Copyright 2026 Synthiam Inc.

Installing ARCx

Install and Run

ARCx is the robot programming platform by Synthiam. It runs on the computer that controls your robot and serves its user interface over HTTP, so you program and drive the robot from a web browser on the same machine or on any phone, tablet or PC that can reach it.
Pick the guide for the computer that will run ARCx:

Computer	Download	Guide
Windows 10 / 11, 64-bit (x64)	Installer (`.exe`)	Windows.md
Raspberry Pi and other 64-bit ARM single board computers, running Ubuntu 24.04 LTS	Zip (`ARCx-Linux-ARM64.zip`)	Raspberry-Pi.md
Linux PC, 64-bit (x64), running Ubuntu 24.04 LTS	Zip (`ARCx-Linux-64.zip`)	Linux-x64.md

On Linux, ARCx supports Ubuntu 24.04 LTS only, on both x64 and ARM64. macOS is not supported yet.

Things that are the same on every platform

ARCx is a server, the browser is the interface

ARCx has no desktop window of its own. It runs in a terminal (console) window and the interface opens in your web browser at:

```
http://127.0.0.1:5000
```

Closing the browser does **not** stop ARCx, and the robot keeps running. To stop ARCx, press **Ctrl+C** in its terminal window (or close that window). A clean shutdown takes around 20 seconds; that is normal.

Only this computer can connect by default

Out of the box ARCx only accepts connections from the computer it runs on. To reach it from another device, start it with the `-b` (bind) option:

```
ARCx -b http://0.0.0.0:5000
```

Then browse to `http://<ip-address-of-the-robot-computer>:5000` from the other device.

Command line options

Option	What it does
`-h`, `--help`	Show the help text and exit
`-f`, `--file <project.arcx>`	Open this project file on start-up
`-b`, `--bind <url>`	The address to listen on. Default `http://127.0.0.1:5000`
`-c`, `--cert <file>`	Server certificate for an `https://` bind (`.pfx`/`.p12`, or `.pem`/`.crt` with `--cert-key`)

`--cert-key <file>`	Private key file, when `--cert` is a `.pem`/`.crt`
`--cert-password <password>`	Password for the `.pfx`/`.p12` bundle or an encrypted key

Examples:

```
ARCx -f MyRobot.arcx
ARCx -b http://0.0.0.0:5000
ARCx -b https://0.0.0.0:5001 --cert server.pfx --cert-password secret
ARCx -b https://0.0.0.0:5001 --cert server.crt --cert-key server.key
```

HTTPS encrypts the connection. It does not add a login.

Where your files go

ARCx keeps your projects, settings, logs and installed robot skills in a folder named **ARCx Data**. It is separate from the program folder, so updating or reinstalling ARCx never touches your projects.

Platform	Your projects and settings	Robot skills and shared files
Windows	`Documents\ARCx Data`	`C:\ProgramData\ARCx Data`
Linux / Raspberry Pi	`~/ARCx Data`	`~/ARCx Data` (same folder)

Projects are saved in `ARCx Data/My Projects`.

Updates

ARCx checks synthiam.com for a newer version when it starts and shows an update badge in the interface when one is available. Each guide explains how to install the update.

Help

Documentation, robot skills, tutorials and the community forum are at synthiam.com.

Command Line

ARCx is started from a terminal (or a shortcut, a service or a startup script) and runs as a web server. The robot is controlled from a web browser pointed at the address ARCx listens on. This page explains every command line option, the environment variables ARCx reads, and how ARCx starts and stops.
Run `ARCx -h` to print a short version of this page.

Contents

1. Quick reference
2. Opening a project at start-up
3. Password protected projects
4. Choosing the address ARCx listens on
5. HTTPS and certificates
6. What happens at start-up
7. Stopping ARCx
8. Exit codes
9. Examples
10. Troubleshooting

Quick reference

```
ARCx [options]
```

Option	Value	What it does
`-h`, `--help`	none	Print the help and exit.
`-f`, `--file`	project file	Open this project when ARCx starts.
`-p`, `--project-password`	password	The password of a password protected project opened with `-f`.
`-b`, `--bind`	URL	The address the web server listens on. Default `http://127.0.0.1:5000`.
`-c`, `--cert`	certificate file	The server certificate for an `https://` bind. A `.pfx`/`.p12` bundle, or a `.pem`/`.crt` used with `--cert-key`.
`--cert-key`	key file	The private key file, when `--cert` is a `.pem`/`.crt`.
`--cert-password`	password	The password of the `.pfx`/`.p12` bundle, or of an encrypted `--cert-key`.
`--no-browser`	none	Do not open the local web browser when ARCx starts. Overrides the **Open a web browser when ARCx starts** preference for this run, without changing it.
`--mcp-stdio`	URL (optional)	Run as the Model Context Protocol stdio bridge instead of starting ARCx: forwards stdin/stdout to a running ARCx's `/mcp`, `http://127.0.0.1:5000/mcp` unless a URL follows. For AI clients that start servers as local programs. See `mcp.md`.
`--token`	token	The MCP access token the bridge sends, when Preferences requires one. Only with `--mcp-stdio`.

Environment variable	What it does
`ARCX_PROJECT_PASSWORD`	The password of a password protected project opened with `-f`. Used when `-p` is not given.
`ARCX_MCP_TOKEN`	The MCP access token for `--mcp-stdio`. Used when `--token` is not given; keeps the token out of the client's configuration file.

`ARCX_ROBOT_BUS`	`1` (or `true`, `on`, `yes`) lets programs outside ARCx use the Robot Bus at `/api/bus`, whatever **Options → Preferences → Robot Bus** says. For a robot with no screen to turn it on from.
`ARCX_ROBOT_BUS_TOKEN`	The access token external programs must send to the Robot Bus. Overrides the token in **Options → Preferences → Robot Bus**.

An option ARCx does not recognize prints the help, followed by an error naming the argument, and ARCx does not start.
On Windows the program is `ARCx.exe`. On Linux and macOS run it as `./ARCx` from its folder.

Opening a project at start-up

```
ARCx -f MyRobot.arcx
```

`-f` (or `--file`) opens a project as soon as ARCx starts, before the web server begins listening. This is how a robot is set up to run its project unattended: start ARCx from the operating system's start-up with `-f`, and the project, its robot skills and its startup script are running before anybody opens a browser.

- The value is a path to a local `.arcx` file. Put the path in quotes if it contains spaces.
- Cloud projects (`cloud://...`) cannot be opened from the command line yet. Open them from **File → Open** in the user interface instead.
- If the project cannot be opened, ARCx prints why and exits rather than starting with an empty project. That includes a project that loads with warnings, such as a robot skill that could not be installed. Open the project from the user interface once to see the warnings and fix them, then save it.
- Without `-f`, ARCx starts with a new project, or whatever the home page offers.

Password protected projects

A project can be saved with a password from **File → Save** by ticking **Password protect**. The whole project file is then encrypted, and nothing in it can be read without the password. To open one from the command line, give ARCx the password in one of two ways.
With the option:

```
ARCx -f MyRobot.arcx -p secret
```

With the environment variable (recommended):

```
# Linux and macOS
ARCX_PROJECT_PASSWORD=secret ./ARCx -f MyRobot.arcx
```

```
:: Windows cmd
set "ARCX_PROJECT_PASSWORD=secret" && ARCx -f MyRobot.arcx
```

```
# Windows PowerShell
$env:ARCX_PROJECT_PASSWORD='secret'; .\ARCx -f MyRobot.arcx
```

The environment variable is the safer choice. A command line argument can be seen by other users of the computer in the process list (Task Manager, `ps`), and it is kept in the shell's history. An environment variable is not.
How the password is used:

- When both are set, `-p` wins.
- ARCx reads `ARCX_PROJECT_PASSWORD` once at start-up and then clears it, so programs

and scripts that ARCx starts do not inherit the password.
- The password is only used when the `-f` project is password protected. For a project that is not, it is ignored.
- A password given without `-f` is ignored, and ARCx says so.
- The password is never written to the log.
- A password that opens the project is remembered for as long as ARCx runs. When the project is saved from the user interface, the password prompt is already filled in with it.
- A missing password and a wrong password both stop ARCx with a message saying which it was.

In the Windows cmd example, keep the quotes around `ARCX_PROJECT_PASSWORD=secret`. Without them cmd includes the space before `&&` in the password, and the password is wrong.
A forgotten project password cannot be recovered. Nothing can open the project without it.

