

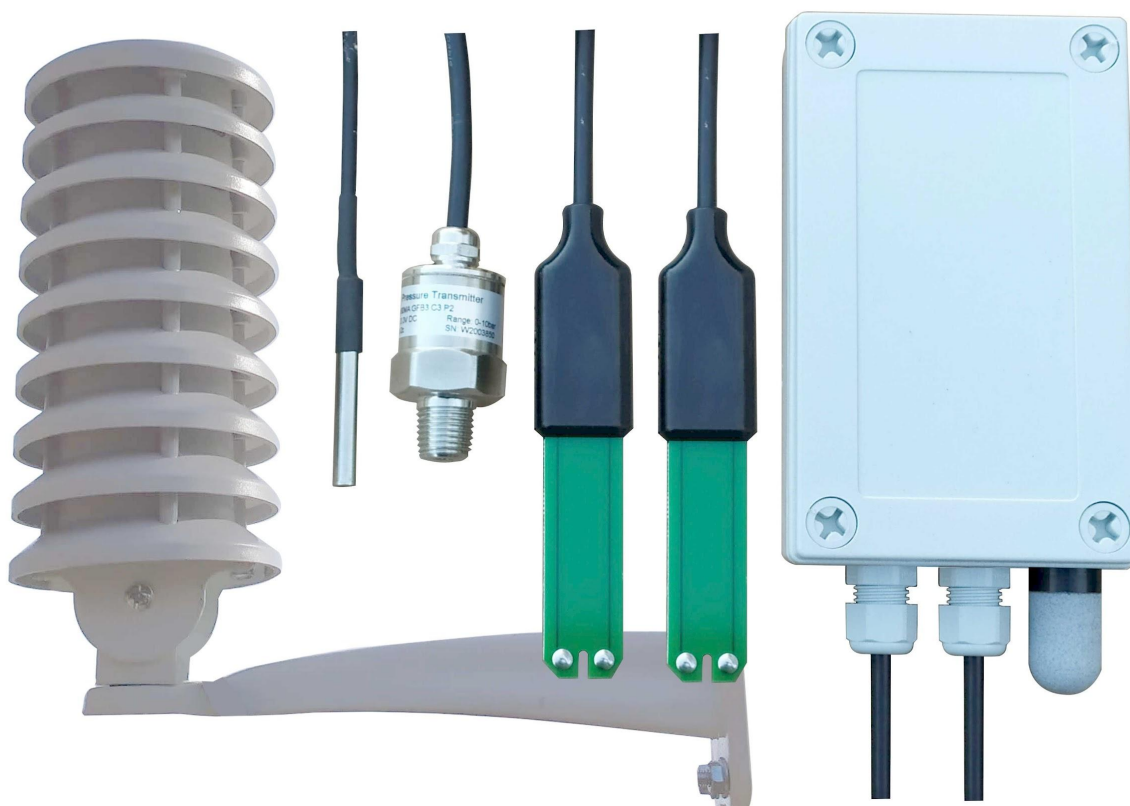


## PM-IO-5-SM LoRaWAN IO device

---

### FEATURES

- LoRaWAN v 1.0.3 class A device, may be activated with OTAA or ABP.
- Setup via USB CDC-ACM virtual COM port.
- Android application for USB setup
- Pulse counter
- Water leak detector and valve close output trigger
- Dust and waterproof IP67
- Optional Single or Double Tinovi Soil Moisture/Temperature/(Beta version EC) sensor option
- Optional BME280 Air Temperature/Humidity/Pressure sensor option (Not waterproof)
- Optional OPT3001 light lux sensor option
- Optional 12v booster pulse solenoid reverse switch output option
- Leaf moisture sensor





## PM-IO-5-SM LoRaWAN IO device

---

### ELECTRICAL PROPERTIES

	Min/Sleep	Typical	Max
Supply voltage (VCC), V	3.3	4.1	6
Working current (VCC=3.6V), mA	Sleep 0.01mA	22	150
Operating Temperature Range, Celsius	-20	25	70

Sensor sleep mode consumes 10uA or 0.01 mA. Active mode consumption 12-24 mA depending on sensor configuration and 150 mA in transmitting mode.

Required battery is **3.7v 18650 (optional 26650)**.

