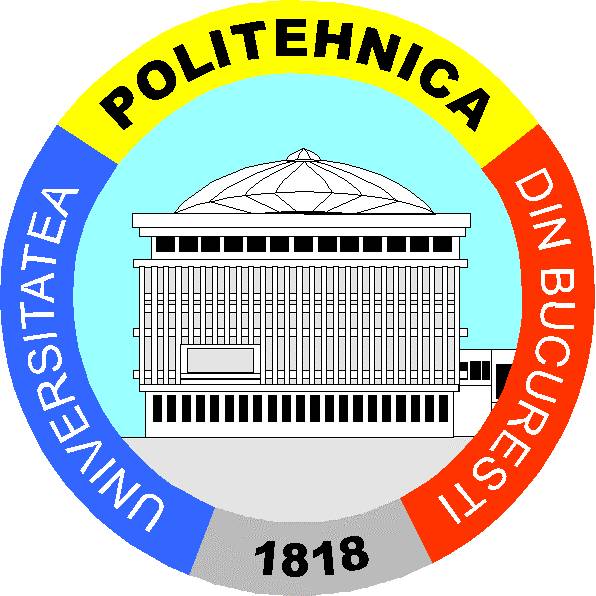
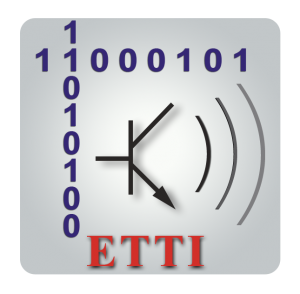
Universitatea Politehnica din Bucuresti

Facultatea de Electronica, Telecomunicatii si Tehnologia Informatiei

Anul 2018-2019

Conducător științific

Andreea Griparis

Manda Mihai

Preda Alin

Urucu Marius

Vodă Mihai

Grupa: 443A

Decodor cod de bare și QR

Contents

[1.Tema si motivatie: 3](#_Toc534462086)

[2.Tehnologii si biblioteci folosite: 4](#_Toc534462087)

[2.1 Python 4](#_Toc534462088)

[2.2 OpenCV 4](#_Toc534462089)

[2.3 QT 5](#_Toc534462090)

[2.4 ZBar(pyzbar) 5](#_Toc534462091)

[3.Conceptul de cod de bare/QR: 6](#_Toc534462092)

[3.1 Codul de bare 6](#_Toc534462093)

[3.2 Codul QR 6](#_Toc534462094)

[4.Descrierea aplicatiei: 7](#_Toc534462095)

[4.1 Prezentare interfata grafica: 7](#_Toc534462096)

[4.2 ImageScan: 8](#_Toc534462097)

[4.3 WebScan: 9](#_Toc534462098)

[4.4 ShowDoc: 10](#_Toc534462099)

[5. Code sample: 11](#_Toc534462100)

[5.1 Main.py: 11](#_Toc534462101)

[5.2 barcode\_scanner\_image.py: 12](#_Toc534462102)

[5.3 barcode\_scanner\_video.py(optional): 13](#_Toc534462103)

[6. Pasi instalare si setup: 13](#_Toc534462104)

[7. Bibliografie: 14](#_Toc534462105)

# 1.Tema si motivatie:

Obiectivul proiectului este sa implementam un scanner de coduri de bare si QR.Acesta este realizat cu ajutorul urmatoarelor tehnologii si biblioteci: Python versiunea 3.XX , OpenCV, QT si ZBar.

Motivatia acestui proiect este nevoie imensa de a usura metodele curente de scanare si plata a produselor pe care le cumparam de zi cu zi. De cate ori nu am stat la cozi imense? Cate ore am pierdut in ultimii ani,in loc sa facem ceva productive? Raspunsul este de multe ori si multe ore pierdute inutil.De aici ne-a venit ideea ca am putea realiza o aplicatie simpla care sa citeasca in prima faza codurile de bare si cele de tip QR

[[1]](#footnote-1)

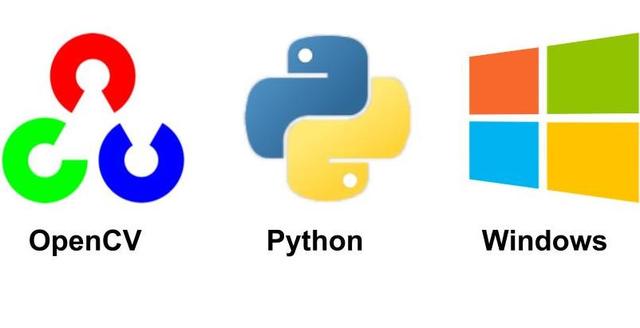
# 2.Tehnologii si biblioteci folosite:

## 2.1 Python

* Python este un limbaj de programare foarte popular care oferă posibilitatea programării structurate dar și orientate pe obiecte.
* Este un limbaj de scripting, ceea ce înseamnă că este interpretat și nu complilat, economisind mult timp în procesul de dezvoltare și debugging .
* API-ul Python oferă module pentru o gamă foarte mare de funcționalități, de la cele de bază pentru lucrurl cu șiruri și fișiere, până la cele pentru lucrul cu procese, threaduri, etc.
* Există o varietate de API-uri precum NumPy, SciPy, Matplotlib.

## 2.2 OpenCV

* OpenCV *“Open Source Computer Vision Library”* este o bibliotecă de funcții open source, scrisă în C, și C++.
* Biblioteca a fost proiectată astfel încât să ofere eficiență computațională pentru aplicații de computer vision în timp real.
* Aceasta conține peste 500 de funcții care acoperă domenii precum machine learning, imagistica medicală, securitate, robotică.
* Scopul bibliotecii este de a pune la dispoziție o infrastructură de procesare a imaginilor, ușor de folosit ce poate fi utilizată în dezvoltarea rapidă a unor aplicații complexe.

