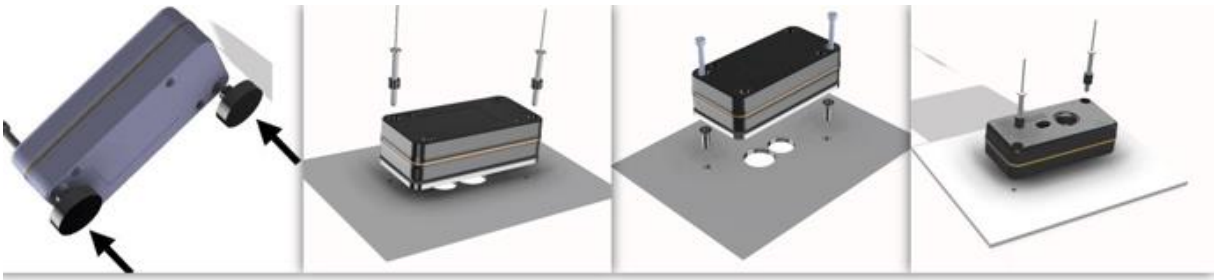
APOLLON Q QUICK START GUIDE

Die Apollon-Q Serie umfasst drahtlose IoT Füllstandsensoren mit präzisen Messergebnissen dank eines einzigartigen Messprinzips aus optischer Messung und Radar. Die Sensoren erfassen Füllstände bis 2,50 Meter – geeignet für Stück-, Flüssig- oder Schüttgut in Behältern, Schächten, Kanälen oder im Smart-Waste-Bereich. Selbst in kleinen Behältern wie Papierkörben liefern sie zuverlässige Daten. Sie bieten eine hohe Reichweite, kurze Mess- und Sendeintervalle sowie erweiterte Funktionen. Unterstützt werden Kommunikationsstandards wie MIOTY®, NB-IoT, LoRaWAN® und LTE-CAT-M1.

Bitte beachten Sie die Warnungen und Hinweise aus der Betriebsanleitung, um die Sicherheit für Sie, Ihr Umfeld und den Sensor zu gewährleisten!

MONTAGE UND INSTALATION

Installieren Sie den Sensor an einer Innenraumwand in einer Höhe zwischen 1,50m und 1,80m. Achten Sie darauf, dass die LEDs sich auf der rechten, unteren Seite liegen. Beachten Sie, dass der Sensor mindestens 20cm Abstand zu Personen hat und sich in einer Umgebung unterhalb von 2000m über dem Meeresspiegel befindet. Nutzen Sie für die Befestigung entweder Magnete, Nieten oder geeignete Schrauben. Die detaillierten Schritte finden Sie in der Betriebsanleitung.



1. Magnetmontage
2. Vorderwandmontage mit Blindnieten
3. Vorderwandmontage mit Blindnietmuttern
4. Rückwandmontage mit Blindnieten

INBETRIEBNAHME

Auf dem Sensor befinden sich zwei Hallsensoren (Magnetfeldschalter). Die folgende Grafik zeigt die Lage der Hallsensoren und die empfohlene Platzierung der Magnete.

Der obere, mittige Magnetfeldschalter (1) kann in drei verschiedenen Modi betrieben werden:

- Behälter ist geschlossen, wenn der Magnet angelegt ist
- der Behälter ist offen, wenn der Magnet angelegt wird
- der Sensor zählt eine Öffnung, wenn der Magnet zweimal durchläuft

Für neuere Sensoren wird die Funktion der Öffnungsdetektion vom Accelometer übernommen.

INBETRIEBNAHME DES SENSORS ÜBER BLE

Für Sensoren mit ToF und ohne Radar steht BLE nicht zur Verfügung. Für diese Ausführungen befolgen Sie bitte die Schritte unter Punkt 0 um diese in Betrieb zu nehmen und zu konfigurieren.

Für alle anderen Sensoren können Sie die bequeme Variante per BLE wie folgt verwenden:

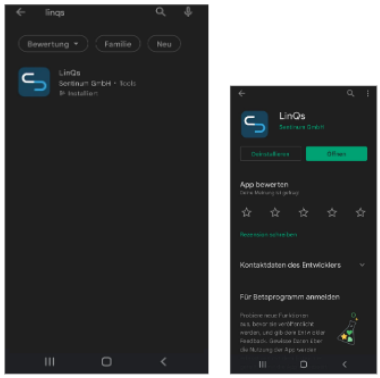
1. Alle weiteren Varianten können per BLE mit dem Handy aktiviert werden. Verwenden Sie dazu einfach unsere LinQs APP, die Sie im APP Store und Google Play Store finden.
2. Aktivieren Sie den Sensor, indem Sie den linken unteren Hallsensor (2) kurz, mind. 2 Sekunden mit einem Magneten auslösen. Der Sensor sollte nun eine akkustische Tonfolge zurück geben. Der Advertising-Modus wird anschließend automatisch gestartet.
3. Nun könnnen Sie den Sensor bequem per BLE von Ihrem Endgerät aus ansteuern. Achten Sie darauf, dass es auf Ihrem Handy eingeschaltet ist, und Sie sich in Reichweite des Sensors befinden um diesen erreichen und ansteuern zu können.

