Part 01

CDP Studio: Control a robot arm

Use CDP Studio and its Kinematics framework to control a Raspberry Pi-based robot arm

MAKER
Phil King

Long-time contributor to *The MagPi*, Phil is a freelance writer and editor with a focus on technology.

@philkingeditor

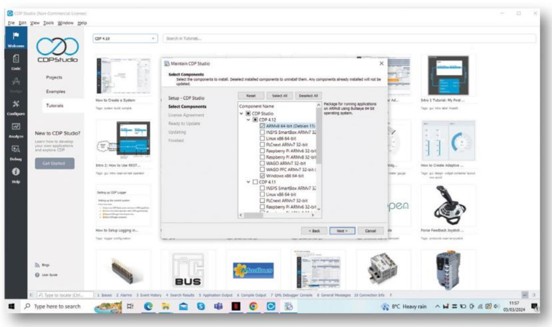
CDP Studio is an 'out of the box' software development tool used by many companies to build industrial control, automation, and edge systems. Yet it's fairly easy to get to grips with its low-code programming environment.

In The MagPi issues 116 and 117, we showed you how to deploy a couple of projects to a Raspberry Pi. This time, we'll be using CDP Studio and its Kinematics framework to program a Raspberry Pi-powered robot arm, the myCobot 280 Pi that we reviewed in issue 137 (**magpi.cc/137**), to perform a pick and place routine. If you don't have the robot arm, you can still run the project and record movement steps to see how they affect the position of a virtual arm shown on screen.

01 Install the software

On your PC, visit **cdpstudio.com/getstarted** and download the free non-commercial version for Windows or Linux. During installation, select the 'ARMv8 64-bit (Debian 11)' component under CDP Studio 4.12, along with the one already ticked for your host PC. You will then be able to deploy projects to the myCobot 280 Pi arm, which uses a 64-bit version of Ubuntu.

If you already have CDP Studio installed, make sure it's updated to version 4.12, then go to Help > Package Manager and select 'Add or remove CDP versions' to add the ARMv8 64-bit (Debian 11) component.



Maintain CDP Studio

Select Components
Select the components to install. Deselect installed components to uninstall them. Any components already installed will not be updated.

Package for running applications on ARMv8 using Bullseye 64-bit operating system.

Adding the ARMv8 64-bit (Debian 11) toolkit required to control the myCobot robot arm

02 Download the project

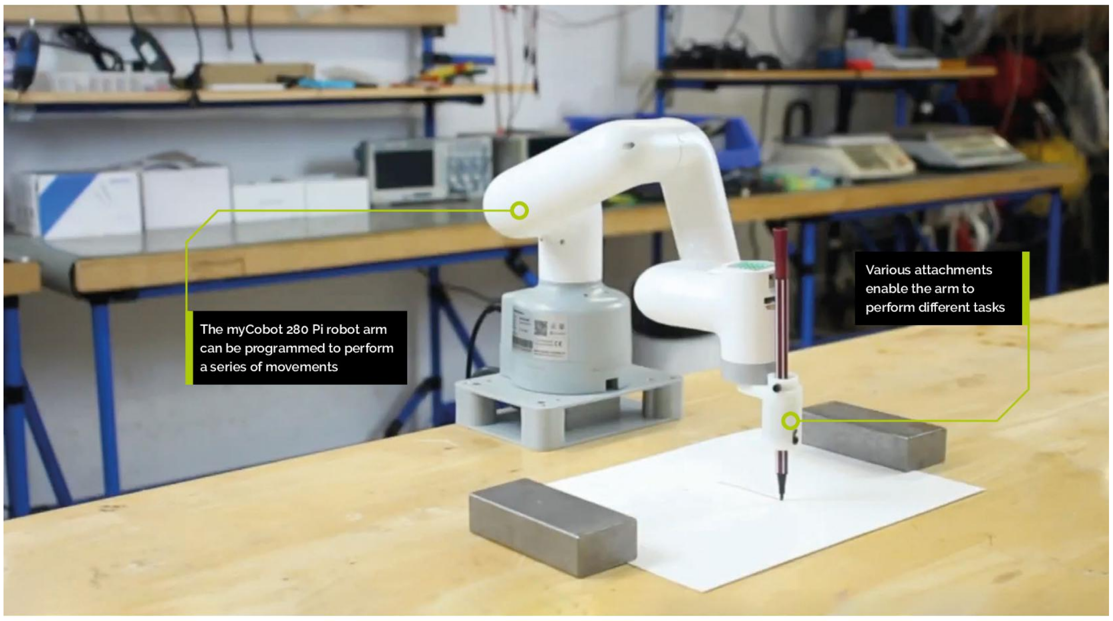
This is a complex project that would be time-consuming to build from scratch, so we'll download it from CDP Studio's GitHub repo. Go to **magpi.cc/recordnplay**, click the green Code button, and select Download ZIP. Unzip the file on your PC. Move the resulting **myCobotRecordNPlay-main** folder to the **CDPStudioWorkspace/systems** folder.

