**PowerMind AI – Technical Architecture Breakdown**

**🔹 1. Monitoring (Sensing + Data Collection)**

What it does: Collects raw data from electricity flow in real time.

**✅ Key Concepts:**

* **Current sensors** (ACS712): detect amperage usage
* **Voltage dividers**: measure voltage if needed (optional for AC safety)
* **MCP3008 ADC**: converts analog sensor data into digital for Raspberry Pi
* **Sampling frequency**: How often data is collected (e.g. every second)
* **CSV + Cloud Logging**: store every measurement with a timestamp

**🧠 What AI learns from here:**

* Patterns of usage over time
* Abnormal spikes
* Idle consumption
* Appliance-level signatures (later phase)

**🔹 2. Optimization (Data Analysis + Energy Efficiency AI)**

What it does: Analyzes past and real-time data to make smarter decisions.

**✅ Key Concepts:**

* **Python-based logic or ML models** for rule-based decisions
* **Time-of-day usage prediction** (e.g., when peak load occurs)
* **Energy efficiency scoring**: room/device/wattage prioritization
* **Reinforcement Learning** (future): AI learns from reward/punishment (e.g., minimize wattage without loss of comfort)

**Example:**

“At 6pm daily, the water heater and fridge overlap usage — delay heater by 20 min to avoid peak surge”

**🔹 3. Control (Relay Switching + Automation)**

What it does: Actually turns devices on/off based on smart decisions.

**✅ Key Concepts:**

* **GPIO control of relays** (Python: GPIO.output(pin, HIGH/LOW))
* **Emergency cut-off** for high-load conditions
* **Priority list**: essential vs. optional devices
* **Manual override**: user can always switch via web/mobile

**Safety Layer:**

* Use **optocouplers** or isolation circuits for relay safety
* Add **thermal shutdown logic** if devices overheat

**🔹 4. Prediction & Resistance Management (AI Models)**

What it does: Predicts resistance drops, power loss, or overconsumption trends.

**✅ Key Concepts:**

* **Ohm’s Law (V = IR)** used to estimate resistance
* **Power loss detection** (resistive heat = I²R)
* **Anomaly detection models** (unsupervised ML or LSTM)
* AI flags:
  + Corroded wires (increased resistance)
  + Unexpected drops
  + Ghost current draw

**📊 Sample Architecture Flow**

Sensors (ACS712 → MCP3008)

↓

Raspberry Pi (Data logger + Real-time processing)

↓

Flask API + Web Dashboard

↓

Control Decisions (Auto-switch via Relay)

↓

Cloud Sync + AI Learning

**🔹 1. Monitoring Layer – The Nervous System**

**Purpose:** Capture all electrical activity (current, voltage, resistance, flow) from the system in real time.

**🚧 Components:**

* **ACS712/ACS758** → measures **current flow**
* **Voltage sensors** (optional) for AC/DC voltage
* **MCP3008** ADC → converts analog signal to digital
* **Raspberry Pi GPIO/ADC interface**

**🧠 Output:**

* Time-series data of **amperage**, optionally **voltage**, and calculated **power**
* Data logged locally to .csv or streamed to dashboard
* Raw input for future AI analysis
* **🔹 2. Optimization Layer – The Brain**

**Purpose:** Analyze electricity usage patterns and make smart efficiency decisions.

**🚧 Technologies:**

* Python logic + **basic heuristics** (rules: “if >10A, cut load”)
* Optionally: **AI models (MLP, Decision Trees, or Edge Impulse AI)**

**🧠 Key Functions:**

* Detect high-consumption periods
* Predict peak usage based on time-of-day trends
* Recommend or auto-optimize:
  + When to delay water heater
  + When solar is strong → prioritize heavy loads
  + Turn off idle devices intelligently

**🔹 3. Control Layer – The Hands**

**Purpose:** Execute decisions by turning devices on/off, rerouting flow, or isolating fault circuits.

**🚧 Components:**

* Relay modules (1–8 channel, 5V or 12V) (up to which channel do we need?)
* Raspberry Pi controls GPIO to activate relays
* Power routing through **smart sockets or contactors (do we need to add this smart sockets or contactors)**

**🧠 Key Features:**

* **Auto-cut on overload**
* Manual override via dashboard/mobile
* Priority control:
  + Essentials (lights/fridge) always on
  + Non-essentials (AC/charger) switched off if needed

**🔹 4. Intelligence Layer – The Soul**

**Purpose:** Long-term learning and prediction based on all historical usage.

**🚧 Tools:**

* AI platforms: **TensorFlow Lite**, **Edge Impulse**, or custom ML in Python (which one we should use)
* Data: Stored in cloud or local DB (MongoDB/SQLite) (which one we should use)
* Algorithms:
  + **Anomaly detection**
  + **Reinforcement learning**
  + **Time-series prediction (LSTM)**

