

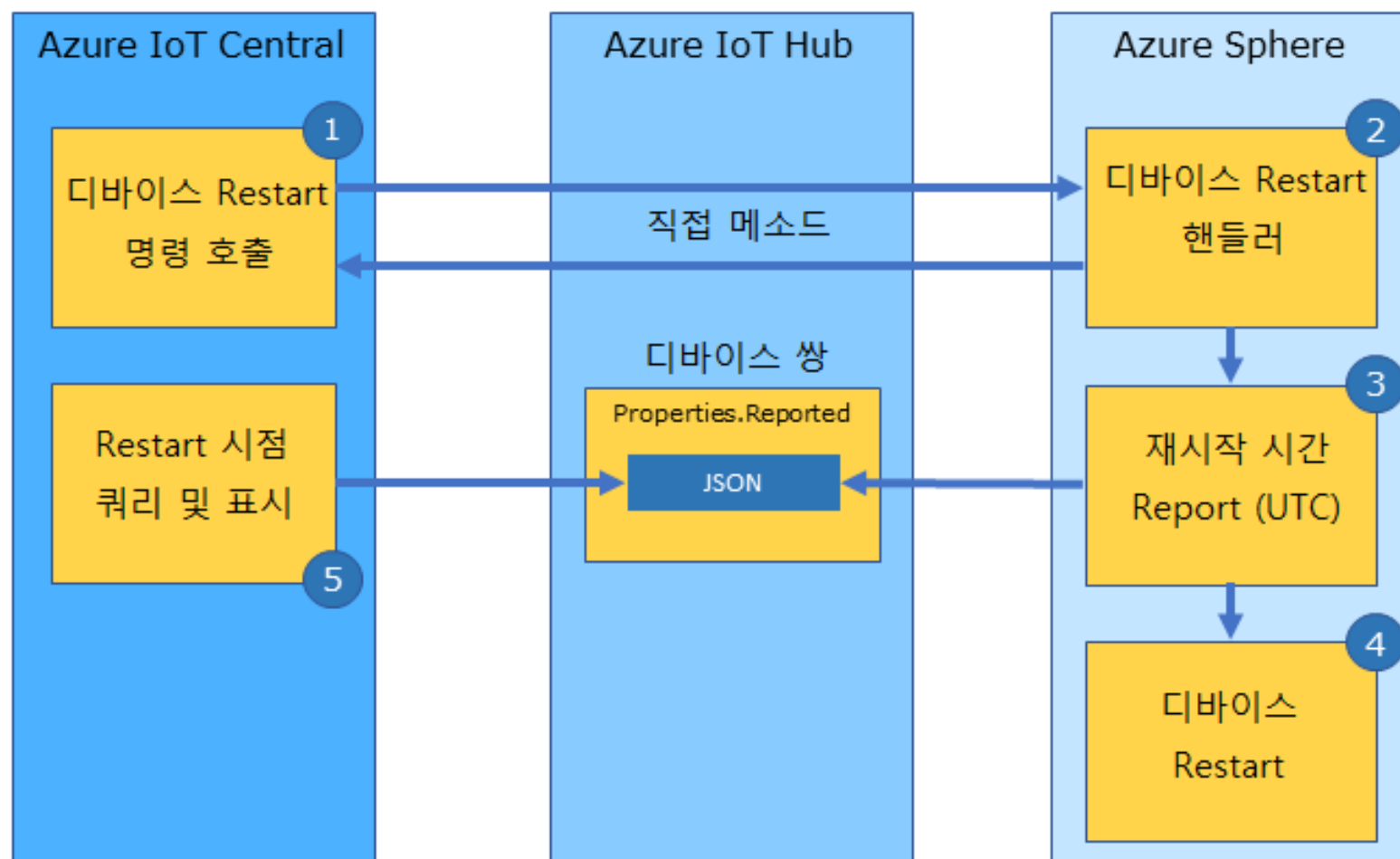


# Azure Sphere

## Episode 5 : Azure RTOS 실시간 센서 앱 배포 실내환경 모니터링

윤기석  
마이크로소프트

# 이제까지 한 것



# 새로운 요구사항

1. 기존보다 랩 온도, 습도, 대기압이 굉장히 안정적이어야 함
2. 추가로 필요한 센서가 타이밍이 아주 중요함
3. 실시간 코어에 연동해서 운영해야 함

