



Dobot CRStudio User Guide

(CR & Nova)



목차

머리말	1.1
1 시작하기	1.2
2 로봇에 연결하기	1.3
3 메인 인터페이스	1.4
3.1 개요	1.4.1
3.2 상단 툴바	1.4.2
3.3 3D 모델 패널	1.4.3
3.4 주요 기능 패널	1.4.4
3.5 조그보드	1.4.5
4 설정	1.5
4.1 공구 좌표계	1.5.1
4.2 사용자 좌표계	1.5.2
4.3 조그 설정	1.5.3
4.4 재생 설정	1.5.4
4.5 안전 설정	1.5.5
4.5.1 안전한 충돌	1.5.5.1
4.5.2 조인트 브레이크	1.5.5.2
4.5.3 종단부하	1.5.5.3
4.5.4 드래그 감도	1.5.5.4
4.5.5 고급 기능	1.5.5.5
4.6 원격 제어	1.5.6
4.7 로봇 자세	1.5.7
4.8 설치	1.5.8
4.9 소프트웨어 설정	1.5.9
4.10 재조소	1.5.10
4.11 전원 전압(CCBOX)	1.5.11
5 모니터링	1.6
5.1 입출력 모니터	1.6.1
5.2 모드버스	1.6.2
5.3 글로벌 변수	1.6.3
5.4 로봇 상태	1.6.4

5.5 실행 로그	1.6.5
5.6 두봇+	1.6.6
6 프로그래밍	1.7
6.1 블록리	1.7.1
6.2 스크립트	1.7.2
7 프로세스	1.8
7.1 궤적 재생	1.8.1
7.2 컨베이어 벨트	1.8.2
8 모범 사례	1.9
부록 A Modbus 레지스터 정의	1.10
부록 B 블록 명령	1.11
B.1 빠른 시작	1.11.1
B.1.1 제어 로봇 동작	1.11.1.1
B.1.2 Modbus 레지스터 데이터 읽기 및 쓰기	1.11.1.2
B.1.3 TCP 통신으로 데이터 전송	1.11.1.3
B.1.4 팔레이타이즈	1.11.1.4
B.2 블록 설명	1.11.2
B.2.1 이벤트	1.11.2.1
B.2.2 제어	1.11.2.2
B.2.3 연산자	1.11.2.3
B.2.4 문자열	1.11.2.4
B.2.5 커스텀	1.11.2.5
B.2.6 출력력	1.11.2.6
B.2.7 모션	1.11.2.7
B.2.8 모션 고급 구성	1.11.2.8
B.2.9 모드버스	1.11.2.9
B.2.10 TCP	1.11.2.10
부록 C 스크립트 명령	1.12
C.1 루아 기본 문법	1.12.1
C.1.1 변수 및 데이터 유형	1.12.1.1
C.1.2 조작자	1.12.1.2
C.1.3 공정 제어	1.12.1.3
C.2 명령 설명	1.12.2
C.2.1 모션	1.12.2.1

C.2.2 모션 파라미터	1.12.2.2
C.2.3 상대 운동	1.12.2.3
C.2.4 IO	1.12.2.4
C.2.5 TCP/UDP	1.12.2.5
C.2.6 모드버스	1.12.2.6
C.2.7 프로그램 제어	1.12.2.7
C.2.8 비전	1.12.2.8

머리말

목적

이 문서는 편리하고 편리한 두봇 산업용 로봇의 모바일 앱인 두봇 CR Studio를 소개합니다.
사용자가 CR Studio를 이해하고 사용하여 산업용 로봇을 제어할 수 있도록 합니다.

대상 청중

이 문서의 대상은 다음과 같습니다.

- 고객
- 영업 엔지니어
- 설치 및 시운전 엔지니어
- 기술 지원 엔지니어

변경 내역

날짜	변경 로그
2023/01/13	일곱 번째 릴리스(Android V4.11.0/iOS V2.12.0)
2022년 12월 9일	6개 릴리스(Android V4.10.0/iOS V2.11.0)
2022년 10월 31일	다섯 번째 릴리스(Android V4.9.0/iOS V2.10.0)
2022/08/25	네 번째 릴리스(Android V4.8.0/iOS V2.9.0)
2021/05/28	세 번째 릴리스
2020/12/04	두 번째 릴리스
2020년 11월 16일	첫 번째 릴리스

기호 규약

이 문서에서 볼 수 있는 기호는 다음과 같이 정의됩니다.

상징	설명
 위험	피하지 않을 경우 사망 또는 중상을 초래할 수 있는 높은 수준의 위험을 나타냅니다.
	그렇지 않은 경우 중간 수준 또는 낮은 수준의 위험이 있는 위험을 나타냅니다.

 경고 G	피하지 않을 경우 경미하거나 중간 정도의 부상, 로봇 팔 손상을 초래할 수 있는 중간 수준 또는 낮은 수준의 위험을 나타냅니다.
 알아채다	피하지 않을 경우 로봇 팔 손상, 데이터 손실 또는 예기치 않은 결과를 초래할 수 있는 잠재적으로 위험한 상황을 나타냅니다.
 메모	본문에서 중요한 사항을 강조하거나 보완하기 위한 추가 정보를 제공합니다.

1 시작하기

씨알 스튜디오에 오신 것을 환영합니다. CR Studio는 산업용 로봇 팔을 위해 Yuexijiang에서 개발한 모바일 제어 소프트웨어입니다. 간단한 기능과 인터페이스, 강력한 실용성을 갖춘 CR Studio는 두봇 산업용 로봇 팔의 사용법을 빠르게 마스터하는 데 도움이 될 수 있습니다.

이 가이드는 CR Studio를 사용하여 CR 로봇(6축)을 제어하는 방법을 주로 설명합니다.

CR Studio를 처음 사용하는 경우 다음 위치에서 이 가이드를 읽는 것이 좋습니다.

주문하다.

1. [Main Interface](#) : 씨알스튜디오의 메인인터페이스에 대해 먼저 알아보고,
CR Studio의 기능을 대략적으로 이해합니다.
2. [로봇에 연결하기](#): CR Studio를 로봇 팔에 연결하고 체험을 시작합니다.
3. [설정](#): 실제 요구 사항에 따라 로봇 팔을 구성합니다. 로봇이 천장이나 벽걸이 모드 또는 특정 각도로 설치된 경우 아래에서 회전 각도와 기울기 각도를 설정해야 합니다.
비활성화 상태. 자세한 내용은 [설치](#)를 참조하십시오 .
4. [모니터링](#) : CR Studio에서 제공하는 모니터링 기능에 대해 알아본다. end를 설치해야 하는 경우
플러그인, 구성은 [Dobot+](#)를 참조하십시오 .
5. [프로그래밍/프로세스](#): CR Studio의 프로그래밍 및 프로세스 모듈에 대해 알고 시도해 보십시오.
나만의 프로젝트 만들기.
6. [원격제어](#) : 프로젝트 개발 후 원격제어를 통해 프로젝트를 실행해 보세요.

2. 로봇에 연결하기

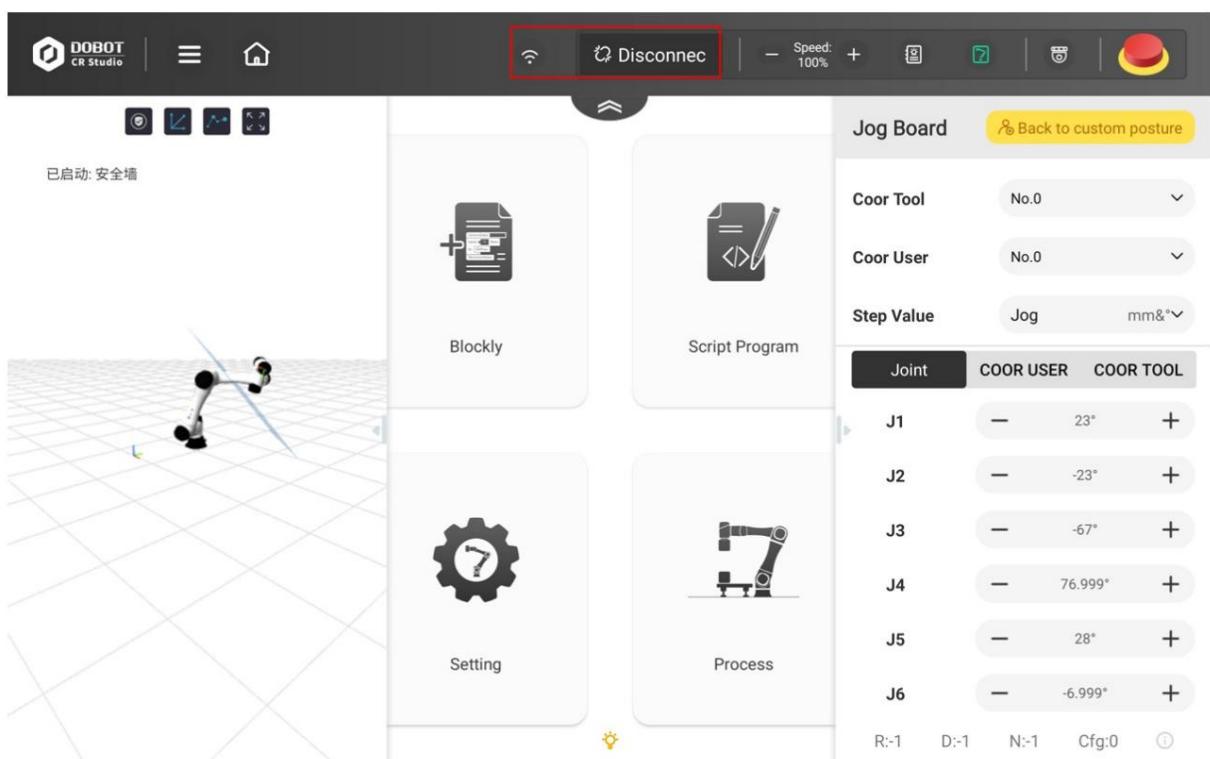
로봇에 연결하기 전에 컨트롤러에 WiFi 모듈이 설치되어 있는지 확인하십시오.

1. 태블릿에서 두봇 컨트롤러 WiFi를 검색하여 연결합니다. WiFi 이름은 "Dobot_WIFI_XXX"(XXX는 팔 베이스의 로봇 색인)이고 WiFi 비밀번호는 기본적으로 1234567890입니다.

관리자 모드의 [소프트웨어 설정](#)에서 WiFi SSID 및 암호를 수정할 수 있습니다. 그만큼 수정 사항은 컨트롤러를 다시 시작한 후에 적용됩니다.

2. 기본 인터페이스에서 연결을 클릭하여 컨트롤러에 연결합니다.

컨트롤러에 성공적으로 연결되면 연결 버튼이 연결 해제로 전환됩니다. 아래와 같이 WiFi 아이콘이 버튼 왼쪽에 표시됩니다.



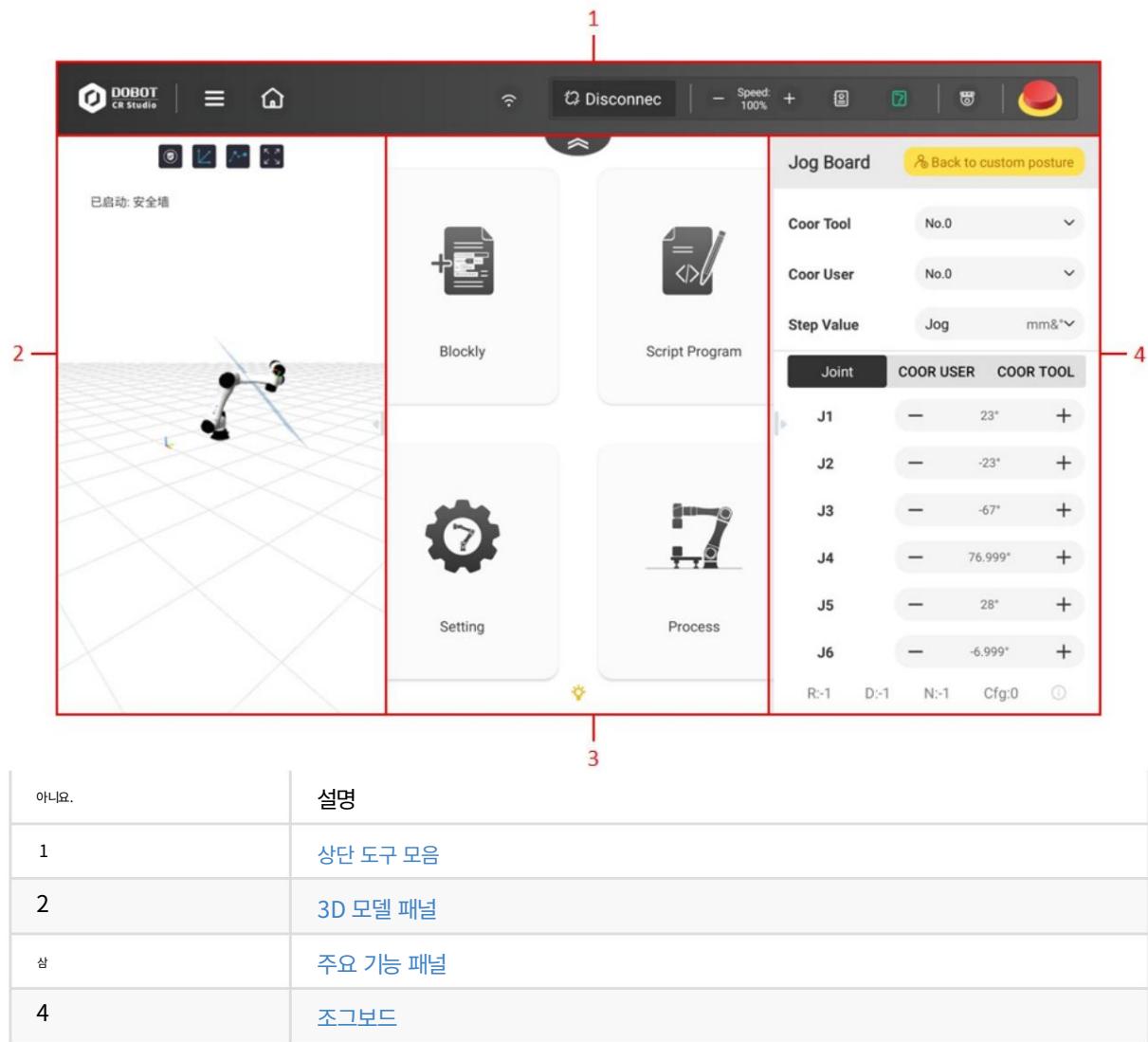
3 메인 인터페이스

- 3.1 개요
- 3.2 상단 툴바 3.3
- 3D 모델 패널 3.4 주요 기
- 능 패널 3.5 조그 보드
-

3.1 개요

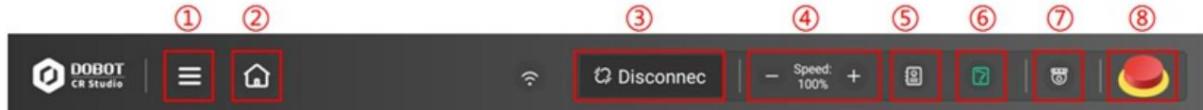
CR Studio는 Android 및 iOS 시스템에서 작업을 지원합니다. Android 태블릿 또는 iPad에 설치하는 것이 좋습니다. 두 시스템의 기본 기능과 UI 디자인은 거의 동일합니다. 이 가이드는 Android 태블릿을 예로 들어 CR Studio 사용을 소개합니다.

아래와 같이 앱을 시작하여 메인 페이지로 들어갑니다. 메인 페이지 기능  , 그리고 당신은에 대한 소개를 볼 수 있습니다 을 클릭합니다.



3.2 상단 툴바

도구 모음은 CR Studio를 사용할 때 항상 화면 상단에 있습니다(도구 모음은 ). 기능은 아래에 나열되어 있습니다.



아이콘	이름	설명
1	메뉴	<p>아이콘을 클릭하면 다음 항목이 나타납니다. 도움말: 도움말 문서 보기</p> <ul style="list-style-type: none"> 잠금: 화면 잠금(시스템 화면이 아닌 앱 화면 잠금). 화면 잠금을 해제하려면 관리자 비밀번호 또는 화면 잠금 비밀번호를 입력하세요. 충돌 로그: 문제 해결에 도움이 되는 앱 충돌 로그를 업로드합니다. 버전 정보: 로봇 및 앱의 버전 정보를 볼니다. 회사 소개: Yuejiang Technology Co., Ltd의 정보를 볼니다. . • • • 앱 종료: CR Studio 앱 종료
2	홈페이지	기본 인터페이스로 돌아가려면 클릭하십시오. 다른 인터페이스에서 메인 인터페이스로 돌아가기 위해 이 버튼을 클릭하면 저장되지 않은 콘텐츠는 손실되지 않습니다.
3	연결	CR 로봇의 WiFi를 연결한 후 연결을 클릭하여 앱을 로봇 팔에 연결합니다. 자세한 내용은 로봇에 연결을 참조하십시오 . 연결 후 버튼이 연결 해제로 변경됩니다. 버튼을 다시 클릭하면 로봇 팔을 분리할 수 있습니다.
4	글로벌 속도 비율	전역 속도 비율을 설정합니다. 전체 속도 비율은 로봇 팔의 실제 실행 속도의 계산 요소입니다. 계산 방법은 조그 설정 및 재생 설정을 참조하십시오 .
5	알람 정보	로봇 암을 비정상적으로 사용하면 알람이 트리거됩니다. 알람이 없으면 아이콘이 흰색이고 알람이 있으면 빨간색입니다. 아이콘을 클릭하면 알람 정보가 표시됩니다 .
6	활성화 버튼	로봇 팔의 활성화 상태를 전환하려면 클릭합니다. 자세한 내용은 활성화 상태를 참조하십시오 .
7	모니터링 버튼	IO 모니터 인터페이스를 열려면 클릭하십시오. 자세한 내용은 IO 모니터를 참조하십시오 .
8	비상 정지 버튼	긴급 상황에서 버튼을 누르면 로봇 팔이 작동을 멈추고 전원이 꺼집니다. 자세한 내용은 비상 정지 버튼을 참조하십시오 .

알람 정보

알람 정보 인터페이스는 아래 그림과 같습니다.

Id	Type	Level	Reason	Solution
12288	controller	0	Press the emergency stop button	Clear the alarm and power on again
12296	controller	0	The robot is powered off	Clear the alarm and power on again, or contact technical support engineer

이 페이지에서 알람 정보를 볼 수 있습니다. Type 은 알람을 보낸 사람인지 구분하기 위해 사용합니다.
로봇 팔 또는 컨트롤러.

알람이 있을 때 로봇 암의 전원이 깨지지 않으면 종료 표시등이 빨간색으로 바뀝니다.

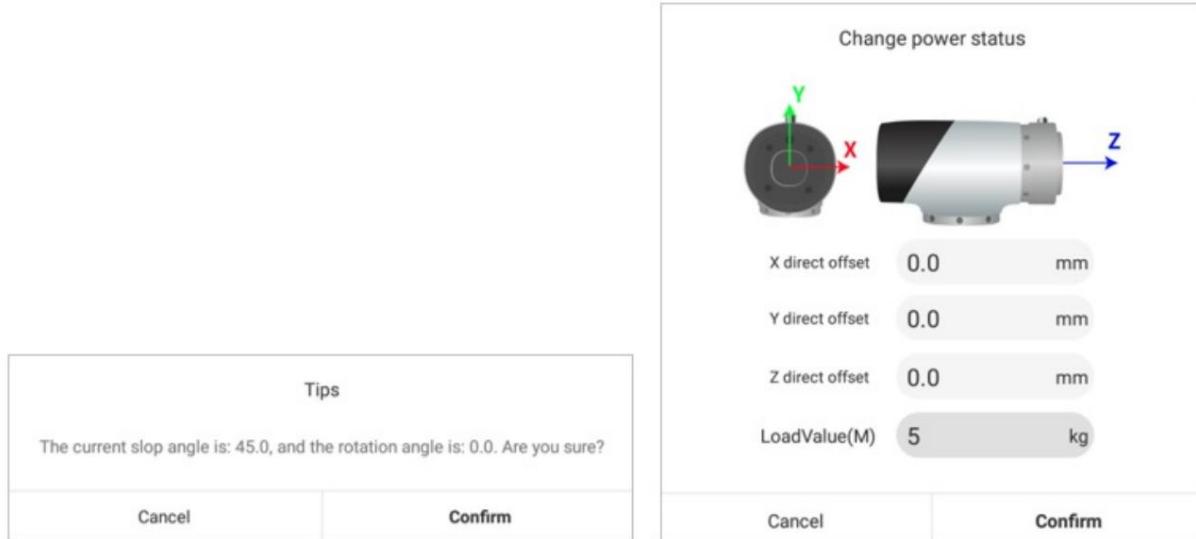
알람의 원인이 된 오류를 해결한 후 알람 해제를 클릭하여 알람을 해제하십시오.

활성화 상태

로봇 팔은 활성화된 상태에서만 작동할 수 있습니다.

- 활성화 버튼이 파란색일 때("Tips" 및), 로봇 팔이 비활성화 상태입니다. 버튼을 클릭하고 "Change power status" 창이 팝업됩니다(끝 하중의 편심 좌표는 J6 축이 0°일 때 설정되어야 하며 하중 값은 (CR 로봇의 최대 허용 하중 중량) 매개 변수를 설정한 후 확인을 클릭하여 로봇을 활성화합니다.로봇 팔이 약간 움직이기 시작한 다음 끝 표시등이 녹색으로 바뀌어

로봇 팔이 활성화되었는지 확인합니다. 동시에 활성화 버튼이 녹색().



- 활성화 버튼이 녹색이면 로봇 팔이 활성화된 상태입니다. 버튼을 클릭하면 확인 상자가 표시됩니다. 확인 후 로봇 팔이 비활성화되기 시작합니다. 로봇 팔 끝에 있는 표시등이 파란색으로 바뀌어 로봇 팔이 비활성화되었음을 나타냅니다. 동시에 활성화 버튼도 파란색으로 바뀝니다.
- 활성화 버튼이 파란색으로 깜박이면 로봇이 드래그 모드에 있는 것입니다. 이 경우 로봇을 비활성화하거나 로봇 동작을 제어할 수 없습니다(프로젝트 실행, 조깅, 지정된 자세로 실행 등). 소프트웨어.
- CR Studio에 연결된 상태에서 로봇 팔의 전원이 꺼지면 활성화 버튼이 흰색으로 바뀝니다. 활성화 상태를 변경하려면 먼저 로봇의 전원을 켜야 합니다.

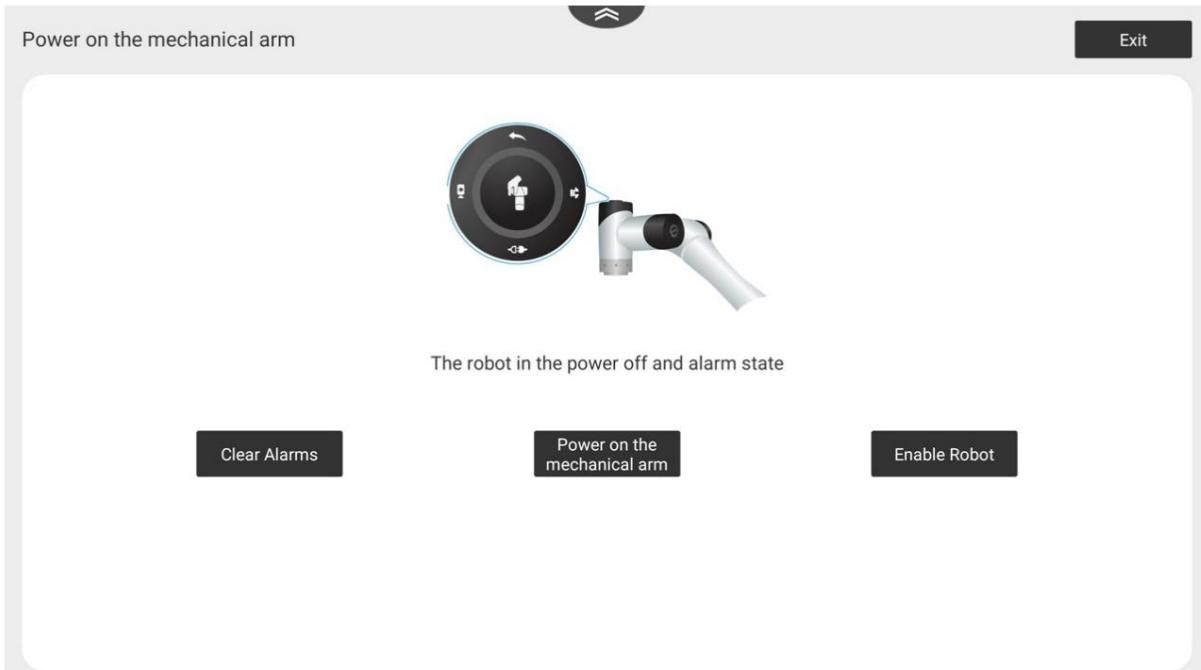
비상 정지 버튼

비상 정지 버튼을 누르면 로봇 팔이 작동을 멈추고 전원이 꺼집니다.

비상 정지 아이콘이 눌린 상태(



, 아래 그림과 같이.



로봇 팔을 다시 활성화해야 하는 경우 비상 정지 버튼을 재설정하고 알람을 지우고 전원을 켜십시오.
로봇을 활성화한 다음 활성화합니다.



메모

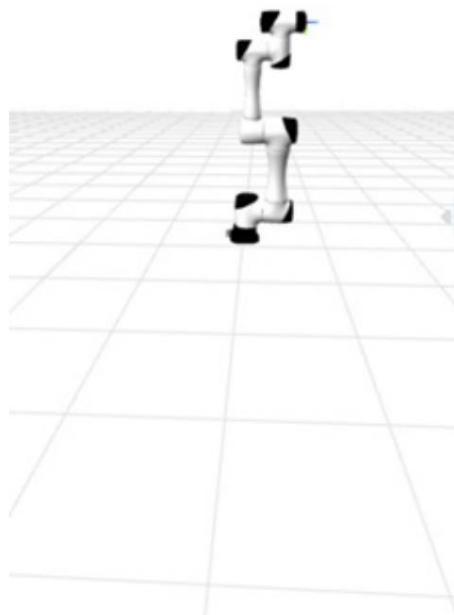
- 물리적 비상 정지 버튼을 누르면 소프트웨어의 비상 정지 버튼 아이콘이 변경되지 않습니다. 알람을 해제하기 전에 먼저 물리적 비상 정지 버튼을 재설정해야 합니다(일반적으로 버튼을 시계 방향으로 회전).
- 원격 제어 모드에서는 페이지가 표시되지 않습니다.

3.3 3D 모델 패널

메인 인터페이스의 왼쪽에 있는 3D 모델 패널은 로봇의 현재 자세를 표시하는 데 사용됩니다. 패널의 오른쪽 여백을 누르고 왼쪽으로 드래그하면 패널을 숨길 수 있습니다.



이 패널은 기본적으로 숨겨져 있는 블록 인터페이스에서도 사용할 수 있습니다. 왼쪽 여백을 누르고 오른쪽으로 드래그하여 패널을 표시 할 수 있습니다.



패널 영역에서 누르고 이동하여 관찰을 위한 시야각을 변경할 수 있습니다. 손가락 제스처를 통해 확대 또는 축소할 수 있습니다. 설명된 바와 같이 패널 상단에 4개의 기능 버튼이 있습니다.

아래에.



: 로봇의 작업 영역을 표시하거나 숨깁니다.



: 기준 좌표계를 표시하거나 숨깁니다.



: 최근 10초간 로봇의 이동 궤적 표시/숨기기

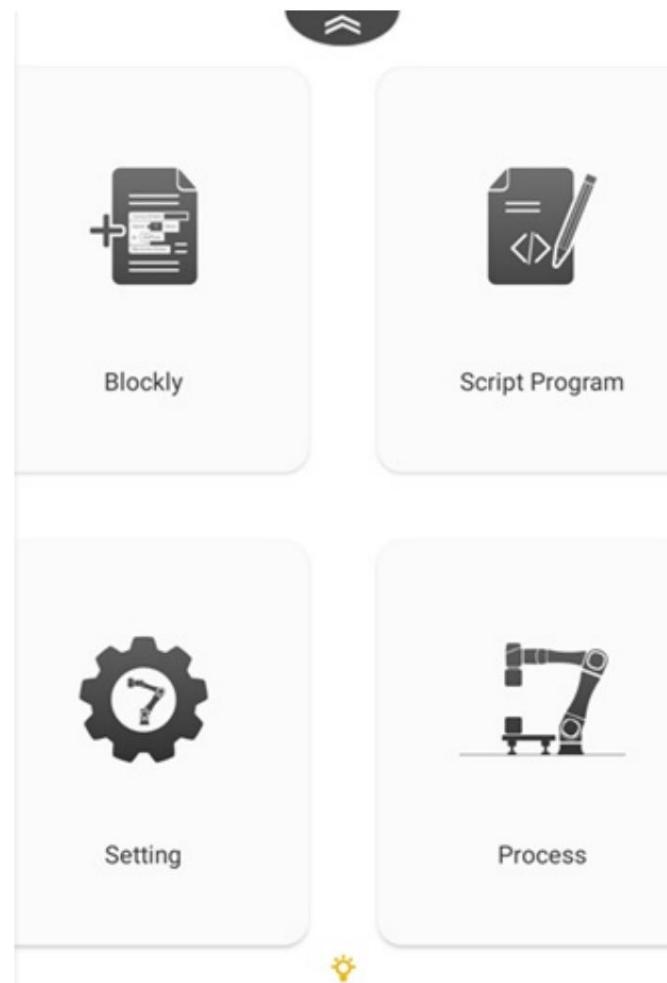


: 3D 모델 표시 영역을 확대/축소합니다.

3.4 주요 기능 패널

주요 기능 모듈은 기본 인터페이스의 중앙 부분에 있으며 이를 통해 액세스할 수 있습니다.

이러한 모듈.



Blockly: 클릭하면 Blockly 페이지로 들어갑니다. 자세한 내용은 [Blockly](#)를 참조하세요 .

스크립트: 클릭하면 스크립트 페이지로 들어갑니다. 자세한 내용은 [스크립트](#)를 참조하십시오 .

설정: 클릭하면 설정 페이지로 들어갑니다. 자세한 내용은 [설정](#)을 참조하십시오 .

프로세스: 클릭하면 프로세스 페이지로 들어갑니다. 자세한 내용은 [프로세스](#)를 참조하십시오 .

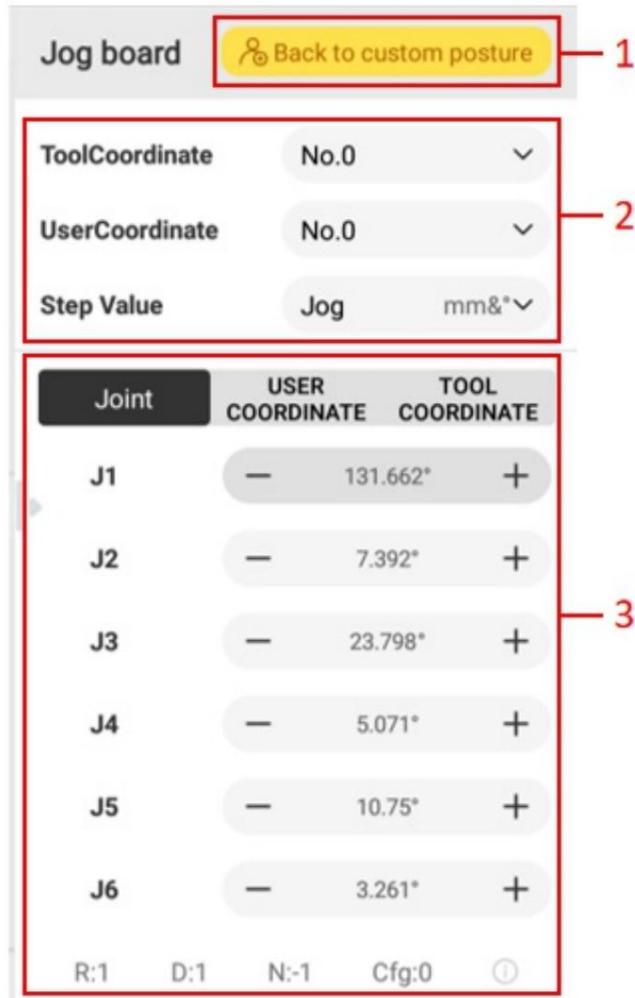
3.5 조그보드

메인 인터페이스의 오른쪽에 있는 조그 보드는 로봇의 움직임을 제어하는 데 사용됩니다. 패널을 누르고 오른쪽으로 드래그하면 보드를 숨길 수 있습니다.



메모

이 패널은 기본적으로 숨겨져 있는 다른 인터페이스에서도 사용할 수 있습니다. 오른쪽 여백을 누르고 왼쪽으로 드래그하여 패널을 표시할 수 있습니다.



맞춤 자세로 돌아가기

맞춤 자세로 돌아가기를 길게 누르면 로봇이 맞춤 자세로 이동합니다. 사용자 정의 자세를 설정하려면 [설정](#)을 참조하십시오.

좌표계 및 단계 값

ToolCoordinate, UserCoordinate 및 단계 값을 선택할 수 있습니다.

1. ToolCoordinate: [도구 좌표계](#) 참조。
2. UserCoordinate: [사용자 좌표계](#) 참조。
3. Step Value: 단일 조그의 변위 값
 - 조그는 조그 버튼을 누르고 있으면 로봇이 계속 움직이고 조그 버튼에서 손을 떼면 움직이지 않는 것을 말합니다.
 - 특정 값(예: 0.1)은 조그 버튼을 누를 때 로봇이 이 값을 이동한 다음 이동을 중지한다는 것을 나타냅니다.
 - 데카르트 좌표계에서 이 값의 단위는 mm이며 0.1은 각 조그당 0.1mm의 변위를 나타냅니다.
 - 관절 좌표계에서 이 값의 단위는 °이며 0.1은 조그당 0.1°의 변위를 나타냅니다.

조그 조작 패널

CR Studio는 3개의 좌표계에서 조그 작업을 지원합니다. 각 축의 + 또는 -를 클릭하거나 길게 눌러 로봇을 조그할 수 있습니다.

조그 보드 하단의 R/D/N/Cfg는 각 암의 방향을 나타냅니다. 설명을 클릭합니다.

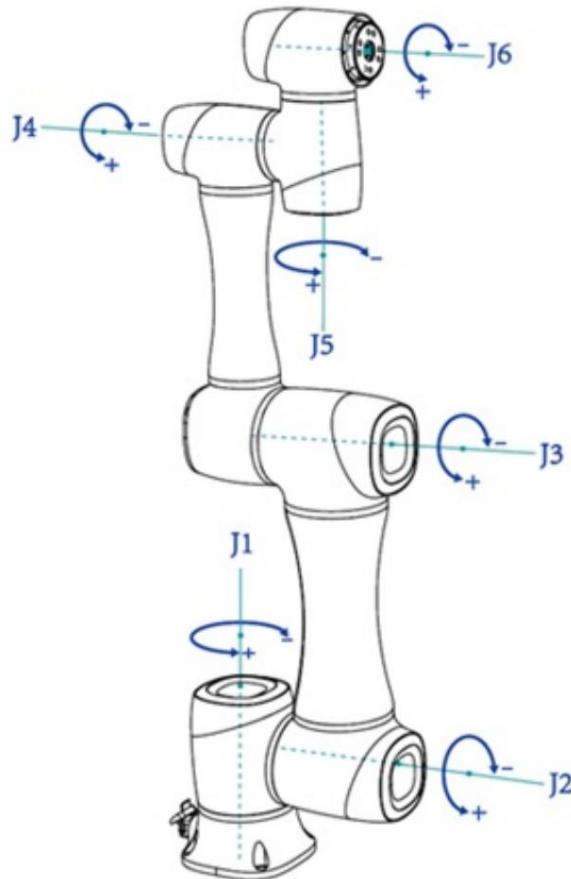


구체적으로 보기 위해

관절 좌표계

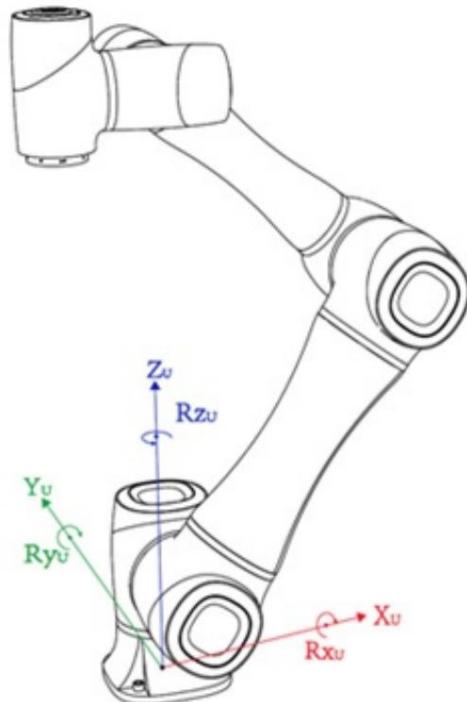
관절 좌표계는 움직이는 관절을 기준으로 결정됩니다. 모든 관절은 다음과 같이 회전 관절입니다.

아래에 나와 있습니다.



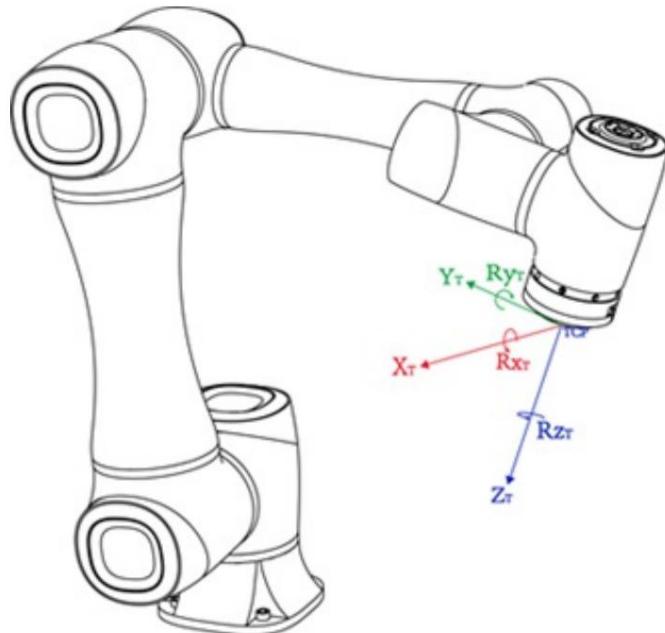
사용자 좌표계

사용자 좌표계는 워크벤치 또는 공작물의 사용자 정의 좌표계입니다. 각 축의 원점과 방향은 실제 필요에 따라 결정될 수 있습니다. 기본 사용자 좌표계는 아래와 같습니다.



도구 좌표계

공구 좌표계는 공구 중심점(TCP)의 위치와 공구 자세를 정의합니다. 끝 공작물의 위치와 각도에 따라 원점과 방향이 끊임 없이 바뀝니다. 기본 도구 좌표계는 아래와 같습니다.



4 설정

- 4.1 공구 좌표계 4.2 사용자 좌표계
- 4.3 조그 설정 4.4 재생 설정 4.5 안
- 전 설정
- - 4.5.1 안전한 충돌
 - 4.5.2 조인트 브레이크
 - 4.5.3 종단부하
 - 4.5.4 드래그 감도
 - 4.5.5 고급 기능
- 4.6 원격 제어
- 4.7 로봇 자세
- 4.8 설치
- 4.9 소프트웨어 설정 4.10
- 제조 4.11 전원 전압
- (CCBOX)

4.1 공구 좌표계

용접 토치나 그리퍼와 같은 앤드 이펙터가 로봇에 장착될 때 로봇을 프로그래밍하고 작동하기 위해서는 툴 좌표계가 필요합니다. 예를 들어 여러 그리퍼를 사용하여 여러 공작물을 동시에 처리하는 경우 각 그리퍼에 대한 도구 좌표계를 설정하여 효율성을 향상시킬 수 있습니다.

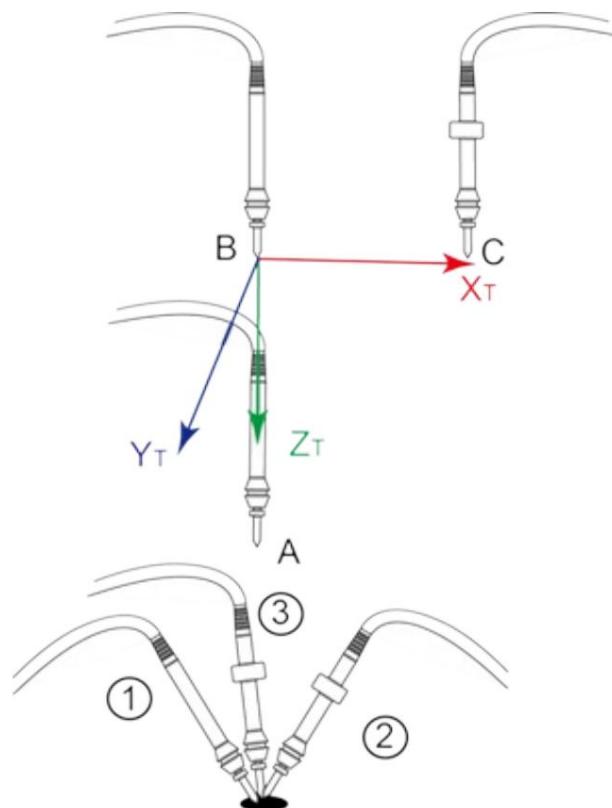
CR Studio는 10개의 도구 좌표계를 지원합니다. 도구 좌표계 0은 도구가 사용되지 않고 변경할 수 없음을 의미하는 기본 좌표계입니다.



메모

도구 좌표계를 생성할 때 참조 좌표계가 기본 도구 좌표계인지 확인하십시오.

6축 도구 좌표계는 3점 보정 방법(TCP + ZX)으로 생성됩니다. 앤드 이펙터를 장착한 후 앤드 이펙터의 방향을 조정하여 TCP(Tool Center Point)가 동일한 지점(기준점)에 정렬되도록 합니다.) 앤드 이펙터의 위치 오프셋을 얻기 위해 세 가지 다른 방향으로. 그런 다음 로봇을 세 개의 다른 지점(A, B, C)으로 조그하여 각도 오프셋을 얻습니다.



도구 좌표계 추가

전제 조건

- 로봇 팔의 전원이 켜졌습니다.
- 로봇 팔이 활성화되었습니다.

절차

The screenshot shows the 'Coor Tool' interface. At the top, there are buttons for '+ Add', 'Cover', 'Modify', and 'Delete'. Below is a table with columns: No, Alias, Offset, X, Y, Z, RX, RY, and RZ. A single row is present with 'No.0' and '0.0 cm' in the Offset column. The X, Y, Z, RX, RY, and RZ values are all 0. Below the table is a 3D coordinate system diagram with a central cube and axes labeled X, Y, and Z. A message at the bottom states: 'There is no tool coordinate yet, you need to calibrate point first'. A note below says: 'Get point (The first three points get the tool position; the latter three points get the tool posture (XYZ direction))'. There are three sections for 'First Point', 'Second Point', and 'Third Point', each with 'Run to' buttons and 'GET' buttons. The entire 'Get point' section is highlighted with a red border.

This screenshot shows the same 'Coor Tool' interface as above, but the 'Get point' section is now empty. The 'First Point', 'Second Point', and 'Third Point' sections have been cleared, and their 'GET' buttons are visible. The entire 'Get point' section is highlighted with a red border.



- 원쪽 상단을 클릭하면 페이지의 지침을 볼 수 있으며 보다 직관적입니다.
문서.
- 9개의 좌표계가 추가된 경우 추가 버튼이 사라지고 더 이상 좌표계를 추가할 수 없음을 나타냅니다.

1. 엔드 이펙터를 로봇에 장착합니다.

2. 로봇을 조그하여 TCP가 첫 번째 항목의 기준점(공간의 한 점)과 정렬되도록 합니다.

방향. 그런 다음 First Point 패널에서 GET을 클릭합니다.

3. TCP가 두 번째 방향의 기준점과 정렬되도록 로봇을 조깅합니다. 그런 다음 밤기를 클릭하십시오

두 번째 점 패널에서

4. 로봇을 조그하여 TCP가 세 번째 방향의 기준점에 정렬되도록 합니다. 그런 다음 밤기를 클릭하십시오

세 번째 점 패널에서.

5. 로봇을 조그하여 TCP가 수직 방향의 기준점에 정렬되도록 합니다. 그런 다음 밤기를 클릭하십시오

Pose 1 패널에서 네 번째 지점(A 지점)을 얻습니다.

6. Z축을 양의 방향으로 조그하여 로봇을 한 지점으로 이동합니다. 그런 다음 Pose 2에서 GET을 클릭합니다.

패널을 사용하여 다섯 번째 지점(B 지점)을 얻습니다.



이 단계는 z축의 양의 방향을 결정하는 것을 목표로 합니다.

7. 양의 방향을 따라 x축을 조그하여 로봇을 지점 C로 이동합니다(지점 A 및 지점 B와 같은 선상에 있지 않음). 그런 다음 Pose 3 패널에서

GET을 클릭하여 여섯 번째 점을 얻습니다.



이 단계는 x축의 양의 방향을 결정하는 것을 목표로 합니다. y축 방향은 오른손 법칙으로 결정할 수 있습니다.

8. 추가를 클릭합니다. 도구 좌표계가 성공적으로 생성된 것을 볼 수 있습니다.

페이지.

기타 작업

해당 좌표계의 번호 (0번 제외)를 클릭하면 좌표계를 선택할 수 있으며, 다른 열을 클릭하면 해당 파라미터의 값을 수정할 수 있습니다 (Offset은 수정할 수 없음).

실행 대상: 실행 대상을 길게 눌러 로봇 팔을 획득한 지점으로 이동합니다.

Cover: 6개 점의 좌표를 구한 후 좌표계를 선택하고 Cover를 클릭하면 이 좌표계가 새로 보정된 좌표계로 바뀝니다.

수정: 기존 좌표계의 좌표값을 수정한 후 좌표계를 선택하고 수정을 클릭하면 좌표값이 업데이트 됩니다.

삭제: 좌표계를 선택하고 삭제를 클릭하면 좌표계 값이 지워집니다.



삭제 작업은 레코드 자체를 삭제하는 것이 아니라 레코드 값을 지우는 것을 의미합니다. 이 좌표계를 재보정할 때 Cover 작업을 사용할 수 있습니다. 0번 좌표계는 삭제할 수 없습니다.

4.2 사용자 좌표계

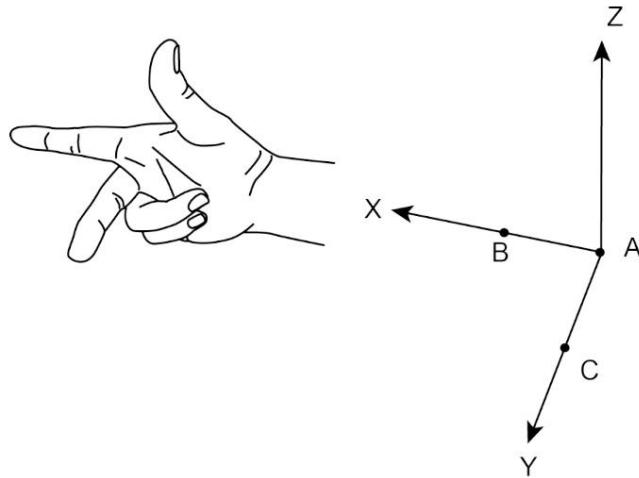
공작물의 위치가 변경되거나 로봇 프로그램을 동일한 유형의 여러 처리 시스템에서 재사용해야 하는 경우 공작물에 사용자 좌표계를 생성하여 모든 경로가 사용자 좌표와 동시에 업데이트되도록 할 수 있습니다. 프로그램 작성.

CR Studio는 10개의 사용자 좌표계를 지원합니다. 사용자 0 좌표계는 변경할 수 없는 기본 좌표계입니다.



사용자 좌표계를 생성할 때 참조 좌표계가 기본 사용자 좌표계인지 확인하십시오.

사용자 좌표계는 3점 캘리브레이션 방식으로 생성됩니다. 로봇을 A, B, C의 임의의 세 지점으로 이동합니다. 지점 A를 원점으로 정의합니다. 점 A에서 점 B까지의 선을 x축의 양의 방향으로 정의합니다. 점 C가 x축에 수직인 선을 y축의 양의 방향으로 정의합니다. (실제 교정에서 점 C는 y축에 있을 필요는 없지만, x축과 y축의 양의 방향이고 x축에 있으면 안 됩니다. 시스템은 y축의 양의 방향을 자동으로 계산할 수 있습니다.) z 축은 오른손 법칙에 따라 정의할 수 있습니다.



사용자 좌표계 추가

전제 조건

- 로봇 팔의 전원이 켜졌습니다.
- 로봇 팔이 활성화되었습니다.

절차

Home/Coor User

Coor User ⚙

+ Add ⌂ Cover ☰ Modify ✎ Delete

No	Alias	X	Y	Z	RX	RY	RZ
No.0		0	0	0	0	0	0
No.1		0	0	0	0	0	0
No.2		0	0	0	0	0	0

First Point ⚙ Run to

X: 0.0 RX: 0.0
Y: 0.0 RY: 0.0
Z: 0.0 RZ: 0.0

Second Point ⚙ Run to

X: 0.0 RX: 0.0
Y: 0.0 RY: 0.0
Z: 0.0 RZ: 0.0

Third point ⚙ Run to

X: 0.0 RX: 0.0
Y: 0.0 RY: 0.0
Z: 0.0 RZ: 0.0

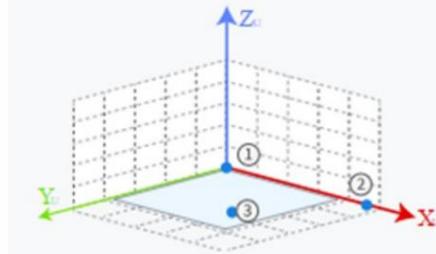
GET GET GET

메모

- 왼쪽 상단을 클릭하면 페이지의 지침을 볼 수 있습니다.
- 문서.
- 9개의 좌표계가 추가된 경우 추가 버튼이 사라지고 더 이상 좌표계를 추가할 수 없음을 나타냅니다.

1. 로봇을 점(좌표계의 원점)으로 조그합니다. 첫 번째 포인트 패널에서 GET을 클릭합니다 .
2. 로봇을 x축의 두 번째 지점으로 조그합니다. 두 번째 포인트 패널에서 GET을 클릭합니다 .
3. 로봇을 세 번째 점으로 이동합니다(점은 양수로 결정된 평면 사분면에 있어야 합니다.

x축과 y축의 방향이며 x축에 있지 않아야 합니다.). 세 번째 지점 패널에서 GET을 클릭합니다 .



4. 추가를 클릭합니다. 에서 새 레코드를 볼 수 있으므로 사용자 좌표계가 성공적으로 생성되었습니다.

페이지.

기타 작업

해당 좌표계의 번호 (0번 제외)를 클릭하면 좌표계를 선택할 수 있으며, 다른 열을 클릭하면 해당 파라미터의 값을 수정할 수 있습니다.

실행 대상: 실행 대상을 길게 눌러 로봇 팔을 획득한 지점으로 이동합니다.

Cover: 세 점의 좌표를 구한 후 좌표계를 선택하고 Cover를 클릭하면 이 좌표계가 새로 보정된 좌표계로 바뀝니다.

수정: 기존 좌표계의 좌표값을 수정한 후 좌표계를 선택하고 수정을 클릭하면 좌표값이 업데이트 됩니다.

삭제: 좌표계를 선택하고 삭제를 클릭하면 좌표계 값이 지워집니다.



- 삭제 작업은 레코드 자체를 삭제하는 것이 아니라 레코드 값을 지우는 것을 의미합니다.
이 좌표계를 재보정할 때 Cover 작업을 사용할 수 있습니다.
- 0번 좌표계는 삭제할 수 없습니다.

4.3 조그 설정



이 기능은 관리자 권한(기본 암호: 000000)이 필요합니다.

관절 좌표계와 직교 좌표계에서 최대 속도와 가속도를 설정할 수 있습니다. 매개변수를 설정한 후 저장을 클릭합니다 .

The screenshot shows the 'Teach Setting' screen with four main sections:

- Joint Velocity:** Contains fields for J1, J2, and J3 joints with values of 12 °/s.
- Joint Acceleration:** Contains fields for J1, J2, and J3 joints with values of 100 °/s².
- Cartesian Velocity:** Contains fields for X, Y, and Z axes with values of 50 mm/s.
- Cartesian Acceleration:** Contains fields for X, Y, and Z axes with values of 300 mm/s².

Each section has a 'Default' button at the bottom right. A 'Save' button is located in the top right corner of the screen.

실제 로봇 속도/가속도 = 설정 속도/가속도 × 전역 속도 × 비율.

해당 모듈의 모든 값을 기본값으로 복원하려면 기본값을 클릭 합니다 .

4.4 재생 설정



이 기능은 관리자 권한(기본 암호: 000000)이 필요합니다.

관절 좌표계와 직교 좌표계에서 속도, 가속도 및 저크 매개변수를 설정할 수 있습니다. 설정 후 저장을 클릭합니다.

