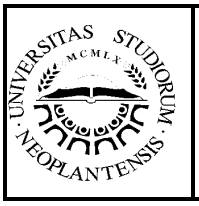


УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ПРОЈЕКТНИ ЗАДАТАК

Кандидати: **Јулија Јелићанин** **RA 62/2020**
Милица Војновић **RA 59/2020**
Михаило Бабић **RA 52/2020**
Павле Васиљевић **RA 207/2020**

Предмет: **Оперативни системи за рад у реалном времену**

Тема рада: **Remote door locking check, alarm and survelience**

Ментор рада: **др Милош Суботић**
МСц Милош Пилиповић

Нови Сад, јануар, 2023.

1. УВОД

1.1. Задатак пројекта

Назив додељеног пројекта је **Remote door locking check, alarm and survelience**, или преведено **Даљинска провера закључавања врата, аларм и надзор**. Пројекат се практично састоји из три одвојене целине, које могу да фигуришу једна без друге. Наш задатак је да направимо одвојене целине и да их користимо заједно, као нека врста кућног надзора.

1.2. Даљинска провера закључавања врата

Идеја за прву целину, је да се направе врата са два уграђена сензора. Један сензор води рачуна о томе да ли су врата отворена/затворена, док други санзор води рачуна о томе да ли је брава закључана/откључана. Сигнали добијени помоћу сензора се шаљу на Raspberry Pi преко GPIO pin-ова. Добијени подаци се користе у python програму, и као излаз добијамо информацију о томе да ли су врата отворена/затворена и закључана/откључана. Такође, у случају да је корисник заборавио да закључа врата, програм му шаље мејл, који га обавештава да је заборавио да закључа врата.

1.3. Аларм

Други део пројекта везан је за сензор који детектује инфрацрвено зрачење. Сензор је преко GPIO pin-ова повезан на Raspberry Pi рачунар, и преко њих шаље информације. Добијене информације се затим користе у python програму, који проверава да ли је сензор регистровао покрет. У случају да јесте, шаље мејл кориснику да је регистрован покрет.

