

251212금_개선 사항

Controller10 → Controller10.1

1. 프로젝트 버전 로깅

개선 사항

프로젝트 버전 로깅 기능을 추가하였습니다.

프로젝트의 Setup 부분에 추가하였습니다.

main.c

```
1 //Setup
2     FUNCTION = SPACE;
3     HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET); //Turn on led
4     HAL_GPIO_WritePin(DEVICE_GPIO_Port, DEVICE_Pin, GPIO_PIN_SET);
5     LOG("[DEBUG]", "STM32 OK!\n");
6     LOG("[VER]", "Controller_v0.10.1\n");
7     ...
8
```

테스트

log.txt

```
1 10:23:40.514 [DEBUG]STM32 OK!
2 10:23:40.514 #[VER]Controller_v0.10.1
```

2. I2C HAL_BUSY로 인한 App 장애

개선 사항

```
1 LOG("[ANS]", "FAIL");
```

해당 로그로 인해 App에서 단계 진행이 안되었습니다.

때문에,

```
1 LOG("[ANS]", "SYSTEM_FAIL");
```

로 로그를 수정하였고,

FAIL이 되어도 App의 단계 진행은 계속되도록 조치하였습니다.

또한 말씀하신 바와 같이 지금은 사용하지 않는 레이저 센서(VL53L0X)를 주석처리 하였습니다.

```

1 //TOF SENSOR
2 // statInfo_t_VL53L0X distanceStr;
3 // initVL53L0X(1, &hi2c1);
4 // uint16_t offset_DistanceTof = VL53L0X_OFFSET(&distanceStr);

```

현재는 주석처리로만 마무리 하였지만 추후에는 하드웨어 계층을 나누어서 해당 계층에서 센서를 비활성화 시키겠습니다.

App 테스트

첫번째 테스트는 이상없이 동작하나

두번째 테스트부터 로드셀의 law data가 비정상적으로 출력됩니다. 모두 0 혹은 음수 값으로 출력됩니다.

터미널 테스트

아래의 과정을 수십번 반복하였고 이상없이 I2C HAL은 동작합니다.

Test Code

main.c

```

1 case HANDSHAKE:
2
3     for (int repeat = 0; repeat < 100; repeat++) {
4         memset(i2c_check_buf, 0, sizeof(i2c_check_buf));
5         for (uint16_t addr = 1; addr < 128; addr++) {
6             HAL_StatusTypeDef status = HAL_I2C_IsDeviceReady(&hi2c1,
7                 addr << 1, 1, 5);
8             // I2C HAL 상태가 HAL_OK라면 I2C 주소 출력
9             if (status == HAL_OK) {
10                 snprintf(i2c_check_buf, sizeof(i2c_check_buf),
11                     "I2C Device Found: 0x%02X\r\n", addr);
12                 LOG("[I2C]", i2c_check_buf);
13                 // I2C HAL 상태가 HAL_BUSY 상태라면
14             } else if (status == HAL_BUSY) {
15                 LOG("[I2C]", i2c_check_buf);
16                 LOG("[I2C]", "HAL_BUSY\r\n");

