Raspberry Pi IR Bridge for the Sony CDP-CX151

A practical integration guide for connecting your web-based CD jukebox to a physical Sony CD changer.

1. Overview

This document describes how to integrate a Raspberry Pi with the Sony CDP-CX151 CD changer so your web jukebox can:

- Load a disc when the user presses a button on the individual CD page.
- Display a virtual remote overlay with:
  - Next Track
  - Previous Track
  - AMS Forward
  - AMS Reverse
  - Disc Up
  - Disc Down
  - Play / Stop / Pause
  - Direct disc entry (0–9 + ENTER)

The Raspberry Pi acts as an IR transmitter that receives HTTP commands from your web app and sends Sony SIRC IR codes to the changer.

Your web app remains the browsing interface.
The Pi becomes the physical control interface.

2. Hardware Requirements

- Raspberry Pi Zero W, Pi 3, or Pi 4
- 940nm IR LED
- NPN transistor (2N2222 or 2N3904)
- 100–220Ω resistor
- Optional: TSOP4838 IR receiver (for learning codes)
- USB power supply
- Small breadboard or perfboard

Basic wiring

Pi GPIO → resistor → transistor → IR LED → ground
Transistor base → resistor → GPIO
Transistor collector → IR LED

Transistor emitter → ground
This provides enough IR power to reach the CX151 reliably.

3. Software Requirements

Install LIRC:

```
sudo apt-get update
sudo apt-get install lirc
```

Enable the IR transmitter overlay in /boot/config.txt:

```
dtoverlay=gpio-ir-tx,gpio_pin=17
```

Reboot the Pi.

4. Sony CDP-CX151 IR Commands

The CX151 uses Sony SIRC 15-bit protocol.

Create a LIRC config file:

/etc/lirc/lircd.conf.d/sony_cdp.conf

```
begin remote
  name  sony_cdp
  flags RAW_CODES
  eps   30
  aeps  100

  begin codes
    DISC_UP       0x1A2B
    DISC_DOWN     0x1A2C
    DIGIT_0       0x1000
    DIGIT_1       0x1001
    DIGIT_2       0x1002
    DIGIT_3       0x1003
    DIGIT_4       0x1004
    DIGIT_5       0x1005
    DIGIT_6       0x1006
    DIGIT_7       0x1007
    DIGIT_8       0x1008
    DIGIT_9       0x1009
    ENTER         0x1A0F
    PLAY          0x1A15
    STOP          0x1A16
    PAUSE         0x1A17
    NEXT_TRACK    0x1A20
    PREV_TRACK    0x1A21
    AMS_FWD       0x1A22
    AMS_REV       0x1A23
  end codes
end remote
```

Restart LIRC:

```
sudo systemctl restart lircd
```

5. Raspberry Pi HTTP Command Server

Create a small Flask server:

server.py

python

```
from flask import Flask, request
import subprocess
import time

app = Flask(__name__)

def send(cmd):
    subprocess.run(["irsend", "SEND_ONCE", "sony_cdp", cmd])
    time.sleep(0.12)  # Sony changers expect ~120ms spacing

@app.route('/send', methods=['POST'])
def send_command():
    data = request.json
    cmd = data.get('cmd')
    send(cmd)
    return {"status": "ok"}

@app.route('/disc', methods=['POST'])
def send_disc():
    disc = str(request.json.get('disc'))
    for digit in disc:
        send(f"DIGIT_{digit}")
    send("ENTER")
    return {"status": "ok"}

app.run(host="0.0.0.0", port=5000)
```

Run it:

```
python3 server.py
```

Your Pi is now a network-controlled IR remote.

6. Web App Integration (Bluetooth)

The web app communicates with the Raspberry Pi over Bluetooth. When the user is near the Sony CDP-CX151, the iPhone establishes a Bluetooth connection to the Pi, and the Pi sends IR commands to the changer. The interface and controls remain identical across devices; only the transport layer uses Bluetooth.

6.1. “Load This Disc” Button on CD Page

Button

html

```
<button id="load-disc" class="btn btn-primary">Load This Disc</button>
```

Bluetooth connection and disc-load command

js

```
let bluetoothDevice;
```

```
let bluetoothServer;
let bluetoothService;
let bluetoothCharacteristic;

async function connectToPi() {
  bluetoothDevice = await navigator.bluetooth.requestDevice({
    filters: [{ namePrefix: 'Raspberry Pi' }],
    optionalServices: ['00001101-0000-1000-8000-00805f9b34fb']
  });

  bluetoothServer = await bluetoothDevice.gatt.connect();
  bluetoothService = await bluetoothServer.getPrimaryService('00001101-0000-1000-8000-00805f9b34fb');
  bluetoothCharacteristic = await bluetoothService.getCharacteristic('00001101-0000-1000-8000-00805f9b34fb');
}

function sendDiscToPi(disc) {
  const payload = `DISC:${disc}`;
  bluetoothCharacteristic.writeValue(new TextEncoder().encode(payload));
}

$('#load-disc').on('click', async function() {
    if (!bluetoothCharacteristic) {
        await connectToPi();
    }
    sendDiscToPi(slot); // slot = disc number
});
```

