프로세싱 & 확장 키트

광운대학교 로봇학부
박광현

프로세싱

개요

- 2001년 MIT 미디어랩 Ben Fry와 Casey Reas
- 아티스트를 위한 편리한 그래픽 작성 도구
- 자바 기반
- 자바스크립트, 파이썬, 안드로이드, …
- 오픈 소스

개요

- 프로세싱:
  - 프로세싱 개발환경 (PDE)
  - 함수 모음
  - 문법
  - 커뮤니티
- 스케치: 작성된 프로그램
- 스케치북: 스케치 저장 폴더

정적 스케치 (Static Sketch)

```
line(10, 20, 80, 90); // x1, y1, x2, y2
```

```
size(600, 400); // width, height
background(255);
stroke(100);
line(10, 20, 80, 90);
```

- `background(gray);`
- `background(r, g, b);`
- `background(#FF7A00);`
- `background(0xFF7A00);`
- ...

- `stroke(gray);`
- `stroke(gray, alpha);`
- `stroke(r, g, b);`
- `stroke(r, g, b, a);`
- `stroke(#FF7A00);`
- `stroke(0xFFFF7A00);`
- ...

정적 스케치 (Static Sketch)

```
size(600, 400);
background(255, 122, 0);
rect(10, 20, 80, 90); // x, y, width, height
```

```
size(600, 400);
background(255, 122, 0);
noStroke();
rect(10, 20, 80, 90);
```

```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
fill(255, 0, 0);
rect(10, 20, 80, 90);
```

```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
strokeWeight(4); // pixel
fill(255, 0, 0);
rect(10, 20, 80, 90);
```

- `noFill();`

정적 스케치 (Static Sketch)

```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
strokeWeight(4);
fill(255, 0, 0);
rect(10, 20, 80, 90, 10); // x, y, width, height, corner
```

```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
strokeWeight(4);
fill(255, 0, 0);
rect(10, 20, 80, 90, 10, 20, 30, 40);
                                   // x, y, w, h, tl, tr, br, bl
```

정적 스케치 (Static Sketch)

```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
strokeWeight(4);
fill(255, 0, 0);
ellipse(100, 200, 80, 90); // x, y, width, height
```

```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
strokeWeight(4);
fill(255, 0, 0);
triangle(100, 20, 10, 100, 200, 100);// x1, y1, x2, y2, x3, y3
```

```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
strokeWeight(4);
fill(255, 0, 0);
point(100, 200);// x, y
```

정적 스케치 (Static Sketch)

```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
strokeWeight(4);
fill(255, 0, 0);
quad(10, 20, 80, 90, 100, 200, 10, 100);
                           // x1, y1, x2, y2, x3, y3, x4, y4
```

```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
strokeWeight(4);
fill(255, 0, 0);
arc(100, 200, 80, 90, 0, HALF_PI);
                         // x, y, width, height, start, stop
```

HALF_PI, PI, QUARTER_PI, TWO_PI

라디안(radian)

기본 형태

```
void setup() {

}

void draw() {

}
```

밝기 애니메이션

```
void setup() {
  size(600, 400);
  stroke(0, 0, 255);
}

void draw() {
  background(255, 122, 0);
  text("frame: " + frameCount, 20, 20);
  fill(frameCount % 256);
  rect(50, 50, 200, 200);
}
```

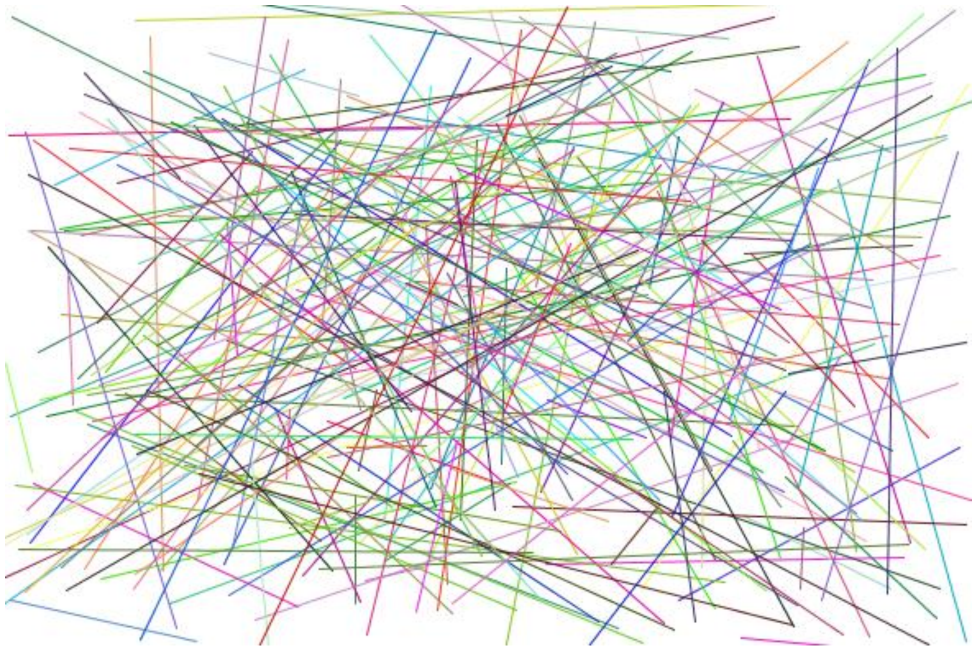
0부터 시작
draw() 호출 때마다 1씩 증가

밝기 애니메이션

```
void setup() {
  size(600, 400);
  stroke(0, 0, 255);
}

void draw() {
  background(255, 122, 0);
  text("frame: " + frameCount, 20, 20);
  pushStyle();
  fill(frameCount % 256);
  rect(50, 50, 200, 200);
  popStyle();
}
```

직선 예술

```
void setup() {
  size(600, 400);
  background(255);
}

void draw() {
  stroke(random(256), random(256), random(256));
  line(random(width), random(height), random(width), random(height));
}
```



- `random(end);`
- `random(start, end);`

end는 포함 안 됨

마우스

```
void setup() {
  size(600, 400);
  background(255, 122, 0);
  stroke(0, 0, 255);
}

void draw() {
  line(200, 200, mouseX, mouseY);
}
```

```
void setup() {
  size(600, 400);
  stroke(0, 0, 255);
}

void draw() {
  background(255, 122, 0);
  line(200, 200, mouseX, mouseY);
}
```

마우스

```
void setup() {
  size(600, 400);
  stroke(0, 0, 255);
}

void draw() {
  line(200, 200, mouseX, mouseY);
}

void mousePressed() {
  background(255, 122, 0);
}
```

마우스

```
void setup() {
  size(600, 400);
  stroke(0, 0, 255);
}

void draw() {
  line(200, 200, mouseX, mouseY);
}

void mousePressed() {
  if(mouseButton == LEFT)
    background(255, 122, 0);
  else
    background(0, 128, 0);
}
```

마우스

- `mouseButton`
- `mouseClicked()`
- `mouseDragged()`
- `mouseMoved()`
- `mousePressed()`
- `mouseReleased()`
- `mouseWheel()`
- `mouseX`
- `mouseY`
- `pmouseX`
- `pmouseY`

마우스

```
void setup() {
  size(600, 400);
  background(255, 122, 0);
  stroke(0, 0, 255);
}

void draw() {
}

void mouseDragged() {
  line(pmouseX, pmouseY, mouseX, mouseY);
}
```

마우스

```
void setup() {
  size(600, 400);
  background(255, 122, 0);
  stroke(0, 0, 255);
}

void draw() {
  background(255);
  fill(0, 255, 0);
  rect(mouseX, mouseY, 30, 30);
}
```

```
void setup() {
  size(600, 400);
  background(255, 122, 0);
  stroke(0, 0, 255);
  frameRate(5);
}

void draw() {
  background(255);
  fill(0, 255, 0);
  rect(mouseX, mouseY, 30, 30);
}
```

키보드

```
void setup() {
  size(600, 400);
  stroke(0, 0, 255);
}

void draw() {
  line(200, 200, mouseX, mouseY);
}

void keyPressed() {
  if(key == 'a')
    background(255, 122, 0);
  else
    background(0, 128, 0);
}
```

키보드

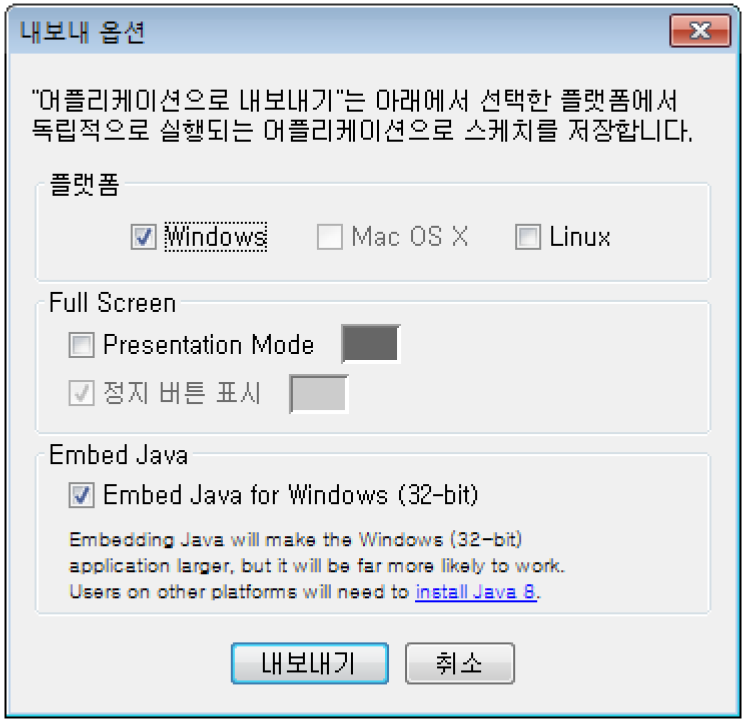
- `key`
- `keyCode`
- `keyPressed()`
- `keyPressed`
- `keyReleased()`
- `keyTyped()`

화면 저장

```
void setup() {
  size(600, 400);
  stroke(0, 0, 255);
}

void draw() {
  line(200, 200, mouseX, mouseY);
}

void keyPressed() {
  if(key == ‘a’)
    saveFrame("a.png");
  else
    background(0, 128, 0);
}
```

응용 프로그램 만들기

- 파일 > 어플리케이션으로 내보내기



내보내 옵션

"어플리케이션으로 내보내기"는 아래에서 선택한 플랫폼에서 독립적으로 실행되는 어플리케이션으로 스케치를 저장합니다.

