DH-Systems Main Module

- **Product Name:** DH-Systems Main Module
- **Document Version:** v2.0 (2026)
- **SKU / Product ID:** DHS-MOD-MAIN02
- **Target Audience:** DIY Builders / System Integrators

Important Safety Notice & Disclaimer

⚠ **READ BEFORE INSTALLATION**

- **DIY Product Notice:** This module is sold as a Do-It-Yourself (DIY) component for custom electronics integration. It is not a certified commercial turn-key product. Final system housing, fuse protection, and overall electrical safety are the sole responsibility of the builder.
- **Risk of Damage:** Incorrect wiring, reversed polarity, or short circuits can cause immediate and permanent damage to the module, connected sensors, and the Master Unit. Always double-check all connections with a multimeter before powering up the system for the first time.
- **Qualified Installation:** Electrical systems in vehicles and off-grid environments carry inherent risks of short circuits, fire, or battery damage. It is strongly recommended that the installation, wiring, and integration of these modules are carried out by a qualified individual with a solid understanding of DC electronics.
- **Liability:** DH-Systems cannot be held liable for any hardware damage, data loss, physical injury, or vehicular damage resulting from the incorrect assembly, programming, configuration, or installation of this equipment.
- This product is an **open-component DIY Development Board (sub-assembly)**, not a finished consumer electronics appliance. It is designed exclusively for system integration by DIY builders and qualified individuals.

1. Introduction & Core Function

The **DH-Systems Main Module** forms the central nervous system of your smart camper, acting as the primary gateway that processes and orchestrates all data toward Home Assistant. Designed to resolve the limitations and high costs of closed commercial setups, it offers an industrial-grade, highly reliable ecosystem for off-grid living.

The Main Module coordinates a hybrid network topology: a local I2C bus for extensions inside or directly adjacent to the central control cabinet, and a robust Modbus RS485 network for remote modules distributed further away across the vehicle.

Key Features:

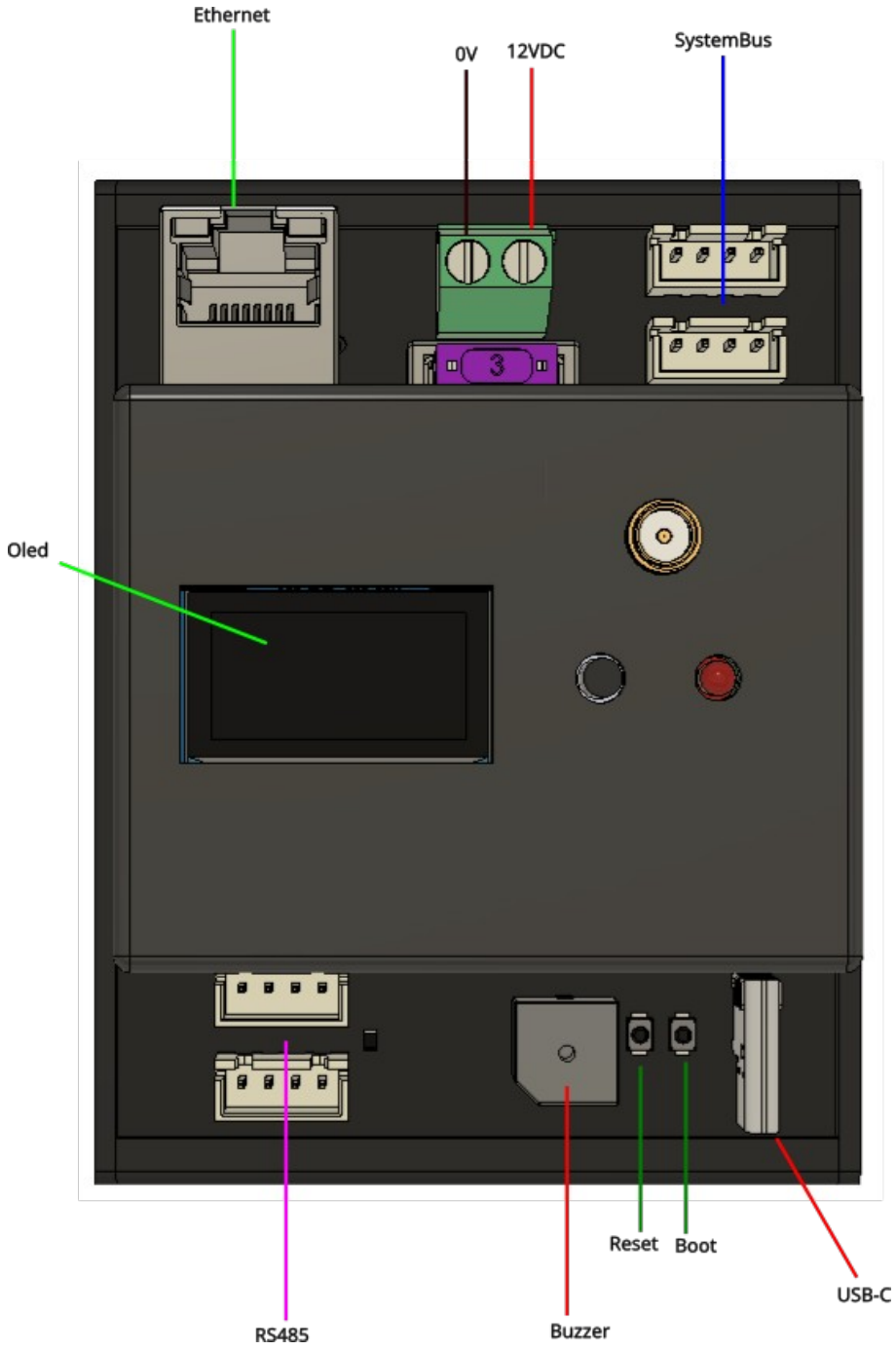
- **High-Performance Core:** Powered by the advanced ESP32-C6-WROOM-1U-N8 microcontroller.
- **Dual Connectivity:** Equipped with a hardware-wired W5500 Ethernet port for rock-solid connection stability, alongside integrated Wi-Fi.
- **Automotive Power Stage:** High-efficiency AP63203 buck regulator converting noisy 12V vehicle power to a ultra-stable 3.3V control rail.
- **Onboard Power Monitoring:** Integrated Texas Instruments INA226 monitoring chip with a 0.025 Ohm onboard shunt to accurately measure the total power draw of the system right at the source.
- **Local Interactive Hub:** Bundled with a Top Board featuring an SSD1306 OLED display for real-time status readings and an interactive hardware button.