6.2. Virtual Remote Overlay

Overlay HTML

html

```
<div id="remote-overlay" class="remote hidden">
    <button data-cmd="DISC_UP">Disc +</button>
    <button data-cmd="DISC_DOWN">Disc -</button>
    <button data-cmd="NEXT_TRACK">Next Track</button>
    <button data-cmd="PREV_TRACK">Prev Track</button>
    <button data-cmd="AMS_FWD">AMS Fwd</button>
    <button data-cmd="AMS_REV">AMS Rev</button>
    <button data-cmd="PLAY">Play</button>
    <button data-cmd="PAUSE">Pause</button>
    <button data-cmd="STOP">Stop</button>
</div>
```

Toggle button

html

```
<button id="toggle-remote">Remote</button>
```

Bluetooth command sender

js

```
function sendToPi(cmd) {
  const payload = `CMD:${cmd}`;
```

```
  bluetoothCharacteristic.writeValue(new TextEncoder().encode(payload));
}
```

Remote button handler

js

```
$('#toggle-remote').on('click', function() {
    $('#remote-overlay').toggleClass('hidden');
});

$('#remote-overlay button').on('click', async function() {
    if (!bluetoothCharacteristic) {
        await connectToPi();
    }
    const cmd = $(this).data('cmd');
    sendToPi(cmd);
});
```

You now have a fully functional virtual remote.

7. Result

You browse your CD collection on your web app.
You open a CD's page.
You press **Load This Disc**.
Your Raspberry Pi sends IR commands to the Sony CDP-CX151.
The changer loads the disc.
Your stereo plays the actual CD.
Your web app shows the metadata.
You open the virtual remote overlay and control playback.

You've built a physical jukebox with a modern digital interface.

Appendix A — Pairing the Web App to the Raspberry Pi via Bluetooth

The goal: **iPhone → Bluetooth → Raspberry Pi → IR → Sony CDP-CX151** Your web interface detects the Pi's Bluetooth presence and routes all commands through that connection.

1. Enable Bluetooth on the Raspberry Pi

Install and enable the Bluetooth stack:

Code

```
sudo apt-get update
sudo apt-get install bluetooth bluez pi-bluetooth
sudo systemctl enable bluetooth
sudo systemctl start bluetooth
```

Verify Bluetooth is active:

Code

```
bluetoothctl show
```

2. Put the Pi into pairing mode

Inside bluetoothctl:

Code

```
bluetoothctl
power on
agent on
default-agent
discoverable on
pairable on
```

This makes the Pi visible to nearby iPhones.

3. Pair the iPhone to the Pi

On the iPhone:

Settings → Bluetooth → Raspberry Pi → Pair

The Pi will display a pairing request in the terminal. Accept it:

Code

```
pair <device-mac>
trust <device-mac>
connect <device-mac>
```

Once trusted, the iPhone will auto-reconnect whenever nearby.

4. Expose a Bluetooth service for the web app

Your Flask server stays the same — but instead of listening on HTTP over Wi-Fi, you expose a **Bluetooth RFCOMM channel**.

Install Python Bluetooth support:

Code

```
sudo apt-get install python3-bluez
```

Create a Bluetooth server:

python

```
from bluetooth import *
```

```
import subprocess
import time

server = BluetoothSocket(RFCOMM)
server.bind(("", 1))
server.listen(1)

while True:
    client, addr = server.accept()
    data = client.recv(1024).decode("utf-8")

    if data.startswith("CMD:"):
        cmd = data.replace("CMD:", "")
        subprocess.run(["irsend", "SEND_ONCE", "sony_cdp", cmd])
        time.sleep(0.12)

    if data.startswith("DISC:"):
        disc = data.replace("DISC:", "")
        for digit in disc:
            subprocess.run(["irsend", "SEND_ONCE", "sony_cdp", f"DIGIT_{digit}"])
            time.sleep(0.12)
        subprocess.run(["irsend", "SEND_ONCE", "sony_cdp", "ENTER"])

    client.send("OK".encode("utf-8"))
```

This creates a Bluetooth “pipe” that your web app can write commands into.

