ANNOTATION

Diploma work 89 pages, 26 figures, 9 tables, 20 sources, 1 appendix

Keywords: home automation, smart home, internet of things, Raspberry Pi, UWP, Windows 10 IoT Core, Microsoft .Net, cloud technologies, Azure.

Object of the study: home automation systems.

Subject of the study: home automation system design and development.

The purpose of the work: to investigate the difficulties of the design and implementation of home automation system using Microsoft .Net technological stack and cloud technologies.

The objectives of the work: to study the theoretical basis of design and development of the complex distributed IoT system hosted in the cloud, to develop the working prototype of such system.

The results obtained and their novelty: the IoT system design and development features are studied, development environment investigated and studied, system prototype developed and tested on the real the real hardware.

Scope, economic efficiency (practical importance): developed automatic control system could be used in home automation independently or as a part of Smart home system. It allows house owner to be aware of the visitors that come to him during his absence.

CONTENTS

SYMBOLS AND ABBREVIATIONS LIST

IoT – Internet of Things;

OOD – Object Oriented Design;

IoC – Inversion of Control;

RPi – Raspberry Pi;

BEP – break-even point.

INTRODUCTION

The developed system refers to a class of home automation systems - systems capable to perform actions and solve certain everyday tasks without human involvement. Homemade automation is seen as a separate case of the Internet of things, it includes home-based Internet access, while Internet-related things include any interconnected via the Internet at all.

The most common examples of automatic actions in a "smart home" are automatic switching on and off, automatic adjustment of the heating system or air conditioner and automatic notification of intrusion, flash or water leakage. The developed system will play the role of "smart intercom", allowing the owner of the home to interact with visitors using a mobile phone program anywhere in the world where the Internet is available.

The purpose of this work is to analyze and design an automatic control system "Smart Intercom" with the usage of modern technologies and equipment. Accordingly, the system must be distributed, must consist of a control object that is located directly on the territory of the managed home, the server for storing and processing the information and mobile application as a presenter.

The control object is based on the Raspberry Pi 3 single-board computer running the Windows 10 IoT Core operating system. The required functionality is provided by an application written using Microsoft .NET and Universal Windows Platform technologies. Since the most promising services for hosting server applications are cloud-based technologies, the cloud-based services of Microsoft Azure will be used in this system. The mobile application is designed to expand on various platforms, as Xamarin technology is used in this project, and it allows to develop mobile applications for different platforms using a common code base.

1 ANALYTICAL REVIEW OF THE LITERATURE AND OF EXISTING METHODS OF SOLVING THE PROBLEM

1.1 General description of the problem and existing approaches to solve it

The developed system relates to a class of home automation systems - systems capable to perform actions and solve certain everyday tasks without human involvement. Homemade automation is seen as a separate case of the Internet of things, it includes home devices and applications accessible via the internet.

This system is a part of a "Smart home" system.

A smart home is an indispensable attribute of any modern home, in which there are so many different engineering systems: lighting, power electric, heating, ventilation, air conditioning, home theater technology, security and fire alarm.

A smart house is an intellectual management system that unites all equipment in a single complex that solves various tasks in the sphere of security, life support, entertainment and communication. Any smart house system consists of sensors, through which information is received, and actuators.

A smart house is a system of intelligent automation for controlling the engineering systems of a modern building.

The Smart home system provides a mechanism for centralized control and intellectual management in residential, office or public premises. With the installation of such a system at home or at work, each user gets next opportunities:

1) To set the parameters of his own individual environment (light, air temperature, sound, etc.);
2) To manage the necessary system (lighting, climate, video surveillance, etc.)
3) To get access to the information about the state of all life support systems at home (whether inside or remote);

The general scheme of the control system is as follows:

1) central processing unit / main control unit;
2) sensors (temperature, light, smoke, traffic, etc.);
3) control devices (dimmers, relays, IR emitters, etc.);
4) control interfaces (pushbuttons, IR and radio control panels, touch panels, web / WAP interface);
5) own management network that combines the above elements
6) managed devices (lamps, air conditioners, home theater components, etc.);
7) auxiliary networks (Ethernet, telephone network, distribution of audio and video signals);
8) project Software.

The Smart Home system provides a real-time management with using a mobile device or a PC located in a local area network or that has an Internet access.

The main goal of this work is to analyze and design an automatic control system "Smart Intercom" using modern technologies and equipment. So, according to the “Smart home” system requirements, the developed system must be distributed, must consist of a control object that is located directly on the territory of the managed home, the server for storing and processing the information and mobile phone with a special application as a presenter.

The developed system is considered to be a part or a subsystem of the “Smart home” system itself or to be implemented as a separate, independent control system, so it should be designed and implemented according to the requirements of an embedded system and IoT infrastructure.

1.2 Technical task analysis.

The main document, which contains the main initial data and requirements for automatic control system “Smart intercom” is specification, when system is designing. Careful analysis of existing specification for the presence of full required information for successful implementation of the system gives required specification items.

Specification contains the main information about automatic control system modes, about requirements for reliability and safety, about devices composition, about characteristics, disturbances, electrical parameters etc.

Automatic control system “Smart intercom” is intended to help householders to interact with their visitors being in any point on our planet where the Internet connection is available

This work should be implemented in 3 stages:

1) specification implementation;
2) detailed design;
3) automatic controls system design.

The main aim of this automatic control system is to provide the user friendly interface for communication with the visitors of the house using a mobile phone application. House holder should be able to see who has come to him, to communicate with the visitor, to see him and to lock or unlock the door using different ways depending on the situation.

The declared functionality of the automatic control system “Smart intercom” is next:

1) user should be able to unlock the door using the smartphone application from any point on the planet where internet connection is available;
2) user should be able to unlock the door being in the building using the smartphone application even without internet connection using wireless technologies;

3) user should be able to unlock the door without smartphone using his credentials;
4) user should be able to provide the special authorization token (QR-Code) to guest, which can be used to unlock the door without interaction with householder. Token should have the expiration date and time which could be set up by householder when the token is creating;
5) door should be automatically closed after the timeout period has ended;
6) householder should be able to check the current state of the door (locked/unlocked) and change it manually on demand using mobile application;
7) all events happened in the system should be logged by centralized logging system and stored in remote storage system.

1.3 Analysis of existing software capabilities

Home automation and the Internet of Things is new modern direction in software development that grows very fast nowadays. Many of modern companies specialized on the electronics and internet technologies has their own projects in this area. It is a good practice to analyze existing solution before the new system design and implementation.

Among the home automation systems and “Smart home” systems there are a lot of examples of competitive solutions. Let’s overview the most relevant of them.

Xiaomi MI Smart Home Suite (Fig. 1.1.)

This system represents a decentralized home automations system. It consists of a main device called Gateway which provides the entry point of the system and allows user to interact with other components of the system using special mobile application via internet or using wireless technologies locally.

Figure 1.1 — Xiaomi MI Smart Home Suite

The functionality of this system is determined by the system composition. Each device or sensor can be set up to the system (through the Gateway device) independently using the wireless technologies depending on the user requirements for the system itself. Some of devices are included to the initial set of the system, and some could be bought separately. The main advantage of this system is very simple set up process. Everything that is needed to configure the system is to install the mobile application, enable power on the Gateway device, pair required sensor and devices with Gateway and system is ready for usage.

But despite all the advantages of this system it does not provide the functionality that developed system provides.

Ring Wi-Fi Enabled Video Doorbell (Fig. 1.2)

This system of home automation provides two-way audio communication, motion sensing, cloud recording etc. But as the previous system it does not cover the main functionality the developed system provides.

Figure 1.2 — Ring Wi-Fi Enabled Video Doorbell

Saful Smart Home Apartment Wireless Wifi Video Door Phone Intercom System (Fig. 1.3.)

This is the most similar home automation system to the developed system. It covers the most of functionality required from the developed system – it supports Wifi, Android & IOS applications, real time video and audio chat, movement detecting, unlocking. But it is not enough secure. First of all this system is centralized and this fact reduces system tolerance, stability and availability.

Figure 1.3 — Saful Smart Home Apartment Wireless Wifi Video Door Phone Intercom System

After exiting solutions overview and analysis next the most reasonable question is what software infrastructure and software technology stack is better to use to implement home automation system. From the requirements described in the technical task it is clear that cloud technologies should be used. After the analysis of the cloud services market Microsoft Azure Cloud was chosen because it is one of the most convenient and functional cloud service. Also Microsoft Azure provides the set of services especially for the Internet of Things and Home Automation systems development called Azure IoT Suite. This suite provides special services that allows home automation system developers perform next actions:

1) establish bi-directional communication with billions of IoT devices;
2) authenticate per device for security-enhanced IoT solutions;
3) register devices at scale with IoT Hub Device Provisioning Service;
4) manage your IoT devices at scale with device management;
5) extend the power of the cloud to your edge device.

If the Microsoft Azure Cloud is chosen as the software infrastructure and environment provider it is better to use full Microsoft software development stack. For embedded development Microsoft also provides a special operating system Microsoft Windows IoT Core which could be installed to the several types of single board computers like Raspberry Pi 3 or MinnowBoard MAX. This operation system was designed specially for IoT development and has all requirement capabilities for this. It can run a Universal Windows Platform applications developed using Microsoft .Net technologies. And the main point that this software development technology stack has native support of Azure services after special SDK has installed.

1.4 Designing tasks set up

Automatic control system “Smart intercom” is a complex distributed software system that requires very wide spectrum of details that should be taken into account during its design.

After detailed analysis of the technical task, existing solutions and software capabilities there is an appropriate time to formulate the main goals and tasks that should be done in this paper.

When the requirements are defined, the main thing should be implemented is the functional scheme of the system. This scheme should reflect the main modules of the system and relations (signals) between them. All requirements should be considered while the scheme is creating.

After the functional scheme is created the software capabilities should be chosen, specific applications designed taking into account their features, protocols of communication between them should be defined.

When the software is designed, there is turn to choose the hardware for the automatic control system implementation. It is required to do this step before the software implementation to take into account all capabilities, features and issues introduced by selected devices.

To make developed automatic control system resistant to different kinds of disturbances, stable, high available cloud technologies should be used as the environment for server software deployment. It will reduce the costs for supporting such system and will increase the convenience of automatic control system usage at all.

The final stage of this work presents the implementation of the prototype of the automatic control system itself. It should be the special stand that represents the main functionality of the system. This prototype should demonstrate that the automatic control system works as expected and provides all declared functionality.

When the automatic control system prototype is ready there is time to test it. Different kinds of test suites should be developed to cover the system with tests. This work is intended to verify the correctness of the automatic control system work and to evaluate the system stability and stability stocks.

Main tasks which should be done in this paper:

1) requirements analysis should be performed;
2) existing solutions and capabilities should be analyzed;
3) functional scheme of the automatic control system should be created according to the requirements;
4) software infrastructure should be chosen for the automatic control system software implementation;
5) hardware infrastructure should be chosen for the automatic control system software implementation;
6) the automatic control system software should be designed and implemented;
7) the automatic control system prototype should be created and tested

8) the automatic control system prototype correctness of work should be verified and corresponding conclusions should be made.

2 PROBLEM INVESTIGATION AND MATHEMATICAL MODELS SYNTHESIS

2.1 Definition and justification of the system functional scheme

Functional diagram is a scheme explaining the processes taking place in the individual functional circuits of the product or the product at all. The functional diagram is an explication of certain types of processes that occur in integral functional blocks and circuits of the device. To develop the functional scheme correctly it is required to understand what actions system will perform and what processes will occur inside it. Knowing the requirements declared in technical task it becomes clear that the system will consist of two main subsystems – server side part (also called as backend) and the user portal part which will be represented as mobile phone application and the Web API for interaction with server side.

The goal of this paper is to design and implement the server side of developed system. As required, the whole system will be distributed, so designing the server side part this fact should be taken into account.

According to the previously mentioned condition the backend subsystem should consist of two parts – the device that will be placed directly in the control building and will be some kind of gateway device and the cloud service that will be running in the cloud and will take responsibility for data processing and system management.

The gateway device should be able to process and transfer data from sensor screen and video camera and also should be able to control and manage the state of the door locker. This device should be protected from physical vulnerabilities and secured from programing threats. And the main requirement is that this device should be able to work independently from other parts of the system, even when other parts of the automatic control system are broken or unavailable (e.g. when there is no internet connection).

When required data is received from sensor devices it should be transferred to the cloud service. This service should be designed as an asynchronous stateless application that can be automatically scaled depending on the network load.

The communication between these parts of subsystem should be performed using special service called IoT Hub. This service is intended to provide two-way communication between cloud hosted service and IoT devices without directly configured IP addresses of the IoT Devices. Each IoT Device uses the special connection string with configured address to the IoT hub, listens the incoming messages from it, and sends messages to it when needed.

When the visitor makes a request to ask the householder to open the door (pressing the special button on the touch screen) this signal comes to the gateway device, gateway device should transfer it to the cloud service using the IoT Hub. Then Cloud service should notify the householder with incoming requests. So developed system should contain the special communication channel between Cloud service and mobile application. Such service is called Notification Hub and is intended to send one way signals from the backend to the mobile phone. When the mobile phone receives the request signal, depending on the householder decision, it could request additional data, like video and audio streams, send the signal to open the door or just send reject to incoming request. This actions should be performed from the mobile application through the special interface application called Web API. This application provides public interface for operating with data managed by the cloud service.

The whole system should log every action happened inside, also statistics data should be collected so the cloud service should store required information in the remote database. Web API should provide the interface to retrieve this data from a mobile application.

The development of mobile application and its support services like Notification Hub and Web API is out of scope of this work, so their design will be skipped in this paper.

Depending on the primary description of the automatic control system “Smart intercom” its functional scheme will take the next form (Fig. 2.1.):

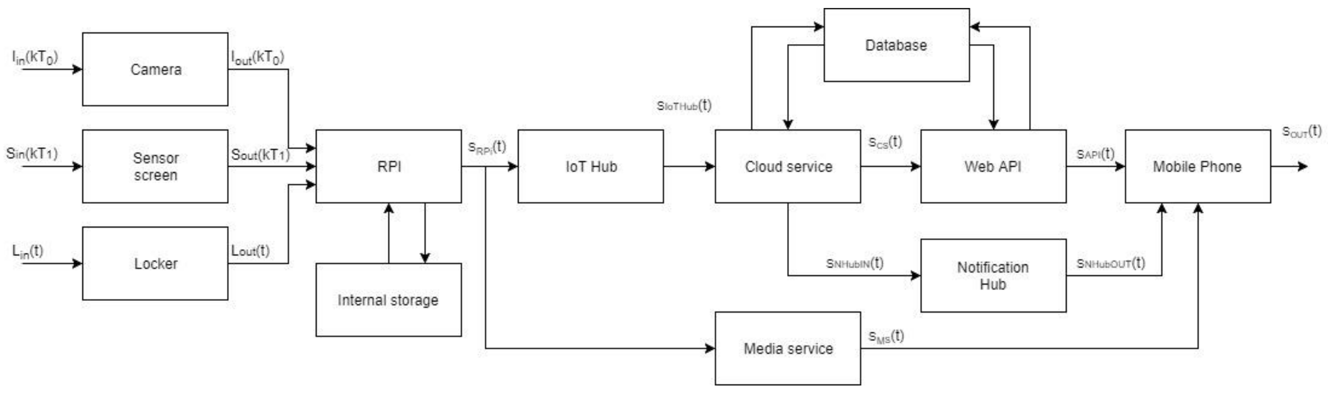


Figure 2.1 — Functional scheme of the automatic control system “Smart intercom”

There are next components on the scheme:

Camera – is a sensor that provides a video and audio stream;

Sensor screen – device, intended to provide the interface for the visitor interaction with the system, converts visitor actions to the electric signal;

Locker – device that manages the state of the door locker;

RPI – a gateway device, represented by a single board computer;

Internal storage – device intended to store the internal configuration of the gateway device and to store temporary information until it will be transferred to the cloud service;

IoT Hub – service that provided a communication channel between the gateway device and cloud service;

Cloud service – service intended to manage and the data received from the gateway device;

Media service – service intended to transfer the video and audio streams from gateway device to the mobile application;

Notification Hub – service intended to provide one way communication between cloud service and mobile application;

Web API – application that provides a public interface for interaction with the system from the mobile application;

Database – storage engine for storing logs and statistics data;

Mobile Phone – smartphone with a developed mobile application that provides a user friendly interface that allows householder to interact with the developed system.

