Busan Software Meister High School

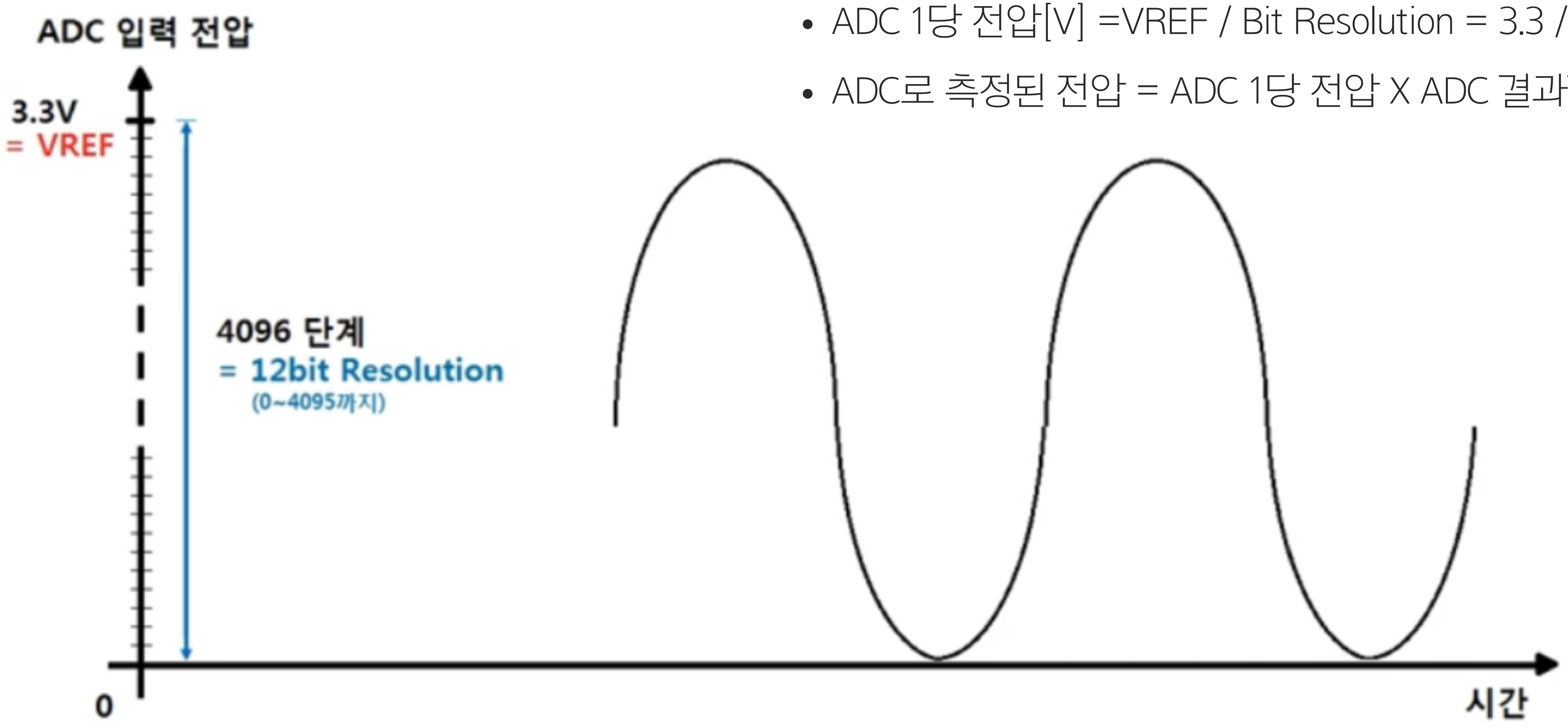
MICROPROCESSOR

2309 양유빈

20230921 마이크로프로세서

ADC 결과 해석

Interpretation of ADC results



ADC의 결과값이 1500이 나왔다면?

- ADC 1당 전압[V] =VREF / Bit Resolution = 3.3 / 4096 = 0.806[mV]
- ADC로 측정된 전압 = ADC 1당 전압 X ADC 결과값 = 0.806 X 1500 = 1.209[V]

