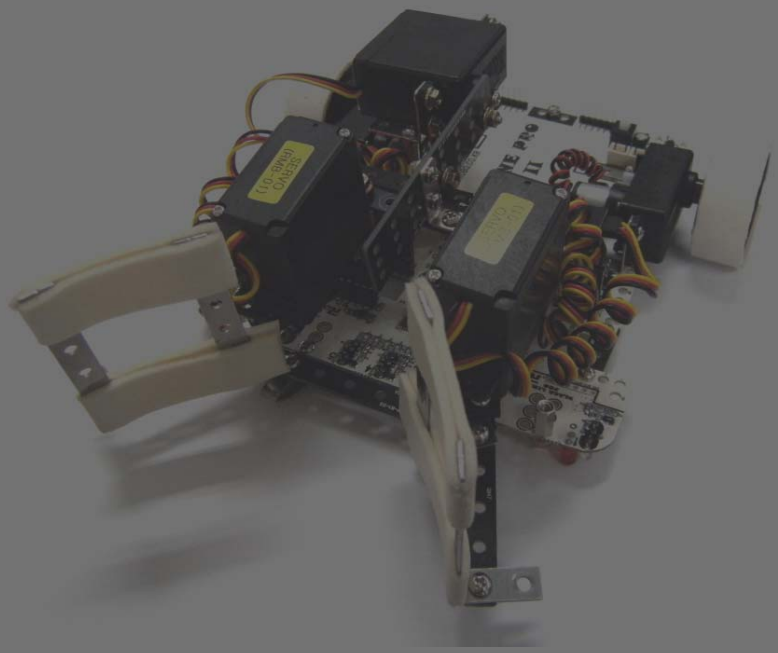
2009

2009 트랜스 포터 매뉴얼

프로그램형 라인트레이서

Black Line Pro 매뉴얼



Yoon-Ki Hong
RoboRobo
2009-07-03

STANDBY();

그립을 모아줌

명령어 설명 그립을 모아줌으로써 주행에 안정을 줄 수 있습니다. 그러나 목표물을 잡을 정도의 간격으로 모아지지 않기 때문에 목표물을 잡을 때는 아래 설명될 'GRIP()' 을 사용합니다.

예 제 "main.h"

```
Int main(void)
{
        start(1,130);           // 크로스 때 마다 buzzer 울림, pc 크로스 시간 0.13 초로 설정
        standBy();              // 그립을 모아줌
        line(pp,ff,20,0);       // 교차로가 나올 때까지 앞으로 라인트레이싱
        end();
        return 0;
}
```

GET (① 속도, ② 지연시간);

목표물을 잡는 함수

명령어 설명 목표물이 있는 직선 크로스에서 본 함수를 호출 하면 목표물을 잡는 시퀀스를 진행 합니다. 첫번째 인자 '① 속도'는 목표물이 있는 지점에 찾아가는 속도를 조절 합니다. 두번째 인자 '② 지연시간'은 그립이 모아지는 지연 시간을 설정하며 '25'가 정당하나 조절 할 필요가 있을 때, 값을 높이면 느려지고 줄이면 빨라집니다.

예 제

```
#include "main.h"
int main(void)
{
        start(1,130);           // 크로스 때 마다 buzzer 울림, pc 크로스 시간 0.13 초로 설정
        line(pp,ff,20,0);       // 교차로가 나올 때까지 앞으로 라인트레이싱
        get(7, 25);             // 목표물 전 교차로에서부터 7 의 속도로 시작하여 목표물에 도달 후,
                                // 25 의 지연 시간으로 그립을 닫는다.
        right(10, 0);           // 오른쪽으로 회전한다.
        right(10, 0);           // 오른쪽으로 회전한다.
        line(pp,ff,20,0);       // 교차로가 나올 때까지 앞으로 라인트레이싱
        line(pp,ff,20,0);       // 교차로가 나올 때까지 앞으로 라인트레이싱
        end();
        return 0;
}
```

WBWB(① 타겟의 위치, ② 배터리 상태, ③ 배터리 상태, ④ 지연시간);

목표물을 목적지에 놓는 함수

명령어 설명 목적지가 있는 직전 크로스에서 호출 하면 목표물을 첫번째 인자 '① 타겟의 위치'에서 지시한 번호에 놓여지게 되어 있는 함수로써 두번째 인자 '② 배터리 상태'는 만약 목표물을 놓으려 하는데 목표물이 중앙에 오지 않게 되면, 중앙에 놓여 지도록 조절 하는 인자 입니다. 인자는 100 을 기준으로 진행하던 방향으로 지나쳐 목표물을 놓을 경우 높여주고 너무 뒤로 물러 나서 놓게 되면 낮춰주면 됩니다. 세번째 인자 '③ 배터리 상태'는 목표물을 놓고 목적지에서 빠져나오는 양을 조절하는 함수로 100 을 기준으로 본인이 원하는 만큼 빠져 나오지 않을 경우 높이거나 줄이면 됩니다. 마지막으로 네번째 인자 '④ 지연시간'은 그립을 너무 빨리 벌리게 되면 목표물이 튕겨져 날라가거나 쓰러지는 경우가 있어 속도를 적당하게 조절 할 수 있도록 하였습니다.

파라미터 범위

① 타겟의 위치 : 1 ~ 4

➔ 진입로를 기준으로 원하는 지정한 눈금 번호에 목표물을 놓습니다.

② 배터리 상태 : 0 ~ 300 (100)

➔ 배터리의 상태에 따라 목표물을 두는 위치가 변할 수 있기 때문에 100 을 기준으로 모터의 힘이 과도 할 때는 100 보다 낮추고, 부족 할 때는 100 보다 높여서 사용합니다. 보통 100 을 입력합니다.