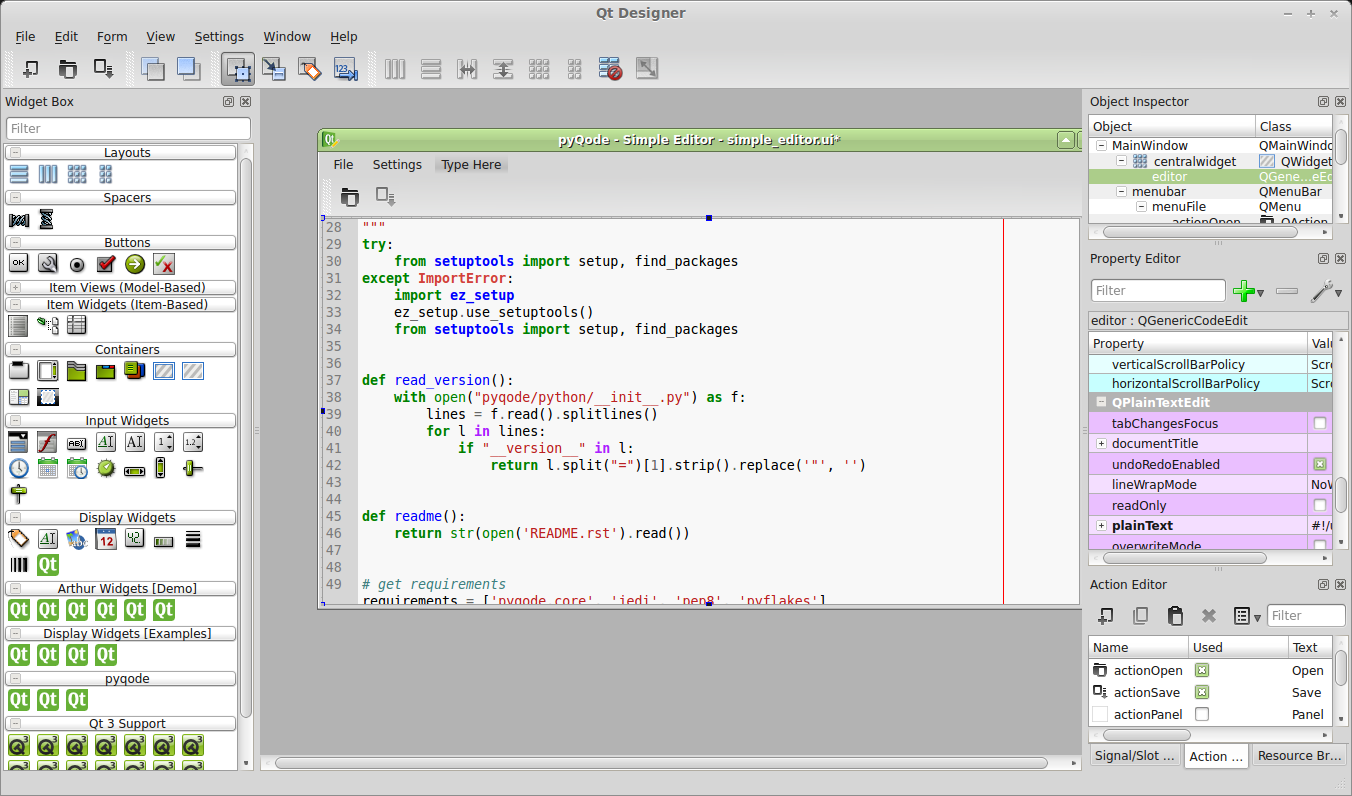
[[2]](#footnote-2)

## 2.3 QT

* Qt facilitează dezvoltarea de aplicații C++ si interfete utilizator independente de platformă. Sunt suportate o largă varietate de platforme hardware și software, de tip desktop, mobile şi embedded.
* Utilizează sistemul grafic al sistemului de operare pe care rulează; astfel, aplicaţiile au aspectul şi comportamentul nativ al sistemului gazdă.
* Qt poate interactiona cu alte limbaje, cum ar fi Java (Qt Jambi), precum si cu limbaje de scripting, cum ar fi QML, JavaScript sau Python.
* Cu aplicația Qt Designer se pot construi cu ușurință interfețe grafice și transpuse în cod Python.

## 2.4 ZBar(pyzbar)

* Cea mai bună bibliotecă pentru detectarea și decodarea codurilor de bare și a codurilor QR de diferite tipuri.
* zbar-py(pyzbar) este un modul (compatibil atât cu Python 2.7 și 3+), care oferă o interfață pentru biblioteca de decodare a codurilor de bare zbar, care poate citi majoritatea formatelor de coduri de bare, precum și codurile QR.
* imaginile de intrare trebuie să fie matrice 2D de tip uint8 (adică imagini 2D în nuanțe de gri).
* Biblioteca zbar în sine a fost împachetată împreună cu zbar-py (este construită ca o extensie python), deci nu sunt necesare dependențe externe.

