A reliable asynchronous protocol for VLC communications based on the rolling shutter effect

Júlia Ferrandiz-Lahuerta*, Daniel Camps-Mur*, and Josep Paradells-Aspas†
*i2CAT Foundation †UPC-Barcelona Tech
julia.ferrandiz.lahuerta@gmail.com, daniel.camps@i2cat.net
josep.paradells@entel.upc.edu

***Abstract*—In this paper we introduce a novel reliable asynchronous protocol that allows to establish a VLC link between a LED luminary and an off-the-shelf smartphone. Our protocol and decoding algorithms benefit from the access to advanced camera settings available in the latest generations of mobile devices to outperform previous VLC designs in the state of the art. In particular, in the paper we provide an experimental evaluation using a commercial Nexus 5 device where it is shown how our designed prototype can achieve transmission speeds of up to 700 bps and operating distances of up to 3 meters. The previous performance figures have the potential to spark a wide set of novel applications.**

I. Introduction

LED luminaries are becoming pervasive in the market due to their advantages in energy efficiency. However, in addition to being energy efficient, LEDs can also sustain very high switching frequencies that is the key to transform traditional lighting gear into fully capable communication devices. Indeed, inexpensive drivers connected to commercial LED luminaries are the only requirement to spark a wide set of indoor applications based on Visible Light Communications (VLC) [1].

Regarding VLC, a large set of work in the state of the art has been devoted to increasing the achievable data rates. A prominent example of these efforts is [2] where the authors propose a OFDM based modulation for VLC. As of today though, the drawback of these solutions is that they require specific hardware interfaces that are not available in devices like smartphones. Thus, an alternative body of work has recently emerged with the goal of enabling VLC communications with smartphones. The work presented in this paper contributes to this body of work.

VLC communication with unmodified smartphones is largely based on the rolling shutter effect of the CMOS camera sensor embedded in these devices. The rolling shutter effect manifests when a CMOS camera sensor, composed by a matrix of photo-detectors, does not capture light with all of its sensors simultaneously but rather does it by progressively exposing groups of photo-detectors in one of the spatial dimensions. The result of such effect is that a light blinking at a frequency high enough not to be perceived by the human eye manifests as a set of dark and bright stripes through such CMOS camera sensor. An illustration of this effect is depicted in Figure 1 and the interested reader is referred to [3] for more details on the rolling shutter effect.

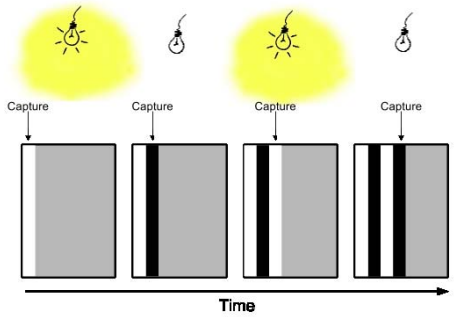


Fig. 1. Rolling shutter effect: Blinking light detected as dark and bright stripes.

The main contribution of this paper is two-fold. First, we introduce a novel reliable asynchronous protocol based on on-off keying (OOK) modulation that allows an LED luminary to communicate with a rolling shutter device achieving effective data rates up to 700 bps. Second, we present a performance evaluation of our designed protocol using a Nexus 5 smartphone, and a commercial LED driven by a Raspberry Pi. The obtained results demonstrate that our implementation can sustain data rates of hundreds of bits per second at distances of up to 3 meters, which are enough for the foreseen applications.

This paper is organized as follows. Section II provides an overview of relevant related work. Section III describes the design of the LED transmitter and the smartphone receiver application. Section IV contains an experimental evaluation of our VLC prototype. Finally, section V summarizes and concludes the paper.

