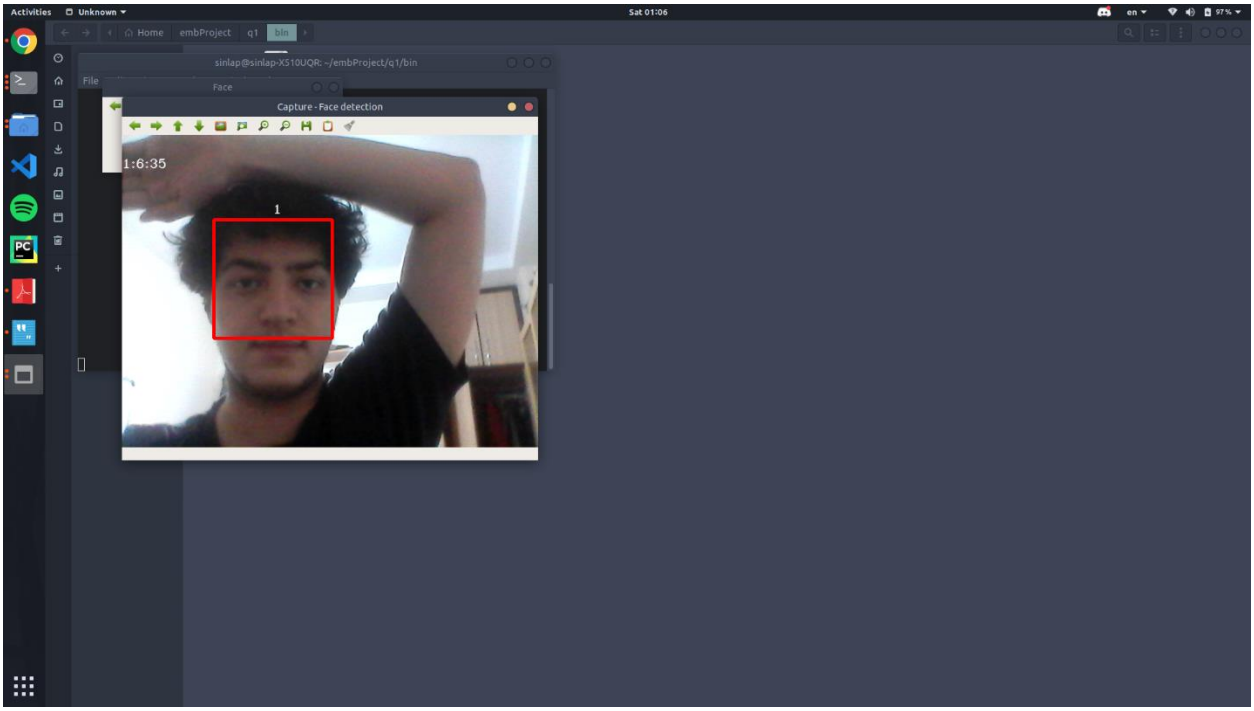
سینا کریمی97105509

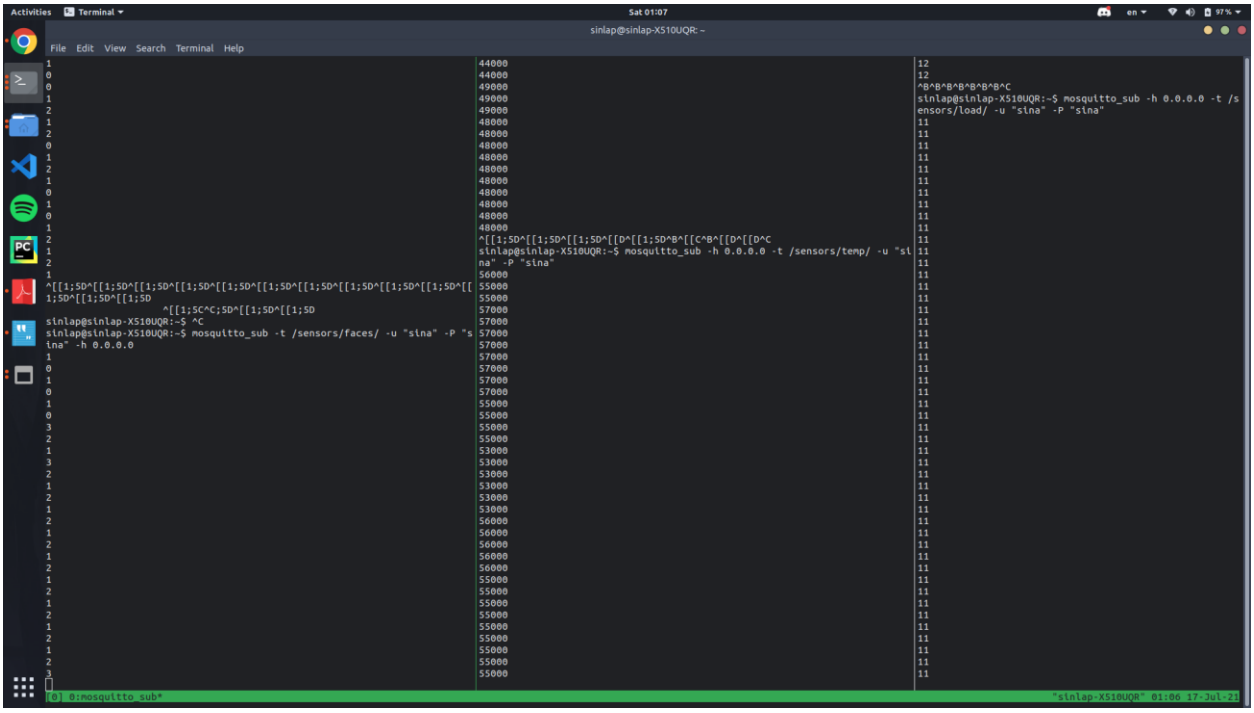
توضیحات سوال اول:

در این سوال از کتابخانه های opencv و pahomqtt.cpp استفاده شده است. ابتدا نتایج را میبینیم:

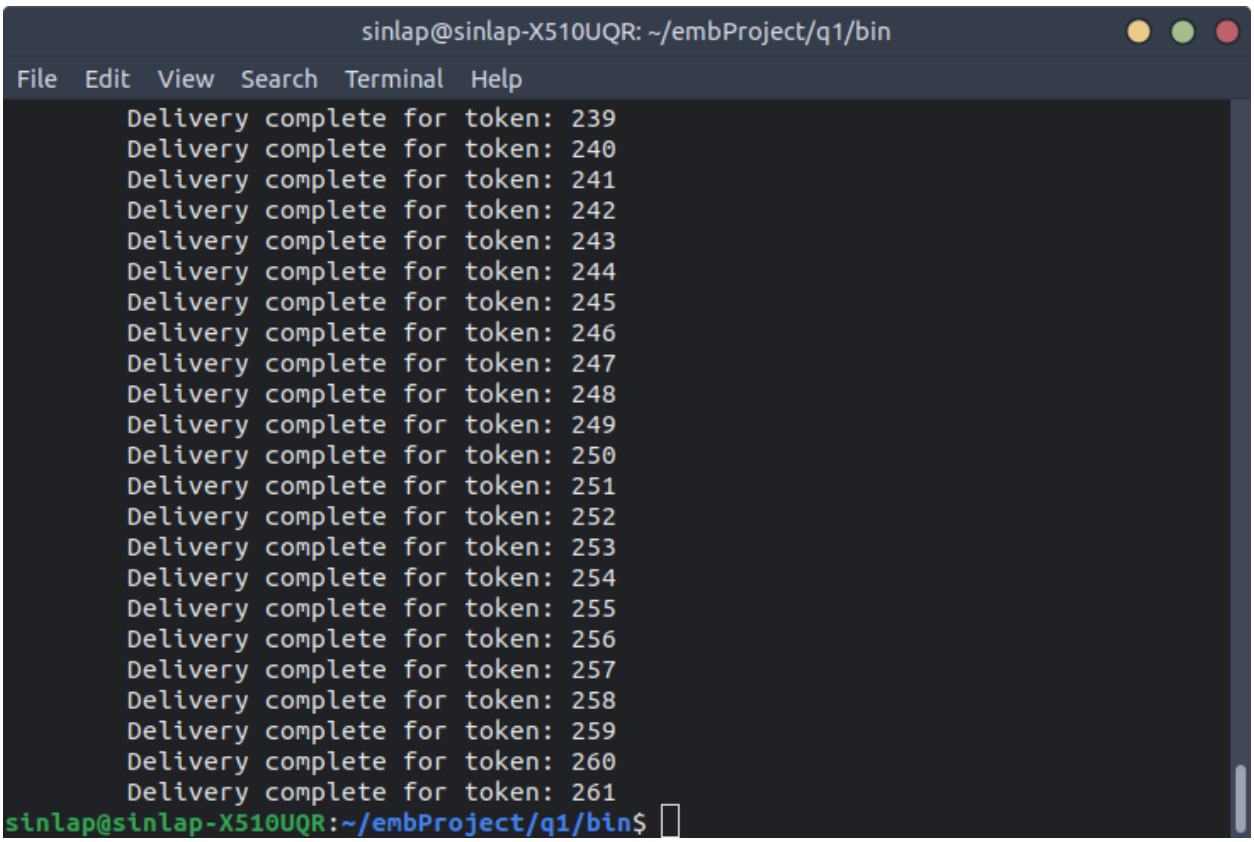
زمانی که برنامه شروع به کار میکند، دور صورت یک مربع قرمز کشیده میشود و بالای مربع شماره صورت تشخیص داده شده نوشته میشود و در بالا سمت چپ نیز زمان سیستم نوشته میشود:



و در هر مرحله که تعداد صورت ها در تصاویر تغییر پیدا کند، لود سی پی یو به صورت درصد و دمای آن به صورت میلی سانتیگراد و تعداد صورت ها به صورت امن فرستاده میشود:



و در عکس بعد نیز مقدار توکن های خود برنامه زمانی که پیام را میفرستد میبینیم:



توضیحات مربوط به کد:

برای تشخیص چهره از دیتابیس Haarcascade استفاده شده است. در آخر برنامه تابعی به نام detectAndDisplay وجود دارد که تصویر را میگیرد و در آن صورت و بقیه اطلاعات را مشخص میکند و نمایش میدهد و به عنوان خروجی تعداد صورت های تشخیص داده شده را میدهد.

3 کلاس اول برنامه مربوط به فرستادن پیام با استفاده از pahomqtt.cpp میباشد که از کد فرستنده آن برداشته شده است. در قسمت تنظیمات ارسال همانطور که میبینید با یوزرنیم و پسورد فرستاده میشود که اتصال امن برقرار شود.

```
auto connOpts = mqtt::connect_options_builder()
    .clean_session()
    .will(mqtt::message(TOPIC, LWT_PAYLOAD, QOS))
    .user_name("sina")
    .password("sina")
    .finalize();
cout<<"sending message"<<endl;
```

در این قسمت عکس از وبکم گرفته میشود و اگر وبکم شناخته نشود ارور گرفته میشود.

```
cv::VideoCapture camera(0);
if (!camera.isOpened()) {
    std::cerr << "ERROR: Could not open camera" << std::endl;
    return 1;
}
```

در این قسمت دیتابیس موردنظر لود میشود و اگر شناخته نشود ارور داده میشود:

```
if( !face_cascade.load( "/home/sinlap/opencv/opencv/data/haarcascades/haarcascade_frontalcatface.xml" ) )
{
    cout << "--(!)Error loading face cascade\n";
    return -1;
};
```

در قسمت بعد آن یک پنجره برای نمایش تصویر درست میشود و بعد از آن متغیر های مربوط به فرستادن اطلاعات از طریق mqtt و تاپیک های آنها مشخص میشوند:

```
namedWindow("Face");
mqtt::message_ptr pubmsg = mqtt::make_message("/sensors/faces/", to_string(faceNumber) );
        pubmsg->set_qos(QOS);
        client.publish(pubmsg)->wait_for(TIMEOUT);
mqtt::message_ptr pubcpuLoad = mqtt::make_message("/sensors/load/", to_string(load) );
        pubcpuLoad->set_qos(QOS);
        client.publish(pubcpuLoad)->wait_for(TIMEOUT);
mqtt::message_ptr pubcputemp = mqtt::make_message("/sensors/temp/", to_string(tempCPU) );
        pubcputemp->set_qos(QOS);
        client.publish(pubcputemp)->wait_for(TIMEOUT);
```

بعد از آن برنامه وارد یک لوپ میشود و هر بار که صورت ها در تصویر شناخته میشود، مقدار آن با مقدار قبلیش مقایسه میشود و اگر متفاوت بودند مقادیر لود و دمای سی پی یو محاسبه میشود و فرستاده میشود:

```
Mat frame;
camera >> frame;
temp = detectAndDisplay( frame );
if (temp != faceNumber){
    faceNumber = temp;


    fp = fopen("/proc/stat","r");
    // CPU Word
    fscanf(fp, "%s ", buff);
    // First Number
    fscanf(fp, "%s ", buff);
    sscanf(buff, "%d", &cpu1);
    //Second Number
    fscanf(fp, "%s ", buff);
    sscanf(buff, "%d", &cpu2);
    //Third Number
    fscanf(fp, "%s ", buff);
    sscanf(buff, "%d", &cpu3);
    // Idle time
```

که مقدار لود cpu از فایل /proc/stat قابل محاسبه است و مقدار دما نیز از فایل /sys/class/thermal/thermal_zone0/temp به صورت میلی سانتیگراد قابل خواندن میباشد. بعد از محاسبه این 2 مقدار به همراه تعداد صورت ها از طریق mqtt فرستاده میشود. برنامه در صورتی که کلیدی را فشار دهید خارج میشود.

برای فایل CMakeLists.txt نیز مکان مربوط به کتابخانه OpenCV به صورت دستی آدرس داده شده است.