Connecting Household Weather Sensors to IoT World

Pavel Moravec, Michal Krumnikl, Petr Olivka, David Seidl

Connecting Household Weather Sensors to IoT World*

Pavel Moravec[1,2], Michal Krumnikl[1,2], Petr Olivka[1], and David Seidl[1]

[1] Department of Computer Science, FEECS VŠB – Technical University of Ostrava, 17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
[2] IT4Innovations National Supercomputing Center, VŠB – Technical University of Ostrava, 17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
{pavel.moravec, michal.krumnikl, petr.olivka, david.seidl}@vsb.cz

Abstract. As the Internet of Things applications become more widespread, we face the need to replace existing implementations with Internet-enabled sensors. However, these sensors often provide worse battery life, have higher costs per unit and are not backward-compatible with existing weather stations. This paper concentrates on the possibility of decoding the weather station data by a cheap device which would forward them to the Internet of Things frameworks. The basics steps of receiving and decoding of the wireless weather sensor data are discussed together with the common problems which are associated with their capture and processing.

Keywords: Wireless sensor; modulation; OOK; On-Off Keying; Internet of things; IoT

1 Introduction

Wireless weather stations with external sensors have become a part of many households, after their prices dropped to reasonable levels. Whilst the base weather station is placed inside, the sensors are battery-powered and have to withstand the outside environment. They are able to work unattended for several months or years without user interaction and forward data to the base station. Unfortunately, there is no common standard for the communication protocol and due to the limited storage of the base station only some of the measured values are kept (e.g. the maximum and minimum of measured values). Only the more expensive weather stations offer a way to connect the station to a computer, often by a serial connection and with a custom application and communication protocol. This means that a computer must be running all the time. Whilst it is possible to connect several wireless sensors to the base station (typically up to 3), they do not build a traditional sensor network by themselves, because the sensors are just simple transmitters, unlike in [10].

* This work has been supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project „IT4Innovations excellence in science – LQ1602" and from the Large Infrastructures for Research, Experimental Development and Innovations project „IT4Innovations National Supercomputing Center ? LM2015070".

In recent years, there were several papers and projects trying to receive data from wireless sensors by a PC using software-defined radio, e.g. [9], the RTL 433 project[3], reverse-engineering the serial line protocol of common weather stations with a USB or serial port and recording data on a PC or Raspberry Pi (e.g. [1], Weather Station Data Logger[4], wview[5] or fowsr [6]). There were also a few projects building a custom weather station (with wired and optional wireless sensors) e.g. the WeatherDuino Pro2 project[7], WxShield[8], or the SparkFun Weather Shield[9]. Some resources describing the serial protocols of more expensive weather stations may be found, e.g. [7].

In this paper, we will concentrate on the preliminaries of building a wireless weather sensor receiver, which would pre-process the data and send them to a one of the *Internet of Things* [5] (*IoT*) enabled platforms. We will discuss several weather stations which use 434 MHz band and analyze the modulation, encoding and data frame structure. It should be mentioned that some wireless sensors use a different frequency range, typically 868 MHz or in case of the US 915 MHz, which would require a drop-in replacement or secondary receiver.

The rest of the paper is organized as follows: In next section, we will discuss the basics of communication with wireless sensors in 434 MHz range and their specifics. In section 3 we will discuss the modulation techniques and encoding schemes. In section 4 we will address some of the common protocol features. In section 5 we will disseminate actual protocols we have managed to capture and provide a description of common frame formats. In section 6, we will mention our prototype solution, concentrating on cheap and available components, designing the device so that its cost does not exceed € 10 (not considering the router to which it may be connected).

2 Wireless Weather Stations

The weather stations have been used for several tens or hundreds of years. With the advent of electronic measurement, personal weather stations operated by private individuals became cheaper and more widespread. They typically measure at least temperature, atmospheric pressure and humidity, more advanced ones also contain an anemometer to measure the wind speed and direction (also the wind gust measurement may not record really short wind gusts) and rain gauge. Some personal weather stations may also measure solar radiation, UV index or soil moisture. However, most of the cheaper models cannot be connected to personal computers or Internet.