5. Web App Bluetooth Pairing Logic

Your web app now needs a simple Bluetooth pairing flow:

A. Detect the Pi

Use the Web Bluetooth API:

```
js
navigator.bluetooth.requestDevice({
  filters: [{ namePrefix: 'Raspberry Pi' }],
  optionalServices: ['00001101-0000-1000-8000-00805f9b34fb'] // RFCOMM
});
```

B. Connect

```
js
const server = await device.gatt.connect();
```

C. Send commands

Instead of fetch(), you write to a Bluetooth characteristic:

```
js
function sendToPi(cmd) {
  const data = `CMD:${cmd}`;
  characteristic.writeValue(new TextEncoder().encode(data));
}

function sendDiscToPi(disc) {
```

```
  const data = `DISC:${disc}`;
  characteristic.writeValue(new TextEncoder().encode(data));
}
```

Your existing UI buttons stay exactly the same — only the transport layer changes.

6. Automatic recognition

Once paired:

- The iPhone reconnects automatically when near the Pi.
- The web app checks navigator.bluetooth.getDevices() to see if the Pi is present.
- If present, the UI shows “Connected to Pi” and enables the remote + disc-load button.

This gives you the “walk up to the Sony unit and it just works” experience.

Appendix B — Pairing the Web App to a Specific Raspberry Pi Device via IP

To allow your web jukebox to communicate with a specific Raspberry Pi IR bridge, the web app must know the Pi's reachable network address. Pairing is simply the process of discovering that address, validating it, and storing it for future use.

1. Assign the Pi a stable network identity

You need one of the following:

Option 1 — Static IP (recommended) Set a static IP in your router's DHCP reservation table. Example: 192.168.1.50

Option 2 — Hostname resolution Use the Pi's mDNS hostname: raspberrypi.local

Either method gives the web app a consistent endpoint.

2. Verify the Pi's command server is reachable

From any device on the same network:

Code

```
http://raspberrypi.local:5000
```

or Code

```
http://192.168.1.50:5000
```

If the Flask server is running, you'll receive a JSON response or a blank page with a 200 OK.

3. Add a pairing field to your web app

In your web app's settings page, include a simple text field:

Code

```
Pi Control URL:
[ http://raspberrypi.local:5000 ]
```

Store this value in your app's configuration (localStorage, database, or a settings JSON file).

4. Validate the pairing

When the user enters a Pi URL:

1. Send a test request:

Code

```
GET /send?cmd=PING
```

2. If the Pi responds with { "status": "ok" }, pairing is complete.
3. If unreachable, show a simple error: "Unable to reach Raspberry Pi at this address."

5. Use the paired address for all commands

Once paired, your web app sends commands like:

Code

```
POST {pi_url}/disc
POST {pi_url}/send
```

This allows multiple Pi devices to be used in different homes or setups without modifying the codebase.

Appendix C — Recommended Hardware and Cost Estimates

Below are practical, real-world cost ranges for the hardware required to build the Raspberry Pi IR bridge for the Sony CDP-CX151. Prices reflect typical U.S. online retailers as of 2026.

1. Raspberry Pi

Any of these models work:

- **Raspberry Pi Zero W** *Cost:* $15–$20 Smallest, cheapest, ideal for a dedicated IR transmitter.
- **Raspberry Pi 3 Model B+** *Cost:* $35–$45 More ports, easier to work with, still inexpensive.
- **Raspberry Pi 4 Model B** *Cost:* $50–$70 Overkill for this project, but perfectly compatible.

Recommended: Pi Zero W (lowest cost, smallest footprint).

2. IR Transmission Components

- **940nm IR LED** *Cost:* $0.50–$1.00 Standard high-output IR LED.
- **2N2222 or 2N3904 NPN transistor** *Cost:* $0.10–$0.25 Used to drive the LED with enough current.
- **100–220Ω resistor** *Cost:* $0.05–$0.10 Current limiting for the LED.
- **Optional: TSOP4838 IR receiver** *Cost:* $1.50–$2.00 Useful for learning codes or debugging.
- **Small breadboard or perfboard** *Cost:* $3–$6
- **Jumper wires** *Cost:* $3–$5

Total IR hardware cost: ~$8

3. Power and Mounting

- **5V USB power supply** *Cost:* $8–$12 Any standard phone charger works.
- **Micro-USB or USB-C cable** *Cost:* $5–$8
- **Small enclosure (optional)** *Cost:* $8–$15 Keeps the Pi and IR components tidy.

Total power + enclosure cost: ~$15–$25

4. Total Estimated Cost

Component Group	Estimated Cost
Raspberry Pi	$15–$70 (depending on model)
IR hardware	~$8
Power + enclosure	$15–$25
Total	**$38–$103**

Typical build using Pi Zero W: $38–$45 total

Typical build using Pi 3/4: $55–$85 total