주위 밝기에 따라 변화하는 LED

LED that changes depending on the surrounding brightness

기본값

- ADC_MAX   4096
- CDS_FACTOR 2.2
- BASE_R   10 // 회로저항

CDS 저항값

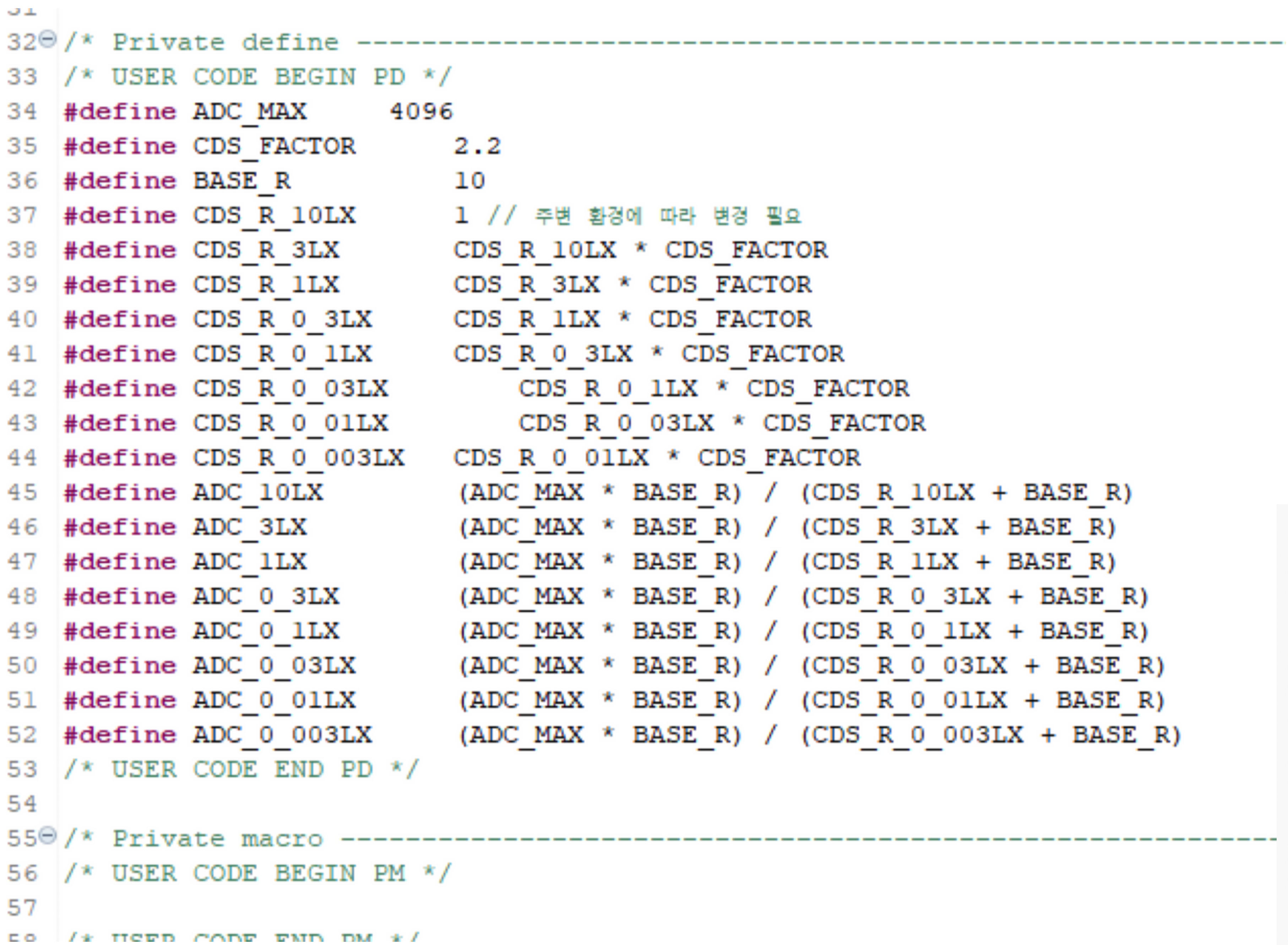
- CDS_R_10LX 35 // CDS저항 - 환경에 맞게 수정해야 될 수도 있음
- CDS_R_3LX CDS_R_10LX * CDS_FACTOR
- CDS_R_1LX CDS_R_3LX * CDS_FACTOR
- CDS_R_0_3LX CDS_R_1LX * CDS_FACTOR
- CDS_R_0_1LX CDS_R_0_3LX * CDS_FACTOR
- CDS_R_0_03LX CDS_R_0_1LX * CDS_FACTOR
- CDS_R_0_01LX CDS_R_0_03LX * CDS_FACTOR
- CDS_R_0_003LX CDS_R_0_01LX * CDS_FACTOR

