CENTRO UNIVERSITARIO DE CIENCIAS EXÁCTAS E INGENIERÍAS

Cámara Ip y Bot de Correo

Práctica 01

Aldo Alexandro Vargas Meza

20/09/2017

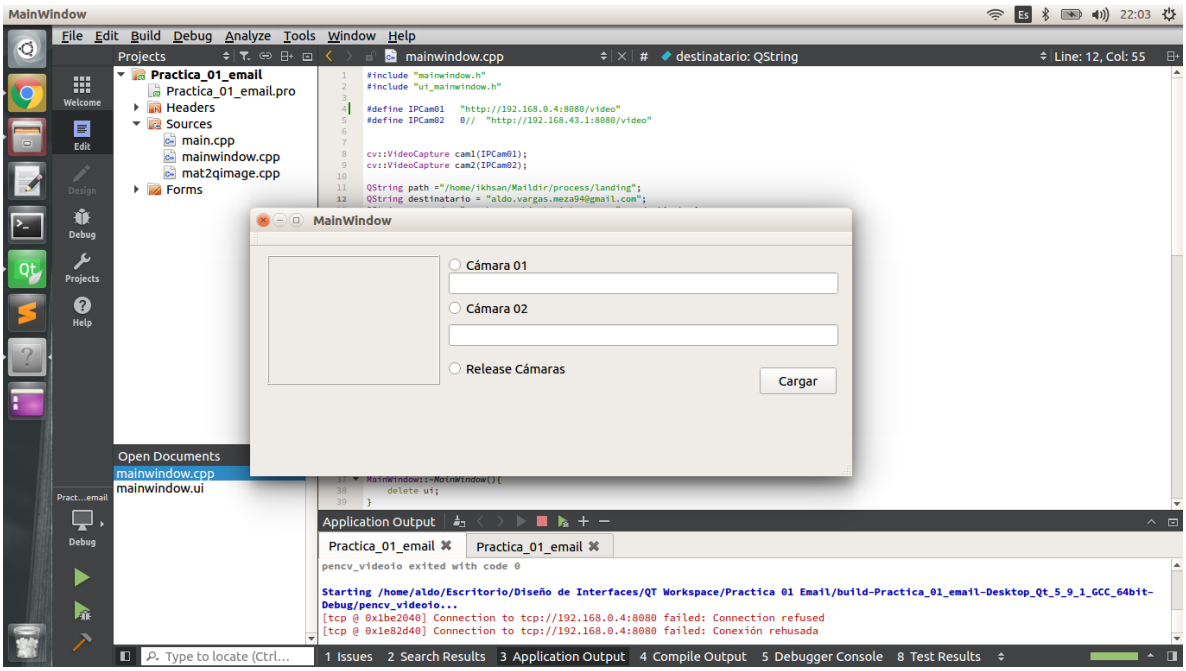
El objetivo es visualizar dos cámaras IP. La visualización de la cámara IP se controlará mediante elementos de entrada (botones, botones radiales). Las direcciones IP de la cámara se podrán modificar, mediante elementos de entrada de texto en QT.

En resumen, el programa toma control de la cámara IP que se le asigne en un principio, la cual será la cámara incorporada en la laptop. Después, iterando un par de funciones, obtenemos un comportamiento que depende enteramente de los botones radiales colocados en la interfaz, con los cuales se selecciona una cámara u otra, así como la reiniciación de las mismas, con el tercer botón.

Los campos para añadir texto, son cargados por una función ejecutada desde el botón CARGAR.

