Serial Plugin 사용 가이드

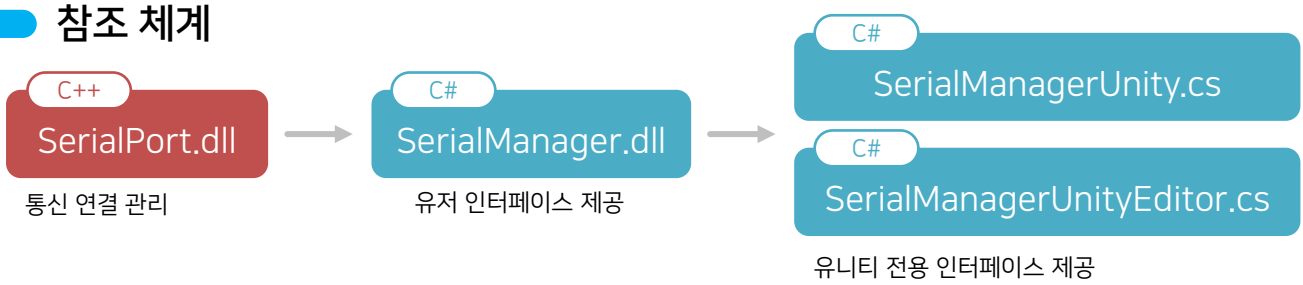
지원하는 프로그래밍 언어

- C++
- C#
- C# - 유니티

지원하는 연결 타입

- USB COM
- NI DAQ
- Bluetooth classic (BT classic)
- Bluetooth low energy (BLE)

참조 체계



설치 방법 (C# - Unity)

- SerialPlugin 폴더 내의 dll파일을 Assets/Plugins/ 아래의 경로에 복사 (경로가 없을 경우 생성)
- 복사가 필요한 dll 파일 목록
  - SerialPort.dll
  - SerialManager.dll
  - SerialManagerUnity.cs
  - /Editor/SerialManagerUnityEditor.cs

- SerialPort.dll 관련 에러가 발생할 시 System32 폴더 내의 dll 파일을 C/Windows/System32/ 경로에 복사
- 복사가 필요한 dll 파일 목록
  - msvcp140d.dll
  - vcruntime140_1.dll
  - vcruntine140.dll

- 위 dll 파일을 복사했는데도 에러가 발생할 시
  - ✓ NI DAQmx가 설치되어 있지 않은 경우
    - → WithoutDAQ 폴더 내의 SerialPort.dll 복사하여 사용
  - ✓ NI DAQmx가 설치되어 있는 경우
    - → 마지막 페이지의 dll 상세 참조 목록에 해당되는 파일들이 지정된 위치에 존재하는 지 확인

Serial Plugin 사용 가이드

Enumeration, Struct

UUID

Variable

- string **service**
  - 서비스 UUID
- string **rx**
  - 블루투스 디바이스의 RX data 특성 UUID
- string **tx**
  - 블루투스 디바이스의 TX data 특성 UUID

BaudRate

Variable

- int **bps1200**
  - 보드레이트 1200으로 설정
- int **bps2400**
  - 보드레이트 2400으로 설정
- int **bps4800**
  - 보드레이트 4800으로 설정
- int **bps9600**
  - 보드레이트 9600으로 설정
- int **bps19200**
  - 보드레이트 19200으로 설정
- int **bps38400**
  - 보드레이트 38400으로 설정
- int **bps57600**
  - 보드레이트 57600으로 설정
- int **bps115200**
  - 보드레이트 115200으로 설정

Serial Plugin 사용 가이드

Enumeration, Struct

DataBit

Variable

- int **bit5**
  - 데이터 비트 5로 설정
- int **bit6**
  - 데이터 비트 6으로 설정
- int **bit7**
  - 데이터 비트 7로 설정
- int **bit8**
  - 데이터 비트 8로 설정

Parity

Variable

- int **None**
  - 패리티 비트 없음
- int **Odd**
  - 홀수 비트 체크
- int **Even**
  - 짝수 비트 체크

StopBit

Variable

- int **bit1**
  - 정지 비트 1로 설정
- int **bit1p5**
  - 정지 비트 1.5로 설정
- int **bit2**
  - 정지 비트 2로 설정