③ 배터리 상태 : 0 ~ 300 (100)

➔ 배터리의 상태에 따라 목적지에서 빠져나오는 양이 변 할 수가 있습니다. 이는 100 을 기준으로 모터의 힘이 과도 할 때는 100 보다 낮추고, 부족 할 때는 100 보다 높여서 사용합니다. 보통 100 을 입력합니다.

④ 지연 시간 : 0 ~ 100 (25)

➔ 그립을 벌리는 시간을 설정합니다. 25 를 기준으로 높이거나 줄이시면 됩니다.

예 제

```
#include “main.h”
Int main(void)
{
        start(1,130);                   // 크로스 때 마다 buzzer 울림, pc 크로스 시간 0.13 초로 설정
        standby();                      // 그립을 모아줌
        line(pp,ff,20,0);               // 교차로가 나올 때까지 앞으로 라인트레이싱
        get(7, 25);                     // 목표물 전 교차로에서부터 시작하여 목표물에 도달 후 집는다.
        right(10, 0);                   // 오른쪽으로 회전한다.
        right(10, 0);                   // 오른쪽으로 회전한다.
        line(pp,ff,20,0);               // 교차로가 나올 때까지 앞으로 라인트레이싱
        line(pp,ff,20,0);               // 교차로가 나올 때까지 앞으로 라인트레이싱
        wbwb(3, 100, 100, 25);          // 목적지의 3 번째 칸에 목표물을 놓고 되돌아 나옴
        end();
        return 0;
}
```

BWBW(① 타겟의 위치, ② 배터리 상태, ③ 배터리 상태, ④ 지연시간);

목표물을 목적지에 놓는 함수

명령어 설명 목적지가 있는 직전 크로스에서 호출 하면 목표물을 첫번째 인자 '① 타겟의 위치'에서 지시한 번호에 놓여지게 되어 있는 함수로써 두번째 인자 '② 배터리 상태'는 만약 목표물을 놓으려 하는데 목표물이 중앙에 오지 않게 되면, 중앙에 놓여 지도록 조절 하는 인자 입니다. 인자는 100 을 기준으로 진행하던 방향으로 지나쳐 목표물을 놓을 경우 높여주고 너무 뒤로 물러 나서 놓게 되면 낮춰주면 됩니다. 세번째 인자 '③ 배터리 상태'는 목표물을 놓고 목적지에서 빠져나오는 양을 조절하는 함수로 100 을 기준으로 본인이 원하는 만큼 빠져 나오지 않을 경우 높이거나 줄이면 됩니다. 마지막으로 네번째 인자 '④ 지연시간'은 그립을 너무 빨리 벌리게 되면 목표물이 튕겨져 날라가거나 쓰러지는 경우가 있어 속도를 적당하게 조절 할 수 있도록 하였습니다.

파라미터 범위

⑤ 타겟의 위치 : 1 ~ 4

➔ 진입로를 기준으로 원하는 지정한 눈금 번호에 목표물을 놓습니다.

⑥ 배터리 상태 : 0 ~ 300 (100)

➔ 배터리의 상태에 따라 목표물을 두는 위치가 변할 수 있기 때문에 100 을 기준으로 모터의 힘이 과도 할 때는 100 보다 낮추고, 부족 할 때는 100 보다 높여서 사용합니다. 보통 100 을 입력합니다.

⑦ 배터리 상태 : 0 ~ 300 (100)

➔ 배터리의 상태에 따라 목적지에서 빠져나오는 양이 변 할 수가 있습니다. 이는 100 을 기준으로 모터의 힘이 과도 할 때는 100 보다 낮추고, 부족 할 때는 100 보다 높여서 사용합니다. 보통 100 을 입력합니다.

⑧ 지연 시간 : 0 ~ 100 (25)

➔ 그립을 벌리는 시간을 설정합니다. 25 를 기준으로 높이거나 줄이시면 됩니다.

예 제

```
#include “main.h”
Int main(void)
{
        start(1,130);                   // 크로스 때 마다 buzzer 울림, pc 크로스 시간 0.13 초로 설정
        standby();                      // 그립을 모아줌
        line(pp,ff,20,0);               // 교차로가 나올 때까지 앞으로 라인트레이싱
        get(7, 25);                     // 목표물 전 교차로에서부터 시작하여 목표물에 도달 후 집는다.
        right(10, 0);                   // 오른쪽으로 회전한다.
        right(10, 0);                   // 오른쪽으로 회전한다.
        line(pp,ff,20,0);               // 교차로가 나올 때까지 앞으로 라인트레이싱
        line(pp,ff,20,0);               // 교차로가 나올 때까지 앞으로 라인트레이싱
        bwbw(3, 100, 100, 25);          // 목적지의 3 번째 칸에 목표물을 놓고 되돌아 나옴
        end();
        return 0;
}
```