- **Bus Type:** Hybrid Gateway — Local SystemBus (I2C) + Remote Network (Modbus / RS485 via twist-pair lines).

2. Technical Specifications

Parameter	Specification
Main Controller	Espressif ESP32-C6-WROOM-1U-N8 (8MB Flash, Wi-Fi 6, Thread/Zigbee, Bluetooth 5)
Ethernet Controller	W5500 (Hardwired SPI Interface)
Operating Voltage	12V DC Nominal (Designed to safely handle automotive/camper battery charging voltages)
Hardware Protection	Onboard input fuse, SS34 reverse polarity protection diode, and PTVS18V Transient Voltage Suppression (TVS) diode
Local Display	SSD1306 OLED Display (I2C Interface) with automated hardware/software blanking
Expansion Buses	1x Local I2C Protected Bus, 1x Modbus RS485 Transceiver Line

3. Pinout & Wiring Diagram



Hardware Interfaces & Connections:

SystemBus Input / Output (2x 4-Pin JST Connector):
These two parallel ports link the Main Module to the local expansion boards (such as the Relay or Input modules). They carry the primary power and local I2C communication lines:

- 12V — Main power input/output from the fused, TVS-protected system rail.
- GND — Common System Ground.
- SDA — I2C Data Line.

- SCL — I2C Clock Line.

Remote Network / RS485 (2x 4-Pin JST Connector):

These two ports handle the long-distance Modbus RS485 network communication across the vehicle. They enable daisy-chaining remote sensor components:

- 12V — Fused power feed for remote modules.
- GND — Ground link for signal reference and power.
- A / D+ — RS485 Differential Non-Inverting Data Line.
- B / D- — RS485 Differential Inverting Data Line.

Flashing & Diagnostics Ports:

- **1x USB-C Interface:** Located on the side of the board for plug-and-play firmware flashing, serial debugging, and initial ESPHome configuration directly from a laptop or PC.
- **TX-RX Solder Pads:** Dedicated hardware pads on the PCB surface providing direct access to the ESP32-C6 hardware UART serial lines for advanced debugging, diagnostics, or recovery flashing if required.

