CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

INGENIERIA EN COMUNICACIONES Y ELECTRONICA

Diseño de interfaces I7262 D-02

Dr. Rubén Estrada Marmolejo
Alumno: Aguilar Rodríguez Carlos Adolfo
Código: 215860049

Tarea 4
Encontrar coordenadas del área más grande y mostrar en el centro.

Viernes 24 de Mayo del 2019

Código mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.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"mat2qimage.h"
#include<QTimer> // Las funciones de cronometro
#include<QDebug>  //Imprimir mensajes en la terminal
#include<QtSerialPort>
#include<QSerialPortInfo>
#include<QtNetwork>
#include<QFile>
#include<QTextStream>
#include<QDateTime>
#include<QDate>
#include <QJsonDocument>
using namespace cv;
using namespace std;

VideoCapture camara(0);
//VideoCapture camara("http://192.168.43.1:8080/video");


void MainWindow::recepcionSerialAsyncrona(){
    if(arduino_esta_conectado && arduino->isReadable()){
        //QByteArray datosLeidos = arduino->readAll();
        QByteArray datosLeidos = arduino->readLine();
        int indice0 = datosLeidos.indexOf("}");
        int indice1 = datosLeidos.indexOf("{");
        QString infoExtraida = datosLeidos.mid(indice1,(indice0-indice1+1));
       // ui->lcdNumber->display(datoEntero);
        if(indice0>=0 && indice1>=0) {
                qDebug() << "Datos extraidos: " <<
infoExtraida.toUtf8().constData();

               //Convertir el QString en una cadenad de bytes
                QByteArray datosEnBytes = infoExtraida.toUtf8().constData();


                //Se convierte la variable anterior a un documento JSON
                QJsonDocument jsonDocumento =
QJsonDocument::fromJson(datosEnBytes);

                //Convertir el documento JSON a un objeto JSON
                QJsonObject jsonObjeto = jsonDocumento.object();

                //Decodifica la primera variable, se llama X
                QJsonValue value = jsonObjeto.value(QString("valor"));


                //La variable decodificada value, se convierte a entero.
                //int variable1Json =  value.toInt();

                //La variable decodificada value, se convierte a un string.
                QString variable1Json = value.toString();

                //QDateTime  tiempoUTC = QDateTime::currentDateTimeUtc();
                QDateTime  tiempoUTC = QDateTime::currentDateTime();

                QString tiempo = tiempoUTC.toString("dd-MM-yyyy hh:mm:ss");
```

```
                QString nombreArchivo = "datalogger1.csv";

                    QFile archivo(nombreArchivo);

                        if(archivo.open(QIODevice::WriteOnly |
QIODevice::Text | QIODevice::Append)){
                            QTextStream datosArchivo(&archivo);
                            datosArchivo << tiempo << "," << variable1Json <<
endl;

                        }
                        archivo.close();

        }

    }

}

void MainWindow::conectarArduino(){


    //Parte # 1, declaración inicial de las variables
    arduino_esta_conectado = false;
    arduino = new QSerialPort(this);
    connect(arduino, &QSerialPort::readyRead, this,
&MainWindow::recepcionSerialAsyncrona);

    QString nombreDispositivoSerial = "";
    int nombreProductID = 0;

    //-2 buscar puertos con los identificadores de Arduino
    qDebug() << "Puertos disponibles: " <<
QSerialPortInfo::availablePorts().length();
    foreach (const QSerialPortInfo &serialPortInfo,
QSerialPortInfo::availablePorts()) {
        qDebug() << "Identificador del fabricante (VENDOR ID): " <<
serialPortInfo.hasVendorIdentifier();
        if(serialPortInfo.hasVendorIdentifier()){
            qDebug() << "ID Vendedor " << serialPortInfo.vendorIdentifier();
            qDebug() << "ID Producto: " << serialPortInfo.productIdentifier();

            if(serialPortInfo.productIdentifier() == 66 ||
serialPortInfo.productIdentifier() == 67){
                arduino_esta_conectado = true;
                nombreDispositivoSerial = serialPortInfo.portName();
                nombreProductID = serialPortInfo.productIdentifier();
            }
        }

