로직 설명서

2023. 06

HYUNDAI MOTOR GROUP

HYUNDAI NGV

5G·UNMANNED VEHICLE RESEARCH CENTER
5G/무인이동체 융합기술 연구센터

I. 서론

1. **본 문서는 ‘무선 통신 기반 차량 협력측위 PoC” 결과물에 대한 로직 설명서임**
2. **연구 목표에 따라 UWB 통신을 이용한 협력측위 알고리즘 구현을 수행하였으며, 동작 방법에 대한 내용이 기술됨**
3. **로직 설명서 구성은 다음과 같음**
    1) 환경 구성: UWB 모듈 환경 구성 방법에 대한 내용
    2) UWB 초기 세팅: Pozyx사의 UWB 통신 모듈 제어 환경 세팅 및 예제 코드 동작
    3) 스크립트 구성: 협력측위 알고리즘 구조와 스크립트에 대한 상세한 내용
    4) 협력측위 동작: UWB와 협력측위 스크립트 실행 방법 설명
    5) 모형차 구성: 협력측위 테스트베드에 사용된 모형차 구성에 대한 상세 내용
    6) 모형차 제어: 모형차 웹서버 제어 방법 및 자율주행 방법 소개

II. 환경 구성

1. UWB 모듈

- 본 과제에서는 벨기에 Pozyx 사의 UWB creator KIT으로 테스트베드를 구성함
- UWB 상세 사양은 https://www.pozyx.io/products/hardware/tags/developer-tag 에서 확인 가능함

<Developer Tag>

<Creator Anchor>

2. 협력측위 동작 스크립트

- [H. Kim, IEEE WCNC 2018]* 논문을 참고하여 ADMM 기반 협력측위 알고리즘을 설계함
- Python 기반 언어로 작성하였으며 Visual Studio Code 프로그램을 사용하여 알고리즘 실행함
  (타 프로그램에서도 동작 가능)

* H. Kim, S. H. Lee and S. Kim, "Cooperative localization with distributed ADMM over 5G-based VANETs," in *Proc. IEEE Wirel. Commun. Netw. Conf. (WCNC)*, 2018.

III. UWB 초기 세팅

1. 구성

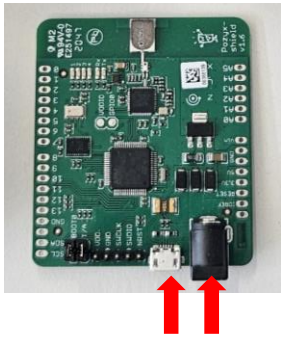
➢ 4개의 앵커와 3개의 태그를 사용하며 각 앵커와 태그에는 ID가 부여됨

<UWB 앵커 및 태그>

<UWB 앵커와 태그 ID 확인 방법>

2. 전원 공급

➢ Micro 5pin 혹은 DC 전원 공급 케이블을 통해 앵커와 태그가 동작함



<UWB 전원 공급 방법>

III. UWB 초기 세팅

3. Pozyx UWB 환경 구성

- Pozyx사에서 제공하는 거리 측정 코드는 Python 언어로 작성되어 사용할 PC에 개발 환경 구성을 해야함
- Pozyx UWB 관련 Python 라이브러리 및 초기 설정은 document (https://docs.pozyx.io/creator/python) 에서 제공
- 라이브러리 설치

Pozyx UWB KIT 동작(거리, 가속도 등 측정)을 위한 라이브러리

  - ✓ PC에 Python와 pypozyx 라이브러리를 설치 → Window, Mac, Linux 마다 설치 방법에 차이가 있음 공유 링크 참고
  - ✓ Pozyx 연결을 위해 pozyx serial connection을 설치 → 공유 링크 참고

Installing this package

Just run `pip install pypozyx`

PyPozyx is now installed. To check whether it is: if you followed all the steps correctly, and know which port your Pozyx is on, the following code should work:

```
from pypozyx import PozyxSerial
port = 'COMX' # on UNIX systems this will be '/dev/ttyACMX'
p = PozyxSerial(port)
```

If your port is correct and the serial connection to the Pozyx isn't used by other software, this will run without any errors.