4. Hardware Configuration & Solder Jumpers

I2C Address Map

The Main Module acts as the central I2C Master Engine. To ensure reliable data routing without bus conflicts, the fixed onboard components are pre-configured to the following specific hardware addresses:

- **Onboard Power Monitor (Mainboard):**
  The integrated INA226 power monitoring chip has its address pins A0 and A1 hardwired directly to Ground (GND). This hardware configuration sets the chip to a fixed I2C address of 0x40.
- **0.96" OLED Display (Top Board):**
  The interactive 0.96-inch screen embedded on the companion Top Board is pre-configured to use the standard I2C address 0x3C.

I2C Bus Management & Pull-Up Configuration

For a clean digital square wave signal, the local I2C SystemBus relies on hardware pull-up resistors. Within the DH-Systems ecosystem, the following architecture applies:

- **The Master Rule:**
  Every individual module in our line-up is provisioned with onboard 4K7 pull-up resistors (R3-R4). However, having too many parallel pull-up resistors on a single bus drops the total resistance too low, over-saturating the lines and blocking communication.

- **Main Module Configuration:**
  The 4K7 pull-up resistors must remain fully enabled on this Main Module, as it dictates the central bus line timing.
- **Expansion Modules Policy:**
  On all intermediate expansion modules (Relay, Input, Dimmer, etc.) connected to the SystemBus, these pull-ups are designed to be bypassed or removed. They are only required on the Main Module itself and, in specific large installations with long wire runs, potentially on the very last module at the far end of the physical I2C chain to preserve signal integrity.

Hub Management & Signals

- **Network Master:**
  Because this module dictates the bus timing for all local extensions and processes the distributed Modbus loop, no external hardware address jumpers need to be modified by the user on the Main Module itself.
- **Top Board Control (OLED & Interaction):**
  - The physical push button on GPIO10 handles local interaction or diagnostic loops. It features an integrated software debounce filter to eliminate accidental double-triggers caused by mechanical bounce.
  - To prevent annoying light pollution during sleep cycles in the camper, the SSD1306 display (0x3C) includes a dedicated hardware-level power-down mechanism linked with an automated software timer in ESPHome.
- **Acoustic Signaling:**
  The onboard alarm buzzer is routed specifically through GPIO21 using an MMBT2222A driver transistor. This precise pin routing eliminates loud, unwanted hardware clicking or boot-tones during firmware flashing cycles.

5. Software Integration & ESPHome Configuration

The system architecture utilizes a highly clean, modular **multi-file package structure**. The main boot configuration, Wi-Fi parameters, and structural I2C bus options are defined in the master configuration file (main-control-module.yaml), while the functional sensor hooks are handled by appending the specific sub-module configuration (main-module-sensors.yaml).

```
esphome:
  name: main-control-module
  friendly_name: Main_Control_Module


esp32:
  board: esp32-c6-devkitc-1
```

```
  framework:
    type: esp-idf

logger:

api:
  encryption:
    key:

ota:
  - platform: esphome
    password:

wifi:
  ssid: !secret wifi_ssid
  password: !secret wifi_password
  ap:
    ssid: "Main-Control-Module"
    password:
  id: wifi_module

captive_portal:

font:
  - file: "gfonts://Roboto"
    id: font_main
    size: 14
  - file: "gfonts://Roboto"
    id: font_small
    size: 10

i2c:
  - id: bus_a
    sda: GPIO6
    scl: GPIO7
    scan: true

packages:
```

```
  - !include main-module-sensors.yaml
  - !include relais_module.yaml
  - !include input_module.yaml
  - !include dimmer_module.yaml
  - !include solar.yaml
```

