04

무선 탈리(Tally) 시스템을 만들어 보자

진신우 SBS 미디어IT팀 부장

방송 제작 현장에서 반드시 쓰이는 탈리(Tally) 시스템이란 현재 선택된 비디오 소스를 출연자들이나 제작 스태프들끼리 공유시켜주는 시스템을 말한다. 쉽게 말하자면, 현재 녹화 중인 카메라에 붉은 등이 켜지게 함으로써 출연자들의 자연스러운 시선 유도를 가져올 수 있으며, TD, PD, 오디오, 비디오 담당자들은 선택된 소스에 좀 더 공을 들이게 된다. 일반적으로 탈리 시스템의 입력으로 비전 믹서가 사용되며, 탈리의 출력은 UMD(Under Monitor Display) 또는 멀티 뷰어, 모니터, 카메라 베이스 등과 연결된다.

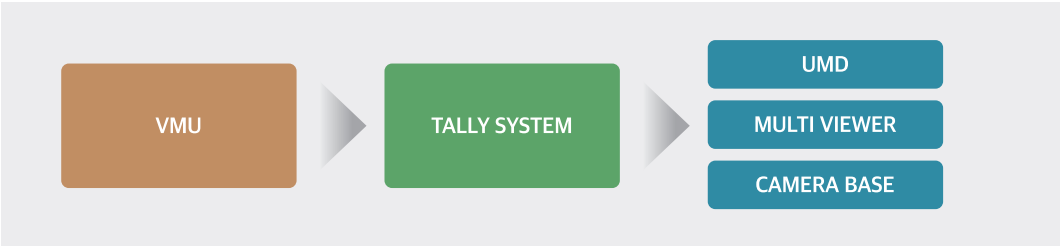

그림 1. 일반적인 탈리 시스템 - 방송제작 시설

TV 스튜디오 용도로 제작된 표준 카메라는 탈리 시스템과 연결이 쉬웠지만, 현재 고도화된 TV 프로그램 제작 현장에서는 다양한 카메라들이 사용 중이며, 탈리 시스템과 연결이 어려운 경우도 발생하고 있다. VMU와 탈리 시스템 간에는 GPIO 인터페이스 또는 시리얼 인터페이스를 사용하게 된다. GPIO란 General Purpose Input Output의 약자로써, On/Off 2가지 상태를 가지는 전기 스위치처럼 정보를 전달한다. GPO는 On/Off 신호를 출력하게 되고, GPI는 On/Off 신호를 수신하는 형태이다.

일반적인 GPI 회로의 입력단을 살펴보자.

GPI 핀과 GND 핀을 결합하면 PC817은 ON 상태가 된다. 이때 MCU_INPUT은 LOW 상태가 되며, MCU(Micro Controller Unit) 프로그램에서 상태 변화를 감지하여 처리할 수 있다.

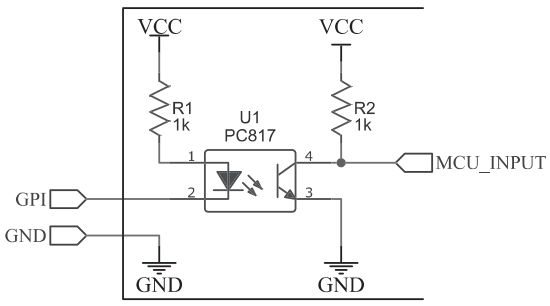

그림 2. GPI 입력 회로

일반적인 GPO 회로의 출력회로를 살펴보자.

MCU 출력에 따라서 Q1은 On/Off 두 가지 상태를 가진다. Q1 상태에 따라서 GPO/GND는 On이거나 High-Impedance 상태를 가질 수 있다. 트랜지스터의 콜렉터에서 신호를 뽑아냈으며, 콜렉터 쪽에는 저항이 없다. 이를 오픈 콜렉터 출력이라고도 한다. 오픈 콜렉터의 장점은, 신호를 전달받는 쪽은 3.3V, 5V TTL, CMOS 등 논리회로 전압 레벨을 가리지 않는다는 점이다. Pull Up 저항을 상대방 회로에서 구현함으로써 서로 다른 논리 전압 레벨을 가진 회로끼리 상호 연동이 쉬워진다. 주변에서 흔히 보이는 FAN의 센서 출력도 이러한 형태를 가지고 있다.