플랫폼

[✓] Windows [ ] Mac OS X [ ] Linux

Full Screen

[ ] Presentation Mode

[✓] 정지 버튼 표시

Embed Java

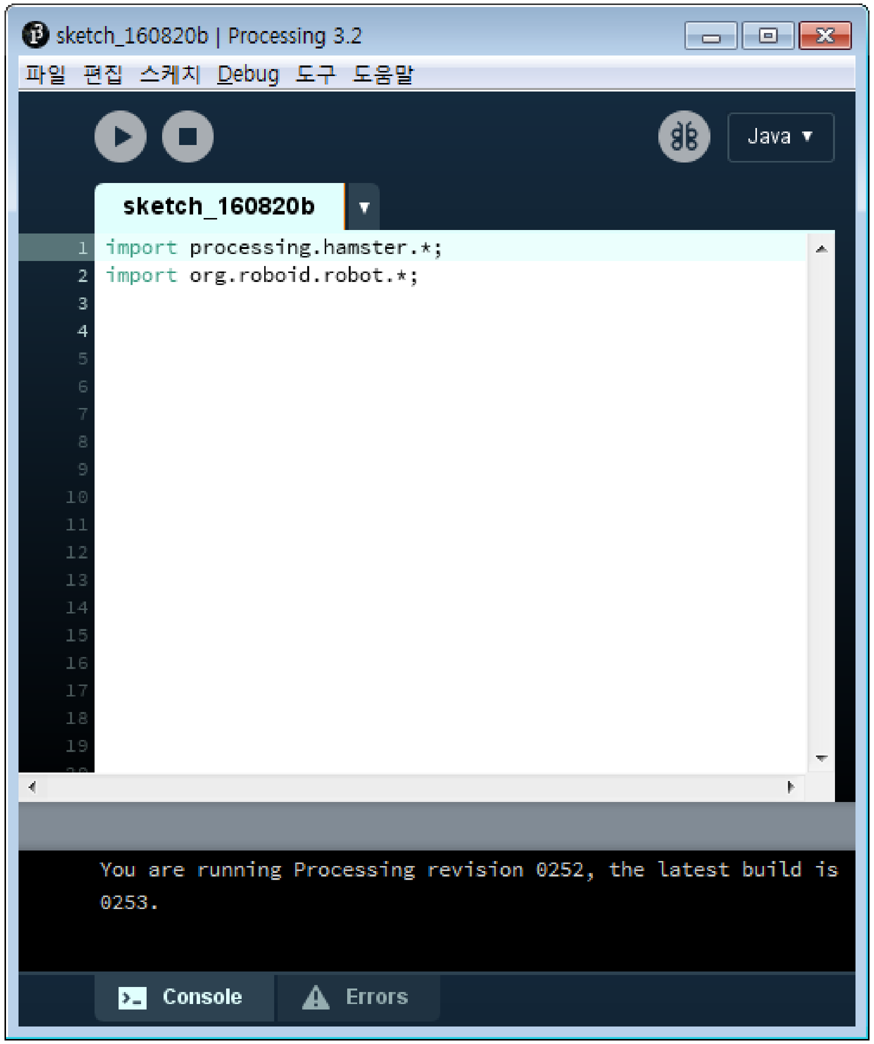
[✓] Embed Java for Windows (32-bit)

Embedding Java will make the Windows (32-bit) application larger, but it will be far more likely to work. Users on other platforms will need to install Java 8.

햄스터

라이브러리 사용

- 스케치 > 내부 라이브러리... > hamster


sketch_160820b | Processing 3.2
파일 편집 스케치 Debug 도구 도움말
Java
sketch_160820b
Console
Errors

```
import processing.hamster.*;
import org.roboid.robot.*;
```

```
You are running Processing revision 0252, the latest build is
0253.
```

기본 동작

```
import org.roboid.robot.*;
import processing.hamster.*;

Hamster hamster;

void setup() {
  hamster = new Hamster(this);
}

// don't forget 'draw'
void draw() {
}
```

createHamster()로 변경 예정

run()으로 변경 예정

```
void control() {
  // move forward
  hamster.write(Hamster.LEFT_WHEEL, 50);
  hamster.write(Hamster.RIGHT_WHEEL, 50);
  delay(500); // ms

  // move backward
  hamster.write(Hamster.LEFT_WHEEL, -50);
  hamster.write(Hamster.RIGHT_WHEEL, -50);
  delay(500);

  // stop
  hamster.write(Hamster.LEFT_WHEEL, 0);
  hamster.write(Hamster.RIGHT_WHEEL, 0);

  // disconnect
  hamster.dispose();
}
```

LED 켜기

```
import org.roboid.robot.*;
import processing.hamster.*;

Hamster hamster;

void setup() {
  hamster = new Hamster(this);
}

// dont' forget 'draw'
void draw() {
}

void control() {
  hamster.write(Hamster.LEFT_LED, Hamster.LED_RED);
  hamster.write(Hamster.RIGHT_LED, Hamster.LED_GREEN);
  delay(500);

  hamster.write(Hamster.LEFT_LED, Hamster.LED_OFF);
  hamster.write(Hamster.RIGHT_LED, Hamster.LED_OFF);

  // disconnect
  hamster.dispose();
}
```

반복

```
import org.roboid.robot.*;
import processing.hamster.*;

Hamster hamster;

void setup() {
  hamster = new Hamster(this);
}

void draw() {
}

void control() {
  for(int i = 0; i < 10; ++i) {
    hamster.write(Hamster.LEFT_WHEEL, 50);
    hamster.write(Hamster.RIGHT_WHEEL, 50);
    delay(500);
    hamster.write(Hamster.LEFT_WHEEL, -50);
    hamster.write(Hamster.RIGHT_WHEEL, -50);
    delay(500);
  }
  hamster.dispose();
}
```

반복

```
import org.roboid.robot.*;
import processing.hamster.*;

Hamster hamster;

void setup() {
  hamster = new Hamster(this);
}

void draw() {
}

void repeat() {
  hamster.write(Hamster.LEFT_WHEEL, 50);
  hamster.write(Hamster.RIGHT_WHEEL, 50);
  delay(500);
  hamster.write(Hamster.LEFT_WHEEL, -50);
  hamster.write(Hamster.RIGHT_WHEEL, -50);
  delay(500);
}
```

근접 센서

```
import org.roboid.robot.*;
import processing.hamster.*;

Hamster hamster;

void setup() {
  hamster = new Hamster(this);
}

void draw() {
}

void repeat() {
  int proximity = hamster.read(Hamster.LEFT_PROXIMITY);
  if(proximity < 50) {
    hamster.write(Hamster.LEFT_WHEEL, 50);
    hamster.write(Hamster.RIGHT_WHEEL, 50);
  } else {
    hamster.write(Hamster.LEFT_WHEEL, -50);
    hamster.write(Hamster.RIGHT_WHEEL, -50);
  }
}
```

라인 트레이서

```
import org.roboid.robot.*;
import processing.hamster.*;

Hamster hamster;

void setup() {
  hamster = new Hamster(this);
}

void draw() {
}

void repeat() {
  int leftFloor = hamster.read(Hamster.LEFT_FLOOR);
  int rightFloor = hamster.read(Hamster.RIGHT_FLOOR);
  int diff = leftFloor - rightFloor;
  hamster.write(Hamster.LEFT_WHEEL, int(30 + diff * 0.4));
  hamster.write(Hamster.RIGHT_WHEEL, int(30 - diff * 0.4));
}
```

멀티 로봇

```
import org.roboid.robot.*;
import processing.hamster.*;

Hamster hamster1;
Hamster hamster2;

void setup() {
  hamster1 = new Hamster(this);
  hamster2 = new Hamster(this);
}

void draw() {
}

void repeat() {
  hamster1.write(Hamster.LEFT_WHEEL, 50);
  hamster1.write(Hamster.RIGHT_WHEEL, 50);
  hamster2.write(Hamster.LEFT_WHEEL, -50);
  hamster2.write(Hamster.RIGHT_WHEEL, 50);
  delay(500);
  hamster1.write(Hamster.LEFT_WHEEL, -50);
  hamster1.write(Hamster.RIGHT_WHEEL, -50);
  hamster2.write(Hamster.LEFT_WHEEL, 50);
  hamster2.write(Hamster.RIGHT_WHEEL, -50);
  delay(500);
}
```

햄스터 + 그래픽

손 따라가기 1/2

```
import org.roboid.robot.*;
import processing.hamster.*;

Hamster hamster;
int leftProximity;
int rightProximity;

void setup() {
  size(200,200);
  noStroke();
  hamster = new Hamster(this);
}

void draw() {
  background(255);
  fill(0);
  text("Left: " + leftProximity, 28, 185);
  text("Right: " + rightProximity, 125, 185);

  // draw bar graph
  rect(30, 20, 30, 150);
  rect(130, 20, 30, 150);
  fill(255);
  rect(30, 20, 30, leftProximity * 2);
  rect(130, 20, 30, rightProximity * 2);
}
```

