Extending IoT Connectivity of Embedded Devices with M2M High-Speed Acoustic Data Transmission

Henrique Gonçalves Nogueira
henogu@utu.fi
University of Turku
Turku, Finland

Jorge Peña Queralta
jopequ@utu.fi
University of Turku
Turku, Finland

Tommi Otsavaara
tommi@darkglass.com
Darkglass Electronics
Helsinki, Finland

Tomi Westerlund
tovewe@utu.fi
University of Turku
Turku, Finland

ABSTRACT

Connectivity is central to the development of new products and services for the Internet of Things (IoT). Over the past two decades, multiple solutions have emerged to provide connectivity to embedded devices and build the IoT as we know it today, from Wi-Fi and Bluetooth modules to the latest developments in low-power wide-area networks. Nonetheless, multiple products exist on the market with embedded processors that have no connectivity. We provide a solution for extending IoT connectivity to musical instruments equipped with pickup transducers through one-way acoustic data transmission. By exploiting existing hardware in a variety of musical instruments, such as electric or bass guitars, we provide a new communication interface that can be utilized to upgrade the firmware or modify the settings of these devices. This process is currently done with a wired connection. We present a system that allows users to send instructions or new firmware to musical instruments utilizing their smartphones through a wireless acoustic link. We have implemented the proposed solution, which is based on OFDM modulation, achieving transmissions speeds of one order of magnitude higher than the most widely used solution at the moment. Through extensive experiments, we analyze the performance of our system under different channel conditions and settings. In addition, we provide a strategy to cope with frequency offsets between emitter and transmitter.

CCS CONCEPTS

• **Computer systems organization** → **Embedded software**; **Sensor networks**; **Embedded software**; • **Hardware** → **Digital signal processing**; • **Networks** → *Link-layer protocols*; *Transport protocols*.

ICCIP'19, November 15–17, 2019, Chongqing, China

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

KEYWORDS

IoT; Embedded Computing; Acoustic Communication; Sound-based Communication; Acoustic Data Transmission; IoT Connectivity;

ACM Reference Format:
Henrique Gonçalves Nogueira, Jorge Peña Queralta, Tommi Otsavaara, and Tomi Westerlund. 2019. Extending IoT Connectivity of Embedded Devices with M2M High-Speed Acoustic Data Transmission . In *The 5th International Conference on Communication and Information Processing - November 15–17, 2019, Chongqing, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The Internet of Things (IoT) is built around connected devices, which monitor data or bring intelligence to multiple domains [18]. In most applications, data transmission systems for embedded devices are implemented using electric signals, transmitted under controlled physical media such as copper traces and wires, or wirelessly with the propagation of radio waves [23]. However, while the number of connected devices is reaching the order of tens of billions [3], multiple products and applications involving embedded devices lack the connectivity required to join the IoT [12]. In this work, we develop a novel wireless communication solution based on acoustic data transmission to extend IoT connectivity to musical instruments exploiting the existing hardware.

End-devices such as sensor nodes or actuators in the IoT are often connected to a local network via wireless communication technologies such as Wi-Fi [22], or Bluetooth [4]. More recently, low-power wide-area network technologies have seen an increasing penetration [17]. A key aspect of some LPWAN solutions such as LoRa is that, with their basic configuration, enable one-way machine-to-machine (M2M) communication without the need for a network protocol [13]. This is not true with cellular networks [14], or local area networks such as Wi-Fi, while Bluetooth does provide support for M2M but requires two-way communication to establish the initial connection. In many applications, one-way communication is enough or even preferable. This is the case, for instance, of firmware updates or change of device settings. Moreover, doing this with M2M communication without a network-level protocol simplifies the implementation, allowing for more embedded devices to dedicate less resources. At the same time, it reduces the power consumption of end-devices and reduces the number of attack vectors in the case of cyber attacks [21]. To achieve this, other wireless

technologies can be utilized, such as ultra-wide band (UWB) data transmission [1]. Recently, research into the use of sound-based or acoustic communication has been explored [9]. The research presented in this paper has been carried out in collaboration with *Darkglass Electronics Oy*, a Helsinki based company specializing in premium bass gear, in particular, bass distortion pedals [10]. Because of the nature of Darkglass products (musical instruments or related devices), we have focused on acoustic data transmission in order to exploit existing hardware in those devices to provide wireless connectivity. Until recently, the only option for upgrading the firmware or updating the settings of these devices was to utilize a desktop application and connect the device through USB. With our proposed solution, we provide users with a mobile application that they can utilize to transmit data over sound waves generated with a phone's speakers. By placing the phone close to their instruments, the embedded processor can receive the data utilizing the pickup sensor in the instruments. With this approach, we are able to extend IoT connectivity to a wide variety of devices with a simple firmware update, even if they were not engineered to support wireless communication.

In summary, this paper provides a set of design strategies for the software implementation of a sound-based data transmission system, requiring little to no additional hardware support for most commercially available embedded platforms. For this purpose, acoustic-based channels provide an attractive solution, not only due to the uncomplicated integration efforts but also because it facilitates a signal path for wireless data transmission. These approaches support high-speed data transfers due to their large signal bandwidth and are preferred over alternatives that rely on mechanical processes, such as the propagation of sound. Nonetheless, other channel characteristics need to be taken into consideration. Aspects such as whether the communication is being performed out of line-of-sight, system hardware limitations, or whether the environment restricts RF propagation, may define the need for the application of non-conventional communication channels. For example, if with underwater wireless communication, due to an unusual ineffectiveness of underwater electromagnetic propagation [16].

The focus of this paper is to develop and document a software communication system intended to run on an embedded microprocessor, and which is implemented using a wireless communication channel based on the propagation of audio signals. Data-over-sound (DoS) is the concept of transmission of digital information through audio, which, unlike conventional speech-based audio communication, relies on symbolic data. Information is thus typically encoded in binary format. When compared with other channel media that make use of electric or electromagnetic signals, acoustic propagation uses a much narrower bandwidth at a lower frequency. However, also due to its slower data-rate nature, no specialized hardware is necessarily required to implement capable modem systems that can efficiently utilize this spectral band. Acoustic signals can be read as a continuous string of audio samples, and if the processing system has enough memory and is fast enough, it is possible to process these samples at a rate that facilitates real-time data transmission. So, unlike radio-based communication, DoS systems can theoretically be implemented on any existing device that already integrates the underlying hardware support for such an interface.

The main contributions of this paper are (1) the design, development and implementation of a one-way data-over-sound communication that exploits existing pickup sensors in musical instruments to provide IoT connectivity; (2) the design and development of a strategy to cope with frequency offsets between receiving and transmitting devices, which extend the application to a wider variety of embedded processors which do not have a high-quality oscillator or its frequency is not precisely known; and (3) the analysis of the proposed acoustic wireless communication solution, achieving data rates over the state-of-the-art for data-over-sound transmissions. The proposed approach can be easily extended to two-way communication with the appropriate protocol definitions. In addition, encryption can be easily integrated with symmetric-key cryptography if each end-decide has a preassigned encryption key that is known by the device transmitting the data.

The remainder of this document is structured as follows. Section 2 reviews related works in acoustic communication for the IoT. Section 3 then introduces the objective application areas and sets the scope for the proposed communication system. Section 4 delves into the main implementation challenges and factors to take into account from the point of view of signal modulation and system coherency. The system implementation and main challenges are outlined in Section 5, with a focus on the embedded design. Experiments and results are described in Section 6. Finally, Section 7 concludes the work and outlines future work directions.

2 RELATED WORK

The proposed system implementation presents several challenges. Nevertheless, arguably, the fundamental difficulty is related to the implementation of real-time digital signal processing of wireless communication algorithms on the limited computational capabilities of embedded software platforms. In recent years, solutions relying on data-over-sound transmissions have emerged to extend the possibilities of connectivity in the IoT. However, most recent advances have happened within the industrial sector, and the solutions are not open-sourced. In this section, we provide an overview of the two most widely used solutions for data-over-sound from *Mutable Instruments SARL* [8] and *Chrip.io* [6].

The company *Mutable Instruments SARL* provides a promising case study for this field of research with the development of digital synthesizer modules supporting audio-based firmware update features. Their designs are publicly available as open-source [8]. For example, the *Elements* modal synthesizer runs on an STM32F405RG Microcontroller chip from *STMicroelectronics*, which coincidentally has very similar characteristics to the chip used for project development, having, in particular, the same Cortex-M4 ARM processor architecture. By consulting the firmware code available at their Github repository, we can infer that the firmware update system is using a single carrier quadrature phase-shift-keying (QPSK) as a signal modulation scheme. Notably, the software implementation is surprisingly short and is kept rather simple, however, the bandwidth usage cannot be very efficient, as it is using a signal correlation estimator based on the sign information alone. Furthermore, even though the firmware update procedure is based on signals produced in the acoustic band, it only supports a wired interface to the audio source, meaning that the implementation does

not have to account for effects due to wireless propagation. In our work, we utilize OFDM modulation to provide higher bandwidth and configurable channel utilization.

A white paper published by *Disruptive Analysis Ltd.* provides an in-depth analysis of DoS as a technology and its current and potential application in different fields [6]. In particular, the concept is gaining traction over fields for the Industrial Internet of Things (IIoT). Multiple companies are already working on DoS as a cross-platform third party technology for the IIoT. One of the solutions with higher market penetration is *chirp.io*, a B2B service-based company developing a suite of software development kits (SDK) that can work under different platforms, including a plethora of operating systems or even support for C development of embedded bare-metal firmware. Their focus is on acoustic transmission through sound propagation using an M-ary frequency-shift-keying (FSK) modulation scheme due to its robustness in multipath-propagation environments, while also being adaptable to different data-rate profiles. There is a clear parallelism between this solution and our proposed system. In particular, the fact that both harness the capabilities of unspecified pre-existing hardware. Nevertheless, dealing with multi-platform development, decisively serves to amplify the challenges faced by the company, as the processes must not only be adapted to the device's computational abilities but also the audio-related hardware interfaces.

3 APPLICATION SCENARIO

The results presented in this paper are the outcome of research collaboration with *Darkglass Electronics Oy* for the development of a novel solution to extend the connectivity of premium bass distortion pedals. These systems are highly configurable, facilitated by the fact that they integrate a microcontroller unit (MCU) that executes firmware responsible for controlling the different device modules. In essence, the software can be rewritten/reconfigured to completely modify the user interface behaviour as well as the operation of some of the analog subsystems and the digital signal processing (DSP) algorithms. The project idea consists of extending the user connectivity, with an alternative channel to the USB interface already implemented, allowing users to configure the devices on-the-go. Since these systems are designed to process audio signals they already possess the necessary analog conversion hardware, and thus the acoustic signalling path is already well established.

Darkglass' system is originally implemented on Microchip's ATSAM4S4A MCU, currently deployed on all the relevant devices, running at 120 MHz frequency with an ARM Cortex-M4 architecture. Most of the interface messages are relatively small in size, typically under 1 kB, with a few exceptions: impulse response filter coefficients start at 2 kB in size but are expected to increase for products further down the line in the future, and firmware update images, which, for now, can theoretically go up to 120 kB in size. Since, ideally, the longest transmission duration should be under around one minute, for the sake of the quality of user experience, the targeted system transmission rate can be defined to be around 16 $kbps$.

The physical interface consists of using the phone's stereo system to provide the acoustic signal that can be picked up by the magnetic pickup-sensor present in the bass guitar. Note, however, that this wireless transmission route is not truly acoustic in nature as the audio signal is propagated through disturbances in the magnetic field. Mainly, the vibrations of the stereo actuator are directly read as an electric signal in the bass pickup-sensor without requiring the mechanical propagation of the signal through the air. This method works similarly to how the bass string vibrations are converted into electric signals and as such presents slightly different characteristics when compared to normal acoustic propagation. Nonetheless, the methods that have been developed can be extended to pure acoustic transmission over the air or other mediums. For longer distance transmissions, multipath propagation needs to be taken into account.

Table 1: Synthesis of the system's specifications

	Specification
Direction	Simplex (user out, device in)
Data integrity	Lossless data transmission
Security	Unencrypted / encrypted with preset key
Data rate	2 kB/s
Route	Wired or wireless(stereo-pickup pair)
Message length	> 64 kB
Sample rate	≃48 kHz

The communication channel is also of type simplex, meaning that data can only travel in one direction: from the user to the device. This poses an additional challenge at a protocol level, since the transmitter does not know when or if the receiver has received a complete message. Given that transmission should be lossless at all times, independently of the form of the message, strategies need to be in place guaranteeing that the receiver may always eventually receive a complete message. The fundamental system specifications are summarized in Table 1.

3.1 Signalling scheme basics

The proposed approach presented in this paper is based on the orthogonal frequency-division multiplexing (OFDM) signalling scheme, enabling the simultaneous transmission of several carrier signals in such a way that they do not interfere with each other. Crucially, the OFDM scheme provides a straightforward path for configuring the system's spectral band utilization, and thus enables its adaptation for different applications and channel conditions. This is achieved by choosing which components to use for the carriers from the set of available frequencies, and thus requires no additional coding efforts.

3.1.1 OFDM scheme. The OFDM scheme works under the principle of orthogonality of time-windowed sinusoidal signals, whereby the central spectral component of two signals does not suffer from the presence of the other, even though the spectrum of the signals may overlap. Essentially, given two sinusoidal signals $x_A(t)$ and $x_B(t)$, of frequency f_a and f_b, windowed in time by the same square pulse of duration T_D and separated in frequency by a distance of $\Delta_f = k/T_D, (k = 1, 2, 3, ...)$, then, the frequency components at those frequencies are independent of the presence of the complementary

signal: $X_A(f_B) = 0$ and $X_B(f_A) = 0$, where $X_A(f) = \mathcal{F}(x_A(t))$ and $X_B(f) = \mathcal{F}(x_B(t))$.

The parameter T_D is defined as the symbol's duration and determines how fast the signal is switching between consecutive symbols, transmitted as OFDM signals. With a longer symbol duration, the number of samples corresponding to the OFDM signal increases and the minimum inter-carrier frequency distance Δ_f value decreases proportionally, meaning that the carrier waves can be more tightly packed together. As such, the data rate does not directly depend on the symbol's duration/rate, but rather on the available bandwidth B and the number of bits transmitted per carrier M. Assuming no signal overhead, the theoretical data rate limit R_{max} can be derived from the Nyquist rate theorem $R_{max} = B \log_2(M)$.

3.1.2 M-QAM carrier modulation. The carriers are generated using an M-ary QAM (M-QAM) modulation, in particular it uses QAM-16, which transmits 4-bit wide symbols at a time. This is achieved by characterizing each of 16 possible symbols to a specific amplitude and phase value of a given sinusoidal component windowed in time, which can be represented as a point-constellation mapped to the complex plane. Choosing the M-QAM constellation is generally a compromise between its bit-density and immunity to noise.

3.1.3 Cyclic prefix. Naturally, the OFDM symbol, or payload, must be sampled by the receiver during the period in which it is transmitted. However, this is nearly impossible to achieve if the entire signal is to be transmitted only once, as it would require an exact timing synchronization between emitter and receiver. The payload signal is thus prepended with a cyclic prefix (CP), consisting of a replica of its ending, essentially extending its presence over time. It is a fundamental concept for OFDM systems as it allows reduced accuracy requirements for the timing synchronization. From a spectral analysis point of view, it is easy to understand since the resulting spectrum, sampled within the CP plus payload block period, will result in the same spectrum, except for a frequency-linear phase change which can be compensated for with coherency strategies.

3.1.4 Guard interval. A signal travelling through a band-limited channel is spread over time even shortly after the ending of its transmission, similarly to the perceived effect of sound echoing. In the case of OFDM signals, this phenomena causes the occurrence of inter-symbol interference (ISI) due to how successively transmitted symbols can negatively affect each other, translating into perceived signal noise. To decrease the impact of ISI, the most straightforward and effective strategy is to introduce guard intervals (GI) between successive symbols, and thus let the remnants of previous symbols to simply *die out*. A recommended strategy is to utilize the GI as a placeholder for the CP, so that, as the synchronization instant approaches the end of the CP, the ISI also decreases.

3.1.5 Reducing the peak-power-to-average-power-ratio. One key challenge with OFDM systems has to do with the containment of the peak-power-to-average-power-ratio (PAPR). Essentially, the OFDM signal carriers can line up in such a way that the resulting wave concentrates a big portion of its energy in a short time-span, leading to quick bursts of energy. These bursts should be avoided as they may result in the breaking of the peak power limitations or dynamic range of the emitter/receptor, leading to signal distortion and, consequentially, loss of signal information. The simplest way to avoid this situation is by generating data entropy, i.e. reducing the likelihood of non-optimal phase alignments in the signal by introducing apparent randomness. For this purpose, all outgoing data is "scrambled" by XORing the data with a predefined data array, immediately before translating the data into the transmitted signal, and the same operation is performed by the receiver for all incoming data, immediately after extracting the symbolic data from the transmitted signal.

4 SYSTEM COHERENCY

The fundamental challenge in the implementation of OFDM-based systems, especially when using M-QAM modulation, consists of achieving good system coherency, i.e., the receiver must be able to retrieve the phase and amplitude information of the original signal, as intended by the transmitter. This involves the application of strategies that not only serve to achieve time-wise synchronization but also, to compensate for channel distortion effects and inaccuracies in the device's main oscillator. With our proposed system architecture, these schemes are founded on the transmission of a preamble signal, whose goal is entirely dedicated to this purpose.

4.1 Timing synchronization scheme

For the timing synchronization scheme, we propose a solution adapted from the works published by Timothy Schmidl and Donald Cox (S&C), where they present an efficient algorithm for calculating precise timing for the start of a frame, or packet, of a given signal [20]. The idea consists of the transmission of a preamble signal, consisting of two consecutive and identical training symbols, T0 and T1, preceded by the CP. Due to its ideal correlation properties, we recommend using the Zadoff-Chu sequence, which can be advantageously generated to occupy the frequency components relevant to the transmission of the OFDM payloads [7].

The underlying idea behind the synchronization algorithm consists of calculating the instant at which the sample signal consists of two equal parts, i.e. the training symbols. The algorithm proposed by (S&C) is essentially a continuous calculation of signal correlation and provides a method to derive the timing synchronization instant from three metrics that can be efficiently computed in real-time. The metric $P(d)$ is a measure of auto-correlation and intends to represent the degree of similarity between the training symbols. It measures the correlation between two subsets of a signal $x(d)$ spaced L samples apart and, importantly, can be calculated with an iterative approach as shown in Eq. 1.

$$P(d+1) = P(d) + x(d)x(d-L) - x(d-L)x(d-2L) \quad (1)$$

To normalize the metric $P(d)$, the authors include the metric $R(d)$, which measures signal energy for the duration of the last L samples, and can also be iteratively calculated according to

$$R(d+1) = R(d) + x(d)^2 - x(d-L)^2 \quad (2)$$

and the final metric $M(d)$ can then be calculated from

$$M(d) = |P(d)|^2 / |M(d)^2)| \quad (3)$$

The metric $M(d)$ represents a measurement of the "L-distanced" autocorrelation, normalized to the signal's energy, varying between the values 0 and 1 according to the degree of correlation between the successive symbols. The synchronization instant can be extracted

from this metric through various methods such as peak-detection, but for most cases, a simple threshold crossing approach should suffice. Notice however that, due to the addition of the CP, the peak is rather perceived as a plateau, and any point in-between can be seen as a proper candidate for the synchronization instant.

4.2 Channel calibration

Even if the timing synchronization is accurate, the signal propagation through a linear channel may cause the distortion of the amplitude and phase information. These signal transformations are predictable and operate similarly while channel conditions remain the same, meaning that if the receiver can extract a model of that behaviour once, at the start of transmission, it can repeatedly predict and compensate for the channel effects. This is the process of channel calibration and, the easiest way to achieve it is to transmit a fixed signal and determine what the inverse signal transformations required to generate the original signal from the receiver's end.

The preamble adopted for the timing synchronization scheme can also be used to transmit fixed reference signals as training symbols, already known from the receivers end. Then, the complex channel compensation factor for each spectral component k, c_k, can be calculated from the values of the expected, e_k, and received, r_k, components: $c_k = e_k/r_k$. From then on, the receiver simply applies the compensation factor to the respective component of the following OFDM symbols, thus extracting a good estimation of its actual value. Notice that the spectral band of the training symbols should correspond to the same band as the one used by the OFDM symbols so that the signal energy is contained only within the relevant frequencies. It is worth noticing that this channel calibration scheme also compensates for slight offsets in the timing synchronization, as long as the synchronization instant is located somewhere within the limits of the CP. This is because any time delay, positive or negative, simply generates a linear phase transformation that can easily be accounted by the same calibration procedure, i.e. it is as if the delay was a channel effect as well. Naturally, though the synchronization instant should still be as accurate as possible to improve the perceived SNR of the following symbols.

4.3 Frequency offset compensation

So far, all design decisions proposed in this section assume the existence of perfect system oscillators, meaning that the transmitter and receiver sampling frequencies perfectly align with their theoretical values. Notice that not only are the absolute frequency values tremendously crucial for maintaining carrier orthogonality with the OFDM scheme but also, the precise timings of the packet signal depend on this factor as well. For this reason, a frequency offset compensation scheme is presented here that can be used to compensate for small sampling frequency discrepancies with little additional processing load.

The works presented by Sliskovic (2001) showcase a process for accurately estimating the frequency offset of an OFDM signal [15]. The idea is to transmit two consecutive training symbols that utilize the relevant frequency components and to compare the phase of the resulting spectrums. When assuming no noise, the frequency-offset Δ_f produces a frequency-dependent phase shift φ between both training symbols, according to a factor ε,

$$\varphi(f) = \varepsilon(\Delta_f) f \tag{4}$$

The correction methodology presented by the previous author assumes that the signals are sampled at the precise instants. However, the frequency-offset also produces an ever-increasing timing misalignment that must be taken into consideration. Since all the essential signal timings are derived from counting the number of samples occurring since the synchronization instant, the frequency offset has the effect of changing the timing offset proportionally to this number. This timing offset ends up producing an effect very similar to the one described by Eq. 4, by inducing a phase shift to all frequency components, while the value of this shift is proportional to the number of samples that span between both sampled signal.

To estimate and correct for this effect, we reuse the preamble's training symbols to derive the relevant phase shift values according to Eq. 4, for each relevant component. Through simulation results, we concluded that the bulk of the perceived distortion is actually due to timing misalignment induced by the frequency offset. As such, the corrective solution applied for our solution can be reduced to a timing offset compensation algorithm, that is described as follows: (1) we separately apply the FFT over each training symbol t0 and t1, obtaining the real and imaginary values of each of the relevant frequency components, expecting that the resulting spectrum T0 and T1 be the same, except for the phase shifts produced by the effects of the frequency offset; (2) divide each component of the spectrum T1 by each component of T0 and derive each of the phases φ_k, which corresponds to the phase shift described in Eq. 4; (3) calculate $\Delta_{\varphi_k} = \varphi_k/N$, where N represents the number of samples composing the training symbol, and the factor Δ represents the induced phase shift per sample for each frequency component; and (4) the value of Δ_{φ_k} can be used in different ways. One way is to calculate the correction phase shift at each relevant point throughout the packet signal by multiplying this factor with the number of samples it takes to reach that point. Alternatively, a more optimal approach is to calculate the phase correction factors that can be continuously re-applied to the channel compensation factors, in order to compensate for this behaviour over time. This approach assumes that future sampling instants are equally spaced apart, which happens to be the case for the signalling scheme used for this system.

5 SYSTEM INTEGRATION

In general, the system implementation details must be adapted to the constraints of its respective platform. For this purpose, the system must not only leverage the notion of reconfigurability, but should also offer the right tools to help with its integration upon the overarching design.

5.1 Mode-switch signalling scheme

In principle, the timing synchronization process already mentioned could also be used as a way to detect the start of message transmission. Nonetheless, while still relatively efficient, this approach would consistently require an excessive amount of continuous computing power. The mode-switch detection algorithm consists of the identification of a predefined signal consisting of a specific

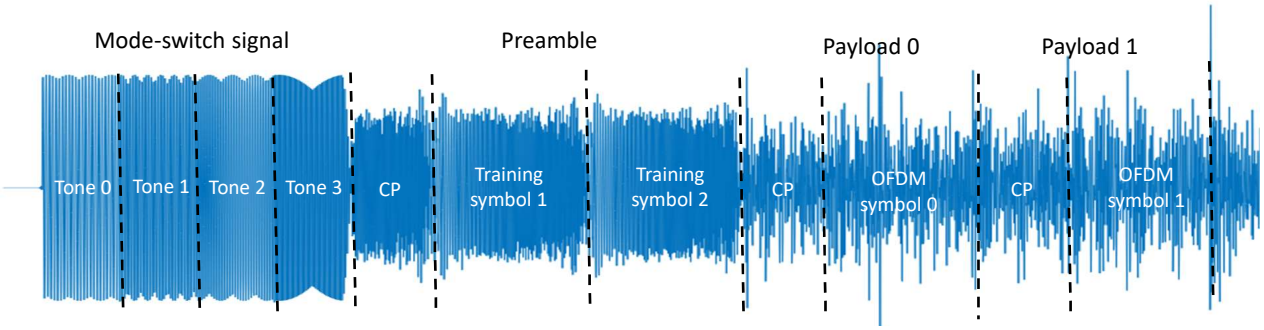

Figure 1: Time graph of the packet signal construction for an arbitrary format.

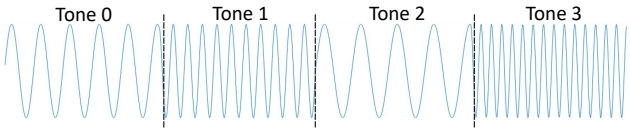

Figure 2: Illustrative example of a tone-sequence.

sequence of successive sinusoidal tones. This frequency pattern defines a tone sequence, which the algorithm tries to match with its internal key value. The graph presented in Figure 2 shows the time-based representation of a tone sequence, composed of four different and consecutive sinusoidal waveforms. This effectively corresponds to an FSK-type modulation, where symbols are defined in terms of frequencies and transmitted in succession. With a simple zero-crossing counter algorithm, we achieve a computationally efficient mode-switch signalling scheme.

5.2 Structure of the frame signal

The transmitted signal is generated as a number of independent frames, each consisting of several blocks transmitted in sequence: one mode-switch block, one preamble block, and several OFDM payload blocks transmitted in sequence. The communication system processes the signal as it is being transmitted, and as such, the order in which the several processing stages are located follows the order of the respective sub-signals transmitted in sequence. Figure 1 presents an example of the first part of the time graph of a frame signal for an arbitrary format configuration. Here the emphasis is on the start of the transmission and the stage progression starting from the mode-switch signal, going to the preamble and finally onto the OFDM payload stages.

All frames start with the mode-switch signal so that the receiver can react to the start of the packet on time to start computing the next stage. As we can see, this block is much higher in amplitude than the remaining blocks due to the fact this stage is much more resistant to signal saturation distortion effects, since the only relevant information is contained in the frequency of zero-crossings. Also, it is important to note that all frames include this sub-signal so that the receiver can identify the message transmission from any arbitrary point in the message, which is cyclically transmitted.

Immediately following is the preamble, which will serve to provide system coherence for the following OFDM payload blocks. Naturally, the stage precedes every first payload stage because this information is crucial for decoding of the OFDM payloads. After this stage, the system can then infer all critical instants within the frame. Finally, several payloads are sent in succession for an arbitrary number of times. Each *Payload* is encoded using the OFDM signalling scheme and contains a subset of the entire packet data.

5.3 Protocol-level considerations

From a transactional point of view, the communication protocol is designed towards the transmission of independent messages, made up of a variable number of fixed-size packets, corresponding to separate signal frames. This approach simplifies the system design, in the sense that no additional signalling is required for dictating the packet decoding control-flow, and at the same time, the packets efficiently tailored towards the specific information density of each frame. The packet size can be chosen to be such that it complies with a whole number of symbols transmitted by the OFDM signals. This ensures that no signal is effectively unused, while simultaneously maintaining complete independence between separate packets.

5.3.1 Data integrity. To achieve lossless data transmission, each packet contains a code, corresponding to a signature of the given data content. This code is generated by a hash function such that, if two data sets are different, then, there is an extremely high probability that the respective codes will differ as well. The integrity check process simply consists of recalculating the hash of the received data-set and comparing it with the hash present in the packet's data content. Even though it was not designed for this purpose explicitly, the cyclic redundant check algorithm (CRC) is a prime candidate for the hash function due to its ubiquity and often present in embedded platforms as dedicated hardware accelerators.

5.3.2 Packet identification. The packet structures represent the atomic piece of data. Since the communication system is unidirectional by nature, with packets being lost in the middle unbeknownst to the transmitter, the packet must include additional information regarding its place in the overall message. For this purpose, we have opted for the most straightforward strategy consisting of identifying each packet with two additional byte numbers, an slot identification byte and the total number of slots, respectively.

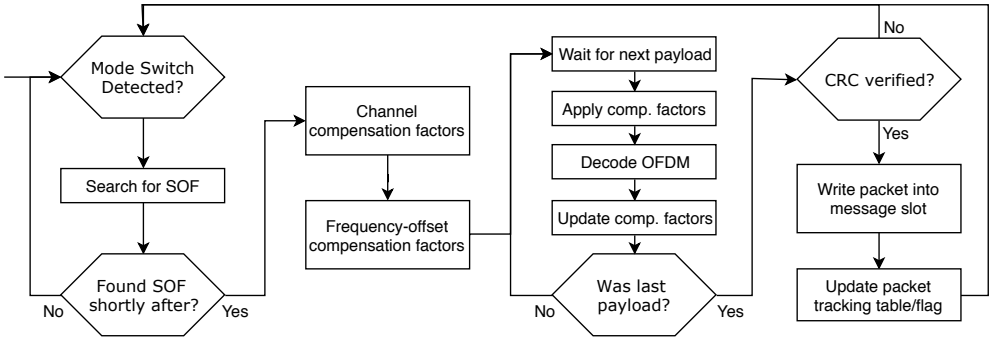

Figure 3: System component level flow diagram.

5.4 Implementation Overview

Figure 3 illustrates a flow diagram representing the system as a sequential process. We can visualize the processing steps, starting from the physical signal processing, and going all the way to the final message. It has been laid out in a way that is tightly related to the actual program processing flow.

The mode-switch detection process is always running in the background, and its sole purpose is to notify the system that the start of a packet has been detected. This is particularly relevant from a system integration's point of view since the processing resources are scarce and must be shared by the entire system. The search for the start of frame (SOF) consists of the calculation of the M metric, mentioned in Section 3.1, and includes the calculation of the packet synchronization instant. The SOF always occurs in a given time window after the detection of the mode-switch signal. If the search exceeds this expected deadline without identifying a SOF, it cannot guarantee precision in timing synchronization, and so, the process falls back to normal mode and ignores the following packet signal.

Once the synchronization instant is known, the program uses the preamble signal to compute the channel compensation factors. The frequency-offset compensation factors are also calculated using the information present in the preamble signal. Before decoding an OFDM signal, the program must wait for the transmission of a number of samples corresponding to the payload block size. The OFDM payload is translated into the frequency domain and the relevant components are corrected using the channel calibration and frequency-offset compensation factors. This is done by first applying the frequency-offset factors over the channel compensation factors, and only then the resulting factors onto the payload's spectrum.

During the OFDM decoding phase, the program computes the data by translating the payload samples of the OFDM signal. Once the data is extracted, it is appended to an internal packet buffer. The compensation factors are updated using the information from the decoded signal, i.e. by assuming that all symbols are correct and calculating the resulting error. At the end of the packet data extraction, the 32-bit CRC value is calculated and compared with the corresponding field embedded in the packet data. The packet corresponds to a chunk of the entire message under transmission. Due to the cyclic nature of the unidirectional message transmission process, these packets do not necessarily arrive in sequence, and so, the content data needs to be copied to the appropriate chunk of the message buffer. The message is only transmitted once all the corresponding packets have been received. For this purpose, a binary table is updated every time, in order to determine whether the packet is new, and a counter is maintained to determine whether the full message has been received. Once the message is complete, a flag is set to signal that the message buffer contains a valid message.

6 EXPERIMENTATION AND RESULTS

This proposed communication scheme had been implemented on the ATSAM4S4A microprocessor, with a 32-bit ARM Cortex-M4 processor architecture, and clocked at 120MHz. Nevertheless, the details described in this paper are entirely platform independent and can be directly applied to many other MCU architectures. To test out the implementation, the system is running in development mode on a Darkglass bass distortion pedal, which contains all the necessary hardware for the audio signal path.

6.1 System configurations

The system modules have been implemented in a way that they support different configurations, depending on the format of the transmission signal, such as OFDM payload, CP length and the bandwidth usage of the signal itself. These configurations directly impact the transmission speed and can essentially determine whether the implementation is viable or not to run on a given system, depending on whether the microprocessor is powerful enough to compute the underlying processes at the required rate.

The main configuration parameters are (1) the CP length (CPL), which corresponds to the number of samples per cyclic prefix, i.e. the length of this signal (larger numbers lead to longer guard intervals, and less inter-symbol interference, while also increases the amount of signal overhead); (2) the OFDM payload length (PL), which corresponds to the number of samples per OFDM signal (larger PLs mean an increase in computational complexity); (3) the carriers per payload (CPP), the number of M-QAM symbol carriers per OFDM signal (proportional to the transmission bandwidth and inversely proportional to average power per carrier and, thus, SNR at the receiver's end); (4) the index of the first carrier (C_0), which corresponds to the index of the first frequency component of the OFDM payload signal, and defines the signal band usage together with the CPP; (5) the number of payloads per packet (PPP)

Table 2: System configuration parameters used for the performance test trial.

Parameter	Value
f_s	46.875 kHz
CPL	256 samples (≃ 5.4 ms)
PL	512 samples (≃ 10.8 ms)
CPP	120 carriers (≃ 11 kHz bandwidth)
$\mathbf{C_0}$	44 (f_0 ≃ 4 kHz)
PPP	16
ML	1024 (≃ 21.6 ms)

(with more payloads being packed into the same packet, there is more content data and less overhead caused by the preamble and mode-switch signals, while fewer payloads also mean finer-grained packets and less data buffer memory requirements); and (6) the mode-switch length (ML), defining the length of the mode-switch signal. This length influences the selectivity of the mode-switch pattern, i.e. by decreasing the likelihood of detecting false positives.

For system testing, the testbench configuration parameters were decided on based on empirical trials, by tweaking parameter values until reaching a setup that shows the best performance and reliability simultaneously. Table 2 provides a compilation of the system configuration parameters used for this test. Considering that packets are continuously transmitted, the theoretical maximum data transmission rate for this system configuration is around 3kB/s. However, the proposed approach can be utilized to reach transmission speeds of up to 30KHz with consistent behaviour.

6.2 Runtime performance analysis

The SEGGER Systemview toolkit was used to conduct a runtime performance analysis of the system implementation [19]. This application profiling tool allows us to track events issued by the system, such as the start and end of specific tasks, with microsecond precision, very low computational overhead, and all in a real-time fashion. This means that we can extract an almost exact model of the running system behaviour and, for this analysis, we are interested in measuring the computation times of different processes and determine the viability of different configurations.

Figure 4 showcases a sequence of processes running over a period of around 320 ms. It corresponds to the period in which a message packet is being interpreted, with each coloured block corresponding to a different process running from the master thread, the slave thread and other unrelated device modules. While there are other concurrent system processes, the following is a descriptive list of the most relevant types that explicitly appear as coloured blocks in the diagram due to higher processor time requirements: **black** blocks represent the *idle* process, which, roughly speaking, includes every process that is not directly related to the communication system; **red** blocks, here rather appearing as stripes, represent the master thread processes, i.e. the processes that are called upon the arrival of every new SAI DMA block; **green** blocks correspond to the *calibrate channel* task; and **blue** blocks correspond to the *OFDM decode payload* task, both performed by the slave thread.

At the start, the processor is mostly idle, while being periodically interleaved by the master thread. The red stripes, even though not always explicitly visible, appear at a fixed rate throughout the whole time-span, however, during this first phase, it is noticeable that these are thicker in form, indicating that the master thread is loaded with more substantial computations. In this case, the master thread is tasked with the *mode-switch detection* task.

Immediately following this initial phase, once the mode-switch signal is detected, the preamble related tasks are issued, and the slave thread takes over most of the processing time. After the preamble is transmitted, the OFDM payload decoding tasks are issued for every OFDM payload transmitted in sequence, appearing as the blue sections. Even though the different payloads are not explicitly distinguishable, for the most part, they should be separated by the black stripes. In essence, this indicates that the slave thread has successfully completed those computations on time, and can withhold of processor usage in the meantime. With this system configuration, the packet is composed of a total of 16 OFDM payloads, which can be confirmed by looking closely at the diagram. After decoding all 16 payload blocks, the cycle repeats, and the system returns to a mostly idle state, while the master thread goes back to running the mode-switch detection task once more.

A crucial metric that can be inferred from the runtime analysis is the processor load. From the previous analysis, we see that the system was running close to its computational load limit. Ideally, there should be more idle blocks in between the slave tasks, i.e. a longer slack time to decrease the restrictiveness of the deadlines.

6.3 Noise immunity

Assuming that CPL and PL remain constant, the system's data rates directly depend on the number of OFDM carriers used, which translates into the bandwidth of the signal. This spectral utilization should not only be adapted to the channel's passband but must also consider how the noise immunity varies with different numbers of OFDM carriers. For example, in the case of the speaker-pickup pair, where the 4 kHz to 16 kHz band typically shows the best SNR performance, the number of carriers can vary between 1 and 120, given the default system configurations presented in Table 2.

To test the system's noise immunity, we collected bit-error-rate (BER) data for different test signals artificially infected with additive gaussian noise. The results are shown in Figure 5, where the BER is plotted against the SNR for different configurations. From these empirical results, we define the noise immunity level as the SNR value at which the BER drops bellow 100 ppm (successful transmission of over 99 % of all packets). As expected, configurations with fewer carriers show improved noise immunity, and the noise immunity is directly proportional to the average power per carrier.

6.4 Testing the limits

To test the limits of the final system, we experimented with different settings of the system while trying to achieve the highest data rates and relatively reliable transmission. With $CPP = 160$, and all other configuration parameters set to the default values, we were able to wirelessly transmit a 160 kB message in 40 s without dropping a single packet (32kbps). Table 3 provides an overview of the basic working principles of the state-of-the-art introduced in Section 2 and our system. In this context, *Chirp.io* naturally offers

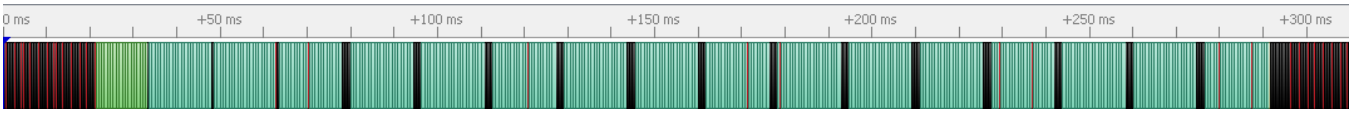

Figure 4: Performance profiling result with the default system configuration.

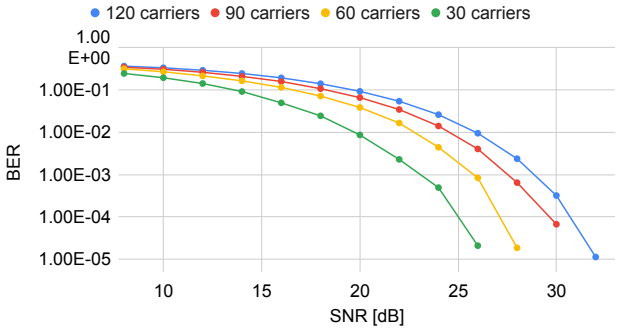

Figure 5: BER vs SNR with different bandwidth utilizations.

Table 3: DoS systems comparison overview.

	Chirp.io	Mutable	Ours
Platform	OS/baremetal	Baremetal	Baremetal
Encryption	optional	no	no (optional)
Data rate	Up to 1 kbps	N/A	Up to 32kbps
Distance	1 cm-100 m	N/A	1cm (extendable)
Spectral band	acoustic	N/A	acoustic
Direction	Simplex/Duplex	Simplex	Simplex
Signalling	M-ary FSK	QPSK	QAM16(OFDM)

the most mature solution. Nonetheless, we have focused on a contact wireless communication solution at low distances. Therefore, the comparison has to be done with the NFC range transmission settings with Chirp, as transmission at longer distances requires lowering the data rate by several orders of magnitude: 150bps at peer-to-peer range (3m) and 15bps at long range (100m) [2, 5]. In summary, our proposed solution reaches data rates that are an order of magnitude larger than existing DoS implementations.

7 CONCLUSION AND FUTURE WORK

We have presented a wireless communication solution to extend IoT connectivity to resource-constrained embedded devices through high-speed acoustic data transmission. The main contribution of this work has been the design and development of an OFDM-based one-way acoustic communication system that can be implemented in a wide variety of low-power microcontrollers. We have implemented the system and demonstrated that our proposed approach can be applied towards improving the connectivity of resource constrained devices, with little to no additional hardware costs. In this paper, we have covered aspects from both the design of the communication interface at the lowest level, as well as the implementation challenges and limitations. The experimental trials show that OFDM works as an effective signalling scheme by providing a simple method for configuration of the system's spectral utilization. With careful carrier placement, the resulting spectrum is such that the frequency band is adapted to the frequency response of the communication channel at hand. From a data format point of view, the proposed packets-based protocol provides a simple design for lossless transmission of independent messages. With the proposed approach, we have been able to obtain transmission speeds of one order of magnitude over the state-of-the-art and the most widely used commercial solutions. In future work, we will explore a fixed-point arithmetic based implementation, as well as introducing two-way communication and built-in encryption.

REFERENCES

[1] A. Bekasiewicz *et al.* 2016. Compact UWB monopole antenna for internet of things applications. *Electronics Letters* 52, 7 (2016), 492–494.
[2] A. Duke. 2019. Achieving Successful Data Rate Higher than 2.5kbps. https://blog.chirp.io/achieving-successful-data-rate-higher-than-2-5kpbs/. Transmission over sound record.
[3] A. Ericsson. 2016. Cellular networks for Massive IoT—Enabling Low Power Wide Area Applications. *no. January* (2016), 1–13.
[4] K. Chang. 2014. Bluetooth: a viable solution for IoT? [Industry Perspectives]. *IEEE Wireless Communications* 21, 6 (December 2014), 6–7.
[5] Chirp.io. 2019. Protocols. https://developers.chirp.io/docs/using-chirp/protocols.
[6] D. Bubley. 2017. Data over Sound Technology: Device-to-device communications and pairing without wireless radio networks. Disruptive Analysis Ltd.
[7] D. Chu. 1972. Polyphase codes with good periodic correlation properties (Corresp.). *IEEE Transactions on Information Theory* 18, 4 (July 1972), 531–532.
[8] E. Gillet. 2019. *eurorack*. Mutable Instruments SARL. https://github.com/pichenettes/eurorack
[9] G. Burns *et al.* 2016. Proximity detection of internet of things (IoT) devices using sound chirps. US Patent 9,438,440.
[10] H. Nogueira. 2019. Acoustic data transmission for embedded software platforms: an empirical study. University of Turku.
[11] I. S. Reed *et al.* 1960. Polynomial Codes Over Certain Finite Fields. (1960).
[12] J. Gubbi *et al.* 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems* 29, 7 (2013), 1645 – 1660.
[13] J. Peña Queralta *et al.* 2019. Comparative Study of LPWAN Technologies on Unlicensed Bands for M2M Communication in the IoT: beyond LoRa and LoRaWAN. *Procedia Computer Science* (2019). The 14th FNC.
[14] M. Shirvanimoghaddam *et al.* 2017. Massive Non-Orthogonal Multiple Access for Cellular IoT: Potentials and Limitations. *IEEE Communications Magazine* (2017).
[15] M. Sliskovic. 2001. Sampling frequency offset estimation and correction in OFDM systems. (2001).
[16] M. T. Anowar *et al.* 2016. A Survey of Acoustic Underwater Communications and Ways of Mitigating Security Challenges. (2016).
[17] R. S. Sinha *et al.* 2017. A survey on LPWA technology: LoRa and NB-IoT. *ICT Express* 3, 1 (2017), 14 – 21.
[18] S. Andreev *et al.* 2015. Understanding the IoT connectivity landscape: a contemporary M2M radio technology roadmap. *IEEE Comm. Magazine* (2015).
[19] SEGGER. 2018. *SEGGER SystemView User Guide*. SEGGER Microcontroller GmbH, In den Weiden 11, D-40721 Hilden, Germany.
[20] T. M. Schmidl *et al.* 1997. Robust Frequency and Timing Synchronization for OFDM. (1997).
[21] T. Yu *et al.* 2015. Handling a Trillion (Unfixable) Flaws on a Billion Devices: Rethinking Network Security for the Internet-of-Things. In *HotNets-XIV*. ACM.
[22] L. Li *et al.* 2011. The applications of WiFi-based Wireless Sensor Network in Internet of Things and Smart Grid. In *IEEE Conference on Industrial Electronics and Applications*. 789–793.
[23] Z. Sheng *et al.* 2015. Recent Advances in Industrial Wireless Sensor Networks Toward Efficient Management in IoT. *IEEE Access* 3 (2015), 622–637.