03 Download and build library

To deploy the project to the robot arm, you'll also need the myCobotLib library. Go to the GitHub repo at **magpi.cc/mycobotlib**, click the Code button and Download ZIP. Extract it and then move the resulting **myCobotLib-main** folder to the **CDPStudioWorkspace/libraries** folder. Open the **myCobotLib** project file (with the .pro suffix) in CDPStudio, then right-click its name in the left panel and select Build.

04 Open the project

Now open the **RecordNPlay** CDP project file (.pro) in CDP Studio. If you click the arrow


The myCobot 280 Pi robot arm can be programmed to perform a series of movements
Various attachments enable the arm to perform different tasks

next to it in the left panel, you'll note that it comprises two main applications. RecordNPlayUI runs the database logic for recording arm movement steps and shows a GUI on the PC to make programming the arm easier. It also has an ArmVisualizer pane that can be used to view the arm positions in 3D. This can be used even if you don't have a real arm connected, so you can still run the project and see how recorded steps affect its movements.

The RecordNPlayIO application is the part of the project that's deployed to the myCobot Pi arm over the network, once paired, enabling CDP Studio to communicate with it.

05 Prepare myCobot

The myCobot arm's Ubuntu OS has a non-standard version of the OpenSSH server. So you'll need to make a small change to a config file so CDP Studio can communicate with it over the network. SSH into the myCobot with the username 'er' at its IP address; the default password is 'Elephant'. Then enter:

```
sudo nano /etc/ssh/sshd_config
```

Locate the line that sets the PubkeyAuthentication parameter and set it to yes (and make sure the line is not commented out). Press **CTRL+X**, then **Y** to exit and save. Then restart the OpenSSH server with:

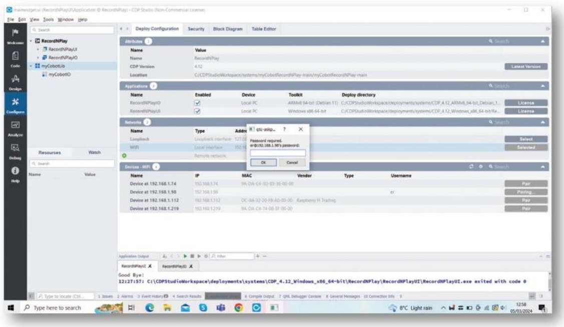
```
sudo systemctl restart sshd
```

06 Pair the arm

Open the Deploy Configuration tab. Under Networks, press the Select button for 'WiFi'. The 'Devices – WiFi' table below should start showing any devices available to pair with CDP Studio.

You'll Need

- Windows or Linux PC
- CDP Studio 4.12 **cdpstudio.com/getstarted**
- myCobot 280 Pi robot arm **magpi.cc/mycobot280**
- myCobot Adaptive Gripper **magpi.cc/mycobotgripper**



Deploy Configuration | Security | Block Diagram | Table Editor

Name	Value
Name	RecordNPlay
CDP Version	4.12
Location	C:/CDPStudioWorkspace/systems/myCobotRecordNPlay-main/myCobotRecordNPlay-main

Name	Enabled	Device	Toolkit	Deploy directory
RecordNPlayIO	✓	Local PC	ARMv8 64-bit (Debian 11)	C:/CDPStudioWorkspace/deployments/systems/CDP_4.12_ARMv8_64-bit_Debian_1...
RecordNPlayUI	✓	Local PC	Windows x86 64-bit	C:/CDPStudioWorkspace/deployments/systems/CDP_4.12_Windows_x86_64-bit/Re...