Section	Joint	Value	Unit	Joint	Value	Unit
Joint Velocity	J1	180	°/s	J4	180	°/s
	J2	180	°/s	J5	180	°/s
	J3	180	°/s	J6	180	°/s
Default						
Joint Acceleration	J1	200	°/s ²	J4	500	°/s ²
	J2	200	°/s ²	J5	500	°/s ²
	J3	200	°/s ²	J6	500	°/s ²
Default						
Joint Jerk	J1	2000	°/s ³	J4	5000	°/s ³
	J2	2000	°/s ³	J5	5000	°/s ³
	J3	2000	°/s ³	J6	5000	°/s ³
Default						

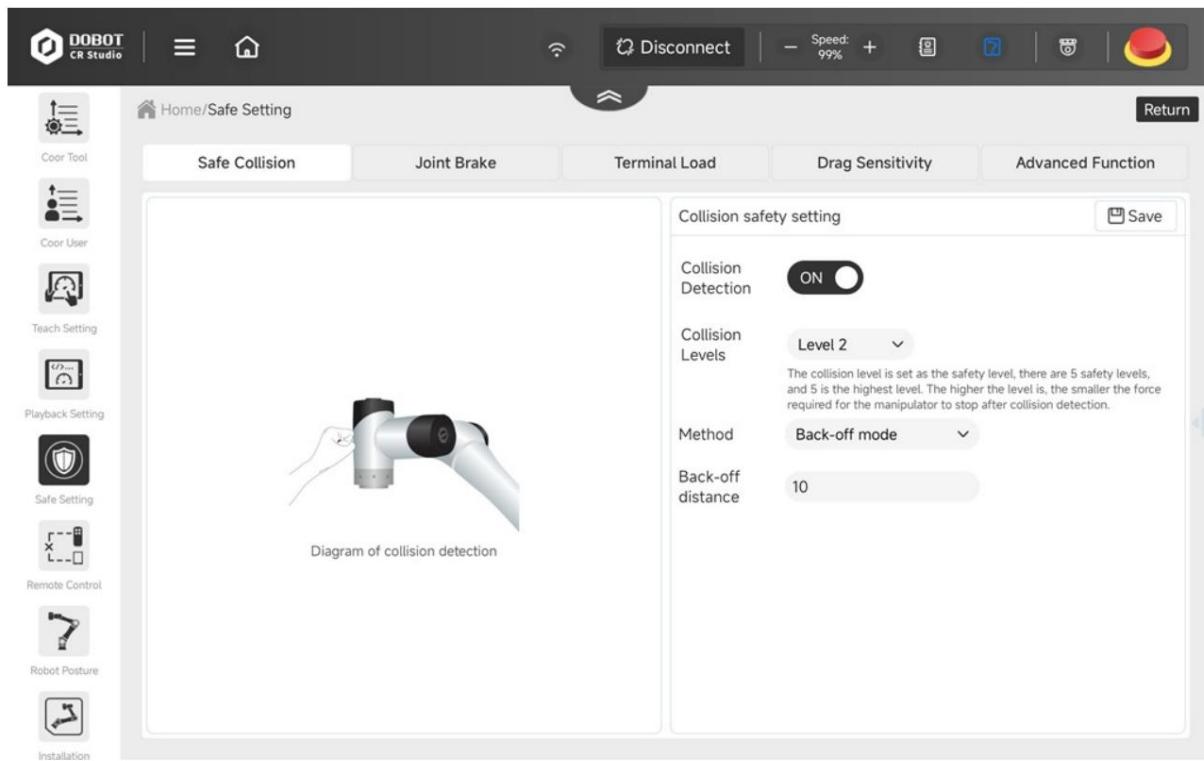
실제 로봇 속도/가속 = 설정 속도/가속 × 전역 속도 비율 × 프로그래밍 시 속도 명령에서 설정된 백분율.

해당 모듈의 모든 값을 기본값으로 복원하려면 기본값을 클릭합니다.

4.5 안전 설정

4.5.1 안전한 충돌

충돌 감지는 주로 로봇 또는 외부 장비의 손상을 방지하기 위해 로봇에 대한 충격을 줄이는 데 사용됩니다. 충돌 감지가 활성화되면 장애물에 부딪히면 로봇 팔이 자동으로 실행을 멈춥니다.

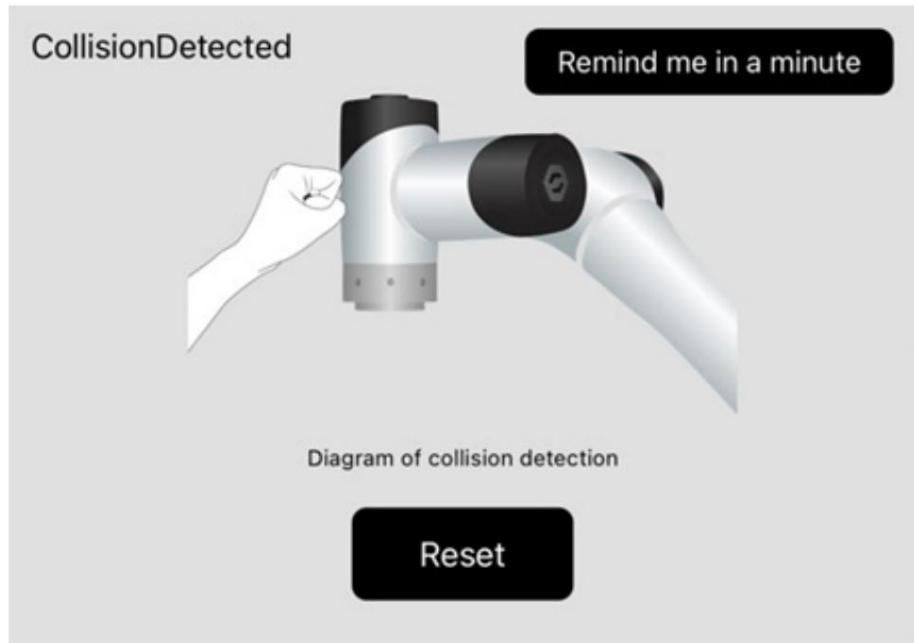


충돌 감지를 클릭하여 충돌 감지 기능을 활성화 또는 비활성화하고 충돌 수준에서 충돌 감지 민감도를 설정할 수 있습니다. 선택할 수 있는 5가지 수준이 있습니다. 더 높은 수준을 선택할수록 충돌 감지 후 로봇이 정지하는 데 필요한 힘이 작아집니다.

방법은 로봇 팔이 프로젝트를 실행하는 동안 충돌 후 처리를 나타냅니다.

- 중지: 로봇 팔이 프로젝트 실행을 중지합니다.
- 일시 중지: 로봇 팔이 일시 중지됩니다. 실제 상황에 따라 충돌 원인을 해결한 후 작업을 재개할지 아니면 작업을 중지 할지 선택해야 합니다.
- 끌기 모드: 로봇 팔이 프로젝트 실행을 종지하고 자동으로 끌기 모드로 들어갑니다.
- 백오프 모드: 로봇 팔은 충돌 전 궤적에 따라 지정된 거리만큼 자동으로 후퇴합니다. 백오프 거리의 범위는 0~50(mm)입니다.

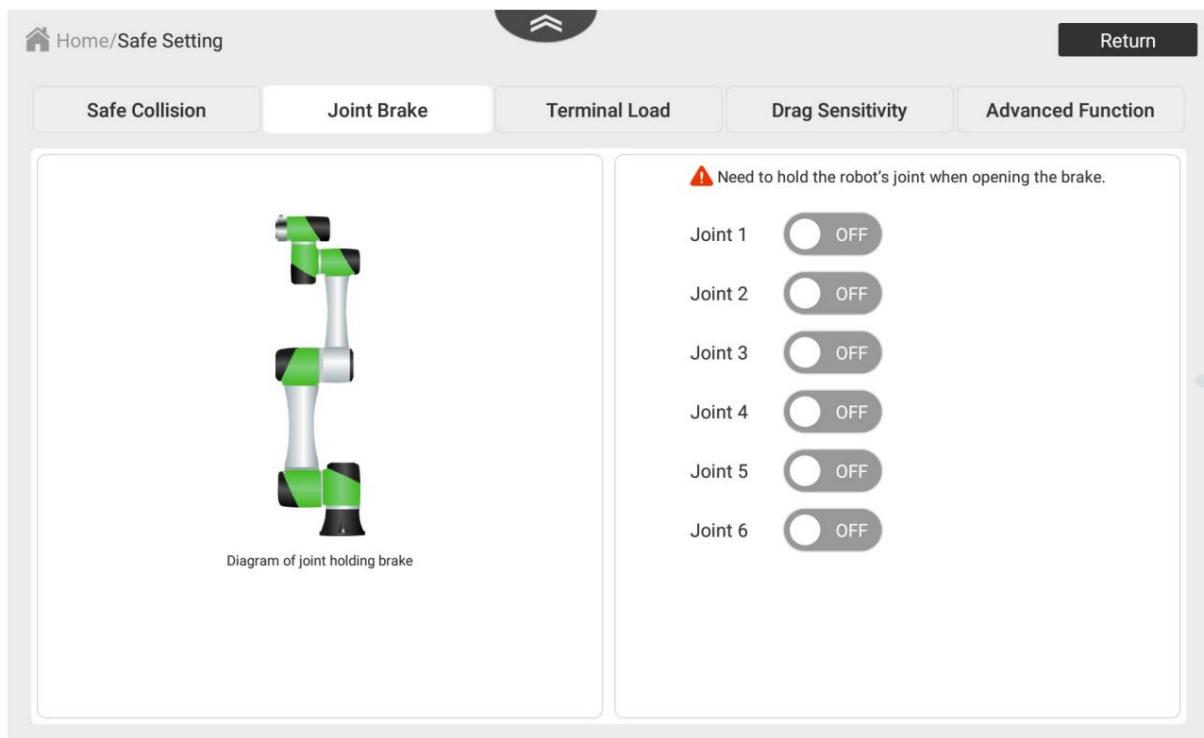
로봇을 조깅할 때 정지에 필요한 힘이 감지되면 "충돌 감지됨" 창이 팝업됩니다. 이 경우 충돌 원인을 해결하고 재설정을 클릭해야 합니다. 충돌 원인을 해결하기 위해 앱을 작동해야 하는 경우 잠시 후 알림을 클릭하여 팝업 창을 일시적으로 닫습니다(1분 후 팝업 메시지가 다시 표시됨).



수정 사항은 저장을 클릭한 후에 적용됩니다 .

4.5.2 조인트 브레이크

제동은 서보 드라이버가 작동하지 않을 때 서보 모터 축이 움직이는 것을 방지하여 모터가 위치를 고정하고 자중이나 외력으로 인해 기계의 움직이는 부분이 움직이지 않도록 합니다.



관절을 수동으로 드래그하려면 브레이크 기능을 활성화하십시오. 즉, 로봇 팔이 비활성화된 후 수동으로 관절을 잡고 해당 관절의 버튼을 클릭하십시오.



알아채다

기능을 활성화할 때 손으로 관절을 잡고 움직이지 않도록 하십시오.

4.5.3 종단부하

최적의 로봇 성능을 보장하려면 엔드 이펙터의 부하 및 편심 좌표가 로봇의 최대 범위 내에 있고 관절 6이 편심되지 않는지 확인하는 것이 중요합니다.

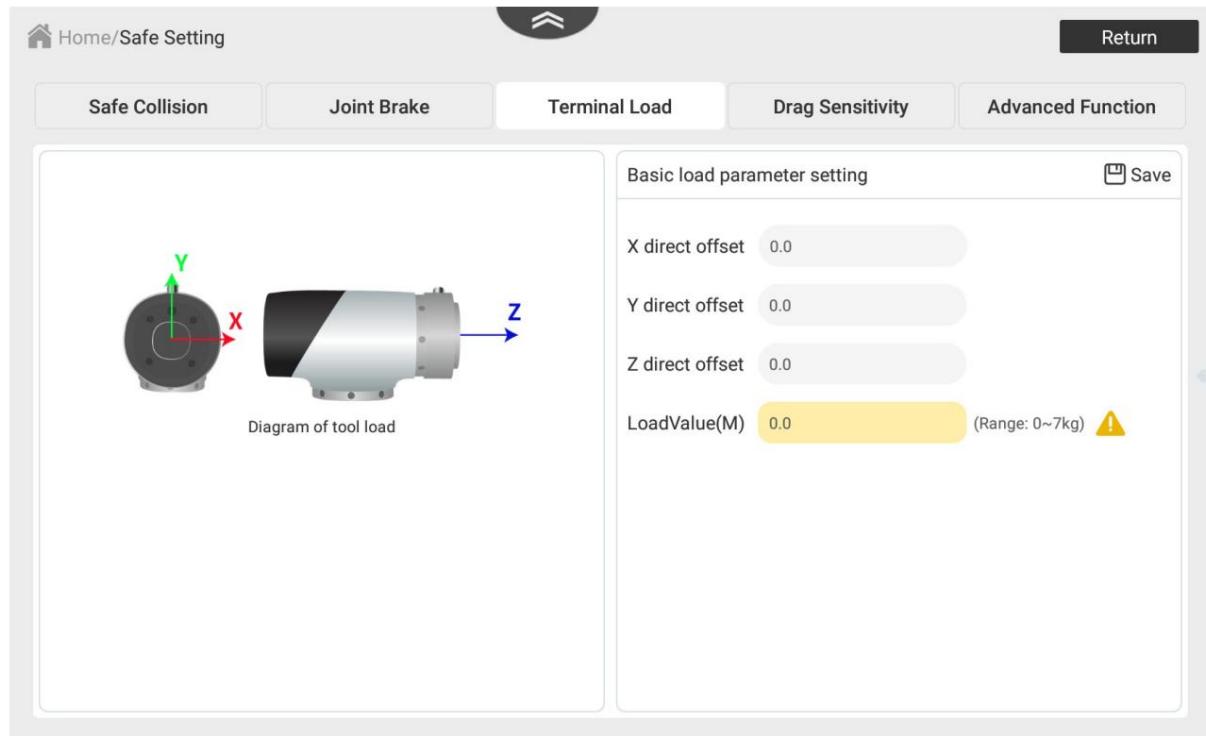
하중과 편심 좌표를 적절하게 설정하면 로봇의 움직임이 좋아지고 진동이 줄고 작업 시간이 단축됩니다.



메모

CR 로봇을 활성화할 때마다 부하 매개변수를 설정해야 하는 "전원 상태 변경" 창이 팝업됩니다. 설정한 매개변수는 "터미널 로드"에 동기화됩니다.

페이지.



- J6 축이 0°일 때 하중의 편심 좌표를 설정해야 합니다.
- 부하 중량에는 엔드 이펙터 및 작업물의 중량이 포함되며 로봇 암의 최대 부하를 초과해서는 안 됩니다. 적재 중량을 0으로 설정하면 노란색 배경에 느낌표가 표시됩니다. 표시는 활성화 아이콘의 오른쪽 하단에도 나타납니다.

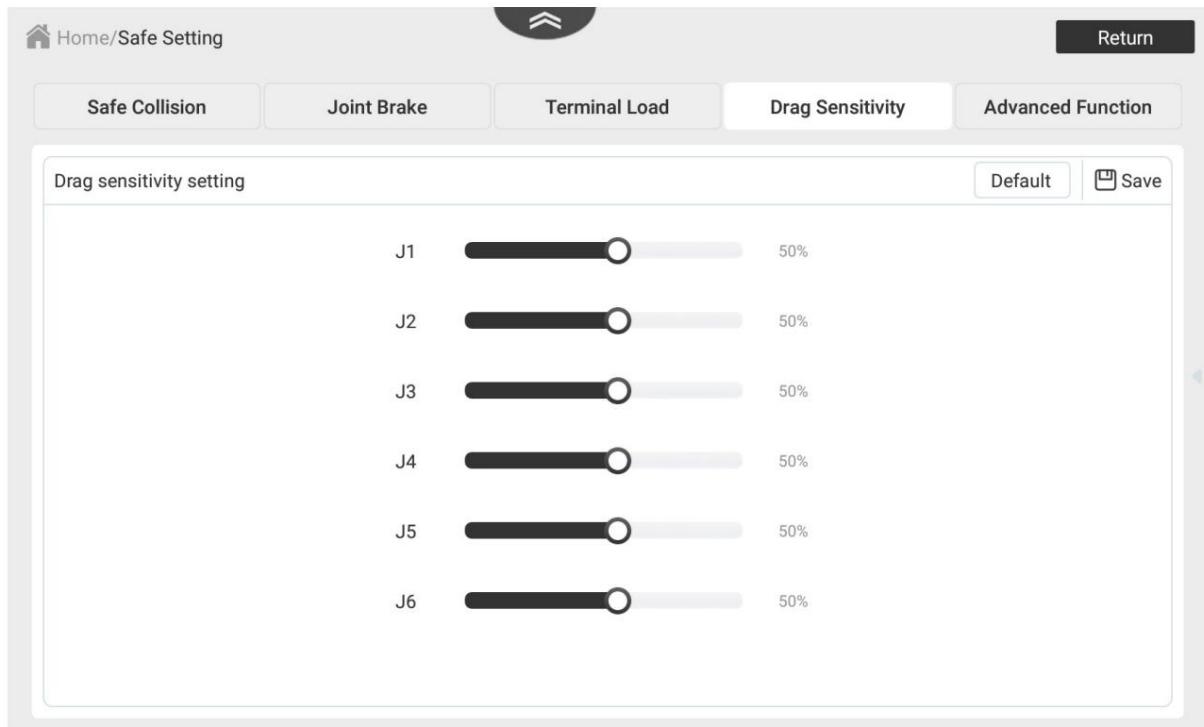
로봇이 활성화된 후.

저장을 클릭하면 설정이 적용됩니다 .

하중과 편심 좌표를 적절하게 설정하십시오. 그렇지 않으면 오류나 과도한 충격이 발생하고 부품의 수명이 단축될 수 있습니다.

4.5.4 드래그 감도

끌기 감도 설정은 주로 끌기 중 관절의 감도를 조정하는 데 사용됩니다.



감도가 낮을수록 드래그하는 동안 저항이 커집니다.

모든 관절의 민감도를 기본값으로 복원하려면 기본값 을 클릭합니다 .

민감도를 설정한 후 저장을 클릭합니다 .

4.5.5 고급 기능



메모

이 기능은 관리자 권한(기본 암호: 000000)이 필요합니다.

사이트에 따라 고급 기능을 활성화 또는 비활성화해야 할 때 이 페이지에서 설정할 수 있습니다.
상태.

The screenshot shows a software interface titled "Home/Safe Setting". At the top, there are tabs for "Safe Collision", "Joint Brake", "Terminal Load", "Drag Sensitivity", and "Advanced Function". The "Advanced Function" tab is currently selected. In the center, there is a "Tip" section with the following text:
TrueMotion: When on, the transition trajectory between Cartesian commands remains consistent at different speeds.
DynamicsOptimal: When on, the preset torque value of joint / mechanical mechanism will be used as the constraint reference value of acceleration.
Compensation: When on, it can effectively improve the TCP accuracy of the terminal, especially in the case of large change in posture.
Below the tip, there are three toggle switches:

- TrueMotion: OFF (button is grey)
- DynamicsOptimal: ON (button is black)
- Compensation: ON (button is black)

특별한 요구 사항이 없는 경우 기본 설정을 유지하는 것이 좋습니다.

4.6 원격 제어

외부 장비는 원격 I/O 모드 및 원격 Modbus 모드와 같은 다양한 원격 제어 모드에서 로봇에 명령을 보낼 수 있습니다(교시된 프로그램 파일 제어 및 실행). 앱을 통해 원격 모드를 설정할 수 있습니다.



메모

- 원격 제어 모드를 전환할 때 로봇 제어 시스템을 다시 시작할 필요가 없습니다.
- 로봇 제어 시스템이 어떤 모드에 있든 비상 정지 스위치는 항상 켜져 있습니다.
효과적인.
- 로봇이 원격 제어 모드에서 실행 중인 경우 다른 작업 모드로 전환할 수 없습니다. 다른 모드로 전환하기 전에 실행을 중지해야 합니다.
- 복원을 클릭하면 인터페이스가 현재 작업 모드로 자동 전환됩니다.

앱-온라인

기본 모드는 App-online 모드이며 앱을 사용하여 로봇을 제어할 수 있습니다.

Modbus-오프라인

외부 장비는 Modbus 오프라인 모드에서 로봇 팔을 제어할 수 있습니다.

The screenshot shows the 'Remote Control' section of the app. At the top, there are five radio button options: 'Current working mode' (selected), 'App-Online', 'Modbus-Offline' (selected), 'IO-Offline', and 'TCP/IP secondary development'. Below these are two dropdown menus: 'Select offline project' set to 'blockly_ARC_2' and 'Advanced Setting'. The main area is divided into two sections: 'Coil register configuration information' and 'Contact register configuration information'. Each section contains a table with rows for various parameters and their values.

	Start	Suspend
Resume	2	Stop
Clear Alarms	5	Power on
Trigger mode	Rising edge	

	Stop status	Suspend status
Alarm status	4	Running status
Warning status	7	Power status
Remote status	9	

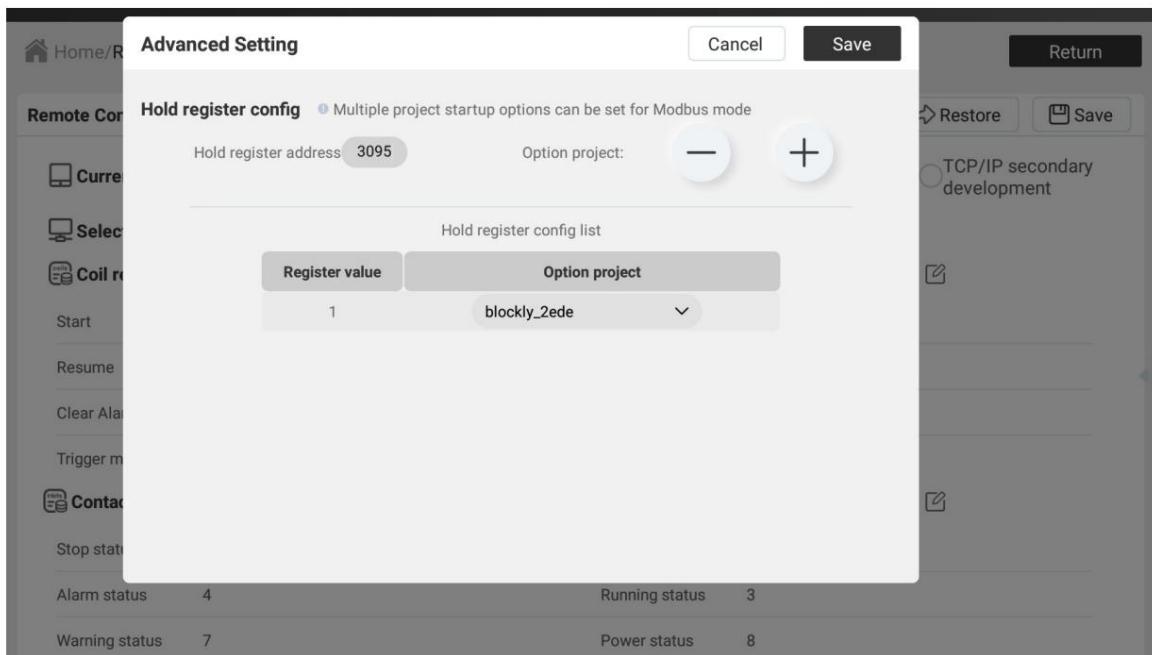
Modbus 레지스터의 기능은 위에 나와 있습니다. Modbus-오프라인 모드에서 프로젝트를 실행하는 절차는 아래와 같습니다.

전제 조건

- 원격 모드로 실행할 프로젝트가 준비되었습니다.
- 로봇은 LAN 인터페이스를 통해 외부 장비에 연결되었습니다. 직접 또는 라우터를 통해 연결할 수 있습니다. CR 로봇과 외부 장비의 IP 주소는 충돌 없이 동일한 네트워크 세그먼트 내에 있어야 합니다.
- 로봇 팔의 전원이 켜졌습니다.

절차

- 원격 제어 페이지에서 Modbus-오프라인을 클릭하고 실행할 오프라인 프로젝트를 선택합니다.
- Modbus를 통해 여러 개의 서로 다른 프로젝트(최대 10개 프로젝트)를 시작해야 하는 경우 고급 설정을 클릭합니다. 고급 설정에서는 다음 그림과 같이 옵션 프로젝트의 홀드 레지스터 주소를 설정하고 옵션 프로젝트 목록을 구성할 수 있습니다.



- 저장을 클릭합니다. 이제 로봇 팔이 원격 Modbus 모드에 들어갔습니다. 비상 정지 명령만 사용할 수 있습니다. 동시에 The Debug log가 인터페이스의 오른쪽 상단에 나타납니다. 클릭하면 로봇팔 작동 중 출력되는 디버그 정보를 볼 수 있습니다.

Debug log

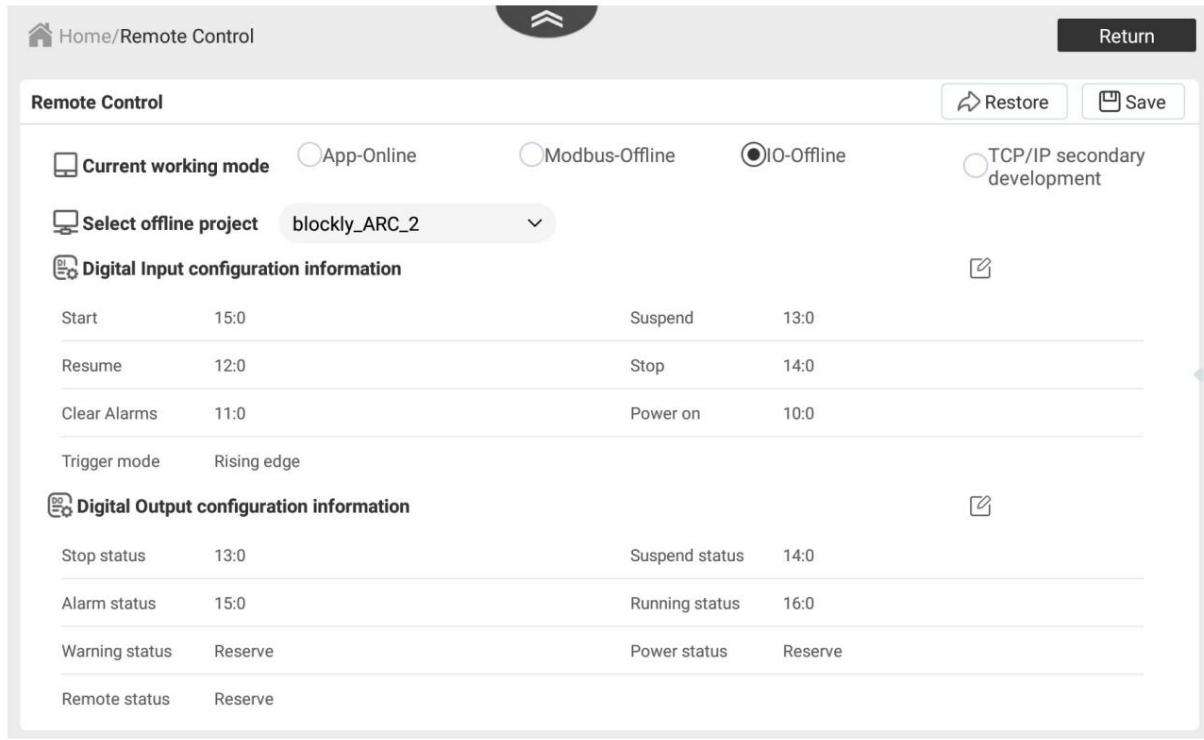
Restore

Save

- 외부 장비에서 시작 신호를 트리거합니다. 선택한 프로젝트 파일로 로봇이 이동합니다.
- 정지 신호가 트리거되면 로봇 암이 이동을 멈춥니다.

IO-오프라인

외부 장비는 IO-오프라인 모드에서 로봇 팔을 제어할 수 있습니다.



The screenshot shows the 'Remote Control' section of a software interface. At the top, there are five radio button options: 'Current working mode' (selected), 'App-Online', 'Modbus-Offline', 'IO-Offline' (selected), and 'TCP/IP secondary development'. Below this, a dropdown menu labeled 'Select offline project' shows 'blockly_ARC_2'. There are two main configuration sections: 'Digital Input configuration information' and 'Digital Output configuration information', each with several rows of data and edit icons.

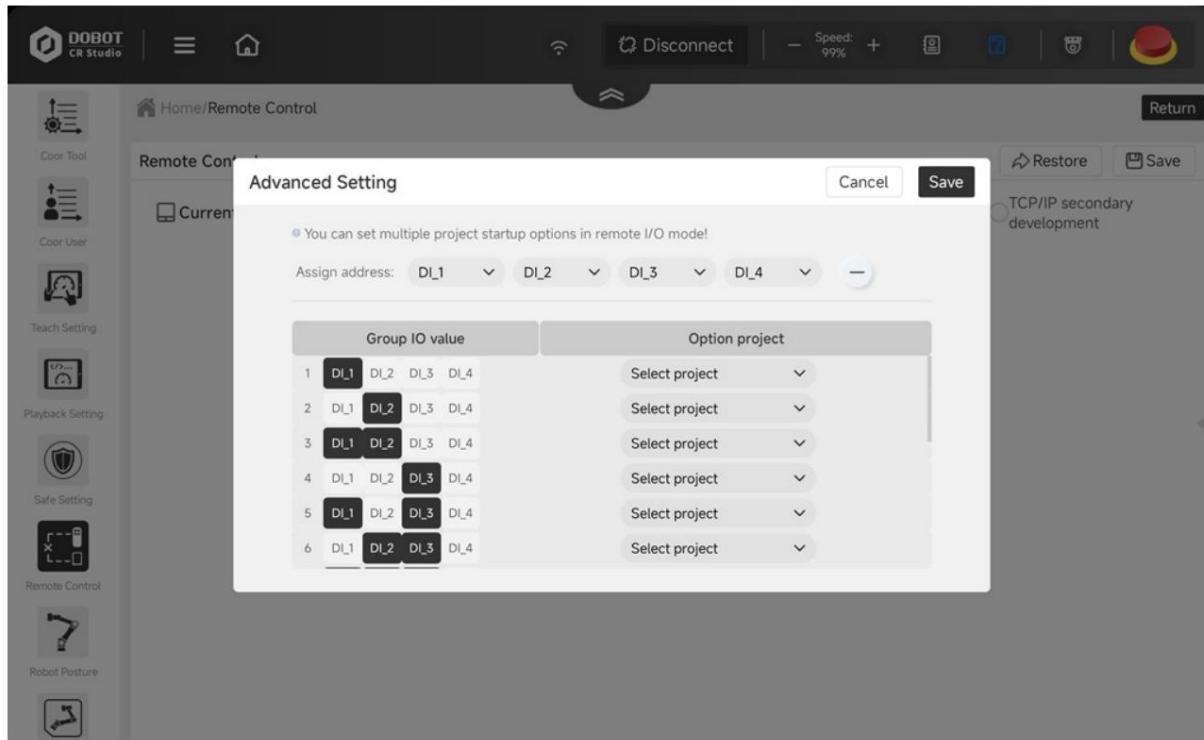
Start	15:0	Suspend	13:0
Resume	12:0	Stop	14:0
Clear Alarms	11:0	Power on	10:0
Trigger mode	Rising edge		

Stop status	13:0	Suspend status	14:0
Alarm status	15:0	Running status	16:0
Warning status	Reserve	Power status	Reserve
Remote status	Reserve		

제어 시스템의 특정 IO 인터페이스 정의는 위 그림에 나와 있습니다. 클릭하여 편집할 수 있습니다. 제어 캐비닛의 유형이 CCBOX인 경우 Safe I/O와 Universal I/O는 동일한 단자를 공유합니다.

[안전 I/O](#)로 구성된 터미널은 원격 I/O로 구성할 수 없습니다.

고급 설정을 클릭하면 그룹 I/O를 통해 여러 프로젝트를 구성할 수 있습니다.



1. + 또는 -를 클릭하여 그룹 I/O에 할당된 주소 수를 늘리거나 줄입니다. 많을수록

할당된 주소가 많을수록 구성 가능한 프로젝트가 더 많아집니다. 0 주

소: 선택적 프로젝트를 구성할 수 없습니다. 1개의 주소: 1개

의 선택적 프로젝트를 구성할 수 있습니다. 2개의 주소:

3개의 선택적 프로젝트를 구성할 수 있습니다. 3개의 주소: 7

개의 선택적 프로젝트를 구성할 수 있습니다. 4개의 주소: 15

개의 선택적 프로젝트를 구성할 수 있습니다.

2. 드롭다운 목록을 통해 할당된 주소를 수정할 수 있습니다. 그룹 I/O에 할당된 주소

원격 I/O 또는 안전 I/O(CCBOX)와 복제할 수 없습니다.

3. 주소를 할당한 후 각 그룹 IO 값에 대해 선택적 프로젝트(최소 하나)를 설정할 수 있습니다.

4. 리모트 I/O 모드에서 프로젝트를 실행하기 전에 설정을 통해 해당 옵션 프로젝트를 선택합니다.

해당(검은색은 ON, 흰색은 OFF를 나타냄) 그룹 IO 값.

5. 위 그림에서 DI1~DI4 할당을 예로 들어 보겠습니다.

- DI1은 ON, DI2, DI3 및 DI4는 OFF: 첫 번째 선택적 프로젝트를 선택합니다. DI1

- 및 DI2는 ON, DI3 및 DI4는 OFF: 세 번째 선택적 프로젝트를 선택합니다. DI1~DI4 모두

- ON: 15번째 옵션 프로젝트를 선택합니다.

6. 모든 그룹 IO가 OFF로 설정되면 이전에 구성한 메인 프로젝트를 선택한다는 의미입니다.

페이지.

7. 저장을 클릭하여 구성성을 완료합니다.

IO-오프라인 모드에서 프로젝트를 실행하는 절차는 다음과 같습니다.

전제 조건

- 원격 모드로 실행할 프로젝트가 준비되었습니다.
- 외부 장비는 I/O 인터페이스를 통해 로봇 팔에 연결되었습니다.

- 로봇 팔의 전원이 켜졌습니다.

절차

1. 원격 제어 페이지에서 IO-오프라인을 클릭하고 실행할 오프라인 프로젝트를 선택합니다.

2. 저장을 클릭합니다. 이제 로봇 팔이 원격 IO 모드에 들어갔습니다. 비상 정지 명령만

사용 가능. 동시에 디버그 로그가 인터페이스의 오른쪽 상단에 나타납니다. 클릭하면 로봇팔 작동 중 출력되는 디버그 정보를 볼 수 있습니다.

Debug log

Restore

Save

3. 외부 장비에서 시작 신호를 트리거합니다. CR 로봇이 선택한 프로젝트로 이동합니다.

파일.

4. 정지 신호가 트리거되면 로봇 팔이 이동을 멈춥니다.

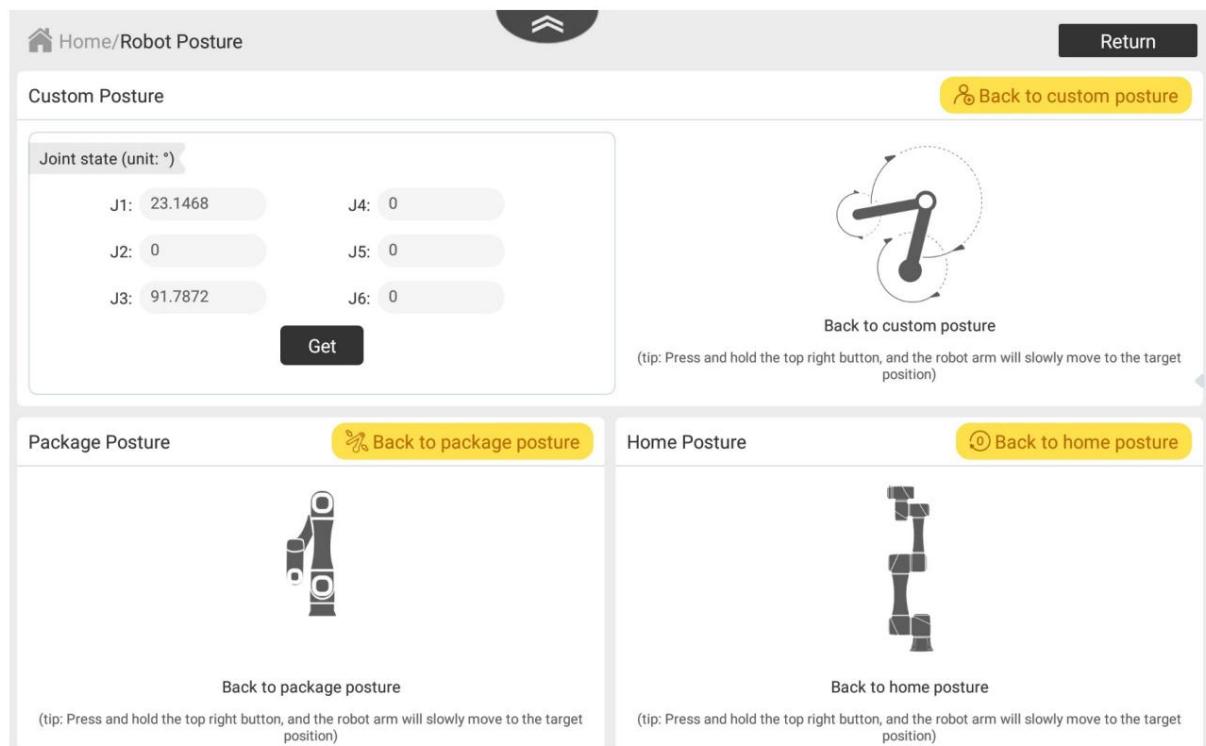
TCP/IP 보조 개발

이 모드는 사용자가 TCP 기반의 제어 소프트웨어를 개발하기 위한 것입니다. 소프트웨어를 개발해야 하는 경우 [두봇 TCP/IP 프로토콜을](#) 참조하십시오. (Github에 배치).

4.7 로봇 자세

CR Studio는 CR 로봇을 일반적인 자세(패키지 자세, 홈 자세 및 맞춤형 자세)로 이동하는 것을 지원합니다. 자세.

- 로봇을 포장자세로 이동시키면 로봇의 공간을 줄일 수 있어 포장이 간편하고 수송.
- 홈 자세는 홈 포인트를 말합니다.
- 맞춤형 자세는 요구 사항에 따라 정의할 수 있으므로 로봇을 해당 위치로 이동하는 것이 편리합니다.



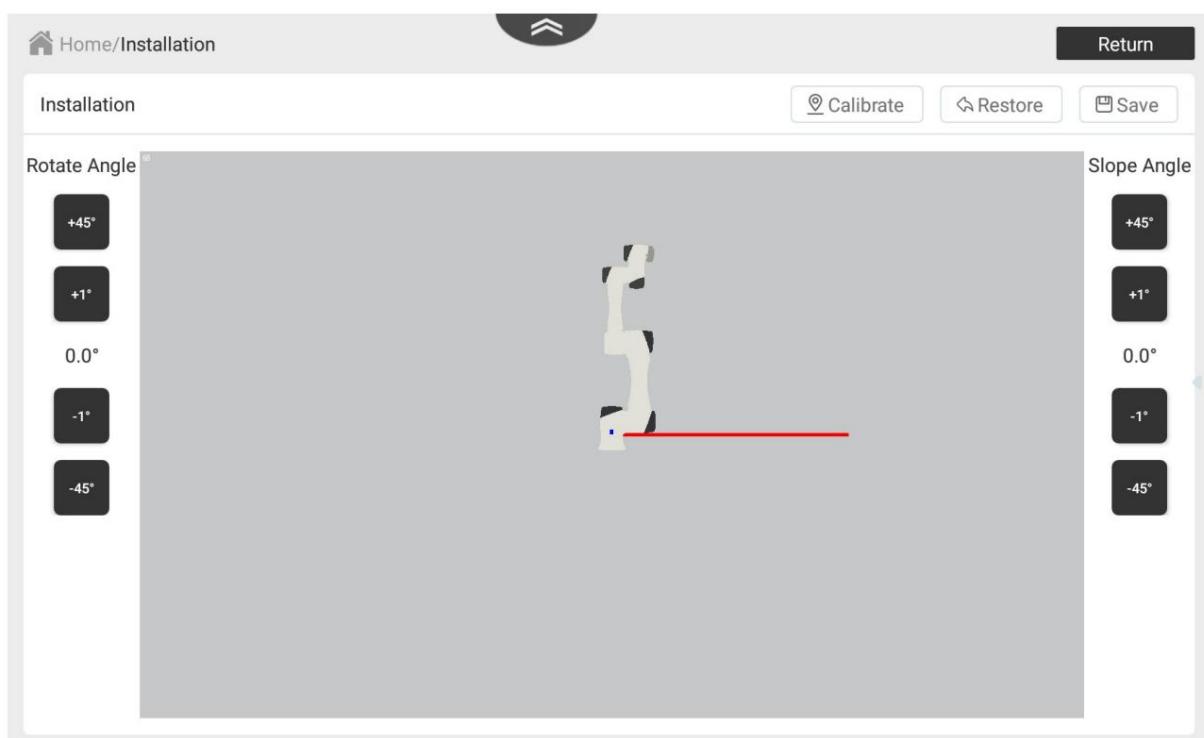
- 패키지 자세: 로봇을 공장 자세로 이동하려면 패키지 자세로 돌아가기를 길게 누릅니다.
- 홈 자세: 로봇을 홈 포인트로 이동하려면 홈 자세로 돌아가 려면 길게 누르십시오.
- 사용자 지정 자세: 로봇을 특정 위치로 이동하고 GET을 클릭하면 사용자 지정 자세가 저장됩니다. 사용자 지정 자세로 돌아가기를 길게 눌러 로봇을 정의된 자세로 이동합니다.

4.8 설치



이 기능은 관리자 권한(기본 암호: 000000)이 필요합니다.

일반적으로 CR 로봇은 안정된 테이블이나 지면에 설치되며 이 경우 별도의 조작이 필요하지 않습니다. 로봇이 천장에 설치되거나 벽에 설치되거나 특정 각도로 설치된 경우 비활성화 상태에서 회전 각도와 기울기 각도를 설정해야 합니다.



로봇을 설치하고 활성화한 후 보정을 클릭하고 팝업 상자에 따라 작동하여 기울기 각도와 회전 각도를 얻습니다.

- **기울기 각도:** 로봇이 원점 위치에서 x축을 기준으로 반시계 방향으로 회전하는 각도입니다.
- **회전 각도:** 로봇이 원점 위치에서 z축을 중심으로 반시계 방향으로 회전하는 각도.

Calibration tip

Please adjust the position of the robot through the joint motion instructions so that the end flange is perpendicular to the ground, and then click the calibrate installation angle button

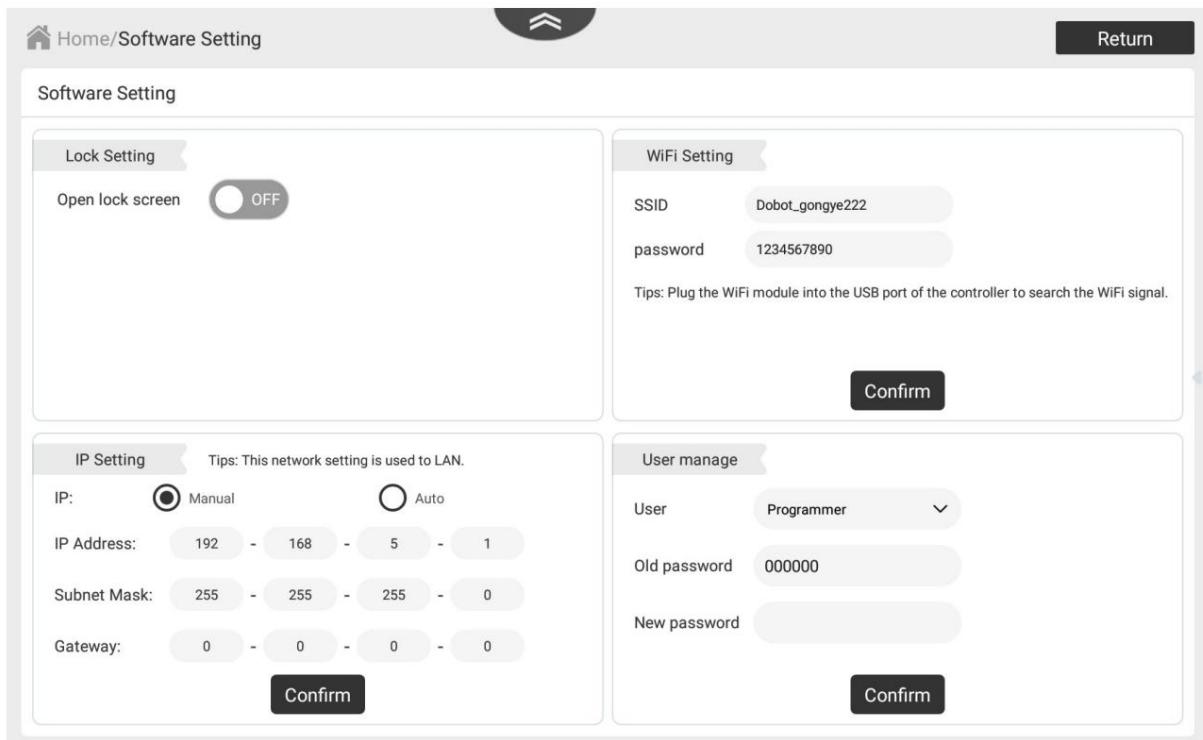
Calibrate installation angle

CANCEL

보정 후 저장을 클릭하여 설정을 저장합니다.

보정된 각도를 기본값으로 복원하려면 복원을 클릭합니다 .

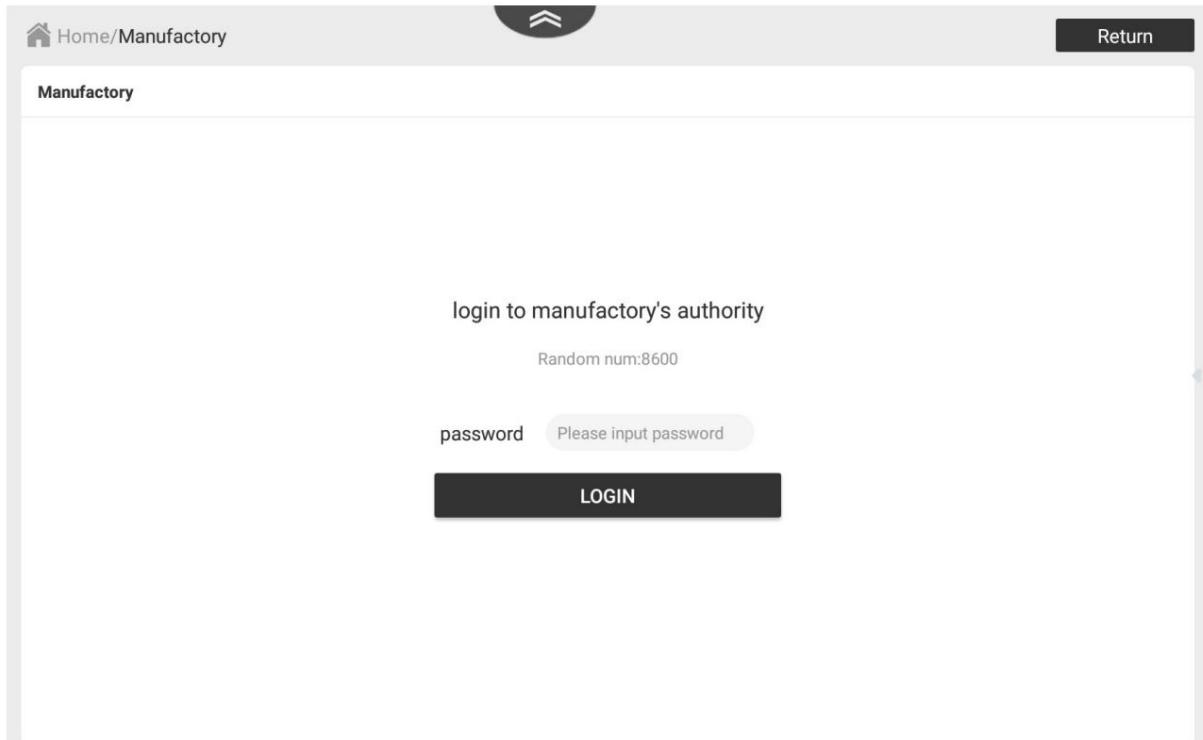
4.9 소프트웨어 설정



- 화면 잠금 설정: 앱의 화면 잠금 시간을 설정하고 필요에 따라 화면 잠금 암호를 변경할 수 있습니다. 지정된 시간 동안 조작하지 않으면 화면이 자동으로 잠깁니다.
관리자 비밀번호 또는 화면 잠금 비밀번호를 사용하여 화면 잠금을 해제할 수 있습니다. 기본 암호는 000000입니다.
- WiFi 설정: 로봇은 WiFi로 외부 장비와 통신할 수 있습니다. "WiFi 설정" 패널에서 WiFi 이름과 암호를 수정한 다음 컨트롤러를 다시 시작하여 적용할 수 있습니다.
기본 암호는 1234567890입니다.
- IP 설정:
로봇은 TCP, UDP 및 Modbus 프로토콜을 지원하는 이더넷 인터페이스를 통해 외부 장비와 통신할 수 있습니다. "IP 설정" 패널에서 IP 주소를 수정할 수 있습니다. CR 로봇의 IP 주소는 충돌 없이 외부 장비와 동일한 네트워크 세그먼트 내에 있어야 합니다.
 - 로봇이 직접 또는 스위치를 통해 외부 장치에 연결되어 있는 경우 수동을 선택하고 IP 주소, 서브넷 마스크 및 기본 게이트웨이를 변경합니다. 그런 다음 확인을 클릭합니다.
 - 로봇이 라우터를 통해 외부 장치에 연결된 경우 자동을 선택하면 IP 주소가 자동으로 할당됩니다. 그런 다음 확인을 클릭합니다.
- 사용자 관리: "사용자 관리" 패널에서 사용자 비밀번호를 수정할 수 있습니다. 기본 비밀번호는 000000입니다.

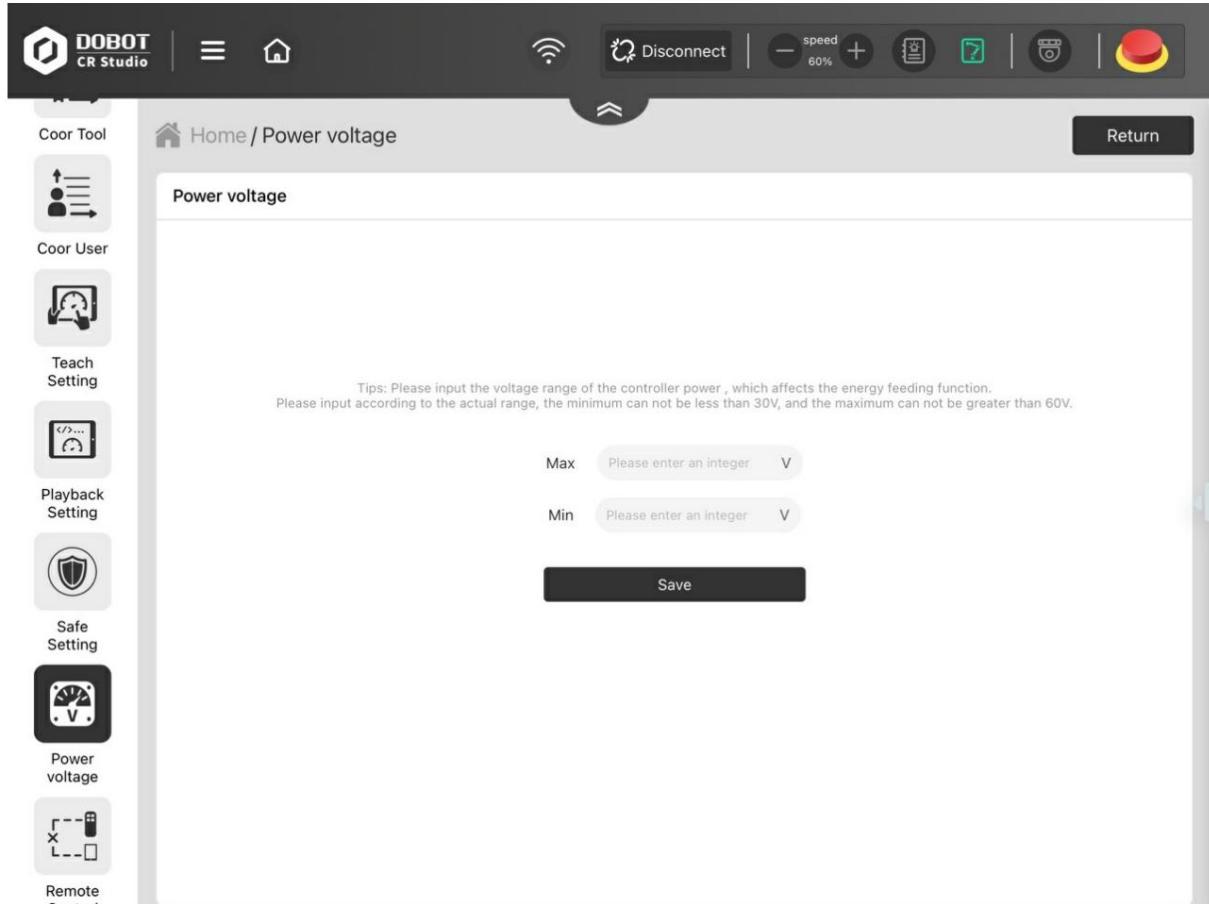
4.10 제조소

이 모듈의 기능은 일반적으로 두봇 기술 지원 또는 공장에서 제공합니다. 이 섹션에서는 모듈에 대한 자세한 설명을 제공하지 않습니다.



4.11 전원 전압(CCBOX)

컨트롤러의 종류가 CCBOX인 경우 컨트롤러의 입력 전압 범위를 설정해야 합니다.



입력 전압의 실제 범위에 따라 설정하십시오. 최대 전압은 60V 이하, 최소 전압은 30V 이상입니다.

