#define TEMAPROJEKAT "ugradbeni/projekat/youtubesub"

#include "mbed.h"

#define MQTTCLIENT\_QOS2 0

#include "easy-connect.h"

#include "MQTTNetwork.h"

#include "MQTTmbed.h"

#include "MQTTClient.h"

#include <string.h>

#include <vector>

#include <string>

#include <algorithm>

#include "stm32f413h\_discovery\_ts.h"

#include "stm32f413h\_discovery\_lcd.h"

TS\_StateTypeDef TS\_State = { 0 };

DigitalIn button1(p5);

DigitalIn button2(p6);

int channelCount = 0;

int currentChannel = 0;

int usernameLength = 0;

char \*channelData;

std::vector<std::string> channelsNames;

std::vector<std::string> channelsSubscribers;

std::vector<std::string> channelsViews;

std::vector<std::string> channelsVideos;

std::vector<std::string> channelsCountries;

void getDefaultValues(){

BSP\_LCD\_Clear(LCD\_COLOR\_WHITE);

BSP\_LCD\_SetTextColor(LCD\_COLOR\_RED);

BSP\_LCD\_FillRect(0, 0, BSP\_LCD\_GetXSize(), 50);

BSP\_LCD\_SetTextColor(LCD\_COLOR\_BLACK);

BSP\_LCD\_SetBackColor(LCD\_COLOR\_RED);

}

void drawChannelNameBanner(){

int n = 0;

if(currentChannel > 0)

n = channelsNames[currentChannel - 1].length();

else

n = 0;

char pom[n];

if(n > 0)

strcpy(pom, channelsNames[currentChannel - 1].c\_str());

else

strcpy(pom, " ");

BSP\_LCD\_FillRect(0, 50, BSP\_LCD\_GetXSize(), 40);

BSP\_LCD\_SetTextColor(LCD\_COLOR\_WHITE);

BSP\_LCD\_SetBackColor(LCD\_COLOR\_BLACK);

BSP\_LCD\_SetFont(&Font12);

BSP\_LCD\_DisplayStringAt(0, 65, (uint8\_t \*)"Channel: ", LEFT\_MODE);

BSP\_LCD\_SetFont(&Font16);

BSP\_LCD\_DisplayStringAt(0, 63, (uint8\_t \*)pom, RIGHT\_MODE);

}

void drawYouTubeTriangleLogo(){

Point points[3];

points[0].X = 30; points[0].Y = 15;

points[1].X = 50; points[1].Y = 25;

points[2].X = 30; points[2].Y = 35;

BSP\_LCD\_SetTextColor(LCD\_COLOR\_WHITE);

BSP\_LCD\_FillPolygon(points, 3);

}

void drawAppTitleBanner(){

getDefaultValues();

drawYouTubeTriangleLogo();

BSP\_LCD\_SetTextColor(LCD\_COLOR\_BLACK);

BSP\_LCD\_SetFont(&Font16);

BSP\_LCD\_DisplayStringAt(0, 15, (uint8\_t \*)"YouTube", RIGHT\_MODE);

BSP\_LCD\_SetFont(&Font12);

BSP\_LCD\_DisplayStringAt(0, 30, (uint8\_t \*)"Subscriber Counter", RIGHT\_MODE);

BSP\_LCD\_SetTextColor(LCD\_COLOR\_BLACK);

}

void drawInfoFrame(){

BSP\_LCD\_SetTextColor(LCD\_COLOR\_RED);

BSP\_LCD\_DrawRect(0, 90, BSP\_LCD\_GetXSize()-1, 109);

}

void setChannelInfo(){

drawInfoFrame();

BSP\_LCD\_SetBackColor(LCD\_COLOR\_WHITE);

BSP\_LCD\_SetTextColor(LCD\_COLOR\_BLACK);