There are next signals on the scheme:

$I_{IN}(kT_0)$ — input signal of the camera, represented by the intensity changes and audio vibrations;

$S_{IN}(kT_1)$ — input signal of the sensor screen represented by the visitors touches;

$L_{IN}(t)$ — input signal if the door locker state;

$I_{OUT}(t)$ — digital output signal from camera;

$S_{OUT}(t)$ — digital output signal from sensor screen;

$L_{OUT}(t)$ — digital output signal from locker;

$S_{RPI}(t)$ – output signal from the gateway device represented by a request over http protocol;

$S_{IoTHub}(t)$ – output signal from the IoT Hub represented by a request over http protocol;

$S_{cs}(t)$ – output signal from the cloud service represented by a request over http protocol;

$S_{NHubIN}(t)$ – output signal from the cloud service to Notification Hub represented by a request over http protocol;

$S_{NHubOUT}(t)$ – output signal from the Notification Hub to mobile application represented by a request over http protocol;

$S_{MS}(t)$ – output signal from Media Service to mobile application performed over UDP protocol;

$S_{API}(t)$ – output signal from the Web API to mobile application represented by a request over http protocol;

$S_{OUT}(t)$ – information displayed to the householder in mobile application.

2.2 Mathematical models

“Smart intercom” is a complex distributed automatic control system that requires a detailed design before the implementation. As a modern software system it should be designed according to a common software development principals and approaches.

The most important design principals in software development are Object Oriented Design principle and SOLID principle. This principles allows software developers to design complex system correctly and make it maintainable in the future.

In case of developed system the most reasonable will be design according to the SOLID principle, because this principle has already included the main ideas of OOD. In computer programming, the term SOLID is a mnemonic acronym for five design principles intended to make software designs more understandable, flexible and maintainable. The principles are:

1) The Single Responsibility Principle

There should never be more than one reason for a class to change. Basically, this means that classes should exist for one purpose only. Responsibility is the heart of this principle, so to rephrase there should never be more than one responsibility per class.

2) The Open Closed Principle

Software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification. Developer can make an object behave differently without modifying it by using abstractions, or by placing behavior (responsibility) in derivative classes. In other words, by creating base classes with override-able functions.

3) The Liskov Substitution Principle

Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it. In other words, if developer is calling a method defined at a base class upon an abstracted class, the function must be implemented properly on the subtype class. Or, when using an object through its base class interface, the derived object must not expect such users to obey preconditions that are stronger than those required by the base class.

4) The Dependency Inversion Principle

Depend on abstractions, not on concretions or High level modules should not depend upon low level modules. Both should depend upon abstractions. Abstractions should not depend upon details. Details should depend upon abstractions. This is very closely related to the open closed principle. By passing dependencies to classes as abstractions, developer remove the need to program dependency specific.

5) The Interface Segregation Principle

Clients should not be forced to depend upon interfaces that they do not use. In other words, when a client depends upon a class that contains interfaces that the client does not use, but that other clients do use, then that client will be affected by the changes that those other clients force upon the class. This principle sounds like the inheritance specific single responsibility principle.

According to principals described above the first thing that should be done during the system design is defining the contracts which will be used for communication between modules of the system. In programming the mathematical representation of the module is the interface it provides.

According to the functional scheme and requirements description the interfaces for each module of the system are:

1) Camera module

The main responsibility of this module is to provide the video and audio stream as a binary stream so its interface will be very simple:

```
public interface ICamera
{
       void Initialize();
       Stream GetVideoAndAudioStream();
 }
```

2) Sensor screen module

This module is responsible for rising the event with arguments when the user presses the controls on the screen. Its interface is:

```
public interface ISensorScreen
{
       event EventHandler<SensorScreenEventArgs> OnInputDataReceived;
}
```

According to this interface the aggregate module (in our case Gateway device) should subscribe on provided event and implement an event handler method that will be able to process *SensorScreenEventArgs* object with input data user passed.

3) Locker module

This module is responsible for 3 actions – to unlock the door, to lock the door and to verify door’s state, so the interface will be next:

```
public interface ILocker
{
       void Lock();
       void Unlock();
       DoorState GetDoorState();
}
```

DoorState object is an enumeration that can has several states: “Locked”, “Unlocked”, “Unavailable”.

4) Gateway module

Gateway module is intended to perform recurring actions for processing data from different sources and transferring this data to the cloud. Its interface should contain operations for primary initialization of aggregated devices and for starting the processing loop.

```
public interface IGateway
{
    void Initialize();
    void Start();
}
```

5) IoT Hub module

IoT Hub module should provide operation for sending the message to the cloud and also an event for receiving messages from the cloud service in asynchronous way in its interface:

```
public interface IIotHubClient
{
    void SentMessage(IotHubMessage message);
    event EventHandler<IotHubMessageEventArgs> OnMessageReceived;
}
```

According to this interface parent module that uses this client should subscribe on provided event and implement an event handler method that will be able to process *IotHubMessageEventArgs* object which contains the incoming message.

6) Cloud service module

The public interface of this module is represented by a Web API module which is a wrapper around the cloud service itself and its interface design is out of scope of this paper and will be described in other.

7) Media service module

This module is intended to transfer the stream captured by gateway from camera module to the mobile application. So the main operations it should provide are operations to open stream and to close the stream by stream identifier.

```
public interface IMediaService
{
        Stream OpenStream(Guid id);
        bool CloseStream(Guid id);
}
```

Guid is a global identifier that is a 128-bit number used to identify information in computer systems. It uses a special algorithms to generate random sequence of bytes to create a globally unique identifier in a distributed system and in the world at all.

8) Repository module

This module will provide the general interface for CRUD operations on all modules of developed system that works with data storing engines. It does not make sense what kind of storage is used – this module should implement the general interface which has operations for retrieving data from storage, for inserting data to storage, for updating existing data and for deleting specific data from storage. The interface will be next:

```
public interface IRepository<TEntity>
{
    TEntity[] GetAll();
    TEntity Get(Guid id);
    TEntity Get(Expression<Func<bool, TEntity>> filter);
    TEntity GetFirst(Expression<Func<bool, TEntity>> filter);
    TEntity Create(TEntity entity);
    TEntity Update(TEntity entity);
    void Delete(TEntity entity);
}
```

Here TEntity is a generic type of the entities stored in database, Expression<Func<bool, TEntity>> is a type of the predicate which will be used to filter the result set of data on the storing engine side, so the client using the implementation of this interface will receive the filtered data passing the filter represented by a lambda expression to the corresponding method.

When the contracts are defined it is required to design each module dependent on the provided interfaces but not on the specific implementation of these interfaces in case when one module is dependent on another. To resolve which implementation for each interface is created there is special tools called IoC (Inversion-of-control) containers where all dependencies should be registered. Using such containers gives possibility to change the implementation of a specific interface without making changes to the code base, and it makes supporting of developed system more convenient and easier.

2.3 Evaluation of simulation results

Developed system is a complex distributed mechanism designed accordingly to the innovative principles of software design. It consists of big amount of different modules and each module designed as an independent unit. Connections between modules are build using abstractions. Before modules implementation there was designed the set if interfaces that provide the contracts for communication between modules. Each module implements his own interface and if this module needs functionality of some other module it also depends on his interface, on abstraction but not on the specific implementation of the module. This approach helps to increase flexibility of the developed system and to simplify the system support. Also when developer needs to interchange one module by another there is no necessity to change modules dependent on this module. It is enough to develop the new module that implements corresponding interface and register new module instead of old one.

If modules don’t depend on other modules but depend on their interfaces there should be mechanism that substitutes required module instead if his interface. This mechanism is called dependency injection. Application that uses such principle has so called dependency container where developer registers mappings between interfaces and modules that implement them.

When application requires some module it asks dependency container to resolve all his dependencies and after this container will return the module. If there is no required module in container there will be error raised.

This behavior can be used as verification of correctly designed architecture. If no one module does not depend on any other and container can resolve each dependency – it means that system was designed, developed and configured correctly.

3 DESIGN OF AN AUTOMATIC CONTROL SYSTEM “SMART INTERCOM”

3.1 Models implementation algorithms

The goal of this paragraph is to determine the main algorithms of processing data in the developed system. The first stage in this complex procedure is to identify how the system works and what data flows are present in this automatic control system. And to resolve this issue the set of special diagrams was created – sequence diagrams. A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart.

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

The main flow of the system is described by the sequence diagram shown on the figure 3.1.

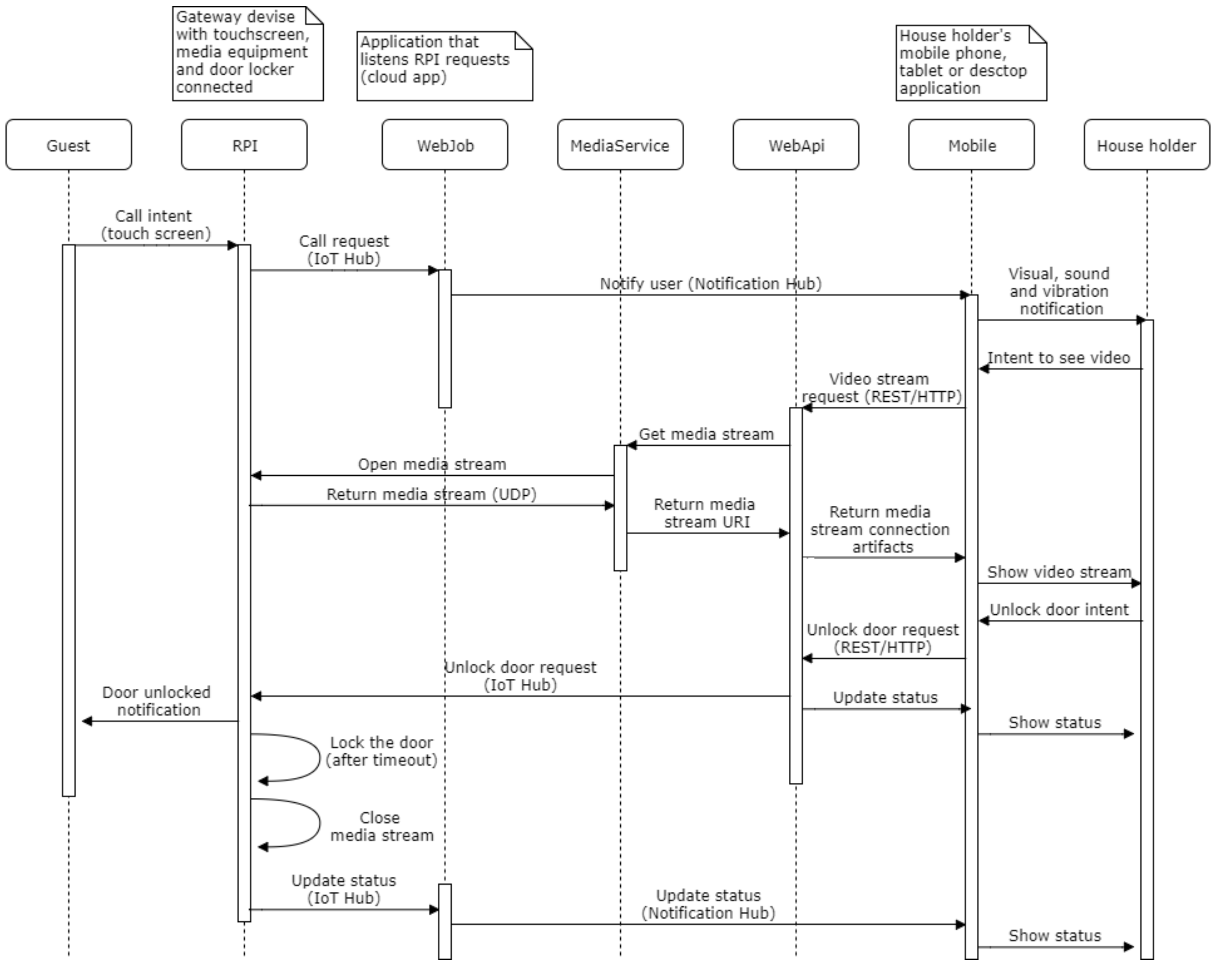

Gateway devise with touchscreen, media equipment and door locker connected
Application that listens RPI requests (cloud app)
House holder's mobile phone, tablet or desctop application
Guest
RPI
WebJob
MediaService
WebApi
Mobile
House holder
Call intent (touch screen)
Call request (IoT Hub)
Notify user (Notification Hub)
Visual, sound and vibration notification
Intent to see video
Video stream request (REST/HTTP)
Get media stream
Open media stream
Return media stream (UDP)
Return media stream URI
Return media stream connection artifacts
Show video stream
Unlock door intent
Unlock door request (REST/HTTP)
Unlock door request (IoT Hub)
Door unlocked notification
Update status
Show status
Lock the door (after timeout)
Close media stream
Update status (IoT Hub)
Update status (Notification Hub)
Show status

Figure 3.1 – The sequence diagram of the automatic control system “Smart intercom” main workflow

According to this workflow there are next events occurred in the system.

Imagine that guest comes to the door and wants to come in. He sees the locked door and the terminal of developed system represented by sensor screen with special graphical user interface. He presses the “Call” screen and the system navigates him to the next screen where guest can select the flat number he wants visit and presses the button “Call” once again. These actions are represented as “Call intent” on the diagram.

The application hosted on Raspberry PI device handles these actions from user, extracts the flat number and verifies it and if data is correct it puts the corresponding message to the IoT Hub. IoT Hub is responsible for this message to be delivered to the Web job application that has the special handler for this actions. This part of functionality of the developed system called “Call request” on the diagram.

The Web job handler retrieves this message from IoT Hub, validates it and tries to find the metadata associated with the house holder’s mobile devices. If metadata was not found the Web job returns the error message to the IoT Hub and Guest sees the error message that call is not possible and he should request another way to be authorized. Id metadata was successfully fetched the Web job creates the notification message and sends it to the notification hub for all devices associated with householder account. These events are represented by a “Notify user” action on the diagram.

Mobile application receives this message and rises the visual, sound or vibration effects depending on configuration set up by user in application settings. When user has reacted to this notification he has three option – to reject the request to open the door, to open the door and to see who has come to his home and only then to make the final decision. In the third case he presses the button “See video” on the mobile device screen and this action is represented by “Video stream request” on the diagram. The mobile application makes a REST/HTTP request to the Web API application.

The Web API application retrieves the special metadata required to request the media stream from Media service and tries to connect to this service (“Get media stream” action on diagram).

The media stream requests Raspberry PI to configure the Camera device, to open the UDP socket and to start content streaming (“Return media stream”). When the stream is opened, Media service sets up the special endpoint for content streaming and returns the URI of this endpoint to Web API so mobile applications can connect to this stream and play the video/audio stream. These action marked as “Return media stream URI” on the sequence diagram.

When the Web API receives the media stream endpoint URI, it prepares the web response to previously received request, fills the response body as a JSON object that contains media stream endpoint URI and returns it back to mobile application web client ("Return media stream connection artifacts" on the diagram).

Mobile application receives these artifacts and tries to connect to this endpoint. If nothing bad happened it starts to play the media content.

At this stage the house holder has possibility to see the visitor and to communicate with him. Now he has two options. First one is to reject the request to come in from visitor. It means that system breaks all connections and shows the visitor the corresponding message that house holder has rejected his intent. This scenario is out of scope of the diagram and is not shown.

In the other case user decides to pass the guest. He presses the button "Unlock" on the screen of his device. At this moment mobile application using a HTTP client makes request to the Web API with data for door unlocking ("Unlock door request" on the sequence diagram). When message is sent this application generates HTTP response that indicates all operations completed successfully and according to this response mobile application updates its status to show house holder that door is unlocked.

When the Web API application receives request for door unlocking it creates the corresponding message and puts it to the IoT Hub. Application hosted on the Raspberry PI device has a special handler for suck messages. Receiving this message it exposes next activities: it generates the signal to the locker to be unlocked, it shows corresponding message on the screen to notify guest that he is able to come in and finally it sets up the lock door timeout.

When the timeout has ended Raspberry PI application closes the media stream, locks the door and puts the corresponding message to the IoT Hub. After this, the Web job application notifies mobile application via notification hub that door has been closed and corresponding controls on the mobile phone screen are being updated.

This is the main positive scenario of the automation control system work, but not the only.

There is a scenario when the house holder wants to unlock the door for himself and he can do it using the same button on his mobile application screen like for guest but without previewing video stream from “Smart intercom”.

Another scenario is when the internet connection was lost by Raspberry PI. For such cases there was introduced the unlock method using QR code token (fig. 3.2).

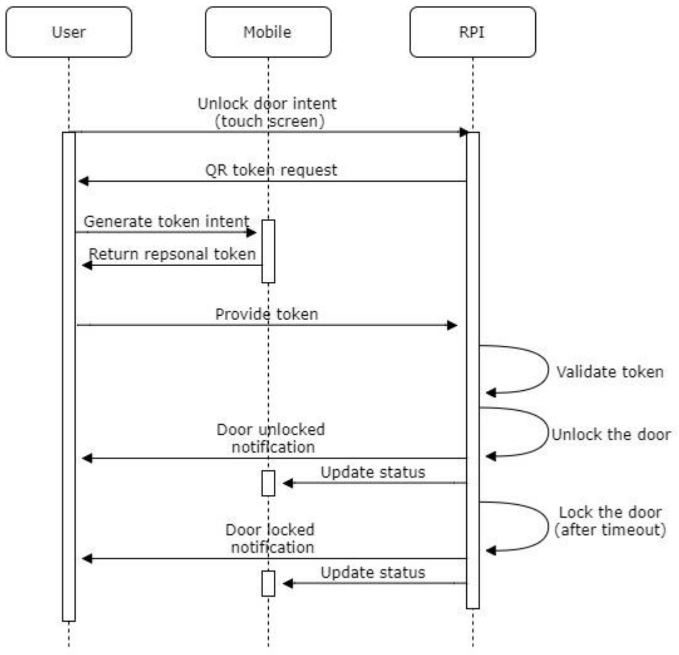


Figure 3.2 – The sequence diagram of the automatic control system “Smart intercom” main workflow

The situation is next: house holder comes to the door and wants to unlock it. But unfortunately the internet connection was lost by the host application and there is no way to do it over internet. Then user presses the “Token” button on the system terminal ("Unlock door intent" on the diagram) and the system asks him to provide token ("QR token request" on the diagram). He generates this token (QR barcode) using corresponding tab in his mobile application and shows it to the terminal camera ("Provide token" on the diagram). Host application scans for some period of time the video stream from the terminal camera trying to detect the token barcode. If barcode was not detected system shows error notification on the screen with suggestion to retry. If barcode was detected the host application tries to parse it and to retrieve content from it. The content is represented by an encrypted byte array encoded to Base 64 string. The next step is to convert this string to byte array. When the bytes are ready system using the specially developed package tries to decrypt it. If token was not interchanged by malefactor system receives useful byte array with information about user identity and token expiration date and time. This information is represented by special object serialized to JSON string. So after decryption system tries to convert byte array to ASCII string and to deserialize the string to object. Token information is ready on this stage.

There is a time to validate the token. The first stage of token validation is to validate user identity. If provided identity was not found in the system database – system rejects the unlock request. If identity is valid – system verifies the expiry data of token. If it detects that user provided expired token – reject action will be applied.

In case of valid token the system will open the door, will notify the user and mobile application about opened door and will set up door locking timeout. Now user is able to come in.

When the timeout has ended system locks the door and notifies the mobile application about status changes and resets its terminal interface to defaults. This is the end of this scenario.

QR barcode is very useful not only for scenarios with unstable internet connection. Imagine the situation when householder receives some guests who may live in his apartments for several days or more. It is very inconvenient to pass them each time using mobile application via internet. This is an appropriate moment to use QR barcode tokens once again. The main idea is to generate such tokens which will be alive for the period guests live in householder's apartments and expires after they have gone. This authorization method is very close to the previous except the token generation logic, where token expiration date and time are set up by householder, but this functionality relates to mobile application development and is out of scope of this paper. It will be explained the other paper where mobile application development will be presented. Figure 3.2 is also applicable for this scenario because the token validation process remains the same.

The final authorization scenario is the reserve way to be authorized. It is applicable when the householder lost his mobile phone and is not able to use previous methods. The main idea of this method is to use admin password to unlock the door. Each householder receives its own password on the initial set up of the system. To be authorized user must press the "Password" button on the terminal screen and on the next view he must provide the flat number and his password. If this pair of values is validated successfully user will see the corresponding notification and the door will be unlocked by a system. If the pair of provided values did not match – system will reject the attempt to unlock the door. If number of unsuccessful attempts exceeds the predefined limit system will block this authorization method for this flat number for configured period of time. This authorization method is reserve and is not recommended to use when other are possible. It is recommended for use only for emergency scenarios.

3.2 Means of implementation

When software application design is ready for development there is appropriate time to decide what hardware will be used to run these applications. Developed system consists of next applications:

1) Terminal application – should be hosted on the single board computer. This application requires hardware that could not be provided by computer itself but selected device should have corresponding connectivity to be able to use these devices. They are video camera, touch screen, microphone, speaker and electro locker. Also selected single board computer should be able to work under control of Microsoft Windows 10 IoT Core.
2) IoT Hub – cloud application, should be hosted on the cloud virtual machine and should be able to auto scale.
3) Cloud service – cloud application, should be hosted on the cloud virtual machine and should be able to auto scale. Cloud should guarantee the hundred percentage availability of this service.
4) Media service – third party service, should be hosted on the owner's machines or in cloud if possible.
5) Database – no SQL storage, hosted in the cloud and able to auto scale.
6) Notification Hub – cloud application, should be hosted on the cloud virtual machine and should be able to auto scale.
7) Web API – cloud application, it is a stateless HTTP application developed using ASP .Net Web API technology, should be hosted as a cloud website.
8) Mobile application – should be developed for three the most popular operation systems – iOS, Android and Windows 10.

Notification Hub, Web API and mobile application are out of scope of this paper and will not be covered.

First step is to choose correct single board computer – the heart of developed system. A single-board computer (SBC) is a complete computer built on a single circuit board, with microprocessor, memory, input/output and other features required of a functional computer. Unlike a desktop personal computer, single board computers often did not rely on expansion slots for peripheral functions or expansion. Some single-board computers are made to plug into a backplane for system expansion. Single board computers have been built using a wide range of microprocessors. Simple designs, such as built by computer hobbyists, often use static RAM and low-cost 8 or 16 bit processors.

Single board computers were made possible by increasing density of integrated circuits. A single-board configuration reduces a system's overall cost, by reducing the number of circuit boards required, and by eliminating connectors and bus driver circuits that would otherwise be used. By putting all the functions on one board, a smaller overall system can be obtained, for example, as in notebook computers. Connectors are a frequent source of reliability problems, so a single-board system eliminates these problems.

The main requirement for single board computer in developed automatic system is to be able to work under Windows 10 IoT Core. According to the operation system documentation it supports next single board computers: Raspberry Pi 2, Raspberry Pi 3, MinnowBoard Max, DragonBoard 410c (Fig. 3.3). Table 3.1 demonstrates the comparison of these devices. It was decided to use the Raspberry Pi 3 device because it has enough computing power provided by quad core ARM processor, a powerful video chip and finally very wide connectivity range. There are enough amount of USB ports to connect required USB devices (video camera, touch screen required by system and keyboard, mouse used for device configuration). Also it has optimal price for provided functionality.

Table 3.1 — Single board computers comparison

	Raspberry Pi 2	Raspberry Pi 3	MinnowBoard Max	DragonBoard 410c
CPU	900MHz Quad-Core ARM Cortex A7	1.2GHz Quad-Core ARM Cortex A53	1.3GHz x86/x64	1.2GHz Quad-Core ARM Cortex A53
Memory	1GB	1GB	2 GB	1GB
GPU	Broadcom Video Core IV @ 250MHz (no DirectX or Hardware Acceleration support)	Broadcom Video Core IV @ 400MHz (no DirectX or Hardware Acceleration support)	Intel HD Graphics	Qualcomm Adreno 306 @ 400MHz (only 720p / 1280 x 720 supported)
USB	4x USB 2.0	4x USB 2.0	1x USB 2.0, 1x USB 3.0	2x USB 2.0
Network	10/100/1000 MBit/s Ethernet	Wi-Fi 802.11 b/g/n Ethernet, Bluetooth 4.1	10/100/1000 MBit/s Ethernet	Wi-Fi 802.11 a/b/g/n, Bluetooth 4.1
Video Output	HDMI, DSI	HDMI, DSI	Micro HDMI	HDMI, DSI
Audio Output	Analog via 3.5 mm jack	Analog via 3.5 mm jack	Digital via HDMI	Digital via HDMI
Peripheral	24x GPIO pins 1x Serial UART 2x SPI bus 1x I2C bus	24x GPIO pins 1x SerialUART 2x SPI bus 1x I2C bus	10x GPIO pins 2x Serial UARTs 1x SPI bus 1x I2C bus	11x GPIO pins 2x Serial UARTs 1x SPI bus 2x I2C bus

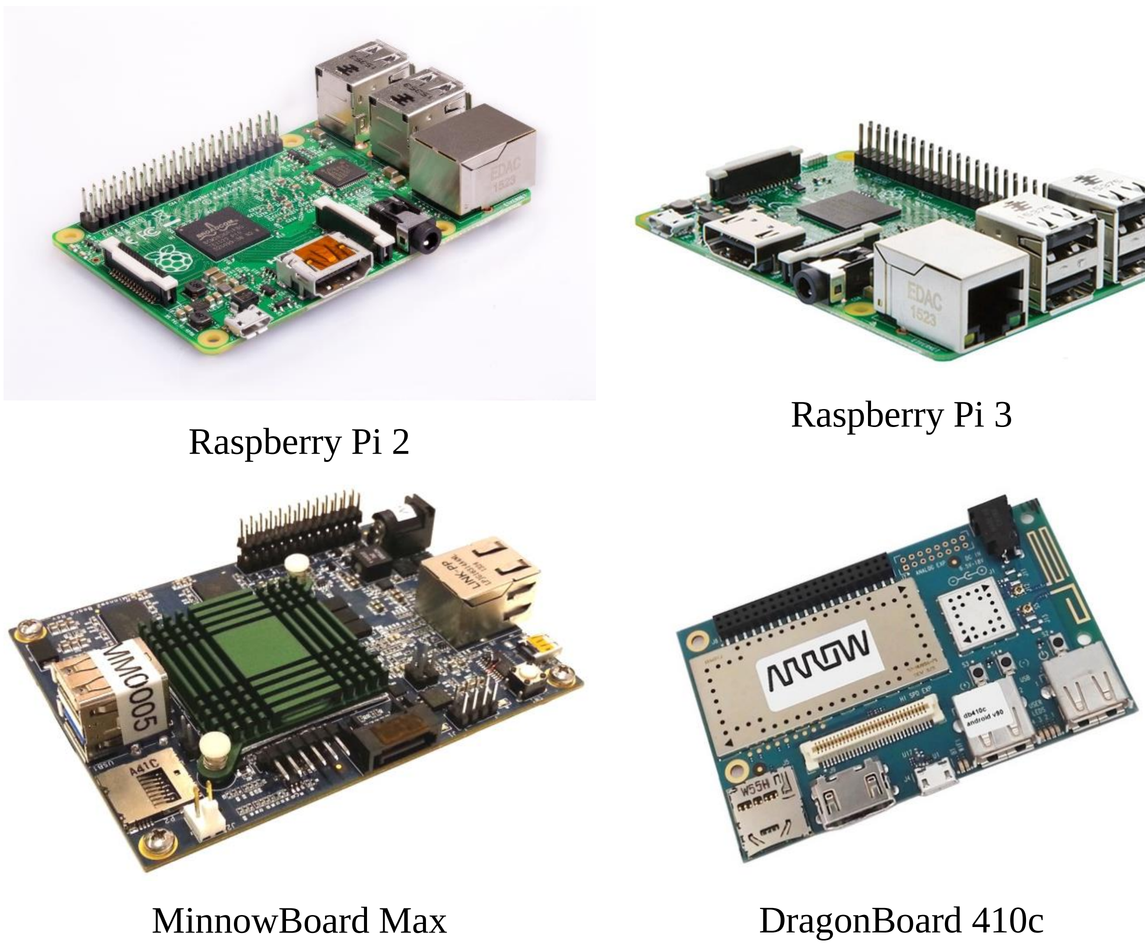


Figure 3.3 – The compared single board computers

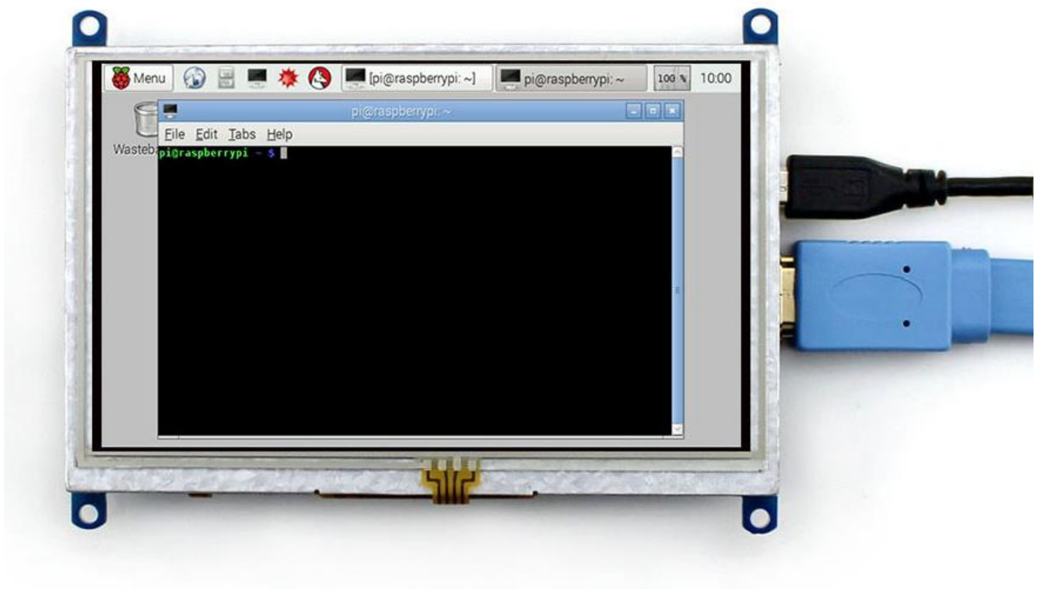
Now it is an appropriate time to choose the support devices for Raspberry Pi 3. They are video camera, microphone and speaker. Nowadays there are a lot of USB cameras are provided with built in microphone, so to simplify the situation one of such cameras will be used in developed automation control system. After market analysis it was decided to use the Microsoft Lifecam HD-3000 (Fig. 3.4), the USB camera with next characteristics:

1) Device Type – web camera
2) Connectivity Technology – Wired
3) Max Digital Video Resolution – 1280 x 720
4) Computer Interface – USB 2.0
5) Audio Support – Yes

6) Audio Support Features – built-in microphone
7) Connector Type – 4 pin USB Type A

Figure 3.4 – Microsoft Lifecam HD-3000

Next step is to choose the touch screen compatible with previously selected single board computer and Windows 10 IoT Core. After continuous market analysis the choice fell on the Waveshare 5inch HDMI LCD (B) (Fig. 3.5) touch screen and there are several weighted arguments for this choice. First of all this device was specially designed for Raspberry Pi single board computers and completely tested with them. Secondly it natively supported by the Microsoft Windows 10 IoT Core operation system, there is no additional drivers required to use this device with chosen operation system. And finally it has simple connectivity method – there are two cables needed: HDMI for image transition and micro USB for power and sensor support.



```
pi@raspberrypi ~ $
```

Figure 3.4 – Waveshare 5inch HDMI LCD (B)

3.3 Interface and functional modules design

The convenience of usage of any system first of all depends on the interface provided by this system to user for interaction with the system. In case of developed system this interface will be implemented as a graphical user interface with touch screen. System has two different entry points with graphical user interface – mobile application and terminal application. Mobile application interface design is out of scope of this paper and will be described in other one. Graphical user interface for terminal application should be designed for 5 inch touch screen with graphical resolution 800 x 480 pixels with landscape orientation. Best practices of user experience design should be considered during interface design.

The graphical user interface of terminal application of automatic control system “Smart intercom” will consist of four functional pages and four alert pages.

Functional pages are:

1) Main page – page that navigates user to thee possible authorization pages – Call intent page, Token authorization page and Password authorization page. It contains three large buttons “Call”, “Token” and “Password” for corresponding pages.

2) Call intent page – page that allows user to make a call to the house holder specifying his flat number. It contains a “Back” button that navigates user to the Main page, flat number displaying element, numeric keyboard with digit buttons from one to zero, clear button that clears full flat number, delete button that removes the last digit and the “Call” button that performs the call itself.

3) Token authorization page – page that allows user to pass the authorization procedure using access token generated by a mobile application as a QR barcode. It contains a “Back” button that navigates user to the Main page, the video previewing element that shows user the video stream from terminal camera and helps him to focus camera on his token image and a button “Scan” that starts scanning procedure bounded by a timeout.

4) Password authorization page – to pass the authorization procedure using flat number and associated with it administrator’s password. It contains a “Back” button that navigates user to the Main page, flat number displaying element, a password displaying element with masked symbols, numeric keyboard with digit buttons from one to zero, clear button that clears fields, delete button that removes the last digit and the “Submit” button that performs authorization verification.

Alert pages are:

1) Door is opened notification page – alert page that notifies user that authorization procedure completed successfully and he is able to come in, the door is open. User can be navigated here automatically from any of authorization pages in case of successful result.

2) Rejected notification page – alert page that notifies user about the fact that house holder rejected his request to come in.

3) Connection failure notification page – alert page that notifies user that there is a connection issue happened and there is no possibility to call house holder directly, it suggests to use another method of authorization.

4) Timeout notification page – alert page that notifies user that QR barcode scanning timeout has ended and no barcode was detected. It navigates user back to the Token authorization page with suggestion to try once more.

Each alert notification page has the only control on its background – the text ox with corresponding notification message. Also every alert page has its time out period of visibility. When timeout has ended application navigates user to predefined page. In case of Door is opened notification page, Rejected notification page and Connection failure notification page user will be navigated to the Main page, in case of Timeout notification page – to the Token authorization page.

The views of previously described navigation pages of developed system are shown on figures 3.5 – 3.13.

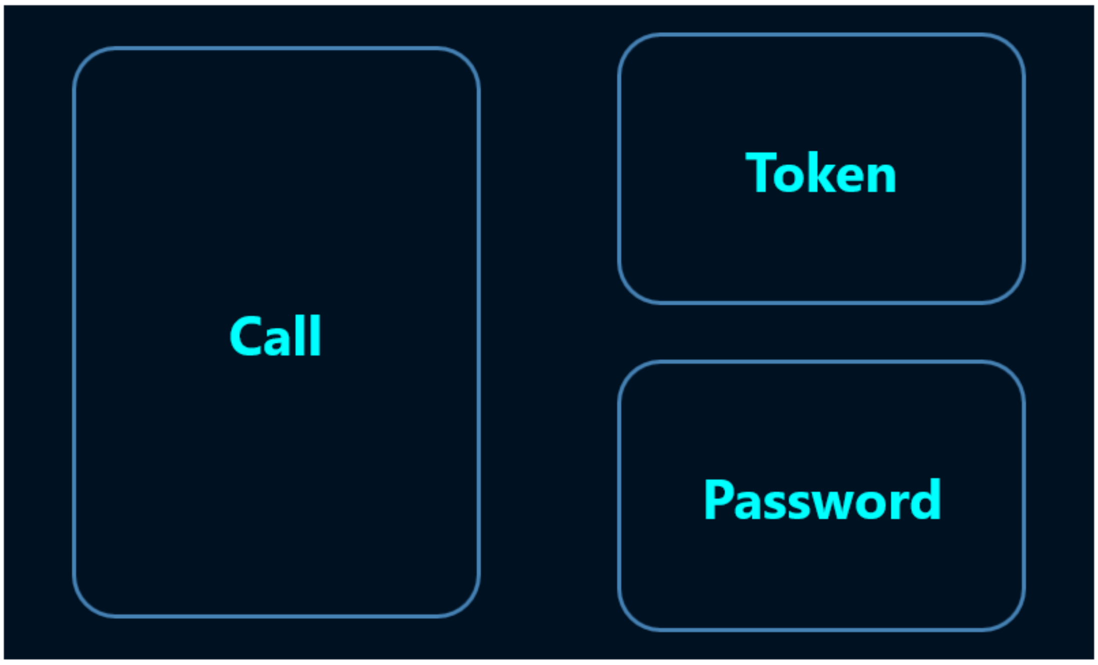


Figure 3.5 – The Main page of the terminal application

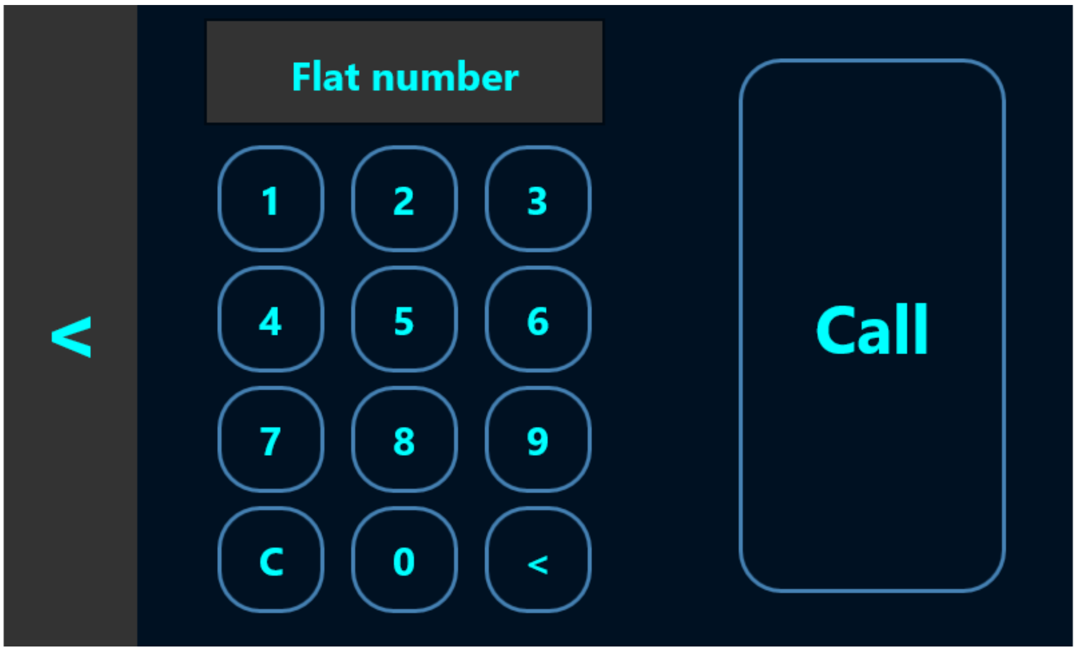


Figure 3.6 – The Call intent page of the terminal application

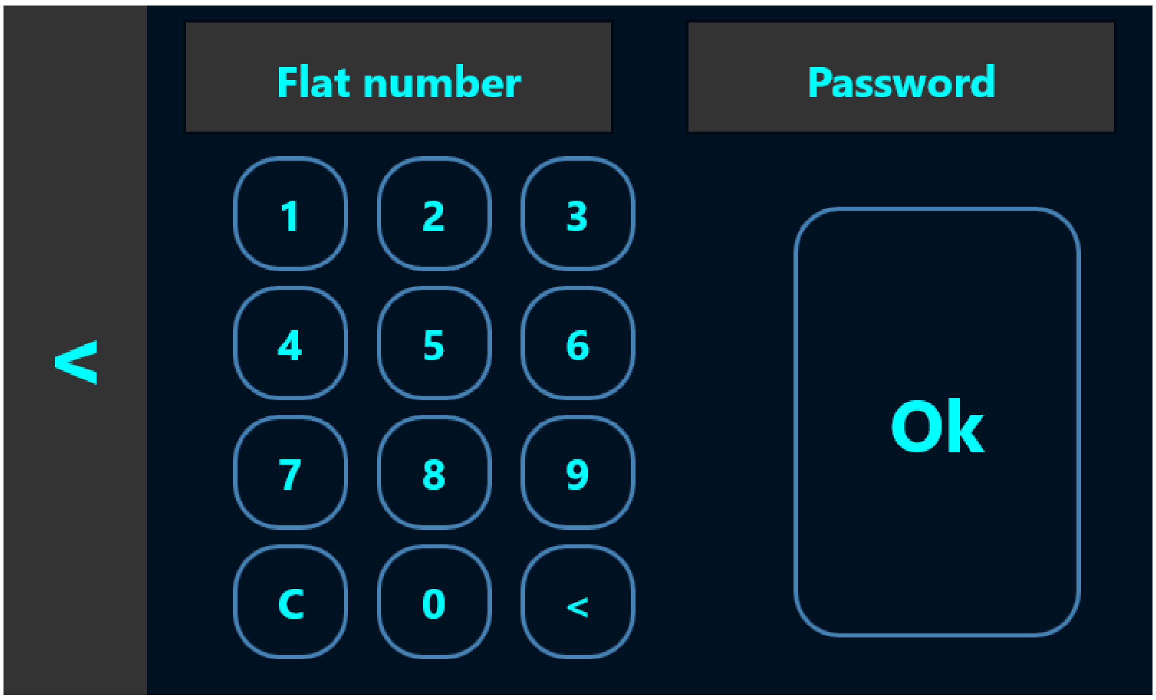


Figure 3.7 – The Password authorization page of the terminal application

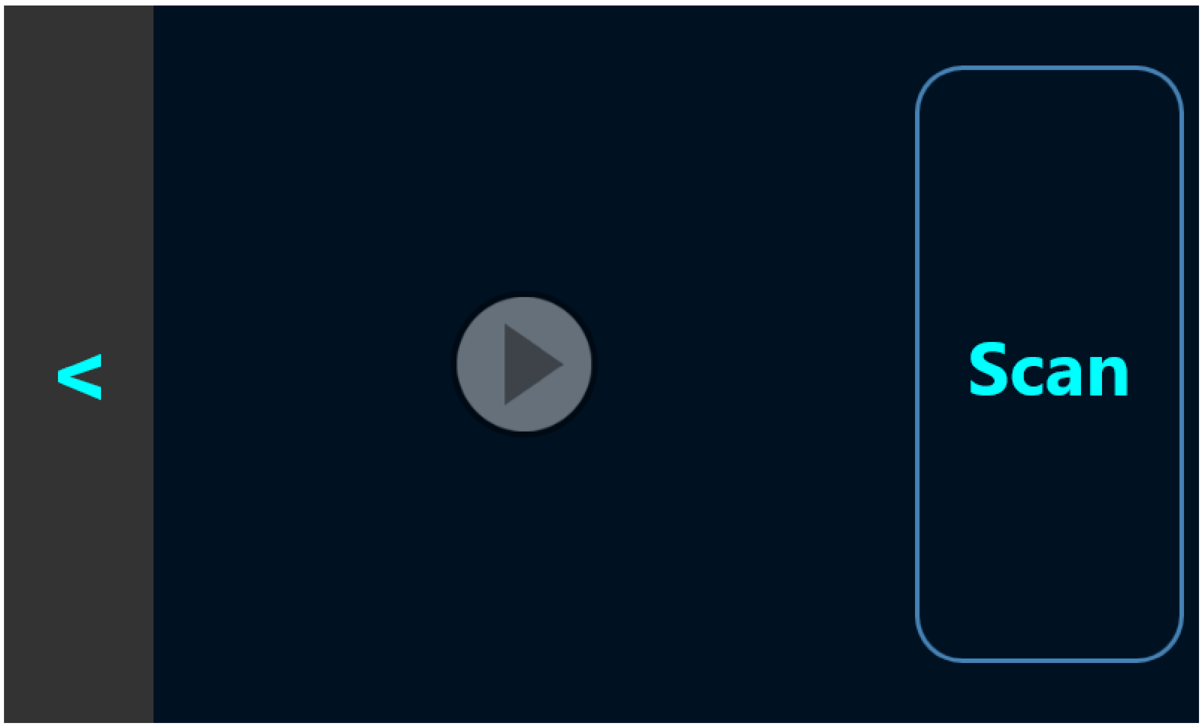
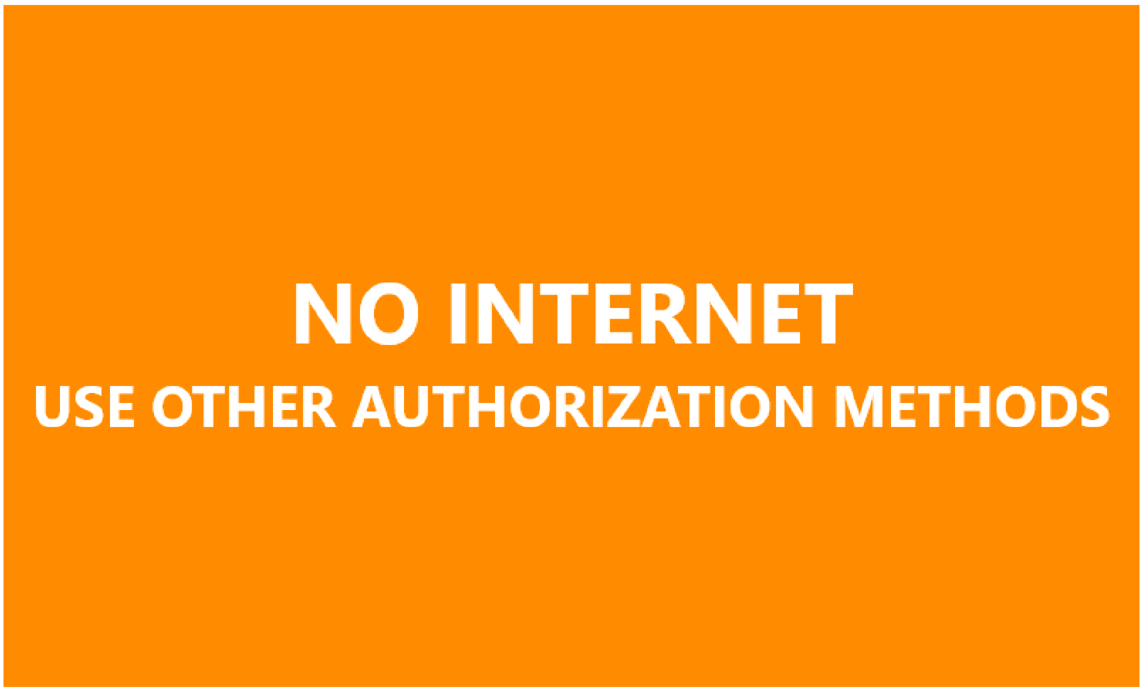


Figure 3.8 – The Token authorization page of the terminal application

Figure 3.9 – The Door is opened notification page of the terminal application

Figure 3.10 – The Door is opened notification page of the terminal application



NO INTERNET

USE OTHER AUTHORIZATION METHODS

Figure 3.11 – The Connection failure notification page of the terminal application

From another side the convenience of the application and the system development itself depends on the approaches and time invested to the system software architectural design. The properly designed application architecture reduces the time spent on the software and system development, reduces costs spent on the system support and simplifies the process of software testing, finalization and future improvements. Also properly designed system structure increases the system stability under the high load.

Developed system designed as a distributed system hosted in the cloud. From the first look it seems to work as the only functional application but in the reality it consists of eleven separate application that works independently of each other. Actually these devices can be divided into four functional groups:

1) Terminal side applications;
2) Backend side applications (server side applications);
3) Mobile side applications;

4) Communication layer applications.

Let’s get down to the functional responsibilities of each group and application they consist of.

The first group is Terminal side applications. This group consists of two applications: terminal UI application and terminal background process, both developed as a Universal Windows Applications. These applications hosted on the Raspberry Pi device and are running continuously during the whole time the system works.

Terminal UI application is the main application that provides user with Graphical User Interface for interacting with the system from terminal side. It handles all user actions performed by touch screen and notifies other system components about performed actions. It is responsible for the direct communication with the system – from terminal to other components.

Terminal background application – is a backward communication channel between system components and terminal. It listens for the messages from the system and handles them on the terminal side performing actions according to the message type and message parameters and arguments. Examples of these actions are opening and closing the media stream, handling the unlock door and lock door requests or reject the visitors intent. Also this application responsible for timeouts handling for lock/unlock operation and alert messages displaying.

Second group is Backend side applications. This group also consists of two applications Web Job and Web API. Both application are Web oriented and hosted in the cloud, they are responsible for bidirectional communication between terminal applications and mobile applications.

Web job is a cloud service that listens for the messages, performs business logic according to these messages, interacts with storage, performs events logging activities and notifies mobile applications according to performed work. It provides directional communication channel between terminal applications and mobile applications.

Web API is a web HTTP application. It is a restful API that provides to mobile applications the interface for interaction with the system. It is responsible for directional communication channel between mobile applications and other system components and also for manipulating with stored data from mobile application.

Mobile applications group consists of three application projects for the most popular mobile phone operation systems: iOS, Android and Universal Windows Platform and the common module for this applications that is used by each of them. This part of the system is out of scope of this paper so it will not be described here.

Last group of applications - Communication layer applications group consists also consists of three applications. These applications don’t perform any business logic work, they are responsible for the messaging between different components of the system. Applications are:

IoT Hub – bidirectional messaging bus that is used for communication of embedded resources like Raspberry Pi computer with corresponding software with server side applications or with the same embedded devices. It is hosted in the cloud as a separate service. Each device registered in this bus by admin can send the message of different types with different payload to this bus and some other application or device can listen for this application and perform some actions on receiving the message.

Notification hub – one directional message bus for communication between web applications and mobile devices. It is also hosted as a standalone web application and requires to register devices before communication set up.

Media service – web application that responsible for media streams flow control. It opens the media stream and sets up the endpoint with required metadata, so other consumers can connection to this endpoint using specially generated URI and subscribe for this stream.

From developers perspective software of the automatic control system “Smart intercom” is developed as a Visual Studio 2017 solution that consists of 12 projects:

1) SmartIntecome.Device.UI – Terminal application with user interface – project of type Universal Windows Platform;
2) SmartIntercom.Device.BackgroundProcess – Terminal application for background data processing - project of type Universal Windows Platform;
3) SmartHome.WebJob – Web job application – project of type Azure Web Job;
4) SmartIntercom.WebAPI – The Web API application – project of type Asp .Net Core 2.0 Web API;
5) SmartIntecom.Database – Project that contains database structure and configuration;
6) SmartIntecom.MediaService – Project of type console application with additional packages;
7) SmartIntecom.IoTHub – Project that contains deployment configuration and settings for IoT Hub application;
8) SmartIntercom.NotificationHub - Project that contains deployment configuration and settings for Notification Hub application;
9) SmartIntercom.Mobile – Project of type Xamarin.Forms application – project that contains shared code between mobile application projects.
10) SmartIntercom.Mobile.Android – Project of type Xamarin application – contains Android specific implementation of mobile application, based on functionality developed in SmartIntercom.Mobile;
11) SmartIntercom.Mobile.iOS – Project of type Xamarin application – contains iOS specific implementation of mobile application, based on functionality developed in SmartIntercom.Mobile;

12) SmartIntercom.Mobile.UWP – Project of type Xamarin application – contains Windows 10 specific implementation of mobile application, based on functionality developed in SmartIntercom.Mobile;

13) SmartIntercom.Common – The project of type Class library, contains shared business logic used in several parts of the application.

4 SYSTEM IMPLEMENTATION

4.1 Software description

Introduction

Software for automatic control system “Smart intercom” and the system itself was designed and developed to introduce the community the new component for home automation systems that allows everyone to be aware of the visitors who has come to their home being outdoor even in any point of our planet where internet connection available. It provides the very simple and convenient mobile application that helps house holder to see, hear and talk to the visitor, and make a decision to allow him to come in or to reject his intent and lock the door.

Software structure and required hardware

From software architectural point of view it is a complex distributed system that consists of eleven separate applications that works together to guarantee the system correct work.

These applications requires the special hardware to be hosted on. Basically applications can be divided on three main groups: terminal applications, cloud applications and mobile applications.

Terminal applications are hosted on the embedded device and placed directly in the user’s building. Raspberry Pi device is chosen as a platform for their hosting. It works under the control of the Window 10 IoT Core operation system. This group of application requires interaction with users, so it has graphical user interface displayed in the touch screen.

Cloud applications was designed to make system distributed logically and geographically with goal to make the system reliable, stable and scalable. Each of these applications hosted on the separate virtual machine in the Microsoft Azure cloud. They are responsible for content sharing between terminal application and mobile devices of each house holder. Only fully configured cloud infrastructure could provide the complete functionality of developed system, so here appears the first and the most benefit of cloud technologies usage. Each cloud application was designed as a stateless application, so when one of them breaks down, cloud guarantees that new copy of application will be deployed and will change the bad one, so the system will continue working.

Finally, the last group is mobile applications. In modern world mobile phones or smartphones became indispensable attribute of each person. It allows people to have multifunctional gadget wherever they are. Developed system is not an exception. It integrates the main part of its functionality to the mobile application of the user and it makes it more demanding, popular and convenient to use. This mobile application was designed and developed for tree most popular platforms – iOS, Android and Windows 10 using the most powerful and progressive mobile development framework – Xamarin.

Operation System

Developed system is implemented using Microsoft technology stack base on Microsoft .Net Framework usage. This fact imposes some restrictions for required operation systems for system applications. For terminal device and application the only supported operation system is Windows IoT Core. All cloud application requires virtual machines with Windows 10 desktop version installed. Each mobile application requires corresponding mobile operation system – iOS starting from version 5, Android starting from version 4.2 and Windows 10 mobile.

Software modules functions

Developed system designed as a distributed system hosted in the cloud. From the first look it seems to work as the only functional application but in the reality it consists of eleven separate application that works independently of each other. Actually these devices can be divided into four functional groups:

5) Terminal side applications;
6) Backend side applications (server side applications);
7) Mobile side applications;
8) Communication layer applications.

Let’s get down to the functional responsibilities of each group and application they consist of.

The first group is Terminal side applications. This group consists of two applications: terminal UI application and terminal background process, both developed as a Universal Windows Applications. These applications hosted on the Raspberry Pi device and are running continuously during the whole time the system works.

Terminal UI application is the main application that provides user with Graphical User Interface for interacting with the system from terminal side. It handles all user actions performed by touch screen and notifies other system components about performed actions. It is responsible for the direct communication with the system – from terminal to other components.

Terminal background application – is a backward communication channel between system components and terminal. It listens for the messages from the system and handles them on the terminal side performing actions according to the message type and message parameters and arguments. Examples of these actions are opening and closing the media stream, handling the unlock door and lock door requests or reject the visitors intent. Also this application responsible for timeouts handling for lock/unlock operation and alert messages displaying.

Second group is Backend side applications. This group also consists of two applications Web Job and Web API. Both application are Web oriented and hosted in the cloud, they are responsible for bidirectional communication between terminal applications and mobile applications.

Web job is a cloud service that listens for the messages, performs business logic according to these messages, interacts with storage, performs events logging activities and notifies mobile applications according to performed work. It provides directional communication channel between terminal applications and mobile applications.

Web API is a web HTTP application. It is a restful API that provides to mobile applications the interface for interaction with the system. It is responsible for directional communication channel between mobile applications and other system components and also for manipulating with stored data from mobile application.

Mobile applications group consists of three application projects for the most popular mobile phone operation systems: iOS, Android and Universal Windows Platform and the common module for this applications that is used by each of them. This part of the system is out of scope of this paper so it will not be described here.

Last group of applications - Communication layer applications group consists also consists of three applications. These applications don’t perform any business logic work, they are responsible for the messaging between different components of the system. Applications are:

IoT Hub – bidirectional messaging bus that is used for communication of embedded resources like Raspberry Pi computer with corresponding software with server side applications or with the same embedded devices. It is hosted in the cloud as a separate service. Each device registered in this bus by admin can send the message of different types with different payload to this bus and some other application or device can listen for this application and perform some actions on receiving the message.

Notification hub – one directional message bus for communication between web applications and mobile devices. It is also hosted as a standalone web application and requires to register devices before communication set up.

Media service – web application that responsible for media streams flow control. It opens the media stream and sets up the endpoint with required metadata, so other consumers can connection to this endpoint using specially generated URI and subscribe for this stream.

Software extensions

Developed software was designed as a standalone system and in the scope of this paper does not require any extension. But flexible architectural design allows to extend this system with new home automation system features on demand.

4.2 Software user guide

Software developer who takes part in the automatic control system “Smart intercom” development or support needs a personal computer and specially configured environment.

The system requirements for personal computer:

1) central processing unit with minimal clock speed of 3.6 GHz;
2) 16GB of random access memory;
3) 100GB of a storage memory;
4) Screen with resolution at least 1920x1080;
5) Internet connection at least 10MBits.

Software requirements:

1) Microsoft Windows 10 operation system;
2) Microsoft .Net Framework 4.6 and higher platform;
3) Microsoft Visual Studio 2017;
4) JetBrains ReSharper 10;
5) Windows 10 SDK;
6) Mobile development SDK;

7) Mobile Emulator;
8) Microsoft SQL Server Express;
9) Microsoft SQL Server Management Studio;
10) Microsoft Azure Subscription;
11) Git for Windows;
12) IoT Dashboard;
13) IoT Remote Client.

Hardware requirements;

1) Raspberry Pi 3 Model B;
2) Waveshare 5inch sensor display;
3) Web camera;
4) 2 micro USB cables;
5) HDMI cable;
6) Patchcord.

Software characteristics

Developed applications work in real time. There was developed the event based architecture. Each modules rises special events on the work performed and includes the result of executed business rule into the metadata. Other modules can subscribe on this events if their business logic depends on this data and continue chain of data transformation. The chain can be considered as completed when last participant does not rise the new event.

Software main feature design

The application hosted on Raspberry PI device handles these actions from user, extracts the flat number and verifies it and if data is correct it puts the corresponding message to the IoT Hub. IoT Hub is responsible for this message to be delivered to the Web job application that has the special handler for this actions.

The Web job handler retrieves this message from IoT Hub, validates it and tries to find the metadata associated with the house holder’s mobile devices. Web job creates the notification message and sends it to the notification hub for all devices associated with householder account.

Mobile application receives this message and rises the visual, sound or vibration effects depending on configuration set up by user in application settings. The mobile application makes a REST/HTTP request to the Web API application.

The Web API application retrieves the special metadata required to request the media stream from Media service and tries to connect to this service.

The media stream requests Raspberry PI to configure the Camera device, to open the UDP socket and to start content streaming. When the stream is opened, Media service sets up the special endpoint for content streaming and returns the URI of this endpoint to Web API so mobile applications can connect to this stream and play the video/audio stream.

When the Web API receives the media stream endpoint URI, it prepares the web response to previously received request, fills the response body as a JSON object that contains media stream endpoint URI and returns it back to mobile application web client. Mobile application receives these artifacts and tries to connect to this endpoint

At this stage the house holder has possibility to see the visitor and to communicate with him.

To unlock the door mobile application using a HTTP client makes request to the Web API with data for door unlocking.

When the Web API application receives request for door unlocking it creates the corresponding message and puts it to the IoT Hub. Application hosted on the Raspberry PI device has a special handler for suck messages. Receiving this message it exposes next activities: it generates the signal to the locker to be unlocked, it shows corresponding message on the screen to notify guest that he is able to come in and finally it sets up the lock door timeout.

When the timeout has ended Raspberry PI application closes the media stream, locks the door and puts the corresponding message to the IoT Hub. After this, the Web job application notifies mobile application via notification hub that door has been closed and corresponding controls on the mobile phone screen are being updated.

Software usage

Before using the system it should be correspondingly deployed. All cloud applications should be deployed to Azure cloud with previously filled configuration files. Then the terminal applications should be deployed to the Raspberry Pi device. If required hardware is already connected to the device and internet connection established applications should be launched. Appeared UI on the touchscreen means successful deployed application. Finally the mobile application should be published to the target smartphone. To test the system administrator should create account for the householder and associate the mobile application with it.

Software input and output data

Software input data is determined by users actions as using terminal application as using mobile applications. System output data represented as notification sent in both directions: screen notifications on terminal application and push, screen, sound and vibration notifications on mobile side. Also logging activities could be considered as output data of the developed system.

4.3 Software testing technology

Software testing is the process of its research in order to obtain information about quality. The purpose of testing is to identify defects in the software. With the help of testing it is impossible to prove the absence of defects and the correct functioning of the analyzed program. Testing complex software products is a creative process that does not reduce to following strict and clear procedures.

Software testing covers a number of activities, very similar to the sequence of software development processes. This includes setting the task for the test, designing, writing tests, verification of the tests, and finally performing the tests and examining the test results. The design of the test plays a decisive role.

The composition and content of the documentation accompanying the testing process is determined by the foreign standard IEEE 829-2008 Standard for Software Test Documentation.

There are several grounds on which it is customary to classify the types of testing.

By the test object:

1) functional testing;
2) performance / load / stress testing;
3) usability testing;
4) user interface testing (UI testing);
5) security testing;
6) localization testing;
7) compatibility testing.

By knowledge about the system under test:

1) black box testing;
2) white box testing;
3) testing by the "gray box" method.

On the level of automation:

1) manual testing;

2) automated testing.

By degree of isolation:

1) unit testing;

2) integration testing;

3) system testing.

On the level of readiness

1) alpha testing;

2) beta testing;

3) acceptance testing.

The testing process is a highly repeatable process that requires a huge patience and attention from the engineer that executes tests. Any change made to the system requires to re-execute the whole suite of tests for changed module and for all modules dependent of it. Sometimes testing effort is much bigger than development. This is the main reason to define resources for automation testing design and implementation. There is special mechanism that helps to understand how many tests should be implemented for each system module and the system at all in software engineering practice – Software Testing Pyramid (Fig. 4.1).