Device may be powered from any 5v Micro-USB power source. After power up sensors wait 10 seconds for USB configuration session connection from android application or USB CDC terminal, if none is connected sensors turns off USB hardware and starts normal operation. Device should be restarted, before attaching to USB for configuration, it means battery should be removed

---

### PHYSICAL PROPERTIES

dimensions 80x130x35

Soil Sensors cable length 1.5m (5m optional)

---

### SOIL SENSOR MEASUREMENT PROPERTIES

Soil Sensor	Resolution	Range /avg Tolerance
Dielectric permittivity ( $\epsilon$ ) (Temperature corrected)	0.1 $\epsilon$	1 (air) to 80 (water) /5%
Volumetric water content - VWC calculation from Dielectric permittivity $\epsilon$ . $VWC = 0.002974 * \text{pow}(\epsilon, 2) + 0.07424 * \epsilon - 1.295$ ;		
Beta version EC, Electrical Conductivity (mS/m) <b>(0.1 mS/m = 1 uS/cm)</b>	0.1 mS/m	0...300 mS/m 20% 300...800 mS/m 40%
Temperature (°C)	0.1°C	-20 to



## PM-IO-5-SM LoRaWAN IO device

		70°C/3%
Degree of water saturation in the soil	0.1%	0 – 100% /8%

### DATA OUTPUT FORMAT

Format what you are usually getting from LoRaWAN is hex string, which you have to convert to bytes. Data output payload is variable length. To properly parse data you have to decode bits in 1st byte::

1. 1-byte setup bits:
  - a. Bit 0 - enable 1st Tinovi PM-WCS-3-I2C Soil sensor
  - b. Bit 1 - enable BME280 air temp - humidity sensor
  - c. Bit 2 - enable OPT3001 light lux sensor
  - d. Bit 3 - enable 2nd Tinovi PM-WCS-3-I2C Soil sensor
  - e. Bit 4 - enable pulse counter
  - f. Bit 5 - enable water pressure sensor
  - g. Bit 6 - Water Leak detector flag - leak detected
  - h. Bit 7 - valve state report - On/Off
2. 1-byte Battery voltage percent (board version 1.1): measures battery voltage and calculates percentage from 2.8v to 4.2v; To calculate actual battery voltage use equation:  $2.1 + 0.022 * (\text{battery reading})$ . for example if battery measurement shows 50%, then actual voltage is  $2.1 + 0.022 * 50 = 3.5\text{v}$  resolution: 1% and range: 2.1v – 0% to 4.2v – 100%

#### --IF Tinovi PM-WCS-3-I2C sensor enabled (Bit 0) in setup bit:

3. 2-byte 16-bit signed integer (divide by 100) Dielectric permittivity ( $\epsilon$ ) (Temperature corrected) resolution: 0.1 $\epsilon$  (avg.Tolerance 5%) and range 1 (air) to 80 (water)
4. 2-byte 16-bit signed integer (divide by 10) Electrical Conductivity (mS/m) resolution: 0.01 mS/m (avg.Tolerance 20%)
5. 2-byte 16-bit signed integer (divide by 100) Temperature (°C) resolution: 0.1°C and range: -20 to 60°C (avg.Tolerance 3%)
6. 2-byte 16-bit signed integer Volumetric Water Content (%) resolution: 1% and range: 0 – 100% VWC Note: VWC is calculated from dielectric permittivity by Topp equation (Topp et al, 1980):  $\theta = 4.3 \cdot 10^{-6} \epsilon^3 - 0.00055 \epsilon^2 + 0.0292 \epsilon - 0.053$

#### --IF BME280 enabled (Bit 1) in setup bit:

7. 2-byte 16-bit signed integer Air Temperature (divide by 100)
8. 2-byte 16-bit signed integer Air Humidity (divide by 100)
9. 2-byte 16-bit signed integer Air pressure (add 50000)

#### --IF OPT3001 light lux sensor enabled (Bit 2) in setup bit :

10. 4-byte 32-bit integer Light lux



### --IF pulse input is enabled (Bit 4) in setup bit:

11. 4-byte 32-bit integer pulse counter output

### --IF 2nd Tinovi PM-WCS-3-I2C sensor enabled (Bit 3) in setup bit:

12. 2-byte 16-bit signed integer (divide by 100) Dielectric permittivity ( $\epsilon$ ) (Temperature corrected) resolution:  $0.1\epsilon$  (avg.Tolerance 5%) and range 1 (air) to 80 (water)
13. 2-byte 16-bit signed integer (divide by 10) Electrical Conductivity (mS/m) resolution: 0.01 mS/m (avg.Tolerance 20%)
14. 2-byte 16-bit signed integer (divide by 100) Temperature ( $^{\circ}\text{C}$ ) resolution:  $0.1^{\circ}\text{C}$  and range:  $-20$  to  $60^{\circ}\text{C}$  (avg.Tolerance 3%)
15. 2-byte 16-bit signed integer Volumetric Water Content (%) resolution: 1% and range: 0 – 100% VWC Note: VWC is calculated from dielectric permittivity by Topp equation (Topp et al, 1980):  $\theta = 4.3 \cdot 10^{-6} \epsilon^3 - 0.00055 \epsilon^2 + 0.0292 \epsilon - 0.053$

### --IF Pressure sensor (Bit 5) is setup:

16. 2-byte 16-bit signed integer (divide by 100) pressure reading

### --IF New protocol versions (need to check if payload has more bytes to decode):

17. 1-byte setup bits 2:
  - a. Bit 0 - not used
  - b. Bit 1 - enable Tinovi Leaf sensor
  - c. Bit 2 - enable ADC sensor
  - d. Bit 3 - enable Wind sensor
  - e. Bit 4 - enable Wei sensor