```

```

17 // 100번 루프 진행
18 for (int i = 0; i < 10; i++) {
19     // I2C 비활성화
20     HAL_I2C_DeInit(&hi2c1);
21     // I2C1 하드웨어 강제 리셋
22     __HAL_RCC_I2C1_FORCE_RESET();
23     HAL_Delay(10);
24     LOG("[I2C]", "FORCE_RST_TRYING...\n");
25     // I2C1이 강제 리셋되었다면
26     if (RCC->APB1RSTR & RCC_APB1RSTR_I2C1RST) {
27         LOG("[I2C]", "I2C1_RESET_BIT_ON\n");
28         LOG("[I2C]", "RELEASE_RST_TRYING...\n");
29         // I2C1 리셋 해제
30         __HAL_RCC_I2C1_RELEASE_RESET();
31         // I2C1 리셋 해제가 되었다면
32         if ((RCC->APB1RSTR & RCC_APB1RSTR_I2C1RST)
33             == 0) {
34             LOG("[I2C]", "I2C1_RESET_BIT_OFF\n");
35             LOG("[I2C]", "I2C1_RESET_SUCCESS\n");
36             // HAL_I2C 활성화
37             HAL_I2C_Init(&hi2c1);
38             // 안정화 DELAY
39             HAL_Delay(10);
40             // HAL_I2C 초기화 후 상태가 HAL_OK가 아니라면
41             if (HAL_I2C_Init(&hi2c1) != HAL_OK) {
42                 LOG("[I2C]", "HAL_INIT_FAIL\n");
43                 continue; // 다음 루프 계속 진행
44             } else {
45                 LOG("[I2C]", "HAL_INIT_OK\n");
46             }
47         }
48     } else {
49         LOG("[I2C1_RST]", "I2C1_RESET_FAIL\n");
50     }
51
52     status = HAL_I2C_IsDeviceReady(&hi2c1,
53                                     addr << 1, 1, 5);
54     // HAL_OK 시 break
55     if (status == HAL_OK) {
56         LOG("[I2C]", "HAL_RECOVERY_SUCCESS\n");
57         break;
58     }
59
60     if (i == 9) {
61         LOG("[I2C]", "HAL_RECOVERY_FAIL\n");
62     }
63 }
64 } else if (status == HAL_TIMEOUT) {
65     LOG("[I2C]", "HAL_TIMEOUT\r\n");
66     __HAL_RCC_I2C1_FORCE_RESET();
67     HAL_Delay(10);
68     if (RCC->APB1RSTR & RCC_APB1RSTR_I2C1RST) {
69         __HAL_RCC_I2C1_RELEASE_RESET();

```

```

70         } else {
71             LOG("[I2C1_RST]", "I2C1_RESET_ERROR\n");
72         }
73         HAL_I2C_DeInit(&hi2c1);
74         HAL_Delay(10);
75         HAL_I2C_Init(&hi2c1);
76
77         status = HAL_I2C_IsDeviceReady(&hi2c1, addr << 1, 1,
78                                         5);
79     }
80 }
81 }
82
83 ...

```

Test Result

log.txt

```

1  12:10:27.233 [DEBUG]STM32 OK!
2  12:10:27.233 #[VER]Controller_v0.10.1
3  12:10:27.236 #12:10:30.288 [CMD]HANDSHAKE
4  12:10:30.918 [ANS]ACK#[I2C]I2C Device Found: 0x29
5  12:10:30.921 #[I2C]I2C Device Found: 0x77
6  12:10:30.926 #[I2C]I2C Device Found: 0x29
7  12:10:30.936 #[I2C]I2C Device Found: 0x77
8  12:10:30.942 #[I2C]I2C Device Found: 0x29
9  12:10:30.952 #[I2C]I2C Device Found: 0x77
10 12:10:30.957 #[I2C]I2C Device Found: 0x29
11 12:10:30.967 #[I2C]I2C Device Found: 0x77
12 12:10:30.972 #[I2C]I2C Device Found: 0x29
13 12:10:30.981 #[I2C]I2C Device Found: 0x77
14 12:10:30.987 #[I2C]I2C Device Found: 0x29
15
16 ...
17
18 12:10:32.412 #[ANS]SIM_OK
19 12:10:33.447 #[ANS]ULTRA_OK#[ANS]LC1_OK
20 12:10:33.459 #[ANS]LC2_OK
21 12:10:33.538 #[ANS]WS_OK
22
23 ...

```

Flutter 코드 분석

해당 프로젝트의 코드를 분석했습니다.

분석한 내용은 아래에 정리해두었습니다.

0. Overview

[0. Overview](#)

1. Architecture

[1. Architecture](#)

2. EntryPoint & Configuration

[config.dart](#)

[main.dart](#)

[router.dart](#)

[router.g.dart](#)

3. Presentation(UI)

screen

[s_driver.dart](#)

[s_opening_door.dart](#)

[s_setting.dart](#)

[s_splash.dart](#)

[s_step1.dart](#)

[s_step2.dart](#)

[s_step3.dart](#)

[s_step4.dart](#)

widget

[box_setting.dart](#)

[btn_home.dart](#)

[btn_number.dart](#)

[btn_setting.dart](#)

[btn_to_connect_port.dart](#)

[btn_to_setting.dart](#)

[circular_prograss.dart](#)

[debug_buttons.dart](#)

