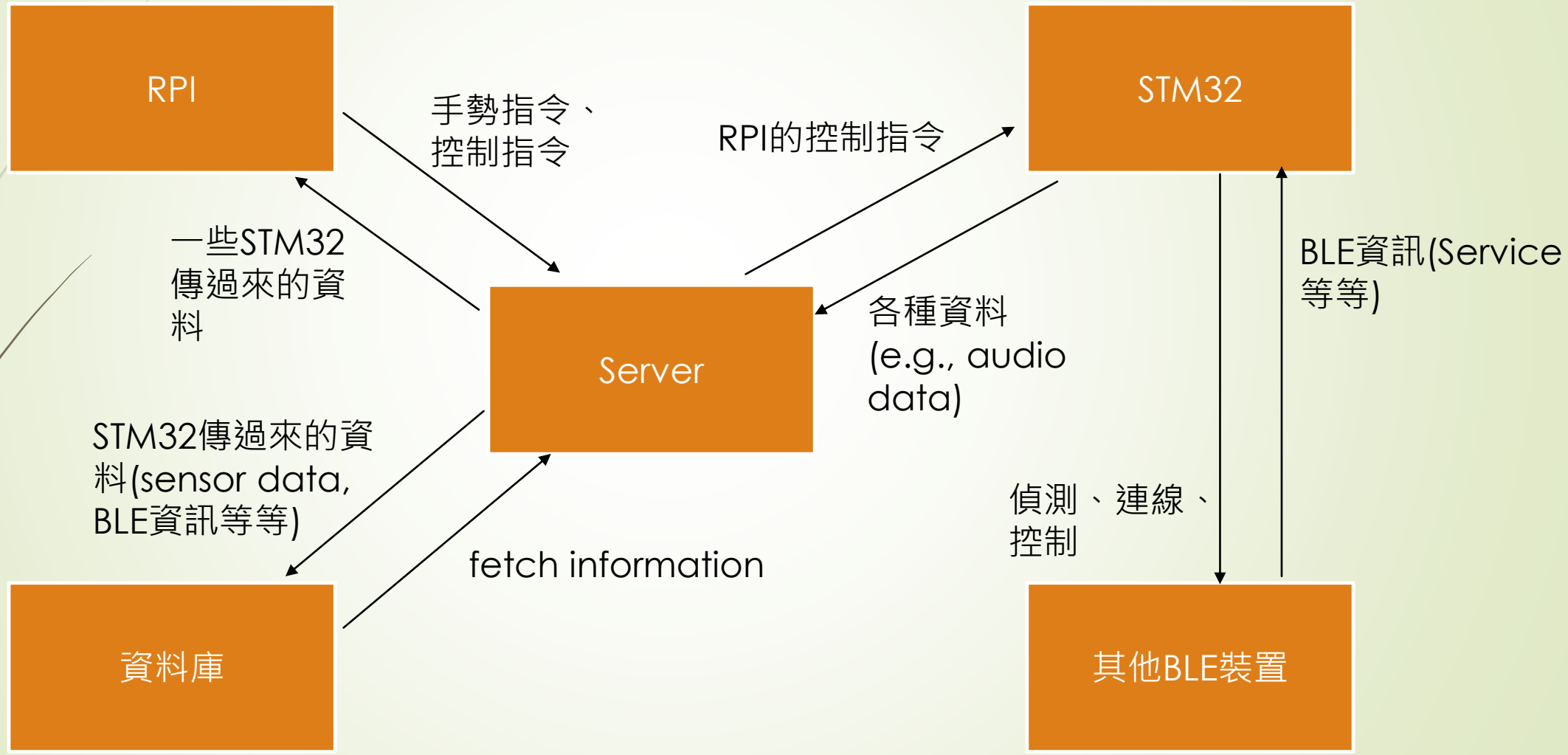
05/13
期末專題-進度報告

B06703027/電機四/李晨滔

What I'm currently working on and will continually work on in the future: (大綱)

