

# 오픈 소스 (Open Source)를 활용한 교내 (PNU)

## LoRaWAN 네트워크 망 구축 및 운용



부산대학교 전기컴퓨터공학부 정보컴퓨터공학전공

School of Electrical and Computer Engineering, Computer Engineering Major

Pusan National University

지도교수 : 김종덕

팀 명 : PBO

인 원 : 201624481 박윤형  
201624517 오세영

# 목차

1. 요구조건 및 제약사항
2. 설계 상세화
3. 과제 추진 계획
4. 구성원 별 진척도
5. 중간 결과

## 1. 요구조건 및 제약사항

### 1.1 요구조건

부산대학교 내부의 비어 있는 주차장 정보를 제공하는 어플리케이션을 제작한다.

- ChirpStack을 이용하여 LoRaWAN 네트워크를 구축
- LoRaWAN 데이터를 수신하기 위한 Web서버 구축
- 주차장 정보 저장을 위한 데이터베이스 구축
- 교내에 LoRa 단말 설치
- 센서 데이터를 LoRa모듈을 통해 네트워크로 전송
- 주차장 정보 제공을 위한 어플리케이션 구현

### 1.2 제약사항

#### 1.2.1 기존 제약사항 및 대책

- IoT에 사용되는 WiFi, LTE-M, 5G에 비해 데이터 전송 속도가 매우 느리다.
  - End node에서 주차 칸의 사용 유무만 데이터로 전송하여 매우 작은 크기 이므로 전송속도가 느려도 원활하게 통신할 수 있다.
- 주차 칸 마다 End node를 설치하여 데이터를 전송해야 해서 신호 간에 간섭 문제가 발생할 가능성이 있다.
  - End node마다 30초의 주기로 데이터를 보내게 하므로, 30초를 배분해서 end node 신호가 겹치지 않게 하여 해결할 수 있다.

#### 1.2.2 추가 제약사항 및 대책

- LoRa모듈을 통한 센서 데이터 전송이 확인되어야 만 교내에 LoRa단말을 실제로 설치하여 테스트할 수 있다.
  - 기존 과제 추진 계획보다 원활히 진행되어 더 빠르게 교내에 LoRa단말을 구축하여 테스트를 진행할 예정.

## 2. 설계 상세화

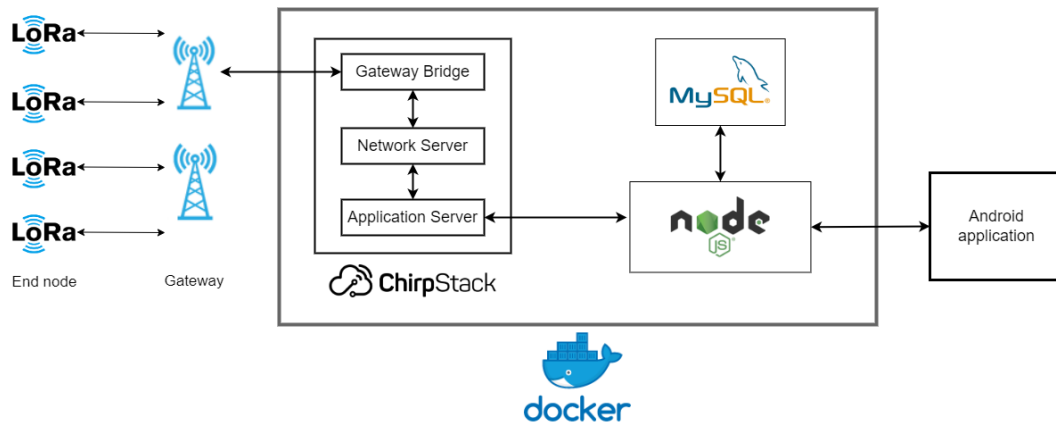


그림1. 네트워크 구조

### 2.1 ChirpStack 서버

- Gateway bridge, network server, application server를 각각 구현
- Gateway를 통해 end node로부터 받은 base64로 인코딩 된 데이터를 서버 내부에서 javascript codec기능으로 디코딩 하여 node.js에 전송

