# Smart Realty Agent

Andrey Pimenov
Center for Design, Manufacturing, and Materials,
Skoltech
Moscow, Russian Federation
andrey.pimenov@skoltech.ru

Artem Vasilyev
Space Center
Skoltech
Moscow, Russian Federation
Artem.Vasilyev@skoltech.ru

***Abstract*—This document is a personal contribution report of “Smart Realty Agent” team project for Skoltech course: “Sensors and Embedded systems for IoT - 2019”.**

***Keywords*—Real Estate, embedded system, ESP32, LIDAR, Digital Camera Interface (DCMI), 3D visualisation.**

## I. Digitalisation of the Real Estate Market

Existing technologies, such as Virtual Reality, make possible choosing a flat or any other real estate without physical attendance of one or the other participant of the rent flat agreement [1]. Digitalisation of this market changes traditional business models [2]. Airbnb [3] is one of the companies, that successfully disrupts it. Some experts expect that the biggest technology that changes the real estate market will be a blockchain [4]. According, these experts this technology in conjunction with inside the flat VR-attendance destroy traditional housing agencies business. So, companies need to adopt and implement new technologies to survive in this market.

VR model of the current real estate object needs enough quality 3D-map of the indoor environment from the one hand, and get it fast from the other. There are two main technical solutions of this problem: a) The simultaneous localization and mapping (SLAM) algorithm [5] with mobile robotics [6]; b) 3D localization or mapping methods based on Lidar sensors [7].

As a solution, we observe the system that supports and gathers data from each room of the flat and send this information to the server. For the course project, we focused on the following: a) design an embedded system based on appropriate MCU for gathering data from the LIDAR and camera and sending this data to the server; b) make a heat map creation algorithm for providing 3D visualisation of the room. I was responsible for design this embedded system.

## II. Project Description

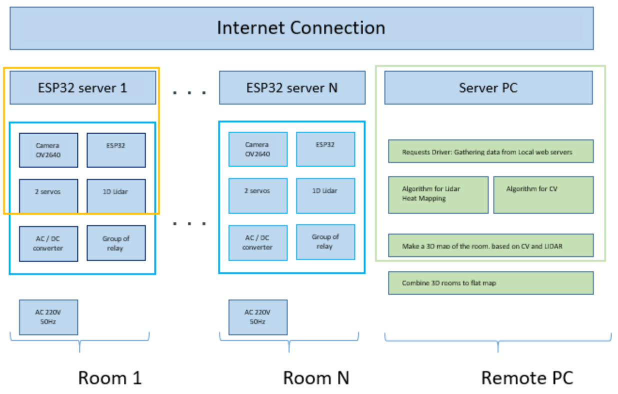
### *A. Goals and Observations of Embedded System*

Embedded system of this studying project was designed for sending data from the sensors (1D LIDAR and DCMI Camera) to the server. It should be mentioned that the motion platform has to provide additional measurements of dimensions to the system. Therefore, I’ve added 2 servos motors. As a result, gathering data from the 3D area by the internet connection have been transported to the server.

### *B. System Architecture*

General Technical requirements for this embedded system were the following:

1. Gathering data from the DCMI camera and send it to the web-server;
2. Gathering data from the LIDAR and send it to the web-server;
3. Opportunity to control the group of relay for imitation of some technical aspects of the smart house, such as a smart locker or opening the blinds, etc.



Picture 1. System Architecture

According to this scheme (Picture 1) System consists of N embedded system devices, and one Server PC with designed software on it. Different colors of the group boxes support different meanings: Blue group box – the embedded system itself. Green group box – software design, my colleague was responsible for this block. Orange group box – which I focused on the project and finally achieved.

Embedded system managed by MCU and controlled both: physical layer (2 servos, 1D Lidar, 1 Camera, relay group) and non-physical layer (ESP32 web server). Data gathering from the sensors:
a) Prepared: pre-filtering of images from the Camera and format changing data from LIDAR
b) Sending data through Wi-Fi to web-server.
c) Asking web-servers by requests from the Server PC for getting information and use it in algorithms of 3D mapping.

### *C. Comparison with other solutions*

The main sensor of our system is TF mini LIDAR [8]. Let’s compare our one from the LIDAR of 3D localization method in indoor environment project [7]. Comparison table demonstrates the different between TF mini LIDAR of our project and HDL-32E Velodyne LIDAR. The second one has a distance in 8.33 larger than there is in our project. Motion system was already installed that makes the movement of the HDL-32E more smoothly. And finally, it has much more points per second.

| TF mini LIDAR | HDL-32E Velodyne LIDAR |
|---|---|
| Up to 12 meters | Up to 100 meters |
| 115200 bits per second / 16 = 7 200 points per second | Up to ~1.39 Million Points per Second |
| 1 channel | 32 channels |
| 1Dimesion | 360° Horizontal FOV |
| | and +10° to -30° Vertical FOV |

Picture 2. Lidar Comparison

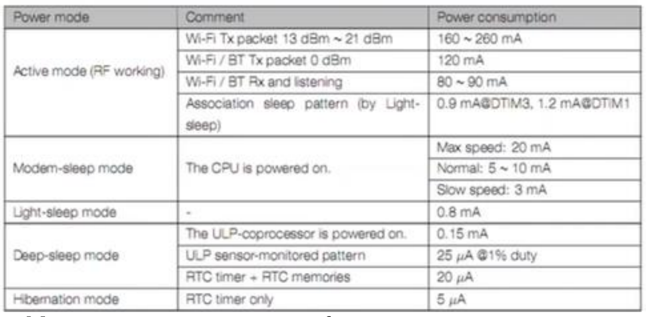
According to this, the solution in paper [8] has a more precise and reliable device on starting.

## III. My Contribution

Choosing the MCU for the embedded system at the begging of the project, was one of the most important contributions. I choose MUC form Espressif Systems [9]. There are reasons why I've chosen ESP32:

1. Xtensa® Dual-Core 32-bit LX6 80-240MHz up to 600 DMIPS
2. Ultra LOW Power solution
3. 448 Kb ROM
4. 520 Kb SRAM
5. Beacon monitoring
6. Power supply: 2.2V to 3.6V
7. Wi-Fi 2.4 GHz
8. Bluetooth 4.2
9. Low cost less than 800rub

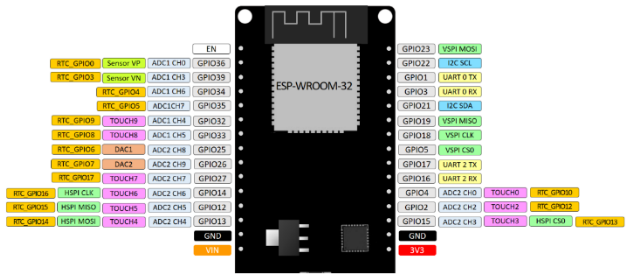
ESP32-WROOM-32 module has few modes for power consumption [9], demonstrated below in Table 1:



| Power mode | Comment | Power consumption |
|---|---|---|
| Active mode (RF working) | Wi-Fi Tx packet 13 dBm ~ 21 dBm | 160 ~ 260 mA |
| | Wi-Fi / BT Tx packet 0 dBm | 120 mA |
| | Wi-Fi / BT Rx and listening | 80 ~ 90 mA |
| | Association sleep pattern (by Light-sleep) | 0.9 mA@DTIM3, 1.2 mA@DTIM1 |
| Modem-sleep mode | The CPU is powered on. | Max speed: 20 mA |
| | | Normal: 5 ~ 10 mA |
| | | Slow speed: 3 mA |
| Light-sleep mode | - | 0.8 mA |
| Deep-sleep mode | The ULP-coprocessor is powered on. | 0.15 mA |
| | ULP sensor-monitored pattern | 25 μA @1% duty |
| | RTC timer + RTC memories | 20 μA |
| Hibernation mode | RTC timer only | 5 μA |

Table 1. Power consumption of ESP32-WROOM-32

Next thing, that needs attention is pin configurations. There are a few different modules drives by ESP32, but in this projec, I've used an ESP32-WROOM-32.


ESP-WROOM-32
EN
RTC_GPIO0 Sensor VP ADC1 CH0 GPIO36
RTC_GPIO3 Sensor VN ADC1 CH3 GPIO39
RTC_GPIO4 ADC1 CH6 GPIO34
RTC_GPIO5 ADC1CH7 GPIO35
RTC_GPIO9 TOUCH9 ADC1 CH4 GPIO32
RTC_GPIO8 TOUCH8 ADC1 CH5 GPIO33
RTC_GPIO6 DAC1 ADC2 CH8 GPIO25
RTC_GPIO7 DAC2 ADC2 CH9 GPIO26
RTC_GPIO17 TOUCH7 ADC2 CH7 GPIO27
RTC_GPIO16 HSPI CLK TOUCH6 ADC2 CH6 GPIO14
RTC_GPIO15 HSPI MISO TOUCH5 ADC2 CH5 GPIO12
RTC_GPIO14 HSPI MOSI TOUCH4 ADC2 CH4 GPIO13
GND
VIN
GPIO23 VSPI MOSI
GPIO22 I2C SCL
GPIO1 UART 0 TX
GPIO3 UART 0 RX
GPIO21 I2C SDA
GPIO19 VSPI MISO
GPIO18 VSPI CLK
GPIO5 VSPI CS0
GPIO17 UART 2 TX
GPIO16 UART 2 RX
GPIO4 ADC2 CH0 TOUCH0 RTC_GPIO10
GPIO2 ADC2 CH2 TOUCH2 RTC_GPIO12
GPIO15 ADC2 CH3 TOUCH3 HSPI CS0 RTC_GPIO13
GND
3V3

Picture 3. Pins configuration

Based on this, I design the connections of the circuit. On the table below, you can find all active connections to the module with comments for them.

| Pins on ESP32 | Circuit Connections | Comments |
|---|---|---|
| EN | RET on camera | Enable |
| 36 | D7 on camera | Data bit 5 |
| 39 | D6 on camera | Data bit 4 |
| 34 | D9 on camera | Data bit 7 |
| 35 | D8 on camera | Data bit 6 |
| 32 | VSYNC | Vertical |
| 33 | HREF | Horizontal |
| 25 | - | Free GPIO |
| 26 | - | Free GPIO |
| 27 | PCLK | Clock |
| 14 | XCLK | Clock |
| 12 | PWM Servo H | Horizontal move |
| 13 | PWM Servo V | Vertical move |
| GND | Servos & LIDAR | Servos & LIDAR |
| VIN | Servos & LIDAR | Power supply |
| 23 | D5 | Data bit 3 |
| 22 | SIOC | I2C_SDA |
| 01 TxD | - | for USB-serial |
| 03 RxD | - | for USB-serial |
| 21 | SIOD | I2C_CLK |
| 19 | D4 | Data bit 2 |
| 18 | D3 | Data bit 1 |
| 5 | D2 | Data bit 0 |
| 17 TxD2 | RxD of LIDAR | UART 2 |
| 16 RxD2 | TxD of LIDAR | UART 2 |
| 4 | Relay module | Smart Locker |
| 2 LED pin | - | Data indication |
| 15 | - | Free GPIO |
| GND | Camera GND | Camera power supply |
| 3v3 | Camera power | |

Table 2. Circuit connection

### A. *Installation ESP32 libraries for Arduino IDE*

From the other hand, this MCU has a disadvantage from an IDE (Integrated Development Environment) point of view. Firstly, to cover this gap I start reading a special hand book: "Kolban's book on ESP32. September 2018" [10]. But then realized that is better to find solutions for more common wide IDEs. On GitHub, I found a channel of one of the developer of ESP32, who writes few libraries for ARDUINO IDE for supporting ESP32. [11]

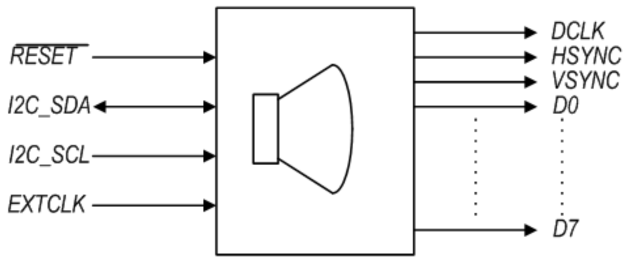
### B. *Wi-Fi Connection. Request. Web Server on ESP32*

For Wi-Fi connection I used special libraries: "esp_http_server.h", and original ARDUINO "WiFi.h". For understanding how to organize a web server on ESP32 I realized step by step following programs:

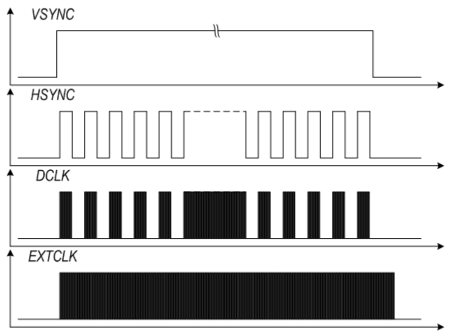
a) Connection ESP32 to hot point from my smartphone
b) GET & POST requests
c) Transfer simple variables from ESP32 to server

### C. *DCMI Camera Connection*

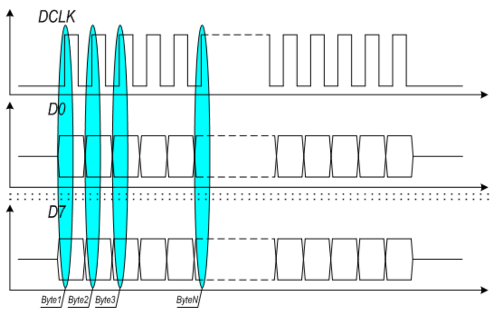
For Camera connection, I read about DCMI – Digital Camera Interface and understand how it works. On Picture 4 presented pins of the camera:



Picture 4. Digital Camera Interface (DCMI)
EXCTL - camera clock signal from outside
DCLK - strobe signal, on the leading or trailing edge of which data is recorded on the camera data bus
HSYNC - horizontal sync signal indicating the beginning of a new line
VSYNC - synchronization signal, the active level of which indicates the beginning of a new frame
D0 ... D7 - 8-bit data bus



Picture 5.
DCLK is only in the activated HSYNC phase. In our case, the camera is configured for a resolution of 640x480, then the HSYNC lays down 640 DCLK pulses during the VSYNC - 480 HSYNC period.



Picture 6.

The most useful for DCMI camera connection was an acritical on Habr – Habr, about STM32 DCMI connection [12].

As for software, there are three modules:
a) The camera configuration file, which represents a bit indexes of the camera chip.
b) Piece of code, that responses for creating Web Server on ESP32 connected to my phone internet hot point. Moreover, command word: “capture” is also designed in this part.
c) Script for transferring bits from the camera to the server.

## D. *UART and LIDAR connection*

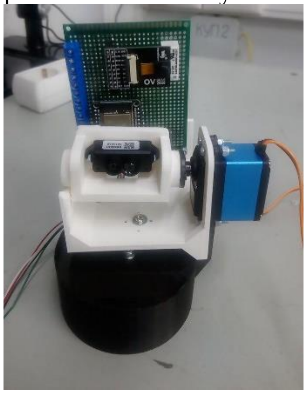
Here I meet unexpected problems. The deal is that Arduino IDE for ESP32 supports only 1 UART from 3, by default.

So, I learn how to create a new hardware UART by pins redefinition in software.

But then I meet the problem with LIDAR output configuration data. It sends to the serial port a HEX-value, but we will need a decimal. Software transformation could take calculation resources. So, I read the documentation for LIDAR TD mini and make a pre-configuration code in void setup {} function of my program.

## E. *3D Model Design*

I’ve designed a 3D models of the LIDAR by officially licensed for Skoltech SOLID WORKS and 3D printers in Masterskya on the base floor of the new campus.


Picture 7 Design of the final system.

By the way it is the third design, 1st one was for stepper motors, but I decide to use servos as it easier for control (there is not needs in driver board or additional power supply), 2nd design was focused on camera connection by flexible bus wires, but few experiments show me that long wires make picture worse. So I designed 3rd one with MCU board and Camera on the device, behind the LIDAR on the moving platform.

## IV. Results and Conclusion

Lesson learnt:
a) Get started with new MCU. It opens the opportunity for design electronics solutions for IoT.
b) Working with requests and creating a web server on MCU. This is kind of a new experience for me.
c) Learn and work with DCMI interface/protocol for camera connection.

Result pictures and all code and data samples on my GitHub Account. [14]

## References

[1] https://www.omnivirt.com/blog/best-uses-virtual-reality-vr-real-estate/
[2] – Harvard Business School article about digitalisation real estate market: https://rctom.hbs.org/submission/digitalization-and-the-death-of-a-real-estate-salesman/
[3] - https://www.airbnb.com/
[4] - https://blockgeeks.com/guides/blockchain-real-estate/
[5] – Simultaneous Localization and Mapping: part I https://ieeexplore.ieee.org/document/1638022/authors or the same paper:
[6] – Roomba mobile robot official website

[7] – A 3D localization method in indoor environments for VR applications. https://hcis-journal.springeropen.com/articles/10.1186/s13673-017-0120-7

[8] – LIDAR TF mini documentation

[9] – Espressif Systems official web site: https://docs.espressif.com

[10] – Kolban's Book on ESP32, September 2018 Neil Kolban

[11] – Official Channel of ESP32 Develop: https://github.com/espressif/arduino-esp32

[12] – Habr Habr of CDMI https://habr.com/ru/post/186980/

[13] – Server connection and requests.

[14] - https://github.com/AndreyPimenov/AP_SDA/tree/master/ESP-32/REPORT