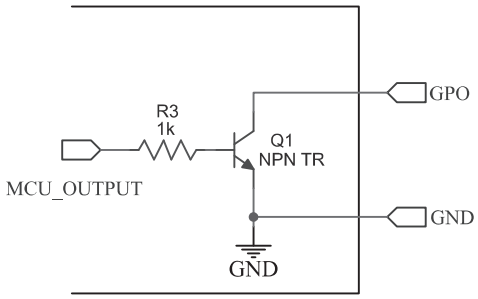

그림 3. GPO 출력 회로

GPIO 방식에 대하여 살펴봤으니, 이제 목적에 부합하는 MCU를 찾아보자. VMU로 부터 10개 정도의 GPO 출력을 받아서, Tally 1번~10번까지 제어할 계획이다. I/O 핀 개수가 충분하고, MicroPython 언어 사용이 가능한 RP2040을 선택했다. RP2040은 3.3V, 133MHz로 동작하는 듀얼코어 MCU이다. OLED 디스플레이용으로 사용할 I2C 통신 기능을 갖추고 있으며, 나중에 설명할 RADIO 모듈과 통신목적으로 사용할 MOSI/MISO 핀도 가지고 있다.
국내 IC 판매 업체나 알리 익스프레스 등을 통해서 RP2040 개발 보드는 손쉽게 구할 수 있으며, 가격 또한 저렴한 편이다. 개발 보드는 2.54mm 피치를 가지고 있어 브레드 보드에 꽂아 테스트해 보기에도 편리하다.

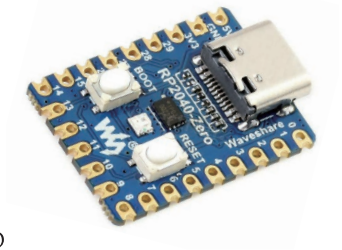

그림 4. RP2040-ZERO

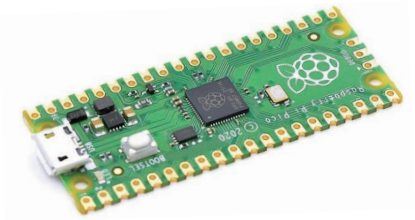
그림 5. Raspberry PI Pico

디지털 무선통신을 가능하게 하는 RADIO 모듈을 찾아보자. SX1278 RA-01 433MHz를 선택하였다. 그 이유는

- MCU와 동일한 3.3V 동작 전압을 가진다.
- SPI 방식으로 MCU와 통신이 가능하다.
- 깃허브에는 해당 모듈에 대한 예제가 많았다.
- 스튜디오 무선 마이크용으로 사용되는 2.4GHz와 혼신 걱정이 없다.
- LoRa[1)] 통신 모드를 사용하여 1km 이상 통신이 가능하다.

1. LoRa(Long Range) : 사물끼리 서로 통신을 주고받을 수 있게 도와주는 저전력 장거리 통신(LPWA, Low Power Wide Area) 기술. 저전력, 긴 전송 거리, 간섭에 강한 특징을 가지며, 농업, 스마트 시티, 산업 자동화 등 다양한 분야에서 사용된다.

그림 6. RA-01 라디오 모듈 (LoRa 통신)

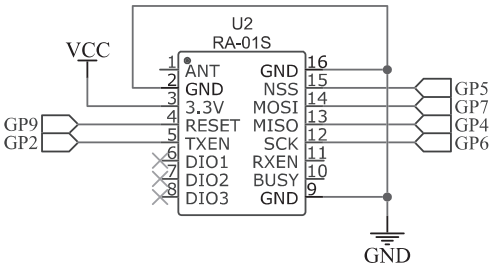

그림 7. RA-07 기본 연결

COLD RESET 기능이 필요할 때 GP9와 연결된 RESET 핀을 사용하며, MOSI/MISO/SCK는 SPI 통신용으로 사용된다. NSS는 SPI 통신을 하는 슬레이브 디바이스가 병렬로 존재할 때 CHIP SELECT 신호로 사용한다. DIO0(TXEN) 핀은 무선 패킷이 도착하여 디코딩 가능할 때 활성화된다. 해당 핀에 MPU 인터럽트를 걸면, 콜백 함수로 수신 기능을 처리할 수 있다.

434MHz 센터 주파수, 500kHz BW 송출 시 스펙트럼을 관찰한 그림이다.