**🧠 Learning Goals:**

* Predict next 24h usage
* Identify energy leaks or abnormal usage
* Recommend load shifting (e.g., night-time laundry)
* Build a smart profile for each household

**🧩 How All Layers Connect (Simple Data Flow)**

mathematica

Copy

Sensors → Pi + Logic → Dashboard/API → AI Model (Cloud or Edge)

↓ ↓ ↓ ↓

Real-time Decision Control Long-term Prediction

* **🔍 Step 1: Overview of the Monitoring Layer**

**🎯 Goal:**

* Capture real-time data on:
  + **Current (Amps)**
  + **Voltage (Volts)**
  + **Power (Watts)** → calculated from Voltage × Current

**📦 Main Hardware:**

| **Sensor** | **Measures** | **Output Type** |
| --- | --- | --- |
| **ACS712 / ACS758** | Current | Analog Voltage (0–5V) |
| **0–25V Voltage Module** | Voltage (DC) | Analog Voltage (scaled down) |
| **MCP3008** | ADC (Analog to Digital) | SPI to Raspberry Pi |

**⚙️ Step 2: Wiring Plan**

**🔌 Current Sensor (ACS712 or ACS758):**

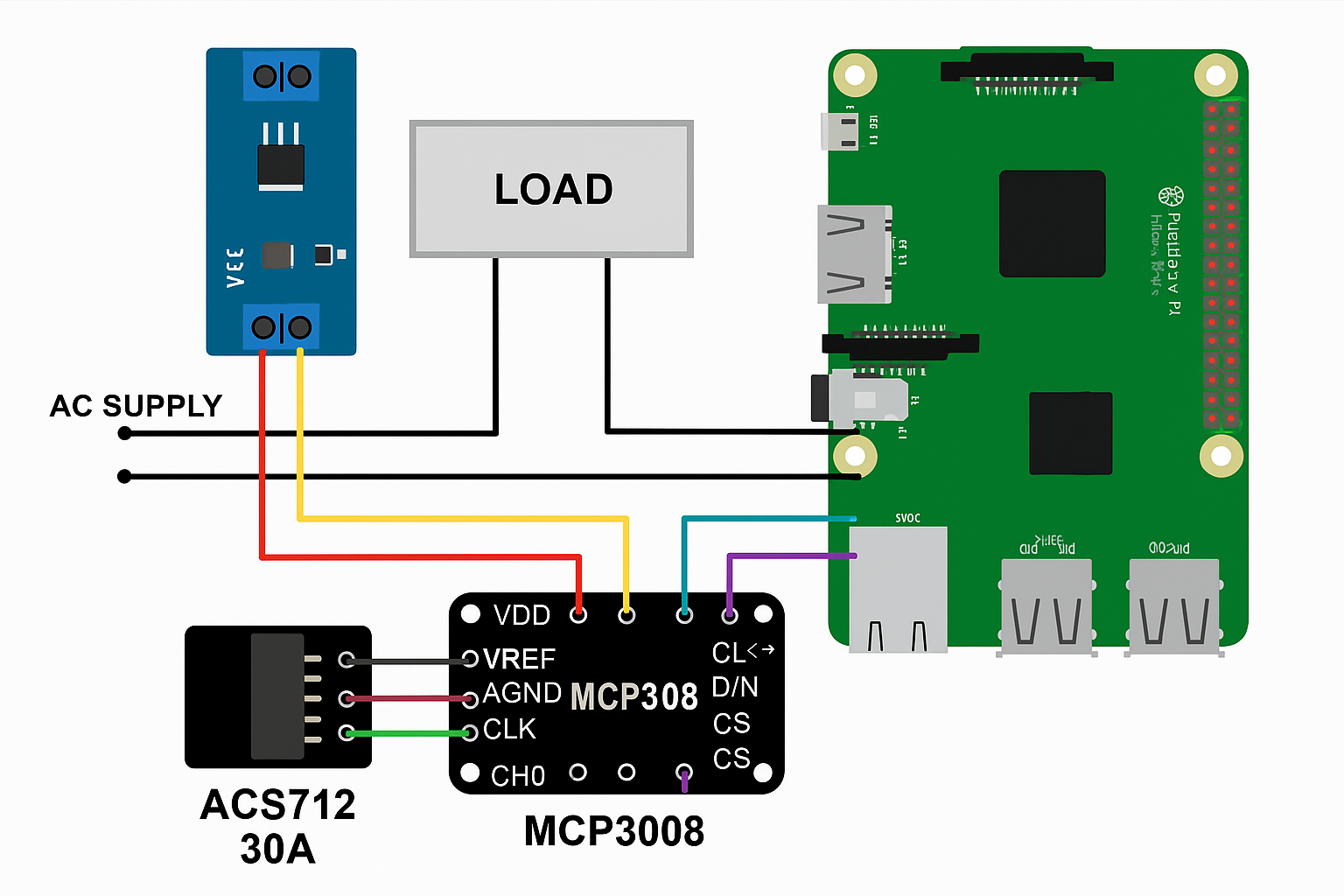
* **VCC** → 5V on Raspberry Pi
* **GND** → Ground on Raspberry Pi
* **OUT** → MCP3008 Channel 0 (CH0)

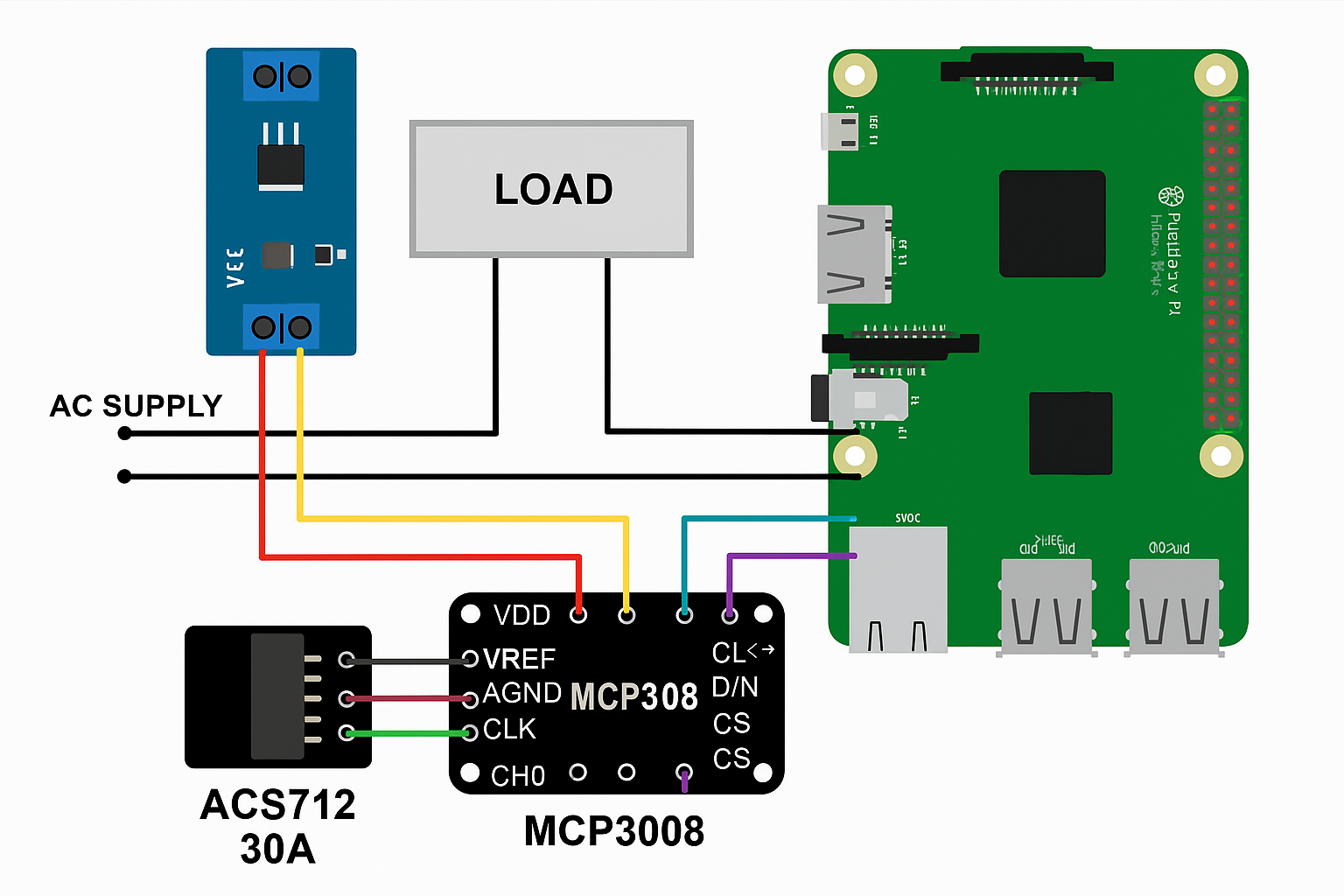
**🔌 Voltage Sensor (0–25V Module):**

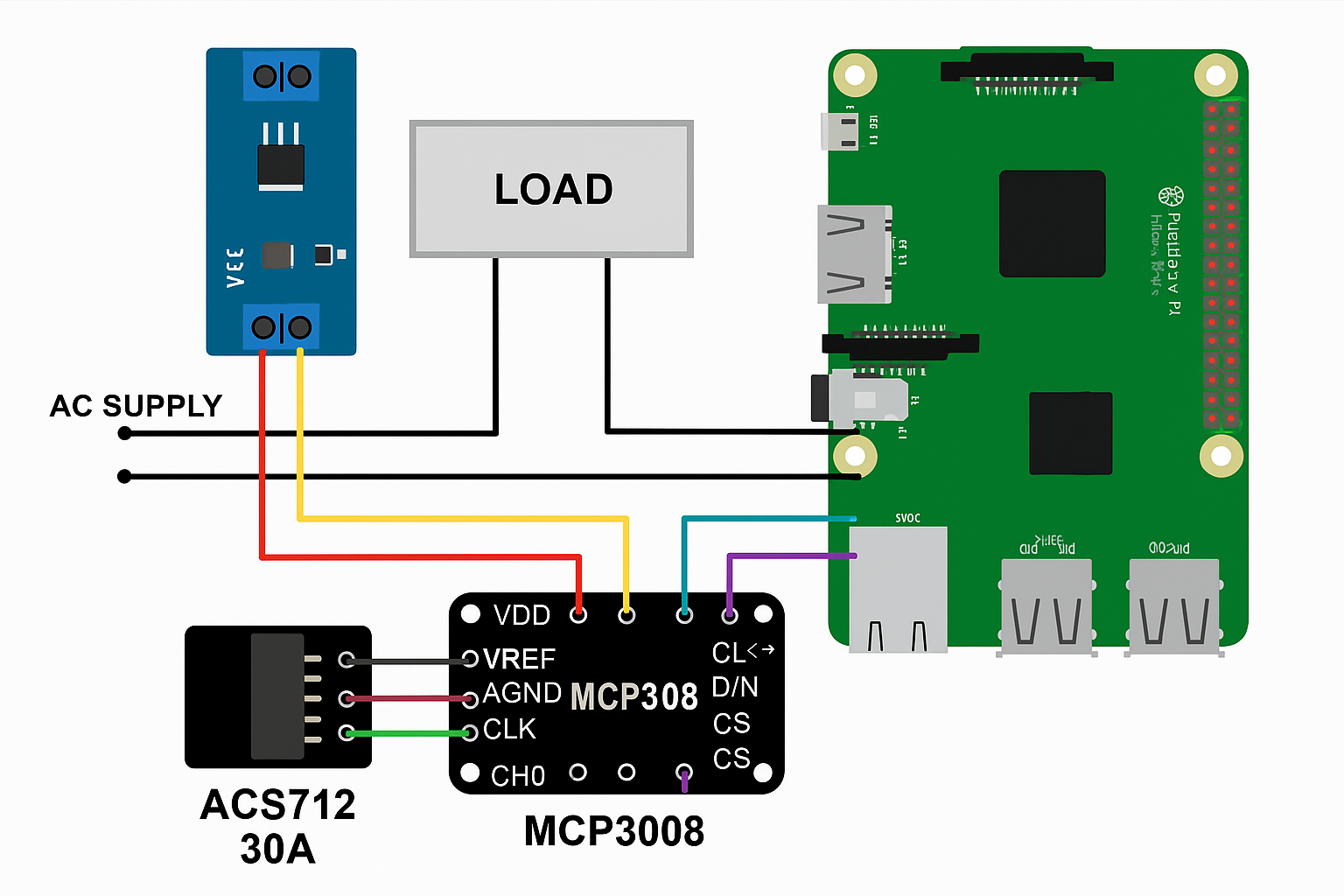
* **VCC** → 5V on Raspberry Pi
* **GND** → Ground
* **OUT** → MCP3008 Channel 1 (CH1)

**🔌 MCP3008 ADC:**

* **CH0** → from ACS712 OUT
* **CH1** → from Voltage Sensor OUT
* **VDD, VREF** → 3.3V
* **AGND, DGND** → GND
* **CLK, DOUT, DIN, CS** → GPIO pins (we’ll wire them in next step)







**🔧 How It Works:**

* Reads voltage from **CH1** (0–25V sensor)
* Reads current from **CH0** (ACS712 30A)
* Calculates power (Watts)
* Prints live status in terminal
* Logs data to power\_log.csv

Top of Form

Bottom of Form