

به نام خدا



دانشکده مهندسی برق

پروژه نهایی

درس سیستم‌های نهفته بی درنگ

محمد امین حاجی خداوردیان

۹۷۱۰۱۵۱۸

استاد: دکتر غلامپور

نیمسال دوم ۱۴۰۰-۱۴۰۱

پروژه را برای راحتی توضیح و توصیف آن به ۴ بخش جدا می‌کنیم و هرکدام را در بخش‌های متناسب با خودش توضیح می‌دهیم.

بخش اول: فرستادن اطلاعات با استفاده از MQTT

برای اینکار با استفاده از تمرین ۶ و کدی که در اختیار ما قرار داده شد بخش تشخیص چهره را استفاده کردیم. برای بخش تشخیص صوت با استفاده از کتابخانه **Alsa** که برای صوت است با جستجو در اینترنت مواردی را یافتیم که در کد از آن‌ها استفاده کردیم. برای لود **Cpu** با استفاده از فایل موجود در **/proc/stat** استفاده می‌کنیم. برای دما نیز باید به مسیر **/sys/class/thermal/zone0/temp** برویم که دما را به صورت میلی سانتی گراد برمی‌گرداند ولی این مسیر در سیستم بنده موجود نبود. با جست و جویی که کردم به علت استفاده از **Vmware** (ماشین مجازی) لینوکس دسترسی به تمامی سنسورها را ندارد و به همین دلیل دستور **lm-sensors** کارایی ندارد و برای این بخش از عدد رندوم استفاده شده است ولی کد درست آن نیز گذاشته شده است که در صورت اجرا در سیستم مناسب بتوان آن را از حالت کامنت خارج کرد و از آن استفاده کرد. حال به توضیح کد و بخش‌های مختلف آن می‌پردازیم:

```
35 void publish(MQTTClient client, char* topic, char* payload) {
36     MQTTClient_message pubmsg = MQTTClient_message_initializer;
37     pubmsg.payload = payload;
38     pubmsg.payloadlen = strlen((char*)pubmsg.payload);
39     pubmsg.qos = 0;
40     pubmsg.retained = 0;
41     MQTTClient_deliveryToken token;
42     MQTTClient_publishMessage(client, topic, &pubmsg, &token);
43     MQTTClient_waitForCompletion(client, token, 1000L);
44     printf("Message '%s' with delivery token %d delivered\n", payload, token);
45 }
46
47
48 int on_message(void *context, char *topicName, int topicLen, MQTTClient_message *message) {
49     char* payload = (char*)message->payload;
50     counter++;
51     printf("%d)Received operation %s\n", counter, payload);
52     MQTTClient_freeMessage(&message);
53     MQTTClient_free(topicName);
54     return 1;
55 }
```

بخش بالا برای پابلیش کردن پیام با استفاده از MQTT است که QOS آن را ۰ در نظر گرفته‌ایم. این بخش با توجه به مطالب کلاس و فایل‌های موجود زده شده است و توضیح خاصی درباره آن نیست.

```
58 static char *device = "default";
59 short buffer[8*1024];
60 int buffer_size = sizeof(buffer) >> 1;
61 double RMS(short *buffer) {
62     int i;
63     long int SquareSum = 0.0;
64     for(i=0; i<buffer_size; i++)
65         SquareSum += (buffer[i] * buffer[i]);
66
67     double rms = sqrt(SquareSum/buffer_size);
68     return rms;
69 }
70
71
72 int detectAndDisplay( Mat frame );
73 CascadeClassifier face_cascade;
```

تصویر صفحه قبل تابعی است که برای گرفتن RMS از ورودی صوت استفاده شده است. با جست و جوی های بنده با استفاده از **Alsa** یک رشته از ورودی های صوت گرفته می شود و ما آن را تک تک مقادیرش را به توان ۲ می-کنیم و با هم جمع می کنیم و در نهایت بر کل سائز صوت تقسیم می کنیم که یک معیاری برای سنجش صوت داشته باشیم.

```

75 int main( )
76 {
77     cv::VideoCapture camera(0);
78     if (!camera.isOpened()) {
79         std::cerr << "ERROR: Could not open camera" << std::endl;
80         return 1;
81     }
82
83     if( !face_cascade.load( "/home/amlnh7325/EnbSys/OpenCV-20220619/OpenCV/opencv/data/haarcascades/haarcascade_frontalcatface.xml" ) )
84     {
85         cout << "--(1)Error loading face cascade\n";
86         return -1;
87     };
88
89     FILE *fp;
90     char St[100];
91     char StTemp[100];
92     char StLoad[100];
93     char StLoadf[100];
94     time_t now = time(0);
95     char* DateTime = ctime(&now);
96     int Cpu0, Cpu1, Cpu2, Idle, Load;
97
98     snd_pcm_t *handle_capture;
99     snd_pcm_sframes_t frames;
100     char StAudio[200];
101     char Temp[100];
102
103     MQTTClient client;
104     MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
105     MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
106     conn_opts.username = "amlnh7325";
107     conn_opts.password = "Amlnhajl";
108
109     MQTTClient_setCallbacks(client, NULL, NULL, on_message, NULL);
110
111

```

در بخش بالا تنها مقادیر و متغیرها تعریف شده است.

```

112     namedWindow("Face");
113     Mat frame;
114     int temp;
115     int NumberofFaces;
116     int OldNumber=0;
117     int rc;
118     int Err;
119
120     Err = snd_pcm_open(&handle_capture, device, SND_PCM_STREAM_CAPTURE, 0);
121     if(Err < 0) {
122         printf("Capture open error: %s\n", snd_strerror(Err));
123         exit(EXIT_FAILURE);
124     }
125
126     Err = snd_pcm_set_params(handle_capture,
127                             SND_PCM_FORMAT_S16_LE,
128                             SND_PCM_ACCESS_RW_INTERLEAVED,
129                             1,
130                             48000,
131                             1,
132                             500000);
133     if(Err < 0) {
134         printf("Capture open error: %s\n", snd_strerror(Err));
135         exit(EXIT_FAILURE);
136     }
137
138     double k = 0.45255;
139     double PValue = 0;
140     double peak = 0;

```

این بخش برای صوت آورده شده است و از تابع های **Alsa** استفاده شده است که با جستجو در اینترنت به آن رسیدیم.