Most of the current wireless personal weather station systems presently consist of a base station placed indoors, which measures internal temperature and humidity as well as the atmospheric pressure, the changes of which are being recorded and used for the prediction of the weather in following 24-48 hours. The base stations may also

[3] https://github.com/merbanan/rtl_433
[4] http://wmrx00.sourceforge.net/
[5] http://www.wviewweather.com/
[6] https://github.com/ajauberg/fowsr
[7] http://goo.gl/3InBIq
[8] http://www.osengr.org/WxShield/Web/WxShield.shtml
[9] https://www.sparkfun.com/products/12081

provide additional functions – local time (sometimes radio-controlled clock adjustment is possible), alarm(s), moon phase and sunrise/sunset times. Since the cheaper base stations do not offer a possibility to access data stored on them from a computer, our solution will have to replace the base station as well.

The wireless sensors are placed outside, they have to withstand harsher environmental conditions and they must co-exist together with other systems which are using the same frequency band. Typically, the base stations support up to 3 wireless temperature (or based on the type of the weather station either the temperature and humidity or temperature and soil moisture) sensors, one rain gauge and one anemometer.

As we have already stated in the Introduction, there are many incompatible protocols for data transmission from the wireless sensor to the weather station. We can compare this situation to the issues some of the Bluetooth Smart accessory is currently facing with incompatible protocols for devices which do not have a common standardized profile.

On one hand, this leads to compatibility issues, on the other hand, it should be possible to operate several sets of wireless sensors and wireless weather stations from different vendors independently in the same area. The wireless sensors use one of the free ISM (Industrial, Science, Medicine) bands for the data transmissions. In central Europe, they mainly operate in the 433,92 MHz band, which they share with some remote control devices such as car keys, garage door openers, wireless doorbells and remotely-controlled electric sockets. Alternatively, some manufacturers use the 868 MHz band, which however starts to be extensively used by home monitoring devices and Internet of Things technologies (e.g. SIGFOX). In some countries, especially in North America, the 915 MHz band is also used (but it is not possible to use it in Europe due to the collision with GSM-900 frequency range).

3 Modulation Techniques

There are many modulation techniques used in wireless transfers for binary data, which are being used nowadays. Ranging from the On-off keying to complex modulation schemes using 256-QAM (Quadrature Amplitude Modulation), OFDM (Orthogonal Frequency Division Multiplexing) and guard intervals. However, in the case of wireless external sensors, we typically encounter simple modulation techniques, which require a less complex hardware, cheaper circuity and less power consumption.

The circuity of the wireless meteorological sensors typically uses the *On-Off Keying* (*OOK*) – a simple variant of *Amplitude Shift Keying* (*ASK*) – which represents the transmitted binary data as a presence of the carrier wave. This modulation technique is similar to the old Morse code transfers, in the basic case representing binary 1 as the presence of the carrier wave and binary 0 as the absence of the carrier wave. Whilst this technique is very easy to implement, it is typically not used just by itself, because the noise and collisions of multiple transmissions would be hard to filter out.

A smaller number of devices (e.g. Fine Offset WH1080 Weather Station sensors) use *Frequency Shift Keying* (*FSK*) or Gaussian GFSK by using two frequencies for representing binary 0 and 1. There is also a possibility of future sensors using the *Fre-*

quency Hopping Spread Spectrum (*FHSS*). Decoding data from such sensors would require the knowledge of the hopping patterns and is out of scope of this paper.

So far, we have identified following modulation techniques used in combination with the OOK variant of amplitude modulation:

1. *Pulse Width Modulation* (*PWM*) – the logical zeros and ones are defined by the actual width of the pulse generated by the transmitter
2. *Pulse Distance Modulation* (*PSM*) – the short pulses have a constant duration and the delay (or distance, space) between them defines logical zeros and ones. Sometimes, the tertiary duration is also being used as a special symbol for signaling (Start of frame, spacer, etc.) or synchronization pulses are sent, to signal the start of the frame.
3. *Return to Zero Encoding* (*RZ*) – a specific variant of return to zero is being used by some transmitters (since standard RZ would require three signal levels). In the middle of the bit interval, the signal is always turned off [2].
4. *Manchester Encoding* – the demodulated signal level is shifted in the middle of the bit interval based on the transmitted symbol – binary 0 is encoded as a shift from high to low and binary 1 as a shift from low to high or vice versa in some implementations. If the level in the beginning of the bit interval is already in the level to which it would be shifted, additional shift in the beginning of the bit interval occurs.
5. Modified *Differential Manchester Encoding* – *u*ses level shift in the beginning of each bit interval for each bit and the presence or absence of level shift in the middle of the bit interval to encode binary 0 and 1.