Choosing the address ARCx listens on

```
ARCx -b http://0.0.0.0:5000
```

`-b` (or `--bind`) sets the address and port of the web server that serves the ARCx user interface.
- The default is `http://127.0.0.1:5000`. It only accepts connections from the same computer.
- `http://0.0.0.0:5000` accepts connections from any computer that can reach this one, such as a phone or tablet on the same network. Browse to `http://<this computer's address>:5000` from the other device.
- To listen on one network adapter only, give that adapter's address, for example `http://192.168.1.20:5000`.
- Only one address can be given. A list separated by `;` is not supported.
- If the port is already in use (for example by another copy of ARCx), ARCx fails to start. Pick another port, or stop the other program.

ARCx has no login. Anyone who can reach the address can control the robot and read or delete files through the file manager. Only bind to an address other than `127.0.0.1` on a network you trust. ARCx writes a warning to the log when it binds to anything else.

HTTPS and certificates

```
ARCx -b https://0.0.0.0:5001 --cert server.pfx --cert-password secret
```

An `https://` bind encrypts the connection between the browser and ARCx. It needs a server certificate, given in one of three ways.
A PKCS#12 bundle (`.pfx` or `.p12`), which holds the certificate and its private key together:

```
ARCx -b https://0.0.0.0:5001 --cert server.pfx --cert-password secret
```

Leave out `--cert-password` if the bundle has no password.
A PEM certificate and key (`.pem` or `.crt`, plus a key file), as issued by Let's Encrypt and most Linux tools:

```
ARCx -b https://0.0.0.0:5001 --cert server.crt --cert-key server.key
```

If the key file is encrypted, add `--cert-password` with its password.
appsettings.json, without any certificate option: set `Kestrel:Certificates:Default:Path` (and `Kestrel:Certificates:Default:Password`) in the `appsettings.json` beside the ARCx program.

Rules ARCx checks before starting:

- `--cert` with an `http://` bind is refused, because the certificate would never be used.
- An `https://` bind with no certificate from any of the three sources is refused, with a list of what to supply.
- The certificate must include its private key.
- An expired certificate still loads, but ARCx logs a warning, because browsers will refuse to connect.

HTTPS encrypts the connection. It does not add a login; see the warning in the previous section.

What happens at start-up

In order:

1. ARCx prints its banner and version, and reads the command line.
2. It creates its data folders under `ARCx Data` in the user's Documents folder if they are missing, and starts its log.
3. It loads the server certificate, for an `https://` bind.
4. It runs the start-up tasks, such as updating installed robot skills.
5. It opens the `-f` project, if one was given.
6. The web server starts listening, and ARCx prints `Listening on <address>`.
7. ARCx opens a web browser at that address. Turn this off in **Options → Preferences → Open a web browser when ARCx starts** on a robot with no screen, or one you always connect to from another device. To skip it for one run without changing the preference, start ARCx with `--no-browser`, which wins over the preference.

Stopping ARCx

ARCx has no exit command in its user interface. Stop it from the terminal it runs in:

- Press **Ctrl+C**, or close the terminal window. ARCx shuts down the robot skills, disconnects the robot controllers and closes its log. This normally takes a few seconds.
- If shutting down hangs, press **Ctrl+C** a second time to exit immediately.

On Linux and macOS, `kill <pid>` (SIGTERM) shuts ARCx down the same way as Ctrl+C.

Exit codes

Code	Meaning
`0`	ARCx stopped normally, or the help was printed. The help is also printed for an unrecognized argument, and that exits with `0` as well, so check the output rather than the code for a mistyped option.
`1`	ARCx could not start, or the `-f` project could not be opened. The reason is printed in the terminal and written to the log.

A start-up script or service manager can use the code to tell a failed start from a normal stop, for example to stop restarting ARCx when its project password is wrong.

Examples

Start with the default settings. Only this computer can connect:

```
ARCx
```

Open a project and let other devices on the network connect:

```
ARCx -f "My Robot.arcx" -b http://0.0.0.0:5000
```

Open a password protected project on a Linux robot, with the password kept out of the process list:

```
ARCX_PROJECT_PASSWORD=secret ./ARCx -f /home/robot/MyRobot.arcx -b http://0.0.0.0:5000
```

Open a project on a robot with no screen, without opening a web browser:

```
ARCx -f "My Robot.arcx" -b http://0.0.0.0:5000 --no-browser
```

Serve over HTTPS with a PKCS#12 bundle:

```
ARCx -b https://0.0.0.0:5001 --cert server.pfx --cert-password secret
```

Serve over HTTPS with a PEM certificate and key:

```
ARCx -b https://0.0.0.0:5001 --cert server.crt --cert-key server.key
```

Run a robot with no screen whose project is driven by programs on the network (a Python program, a ROS node through the ROS Connector), with the Robot Bus turned on and protected by a token:

```
ARCX_ROBOT_BUS=1 ARCX_ROBOT_BUS_TOKEN=secret ./ARCx -f /home/robot/MyRobot.arcx -b http://
0.0.0.0:5000 --no-browser
```

The programs then connect to `http://<robot>:5000/api/bus` with that token, and the Python and JavaScript clients can be downloaded from the robot at `/sdk/arcx_bus.py` and `/sdk/arcx-bus.js`.
Run a second copy beside one that is already running, on another port:

```
ARCx -b http://127.0.0.1:5099
```

Troubleshooting

"The project ... is password protected. Give its password with -p/--project-password or the ARCX_PROJECT_PASSWORD environment variable." The project was saved with a password and none was given. Add `-p` or set the variable.
"The password for the project ... is incorrect." Check the password. In Windows cmd, check the quotes around the `set` command, because without them a trailing space ends up in the password. If `-p` is given, it overrides the environment variable.
"There are warnings attempting to load ..." The project opened, but something in it did not, such as a robot skill that could not be downloaded. Open the project from the user interface to see each warning, fix it, and save the project.
"Opening a cloud project from the command line ... is not supported yet." Use a local copy of the project with `-f`, or open the cloud project from the user interface.
"A certificate was specified but the bind address ... is not https." Change the bind to `https://...`, or remove `--cert`.
"Cannot bind to https://... because no server certificate is configured." Give a certificate with `--cert` (see HTTPS and certificates).
"ARCx failed to start on ..." Usually the port is already in use. Choose another port with `-b`, or stop whatever is using it. A copy of ARCx that was just closed can take a few seconds to release its port.
The browser cannot connect from another device. The default bind only accepts connections from the same computer. Start ARCx with `-b http://0.0.0.0:5000` and check that the computer's firewall allows the port.
Unrecognized argument. ARCx printed the help and an error naming the argument it did not understand. Check the spelling. Options that take a value need the value after them,

separated by a space.

Linux x64

This guide is for 64-bit Intel and AMD computers running Ubuntu 24.04 LTS: desktop PCs, laptops, mini PCs and x64 single board computers such as LattePanda and UP boards. For a Raspberry Pi or another ARM board, use Raspberry-Pi.md instead.

Requirements

- A **64-bit x64** computer (Intel or AMD).
- **Ubuntu 24.04 LTS (64-bit)**, the Desktop or the Server (headless) edition.
- An internet connection.

Ubuntu 24.04 LTS is the only Linux distribution ARCx supports, on x64 and on ARM64 alike. Other distributions and other Ubuntu versions are not tested and are not supported.
You do not need to install .NET. It is included with ARCx.

Check the computer and the OS

Open a terminal (or connect with SSH) and run:

```
uname -m
grep -E '^(ID|VERSION_ID)=' /etc/os-release
```

It must print `x86_64`, `ID=ubuntu` and `VERSION_ID="24.04"`. If `uname -m` prints `aarch64`, the computer is ARM; use Raspberry-Pi.md.

