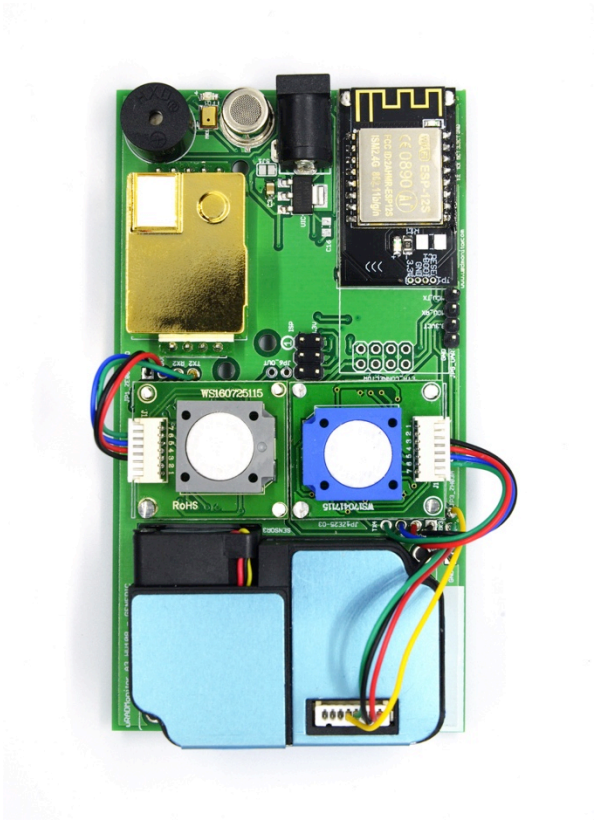

uRADMonitor HW108 motherboard

This document describes the uRADMonitor A3 data payload structure for direct data access.

Version

- The data structure corresponds to version HW108 and firmware version SW78 or newer.

Applications

- Decentralized monitoring
- Private monitoring networks
- LoRaWAN payload decoding
- Off grid monitoring

Description

uRADMonitor A3 is an automated, fixed monitoring station with an array of sensors that tracks a total of 11 important environmental parameters. The hardware version HW108 measures Temperature, Barometric pressure, Relative Humidity, Volatile organic compounds (VOC), Formaldehyde, Ozone, Particulate matter PM1, PM2.5, PM10, Carbon Dioxide, Noise level . The values are packed in a so called payload, a sequence of bytes that uses minimal bandwidth so it can be used across a multitude of networks including LoRaWAN. The payload is machine friendly, but not plain text. So using it outside the uRADMonitor Network and API requires knowing its structure in order to interpret it correctly.

This document presents the payload decoding mechanisms.

Payload Encryption

To protect the integrity of the uRADMonitor Network, the data is encrypted before it is transmitted from the uRADMonitor model A3 to the server. The encryption includes a checksum and a timestamp so it can assure protection against invalid data injection and also repeated injection attacks. The uRADMonitor Data server receives the encrypted payloads then decrypts them before making the data available via the uRADMonitor API.
For Direct Data Access, the payload encryption on the uRADMonitor A3 must first be disabled via USB commands because the decryption algorithm is not publicly available. Once the encryption is disabled, the payload can be decoded as further explained in this document, but the uRADMonitor server will no longer accept the data sent by the device.
To disable payload encryption, connect to the uRADMonitor A3 via USB (baudrate 9600bps) and send the following command: "encrypt","0" . Then check that the change is in place with a second command: "getsettings" . The returned response must show that the encrypt parameter is now 0:

Disabling the payload encryption via USB commands

The results are immediate. The encrypted data payload on the A3 HW108 and firmware version 78 or newer, **starts with 6E** and has 33bytes, while the plain payload will start with the device ID and only has 32 bytes:

6E1F9AA75641C435C39366CFA9C5D633A3EC736890074A2682E93A5072E22A09E2
Encrypted payload

820002846C4E00012E1C0A1B88BC660131298D026A00530014005500 6F009CE2
Non-encrypted payload

LoRaWAN Network Server data

The A3 HW108 / FW 78 or newer with LoRaWAN connectivity will send the payloads to the Network server. If encryption is disabled as presented above, the plain payload will be accessible directly on the Network Server. The next step is to interpret the packed data and extract all measurements.

