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](#_Toc137473371)

[Описание 4](#_Toc137473372)

[Разработка решения 5](#_Toc137473373)

[Реализация 7](#_Toc137473374)

[1. Файл «main.cpp» 7](#_Toc137473375)

[2. Файлы «tgbot.cpp» и «tgbot.h» 7](#_Toc137473376)

[3. Файлы «route.cpp» и «route.h» 13](#_Toc137473377)

[4. Файл «CMakeLists.txt» 21](#_Toc137473378)

[5. Файл «Dockerfile» 21](#_Toc137473379)

[Вывод 24](#_Toc137473380)

# 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.

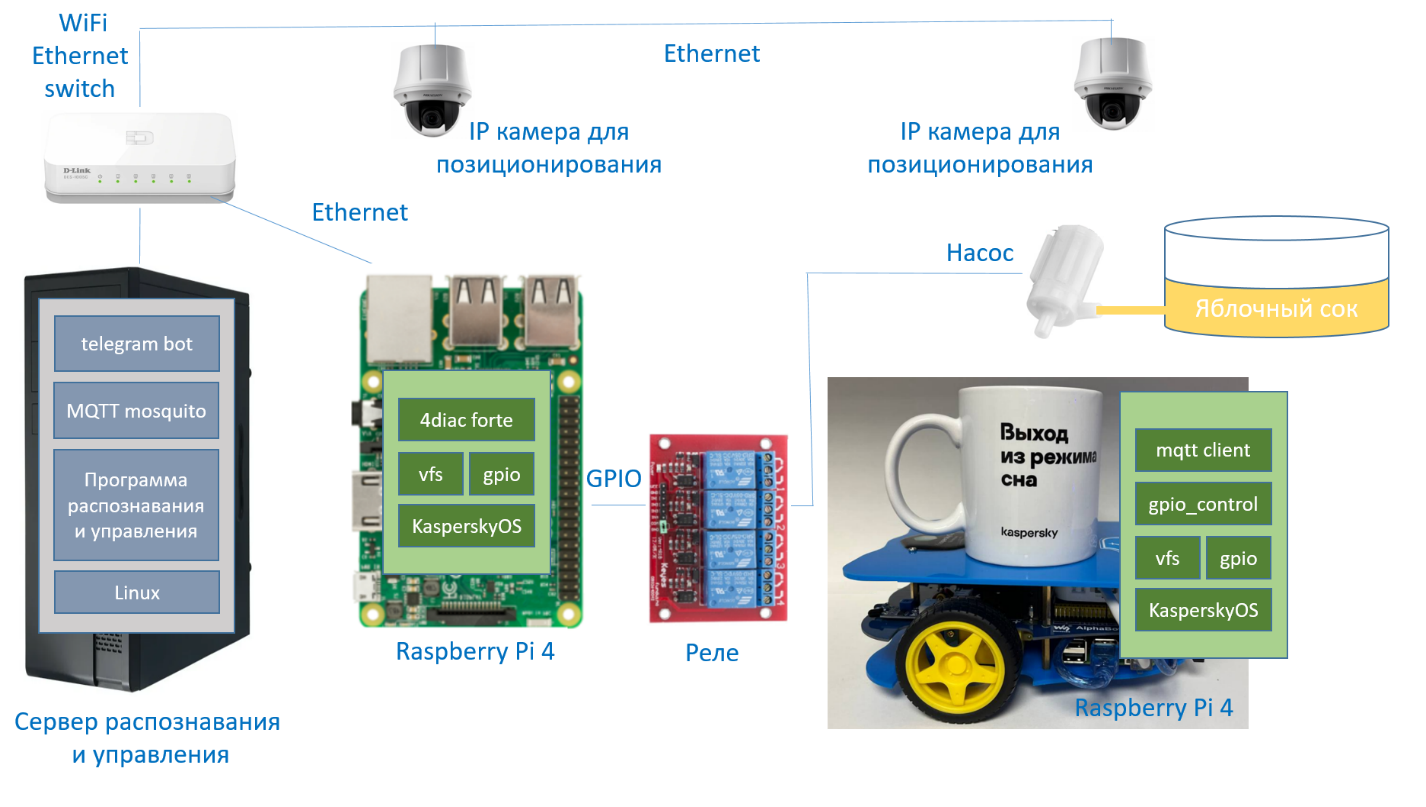


Рисунок 1 — Общая архитектура

Мы разработали диаграмму прецедентов, описывающую взаимодействие пользователя с системой (рис. 2).

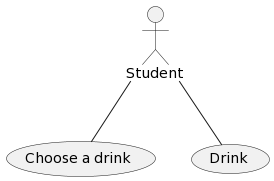


Рисунок 2 — Диаграмма прецедентов

Концептуальная модель необходима, чтобы разобраться с архитектурой системы (рис. 3).