In the rest of the paper, we will be concentrating on the devices which use the On-Off Keying, because they require a cheaper receiver module and there are many ready-made receivers available. Automatic detection of used modulation is also possible, see [3,6,13].

4 Weather Station Data Transmission

In this section, we will describe some common frame formats used to transmit data from a wireless sensor to the weather station.

The wireless sensor circuity is generally as simple and as cheap as possible. To save money, the sensor contains only a transmitter (unlike the wireless sensor network nodes, which provide both transmitter and receiver). As a result, the measured data transmissions are unacknowledged, and it is not possible to sense multiple access on medium, i.e. methods such as CSMA (carrier sense multiple access) are out of question. To improve a chance of successful data delivery, the frame – which is quite short, consisting of only a few nibbles[10] or bytes – is usually repeated several times during the transmission.

Two good sources of this information are available on-line [2,12]. Whilst [12] contains incomplete information, it is a good starting point for further analysis. In fact, we

[10] A nibble consists of 4 bits, and can be represented as a hexadecimal digit.

have encountered a clone of such station when analyzing the data. We independently deciphered the data frame structure finding some extra information which was not part of the original document, as shown in following section.

For all weather stations mentioned in the following text, we were able to identify a set of parameters which play an important role for correct decoding of transmitted data besides the modulation scheme used, which was mentioned in previous section:

1. General transmission speed based on the pulse width and distance. Since some of the line codes do not use same length of time intervals for binary 0 and binary 1, we will rather speak about the on and off pulse lengths for individual symbols in the rest of the paper and not the bit speed.
2. *Bit ordering* of the transmitted data, both for the whole frame and for the bits in individual nibbles (4-bit sequences). There are two ways how to transmit data – the first one sends the least significant bit of a nibble (*LSB*) first, whilst the second sends the most significant bit first (*MSB*). From the global point of view, we may also send either the most or the least significant nibble first.
3. *Preamble format* differs for different implementations. The preamble is placed before the frame to synchronize the transmitter and the receiver. It is often followed by one or more synchronization bits.
4. *Number of synchronization bits* (*sync bits*) which are often placed between the preamble and the first frame and between individual frame repetitions.
5. *Frame check sequence* (*FCS*) is generally transmitted after the data frame and it is checked before the frame is processed further. However, some transmitters do not use FCS at all, other include a simple FCS inside of data portion of the frame and other combine both variants. In case of simple implementations, it is possible to determine the checksumming algorithm and verify that the data in the frame appears intact. However, if the formula is not possible to determine, we may have to use other ways how to check for data validity.
6. *Number of bits in a single frame* is the last but not least indicator which helps us to detect if the whole frame has been received. Some sensors may even stop the transmission in the middle of the last frame repetition.

The data encoded in the data frame will usually include the random sensor IDs, the channel ID for temperature and humidity measurements, battery state information, a simple abovementioned checksum and sometimes an indication that a measurement was forced manually (e.g. by pressing a button). The temperature and humidity sensors – which will be mostly discussed in this paper – will also provide the temperature (3-4 nibbles) and humidity (typically 2 nibbles) measurements, and sometimes the temperature trend as well.

The measured values stored in the data frame may also use different representations of measured data, sometimes even in the same frame. The values may be stored by individual digits (*binary-coded decimal – BCD*) or a signed or unsigned binary value. The signed values may encode the numbers with a sign bit (typically the most significant bit), or use either one's complement or two's complement representation.