Serial Plugin 사용 가이드

Enumeration, Struct

FlowControl

Variable

- int **None**
  - 흐름 제어 없음
- int **XonXoff**
  - 소프트웨어 흐름 제어 사용 (Xon, Xoff 문자 사용)
- int **RTSCTS**
  - 하드웨어 흐름 제어 사용 (RTS, CTS 케이블)

EncodingType

Variable

- int **DEC**
  - 패킷 바이트 데이터 10진수로 변환
  - 가독성을 위해 바이트 데이터 간 "-" 문자열이 추가됨
- int **HEX**
  - 패킷 바이트 데이터 16진수로 변환
- int **ASCII**
  - 패킷 바이트 데이터 ASCII 디코딩
- int **UTF8**
  - 패킷 바이트 데이터 Unicode(UTF-8) 디코딩
- int **UTF16**
  - 패킷 바이트 데이터 Unicode(UTF-16) 디코딩

Serial Plugin 사용 가이드

Class

SerialData

- SerialManager의 onDataReceived 이벤트 매개변수로 사용
- 패킷 및 데이터를 반환

Variable

`string` **packet**

- COM, BT classic, BLE : 패킷 데이터
- NI DAQ : 공백

`double[]` **data**

- COM, BT classic, BLE : null
- NI DAQ : 설정한 AI 채널의 데이터

SerialLog

- SerialManager의 onScanEnded, onLogReceived 이벤트 매개변수로 사용
- 로그 및 스캔 디바이스를 반환

Variable

`string` **log**

- 플러그인으로부터 수신한 로그

`string[]` **devices**

- 현재 PC에서 사용 가능한 포트

Serial Plugin 사용 가이드

Class

SerialHandle

Variable

int logLevel

- onLogReceived로 수신할 수 있는 로그 레벨 설정
- 0 : Error 로그 // 1 : Error + Normal 로그 // 2 : Error + Normal + Developer 로그

bool getPPSOnDataReceived

- true로 설정 시 패킷 수신 속도 측정
- default value : true

int **readonly** PPS

- getPPSOnDataReceived가 true일 때 초당 수신 패킷 수 반환
- COM, BT classic, BLE : 유효 string packet 수신 개수 계산
- NI DAQ : double[] 어레이 수신 개수 계산

bool **readonly** isConnected

- 현재 연결 상태 반환

int receiveByteSize

- COM, BT classic, BLE : 한 번에 수신하는 패킷의 byte 크기 설정
- NI DAQ : 설정값 무시
- default value : 1

int receiveBufferSize

- COM, BT classic, BLE : 수신 데이터를 저장하는 버퍼의 byte 크기 결정
- NI DAQ : 설정값 무시
- 해당 값이 클 수록 데이터를 안정적으로 수신하지만, 딜레이가 발생할 수 있음
- default value : 10,000

EncodingType encodingType

- COM, BT classic, BLE : 수신된 바이트 배열의 패킷 디코딩 타입 설정
- NI DAQ : 설정값 무시
- default value : ASCII

Serial Plugin 사용 가이드

Class

SerialHandle

Variable

byte[] **stopByte**

- COM, BT classic, BLE : 패킷 종료 지점 설정
- NI DAQ : 설정값 무시
- default value : null

ex

```
SerialManager _manager = new SerialManager();
_manager.stopByte = new byte[] { 69 } // ASCII 문자열 : E

// >> 문자열 ABCDEF 수신 시, ABCDE를 하나의 패킷으로 인식함
// >> stopByte를 { 68, 69 } 어레이로 설정할 경우, 문자열 DE을 패킷의 끝으로 인식함
```

int **packetLength**

- COM, BT classic, BLE : stopByte가 null이 아닐 때, stopByte로 끊어서 인식한 패킷의 길이가 주어진 packetLength와 일치할 때만 packet 반환
- 0 이하의 값이 설정되면 사용되지 않음
- NI DAQ : 설정값 무시
- default value : 0

bool **usePPSLimit**

- COM, BT classic, BLE : 리소스 관리를 위한 PPS 제한 사용 여부
- NI DAQ : 설정값 무시 (항상 1,000 Hz 로 샘플링)
- default value : false

bool **useCPULimit**

- COM, BT classic, BLE : CPU 점유율이 주어진 임계 값을 넘을 경우 데이터 수신 속도를 강제로 늦추는 알고리즘 적용 여부
- NI DAQ : 설정값 무시
- default value : false

bool **useMemoryLimit**

- COM, BT classic, BLE : RAM 잔여 량이 주어진 임계 값보다 낮을 경우 데이터 수신 속도를 강제로 늦추는 알고리즘 적용 여부
- NI DAQ : 설정값 무시
- default value : false