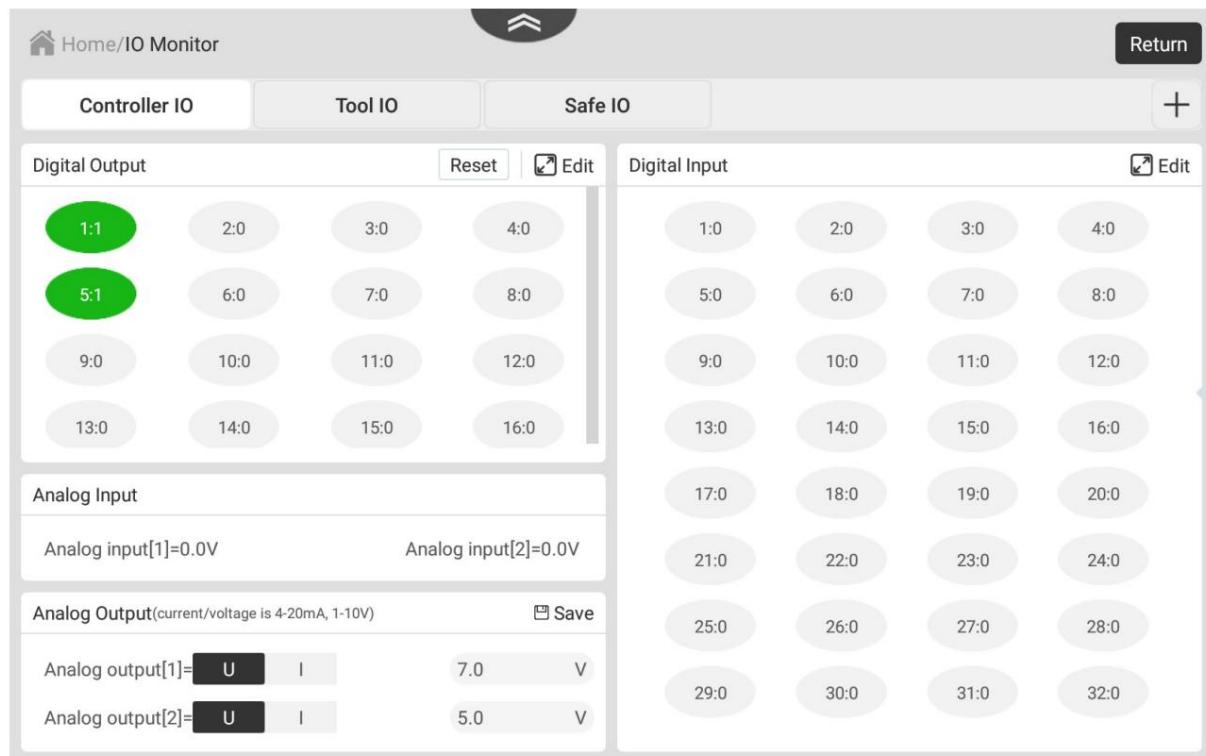
5 모니터링

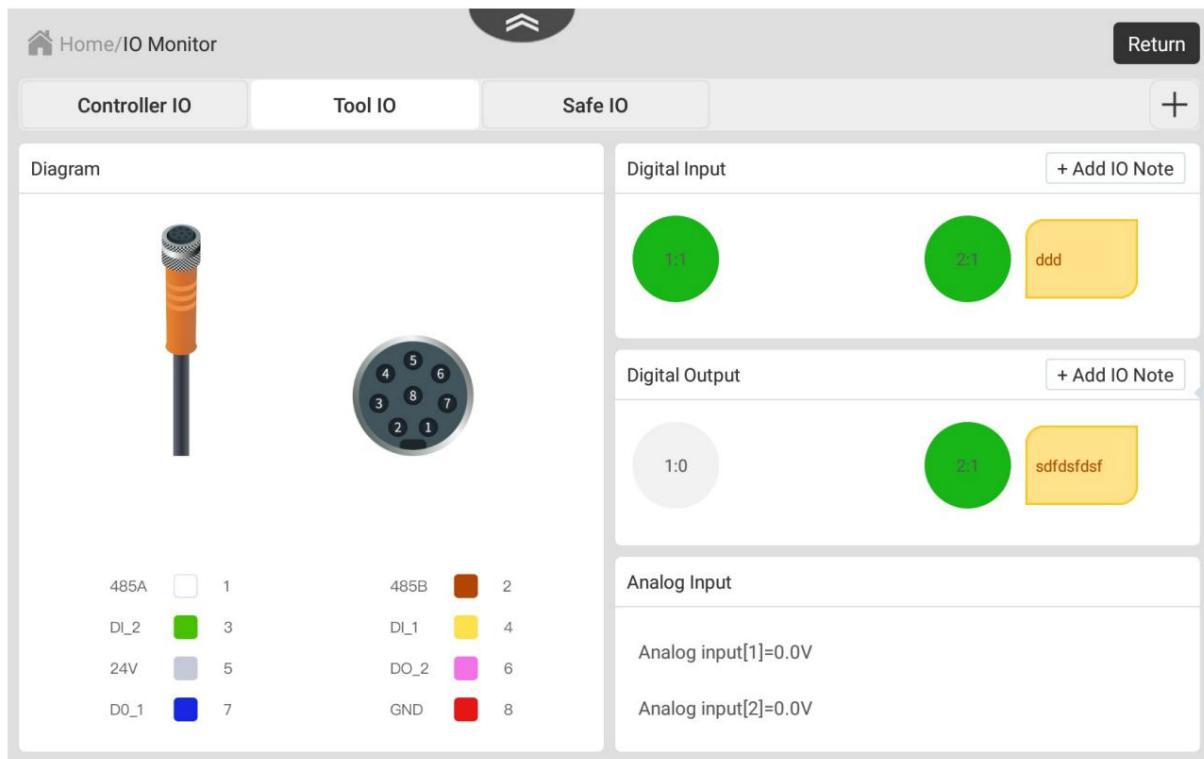
- 5.1 입출력 모니터
- 5.2 모드버스
- 5.3 글로벌 변수
- 5.4 로봇 상태
- 5.5 실행 로그
- 5.6 두봇+

5.1 입출력 모니터

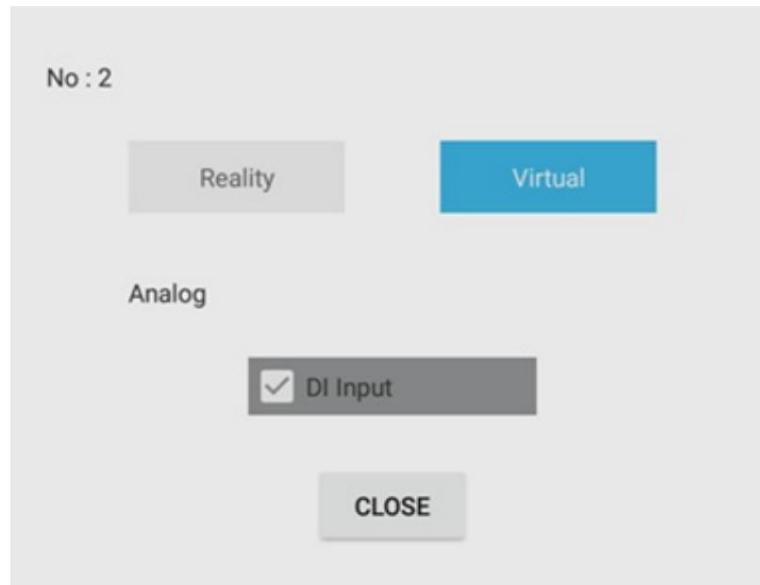
컨트롤러 및 도구 IO

앱에서 컨트롤러 및 종료 도구의 I/O 상태를 설정할 수 있습니다. I/O 정의는 해당 CR 로봇 하드웨어 가이드의 IO 설명을 참조하십시오. 이제 CR 로봇을 예로 들어 I/O 페이지를 소개합니다.

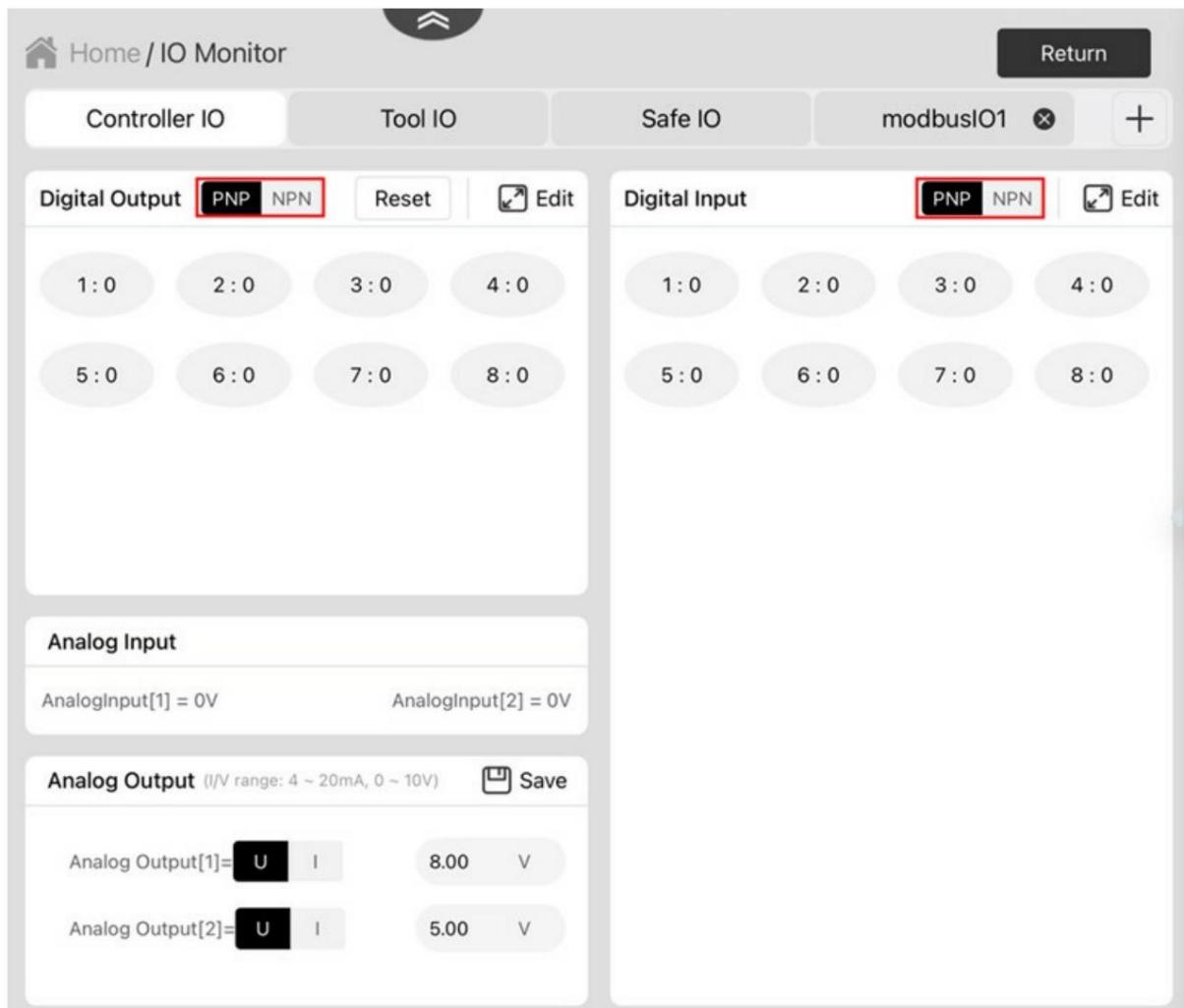




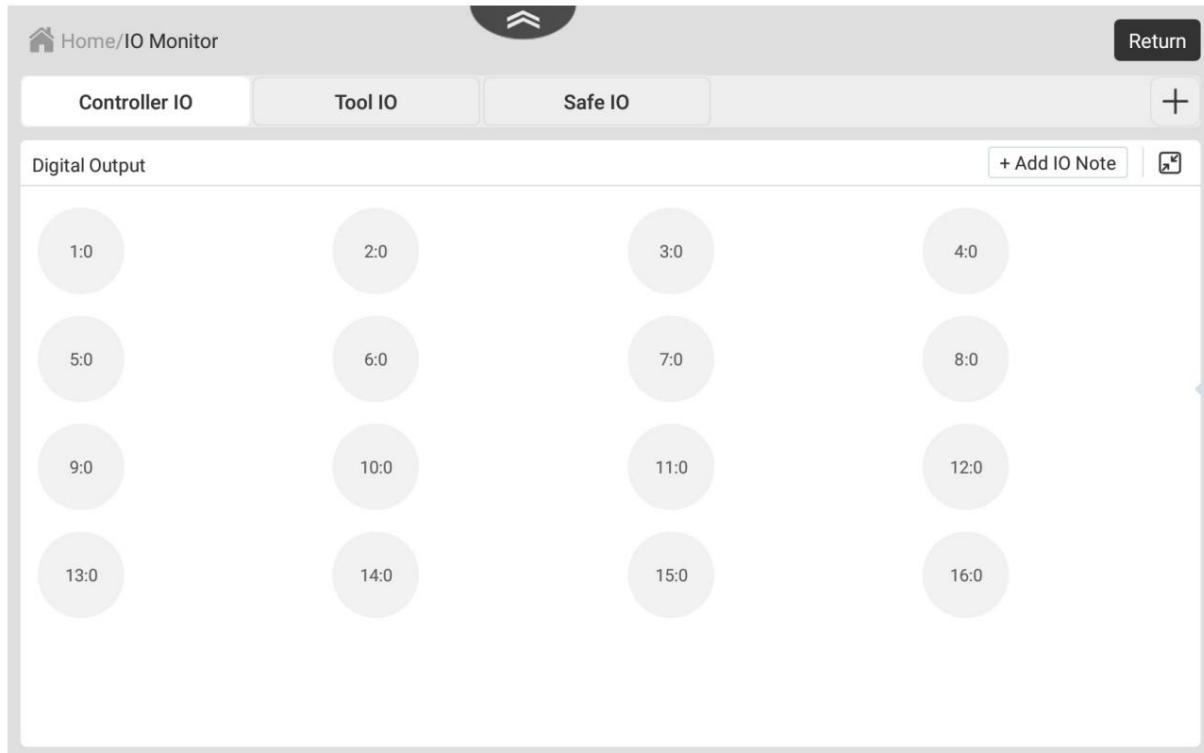
I/O 모니터에는 출력, 모니터 및 시뮬레이션의 세 가지 기능이 있습니다. - 출력: 디지털 출력 또는 아날로그 출력을 설정합니다. (아날로그 출력은 설정이 유지되어야만 유효합니다.) Digital Output 패널에서 Reset을 클릭하면 수정된 출력 값을 모두 초기화할 수 있습니다. - 모니터: 입력 및 출력 상태를 모니터링합니다. - 시뮬레이션: 아래와 같이 프로그램 디버깅을 위한 디지털 입력 상태를 시뮬레이션합니다.



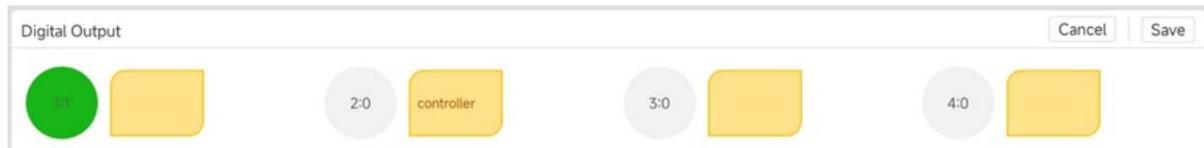
제어 캐비닛 유형이 CCBOX인 경우 아래와 같이 디지털 입력/출력 유형을 PNP(상한 수준 유효) 또는 NPN(하한 수준 유효)로 설정 할 수 있습니다.



Digital Output 또는 Digital Input 모듈 우측 상단의 Edit를 클릭하면 해당 IO 모듈이 확대됩니다. 아래와 같이 디지털 출력을 예로 들어 보겠습니다.



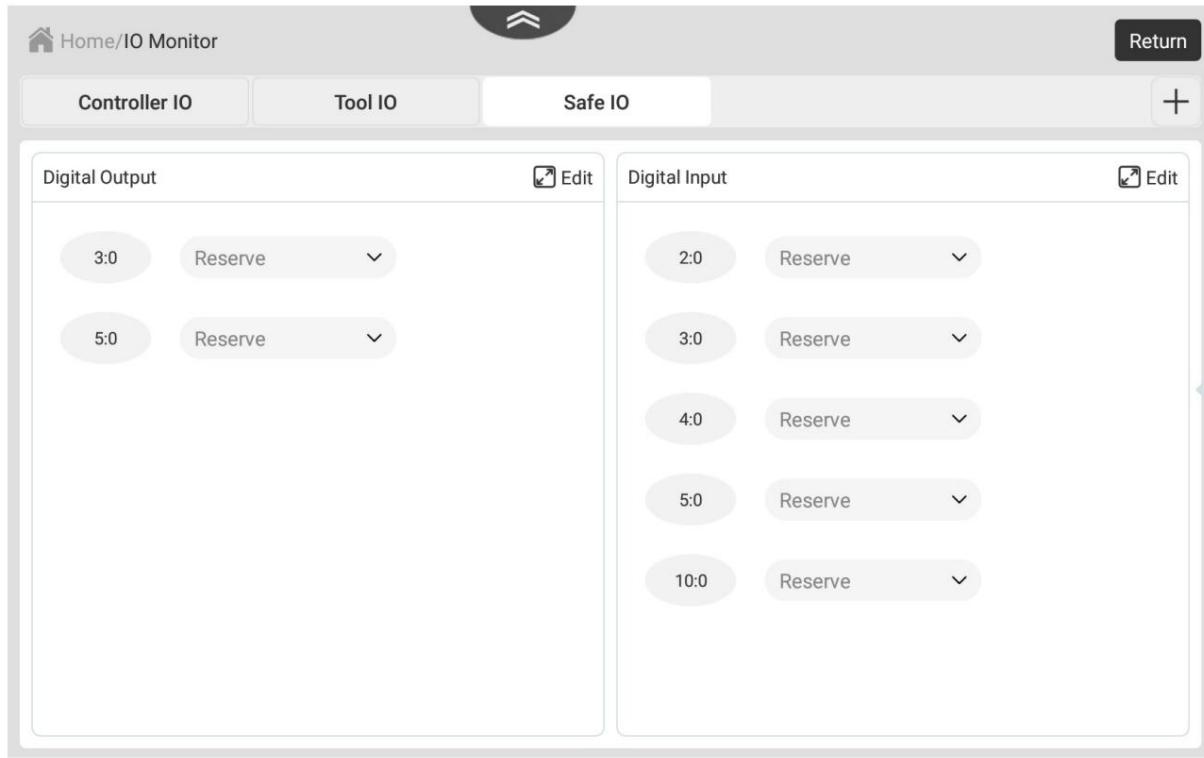
IO Note 추가를 클릭하여 메모 설정 페이지로 들어갑니다. 포트 오른쪽에 있는 메모 상자를 클릭하면 해당 포트에 대한 메모를 추가하거나 수정할 수 있습니다. 설정 후 저장을 클릭합니다.



안전한 I/O

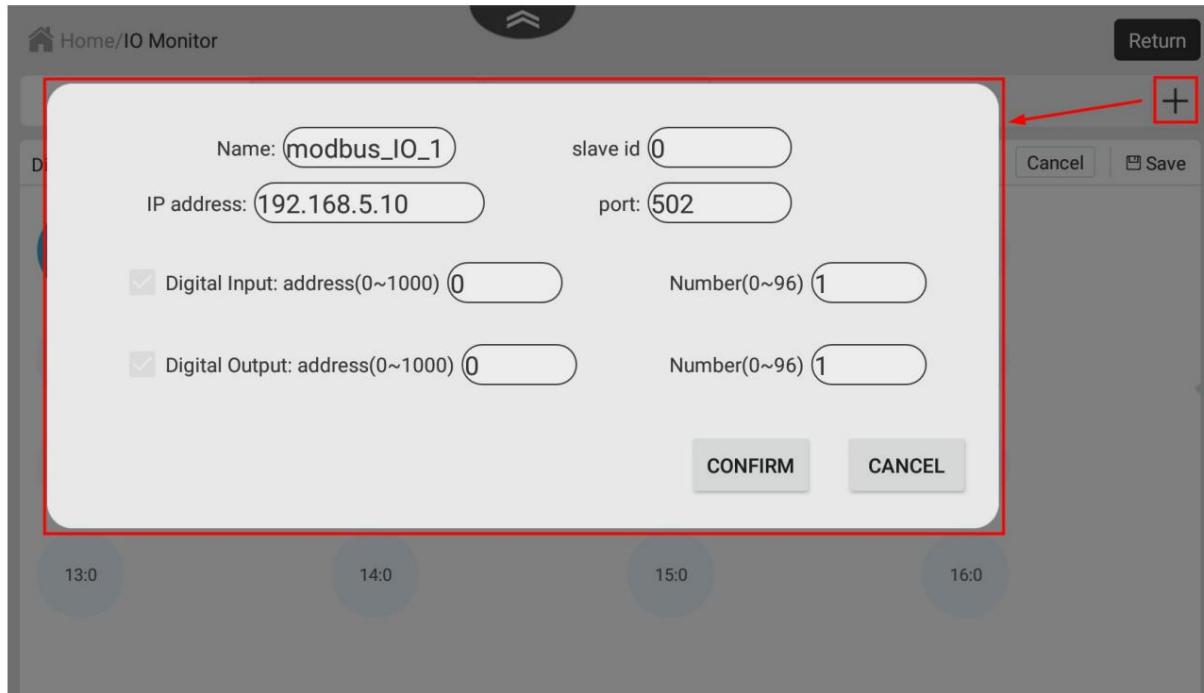
Safe IO 페이지에서 각 안전 I/O 인터페이스의 기능을 설정할 수 있습니다. 편집을 클릭하여 기능을 수정하고 설정 후 저장을 클릭합니다. safe I/O에 대한 자세한 내용은 해당 로봇 하드웨어 가이드의 IO 설명을 참조하십시오. 제어 캐비닛의 유형이 CCBOX인 경우 Safe I/O와 Universal I/O는 동일한 단자를 공유합니다. 리모트 I/O(고급 설정 포함)로 구성된 단자는 다음과 같이 설정할 수 없습니다.

안전한 I/O.



확장 IO

기본 탭 외에 +를 클릭하여 아래와 같이 Modbus 통신에서 IO를 모니터링하기 위한 사용자 지정 탭을 추가할 수 있습니다.



- 이름: 탭의 이름입니다.

- 슬레이브 ID: 장치 ID를 입력합니다.
- IP 주소: Modbus 장치의 주소를 입력합니다.
- 포트: Modbus 통신의 포트 번호를 입력합니다.
- 디지털 입력/디지털 출력: 레지스터 선택 후 DI/DO의 레지스터 주소와 번호를 구성합니다.
기능.

확인을 클릭하면 IO 모니터 모듈에 새 탭이 나타납니다. 이 기능은 컨트롤러를 다시 시작한 후에만 적용됩니다.

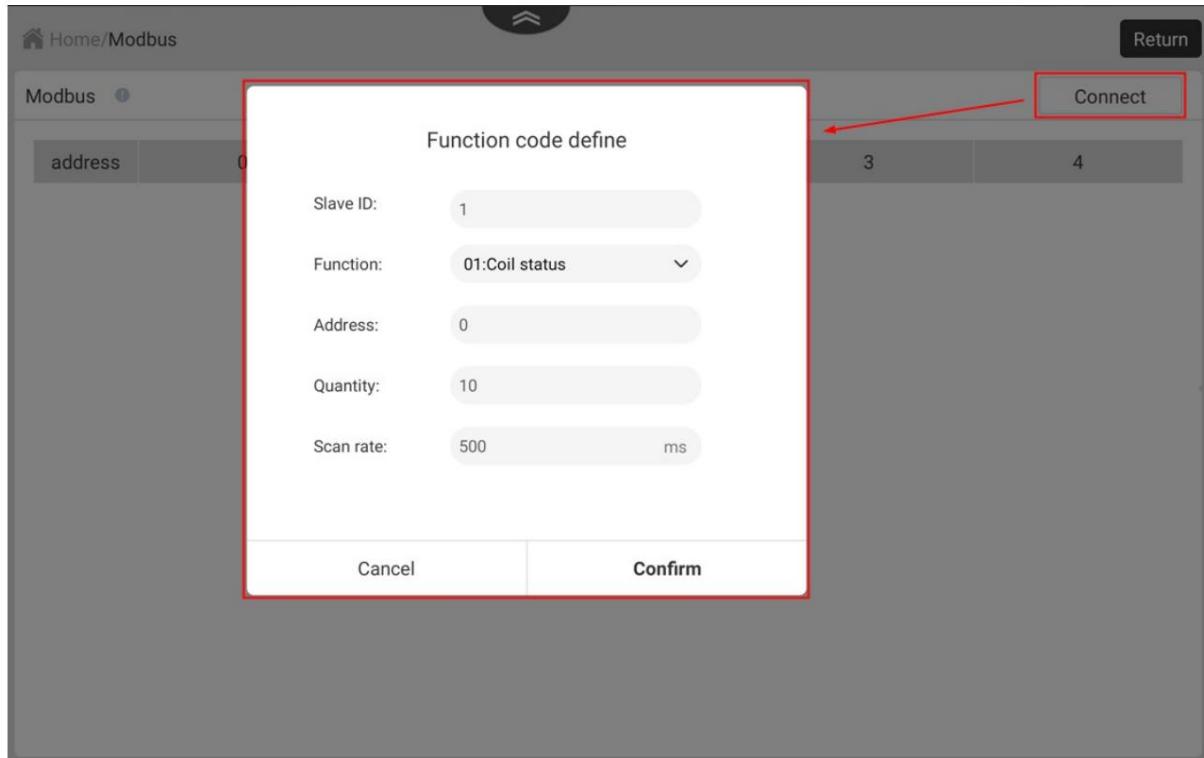
클릭



탭 오른쪽에 있는 탭을 삭제할 수 있습니다.

5.2 모드버스

이 모듈은 Modbus 슬레이브를 연결하는 데 사용됩니다.



- 슬레이브 ID: 슬레이브 장치 ID.
- 기능: 코일 레지스터, 이산 입력, 홀드 레지스터 및 입력 레지스터를 포함한 슬레이브 장치의 기능 유형을 선택합니다. 주소/수량: 주소 및 레지스터 수.
-
- 스캐닝 속도: 로봇 팔이 슬레이브 스테이션을 스캐닝하는 시간 간격.

연결을 클릭하여 슬레이브 스테이션에 연결을 시작합니다. Disconnect를 클릭하여 슬레이브와의 연결을 끊습니다.
역.

The screenshot shows a software interface for Modbus communication. At the top left is a house icon labeled "Home/Modbus". In the top right is a "Return" button. The central area displays "Modbus Slave ID:1 Function:02 Scan rate:500ms". A "Disconnect" button is located at the top right of this section. Below is a table with the following data:

address	0	1	2	3	4
000000	0	1	0	0	0
000005	0	0	0	1	0

- 슬레이브 기능이 코일 레지스터 또는 홀드 레지스터인 경우 테이블의 셀을 클릭하면 해당 주소의 값과 별칭을 수정할 수 있습니다.
- 슬레이브 기능이 아산 입력 또는 입력 레지스터인 경우 테이블의 셀을 클릭하면 해당 주소의 별칭을 수정할 수 있습니다.

5.3 글로벌 변수

모듈은 전역 변수를 구성하고 확인하는 데 사용됩니다.

전역 변수를 설정한 후 아래와 같이 Blockly 프로그래밍에서 해당 블록을 통해 변수를 호출하거나 Script 프로그래밍에서 변수 이름을 통해 변수를 호출할 수 있습니다.

The screenshot shows the 'Variables Settings' page in CR Studio. At the top, there are buttons for 'Delete', 'Change', and 'Add'. Below is a table with columns: Variable Name, Type, Global Hold, and Value. Two rows are listed:

Variable Name	Type	Global Hold	Value
var_1	bool	true	true
var_2	string	true	Great

CR Studio는 다음 유형의 전역 변수를 지원합니다.

- 부울: 부울 값
- String: String
- int: Integer
- float: 배정도 부동 소수점: 로봇의 점은 아래 그림과 같이 지정된 위치로 로봇을 이동하여 얻을 수 있습니다.

전역 보류로 설정된 변수의 경우 전역 변수의 같은 블록 또는 스크립트 프로그래밍에서 수정된 후 여기에서 동기화됩니다. 그렇지 않으면 수정된 값은 해당 프로젝트에만 적용되며 전체적으로 동기화되지 않습니다.

Variable Name	point_1	
Variable Type	point	
Value		
X	Y	Z
RX	RY	RZ
ARM	USER	TOOL
Get Point		
<input checked="" type="checkbox"/> Global Hold		
Cancel	Add	

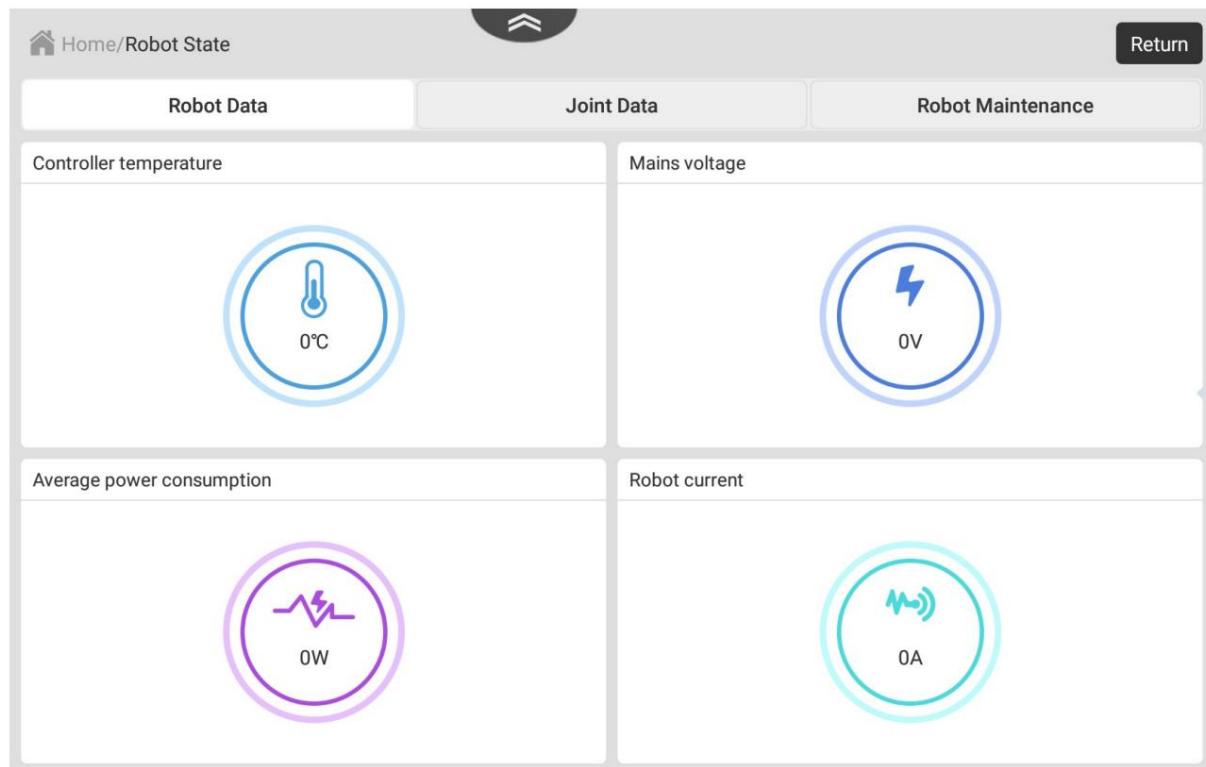
5.4 로봇 상태

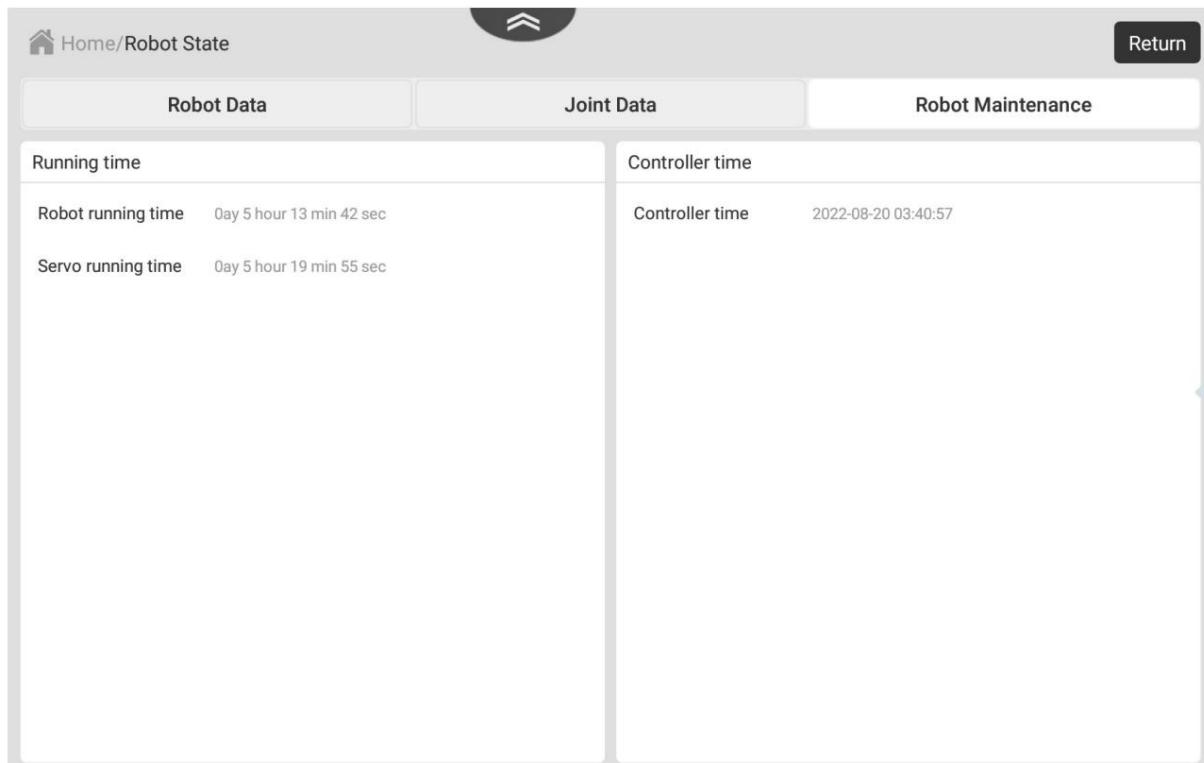
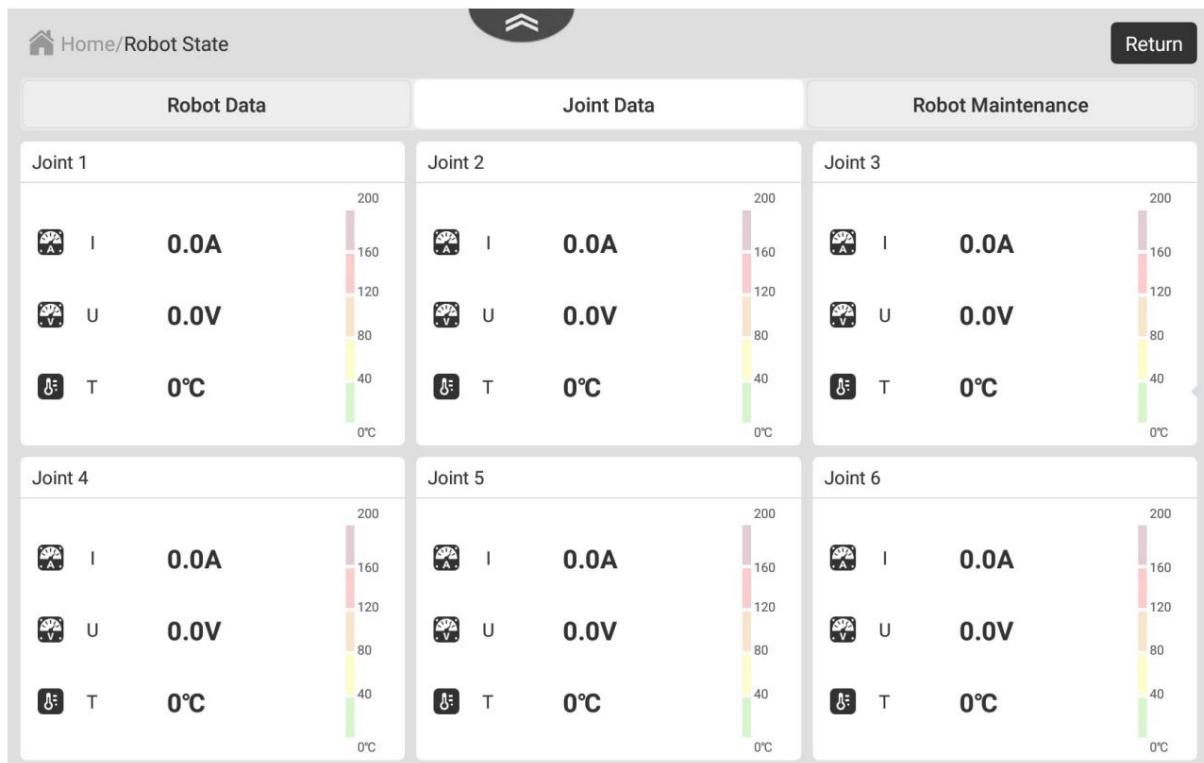
이 모듈은 로봇의 실시간 상태를 확인하는 데 사용됩니다. 제어 캐비닛 유형이 CCBOX인 경우 평균 소비 전력 및 로봇 전류는 표시되지 않습니다.



메모

관리자 모드에서만 런타임 데이터를 지우고 컨트롤러 시스템 시간을 수정할 수 있는 권한이 있습니다.





5.5 실행 로그

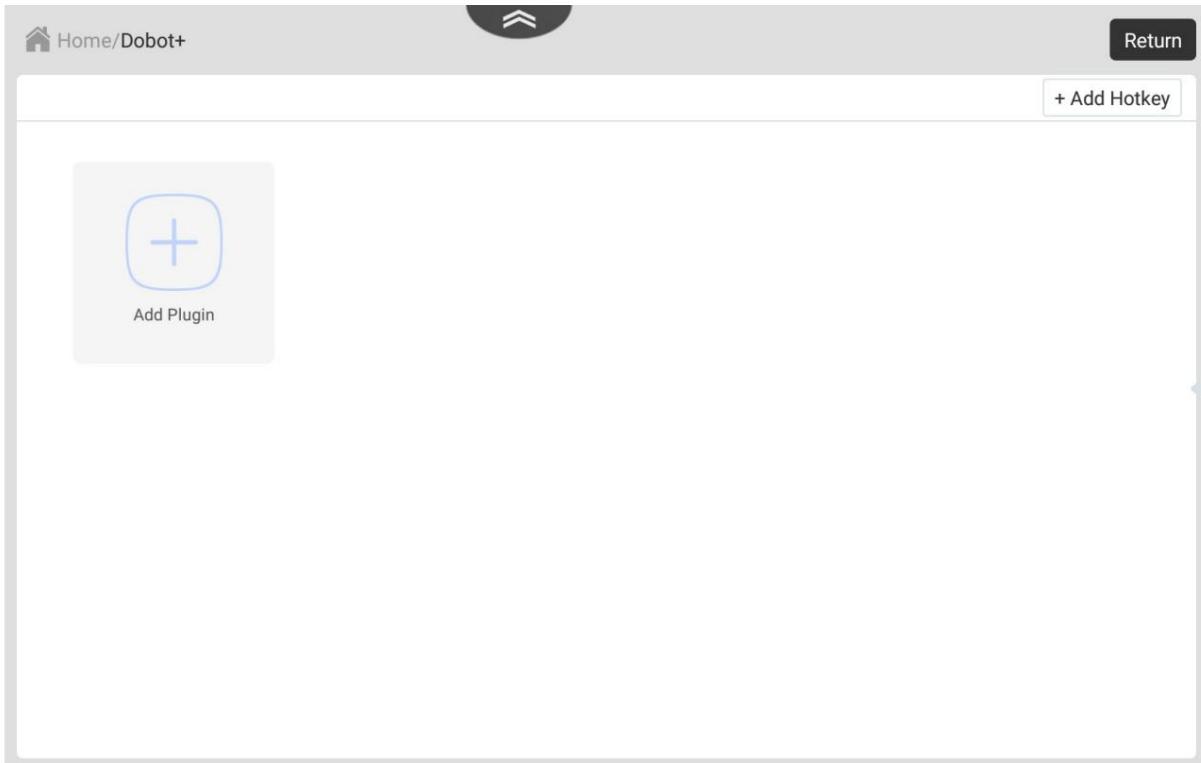
실행 로그를 보면 로봇의 동작 이력을 알 수 있습니다.

The screenshot shows a user interface for viewing run logs. At the top, there is a navigation bar with icons for Home, Run Log, and a central search bar. To the right of the search bar are 'Return', 'Clear', and 'Export' buttons. Below the search bar, there are fields for 'Start time' (set to 2022-08-20) and 'End time' (set to 2022-08-20), and a 'Key word' input field with a magnifying glass icon. The main area displays a list of log entries:

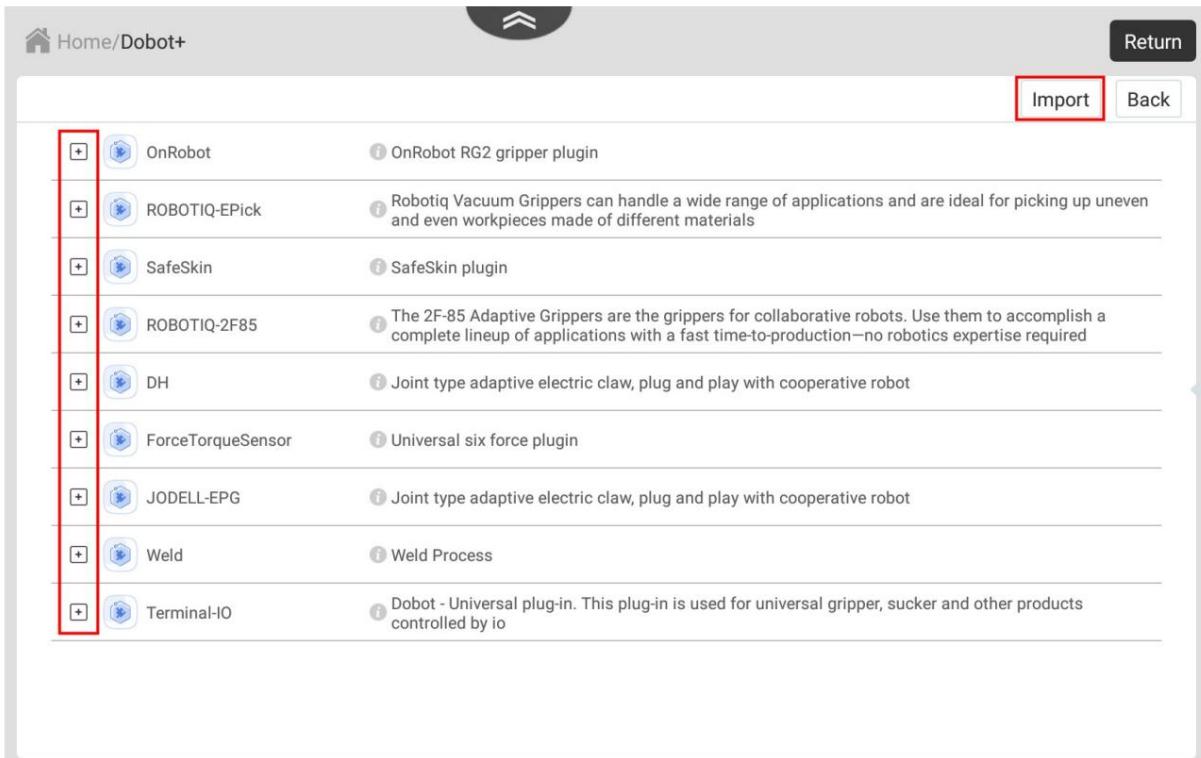
```
Run log:  
2022-08-18 15:06:56 Connect device  
2022-08-18 15:07:08 Connect device  
2022-08-18 15:07:36 Connect device  
2022-08-18 15:07:47 Connect device  
2022-08-18 15:46:23 Connect device  
2022-08-18 15:56:48 Connect device  
2022-08-18 17:49:11 Connect device  
2022-08-18 17:49:45 Change power status:Disable  
2022-08-18 17:52:10 Change power status:Enable  
2022-08-18 17:53:09 Add point:P1  
2022-08-19 10:59:10 Connect device  
2022-08-19 11:25:41 Emergency stop  
2022-08-19 11:25:42 Change power status:Disable  
2022-08-19 11:25:43 Id:12288 Type:controller Reason:Press the emergency stop button  
2022-08-19 11:25:43 Id:12296 Type:controller Reason:The robot is powered off  
2022-08-19 12:01:15 Emergency stop  
2022-08-20 10:08:07 Connect device
```

5.6 두봇+

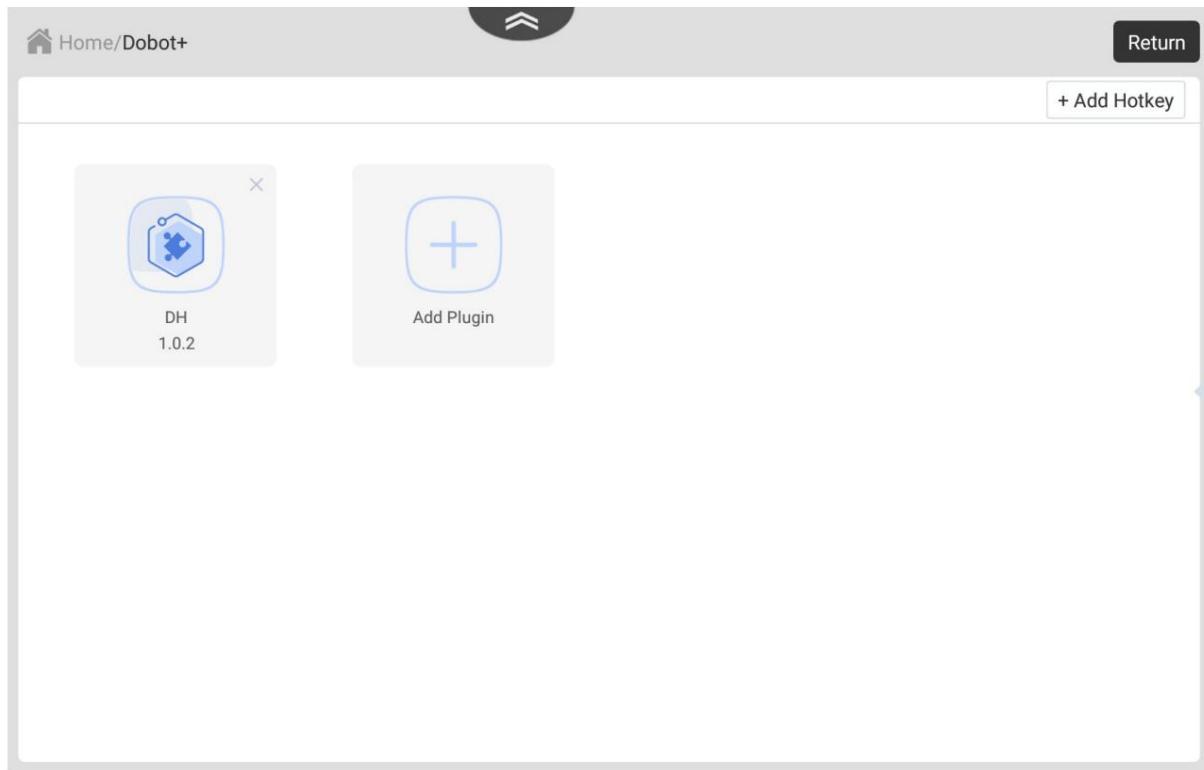
이 페이지는 로봇의 엔드 툴을 제어하기 위한 엔드 플러그인을 설치하고 구성하는 데 사용됩니다.



플러그인 추가를 클릭하여 플러그인 목록을 봅니다. +를 클릭하여 플러그인을 설치하거나 가져오기를 클릭하여 플러그인을 업로드할 수 있습니다.



플러그인은 설치 후 두봇+ 메인 페이지에 표시됩니다.



한편 Blockly 프로그래밍과 Script 프로그래밍에는 관련 블록/명령어가 추가될 예정이다.

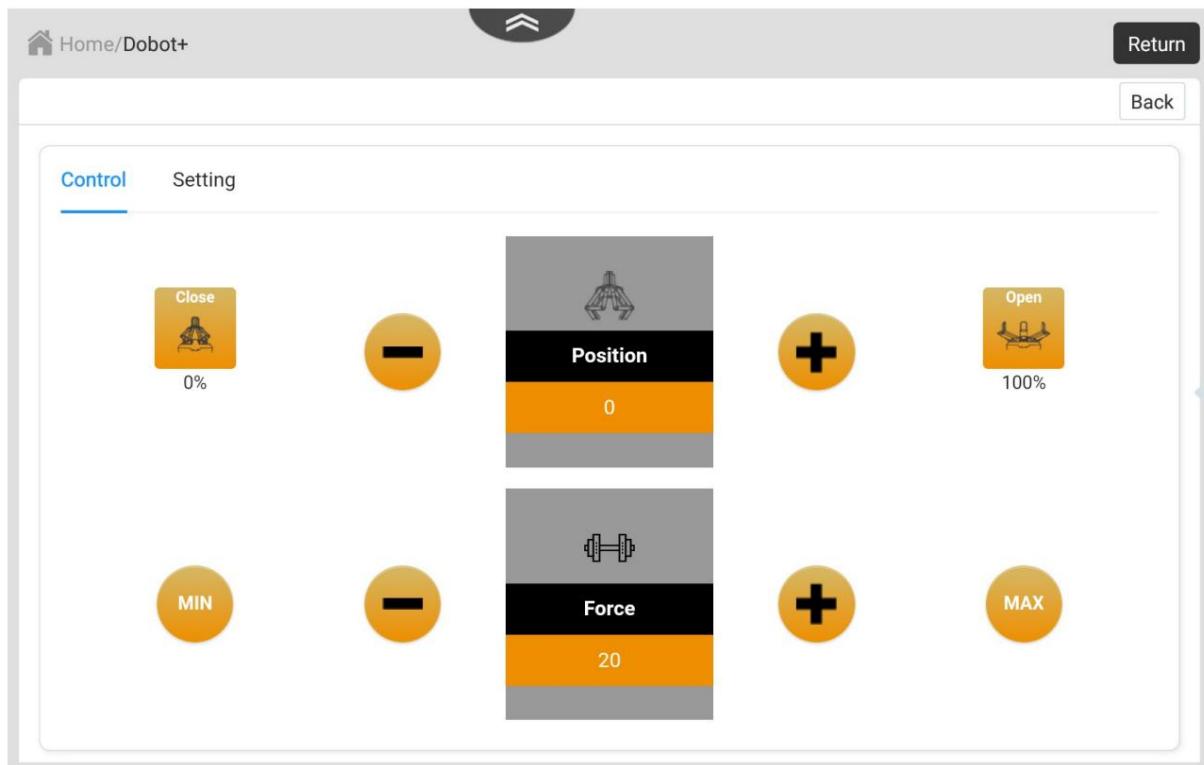
이제 아래 그림과 같이 DH 그리퍼를 예로 들어 보겠습니다.

The image compares two programming environments: 'Program/Script' on the left and 'Program/Blockly' on the right.

Program/Script: This interface shows a 3D simulation of a robotic arm with a gripper. Below it is a toolbar with four buttons: play, stop, pause, and a script editor icon. A 'Name:' field is present above a 'Func List' section. The 'Func List' includes categories like String, Custom, IO, Modbus, Program Manage, Advanced, and DH. The 'DH' category is highlighted with a red box and contains sub-blocks: DhInit, DhOpen, DhClose, DhSetForce, DhSetPosition, DhGetStatus, and DhGetPosition.

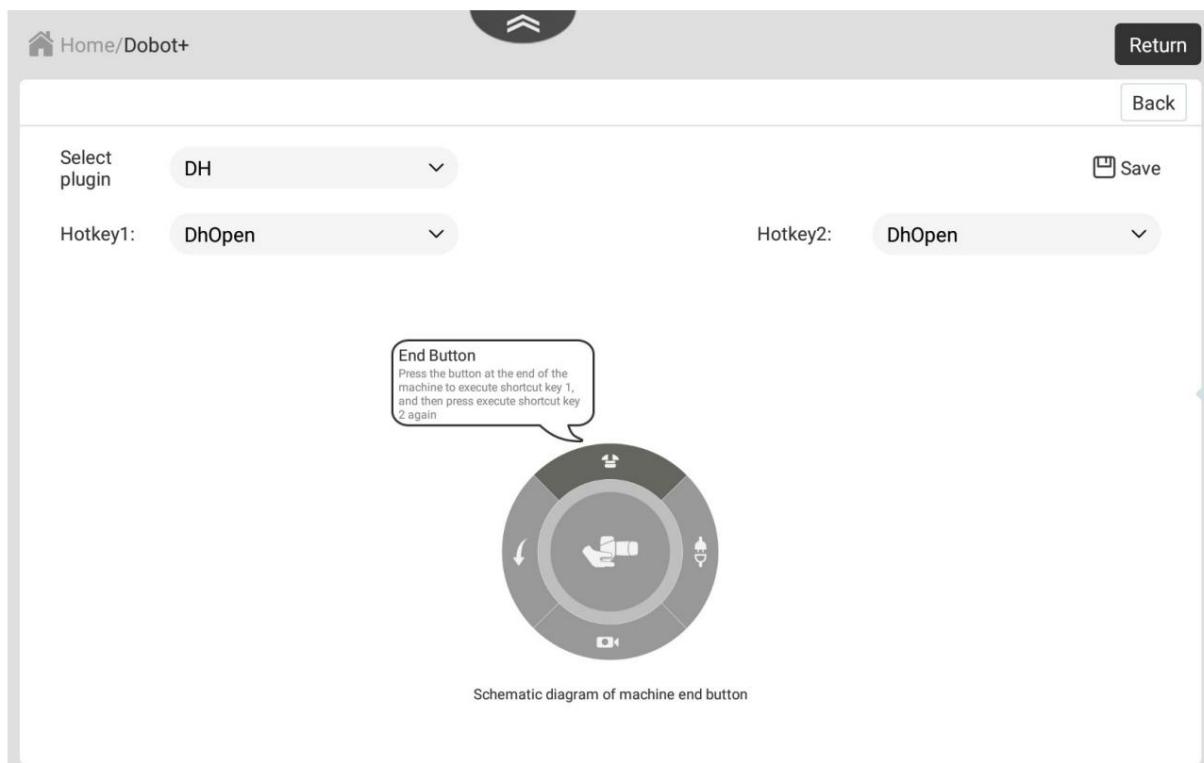
Program/Blockly: This interface shows a similar 3D simulation. It features tabs for Program, Point, and Debug, with a 'Name:N' input field. The 'Program' tab displays a list of blocks under the 'DH' category, which is also highlighted with a red box. These blocks include Init DH, Open DH with, Close DH with, Set DH force as, Set DH position as, Get current DH status ID, and Get current DH position ID. There is also a 'DH' block at the bottom of the list.