  - ✓ 라이브러리 설치가 끝났다면, UWB 태그를 PC와 연결하고 아래 작업을 수행하여 연결 port를 찾음 → Port가 찾아지면 설치 정상적 완료, 에러 발생 시 라이브러리 설치 및 공유 링크 document 확인

But! How do I know what port my Pozyx is on?

- You can see the COM ports on your system easily using Python with: `python -c "from pypozyx import *;list_serial_ports()"`
- **NEW** You can quickly find whether there's a recognized Pozyx device using: `python -c "from pypozyx import *;print(get_first_pozyx_serial_port())"`

III. UWB 초기 세팅

4. 단말 간 거리 측정

➢ 앞서 공유된 링크의 메뉴 'Getting started'에서 UWB를 이용한 다양한 기능 예시가 제공됨

➢ Pozyx UWB 두 개를 사용할 때 단말 간에 거리 측정 코드는 다음과 같음

CODE

```
1  if __name__ == "__main__":
2      port = 'COM1'                  # COM port of the Pozyx device
3      remote_id = 0x605D             # the network ID of the remote device
4      remote = False                 # whether to use the given remote device for ranging
5      if not remote:
6          remote_id = None
7      destination_id = 0x1000        # network ID of the ranging destination
8      range_step_mm = 1000           # distance between each LED lighting up.
9      ranging_protocol = POZYX_RANGE_PROTOCOL_PRECISION #ranging protocol
```

PC에 포직스가 연결된 port

UWB ID

거리 측정값을 수집하는 UWB

Remote_id의 측정값 무선 수신 유/무선 선택

UWB ID

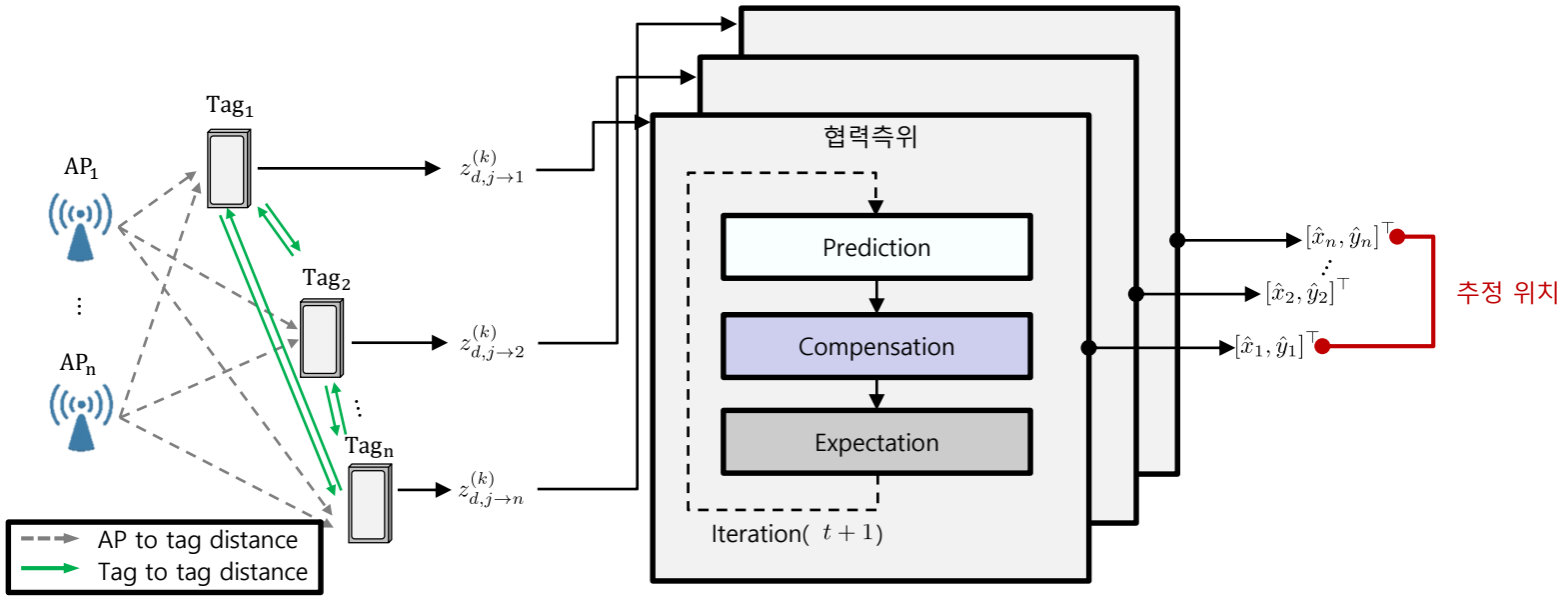
거리 측정 대상

✓ 두 단말에 전원 공급 후에 코드를 실행하면 실시간 거리 측정값 확인 가능
→ 에러 문구가 나오면 UWB 태그 연결 상태 또는 UWB ID 확인

Ⅳ. 스크립트 구성

1. 스크립트 동작 구조 설명

➢ 협력측위 동작 스크립트는 초기값 및 파라미터 세팅, UWB 거리 측정, 협력측위 알고리즘으로 구성됨



<협력측위 동작 프로세스 모식도>

➢ 협력측위 알고리즘에는 prediction step, compensation step, expectation step으로 구성되며, 여기서 prediction step과 compensation step은 스크립트에서 함수화 되어있음

IV. 스크립트 구성

2. 라이브러리 설정

➢ 스크립트 동작을 위해 pypozyx, pandas, numpy의 라이브러리가 사용됨

- ✓ Pypozyx: Pozyx UWB 동작 관련 라이브러리
- ✓ Pandas: 데이터 조작 및 분석을 위한 라이브러리
- ✓ Numpy: 다차원 배열 연산을 위한 라이브러리

3. 초기값 설정

➢ 스크립트 내에 UWB ID, 태그 초기 위치, 앵커 위치, 추정 초기값, 보상 단계 (compensation step) 초기값, 차량 수, 패널티 계수 등의 변수 초기값 설정이 필요함

```
Tag1 = np.array([3640, 6325]).reshape(2,1) #태그 실제 위치
# print(Test) 태그 위치 초기값 (x, y)
Tag2 = np.array([3640, 9730]).reshape(2,1) #태그 실제 위치
Tag3 = np.array([3640,3100]).reshape(2,1) #태그 실제 위치
                추정 초기값 (x, y)
est1 = np.array([100, 140]).reshape(2,1) #추정 초기값
est2 = np.array([200, 320]).reshape(2,1) #추정 초기값
est3 = np.array([250, 250]).reshape(2,1) #추정 초기값
ap1 = np.array([0, 0]).reshape(2,1) #1번 앵커 위치
ap2 = np.array([3600, 0]).reshape(2,1) #2번 앵커 위치
ap3 = np.array([0, 3600]).reshape(2,1) #3번 앵커 위치
ap4 = np.array([3600, 3600]).reshape(2,1) #4번 앵커 위치
                앵커 위치 (x, y)
```

```
t1_comp1=np.array([0,0]).reshape(2,1) #Compensation 초기값
t1_comp2=np.array([0,0]).reshape(2,1) #Compensation 초기값
t1_compAP1=np.array([0,0]).reshape(2,1) #Compensation 초기값
t1_compAP2=np.array([0,0]).reshape(2,1) #Compensation 초기값
t1_compAP3=np.array([0,0]).reshape(2,1) #Compensation 초기값
t1_compAP4=np.array([0,0]).reshape(2,1) #Compensation 초기값
태그1 보상 단계 초기값 (x, y)
t2_comp1=np.array([0,0]).reshape(2,1) #Compensation 초기값
t2_comp2=np.array([0,0]).reshape(2,1) #Compensation 초기값
t2_compAP1=np.array([0,0]).reshape(2,1) #Compensation 초기값
t2_compAP2=np.array([0,0]).reshape(2,1) #Compensation 초기값
t2_compAP3=np.array([0,0]).reshape(2,1) #Compensation 초기값
t2_compAP4=np.array([0,0]).reshape(2,1) #Compensation 초기값
태그2 보상 단계 초기값 (x, y)
t3_comp1=np.array([0,0]).reshape(2,1) #Compensation 초기값
t3_comp2=np.array([0,0]).reshape(2,1) #Compensation 초기값
t3_compAP1=np.array([0,0]).reshape(2,1) #Compensation 초기값
t3_compAP2=np.array([0,0]).reshape(2,1) #Compensation 초기값
t3_compAP3=np.array([0,0]).reshape(2,1) #Compensation 초기값
t3_compAP4=np.array([0,0]).reshape(2,1) #Compensation 초기값
태그3 보상 단계 초기값 (x, y)
```

