DesignCon 2019

Lowest-cost communication with light from an IoT device to smartphone

Prof. Laszlo Arato, NTB Buchs
laszlo.arato@ntb.ch, +41 81 755 3377

Prof. Rolf Grun, NTB Buchs
rolf.grun@ntb.ch

Prof. Dr. Carlo Bach, NTB Buchs
carlo.bach@ntb.ch

Dr. Alexander Schoech, NTB Buchs
alexander.schoech@ntb.ch

Christoph Capiaghi, NTB Buchs
christoph.capiaghi@ntb.ch

Simon Fink, NTB Buchs
Simon.fink@ntb.ch

Abstract

While fully integrated smart sensors and devices are getting smaller and cheaper, the cost of communicating their data grows more significant. Conventional links like USB, WLAN, Bluetooth, ZigBee, and others are too expensive, when selling price shall drop below $ 5.-.

Implemented for a medication transport temperature-tracking device of a Swiss company, we demonstrate the unidirectional data transfer of 84 Bytes of Data from the LTI device (Long Term Indicator) to almost any smartphone in less than 10 seconds at almost no additional cost using only three LEDs.

Author(s) Biography

Laszlo Arato is Professor for Electronics, Signal Transmission- and Processing at the Interstate University of Applied Sciences of Technology NTB in Buchs [1]. His research interests include smart sensors and their implementation on FPGAs and ASICs. He received his engineering degree at the ETH in Zurich in 1990. He worked at Schmid Telecom AG in Zurich, Conexant Systems Inc. in Red Bank, NJ and Irvine, CA, as well as Qualcomm Inc. in San Diego before switching to academics in 2006 at the FHNW.

Rolf Grun is Professor for Computer Science at the Interstate University of Applied Sciences of Technology NTB in Buchs. He received his engineering degree at the ETH in Zurich in 1994. From 1994 - 1999 he worked for Telekurs AG Zurich, where he cryptographically protected the Swiss ATM and EFT-POS System. From 1999 to 2004 he was lecturer at the University of Applied Sciences HTW in Chur.

Dr. Carlo Bach is Professor for Computer science at the Interstate University of Applied Sciences of Technology NTB, Buchs. After his Ph.D. at the ETH Zurich he worked for eight years as project lead for software before becoming professor at the NTB in 1999. He currently heads the Machine Vision group and works on challenges of automated visual inspection of difficult materials.

Dr. Alexander Schöch received his B.Sc. in computer engineering and his M.Sc. in industrial engineering from the NTB on crypto analysis with FPGAs. He received his Ph.D. degree from the University of Padua for metrology at elevated temperatures. He is a research associate at the Institute for Production Metrology and Optics (PWO) of NTB.

Christoph Capiaghi completed his B.Sc. and M.Sc. at the NTB, and has also worked as hardware designer for IMT Medical in Buchs for several years before joining the NTB again for this project.

Simon Fink received his B.Sc. in 2016 and is currently continuing his M.Sc. studies while working part-time at the NTB at the institute for Engineering Informatics.

1. Introduction

A significant portion of pharmaceuticals are rather sensitive to heat or cold. From fabrication to usage they need to be stored within a product specific temperature band, it must not be kept below a certain temperature or above a certain other temperature for extended period, or it will decay and become not only useless but also potentially harmful. This applies to storage, handling and transportation.

Figure 1: A Multi-Level PDF Indicator shipped with a collection of pharmaceutics.

There are many pharmaceuticals in need of a continuous cool chain, as they would rapidly lose their benefit even at room temperature. This is not always easy to guarantee, and if unchecked transport companies would likely cut corners and turn off expensive cooling – which is generally difficult to prove, and even more so in poorer, hot and large countries.

The industry partner for our research is for many years a supplier of LTI (Long Term Indicators) to monitor temperature-sensitive medication shipments. Their typical product is the size of an old cell phone and connects with USB to any PC workstation for data-readout at the end of the shipment.

Figure 2: LTI Long-Term Indicators of our industry partner [2]

«Libero C» LTI devices cost about $ 30.- and are thus only viable to track large crates.

To track individual boxes, the company decided to invest into a new device has the size of a 3mm thick postal stamp, costs less than $ 5.- and works up to 5 years.

Figure 3: The new, much smaller LTI is the size of a postal stamp [3]

Pressing the single button, the user sees on a green, yellow or red LED, indicating the temperature stability of the medication. That "go/no-go" information is good enough for most cases. However, especially if the drugs are no longer safe to use, one would like to know what went wrong – when and where was the box exposed to degradation … and for how long.

This is when a data download is needed, somewhere in the world at a doctor's office.

Big challenge for the new device was this data communication with the end clients, where any current technology like USB, Bluetooth, WLAN or ZigBee is far too expensive to include and too power-hungry for the integrated battery.

2. Hardware Facts

The product was to consist of a single mixed-signal ASIC chip, a button, 3 LEDs, a break-off latch to trigger the start of recording when removed together with the self-adhesive tape cover when glued to the box of pharmaceuticals. No additional elements were allowed to reduce costs.

The size and life expectation dictated the battery to be a 3.0V lithium-ion button cell [4] battery with a maximum current of 3.0 mA. Everything needed to fit into this voltage and current budget. This is barely sufficient to power one LED, but not three.

The receiver must be a regular smartphone without any hardware modifications. We used the Samsung S7 as reference for Android systems, and iPhone 8 for Apple, with the intention to migrate the code and receiver support later for a wide range of cell phones.

