ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ, ФИЛИАЛ ПЛОВДИВ
ФАКУЛТЕТ ПО ЕЛЕКТРОНИКА И АВТОМАТИКА

Курсова задача

по

Разпределени вградени системи

на

Валдис Златанов, Михаела Малинова, Иван Янчев

Тема: ″Контролер за управление и извличане на информация от USB камера″

Специалност: КСТ

Образователно-квалификационна степен: Магистър

СЪДЪРЖАНИЕ

УВОД

Целта на текущата курсовата задача е да се изготви разпределена вградена система за управление и извличане на информация, както от USB камера, така и от GPS модул. Контролерът може да управлява настройките на камерата и режимите ѝ на работа, като под режими на работа се има в предвид включване/изключване на камерата, както и дали да е в режим запис на видео или заснемане на снимка (screenshot) от видеото, като снимката се прави от страна на сървъра. Освен това, има допълнителна функционалност за запис на видеото, което се предава към сървъра. Сървърът от своя страна също може да записва видеото, което получава. Записът се осъществява чрез натискане на бутон Record Video, което се запазва в папката, от която е стартирано приложението в avi формат.

Поточното видео се предават чрез Интернет (TCP/IP) към сървър, заедно с координатите, определени и извлечени от GPS модулът от страна на клиента(вградената система) към сървъра(персонален компютър), където също се изобразяват чрез подходящ потребителски интерфейс.

За изпълнението на курсовата задача са използвани Raspberry Pi 4 платформа, периферни устройства като уеб камера и GPS приемник, както и подходяща за целта среда за разработка – Qt Creator. Всеки един от тези компоненти и използвани средства са описани в *Глава 2*.

ГЛАВА 1

1.1 Идея и решение

Идеята на поставената задача е да бъде изпълнена чрез комуникационна връзка между сървър и клиент. Сървърът е персонален компютър, чиято основна функция е да предостави интерфейс между потребителя и предаваните данни от вграденото устройство, чрез който да изобрази получените данни от клиента. Използването на компютър с общо предназначение е често срещано при работа с вградени устройства. Често информацията от тях се предава чрез web страници, които могат да бъдат визуализирани чрез интернет търсачка. Тази функционалност не е необходима, поради предоставените от средата за разработка**(QT Creator)** интуитивни инструменти за разработване на графичен потребителски интерфейс, спестяващи много програмно време.

Недостатък на споменатия подход е, че данните ще бъдат достъпни само до оператора/ите на клиентската машината, която е една на брой. За целите на поставената задача споменатият недостатък е допустим. Подробно обяснение на интерфейсът е поместено в *Глава 3*.

Клиентът от друга страна е вградена система, което по терминология означава, че изпълнява задача, или набор от задачи с точно определена цел, за разлика от компютрите с общо предназначение, които имат много по-широк спектър от функционалности.

Неговата задачата ще бъде да приема и обработва информацията получена от външни сензори (обикновена USB камера и обикновено GPS устройство, засичащо географски координати – географска ширина и географска дължина, текущо време) и да я препраща към сървъра.

1.2. Функционалност

Клиентът (вградената система Raspberry Pi 4) получава данни от сензорите (USB камера и GPS устройство) в реално време чрез хардуерна връзка (USB кабел) и изпраща данните под формата на Интернет пакети към клиента (персонален компютър). Целта на транспортирането на данните през Интернет мрежа е да се разшири пространственият диапазон, в който ще може да се осъществи надеждна и сигурна връзка.

При използването на други методи за комуникация (кабелна връзка, Bluetooth, Zigbee и т.н.) съществува пространствено ограничение между две устройства, което не трябва да надвишава стойности от порядъка на десетки метри. Чрез комуникация през Интернет максималното

разстояние между две устройства не е от съществено значение и влияние, стига и двете устройства да имат достъп до съответната мрежа.

За интегритет на данните, надеждност на съобщенията и контрол на трансмисията се грижи TCP/IP протоколът.

Нужно е да бъде отчетен и проблемът със сигурността. Конфиденциалността, както на информацията, така и на работата на цялата система, са от ключово значение. За решение на този проблем данните биват компресирани при предаване чрез кодиращ алгоритъм *tobase64*. Освен ако атакуващата страна не познава методът за декодиране и какво съдържат пакетите, за нея те ще бъдат безполезни. Други мерки за сигурност не са взети. Системата е уязвима от атаки, свързани с промяна на съдържанието на пакети, неоторизиран достъп и др.

Клиентът(вградената система) изпраща команди към сървъра, с които се установява TCP/IP връзката със сървъра, след като той от своя страна стартира сървъра, към който могат да се правят кънекции, стартира се предаването на видео поток, стартира се предаването на GPS координати, подава се сигнал за записване на видео потока.

За по-добра надеждност, функционалност и контрол GPS-ът и камерата са разделени на отделни сървъри, като слушат и пишат на различни един от друг портове.

ГЛАВА 2

2.1. Използвани платформи и компоненти

Използваната платформа за разработка е Raspberry Pi 4. Raspberry Pi 4 има подобен дизайн и размери като своите предшественици, но това всъщност е изцяло нова платформа, захранвана от нов процесор, Broadcom BCM2711B0. Raspberry Pi всъщност представлява платка от серия малки single-board компютри (SBCs), която поради ниската си цена, отворен дизайн и приемането на HDMI и USB устройства се използва широко в много области. Хардуерът на Raspberry Pi се е развил чрез няколко версии, които се отличават с вариации във вида на централния процесор, обема на паметта, поддръжката на мрежа и поддръжката на периферни устройства.

На споменатия вече Raspberry Pi 4 е инсталирана операционна система Raspbian. Един от най-бързите и лесни начини за инсталиране на операционната система е чрез Raspberry Pi Imager от microSD карта. Като алтернатива може да се изтегли и инсталира и ръчно. Raspbian OS е безплатна операционна система, базирана на Debian и оптимизирана за хардуера на Raspberry Pi. Raspberry Pi OS е силно оптимизирана за линията Raspberry Pi от компактни едноплаткови компютри с ARM процесори. Работи на всеки Raspberry Pi с изключение на микроконтролера Pico.

Използваните периферни устройства са съответно TRACER HD WEB008 уеб камера и Mulberry USB G-Mouse GPS приемник.

Основните характеристики на избраната уеб камера са:

- Изображение с HD качество;
- 100° ъгъл на гледане на обектива;
- Автоматична корекция на осветлението;
- Вграден микрофон;
- Ръчно регулиране на фокуса.

GPS модулът от своя страна е с изключително ниска мощност и висока точност USB устройство.

Основните характеристики на USB GPS устройството са:

- Икономично и наистина бързо сателитно прехващане;
- 56 канала;
- Чувствителност: -165 dBm;
- Консумирана мощност: Макс. 25mA при 3.3V;
- Интерфейс: USB;
- GPS протокол: Stardard NMEA 0183.