[[3]](#footnote-3)

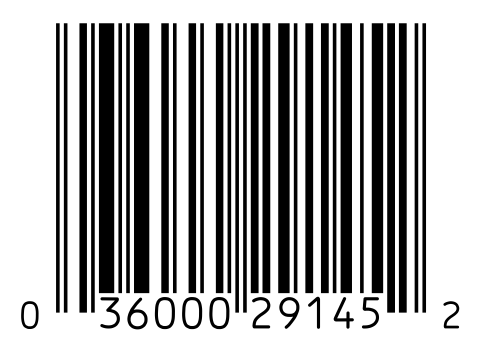
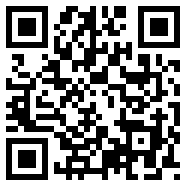
# 3.Conceptul de cod de bare/QR:

## 3.1 Codul de bare

* Codul de bare este o reprezentare de date codificată (cifrată), destinată a fi citită pe cale optică. Codurile de bare sunt folosite în multe domenii, îndeosebi industriale.
* Un cod de date are aspectul unui șir de bare negre de diverse grosimi pe un fundal alb. În general fiecare cifră sau literă se reprezintă printr-o anumită combinație de 1 sau mai multe bare.
* Setul de simboluri utilizate este în general restrâns (cuprinde de obicei literele, cifrele și câteva semne speciale).
* Toate codurile de bare dispun de un caracter special de start/stop ce permite citirea atât de la stânga la dreapta, cât și de la dreapta la stânga. Prin convenție caracterul din stânga este considerat caracterul de start și caracterul din dreapta caracterul de stop.

## 3.2 Codul QR

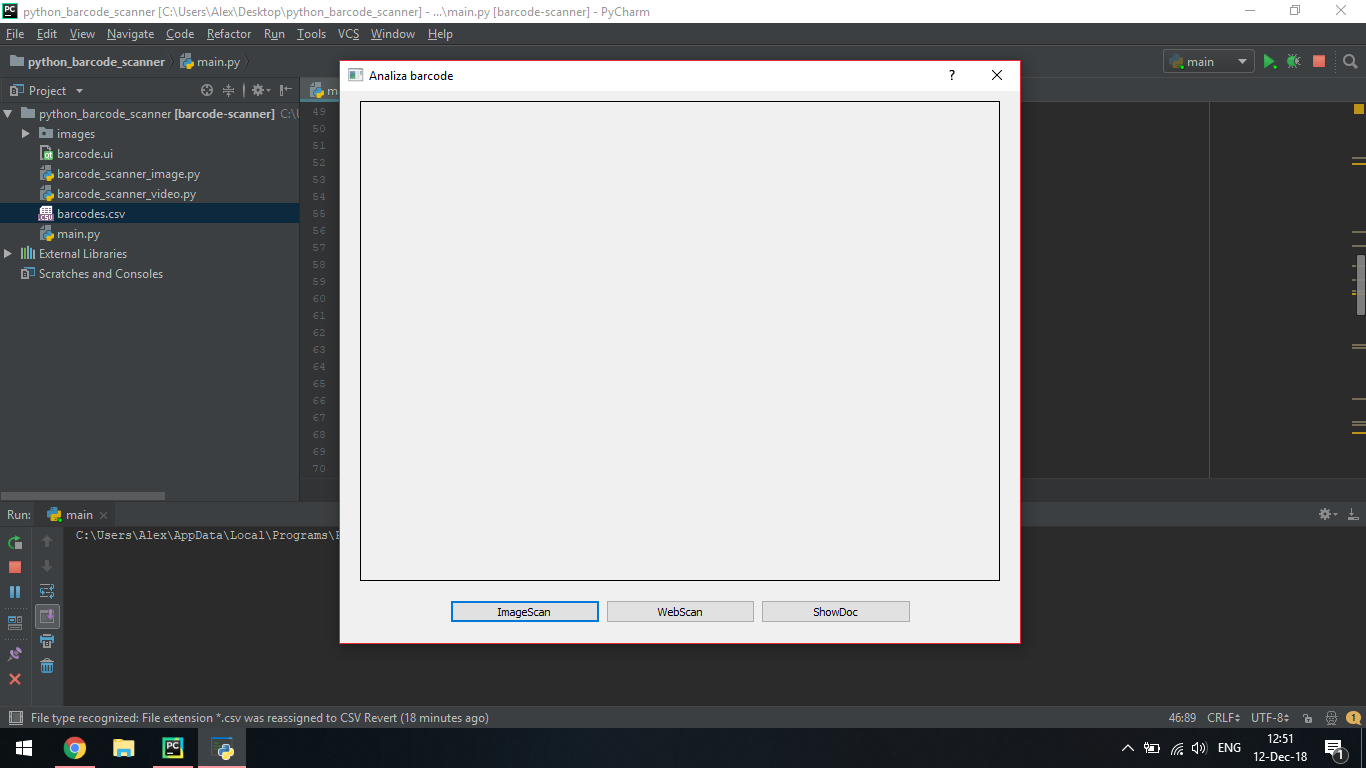
* Codul QR este o gamă de standarde de codare cu formă de bare bidimensionale (cod matrice). QR este o prescurtare din engleză de la quick response („răspuns rapid”).
* Sunt utilizate cu regularitate pe telefoanele mobile de tip smartphone, deoarece codurile QR pot reprezenta (în mod codificat) adrese de site-uri Internet (de tip URL); pentru acces rapid la un site dorit utilizatorul trebuie doar să îi scaneze codul QR cu ajutorul camerei foto din telefonul său mobil. Un software cititor precum cel realizat de noi interpretează imaginea și descifrează codul.

[[4]](#footnote-4) [[5]](#footnote-5)

# 4.Descrierea aplicatiei:

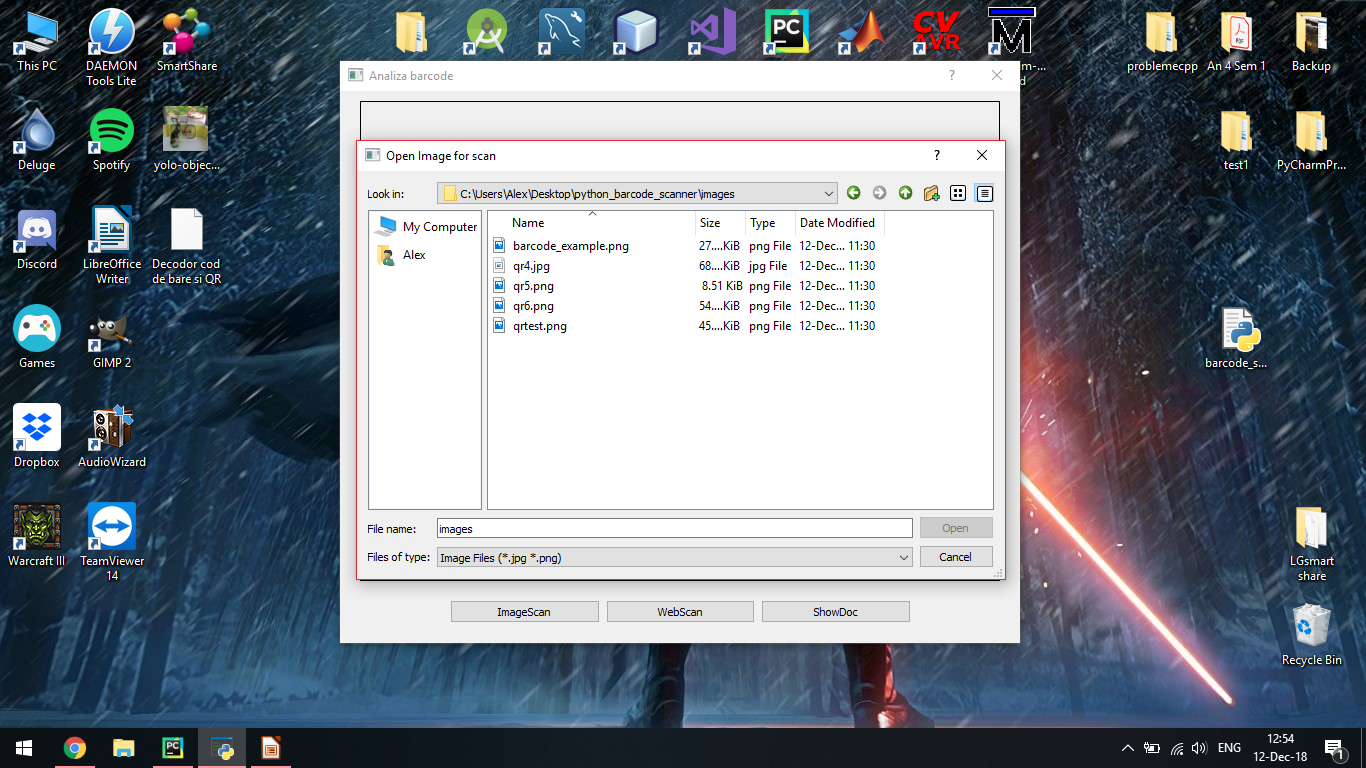
* Folosindu-ne de modulele OpenCV se va capta și procesa imaginile și streamul video.
* Odată captate se va folosi biblioteca Python de decodare Zbar(pyzbar).
* Biblioteca Zbar va decoda codul de bare sau QR, iar cu OpenCV se vor face prelucrări suplimentare pentru a afișa rezultatul.
* Totul va fi integrat într-un GUI realizat cu framework-ul Qt.