손 따라가기 2/2

```
void repeat() {
  leftProximity = hamster.read(Hamster.LEFT_PROXIMITY);
  rightProximity = hamster.read(Hamster.RIGHT_PROXIMITY);
  // left wheel
  if(leftProximity > 15) {
    hamster.write(Hamster.LEFT_WHEEL, (40 - leftProximity) * 4);
  } else {
    hamster.write(Hamster.LEFT_WHEEL, 0);
  }

  // right wheel
  if(rightProximity > 15) {
    hamster.write(Hamster.RIGHT_WHEEL, (40 - rightProximity) * 4);
  } else {
    hamster.write(Hamster.RIGHT_WHEEL, 0);
  }
}
```

마우스 조종기 1/2

```
import org.roboid.robot.*;
import processing.hamster.*;

Hamster hamster;
int centerX, centerY;

void setup() {
  size(200,200);
  centerX = 100;
  centerY = 100;
  hamster = new Hamster(this);
}

void draw() {
  background(255);
  fill(0);
  text("Press a button to move..", 10, 16);
  ellipse(100,100, 30, 30);
  line(100 ,100, mouseX, mouseY);
}
```

마우스 조종기 2/2

```
void repeat() {
  int dx = centerX - mouseX;
  int dy = centerY - mouseY;

  hamster.write(Hamster.LEFT_WHEEL, 0);
  hamster.write(Hamster.RIGHT_WHEEL, 0);

  if(!mousePressed) return;

  if(abs(dx) > 15 || abs(dy) > 15) {
    if(dy < 0) {
      hamster.write(Hamster.LEFT_WHEEL, dy / 2 + dx / 2);
      hamster.write(Hamster.RIGHT_WHEEL, dy / 2 - dx / 2);
    } else {
      hamster.write(Hamster.LEFT_WHEEL, dy / 2 - dx / 2);
      hamster.write(Hamster.RIGHT_WHEEL, dy / 2 + dx / 2);
    }
  }
}
```

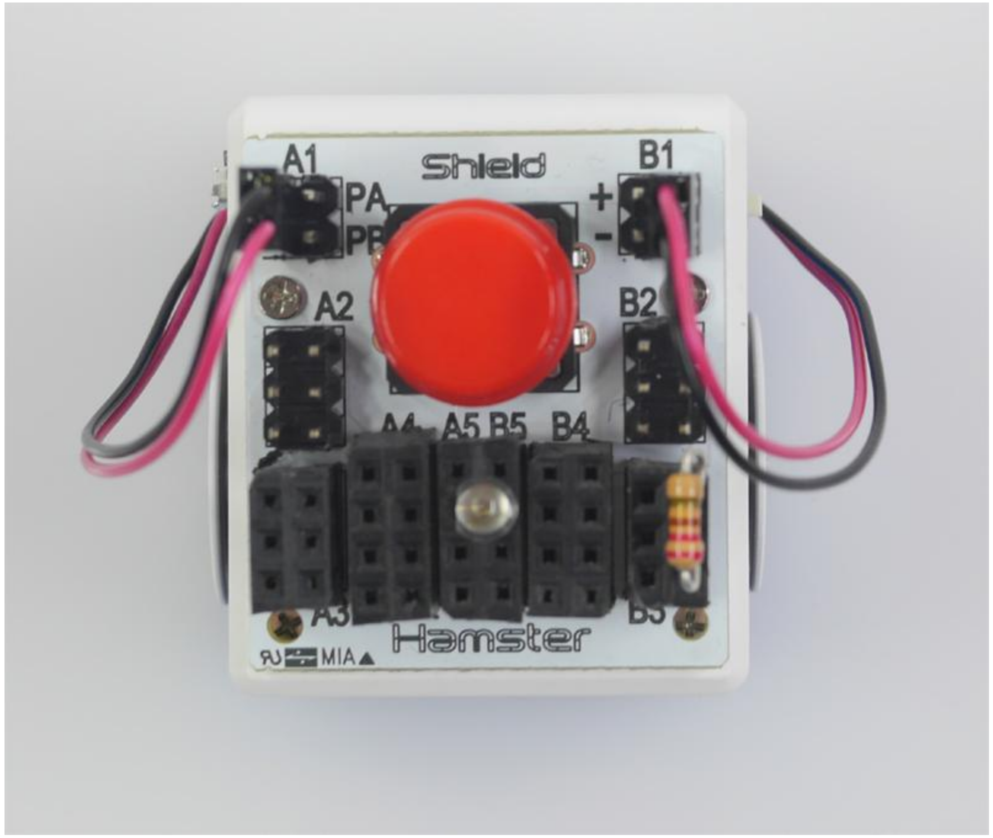
키보드 조종기

```
import processing.hamster.*;
import org.roboid.robot.*;

Hamster hamster;

void setup() {
  hamster = new Hamster(this);
}

void draw() {
}
```

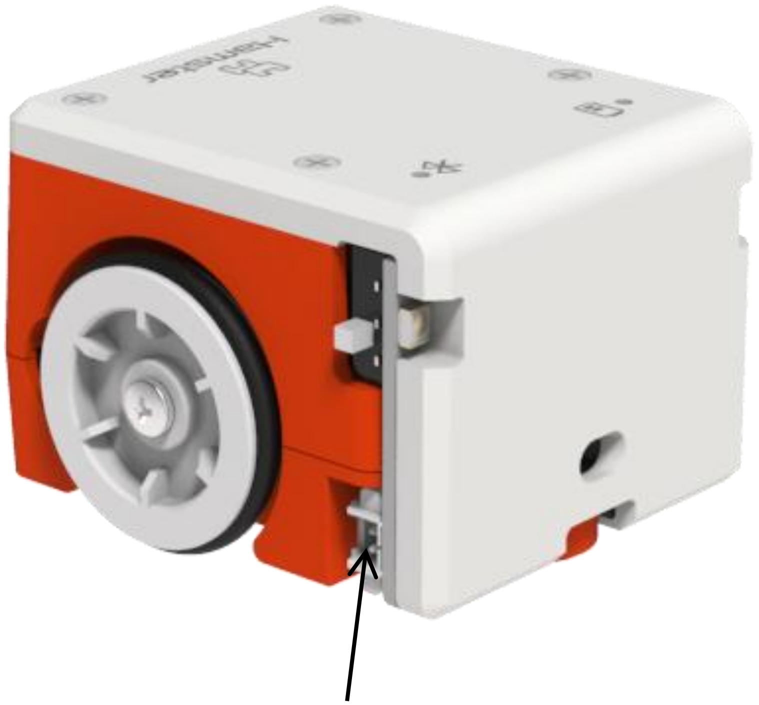
```
void keyPressed() {
  if(key == CODED) {
    switch(keyCode) {
      case UP:
        hamster.write(Hamster.LEFT_WHEEL, 30);
        hamster.write(Hamster.RIGHT_WHEEL, 30);
        break;
      case DOWN:
        hamster.write(Hamster.LEFT_WHEEL, -30);
        hamster.write(Hamster.RIGHT_WHEEL, -30);
        break;
      case LEFT:
        hamster.write(Hamster.LEFT_WHEEL, -30);
        hamster.write(Hamster.RIGHT_WHEEL, 30);
        break;
      case RIGHT:
        hamster.write(Hamster.LEFT_WHEEL, 30);
        hamster.write(Hamster.RIGHT_WHEEL, -30);
        break;
    }
  } else if(key == ' ') {
    hamster.write(Hamster.LEFT_WHEEL, 0);
    hamster.write(Hamster.RIGHT_WHEEL, 0);
  }
}
```