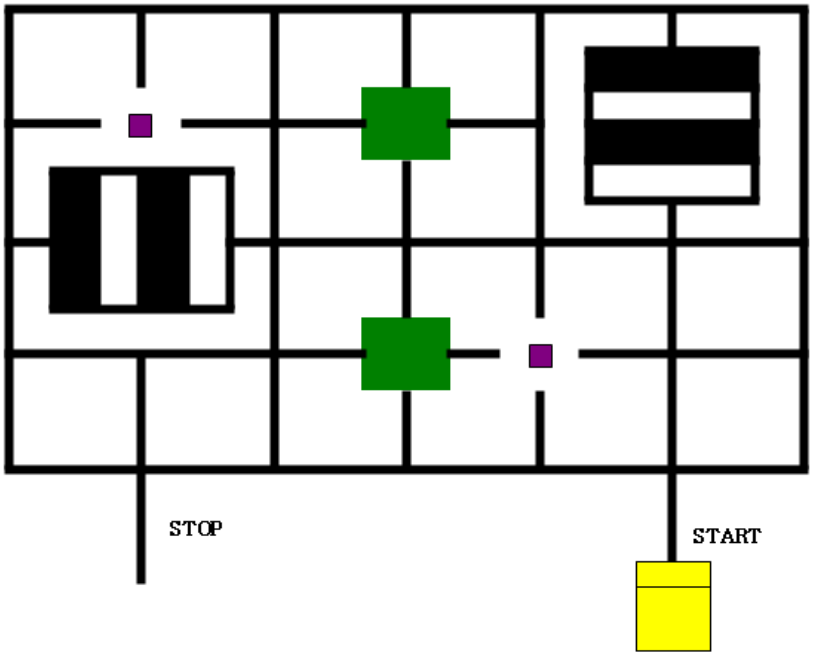
예제 프로그램

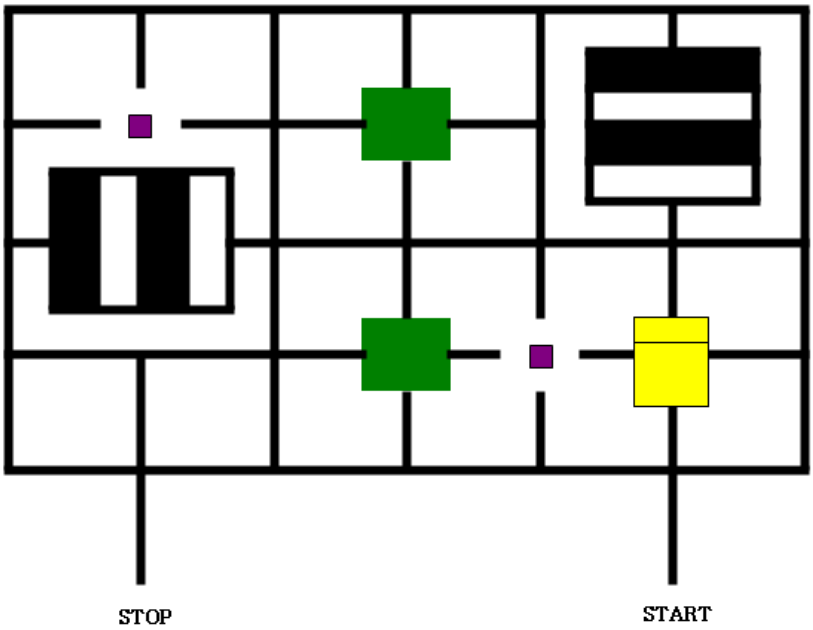
간단한 예제를 통해 전체 동작을 살펴 보세요.

예제 소스

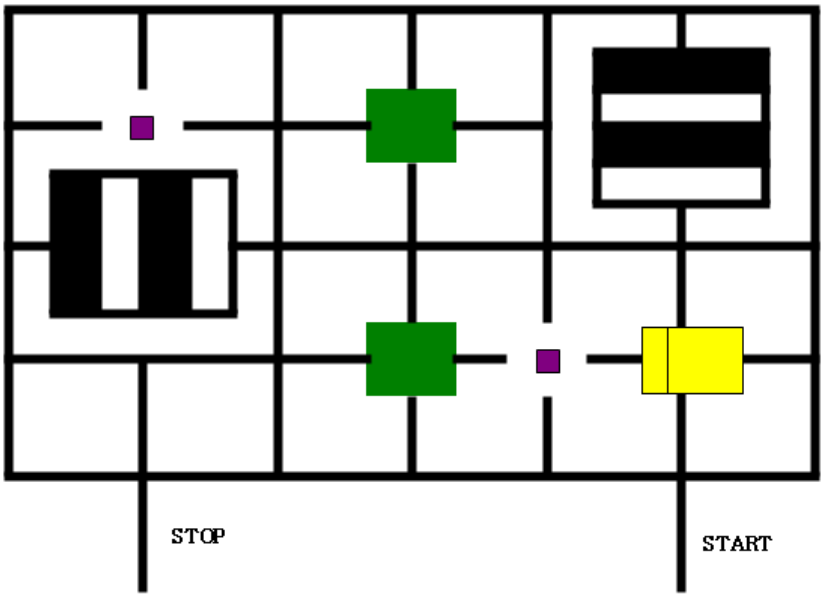
```
Int main(void)
{
        standby();
         delay(1000);
         line(pp, ff, 15, 0);                    // 1
         line(pp, ff, 15, -130);                 // 1
         exline(-8, 0, sensor2, 0);              // 2
         get(7, 25);                             // 3
         right5(8, 0);                           // 4
         line(pp, ff, 5, 0);                     // 4
         wbwb(3, 100, 100, 25);                  // 5
         line(pp, ff, 15, 0);                    // 6
         left3(8, 0);                            // 6
         line(pp, ff, 15, 0);                    // 6
         right5(8, 0);                           // 7
         line(tt, ff, 15, 0);                    // 7
         line(pp, ff, 15, 0);                    // 7
         left3(8, 0);                            // 8
         line(tt, ff, 20, 0);                    // 8
         line(tt, ff, 20, 0);                    // 8
         line(tt, ff, 20, 0);                    // 8
         line(tt, ff, 15, 0);                    // 8
         left3(8, 0);                            // 9
         line(tt, ff, 15, 0);                    // 9
        exline(-8,0,sensor2,0);                  // 10
         get(7, 25);                             // 11
         left3(8, 0);                            // 12
         line(tt, ff, 15, 0);                    // 13
         right5(8, 0);                           // 13
         line(tt, ff, 15, 0);                    // 13
         line(tt, ff, 15, 0);                    // 13
         line(tt, ff, 15, 0);                    // 13
         line(tt, ff, 15, 0);                    // 13
         right5(8, 0);                           // 14
         line(tt, ff, 5, 0);                     // 14
         line(pp, ff, 5, 0);                     // 14
         right5(8, 0);                           // 15
         line(pp, ff, 5, 0);                     // 15
         line(pp, ff, 5, 0);                     // 15
        wbwb(4, 100, 100, 25);                   // 16
         line(pp, ff, 12, 0);                    // 17
        left3(8, 0);                             // 17
         line(pp, ff, 20, 0);                    // 17
         line(pp, ff, 20, 0);                    // 17
         right5(8, 0);                           // 18
         line(pp, ff, 20, 0);                    // 18
        left3(8, 0);                             // 18
         lostline(0, 10, 500, 20);               // 18
}
```