Serial Plugin 사용 가이드

Class

SerialHandle

Variable

int **PPSLimit**

- COM, BT classic, BLE : PPS 제한 값 설정 (usePPSLimit이 true일 경우에만 적용, 단위 : Hz)
- 0 이하 값이 설정되면 사용되지 않음
- 1,000 초과 값이 설정되면 1,000으로 고정됨
- NI DAQ : 설정값 무시
- default value : 300

double **CPULimit**

- COM, BT classic, BLE : CPU 제한 값 설정 (useCPULimit이 true일 경우에만 적용, 단위 : %)
- 0 이하 값이 설정되면 사용되지 않음
- NI DAQ : 설정값 무시
- default value : 95

int **memoryLimit**

- COM, BT classic, BLE : RAM 제한 값 설정 (useMemoryLimit이 true일 경우에만 적용, 단위 : MB)
- 0 이하 값이 설정되면 사용되지 않음
- NI DAQ : 설정값 무시
- default value : 200

❖ CPU, Memory 제한 값 초과 시
- ✓ 현재 PPS의 1/10 속도로 데이터 최대 수신 속도를 줄임
- ✓ 10초간 CPU 및 Memory 사용량 모니터링
- ✓ 10초 뒤 CPU 및 Memory 사용량이 제한 값을 넘지 않을 경우 데이터 최대 수신 속도를 원상태로 되돌림
- ✓ 10초 뒤에도 CPU 및 Memory 사용량이 제한 값을 넘었을 경우 최대 수신 속도를 1/10으로 더 줄인 후, 10초간 모니터링 → 이후 과정 반복

❖ CPULimit, MemoryLimit 옵션은 리소스 사용량 과다로 인한 크래시 및 프리징 현상을 막기 위한 응급 대처로, 갑자기 수신 속도가 줄어 부자연스러운 프로그램 동작을 초래할 수 있음 → CheckResources 메서드를 통해 CPU 및 Memory 사용량을 런타임 중에 모니터링 하고, PPSLimit을 통해 리소스 점유율을 조절하는 것을 권장함

Serial Plugin 사용 가이드

Class

SerialHandle

Method

void Connect(string deviceName)

- 주어진 이름을 가지는 Bluetooth classic 디바이스에 연결

void Connect(string deviceName, UUID uuid)

- 주어진 이름과 UUID에 일치하는 BLE 디바이스에 연결

HM-10 기본 값

```
Service : 0000FFE0-0000-1000-8000-00805F9B34FB
RX      : 0000FFE1-0000-1000-8000-00805F9B34FB
TX      : 0000FFE1-0000-1000-8000-00805F9B34FB
```

Nordic Semiconductor (UART) 기본 값

```
Service : 6E400001-B5A3-F393-E0A9-E50E24DCCA9E
RX      : 6E400002-B5A3-F393-E0A9-E50E24DCCA9E
TX      : 6E400003-B5A3-F393-E0A9-E50E24DCCA9E
```

void Connect(string comPort, BaudRate baudRate, DataBit dataBit, Parity parity, StopBit stopBit, FlowControl flowControl)

- 주어진 설정을 만족하는 COM 포트에 연결
- comPort는 "COM1"과 같이 COM + int 조합으로 주어져야 함
- dataBit ~ flowControl은 입력되지 않을 시 기본값 사용

기본 값

- dataBit = DataBit.bit8
- parity = Parity.None
- stopBit = StopBit.bit1
- flowControl = FlowControl.None

void Connect(string deviceName, int[] ports)

- 주어진 이름에 해당하는 NI DAQ에 연결
- deviceName은 "Dev2"와 같이 Dev + int 조합으로 주어져야 함
- ports에는 analog input 포트 입력
  (AI0, AI1 = int[] { 0, 1 })

void Disconnect()

- 연결 해제

Serial Plugin 사용 가이드

Class

SerialHandle

Method

void ScanDevices()

