ESP32 Current Monitoring System

Non-invasive AC current monitoring using ESP32 and SCT013 current transformers, with MQTT data publishing.

Project Overview

This system monitors AC current on electrical circuits using clamp-on current transformers (CTs) and publishes readings via MQTT every 5 seconds. Designed for marine/boat environments with plans for waterproof enclosure.

Key Features:

- Non-invasive measurement (no wire cutting required)
- Real-time current monitoring with RMS calculation
- MQTT publishing for integration with home automation
- NTP time synchronization
- Automatic WiFi configuration portal
- Support for multiple CT sensors (expandable to 4+ circuits)

Hardware Components

Required Parts

- **ESP32 Development Board** (ESP32-WROOM-32 or similar)
- **SCT013-050 CT Sensor** (50A/1V voltage output model) - or SCT013-000V (100A/1V)
- **Resistors:** 2x 10kΩ (±5% tolerance acceptable)
- **Capacitor:** 1x 10μF electrolytic capacitor
- **3.5mm Audio Jack:** PCB mount (optional but recommended - e.g., Tegg PJ-307)
- **Breadboard** (for prototyping)
- **Perfboard** (5x7cm or similar, for permanent build)
- **Waterproof Enclosure** (IP65+ rated with cable glands)
- **USB Cable** (for power and programming)

Optional Components

- **Additional CT Sensors** (for monitoring multiple circuits)
- **Desiccant Pack** (for moisture control in enclosure)
- **Conformal Coating** (for PCB protection in marine environment)

Circuit Design

Schematic

```
ESP32 3.3V ───┬─── 10kΩ R1 ───┬─── ESP32 GPIO34 (ADC)
          │         │
                  ├─── 10μF Cap (+) ───┬─── CT White Wire (Signal)
                  │              │
                  └─── 10kΩ R2 ───────┬──────┴─── CT Red Wire (Ground)
                            │
ESP32 GND ─────────────────────────────────────────────────────────────┘
```

How It Works

1. **Voltage Divider (R1 + R2):** Creates 1.65V DC bias at the ADC input (midpoint between 0-3.3V)
2. **AC Coupling Capacitor:** Blocks DC, passes only the AC signal from the CT sensor
3. **CT Sensor:** Outputs 0-1V AC proportional to measured current (50A = 1V for SCT013-050)
4. **ESP32 ADC:** Samples the AC waveform at ~24kHz and calculates RMS current

Why This Circuit?

The SCT013-050 is a **voltage output** CT with an internal burden resistor. It cannot be connected directly in parallel with a voltage divider (this pulls the circuit to the rails). Instead, the AC signal must be coupled through a capacitor while maintaining a separate DC bias at the ADC pin.

Critical Design Notes:

- The CT sensor wires are interchangeable (red/white can be swapped)
- The shield wire (if present) is NOT needed and can be left disconnected
- GPIO34 is recommended for ADC (ADC1 pins: 32, 33, 34, 35, 36, 39 are all suitable)

Pin Connections

Component	ESP32 Pin	Breadboard Row
Power	3.3V	Row 10
Ground	GND	Row 25
ADC Input	GPIO34	Row 15
R1 (10kΩ)	3.3V → GPIO34	Row 10-15
R2 (10kΩ)	GPIO34 → GND	Row 15-25
Capacitor (+)	GPIO34 side	Row 15
Capacitor (-)	CT signal side	Row 20
CT Red Wire	GND	Row 25
CT White Wire	Through capacitor	Row 20

Software Setup

Arduino IDE Configuration

1. **Install ESP32 Board Support:**
    - File → Preferences → Additional Board Manager URLs
    - Add: `https://dl.espressif.com/dl/package_esp32_index.json`
    - Tools → Board → Boards Manager → Search "ESP32" → Install
2. **Install Required Libraries:**
    - WiFiManager (by tzapu) - v2.0.17
    - PubSubClient (by Nick O'Leary) - v2.8
    - ESPmDNS (included with ESP32 board package)
    - Preferences (included with ESP32 board package)
3. **Board Settings:**
    - Board: "ESP32 Dev Module"
    - Upload Speed: 921600
    - CPU Frequency: 240MHz
    - Flash Frequency: 80MHz
    - Flash Mode: QIO
    - Flash Size: 4MB

