보충

Arduino EEPROM Control

EEPROM 응용 - 모터 성능과 IR센서 감도 보정값 활용

EEPROM 활용 - 모터 성능과 IR센서 감도의 보정값 쓰고 읽기

EEPROM에 값을 저장하고 읽기

모터 성능 보정 비율을 EEPROM에 저장하고, 저장된 값을 읽어서 사용하기

EEPROM에 값을 저장하고 읽기

EEPROM 이란?

- 메모리 : 데이터가 저장되는 공간
  - 휘발성 메모리 – 장치의 전원이 꺼지면 저장되었던 데이터가 리셋 되는 메모리
  - **비휘발성 메모리** – 장치의 전원이 꺼져도 저장된 데이터가 그대로 남아있는 메모리
    EEPROM(Electrically Erasable Programmable Read-Only Memory)

- Arduino EEPROM 크기: Uno/Nano= 1K BYTE, Mega= 4K BYTE

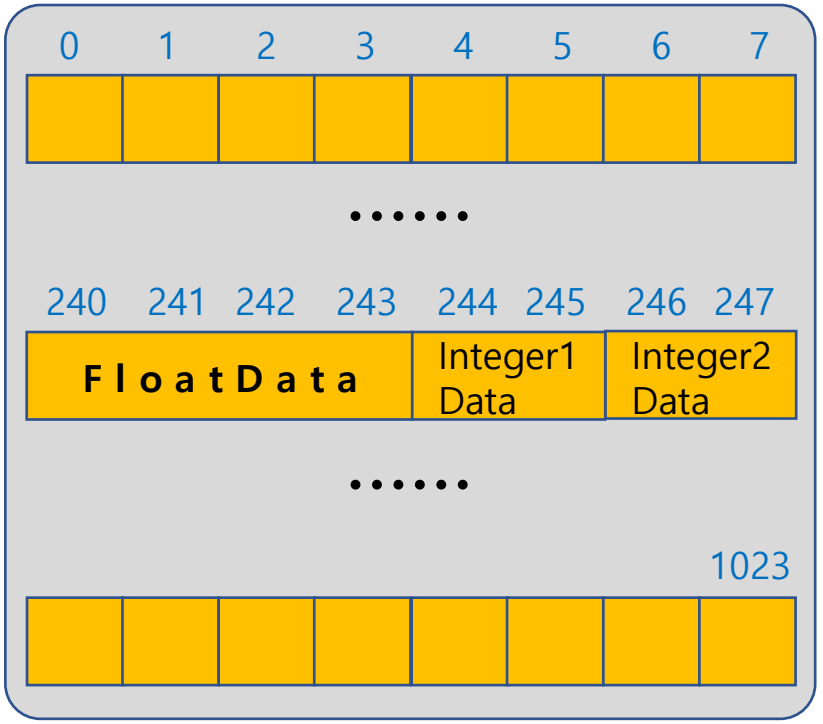
- 아두이노 소스코드 사용 예 (EEPROM에 BYTE 값 쓰기와 읽기)

```
#include <EEPROM.h>  // EEPROM 사용에 필요한 헤더파일 포함하기

EEPROM.write( address, value );  // address(오프셋) 위치에 1바이트 value(값) 쓰기

value = EEPROM.read( address );  // address(오프셋) 위치에 저장된 1바이트 값을 value 변수로 읽기
```

EEPROM에 값을 저장하고 읽기

EEPROM에 특정 값을 쓰고 읽기

EEPROM 저장 구조 (Size= 1KB, 1칸= 1BYTE)




```
#define EEPROM_TEST_FLOAT     240 // 테스트용 데이터 저장 시작 위치(offset)
#define EEPROM_TEST_INT1      (EEPROM_TEST_FLOAT + 4) // 실수형= 4바이트
#define EEPROM_TEST_INT2      (EEPROM_TEST_INT1 + 2)  // 정수형= 2바이트
```

```
float FloatData = 3.14; // 실수형 데이터는 4바이트 차지함
int   Integer1Data = -12345; // 정수형 데이터는 2바이트 차지함
unsigned int   Integer2Data = 65535; // 0 ~ 65535 (2바이트)
```

◆ EEPROM의 240 번째 바이트부터 저장할 데이터 3 종류

- FloatData = 3.14 (실수형, 4바이트 저장 공간이 필요)
- Interger1Data = -12345 (정수형, 2바이트)
- Interger2Data = 65535 (부호가 없는 정수형, 2바이트)

◆ EEPROM 테스트 동작 순서

① 240 번지부터 차례로 3가지 데이터를 저장합니다.
② 3가지 변수의 값을 모두 0으로 초기화 합니다.
③ 저장한 값들을 읽어서 시리얼 모니터로 출력합니다.