## 4.1 Prezentare interfata grafica:

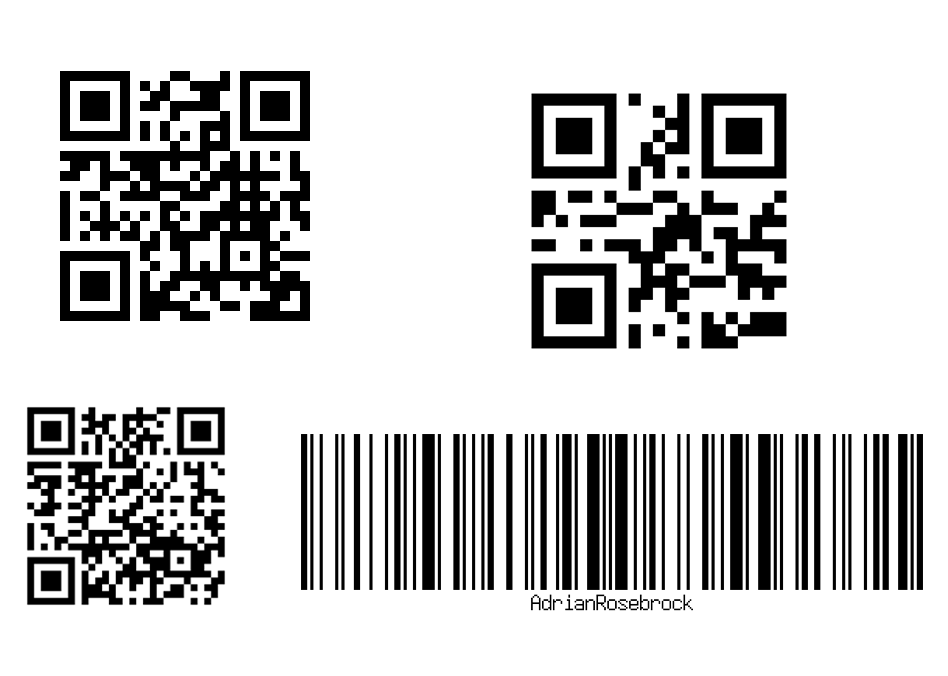
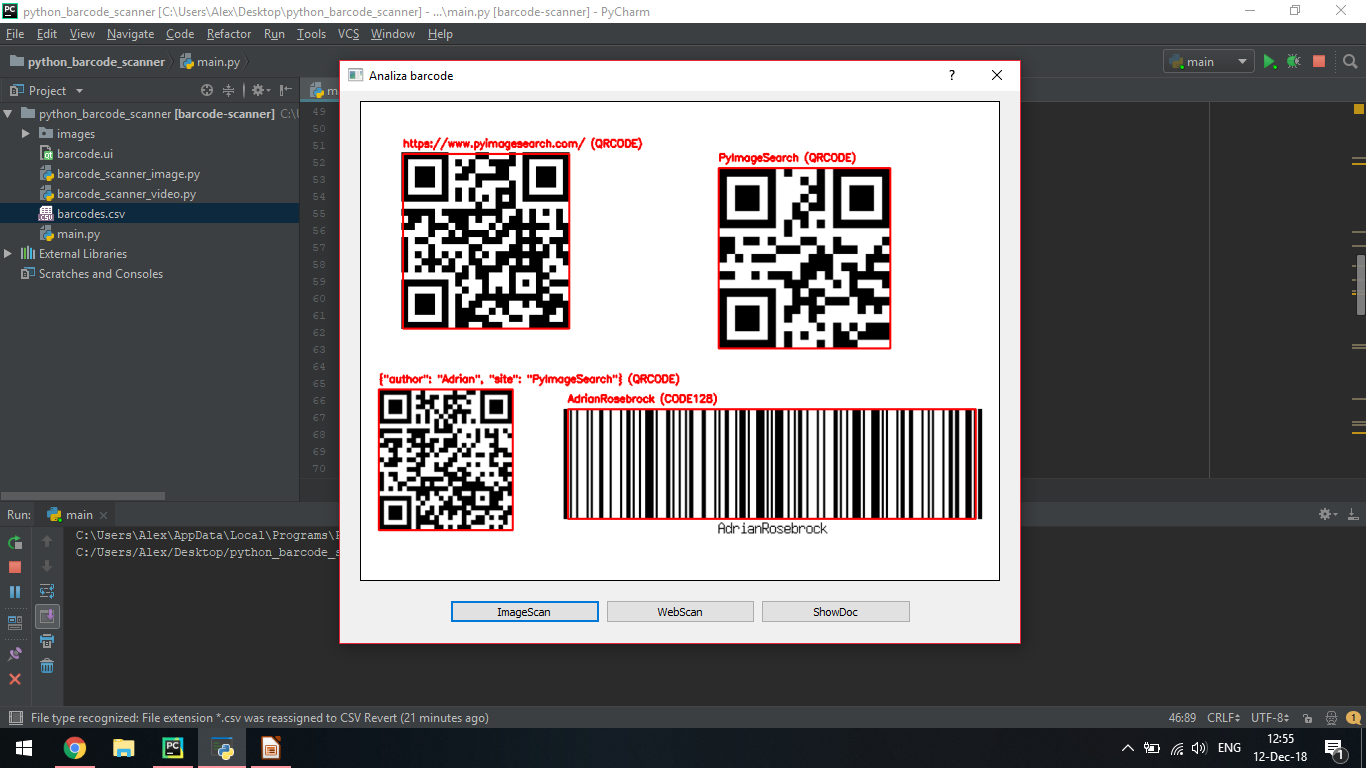
**[[6]](#footnote-6)**

Interfata a fost realizata cu ajutorul framework-ului QT,folosind unealta sa,QT Designer,apoi a fost transpus in cod Python.Interfata contine o zona pentru afisarea imaginii analizate sau a frame-urilor unui video/captura webcam si 3 butoane. Prin apasarea butonului ImageScan se va deschide un overlayer care te va lasa sa selectezi si uploadezi imagina care va urma sa fie scanata si prelucrata. Prin apasarea “WebScan” se deschide un flux video de la webcam si se vor analiza frame-urile acestui flux. Butonul “ShowDoc” arata log-ul ultimelor detectii web(daca s-a detectat un cod se va scrie in fisier ,daca nu fisierul ramane gol).