동작

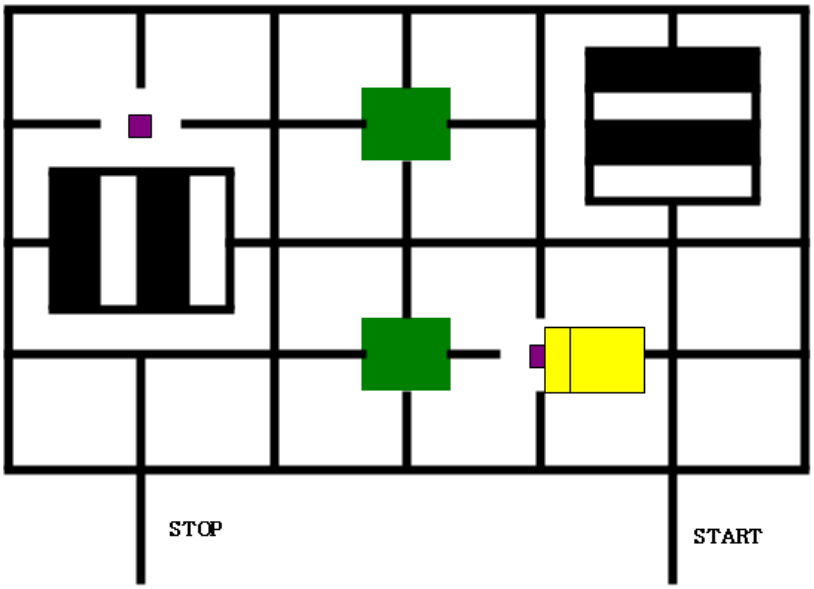




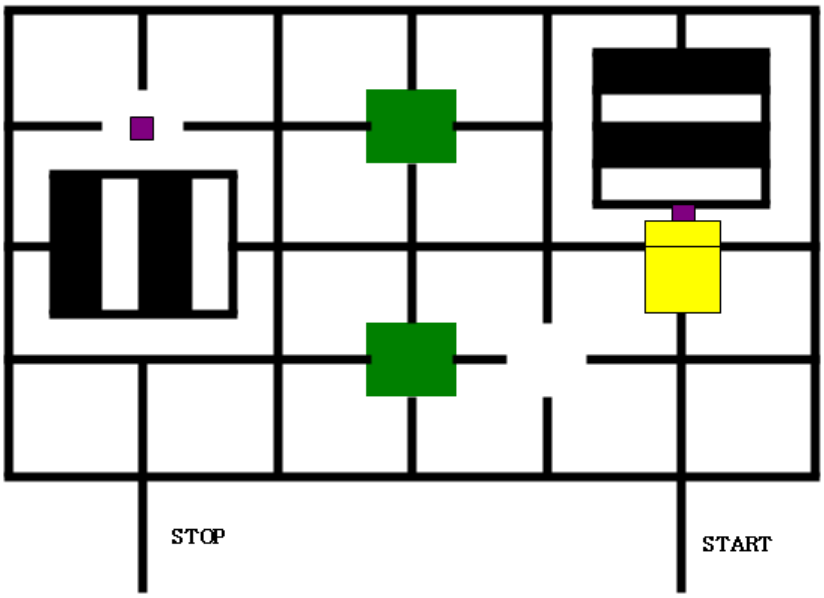

```
line(pp, ff, 15, 0);                    // 1
line(pp, ff, 15, -130);                 // 1
```



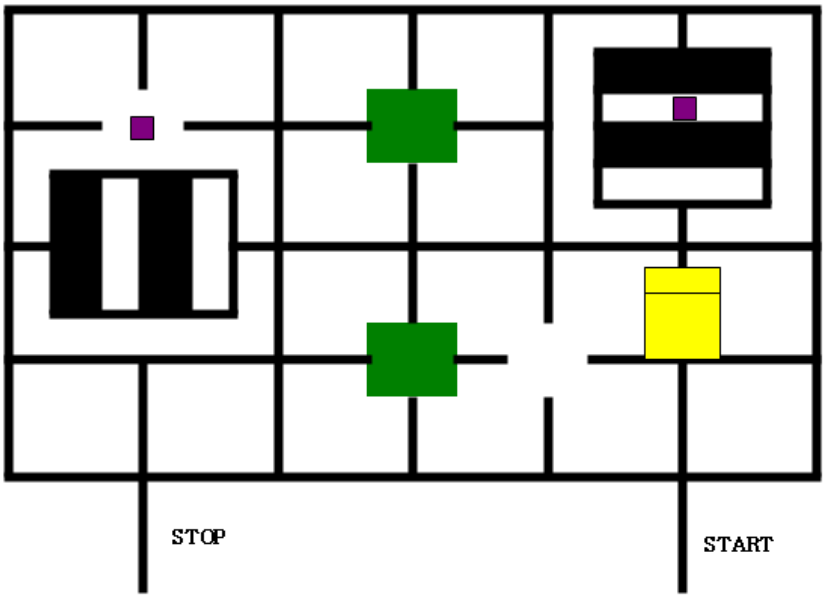

```
exline(-8, 0, sensor2, 0);                    // 2
```