EEPROM에 값을 저장하고 읽기

EEPROM에 특정 값을 쓰고 읽기

소스: EEPROM-01-Basic.ino

```
15 void  WriteData() // EEPROM에 데이터 쓰기
16 {
17   // 데이터 유형에 따라 자동으로 저장할 때 --> put 함수 사용
18
19   EEPROM.put( EEPROM_TEST_FLOAT, FloatData );
20   EEPROM.put( EEPROM_TEST_INT1, Integer1Data );
21
22   // 데이터를 1-BYTE 단위로 분리해서 저장할 때 --> write 함수 사용
23
24   byte  byte0 = Integer2Data & 0x00FF;  // 하위 바이트
25   byte  byte1 = (Integer2Data & 0xFF00) >> 8; // 상위 바이트
26
27   EEPROM.write( EEPROM_TEST_INT2 + 0, byte0 );
28   EEPROM.write( EEPROM_TEST_INT2 + 1, byte1 );
29 }
30
31 void  ReadData()  // EEPROM에 저장된 데이터 읽기
32 {
33   // 데이터 유형에 따라 자동으로 값을 읽을 때 --> get 함수 사용
34
35   EEPROM.get( EEPROM_TEST_FLOAT, FloatData );
36   EEPROM.get( EEPROM_TEST_INT1, Integer1Data );
37
38   // 데이터를 BYTE 단위로 분리해서 값을 읽을 때 --> read 함수 사용
39
40   byte  byte0 = EEPROM.read( EEPROM_TEST_INT2 + 0 );
41   byte  byte1 = EEPROM.read( EEPROM_TEST_INT2 + 1 );
42
43   Integer2Data = ((unsigned int)byte1 << 8) + byte0;
44 }
```

```
48 void setup()
49 {
50   Serial.begin( 9600 ); // 시리얼 통신 속도 설정
51
52   Serial.print( "Writing values to EEPROM : " );
53   WriteData(); // EEPROM에 데이터 쓰기
54   Serial.println( "done!" );
55
56   // EEPROM에 쓴 값들을 읽기 전에 모두 0으로 초기화
57   FloatData = 0.0;
58   Integer1Data = 0;
59   Integer2Data = 0;
60   Serial.println( "All values are initialized to 0." );
61
62   Serial.print( "Reading values from EEPROM : " );
63   ReadData();  // EEPROM에 저장된 데이터 읽기
64   Serial.println( "done!" );
65
66   Serial.print( "FloatData= " );  // 읽은 데이터 값들 출력
67   Serial.print( FloatData );
68   Serial.print( ", Integer1Data= " );
69   Serial.print( Integer1Data );
70   Serial.print( ", Integer2Data= " );
71   Serial.println( Integer2Data );
72   Serial.println();
73 }
```

```
COM3
Writing values to EEPROM : done!
All values are initialized to 0.
Reading values from EEPROM : done!
FloatData= 3.14, Integer1Data= -12345, Integer2Data= 65535
```

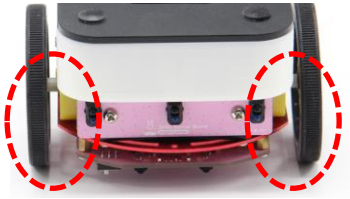
EEPROM 활용 - 바퀴모터 성능 조정값

모터 성능비율을 사용하여 로봇의 직진성 보정하기

좌우 모터의 성능에 차이가 있는 경우 좌우 모터를 같은 세기로 회전시켜도 로봇의 전진 또는 후진 방향이 직선으로 되지않고 곡선처럼 휘게 됩니다.

성능 보정하기 전
(왼쪽 모터가 약함)

Power1Ratio = 1.00
Power2Ratio = 1.00



오른쪽 모터 성능 비율을 0.01씩 낮추고 실행하기를 반복하면서 바르게 직진이 잘 되는 비율을 찾아서 EEPROM에 저장합니다.