    }

    //3-Conexion
    if(arduino_esta_conectado){

        arduino ->setPortName(nombreDispositivoSerial);
        arduino->open(QIODevice::ReadWrite);
        arduino->setDataBits(QSerialPort::Data8);
        arduino ->setBaudRate(QSerialPort::Baud115200);
        arduino->setParity(QSerialPort::NoParity);
        arduino->setStopBits(QSerialPort::OneStop);
        arduino->setFlowControl(QSerialPort::NoFlowControl);
         ui->label_2->clear();
        qDebug() << "Producto: " << nombreProductID;
        if(nombreProductID == 67) ui->label_2->setText("Arduino UNO R3
conectado");
```

```
        else if(nombreProductID == 66) ui->label_2->setText("Arduino Mega
conectado");
        else ui->label_2->setText("Error 3");

    }
    else{
        ui->label_2->clear();
        ui->label_2->setText("No hay arduino");
    }
}

void MainWindow::cronometro(){
  //qDebug() << "La funcion se ejecuto ";

  Mat imagen;  // La variable donde se guarda la imagen de la camara
  Mat imagenChica;  // Sera donde se guarde el cambio de tamao de la imagen
orignal

  // Obtener la imagen de la camara y guardarla en una variable

  camara >> imagen;

  if(!imagen.empty()){
      //Aplica un cambio de tamao usando la funcion resize.
      cv::resize(imagen,imagenChica,Size(250,250),0,0,INTER_LINEAR);

      // Cambiar la imagen de opencv a una imagen de qt
      QImage qImage = Mat2QImage(imagenChica);

      // Cambiar la imagen de qt a un mapa de pixeles de qt
      QPixmap pixmap = QPixmap::fromImage(qImage);

      // Limipo el contenido de la etiqueta
      ui->label_5->clear();

      //Se muestra el mapa de pixeles
      ui->label_5->setPixmap(pixmap);


      //Cambiar el espacio de color de la imagenChica

      //De BGR a Escala de Grisis
      //https://hetpro-store.com/TUTORIALES/opencv-cvtcolor/
      Mat imagenGris;
      cvtColor(imagenChica,imagenGris,CV_BGR2GRAY);

      //De BGR a HSV
      Mat imagenHSV;
      cvtColor(imagenChica,imagenHSV,CV_BGR2HSV);

      //De BGR a LAB
      Mat imagenLab;
      cvtColor(imagenChica,imagenLab,CV_BGR2Lab);

      //Filtro del tipo Ventana (PASA BANDA) para la imagen HSV
      Mat imagenColor1; //Matriz en blanco y negro, pixeles blancos, son los
pixeles cuyo color estamos buscando

inRange(imagenHSV,Scalar(Color1C0min,Color1C1min,Color1C2min),Scalar(Color1C0m
ax, Color1C1max, Color1C2max),imagenColor1);
      //Filtro del tipo Ventana (PASA BANDA) para la imagen HSV
      Mat imagenColor2; //Matriz en blanco y negro, pixeles blancos, son los
pixeles cuyo color estamos buscando

inRange(imagenHSV,Scalar(Color2C0min,Color2C1min,Color2C2min),Scalar(Color2C0m
ax, Color2C1max, Color2C2max),imagenColor2);

      //Filtro del tipo Ventana (PASA BANDA) para la imagen HSV
```

```
      Mat imagenColor3; //Matriz en blanco y negro, pixeles blancos, son los
pixeles cuyo color estamos buscando

inRange(imagenHSV,Scalar(Color3C0min,Color3C1min,Color3C2min),Scalar(Color3C0m
ax, Color3C1max, Color3C2max),imagenColor3);

      vector<vector<Point> > contornosColor1;
      vector<vector<Point> > contornosColor2;
      vector<vector<Point> > contornosColor3;
      vector<vector<Point> > contornosprueba;

      vector<Vec4i> jerarquiaColor1;
      vector<Vec4i> jerarquiaColor2;
      vector<Vec4i> jerarquiaColor3;

       vector<Vec4i>jerarquiaColorprueba;
      //Crear una copia de las imagenes binarias
      Mat color1Copia, color2Copia, color3Copia,prueba;

      color1Copia = imagenColor1;
      color2Copia = imagenColor2;
      color3Copia = imagenColor3;
      prueba=imagenColor1;