ADC값

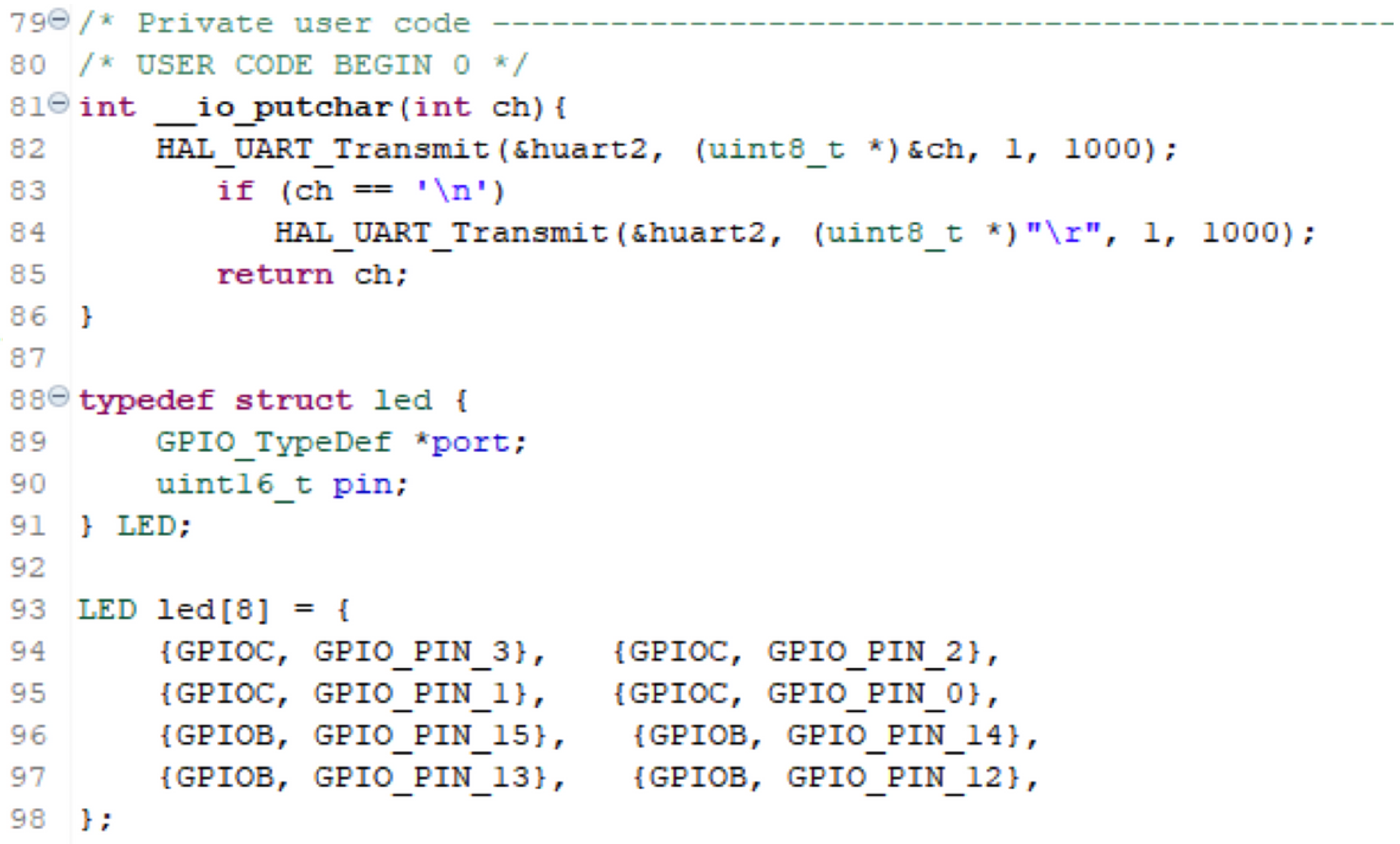
- ADC_10LX   (ADC_MAX * BASE_R)/ (CDS_R_10LX + BASE_R)
- ADC_3LX   (ADC_MAX * BASE_R) / CDS_R_3LX + BASE_R)
- ADC_1LX   (ADC_MAX * BASE_R) / CDS_R_1LX + BASE_R)
- ADC_0_3LX (ADC_MAX * BASE_R) / CDS_R_0_3LX + BASE_R)
- ADC_0_1LX (ADC_MAX * BASE_R) / CDS_R_0_1LX + BASE_R)
- ADC_0_03LX (ADC_MAX * BASE_R) / CDS_R_0_03LX + BASE_R)
- ADC_0_01LX (ADC_MAX * BASE_R) / CDS_R_0_01LX + BASE_R)
- ADC_0_003LX (ADC_MAX * BASE_R) / CDS_R_0_003LX + BASE_R)