### --IF Tinovi Leaf Moisture sensor enabled (Bit 2:1) in setup2 bit:

18. 2-byte 16-bit signed integer Air Temperature (divide by 100)
19. 2-byte 16-bit signed integer Leaf Humidity (divide by 100)

### --IF ADC enabled (Bit 2:2) in setup2 bit:

20. 4-byte 32-bit float ADC output

### --IF Wind enabled (Bit 2:3) in setup2 bit:

21. 1-byte wind direction output
22. 4-byte 32-bit float Wind Speed output

### --IF Scale enabled (Bit 2:4) in setup2 bit:

23. 4-byte 32-bit int Wind Speed output (divide by 10)



## PM-IO-5-SM LoRaWAN IO device

---

### COMMAND LINE

LoRaWAN configuration may be done via USB CDC-ACM virtual com port with 115200 baudrate. Before configuration you need to switch off device and then connect to usb port of your computer. On linux it will be detected as `/dev/ttyACM0`.

Device waits for 10 second after power up for USB CDC-ACM connection, then it goes to normal operation mode and is not reachable via USB CDC-ACM port. For linux permission settings please [consult](#).

On Linux you may use [minicom](#) utility to configure sensor.

For windows [PuTTY](#) may be [used](#).

We recommend Android mobile application for sensor setup and data reading/storage

### COMMAND LINE DEVICE CONFIGURATION COMMANDS

Command	Response	Explanation
<b>int</b> <val>	<Val> OK	Setup data sens interval in seconds
<b>appeui</b> <val>	OK	Sets <b>Application EUI</b> for OTAA activation. Default AppEUI: <b>70B3D57ED0015AA3</b>
<b>deveui</b> <val>	OK	Sets <b>Device EUI</b> for OTAA activation.
<b>key</b> <val>	OK	Sets <b>App Key</b> for OTAA activation.
<b>addr</b> <val>	OK	Sets <b>Device Address</b> for ABP activation.
<b>nwks</b> <val>	OK	Sets <b>Network Session Key</b> for ABP activation.
<b>apps</b> <val>	OK	Set APPSKEY for ABP
<b>check</b>	OK	0-Disable/1-Enable OTAA link check mode EU868 (helps reconnect device on gateway outage)
<b>sb</b>	OK	Set LoRaWAN subband 0..7 (US915 and AU921)
<b>window</b>	OK	Set LoRaWAN downlink window expanding 0..100 (%)
<b>csv</b>	Config csv	Get config in values comma separated
<b>ver</b>	<version>	print device type and version number
<b>ec</b> <uS/m> <0 or 1>	OK	Submerge sensor in to calibration fluid and input sensor number starting from 0 (0 - 1st, 1 - 2nd) and EC value in <b>uS/cm</b>
<b>air</b> <0 or 1>	OK	Hold sensor in air and execute command 0% offset() (0 - 1st; 1 - 2nd sensor)
<b>water</b> <0 or 1>	OK	Submerge sensor in to water or soil with water and execute the command (0 - 1st; 1 - 2nd sensor)
<b>debug</b>		Enable debug mode, device does not go to sleep



## PM-IO-5-SM LoRaWAN IO device

<b>reset</b>		Reset device
<b>Set &lt;b1&gt; &lt;b2&gt;</b>	OK	Set config byte1:  a. Bit 0 - enable 1st Tinovi PM-WCS-3-I2C Soil sensor b. Bit 1 - enable BME280 air temp - humidity sensor c. Bit 2 - enable OPT3001 light lux sensor d. Bit 3 - enable 2nd Tinovi PM-WCS-3-I2C Soil sensor e. Bit 4 - enable pulse counter f. Bit 5 - reserved g. Bit 6 - Water Leak detector flag - leak detected h. Bit 7 - valve state report - On/Off  setup2 bits:  i. Bit 0 - not used j. Bit 1 - enable Tinovi Leaf sensor k. Bit 2 - enable ADC sensor l. Bit 3 - enable Wind sensor m. Bit 4 - enable Wei sensor
<b>pulse</b>	<Val> OK	Set pulse counter offset
<b>vwcon</b>	OK	Set valve ON switch trigger VWC limit
<b>vwcoff</b>	OK	Set valve OFF switch trigger VWC limit
<b>vtime</b>	OK	Set valve pulse duration in milliseconds
<b>switch</b>	OK	1- Switch valve ON , 0 - Switch valve OFF
<b>leak</b>	OK	1- Switch valve OFF on water leak enabled/ 0- disabled

### DOWNLINK MESSAGE

Device supports LoRaWAN downlink messages on port 1. Downlink message contains 1st command byte followed by value:

- 0x01 + device sleep time (data sending interval) adjustment. Message should contain 5 bytes, 1st byte is 0x01 and following 4 bytes is unsigned 32-bit integer time in seconds to sleep. For example sending HEX 010000000A will set interval to 10 seconds, but 0100000E10 will set interval to 3600 secs = 1 hour.
- 0x02 + data config byte1 :
  - Bit 0 - enable Tinovi PM-WCS-3-I2C Soil sensor
  - Bit 1 - enable BME280 air temp - humidity sensor
  - Bit 2 - enable OPT3001 light lux sensor
  - Bit 3 - 2nd soil sensor
  - Bit 4 - enable pulse counter
  - Bit 5 - reserved
  - Bit 6 - Water Leak detector flag - leak detected



- h. Bit 7 - read only -valve status

Optional Data config byte 2

- i. Bit 0 - not used
- j. Bit 1 - enable Tinovi Leaf sensor
- k. Bit 2 - enable ADC sensor
- l. Bit 3 - enable Wind sensor
- m. Bit 4 - enable Wei sensor

- 3. 0x03 +1-byte 0x01 -valve ON or 0x00 - valve off
- 4. 0x04 +2-byte valve pulse time millis
- 5. 0x05 +4-byte max valve on time millis, 0 -disabled
- 6. 0x06 +2-byte valve trigger ON VWC limit
- 7. 0x07 +2-byte valve trigger OFF VWC limit
- 8. 0x08 +4-byte pulse counter offset
- 9. 0x09 +1-byte water leak valve OFF trigger enable =1 , disable =0
- 10. 0x0A +1-byte sensor number 0 or 1 - if soil moisture present - calibrate AIR
- 11. 0x0B +1-byte sensor number 0 or 1 - if soil moisture present - calibrate WATER
- 12. 0x0C +2-byte EC value + 1-byte sensor number 0 or 1 - if soil moisture present - calibrate EC. Example sending 0C058500 will calibrate sensor for 1413 uS/cm . 0C058500 for 1st or 0C058501 for 2nd sensor.
- 13. 0x0D +4-byte valve open time in seconds - valve will be closed after this time (device will sleep all that time and close valve on wake up)
- 14. 0x0E +1-byte 0x01 -valve ON or 0x00 - valve off; +1-byte actual pressure less (<)=0 or greater (>)=1; +2-byte pressure in atmosphere \* 100 check pressure value; +2-byte check time after in seconds; +1-byte check times. Send command to open/close valve and later check if system is pressurised accordingly, and if not then repeat operation for n times if condition not met. Total 8 bytes. See sample **0E010100C8000A02:**
  - 0E** -> cmd id
  - 01** -> open valve
  - 01** -> current pressure is greater
  - 00** -> Atm first byte, see below
  - C8** -> 2.0 \* 100 = 200 is C8 hex, this means that after switch operation, pressure shall be greater than 2 atm.
  - 00** -> Time first byte, see below
  - 0A** -> time in seconds to check after walve is triggered
  - 02** -> repeat checking 2 times. If condition is met then recheck is not continued



**Manual Data rate set**

```
0: DR_SF12 = EU868_DR_SF12,  
1: DR_SF11 = EU868_DR_SF11,  
2: DR_SF10 = EU868_DR_SF10,  
3: DR_SF9  = EU868_DR_SF9,  
4: DR_SF8  = EU868_DR_SF8,  
5: DR_SF7  = EU868_DR_SF7,  
6: DR_SF7B = EU868_DR_SF7B,  
7: DR_FSK  = EU868_DR_FSK,  
8: DR_NONE = EU868_DR_NONE  
Default is DR_SF7 14  
drtx 5 14
```





### VALVE CONTROL

#### 1. from battery boosted DC reversible pulsed output for 12VDC latching solenoid valve

Valve controller switch max output continues current is 0.5A, (1A max pulse). Battery max current draw is almost 3 times more than your load current, if you apply 0.5A load current, your battery should support 1.5A current. Switch Supports reverse polarity latching relay. In open direction Red wire is Positive, Black negative, but in close direction Black - Positive and Red - Negative. Valve may be controlled via downlink messages, send 0x0301 to open valve, and 0x0300 to close valve. Device remembers last valve status sends it in read only setup byte bit 7: if setup bit 7 is set, then valve is open, if bit is cleared - valve is closed state. (See payload decoder sample- parameter "valve" - indicates valve status 0-close, 1-open). See DOWNLINK MESSAGES section for other valve control commands. For new models as alternative to booster dedicated 12v battery for switch may be used. That allows up to 3A current via switching circuit from battery.

#### 2. latching relay FTR-B4GB003Z-B05

(Normally open)NO (normally closed)NC (Latching) relay. The device can be powered 5 to 24VAC or 5 to 24VDC, or a standard 18650 battery

Relay parameters:

AC contacts rating 0.3A / 125V AC

DC contacts rating 1A / 30V DC

Contact current max. 2A

Switched voltage max. 220V DC, max. 250V AC

#### 3. SSR switch for AC coil valve

Switching AC voltage up to 1.2A. screw terminal inside the enclosure and additional gland(ventilation gland)  
Device from external power AC or DC 5 to 24V

---

### PULSE COUNTER

Pulse counter supports normally open contact reed switch pulse output. Use Red and Black wires for pulse counter switch. Black wire is pulse input, Red wire is +3.3 V voltage reference. To count pulse 3.3V voltage (Red) should be applied to pulse input (Black). Counter allows max 50 pulses per second. To account pulse device shortly wakes up, this means many pulse may consume device battery faster. Currently pulse offset is not preserved on battery removal. Pulse current value may be set by USB-SERIAL or android application or downlink message. If downlink message [08] is received with zero pulse offset value, then current pulse value is stored to flash memory and loaded on startup. Offered workaround - application may update device pulse offset to latest pulse counter offset in case device gives pulse offset smaller than latest pulse offset received from device by application.

---

### WATER LEAK DETECTOR

Water leak detector on trigger will interrupt device sleep and send message immediate with water leak flag enabled in settings byte. It will repeat sending this flag in 5 messages, then will reset the flag. If valve OFF on water leak trigger is enabled, then valve off will be triggered. 2 electrode water leak detector is



## PM-IO-5-SM LoRaWAN IO device

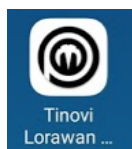
supported, triggers when resistance between electrodes is less than 250 kOhm.  
For water leak detector are used Green and White wires.

### SOIL SENSOR CALIBRATION ION

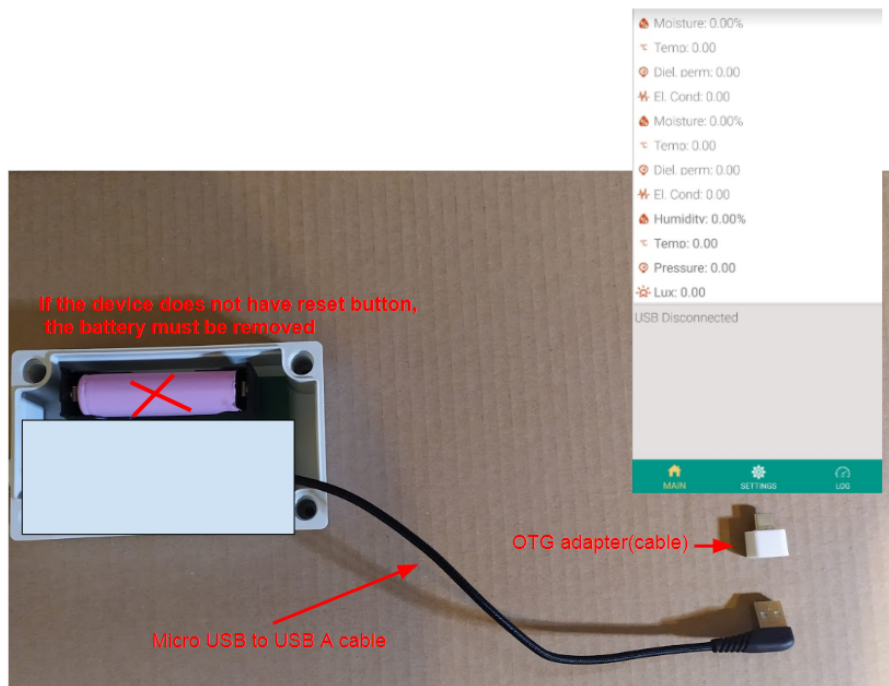
Sensors are already factory calibrated, but in case needed they may be recalibrated using USB terminal interface as described for device specific commands.  
Hold sensor in air and issue this command **air**, response OK.  
Put sensor in to water and issue command **water**, response OK.  
For **ec** <uS/m> calibration put sensor into liquid and set right sensor reading value in uS/cm.

We recommend Android mobile application for sensor setup and data reading/storage

### SOIL SENSOR CALIBRATION ON ANDROID APP



1. Open android application,
2. If the device does not have reset button, the battery must be removed
3. Connect the device to the android phone through THE OTG adapter

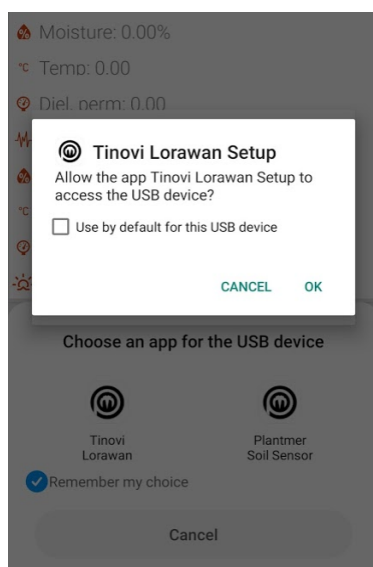


4. Press device's reset button if your device has one and the battery is inserted
5. Check "Remember my device" and tap "Tinovi Lorwan" icon, tap "OK" to the question "Allow the Tinovi Lorawan Setup to access the USB device". These messages may be different on different phones.

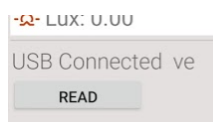


## PM-IO-5-SM LoRaWAN IO device

---



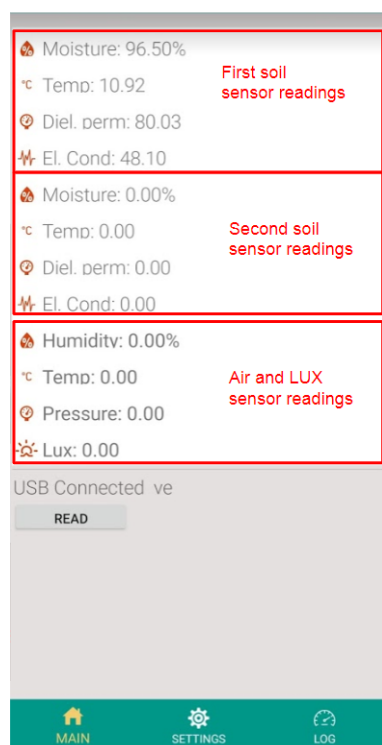
6. wait a few seconds for USB connected status or connect
7. The connection must be made in 10 seconds since the device is connected to your phone, otherwise the LED of the device starts to flicker and that means the device is in working mode.



8. When connecting, press the READ button. Device LED will flash and the application MAIN tab will display sensor data(Reads only sensors which are set in the Setting tab)

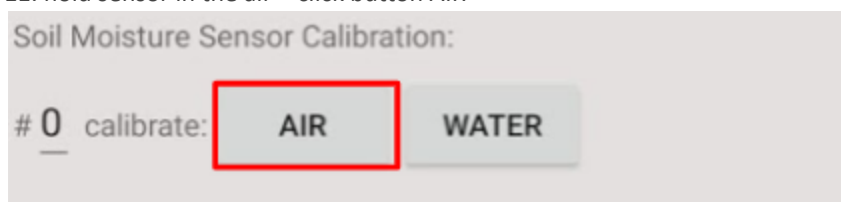


## PM-IO-5-SM LoRaWAN IO device



9. Make sure that the soil moisture sensor is dry before calibration in the air  
10. go to SETTINGS tab,

11. hold sensor in the air – click button AIR



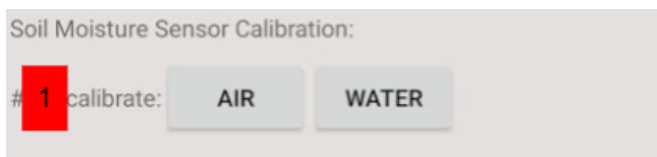
12. Go to MAIN tab click READ to see values after calibration  
13. submerge sensor in the water or soil with water, click button WATER  
14. Go to MAIN tab click READ to see values after calibration

15. To calibrate the second soil moisture sensor, type number 1 in red mark field picture below and repeat points 9 to 14



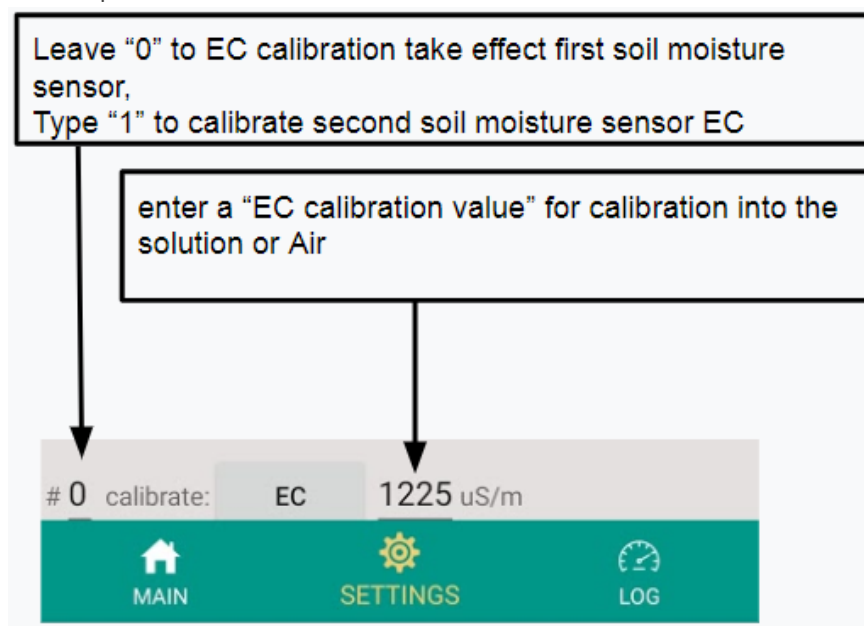
## PM-IO-5-SM LoRaWAN IO device

---



### SOIL SENSOR Beta version Electrical conductivity, EC CALIBRATION ON ANDROID APP

1. Connect phone to the device as described on section **SOIL SENSOR CALIBRATION**



2. Enter "EC calibration value" field "0", hold sensor in the air – click button EC
3. Type in uS/cm CONDUCTIVITY CALIBRATION SOLUTION value across your solution manufacturer's data table



## PM-IO-5-SM LoRaWAN IO device

---

°C	°F	µS/cm
0	32.0	776
5	41.0	896
10	50.0	1020
15	59.0	1147
16	60.8	1173
17	62.2	1199
18	64.4	1225
19	66.2	1251
20	68.0	1278
21	69.8	1305
22	71.6	1332
23	73.4	1359
24	75.2	1386
25	77.0	1413
26	78.8	1440
27	80.6	1467
28	82.4	1494
29	84.2	1521
30	86.0	1548
31	87.8	1575

EXAMPLE.

4. Submerge soil moisture sensor EC pads(or Stainless steel rivets) into the calibration solution and press EC button
5. Go to MAIN tab click READ to see values after calibration

---

### ANDROID APP

<https://play.google.com/store/apps/details?id=com.tinovi.soilmaster>



Android application lets you configure and calibrate soil moisture sensors your device.

1. When installing application you have to allow application to use your phone's USB port.
2. You need USB-OTG connector - adapter in your phone side - Micro-USB or USB-Type-C and Micro-USB charging cable to connect device to USB OTG phone adapter.
3. Before connecting to your phone you have to remove battery from device to reset device.
4. After connecting device android phone will ask you to allow connect to USB, approve it.



## PM-IO-5-SM LoRaWAN IO device

---

5. Wait while application shows USB Connected in main page.
6. Click read in main page to read sensor values.
7. Switch to settings tab to setup your device

---

### THINGS PARK INTEGRATION

There is option to feed data from your thingspark device to Tinovi cloud dashboard. **Note! This is demo cloud, and we do not guarantee your data retention and service uptime.**

Login with google account to [tinovi.com/graf](https://tinovi.com/graf) and under Organization->Application click on top choose:

- [Create Application - Device type](#)

then type In some name for your app - single word and specify *io.zerver.cd.board.decoder.SensSwitchDecoder* in decoder field, then click save, then again edit and in the top you will see:

### Device Type - Application Id

#### <YourNewAppId>

use attach this app id at the end of the URL for thingpark integration like:  
<http://tinovi.com/api/v1/thingpark/<MyAppId>>



### JavaScript PAYLOAD DECODER

```
var bytesToFloat32 = function(/*byte[]*/byteArray) {
    var buf = new ArrayBuffer(4);
    var view = new DataView(buf);

    // set bytes
    byteArray.forEach(function (b, i) {
        view.setUint8(i, b);
    });
    return view.getFloat32(0);
};

var bytesToInt = function(/*byte[]*/byteArray, dev) {
    var value = 0;
    for (var i = 0; i < byteArray.length; i++) {
        value = (value * 256) + byteArray[i];
    }
    return value/dev;
};

var bytesToSignedInt = function(bytes, dev) {
    var sign = bytes[0] & (1 << 7);
    var x = ((bytes[0] & 0xFF) << 8) | (bytes[1] & 0xFF);
    if (sign) {
        x = 0xFFFF0000 | x;
    }
    return x/dev;
};

function decodeUplink(input) {
    var bytes = input.bytes;
    var decoded = {};
    var pos = 1;
    decoded.valv=((bytes[0] >> 7) & 1);
    decoded.leak=((bytes[0] >> 6) & 1);
    decoded.bat = bytes[pos++];
    if(((bytes[0] >> 0) & 1)===1){ //SOIL
        decoded.e25=bytesToInt(bytes.slice(pos,pos+2),100);
        pos = pos+2;
        decoded.ec=bytesToInt(bytes.slice(pos,pos+2),10);
        pos = pos+2;
        decoded.temp=bytesToSignedInt(bytes.slice(pos,pos+2),100);
        pos = pos+2;
        decoded.vwc=bytesToInt(bytes.slice(pos,pos+2),1);
        pos = pos+2;
    }
    if(((bytes[0] >> 1) & 1)===1){ //BME
        decoded.airTemp=bytesToSignedInt(bytes.slice(pos,pos+2),100);
        pos = pos+2;
        decoded.airHum=bytesToInt(bytes.slice(pos,pos+2),100);
    }
}
```





```
pos = pos+2;
var airPressuse = bytesToInt(bytes.slice(pos,pos+2),1)+50000;
if(airPressuse!=65536){
    decoded.airPres=airPressuse;
}
pos = pos+2;
}
if(((bytes[0] >> 2) & 1)===1){ //OPT
    decoded.lux=bytesToInt(bytes.slice(pos,pos+4),100);
    pos = pos+4;
}
if(((bytes[0] >> 4) & 1)===1){ //PULSE
    decoded.pulse=bytesToInt(bytes.slice(pos,pos+4),1);
    pos = pos+4;
}
if(((bytes[0] >> 3) & 1)===1){ //SOIL
    decoded.e25_1=bytesToInt(bytes.slice(pos,pos+2),100);
    pos = pos+2;
    decoded.ec_1=bytesToInt(bytes.slice(pos,pos+2),10);
    pos = pos+2;
    decoded.temp_1=bytesToSignedInt(bytes.slice(pos,pos+2),100);
    pos = pos+2;
    decoded.vwc_1=bytesToInt(bytes.slice(pos,pos+2),1);
    pos = pos+2;
}
if(((bytes[0] >> 5) & 1)===1){ //PRESSURE
    decoded.press=bytesToInt(bytes.slice(pos,pos+2),100);
    pos = pos+2;
}
if(bytes.length>(pos+1)){
    var set1 = bytes[pos++];
    if(((set1 >> 1) & 1)===1){ //LEAF
        decoded.leafHum=bytesToInt(bytes.slice(pos,pos+2),100);
        pos = pos+2;
        decoded.leafTemp=bytesToSignedInt(bytes.slice(pos,pos+2),100);
        pos = pos+2;
    }
    if(((set1 >> 2) & 1)===1){ //ADC
        decoded.adc=bytesToFloat32(bytes.slice(pos,pos+4));
        pos = pos+4;
    }
    if(((set1 >> 3) & 1)===1){ //WIND
        decoded.windDir = bytes[pos++];
        decoded.windSpeed=bytesToFloat32(bytes.slice(pos,pos+4));
        pos = pos+4;
    }
    if(((set1 >> 4) & 1)===1){ //SCALE
        decoded.scale=bytesToInt(bytes.slice(pos,pos+4),10);
        pos = pos+4;
    }
}
```



```
}  
return {  
    data: decoded,  
    warnings: [],  
    errors: []  
};  
}
```

### JAVA PAYLOAD DECODER (OLD)

```
private byte[] hexStrToByte(String sourceHex){  
    byte[] b = new byte[sourceHex.length() / 2];  
    for (int i = 0; i < b.length; i++) {  
        int index = i * 2;  
        int v = Integer.parseInt(sourceHex.substring(index,  
index + 2), 16);  
        b[i] = (byte) v;  
    }  
    return b;  
}  
  
private double bytesToVal(byte[] bytes, int start, double dev)  
{  
    int value = ((bytes[start] & 0xFF) << 8) | (bytes[start+1]  
& 0xFF);  
    return value/dev;  
}  
  
private double bytesTo32Val(byte[] bytes, int start, double  
dev) {  
    int value = bytes[start] << 24;  
    value = value | bytes[start+1] << 16;  
    value = value | bytes[start+2] << 8;  
    value = value | bytes[start+3];  
    return value/dev;  
}  
  
private double bytesToSignedVal(byte[] bytes, int start,  
double dev) {  
    int value = bytes[start] << 8;  
    value = value | bytes[start+1];  
    return value/dev;  
}  
  
public DecodedObj Decoder(String sourceHex) {  
    byte[] bytes = hexStrToByte(sourceHex);  
    System.out.println(Arrays.toString(bytes));  
    DecodedObj decoded = new DecodedObj(); //return object  
    byte setup = bytes[0];  
    Integer pos = 1;
```



```
    decoded.valve=((bytes[0] >> 7) & 1);
    decoded.leak=((bytes[0] >> 6) & 1);
    decoded.bat = bytes[pos++];
    if((setup >> 0) & 1)==1){ //1st SOIL
        decoded.e25=bytesToVal (bytes,pos,100.0);
        pos = pos+2;
        decoded.ec=bytesToVal (bytes,pos,10.0);
        pos = pos+2;
        decoded.temp=bytesToSignedVal (bytes,pos,100.0);
        pos = pos+2;
        decoded.vwc=bytesToVal (bytes,pos,1.0);
        pos = pos+2;
    }
    if((setup >> 1) & 1)==1){ //BME280
        decoded.airTemp=bytesToSignedVal (bytes,pos,100.0);
        pos = pos+2;
        decoded.airHum=bytesToVal (bytes,pos,100.0);
        pos = pos+2;
        decoded.airPres=bytesToVal (bytes,pos,1)+50000.0;
        pos = pos+2;
    }
    if((setup >> 2) & 1)==1){ //OPT3001
        decoded.lux=bytesTo32Val (bytes,pos,100.0);
        pos = pos+4;
    }
    if((setup >> 4) & 1)==1){ //PULSE
        decoded.pulse=bytesTo32Val (bytes,pos,1.0);
        pos = pos+4;
    } //
    if((setup >> 0) & 1)==3){ //2nd SOIL
        decoded.e25=bytesToVal (bytes,pos,100.0);
        pos = pos+2;
        decoded.ec=bytesToVal (bytes,pos,10.0);
        pos = pos+2;
        decoded.temp=bytesToSignedVal (bytes,pos,100.0);
        pos = pos+2;
        decoded.vwc=bytesToVal (bytes,pos,1.0);
        pos = pos+2;
    }

    return decoded;
}

return decoded;
}
```