🤖 DESKBOT — Complete Build Guide

Hardware Assembly · Wiring · Software Setup · Usage

Table of Contents

📦 Component List

From Your Existing JINX Build (Already Have)

Component	Purpose in Deskbot
ESP32-WROOM-32 DevKit	Robot brain (motors, display, LEDs, sensors)
2.4" TFT ILI9341 Display	Animated eyes
2× SG90 Servo Motors	Head pan + tilt for face tracking
WS2812B LED Strip	Mood lighting
DFPlayer Mini + 3W Speaker	Sound effects
HC-SR04 Ultrasonic (×2)	Obstacle avoidance (front + sides)
IR Sensors (×2)	Desk edge detection (table fall prevention)
Active Buzzer	Alerts
18650 Battery Pack (2S2P)	Power

Component	Purpose in Deskbot
2S BMS Board	Battery protection
TP4056 Charging Modules	Charging
L298N Motor Driver	Motor control
Redmi Note 12	Primary camera + mic + Gemini voice
UP Govt Tablet	Main display (skeleton/scanning feed)
ThinkPad T61 Chassis Parts	Decoration + structural body
Jumper Wires, Breadboard	Wiring

Additional Components to Get (Budget: ~₹800)

Component	Cost	Why Better
VL53L0X ToF Sensor (×2)	₹150 each	Precise depth (2m range, 1mm accuracy) — detects table edges and drops far better than IR
Mini Rocker Switch	₹20	Main power
3D-printed / cardboard body	₹0-200	Structure around the ThinkPad parts
USB-C breakout board	₹60	Clean charging port
10kΩ resistors (×4)	₹10	Voltage divider for battery monitor

> **Why VL53L0X over IR for depth?**
> IR sensors only detect "object present / not present". The VL53L0X gives you an exact distance in mm via I2C — perfect for knowing you're 5cm from a table edge vs 50cm. One facing down under the base prevents table falls; one facing forward gives precise obstacle distance.

🔧 Improvements Over Original JINX Plan

Original Plan	Upgraded Version	Reason
pyttsx3 (robotic voice)	**edge-TTS** (Microsoft Neural)	Free, runs offline after first cache, sounds fully human
Gemini only	**Gemini + local Ollama fallback**	Works without internet
IP Webcam stream	**DroidCam + WebRTC**	Lower latency, better quality

Original Plan	Upgraded Version	Reason
Streamlit dashboard	**Flask + React frontend**	Phone controllable, real-time WebSocket
Network anomaly only	**Network + Document RAG agent**	Upload PDFs, ask questions about them
No code agent	**LLM code review via file watch**	Connect laptop, auto-reviews on save
IR for table edge	**VL53L0X ToF**	Exact depth measurement

🔩 Mechanical Assembly

Phase 1: Base Structure

Step-by-Step Assembly

Step 1 — Base Plate

- Use the ThinkPad T61 bottom cover as your base plate
- Drill/cut 4 corner holes if needed for rubber feet
- Mount ESP32 and L298N using standoffs or hot glue on the base
- Place battery pack at the back for counterweight

Step 2 — Sensor Mounting

- Mount 2× HC-SR04 on the front, angled outward ~15° for wider obstacle coverage
- Mount 2× VL53L0X: one facing **straight down** under the front lip (table edge), one facing **forward** (precise distance)
- Mount 2× IR sensors: one on each side facing DOWN — these catch the exact table edge moment the ToF misses from below

Step 3 — Neck Assembly

- Stack servos: Pan servo horizontal on base → Tilt servo vertical → Head plate on top
- Use servo horns and small L-brackets (can be cut from ThinkPad casing)
- Secure with hot glue + small screws

Step 4 — Head Unit

- Build a small rectangular enclosure (cardboard/3D print/ThinkPad parts)
- Mount 2.4" TFT display centered on the front face
- Redmi Note 12 mounts on the back of the head (camera facing out through a hole, or just have it protrude slightly for better FOV)
- DFPlayer Mini + speaker mount inside or under the head

Step 5 — LED Strip

- Run WS2812B strip along the base perimeter
- Can also add strips behind the head for a halo glow effect
- Secure with zip ties and hot glue

Step 6 — Decoration

- Glue ThinkPad keyboard keys, RAM sticks, HDD platters around the base
- Add the ThinkPad vent grills with LED backlighting underneath
- Route all wires cleanly — use the ThinkPad's internal cable channels