Password required.
er@192.168.1.98's password:

Name	IP	MAC	Vendor	Type	Username
Device at 192.168.1.74	192.168.1.74	9A-DA-C4-3D-03-36-00-00			
Device at 192.168.1.98	192.168.1.98				er
Device at 192.168.1.112	192.168.1.112	DC-A6-32-20-F8-A0-00-00	Raspberry Pi Trading		
Device at 192.168.1.219	192.168.1.219	9A-DA-C4-74-08-5F-00-00			

```
Good Bye!
12:27:57: C:\CDPStudioWorkspace\deployments\systems\CDP_4.12_Windows_x86_64-bit\RecordNPlay\RecordNPlayUI\RecordNPlayUI.exe exited with code 0
```

Pairing the myCobot robot arm with CDP Studio over the Wi-Fi network

Click the Username field for your myCobot (based on its IP address) and enter 'er', then click the Pair button next to it. You will be prompted to enter the password – the default is 'Elephant'.

Under Applications, change the Device for RecordNPlayIO application to your myCobot device name, then change the Toolkit to ARMv8 64-bit (Debian 11). When you run the RecordNPlay project, this will then be deployed over the network to the robot arm.

07 Run the project

Right-click RecordNPlay in the left panel and select Run & Connect. After a few moments, a new Arm Record'n'Play window should appear, showing the GUI for recording arm movements. First, enter a name for the sequence and click Add. Then click Record to start recording steps. You can move the sliders, but it's a lot easier to move the robot arm around and then click its LED panel button to add each position as a step. Recorded steps are shown in a list and can be updated or deleted individually using the buttons at the bottom right. The 'Step duration' bar sets the time for which an arm position is held.

Try recording some steps and then hit the top-left Play button to play the sequence. If you have a myCobot arm connected, it should follow the movements you recorded; if not, select ArmVisualizer in the project's left panel, then the DHChain Visualizer tab to view a 3D representation of the arm with its six joints. As you move between two steps, the visualisation shows both and the movement of the arm's head with a red line. The gripper status is indicated by a green (closed) or grey (open) dot.

Top Tip

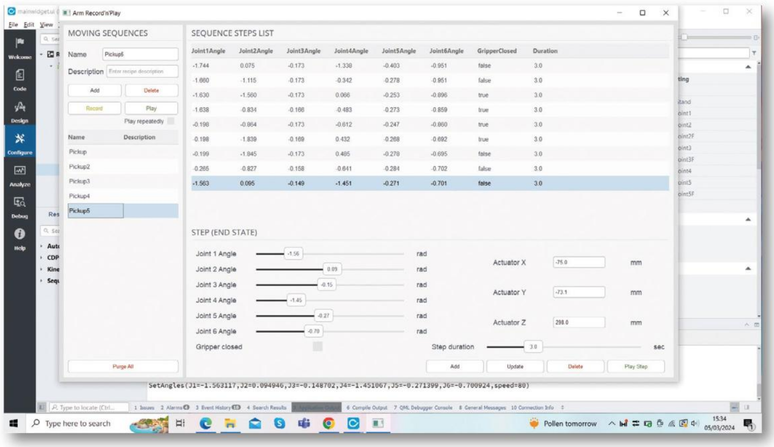
Another arm

While this project is designed for the myCobot 280 Pi, you could use a different robot arm – you'd just need to create a new IO library to communicate with it.

08 Pick and place

Now, let's get our arm to pick up an object and place it down in another location. We placed a pencil sharpener on top of a biscuit tin, high enough to give the arm plenty of space to pick it up without the bottom of the gripper hitting the table.

> "We placed a sharpener on top of a biscuit tin"