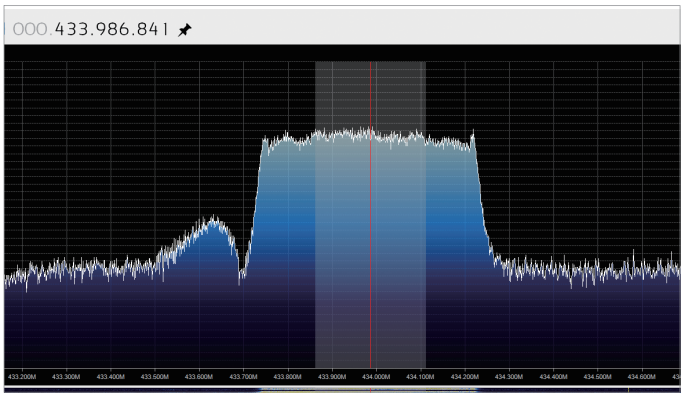

그림 8. 434MHz LoRa 송출 스펙트럼

RADIO 모듈을 제어하는 MicroPython 코드를 살펴보자.

```
tr = RADIO(mode=0)  # LORA=0  FSK=1 OOK=2
tr.setFrequency(465000) # set freq in kHz
tr.setPower(10, True)       # power dBm (RFO pin if False or PA_BOOST pin if True)
tr.setHighPower(False)      # add +3 dB (up to +20 dBm power on PA_BOOST pin)
tr.setOCP(120, True)        # set OCP trimming (> 120 mA if High Power is on)
tr.enableCRC(True, True)   # CRC, CrcAutoClearOff (FSK/OOK mode)
tr.setPllBW(2)
tr.setBW(500.)        # BW: 7.8...500 kHz
tr.setCR(8)           # CR: 5..8
tr.setSF(10)          # SF: 6...12
tr.setLDRO(False)     # Low Datarate Optimize
tr.setPreamble(6)     # 6..65535 (8 by default)
tr.setSW(0x12)        # SW allways 0x12   sync word

tr.dump()
tr.collect()

tr.onReceive(on_receive) # set the receive callback
```

코드 1. RA-01 제어 코드 (MicroPython)

먼저 LoRa 모드로 라디오 객체를 생성하고, 주파수, 송출 파워, CRC, 프리엠블 등을 설정한 후, 수신용 콜백 함수를 등록한다. RADIO 클래스는 깃허브에서 가져 왔으며, 개발 용도에 맞게 일부 메서드를 추가하고 수정하였다.

OLED 디스플레이 모듈을 찾아보자. 0.96인치 128×64 OLED 모듈은 MCU 업계에서 아주 흔하게 사용되는 디스플레이 장치이다. I2C 방식으로 CLOCK, DATA 2개의 핀으로 디지털 디스플레이를 구현할 수 있다.

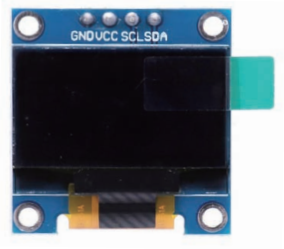

그림 9. 0.96인치 OLED 디스플레이 모듈

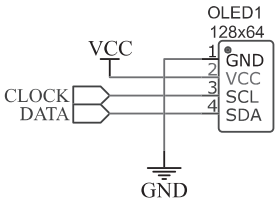

그림 10. MCU, OLED 연결

이제 무선 탈리 수신기의 전원부를 설계해 보자.

충전을 쉽게 하기 위해서는 USB-C 단자가 필요하다. 에너지원으로 리튬이온 18650 배터리를 사용한다. 전원 Off 시 대기 전류가 적어야 낮은 자연 방전율을 가진다. 이때 USB-C 단자를 통해서 충전도 가능해야 한다. 배터리 잔량을 표시하기 위해서 배터리와 MCU의 ADC(Analog to Digital Converter)는 전기적 연결이 필요하며, 전원을 끄면 전류 소모를 줄이기 위해서 ADC와는 분리되는 회로 구조여야 한다.

3.3V LDO Regulator(Low Dropout Regulator)를 사용할 예정이므로, 리튬이온 공칭 전압인 3.8V는 반드시 4.5V 이상 승압하여 사용해야 한다. 소형이면서 배터리 충·방전 및 승압 기능까지 포함한 모듈을 찾아보니, 세상에는 그런 모듈이 이미 존재하고 있었다.

