

IoT 제작

주 정 인

Open API

INDEX

01 진행 상황

02 최종 목적

03 UUID DataCollect

04 Hybrid WebApp

05 참조(진행상황 캡처)

01

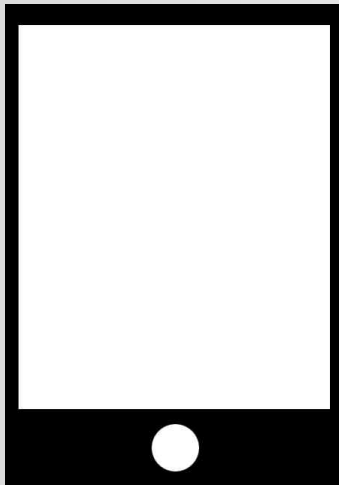
진행상황

| | | | | | | |
|-------------|------|----|----|----|-------------------|----|
| 9/16 | 17 | 18 | 19 | 20 | 21 | 22 |
| Testing LIB | | | | | | |
| 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| | | | | | | |
| 30 | 10/1 | 2 | 3 | 4 | 5 | 6 |
| | | | | | 캡스톤 보고서 (~11월) | |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| | | | | | | |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| | | | | | | |

| 구분 | 단계 | 내용 | 비고 |
|------|----|-------------------------|----|
| UUID | 1 | bluetooth 장치의 UUID를 군집화 | △ |
| APP | 2 | bluetooth connect LIB | ○ |
| | 3 | bluetooth list LIB | △ |
| | 4 | wifi connect LIB | ○ |
| | 5 | wifi list LIB | △ |
| UNO | 6 | bluetooth connect | △ |
| | 7 | wifi connect | △ |
| WEB | 8 | Log print | △ |

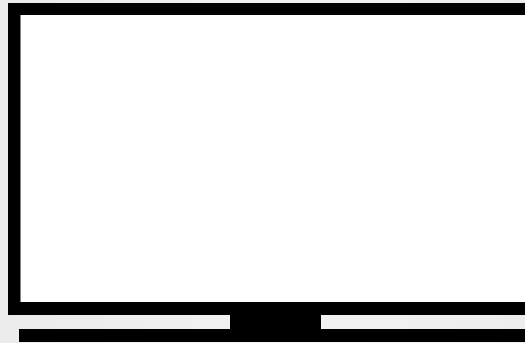
02

최 종 목 적



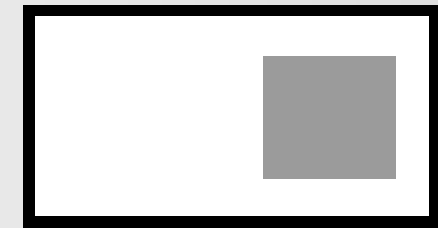
Smart Device

1. Bluetooth_connect
 2. Bluetooth_list
 3. WIFI_connect
 4. WIFI_list
 5. WEB_connect
- + Module_search
+ Voice_record_cmd
+ SNS_share



WEB SERVER

1. LOG_time
- + Module_list



IoT Platform

1. Bluetooth_connect
 2. WIFI_connect
 3. wire_PIN
- + LED_controll
+ Voice_record

» 편리한 개발 환경

function

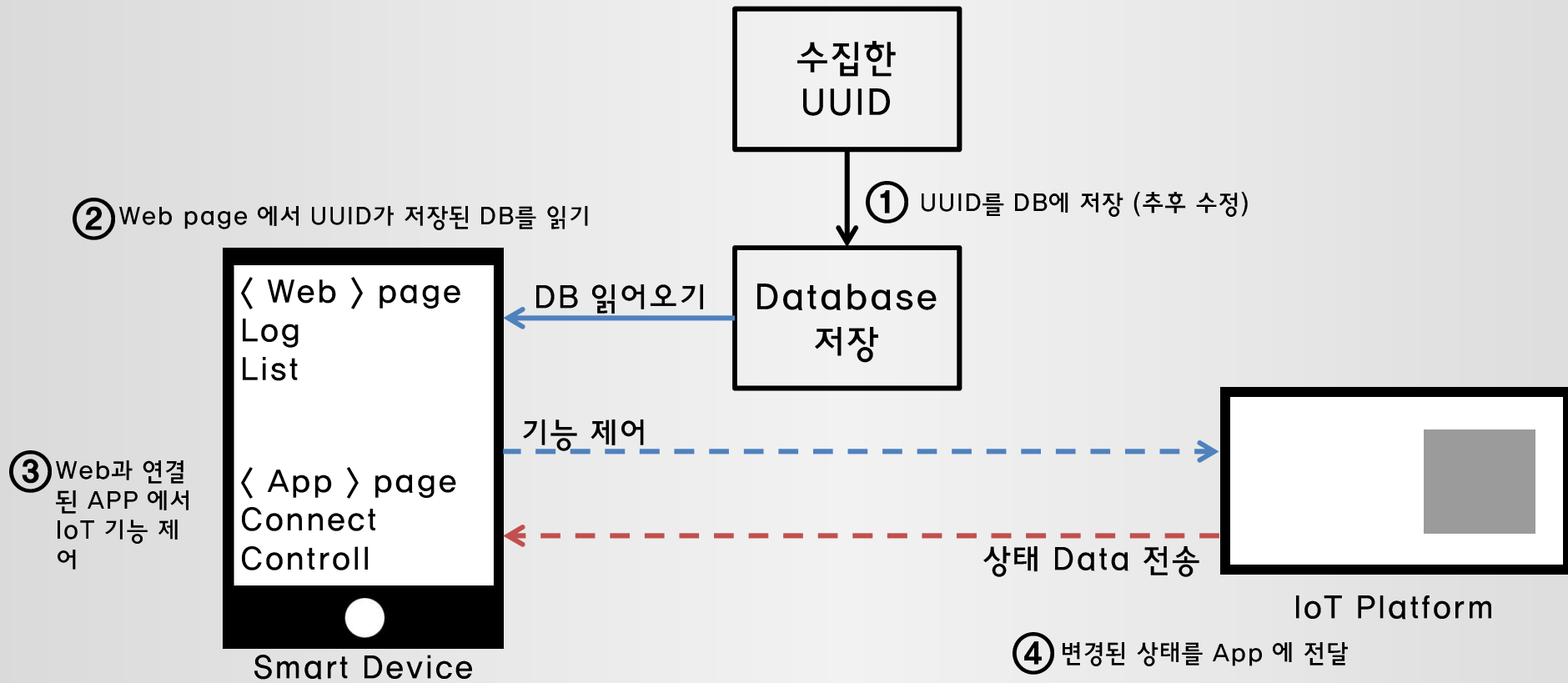
Bluetooth_connect
Bluetooth_list
WIFI_connect
WIFI_list
WEB_connect
LOG_time
Module_search
Voice_record_cmd
SNS_share
Module_list
LED_controll
Voice_record

Action

App과 IoT의 bluetooth 연결 요청
Device에 연결된 bluetooth 장비 목록 읽기
App과 IoT의 WIFI 연결 요청
Device에 연결 가능한 WIFI 목록 읽기
App 실행시 Web 연결
시간형식에 맞는 log data 읽기
Bluetooth와 wifi로 연결된 Module 탐색
녹음 기능 및 녹음 명령 전송
SNS 공유기능
APP 에 연결된 Module을 WEB에 출력
LED 제어
녹음 기능

03

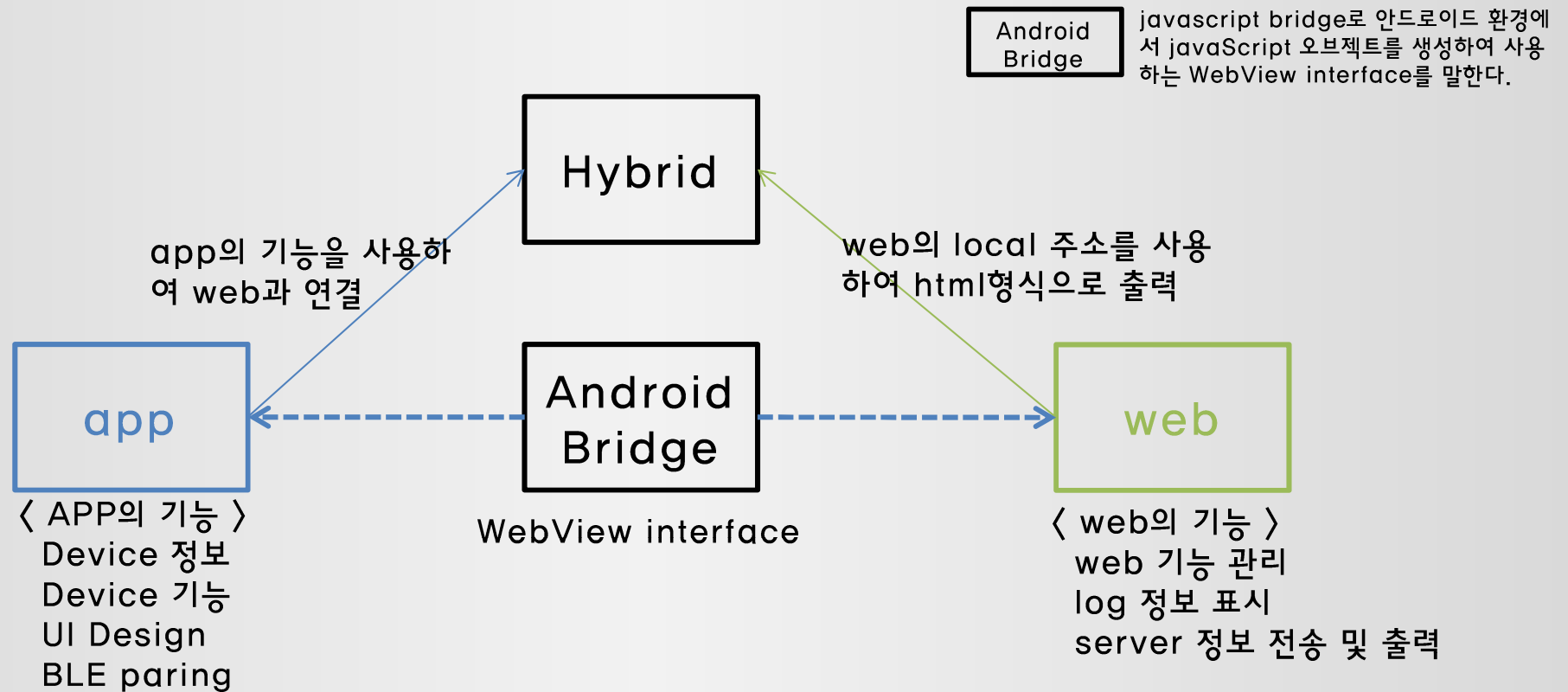
UUID Data Collect



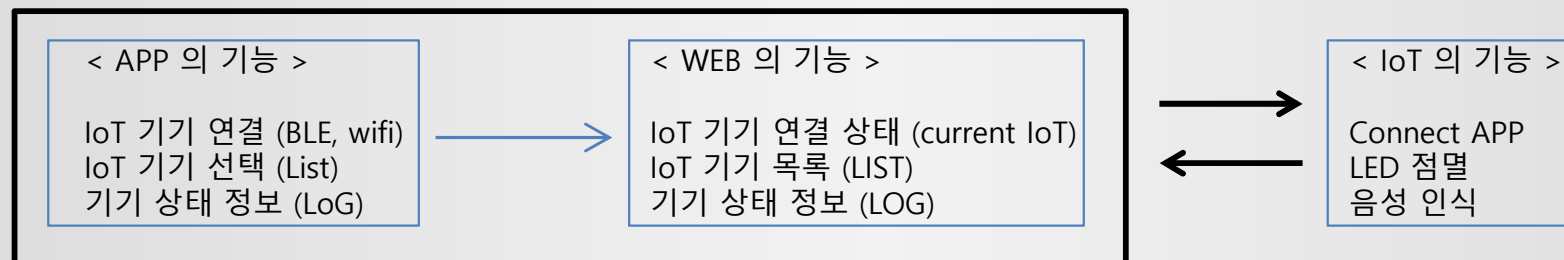
수집한 UUID를 기반으로 WebApp을 통해 IoT Control

04

Hybrid Application lib



➤ Android Bridge class 를 통해 app과 web의 데이터 교환



➡➡ App 하나로 web 기능을 사용, 관리 가능

05

참조 (LIB 사진)

```
void sendData() throws IOException
{
    String msg = myTextbox.getText().toString();
    if ( msg.length() == 0 ) return;

    msg += "\n";
    Log.d(msg, msg);
    mmOutputStream.write(msg.getBytes());
    myLabel.setText("Data Sent");
    myTextbox.setText(" ");
}
```

```
public void ErrorDialog(String text)
{
    if (activity.isFinishing()) return;

    FragmentManager fm = MainActivity.this.getSupportFragmentManager();
    MyDialogFragment alertDialog = MyDialogFragment.newInstance(ERROR_DIALOG, text);
    alertDialog.show(fm, tag: "");
}
```

```
public void DeviceDialog()
{
    if (activity.isFinishing()) return;

    FragmentManager fm = MainActivity.this.getSupportFragmentManager();
    MyDialogFragment alertDialog = MyDialogFragment.newInstance(DEVICES_DIALOG, text: "");
    alertDialog.show(fm, tag: "");
}
```

```
public void doConnect(BluetoothDevice device) {
    mmDevice = device;

    //Standard SerialPortService ID
    UUID uuid = UUID.fromString("00001101-0000-1000-8000-00805f9b34fb");

    try {
        // 4. 지정한 블루투스 장치에 대한 특정 UUID 서비스를 하기 위한 소켓을 생성합니다.
        // 여기선 시리얼 통신을 위한 UUID를 지정하고 있습니다.
        mmSocket = mmDevice.createRfcommSocketToServiceRecord(uuid);
        // 5. 블루투스 장치 검색을 중단합니다.
        mBluetoothAdapter.cancelDiscovery();
        // 6. ConnectTask를 시작합니다.
        new ConnectTask().execute();
    } catch (IOException e) {
        Log.e("", e.toString(), e);
        ErrorDialog("doConnect "+e.toString());
    }
}
```

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mWebView = (WebView) findViewById(R.id.webview);
    mTextView = (TextView) findViewById(R.id.textview);
    mButton = (Button) findViewById(R.id.button);
    mEditText = (EditText) findViewById(R.id.editText01);