주위 밝기에 따라 변화하는 LED

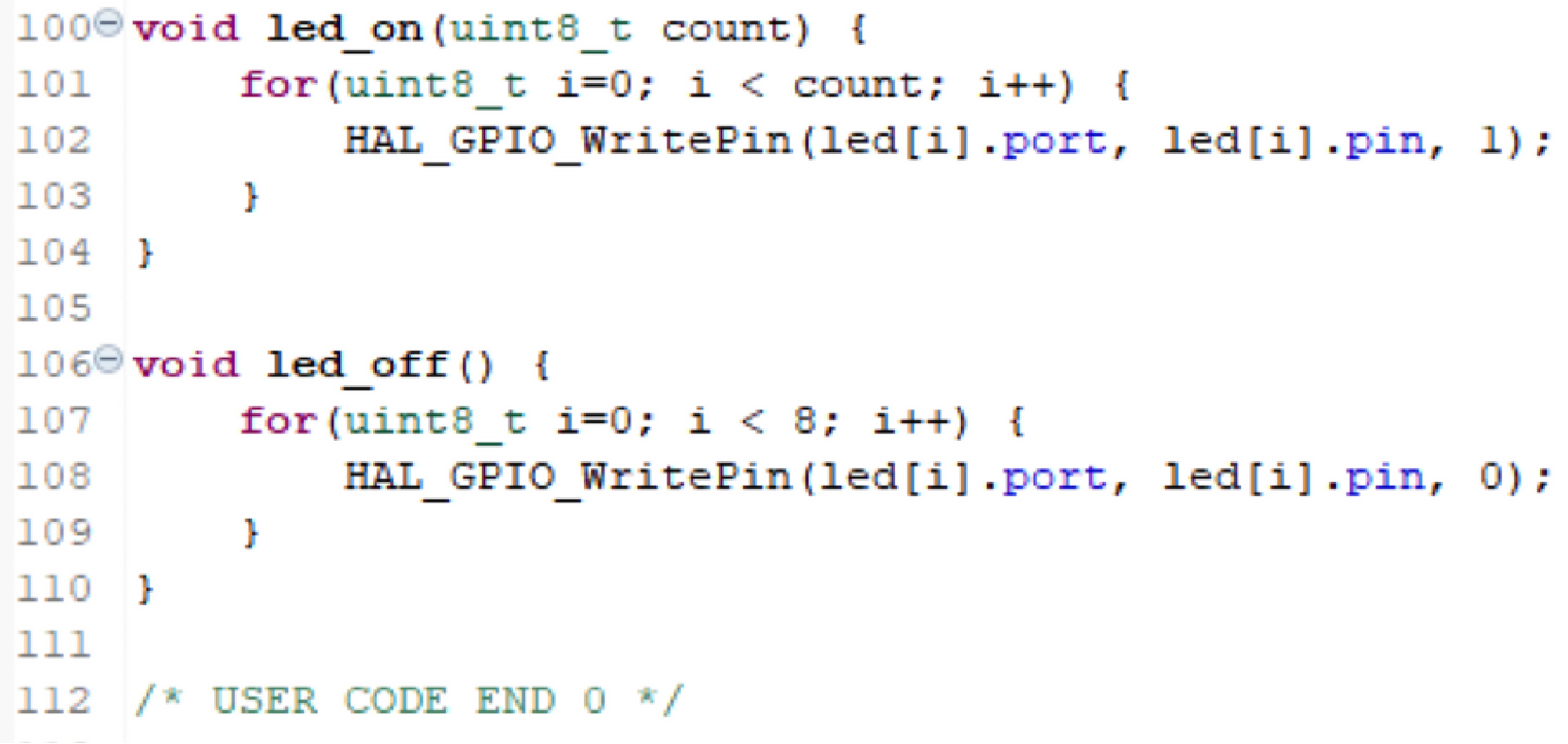
LED that changes depending on the surrounding brightness



```c
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define ADC_MAX      4096
#define CDS_FACTOR       2.2
#define BASE_R           10
#define CDS_R_10LX       1 // 주변 환경에 따라 변경 필요
#define CDS_R_3LX        CDS_R_10LX * CDS_FACTOR
#define CDS_R_1LX        CDS_R_3LX * CDS_FACTOR
#define CDS_R_0_3LX      CDS_R_1LX * CDS_FACTOR
#define CDS_R_0_1LX      CDS_R_0_3LX * CDS_FACTOR
#define CDS_R_0_03LX         CDS_R_0_1LX * CDS_FACTOR
#define CDS_R_0_01LX         CDS_R_0_03LX * CDS_FACTOR
#define CDS_R_0_003LX    CDS_R_0_01LX * CDS_FACTOR
#define ADC_10LX         (ADC_MAX * BASE_R) / (CDS_R_10LX + BASE_R)
#define ADC_3LX          (ADC_MAX * BASE_R) / (CDS_R_3LX + BASE_R)
#define ADC_1LX          (ADC_MAX * BASE_R) / (CDS_R_1LX + BASE_R)
#define ADC_0_3LX        (ADC_MAX * BASE_R) / (CDS_R_0_3LX + BASE_R)
#define ADC_0_1LX        (ADC_MAX * BASE_R) / (CDS_R_0_1LX + BASE_R)
#define ADC_0_03LX       (ADC_MAX * BASE_R) / (CDS_R_0_03LX + BASE_R)
#define ADC_0_01LX       (ADC_MAX * BASE_R) / (CDS_R_0_01LX + BASE_R)
#define ADC_0_003LX      (ADC_MAX * BASE_R) / (CDS_R_0_003LX + BASE_R)
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */
```