      findContours(color1Copia, contornosColor1, jerarquiaColor1,
CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, Point(0, 0) );
      findContours(color2Copia, contornosColor2, jerarquiaColor2,
CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, Point(0, 0) );
      findContours(color3Copia, contornosColor3, jerarquiaColor3,
CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, Point(0, 0) );

      findContours(prueba, contornosprueba, jerarquiaColorprueba,
CV_RETR_EXTERNAL,CV_CHAIN_APPROX_NONE);
      for (int i = 0; i < contornosprueba.size(); i++)
       {
       drawContours(imagenChica, contornosprueba, i, Scalar(0,0,255), 2,
8,jerarquiaColorprueba);
       }
       // declare Mat variables, thr, gray and src
       Mat thr, gray;
       gray=imagenColor1;
       threshold( gray, thr, 100,255,THRESH_BINARY );
       // find moments of the image
       Moments m = moments(thr,true);
       Point p(m.m10/m.m00, m.m01/m.m00);
       // coordinates of centroid
       int x=p.x;
       int y=p.y;
       //cout<< Mat(p)<< endl;
       QString coordenadas= QString::number(x)+","+QString::number(y);
       qDebug()<<coordenadas;
       // show the image with a point mark at the centroid
       circle(imagenChica, p, 1, Scalar(0,255,0), -3);
       putText(imagenChica, coordenadas.toUtf8().constData(), Point( p ),
FONT_HERSHEY_DUPLEX, .3,
      Scalar(255,255,255), 1);
       qImage = Mat2QImage(imagenChica);
       // Cambiar la imagen de qt a un mapa de pixeles de qt
       pixmap = QPixmap::fromImage(qImage);
       // Limipo el contenido de la etiqueta
       ui->label_9->clear();
       //Se muestra el mapa de pixeles
       ui->label_9->setPixmap(pixmap);

      //Para vector de contonrnos, se realiza lo siguiente
       for(int i=0; i<(int)contornosColor1.size();i++ ){
        int area1=contourArea(contornosColor1[i]);
        if(area1>areaColor1){
```

```
areaColor1=area1;
}
}
for(int i=0; i<(int)contornosColor2.size();i++ ){
int area2=contourArea(contornosColor2[i]);
if(area2>areaColor2){
areaColor2=area2;
}
}
for(int i=0; i<(int)contornosColor3.size();i++ ){
int area3=contourArea(contornosColor3[i]);
if(area3>areaColor3){
areaColor3=area3;
}
}
if(ui->pushButton_4->isChecked()){
    //Mostrar de acuerdo a los radio button
    if(ui->radioButton_2->isChecked()){
        // Cambiar la imagen de opencv a una imagen de qt
        qImage = Mat2QImage(imagenColor1);

        // Cambiar la imagen de qt a un mapa de pixeles de qt
        pixmap = QPixmap::fromImage(qImage);

        // Limipo el contenido de la etiqueta
        ui->label_6->clear();

        //Se muestra el mapa de pixeles
        ui->label_6->setPixmap(pixmap);
    }
    else if(ui->radioButton_3->isChecked()){
        // Cambiar la imagen de opencv a una imagen de qt
        qImage = Mat2QImage(imagenColor2);

        // Cambiar la imagen de qt a un mapa de pixeles de qt
        pixmap = QPixmap::fromImage(qImage);

        // Limipo el contenido de la etiqueta
        ui->label_6->clear();

        //Se muestra el mapa de pixeles
        ui->label_6->setPixmap(pixmap);
    }
    else if(ui->radioButton_4->isChecked()){
        // Cambiar la imagen de opencv a una imagen de qt
        qImage = Mat2QImage(imagenColor3);

        // Cambiar la imagen de qt a un mapa de pixeles de qt
        pixmap = QPixmap::fromImage(qImage);

        // Limipo el contenido de la etiqueta
        ui->label_6->clear();

        //Se muestra el mapa de pixeles
        ui->label_6->setPixmap(pixmap);
    }
    else{
        // Cambiar la imagen de opencv a una imagen de qt
        qImage = Mat2QImage(imagenColor1);

        // Cambiar la imagen de qt a un mapa de pixeles de qt
        pixmap = QPixmap::fromImage(qImage);

        // Limipo el contenido de la etiqueta
        ui->label_6->clear();

        //Se muestra el mapa de pixeles
        ui->label_6->setPixmap(pixmap);
```

```
            }
        }
        else{
            //Mostrar maquina de estado
            if(contador1 == 0){
                // Cambiar la imagen de opencv a una imagen de qt
                qImage = Mat2QImage(imagenChica);