BSP\_LCD\_SetFont(&Font16);

char channelNumbers[20];

if(channelCount != 0)

sprintf(channelNumbers, "Channel number: %d", currentChannel);

else

sprintf(channelNumbers, "Channel number: ");

char subscribers[50];

if(channelsSubscribers.size() > 0 && channelsSubscribers[0].length() != 0)

sprintf(subscribers, "Subscribers: %s", channelsSubscribers[currentChannel - 1].c\_str());

else

sprintf(subscribers, "Subscribers: ");

char views[50];

if(channelsViews.size() > 0 && channelsViews[0].length() != 0)

sprintf(views, "Views: %s", channelsViews[currentChannel - 1].c\_str());

else

sprintf(views, "Views: ");

char uploads[20];

if(channelsVideos.size() > 0 && channelsVideos[0].length() != 0)

sprintf(uploads, "Uploads: %s", channelsVideos[currentChannel - 1].c\_str());

else

sprintf(uploads, "Uploads: ");

char country[20];

if(channelsCountries.size() > 0 && channelsCountries[0].length() != 0)

sprintf(country, "Country: %s", channelsCountries[currentChannel - 1].c\_str());

else

sprintf(country, "Country: ");

BSP\_LCD\_DisplayStringAt(0, 95, (uint8\_t \*)channelNumbers, LEFT\_MODE);

BSP\_LCD\_DisplayStringAt(0, 115, (uint8\_t \*)subscribers, LEFT\_MODE);

BSP\_LCD\_DisplayStringAt(0, 135, (uint8\_t \*)views, LEFT\_MODE);

BSP\_LCD\_DisplayStringAt(0, 155, (uint8\_t \*)uploads, LEFT\_MODE);

BSP\_LCD\_DisplayStringAt(0, 175, (uint8\_t \*)country, LEFT\_MODE);

}

void drawInstructionBanner(){

BSP\_LCD\_SetTextColor(LCD\_COLOR\_BLACK);

BSP\_LCD\_FillRect(0, BSP\_LCD\_GetYSize()-40, BSP\_LCD\_GetXSize(), 40);

BSP\_LCD\_SetTextColor(LCD\_COLOR\_GREEN);

BSP\_LCD\_SetBackColor(LCD\_COLOR\_BLACK);

BSP\_LCD\_SetFont(&Font12);

BSP\_LCD\_DisplayStringAt(0, 205, (uint8\_t \*)"Button 1: Next Channel", LEFT\_MODE);

BSP\_LCD\_DisplayStringAt(0, 220, (uint8\_t \*)"Button 2: Previous Channel", LEFT\_MODE);

}

void setLCD(){

BSP\_LCD\_Init();

drawAppTitleBanner();

drawChannelNameBanner();

setChannelInfo();

drawInstructionBanner();

}

void getChannelUploadsAndPutIntoArray(char \*channelInfo) {

std::string delimiter = ";";

size\_t pos = 0;

std::string channelInfoString = channelInfo;

std::string info;

pos = channelInfoString.find(delimiter);

info = channelInfoString.substr(0, pos);

channelsVideos.push\_back(info);

channelInfoString.erase(0, pos + delimiter.length());

}

void getChannelSubscribersAndPutIntoArray(char \*channelInfo) {

std::string delimiter = ";";

size\_t pos = 0;

std::string channelInfoString = channelInfo;

std::string info;

pos = channelInfoString.find(delimiter);

info = channelInfoString.substr(0, pos);

channelsSubscribers.push\_back(info);

channelInfoString.erase(0, pos + delimiter.length());

char \*channelData;

strcpy(channelData, channelInfoString.c\_str());

getChannelUploadsAndPutIntoArray(channelData);

}

void getChannelViewsAndPutIntoArray(char \*channelInfo) {

std::string delimiter = ";";

size\_t pos = 0;

std::string channelInfoString = channelInfo;

std::string info;

pos = channelInfoString.find(delimiter);

info = channelInfoString.substr(0, pos);

channelsViews.push\_back(info);

channelInfoString.erase(0, pos + delimiter.length());

char \*channelData;

strcpy(channelData, channelInfoString.c\_str());

getChannelSubscribersAndPutIntoArray(channelData);

}

void getChannelCountriesAndPutIntoArray(char \*channelInfo) {

std::string delimiter = ";";

size\_t pos = 0;

std::string channelInfoString = channelInfo;

std::string info;

pos = channelInfoString.find(delimiter);

info = channelInfoString.substr(0, pos);

channelsCountries.push\_back(info);

channelInfoString.erase(0, pos + delimiter.length());

char \*channelData;

strcpy(channelData, channelInfoString.c\_str());

printf("%s", channelData);

getChannelViewsAndPutIntoArray(channelData);

}

bool getChannelNameAndPutIntoArray(char \*channelInfo) {

std::string delimiter = ";";

size\_t pos = 0;

std::string channelInfoString = channelInfo;

std::string info;

pos = channelInfoString.find(delimiter);

info = channelInfoString.substr(0, pos);

if(std::find(channelsNames.begin(), channelsNames.end(), info) == channelsNames.end()){

channelsNames.push\_back(info);

channelInfoString.erase(0, pos + delimiter.length());

char \*channelData;

strcpy(channelData, channelInfoString.c\_str());

printf("%s", channelData);

getChannelCountriesAndPutIntoArray(channelData);

return true;

}

return false;

}

void removeChannel(int channelNumber) {

channelCount--;

if(currentChannel >= channelNumber && (currentChannel > 1 || (currentChannel == 1 && channelsNames.size() == 1)))

currentChannel--;

channelsNames.erase(channelsNames.begin() + channelNumber - 1);

channelsSubscribers.erase(channelsSubscribers.begin() + channelNumber - 1);

channelsVideos.erase(channelsVideos.begin() + channelNumber - 1);

channelsViews.erase(channelsViews.begin() + channelNumber - 1);

channelsCountries.erase(channelsCountries.begin() + channelNumber - 1);

setLCD();

}

void messageArrived(MQTT::MessageData& md)

{

MQTT::Message &message = md.message;

channelData = (char\*)message.payload;

if(!(channelData[0] >= '0' && channelData[0] <= '9')){

if(getChannelNameAndPutIntoArray(channelData)){

channelCount++;

if(currentChannel == 0) currentChannel++;

if(channelCount > 1)

currentChannel++;

setLCD();

}

}

else if(channelsNames.size() > 0 && channelData[0] >= '0' && channelData[0] <= '9'){

std::string deleted = channelData;

int deletedIndex = stoi(deleted);

removeChannel(deletedIndex);

}

}

int main() {

BSP\_LCD\_Init();

setLCD();

printf("Ugradbeni sistemi\r\n");

printf("Test projekta\r\n\r\n");

NetworkInterface \*network;

network = NetworkInterface::get\_default\_instance();

if (!network) {

return -1;

}

MQTTNetwork mqttNetwork(network);

MQTT::Client<MQTTNetwork, Countdown> client(mqttNetwork);

const char\* hostname = "broker.hivemq.com";

int port = 1883;

printf("Connecting to %s:%d\r\n", hostname, port);

int rc = mqttNetwork.connect(hostname, port);

if (rc != 0)

printf("rc from TCP connect is %d\r\n", rc);

MQTTPacket\_connectData data = MQTTPacket\_connectData\_initializer;

data.MQTTVersion = 3;

data.clientID.cstring = "ugradbeni";

data.username.cstring = "";

data.password.cstring = "";

if ((rc = client.connect(data)) != 0)

printf("rc from MQTT connect is %d\r\n", rc);

MQTT::Message message;

char buf[100];

while (1) {

BSP\_TS\_GetState(&TS\_State);

rc = client.subscribe(TEMAPROJEKAT, MQTT::QOS0, messageArrived);

if(button2){

if(currentChannel < channelsNames.size())

currentChannel++;

setLCD();

}

if(button1){

if(currentChannel > 1)

currentChannel--;

setLCD();

}

//printf("%d - %d\n", channelCount, currentChannel);

wait\_ms(10);

}

}