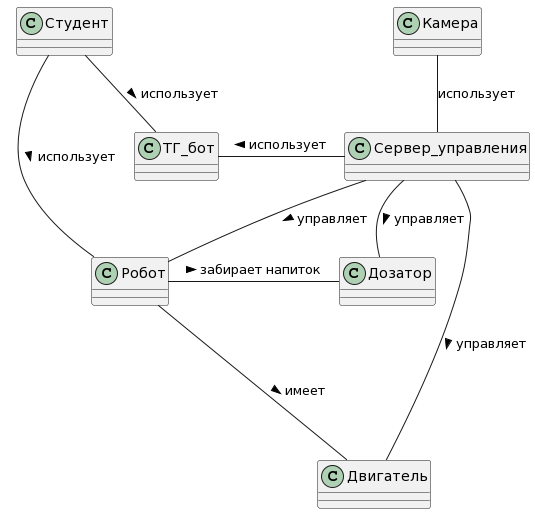


Рисунок 3 — Концептуальная модель

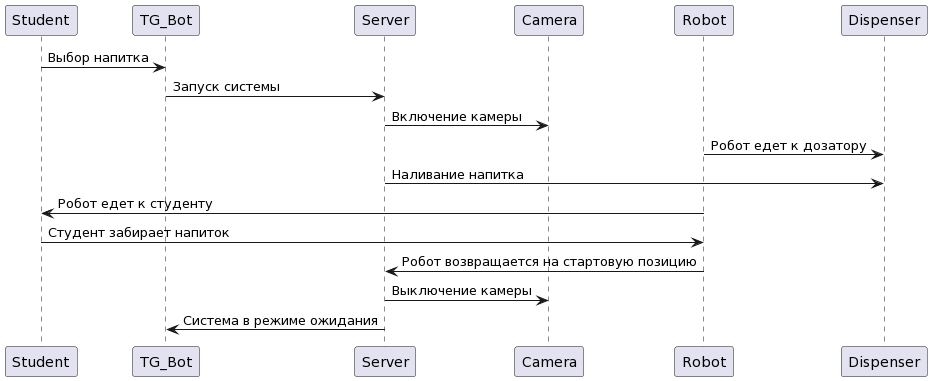
Также была разработана диаграмма последовательности, представленная на рисунке 4.

Рисунок 4 — Диаграмма последовательности

# Реализация

## 1. Файл «main.cpp»

Содержание «main.cpp»: подключение к хосту для управления роботом с указанием номера порта и IP сервера; указывание токена тг-бота, с помощью которого будет происходить управление роботом; обозначение пароля для доступа к управлению роботом.

#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» реализован брокер mosquitto, а также алгоритм работы с тг-ботом. После приветствия пользователя бот требует ввести пароль. Если пароль верен, то начинается отправка команд роботу с помощью методов библиотеки mosquito в брокер.

#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»

В файле «rout.cpp» содержится код, позволяющий обрабатывать данные с камеры видеонаблюдения в реальном времени с использованием библиотеки OpenCV. В ходе выполнения программы подсчитывается угол поворота и расстояние робота до места старта, дозатора и студента.

#include "route.h"

vector<vector<double>> coords() {

char thresholdWindow\_pink[] = "Threshold\_pink";

char thresholdWindow\_blue[] = "Threshold\_blue";

char thresholdWindow\_yellow[] = "Threshold\_yellow";

int min = 0, max = 1000;

int pink\_hmin = 157, pink\_smin = 119, pink\_vmin = 187,

pink\_hmax = 186, pink\_smax = 200, pink\_vmax = 255;

int blue\_hmin = 92, blue\_smin = 221, blue\_vmin = 156,

blue\_hmax = 118, blue\_smax = 255, blue\_vmax = 207;

int yellow\_hmin = 43, yellow\_smin = 47, yellow\_vmin = 170,

yellow\_hmax = 87, yellow\_smax = 87, yellow\_vmax = 255;

Mat frame, HSV, threshold\_pink, threshold\_blue, threshold\_yellow, blurred;

// If the input is the web camera, pass 0 instead of the video file name

VideoCapture cap(0);

// Check if camera opened successfully

if (!cap.isOpened()) {

cout << "Error opening video stream or file" << endl;

//return -1;

}

/\*цикл чтения с камеры\*/

vector <vector<double>> all\_data = {{307, 307}, {307, 307}, {307, 307}};

while (1) {

// Capture frame-by-frame

cap >> frame;