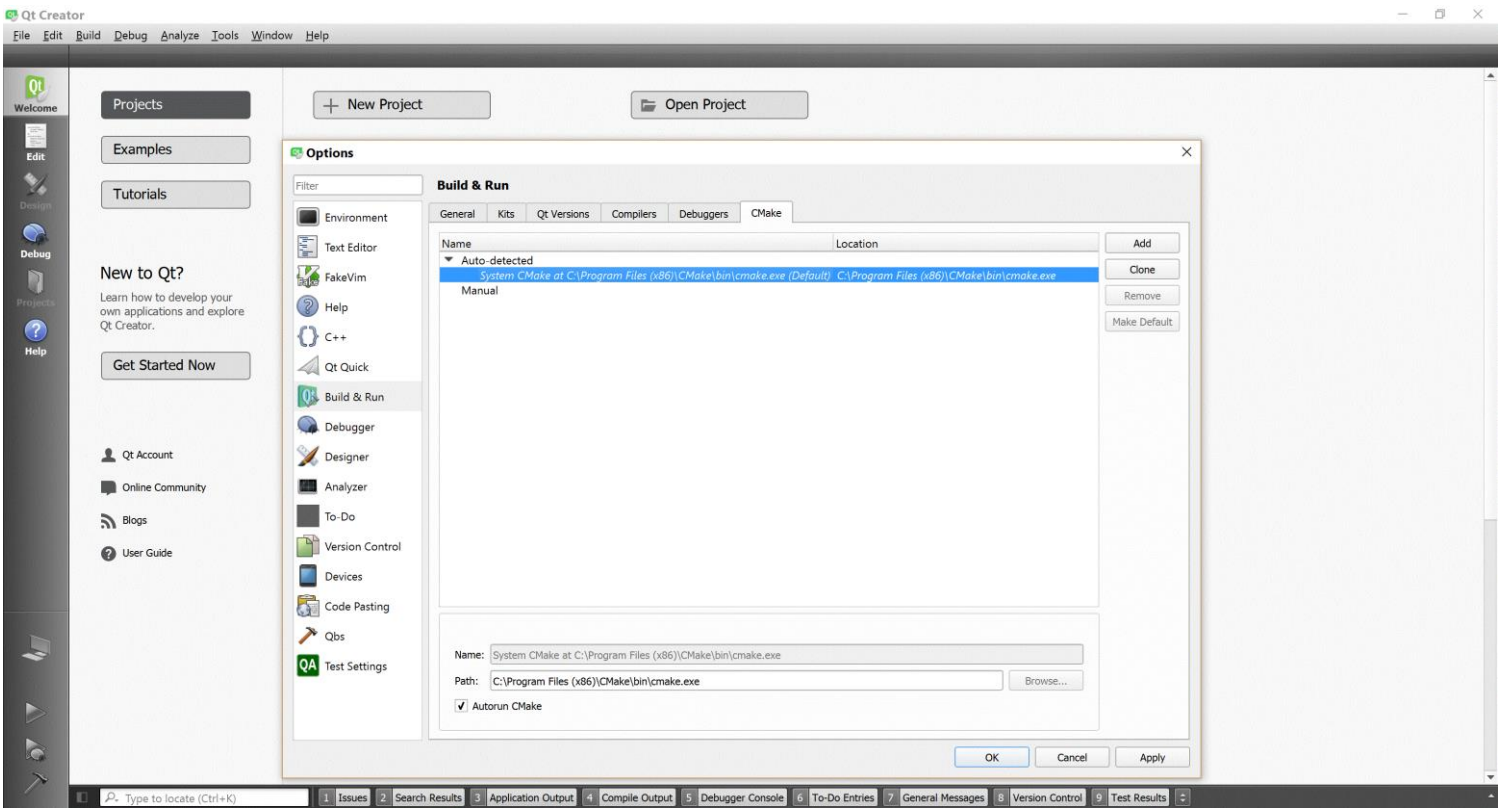
2.2. Използвани инструментални средства за разработка

За изготвянето на поставената курсова задача е използван езикът за програмиране C++. C++ е разширение на програмният език C и е обектно ориентиран език със статични типове. Повечето програми на C могат директно или със съвсем малки модификации да бъдат компилирани с компилатор за езика C++.

Една голяма част от приложните програми на много операционни системи, както и някои от самите операционни системи, са написани на този език. Някои от основните характеристики са, че:

- C++ е моделиран като статичен език за общи цели, като запазва ефикасността и преносимостта на C;
- C++ е създаден да поддържа множество стилове на програмиране (процедурно програмиране, абстракция на данните, обектно-ориентирано програмиране и обобщено програмиране)
- C++ е създаден да дава избор на програмиста, дори той да е неправилен;
- C++ е моделиран като умерен преход от C;
- C++ избягва функции, които са платформенозависими;
- C++ е създаден да работи без сложна среда за разработка.

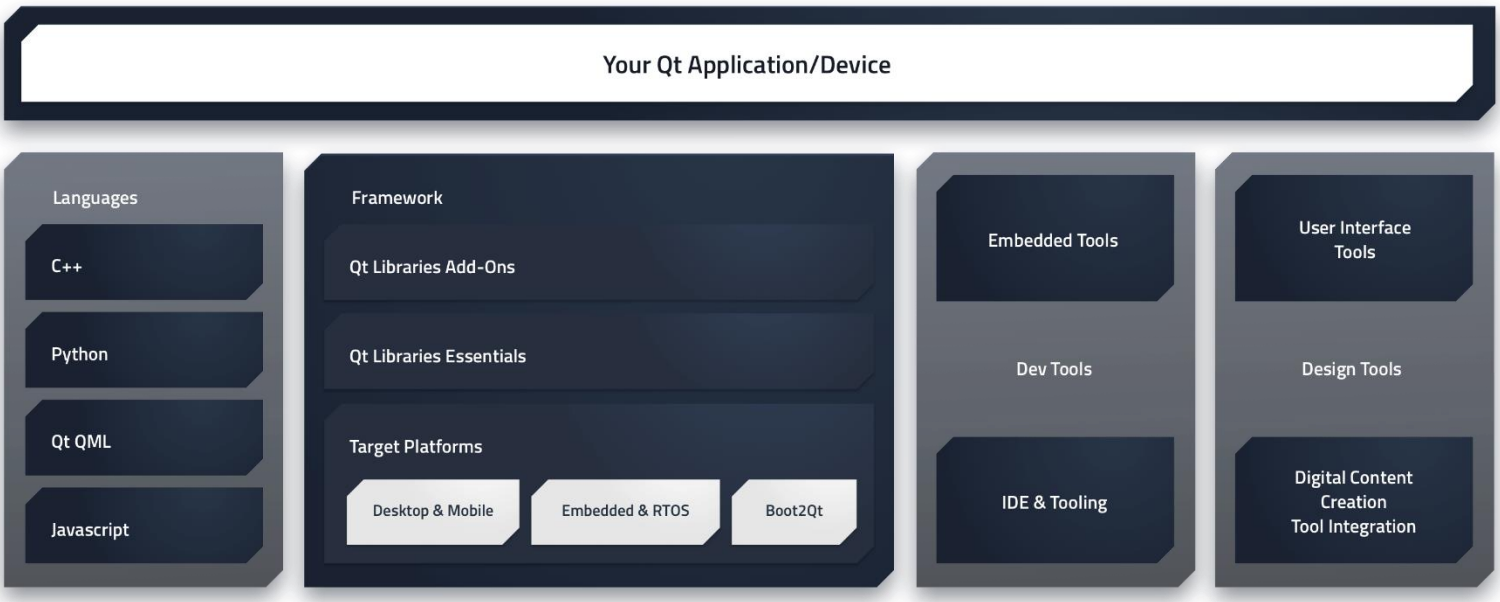
Използваната среда за разработка е Qt Creator – cross-platform интегрирана среда за разработка.