    - Partition Scheme: Default
    - Port: (select your USB serial port)

Initial Configuration

On first boot, the ESP32 will create a WiFi access point:

- **SSID:** `ESP32-CurrentMonitor`
- **Connect to it** with your phone/computer
- **Web portal** will open automatically (or navigate to 192.168.4.1)
- **Enter:**
    - Your WiFi network credentials
    - MQTT broker IP address (e.g., 10.0.0.142)
- **Save** - ESP32 will reboot and connect

Calibration

The `CT_RATIO` constant in the code must be calibrated for accurate readings:

1. **Connect a known load** (use a Kill-A-Watt meter or clamp meter to measure actual current)
2. **Compare ESP32 reading to actual current**
3. **Calculate calibration factor:**

```
Calibration Factor = ESP32_Reading ÷ Actual_Current
New_CT_RATIO = Current_CT_RATIO ÷ Calibration_Factor
```

Example:

- Kill-A-Watt shows: 8.24A
- ESP32 shows: 23.4A
- Calibration Factor: 23.4 ÷ 8.24 = 2.84
- New CT_RATIO: 50 ÷ 2.84 = 17.6

Update the code:

```
cpp
```

```
#define CT_RATIO 17.6  // Calibrated value
```

MQTT Topics

Data is published to:

```
gooddecisions/{MAC_ADDRESS}/current    - Current in Amps (float, 2 decimals)
gooddecisions/{MAC_ADDRESS}/timestamp  - ISO 8601 timestamp or BOOT+seconds
gooddecisions/{MAC_ADDRESS}/version    - Firmware version (retained)
```

Example:

```
gooddecisions/441D64F83968/current → "8.24"
gooddecisions/441D64F83968/timestamp → "2025-10-23T00:46:32Z"
gooddecisions/441D64F83968/version → "v1.0 Oct 23 2025 17:15:42"
```

Operation

Normal Operation

- **LED Behavior:**
  - Boot: Solid ON
  - Normal: Quick blink every 30 seconds (heartbeat)
  - Connection issues: Fast blink (2Hz) or slow blink (every 30s)
- **Serial Output:**
  - Baud rate: 115200
  - Debug information printed every 5 seconds:

```
  RAW ADC: avg=1835 min=1500 max=2160
  Avg Voltage: 1.48V
  Samples: 2000, Rate: 24000 Hz, RMS Voltage: 0.234V, Current: 8.24A
  Published - Current: 8.24A at 2025-10-23T00:46:32Z
```

Measurement Accuracy

- **Optimal Range:** 5-50A (for 50A CT) - accuracy within ±5%
- **Acceptable Range:** 3-50A - reasonable accuracy

- **Poor Accuracy:** Below 3A - noise dominates, readings unreliable
- **Maximum:** Do not exceed CT rating (50A for SCT013-050)

Why low currents are inaccurate: CT sensors are optimized for their rated range. At currents below 10% of rating (~5A for a 50A CT), the signal is too small compared to the noise floor (~17A equivalent ADC noise).

Reset and Reconfiguration

To reset WiFi and MQTT settings:

- Hold the **BOOT button** for 3+ seconds during power-up
- ESP32 will clear all settings and restart in configuration mode

Automatic reset conditions:

- 3 consecutive boot failures → Opens configuration portal
- Persistent MQTT connection failures → Sends notification via AWS heartbeat

Troubleshooting

Problem: Readings stuck at 146.54A or 4095 ADC

Cause: CT sensor connected incorrectly (in parallel with voltage divider instead of through capacitor)

Solution: Verify circuit matches schematic - CT must connect through the capacitor, not directly to the voltage divider

Problem: Readings show ~17A with no load

Cause: Normal ADC noise floor - this is expected

Solution: No action needed. This noise disappears when measuring actual loads >5A

Problem: Brownout detector triggered / constant reboots

Cause: Insufficient USB power supply

Solution:

- Use a higher-current USB power adapter (2A minimum)
- Use a powered USB hub
- Check USB cable quality

Problem: WiFi won't connect

Cause: Incorrect credentials or out of range

Solution:

- Reset settings (hold BOOT button 3 seconds on power-up)
- Move closer to WiFi router during setup
- Check WiFi password is correct