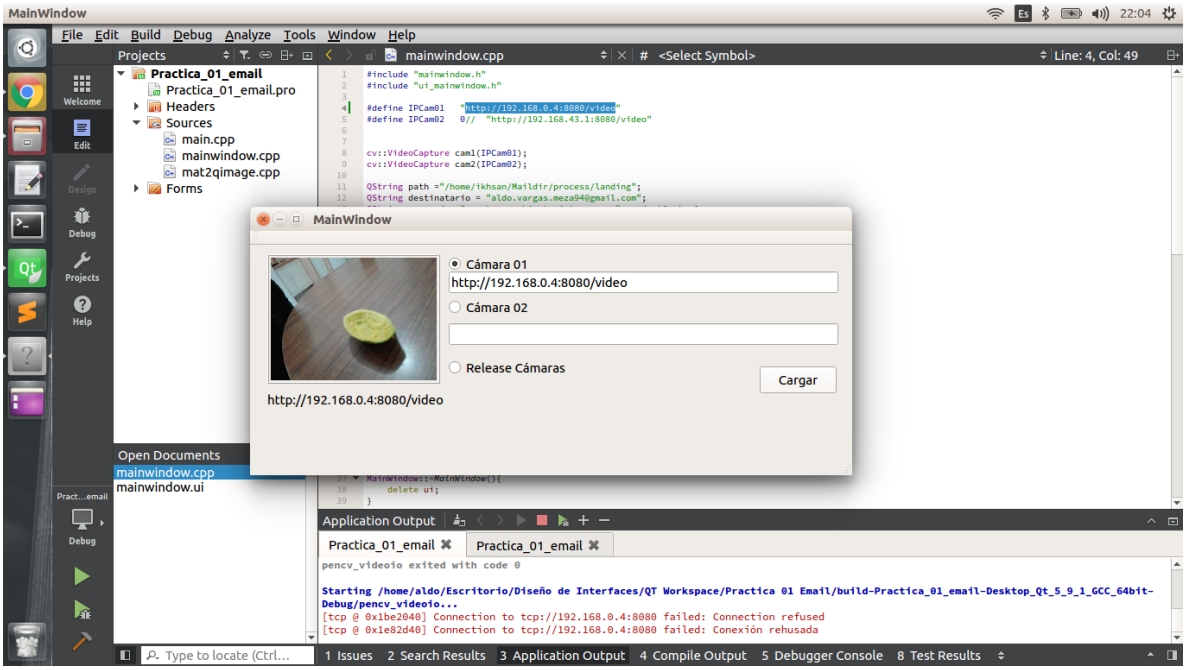
Con esta pequeña interfaz, podemos cambiar de cámaras en una red local configurada, con una entrada de texto.

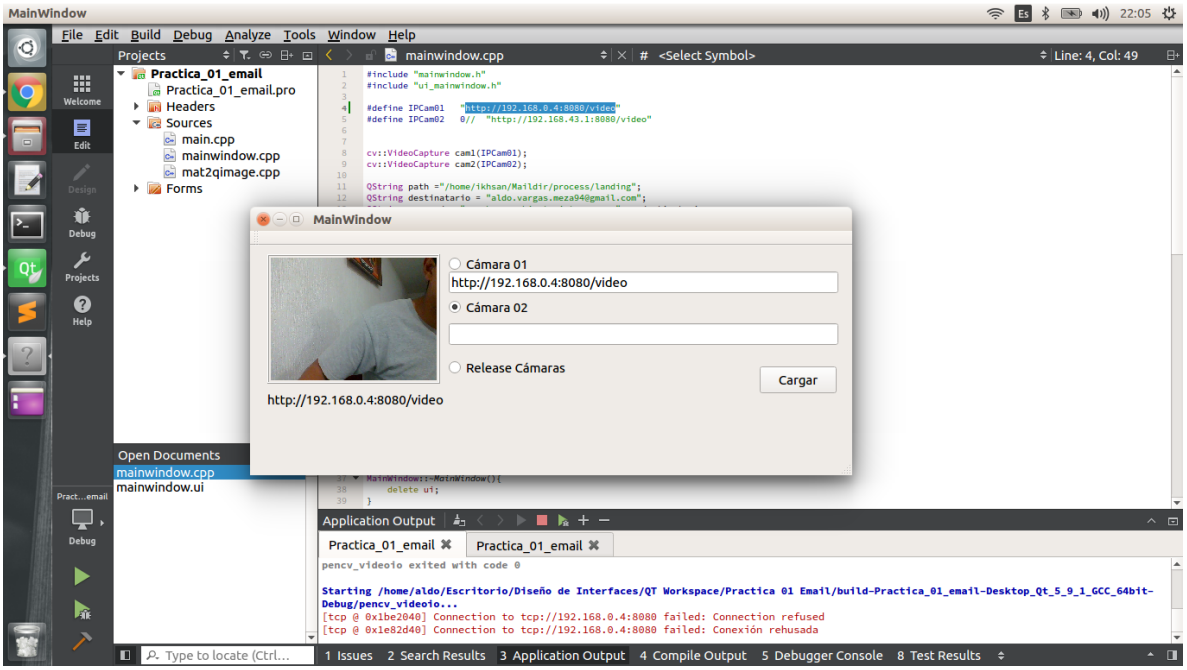
Una vez añadidos los archivos y librerías necesarias para el funcionamiento de Open CV y Mat2Qimage, el programa comienza inicializando dos camaras, que están conectadas por medio de un servidor IP a la computadora que ejecuta el programa.



