00 // QUICK START — YOUR FIRST WIN IN UNDER 60 MINUTES

This page is the only thing you need to read first. Follow these five steps in order.
You will hear JARVIS say your name before the hour is up. Everything else builds from there.

BEFORE YOU BEGIN — CREATE THESE 3 ACCOUNTS (FREE TIERS ARE FINE)

01 ANTHROPIC — console.anthropic.com

Brain. Create account → API Keys → New Key. Save it somewhere safe.

02 ELEVENLABS — elevenlabs.io

Voice. Free tier = 10k chars/month. More than enough to start.

03 N8N — n8n.io

Automation layer. Skip for now unless you want business workflows.

STEP 1 — INSTALL PYTHON 3.11+

Windows: `winget install Python.Python.3.11` (PowerShell, run as admin)
Mac: `brew install python@3.11`
Verify: `python --version` should show 3.11.x or higher

STEP 2 — INSTALL CORE DEPENDENCIES

```
pip install anthropic elevenlabs sounddevice pyaudio pyttsx3 numpy scipy requests
```

NOTE: pyaudio needs extra setup on some machines.
Windows: the command above usually works directly.
Mac: run `brew install portaudio` first, then pip install pyaudio

STEP 3 — CREATE YOUR .ENV FILE

Create a file called `.env` in your project folder. Paste these two lines:

```
ANTHROPIC_API_KEY=   sk-ant-YOUR-KEY-HERE
ELEVENLABS_API_KEY=   YOUR-ELEVENLABS-KEY-HERE
```

STEP 4 — YOUR FIRST JARVIS SCRIPT (COPY THIS EXACTLY)

```
# jarvis_hello.py — paste this into a new file
import os, anthropic
from dotenv import load_dotenv
from elevenlabs.client import ElevenLabs
from elevenlabs import play

load_dotenv()

YOUR_NAME = "Yash"   # change this

brain   = anthropic.Anthropic()
voice   = ElevenLabs()

response = brain.messages.create(
    model   = "claude-opus-4-5",
    max_tokens = 80,
    messages = [{
        "role": "user",
        "content": f"Say a short greeting to {YOUR_NAME}. Be direct."
    }]
)

text = response.content[0].text
print(f"JARVIS: {text}")

audio = voice.generate(text=text, voice="Adam")
play(audio)   # you will hear JARVIS speak
```

Also run: `pip install python-dotenv` if load_dotenv throws an error.

STEP 5 — RUN IT

Open terminal in your project folder.

Run: `python jarvis_hello.py`

You should see the greeting printed, then hear it spoken aloud.

If audio fails: check your speaker is not muted and try again.

If API error: double-check your .env keys have no spaces around the = sign.

IT WORKS. YOU JUST BUILT THE FOUNDATION OF YOUR AI SYSTEM.

JARVIS said your name. Everything from here is architecture.
Follow @larossatech — the next layer goes up in public.

WHAT YOU HAVE BUILT SO FAR

[BRAIN] Claude API connected. JARVIS can think, reason, and respond.

[VOICE] ElevenLabs connected. JARVIS can speak in a natural AI voice.

[ENV] API keys stored securely. Not hardcoded. Professional setup.

[SCRIPT] A working Python entrypoint. This is the seed of the full system.

TROUBLESHOOTING — MOST COMMON ISSUES

No audio output:
Run: `python -c "import sounddevice; print(sounddevice.query_devices())"`
Find your speaker index and add: `sd.default.device = [INDEX]`

pyaudio fails:
Windows: `pip install pipwin` then `pipwin install pyaudio`

API key error:
Check .env has no quotes around keys. Key=value not Key="value"

ModuleNotFound:
Run all pip install commands again inside your virtual env

05 // WHAT YOU CAN BUILD FROM HERE

You now have a working voice AI that can think and speak. The full JARVIS system is this foundation multiplied by six months of architecture decisions, agent design, and integration work. Below is what the complete system adds — documented separately.

[REDACTED]	**WAKE WORD DETECTION**	Offline phoneme matching. Three trigger phrases, sub-400ms latency.
[REDACTED]	**CONVERSATION MEMORY**	Rolling history buffer passed to Claude. JARVIS remembers the session.
[REDACTED]	**MORNING BRIEF SYSTEM**	Google Calendar + weather + priorities read aloud at 09:00 daily.
[REDACTED]	**BUSINESS AGENTS**	Specialised modules for finance, sales, ops, content, strategy.
[REDACTED]	**n8n AUTOMATION LAYER**	Workflows running 24/7. Lead follow-ups, reports, alerts, CRM sync.
[REDACTED]	**PWA DEPLOYMENT**	JARVIS lives on your iPhone home screen. No app store needed.
[REDACTED]	**DOUBLE-CLAP WAKE**	Hardware pattern detection. Clap twice to activate, no phrase needed.
[REDACTED]	**WHOOP INTEGRATION**	Biometric data piped into morning brief. HRV, sleep score, readiness.
[REDACTED]	**SECURITY LAYER**	API key rotation, PIN auth, access control, anomaly detection.
[REDACTED]	**VLC MUSIC SERVER**	Music on launch, scene-based playlists, volume control via voice.