Qt Creator работи на настолни операционни системи Windows, Linux и macOS и позволява на разработчиците да създават софтуер за настолни, мобилни и вградени платформи. Редактор на код на Qt Creator позволява да се пише софтуер както на C++, така и на JavaScript, Python и други езици. Той включва autocomplete, подчертаване на синтаксиса, рефакторинг и има вградена документация с примери. Включена е поддръжка за CMake и кръстосано компилиране с инструменти за изграждане на CMake или qmake. Също така както и в други популярни среди за разработка, тук също има възможност за изграждане на графичен потребителски интерфейс, както и има вграден емулатор, с който може лесно да бъде тествано приложението в условия, практически идентични с тези на даденото целево устройство.

Съвместно с Qt Creator IDE-то е използван и Qt framework, който е написан на C++. Framework-а е нещо като платформа за създаване/разработка на софтуерни решения. Нещо, което ни казва как да се случват нещата и ни поставя някаква „рамка“. При използването на framework често се наблюдава IoC (Inversion of Control), понеже контролът от нас като програмисти, се прехвърля вече върху фреймуърка и той е този който „вика“ нещата (функциите).

Поддържаните платформи от Qt Creator са Linux, OS X, Windows, VxWorks, QNX, Android, iOS, BlackBerry, Sailfish OS и други. Предварителен процесор/Препроцесор, наречен MOC (Meta-Object Compiler), се използва за разширяване на езика C++. Преди стъпката на компилация, MOC анализира изходните файлове, написани на Qt-разширен C++ и генерира от тях стандартно съвместим C++ код. Така цялото приложение, написано чрез този Qt framework може да бъде компилирано от всеки стандартен съвместим C++ компилатор като Clang, GCC, ICC, MinGW и MSVC.



ГЛАВА 3

3.1. Изграждане на TCP Client-Server

Както вече бе споменато по-рано, за осъществяване на комуникацията между вградената система (Raspberry Pi 4) и сървърът, стартиран на PC, се използва TCP/IP протоколът чрез изградена TCP Client-Server връзка.

За целта са изградени два TCP сървъра. Първо се изпълнява кодът на сървъра, с което се отваря порт за видео потока или за данните от GPS и се слуша за входящи заявки от страна на клиента. След като клиента се свърже към един от двата или и двата (сървър) портове, клиентът или сървърът могат да изпращат съобщение. В случая, след като съобщението вече е изпратено от клиента, сървърът, който го получава го обработва и изобразява съответно.

За да се създаде връзката клиент-сървър между системите (Raspberry Pi 4 и PC), се следват съответните конвенции:

- Системите се свързват с подходящ мрежов хардуер и софтуер;
- Системите комуникират помежду си чрез TCP/IP.

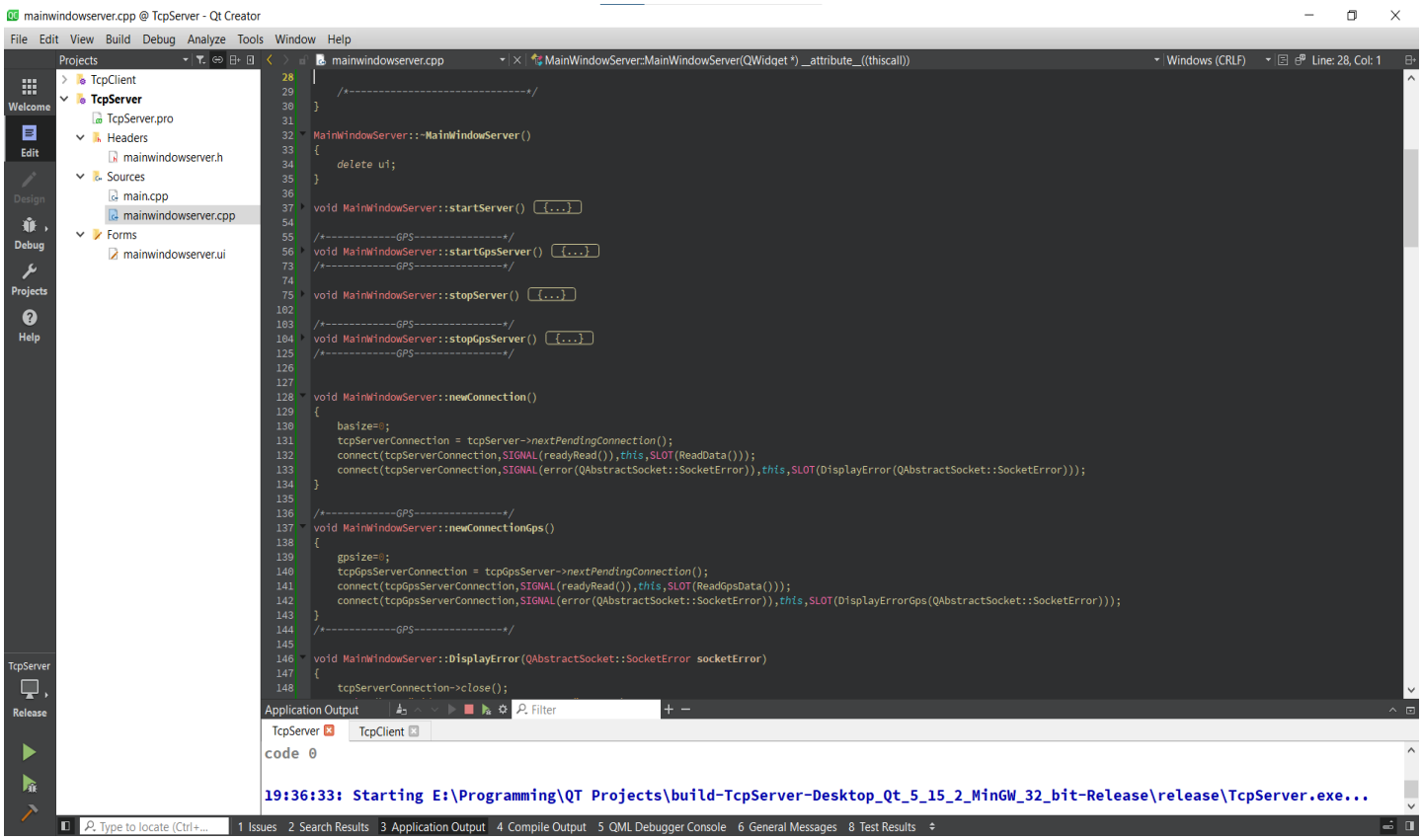
Използвайки тези конвенции, общата процедура за установяване на TCP връзката се състои в издаване на команда OPEN към избран TCP сървър и стартирането му.