Install

Run these commands in a terminal:

```
# Install the system libraries ARCx uses (camera and computer vision, audio, graphics)
sudo apt update
sudo apt install -y unzip libicu74 libasound2t64 libjack-jackd2-0 libfontconfig1 libgl1 \
  libgtk-3-0t64 ocl-icd-libopencl1

# And the ones the Chromium browser windows use (web pages on the robot's own screen)
sudo apt install -y libnss3 libatk-bridge2.0-0t64 libcups2t64 libgbm1 libxkbcommon0 \
  libxcomposite1 libxdamage1 libxrandr2 libpango-1.0-0 libatspi2.0-0t64 fonts-dejavu-core

# Start in your home folder
cd ~

# Download ARCx
wget -nd https://synthiam.com/api/download/arcx/ARCx-Linux-64.zip

# Unzip it into a folder named ARCx
unzip ARCx-Linux-64.zip -d ARCx

# Go into the folder
cd ARCx

# Allow the ARCx program to run
chmod +x ARCx

# Start ARCx
./ARCx
```

If `apt install` reports that a package cannot be found, the OS is not Ubuntu 24.04. See Check the computer and the OS.
If `ls` inside `~/ARCx` shows a single folder instead of the `ARCx` program, `cd` into that folder before running `chmod` and `./ARCx`.
The zip file is no longer needed once it is unzipped. Delete it with `rm ~/ARCx-Linux-64.zip`.

Open the interface

ARCx prints its start-up messages in the terminal. Leave the terminal open; it *is* ARCx.

On the same computer

Open any browser and go to:

```
http://127.0.0.1:5000
```

On Linux, ARCx does not open the browser for you by default. You can turn that on in the ARCx preferences.

From another computer, phone or tablet

Start ARCx so it accepts connections from the network:

```
cd ~/ARCx
./ARCx -b http://0.0.0.0:5000
```

Find this computer's IP address:

```
hostname -I
```

On the other device, browse to `http://<that-ip-address>:5000`, for example `http://192.168.1.30:5000`.
If the firewall is turned on (Ubuntu's `ufw`, for example), allow the port:

```
sudo ufw allow 5000/tcp
```

Warning: ARCx has no login. Anyone who can reach that address can control the robot and read or delete files on this computer. Only do this on a network you trust.

Stop

Press **Ctrl+C** in the terminal where ARCx is running. Allow around 20 seconds for a clean shutdown.
Closing the browser does not stop ARCx. If you started ARCx over SSH, closing the SSH session stops it. To keep it running after you disconnect, run it as a service (below).

Give ARCx access to hardware

ARCx talks to cameras, microphones, speakers, serial devices (USB serial adapters, servo controllers) and game controllers as the user who runs it. Linux controls access to each kind of device with a group: a program can open a device only if its user is in the group that owns the device. On Ubuntu a new user is **not** in the serial port group, so ARCx cannot open a serial port until you add it.
Add the user to all the groups ARCx uses with one command:

```
sudo usermod -aG dialout,video,audio,input,render $USER
```

To add only the serial port group, the command is:

```
sudo usermod -a -G dialout $USER
```

Group	What it gives access to	Devices
`dialout`	Serial ports: USB serial adapters (FTDI, CH340, CP210x), Arduino and other USB microcontroller boards, servo controllers, motor controllers and lidars on a serial port, and built-in serial ports	`/dev/ttyUSB*`, `/dev/ttyACM*`, `/dev/ttyS*`
`video`	USB and built-in cameras	`/dev/video*`

`audio`	Sound cards, speakers and microphones	`/dev/snd/*`
`input`	Gamepads, joysticks and other input devices	`/dev/input/*`
`render`	The graphics processor (GPU), for GPU acceleration in browser windows and in robot skills that run AI models	`/dev/dri/renderD*`

While you are logged in to the Ubuntu desktop, the desktop session lets you use cameras, sound, controllers and the GPU without these groups. Serial ports are never included, and none of it applies when ARCx runs over SSH or as a service. Add all the groups so ARCx works the same way however it is started.
Things to know:

- **Always include `-a`.** `-a` adds the groups to the ones the user is already in. Without it, `usermod -G` *replaces* all the user's groups, including `sudo`, and the user can no longer run administrator commands. `-aG` and `-a -G` do the same thing.
- **`$USER` is the user typing the command.** If ARCx runs as a service under a different user, put that user name in its place, for example `sudo usermod -aG dialout,video,audio,input,render robot`.
- **The change takes effect at the next login.** Log out and back in, or reboot. A terminal or SSH session that is already open keeps the old groups, so close it and connect again. If ARCx runs as a service, restart it so it picks up the new groups: `sudo systemctl restart arcx`.
- **To check**, run `groups`. It lists the groups of the current session, so it shows the new ones only after you log in again. `groups $USER` lists the groups that are saved for the user, even before you log in again.

To see which group a device needs, list it with `ls -l`. The fourth column is the group:

```
ls -l /dev/ttyUSB0
# crw-rw---- 1 root dialout 188, 0 Oct  4 10:12 /dev/ttyUSB0
```

To find which port a USB serial device is on, run `ls -l /dev/serial/by-id/`. It lists each connected USB serial device by its name and shows the port (`ttyUSB0`, `ttyACM0`, ...) it uses.

The X display (browser windows)

There is no group for the X display. Access to it belongs to the user who started it, not to a group, so adding the user to `video` or any other group does not change it.

- The kiosk display service in Browser windows accepts every user on the same computer, so nothing needs to be added.
- The Ubuntu desktop's display belongs to the user who is logged in to it. Run ARCx as that same user and it can draw on it. If the ARCx service runs as a different user, allow that user from a terminal on the desktop: `xhost +SI:localuser:<user name>`. The permission ends when the desktop user logs out.

Gamepads and HID devices

Game controllers are read from `/dev/input`, which belongs to the `input` group added above. While you are logged in to the desktop, Ubuntu usually lets your session use a plugged-in controller anyway, but ARCx started over SSH or as a service needs the group. **Views, Gamepads And HID** in ARCx shows every controller it can see; if it shows none, the Diagnostics page says whether permission is the reason.
A robot skill that talks to a HID device directly (a custom USB board, a sensor, a button box) opens its `/dev/hidraw*` node, which only root can open by default. Give the `input` group access to that one device with a udev rule, using the vendor and product ID shown on the

Gamepads And HID page (or by `lsusb`), in lowercase:

```
echo 'KERNEL=="hidraw*", ATTRS{idVendor}=="1234", ATTRS{idProduct}=="abcd", GROUP="input",
MODE="0660"' | sudo tee /etc/udev/rules.d/99-arcx-hid.rules
sudo udevadm control --reload-rules && sudo udevadm trigger
```

Then unplug the device and plug it back in.

USB serial adapter disappears as soon as it is plugged in

On Ubuntu, the `brltty` package (support for braille displays) grabs some common USB serial chips, including the CH340/CH341, and the port vanishes. If you do not use a braille display, remove it:

```
sudo apt remove brltty
```

Then unplug the adapter and plug it back in. It appears as `/dev/ttyUSB0` (or `/dev/ttyACM0`).

Browser windows

Robot skills and scripts can show web pages - a robot face, a dashboard, a video call, a kiosk screen - in Chromium windows on the monitors plugged into this computer. **Views, Browser Windows** in ARCx lists them and has a form to try one. Chromium ships inside ARCx; it only needs the system libraries from the Install step and an X display to draw on.

With a desktop

Nothing to set up. ARCx started from the desktop uses it, and an ARCx started as a service finds the desktop's display by itself (it looks for the X server on `:0`, which Ubuntu's desktop runs through XWayland).

Without a desktop (kiosk)

A desktop is not needed. A bare X server with no desktop, no window manager and no login screen is all a kiosk needs, and it uses far less memory. ARCx places its windows and makes them fullscreen itself.

1. Install the X server:

```
sudo apt install -y xserver-xorg xinit x11-xserver-utils
```

2. Start it at boot, on the screen, with the screen saver and power saving off. (To start it only when you need it instead, see Run the X server only when you need it.)

```
sudo tee /etc/systemd/system/arcx-display.service > /dev/null <<EOF
[Unit]
Description=X display for ARCx browser windows
After=systemd-user-sessions.service

[Service]
Type=simple
ExecStart=/usr/bin/Xorg :0 vt7 -nolisten tcp -s 0 -dpms
Restart=always
RestartSec=2

[Install]
WantedBy=multi-user.target
EOF

sudo systemctl daemon-reload
sudo systemctl enable --now arcx-display
```

The screen turns black: that is the empty X server, waiting for a window. On a touch screen add `-nocursor` to the `ExecStart` line to hide the mouse pointer everywhere.

3. If ARCx runs as a service (below), add these two lines under `[Unit]` in `/etc/systemd/system/arcx.service`, so ARCx starts after the display, then `sudo systemctl daemon-reload`:

```
Wants=arcx-display.service
After=arcx-display.service
```

ARCx finds the display on `:0` by itself. To use another one, set it in Views, Browser Windows. To turn the screen to portrait, run `DISPLAY=:0 xrandr --output HDMI-1 --rotate left` (the output name is listed by `DISPLAY=:0 xrandr`), or add it as an `ExecStartPost=` line to the display service.