- 1. 伺服器端以及資料庫端的相關操作、維持(與STM32、RPI之間的資料傳輸)
- STM32上的microphones(MP34DT01)的相關audio操作
- 對MP34DT01收到的audio音訊進行DSP: real-time DSP(目標)
- 一些整合功能上所遇到的issues的debugging

整體Project架構: Review



伺服器端以及資料庫端

- Socket programming by python
- 資料庫: pymongo (mongoDB)
- 會將STM32回傳到server的資料，進一步上傳到資料庫(sensor data, BLE資訊等等)，以便後需要相關資料時可以fetch，或者有機會可以再做資料的視覺化處理
- STM32傳過來的資料格式會在server端parsing成json格式(或者說，dictionary格式)，接著才會上傳到雲端資料庫
- Server端也會順便去分辨資料是從STM32還是從RPI傳過來的

伺服器端以及資料庫端

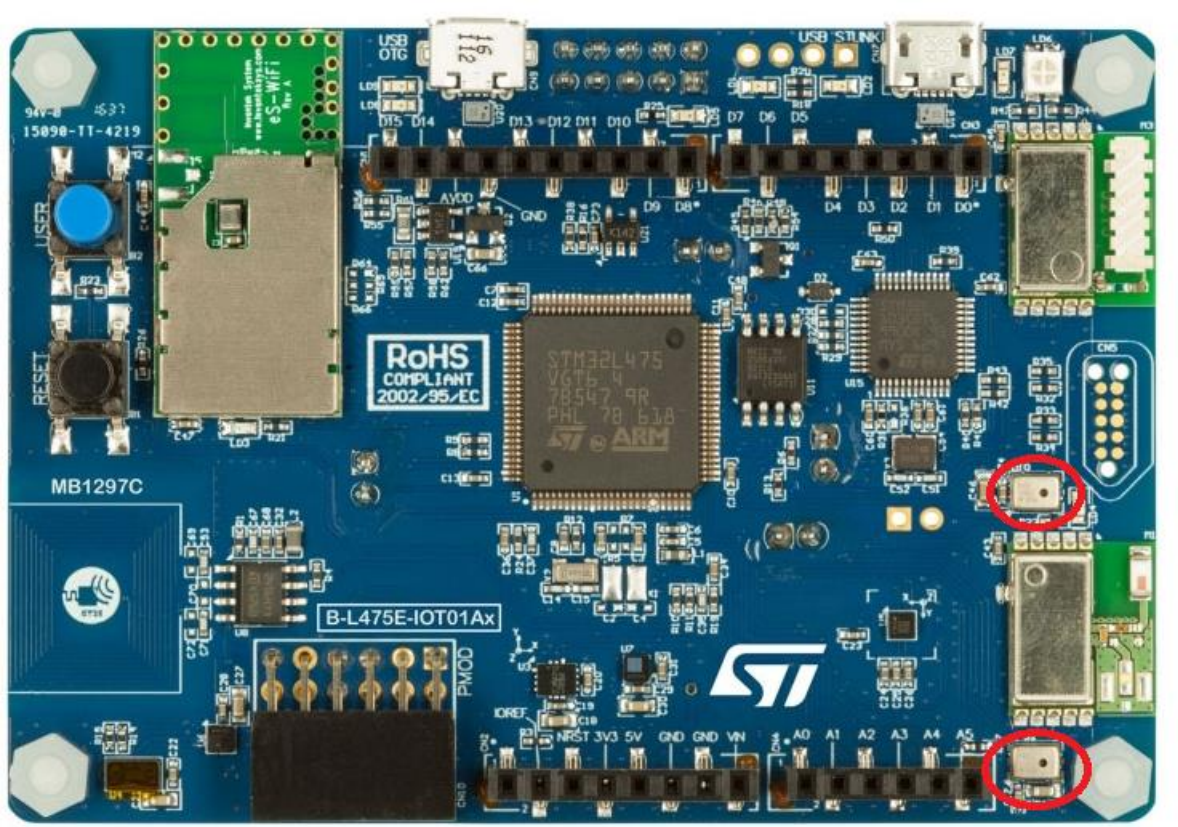
```python
with conn:

    print('Connected by', addr)
    conn.send(bytes("Successfully connected.", "utf-8"))
    while True:
        sensor_data = conn.recv(200).decode('utf-8')
        print(sys.getsizeof(sensor_data))
        upload_dict=get_dict_of_sensor_data(str(sensor_data))
        lis = []
        lis.append(upload_dict)
        collectionName.insert_many(lis)
```

```
_id: ObjectId("627bd1120c68c00688d07b63")
T: 30.19
H: 76.99
P: 1011.72
Mag_X: 519
Mag_Y: -419
Mag_Z: -548
Gyr_X: 560
Gyr_Y: -2590
Gyr_Z: 560
Acc_X: 70
Acc_Y: -8
Acc_Z: 1001
```

```
Received from socket server :  {"T":30.19, "H":76.99, "P":1011.72, "{"Mag_X":519, "Mag_Y":-419, "Mag_Z":-548}", "{"Gyr_X":560.00, "Gyr_Y":-2590.00, "Gyr_Z":560.00}", "{"Acc_X":70, "Acc_Y":-8, "Acc_Z":1001}"}
This message is from STM32.
```