                // Cambiar la imagen de qt a un mapa de pixeles de qt
                pixmap = QPixmap::fromImage(qImage);

                // Limipo el contenido de la etiqueta
                ui->label_6->clear();

                //Se muestra el mapa de pixeles
                ui->label_6->setPixmap(pixmap);
            }
            else if(contador1 == 1){
                // Cambiar la imagen de opencv a una imagen de qt
                qImage = Mat2QImage(imagenColor1);

                // Cambiar la imagen de qt a un mapa de pixeles de qt
                pixmap = QPixmap::fromImage(qImage);

                // Limipo el contenido de la etiqueta
                ui->label_6->clear();

                //Se muestra el mapa de pixeles
                ui->label_6->setPixmap(pixmap);
            }
            else if(contador1 == 2){
                // Cambiar la imagen de opencv a una imagen de qt
                qImage = Mat2QImage(imagenColor2);

                // Cambiar la imagen de qt a un mapa de pixeles de qt
                pixmap = QPixmap::fromImage(qImage);

                // Limipo el contenido de la etiqueta
                ui->label_6->clear();

                //Se muestra el mapa de pixeles
                ui->label_6->setPixmap(pixmap);
            }
            else if(contador1 == 3){
                // Cambiar la imagen de opencv a una imagen de qt
                qImage = Mat2QImage(imagenColor3);

                // Cambiar la imagen de qt a un mapa de pixeles de qt
                pixmap = QPixmap::fromImage(qImage);

                // Limipo el contenido de la etiqueta
                ui->label_6->clear();

                //Se muestra el mapa de pixeles
                ui->label_6->setPixmap(pixmap);
            }
        }


    }
}
void MainWindow::cronometro2(){
    ui->lcdNumber->display(contador1);
    contador1++;

    //Crear el tope
    if(contador1 > 3){ contador1 = 0;  }
```

```
}

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    QTimer *maquinaEstado = new QTimer(this);
    connect(maquinaEstado,SIGNAL(timeout()),this,SLOT(cronometro2()));
    maquinaEstado->start(1000);



    conectarArduino();


    // Crear el cronometro
    QTimer *temporizador = new QTimer(this);

    // Conectar el temporizador a la funcion cronometro
    connect(temporizador,SIGNAL(timeout()),this,SLOT(cronometro()));

    // Iniciar el temporizador
    temporizador->start(30);


}

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

void MainWindow::on_pushButton_clicked()
{
    //Crear una variable para guardar una imagen
    Mat imagenBoton;
    Mat imagenBotonChica;

    //Catpurar una imagen y guardarla en la variable
    camara >> imagenBoton;

    //Cambiar el tamao de la imagen original
    cv::resize(imagenBoton,imagenBotonChica,Size(249,249),0,0,INTER_CUBIC);

    //Guardar la imagen original en formato jpg
    imwrite("imagen1.jpg",imagenBoton);

    //Mostrar en etiqueta2.

    // Cambiar la imagen de opencv a una imagen de qt
    QImage qImage = Mat2QImage(imagenBotonChica);

    // Cambiar la imagen de qt a un mapa de pixeles de qt
    QPixmap pixmap = QPixmap::fromImage(qImage);

    // Limipo el contenido de la etiqueta
    ui->label_2->clear();

    //Se muestra el mapa de pixeles
    ui->label_2->setPixmap(pixmap);

}

void MainWindow::on_pushButton_2_clicked()
{
```

```
    int numeroPin = ui->spinBox->value();
    int estadoPin = 0;
    if(ui->radioButton->isChecked()){
        estadoPin = 1;
    }
    else{
        estadoPin = 0;
    }
    QString paqueteDatos =
"{\"pin\":\""+QString::number(numeroPin)+"\",\"estado\":\""+QString::number(es
tadoPin)+"\"}";

    ui->label_4->setText(paqueteDatos);

    if(arduino_esta_conectado){
        if(arduino->isWritable()){
            arduino->write(paqueteDatos.toUtf8().constData());