그림 11. 충·방전 전원 모듈

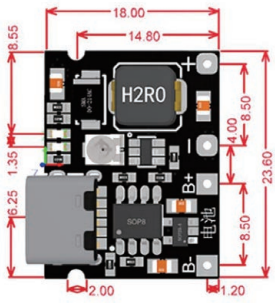

그림 12. 전원 모듈 footprint

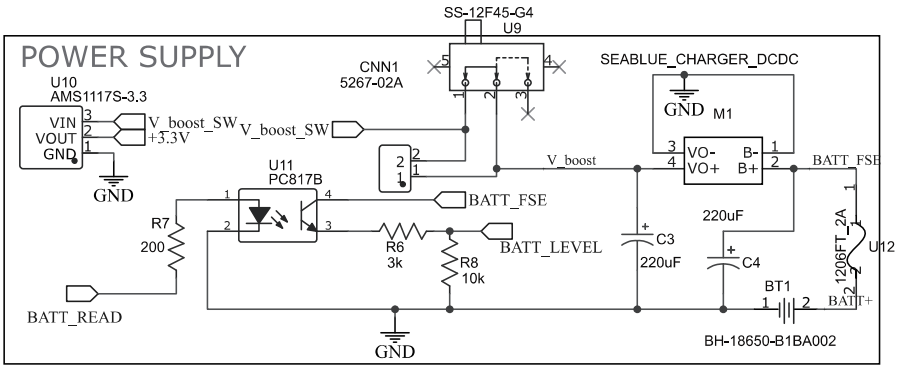

그림 13. 무선 탈리 수신기 전원부 디자인

자세한 치수까지 표시되어 있어, PCB 디자인으로 사용할 footprint 제작은 쉬워졌다.

회로 오동작으로 배터리 과전류를 방지하기 위해서 휴즈를 설치했고, 배터리 플러스 전압을 읽어내는 회로는 포토 커플러를 통해서 MCU와 완전히 분리하였다. 스위치 Off 시 승압 된 배터리 전압(V_boost)은 대략 4.5V로 측정되지만 대기 전류는 수십 μA 단위로 매우 적었다. 대략 1년 이상은 방치를 해도, 과방전으로부터 안전할 수 있다.

탈리의 기본 기능은 붉은색 등을 표시하는 것이다. 밝기 조정까지 가능하면 좋겠다. MCU를 사용하여 LED 밝기를 조절하기 위해서는 PWM 기능을 사용해야 한다.
PWM DUTY가 크면 LED는 밝아질 것이고, 작으면 어두워질 것이다. 이 회로의 단점은 저항, TR 등 주변 소자들이 붙어야 하고, LED 개수가 많을수록 더욱 많은 저항을 필요로 하며, PCB 디자인 소형화에 불리해진다.

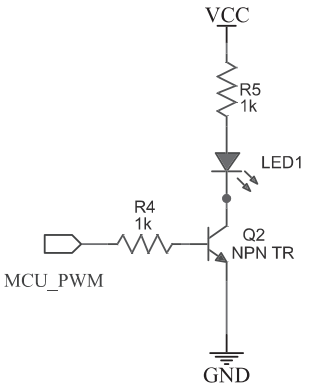

그림 14. LED On/Off (PWM 사용)

그래서 WS2812 LED 시리즈가 등장하였다. NeoPixel 또는 Addressable LED라고도 부른다. 길거리를 걷다 보면 길게 생긴 LED 스트립에서 다양한 애니메이션 효과가 표시되는 것을 볼 수 있다.

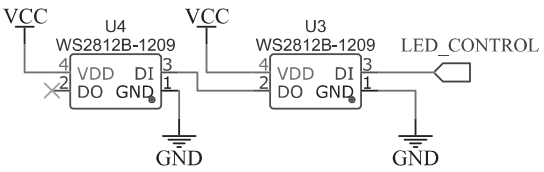

그림 15. WS2812 LED 연결 및 제어

LED 모듈에 전원만 연결하고, LED_CONTROL 신호만 주면 LED끼리는 데이지 체인 방식으로 연결되어 수백 개의 LED를 원하는 RGB 값으로 제어할 수 있다. NeoPixel를 사용하는 라이브러리는 C++ 또는 마이크로 파이선 형태로 널리 보급되어 있다. NeoPixel을 제어하는 C++ 코드를 살펴보자. for 문은 LED 개수만큼 실행되면서 R, G, B값을 설정하고, 마지막에 show를 호출한다.

```
void allred()
{
  _pixels->clear();
  for (int i = 0; i < _led_cnt; i++)
  {
    _pixels->setPixelColor(i, _pixels->Color(_brightness, 0, 0));
  }
  _pixels->show();
}
```

코드 2. NeoPixel 제어 코드 (C++)

그림 16. 다양한 NeoPixel 모듈

이제 PUSH 버튼을 구현하고 객체를 만들어 보자. UP/DOWN/ENTER 역할을 하는 물리 버튼 3개를 사용할 예정이다. 각 버튼은 MCU의 INPUT 핀과 연결되어 있으며 평소에는 PULL UP 저항에 의해서 항상 논리값 HIGH를 유지하게 된다. PULL UP 저항은 회로 구성 시 별도로 설치할 필요는 없고, MCU의 GPI 핀 설정 시 PULL UP 옵션을 활성화할 수 있다.