// If the frame is empty, break immediately

if (frame.empty())

break;

int Xc\_pink = 0;

int Yc\_pink = 0;

int counter\_pink = 0; // счётчик числа белых пикселей (pink)

int Xc\_blue = 0;

int Yc\_blue = 0;

int counter\_blue = 0; // счётчик числа белых пикселей (blue)

int Xc\_yellow = 0;

int Yc\_yellow = 0;

int counter\_yellow = 0; // счётчик числа белых пикселей (yellow)

int X\_dispenser = 100; // центр дозатора

int Y\_dispenser = 100;

int X\_start = 100; // центр точки старта

int Y\_start = 300;

cvtColor(frame, HSV, COLOR\_BGR2HSV);

medianBlur(HSV, blurred, 21);

inRange(blurred, Scalar(blue\_hmin, blue\_smin, blue\_vmin), Scalar(blue\_hmax, blue\_smax, blue\_vmax), threshold\_blue);

for(int y = 0; y < threshold\_blue.rows; y++){

for(int x = 0; x < threshold\_blue.cols; x++){

int value = threshold\_blue.at<uchar>(y, x);

if(value == 255){

Rect rect;

int count = floodFill(threshold\_blue, Point(x, y), Scalar(200), &rect);

Xc\_blue += x;

Yc\_blue += y;

counter\_blue++;

if(rect.width >= min && rect.width <= max

&& rect.height >= min && rect.height <= max)

{

rectangle(frame, rect, Scalar(255, 0, 0, 4));

}

}

}

}

inRange(blurred, Scalar(pink\_hmin, pink\_smin, pink\_vmin), Scalar(pink\_hmax, pink\_smax, pink\_vmax), threshold\_pink);

for(int y = 0; y < threshold\_pink.rows; y++){

for(int x = 0; x < threshold\_pink.cols; x++){

int value = threshold\_pink.at<uchar>(y, x);

if(value == 255){

Rect rect;

int count = floodFill(threshold\_pink, Point(x, y), Scalar(200), &rect);

Xc\_pink += x;

Yc\_pink += y;

counter\_pink++;

if(rect.width >= min && rect.width <= max

&& rect.height >= min && rect.height <= max)

{

rectangle(frame, rect, Scalar(255, 0, 255, 4));

}

}

}

}

inRange(blurred, Scalar(yellow\_hmin, yellow\_smin, yellow\_vmin), Scalar(yellow\_hmax, yellow\_smax, yellow\_vmax), threshold\_yellow);

for(int y = 0; y < threshold\_yellow.rows; y++){

for(int x = 0; x < threshold\_yellow.cols; x++){

int value = threshold\_yellow.at<uchar>(y, x);

if(value == 255){

Rect rect;

int count = floodFill(threshold\_yellow, Point(x, y), Scalar(200), &rect);

Xc\_yellow += x;

Yc\_yellow += y;

counter\_yellow++;

if(rect.width >= min && rect.width <= max

&& rect.height >= min && rect.height <= max)

{

rectangle(frame, rect, Scalar(255, 255, 0, 4));

}

}

}

}

if (counter\_pink != 0 && counter\_blue != 0 && counter\_yellow != 0)

{

double x1 = double(Xc\_pink) / counter\_pink - double(Xc\_blue) / counter\_blue; // Px - Bx

double y1 = double(Yc\_pink) / counter\_pink - double(Yc\_blue) / counter\_blue; // Py - By

double x2 = double(Xc\_yellow) / counter\_yellow - double(Xc\_blue) / counter\_blue; // Yx - Bx

double y2 = double(Yc\_yellow) / counter\_yellow - double(Yc\_blue) / counter\_blue; // Yy - By

double x3 = double(X\_dispenser) - double(Xc\_blue) / counter\_blue; // Dx - Bx

double y3 = double(Y\_dispenser) - double(Yc\_blue) / counter\_blue; // Dy - By

double x4 = double(X\_start) - double(Xc\_blue) / counter\_blue; // Sx - Bx

double y4 = double(Y\_start) - double(Yc\_blue) / counter\_blue; // Sy - By

double BP = sqrt(x1 \* x1 + y1 \* y1);

double BY = sqrt(x2 \* x2 + y2 \* y2);

double BD = sqrt(x3 \* x3 + y3 \* y3);

double BS = sqrt(x4 \* x4 + y4 \* y4);

double cos\_to\_puf = (x1 \* x2 + y1 \* y2) / (BY \* BP);

double cos\_to\_disp = (x1 \* x3 + y1 \* y3) / (BP \* BD);

double cos\_to\_start = (x1 \* x4 + y1 \* y4) / (BP \* BS);

double dist\_to\_puf = sqrt(pow(double(Xc\_yellow) / counter\_yellow - double(Xc\_pink) / counter\_pink, 2) +\

pow(double(Yc\_yellow) / counter\_yellow - double(Yc\_pink) / counter\_pink, 2));

double dist\_to\_disp = sqrt(pow(double(X\_dispenser)- double(Xc\_pink) / counter\_pink, 2) +\

pow(double(Y\_dispenser) - double(Yc\_pink) / counter\_pink, 2));

double dist\_to\_start = sqrt(pow(double(X\_start)- double(Xc\_pink) / counter\_pink, 2) +\

pow(double(Y\_start) - double(Yc\_pink) / counter\_pink, 2));

for (int i = 0; i < 3; ++i)

{

if (i == 0)

{

all\_data[0][0] = acos(cos\_to\_start);

all\_data[0][1] = dist\_to\_start;

} else if (i == 1)

{

all\_data[1][0] = acos(cos\_to\_disp);

all\_data[1][1] = dist\_to\_disp;

} else

{

all\_data[2][0] = acos(cos\_to\_puf);

all\_data[2][1] = dist\_to\_puf;

}

}

break;

}

// Press ESC on keyboard to exit

char c = (char)waitKey(25);

if (c == 27)

break;

}

cap.release();

destroyAllWindows();

return all\_data;

}

Содержание заголовочного файла «route.h».

#pragma once

#include <vector>

#include <iostream>

#include <math.h>

#include "opencv2/opencv.hpp"

using namespace std;

using namespace cv;

vector<vector <double>> coords();

## 4. Файл «CMakeLists.txt»

Файл «CMakeLists.txt», используемый в нашей работе, содержит инструкцию по сборке проекта для системы CMake, которая используется для автоматической генерации файлов.

cmake\_minimum\_required(VERSION 3.0)

project( robot )

find\_package(PkgConfig REQUIRED)

pkg\_check\_modules(Mosquitto IMPORTED\_TARGET libmosquitto REQUIRED)

find\_package(json-c CONFIG)

find\_package( OpenCV REQUIRED)

find\_package( TgBot REQUIRED)

include\_directories( ${OpenCV\_INCLUDE\_DIRS} ${TgBot\_INCLUDE\_DIRS} )

add\_executable( robot main.cpp route.h route.cpp tgbot.h tgbot.cpp)

target\_link\_libraries( robot PkgConfig::Mosquitto json-c::json-c ${OpenCV\_LIBS} ${TgBot\_LIBS})

## 5. Файл «Dockerfile»

В «Dockerfile» содержится инструкция для создания docker-образов, которые могут запускать в контейнере Docker.

FROM ubuntu:18.04

RUN apt update

RUN apt install -y software-properties-common

RUN add-apt-repository 'deb http://security.ubuntu.com/ubuntu xenial-security main'

RUN apt-add-repository ppa:mosquitto-dev/mosquitto-ppa

RUN apt-get update

RUN apt-get install -y cmake git libgtk2.0-dev pkg-config libavcodec-dev \

libavformat-dev libswscale-dev python-dev python-numpy libtbb2 libtbb-dev \

libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev unzip

RUN apt install -y libjson-c-dev

RUN apt install -y mosquitto-clients libmosquitto-dev

RUN apt-get install -y libcanberra-gtk-module

RUN apt-get install -y wget

RUN apt-get install -y vim

RUN cd \

&& wget https://github.com/opencv/opencv/archive/3.2.0.zip \

&& unzip 3.2.0.zip \

&& cd opencv-3.2.0 \

&& mkdir build \

&& cd build \

&& cmake .. \

&& make -j24 \

&& make install \

&& cd \

&& rm 3.2.0.zip

RUN cd \

&& wget https://github.com/opencv/opencv\_contrib/archive/3.2.0.zip \

&& unzip 3.2.0.zip \

&& cd opencv-3.2.0/build \

&& cmake -DOPENCV\_EXTRA\_MODULES\_PATH=../../opencv\_contrib-3.2.0/modules/ .. \

&& make -j24 \

&& make install \

&& cd ../.. \

&& rm 3.2.0.zip

RUN cd \

&& git clone https://github.com/oreillymedia/Learning-OpenCV-3\_examples.git \

&& cd Learning-OpenCV-3\_examples \

&& mkdir build \

&& cd build \

&& cmake .. \

&& make -j8

RUN mkdir /workspace

WORKDIR /workspace

ENV LC\_ALL C.UTF - 8

ENV LANG C.UTF - 8

ENV LANGUAGE C.UTF - 8

ENTRYPOINT ["/bin/bash", "-l"]

# Вывод

В ходе выполнения проекта по созданию робота-доставщика получен практический опыт по работе c унифицированным языком моделирования UML, кроссплатформенной системой автоматической сборки CMake, платформой контейнеризации Docker, сетевым протоколом MQTT, телеграм-ботом. Изучены и использованы такие инструменты, как брокер MQTT-сообщений mosquitto, библиотека компьютерного зрения OpenCV, ПО для построения UML-диаграмм PlantUML. Использование данных инструментов позволяет создать устройство для доставки напитков в коворкинге и получить полезные знания и ценные навыки.