Microsoft

Azure Sphere

Episode 2 : 하드웨어 및 이벤트 프로그래밍 이해
Azure Sphere 설정하기

윤기석
마이크로소프트

GPIO? PERIPHERAL(주변장치)?

- GPIO (General-Purpose Input/Output)

  주로 LED / 릴레이 등의 상태 제어 혹은 버튼 등의 입력

- PWM (Pulse-Width Modulation)

  디지털 신호를 출력하되 펄스폭을 조절하여 아날로그 신호와 같은 효과

  LED 밝기 혹은 모터 속도 조절 등에 주로 쓰임

- I2C (Inter-integrated Circuit)

  주로 각종 센서 및 제어 용도의 저속 주변기기를 연결하는데 사용 / 예) 휴대폰/스마트워치의 센서

- SPI (Serial Peripheral Interface)

  동시 양방향 통신 지원 / IC2 보다 고속의 센서 등 연결 / 예) 드론 MEMS 센서 퓨전 시

- ADC (Analog-to-Digital Converter)

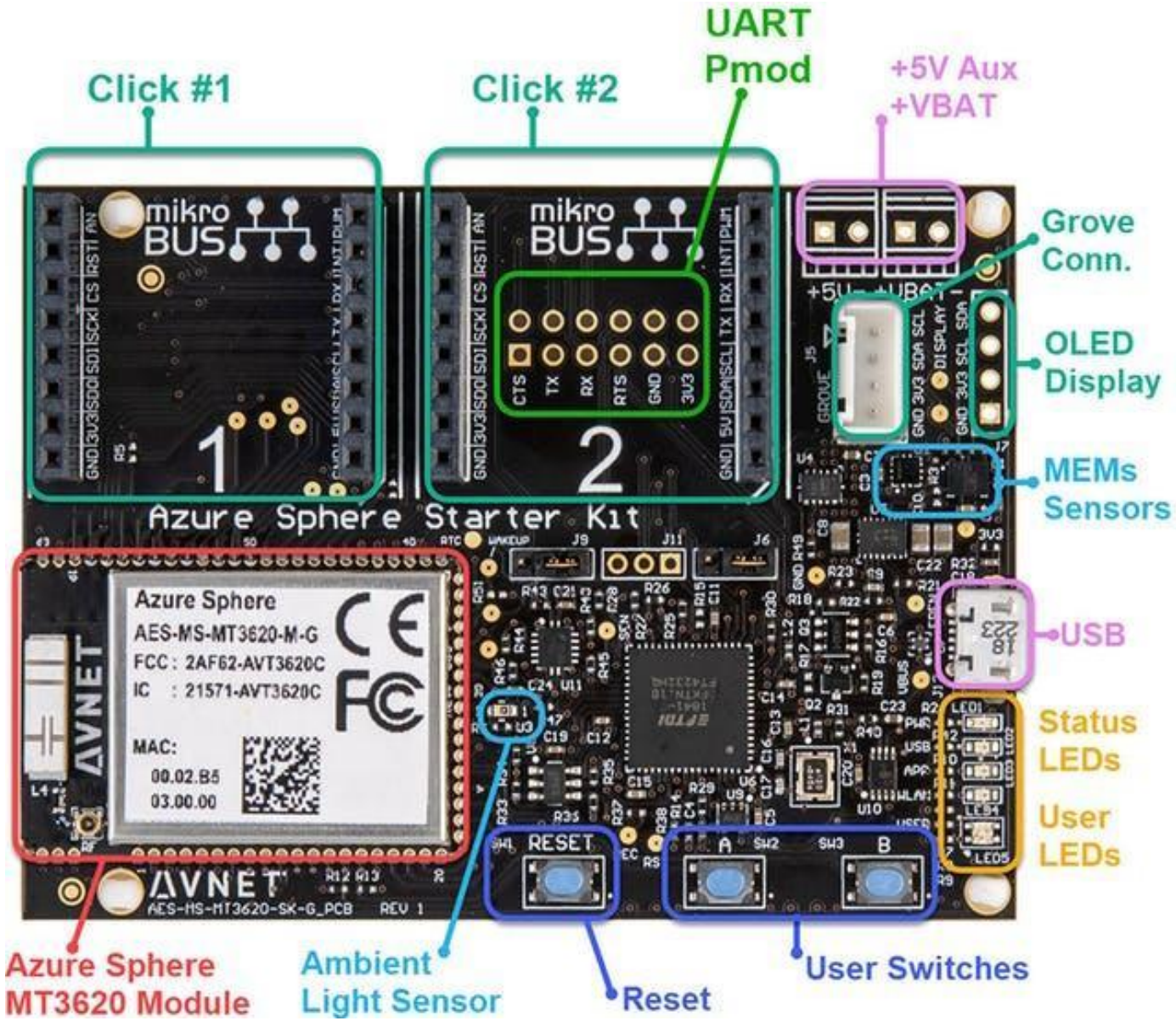
  아날로그 전기 신호를 디지털 신호로 변환 / 아날로그 센서 입력, 전압 검지 등에 사용

- UART (Universal Asynchronous Receiver/Transmitter)

  범용 비동기 통신으로 RS-232/RS422/RS-485 등의 통신 표준과 함께 사용

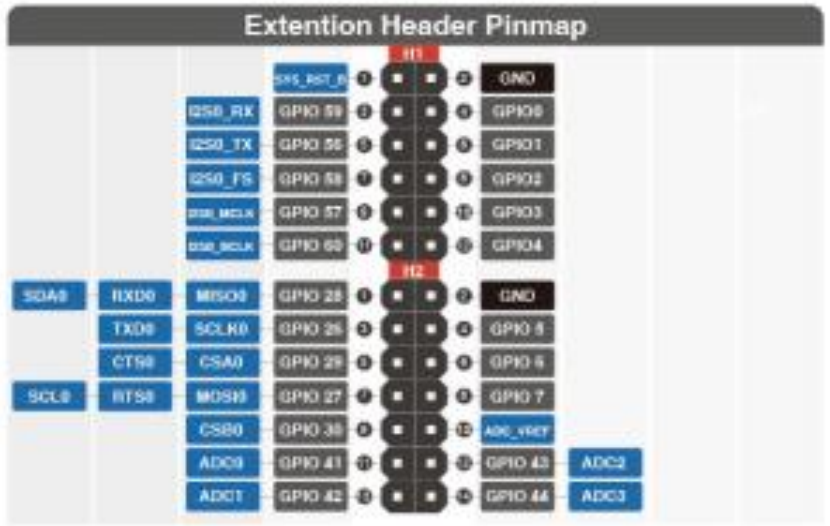
  MCU <-> 모듈 인터페이스 혹은 디버그 용도로도 사용

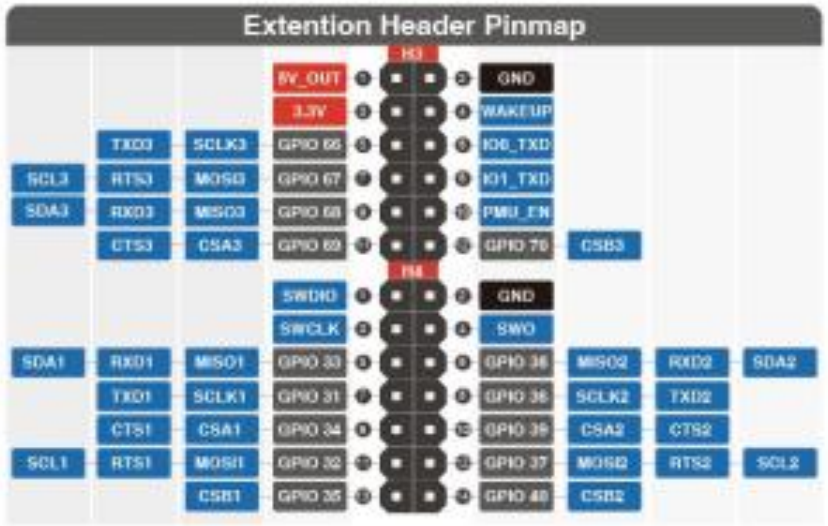
개발 보드 – Avnet Azure Sphere MT3620 Starter Kit



[User Guide 문서]

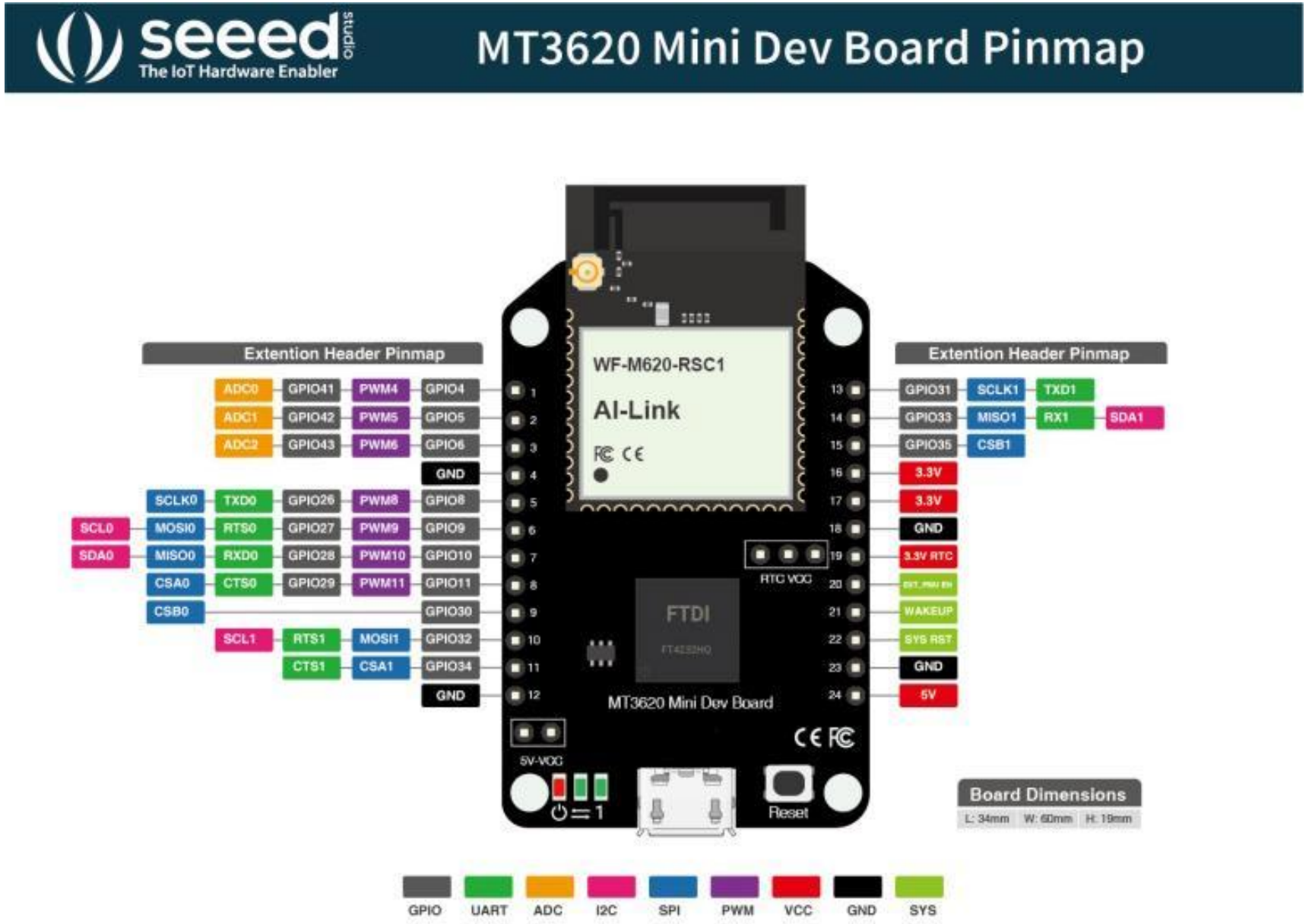
개발 보드 – Seeed Studio Azure Sphere MT3620 Development Kit





[User Guide 문서]

개발 보드 – Seeed Studio Azure Sphere MT3620 Mini DK



[User Guide 문서]

Azure Sphere 실습 자료

- Azure Sphere 실습 자료 Repository

1. https://aka.ms/azure-sphere-learn 리포지토리를 로컬 디스크에 저장하여 사용

   - 가장 짧은 경로 예) C:\lab 에 저장 (길면 CMake 빌드 시 에러 가능성)

2. https://aka.ms/azure-sphere-samples 을 기반으로 모범 사례를 위해 내용 추가

   - 추가된 C 함수에는 lp_ 접두사 / typedef 와 enum 에는 LP_ 접두사

GPIO 설정 및 사용

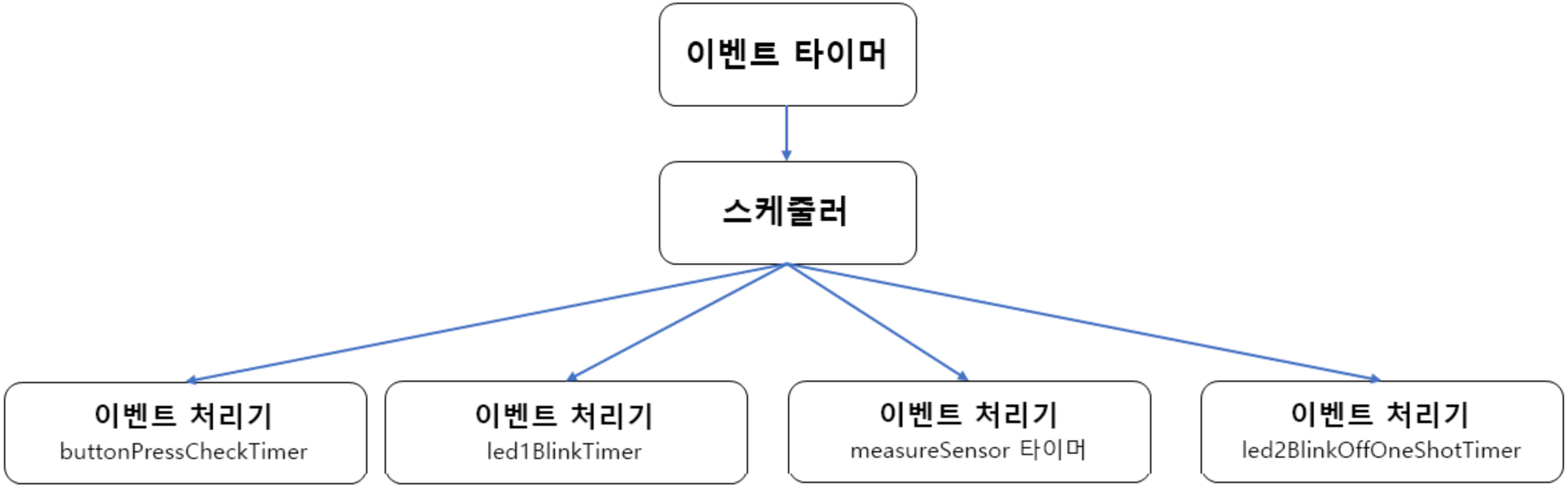
- GPIO LED 출력

```
static LP_GPIO alertLed = {
    .pin = ALERT_LED,                   // The GPIO pin number
    .direction = LP_OUTPUT,             // for OUTPUT
    .initialState = GPIO_Value_Low,     // Set the initial state on the pin when opened
    .invertPin = true,                  // Should the switching logic be reverse for on/off, high/low
    .name = "alertLed" };               // An arbitrary name for the peripheral
```

- GPIO 버튼 입력

```
static LP_GPIO buttonA = {
    .pin = BUTTON_A,
    .direction = LP_INPUT,
    .name = "buttonA" };
```

이벤트 기반 프로그래밍


이벤트 타이머
스케줄러
이벤트 처리기
buttonPressCheckTimer
이벤트 처리기
led1BlinkTimer
이벤트 처리기
measureSensor 타이머
이벤트 처리기
led2BlinkOffOneShotTimer

Periodic Timers(주기적 타이머)

LP_TIMER 형식의 measureSensorTimer

10 초 주기로 초기화 / 이벤트 타이머 트리거 되면 MeasureSensorHandler 호출

```
static LP_TIMER measureSensorTimer = {
    // Fire the timer event every 10 seconds + zero nanoseconds.
    .period = { 10, 0 },
    // An arbitrary name for the timer, used for error handling
    .name = "measureSensorTimer",
    // The function handler called when the timer triggers.
    .handler = MeasureSensorHandler
};
```

이벤트 주기를 0.5 (500ms) 로 할 때는 .period = { 0, 500000000 } 으로 설정

{ s(초) , ns(나노초) }

MeasureSensorHandler 함수

```
/// <summary>
/// Read sensor and send to Azure IoT
/// </summary>
static void MeasureSensorHandler(EventLoopTimer* eventLoopTimer)
{
    static int msgId = 0;
    static LP_ENVIRONMENT environment;

    if (ConsumeEventLoopTimerEvent(eventLoopTimer) != 0)
    {
        lp_terminate(ExitCode_ConsumeEventLoopTimeEvent);
    }
    else {
        if (lp_readTelemetry(&environment) &&
            snprintf(msgBuffer, JSON_MESSAGE_BYTES, msgTemplate,
                environment.temperature, environment.humidity, environment.pressure, msgId++) > 0)
        {
            Log_Debug(msgBuffer);
        }
    }
}
```

센서 값 읽고 **JSON** 으로 **Serialize** 해서 디버그 콘솔에 표시

One-shot Timer(일회성 타이머)

LP_TIMER 형식의 alertLedOffOneShotTimer

0 초 주기로 초기화 되면 one-shot timer

```
static LP_TIMER alertLedOffOneShotTimer = {
    .period = { 0, 0 },
    .name = "alertLedOffOneShotTimer",
    .handler = AlertLedOffToggleHandler };
```

예제)

ButtonPressCheckHandler -> LED_ON -> 1초 one-shot timer set -> 1초 뒤 handler 호출 -> LED_OFF

ButtonPressCheckHandler 버튼 입력 확인

```
/// <summary>
/// Handler to check for Button Presses
/// </summary>
static void ButtonPressCheckHandler(EventLoopTimer* eventLoopTimer)
{
    static GPIO_Value_Type buttonAState;

    if (ConsumeEventLoopTimerEvent(eventLoopTimer) != 0) {
        lp_terminate(ExitCode_ConsumeEventLoopTimeEvent);
    }
    else {
        if (lp_gpioStateGet(&buttonA, &buttonAState))
        {
            lp_gpioOn(&alertLed);
            lp_timerOneShotSet(&alertLedOffOneShotTimer, &(struct timespec){1, 0});
        }
    }
}
```

1초 one-shot timer -> AlertLedOffToggleHandler 가 LED 를 OFF

```
/// <summary>
/// One shot timer handler to turn off Alert LED
/// </summary>
static void AlertLedOffToggleHandler(EventLoopTimer* eventLoopTimer) {
    if (ConsumeEventLoopTimerEvent(eventLoopTimer) != 0) {
        lp_terminate(ExitCode_ConsumeEventLoopTimeEvent);
    }
    else {
        lp_gpioOff(&alertLed);
    }
}
```

간단하게 확장

이 모델을 사용하면 간단하게 주변장치 / 타이머 등 추가해서 확장 가능

```
static LP_GPIO fanControl = {
    .pin = FAN1,                        // The GPIO pin number
    .direction = LP_OUTPUT,             // for OUTPUT
    .initialState = GPIO_Value_Low,     // Set the initial state on the pin when opened
    .invertPin = true,                  // Should the switching logic be reverse for on/off, high/low
    .name = "FanControl"                // An arbitrary name for the senor.
};
```

```
LP_GPIO* gpioSet[] = { &buttonA, &networkConnectedLed, &alertLed, &fanControl };
```

Azure Sphere 설정하기

- 체크리스트
- 최신 Azure Sphere SDK 설치
- Visual Studio Code 혹은 Visual Studio 에 Azure Sphere 확장팩 설치
- CMake 및 Ninja 환경 설정 (Visual Studio Code의 경우)
- Azure Sphere 에 로그인
- 테넌트 추가 후 디바이스 클레임

    - 디바이스 클레임은 되돌릴 수 없는 1회성이므로 계정 및 테넌트 선택에 주의합니다.

- 디바이스 네트워크 설정

실습 Repository 및 Azure Sphere Sample 복사

실습 Repository 를 로컬 디스크에 복사

```
git clone --depth 1 https://github.com/MicrosoftDocs/Azure-Sphere-Developer-Learning-Path.git Azure-Sphere
```

```
PS C:\lab> git clone --depth 1 https://github.com/MicrosoftDocs/Azure-Sphere-Developer-Learning-Path.git Azure-Sphere
Cloning into 'Azure-Sphere'...
remote: Enumerating objects: 1151, done.
remote: Counting objects: 100% (1151/1151), done.
remote: Compressing objects: 100% (783/783), done.
remote: Total 1151 (delta 379), reused 1001 (delta 333), pack-reused 0
Receiving objects: 100% (1151/1151), 146.85 MiB | 17.82 MiB/s, done.
Resolving deltas: 100% (379/379), done.
Checking out files: 100% (1998/1998), done.
PS C:\lab>
```

Azure Sphere Sample 복사

```
git clone https://github.com/Azure/azure-sphere-samples.git
```

```
PS C:\lab> git clone https://github.com/Azure/azure-sphere-samples.git
Cloning into 'azure-sphere-samples'...
remote: Enumerating objects: 128, done.
remote: Counting objects: 100% (128/128), done.
remote: Compressing objects: 100% (111/111), done.
remote: Total 2598 (delta 26), reused 62 (delta 10), pack-reused 2470
Receiving objects: 100% (2598/2598), 98.69 MiB | 15.78 MiB/s, done.
Resolving deltas: 100% (1438/1438), done.
Checking out files: 100% (760/760), done.
PS C:\lab>
```

실습 Azure Sphere 보드를 개발 모드로 설정

Azure Sphere 개발자 명령 프롬프트 또는 linux 터미널에서 아래 명령을 실행

1) 디바이스의 기존 애플리케이션 지우기

```
PS C:\Users\kiyun> azsphere device sideload delete
Component '25025d2c-66da-4448-bae1-ac26fcdd3627' deleted.
PS C:\Users\kiyun>
```

2) Azure Sphere 보드를 재부팅

```
PS C:\Users\kiyun> azsphere device restart
Restarting device.
Device restarted successfully.
PS C:\Users\kiyun>
```

3) 보드를 HL_app 개발 모드로 설정 (Cortex-A7 core)

```
PS C:\Users\kiyun> azsphere device enable-development
Getting device group 'Development' for product 'DIWALI2020'.
warn: The device already has the 'Enable App development' capability. No changes will be applied to its existing capabilities.
Setting device group to 'Development' with ID '4b3f9c55-ffac-49ab-8019-[redacted]d3733fa32d'.
Successfully disabled application updates.
Installing debugging server to device..
Deploying 'C:\Program Files (x86)\Microsoft Azure Sphere SDK\DebugTools\gdbserver.imagepackage' to the attached device.
Image package 'C:\Program Files (x86)\Microsoft Azure Sphere SDK\DebugTools\gdbserver.imagepackage' has been deployed to the attached device
.
Previous device capabilities retained. Ensure that you have the correct development capabilities installed before continuing.
Successfully set up device for application development, and disabled application updates.
```

4) RT_app 개발모드(Cortex-M4) 는 관리자 권한으로 실행 후 azsphere device enable-development –r 실행

다음 에피소드

- EP1
  - Azure Sphere 아키텍처 / 개발환경
  - 가상의 보안 IoT 프로젝트 정의
- EP2
  - 하드웨어 및 이벤트 기반 프로그래밍 이해
  - Azure Sphere 설정 방법
- **EP3**
  - Azure IoT Central 에 실내 환경 센서를 연결
  - Azure Sphere 를 보호하는 방법
  - Azure Sphere 에 HL App(고급 애플리케이션) 배포
- EP4
  - Azure IoT 디바이스 쌍으로 실내 온도 설정
  - Azure IoT 직접 메서드로 Azure Sphere 원격 제어
- EP5
  - Azure RTOS 실시간 센서 앱 배포 / 실내 환경 모니터링
- EP6
  - Azure RTOS 실시간 실내 환경 센서 데이터를 IoT Central 에 전송
- EP7
  - 간단하게 OTA 업데이트 사용하기