**Project Summary**

This project converts an old rotary telephone into a **custom audio playback device**. When you lift the handset, instead of dialing, it automatically plays **preloaded voice recordings or sound clips** (family, celebrities, fictional characters, etc.) through the original phone’s speaker.

(A UN-showcase ready build guide)

**1. Core Requirements**

* **Vintage rotary/feature phone** (working or not — only casing + handset needed).
* **High-quality, stable audio playback**:
  + Plays **10+ min audio tracks** (podcasts, science narratives, personal voices).
  + Stops playback on hang-up, resumes with new random track on next lift.
  + Audio quality ≥ **44.1 kHz / 16-bit WAV/MP3** (CD quality).
* **Durability & Reliability**:
  + Must work flawlessly during multi-day exhibit.
  + Electronics should be neatly fitted, hidden, and robust.
* **Speaker Quality**:
  + Test vintage handset speaker.
  + If poor, replace with **8Ω dynamic speaker (same size/shape)**, matched to amplifier.

**2. Hardware Components**

**Phone & Mechanical**

* Vintage rotary or feature phone (clean, sturdy shell).
* Original **handset with speaker & mic** (if mic unused, keep for aesthetics).

**Control & Playback**

* **Raspberry Pi Zero 2 W** (small, reliable, good audio with DAC).
  + Alternative: **Pi 3A+** (if more power needed).
* **MicroSD card** (16–32GB, Class 10) → stores OS + audio files.
* **USB power supply** (5V 2A, high quality — stable).

**Audio Chain**

* **I²S DAC HAT for Raspberry Pi** (e.g., Adafruit MAX98357A or HiFiBerry DAC Zero).
  + Ensures high-fidelity output vs. noisy Pi headphone jack.
* **Amplifier (if needed)**:
  + PAM8403 or MAX98357A (3W, 5V) → to drive 8Ω handset speaker.

**Switching & Control**

* **Magnetic reed switch / microswitch** (detects handset on/off hook).
  + Mounted under hook switch → closes circuit when handset lifted.
* **GPIO breakout wires**.
* **Resistors (10kΩ pull-down for GPIO)**.

**3. Wiring & Assembly**

1. **Disassemble Phone**:
   1. Remove old dial mechanism (optional, for space).
   2. Identify hook switch terminals (the two contacts that open/close with handset).
2. **Speaker Check**:
   1. Measure with multimeter (~8Ω typical).
   2. Test with small amp + phone source (if muffled/distorted → replace with modern 8Ω handset speaker, same size).
3. **Install Reed/Micro Switch**:
   1. Wire one side → **Pi GPIO pin**.
   2. Other side → **GND**.
   3. Add 10kΩ resistor pull-down.

→ Logic: GPIO HIGH when handset lifted, LOW when down.

1. **DAC/Amplifier**:
   1. Pi I²S pins → DAC HAT → Amplifier → Handset speaker.
   2. Mount amp inside phone base.
2. **Power Routing**:
   1. 5V 2A USB → Pi & Amp.
   2. Optional: hidden inline switch for reset.

**4. Software Setup**

**Raspberry Pi OS Lite**

1. Flash **Raspberry Pi OS Lite** (no desktop) to microSD.
2. SSH enabled for headless config.

**Install Audio Tools**

sudo apt update && sudo apt upgrade  
sudo apt install mpg123 python3-pip  
pip3 install pygame RPi.GPIO

**Audio Folder**

* /home/pi/audio/ → store .mp3 or .wav files.
* 44.1kHz/16bit recommended.

**5. Python Control Script**

Core logic:

* On handset lift → play random audio.
* On hang-up → stop immediately.
* Loop.

import os, random, RPi.GPIO as GPIO, subprocess, signal, time  
  
HOOK\_PIN = 18 # GPIO pin connected to hook switch  
AUDIO\_DIR = "/home/pi/audio/"  
current\_process = None  
  
GPIO.setmode(GPIO.BCM)  
GPIO.setup(HOOK\_PIN, GPIO.IN, pull\_up\_down=GPIO.PUD\_DOWN)  
  
def play\_random\_audio():  
 global current\_process  
 file = random.choice(os.listdir(AUDIO\_DIR))  
 path = os.path.join(AUDIO\_DIR, file)  
 current\_process = subprocess.Popen(["mpg123", path])  
  
def stop\_audio():  
 global current\_process  
 if current\_process:  
 current\_process.send\_signal(signal.SIGTERM)  
 current\_process = None  
  
try:  
 while True:  
 if GPIO.input(HOOK\_PIN): # Handset lifted  
 if not current\_process:  
 play\_random\_audio()  
 else: # Handset down  
 stop\_audio()  
 time.sleep(0.2)  
except KeyboardInterrupt:  
 GPIO.cleanup()

* Runs on boot via crontab or systemd.

**6. Testing & QA**

1. Test **hook switch logic** with LED first.
2. Verify **speaker frequency response** (replace if crackly).
3. Test **10+ min audio playback** (no lag/skips).
4. Verify **random file change** each pickup.
5. Stress test: 100+ lift/hang cycles.

**7. Presentation Polish**

* Hide Pi + amp inside base.
* Secure with hot glue/3D mounts.
* Keep original phone dial/keys intact for authenticity.
* Optionally add **label plate** with project info + QR code.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Step-by-Step Process**

1. **Materials**
   1. Old rotary telephone
   2. USB cable (for power and file loading)
   3. **Adafruit Audio FX Soundboard** (the key component for storing and playing sounds)
   4. Soldering iron + connectors/jumper wires
2. **Disassembly**
   1. The creator opens up the rotary phone and removes unnecessary mechanical parts.
   2. Keeps the **handset speaker** to reuse for sound output.
3. **Wiring & Connectors**
   1. Prepares wires: U-shaped connectors + jumper wires.
   2. Soldered pins onto the Adafruit soundboard.
   3. Connected the soundboard to the correct **rotary phone terminals** so it’s triggered when the handset is lifted.
4. **Programming the Soundboard**
   1. Loads custom audio files onto the Adafruit board (e.g., “Hi, this is Grandpa,” “Hello, I’m Batman”).
   2. Board is configured to automatically play a sound file once the phone goes off-hook.
5. **Testing & Assembly**
   1. Tests playback through the original phone’s earpiece.
   2. Once working, reassembles the phone with the hidden soundboard inside.

**Outcome**

* The rotary phone looks authentic but **acts like a voice jukebox**.
* Lifting the handset triggers random or specific recordings.
* Could be used for family memories, fun projects, or interactive art installations.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Key checks before final assembly:

* Use a DAC HAT rather than the Pi headphone jack to guarantee clean, 44.1 kHz audio.
* Confirm handset speaker response with test tones (200 Hz–8 kHz). Replace if weak.
* Mount reed/micro switch firmly to avoid misfires in long demos.
* Stress-test playback with ≥10 min tracks to confirm no buffer underruns.
* Use systemd instead of crontab for boot script—more reliable for exhibits.