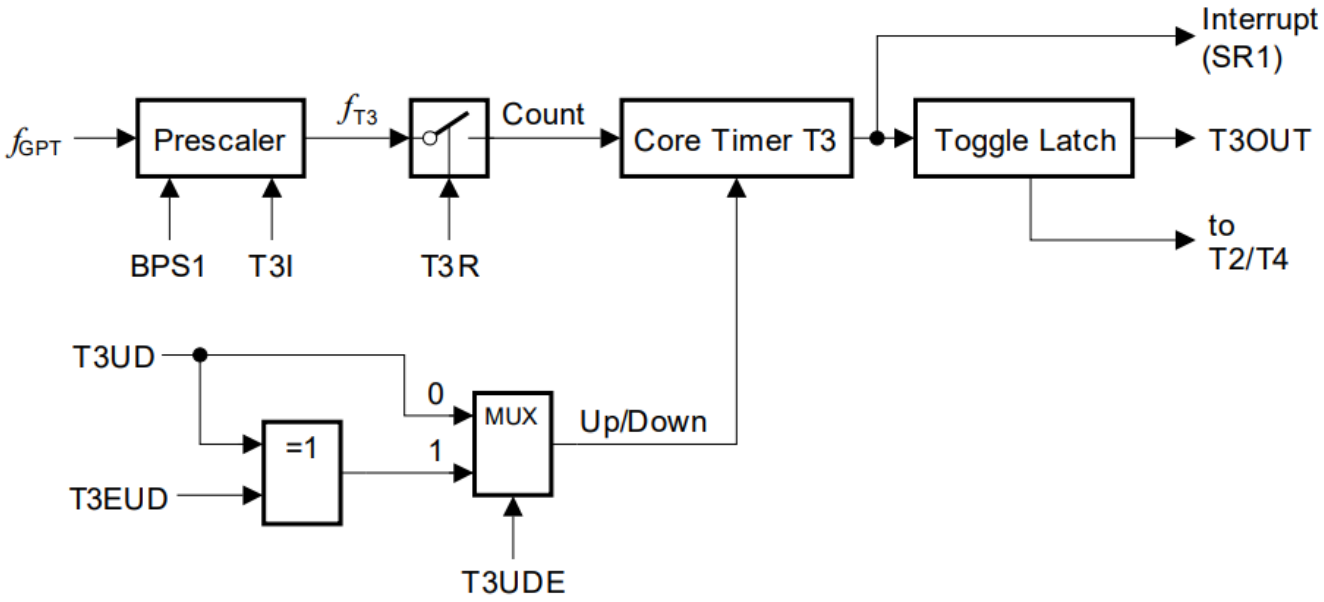


GPT1 코어 타이머 설정

- **GPT1의 T3 코어 타이머 제어**
  - 타이머 모드 동작 순서
    - 타이머 입력 주파수 계산

$$f_{Tx} = \frac{f_{GPT}}{F(BPS1) \times 2^{<TxI>}}$$

    - T3UD Bit field로 Up/ Down Count 설정
    - T3R = 1 일 때, 타이머 시작
    - 특정 조건으로 인터럽트 가능



GPT1 코어 타이머 설정

- **GPT1의 T3 코어 타이머 모드 frequency 설정 방법**
  - $f_{GPT} = f_{SPB} = 100MHz$ (기본 시스템 클럭)
  - f_{GPT} 를 BPS1, T3I Bit field를 참고하여 Prescale하여 f_{T3} 를 계산
  - ex) T3I = 000_B, BPS1 = 01_B 로 설정했을 때,

    → 100MHz / 4 = 25.0MHz (= f_{T3})

Table 27-7 GPT1 Overall Prescaler Factors for Internal Count Clock (Timer Mode and Gated Timer Mode)

Individual Prescaler for Tx	Common Prescaler for Module Clock[1)]			
	BPS1 = 01_B	**BPS1 = 00_B**	**BPS1 = 11_B**	**BPS1 = 10_B**
TxI = 000_B	4	8	16	32
TxI = 001_B	8	16	32	64
TxI = 010_B	16	32	64	128
TxI = 011_B	32	64	128	256
TxI = 100_B	64	128	256	512
TxI = 101_B	128	256	512	1024
TxI = 110_B	256	512	1024	2048
TxI = 111_B	512	1024	2048	4096

GPT1 코어 타이머 설정

- **GPT1의 T4 코어 타이머 모드 frequency 설정 방법**
  - $f_{GPT} = f_{SPB} = 100MHz$ (기본 시스템 클럭)
  - f_{GPT} 를 BPS1, T3I Bit field를 참고하여 Prescale하여 f_{T3} 를 계산
  - ex) T4I = 101_B, BPS1 = 10_B 로 설정했을 때,
    → 100MHz / 1024 = 97.65625 KHz (T_{tick} = 10.24 us)

Table 27-7 GPT1 Overall Prescaler Factors for Internal Count Clock (Timer Mode and Gated Timer Mode)

Individual Prescaler for Tx	Common Prescaler for Module Clock[1)]			
	BPS1 = 01_B	**BPS1 = 00_B**	**BPS1 = 11_B**	**BPS1 = 10_B**
TxI = 000_B	4	8	16	32
TxI = 001_B	8	16	32	64
TxI = 010_B	16	32	64	128
TxI = 011_B	32	64	128	256
TxI = 100_B	64	128	256	512
TxI = 101_B	128	256	512	1024
TxI = 110_B	256	512	1024	2048
TxI = 111_B	512	1024	2048	4096

목 차

타이머 모듈 활용

- **gpt12.c 파일 내 관련 함수 제공되어 있음**
  - 타이머 초기화 함수에서 T4 타이머를 아래와 같이 설정함

```
void init_gpt1(void) {
    /* Initialize the GPT12 module */
    IfxGpt12_enableModule(&MODULE_GPT120); /* Enable the GPT12 module */

    /* Initialize the Timer T3 */
    MODULE_GPT120.T3CON.B.BPS1 = 0x2; /* Set GPT1 block prescaler: 32 */

    ... 중략 ...
    /* Initialize the Timer T4 for Ultrasonic */
    IfxGpt12_T4_setMode(&MODULE_GPT120, IfxGpt12_Mode_timer);
    IfxGpt12_T4_setTimerDirection(&MODULE_GPT120, IfxGpt12_TimerDirection_up);
    IfxGpt12_T4_setTimerPrescaler(&MODULE_GPT120, IfxGpt12_TimerInputPrescaler_32);
    IfxGpt12_T4_setTimerValue(&MODULE_GPT120, 0u);
    ... 하략 ...
}
```

※ GPT1 Prescaler 설정
- Block Prescaler: 32

※ T4 타이머 설정
- T4 Input Prescaler: 2^5=32
- Total Prescaler: 1024
- Count Direction : Up

타이머 모듈 활용

T4 타이머 시작 / 종료

```
void runGpt12_T4()
{
    IfxGpt12_T4_run(&MODULE_GPT120, IfxGpt12_TimerRun_start);
}

void stopGpt12_T4()
{
    IfxGpt12_T4_run(&MODULE_GPT120, IfxGpt12_TimerRun_stop);
}
```

T4 타이머 값 읽기 / 쓰기

```
void setGpt12_T4(unsigned short value)
{
   IfxGpt12_T4_setTimerValue(&MODULE_GPT120, value);}

unsigned short getGpt12_T4()
{
   return IfxGpt12_T4_getTimerValue(&MODULE_GPT120);
}
```

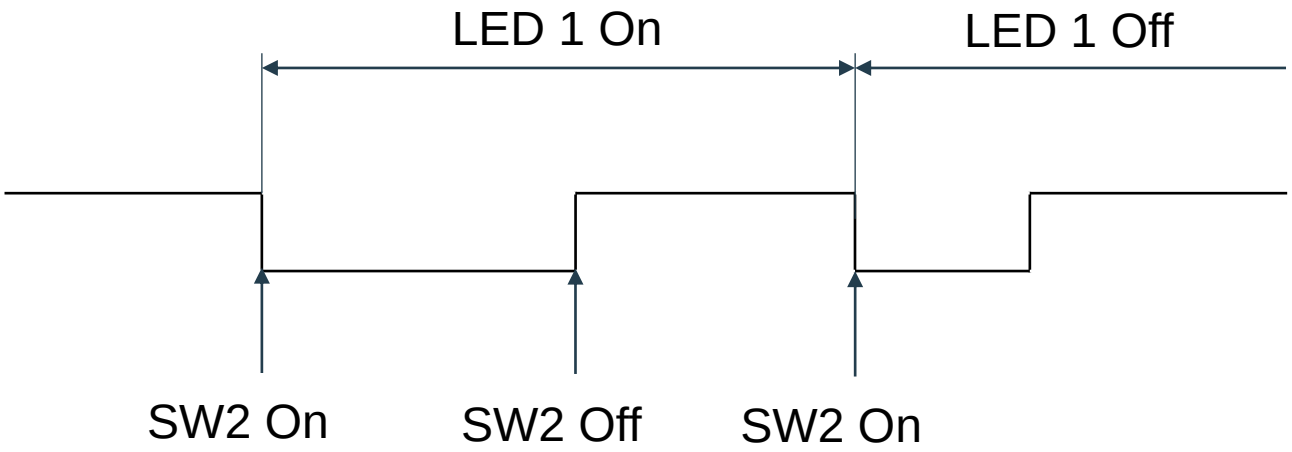
타이머 예제

- 두 소스 코드의 실행 시간을 측정하라

```c
#include "main.h"

int core0_main(void) {
   volatile int i, j;
   unsigned int timer_end;
   float execTime;
   char buf[100];

   _init_uart3();
   init_gpt1();

   setGpt12_T4(0);

   runGpt12_T4();
   for (i = 0; i < 1; i++) {
      for (j = 0; j < 18200; j++) {
         continue;
      }
   }
   stopGpt12_T4();

   timer_end = getGpt12_T4();
   execTime = (timer_end - 0) * 10.24;

   sprintf(buf, "Execution time : %f us\n", execTime);
   my_printf("%s", buf);

   return 0;
}
```