- PC에서 사용 가능한 포트 검색

float[] CheckResources()

- CPU 및 Memory 점유율 확인
- CPU 및 Memory 점유율은 디바이스가 연결된 후 동작하기 때문에, 디바이스가 연결되지 않은 상태라면 제대로 된 값을 반환하지 않음
- 반환된 변수는 길이가 2인 array로써, 첫 번째 멤버로 CPU 점유율, 두 번째 멤버로 Memory 잔여 량을 반환함

void SendData(string data)

- COM, BT classic, BLE : data ASCII 타입으로 인코딩하여 송신
- NI DAQ : 호출 무시

void SendData(byte[] data)

- COM, BT classic, BLE : data 인코딩 없이 송신
- NI DAQ : 호출 무시

Serial Plugin 사용 가이드

Class

SerialHandle

Event

void OnScanEnded(SerialLog e)

- 스캔 완료 시 호출
- ScanDevices() 메서드 사용 시 사용 가능한 포트 스캔 후 호출됨
- e.devices로 스캔 완료된 디바이스 타입 및 이름 확인 가능

void OnLogReceived(SerialLog e)

- 로그 수신 시 호출
- SerialManager 클래스의 log level에 따라 출력 로그 종류가 변화함
- e.log로 수신된 로그 확인 가능

void OnConnected()

- 디바이스 연결 성공 시 호출

void OnConnectionFailed()

- 디바이스 연결 실패 시 호출

void OnDisconnected()

- 디바이스 연결 종료 시 호출

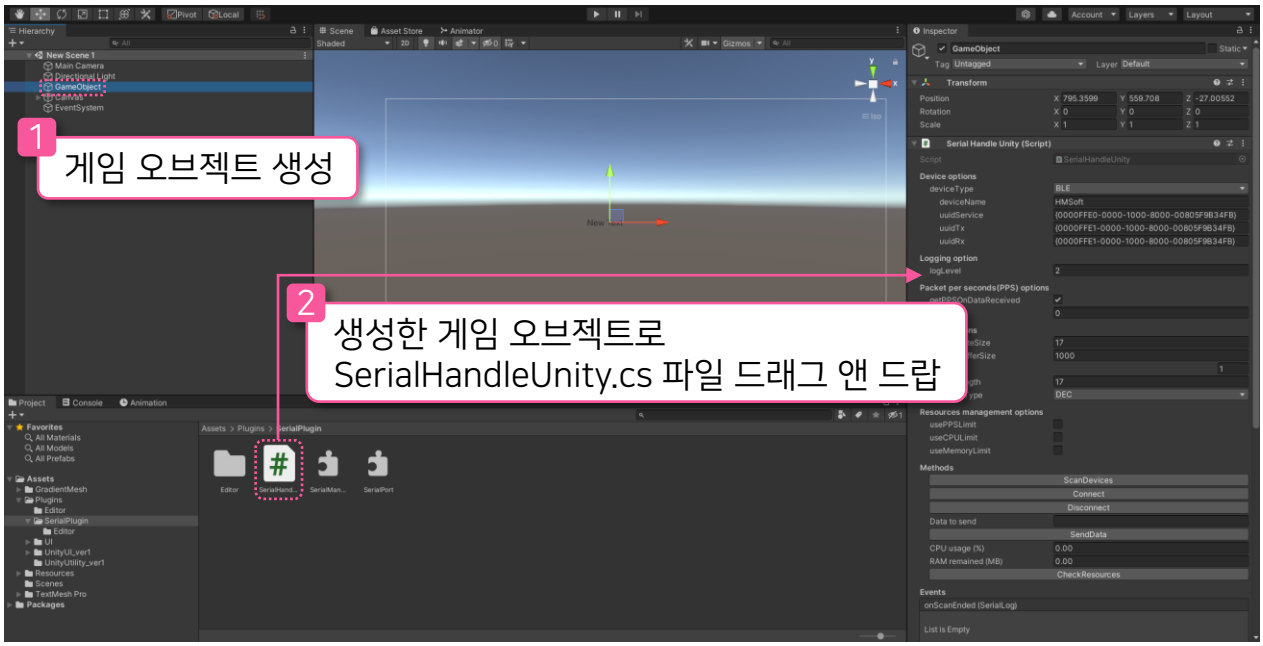
void OnDataReceived(SerialData e)