플러그인 아이콘을 클릭하여 플러그인의 기본 기능을 구성합니다. 다른 플러그인은 구성 방법이 다릅니다. 이 섹션에서는 DH 그리퍼를 예로 들어 구성하는 방법을 소개합니다.



페이지 오른쪽 상단의 단축키 추가를 클릭하여 플러그인에 대한 단축키를 설정할 수 있습니다. 저장 후 종료 버튼을 통해 종료 도구를 제어할 수 있습니다.

예를 들어 DH를 선택합니다. Hotkey1을 DhOpen 으로 , Hotkey2를 DhClose 로 설정하고 저장을 클릭합니다. 그런 다음 아래와 같이 로봇 끝에 있는 버튼을 누르면 그립퍼가 열립니다. 버튼을 다시 누르면 그립퍼가 닫힙니다.



6 프로그래밍

- [6.1 블록리](#)
- [6.2 스크립트](#)

6.1 블록리

CR Studio는 블록 프로그래밍을 제공합니다. 블록을 드래그하여 제어할 수 있습니다.

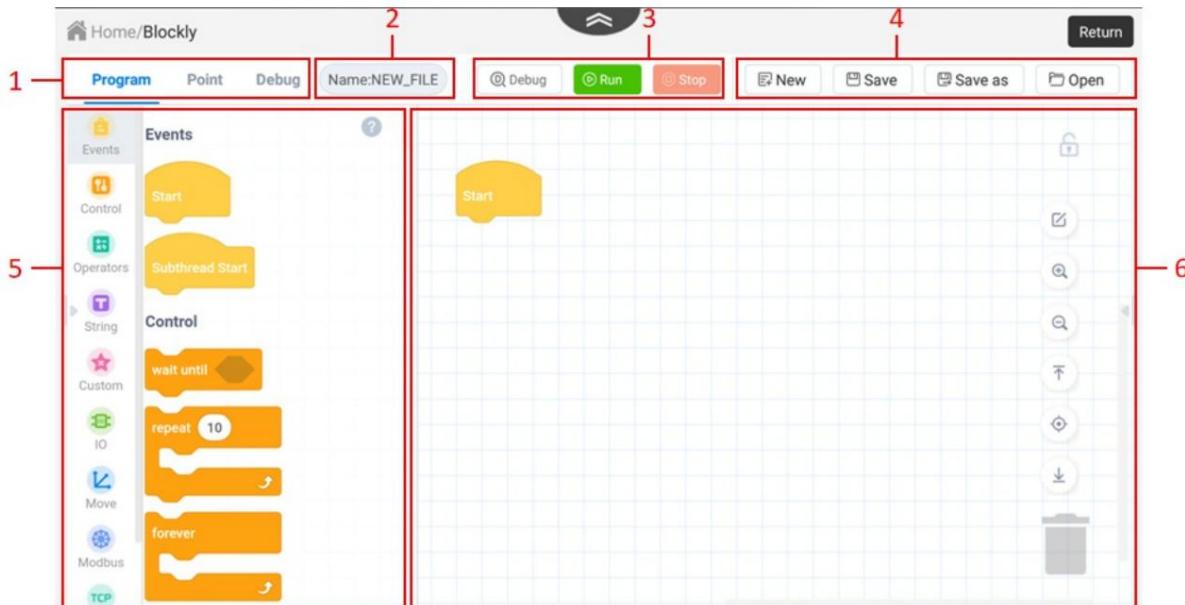
로봇.



이 문서는 블록 프로그래밍의 사용만을 소개합니다. 블록에 대한 구체적인 설명은 [부록 B](#)를 참조하십시오.

프로그램

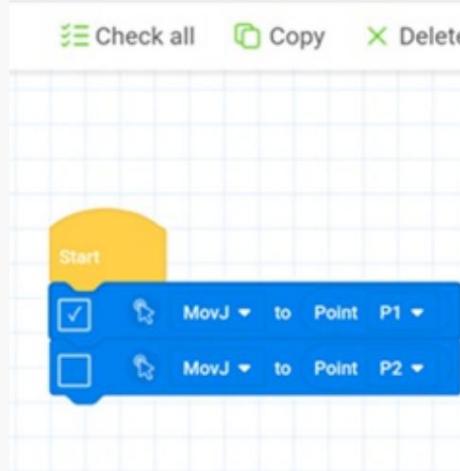
프로그램 인터페이스는 아래와 같이 Blockly 프로그래밍의 주요 인터페이스입니다.



아니요.	안건	설명
1	기능 탭	Blockly 프로그래밍의 기능 탭을 전환합니다.
2	프로젝트 이름	현재 프로젝트 이름 표시
3	프로젝트 제어 버튼	디버그(단일 단계 실행), 실행, 일시 중지 또는 중지할 프로젝트 제어
4	프로젝트 관리 버튼	<p>프로젝트 파일을 관리합니다. 열기를 클릭하면 저장된 프로젝트를 열어 다음과 같은 기능을 구현할 수 있습니다.</p> <ul style="list-style-type: none"> • 블록 프로그램을 스크립트 프로그램으로 변환하고 변환 후 스크립트 모듈에서 엽니다. • 선택한 프로젝트 파일을 로컬로 내보내고 프로젝트 파일(서버에서 자동으로 백업한 프로젝트 파일 포함)을 가져옵니다.

		앱) 로컬에서.
5	블록 영역	프로그래밍에 사용되는 블록을 다양한 색상과 카테고리로 나누어 제공합니다. 모듈의 오른쪽 상단을 클릭하면 해당 블록에 대한 설명이 표시됩니다.
6	프로그램 작성 영역	프로그램 편집 영역. 블록을 영역으로 드래그하여 프로그램을 편집할 수 있습니다. 프로그래밍 영역에서 블록을 마우스 오른쪽 버튼으로 클릭하여 블록 복사, 블록 삭제 및 블록 그룹(이벤트 블록 제외)을 서브루틴으로 전환하는 메뉴를 엽니다.  블록이 수정되었지만 저장되지 않은 경우 블록 왼쪽에 블록이 수정되었다는 메시지가 표시됩니다.

프로그래밍 영역 오른쪽에 있는 아이콘은 다음과 같습니다.

상	설명
	편집 모드로 들어갑니다. 편집 모드에서 복사하거나 삭제할 블록을 여러 개 또는 모두 선택할 수 있습니다. 확인 취소를 클릭하거나 프로그래밍 영역에서 다른 작업을 수행하여 편집 모드를 종료합니다.
	 Check all Copy Delete Cancel checking
	프로그래밍 영역 잠금/잠금 해제.
	확대/축소/프로그래밍 영역 복원.
	블록 상단으로/중앙 블록/블록 하단으로 돌아가기.
	블록을 이 아이콘으로 드래그하여 삭제하거나 블록을 길게 누르고 블록 삭제를 선택하여 삭제합니다.

가리키다

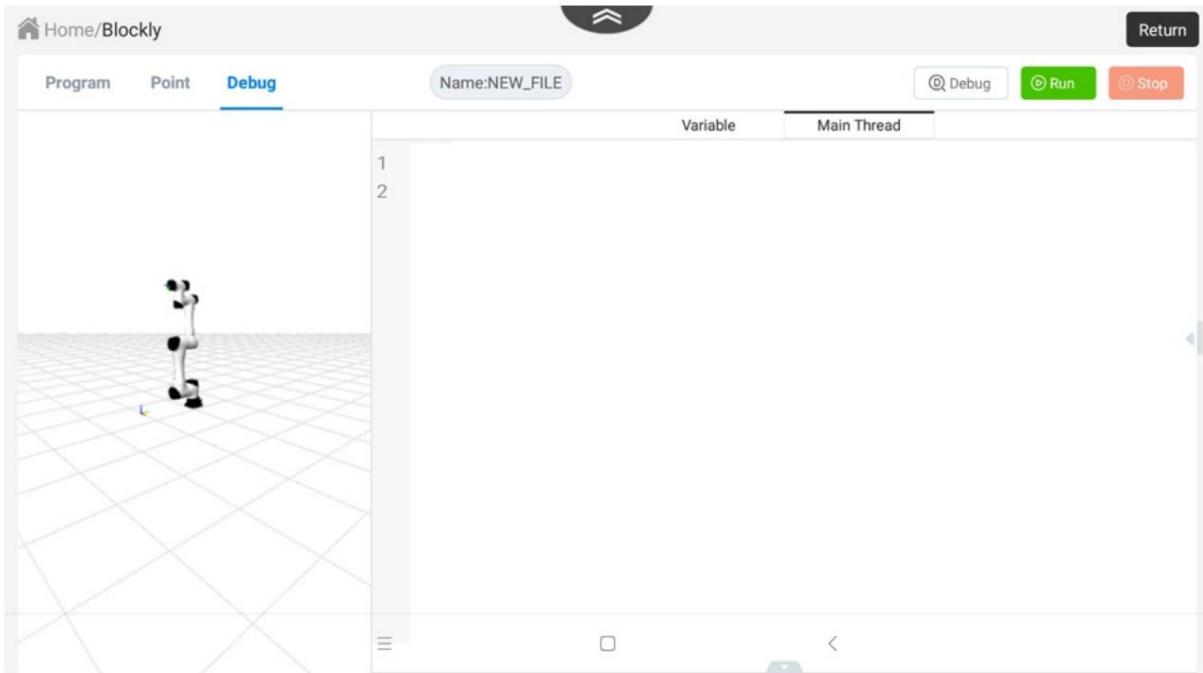
Point 인터페이스는 아래와 같이 프로그래밍에서 포인트를 관리하는 데 사용됩니다.



아니요.	안건	설명
1	가리키다 관리 버튼	<ul style="list-style-type: none"> 로봇을 지정된 지점으로 이동한 후 추가를 클릭하여 새 레코드를 추가합니다. 단일 포인트를 선택한 후 커버를 클릭하면 선택한 포인트가 현재 포인트로 커버됩니다. 포인트를 선택한 후 삭제를 클릭하면 현재 포인트가 삭제됩니다. 저장을 클릭하여 현재 지점을 저장합니다.
2	포인트 목록	추가된 포인트를 표시합니다. Tool은 공구 좌표계를 의미하고 User는 사용자 좌표계를 의미합니다.
삼	기능 실행	단일 지점을 선택한 후 모드를 선택합니다. 그런 다음 Run to를 길게 눌러 로봇을 선택한 지점으로 이동합니다.

디버그

디버그 인터페이스는 아래 그림과 같이 블록에 해당하는 스크립트, 프로젝트 작업 로그 및 로봇 동작의 3D 모델을 보고 실행 프로세스 및 결과가 기대에 부합하는지 확인하는 데 사용됩니다.



- Main thread 탭에는 Blockly 프로그램에서 변환된 스크립트가 표시됩니다. 실행 과정에서 실행 중인 명령을 실시간으로 확인할 수 있습니다.
- Variable 탭은 Blockly 프로그래밍에서 자체 정의된 변수를 표시합니다.
- 디버그를 클릭하면 버튼이 단계로 바뀝니다. 이 버튼을 클릭하면 이 프로그램을 단계별로 실행할 수 있습니다(단계는 여러 줄의 스크립트에 해당할 수 있는 단일 블록 실행을 나타냅니다).

프로그래밍 영역 및 오류를 포함한 실행 정보
실행 및 디버깅 프로세스 중 정보가 여기에 표시됩니다.

운영 절차

다음 예제는 두 지점 사이를 반복적으로 이동하도록 로봇을 제어하기 위해 블록 프로그램을 편집하는 절차를 설명합니다.

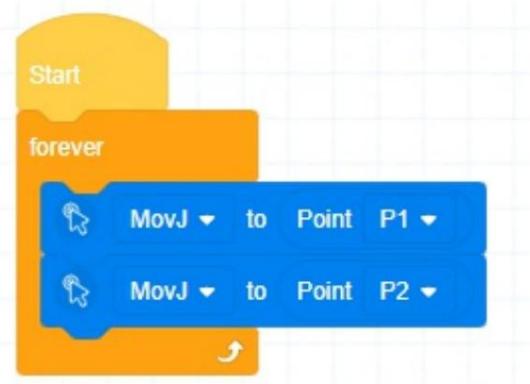
1. 포인트 페이지를 엽니다. 로봇 팔을 포인트(P1)로 이동하고 추가를 클릭하여 포인트 P1을 저장합니다.
2. 로봇 팔을 포인트(P2)로 이동하고 추가를 클릭하여 포인트 P2를 저장합니다.

No	name	Alias	X	Y	Z	RX	RY	RZ	Tool	User
<input type="checkbox"/>	P1		641.75	18.514	641.994	-96.391	-18.414	-127.586	No.0 ▾	No.0 ▾
<input type="checkbox"/>	P2		658.5202	190.8324	671.8841	-97.3992	-14.578	-100.7782	No.0 ▾	No.0 ▾

3. 프로그램 페이지를 엽니다. 블록 영역에서 영원히 블록을 드래그하여 시작 아래에 놓습니다.
차단하다.

4. 드래그 무한 블록 내부에서 P1을 대상 지점으로 선택합니다.

5. 드래그 이전 블록 아래에서 P2를 목표점으로 선택합니다.



6. 저장을 클릭하고 프로젝트 이름을 입력한 후 저장을 클릭합니다.

7. 실행을 클릭하면 로봇이 움직이기 시작합니다.

6.2 스크립트

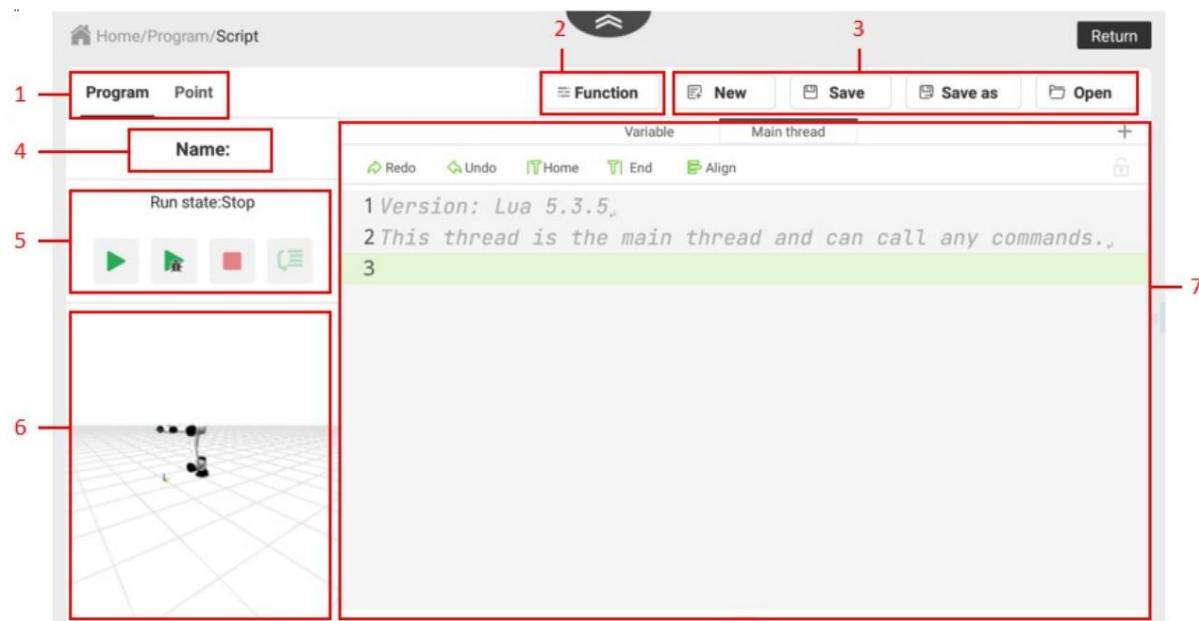
CR은 2차 개발을 위해 Lua 언어를 사용하는 모션 명령, TCP/UDP 명령 등과 같은 다양한 API를 지원합니다. CR Studio는 Lua 스크립트를 위한 프로그래밍 환경을 제공합니다. 자신만의 Lua 스크립트를 작성하여 로봇 작동을 제어할 수 있습니다.



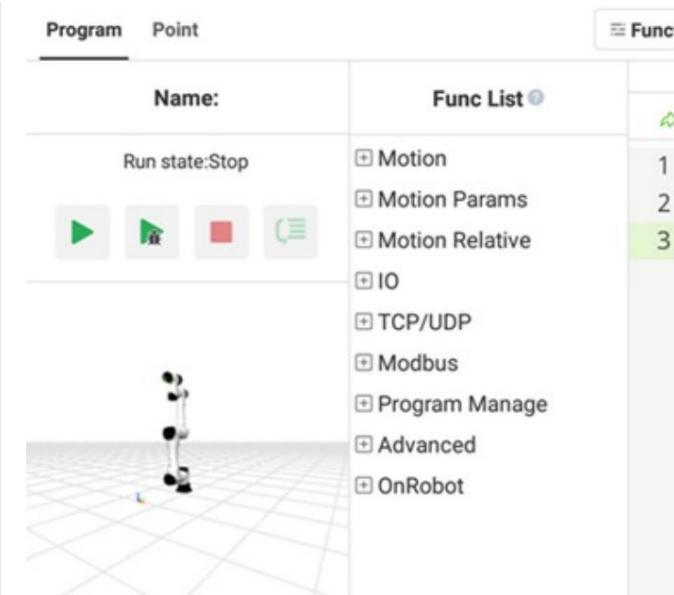
이 문서는 스크립트 프로그래밍의 사용에 대해서만 소개합니다. 명령에 대한 구체적인 설명은 [부록 C](#)를 참조하십시오.

프로그램

프로그램 인터페이스는 아래와 같이 스크립트 프로그래밍의 주요 인터페이스입니다.



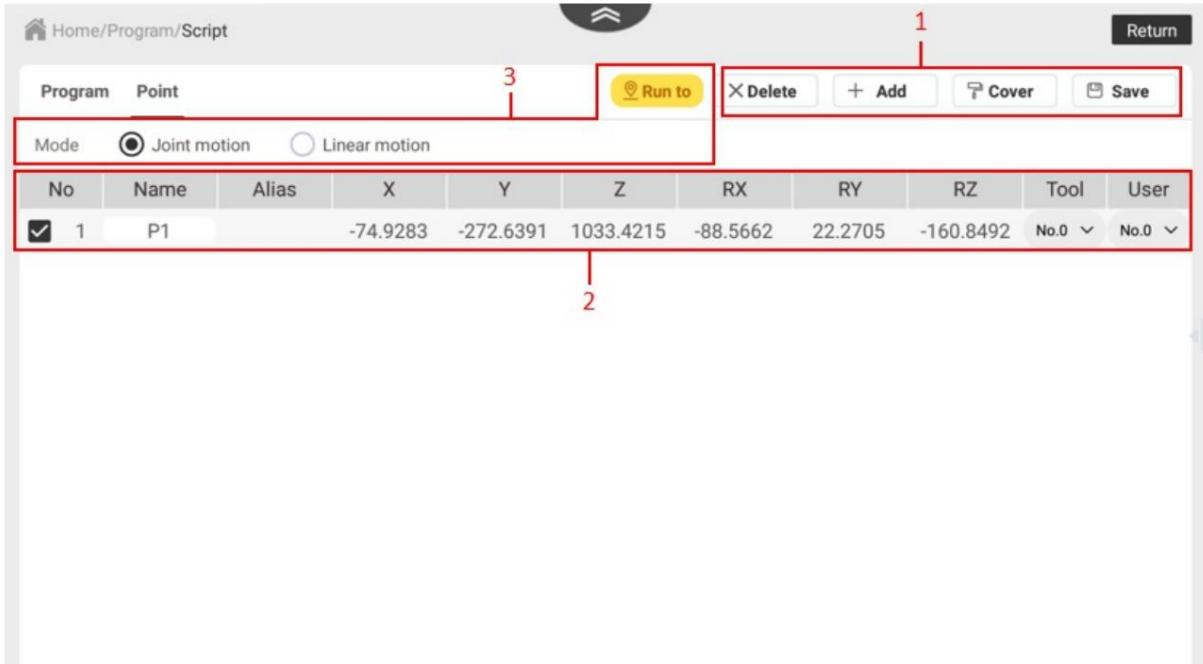
아니요.	안건	설명
1	기능 탭	스크립트 프로그래밍의 기능 탭을 전환합니다.
		기능 목록 표시/숨기기

		 <p>함수 목록에서 두봇 API를 클릭하면 매개변수를 구성하여 해당 코드를 생성 할 수 있습니다. 클릭 가이드</p> <p> API를 보려면</p>
2	기능 목록	<p>프로젝트 파일을 관리합니다. 열기를 클릭하면 선택한 프로젝트 파일을 로컬로 내보내고 로컬에서 프로젝트 파일(앱에서 자동으로 백업한 프로젝트 파일 포함) 가져올 수 있습니다.</p>
4	프로젝트 이름	현재 프로젝트 이름 표시
5	프로젝트 제어 영역	<p>프로젝트 상태 표시 및 프로젝트 실행 제어</p> <p> : 프로그램 실행을 시작하려면 클릭하십시오.</p> <p> : 클릭하면 디버그 모드로 들어갑니다. 아이콘이 디버그 모드로 바뀝니다. 이 경우 중단점을 사용하여 프로그램을 실행하거나 프로그램을 단계별로 디버깅할 수 있습니다.</p> <p> : 클릭하면 프로그램 실행이 중지됩니다.</p> <p> : 프로그램을 단계별로 실행하려면 클릭하십시오(디버그 모드에서만 사용됨).</p>
6	3D 로봇 모델	실시간으로 3D 로봇 모델 표시
7	코드 영역	<p>Lua 코드 편집 영역 메인 스</p> <ul style="list-style-type: none"> • 레드는 로봇의 메인 코드를 실행하는 데 사용됩니다. • 변수 탭은 프로젝트에서 사용되는 변수를 정의합니다. • (안드로이드만 해당) 좌측 인덱스 번호 클릭 <p> 중단점을 추가하는 코드입니다. 디버그 모드에서 클릭하여 중단점 실행을 시작하면 중단점까지 실행될 때 프로그램이 자동으로 일시 중지됩니다.</p> <ul style="list-style-type: none"> • I/O 제어와 같이 기본 스레드와 병렬로 실행되는 하위 스레드(5개 이하)를 추가하려면 오른쪽 상단 모서리에 있는 +를 클릭합니다. • 실행 또는 디버깅 중 오류 메시지를 포함하여 실행 정보를 표시하려면 코드 영역 아래를 클릭하십시오.



가리키다

포인트 페이지는 아래와 같이 프로그래밍에서 포인트를 관리하는 데 사용됩니다.



아니요.	안건	설명
1	포인트 관리 버튼	<ul style="list-style-type: none"> 로봇을 지정된 지점으로 이동한 후 추가를 클릭하여 새 레코드를 추가합니다. 단일 포인트를 선택한 후 커버를 클릭하면 선택한 포인트가 현재 포인트로 커버됩니다. 포인트를 선택한 후 삭제를 클릭하면 현재 포인트가 삭제됩니다. 저장을 클릭하여 현재 지점을 저장합니다.
2	포인트 목록	추가된 포인트를 표시합니다. Tool은 공구 좌표계를 의미하고 User는 사용자 좌표계를 의미합니다.
3	기능 실행	단일 지점을 선택한 후 모드를 선택합니다. 그런 다음 Run to를 길게 눌러 로봇을 선택한 지점으로 이동합니다.

운영 절차

다음 예제는 로봇이 두 지점 사이를 반복적으로 이동하도록 제어하기 위해 스크립트 프로그램을 편집하는 절차를 설명합니다.

- 포인트 패널을 엽니다. 로봇 팔을 포인트(P1)로 이동하고 추가를 클릭하여 포인트 P1을 저장합니다.

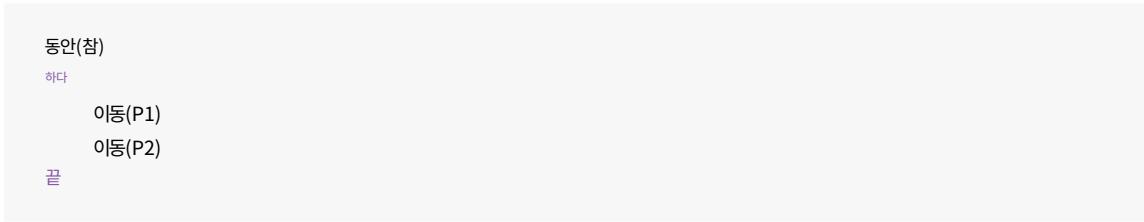
2. 로봇 팔을 다른 지점(P2)으로 이동하고 추가를 클릭하여 지점 P2를 저장합니다.

Program Point											
Mode	Joint motion		Linear motion		Run to		Delete		+Add	Cover	Save
No	name	Alias	X	Y	Z	RX	RY	RZ	Tool	User	
<input type="checkbox"/> 1	P1		658.5202	190.8324	671.8841	-97.3992	-14.578	-100.7782	No.0	No.0	
<input type="checkbox"/> 2	P2		626.9375	244.8497	719.0871	-93.2915	-9.6654	-89.7389	No.0	No.0	

3. 프로그래밍 영역에서 루프 명령을 추가합니다.

4. 루프 명령 아래에 모션 명령을 추가하고 P1을 목표 지점으로 설정합니다.

5. 다른 동작 명령을 추가하고 P2를 목표점으로 설정합니다.



6. 저장을 클릭하고 프로젝트 이름을 입력한 다음 예를 클릭합니다.

7. 클릭  , 로봇이 움직이기 시작합니다.

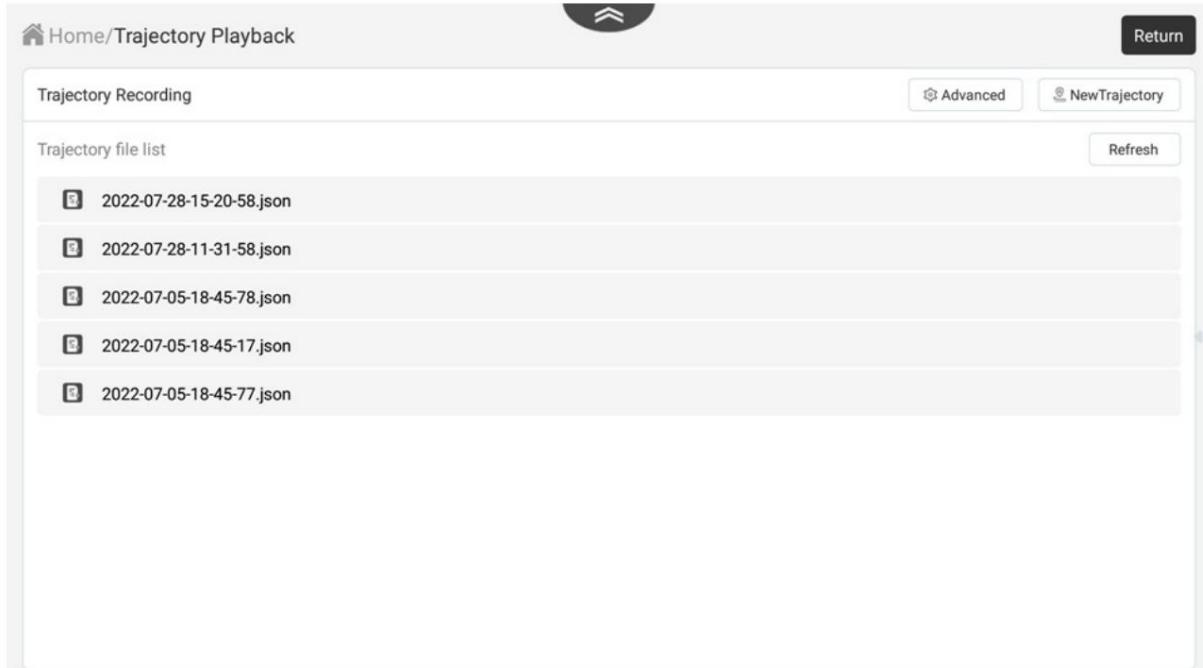
7 프로세스

- 7.1 궤적 재생 7.2 컨베이어
- 벨트

7.1 궤적 재생

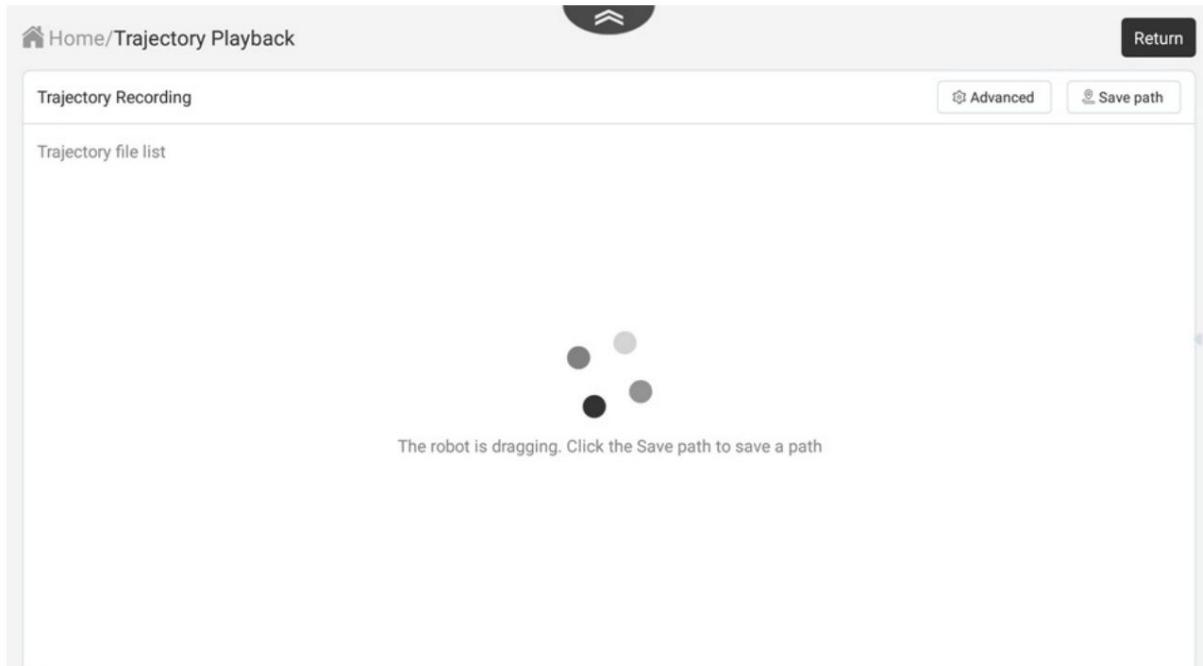
궤적 재생은 로봇을 수동으로 드래그하는 동안 기록된 궤적을 재생하는 데 사용됩니다.

팔. 기본 인터페이스는 아래와 같습니다.

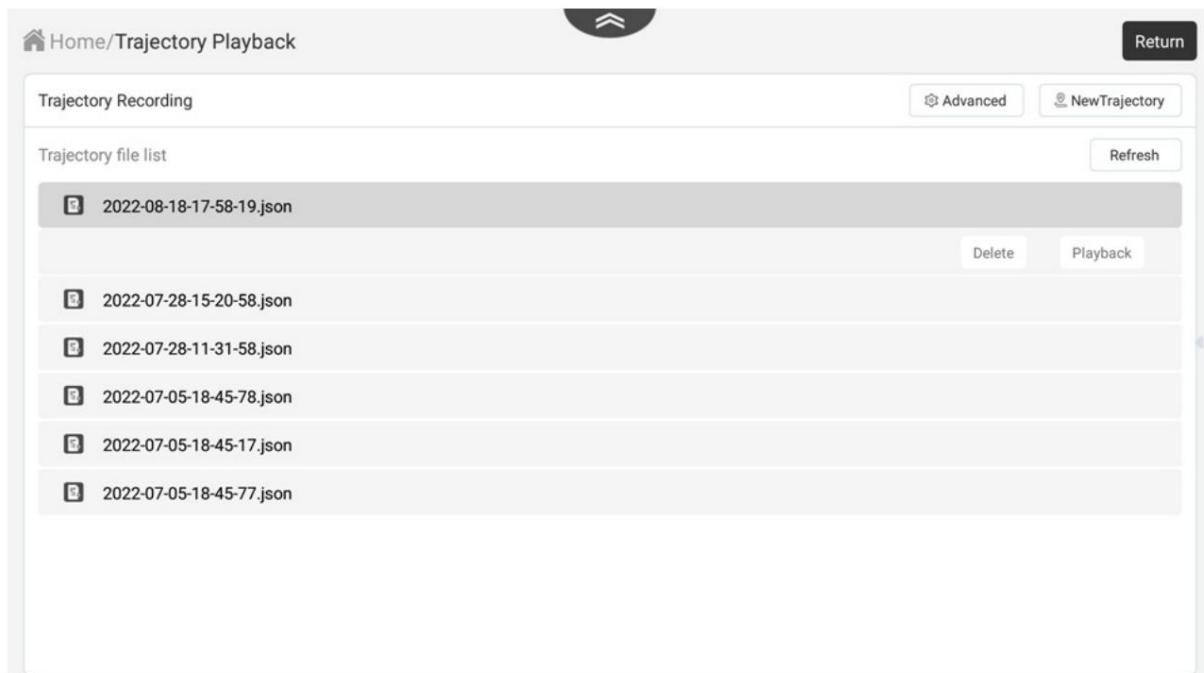


오른쪽 상단의 NewTrajectory를 클릭하면 로봇 끝에 있는 표시등이 노란색으로 바뀝니다.

이제 로봇을 끌 수 있습니다. 궤적은 재생을 위해 기록됩니다.



궤적을 기록한 후 오른쪽 상단의 경로 저장을 클릭합니다. 이제 표시등이 녹색으로 바뀌고 기록 시간을 따라 명명된 새 파일 목록에 추가됩니다.



저장된 궤적을 클릭하면 삭제, 이름 바꾸기 및 재생의 두 가지 버튼이 표시됩니다.

- 재생을 클릭하면 로봇의 기록된 궤적을 재생합니다. 이 경우 로봇 끝에 있는 표시등이 노란색으로 깜박이고 로봇이 움직이기 시작합니다. 재생 후 로봇은 움직임을 멈추고 표시등이 녹색으로 바뀝니다.

- Rename을 클릭하여 궤적 파일의 이름을 수정합니다. - 이 궤적을 삭제하려면 삭제를 클릭합니다. > 참고 >> 저장 된 궤적 파일은 블록 및 스크립트 프로그래밍에서 궤적을 통해 관련 호출 명령을 재생할 수 있습니다. 고급을 클릭하여 재생 모드를 설정합니다. 일정한 속도로 되풀이를 선택하지 않으면 재생 속도 비율을 설정할 수 있습니다.

Advanced

Recurring at a uniform speed

Playback Times **1**

Cancel

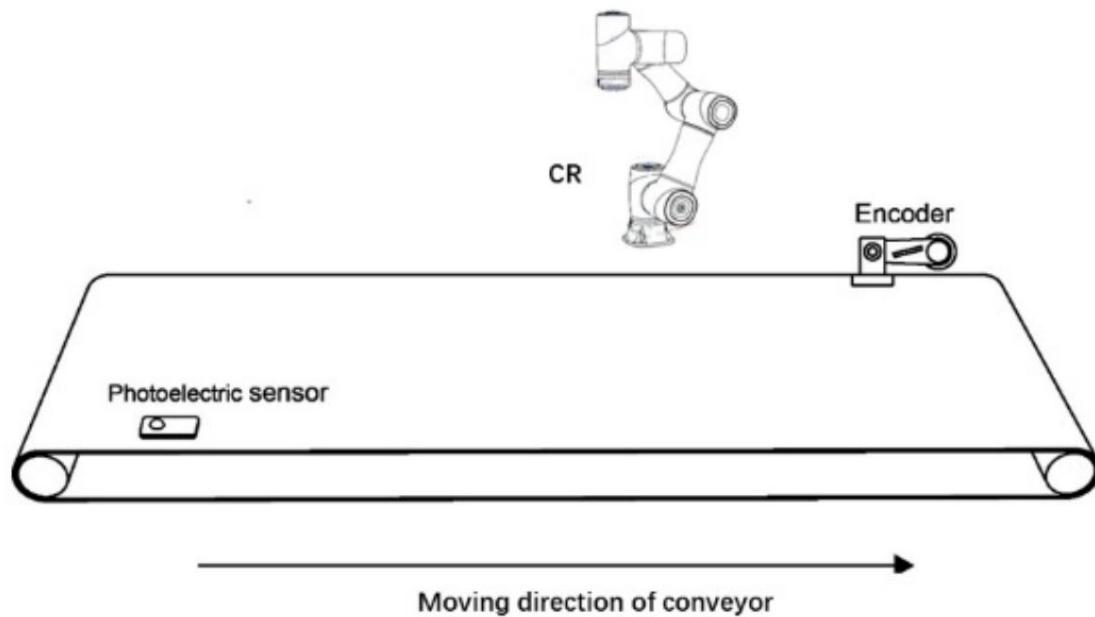
Confirm

7.2 컨베이어 벨트

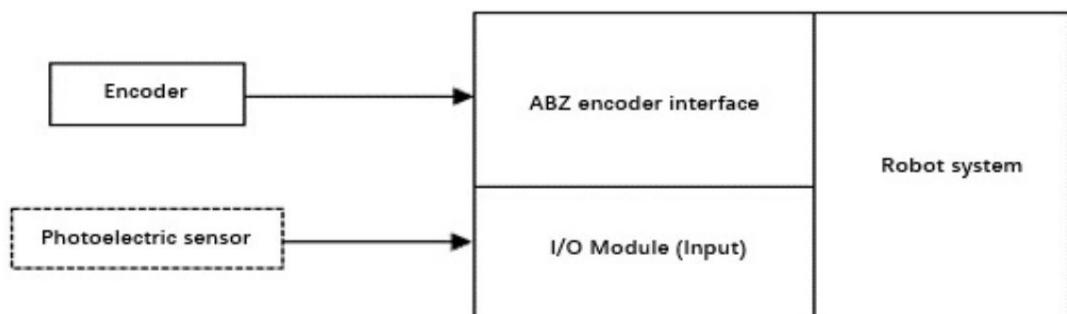
컨베이어 프로세스는 컨베이어가 움직일 때 광전 센서가 컨베이어에 있는 물체를 감지하고 로봇이 움직이는 물체를 학습하는 것입니다.

건축 환경

아래 그림은 컨베이어 추적의 전체 프로세스 환경을 보여줍니다.



통신 다이어그램은 다음과 같습니다.



인코더

엔코더를 사용하여 컨베이어 이동거리와 공작물 위치를 기록하고 이를 CR 로봇에 보고합니다. 엔코더는 아래와 같이 CR 컨트롤러의 ABZ 엔코더 인터페이스에 연결해야 합니다. 권장 엔코더 모델은 E6B2-CWZ1X(1000P/R)입니다.

E6B2-CWZ1X(1000P/R) 인코더를 예로 들어 핀 연결은 아래 표에 나와 있습니다.

색상	포트
검은색	A+
검정과 빨강	ㅏ
하얀색	B+
흰색과 빨간색	비
주황색	Z+
오렌지와 레드	지
갈색	입력/출력 5V
파란색	입출력 0V

광전 센서

광전 센서는 공작물 감지 여부에 따라 다른 레벨 신호를 출력합니다. 제어 시스템은 신호 에지를 통해 공작물을 감지할 수 있습니다. 아래와 같이 광전 센서를 컨트롤러의 DI 인터페이스에 연결합니다.

교정 컨베이어

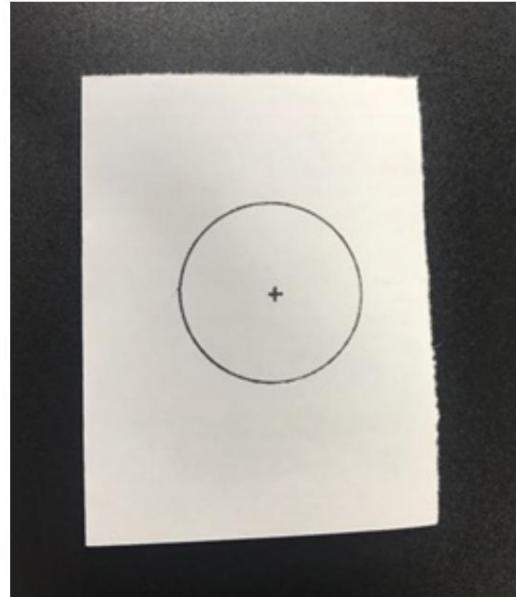
컨베이어 프로세스를 구성하기 전에 컨베이어와 로봇 간의 위치 관계를 얻기 위해 컨베이어를 보정해야 합니다. 다음 단계에서는 사용자 좌표계가 사용됩니다.

교정을 위해.

캘리브레이션을 하기 전에 로봇 끝에 캘리브레이션 바늘을 설치하고 라벨을 준비해야 합니다.

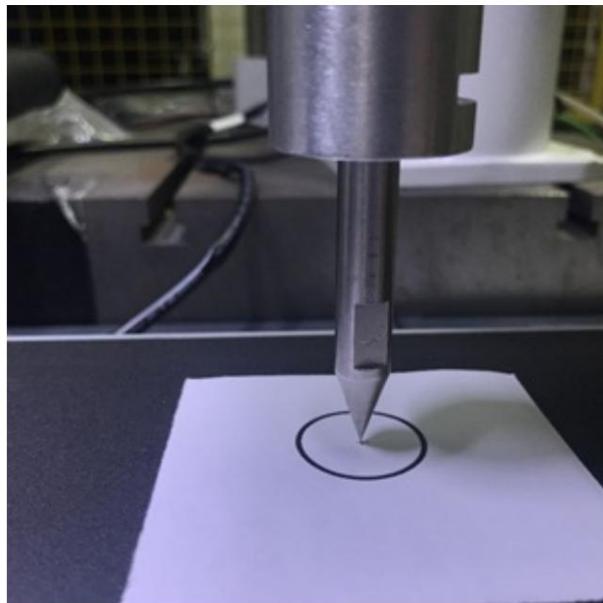
구경 측정.

1. 아래와 같이 컨베이어에 라벨을 붙입니다.



2. 사용자 좌표계 페이지로 들어갑니다.

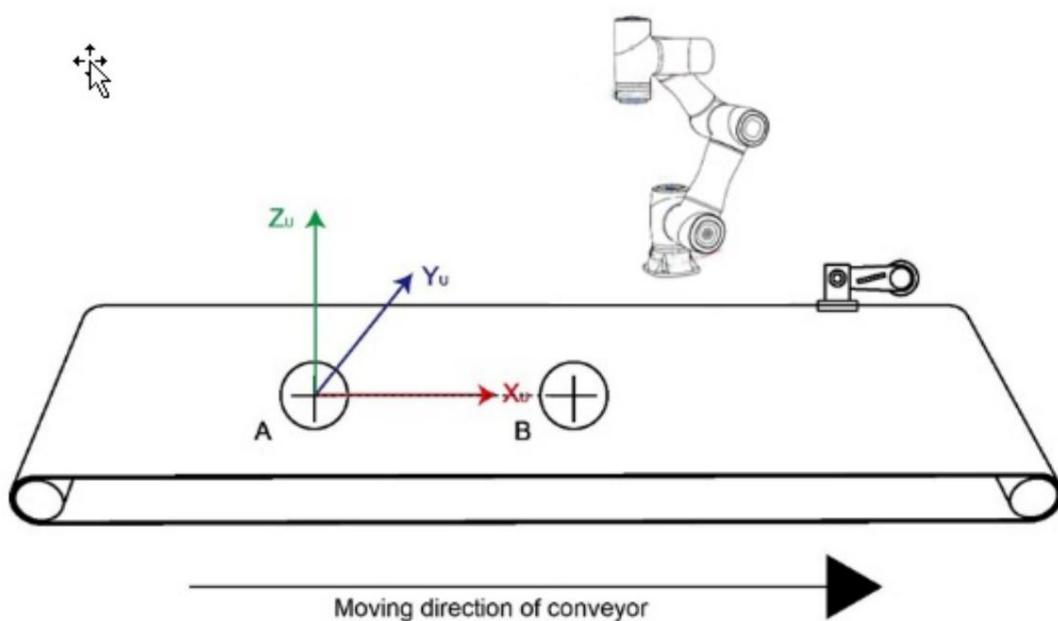
3. 로봇을 활성화합니다. 컨베이어의 라벨 위치로 로봇을 조그하여 첫 번째 지점(지점 A, 사용자 좌표계의 원점).



4. 컨베이어를 제어하여 지정된 거리를 이동합니다.

5. 로봇을 컨베이어의 라벨 위치로 이동하여 두 번째 지점(지점 B, x축)

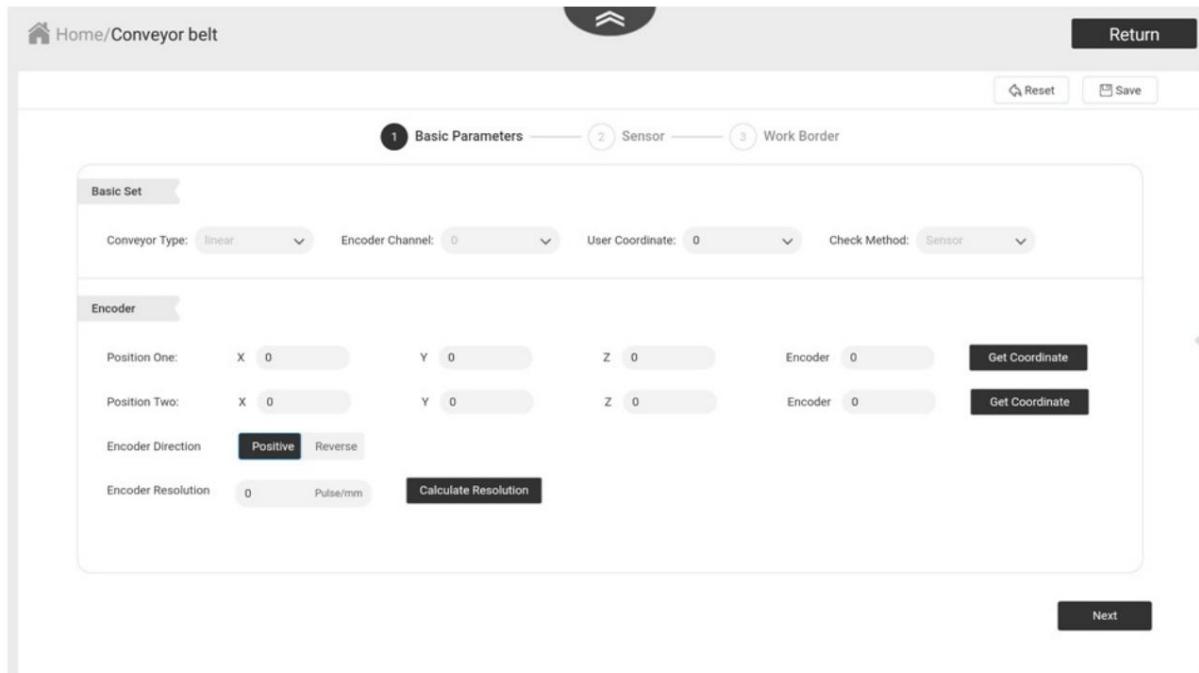
6. 로봇을 컨베이어의 지점 A와 B에 의해 결정된 라인에 없는 지점으로 이동하여 세 번째 위치를 얻습니다. 점(y축을 결정하기 위한 점 C)



7. 좌표계를 추가하거나 덮고 저장합니다.

컨베이어 구성

프로세스 페이지에서 컨베이어 벨트를 클릭하여 컨베이어 구성 시작합니다.



기본 설정

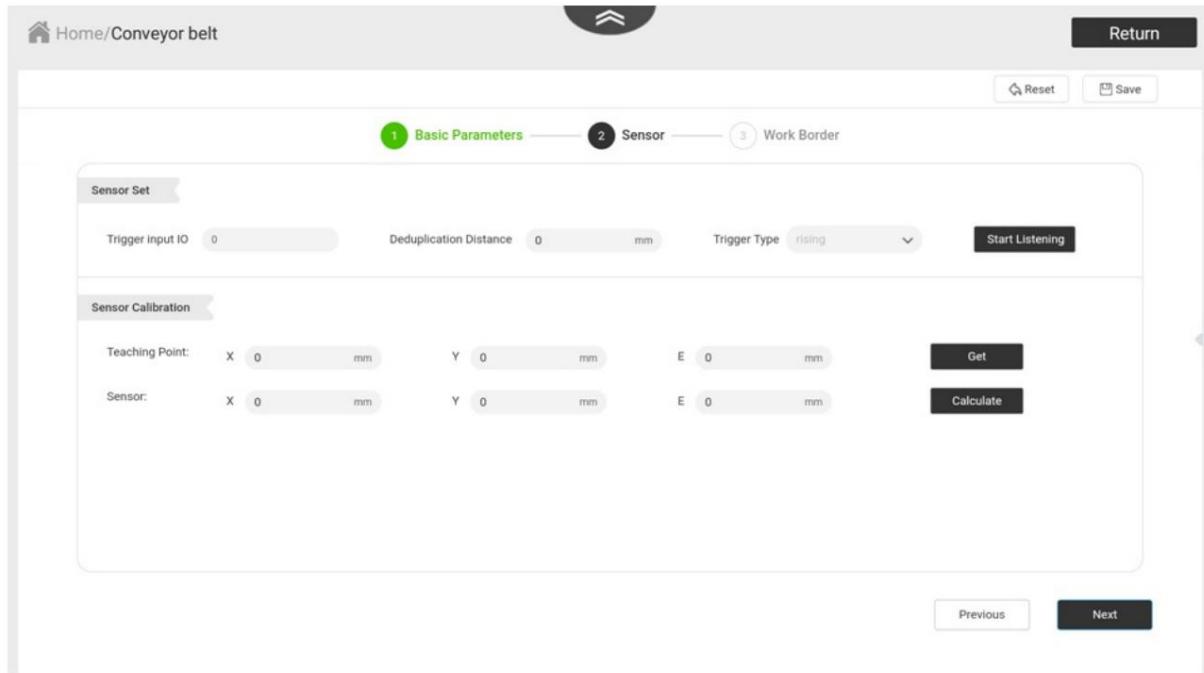
- 컨베이어 유형: 선형 유형만 지원됩니다(구성할 수 없음).
- 인코더 채널: 채널이 기본입니다(구성할 수 없음).
- 사용자 좌표: 마지막 단계에서 저장한 컨베이어 좌표계를 선택합니다.
- 확인 방법: 센서만 지원됩니다(구성할 수 없음).

인코더

인코더 해상도를 보정합니다. 엔코더 분해능은 컨베이어가 움직이는 단위 길이당 엔코더의 펄스 증분입니다.

- 컨베이어에 라벨을 붙입니다. 로봇을 컨베이어의 라벨 위치로 이동하고 Get을 클릭합니다.
Position 1 라인의 좌표를 조정하여 Position 1의 값을 얻습니다.
- 컨베이어를 제어하여 지정된 거리만큼 이동하고 정지시킵니다.
- 로봇을 컨베이어의 라벨 위치로 조그하십시오. 위치 한 줄에서 좌표 가져오기를 클릭하여 위치 2의 값을 얻습니다.
- 실제 조건에 따라 엔코더의 방향을 설정합니다(Positive를 예로 들어).
- 인코더 해상도를 얻으려면 해상도 계산을 클릭합니다.

이 페이지에서 설정한 후 다음을 클릭하여 센서를 설정합니다.



센서 설정

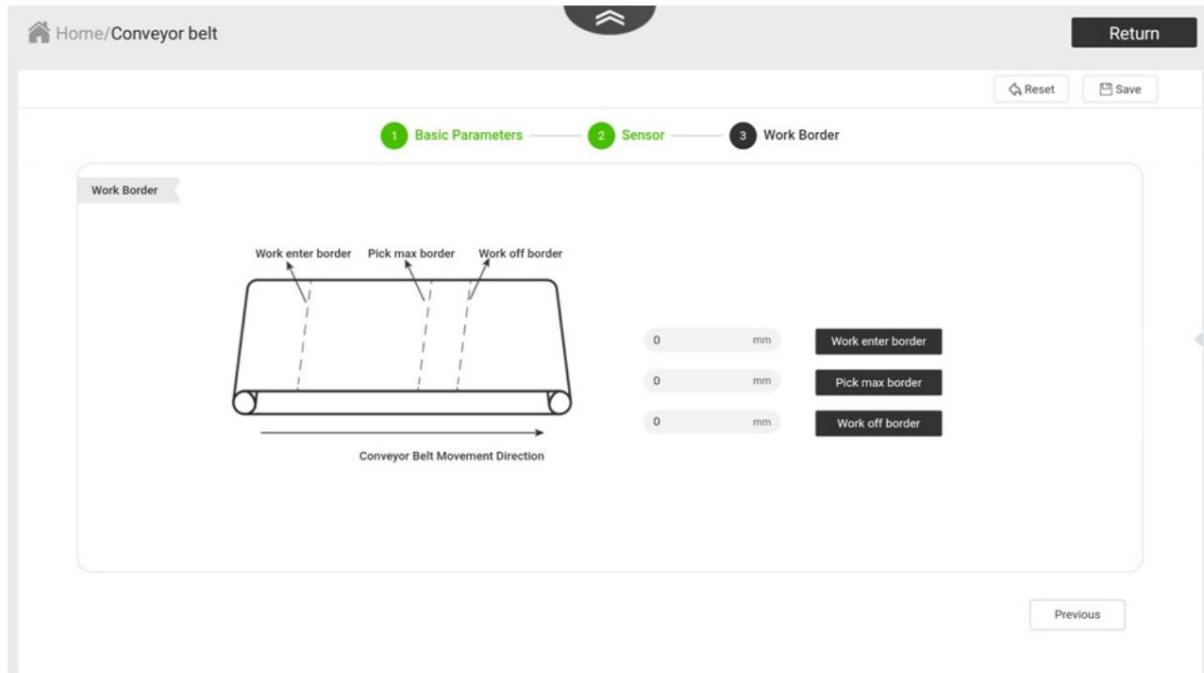
- 트리거 입력 IO: 센서가 연결된 컨트롤러의 DI 포트 인덱스를 설정합니다.
- 중복 제거 거리: 유효한 신호가 감지된 후 후속 거리 내에서 신호가 변경되면 신호가 유효하지 않은 것으로 간주되어 자동으로 제거됩니다. 값은 다음과 같이 설정할 수 있습니다.
실제 상황에 따라 2mm ~ 5mm.
- 트리거 유형: 상승 에지 트리거만 지원됩니다(구성할 수 없음).