При повторно стартиране се извежда съобщение „Server #port already started”.

При неуспешно стартиране (съответният порт вече се използва или поради друга причина) се извежда грешка „Unable to start the server…“. Това се случва при startServer() функцията.

При stopServer() функцията се проверя първо дали на дадения порт, който затваряме се слуша, за да се затвори. В противен случай се извежда съобщение „Server #port already closed“.

След като се затвори сървърът, се затваря и съответната кънекция, ако все още е отворена, а в противен случай отново се извежда подходящо съобщение.

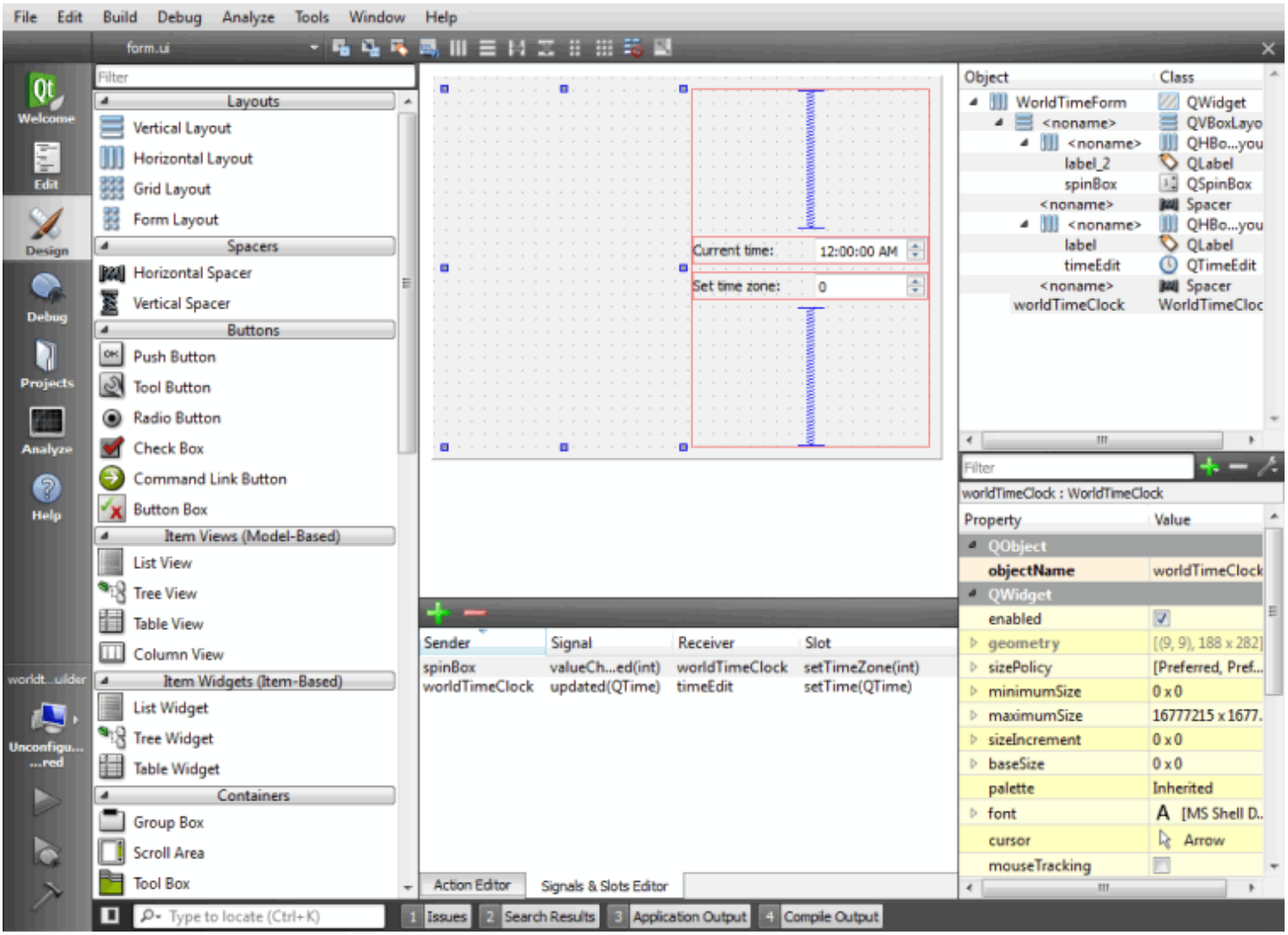


3.2. Създаване на графичен потребителски интерфейс

Графичният потребителски интерфейс на приложението е изграден изцяло благодарение на Qt Creator средата за разработка. Подобно на популярните Visual Studio и Android Studio, Qt Creator има помощни средства, чрез които с избиране и провлачване на даден widget (бутон, текстово поле, място за снимка, формуляр и други) много лесно може да се „сглоби“ изцяло завършен и добре изглеждащ UI.
Могат да бъдат изградени приложения в режим Edit или да се използва отделен визуален редактор, Qt Design Studio.

Всички „джаджи“ се интегрират безпроблемно с програмиран код чрез използване на механизма на Qt сигнали и слотове, който позволява лесно да се присвояват различни поведения на графичните елементи. Всички свойства, зададени в Qt Designer, могат да се променят динамично в кода.

Освен създаването му, редактирането на цвят, видимост, шрифт също са възможни. Чрез Qt Creator може също така да се използват „съветници“, за създаването на Qt Quick проекти, съдържащи стандартен код, който може да се редактира в режим на редактиране.