3. LED Signaling

3.1 Brightness Signaling

One way to signal different values with an LED is to use pulse-width modulation (PWM) to control the LED brightness, and use this to indicate different bit values. A PWM frequency of 250kHz proved to be slow enough to turn the LEDs completely on and off during each cycle, but was fast enough to ensure that the emitted light will be integrated across each pixel during regular movie recording settings on the smartphone camera.
As a 10% duty cycle has the LED just being active during 10% of the time compared to a 100% activity, the emitted amount of light is also effectively 10 times larger during 100% activity than during 10% activity.

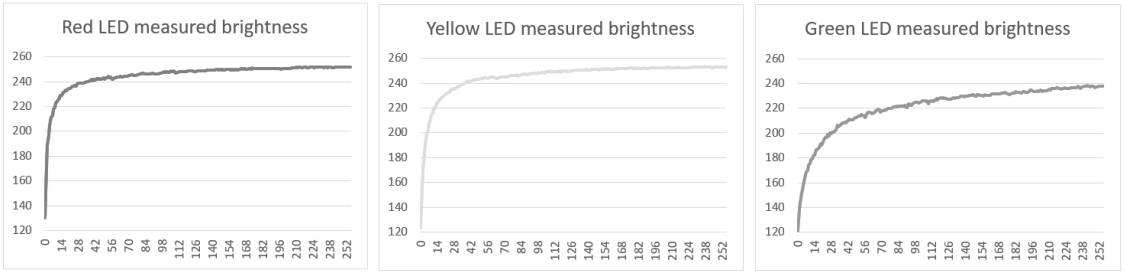

Figure 4: Recognized brightness levels for different LED colors

It was somewhat surprising to find, that this is not reflected in the camera observed brightness values! The three diagrams on the right show the LED activation in steps of 256 PWM steps from 0 = never activated to 255 = 100% activated.
The vertical starting value around 120 is due to additional background light, while all values are computed across all significant pixels for a particular LED.

As we found out during our investigation, this non-linearity is not caused by the smartphone camera sensitivity itself, but due to a saturation in the center of the recorded LED. The graphics on the right shows the enlarged 40 by 40 pixel area of the recorded LED and the per-pixel brightness values.

The diagram on the next page shows a 10% LED illumination with saturated pixel marked bright yellow. For higher PWM values, the number of saturated pixels increases significantly, making the brightness distinctions very difficult. There is simply no additional information, on how much a saturated pixel is saturated. Different approaches to give saturated pixels a non-linear value depending on the number of saturated pixels proved to be futile and lead not to a reliable algorithm. To decrease camera sensitivity to avoid saturated pixels was also unsuccessful, as it was making the camera image too dark for proper aiming while still not resulting in a usable linear reception.

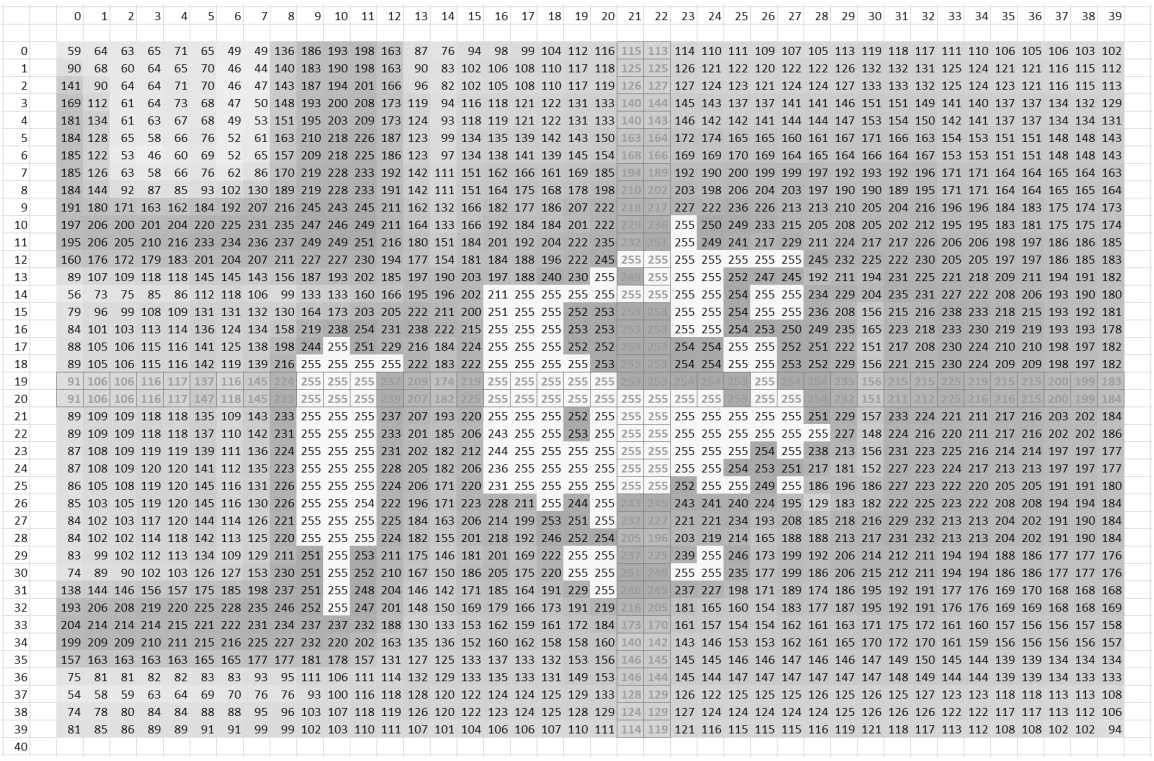

Figure 5: Camera image of an LED with pixel-values and saturation at only 10% PWM modulation

A much more successful approach was to pre-distort the PWM modulation to create a linear line of brightness values at the receiver. Using only 9 steps of brightness, the values of 0 through 8 were exponentially assigned PWM cycles to result in an optimal decoding distance between all levels on the receiver side.
The table and diagram show for the lowest step (0) no LED activation, and full LED activation for the highest step (8).