II. Related Work

The authors in [6] present a system to convey visual codes to a rolling shutter device using commercial video displays. The proposed method consists of inserting color patterns in the video signal that appear as a single color to the human eye, but that can be detected by a rolling shutter camera sensor. Thus, information can be transmitted by associating a set of bits to the different color patterns. Our work differs from [6], in the sense that we use a blinking light instead of a video display to encode information, which delivers higher data rates. In [7] the authors present a method for indoor positioning using VLC that achieves centimeter level accuracy. Their proposed method consists of a camera sensor capturing

978-1-4799-5952-5/15/$31.00 ©2015 IEEE

multiple LED luminaries positioned on the ceiling, which use a unique blinking frequency as device identifier. Thus, by querying a central system, the mobile device translates the LED identifiers into location coordinates that can be used to derive its own position via triangulation. Unlike [7], the paper at hand focuses on the VLC link itself rather than on its application for indoor positioning. Another approach for VLC communication applied to indoor positioning is presented in [5] where the authors introduce a Frequency Shift Keying (FSK) modulation applied to blinking LED luminaries. The proposed approach achieves a transmission data rate of 10 bps, which suffices to encode luminary identifiers for indoor location but falls short as a generic VLC communications link. It is worth noticing though that this kind of applications are already finding a place in the market [8]. Finally, the work in [4] is the most closely related to the method presented in this paper. In [4] the authors introduce an asynchronous protocol using on-off keying (OOK) between an LED and a mobile device while reporting the following performance figures. Setting up the LED light at 35 cm from an illuminated surface, and the mobile device at 9 cm from that same surface the authors manage to decode a 16 ASCII character long message in 2.5 seconds, which translates to a data rate around 51 bps. The method presented in the paper at hand, by means of using more advanced camera settings on the mobile device, vastly exceeds the performance figures reported in [4].

III. System Design

In this section we describe the transmitter and receiver designs of our VLC system.

A. Transmitter

In order to simplify our LED controller design we decide to use a commercial Raspberry Pi (RPi) device that includes a variety of digital interfaces. In particular, the embedded UART controller can be used for asynchronous communications with a configurable data rate between 110 bps and 115.2 Kbps. A UART enables the asynchronous transmission of ASCII characters by translating each character into a string of 8 bits, whilst adding for each character a start bit (0) and stop bit (1). Notice that such a serial interface is not optimal regarding achievable data rates but is very convenient for implementation purposes. Thus, we build our asynchronous protocol on top of this serial communication channel.

1) Selection of UART data rate: An important protocol design decision is the bit rate at which the UART controller will operate to drive the LED luminary. Notice that increasing the data rate, decreases the bit time, and consequently the width of the dark and bright stripes detected by the CMOS camera sensor on the mobile device. Thus, in our system the upper bound on the maximum data rate is set by the ability of the mobile device to discern these bright and dark stripes. This ability in turn depends on the camera resolution, i.e. the more pixels in a bit time the higher the reliability of the detection algorithm, on the time each pixel is exposed to light, which can be controlled through the exposure time, and on the camera gain (ISO). These settings could not be modified in older versions of the Android or iOS APIs, which limited the performance achieved by previous works like [4]. However, the Android 5 API provides fine control on camera settings [9], which we can use to enhance the VLC receiver. Hence, in order to optimize transmission speed we follow an empirical approach whereby we increase the data rate observing for each case the number of pixels occupied by the transmitted bits. In our experiments we always set the exposure time below the UART bit duration, in order to minimize the cases where a photo-detector captures light from more than one bit, and consequently increase the camera gain to compensate for the short exposure time. Table I illustrates the measured relations.

TABLE I
Measured relation between UART speed and pixels per bit

UART speed (bps)	**Exposure time**	**ISO**	**Pixels/bit 2562x1944**
1200	1/6000	10000	50
2400	1/6000	10000	25
4800	1/6000	10000	12
9600	1/10000	10000	6

Table I reports average bit widths, where we can observe how as expected the camera resolution affects the number of pixels that fall within a single bit time. Thus, the more pixels in a bit the more reliable the decoding algorithm will be, but the shorter the bit time the higher the data rates that can be achieved by the VLC link. In addition, selecting a very small exposure time results in fewer light received by the camera and may translate in decreased ranges. Consequently, looking at Table I we decide to set the bit rate in the UART interface to 4800 bps.

2) Error detection and correction: Notice that the human visual system is not able to perceive an LED luminary blinking at the selected UART speed. However, when encoding a generic set of input bits through a UART interface, these are packed into 10 bit characters that differ in average brightness, thus resulting in visible flickering in our experiments. Therefore, a code that results in an even number of 1s and 0s is used, which halves the effective data rate but removes any visible flickering.

Given that we are constructing a broadcast channel without feedback between the LED luminary and the mobile device, and given the inherent variability of the light channel, we decide to apply another layer of error detection using CRC codes. In particular, the input bit stream is partitioned into 24 equally sized groups and an individual CRC4 is computed for each group. The set of 24 CRCs is appended at the end of the bit stream to be transmitted. Notice that transmitting individual CRCs allows the receiver to isolate the group of transmission blocks[1] in error, thus easing the process of error recovery, which is performed in our protocol by simply having the transmitter periodically repeat the same set of input bits.

3) Protocol structure: Our protocol is built around the concept of transmission blocks, where a transmission block

[1]The concept of block is introduced in the next section.

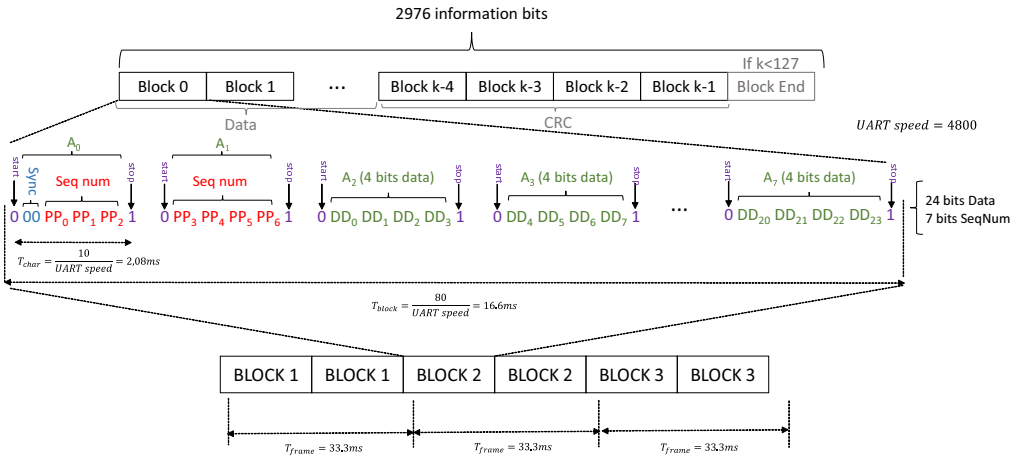


Fig. 2. Protocol Structure

contains a certain number of UART characters. Thus, the set of input bits is coded for error detection, and progressively mapped into transmission blocks. Two concepts are important for the design of the transmission blocks, namely the preamble required to detect the beginning of a block, and the design of the sequence number space that will determine the maximum number of transmission blocks.

Regarding the design of the preamble, we notice that given the used encoding to avoid flickering the transmitted sequence will never contain more than two consecutive bits with the same value. Thus, the preamble used to synchronize the beginning of a block is designed by simply transmitting two zeros after the initial start bit (0) in the first UART character of the block. Notice that using three bits for the preamble, not considering the final stop bit, and taking into account encoding, leaves three bits in the first UART character to encode the block sequence number, which is small for practical purposes. Hence, in order to increase the sequence number space we decide to use the first two UART characters in a block to encode the sequence number, thus allowing to transmit bit sequences of up to $2^7 = 128$ blocks. In addition, the basic frame structure of 128 blocks could be expanded in a hierarchical way to allow for even longer input bit streams, however we leave a detailed design of such scheme as future work. Figure 2 depicts the described block structure, while highlighting the first two UART characters that contain the preamble and the block sequence number.

In order to complete the protocol design, we need to decide the number of UART characters that will be transmitted in each block. Consider N to be the number of blocks that fit in a single frame of duration T_{frame} captured by the shutting roller device, and let R_{uart} be the transmission rate of the UART controller. Thus, the number of bits in each of the blocks contained in the captured frame can be computed as $B = \frac{T_{frame} \times R_{uart}}{N}$. Notice though, that the first two UART characters do not contain data bits, all UART characters contain a start and a stop bit, and encoding is used. Hence the number of effective data bits in a block is $B_{eff} = \frac{2}{5}(B - 20)$. Now, consider the lower part of Figure 2 where it can be seen how the receiver frame capture window is in general not aligned with the beginning of a block transmission by the LED luminary, thus in general one of the N transmitted blocks will fall between two capture windows and will be lost[2]. Therefore, considering a typical capture rate of 30fps in the receiver, the effective transmitted data rate equals $R = 12(N-1)(B-20)$. Consequently, considering $R_{uart} = 4800$ bps and $T_{frame} = 32.5$[3] ms the optimum number of blocks per frame that maximizes the transmitted data rate is $N = 2.81$. In practice though, either $N = 2$ or $N = 3$ need to be chosen resulting respectively in data rates of $R_{N=2} = 696$ bps and $R_{N=3} = 768$ bps.

Finally, regardless of the chosen N a repetition code at the block level should be used. Otherwise, blocks that fall between captured frames in the receiver will only be recovered after the sequence repeats, which for long input sequences could potentially be a very long time. Given the close maximum data rates for $N = 2$ and $N = 3$, we opt for a simpler design with $N = 2$ together with a simple repetition code illustrated in the lower part of Figure 2, which ensures that each block can be captured regardless of the misalignment between transmitter and receiver. Considering $N = 2$ results in 8 UART characters for each block and a block duration of 16.6 ms as illustrated in Figure 2. Consequently, the designed protocol allows for an input sequence of 2976 bits (see Figure 2), which suffices for most typical applications.

B. Receiver

1) Frame processing: The first decision in the receiver design is how to process the received frames. In this regard, a mobile device could record a transmission for post-processing or process the received frames in real time. We experimented

[2]We do not consider the receiver doing frame stitching as in [5], since this is an expensive process, and in general mobile devices may experience random delays between captured frames.

[3]This is the measured capture time, which is slightly below the nominal 33 ms since some processing is required between frames.

with both designs but determined that the former was too slow to be used in practice. Thus, a real-time frame decoder is designed that accesses the raw data from the Android camera API, and spawns a different thread for each received frame where the actual frame decoding is performed. We validated empirically that the designed scheme is able to keep up with the frame rate of the camera. In particular, Figure 3 depicts the histogram of the activation time between consecutive threads, which we can observe to be very close to 33 ms that is the nominal camera frame rate.

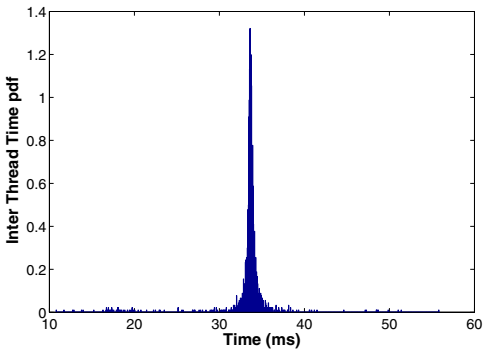


Fig. 3. Inter frame capture time histogram

2) Decoding algorithm: Each frame is received in YUV format, where the Y component corresponds to luminance and the U and V components are the color components. Our decoding algorithm though, only needs to decide between dark and bright stripes, corresponding respectively to logical 1s and logical 0s, thus only the Y component is extracted, which is observed to vary between 0 and 255. Regarding the decoding algorithm, it is to be noted that the receiver process in the mobile device may at any moment be preempted by other processes of higher priority. Thus, the main criteria in the design of the decoder is to minimize computational complexity and to reduce decoding time.

In particular, an algorithm based on a fixed inter-bit threshold T_Y is used, which works in the following way. Upon reception of a new frame, the average luminance of each line is computed for all pixels in the image. After that the algorithm starts from the left-most value comparing successive luminance values until a value difference exceeding T_Y is found. If an increase in luminance is observed the algorithm decides that at that pixel a logical 1 begins, otherwise it decides that a logical 0 begins. Once the first pixel of the first bit has been detected, the algorithm continues counting the number of pixels until a luminance variation above T_Y is observed, where a bit transition is detected. Thus, upon detecting a bit transition the algorithm compares the number of counted pixels to the known number of pixels per bit (see Table I) to decide on the number of consecutive bits with the same value. This process continues until all bits in the frame have been decoded. In our implementation, and considering that the luminance component varies between 0 and 255, T_Y is empirically set to 10.

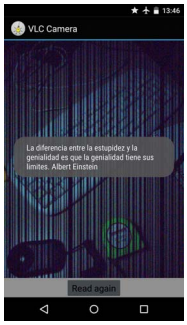


Fig. 4. VLC Android application. Notice the camera capturing a scene illuminated with the modulated LED.

Finally, while decoding the bits in the captured frame, a sliding window is used to detect the preamble that defines the start of a block, and sends the received block to a collector thread that implements block reordering and CRC checking. Figure 4 contains a snapshot of the Android application used to decode the VLC data. In the figure the bright and dark stripes correspond to the pattern created by the transmitted data sequence, where the block preambles can be easily distinguished as wider dark stripes.

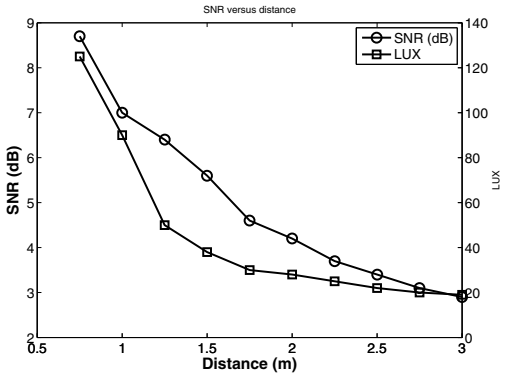
Notice that while minimizing complexity, this algorithm may struggle when the received SNR is very low or when there are opaque objects obstructing parts of the captured frame. We leave as future work the design of more advanced decoding algorithms.

IV. Performance Evaluation

In this section we introduce an experimental performance evaluation that has been carried out to validate our VLC prototype. The evaluation is performed using a Nexus 5 smartphone, and a 20 Watt commercial LED luminary controlled by the UART in the RPi device. Experiments are performed with the LED luminary placed at a distance d from one of the walls in our lab and the mobile device having its camera also pointing to the same wall and at the same distance d. In our experiments, we increase d between 0.75 meter and 3 meters, and vary the size of the input bit stream, while measuring the receiver throughput as main performance indicator. Notice that this is a challenging scenario because the mobile device is not directly exposed to the LED. We have validated that the conclusions derived from this set up also hold when the mobile device is directly exposed to the LED. Figure 5(a) illustrates our experimental set up.

As a first step in our evaluation we measure the received SNR and light intensity (lux level) for different distances, where SNR at distance d is defined as the measured ratio between the luminance value received by the camera when the LED is on and off. Figure 5(b) depicts the measured SNR and lux levels at different locations with our 20 Watt LED. Notice that the measured lux levels are small compared with recommended indoor levels [10], meaning that in practice we would expect to operate in the high SNR area.

(a) Experiment set up. Nexus 5 and LED held at distance $0.75 \leq d \leq 3$ meters from the wall.



(b) Measured SNR and light intensity versus distance with a commercial 20 Watt LED.

Fig. 5. Experimental setup.

Figures 6(a) depicts the effective throughput obtained when increasing distance d, where effective throughput is defined as the ratio between the transmitted message size, which is set to 960 bits, and the time required to obtain and reorder all the transmission blocks of the message. In order to gain statistical confidence each experiment is repeated 40 times, and the results are depicted using boxplots. In the boxplots the red line represents the median of the distribution, the edges represent the 25th and 75th percentiles with values q_{25} and q_{75}, the upper and lower whiskers extend respectively to $q_{75}+1.5\times(q_{75}-q_{25})$ and $q_{25}-1.5\times(q_{75}-q_{25})$, and the red crosses are considered to be outliers.

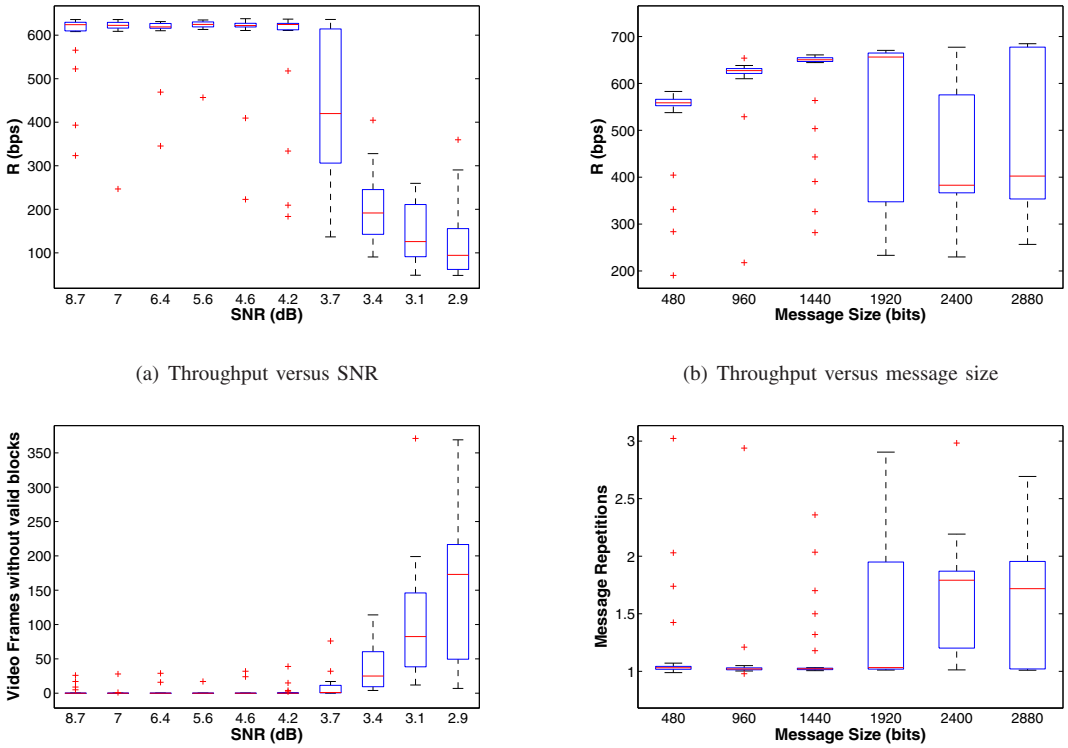
We can see in 6(a) how when $d < 2$ meters, i.e. $SNR > 4.2$ dB, the effective data rate stays close to the theoretical maximum of $R_{N=2} = 696$ bps, but when $d > 2$ meters the data rate suddenly drops to 100 bps. In addition, when $d > 2$ meters the data rate is fairly constant across experiments, with the exception of some outliers (red crosses), but when $d > 2$ meters the variability of the measured throughput across experiments also experiences a sudden increase. To explain the observed dynamics Figure 6(c) depicts for the same experiments the number of received video frames where no valid data block could be decoded by the Nexus 5. Notice that in the region of consistently high data rates, all video frames contain a valid block, with the exception of a few experiments that result in the outliers (red crosses). The reason for these outliers is the following. The nominal block size in our protocol, depicted in Figure 2, is $T_{block} = 16.6$ ms, which is slightly above $\frac{T_{frame}}{2}$ ms, where $T_{frame} = 32.5$ ms is the measured frame capture time in the Nexus 5 camera. This can result in situations where the two blocks fall exactly in the middle of the video frame captured by the camera, and thus no block can be decoded. This situation though is temporary because T_{frame} and T_{block} not being exact multiples, the position of each protocol block in the video frames shifts with time. In addition, the missing blocks can be recovered once the LED repeats the broadcasted message. Notice thus, that this is a seldom effect but that has a potentially big impact on perceived throughput when it occurs. Finally, when $d > 2$ meters the degraded SNR starts impacting the receiver performance, as frames appear with corrupted blocks that are discarded and only recovered once the message is repeated.

Figure 6(b) analyzes the effective throughput when the transmitted message size increases. In this experiment the Nexus 5 is located at a distance $d < 0.75$ meters from the illuminated wall. We can see in Figure 6(b) how the throughput consistently increases until a message size of 1440 bits, because higher message sizes better amortize the header and CRC overheads of the protocol. However, after 1440 bits the variability of the measured throughput across experiments suddenly increases obtaining values between 200 bps and 700 bps. The reason for this variability is the effect described in the previous paragraph, whereby even in good SNR conditions some video frames may not be able to decode valid blocks. This effect happens with a higher probability as the message size increases, which explains the results in Figure 6(b). Indeed, this is confirmed in Figure 6(d) that shows the number of times that the Nexus 5 needs to receive the full message before being able to recover all the blocks. Notice that this effect could be mitigated by either reducing the block size, which would however decrease the maximum achievable throughput, or by stitching together video frames in the receiver, which would however increase CPU and power consumption of the decoder. As part of our future work we will investigate effective techniques to mitigate this effect.

Finally, it is worth mentioning that an average power consumption of 1500 mW was measured on the Nexus 5 device while running our VLC application. This value is consistent with the recording power consumption of current mobile devices [11].

V. Conclusions

In this paper we have introduced a novel VLC asynchronous protocol that allows a commercial LED luminary to broadcast information to commercial smartphones. The proposed protocol is built upon a UART interface controlling the LED,



(a) Throughput versus SNR

(b) Throughput versus message size

(c) Number of video frames without valid blocks versus SNR

(d) Number of complete message repetitions versus message size

Fig. 6. Results from experimental evaluation.

and can thus be easily implemented in a variety of platforms. On the smartphone side, the proposed protocol requires a CMOS camera sensor subject to the rolling shutter effect. An experimental evaluation using a Nexus 5 smartphone is described demonstrating how our designed prototype outperforms previous works in the state of the art by achieving data rates up to 700 bps, and maintaining a functioning link at distances of up to 3 meters.

As potential lines of future work we consider the following. First, improving the achieved data rates by means of configuring the transmitter interface at higher data rates, and enhancing the decoding algorithm in the smartphone. Second, studying the applicability of FEC schemes to recover lost blocks without waiting for retransmissions. Third, studying the applicability of the proposed protocol in scenarios with densely packed LED luminaries, for which medium access control mechanisms between LEDs may be required. We notice that TDMA and CDMA seem plausible candidates.

VI. Acknowledgements

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement 618098, and from the Spanish Ministry of Economy and Competitiveness (MINECO), under research grants TEC2013-47960-C4-4-P and TEC2012-32531.

References

[1] Armstrong, J.; Sekercioglu, Y.; Neild, A., *Visible light positioning: a roadmap for international standardization*, Communications Magazine, IEEE , vol.51, no.12, pp.68,73, December 2013

[2] Elgala, H. ; Mesleh, R. ; Haas, H. , *Indoor broadcasting via white leds and OFDM, IEEE Transactions on Consumer Electronics*, vol. 55, no. 3, pp. 1127, 1134, 2009

[3] Kodak Image Sensor Solutions, *Shutter operations for CCD and CMOS image censors*, Application note: MTD/PS0259, Rev. 3, June 2011.

[4] Danakis, C.; Afgani, M.; Povey, G.; Underwood, I.; Haas, H., *Using a CMOS camera sensor for visible light communication*, Globecom Workshops (GC Wkshps), 2012 IEEE , pp.1244,1248, 3-7 Dec. 2012

[5] Rajagopal, H. Lazik, P.; Rowe. A., 2014 *Visual light landmarks for mobile devices*. In Proceedings of the 13th international symposium on information processing in sensor networks (IPSN 14). IEEE Press, Piscataway, NJ, USA, pp. 249,260.

[6] Woo, G.; Lippman, A.; Raskar. R., 2012. *VRCodes: Unobtrusive and active visual codes for interaction by exploiting rolling shutter*. In Proceedings of the 2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR) (ISMAR 12). IEEE Computer Society, Washington, DC, USA, pp. 59,64.

[7] Kuo, S.; Pannuto, P.; Hsiao, K.J.; Dutta. P.; 2014. *Luxapose: indoor positioning with mobile phones and visible light*. In Proceedings of the 20th annual international conference on Mobile computing and networking (MobiCom 14). ACM, New York, NY, USA, pp. 447,458.

[8] Ganick . A.; Ryan, D.; *Self identifying modulated light source*. Patent No. US20130026940, 01 2013.

[9] https://developer.android.com/about/versions/android-5.0.html#Camera-v2

[10] Richman, E., *Requirements for Lighting Levels*, 2012, available at: https://www.wbdg.org/pdfs/usace_lightinglevels.pdf

[11] Chen, Xiang, et al. *How is energy consumed in smartphone display applications?*. Proceedings of the 14th Workshop on Mobile Computing Systems and Applications. ACM, 2013.