STM32上的microphones(MP34DT01)



- B-L475E-IOT01A上面是有麥克風的!
- 以前年度的final projects，頂多是用RPI去收音，我們想說可以嘗試用STM32上面的麥克風
- Lack of mbed audio-related examples→more challenging
- 本周三的時候有成功錄到音!

STM32上的microphones(MP34DT01)

```
6   static uint16_t PCM_Buffer[PCM_BUFFER_LEN / 2];
7   static BSP_AUDIO_Init_t MicParams;
8
9   static EventQueue eventqueue;
10
11  // Place to store final audio (alloc on the heap), here two seconds...
12  static size_t TARGET_AUDIO_BUFFER_NB_SAMPLES = AUDIO_SAMPLING_FREQUENCY * 2;
13  static int16_t *TARGET_AUDIO_BUFFER = (int16_t*)calloc(TARGET_AUDIO_BUFFER_NB_SAMPLES, sizeof(int16_t));
14  static size_t TARGET_AUDIO_BUFFER_IX = 0;
15
16  // we skip the first 50 events (100 ms.) to not record the button click
17  static size_t SKIP_FIRST_EVENTS = 50;
18  static size_t half_transfer_events = 0;
19  static size_t transfer_complete_events = 0;
20
```

STM32上的microphones(MP34DT01)

```
33      // create WAV file
34      size_t wavFreq = AUDIO_SAMPLING_FREQUENCY;
35      size_t dataSize = (TARGET_AUDIO_BUFFER_NB_SAMPLES * 2);
36      size_t fileSize = 44 + (TARGET_AUDIO_BUFFER_NB_SAMPLES * 2);
37
38      unsigned wav_header[44] = {
39          0x52, 0x49, 0x46, 0x46, // RIFF
40          fileSize & 0xff, (fileSize >> 8) & 0xff, (fileSize >> 16) & 0xff, (fileSize >> 24) & 0xff,
41          0x57, 0x41, 0x56, 0x45, // WAVE
42          0x66, 0x6d, 0x74, 0x20, // fmt
43          0x10, 0x00, 0x00, 0x00, // length of format data
44          0x01, 0x00, // type of format (1=PCM)
45          0x01, 0x00, // number of channels
46          wavFreq & 0xff, (wavFreq >> 8) & 0xff, (wavFreq >> 16) & 0xff, (wavFreq >> 24) & 0xff,
47          0x00, 0x7d, 0x00, 0x00, //  (Sample Rate * BitsPerSample * Channels) / 8
48          0x02, 0x00, 0x10, 0x00,
49          0x64, 0x61, 0x74, 0x61, // data
50          dataSize & 0xff, (dataSize >> 8) & 0xff, (dataSize >> 16) & 0xff, (dataSize >> 24) & 0xff,
51      };
```