Run the X server only when you need it

The service above runs the X server from every boot. To try browser windows first, or to use the screen only now and then, install the X server (step 1) and start it from the command line instead - over SSH works too:

```
# Start the X server in the background (asks for your password first)
sudo -b Xorg :0 vt7 -nolisten tcp -s 0 -dpms > /dev/null 2>&1

# Then start ARCx as usual; it finds the display on :0 by itself
cd ~/ARCx
./ARCx
```

`sudo` is needed because Xorg only lets a user sitting at the computer's own keyboard start it, not a user connected over SSH. `-b` sends it to the background once the password is typed, so the same terminal can start ARCx. Start the X server before ARCx, so ARCx finds the display. `-nocursor` works here too.
To stop the X server when you are done:

```
sudo pkill -x Xorg
```

The screen goes back to the text console (press **Ctrl+Alt+F1** if it stays black). Nothing is set to run at boot, so the X server does not come back until you start it again.

If a window does not appear

Open **Diagnostics** in ARCx. The Browser Windows section says whether a display was found, and lists any system library Chromium could not load (install it with `apt`). If windows open but stay black or blank, turn off GPU acceleration in Views, Browser Windows and press Restart.

Computers with little memory

On a computer with 2 GB of RAM or less, Chromium and a video call can run it out of memory, and with swap on slow storage the whole computer stops responding. Turn on **One renderer process** and **Low end device mode** in Views, Browser Windows and press Restart. Turning off GPU acceleration saves a little more.

Start ARCx automatically when the computer boots

Run ARCx as a systemd service so it starts at boot, keeps running when you disconnect, and restarts if it stops unexpectedly.

1. Create the service. Paste this whole block into the terminal as the user who will run ARCx (it fills in your user name and home folder):

```
sudo tee /etc/systemd/system/arcx.service > /dev/null <<EOF
[Unit]
```

```
Description=ARCx robot software by Synthiam
Wants=network-online.target
After=network-online.target

[Service]
Type=simple
User=$USER
WorkingDirectory=$HOME/ARCx
ExecStart=$HOME/ARCx/ARCx -b http://0.0.0.0:5000
Restart=on-failure
RestartSec=5
TimeoutStopSec=60

[Install]
WantedBy=multi-user.target
EOF
```

To open a project on start-up, add `-f` and the project path to the `ExecStart` line, in double quotes because the folder name has a space:
`ExecStart=/home/<you>/ARCx/ARCx -b http://0.0.0.0:5000 -f "/home/<you>/ARCx Data/My Projects/MyRobot.arcx"`

2.
Turn it on and start it:

```
sudo systemctl daemon-reload
sudo systemctl enable --now arcx
```

Useful service commands:

Command	What it does
`sudo systemctl status arcx`	Is ARCx running?
`journalctl -u arcx -f`	Watch the ARCx console output live (Ctrl+C to stop watching)
`sudo systemctl stop arcx`	Stop ARCx
`sudo systemctl restart arcx`	Restart ARCx
`sudo systemctl disable arcx`	Stop starting ARCx at boot
`top -H -p $(pgrep -of ARCx)`	Show only ARCx, one line per thread (press q to quit)

On Ubuntu Desktop, if sound works when you run `./ARCx` from a terminal but not from the service, the desktop's audio server is only available inside a logged-in session. Either turn on automatic login (**Settings > System > Users > Automatic Login**), or start ARCx from the desktop session instead of from a service. Ubuntu Server has no desktop audio server and plays sound directly, so this does not apply.

Update

When the interface shows that an update is available:

```
# Stop ARCx (press Ctrl+C in its terminal), or if it runs as a service:
sudo systemctl stop arcx

# Remove the old program folder. Your projects are in ~/ARCx Data and are not touched.
```

```
cd ~
rm -rf ~/ARCx

# Download and unzip the new version
wget -nd https://synthiam.com/api/download/arcx/ARCx-Linux-64.zip
unzip ARCx-Linux-64.zip -d ARCx
rm ARCx-Linux-64.zip
chmod +x ~/ARCx/ARCx

# Start it again, or if it runs as a service:
sudo systemctl start arcx
```

Removing the old folder first, rather than unzipping over it, makes sure no files from the old version are left behind.

Uninstall

```
# If you set up the service
sudo systemctl disable --now arcx
sudo rm /etc/systemd/system/arcx.service
sudo systemctl daemon-reload

# Remove the program
rm -rf ~/ARCx

# Optional: remove your projects, settings and installed robot skills
rm -rf ~/"ARCx Data"
```

Troubleshooting

Problem	**Fix**
`Permission denied` when running `./ARCx`	Run `chmod +x ARCx` in the ARCx folder.
`cannot execute binary file: Exec format error`	This is the wrong build for the computer. Check with `uname -m`; an `aarch64` computer needs the ARM64 build in Raspberry-Pi.md.
`No such file or directory` when running `./ARCx`	You are not in the folder that contains the `ARCx` program. Run `ls` and `cd` into the right folder.
ARCx exits at start-up with an error that mentions ICU, fontconfig or a missing `.so` library	Re-run the `apt install` line from the Install step. If apt cannot find the packages, the OS is not Ubuntu 24.04.
The camera shows no picture and the log says `Unable to load shared library 'OpenCvSharpExtern'`	Re-run the `apt install` line from the Install step, then run `ldd ~/ARCx/libOpenCvSharpExtern.so | grep "not found"` to list anything still missing.
No sound or microphone, and the log says `PortAudio could not be initialized` and `libjack.so.0: cannot open shared object file`	The audio library needs the JACK client library. Run `sudo apt install libjack-jackd2-0` and restart ARCx. This does not start a JACK server; ARCx still plays sound through ALSA. Run `ldd ~/ARCx/libportaudio.so | grep "not found"` to list anything still missing.
ARCx exits straight away with an address or port error	Another program is already using port 5000, or another copy of ARCx is running (check

	`sudo systemctl status arcx`). Stop the other copy, or use another port: `./ARCx -b http://0.0.0.0:5001`
Another device cannot connect	Check that ARCx was started with `-b http://0.0.0.0:5000`, that the firewall allows port 5000, that you used the IP from `hostname -I`, and that both devices are on the same network.
A serial port, camera or microphone is missing or gives "access denied"	Add your user to the hardware groups (see Give ARCx access to hardware) and log out and back in. For a USB serial adapter that vanishes, see the `brltty` note.
No cameras are listed, or `v4l2-ctl --list-devices` says `Failed to open /dev/video0: Permission denied`	The user that runs ARCx cannot open the camera. See A USB camera is not detected.
No sound or microphone, and `aplay -l` says `no soundcards found`	The user that runs ARCx cannot open the sound card. See No sound or microphone.
Start-up log	`~/ARCx Data/Logs`

A USB camera is not detected

ARCx lists a USB camera only if the user that runs ARCx can open the camera's `/dev/video*` device. To see what the system detects, install the camera tools and list the cameras:

```
sudo apt install -y v4l-utils
v4l2-ctl --list-devices
```

Each camera should be listed with its `/dev/video*` devices. Most USB cameras have two; the first one carries the video.
If it prints `Failed to open /dev/video0: Permission denied`, add your user to the `video` group:

```
sudo usermod -aG video $USER
```

Log out and back in (or reboot). The change does not apply to a terminal or SSH session that is already open. If ARCx runs as a service, restart it with `sudo systemctl restart arcx`. Run `groups` to check that `video` is listed, then run `v4l2-ctl --list-devices` again. If it still says `Permission denied`, run `ls -l /dev/video*`. The devices should belong to the `video` group (`crw-rw----+ 1 root video ...`). If they belong to a different group, add a rule that gives them to `video`, then unplug the cameras and plug them back in:

```
echo 'SUBSYSTEM=="video4linux", GROUP="video", MODE="0660"' | sudo tee /etc/udev/rules.d/99-video.rules
sudo udevadm control --reload-rules && sudo udevadm trigger
```

If there are no `/dev/video*` devices at all, the camera was not recognised. Run `lsusb` to check that it is listed, and try another USB port or cable.

