Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 8523078, 16 pages
https://doi.org/10.1155/2018/8523078

Research Article

BlinkComm: Initialization of IoT Devices Using Visible Light Communication

Toni Perković,[1] **Tonko Kovačević**,[2] **and Mario Čagalj**[3]

[1]*University Department of Forensic Studies, University of Split, Croatia*
[2]*University Department of Professional Studies, University of Split, Croatia*
[3]*Department of Electrical Engineering, FESB, University of Split, Croatia*

Correspondence should be addressed to Mario Čagalj; mcagalj@fesb.hr

Received 22 December 2017; Accepted 29 April 2018; Published 7 June 2018

Academic Editor: Dario Bruneo

Many applications from the Internet of Things (IoT) domain used in healthcare, smart homes, and cities involve a large number of interconnected wireless devices. To ensure privacy, confidentiality, and integrity of the information, devices should be initialized prior to any communication. In this paper, we present a secure initialization method for constrained IoT devices such as wireless sensors devices and/or actuators. The solution uses visible light communication (VLC) for the initial configuration of the IoT devices. The VLC system consists of a modulated light source such as a smartphone screen and a very simple photodetector. We analyze known coding and modulation techniques used for the VLC and propose BlinkComm, a differential coding technique that achieves threefold increase in transmission speed compared to existing solutions. We showed through experiments with 32 participants that the proposed solution achieves fast completion times and low error rates as well as high user satisfaction levels.

1. Introduction

Today we are surrounded by plethora of Internet of Things (IoT) devices that find their application in many areas of our everyday lives. These devices include numerous medical devices for patient monitoring, in smart homes for lighting, heating, cooling, or access control, in smart cities for emission and pollution control, etc. IoT devices can communicate with each other as well as with personal devices such as smartphones, tablets, smart TVs, or computers, through different wireless networking technologies (such as Bluetooth, ZigBee, WiFi, 6LoWPAN, and LoRaWAN). Many of these IoT devices use mobile communication techniques to establish a direct access to remote web servers. Some of the IoT devices can also interact with cloud servers indirectly using various WiFi proxies (WiFi access points or different hotspots). However, prior to any communication between these devices, they should be configured to ensure the privacy, confidentiality, and integrity of the information transmitted between them, as well as between devices and cloud system [1–4]. In other words, IoT device must be initialized to ensure secure communication.

The problem of initializing/bootstrapping secure communication between wireless devices presents a big challenge especially for devices such as *Proximity and Location Beacons* [5, 6] or *AWX IoT Buttons* [7], which do not have traditional user interfaces (such as keyboards or touchscreens) for interaction. Some commercial devices like Nest Protect [8], Belkin Wemo devices [9], and Fitbit Aria WiFi Smart Scales [10], as one of the solutions presented in [11], use a web-based configuration mechanism. Moreover, today there are numerous IoT devices on the market such as *Waspmote* from Libelium [12] or IoT devices from Link Labs [13]. Some of these commercially available solutions, such as SmartCitizen [14], require cable connection to convey keying information (such as SSID and password). In many cases users encounter very complicated configuration procedures when using these IoT devices, which present a major deterrent for accepting this technology especially given the fact that predictions talk about millions of IoT devices.

Some of the existing solutions for the secure initialization of IoT devices assume that the devices are preinstalled by the manufacturer [15–19] or even configuration is performed over an unreliable communication channel supposing the attacker will not be present during the bootstrap phase. The detected security vulnerabilities in smart bulbs [20–22] show that this access with preinstalled shared secret keys poses a great risk because compromising a single device can compromise the whole network or a whole set of devices. The solutions *Message-in-a-Bottle* (MiB) [23] and *KALWEN* [24] are fairly secure, but they are quite expensive and demanding for the end user because they rely on specialized hardware (Faraday's cage).

Various solutions based on multichannel protocols [25] for the secure initialization have been presented, where communication between network devices is performed over two channels. In these solutions, devices use an unsecured radio channel, and a specialized Out-of-Band (OoB) channel visible, infrared light, or even acoustic waves. Pan and Chen [26] propose a magnetometer based near-field communication protocol with transfer rate up to $110bps$ within $10cm$. However, the main drawback of the proposed solution is that the devices must have specialized hardware such as loop antenna and dedicated chip.

The use of visible light communication (VLC) for the initial configuration of the IoT devices (such as wireless sensor devices) presents an acceptable and commercially viable solution. Some existing solutions such as LIRA [27] and flashing displays [28] use Manchester coding in combination with "ON-OFF" keying, but that solution achieves transmission speeds up to 10 bps (20 Manchester encoded bps). Some commercial products such as ElectricImp [29] platform also uses flashing screens for information transmission over the VLC (using a smartphone and/or a tablet), and they use it to initialize a device with an SSID and a password. However, our tests show that the implemented coding technique enables transmission speeds at max $30bps$ which is still quite slow in terms of information transmission over the VLC. Jewell et al. [11] proposed a similar technique to convey information to sensor device by sending symbols in form of various brightness levels from smartphone screen to the end sensor device. The main drawback of the presented solution is the low transmission rate over the VLC channel resulting in 33 seconds to convey network name (SSID) and passphrase.

We experimentally estimate the capacity of VLC channel from screen as a transmitter to the photodiode as a receiver. The estimated capacity of $300bps$ indicates the requirement for finding coding and/or modulation techniques that achieve transmission speed larger than the proposed solutions. In this work, we propose BlinkComm, a differential coding scheme that achieves transmission rate up to $100bps$. This is more than a threefold increase in transmission speed compared to the existing solutions such as ElectricImp [29]. The proposed coding scheme operates in real-time and does not require any complex error detection or correction mechanisms. These are desirable characteristics given that typical IoT devices are constrained in terms of memory and processing power. We show that proposed coding scheme can be implemented using off-the-shelf components such as smartphones screen at transmitter side and a simple photodiode (or LED) at the receiver side. As an alternative transmitter, we could use LED as a transmitter that would achieve larger transmission speeds; however, smartphones or monitors are ubiquitous in our everyday lives and provide significant convenience and capabilities to users. Thorough usability tests with 32 users show us that the solution is easy to use and provides high satisfaction level and low error rate.

The rest of this paper is organized as follows: related work is given in Section 2, while in Section 3 we estimate the capacity of the visible light communication channel. We analyze known coding and modulation techniques used for the VLC in Section 4. In Section 5, we propose the novel coding technique which will increase the VLC transmission speed. The usability evaluation of the proposed solution is given in Section 6, while conclusions are drawn in Section 7.

2. Related Work

The main aim of this paper is to propose a solution that enables an end user to have a simple and secure initial configuration of IoT devices which does not require the use of any specialized hardware. In this section, we provide an overview of the other proposed solutions used for the initial configuration of wireless constrained devices.

There are many different solutions for the secure initialization of IoT devices which assume that the devices are preinstalled by the manufacturer or even configuration is performed over an unreliable communication channel assuming the attacker will not be present during the bootstrap phase [15–18, 30–33]. However, this way of initialization of IoT devices is insecure because an attacker can discover the secret information during the network setup phase.

The solutions *Message-in-a-Bottle* (MiB) [23] and *KALWEN* [24] are fairly secure because they rely on a physical specialized hardware such as Faraday cage. But these solutions are very expensive to deploy and very demanding for an end user. In the *Shake Them Up solution* [34], two devices are initially configured only when the user keeps them in their hands and shakes. There are similar schemes such as *Smart-Its Friends* [35] and *Are You with Me* [36] that are based on the movement of the devices during the initial configuration. These solutions require the use of a motion accelerometer and are intended for pairing two devices. They are not suitable for initializing a larger number of IoT devices because they require an additional effort from the user (shaking devices). Additionally, the security of these solutions can be compromised because two wireless devices can differentiate by radio fingerprinting [37].

A number of specialized solutions have been proposed for the applications in Wireless Body Area Networks (WBAN). These solutions are proposed in the works [38–42] and their main drawback is that they are intended for wireless sensors that measure the same physiological signals and are only applicable for applications in WBAN. An excellent and thorough survey of secure device pairing schemes, including a number that have been adopted as standards, can be found in [43].

There are numerous solutions for the secure initialization of wireless network devices based on multichannel protocols [25, 44, 45], where communication between network devices is performed over two channels. In these solutions, one channel is an unsecured radio channel, and the other is a special Out-of-Band (OoB) channel that uses visible or infrared light or even acoustic waves. In the HAPADEP [46] data are sent over an audio channel and thus the wireless devices have to be equipped with speakers and microphones. Perkovic et al. in [47] propose a solution that allows an unaided user to initialize a relatively large number of wireless devices. The proposed solution is based on a multichannel protocol in which information is transmitted over both a radio and VLC channel, where user performs the key verification by visually comparing synchronized LED blinking from wireless devices. Kovacevic et al. propose solutions LIRA [27] and *flashing displays* [28] which also exploit VLC for the secure initialization of IoT devices, while Gauger et al. [48] propose screen-to-photodiode communication from PDA to sensor node. These solutions are very user friendly, but the data transfer rate via VLC is very low, up to 10 bps. Other solutions such as Zhang et al. [49, 50] use VLC channel, in particular screen to camera channel to convey information from one device to another, while Saxena et al. [51] use VLC channel for mutual authentication of messages exchanged over a radio channel.

Jewell et al. in the work [11] address the challenge of supporting end-users in connecting low-power and low-cost WiFi devices with minimal user interfaces to an existing and secure WiFi infrastructure, but again the data rate via VLC is very low.

3. Estimating the Channel Capacity

The commercial available solution BlinkUp from Electric Imp [29] and the flashing solution [11] use the visible light communication to convey information from a transmitter (a screen of PC, tablet, or smartphone) to a receiver (a device equipped with a photo-receiver). The maximum transmission rate at which those solutions operate is 30bps. However, none of these techniques has estimated the channel capacity, i.e., the maximum allowable transmission rate at which those solutions may convey information. Therefore, in this section we first estimate the channel capacity used for the VLC experimentally from thorough tests on various smartphone devices. After estimating the channel capacity, in Sections 4 and 5 we identify and implement various coding techniques with transmission rates larger than existing solutions. This research is based on the results of the research presented in the doctoral dissertation given in [52].

We express the estimated channel capacity as a maximum transmission rate in bits per second (bps). In our channel model, we assume Additive White Gaussian Noise (AWGN) characteristic [53]. The noise at the receiver side is generated by the ambient light and noise from the DC signal component of the screen backlight (especially present on LCD screens). The maximum channel capacity can be expressed using Shannon-Hartley [54, 55] equation:

$$C = W_{fps} \cdot \log_2 \left(1 + \frac{S}{N}\right) [\text{bps}], \tag{1}$$

where W_{fps} is the screen frame rate (the number of frames the screen emits in one second), while S and N present the signal and noise power in Watts [W], respectively. It can be seen that the channel capacity depends on the frame rate W_{fps} which limits how fast the screen can emit information symbols, while S/N limits how much information can be inserted in every transmitted symbol. We also quantify the signal quality at the receiver side using the average S/N as

$$\frac{S}{N} = \frac{P_{avg}^2}{\sigma_n^2}, \tag{2}$$

where P_{avg} denotes the average light transmission intensity at the receiver side. On the other hand, σ_n^2 denotes *AWGN* ambient noise and the noise from the DC signal component from the screen backlight at the receiver side.

We estimated the channel capacity from experimental results conducted on various devices. During the tests we placed the receiver device with a photodiode at the top of the screen. The transmitting area of the flashing screen was $3 \times 3\text{cm}^2$, while a photosensitive area of the photodiode BPW34 was $7mm^2$ ($2.65 \times 2.65\text{mm}^2$). The photodiode was drawn 1.5mm into the device case to prevent the influence of ambient light on a transmitted signal (Figure 1(b)). Since the complete transmitting area of the screen is covered by the device (as shown in Figure 1), and the photodiode is close to the screen (1.5mm), we assume that both the impact of ambient light and path loss are negligible. In our tests on all screens, the brightness was set to maximum level with default contrast values. The signal on the input of the analog-to-digital converter presents the voltage drop on the resistor R (Figure 2), and we use analog-to-digital conversion to determine S/N ratio.

We estimate channel capacity for the original trace but also on filtered signal after using a moving average filter [56] to reduce random noise. The proposed moving average filter is simple to implement and only requires storing a sequence of M input data points at the receiver side. The filter simply averages a sequence of input data and can be expressed with the following equation:

$$u_i = \frac{1}{M} \sum_{j=0}^{M-1} x_{i-j}, \tag{3}$$

where x_i presents an input signal and u_i an output signal, while M is the sequence of data points in the moving average (frame of input data). Since our receiver samples data every millisecond, M data points correspond to the period of M milliseconds.

Figure 3 shows the gamma curve, i.e., the received light level values (P_i) for the generated gray colors (RGB colors from range (0,0,0)-(255,255,255)) using LG Spirit smartphone. As expected, we can see a nonlinear behavior

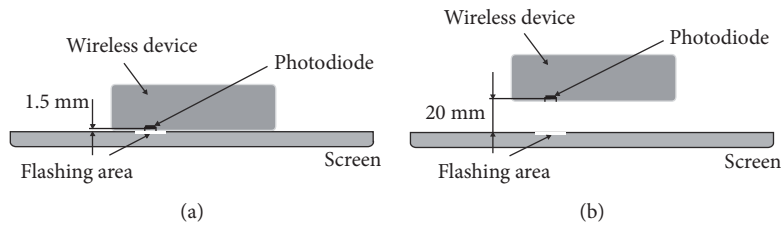

Figure 1: Implementation of receiver device comprising a simple photodiode. (a) Photodiode is placed at the bottom of the device to prevent the impact of ambient noise during the VLC. (b) We tested the impact of ambient light on correct signal reception and decoding afterwards by placing the device 2 cm from the screen surface.

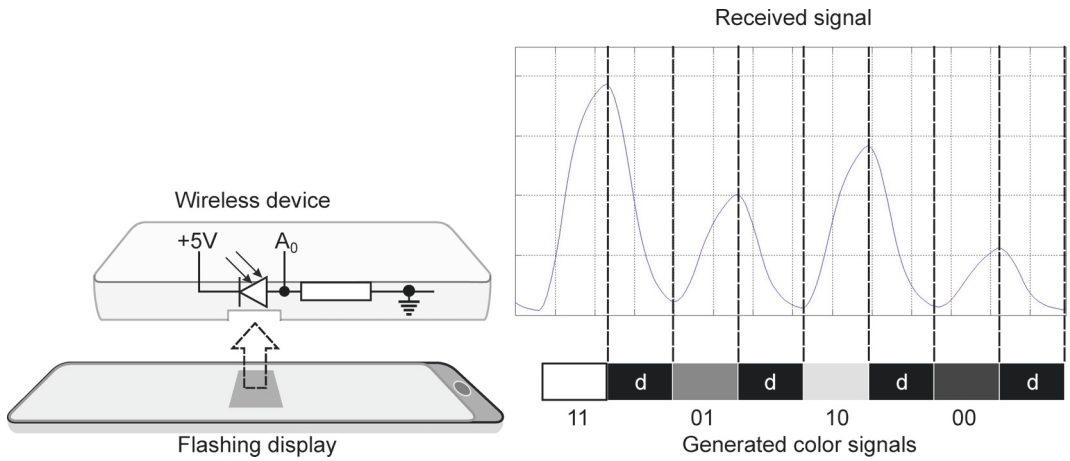

Figure 2: Transmission of data over the visible light communication from the display-equipped device to the receiver equipped with a simple photodiode.

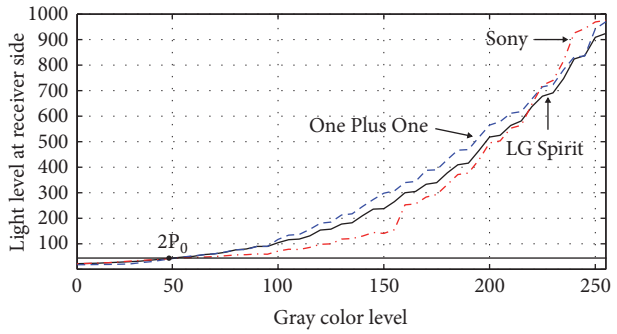

Figure 3: Gamma curve (gray color / light level) for three smartphones.

between gray color and luminance due to the nonlinear screen characteristics. Note that light level values look similar for small RGB values (gray colors). For this reason, and to avoid errors caused by the presence of the screen backlight and noise in the signal we placed the detection threshold of the receiver 2.5 times larger than P_0 (light intensity corresponding to the black color frame). The maximum level of the received signal corresponding to the gray color 255 is indicated by P_{255}, while the average power of the received signal S is given by

$$S = P_{avg}^2 = \left(\frac{P_{255} + 2.5 \cdot P_0}{2} \right)^2 . \tag{4}$$

Although we observe a nonlinear behavior in gamma curve (Figure 3), the average power S was calculated under assumption of a linear behavior in gamma curve, which presents an upper bound. Similarly, the noise power is calculated as

$$N = \sigma_n^2 = P_0^2 + \widehat{\text{var}}\,(P)\,, \tag{5}$$

where P_0 is the power of a DC signal component from the screen backlight (black frame), while $\widehat{\text{var}}(P)$ denotes the white noise power. The white noise power $\widehat{\text{var}}(P)$ was calculated from the variance of amplitude for every gray color signal $\text{var}(P_i)$, out of 256, as

$$\widehat{\text{var}}\,(P) = \frac{1}{256} \sum_{i=0}^{255} \text{var}\,(P_i)\,. \tag{6}$$

In Table 1, we can see the channel capacity estimated from the tests performed on LG Spirit, Sony Xperia Z3, and OnePlus One smartphone devices. The estimated channel

Table 1: The channel capacity estimation for three devices.

Parameter	LG Spirit - 2cm		LG Spirit		Sony Z3	One Plus
	Unfiltered	Filtered	Unfiltered	Filtered	Filtered	Filtered
S/N ratio	3.08	3.23	495.64	623.82	573.84	801.32
Capacity [bps]	61.08	62.77	269.60	279.66	273.48	290.42

capacity for LG Spirit device equals 269.60bps for the raw signal and 279.66bps for the filtered signal. It is important to emphasize that filtered signal is obtained after applying a moving average filter to a set of input signals, and, hence, we achieve better estimation of the channel capacity after eliminating the impact of random noise. While estimating channel capacity, the receiver samples data every millisecond, and to reduce random noise while keeping sharp step response, we selected a sequence of $M = 8$ samples for our moving average filter corresponding to 8ms of data, which is smaller than duration of one frame on modern devices (approx. 16ms). Estimated channel capacity for Sony Xperia Z3 and OnePlus One device is similar to LG Spirit smartphone (290.42bps and 273.48bps for OnePlus One and Sony Xperia Z3, respectively), since these devices use similar LCD technology.

While estimating the channel capacity we assumed a specific scenario in which the photodiode is placed at the bottom of the device and positioned at the screen surface, so that the impact of ambient noise is negligible. To cover a more general scenario where ambient noise is present during VLC (such as ElectricImp [29]) we also estimate the channel capacity for LG Spirit device in scenario in which the device (the photodiode) is placed 2cm from the screen surface leaving it exposed to the ambient light. Here the photodiode is exposed to the impact of ambient (surrounding) light from the fluorescent lamp placed 2m above the screen surface. The measurements were made in a room with brightness 1100 lx (for comparison, average brightness in the office is 500 lx). The channel capacity estimations for the raw and filtered signals were 61.08bps and 62.77bps, respectively. This indicates that surrounding light increases the noise and therefore reduces the channel capacity.

4. Analysis of the Channel Coding Techniques

In this section we analyze some known coding and modulation techniques that would bring the transmission speed closer to the estimated maximum channel transmission speeds (channel capacity) using flashing screens. Also note that in our model we assume receiver devices to be constrained in terms of memory and processing power. This way, we do not consider implementing any error detection or correction schemes because they require the receiver device to save the complete trace (samples) for further offline processing. Therefore, one of the goals is to propose coding or modulation techniques that operate in real-time and at the same time are resilient to errors.

The preference of choosing coding and modulation techniques depends on two aspects: transmission rate and resilience to errors during decoding. An overview of modulation and coding techniques to convey information over the VLC channel can be found in [57]. Before we introduce these techniques, we first describe the effects we observed on monitor-equipped devices that induce error during decoding phase.

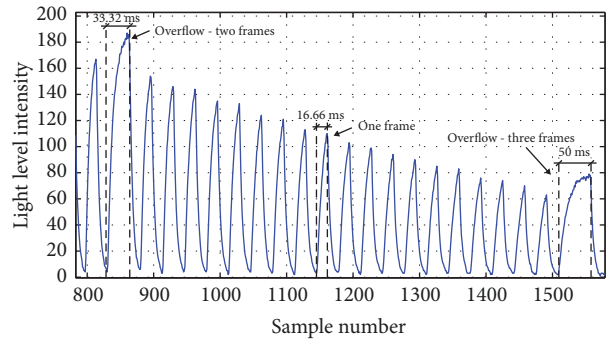

Figure 4: A snippet of the frame overflow with two and three consecutively repeated frames.

4.1. Frame Duration and Intersymbol Interference. The majority of screens today have refresh rate up to $60Hz$, meaning that it is possible to display 60 unique frames (figures) in one second [fps], leading to the duration of one frame up to 16.66ms ($1/60Hz$). Tests made on 30 devices (smartphones, tablets, and computer screens) using different LCD and AMOLED [54, 55] screens show that sometimes single picture repeats in two or three consecutive frames. This effect comes from the synchronization problem with frame presented by the graphic card and the screen refresh rate, called "jank" [58–61]. Throughout this paper, we refer to this problem as the "frame overflow".

This effect was noticed on monitors connected to PCs with Windows OS during VLC from application running in various browsers (Chrome, Firefox, and Internet Explorer). We noticed similar behavior on Android and iOS based smartphone and tablet devices equipped with application for information generation and transmission over the VLC. On both types of screens, LCD and AMOLED, we observe the frame overflow effect as shown in Figure 4 by alternating between a gray and black color pictures. As can be seen, one picture repeats in the two or three consecutive frames, leading to symbol duration twice or triple the expected duration (33.33ms or 50ms), respectively. If we assume that receiver is synchronized to the framerate of the screen (the duration of one symbol equals one frame), the presence of frame overflow where one symbol is sent in two frames may result in receiver decoding two symbols instead of one. During extensive tests on 30 devices, where we recorded 20 traces from every device,

we observed frame overflow with two frames on all 30 tested devices. On some devices the effect was observed only in one test, while on some devices the effect occurred twice during a single test. On two devices we observed one picture in three frames.

In addition, we also observed the presence of intersymbol interference, where the symbol light level interferes with the light level of the subsequent symbol. Taking into account the fact that the majority of devices have nonlinear behavior between gray color and luminance (gamma curve in Figure 3), the presence of intersymbol interference may result in an error during decoding phase.

Taking into account the above-mentioned effects of frame overflow and intersymbol interference, in the following section we analyze some existing coding and modulation techniques to convey information over the VLC.

4.2. on-off Keying, Manchester Coding, and Pulse-Width Modulation. The simplest way would be to use On-Off keying, where we switch from white to black frame to represent bits 1 and 0, respectively. Since the majority of screens have refresh rate up to 60Hz, switching status every 16.7ms would achieve transmission up to 60bps. The following modulation would operate under assumption in which sender device is synchronized with the receiver device, where symbol duration is exactly one frame. Unfortunately, due to the frame overflow, the receiver will interpret these frames as independent symbols.

One approach to recover from frame overflow would be to use Manchester coding (as proposed in [27, 28]), where bit "1" is encoded as "10", while bit "0" is encoded as "01". Using this coding technique, we can achieve speed up to 30bps Manchester encoded (15bps information bits). However, to ensure that the proposed modulation technique is resilient to frame overflow, it is necessary to increase the number of frames to present at least one Manchester encoded bit with no less than three frames, resulting in 10bps transmission speed of the original information bits (for 60Hz screen). In another solution presented by [11] along with black and white frame, a gray frame is proposed to recover from frame overflow if a sequence of two identical bits is transmitted ("0" or "1"). For example, if a sequence of Manchester encoded bits "1001" is transmitted, the first "0" bit is encoded with white or black color, respectively, while the second "0" bit will be presented with gray color. Using the proposed coding technique, around 65ms (4 frames) is required in [11] to convey one bit of information resulting in transmission speed of 15bps.

Another approach would be to use Pulse-Width Modulation (PWM), where the duration of one bit (e.g., bit "1") is longer than another (e.g., bit "0"). Also, to recover from frame overflow, here the transmission duration of longer bit has to be larger at least three times. For example, if we assume that the duration of bit "1" is 16.66ms (one frame), while the duration of "0" is 50ms (three frames), we can achieve a 30bps transfer rate.

4.3. Pulse Amplitude Modulation. Recalling from previous section, we have shown that it is quite difficult in practical situations to control screen framerate, where effects such as frame overflow require designers of coding and modulation schemes to introduce mechanisms to recover from errors during decoding phase at the price of reduced transmission speed. To counter these problems we propose Pulse Amplitude Modulation (PAM) to convey data. Let us denote the group of symbols as $\mathcal{S} = \{S_0, S_1, \ldots, S_{2^M-1}\}$, where S_i denotes the identity of ith symbol. In PAM modulation one data symbol consists of M data bits, where every symbol is translated to a set of predefined 2^M gray color data frame. To provide immunity against intersymbol interference, a guard interval consisting of one black color frame is added to every data symbol. The transmission of data information is composed of alternating between a gray data frame and a guard black frame. The receiver equipped with a photodiode interprets these data symbols as predefined 2^M light levels. Similar PAM modulation technique was proposed by ElectricImp [29] for $M = 1$, which results in 30bps transmission speed. In this paper, our goal is to find modulation and coding techniques that achieve speeds larger than proposed solutions.

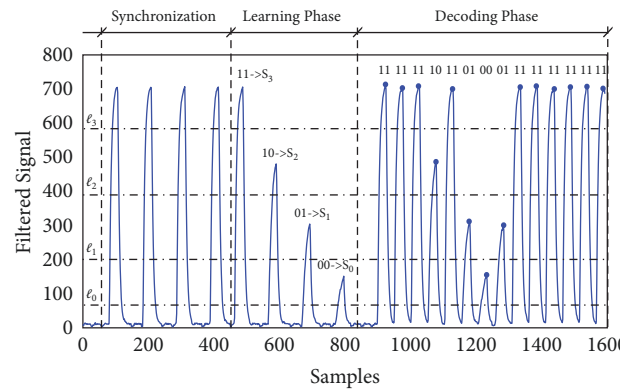

FIGURE 5: An example amplitude modulation with $M = 2$ and a delimiter sequence.

Prior to the start of data transmission, the VLC transmitter sends synchronization and preamble (learning) frames as shown in Figure 5. From synchronization frames, the receiver learns the beginning of VLC transmission as well as the sender frame rate. Synchronization frame comprises sequence of white color frame and three black color frames, transmitted four times (Figure 5). The sender uses the synchronization sequence to distinguish between transmissions from other ambient light sources with similar transmission period, such as a fluorescent lamp with 100Hz light transmission frequency. After synchronization phase, a preamble is sent in learning phase to learn threshold for every data symbol (out of 2^M data symbols). The preamble comprises transmitting 2^M data symbols to learn amplitudes of every symbol frame. As can be seen in Figure 5 in this phase we also add to every data frame a guard black frame to recover from the impact of intersymbol interference. The preamble is transmitted so that the set $\mathcal{S}$ is ordered with respect to the assigned gray colors ($S_{2^M-1} > S_{2^M-2} > \ldots > S_0$). By sending preamble the receiving device learns amplitudes A_i for every data symbol S_i, the delimiter amplitude d (black frame), and calculates thresholds ℓ_{i-1} as mean between two symbols A_i

and A_{i-1}, as well as between A_1 and delimiter d (threshold ℓ_0).

After learning the threshold values the communication of data symbols starts by alternating between the data symbol frame and black guard frame (Figure 5). The decoding is performed in the following way:

(1) When sampled value goes above the delimiter threshold ℓ_0, find the maximum sample amplitude value A_j.

(2) When sampled value goes below the threshold ℓ_0, check the position of previously sampled amplitude A_j within thresholds ℓ_{j+1} and ℓ_j and decode it back into data symbol S_j.

(3) Decode symbol S_j into data bits.

The receiver sampling speed is one sample per millisecond, while the sender frame rate is one frame every 16.7ms (60Hz screen). Please note, to eliminate noise during transmission over the VLC, the sampled signal in both learning phase and decoding phase passes through a moving average filter. However, the main drawback of the proposed method is that only half of frames are used for data transmission, resulting in the transmission speed up to $M \cdot W_{fps}/2$ data bits. Also note, by introducing non-data guard frame the effect of frame overflow can be easily detected and eliminated. It is very easy for the decoder to recover from frame overflow since between two guard frames only one symbol can be transmitted, despite the presence of two (or three) picture frames. During the decoding process, the receiver has to find the maximum amplitude light level of symbol between two guard frames, which makes the solution easy to operate in real-time. In Section 5 we explore other amplitude modulation techniques we use to minimize the number of non-data frames aimed at increasing the transmission speed.

We tested PAM for $M = 1$, $M = 2$, $M = 3$, $M = 4$ and 5 data bits. For $M = 1$ and 2 data bits we did not observe any error during decoding phase on all tested 30 devices. It is interesting to note that we also successfully tested the proposed modulation technique within the presence of ambient light. We placed the device 2cm from the screen surface and successfully tested transmissions with $M = 2$ data bits (Figure 5) that achieves speeds up to 60bps in the room with brightness level 500 lx.

For $M = 3$ ($2^{M=3} = 8$ symbol/gray colors), we observed error during decoding phase with some devices. Since gamma curves are nonlinear and unequal on all devices (Figure 3), it is difficult to choose gray colors for symbols that would result in similar threshold light values on all tested devices. Due to the impact of intersymbol interference and small threshold values, some symbols will be incorrectly decoded (despite the presence of guard interval frame between two consecutive data frames). We detected errors during decoding phase on all tested devices for $M = 4$ and 5 symbols.

Recall, our goal was to create a simple coding technique that operates in real-time, without any additional requirements for complex error detection or correction techniques that require from devices (sometimes constrained in terms of memory) to save the complete trace for offline sample processing. In this section, we introduced PAM modulation technique that transmits non-data frame half the time during symbol transmission. Although non-data frame was used to recover from intersymbol interference and frame overflow, the overall transmission period of data symbols is reduced by half; i.e., out of 60 frames, 30 frames are used for data transmission. In the following section, we propose modulation technique aimed at reducing the number of non-data frames.

5. BlinkComm: Differential Coding Technique

In this section, we propose a signal modulation technique where every consecutive symbol will be presented with light level different from the previous one, which will allow us to create a receiver/decoder that only detects sample difference, BlinkComm. More particularly, we introduce two BlinkComm coding schemes: differential coding with two symbols (BlinkComm2) and differential coding with four symbols (BlinkComm4) with transmission speed up to 60bps and 100bps, respectively.

As we show later, the proposed BlinkComm2 coding does not require a non-data delimiter symbol to recover from frame overflow and intersymbol interference. Compared to PAM modulation with two symbols, simply by eliminating a non-data delimiter frame, the overall transmission speed significantly increases.

5.1. Differential Coding with Two Symbols (BlinkComm2). In this section, we introduce a differential coding technique with a main idea to present two neighboring symbols with unequal gray color levels that will allow us to create a receiver based on detecting difference in light levels between the sampled values. This form of coding would allow us to create a modulation technique without any requirement for non-data frame, but still resilient to intersymbol interference and frame overflow. In the proposed differential coding technique with two symbols (namely, BlinkComm2) we use three gray frames to convey information over the VLC: white frame, gray color frame, and a black frame. The learning preamble is identical as in PAM modulation. Table 2 gives a brief explanation of symbols used in this sections as well as their meaning. The coding of BlinkComm2 technique can be described as follows:

(i) If data symbol S_1 is transmitted after symbol S_0, then S_1 is presented with a white frame. Also, if data symbol S_0 is transmitted after symbol S_1, then S_0 is presented with a gray color frame.

(ii) When a sequence of identical symbols is transmitted in a row, the first symbol is presented with assigned color (S_1 with white or S_0 with gray color, respectively), the second symbol is presented with a delimiter frame (black screen), the third once again with assigned color, etc.

As can be seen in Figure 6, BlinkComm2 signal coding will result in signals at the receiver side where neighboring symbols are presented with unequal light level. As we show, simply by calculating difference in light level between two

Input: sequence $\mathcal{Z} = \{z_1, z_2, \ldots, z_k\}$ and $\mathcal{U} = \{u_1, u_2, \ldots, u_k\}$ and threshold values ℓ_0, ℓ_1

```
(1)  procedure BlinkComm2 decoding
(2)    i ← 0
(3)    while true do
(4)
(5)      if ((z_{k/2-1} ≥ 0 AND z_{k/2} < 0) OR (z_{k/2} is local max in Z AND z_{k/2} ≤ 0)) OR
             ((z_{k/2-1} < 0 AND z_{k/2} ≥ 0) OR (z_{k/2} is local min in Z AND z_{k/2} ≥ 0)) then
(6)
(7)        if u_{k/2} ≥ ℓ_0 AND u_{k/2} < ℓ_1 then
(8)          S^i = S_0
(9)
(10)       else if u_{k/2} ≥ ℓ_1 then
(11)         S^i = S_1
(12)
(13)       else if u_{k/2} < ℓ_0 then
(14)         S^i = S^{i-1}
(15)     U.push(u)          ▷ Pushing new sample u to the sliding window U
(16)     Z.push(z)          ▷ Pushing new sample z to the sliding window Z
(17)     i ← i + 1
```

Algorithm 1: BlinkComm2 decoding.

Table 2: Symbols used in Section 5 and in Algorithm 1.

Notation	Meaning
ℓ_0	delimiter threshold level
ℓ_1	threshold level between symbols S_0 and S_1
S_0	symbol used to encode bit 0
S_1	symbol used to encode bit 1
S_i	ith decoded symbol
x_i	sampled light level at ith interval
u_j	moving average filter applied over a sequence x_i
z_j	difference between two light signals u_i and u_{i-1}
$\mathcal{Z}$	Sliding window containing k successive samples z_i
$\mathcal{U}$	Sliding window containing k successive samples u_i

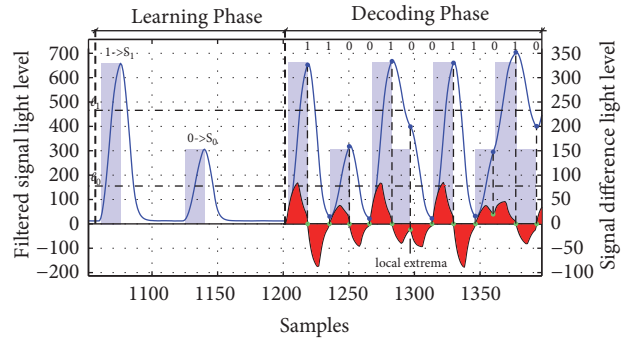

Figure 6: An example of BlinkComm2 coding and decoding procedure: every symbol is presented with given gray color level. When two identical symbols are transmitted, first symbol is presented with gray color, while the second with a delimiter frame. The decoding process is based on calculating signal difference between consecutive samples to detect symbol peaks.

neighboring samples, we can create a decoder that operates in real-time. More particularly, we have to detect signal peaks (both minimum and maximum) that correspond to symbols. To accomplish this, we use the first derivative of sample values (sample difference). As can be seen in Figure 6, the first derivative of the peak has downward-going zero-crossing (over the x-axis) at the peak maximum. Also, the first derivative of the minimum has upward-going zero-crossing at the peak minimum. Either way, our goal is to find a point where the derivative signal crosses the x-axis going either from positive to negative or vice versa. In specific scenarios (described below), the first derivative of a peak maximum or minimum also has a local extrema at the peak maximum. At the reception side, decoding of an output signal is described by an Algorithm 1:

(i) After learning threshold values ℓ_0 and ℓ_1 in learning phase (Figure 6), the receiver calculates signal difference z_i from two output signals u_i and u_{i-1} by calculating $z_i = u_i - u_{i-1}$.

(ii) The samples z_i and u_i are stored within two sliding windows $\mathcal{Z}$ and $\mathcal{U}$ of sample size k, respectively. The sliding window size depends on the receiver sampling speed, as well on the sender frame rate (recall, the receiver learns frame rate in synchronization phase in Section 4.3). The duration of one frame is $16.6ms$ (for $60\,fps$ screens), while the receiver sampling period is $1ms$. In our implementation, we used the sampling window size between half frame and full frame size to capture first derivative zero-crossing and local extrema of the signal difference at the peak maximum/minimum ($8 < k < 16, k = 2 \times m$).

(iii) The receiver checks whether the signal difference between two samples within the center of sliding window $\mathcal{Z}$ ($z_{k/2}$ and $z_{k/2-1}$) has a downward or upward-going zero-crossing. Alternatively, the receiver checks

if a sample within the center of a sliding window $z_{k/2}$ is a local extrema and only considers those samples that satisfy the conditions of local maximum for $z_{k/2} \leq 0$, or conditions of local minimum for $z_{k/2} \geq 0$, as can be seen in Figure 6.

(iv) After finding these samples, the receiver looks at the signal value $u_{k/2}$ within the sliding window $\mathcal{U}$ and performs the following decision: if $u_{k/2}$ is larger than threshold ℓ_1 the symbol is decoded into symbol S_1, if $u_{k/2}$ is between thresholds ℓ_0 and ℓ_1, then it is decoded into symbol S_0 (Figure 6). Alternatively, if $u_{k/2}$ is smaller than ℓ_0, then the symbol equals the previously decoded symbol.

Using BlinkComm2 coding technique, we can accomplish transmission speeds equal to the frame rate of modern smartphones, i.e., 60bps for 60fps screens. Moreover, at the small price of additional two memory buffers of size $2 \times k$ bytes we can achieve a coding technique that is resilient to frame overflow and intersymbol interference and operates in real-time. Compared to PAM modulation with four symbols, here we require only two symbols to achieve identical transmission speed up to 60bps (along with black frame in both modulations). Since BlinkComm2 coding uses two symbols, the difference between threshold levels is large enough so we can consider the impact of intersymbol interference to be negligible, as shown in Figure 6. Indeed, tests with smartphones showed resilience to errors during decoding phase.

5.2. Differential Coding with Four Symbols (BlinkComm4). In this section, we show how to increase the transmission speed using differential coding technique to get it closer to the estimated channel capacity (Section 3). Compared to differential coding technique with two symbols, here four symbols are used to convey information over the VLC channel.

The coding in BlinkComm4 goes as follows (Figure 7) and can be distinguished in four possible cases.

Case 1 ($|S^i - S^{i-1}| = 1$). When the absolute signal difference between two symbols is 1 ($|S^i - S^{i-1}| = 1$), every data symbol is assigned with gray color.

Case 2 ($|S^i - S^{i-1}| = 0$). When two identical symbols are transmitted the first symbol will be presented with assigned color frame, and the second one with a delimiter frame.

To further explain scenario coding when the difference between two symbols is larger than 1 ($|S^i - S^{i-1}| > 1$) we have to observe a sequence of three symbols.

Case 3 ($(|S^{i-1} - S^{i-2}| = 1) \wedge (|S^i - S^{i-1}| > 1)$). In scenario where a sequence of three symbols is transmitted such that the difference between first two equals 1 ($|S^i - S^{i-1}| = 1$) while the difference between the second and third symbols is larger than one $|S^i - S^{i-1}| > 1$, the first two symbols are presented with assigned gray colors which is followed by one delimiter d (black frame) and the transmission of third symbol presented with assigned gray color.

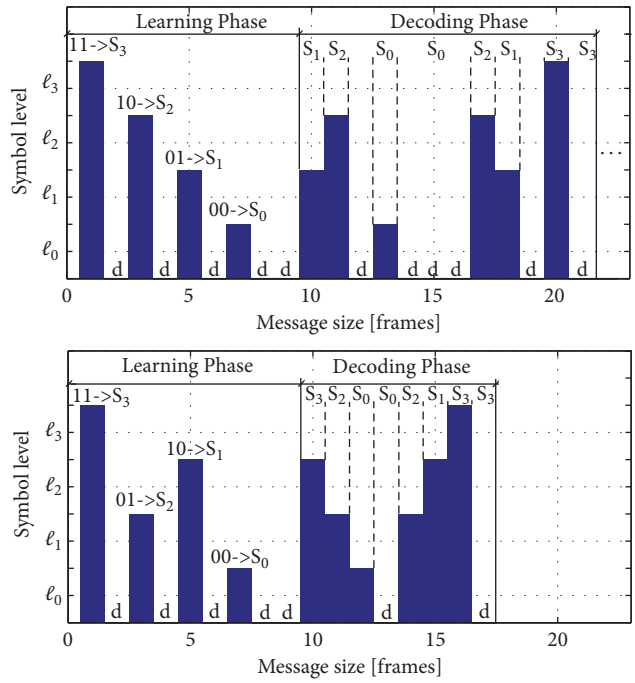

FIGURE 7: An example of BlinkComm4 symbol transmission (up). If two identical symbols are transmitted, a triple delimiter is inserted. Also, a single delimiter is inserted if difference between two symbols is larger than one. Optimizing/minimizing the number of non-data symbols by alternating the symbol signal level in the learning phase (down).

Case 4 ($(|S^{i-1} - S^{i-2}| = 0) \wedge (|S^i - S^{i-1}| > 1)$). In scenario where a sequence of three symbols is transmitted such that first two symbols are identical ($|S^i - S^{i-1}| = 1$) while the difference between the second and third symbols is larger than one $|S^i - S^{i-1}| > 1$, the first symbol will be presented with assigned color frame, second one with three delimiter frames, and the third symbol again with assigned gray color. By inserting three delimiters (instead of one as can be seen in Figure 7 (up)), our goal is to differentiate between scenarios in which one delimiter frame is inserted between two neighboring symbols with absolute signal difference larger than 1 ($|S^i - S^{i-1}| > 1$), such that the symbol S^{i-1} is presented with assigned gray color. We use three delimiters instead of two to recover from possible frame overflow effect; recalling from Section 4.1, there is negligible probability that one picture (delimiter) repeats in three consecutive frames due to frame overflow, but also negligible probability that frame overflow occurs with all three consecutive pictures.

5.3. Evaluating the Expected Rate of Coding Scheme DS4. We evaluate the rate under the assumption of uniformly distributed symbols $\{S_1, S_2, S_3, S_4\}$. Referring to four cases described in previous section, the expected rate is given by the following expression:

$$r = \frac{1}{E(T)} \frac{[sym]}{[s]} = \frac{2}{\sum_{j=1}^{4} P_j[] \cdot T_j} \frac{[b]}{[s]}, \tag{7}$$

where $P[]$ denotes the probability of symbol transmission given the four cases introduced in previous section, while T_j denotes the symbol transmission time.

Case 1 ($|S^i - S^{i-1}| = 1$). There are 6 possible symbol combinations in which the absolute difference between two neighboring symbols equals one (out of 16). Therefore, the probability that the symbol is transferred within time period of one frame ($T_1 = (1/60)s$) equals

$$P_1\left[\left|S^i - S^{i-1}\right| = 1\right] = \frac{6}{16}. \tag{8}$$

Case 2 ($|S^i - S^{i-1}| = 0$). There are 4 possible combinations where two identical symbols are transmitted in sequence, and the probability that the symbol will be transmitted within one frame ($T_2 = (1/60)s$) also equals

$$P_2\left[\left|S^i - S^{i-1}\right| = 0\right] = \frac{4}{16}. \tag{9}$$

Case 3 ($(|S^{i-1} - S^{i-2}| = 1) \wedge (|S^i - S^{i-1}| > 1)$). For the sequence of three symbols S^i, S^{i-1}, and S^{i-2}, there are 64 possible unique sequence combinations. The favorable combinations of symbol/frames are ones that require three frames to convey three symbols. There are overall 24 unfavorable combinations that reduce the transmission speed because of inserting at least one non-data delimiter (Figure 7 (up)). Out of 24 unfavorable combinations, in 18 combinations we insert one non-data delimiter d in scenario in which the absolute difference between two neighboring symbols is larger than one ($|S^i - S^{i-1}| > 1$). Therefore, the probability that the symbol is transferred within the time period of $T_4 = (1.5/60)s$ is given by the following equation:

$$P_3\left[\left(\left|S^{i-1} - S^{i-2}\right| = 1\right) \wedge \left(\left|S^i - S^{i-1}\right| > 1\right)\right] = \frac{18}{64} = \frac{4.5}{16}. \tag{10}$$

Case 4 ($(|S^{i-1} - S^{i-2}| = 0) \wedge (|S^i - S^{i-1}| > 1)$). For three consecutive symbols in which first two symbols are identical, while the absolute difference between the second and the third is larger than one, the time required for transmission of one symbol equals $T_4 = (1.67/60)s$, while the probability that symbol will be transmitted within this period is given by

$$P_4\left[\left(\left|S^{i-1} - S^{i-2}\right| = 0\right) \wedge \left(\left|S^i - S^{i-1}\right| > 1\right)\right] = \frac{6}{64} = \frac{1.5}{16}. \tag{11}$$

By plugging in the expressions (8), (9), (10), and (11) in (7), we obtain the expected transmission speed $r = 99.74bps$.

To validate the expected transmission rate, we implemented a simulator in MATLAB, where we generated a sequence of 100 BlinkComm4 symbols, each sequence randomly generated 1,000,000 times.

Figure 8 shows the transmission speed calculated with 95% confidence interval. On average, the transmission speed is 100bps, which corresponds to the expected rate of 99.74bps. Also, the smallest achieved transmission speed is around 95bps, while it is possible to achieve transmission speed up to 104bps. The transmission speed of BlinkComm4 modulation is decreased due to the presence of non-data symbols used to recover from intersymbol interference.

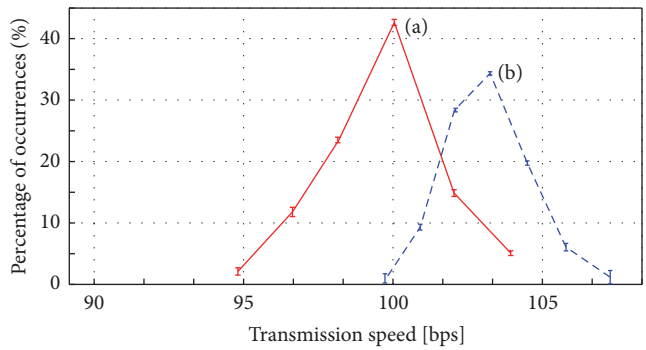

FIGURE 8: (a) Transmission speed distribution with confidence interval of 95% for BlinkComm4 modulation technique. (b) Transmission speed of BlinkComm4 modulation after applying optimization at the source encoder.

Hence, to reduce the number of negative transitions and thus non-data frames, we apply a simple optimization technique prior to beginning of transmission (at the source encoder). Let us take a look at the following example shown in Figure 7 (up). Recalling from Section 4.3, in learning phase the symbols are ordered according to the assigned gray colors, in a way that larger symbol (its identity) is assigned with larger gray color (and thus larger light level). However, note that gray colors can be assigned to any of four possible symbols, which results in 24 (4!) possible permutations of unique symbol/gray color combinations in the learning sequence. By alternating between various combinations of symbol/gray color we can achieve faster transmission speeds (Figure 7 (down)). Our goal is to find such combination that will reduce the number of non-data symbols in transmission phase. Please note, the decoding process is still performed in real-time (online) by the receiver devices and it is not computationally demanding for them.

Figure 7 (down) shows the transmission of sequence after applying optimization of the source encoder. As we can see, 17 frames are now required to convey all data over the VLC channel (compared to initial 21 frames). The simulation results indicate (Figure 8) that the average speed is approximately 103bps (compared to 100bps). In addition, in the most unfavorable scenario, the transmission speed will not go below 100bps, while the average transmission speeds can go up to 107bps.

5.4. Discussion. Various signal coding techniques were proposed to convey information over the visible light communication channel to ensure high transmission speed with low error rate. All discussed techniques were implemented without using any error correction and detection techniques with the main goal to create a simple coding solution applicable to resource constrained devices. Due to the small

Table 3: Proposed coding techniques regarding transmission speed and resilience to errors.

Coding technique	Max. transm. speed [b/s]	Resilience to Frame Overflow	Resilience to *ISI*
Manchester Coding	15	−	+
Two symbols and delimiter	30	+	+
Electric Imp [29]	30	+	+
Four symbols and delimiter	60	+	+
Eight symbols and delimiter	90	+	−
Differential BlinkComm2	60	+	+
Differential BlinkComm4	107	+	+

amount of available memory, signal decoding should operate in real-time without any requirement for storing the complete signal sequence in memory.

Table 3 gives the comparison of the coding techniques discussed in this section regarding transmission speed and resistance to errors caused by frame overflow and intersymbol interference. Manchester coding technique is very simple and resilient to frame overflow and intersymbol interference, but it results in a relatively low transmission speed of just 15*bps*.

Coding with two symbols and delimiter sequence is very easy to implement and ensures transmission speeds up to 30bps with resilience to frame overflow and intersymbol interference. This coding has already been proposed by Electric Imp [29]. Coding technique with four symbols and delimiter sequence provides transmission rates up to 60bps with resistance to intersymbol interference and frame overflow. Further increase in the number of levels leads to eight symbols and delimiter sequence with transmission speeds up to 90bps. However, our test on 30 devices (monitors and smartphones) shows that the proposed solution is not resistant to intersymbol interference on some smartphone devices (Section 4.1). On the other hand, differential coding with 2 symbols (D2A) is very easy to implement and provides transmission speeds up to 60bps with good resistance to frame overflow and intersymbol interference. The maximum transmission speed of 107bps is achieved using differential coding technique with 4 symbols (BlinkComm4) after applying optimization at the source encoder. This encoding technique is also characterized by high resilience to frame overflow and intersymbol interference.

It is also important to note that the estimated channel capacity is approximately 300bps, as shown in Section 3. The use of differential coding technique with four symbols ensures transmission speed of approximately 100bps, which is one-third of the estimated channel capacity. As we showed throughout the paper, errors from intersymbol interference and frame overflow restricted us to focus on finding such coding techniques that would minimize their impact but still provide sufficiently large transmission speed, which are still larger than commercially available solutions. At the same time our goal was to create the solution that does not require from devices to store the entire sampling sequence but instead operates in real-time. Future work will focus on the finding coding techniques that will increase the transmission speed close to the estimated channel capacity.

6. Usability Evaluation of Wireless Sensor Device Connection to Cloud Server

In this section, we show one possible implementation of the proposed coding method over the VLC channel. In the proposed solution, the wireless sensing device wishes to send sensor data to the cloud over the WiFi channel. Since the device initially does not have keying information (SSID and password) about neighboring access points, we use visible light communication channel to easily convey information from the screen to the wireless sensing device, so the device can send data to cloud server via access point. This is particularly important since wireless sensing device is not equipped with user interface such as keyboard and/or display. More notably, compared to cable initialization, with VLC channel users would not have to perform any additional steps such as installation of specialized software or driver to detect wireless device connected to it. In the proposed solution VLC channel is used to convey information from display-equipped device to the wireless communication sensing device. Once the device receives information, it is ready to connect to the AP and send sensor data to the cloud (temperature and humidity in our case). Our goal in this study was to test the hypothesis whether users find the proposed key deployment solution with screens easy to use. Participants were mainly students of electrical engineering and people working in the ICT field.

6.1. Implementation. In this section, we describe a detailed implementation of the system used to convey information over the VLC from screens. We investigate the characteristics of signals in modern screens to convey information over the VLC. Our goal is to implement a cheap and simple receiver that can detect and decode information transmitted from modern display devices in real-time, without significant memory requirements, as wells as requirements for complex error detection and correction schemes. The receiver is also a device that has limited energy, memory, and computational power and lacks rich user interfaces. The transmitter is designed using off-the-shelf components such as smartphone or display screen. The sender embeds data within signals transmitted in form of light generated by the screen. The receiver captures the signals in form of light intensity and decodes it back into the data.

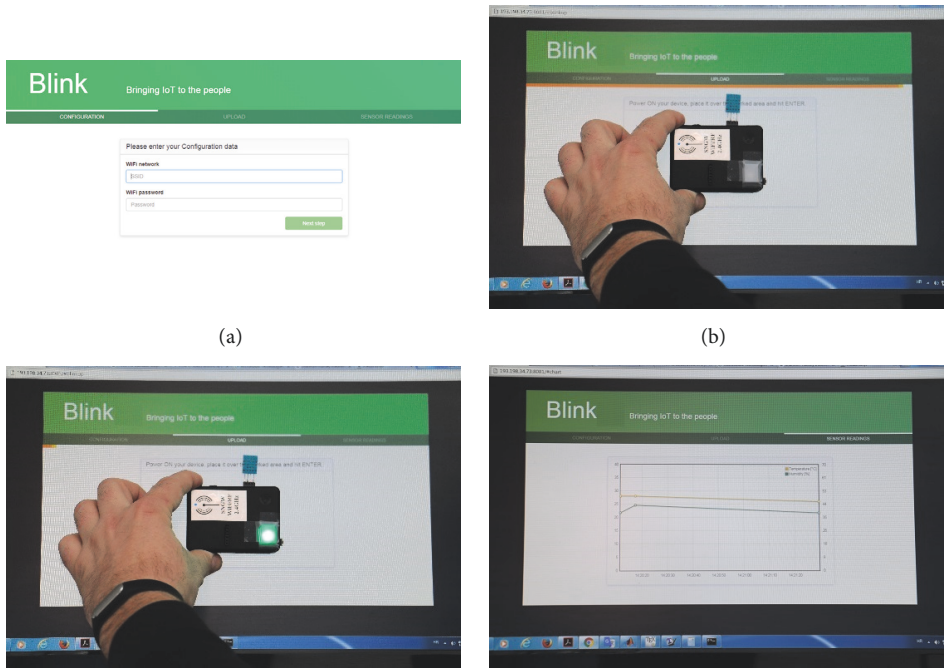

FIGURE 9: The wireless device initialization procedure: (a) the user enters SSID and password of the access point the wireless device connects to; (b) user places the device on the top of the screen at designated area; (c) after the device initialization is over, the wireless device initiates the connection to the access point; and (d) wireless device sends sensor data (temperature and humidity) for visualization in real-time in the browser.

For information transmission via VLC, we used PAM modulation technique that uses 4 symbols and non-data delimiter (guard) symbol d (as described in Section 4.3) that achieves transmission speeds up to 60bps (for 60Hz screens). The transmitter was a classic 60Hz monitor connected to the PC with Windows OS, while for information transmission we implemented a web application (Blink) as shown in Figure 9.

Our receiver is a simple reverse biased photodiode that measures light intensity from the screen. In our implementation of the receiver we used BPW34 [62] photodiode connected to the analog input of an Arduino microcontroller as shown in Figure 2. More precisely, we used Arduino Uno platform [63] based on ATmega328, 32 KB flash memory, 2 KB SRAM, 1 KB EEPROM, 16 MHz clock speed, and WiFi module ESP8266 [64]. The signal sampling frequency at the analog input is 1 kHz, which means that samples are taken every millisecond. Although the screen refresh rate is 60 Hz, the signal is sampled at a significantly higher frequency (f_s = 1 kHz) to eliminate the noise impact in the channel but also to obtain a good quality signal image with a satisfactory response time. The signal reception time at the analog input of Arduino is 100 μs, which leaves sufficient time for the information processing in real-time with Arduino.

The receiver sensitivity (Figure 2) depends on the reverse polarization voltage of the photodiode and the resistor R. Also note that signal from the photodiode is brought at the analog pin of Arduino Uno that uses a 10-bit analog-digital converter, which indicates that the received signal level can range from 0 to 1023. By applying a signal up to 5V, the resolution of an analog input will be 4.9 mV ($5\ \text{V}/2^{10} = 5\ \text{V}/1024 = 4,9\ \text{mV}$). We should also take into account if the signal enters saturation, the receiver will not be able to show light intensity larger than 1023, a maximum voltage of 5V on input of analog-digital converter. To keep the reception signal within the saturation limits, we tested the signal transmission from the screen of 30 different devices with the photodiode reverse polarization voltage of 5V and the resistance R values ranging from 1MΩ to 1.8MΩ.

In our implementation, we used the inexpensive and commercial off-the-shelf BPW34 photodiode sensitive to visible light signal. We could also use LED as a receiver (as proposed in [65]), since LED is most sensitive to the incoming wavelength around the wavelength radiating itself due to the so-called MIMS effect [66, 67]. Other analog photosensitive elements could also be used such as photo-resistors and photo-transistors or ambient-level digital sensors, such as the BH1730FVC sensor [68].

6.2. Usability Evaluation. Overall, 32 participants took part in our study (Table 4). To fully understand in which part of the usability evaluation errors were conducted by users the complete evaluation was recorded with a camera. Although a formal IRB review was not required at our university, we took all possible measures to make sure that all legal and ethical issues were properly handled. For instance, all users were well informed in advance (before the user studies) about the purpose of the study and how the data would be processed and used in our paper. All the data collected

Table 4: Users' demographic as well as touchscreen and Internet usage.

Years			Using devices with screen (hours/day)				Familiar with VLC	
< 25	25-30	> 30	0-1	1-2	2-4	> 4	Y	N
12	7	13	0	4	10	18	18	14
Feel secure using WiFi			**Using TS devices (hours/day)**				**Gender**	
Y	N	Neutral	0-1	1-2	2-4	> 4	Male	Female
12	11	9	4	13	7	8	20	12

TS: touchscreen; VLC: visible light communication.

from users was shared only among the coauthors of the paper. The task of every user was to connect the wireless sensing device to a cloud system via access point using the proposed initialization via flashing screen. If in any phase of the device initialization user performs an error, he/she will be noticed with a continuously blinking red LED. On average, every user completed the usability evaluation in 10 minutes that comprised a pretest questionnaire, evaluation, and posttest questionnaire. Figure 9 shows an example of device initialization that can be divided into the following phases:

(i) User initiates blink application and enters information required to connect wireless device to a WiFi network (SSID and password of an access point as shown in Figure 9(a)).

(ii) User places wireless sensing device on the area of the screen responsible for VLC transmission and hits Enter to initiate the communication (Figure 9(b)).

(iii) After the communication over VLC completes (during the transmission the user is presented with a progress bar), the wireless sensing device initiates WiFi connection with an AP, and upon successful WiFi connection starts sending data to a cloud system. The sending process is signalized with a green blinking LED (Figure 9(c)), while the screen is updated in parallel with real-time data from the cloud, as shown on a screen (Figure 9(d)). In particular, the wireless sensing device sends temperature and humidity obtained from DHT11 sensor.

6.3. Overall Initialization Time. Figure 10 shows overall initialization time each user (out of 32) took part in our study. This time period included opening web application and entering the relevant data to be transmitted over the VLC (such as AP's SSID and password), VLC from the screen to the wireless device, time required to connect wireless device to an AP, and finally connection to a cloud device. The average initialization time for 32 users was 39.91s (std = 9.86s).

6.4. Error Prevention and Detection. Overall, in 6 cases users unsuccessfully completed the initialization procedure. Out of 6 users, 4 of them lifted the sensing device during the VLC transmission from the screen despite the presence of progress bar; users were actually interested in the VLC from the screen to the device. Since the device did not receive all relevant information (SSID and password), it could not connect to

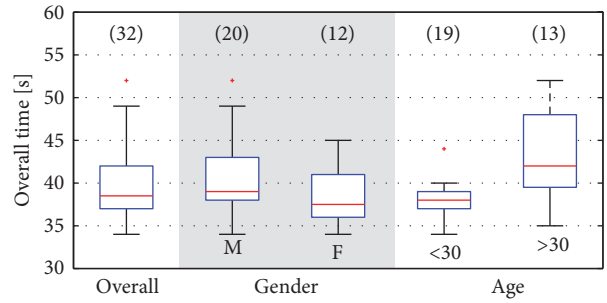

Figure 10: Average protocol execution time.

the AP, which in the end resulted in a red blinking LED. Besides that, 2 users have incorrectly entered SSID and/or password in web application. It is important to note that all users successfully initialized the device after repeating the initialization procedure.

6.5. Usability. After the tests every user had to fill in a posttest questionnaire that contained questions about user satisfaction of the proposed solution along with SUS test [69]. Average SUS score was 88.75 (std = 9.86s), while the average user satisfaction level was 4.81. These results indicate that the proposed solution is easy to use for the end user.

6.6. Impact of Gender. The results of evaluation did not show any significant effect of task completion time, error recognition, user's ease-of-use perception, and the overall satisfaction of the proposed solution. Average time required to complete the initialization was 40.8s (std = 4.94s) for male and 38.42s (std = 3.65s) for female participants which indicates that there is no significant difference in time completion between two genders (p = 0.08). Paired t-test showed that the SUS score did not show significant results between two genders (p = 0.14). Average SUS score for male participants was 90.25 (std = 10.44s), while the average SUS score for female users was 86.25 (std = 8.63s). The average user satisfaction level was 4.94 for male and 4.83 for female users which indicates that there is no significant difference in time completion between the two groups (p = 0.41).

6.7. Impact of Age. Similarly as in previous section, the results of evaluation did not show any significant effect of task completion, error recognition, user's ease-of-use perception, and the overall satisfaction of the proposed solution between

two age groups. The average time required to complete the initialization was 43.08s ($std = 5.30$s) for users over 30 and 37.67s ($std = 2.35$s) for other group of users (under 30). However, we have noticed that older participants require more time to enter information into web application (SSID and password) and place the device on screen ($p = 2.46 \times 10^{-3}$). The average SUS score for older and younger participants was 86.94 ($std = 10.27$s) and 92.31 ($std = 8.91$s), respectively. Paired t-test showed that SUS results do not significantly differ between two age groups ($p = 0.07$). Average user satisfaction level was 4.78 and 4.92 for younger and older group of users, which indicates that there is no significant effect of age on user satisfaction level ($p = 0.15$).

6.8. Summary. The results of usability evaluation indicate that users achieve fast completion time and low error rates as well as high user satisfaction level. Some users suggested using a better LED indication state on wireless sensing device, so the users could have a better knowledge of the state in which the device is in every phase of initialization.

7. Conclusion

In this paper, we explore coding techniques for the visible light communication channel aimed at resource constrained IoT devices, such as wireless sensor devices. The results of estimated VLC channel capacity of approximately 300*bps* indicate the requirement for finding coding schemes with transmission speeds higher than proposed solutions that achieve speeds up to 30*bps*. We introduced BlinkComm, a differential coding scheme that achieves transmission rate up to 100*bps* which is more than a threefold increase in transmission speed compared to the existing solutions.

The proposed coding techniques have minimal hardware requirements such as one LED and one photodiode and operate in real-time, without any complex error detection or correction techniques which make them desirable given that typical IoT devices are constrained in terms of memory and processing power.

We implemented coding schemes on commercially available platform and demonstrated through usability study that our solution has good performance and is easy to use.

In the future we plan to work on finding more efficient coding techniques for transmission over VLC channel that will increase the transmission speed close to the estimated VLC channel capacity.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

References

[1] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: the road ahead," *Computer Networks*, vol. 76, pp. 146–164, 2015.

[2] S. S. M. Elkhodr and H. Cheung, "The internet of things: New interoperability, management and security challenges," *The International Journal of Network Security & Its Applications (IJNSA)*, vol. 8, no. 2, pp. 85–102, 2016.

[3] J. M. De Fuentes, L. González-Manzano, and O. Mirzaei, "Privacy Models in Wireless Sensor Networks: A Survey," *Journal of Sensors*, vol. 2016, Article ID 4082084, 2016.

[4] C.-Y. Chen and H.-C. Chao, "A survey of key distribution in wireless sensor networks," *Security and Communication Networks*, vol. 7, no. 12, pp. 2495–2508, 2014.

[5] Estimote, "Real-world context for your apps," http://estimote.com/, [Online; accessed 25-July-2017].

[6] Stackoverflow, "Pairing iBeacon silently, "http://stackoverflow.com/questions/20077318/pairing-ibeacon-silently, [Online; accessed 25-November-2017].

[7] Amazon, https://aws.amazon.com/iot/button/, [Online; accessed 20-November-2017].

[8] Nest, https://nest.com/uk/smoke-co-alarm/overview/, [Online; accessed 20-9-2017].

[9] Wemo, http://www.belkin.com/us/Products/home-automation/c/wemo-home-automation/, [Online; accessed 25-9-2017].

[10] Fitbit, https://www.fitbit.com/eu/aria2, [Online; accessed 20-11-2017].

[11] M. O. Jewell, E. Costanza, and J. Kittley-Davies Agents, "Connecting the things to the internet: an evaluation of four configuration strategies for wi-fi devices with minimal user interfaces," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 767–778, ACM, 2015.

[12] Libelium, http://www.libelium.com/, [Online; accessed 20-November-2017].

[13] L. Labs, http://www.link-labs.com/, [Online; accessed 20-November-2017].

[14] S. Citizen, https://smartcitizen.me, [Online; accessed 6-December-2017].

[15] N. Kang, "A First step towards security for Internet of small things," *International Journal of Security and Its Applications*, vol. 10, no. 6, pp. 13–22, 2016.

[16] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[17] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and communications security*, pp. 41–47, ACM, 2002.

[18] D. Liu, P. Ning, and L. I. Rongfang, "Establishing pairwise keys in distributed sensor networks," *ACM Transactions on Information and System Security (TISSEC)*, vol. 8, no. 1, pp. 41–77, 2005.

[19] E. Ronen, A. Shamir, A. Weingarten, and C. OFlynn, "IoT Goes Nuclear: Creating a ZigBee Chain Reaction," in *Proceedings of the Symposium on Security and Privacy (SP)*, pp. 195–212, 2017.

[20] arstechnica, "Crypto weakness in smart LED lightbulbs exposes Wi-Fi passwords," http://arstechnica.com/security/2014/07/, 2014, [Online; accessed 25-July-2017].

[21] L. Franceschi-Bicchierai, "Afraid of the Dark? Too Bad, Your Smart Bulbs Can Be Hacked," https://motherboard.vice.com/en_us/article/d7yxxw/hackers-could-take-control-of-your-smart-light-bulbs-and-cause-a-blackout, 2016, [Online; accessed 2-September-2017].

[22] P. Morgner, S. Mattejat, Z. Benenson, C. Müller, and F. Armknecht, "Insecure to the touch: Attacking ZigBee 3.0 via

touchlink commissioning," in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks ser. WiSec '17*, ACM, 2017.

[23] C. Kuo, M. Luk, R. Negi, and A. Perrig, "Message-in-a-bottle: user-friendly and secure key deployment for sensor nodes," in *Proceedings of the International Conference on Embedded Networked Sensor Systems*, ACM, 2007.

[24] Y. W. Law, G. Moniava, Z. Gong, P. Hartel, and M. Palaniswami, "Kalwen: A new practical and interoperable key management scheme for body sensor networks," *Security and Communication Networks*, vol. 4, no. 11, pp. 1309–1329, 2011.

[25] F. L. Wong and F. Stajano, "Multichannel security protocols," *IEEE Pervasive Computing*, vol. 6, no. 4, pp. 31–39, 2007.

[26] H. Pan, Y. Chen, G. Xue, and X. Ji, "Magnecomm: Magnetometer-based near-field communication," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking, ser. MobiCom '17*, pp. 167–179, ACM, NY, USA, 2017.

[27] T. Kovacevic, T. Perkovic, and M. Cagalj, "Lira: A new key deployment scheme for wireless body area networks," in *Proceedings of the Telecommunications and Computer Networks (SoftCOM), 2013 21st International Conference on*, IEEE, 2013.

[28] T. Kovačević, T. Perković, and M. Čagalj, "Flashing displays: user-friendly solution for bootstrapping secure associations between multiple constrained wireless devices," *Security and Communication Networks*, vol. 9, no. 10, pp. 1050–1071, 2015.

[29] Electricimp, http://electricimp.com/, [Online; accessed 25-November-2017].

[30] S. Zhu, S. Setia, and S. Jajodia, "Leap: Efficient security mechanisms for large-scale distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security ser. CCS*, pp. 62–72, 2003.

[31] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: a link layer security architecture for wireless sensor networks," in *Proceedings of the Second International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 162–175, November 2004.

[32] R. Anderson, . Haowen Chan, and A. Perrig, "Key infection: smart trust for smart dust," in *Proceedings of the Network Protocols, ICNP Proceedings of the 12th IEEE International Conference on*, pp. 206–215, IEEE, 2004.

[33] K. Paek, J. Kim, C. Hwang, and U. Song, "An energy-efficient key management protocol for large-scale wireless sensor networks," in *Proceedings of the Multimedia and Ubiquitous Engineering, MUE'07. International Conference on*, pp. 201–206, IEEE, 2007.

[34] C. Castelluccia and P. Mutaf, *Shake them up!*, Usenix Mobisys, 2005.

[35] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.-W. Gellersen, "Smart-its friends: A technique for users to easily establish connections between smart artefacts," in *Proceedings of the Ubicomp: Ubiquitous Computing*, Springer, 2001.

[36] J. Lester, B. Hannaford, and G. Borriello, "Are you with me?-using accelerometers to determine if two devices are carried by the same person," in *Proceedings of the Pervasive Computing*, Springer, 2004.

[37] K. B. Rasmussen and S. Capkun, "Implications of radio fingerprinting on the security of sensor networks," in *Proceedings of the Security and Privacy in Communications Networks and the Workshops (SecureComm)*, 2007.

[38] M. Li, S. Yu, W. Lou, and K. Ren, "Group device pairing based secure sensor association and key management for body area networks," in *Proceedings of the INFOCOM, 2010 Proceedings IEEE*, pp. 1–9, IEEE, 2010.

[39] S. Cherukuri, K. K. Venkatasubramanian, and S. K. S. Gupta, "Biosec: a biometric based approach for securing communication in wireless networks of biosensors implanted in the human body," in *Proceedings of the International Conference on Parallel Processing Workshops*, IEEE, 2003.

[40] F. M. L. J. Y. Li and Y. Zhang, "Aes based biometrics security solution for body area sensor networks," 2009.

[41] M. Mana, M. Feham, and B. A. Bensaber, "Sekes (secure and efficient key exchange scheme for wireless body area network)," *IJCSNS International Journal of Computer Science and Network Security*, vol. 9, no. 11, 2009.

[42] L. Shi, J. Yuan, S. Yu, and M. Li, "Ask-ban: Authenticated secret key extraction utilizing channel characteristics for body area networks," in *Proceedings of the sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pp. 155–166, ACM, 2013.

[43] M. Fomichev, F. Alvarez, D. Steinmetzer, P. Gardner-Stephen, and M. Hollick, "Survey and Systematization of Secure Device Pairing," *IEEE Communications Surveys & Tutorials*, 2017.

[44] F.-L. Wong and F. Stajano, "Multi-channel protocols for group key agreement in arbitrary topologies," in *Proceedings of the Pervasive Computing and Communications Workshops, 2006, PerCom Workshops Fourth Annual IEEE International Conference on*, IEEE, 2006.

[45] R. Mayrhofer, J. Fuss, and I. Ion, "UACAP: A unified auxiliary channel authentication protocol," *IEEE Transactions on Mobile Computing*, vol. 12, no. 4, pp. 710–721, 2013.

[46] C. Soriente, G. Tsudik, and E. Uzun, "Hapadep: human-assisted pure audio device pairing," in *Proceedings of the International Conference on Information Security*, vol. 5222, Springer, 2009.

[47] T. Perković, M. Čagalj, T. Mastelić, N. Saxena, and D. Begušić, "Secure initialization of multiple constrained wireless devices for an unaided user," *Mobile Computing, IEEE Transactions on*, vol. 11, no. 2, pp. 337–351, 2012.

[48] M. Gauger, O. Saukh, and P. J. Marrón, "Enlighten me! Secure key assignment in wireless sensor networks," in *Proceedings of the IEEE 6th International Conference on Mobile Adhoc and Sensor Systems*, 2009.

[49] B. Zhang, K. Ren, G. Xing, X. Fu, and C. Wang, "SBVLC: Secure barcode-based visible light communication for smartphones," *IEEE Transactions on Mobile Computing*, vol. 15, no. 2, pp. 432–446, 2016.

[50] W. Du, J. C. Liando, and M. Li, "Soft Hint Enabled Adaptive Visible Light Communication over Screen-Camera Links," *IEEE Transactions on Mobile Computing*, vol. 16, no. 2, pp. 527–537, 2017.

[51] N. Saxena, J.-E. Ekberg, K. Kostiainen, and N. Asokan, "Secure device pairing based on a visual channel: Design and usability study," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 1, pp. 28–38, 2011.

[52] T. Kovacevic, *Scalable and Secure Methods for Initialization of Wireless Sensor Networks [Doctoral Thesis]*, University of Split, 2016.

[53] T. Komine and M. Nakagawa, "Fundamental analysis for visible-light communication system using LED lights," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 1, pp. 100–107, 2004.

[54] K. Carlon, "Smartphone screens explained: display types, resolutions and more," https://www.androidpit.com/smartphone-displays-explained, 2016, [Online; accessed 10-November-2017].

[55] D. Nield, "Gadget tech explained: AMOLED vs. IPS displays," http://www.gizmag.com/amoled-vs-ips-display-technology/39196/, 2015, [Online; accessed 10-November-2017].

[56] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, California Technical Pub., 1997.

[57] P. H. Pathak, X. Feng, P. Hu, and P. Mohapatra, "Visible light communication, networking, and sensing: a survey, potential and challenges," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2047–2077, 2015.

[58] P. Lewis, "Rendering performance," https://developers.google.com/web/fundamentals/performance/rendering/, 2016, [Online; accessed 24-November-2017].

[59] J. Jongerius, "Rendering performance," http://www.vsynctester.com/manual.html, 2015, [Online; accessed 19-November-2017].

[60] D. Wilson, "Triple Buffering: Why We Love It," http://www.anandtech.com/show/2794/2, 2016, [Online; accessed 15-November-2017].

[61] K. Ghazi, "The Gamer's Graphics and Display Settings Guide," http://www.tweakguides.com/Graphics9.html, 2016, [Online; accessed 15-November-2017].

[62] B. Photodiode, http://www.vishay.com/docs/81521/bpw34.pdf, [Online; accessed 10-3-2013].

[63] Arduino, https://www.arduino.cc, [Online; accessed 7-3-2012].

[64] Espressif, http://espressif.com/en/products/esp8266/, [Online; accessed 10-November-2017].

[65] W. Y. P. l. Dietz and D. Leigh, http://math.hws.edu/vaughn/cpsc/336/docs/led-sensor.pdf, [Online; accessed 10-3-2017].

[66] F. M. M. III, *Siliconnections: Coming of Age in the Electronic Era*, McGraw-Hill, NY, USA, 1986.

[67] *LED Circuits And Projects*, Howard W. Sams and Co., Inc., NY, USA, 1973.

[68] "Digital 16bit serial output type ambient light sensor ic," http://www.rohm.com/web/global/datasheet/BH1730FVC/bh1730fvc-e, [Online; accessed 10-1-2017].

[69] J. Brooke, "SUS: A Quick and Dirty Usability Scale," in *Usability Evaluation in Industry*, vol. 189, Taylor and Francis, 1996.