8월4주차 (2024.08.19~2024.08.25)

💡 **오프라인 회의 일정**

- **장소**: 정보대 7호관 3층 & 취창업라운지
- **일정**:
  - 2024.08.20 (화)
    - 11:20 ~ 13:00 (1h 40m) - 수진
    - 13:00 ~ 16:40 (3h 40m) - 지원, 지윤
    - 15:00 ~ 16:00 (1h) - 다은
  - 2024.08.22 (목)
    - 09:00 ~ 16:00 (5h 30m)

데이터 수집 & 테스트 전 사전 정리

- **데이터 수집 진행 계획**

  → 사전 개발한 앱을 통해 7호관(정보기술대학) 3층 데이터 수집 진행

  <1차 데이터 수집>

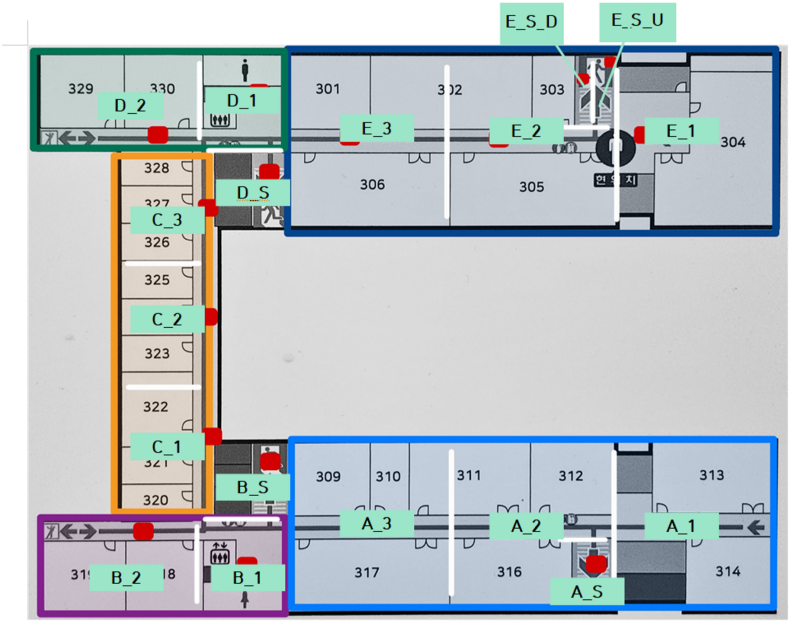
A 구역 (A_1 / A_2 / A_3 / A_S)	B, D 구역 (B_1 / B_2 / D_1 / D_2 / D_S)	C 구역 (C_1 / C_2 / C_3 / B_S)	E 구역 (E_1 / E_2 / E_3 / E_S_D / E_S_U)
수진	지원	지윤	다은

  - **데이터 수집 시간** : 각 구역 별 10분 / 총 180분
  - **학습 데이터**: 세부 구역 내에서 이동하며 데이터 수집(ex A_1구역 내에서 이동)

  <2차 데이터 수집>

A 구역 (A_1 / A_2 / A_3)	B, D 구역 (B_1 / B_2 / D_1 / D_2)	C 구역 (C_1 / C_2 / C_3)	E 구역 (E_1 / E_2 / E_3)
다은	수진	지원	지윤

- **데이터 수집 시간** : 각 구역 별 15분(계단 제외) / 총 195분
- **학습 데이터**: 세부 구역 내에서 이동하며 데이터 수집(ex A_1구역 내에서 이동)
- **테스트 데이터**: 동일 구역내에서 세부구역 간 이동하며 데이터 수집 (ex A구역에서 이동)



( ) - S : (구역)의 S(계단)
E-S-D : E 구역의 - S(계단)의 - D(Down - 아래층(2층 방향 반층 내려간 오른쪽 벽))
E-S-U : E 구역의 - S(계단)의 - U(Up - 위층(4층 방향 반층 올라간 오른쪽 벽)

- **테스트 진행**
  - **사용 모델** : Random Forest
  - **선정이유**: 아래의 논문을 참고하여 테스트한 결과 랜덤 포레스트 모델의 예측 정확도가 가장 높았음
  - 전수영, 이대국 and 주아림. (2023). RSSI 데이터 기반 머신러닝 실내 측위 연구. Journal of The Korean Data Analysis Society, 25(6), 2159-2170.