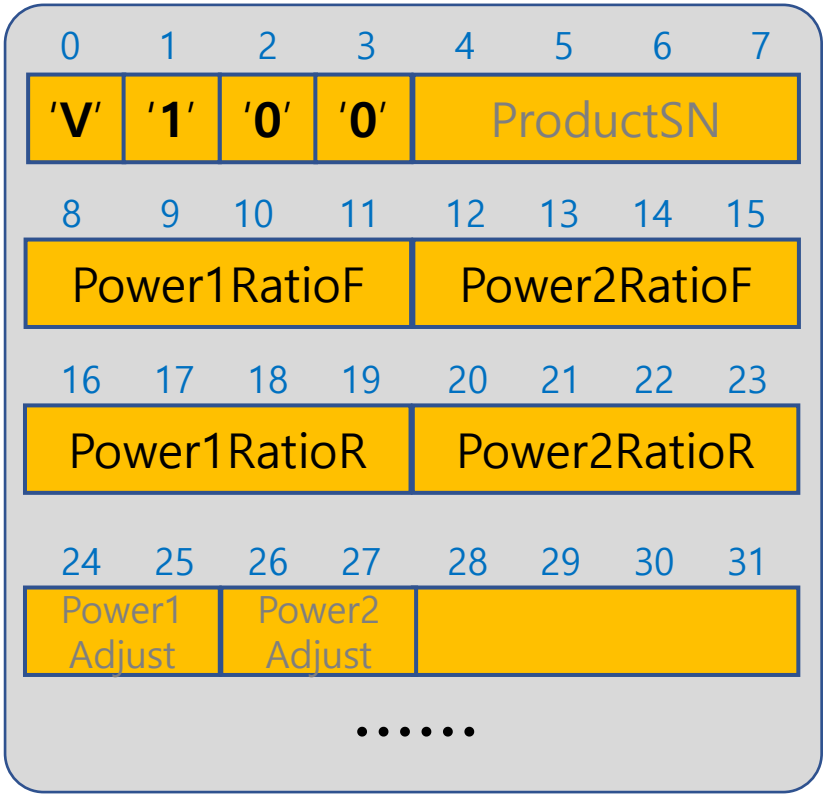
성능 보정한 후
(양쪽 모터 균형)

Power1Ratio = 1.00
Power2Ratio = 0.96

EEPROM 활용 - 바퀴모터 성능 조정값

모터 성능 보정을 위한 EEPROM 데이터 저장 구조

EEPROM 데이터 저장 구조 (모터 성능 보정용)



EEPROM_BASE
(= EEPROM_DATA_VER0)

"V100"은 Header 값으로 고정
ProductSN은 제품 번호 (미사용)

EEPROM_POWER_RATIOF
전진용 좌/우 모터 성능비율

1번 모터는 왼쪽 (전진 보정)
2번 모터는 오른쪽

EEPROM_POWER_RATIOR
후진용 좌/우 모터 성능비율

1번 모터는 왼쪽 (후진 보정)
2번 모터는 오른쪽

EEPROM_POWER_ADJUST
전진용 좌/우 모터 성능비율

1번 모터는 왼쪽 (추가 보정용, 미사용)
2번 모터는 오른쪽 (미사용)
(→ 필요한 경우 추가로 사용 가능)

※ EEPROM에 유효한 보정 데이터가 저장되어 있는지 확인하고, 있다면 그 값을 활용하여 직진하는 프로그램을 만들어보세요.

EEPROM 활용 - 바퀴모터 성능 조정값

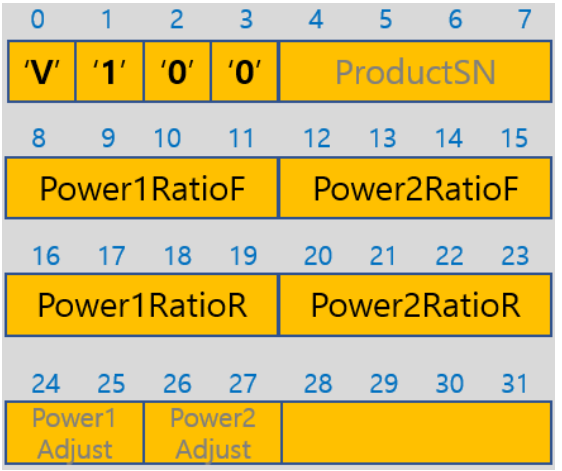
모터 성능 보정을 위한 EEPROM 데이터 활용 코딩 실습 (1)