# Building Blocks of Azure RTOS

## Seamless Turnkey Solution for Constraint Devices

### ThreadX

a high-performance real-time operating system kernel

### USBX

USB stack that provides host, device, and OTG support

### FileX

High performance embedded FAT file system  
(fault tolerance and flash memory wear leveling support)

### NetX Duo

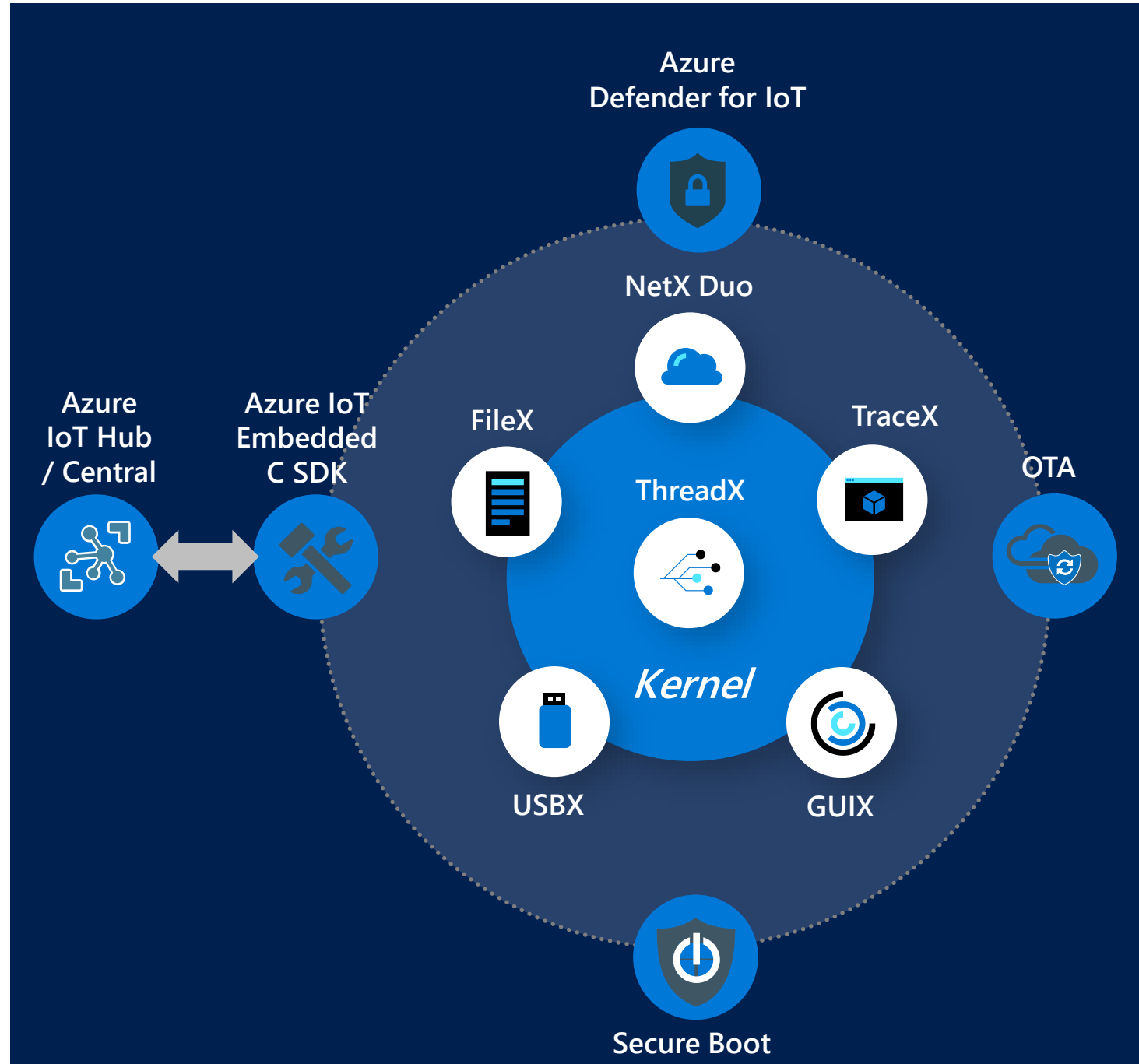
TCP/IP IPV4/IPv6 embedded network stack that supports IPSec, TLS / DTLS security protocols