### 2.2 Node.js

- ChirpStack application server에서 받은 post데이터를 저장
- Post데이터를 추출하여 DB내의 데이터 업데이트
- DB내의 데이터를 json형식으로 출력
- ChirpStack과 어플리케이션이 web server로 요청을 하는데 동작을 다르게 하기 위해 url의 path로 동작을 구분  
path가 /chirpstack일 경우 post 요청을 처리하고 /app일 경우 query로 들어온 건물번호의 데이터를 데이터베이스로부터 받아와 출력

### 2.3 End Node

- 초음파 센서 모듈의 trig에 신호를 주고 echo로 신호를 받는 함수를 구현
- 기존에 payload에 건물번호 및 복수의 센서 측정데이터를 6bytes 길이의 정보로 보낼 예정이었지만 최적화를 위해 end node하나에 하나의 센서 측정데이터만 보내고, 건물번호는 서버에서 설정하도록 해 1byte의 정보만 보내도록 할 예정

### 2.4 안드로이드 어플리케이션

- Node.js로부터 제공된 데이터를 기반으로 앱에 주차장 정보를 표시
- 메인 화면이 따로 없어 추가 예정
- 주차장 별로 모양이 다르기 때문에 레이아웃을 추가할 예정

### 3. 과제 추진 계획

5월		6월					7월				8월				9월				
3주	4주	1주	2주	3주	4주	5주	1주	2주	3주	4주	1주	2주	3주	4주	1주	2주	3주	4주	5주
LoraWAN 및 chirpstack 관련 지식 스터디																			
				Chirpstack 서버 구축 및 lora gateway 연결															
						Web 서버, mysql 서버 구축 및 연동													
							어플리케이션 개발												
								초음파 센서 연결											
										중간보고									
											Lora gateway 및 end node 설치								
														안정성 및 성 능 평가					
														오류 수정 및 문제점 파악					
														최종 보고서 작성 및 발표 준비					

### 4. 구성원별 진척도

박윤형	<b>Docker를 이용한 서버 구축 및 각 서버 연동</b> <ul style="list-style-type: none"> <li>- Chirpstack 서버 구축</li> <li>- node.js 서버 구축</li> <li>- mysql 서버 구축</li> </ul>
오세영	<b>디바이스 펌웨어 작성</b> <ul style="list-style-type: none"> <li>- gateway 설정</li> <li>- lora 통신 구현</li> <li>- end node에 초음파센서 연결</li> </ul> <b>안드로이드 어플리케이션 개발</b>

## 5. 중간 결과

### 5.1 chirpstack 서버 구축

Chirpstack 서버를 구축 및 배포할 수 있는 도커 이미지를 다운받아 컨테이너화 한다.

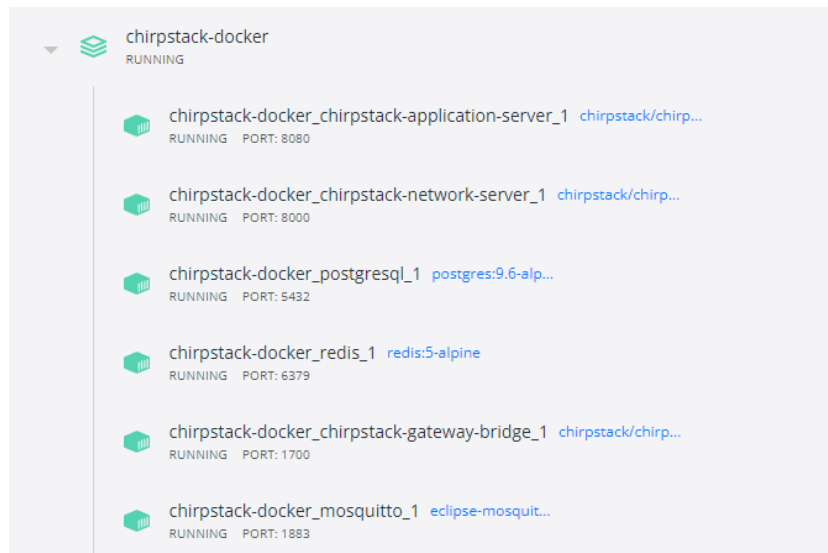


그림2. chirpstack서버의 컨테이너

Chirpstack-docker를 실행해서 chirpstack 서버를 배포한다.

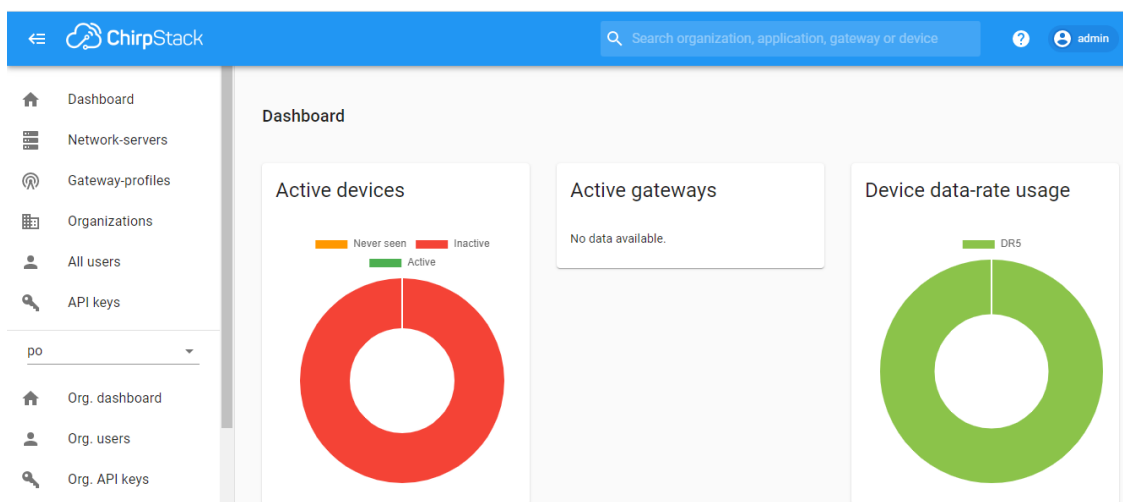


그림3. Chirpstack 대쉬보드

chirpstack서버에서 사용할 device profile과 application profile, service profile을 생성한다. profile을 생성하면서 lora 디바이스에서 사용할 EUI와 app session key, network session key를 생성한다.

The screenshot shows the 'Device-profiles / gateway1' configuration page. It has tabs for GENERAL, JOIN (OTAA / ABP), CLASS-B, CLASS-C, CODEC, and TAGS. The GENERAL tab is active, showing the following fields:

- Device-profile name \***: gateway1 (Description: A name to identify the device-profile.)
- LoRaWAN MAC version \***: 1.0.3 (Description: The LoRaWAN MAC version supported by the device.)
- LoRaWAN Regional Parameters revision \***: RP002-1.0.3 (Description: Revision of the Regional Parameters specification supported by the device.)
- ADR algorithm \***: Default ADR algorithm (LoRa only) (Description: The ADR algorithm that will be used for controlling the device data-rate.)

그림4. Device profile

gateway에서 chirpstack으로 데이터를 보낼 때 Base64로 인코딩해서 보내기 때문에 서버에서 다시 16진수로 디코딩 하는 코덱을 정의해야 한다.

```
function toHexString(bytes) {
  return bytes.map(function(byte) {
    return ("00" + (byte & 0xFF).toString(16)).slice(-2)
  }).join('')
}

function Decode(fPort, bytes) {

  var tohex = toHexString(bytes);
  //var toascii = hex_to_ascii(tohex);

  return {"mydata": tohex};
}
```

그림5. Base64에서 16진수로 디코딩하는 함수



## 5.2 end node 및 gateway 설정

end node에 STM32CubeIDE로 I-CUBE-LPWAN 펌웨어를 올리고 chirpstack에서 생성한 EUI와 app session key, network session key를 "LoRaWAN > App > se-identity.h" 파일에 적어준다.

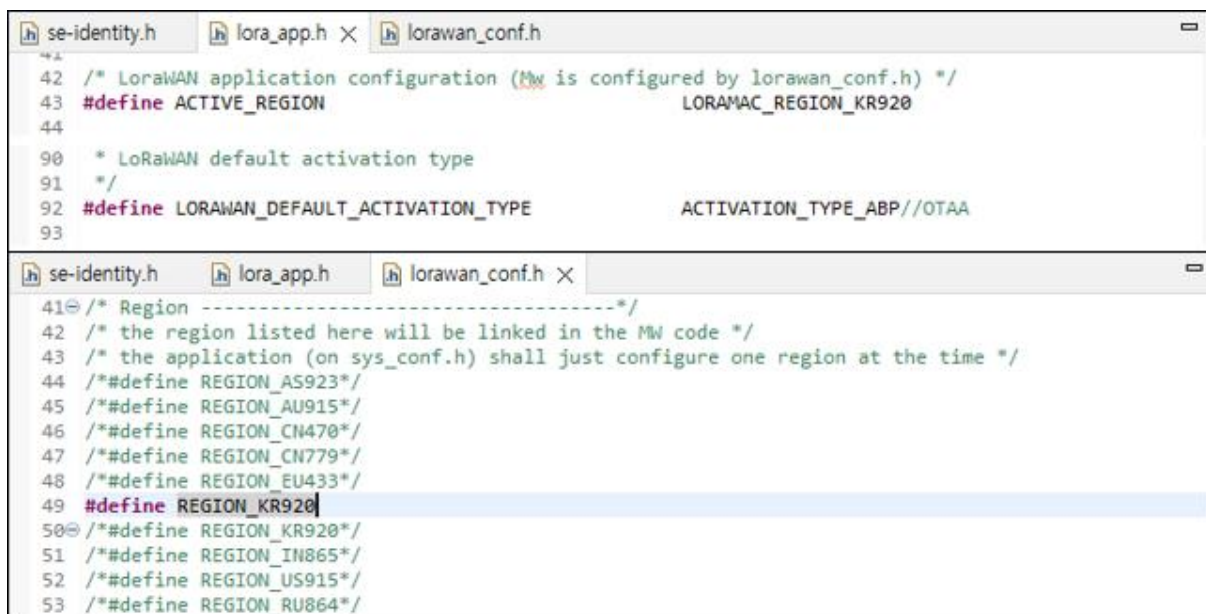
```

)/*!
 * Application session key
 */
#define LORAWAN_APP_S_KEY                97,0F,8A,A4,6A,AC,3D,85,08,53,79,13,F8,C0,8E,ED

)/*!
 * Forwarding Network session key
 */
#define LORAWAN_NWK_S_KEY                79,C3,CF,FF,29,00,91,2A,FB,82,DC,F7,5A,FC,FF,AC

```

그림6. 세션 키 설정



```

42 /* LoRaWAN application configuration (Mw is configured by lorawan_conf.h) */
43 #define ACTIVE_REGION                LORAWAN_REGION_KR920
44
90 * LoRaWAN default activation type
91 */
92 #define LORAWAN_DEFAULT_ACTIVATION_TYPE    ACTIVATION_TYPE_ABP//OTAA
93
41 /* Region -----*/
42 /* the region listed here will be linked in the MW code */
43 /* the application (on sys_conf.h) shall just configure one region at the time */
44 /*#define REGION_AS923*/
45 /*#define REGION_AU915*/
46 /*#define REGION_CN470*/
47 /*#define REGION_CN779*/
48 /*#define REGION_EU433*/
49 #define REGION_KR920
50 /*#define REGION_KR920*/
51 /*#define REGION_IN865*/
52 /*#define REGION_US915*/
53 /*#define REGION_RU864*/

```

그림7. 지역 설정

gateway에 STM32CubeProgrammer로 펌웨어를 올린 후 터미널로 설정을 한다. AT+MAC 명령어로 gateway의 MAC주소를 설정하고, AT+CH 명령어로 지역을 설정해준다. AT+PKTFWD 명령어로 gateway bridge 주소를 설정한다.

```
AT+MAC=0080E1015480
+MAC: 00:80:E1:01:54:80

AT+CH=KR920
+CH: KR920
+CH: 0, 922100000, A, SF7/SF12, BW125KHz (LORA_MULTI_SF)
+CH: 1, 922300000, A, SF7/SF12, BW125KHz (LORA_MULTI_SF)
+CH: 2, 922500000, A, SF7/SF12, BW125KHz (LORA_MULTI_SF)
+CH: 3, 922700000, A, SF7/SF12, BW125KHz (LORA_MULTI_SF)

AT+PKTFWD=nan1.cloud.thethings.network,1700,1700
+PKTFWD: nan1.cloud.thethings.network, 1700, 1700
AT+RESET
+RESET: OK
Restarting...
```

그림8. 시리얼통신으로 게이트웨이 설정

### 5.3 end node와 gateway 간의 통신

End node에서 gateway로 데이터를 보내는 함수를 수정하여 센서에서 측정된 데이터를 mydata라는 이름으로 보내도록 했다.

```
static void SendTxData(void)
{
    UTIL_TIMER_Time_t nextTxIn = 0;

    uint32_t spacecheck = ReadJlt();
    uint32_t tempdata = 0x0d00000000;
    tempdata = tempdata + spacecheck + pow(2,7);
    AppData.Buffer[i++] = (uint8_t)0x01;
    AppData.Buffer[i++] = (uint8_t)0x39;
    AppData.Buffer[i++] = (uint8_t)tempdata;
    AppData.Buffer[i++] = (uint8_t)0x0d00000000;
    AppData.Buffer[i++] = (uint8_t)0x0d00000000;
    AppData.Buffer[i++] = (uint8_t)0x0d00000000;

    AppData.BufferSize = i;

    if (LORAMAC_HANDLER_SUCCESS == LmHandlerSend(&AppData, LORAWAN_DEFAULT_CONFIRMED_MSG_STATE, &nextTxIn, false))
    {
        APP_LOG(TS_ON, VLEVEL_L, "SEND REQUEST\r\n");
    }
    else if (nextTxIn > 0)
    {
        APP_LOG(TS_ON, VLEVEL_L, "Next Tx in : ~Xd second(s)\r\n", (nextTxIn / 1000));
    }

    /* USER CODE END SendTxData_1 */
}
```

그림9. 데이터 전송 함수



그림10. End node로부터 chirpstack으로 원하는 데이터 전송

#### 5.4 end node에 초음파 센서(HC-SR04) 연결

데이터를 수집하기 위해 먼저 end node에 센서를 연결할 필요가 있다. 센서의 trig핀과 echo핀을 각각 output mod, input mod로 연결하고 아래의 코드를 사용했다. 거리 측정 코드는 HC-SR04의 datasheet를 참고해서 만들었다. 정확성을 위해 기존의 라이브러리에 존재하던 ms단위의 딜레이 함수를 사용하지 않고  $\mu$ s 단위의 딜레이 함수를 새로 만들어 사용했다.

```

uint32_t ReadUltr() {
    uint32_t echo = 0, distance = 0;
    BSP_ULTR_TRIG_Off();
    HAL_MSDelay(2);
    BSP_ULTR_TRIG_On();
    HAL_MSDelay(10);
    BSP_ULTR_TRIG_Off();
    HAL_MSDelay(100);
    //echo = BSP_ULTR_ECHO_Read();
    while(!echo) {
        echo = BSP_ULTR_ECHO_Read();
    }
    while(BSP_ULTR_ECHO_Read()) {
        distance = distance + 1;
    }
    distance = (distance + 48) / 2500;
    APP_LOG(TS_OFF, VLEVEL_M, "Ultr##### distance= %d\r\n", distance);

    if (distance <= 60) return 1;
    else return 0;
}

```

그림11. 거리 측정 코드

```

Tera Term - [disconnected] VT
File Edit Setup Control Window Help

##### ===== MCPS-Indication =====
60s078:temp= 32

##### distance= 3
60s097:TX on freq 922100000 Hz at DR 5
60s114:SEND REQUEST
60s166:MAC txDone
61s149:RX_1 on freq 922100000 Hz at DR 5
61s220:MAC rxDone

##### ===== MCPS-Confirm =====
90s078:temp= 32

##### distance= 3
90s097:TX on freq 922300000 Hz at DR 5
90s114:SEND REQUEST
90s166:MAC txDone
91s149:RX_1 on freq 922300000 Hz at DR 5
91s220:MAC rxDone

##### ===== MCPS-Confirm =====
120s078:temp= 32

##### distance= 100
120s104:TX on freq 922300000 Hz at DR 5
120s121:SEND REQUEST
120s172:MAC txDone
121s156:RX_1 on freq 922300000 Hz at DR 5
121s227:MAC rxDone

```

그림12. 초음파 센서로 측정한 거리를 시리얼 통신으로 확인

## 5.5 node.js 서버 구축

Node.js 16.16.0버전을 기반으로 도커 파일을 작성했다. 3030포트로 외부에서도 접속할 수 있도록 했다.

```

EXPLORER
  LORAWAN
    .vscode
    chirpstack-docker
    database
    frontend
      node_app
        node_modules
        index.html
        JS main.js
        package-lock.json
        package.json
        Dockerfile
        README
  Dockerfile U
  JS main.js U

Dockerfile U
1 FROM node:16.16.0
2
3 RUN npm install -g http-server
4
5
6 WORKDIR /home/node/app
7
8 CMD ["http-server", "-p", "3030", "./public"]
9

```

그림13. Node.js의 dockerfile

web server의 path가 '/app'일 경우 query의 건물번호에 해당하는 데이터를 출력한다.



그림14. 웹서버의 path가 app일 때 313의 query 결과

web server의 path가 '/chirpstack'일 경우, chirpstack으로부터 보내진 post 요청을 처리한다. Post data중 device data를 추출해 데이터베이스에 저장한다.

## 5.6 데이터베이스 구축

데이터베이스를 구축하기 위해 mysql을 기반으로 한 docker file 작성

```
database > Dockerfile > ...
1 FROM mysql:8.0.29
2
3 ENV MYSQL_USER mysql_user
4 ENV MYSQL_PASSWORD 1234
5 ENV MYSQL_ROOT_PASSWORD 1234
6 ENV MYSQL_DATABASE lorawan
7
8 COPY ./scripts/ /docker-entrypoint-initdb.d/
9
```

그림15. Database docker file

주차장의 위치가 될 건물번호와 주차 칸 번호를 key로 가지는 table을 생성하고 임의의 데이터 생성

	building_number	parking_lot_number	occupied
▶	311	1	1
	311	2	0
	311	3	1
	311	4	1
	312	1	1
	312	2	0
	312	3	0

그림16. Database table

## 5.7 어플리케이션 개발

메인 화면으로 부산대 지도 위에 건물번호로 마커를 생성해 마커 터치 시 해당 건물의 주차장을 화면에 표시하도록 구현.

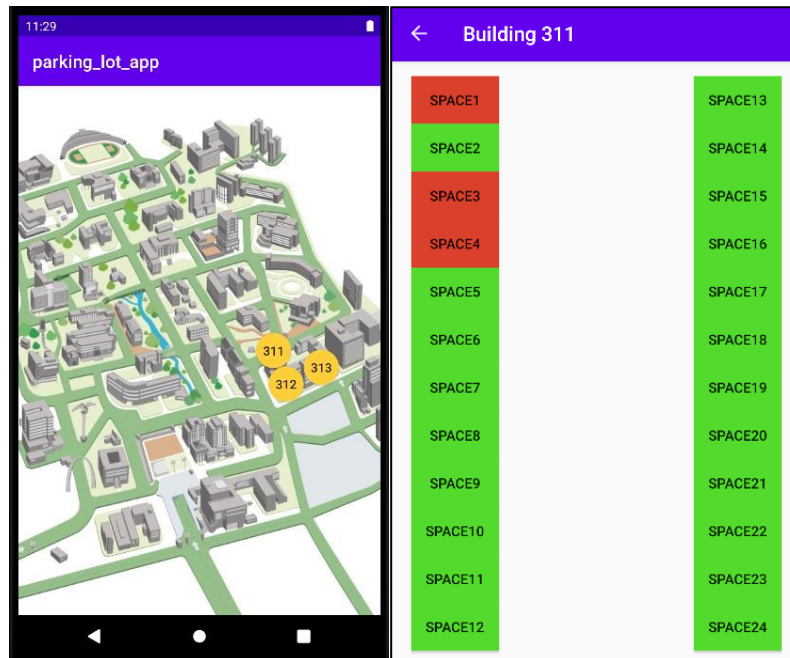


그림17. 좌-메인 화면, 우-주차장 화면