3.3. Инсталиране и настройка на OpenCV за обработка на изображения

За обработка на изображението в текущата курсова задача се използва OpenCV. OpenCV (Open Source Computer Vision Library) е библиотека от функции за програмиране, насочени главно към компютърно зрение, машинно обучение и обработка на изображения и играе основна роля в работата в реално време, което е от съществено значение за днешните системи. Библиотеката е междуплатформена и безплатна за използване под лицензa Apache 2 с отворен код.

OpenCV е написан на C++ и основният му интерфейс е на C++, но все още запазва и по-старият си C интерфейс. Разработени са и APIs на няколко други езика за програмиране, за да насърчат приемането ѝ от по-голяма аудитория.

OpenCV се стартира на десктоп операционни системи като: Windows, Linux, macOS, FreeBSD, NetBSD, OpenBSD и на мобилни операционни системи като: Android, iOS, Maemo, BlackBerry.

Чрез използването на OpenCV библиотеката става възможно управлението на камерата, настройките на камерата, като: Brightness, Saturation, HUE.) Използвани команди за инсталирането на OpenCV на Raspberry PI

- sudo apt-get install build-essential
- sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev
- sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev
- mkdir opencv
- cd ~/opencv
- git clone https://github.com/opencv/opencv.git
- git clone https://github.com/opencv/opencv_contrib.git
- opencv$ git checkout 4.5.4
- opencv_contrib$ git checkout 4.5.4
- mkdir build
- cd build
- cmake -DCMAKE_BUILD_TYPE=RELEASE -DCMAKE_INSTALL_PREFIX=/usr/local -DCMAKE_EXTRA_MODULES_PATH=../opencv_contrib/modules -DPYTHON3_PACKAGES_PATH=/usr/lib/python3/dist-packages -DBUILD_opencv_java=OFF -DWITH_QT=ON -DOPENCV_GENERATE_PKGCONFIG=ON -DINSTALL_PYTHON_EXAMPLES=OFF -DINSTALL_C_EXAMPLES=OFF -DBUILD_TESTS=OFF -DBUILD_EXAMPLES=OFF -DWITH_OPENMP=ON -DWITH_V4L=ON -DWITH_LIBV4L=ON ../opencv