Arm Record'n'Play

MOVING SEQUENCES

Name: Pickup5

Name	Description
Pickup	
Pickup2	
Pickup3	
Pickup4	
Pickup5	

SEQUENCE STEPS LIST

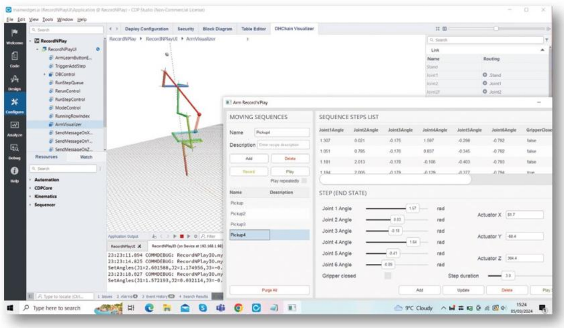
Joint1Angle	Joint2Angle	Joint3Angle	Joint4Angle	Joint5Angle	Joint6Angle	GripperClosed	Duration
-1.744	0.075	-0.173	-1.338	-0.403	-0.951	false	3.0
-1.660	-1.115	-0.173	-0.342	-0.278	-0.951	false	3.0
-1.630	-1.560	-0.173	0.066	-0.253	-0.896	true	3.0
-1.638	-0.834	-0.166	-0.483	-0.273	-0.859	true	3.0
-0.198	-0.864	-0.173	-0.612	-0.247	-0.860	true	3.0
-0.198	-1.839	-0.169	0.432	-0.268	-0.692	true	3.0
-0.199	-1.845	-0.173	0.485	-0.278	-0.695	false	3.0
-0.265	-0.827	-0.158	-0.641	-0.284	-0.702	false	3.0
-1.563	0.095	-0.149	-1.451	-0.271	-0.701	false	3.0

STEP (END STATE)

Joint 1 Angle	-1.56	rad
Joint 2 Angle	0.09	rad
Joint 3 Angle	-0.15	rad
Joint 4 Angle	-1.45	rad
Joint 5 Angle	-0.27	rad
Joint 6 Angle	-0.70	rad
Actuator X	-75.0	mm
Actuator Y	-73.1	mm
Actuator Z	298.0	mm
Step duration	3.0	sec

```
SetAngles(J1=-1.563117,J2=0.094946,J3=-0.148702,J4=-1.451067,J5=-0.271399,J6=-0.700924,speed=80)
```

Sequences of steps are recorded in the GUI, so you can play them back to make the robot arm follow them



```
23:23:11.894 COMMDEBUG: RecordNPlayIO.my
23:23:14.825 COMMDEBUG: RecordNPlayIO.my
SetAngles(J1=2.601588,J2=1.174956,J3=-0.
23:23:18.027 COMMDEBUG: RecordNPlayIO.my
SetAngles(J1=1.572193,J2=0.032114,J3=-0.
```

Joint1Angle	Joint2Angle	Joint3Angle	Joint4Angle	Joint5Angle	Joint6Angle	GripperClose
1.307	0.021	-0.175	1.597	-0.298	-0.792	false
1.051	0.795	-0.176	0.837	-0.345	-0.792	false
1.181	2.013	-0.178	-0.106	-0.403	-0.793	false
1.184	2.005	-0.179	-0.129	-0.377	-0.794	true

As you play a sequence, you can see the effects on a virtual arm in the ArmVisualizer pane

The myCobot's adaptive gripper attachment can be used to pick up and drop objects

Move the arm between positions and press the LED panel button to record each step. You can also open and then close the gripper manually to program it. Make sure the arm is stationary, in the right position, before closing it. Then lift the arm up and move it round and down to where you want to place the object. After opening the gripper to drop it, move the arm straight up so you don't bump into the item. You can adjust step positions in the GUI if needed. The steps are stored in an SQLite database too, so you could always edit that manually.

09 Kinematics

This project makes used of CDP Studio's Kinematics framework, in the form of the DHChain component. The basic concept of kinematics is that if you input joint angles for a robot arm, or chain of links, the framework can calculate the end position in 3D space – as shown in our project's ArmVisualizer pane, with the X/Y/Z coordinates shown in the Arm Record'n'Play GUI.

The method can also be used in the reverse direction, to convert a desired 3D end position into the required joint angles; this is known as 'inverse kinematics'.

Kinematics has many uses in the field of engineering, helping to calculate positions and velocities of moving parts such as those in an industrial robotic arm, or a bionic limb or exoskeleton. An example real-world case is the use of CDP Studio and kinematics is for controlling deck cranes on ships.

10 Exploring the project

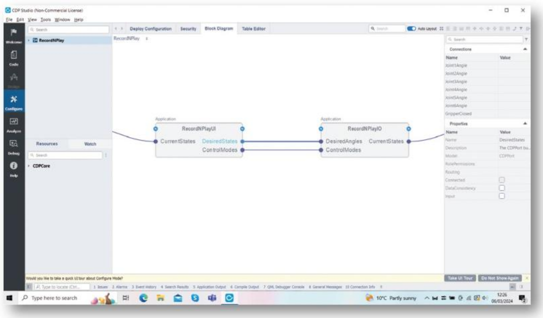
You can click the Block Diagram tab to see how the project's block-based components have been put together. At the highest level, there are two main blocks: for the RecordNPlayUI application for the GUI step recorder, and the RecordNPlayIO one for communicating with the myCobot arm. Here, the UI block's DesiredStates port links to the IO block's DesiredAngles port; it sends the angles set for the six joints, along with the gripper status, so that the arm will move accordingly. The ControlModes link is used to determine whether the arm should maintain a position or be allowed to move freely, for when you're recording moves. CurrentStates is a feedback port from IO that's used by UI to know what is the current position of the arm joints, gripper and LED button; this information is used by the recording process in UI.

