Implementation of a Multi-Channel DASH7 IoT Communication System for Packet Investigation and Validation

Dennis Joosens DENNIS.JOOSENS@UANTWERPEN.BE
Rafael Berkvens RAFAEL.BERKVENS@UANTWERPEN.BE
Maarten Weyn MAARTEN.WEYN@UANTWERPEN.BE
University of Antwerp - IDLab - imec, Sint-Pietersvliet 7, 2000 Antwerp, Belgium

Noori BniLam NOORI.BNILAM@ESA.INT
European Space Agency, Keplerlaan 1, 2201 AZ Noordwijk, The Netherlands

Abstract

The Internet of Things market has emerged over the past decades and continues to grow. Therefore, IoT devices have become omnipresent. Many of these devices use a wireless connection to send and receive data which are based on Low Power Wide Area Network protocols. The most used LPWAN technologies are Narrowband IoT (NB-IoT), LoRaWAN, LTE-M, and Sigfox. Due to the increasing amount of IoT devices, these LPWAN protocols have become even more important. However, many of these protocols are proprietary, and therefore, it remains unknown how they exactly operate. We dive into the inner workings of one of these LPWAN protocols. More specifically, we investigate the PHY of the DASH7 Alliance Protocol (D7AP). We present a fully-fledged DASH7 communication system by using GNU Radio. The software can be used as a simulation instrument and can be applied in real-world scenarios by using low-cost Software-Defined Radios. In this way, it is possible to investigate a complete IoT transceiver system that is open-source and easily adaptable. Furthermore, it can be used to build up, investigate, and validate DASH7 data packets.

1. Introduction

The Internet of Things (IoT) is a concept that has been evolving in the last decade to provide internet connectivity to objects that we use in everyday life. These objects are very versatile and can be generic devices such as bicycles, motors, home appliances, and devices such as temperature, humidity, air quality, or inertial measurement unit sensors. These "things" are usually equipped with extra hardware that constitutes mainly a processing unit and a transceiver for digital communications. Consequently, this hardware will enable the objects to communicate with one another or to send information to the internet (Zanella et al., 2014). The IoT concept also fosters interaction among objects to achieve a high operational efficiency (Centenaro et al., 2016; Bni Lam, 2021a). Accordingly, the IoT concept has been attracting the attention of many industries as well where automation has a key role. Nowadays, applications can be found in home and industrial automation, medical aids, energy management, automotive and smart cities.

Moreover, Low Power Wide Area Networks (LPWAN) have emerged to provide the communication means that have enabled the IoT concept. Most LPWAN technologies can provide connectivity to millions or even billions of objects. This communication revolution can be attributed to the LPWAN capability of establishing long-range communication links that go up to several kilometers while using low-power transceivers (BniLam et al., 2021b). Additionally, due to the low production cost, LPWAN transceivers are massively deployed in large-scale environments; i.e., on the scale of cities or even countries. The most commonly used LPWAN technologies today are LoRaWAN NB-IoT and LTE-M (Janssen, 2023; Sinha, 2024).

During the last decades, Software-Defined Radio (SDR) technology has also become a very popular and powerful prototyping tool (BniLam et al., 2018; 2019). Before the introduction of the SDR technology, researchers and manufacturers were only relying on custom-made expensive hardware and simulation tools. Simulation tools can be very powerful in describing any physical behavior. However, they cannot include all the physical variables that are associated with complex engineering problems. On the other hand, the SDR technology can provide a physical prototype that can be subject to testing scrutiny. In recent

Proceedings of the 14^{th} GNU Radio Conference, Copyright 2024 by the author(s).

years, the SDR technology has been used as a final product for many communication system technologies which is due to the decreasing production cost.

In this work, we make use of the SDR technology to design the physical layer for LPWAN communication systems. The design steps are generic and can be deployed with any LPWAN standard. However, we have adopted the DASH7 Alliance Protocol (D7A) (D7A, 2024). The DASH7 Alliance Protocol (D7A) originates from the ISO/IEC 18000-7 standard which was ratified by ISO in 2004. This standard has been used by the U.S. Department of Defense (DoD) for container inventory. In 2009, the DASH7 Alliance was established and repurposed to an open-source standard for bi-directional Wireless Sensor and Actuator Networks (WSAN). Nowadays the standard can be used for commercial IoT applications (Schneider, 2010). This communication standard has been selected to represent IoT communication systems because its specification is freely available (D7A, 2018), and an open-source software stack is ready to use (Sub-IoT, 2024a). The proposed SDR implementation of the DASH7 standard has been validated using simulation and experimental analyses. Furthermore, we recorded a DASH7 data set during the deterministic testing of a measurement setup.

The remainder of this paper is organized as follows. In Section 2, the PHY layer of the DASH7 Alliance Protocol is introduced. Section 3 explains the design of a DASH7 transmitter in GNU Radio while Section 4 discusses the DASH7 receiver chain. Section 5 shows the working of the full transceiver system through some experiments. Finally, Section 6 discusses the conclusions, and we have an outlook on the future.