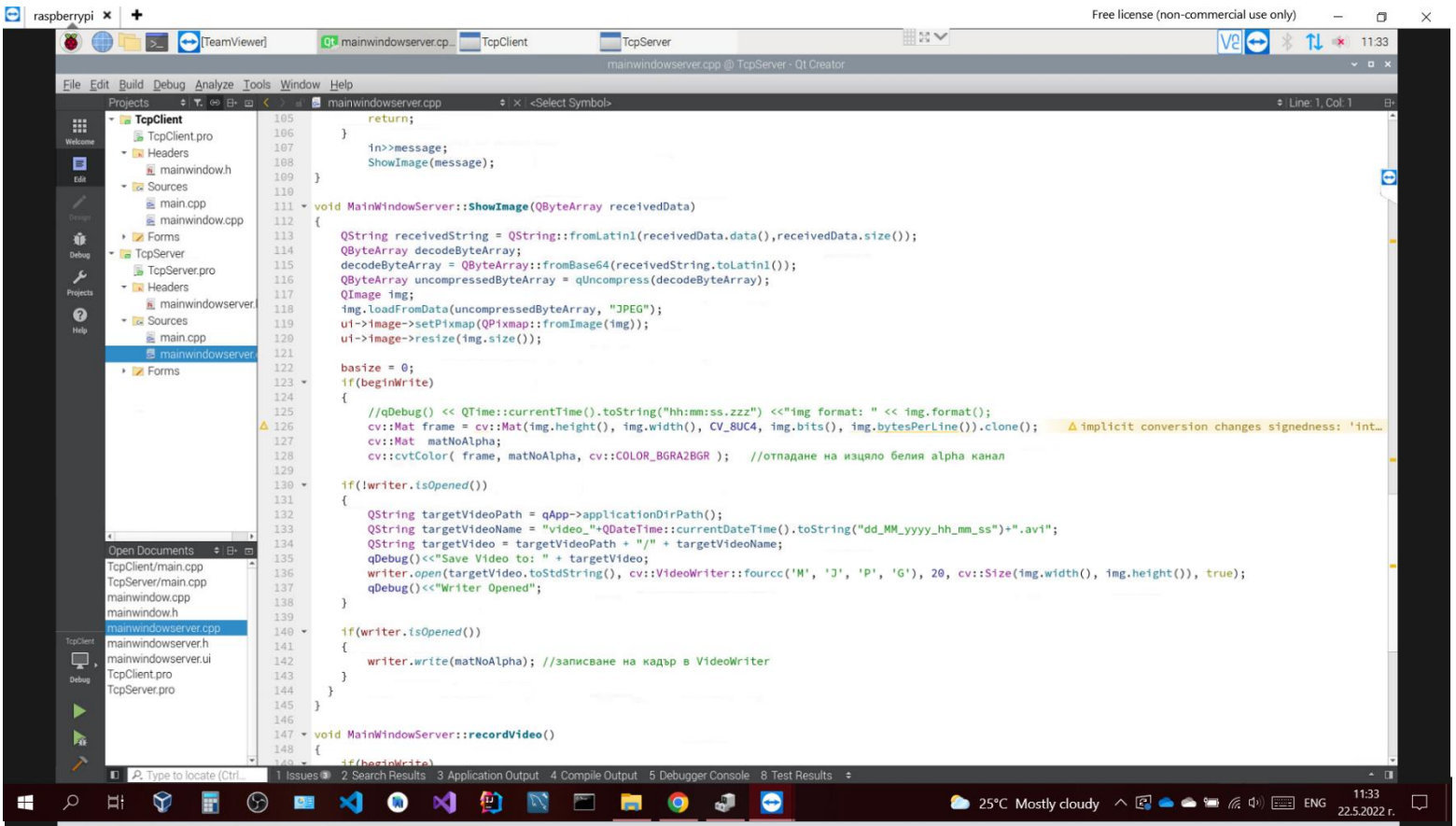
- make -j4 # runs 4 jobs in parallel

- sudo make install

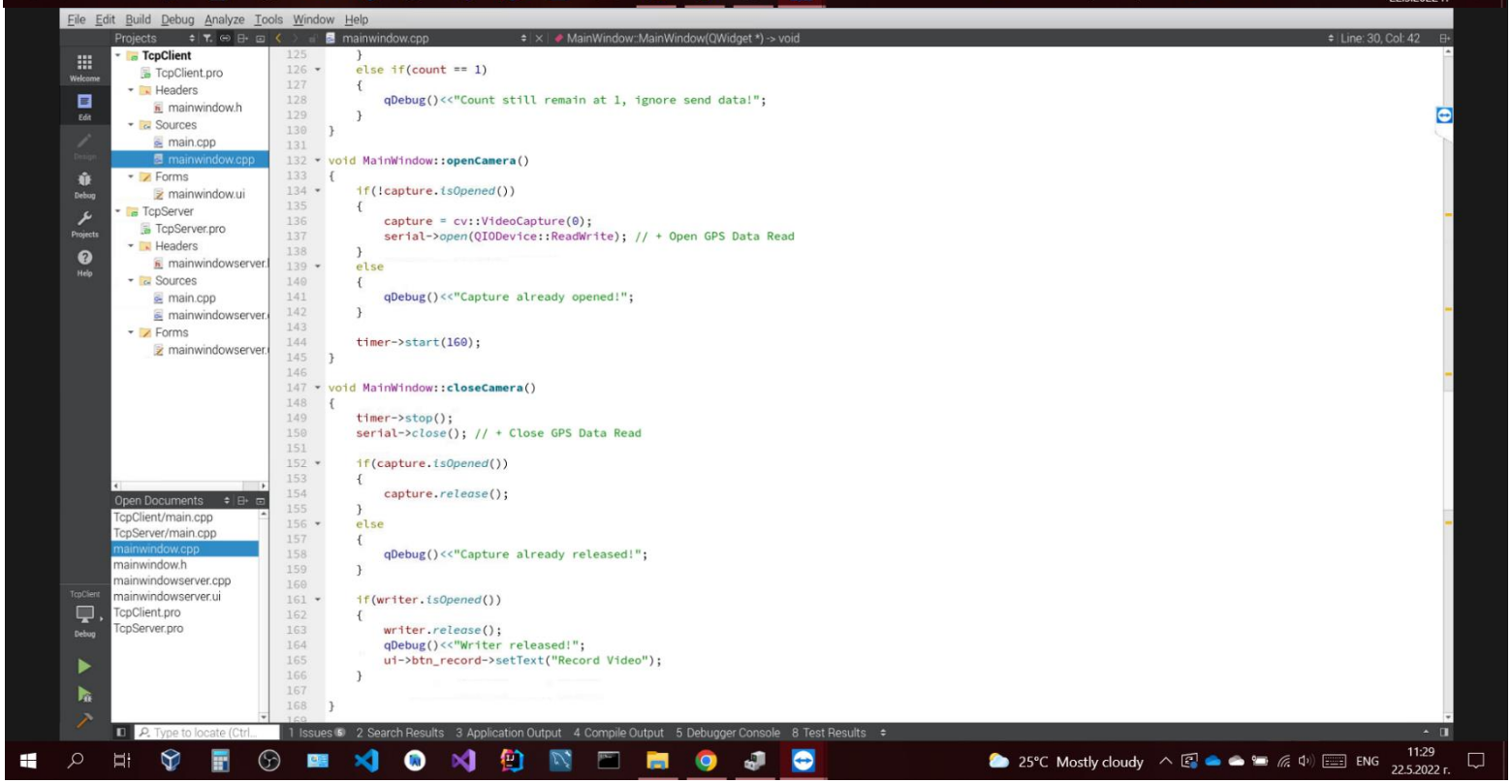
3.4. Инсталиране и настройка на USB камера

Основните характеристика на използваната от нас USB камера бяха вече споменати в *Глава 2*. Използваната камера е избрана, главно заради наличието на USB интерфейс, поради което не е необходимо да се използват странични кабели за връзка.

С помощта на USB камерата се осъществява непрекъснат видео поток от клиента към сървъра през TCP/IP връзката при стартиране на видеото.



```
        return;
    }
        in>>message;
        ShowImage(message);
}

void MainWindowServer::ShowImage(QByteArray receivedData)
{
    QString receivedString = QString::fromLatin1(receivedData.data(),receivedData.size());
    QByteArray decodeByteArray;
    decodeByteArray = QByteArray::fromBase64(receivedString.toLatin1());
    QByteArray uncompressedByteArray = qUncompress(decodeByteArray);
    QImage img;
    img.loadFromData(uncompressedByteArray, "JPEG");
    ui->image->setPixmap(QPixmap::fromImage(img));
    ui->image->resize(img.size());

    basize = 0;
    if(beginWrite)
    {
        //qDebug() << QTime::currentTime().toString("hh:mm:ss.zzz") <<"img format: " << img.format();
        cv::Mat frame = cv::Mat(img.height(), img.width(), CV_8UC4, img.bits(), img.bytesPerLine()).clone();    implicit conversion changes signedness: 'int…
        cv::Mat  matNoAlpha;
        cv::cvtColor( frame, matNoAlpha, cv::COLOR_BGRA2BGR );   //отпадане на изцяло белия alpha канал

    if(!writer.isOpened())
    {
        QString targetVideoPath = qApp->applicationDirPath();
        QString targetVideoName = "video_"+QDateTime::currentDateTime().toString("dd_MM_yyyy_hh_mm_ss")+".avi";
        QString targetVideo = targetVideoPath + "/" + targetVideoName;
        qDebug()<<"Save Video to: " + targetVideo;
        writer.open(targetVideo.toStdString(), cv::VideoWriter::fourcc('M', 'J', 'P', 'G'), 20, cv::Size(img.width(), img.height()), true);
        qDebug()<<"Writer Opened";
    }

    if(writer.isOpened())
    {
        writer.write(matNoAlpha); //записване на кадър в VideoWriter
    }
  }
}

void MainWindowServer::recordVideo()
{
    if(beginWrite)
```



```
    }
    else if(count == 1)
    {
        qDebug()<<"Count still remain at 1, ignore send data!";
    }
}

void MainWindow::openCamera()
{
    if(!capture.isOpened())
    {
        capture = cv::VideoCapture(0);
        serial->open(QIODevice::ReadWrite); // + Open GPS Data Read
    }
    else
    {
        qDebug()<<"Capture already opened!";
    }

    timer->start(160);
}

void MainWindow::closeCamera()
{
    timer->stop();
    serial->close(); // + Close GPS Data Read

    if(capture.isOpened())
    {
        capture.release();
    }
    else
    {
        qDebug()<<"Capture already released!";
    }

    if(writer.isOpened())
    {
        writer.release();
        qDebug()<<"Writer released!";
        ui->btn_record->setText("Record Video");
    }

}
```

Данните се записват и изпращат благодарение на имплементираните функции openCamera(), closeCamera(), readData(), showImage().

