**4. óra: Mozgásérzékelő**

1. 5 perces beszámoló tetszőleges tartalommal és üzenettel valamint tetszőlegesen választott retorikai eszközökkel. Nekünk kell eldönteni, hogy mi lehetett az üzenet. Figyelni kell a beszéd felépítésére (bevezető, fő rész és konklúzió) és arra, hogy a tartalom minden eleme támogassa az üzenetet, illetve, hogy használjunk retorikai eszközöket.
2. 3 perces véleményalkotás  
   - bevezető, fő rész, konklúzió  
   - 2 pozitív és egy konstruktív javaslat  
   - konklúzióban összefoglalni mi lehetett az üzenet, volt-e retorikai eszköz használva és adni még egy pozitív megjegyzést.
3. Villanyrendőr készítés:
   1. *Bevezető*:  
      Sokszor előfordul veled, hogy tanulás helyett inkább a barátokkal beszélgetsz vagy netezel vagy egyéb jobb elfoglaltságot találsz a szobádban. Természetesen a szülők előszeretettel veszik ezt észre és kihasználják, hogy megdorgálhatnak. Emiatt elgondolkoztál azon, hogy lehetne előre tudni, hogy ők a szoba felé tartanak, anélkül, hogy rájönnének, hogy meglesed őket.

Egy jól elrejtett megfigyelő kamera jól hangzik, ami a szobád előtti teret figyeli. Sajnos a kamera képét nem használhatod, mert a szüleid is meglátnák, hogy megfigyeled a környéket. Így arra gondoltál, hogy a kamera képéből egy algoritmus alapján megállapítod van-e mozgás, és ha igen, akkor egy LED izzóval jelzed magadnak, hogy valaki közelít.

* 1. *Előkészületek*:  
     - Elkészíteni az áramkört.

- Beimportálni a csomagokat amik segtik a munkánkat: cv2, time, numpy, matplotlib, datetime, gpiozero és saját függvények.

- Létrehozni a kapcsolatot a webkamerával az opencv csomag használatával.

- Inicializálni a LED-et.

- Képelemző függvényt írni, ami érzékeli a mozgást.

- Függvényt írni a LED vezérlésére.

- Kinyerni a kamera képnek szélességét és magasságát.

- Inicializálni egy ábrát, ami jelzi majd, ha mozgás történik.

- Végtelen while ciklusban rögzíteni és elemezni a kamera képet, eldönteni volt-e mozgás, frissíteni a mozgást jelző grafikonunkat.

- A q billentyű lenyomásával jelezni, hogy befejeztük.

- Bezárni a kapcsolatot a kamerával.

* 1. *Elektronika:*  
     - Legyen a webcam előkészítve.  
     - Szereljük a LEDet a breadboardra a hozzájuk tartozó ellenállásokkal és kössük őket össze a Raspi GPIO-ival.
  2. *Kód:*  
     - teszteljük le a kamera működését avval, hogy kiolvasunk képet belőle és ábrázoljuk.  
     - Ellenőrízzük le, hogy a LED vezérelhető-e a Raspberryről.  
     - A raspberry\_functions\_studentVersion.py kódban egészítsük ki a *prepare\_data* függvényt, hogy az előkészítse az adatokat.  
       
     - Mutassuk be a kamera pixelméreteit lekérő függvényeket:

1. cap = cv2.VideoCapture(0)
2. height = cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT)
3. width = cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH)

- Szemléltessük a *cv2.equivaliyeHist* függvény működését:  
plt.imshow(cv2.equalizeHist(apple), cmap='gray')  
  
- Mutassuk meg, mi történik, ha egy háttérből két képet készítünk, és mi ha mozgás történik közben rajta. Mi lesz, ha kivonyjuk egymásból a képeket:  
bck = cv2.imread('./prog03/bckgnd.jpg', 0) # hatter beolvasasa

bck = cv2.equalizeHist(bck)

apple = cv2.imread('./prog03/apple.jpg', 0) # uj kep beolvasasa

apple = cv2.equalizeHist(apple)

height, width = bck.shape # kep meretenek meghatarozasa

fig, ax = plt.subplots(1,2, figsize=(15,30)) # abra definialas

ax[0].imshow(bck, cmap='gray') # hatter plot

ax[1].imshow(apple, cmap='gray') # almas kep plot  
delta = apple - bck # hatter levonasa az almas kepbol

delta\_bck = bck - bck # hatter levonasa hatterbol

# thresholdolas

ret, delta\_thresh = cv2.threshold(delta, 50, 255, cv2.THRESH\_BINARY)

fig, ax = plt.subplots(1,3, figsize=(15,30))

ax[0].imshow(delta, cmap='gray')

ax[1].imshow(delta\_thresh, cmap='gray')

ax[2].imshow(delta\_bck, cmap='gray')  
- Mutassuk be, mit csinál a cv2.threshold függvény:

ret, frame = cv2.threshold(frame, limit, 255, cv2.THRESH\_BINARY)

* 1. *Tesztelés:*  
     - teszteljük a kamerák működését.
  2. *Fejlesztés:*  
     -Változtassuk meg a led\_control függvényben a threshold paraméter értékét és figyeljük meg az hogy befolyásolja a mozgásérzékelőnk működését.

- Változtassuk meg a motion\_sensor függvényben a limit paraméter értékét és figyeljük meg az hogy befolyásolja a mozgásérzékelőnk működését.

Kód minta:

import cv2, time

import numpy as np

import datetime as dt

import matplotlib.pyplot as plt

from gpiozero import LED

from raspberry\_functions\_studentVersion import prepare\_data, frame2grayscale

# inicializald a kamerat

cap =

# inicializald a ledet

led =

def motion\_sensor(frame, calibration, limit=2):

    # 1) alkalmazd a cv2.threshold fuggvenyt a frame kepen, aminek min erteke a limit (bemeneti ertekkent megadhato)

    # max erteke legyen 255, es hasznald a cv2.THRESH\_BINARY modszert

    # 2) szamitsd ki a thresholdolt matrix osszes elemenek osszeget es oszd el a calibration bemeneti valtozoval

    # az igy kapott ertek lesz a visszaadott erteke a fuggvenynek

    pass

def led\_control(led, value, threshold = 0.8):

    # ha a value erteka nagyobb mint threshold erteke

    # akkor bekapcsolni a ledet

    # ha kisebb akkor kikapcsolni a ledet

    pass

height = cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT)

width = cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH)

calibration = height \* width \* 256

# olvasd ki a kepet

ret, frame =

# alakitsd szurke szinskalara a frame2grayscale fuggvenyt hasznalva

frame =

# javitsd fel a kontrasztot a cv2.equalizeHist fuggveny segitsegevel

frame =

max\_count = 20

dplot = [dt.datetime.now()]

# ird be a mozgas indikator erteket a tplot listaba (a motion\_sensor fuggveny altal adott ertek)

tplot = []

plt.ion()

figure, ax = plt.subplots(1,1, figsize=(8,6))

line1, = ax.plot(dplot, tplot, 'o-')

while True:

    # olvasd ki az uj kepet

    ret, frame1 =

    # alakitsd at szurke szinskalara

    frame1 =

    # javitsd fel a kontrasztot

    frame1 =

    # szamold ki a hatter levont kepet (mostani kep - elozo kep)

    delta\_frame =

    # hatarozd meg a mozgas indikator erteket a delta\_frame-bol

    motion =

    dplot, tplot = prepare\_data(dt.datetime.now(), motion, dplot, tplot)

    # vezereld a led mukodeset a led\_control fuggvennyel

    ........

    # frissitsd a hatterkepet (a jelenlegi uj kep = regi kep)

    frame =

    line1.set\_xdata(dplot)

    line1.set\_ydata(tplot)

    ax.set\_ylim(min(tplot)\*0.99,max(tplot)\*1.01) # +1 to avoid singular transformation warning

    ax.set\_xlim(min(dplot),max(dplot))

    figure.canvas.draw()

    figure.canvas.flush\_events()

    plt.gcf().autofmt\_xdate()

    if cv2.waitKey(100) & 0xFF == ord('q'):

        break

    time.sleep(0.2)

cap.release()

# Bezarunk minden ablakot, amit a program megnyitott

cv2.destroyAllWindows()