소스: EEPROM-02-Motor.ino

```
1  ////////////////////  EEPROM 데이터 쓰기/읽기
2
3  #include <EEPROM.h>
4
5  #define EEPROM_BASE            0
6  #define EEPROM_DATA_VER0       (EEPROM_BASE + 0)   // 'V'
7  #define EEPROM_DATA_VER1       (EEPROM_BASE + 1)   // '1'
8  #define EEPROM_DATA_VER2       (EEPROM_BASE + 2)   // '0'
9  #define EEPROM_DATA_VER3       (EEPROM_BASE + 3)   // '0'
10
11 #define EEPROM_PRODUCT_SN      (EEPROM_BASE + 4)   // Reserved
12 #define EEPROM_POWER_RATIOF    (EEPROM_BASE + 8)   // 1.00, 1.00
13 #define EEPROM_POWER_RATIOR    (EEPROM_BASE + 16)  // Reserved
14 #define EEPROM_POWER_ADJUST    (EEPROM_BASE + 24)  // Reserved
15
16 boolean IsEepromDataValid = false;
17
18 boolean CheckEepromDataHeader()
19 {
20   if( (EEPROM.read( EEPROM_DATA_VER0 ) == 'V') &&
21       (EEPROM.read( EEPROM_DATA_VER1 ) == '1') &&
22       (EEPROM.read( EEPROM_DATA_VER2 ) == '0') &&
23       (EEPROM.read( EEPROM_DATA_VER3 ) == '0') )
24       return  true;
25
26   return  false;
27 }
28
```

좌우 모터의 성능 비율을 읽는 함수

```
29 ////////////  좌/우 바퀴모터 속력 보정비율 (속력 = 파워 * 보정비율)
30
31 // 전진 속력 보정비율 [0 .. 1], 대부분 [0.94 ~ 1.00] 사이로 설정함
32 // 성능이 나쁜쪽을 1.0으로, 좋은쪽 비율을 균형에 맞도록 낮게 설정합니다.
33 float Power1RatioF = 1.00;  // 왼쪽 모터 (기본= 1.0)
34 float Power2RatioF = 1.00;  // 오른쪽 모터 (기본= 1.0)
35
36 void  ReadPowerRatio()  // EEPROM에 저장된 모터 성능비율 읽기
37 {
38   EEPROM.get( EEPROM_POWER_RATIOF, Power1RatioF );
39   EEPROM.get( EEPROM_POWER_RATIOF + 4, Power2RatioF );
40 }
```



EEPROM 활용 - 바퀴모터 성능 조정값

모터 성능 보정을 위한 EEPROM 데이터 활용 코딩 실습 (2)

소스: EEPROM-02-Motor.ino

```
63 void  drive(int dir1, int power1, int dir2, int power2)
64 {
65   boolean dirHighLow1, dirHighLow2;
66   int     p1, p2;
67
68   if(dir1 == FORWARD)  // 1번(왼쪽)모터 방향
69     dirHighLow1 = HIGH;
70   else // BACKWARD
71     dirHighLow1 = LOW;
72   p1 = power1 * Power1RatioF;
73
74   if(dir2 == FORWARD)  // 2번(오른쪽)모터
75     dirHighLow2 = LOW;
76   else // BACKWARD
77     dirHighLow2 = HIGH;
78   p2 = power2 * Power2RatioF;
79
80   digitalWrite(pinDIR1, dirHighLow1);
81   analogWrite(pinPWM1, p1);
82
83   digitalWrite(pinDIR2, dirHighLow2);
84   analogWrite(pinPWM2, p2);
85 }
86
87 void  Forward( int power )  // 전진
88 {
89   drive(FORWARD, power, FORWARD, power);
90 }
```

EEROM 헤더가 있는 경우 저장된 모터 성능 비율 읽기

```
133   // EEPROM에 저장되어 있는 데이터(좌우 모터와 바닥면의 IR 조정값) 읽기
134   if( IsEepromDataValid = CheckEepromDataHeader() )
135   {
136     // 이전에 EEPROM에 저장된 올바른 데이터가 있음
137
138     ReadPowerRatio(); // 좌우 모터 속도 조정 비율값 읽기
139
140     Serial.print( "Power1RatioF= " );
141     Serial.print( Power1RatioF );       // 왼쪽 모터 성능 비율
142     Serial.print( ", Power2RatioF= " );
143     Serial.println( Power2RatioF );     // 오른쪽 모터 성능 비율
144   }
145   else
146   {
147     Serial.println( "Data not found from EEPROM." );
148   }
```

COM3

```
Data not found from EEPROM.
```

COM3

```
Power1RatioF= 0.95, Power2RatioF= 1.00
```