2. The DASH7 Alliance Protocol

Although the DASH Alliance Protocol (D7A) is represented by a complete stack, to grasp what is happening to transmit or receive bits we need to unravel the ins and outs of the physical layer. A detailed overview of the full stack is shown in Figure 1.

2.1. The Physical Layer

At the PHY, the encoding types, modulation scheme, supported bands, and data rates are specified. DASH7 originally supported only the unlicensed 433 MHz ISM/SRD band; when the DASH7 Alliance adopted the protocol, the 868 MHz and 915 MHz band have been added. Each band is split into a finite amount of channels. The number of channels depends on the channel class that is used. Each channel class of DASH7 has its specific channel spacing, symbol rate, modulation index, and frequency deviation. An overview of the available channel classes and bands can

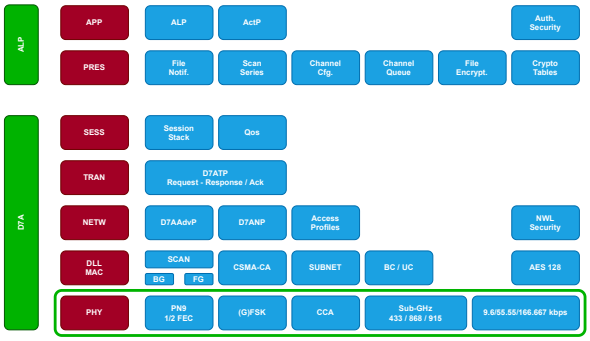

Figure 1. Overview of the full stack specification.

be found in Table 1 and Table 2.

By default, PN9 encoding is applied to the payload, which scrambles the data using a 9-bit pseudo-random number. In this way, the data gets whitened. Optionally, a $\frac{1}{2}$-Forward Error Correction encoding can be used. In this case, the data firstly is FEC encoded and afterwards, PN9 encoded. The first stage of the FEC operation is the encoding using a $\frac{1}{2}$ rate convolutional code with a constraint length of four, and the second stage is a four-by-four matrix interleaver that minimizes the impact of burst errors (Weyn et al., 2013). Afterwards, this output is PN9 encoded.

Furthermore, the PHY also defines Clear Channel Assessment (CCA). CCA determines the current state of the medium. This is a mechanism to access the channel and is used when multiple transmitters are applied in a network. However, we have not implemented this into the designed system.

To assemble a packet, we first need to know which fields define a DASH7 packet. D7A packets contain a preamble, sync word, and a payload. The preamble is used for calibrating data rate circuits and time synchronization. This sequence is a set of alternating ones and zeroes and can typically have 32 or 48 symbols. The sync word is a block of 16 binary symbols and is used to align the packet payload. The packet payload is typically PN9 encoded, and a 16-bit Cyclic Redundancy Check (CRC) is applied to it. A more in-depth analysis is explained in the following section.

2.2. Packet structure

A DASH7 packet has a default frame structure, which is shown in Figure 2. It consists of a preamble, a sync word, and a payload preceded by a power ramp-up and succeeded by a power ramp-down, which are needed to meet the band stop channel requirements (Weyn et al., 2013; D7A, 2018).

The power ramp-up is the time that is needed before the first symbol transmission. This means that the carrier fre-

Table 1. DASH7 Channel Classes specify the used channel spacing, symbol rates, modulation scheme, modulation index, and frequency deviation.

Channel Class	Channel Spacing (c) (kHz)	Symbol Rate (kbps)	Modulation Scheme	Modulation Index	Frequency Deviation (Δf) (kHz)
Lo-Rate	25	9.6	2-(G)FSK	1	$\pm$ 4.8
Normal	200	55.555	2-(G)FSK	1.8	$\pm$ 50
Hi-Rate	200	166.667	2-(G)FSK	0.5	$\pm$ 41.667

Table 2. DASH7 channel bands and their allowed channel indexes. The stars in the table indicate the following:
* Worldwide coverage with local regulatory limitations,
** EN 300 220 (Europe) with local regulatory limitations,
*** FCC part 15 in the United States of America.

RF band	Lo-Rate (d)	Normal and Hi-Rate (d)	Start (MHz) (b)	End (MHz)
433 MHz*	0, 1, ..., 68	0, 8, 16, ..., 56	433.06	434.785
868 MHz**	0, 1, ..., 279	0, 8, 16, ..., 216, 229, 239, 257, 270	863	870
915 MHz***	0, 1, ..., 1039	0, 8, 16, ..., 1032	902	928

Table 3. DASH7 sync word classes and coding schemes. Currently, only CS0 and CS2 are used. CS1 and CS3 are reserved for future use.

Sync Word Class	Coding Scheme			
	CS0	CS1	CS2	CS3
0	0xE6D0	RFU	0xF498	RFU
1	0x0B67	RFU	0x192F	RFU

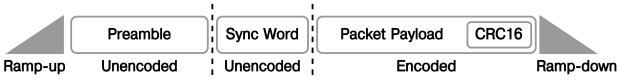

Figure 2. DASH7 frame structure.

quency is ramped from idle power to transmit power and settles to a stable state. The reverse operation occurs when ramping down. If this ramp-down is too fast, a part of the encoded frame might be affected, and therefore, the packet can become unrecoverable. Typically, these ramps have a period of 8 symbols but can go up to 32 symbols. At the Data Link Layer (DLL) of DASH7, we see two types of frames defined: background frames and foreground frames. Background frames have a static payload size of six bytes, whereas foreground frames can have a length of up to 256 bytes.