3.5. Инсталиране и настройка на MOD-GPS

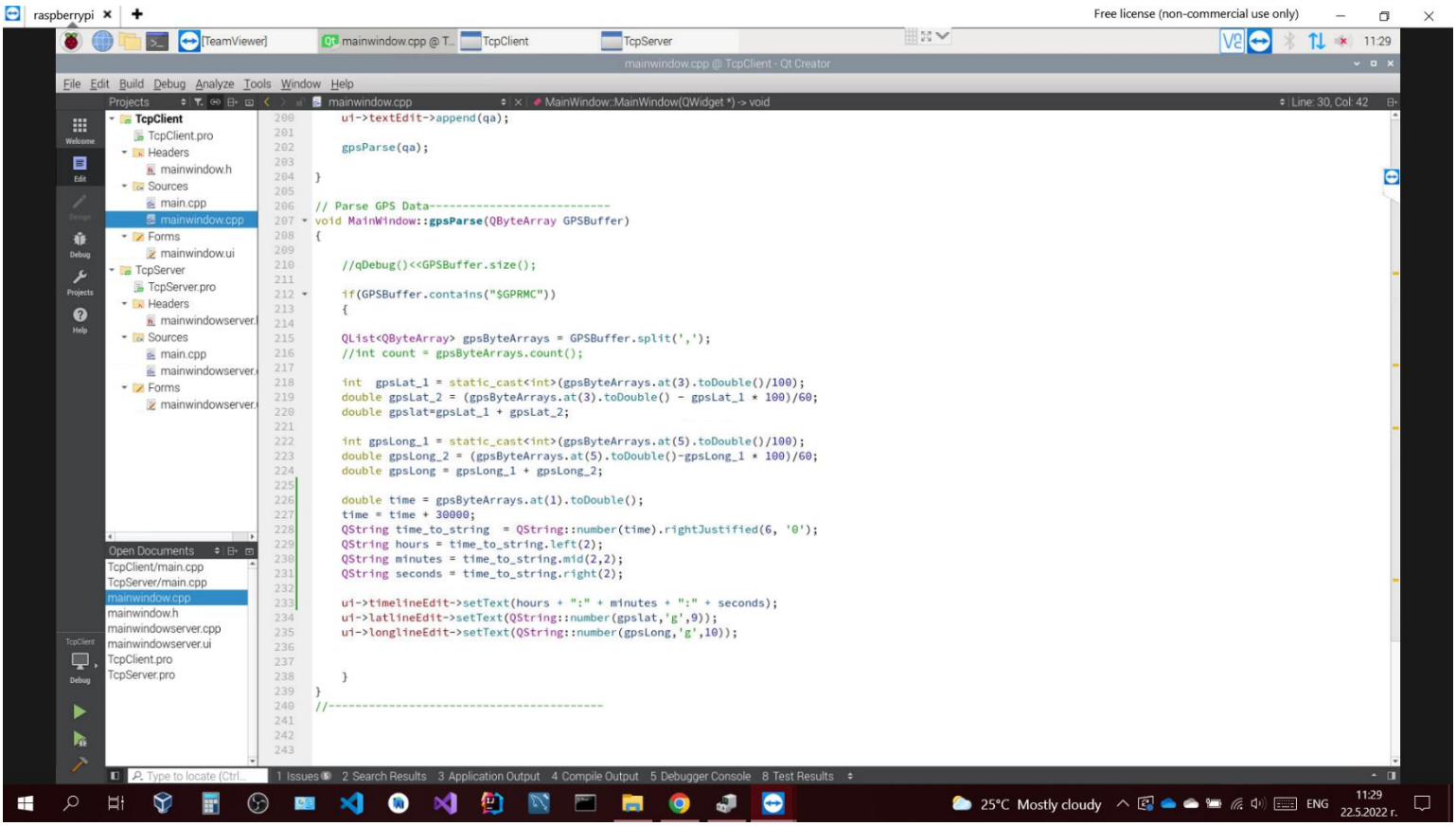
Основните характеристика на използвания от нас GPS модул бяха вече споменати в *Глава 2*. Mulberry USB G-Mouse GPS е избран отново, поради наличието на USB интерфейс.

С помощта на GPS устройството се прехващат данните от тип GPRMC и се елиминира излишната ни информация с помощта на функцията gpsParse().

Това се случва с помота на един ByteArray, в който се пълнят всички данни от тип GPRMC, след което от този масив се изваждат координатите, които изобразяваме на крайния потребител – latitude(географска ширина), longitude(географска дължина) и времето в часове, минути и секунди. Цялата тази информация, след основна проста обработка се изобразява в съответните полета.

ИНИЦИАЛИЗИРАНЕ НА СЕРИЕН ПОРТ (Raspberry PI)

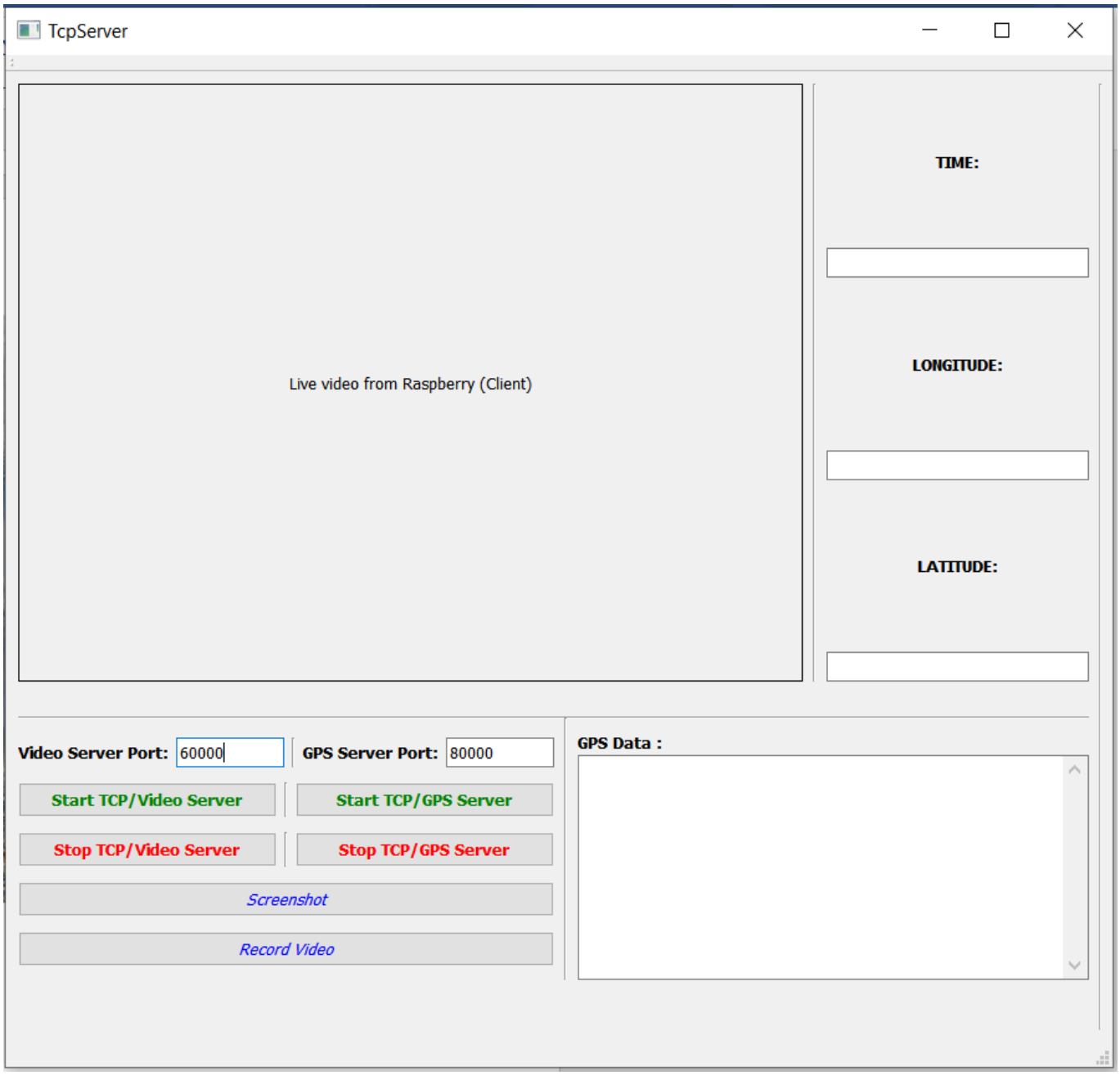
```
// init serial port ---------------------------------------- GPS
connect(serial, SIGNAL(readyRead()), this, SLOT(serialReadSendData()));
serial->setPortName("/dev/ttyACM0"); //
serial->setBaudRate(QSerialPort::Baud9600);
serial->setDataBits(QSerialPort::Data8);
serial->setParity(QSerialPort::NoParity);
serial->setStopBits(QSerialPort::OneStop);
serial->setFlowControl(QSerialPort::NoFlowControl);
//serial->open(QIODevice::ReadWrite);
//serial->write("Working");
// init serial port ---------------------------------------- GPS
```