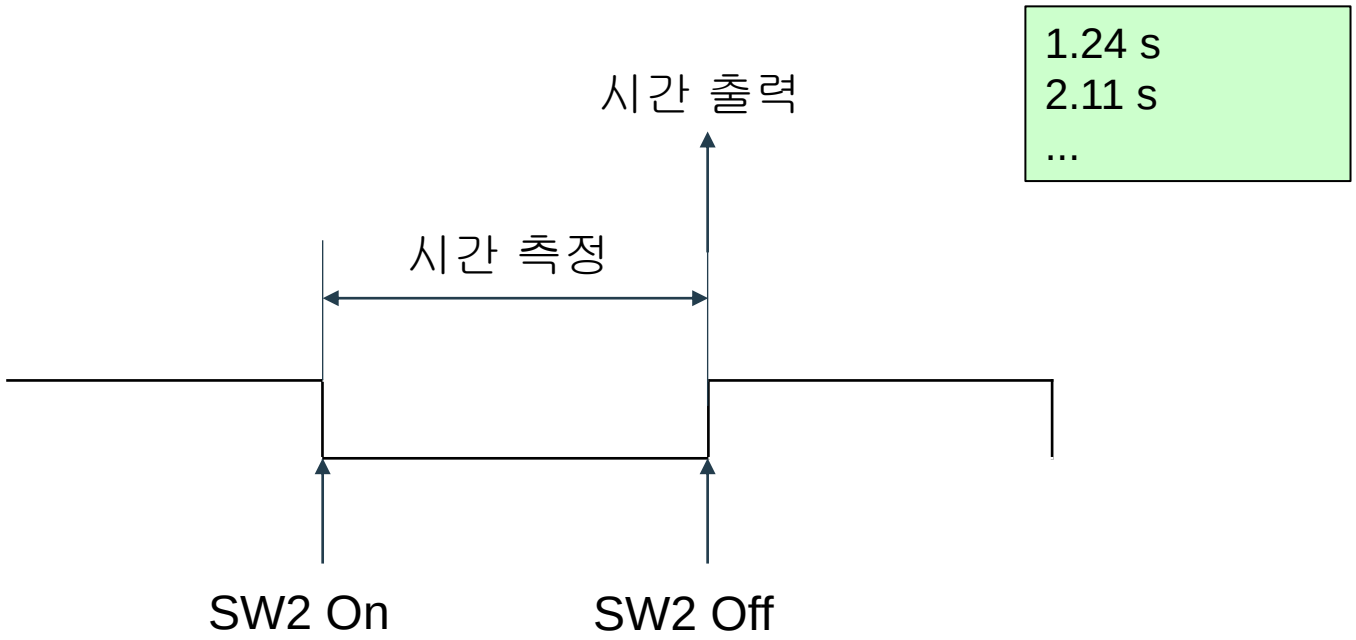
Push Button을 이용한 LED 실습

- **Push 버튼을 누르면 LED 1 이 On / Off 되도록 작성하라.**
  - 처음 누르면 LED 1 켜짐
  - 두번째 누르면 LED 1 꺼짐



타이머 실습

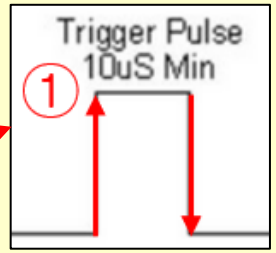
- **SW2 을 누르고 있는 시간을 측정하여 출력하라.**
  - SW2를 누르면 측정 시작
  - SW2를 Off 하면 my_printf() 시간 출력

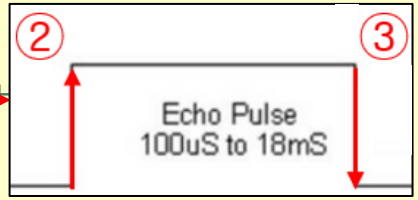


초음파 센서 예시

```
int core0_main(void) {
  volatile int j=0;
  unsigned int timer_end;
  float rear_duration, distance;

  _init_uart3();
  Init_Ultrasonics();

  while(1){
    /* Send Trigger Pulse */
    IfxPort_setPinHigh(REAR_TRIG); // Rear TRIG_HIGH
    for(j=0; j<1000; j++) continue;
    IfxPort_setPinLow(REAR_TRIG);  // Rear TRIG_LOW

    /* Calculate Distance */
    IfxGpt12_T4_setTimerValue(&MODULE_GPT120, 0);
    while (IfxPort_getPinState(REAR_ECHO) == 0); // wait for Rear ECHO_HIGH
    IfxGpt12_T4_run(&MODULE_GPT120, IfxGpt12_TimerRun_start);
    while (IfxPort_getPinState(REAR_ECHO) == 1); // wait for Rear ECHO_LOW
    IfxGpt12_T4_run(&MODULE_GPT120, IfxGpt12_TimerRun_stop);

    timer_end = IfxGpt12_T4_getTimerValue(&MODULE_GPT120);
    rear_duration = (timer_end - 0) * 10.24;

    distance = 0.0343 * rear_duration / 2.0;    // cm

    my_printf("Distance : %d cm\n", (int)distance);
  }
  return 0;
}
```

Distance : 16 cm
Distance : 17 cm
Distance : 16 cm
…





④ 시간 간격 측정

⑤ 거리 계산

UltraSonic 모듈

- **UltraSonic 모듈 제공 함수**
  - **void Init_Ultrasonics(void);**
  - **float ReadRearUltrasonic_noFilt();**

```
#define REAR_TRIG        &MODULE_P00,0
#define REAR_ECHO        &MODULE_P00,1

void Init_Ultrasonics()
{
  /* Init Rear Ultrasonic Pin */
  IfxPort_setPinModeOutput(REAR_TRIG, IfxPort_OutputMode_pushPull, IfxPort_OutputIdx_general);
  IfxPort_setPinModeInput(REAR_ECHO, IfxPort_InputMode_pullDown);
  (...)

  // Init GPT1 for T4 timer
  init_gpt1();
}
```

UltraSonic 모듈

float ReadRearUltrasonic_noFilt()

```
float ReadRearUltrasonic_noFilt()
{
  volatile int j=0;
  unsigned int timer_end;
  float rear_duration;
  float distance;

  /* Send Trigger Pulse */
  IfxPort_setPinHigh(REAR_TRIG); // Rear TRIG_HIGH
  for(j=0; j<1000; j++) continue;
  IfxPort_setPinLow(REAR_TRIG); // Rear TRIG_LOW

  /* Calculate Distance */
  IfxGpt12_T4_setTimerValue(&MODULE_GPT120, 0);
  while (IfxPort_getPinState(REAR_ECHO) == 0); // wait for Rear ECHO_HIGH
  IfxGpt12_T4_run(&MODULE_GPT120, IfxGpt12_TimerRun_start);
  while (IfxPort_getPinState(REAR_ECHO) == 1); // wait for Rear ECHO_LOW
  IfxGpt12_T4_run(&MODULE_GPT120, IfxGpt12_TimerRun_stop);

  timer_end = IfxGpt12_T4_getTimerValue(&MODULE_GPT120);
  rear_duration = (timer_end - 0) * 10.24;

  distance = 0.0343 * rear_duration / 2.0; // cm/us
  return distance;
}
```

초음파 센서 예제

- 초음파 센서의 거리가 **10cm** 이하이면 **LED 1**을 **On**하라.
  - 10cm 이상이면 LED 1은 Off 됨

```
#include "main.h"
#include "asclin.h"
#include "GPIO.h"
#include "Ultrasonic.h"

int core0_main(void) {
  _init_uart3();
  Init_Ultrasonics();
  Init_GPIO();

  while (1) {
    float distance = ReadRearUltrasonic_noFilt();
    if (distance < 10.0f) {
       setLED1(1);
    } else
       setLED1(0);

    my_printf("Distance = %d\n", (int) distance);
    delay_ms(10);
  }

  return 0;
}
```

목 차

- 초음파 센서 제어
  - 초음파 센서 개요
  - 타이머 모듈 동작 원리
  - 타이머 모듈을 활용한 초음파 센서 거리 측정
  - 필터를 이용한 초음파 센서 실습
- 수동 부저 제어
- 후방주차센서 실습