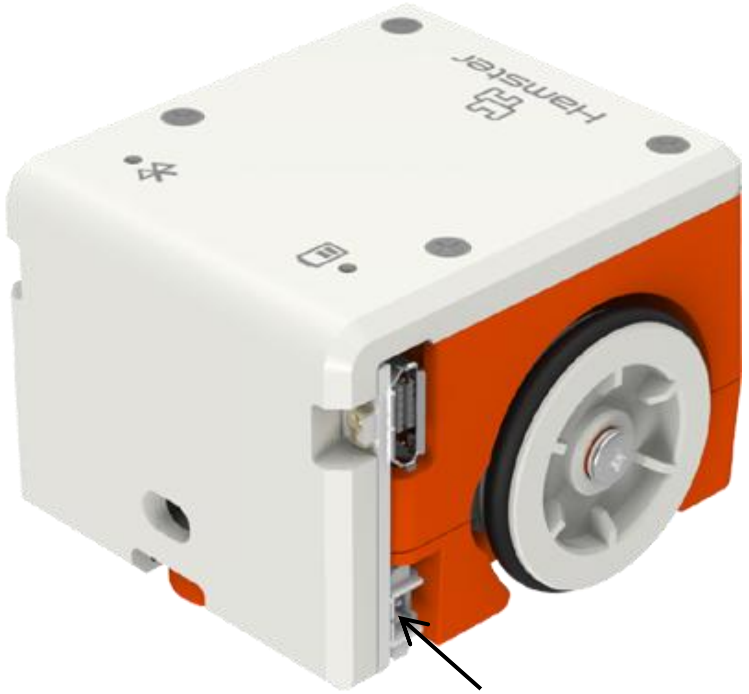
확장 보드

확장 보드

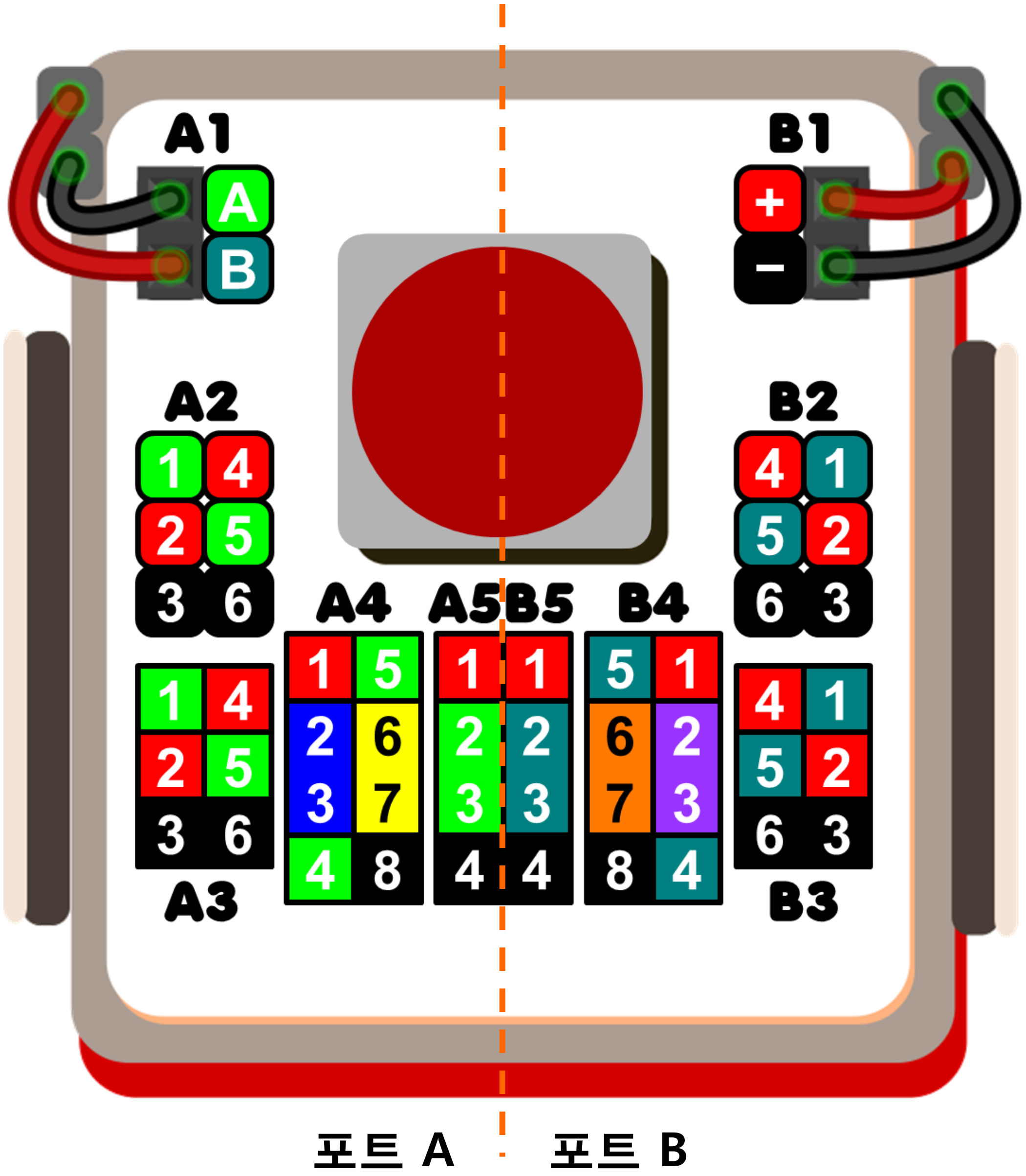
B1
+
-
Shield
PA
A1

A1
Shield
B1
PA
PB
+
-
A2
B2
A4
A5
B5
B4
A3
B3
Hamster
MIA

외부 확장 포트


보조 전원 단자
3.7V 리튬 폴리머 전지

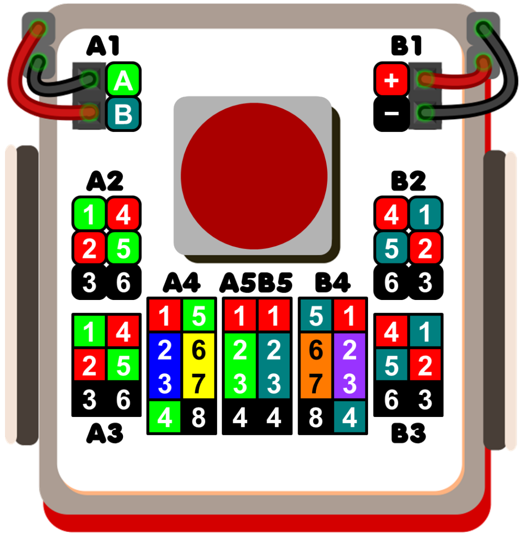

외부 입출력 단자 (포트A, 포트B)
디지털 입력, ADC 입력
디지털 출력, 아날로그(PWM) 출력
아날로그 서보 제어 출력

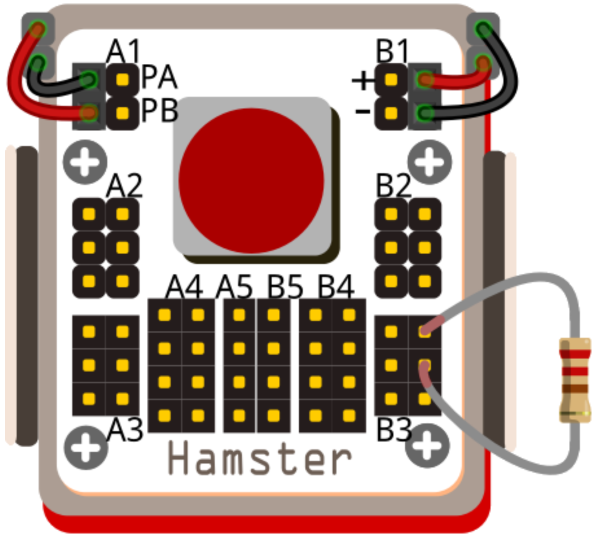
핀/소켓 배치 살펴보기



VCC	GND	포트A	포트B
🟥	⬛	🟩	🟦

디지털 입력: 버튼을 누르면 삐 소리가 나요

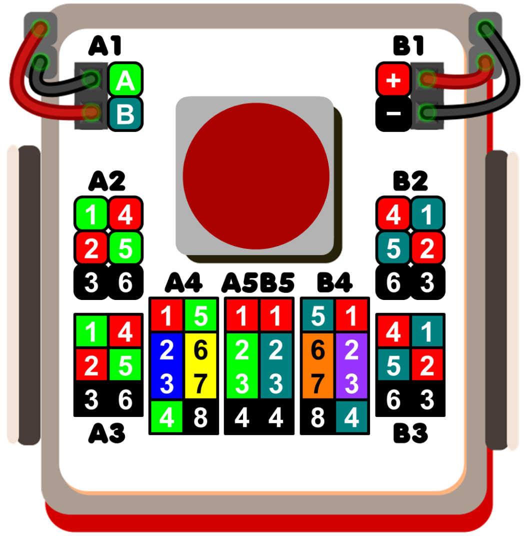


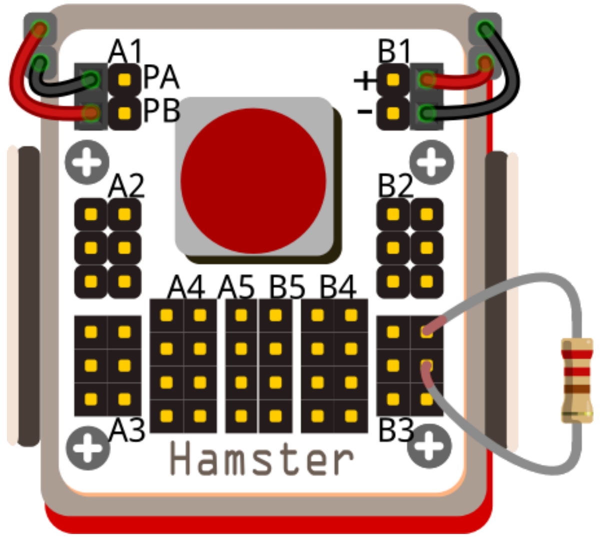


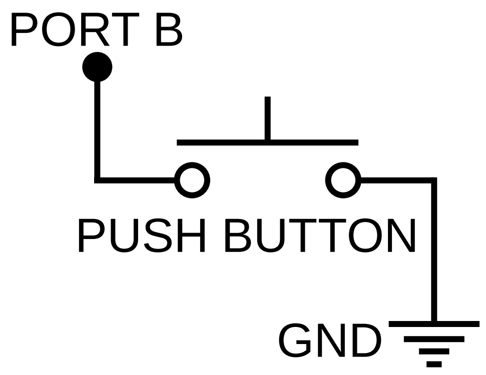
저항 값 읽는 방법

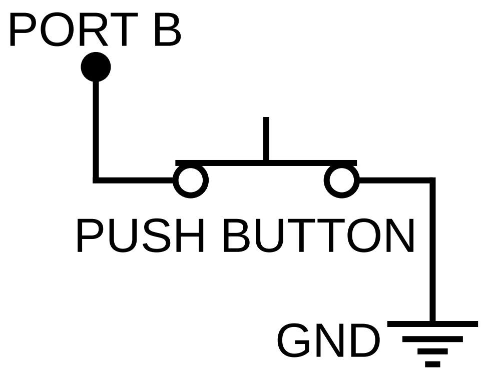
색	첫 번째 띠	두 번째 띠	세 번째 띠	네 번째 띠(오차)
검은색	0	0	$\times 10^0$	
갈색	1	1	$\times 10^1$	±1%
빨간색	2	2	$\times 10^2$	±2%
주황색	3	3	$\times 10^3$	
노란색	4	4	$\times 10^4$	
초록색	5	5	$\times 10^5$	±0.5%
파란색	6	6	$\times 10^6$	±0.25%
보라색	7	7	$\times 10^7$	±0.1%
회색	8	8	$\times 10^8$	±0.05%
흰색	9	9	$\times 10^9$	
금색			$\times 0.1$	±5%
은색			$\times 0.01$	±10%
없음				±20%