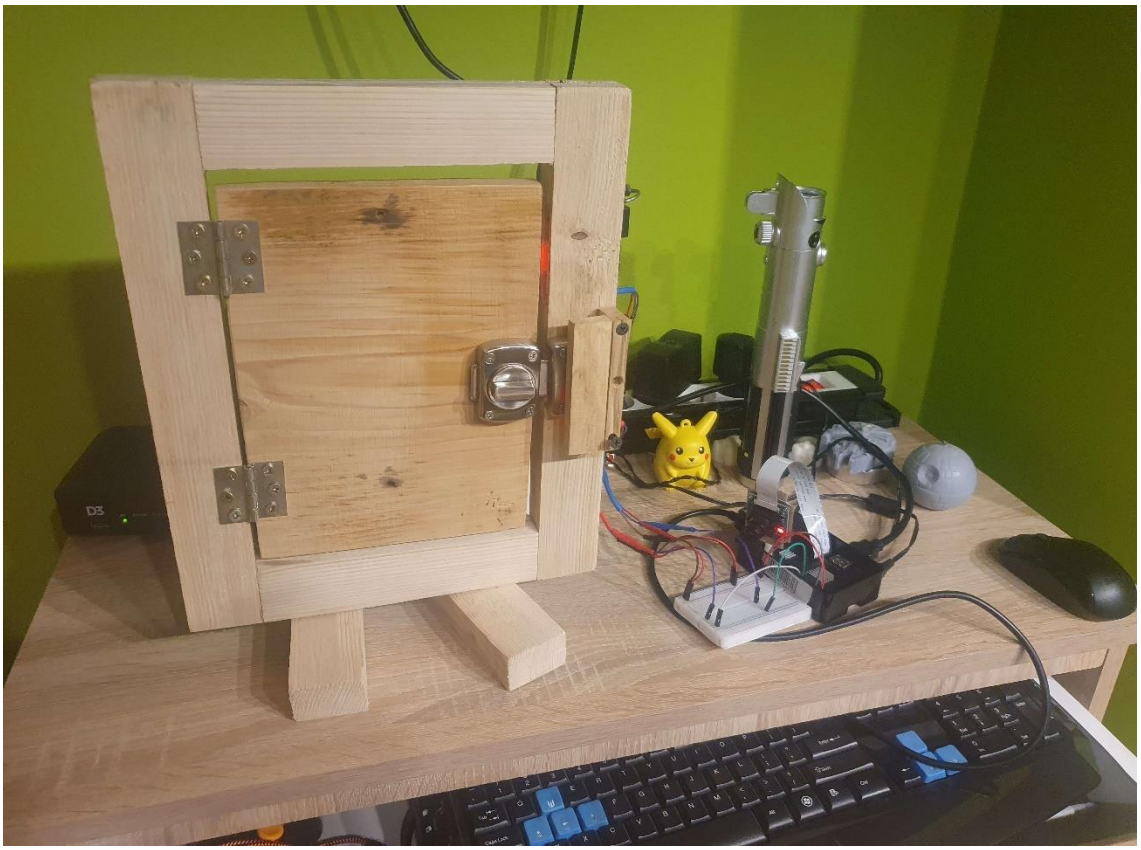
1.4. Надзор

Надзор је трећи део пројекта везан за Raspberry Pi камеру која је преко посебног интерфејса повезан са Raspberry Pi рачунаром. Камера шаље слике које хвата, и ставља их у buffer. Садржај тог buffer-а се приказује преко HTML странице, где се низ слика практично претвара у видео. Страници се приступа преко локалног HTTP сервера, који је покренут на Rasspberry Pi-у и омогућава кориснику прегледање видеа у претраживачу.

2. КОНЦЕПТ РЕШЕЊА

2.1. Повезивање

За израду овог дела пројекта коришћене су компоненте: **два сензора близине, сензор покрета и Raspberry Pi камера.** Такође је коришћен функционални модел врата, направљен од дрвета, са металном бравом. На врата су уграђени сензори близине, један за отварање/затварање, и други за закључавање/откључавање, као што смо и претходно рекли.



PIR (Passive Infrared) сензори су сензори који детектују инфрацрвено зрачење емитовано од стране објеката са високом температуром. Сензори су пасивни у смислу да не емитују никакво сопствено зрачење за детекцију, већ само детектују инфрацрвено зрачење које емитују објекти у окружењу.

На самом сензору налазе се три пина за повезивање са Raspberry Pi рачунаром (VCC pin од 5V, output pin и ground pin). Такође се на сензору налазе два потенциометра који нам омогућавају калибрисање сензора, тј. прилагођавање осетљивости сензора као и временског периода између узастопних слика.

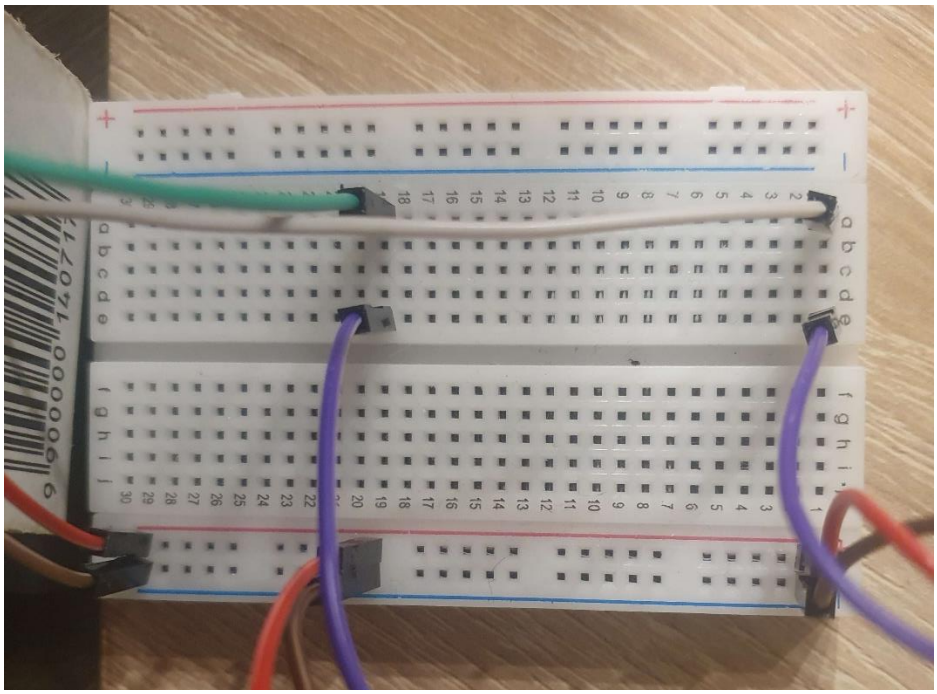
Raspberry Pi камера је повезана преко CSI (Camera Serial Interface) на Raspberry Pi. Даље се све ради софтверски.