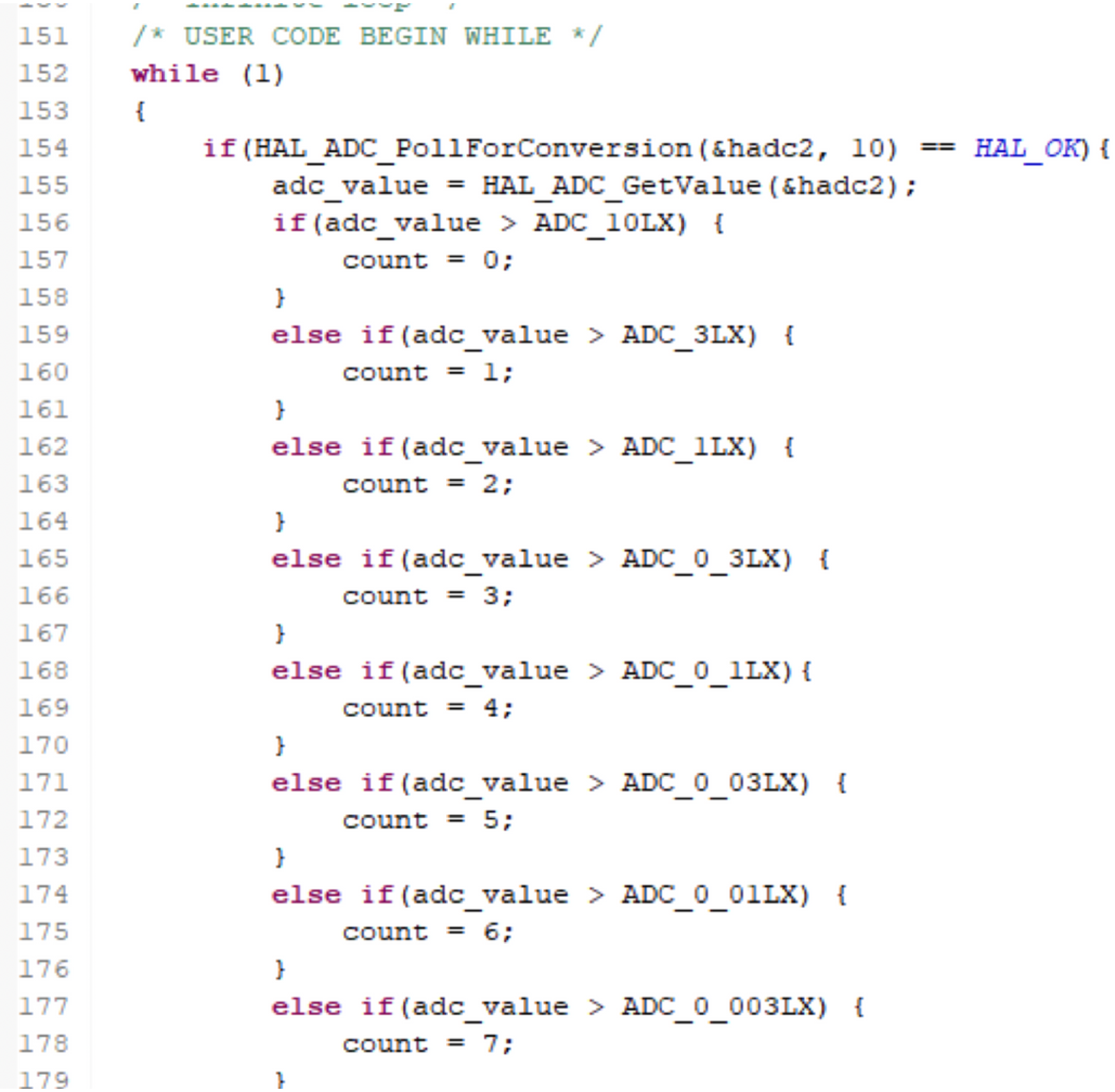
```c
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
int __io_putchar(int ch){
    HAL_UART_Transmit(&huart2, (uint8_t *)&ch, 1, 1000);
       if (ch == '\n')
          HAL_UART_Transmit(&huart2, (uint8_t *)"\r", 1, 1000);
       return ch;
}

typedef struct led {
    GPIO_TypeDef *port;
    uint16_t pin;
} LED;

LED led[8] = {
    {GPIOC, GPIO_PIN_3},    {GPIOC, GPIO_PIN_2},
    {GPIOC, GPIO_PIN_1},    {GPIOC, GPIO_PIN_0},
    {GPIOB, GPIO_PIN_15},    {GPIOB, GPIO_PIN_14},
    {GPIOB, GPIO_PIN_13},    {GPIOB, GPIO_PIN_12},
};
```