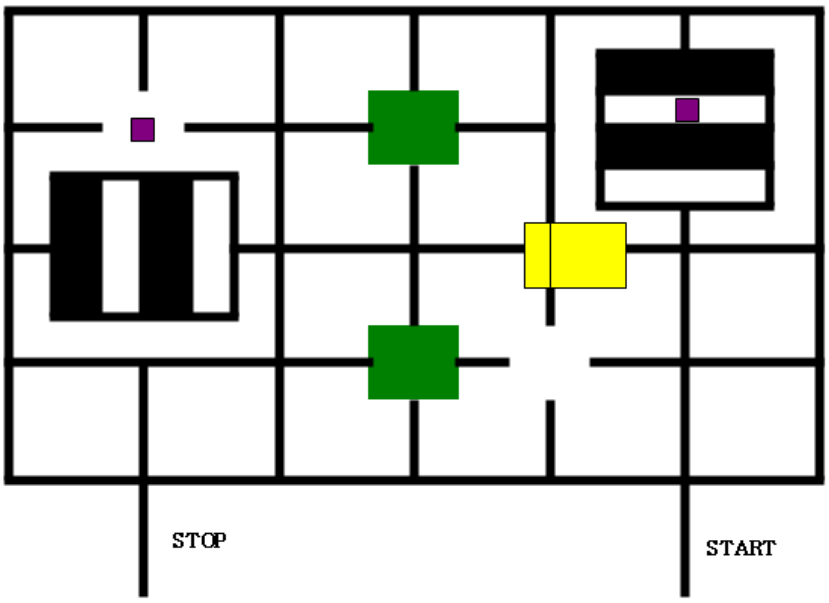

```
get(7, 25);                                   // 3
```



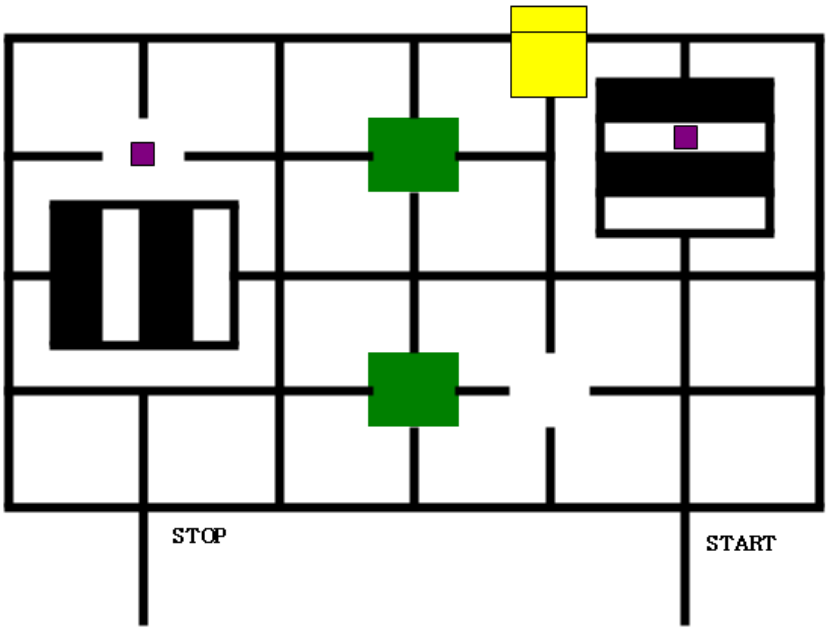

```
right5(8, 0);                    // 4
line(pp, ff, 5, 0);              // 4
```



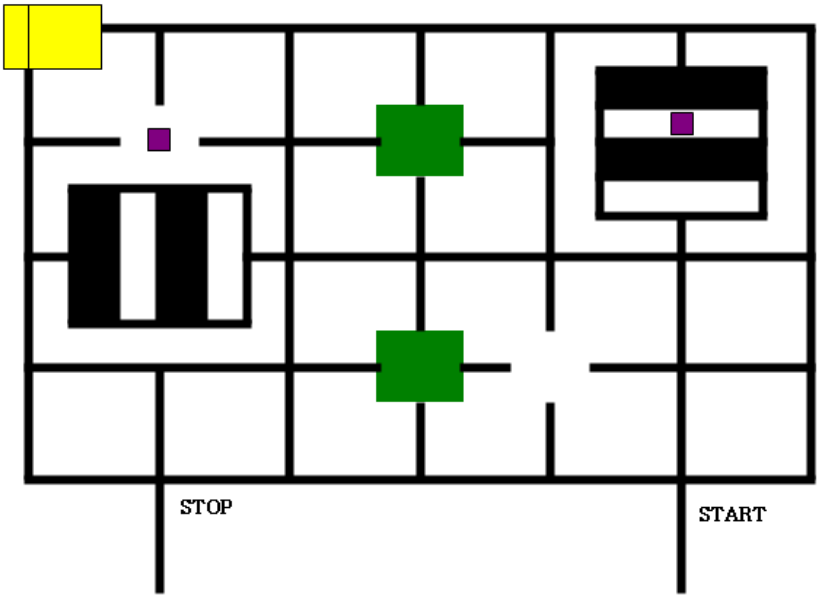

```
wbwb(3, 100, 100, 25);           // 5
```