디지털 입력: 버튼을 누르면 삐 소리가 나요



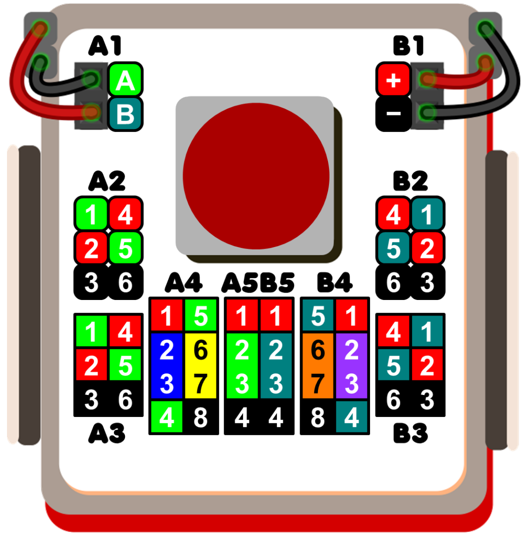


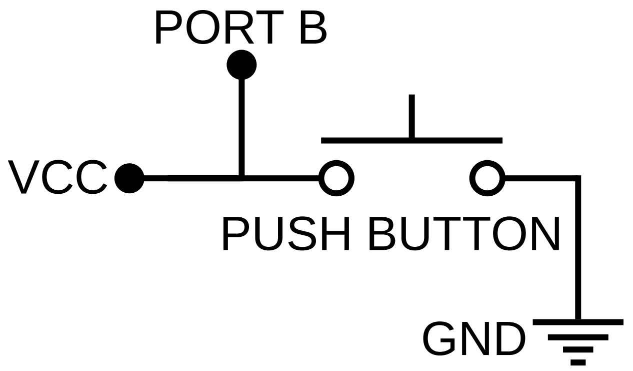


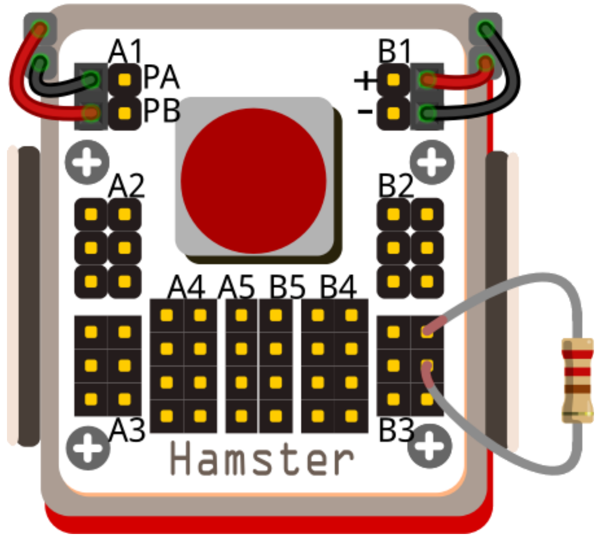


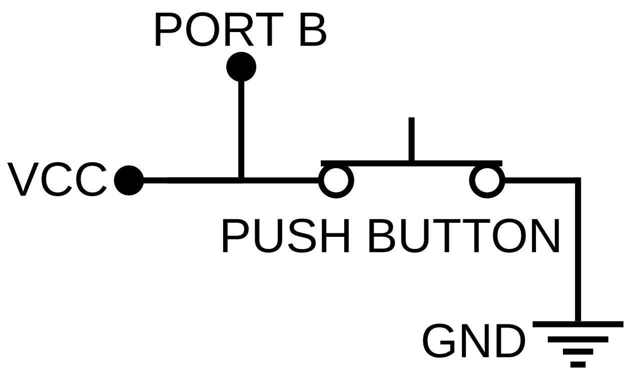
플로팅 상태

디지털 입력: 버튼을 누르면 삐 소리가 나요

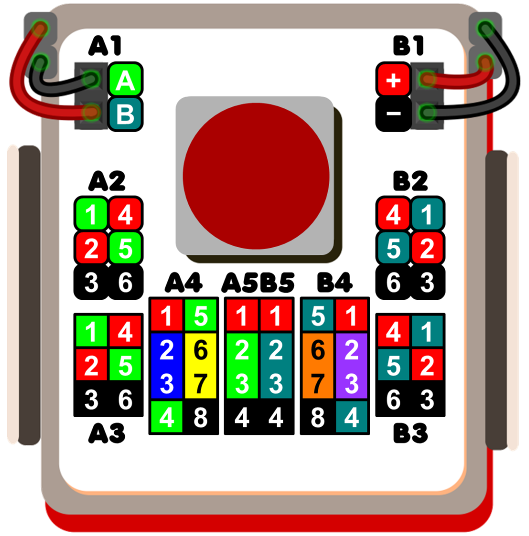


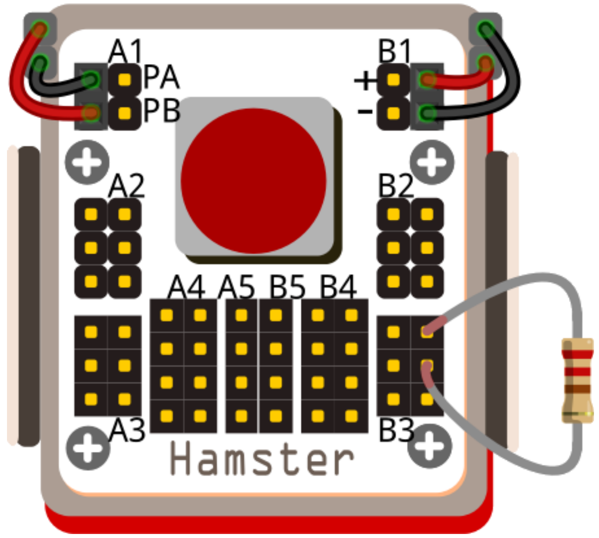




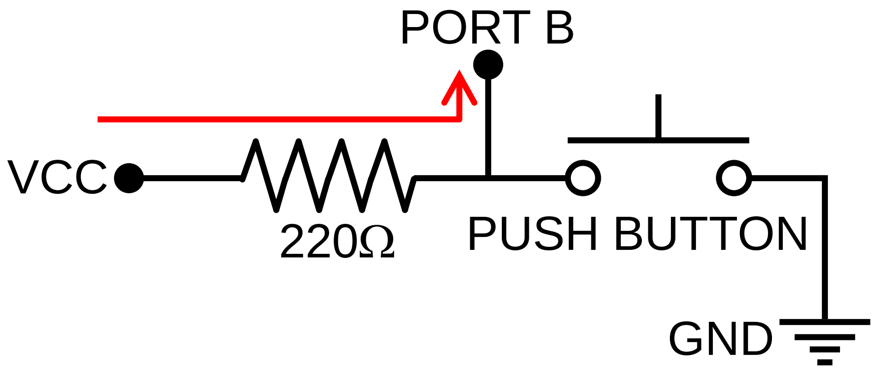


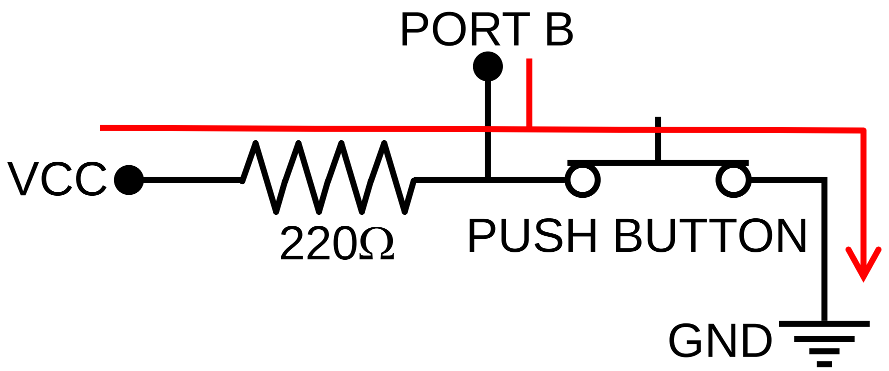
디지털 입력: 버튼을 누르면 삐 소리가 나요





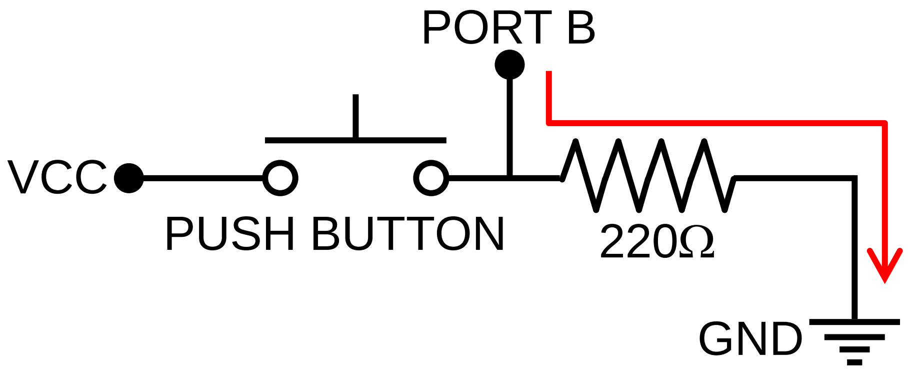
풀업 저항

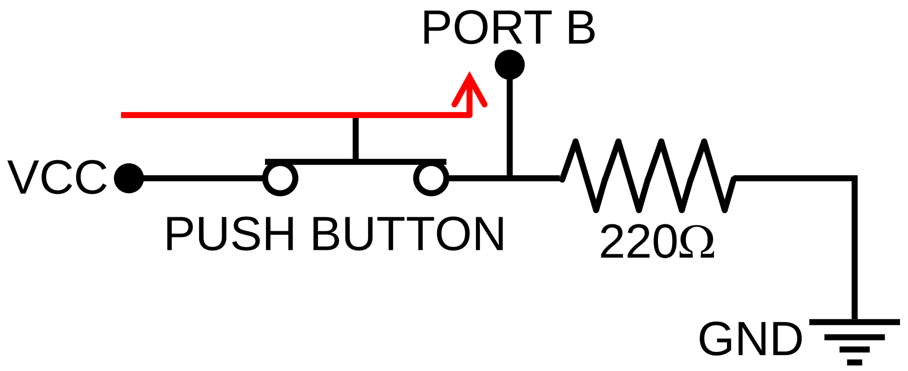




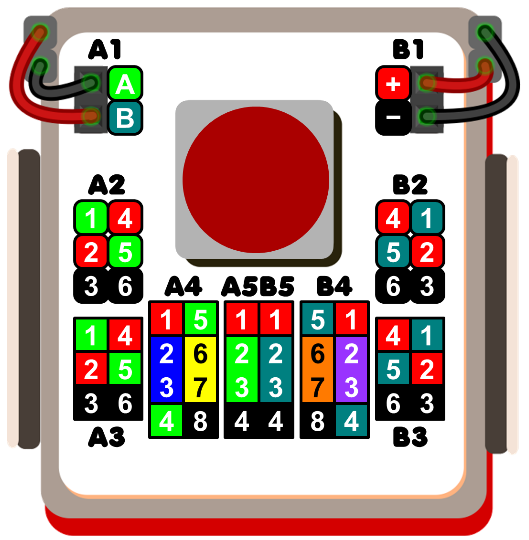