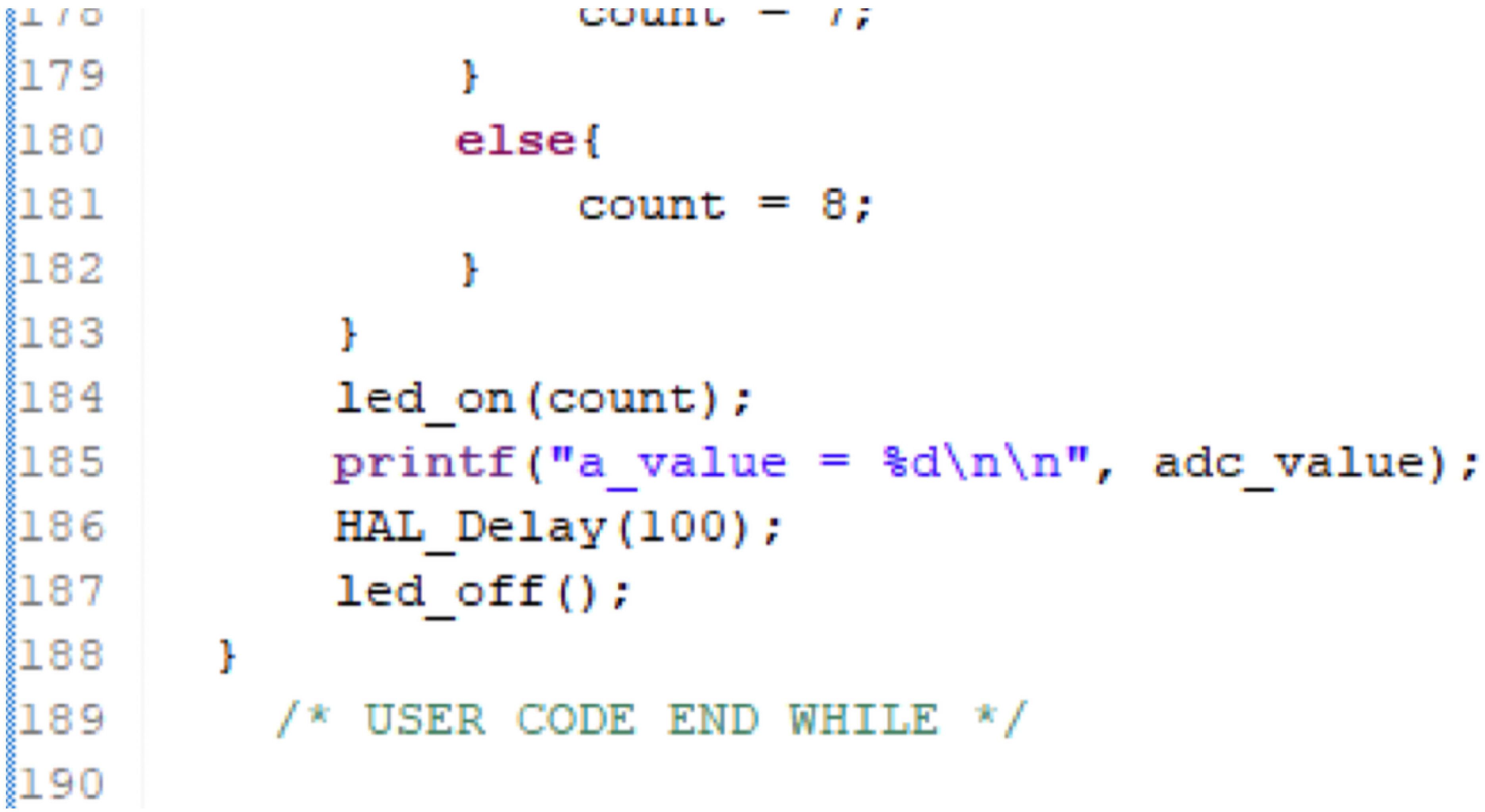
```c
void led_on(uint8_t count) {
    for(uint8_t i=0; i < count; i++) {
        HAL_GPIO_WritePin(led[i].port, led[i].pin, 1);
    }
}

void led_off() {
    for(uint8_t i=0; i < 8; i++) {
        HAL_GPIO_WritePin(led[i].port, led[i].pin, 0);
    }
}

/* USER CODE END 0 */
```

주위 밝기에 따라 변화하는 LED

LED that changes depending on the surrounding brightness



```c
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
      if(HAL_ADC_PollForConversion(&hadc2, 10) == HAL_OK){
          adc_value = HAL_ADC_GetValue(&hadc2);
          if(adc_value > ADC_10LX) {
              count = 0;
          }
          else if(adc_value > ADC_3LX) {
              count = 1;
          }
          else if(adc_value > ADC_1LX) {
              count = 2;
          }
          else if(adc_value > ADC_0_3LX) {
              count = 3;
          }
          else if(adc_value > ADC_0_1LX){
              count = 4;
          }
          else if(adc_value > ADC_0_03LX) {
              count = 5;
          }
          else if(adc_value > ADC_0_01LX) {
              count = 6;
          }
          else if(adc_value > ADC_0_003LX) {
              count = 7;
          }
```