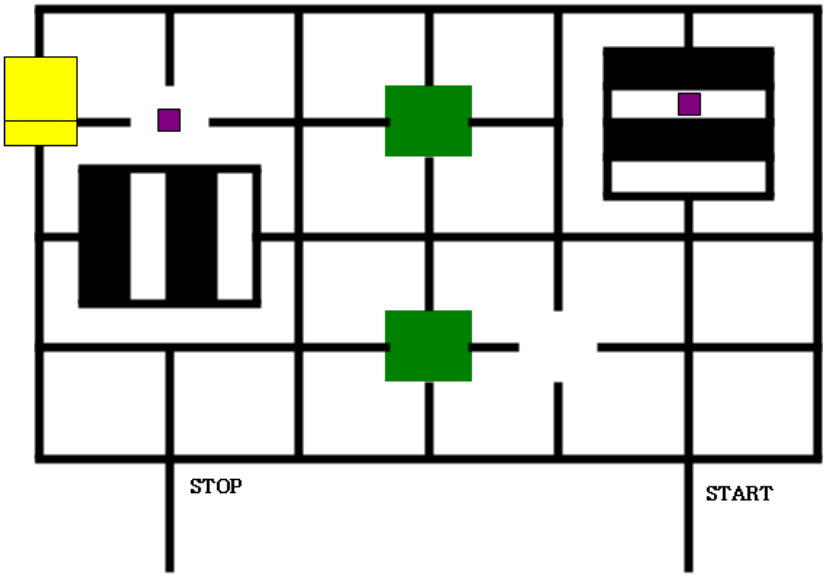

```
line(pp, ff, 15, 0);                    // 6
left3(8, 0);                            // 6
line(pp, ff, 15, 0);                    // 6
```



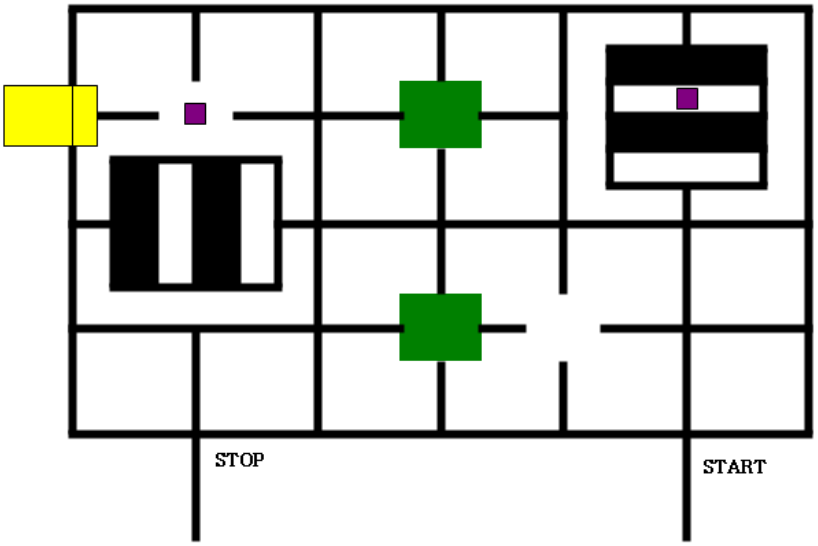

```
right5(8, 0);                           // 7
line(tt, ff, 15, 0);                    // 7
line(pp, ff, 15, 0);                    // 7
```



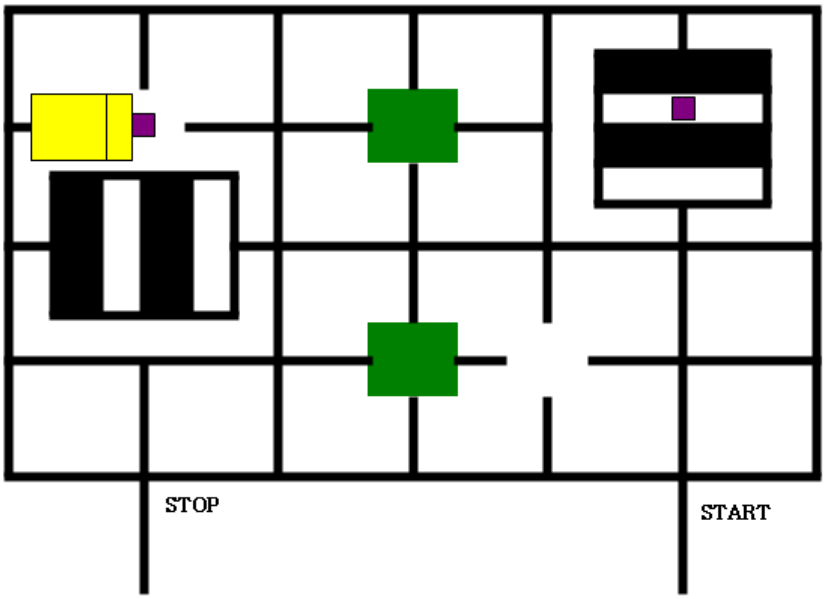

```
left3(8, 0);                    // 8
line(tt, ff, 20, 0);            // 8
line(tt, ff, 20, 0);            // 8
line(tt, ff, 20, 0);            // 8
line(tt, ff, 15, 0);            // 8
```



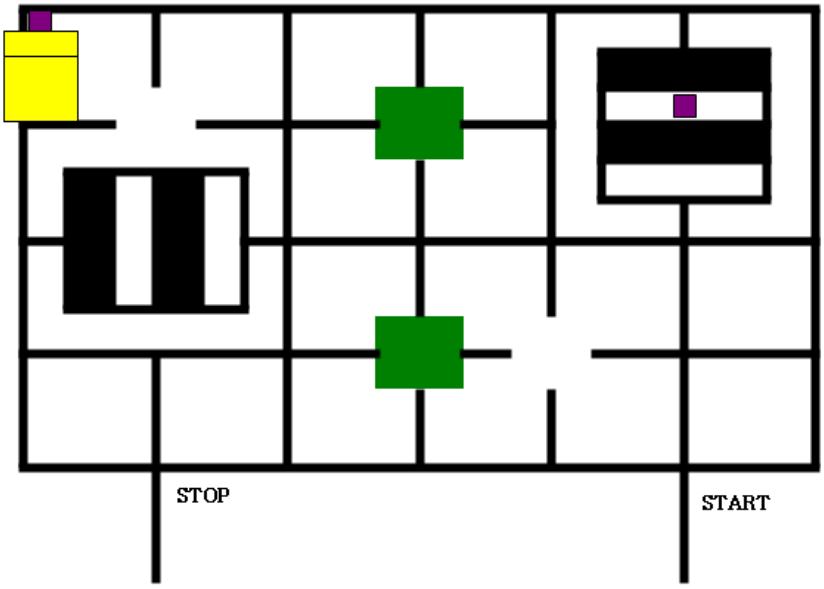

```
left3(8, 0);                    // 9
line(tt, ff, 15, 0);            // 9
```