센서 교정

센서 캘리브레이션은 워크가 컨베이어를 따라 이동할 때 좌표 오프셋을 기반으로 매 순간 사용자 좌표계에서 워크의 위치를 계산할 수 있도록 센서가 워크를 감지하는 위치를 얻는 것을 목표로 합니다.

1. 센서 설정에서 듣기 시작을 클릭하여 센서 신호 듣기를 시작합니다.
2. 컨베이어의 업스트림에 작업물을 놓습니다. 컨베이어를 제어하여 이동합니다. 센서가 작업물을 감지하고 작업물이 로봇의 작업 공간 내에 있으면 컨베이어를 멈춥니다.
3. 공작물 중심으로 로봇을 조그한 다음 가져오기를 클릭하여 현재 위치를 가져옵니다.
4. 계산을 클릭하여 센서의 위치를 구합니다.

이 페이지에서 설정한 후 다음을 클릭하여 작업 경계를 설정합니다.



작업 테두리는 컨베이어에서 로봇의 작업 영역을 설정하는 데 사용됩니다.

- 작업 진입 경계: 작업물이 이 경계를 넘은 후 로봇 팔이 작업물을 추적하고 픽업하기 시작합니다.
- Pick max border: 공작물이 경계를 넘을 때 로봇 팔이 공작물 픽업을 시작하지 않은 경우 공작물 픽업을 완료할 수 없는 것으로 간주되어 공작물을 픽업하지 않습니다. 경계는 컨베이어의 이동 속도와 실제 경험을 기준으로 설정해야 합니다. 최적의 값을 얻기 전에 여러 번 디버깅하는 것이 좋습니다.
- 워크 오프 경계: 공작물이 경계를 넘은 후 로봇 팔은 공작물 추적을 중지하고(로봇이 작업 영역 밖으로 추적하는 것을 방지하기 위해) 알람을 트리거합니다.

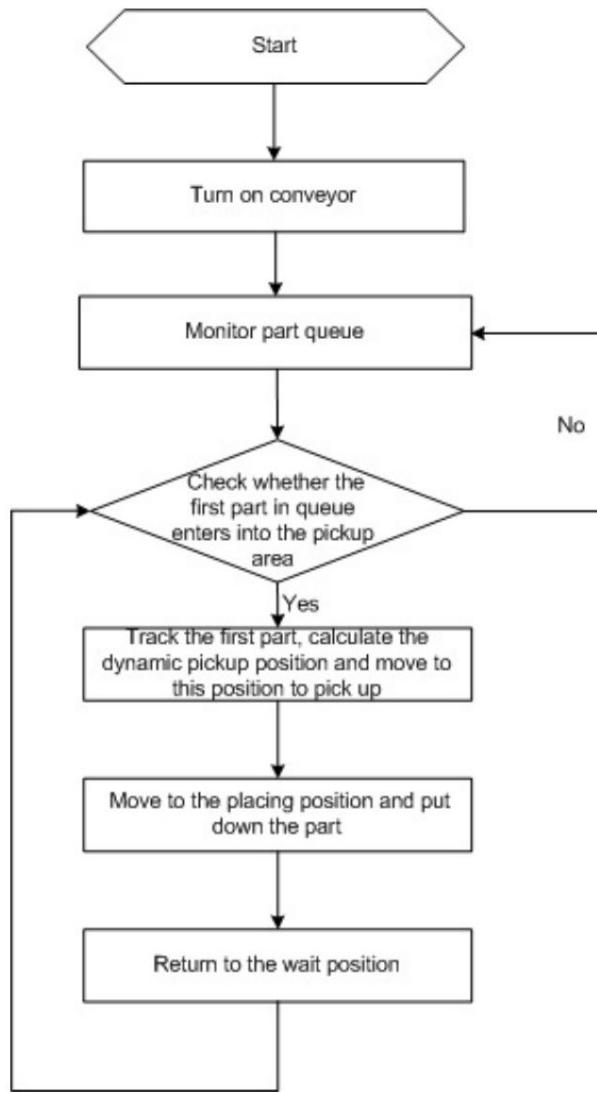
각 테두리의 해당 위치로 로봇을 조그하고 버튼을 클릭하여 값을 설정합니다.

오른쪽 상단 모서리에 있는 저장을 클릭합니다. 이제 프로젝트에서 컨베이어 구성을 사용할 수 있습니다.

예

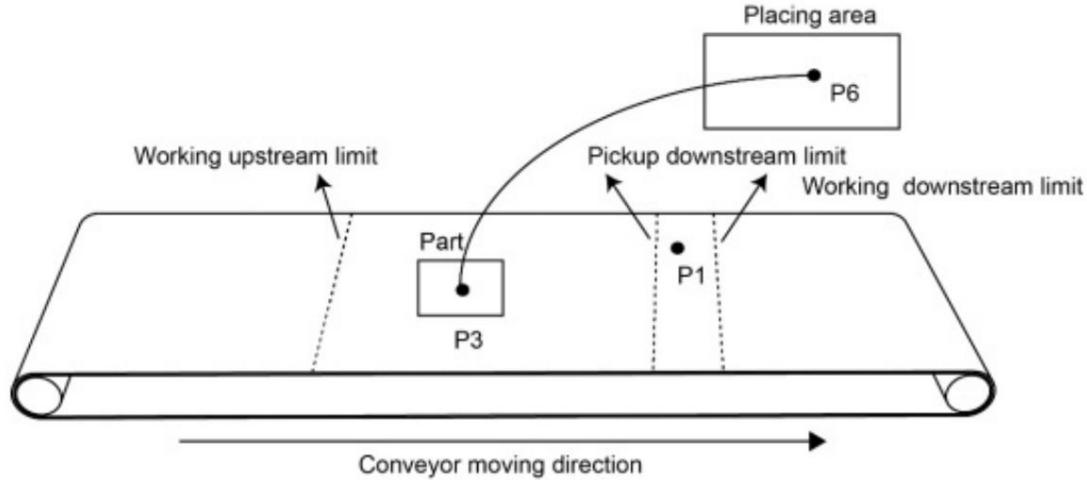
컨베이어 벨트의 매개변수를 구성한 후 컨베이어 벨트 API를 호출하여 스크립트를 편집하여 컨베이어 추적을 실현할 수 있습니다.

컨베이어 벨트 추적 프로세스는 다음과 같습니다.



이 예에서는 컨베이어 벨트를 구성할 때 설정한 사용자 좌표계 아래에 6개의 지점을 학습시켜야 합니다.

- 대기 장소: P1
- 추적 지점: P2
- 픽업 장소: P3
- 리프팅 포인트: P4
- 배치 지점 위의 지점: P5
- 배치 지점: P6



스크립트는 다음과 같습니다.

```

CnvVision(0) -----//컨베이어 시작

DO(9,0) -----//DO1 및 D를 통해 공기 펌프의 시작 및 상태 제어
O2

DO(2,0)

로컬 플래그 -----//작업물 라벨이 있는지 여부

local typeObject -----//공작물의 종류

로컬 포인트 = {0,0,0} -----//Queue 공작물 좌표

사실인 동안

Go(P1,"Speed=100 Accel=100 SYNC=1") ---//대기점, 일반적으로 워크스파 내에서 설정
CE

print("테스트")

사실인 동안

flag,typeObject,point = GetCnvObject(0,0) --//작업물이 있는지 확인
이자형. 있는 경우 중단하십시오.

if 플래그 == true이면

    부서지다

    끝

수면(20)

끝

SyncCnv(0) -----//컨베이어 벨트 동기화 및 추적 시작

```

Move(P2,"SpeedS=100 AccelS=100") -----//픽업 지점 위로

Move(P3,"SpeedS=100 AccelS=100") -----//픽업 지점

대기(100)

DO(9,1) -----//공기 펌프를 켜서 공작물을 픽업하십시오.

--DO(2,1)

대기(100)

Move(P4,"SpeedS=100 AccelS=100 SYNC=1") -----//작업물 리프트

StopSyncCnv() -----//컨베이어 추적 중지

수면(20)

Go(P5,"Speed=100 Accel=100") -----//배치 지점 위로

Go(P6,"Speed=100 Accel=100 SYNC=1") -----//포인트 배치

수면(1000)

DO(1,0) -----//공기 펌프 끄기

DO(2,0,"동기|=1")

수면(1000)

Go(P5,"속도=100 가속=100")

끝

8 모범 사례

이 장에서는 원격 I/O를 통해 로봇 팔을 제어하는 전체 프로세스를 설명합니다.

Dobot CRStudio의 다양한 기능이 조정된 방식으로 사용되는 방식을 이해합니다.

이제 다음 장면을 가정합니다. 시작 버튼을 누르면 실행 표시등이 켜집니다. 로봇팔은 피킹 지점에서 엔드 그리퍼를 통해 소재를 잡고, 목표 지점까지 이동해 소재를 풀고, 다시 피킹 포인트로 돌아와 소재를 잡는… 과정을 반복합니다.

위의 장면을 구현하려면 로봇 팔 끝에 그리퍼를 설치하고(DH 그리퍼를 예로 들면 설치 방법은 DH 그리퍼 사용 설명서 참조) 컨트롤러에 버튼과 표시기를 연결해야 합니다. I/O 인터페이스(시작 버튼이 DI11에 연결되고 정지 버튼이 DI12에 연결되고 실행 표시기가 DO11에 연결되고 알람 표시기가 DO12에 연결된다고 가정합니다. 배선은 해당 하드웨어 가이드를 참조하십시오. 로봇).

전반적인 과정

하드웨어를 설치하고 로봇 팔의 전원을 켜 후 다음과 같이 소프트웨어 작업을 수행합니다.

다음과 같습니다.

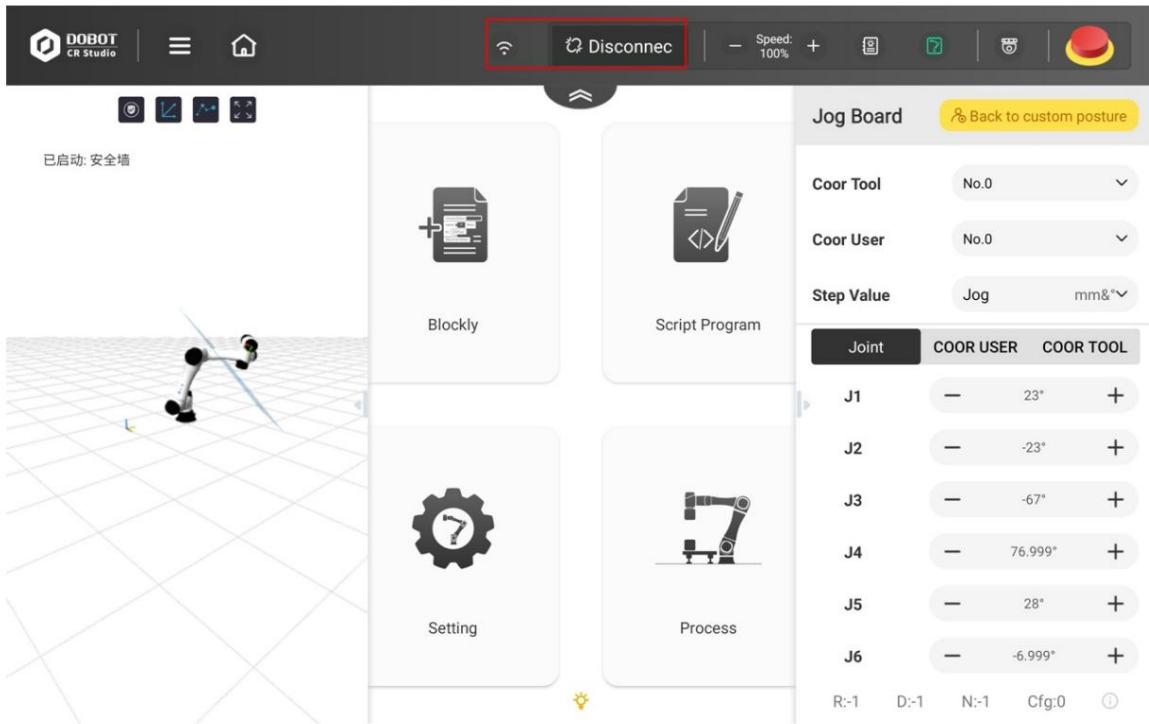
1. 로봇 연결
2. 두봇+ 플러그인 설치(선택사항)
3. 공구 좌표계 설정 및 선택 4. 프로젝트 파일 편집 5. 원
- 격 I/O 모드 구성 및 입력

절차

로봇 연결 및 활성화

로봇과의 연결에 대한 자세한 내용은 로봇과의 연결을 참조하십시오 .

1. 두봇 컨트롤러 WiFi 이름을 검색하여 연결한다. WiFi SSID는 Dobot_WIFI_XXX입니다(XXX는 베이스에 위치한 로봇 인덱스), WiFi 비밀번호는 기본적으로 1234567890입니다.
2. CRStudio 인터페이스 상단에서 로봇을 선택하고 연결을 클릭합니다.



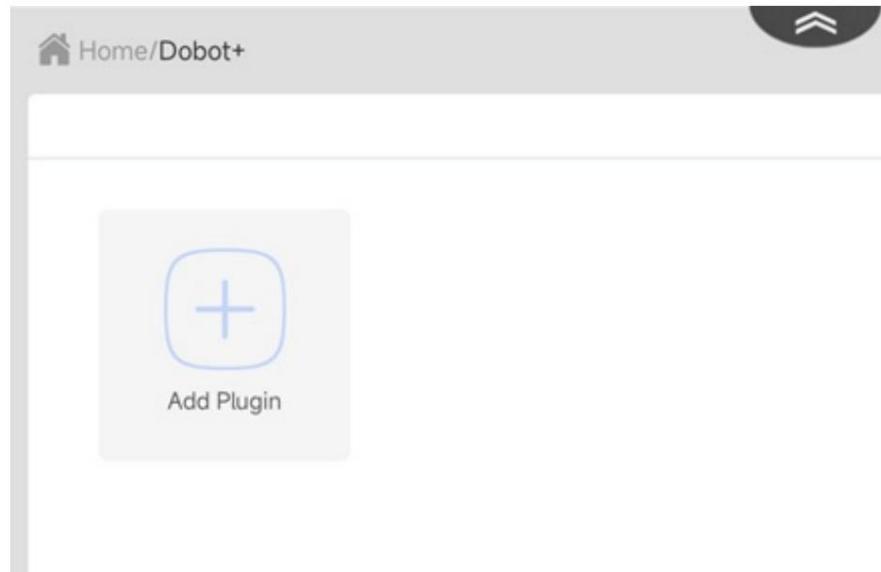
3. 활성화 버튼을 클릭하고 로드 매개변수를 설정하여 로봇을 활성화합니다.

Dobot+ 플러그인 설치(선택 사항)

두봇+ 플러그인에 대한 자세한 내용은 두봇+를 참조하십시오. 다음은 DH 플러그인을 예로 들어 설명합니다.

설치된 종료 도구에 해당 플러그인이 없으면 이 단계를 건너뛰고 후속 프로그래밍에서 I/O 명령을 통해 종료 도구를 직접 제어하십시오.

1. 모니터 > 두봇+를 클릭하여 두봇+ 패널을 엽니다.
2. 플러그인 추가를 클릭 한 후 DH 플러그인 왼쪽의 +를 클릭하면 설치가 완료됩니다.



공구 좌표계 설정 및 선택

공구 좌표계에 대한 자세한 내용은 [공구 좌표계를 참조하십시오.](#)

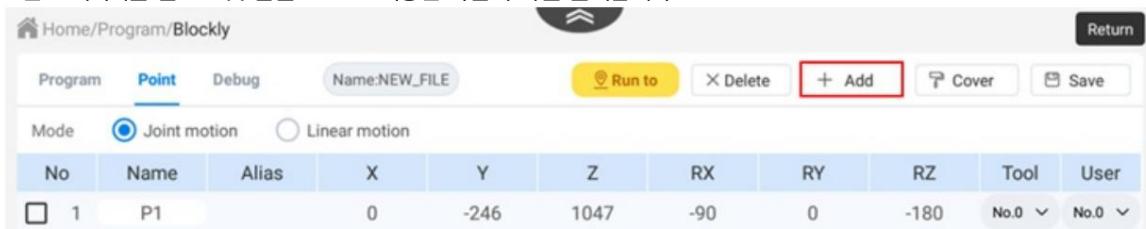
프로젝트 편집

프로그래밍에 대한 자세한 내용은 [Blockly](#) 및 Script를 참조하십시오 . 다음은 Blockly를 예로 들어 설명합니다.

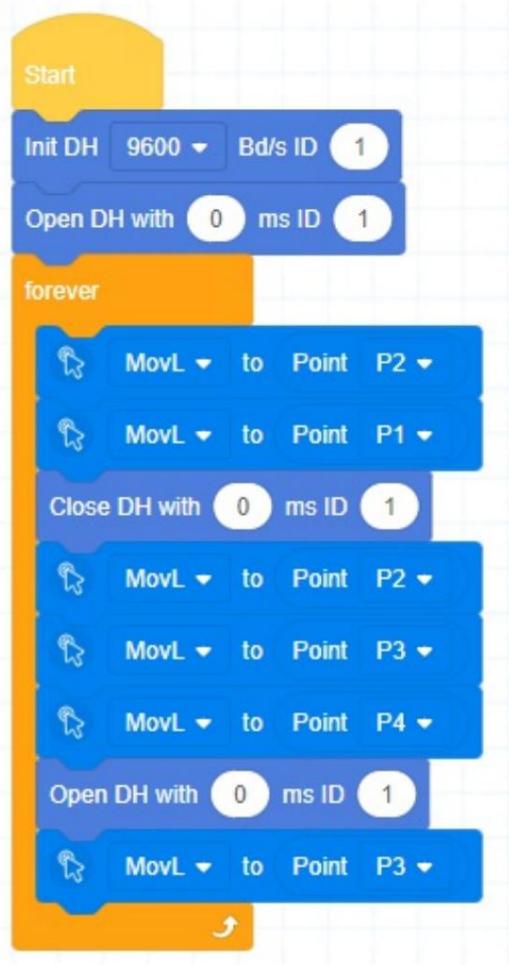
이 장의 시작 부분에서 설명한 장면을 달성하려면 4개 지점, 즉 선택 지점 P1, 전환 지점 P2(선택 지점 위), 전환 지점 P3(업로드 지점 위) 및 업로드 지점을 가르쳐야 합니다. 포인트 P4.



- 포인트 페이지를 열고 로봇 팔을 P1으로 이동한 다음 추가를 클릭합니다.



- 같은 방법으로 P2, P3, P4를 추가합니다.
- 블록을 프로그래밍 영역으로 드래그하여 재료 선택 및 언로드를 실현합니다. 그림 아래는 참조용으로 간단한 프로그램을 보여줍니다.

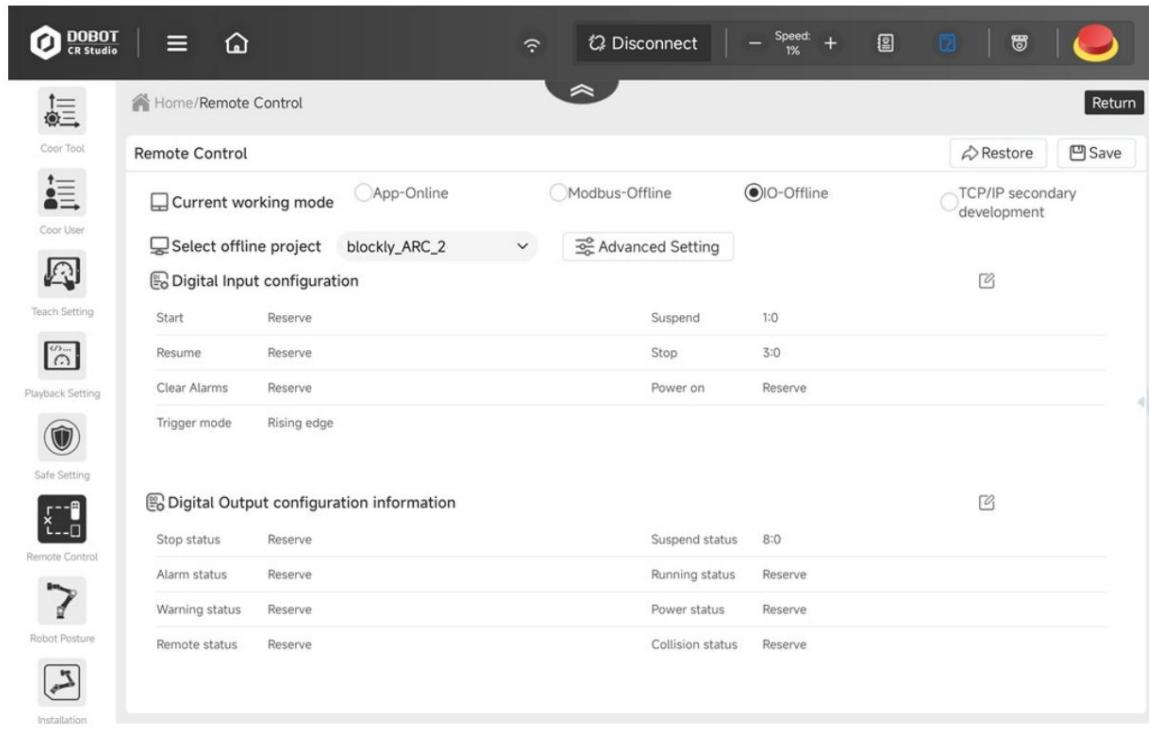


4. 프로젝트를 저장합니다.

원격 I/O 모드 구성 및 시작

원격 제어에 대한 자세한 내용은 원격 제어를 참조하세요 . 여기서는 예제 장면을 기반으로 원격 I/O 모드를 구성하고 입력하는 단계만 설명 합니다.

1. 설정 > 원격 제어 페이지를 엽니다.
2. 현재 작업 모드를 IO-오프라인 으로 설정합니다.
3. 이전에 저장한 Blockly 프로젝트를 선택합니다.
4. 클릭  이 시작 부분에서 설명한 장면에 따라 I/O 구성을 수정하려면
장.
5. 저장을 클릭하여 원격 IO 모드로 들어갑니다.



리모트 I/O 모드 진입 후 로봇팔 컨트롤러에 연결된 시작 버튼을 누르면 로봇팔이 프로젝트 실행을 시작합니다.

부록 A Modbus 레지스터 정의

Modbus 데이터에는 주로 코일 상태, 이산 입력, 입력 레지스터 및 흘링 레지스터의 네 가지 유형이 포함됩니다.

로봇 메모리 공간을 기반으로 외부 장비와 로봇 시스템 간의 데이터 상호 작용을 위한 코일, 접점(이산 입력), 입력 및 유지 레지스터의 네 가지 유형의 레지스터가 정의됩니다. 각 레지스터에는 4096개의 주소가 있습니다. 자세한 내용은 아래 설명을 참조하십시오.

1 코일 레지스터(제어 로봇)

PLC 주소	스크립트 주소(Get/SetCoils)	레지스터 유형	기능
00001	0	조금	시작
00002	1	조금	정지시키다
00003	2	조금	계속하다
00004	삼	조금	멈추다
00005	4	조금	비상 정지
00006	5	조금	알람 지우기
00007	6	조금	초기화
00051~00066	50~65	조금	기본 IO: DO1~DO16
00067~00070	66~69	조금	종료 IO: DO17~DO20
03096~04096	3095~4095	조금	사용자 정의

2 개별 입력(로봇 상태)

PLC 주소	스크립트 주소(GetInBits)	레지스터 유형	기능
10002	1	조금	정지 상태
10003	2	조금	일시 중지 상태
10004	삼	조금	실행 상태
10005	4	조금	알람 상태
10006	5	조금	예약된
10007	6	조금	충돌 상태
10008	7	조금	수동/자동 모드
10051~10066	50~65	조금	기본 IO: DI1~DI16

10067~10070

66~69

조금

종료 IO: DI17~DI20

3 입력 레지스터

PLC 주소	스크립트 주소(GetInRegs)	데이터 형식	기능
30203	202	F32	로봇 실행 위치(관절 각도 1)
30205	204	F32	로봇 실행 위치(관절 각도 2)
30207	206	F32	로봇 실행 위치(관절 각도 3)
30209	208	F32	로봇 실행 위치(관절 각도 4)
30211	210	F32	로봇 실행 위치(관절 각도 5)
30213	212	F32	로봇 실행 위치(관절 각도 6)
30243	242	F32	로봇 실행 위치(x)
30245	244	F32	로봇 실행 위치(y)
30247	246	F32	로봇 실행 위치(z)
30249	248	F32	로봇 실행 위치(a)
30251	250	F32	로봇 실행 위치(b)
30253	252	F32	로봇 실행 위치(c)

4 홀딩 레지스터(로봇과 PLC)

PLC 주소	스크립트 주소 (가져오기/SetHoldRegs)	데이터 형식	기능
40001~41281	0~1280	U16	팔레타이징
41301	1300년	U16	HMI 조그 모드로 전환
41302	1301	U16	HMI 조그 모드 전환 준비 완료
41303	1302	U16	조그 또는 스텝 모드: 관절/데카르트
41304	1303	U16	조그/스텝 선택

41305	1304년	U16	전역 속도: 백분율
41306	1305년	F32	단계 거리: mm
41308	1307	F32	단계 각도: °
41310	1309년	U16	단계 각도: °
41311	1310년	U16	사용자 좌표계 선택: 인덱스
41312	1311	U16	손 좌표계
41313	1312년	U16	매개변수 수정 알림
41314	1313	U16	조깅 시작
41315	1314년	U16	J1+/X+
41316	1315년	U16	J1-/X
41317	1316년	U16	J2+/Y+
41318	1317	U16	J2-/Y
41319	1318년	U16	J3+/Z+
41320	1319년	U16	J3-/Z
41321	1320년	U16	J4+/A+
41322	1321년	U16	J4-/A
41323	1322년	U16	J5+/B+
41324	1323년	U16	J5-/B
41325	1324년	U16	J6+/C+
41326	1325년	U16	J6-/C
41327	1326년	F32	P1(엑스)
41329	1328년	F32	P1(Y)
41331	1330년	F32	P1(지)
41333	1332년	F32	P1(R/A)
41335	1334년	F32	P1(비)
41337	1336년	F32	P1(C)
41339	1338	U16	P1(팔)
41340	1339년	U16	P1(사용자)
41341	1340년	U16	P1(공구)
41342~41551	1341~1550	F32&U16	P2~P15
41552	1551년	F32	P16(엑스)

41554	1553년	F32	P16(Y)
41556	1555년	F32	P16(Z)
41558	1557년	F32	P16(R/A)
41560	1559년	F32	P16(비)
41562	1561년	F32	P16(C)
41564	1563년	U16	P16(팔)
41565	1564년	U16	P16(사용자)
41566	1565년	U16	P16(공구)
41567	1566년	U16	포인트 저장
41568	1567년	U16	RUNTO: 이동/이동
41569	1568년	U16	RUNTO: 포인트 지수
41570	1569년	U16	런토: 시작
41571	1570년	U16	알람 지우기
42010	2009년	F32	다중 PC1(마스터) x
42012	2011년	F32	다중 PC1(마스터) y
42014	2013년	F32	다중 PC1(마스터) r
42016	2015년	F32	다중 PC1(마스터) encCount
42018	2017년	U16	멀티 PC1(마스터) 탑입
42019	2018년	U16	다중 PC1(마스터) 사용 가능
42020~42029	2019~2028	U16	예약된
42030	2029년	F32	다중 PC2(슬레이브) x
42032	2031년	F32	다중 PC2(슬레이브) y
42034	2033년	F32	다중 PC2(슬레이브) r
42036	2035년	F32	다중 PC2(슬레이브) encCount
42038	2037년	U16	다중 PC2(슬레이브) 유형
42039	2038년	U16	다중 PC2(슬레이브) 사용 가능
42040~42049	2039~2048	U16	예약된
42050	2049년	F32	다중 PC3(슬레이브) x
42052	2051년	F32	다중 PC3(슬레이브) y
42054	2053년	F32	다중 PC3(슬레이브) r
42056	2055년	F32	다중 PC3(슬레이브) encCount
42058	2057년	U16	다중 PC3(슬레이브) 유형

42059	2058년	U16	다중 PC3(슬레이브) 사용 가능
43095~44095	3095~4095	U16	사용자 정의

부록 B 블록 명령

- [B.1 빠른 시작](#)
 - [B.1.1 제어 로봇 동작](#)
 - [B.1.2 Modbus 레지스터 데이터 읽기 및 쓰기](#) [B.1.3 TCP](#)
 - [통신으로 데이터 전송](#)
 - [B.1.4 팔레타이즈](#)
- [B.2 블록 설명](#)
 - [B.2.1 이벤트](#)
 - [B.2.2 제어](#)
 - [B.2.3 연산자](#) [B.2.4](#)
 - [문자열](#)
 - [B.2.5 커스텀](#)
 - [B.2.6 입출력](#)
 - [B.2.7 모션](#)
 - [B.2.8 모션 고급 구성](#)
 - [B.2.9 모드버스](#)
 - [B.2.10 TCP](#)

빠른 시작

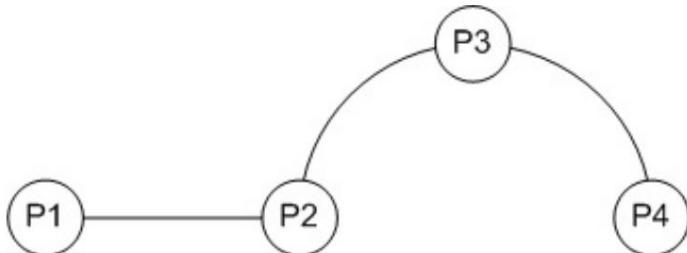
제어 로봇 움직임

장면 설명

블록 프로그래밍을 통해 로봇 팔의 움직임을 제어하는 방법을 경험하기 위해 아래와 같은 간단한 로딩 및 언로딩 장면을 가정할 수 있습니다.

로봇 팔은 P1에서 재료를 집고, P2까지 선형으로 이동하고, P3를 통해 원호 운동을 통해 P4로 이동한 다음 다시 돌아와서 같은 방식으로 재료를 집는다. 로봇 팔은 전체 프로세스를 반복합니다.

타입스.



이 장면을 달성하려면 네 가지 포인트를 가르쳐야 합니다.

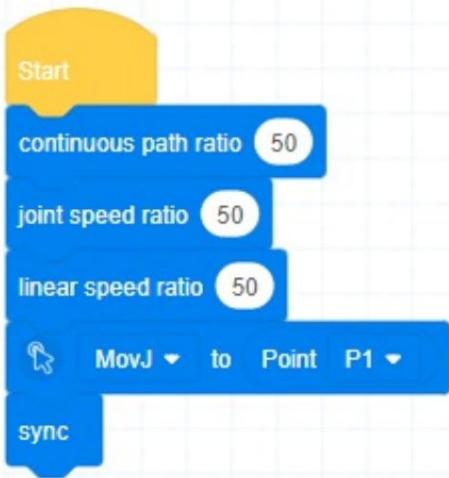
그런 다음 다음을 가정합니다.

- 그리퍼는 로봇 암의 끝에 설치되었으며 도구 DO1에 의해 제어됩니다(켜짐: 그리퍼가 잡기 위해 트리거, 꺼짐: 그리퍼가 해제되도록 트리거).
- 제어 캐비닛 DI1에 연결된 피킹 지점에 센서가 설치되었습니다(켜짐: 재료 있음, 꺼짐: 재료 없음).
- 외부 알람 장치가 제어 캐비닛 DO1에 연결됩니다(켜짐: 알람 트리거).

디버깅 중 모니터링 페이지를 통해 DI를 시뮬레이션하고 DO를 모니터링할 수 있습니다.

프로그래밍 단계

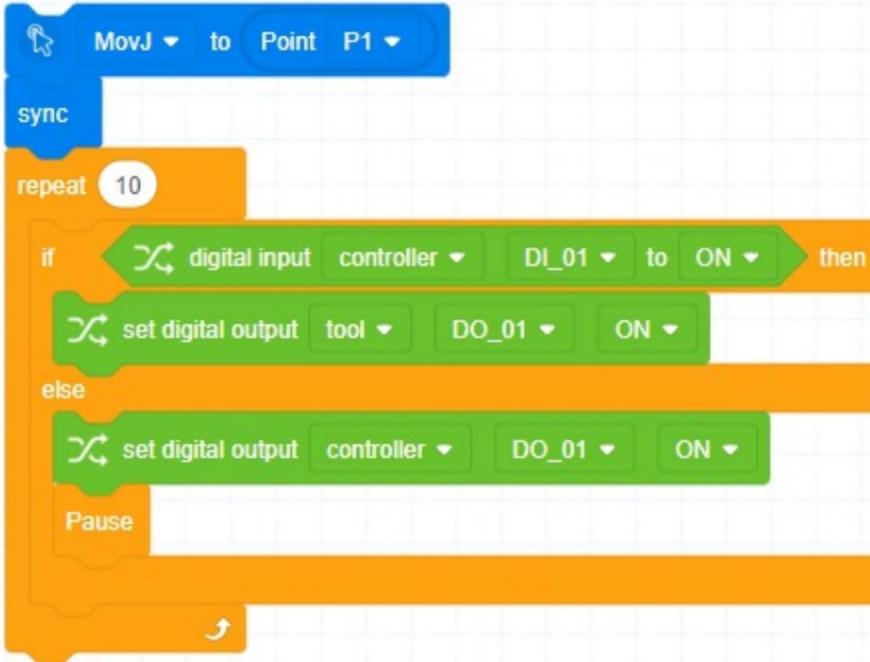
1. 로봇 팔의 모션 매개변수를 설정합니다. 다음은 단지 예일 뿐입니다. 실제에 따라 설정하십시오
요구 사항.
2. 관절 동작을 통해 로봇 팔을 P1으로 이동합니다. 동기화 명령을 설정하여 로봇 팔이 실행되도록 합니다.
이동을 완료한 후 후속 명령.



3. 10회 루프를 설정하고 그 안에 다른 블록을 삽입합니다.

4. 피킹에 자재가 있을 때 파악하도록 그리퍼를 제어하는 조건부 판단 블록 추가

자료가 없을 때 포인트를 지정하고 알람을 트리거하고 프로젝트를 일시 중단합니다.

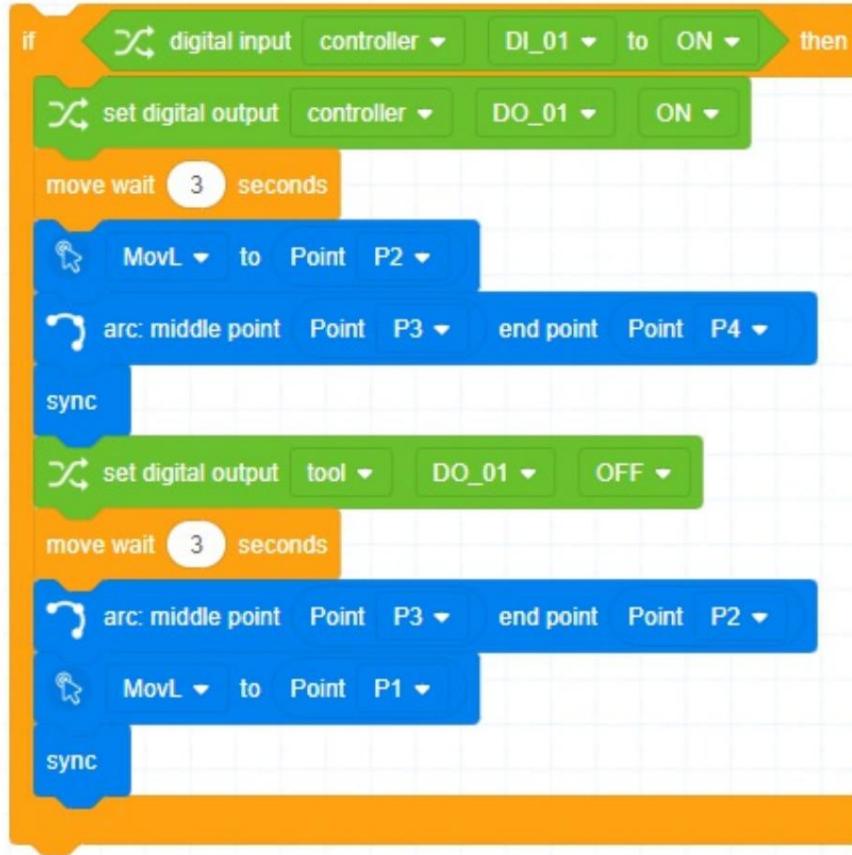


5. 파지 명령을 내리고 3초간 기다립니다. 그런 다음 재료를 성공적으로 선택한 후 로봇 팔을 이동합니다. 실제 상황에 따라 대기 시간을 설정할 수 있습니다.

6. 선형 동작을 통해 로봇 팔을 P2로 이동한 다음 원호 동작을 통해 P3을 통해 P4로 이동합니다. 동기화 설정 명령.

7. 그리퍼 해제 명령을 전달하고 3초간 기다립니다. 그런 다음 로봇 팔을 제어하여 함께 돌아갑니다.

같은 방법으로 다음 주기를 준비합니다.



Modbus 레지스터 데이터 읽기 및 쓰기

장면 설명

블록 프로그래밍을 통해 Modbus 데이터를 읽고 쓰는 방법을 경험하려면 다음 장면을 가정할 수 있습니다.

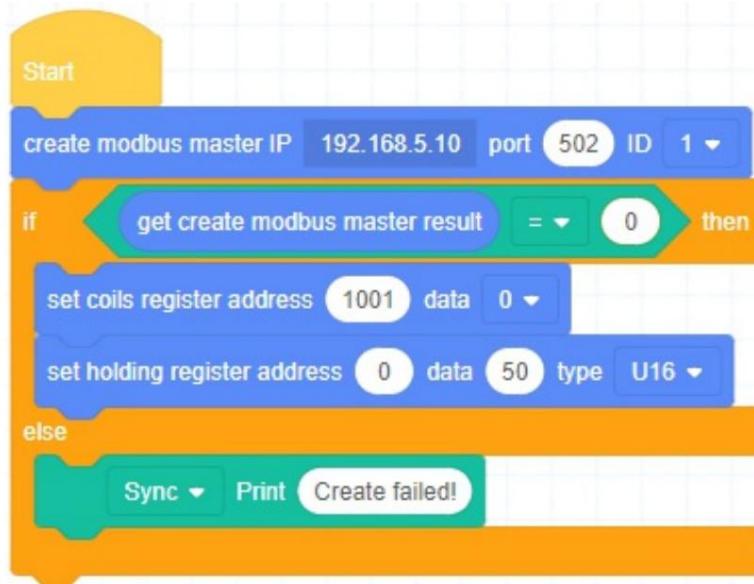
1. 코일 레지스터와 홀딩 레지스터에 각각 값을 씁니다.
2. 입력 레지스터에서 양의 정수 값을 읽고 변수에 할당합니다.
3. 위의 변수 값이 0일 때 멈추는 루프를 설정합니다.
4. 루프의 코일 레지스터에서 값을 읽습니다. 1이면 모션 명령을 실행하고 뺍니다.

위의 변수를 1로.

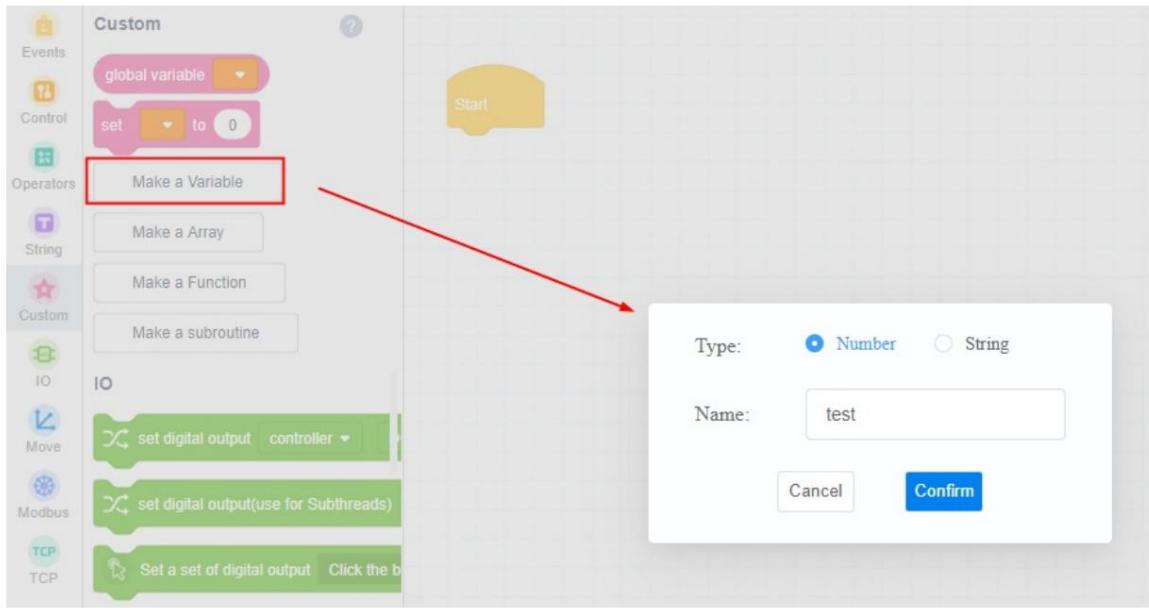
프로그래밍 단계

1. 마스터 스테이션을 생성하고 슬레이브 스테이션에 연결합니다. 슬레이브의 IP, 포트 및 ID 설정 실제 상황에 따라.
2. 마스터 스테이션이 성공적으로 생성되었는지 확인하고 "Create failed!"를 인쇄합니다. 만약 그렇다면 실패.
3. 마스터 스테이션이 성공적으로 생성되면 코일 레지스터 및 홀딩 레지스터에 값을 씁니다.

각기. 실제 조건에 따라 주소와 값을 설정하십시오.



4. 사용자 지정 변수 이름으로 새 변수를 만듭니다.



1. 입력 레지스터에서 값을 읽고 변수에 할당합니다.

2. 변수가 0일 때 멈추는 루프를 설정합니다.

3. 루프에서 지정된 코일 레지스터의 값이 1일 때 로봇 팔을 움직입니다.

(아래 그림은 하나의 모션 명령만 사용합니다. 실제 조건에 따라 루프에서 여러 모션 명령을 사용할 수 있습니다.)

이 변수를 1씩 뺍니다.

4. 루프가 끝나면 Modbus 마스터를 닫습니다.



TCP 통신으로 데이터 전송

장면 설명

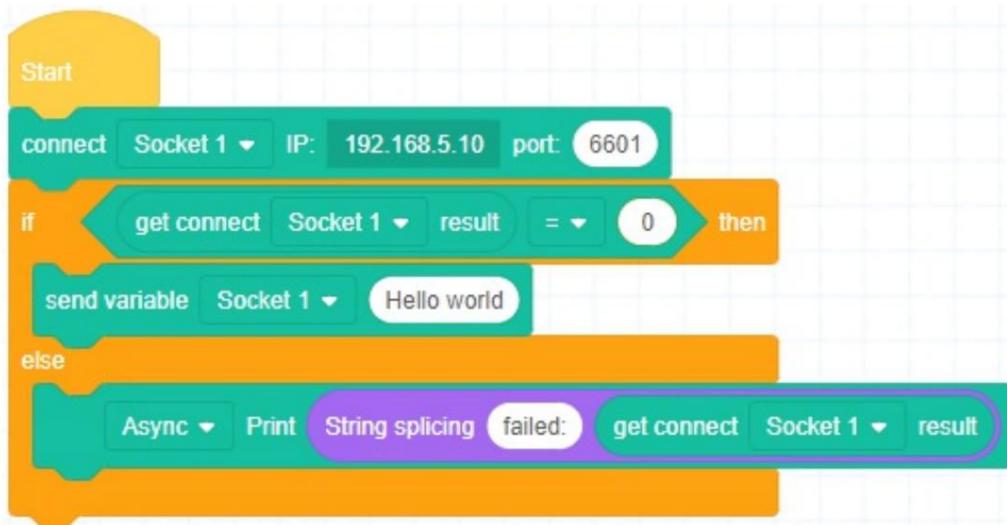
블록 프로그래밍을 통해 TCP 통신을 수행하는 방법을 경험하려면 다음을 가정합니다.

장면.

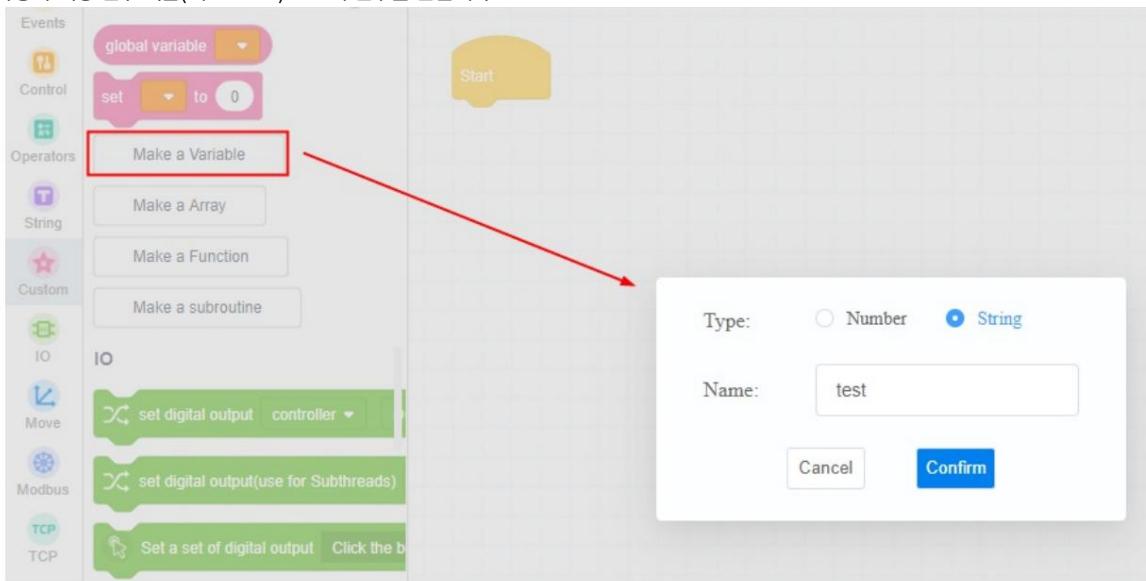
1. TCP 포트를 열고 TCP 클라이언트로 TCP 서버에 연결합니다(TCP 서버를 만들 수도 있습니다. 여기서는 클라이언트를 예로 들었습니다).
2. 문자열을 서버로 보냅니다.
3. 문자열이 "10,
15, 20, 25".
4. 로봇 팔의 루프 동작을 제어합니다. 주기 수는 의 첫 번째 쉼표 앞의 값입니다.
이 문자열.

프로그래밍 단계

1. 소켓 1을 연결하고 TCP 서버에 연결합니다. 예 따라 서버의 IP와 포트를 설정합니다.
실제 상태.
2. 연결이 성공했는지 확인합니다. 인쇄 "실패:" 더하기 연결 반환 값
연결이 실패했을 때의 결과입니다.
3. 연결에 성공하면 "Hello world"와 같은 변수를 서버로 보냅니다.



4. 사용자 지정 변수 이름(예: "test")으로 새 변수를 만듭니다.



5. 앞에 라벨 삽입

6. 변수를 보낸 결과를 판단합니다. 전송에 실패한 경우 레이블로 이동하여 다시 전송하십시오.

변하기 쉬운.

7. 전송에 성공하면 서버에서 문자열형 변수를 받아 "test"에 저장한다.



8. 변수 "test"를 배열로 변환합니다.

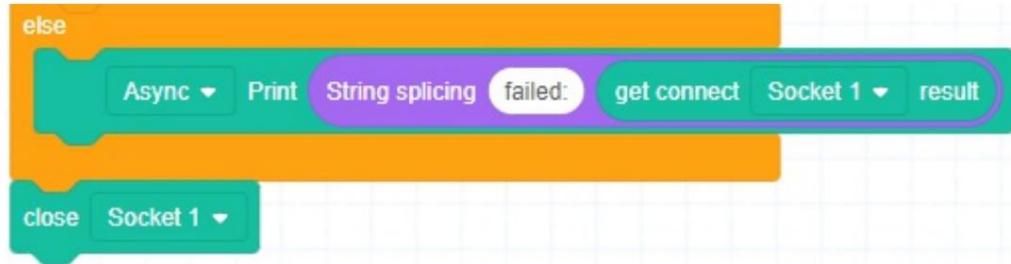
9. 위 배열의 첫 번째 요소를 루프 시간으로 사용하여 루프를 설정합니다.

10. 루프에서 로봇 팔을 이동합니다. (아래 그림의 예는 하나의 모션 명령만 사용합니다.)

실제 조건에 따라 루프에 여러 모션 명령을 추가할 수 있습니다.)



11. 소켓 1을 닫습니다.

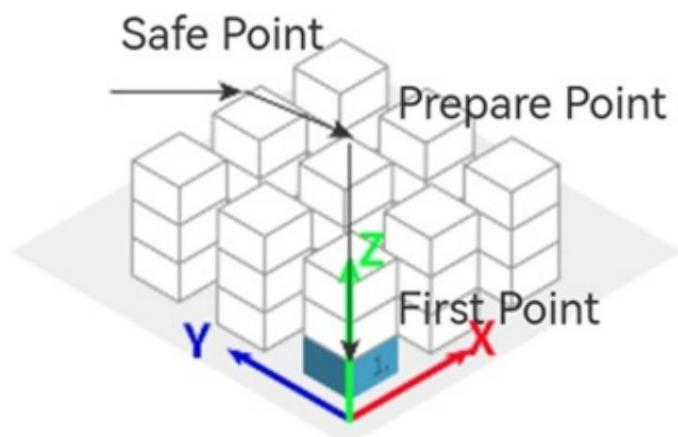


팔레이즈

장면 설명

운반할 자료가 규칙적이고 균일한 간격으로 배열된 경우 각 자료의 위치를 하나씩 티칭하는 것은 큰 오류와 낮은 효율성을 초래할 수 있습니다. 팔레이징 공정은 이러한 문제를 효과적으로 해결할 수 있습니다.

재료를 큐브에 쌓아야 한다고 가정합니다. 대상 스택 유형을 수동으로 팔레트화한 다음 관련 사항을 가르쳐야 합니다.



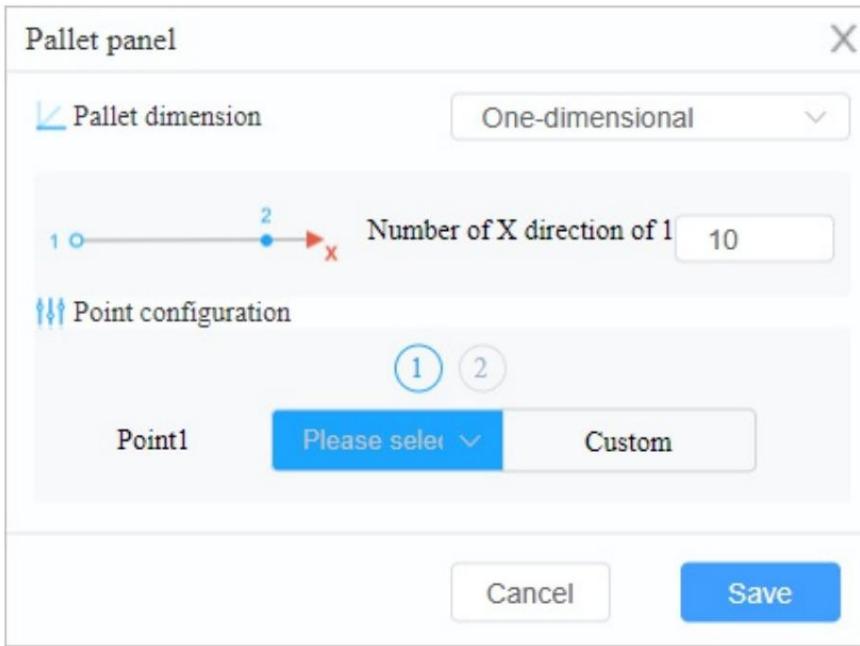
- 안전 지점(P1): 안전한 전환을 위해 스택을 조립하거나 해체할 때 로봇이 이동해야 하는 지점입니다. 피킹 포인트 위의 포인트로 설정할 수 있습니다.
- 선택 지점(P2).
- 준비점과 목표점을 일일이 가르칠 필요는 없습니다. 스택 유형 구성을 참조하십시오.

그런 다음 컨트롤러 DO1에 의해 제어되어 재료를 잡거나 해제하는 그리퍼 또는 흡입 컵이 로봇 팔 끝에 설치되었다고 가정합니다.

스택 유형 구성

팔레이트 블록을 프로그래밍 영역으로 드래그하고 블록을 클릭하여 팔레트 패널을 엽니다.





팔레트 치수

- 1차원: 재료가 일렬로 배열되어 있으며 재료의 총 수는 다음과 같습니다.
X 방향의 숫자.
- 2차원: 재료가 정사각형으로 배열되어 있으며 재료의 총 개수는 X 방향과 Y 방향의 개수를 곱한 것과 같습니다.
- 3차원: 재료가 정육면체로 쌓이고 재료의 총 수는 세 방향의 수의 곱과 같습니다.

이 섹션에서는 특정 구성을 설명하기 위해 3차원 스택을 예로 들어 설명합니다.

각 방향의 재료 수를 설정한 후 높이 오프셋을 설정합니다. 쌓기 과정에서 로봇 팔은 각 재료의 목표 지점(즉, 장면 설명의 준비 지점) 위로 지정된 높이로 이동한 다음 목표 쌓기 지점으로 하강합니다.

포인트 구성 3차원 스택을 예로 들어 큐브의 8개 모서리에 있는 재료 위치에 해당하는 8개의 포인트를 구성해야 합니다. 제어 시스템은 8개 자점과 재료 수를 통해 각 재료의 목표 지점을 자동으로 계산한 후 X -> Y -> Z 좌표축의 순서로 팔레타이징을 수행합니다.