WANT THE FULL BUILD?

The complete JARVIS system is a live build documented in real time.

Follow along: @larossatech // Building in public.

JARVIS PRO — full deployment — coming soon. Join the waitlist.

J.A.R.V.I.S.

JUST A RATHER VERY INTELLIGENT SYSTEM

SYSTEM ARCHITECTURE + DEPLOYMENT OVERVIEW

RUNTIME ENV	VOICE ENGINE	BRAIN	INTERFACE	TRIGGER MODES
Python 3.11+	ElevenLabs API	Claude API	HTML5 / PWA	3 Wake Phrases

01 // SYSTEM REQUIREMENTS

JARVIS operates across a multi-layer stack requiring careful environment configuration. Dependency conflicts between audio libraries are the most common point of failure. Ensure Python version compatibility before proceeding. Windows 10/11 recommended. A dedicated machine or partition is strongly advised for stable wake-word detection.

[CORE]	python >= 3.11	sounddevice	pyaudio
[AUDIO]	pyttsx3	elevenlabs-sdk	vlc (system)
[BRAIN]	anthropic	requests	threading
[UI]	http.server	webbrowser	subprocess
[CLAP]	numpy	scipy	audioop

NOTE: AUDIO DEVICE INDEXING VARIES PER SYSTEM. EXPECT 1-3 HOURS CALIBRATION.

02 // ARCHITECTURE OVERVIEW

The system runs as a persistent Python process listening on a calibrated microphone channel. A rolling audio buffer (2-second window) is continuously compared against phoneme fingerprints for the three trigger phrases. Match confidence threshold is tunable — set too low and ambient noise triggers false wakes; too high and it misses.

```
MIC INPUT  -->  BUFFER  -->  WAKE DETECTOR
                                   |
                          [ PHRASE MATCHED ]
                           /        |        \
                  FULL LAUNCH    CHAT ONLY    SCREENSAVER
                      |              |
                ElevenLabs TTS   Claude API
                + VLC Music      Conversation Loop
```

03 // WAKE WORD CONFIGURATION

Three discrete trigger states are mapped to system behaviours. Phrase detection uses a rolling comparison window — the system does NOT use cloud-based hotword detection. This keeps latency under 400ms but requires careful mic gain calibration in your specific acoustic environment. USB condenser mics perform significantly better than built-in laptop arrays for phrase isolation.

```
PHRASE 1:  "Wake up JARVIS"    -->  Full launch + music + greeting
PHRASE 2:  "Hey JARVIS"        -->  Chat mode only (no music launch)
PHRASE 3:  "Goodnight JARVIS"  -->  Matrix screensaver + system sleep

Sensitivity:  tuned via threshold float in config block
False positive rate: ~2-4% in quiet environments (acceptable)
```

04 // VOICE + AI INTEGRATION

Two-layer voice pipeline: pyttsx3 handles offline fallback; ElevenLabs delivers the final production-grade voice output via API streaming. The system routes through ElevenLabs by default and falls back automatically on API failure. Conversation history is maintained in a rolling buffer passed to Claude on each turn — this gives JARVIS contextual memory within a session.

```
ELEVENLABS VOICE ID:  [ configured in env / not shown here ]
ANTHROPIC MODEL:      claude-sonnet (via API key in .env)
CONVERSATION TURNS:   full history passed per request
FALLBACK:             pyttsx3 local TTS (David / Mark voice)
STREAMING:            disabled — full response before playback
```

05 // WHAT'S NOT IN THIS GUIDE

This document covers the foundational architecture only. The following components are outside the scope of this release and are documented separately — or not at all.

```
[ REDACTED ]  WHOOP biometric integration + health briefing pipeline
[ REDACTED ]  n8n automation layer + business agent orchestration
[ REDACTED ]  PWA deployment + iPhone home screen configuration
[ REDACTED ]  Double-clap wake detection (hardware pattern matching)
[ REDACTED ]  Google Calendar morning briefing sequence
[ REDACTED ]  Full environment variable + API key management
[ REDACTED ]  VLC music server + playback control integration
```

// ADVANCED DOCUMENTATION NOT AVAILABLE IN PUBLIC RELEASE.
// FOLLOW @LAROSSA_TECH FOR ONGOING BUILD UPDATES.

WANT THE FULL BUILD?

The complete JARVIS system is a 6-month live build documented in real time.
Follow along: @larossa_tech // Building in public.