No sound or microphone

When ARCx is started over SSH or as a service, it can use the sound card only if its user is in the `audio` group. (A user logged in to the Ubuntu desktop gets access through the desktop session, so sound can work from a desktop terminal but not over SSH.) To check, install the sound tools and list the sound cards as your user and as root:

```
sudo apt install -y alsa-utils
aplay -l
sudo aplay -l
```

If `aplay -l` prints `no soundcards found` but `sudo aplay -l` lists cards, add your user to the `audio` group:

```
sudo usermod -aG audio $USER
```

Log out and back in (or reboot). If ARCx runs as a service, restart it with `sudo systemctl restart arcx`. Run `groups` to check that `audio` is listed.
If `sudo aplay -l` does not list any cards either, the sound hardware was not recognised. For a USB speaker or microphone, run `lsusb` to check that it is listed.
To test the speaker and the microphone:

```
# Play a test sound ("Front Left, Front Right")
speaker-test -c 2 -t wav -l 1

# Record 5 seconds from the microphone, then play it back
arecord -d 5 -f cd /tmp/test.wav
aplay /tmp/test.wav
```

If the cards are listed but nothing plays, run `alsamixer`. Press **F6** to choose the card, press **M** on any channel marked `MM` to unmute it, and raise the volume with the arrow keys. With more than one card, add `-D plughw:<card>,<device>` to `speaker-test`, `arecord` or `aplay` to test a specific one, using the numbers from `aplay -l` or `arecord -l`.

Raspberry Pi ARM64

These steps also work on other 64-bit ARM (ARM64) single board computers, such as Orange Pi and Rock Pi boards, as long as they run **Ubuntu 24.04 LTS (64-bit)**.

Requirements

- A Raspberry Pi that can run a 64-bit OS: Pi 3, Pi 4, Pi 5 or Pi Zero 2 W. A Pi 4 or Pi 5 is recommended.
- **Ubuntu 24.04 LTS (64-bit)**. Ubuntu Server suits a robot with no screen. Ubuntu Desktop needs a Pi 4 or Pi 5 with at least 4GB of memory.
- An internet connection.

Raspberry Pi OS is not supported. ARCx's computer vision library is built for Ubuntu 24.04 and needs system libraries that Raspberry Pi OS does not provide. On Raspberry Pi OS, ARCx starts but the camera shows no picture. Other Ubuntu versions, including 22.04 and 26.04, do not work either.
You do not need to install .NET. It is included with ARCx.

Install Ubuntu on the Pi

1. Open **Raspberry Pi Imager** on another computer.
2. Under **Operating System**, choose **Other general-purpose OS > Ubuntu**, then **Ubuntu Server 24.04.x LTS (64-bit)** or **Ubuntu Desktop 24.04.x LTS (64-bit)**.
3. In the customisation settings, set the user name and password, the Wi-Fi network, and turn on SSH if the Pi will have no screen.
4. Write the card, put it in the Pi and start it. The first boot takes a few minutes.

Check the OS

Open a terminal (or connect with SSH) and run:

```
grep -E '^(ID|VERSION_ID)=' /etc/os-release
dpkg --print-architecture
```

It must print `ID=ubuntu`, `VERSION_ID="24.04"` and `arm64`. If it shows anything else, reinstall the card with Ubuntu 24.04 LTS (64-bit) as described above.

Install

Run these commands in a terminal on the Pi:

```
# Install the system libraries ARCx uses (camera and computer vision, audio, graphics)
sudo apt update
sudo apt install -y unzip libicu74 libasound2t64 libfontconfig1 libgl1 \
  libjpeg-turbo8 libavcodec60 libavformat60 libavutil58 libswscale7 \
  libtiff6 libwebp7 libwebpdemux2 libwebpmux3 libtesseract5 libgtk-3-0t64 ocl-icd-libopencl1

# And the ones the Chromium browser windows use (web pages on the robot's own screen)
sudo apt install -y libnss3 libatk-bridge2.0-0t64 libcups2t64 libgbm1 libxkbcommon0 \
  libxcomposite1 libxdamage1 libxrandr2 libpango-1.0-0 libatspi2.0-0t64 fonts-dejavu-core

# Start in your home folder
cd ~

# Download ARCx
wget -nd https://synthiam.com/api/download/arcx/ARCx-Linux-ARM64.zip

# Unzip it into a folder named ARCx
unzip ARCx-Linux-ARM64.zip -d ARCx

# Go into the folder
cd ARCx

# Allow the ARCx program to run
chmod +x ARCx

# Start ARCx
./ARCx
```

If `apt install` reports that a package cannot be found, the OS is not Ubuntu 24.04. See Check the OS.
If `ls` inside `~/ARCx` shows a single folder instead of the `ARCx` program, `cd` into that folder before running `chmod` and `./ARCx`.
The zip file is no longer needed once it is unzipped. Delete it with `rm ~/ARCx-Linux-ARM64.zip`.

Open the interface

ARCx prints its start-up messages in the terminal. Leave the terminal open; it *is* ARCx.

On the Pi's own desktop

Open Firefox (or any browser) on the Pi and go to:

```
http://127.0.0.1:5000
```

On Linux, ARCx does not open the browser for you by default. You can turn that on in the ARCx preferences.

From another computer, phone or tablet (headless Pi)

A Pi inside a robot usually has no screen. Start ARCx so it accepts connections from the network:

```
cd ~/ARCx
./ARCx -b http://0.0.0.0:5000
```

Find the Pi's IP address:

```
hostname -I
```

On the other device, browse to `http://<that-ip-address>:5000`, for example `http://192.168.1.40:5000`.
Warning: ARCx has no login. Anyone who can reach that address can control the robot and read or delete files on the Pi. Only do this on a network you trust.

Stop

Press **Ctrl+C** in the terminal where ARCx is running. Allow around 20 seconds for a clean shutdown.
Closing the browser does not stop ARCx. If you started ARCx over SSH, closing the SSH session stops it. To keep it running after you disconnect, run it as a service (below).

Give ARCx access to hardware

ARCx talks to cameras, microphones, speakers, serial devices (USB serial adapters, servo controllers, the Pi's UART) and game controllers as the user who runs it. Linux controls access to each kind of device with a group: a program can open a device only if its user is in the group that owns the device.
Add the user to all the groups ARCx uses with one command:

```
sudo usermod -aG dialout,video,audio,input,render $USER
```

To add only the serial port group, the command is:

```
sudo usermod -a -G dialout $USER
```

Group	What it gives access to	Devices
`dialout`	Serial ports: USB serial adapters (FTDI, CH340, CP210x), Arduino and other USB microcontroller boards, servo controllers, motor controllers and lidars on a serial port, and the Pi's GPIO serial port (UART)	`/dev/ttyUSB*`, `/dev/ttyACM*`, `/dev/ttyAMA0`, `/dev/serial0`
`video`	USB cameras	`/dev/video*`
`audio`	Sound cards, speakers and microphones	`/dev/snd/*`
`input`	Gamepads, joysticks and other input devices	`/dev/input/*`
`render`	The graphics processor (GPU), for GPU acceleration in browser windows and in robot skills that run AI models	`/dev/dri/renderD*`

The user created by Raspberry Pi Imager during the Ubuntu install is usually already in `dialout`, `video` and `audio`, but not in `input` or `render`. A user you add later is in none of them. Run the command anyway; groups the user is already in are not affected.
Things to know:

- **Always include `-a`.** `-a` adds the groups to the ones the user is already in. Without it, `usermod -G` *replaces* all the user's groups, including `sudo`, and the user can no longer run administrator commands. `-aG` and `-a -G` do the same thing.
- **`$USER` is the user typing the command.** If ARCx runs as a service under a different user, put that user name in its place, for example `sudo usermod -aG dialout,video,audio,input,render robot`.
- **The change takes effect at the next login.** Log out and back in, or reboot. A terminal or SSH session that is already open keeps the old groups, so close it and connect again. If ARCx runs as a service, restart it so it picks up the new groups: `sudo systemctl restart`

`arcx`.

- **To check**, run `groups`. It lists the groups of the current session, so it shows the new ones only after you log in again. `groups $USER` lists the groups that are saved for the user, even before you log in again.

To see which group a device needs, list it with `ls -l`. The fourth column is the group:

```
ls -l /dev/ttyUSB0
# crw-rw---- 1 root dialout 188, 0 Oct  4 10:12 /dev/ttyUSB0
```

To find which port a USB serial device is on, run `ls -l /dev/serial/by-id/`. It lists each connected USB serial device by its name and shows the port (`ttyUSB0`, `ttyACM0`, ...) it uses.