The preamble is an unencoded structure of alternating binary symbols which can have a length of up to 128 bits. This is used to settle the receiver by calibrating its data rate circuits. The DASH7 specification defines a typical preamble length of 32 bits for the Lo-Rate and Normal-Rate channel class while recommending 48 bits for the Hi-Rate channel class (Singh et al., 2020). The preamble is followed by an unencoded sync word of 16 binary symbols which is used to identify the start of the payload. Currently, the physical layer supports two sync word classes which depend on the type of frame that is used. An overview can be found in Table 3. Background frames use Sync Word Class 0 while foreground frames use Sync Word Class 1. A second categorization that makes up a specific sync word is the type of encoding that is applied to the payload. Finally, the payload field, which can have a length between 5 and 256 bytes contains at least a length byte, a subnet byte, and a control byte (CTRL), the packet data, and two CRC16/CCITT_FALSE bytes. The CRC operation is applied to the payload before the encoding process. Afterward, the payload is encoded using a PN9 scrambler or a combination of a $\frac{1}{2}$ Forward Error Correction code (FEC) and PN9 scrambler depending on the chosen coding scheme (Hoel, 2007).

3. DASH7 transmitter Design

To create a DASH7 transmitter, we need to convert our data bits to a modulated wave. For convenience, we split this entire process into four parts,

- packet assembly and data formatting
- data mapping
- symbol-to-waveform conversion
- baseband modulation

These parts will be discussed in the following sections.

3.1. Packet Assembly and Data Formatting

To assemble a DASH7 packet in GNU Radio, we start by implementing the structure as discussed in Section 2.2. The unencoded preamble is followed by an unencoded sync word. The sync word identifies the start of the payload, how it is encoded, and ultimately defines what kind of payload frame is used. Background frames use Sync Word Class 0 (SWC0) while foreground frames use Sync Word Class 1 (SWC1). Additionally, DASH7 uses two types of coding schemes. Table 3 shows the currently implemented coding schemes, i.e. Coding Scheme Zero (CS0) and Coding Scheme Two (CS2). CS0 indicates that the payload is scrambled using a 9-bit pseudo-random number generator (PN9) which is based on the polynomial $x^9 + x^5 + x^0$ (Christiansen, 2010). CS2 adds an extra step to the scrambled payload i.e. a $\frac{1}{2}$ Forward Error Correction ($\frac{1}{2}$ FEC).

In this work, we use SWC1 and applied CS0. This indicates that the payload is a foreground frame and is PN9 encoded. Finally, a 16-bit Cyclic Redundancy Check, i.e. CRC16-CCITT, is calculated on the scrambled payload, resulting in a checksum of two bytes, which are added to the scrambled payload and form the complete payload (Weyn et al., 2013). In GNU Radio, these fields are defined as decimal values using multiplexed vector sources. The payload encoding and the error detection code are handled in a separate Python module. The output of this module is added to the payload vector source.

3.2. Data Mapping

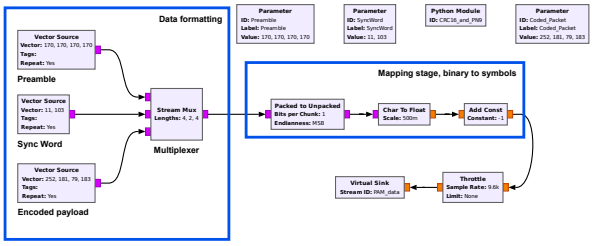

Figure 3. Packet assembly and data formatting continued with the mapping stage in GNU Radio.

When the packet is assembled, the binary message needs to be converted to a physical waveform to modulate the packet correctly. The first step is to map the bits to a constellation value. For 2-(G)FSK, the mapping consists of two possible values, namely -1 and 1. A zero value bit is represented as a -1 value, which forms the space frequency and a one value bit is represented as a value of 1 and forms the mark frequency. Figure 3 shows the data formatting and the mapping stage. The output after mapping is depicted in Figure 4.

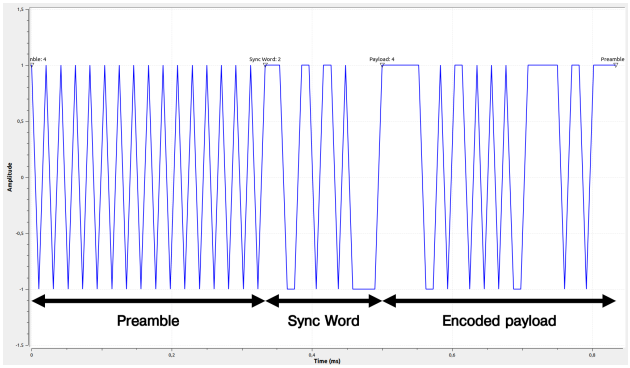

Figure 4. Output of the data after the mapping operation. The values are mapped between -1 and 1, and the preamble, sync word, and payload are detected.