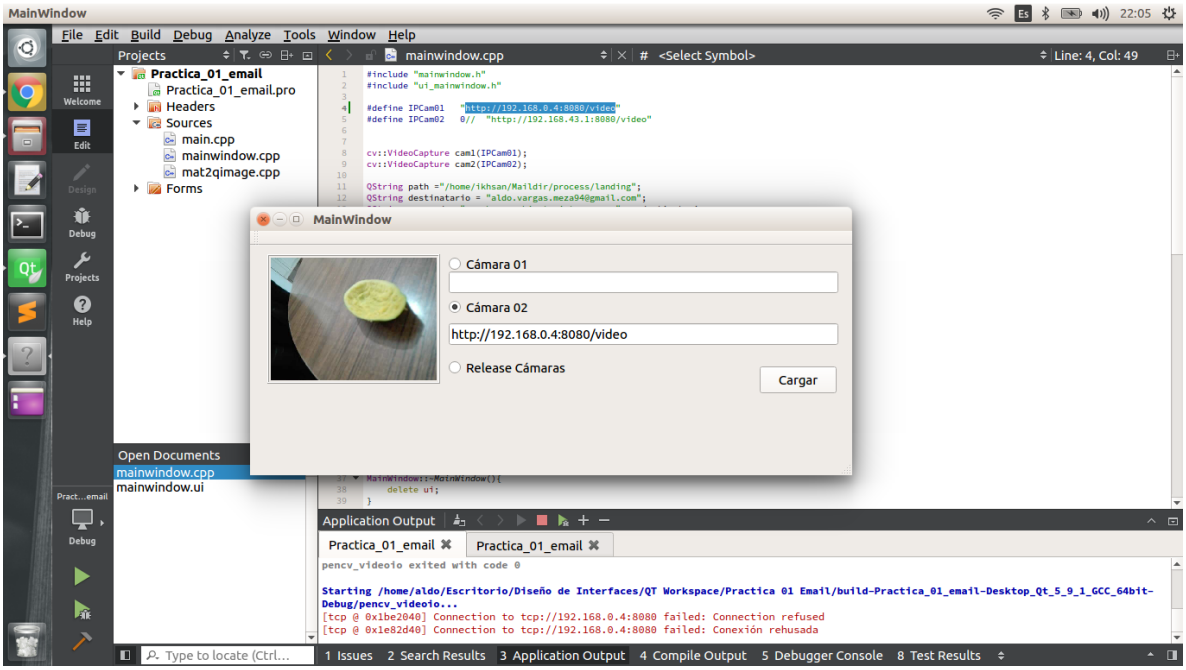
En el programa principal, se declaran 2 timmers, los cuales ejecutan una función cada uno, definida como loop y loop2, cada 30 y 100 milisegundos, respectivamente.