이동 평균 필터

- 이동 평균 필터
  - 인접한 n개 데이터의 평균을 구하여 순차적으로 데이터의 평균을 구하는 방법

n개의 데이터에 대한 이동평균

$$\overline{x_k} = \frac{x_{k-n+1}+x_{k-n+2}+\cdots+x_k}{n}$$

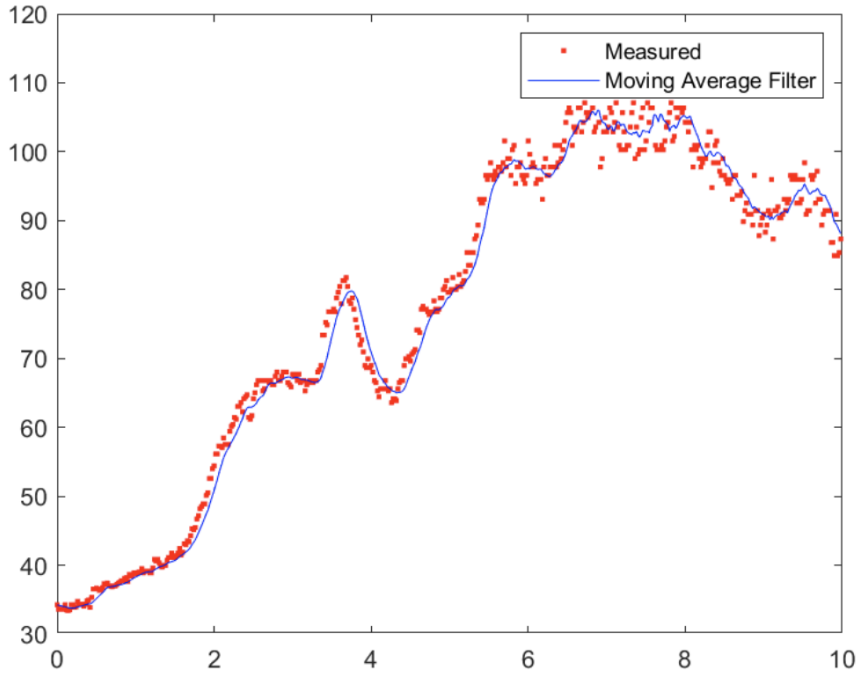
이전 스텝의 이동평균

$$\overline{x_{k-1}} = \frac{x_{k-n}+x_{k-n+1}+\cdots+x_{k-1}}{n}$$

이동평균 필터

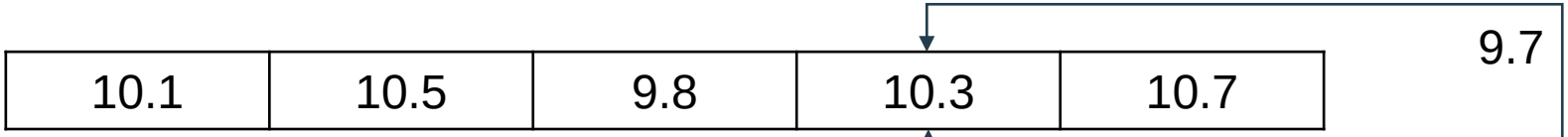
$$\overline{x_k} - \overline{x_{k-1}} = \frac{x_k - x_{k-n}}{n}$$

$$\overline{x_k} = \overline{x_{k-1}} + \frac{x_k - x_{k-n}}{n}$$



이동 평균 필터 구현 방법

- 순환 버퍼 활용
  - 센서의 값을 일정한 크기의 배열에 순차적으로 저장함 (4 -> 0)
  - 가장 오래된 값이 저장된 인덱스를 알 수 있음 (Old_Index)
  - 새로운 데이터를 읽으면, 가장 오래된 값을 지우고 새로운 값으로 변경함 (Old_Index 는 +1 됨)
  - 기존 평균 : (10.1 + 10.5 + 9.8 + 10.3 + 10.7) / 5 = 10.28
  - 새로운 이동 평균 : 10.28 + (9.7 – 10.3)/5 = 10.16



이동평균 필터

$$\overline{x_k} - \overline{x_{k-1}} = \frac{x_k - x_{k-n}}{n}$$

$$\overline{x_k} = \overline{x_{k-1}} + \frac{x_k - x_{k-n}}{n}$$

이동 평균 필터 예제

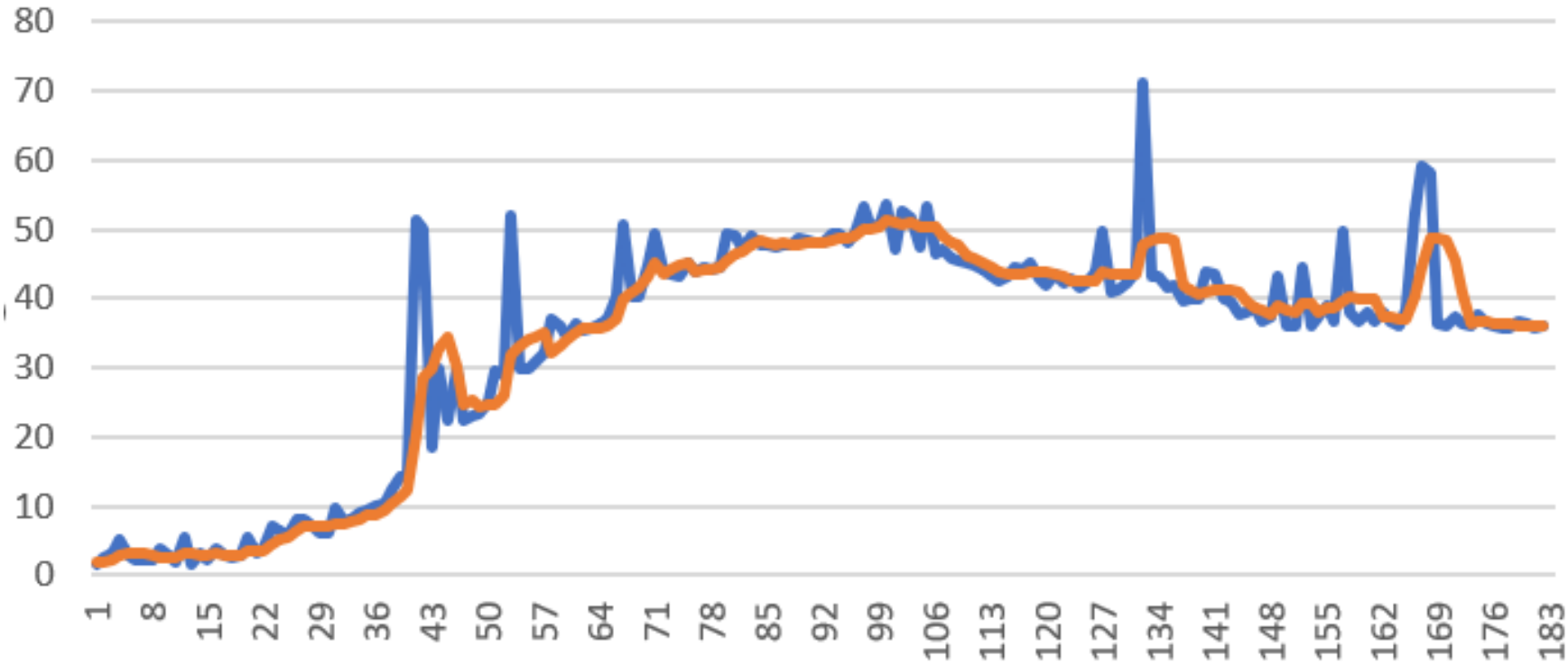
```c
#include "main.h"
#include "GPIO.h"
#include "Ultrasonic.h"

#define FILT_SIZE 5
int core0_main(void) {
   float distance_nofilt;
   float avg_filt_buf[FILT_SIZE] = { 0, };
   int old_index = 0;
   float distance_filt;
   int sensorRxCnt = 0;
   char buf[100];

   _init_uart3();
   Init_Ultrasonics();
```

```c
   while (1) {
      distance_nofilt = ReadRearUltrasonic_noFilt();

      ++old_index;
      old_index %= FILT_SIZE;  // Buffer Size = 5
      avg_filt_buf[old_index] = distance_nofilt;
      sensorRxCnt++;

      /* Calculate Moving Average Filter */
      if (sensorRxCnt >= FILT_SIZE) {
         float sum = 0;
         for (int i = 0; i < FILT_SIZE; i++) {
           sum += avg_filt_buf[i];
         }
         distance_filt = sum / FILT_SIZE;
      } else
        distance_filt = distance_nofilt;

      sprintf(buf, "%f, %f\n", distance_nofilt, distance_filt);
      my_printf("%s", buf);
   }
   return 0;
}
```

```
36.703743, 35.228565
36.879356, 35.333935
36.176895, 35.298817
30.381567, 35.298817
29.503487, 33.929008
37.230591, 34.034382
36.352509, 33.929008
20.195839, 30.732800
18.615295, 28.379543
19.493376, 26.377522
20.547071, 23.040819
```

이동 평균 필터 결과

- 파랑색 : 필터 미적용
- 주황색 : 이동 평균 필터 적용 (최근 **5**개)



UltraSonic 모듈

- **UltraSonic 모듈 제공 함수**
  - void Init_Ultrasonics(void);
    - 초음파 센서 초기화 함수
  - float ReadRearUltrasonic_noFilt();
  - float ReadRearUltrasonic_Filt();
    - 후방 초음파 센서 값 수신 함수
  - float ReadLeftUltrasonic_noFilt();
  - float ReadLeftUltrasonic_Filt();
    - 좌측 초음파 센서 값 수신 함수
  - float ReadRightUltrasonic_noFilt();
  - float ReadRightUltrasonic_Filt();
    - 우측 초음파 센서 값 수신 함수

이동 평균 필터 적용 초음파 센서 값 읽기 예제

- **Ultrasonic.c** 파일에 아래 함수 참조

```
#define FILT_SIZE 5

float ReadRearUltrasonic_Filt()
{
  float distance_nofilt;
  static float avg_filt_buf[10] = {0,};
  static int old_index = 0;
  float distance_filt;
  static int sensorRxCnt = 0;

  distance_nofilt = ReadRearUltrasonic_noFilt();

  ++old_index;
  old_index %= FILT_SIZE;  // Buffer Size = 5
  avg_filt_buf[old_index] = distance_nofilt;
  sensorRxCnt++;

  /* Calculate Moving Average Filter */
  if (sensorRxCnt > FILT_SIZE) {
    float sum = 0;
    for (int i = 0; i < FILT_SIZE; i++) { sum += avg_filt_buf[i]; }
    distance_filt = sum / FILT_SIZE;
  }
  else
    distance_filt = distance_nofilt;

  return distance_filt;
}
```

이동 평균 필터 적용 초음파 센서 값 읽기 예제

- ReadRearUltrasonic_Filt() 활용 센서 값 예제

```
int core0_main(void) {
  float distance_filt;
  char buf[100];

  _init_uart3();
  Init_Ultrasonics();

  while (1) {
    distance_filt = ReadRearUltrasonic_Filt();

    sprintf(buf, "%f\n", distance_filt);
    my_printf("Distance : %s", buf);
  }
  return 0;
}
```

```
Distance : 9.202278
Distance : 9.167154
Distance : 9.167154
Distance : 9.167154
Distance : 9.167154
Distance : 9.167154
Distance : 9.202278
Distance : 9.202278
Distance : 9.272524
Distance : 9.272524
Distance : 9.237401
...
```

이동 평균 필터 적용 초음파 센서 값 읽기 실습

- 아래 프로그램은 정상 동작하지 않는다.
  - 앞의 예제와 비교하여 원인을 파악하고 개선하라

```c
int core0_main(void) {
  float distance_filt;
  char buf[100];

  _init_uart3();
  Init_Ultrasonics();

  while(1)
  {
    distance_filt = ReadRearUltrasonic_Filt();

    sprintf(buf, "%f\n", distance_filt);
    my_printf("%s", buf);
  }

  return 0;
}
```

16.683519

```c
float ReadRearUltrasonic_noFilt()
{
    volatile int j=0;
    unsigned int timer_end;
    float rear_duration;
    float distance;

    /* Send Trigger Pulse */
    MODULE_P00.OUT.B.P0 = 1; // Rear TRIG_HIGH
    for(j=0; j<1000; j++) continue;
    MODULE_P00.OUT.B.P0 = 0; // Rear TRIG_LOW

    /* Calculate Distance */
    setGpt12_T4(0);
    while (MODULE_P00.IN.B.P1 == 0); // wait for Rear ECHO_HIGH
    runGpt12_T4();
    while (MODULE_P00.IN.B.P1 == 1); // wait for Rear ECHO_LOW
    stopGpt12_T4();

    timer_end = getGpt12_T4();
    rear_duration = (timer_end - 0) * 10.24;

    distance = 0.0343 * rear_duration / 2.0; // cm/us
    return distance;
}
```

목 차

- 초음파 센서 제어
- 수동 부저 제어
  - 수동 부저 개요
  - 폴링 기반 수동 부저 구동
  - 타이머 인터럽트 기반 수동 부저 구동
- 후방주차센서 실습

부저(Buzzer)

- **Passive Buzzer(수동 부저)**
  - 진동수(Hz)에 따라 소리의 높낮이를 다르게 출력하는 스피커
  - 아래 표를 참고하여 주파수를 입력하면 원하는 음을 출력할 수 있음

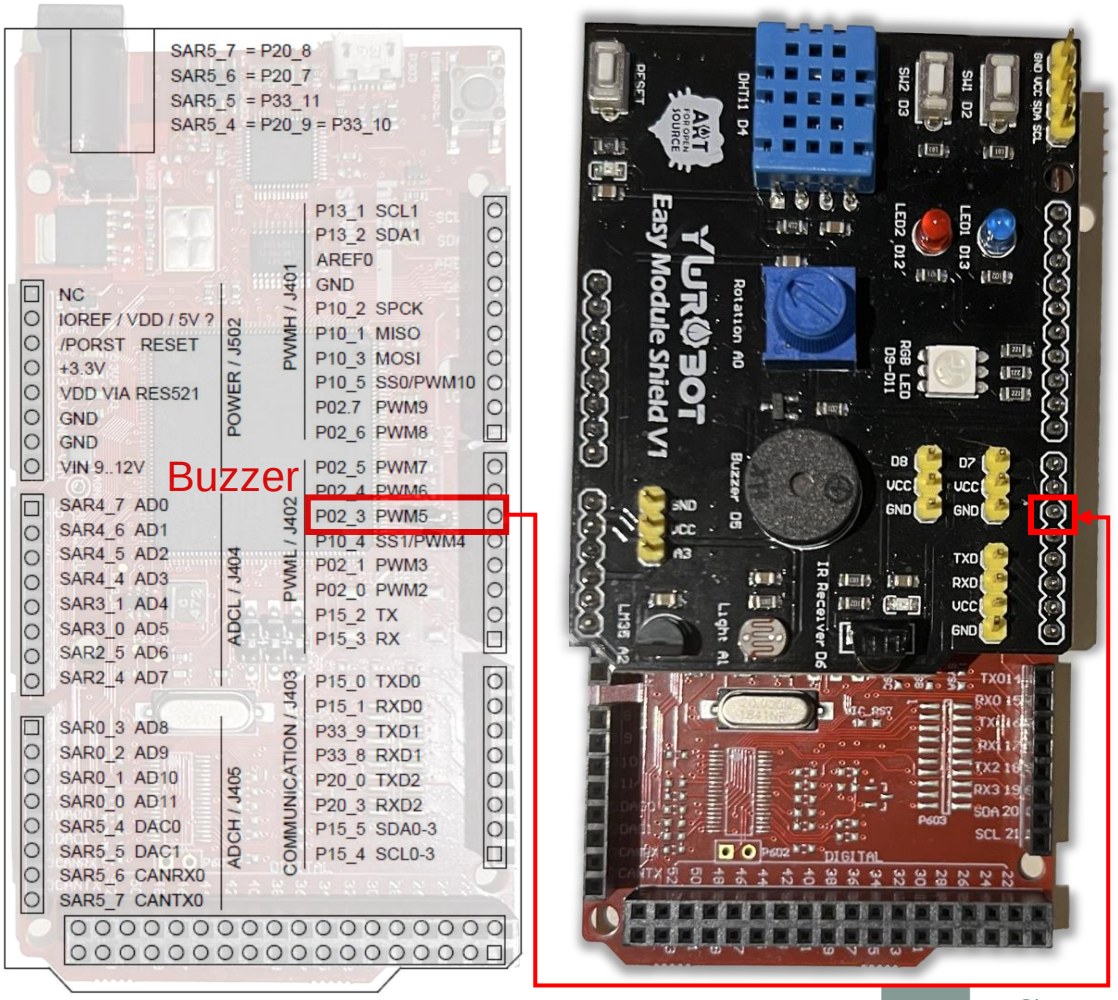
(단위 : Hz)

옥타브 / 음계	1	2	3	4	5	6	7	8
C(도)	32.7032	65.4064	130.8128	261.6256	523.2511	1046.502	2093.005	4186.009
C#	34.6478	69.2957	138.5913	277.1826	554.3653	1108.731	2217.461	4434.922
D(레)	36.7081	73.4162	146.8324	293.6648	587.3295	1174.659	2349.318	4698.636
D#	38.8909	77.7817	155.5635	311.1270	622.2540	1244.508	2489.016	4978.032
E(미)	41.2034	82.4069	164.8138	329.6276	659.2551	1318.510	2637.020	5274.041
F(파)	43.6535	87.3071	174.6141	349.2282	698.4565	1396.913	2793.826	5587.652
F#	46.2493	92.4986	184.9972	369.9944	739.9888	1479.978	2959.955	5919.911
G(솔)	48.9994	97.9989	195.9977	391.9954	783.9909	1567.982	3135.963	6271.927
G#	51.9130	103.8262	207.6523	415.3047	830.6094	1661.219	3322.438	6644.875
A(라)	55.0000	110.0000	220.0000	440.0000	880.0000	1760.000	3520.000	7040.000
A#	58.2705	116.5409	233.0819	466.1638	932.3275	1864.655	3729.310	7458.620
B(시)	61.7354	123.4708	246.9417	493.8833	987.7666	1975.533	3951.066	7902.133

부저 핀 연결

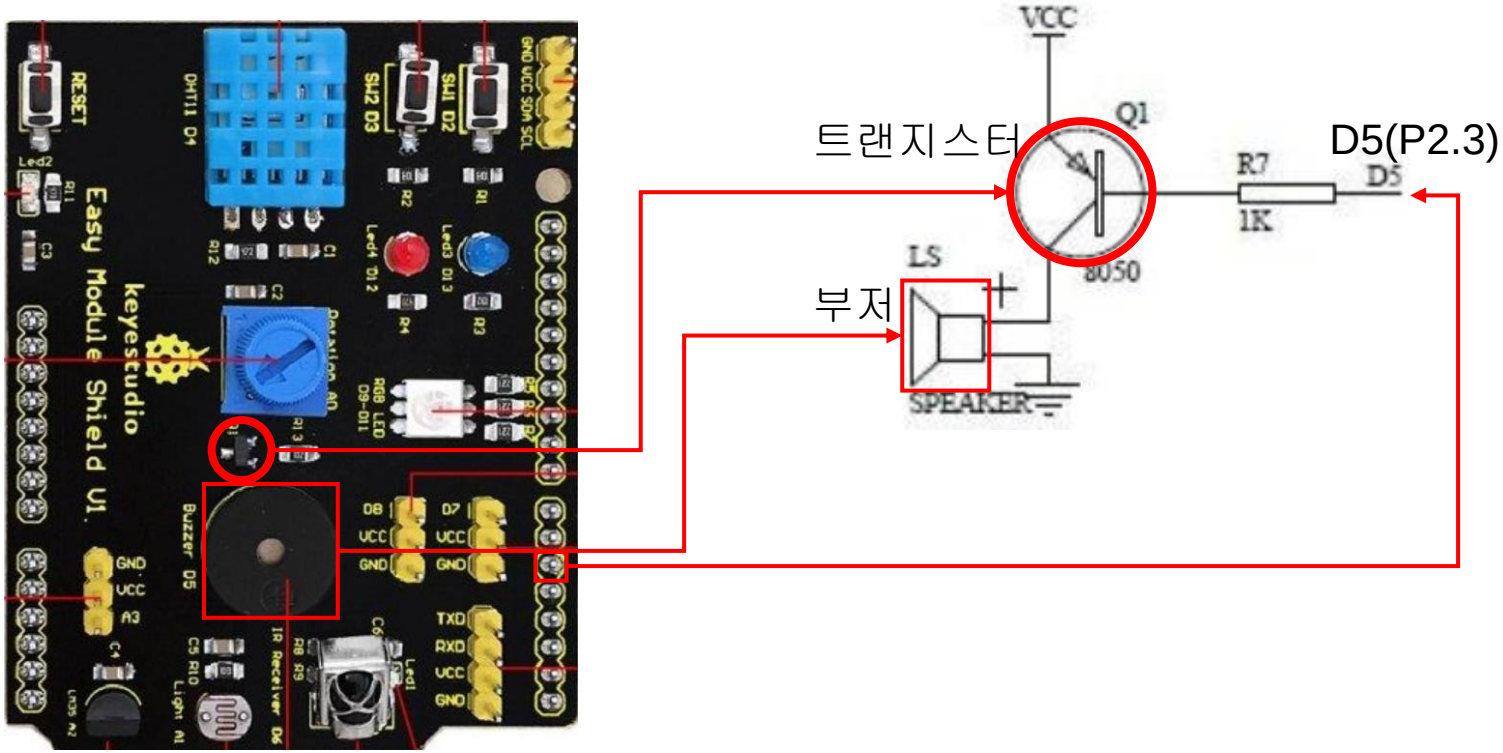
- **TC275와 Easy Shield Module 결합**
  - Buzzer(D5)는 P2.3과 연결되어 있음

Digital pin 0 (RX0)	PWML.1	P15.3
Digital pin 1 (TX0)	PWML.2	P15.2
Digital pin 2 (PWM)	PWML.3	P2.0
Digital pin 3 (PWM)	PWML.4	P2.1
Digital pin 5 (PWM)	PWML.6	P2.3
Digital pin 6 (PWM)	PWML.7	P2.4
Digital pin 7 (PWM)	PWML.8	P2.5



부저 회로도

- **Easy Shield Module의 부저 회로도 및 동작 원리**
  - D5(P2.3) 핀을 High로 출력하면, 트랜지스터에 의해 부저를 울리기에 충분한 전류가 부저로 흐름



목 차

부저 동작을 위한 레지스터 설정

- 프로젝트 소스코드 작성
  - 출력 핀(P2.3) 설정 ➔ P02_IOCR0 레지스터

Table 14-5 PCx Coding			
PCx[4:0]	**I/O**	**Characteristics**	**Selected Pull-up / Pull-down / Selected Output Function**
$0XX00_B$	Input	–	No input pull device connected, tri-state mode
$0XX01_B$			Input pull-down device connected
$0XX10_B$			Input pull-up device connected[1)]
$0XX11_B$			No input pull device connected, tri-state mode
10000_B	Output	Push-pull	General-purpose output
10001_B			Alternate output function 1
10010_B			Alternate output function 2
10011_B			Alternate output function 3
10100_B			Alternate output function 4
10101_B			Alternate output function 5
10110_B			Alternate output function 6

```
#include "main.h"

int core0_main(void) {

   /* Set P02.3(Buzzer) as push-pull output */
   MODULE_P02.IOCR0.B.PC3 = 0b10000;

   return 0;
}
```

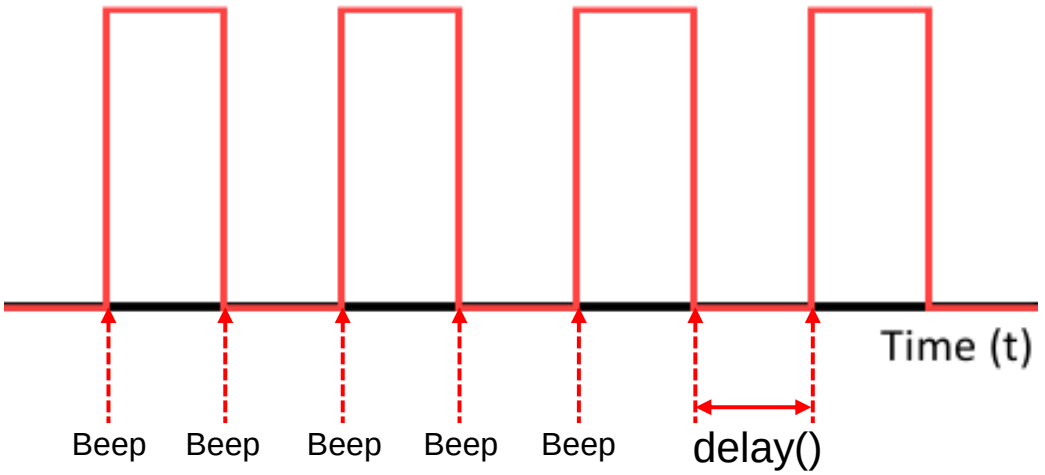
포트02 모듈의 3번 핀을 출력으로 설정하기 위해
P02_IOCR0 레지스터의 PC3 필드에 0b10000 값을 씀

부저 동작 예제

- 주파수
  - 130Hz=1/130=0.0076923…second=약 7,692us
  - ex) 7692 마이크로초 간격으로 부저 HIGH, LOW 신호를 출력하면 3옥타브 도 음계 출력됨

옥타브 / 음계	1	2	3	
C(도)	32.7032	65.4064	130.8128	2
C#	34.6478	69.2957	138.5913	2
D(레)	36.7081	73.4162	146.8324	2
D#	38.8909	77.7817	155.5635	3
E(미)	41.2034	82.4069	164.8138	3
F(파)	43.6535	87.3071	174.6141	3
F#	46.2493	92.4986	184.9972	3

※ P2.3 의 전압 레벨



부저 동작 예제

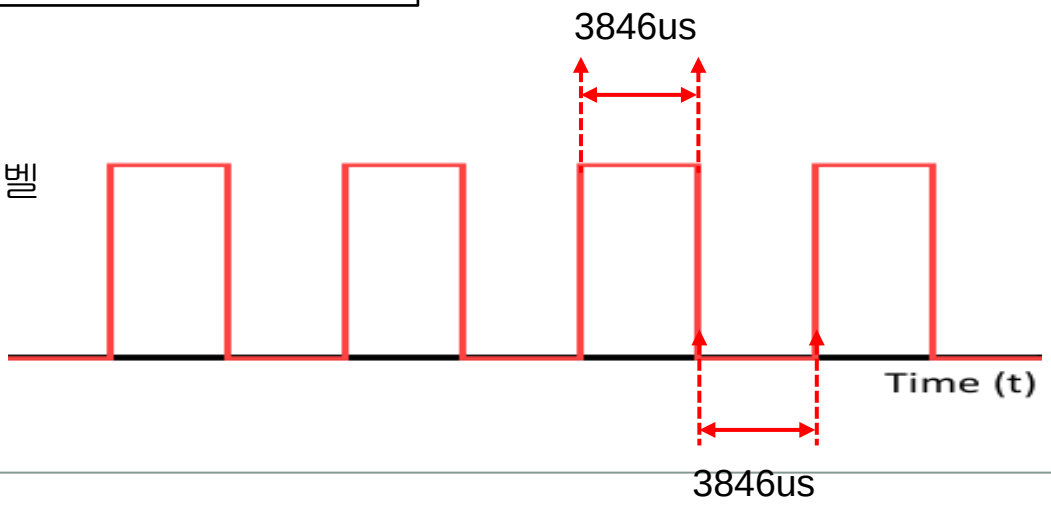
- 부저 동작(3옥타브 도) 예시
  - 1 cycle=7692us가 되어야 하므로 상승, 하강 엣지는 각각 130hz의 절반이어야 함
  - Beep() 함수는 Buzzer.c 파일에 개발되어 있음

```
void Beep(unsigned int hz) {
   volatile int loop = 1000000 / hz / 2;
   for (int i = 0; i < loop; i++)
      for (int j = 0; j < 1; j++)
         for (int k = 0; k < 1; k++)
            _nop();
}
```

입력 받은 hz 값에 따른 지연 가능한 loop 값 계산
- Duty 비는 항상 50% 이고 신호의 길이를 변경
- 아래와 같이 50% 듀티 신호가 연속적으로 입력되어야 부저 소리 발생함