حال به سراغ بخش های مختلفی که در صورت پروژه نیاز به ارسال آن بود می رویم و هر کدام را با **topic** مناسب ارسال می کنیم.

```

169 //FaceDetect Publish
170 camera >> frame;
171 NumberofFaces = detectAndDisplay(frame);
172 if (NumberofFaces != OldNumber){
173     time_t now = time(0);
174     DateTime = ctime(&now);
175     sprintf(St, "%d, %s", NumberofFaces, DateTime);
176     OldNumber = NumberofFaces;
177     if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
178         printf("Failed to connect, return code %d\n", rc);
179         exit(-1);
180     }
181
182     publish(client, "sensors/Faces", St);
183     printf("Face Sensors Publish Success!\n");
184 }

```

بخش بالا برای تغییر تعداد تصویر است که با توجه به تابع detectAndDisplay تمرین ۶ اگر تعداد چهره‌ها متفاوتی در آن ایجاد شد با استفاده از time و تعداد چهره پیامی را با MQTT با topic sensors/Faces ، ارسال کنیم.

```

186 // Audio Publish
187 time_t now = time(0);
188 DateTime = ctime(&now);
189 if((int)(20 * log10(PValue)) > 35) {
190     sprintf(StAudio, "%d, %s", (int)(20 * log10(PValue)), DateTime);
191     publish(client, "sensors/Audio", StAudio);
192     printf("Audio Sensors Publish Success!\n");
193 }

```

بخش بالا برای صوت است و مقداری که برای آن در نظر گرفته ایم ۳۵ است این عدد می‌تواند با توجه به محیط و میکروفون فرد تغییر پیدا کند. در این بخش نیز با sensors/Audio.topic ارسال صورت گرفته است.

```

195 // Load and Temp publish
196 fp = fopen("/proc/stat", "r");
197 // CPU Word
198 fscanf(fp, "%s ", StLoad);
199 // First Number
200 fscanf(fp, "%s ", StLoad);
201 sscanf(StLoad, "%d", &cpu0);
202 //Second Number
203 fscanf(fp, "%s ", StLoad);
204 sscanf(StLoad, "%d", &cpu1);
205 //Third Number
206 fscanf(fp, "%s ", StLoad);
207 sscanf(StLoad, "%d", &cpu2);
208 // Idle time
209 fscanf(fp, "%s ", StLoad);
210 sscanf(StLoad, "%d", &idle);
211 fclose(fp);
212
213 Load = ((cpu0+cpu1+cpu2)/(idle+100));
214 sprintf(StLoadF, "%d", Load);
215 //int tempCPU=0;
216 //fp = fopen("/sys/class/thermal/thermal_zone0/temp", "r");
217 //fscanf(fp, "%s ", StTemp);
218 //sscanf(StTemp, "%d", &tempCPU);
219 temp = 40;
220 temp = temp + rand()/500000000;
221 sprintf(StTemp, "%d", temp);
222 publish(client, "sensors/Temp", StTemp);
223 printf("Temp Sensors Publish Success!\n");
224 publish(client, "sensors/Load", StLoadF);
225 printf("Load Sensors Publish Success!\n");
226 MQTTClient_disconnect(client, 100);
227 }
228
229
230

```

بخش آخر نیز برای لود و دمای cpu است که چون ما ۳ core را در اختیار vmware قرار داده ایم مقادیر آن را می‌خوانیم و با استفاده از رابطه‌ای که با جستوجو بدست آوردیم مقدار لود را حساب می‌کنیم. بخشی از کد بالا کامنت شده است که همانطور اشاره کردم برای حالتی است که این مسیر وجود داشته باشد و کد بر روی

vmware اجرا نشود در غیر اینصورت برای دما عدد رندوم ارسال می‌شود(ارسال عدد رندوم از taها پرسیده شد و تایید کردند) برای این بخش ها نیز از topic های، sensors/Temp و sensors/Load استفاده کرده‌ایم.

در انتها نیز کد مربوط به تابع تشخیص چهره وجود دارد که از OpenCV استفاده شده است.

این بخش به همراه کد Cmake آن در پوشه‌ای با عنوان Pub آورده شده است.

برای تست این بخش نیاز به بخش دوم داریم که در ادامه به سراغ آن می‌رویم.

بخش دوم: دریافت اطلاعات و ذخیره در پایگاه داده

در این بخش یک سابسکرایبر برای دریافت اطلاعات MQTT و ذخیره topic های صوت و تصویر در پایگاه داده توضیح داده می‌شود. از آنجایی که در پیام ارسالی MQTT زمان را دخیل کردیم و به صورت string است نیاز است که این پیام‌ها را جدا کنیم. به همین یک تابع برای تبدیل string به int زده شده است که عدد تعداد چهره و یا شدت صوت را بتواند جدا کند:

```
24 int stringtoint(const char* s){
25     int res = 0, fact = 1;
26     if (*s == '-'){
27         s++;
28         fact = -1;
29     };
30     for (int check = 0; *s; s++){
31         int d = *s - '0';
32         if (d >= 0 && d <= 9){
33             res = res * 10 + d;
34         };
35     };
36 };
37 return res * fact;
38};
```

در ادامه با توجه به , که در پیام ارسالی وجود دارد بین عدد و متن زمان جداسازی را انجام می‌دهیم. چون قرار است دو جدول جدا برای صوت و تصویر داشته باشیم دوتابع با نام‌های sendToDBFace و

sendToDBAudio ساخته شده است. ولی محتوای آن‌ها یکسان است پس به توضیح یکی از آن‌ها بسنده می‌کنیم. تصویر کد در صفحه بعدی آورده شده است. با توجه به تصویر صفحه بعد جدا سازی با توجه به ',' که گفته شد صورت می‌گیرد و سپس با یک query به MySQL اطلاعات در جدول FaceT ذخیره می‌شود. فقط به این نکته توجه داشته باشید که برای اجرای آن نیاز است که ابتدا یک پایگاه داده جدید با استفاده از query دادن به MySQL قبل از اجرای کد با عنوان ProjectDB ایجاد کنیم در غیر اینصورت کد به درستی اجرا نخواهد شد.

```

40 void sendToDBFace(char *values)
41 {
42     MYSQL *conn = mysql_init(NULL);
43     char buff[200];
44     int Integer[1];
45     char Message[27];
46     int i = 0;
47     int j = 0;
48     while(i < 2) {
49         char Mess[100];
50         int k = 0;
51         while(values[j] != ',' && values[j] != '\0') {
52             Mess[k] = values[j];
53             k++;
54             j++;
55         }
56         if(values[j] == ',') {
57             j++;
58             //
59             if(i != 1)
60                 Integer[i] = stringtoint(Mess);
61             else {
62                 for(int y = 0; y < 27; y++) {
63                     if((y == 0) || (y == 26)) {
64                         Message[y] = '\0';
65                     }
66                     else {
67                         Message[y] = Mess[y - 1];
68                     }
69                 }
70                 k = 0;
71                 i++;
72             }
73         }
74         Message[27] = '\0';
75         sprintf(buff, "INSERT INTO FaceT VALUES(%d, %d, %s)", counterFace, (int)Integer[0], Message);
76
77
78         if (mysql_real_connect(conn, "localhost", "root", "amlnh7325",
79                               "ProjectDB", 0, NULL, 0) == NULL)
80             r

```

حال در کد زیر با توجه به نوع topic پیام دریافتی ارسال به پایگاه داده را با توجه به نوع پیام تعیین می‌کنیم. اگر پیام از نوع sensors/Audio باشد از تابع sendToDBAudio استفاده می‌شود که این نوع پیام در بخش اول مشخص شد و اگر از نوع صوت باشد از تابع sendToDBFace استفاده می‌کنیم.

```

165 int on_message(void *context, char *topicName, int topicLen, MQTTClient_message *message) {
166     char* payload = message->payload;
167     int identify;
168     counter++;
169
170     time_t rawtime;
171     struct tm * timeinfo;
172     foc = fopen("log.txt", "a");
173
174     time ( &rawtime );
175     timeinfo = localtime ( &rawtime );
176     fprintf(foc, "%s : %s\n", asctime(timeinfo), payload);
177
178     printf("%d)Received operation %s\n", counter, payload);
179     identify = topicNamecropper(topicName);
180     if (strcmp(topicName , "sensors/Audio") == 0){
181         counterAudio++;
182         sendToDBAudio(payload);
183     }
184     if (strcmp(topicName , "sensors/Faces") == 0){
185         counterFace++;
186         sendToDBFace(payload);
187     }
188
189     fclose(foc);
190     MQTTClient_freeMessage(&message);
191     MQTTClient_free(topicName);
192     return 1;
193 }

```

بخش آخر کد نیز که در صفحه بعد آمده است شامل ساخت جدول‌های FaceT و AudioT است که اطلاعات را در آن‌ها ذخیره کنیم.

```

198 int main(int argc, char* argv[]) {
199     MQTTClient client;
200     MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
201     MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
202     conn_opts.username = "aminh7325";
203     conn_opts.password = "AminHajl";
204
205     MYSQL *con = mysql_init(NULL);
206     foc = fopen("log.txt", "w");
207     fprintf(foc, "Time: Faces , Temp , Load , Data&Time\n");
208     fclose(foc);
209
210     if (mysql_real_connect(con, "localhost", "root", "aminh7325",
211         "ProjectDB", 0, NULL, 0) == NULL)
212     {
213         finish_with_error(con);
214     }
215
216     if (mysql_query(con, "DROP TABLE IF EXISTS FaceT")) {
217         finish_with_error(con);
218     }
219
220     if (mysql_query(con, "CREATE TABLE FaceT(id INT, Faces INT, Date_Time TEXT)")) {
221         finish_with_error(con);
222     }
223
224     if (mysql_query(con, "DROP TABLE IF EXISTS AudioT")) {
225         finish_with_error(con);
226     }
227
228     if (mysql_query(con, "CREATE TABLE AudioT(id INT, AudioDB INT, Date_Time TEXT)")) {
229         finish_with_error(con);
230     }
231     mysql_close(con);
232
233     MQTTClient_setCallbacks(client, NULL, NULL, on_message, NULL);

```

ادامه این بخش نیز خواندن پیام از بروکر است.

حال با توجه به اینکه هر دو بخش را در اختیار داریم به سراغ تست می‌رویم.

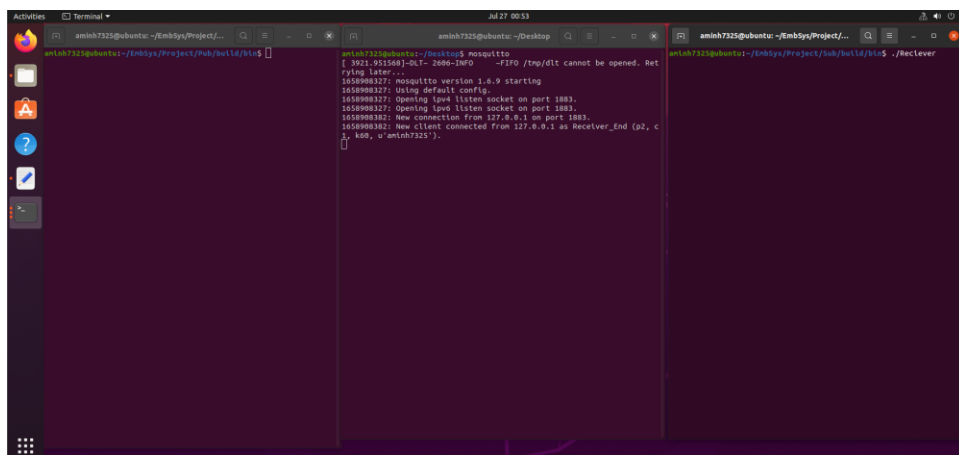
کدهای این بخش به همراه فایل cmake در پوشه‌ای با نام Sub یافت می‌شود.

ابتدا با استفاده از mosquitto یک بروکر برای ارتباط بین پابلیشر و سابسکرایبر ایجاد می‌کنیم.

