Architecture

Complete system overview: FreeRTOS task model, the sensor polling loop, protobuf encoding, UDP transmission, inbound packet dispatch, and the memory layout. All application logic lives in a single FreeRTOS task (MainTask) defined in src/sensor_board/main.c.

Initialization Sequence

Phase 1: Hardware Setup (init_board, before the kernel starts)

```
void init_board() {
  MPU_Config_wrapper();
  SCB_EnableICache();
  SCB_EnableDCache();
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  /* NOTE: no threads here, kernel not initialized yet.
   * osKernelInitialize() is called by cubemx_main.c afterwards. */
}
```

Phase 2: Driver Initialization (start of MainTask)

1. BSP LEDs (Green, Blue, Red)
2. Logging over the ST-Link VCP UART (LOG_init(&hcom_uart[COM1]), 115200 baud)
3. IMU (imu_sensor_init)
4. pH sensor (ph_sensor_init with 3.3 V reference)
5. Two HX711 load cells with their GPIO map (PA5/PC7 and PA6/PB5), each powered up and auto-tared
6. Two pressure/FSR sensors (pressure_sensor_init)
7. Flow sensor (flow_sensor_init, pulse counting via EXTI callback)
8. Pump (pump_init on TIM3 CH3 PWM, then commanded to 50 percent and enabled as a startup default)

Phase 3: Communication Setup

1. ETH_init with static IP 192.168.0.111, netmask 255.255.255.0, gateway 192.168.0.1 and a link status callback that re-adds the static ARP entry when the link comes up
2. MAC address filtering for three allowed source MACs (ETH_setup_MAC_address_filtering)
3. Two statically allocated prioritised UDP transmit queues (80 entries each)
4. Packet dispatcher registration for five inbound message types (pH, IMU, load cell, pressure, pump)
5. ETH_udp_init(2, send_queues, DispatchPacket) and a static ARP entry for the destination board 192.168.0.222

Phase 4: Main Loop

The loop runs forever with a 5000 ms period (MAIN_TASK_DELAY_MS). Each iteration:

1. Read free heap; if below 4096 bytes log CRITICAL and sleep 10 s instead of polling
2. Toggle the three LEDs (visual heartbeat)
3. Build a SensorBoardDiagnostics struct (state OPERATING)
4. Poll pH, IMU, then load cells and pressure sensors (index loop over both units)
5. Poll the flow sensor (rate computed from pulses counted by the EXTI ISR since the last poll)
6. Build the pump status from commanded state, cross-checked against measured flow
7. Wrap each sensor message in a PBEnvelope and send it as a UDP datagram to the sample board
8. osDelay(MAIN_TASK_DELAY_MS)

Protobuf Encoding and UDP Transmission

Every outbound message is one PBEnvelope with a oneof payload. Encoding uses nanopb via pb_message_encode, which allocates a heap buffer that is freed after sending.

```
static void udp_send_envelope(uint8_t dest_ip[4], PBEnvelope *env) {
  if (!sendUDP) { return; }   /* transmit gate, false by default */
  uint8_t *encoded = NULL;
  size_t size = 0;
  result_t result = pb_message_encode(env, PBEnvelope_fields, &encoded, &size);
  if (result == RESULT_OK) {
    ETH_udp_send(dest_ip, PORT, encoded, (uint16_t)size, 1);
  }
  free(encoded);
}
```

Important: the static flag `sendUDP` in main.c is currently `false`, so envelope encoding and transmission are skipped entirely. Set it to true to actually transmit. There is a similar development flag `skip_sensor_polling` (currently false) that disables all sensor polling when true.

Inbound Packet Dispatcher

Received UDP packets are decoded by the shared packet dispatcher (components/common/packet_dispatcher). Handlers are registered with PACKET_HANDLER_CONFIG_STATIC per envelope payload tag:

Envelope tag	Handler	Behavior
ph_info	handle_sensor_ph_info	Log only
imu_info	handle_sensor_imu_info	Log only
load_cell_info	handle_sensor_load_cell_info	Log only
pressure_info	handle_sensor_pressure_info	Log only
pump_info	handle_sensor_pump_command	Actuates the pump: applies enabled, direction, and speed_percent to the hardware

Sensor Status Model

SensorState (operating state)

Code	Meaning
SENSOR_IDLE	Not connected, not implemented, or intentionally off
SENSOR_OPERATING	Normal operation, valid data
SENSOR_ERROR	Communication failure or invalid data

SensorStatus (connection status)

Code	Meaning
STATUS_OK	Healthy
STATUS_DISCONNECTED	No hardware detected (poll returned UNIMPLEMENTED or COMMS)
STATUS_ERROR	Unexpected failure

STATUS_INITIALIZING	Warming up (flow sensor first sample window)

Poll Result Mapping

handle_sensor_poll_result() maps driver results uniformly:

- RESULT_ERR_UNIMPLEMENTED or RESULT_ERR_COMMS: state IDLE, status DISCONNECTED (sensor not connected or driver not wired to hardware yet)
- Any other non-OK result: state ERROR, status ERROR
- RESULT_OK: caller then validates the data and picks OPERATING or ERROR plus a driver-specific error code

Every sensor produces one uniform log line per loop: `name | STATUS | STATE | detail`.

Pump Health Cross-Check

The pump is an open loop actuator (no current sense or fault line), so firmware cannot directly detect a connected pump. The only on-board proof that fluid is moving is the inline flow sensor, so the main loop derives pump status from it:

- Not initialised: ERROR / ERROR
- Commanded off (disabled or 0 percent): IDLE / OK (healthy, intentionally off)
- Commanded on and flow detected: OPERATING / OK
- Commanded on and no flow: OPERATING / DISCONNECTED (pump absent, dry, stalled, or the flow sensor is not installed)

Memory Layout

Region	Use
FreeRTOS heap, 64 KB	Task stacks, queues, protobuf encode buffers (malloc/free per message)
0x30000000, 32 KB, MPU non-cacheable	Ethernet DMA descriptors and buffers
0x30004900, 16 KB	LwIP RAM heap (MEM_SIZE)
Static queues	Two UDP send queues, 80 entries each, allocated at compile time (xQueueCreateStatic)

Error Handling Strategy

- Drivers never crash the loop: missing hardware degrades to IDLE/DISCONNECTED and the loop continues
- Each sensor unit reports independently (separate envelope, separate error code enum)
- Heap exhaustion protection: below 4096 bytes free, polling pauses for 10 s per iteration
- Encode failures are logged with result_to_short_str / result_to_desc_str and the buffer is freed in all paths

---

Revision #8
Created 2026-04-14 15:56:40 UTC by Shishir Nambiar
Updated 2026-07-08 14:26:42 UTC by Shishir Nambiar