디지털 입력: 버튼을 누르면 삐 소리가 나요

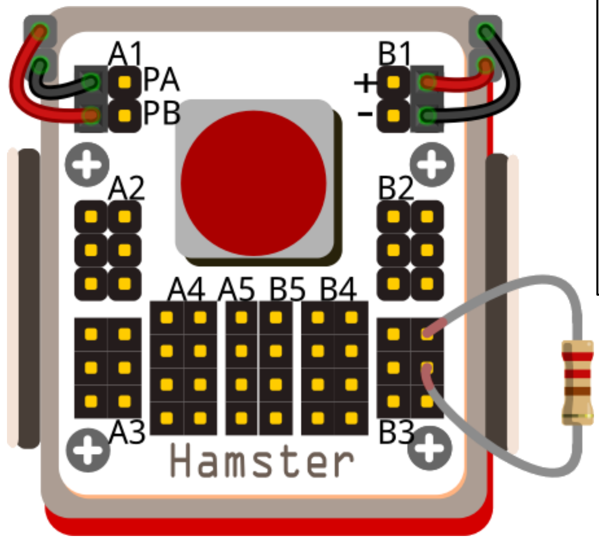
풀다운 저항





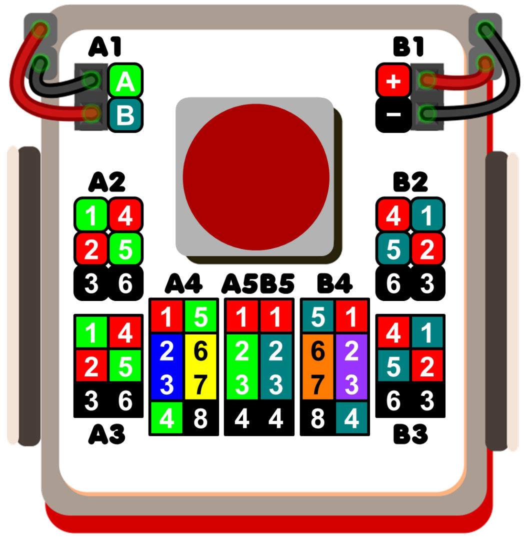
디지털 입력: 버튼을 누르면 삐 소리가 나요



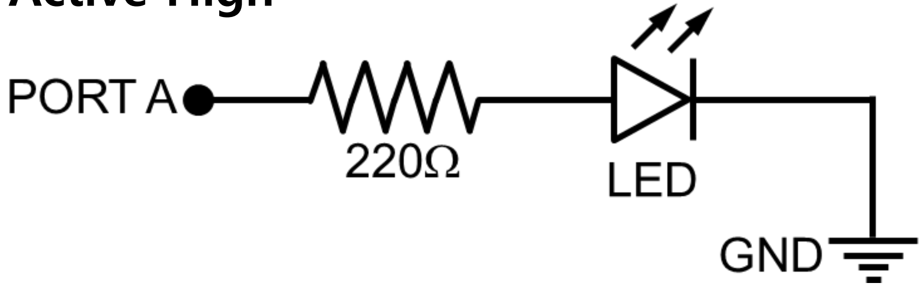


```
import processing.hamster.*;
import org.roboid.robot.*;

Hamster hamster;

void setup() {
  hamster = new Hamster(this);
  hamster.write(Hamster.IO_MODE_B, Hamster.IO_MODE_DI);
}

void draw() {
}

void repeat() {
  if(hamster.read(Hamster.INPUT_B) == 0) {
    hamster.write(Hamster.BUZZER, 1000);
  } else {
    hamster.write(Hamster.BUZZER, 0);
  }
}
```