La función loop, se encarga de checar la posición de los botones radiales, los cuales ejecutarán acciones diferentes ya establecidas, las cuales son prácticamente las mismas, pero ejecutadas en distintas cámaras.



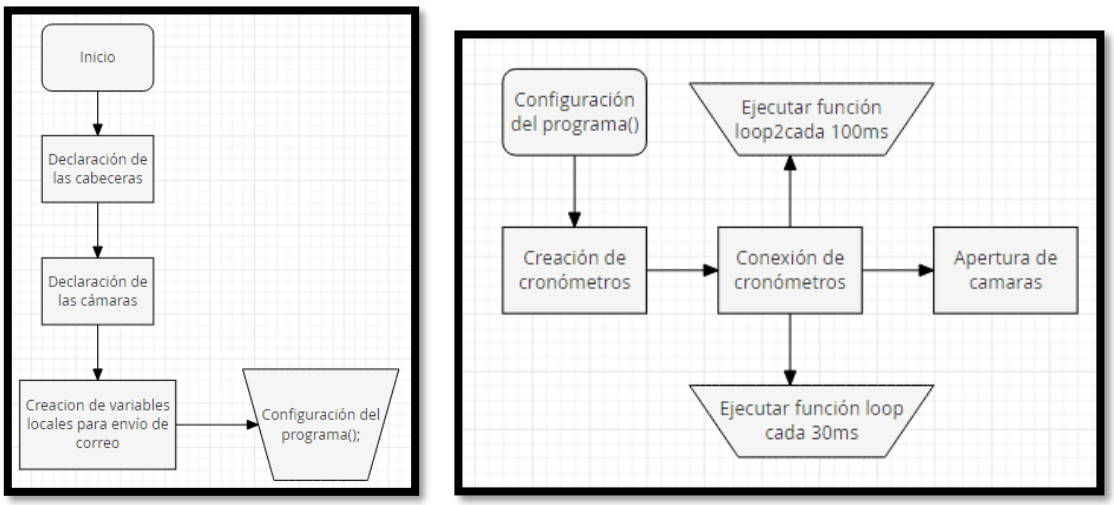


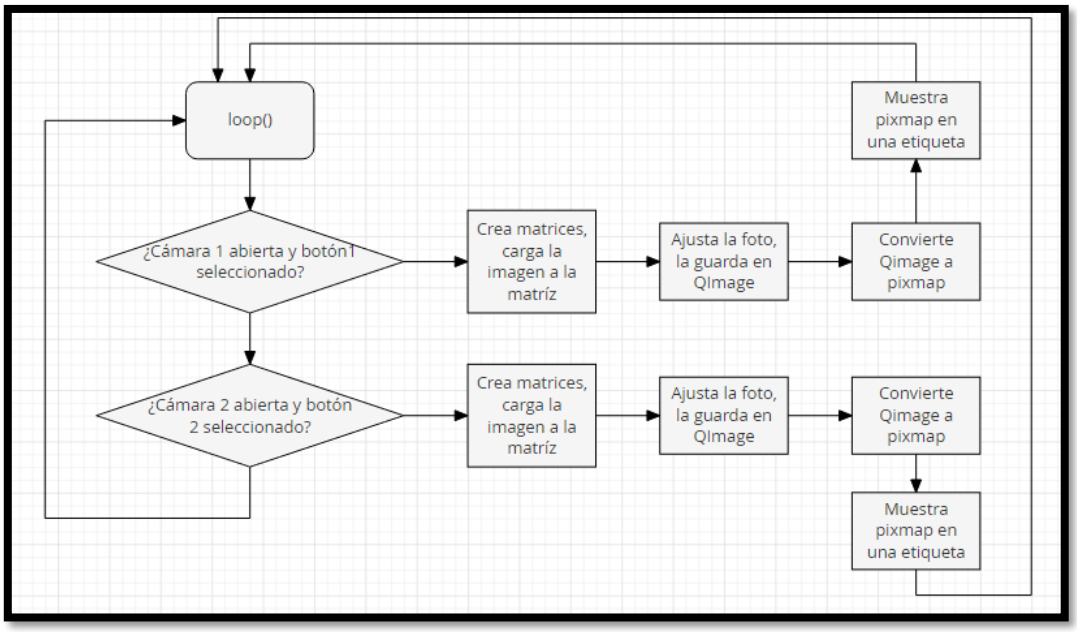
La función loop2 se encarga de crear un archivo con la imagen tomada, para después ser adjuntado a un posible envío de correo. Esta función mandaría un correo a un destinatario ya preestablecido.

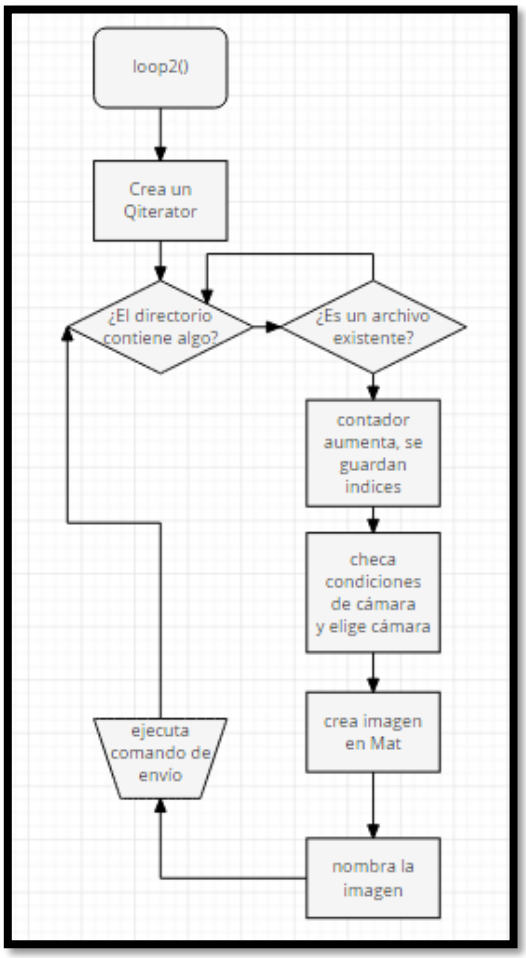


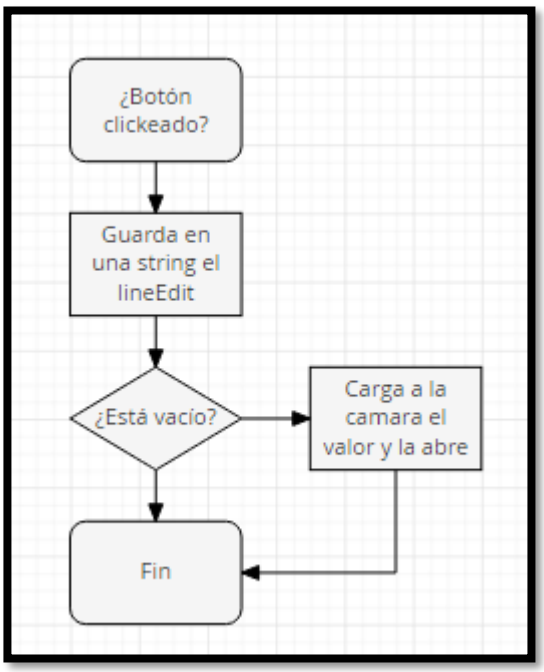
Finalmente, el botón cargar, reinicia la cámara seleccionada, con la dirección que se establezca en los campos de escritura.