Top Tip

Sturdy base

You'll need a base for your robot arm, to stop it falling over as it moves. We used the G-Base 2.0 to clamp ours to a table.

There are two main application blocks, the UI step recorder running on the PC, and the IO for communication with the arm



Part 02

CDP Studio: Control a robot arm with a Wii Remote

Use CDP Studio with a Nintendo Wii controller to manipulate a Raspberry Pi-based robot arm

MAKER
Phil King

Long-time contributor to *The MagPi*, Phil is a freelance writer and editor with a focus on technology.

@philkingeditor

Last issue, we showed you how to use the CDP Studio software development tool and its Kinematics framework to record movements for a Raspberry Pi-based robotic arm – the myCobot 280 Pi from Elephant Robotics – that could then be played back to perform a 'pick and place' routine using the Adaptive Gripper add-on. This time we'll using the same setup, but controlling the arm manually using a Nintendo Wii Remote and Nunchuk.

If you don't have the robot arm, you can still run the project by deploying it to a Raspberry Pi and viewing the movements in the on-screen arm visualiser.

01 Install the software

If you haven't already done it for the previous tutorial, you'll need to install CDP Studio. On your PC, visit **cdpstudio.com/getstarted** and download the free non-commercial version for Windows or Linux. During installation, select the 'ARMv8 64-bit (Debian 11)' component under CDP Studio 4.12, along with the one already ticked for your host PC. You will then be able to deploy projects to the myCobot 280 Pi arm, which uses a 64-bit version of Ubuntu.

If you already have CDP Studio installed, make sure it's updated to the latest version (Help > Check For Updates). Then go to Help > Package Manager and select 'Add or remove CDP versions' to add the ARMv8 64-bit (Debian 11) component if not already added.

02 Download the project

This is a complex project that would be time-consuming to build from scratch, so we'll download it from CDP Studio's GitHub repo. Go to **magpi.cc/mycobotwiiremote**, click the green Code button, and select 'Download ZIP'. Unzip the file on your PC. Move the resulting **myCobotWiiRemote-main** folder to the **CDPStudioWorkspace/systems** folder.

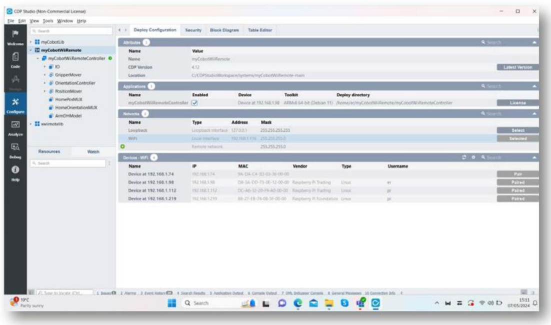
03 Download and build library

To deploy the project to the robot arm, you'll need a couple of libraries. If you don't already have the myCobotLib library from the previous tutorial, go to the GitHub repo at **magpi.cc/mycobotlib**, click the Code button and Download ZIP again. Extract it and then move the resulting **myCobotLib-main** folder to the **CDPStudioWorkspace/libraries** folder.

Open the myCobotLib.pro project file in CDP Studio, check ARM 64-bit is checked in the Deploy Configuration tab, then right-click its name in the left panel and select Build.