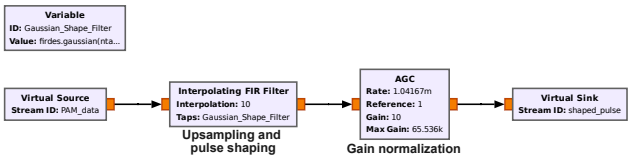

Figure 5. Symbol-to-waveform stage where the mapped data is upsampled and a Gaussian shape filtering process is applied.

3.3. Symbol-to-Waveform Conversion

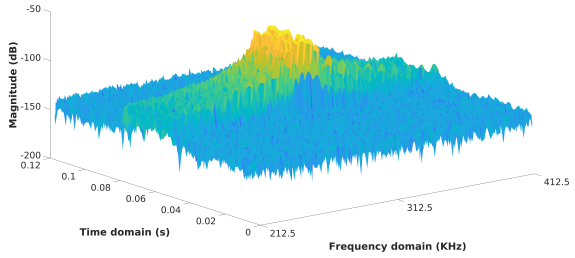

Figure 6. Recorded FSK signal when no pulse shaping is applied. The spectrum is occupied with unused sidelobes.

After creating the mapped symbols, we can start building the physical waveform. This is achieved by upsampling the symbols and applying a pulse shape filter to the upsampled signal. In this way, a time dimension or specific symbol time is created i.e. the symbols will span more samples. The symbol width and amplitude can be altered by applying a specific pulse shape filter. Although DASH7 supports FSK modulation, typically, a Gaussian pulse shape filter with a specific Bandwidth-Symbol Time product (BT) is applied to create smooth transitions between symbols in the time domain. In the frequency domain, it optimizes the power in the main lobe, decreases the occupied bandwidth, and therefore, reduces Inter-Channel Interference

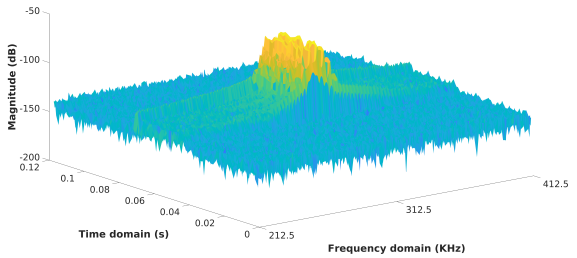

Figure 7. Recorded GFSK signal with a Bandwidth-Symbol Time product of 0.5 which is a typical value used in DASH7. The spectrum is more concentrated and cleaner.

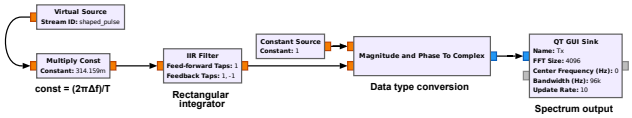

Figure 8. Baseband modulator process.

(ICI). The implementation of the process in GNU Radio is shown in Figure 5 by using an *Interpolating FIR Filter*-block and an *Automatic Gain Control*-block (AGC) which normalizes the output from the FIR filter. The effect with and without Gaussian shape filtering is depicted in Figure 6 and Figure 7 respectively.

3.4. Baseband Modulation

When our symbols are shaped, we can start modulating the symbols on a baseband level. The transmitted waveform can be seen as:

$$s(t) = Ae^{j\Phi(t)}\ , \tag{1}$$

where A is the transmitted signal's amplitude and $\Phi(t)$ is the angular phase which can be expressed as follows:

$$\Phi(t) = 2\pi h \int_0^t \alpha(\tau)d\tau\ , \tag{2}$$

where h or the *h-factor* is the modulation index defined by $\frac{\Delta f}{f_m}$. Δf is the frequency deviation and f_m is the frequency of the message signal. $\alpha(t)$ for an FSK signal can be expressed as:

$$\alpha(t) = \sum_{i=0}^{L} a(i)q(t - iT_s)\ , \tag{3}$$

where $a = \pm 1$, is the transmitted bit value, L is the number of transmitted bits, and $q(t - iT_s)$ is the shape filter response of the transmitted signal (i.e. for GFSK, the $q(t)$ is a Gaussian filter response). For the signal that we transmit from the SDR, we start from the equations of a Frequency Modulated (FM) signal. The waveform at the SDR front end can then be defined as,

$$Re[s(t)] = A_c e^{\left(2\pi f_c t + 2\pi\Delta f \int_0^t m(\tau)d\tau\right)}, \tag{4}$$

where A_c is the carrier amplitude, f_c is the carrier frequency, Δf is the frequency deviation and $m(\tau)$ is the message. When looking at the symbols of 2-(G)FSK in the frequency domain, we see that symbols are altered around a defined center frequency, which at baseband is 0 Hz. We need to obtain the instantaneous frequency ω_i of that symbol which is defined as,

$$f_i = \frac{1}{2\pi}\frac{d\theta_i}{dt} \rightarrow \theta_i = \int \omega_i dt. \tag{5}$$

