Light & Engineering
Vol. 27, No. 6, pp. 152–158, 2019 https://doi.org/10.33383/2019–059

NAVIGATION SYSTEM BASED ON VLC TECHNOLOGY FOR STAFF OF HERMITAGE MUSEUM

Emil Z. Gareev, Yuri B. Sorokin, Igor M. Antropov, Anton E. Kurako, Antonina A. Puchkovskaya, and Vladislav E. Bougrov

ITMO University, Saint Petersburg
E-mail: gareev.e@itmo.ru

ABSTRACT

We developed a navigation system based on wireless visible light data transmission channel and an algorithm for the decoding on smartphones. The work aims to create an interactive navigation system inside the Hermitage Museum for museum staff. The system was designed for using a modern smart-phone device as a receiver, a conventional LED illuminator as transmitter and a RGB diode as a navigation point in each room of the museum. We developed a modulator for data transmission, an algorithm for receiving and processing information using a stock camera of an iOS-based smart-phone, organized a point-to-point network between the LED illuminators and the server with a full back-end and front-end communication. The system allows transmitting data with rates up to 2 kbps on distance up to 1 meter.

Keywords: navigation system, visible light transmission channels, VLC, RGB diodes, LED illuminator, luminous flux modulation

1. INTRODUCTION

In recent years, technologies that allow us to combine advanced functionality in conventional things have started emerging [1]. This trend has not bypassed the LED systems, an important task of which became the wireless communication through modulation luminous flux. Thus, VLC technology arose, using LED as an optical signal transmitter and a photo detector as a receiver [1]. The main signal transmission principle by such a system is based on the light emission modulation with a frequency of the order of several kHz and higher. The human eye is not able to perceive the frequency of flicker above 100 Hz, which allows you to make a data transfer using VLC technology imperceptible to a user. A conventional modulation of LED illumination On-Off Keying is based on changes of luminous flux intensity. Using a mobile device as a receiver in such data transmission systems allows you to make any information accessible to a large number of users, due to the absence of the need to have a special separate client module acting as a receiver, however, a mobile camera module shooting frame rate is often limited to (120–240) Hz, due to the lack of a camera matrix possession ability, which makes it necessary to apply special processing methods of shots. The target of this work was to develop the navigation system based on the LED illumination and VLC technology, and also create an algorithm allows using the maximum of channel capacity by special methods of working with camera frames.

2. SYSTEM DESCRIPTION

To demonstrate the work, a system was created consisting of an *Arduino* microcontroller with two LED modules: white phosphors LEDs and RGB LED. Whole work assumes that the Hermitage museum will have hundreds of similar *IoT* devices that are navigation gateways, so the server side should be capable of communicating with all the devices with low latency. This system has two main tasks:

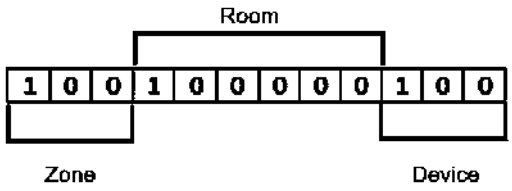


Fig.1. System number example

• Continuously send a message about its number using white diodes to smart-hone camera;

• Change the colour of the RGB diode to represent the path for user.

The system contains hundreds of devices with high frequency requests; it is possible to communicate with all the devices via one hub. Architecture consists of one hub and a number of gateways, where the hub operates the state of the navigation system and sends commands to gateways, which change their status for LED modules according to the proper path. In order to achieve maximum transmission capacity of the system, it needs secured local Wi-Fi connection, so all the devices will be located in one local network, at least, major part of the devices, and this can be achieved by using Wi-Fi boost adapters. Second, we should use optimized connection so that we could minimize overhead of transportation. We used UDP protocol as a transport layer for communication between the gateways and the hub, and compact byte data format. Each device of the system has its own number (Fig. 1). The number consists of three parts. 3 bits – zone number, 6 bits – room number, 3 bits – device number in the room.

System topology assumes that chances for package lost for most of the devices that are in local network are minimal and for the devices outside the local network in case of package drop we implement our own protocol handshake mechanism. Each gateway responses to a hub message, so in case of package drop, the hub that controls the state of the system makes additional attempts to contact with lost gateway. In case the gateway is dead, user who gets the directions is notified of a dead gateway in specific room on his path. All the mentioned above mechanisms make the system robust. In case of bigger scale, it is easy to make special services that read response messages from Kafka and synchronize the state of the navigation through Redis, in such case system can be scalable as any other micro-service architecture system and reliable even in high workloads with bigger scale.

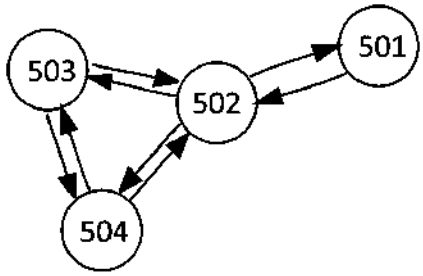


Fig.2. Example of maps of four connected rooms as a direct graph

The system always controls the whole state where each user path is mapped by colour so such structure will allow multiple search algorithm optimizations.

The system uses simple http protocol for communicating with smart-phone. Such approach allows a better and faster integration of system in third party clients and systems due to low entry threshold for http technology. In future hub will be rewritten in reactive stack for much more performance.

The whole map of the rooms are represented as direct graphs, so searching in such a data structure is not as simple as just in a simple graph (shown in Fig. 2).

The complete high-level navigation pipeline consists of several steps:

1. User scans nearby the gateway with its smartphone camera;
2. User prompts the destination room;
3. Hub receives data from user;
4. Hub calculates optimum and free route;
5. Hub notifies those gateways that are on the route and remembers their association with specific user;
6. Hub waits for responses from the gateways;
7. Hub retries N times for gateways that didn't respond in time.

Hub software was developed using Java language and can be used on multiple platforms not only x86 and ARM64, but also embedded devices with or without AOT compilation. Library has minimum third party dependencies and implements most of the functional using standard JDK library and API.

A smart-phone with a camera is an essential part for the system. It's a powerful and easy-to-use tool that connects sensors and the server and able to provide visual interface for user. The smart-phone has its own working algorithm that consists of following steps:

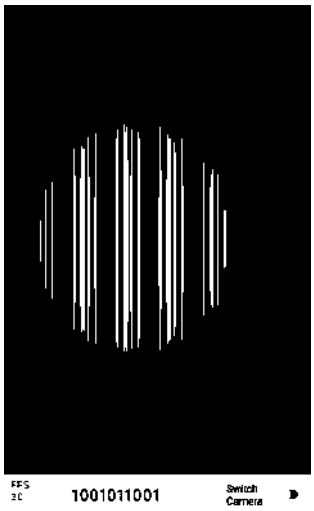
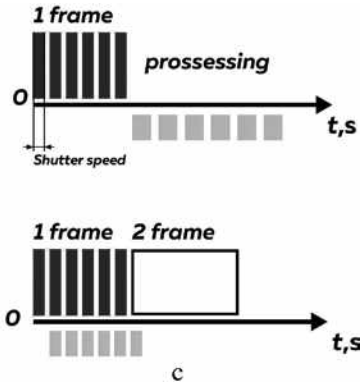


Fig. 3. Stripes

1. Setup the camera;
2. Read the signal, represented in bytes;
3. Decode the bytes sequence and convert it to the location;
4. Make server request with decoded location;
5. Display the path and additional information that comes from the server.

All the steps except the first one are made by software and can be implements on any phone. The first step, though, requires the smart-phone to work on the operation system that provides an interface to work with camera settings: exposure and ISO. Two most popular mobile platforms iOS and Android both satisfy the requirement. Due to efficiency and simplicity of the camera interface we have chosen iOS platform. All devices with iOS version are older than 4.0 have an ability to configure camera and capture visual frames with certain resolution.

As it was already mentioned, the first step is to setup the camera configuration in a way that allows us to see the black-white strips on the frames (Fig. 3). iOS platform allows us to change the exposure value between 1/2 and 1/1000000 and change ISO value between 29 and 464. After some experiments the ideal values occurred to be max possible ISO value and 1/6000 exposure value. With this configuration everything is ready to start capturing the frames. Every visual frame is a two-dimensional matrix with pixels as elements. The size of the matrix depends on the camera resolution of the certain device. The average size is around 1920x1080 pixels. Every pixel can be represented in different ways. For our purposes the image shouldn't be coloured, because bytes data can be represented as only white and black stripes. So, we have chosen *YCbCr* format in which every pixel consists of three components: luma component(*Y*) and two chroma components (*Cb* and *Cr*). The *Y* image is essentially a gray scale image of the main image.



c

Fig.4. Classical sequential (top diagram) and developed parallel (bottom diagram) algorithm with the rolling shutter effect implementation, where black lines are the group of camera module matrix rows that were exposed sequentially, according to the rolling shutter effect, and grey lines depict the time periods needed to process each of the matrix rows group

We created an algorithm that uses the "rolling shutter" effect. This effect is described in detail by N. Rajagopal et al. in [2]. The developed algorithm allowed us to process data at speeds up to 2 kbps with 5frames/sec shooting. The algorithm makes possible to minimize the loss of transmitted information. The Fig. 4 illustrated the main principal of rolling shutter effect and the implementation of this effect into developed algorithm.

When the session of the camera is started, the sequence of frames with the frequency of current exposure asynchronously comes for analysis. Each frame has three buffers, one for each component. First buffer contains *Y* image, so iterating through it gives the values for further decoding.

Even though only luma image is used, a lot of interference can cause the problems for results. Sensors emit consistent signal, which should be identical for the whole column of the frame. Finding the average luma value for each column reduces the amount of interference.

Luma value for each pixel is 1 byte that means that it can be any number between 0 and 255, so as 0 is black (dark) and 255 is white (bright). After averaging the column values, the array of the reliable values is got and it's saved to the FIFO buffer with arbitrary size. Bigger size of the buffer gives more accuracy but increases the amount of decoding time. When the buffer gets full, the analyzing

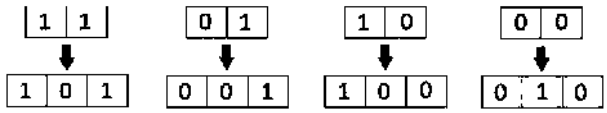


Fig. 5. Example of encoding

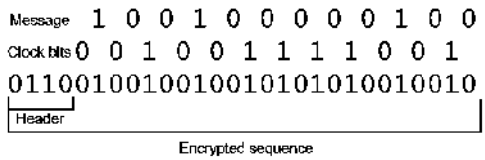


Fig.6. Full encoding example

is proceeded for every element. If all the elements transform into the same result it's sent to the server, otherwise the buffer cleans up, and cycles repeat. This sequence is another measure of correctness of the result.

3. RESULTS & DISCUSSION

Decoding at the smart-phone part is located at the sensor of camera. In order to get this value, the array of bytes goes through several steps:

1. Calculate border between 0 and 1;
2. Calculate min width of 0 and 1;
3. Iterate through sequence of widths, comparing current width with min width;
4. Decode location.

Even though the interference is reduced multiple times, the luma values are not perfect yet. Depending on the light, the range of the luma values is not always [0, 255], it can be shorter. To execute further calculations correctly, at first, the max and min value of current luma range is found, and the average value of these two is set as 0 and 1 border. It is used in next calculations for determining whether current luma value stands for 0 or 1.

The sensor emits a sequence of bits, which represents his location. This sequence can contain long sub-sequences of 0 or 1. The longer these sub-sequences are the higher change of false decoding is. In order to calculate the number of same bits in subsequence, firstly, the min width is calculated. Iterating through the array and comparing the current value with border value the array of widths is calculated. After that the min width values for both 1 and 0 are got. In some edge case, there is a chance that there is no single 0 or 1 in the sequence. For these reasons sensors always emit key sequence 0101, which always helps to determine single 0 and 1 width. After all the necessary values are got – the array of widths and min widths, it is easy to calculate the final bit sequence, just comparing current width with the min width at the same number (0 or 1).

The last step of the decoding process is to calculate the location. Location is represented by ids of the sensor and consists of different parts as shown above. Each part determines the zone, the room and the device (door where sensor is set). After converting the result sequence of 0 and 1 into decimal numbers the decoding process is complete.

At the device part a zero is encoded as 10 if preceded by a zero, and 00 if preceded by a one; a one is always encoded as 01 (Fig. 5). This allows we can avoid the case of two 1 bits following each other. We decided to use a header 0110, to uniquely identify the beginning of the message. Also this can help to count the width of columns.

Final encoded message is shown in Fig. 6. For correct representation of the path, the system contacts the server using the UDP protocol. When receiving the new path command from the server, system adds it to the list. The colour change of the RGB LED occurs according to the list of paths and colours. When receiving the finish command, the system removes the path from the list.

The whole logic of path between rooms is performed by the server. In order to provide it necessary data (which is start and destination point) the network request is performed. Using ordinary http request with data in body parameters the data is provided to the server. As soon as the server responds with the path it is displayed on the screen. The path is represented by sequence of rooms and the colour, which is unique and only assigned to this path.

Search algorithm at the server part plays important role in such a system because it essentially is the most complex computation activity that is presented over the whole application. The more efficient this part can be, the more responsive whole pipeline can be.

Core algorithm for path search we are using is Dijkstra algorithm. This is one of the classical algorithms for finding the shortest route in graphs. As our system uses direct graph for map, such algorithm is also working with this data structure. As our search algorithm should not just find the shortest path, but also a free path (where one LED colour is not occupied by other route), each node in directed graph also contains a set of occupied colours.

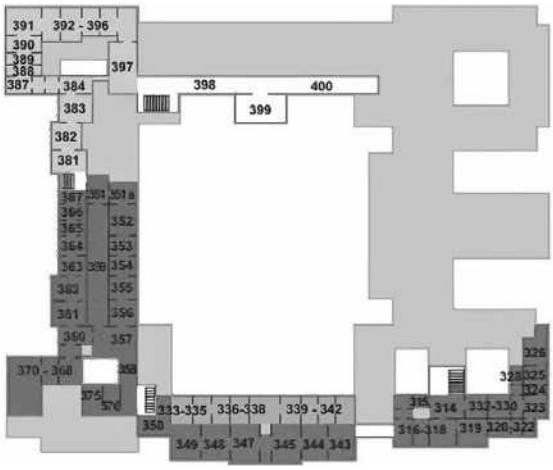


Fig.7. Visualized map split into four sectors

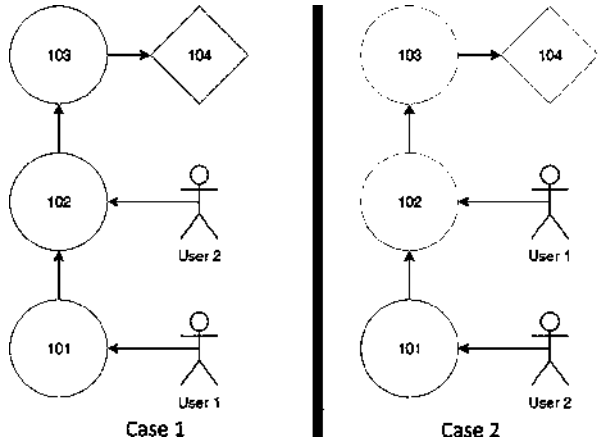


Fig.8. Route optimization strategies

Fundamental principle of such an algorithm is that it literally searches through all possible routes to find the shortest one. This is not quite acceptable in our situation due to performance issues. We managed to optimize such an algorithm and visualization of such a map with an assumption that our map consists of rooms. We can split it in sectors, where each sector is minimized by its enter sectors (rooms that correlate with rooms from another sector), it means that rule to split map in sectors is that each sector should have minimum rooms that collide with other sectors and the size of the sector is reusable by other potential sectors. Such work is processed by humans as for now, but could be delegated by introducing special algorithm for dynamic sector mapping.

Such an approach with sectors allows us to remember shortest paths from boarder rooms from one sector to another, so in future we can just jump through sectors as after first full Dijkstra algorithm scan we remember the shortest jump routes from one sector to another. This mechanism can be applied on bigger scale for different map sizes that are direct graphs. Such algorithm allows us gain significant performance boost and make Hub even reliable and responsive as a part of such architecture.

Visualization represents four sectors with a number of Hermitage museum rooms, where each sector is grouped by its geographic location (Fig. 7). With such sectored map, it will be possible to easily jump through red sector when the route is from any of the rooms from purple sector to yellow sector. This can be done after first full calculation of all routes using standard Dijkstra's algorithm, after that algorithm will remember jump paths through red sector and next calculation will take a lot less time.

Implemented search algorithm also allows optimization for already initiated routes. Two strategies can be applied for route optimization (Fig. 8). In first case, if User 1 calculates route and such route is assigned to blue colour. Then every other user that is building route from every room on such route will be assigned to same blue colour. So technically route will be shared for all users who have same destination and their start room is presented on such route.

In second case, User 1 calculates a route and gets orange colour assigned. Then User 2 wants to calculate route from room 101 but his destination room is the same as User's 1. In this case, first route can be expanded backwards up to room 101 and then assigned to User 2. However, in some cases User 2 can be assigned with another colour and when he reaches room 102, it's notified that his path colour has been switched up to yellow colour.

Such a mechanism allows reusing routes and expanding them if necessary. Also in some cases when there are many users it is essential to minimize colours so gateways will have as few colours as possible and their switch timeframe will be minimal. For those purposes, users can have multiple colours on their route to ensure that switch timeframes will be minimal and maximum amount of routes will be reused and linked together for different type of users. However, there should be a threshold for such a case, so there should be maximum 3–4 colour switches during the route in order to have an appropriate user experience, but it also depends on what scale system is operating.

As it was already mentioned, UDP is used for protocol for communication between hub and gateway. First reason for UDP protocol as mentioned above is the ability to minimize protocol mecha-

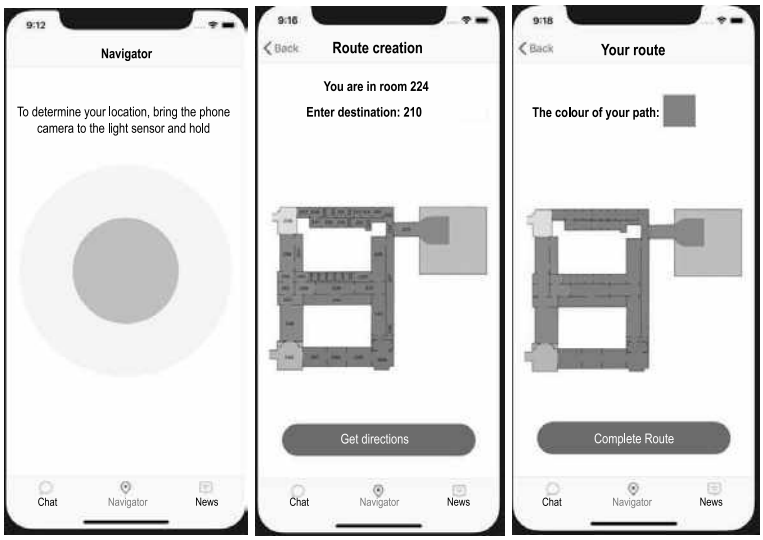


Fig. 9. iOS application

nisms for handshake or another communication between clients during package sending. UDP allows making simple things as package sends without any guarantees of receiving from protocol but with minimal package size and transfer speed. System protocol utilizes UDP broadcast technology that allows broadcasting package through whole network for better and faster data propagation. All those criteria are essential for *IoT* devices and whole system responsiveness itself.

Custom implementation for receiving handshake mechanism allows us to perform a more precise control over the way how gateway and hub communicate with each other and to ensure that workflow of the whole navigation process can be controlled from hub by developers. Also implementing custom handshake mechanism allowed us to minimize latency in comparison with TCP protocol.

UDP protocol allows us to utilize wireless connection at its maximum by just sending only valuable payload with minimum latency. In addition, other protocols such as TCP require much more resources, what may be unacceptable for *IoT* devices. Such protocol implementation allows us to use cheap hardware and perform most of calculation and business logic on the hub side.

Package that contains commands for the gateway is compact as it can be; it contains only route id and colour for that route. All those communication mechanisms and optimization on protocol levels as a package compression allows minimizing costs for implementing and supporting such system, which is crucial for business on initial phrase when the question of considering technology integration is opened.

4. CONCLUSION

In this work we demonstrated the data transmission via LEDs and mobile device camera and developed the processing algorithm for camera capable to decode signals with modulation frequency up to 2 kHz with the MFM algorithm with modifications OOK. Also we developed iOS application: screenshots shown in Fig 9.

This system can be used at the Hermitage museum for staff navigation. The colour of RGB diode will give intuitive path for each museum worker (smart-phone user). An exemplary view of the indoor system is shown in Fig. 10.

The future plan is the increasing of data transmission speed due to the introduction of special correction factors in the data processing algorithm, which makes camera solve long sequences better. Also the system has multiple extension points as well as optimization ones. The first system topology can be restructured using ZigBee protocol. Such

Fig.10. Application indoor example

a protocol could ensure that system stability will not depend on single point of failure as a local network with Wi-Fi router. ZigBee also introduces such a concept as coordinator but the whole topology is a communication between each other without using single channel such as Wi-Fi. In addition, the technology has far less energy consumption, which is also critical for IoT devices. So, there is no need of infrastructure setup for such a system, only the gateways should be installed. However, modules with such a technology cost more than any other hardware, so the system's price will be quite higher. This is an option and it depends on business requirements over all.

Reactive stack for http hub communication is for better performance. In addition, architecture can be expanded with facade services that reroute http requests to hub and play role of balancers to ensure hub stability and expect system failures in case of DDoS attacks.

ACKNOWLEDGEMENTS

This work was supported by the Ministry of Science and Higher Education of the Russian Federation in the framework of the Federal Target Program "Research and development on priority directions of scientific-technological complex of Russia for 2014–2020", the code 2017–14–582–0001, agreement № 14.581.21.0029 of October 23, 2017, unique ID RFMEFI58117X0029.

REFERENCES

1. Haas H, Yin L, Wang Y, and Chen C2016 What is Li-Fi? *J. of Lightw. Tech.* 34, 1544.

2. Rajagopal N, Lazik P and Rowe A 2014 *Visual Light Landmarks for Mobile Devices* (IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks).

Emil Z. Gareev,
graduated from the ITMO University magistracy in 2016. At present, he is post graduate student of the Laser Photonics and Optoelectronics Faculty, ITMO University

Anton E. Kurako
graduated from the bachelor program of the Computer Engineering Department of ITMO University In 2018 and up to now he is continuing his studies in the magistracy, ITMO University

Igor M. Antropov
graduated from the bachelor program of the Mathematics and Mechanics Faculty of the Saint Petersburg State University in 2018. At present, he is the Master student at ITMO University

Yuri B. Sorokin
graduated from the Faculty of Software Engineering and Computer Engineering of ITMO University in 2018. At present, he continues his studies in the ITMO University magistracy

Antonina A. Puchkovskaya,
Ph.D. in Cultural Science, she was a post graduate student at the Institute of Philosophy of Saint Petersburg State University in 2012. At present, she is a Head of the International Laboratory for Digital Humanities Research at ITMO University

Vladislav E. Bougrov,
Associate Professor, Doctor of Phys.-Math. Sciences.
In 1996, he graduated from the Department of Optoelectronics of the Saint Petersburg State Electrotechnical University named after V.I. Ulyanov (Lenin). At present, he is Director of the mega-faculty of Photonics, Professor at the Faculty of Laser Photonics and Optoelectronics at ITMO University