Functional Sensors Package: main-module-sensors.yaml

```
esphome:
  on_boot:
    priority: -100 # Waits until displays and sensors are fully initialized
    then:
      - lambda: |-
          id(display_aan) = false;
          id(my_display).turn_off();

output:
  - platform: ledc
    pin: GPIO16
    id: buzzer_output
    inverted: false # Ensures 0 equals absolute off

  - platform: ledc
    pin: GPIO11
    id: rode_led

light:
  - platform: monochromatic
    name: "Master Unit Status LED"
    id: master_unit_status_led
    output: rode_led
    restore_mode: ALWAYS_OFF

globals:
  - id: display_aan
    type: bool
    restore_value: no
```

```
    initial_value: 'false'

switch:
  - platform: template
    name: "Master Unit Display Power"
    id: master_unit_display_switch
    icon: "mdi:monitor"
    lambda: |-
      return id(display_aan);
    turn_on_action:
      - lambda: |-
          id(display_aan) = true;
          id(my_display).turn_on();
    turn_off_action:
      - lambda: |-
          id(display_aan) = false;
          id(my_display).turn_off();

  - platform: output
    name: "Master Unit Buzzer"
    id: master_unit_buzzer
    output: buzzer_output
    icon: "mdi:volume-high"

button:
  - platform: restart
    name: "Master Unit Reboot"

binary_sensor:
  - platform: gpio
    pin:
      number: GPIO10
      inverted: true
      mode: INPUT_PULLUP
    id: mijn_knop
    filters:
      - delayed_on: 50ms
    on_press:
```

```
    then:
      - lambda: |-
          if (id(display_aan)) {
            id(display_aan) = false;
            id(my_display).turn_off();
          } else {
            id(display_aan) = true;
            id(my_display).turn_on();
          }

sensor:
  - platform: ina226
    address: 0x40
    shunt_resistance: 0.025 ohm
    max_current: 3.0A
    current:
      name: "SystemBus Current"
      id: "module_stroom"
      unit_of_measurement: "A"
      accuracy_decimals: 3
    bus_voltage:
      name: "SystemBus Voltage"
      id: "module_spanning"
      unit_of_measurement: "V"
      accuracy_decimals: 2
      filters:
        - offset: 0.27
    power:
      name: "SystemBus Power"
      unit_of_measurement: "W"
      accuracy_decimals: 2
    update_interval: 5s

  - platform: wifi_signal
    name: "Master Unit Wifi Signal"
    update_interval: 60s

text_sensor:
```

```
  - platform: wifi_info
    ip_address:
      name: "Master Unit IP Address"

display:
  - platform: ssd1306_i2c
    model: "SSD1306 128x64"
    id: my_display
    address: 0x3C
    update_interval: 1s
    lambda: |-
      if (!id(display_aan)) {
        return;
      }

      if (id(wifi_module).is_connected()) {
        char buf[16];
        id(wifi_module).get_ip_addresses()[0].str_to(buf);
        it.printf(0, 0, id(font_main), "IP: %s", buf);
      } else {
        it.print(0, 0, id(font_main), "Wifi: Searching...");
      }

      it.line(0, 16, 127, 16);

      if (id(module_spanning).has_state() && id(module_stroom).has_state()) {
        it.printf(0, 22, id(font_main), "%.2fV | %.3fA",
                  id(module_spanning).state,
                  id(module_stroom).state);
      }

      it.print(0, 42, id(font_main), "Status: OK");
```

Exposed Home Assistant Entities

Once deployed, ESPHome automatically pushes the following high-precision entities to Home Assistant via the Native API:

- **SystemBus Current (sensor.systembus_current):**
  Measures the current of the central system bus in Amperes (with 3 decimal precision).

- **SystemBus Voltage (sensor.systembus_voltage)**:
  Measures the voltage of the central bus in Volts (with 2 decimal precision and an applied offset of +0.27V).
- **SystemBus Power (sensor.systembus_power)**:
  Calculates the active total power consumption in Watts, updated every 5 seconds.
- **Master Unit Wifi Signal (sensor.master_unit_wifi_signal)**:
  Tracks the RSSI signal strength of the Wi-Fi connection (updated every 60 seconds).
- **Master Unit IP Address (text_sensor.master_unit_ip_address)**:
  Exposes the local IP address of the gateway.
- **Mijn Knop (binary_sensor.mijn_knop)**:
  Captures the state of the physical hardware button on GPIO10 (equipped with a 50ms software debounce filter).
- **Master Unit Display Power (switch.master_unit_display_switch)**:
  Controls the software/hardware power state of the onboard SSD1306 OLED screen.
- **Master Unit Buzzer (switch.master_unit_buzzer)**:
  Toggles the onboard acoustic alarm buzzer on GPIO16.
- **Master Unit Reboot (button.master_unit_reboot)**:
  A button entity allowing you to remotely trigger a secure hardware restart of the ESP32-C6.
- **Master Unit Status LED (light.master_unit_status_led)**:
  Operates the red status indicator LED on GPIO11 via a monochromatic light platform.
-

6. Troubleshooting & Safety Warnings

- ⚠ **WARNING:** Always disconnect your solar panels and the main 12V camper battery system before wiring or making changes to the local I2C network bus to prevent accidental short circuits or ground loops.

Common Issue - Entity showing NaN or Unavailable:

- Verify that the SDA and SCL lines are not swapped.
- Ensure your central Master Unit is powered on and supplying a stable 3.3V to the module.
- Check if the I2C address matches your YAML configuration file (Default: 0x40).