ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

**Московский институт эле ктроники и математики им. А.Н. Тихонова**

Департамент электронной инженерии

**Конструкторская документация к проекту 601  
«Умный контрольно-пропускной пункт»**

Москва 2021 г.

Оглавление

[Компоненты 3](#_Toc69329151)

[Архитектура 9](#_Toc69329152)

# Компоненты

Аппаратная часть нашего проекта был выполнен на основе микроконтроллера Arduino Uno, с возможностью перехода на Arduino Nano или Arduino Mega из-за их совместимости и достаточного количества датчиков, с помощью которых и строится функционал нашей системы.

Список устройств и компонентов, используемых для создания аппаратной части нашей системы:

* Микроконтроллер Arduino Uno.

Изображение выглядит как текст, электроника, цепь

Автоматически созданное описание

Рисунок . Микроконтроллер Arduino Uno

* Изображение выглядит как электроника

  Автоматически созданное описание Датчик температуры и влажности.

Рисунок . Датчик температуры DHT11

* Светофор.

Изображение выглядит как гиря

Автоматически созданное описание

Рисунок . Светофор

* Изображение выглядит как электроника, микрофон

  Автоматически созданное описаниеДатчик расстояния  HC-SR04

Рисунок . Датчик расстояния  HC-SR04

* Датчик газа MQ-2

Изображение выглядит как внутренний, синий

Автоматически созданное описание

Рисунок .Датчик газа MQ-2

* Датчик Движения SR602

Рисунок . Датчик Движения SR602

* Сервопривод SG90

Рисунок . Сервопривод SG90

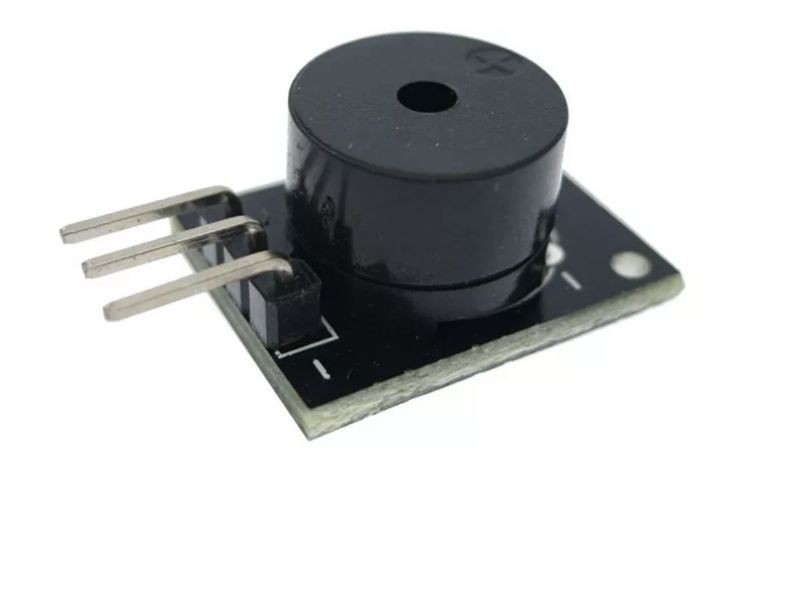
* Пассивный динамик зуммер

Рисунок . Пассивный динамик зуммер

* USB to UART преобразователь



Рисунок 9. USB to UART преобразователь

В данный список мы не включили такие позиции как адаптер питания, светодиоды, соединительные провода и тд. Т.к это расходный материал и они могут быть легко заменены без особо значимых изменений в схеме.

# Архитектура

С учетом всех выставленных требований мы построили принципиальную схему нашей системы, они приведена ниже:

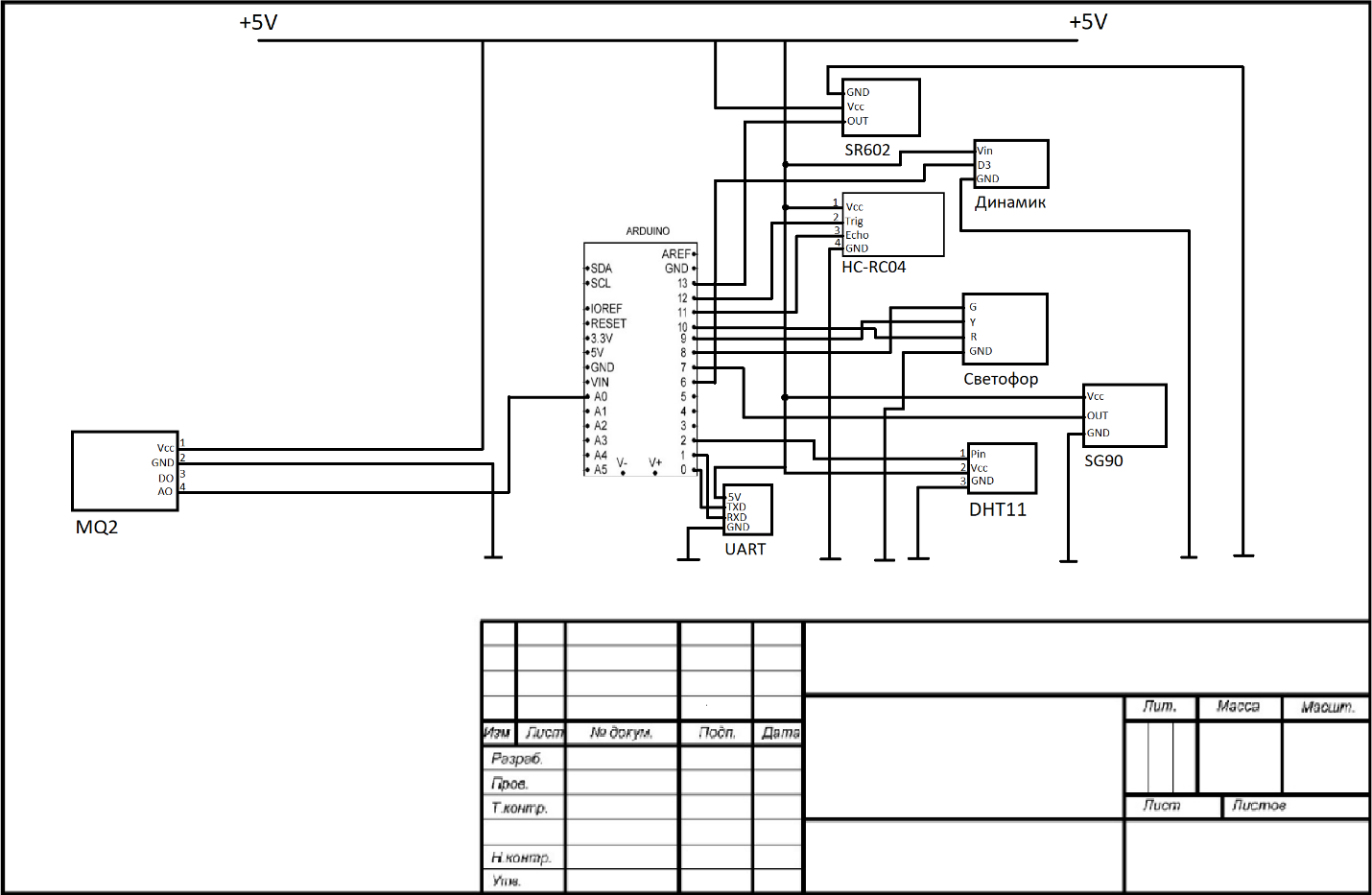


Рисунок 10. Схема электрическая принципиальная

В данной схемы мы отобразили принципы подключения всех датчиков и прочих девайсов для корректного функционирования нашей системы. При использования аналогичной платы данная схема будет тоже верна, поскольку названия портов и их функционал никак не изменятся.

Скетч Arduono UNO:

#include <DHT.h>

#define PIR 13

#define LED 6

int ledPin = 13;

byte incomingByte;

bool b=true;

#include <Servo.h>

Servo eservo;

#define PIN\_TRIG 12

#define PIN\_ECHO 11

long duration, cm;

int cm1 = 0;

int cm2 = 0;

#define smokePin A0 // определяем аналоговый выход к которому подключен датчик

int sensorThres = 40; // пороговое значение АЦП, при котором считаем что газ есть

const int LED\_RED = 10; // Порт 10, красный светодиод

const int LED\_YELLOW = 9; // Порт 9, желтый светодиод

const int LED\_GREEN = 8; // Порт 8, зеленый светодиод

int p = 6;

const int TIMEOUT\_GREEN = 3000;

DHT dht(2, DHT11); // сообщаем на каком порту будет датчик

bool ON1; //переменная сохраняет состояние светодиода

bool ON2; // для каждого светодиода своя переменная

bool ON3;

int f1();

int n=0;

void keydelay(unsigned long a)

{

unsigned long long int time = millis();

while (millis()<time+a)

{

String a="";

a=Serial.readString();

if(a == "tem"){

digitalWrite(ledPin, HIGH);

float t = dht.readTemperature();

Serial.print("Температура: ");

Serial.print(t);

}

if (a == "hum"){

digitalWrite(ledPin, HIGH);

float h = dht.readHumidity();

Serial.print("Влажность: ");

Serial.print(h);

}

if ( a == "smoke") {

digitalWrite(ledPin, HIGH);

Serial.print("Газ: ");

int analogSensor = analogRead(smokePin);

Serial.print(analogSensor);

}

if ( a == "echo") {

f1();

digitalWrite(ledPin, HIGH);

Serial.print("Расстояние: ");

Serial.print(cm);

}

if (a == "ser1"){

{

digitalWrite(LED\_RED, LOW);

keydelay(1000);

digitalWrite(LED\_YELLOW, HIGH);

keydelay(1000);

digitalWrite(LED\_YELLOW, LOW);

eservo.write(90);

digitalWrite(LED\_GREEN, HIGH);

keydelay(1000);

f1();

while (cm < 10)

{

tone (p, 500, 500);

f1();

}

}

digitalWrite(LED\_GREEN, LOW);

digitalWrite(LED\_YELLOW, HIGH);

keydelay(1000);

digitalWrite(LED\_YELLOW, LOW);

keydelay(1000);

digitalWrite(LED\_RED, HIGH);

eservo.write(10);

}

if (a == "ser2") {

digitalWrite(LED\_RED, HIGH);

digitalWrite(LED\_YELLOW, LOW);

digitalWrite(LED\_GREEN, LOW);

eservo.write(0);

}

if (a == "ser3") {

digitalWrite(LED\_RED, LOW);

digitalWrite(LED\_YELLOW, HIGH);

digitalWrite(LED\_GREEN, LOW);

eservo.write(100);

}

if ( a == "1") {

digitalWrite(ledPin, HIGH);

}

if ( a == "move1") {

int pirVal = digitalRead(PIR);// переменная для датчика движения

if (pirVal == HIGH) {

Serial.print("1");

digitalWrite(LED, HIGH);

//delay(100);

}

else {

Serial.print("0");

digitalWrite(LED,LOW);

//delay(100);

}

}

else if (a == "0"){

digitalWrite(ledPin, LOW);

}

}

}

unsigned long VREMYA1 = millis(),VREMYA2,VREMYA3; //считаем время с начала работы контроллера

void setup() {

eservo.write(10);

pinMode(PIN\_TRIG, OUTPUT);

pinMode(PIN\_ECHO, INPUT);

pinMode(ledPin, OUTPUT);

dht.begin(); // запускаем датчик DHT11

Serial.begin(9600);

eservo.attach(7);

// Все порты светодиодов будут у нас установлены в режим "внешняя нагрузка", OUTPUT

pinMode(LED\_RED, OUTPUT);

pinMode(LED\_YELLOW, OUTPUT);

pinMode(LED\_GREEN, OUTPUT);

// Устанавливаем начальное значение светодиодов

digitalWrite(LED\_RED, HIGH);

digitalWrite(LED\_YELLOW, LOW);

digitalWrite(LED\_GREEN, LOW);

pinMode(PIR, INPUT);

pinMode(LED,OUTPUT);

keydelay(1000000);

}

void loop() {

f1();

int pirVal = digitalRead(PIR);// переменная для датчика движения

int analogSensor = analogRead(smokePin);

//Serial.print (analogSensor);

float h = rand();//dht.readHumidity();

float t = rand();//dht.readTemperature();

}

int f1();

int f1() {

digitalWrite(PIN\_TRIG, LOW);

delayMicroseconds(5);

digitalWrite(PIN\_TRIG, HIGH);

// Выставив высокий уровень сигнала, ждем около 10 микросекунд. В этот момент датчик будет посылать сигналы с частотой 40 КГц.

delayMicroseconds(10);

digitalWrite(PIN\_TRIG, LOW);

// Время задержки акустического сигнала на эхолокаторе.

duration = pulseIn(PIN\_ECHO, HIGH);

// Теперь осталось преобразовать время в расстояние

cm = (duration / 2) / 29.1;

return cm;

}

Исходный код программы:

/\*

параметры конфигурации

\*/

#define admin\_login "admin" //логин для входа

#define admin\_password "nimda"//пароль для входа

#define uart\_port "/dev/ttyUSB0"//порт uart

#define image\_directory "Images/"

#define database\_name "database.csv"

//стрим высокого разрешения для распознавания номеров

#define main\_stream\_url "rtsp://admin:180500Kn@172.18.18.3:554/live/0/MAIN"

//стрим 800\*448 10FPS

#define sub\_stream\_url "rtsp://admin:180500Kn@172.18.18.3:554/live/0/SUB"

/\*

библиотеки

\*/

//подключаем gtk3

#include <gtk/gtk.h>

#include <gdk/gdk.h>

#include <glib.h>

//подключаем opencv

#include <opencv2/core/core.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/imgproc/imgproc.hpp>

#include <opencv2/objdetect/objdetect.hpp>

#include <opencv2/core/core.hpp>

//просто нужные библиотеки

#include <cstring>

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <chrono>

#include <fstream>

#include <vector>

// подключаем библиотеки для uart

#include <termios.h>

#include <errno.h>

#include <unistd.h>

#include <fcntl.h>

//теперь приложение многопоточное

#include <thread>

//including tesseract

#include <tesseract/baseapi.h>

#include <leptonica/allheaders.h>

/\*

Глобальные переменные

\*/

//виджеты первого окна

GtkWidget\* window;

GtkWidget\* fixed1;

GtkWidget\* button1\_1;

GtkWidget\* button1\_2;

GtkWidget\* login\_entry;

GtkWidget\* password\_entry;

GtkWidget\* pass\_incorrect;

//виджеты второго окна

GtkWidget\* window2;

GtkWidget\* fixed2;

GtkWidget\* button2\_1;

GtkWidget\* button2\_2;

GtkWidget\* button2\_3;

GtkWidget\* button2\_4;

GtkWidget\* button2\_5;

GtkWidget\* button2\_6;

GtkWidget\* entry2\_1;

GtkWidget\* toggle1;

GtkWidget\* toggle2\_1;

GtkWidget\* toggle2\_2;

GtkWidget\* video1;

GtkWidget\* photo;

GtkWidget\* text2\_1;

GtkWidget\* text\_number;

GtkWidget\* text\_familia;

GtkWidget\* text\_imya;

GtkWidget\* text\_otchestvo;

GtkWidget\* text\_dolzhnost;

GtkWidget\* text\_avto;

GtkWidget\* text\_tsvet\_avto;

GtkWidget\* text\_other;

//какая-то неведомая поебень

GtkBuilder\* builder;

GdkPixbuf\* ccc;

//служебные переменныые

int serial\_port;

bool o=true;

bool toggle2\_1\_on = false;

bool toggle2\_2\_on = false;

char\* sensor\_data;

tesseract::TessBaseAPI\* OCR;

/\*

прототипы функций

\*/

//функции кнопок

void on\_button1\_1\_clicked(GtkButton \*button, gpointer user\_data);

void on\_button1\_2\_clicked(GtkButton \*button, gpointer user\_data);

void on\_button3\_clicked(GtkButton \*button, gpointer user\_data);

void on\_button26\_clicked(GtkButton \*button, gpointer user\_data);

void toggle1\_toggled(GtkButton \*button, gpointer user\_data);

void toggle2\_1\_toggled(GtkButton \*button, gpointer user\_data);

void toggle2\_2\_toggled(GtkButton \*button, gpointer user\_data);

//функции для uart

int set\_interface\_attribs (int fd, int speed, int parity);

void set\_blocking (int fd, int should\_block);

bool stream();

void request\_sensors();

bool recognizer();

void request\_data\_from\_sensor (char\* sensor, int length);

bool update\_sensor\_widget (GtkWidget\* widget);

inline int getRandomNumber(int min, int max)

{

static const double fraction = 1.0 / (static\_cast<double>(RAND\_MAX) + 1.0);

return static\_cast<int>(rand() \* fraction \* (max - min + 1) + min);

}

bool update\_stream(GdkPixbuf\* frame)

{

gtk\_image\_set\_from\_pixbuf(GTK\_IMAGE(video1),frame);

return FALSE;

}

struct record

{

std::string nomer;

std::string familia;

std::string imya;

std::string otchestvo;

std::string dolzhnost;

std::string avto;

std::string tsvet\_avto;

std::string other;

std::string filename;

};

class database

{

public:

record\* data;

unsigned int length;

void parse (std::string name)

{

std::fstream fin;

fin.open(name);

std::string line;

fin >> line;

length = stoi(line);

data = new record [length];

fin >> line;

int pos;

for (unsigned int i = 0; i<length; i++) {

fin >> line;

pos = line.find(";");

data[i].nomer=line.substr(0,pos);

line.erase(0,pos+1);

pos = line.find(";");

data[i].familia=line.substr(0,pos);

line.erase(0,pos+1);

pos = line.find(";");

data[i].imya=line.substr(0,pos);

line.erase(0,pos+1);

pos = line.find(";");

data[i].otchestvo=line.substr(0,pos);

line.erase(0,pos+1);

pos = line.find(";");

data[i].dolzhnost=line.substr(0,pos);

line.erase(0,pos+1);

pos = line.find(";");

data[i].avto=line.substr(0,pos);

line.erase(0,pos+1);

pos = line.find(";");

data[i].tsvet\_avto=line.substr(0,pos);

line.erase(0,pos+1);

pos = line.find(";");

data[i].other=line.substr(0,pos);

line.erase(0,pos+1);

pos = line.find(";");

data[i].filename=image\_directory+line.substr(0,pos);

line.erase(0,pos+1);

}

fin.close();

}

record search (std::string text)

{

for (unsigned long long int i = 0; i<length; i++)

{

if (data[i].nomer==text) return data[i];

}

record none;

none.nomer="";

return none;

}

};

database dbase;

bool draw\_responsed\_data ();

record wtf;

int main (int argc, char\* argv[])

{

gtk\_init(&argc, &argv);

srand(time(0));

sensor\_data = new char [50];

dbase.parse(database\_name);

wtf.nomer="";

OCR = new tesseract::TessBaseAPI();

OCR[0].Init(NULL, "eng");

OCR[0].SetVariable("tessedit\_char\_whitelist","abekmhopctyxABEKMHOPCTYX0123456789");

OCR[0].SetVariable("user\_defined\_dpi", "100");

builder = gtk\_builder\_new\_from\_file("Resources/app.glade");

//окно входа

window = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "window"));

fixed1 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "fixed1"));

button1\_1 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "button1\_1"));

button1\_2 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "button1\_2"));

login\_entry = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "login\_entry"));

password\_entry = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "password\_entry"));

pass\_incorrect = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "pass\_incorrect"));

//основное окно

window2 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "main\_window"));

fixed2 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "fixed2"));

button2\_1 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "button2\_1"));

button2\_2 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "button2\_2"));

button2\_3 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "button2\_3"));

button2\_4 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "button2\_4"));

button2\_5 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "button2\_5"));

button2\_6 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "button2\_6"));

text2\_1 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "text2\_1"));

text\_number = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "text\_number"));

text\_familia = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "text\_familia"));

text\_imya = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "text\_imya"));

text\_otchestvo = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "text\_otchestvo"));

text\_dolzhnost = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "text\_dolzhnost"));

text\_avto = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "text\_avto"));

text\_tsvet\_avto = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "text\_tsvet\_avto"));

text\_other = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "text\_other"));

entry2\_1 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "entry2\_1"));

toggle1 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "toggle1"));

toggle2\_1 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "toggle2\_1"));

toggle2\_2 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "toggle2\_2"));

video1 = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "video1"));

photo = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "photo"));

//подсоединяем функции кнопок

g\_signal\_connect(window, "destroy", G\_CALLBACK(gtk\_main\_quit), NULL);

g\_signal\_connect(window2, "destroy", G\_CALLBACK(gtk\_main\_quit), NULL);

g\_signal\_connect(toggle1, "toggled", G\_CALLBACK(toggle1\_toggled), NULL);

g\_signal\_connect(toggle2\_1, "toggled", G\_CALLBACK(toggle2\_1\_toggled), NULL);

g\_signal\_connect(toggle2\_2, "toggled", G\_CALLBACK(toggle2\_2\_toggled), NULL);

g\_signal\_connect(GTK\_BUTTON(button2\_1), "clicked", G\_CALLBACK(on\_button3\_clicked), NULL);

g\_signal\_connect(GTK\_BUTTON(button2\_6), "clicked", G\_CALLBACK(on\_button26\_clicked), NULL);

g\_signal\_connect(GTK\_BUTTON(button1\_2), "clicked", G\_CALLBACK(on\_button1\_2\_clicked), NULL);

g\_signal\_connect(GTK\_BUTTON(button1\_1), "clicked", G\_CALLBACK(on\_button1\_1\_clicked), NULL);

serial\_port = open(uart\_port, O\_RDWR | O\_NOCTTY | O\_SYNC);

if (serial\_port < 0)

{

std::cout<<"Error from open:"<<errno<<std::endl<<strerror(errno)<<std::endl;

}

set\_interface\_attribs (serial\_port, B9600, 0);

set\_blocking (serial\_port, 0);

gtk\_widget\_show(window);

std::thread thread1(stream);

std::thread thread2(request\_sensors);

std::thread thread3(recognizer);

thread1.detach();

thread2.detach();

thread3.detach();

gtk\_main();

close(serial\_port);

return 0;

}

inline int set\_interface\_attribs (int fd, int speed, int parity)

{

struct termios tty;

if (tcgetattr (fd, &tty) != 0)

{

std::cout<<"error from tcgetattr"<<errno;

return -1;

}

cfsetospeed (&tty, speed);

cfsetispeed (&tty, speed);

tty.c\_cflag = (tty.c\_cflag & ~CSIZE) | CS8; // 8-bit chars

// disable IGNBRK for mismatched speed tests; otherwise receive break

// as \000 chars

tty.c\_iflag &= ~IGNBRK; // disable break processing

tty.c\_lflag = 0; // no signaling chars, no echo,

// no canonical processing

tty.c\_oflag = 0; // no remapping, no delays

tty.c\_cc[VMIN] = 0; // read doesn't block

tty.c\_cc[VTIME] = 5; // 0.5 seconds read timeout

tty.c\_iflag &= ~(IXON | IXOFF | IXANY); // shut off xon/xoff ctrl

tty.c\_cflag |= (CLOCAL | CREAD);// ignore modem controls,

// enable reading

tty.c\_cflag &= ~(PARENB | PARODD); // shut off parity

tty.c\_cflag |= parity;

tty.c\_cflag &= ~CSTOPB;

tty.c\_cflag &= ~CRTSCTS;

if (tcsetattr (fd, TCSANOW, &tty) != 0)

{

std::cout<<"error from tcsetattr"<<errno<<std::endl;

return -1;

}

return 0;

}

inline void set\_blocking (int fd, int should\_block)

{

struct termios tty;

memset (&tty, 0, sizeof tty);

if (tcgetattr (fd, &tty) != 0)

{

std::cout<<"error from tggetattr"<<errno<<std::endl;

return;

}

tty.c\_cc[VMIN] = should\_block ? 1 : 0;

tty.c\_cc[VTIME] = 5; // 0.5 seconds read timeout

if (tcsetattr (fd, TCSANOW, &tty) != 0)

std::cout<<"error setting term attributes"<<errno<<std::endl;

}

void on\_button1\_1\_clicked(GtkButton \*button, gpointer user\_data)

{

std::string login;

const char\* log = gtk\_entry\_get\_text(GTK\_ENTRY((GtkWidget\*) login\_entry));

std::string password;

const char\* pas = gtk\_entry\_get\_text(GTK\_ENTRY((GtkWidget\*) password\_entry));

guint16 len = gtk\_entry\_get\_text\_length(GTK\_ENTRY((GtkWidget\*) login\_entry));

login.assign(log, len);

len = gtk\_entry\_get\_text\_length(GTK\_ENTRY((GtkWidget\*) password\_entry));

password.assign(pas, len);

if ((login == admin\_login)&&(password==admin\_password))

{

gtk\_widget\_destroy(window);

gtk\_widget\_show(window2);

gtk\_main();

}

else gtk\_label\_set\_text(GTK\_LABEL(pass\_incorrect), "Неправильный логин/пароль!");

}

void on\_button1\_2\_clicked(GtkButton \*button, gpointer user\_data)

{

gtk\_entry\_set\_text(GTK\_ENTRY(login\_entry), "");

gtk\_entry\_set\_text(GTK\_ENTRY(password\_entry), "");

}

void on\_button3\_clicked (GtkButton \*button, gpointer user\_data)

{

write (serial\_port, "ser1", 4);

}

void toggle1\_toggled(GtkButton \*button, gpointer user\_data)

{

std::string msg;

if (o) msg="ser2";

else msg="ser3";

write (serial\_port,msg.c\_str(), 100);

o=!o;

}

void toggle2\_1\_toggled(GtkButton \*button, gpointer user\_data)

{

toggle2\_1\_on = !toggle2\_1\_on;

if ((!toggle2\_1\_on)&&(toggle2\_2\_on))

{

gtk\_toggle\_button\_set\_active(GTK\_TOGGLE\_BUTTON(toggle2\_2), FALSE);

}

}

void toggle2\_2\_toggled(GtkButton \*button, gpointer user\_data)

{

toggle2\_2\_on = !toggle2\_2\_on;

if ((!toggle2\_1\_on)&&(toggle2\_2\_on))

{

gtk\_toggle\_button\_set\_active(GTK\_TOGGLE\_BUTTON(toggle2\_1), TRUE);

}

}

bool stream()

{

GdkPixbuf\* pix;

unsigned long long int a=0;

std::string url="rtsp://admin:180500Kn@172.18.18.3:554/live/0/SUB";

cv::VideoCapture cap(url);

if (!cap.isOpened()) {std::cout<<"No video";return 1;}

cv::Mat frame;

while (true) {

cap>>frame;

cvtColor(frame, frame, cv::COLOR\_BGR2RGBA);

pix = gdk\_pixbuf\_new\_from\_data(frame.data, GDK\_COLORSPACE\_RGB, true, 8, frame.cols, frame.rows, frame.step, NULL, NULL);

g\_idle\_add(GSourceFunc(update\_stream), pix);

usleep(9000);

}

cap.release();

return TRUE;

}

void request\_sensors ()

{

//echo каждые 10 сек

//tem каждые 90 сек

//hum каждые 30 сек

//газ каждые 60 сек

auto t1 = std::chrono::high\_resolution\_clock::now();

auto t2 = t1;

bool update\_needed = true;

int last\_updated\_interval=0;

while (true)

{

t2=std::chrono::high\_resolution\_clock::now();

int duration = std::chrono::duration\_cast<std::chrono::seconds>(t2 - t1).count();

if (duration>90)

{

t1 = std::chrono::high\_resolution\_clock::now();

t2 = t1;

update\_needed = true;

last\_updated\_interval=0;

continue;

}

if (duration/10>last\_updated\_interval)

{

last\_updated\_interval++;

update\_needed=true;

}

if (update\_needed)

{

switch (duration/10)

{

case 0:

break;

case 3:

{

request\_data\_from\_sensor("hum", 3);

g\_idle\_add(GSourceFunc(update\_sensor\_widget), button2\_3);

update\_needed = false;

break;

}

case 6:

{

request\_data\_from\_sensor("smoke", 5);

g\_idle\_add(GSourceFunc(update\_sensor\_widget), button2\_4);

update\_needed = false;

break;

}

case 9:

{

request\_data\_from\_sensor("tem", 3);

g\_idle\_add(GSourceFunc(update\_sensor\_widget), button2\_2);

update\_needed = false;

break;

}

default:

{

request\_data\_from\_sensor("echo", 4);

g\_idle\_add(GSourceFunc(update\_sensor\_widget), button2\_5);

update\_needed = false;

break;

}

}

}

}

}

void request\_data\_from\_sensor (char\* sensor, int length)

{

int n = 0;

sensor\_data = new char [50];

for (int i=0; i<50; i++)

{

sensor\_data[i]='\0';

}

auto t1 = std::chrono::high\_resolution\_clock::now();

auto t2 = t1;

write (serial\_port, sensor, length);

usleep ((7 + 25) \* 100);

while (n<=0)

{

n = read (serial\_port, sensor\_data, 50);

t2 = std::chrono::high\_resolution\_clock::now();

int duration = std::chrono::duration\_cast<std::chrono::seconds>(t2 - t1).count();

if (duration>5)

{

break;

}

}

}

bool update\_sensor\_widget (GtkWidget\* widget)

{

gtk\_button\_set\_label(GTK\_BUTTON(widget), sensor\_data);

return FALSE;

}

bool recognizer()

{

std::string last\_number = "123";

cv::Mat source\_img;

cv::Mat gray;

cv::CascadeClassifier cascadePlate;

cascadePlate.load("haarcascade\_russian\_plate\_number.xml");

cv::VideoCapture cap;

while(true)

{

if (toggle2\_1\_on)

{

cap.open(main\_stream\_url);

cap>>source\_img;

if (!source\_img.empty())

{

cv::cvtColor(source\_img, gray, cv::COLOR\_BGR2GRAY);

threshold(gray, gray, 0, 255, cv::THRESH\_OTSU|cv::THRESH\_BINARY);

std::vector<cv::Rect> plate;

cascadePlate.detectMultiScale(gray, plate);

cv::Point rbeg;

cv::Point rend;

cv::Mat cropped;

for (auto& i:plate)

{

rbeg = cv::Point(i.x, i.y);

rend = cv::Point(i.x+i.width, i.y+i.height);

cv::Mat ROI(source\_img, cv::Rect(i.x, i.y,i.width, i.height));

ROI.copyTo(cropped);

imwrite("tess.jpg", cropped);

Pix\* pix = pixRead("tess.jpg");

OCR[0].SetImage(pix);

std::string ab = OCR[0].GetUTF8Text();

for (int i=0;i<ab.size();i++)

{

ab[i]=toupper(ab[i]);

if (ab[i]=='\n') ab.erase(i,1);

}

//std::cout<<ab<<'\n';

if ((ab.size()==8)||(ab.size()==9))

{

wtf = dbase.search(ab);

if ((wtf.nomer!="")&&(wtf.nomer!=last\_number))

{

g\_idle\_add(GSourceFunc(draw\_responsed\_data), NULL);

sleep(1);

if (toggle2\_2\_on)

{

write (serial\_port, "ser1", 4);

usleep ((7 + 25) \* 100);

}

}

}

}

}

cap.release();

}

}

return 0;

}

bool draw\_responsed\_data ()

{

gtk\_entry\_set\_text(GTK\_ENTRY(text\_number), ("Номер: "+wtf.nomer).c\_str());

gtk\_entry\_set\_text(GTK\_ENTRY(text\_familia), ("Фамилия: "+wtf.familia).c\_str());

gtk\_entry\_set\_text(GTK\_ENTRY(text\_imya), ("Имя: "+wtf.imya).c\_str());

gtk\_entry\_set\_text(GTK\_ENTRY(text\_otchestvo), ("Отчество: "+wtf.otchestvo).c\_str());

gtk\_entry\_set\_text(GTK\_ENTRY(text\_dolzhnost), ("Должность: "+wtf.dolzhnost).c\_str());

gtk\_entry\_set\_text(GTK\_ENTRY(text\_avto), ("Автомобиль: "+wtf.avto).c\_str());

gtk\_entry\_set\_text(GTK\_ENTRY(text\_tsvet\_avto), ("Цвет авто: "+wtf.tsvet\_avto).c\_str());

gtk\_entry\_set\_text(GTK\_ENTRY(text\_other), ("Другое: "+wtf.other).c\_str());

cv::Mat frame = cv::imread(wtf.filename);

if (frame.empty()) return 0;

cv::resize(frame, frame, cv::Size(300,400));

GdkPixbuf\* pix;

cvtColor(frame, frame, cv::COLOR\_BGR2RGBA);

pix = gdk\_pixbuf\_new\_from\_data(frame.data, GDK\_COLORSPACE\_RGB, true, 8, frame.cols, frame.rows, frame.step, NULL, NULL);

gtk\_image\_set\_from\_pixbuf(GTK\_IMAGE(photo),pix);

wtf.nomer="";

return FALSE;

}

void on\_button26\_clicked(GtkButton \*button, gpointer user\_data)

{

std::string search\_number = std::string(gtk\_entry\_get\_text(GTK\_ENTRY(entry2\_1)));

record current;

current = dbase.search(search\_number);

gtk\_entry\_set\_text(GTK\_ENTRY(text\_number), ("Номер: "+current.nomer).c\_str());

gtk\_entry\_set\_text(GTK\_ENTRY(text\_familia), ("Фамилия: "+current.familia).c\_str());

gtk\_entry\_set\_text(GTK\_ENTRY(text\_imya), ("Имя: "+current.imya).c\_str());

gtk\_entry\_set\_text(GTK\_ENTRY(text\_otchestvo), ("Отчество: "+current.otchestvo).c\_str());

gtk\_entry\_set\_text(GTK\_ENTRY(text\_dolzhnost), ("Должность: "+current.dolzhnost).c\_str());

gtk\_entry\_set\_text(GTK\_ENTRY(text\_avto), ("Автомобиль: "+current.avto).c\_str());

gtk\_entry\_set\_text(GTK\_ENTRY(text\_tsvet\_avto), ("Цвет авто: "+current.tsvet\_avto).c\_str());

gtk\_entry\_set\_text(GTK\_ENTRY(text\_other), ("Другое: "+current.other).c\_str());

cv::Mat frame = cv::imread(current.filename);

GdkPixbuf\* pix;

if (frame.empty()) return;

cv::resize(frame, frame, cv::Size(300,400));

cvtColor(frame, frame, cv::COLOR\_BGR2RGBA);

pix = gdk\_pixbuf\_new\_from\_data(frame.data, GDK\_COLORSPACE\_RGB, true, 8, frame.cols, frame.rows, frame.step, NULL, NULL);

gtk\_image\_set\_from\_pixbuf(GTK\_IMAGE(photo),pix);

}