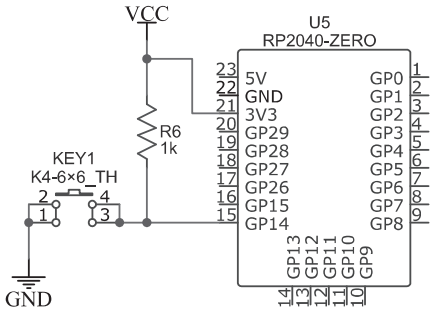

그림 17. MCU, PUSH 버튼 회로도

버튼을 디자인할 때 고려 사항은 다음과 같다.

- 버튼은 이전 상태, 현재 상태를 고려할 때 4가지 상태를 가진다.
- 버튼을 누르거나 떼는 순간, 논리값은 깨끗하게 구분되지 않는다.
  불안정한 상태를 몇 번 가질 수 있으므로 하드웨어적이든, 소프트웨어적이든 디바운싱이 필요하다.
- 버튼 객체에 call-back 를 등록하면 소프트웨어가 간단해진다.

C++로 구현된 버튼 클래스의 scan 메서드 예제이다.

```
int scan(size_t tm_now)
{
  int status_now = _gpi->value();
  int decision = (_prev << 1) + status_now;
  // Serial.println();
  _prev = status_now;
  if (_is_ledon && ((tm_now - _tm_ledon) > 400))
  {
    _is_ledon = false;
    _swled->value(LOW);
  }
  int result = 0;
  switch (decision)
  {
    case 3:   // off state
      break;

    case 0:   // on state
      break;

    case 2: // Pressed....
      result = (tm_now - _tm_last) > 100;
      _tm_last = tm_now;
      if (result)
      {
        _tm_ledon = tm_now;     // Record push button led on time
        _is_ledon = true;
        _swled->value(HIGH);    // Turn on LED
        if (! (_callback == NULL))
        {
          _callback((void *)"");   // do callback function,
        }
      }
      break;

    case 1: // Released....
      _tm_last = tm_now;
      break;
  }
  return result;
}
```

코드 3. 버튼 클래스, scan 메서드 (c++)

버튼의 이전 상태와 현재 상태값을 사용하여 00, 01, 10, 11 4가지 상태를 만든 후, 이를 기준으로 버튼이 눌러진 순간 call-back을 호출하도록 하였다. 버튼을 누르는 순간, 버튼에 내장된 LED는 400ms 동안 켜지도록 하였으며, 디바운싱 타임은 100ms로 설정되어 있다.

RA-01 라디오 모듈을 관리하는 sx127x 클래스의 기본 예제를 보면 수신 시 인터럽트에 의해 호출된 콜백 함수를 사용한다. 이때 전달되는 객체에는 RSSI 값과 SNR 값이 포함되어 있다. OLED 표시 창에 이 값을 항상 표시함으로써 무선 수신 상태를 사용자가 확인할 수 있도록 하였다.

그림 18. 수신기 디스플레이(RSSI -51, SNR 6.25dB)

수신기의 OLED 표시 함수를 살펴보자.

```
def draw_display():
    global oled, m, heartbeat, payload_string, bl, indr
    oled.fill(0)
    oled.text(str(bl.get_level()), 80, 0)
    oled.text(m.get_item(), 5, 10)
    oled.text("RX", 100, 10)
    oled.text(str(m.get_value(m.get_item())), 5, 20)
    oled.line(0,35, 127,35,1)
    oled.text(f'{indr.rssi} / {indr.snr}', 5, 40)
    oled.text(f'r={indr.payload_string}', 5, 50)
    oled.text(f'[{heartbeat.get_next()}]', 100, 20)
    oled.show()
```

코드 4. OLED 표시 함수 (MicroPython)

OLED 화면을 지운 후 사용자에게 보여줄 데이터들을 표시해 준다. 해당 함수는 반복 타이머 클래스에 의해서 300ms마다 호출되거나, 사용자 버튼 입력이 있을 때, 또는 데이터 수신이 발생했을 때마다 실행된다.

무선 탈리 시스템의 전체적인 계통도는 다음과 같다.