STM32上的microphones(MP34DT01) wave file result


raw2.txt - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
524946462cfa00057415645666d74201000000001000100803e0000007d000002001000646174610 0fa00008be356e34ae330e339e34
b7e2ace2bee2e3e200e315e31be315e310e312e315e326e33fe369e3a2e3d2e3c1e3cfe3cfe3c3e3eee322e434e441e42ee417e407e41
d1e3d3e3e4e320e446e463e46fe45ce43ae436e433e445e444e464e45fe452e442e44de458e44de449e41de42be44ce45be461e47ee48
a7e4b9e4a9e471e446e45ae473e472e440e4eee3f0e3f2e3fae30fe4f7e3c3e3ade3ade3b2e3d1e3d3e3e0e3efe3e3e3b1e38ce391e37
3be32fe306e3f8e2fbe2e9e2bde2a3e29ce298e28ae289e29ae2a6e2b2e2b1e2bbe2cde2f7e21ee315e31ce318e314e318e333e33be32
1ae44be443e42ce431e424e40ee4eae313e43ce47fe484e47ee4b4e4ede4fde4dae49de4bee4d2e4c3e4dbe4c2e4f7e453e5a4e5dce5f
2ce3b7e34ee4b7e4ace402e53ee5fce470e474e4a1e489e479e4a3e46ce40ae4b9e3bae3eee3f5e3d1e36be332e352e396e325e4c7e4f
5fe587e561e5e6e426e4a3e29ae14fe29fe3bbe4c7e4a7e477e49ce4aee485e4a6e441e69de76ee76eee665e5b8e444e411e459e5ace60
61e498e4cfe4f5e437e5afe4e8e4afe4bae439e532e556e5c8e53ae59fe559e5d3e5e6e590e5eee54de54ee512e59ee4cee42ee416e48