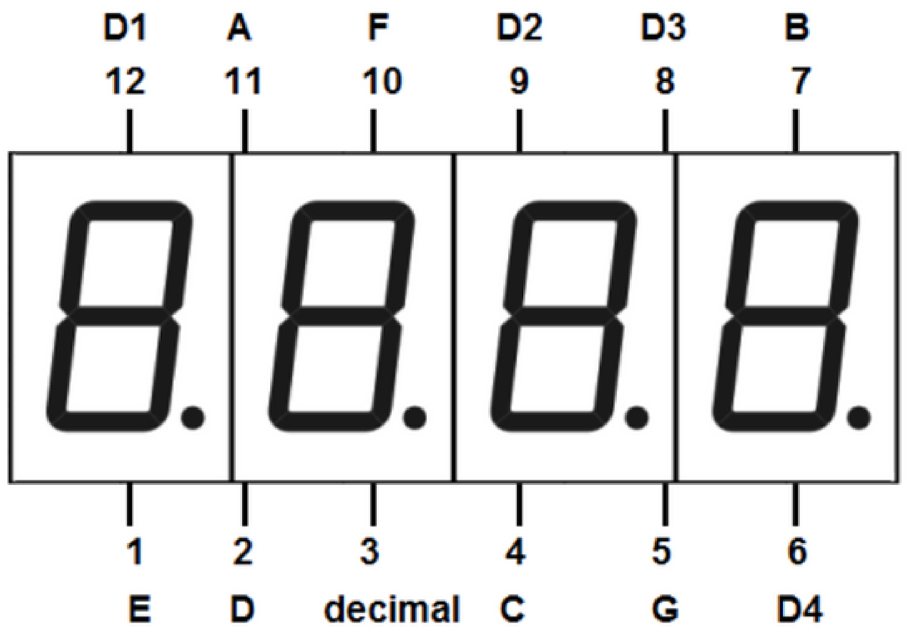


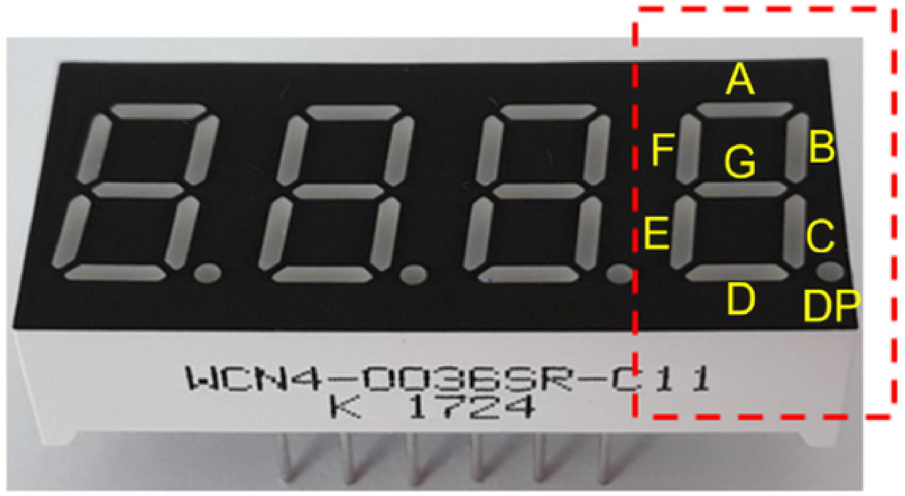
```c
              count = 7;
          }
          else{
              count = 8;
          }
      }
      led_on(count);
      printf("a_value = %d\n\n", adc_value);
      HAL_Delay(100);
      led_off();
  }
    /* USER CODE END WHILE */
```