El funcionamiento principal de la práctica está descrito en el código MainWindow el cual está representado en el siguiente diagrama de flujo:




loop()
¿Cámara 1 abierta y botón1 seleccionado?
Crea matrices, carga la imagen a la matríz
Ajusta la foto, la guarda en QImage
Convierte Qimage a pixmap
Muestra pixmap en una etiqueta
¿Cámara 2 abierta y botón 2 seleccionado?
Crea matrices, carga la imagen a la matríz
Ajusta la foto, la guarda en QImage
Convierte Qimage a pixmap
Muestra pixmap en una etiqueta


loop2()
Crea un Qiterator
¿El directorio contiene algo?
¿Es un archivo existente?
contador aumenta, se guardan indices
checa condiciones de cámara y elige cámara
crea imagen en Mat
nombra la imagen
ejecuta comando de envío


¿Botón clickeado?
Guarda en una string el lineEdit
¿Está vacío?
Carga a la camara el valor y la abre
Fin

Codigos

```
#include "mainwindow.h"

#include "ui_mainwindow.h"


#define IPCam01   "http://192.168.0.4:8080/video"

#define IPCam02   0//  "http://192.168.43.1:8080/video"


// Declaracion de camaras

cv::VideoCapture cam1(IPCam01);

cv::VideoCapture cam2(IPCam02);


// Variables para envío de correo

QString path ="/home/ikhsan/Maildir/process/landing";

QString destinatario = "aldo.vargas.meza94@gmail.com";

QString command = "mpack -s subject picture.png "  + destinatario;


// Contador para archivos

int count;



MainWindow::MainWindow(QWidget *parent) :

   QMainWindow(parent),
```

```
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    // Creacion de timmers
    QTimer* cronometro = new QTimer(this);
    // Ejecucon de funcion cada ms
    connect(cronometro, SIGNAL(timeout()), this, SLOT(loop()));
    cronometro->start(30);

    QTimer* cronometro2 = new QTimer(this);
    connect(cronometro2, SIGNAL(timeout()), this, SLOT(loop2()));
    cronometro2->start(100);
    // Aseguramiento de camaras abiertas
    if(!cam1.isOpened() || !cam2.isOpened())
    {
       cam1.open(IPCam01);
       cam2.open(IPCam02);
    }

    // Limpia la etiqueta
    ui->label->clear();
```

```
}


MainWindow::~MainWindow()
{
   delete ui;
}


void MainWindow::loop()
{

   // Si la camara seleccionada esta abierta
   if(cam2.isOpened() && ui->radioButton_2->isChecked())
   {
      // Crea una matriz
      Mat img1, img2;
      // Guarda la imagen ahi
      cam2 >> img1;
      // Ajusta tamaño
      cv::resize(img1,img2,Size(200,150),0,0,0);
      // Convierte a Qimage
      QImage qImage = Mat2QImage(img2);
      // Mapa de pixeles
```

```
        QPixmap pixmap = QPixmap::fromImage(qImage);
        // Muestra por la etiqueta
        ui->label->clear();
        ui->label->setPixmap(pixmap);

    }
    else if(cam1.isOpened() && ui->radioButton->isChecked())
    {
        Mat img1, img2;
        cam1 >> img1;
        cv::resize(img1,img2,Size(200,150),0,0,0);
        QImage qImage = Mat2QImage(img2);
        QPixmap pixmap = QPixmap::fromImage(qImage);

        ui->label->clear();
        ui->label->setPixmap(pixmap);

    }
}

void MainWindow::loop2()
{
```

```
// Crea el iterator
QDirIterator dir(path, QDir::NoFilter);
count = 0;
// Si hay archivos
while(dir.hasNext())
{
   // Crea archivo con credenciales
   QFileInfo infoArchivo(dir.next());
   // si existe y es archivo
   if(infoArchivo.isFile() && infoArchivo.exists())
   {
      count++;
      // Guarda indices del destinatario
      int arr = destinatario.indexOf("@");
      int com = destinatario.indexOf(".com");
      // Si esta correcta la escritura
      if(arr>=0 && com>= 0)
      {
         // Crea ua matriz
         Mat IMAGEN;
         // El botón que este seleccionado
         if(ui->radioButton->isChecked())
```

```
        {
            // Carga la camara a la imagen
            cam1 >> IMAGEN;
        }
        else if(ui->radioButton_2->isChecked())
        {
            cam2 >> IMAGEN;
        }
        else
        {
            ui->label->clear();
        }
        // Nombra el archivo
        cv::imwrite("picture.png",IMAGEN);
        // Ejecuta el comando
        system(command.toUtf8().constData());
      }

   }

}
```

```
}

void MainWindow::on_radioButton_3_clicked()
{
    // Limpia las camaras y la etiqueta
    cam1.release();
    cam2.release();
    ui->label->clear();
}

void MainWindow::on_pushButton_3_clicked()
{
    // Guarda los linedit en variables cadena
    QString camP1 = ui->lineEdit->text();
    ui->label_2->setText(camP1);
    QString camP2 = ui->lineEdit_2->text();

    // si hay algo escrito, lo carga a la camara
    if(ui->radioButton->isChecked() && !camP1.isEmpty())
    {
        cam1.open(camP1.toUtf8().constData());
```

```
    }
    else if(ui->radioButton_2->isChecked() && !camP2.isEmpty())
    {
        cam2.open(camP2.toUtf8().constData());

    }

}

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

// Incluye al proyecto los archivos y cabeceras

#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "mat2qimage.h"
#include<qdebug.h>

#include<opencv2/core/core.hpp>
#include<opencv2/ml/ml.hpp>
```

```
#include<opencv/cv.h>

#include<opencv2/imgproc/imgproc.hpp>

#include<opencv2/highgui/highgui.hpp>

#include<opencv2/video/background_segm.hpp>

#include<opencv2/videoio.hpp>

#include<opencv2/imgcodecs.hpp>


#include<QTimer>

#include<QDirIterator>

#include<QFileInfo>

#include<unistd.h>


// Utiliza funciones de OPENCV

using namespace cv;


namespace Ui

{

class MainWindow;

}


class MainWindow : public QMainWindow

{
```

```
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

public slots:
    // Funciones definidas por el programador
    void loop();
    void loop2();
private slots:
    // Funciones de la interfaz

    void on_radioButton_3_clicked();

    void on_pushButton_3_clicked();

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