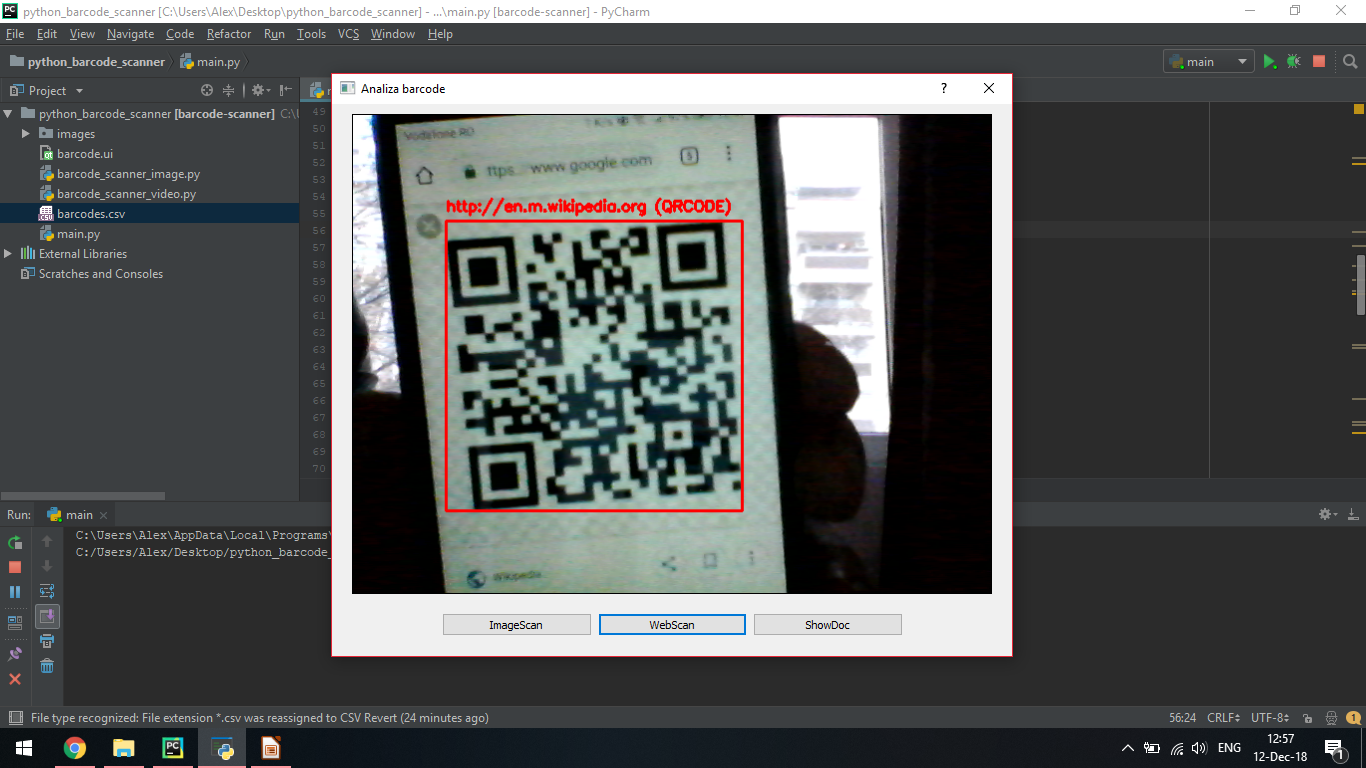
## 4.2 ImageScan:

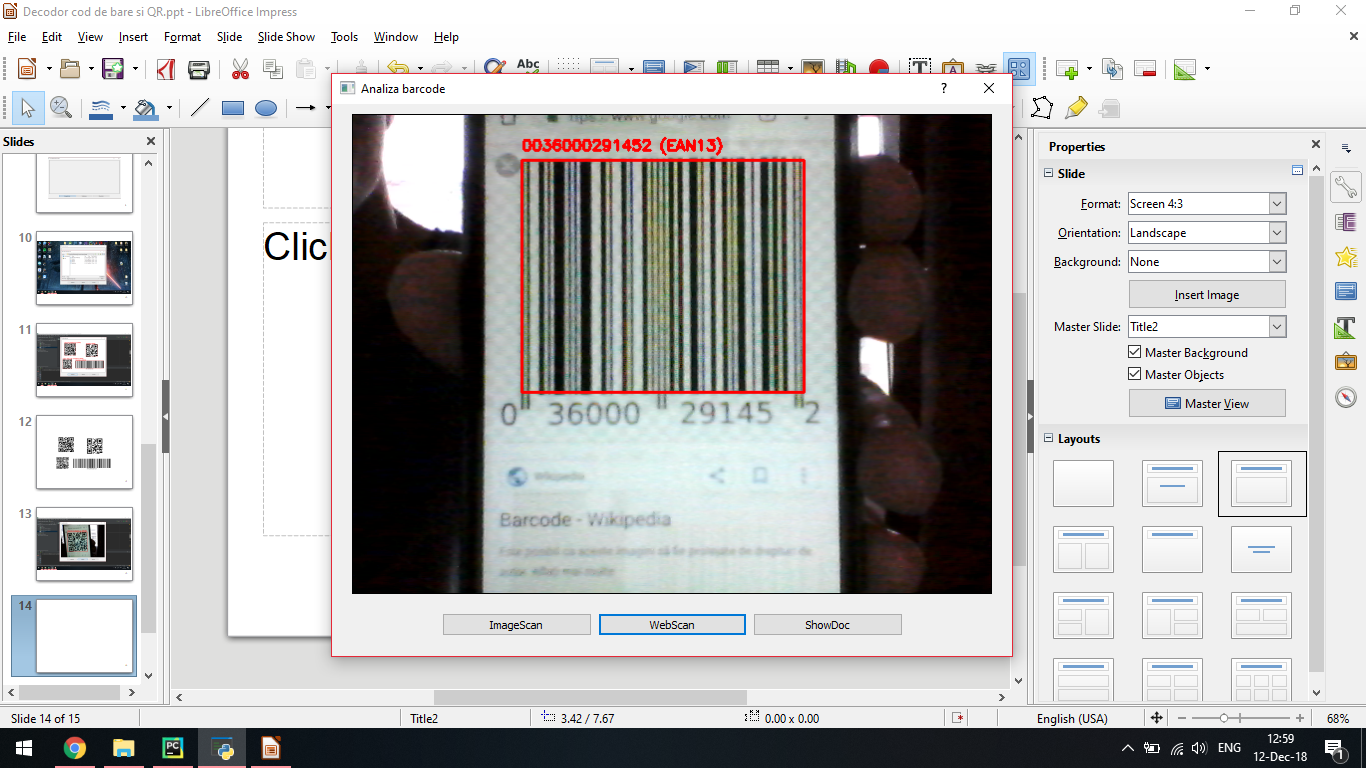
****

Imaginile prelucrate corect se vor incadra intr-un chenar rosu si deasupra se va scrie rezultatul prelucrarii.