        }
    }
}

void MainWindow::on_horizontalScrollBar_sliderMoved(int position)
{
    //C0min = position;
    if(ui->radioButton_2->isChecked()){
        //Actualizo color1
        Color1C0min = position;
    }
    else if(ui->radioButton_3->isChecked()){
        //Actualizo color2
        Color2C0min = position;
    }
    else if(ui->radioButton_4->isChecked()){
        //Actualizo color3
        Color3C0min = position;
    }
    else{
        //Actualizo color1
        Color1C0min = position;
    }

}

void MainWindow::on_horizontalScrollBar_2_sliderMoved(int position)
{
    //C0min = position;
    if(ui->radioButton_2->isChecked()){
        //Actualizo color1
        Color1C0max = position;
    }
    else if(ui->radioButton_3->isChecked()){
        //Actualizo color2
        Color2C0max = position;
    }
    else if(ui->radioButton_4->isChecked()){
        //Actualizo color3
        Color3C0max = position;
    }
    else{
        //Actualizo color1
        Color1C0max = position;
    }
}

void MainWindow::on_horizontalScrollBar_6_sliderMoved(int position)
{
    //C0min = position;
```

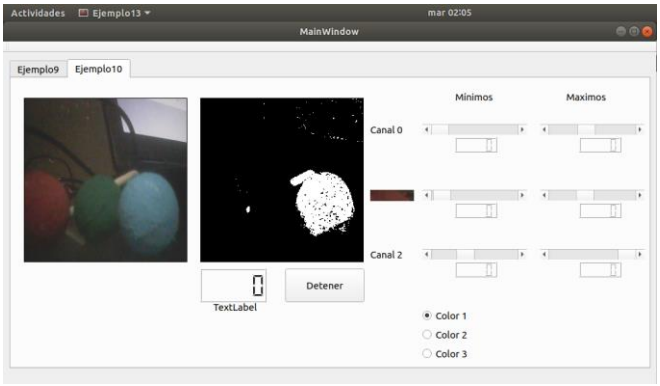
```
    if(ui->radioButton_2->isChecked()){
        //Actualizo color1
        Color1C1min = position;
    }
    else if(ui->radioButton_3->isChecked()){
        //Actualizo color2
        Color2C1min = position;
    }
    else if(ui->radioButton_4->isChecked()){
        //Actualizo color3
        Color3C1min = position;
    }
    else{
        //Actualizo color1
        Color1C1min = position;
    }
}

void MainWindow::on_horizontalScrollBar_5_sliderMoved(int position)
{
    //C0min = position;
    if(ui->radioButton_2->isChecked()){
        //Actualizo color1
        Color1C1max = position;
    }
    else if(ui->radioButton_3->isChecked()){
        //Actualizo color2
        Color2C1max = position;
    }
    else if(ui->radioButton_4->isChecked()){
        //Actualizo color3
        Color3C1max = position;
    }
    else{
        //Actualizo color1
        Color1C1max = position;
    }
}

void MainWindow::on_horizontalScrollBar_8_sliderMoved(int position)
{
    if(ui->radioButton_2->isChecked()){
        //Actualizo color1
        Color1C2min = position;
    }
    else if(ui->radioButton_3->isChecked()){
        //Actualizo color2
        Color2C2min = position;
    }
    else if(ui->radioButton_4->isChecked()){
        //Actualizo color3
        Color3C2min = position;
    }
    else{
        //Actualizo color1
        Color1C2min = position;
    }
}