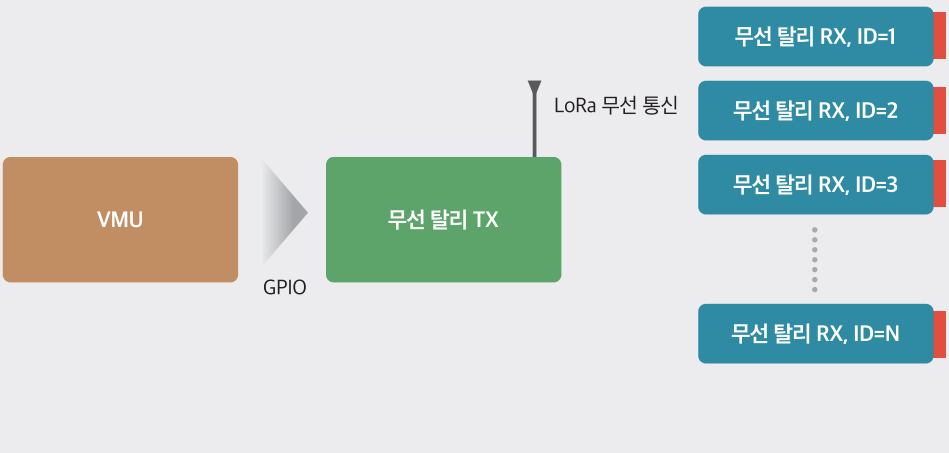

그림 19. 무선 탈리 개요도

카페에서 주문하고 받는 진동벨과 유사한 구조이다. 무선 탈리 TX에서 해당 ID의 탈리에 신호를 주면, 그 탈리는 붉은 LED를 밝히게 된다.

이제 회로를 만들어 보자.

먼저 심볼을 사용하여 Schematic 회로도를 그리고, footprint를 사용하여 PCB 디자인을 한다. 모든 부품들은 시장에서 쉽게 구할 수 있어야 한다.
무료로 사용할 수 있는 회로 디자인 툴은 Ki-CAD 또는 EasyEDA를 사용한다. EasyEDA는 사용자들이 만든 다양한 최신 라이브러리를 사용하기 편리하고, PCB 주문까지 원클릭으로 손쉽게 할 수 있다. 과거에는 회로를 디자인하고, PCB를 제작하는데 많은 노력과 비용이 들었지만, PCB 제작은 국경 없는 시장이 되었다. 온라인으로 주문하고 받는데 1주일도 걸리지 않는다. 소형 PCB들은 배송비 포함 5만 원 이내로 제작이 가능해졌다.

송신용 회로는 VMU와 연동할 GPI 커넥터를 포함하고 있어야겠다. 수신용 회로는 배터리 및 배터리 관련 회로까지 포함되어야 한다.