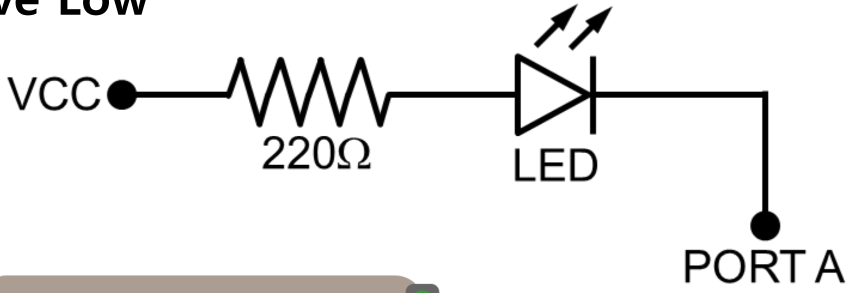
디지털 출력: 반짝반짝 LED를 깜박여요

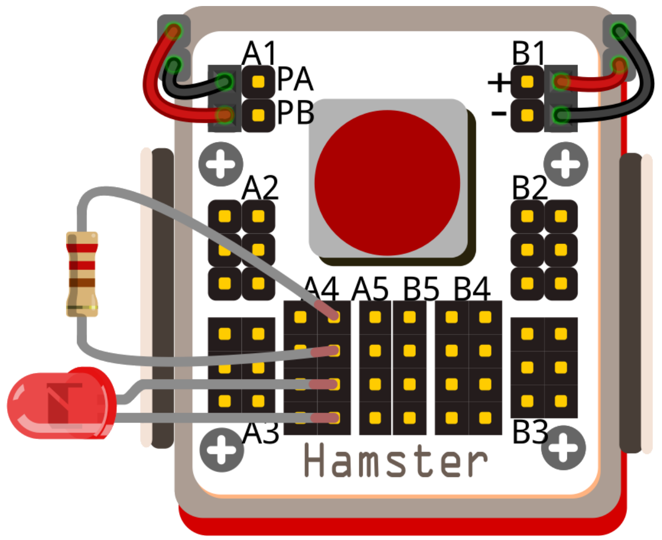


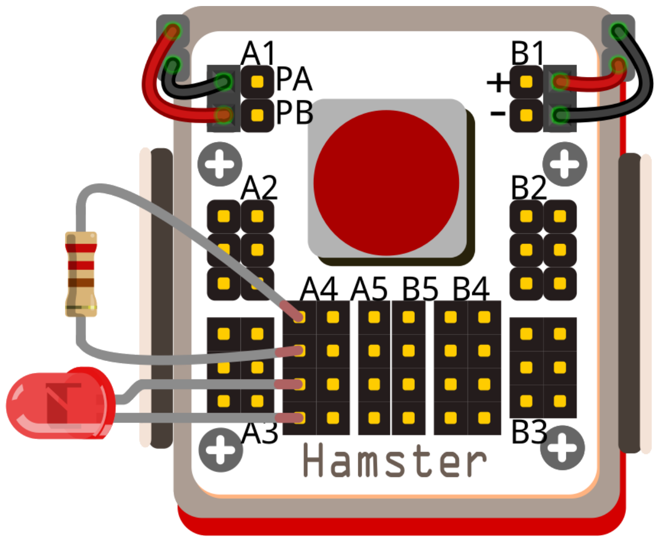
Active High



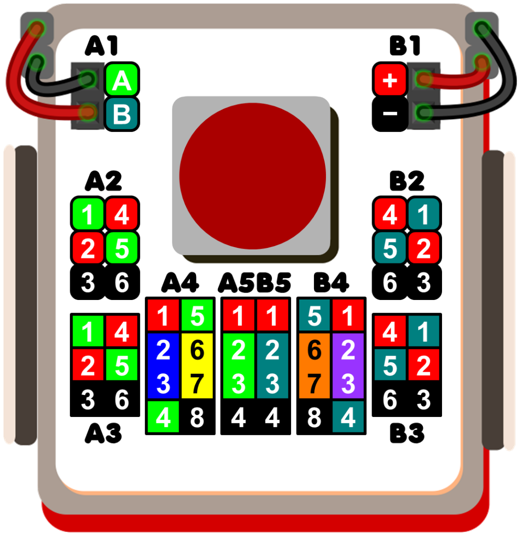
Active Low

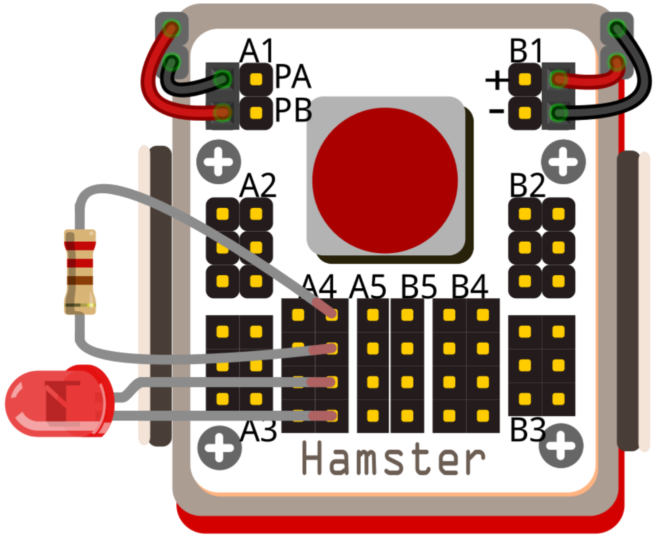






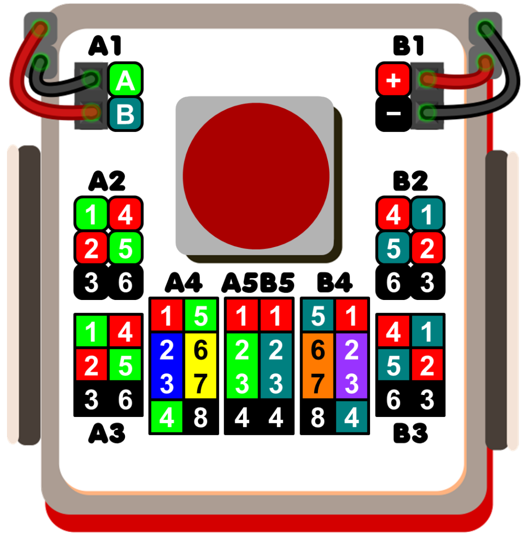
디지털 출력: 반짝반짝 LED를 깜박여요



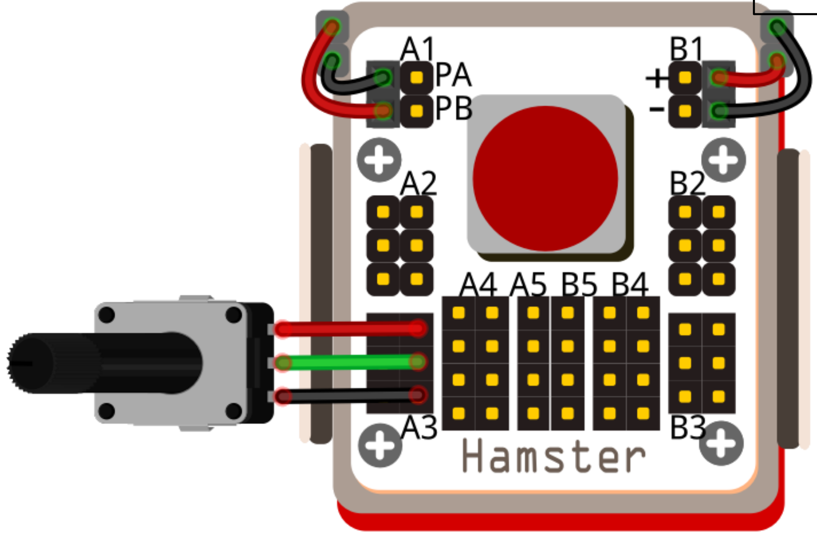


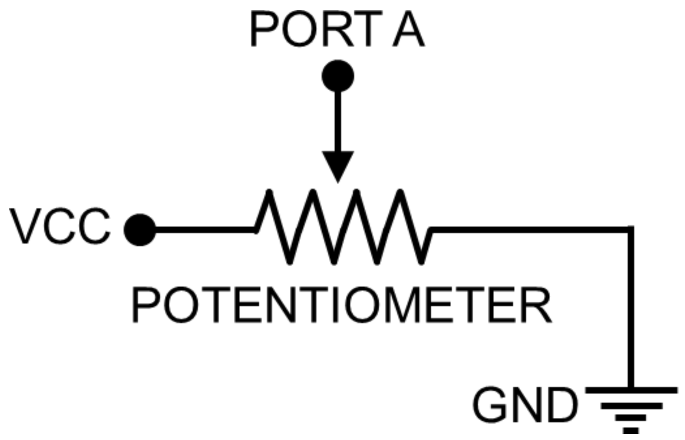
```
import processing.hamster.*;
import org.roboid.robot.*;

Hamster hamster;

void setup() {
  hamster = new Hamster(this);
  hamster.write(Hamster.IO_MODE_A, Hamster.IO_MODE_DO);
}

void draw() {
}

void repeat() {
  hamster.write(Hamster.OUTPUT_A, 1);
  delay(1000);
  hamster.write(Hamster.OUTPUT_A, 0);
  delay(1000);
}
```

아날로그 입력: 포텐셔미터를 돌리면 음 높이가 달라져요

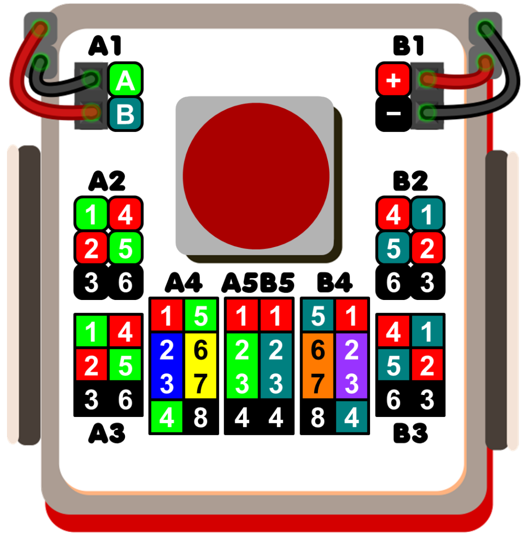


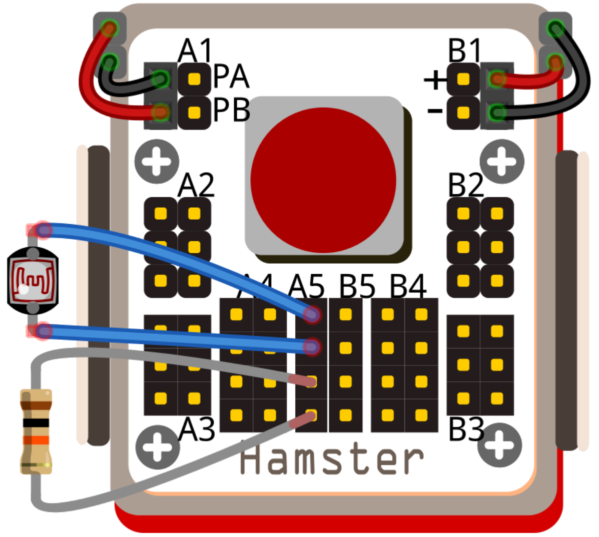
```
import processing.hamster.*;
import org.roboid.robot.*;

Hamster hamster;

void setup() {
  hamster = new Hamster(this);
  hamster.write(Hamster.IO_MODE_A, Hamster.IO_MODE_ADC);
}

void draw() {
}

void repeat() {
  int hz = hamster.read(Hamster.INPUT_A) * 10;
  hamster.write(Hamster.BUZZER, hz);
}
```