void MainWindow::on_horizontalScrollBar_7_sliderMoved(int position)
{
    //C0min = position;
    if(ui->radioButton_2->isChecked()){
        //Actualizo color1
        Color1C2max = position;
    }
    else if(ui->radioButton_3->isChecked()){
        //Actualizo color2
```

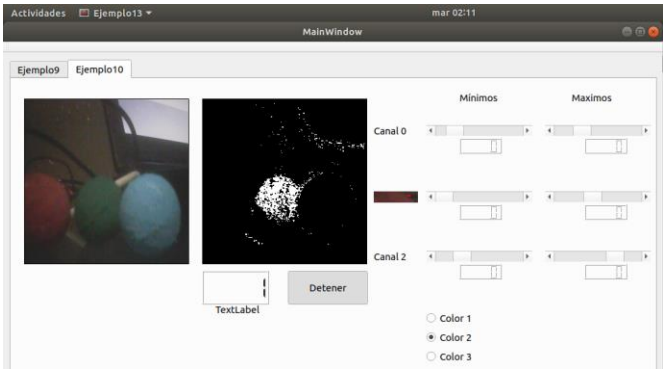
```
        Color2C2max = position;
    }
    else if(ui->radioButton_4->isChecked()){
        //Actualizo color3
        Color3C2max = position;
    }
    else{
        //Actualizo color1
        Color1C2max = position;
    }
}
```

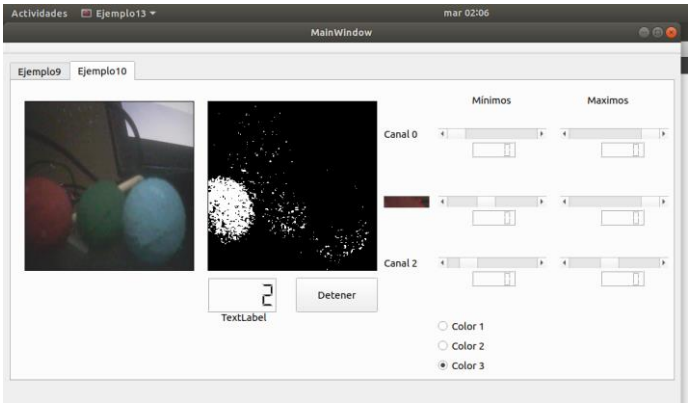
Código mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <QtSerialPort/QSerialPort>
namespace Ui {
class MainWindow;
}
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
public slots:
    void cronometro();
    void cronometro2();
private slots:
    void on_pushButton_clicked();
    void recepcionSerialAsyncrona();
    void conectarArduino();
    void on_pushButton_2_clicked();
    void on_horizontalScrollBar_sliderMoved(int position);
    void on_horizontalScrollBar_2_sliderMoved(int position);
    void on_horizontalScrollBar_6_sliderMoved(int position);
    void on_horizontalScrollBar_5_sliderMoved(int position);

    void on_horizontalScrollBar_8_sliderMoved(int position);

    void on_horizontalScrollBar_7_sliderMoved(int position);


private:
    Ui::MainWindow *ui;
    QSerialPort *arduino;
    bool arduino_esta_conectado = false;
    int contador1 = 0;
    int Color1C0min = 0, Color1C0max = 0;
    int Color1C1min = 0, Color1C1max = 0;
    int Color1C2min = 0, Color1C2max = 0;
    int areaColor1=0;
    int areaColor2=0;
    int areaColor3=0;
    int Color2C0min = 0, Color2C0max = 0;
    int Color2C1min = 0, Color2C1max = 0;
    int Color2C2min = 0, Color2C2max = 0;

    int Color3C0min = 0, Color3C0max = 0;
    int Color3C1min = 0, Color3C1max = 0;
    int Color3C2min = 0, Color3C2max = 0;
};

#endif // MAINWINDOW_H
```

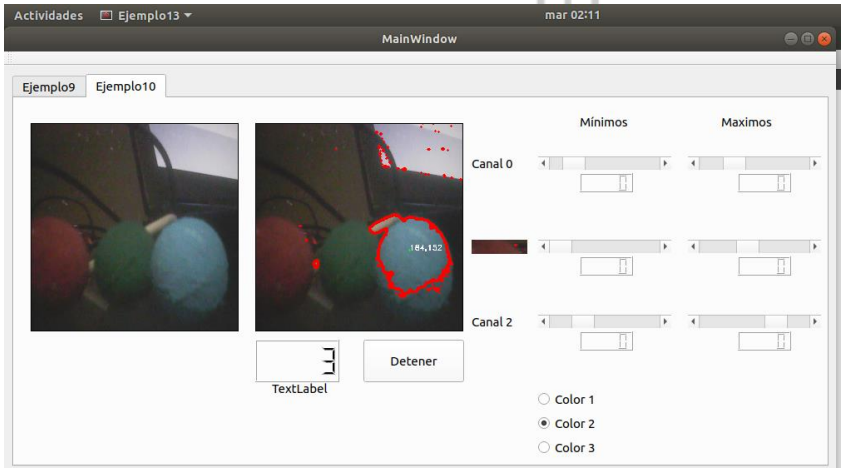
Tarea 4







Área más grande encontrada con las coordenadas en su centro



Aguilar Rodríguez Carlos Adolfo 215860049