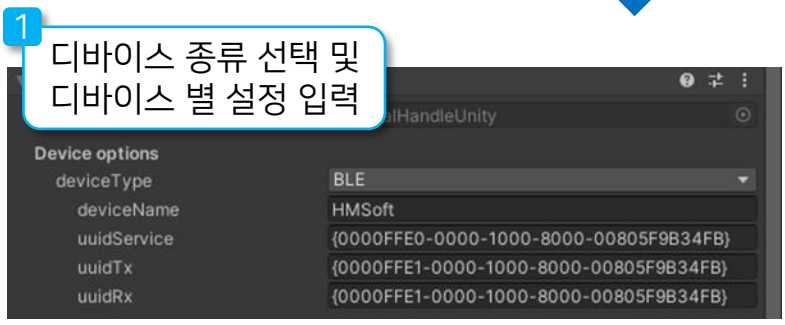
- 디바이스로부터 데이터 수신 시 호출
- COM, BT classic, BLE : e.packet으로 수신 데이터 반환
- NI DAQ : e.data로 수신 데이터 반환

Serial Plugin 사용 가이드

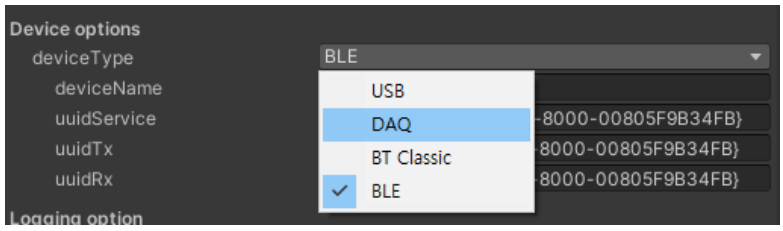
예제

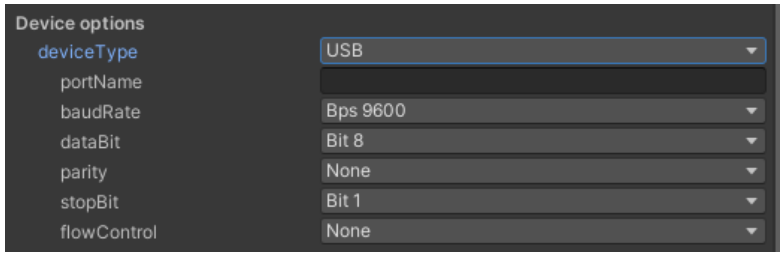
- 유니티 애드온 사용법





- deviceType 선택 시 드랍다운 메뉴 확장
- deviceType 변경 시 디바이스에 설정에 필요한 변수로 변경됨



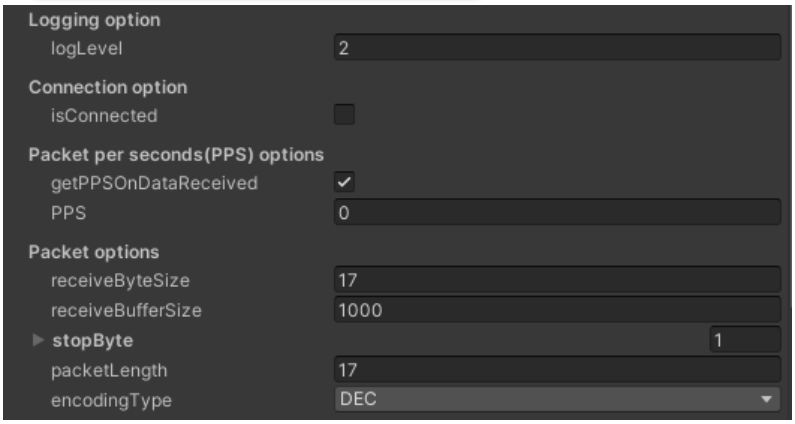


Serial Plugin 사용 가이드

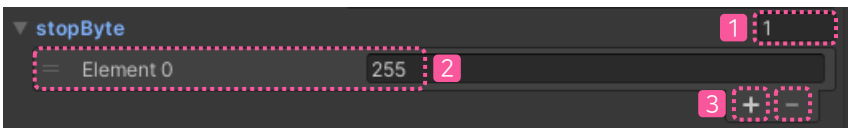
예제

- 유니티 애드온 사용법

2 로깅, 연결 상태 및 패킷 옵션

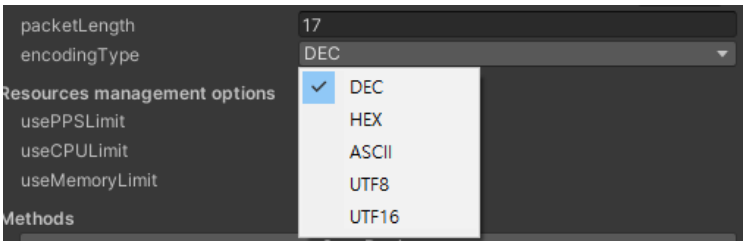


- isConnected, PPS는 인스펙터나 스크립트에서 변경이 불가능하고, 읽을 수만 있음
- stopByte는 변수명을 클릭하면 아래와 같이 설정 가능



1. stopByte 어레이의 길이
2. 현재 추가되어 있는 stopByte 멤버
3. 멤버 추가 및 삭제

- encodingType 선택 시 드랍다운 메뉴 확장



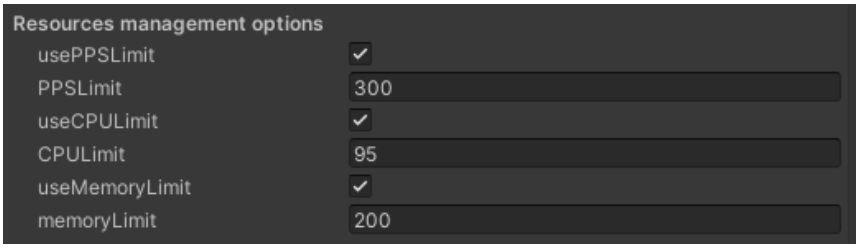