Equation (5) shows directly the relation between instantaneous frequency and instantaneous phase, which is that the instantaneous frequency f_i is the rate of instantaneous phase change over time whereas the instantaneous phase is obtained by integrating the instantaneous frequency over time. It is more convenient to implement a phase integrator in GNU Radio since magnitude and phase are easier obtained from standard software blocks. Therefore, we need to discretely integrate the symbols. When translating this to the discrete domain, we can closely approach the integration by using a Riemann sum. We can substitute the message from Equation (2) then to:

$$\int_0^t x(\tau)d\tau \approx \sum_{k=0}^{n} x(nT)T \tag{6}$$

that leads to:

$$s[n] = A_c e^{\left(\omega_c + 2\pi\Delta f \sum_{k=0}^{n} x(nT)T,\right)}, \tag{7}$$

where T is the sampling interval or reciprocal of the IQ sample rate. Furthermore, it can be expressed as:

$$s[n] = A_c e^{\left(\omega_c + 2\pi h \sum_{k=0}^{n} x(nT)\right)}. \tag{8}$$

If we extract the summation of the symbols from Equation (8) and write:

$$y[n] = \sum_{k=0}^{n} x(nT) \tag{9}$$

and apply rectangular integration, we get:

$$y[n] - y[n-1] = \sum_{k=0}^{n} x(nT) - \sum_{k=0}^{n-1} x[nT] = x[nT] \tag{10}$$

which leads to

$$y[n] = y[n-1] + x[nT], \tag{11}$$

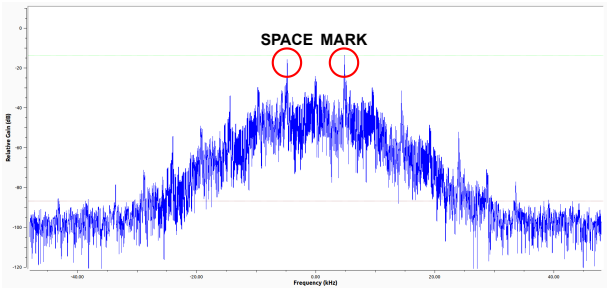

Figure 9. The baseband modulated signal. The two distinctly visible peaks represent the two symbols, i.e. `-1` and `1`.

where $y[n]$ is the output of the filter at time n, $y[n-1]$ output of the filter at time $n-1$ and $x[nT]$ is the input of the filter at time n. We can rewrite this as

$$Y(nT) = Y(nT - T) + X(nT)T. \tag{12}$$

The Z-transform of Equation (12) can be expressed as follows:

$$Y(z) = Y(z)z^{-1} + X(z)T, \tag{13}$$

which leads to a transfer function $H(z)$ of

$$H(z) = \frac{Y(z)}{X(z)} = T\frac{1}{1 - z^{-1}}. \tag{14}$$

Equation (14) is the single pole difference equation of an Infinite Impulse Response (IIR) filter. The implementation can be found in Figure 8 and the baseband modulated signal in the frequency domain is depicted in Figure 9.

4. DASH7 Receiver Design

To create a DASH7 receiver, we split the process into six parts,

- SDR modulation and demodulation
- frequency shifting
- demodulation
- time synchronization
- frame synchronization
- payload decoding

These parts will be discussed in the following sections.

4.1. SDR Modulation and Demodulation

At the transmitter, we need to tune the center frequency of the SDR to a specific channel index. A DASH7 channel frequency can be calculated as follows,

$$f(b, c, d) = \text{Start}(b) + 0.025d + \frac{\text{Chan. Spacing}(c)}{2}. \tag{15}$$

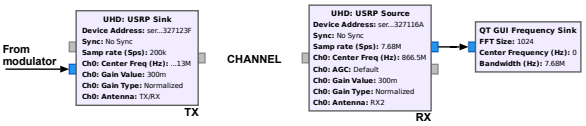

Figure 10. TX and RX settings on two USRP devices.

The parameters b, d and c can be found in Table 1 and in Table 2.

The signal received by the SDR can be described as

$$\hat{r}(t) = A_r \exp\left(j2\pi\Delta f \int_0^t \frac{m(\tau)}{m_p} d\tau\right). \tag{16}$$

The first demodulation step is tuning the SDR RX channel to a center frequency of $866.5\,\text{MHz}$ and capturing a bandwidth of $7.68\,\text{MHz}$. In this way, we investigate the complete $868\,\text{MHz}$ band. This means that the collected data set is down-converted to an appropriate intermediate frequency (IF). In the recorded data sets, we only made use of the $868\,\text{MHz}$ band where channels are assigned between $863\,\text{MHz}$ and $870\,\text{MHz}$ as can be seen in Table 2. The transmission and reception with separate USRPs in GNU Radio can be found in Figure 10.

4.2. Frequency Shifting

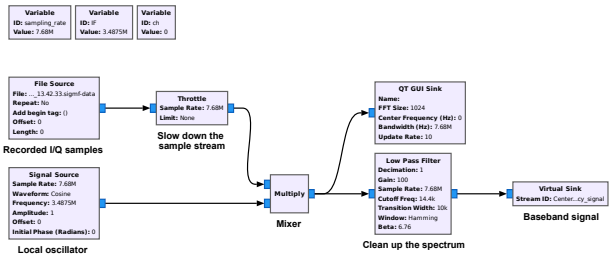

Figure 11. Mixing down the signal to zero and filtering out noise.

In GNU Radio, we set the frequency of an external oscillator to the center frequency of a predetermined DASH7 channel and mix the received signal with this tuned oscillator. This operation downconverts the signal to a baseband signal. The complete flowgraph is shown in Figure 11. The DASH7 specification defines the calculation for channel frequency as,

4.3. Demodulation

To demodulate the signal, we apply the opposite operation as we did in the transmitter design. Obtaining the phase is achieved by using a *Complex-to-Arg*-block. This phase needs to be differentiated and will result in,

$$2\pi\Delta f\frac{m(t)}{m_p} = 2\pi k_f m(t). \tag{17}$$

To implement the differentiator, we recognize that in discrete time,

$$\frac{dx(t)}{dt} \approx \frac{x[n]-x[n-1]}{1} \tag{18}$$

The differentiator can be implemented using an *Interpolating FIR Filter* block with the same taps we used in the transmitter design. This output is passed through a low-pass filter since the differentiation tends to increase high-frequency noise. Eventually, the mapped symbols are retrieved. The demodulation flowgraph can be found in Figure 12.

4.4. Time Synchronization

The purpose of symbol time synchronization or clock recovery process is to find the optimal instants when down-sampling a sequence of samples into a series of symbols. This means that the system needs to select the best sample out of every group of samples, such that this selected sample can better represent the transmitted symbol. The chosen sample is then passed on to the symbol detector, which, in our case is set to the Mueller and Müller timing error detection algorithm. Note that this system is a feedback Phase Locked Loop (PLL) which is set with static parameters. This indicates that there is a trade-off between acquisition speed and tracking stability of the symbol clock estimate. In our implementation, we configure the interpolation resampler as a Polyphase Filter Bank (PFB) which implements a Gaussian-matched filter similar to the one used at the transmitter to reshape the symbols.

4.5. Frame synchronization

After the symbol synchronization, the binary conversion process takes place using the binary slicer. When the symbols are converted to bits, the stream is correlated with a predefined preamble and sync word. Afterward, tags are added to the stream and indicate the start of these fields. At this moment it is predefined what the length of the preamble is, which sync word class and coding scheme is used, and when the encoded payload exactly starts.

4.6. Payload decoding

The added tags, as discussed in section 4.5, are exploited by the *DASH7_demod_py_bb*-block. The *DASH7_demod_py_bb*-block decodes the received payload and calculates a CRC16 to validate if bit errors occurred during transmission. If there is an error, the check will show the message "Check is False". Finally, the data type of the block is converted to float, and the recovered data is

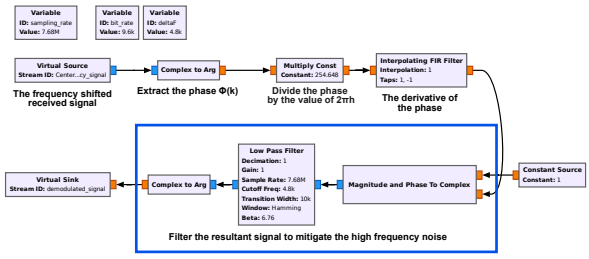

Figure 12. Demodulation process. The phase is extracted from the complex data and subsequently derived and filtered.

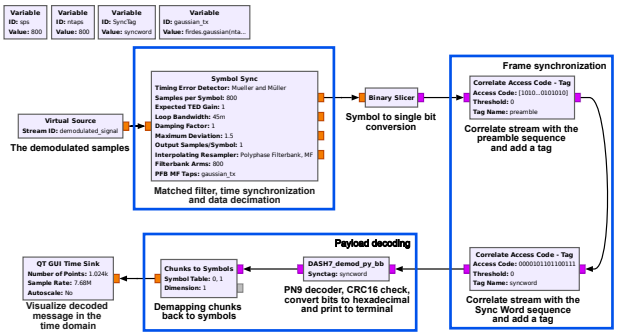

Figure 13. Flowgraph for time synchronization, frame synchronization, and payload decoding.

plotted using a time sink. The payload is printed in decimal and hexadecimal form. The complete implementation of time synchronization, frame synchronization, and payload decoding is found in Figure 13.

If a packet is successfully demodulated, the message, as depicted in Figure 14, is printed to the console window of GNU Radio. The total length of the packet is the first byte that is found. The actual payload `[0x00, 0xAB, 0xCD]` is preceded by the length byte of the payload and is succeeded by two CRC bytes. It is noteworthy that the packet has a total length of 25 bytes, which is due to the other stack layers adding their minimum fields. The complete payload is printed in decimal and hexadecimal form, and a CRC is calculated. The DASH7 spec provides an overview of these fields.