In case of values with decimal places, the value is multiplied by some coefficient (e.g. 10, 4, 2) and sent as an integer. So far, we have not identified a sensor, which would use floating-point number representation, possibly due to the measured ranges

Fig. 1. The wireless sensors used for the evaluation (left-to right): H13716A-TX, Z31130-TX and Z31915-TX

not requiring values which would mandate the mantissa and exponent approach, or the problematic (or missing) implementation of floating-point operations on such simple hardware.

5 Decoding Captured Sensor Data

In this section, we will discuss data captured from several different weather station wireless sensors, which use individual protocols. We will also describe how to decode them.

Three weather stations were available, the data from the fourth were decoded its sensor transmissions have been captured "in the wild".

The first sensor has been manufactured in 2008, contains an LCD display showing temperature and humidity values and has a type number H13716A-TX placed inside on the PCB. After decoding its data, we found document [12], which describes the same protocol (but is missing some fields). It provides both temperature and humidity data.

The second sensor has been manufactured in early 2013, it does not have an LCD display. The sensor has a type number Z31130-TX on a sticker outside and HQ-TX001 (MB) R-1 printed inside on the PCB. It also provides both temperature and humidity data.

The third sensor has been manufactured in late 2013 or early 2014. It does not have an LCD display, either. The sensor has a type number Z31915-TX on a sticker outside and TX06K-THC V2 printed inside on the PCB. It also provides both temperature and humidity data.

The last sensor has not been physically available to us, but it provides only the temperature data. We will label it UWTT (unknow wireless temperature transmitter) in the following text.

The weather stations using the second and third sensor look almost the same, they were sold under the same brand and use the same set of functions and graphics on LCD screen. One would expect that the sensors use same or similar protocols, but as we will show later, this is not the case.

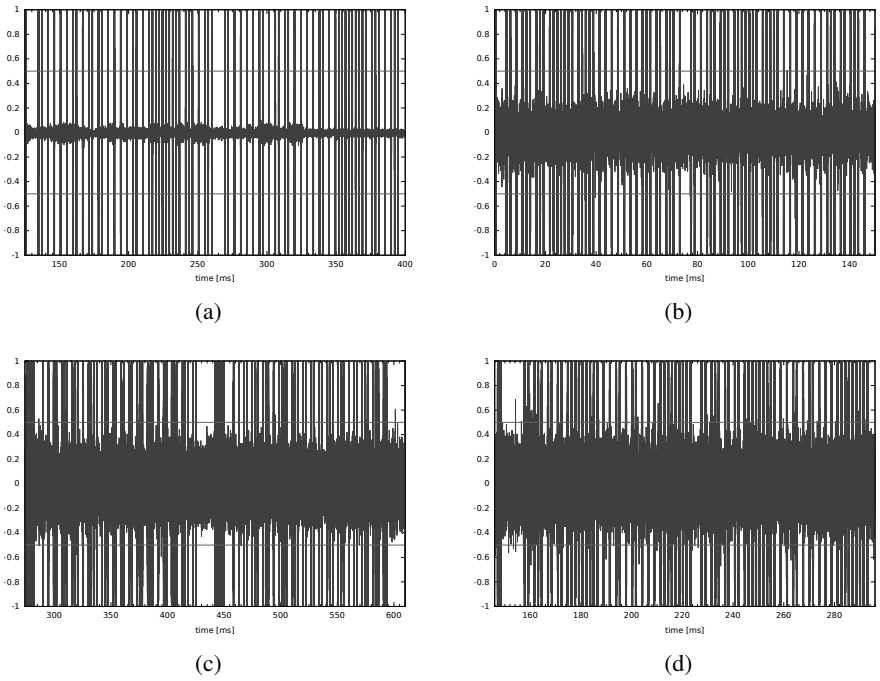

Fig. 2. Captured analog signal (I component) – first two complete frame repetitions: (a) H13716A-TX, (b) Z31130-TX (c) Z31915-TX (d) UWTT

In Fig. 2, we can see the I component of the analog signal, normalized to $\langle -1, 1 \rangle$ range. The samples for individual sensors were taken over a longer period of time, so the actual measured values will differ from each other. In the following text, we will address the individual steps needed for decoding of the signal.