Problem: MQTT connection fails

Cause:

- Mosquitto broker not running
- Broker not configured to listen on network (only localhost)

Solution:

```bash
# Add to /opt/homebrew/etc/mosquitto/mosquitto.conf:
listener 1883 0.0.0.0
allow_anonymous true

# Restart broker
brew services restart mosquitto
```

Marine/Boat Installation

Enclosure Requirements

- **Rating:** IP65 minimum (waterproof when closed)
- **Material:** Nylon or ABS plastic (corrosion resistant)
- **Size:** Large enough for ESP32 + perfboard + strain relief
- **Cable Glands:** PG7 or PG9 size for CT sensor cables

Environmental Protection

1. **Conformal Coating:** Apply to perfboard after soldering (protects from salt air/moisture)
2. **Desiccant Pack:** Place inside enclosure to absorb moisture

3. **Breather Vent:** Consider Gore-Tex membrane vent for pressure equalization
4. **Cable Routing:** Use proper marine-grade cable glands with O-rings

Installation Location

- **Dry location preferred** - avoid bilge or areas prone to water
- **Accessible** - for troubleshooting and CT sensor adjustment
- **Near power source** - USB power adapter or 12V→5V converter
- **WiFi coverage** - test signal strength before permanent installation

Mounting

- Use stainless steel screws/fasteners
- Vibration: Secure perfboard with standoffs, use strain relief on cables
- Heat: Ensure some ventilation, avoid direct sunlight

Expanding the System

Adding More CT Sensors

The ESP32 has multiple ADC pins available:

Available ADC1 Pins: GPIO32, GPIO33, GPIO35, GPIO36, GPIO39

For each additional CT:

1. **Replicate the circuit** (voltage divider + capacitor + CT)
2. **Use a different GPIO pin**
3. **Update code:**

```cpp
#define CT_PIN_1 34
#define CT_PIN_2 35
#define CT_PIN_3 36
// etc.
```

4. **Publish to different MQTT topics:**

```
gooddecisions/{MAC}/current1
gooddecisions/{MAC}/current2
gooddecisions/{MAC}/current3
```

Power Consumption

- **Active (WiFi connected, sampling):** ~80-120mA @ 5V
- **Sleep mode possible:** Could reduce to <10mA with deep sleep between readings
- **For 24/7 operation:** ~1-2W continuous power draw

Technical Specifications

Parameter	**Value**
Supply Voltage	5V USB (3.3V regulated internally)
ADC Resolution	12-bit (0-4095)
ADC Input Range	0-3.3V
Sample Rate	~24,000 samples/second
Samples per Reading	2000 (allows accurate RMS calculation)
Publish Interval	5 seconds
CT Sensor Range	0-50A AC (or 0-100A for SCT013-000V)
Measurement Accuracy	±5% @ 5-50A range
WiFi	802.11 b/g/n 2.4GHz
MQTT	QoS 0, unencrypted

Safety Warnings

⚠️ **ELECTRICAL SAFETY:**

- CT sensors provide galvanic isolation - the high voltage wire never touches the ESP32
- Always turn off circuit power before clamping CT sensor
- Do not exceed CT sensor's rated current (50A or 100A depending on model)
- Verify proper polarity on electrolytic capacitor (incorrect can cause failure/heat)

⚠️ **MARINE ENVIRONMENT:**

- Use appropriate IP-rated enclosure
- Follow marine electrical standards (ABYC if in USA)

- Consider circuit protection (fuse/breaker on power supply)
- Regular inspection for corrosion

Future Enhancements

□ Deep sleep mode for battery-powered operation
□ Local data logging to SD card
□ Web dashboard (serve from ESP32)
□ OTA (Over-The-Air) firmware updates
□ Multiple CT sensors (parallel monitoring)
□ Power factor measurement (requires voltage sensing)
□ Energy integration (kWh calculation)
□ Configurable publish intervals
□ MQTT authentication (TLS/SSL)

License

Credits

Hardware: ESP32 by Espressif, SCT013 by YHDC
Software: Arduino ESP32 Core, WiFiManager, PubSubClient, EmonLib concepts

Built with assistance from Claude (Anthropic) for circuit design, troubleshooting, and documentation.

Version: 1.0
Last Updated: October 23, 2025
Author: Robbie Cape