5. Experimental results

To test the flowgraphs in a real-world experiment, we executed several cabled experiments. The experimental measurement setup contained six B-L072Z-LRWAN1 STM32 LoRaWAN Discovery Boards, an Ettus Research USRP B210 SDR, an RF-splitter and a 40 dB RF-attenuator. The development boards were programmed as DASH7 nodes by using the Sub-IoT stack which contains an implementation of the DASH7 Alliance Protocol (Sub-IoT, 2024a).

```
Found packet 1 sync word at location 10200
2024-08-06 14:43:46.931153
MsgLength = 25 bytes
Msg = 24 1 110 32 33 49 55 52 52 0 45 0 23 128 110 0
32 64 0 3 0 171 205 199 20
Hex:
18 01 6e 20 21 31 37 34 34 00 2d 00 17 80 6e 00
20 40 00 03 00 ab cd c7 14
CRC read = 1 1 0 0 0 1 1 1 0 0 0 1 0 1 0 0
CRC calc = 1 1 0 0 0 1 1 1 0 0 0 1 0 1 0 0
Check is True
correlate_access_code_tag_bb :debug:
writing tag at sample 11577
```

Figure 14. Decoded output of a DASH7 packet with a total length of 25 bytes as seen in the console window of GNU Radio. The first payload byte, in bold, is the total length of the effective payload. The original payload of three bytes `[0x00, 0xAB, 0xCD]` is then found and followed by two CRC bytes.

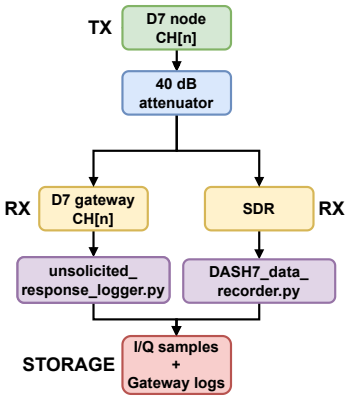

Figure 15. System block diagram of the measurement setup.

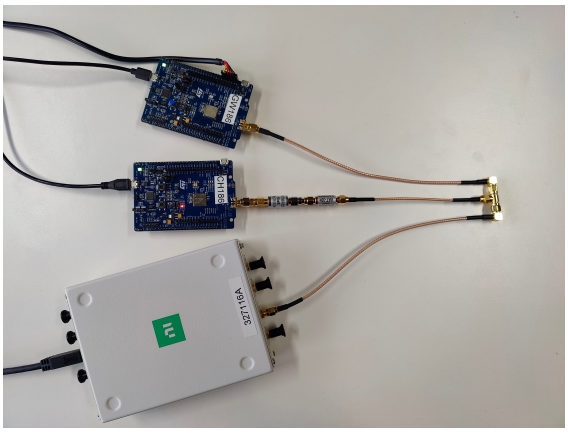

Figure 16. Experimental setup containing one development board acting as a DASH7 gateway and a second board acting as a DASH7 node which transmits a message on Lo-Rate channel 186. The setup is accompanied by an SDR to record the transmitted message.

The nodes use the push communication model and transmit DASH7 messages with a predefined payload. We executed tests on Lo-Rate channels 0, 93 and 186. Therefore, three other boards were programmed as DASH7 gateways. The configured payload of a DASH7 transmission was set to [counter_value, 0xAB, 0xCD]. The DASH7 gateways demodulate and log the amount of successfully received packets.

The *unsolicited_response_logger.py* script is part of pyd7a which is a collection of Python modules that supports the usage of the D7AP (Sub-IoT, 2024b). The script logs the gateway data of each received packet i.e. the timestamp, channel, payload, and Received Signal Strength Indicator (RSSI). Each log entry indicates that a packet has been successfully received by the gateway. The *DASH7_data_recorder.py* script records the transmitted DASH7 packets via a USRP B210 SDR. These recordings were saved using the Signal Metadata Format (SigMF) and can be fed to the created flowgraphs to demodulate and decode messages in post-processing. In total ten recordings were made per channel, forming a complete data set of 30 recordings. Figure 15 shows the block diagram of the measurement setup. Figure 16 shows the bench setup for one channel, i.e. channel 186. The GNU Radio flowgraphs, accompanied by the recorded data set and gateway logs, can be consulted on GitLab (Joosens et al., 2024a) and Zenodo (Joosens et al., 2024b). Furthermore, we extended the testing of the software by a wireless measurement campaign in an indoor and outdoor environment (Joosens et al., 2024c) which has a more extended data set containing more transmissions per recording (Joosens et al., 2024d).

6. Conclusion and Future Work

In this paper, we have investigated the physical layer operation of the DASH7 Alliance Protocol. We created a DASH7 transmitter and receiver design in software that incorporates the functions of the physical layer. These created software tools have been tested in simulation and with real data recordings and perform well. Thus, these tools can be used as a simulation and validation tool for DASH7 packets. In future work, we see several extensions and improvements for the software. One of these extensions is the addition of the Normal-Rate and Hi-Rate channel classes and the implementation of Coding Scheme 2, which uses Forward Error Correction. Furthermore, an automatic frequency control system can be implemented as well. Finally, implementing parts of other layers of the DASH7 stack into the transceiver system can make the tool even more useful for investigating and validating DASH7 packets.

Disclaimer: The content of the present work reflects solely the authors' view and by no means represents the official ESA view.

References

Bni Lam, Noori HN. *Angle of arrival estimation for low power and long range communication networks*. PhD thesis, University of Antwerp, 2021a.

BniLam, Noori, Steckel, Jan, and Weyn, Maarten. Synchronization of multiple independent subarray antennas: An application for angle of arrival estimation. *IEEE Transactions on Antennas and Propagation*, 67(2): 1223–1232, 2018.

BniLam, Noori, Joosens, Dennis, Steckel, Jan, and Weyn, Maarten. Low cost aoa unit for iot applications. In *2019 13th European Conference on Antennas and Propagation (EuCAP)*, pp. 1–5. IEEE, 2019.

BniLam, Noori, Janssens, Robin, Steckel, Jan, and Weyn, Maarten. Aoa estimates for lpwan technologies: Indoor experimental analysis. In *2021 15th European Conference on Antennas and Propagation (EuCAP)*, pp. 1–5. IEEE, 2021b.

Centenaro, Marco, Vangelista, Lorenzo, Zanella, Andrea, and Zorzi, Michele. Long-range Communications in Unlicensed Bands: The Rising Stars in the IoT and Smart City Scenarios. *IEEE Wireless Communications*, 23(5): 60–67, 2016.

Christiansen, Grant. *Design Note DN509 - Data Whitening and Random TX Mode*. Texas Instruments, June 2010. URL https://www.ti.com/lit/an/swra322/swra322.pdf?ts=1725805976825. [Online; accessed 13. Sep. 2024].

D7A. DASH7 Alliance Wireless Sensor and Actuator Network Protocol Version 1.2. *DASH7 Alliance Specification*, pp. 1–68, 2018.

D7A. DASH7 Alliance. http://www.dash7-alliance.org, 2024. [Online; accessed 13. Sep. 2024].

Hoel, Robin. Fec implementation, April 2007. URL https://www.ti.com/lit/an/swra113a/swra113a.pdf?ts=1725805949857. [Online; accessed 13. Sep. 2024].

Janssen, Thomas. *Energy-efficient Positioning for the Internet of Things*. University of Antwerp, 2023.

Joosens, Dennis, BniLam, Noori, Berkvens, Rafael, and Weyn, Maarten. DASH7 / Multi-Channel DASH7 IoT Communication System · GitLab, September 2024a. URL https://gitlab.ilabt.imec.be/dash7/multi-channel-dash7-iot-communication-system. [Online; accessed 4. Sep. 2024].

Joosens, Dennis, BniLam, Noori, Berkvens, Rafael, and Weyn, Maarten. Implementation of a Multi-Channel DASH7 IoT Communication System for Packet Investigation and Validation, September 2024b. URL https://zenodo.org/doi/10.5281/zenodo.13734532. [Online; accessed 13. Sep. 2024].

Joosens, Dennis, BniLam, Noori, Berkvens, Rafael, and Weyn, Maarten. Sdr-based iot communication systems: An application for the dash7 alliance protocol. [unpublished manuscript]. *Applied Sciences*, 2024c.

Joosens, Dennis, BniLam, Noori, Weyn, Maarten, and Berkvens, Rafael. SDR-based IoT Communication Systems: An Application for the DASH7 Alliance Protocol, September 2024d. URL https://zenodo.org/doi/10.5281/zenodo.10961311. [Online; accessed 13. Sep. 2024].

Schneider, David. Dash7 Wireless Networking Gains Momentum. *IEEE Spectrum*, 2010. URL https://spectrum.ieee.org/dash7-wireless-networking-gains-momentum. [Online; accessed 13. Sep. 2024].

Singh, Ritesh Kumar, Puluckul, Priyesh Pappinisseri, Berkvens, Rafael, and Weyn, Maarten. Energy Consumption Analysis of LPWAN Technologies and Lifetime Estimation for IoT Application. *Sensors*, 20(17): 4794, aug 2020. ISSN 1424-8220. doi: 10.3390/s20174794.

Sinha, Satyajit. LPWAN market 2024: Licensed technologies boost their share among global 1.3 billion connections as LoRa leads outside China, September 2024. URL https://iot-analytics.com/lpwan-market. [Online; accessed 13. Sep. 2024].

Sub-IoT. Sub-IoT, 2024a. URL https://github.com/Sub-IoT. [Online; accessed 13. Sep. 2024].

Sub-IoT. pyd7a, 2024b. URL https://github.com/Sub-IoT/pyd7a. [Online; accessed 13. Sep. 2024].

Weyn, Maarten, Ergeerts, Glenn, Wante, Luc, Vercauteren, Charles, and Hellinckx, Peter. Survey of the dash7 alliance protocol for 433 mhz wireless sensor communication. *International Journal of Distributed Sensor Networks*, 9(12):870430, 2013.

Zanella, Andrea, Bui, Nicola, Castellani, Angelo, Vangelista, Lorenzo, and Zorzi, Michele. Internet of Things for Smart Cities. *IEEE Internet of Things journal*, 1(1): 22–32, 2014.