The X display (browser windows)

There is no group for the X display. Access to it belongs to the user who started it, not to a group, so adding the user to `video` or any other group does not change it.

- The kiosk display service in Browser windows accepts every user on the Pi, so nothing needs to be added.
- The Ubuntu desktop's display belongs to the user who is logged in to it. Run ARCx as that same user and it can draw on it. If the ARCx service runs as a different user, allow that user from a terminal on the desktop: `xhost +SI:localuser:<user name>`. The permission ends when the desktop user logs out.

Gamepads and HID devices

Game controllers are read from `/dev/input`, which belongs to the `input` group added above. A Pi running ARCx headless or as a service needs the group to see any controller. **Views, Gamepads And HID** in ARCx shows every controller it can see; if it shows none, the Diagnostics page says whether permission is the reason. Bluetooth controllers are paired with `bluetoothctl` (`scan on`, `pair`, `trust`, `connect`) while the controller's pairing lights are flashing.

A robot skill that talks to a HID device directly (a custom USB board, a sensor, a button box) opens its `/dev/hidraw*` node, which only root can open by default. Give the `input` group access to that one device with a udev rule, using the vendor and product ID shown on the Gamepads And HID page (or by `lsusb`), in lowercase:

```
echo 'KERNEL=="hidraw*", ATTRS{idVendor}=="1234", ATTRS{idProduct}=="abcd", GROUP="input",
MODE="0660"' | sudo tee /etc/udev/rules.d/99-arcx-hid.rules
sudo udevadm control --reload-rules && sudo udevadm trigger
```

Then unplug the device and plug it back in.

Using the Pi's GPIO serial port (UART)

Ubuntu uses the Pi's TX/RX pins for a login console by default. To connect a controller to them instead of USB, turn the console off:

1. Open the kernel command line: `sudo nano /boot/firmware/cmdline.txt`
2. Remove `console=serial0,115200` from the line (leave everything else on it, all on one line). Save with **Ctrl+O**, **Enter**, then exit with **Ctrl+X**.
3. Stop the login service on the port: `sudo systemctl disable --now serial-getty@ttyAMA0.service serial-getty@ttyS0.service`
4. Reboot.

The port then appears in ARCx as `/dev/serial0`.

Browser windows

Robot skills and scripts can show web pages - a robot face, a dashboard, a video call, a kiosk screen - in Chromium windows on a screen plugged into the Pi (HDMI, or the DSI touch display). **Views, Browser Windows** in ARCx lists them and has a form to try one. Chromium ships inside ARCx; it only needs the system libraries from the Install step and an X display to draw on. A Pi 4 or Pi 5 with 4GB or more is recommended: each window takes 100 to 200MB.

With Ubuntu Desktop

Nothing to set up. ARCx started from the desktop uses it, and an ARCx started as a service finds the desktop's display by itself (it looks for the X server on `:0`, which Ubuntu's desktop runs through XWayland).

Ubuntu Server, without a desktop (kiosk)

A desktop is not needed. A bare X server with no desktop, no window manager and no login screen is all a kiosk needs, and it leaves far more memory for the robot. ARCx places its windows and makes them fullscreen itself.

1. Install the X server:

```
sudo apt install -y xserver-xorg xinit x11-xserver-utils
```

2. Start it at boot, on the screen, with the screen saver and power saving off. (To start it only when you need it instead, see Run the X server only when you need it.)

```
sudo tee /etc/systemd/system/arcx-display.service > /dev/null <<EOF
[Unit]
Description=X display for ARCx browser windows
After=systemd-user-sessions.service

[Service]
Type=simple
ExecStart=/usr/bin/Xorg :0 vt7 -nolisten tcp -s 0 -dpms
Restart=always
RestartSec=2

[Install]
WantedBy=multi-user.target
EOF

sudo systemctl daemon-reload
sudo systemctl enable --now arcx-display
```

   The screen turns black: that is the empty X server, waiting for a window. On a touch screen add `-nocursor` to the `ExecStart` line to hide the mouse pointer everywhere.

3. If ARCx runs as a service (below), add these two lines under `[Unit]` in `/etc/systemd/system/arcx.service`, so ARCx starts after the display, then `sudo systemctl daemon-reload`:

```
Wants=arcx-display.service
After=arcx-display.service
```

ARCx finds the display on `:0` by itself. To use another one, set it in Views, Browser Windows. To turn the screen to portrait, run `DISPLAY=:0 xrandr --output HDMI-1 --rotate left` (the output name is listed by `DISPLAY=:0 xrandr`), or add it as an `ExecStartPost=` line to the display service.

Run the X server only when you need it

The service above runs the X server from every boot. To try browser windows first, or to use the screen only now and then, install the X server (step 1) and start it from the command line instead - over SSH works too:

```
# Start the X server in the background (asks for your password first)
sudo -b Xorg :0 vt7 -nolisten tcp -s 0 -dpms > /dev/null 2>&1

# Then start ARCx as usual; it finds the display on :0 by itself
cd ~/ARCx
./ARCx
```

`sudo` is needed because Xorg only lets a user sitting at the Pi's own keyboard start it, not a user connected over SSH. `-b` sends it to the background once the password is typed, so the same terminal can start ARCx. Start the X server before ARCx, so ARCx finds the display. `-nocursor` works here too.
To stop the X server when you are done:

```
sudo pkill -x Xorg
```

The screen goes back to the text console (press **Ctrl+Alt+F1** if it stays black). Nothing is set to run at boot, so the X server does not come back until you start it again.

If a window does not appear

Open **Diagnostics** in ARCx. The Browser Windows section says whether a display was found, and lists any system library Chromium could not load (install it with `apt`). If windows open but stay black or blank - which some Pi graphics driver setups cause - turn off GPU acceleration in Views, Browser Windows and press Restart.

Pis with little memory

On a Pi with 2 GB of RAM or less, Chromium and a video call can run it out of memory, and with swap on an SD card the whole Pi stops responding. Turn on **One renderer process** and **Low end device mode** in Views, Browser Windows and press Restart.

Start ARCx automatically when the Pi boots

Run ARCx as a systemd service so it starts at boot, keeps running when you disconnect, and restarts if it stops unexpectedly.

1. Create the service. Paste this whole block into the terminal as the user who will run ARCx (it fills in your user name and home folder):

```
sudo tee /etc/systemd/system/arcx.service > /dev/null <<EOF
[Unit]
Description=ARCx robot software by Synthiam
Wants=network-online.target
After=network-online.target

[Service]
Type=simple
User=$USER
WorkingDirectory=$HOME/ARCx
ExecStart=$HOME/ARCx/ARCx -b http://0.0.0.0:5000
Restart=on-failure
RestartSec=5
TimeoutStopSec=60

[Install]
WantedBy=multi-user.target
EOF
```

To open a project on start-up, add `-f` and the project path to the `ExecStart` line, in double quotes because the folder name has a space:

```
ExecStart=/home/<you>/ARCx/ARCx -b http://0.0.0.0:5000 -f "/home/<you>/ARCx Data/My Projects/MyRobot.arcx"
```

2.
Turn it on and start it:

```
sudo systemctl daemon-reload
sudo systemctl enable --now arcx
```

Useful service commands:

Command	What it does
`sudo systemctl status arcx`	Is ARCx running?
`journalctl -u arcx -f`	Watch the ARCx console output live (Ctrl+C to stop watching)
`sudo systemctl stop arcx`	Stop ARCx
`sudo systemctl restart arcx`	Restart ARCx
`sudo systemctl disable arcx`	Stop starting ARCx at boot
`top -H -p $(pgrep -of ARCx)`	Show only ARCx, one line per thread (press q to quit)

On Ubuntu Desktop, if sound works when you run `./ARCx` from a terminal but not from the service, the desktop's audio server is only available inside a logged-in session. Either turn on automatic login (**Settings > System > Users > Automatic Login**), or start ARCx from the desktop session instead of from a service. Ubuntu Server has no desktop audio server and plays sound directly, so this does not apply.