Table 3. Results of localization(BUILDING-FLOOR) using latitude and longitude

Machine learning classifier	Precision	Recall	F1-score	Accuracy
SVM	0.21	0.33	0.24	0.33
DecisionTree	0.73	0.72	0.71	0.72
ExtraTrees	0.73	0.71	0.70	0.73
RandomForest	**0.81**	**0.76**	**0.76**	**0.77**
KNN	0.71	0.71	0.71	0.71

1차 데이터 테스트

1차 데이터 수집 결과

- Total Data: 6,200개
- Train Data: 4,960개
- Val Data: 1,240개

```
전체 Train 데이터 수: 4960
전체 Validation 데이터 수: 1240

Zone 'A_1' - Train 데이터 수: 177, Validation 데이터 수: 44
Zone 'A_2' - Train 데이터 수: 211, Validation 데이터 수: 53
Zone 'A_3' - Train 데이터 수: 227, Validation 데이터 수: 57
Zone 'B_1' - Train 데이터 수: 410, Validation 데이터 수: 102
Zone 'B_2' - Train 데이터 수: 292, Validation 데이터 수: 73
Zone 'B_S' - Train 데이터 수: 363, Validation 데이터 수: 91
Zone 'C_1' - Train 데이터 수: 232, Validation 데이터 수: 58
Zone 'C_2' - Train 데이터 수: 286, Validation 데이터 수: 72
Zone 'C_3' - Train 데이터 수: 246, Validation 데이터 수: 62
Zone 'D_1' - Train 데이터 수: 306, Validation 데이터 수: 76
Zone 'D_2' - Train 데이터 수: 540, Validation 데이터 수: 135
Zone 'D_S' - Train 데이터 수: 306, Validation 데이터 수: 77
Zone 'E_1' - Train 데이터 수: 320, Validation 데이터 수: 80
Zone 'E_2' - Train 데이터 수: 271, Validation 데이터 수: 67
Zone 'E_3' - Train 데이터 수: 275, Validation 데이터 수: 69
Zone 'E_S_D' - Train 데이터 수: 213, Validation 데이터 수: 53
Zone 'E_S_U' - Train 데이터 수: 181, Validation 데이터 수: 45
Zone 'nan' - Train 데이터 수: 104, Validation 데이터 수: 26

모델 검증 정확도: 0.3371
```

1차 데이터 모델 테스트 결과

- 세부 구역이 아닌, 큰 구역에 대한 테스트 데이터로 사전 모델 테스트 실험
- **모델 정확도**: 0.32

  → 학습데이터 수가 부족해 정확도가 낮게 나옴.

```csv
No.,TimeStamp,MAC Address,RSSI,Beacon_Encoded,Predicted_Zone_Encoded,Predicted_Zone
1,16:53:00,60:98:66:32:8E:28,-94,2,6,C_1
2,16:53:04,60:98:66:2F:CF:9F,-80,0,9,D_1
3,16:53:04,60:98:66:32:AA:F8,-93,5,6,C_1
4,16:53:06,60:98:66:2F:CF:9F,-76,0,7,C_2
5,16:53:07,A0:6C:65:99:DB:7C,-97,17,5,B_S
6,16:53:08,60:98:66:30:A9:6E,-69,1,7,C_2
7,16:53:09,60:98:66:32:CA:74,-74,12,6,C_1
8,16:53:09,A0:6C:65:99:DB:7C,-95,17,7,C_2
9,16:53:10,60:98:66:30:A9:6E,-74,1,7,C_2
10,16:53:10,60:98:66:2F:CF:9F,-75,0,10,D_2
11,16:53:11,60:98:66:32:CA:74,-70,12,7,C_2
12,16:53:15,60:98:66:32:AA:F8,-84,5,11,D_S
13,16:53:17,60:98:66:2F:CF:9F,-72,0,10,D_2
14,16:53:17,60:98:66:32:AA:F8,-88,5,12,E_1
15,16:53:19,60:98:66:30:A9:6E,-91,1,2,A_3
16,16:53:20,60:98:66:32:CA:59,-96,11,11,D_S
17,16:53:21,60:98:66:30:A9:6E,-83,1,3,B_1
```

예측 방식 논의:

- 바로 예측하기보다 앞뒤 데이터를 활용한 예측이 효과적일 것으로 제안.
- 묶음 데이터를 활용하여 예측하는 방식 제안.

2차 데이터 수집

2차 데이터 수집 결과

- 1차 데이터 수집 + 2차 데이터 수집 + 13번 분실 비콘 데이터 제외
    - Total Data: 12,805개
    - Train Data: 10,190개
    - Val Data: 2,615개

```
==================================================
전체 요약 정보
전체 학습 윈도우 수: 2038
전체 검증 윈도우 수: 523
전체 학습 데이터 포인트 수: 10190
전체 검증 데이터 포인트 수: 2615
전체 모델 검증 정확도: 0.6711
==================================================
```

2차 데이터 모델 테스트 결과

- **모델 정확도**: 0.71

  → 구역마다 정확도는 다르나, 가장 높은 정확도는 0.71이다.

예측 방식 논의:

- **큐 방식이 유리한 경우**:
  - **신호의 변동성이 큰 환경**에서, 단일 RSSI 값이 일관되지 않거나 신뢰성이 떨어지는 경우.
  - **시간적 패턴**이나 **신호의 추세**를 반영하고자 하는 경우.
  - **예측의 안정성**이 중요하고, 약간의 지연을 감수할 수 있는 경우.
- **단일 RSSI 방식이 유리한 경우**:
  - **실시간 반응**이 중요한 경우.
  - 시스템의 **단순성**과 **빠른 처리**가 필요한 경우.
  - **신호가 안정적인 환경**에서, RSSI 값이 크게 변동하지 않는 경우.

▼ 큐 방식 코드

```
import os
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import joblib
from collections import deque


# ForestModelTrainer 클래스 정의
class ForestModelTrainer:
    def __init__(self, folder_path, result_folder_path, wi
        self.folder_path = folder_path
        self.result_folder_path = result_folder_path
```

```
        self.model = None
        self.le_mac = LabelEncoder()
        self.le_zone = LabelEncoder()
        self.data = None

        # 단일 큐 초기화
        self.rssi_queue = deque(maxlen=window_size)
        self.window_size = window_size  # 큐의 크기, 즉 사용히

    def load_data(self):
        df_list = []
        for file in os.listdir(self.folder_path):
            if file.endswith('.csv'):
                file_path = os.path.join(self.folder_path,
                df = pd.read_csv(file_path)
                df_list.append(df)
        self.data = pd.concat(df_list, ignore_index=True)
        print("데이터 로드 완료")

    def update_rssi_queue(self, mac_address, rssi_value):
        self.rssi_queue.append((mac_address, rssi_value))
        return list(self.rssi_queue)

    def encode_labels(self):
        self.data['Beacon_Encoded'] = self.le_mac.fit_tran
        self.data['Zone_Encoded'] = self.le_zone.fit_trans
        print("레이블 인코딩 완료")

        self.data['RSSI_Queue'] = self.data.apply(
            lambda row: self.update_rssi_queue(row['MAC Ad
        )

        self.data = self.data[self.data['RSSI_Queue'].appl

        for i in range(self.window_size):
            self.data[f'RSSI_{i+1}'] = self.data['RSSI_Que

        print("레이블 인코딩 및 RSSI 피처 생성 완료")
```

```
    def split_data(self):
        rssi_columns = [f'RSSI_{i+1}' for i in range(self.
        X = self.data[rssi_columns + ['Beacon_Encoded']]
        y = self.data['Zone_Encoded']
        X_train, X_test, y_train, y_test = train_test_spli
        print("데이터 분할 완료")
        return X_train, X_test, y_train, y_test

    def train_model(self, X_train, y_train):
        self.model = RandomForestClassifier(n_estimators=1
        self.model.fit(X_train, y_train)
        print("모델 학습 완료")

    def predict_zone(self, mac_address, rssi_value):
        rssi_sequence = self.update_rssi_queue(mac_address
        if len(rssi_sequence) == self.window_size:
            rssi_features = [rssi for _, rssi in rssi_sequ
            beacon_encoded = self.le_mac.transform([mac_ad
            features = rssi_features + [beacon_encoded]
            prediction = self.model.predict([features])
            zone_predicted = self.le_zone.inverse_transfor
            return zone_predicted
        else:
            return None  # 데이터가 충분하지 않으면 예측하지 않음

    def evaluate_model(self, X_test, y_test):
        score = self.model.score(X_test, y_test)
        print(f"모델 검증 정확도: {score:.4f}")
        return score

    def save_results(self, X_train, X_test, y_train, y_tes
```

```
        os.makedirs(self.result_folder_path, exist_ok=True
        result_file_path = os.path.join(self.result_folder
        with open(result_file_path, 'w') as f:
            f.write(f"전체 Train 데이터 수: {len(X_train)}\n"
            f.write(f"전체 Validation 데이터 수: {len(X_test)

            for zone in sorted(y_train.unique()):
                train_count = sum(y_train == zone)
                val_count = sum(y_test == zone)
                zone_name = self.le_zone.inverse_transform
                f.write(f"Zone '{zone_name}' - Train 데이터

            f.write(f"\n모델 검증 정확도: {score:.4f}\n")
        print("결과 저장 완료")

    def save_model_and_encoders(self):
        model_file_path = os.path.join(self.result_folder_
        joblib.dump(self.model, model_file_path)

        encoder_mac_file_path = os.path.join(self.result_f
        encoder_zone_file_path = os.path.join(self.result_
        joblib.dump(self.le_mac, encoder_mac_file_path)
        joblib.dump(self.le_zone, encoder_zone_file_path)
        print("모델과 인코더 저장 완료")

    def run(self):
        self.load_data()
        self.encode_labels()
        X_train, X_test, y_train, y_test = self.split_data
        self.train_model(X_train, y_train)
        score = self.evaluate_model(X_test, y_test)
        self.save_results(X_train, X_test, y_train, y_test
        self.save_model_and_encoders()

def main():
    folder_path = "./_2_7호관3층_train"
    result_folder_path = "./_2_7호관3층_result/forest"
```

```
    trainer = ForestModelTrainer(folder_path, result_folde
    trainer.run()

if __name__ == "__main__":
    main()
```

다음 주 계획

- 데이터 수집 결과에 따른 추가 테스트 및 모델 개선.
- 큐 방식과 단일 RSSI 방식의 실험 결과를 비교하여 최적화된 예측 모델 구축.