pH Sensor

The pH sensor provides water quality measurement critical for environmental monitoring and anomaly detection.

Hardware Specifications

Parameter	Value
Model	DFRobot SEN0161 (Analog pH meter)
Interface	Analog ADC (compile gated, see Hardware Status below)
Reference Voltage	Configurable at init (3.3 V used in main.c; SEN0161 itself prefers a stable 5.0 V supply)
Board pin	PD14, labeled PH_ANALOG_DATA (must be moved, PD14 has no ADC function)
Measurement Range	0 to 14 pH units (clamped in software)
Accuracy	±0.1 pH @ 25°C (sensor datasheet)
Sample Averaging	40 samples, min and max excluded

Hardware Status

No ADC is enabled in CubeMX yet, so the ADC path is compile gated by `PH_SENSOR_USE_ADC`. Without that flag, `poll_ph_sensor()` returns RESULT_ERR_UNIMPLEMENTED and the sensor reports IDLE / DISCONNECTED (the firmware still links and runs). To enable:

1. In CubeMX enable an ADC (for example ADC1) and a channel on the pH input pin. PD14 (the current PH_ANALOG_DATA label) has NO ADC function on the STM32H753, so move the pH input to an ADC-capable pin (PA0, PC0, ...).
2. The SEN0161 is a 5 V board with output up to about 3 V. The STM32 ADC tops out at 3.3 V, so power or scale the board so its output never exceeds 3.3 V.
3. Build with `-D PH_SENSOR_USE_ADC` (optionally -D PH_SENSOR_ADC_HANDLE=hadc1, -D PH_SENSOR_ADC_MAX=65535).

Calibration Model

The sensor uses linear voltage-to-pH conversion (DFRobot SEN0161 formula):

```
pH = slope × Voltage + offset
Voltage = averaged_ADC / adc_max × reference_voltage
```

Default Parameters for SEN0161 @ 25°C:

- **Slope**: 3.5 (PH_DEFAULT_SLOPE)
- **Offset**: 0.0 by default, set via user calibration

The computed pH is clamped to the 0 to 14 range inside ph_sensor_update().

Data Structure

```
typedef struct {
    uint16_t raw_value;              /* averaged raw ADC value */
    float voltage;                   /* converted voltage */
    float ph_value;                  /* calculated pH (0-14), init 7.0 */
    float reference_voltage;         /* ADC reference */
    ph_calibration_t calibration;    /* { offset: float, slope: float } */
    uint16_t sample_buffer[40];      /* last 40 samples (PH_SAMPLE_COUNT) */
    uint8_t sample_index;            /* current position in buffer */
    uint8_t samples_collected;       /* samples collected so far (0-40) */
} ph_sensor_t;
```

Initialization & Usage

Initialize pH Sensor

```
ph_sensor_t ph_sensor;
ph_sensor_init(&ph_sensor, 3.3f);  /* 3.3 V reference, as in main.c */
```

Poll pH Sensor

```
result_t ph_result = poll_ph_sensor(&ph_sensor);

if (ph_result == RESULT_OK) {
    float ph_value, voltage;
```

```
    ph_sensor_get_value(&ph_sensor, &ph_value);
    ph_sensor_get_voltage(&ph_sensor, &voltage);
}
/* RESULT_ERR_UNIMPLEMENTED: ADC path not compiled in
 * RESULT_ERR_COMMS: HAL ADC start/conversion failed */
```

Manual Sample Addition

```
/* For manual sampling at regular intervals (e.g. every 20 ms) */
uint16_t adc_reading = 2048;
ph_sensor_add_sample(&ph_sensor, adc_reading);

/* Or feed a reading through the full pipeline (average + convert) */
ph_sensor_update(&ph_sensor, adc_reading, 4095);
```

Validation

```
result_t validate_ph_value(float ph_value);
/* Returns RESULT_OK if 0 <= ph_value <= 14
 * Returns RESULT_ERR_INVALID_DATA otherwise */
```

Sample Averaging Strategy

Parameter	Value
Sample Buffer Size	40 samples (PH_SAMPLE_COUNT)
Method	Circular buffer average, minimum and maximum values excluded (DFRobot sample code algorithm); simple average while fewer than 5 samples collected
Purpose	Noise filtering and stable readings

Averaging Algorithm

```
1. ADC sample added to circular buffer
2. Buffer averaged with min and max excluded
3. Averaged value converted to voltage
```

```
4. Voltage converted to pH via calibration (slope, offset)
5. pH clamped to 0-14
```

Two-Point Calibration Procedure

Step 1: Neutral Point (pH 7.0)

```
1. Short the BNC input or immerse the electrode in pH 7.0 buffer solution
2. Wait for a stable reading (~2 minutes)
3. Record the reported pH value
4. offset = 7.0 - recorded_value
```

Step 2: Slope Calibration (pH 4.0 or 10.0)

```
1. Immerse the electrode in a second known pH solution (pH 4.0 buffer)
2. Wait for a stable reading
3. Adjust the board gain potentiometer until the reading is 4.0,
   or compute: slope = (pH_reference - 7.0) / (V_reference - V_neutral)
4. Apply with ph_sensor_calibrate(&sensor, offset, slope)
```

Protobuf Message Format

```
message SensorBoardPHInfo {
    float ph_value;
    float voltage;
    SensorState state;
    PHErrorCode error_code;
}

enum PHErrorCode {
    PH_NO_ERROR = 0;
    PH_COMMUNICATION_FAILURE = 1;
    PH_INVALID_DATA = 2;
}
```

Error Handling (as in main.c)

```
if (ph_result == RESULT_ERR_UNIMPLEMENTED || ph_result == RESULT_ERR_COMMS) {
    /* Hardware not connected / ADC not enabled */
    diagnostics.ph_sensor.state = SensorState_SENSOR_IDLE;
    diagnostics.ph_sensor.error_code = PHErrorCode_PH_COMMUNICATION_FAILURE;
} else if (ph_result == RESULT_OK) {
    if (validate_ph_value(ph_value) == RESULT_OK) {
        diagnostics.ph_sensor.state = SensorState_SENSOR_OPERATING;
        diagnostics.ph_sensor.error_code = PHErrorCode_PH_NO_ERROR;
    } else {
        diagnostics.ph_sensor.state = SensorState_SENSOR_ERROR;
        diagnostics.ph_sensor.error_code = PHErrorCode_PH_INVALID_DATA;
    }
}
```

Integration Notes

- Single sensor instance in the main application, initialized with a 3.3 V reference
- Updates transmitted to the network at the main loop interval (5 seconds default), when the sendUDP flag is enabled
- Temperature compensation not implemented (assumes ~25°C)
- Because ph_sensor_update() clamps to 0-14, validate_ph_value() cannot fail on driver output; it protects against values from other sources
- Electrode response time: ~100-300 ms depending on pH change magnitude
- Unit tested on host: initialization defaults, voltage conversion, clamping, calibration (test/sensor_board/test_ph_sensor)

Revision #4
Created 2026-04-14 15:31:41 UTC by Shishir Nambiar
Updated 2026-07-08 14:27:42 UTC by Shishir Nambiar