[w_header.dart](#)

[w_left_triangle.dart](#)

[w_step_nev.dart](#)

4. ViewModel

[vm_driver.dart](#)

[vm_serial_port.dart](#)

[vm_setting.dart](#)

[vm_step1.dart](#)

[vm_step2.dart](#)

[vm_step3.dart](#)

[vm_step4.dart](#)

5. Model

[ui_state_setting.dart](#)

[ui_state_step1](#)

[ui_state_step2](#)

[ui_state_step3](#)

[ui_state_step4](#)

[ui_state_usb_port](#)

6. Common

[app_commands.dart](#)

[app_strings.dart](#)

[custom_toast.dart](#)

[show_toast.dart](#)

[toast_provider.dart](#)

[toast_state.dart](#)

차후 계획 및 진행 사항

STM32 H/W 및 F/W

0. 오류 개선

1. 비정상 센서 출력 오류 해결

1. 코드 간소화

개선 사항

아래와 같이 case마다 모듈화 및 함수 단일화를 진행하고 있습니다.

main.c

```
1  ...
2
3  switch (FUNCTION) {
4      case HANDSHAKE:
5          Handshake();
6          break;
7
8      case VALIDATION:
9          Validation();
10         break;
11
12     case POST_UCO:
13         PostUCO();
14         break;
15
16     case POST_PER:
17         PostPeriodic();
18         break;
19
20     case OPEN_INPUT:
21         DoorOpen_Measure();
22         break;
23
24     case CLOSE_INPUT:
25         DoorClose_Measure();
26         break;
27
28     case RECHECK:
29         Recheck();
30         break;
31
32     case UNLOCK_DOOR:
33         UnlockDoor();
34         break;
35
36     case SLEEP:
37         SleepMode();
38         break;
39
40     case ALARM_WAKEUP:
41         AlarmwakeupMode();
42         break;
43
44  ...
```

테스트

현재는 변수 중복선언에 의한 에러가 빈번히 발생해서 개선중에 있습니다.

4. 파일 트리 개선

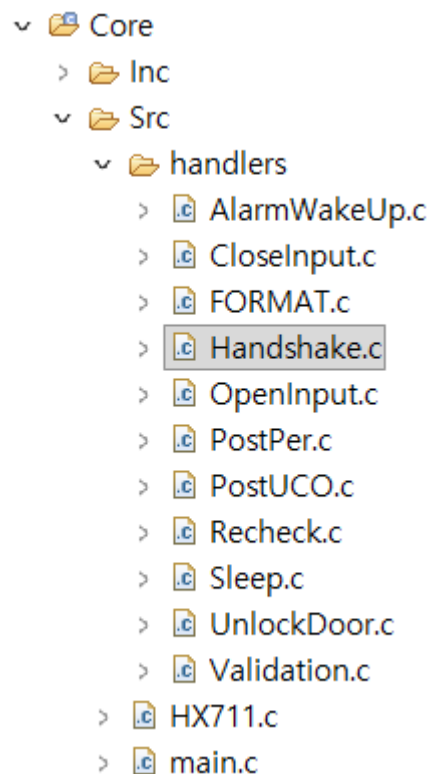
개선 사항

아래와 같이 switch-case 문의 상태에 따른 동작들을 handlers 디렉터리에 정리하였습니다.

추후에는 프로젝트의

- 하드웨어 드라이버 모듈
- 하드웨어 점검 및 초기화 모듈
 - 하드웨어 리셋
 - 펌웨어 레지스터
- 에러 및 예외 처리 모듈
- 로그 출력 모듈
- 더미 데이터 출력 모듈
- 기능 테스트 모듈
- 통합 테스트 모듈

등으로 디렉터리를 구조화 및 정돈할 예정입니다.



테스트

현재는 변수 중복선언에 의한 에러가 빈번히 발생해서 개선중에 있습니다.

Git/GitHub를 통한 App 빌드 업무 개선

개선 사항

Flutter App 설치를 위한 GitHub 리포지토리를 생성하였고 App 설치 과정을 조금 더 편리하게 하였습니다.

추수에는 CI/CD를 통해 Git Push 만으로 App 빌드가 되도록 개선하겠습니다.

테스트

미진행 상태입니다.