5.1 Decoding the Demodulated Signal

The samples provided in Fig. 2 show, that all sensors in our evaluation use Pulse Distance Modulation and tri-state encoding (0, 1, space between the frames – S). The receiver circuity will demodulate the signal and provide either an analog signal, based on the input voltage, which would correspond to the non-negative part of examples shown in Fig. 2, or digital TTL levels represented by 0 or 1 values.

When processing such signal, we have to determine the pulse lengths and distances first. This may be tried semi-automatically, which will work for many receivers, but the auto-detection process may be thrown off e.g. by the synchronization bits, which can be seen in some of the measured data samples see Fig. 2(c),(d).

After successful detection of pulses, we may decode the individual bits and detect the frame length, which should be the same for all frames in the single transmission,

Table 1. Approximate timing of sensor transmissions (changes with temperature and sensor)

Sensor	Pulse	0 delay	1 delay	S delay	Sync bit On/Off	Synchronization pattern
H13716A-TX	0.4 ms	2 ms	4 ms	9 ms	–	S bit
Z31130-TX	0.5 ms	1 ms	2 ms	4 ms	–	S bit
Z31915-TX	0.75 ms	1.75 ms	4 ms	7.75 ms	1/1 ms	(2×S delay) + 4 sync bits + S bit
UWTT	0.5 ms	1 ms	2 ms	3.75 ms	0.5/0.5 ms	1 sync bit + S bit/S bit[11]

creating the super-frame. We should, however, note, that the timings are far from being exact. The actual timing will depend on outside temperature and the oscillator used in the wireless sensor circuity. As a result we may define thresholds, ignoring pulses and delays shorter than the initial delay and sampling 0, 1 and S symbols based on additional thresholds. On one hand, it is a problem, since we will not be able to demonstrate decoding of other signals, on the other hand it shows that this type of modulation is quite common for this type of sensors.

The result of this step will be a bit vector with the data to be processed together with meta-information which may be used for correct selection of the decoder (there may be several candidates based on the timing, e.g. Z31130-TX and UWTT use the same timing and the only difference is the presence of sync bit before S bit in UWTT).

5.2 Detecting Parts of Decoded Frame

The results of the previous step are processed and the measured data and additional information is obtained. To do so, we must first determine which sensor is being used to obtain correct frame format and interpretation. This may be done directly on the device which captures data, or the frame may be submitted to the server for further processing.

The decoded data is shown in Table 2, which provides both basic variants of data organization: LSB and MSB for the whole frame. We have marked the identifiable temperature data in bold, channel IDs in italics and humidity data in bold italics. We could also provide data reordered by individual bytes/nibbles, or bit-inverted data (in the case when a manufacturer would decide to use longer delay for 0 and shorter for 1), but it was not necessary in this case.

To find which parts of the frame/bits represent the measured values, it is best if we have access to the base station which should display the transmitted value (or a value close to it) on screen. The temperature is typically multiplied by 10 and stored as an

[11] First frame in sequence has the delay after sync bit (2 × S delay), the repetitions do not have a sync bit.

Table 2. Basic frame information and decoded data (MSB and LSB first)

Sensor	Frame	MSB first data	LSB first data	Temperature	Humidity	Channel
H13716A-TX	36 bits	0x7DAFB004C	0x3***20***0**DF**5BE	22.3 °C (**0x0DF**)	***20*** (0x14)	3
Z31130-TX	36 bits	0x648**10F**F***29***	0x94FF08126	27.1 °C (**0x10F**)	41 (***0x29***)	1
Z31915-TX	40 bits	0xD262552***69***2	0x4964AA464B	7.8 °C (0x04E)	***69*** (0x45)	*2*
UWTT	36 bits	0x4A8**076**F00	0x00F6E0152	11-13 °C	N/A	N/A

12-bit integer, so we may use its value to find the position in the frame, if available. The same way, we may look for humidity, which will be typically a 8-bit value (unless decimal places are used), channel number, or some other type of measured value which can be displayed on screen of the base station (wind speed and gust, total rainfall, etc.). The transmitted data may be further analyzed for low battery data, button pushed indication, etc.