Hz = 130 인 경우, 약3846us delay

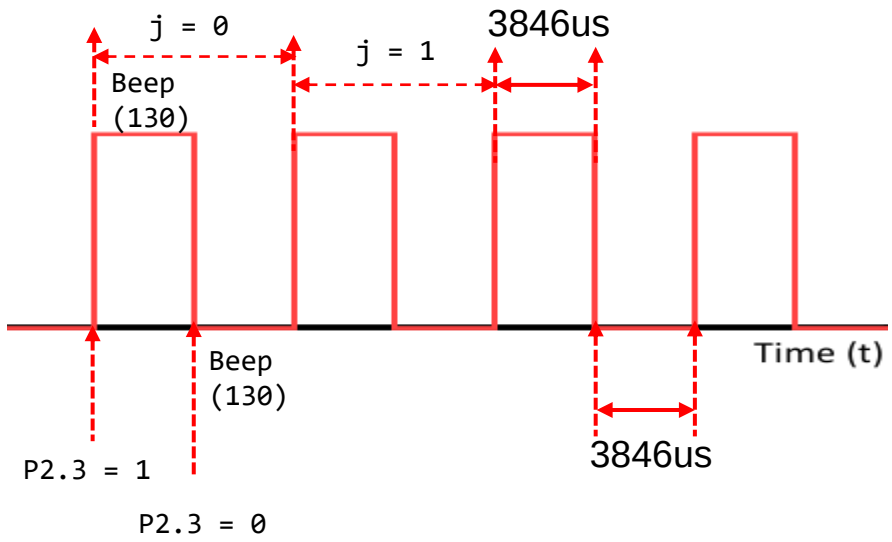
※ P2.3 의 전압 레벨



부저 동작 예제

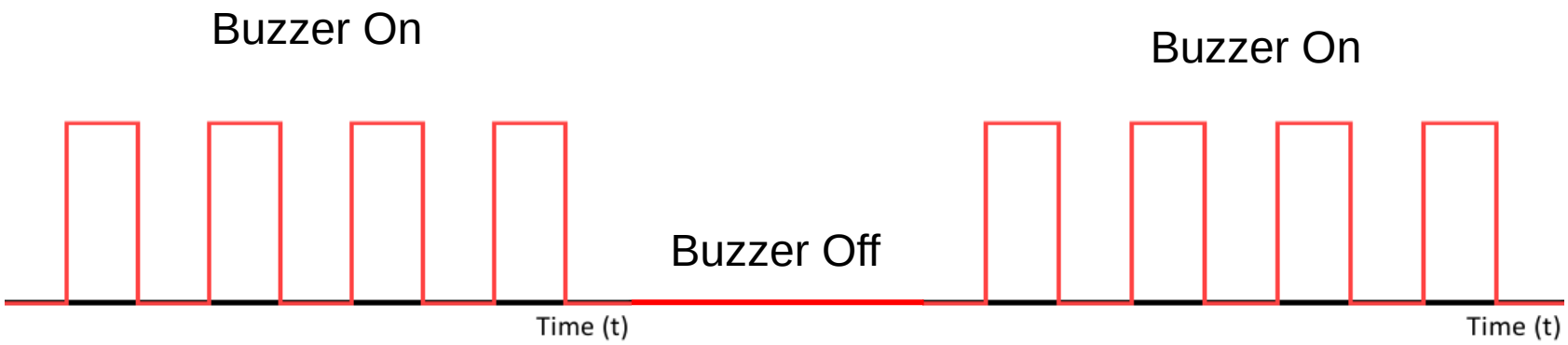
- 부저 동작(3옥타브 도) 예시
  - 1 cycle=7692us가 되어야 하므로 상승, 하강 엣지는 각각 130hz의 절반이어야 함

```
#define Buzzer    &MODULE_P02,3

#include "main.h"

int core0_main(void) {

    IfxPort_setPinModeOutput(Buzzer,
      IfxPort_OutputMode_pushPull,
      IfxPort_OutputIdx_general);

  volatile unsigned int j = 0;
  while (j++ < 1000) {
    IfxPort_setPinHigh(Buzzer);
    Beep(130);
    IfxPort_setPinLow(Buzzer);
    Beep(130);
  }

  return 0;
}
```



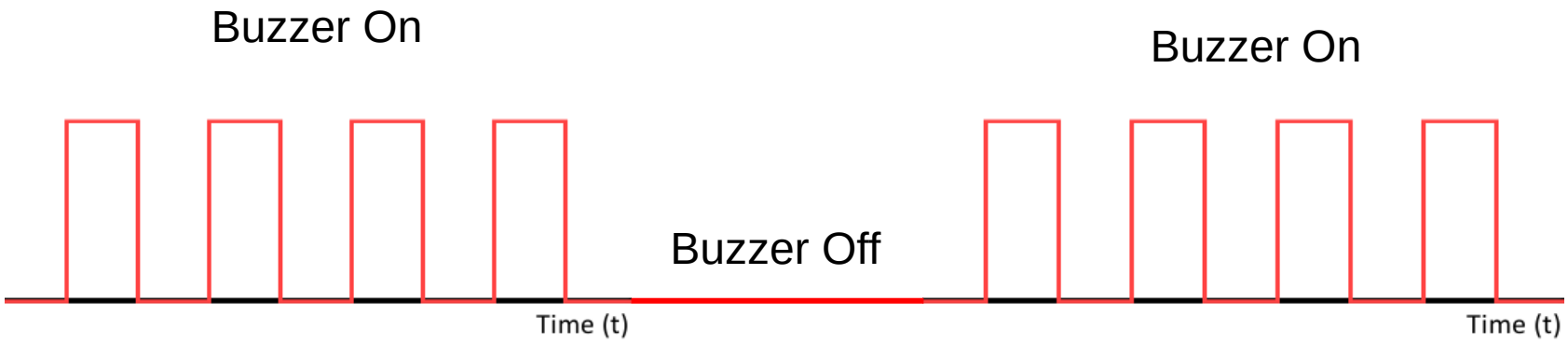
부저 실습

- 부저를 '삑삑' 소리가 나도록 제어 하라
  - Buzzer On → Buzzer Off → Buzzer On → Buzzer Off 반복



폴링 방식의 부저 구동 문제점

- **Beep() 함수는 CPU가 nop()를 실행하여 다른 일을 동시에 수행할 수 없음**
  - 다양한 기능을 동시에 구동하기 위해서는 main() 함수와 별도로 동작하여야 함




```
void Beep(unsigned int hz) {
   volatile int loop = 1000000 / hz / 2;
   for (int i = 0; i < loop; i++)
      for (int j = 0; j < 1; j++)
         for (int k = 0; k < 1; k++)
            _nop();
}
```

CPU 사용률 : 100%

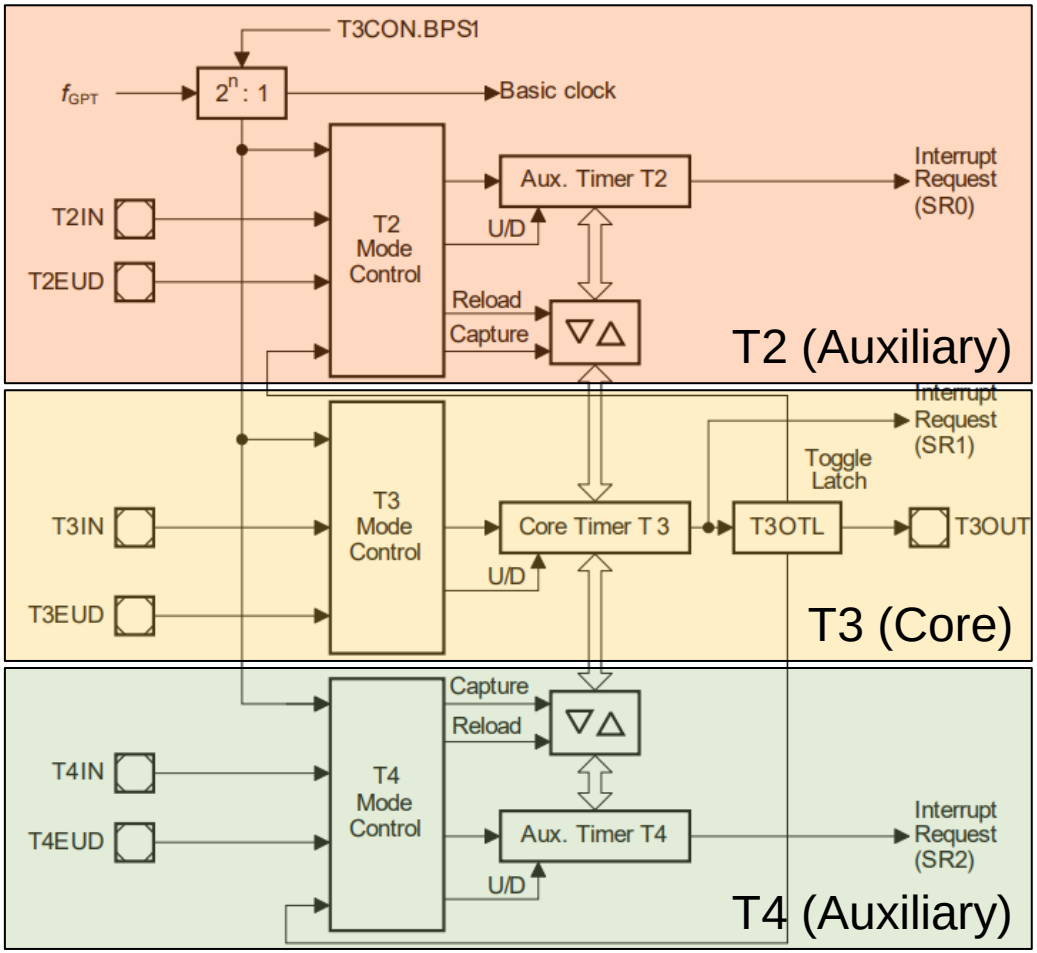
목 차

- 초음파 센서 제어
- 수동 부저 제어
  - 수동 부저 개요
  - 폴링 기반 수동 부저 구동
  - 타이머 인터럽트 기반 수동 부저 구동
- 후방주차센서 실습