****

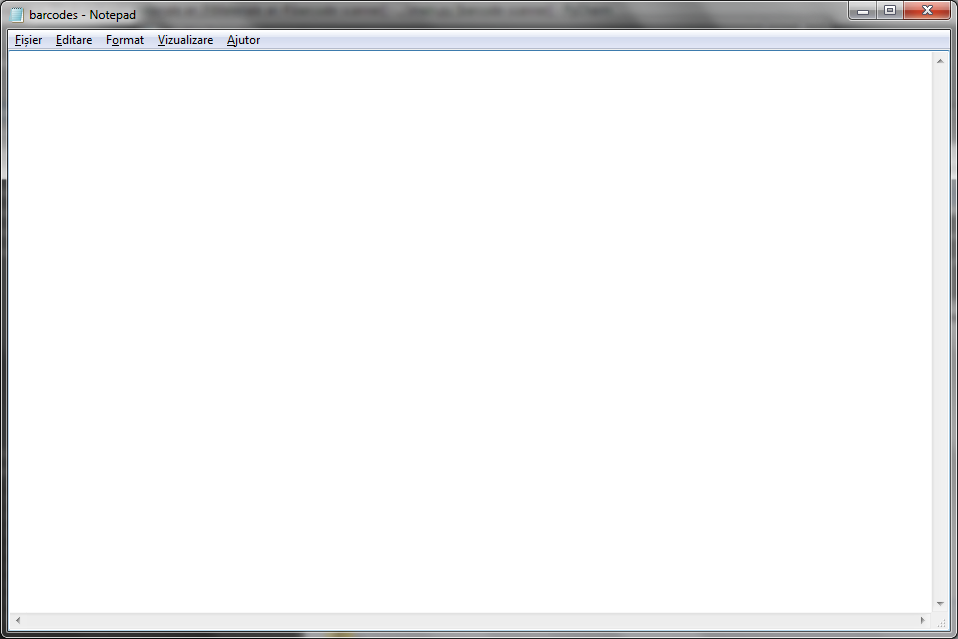
## 4.3 WebScan:

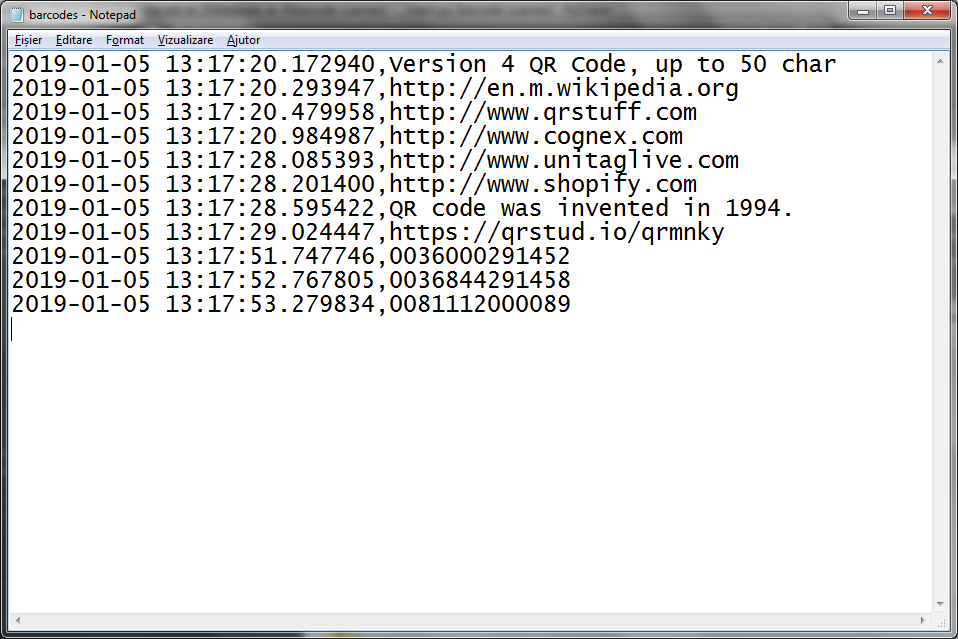
****

****

Acelasi eveniment este realizat si pentru detectia live prin webcam,singura diferenta este ca in detectia live se fac prelucrari constante pana la oprirea fluxului video.

## 4.4 ShowDoc:

****

****

Observatie: La fiecare rulare a programului,documentul „barcodes” de va reseta,deci se sterge tot ce a fost scris in el la detectia anterioara.

# 5. Code sample:

## 5.1 Main.py:

class barcode(QDialog):  
 def \_\_init\_\_(self):  
 super(barcode, self).\_\_init\_\_()  
 loadUi('barcode.ui', self)  
  
 # construct the argument parser and parse the arguments  
 ap = argparse.ArgumentParser()  
 ap.add\_argument("-o", "--output", type=str, default="barcodes.csv",  
 help="path to output CSV file containing barcodes")  
 args = vars(ap.parse\_args())  
 self.csv = open(args["output"], "w")  
 self.found = set()  
  
 self.image = None  
 self.image\_frame = None  
 self.timerChecker = False  
  
 self.ImageScan.clicked.connect(self.image\_scan)  
 self.WebScan.clicked.connect(self.web\_scan)  
 self.ShowDoc.clicked.connect(self.show\_doc)

if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 window = barcode()  
 window.setWindowTitle('Analiza barcode')  
 window.show()  
 sys.exit(app.exec\_())

## 5.2 barcode\_scanner\_image.py:

# load the input image  
image = cv2.imread(args["image"])  
  
# find the barcodes in the image and decode each of the barcodes  
barcodes = pyzbar.decode(image)

apelam pyzbar.decode pentru a detecta si decoda codurile din imagine

# loop over the detected barcodes  
for barcode in barcodes:  
 # extract the bounding box location of the barcode and draw the  
 # bounding box surrounding the barcode on the image  
 (x, y, w, h) = barcode.rect  
 cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)  
  
 # the barcode data is a bytes object so if we want to draw it on  
 # our output image we need to convert it to a string first  
 barcodeData = barcode.data.decode("utf-8")  
 barcodeType = barcode.type  
  
 # draw the barcode data and barcode type on the image  
 text = "{} ({})".format(barcodeData, barcodeType)  
 cv2.putText(image, text, (x, y - 10), cv2.FONT\_HERSHEY\_SIMPLEX,  
 0.5, (0, 0, 255), 2)  
  
  
return image

In aceasta bucla, procedam astfel:

* Extragem caseta de incadrare (x, y) - coordonate de la obiectul barcode.rect (Linia 29), care ne permite sa localizam si sa determinam unde in imaginea de intrare este codul de bare curent.
* Desenam un dreptunghi de incadrare pe imaginea din jurul codului de bare detectat (linia 30).
* Decodam codul de bare într-un sir de tip "utf-8" si extragem tipul de cod de bare (linii 34 și 34). Este critic pentru a apela functia .decode ("utf-8") pe obiect pentru a converti dintr-un byte array într-un string.
* Formam și desenam barcodeData si barcodeType pe imagine (Linii 38-40).

## 5.3 barcode\_scanner\_video.py(optional):

vs = cv2.VideoCapture(0)  
  
# open the output CSV file for writing and initialize the set of  
# barcodes found thus far  
csv = open(args["output"], "w")  
found = set()  
  
# loop over the frames from the video stream  
while True:  
 if not vs.isOpened():  
 print('Unable to load camera.')  
 time.sleep(5)  
 pass  
 # grab the frame from the threaded video stream and resize it to  
 # have a maximum width of 400 pixels  
 ret, frame = vs.read()  
  
 # find the barcodes in the frame and decode each of the barcodes  
 barcodes = pyzbar.decode(frame)

# 6. Pasi instalare si setup:

Cerinte preliminare si instalare Windows:

Pentru a rula „samplet-ul” nostru trebuiesc instalate urmatoarele:

* Python v3.6.4(sau orice versiune 3.X.X) de pe site-ul oficial:

<https://www.python.org/downloads/release/python-364/> . Instalati din executabil.

* OpenCV v3.4.1(sau mai mare) de la: <https://opencv.org/opencv-3-4-1.html>

Se downloadeaza si ruleaza executabilul.Dupa instalare pentru a il face compatibil cu python 3.X.X se instaleaza suplimentar cateva fisiere „Wheel” : <https://www.youtube.com/watch?v=izN-NLpS5t8>. Sau se da cu pip install si ar trebui sa se instaleze tot ce e necesar.

* Se instaleaza pyzbar prin comanda : pip install pyzbar
* Se instaleaza framework-ul QT prin : pip install pyqt5
* Pentru Qt designer se foloseste si : pip install pyqt5-tools
* Instalati orice ide compatibil cu python doriti(in acest proiect s-a folosit PyCharm)

Instalare Ubuntu:

* Python : <https://docs.python-guide.org/starting/install3/linux/>
* OpenCV : <https://docs.opencv.org/3.4.1/d2/de6/tutorial_py_setup_in_ubuntu.html>
* Se instaleaza pyzbar prin comanda : pip install pyzbar
* Se instaleaza framework-ul QT prin : pip install pyqt5
* Pentru Qt designer se foloseste si : pip install pyqt5-tools
* Instalati orice ide compatibil cu python doriti(in acest proiect s-a folosit PyCharm)

# 7. Bibliografie:

* [**https://www.pyimagesearch.com/2018/05/21/an-opencv-barcode-and-qr-code-scanner-with-zbar/?fbclid=IwAR1JaXG7oDVWCt6a5qRmDp2ZxbqOWz8nS4nfiVQhHyAasFdFk0q5EZPYa0k**](https://www.pyimagesearch.com/2018/05/21/an-opencv-barcode-and-qr-code-scanner-with-zbar/?fbclid=IwAR1JaXG7oDVWCt6a5qRmDp2ZxbqOWz8nS4nfiVQhHyAasFdFk0q5EZPYa0k)
* [**https://opencv.org/**](https://opencv.org/)
* [**http://doc.qt.io/qt-5/qtdesigner-manual.html**](http://doc.qt.io/qt-5/qtdesigner-manual.html)
* [**https://en.wikipedia.org/wiki/ZBar**](https://en.wikipedia.org/wiki/ZBar)
* [**https://ro.wikipedia.org/wiki/Cod\_QR**](https://ro.wikipedia.org/wiki/Cod_QR)
* [**https://en.wikipedia.org/wiki/Barcode**](https://en.wikipedia.org/wiki/Barcode)
* [**https://en.wikipedia.org/wiki/Python\_(programming\_language**](https://en.wikipedia.org/wiki/Python_(programming_language)**)**

1. https://storage0.dms.mpinteractiv.ro/media/401/341/5846/15127716/1/coada.jpg?width=620 [↑](#footnote-ref-1)
2. https://www.learnopencv.com/wp-content/uploads/2017/11/opencv-python-windows.jpg [↑](#footnote-ref-2)
3. Fig 1 Ex QT Designer [↑](#footnote-ref-3)
4. https://ro.wikipedia.org/wiki/Cod\_de\_bare [↑](#footnote-ref-4)
5. https://ro.wikipedia.org/wiki/Cod\_QR [↑](#footnote-ref-5)
6. Fig 2: Interfata la pornire [↑](#footnote-ref-6)