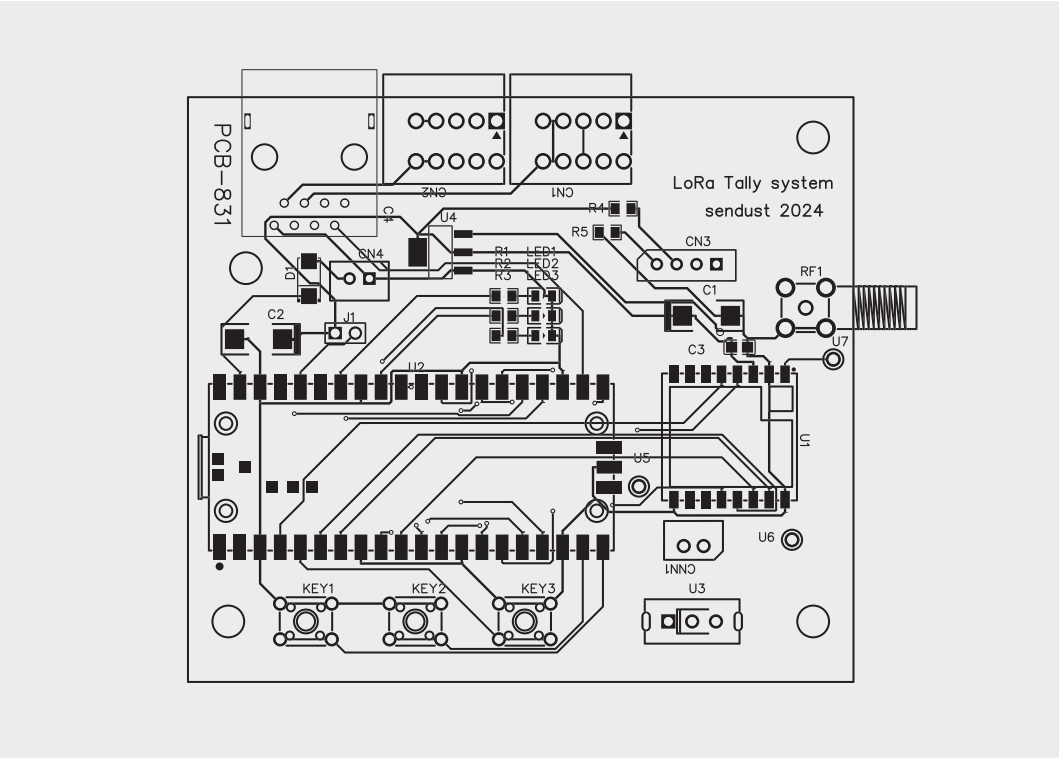

그림 20. 무선 탈리 PCB '송신기'

완성된 송신기용 PCB 모습이다. RJ-45 커넥터를 사용하여 전원 및 GPIO 1~4 입력을 받을 수 있으며, GPIO 5~10은 DINKLE 단자를 사용하였다. Raspberry PI Pico 풀사이즈 보드를 사용했으며, SMA 커넥터를 사용하는 안테나 단자는 우측에 포함되어있다.

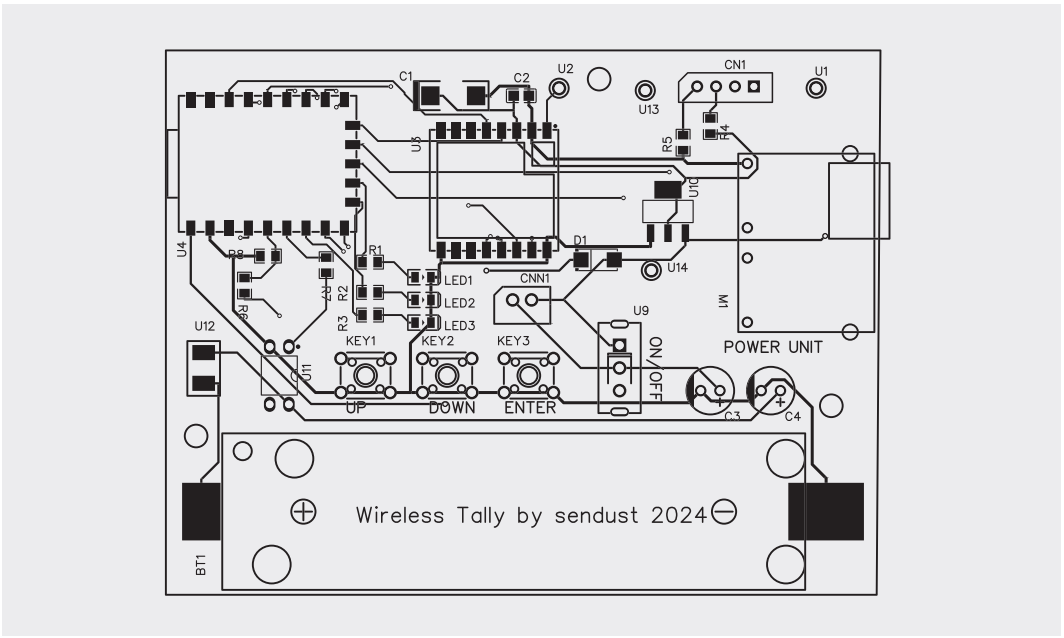

그림 21. 무선 탈리 PCB '수신기'

완성된 수신기의 모습이다. 18650 배터리는 아래쪽에 자리 잡고 있으며, RP2040-ZERO 초소형 개발보드를 사용하였다. 별도의 안테나 없이, PCB 테두리를 휘감는 라우팅 트레이스를 마련함으로써 안테나 역할을 하도록 하였다.

MicroPython으로 송수신기 소프트웨어를 구현해 보자.
MicroPython은 MCU로 구동하는 소형 파이선쯤으로 생각하면 되겠다. MCU에서만 특별히 쓸 수 있는 Machine, NeoPixel, Bluetooth, Network 라이브러리 등이 포함되어 있다. MCU 핀에 대한 입출력을 정의하거나 시리얼 통신 등을 구현할 때는 Machine 모듈을 사용한다.

송신기 소프트웨어 구조는 비교적 간단하다. 주기적으로 VMU의 GPO를 읽은 후 RA-01 RADIO 모듈을 사용하여 GPO 1~10 상태를 송출하면 된다.

클래스 이름	매서드 종류	역할
ssd1306	fill, rect, text, show	OLED 표시
sx127x	setFrequency, setPower, setBW, send	RADIO 모듈 제어
menu	load, save, get_item, get_value	사용자 UI 메뉴 제어
button	read	사용자 버튼 입력
timer_once	set_interval, run	타이머 (1회용)
timer_repeat	set_interval, run, reset, halt	타이머 (반복작업용)
tallys	get_tally	탈리 GPIO 작업

주파수는 400~500MHz 사이에서 사용자가 원하는 주파수를 선택할 수 있도록 하였다. 송출 Power를 조절하여 근거리에서만 사용할 경우도 생각하였다. 탈리 테스트 모드를 추가하여 수신기의 동작 여부를 쉽게 확인할 수 있도록 하였다.

수신기의 소프트웨어는 송신기보다는 복잡하다.

클래스 이름	매서드 종류	역할
ssd1306	fill, rect, text, show	OLED 표시
sx127x	setFrequency, setPower, setBW, send	RADIO 모듈 제어
menu	load, save, get_item, get_value	사용자 UI 메뉴 제어
button	read	사용자 버튼 입력
timer_once	set_interval, run	타이머 (1회용)
timer_repeat	set_interval, run, reset, halt	타이머 (반복작업용)
battery	read, get_level	배터리 잔량 표시
neoled	set_dim, set_on, set_off	탈리 밝기 조절, on/off

송신기와 마찬가지로 주파수 변경이 가능하며, 탈리의 밝기, ID 변경도 가능하도록 만들었다. MicroPython은 내부 파일시스템을 가지고 있다. 메뉴 변경 시 그 값을 json 형식으로 내부 파일시스템에 저장함으로써 전원을 껐다가 켜도 마지막 설정값을 다시 불러올 수 있다.

다음은 제작 중인 TX, RX 모듈을 보여준다.

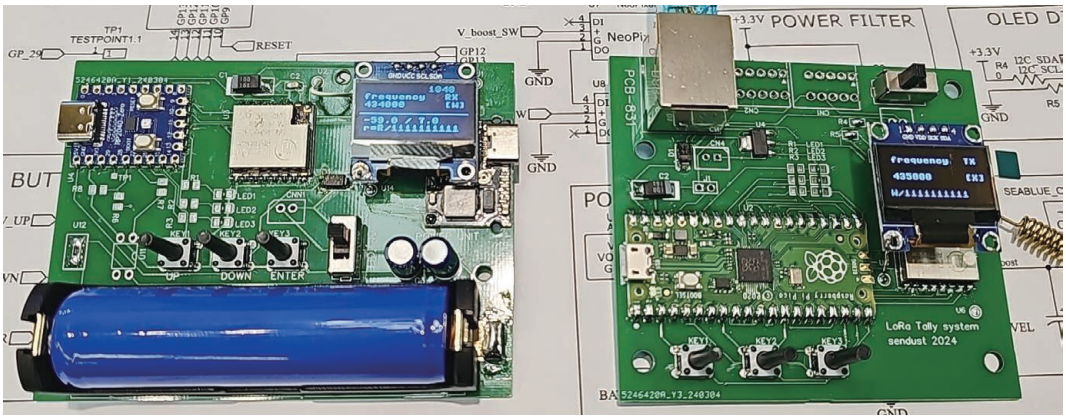

그림 22. 좌 : 수신기 / 우 : 송신기

아직 제작 중인 상태라 미완성된 부분들이 있지만, 디지털 무선통신은 정상적으로 잘 이루어지고 있음을 확인할 수 있다.

지금까지 MCU와 LoRa 무선 모듈을 활용한 무선 탈리 시스템 제작에 대하여 살펴보았다. 수많은 방송장비들이 채택하고 있는 GPIO의 회로 내부를 들여다보았으며, 다양한 모듈들을 결합하여 필요한 기능을 구현하는 새로운 제품을 만들어 낸 셈이다. 시작은 하드웨어 결합으로 출발했지만, 그 하드웨어들은 소프트웨어 세상에서 객체의 형태로 구현되며, 타이머에 의해서 여러 가지 동작을 동시에 수행하게 된다. 마이크로 파이선은 PC에서처럼 스레드를 자유롭게 사용할 수 없다. 타이머에 의존하여 다양한 루틴을 실행하게 된다.

우리 방송기술인들은 AI 시대에 살고 있다. AI에 의해서 루틴 한두 개는 만들 수 있다. 그렇지만 그 루틴을 조화롭게 결합하여 살아 있는 듯한 완성체를 만들어 내는 것은 여전히 설계자인 사람의 몫이다.