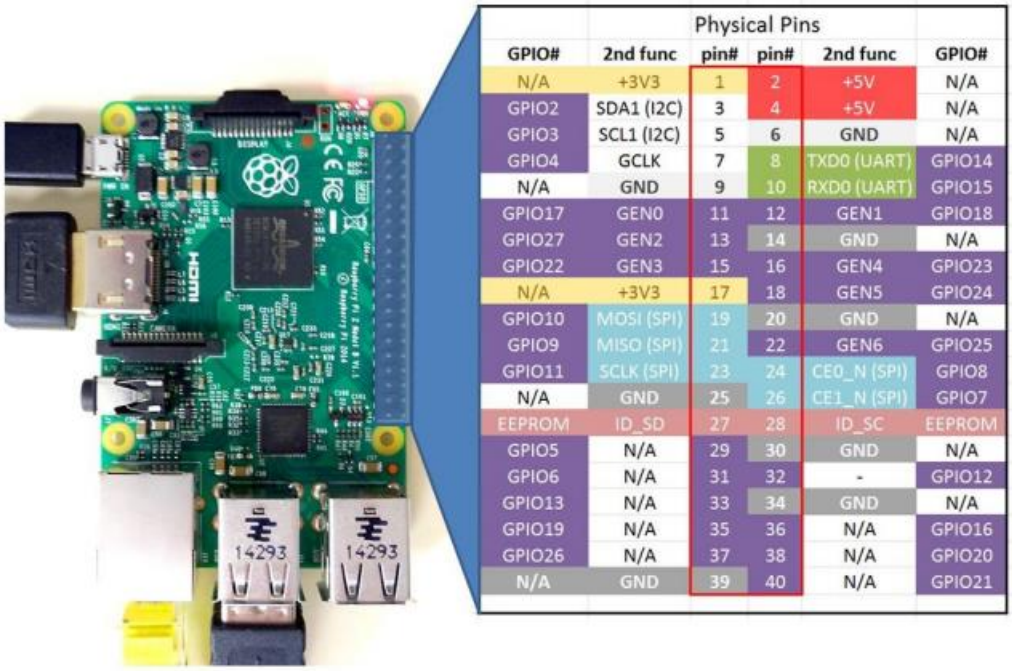
Сензори близине (Proximity sensor) су постављени, један у раму врата , а други на месту где улази брава. Они пројектују инфрацрвено зрачење, док друга диода у сензору региструје промену. Ако региструје објекат у својој близини, у овом случају се ради о раздаљини од 1 cm, сензор враћа логичку ‘0’ и пали се LED диода, која се налази на штампаној плочи на којој је сензор.

Сензор близине за закључавање/откључавање:
Ред 1 протоборда. Колона E **output**. Колона A **input - GPIO pin 24** (pin 18).

Сензор близине за отварање/затварање:
Ред 19 протоборда. Колона E **output**. Колона A **input - GPIO pin 23** (pin 16).
За напајања су коришћене црвене жице, а браон жице су маса.

Сензор покрета:
GPIO pin4 (pin 7) **output**. Напон на физички pin 2, ground на pin 6.





		Physical Pins			
GPIO#	2nd func	pin#	pin#	2nd func	GPIO#
N/A	+3V3	1	2	+5V	N/A
GPIO2	SDA1 (I2C)	3	4	+5V	N/A
GPIO3	SCL1 (I2C)	5	6	GND	N/A
GPIO4	GCLK	7	8	TXD0 (UART)	GPIO14
N/A	GND	9	10	RXD0 (UART)	GPIO15
GPIO17	GEN0	11	12	GEN1	GPIO18
GPIO27	GEN2	13	14	GND	N/A
GPIO22	GEN3	15	16	GEN4	GPIO23
N/A	+3V3	17	18	GEN5	GPIO24
GPIO10	MOSI (SPI)	19	20	GND	N/A
GPIO9	MISO (SPI)	21	22	GEN6	GPIO25
GPIO11	SCLK (SPI)	23	24	CE0_N (SPI)	GPIO8
N/A	GND	25	26	CE1_N (SPI)	GPIO7
EEPROM	ID_SD	27	28	ID_SC	EEPROM
GPIO5	N/A	29	30	GND	N/A
GPIO6	N/A	31	32	-	GPIO12
GPIO13	N/A	33	34	GND	N/A
GPIO19	N/A	35	36	N/A	GPIO16
GPIO26	N/A	37	38	N/A	GPIO20
N/A	GND	39	40	N/A	GPIO21

2.2. Имплементација

2.2.1. RPi.GPIO библиотека

RPi.GPIO је Python библиотека која омогућава рад са GPIO pin-овима на Raspberry Pi рачунару. Користи се да се омогући комуникација са различитим сензорима и периферним уређајима који су повезани са Raspberry Pi рачунаром преко GPIO pin-ова. У овом пројекту, RPi.GPIO библиотека се користи да се омогући компуникација са свим сензорима. Импортовањем ове библиотеке, програм ће имати приступ функцијама које омогућавају читање података са сензора и контролу рада сензора преко одговарајућих pin-ова на Raspberry Pi рачунару.

Постављање мода на **GPIO.BCM** преко **GPIO.setup( )** омогућава да се користе одговарајуће функције библиотеке RPi.GPIO за рад са pin-овима на Raspberry Pi рачунару према Broadcom-овом систему нумерације, док постављање мода на **GPIO.BOARD** користи физички систем нумерације.

GPIO.setup(,) функција се користи, како би се pin 7 (GPIO4), pin 16 и pin 18 конфигурисали као улазни pin-ови, што омогућава да Raspberry Pi рачунар правилно комуницира са сензорима који су повезани на те pin-ове, омогућавајући програму да чита податке од сензора. Pin 7 служи за сензор покрета, pin 16 за сензор врата, и pin 18 за сензор браве.

2.2.2. sendMail( ) функција

Први корак ове функције је да импортујемо smtplib библиотеку која се користи за рад са **Simple Mail Transfer Protocol (SMTP)** серверима, што је протокол који се користи за слање e-mail-a. Овом библиотеком се омогућава коришћење Python кода за слање e-mail-a преко SMTP сервера.

За слање e-mail-a потребно је унети информације о налогу који се користи за слање (email_address и email_password) као и адресу примаоца e-mail-a. Такође је потребно да креирамо садржај e-mail-a који нам садржи информације, шта је активирало слање обавештења, као и линк за приступ stream-у који је покренут заједно са позивом ове функције.

Након повезивања са Gmail сервером за слање e-mail-a (коришћењем SMTP протокола и одговарајућег порта (587)) активира се TLS – криптографски протокол који се користи за шифровање комуникације између клијента и сервера.

```
def SendEmail():
    # Set the email
    ip = netifaces.ifaddresses('wlan0')[netifaces.AF_INET][0]['addr']
    subject = 'Detektovano kretanje'
    body = 'Aktivniran je senzor pokreta, proveri kameru na streamu:\n' + 'http://' + ip \
           + ':8080/index.html' + '\n\n\n' + 'Vreme: ' + time.strftime("%H:%M:%S", time.localtime())

    # Set up the SMTP server
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login('osurv.projekat@gmail.com', 'gtmvzytwxconqudt')

    # Set up the message
    msg = "Subject: {}\n\n{}".format(subject, body)

    # Send the email
    server.sendmail('osurv.projekat@gmail.com', 'osurv.projekat@gmail.com', msg)

    # Disconnect from the server
    server.quit()
    time.sleep(15)
```

Логујемо се на налог за слање e-mail-а користећи унете информације и шаљемо e-mail чији саджај се шаље на адресу примаоца e-mail-а. Након што се e-mail успешно пошаље, конекција са сервером се затвара.

Ова функција се позива у два наврата. При детекцији покрета, обавештава корисника и пошаље кориснику поруку. Исто тако, када су врата затворена и откључана дуже од 2 минуте, обавештава корисника о томе.

2.2.3. write_door_state(,) функција

Као параметре, ова функција прима стање врата и браве. Врата имају два стања: отворено (логичко ‘1’) и затворено (логичко ‘0’). Исто тако, брава има два стања: откључано (логичко ‘1’) и закључано (логичко ‘0’). Добијене информације се уписују у **status.txt** датотеку. Из ове датотеке се касније чита и на сваку добијену слику са Raspberry Pi камере се исписује стање врата. Слика на преносу се стално мења, па тако и испис везан за стање врата.

```
42
43  def write_door_state(door,lock):
44      doorOmsg=" door:open"
45      doorCmsg=" door:closed"
46      lockOmsg=" lock:open"
47      lockCmsg=" lock:closed"
48      buff=open('status.txt','w')
49      if(door):
50          buff.write(doorOmsg+'\n')
51      else:
52          buff.write(doorCmsg+'\n')
53      if(lock):
54          buff.write(lockOmsg)
55      else:
56          buff.write(lockCmsg)
57      buff.close()
58
```

2.2.4. главни део door.py датотеке

Бесконачна while петља је главни део овога програма. Она стално проверава стање инпута са GPIO pin-ова. Позива функцију за исписивање стања врата. Истовремена проверава да ли су врата затворена и откључана, јер би то значило да је корисник заборавио да их закључа. Након што је прошло одређено време без промене овога стања (2 минуте), позива се функција за слање e-mail-а, која обавештава корисника.

2.2.5. doorStatus( ) функција

Функција која из status.txt датотеке, у којој се налази стање врата, стално чита и уписује на слику добију из Raspberry Pi камере.

```
def doorStatus(): #outputs  door and lock status to the stream
    while threadactiveflag:
        buff=open('status.txt','r')
        overlaytext=buff.read()
        camera.annotate_background=picamera.Color('black')
        camera.annotate_text=overlaytext
        buff.close()
```

2.2.6. DataStream( ) класа

Класа се састоји од три компоненте: frame, buffer, condition. Затим, класа узима слику по слику добијену са Raspberry Pi камере, што су у суштини фрејмови видео снимка који се смештају на buffer.

```
class DataStream(object):
    def __init__(self):
        self.frame = None
        self.buffer = io.BytesIO()
        self.condition = threading.Condition()

    def write(self, streambuf):  #producer via start recording method
        if streambuf.startswith(b'\xff\xd8'):
            # New frame, copy the existing buffer's content and notify all
            # clients it's available
            self.buffer.truncate()
            with self.condition:
                self.frame = self.buffer.getvalue()
                self.condition.notify_all()
            self.buffer.seek(0)
        return self.buffer.write(streambuf)
```