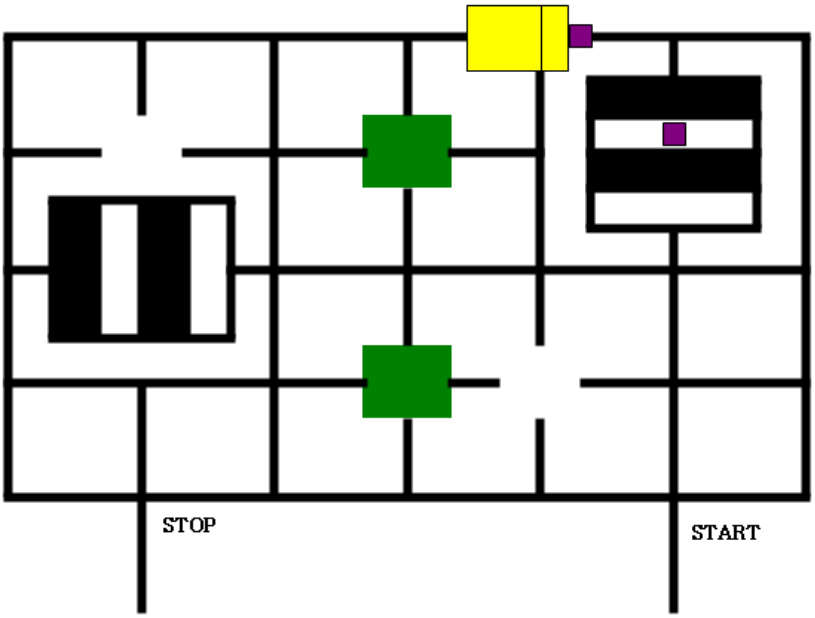

```
exline(-8, 0, sensor2, 0);                    // 10
```



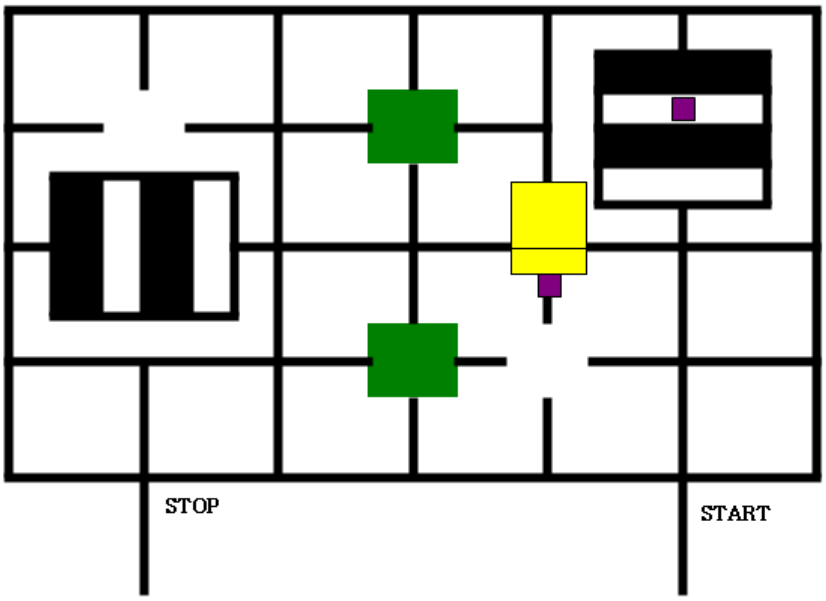

```
get(7, 25);                    // 11
```



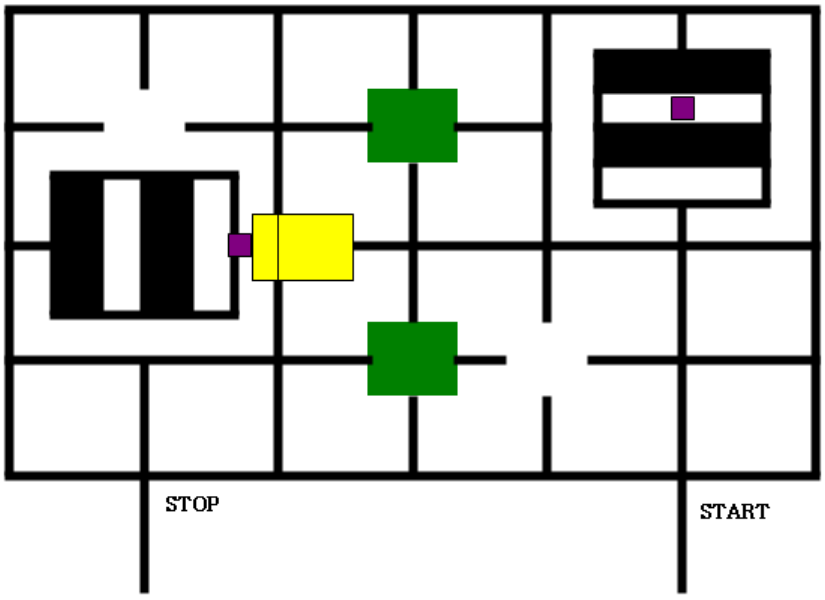

```
left3(8, 0);                    // 12
```