You'll also need the xwiiremotelib library at **magpi.cc/xwiimotelib** and the xwiimote

▼ You need to pair CDP Studio with the arm's Raspberry Pi over the Wi-Fi network to deploy the project



Attributes

Name	Value
Name	myCobotWiiRemote
CDP Version	4.12
Location	C:/CDPStudioWorkspace/systems/myCobotWiiRemote-main

Applications

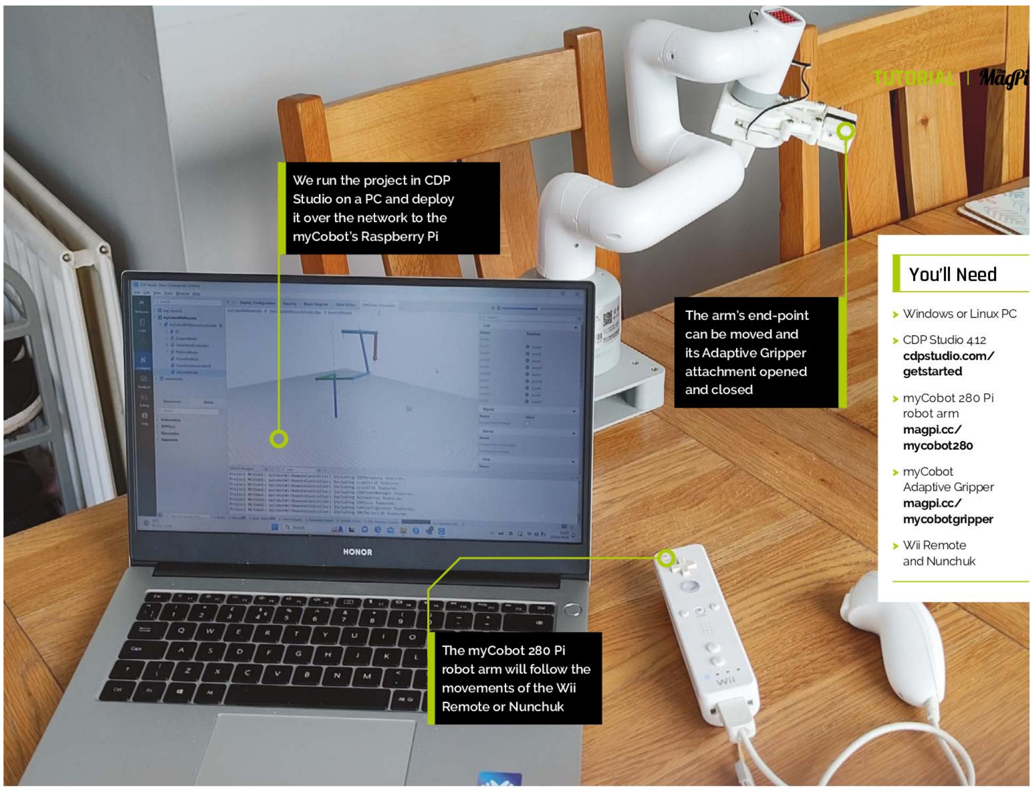
Name	Enabled	Device	Toolkit	Deploy directory
myCobotWiiRemoteController	✓	Device at 192.168.1.98	ARMv8 64-bit (Debian 11)	/home/er/myCobotWiiRemote/myCobotWiiRemoteController

Networks

Name	Type	Address	Mask
Loopback	Loopback interface	127.0.0.1	255.255.255.255
WiFi	Local interface	192.168.1.116	255.255.255.0
	Remote network		255.255.255.0

Devices - WiFi

Name	IP	MAC	Vendor	Type	Username	
Device at 192.168.1.74	192.168.1.74	9A-DA-C4-3D-03-36-00-00				Pair
Device at 192.168.1.98	192.168.1.98	D8-3A-DD-75-0E-12-00-00	Raspberry Pi Trading	Linux	er	Paired
Device at 192.168.1.112	192.168.1.112	DC-A6-32-20-F9-A0-00-00	Raspberry Pi Trading	Linux	pi	Paired
Device at 192.168.1.219	192.168.1.219	B8-27-EB-74-08-5F-00-00	Raspberry Pi Foundation	Linux	pi	Paired


We run the project in CDP Studio on a PC and deploy it over the network to the myCobot's Raspberry Pi
The arm's end-point can be moved and its Adaptive Gripper attachment opened and closed
The myCobot 280 Pi robot arm will follow the movements of the Wii Remote or Nunchuk

You'll Need

- Windows or Linux PC
- CDP Studio 4.12 **cdpstudio.com/getstarted**
- myCobot 280 Pi robot arm **magpi.cc/mycobot280**
- myCobot Adaptive Gripper **magpi.cc/mycobotgripper**
- Wii Remote and Nunchuk

submodule on which it's based at **magpi.cc/xwiimote**. You can either clone the library with the recursive option:

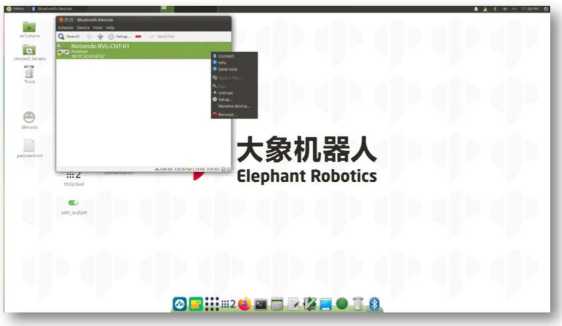
```
$ git clone --recursive https://github.com/
CDPTechnologies/xwiimotelib.git
```

Or you can download and extract the ZIP file, then download and extract the xwiimote ZIP and place the folder's contents into the library's xwiimote subfolder.

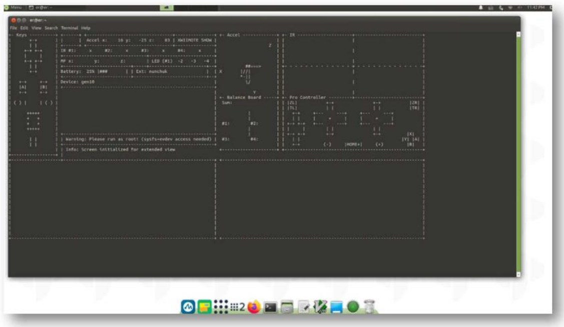
Move the **xwiimotelib-main** folder to **CDPStudioWorkspace/libraries**, then open the xwiiremotelib.pro project file in CDP Studio. Before building the library, open the Deploy Configuration tab for the wiimoteIO component and check that the ARMv8 64-bit (Debian 11) toolkit is enabled. If you get errors when building it, make sure CDP Studio has been updated (see Step 1).

04 Open the project

Now open the main **myCobotWiiRemote.pro** project file in CDP Studio. In the left panel of the Configure window, you'll note that it comprises a myCobotWiiRemoteController application with various components. A key part is the Kinematics-based ArmDHModel, which features a DHChain Visualizer pane that can be used to view the arm positions on-screen in 3D. This can be used even if you don't have a real arm connected, so you can still run the project and see how your Wii Remote and Nunchuk actions affect its movements.



Pair and then connect the Wii Remote controller with the arm using the Bluetooth Manager in Ubuntu



You can test the Wii Remote with the 'xwiishow 1' command in a terminal in the arm's Ubuntu OS

Top Tip

Another arm

While this project is designed for the myCobot 280 Pi, you could use a different robot arm – you'd just need to create a new IO library to communicate with it.

05 Prepare myCobot

If you've already set this up in the first tutorial, you can skip the first part of this step. The myCobot arm's Ubuntu OS has a non-standard version of the OpenSSH server. So you'll need to make a small change to a config file so CDP Studio can communicate with it over the network. SSH into the myCobot with the username 'er' at its IP address; the default password is Elephant. Then enter:

```
$ sudo nano /etc/ssh/sshd_config
```

Locate the line that sets the PubkeyAuthentication parameter and set it to yes (and make sure the line is not commented out). Press **CTRL+X**, then **Y** to exit and save.

Restart the OpenSSH server with:

```
$ sudo systemctl restart sshd
```

06 Pair the arm with CDP

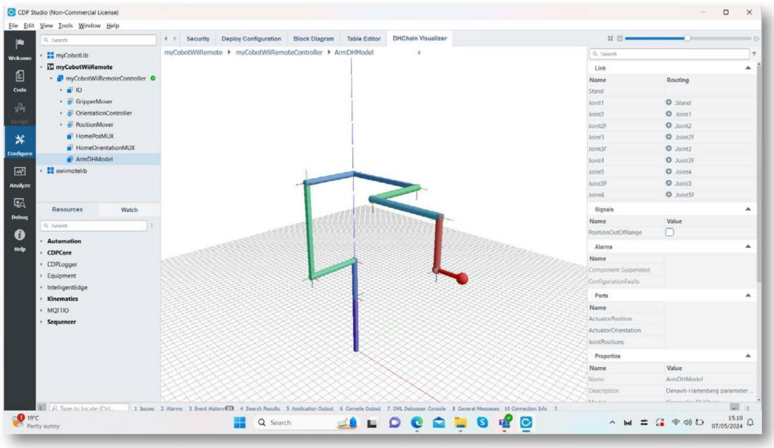
Open the Deploy Configuration tab for myCobotWiiRemoteController. Under Networks, press the Select button for 'WiFi'. The 'Devices – WiFi' table below should start showing any devices available to pair with CDP Studio. Click the Username field for your myCobot (based on its IP address) and enter 'er', then click the Pair button next to it. You will be prompted to enter the password – the default is Elephant.