---

🔌 Wiring Diagram

ESP32 GPIO Allocation

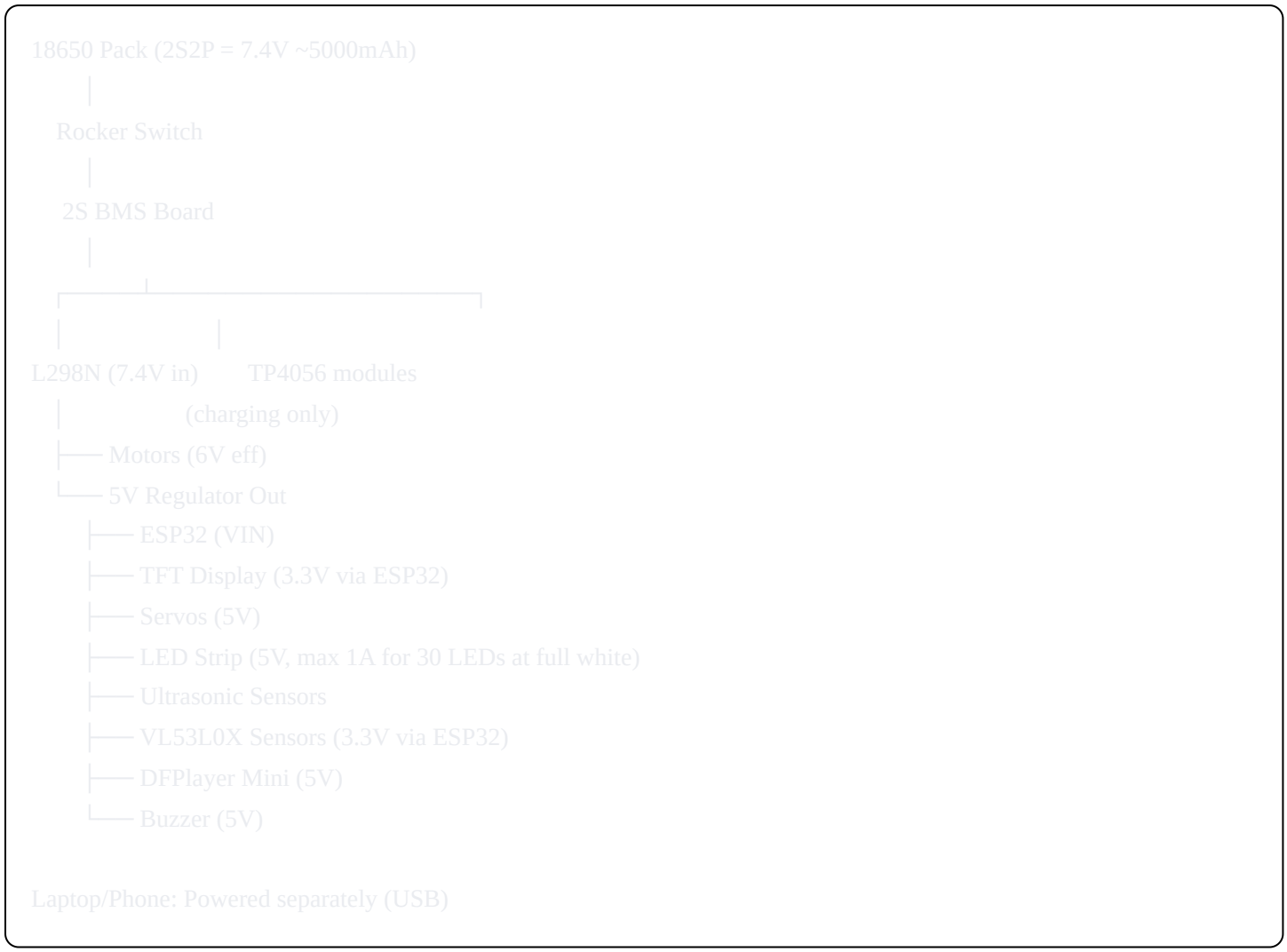
```
TFT DISPLAY (ILI9341):          MOTORS (via L298N):
├── GPIO 18 → SCK               ├── GPIO 25 → IN1
├── GPIO 23 → MOSI              ├── GPIO 26 → IN2
├── GPIO 15 → CS                ├── GPIO 27 → IN3
├── GPIO 2  → DC                ├── GPIO 14 → IN4
├── GPIO 4  → RST               ├── GPIO 32 → ENA (PWM)
└── 3.3V    → VCC + LED         └── GPIO 33 → ENB (PWM)

SERVOS (Pan-Tilt):              I2C BUS (ToF Sensors):
├── GPIO 19 → Pan Servo         ├── GPIO 21 → SDA
└── GPIO 22 → Tilt Servo        └── GPIO 22 → SCL
                                    (VL53L0X_1 XSHUT → GPIO 13)
LED STRIP (WS2812B):                (VL53L0X_2 XSHUT → GPIO 12)
└── GPIO 13 → DATA

ULTRASONIC SENSORS:             IR SENSORS:
├── GPIO 5  → US1 TRIG          ├── GPIO 36 → IR Left
├── GPIO 34 → US1 ECHO          └── GPIO 39 → IR Right
├── GPIO 0  → US2 TRIG
└── GPIO 35 → US2 ECHO

DFPLAYER MINI:                  BATTERY MONITOR:
├── GPIO 17 → TX                └── GPIO 34 → ADC (10k+10k divider)
└── GPIO 16 → RX

BUZZER:
└── GPIO 12 → Signal

NOTE: GPIO 21 and 22 are used for I2C. If you use 22 for tilt servo,
move servo to GPIO 11 or another free GPIO. I2C takes priority.
```