Payload structure and decoding

The A3 HW108 / FW 78 or newer payload is a packed byte sequence optimised for low bandwidth. We will take the above plain text payload as an example:

```
82000284 6C 4E 00012E1C 0A1B 88BC 66 013129 8D 026A 0053 0014 0055 006F 009C E2
```

In order, the structure is as follows:

```
ID 4B - "82000284"
VERHW 1B - "6C" (decimal 108, meaning A3 HW108)
VERSW 1B - "4E" (decimal 78, meaning firmware version 78)
TIMESTAMP 4B - "00012E1C" (decimal 77340 sec)
TEMPERATURE 2B - “0A1B" , value 25.87°C, DI16 encoded with 2 digits **
PRESSURE 2B - "88BC" offset encoded (decimal value 35004 + 65535 = 100539 Pa) ***
HUMIDITY 1B - "66" multiplied by 2, 0.5RH resolution, decimal 102 / 2 = 51RH
VOC 3B - "013129" (decimal 78121ohm, or 78KO)
NOISE 1B - "8D" like humidity, multiplied by 2, 0.5dBA resolution, decimal 141/2 = 70.5dBA
CO2 2B - "026A" (decimal 618ppm CO2)
FORMALDEHYDE 2B - "0053" (decimal 83ppb CH2O)
OZONE 2B - "0014" (decimal 20ppb O3)
PM1 2B - "0055" (decimal 85 µg/m³)
PM2.5 2B - "006F" ( decimal 111 µg/m³)
PM10 2B - "009C" (decimal 156 µg/m³ ))
CRC 1B - “E2" ****
```

*DI24 encodes a float to a 24bit integer, with specified number of decimals. To decode, use:

```
C/C++:
double int24ToCoord(uint32_t coord, uint32_t multiplier) {
      if (coord & 0x800000)
            return (coord & 0x7FFFFF) / -(double)multiplier;
      else
            return coord / (double)multiplier;
}

PHP:
public static function uint24float($int, $multiplier) {
            $temp = hexdec($int);
            if ($temp & 0x800000) return ($temp & 0x7FFFFF) / -(double)
$multiplier;
                  else
            return $temp / (double)$multiplier;
      }
```

**DI16 encodes a float to a 16bit integer, with specified number of decimals. To decode, use:

PHP:

```
public static function uint16float($int, $multiplier) {
            $temp = hexdec($int);
            if ($temp & 0x8000) return ($temp & 0x7FFF) / -(double)
$multiplier;
                  else
            return $temp / (double)$multiplier;
      }
```

*** pressure is represented in Pascals, and the normal atmospheric values are a number close to 100000 that exceeds 2bytes to store. The normal fluctuation interval however, is a smaller value. Therefore an offset is used to decrease the number needed to store pressure in Pascals, while still being able to keep track of the ground level fluctuations correctly. An offset of 65535 is applied.

The above example translates too:
Pressure = 0x8BCF = 35791+ 65535 = 101326 Pa = 1013hPa

**** CRC is 1 byte CRC8

C/C++:

```
//CRC-8/MAXIM - based on the CRC8 formulas by Dallas/Maxim
uint8_t crc8(const uint8_t *buffer, uint8_t length) {
      uint8_t crc = 0x00;
      for (int i=0;i<length;i++) {
            crc = crc ^ *(buffer++);
            for (int j=0;j<8;j++) {
                  if (crc & 1)
                              crc = (crc >> 1) ^ 0x8C;
                  else
                              crc = crc >> 1;
            }
      }
      return crc; // not-complemented!
}
```

For a scale comparison on the numbers, you can connect via USB and send the “getdata” command to get device data and compare it to your decoding.