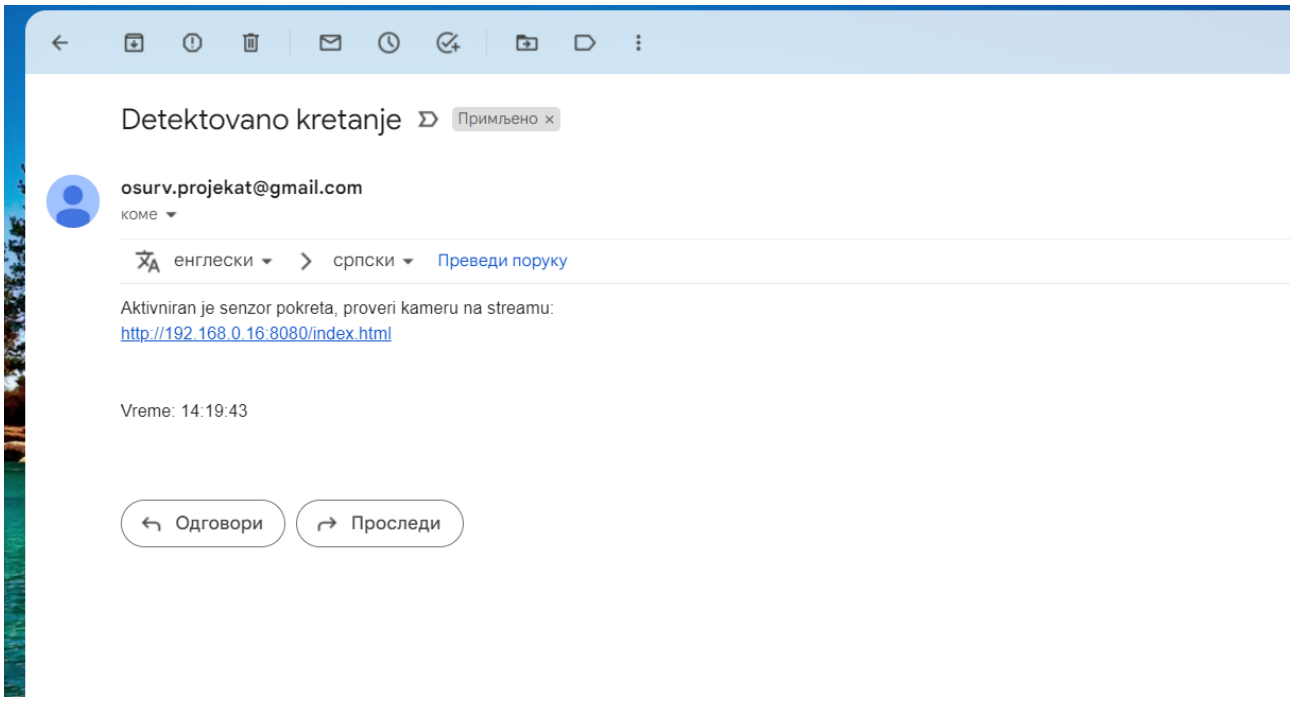
2.2.7. StreamHandler( ) класа

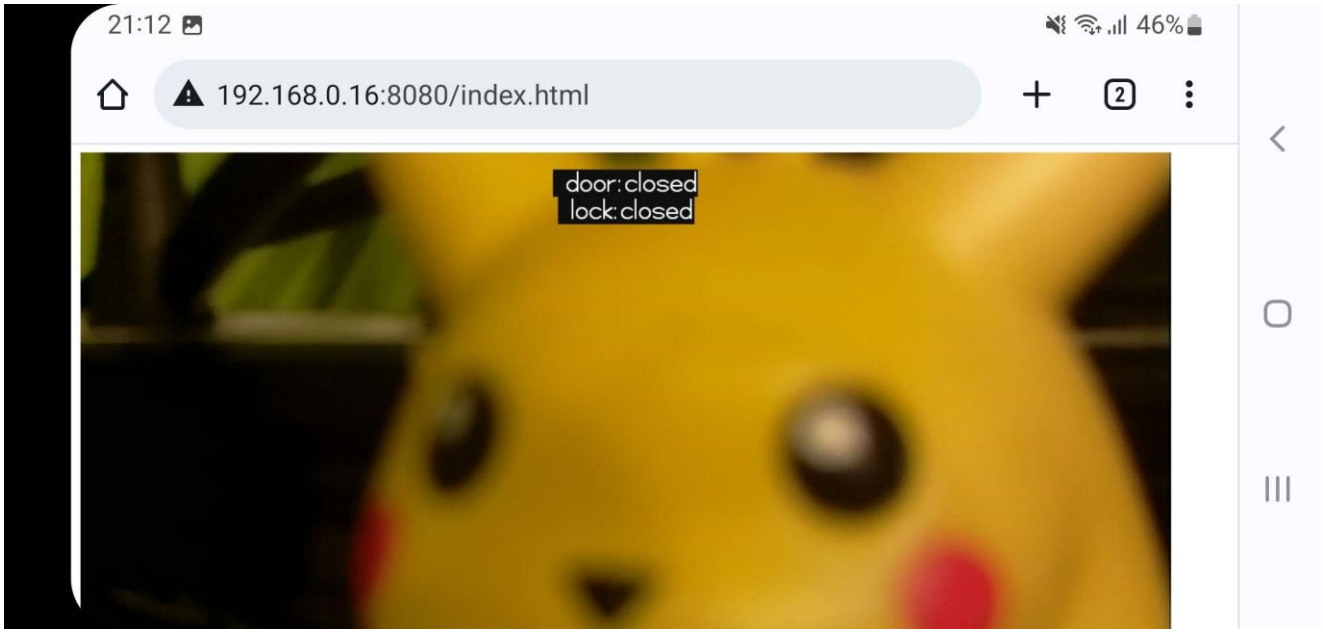
Класа која позива **do_SET( )** функцију. Ова функција је у суштини везана за приказ stream-а у веб претрживачу. Додељују се параметри везани за stream. У try-блоку се налази ‘потрошач’ за фрејмове, који су направљени у оквиру **StartRecording** методе, и уписани помоћу DataStream класе, write методе. Она обавештава потрошача сваки пут када је спреман нови фрејм.