    mStringHead = "Message from HTML ";

    // 웹뷰에서 자바스크립트실행가능
    mWebView.getSettings().setJavaScriptEnabled(true);
    // Bridge 인스턴스 등록
    mWebView.addJavaScriptInterface(new AndroidBridge(), "MainActivity");

    mWebView.loadUri("file:///android_asset/test.html"); // 로컬 HTML 파일 로드

    mWebView.setWebViewClient(new HelloWebViewClient()); // WebViewClient 지정

    mButton.setOnClickListener( (view) -> {
        mWebView.loadUri("javascript:setMessage(""+mEditText.getText()+"");
    });
}

```

```

private class AndroidBridge {
    public void setMessage(final String arg) { // must be final
        handler.post(() -> {
            Log.d( tag: "MainActivity", msg: "setMessage("+arg+")");
            mTextView.setText(mStringHead+arg);
        });
    }
}

```

```

public static class Basic {
    public static UUID service = UUID.fromString("0000fee0-0000-1000-8000-00805f9b34fb");
    public static UUID batteryCharacteristic = UUID.fromString("00000006-0000-3512-2118-0009af100700");
}

public static class AlertNotification {
    public static UUID service = UUID.fromString("00001802-0000-1000-8000-00805f9b34fb");
    public static UUID alertCharacteristic = UUID.fromString("00002a06-0000-1000-8000-00805f9b34fb");
}

public static class HeartRate {
    public static UUID service = UUID.fromString("0000180d-0000-1000-8000-00805f9b34fb");
    public static UUID measurementCharacteristic = UUID.fromString("00002a37-0000-1000-8000-00805f9b34fb");
    public static UUID descriptor = UUID.fromString("00002902-0000-1000-8000-00805f9b34fb");
    public static UUID controlCharacteristic = UUID.fromString("00002a39-0000-1000-8000-00805f9b34fb");
}

public static class Information {
    public static UUID service = UUID.fromString("0000fee0-0000-1000-8000-00805f9b34fb");
    public static UUID Characteristic = UUID.fromString("00000007-0000-3512-2118-0009af100700");
    public static UUID descriptor = UUID.fromString("00002902-0000-1000-8000-00805f9b34fb");
}

```