In our case, we were directly able to identify the temperature and humidity positions for H13716A-TX, which uses LSB ordering, 12-bit temperature value and BCD-encoded humidity value. The Z31130-TX also provides both the temperature and humidity values, but in MSB order – both are hexadecimal values.

We may also expect, that UWTT temperature was 11.8 °C and the transmitter uses MSB ordering, however more measurements had to be be gathered to make sure of our calculation.

5.3 Interpreting the Data Fields

Even if we were successful in identifying the data fields, at least one measurement with a negative temperature should be also made in most cases to detect the signed value representation used for temperature measurement. Example of such data is shown in Table 3. Other samples with different channel ID settings should provide the place of the channel number and battery low/high indication.

However, we have encountered a problem – the Z31915-TX did not provide any meaningful values for the temperature measurement. On the other hand, we were able to identify that data uses the MSB first order and find the positions of channel number and humidity value.

After gathering tens of samples and recording the displayed temperature, the trend started to appear. We were able to find the position of temperature measurement and determine, that the values follow a formula similar to Fahrenheit to Celsius conversion ($y = (x - 32) \cdot \frac{5}{9}$, where x is the temperature in Fahrenheit and y in Celsius).

There were however two problems – the value had still to be divided by 10 and the basic value did not match. Finally, we were able to find the correct formula for

Table 3. Decoded data – subzero temperatures

Sensor	Data (in MSB format)	Temperature	Signed value	Channel	ID	Battery
H13716A-TX	0xD20**FF35*A0***	-1.3°C (**0xFF3**)	2-complement	1	***0xA0***	Low
Z31130-TX	0x***7A***9**FE5**F14	-2.7°C (**0xFE5**)	2-complement	*2*	***0x7A***	OK
Z31915-TX	0x***8B***624A838*3*	-1.5°C	N/A	*3*	***0x8B***	N/A

temperature calculation, which was (due to integer-based calculations prior the final step defined as follows):

$$y = \frac{\lfloor (z - 1220) \cdot \frac{5}{9} \rfloor}{10} \quad [^\circ\mathrm{C}]. \tag{1}$$

The z stands for the measured value obtained from received frame and y for the temperature in Celsius.

All four evaluated sensors had a floating ID, which has been changed each time the batteries have been replaced (it means that a new scan for sensors has to be executed after battery change). The random ID allows the coexistence of different sensors with the same protocol and same channel number (or without a support for channel numbers) and ensures that the stored base values have to be reset after sensor reinitialization (e.g. for rainfall gauge). For all four sensors, the random ID has been recorded in the first two transmitted nibbles (first byte).

The frame check sequence (FCS) is often present, but unless a simple calculation has been used (like in the case of Z31130-TX), we may not be able to verify it. The channel number may appear in data directly, as a bit flag on given position, or as a value decreased by 1 (e.g. Z31130-TX uses 0 for channel 1).

There are a few additional observations to make. When using sensors other than temperature/humidity with the base station, the basic frame format (timing, repetitions, bit ordering, etc.) stays generally the same, but the sync bits and checksum calculation may change, as seen in [12].

5.4 Determined Frame Formats

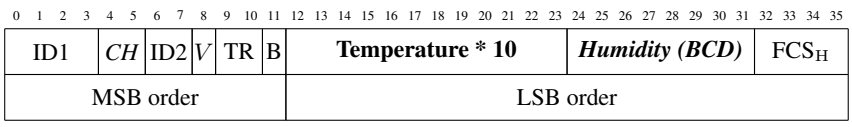

Fig. 3. Frame sent by H13716A-TX

In the following text, we will describe the final decoded frame formats. The following notation will be used: *CH* is the *channel number*, V is a bit flag indicating *low voltage*, $\overline{V}$ is a bit flag indicating that *voltage is OK*, *TR* indicates the *temperature trend* calculated from last measurements (00 – stable, 01 – increasing, 10 – decreasing), *B* indicates that a *button on the sensor has been pushed or the batteries were changed*, *FCS* is the *frame check sequence*.

Fig.3 depicts the frame format for H13716A-TX. Detailed examination has shown, that only data from fourth nibble onward are ordered in LSB order, i.e. the channel number and trend are encoded in MSB order. The FCS is calculated (in LSB order) as

$$\mathrm{FCS_H} = 0\mathrm{xf} - (\sum_{i=0}^{7} \mathrm{nibble}_i) \;\&\; 0\mathrm{xf} \tag{2}$$

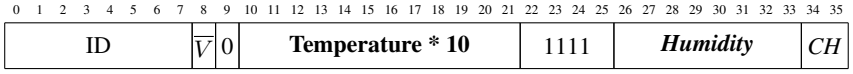

Fig. 4. Frame sent by Z31130-TX

Fig. 4 depicts the frame format for Z31130-TX. We were not able to find the FCS field, moreover the sensor is not equipped with a button, but there is a possibility that bit 9 is reserved for button press indication, but we were not able to verify this. The channel numbers start with 0 (for channel 1).

Fig. 5. Frame sent by Z31915-TX

Fig. 5 depicts the frame format for Z31130-TX. The sensor is not equipped with a button (there is a possibility that one of the bits 8-13 is reserved for button press indication, but we were not able to verify this). Also, we were not able to figure out the FCS calculation formula even with a lot of captured samples. Although it is not needed for temperature decoding, it would help us to check for erroneous frames. When we simulated the transmissions, the FCS was checked and frames with invalid FCS were dropped by the base station. However, in some cases the base station emitted a beep, so we suppose that the "button push" event may also be encoded in the, FCS but we were not able to pin it to any specific bit.

Lastly, we will describe the last weather sensor, which we were able to capture "in the wild". Fig. 6 shows the sensor data we were able to obtain from several samples. Some captures done later indicated the battery bit, they were obtained several days

before the sensor stopped transmitting. We were also able to capture a frame with bits 10-11 set to 01, so we are pretty sure that they record channel ID, its values starting with 0 for channel 1. Further, the last 3 nibbles of the frame always read 0xf00, so there is a possibility, that this is a preset reserved value for transmitter without a humidity sensor and two of these three last nibbles in the frame may be used for humidity value in other sensors.

6 Providing the Captured Data to Internet of Things Applications

In this section, we will describe the prototype solution, which has been used to send data to the home server and to the ThingsSpeak IoT API.

The prototype implementation uses an OOK/ASK 433 MHz receiver (two possible receiver modules (the smaller one using Synoxo SYN470R [11] ASK receiver) are shown in Fig. 7 in top left corner), which are connected to the orange marked connector on the prototype board which uses a widespread cheap 8-bit microcontroller ATMEGA 328P [4].

To simulate fully the base station, we have also included humidity, barometric pressure and temperature sensors. The prototype has been connected to a router running OpenWRT firmware, which reads data from the serial port emulated by the USB-to-serial converter (top right corner). In our case we have used TP-LINK TL-WR703N. We could connect the TTL levels to on-board serial port, but decided against it. It would be also possible to use a PC or Raspberry Pi instead of the router, but then we would have to solve the connectivity anyway.

The prototype has preset hard-coded parameters for individual sensors, with the possibility to replace the parameters for the first three sensors with a new custom code. There are separate configurations for both the general frame format and individual wireless sensor to make it possible to define other data types. The DIP switches in the bottom part of the prototype PCB allow us to enable/disable individual decoders (red DIP switch) and channel IDs (blue DIP switch). We have also added a LED indicating that the frame has been received, which can be disconnected (bottom right corner).

We were able to extract individual frames from the receiver and send the signal with meta-information for the processing to the router for all abovementioned wireless sensors. We are currently working on the second prototype, which uses ESP8266 SoC [8] solution with a built-in WiFi functionality, which would make the selection of wireless sensors and general configuration much easier. The device would be able to operate in a true IoT fashion, not needing the router for data preprocessing. However, the disadvantage is the increased power consumption. We are also considering the hybrid approach, with the basic decoding still done by the ATMEGA 328P processor which would pro-

Fig. 6. Frame sent by unknown wireless temperature transmitter (UWTT)

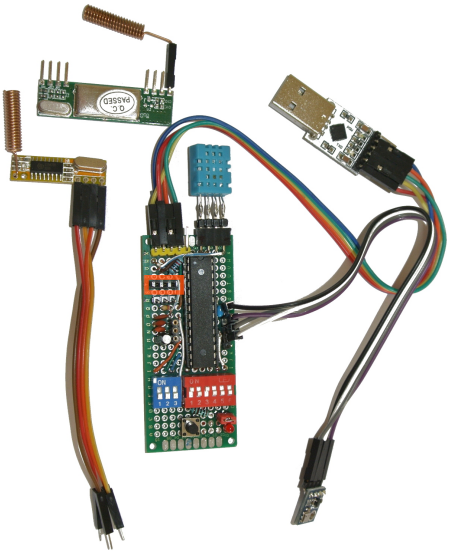

Fig. 7. Disassembled first prototype solution using ATMEGA 328P

vide the last captured codes on-demand when the ESP8266 module wakes from sleep, reducing the overall power consumption.

7 Conclusion

In this paper, we have provided a groundwork for the solution which would allow us to connect existing cheap home wireless weather sensors to the IoT world. We have discussed the ways how to extract data from their transmission, concentrating on the possible modulations, frame formats and data organization specifics. The paper discussed mainly the temperature and humidity sensors, but we can easily add other types of sensors as well. We have also provided examples of the frames both for the transmitters with known measured data and for a captured transmission with unknown values.

In future, we plan to build a custom solution, which would allow users to select the decoder from one of the pre-defined formats or auto-detect the possible modulation and encoding schemes and data formats. Some features, for example the used modulation, the pulse lengths and delays and used scheme, may be auto-detected when using software-defined radio to capture the sensor data. Also, we should be able to use the proposed solution together with the sensor placed in the same area to auto-detect the parameters of wireless temperature (and humidity) sensors by obtaining the data from built-in sensors and trying to match them to the received transmissions.

References

1. WMR918 PCLINK Protocol (2005), http://www.netsky.org/WMR/Protocol.htm
2. Oregon Scientific RF Protocol Description (2012), http://www.osengr.org/WxShield/Downloads/OregonScientific-RF-Protocols-II.pdf
3. Ahn, W., Choi, J., Park, C., Seo, B., Lee, M.: Automatic modulation classification of digital modulation signals based on gaussian mixture model. In: UBICOMM 2014 : The Eighth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies. pp. 275–280 (2014)
4. Atmel Corporation: ATmega48A/PA/88A/PA/168A/PA/328/P DATASHEET (2015)
5. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Computer networks 54(15), 2787–2805 (2010)
6. Azzouz, E., Nandi, A.: Automatic modulation recognition of communication signals. Springer Science & Business Media (2013)
7. Davis Instruments Corp.: Vantage Pro™, Vantage Pro2 ™and Vantage Vue ™Serial Communication Reference Manual (2013), http://www.davisnet.com/support/weather/download/VantageSerialProtocolDocs_v261.pdf
8. Espressif Systems IOT Team: ESP8266EX Datasheet, Version 4.3 (2015), https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266__Datasheet__EN_v4.3.pdf
9. Ferrari, P., Flammini, A., Sisinni, E.: New architecture for a wireless smart sensor based on a software-defined radio. IEEE Transactions on Instrumentation and Measurement 60(6), 2133–2141 (June 2011)
10. Mainetti, L., Patrono, L., Vilei, A.: Evolution of wireless sensor networks towards the internet of things: A survey. In: Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on. pp. 1–6 (Sept 2011)
11. Synoxo: SYN470R Datasheet (300-450MHz ASK Receiver) (2010), http://www.jmrth.com/download.asp?n=470R.pdf
12. TFD: RF transmission protocol of Auriol H13726 Ventus WS155, Hama EWS 1500, Meteoscan W155/W160 wireless weather stations v2.0 (2011), http://www.tfd.hu/tfdhu/files/wsprotocol/auriol_protocol_v20.pdf
13. Thakur, P.S., Madan, S., Madan, M.: Trends in automatic modulation classification for advanced data communication networks. International Journal of Advanced Research in Computer Engineering and Technology (IJARCET) 4, 496–507 (2015)