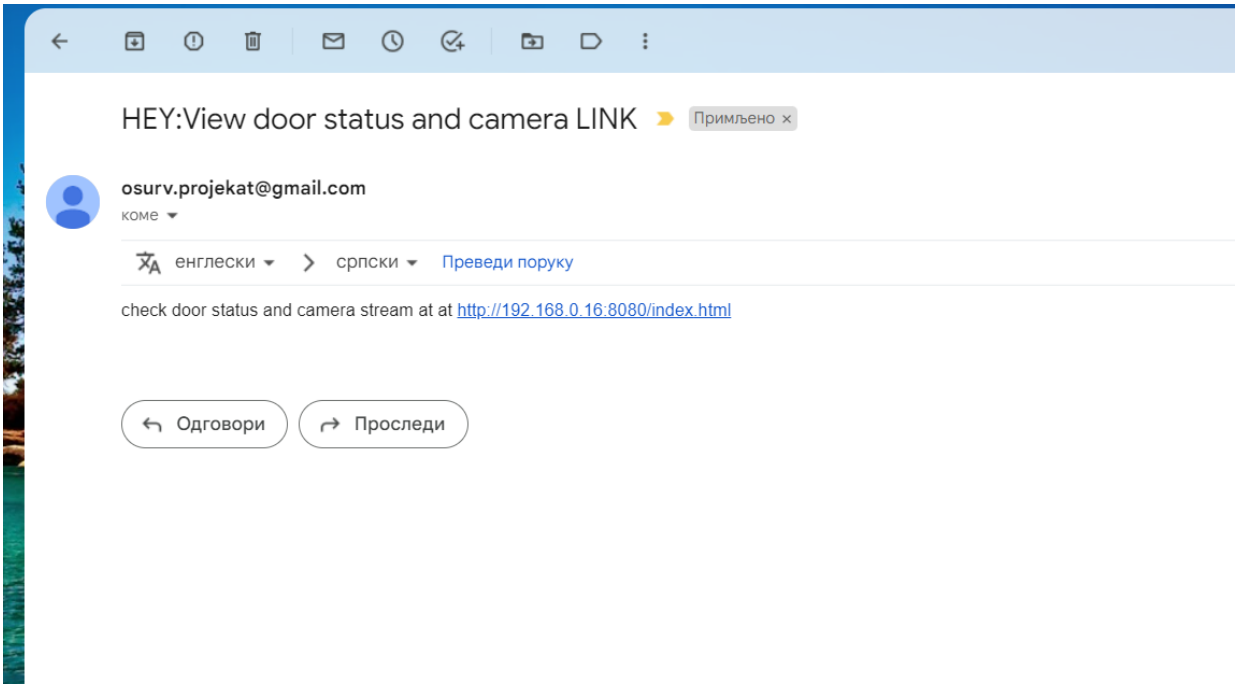
```
class StreamHandler(server.SimpleHTTPRequestHandler):
    def do_GET(self):
        if self.path == '/index.html':
            content = WPAGE.encode('utf-8')
            self.send_response(200)
            self.send_header('Content-Type', 'text/html')
            self.send_header('Content-Length', len(content))
            self.end_headers()
            self.wfile.write(content)
        elif self.path == '/stream.mjpg':
            self.send_response(200)
            self.send_header('Content-Type', 'multipart/x-mixed-replace; boundary=FRAME')
            self.end_headers()
            try:
                while True: #consumer for the frames
                    with output.condition:
                        output.condition.wait()
                        frame = output.frame
                    self.wfile.write(b'--FRAME\r\n')
                    self.send_header('Content-Type', 'image/jpeg')
                    self.send_header('Content-Length', len(frame))
                    self.end_headers()
                    self.wfile.write(frame)
                    self.wfile.write(b'\r\n')
            except Exception as e:
                logging.warning(
                    'Removed streaming client %s: %s',
                    self.client_address, str(e))
        else:
            self.send_error(404)
            self.end_headers()
```

3. ЗАКЉУЧАК

Након имплементирања кода, коришћења Raspberry Pi рачунара и периферијских уређаја, долазимо до добијеног резултата. Сви сензори и камера, повезани са жицама, раде оно што је било задато. Добијамо улазне информације, и користимо Python програм за обраду ових информација. За многе ствари постоје python библиотеке, којима смо се послужили приликом израде овог пројекта. Као излаз, добијамо видео пренос камере, испис стања врата, и добијамо e-mail-ове по потреби.




21:12
46%
192.168.0.16:8080/index.html
door: closed
lock: closed


HEY:View door status and camera LINK
Примљено
osurv.projekat@gmail.com
коме
енглески
српски
Преведи поруку
check door status and camera stream at at http://192.168.0.16:8080/index.html
Одговори
Проследи

Садржај: