

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ИНСТИТУТ)

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра 307 «Цифровые технологии и информационные системы»

ОТЧЕТ

О выполнении задания по предмету

«Архитектура информационных систем»

«Робот-доставщик напитков»

Выполнили студенты группы М3О-221Б-21:

Юзлибаева Алина Римовна Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Олейник София Сергеевна Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил преподаватель:

Максимов Алексей Николаевич Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва 2023 г.

Содержание

[Цель работы 3](#_Toc136697665)

[Описание 4](#_Toc136697666)

[Разработка решения 5](#_Toc136697667)

[Реализация 7](#_Toc136697668)

[1. Файл «main.cpp» 7](#_Toc136697669)

[2. Файлы «tgbot.cpp» и «tgbot.h» 7](#_Toc136697670)

[3. Файлы «route.cpp» и «route.h» 14](#_Toc136697671)

# Цель работы

Целью работы является проектирование робота-доставщика напитков в коворкинге. Реализовать робота на платформе AlphaBot под управлением Raspberry PI 4.

# Описание

Список требований, которым должна удовлетворять проектируемая система:

1. Предоставлять механизм передачи роботу непосредственного управления роботу (вперед, назад, поворот налево, поворот направо, стоп).
2. Предоставлять механизм передачи роботу высокоуровневых команд (привезти сок).
3. Сок должен наливаться автоматическим дозатором.
4. Система должна автоматически определять положение робота.
5. Система должна знать положение дозатора.
6. Система должна автоматически находить и знать позицию «студента-клиента».
7. Необходимо предотвращать передачу команд от других студентов в процессе выполнения заказа.
8. Робот должен прокладывать маршрут и автоматически приезжать к «студенту-клиенту» из любой точки коворкинга и позволять поставить чашку на поднос.
9. Робот должен прокладывать маршрут и автоматически приезжать от студента клиента к дозатору от «студенту-клиенту».
10. Робот должен дождаться налива сока в стакан из дозатора.
11. Робот должен прокладывать маршрут и автоматически возвращаться с чашкой сока к «студенту-клиенту» от дозатора.

# Разработка решения

Общая архитектура нашего решения представлена на рисунке 1.

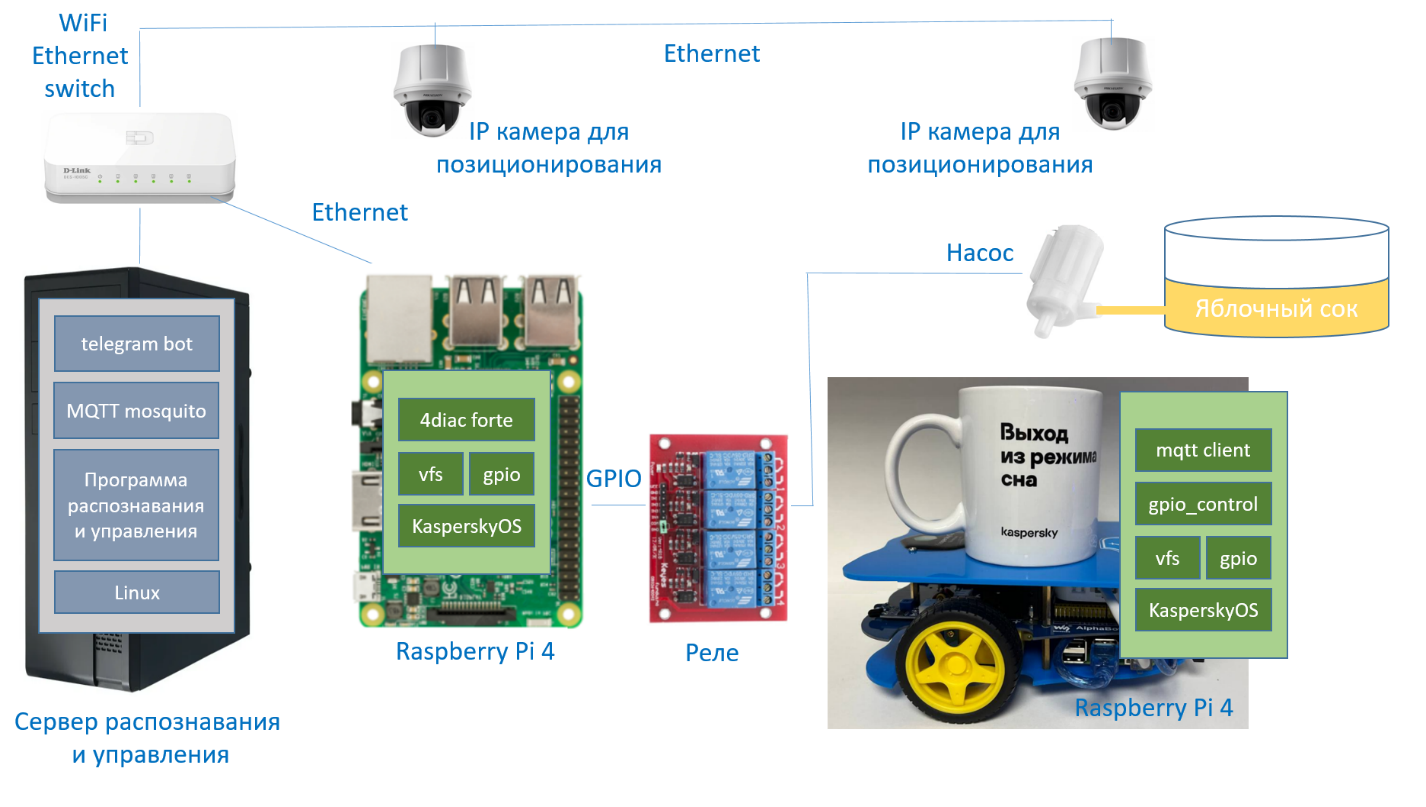
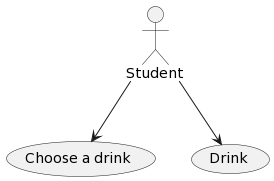
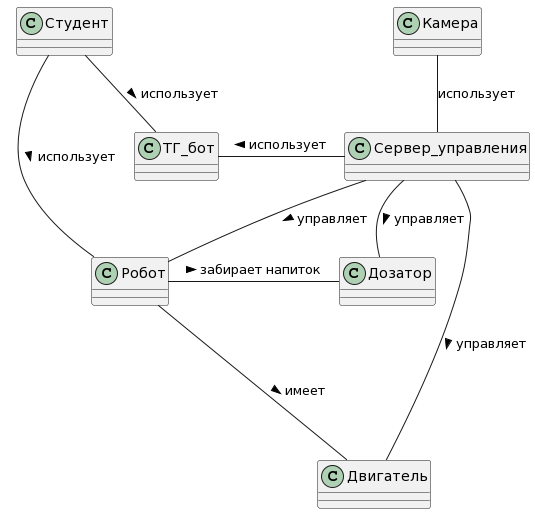


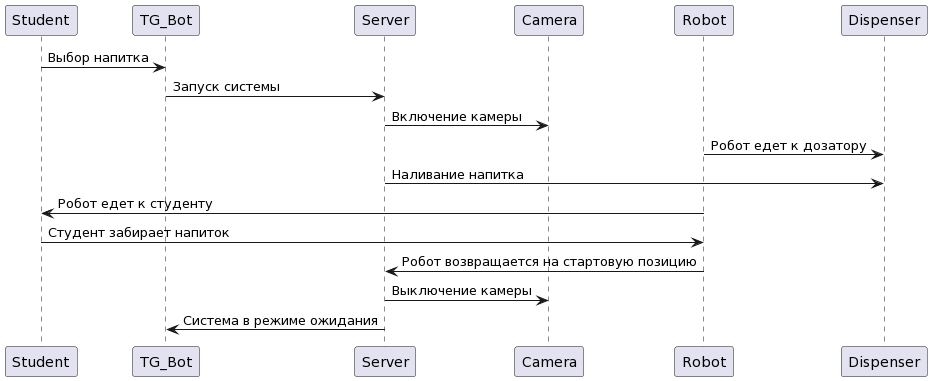
Рисунок 1 — Общая архитектура

Мы разработали диаграмму прецедентов, описывающую взаимодействие пользователя с системой.



Концептуальная модель необходима, чтобы разобраться с архитектурой системы (рис.)



Также была разработана диаграмма последовательности, представленная на рисунке .

# Реализация

## 1. Файл «main.cpp»

Содержание «main.cpp» (мб тут тоже че то написать?):

#include "tgbot.h"

BOT\_TOKEN = "6092622281:AAEzaSMdLB4LaKGzP23sEmAdfTYgL6IqeKc";

MQTT\_TOPIC = "abot/command/alex";

CHAT\_ID;

MQTT\_SERVER\_ADDRESS = "10.0.2.10";

MQTT\_SERVER\_PORT = 1883;

password = "1234";

int main() {

robot\_launch();

return 0;

}

## 2. Файлы «tgbot.cpp» и «tgbot.h»

В файле «tgbot.cpp» \*Соф, опиши что делает тут код\*

#include "tgbot.h"

mosq = mosquitto\_new(NULL, true, NULL);

flag = false;

password\_flag = false;

password;

first = false;

// ОТПРАВКА КОМАНДЫ РОБОТУ

void message(double angle, double dist) {

angle = round(angle \* 10);

dist = round(dist \* 10);

string str\_angle = to\_string(angle);

string str\_dist = to\_string(dist);

string cmd;

if (angle >= 0) {

cmd = "{\"cmd\":\"right\",\"val\":" + str\_angle + ", \"spd\":0.6}";

mosquitto\_publish(mosq, NULL, MQTT\_TOPIC.c\_str(), cmd.length(), cmd.c\_str(), 0, false);

cmd = "{\"cmd\":\"forward\",\"val\":" + str\_dist + ", \"spd\":0.6}";

mosquitto\_publish(mosq, NULL, MQTT\_TOPIC.c\_str(), cmd.length(), cmd.c\_str(), 0, false);

}

if (angle < 0) {

cmd = "{\"cmd\":\"left\",\"val\":" + str\_angle + ", \"spd\":0.6}";

mosquitto\_publish(mosq, NULL, MQTT\_TOPIC.c\_str(), cmd.length(), cmd.c\_str(), 0, false);

cmd = "{\"cmd\":\"forward\",\"val\":" + str\_dist + ", \"spd\":0.6}";

mosquitto\_publish(mosq, NULL, MQTT\_TOPIC.c\_str(), cmd.length(), cmd.c\_str(), 0, false);

}

}

enum class Rob\_State {

Start = 0,

To\_Dispenser = 1,

To\_Student = 2,

};

state = Rob\_State::Start;

void ProcessFiniteAutomat(TgBot::Bot& bot)

{

switch (state)

{

case Rob\_State::Start:

{

vector <vector<double>> current\_data = coords();

while (abs(current\_data[1][0]) > 0.1 || current\_data[1][1] > 0.1) // угол и расст

{

current\_data = coords();

message(current\_data[1][0], current\_data[1][1]);

bot.getApi().sendMessage(CHAT\_ID, "I'm going to the dispenser");

}

bot.getApi().sendMessage(CHAT\_ID, "Arrived at the dispenser");

state = Rob\_State::To\_Dispenser;

}

case Rob\_State::To\_Dispenser:

{

bot.getApi().sendMessage(CHAT\_ID, "Waiting for the preparation of the drink! This will take 20 seconds");

Sleep(5000);

vector <vector<double>> current\_data = coords();

while (abs(current\_data[2][0]) > 0.1 || current\_data[2][1] > 0.1)

{

current\_data = coords();

message(current\_data[2][0], current\_data[2][1]);

bot.getApi().sendMessage(CHAT\_ID, "Going to the student");

}

bot.getApi().sendMessage(CHAT\_ID, "Got to the student");

state = Rob\_State::To\_Student;

}

case Rob\_State::To\_Student:

{

bot.getApi().sendMessage(CHAT\_ID, "Waiting 20 seconds for you to pick up the drink");

Sleep(5000);

vector <vector<double>> current\_data = coords();

while (abs(current\_data[0][0]) > 0.1 || current\_data[0][1] > 0.1)

{

current\_data = coords();

message(current\_data[0][0], current\_data[0][1]);

bot.getApi().sendMessage(CHAT\_ID, "Going to the start point");

}

bot.getApi().sendMessage(CHAT\_ID, "Returned to the starting point");

state = Rob\_State::Start;

}

}

}

void robot\_launch()

{

setlocale(LC\_ALL, "Rus");

TgBot::Bot bot(BOT\_TOKEN);

mosquitto\_lib\_init();

mosquitto\_connect(mosq, MQTT\_SERVER\_ADDRESS.c\_str(), MQTT\_SERVER\_PORT, 0);

mosquitto\_subscribe(mosq, NULL, MQTT\_TOPIC.c\_str(), 0);

bot.getEvents().onAnyMessage([&bot](TgBot::Message::Ptr message) {

if (!first) {

CHAT\_ID = message->chat->id;

bot.getApi().sendMessage(CHAT\_ID, "Hi! I'm robot Petya, delivering beer and troubles. Write password");

}

if (password\_flag) {

if (message->text == "start") {

flag = true;

}

else {

bot.getApi().sendMessage(CHAT\_ID, "You've written " + message->text + ", but I don't care");

}

}

if (message->text == password) {

password\_flag = true;

bot.getApi().sendMessage(CHAT\_ID, "Password is correct");

}

if (first && !password\_flag) {

bot.getApi().sendMessage(CHAT\_ID, "Password is wrong!");

}

first = true;

});

while (true) {

try {

TgBot::TgLongPoll longPoll(bot);

while (!flag) {

longPoll.start();

}

ProcessFiniteAutomat(bot);

}

catch (TgBot::TgException& e) {

printf("error: %s\n", e.what());

}

}

bot.getApi().deleteWebhook();

mosquitto\_disconnect(mosq);

mosquitto\_destroy(mosq);

mosquitto\_lib\_cleanup();

}

Содержание заголовочного файла «tgbot.cpp»:

#pragma once

#include <cstring>

#include <tgbot/tgbot.h>

#include <mosquitto.h>

#include <Windows.h>

#include "route.h"

const string BOT\_TOKEN;

const string MQTT\_TOPIC;

long long CHAT\_ID;

const string MQTT\_SERVER\_ADDRESS;

const int MQTT\_SERVER\_PORT;

struct mosquitto\* mosq;

bool flag;

bool password\_flag;

string password;

bool first;

void message(double angle, double dist);

enum class Rob\_State;

Rob\_State state;

void ProcessFiniteAutomat(TgBot::Bot& bot);

void robot\_launch();

## 3. Файлы «route.cpp» и «route.h»