→ 보상 단계 초기값은 인접 단말 수만큼 증가

IV. 스크립트 구성

3. 초기값 설정

➢ 스크립트 내에 UWB ID, 태그 초기 위치, 앵커 위치, 추정 초기값, 보상 단계 (compensation step) 초기값, 차량 수, 패널티 계수 등의 변수 초기값 설정이 필요함

```
J=6# 인접 단말 수 인접 단말 수
T=10 ADMM 기반 협력측위 계산 반복 횟수

a = np.zeros((2,T+1)) #실시간 태그 위치 저장 변수
b = np.zeros((2,T+1)) #실시간 태그 위치 저장 변수
c = np.zeros((2,T+1)) #실시간 태그 위치 저장 변수

aa = np.zeros((2,T)) #실시간 태그 위치 저장 변수
bb = np.zeros((2,T)) #실시간 태그 위치 저장 변수
cc = np.zeros((2,T)) #실시간 태그 위치 저장 변수

JJ = 1/(J+1)


p = 100 # ADMM 알고리즘 패널티 계수 파라미터
패널티 계수 파라미터
```

```
remote_id = 0x6824 1번 태그   # UWB tag1 ID
remote = True                 # Remote 사용 여부 결정 (True or False)
if not remote:        Remote 사용 여부
    remote_id = None

remote_id1 = 0x6846  #UWB tag2 ID
remote = True    2번 태그
if not remote:
    remote_id = None

remote_id2 = 0x6862 #UWB tag3 ID
remote = True    3번 태그
if not remote:
    remote_id = None

# destination_id1 = 0x0D5D
# destination_id2 = 0x687B
# destination_id3 = 0x684E
# destination_id4 = 0x684F

destination_id1 = 0x1162   1번 앵커
destination_id2 = 0x114A   2번 앵커
destination_id3 = 0x1126   3번 앵커
destination_id4 = 0x1136   4번 앵커
```

IV. 스크립트 구성

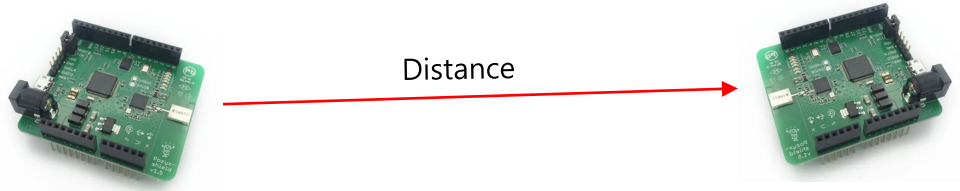
4. 거리 측정 스크립트

- Pozyx사에서 제공해주는 함수 ReadyToRange를 사용하여 입력한 태그와 앵커 간에 거리가 측정됨
- 함수에서 destination_id와 remote_id만 변경, 다른 파라미터는 설정 바꾸지 않음

```
r = ReadyToRange(pozyx, destination_id1, range_step_mm, ranging_protocol, remote_id) #Remote id와 destination1 사이 거리측정
```