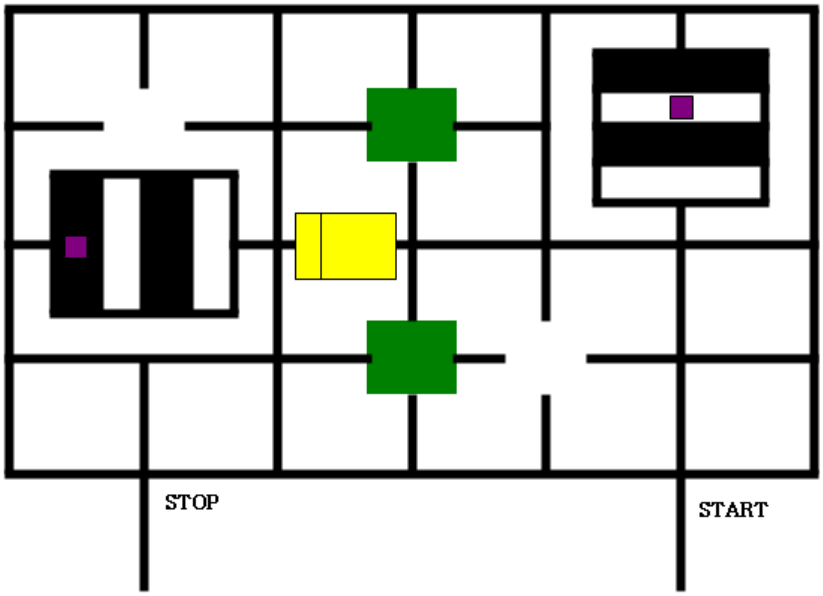

```
line(tt, ff, 15, 0);            // 13
right5(8, 0);                   // 13
line(tt, ff, 15, 0);            // 13
line(tt, ff, 15, 0);            // 13
line(tt, ff, 15, 0);            // 13
line(tt, ff, 15, 0);            // 13
```



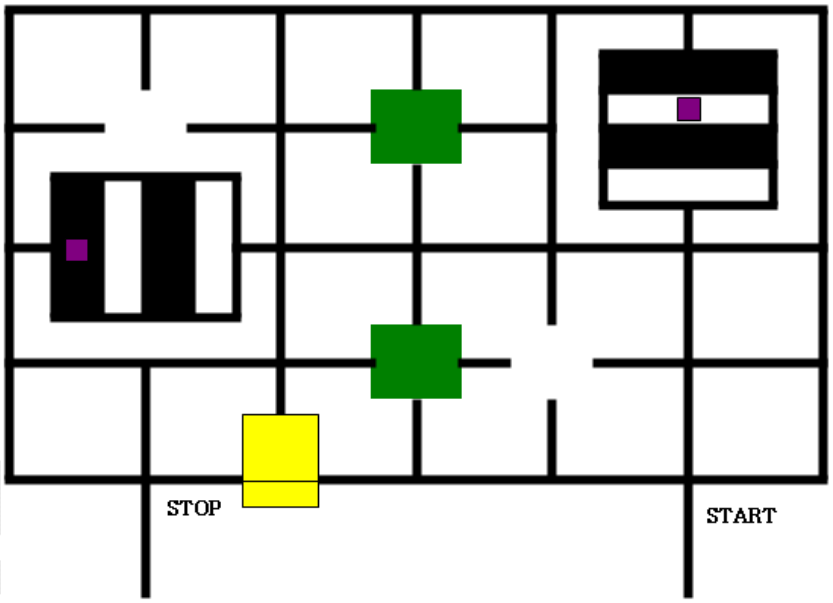

```
right5(8, 0);                    // 14
line(tt, ff, 5, 0);              // 14
line(pp, ff, 5, 0);              // 14
```



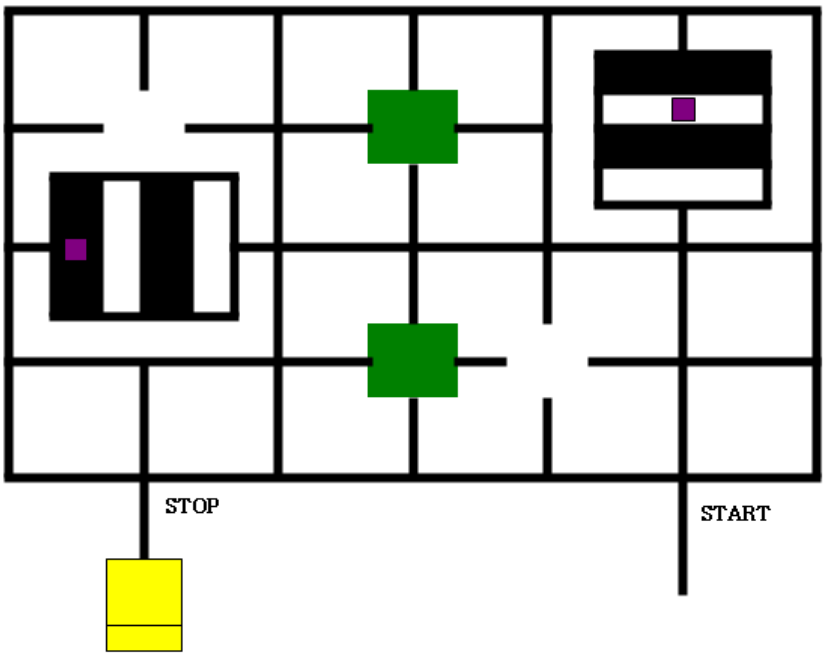

```
right5(8, 0);                    // 15
line(pp, ff, 5, 0);              // 15
line(pp, ff, 5, 0);              // 15
```




```
wbwb(4, 100, 100, 25);                // 16
```




```
line(pp, ff, 12, 0);                  // 17
left3(8, 0);                          // 17
line(pp, ff, 20, 0);                  // 17
line(pp, ff, 20, 0);                  // 17
```




```
right5(8, 0);                    // 18
line(pp, ff, 20, 0);             // 18
left3(8, 0);                     // 18
lostline(0, 10, 500, 20);        // 18
```