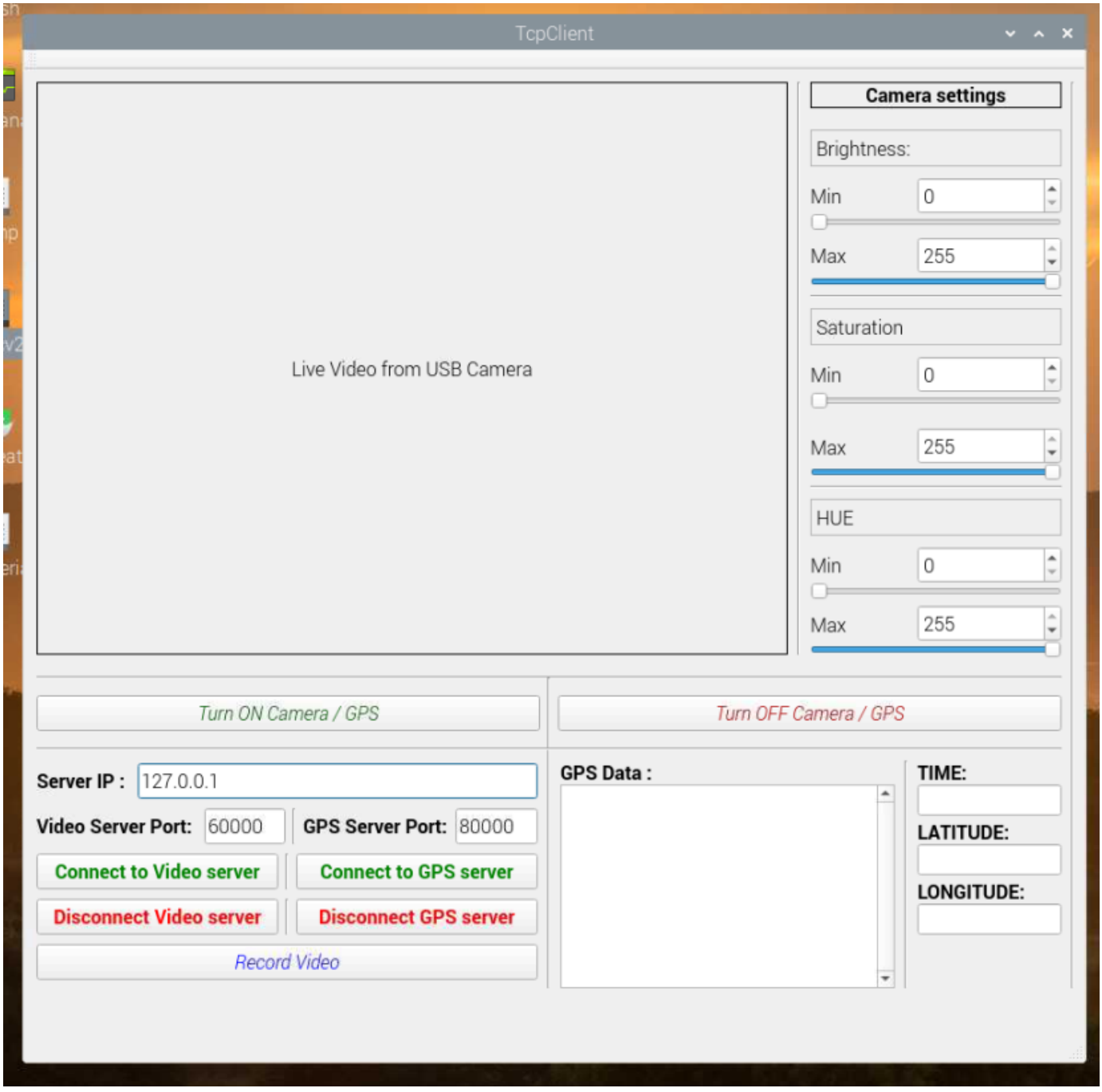
Допълнително нещо, което може да се направи тук е да се елиминира излишният поток от GPS данните, които текат по мрежата чрез проверка за промяна на координатите, тоест да се изпращат координати, само и единствено при промяна на местоположението. Това обаче е една идея по-сложно(tricky), понеже GPS модулът има някакво отклонение, чрез което всяка секунда се изпращат променени GPS координати (с променени десетки или стотни от единицата), поради което е много трудно да се прехванат напълно идентични стойности.

3.6. Краен резултат от работата (снимки)

TCP - СЪРВЪР



TCP - КЛИЕНТ


TcpClient
Live Video from USB Camera
Camera settings
Brightness:
Min
0
Max
255
Saturation
Min
0
Max
255
HUE
Min
0
Max
255
Turn ON Camera / GPS
Turn OFF Camera / GPS
Server IP :
127.0.0.1
Video Server Port:
60000
GPS Server Port:
80000
Connect to Video server
Connect to GPS server
Disconnect Video server
Disconnect GPS server
Record Video
GPS Data :
TIME:
LATITUDE:
LONGITUDE: