УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

ПРОЈЕКАТ

из предмета пројектовање електронских система

Препознавање и алкотестирање човека, уз публиковање података на *Cloud* сервер

Студенти:

Јован Славујевић
Стефан Швендерман

Ментор:

проф. др Рајс Владимир

Нови Сад, 2020. год.

Садржај

1 Увод

Тема пројекта јесте креирање апликације која врши детекцију људи и препознавање лица од интереса, њихово алкотестирање, а потом публиковање обрађених података на *Cloud* сервер. Као ресурсе апликација на располагању има *Web* камеру рачунара и *MQ-2* сензор (сензор за детекцију гаса у ваздуху, у овом случају молекула алкохола) који је повезан на *RaspberryPi* и податке доставља путем његових *GPIO* пинова.

Сваки од наведених сегмената самостално су веома корисни и увелико имају практичну примену, а њихова комбинација представља интересантан склоп за *Internet of Things* пројекат.

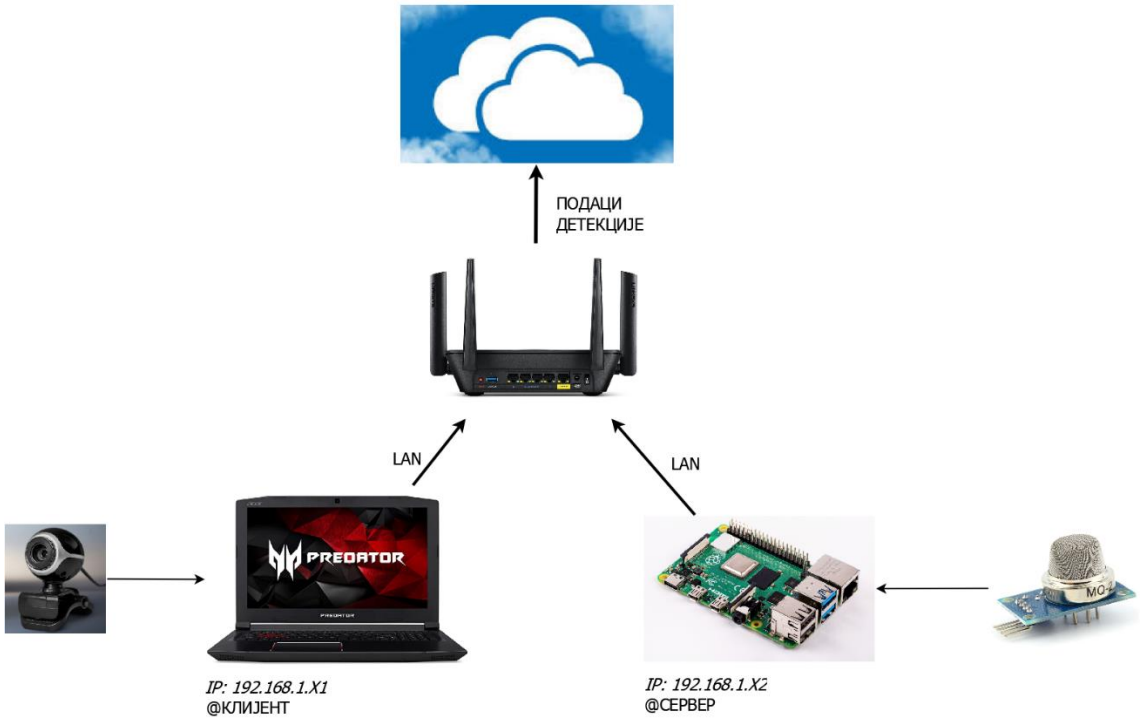
Детекција лица човека, а затим и његово препознавање спадају под област вештачке интелигенције (енг. *Artificial Intelligence*).

2 Блок шема целог система

Овај систем се састоји из две апликације. Први део у склопу овог система одвија се на персоналном рачунару и своди се на апликацију која препознаје присуство човека на текућем фрејму уз помоћ стандардног *OpenCV* класификатора људског лика, затим ексклузивно у случају детекције људског лика следећи део апликације врши рекогницију и утврђује да ли се ради о познатом лицу из базе података коришћењем истренираног неуронске мреже (*Tensorflow* тренинг модела).

Уколико је детектовано људско присуство на камери следи преузимање података са сензора алкохола. Други део у склопу овог система односи се на мерење присуства алкохола у ваздуху код особе која стоји испред алкотеста. Овај сегмент представља засебну апликацију која у међурелацији ове две апликације игра улогу сервера који шаље клијенту (рачунарском делу апликације) податке са сензора. Сензор поседује како аналогни тако и дигитални пин, а у овом пројекту је сензор повезан на *RaspberryPi* преко дигиталног пина који враћа логичку нулу за присуство алкохола, у супротном шаље јединицу. За случај да на фрејму није детектован човек, закључује се да нема потребе за алкотестирањем.

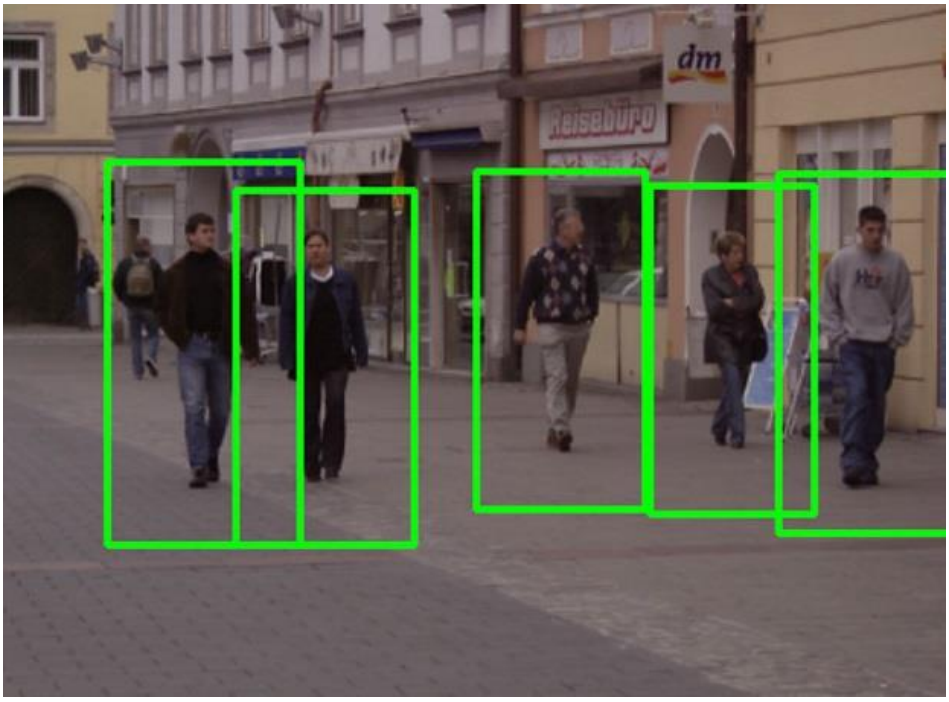
Следећи сегмент се такође одвија на рачунарском делу овог система и представља публиковање прикупљених података детекције на *Cloud* сервер. Овај процес извршава се у периодично на сваких 5 секунди. У питању је *IoT Cloud* сервер намењен за рад студената са Катедре за електронику. Сервер је направљен и уступљен од стране компаније *Wolkabout*.



Слика 1. Блок шема целокупног система

3 Платформа за детекцију људског лика – *OpenCV*

OpenCV представља колекцију библиотека писаних у *C/C++*, *Python* и *Java* програмском језику које се првенствено користе за компјутерску визију у реалном времену. Првобитно ју је развила компанија *Intel*, а данас је доступна као софтвер отвореног кода. Неке од примена ових библиотека укључују: сегментацију, праћење покрета, препознавање лица, препознавање гестова, померање камера унутар *3D* простора (енг. *Egomotion*), али и препознавање објеката помоћу тренираних неуронских мрежа.



Слика 2. Пример детекције објеката помоћу OpenCV платформе

У оквиру овог пројекта *OpenCV* платформа задужена је поред детекције људског лика на текућем фрејму и за повезивање са драјвером *Web* камере, за прикупљање фрејмова и њихово паковање у бинарне матрице, управљање брзином преноса фрејмова у секунди (енг. *Frames per Second – fps*), као и за коначну визуализацију обрађеног фрејма.

4 Платформа за рекогницију људи од интереса – *Tensorflow*

Tensorflow је библиотека отвореног кода отворена за *Python* за нумеричка израчунавања која чини аутоматско учење бржим и лакшим. Машинско учење (енг. *Machine Learning*) је сложена дисциплина. Међутим, примена модела машинског учења је далеко лакша и мање комплексна него што је то била захваљујући оквирима за учење машина као што је *Google Tensorflow*. Ово олакшава процес аквизиције података, обучавања модела, предвиђања и унапређивања будућих резултата.

Креиран је од стране *Google Brain* тима, представља библиотеку која повезује мноштво аутоматских и дубоких учења (енг. *Deep Learning*), као и алгоритама, те их чини корисним путем заједничке сврхе. Користи се *Python* како би обезбедио погодан *front-end API* (енг. *Application Programming Interface* – Апликативни програм за корисничку употребу) за изградњу апликација са оквиром, док изводи те апликације у *C++* са високим перформансама.

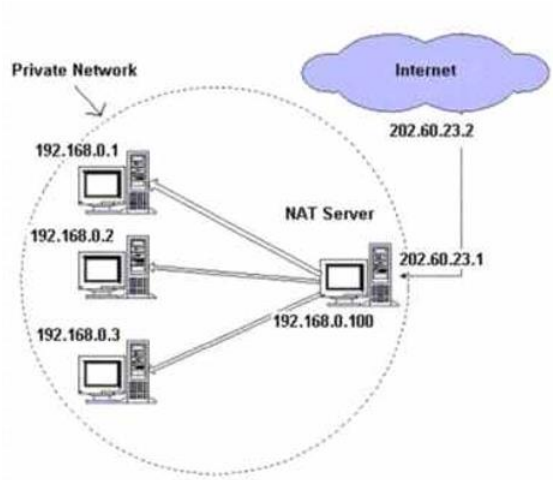
Слика 3. Лого Tensorflow платформе

У овом пројекту *Tensorflow* је коришћен за тренирање неуронске мреже, као и за њену примену у оквиру апликације – препознавање лица из базе података тренираног модела на текућем фрејму.

5 Комуникација између уређаја

Као што је већ елаборирано у склопу овог пројекта, две засебне хардверске целине чине рачунар са једне и *RaspberryPi* са друге стране. На рачунару је коришћена *Web* камера, а оквиру *RaspberryPi*-а је коришћен сензор *MQ-2*, који детектује присуство алкохола. Да би се искористила оба сензора неопходно је обезбедити покретање апликације у оквиру обе машине. Рачунарска апликација требује податке са сензора, што значи да у овом случају рачунар игра улогу клијента, а *RaspberryPi* игра улогу сервера.

Комуникација између ова два уређаја базирана је на повезивању преко *LAN* конекције, односно оба уређаја морају бити повезана на исту мрежу, те се распознају помоћу додељених *IP* (енг. *Internet Protocol*) адреса. Клијентска апликација се покреће тек након успешног покретања серверске апликације, јер у оквиру њене иницијализације потпада и конекција са сервером. У моменту када клијент (рачунар) жели да покупи податке са сервера (податке са *MQ-2* сензора), он заправо шаље податак у виду једног бајта и то *ASCII* (енг. *American Standard Code for Information Interchange*) нуле, а заузврат дигитални излаз сензора му враћа да ли је детектовано присуство алкохола ('1') или није ('0').



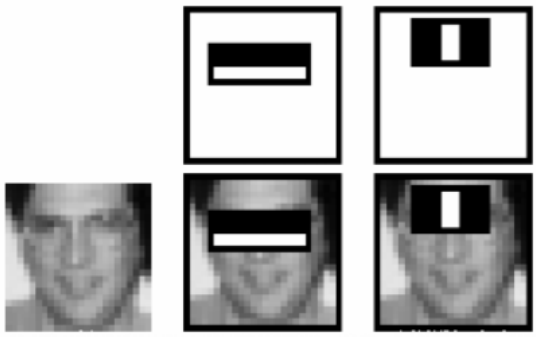
Слика 4. Принцип функционисања приватне мреже

6 Имплементација алгоритма за детекцију и препознавање

У склопу овог задатка било је потребно урадити детекцију човековог лика, а након тога препознавање лица од интереса (админа) уз помоћ истрениране неуронске мреже.

Први део задатка рађен је уз помоћ *OpenCV* платформе. Најпре је било неопходно преузети слику са *Web* камере, што *OpenCV* чини уз помоћ класе *cv2.VideoCapture* чијим инстанцирањем је као параметар могуће проследити путању видеа, или у овом случају број који представља редни број камере којима рачунар тренутно располаже, па је тако у ситуацији где су на рачунар повезане две камере приступање једној ће бити преко цифре 0, а другој преко 1. Даље, податке које ова инстанца преузима у реалном времену потребно је проследити бинарној матрици хексадецималних вредности за боје пиксела слике (фрејма) чији је репрезент у случају програмског језика *Python* такође *OpenCV* класа *cv2.Mat*. На такав начин текућим фрејмом се управља у виду локалне променљиве коју је после тога могуће проследити на даље процесуирање.

Процесуирање у оквиру *OpenCV* платформе заправо представља детекцију уз помоћ већ постојећег *OpenCV* стандардног класификатора за детекцију људског лица. Овај класификатор функционише по принципу да сваки објекат на слици који има контуре и обрисе човека – два ока и нос, односно конкретно се тражи контраст између тамнијих и светлијих карактеристика лица.



Слика 5. Начин функционисања OpenCV класификатора за детекцију лица

Класификатор представља фајл са *XML* (енг. *Extensible Markup Language*) екстензијом, а њега је могуће укључити (енг. *Import*) у апликацију помоћу инстанце *cv2.CascadeClassifier* којој се као аргумент прослеђује стринг који представља апсолутну путању на којој се класификатор налази. Процес детекције у *Python* програмском језику врши се преко функције *cv2.DetectMultiscale* којој је неопходно проследити класификатор и улазну слику. Функција враћа податке о локацији детектованог објекта на слици, а то су координате почетне тачке, као и ширина и дужина правоугаоника који се црта на објекту. Уколико није детектован ниједан објекат за којим класификатор трага, функција неће вратити ништа (заправо враћа *None* који представља пандан *Null* пољу из програмског језика *C/C++*). Пошто *Python* спада под програме који немају процес билдовања већ се компајлирање и линковање извршавају у оквиру покретања апликације, неопходно је

предвидети могуће случајеве и заштити апликацију од тзв. „пуцања“. Уколико функција враћа координате, закључује се да је детектован човек на фрејму у супротном се сматра да човек није детектован, а самим тим не може се извршити процес рекогниције и алкотестирања те се аутоматски сматра да на фрејму није препознат админ (лице од интереса) и да, логично, човек није у алкохолисаном стању. Добијене координате детекције складиште се за евентуалну визуализацију.

```
def human_detection(image, cascade = cv2.CascadeClassifier):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faces = cascade.detectMultiScale(gray, 1.1, 4)
    visualization = image.copy()

    if(faces is None):
        return False

    human = Human()
    human.image = visualization
    if(size > 0):
        human.detected = True
        coordinate_set = faces[0]
        human.coordinates = coordinate_set
    return human
```

Слика 6. Приказ импелементације детекције човека на фрејму

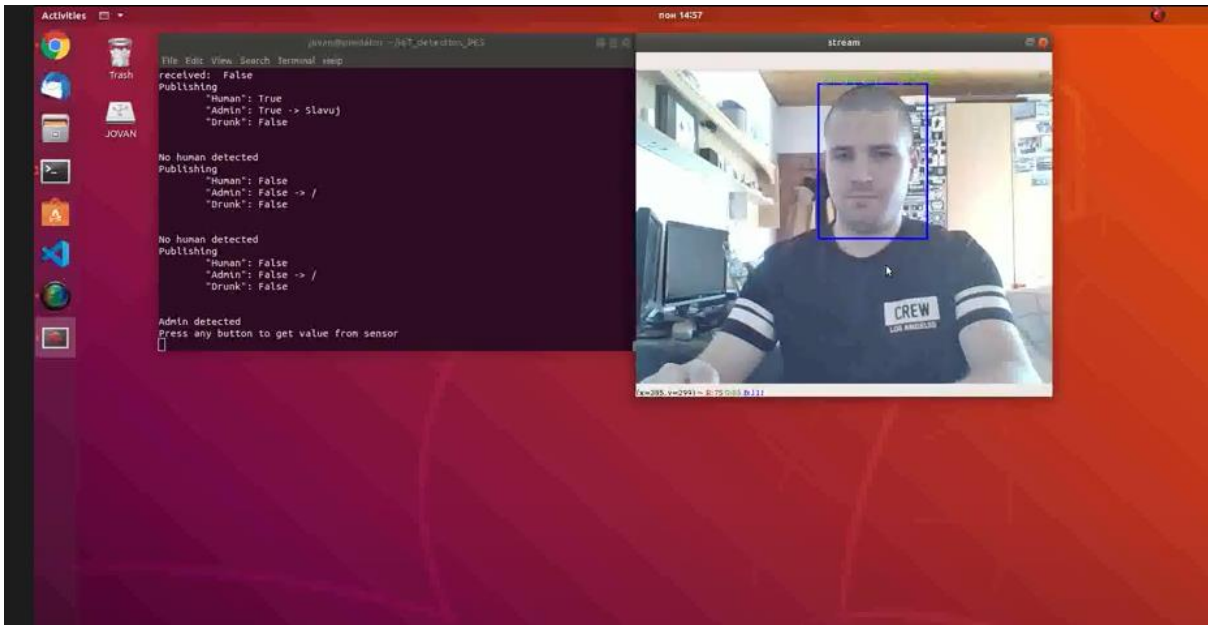
Процес рекогниције одвија се уз помоћ *Tensorflow* програмске платформе. Да би препознавање лица од интереса било могуће, неопходно је обезбедити тренинг модел који је научен да препознаје лица за која се корисник определио. У случају овог пројекта улогу админа су имали аутори пројекта. Да би тренинг поступак био могућ, неопходно је обезбедити довољан сет слика за лик оба аутора, те уз помоћ *LabelImg.exe* апликације маркирати поље на којем се тражени лик налази. Апликација потом генерише *XML* фајл засебно за сваку слику. Након тога, неопходна ставка у тренинг процесу представља поседовање *nVidia* графичке картице у машини на којој се тренирање одвија, као и инсталација *CUDA* (енг. *Compute Unified Device Architecture*) и *CUDNN* (енг. *CUDA Deep Neural Network*) драјвера за графичку картицу. Тренинг процес одиграва се помоћу *Tensorflow Models* окружења (енг. *Framework)* које је могуће преузети преко званичног *Tensorflow GitHub* репозиторијума. Уколико је све спремно, тренинг процес може да почне. Тренинг траје око 12 сати. Трајање тренинга умногоме зависи од количине слика које треба да се обраде, као и перформанси машине на којој се извршава.

Коначан корак препознавања лица од интереса могућ је након линковања и иницијализације *Tensorflow* библиотека у оквиру апликације. Тренинг модел се као и *OpenCV* каскадни класификатор увезује у апликацију прослеђивањем његове апсолутне путање инстанци класе *Detection Graph* из *Tensorflow Obejct Detection API.* Још једна

сличност састоји се у томе да је коначни резултат детекције сет координата правоугаоника на којем се препознато лице налази, али и његов *ID*, што је у овом случају име админа (додељени идентификатори у пројекту су Славуј и Швендерман), а сам процес детекције (рекогниције) се врши уз помоћ *tf.session.run* функције. Добијене координате детекције складиште се за евентуалну визуализацију.

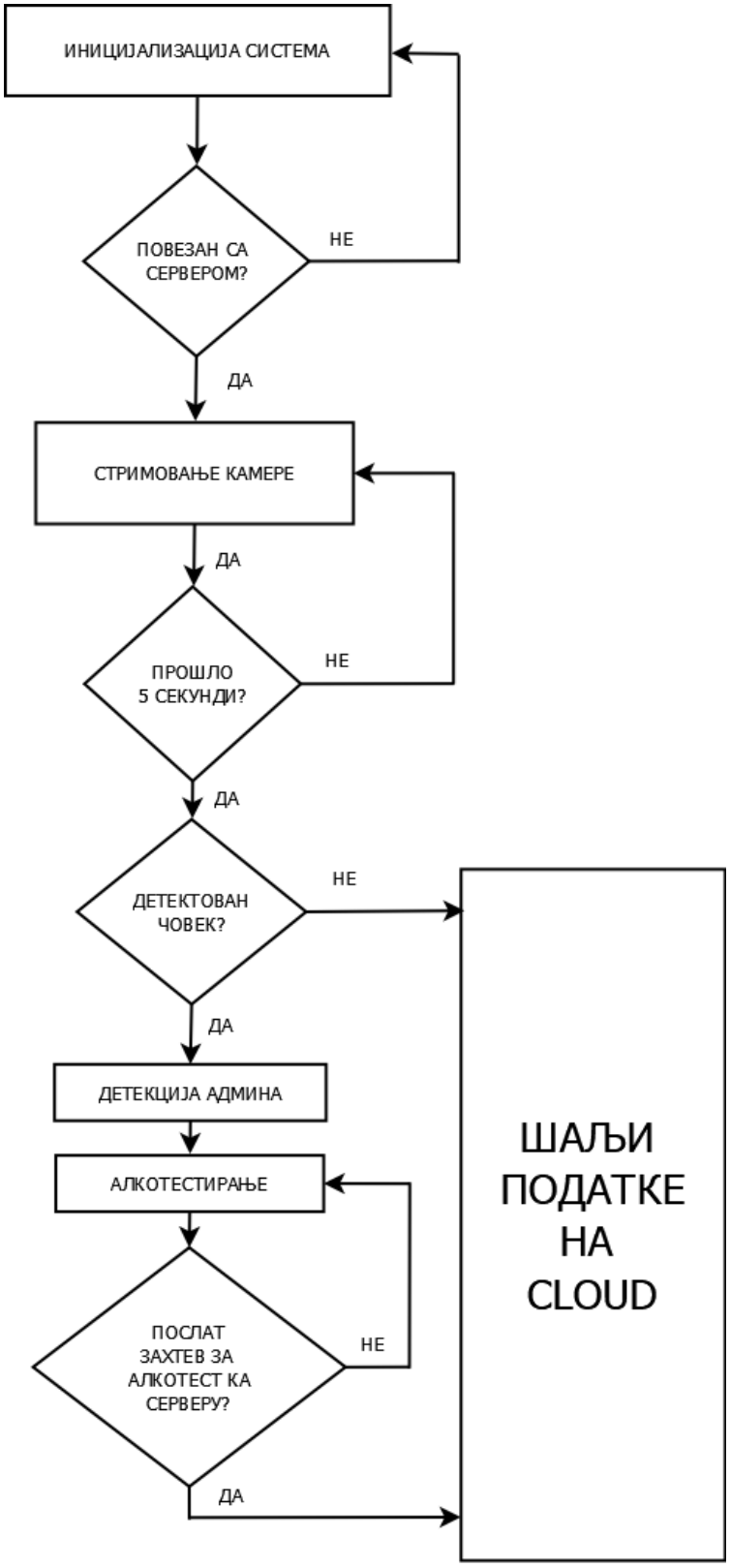
```
def admin_detection(image, det_model = Detection_Model):
    image_expanded = np.expand_dims(image, axis = 0)

    (boxes, scores, classes, num) = det_model.session.run(
        [det_model.detection_boxes, det_model.detection_scores, det_model.detection_classes, det_model.num_detections],
        feed_dict ={det_model.image_tensor: image_expanded})
```

Слика 7. Приказ дела имплементације рекогнције лица од интереса



Слика 8. Препознавање админа уз помоћ платформе Tensorflow

Мењање фрејмова, односно стримовање камере, могуће је помоћу *cv2.waitKey* функције којој се прослеђује број као параметар који апроксимира брзину промене фрејмова (*fps*). Уколико се проследи јединица, фрејмови ће се прослеђивати у реалном времену.



Слика 9. Алгоритам рада рачунарског дела апликације

7 Публиковање података на *Cloud* сервер

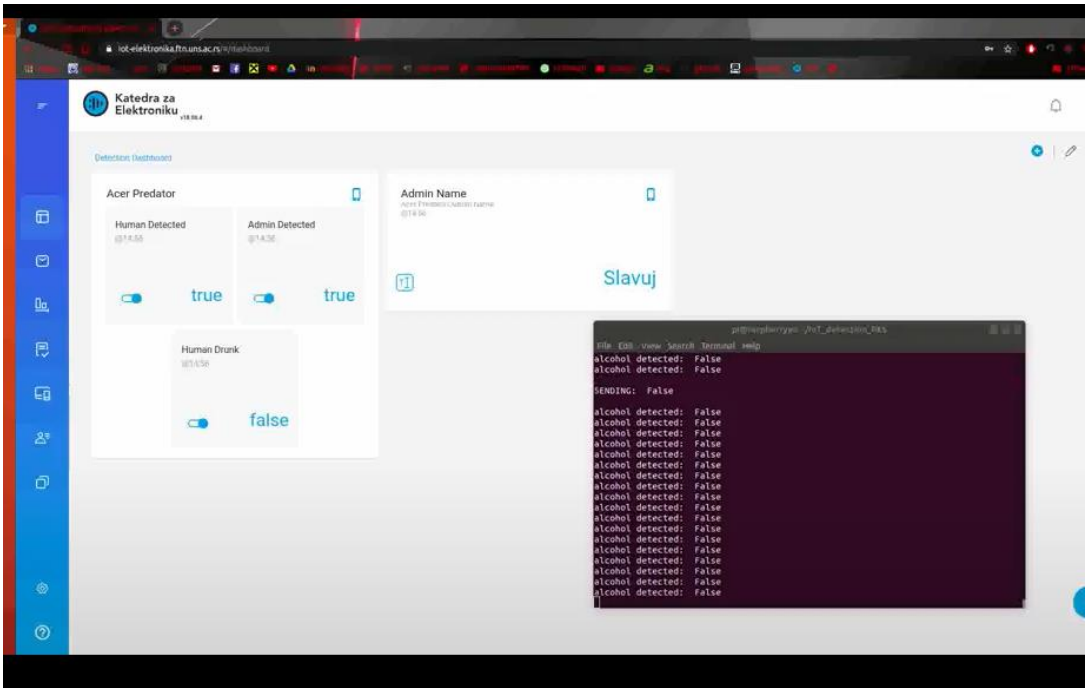
Као што је у склопу објашњења алгоритма рада и наведено, након прикупљања и обраде слика (фрејмова), те извршене детекције лика и преузетих података са алкотеста, неопходно је похранити резултат на *IoT Cloud* платформу.

Да би публиковање било могуће неопходно је инсталирати *Wolkabout*-ове *Python* библиотеке које омогућавају публиковање на тај сервис. Поред тога на *Web* страници потребно је креирати уређај, као и *Dashboard* табелу на којој ће се ажурирати примљени подаци. Табела може по потреби да има различите начине приказа, као што су графикони. У случају овог пројекта коришћен је обичан *status bar* јер су подаци прилично тривијални и своде се на 3 *bool* и 1 *string* поруку. Креирањем уређаја генерише се идентификациони кључ (*ID* и лозинка) који се прослеђује програму приликом иницијализације *Wolkabout* дела.

Апликација на сваких 5 секунди врши пресек стања те шаље резултујуће податке на сервер. Пре самог публиковања чекају се подаци са *RaspberryPi* стране. Време утрошено на чекање клијента да поднесе захтев од сервера (да притисне било које дугме на тастатури) није урачунато у периоду од 5 секунди.

```
def to_Cloud(info1=bool, info2=bool, info3=str, info4=bool, device=wolk.WolkConnect):
    device.add_sensor_reading("Human", info1)
    device.add_sensor_reading("Admin", info2)
    device.add_sensor_reading("Which Admin", info3)
    device.add_sensor_reading("Drunk", info4)
    device.publish()
    print('Publishing \n\t"Human": ' + str(info1) + '\n\t"Admin": ' + str(info2) + ' -> ' + info3 + '\n\t"Drunk": ' + str(info4) + '\n')
```

Слика 10. Приказ имплементације слања података на Cloud



Слика 11. Приказ публикованих података на Cloud сервер

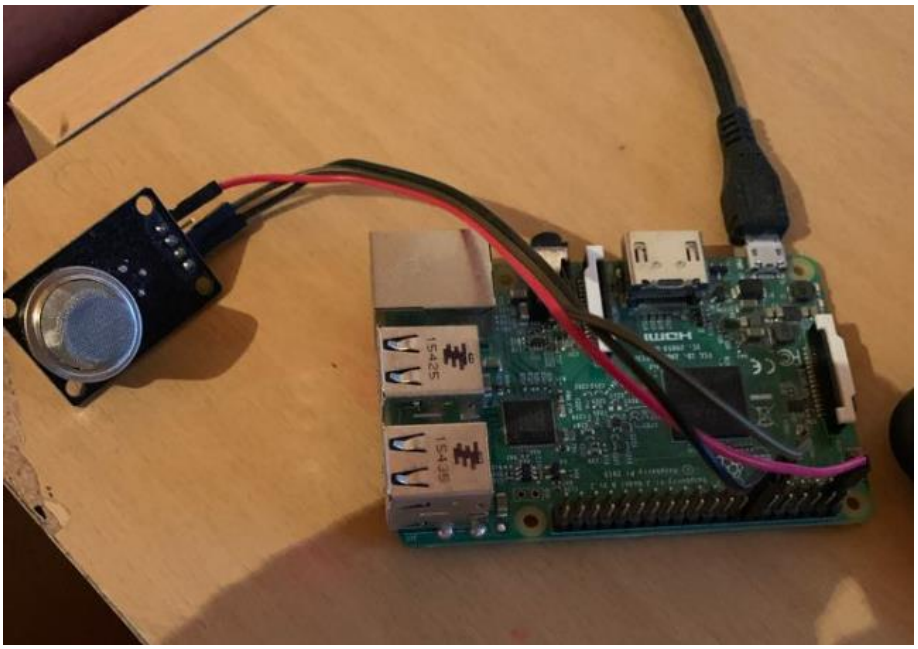
8 Имплементација алгоритма за алкотестирање

Како би се проширила функционалност пројекта, у исти је додат и алкотест. Алкотестирање могуће је захваљујући *MQ-2* сензору, који ради на принципу пироелектричног ефекта, мерећи промену у проводности калај-диоксида (SnO_2) у присуству запаљивих гасова, у шта спадају и алкохолна испарења. *MQ-2* сензор поред пинова за напајање садржи и два излазна пина – дигитални и аналогни. Аналогни пин обезбеђује прецизнију информацију о концентрацији алкохола, док дигитални представља бинарни излаз. Уколико се у ваздуху налази довољна концентрација алкохола на дигиталном излазу враћа се логичка нула, у супротном јединица. За потребе овог пројекта довољно је било коришћење дигиталног излаза сензора.

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(17,GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
```

Слика 12. Приказ иницијализације сензора

Сензор се повезује на *RaspberryPi*. Дигитални излазни пин може се повезати на било који од слободних *GPIO* пинова овог мини рачунара. Напајање сензора је 5 *V*.

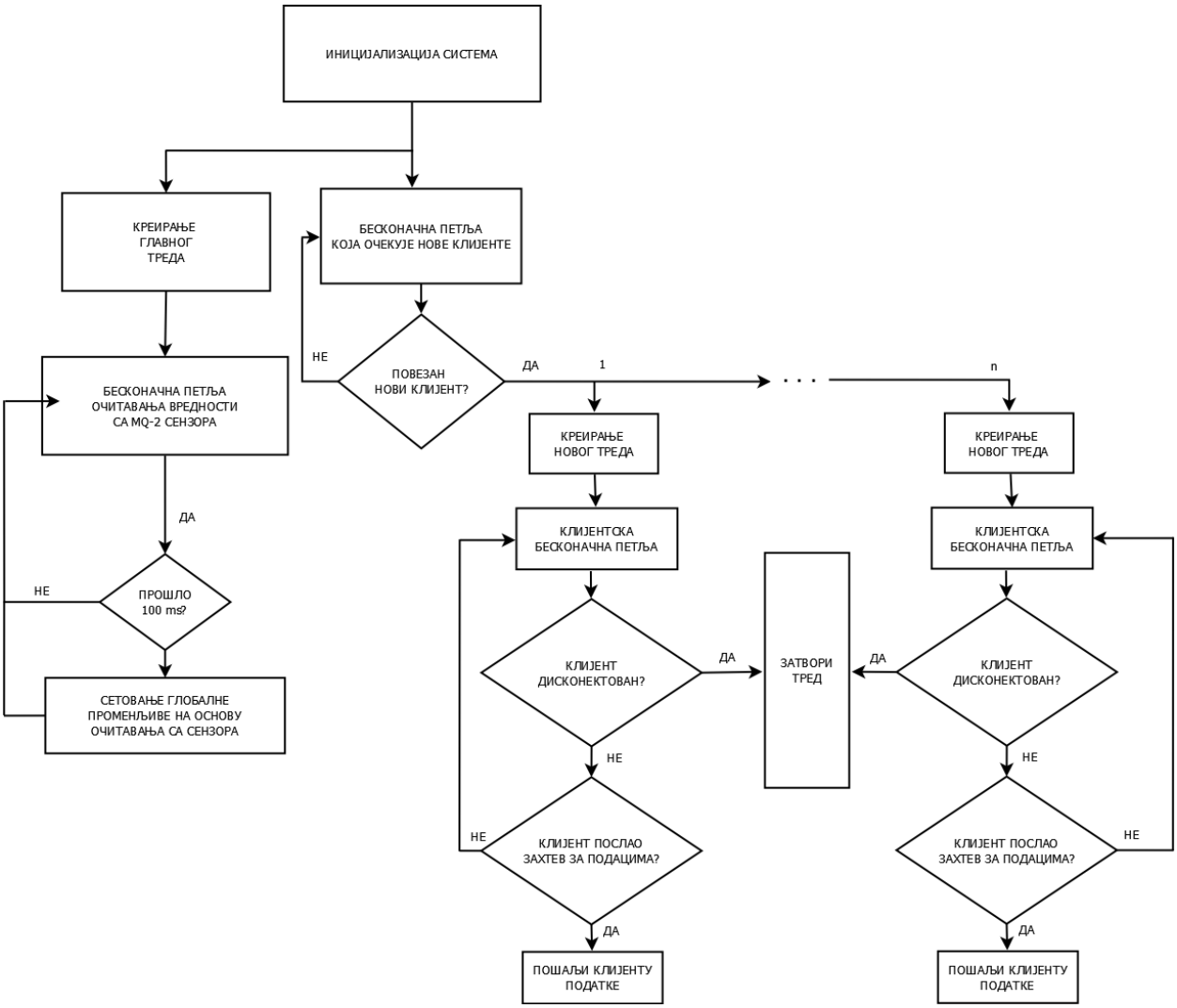


Слика 13. Приказ физичког повезивања сензора

Серверска апликација (апликација која је покренута на *RaspberryPi*) функционише по *Multithreading* принципу, где улогу главног треда игра бесконачна петља у оквиру које се на сваких 100 *ms* преузимају подаци са дигиталног излаза сензора и исписују на излазу конзоле. Сваким новим повезивањем клијента креира се нови тред који реагује на догађај (енг. *Event*) у виду захтева клијента да добије тренутне податке са сензора.

```
def listener(client, address):
    print ("\nAccepted connection from: ", address,'\n')
    global sensor_val
    with clients_lock:
        clients.add(client)
    try:
        while True:
            data = client.recv(1024)
            if int(data) == 0:
                catch = sensor_val
                print('\nSENDING: ', str(catch),'\n')
                char = str(int(catch))
                client.send(bytes(char,encoding='utf8'))
    except ValueError:
        print("\nClient Disconnected\n")
```

Слика 14. Приказ имплементације петље за конектованог клијента



Слика 15. Алгоритам рада RaspberryPi дела апликације

9 Закључак

Након повезивања свих саставних компоненти система и тестирања, може се закључити да систем успешно функционише. *Web* камера у континуитету снима и прослеђује снимак на даљу обраду, чиме се може закључити да ли је особа која се испред ње налази ауторизована да приступи систему (админ) или није.

Евентуално би се могло добити на функционалности када би детекционо-рекогниционi сегмент програма био имплементиран на *RaspberryPi*-у, док би рачунар у том случају био искључен из пројекта. На *RaspberryPi* би осим сензора морала бити повезана *RaspberryPi* камера која се повезује на *RaspberryPi* другачије у односу на обичне *USB* камере. Ово би поред минијатурности допринело и мобилности самог уређаја, али мана такве реализације је губитак перформанси детекције и препознавања. Поред тога што нема посебну графичку картицу, овај мини рачунар има знатно лошије процесорске перфомансе од *PC*-а те би извршавање тако захтевне апликације за њега представљало озбиљно оптерећење и дошло би до прегревања уређаја. Осим тога, *RaspberryPi* камера има посебан начин преузимања фрејмова у односу на остале камере те би морало доћи до промене у иницијализацији и имплементацији система.

10 Референце

[1] https://github.com/jovanSlavujevic96/IoT_detection_PES - *Github* репозиторијум пројекта, приступљено: април 2020.

[2] https://www.raspberrypi.org/documentation/usage/gpio/ - *GPIO* пинови, *RaspberryPi* званична документациона страница, приступљено: април 2020.

[3] https://docs.particle.io/assets/datasheets/electronsensorkit/MQ-2.pdf - *MQ-2* званична документација, приступљено: април 2020.

[4] https://www.tensorflow.org/install/pip – упутство за инсталацију *Tensorflow* платформе, званична *Tensorflow* страница, приступљено: јануар 2020.

[5] https://www.geeksforgeeks.org/ml-training-image-classifier-using-tensorflow-object-detection-api/ - упутство за тренирање неуронске мреже помоћу *Tensorflow Models framework*-а, *Geeks-for-Geeks* чланак, приступљено: март 2020

[6] https://opencv.org/ - званична *OpenCV* страница, приступљено: април 2020.

[7] https://towardsdatascience.com/face-detection-in-2-minutes-using-opencv-python-90f89d7c0f81 – упутство за детектовање човековог лица помоћу *OpenCV* платформе, *Towards Data Science* чланак, приступљено: март 2020

[8] http://www.willberger.org/cascade-haar-explained/ - чланак о функционисању детекције уз помоћ класификатора, приступљено: април 2020.