Under Applications, for Device select your myCobot IP address or name, then change the Toolkit to ARMv8 64-bit (Debian 11). When you run the myCobotWiiRemote project, it will then be deployed over the wireless network to the robot arm.

07 Pair the Wii controllers

You'll need to pair the Wii Remote and Nunchuk controllers via Bluetooth with the robot arm's Raspberry Pi. First, open the battery compartment of the Wii Remote and press the small red Sync button to start pairing; the Remote's blue player LED should start blinking.

Now, from the robot arm's Ubuntu MATE desktop (viewed on a monitor or remotely via



You can view the arm's movements on-screen in the ArmDHModel DHChain Visualizer

VNC), open the Menu (top left) then search for and open Bluetooth Manager. Click Search and you should see the Wii Remote, possibly as 'Nintendo RVL-CNT-01'; select it and choose Pair. The Remote's LED should stop blinking.

08 Run the project

Right-click myCobotWiiRemote in the left panel of CDP Studio and select Run & Connect. If you have a myCobot arm connected, it should follow the movements you make with the Wii controllers; if not, select ArmDHModel in the project's left panel to view a 3D representation of the arm with its six joints. The gripper status is indicated by a green (closed) or grey (open) dot.

By default, the arm's servos are released. Press the Remote's 2 button to engage them. Now hold the A button and tilt the Remote to tilt the arm's endpoint accordingly. Note that if you move outside the arm's range, the Remote will rumble. Pressing the Home button returns the arm to a preset home position.

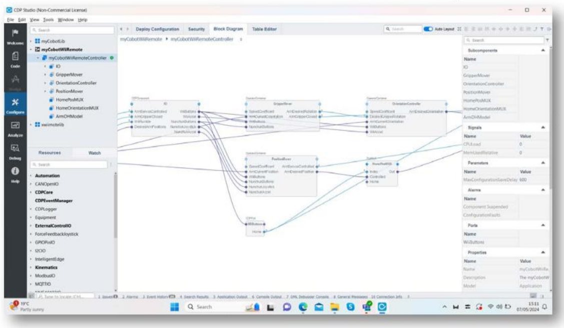
The Nunchuk's joystick can be used to move the arm's endpoint forward, back, left, and right. When holding the Z button, you can also tilt the Nunchuk to move the endpoint up and down. Pressing the C button toggles the gripper status between open and closed.

To disengage the servos again, press the Remote's 1 button, but be ready to catch the limp arm as it falls!

09 Kinematics

This project makes used of CDP Studio's Kinematics framework, in the form of the DHChain component. The basic concept of kinematics is that if you input joint angles for a robot arm, or chain of links, the framework can calculate the end position in 3D space. This is used to control the arm's joint movements to reach a certain x/y/z position, such as for the home position and the those reached via the Wii and Nunchuk movements.

Kinematics has many uses in the field of engineering, helping to calculate positions and velocities of moving parts such as those in an industrial robotic arm, or a bionic limb or exoskeleton. An example real-world case is the use of CDP Studio and kinematics is for controlling deck cranes on ships.


You can see how the project's components connect to each other in the Block Diagram view

10 Exploring the project

You can click the Block Diagram tab to see how the project's block-based components have been put together. These include an IO block to interpret the controller actions, along with blocks for the desired arm position, gripper orientation, and gripper open/closed status. A couple of mux blocks enable the arm's home position and orientation to be triggered. The ArmDHModel block calculates the arm joint angles required for a required position, and also triggers the Wii Remote rumble if a position is out of range.

To take the project even further, you could add some extra actions for the unused controller buttons – by opening the **xwiimoteIO.cpp** code file and adding events and return codes for them there, then creating actions for the latter in the main application.

Top Tip

No arm?

You can still run the project without a robot arm, by deploying it to a Raspberry Pi (or other Linux PC). Just make sure the ARM toolkit used matches the 32-bit or 64-bit architecture of its OS.

Using the Nunchuk, you can manoeuvre the arm's Adaptive Gripper and close it to pick up an object