Serial Plugin 사용 가이드

예제

- 유니티 애드온 사용법

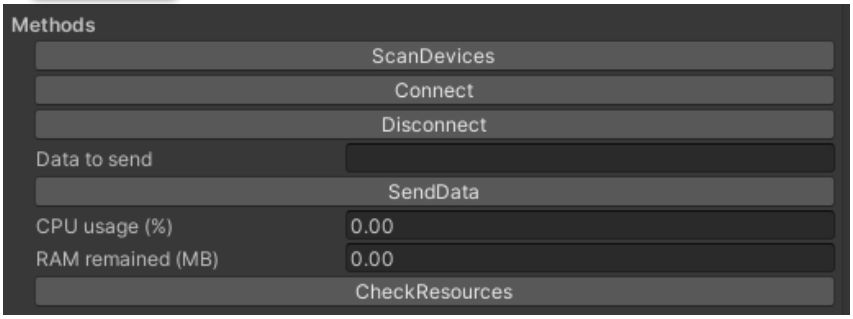
3 리소스 관리 옵션

- usePPSLimit, useCPULimit, useMemoryLimit은 토글해서 값을 true로 만들면 각 limit 값을 입력할 수 있는 창이 생성됨



- usePPSLimit, useCPULimit, useMemoryLimit, PPSLimit, CPULimit, memoryLimit은 런타임 중에도 인스펙터 및 스크립트로 변경이 가능

4 메서드



- Data to send에 string 입력 후 SendData 클릭 시 입력된 문장을 송신함
  - Data to send에 입력된 문장은 인스펙터의 SendData 버튼에만 영향을 끼치며, 스크립트로 호출하는 SendData 메서드와는 무관함
  - Data to send에 입력된 문장은 ASCII로 자동 인코딩 되어 전송됨
- 디바이스가 연결된 상태에서 CheckResources 클릭 시 CPU usage 및 RAM remained에 해당하는 리소스가 표시됨

Serial Plugin 사용 가이드

예제

- 유니티 애드온 사용법

5 이벤트

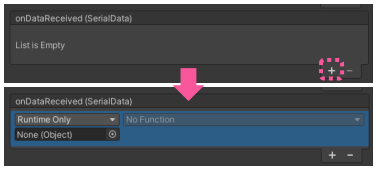
- onLogReceived는 자동으로 처리하여 로그를 Unity의 Log 창에 표시함 (따로 메서드를 등록할 필요 없음)