거리 측정 대상 단말 ID

거리 측정 태그 ID



<UWB 거리 측정 모식도>

거리 측정 및 수집 스크립트 설명

태그1은 4개의 앵커와 2개의 인접 태그(2, 3)로부터 총 6개의 거리 측정하여 변수 d1에 저장

태그 1 거리 측정

```
r = ReadyToRange(pozyx, destination_id1, range_step_mm, ranging_protocol, remote_id) #Remote id와 destination1 사이 거리측정
r1 = ReadyToRange(pozyx, destination_id2, range_step_mm, ranging_protocol, remote_id) #Remote id와 destination2 사이 거리측정
r2 = ReadyToRange(pozyx, destination_id3, range_step_mm, ranging_protocol, remote_id) #Remote id와 destination3 사이 거리측정
r3 = ReadyToRange(pozyx, destination_id4, range_step_mm, ranging_protocol, remote_id) #Remote id와 destination4 사이 거리측정
r4 = ReadyToRange(pozyx, remote_id1, range_step_mm, ranging_protocol, remote_id) #Remote id와 remote id1 사이 거리측정
r5 = ReadyToRange(pozyx, remote_id2, range_step_mm, ranging_protocol, remote_id) #Remote id와 remote id2 사이 거리측정
```

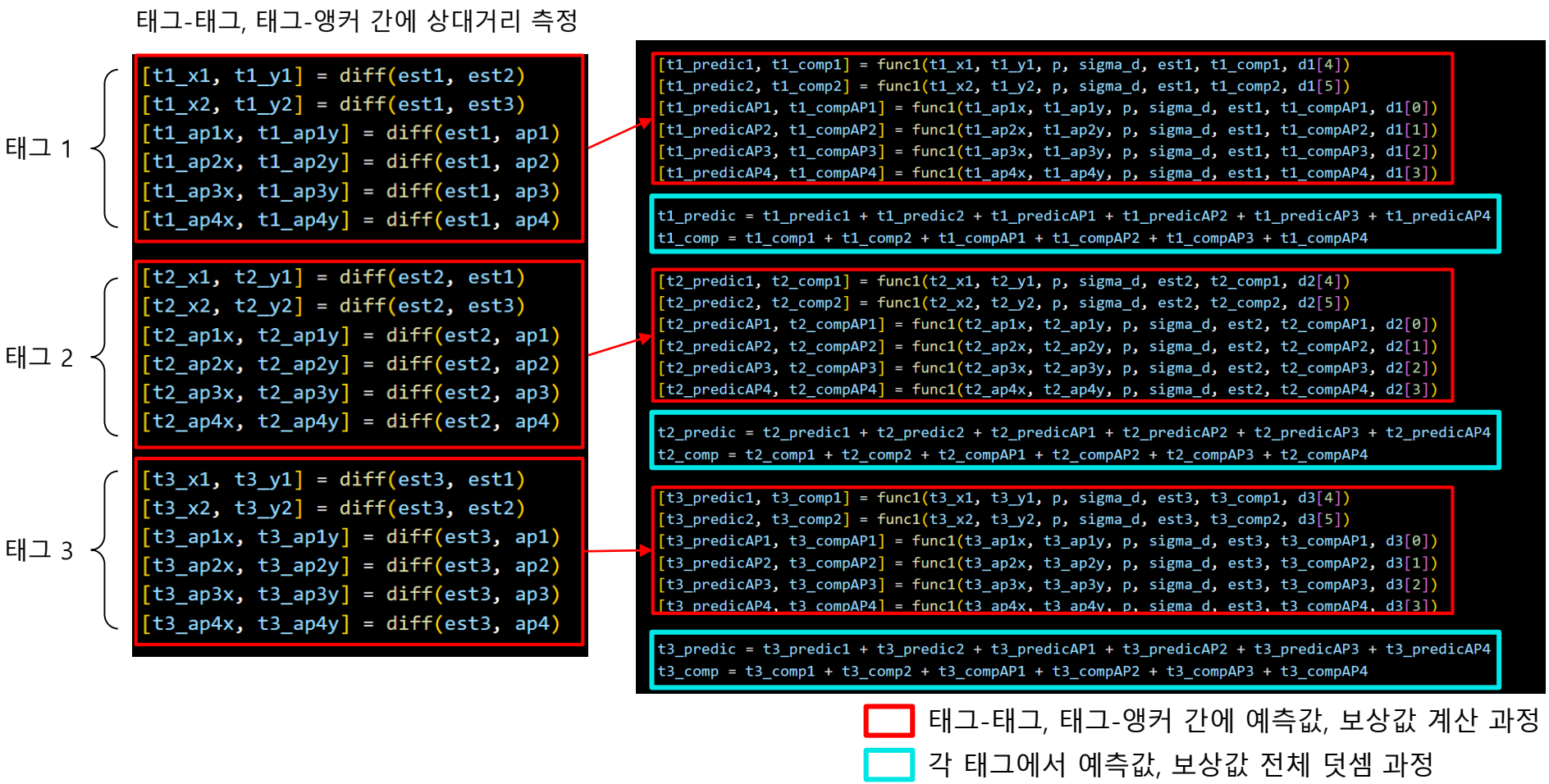
측정값 저장

```
d1= np.array([r.loop(), r1.loop(), r2.loop(), r3.loop(), r4.loop(), r5.loop()]) # 태그 1번 거리 측정값 저장
```