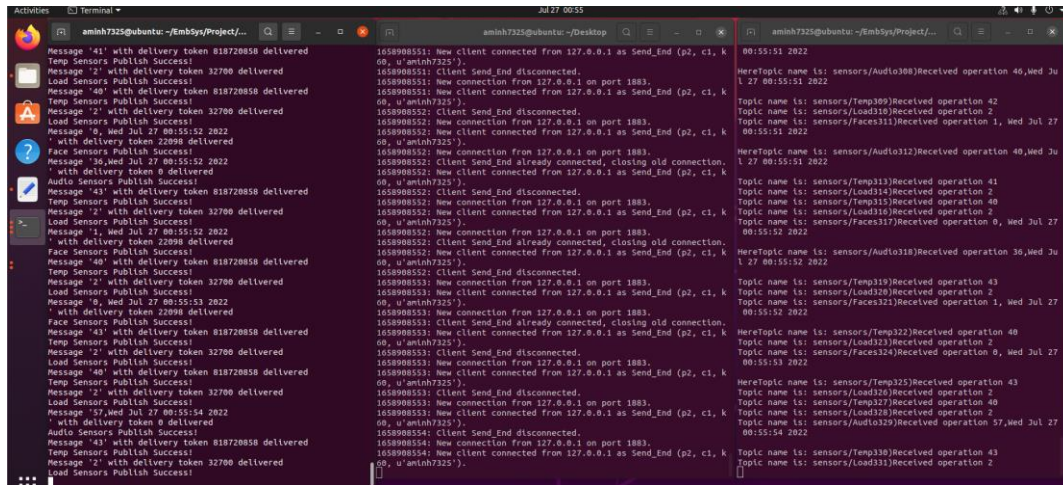
توجه: ممکن است آدرس mosquitto مورد استفاده باشد در این صورت از دستورات زیر باید استفاده کنیم.

```
ps -ef | grep mosquitto
sudo kill 12345
```

به جای عدد ۱۲۳۴۵ در دستور بالا باید عدد عملیاتی که دستور اول می‌دهد را جایگزین کنیم.

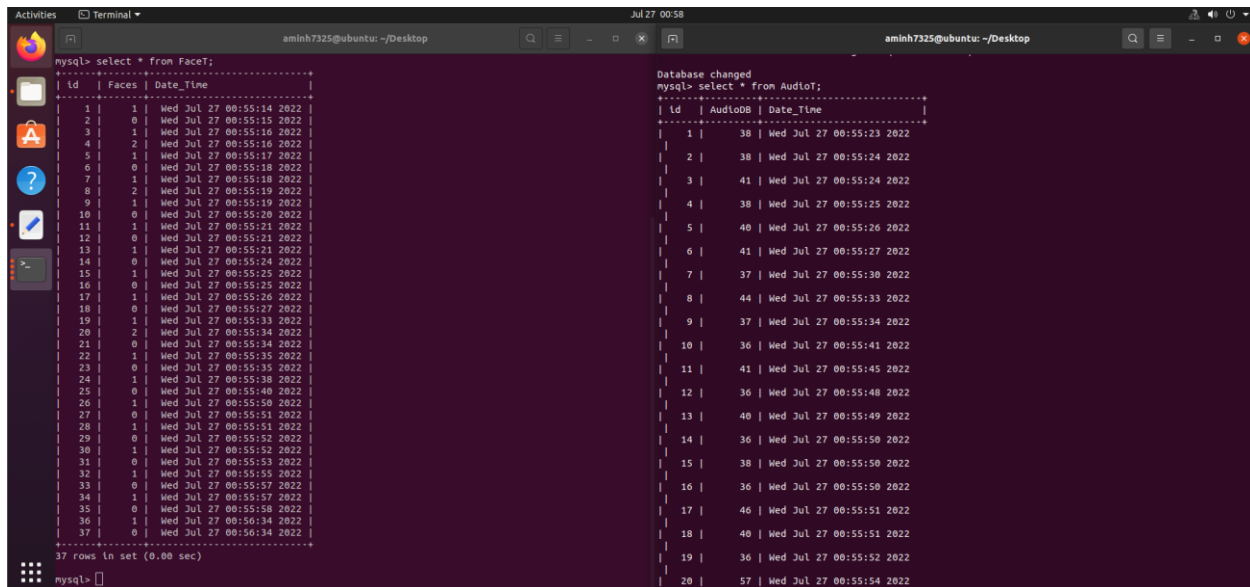


همانطور که در تصویر قبل مشاهده می کنید دریافت کننده به بروکر وصل شده است و منتظر پیام از پابلیشر است. با اجرای پابلیشر:



The image shows three terminal windows. The left window displays MQTT broker logs with messages like 'Message "41" with delivery token 818720858 delivered' and 'Temp Sensors Publish Success!'. The middle window shows MQTT client logs with messages like '1658908551: New client connected from 127.0.0.1 as Send_End (p2, c1, k 00, u "amiah7325")'. The right window shows MQTT topic logs with messages like 'HereTopic name is: sensors/Audio308)Received operation 40,Wed Jul 27 00:55:51 2022'.

دیده می شود که پیام های مختلف با تایپیک های مختلف دریافت می شود و در بین آن ها صوت و تصویر نیز وجود دارد. حال به سراغ چک کردن پایگاه داده می رویم.



The image shows two terminal windows. The left window displays a MySQL query: 'mysql> select * from Facet;' and its results, which include columns 'id', 'Facet', and 'Date_Time'. The right window displays a MySQL query: 'mysql> select * from Audio;' and its results, which include columns 'id', 'AudioID', and 'Date_Time'.

همانطور که دیده می شود اطلاعات در هردو جدول پایگاه داده به درستی ذخیره شده اند. برای تشخیص چهره از عکس پرسنلی در مقابل دوربین استفاده کردم.

تا به اینجا دو بخش اول پروژه که جمع آوری و ذخیره در پایگاه داده و ارسال اطلاعات دما و لود و تصویر و صوت با استفاده از MQTT بود انجام شد. در بخش بعد به توضیح سرور می پردازیم.

بخش سوم: سرور با توانایی فرستادن درخواست http از نوع get

برای این بخش از کتابخانه boost/beast استفاده شده است و با استفاده از مثال موجود در سایت boost با عنوان web_server_fast این بخش انجام شده است. برای بخشی که باید تعداد n سطر آخر پایگاه داده را با توجه به نوع دستور نمایش دهد با استفاده از سرچ و جستجو زده شده است. ابتدا به تغییراتی که در مثال سایت boost ایجاد کرده ایم می پردازیم:

```
185 void finish_with_error(MYSQL *con){
186     fprintf(stderr, "%s\n", mysql_error(con));
187     mysql_close(con);
188     exit(1);
189 }
190
191 void process_request(http::request<request_body_t, http::basic_fields<alloc_t>> const& req)
192 {
193     switch (req.method())
194     {
195     case http::verb::get: {
196         std::map<std::string, std::string> fields;
197         std::string path;
198
199         std::string target(req.target());
200         std::stringstream targetstream(target);
201
202         if (target.find(":") != std::string::npos) {
203             std::string segment;
204             bool path_extracted = false;
205             int t;
206             while (std::getline(targetstream, segment, ':')) {
207                 if (path_extracted) {
208                     t = segment.find("=");
209                     if (t != std::string::npos) {
210                         fields[segment.substr(0, t)] = segment.substr(t + 1, segment.size());
211                     }
212                 } else {
213                     path = segment;
214                     path_extracted = true;
215                 }
216             }
217         } else {
218             path = target;
219         }
220     }
```

تابع اول finish_with_error برای کار با پایگاه داده قرار داده شده است. در بخش process_request تغییراتی که ایجاد کردیم ابتدا ورودی دستور get را به صورت string دریافت می کنیم اگر در داخل آن ‘:’ وجود داشت شروع به جدا سازی آن می کنیم. علت این بخش این است که می خواهیم برای دریافت n درخواستی که ارسال می شود به شکل /FaceDB:n=5 باشد اگر علامت ‘:’ تشخیص داده شد پس از ‘=’ را می یابیم و تعداد n را دریافت می کنیم در غیر این صورت که علامت ‘:’ وجود نداشته باشد کل ورودی به عنوان درخواست get در نظر گرفته می شود.

```
221     if (path == "/Pic") {
222
223         cv::VideoCapture camera(0);
224         cv::Mat frame;
225         camera >> frame;
226         cv::imwrite("image.png", frame);
227         send_file("/image.png");
228     }
```

در تصویر بالا بخش اول است که اگر درخواست Pic/ ارسال شود دوربین باز شده و یک عکس با نام image.png ذخیره می‌شود.

```

229     } else if (path == "/FaceDB") {
230         if (fields.count("n") > 0)
231             std::cout << fields["n"] << std::endl;
232         MYSQL *con = mysql_init(NULL);
233
234         if (mysql_real_connect(con, "localhost", "root", "aminh7325",
235 "ProjectDB", 0, NULL, 0) == NULL)
236         {
237             finish_with_error(con);
238         }
239         char query[300];
240         sprintf(query, "select * from FaceT order by id desc limit %d", stoi(fields["n"]));
241         if (mysql_query(con, query)) {
242             finish_with_error(con);
243         } else {
244
245             MYSQL_RES *result = mysql_store_result(con);
246
247             if (result == NULL)
248             {
249                 finish_with_error(con);
250             }
251
252             int num_fields = mysql_num_fields(result);
253
254             MYSQL_ROW row;
255
256             std::ofstream Data ("DataLog.txt");
257             Data << " id      Faces      Date & Time      \n";
258             while ((row = mysql_fetch_row(result)))
259             {
260                 Data << " ";
261                 int fieldlen[3] = {15, 15, 30};
262                 for(int i = 0; i < num_fields; i++)
263
264                     {
265                         char Mem[fieldlen[i]];
266                         int Mflag = 0;
267                         for(int m = 0; m < fieldlen[i]; m++){
268                             if((row[i])[m] != '\0') {
269                                 Mem[m] = (row[i])[m];
270                             }
271                             else {
272                                 Mflag = 1;
273                             }
274                             if(Mflag == 1) {
275                                 Mem[m] = ' ';
276                             }
277                         }
278                         printf("%s ", Mem ? Mem : "NULL");
279                         Data << Mem ? Mem : "NULL";
280                         Data << " ";
281                     }
282
283                     printf("\n");
284                     Data << "\n";
285                 }
286
287                 Data << "\n";
288                 mysql_free_result(result);
289                 Data.close();
290                 send_file("/DataLog.txt");
291             }

```

در دو تصویر بالا برای دریافت اطلاعات از بخش صوت آورده شده است که این بخش با استفاده از query دادن به پایگاه داده و ذخیره اطلاعات آن در یک فایل با عنوان Datalog.txt و ارسال آن به سرور مطالب مدنظر را نمایش می‌دهیم.

برای بخش صوت نیز به همین صورت است فقط اطلاعات آنجا از جدول AudioT که قبلا به آن اشاره شد خوانده می شود.

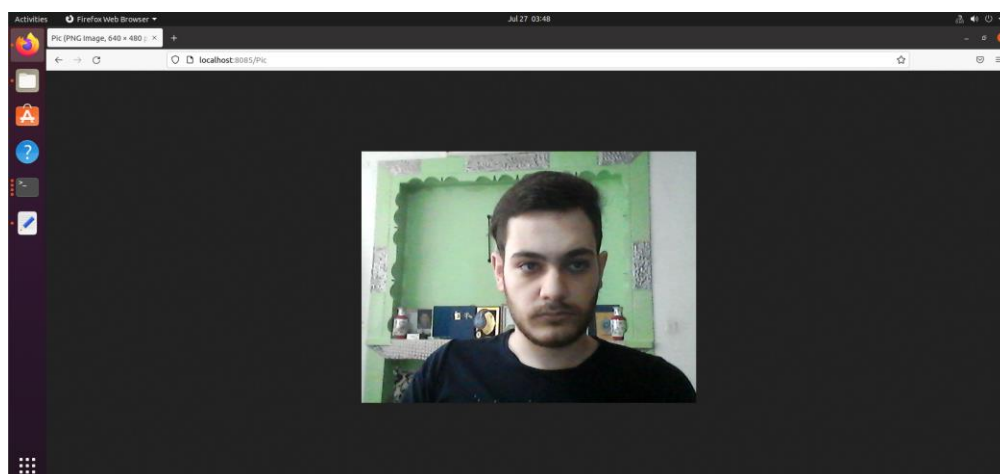
فایل ها و cmake این بخش نیز در پوشه ای با عنوان Server یافت می شود.

حال به سراغ تست و نتیجه گرفتن از این بخش می رویم.

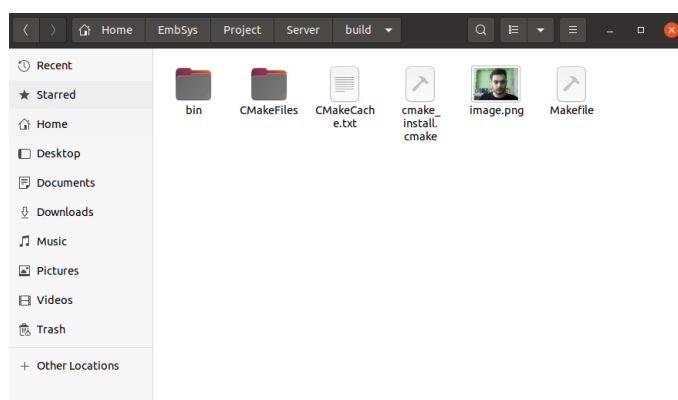
ابتدا با استفاده از دستور زیر سرور را بالا می آوریم:

```
aminh7325@ubuntu:~/EmbSys/Project/Server/build$ ./bin/Server 0.0.0.0 8085 . 100  
block
```

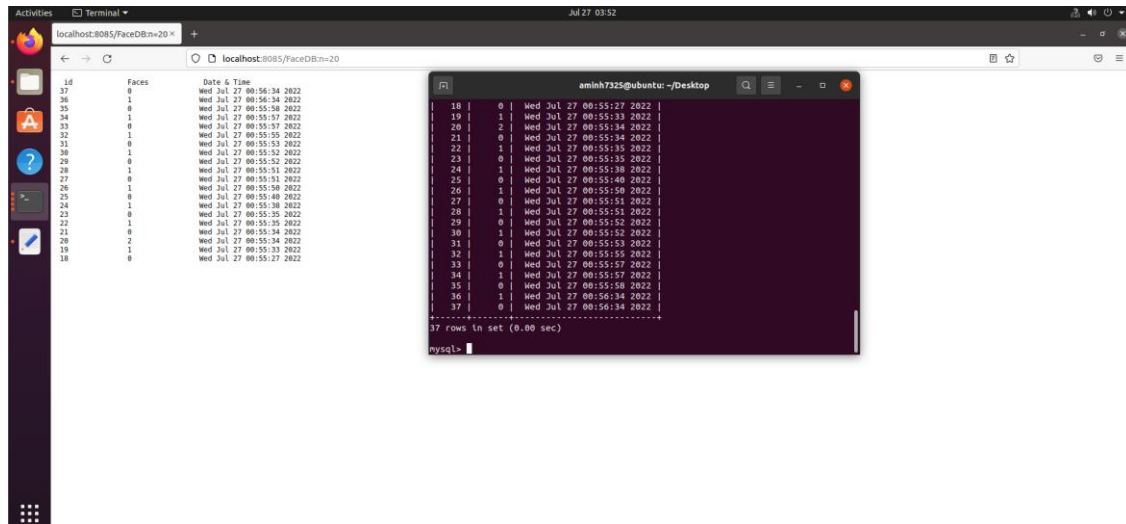
سپس از طریق مرورگر به آن درخواست می دهیم:



همانطور که دیده می شود با درخواست /Pic وبکم باز شده و عکس می گیرد و عکس در مسیر سرور ذخیره می شود و به ما نمایش داده می شود. فایل در مسیر مشخص ذخیره شده است.

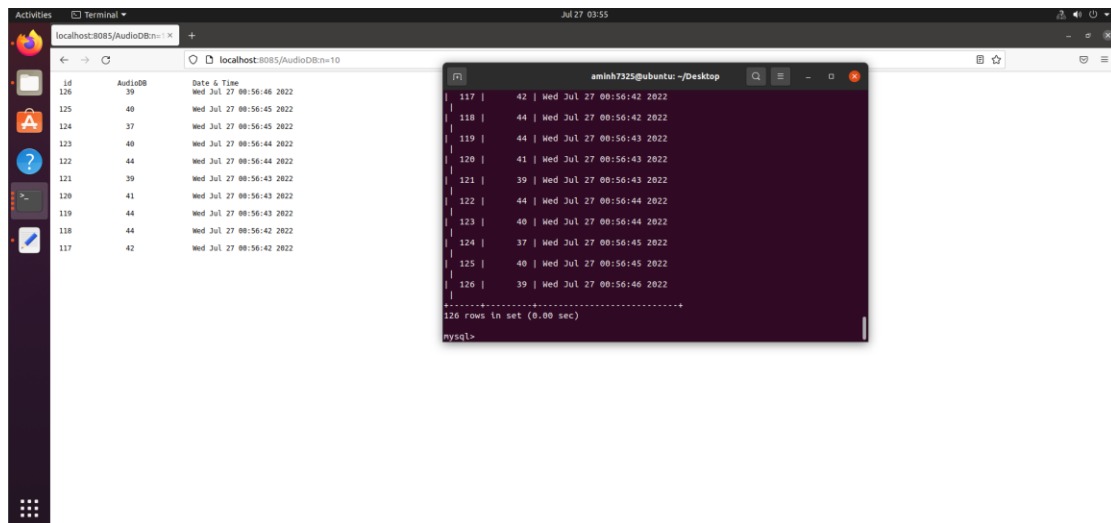


حال به سراغ دستور دادن برای دریافت اطلاعات چهره از پایگاه داده می‌کنیم:



همانطور که دیده می‌شود با دستور `/FaceDB:n=20` سطر آخر پایگاه داده برای ما به نمایش درآمده است که با چک کردن آن با توجه به پایگاه داده اصلی تفاوتی بین آن‌ها وجود ندارد.

برای صوت نیز داریم:



دیده می‌شود که برای این بخش نیز با دستور `/AudioDB:n=10` سطر آخر این جدول به درستی نمایش داده شده است.

در بخش آخر به بررسی سرویس‌ها می‌پردازیم.

بخش چهارم: سرویس ها

برای دو بخش اول که سابسکرایبر و پابلیشر هستند مشابه درس کدی مانند زیر برای آن ها زده شده است:

```
1 [Unit]
2 Description = Publisher
3 After=multi-user.target
4
5 [Service]
6 Type=simple
7 ExecStart=/home/aminh7325/EmbSys/Project/Pub/build/bin/Publisher
8 StandardOutput=journal+console
9 StandardError=journal+console
10 Restart=always
11 [Install]
12 WantedBy= multi-user.target
```

سپس با استفاده از دستورات زیر سرویس را فعال می کنیم:

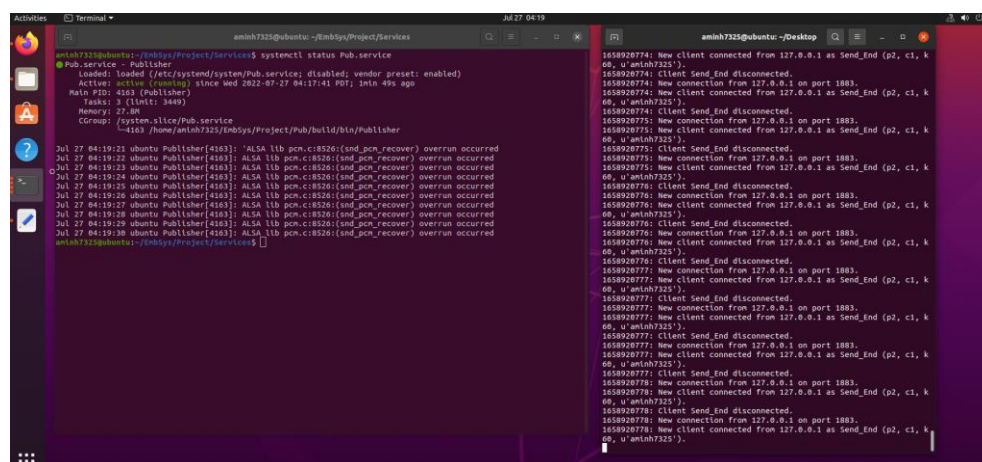
```
sudo cp Pub.service /etc/systemd/system
```

```
systemctl daemon-reload
```

```
systemctl start Pub
```

```
systemctl enable Pub
```

برای چک کردن اجرا شدن این کد کافی است به بروکر و status آن نگاه کنیم می بینیم فعال است ولی اگر به جورنال نگاه کنیم مشکلی که در این سرویس در جورنال نوشته شده است این می باشد که این برنامه نمی تواند نمایشگری پیدا کند زیرا به صورت سرویس در حال اجرا است و بعد از چندین بار ریستارت شدن، خود سیستم این برنامه را می بندد



The image shows two terminal windows. The left window, titled 'aminh7325@ubuntu: ~/EmbSys/Project/Services', displays the output of 'systemctl status Pub.service'. It shows the service is loaded, active (running), and has been running since Wed 2022-07-27 04:17:41 PDT. Below this, it lists the main PID as 4163 (Publisher) and shows a log of 'ALSA lib pcm.c:18526:(snd_pcm_recover) overrun occurred' repeated multiple times. The right window, titled 'aminh7325@ubuntu: ~', shows a log of network connections from 127.0.0.1 to port 1883, with messages like 'New client connected from 127.0.0.1 as Send_End (p2, c1, k 60, u'aminh7225')' and 'Client Send_End disconnected'.

برای رفع آن کافی است تمامی دستورات نمایشی را حذف کنیم. مشابه عملیات بالا برای بخش سابسکرایبر نیز انجام می‌شود.

برای بخش سرور ولی نیاز است که یک فایل دیگری نیز داشته باشیم زیرا برای اجرای آن مقادیری به عنوان ورودی باید بدهیم. سرویس این بخش به صورت زیر است:

```
1 [Unit]
2 Description = ServerMe
3 After=multi-user.target
4
5 [Service]
6 Type=simple
7 EnvironmentFile=/etc/ntpService
8 ExecStart=/home/aminh7325/EmbSys/Project/Server/build/bin/Server $ARG1 $ARG2 $ARG3 $ARG4 $ARG5
9 StandardOutput=journal+console
10 StandardError=journal+console
11 Restart=always
12 [Install]
13 WantedBy= multi-user.target
14
```

و فایل ntpService هم به صورت زیر است:

```
1 ARG1 = 0.0.0.0
2 ARG2 = 8085
3 ARG3 = .
4 ARG4 = 100
5 ARG5 = block
6
```

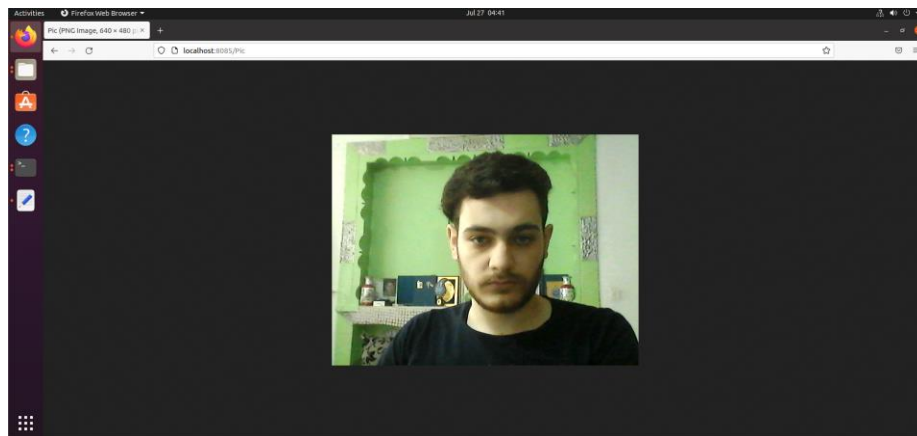
پس از انتقال سرویس به محل مدنظر و اجرای آن مشابه عکس زیر دیده می‌شود که سرویس به درستی در حال کار کردن است.

```
aminh7325@ubuntu:~/EmbSys/Project/Services$ sudo cp ServerMe.service /etc/systemd/system
aminh7325@ubuntu:~/EmbSys/Project/Services$ sudo cp ntpService /etc
aminh7325@ubuntu:~/EmbSys/Project/Services$ systemctl daemon-reload
aminh7325@ubuntu:~/EmbSys/Project/Services$ systemctl start ServerMe
aminh7325@ubuntu:~/EmbSys/Project/Services$ systemctl status ServerMe.service
● ServerMe.service - ServerMe
   Loaded: loaded (/etc/systemd/system/ServerMe.service; disabled; vendor preset: enabled)
   Active: active (running) since Wed 2022-07-27 04:40:51 PDT; 6s ago
     Main PID: 6115 (Server)
       Tasks: 1 (limit: 3449)
        Memory: 18.3M
         CGroup: /system.slice/ServerMe.service
                └─6115 /home/aminh7325/EmbSys/Project/Server/build/bin/Server 0.0.0.0 8085 . 100 block

Jul 27 04:40:51 ubuntu systemd[1]: Started ServerMe.
```

پس از آن مجدد از طریق مرورگر درخواست به سرور می‌فرستیم و مشابه تصویر بعدی نتیجه را دریافت می‌کنیم و سرویس دائماً در حال اجرا است.