```
#-------------------------------------------------

#

# Project created by QtCreator 2017-09-18T23:44:25

#

#-------------------------------------------------


QT       += core gui


greaterThan(QT_MAJOR_VERSION, 4): QT += widgets


# Nombre del archivo de salida

TARGET = pencv_videoio

TEMPLATE = app


# The following define makes your compiler emit warnings if you use

# any feature of Qt which as been marked as deprecated (the exact warnings

# depend on your compiler). Please consult the documentation of the

# deprecated API in order to know how to port your code away from it.

DEFINES += QT_DEPRECATED_WARNINGS


# You can also make your code fail to compile if you use deprecated APIs.

# In order to do so, uncomment the following line.
```

```
# You can also select to disable deprecated APIs only up to a certain version of Qt.

#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0



SOURCES += \

    main.cpp \

    mainwindow.cpp \

    mat2qimage.cpp



HEADERS += \

    mainwindow.h \

    mat2qimage.h



FORMS += \

    mainwindow.ui



# Config OPEN CV

INCLUDEPATH += /usr/local/include/opencv2



LIBS += -L/usr/local/lib -lopencv_core -lopencv_imgcodecs -lopencv_highgui -opencv_videoio
```

```
CONFIG += link_pkgconfig

PKGCONFIG += opencv
```