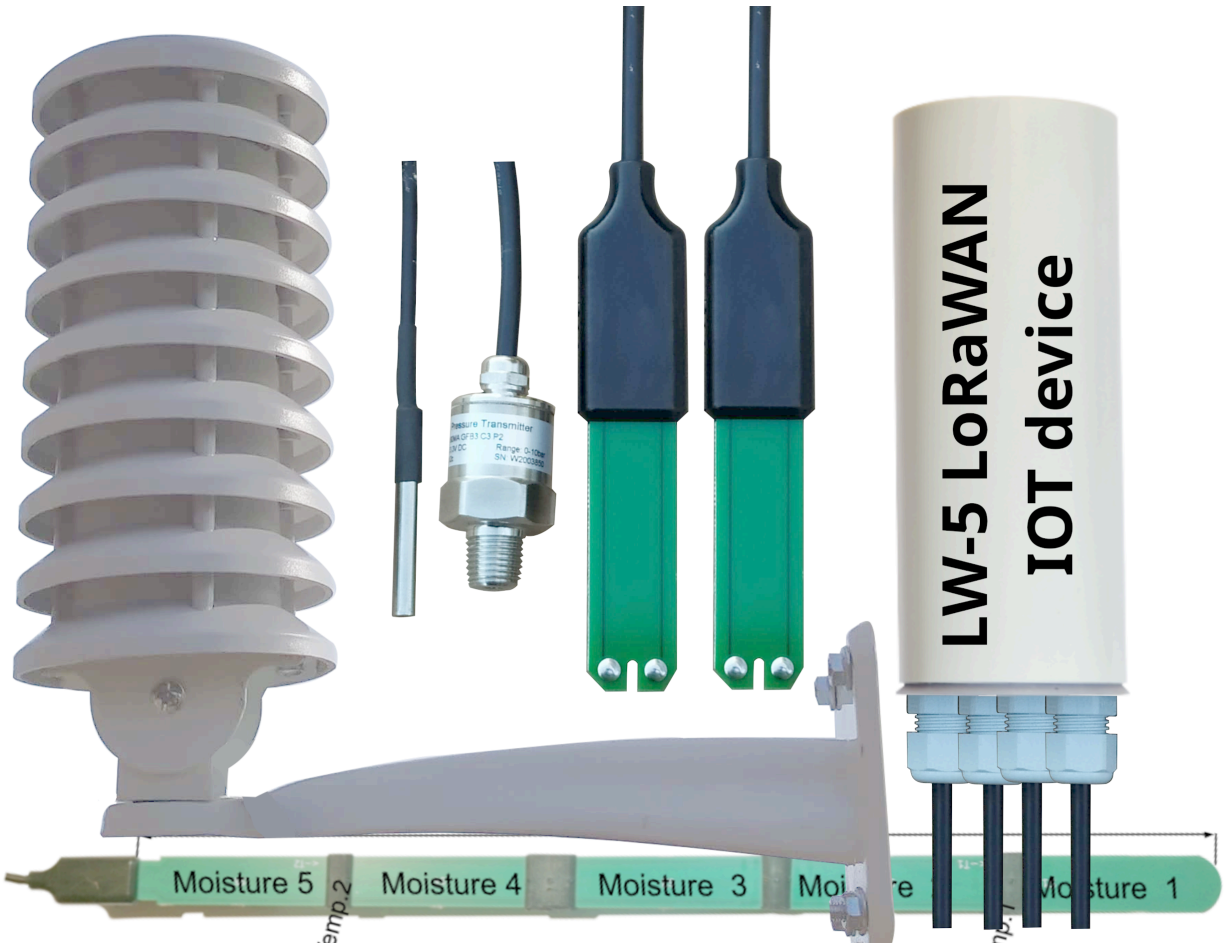
FEATURES

- LoRaWAN v 1.0.3 class A device, may be activated with OTAA or ABP.
- Setup via AT UART or Bluetooth to Android adapter.
- Android application for Bluetooth 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 SHT Air Temperature/Humidity
- Optional 12v booster pulse solenoid reverse switch output option
- Leaf moisture sensor



ELECTRICAL PROPERTIES

	Min/Sleep	Typical	Max
Supply voltage (VCC), V	3	4.1	5
Working current（VCC=3.6V）, mA	Sleep 0.004mA	22	150
Operating Temperature Range, Celsius	-20	25	70

Sensor sleep mode consumes 4uA. 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.

PHYSICAL PROPERTIES

dimensions 80x130x35
Soil Sensors cable length 1.5m (5m optional)

SOIL SENSOR MEASUREMENT PROPERTIES

Soil Sensor	**Resolution**	**Range /avg Tolerance**
Dielectric permittivity (ε) (Temperature corrected)	0.1ε	1 (air) to 80 (water) /5%
Volumetric water content - VWC calculation from Dielectric permittivity ε. VWC = 0.002974 * pow(ε,2) + 0.07424 * ε -1.295;		
Beta version EC, Electrical Conductivity (mS/m) **(0.1 mS/m = 1 uS/cm)**	0.1 mS/m	0…300 mS/m 20% 300…800 mS/m 40%
Temperature (°C)	0.1°C	-20 to 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 senso
    e. Bit 4 - enable pulse counter
    f. Bit 5 - eble 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*(*battery reading). for example if battery measurement shows 50% ,then actual voltage is 2.1+*0.022**50*=3.5v 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 (ε) (Temperature corrected) resolution: 0.1ε (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): $\theta = 4.3 \cdot 10{-}6\varepsilon 3 - 0.00055\varepsilon 2 + 0.0292\varepsilon - 0.053)$

--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 (ε) (Temperature corrected) resolution: 0.1ε (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 (°C) resolution: 0.1°C and range: -20 to 60°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): $\theta = 4.3 \cdot 10{-}6\varepsilon 3 - 0.00055\varepsilon 2 + 0.0292\varepsilon - 0.053)$

--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)

COMMAND LINE

LoRaWAN configuration may be done via serial TTL adapter connected to the board pin headers or using plugin Bluetooth BLE module connection to android application. Serial communication speed is 9600

COMMAND LINE DEVICE CONFIGURATION COMMANDS

AT? to list all available functions
AT+<CMD>? : Help on <CMD>
AT+<CMD> : Run <CMD>
AT+<CMD>=<value> : Set the value
AT+<CMD>=? : Get the value
ATZ Trig a MCU reset
AT+VL=<Level><CR>. Set the Verbose Level=[0:Off .. 3:High]
AT+APPEUI=<XX:XX:XX:XX:XX:XX:XX:XX><CR>. Get or Set the App Eui
AT+NWKKEY=<XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX><CR>: Get or Set the Network Key
AT+APPKEY=<XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX><CR>: Get or Set the Application Key
AT+NWKSKEY=<XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX><CR>: Get or Set the Network Session Key
AT+APPSKEY=<XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX><CR>: Get or Set the Application Session Key
AT+DADDR=<XXXXXXXX><CR>. Get or Set the Device address
AT+DEUI=<XX:XX:XX:XX:XX:XX:XX:XX><CR>. Get or Set the Device EUI
AT+NWKID=<NwkID><CR>. Get or Set the Network ID=[0..127]
AT+JOIN=<Mode><CR>. Joinnetwork with Mode=[0:ABP, 1:OTAA]
AT+LINKC. Piggyback a Link Check Request to the next uplink
AT+BAT Get the battery Level in mV
AT+int=<XXXX> set sleep time seconds
AT+stime=<XXXX> set sensor on time millis
AT+air=0 calibrate soil sensor in air <sensor id 0,1,2-multi>
AT+water=0 calibrate soil sensor in water <sensor id 0,1,2-multi>
AT+ec=0:1400 calibrate soil sensor EC <sensor id 0,1,2> <uS EC val>
AT+rescal=2 reset calibrate soil sensor <sensor id 0,1,2-multi>
AT+respulse reset pulse counter
AT+set=0:32 enter settings bytes 0 0
AT+jtype=<X> joint type 0 - none 1-ABP 2-OTAA 3-Local Gateway
AT+csv read settings
AT+lora read lora settings
AT+defaults reset default settings
AT+confirmed uplink confirmed=1 ; unconfirmed=0

DOWNLINK MESSAGE

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

1. 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.
2. 0x02 + data config byte1 :
    a. Bit 0 - enable Tinovi PM-WCS-3-I2C Soil sensor
    b. Bit 1 - enable SHT air temp - humidity sensor
    c. Bit 2 - enable OPT3001 light lux sensor
    d. Bit 3 - 2nd soil sensor

e. Bit 4 - enable pulse counter
f. Bit 5 - reserved
g. 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 Scale sensor
n. Bit 5 - enable Soil-Multi-5 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. 0x08 +4-byte pulse counter offset
7. 0x09 +1-byte water leak valve OFF trigger enable =1 , disable =0
8. 0x0A +1-byte sensor number 0 or 1 - if soil moisture present - calibrate AIR
9. 0x0B +1-byte sensor number 0 or 1 - if soil moisture present - calibrate WATER
10. 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.
11. 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)

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

SETUP

Sensors are already factory calibrated, but in case needed they may be recalibrated using Android to Bluetooth as described for device specific commands.
Hold sensor in air and issue command, response OK.

AT+air=0 calibrate soil sensor in air <sensor id 0,1,2-multi>

Put sensor in to water and issue command, response OK.

AT+water=0 calibrate soil sensor in water <sensor id 0,1,2-multi>

Put sensor into known esc solution and issue command, response OK

AT+ec=0:1400 calibrate soil sensor EC <sensor id 0,1,2>:<uS EC val>

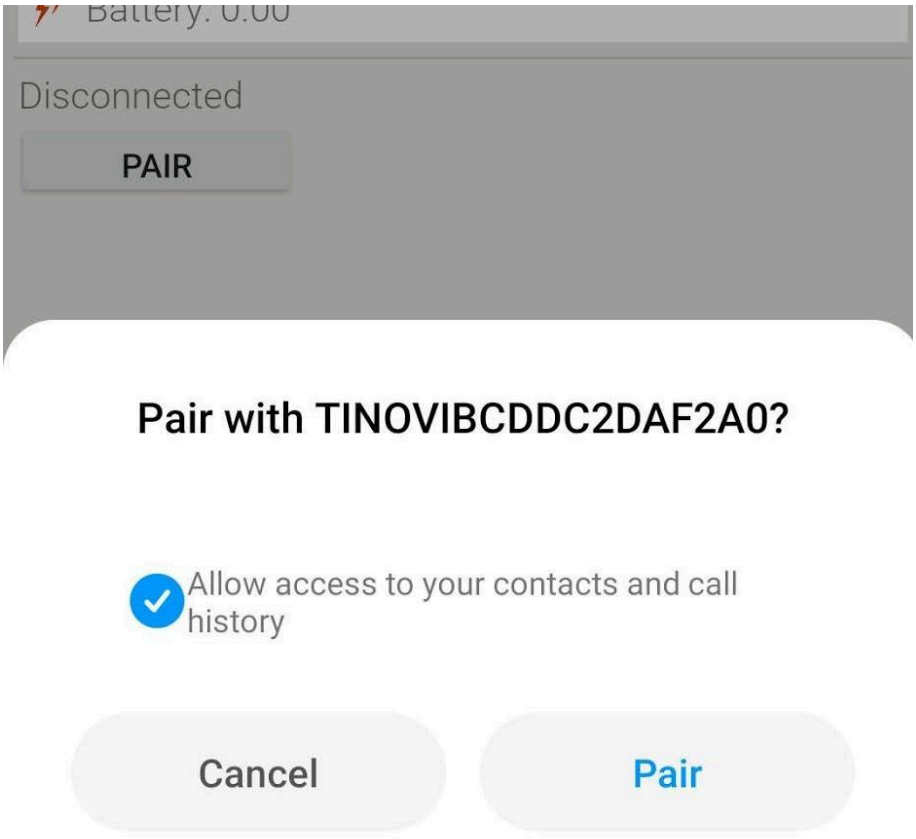
We recommend Android mobile application for sensor setup and data reading/storage

SOIL SENSOR CALIBRATION ON ANDROID APP

[Tinovi LW Configurator - Apps on Google Play]

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

1. Your phone should support Bluetooth..
2. Turn on Bluetooth on your phone
3. Connect device setup - programming module
4. Ensure there are batteries plugged in to your device
5. Restart device by pressing restart button on device main board or removing and replacing battery.
6. Press the pair button and choose TNXXXXXXXXXXXXX device (pair with device, if adding for the first time).



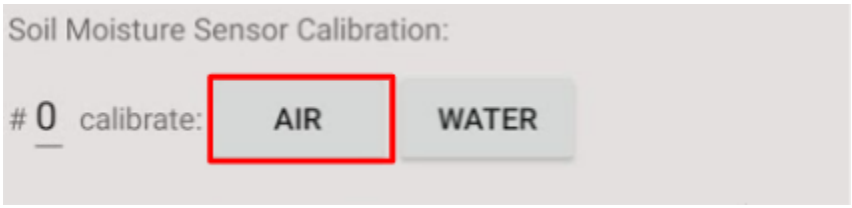
---

7. Wait up to 5-10 second while application shows Connected TNXXXXXXXXXX in main page, if no connection, repeat restart device and pair operation. Device may be restarted by removing, replacing battery or clicking the button in PCB board left bottom corner, when battery looks upwards.

CALIBRATION

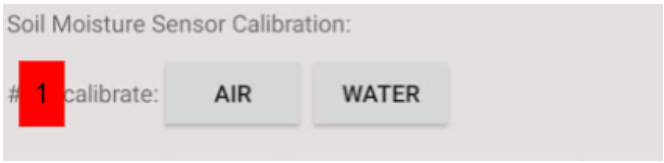
1. go to SETTINGS tab,

2. 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

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



1. Go to MAIN tab click READ to see values after calibration

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;
  }
  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;
  }
  if(((set1 >> 5) & 1)===1){ //BIT1_SOIL_MULTI
    var mult_v=bytes[pos++];//ver
    for(var c=0;c < 6;c++){
    var dp = bytesToInt(bytes.slice(pos,pos+2),100);
    decoded["dp"+(c+1)] = dp;
    pos = pos+2;
    decoded["wvc"+(c+1)] = Math.round((4.3e-6 * Math.pow(dp,3) -
5.5e-4*Math.pow(dp,2) + 2.92e-2*dp - 5.3e-2)*10000)/100;
```

```
    }
    for(var i = 0;i < 3;i++){
    decoded["temp"+(i+1)] = bytesToSignedInt(bytes.slice(pos,pos+2),100);
    pos = pos+2;
    }
  }
 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;
}
```