포인트를 구성할 때 프로젝트에서 학습된 포인트를 선택하거나 사용자 정의를 클릭하여 로봇 팔의 현재 포인트를 얻을 수 있습니다. 구성된 포인트 아이콘이 녹색으로 바뀝니다.

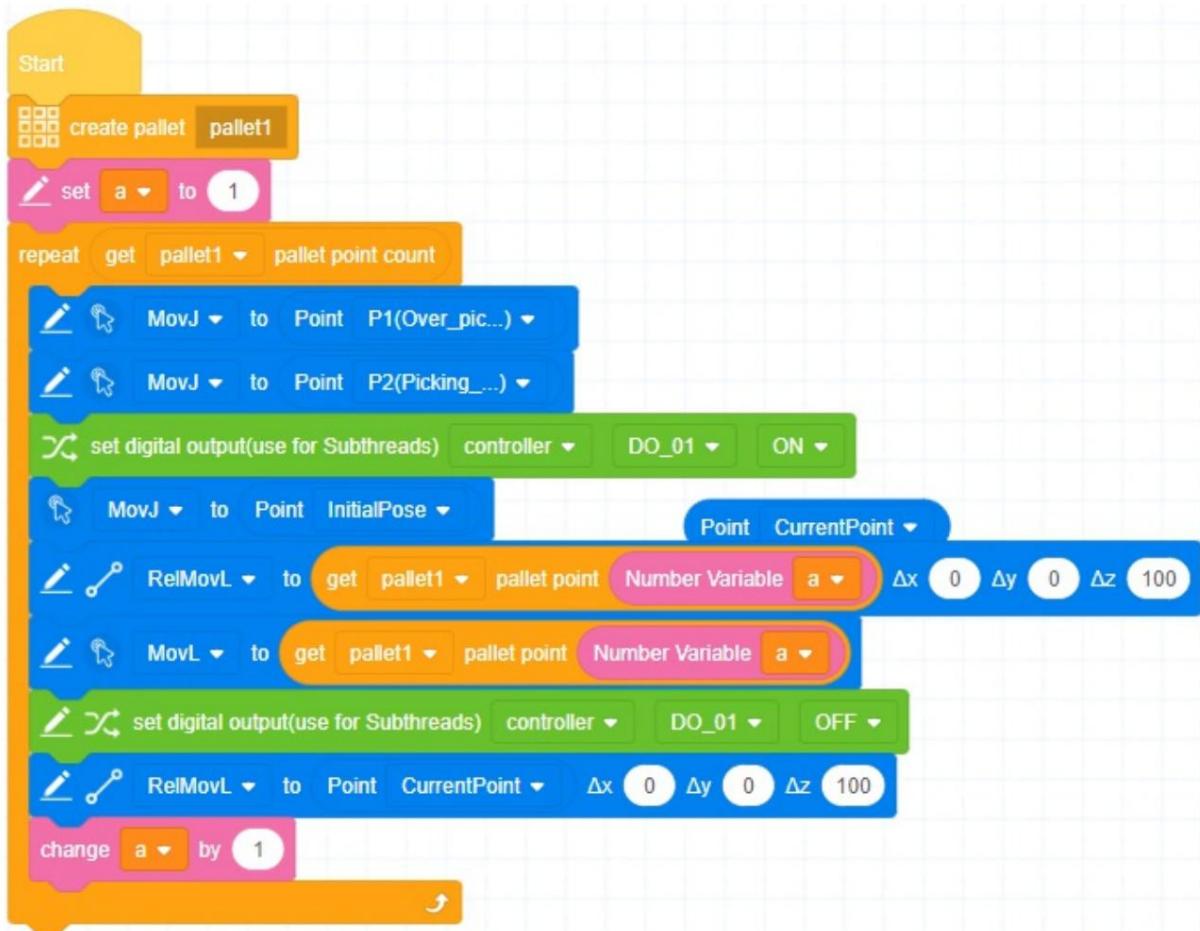
프로그래밍 단계

1. 사용자 지정 숫자 변수를 만들고 반복 횟수를 기록하는 데 사용되는 1로 설정합니다.
2. 반복 블록을 추가하고 반복 횟수를 해당 팔레트의 포인트 수로 설정합니다.
3. 모션 및 IO 블록을 추가하여 재료를 학습하도록 로봇 팔을 제어합니다.

4. 모션 및 IO 블록을 추가하여 로봇 팔을 제어하여 현재 팔레트 지점에 재료를 배치합니다.

현재 팔레트 지점의 인덱스는 현재 반복 시간과 같습니다.

5. 반복 횟수는 1씩 증가합니다.



이 섹션의 프로그램은 단순한 예일 뿐입니다. 재료가 피킹되지 않으면 후속 작업을 수행하지 않는 등 실제 조건에 따라 더 많은 IO 제어 및 판단 명령을 추가할 수 있습니다.

블록 설명

이벤트

이벤트 명령은 프로그램 실행을 시작하는 표시로 사용됩니다.

시작 명령



설명: 프로그램의 메인 스레드의 표시입니다. 새 프로젝트를 만든 후 기본적으로 프로그래밍 영역에 시작 블록이 있습니다. 시작 블록 아래에 다른 비 이벤트 블록을 배치하십시오.

프로그램.

제한 사항: 프로젝트에는 하나의 시작 블록만 있을 수 있습니다.

하위 스레드 시작 명령



설명: 프로그램의 하위 스레드 표시입니다. 하위 스레드는 주 스레드와 동시에 실행되지만 하위 스레드는 로봇 제어 명령을 호출할 수 없습니다. 변수 연산 또는 I/O 제어만 수행할 수 있습니다. 논리 요구 사항에 따라 하위 스레드를 사용할지 여부를 결정하십시오.

제한 사항: 프로젝트에는 5개의 하위 스레드만 있을 수 있습니다.

제어

제어 블록은 프로그램의 실행 경로를 제어하는 데 사용됩니다.

까지 기다리세요…



설명: 프로그램 실행이 일시 중지되고 매개변수가 true 인 경우 계속 실행됩니다.

매개변수: 다른 육각 블록을 매개변수로 사용합니다.

n회 반복



설명: 블록 내부에 다른 블록을 포함하면 포함된 블록 명령이 지정된 시간 동안 반복적으로 실행됩니다.

매개변수: 실행 반복 횟수.

지속적으로 반복



설명: 이 블록 안에 다른 블록이 내장되어 있으면 내장된 명령이



만날 때까지 반복 실행

반복 종료



설명: 실행-반복 블록 내부에 삽입하여 사용합니다. 프로그램이 이 블록까지 실행되면 바로 반복을 종료하고 실행-반복 블록 다음 블록을 실행합니다.

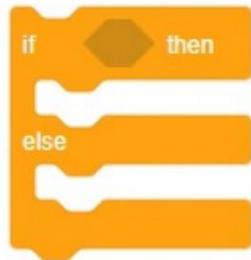
만약…그렇다면…



설명: 매개변수가 true이면 포함된 블록을 실행합니다. 매개변수가 false이면 직접 점프 다음 블록으로.

매개변수: 부울 값(참 또는 거짓)을 매개변수로 반환하는 다른 육각형 블록을 사용합니다.

만약…그렇다면…그렇지 않으면…



설명: 매개변수가 true이면 "else" 전에 포함된 블록을 실행합니다. 매개변수가 거짓이면 "else" 다음에 포함된 블록을 실행합니다.

매개변수: 부울 값(참 또는 거짓)을 매개변수로 반환하는 다른 육각형 블록을 사용합니다.

까지 반복…



설명: 매개변수가 true가 될 때까지 임베디드 블록을 반복 실행합니다.

매개변수: 부울 값(참 또는 거짓)을 매개변수로 반환하는 다른 육각형 블록을 사용합니다.

라벨 설정

Set label name as

Goto label ▾

설명: 라벨을 설정하면 다음을 통해 라벨로 이동할 수 있습니다.

매개변수: 레이블 이름은 문자로 시작해야 하며 공백과 같은 특수 문자는 사용할 수 없습니다.
사용된.

고토 라벨



Goto label **label1** ▾

설명: 프로그램이 블록으로 실행되면 지정된 레이블로 직접 점프하여 실행됩니다.
레이블 뒤의 블록.

매개변수: 레이블 이름

접기 명령



Script collapse **description**

설명: 임베디드 블록을 접습니다. 제어 효과는 없지만 프로그램을 더 읽기 쉽게 만듭니다.

Parameter: 접힌 블록을 설명하기 위한 이름

정지시키다



Pause

설명: 블록까지 실행한 후 프로그램이 자동으로 일시 중지됩니다. 제어 소프트웨어 또는 원격 제어 작업을 통해서만 계속 실행할 수 있습니다.

충돌 감지 설정



Set collision detection to **Close** ▾

설명: 충돌 감지를 설정합니다. 이 블록을 통해 설정한 충돌 감지 수준은 프로젝트가 실행 중일 때만 유효하며, 프로젝트 중지 후 이전 값으로 복원됩니다.

매개변수: 충돌 감지 감도를 선택합니다. 끄거나 1단계부터 5단계까지 선택할 수 있습니다. 단계가 높을수록 충돌 감지 감도가 높아집니다.

사용자 좌표계 설정

Set user coordinate system 0 ▾ as X 0 Y 0 Z 0 Rx 0 Ry 0 Rz 0

설명: 지정된 사용자 좌표계를 수정합니다. 수정은 프로젝트가 실행 중일 때만 유효하며 좌표계는 프로젝트가 중지된 후 이전 값을 복원합니다.

매개변수:

- 사용자 좌표계 인덱스 지정
- 수정된 사용자 좌표계의 매개변수 지정

도구 좌표계 설정

Set tool coordinate system 0 ▾ as X 0 Y 0 Z 0 Rx 0 Ry 0 Rz 0

설명: 지정된 공구 좌표계를 수정합니다. 수정은 프로젝트가 실행 중일 때만 유효하며 좌표계는 프로젝트가 중지된 후 이전 값을 복원합니다.

매개변수:

- 공구 좌표계 인덱스 지정
- 수정된 도구 좌표계의 매개변수 지정

팔레트 만들기

create pallet pallet1

설명: 팔레트의 적재 유형을 생성합니다. 자세한 내용은 [팔레타이징](#)을 참조하십시오 .

매개변수: : 팔레트 이름

팔레트 포인트 카운트 얻기

get pallet1 ▾ pallet point count

설명: 지정된 팔레트의 목표 지점 수를 얻습니다.

매개변수: : 팔레트 이름

팔레트 포인트 좌표 얻기

get pallet1 ▾ pallet point count

설명: 지정된 팔레트의 지정된 포인트 좌표를 얻습니다.

매개변수: :

- 1부터 시작하는
- 팔레트 이름 포인트 인덱스

실행 지연



설명: 프로그램이 블록까지 실행되면 지정된 시간 동안 일시 중지한 후 계속 진행합니다.

달리다.

매개변수: 프로그램의 일시정지 시간

모션 대기



설명: 모션 블록 전후에 모션 명령의 전달을 지연시키거나 이전 모션이 완료된 후 다음 명령의 전달을 지연시키는 데 사용됩니다.

매개변수: 명령을 전달하기 위한 지연 시간

시스템 시간 가져오기



설명: 시스템의 현재 시간을 가져옵니다.

반환: 현재 시스템 시간의 Unix 타임스탬프.

운영자

연산자 명령은 변수 또는 상수를 계산하는 데 사용됩니다.

산술 명령



설명: 매개변수에 더하기, 빼기, 곱하기 또는 나누기를 수행합니다.

매개변수:

- 변수 또는 상수로 두 빙칸을 모두 채우십시오. 숫자 값을 반환하는 타원형 블록을 사용하거나 빙칸에 값을 직접 입력할 수 있습니다.
- 연산자를 선택합니다.

반환: 작업 후 값

비교 명령



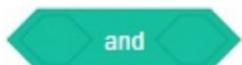
설명: 매개변수를 비교합니다.

매개변수:

- 변수 또는 상수로 두 빙칸을 모두 채우십시오. 숫자 값을 반환하는 타원형 블록을 사용하거나 빙칸에 직접 값을 입력할 수 있습니다.
- 비교 연산자를 선택합니다.

Return: 비교 결과가 참이면 참을 반환하고 거짓이면 거짓을 반환합니다.

A 및 B 명령

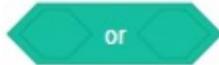


설명: 매개변수에 대한 수행 및 조작.

매개변수: 두 빙칸을 변수로 채웁니다(육각형 블록 사용).

반환: 두 매개 변수가 true이면 true를 반환하고 둘 중 하나라도 false이면 false를 반환합니다.

A 또는 B 명령



설명: 매개변수를 수행하거나 조작 합니다.

매개변수: 두 빈칸을 모두 채우십시오(육각형 블록 사용).

반환: 매개변수 중 하나라도 참이면 참을 반환하고 둘 다 거짓이면 거짓을 반환합니다.

명령이 아님

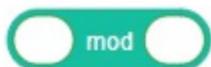


설명: 매개변수에 대한 작업을 수행 하지 않습니다 .

매개변수: 빈칸을 변수로 채웁니다(육각형 블록 사용).

반환: 매개변수가 참이면 거짓을 반환하고 매개변수가 거짓이면 참을 반환합니다 .

나머지 가져오기



설명: 나머지 매개변수를 가져옵니다.

매개변수: 변수 또는 상수로 두 공백을 모두 채웁니다. 숫자를 반환하는 타원형 블록을 사용할 수 있습니다.

값을 입력하거나 빈칸에 값을 직접 입력하세요.

반환: 작업 후 값

반올림 작업



설명: 매개변수에 반올림 연산을 수행합니다.

매개변수: 변수 또는 상수로 빈칸을 채웁니다. 숫자를 반환하는 타원형 블록을 사용할 수 있습니다.

값을 입력하거나 빈칸에 직접 값을 채웁니다.

반환: 작업 후 값

모나딕 연산



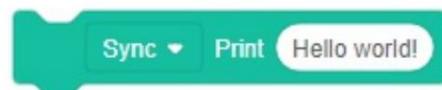
설명: 매개변수에 대해 다양한 모나딕 연산을 수행합니다.

매개변수:

- 연산자를 선택합니다.
 - 복근
 - 바닥
 - 천장 평방
 -
 - 죄
 - 코사인
 - 텐 캠칠
 - 아신
 - 아코스
 - 아탄
 - 인
 - 로
 - 예^
 - 10^{\wedge}
- 변수 또는 상수로 빙칸을 채우십시오. 숫자 값을 반환하는 타원형 블록을 사용하거나 빙 값을 직접 채울 수 있습니다.

반환: 작업 후 값

인쇄



설명: 주로 디버깅에 사용되는 매개 변수를 콘솔에 출력합니다.

매개변수:

- 동기화 또는 비동기를 선택합니다. Sync의 경우 전달된 모든 명령이 실행된 후 정보를 인쇄합니다. Async의 경우 프로그램이 블록으로 실행될 때 즉시 정보를 인쇄합니다.
- 출력할 변수 또는 상수. 타원형 블록을 사용하거나 빙칸을 직접 채울 수 있습니다.

21

문자열 명령에는 문자열 및 배열의 일반적인 기능이 포함됩니다.

문자열의 특정 위치에 있는 문자 가져오기



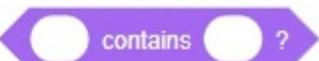
설명: 문자열의 지정된 위치에 있는 문자를 가져옵니다.

매개변수:

- 1차 매개변수: 문자열에서 반환할 문자의 위치를 지정합니다. 2차 매개변수: 문자열, 다
- 른 타원형 블록을 사용하거나 직접 채울 수 있습니다.

반환: 문자열의 지정된 위치에 있는 문자

문자열 A에 문자열 B가 포함되어 있는지 확인



설명: 첫 번째 문자열에 두 번째 문자열이 포함되어 있는지 확인합니다.

매개변수: 두 개의 문자열. 문자열을 반환하는 타원형 블록을 사용하거나 직접 채울 수 있습니다.

반환: 첫 번째 문자열에 두 번째 문자열이 포함되어 있으면 true를 반환하고 그렇지 않으면 false를 반환합니다.

두 개의 문자열을 연결



설명: 두 개의 문자열을 하나의 문자열로 연결합니다. 두 번째 문자열은 첫 번째 문자열을 따릅니다.

매개변수: 연결할 두 문자열. 문자열을 반환하는 타원형 블록을 사용하거나 직접 채울 수 있습니다.

반환: 연결된 문자열.

문자열 또는 배열의 길이 가져오기



설명: 지정된 문자열 또는 배열의 길이를 가져옵니다. 문자열의 길이는 문자열에 포함된 문자 수를 나타내고 배열의 길이는 배열에 포함된 요소 수를 나타냅니다.

매개변수: 문자열 또는 배열. 문자열 또는 배열을 반환하는 타원형 블록을 사용할 수 있습니다.

반환: 문자열 또는 배열의 길이

두 문자열 비교

String comparison

설명: ACS II 코드에 따라 두 문자열의 크기를 비교합니다.

매개변수: 비교할 두 문자열. 문자열을 반환하는 타원형 블록을 사용하거나 직접 채울 수 있습니다.

반환: 문자열 1과 문자열 2가 같으면 0, 문자열 1이 문자열 2보다 작으면 -1, 문자열 1이 문자열 2보다 크면 1을 반환합니다.

배열을 문자열로 변환

Convert array to string Array :

Separator :

설명: 지정된 배열을 문자열로 변환하고 문자열의 다른 배열 요소는 지정된 구분 기호로 구분됩니다. 예를 들어 배열이 {1,2,3}이고 구분 기호가 |이면 변환된 문자열은 "1|2|3"입니다.

매개변수:

- 문자열로 변환할 배열입니다. 문자열을 반환하는 타원형 블록을 사용할 수 있습니다.
- 변환에 사용되는 구분 기호

반환: 변환된 문자열.

문자열을 배열로 변환

Convert string to array string :

Separator :

설명: 지정된 구분 기호를 사용하여 지정된 문자열을 배열로 변환하여 문자열을 구분합니다. 예를 들어 배열이 "1|2|3"이고 구분 기호가 |이면 변환된 배열은 {[1]=1,[2]=2,[3]=3}입니다.

매개변수:

- 배열로 변환할 문자열입니다. 문자열을 반환하거나 직접 채우는 타원형 블록을 사용할 수 있습니다.
- 변환에 사용되는 구분 기호

반환: 변환된 배열.

배열의 특정 위치에 있는 요소 가져오기

Array: Access subscript: 1

설명: 지정된 배열에서 지정된 아래 첨자 위치에 있는 요소를 가져옵니다. 아래 첨자는 배열에서 요소의 위치를 나타냅니다. 예를 들어 배열 {7,8,9}에서 8의 첨자는 2입니다.

매개변수:

- 배열 값을 반환하는 타원형 블록을 사용하는 대상 배열. 지정된 요소의 첨자.
-

반환: 배열의 지정된 위치에 있는 요소의 값.

문자열의 지정된 여러 문자 가져오기

Array: Start subscript: 1 End subscript: 1 Step value: 1

설명: 지정된 배열의 지정된 아래 첨자 위치에서 여러 요소를 가져옵니다. 시작 및 끝 첨자 범위 내의 단계 값을 기준으로 요소를 가져옵니다.

매개변수:

- 배열 값을 반환하는 타원형 블록을 사용하는 대상 배열.
- 시작 첨자와 끝 첨자로 요소의 범위를 지정합니다.
- 단계 값은 요소를 얻는 빈도를 결정하는 데 사용됩니다. 1은 모든 요소를 얻는 것을 의미하고 2는 다른 모든 요소를 얻는 것을 의미합니다.

반환: 지정된 요소의 새 배열.

배열의 지정된 문자 설정

Set array elements Name: Index: 1 Value: 1

설명: 배열의 지정된 위치에 있는 요소의 값을 설정합니다.

매개변수:

- 배열 값을 반환하는 타원형 블록을 사용하는 대상 배열. 요소의 첨자.
-
- 요소의 값.

관습

사용자 지정 명령은 사용자 지정 블록을 생성 및 관리하고 전역 변수를 호출하는 데 사용됩니다.

전역 변수 호출



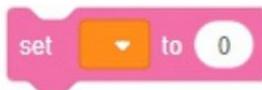
global variable

설명: 제어 소프트웨어에 설정된 전역 변수를 호출합니다.

매개변수: 전역 변수의 이름입니다.

반환: 전역 변수의 값.

전역 변수 설정



set [variable] to [value]

설명: 지정된 변수의 값을 설정합니다. 글로벌 변수 설정용 블록과 커스텀 변수 설정용 블록은 모양은 같지만 기능이 약간 다릅니다.

매개변수:

- 수정할 변수를 선택합니다.
- 수정 후 값입니다. 빈칸에 직접 값을 채우거나 다른 타원형 블록을 사용할 수 있습니다.

변수 만들기



Make a Variable

변수를 생성하려면 클릭하십시오. 변수 이름은 문자로 시작해야 하며 공백과 같은 특수 문자를 포함할 수 없습니다. 하나 이상의 변수를 생성하면 블록에 다음과 같은 변수 블록이 표시됩니다.
목록.

맞춤 숫자 변수



Number Variable

설명: 새로 생성된 커스텀 숫자 변수(기본값: nil)는 할당 후 사용을 권장합니다. 변수 이름을 수정하거나 변수 드롭다운 목록을 통해 변수를 삭제 할 수도 있습니다.

반환: 변수 값

사용자 지정 숫자 변수 값 설정



설명: 지정된 숫자 변수의 값을 설정합니다. 글로벌 변수를 설정하는 블록과 커스텀 변수를 설정하는 블록은 모양은 같지만 기능이 약간 다릅니다.

매개변수:

- 수정할 변수를 선택합니다.
- 수정 후 값입니다. 빈칸에 직접 값을 채우거나 다른 타원형 블록을 사용할 수 있습니다.

숫자 변수의 값 추가



설명: 숫자 변수에 지정된 값을 추가합니다.

매개변수:

- 수정할 변수를 선택합니다.
- 부가 가치. 빈칸에 직접 값을 채우거나 다른 타원형 블록을 사용할 수 있습니다. 음수 값 가치 하락을 말합니다.

맞춤 문자열 변수



설명: 새로 생성된 사용자 지정 문자열 변수(기본값: nil)는 할당 후 사용하는 것을 권장합니다. 변수 이름을 수정하거나 변수 드롭다운을 통해 변수를 삭제할 수도 있습니다.

목록.

반환: 변수 값

사용자 지정 문자열 변수의 값 설정



설명: 지정된 문자열 변수를 설정합니다.

매개변수:

- 수정할 변수를 선택합니다.
- 수정 후 값입니다. 공백을 문자열로 직접 채울 수 있습니다.

어레이 생성

Make a Array

사용자 지정 배열을 만들려면 클릭하십시오. 배열 이름은 문자로 시작해야 하며 공백과 같은 특수 문자를 포함할 수 없습니다. 최소한 하나의 배열을 생성하면 다음 배열 블록이 표시됩니다.
차단 목록.

맞춤 배열



설명: 새로 생성된 사용자 지정 배열은 기본적으로 빈 배열입니다. 할당 후 사용을 권장합니다. 배열의 이름을 수정하거나 배열을 삭제하려면 블록 목록에서 블록을 마우스 오른쪽 버튼으로 클릭(PC)/길게 누르십시오(Android 또는 iOS). 현재 선택된 배열의 이름을 수정하거나 다른 배열 블록의 배열 드롭다운 목록을 통해 배열을 삭제할 수도 있습니다. 배열 블록의 왼쪽에 있는 확인란은 아무 소용이 없으므로 무시해도 됩니다.

반환: 배열 값.

배열에 변수 추가



설명: 지정된 배열에 변수를 추가합니다. 추가된 변수는 배열의 마지막 항목이 됩니다.

매개변수:

- 추가할 변수입니다. 빈칸에 변수를 직접 채우거나 다른 타원형 블록을 사용할 수 있습니다.
- 수정할 어레이를 선택합니다.

배열의 항목 삭제



설명: 지정된 배열의 항목을 삭제합니다.

매개변수:

- 수정할 어레이를 선택합니다.

- 아이템 인덱스. 빈칸에 색인을 직접 채우거나 숫자를 반환하는 다른 타원형 블록을 사용할 수 있습니다. 가치.

배열의 모든 항목 삭제



delete all of **[arr v]**

설명: 배열의 모든 항목을 삭제합니다.

Parameter: 수정할 배열을 선택합니다.

배열에 항목 삽입



insert **[thing v]** at **[1 v]** of **[arr v]**

설명: 배열의 지정된 위치에 항목을 삽입합니다.

매개변수:

- 수정할 어레이를 선택합니다. 삽입 위치. 빈
- 칸에 인덱스를 직접 채우거나 반환하는 다른 타원형 블록을 사용할 수 있습니다.
- 숫자 값.
- 추가할 변수입니다. 빈칸에 변수를 직접 채우거나 다른 타원형 블록을 사용할 수 있습니다.

배열 항목 바꾸기



insert **[thing v]** at **[1 v]** of **[arr v]**

설명: 배열의 항목을 지정된 변수로 바꿉니다.

매개변수:

- 수정할 어레이를 선택합니다.
- 아이템 인덱스. 빈칸에 색인을 직접 채우거나 숫자를 반환하는 다른 타원형 블록을 사용할 수 있습니다.
- 가치.
- 교체 후 변수. 빈칸에 변수를 직접 채우거나 다른 타원형 블록을 사용할 수 있습니다.

배열 항목 가져오기



item **[1 v]** **of** **[arr v]**

설명: 배열의 지정된 항목 값을 가져옵니다.

매개변수:

- 어레이를 선택합니다.
- 아이템 인덱스. 빈칸에 색인을 직접 채우거나 숫자를 반환하는 다른 타원형 블록을 사용할 수 있습니다.
가치.

반환: 지정된 항목의 값

배열의 항목 수 가져오기

`length of arr ▾`

설명: 배열의 항목 수를 가져옵니다.

매개변수: 배열을 선택합니다.

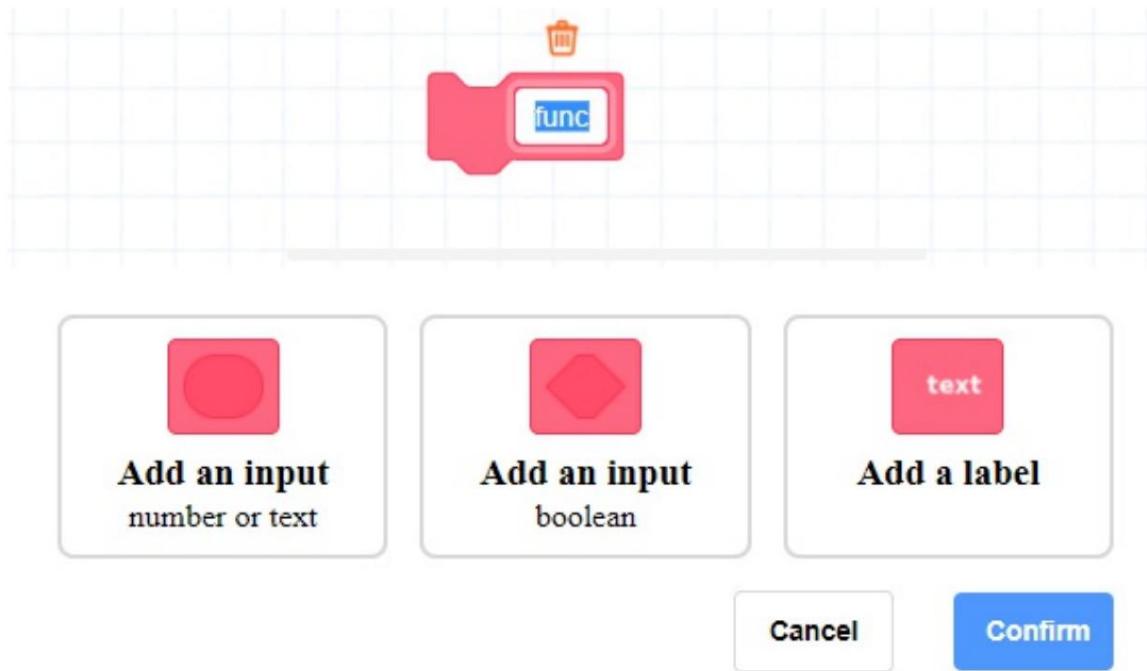
반환: 배열의 항목 수입니다.

함수 만들기

`Make a Function`

새 기능을 만들려면 클릭하십시오. 함수는 고정된 프로그램 세그먼트입니다. 특정 기능을 구현하는 블록 그룹을 기능으로 정의할 수 있습니다. 기능을 사용하고 싶을 때마다 이 기능을 호출하기만 하면 동일한 블록 그룹을 반복해서 빌드할 필요가 없습니다. 새로 생성된 함수를 선언하고 정의해야 합니다. 새 기능이 성공적으로 생성되면 해당 기능 블록이 블록 목록에 나타납니다.

1. 함수 선언



이 인터페이스에서 함수의 이름과 입력(파라미터)의 유형, 수량 및 이름을 정의해야 합니다. 함수 및 매개변수 이름에는 공백과 같은 특수 문자가 포함되지 않아야 합니다. 함수 또는 입력에 대한 주석으로 사용할 수 있는 레이블을 함수에 추가할 수도 있습니다.

1. 기능 정의

함수 선언을 완료하면 프로그래밍에서 정의 헤더 블록을 볼 수 있습니다.

영역.



함수를 정의하려면 헤더 블록 아래에 프로그래밍해야 합니다.

실제로 함수를 매개변수로 호출할 때 입력을 사용함을 나타내는 아래 블록에서 사용할 헤더 블록의 입력을 드래그할 수 있습니다.

커스텀 기능



설명: 이름과 입력 매개 변수가 사용자에 의해 정의된 사용자 지정 함수 블록은 정의된 함수를 호출하는 데 사용됩니다. 블록 목록의 블록을 마우스 오른쪽 버튼으로 클릭(PC)/길게 누르기(App)하면 함수 선언을 수정할 수 있습니다. 함수를 삭제해야 하는 경우 정의 헤더를 삭제합니다.

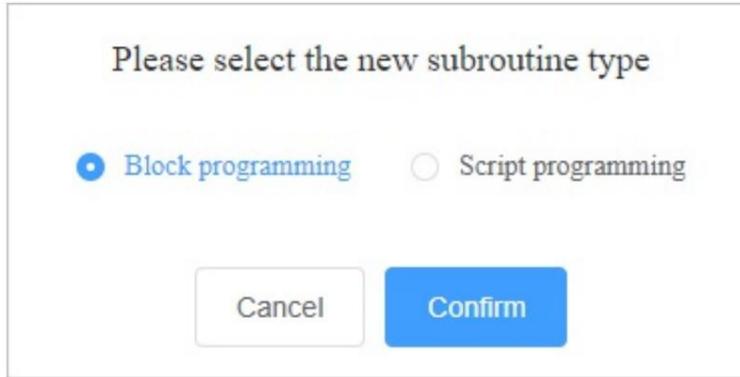
함수의 블록.

서브루틴 만들기

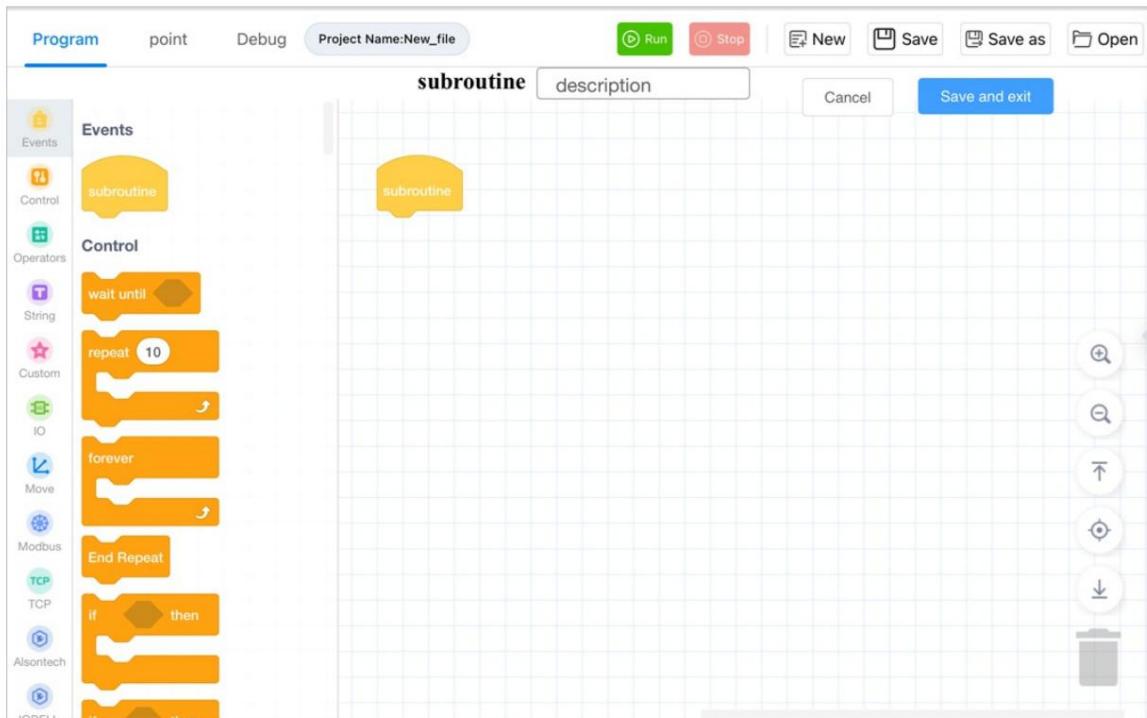
Make a subroutine

새 서브루틴을 만들려면 클릭합니다. 블록 프로그래밍은 최대 2개의 임베디드 레벨로 블록 프로그래밍 및 스크립트 프로그래밍이 될 수 있는 서브루틴 임베딩 및 호출을 지원합니다.

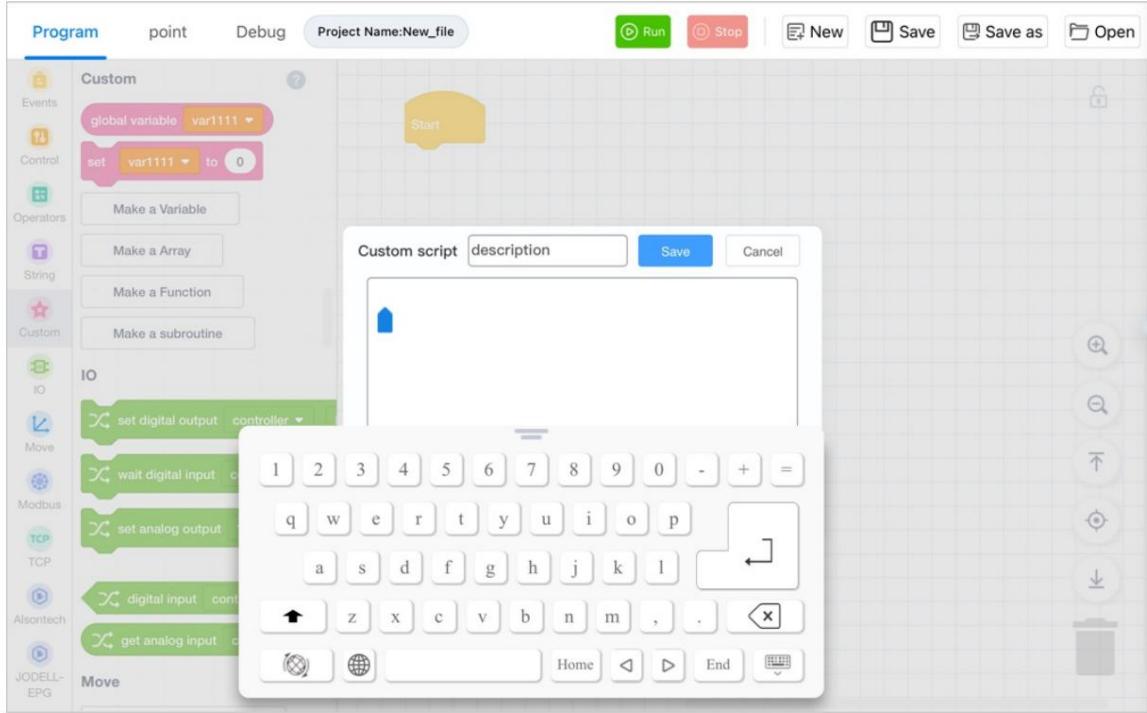
새 서브루틴이 성공적으로 생성된 후 해당 서브루틴 블록이 차단 목록.



- 블록 프로그래밍을 선택하면 서브루틴 블록 프로그래밍 페이지가 표시됩니다. 서브루틴 설명을 설정하고 서브루틴을 작성할 수 있습니다.

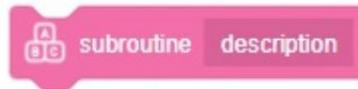


- 스크립트 프로그래밍을 선택하면 서브루틴 스크립트 프로그래밍 창이 나타납니다. 서브 루틴 설명을 설정하고 서브 프로그램을 작성할 수 있습니다.



서브루틴

- 차단 서브루틴



- 스크립트 서브루틴

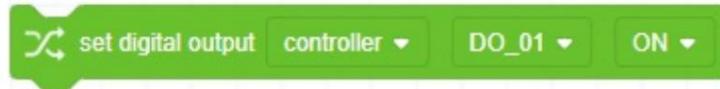


설명: 서브루틴 생성 시 사용자가 정의한 서브루틴 블록은 저장된 서브루틴을 불러오기 위해 사용됩니다. 차단 목록에 있는 차단을 우클릭(PC)/길게 누르기(앱)로 수정할 수 있습니다.
또는 하위 루틴을 삭제하십시오.

IO

IO 블록은 로봇 암의 IO 터미널의 입력 및 출력을 관리하는 데 사용됩니다. 입력 및 출력 포트의 값 범위는 로봇 암의 해당 터미널 수에 따라 결정됩니다. 해당 로봇 암의 하드웨어 가이드를 참조하십시오.

디지털 출력 설정



설명: 디지털 출력 포트의 ON/OFF 상태를 설정합니다.

매개변수:

- 컨트롤러 및 도구를 포함하여 DO 포트의 위치를 선택합니다.
- DO 포트 인덱스를 선택합니다.
- 출력 상태 선택(ON 또는 OFF)

디지털 출력 설정(서브 스레드용)



설명: 디지털 출력 포트의 ON/OFF 상태를 설정합니다. 서브에서 설정할 때 이 블록을 사용하세요.
실.

매개변수:

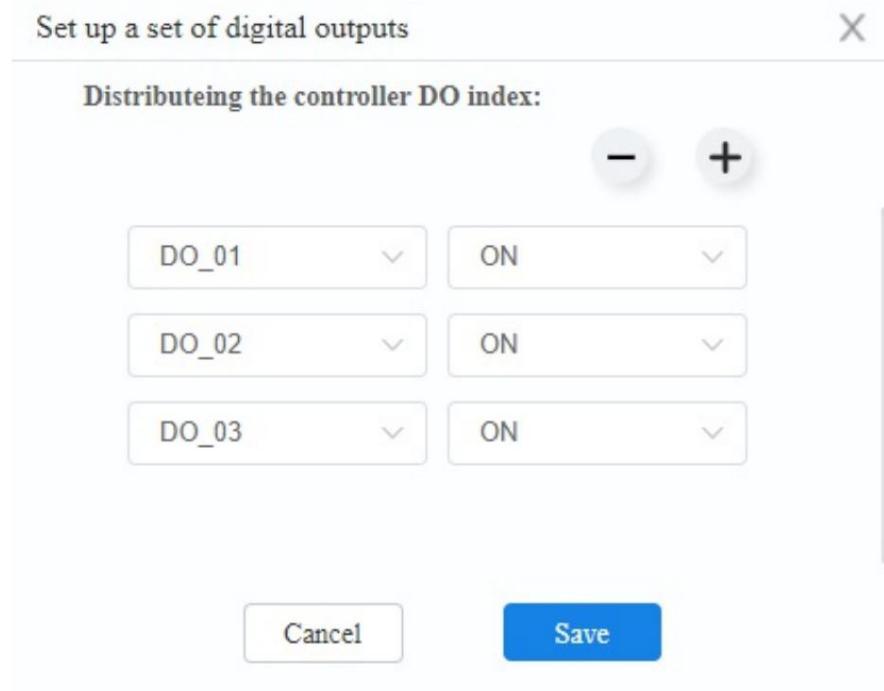
- 컨트롤러 및 도구를 포함한 DO 포트의 위치 선택
- DO 포트 인덱스 선택
- 출력 상태 선택(ON 또는 OFF)

디지털 출력 그룹 설정



설명: DO 그룹을 설정합니다. 블록을 프로그래밍 영역으로 드래그하고 클릭하여 설정할 수 있습니다.

매개변수:



- + 또는 -를 클릭하여 DO 수를 늘리거나 줄입니다.
- DO 포트 인덱스 선택
- 출력 상태 선택(ON 또는 OFF)

디지털 입력 대기



설명: 지정된 DI가 조건을 충족할 때까지 기다리거나 후속 블록 명령을 실행하기 전에 시간 초과를 기다립니다.

매개변수:

- 컨트롤러 및 도구를 포함한 DI 포트의 위치를 선택합니다.
- DI 포트 인덱스를 선택합니다.
- 대기 시간 초과 상태(ON 또는 OFF)를 선
- 택합니다(0은 조건이 충족될 때까지 대기함을 의미함).

아날로그 출력 설정



설명: 아날로그 출력 포트 값을 설정합니다.

매개변수:

- 아날로그 출력 포트 인덱스를 선택합니다.
- 아날로그 출력 값. 빈칸에 값을 직접 입력하거나 숫자 값을 반환하는 타원형 블록을 사용할 수 있습니다.

디지털 입력 상태 결정



설명: 지정된 DI의 현재 상태가 조건을 충족하는지 여부를 결정합니다.

매개변수:

- 컨트롤러 및 도구를 포함한 DI 포트의 위치를 선택합니다.
- DI 포트 인덱스를 선택합니다.
- 참으로 간주되는 상태를 선택하십시오.

Return: 지정한 DI의 현재 상태가 조건을 만족하면 true를, 그렇지 않으면 return 거짓.

아날로그 입력 받기



설명: 아날로그 입력 값을 가져옵니다.

매개변수:

- 컨트롤러 또는 도구를 포함하여 아날로그 입력 위치를 선택합니다.
- 포트 인덱스를 선택합니다.

반환: 아날로그 입력 값

동작 명령

모션 명령은 로봇 팔의 움직임을 제어하고 모션 관련 설정을 하기 위해 사용됩니다.

매개변수.

모션 블록은 모두 비동기 명령입니다. 즉, 명령이 성공적으로 전달된 후 로봇이 현재 이동을 완료할 때까지 기다리지 않고 다음 명령이 실행됩니다. 후속 명령을 실행하기 전에 전달된 명령이 실행될 때까지 기다려야 하는 경우 동기화 명령을 사용할 수 있습니다.

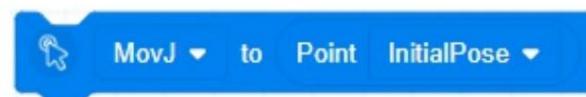
포인트 매개변수는 프로젝트의 "포인트" 페이지에 추가된 후 여기에서 선택할 수 있습니다. 모션 블록은 기본 변수 블록을 끌어서 데카르트 점 좌표를 반환하는 다른 타원형 블록으로 대체하는 것도 지원합니다.

고급 구성

[Advanced configuration](#)

미리 설정된 모션 블록이 프로그래밍 요구 사항을 충족하지 못하는 경우 고급 구성을 통해 로봇 모션을 제어하는 블록을 생성할 수 있습니다. 생성된 블록이 프로그래밍 영역에 나타납니다. 자세한 내용은 [모션 고급 구성](#)을 참조하십시오.

목표지점으로 이동

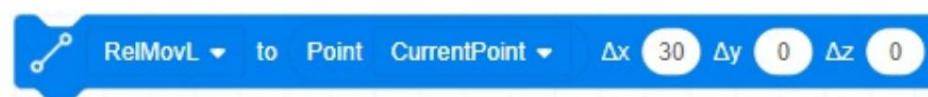


설명: 로봇이 현재 위치에서 목표 지점으로 이동하도록 제어합니다. 블록을 프로그래밍 영역으로 드래그한 후 더블 클릭하여 고급 구성을 수행합니다. 자세한 내용은 [모션 고급 구성](#)을 참조하십시오.

매개변수:

- 관절 모션(MovJ) 및 선형 모션(MovL)을 포함한 모션 모드를 선택합니다. 관절 동작의 경우 궤적의 비선형이며 모든 관절이 동시에 동작을 완료합니다. 목표 지점
-

대상점으로 이동(오프셋 포함)



설명: 로봇이 현재 위치에서 오프셋 후 목표 지점으로 이동하도록 제어합니다. 현재 지점을 목표 지점으로 설정할 수 있습니다.

매개변수:

- 상대 관절 모션(RelMovJ) 및 상대 선형 모션(RelMovL)을 포함한 모션 모드를 선택합니다. 직교 좌표계에서 목표점을 기준으로 X축, Y축 및
- Z축 방향으로
- 목표점 오프셋. 단위: 밀리미터

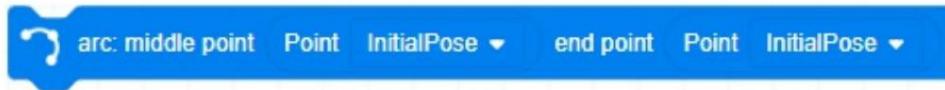
조인트 오프셋 동작



설명: 현재 위치에서 지정된 오프셋을 이동하도록 로봇을 제어합니다.

매개변수: 조인트 오프셋. 단위: °

아크 모션

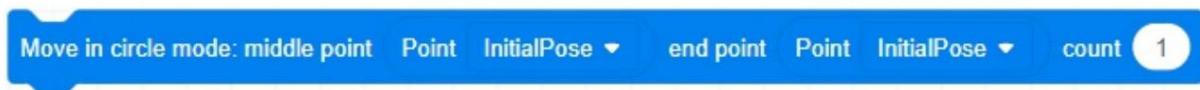


설명: 직교 좌표계에서 아크 보간 모드로 로봇이 현재 위치에서 목표 위치로 이동하도록 제어합니다. 현재 위치의 좌표는 중간점과 끝점으로 결정되는 직선 상에 있지 않아야 합니다.

매개변수:

- 중간점은 호를 결정하는 중간점입니다.
- 종점은 목표 지점입니다.

서클 모션



설명: 로봇팔이 전회전 보간 모드에서 현재 위치에서 이동하고 지정된 수의 원을 이동한 후 현재 위치로 돌아가도록 제어합니다. 현재 위치의 좌표는 중간점과 끝점으로 결정되는 직선 상에 있지 않아야 합니다.

매개변수:

- 중간점은 전체 원을 결정하는 중간점입니다.
- 끝점은 전체 원을 결정하는 데 사용됩니다.

- 원 이동을 위한 원의 수를 입력합니다. 값 범위: 1~999.

궤적 재생

trajectory playback ▾

speed Uniform ▾

설명: 궤적을 재생하도록 로봇을 제어합니다. 궤적은 궤적 재생 과정에서 기록되어야 합니다.

매개변수:

- 궤적 파일을 선택합니다.
- 재생 시 이동 속도 선택:
 - 등속 0.25배속
 - 0.5배속 1배
 - 속 2배속
 -
 -

동기화 명령

sync

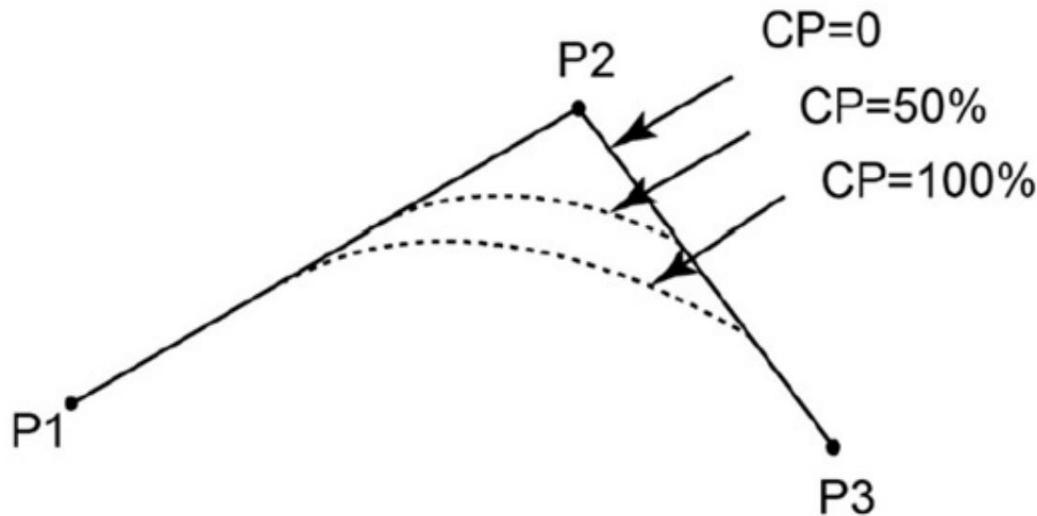
설명: 프로그램이 이 명령으로 실행되면 로봇 팔이 이전에 전달된 모든 명령을 실행하기를 기다린 다음 후속 명령을 계속 실행합니다.

CP 비율 설정

continuous path ratio

50

설명: 동작 중 연속 경로 비율, 즉 로봇이 시작점에서 중간점을 거쳐 끝점으로 이동할 때 중간점을 직각으로 통과하는지 곡선으로 통과하는지 아래와 같이 설정합니다.



매개변수: 연속 경로 비율. 값 범위: 0~100.

관절 속도 비율 설정

joint speed ratio 50

설명: 관절 운동의 속도 비율을 설정합니다.

매개변수: 관절 속도 비율, 범위: 0~100. 실제 로봇 속도 = 블록에 설정된 백분율 × 재생 설정 속도 × 전체 속도 비율.

관절 가속 비율 설정

joint accel ratio 50

설명: 관절 운동의 가속 비율을 설정합니다.

매개변수: 관절 가속 비율, 범위: 0~100. 실제 로봇 가속 = 블록에 설정된 백분율 × 재생 설정의 가속 × 전체 속도 비율.

선형 속도 비율 설정

set linear speed percentage 10 %

설명: 직선운동과 원호운동의 속도비를 설정합니다.

매개변수:

- 선형 및 아크 속도 비율(값 범위: 0~100). 실제 로봇 속도 = 블록에 설정된 백분율 × 재생 설정 속도 × 전체 속도 비율.

선형 가속 비율 설정

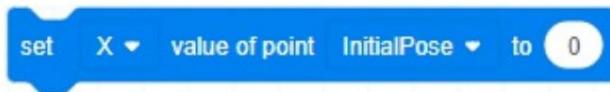


설명: 직선운동과 원호운동의 가속비를 설정합니다.

매개변수:

- 선형 및 아크 가속 비율(값 범위: 0~100). 실제 로봇 가속 = 블록에 설정된 백분율 × 재생 설정의 가속 × 전체 속도 비율.

좌표 수정

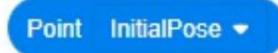


설명: 지정된 데카르트 좌표축에서 지정된 점의 값을 수정합니다.

매개변수:

- 점을 선택합니다.
- 좌표축을 선택합니다.
- 좌표 값을 설정합니다.

좌표 얻기



설명: 데카르트 좌표계에서 지정된 점의 좌표를 가져옵니다.

매개변수: 좌표를 구할 점을 선택합니다.

반환: 지정된 점의 데카르트 좌표

지정된 축의 좌표 얻기



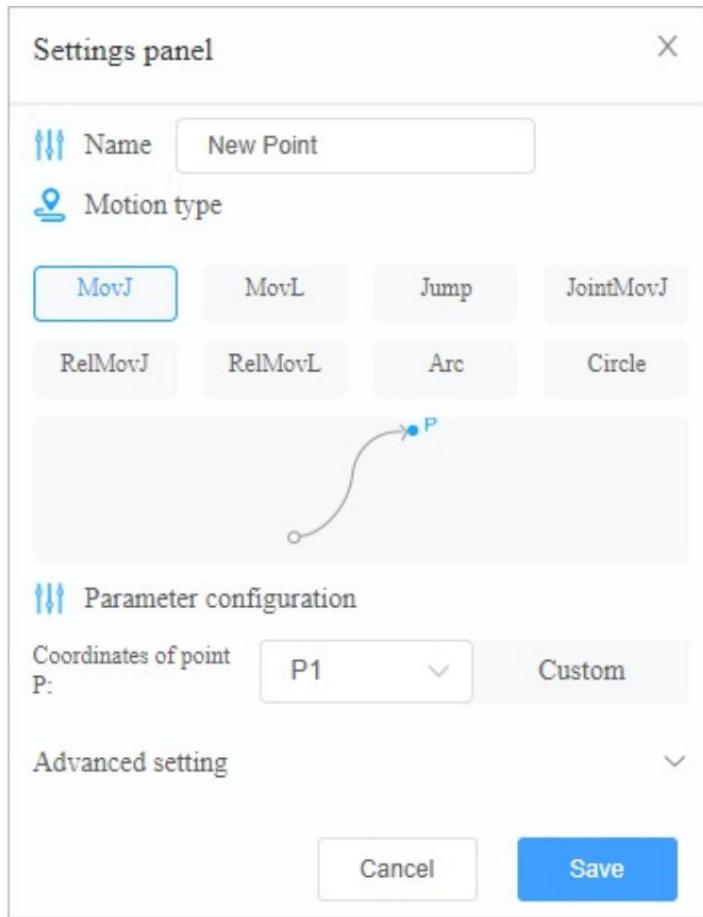
설명: 지정된 데카르트 좌표축에서 지정된 지점의 값을 가져옵니다.

매개변수:

- 좌표 값을 가져올 점을 선택합니다.
- 좌표 치수를 선택합니다.

반환: 지정된 데카르트 좌표축의 값

모션 고급 구성

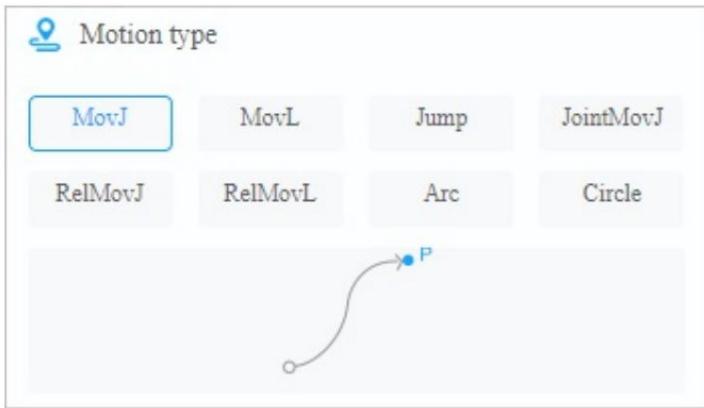


고급 구성은 로봇의 움직임을 제어하는 블록을 만듭니다. 구성에는 블록 이름, 동작 모드 및 동작 매개변수가 포함됩니다. 서로 다른 동작 모드는 구성할 동작 매개변수가 다릅니다.

실제 로봇 속도/가속 = 명령에 설정된 비율 × 재생 설정의 속도/가속 × 전체 속도 비율.

MovJ

동작 모드: 관절 보간 모드에서 직교 좌표계 아래 현재 위치에서 목표 위치로 이동합니다.



기본 설정:

P: 포인트 페이지에 추가되거나 이 페이지에서 정의된 후 여기에서 선택할 수 있는 타겟 포인트.

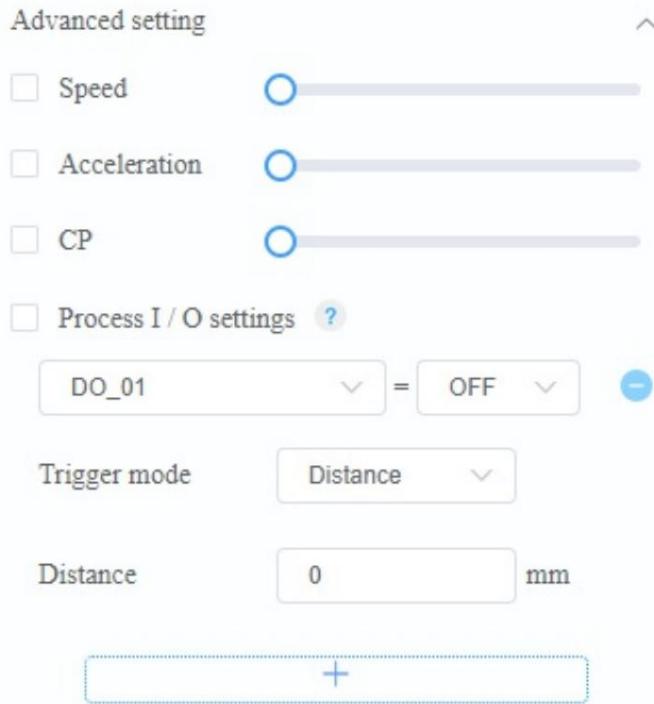
Parameter configuration

Coordinates of point P:	P1	Custom	
X	-0.000	Y -247.528	Z 1050.506
RX	-90.000	RY 0.000	RZ 180.000
ARM1,1,-1,-1	USER 0	TOOL 0	
Get coordinates			

고급 설정:

필요에 따라 고급 매개변수를 선택하고 구성합니다.

- 속도: 속도 비율, 범위: 1~100.
- 가속(Accel): 가속 비율, 범위: 1~100.
- CP: 움직이는 연속 경로 설정, 범위: 0~100. 자세한 내용은 이 섹션 끝에 있는 연속 경로(CP)를 참조하십시오.
- 프로세스 I/O 설정: 로봇 팔이 지정된 거리 또는 백분율로 이동하면 지정된 DO가 트리거됩니다. 거리가 양수이면 시작점에서 떨어진 거리를 나타냅니다. 거리가 음수이면 목표 지점에서 떨어진 거리를 나타냅니다. 아래의 "+"를 클릭하여 프로세스 IO를 추가하고 오른쪽의 "-"를 클릭하여 해당 프로세스 IO를 삭제할 수 있습니다.



MovL

모션 모드: 선형 보간 모드에서 현재 위치에서 직교 좌표계 아래의 목표 위치로 이동합니다.



기본 설정: P: 포인트 페이지에 추가된 후 여기에서 선택하거나 이 페이지에서 정의할 수 있는 대상 포인트.

 Parameter configuration

Coordinates of point P:

X	-0.000	Y	-247.528	Z	1050.506
RX	-90.000	RY	0.000	RZ	180.000
ARM1,1,-1,-1	USER	0	TOOL	0	

고급 설정:

필요에 따라 고급 매개변수를 선택하고 구성합니다.

- 속도: 속도 비율, 범위: 1~100.
- 가속(Accel): 가속 비율, 범위: 1~100.
- CP: 움직이는 연속 경로 설정, 범위: 0~100. 이 섹션의 끝에 있는 연속 경로(CP)를 참조하십시오.
[자세한 내용은](#)
- 프로세스 I/O 설정: 로봇 팔이 지정된 거리 또는 백분율로 이동하면 지정된 DO가 트리거됩니다. 거리가 양수이면 시작점에서 떨어진 거리를 나타냅니다. 거리가 음수이면 목표 지점에서 떨어진 거리를 나타냅니다. 아래의 "+"를 클릭하여 프로세스 IO를 추가하고 오른쪽의 "-"를 클릭하여 해당 프로세스 IO를 삭제할 수 있습니다.

Advanced setting ^

Speed

Acceleration

CP

Process I / O settings ?

DO_01 = OFF

Trigger mode

Distance mm

도약

모션 모드: 현재 위치에서 문 모양 모드로 직교 좌표계 아래의 목표 위치로 이동합니다.

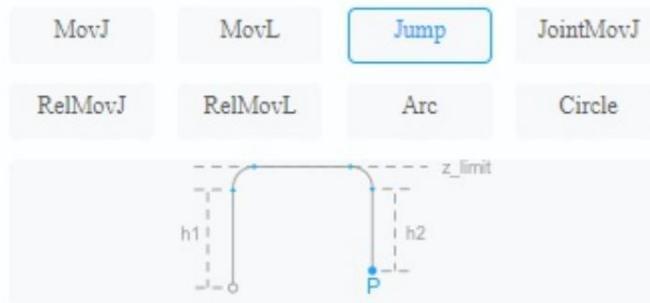
1. 로봇 팔은 먼저 지정된 높이를 수직으로 올린 다음 최대 높이로 전환합니다.

키.

2. 로봇 팔은 선형 모드에서 대상 지점을 향해 이동합니다.

3. 목표 지점 근처로 이동할 때 목표 지점 위의 지정된 높이로 전환한 다음 목표 지점까지 수직으로 하강합니다.

Motion type



기본 설정:

- 포인트 P의 좌표: 포인트 페이지에 추가된 후 여기에서 선택하거나 이 페이지에서 정의할 수 있는 대상 포인트 좌표입니다.
- 상승 높이(h1): 시작점 상승 높이 하
- 강 높이(h2): 종점 하강 높이 최대 높이(z_limit): 최대 상승
- 높이. 세 가지 높이 간의 관계는 위의 다이어그램을 참조할
- 수 있습니다.

Parameter configuration

Coordinates of point P:

Raise height h1 mm

Descent height h2 mm

Max height z_limit mm

고급 설정:

필요에 따라 고급 매개변수를 선택하고 구성합니다.

- 속도: 속도 비율, 범위: 1~100.

- Accel: 가속 비율, 범위: 1~100.



JointMovJ

동작 모드: 관절 보간 모드에서 현재 위치에서 목표 관절 각도로 이동합니다.



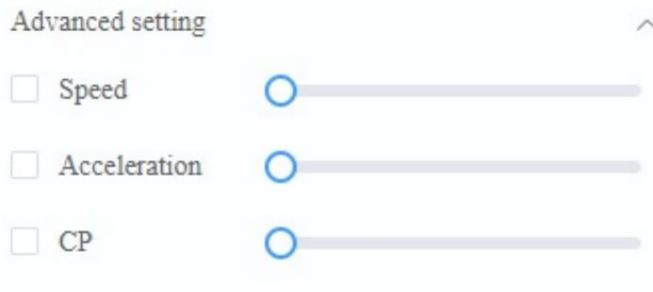
기본 설정: 터치를 통해 정의할 수 있는 목표 관절 각도.

Custom	
J1 0.000	J2 0.000
J3 0.000	J4 0.000
J5 0.000	J6 0.000
Get coordinates	

고급 설정:

필요에 따라 고급 매개변수를 선택하고 구성합니다.

- 속도: 속도 비율, 범위: 1~100.
- 가속(Accel): 가속 비율, 범위: 1~100.
- CP: 움직이는 연속 경로 설정, 범위: 0~100. 이 섹션의 끝에 있는 연속 경로(CP)를 참조하십시오.
자세한 내용은



RelMovJ

모션 모드: 관절 보간 모드에서 직교 좌표계 아래 현재 위치에서 목표 오프셋 위치로 이동합니다.



기본 설정: 직교 좌표계에서 X축, Y축 및 Z축 오프셋, 단위: mm

Offset

ΔX	0	mm	ΔZ	0	mm
ΔY	0	mm			

고급 설정:

필요에 따라 고급 매개변수를 선택하고 구성합니다.

- 속도: 속도 비율, 범위: 1~100.
- 가속(Accel): 가속 비율, 범위: 1~100.
- CP: 움직이는 연속 경로 설정, 범위: 0~100. 이 섹션의 끝에 있는 연속 경로(CP)를 참조하십시오.



RelMovL

모션 모드: 선형 보간 모드에서 현재 위치에서 직교 좌표계 아래 목표 오프셋 위치로 이동합니다.



기본 설정: 직교 좌표계에서 X축, Y축 및 Z축 오프셋, 단위: mm

Offset

ΔX	<input type="text" value="0"/>	mm	ΔZ	<input type="text" value="0"/>	mm
ΔY	<input type="text" value="0"/>	mm			

고급 설정:

필요에 따라 고급 매개변수를 선택하고 구성합니다.

- 속도: 속도 비율, 범위: 1~100.
- 가속(Accel): 가속 비율, 범위: 1~100.
- CP: 움직이는 연속 경로 설정, 범위: 0~100. 이 섹션의 끝에 있는 연속 경로(CP)를 참조하십시오.

자세한 내용은

Advanced setting ^

<input type="checkbox"/> Speed	<input type="range"/>
<input type="checkbox"/> Acceleration	<input type="range"/>
<input type="checkbox"/> CP	<input type="range"/>

호

모션 모드: 직교 좌표계에서 호 보간 모드로 현재 위치에서 목표 위치로 이동합니다. 현재 위치는 점 A와 점 B로 결정되는 직선 위에 있지 않아야 합니다.

 Motion type

기본 설정:

- 중간점 A 좌표: 호의 중간점 좌표 끝점 B 좌표: 목표점 좌표. 포인트 페이지에 추가
- 하거나 이 페이지에서 정의한 후 두 포인트를 여기에서 선택할 수 있습니다.

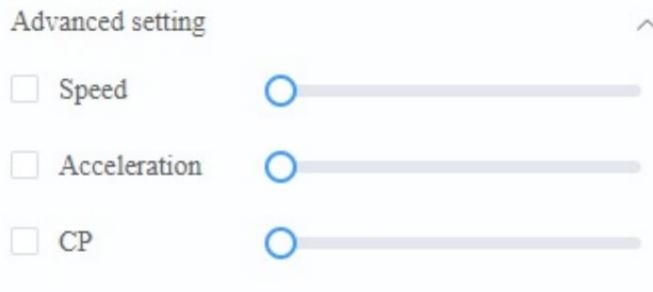
 Parameter configuration

Intermediate point A coordinate:	P1 <input type="button" value="▼"/>	Custom
End point B coordinate:	P1 <input type="button" value="▼"/>	Custom

고급 설정:

필요에 따라 고급 매개변수를 선택하고 구성합니다.

- 속도: 속도 비율, 범위: 1~100.
 - 가속(Accel): 가속 비율, 범위: 1~100.
 - CP: 움직이는 연속 경로 설정, 범위: 0~100. 이 섹션의 끝에 있는 연속 경로(CP)를 참조하십시오.
- 자세한 내용은



원

모션 모드: 원 보간 모드에서 현재 위치에서 이동하고, 지정된 원을 이동한 후 현재 위치로 돌아옵니다. 현재 위치는 A지점과 B지점으로 결정되는 직선상에 있지 않아야 하며, 세 지점으로 결정되는 원은 이동 범위를 초과할 수 없습니다.

로봇팔.

Motion type



기본 설정:

- 중간점 A 좌표: 원의 중간점 좌표를 결정하는데 사용됩니다.
- 끝점 B 좌표: 원의 끝점 좌표를 결정하는데 사용됩니다. 포인트 페이지에 추가하거나 이 페이지에서 정의한 후 두 포인트를 여기에서 선택할 수 있습니다.
- 원의 수: 원 운동의 원, 범위: 1~999.

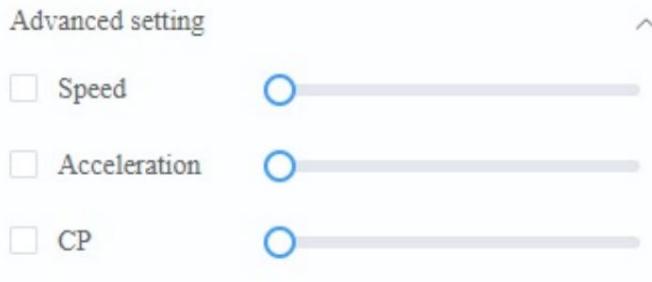
Parameter configuration

Intermediate point A coordinate:	<input style="border: 1px solid #ccc; padding: 2px 10px;" type="button" value="P1"/> <input style="border: 1px solid #ccc; padding: 2px 10px;" type="button" value="Custom"/>
End point B coordinate:	<input style="border: 1px solid #ccc; padding: 2px 10px;" type="button" value="P1"/> <input style="border: 1px solid #ccc; padding: 2px 10px;" type="button" value="Custom"/>
Number of cycles:	<input type="text" value="1"/>

고급 설정:

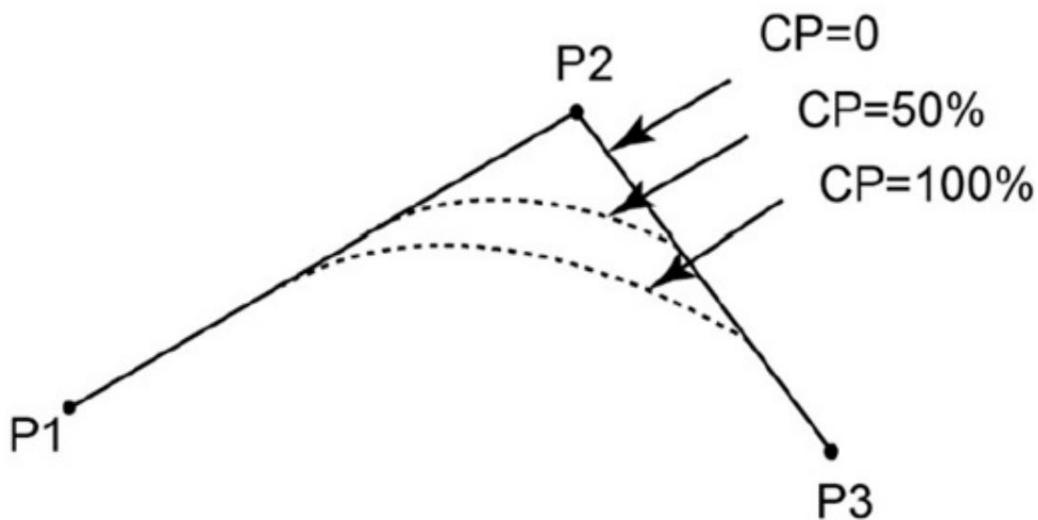
필요에 따라 고급 매개변수를 선택하고 구성합니다.

- 속도: 속도 비율, 범위: 1~100.
- 가속(Accel): 가속 비율, 범위: 1~100.
- CP: 움직이는 연속 경로 설정, 범위: 0~100. 이 섹션의 끝에 있는 연속 경로(CP)를 참조하십시오.
자세한 내용은



연속 경로(CP)

연속경로(CP)는 로봇팔이 시작점에서 중간점을 경유하여 끝점으로 이동할 때 아래 그림과 같이 중간점을 통과할 때 직각으로 또는 곡선으로 전환하는지 여부를 의미합니다.



모드버스 명령

Modbus 명령은 Modbus 통신과 관련된 작업에 사용됩니다.

Modbus 마스터 생성



설명: Modbus 마스터를 생성하고 슬레이브와 연결을 설정합니다.

매개변수:

- Modbus 슬레이브의 IP 주소
- Modbus 슬레이브 포트
- Modbus 슬레이브 ID, 범위: 1~4

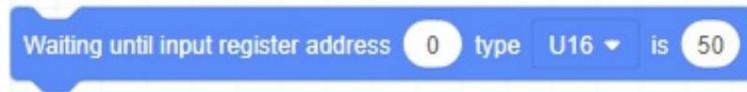
Modbus 마스터 생성 결과 얻기

get create modbus master result

설명: Modbus 마스터 생성 결과를 가져옵니다.

반환: Modbus 마스터가 성공적으로 생성되면 0을 반환하고, Modbus 마스터 생성에 실패하면 1을 반환합니다.
만들어진.

입력 레지스터 대기

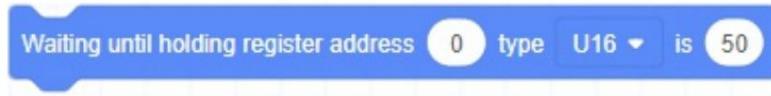


설명: 다음 명령을 실행하기 전에 입력 레지스터의 지정된 주소 값이 조건을 충족할 때까지 기다립니다.

매개변수:

- 주소: 입력 레지스터의 시작 주소. 값 범위: 0~4095.
- 데이터 유형
 - U16: 16비트 부호 없는 정수(2바이트, 하나의 레지스터 점유)
 - U32: 32비트 부호 없는 정수(4바이트, 2개의 레지스터 점유)
 - F32: 32비트 단정밀도 부동 소수점 수(4바이트, 2개의 레지스터 점유)
 - F64: 64비트 배정밀도 부동 소수점 수(8바이트, 4개의 레지스터 점유).
- 값이 충족되어야 하는 조건

홀딩 레지스터 대기

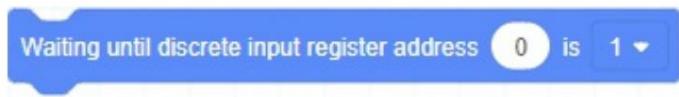


설명: 다음 명령을 실행하기 전에 조건을 충족하기 위해 홀딩 레지스터의 지정된 주소 값을 기다립니다.

매개변수:

- 주소: 홀딩 레지스터의 시작 주소. 값 범위: 0~4095.
- 데이터 유
 - 형 U16: 16비트 부호 없는 정수(2바이트, 하나의 레지스터 점유)
 - U32: 32비트 부호 없는 정수(4바이트, 2개의 레지스터 점유)
 - F32: 32비트 단정밀도 부동 소수점 수(4바이트, 2개의 레지스터 점유)
 - F64: 64비트 배정밀도 부동 소수점 수(8바이트, 4개의 레지스터 점유).
- 값이 충족되어야 하는 조건

불연속 입력 레지스터 대기

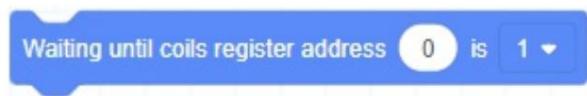


설명: 다음 명령을 실행하기 전에 이산 입력 레지스터의 지정된 주소 값이 조건을 충족할 때까지 기다립니다.

매개변수:

- 주소: 이산 입력 레지스터의 시작 주소. 값 범위: 0~4095. 값이 충족되어야 하는 조건
-

코일 레지스터 대기



설명: 다음 명령을 실행하기 전에 입력 레지스터의 지정된 주소 값이 조건을 충족할 때까지 기다립니다.

매개변수:

- 주소: 코일 레지스터의 시작 주소. 값 범위: 0~4095. 값이 충족되어야 하는 조건
-

입력 레지스터 가져오기

get input register address 0 type U16 ▾

설명: 입력 레지스터의 지정된 주소 값을 가져옵니다.

매개변수:

- 주소: 입력 레지스터의 시작 주소. 값 범위: 0~4095.
- 데이터 유형
 - U16: 16비트 부호 없는 정수(2바이트, 하나의 레지스터 점유)
 - U32: 32비트 부호 없는 정수(4바이트, 2개의 레지스터 점유)
 - F32: 32비트 단정밀도 부동 소수점 수(4바이트, 2개의 레지스터 점유)
 - F64: 64비트 배정밀도 부동 소수점 수(8바이트, 4개의 레지스터 점유).

반환: 입력 레지스터 값

홀딩 레지스터 가져오기

get holding register address 0 type U16 ▾

설명: 홀딩 레지스터의 지정된 주소 값을 가져옵니다.

매개변수:

- 주소: 홀딩 레지스터의 시작 주소. 값 범위: 0~4095.
- 데이터 유형
 - U16: 16비트 부호 없는 정수(2바이트, 하나의 레지스터 점유)
 - U32: 32비트 부호 없는 정수(4바이트, 2개의 레지스터 점유)
 - F32: 32비트 단정밀도 부동 소수점 수(4바이트, 2개의 레지스터 점유)
 - F64: 64비트 배정밀도 부동 소수점 수(8바이트, 4개의 레지스터 점유).

반환: 레지스터 값 유지

개별 입력 받기

get discrete input register address 0

설명: 개별 입력 레지스터의 지정된 주소 값을 가져옵니다.

매개변수: 이산 입력 레지스터의 시작 주소. 값 범위: 0~4095

반환: 이산 입력 값

코일 레지스터 가져오기

get coils register address 0

설명: 코일 레지스터의 지정된 주소 값을 가져옵니다.

매개변수: 코일 레지스터의 시작 주소. 값 범위: 0~4095

반환: 코일 레지스터 값

코일 레지스터의 여러 값 가져오기

get coils register array address 0 bits 1

설명: 코일 레지스터의 지정된 주소의 여러 값을 가져옵니다.

매개변수:

- 코일 레지스터의 시작 주소. 값 범위: 0~4095.
- 레지스터 비트 수.

반환: 테이블에 저장된 코일 레지스터 값. 표의 첫 번째 값은 시작 주소의 코일 레지스터 값에 해당합니다.

홀딩 레지스터의 여러 값 가져오기

set holding register address 0 data 50 type U16 ▾

설명: 홀딩 레지스터의 지정된 주소의 여러 값을 가져옵니다.

매개변수:

- 입력 레지스터의 시작 주소. 값 범위: 0~4095.
- 읽을 값의 수.
- 데이터 유
 - 형 U16: 16비트 부호 없는 정수(2바이트, 하나의 레지스터 점유)
 - U32: 32비트 부호 없는 정수(4바이트, 2개의 레지스터 점유)
 - F32: 32비트 단정밀도 부동 소수점 수(4바이트, 2개의 레지스터 점유)
 - F64: 64비트 배정밀도 부동 소수점 수(8바이트, 4개의 레지스터 점유).

반환: 테이블에 저장된 레지스터 값을 유지합니다. 테이블의 첫 번째 값은 시작 주소의 홀딩 레지스터 값에 해당합니다.

코일 레지스터 설정

set coils register address 0 data 0 ▾

설명: 코일 레지스터의 지정된 주소에 값을 씁니다.

매개변수:

- 코일 레지스터의 시작 주소. 값 범위: 6~4095.
- 코일 레지스터에 기록된 값. 값 범위: 0 또는 1.

다중 코일 레지스터 설정

setting multiple coil registers address 0 data 0,0,0,0,0,0,0,0,0

설명: 코일 레지스터의 지정된 주소에 여러 값을 씁니다.

매개변수:

- 코일 레지스터의 시작 주소. 값 범위: 0~4095.
- 쓸 값 비트 수입니다.
- 코일 레지스터에 기록된 값. 쓰여진 비트 수와 동일한 길이로 배열을 채웁니다. 각 비트는 0 또는 1만 될 수 있습니다.

홀딩 레지스터 설정

set holding register address 0 data 50 type U16 ▾

설명: 홀딩 레지스터의 지정된 주소에 값을 씁니다.

매개변수:

- 입력 레지스터의 시작 주소. 값 범위: 0~4095.
- 기록할 값으로, 선택한 데이터 유형에 해당해야 합니다.
- 데이터 유
 - 형 U16: 16비트 부호 없는 정수(2바이트, 하나의 레지스터 점유)
 - U32: 32비트 부호 없는 정수(4바이트, 2개의 레지스터 점유)
 - F32: 32비트 단정밀도 부동 소수점 수(4바이트, 2개의 레지스터 점유)
 - F64: 64비트 배정밀도 부동 소수점 수(8바이트, 4개의 레지스터 점유).

Modbus 마스터 닫기

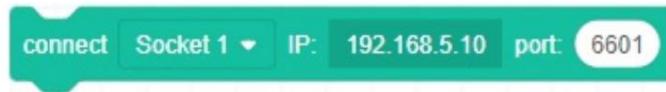
close modbus master

설명: Modbus 마스터를 닫고 모든 슬레이브에서 연결을 끊습니다.

TCP 명령

TCP 명령은 TCP와 관련된 작업에 사용됩니다.

소켓 연결

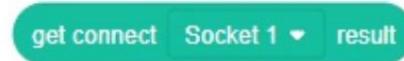


설명: 지정된 TCP 서버와 통신할 TCP 서버를 생성합니다.

매개변수:

- SOCKET 인덱스를 선택합니다(최대 4개의 TCP 통신 링크를 설정할 수 있음).
- TCP 서버의 IP 주소입니다.
- TCP 서버 포트.

SOCKET 연결 결과 얻기



설명: TCP 통신 연결 결과를 가져옵니다.

매개변수: SOCKET 인덱스를 선택합니다.

반환: 연결에 성공하면 0을, 연결에 실패하면 1을 반환합니다.

소켓 생성



설명: 클라이언트로부터 연결을 기다리는 TCP 서버를 생성합니다.

매개변수:

- 소켓 인덱스(최대 4개의 TCP 통신 링크를 설정할 수 있음).
- TCP 서버의 IP 주소입니다.
- TCP 서버 포트: 포트는 502 및 8080으로 설정할 수 없습니다. 그렇지 않으면 Modbus 기본 포트 또는 컨베이어 추적에 사용되는 포트와 충돌하여 TCP 서버 생성에 실패합니다.

SOCKET 생성 결과 얻기

get create Socket 1 ▾ result

설명: TCP 서버 생성 결과를 가져옵니다.

매개변수: SOCKET 인덱스를 선택합니다.

반환: 생성에 성공하면 0을, 생성에 실패하면 1을 반환합니다.

소켓 닫기

close Socket 1 ▾

설명: 지정된 SOCKET을 닫고 통신 링크를 끊습니다.

매개변수: SOCKET 인덱스를 선택합니다.

변수 가져오기

get variable Socket 1 ▾ type: string ▾ name: 0 s

설명: TCP 통신을 통해 변수를 가져와서 저장합니다.

매개변수:

- 소켓 인덱스
- 변수 유형: 문자열 또는 숫자.
- 이름은 생성된 변수 블록을 사용하여 받은 변수를 저장하는 데 사용됩니다. 대기 시간: 대기 시간이 0이면 변수를 받을 때까지 대기합니다.

변수 보내기

send variable Socket 1 ▾ Hello world

설명: TCP 통신을 통해 변수를 보냅니다.

매개변수:

- 소켓 색인.
- 보낼 데이터. 문자열 또는 숫자 값을 반환하는 타원형 블록을 사용하거나 직접 채울 수 있습니다. 공백.

변수 전송 결과 가져오기

get Socket 1 ▾ send result

설명: 변수를 보낸 결과를 가져옵니다.

매개변수: 소켓 색인.

Return: 변수가 전송되면 0을 반환하고, 변수 전송에 실패하면 1을 반환합니다.

부록 C 스크립트 명령

- [C.1 Lua 기본 문법](#) [C.1.1 변수](#)
 - [및 데이터 유형](#) [C.1.2 연산자](#)
 -
 - [C.1.3 공정 제어](#)
- [C.2 명령 설명](#)
 - [C.2.1 모션](#)
 - [C.2.2 모션 파라미터](#)
 - [C.2.3 상대 운동](#)
 - [C.2.4 IO](#)
 - [C.2.5 TCP/UDP](#)
 - [C.2.6 모드버스](#)
 - [C.2.7 프로그램 제어](#)
 - [C.2.8 비전](#)

루아 기본 문법

변수 및 데이터 유형

루아 프로그래밍 관련 지식을 체계적으로 배우고 싶다면 인터넷에서 루아 튜토리얼을 검색해보세요. 이 가이드는 빠른 참조를 위한 기본 Lua 구문 중 일부만 나열합니다.

변수는 값을 저장하거나 값을 매개변수로 전달하거나 값을 결과로 반환하는 데 사용됩니다. 변수는 "="로 할당됩니다.

Lua의 변수는 "local"을 사용하여 명시적으로 지역 변수로 선언하지 않는 한 기본적으로 전역 변수입니다.

지역 변수의 범위는 선언 위치부터 변수가 있는 블록 끝까지입니다.
위치.

```
a = 5          -- 전역 변수
로컬 b = 5    -- 지역 변수
```

변수 이름은 숫자로 시작할 수 없는 문자, 밑줄 및 숫자로 구성된 문자열일 수 있습니다. Lua에서 예약한 키워드는 변수 이름으로 사용할 수 없습니다.

Lua 변수의 경우 유형을 정의할 필요가 없습니다. 변수에 값을 할당하면 Lua는 값에 따라 변수의 유형을 자동으로 판단합니다.

Lua는 숫자, 부울, 문자열 및 테이블을 포함한 다양한 데이터 유형을 지원합니다. Lua의 배열은 유형입니다.
테이블의.

Lua에는 특수한 데이터 유형인 nil도 있습니다. 이는 무효(유효한 값이 없음)를 의미합니다. 예를 들어 할당되지 않은 변수를 인쇄하면 nil 값이 출력됩니다.

숫자

Lua의 숫자는 배정도 부동 소수점 숫자이며 다양한 연산을 지원합니다. 다음 형식은 모두 숫자로 간주됩니다.

- 2
- 2.2
- 0.2
- 2e+1
- 0.2e-1
- 7.8263692594256e-06

부울

부울 유형에는 true와 false의 두 가지 선택적 값만 있습니다. Lua는 false와 nil을 false로 처리하고 숫자 0을 포함하여 나머지는 true로 처리합니다.

끈

문자열은 숫자, 문자 및/또는 밑줄로 구성될 수 있습니다. 문자열은 세 가지 방법으로 표현할 수 있습니다.

- 작은따옴표 사이의 문자입니다.
- 큰따옴표 사이의 문자입니다. [[와]] 사이의 문
- 자

일련의 숫자에 대해 산술 연산을 수행할 때 Lua는 일련의 숫자를 숫자로 변환하려고 시도합니다.

Lua는 문자열 연산을 지원하는 많은 함수를 제공합니다.

기능	설명
string.upper(인수)	대문자로 변환
string.lower(인수)	소문자로 변환
string.gsub(mainString, findString, replaceString, num)	문자열의 문자를 바꿉니다. MainString 은 소스 문자열, findString 은 교체할 문자, replaceString은 교체 문자, num 은 대체 수(무시할 수 있음)
string.find(str, substr, [init, [end]])	대상 문자열 str에서 지정된 콘텐츠 substr을 검색합니다 . 일치하는 부분 문자열이 발견되면 부분 문자열의 시작 및 끝 인덱스가 반환되고 없으면 nil이 반환됩니다.
string.reverse(인수)	문자열이 반전됨
문자열.형식(...)	printf 와 유사한 형식의 문자열을 반환합니다.
string.char(arg) 및 string.byte(arg[,int])	char 는 정수를 문자로 변환하고 연결하는 데 사용됩니다. 바이트는 문자를 정수 값으로 변환하는 데 사용됩니다.
string.len(인수)	문자열의 길이 구하기
string.rep(문자열, n)	문자열 복사, n은 복제 횟수를 나타냅니다.
..	두 문자열을 연결하는 데 사용
string.gmatch(str, 패턴)	반복자 함수입니다. 이 함수가 호출될 때마다 패턴 설명과 일치하는 str 에서 찾은 다음 하위 문자열을 반환합니다 . pattern으로 설명된 하위 문자열이 없으면 반복자는 nil을 반환합니다.
string.match(str, 패턴, 초기화)	대상 문자열 str에서 Pattern의 설명과 일치하는 지정된 콘텐츠를 검색합니다. Init은 검색을 위한 시작 색인을 지정하는 선택적 매개변수이며 기본값은 1입니다. 소스 str에서 첫 번째로 일치하는 항목만 찾습니다. 일치하는 문자가 발견되면 일치하는 문자열이 반환됩니다. 캡처 플래그가 설정되지 않은 경우 일치하는 전체 문자열이 반환됩니다. 성공적으로 일치하지 않으면 nil을 반환합니다.
string.sub(s, i [, j])	문자열을 가로채는 데 사용됩니다. s 는 잘라낼 소스 문자열, i 는 시작 색인, j 는 끝 색인, 기본값은 마지막 문자를 나타내는 -1입니다.

예:

```

str = "루아"
print(string.upper(str))          --대문자로 변환하고 결과를 출력합니다. LUA
print(string.lower(str))          --소문자로 변환하고 결과를 출력합니다. lua
print(string.reverse(str))        --문자열이 반전되고 결과가 출력됩니다. aul
print(string.len("abc"))          --문자열 ABC의 길이를 계산하고 결과를 출력합니다.

: 삼
print(string.format(" 값은: %d",4)) print(string.rep(str,2))      --결과 인쇄: 값:4
                                                               --문자열을 두 번 복사하고 결과를 출력합니다. LuLua

문자열1 = "cn." string2
= "dobot" string3 = ".cc"
print("주"
소 :" ,string1..string2..string3)                         --사용..문자열을 만들고 결과를 인쇄합니다. 추가
자료:cn.dobot.cc

문자열1 = [[aaaa]]
print(string.gsub(문자열1,"a","z",3))                      -- 문자열을 교체하고 결과를 출력합니다: zzza

print(string.find("Hello Lua user", "Lua", 1)) e 하위 문자열의 시작 및 끝 인      --문자열에서 Lua를 검색하고 th를 반환합니다.
덱스, 결과 인쇄: 7,9

sourcestr = "접두사--runoobgoogletaobao--접미사" sub = string.sub(sourcestr,
1, 8)                                         --문자열 print("\n result", string.format("%q", sub)) 의 첫 번째부터 여덟
번째 문자를 가져
옵니다.                                         --인쇄: 결과: "prefix--"

```

< >

테이블

테이블은 인덱스가 있는 데이터 그룹입니다.

- 테이블을 생성하는 가장 간단한 방법은 빈 테이블을 생성하는 {}를 사용하는 것입니다. 이 메서드는 테이블을 직접 초기화합니다.
- 테이블은 연관 배열을 사용할 수 있습니다. 배열의 인덱스는 모든 유형의 데이터가 될 수 있지만 값은 nil일 수 없습니다.
- 테이블의 크기는 고정되어 있지 않으며 필요에 따라 확장할 수 있습니다.
- "#" 기호를 사용하여 테이블의 길이를 얻을 수 있습니다.

```

tbl = {[1] = 2, [2] = 6, [3] = 34, [4] =5} print("tbl 길이", #tbl) -- 인쇄 결
과는 4

```

Lua는 테이블의 동작을 지원하기 위해 많은 기능을 제공합니다.

기능	설명
table.concat (테이블 [, sep [, 시작 [, 끝]]])	table.concat() 함수는 지정된 구분 기호(sep)로 구분된 지정된 배열의 모든 요소를 처음부터 끝까지 나열합니다.
테이블.삽입	

(테이블, [pos,] 값)	테이블의 끝을 기본값으로 하는 선택적 매개변수입니다.
table.remove (테이블 [, pos])	지정된 위치(pos)에 있는 테이블의 요소를 반환하면 뒤따르는 요소가 앞으로 이동합니다. pos는 선택적 매개변수이며 기본값은 마지막 요소에서 삭제된 테이블 길이입니다.
table.sort (테이블 [, comp])	테이블의 요소는 오름차순으로 정렬됩니다.

- 예 1:

```

fruits = {} fruits =
--테이블 초기화
["banana", "orange", "apple"] --테이블에 할당

print("String after concatenation", table.concat(fruits, ", ", 2,3)) --테이블에서 지정한 인덱스의 요소를 가져와 연결, String 연결 후 orange, app

르

--마지막에 요소 삽입
table.insert(fruits,"mango") print("인덱스
가 4인 요소는",fruits[4]) --결과 출력: inde가 있는 요소
x 4는 망고

-- 인덱스 2에 요소 삽입
table.insert(fruits,2,"포도") print("인덱스가 2인 요
소는",fruits[2]) --결과 출력: inde가 있는 요소
x 2는 포도

print("마지막 요소는",fruits[5]) table.remove(fruits) print("제
거 후 마지막 요소는",fruits[5]) --
결과를 출력합니다: 마지막 요소
제거 후 없음
--결과 출력: 마지막 요소는 mango입니다.

```

- 예 2:

```

fruits = {"바나나", "오렌지", "사과", "포도"} print("이전") for k,v in
ipairs(fruits) do

    인쇄(k,v)
    끝
--오름차순 table.sort(fruits)
print("After") for k,v in
ipairs(fruits) do

    인쇄(k,v)
    끝
--결과 출력: 바나나 오렌지 사과 포도
--결과 출력: 사과 바나나 포도 오렌지

```

정렬

배열은 특정 순서로 정렬된 동일한 데이터 유형의 요소 모음입니다. 1차원일 수도 있고 다차원일 수도 있습니다. 배열의 인덱스는 정수로 나타낼 수 있으며 배열의 크기는 고정되어 있지 않습니다.

- 1차원 배열: 선형 테이블의 논리적 구조를 가진 가장 단순한 배열입니다.
- 다차원 배열: 배열을 포함하거나 1차원 배열의 인덱스가 배열에 해당합니다.

예제 1: for 루프 명령을 통해 1차원 배열을 할당하거나 읽을 수 있습니다. 정수 인덱스는 배열 요소에 액세스하는 데 사용됩니다. 인덱스에 값이 없으면 배열은 nil을 반환합니다.

```
array = {"Lua", "Tutorial"} --i= 0, 2 do에 대한 1차원 배열 만들기

print(배열[i])           --결과 출력 :nil Lua Tutorial
끝
```

Lua에서 배열 인덱스는 1 또는 0에서 시작합니다. 또는 음수를 배열 인덱스로 사용할 수 있습니다.

```
배열 = {}
for i=-2, 2 할
    배열[i] = i*2+1           --1차원 배열에 값 할당
끝
i = -2,2에 대해
print(배열[i])           --결과 출력 : -3 -1 1 3 5
끝
```

예 2: 3개의 행과 3개의 열로 구성된 배열

```
-- 배열 초기화 array = {} for i=1,3 do

    배열[i] = {} for j=1,3
        do 배열[i][j] = i*j
    끝
끝

-- i=1,3에 대한 배열 액세스
do
    for j=1,3 do
        print(array[i][j])           --결과 출력 :1 2 3 2 4 6 3 6 9
    끝
끝
```

운영자

산술 연산자

명령	설명
+	덧셈
-	빼기
*	곱셈
/	부동 소수점 나눗셈
//	플로어 구분
%	나머지
^^	지수화
&	그리고 연산자
\	OR 연산자
~	XOR 연산자
<<	왼쪽 시프트 연산자
>>	오른쪽 시프트 연산자

• 예

```
a=20
b=5
print(a+b)           --a + b에 대한 결과 출력: 25
print(ab)            --a 빼기 b의 결과 출력: 15
print(a*b)           --a 곱하기 b의 결과 출력: 100
print(a/b)           --a를 b로 나눈 결과 출력: 4
print(a//b)          --a를 b로 나눈 결과를 출력합니다: 4
print(a%b)           --a를 b로 나눈 나머지를 출력: 0
print(a^b)           --a의 b승에 대한 결과 출력: 3200000
print(a&b)           --a와 b의 결과 출력: 4
print(a|b ) 인      --a OR b의 결과 출력: 21
색(a~b) 인색        --XOR b의 결과 출력: 17
(a<<b) 인색        --왼쪽으로 이동한 결과 출력 b: 640
(a>>b)             --오른쪽으로 이동한 결과 출력 b: 0
```

관계 연산자

명령	설명
==	동일한

$\sim=$	같지 않음
\leq	같거나 작음
\geq	같거나 큼
$<$	미만
$>$	보다 큼

- 예

```
a=20
b=5

인쇄(a==b) 인쇄          --a가 b와 같은지 확인: false
(a~=b) 인쇄              --a가 b와 같지 않은지 확인: true
(a<=b) 인쇄              --a가 b보다 작거나 같은지 확인: false
(a>=b) 인쇄              --a가 b보다 크거나 같은지 확인: true
(a<b) 인쇄               --a가 b보다 작은지 확인: false
(a>b)                   --a가 b보다 큰지 확인: true
```

논리 연산자

명령	설명
그리고	논리 AND 연산자. 결과는 양쪽 모두 참이면 참, 한쪽만 거짓이면 거짓
또는	논리 OR 연산자. 결과는 적어도 한쪽이 참이면 참, 양쪽 모두 거짓이면 거짓
- 아니다	논리적 NOT 연산자. 판정 결과는 정반대

- 예

```
a=참
b=거짓

print(a 및 b) print(a      --참과 거짓, 결과는 거짓
또는 b) print(20 >      --참 또는 거짓, 결과는 참
5는 참이 아님)           --참과 거짓, 결과는 거짓
```

프로세스 제어

명령	설명
if...then... elseif... then... else...끝	조건부 명령(if). 조건이 위에서 아래로 유효한지 확인합니다. 조건 판단이 참일 경우 해당 코드 블록이 실행되고 이후의 조건 판단은 바로 무시되어 더 이상 실행되지 않습니다.
하는 동 안...	루프 명령(동안). 조건이 참이면 프로그램이 해당 코드 블록을 반복적으로 실행하도록 합니다. 명령 문이 실행되기 전에 조건이 참인지 확인합니다.
...을 위해 ... 끝	루프 명령(for), 지정된 문을 반복적으로 실행하고 for 문에서 반복 횟수를 제어할 수 있습니다.
반복...까 지()	루프 명령(반복), 지정된 조건이 참일 때까지 루프가 반복됩니다.

- 예

1.조건부 명령(if)

```
a = 100;
b = 200;
if(a == 100)
그 다음에
    if(b == 200)
        그 다음에
            print("이것은: ", a );
            print("이것은 b입니다: ", b );
        끝
    끝
```

2.Loop 명령(동안)

```
a=10
동안( a < 20 )
하다
    print("이것은: ", a)
    a = a+1
끝
```

3.루프 명령(for)

```
i=10,1,-1에 대해 print(i)를
    수행합니다.
끝
```

4.루프 명령(반복)

```
a = 10  
반복  
    print("이것은: ", a)  
    a = a + 1  
(a > 15)까지
```

명령 설명

운동

동작 명령은 로봇 팔의 움직임을 제어하는 데 사용됩니다. 모션 속도 비율/가속 비율은 Motion 파라미터에서도 설정할 수 있습니다. 모션 명령과 모션 파라미터 명령 모두에 파라미터를 설정하면 모션 명령의 값이 우선합니다.

실제 로봇 속도/가속 = 명령에서 설정된 비율 × 재생 설정에서 속도/가속 × 전체 속도 비율.

가다

명령:

```
Go(P,"사용자=1 도구=2 CP=1 속도=50 가속=20 SYNC=1")
```

설명:

point-to-point 모드에서 현재 위치에서 직교 좌표계 아래 목표 위치로 이동합니다. 관절 운동의 궤적은 선형이 아니며 모든 관절이 동시에 운동을 완료합니다.

필수 매개변수:

P: 사용자 정의되거나 포인트 페이지에서 얻은 목표 포인트. 데카르트 좌표점만 지원됩니다.

선택적 매개변수:

- 사용자: 설정에 추가된 후 여기에서 사용할 수 있는 사용자 좌표계 인덱스입니다.
- 도구: 설정에 추가된 후 여기에서 사용할 수 있는 도구 좌표계 인덱스입니다.
- CP: 모션의 연속 경로 설정([모션 매개변수의 CP 명령 참조](#)), 범위: 0~100.
- 속도: 속도 비율, 범위: 1~100.
- Accel: 가속 비율, 범위: 1~100.
- SYNC: 동기화 플래그, 범위: 0 또는 1(기본값: 0).
 - SYNC=0: 비동기 실행, 실행 프로세스와 관계없이 호출된 직후에 반환됨을 의미합니다.
 - SYNC=1: 동기 실행. 이는 명령이 완전히 실행될 때까지 호출된 후 반환되지 않음을 의미합니다.

예:

```
이동(P1)
```

로봇은 기본 설정으로 point-to-point 모드에서 P1으로 이동합니다.

이동하다

명령:

```
Move(P,"User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
```

설명:

선형 모드에서 현재 위치에서 직교 좌표계 아래 목표 위치로 이동합니다.

필수 매개변수:

P: 사용자 정의되거나 포인트 페이지에서 얻은 목표 포인트. 대카르트 좌표점만 지원됩니다.

선택적 매개변수:

- 사용자: 설정에 추가된 후 여기에서 사용할 수 있는 사용자 좌표계 인덱스입니다.
- 도구: 설정에 추가된 후 여기에서 사용할 수 있는 도구 좌표계 인덱스입니다.
- CP: 모션의 연속 경로 설정([모션 매개변수의 CP 명령 참조](#)), 범위: 0~100.
- SpeedS: 속도 비율. 값 범위: 1~100 AccelS: 가속도 비율. 값 범위: 1~100 SYNC: 동기화 플래그, 범위: 0 또는 1(기본값: 0).
 - SYNC=0: 비동기 실행, 실행 프로세스와 관계없이 호출된 직후에 반환됨을 의미합니다.
 - SYNC=1: 동기 실행. 이는 명령이 완전히 실행될 때까지 호출된 후 반환되지 않음을 의미합니다.

예:

```
이동(P1)
```

로봇 팔은 기본 설정으로 선형 모드에서 P1로 이동합니다.

무브제이

명령:

```
MoveJ(P,"CP=1 속도=50 가속=20 SYNC=1")
```

설명:

점대점 모드(관절 동작)에서 현재 위치에서 목표 관절 각도로 이동합니다.

필수 매개변수:

P: 관절 각도를 통해서만 정의할 수 있는 목표 지점.

선택적 매개변수:

- CP: 모션의 연속 경로 설정([모션 매개변수의 CP 명령 참조](#)), 범위: 0~100.
- 속도: 속도 비율, 범위: 1~100.
- Accel: 가속 비율, 범위: 1~100.
- SYNC: 동기화 플래그, 범위: 0 또는 1(기본값: 0).
 - SYNC=0: 비동기 실행, 실행 프로세스와 관계없이 호출된 직후에 반환됨을 의미합니다.
 - SYNC=1: 동기 실행. 이는 명령이 완전히 실행될 때까지 호출된 후 반환되지 않음을 의미합니다.

예:

```
로컬 P = {조인트={0,-0.0674194,0,0,0,0}}
이동J(P)
```

관절 좌표점 P를 정의합니다. 기본 설정으로 로봇을 P로 이동합니다.

Circle3

명령:

```
Circle3(P1,P2,카운트,"사용자=1 도구=2 CP=1 속도S=50 AccelS=20 SYNC=1")
```

설명:

원 보간 모드에서 현재 위치에서 이동하고, 지정된 원을 이동한 후 현재 위치로 돌아옵니다. 현재 위치인 P1과 P2를 통해 원을 결정해야 하므로 현재 위치는 P1과 P2에 의해 결정된 직선 상에 있어서는 안 되며, 세 점에 의해 결정된 원은 로봇 팔의 이동 범위를 초과할 수 없습니다.

필수 매개변수:

- P1: 사용자 정의 또는 포인트 페이지에서 얻은 중간 포인트. 데카르트 좌표점만 지원됩니다.
- P2: 사용자 정의되거나 포인트 페이지에서 얻은 끝점. 데카르트 좌표점만 지원됩니다.
- 개수: 원의 수, 범위: 1 - 999.

선택적 매개변수:

- 사용자: 설정에 추가된 후 여기에서 사용할 수 있는 사용자 좌표계 인덱스입니다.
- 도구: 설정에 추가된 후 여기에서 사용할 수 있는 도구 좌표계 인덱스입니다.
- CP: 모션의 연속 경로 설정([모션 매개변수의 CP 명령 참조](#)), 범위: 0~100.

- SpeedS: 속도 비율. 값 범위: 1~100 AccelS: 가속 비율.
- 값 범위: 1~100 SYNC: 동기화 플래그, 범위: 0 또는 1(기본값: 0).
 - SYNC=0: 비동기 실행, 실행 프로세스와 관계없이 호출된 직후에 반환됨을 의미합니다.
 - SYNC=1: 동기 실행. 이는 명령이 완전히 실행될 때까지 호출된 후 반환되지 않음을 의미합니다.

예:

```
이동(P1)
Circle3(P2,P3,1)
```

로봇 팔은 P1으로 이동한 다음 P1, P2 및 P3에 의해 결정된 전체 원을 이동합니다.

아크3

명령:

```
Arc3(P1,P2,"사용자=1 도구=2 CP=1 속도S=50 가속도=20 동기=1")
```

설명:

직교 좌표계에서 호 보간 모드로 현재 위치에서 목표 위치로 이동합니다. 호는 현재 위치인 P1과 P2를 통해 결정되어야 하므로 현재 위치는 P1과 P2에 의해 결정된 직선 위에 있지 않아야 합니다.

필수 매개변수:

- P1: 사용자 정의 또는 포인트 페이지에서 얻은 중간 포인트. 데카르트 좌표점만 지원됩니다.
- P2: 사용자 정의되거나 포인트 페이지에서 얻은 대상 포인트. 데카르트 좌표점만 지원됩니다.

선택적 매개변수:

- 사용자: 설정에 추가된 후 여기에서 사용할 수 있는 사용자 좌표계 인덱스입니다.
- 도구: 설정에 추가된 후 여기에서 사용할 수 있는 도구 좌표계 인덱스입니다.
- CP: 모션의 연속 경로 설정([모션 매개변수의 CP 명령 참조](#)), 범위: 0~100.
- SpeedS: 속도 비율. 값 범위: 1~100 AccelS: 가속 비율.
- 값 범위: 1~100 SYNC: 동기화 플래그, 범위: 0 또는 1(기본값: 0).
 - SYNC=0: 비동기 실행, 실행 프로세스와 관계없이 호출된 직후에 반환됨을 의미합니다.
 - SYNC=1: 동기 실행. 이는 호출된 후 반환될 때까지 반환되지 않음을 의미합니다.

명령이 완전히 실행됩니다.

예:

```
이동(P1)
아크3(P2,P3)
```

로봇은 P1으로 이동한 다음 아크 보간 모드에서 P2를 통해 P3으로 이동합니다.

GolO

명령:

```
GolO(P,[{Mode,Distance,Index>Status},{Mode,Distance,Index>Status}...], "사용자=1 도구=2 CP=1 속도=50 가속 =20 SYNC=1")
```

설명:

직교 좌표계에서 점대점 모드(관절 동작)로 현재 위치에서 목표 위치로 이동하고 로봇이 이동할 때 디지털 출력 포트의 상태를 설정합니다.

필수 매개변수:

- P: 사용자 정의되거나 포인트 페이지에서 얻은 목표 포인트. 데카르트 좌표점만 지원됩니다.
- 디지털 출력 매개변수: 로봇 팔이 지정된 거리 또는 백분율로 이동할 때 트리거할 지정된 DO를 설정합니다. 각각 다음을 포함하는 여러 그룹을 설정할 수 있습니다.

매개변수:

- 모드: 트리거 모드. 0: 거리 백분율; 1: 거리 값 거리: 지정된 거리
- - Distance가 양수이면 시작점에서 떨어진 거리를 의미합니다. Distance가 음수이면 목표 지점에서 떨어진 거리를 나타냅니다. Mode가 0이면 Distance는 전체 거리에 대한 백분율을 나타냅니다. 범위: 0~100 Mode가 1이면 Distance는 거리 값을 의미합니다. 단위: 밀리미터
- 인덱스: DO 인덱스
- 상태: DO 상태. 0: 꺼짐; 1: 켜짐

선택적 매개변수:

- 사용자: 설정에 추가된 후 여기에서 사용할 수 있는 사용자 좌표계 인덱스입니다.
- 도구: 설정에 추가된 후 여기에서 사용할 수 있는 도구 좌표계 인덱스입니다.
- CP: 모션의 연속 경로 설정([모션 매개변수의 CP 명령 참조](#)), 범위: 0~100.
- 속도: 속도 비율, 범위: 1~100.
- Accel: 가속 비율, 범위: 1~100.
- SYNC: 동기화 플래그, 범위: 0 또는 1(기본값: 0).

- SYNC=0: 비동기 실행, 실행 프로세스와 관계없이 호출된 직후에 반환됨을 의미합니다.
- SYNC=1: 동기 실행. 이는 명령이 완전히 실행될 때까지 호출된 후 반환되지 않음을 의미합니다.

예:

```
GolO(P1, {0, 10}, 2, 1)
```

로봇 팔은 기본 설정으로 P1 쪽으로 이동합니다. 시작점에서 10% 거리를 이동할 때 DO2를 ON으로 설정합니다.

MoveIO

명령:

```
MoveIO(P,{{모드,거리,인덱스,상태},{모드,거리,인덱스,상태}}, "사용자=1 도구=2 CP=1 속도 dS=50 AccelS=20 SYNC=1")
```

설명:

현재위치에서 목표위치까지 직선모드로 직교좌표계로 이동하고 로봇이 이동할 때 디지털 출력포트의 상태를 설정한다.

필수 매개변수:

- P: 사용자 정의되거나 포인트 페이지에서 얻은 목표 포인트. 데카르트 좌표점만 지원됩니다.
- 디지털 출력 매개변수: 로봇 팔이 지정된 거리 또는 백분율을 이동할 때 트리거할 지정된 DO를 설정합니다. 각각 다음 매개변수를 포함하는 여러 그룹을 설정할 수 있습니다. 모드: 트리거 모드. 0: 거리 백분율; 1: 거리 값 거리: 지정된 거리
 - Distance가 양수이면 시작점에서 떨어진 거리를 의미합니다. Distance가 음 수이면 목표 지점에서 떨어진 거리를 나타냅니다. Mode가 0이면 Distance는 전체 거리에 대한 백분율을 나타냅니다. 범위: 0~100 Mode가 1이면 Distance는 거리 값을 의미합니다. 단위: 밀리미터
 - 인덱스: DO 인덱스
 - 상태: DO 상태. 0: 꺼짐; 1: 켜짐

선택적 매개변수:

- 사용자: 설정에 추가된 후 여기에서 사용할 수 있는 사용자 좌표계 인덱스입니다.
- 도구: 설정에 추가된 후 여기에서 사용할 수 있는 도구 좌표계 인덱스입니다.
- CP: 모션의 연속 경로 설정(모션 매개변수의 CP 명령 참조), 범위: 0~100.
- 속도: 속도 비율, 범위: 1~100.

- Accel: 가속 비율, 범위: 1~100.
- SYNC: 동기화 플래그, 범위: 0 또는 1(기본값: 0).
 - SYNC=0: 비동기 실행, 실행 프로세스와 관계없이 호출된 직후에 반환됨을 의미합니다.
 - SYNC=1: 동기 실행. 이는 명령이 완전히 실행될 때까지 호출된 후 반환되지 않음을 의미합니다.

예:

```
MoveIO(P1, {0, 10}, 2, 1)
```

로봇은 기본 설정으로 P1 쪽으로 이동합니다. 시작점에서 10% 거리를 이동할 때 DO2를 ON으로 설정합니다.

동작 매개변수

모션 매개변수는 로봇의 관련 모션 매개변수를 설정하거나 가져오는 데 사용됩니다.

동조

명령:

동조()

설명:

이 명령은 대기열 명령을 실행하는 프로그램을 차단하는 데 사용됩니다. 모든 대기열 명령이 실행될 때까지 반환한 다음 후속 명령을 실행합니다. 일반적으로 로봇 팔이 움직임을 완료하기를 기다리는 데 사용됩니다.

예:

이동(P1)
이동(P2)
동조()

로봇 팔은 P1으로 이동한 다음 후속 명령을 실행하기 위해 돌아오기 전에 P2로 이동합니다.

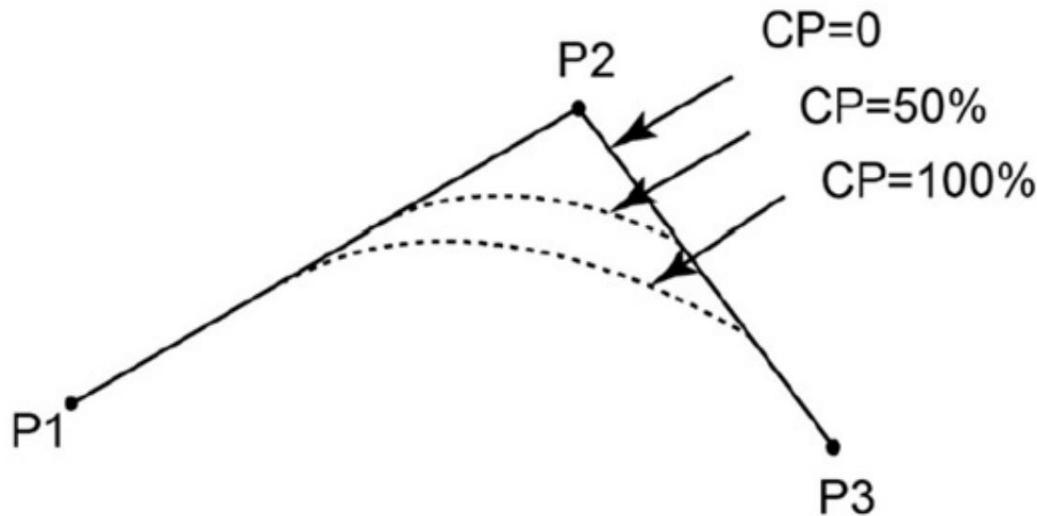
CP

명령:

CP(R)

설명:

연속 경로(CP) 비율, 즉 로봇 팔이 여러 점을 통해 연속적으로 이동할 때 중간 지점을 통과할 때 직각으로 전환하는지 곡선 방식으로 전환하는지 여부를 설정합니다.



필수 매개변수:

R: 연속 경로 비율, 범위: 0~100

예:

CP(50)

이동(P1)

이동(P2)

이동(P3)

로봇은 50% 연속 경로 비율로 P2를 통해 P1에서 P3으로 이동합니다.

속도

명령:

속도(R)

설명:

조인트 모션의 속도 비율을 설정합니다. 실제 로봇 속도 = 명령에 설정된 백분율 × 재생 설정 속도 × 전체 속도 비율.

필수 매개변수:

R: 속도 비율, 범위: 1~100

예:

속도(20)

이동(P1)

로봇은 20%의 속도로 P1으로 이동합니다.

가속

명령:

가속(R)

설명:

조인트 모션의 가속도를 설정합니다. 실제 로봇 가속 = 명령에 설정된 비율 × 재생 설정의 가속 × 전체 속도 비율.

필수 매개변수:

R: 가속도, 범위: 0~100

예:

엑셀(50)
이동(P1)

로봇은 50%의 가속도로 P1으로 이동합니다.

속도S

명령:

스피드S(R)

설명:

직선운동과 원호운동의 속도비를 설정합니다. 실제 로봇 속도 = 명령에 설정된 백분율 × 재생 설정 속도 × 전체 속도 비율.

필수 매개변수:

R: 속도 비율, 범위: 0~100

예:

스피드S(20)
이동(P1)

로봇은 20%의 속도로 P1으로 이동합니다.

AccelS

명령:

```
AccelS(R)
```

설명:

선형 및 원호 운동의 가속 비율을 설정합니다. 실제 로봇 가속 = 명령에 설정된 비율 × 재생 설정의 가속 × 전체 속도 비율.

필수 매개변수:

R: 가속비, 범위: 1~100

예:

```
엑셀S(50)  
이동(P1)
```

로봇은 50% 가속 비율로 P1으로 이동합니다.

GetPose

명령:

```
GetPose()
```

설명:

직교 좌표계에서 로봇 팔의 실시간 자세를 가져옵니다. 사용자 좌표계 또는 도구 좌표계를 설정한 경우 획득한 자세는 현재 좌표계 아래에 있습니다.

반품:

현재 자세의 데카르트 좌표

예:

```
로컬 currentPose = GetPose()  
이동(P1)  
이동(현재 포즈)
```

로봇은 P1으로 이동한 후 현재 자세로 돌아갑니다.

GetAngle

명령:

```
GetAngle()
```

설명:

관절 좌표계에서 로봇 팔의 실시간 자세를 가져옵니다.

반품:

현재 자세의 관절 좌표

예:

```
로컬 currentAngle = GetAngle()  
이동(P1)  
MoveJ(현재 각도)
```

로봇은 P1으로 이동한 후 현재 자세로 돌아갑니다.

체크고

명령:

```
체크고(P)
```

설명:

관절 모션 명령의 운용성을 확인하십시오.

필수 매개변수:

P: 사용자 정의되거나 포인트 페이지에서 얻은 목표 포인트. 대카르트 좌표점만 지원됩니다.

반품:

결과를 확인하십시오.

- 0: 오류 없음
- 16: 솔더 특이점에 가까운 끝점
- 17: 솔루션이 없는 끝점 역기구학 오류
- 18: 작업 영역을 벗어난 결과의 역기구학 오류
- 22: 팔 방향 오류
- 26: 손목 특이점에 가까운 끝점
- 27: 팔꿈치 특이점에 닫힌 끝점
- 29: 속도 매개변수 오류

- 32: 궤적의 어깨 특이점
- 33: 궤적에 솔루션이 없는 역기구학 오류
- 34: 궤적에서 작업 영역을 벗어난 결과의 역기구학 오류
- 35: 궤적의 손목 특이점
- 36: 궤적의 팔꿈치 특이점
- 37: 관절 각도가 180도 이상 변경됨

예:

```
로컬 상태=CheckGo(P1) if(상태==0)
```

그 다음에

이동(P1)

끝

조인트 모션 기본 설정을 통해 로봇 팔이 P1에 도달할 수 있는지 확인합니다. 가능하면 조인트 모션을 통해 P1으로 이동합니다.

체크무브

명령:

```
체크무브(P)
```

설명:

직선 운동 명령의 조작성을 확인하십시오.

필수 매개변수:

P: 사용자 정의되거나 포인트 페이지에서 얻은 목표 포인트. 데카르트 좌표점만 지원됩니다.

반품:

결과를 확인하십시오.

- 0: 오류 없음
- 16: 솔더 특이점에 가까운 끝점
- 17: 솔루션이 없는 끝점 역기구학 오류
- 18: 작업 영역을 벗어난 결과의 역기구학 오류
- 22: 팔 방향 오류
- 26: 손목 특이점에 가까운 끝점
- 27: 팔꿈치 특이점에 닫힌 끝점
- 29: 속도 매개변수 오류
- 32: 궤적의 어깨 특이점

- 33: 궤적에 솔루션이 없는 역기구학 오류
- 34: 궤적에서 작업 영역을 벗어난 결과의 역기구학 오류
- 35: 궤적의 손목 특이점
- 36: 궤적의 팔꿈치 특이점
- 37: 관절 각도가 180도 이상 변경됨

예:

```
로컬 상태=CheckMove(P1) if(상태==0)
```

그 다음에

이동(P1)

끝

선형 모션 기본 설정을 통해 로봇 팔이 P1에 도달할 수 있는지 확인하십시오. 가능하면 직선 운동을 통해 P1으로 이동하십시오.

상대 운동

동작 명령은 로봇 팔의 움직임을 제어하는 데 사용됩니다. 모션 속도 비율/가속 비율은 Motion 파라미터에 서도 설정할 수 있습니다. 모션 명령과 모션 파라미터 명령 모두에 파라미터를 설정하면 모션 명령의 값이 우선합니다.

실제 로봇 속도/가속 = 명령에서 설정된 비율 × 재생 설정에서 속도/가속 × 전체 속도 비율.

RP

명령:

```
RP(P, {오프셋X, 오프셋Y, 오프셋Z})
```

설명:

직교 좌표계 아래 점의 X축, Y축, Z축 오프셋을 설정하여 새로운 직교 좌표점을 반환합니다.

필수 매개변수:

- P1: TeachPoint 페이지에서 가져오거나 사용자 정의한 오프셋 앞의 지점입니다. 직교 좌표점만 지원됩니다. OffsetX, OffsetY, OffsetZ: 직교 좌표계에서
- X축, Y축, Z축 오프셋; 단위: 밀리미터

반품:

오프셋 후 데카르트 좌표점

예:

```
이동(RP(P1, {30,50,10}))
```

P1을 X축, Y축, Z축으로 각각 일정한 거리만큼 변위시킨 후,
오프셋.

알제이

명령:

```
RJ(P1, {오프셋1, 오프셋2, 오프셋3, 오프셋4, 오프셋5, 오프셋6})
```

설명:

관절 좌표계 아래 지정된 지점의 J1~J6 축의 각도 오프셋을 설정하고 새로운 관절 좌표점을 반환합니다.

필수 매개변수:

- P1: TeachPoint 페이지에서 얻을 수 있는 오프셋 이전의 포인트. 데카르트 좌표점만 지원됩니다.
- 오프셋1~오프셋6: J1~J6 축 오프셋. 단위: °

반품:

오프셋 후 관절 좌표점

예:

```
MoveJ(RJ(P1, {60,50,32,30,25,30}))
```

J1~J6 축의 각도 옵셋을 설정하고 옵셋 후 목표점으로 로봇팔을 이동시킨다.

GoR

명령:

```
GoR({OffsetX, OffsetY, OffsetZ}, "사용자=1 도구=2 CP=1 속도=50 가속=20 SYNC=1")
```

설명:

직교 좌표계에서 절대점 모드(관절 동작)로 현재 위치에서 오프셋 위치로 이동합니다. 관절 운동의 궤적은 선형이 아니며 모든 관절이 운동을 완료합니다.
동시에.

필수 매개변수:

OffsetX, OffsetY, OffsetZ: 직교 좌표계에서 X축, Y축 및 Z축 오프셋, 단위:
mm

선택적 매개변수:

- 사용자: 설정에 추가된 후 여기에서 사용할 수 있는 사용자 좌표계 인덱스입니다.
- 도구: 설정에 추가된 후 여기에서 사용할 수 있는 도구 좌표계 인덱스입니다.
- CP: 모션의 연속 경로 설정([모션 매개변수의 CP 명령 참조](#)), 범위: 0~100.
- 속도: 속도 비율, 범위: 1~100.
- Accel: 가속 비율, 범위: 1~100.
- SYNC: 동기화 플래그, 범위: 0 또는 1(기본값: 0).
 - SYNC=0: 비동기 실행, 즉 호출 후 즉시 반환,

실행 과정과 상관없이.

- SYNC=1: 동기 실행. 이는 명령이 완전히 실행될 때까지 호출된 후 반환되지 않음을 의미합니다.

예:

```
GoR({10,10,10})
```

로봇 팔은 기본 설정으로 관절 대 관절 모드에서 목표 지점으로 이동합니다.

발동력

명령:

```
MoveR({OffsetX, OffsetY, OffsetZ}, "사용자=1 도구=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
```

설명:

현재 위치에서 직교 좌표 아래 선형 모드에서 오프셋 위치로 이동
체계.

필수 매개변수:

OffsetX, OffsetY, OffsetZ: 직교 좌표계에서 X축, Y축 및 Z축 오프셋, 단위:
mm

선택적 매개변수:

- 사용자: 설정에 추가된 후 여기에서 사용할 수 있는 사용자 좌표계 인덱스입니다.
- 도구: 설정에 추가된 후 여기에서 사용할 수 있는 도구 좌표계 인덱스입니다.
- CP: 모션의 연속 경로 설정([모션 매개변수의 CP 명령 참조](#)), 범위: 0~100.
- SpeedS: 속도 비율, 범위: 1~100.
- AccelS: 가속 비율, 범위: 1~100.
- SYNC: 동기화 플래그, 범위: 0 또는 1(기본값: 0).
 - SYNC=0: 비동기 실행, 실행 프로세스와 관계없이 호출된 직후에 반환됨을 의미합니다.
 - SYNC=1: 동기 실행. 이는 명령이 완전히 실행될 때까지 호출된 후 반환되지 않음을 의미합니다.

예:

```
무버({10,10,10})
```

로봇팔은기본설정으로직선이동을통해목표지점으로이동합니다.

MoveJR

명령:

```
MoveJR({Offset1, Offset2, Offset3, Offset4, Offset5, Offset6}, "사용자=1 도구=2 CP=1 속도=50 가속=20 SYNC=1")
```

설명:

점대점 모드(관절 동작)에서 현재 위치에서 관절 오프셋 각도로 이동합니다.

필수 매개변수:

Offset1~Offset6: 직교 좌표계에서 J1 - J6 축 오프셋. 단위: °

선택적 매개변수:

- 사용자: 설정에 추가된 후 여기에서 사용할 수 있는 사용자 좌표계 인덱스입니다.
- 도구: 설정에 추가된 후 여기에서 사용할 수 있는 도구 좌표계 인덱스입니다.
- CP: 모션의 연속 경로 설정([모션 매개변수의 CP 명령 참조](#)), 범위: 0~100.
- 속도: 속도 비율, 범위: 1~100.
- Accel: 가속 비율, 범위: 1~100.
- SYNC: 동기화 플래그, 범위: 0 또는 1(기본값: 0).
 - SYNC=0: 비동기 실행, 실행 프로세스와 관계없이 호출된 직후에 반환됨을 의미합니다.
 - SYNC=1: 동기 실행. 이는 명령이 완전히 실행될 때까지 호출된 후 반환되지 않음을 의미합니다.

예:

```
MoveJR({20,20,10,0,10,0})
```

로봇 팔은 기본 설정으로 조인트 동작을 통해 오프셋 각도로 이동합니다.

IO

IO 명령은 시스템 IO를 읽고 쓰고 관련 매개변수를 설정하는 데 사용됩니다.

DI

명령:

```
DI(인덱스)
```

설명:

디지털 입력 포트의 상태를 가져옵니다.

필수 매개변수:

색인: DI 색인

반품:

해당 DI 포트의 레벨(ON/OFF)

예:

```
if (DI(1)==ON) 다음
    이동(P1)
    끝
```

로봇은 DI1의 상태가 ON일 때 리니어 모션으로 P1으로 이동합니다.

WaitDI

명령:

```
WaitDI(인덱스,ON|OFF,기간)
```

설명:

디지털 입력 포트의 상태를 가져옵니다. 상태가 지정된 상태와 일치하면 프로그램이 계속 실행됩니다. 그렇지 않으면 지정된 간격으로 디지털 입력 포트의 상태를 가져옵니다.

필수 매개변수:

- 색인: DI 색인

- 켜짐 | OFF: DI 포트의 상태. ON: 높은 수준; 꺼짐: 낮은 수준

선택적 매개변수:

- 기간: DI 상태를 얻기 위한 기간. 단위: ms, 기본적으로 50ms

예:

```
WaitDI(1, ON)  
이동(P1)
```

50ms마다 DI1의 상태를 가져옵니다. DI1이 높으면 프로그램이 계속 실행되고 로봇 팔이 선형 모드의 P1.

~하다

명령:

```
DO(인덱스,ON|OFF)
```

설명:

디지털 출력 포트의 상태를 설정합니다.

필수 매개변수:

- 인덱스: DO 인덱스
- ON/OFF: DO 포트의 상태. 켜기: 높은 수준; 꺼짐: 낮은 수준

예:

```
DO(1,ON)
```

DO1의 상태를 ON으로 설정합니다.

DOExecute

명령:

```
DOExecute(인덱스,ON|OFF)
```

설명:

현재 명령 대기열에 관계없이 즉시 디지털 출력 포트의 상태를 설정합니다.

필수 매개변수:

- 인덱스: DO 인덱스
- ON/OFF: DO 포트의 상태. 켜기: 높은 수준; 꺼짐: 낮은 수준

예:

```
DOExecute(1,ON)
```

현재 명령 대기열에 관계없이 즉시 DO1의 상태를 ON으로 설정합니다.

ToolDO

명령:

```
도구DO(인덱스,ON|OFF)
```

설명:

도구 디지털 출력 포트의 상태를 설정합니다.

필수 매개변수:

- 인덱스: 도구 DO 인덱스
- ON/OFF: DO 포트의 상태. 켜기: 높은 수준; 꺼짐: 낮은 수준

예:

```
도구DO(1,ON)
```

공구 DO1의 상태를 ON으로 설정합니다.

도구DOExecute

명령:

```
ToolDOExecute(인덱스,ON|OFF)
```

설명:

현재 명령 대기열에 관계없이 도구 디지털 출력 포트의 상태를 즉시 설정하십시오.

필수 매개변수:

- 인덱스: 도구 DO 인덱스
- ON/OFF: DO 포트의 상태. 켜기: 높은 수준; 꺼짐: 낮은 수준

예:

```
도구DOExecute(1,ON)
```

현재 명령 대기열에 관계없이 공구 DO1의 상태를 즉시 ON으로 설정하십시오.

도구DI

명령:

```
툴DI(인덱스)
```

설명:

도구 입력 포트의 상태를 가져옵니다.

필수 매개변수:

색인: 도구 DI 색인

반품:

해당 DI 포트의 레벨(ON/OFF)

예:

```
if (ToolDI(1)==ON) 다음  
이동(P1)  
끝
```

공구 DI1의 상태가 ON일 때 로봇은 리니어 모드에서 P1으로 이동합니다.

툴AI

명령:

```
ToolAI(인덱스)
```

설명:

도구 아날로그 입력 포트의 값을 가져옵니다. 사용하기 전에 ToolAnalogMode를 통해 포트를 전압 입력 모드로 설정해야 합니다.

필수 매개변수:

색인: 도구 AI 색인

반품:

해당 AI 인덱스 값

예:

```
테스트 = ToolAI(1)
```

도구 AI1의 값을 가져와 변수 "test"에 할당합니다.

도구아날로그 모드

명령:

```
도구아날로그모드(모드)
```

설명:

도구 아날로그 입력 포트의 상태를 설정합니다.

필수 매개변수:

모드: 아날로그 입력 포트의 모드

- 00: 기본, 485 모드
- 10: 전류 획득 모드 11:
- 0~3.3V 전압 입력 모드 12:
- 0~10V 전압 입력 모드

예:

```
도구아날로그모드(11)
```

공구 아날로그 입력 포트 상태를 0~3.3V 전압 입력 모드로 설정합니다.

AO

명령:

```
AO(인덱스, 값)
```

설명:

아날로그 출력 포트의 전압을 설정합니다.

필수 매개변수:

- 색인: AO 색인

- 값: 전압, 범위: 0~10

예:

AO(1,2)

AO1의 전압을 2V로 설정합니다.

AOExecute

명령:

AOExecute(인덱스, 값)

설명:

현재 명령 대기열에 관계없이 아날로그 출력 포트의 전압을 즉시 설정하십시오.

필수 매개변수:

- 색인: AO 색인
- 값: 전압, 범위: 0~10

예:

AO실행(1,2)

현재 명령 대기열에 관계없이 즉시 AO1의 전압을 2V로 설정합니다.

일체 포함

명령:

인공지능(인덱스)

설명:

아날로그 입력 포트의 전압을 가져옵니다.

필수 매개변수:

인덱스: AI 인덱스

반품:

해당 AI 인덱스 값

예:

```
테스트 = AI(1)
```

도구 AI1의 값을 가져와 변수 test에 할당합니다.

SetTool485

명령:

```
SetTool485(보드,파리티,스톱비트)
```

설명:

엔드 툴의 RS485 인터페이스에 해당하는 데이터 유형을 설정합니다.

필수 매개변수:

- baud: RS485 인터페이스의 전송 속도
- 패리티: 패리티 비트가 있는지 여부. "O"는 홀수, "E"는 짝수, "N"은 패리티 비트 없음을 의미합니다. stopbit: 정지 비트
- 길이. 범위: 1, 1.5, 2.

예:

```
SetTool485(115200,"N",1)
```

엔드 툴의 RS485 인터페이스에 해당하는 전송 속도를 115200Hz로, 패리티 비트를 N으로, 정지 비트 길이를 1로 설정합니다.

SetABZPPC

명령:

```
SetABZPPC(해상도)
```

설명:

ABZ 엔코더의 해상도를 설정합니다.

필수 매개변수:

resolution: 인코더의 해상도. 단위: 맥박/mm

예:

```
SetABZPPC(1000)
```

ABZ 엔코더의 분해능을 1000 펄스/mm로 설정합니다.

GetABZ

명령:

```
GetABZ()
```

설명:

구성된 ABZ 인코더의 해상도를 가져옵니다.

반품:

resolution: 인코더의 해상도. 단위: 맥박/mm

예:

```
로컬 abz = GetABZ()
```

구성된 ABZ 인코더의 해상도를 가져오고 변수 "abz"에 값을 할당합니다.

TCP/UDP

TCP/UDP 명령은 TCP/UDP 통신에 사용됩니다.

TCPCreate

명령:

```
TCPCreate(isServer, IP, 포트)
```

설명:

TCP 네트워크를 만듭니다. 하나의 TCP 네트워크만 지원됩니다.

필수 매개변수:

- isServer: 서버 생성 여부. true: 서버를 생성합니다. false: 클라이언트 생성
- IP: 충돌 없이 클라이언트의 동일한 네트워크 세그먼트에 있는 서버의 IP 주소입니다. 서버 생성 시 로봇 팔의 IP 주소, 클라이언트 생성 시 피어의 주소 만들어진.
- 포트: 서버 포트. 로봇이 서버 역할을 할 때 포트는 502 및 8080으로 설정할 수 없습니다. 그렇지 않으면 Modbus 기본 포트 또는 컨베이어 추적에 사용되는 포트와 충돌하여 TCP 네트워크 생성에 실패합니다.

반품:

- 오류:
 - 0: TCP 네트워크가 성공적으로 생성되었습니다.
 - 1: TCP 네트워크 생성 실패
- 소켓: 소켓 객체

예 1:

```
local ip="192.168.5.1" // 로봇의 IP 주소를 서버로 설정 local port=6001 // 서버 포트
로컬 오류=0
로컬 소켓=0
오류, 소켓 = TCPCreate(true, ip, port)
```

TCP 서버를 생성합니다.

예 2:

```
local ip="192.168.5.25" //카메라와 같은 외부 장비의 IP 주소를 서버로 설정
```

```
버전
로컬 포트=6001 //서버 포트
로컬 오류=0
로컬 소켓=0
오류, 소켓 = TCPCreate(거짓, IP, 포트)
```

TCP 클라이언트를 생성합니다.

TCP시작

명령:

```
TCPStart(소켓, 타임아웃)
```

설명:

TCP 연결을 설정합니다. 로봇 팔은 서버 역할을 할 때 클라이언트와 연결되기를 기다리고 클라이언트 역할을 할 때 서버에 연결합니다.

필수 매개변수:

- 소켓: 소켓 개체 시간
- 초과: 대기 시간 초과. 단위: 에스. timeout이 0이면 연결이 성공적으로 설정될 때까지 기다립니다.
그렇지 않은 경우 시간 초과 후 연결 실패를 반환하고,

반품:

연결 결과.

- 0: TCP 연결 성공
- 1: 입력 매개변수가 잘못됨 2: 소켓 개체를 찾을 수 없음 3: 시
- 간 초과 설정이 잘못됨 4: 연결
- 실패

예:

```
err = TCPStart(socket, 0) // 소켓은 TCPCreate에 의해 반환된 소켓 개체입니다.
```

연결이 성공할 때까지 TCP 연결 설정을 시작합니다.

TCP읽기

명령:

```
TCPRead(소켓, 타임아웃, 타입)
```

설명:

TCP 피어에서 데이터를 수신합니다.

필수 매개변수:

- 소켓: 소켓 객체

선택적 매개변수:

- 타임아웃: 대기 타임아웃. 단위: 에스. timeout이 0이면 실행하기 전에 데이터를 완전히 읽을 때까지 기다리십시오. 그렇지 않은 경우 제한 시간을 초과한 후 계속 실행합니다. 유형: 반환
- 값의 유형. type이 설정되지 않은 경우 RecBuf의 버퍼 형식은 테이블입니다. type이 문자열로 설정된 경우 버퍼 형식은 문자열입니다.

반품:

- 오류:
 - 0: 데이터가 성공적으로 수신되었습니다.
 - 1: 데이터 수신에 실패했습니다.
- Recbuf: 데이터 버퍼

예:

```
// socket은 TCPCreate err에 의해 반환된 소켓 객체입니다. RecBuf =  
TCPRead(socket,0,"string") // RecBuf의 데이터 유형은 string err입니다. RecBuf = TCPRead(socket, 0) // RecBuf  
의 데이터 유형입니다. 테이블이다
```

TCP 데이터를 수신하고 데이터를 각각 문자열 및 테이블 형식으로 저장합니다.

TCP쓰기

명령:

```
TCPWrite(소켓, 버프, 타임아웃)
```

설명:

TCP 피어에 데이터를 보냅니다.

필수 매개변수:

- socket: 소켓 객체
- buf: 보낸 데이터

선택적 매개변수:

타임아웃: 대기 타임아웃. 단위: 에스. timeout이 0이면 피어가 데이터를 수신할 때까지 프로그램이 계속 실행되지 않습니다. timeout이 0이 아닌 경우 시간 초과 후에도 프로그램이 계속 실행됩니다.

반품:

데이터 전송 결과입니다.

- 0: 데이터 전송 성공 1: 데이터 전송 실패.
-

예:

```
TCPWrite(socket, "test") // 소켓은 TCPCreate에 의해 반환된 소켓 개체입니다.
```

TCP 데이터 "테스트"를 보냅니다.

TCP파괴

명령:

```
TCPDestroy(소켓)
```

설명:

TCP 네트워크 연결을 끊고 소켓 개체를 파괴합니다.

필수 매개변수:

소켓: 소켓 개체

반품:

실행 결과.

- 0: 성공적으로 실행되었습니다.
- 1: 실행에 실패했습니다.

예:

```
TCPDestroy(socket) // 소켓은 TCPCreate에 의해 반환된 소켓 개체입니다.
```

TCP 피어와의 연결을 끊습니다.

UDP만들기

명령:

```
UDPCreate(isServer, IP, 포트)
```

설명:

UDP 네트워크를 만듭니다. 하나의 UDP 네트워크만 지원됩니다.

필수 매개변수:

- isServer: 거짓
- IP: 충돌이 없는 클라이언트의 동일한 네트워크 세그먼트에 있는 피어의 IP 주소 포트: 피어 포트
-

반품:

- 오류:
 - 0: UDP 네트워크가 성공적으로 생성되었습니다.
 - 1: UDP 네트워크 생성 실패
- 소켓: 소켓 객체

예:

```
local ip="192.168.5.25" //카메라와 같은 외부 장치의 IP를 피어의 IP 주소로 설정

로컬 포트=6001 //피어 포트
로컬 오류=0
로컬 소켓=0

오류, 소켓 = UDPCreate(false, ip, port)
```

UDP 네트워크를 만듭니다.

UDP읽기

명령:

```
UDPRead(소켓, 타임아웃, 타입)
```

설명:

UDP 피어에서 데이터를 수신합니다.

필수 매개변수:

- 소켓: 소켓 객체

선택적 매개변수:

- 타임아웃: 대기 타임아웃. 단위: 에스. timeout이 0이면 실행하기 전에 데이터를 완전히 읽을 때까지 기다리십시오. 그렇지 않은 경우 제한 시간을 초과한 후 계속 실행합니다. 유형: 반환
- 값의 유형. type이 설정되지 않은 경우 RecBuf의 버퍼 형식은 테이블입니다. type이 문자열로 설정된 경우 버퍼 형식은 문자열입니다.

반품:

- 오류:
 - 0: 데이터가 성공적으로 수신되었습니다.
 - 1: 데이터 수신에 실패했습니다.
- Recbuf: 데이터 버퍼

예:

```
// 소켓은 UDPCreate에 의해 반환된 소켓 객체 err, RecBuf =
UDPRRead(socket,0,"string") // RecBuf의 데이터 유형은 문자열 err, RecBuf = UDPRead(socket, 0) // RecBuf의 데이터 유형 테이블이다
```

UDP 데이터를 수신하고 데이터를 각각 문자열 및 테이블 형식으로 저장합니다.

UDP쓰기

명령:

```
UDPPWrite(소켓, 버프, 타임아웃)
```

설명:

UDP 피어에 데이터를 보냅니다.

필수 매개변수:

- socket: 소켓 객체
- buf: 로봇이 보낸 데이터

선택적 매개변수:

타임아웃: 대기 타임아웃. 단위: 에스. timeout이 0이면 피어가 데이터를 수신할 때까지 프로그램이 계속 실행되지 않습니다. timeout이 0이 아닌 경우 시간 초과 후에도 프로그램이 계속 실행됩니다.

반품:

데이터 전송 결과입니다.

- 0: 데이터가 성공적으로 전송되었습니다.
- 1: 데이터 전송에 실패했습니다.

예:

```
UDPPWrite(socket, "test") // 소켓은 UDPCreate에 의해 반환된 소켓 개체입니다.
```

UDP 데이터 "테스트"를 보냅니다.

모드버스

Modbus 명령은 Modbus 통신에 사용됩니다.

모드버스만들기

명령:

```
모드버스 생성()
```

설명:

Modbus 마스터 스테이션을 생성하고 슬레이브 스테이션과 연결을 설정합니다.

필수 매개변수:

- IP: 슬레이브 스테이션의 IP 주소. IP를 지정하지 않거나 127.0.0.1 또는 0.0.0.1인 경우 로컬 Modbus 슬레이브에 연결되었음을 나타냅니다. port: 슬레이브 포트
- 브 스테이션 포트 slave_id:
- 슬레이브 스테이션의 ID. 범위: 1~4

반품:

- 오류:
 - 0: Modbus 마스터 스테이션이 성공적으로 생성되었습니다.
 - 1: Modbus 마스터 스테이션 생성에 실패했습니다.
- id: 슬레이브 스테이션의 장치 ID

예 1:

```
로컬 ip="192.168.5.123" //슬레이브 ID
로컬 포트=503 //슬레이브 포트
로컬 오류=0
로컬 ID=0
오류, id = ModbusCreate(ip, 포트, 1)
```

Modbus 마스터를 생성하고 지정된 슬레이브와 연결합니다.

예 1:

다음 명령은 모두 Modbus 슬레이브 스테이션 연결을 나타냅니다.

```
모드버스 생성()
```

모드버스생성("127.0.0.1")

모드버스생성("0.0.0.1")

ModbusCreate("127.0.0.1",xxx,xxx) // xxx 임의의 값

ModbusCreate("0.0.0.1",xxx,xxx) // xxx 임의의 값

GetInBits

명령:

GetInBits(ID, 주소, 개수)

설명:

Modbus 슬레이브에서 개별 입력 값 읽기

필수 매개변수:

- id: 슬레이브 ID
- addr: 이산 입력의 시작 주소, 범위: 0~4095 카운트: 이산 입력의 수
-

반품:

테이블에 저장된 이산 입력 값, 여기서 테이블의 첫 번째 값은 시작 주소의 이산 입력 값에 해당합니다.

예:

inBits = GetInBits(id,0,5)

주소 0부터 시작하여 5개의 개별 입력을 읽습니다.

GetInRegs

명령:

GetInRegs(ID, 주소, 개수, 유형)

설명:

Modbus 슬레이브에서 지정된 데이터 유형으로 입력 레지스터 값을 읽습니다.

필수 매개변수:

- id: 슬레이브 ID
- addr: 입력 레지스터의 시작 주소, 범위: 0~4095 카운트: 입력 레지스터 값의
- 개수, 범위: 0~4096

선택적 매개변수:

유형: 데이터 유형

- 비어 있음: 기본적으로 U16
- U16: 16비트 부호 없는 정수(2바이트, 하나의 레지스터 점유)
- U32: 32비트 부호 없는 정수(4바이트, 2개의 레지스터 점유)
- F32: 32비트 단정밀도 부동 소수점 숫자(4바이트, 2개의 레지스터 점유)
- F64: 64비트 배정밀도 부동 소수점 숫자(8바이트, 4개의 레지스터 점유)

반품:

테이블에 저장된 입력 레지스터 값, 여기서 첫 번째 값은 시작 주소의 입력 레지스터 값에 해당합니다.

예:

```
데이터 = GetInRegs(id, 2048, 1, "U32")
```

주소 2048에서 시작하여 부호 없는 32비트 정수를 읽습니다.

GetCoils

명령:

```
GetCoils(ID, 주소, 개수)
```

설명:

Modbus 슬레이브에서 코일 레지스터 값을 읽습니다.

필수 매개변수:

- id: 슬레이브 ID
- addr: 코일 레지스터의 시작 주소, 범위: 0~4095 카운트: 코일 레지스터 값
- 의 개수

반품:

테이블에 저장된 코일 레지스터 값, 여기서 첫 번째 값은 시작 주소의 코일 레지스터 값에 해당합니다.

예:

```
코일 = GetCoils(id,0,5)
```

0번지부터 연속해서 5개의 값을 읽는다.

GetHoldRegs

명령:

```
GetHoldRegs(ID, 주소, 개수, 유형)
```

설명:

Modbus 슬레이브에서 지정된 데이터 유형으로 홀딩 레지스터 값을 읽습니다.

필수 매개변수:

- id: 슬레이브 ID
- addr: 홀딩 레지스터의 시작 주소, 범위: 0~4095 카운트: 홀딩 레지스터 값의 개수
- 수

선택적 매개변수:

유형: 데이터 유형

- 비어 있음: 기본적으로 U16
- U16: 16비트 부호 없는 정수(2바이트, 하나의 레지스터 점유)
- U32: 32비트 부호 없는 정수(4바이트, 2개의 레지스터 점유)
- F32: 32비트 단정밀도 부동 소수점 숫자(4바이트, 2개의 레지스터 점유)
- F64: 64비트 배정밀도 부동 소수점 숫자(8바이트, 4개의 레지스터 점유)

반품:

테이블에 저장된 홀딩 레지스터 값, 여기서 첫 번째 값은 시작 주소의 홀딩 레지스터 값에 해당합니다.

예:

```
데이터 = GetHoldRegs(id, 2048, 1, "U32")
```

주소 2048에서 시작하여 부호 없는 32비트 정수를 읽습니다.

세트코일

명령:

```
SetCoils(id, addr, count, table)
```

설명:

코일 레지스터의 지정된 주소에 지정된 값을 쓰습니다.

필수 매개변수:

- id: 슬레이브 ID
- addr: 코일 레지스터의 시작 주소, 범위: 6~4095 카운트: 코일 레지스터에 쓸 값의 개수, 범위: 0 ~ 4096 테이블: 코일 레지스터에 쓸 값을 저장합니다.
- 테이블의 첫 번째 값은 코일 레지스터의 시작 주소에 해당합니다.

예:

```
로컬 코일 = {0,1,1,1,0}
SetCoils(id, 1024, #coils, 코일)
```

1024번지부터 코일레지스터에 연속해서 5개의 값을 쓴다.

SetHoldRegs

명령:

```
SetHoldRegs(id, addr, count, table, type)
```

설명:

지정된 데이터 유형에 따라 지정된 값을 홀딩 레지스터의 지정된 주소에 쓹습니다.

필수 매개변수:

- id: 슬레이브 ID
- addr: 홀딩 레지스터의 시작 주소, 범위: 0~4095 카운트: 홀딩 레지스터에 쓸 값의 수 테이블: 코일 레지스터에 쓸 값을 저장합니다. 테이블의 첫 번째 값은 홀딩 레지스터의 시작 주소에 해당합니다.

선택적 매개변수:

유형: 데이터 유형

- 비어 있음: 기본적으로 U16
- U16: 16비트 부호 없는 정수(2바이트, 하나의 레지스터 점유)
- U32: 32비트 부호 없는 정수(4바이트, 2개의 레지스터 점유)
- F32: 32비트 단정밀도 부동 소수점 숫자(4바이트, 2개의 레지스터 점유)
- F64: 64비트 배정밀도 부동 소수점 숫자(8바이트, 4개의 레지스터 점유)

예:

```
로컬 데이터 = {95.32105}
SetHoldRegs(ID, 2048, #데이터, 데이터, "F64")
```

주소 2048에서 시작하여 헤딩 레지스터에 배정밀도 부동 소수점 숫자를 씁니다.

모드버스닫기

명령:

```
모드버스닫기(id)
```

설명:

Modbus 슬레이브 스테이션과 연결을 끊습니다.

선택적 매개변수:

id: 슬레이브 ID

반품:

- 0: Modbus 슬레이브가 성공적으로 연결 해제되었습니다.
- 1: Modbus 슬레이브 연결 해제에 실패했습니다.

예:

```
ModbusClose(id) // id는 ModbusCreate에 의해 반환된 슬레이브 ID입니다.
```

Modbus 슬레이브와의 연결을 끊습니다.

프로그램 제어

프로그램 제어 명령은 프로그램 제어와 관련된 일반적인 명령입니다. while, if 및 for는 루아의 흐름 제어 명령입니다. [루아 기본 문법 - 프로세스 제어](#)를 참고하세요. 인쇄는 콘솔에 정보를 출력하는 데 사용됩니다.

잠

명령:

수면 시간)

설명:

다음 명령의 실행을 지연시킵니다.

필수 매개변수:

시간: 지역 시간, 단위: ms

예:

DO(1,ON)
수면(100)
DO(1,OFF)

DO1을 ON으로 설정하고 100ms를 기다린 다음 DO1을 OFF로 설정합니다.

기다리다

명령:

기다리는 시간)

설명:

모션 명령을 지연하여 전달하거나 현재 모션이 완료된 후 지연하여 다음 명령을 전달합니다.

필수 매개변수:

시간: 지역 시간, 단위: ms

예:

```
DO(1,ON)
대기(100)
이동(P1)
대기(100)
DO(1,OFF)
```

DO1을 ON으로 설정하고 100ms를 기다린 다음 로봇을 P1으로 이동합니다. 100ms 지연 후 DO1을 OFF로 설정합니다.

정지시키다

명령:

```
정지시키다()
```

설명:

프로그램 실행을 일시 중지합니다. 프로그램은 소프트웨어 제어 또는 원격을 통해서만 계속 실행할 수 있습니다.
제어.

예:

```
이동(P1)
정지시키다()
이동(P2)
```

로봇이 P1으로 이동한 다음 실행을 일시 중지합니다. 외부를 통해서만 P2로 계속 이동할 수 있습니다.
제어.

충돌 수준 설정

명령:

```
SetCollisionLevel(레벨)
```

설명:

충돌 감지 수준을 설정합니다. 이 인터페이스를 통해 설정된 충돌 감지 수준은 프로젝트가 실행 중일 때만
유효하며 프로젝트가 중지된 후 이전 값으로 복원됩니다.

필수 매개변수:

수준: 충돌 감지 수준, 범위: 0~5. 0은 충돌 감지를 끄는 것을 의미합니다. 1에서 5까지의 수준이 높을수록 충돌 감지가 더
민감합니다.

예:

```
SetCollisionLevel(2)
```

충돌 감지를 레벨 2로 설정합니다.

재설정 경과 시간

명령:

```
재설정 경과 시간()
```

설명:

이 명령이 완전히 실행되기 전에 모든 명령이 실행된 후 타이밍을 시작하십시오. 이 명령은 작동 시간을 계산하기 위해 ElapsedTime() 명령과 함께 사용해야 합니다.

예:

ElapsedTime의 예를 참조하십시오.

경과 시간

명령:

```
경과 시간()
```

설명:

타이밍을 멈추고 시차를 반환합니다. 이 명령은 ResetElapsedTime() 명령과 함께 사용해야 합니다.

반품:

타이밍의 시작과 끝 사이의 시간.

예:

```
Go(P2, " 속도=100 가속=100")
ResetElapsedTime() for
i=1,10 do
    점프(P1, " 속도=100 가속=100 시작=0 종료=0 ZLimit=185")
    점프(P2, " 속도=100 가속=100 시작=0 종료=0 ZLimit=185")
    끝
인쇄 (경과 시간())
```

로봇팔이 P1과 P2 사이를 10번 왕복하는 시간을 계산하고 출력한다.

콘솔.

시스템타임

명령:

```
시스템타임()
```

설명:

현재 시스템 시간을 가져옵니다.

반품:

현재 시간의 Unix 타임스탬프

예:

```
현지 시간 = Systime()
```

현재 시스템 시간을 가져와 변수 "time"에 저장합니다.

사용자 설정

명령:

```
SetUser(인덱스,테이블)
```

설명:

지정된 사용자 좌표계를 수정합니다. 수정은 프로젝트가 실행 중일 때만 유효하며 좌표계는 프로젝트가 종지된 후 이전 값을 복원합니다.

필수 매개변수:

- index: 사용자 좌표계 인덱스, 범위: 0~9 (0은 기본 사용자 좌표계) table: 사용자 좌표계 매트릭스, {x, y, z, rx, ry, rz} 형식

예:

```
SetUser(1,{10,10,10,0,0,0})
```

사용자 좌표계 1을 "X=10,Y=10,Z=10,RX=0,RY=0,RZ=0"으로 수정합니다.

SetTool

명령:

```
SetTool(인덱스,테이블)
```

설명:

지정된 공구 좌표계를 수정합니다. 수정은 프로젝트가 실행 중일 때만 유효하며 좌표계는 프로젝트가 중지된 후 이전 값을 복원합니다.

필수 매개변수:

- 색인: 도구 좌표계 색인, 범위: 0~9(0은 기본 사용자 좌표계) 표: 도구 좌표계용 행렬, {x, y, z, rx, ry, rz}
- 형식

예:

```
SetTool(1,{10,10,10,0,0,0})
```

공구 좌표계 1을 "X=10,Y=10,Z=10,RX=0,RY=0,RZ=0"으로 수정합니다.

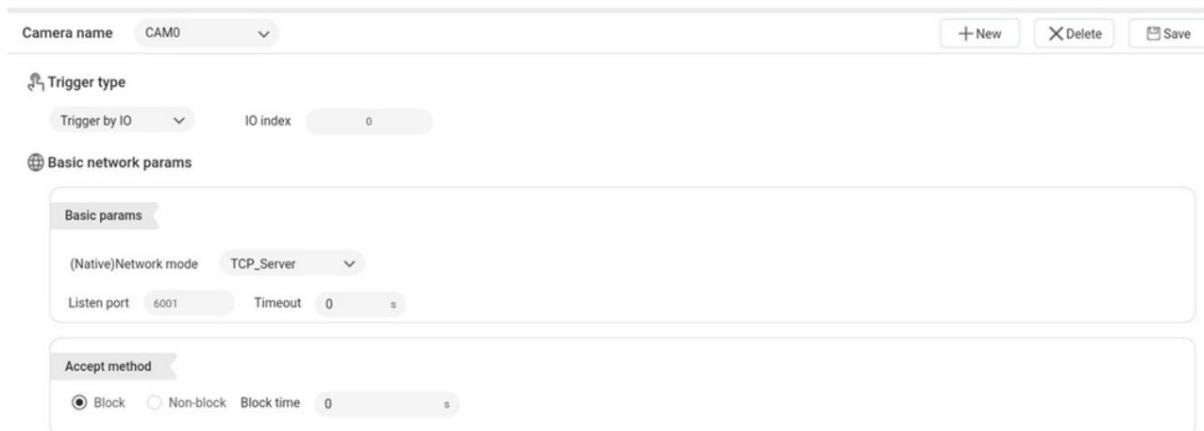
비전

비전 모듈은 관련 카메라 설정을 구성하는 데 사용됩니다. 카메라는 로봇의 작동 범위 내에 고정됩니다. 그 위치와 시야는 고정되어 있습니다. 카메라는 로봇의 눈 역할을 하며 이더넷 통신 또는 I/O 트리거링을 통해 로봇과 상호 작용합니다.

카메라 설치 및 구성 방법은 카메라마다 다릅니다. 이 섹션에서는 자세히 설명하지 않습니다.

비전 프로세스 구성

Vision 명령 오른쪽에 있는 Vision Config를 클릭하여 카메라 구성 시작합니다. 카메라를 처음 구성하는 경우 새로 만들기 를 클릭하고 카메라 이름을 입력하여 카메라 구성 생성합니다. 그러면 다음 페이지가 표시됩니다.



트리거 유형

카메라를 트리거할 유형을 설정합니다.

- IO에 의한 트리거: 카메라를 로봇의 DO 인터페이스에 연결합니다. 전기 배선 포트에 따라 해당 출력 포트를 구성해야 합니다.

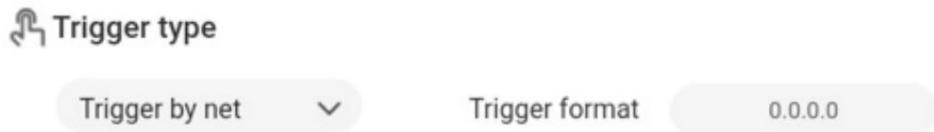
Trigger type

Trigger by IO

IO index

0

- 네트워크로 트리거: 카메라를 로봇의 이더넷 포트에 연결합니다. 카메라를 트리거하기 위해 로봇이 네트워크를 통해 보내는 문자열을 구성해야 합니다.



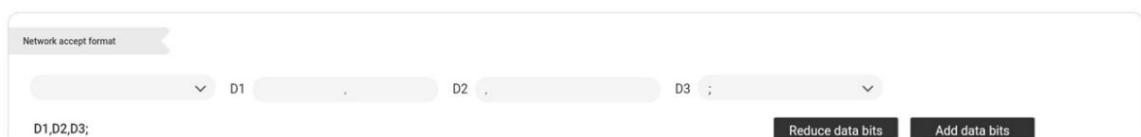
기본 네트워크 매개변수

기본 네트워크 매개변수는 다음 모드를 포함하여 카메라와 로봇 사이의 통신 모드를 설정하는 데 사용됩니다.

- UDP_Client: UDP 통신. 로봇은 클라이언트 역할을 하고 카메라는 서버 역할을 합니다. 카메라의 IP 주소와 포트를 구성해야 합니다.
- UDP_Server: UDP 통신. 로봇은 서버 역할을 하고 카메라는 클라이언트 역할을 합니다. 포트 및 시간 초과를 구성해야 합니다.
- TCP_Client: TCP 통신. 로봇은 클라이언트 역할을 하고 카메라는 서버 역할을 합니다. 카메라의 IP 주소, 포트 및 시간 제한을 구성해야 합니다.
- TCP_Server: TCP 통신. 로봇은 서버 역할을 하고 카메라는 클라이언트 역할을 합니다. 카메라의 포트와 타임아웃을 구성해야 합니다.

수신 방법에는 블록 및 비블록의 두 가지 모드가 있습니다. 프로젝트 스크립트에 따라 선택하세요.

- 블록: 트리거 신호를 보낸 후 프로그램은 블록 시간 동안 데이터 수신 라인에 머물며 카메라에서 보낸 데이터를 받을 때까지 프로그램이 계속 실행됩니다. 차단 시간을 0으로 설정하면 프로그램은 카메라에서 보낸 데이터를 받을 때까지 데이터 수신 라인에서 대기합니다.
- Non-block: 트리거 신호를 보낸 후 프로그램은 계속해서 실행됩니다. 카메라에서 데이터를 수신하거나 수신하지 않습니다.



네트워크 수락 형식은 구문 분석에 사용되는 카메라가 보낸 데이터 유형을 나타냅니다. 현재 기본 데이터 비트가 충분하지 않은 경우 데이터 비트 추가를 클릭하여 수신된 데이터의 길이를 최대 8비트(No, D1, D2, D3, D4, D5, D6, STA)로 늘릴 수 있습니다. 여기서 No는 비트를 나타냅니다. 시작 비트 템플릿 번호, STA는 종료 비트(상태 비트)를 나타냅니다.

다음과 같은 다양한 데이터 형식을 설정할 수 있습니다.

- 시작 비트 및 종료 비트 없음: XX, YY, CC;
- 시작 비트는 있지만 끝 비트는 없음: No, XX, YY, CC;
- 시작 비트는 없지만 종료 비트: XX, YY, CC, STA;
- 시작 비트와 종료 비트 포함: No, XX, YY, CC, STA;

구성 후 오른쪽 상단 모서리에 있는 저장을 클릭합니다.

초기화캠

명령:

```
초기화캠(CAM)
```

설명:

지정된 카메라에 연결하고 초기화합니다.

필수 매개변수:

CAM: 비전 프로세스에서 구성된 카메라와 일치해야 하는 카메라의 이름

반품:

초기화 결과.

- 0: 초기화 성공
- 1: 초기화 실패

예:

```
초기화캠("CAM0")
```

CAM0 카메라에 연결하고 초기화합니다.

TriggerCam

명령:

```
트리거캠(CAM)
```

설명:

초기화된 카메라를 트리거하여 사진을 찍습니다.

필수 매개변수:

CAM: 비전 프로세스에서 구성된 카메라와 일치해야 하는 카메라의 이름

반품:

결과를 트리거합니다.

- 0: 성공적으로 트리거
- 1: 트리거 실패

예:

```
TriggerCam("CAM0")
```

CAM0 카메라를 트리거하여 사진을 찍습니다.

센드캠

명령:

```
SendCam(CAM,데이터)
```

설명:

초기화된 카메라로 데이터를 보냅니다.

필수 매개변수:

- CAM: 비전에 구성된 카메라와 일치해야 하는 카메라의 이름
프로세스
- 데이터: 카메라로 전송된 데이터

반품:

데이터 전송 결과입니다.

- 0: 성공적으로 전송
- 1: 전송 실패

예:

```
SendCam("CAM0","0,0,0,0")
```

CAM0 카메라에 데이터("0,0,0,0")를 보냅니다.

RecvCam

명령:

```
RecvCam(CAM,유형)
```

설명:

초기화된 카메라에서 데이터를 받습니다.

필수 매개변수:

CAM: 비전 프로세스에서 구성된 카메라와 일치해야 하는 카메라의 이름

선택적 매개변수:

유형: 데이터 유형, 값 범위: 숫자 또는 문자열(기본값은 숫자)

반품:

- 오류: 오류 코드
 - 0: 데이터를 올바르게 수신
 - 1: 시간 초과
 - 2: 구문 분석할 수 없는 잘못된 데이터 형식
 - 3: 네트워크 연결 끊김
- n: 카메라가 보낸 데이터 그룹의 수.
- 데이터: 카메라에서 보낸 데이터는 2차원 배열에 저장됩니다.

예:

```
local err,n,data = RecvCam("CAM0","숫자")
```

CAM0 카메라로부터 데이터를 수신하며 데이터 타입은 숫자입니다.

DestroyCam

명령:

```
디스트로이캠(CAM)
```

설명:

카메라와의 연결을 해제합니다.

필수 매개변수:

CAM: 비전 프로세스에서 구성된 카메라와 일치해야 하는 카메라의 이름

반품:

- 0: 카메라가 연결 해제되었습니다.
- 1: 카메라 연결 해제에 실패했습니다.

예:

```
DestroyCam("CAM0")
```

카메라 CAM0과의 연결을 해제합니다.

예

비전 매개변수를 설정한 후 카메라에서 데이터를 수신하도록 프로그래밍하기 위해 비전 API를 호출할 수 있습니다. 아래 데모는 CAM0에서 데이터를 가져와 포인트 2에 값을 할당하는 방법에 관한 것입니다.

```

while true do
    ::create_camera::
    resultInit = InitCam("CAM0")                                --Connect CAM0 camera
    if resultInit ~= 0 then
        print("Connect camera failed, code:", resultInit)
        Sleep(1000)
        goto create_camera
    end
    while true do
        TriggerCam("CAM0")
        SendCam("CAM0", "1,2,3,0;")
        err,visionNum,visionData = RecvCam("CAM0","number")
        if err ~= 0 then
            print("Failed to read data")
            Sleep(1000)
            break
        end

        print("(visionNum):", (visionNum))                      --Print how many sets of CAM0 camera data received
        print("(visionData[1][1]):", (visionData[1][1]))          --Print the first data of the first group received
        i = 1
        while not ((visionNum) < i) do
            print(type(P2.coordinate[1]))
            print(P2)
            P2.coordinate[1]=(visionData[i][1])
            P2.coordinate[2]=(visionData[i][2])
            Go(P2,"SYNC=1")
            i = i + 1
        end
        Sleep(10)
    end
    Sleep(10)
end

```