NFC INBETRIEBNAHME, PARAMETRISIERUNG UND LAGE DES NFC TAGS

Die Aktivierung kann über eine NFC App erfolgen. Dazu wird ein Smartphone benötigt. Die App kann in den jeweiligen App Stores heruntergeladen werden. Suchen Sie dazu einfach nach “Sentinum LinQs” und laden Sie die LinQs App herunter.

Google Play Store

iOS App Store



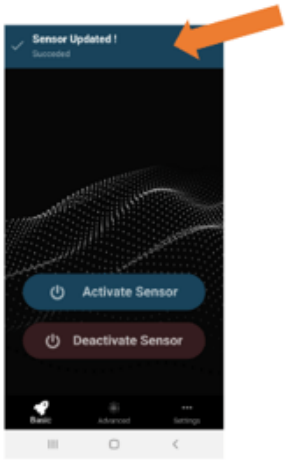
Lokalisieren Sie zuerst den Tag auf dem Sensor und dann den Reader an Ihrem Endgerät. Die Lage des NFC Tags finden Sie an der Position des orangenen Pfeils.

Öffnen Sie die App und aktivieren Sie den Sensor. Um den Sensor in den Grundeinstellungen zu starten, klicken Sie im Startmenü der App auf die Schaltfläche "Sensor aktivieren". Legen Sie nun Ihr Gerät auf die NFC-Markierung des Sensors.

1.

2.

3.



Wenn der Sensor aktiviert ist, wird "Sensor aktualisiert!" angezeigt. Danach können Sie mit der Aktivierung der anderen Sensoren fortfahren.

PAYLOAD DECODER APOLLON Q SERIE

```
// Up to Date Apollon decoder
function Decoder(bytes, port) {

    // Conversion of signed integers
    function uncomplement(val, bitwidth) {
        var isnegative = val & (1 << (bitwidth - 1));
        var boundary   = (1 << bitwidth);
        var minval     = -boundary;
        var mask       = boundary - 1;
        return isnegative ? minval + (val & mask) : val;
    }

    var decoded = {};

    if (port === 1) {

        // Attributes
        decoded.base_id         = bytes[0] >> 4;
        decoded.major_version   = bytes[0] & 0x0F;
        decoded.minor_version   = bytes[1] >> 4;
        decoded.product_version = bytes[1] & 0x0F;

        // Telemetry
        decoded.up_cnt               = bytes[2];
        decoded.battery_voltage      = ((bytes[3] << 8) | bytes[4]) /
1000.0;
        decoded.internal_temperature = bytes[5] - 128;
        decoded.alarm                = bytes[6] ? "ALARM" : "NO ALARM";
        decoded.master_value         = ((bytes[7] << 8) | bytes[8]); // in
mm

        // Start of version specific fields
        var byte_cnt = 9;

        // ToF onboard
        if (bytes[1] & 0x01) {
            decoded.tof_status   = bytes[byte_cnt++];
            decoded.tof_distance = ((bytes[byte_cnt++] << 8) |
bytes[byte_cnt++]); // in mm
            decoded.tof_index    = bytes[byte_cnt++];
        }

        // Radar onboard
        if (bytes[1] & 0x02) {
            decoded.radar_status      = bytes[byte_cnt++];
            decoded.radar_no_peaks    = bytes[byte_cnt++];
            decoded.radar_distance_1  = ((bytes[byte_cnt++] << 8) |
bytes[byte_cnt++]);
            decoded.radar_amplitude_1 = uncomplement((bytes[byte_cnt++] <<
8) | bytes[byte_cnt++], 16);
            decoded.radar_distance_2  = ((bytes[byte_cnt++] << 8) |
bytes[byte_cnt++]);
            decoded.radar_amplitude_2 = uncomplement((bytes[byte_cnt++] <<
8) | bytes[byte_cnt++], 16);
```

```
            decoded.radar_distance_3  = ((bytes[byte_cnt++] << 8) |
bytes[byte_cnt++]);
            decoded.radar_amplitude_3 = uncomplement((bytes[byte_cnt++] <<
8) | bytes[byte_cnt++], 16);
        }

        // ACC onboard
        if (bytes[1] & 0x04) {
            decoded.acc_status      = bytes[byte_cnt++] ? "Fehler" : "OK";
            decoded.acc_orientation = bytes[byte_cnt++];
            decoded.acc_open        = bytes[byte_cnt++];
            decoded.acc_open_cnt    = ((bytes[byte_cnt++] << 8) |
bytes[byte_cnt++]);
            decoded.acc_impact      = bytes[byte_cnt++] ? "Vandalismus" :
"OK";
        }

        // Hall onboard
        if (bytes[1] & 0x08) {
            decoded.hall_open     = bytes[byte_cnt++];
            decoded.hall_open_cnt = ((bytes[byte_cnt++] << 8) |
bytes[byte_cnt++]);
        }
    }
    return decoded;
}
```