### TraceX

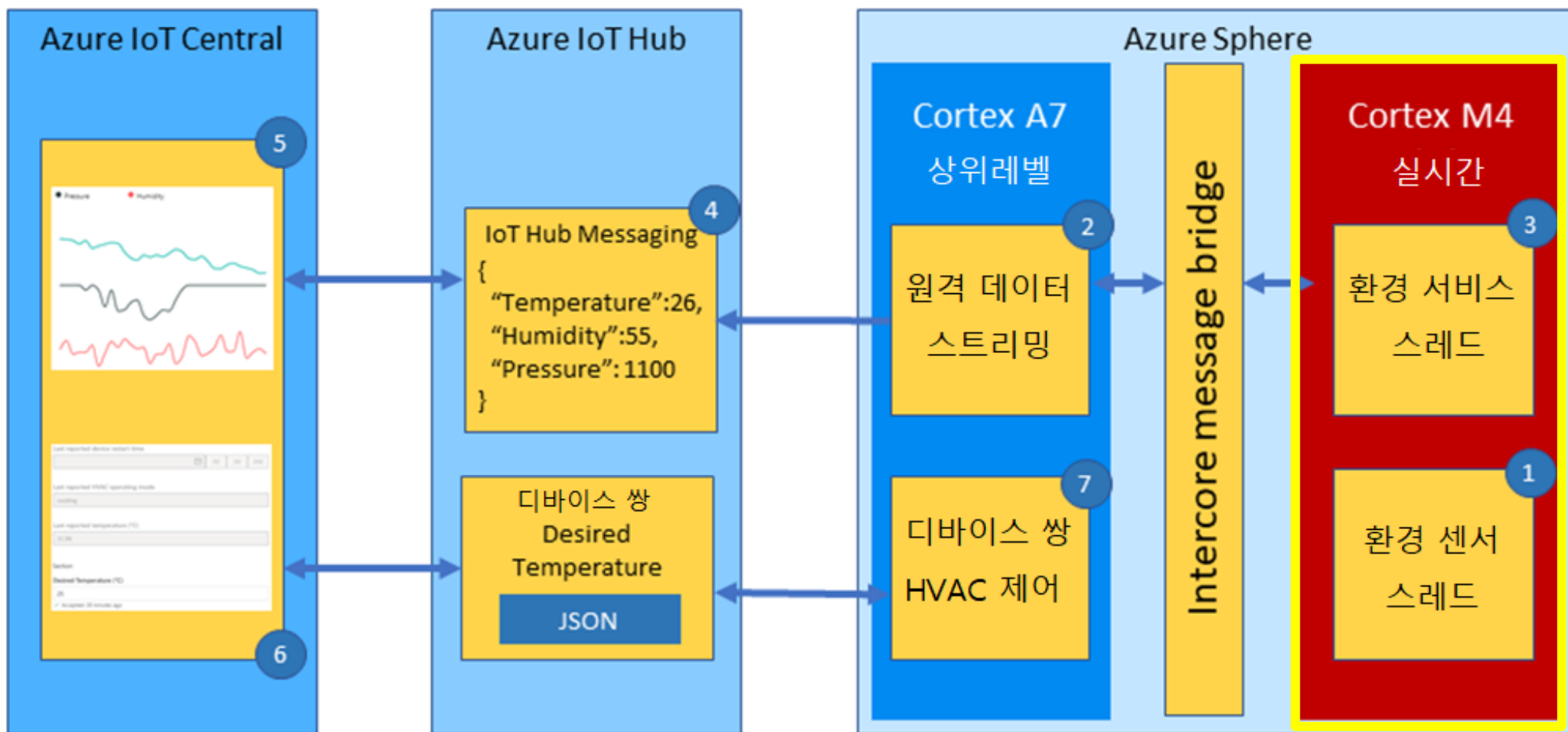
Graphical view of real-time events tracing to analyze, debug and tune system-level behavior

### GUIX

2D graphical user interfaces



# 앞으로 할 것



# 코어 간 통신 (inter-core communication)

- 보안 상 이유로 실시간 코어 앱은 네트워크 리소스에 접근 할 수 없음
  - 코어 간 통신을 활용하여 상위 레벨 앱에서 클라우드로 데이터를 보냄
  - 커맨드 / 데이터 타입 / 형태 사전 정의 필요
- 코어 간 통신 선언 구조체 (intercore\_contract.h)

```
typedef enum
{
    LP_IC_UNKNOWN,
    LP_IC_HEARTBEAT,
    LP_IC_ENVIRONMENT_SENSOR,
} LP_INTER_CORE_CMD;

typedef struct
{
    LP_INTER_CORE_CMD cmd;
    float temperature;
    float pressure;
    float humidity;
} LP_INTER_CORE_BLOCK;
```

# 실시간 코어 보안 및 통신

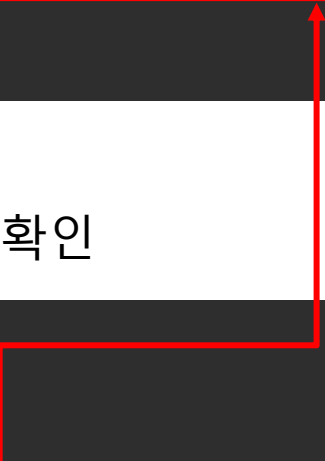
- 실시간 앱의 app\_manifest.json 에 주변 장치 및 다른 코어 앱의 연결 권한 추가

→ 상위 레벨 앱의 componentID 를 추가

```
{
  ...
  "AllowedApplicationConnections": [ "25025d2c-66da-4448-bae1-ac26fcdd3627" ]
  ...
}
```

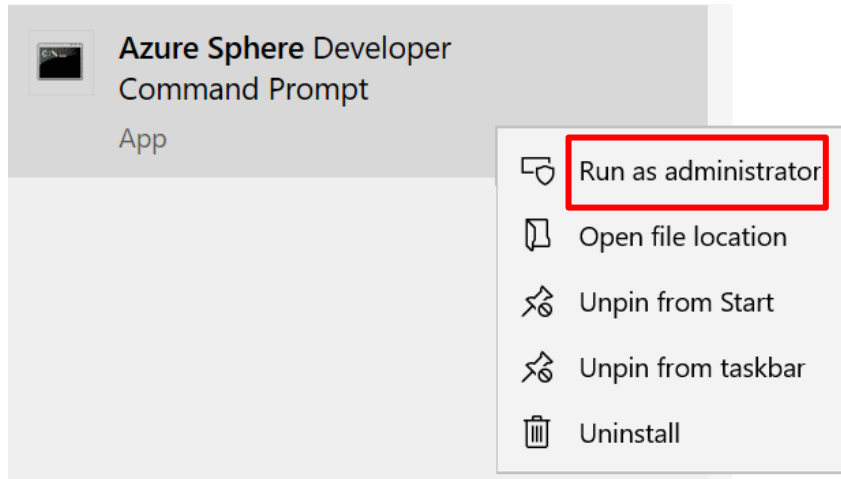
- 상위 레벨 앱의 app\_manifest.json 에서 componentID 확인

```
{
  "SchemaVersion": 1,
  "Name": "AzureSphereIoTCentral",
  "ComponentId": "25025d2c-66da-4448-bae1-ac26fcdd3627",
  ...
}
```



# 직접 해보기 – RT\_app 배포 (Cortex-M4)

- 관리자 모드로 Azure Sphere CLI 실행 후 실시간 앱 디버그 모드 설정



→ `azsphere device enable-development -enablertcoredebugging`

- 돌고 있는 앱 지우고 보드 재시작하기

`azsphere device sideload delete`

→ `azsphere device restart`



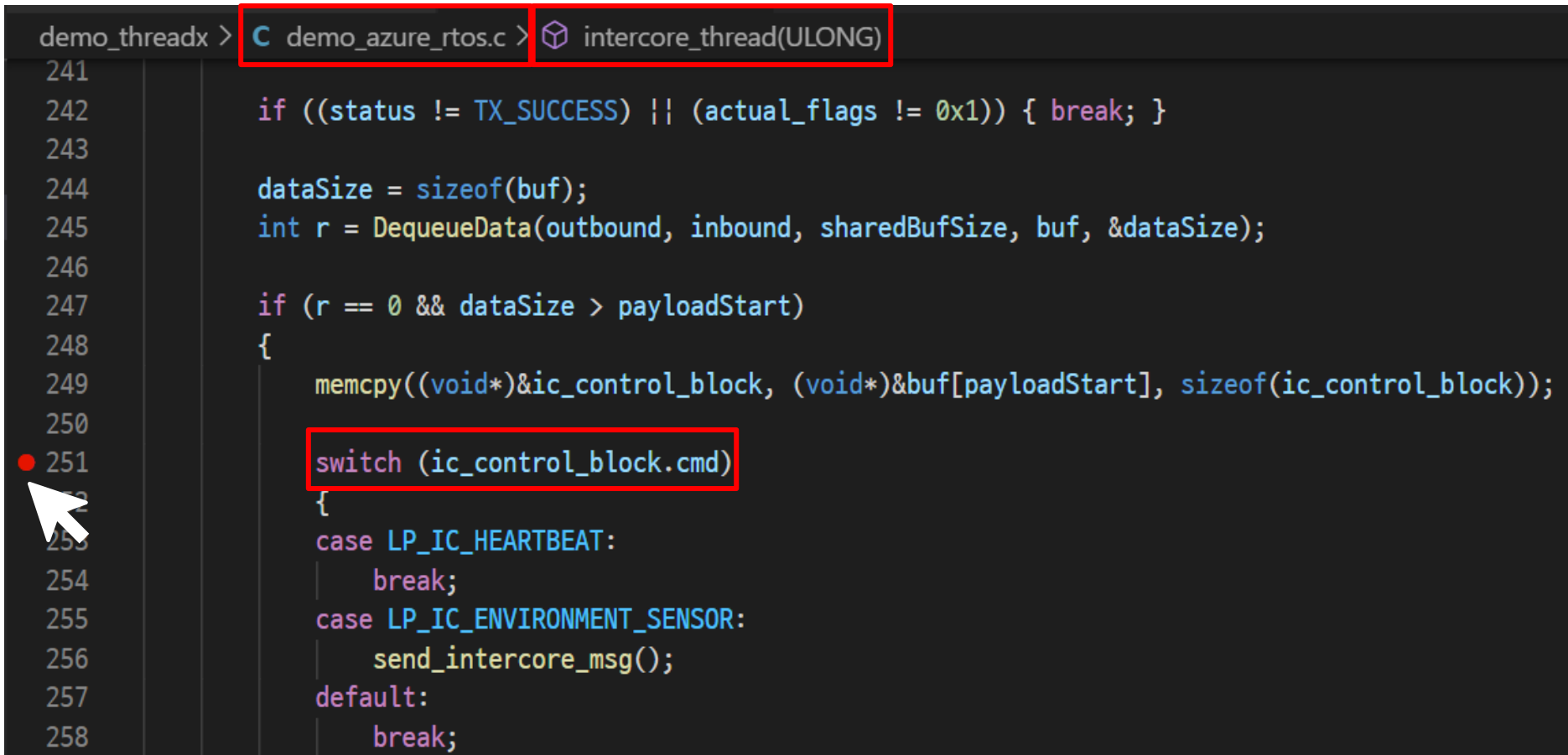
# 직접 해보기 – RT\_app 배포 (Cortex-M4)

- Lab\_6\_AzureRTOS\_Environment 폴더 열기
- demo\_azure\_rtos.c 의 intercore\_thread(ULONG) 에 breakpoint 설정하고 F5 눌러서 실행

```
demo_threadx > C demo_azure_rtos.c > intercore_thread(ULONG)
241
242     if ((status != TX_SUCCESS) || (actual_flags != 0x1)) { break; }
243
244     dataSize = sizeof(buf);
245     int r = DequeueData(outbound, inbound, sharedBufSize, buf, &dataSize);
246
247     if (r == 0 && dataSize > payloadStart)
248     {
249         memcpy((void*)&ic_control_block, (void*)&buf[payloadStart], sizeof(ic_control_block));
250
251         switch (ic_control_block.cmd)
252         {
253             case LP_IC_HEARTBEAT:
254                 break;
255             case LP_IC_ENVIRONMENT_SENSOR:
256                 send_intercore_msg();
257             default:
258                 break;
```

# 직접 해보기 – RT\_app 배포 (Cortex-M4)

- Lab\_6\_AzureRTOS\_Environment 폴더 열기
- demo\_azure\_rtos.c 의 intercore\_thread(ULONG) 에 breakpoint 설정하고 F5 눌러서 실행



```
demo_threadx > C demo_azure_rtos.c > intercore_thread(ULONG)
241
242     if ((status != TX_SUCCESS) || (actual_flags != 0x1)) { break; }
243
244     dataSize = sizeof(buf);
245     int r = DequeueData(outbound, inbound, sharedBufSize, buf, &dataSize);
246
247     if (r == 0 && dataSize > payloadStart)
248     {
249         memcpy((void*)&ic_control_block, (void*)&buf[payloadStart], sizeof(ic_control_block));
250
251         switch (ic_control_block.cmd)
252         {
253             case LP_IC_HEARTBEAT:
254                 break;
255             case LP_IC_ENVIRONMENT_SENSOR:
256                 send_intercore_msg();
257             default:
258                 break;
```

- 상위레벨 앱에서 센서 데이터 요청오면 breakpoint 에 멈춤

# 센서 값 읽는 것 확인

The screenshot displays a debugger interface for the file `imu_temp_pressure.c`. The left sidebar contains three panels: **VARIABLES**, **WATCH**, and **CALL STACK**.

- VARIABLES**: Under the **Locals** section, the variable `lps22hhTemperature_degC` is highlighted with a red box, showing a value of `31.34...`.
- WATCH**: This panel is currently empty.
- CALL STACK**: It shows the current function `lp_get_temperature_lps22h()` and the thread `read_sensor_thread(ULONG thread_i`. The status is **PAUSED ON STEP**.

The main window shows the source code of `imu_temp_pressure.c` at line 353. The code is as follows:

```
336 {  
337     return NAN;  
338 }  
339  
340 if (lps22hhDetected)  
341 {  
342     i16bit = 0;  
343  
344     lps22hh_read_reg(&pressure_ctx, LPS22HH_STATUS, (uint8_t*)&lps22hhReg, 1);  
345  
346     //Read output only if new value is available  
347  
348     if ((lps22hhReg.status.p_da == 1) && (lps22hhReg.status.t_da == 1))  
349     {  
350         lps22hh_temperature_raw_get(&pressure_ctx, &i16bit);  
351         lps22hhTemperature_degC = lps22hh_from_lsb_to_celsius(i16bit);  
352     }  
353     return lps22hhTemperature_degC;  
354 }  
355 return NAN;  
356 }  
357
```

Lines 350 and 351 are highlighted with a red box, corresponding to the variable shown in the **VARIABLES** pane. Line 353 is highlighted with a green background, indicating the current execution point.

# 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