2ee7f9e6cbe6afe6bbe6c0e664e62ee634e64ae661e66be690e69ee6b1e6a6e67ae641e63be60be6c5e5ace5cfe5c3e5c2e5b4e5f2e5f
cce7f8e7d4e7c2e7c8e7b9e7a4e79de7c8e7e3e704e8fce707e80fe806e8f9e708e8e2e7b5e7a1e770e77ae763e749e767e78be77fe76
35e833e82de842e859e84be842e820e824e83ee84ce853e854e86ce888e88de884e88ae8a0e8a5e8b4e8c4e8c5e8e6e815e914e901e90
50e968e94de95ae946e91fe9f2e8cde8cde8fee82de95fe982e986e992e9a6e9a6e9b7e9d2e9c0e9a7e95ee956e986e9bae9d3e9bfe9a
29ea52ea73ea69ea40ea18eafde9e3e9a9e957e95de989e9cbe9f4e9f7e9e7e9f4e9dce996e94de91ee93be96ae99ae9e1e9eae9f8e91
f2e9dee9dfe9f8e91cea35ea52ea8beacdeaebeae3ead1eab4ea95ea8fea90ea88ea7aea5eea59ea5cea5aea80eab8eae3eac8eaa5ea9
f4e8f1e8fbe816e90ce91fe936e91de923e940e940e94fe98ce9b7e9b3e9a0e9abe9b4e9b3e9cce9b5e995e99de99de962e94be945e94
90e97de970e97ee99be9bae9dfe906ea12eaf0e9bce982e995e9aae9bfe9c4e9c5e9e9e9fae9eae9d3e9bce9a0e979e94ae95de951e95
27e82ee812e8f5e7eee7c1e7b7e7a4e798e7cbe7f1e706e815e827e857e87ee893e878e84ae865e878e897e891e868e899e8abe892e88
efe7c9e78ee773e78de7d0e7d6e7cae7bee7c4e7a1e7a4e768e711e7fce6f0e6ffe60de7ece6f8e6dee6c0e6c0e69ce690e6aee6d3e6e
c2e7d4e706e821e82be850e848e84de847e83de83be827e80de8f4e7c0e7b1e7a9e76ce744e738e721e7f8e6d5e6d6e6b7e69fe67be63