با توجه به زمان تصویر می‌توان از صحت این بخش اطمینان کرد.



فایل‌های این بخش نیز در پوشه ای با عنوان **Services** موجود است.

یک بخش دیگر در انتها وجود دارد که برای سؤالاتی است که در امتحان پرسیده شده بود که به پاسخ دادن به آن‌ها می‌پردازیم.

بخش پنجم: پاسخ به سوال امتحان پایانترم

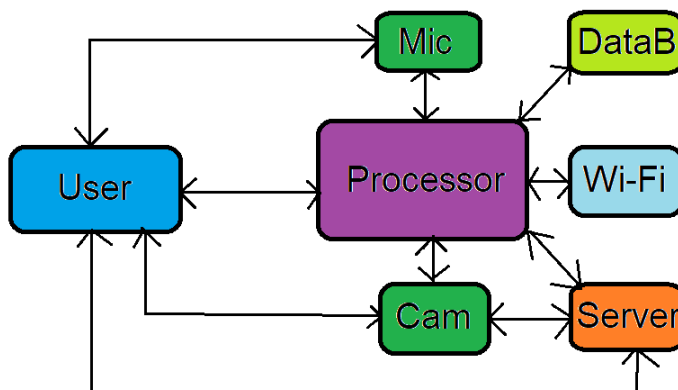
سوال اول: اجزاء سیستم نهفته پروژه درس و کارکردهای نهایی آن و **Library** ها و ابزارهایی که به کار گرفته اید را به اختصار شرح دهید.

جواب: برای این پروژه ما نیاز به یک دوربین با توانایی ثبت تصاویر رنگی، یک عدد میکروفون برای دریافت صوت، یک ماژول برای اتصال به اینترنت و یک پردازنده که می‌تواند لپتاپ یا هر چیز دیگری باشد.

کتابخانه‌های مورد استفاده: اولین کتابخانه **OpenCV** است که برای پردازش و تشخیص تصاویر از آن استفاده شده است.

برای پردازش صوت از کتابخانه **Alsa** استفاده شده است و برای بخش سرور نیز از **Boost/beast** و فایل‌های موجود آن استفاده کرده‌ایم. همچنین برای پایگاه داده **MySQL** و برای فرستادن پیام از **MQTT** و **Paho** برای ساختن بروکر و بخش پابلیشر و سابسکرایبر استفاده کردیم. از یک سری کتابخانه از جمله **math.h** و **string.h** نیز در بخش‌های مختلف از آن استفاده کردیم. روند کلی و این ابزارها به طور کامل در بخش‌های قبل توضیح داده شده است.

سوال دوم: با رسم یک دیاگرام بلوکی ارتباط بین این اجزاء را روشن کنید.



سوال سوم: نحوه اجرای برنامه اصلی را بصورت یک یا چند سرویس بیان نموده و فایل‌های service مناسب را نوشته و تشریح نمایید.

جواب: از ۳ سرویس استفاده شده است و در بخش چهارم به طور کامل توضیح داده شده است. فایل‌های مربوط به آن را نیز در پوشه Services می‌توانید بیابید.

سوال چهارم: نحوه پیاده سازی رابط وب (web interface) سیستم، فایل‌ها و محل قرار گرفتن آنها را روشن نمایید.

جواب: برای این بخش از کتابخانه Boost استفاده می‌کنیم به طور کلی برای این سرور سه درخواست داریم. درخواست اول که یک تصویر از وبکم را بگیرد و در فایل ذخیره کند و آن را به عنوان خروجی به سرور بدهد. دو درخواست دیگر دریافت اطلاعات از پایگاه داده است که برای آن ابتدا با استفاده از MySQL اطلاعات را به صورت یک فایل نوشتاری (text) در می‌آوریم و این فایل را برای سرور می‌فرستیم. این فایل‌ها در مسیری که فایل سرور وجود دارد وجود می‌آیند.

سوال پنجم: نحوه تعامل بخش (ج) و (د) را شرح دهید.

جواب: سرویس‌ها دائم در حال اجرا هستند و باعث می‌شوند فایل‌هایی که برای سرور قرار است فرستاده شود دائما در حال تغییر و آپدیت شدن باشد. به همین دلیل هربار درخواستی که برای سرور فرستاده شود جدیدترین اطلاعات برای آن فرستاده خواهد شد.

سوال ششم: همچنین روند انتخاب اجزاء و پارامترهای سیستم را بر اساس MRD ی که تهیه کرده اید بیان نمایید.

با توجه به کاربرد می تواند انتخاب ما تحت تغییراتی قرار بگیرد به عنوان مثال برای تشخیص چهره در یک فضای عمومی برای شناسایی مجرم نیاز به کیفیت دوربین بالا با قابلیت تصویر برداری با فریم ریت بالا است. و یا برای تشخیص سر و صدا در یک انباری برای فهمیدن حضور شخصی در آن نیاز است که به خوبی رفع نویز را انجام بدهد و از میکروفون با کیفیتی استفاده بکنیم. در کاربردهای کم ریسک تر می توان به تبع نیاز از خوب بودن این قطعات کاست. انتخاب پردازنده نیز بسیار مهم است و باید با توجه به کیفیت تصاویر و اینکه تا چه حدی پردازش ها سنگین است استفاده کنیم. نباید پردازنده بسیار قوی انتخاب شود که از درصد خوبی از قابلیت های آن استفاده نشود و نه اینکه انقدر از آن استفاده سنگین کنیم که درموارد کریتیکال سیستم داون شود.