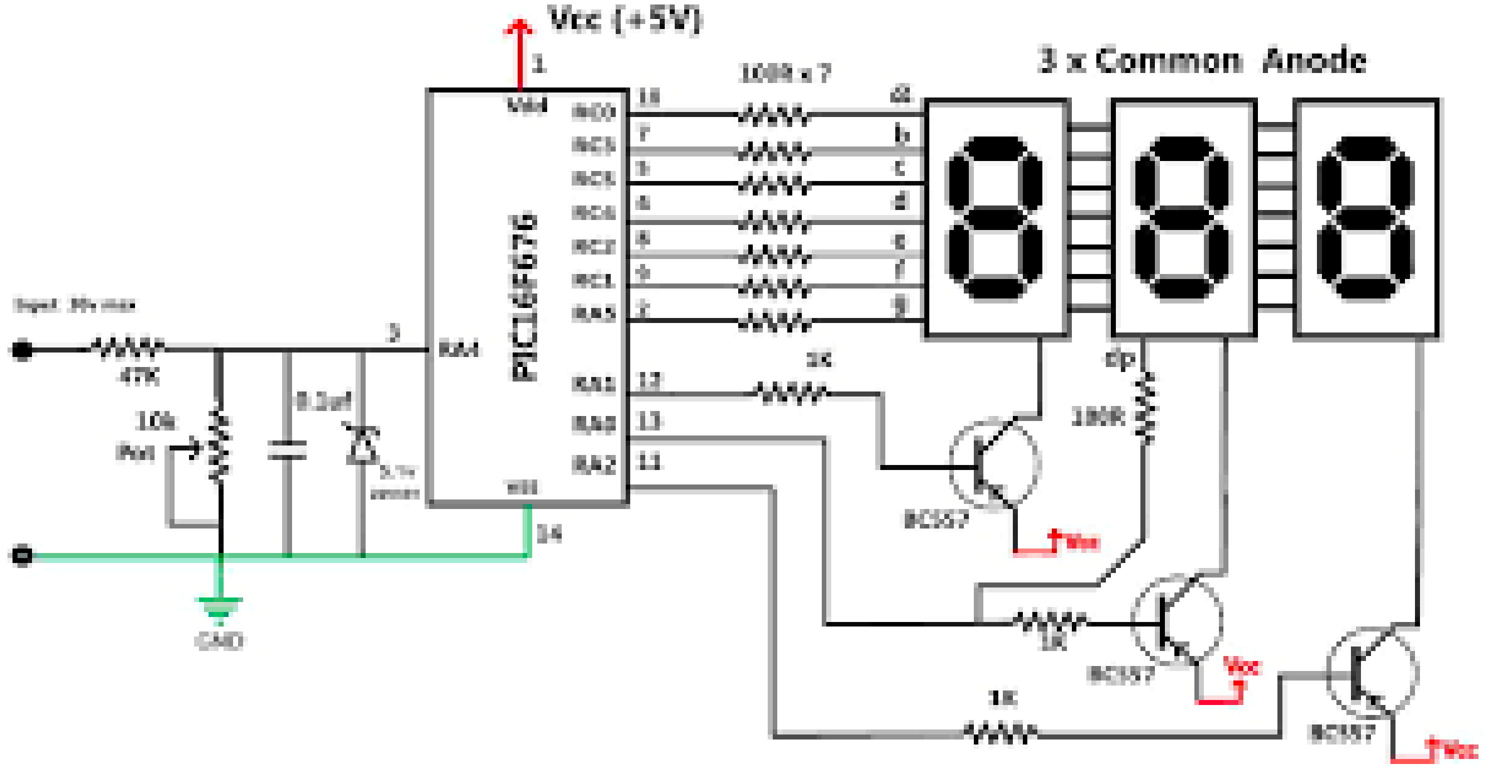
FND

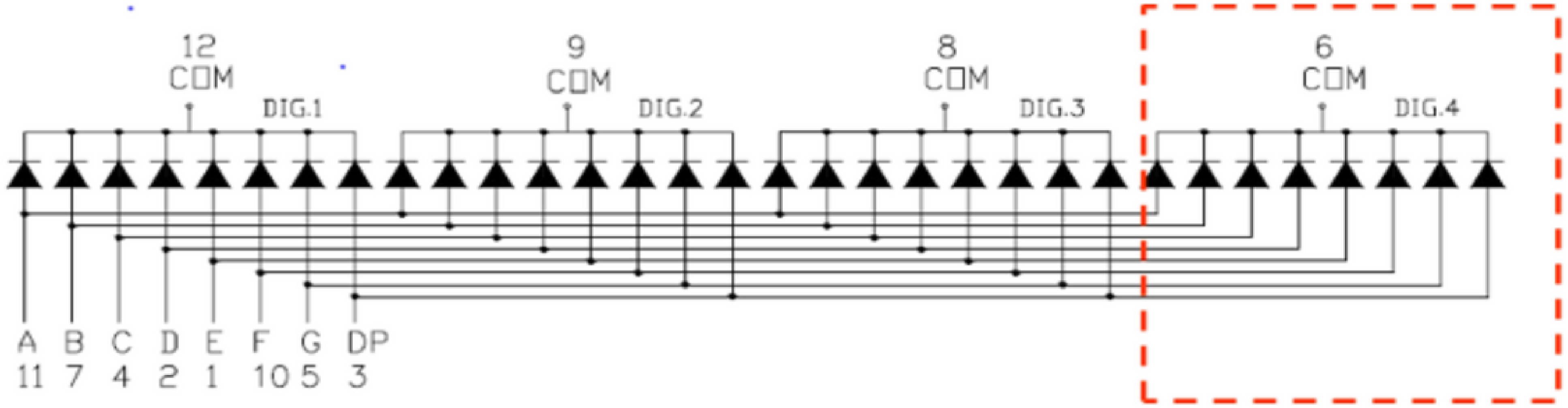
Flexible Numeric Display

- 가변 숫자 표시기
- ‘7-segment 디스플레이’라고도 함
- 보통 1~4개 정도의 숫자를 표시
- CA(Common Anode) 및 CC(Common Cathode) 타입
- 엘리베이터 층수 표시기 등에 사용
- JKIT-Nucleo-64에서는 4-digit CC FND 사용


D1
12
A
11
F
10
D2
9
D3
8
B
7
1
E
2
D
3
decimal
4
C
5
G
6
D4




Vcc (+5V)
1
100R x 7
3 x Common Anode
PIC16F676
Vdd
RC0
RC5
RC5
RC4
RC2
RC1
RA5
RA4
RA1
RA0
RA2
Vss
10
7
9
6
8
9
2
3
12
13
11
14
a
b
c
d
e
f
g
dp
47K
10k
Pot
0.1uf
5.1V zener
1K
180R
BC557
Vcc
GND


12
COM
DIG.1
9
COM
DIG.2
8
COM
DIG.3
6
COM
DIG.4
A B C D E F G DP
11 7 4 2 1 10 5 3