IV. 스크립트 구성

5. ADMM 기반 협력측위 알고리즘

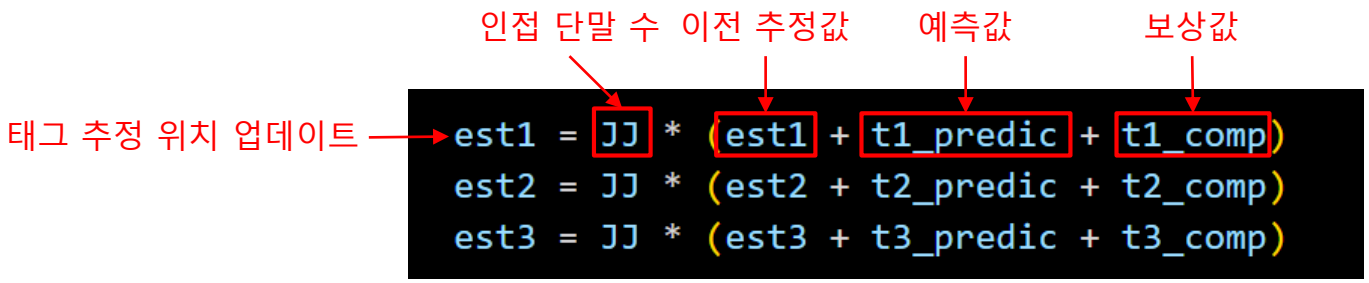
➢ 예측 단계 (prediction step), 보상 단계는 하나의 함수에서 수행되어 결과 출력됨



IV. 스크립트 구성

6. ADMM 기반 협력측위 알고리즘

➢ 기대 단계 (expectation step)에서 예측값, 보상값, 인접 단말 수에 대한 각 태그의 추정 위치를 업데이트함



➢ 협력측위 알고리즘 반복 횟수에 마다 추정 위치 저장 수행

```
aa[0, i] = est1[0]
aa[1, i] = est1[1]
```

반복 횟수 마다 추정 위치 저장

```
bb[0, i] = est2[0]
bb[1, i] = est2[1]

cc[0, i] = est3[0]
cc[1, i] = est3[1]
```

V. 협력측위 동작

1. UWB 단말 전원 공급

➢ 스크립트 실행 전, 태그와 앵커는 실험할 공간에 구성하여 전원을 공급해줌

<UWB 앵커 전원 공급 상태>

<UWB 태그 전원 공급 상태>

전원 공급 시 LED 점등

2. 스크립트 실행 및 상태 확인

➢ 터미널 창에 "python3 스크립트 이름.py"을 입력 시 협력측위가 동작하며, 터미널에서 측위 결과 확인 가능