- 지정된 매개 변수를 받는 public 메서드를 등록할 수 있음 (onConnected, onConnectionFailed, onDisconnected : 매개 변수 없음 onScanEnded : SerialLog, onDataReceived : SerialData)
- 이벤트는 아래와 같은 방법으로 인스펙터에 등록하거나, 스크립트로 등록할 수 있음

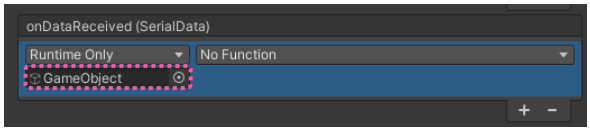
인스펙터로 등록

```
public class EventTest : MonoBehaviour
{
    public void OnDataReceived(SerialData e)
    {
        Debug.Log(e.packet);
    }
}
```

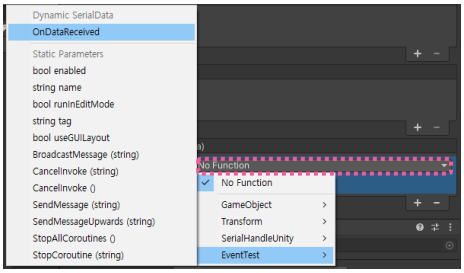
1. 클래스 및 메서드 생성



2. + 버튼을 눌러 입력창 생성



3. 입력창에 생성한 클래스를 component로 보유한 오브젝트 드래그 앤 드랍



4. 원하는 메서드 등록

5. 등록 완료

Serial Plugin 사용 가이드

예제

- 유니티 애드온 사용법

5 이벤트

- onLogReceived는 자동으로 처리하여 로그를 Unity의 Log 창에 표시함 (따로 메서드를 등록할 필요 없음)

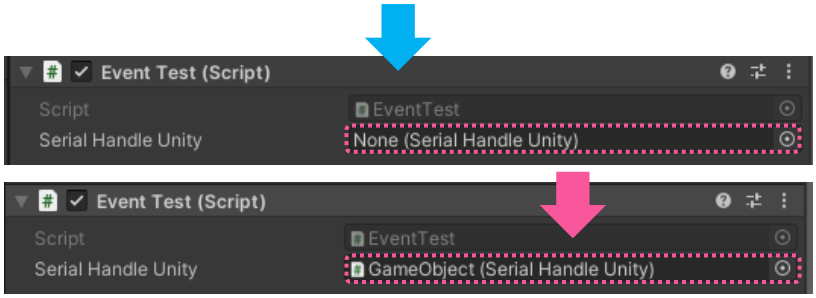
스크립트로 등록

```
public class EventTest : MonoBehaviour
{
    public SerialHandleUnity serialHandleUnity;

    private void Start()
    {
        serialHandleUnity.onDataReceived.AddListener(OnDataReceived);
    }

    public void OnDataReceived(SerialData e)
    {
        Debug.Log(e.packet);
    }
}
```

1. 이벤트 등록 대상 component 선언
2. OnDataReceived 메서드를 onDataReceived 이벤트에 등록
3. OnDataReceived 메서드 선언



- SerialHandleUnity 컴포넌트를 가진 게임 오브젝트 인스펙터로 드래그 앤 드랍
  (여기서 넣어준 SerialHandleUnity의 이벤트에 메서드가 등록됨)

Serial Plugin 사용 가이드

예제

- C#, 블루투스 연결

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
// 사용 환경에 맞게 네임스페이스 참조
using SerialManager; // SerialPlugin 사용을 위한 네임스페이스 참조

public class SerialTest : MonoBehaviour // SerialTest에 해당하는 클래스 이름은 자유롭게 설정
{
    SerialHandle _manager;

    void Start()
    {
        _manager = new SerialHandle();
        _manager.logLevel               = 1; // Error, Normal 로그 수신
        _manager.getPPSOnDataReceived   = true;
        _manager.onLogReceived          += OnLogReceived;
        _manager.onScanEnded            += OnScanEnded;
        _manager.onConnected            += OnConnected;
        _manager.onConnectionFailed     += OnConnectionFailed;
        _manager.onDisconnected         += OnDisconnected;
        _manager.onDataReceived         += OnDataReceived;
    }