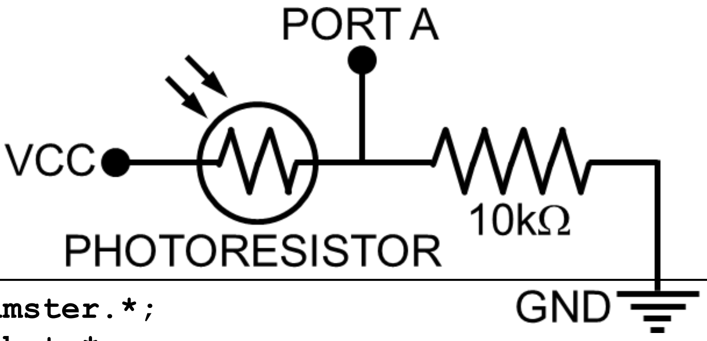




아날로그 입력: 빛을 따라 움직여요

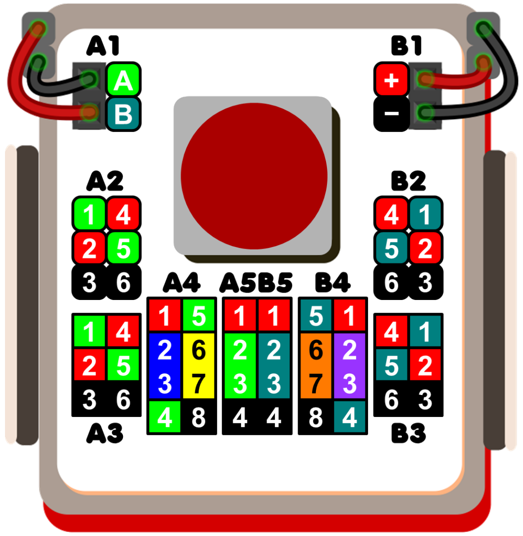


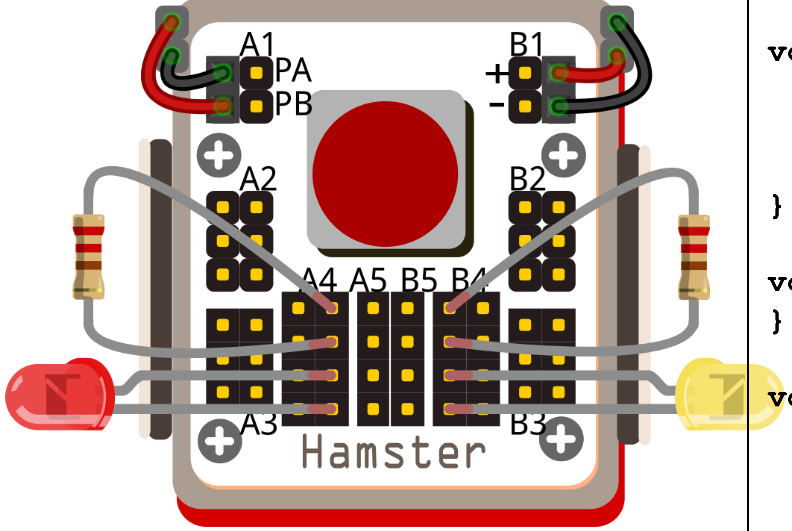


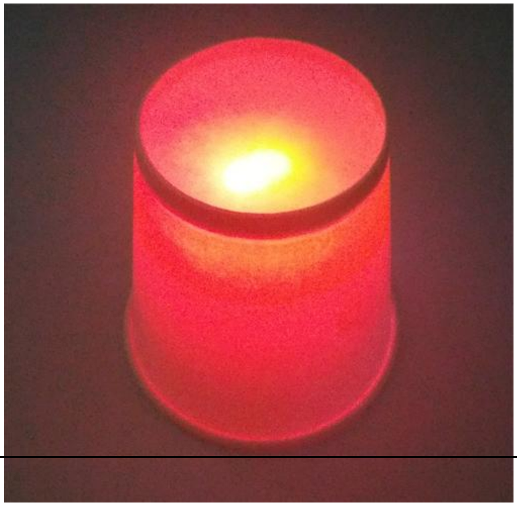


```
import processing.hamster.*;
import org.roboid.robot.*;

Hamster hamster;

void setup() {
  hamster = new Hamster(this);
  hamster.write(Hamster.IO_MODE_A, Hamster.IO_MODE_ADC);
}

void draw() {
}

void repeat() {
  if(hamster.read(Hamster.LIGHT) > 180) {
    hamster.write(Hamster.LEFT_WHEEL, 30);
    hamster.write(Hamster.RIGHT_WHEEL, 30);
  } else if(hamster.read(Hamster.INPUT_A) > 100) {
    hamster.write(Hamster.LEFT_WHEEL, -30);
    hamster.write(Hamster.RIGHT_WHEEL, -30);
  } else {
    hamster.write(Hamster.LEFT_WHEEL, 0);
    hamster.write(Hamster.RIGHT_WHEEL, 0);
  }
}
```

PWM 출력: LED 촛불이 바람에 흔들려요

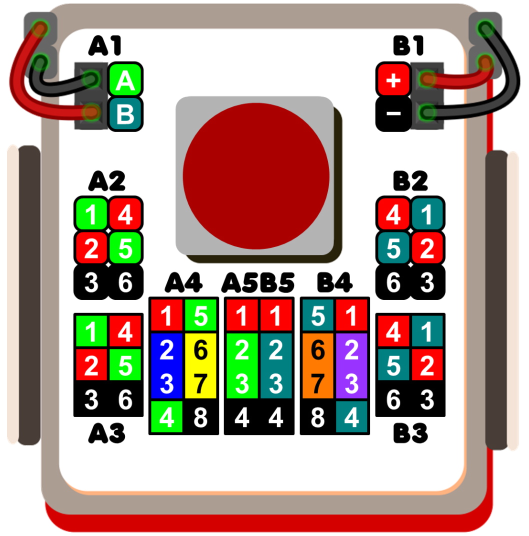




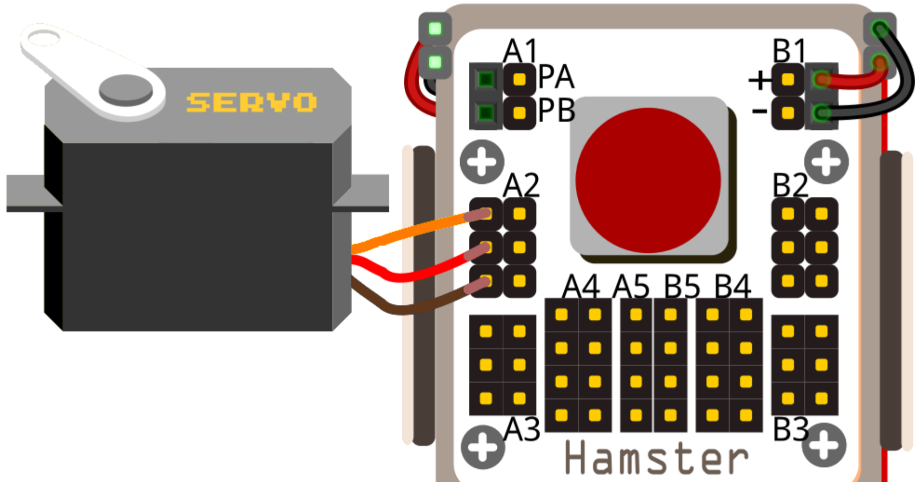


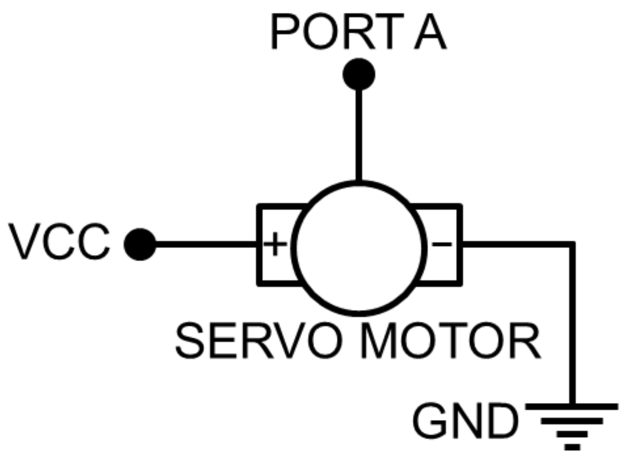
```
import processing.hamster.*;
import org.roboid.robot.*;

Hamster hamster;

void setup() {
  hamster = new Hamster(this);
  hamster.write(Hamster.IO_MODE_A, Hamster.IO_MODE_PWM);
  hamster.write(Hamster.IO_MODE_B, Hamster.IO_MODE_PWM);
}

void draw() {
}

void repeat() {
  hamster.write(Hamster.OUTPUT_A, int(random(100, 256)));
  hamster.write(Hamster.OUTPUT_B, int(random(100, 256)));
  delay(int(random(0, 100)));
}
```

서보 출력: 햄스터 로봇에게 꼬리가 생겼어요



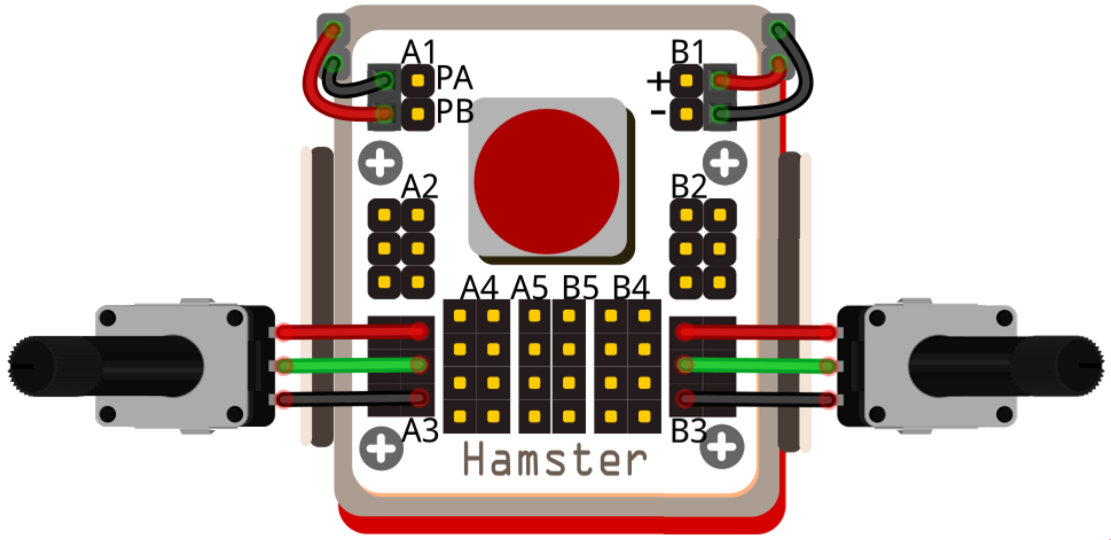
```
import processing.hamster.*;
import org.roboid.robot.*;

Hamster hamster;

void setup() {
  hamster = new Hamster(this);
  hamster.write(Hamster.IO_MODE_A, Hamster.IO_MODE_SERVO);
}

void draw() {
}

void repeat() {
  hamster.write(Hamster.OUTPUT_A, 10);
  delay(1000);
  hamster.write(Hamster.OUTPUT_A, 180);
  delay(1000);
}
```

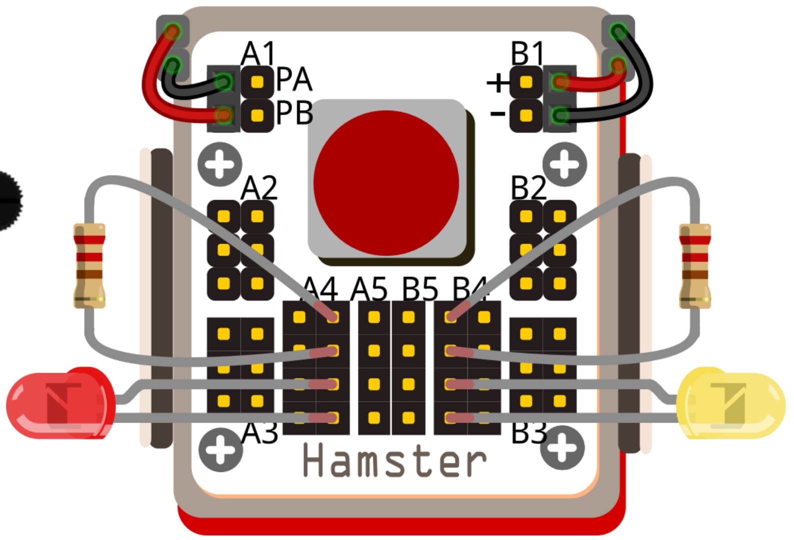




미션

- 햄스터 조종기 (2인 1조)
  - 첫 번째 햄스터의 포텐셔미터를 돌려서
  - 두 번째 햄스터를 조종하기
  - 두 번째 햄스터는 방향에 따라 LED 깜박이기





수고하셨습니다.

http://hamster.school

akaii@kw.ac.kr