Update

When the interface shows that an update is available:

```
# Stop ARCx (press Ctrl+C in its terminal), or if it runs as a service:
sudo systemctl stop arcx

# Remove the old program folder. Your projects are in ~/ARCx Data and are not touched.
cd ~
rm -rf ~/ARCx

# Download and unzip the new version
wget -nd https://synthiam.com/api/download/arcx/ARCx-Linux-ARM64.zip
unzip ARCx-Linux-ARM64.zip -d ARCx
rm ARCx-Linux-ARM64.zip
chmod +x ~/ARCx/ARCx

# Start it again, or if it runs as a service:
sudo systemctl start arcx
```

Removing the old folder first, rather than unzipping over it, makes sure no files from the old version are left behind.

Uninstall

```
# If you set up the service
sudo systemctl disable --now arcx
sudo rm /etc/systemd/system/arcx.service
sudo systemctl daemon-reload

# Remove the program
rm -rf ~/ARCx

# Optional: remove your projects, settings and installed robot skills
```

```
rm -rf ~/"ARCx Data"
```

Troubleshooting

Problem	Fix
`Permission denied` when running `./ARCx`	Run `chmod +x ARCx` in the ARCx folder.
`cannot execute binary file: Exec format error`	The OS is 32-bit. Check with `dpkg --print-architecture` and install Ubuntu 24.04 LTS (64-bit). On an Intel or AMD PC, use the x64 build in Linux-x64.md instead.
The camera shows no picture and the log says `Unable to load shared library 'OpenCvSharpExtern'`	The OS is not Ubuntu 24.04, or the system libraries from the Install step are missing. Run `ldd ~/ARCx/libOpenCvSharpExtern.so | grep "not found"` to list what is missing. On Ubuntu 24.04, re-run the `apt install` line. On any other OS, reinstall with Ubuntu 24.04 LTS (64-bit).
`No such file or directory` when running `./ARCx`	You are not in the folder that contains the `ARCx` program. Run `ls` and `cd` into the right folder.
ARCx exits straight away with an address or port error	Another program is already using port 5000, or another copy of ARCx is running (check `sudo systemctl status arcx`). Stop the other copy, or use another port: `./ARCx -b http://0.0.0.0:5001`
Another device cannot connect	Check that ARCx was started with `-b http://0.0.0.0:5000`, that you used the IP from `hostname -I`, and that both devices are on the same network.
A serial port, camera or microphone is missing or gives "access denied"	Add your user to the hardware groups (see Give ARCx access to hardware) and log out and back in.
No cameras are listed, or `v4l2-ctl --list-devices` says `Failed to open /dev/video0: Permission denied`	The user that runs ARCx cannot open the camera. See A USB camera is not detected.
No sound or microphone, and `aplay -l` says `no soundcards found`	The user that runs ARCx cannot open the sound card. See No sound or microphone.
Start-up log	`~/ARCx Data/Logs`

A USB camera is not detected

ARCx lists a USB camera only if the user that runs ARCx can open the camera's `/dev/video*` device. To see what the Pi detects, install the camera tools and list the cameras:

```
sudo apt install -y v4l-utils
v4l2-ctl --list-devices
```

Each camera should be listed with its `/dev/video*` devices. Most USB cameras have two;

the first one carries the video.
If it prints `Failed to open /dev/video0: Permission denied`, add your user to the `video` group:

```
sudo usermod -aG video $USER
```

Log out and back in (or reboot). The change does not apply to a terminal or SSH session that is already open. If ARCx runs as a service, restart it with `sudo systemctl restart arcx`. Run `groups` to check that `video` is listed, then run `v4l2-ctl --list-devices` again. If it still says `Permission denied`, run `ls -l /dev/video*`. The devices should belong to the `video` group (`crw-rw----+ 1 root video ...`). If they belong to a different group, add a rule that gives them to `video`, then unplug the cameras and plug them back in:

```
echo 'SUBSYSTEM=="video4linux", GROUP="video", MODE="0660"' | sudo tee /etc/udev/rules.d/99-video.rules
sudo udevadm control --reload-rules && sudo udevadm trigger
```

If there are no `/dev/video*` devices at all, the camera was not recognised. Run `lsusb` to check that it is listed, and try another USB port or cable. A USB hub without its own power supply may not give the camera enough power.

No sound or microphone

When ARCx is started over SSH or as a service, it can use the sound card only if its user is in the `audio` group. (A user logged in to the Ubuntu desktop gets access through the desktop session, so sound can work from a desktop terminal but not over SSH.) To check, install the sound tools and list the sound cards as your user and as root:

```
sudo apt install -y alsa-utils
aplay -l
sudo aplay -l
```

If `aplay -l` prints `no soundcards found` but `sudo aplay -l` lists cards, add your user to the `audio` group:

```
sudo usermod -aG audio $USER
```

Log out and back in (or reboot). If ARCx runs as a service, restart it with `sudo systemctl restart arcx`. Run `groups` to check that `audio` is listed.
If `sudo aplay -l` does not list any cards either, the sound hardware was not recognised. For a USB speaker or microphone, run `lsusb` to check that it is listed.
To test the speaker and the microphone:

```
# Play a test sound ("Front Left, Front Right")
speaker-test -c 2 -t wav -l 1

# Record 5 seconds from the microphone, then play it back
arecord -d 5 -f cd /tmp/test.wav
aplay /tmp/test.wav
```

If the cards are listed but nothing plays, run `alsamixer`. Press **F6** to choose the card, press **M** on any channel marked `MM` to unmute it, and raise the volume with the arrow keys. With more than one card, add `-D plughw:<card>,<device>` to `speaker-test`, `arecord` or `aplay` to test a specific one, using the numbers from `aplay -l` or `arecord -l`.

Windows 64

Requirements

- Windows 10 or Windows 11, **64-bit (x64)**
- An internet connection for the download, the robot skill store and sign-in

Nothing else needs to be installed first. ARCx includes everything it needs to run.

Install

1.
Download the installer:
https://synthiam.com/api/download/builder?arcx=true
The file is named `ARCx Installer Windows 64 (Pro).exe`.

2.
Run the installer and follow the prompts.

3.
The installer adds three shortcuts to the Start menu and the desktop. They are explained in the next section.

How ARCx runs: a server and a browser

ARCx is not a normal Windows program with its own window. It comes in two parts:

- **The ARCx server.** This is the program that does the work. It connects to your robot, runs your project, scripts and robot skills, and talks to the Synthiam website. It runs in a black console window full of text messages. That console window *is* ARCx. If it is closed, ARCx is not running.
- **The ARCx interface.** This is what you see and click on. It is a web page that the server provides, so you view it in a web browser such as Edge, Chrome or Firefox. The browser is only a screen for the server. Closing the browser does not stop ARCx or your robot, and opening the browser does not start ARCx.

Because the interface is a web page, it can be opened from this PC or, if you allow it, from a phone, tablet or another computer. The server shortcut you choose decides who is allowed to connect.

You only ever run **one** server at a time. Pick one of the two server shortcuts; do not start both.

The three shortcuts

Shortcut	What it starts	Who can open the interface
ARCx Server (Local Only)	The ARCx server	Only this PC
ARCx Server (Bind Any)	The ARCx server	This PC, plus any device that can reach this PC over the network
ARCx GUI (Launch Browser)	Nothing. It only opens a browser tab	Connects to the server already running on this PC

ARCx Server (Local Only)

Use this one unless you have a reason not to. It is the safest choice.

- It starts the ARCx server and opens the interface in your web browser at `http://127.0.0.1:5000`.
- The server only accepts connections from this PC. A phone, tablet or another computer cannot open the interface, even on your home network, and nothing from the internet can reach it.
- Use it when you sit at the PC that is connected to the robot.

ARCx Server (Bind Any)

Use this one when you want to control ARCx from a different device, such as a phone, a tablet, or a laptop in another room.

- It starts the same ARCx server, but the server accepts connections on every network the PC is connected to (it is started with `-b http://0.0.0.0:5000`).
- On this PC, the interface still opens at `http://127.0.0.1:5000`.
- Any device on the same network can open the interface by browsing to this PC's address, for example `http://192.168.1.25:5000`. See Connect from another device for how to find the address.
- The first time you start it, Windows Firewall asks whether to allow ARCx. Allow it on **Private networks** only.
- It is **not** reachable from the internet unless you deliberately open or forward port 5000 in your router or firewall. If you do that, anyone on the internet who finds the address can use ARCx.

Warning: ARCx has no login screen of its own. Anyone who can open the interface can control the robot, run scripts, and read or delete files on this PC. Only use **Bind Any** on a network you trust, such as your home network. Never use it on public Wi-Fi (coffee shops, hotels, schools, airports).