    public void Connect()
    {
        _manager.receiveByteSize = 1; // 1 바이트씩 데이터 수신
        _manager.receiveBufferSize = 1000; // 수신 버퍼 1000 바이트로 설정
        _manager.packetLength = 5; // 패킷 길이를 5 바이트로 인식
        _manager.stopByte = new byte[] { 10 };
        // “\n“ (줄바꿈-LF)을 stopByte로 인식
        // 아두이노의 경우, println을 사용하면 패킷의 끝에 “\n”을 추가함
        _manager.encodingType = EncodingType.ASCII; // ASCII 바이트 데이터 변환
        _manager.usePPSLimit = true;
        _manager.PPSLimit = 300; // PPS 제한 설정 후, 300Hz로 제한
        _manager.Connect("HC-06"); // “HC-06” 이름을 가지는 BT classic 디바이스에 연결
    }

    public void ScanDevice()
    {
        // 사용 가능한 포트 검색
        _manager.ScanDevice();
    }

    public void Send(string message)
    {
        // 전달 받은 string을 ASCII 타입으로 인코딩하여 연결된 디바이스로 출력
        _manager.SendData(message);
    }

    public void Send(byte[] data)
    {
        // 전달 받은 byte data를 연결된 디바이스로 출력
        _manager.SendData(data);
    }

    ...
```

Serial Plugin 사용 가이드

예제

- Unity, 블루투스 연결

```
...

    public void Disconnect()
    {
        // 블루투스 연결 해제
        _manager.Disconnect();
    }

    private void OnLogReceived(SerialLog e)
    {
        Debug.Log(e.log);
    }

    private void OnScanEnded(SerialLog e)
    {
        for (int i = 0; i < e.devices.Length; i++) {
            Debug.Log(e.devices[i]);
        }
    }

    private void OnConnected()
    {
        Debug.Log("Connected");
    }

    private void OnConnectionFailed()
    {
        Debug.Log("Connection failed");
    }

    private void OnDisconnected()
    {
        Debug.Log("Disconnected");
    }

    private void OnDataReceived(SerialData e)
    {
        Debug.Log($"{_manager.pps} : {e.packet}");
    }

    private void OnApplicationQuit()
    {
        _manager.Disconnect();
        // 디바이스 연결이 종료되지 않은 상태로 유니티 플레이가 종료되는 것을 방지
        // 디바이스 연결은 별도의 스레드에서 관리되기 때문에, 유니티 플레이 종료 전에
        // 연결을 종료하지 않으면 크래시 발생
    }
}
```

- 여러 개의 디바이스 연결 → 위 코드를 여러 개 만들면 됨
- 수신 바이트 수가 정해져 있을 경우, bufferSize를 거기에 맞추는 것이 좋음
- 수신 바이트 수가 정해져 있지 않을 경우, stopByte를 설정하고 bufferSize를 1로 설정하는 것이 좋음
- BT classic 디바이스의 경우, 페어링이 되어 있지 않으면 플러그인이 페어링을 요청하고, 페어링에 성공할 시 다시 연결을 시도함
- BLE 디바이스의 경우, 페어링이 되어 있지 않으면 연결 실패를 출력

Serial Plugin 사용 가이드

파일 별 상세 참조 파일 및 경로

- SerialPort.dll
  - C:₩Windows₩System32₩kernel32.dll
  - C:₩Windows₩System32₩ole32.dll
  - C:₩Windows₩System32₩nicaiu.dll
  - C:₩Windows₩System32₩MSVCP140.dll
  - C:₩Windows₩System32₩Setupapi.dll
  - C:₩Windows₩System32₩BluetoothApis.dll
  - C:₩Windows₩System32₩VCRUNTIME140_1.dll
  - C:₩Windows₩System32₩VCRUNTIME140.dll
  - C:₩Windows₩System32₩ucrtbased.dll

- SerialManager.dll
  - C:₩Windows₩SysWOW64₩mscoree.dll
  - C:₩Windows₩Microsoft.NET₩Framework64₩v4.0.30319₩mscorlib.dll
  - C:₩Windows₩Microsoft.NET₩assembly₩GAC_MSIL₩System₩v4.0_4.0.0.0__b77a5c561934e089₩System.dll
  - C:₩Windows₩Microsoft.NET₩assembly₩GAC_MSIL₩System.Core₩v4.0_4.0.0.0__b77a5c561934e089₩System.Core.dll