GPT1 블록 다이어그램

GPT1 블록 다이어그램

- 1개 코어 타이머(T3), 2개 보조 타이머(T2, T4)로 구성



부저 Reload 타이머

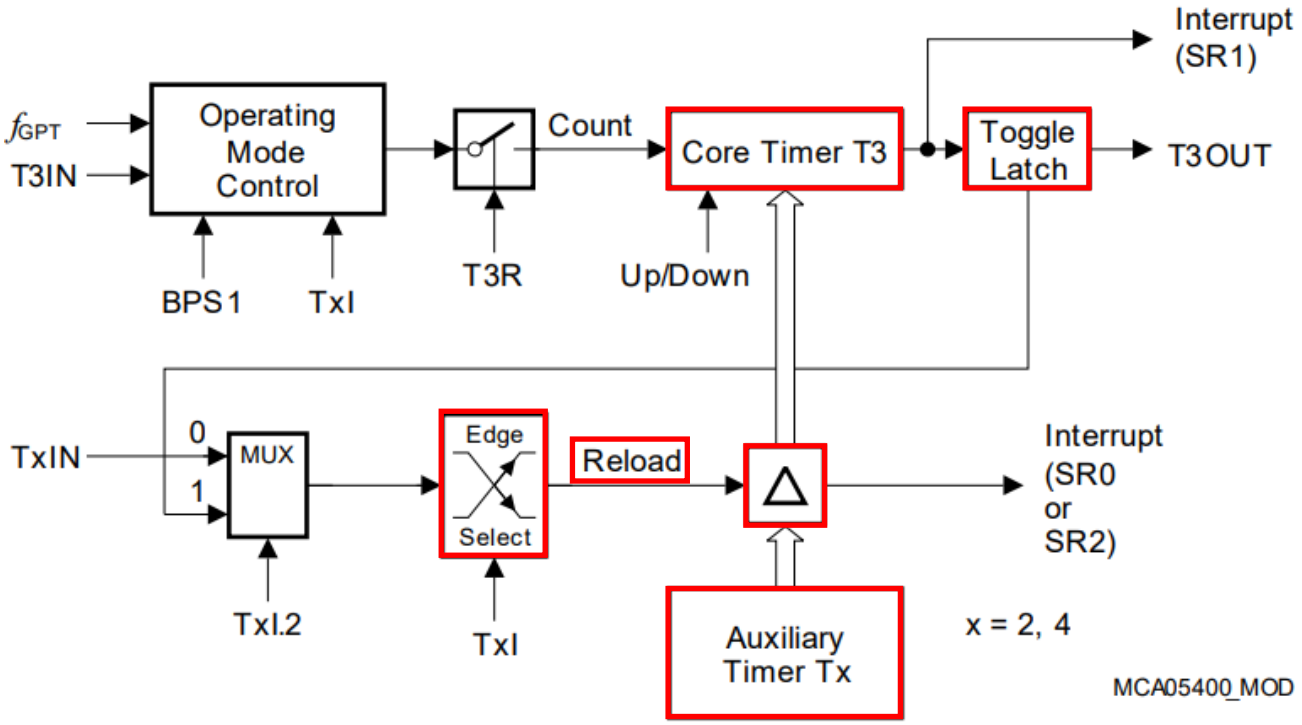
부저 main 타이머

```
T3 = T3 -1
If(T3 <0) {
   ISR실행
   T3 = T2
}
```

초음파 센서에서 사용

Reload Mode

- **Reload 모드**
  - 코어 타이머 T3가 overflow or underflow가 발생할 때,
  - Toggle Latch에서 신호를 발생시켜 보조 타이머 T2에 전달하고,
  - T2는 T3 신호의 Rising/Falling Edge를 감지하여
  - T2가 가지고 있던 Timer 값을 T3에 reload
    - T3에 Uptime/Downtime을 번갈아가며 reload→듀티비 제어



GPT1 코어 타이머 설정

- **GPT1의 T3 코어 타이머 모드 frequency 설정 방법**
  - $f_{GPT} = f_{SPB} = 100MHz$ (기본 시스템 클럭)
  - f_{GPT} 를 BPS1, T3I Bit field를 참고하여 Prescale하여 f_{T3} 를 계산
  - ex) T3I = 101_B, BPS1 = 10_B 로 설정했을 때,

    → 100MHz / 1024 = 97.65625 KHz (T_{tick} = 10.24 us)

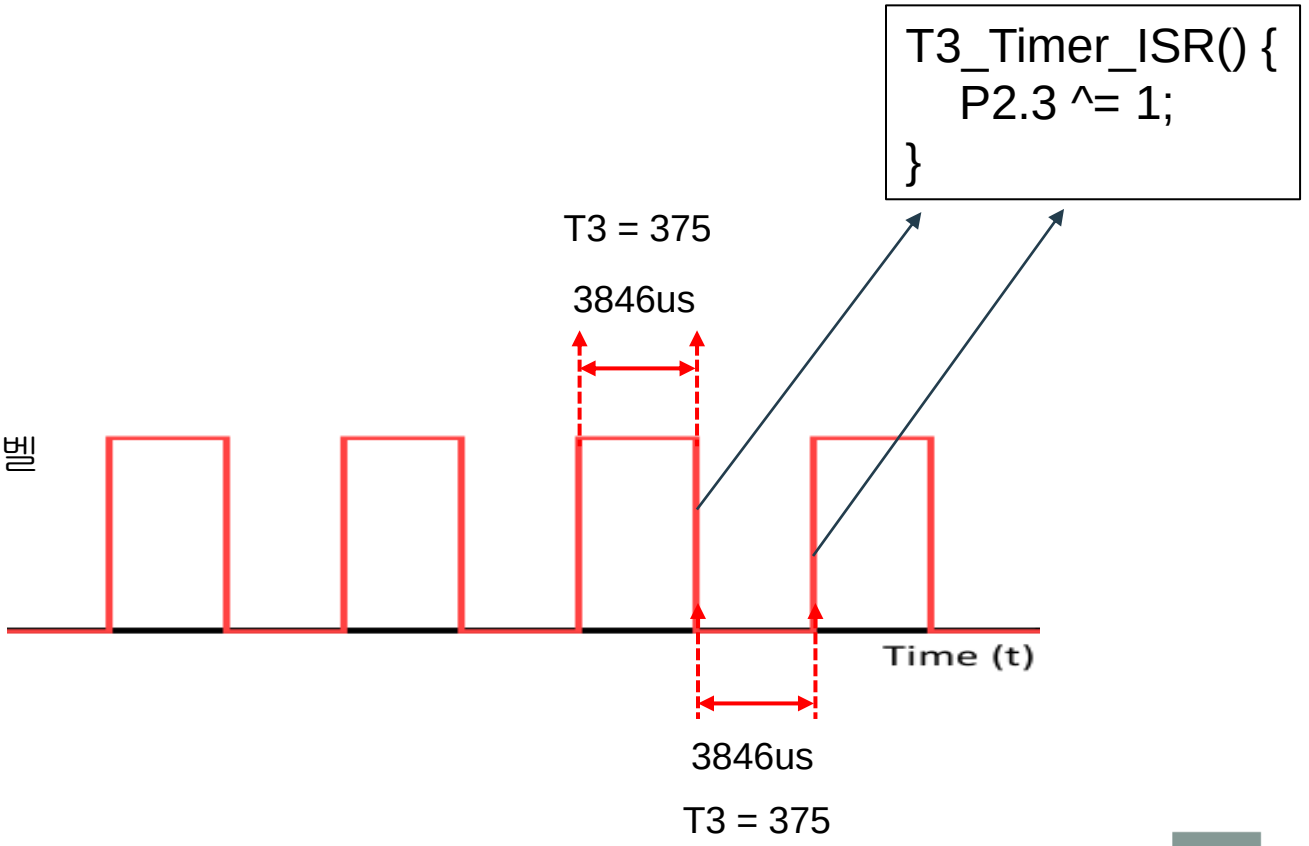
Table 27-7 GPT1 Overall Prescaler Factors for Internal Count Clock (Timer Mode and Gated Timer Mode)

Individual Prescaler for Tx	Common Prescaler for Module Clock[1)]			
	BPS1 = 01_B	BPS1 = 00_B	BPS1 = 11_B	BPS1 = 10_B
TxI = 000_B	4	8	16	32
TxI = 001_B	8	16	32	64
TxI = 010_B	16	32	64	128
TxI = 011_B	32	64	128	256
TxI = 100_B	64	128	256	512
TxI = 101_B	128	256	512	1024
TxI = 110_B	256	512	1024	2048
TxI = 111_B	512	1024	2048	4096

타이머 기반 부저 동작 방식

- 주파수
  - 130Hz=1/130=0.0076923…second=약 7,692us
  - 1 tick = 10.24 us 이므로, 3846 / 10.24 = 375 로 설정하면 130 Hz

※ P2.3 의 전압 레벨



타이머 초기화 및 인터럽트 처리 함수

- **gpt12.c 파일에 이미 포함되어 있음**
  - 부저와 무관한 T4 타이머 설정은 생략함

```
IFX_INTERRUPT(IsrGpt120T3Handler_Beep, 0, ISR_PRIORITY_GPT1T3_TIMER);   // 인터럽트 등록
void IsrGpt120T3Handler_Beep()
{
    ... 생략 ... (이후 페이지 참조)
}

void init_gpt1(void) {
   /* Initialize the GPT12 module */
    IfxGpt12_enableModule(&MODULE_GPT120);
   /* Initialize the Timer T3 (PWM) */
    IfxGpt12_setGpt1BlockPrescaler(&MODULE_GPT120, IfxGpt12_Gpt1BlockPrescaler_32);
    IfxGpt12_T3_setMode(&MODULE_GPT120, IfxGpt12_Mode_timer);
    IfxGpt12_T3_setTimerDirection(&MODULE_GPT120, IfxGpt12_TimerDirection_down);
    IfxGpt12_T3_setTimerPrescaler(&MODULE_GPT120, IfxGpt12_TimerInputPrescaler_32);
   /* Timer T2: reloads the value DutyDownTime in timer T3 */
    IfxGpt12_T3_setTimerValue(&MODULE_GPT120, 100);
    IfxGpt12_T2_setMode(&MODULE_GPT120, IfxGpt12_Mode_reload);
    IfxGpt12_T2_setReloadInputMode(&MODULE_GPT120, IfxGpt12_ReloadInputMode_bothEdgesTxOTL);
    IfxGpt12_T2_setTimerValue(&MODULE_GPT120, 100);
   /* Initialize the interrupt */
    volatile Ifx_SRC_SRCR *src = IfxGpt12_T3_getSrc(&MODULE_GPT120)
    IfxSrc_init(src, ISR_PROVIDER_GPT12_TIMER, ISR_PRIORITY_GPT1T3_TIMER);
    IfxSrc_enable(src);

    ... T4 타이머 설정 생략 ...
}
```

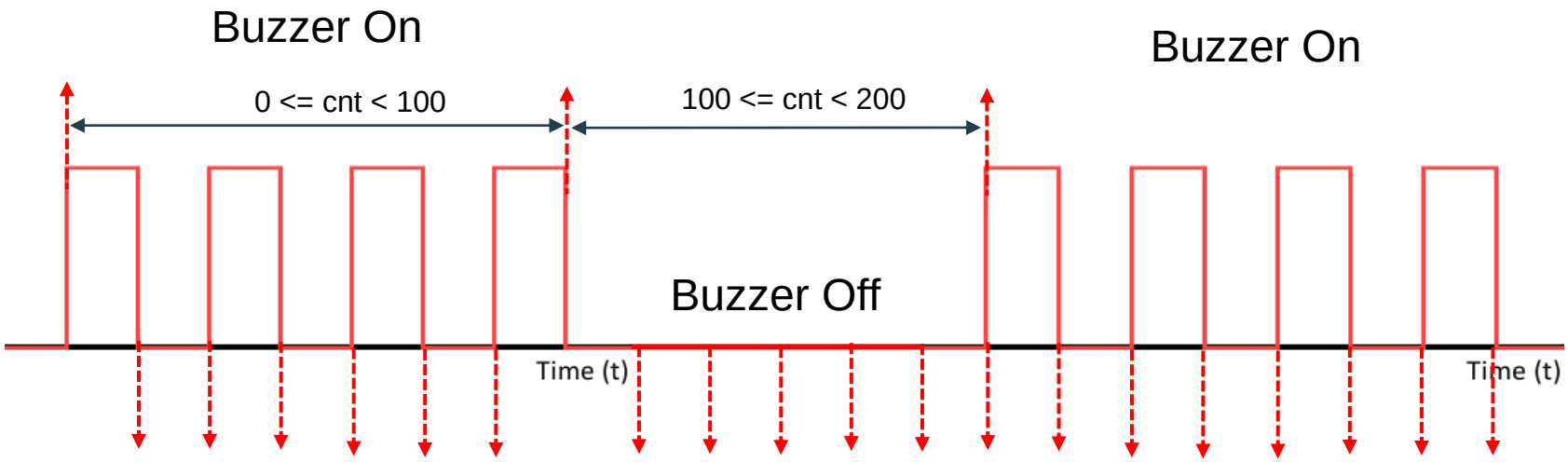
타이머 기반 부저 구동 예제

- 타이머 모듈에 의해 **130 Hz** 신호가 부저로 입력됨
  - 삑~~ 부저음 출력

```
#include "main.h"
#include "Buzzer.h"

int core0_main(void) {
   …중략…
  Init_Buzzer();

  while (1) {
     setBeepCycle(130);
  }

  return 0;
}
```

타이머 인터럽트를 활용한 부저 구동

- 다양한 기능을 동시에 구동하기 위해 인터럽트 활용
  - 아래와 같이 인터럽트에서 출력 신호를 조정하여 부저음 제어 가능



타이머 인터럽트 발생

```
T3_Timer_ISR() {
    P2.3 ^= 1;
}
```

```
T3_Timer_ISR() {
    P2.3 = 0;
}
```

```
T3_Timer_ISR() {
    P2.3 ^= 1;
}
```

타이머 기반 부저 구동 예제

- “삑삑” 소리 나도록 구동

```
int beepCnt = 0;
int beepOnOff = 0;

void IsrGpt120T3Handler_Beep() {
    if ((beepCnt < beepOnOff) || (beepOnOff == 1)) {
        IfxPort_togglePin(Buzzer);
    } else if (beepCnt < beepOnOff * 2) {
        IfxPort_setPinLow(Buzzer);
    } else {
        beepCnt = 0;
    }
    beepCnt++;
}
```

부저 모듈

- **Buzzer 모듈 제공 함수**
  - `void Init_Buzzer(void);`
  - `void setBeepCycle(int cycle);`
  - `void Beep(unsigned int hz);`

```
#define Buzzer        &MODULE_P02,3

void Init_Buzzer()
{
  IfxPort_setPinModeOutput(Buzzer, IfxPort_OutputMode_pushPull, IfxPort_OutputIdx_general);
  init_gpt1();
  runGpt12_T3();
}

void setBeepCycle(int cycle)
{
  beepOnOff = cycle;
}

void Beep(unsigned int hz)
{
  volatile int loop = 1000000 / hz / 2;          /* 주파수에 따른 us 계산 */
  for (volatile int i = 0; i < loop; i++)        /* loop * 1us 동안 delay */
    for (volatile int j = 0; j < 1; j++)
      for (volatile int k = 0; k < 1; k++)
        __nop();
}
```

타이머 기반 부저 구동 예제

- 시간에 따라 경고음 속도를 빠르게 하는 프로그램을 작성하라

```
#include "main.h"

int core0_main(void) {
  …중략
  Init_Buzzer();

  while (1) {
    setBeepCycle(260);  // on/off every 1s (3846us * 260)
    delay_ms(5000);
    setBeepCycle(50);
    delay_ms(3000);
    setBeepCycle(10);
    delay_ms(3000);
    setBeepCycle(1);         // always on
    delay_ms(3000);
    setBeepCycle(0);         // always off
    delay_ms(3000);
  }

  return 0;
}
```

후방 주차 경고음 실습

- 초음파 센서의 값을 받아, 장애물과의 거리가 가까울수록 빠르게 경고음을 울리는 프로그램을 작성하라.

목 차

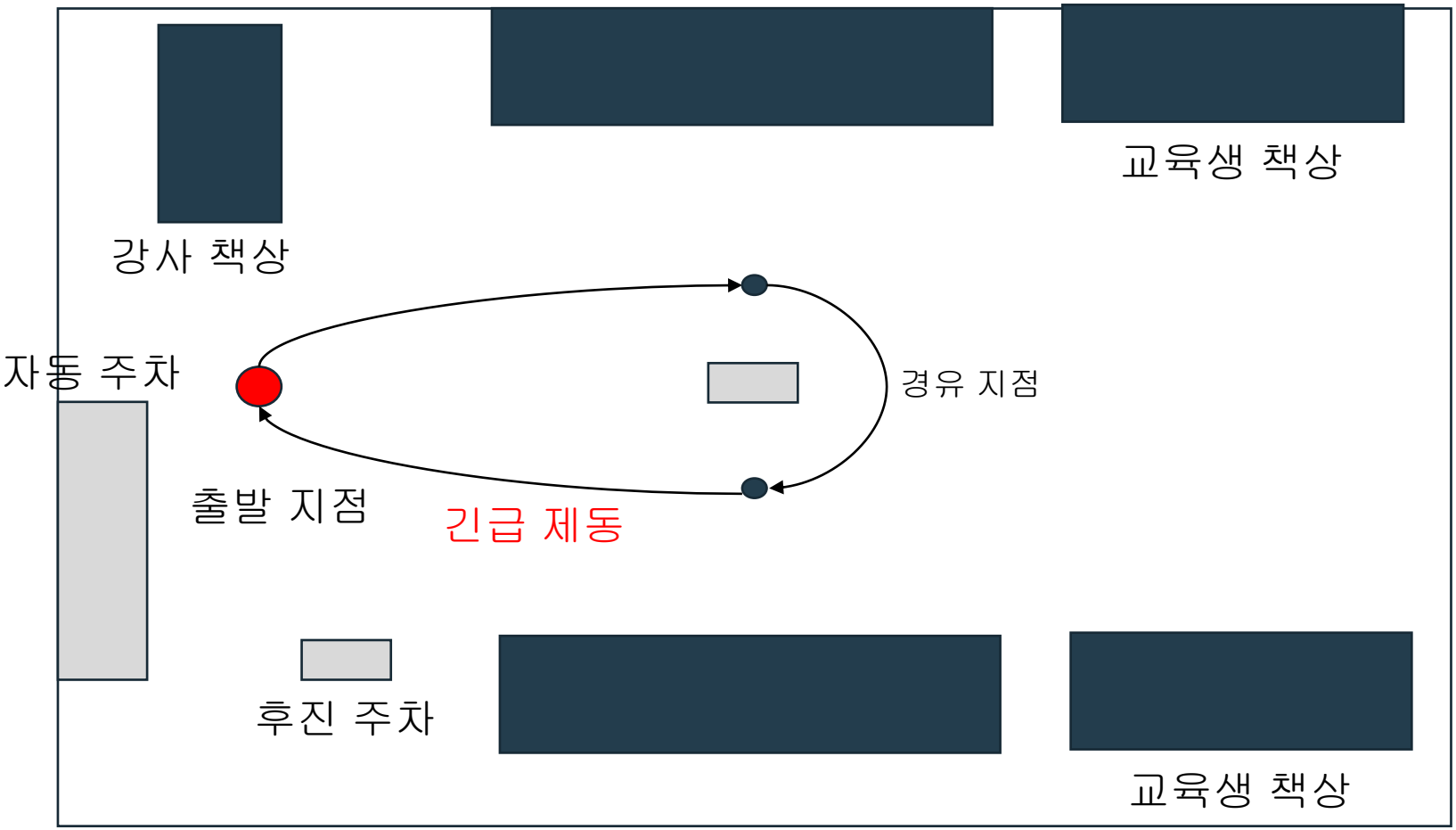
- 초음파 센서 제어
- 수동 부저 제어
- 후방주차센서 실습

팀 프로젝트 개요

- **프로젝트 SW 요구 사항**
  - 1) 시리얼 통신으로 RC 카를 제어함
  - 2) 후진 중에 뒤에 장애물이 있는 경우 거리가 가까울 수록 빠르게 경고음을 출력함
  - 3) 일정 거리 이하인 경우 시리얼 통신의 후진 제어 명령을 무시하고 차량을 멈춤
  - 4) 자동으로 멈춘 경우 후진이 아닌 경우 움직임 수 있음

- **프로젝트 HW 요구 사항**
  - 후방 주차 센서는 초음파 센서를 사용할 것

RC카 기능 통합 및 데모



감사합니다