Voltage Divider for Battery Monitor

```
Battery (+) ──┬── 10kΩ ──┬── 10kΩ ── GND
              │          │
              │      GPIO 34 (ADC)
              │
        7.4V max → ADC reads 3.7V max (within ESP32's 3.3V range with offset)
```

> **Safety**: Use a 100kΩ + 100kΩ divider instead if your battery is 8.4V full — this gives 4.2V at pin which is over 3.3V. Better: use 68kΩ + 100kΩ to get readings safely within ADC range. Test with a multimeter first.

⚡ Power System



Estimated Runtime: 3–5 hours
Low Battery Alert: Triggered at <15% (ESP32 ADC monitoring)

💻 Software Installation

Prerequisites

- Python 3.10+
- Node.js 18+ (for web control frontend)
- Arduino IDE 2.x
- Mosquitto MQTT broker

Step 1 — Clone and Setup

bash

```
git clone <your-repo>
cd DESKBOT

# Create virtual environment
python -m venv venv
source venv/bin/activate  # Linux/Mac
# venv\Scripts\activate   # Windows

# Install Python dependencies
pip install -r requirements.txt
```

Step 2 — Install System Dependencies

bash

```
# Ubuntu/Debian
sudo apt install mosquitto mosquitto-clients espeak-ng cmake

# Arch/EndeavourOS
sudo pacman -S mosquitto espeak-ng cmake

# Start MQTT broker
sudo systemctl enable mosquitto
sudo systemctl start mosquitto
```

Step 3 — Configure

bash

```
# Copy example config and edit
cp server/dna_example.py server/dna.py
nano server/dna.py

# Set:
# LAPTOP_IP = "your laptop IP"
# PHONE_IP = "Redmi Note 12 static IP"
# TABLET_IP = "tablet IP"
# GEMINI_API_KEY = "from aistudio.google.com"
# ELEVENLABS_API_KEY = "optional, for ultra-realistic voice"
```

Step 4 — Download ML Models

bash

```
# Vosk offline STT
python scripts/download_models.py

# YOLOv5 (auto-downloads on first run)
python -c "from ultralytics import YOLO; YOLO('yolov5n.pt')"
```

Step 5 — Train Audio CNN (Optional but recommended)

```
bash

# Download UrbanSound8K dataset first:
# https://urbansounddataset.weebly.com/urbansound8k.html
# Place in data/urbansound8k/

python training/train_audio_cnn.py
```

Step 6 — Register Your Face

```
bash

# Place your photo in data/known_faces/yourname.jpg
# Or run the live registration:
python scripts/register_face.py --name "YourName" --label safe
```

🔧 ESP32 Firmware

Required Arduino Libraries

Install via Library Manager in Arduino IDE:

- `TFT_eSPI` by Bodmer
- `Adafruit NeoPixel`
- `PubSubClient` by Nick O'Leary
- `ArduinoJson` by Benoit Blanchon
- `ESP32Servo`
- `DFRobotDFPlayerMini`
- `VL53L0X` by Pololu

TFT_eSPI Configuration

Edit `User_Setup.h` in the TFT_eSPI library folder:

```
cpp
```

```
#define ILI9341_DRIVER
#define TFT_CS   15
#define TFT_DC    2
#define TFT_RST   4
#define TFT_MOSI 23
#define TFT_SCLK 18
#define TFT_MISO -1  // Not used
#define SPI_FREQUENCY  40000000
```

Flash Steps

1. Open `arduino/deskbot_esp32/deskbot_esp32.ino`
2. Edit `config.h` — set WiFi SSID, password, laptop IP
3. Select Board: `ESP32 Dev Module`
4. Upload Speed: `115200`
5. Click Upload
6. Open Serial Monitor at 115200 baud to verify connection

📱 Phone & Tablet Setup

Redmi Note 12 (Primary Sensor)

1. Install **DroidCam** from Play Store (better than IP Webcam)
2. Connect to same WiFi as laptop
3. Note the IP address shown in app
4. Set static IP in router settings for this device
5. Update `PHONE_IP` in `server/dna.py`
6. Optionally: keep phone plugged in via USB-C (it'll be running constantly)

UP Govt Tablet (Display)

1. Connect to same WiFi
2. Open Chrome/Firefox browser
3. Navigate to `http://LAPTOP_IP:8501` for Streamlit dashboard OR `http://LAPTOP_IP:5000` for the web control panel
4. Set browser to fullscreen (F11)
5. Optionally: install **Fully Kiosk Browser** for auto-launch

Web Control (Phone App)

1. On any phone on the same network
2. Navigate to `http://LAPTOP_IP:5000`
3. You get the full control panel: mode switching, camera feed, voice commands, face management

---

✅ First Boot Checklist

```
☐ MQTT broker running: sudo systemctl status mosquitto
☐ Redmi Note 12: DroidCam running, IP noted
☐ Tablet: Browser open at http://LAPTOP_IP:8501
☐ ESP32: Powered on, Serial Monitor showing "ONLINE"
☐ dna.py: All IPs and API keys set correctly
☐ known_faces/: At least one face image registered
☐ Battery: >50% before first test
☐ Python venv: Activated

# Run it:
python server/genesis.py

# Or with flags:
python server/genesis.py --no-audio    # skip audio CNN (faster startup)
python server/genesis.py --no-network  # skip network scanning
```

🔥 Troubleshooting

Problem	Fix
`face_recognition` install fails	`pip install cmake dlib face_recognition` — need cmake first
Camera feed not loading	Check DroidCam is running, verify PHONE_IP in dna.py
ESP32 not connecting	Verify WiFi credentials in config.h, check LAPTOP_IP
TFT display shows garbage	Check TFT_eSPI User_Setup.h pin mapping
Voice not working	Check mic permissions, try `python -m sounddevice`
MQTT errors	`sudo systemctl restart mosquitto`, check port 1883
Gemini errors	Verify API key, check internet connection
Servos jittering	Separate servo power supply from ESP32 5V

Problem	Fix
LEDs dim/wrong colors	Check WS2812B data wire, add 300Ω series resistor
Battery reading wrong	Calibrate voltage divider values in dna.py
Table edge not detected	Lower `DEPTH_DANGER_THRESHOLD` in dna.py

🗂️ Project Structure

```
DESKBOT/
├── server/
│   ├── genesis.py        # Main startup — launches everything
│   ├── dna.py            # All config, IPs, thresholds, API keys
│   ├── blackbox.py       # SQLite logging
│   ├── psyche.py         # Personality, joke prompts, roast templates
│   ├── optic.py          # Vision: face detection/recognition, pose, mesh
│   ├── vocoder.py        # Voice: STT, TTS (edge-TTS), Gemini, commands
│   ├── echo_hunter.py    # Audio classification CNN
│   ├── ice_wall.py       # Network monitoring
│   ├── synapse.py        # MQTT hub
│   ├── hivemind.py       # Sensor fusion
│   ├── agent.py          # AI agent: code review, document Q&A, RAG
│   └── home_auto.py      # Home automation (smart lights, etc.)
├── dashboard/
│   └── nexus.py          # Streamlit cyberpunk dashboard
├── web_control/
│   ├── app.py            # Flask web control server
│   ├── templates/
│   │   └── index.html    # Phone/web control UI
│   └── static/           # CSS/JS
├── arduino/
│   └── deskbot_esp32/
│       ├── deskbot_esp32.ino
│       ├── config.h
│       ├── eyes.h
│       ├── motors.h
│       ├── leds.h
│       ├── sensors.h
│       └── servos.h
├── training/
│   ├── train_audio_cnn.py
│   └── evaluate_models.py
├── scripts/
│   ├── download_models.py
```

```
│   └── register_face.py
├── models/
│   └── (auto-generated model files)
├── data/
│   ├── known_faces/       # Add face JPGs here
│   ├── documents/         # Upload PDFs/docs for agent to read
│   └── alerts/            # Auto-saved alert screenshots
├── requirements.txt
├── BUILD_GUIDE.md
└── README.md
```

Built with ₹4,550 worth of components (original ₹3,750 + ~₹800 for ToF sensors and misc upgrades). The intelligence is free.