90e86be849e83ee834e845e841e83ae847e875e8a9e8c5e8e9e80de93ce94ce923e900e9dce8f0e8f5e8c9e8abe886e881e86ce867e82
70e967e96be977e995e988e98ce993e977e952e93de9ece8d8e8d1e8abe87ce856e84be82ee80de805e8f1e7c9e7a2e76fe774e775e78
03e8efe7e7e70fe845e868e82fe813e812e82ee82ee81ae80fe80de801e807e80de833e872e89fe8d0e8c9e8a3e88ee89ce8c8e8d6e8c
75e65be667e69ce6cce6d6e6e1e6ebe6eae6f8e62ce755e780e79be7d1e7e1e706e82ae82de810e8efe7f0e7f4e7fee71ee820e8fbe7e
9ae847e827e81de815e842e850e8abe8f4e842e931e9ede8a4e881e85de873e885e8a5e8dee8eae82fe96ae9aae991e9bae9cce9cde9e
b6e88ae885e8cae826e951e984e9aae902ea2dea07eaa9e96fe93ee90ce9d1e8b0e8ace8d6e8f7e82ee94ee962e918e90de938e967e97
18e78de723e831e95ee9a2e802e896e7b2e727e84fe88be870e819e835e834e816e8cbe7bae7cce7f1e76fe814e993e970e9e2e880e86
5be90ce9e3e8dde8e7e810e933e957e980e9b0e9d7e9cbe9cfe9d5e9abe9c4e9dae9dbe9ede9cfe9bce9bce9d3e9fce924ea24ea10ea1
9fe82ae831e817e840e873e88ee95ae9b1e8ede78be62ee6e9e62be719e8c3e804e91ce959e88be720e742e7cbe73ee8a0e842e83fe84
d4e985e98ae85ae81de804e8dde8d9e95de970e724e632e61ce706e9a2e930e926e846e6b2e62be84ae962e91be98ae917e97ce847e77