Step	LED activation
0	0
1	5
2	9
3	19
4	34
5	61
6	100
7	150
8	255

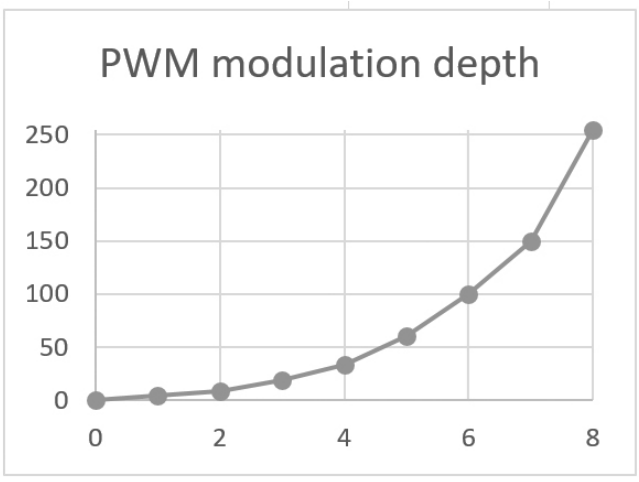

Figure 6: LED PWM cycles for 9 brightness levels

3.2 Rolling Shutter Usage

A completely different approach to detect PWM width on the receiving side is the usage of the rolling shutter effect. CMOS cameras typically read and digitize image information not simultaneously for the entire image, but line-by-line. This is of course primarily an economic aspect of each camera chip, where it only needs 2592 analog-digital converters implemented in parallel with the line-by-line method rather than 5 million for a 5-megapixel imager.
The result, among others, is the distortion of any fast moving object in the picture due to the sampling delay, typically between vertical lines [5].

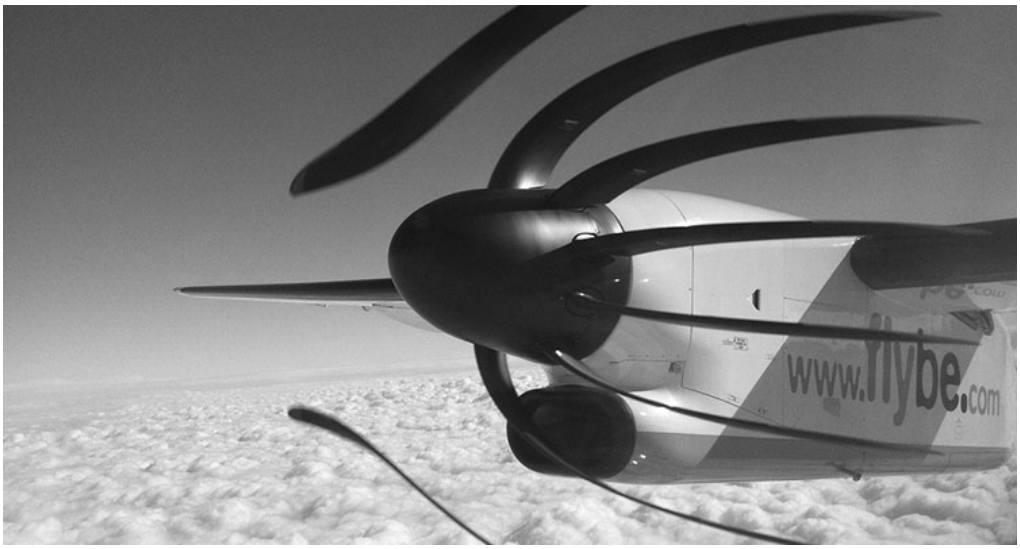

Figure 7: Image of fast-moving propeller blades exposing the rolling shutter effect by Jason Mullins

Configuring the camera to very high sensitivity (ASA) and short exposure times, combined with a very close and intentionally blurred video recording will result in pictures as in Figure 8. Recorded at a distance of about 10mm, the three LEDs form each a rather big and blurred spot with a width around 100 pixel.

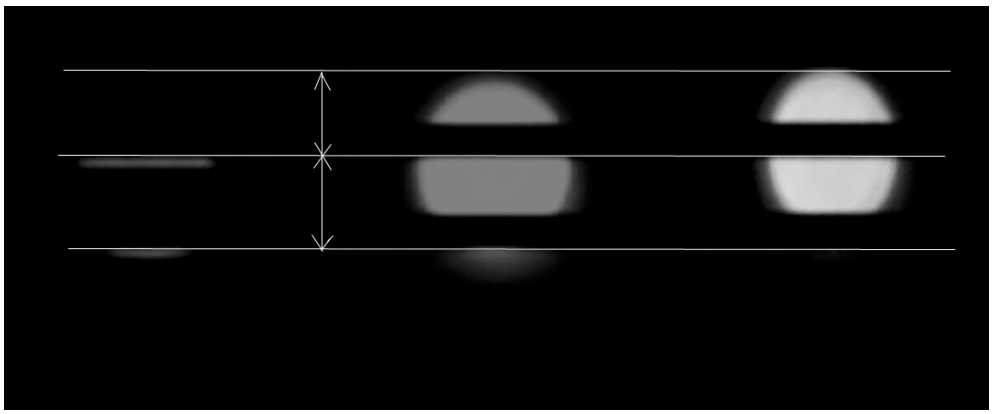

Figure 8: PWM-cycles captured using rolling-shutter effect

With the rolling shutter moving vertically from top to bottom, we can identify at the middle added grey horizontal line the common PWM start for all three LEDs. The distance to the bottom added horizontal line is proportional to the time of the full 255 PWM cycles, after which the PWM modulation restarts. Measuring the distance of the light and dark vertical sections for each LED will give a good level detection – and even more so when using a linear and not exponentially distributed PWM brightness levels.

During these very short but vertically distributed exposure times, the PWM modulation of effectively 83 kHz results in active and not-active zones per LED. As the PWM modulation for the LEDs starts simultaneously with "on" for all three LEDs, they show up starting at the same horizontal line independent of the real physical position of the LED. The longer the active side of each LED's PWM modulation is, the wider its image stripe becomes. The distance between two "starting" sides of the stripes is equal to 255 PWM steps. Knowing this it is easy to measure image-by-image the PWM active cycle per LED and its step value.

Using the levels as defined in Figure 6, the symbol in Figure 8 translates to level 5 for the green LED (61 of 255 PWM cycles), level 7 for the red LED (150 of 255 PWM cycles), and level 4 for the yellow LED (34 of 255 PWM cycles).
With time as the vertical axis, and no longer only the LED position, there are strange effects to observe: The smartphone in Figure 8 is not perfectly aligned with the LTI device, so the line of LEDs is somewhat tilted counterclockwise and the yellow LED dot higher than the red one, and the green LED dot lower.
At the time of the first PWM modulation start in the picture, the blurred dot of the yellow LED is visible, but not yet the red and green LED dots. By the time the green LED dot would become visible, the PWM modulation for the green LED is already turned off again, so we do not see anything in the upper half of the green LED spot position. The red PWM modulation is much longer active, and so it becomes visible, as the LED is active at the time of the horizontal shutter moving downwards.
At the time of the third start of PWM modulation (bottom grey line), the green and red LEDs are still visible, while the spot of the yellow LED is also active but higher up and not where the shutter is capturing the photons – so it remains dark at the bottom.

The "rolling shutter" method would be great for high data rate detection – were there not two distinct drawbacks:

- This particular usage was never intended by the camera and camera-driver designers, and so it is not really supported as a feature.

- Sony and other image sensor developers see the rolling shutter effect as something bad and unwanted, and are working hard to eliminate it. It can be expected to become more and more difficult to expose in future smartphones.

4. Symbols and Symbol Rate

Using nine brightness levels per LED enables us to have a unique "off" value (value 0), as well as eight brightness levels to encode 3 bit of information per LED. With three LEDs, this results in 8 bits and an additional one-bit parity information. Just like in any other modulation, we call this arrangement a "symbol".

We also implemented a transmission mode with only five instead of nine brightness levels, resulting in a much larger brightness distance between the individual levels and thus a better "signal-to-noise" ratio and far more robust detection. However, this also results in only 2 bits encoding per LED, and thus only six data bits transmitted per symbol. To simplify the transmitting finite state machine we implemented two symbols per data byte and parity, which is then taking almost twice as long as 9-levels per LED transmission. This mode is only used as a "fallback" for a second data transmission if the first one was unsuccessful.

As the sender with the LEDs is not synchronized to the receiver (CMOS smartphone camera), we need to send the symbols at a slower rate than the recording camera. All globally available smartphones support a video capture rate of 30 frames per second, derived from the old NTSC television standard – which is most of the time effectively 29.97 frames per second. As we observed, some smartphones are not even guaranteeing this frame rate, but record as a "best effort" frame rate.
To stay on the safe side, and as there is no information from the used smartphone to the transmitting LTI device, we selected a transmission symbol rate of only 15 Hz. This will capture most symbols twice, but it guarantees that every sent symbol is captured at least once correctly for detection.

To select the correct images for data retrieval, and ignore duplicates is supported by the design of the signaling protocol and its rigid frame structure, together with synchronization symbols which can be unequivocally distinguished from payload data.

Instead of using a training sequence of known brightness levels to adjust the receiver the decision was to implement a simple 8-bit data scrambler to force the transmitted data to look random and hit all LED brightness levels about equally often. "Block-ID" and synchronization symbols are excluded from scrambling.

Before decoding the data, the receiver is then first assembling a list of all received brightness levels for each LED (excluding block-ID and synchronization symbols), and is grouping the levels per LED into eight levels with the k-means clustering algorithm. With these recognized levels, the payload is extracted from the symbols and restored to its original content through a second application of the scrambling sequence.

5. Parity information and Gray code

When distributing eight data bits evenly on three LEDs we get an additional 9th spare bit, which is used as parity information and first level model-based error correction. The parity bit is a simple even-extension of the payload data byte by XOR-ing all 8 bits together [6]. I can identify a single-bit error, but not any even number of bit-changes.

The most likely error case is a misinterpretation of the received brightness in one of the three LEDs, when it is decoded as either one level too low or one level too high. However, this will only work, if adjacent brightness levels always differ from each other by exactly one bit – if they differ by two bits a misinterpretation will not cause a parity error. Figure 9 shows the nine brightness levels. The lowest level (0) is not used for payload data but exclusively for "block-ID" and synchronization symbols. The other eight levels 1 through 8 have each a 3-bit data segment associated. From bottom to top, level 0 through 8 the applied Gray coding [7] is visible, where from any level to any other level only one bit changes.

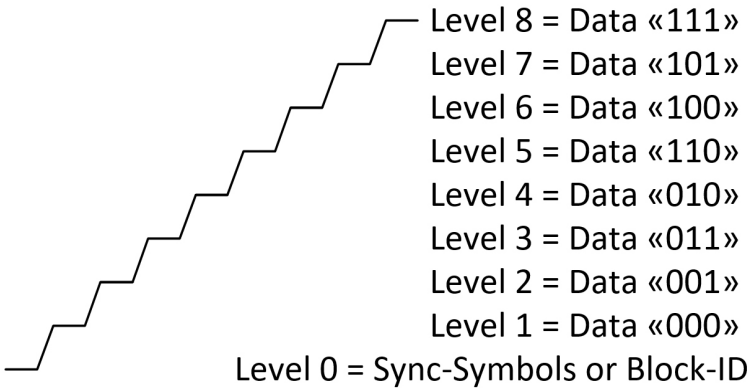

Figure 9: Gray code mapping of data to levels

The most likely error case is the misinterpretation of the brightness level of a single LED, while the other two LEDs are decoded correctly. This will then result in a parity error, at which time the model based first level of error correction can try to correct the mistake:

- If one of the three decoded LED brightness levels is particularly distant from the average brightness level value as derived with the k-means clustering algorithm, it is most likely the culprit and belongs to a different level

- If the green LED was decoded as brightness level 0, 1 or 2, it is most likely the one causing the error. This is a model based error correction, as countless experiments and testing have shown the green LED at the lower levels to be the most difficult one to detect – independent of the actual LED manufacturer!

6. PWM modulation of the LEDs

6.1 Round-robin activation

Once again, this proved to be a bit more challenging than expected. As mentioned in the beginning, the battery current output is typically limited to 3mA, which is reduced further near the battery end-of-life to less than 2 mA.
Chip LEDs are typically optimized for high output, and not for lowest power consumption. Even the smallest like ROHM PICOLED series [8] have a forward current I_F of 1mA. This means that there must never be more than one LED active at any point in time, or the battery voltage could drop below the chip minimal voltage causing brown-out effects.

Our solution is to use a round-robin activation scheme, where the LEDs are only active one after the other, if indicated by their individual PWM modulation duration.

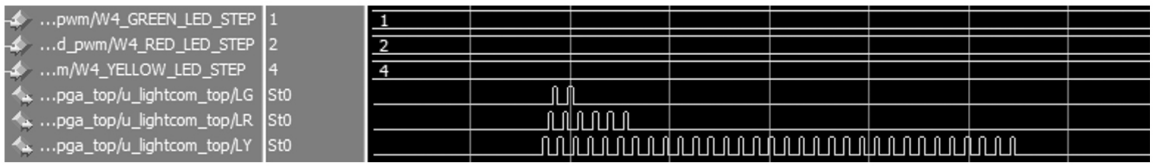

Figure 10: Round-robin LED activity simulation

Figure 10 shows the ModelSim waveforms for the LEDs in a typical PWM signaling. "LG" stands for the green LED, which in this example has only 2 PWM cycles, while "LR" for the red LED has 6 PWM cycles and "LY" for the yellow LED has 32. The PWM cycles start together for all 3 LEDs, and turning off individually when their duration is over. The PWM modulation frequency of 250 kHz is thus effectively only 83.3 kHz per LED, and the maximum LED brightness is even in full (255 of 255 cycles) activation only 1/3 of what it could be.

By using a black area around the LED openings on the device package, we found that this reduced illumination of the LEDs is still sufficient except for the usage in bright sunlight. As the product is typically used in offices and storage facilities, we consider this a workable concession to the small and long-lasting battery.

6.2 Symbol-change to PWM synchronization

Another important issue was to synchronize symbol-changes to the PWM modulation. Otherwise a symbol change for example of the red LED from a 5 cycle modulation to a 64 cycle modulation can have a "restart" effect, if the symbol-change takes place for example when the PWM counter is at 30. If not synchronized, the red LED will in this case turn on again at PWM cycle 30 and be active all through cycle 64. This synchronization needed in our case another 12 Flip-Flops, but proved significant for "rolling-shutter" based LED decoding.

7. Design of the signaling protocol

7.1 Symbol coding

Based on the 9-level brightness per LED and thus 8 data bits per symbol there are 84 symbols needed to transmit the LTI payload data. A rigid frame structure with additional "block-ID" and synchronization symbols ensures that payload data is correctly identified.

To make them unique and clearly distinguishable from payload data, "Block-ID" and synchronization symbols contain the otherwise never used level 0.

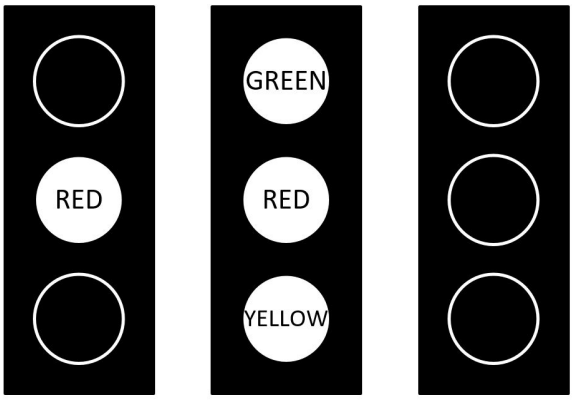

Figure 11: From left to right: block-ID, data- and synchronization symbols

Figure 11 shows left the "Block-ID" symbol. While the green and yellow LEDs are completely turned off, the red LED can have level 1, 3, 5 or 7 identifying it as "Block-ID 1" through "Block-ID 4". This identification permits an "out-of-sequence" payload decoding and assembly, and thus 4 possible entry points to start receiving and decoding the data transmission. The red spot is marked with color shading to indicate that it can hold different values.