Do not tick "Stay signed in" if ARCx can be reached from the internet. When you sign in to your Synthiam account in ARCx, the **Stay signed in** box keeps your account signed in on this PC. Anyone who connects to the interface is then using ARCx as you, with your account and your subscription. If ARCx is exposed to the internet, leave **Stay signed in** unticked.

ARCx GUI (Launch Browser)

This shortcut does **not** start ARCx. It is only a web link to `http://127.0.0.1:5000`, the same as typing that address into your browser.

- Use it when the ARCx server is already running (the console window is open) but you closed the browser tab or window. Click it to get the interface back.
- If the server is not running, the browser shows that it cannot connect. Start **ARCx Server (Local Only)** or **ARCx Server (Bind Any)** first.
- You do not have to use this shortcut. Typing `http://127.0.0.1:5000` or `localhost:5000` into the address bar of any browser does the same thing. You can also bookmark it.

Run

1. Start **ARCx Server (Local Only)** or **ARCx Server (Bind Any)** from the Start menu or desktop.
2. A console window opens and shows the ARCx start-up messages. Leave it open; this window *is* ARCx. You can minimize it.
3. Your default web browser opens the interface at `http://127.0.0.1:5000`. If it does not, or if you closed the tab, use the **ARCx GUI (Launch Browser)** shortcut or type that address into any browser.

Stop

Click the ARCx console window and press **Ctrl+C**, or close the console window. Allow around 20 seconds for a clean shutdown.

Closing the browser tab does not stop ARCx. The server and your robot keep running until the console window is closed.

Open a project automatically

To open a project every time ARCx starts, add `-f` and the project file to the server shortcut you use:

1.
Right-click the **ARCx Server (Local Only)** or **ARCx Server (Bind Any)** shortcut and choose **Properties**.

2.
At the end of **Target**, after the closing quote and after anything already there, add:

```
 -f "C:\Users\<you>\Documents\ARCx Data\My Projects\MyRobot.arcx"
```

3.
Click **OK**.

Connect from another device

To use a phone, tablet or another PC:

1. On the PC connected to the robot, start ARCx with the **ARCx Server (Bind Any)** shortcut. Close any ARCx that is already running first.
2. When Windows Firewall asks, allow ARCx on **Private networks** only.
3. Find this PC's IP address: open a Command Prompt and run `ipconfig`. Look for **IPv4 Address** (for example `192.168.1.25`).
4. On the other device, connect to the same network and browse to `http://192.168.1.25:5000` (use your own address). Do not use `127.0.0.1` or `localhost` on the other device; those addresses always mean "this device", not the ARCx PC.

Warning: ARCx has no login. Anyone who can reach that address can control the robot and read or delete files on this PC. Do not allow ARCx on public networks, and do not tick **Stay signed in** if ARCx can be reached from the internet.

Browser windows

Robot skills and scripts can show web pages - a robot face, a dashboard, a video call, a kiosk screen - in Chromium windows on the monitors plugged into this PC. Chromium ships inside ARCx, so there is nothing to install. **Views, Browser Windows** lists them and has a form to try one. If a window opens but stays black or blank, turn off GPU acceleration there and press Restart.
Browser data (cookies and site storage) is kept in `C:\ProgramData\ARCx Data\Browser Data`.

Update

When the interface shows that an update is available:

1. Stop ARCx (Ctrl+C in the console window).
2. Download and run the latest installer from the link above.

Your projects and settings in `Documents\ARCx Data` are kept.

Uninstall

1. Open **Settings > Apps > Installed apps**, find **ARCx**, and choose **Uninstall**.
2. Optional: to remove your projects and settings too, delete these folders:

```
Documents\ARCx Data
C:\ProgramData\ARCx Data
```

Troubleshooting

Problem	Fix
The console window opens and closes straight away	Another program is already using port 5000, or another copy of ARCx is running (for example, both server shortcuts were started). Close the other copy, or start ARCx with a different port: `-b http://127.0.0.1:5001`

The browser says it cannot connect	ARCx is still starting, or it is not running. The **ARCx GUI (Launch Browser)** shortcut does not start ARCx; check that the console window is open, then refresh.
Another device cannot connect	Check that ARCx was started with the **ARCx Server (Bind Any)** shortcut, that the firewall allowed it on the network you are using, that both devices are on the same network, and that you used the PC's IPv4 address, not `127.0.0.1`.
Start-up log	`Documents\ARCx Data\Logs`
A browser window does not appear	Open **Diagnostics** in ARCx; the Browser Windows section says why. Chromium's own log is `C:\ProgramData\ARCx Data\Logs\browser-cef.log`.

Requirements

- Windows 10 or Windows 11, **64-bit (x64)**
- An internet connection for the download, the robot skill store and sign-in

Nothing else needs to be installed first. ARCx includes everything it needs to run.

Install

1.
Download the installer:
https://synthiam.com/api/download/builder?arcx=true
The file is named `ARCx Installer Windows 64 (Pro).exe`.

2.
Run the installer and follow the prompts.

3.
The installer adds two shortcuts to the Start menu and the desktop:

Shortcut
What it does

ARCx
Starts ARCx

ARCx GUI
Opens the ARCx interface (`http://127.0.0.1:5000`) in your web browser

Run

1. Start **ARCx** from the Start menu or desktop.
2. A console window opens and shows the ARCx start-up messages. Leave it open; this window *is* ARCx.
3. Your default web browser opens the interface at `http://127.0.0.1:5000`. If it does not, or if you closed the tab, use the **ARCx GUI** shortcut or type that address into any browser.

Stop

Click the ARCx console window and press **Ctrl+C**, or close the console window. Allow around 20 seconds for a clean shutdown.
Closing the browser tab does not stop ARCx.

Open a project automatically

To open a project every time ARCx starts, add `-f` and the project file to the shortcut:

1.
Right-click the **ARCx** shortcut and choose **Properties**.

2.
At the end of **Target**, after the closing quote, add:
```
 -f "C:\Users\<you>\Documents\ARCx Data\My Projects\MyRobot.arcx"
```

3.
Click **OK**.

Connect from another device

By default only this PC can open the interface. To use a phone, tablet or another PC:

1.
Right-click the **ARCx** shortcut, choose **Properties**, and add this to the end of **Target**:
```
 -b http://0.0.0.0:5000
```

2.
Start ARCx from that shortcut. When Windows Firewall asks, allow ARCx on **Private networks** only.

3.
Find this PC's IP address: open a Command Prompt and run `ipconfig`. Look for **IPv4 Address** (for example `192.168.1.25`).

4.
On the other device, browse to `http://192.168.1.25:5000` (use your own address).

Warning: ARCx has no login. Anyone who can reach that address can control the robot and read or delete files on this PC. Do not allow ARCx on public networks.

Browser windows

Robot skills and scripts can show web pages - a robot face, a dashboard, a video call, a kiosk screen - in Chromium windows on the monitors plugged into this PC. Chromium ships inside ARCx, so there is nothing to install. **Views, Browser Windows** lists them and has a form to try one. If a window opens but stays black or blank, turn off GPU acceleration there

and press Restart.
Browser data (cookies and site storage) is kept in `C:\ProgramData\ARCx Data\Browser Data`.

Update

When the interface shows that an update is available:

1. Stop ARCx (Ctrl+C in the console window).
2. Download and run the latest installer from the link above.

Your projects and settings in `Documents\ARCx Data` are kept.

Uninstall

1. Open **Settings > Apps > Installed apps**, find **ARCx**, and choose **Uninstall**.
2. Optional: to remove your projects and settings too, delete these folders:

```
Documents\ARCx Data
C:\ProgramData\ARCx Data
```

Troubleshooting

Problem	**Fix**
The console window opens and closes straight away	Another program is already using port 5000, or another copy of ARCx is running. Close the other copy, or start ARCx with a different port: `-b http://127.0.0.1:5001`
The browser says it cannot connect	ARCx is still starting, or it is not running. Check the console window, then refresh.
Another device cannot connect	Check that ARCx was started with `-b http://0.0.0.0:5000`, that the firewall allowed it on the network you are using, and that both devices are on the same network.
Start-up log	`Documents\ARCx Data\Logs`
A browser window does not appear	Open **Diagnostics** in ARCx; the Browser Windows section says why. Chromium's own log is `C:\ProgramData\ARCx Data\Logs\browser-cef.log`.