50e8bee895e993ea65e9b1e87de86de8ade9fae9a4e93ee9fee747e730e797e740e79de64de683e6fee6fbe629e7ece671e6cae605e85
```

cat raw2.txt | node converter/hex-to-buffer.js test2.wav

對MP34DT01收到的audio音訊進行DSP

- 我們有以下兩種想嘗試的方向:

  (1) 對音訊進行real-time DSP, e.g., a real-time low-pass filter，直接在buffer上做音訊處理

  (2) 錄好音訊後，對音訊進行DSP，接著回傳給RPI或者Server端，達成遠距監控

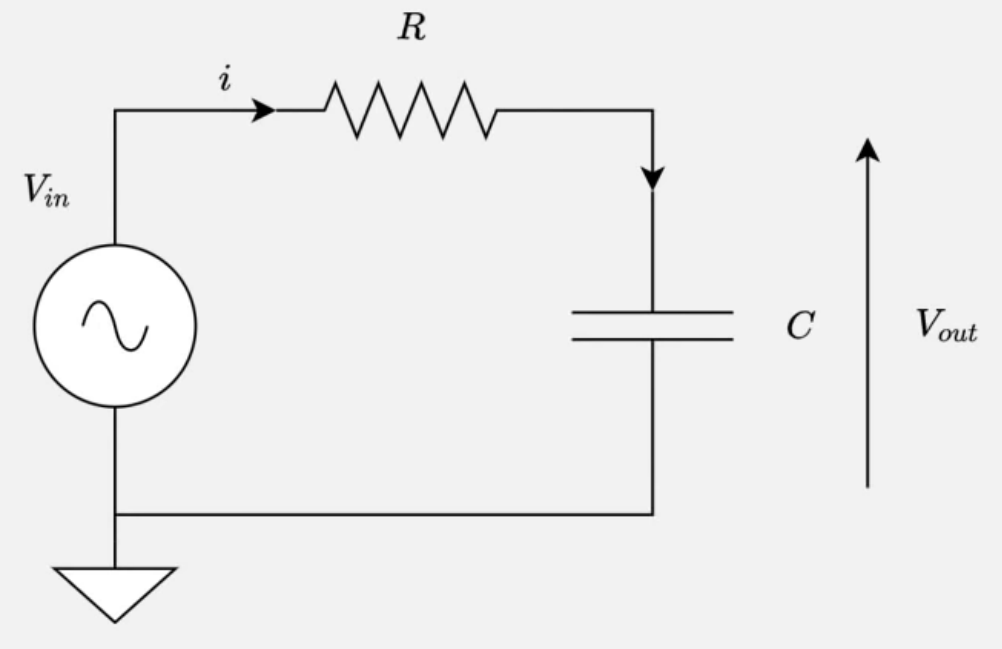
- 方法: 直接使用CMSIS的DSP庫，或者我們自己寫一個簡單的DSP program(difference equations)

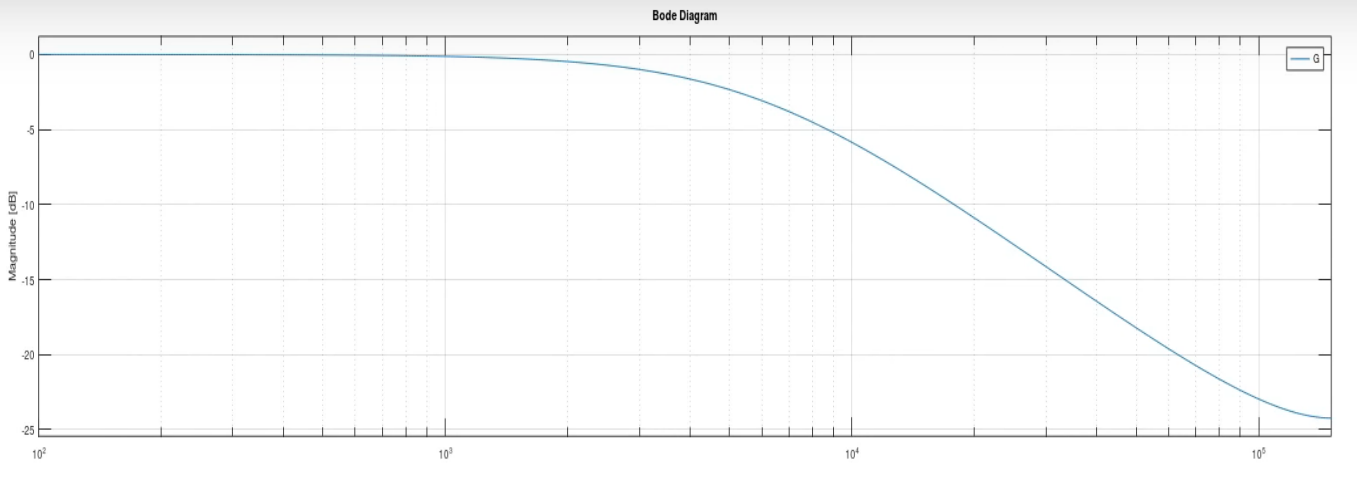
The more general form of a discrete-time nonrecursive filter is

$$y[n] = \sum_{k=-N}^{M} b_k x[n-k],$$

對MP34DT01收到的audio音訊進行DSP

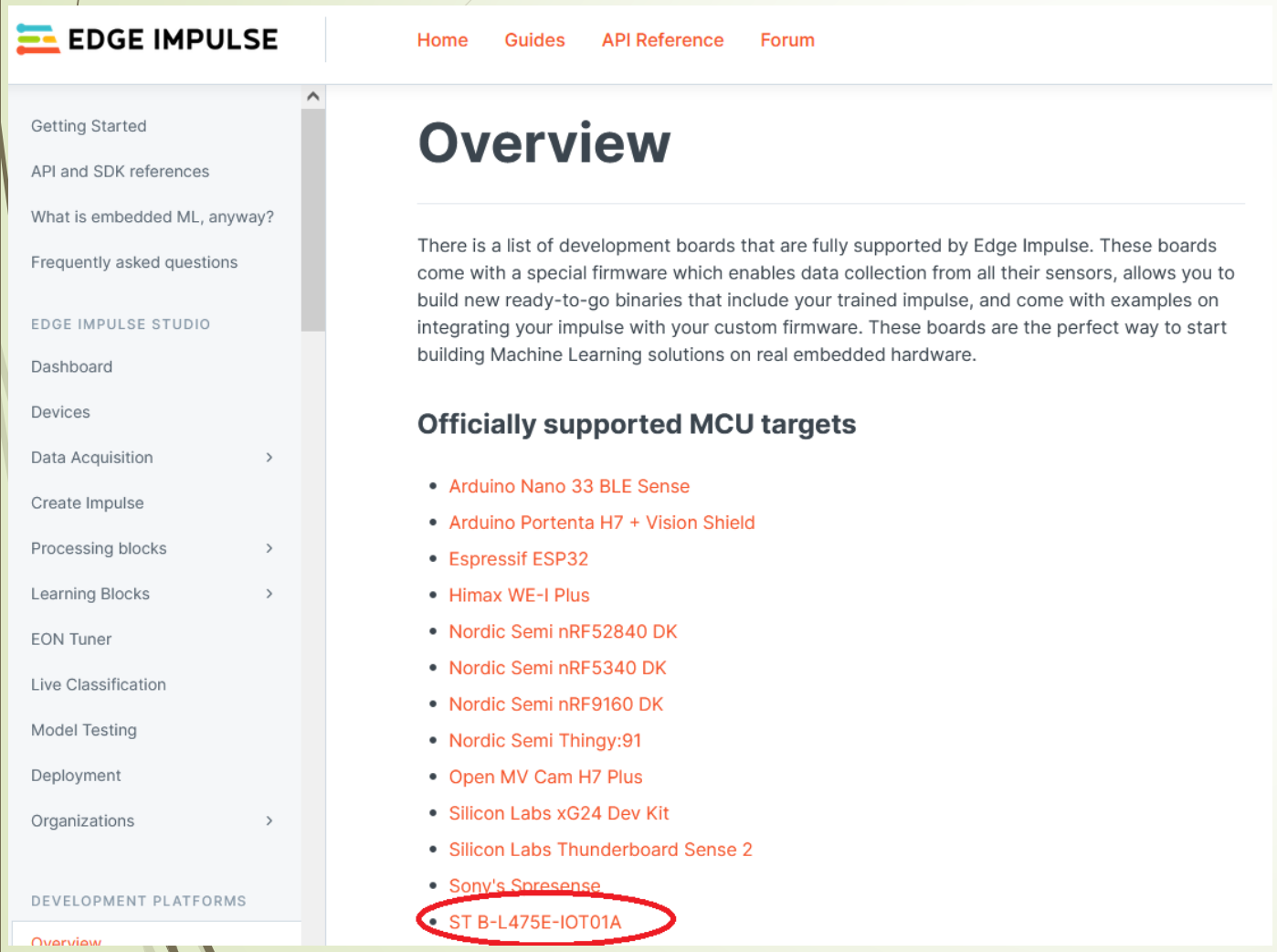
Analogue RC filter prototype





$$V_{out}[n] = \frac{\alpha}{1+\alpha} \cdot V_{in}[n] + \frac{1}{1+\alpha} \cdot V_{out}[n-1] \qquad (\alpha = 2\pi \cdot \frac{f_c}{f_s})$$

其他



- 一些整合功能上所遇到的issues的debugging
- Perhaps: tinyML, Edge Impulse (still under investigation)

Ending

- 謝謝大家聆聽~
- Ref:
- https://docs.edgeimpulse.com/docs/development-platforms/fully-supported-development-boards
- https://ithelp.ithome.com.tw/m/articles/10265166
- https://www.st.com/en/evaluation-tools/b-l475e-iot01a.html