Figure 11 shows in the middle a regular payload symbol, where each LED can have any brightness levels 1 through 8. The rightmost symbol shows the synchronization symbol where no LED is illuminated (all LEDs at level 0).

7.2 Framing structure

Figure 12 lists the entire framing structure with its 96 symbols to transmit 84 bytes of payload data and checksum together with the "Block-ID" and synchronization symbols.

Transmission starts with "Block-ID 1", marking this as the real beginning of the frame. This is then followed by 23 symbols as the "data block 1". This is repeated 4 times for a total of 96 symbols including 84 bytes of payload data and checksum. After that, the whole frame starts over again, repeating the frame for as long as it is configured. At a symbol rate of 15 Hz and 96 symbols, one frame lasts for 6.4 seconds.

With a “simple” receiver logic, the reception has to start with a “Block-ID” and has thus to wait in case of just missing a “Block-ID” for 23 symbols, which translates to another 1.53 seconds. With this, maximum recording time for the smartphone receiver must be 8 seconds.

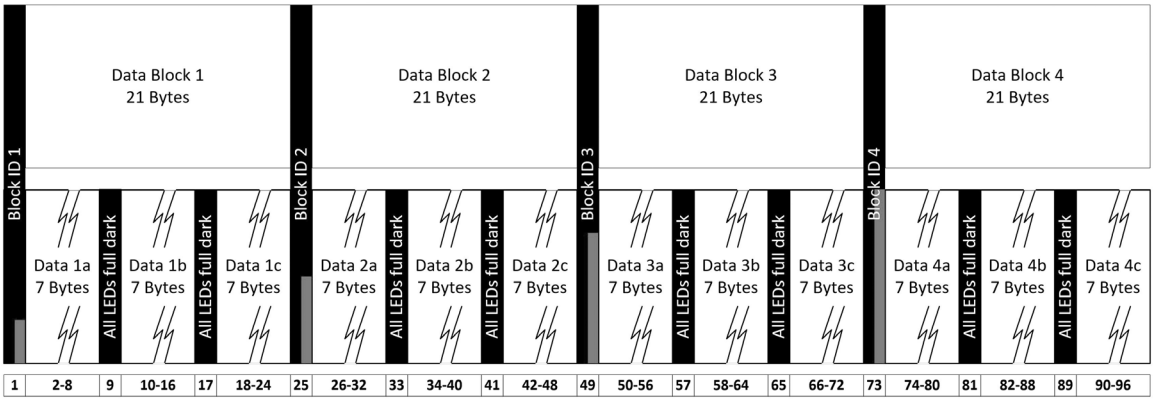

Figure 12: Framing structure with 4 data blocks

However, with a smarter receiver logic reception can start with any “Block-ID” or synchronization frame, which reduces receiver recording time to only 7 seconds. We have found that any successful decoding must start with a “Block-ID” or synchronization symbol, as this is really needed to identify the correct video frames to use for decoding.

Transmission with repetition of the frames can last as long as it is needed – but again there is no feedback from the receiver when it is done. We have found that one minute is typically sufficient to press the start button, grab the smartphone with the App already running, aim at the LEDs and record the blinking sequence for 8 seconds.

7.3 Receiver challenges using smartphones

Estimating the workload of the smartphone processor and the available real-time resources, we expected the main processor of a smartphone to perform the video compression during recording. If we therefore use image signal processing instead compressing and storing a video stream, the processor workload should be lower than during regular video recording.

This proved to be wrong. As we found, almost all modern smartphones are using CMOS camera sensors from a single manufacturer and technology: Sony Exmor RS [9]. Exmor RS is a stacked chip technology, with a back-illuminated CMOS image sensor on the first layer, and an added (stacked) second layer chip for advanced functionality [10]. Part of this functionality is the real-time video compression, which is offloading work from the main smartphone processor. This is actually bad news for this type of application, since the raw image data is no longer directly available on the processor, but takes additional processing power to extract the image-by-image data from the captured video stream.

Another finding was a widely used but little know technique called "Variable Frame Rate" VFR. This is an inherent part of modern video compression formats like MPEG-4 [11], where images are no longer recorded at a constant rate, but rather at "best effort" depending on the video compression needs and sometimes the configured constant output data rate. While this is compensated on the playback devices using individual time-stamps per frames, it poses an additional obstacle to video data processing for communication.
On a side note, this problem can be observed when regular video editing tools specialized for constant frame rate processing are fed with smartphone videos. Forcing the variable-frame-rate video stream image-by-image onto a second video stream (picture-in-picture or smooth transitions) will always cause synchronization faults between the video and audio stream [12].

To make the transmission more robust against fluctuations in the recorded frame rate of the smartphone, every set of 7 data bytes is framed to the left and right by either a "Block-ID" or synchronization symbol. As a result, the decoding is still correct for a set of 8 symbols
(7 data- and 1 synchronization symbols) if the camera records 15 or 17 images instead of the expected 16. This translates into a variation of +/- 6 percent of the captured frame rate. As we observed on different smartphone cameras, the overall frame rate is stable, but there are often significant short-term variations in the video recordings.

8 Checksum and Forward Error Correction

8.1 Reed-Solomon checksum generation on the transmitter side

In order to check data integrity and provide some error correction capability, we implemented the computation of 4 Reed Solomon syndromes over the 80 bytes of payload data.
This enables us to either correct up to two errors at any two locations, or up to four errors with known location. Based on the symbol-wise parity bit we do have a possible location information, which could be used to directly correct the errors or just feed them into the Reed Solomon algorithm.

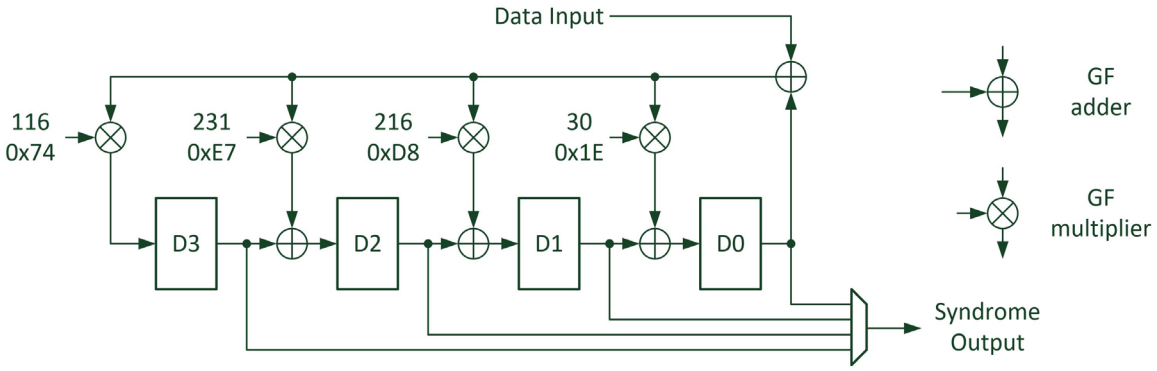

Figure 13: Reed Solomon syndrome computation

Syndrome computation for Reed Solomon is rather simple in hardware [13]; all it takes are 8 Flip-Flops per syndrome and a 2^8 Galois multiplier. While an addition in the Galois field is nothing but a simple bit-wise XOR, the Galois multiplier can be implemented as either 142
2-Input XOR gates, or in a bit-serial way spread over 8 cycles with fewer XOR gates but additional control and MUX logic. Direct comparison of the respective synthesis results proved the direct approach with only XOR gates to be more area efficient.
The multiplier itself could also be shared between the four syndromes, or simply be implemented four times in parallel. In this case the synthesis results showed that the control and multiplexing logic for shared hardware is more area efficient than a direct parallel implementation. This implementation then resulted in a small FSM with 6 states.

8.2 Error correction on the receiver side

While hardware efficiency on the LTI device side was very important, we had much more computation power on the receiving smartphone. Instead of just implementing parity-bit based local error correction and Reed-Solomon only once, we decided to use a multi-pronged error correction strategy trying many different paths of correcting or not correcting parity-bit errors, or only correcting some of these errors before feeding each into the Reed Solomon algorithm. Whenever we had a "clean" result, we used that instead of requesting another data transmission.

8.3 The Downside of Reed-Solomon error correction

If the Reed-Solomon error correction algorithm comes up "clean" with no errors corrected, we have a $1 : 2^{32}$ chance against having errors such that they still result in a "clean" check – which is a chance of 1 against 4 billion and thus good enough for every data usage.

However, it is rather difficult to specify the chances of accidentally identifying incorrect data with exactly one Reed-Solomon error correction. Common sense would claim that the correction of one Reed-Solomon error uses 2 bytes of the checksum, leaving two other bytes to guarantee a $1 : 2^{16}$ chance against errors – which is a chance of 1 against 65'565. This is not very good for medical applications, and mathematicians may prove even this assumption to be too optimistic.

In case of two error corrections with Reed-Solomon, there is no indication at all, whether there were only two errors at unknown locations to begin with, or whether there were more errors and only two corrections were done. As we observed, the Reed-Solomon algorithm not only gives no hint whether there were two or more errors – but even the two corrections in case of more than two errors are typically wrong … changing otherwise correct data just to match the Reed-Solomon equations.

We therefore should have either replaced the Reed-Solomon syndrome computation with a much smaller CRC32 algorithm - or add a CRC32 checksum on top of the payload data before computing the 4 Reed-Solomon syndromes. This would then result in a very high confidence of data corrected with two Reed-Solomon replacements.

For the ASIC, this finding came too late, as it would not only add 32 Flip-Flops, some multiplexers and glue logic to the design, but also extend the data trans-mission to 88 bytes … what is then no longer evenly dividable across the blocks and segments. As such, it would have changed the entire framing structure, which was not welcome that late in the project. This is therefore definitively something to do better next time.

9. Fallback mode and other options

Should the first transmission not go through, any subsequent start of a transmission within less than 2 minutes will trigger the "fallback" mode. In this case the 9-level signaling per LED will be reduced to a 5-level signaling and thus using 2 symbols per byte. This will give a significantly better LED brightness detection at the cost of a longer download time of 12 seconds instead of 6.4 seconds.

As part of the research, many other configurable options were designed and included, like 10Hz and 30Hz symbol rates, but the methods described here were the most useful and successful. All flexibility in the LTI device is useless, if you have no communication to change it after shipping the devices. There is the necessity to select one best method of operation and then stick to it, as thousands of devices are shipped and remain in the field for years to come.

With the transmitter cast in an ASIC, the only thing to improve further is the receiver App. With software upgrade distribution through App stores at almost no additional costs, there is a lot of flexibility and less roll-out pressure going forward.

10. Implementation

Except for the data RAM access, which was provided by the mixed signal ASIC design house, the entire code on the transmit side is written in Verilog. It consists of 8 files with 1200 lines of actual code, which may not seem that much. However, over the course of 1 ½ years, we did a lot of FPGA verification and iterative improvements.

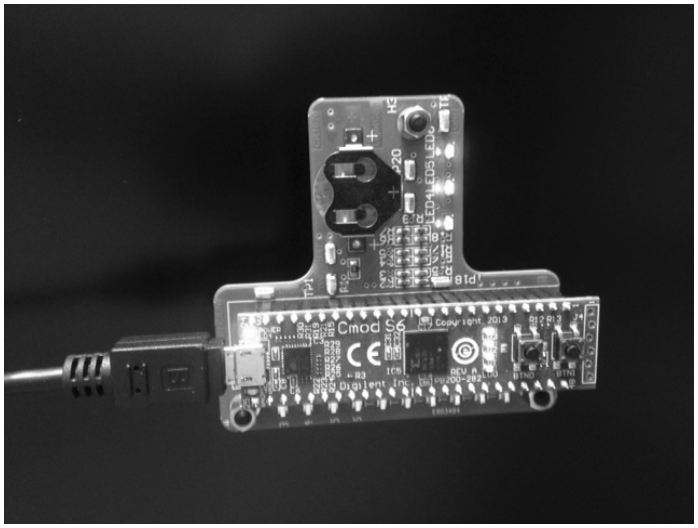

Figure 14: FPGA implementation of the transmitter

Figure 14 shows one of the verification systems we used. This consist of a simple Xilinx Spartan-6 FPGA board with a custom extension holding the battery and the three LEDs.

With this methodology, we could develop and calibrate the transmitter as well as the receiver Apps long before the ASIC tape-out and thus reducing the risk for the chip.

The transmitter implementation by the name "LightCom" needed 64 D-Flip-Flops and 808 logic gates. Implemented on the 0.35µm XFAB process in requires with routing less than 90'000 μm^2, or a square of roughly 300 x 300 µm. With about 21% of the logic area and 9% of the entire mixed signal ASIC area, this implementation does not come entirely free. But then again, the cost is in the single-digit cents range per chip.

11. Conclusion

Within a Swiss government funded research project, we had the opportunity to explore, design and implement a one-way communication from a low-cost battery powered device to a smartphone. We have shown how such a communication can download data at better than 10 Bauds, which is good enough for smart sensor data and/or sensor history.

The data rate is not impressive when compared to other typical linking technologies – but the price is unchallenged:

- On the transmitter side:
    - 10 cents for the logic area and drivers on a 0.35µm mixed-signal ASIC
    - 0.4 cents per LED at 10k quantities
      = 20 cents per device or less

- On the receiver side:
    - Only a regular smartphone needed – no additional costs
    - Only a software App needed – no costs for software distribution

The technology, signaling method and framework are free to use. There are to our best knowledge no patents covering this field, and our industry partners have no plan to file for such patents. This publication is one step to ensure, that it becomes common knowledge and nobody else can claim this method.

We started our project using three existing LEDs, as they were already part of other functions of the device. For other applications, two or four LEDs may have a better transmission speed-to-cost performance. The signaling, symbol- and frame definitions can be easily adopted.

As for the team, we have learned a lot during this project, implementing our best ideas and see them grow into a viable product. Many thanks to the CTI (Commission for Technology and Innovation) Switzerland for supporting us. Also many thanks to the managers and engineers at ELPRO BUCHS AG for their partnership. In addition, many thanks to all our colleagues at the Interstate University of Applied Sciences of Technology for motivation, lunch conversations and feedback on new ideas.

References

[1] Interstate University of Applied Sciences of Technology Buchs, NTB
https://www.ntb.ch/en/rd/

[2] ELPRO BUCHS AG, manufacturer of stability monitoring for products in transit
https://shop.elpro.com/EntryContentPageServlet

[3] Libero ITS by ELPRO, Inexpensive multi-level indicator with wireless read-out
https://shop.elpro.com/ArticleContentPageServlet?action=show
&artdetail=101&key=ARTIKELNR&value=900620

[4] FDK Lithium, Cell Type CR2025 specifications datasheet
https://media.digikey.com/pdf/Data%20Sheets/FDK/CR2025.pdf

[5] Computer Vision on Rolling Shutter Cameras, Linköpings Universitet
http://www.cvl.isy.liu.se/education/tutorials/rolling-shutter-tutorial/

[6] Netfuture - Parity Bits – Error detection capability
https://netfuture.ch/tutorials/crc/parity-bits/

[7] Kautz, William H. (1954). "Optimized Data Encoding for Digital Computers".
Convention Record IRE (part 4): 47–57.

[8] ROHM PICOLED™ SML-P11x Series LED Datasheet
https://www.rohm.com/datasheet/SML-P11UT(R)/sml-p11-e

[9] Wikipedia List of Exmor RS sensors and their utilizing devices
https://en.wikipedia.org/wiki/Exmor#List_of_Exmor_RS_sensors

[10] Sony Exmor RS - Stacked CMOS Image Sensor
Back-illuminated and stacked structure provide advanced functionality
https://www.sony-semicon.co.jp/products_en/IS/sensor1/technology/exmor-rs.html

[11] Waggoner, Ben (2009-11-16). Compression for Great Video and Audio:
Master Tips and Common Sense. Taylor & Francis US. pp. 150–.
ISBN 9780240812137

[12] Allan Tépper, TecnoTur LLC: ProVideoCoalition “Framerate workflow for iOS”
https://www.provideocoalition.com/understanding-iphone-framerates-for-shooting-editing-distribution/

[13] C.P. Clarke: BBC R&D White Paper WHP 031, “Reed-Solomon error correction”
http://downloads.bbc.co.uk/rd/pubs/whp/whp-pdf-files/WHP031.pdf