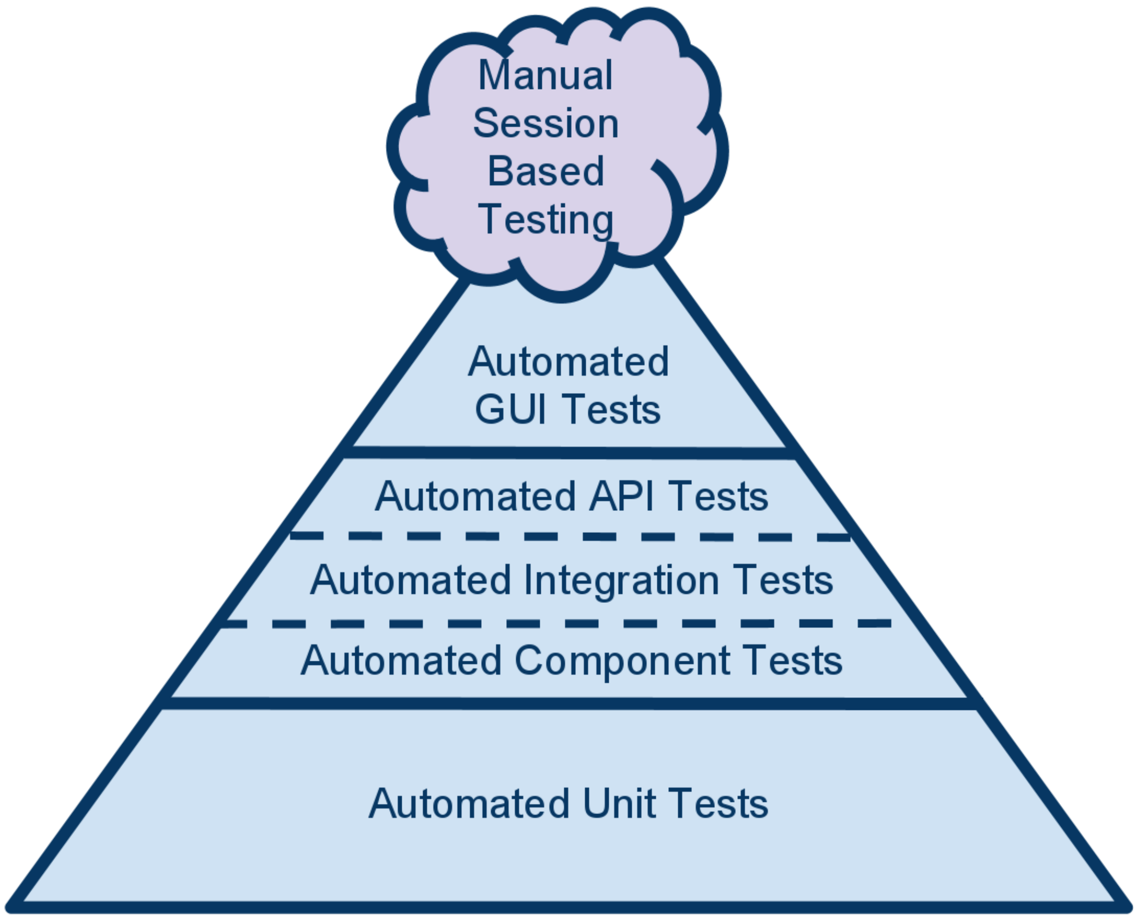

Manual Session Based Testing
Automated GUI Tests
Automated API Tests
Automated Integration Tests
Automated Component Tests
Automated Unit Tests

Figure 4.1 — Software testing automation pyramid

The test automation pyramid is a graphical strategy guide for implementing automated software testing. The model splits types of testing into three layers based on the return on investment offered by automating that particular type. The components of each layer can vary from one organization to another, but the main principles remain the same.

The foundation of this pyramid is Unit test automation.

The unit test is an important part of writing high-quality code. When people in software organizations speak of test automation, they tend to think of tools such as Unified Functional Testing or Selenium, which provide test automation frameworks. However, developers should write the majority of automation tests at the unit test level.

Developers can use unit test frameworks such as nUnit or Microsoft's Visual Studio Unit Testing Framework to create automated tests for small units of code. Some agile teams use test-driven development, a technique in which you write the unit test before the code to help drive code design. Some developers write the code first, but don't consider the code complete until they've developed an associated automated unit test. Each developer can assess whether each code path has been tested with test a coverage tool such as DotCover.

Automated unit tests are extremely fast to execute, and you'll want to run them after every build. This approach will give your team immediate feedback when regressions occur, as your code base continues to grow and evolve. Because the tests are so small and specific, it's easy to troubleshoot them when you have a failure. Having these tests gives your development team the peace of mind to refactor with confidence, safe in the knowledge that they'll quickly detect any new code that causes regressions.

The middle layer of the pyramid means the testing at service layer.

It's also known as the layer for automated API tests, automated component tests, or acceptance tests. Developers use this automation layer to test the business logic without involving the user interface (UI). By testing outside the UI, executor can test the inputs and outputs of the APIs or services without all the complications the UI introduces.

Testing at this level gives testers the option to set up data and go through a series of tests with the inputs and expected outputs you've defined in separate spreadsheets or files. This lets team create automated tests against boundary conditions, edge cases, or error conditions, without involving the UI. These tests are slower and more complicated than unit tests because they may need to access a database or other components. Every developer should absolutely use them, however, as they're still much faster and more reliable than UI tests.

The cherry on top of the pyramid is the UI layer.

Now that the majority of your code and business logic has been tested, most testing at the UI level has been eliminated. Developer’s focus now at the UI level is simply to ensure that the UI itself is working correctly. UI tests are very brittle, so it is good practice keep these tests to a minimum. These automation tests will need maintenance any time the UI changes, and because there are so many factors that come into play when you run a test that emulates clicks on a screen (such as network speed), such tests can result in false test failures. Developer can't ignore those test failures, but he does not want to end up spending more time troubleshooting and maintaining UI tests than he spends finding actual code defects.

Usage of this testing approach increases the development performance, developed software reliability and simplifies the software development process at all. This strategy was used during the automation control system “Smart intercom” implementation.

5 EXPERIMENTAL AND PRACTICAL PART

5.1 Tests suite

When system functionality is developed it is the time to verify if the system works as expected and there is no issues in a working progress. The software testing is a complex procedure that involves different specialists of software development industry: software developers, automation testing engineers, manual testing engineers etc. The testing complexity grows exponentially depending on different factors: functions count, modules count and in case of developed system it depends on applications the system consists of count.

According to the testing pyramid strategy testing process starts from unit testing. This testing could be performed in parallel to the development process. The idea is to cover each function that performs some business logic with test or several tests to verify that this function works correctly. It works next way: test creates the input data for the method and declares the expected data for this input. Test method passes the input data to the function under testing and retrieves function output. Then it verifies the actual data with expected and if they are equal – function considered as correct and test as passed. If data differs – function has wrong behavior and test became failed. If the function under testing has complex nonlinear logic, for example has some branching logic – there more than one test should be written to verify each scenario. And if at least one test is failed the function considered as wrong.

Unit testing is a responsibility of developer because it requires close interactions with the code. It has many advantages comparing to other testing methods. First of all it is very fast. Unit tests provide the quickest feedback to developer especially when existing code is changed and these changes broke some business rule. Also these tests reduces the time for searching the problem epicenter because they cover the smallest peace of software – the function.

When the unit tests suite passed it is time to verify the next composition unit of software system – module. This part is also performed by a developer because it requires the interaction with the code base and called integration testing. The aim is to verify that the module works correctly. It looks like the unit test but it works with much bigger piece of code. Other difference between integration tests and unit tests – is that integration could verify not only the output value but also the behavior of the module. For example, when test calls executes the functionality of the module under testing, it provides him some data and expects this data to be saved to database. It happens that all functions from module passed all tests and actually works correctly but because of module configuration issue it works wrong. So integration tests basically intended to verify the configuration correctness. These tests are also used to verify applications correctness when the system consists of more than one application. It considers the application as a module and verifies it work. And finally these tests verifies that the interactions between system components also work correctly.

When the system passed all tests: unit and integration there is time to verify interface if this system. In case of developed system it is graphical user interface. These tests could be performed both by developers and by testers. The aim is to verify that interface works correctly and interaction with business layer has no issues. This is the last step that can be automated.

The last step is acceptance testing. This is the test suite that verifies the features of the system. The scenarios of these tests are similar to user’s actions when they use the developed system. This tests are executed by testing engineers manually. When all test passed successfully the system considered as completed.

5.2 Testing results

In the scope of this paper there will be testing results only for server side part of the developed system. The system test suite consists of 154 unit tests, 43 integration tests 24 user interface test and suite for acceptance testing. By the moment of writing this paper all tests were executed successfully. It means that the system on each level works successfully. The results of acceptance testing are shown in the figures 5.1 – 5.7.

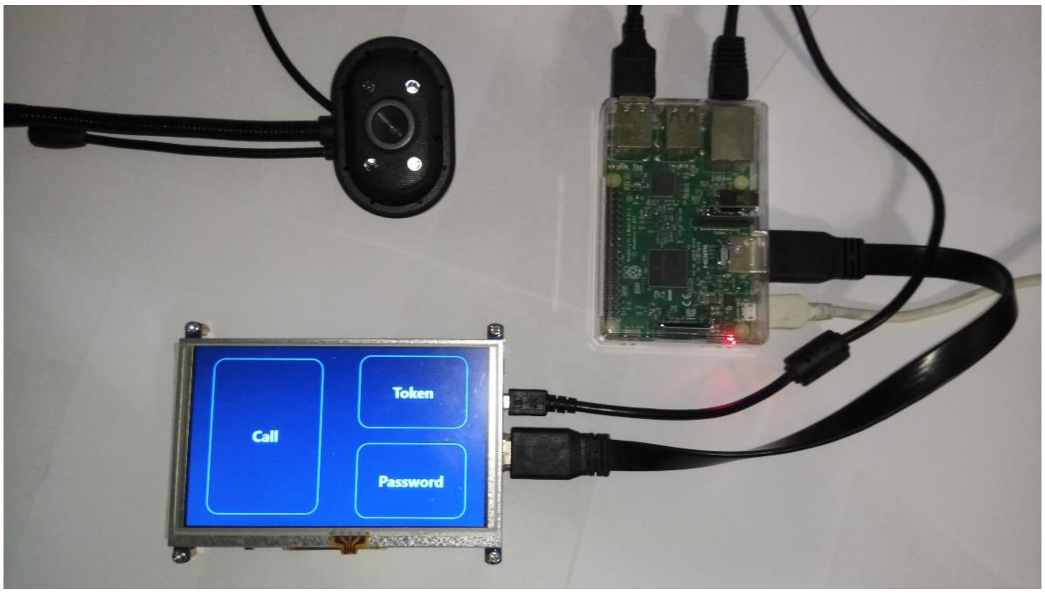


Figure 5.1 – The Main page of the terminal application

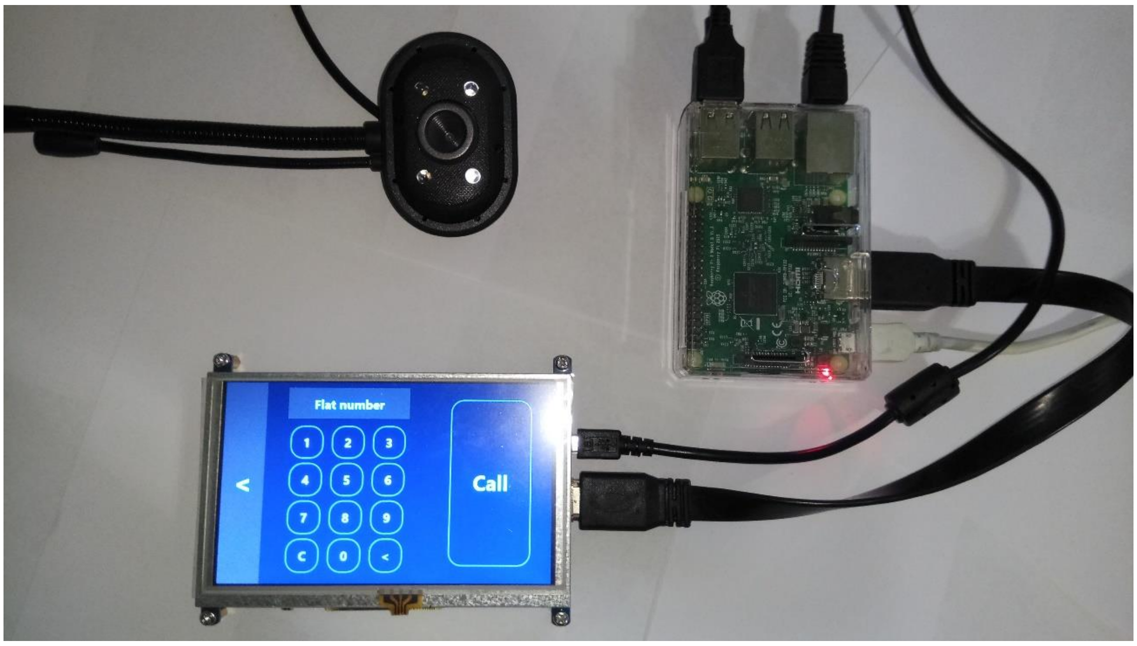


Figure 5.2 – The Call intent page of the terminal application

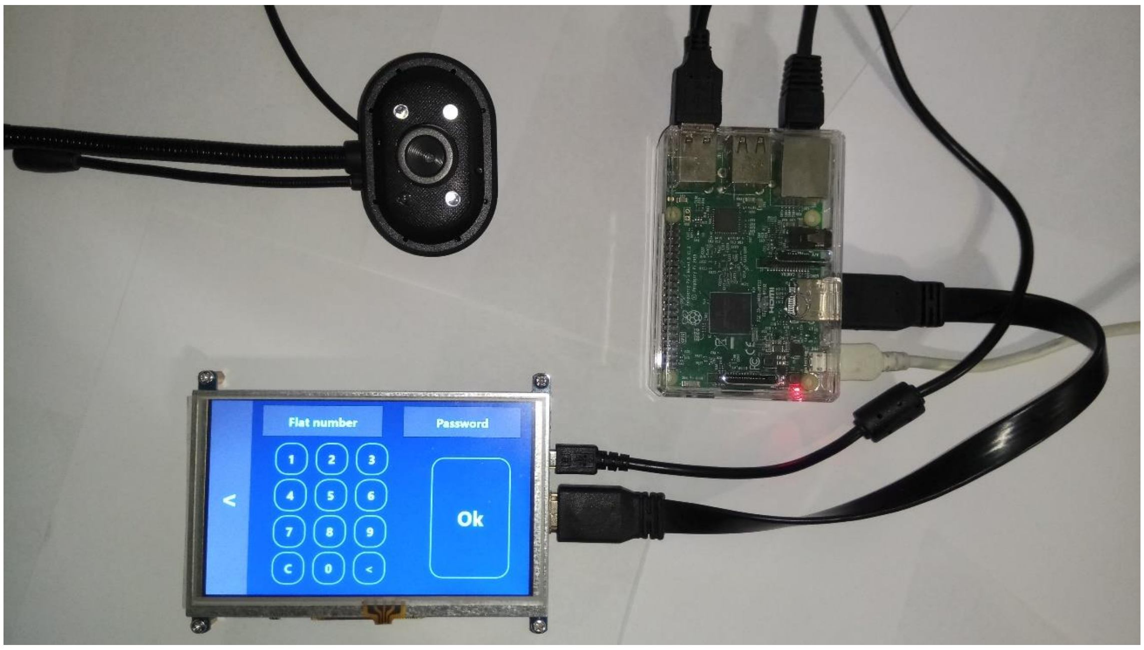


Figure 5.4 – The Password authorization page of the terminal application

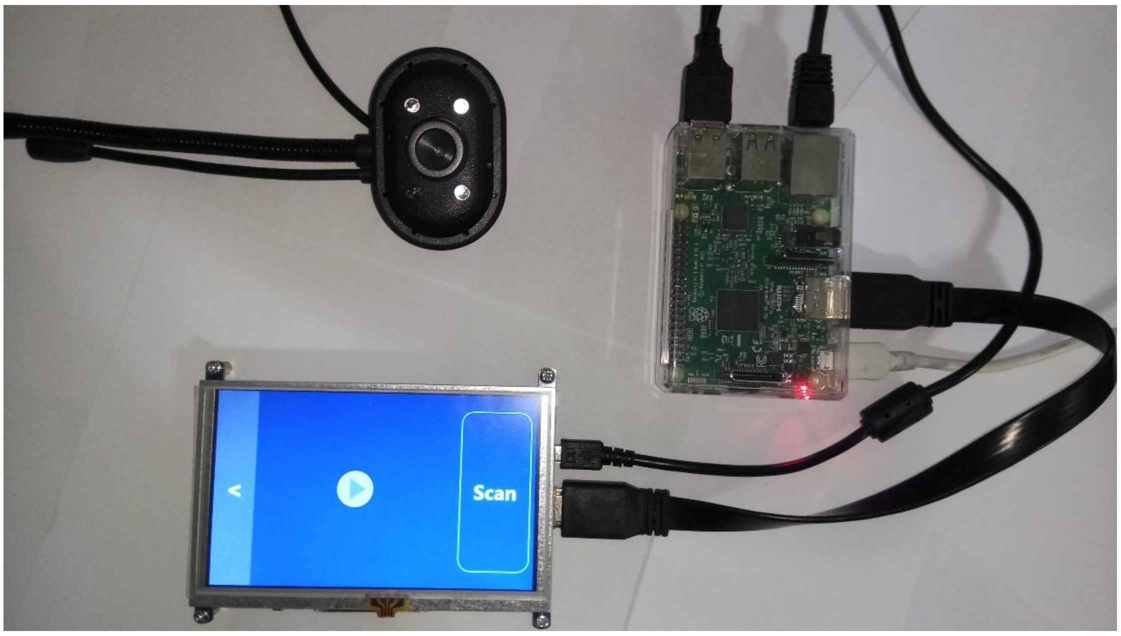


Figure 5.3 – The Token authorization page of the terminal application

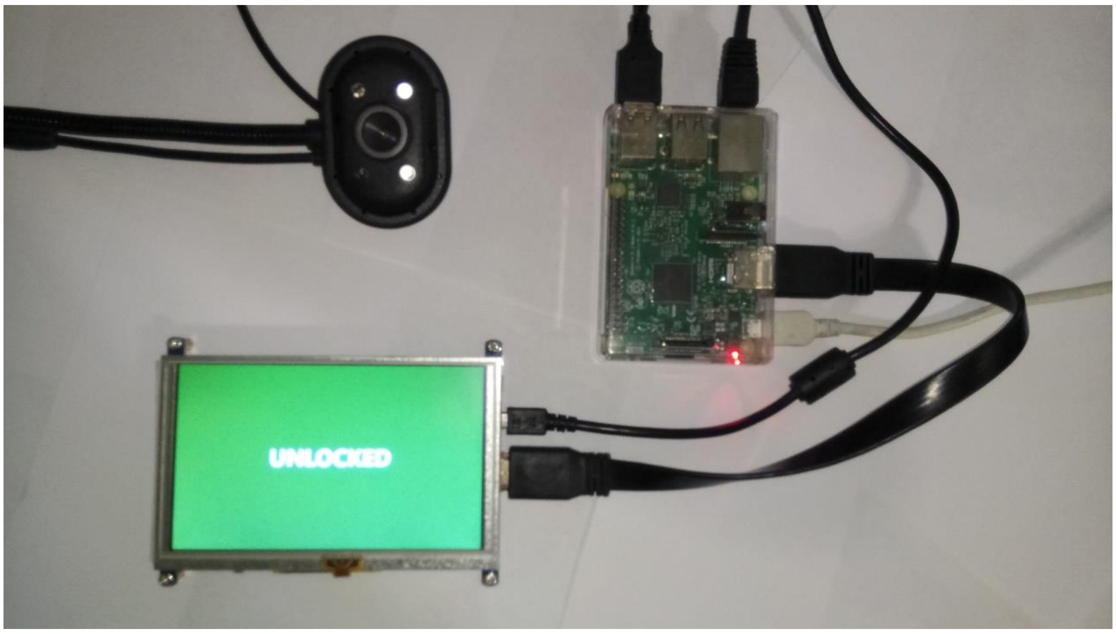


Figure 5.5 – The Door is opened notification page of the terminal application

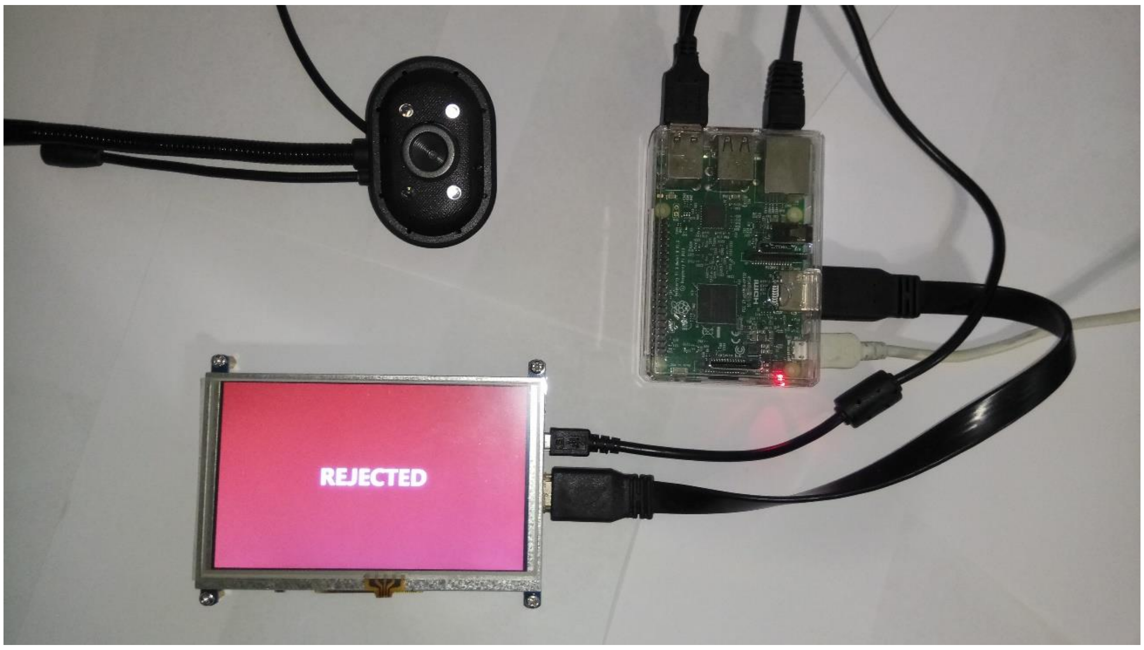


Figure 5.6 – The Door is opened notification page of the terminal application

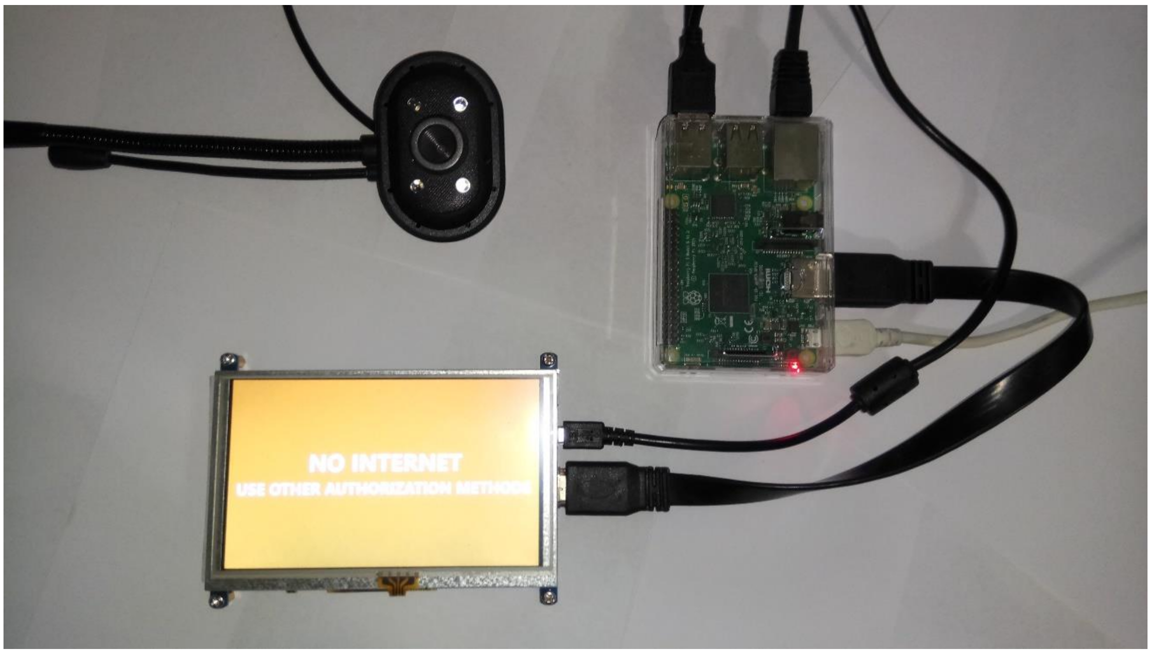


Figure 5.7 – The Connection failure notification page of the terminal application

5.3 Conclusions

Testing process was performed according to the best practices in software testing engineering. It was designed according to the software testing pyramid principle that allowed to automate the overwhelming amount of tests and reduce the effort of manual testing. There was used modern powerful technologies for system testing – nUnit framework for unit testing and Selenium tool for graphical interface testing.

The system test suite consists of 154 unit tests, 43 integration tests 24 user interface test and suite for acceptance testing. By the moment of writing this paper all tests were executed successfully.

6 ECONOMIC JUSTIFICATION FOR THE SYSTEM DEVELOPMENT

6.1 System description and practical results investigation

The automatic control system “Smart intercom” is offered for production. System intended to establish convenient communication channel between house holder and his visitors using mobile application being in the any point of the planet where internet connection is available.

The application area is different kinds of home automation system. Developed system can be used as standalone component or being embedded to existing system.

6.2 Market segmentation

Market segmentation is the process of dividing consumers into groups taking into account different principles and factors of segmentation.

In the process of segmentation, market segments are identified. A segment is a group of consumers who respond equally to a product. The market segment is estimated by a number of characteristics number of possible customers, the market capacity, the possible growth rates of capacity by years, consumer prices, profitability of sales, etc.

The general order of the market segmentation can be represented as follows:

1) Identification of the basic principles and factors of segmentation for the product;
2) the market segmenting: determining the composition of consumer groups, dividing the market into segments, describing the profile of each segment, calculating the annual capacity of segments and the entire market;
3) the results of segmentation registration in the tables form;

4) analysis of information about segments, selection of segments for further analysis (segments with a small capacity can no longer be considered);
5) positioning the goods in selected segments, determining the target capacity.

Consumers of the unit being developed are various spheres of life. The main characteristic of the segment is capacity - the number of products that can be sold in a year.

Calculations of market capacity are performed after determining the composition of segments and begin by determining the total demand for a product of this type:

$$S_{\text{full.}} = \sum_{\text{i}=1}^{\text{L}} S_{1\ \text{full i.}}\ , \tag{6.1}$$

Where $S_{full.}$ – full product demand in the all segments, pcs / year;

$S_{1\ full\ i.}$ – the full demand of the one segment, pcs / year;

i – segment number;

L – segments amount, pcs.

The total demand of the segment is calculated taking into account the specific features of the product and segments. For many types of goods for individual and industrial use $S_{1 ful\ i}$ can be calculated by the following formula:

$$S_{1\ full\ i.} = N_1 \cdot Q_1 \cdot m_1, \tag{6.2}$$

where N1 – amount of consumers for which the developed system will be delivered;

Q1 – average annual program for products in current segment, which the developed product will be delivered for;

m1 - the number of components that combines into one product (pcs.).

Segmentation and calculation of market capacity is presented in Table 6.1.

Table 6.1 - Segmentation and calculation of market capacity

Manufacturer	Parameters			
	N_1	Q_1, pcs.	m_1, pcs.	S_1, pcs
“Cyfral”	6	200	1	1200
“World Security”	1	80	1	80
“NeoLight”	5	180	1	900
“Sevidom”	3	140	1	420
Total	15	600	3	2600

Table 6.1 shows that the full capacity of the market is 2600 pcs/year.

6.3 Competitiveness analysis

Competitiveness of the product is the degree of its conformity to the chosen market on commercial, technical and economic indicators, providing opportunities of sale of the product in this market. These are the characteristics that distinguish this product from competitive analogs.

Let’s assess the competitiveness of the system being developed.

The analysis of competitiveness is conducted by comparing the characteristics of the system with the characteristics of analogs for certain parameters. We use the method of complex quality indicators with the calculation of generalized quality indicators. For products of competitors the following products were accepted:

1) “Cyfral”, Slinex SL-07M;
2) “World Security”, Infinitex MX471;
3) “NeoLight”, Kappa+.

Next parameters will be considered as a quality criteria:

1) Video call;

2) Guest pass;
3) Access via internet;
4) Smart home integration;
5) Mobile application;
6) Decentralized structure.

Let’s determine the absolute value of i-th criteria of j of P_{ij} variants in points. Let’s assign weight coefficients b_i to each quality criteria:

$$\sum_{i=1}^{n} b_i = 1 \text{ и } b_i<0, \ i=1, \tag{6.3}$$

where n – the number of quality criteria.

Quality criterias are divided into minimized and maximized and form a hypothetical (reference) version.

Let’s calculate the relative values of i-th indexes (k_{ij}) for each j-th variant comparing P_{ij} with $P_{i\,hypo}$ (with reference to $k_{ij} \leq 1$) for minimized and maximized values.

Retrieved values are shown in table 6.2.

Criteria	Significance coefficient	Variants			
		1	2	3	4
Video call	0.3	1	1	1	1
Guest pass	0.2	1	1	0	1
Access via internet	0.2	1	0	0	1
Smart home integration	0.05	0	1	1	1
Mobile application	0.2	0	0	0	1
Decentralized structure	0.05	1	0	0	1

Relative values					
Video call	0.3	1	1	1	1
Guest pass	0.2	1	1	0	1
Access via internet	0.2	1	0	0	1
Smart home integration	0.05	0	1	1	1
Mobile application	0.2	0	0	0	1
Decentralized structure	0.05	1	0	0	1
	1	0.75	0.55	0.35	1

where 1 – Slinex SL-07M;

2 – Infinitex MX471;

3 – Kappa+;

4 – hypothetical option.

Thus, the generalized indicator of the quality of the developed system is the highest, K = 1, as can be seen from Table 6.2. It means that this product is fully competitive.

Received generalized indicators k_{j0} for all the options considered. Let's calculate the quality levels of our system in comparison with competing complexes:

$$Y_{j-b} = \frac{k_{j0}}{k_{k0}}, \tag{6.4}$$

where k_{j0} generalized indicator of the competitor's device.

Levels of quality of our system in comparison to systems competitors are:

$$Y_{j-b} = \frac{1}{0.75} = 1.33, \;\; Y_{j-b} = \frac{1}{0.55} = 1.82, \;\; Y_{j-b} = \frac{1}{0.35} = 2.86.$$

6.4 Calculation of the cost price and the price of the designed product

The cost price of the product consists of a number of cost items. This includes costs for basic materials, large component parts, direct and additional wages, maintenance and operation costs of equipment, transport maintenance, as well as a number of state taxes and deductions.

We calculate the cost of purchased products needed to develop required software and to manufacture and launch the system at all.

The list of purchased products is made taking into account the list of sub-blocks of the functional scheme (Table 6.3). The prices are given in UAH.

Table 6.3 – The list of bought products

Name of product	Technical designation	Quantity of products in a system	Unit price, UAH	Amount UAH
1	2	3	4	5
Single board computer	Raspberry Pi 3	1	1000	1000
Touch screen	Waveshare 5 inch LDC	1	1300	1300
HDMI cable	HDMI	1	70	70
USB cable	USB	2	35	70
Total				2440

To design, develop and deploy, it is necessary to participate the following working staff: lead software engineer, software engineer, software testing engineer, systems engineer. The average working month lasts 22 days. The composition of the performers is given in Table. 6.4.

Let’s calculate the duration of development by type of work. The result of calculations is contained in Table. 6.4.

Table 6.4 - Composition of work executors

Positions	Official salaries, UAH	
	Month	Day
Lead software engineer	40000	1818.18
Software engineer	25000	1136.36
Software testing engineer	15000	618.82
Systems engineer	20000	909.09

Let's calculate the labor intensity of the work. The calculation of the results is given in table 6.5.

Table 6.4 - The labor intensity of the work

Type of work	Duration, days	Labor intensity, person / day	Performer			
			Lead software engineer	Software engineer	Software testing engineer	Systems engineer
1	2	3	4	5	6	7
Formulation of the problem	1	1	+	-	-	-
Development of work schedule	2	2	+	-	-	-
Determining the requirements for the development of a board	4	4	+	-	-	-
Development of technical specification	3	3	+	-	-	-
Technical task approval	1	1	+	-	-	-
Software design						
Preparatory work	1	1	+	-	-	-
Assembly work	2	4	-	+	-	+

Acceptance works	2	4	+	-	+	-
Software development						
Development	20	40	+	+	-	-
Testing	7	14	-	+	+	-
Debugging	4	8	-	+	-	+
Deployment	3	12	+	+	+	+
Introduction						
Adjusting the product	2	6	-	+	+	+
Testing and commissioning	2	4	+	-	+	-
Total	54	104	39	38	16	11

Thus, the total duration of work will be 54 days, the total labor intensity - 104 people / days. The labor intensity of the lead software engineer will be 39 days, the software engineer - 38 days, the software testing engineer - 16 days, the systems engineer - 4 days.

Let's calculate the basic wage of employees involved in the system design, development, testing and deployment, taking into account labor costs, the number of performers and the average daily salary. For this, the number of days worked by individual performers is multiplied by their daily salaries:

$$F_{wage} = 1818.18 \cdot 39 + 1136.36 \cdot 38 + 618.82 \cdot 16 + 909.09 \cdot 11 \\ = 133991.81 \text{ грн.}$$

Let's calculate the additional wage, which is 15% of the wage:

AWAGE = 15% · WAGE = 0.15 · 133991.81 = 20098.77 UAH. (6.5)

Let's calculate the single social contribution:

SSC = 22% · (WAGE+AWAGE)= 0.22 · (133991.81 + 20098.77) =
= 33899.93 грн. (6.6)

We also calculate the amortization (Am) based on the value of fixed assets. The cost of fixed assets is shown in Table. 6.6.

Table 6.6 - Fixed assets

№	Equipment	Price, UAH	Quantity	Amount, UAH
1	Table	600	4	2400
2	Chair	120	4	480
3	Personal computer	27000	4	108000
4	Software	5000	3	15000
	Total			147480

$$\text{Aм} = \frac{25\% \cdot total}{12 \cdot 22} = \frac{0.25 \cdot 147480}{12 \cdot 22} = 139.65\ UAH \tag{6.7}$$

Also calculate the amount of factory costs:

$$\text{P}_{wage} = WAGE \cdot \frac{\text{H}_{WAGE}}{100}, \tag{6.8}$$

where H_{WAGE} - the standard of general factory expenses. So:

$$\text{P}_{wage} = 133991.81 \cdot \frac{20}{100} = 26798.36. \tag{6.9}$$

Calculate the costs of shop management:

$$\text{P}_{SM} = \text{ОЗП} \cdot \frac{\text{H}_{SM}}{100}, \tag{6.10}$$

where P_{SM} – standard of shop management. So:

$$\text{P}_{SM} = 133991.81 \cdot \frac{10}{100} = 13399.18\ UAH \tag{6.11}$$

Table 6.7 shows the calculation of the cost and price of the product.

Table 6.7 – Calculation of production costs and product prices by item

№	Topics	Amount, UAh	Description
1	Materials and bought products	2440	From table 6.3
2	Wage	133991.81	
3	Additional wage	20098.77	15% from wage
4	Deductions to social funds	33899.93	22% from wage and additional wage
5	Amortization	139.65	$\frac{25\% \cdot p}{12 \cdot 22}$
6	Shop management costs	13399.18	$P_{\text{уц}} = \text{WAGE} \cdot \frac{H_{SM}}{100}$
7	General plant costs	26798.36	$P_{\text{wage}} = \text{WAGE} \cdot \frac{H_{\text{wage}}}{100}$
8	Cost price (C)	230767.7	p.1+….+p.7
9	Profit (P)	46153,54	20% from C
10	Price without VAT	276921,24	P+C
11	VAT	55384.25	20% from price without VAT
12	Price with VAT	332305.5	p.10+p.11

Thus, the price of automatic control system without VAT is 276921.24 UAH, taking into account VAT - 332305.5 UAH.

Profitability of products (rate of profit) is the ratio of the amount of profit to the cost of production and sales of the product (the relative amount of profit per 1 UAH of current costs):

$$P_P = \frac{Price - Cost\ price}{Cost\ price} \cdot 100\% = \frac{332305.5 - 230767.7}{230767.7} = 44\%. \quad (6.12)$$

6.5 Calculation of breakeven point

When implementing a product, it is important to know whether this production process will become profitable and whether it will bring the desired profit. To do this, you need to determine the break-even point (BEP) and graph it graphically.

To confirm the sustainability of the project, it is necessary that the BEP value is less than the nominal production volume. The further from them the value of BEP (in percentages), the more sustainable the project. The project is generally recognized as sustainable.

Break-even point can be calculated by the formula:

$$N_{BEP} = \frac{К}{P - С}, \tag{6.13}$$

where K - conditional-constant costs, we accept equal to the price of the topic; P, C - the price and unit cost of the product.

The price of the topic is determined by the formula:

$$P_{topic} = С_{topic} + R_{topic}, \tag{6.14}$$

Where $С_{topic}$ - the general estimate of expenses (cost price) of a subject, UAH; R_{topic} - planned profit, ensuring cost-effective work of the direct executors of the topic, UAH.

We calculate the price of the topic:

$$P_{topic} = 230767.7+46153.54= 276921.24 \text{ UAH.}$$

The calculated price is the pre-contract price of the developer - this is the minimum acceptable price, taking into account the cost estimates for the development of the topic and the profit calculated according to the set ratio of profitability.

With the final appointment of the price of the topic, it is necessary to take into account the surcharges associated with the sale of the product. Value added tax is accepted at 20% of the price of the topic. Break-even point is:

$$N_{BEP} = \frac{276921.24}{332305.5 - 230767.7} = 2.73 \approx 3\ pcs.$$

Thus, the indicator BEP = 3. This means that the implementation of the third device will ensure break-even of the project.

The graphical representation of TB is shown in Fig. 6.1.

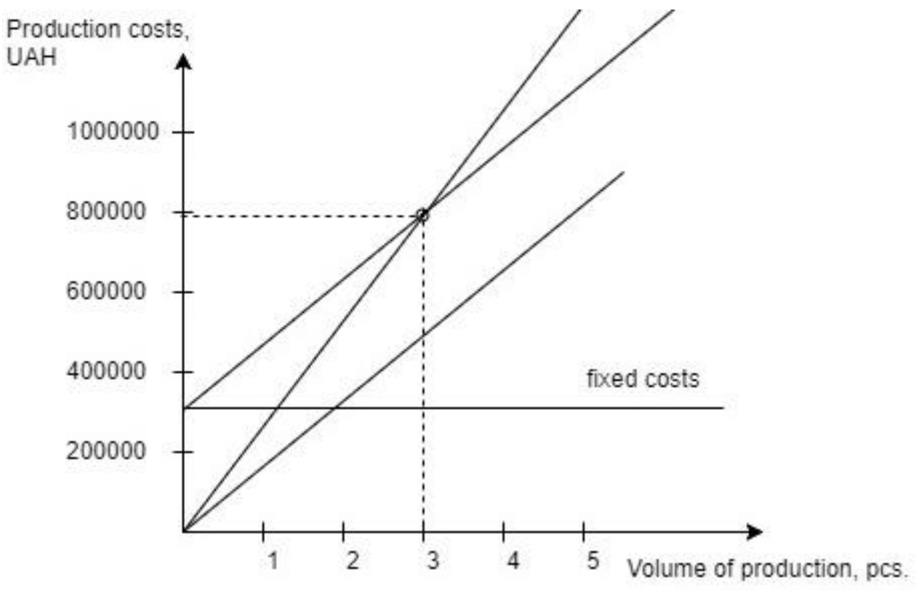


Figure 6.1 - Break-even point figure

The break-even point shows that with sales of 3 units, the income from the sale of products exceeds the total costs, so in the zone between them we make a profit.

6.6 Conclusions

In this section, the calculation of the price and the cost of tautomatic control system “Smart intercom” was made. The price of the product without VAT amounted to 276921,24 UAH, the price with VAT amounted to 332305.5 UAH. The cost of the payment is 55384.25 UAH. The calculation was carried out taking into account all the necessary labor costs.

This system was associated with the products-competitors and as a result the system developed was quite competitive.

The complexity of stages, the arrangement of performers are determined. The total labor intensity was 104 people / days, the lead software engineer labor intensity was 39 days, the software engineer - 38 days, the software testing engineer - 16 days, the systems engineer - 11 days. The calculation of wages is carried out, the price of the topic is determined. The basic salary is equal to 133991.81 UAH, and the price of the topic is 276921.24 UAH. Profit from the sale of products will be 44%.

Based on the found break-even point - 3 pcs. As well as the total capacity of the market in 2600 pcs. per year, we can say that the system will quickly pay off and will bring a stable income.

CONCLUSION

The purpose of this work is to design an automatic control system "Smart Intercom" with the usage of modern technologies and equipment. The developed system refers to a class of home automation systems - systems capable to perform actions and solve certain everyday tasks without human involvement.

The control object is based on the Raspberry Pi 3 single-board computer running the Windows 10 IoT Core operating system. The required functionality is provided by an application written using Microsoft .NET and Universal Windows Platform technologies. Since the most promising services for hosting server applications are cloud-based technologies, the cloud-based services of Microsoft Azure will be used in this system. The mobile application is designed to expand on various platforms, as Xamarin technology is used in this project, and it allows to develop mobile applications for different platforms using a common code base.

As a modern software system automatic control system “Smart intercom” was designed according to a common software development principals and approaches: Object Oriented Design principle and SOLID principle.

Testing process was performed according to the best practices in software testing engineering. It was designed according to the software testing pyramid principle that allowed to automate the overwhelming amount of tests and reduce the effort of manual testing. The system test suite consists of 154 unit tests, 43 integration tests 24 user interface test and suite for acceptance testing. By the moment of writing this paper all tests were executed successfully.

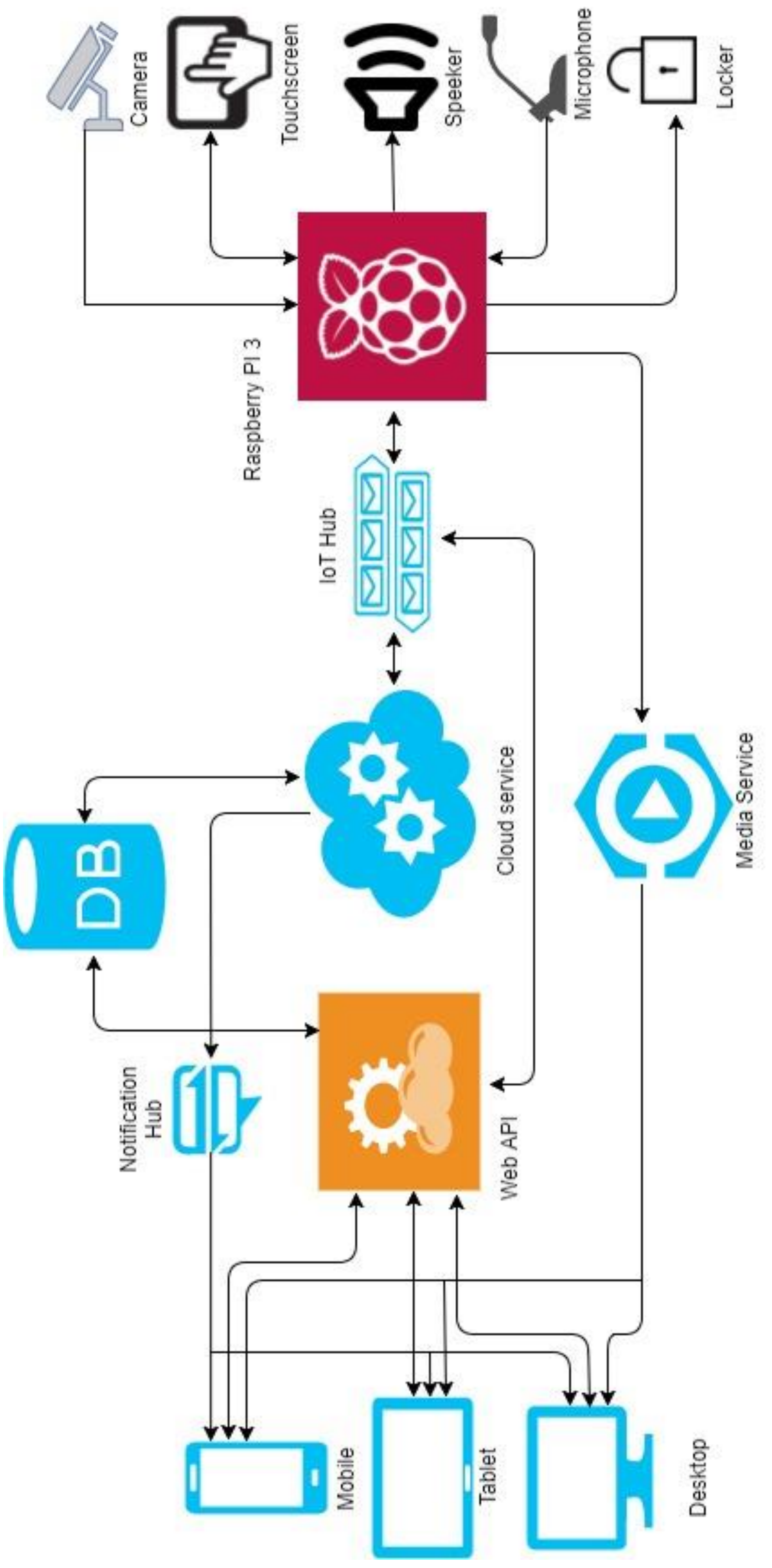
There is 4 people involved to the system development: lead software engineer, software engineer, software testing engineer and systems engineer. The total labor intensity of the system development was 104 people/day. The price of the product without VAT amounted to 276921,24 UAH, the price with VAT amounted to 332305.5 UAH. Based on the found break-even point - 3 pcs. As well as the total capacity of the market in 2600 pcs/year, the system will quickly pay off and will bring a stable income.

REFFERENCES

1. Programming for the Internet of Things: Using Windows 10 IoT Core and Azure IoT Suite / Dawid Borycki – Redmond, Washington: Microsoft Press, 2017. – 1276 p.
2. The Silent Intelligence: The Internet of Things / Daniel Kellmereit - DND Ventures LLC, 2013 – 341p.
3. The Real Internet of Things / Daniel Miessler - Amazon Digital Services LLC, 2017 – 103p.
4. Smart Homes: Design, Implementation and Issues (Smart Sensors, Measurement and Instrumentation) / Nagender Kumar Suryadevara – Springer, 2015 – 180p.
5. Xamarin Cross-Platform Application Development / Carl Bilgin – Redmond, Washington: Print-on-Demand, 2016. – 390 p.
6. Cloud Design Patterns / Alex Homer, John Sharp, Larry Brader – Microsoft corp., 2014 – 236p.
7. Software Architecture Patterns / Mark Richards - O’Reilly Media, Inc., 2015 – 55p.
8. Microsoft .NET: Architecting Applications for the Enterprise, Second Edition/ Dino Esposito, Andrea Saltarello - Microsoft Press, 2014 – 99p.
9. Design Patterns: Elements of Reusable Object-Oriented Software Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, 1994 – 395p.
10. Software Testing Techniques, 2nd Edition / Boris Beizer - Itp – Media, 1990 - 580p.
11. Экономическая оценка инженерных решений: учеб. пособие/В.Н. Гавва, М.Я. Голованова– Х.: Нац. аэрокосм. ун-т «ХАИ», 1999 – 134 с.
12. https://developer.microsoft.com/ru-ru/windows/iot
13. https://www.xamarin.com/university

14. https://metanit.com/sharp/
15. https://msdn.microsoft.com
16. https://stackoverflow.com
17. https://www.pluralsight.com/
18. https://www.hackster.io/
19. https://www.raspberrypi.org/
20. https://wikipedia.org/

APPENDIX A


Camera
Touchscreen
Speeker
Microphone
Locker
Raspberry PI 3
IoT Hub
Cloud service
Media Service
DB
Notification Hub
Web API
Mobile
Tablet
Desktop