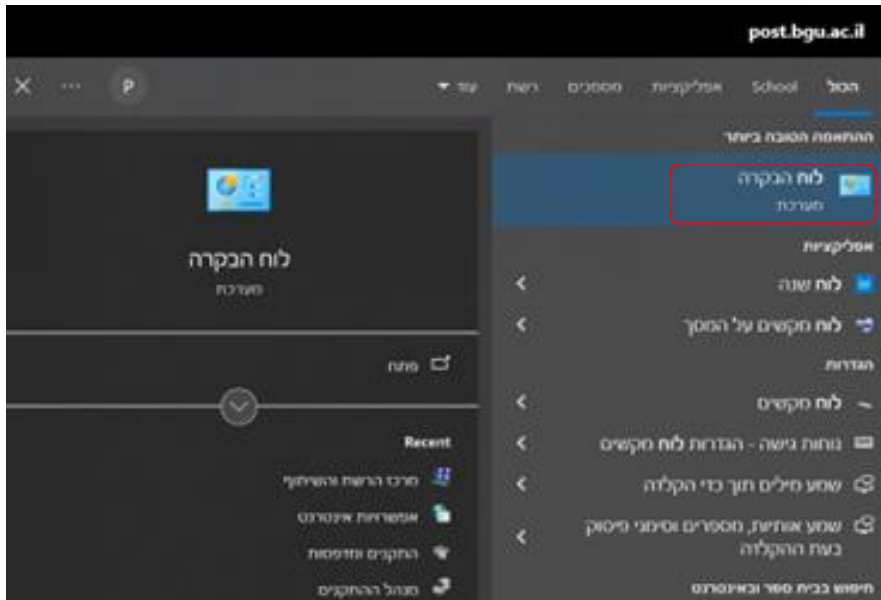


Jetson Manual

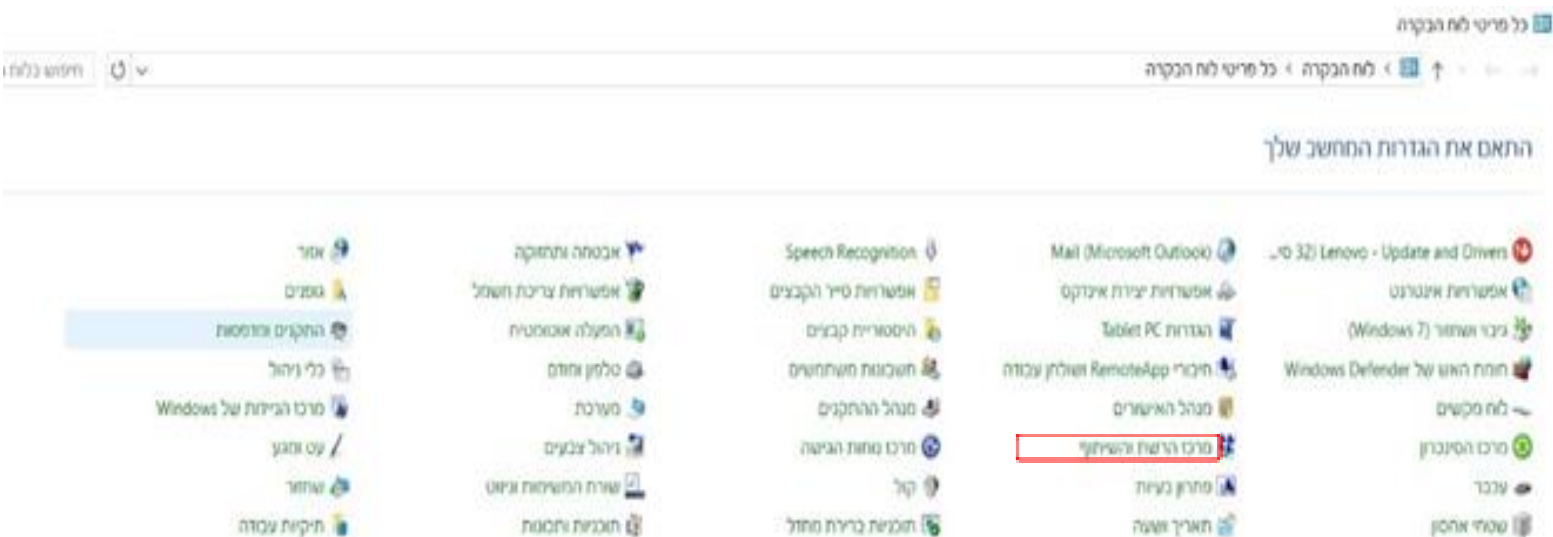
שיתוף אינטרנט למעבד ה-Jetson Nano

לעיתים יש בעיה לחבר את ה-Jetson ישירות לאינטרנט, על כן נשתמש במחשב נייד אשר מחובר לרשת WIFI מסוימת ונשתמש במחשב הנייד בתור נתב.

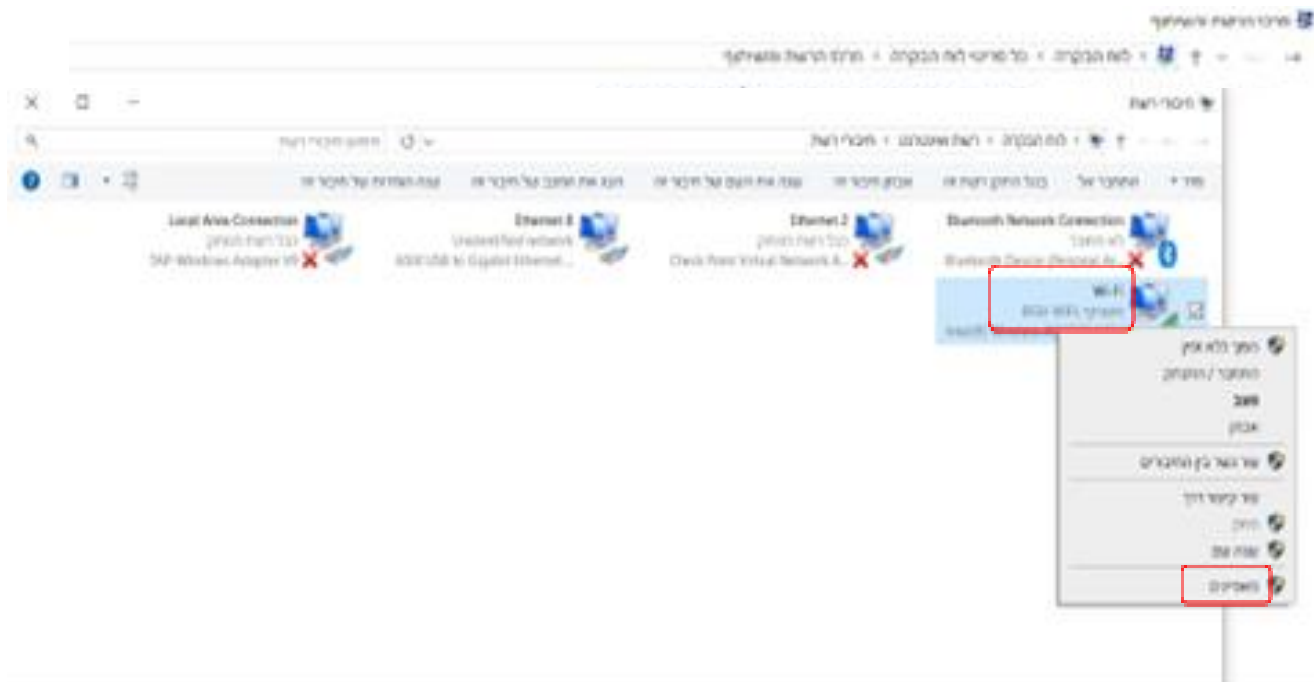
נכנסים במחשב ללוח הבקרה



לוחצים על מרכז הרשת והשיתוף

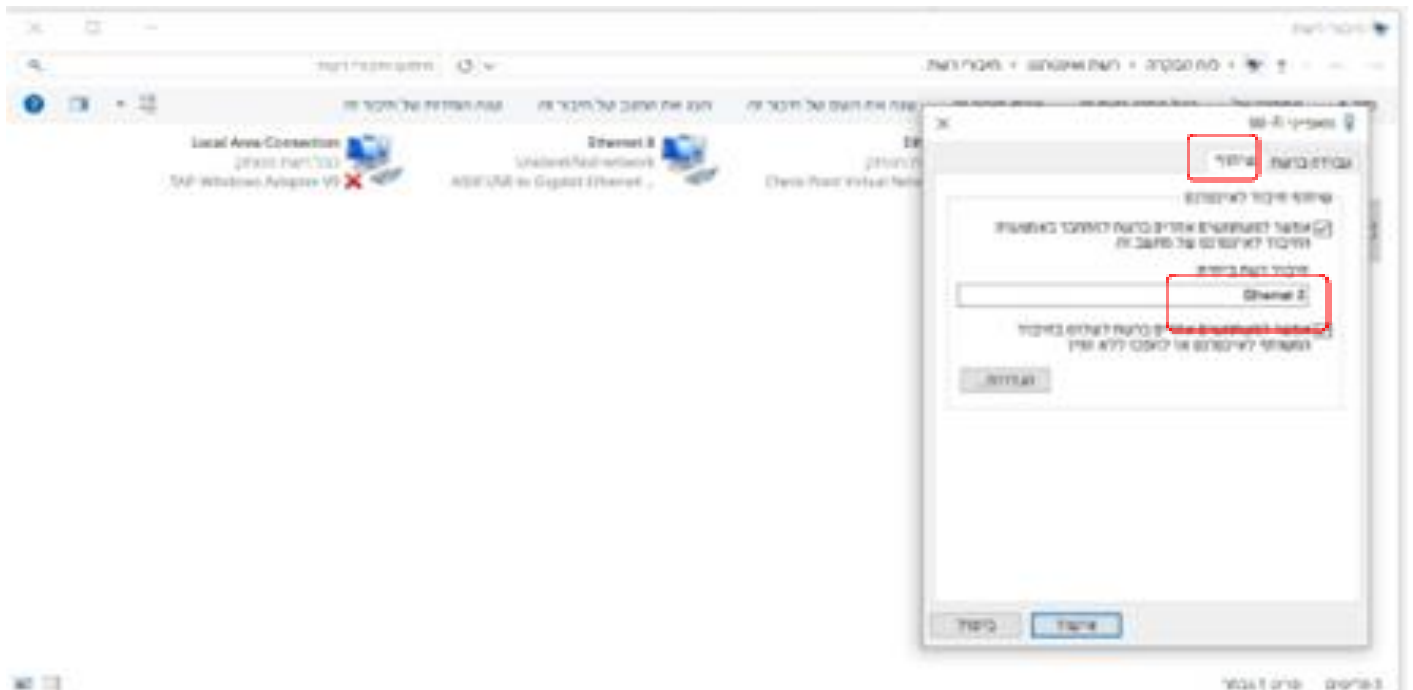


לוחצים על שנה הגדרות מתאם



לוחצים עם המקש הימני על אייקון ה-wifi ואז בוחרים במאפיינים .

בסימניה של שיתוף, ב"חיבור רשת ביתית" יש לבחור את אחת האופציות מבין Ethernet X (X מהווה מספר כלשהו).



לאחר בערך כדקה אמור להיות אינטרנט ב – Jetson.
אם אין חיבור לאינטרנט אז יש לחזור על הפעולה האחרונה שוב ולבחור את אופציית האינטרנט האחרת .

צריבת Ubuntu 18.04

ראשית נוריד על כרטיס זיכרון (sd card) את ubuntu 18.04.
*ניתן לעשות זאת על כרטיס זיכרון בעל נפל של 32GB, אך אנו לקחנו כרטיס זיכרון של 256GB על מנת שיהיה מספיק שטח אחסון לאחסון התמונות שהמצלמה תצלם.

נבצע זאת על ידי הורדה מהאתר הרשמי של ג'טסון ננו:

<https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit>

נבחר בצד באופציה של Write Image to the microSD Card ובחלון נבחר בהורדה של ה-
img ולא ר מכן ב-linux:

Write Image to the microSD Card

To prepare your microSD card, you'll need a computer with Internet connection and the ability to read and write SD cards, either via a built-in SD card slot or adapter.

1. Download the Jetson Nano Developer Kit SD Card Image, and note where it was saved on the computer.

2. Write the image to your microSD card by following the instructions below according to your computer's operating system:
Windows, macOS, or Linux.

INSTRUCTIONS FOR WINDOWS

INSTRUCTIONS FOR MACOS

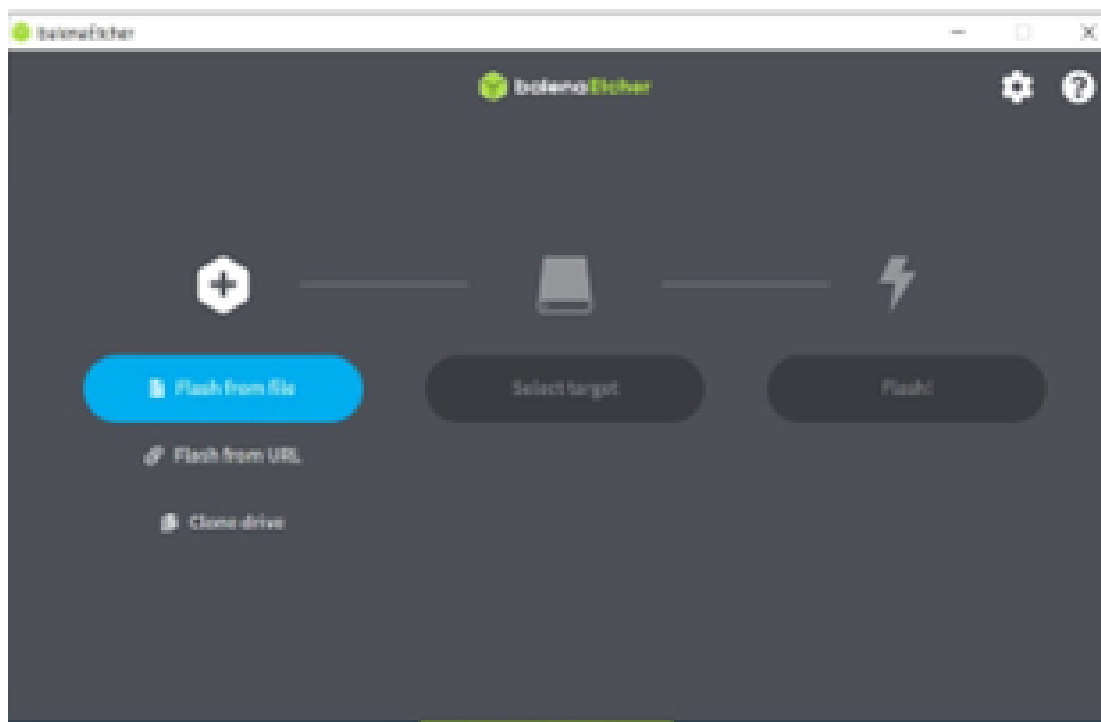
INSTRUCTIONS FOR LINUX

You can either write the SD card image using a graphical program like Etcher, or via command line.

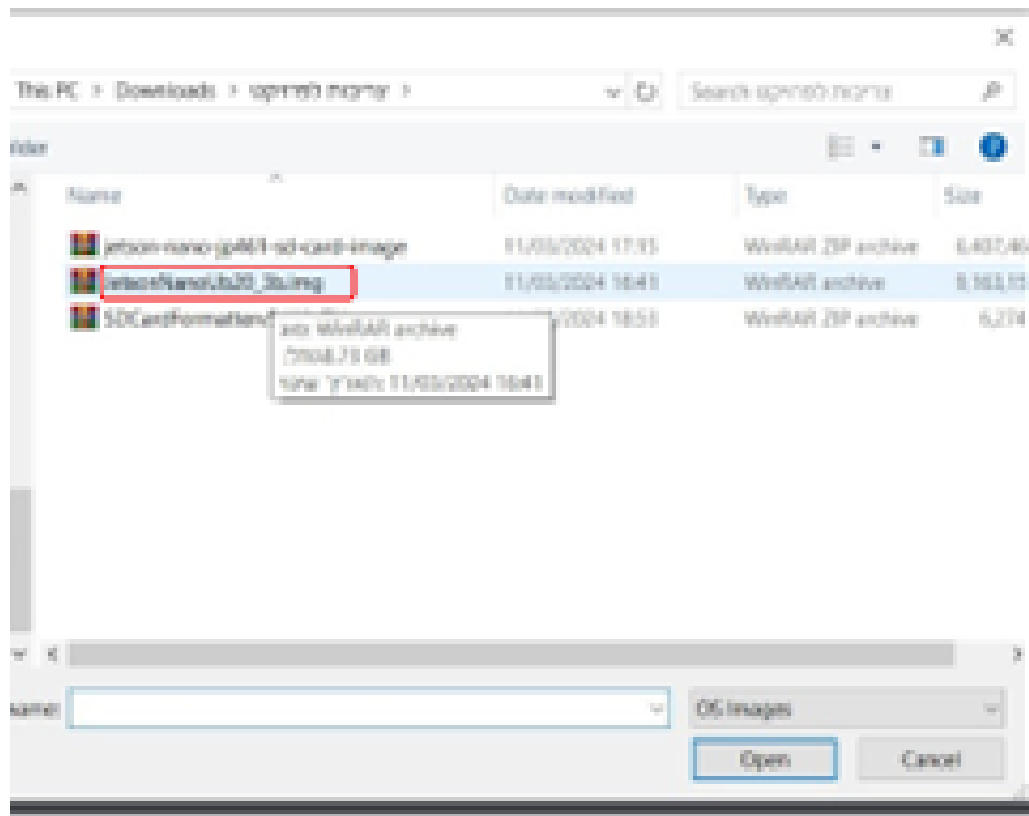
נוריד את תוכנת ה-Etcher על מנת לכתוב לכרטיס זיכרון. את התוכנה ניתן להוריד מ:
<https://etcher.balena.io>



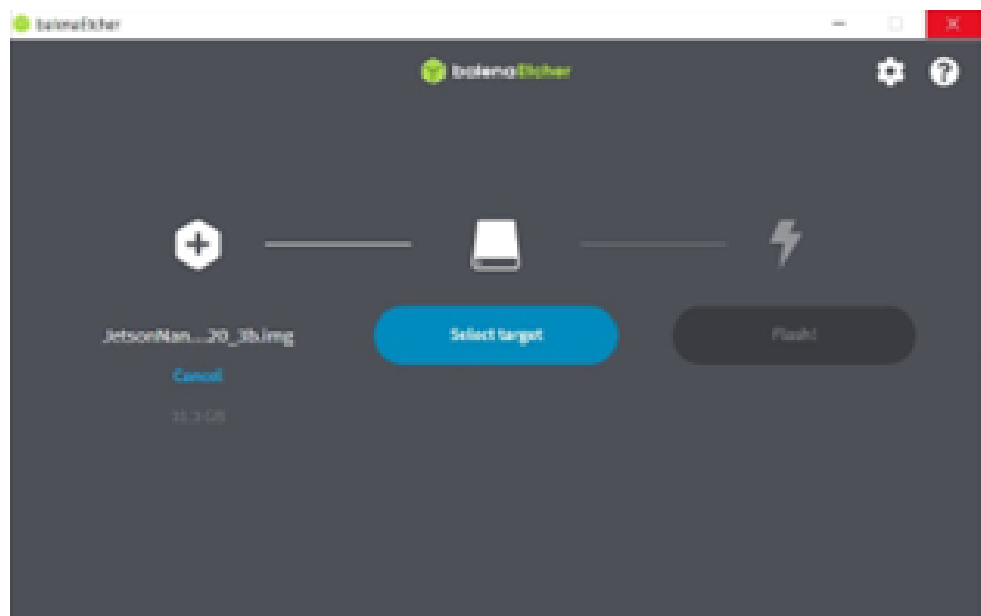
לאחר שהתקנו את התוכנה על המחשב האישי שלנו, נפתח את התוכנה . נבחר לצרוב את
ה-img שהורדנו קודם.



נבחר את ה-Img שהורדנו:



נבחר לאן נצרוּב:



נבחר בכרטיס זיכרון שלנו ונאשר.

הפעולה עלולה לקחת מספר דקות (בערך 20 דקות), ולאחר מכן ה-ubuntu 18.04 יהיה צרוב על הכרטיס זיכרון.

הפעלת ה - Jetson Nano

לאחר הצריבה של ה - ubuntu 18.04 נכניס את כרטיס הזיכרון ל- nano jetson .
(לאחר התקנת ה- ubuntu 18.04 לא עושים לו עדכוני גרסה או דברים כאלה, גם אם רשום
באינטרנט במדריכים שונים)

התקנת ROS

יש להיכנס לקישור (על ידי שם המשתמש והסיסמה של המעבדה):

<https://www.udemy.com/course/ros2-for-beginners/learn/lecture/33147228?start=165#overview>

ולעקוב אחרי ההוראות שיש בסרטון

בקישור הבא יש הסבר על איך להתקין את Ros2 eloquent. צריך לעקוב אחרי ההוראות
באתר ולרשום את שורות הקוד בטרמינל בהתאם למה שמופיע באתר :

<https://docs.ros.org/en/eloquent/Installation/Linux-Install-Debians.html>

Set locale

Make sure you have a locale which supports `UTF-8`. If you are in a minimal environment (such as a docker container), the locale may be something minimal like `POSIX`. We test with the following settings. However, it should be fine if you're using a different UTF-8 supported locale.

```
locale # check for UTF-8

sudo apt update && sudo apt install locales
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8

locale # verify settings
```

Setup Sources

You will need to add the ROS 2 apt repositories to your system. To do so, first authorize our GPG key with apt like this:

```
sudo apt update && sudo apt install curl gnupg2 lsb-release
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
```

And then add the repository to your sources list:

```
sudo sh -c 'echo "deb [arch=$(dpkg --print-architecture)] http://packages.ros.org/ros2/ubuntu $(lsb_rel
```

****נשים לב כי נרצה לבחור את Bare bones:**

Install ROS 2 packages

Update your apt repository caches after setting up the repositories.

```
sudo apt update
```

Desktop Install (Recommended): ROS, RViz, demos, tutorials.

```
sudo apt install ros-eloquent-desktop
```

ROS-Base Install (Bare Bones): Communication libraries, message packages, command line tools. No GUI tools.

```
sudo apt install ros-eloquent-ros-base
```

Environment setup

Sourcing the setup script

Set up your environment by sourcing the following file.

```
source /opt/ros/eloquent/setup.bash
```


הורדת vscode

על מנת להוריד vscode לגטסון ננו גרסת 18 ubuntu נרשום בטרמינל את הפקודות הבאות:

```
wget https://update.code.visualstudio.com/1.60.0/linux-deb-arm64/stable -O code_1.60.0-1630458161_arm64.deb
```

```
sudo dpkg -i code_1.60.0-1630458161_arm64.deb
```

```
sudo apt-get install -f
```

לאחר ההתקנה נוריד דרך ה VSCODE עצמו את התוכנות הבאות:

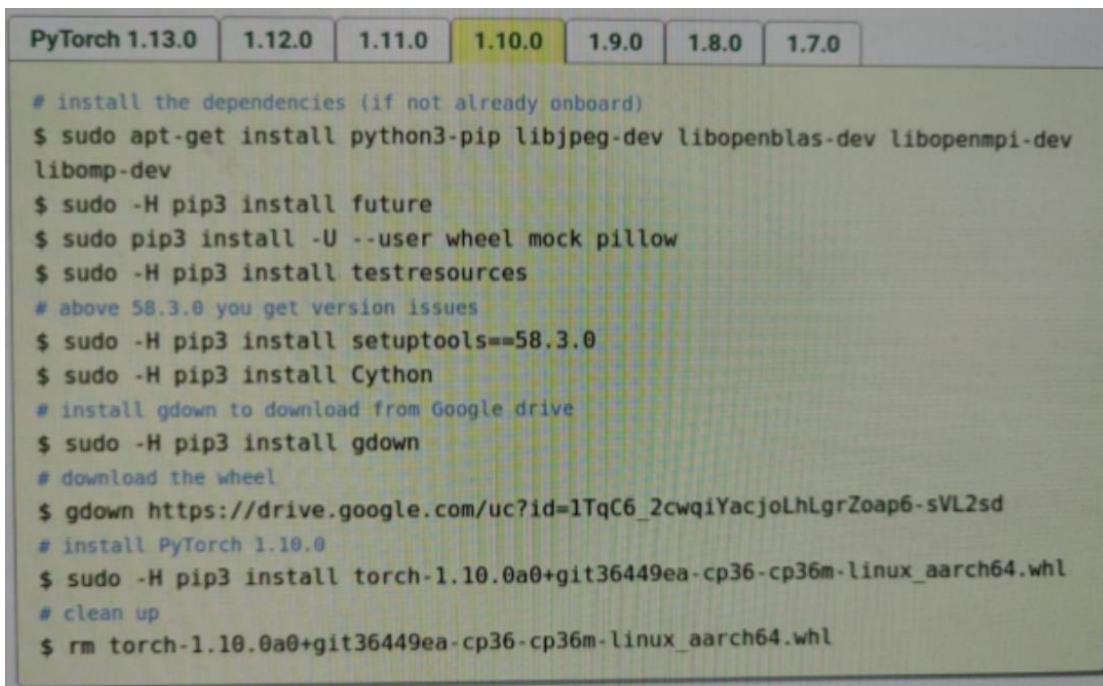
- Remote ssh
- Python
- ++c/c
- cmake
- jupyter

הורדת pytorch

ניתן להוריד מן האתר:

<https://qengineering.eu/install-pytorch-on-jetson-nano.html>

בשביל הגטסון ננו גרסת 18 ubuntu נוריד את הגרסה 1.10.0 של pytorch על ידי הפקודות האלה :



```
PyTorch 1.13.0 1.12.0 1.11.0 1.10.0 1.9.0 1.8.0 1.7.0

# install the dependencies (if not already onboard)
$ sudo apt-get install python3-pip libjpeg-dev libopenblas-dev libopenmpi-dev libomp-dev
$ sudo -H pip3 install future
$ sudo pip3 install -U --user wheel mock pillow
$ sudo -H pip3 install testresources
# above 58.3.0 you get version issues
$ sudo -H pip3 install setuptools==58.3.0
$ sudo -H pip3 install Cython
# install gdown to download from Google drive
$ sudo -H pip3 install gdown
# download the wheel
$ gdown https://drive.google.com/uc?id=1TqC6_2cwqiYacjoLhLgrZoap6-sVL2sd
# install PyTorch 1.10.0
$ sudo -H pip3 install torch-1.10.0a0+git36449ea-cp36-cp36m-linux_aarch64.whl
# clean up
$ rm torch-1.10.0a0+git36449ea-cp36-cp36m-linux_aarch64.whl
```

הפעלת cuda:

ראשית יש לוודא שעובדים עם ubuntu 18.04.
לאחר מכן נבצע את הפקודות הבאות:

```
sudo apt-get update
```

```
sudo apt-get install nano
```

```
nano ~/.bashrc
```

```
export PATH=/usr/local/cuda/bin${PATH:+:${PATH}}
```

```
export
```

```
LD_LIBRARY_PATH=/usr/local/cuda/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

```
source ~/.bashrc
```

Justify To Spell C Cur Pos Go To Line

install nano?

PT

to" is not installed on your system, you can install it using the package manager of your Linux distribution. Since you're using a Jetson Nano, which typically runs Ubuntu-based distributions, you can install "nano" using the "apt" package manager. Here's how:

```
bash
sudo apt-get update
sudo apt-get install nano
```

This will update the package lists and then install "nano" on your system. After the installation is complete, you can use "nano" to edit files, including the ".bashrc" file as you intended.

Once installed, you can then proceed to edit the ".bashrc" file by running:

```
bash
nano ~/.bashrc
```

Follow these steps:

vim:

10.2.300

1. Press "Ctrl+O" to write to file.

2. Press "i" to enter insert mode.

3. Add the following lines at the end of the file:

```
bash
export PATH=/usr/local/cuda/bin${PATH:+:${PATH}}
export LD_LIBRARY_PATH=/usr/local/cuda/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

4. Press "Esc" to exit insert mode.

5. Type ":wq" and press "Enter" to save and exit "vim".

6. Source the updated ".bashrc" file to apply the changes:

```
bash
source ~/.bashrc
```

Verify CUDA Toolkit

Message ChatGPT

ChatGPT can make mistakes. Check important info.

DELL

This will update the package lists and then install `nano` on your system. After the installation is complete, you can use `nano` to edit files, including the `.bashrc` file as you intended.

Once installed, you can then proceed to edit the `.bashrc` file by running:

2. Press `~i` to enter insert mode.
3. Add the following lines at the end of the file:

4. Press `Esc` to exit insert mode.
5. Type `:wq` and press `Enter` to save and exit `vim`.
6. Source the updated `.bashrc` file to apply the changes:

5. Type `:wq` and press `Enter` to save and exit `vim`.
6. Source the updated `.bashrc` file to apply the changes:

Verify CUDA Toolkit

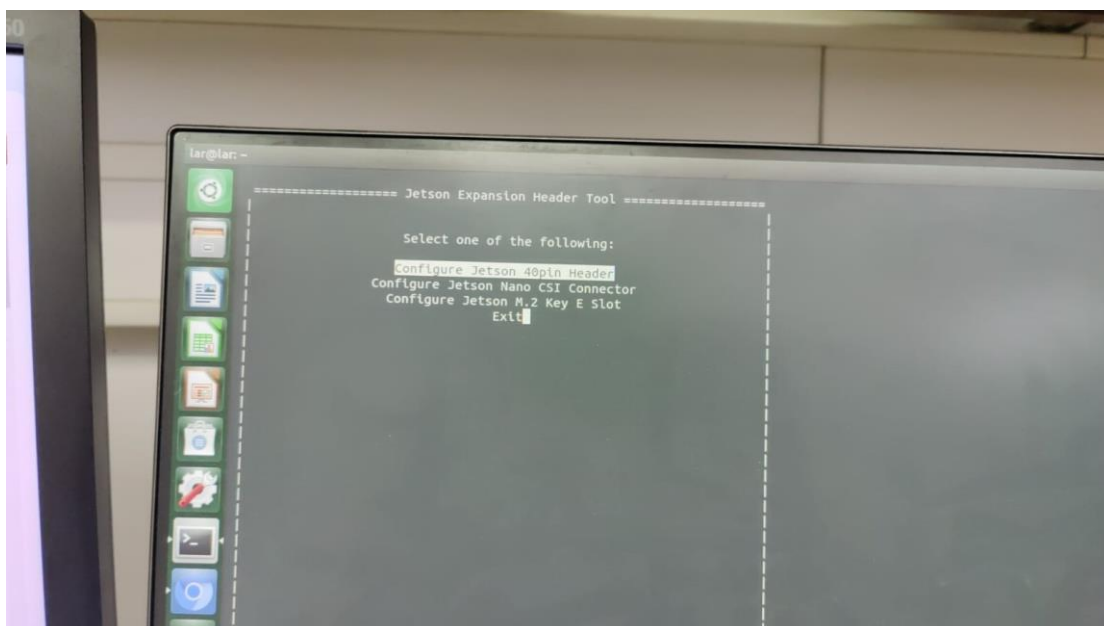
 Message ChatGPT

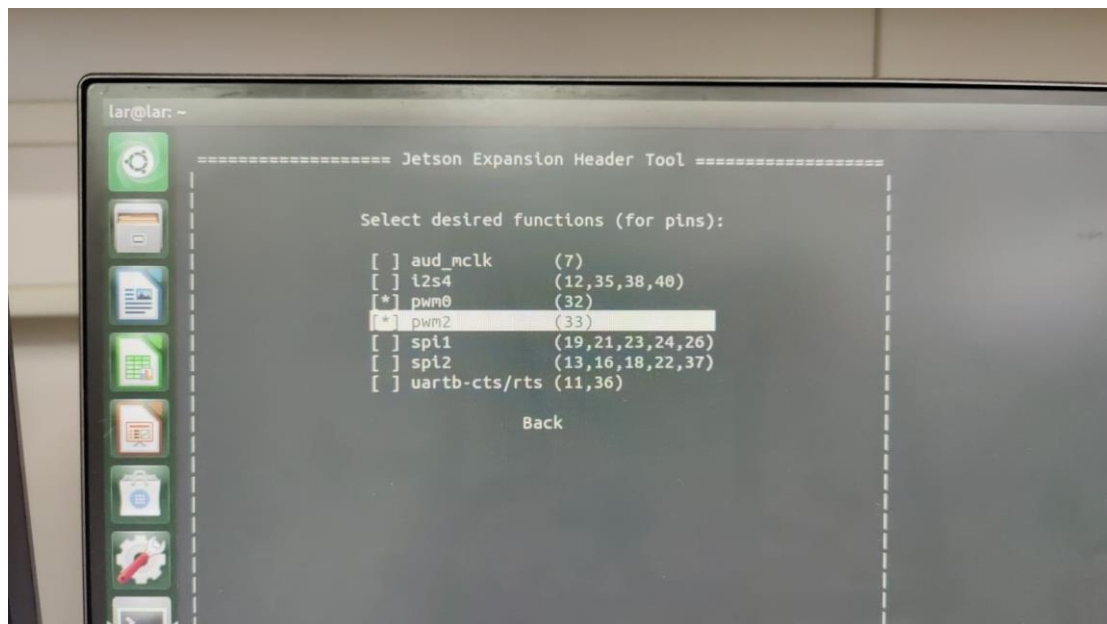
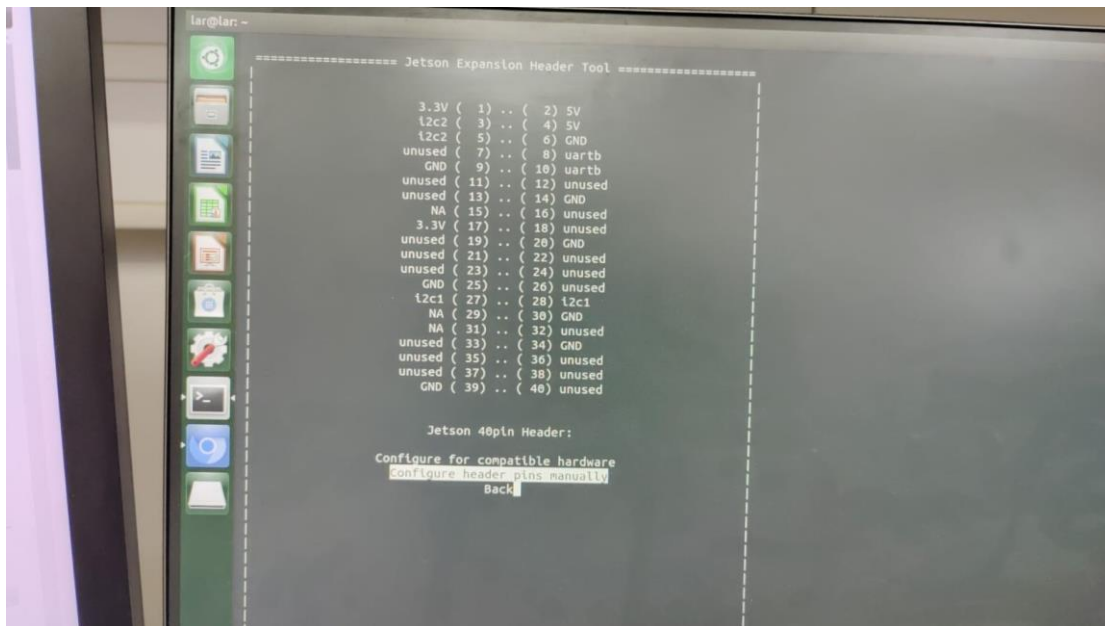
ChatGPT can make mistakes. Check important info.

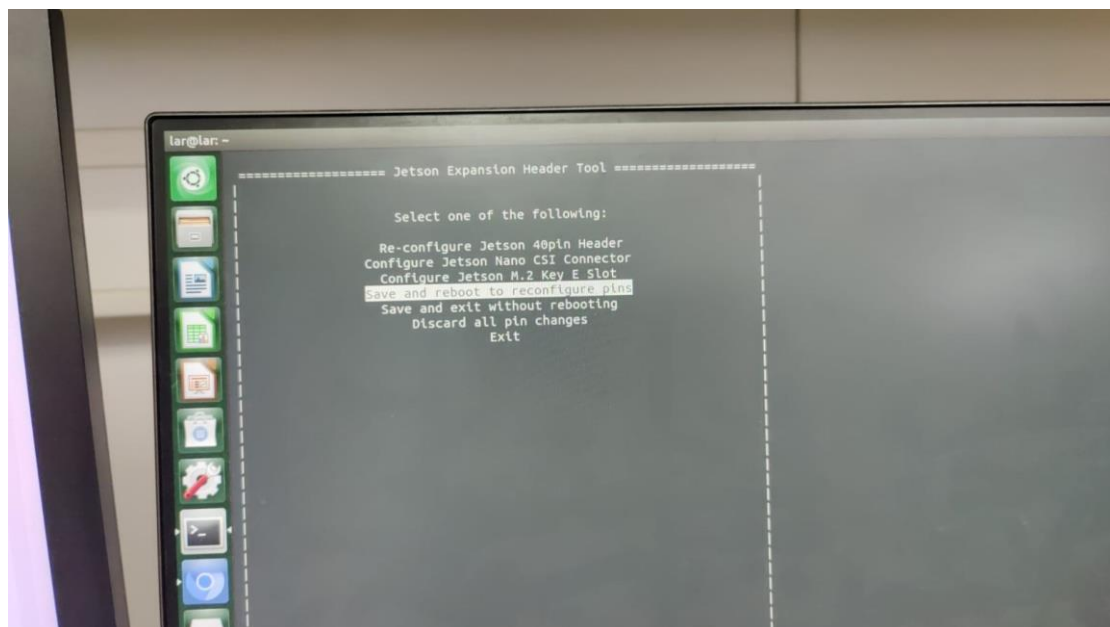
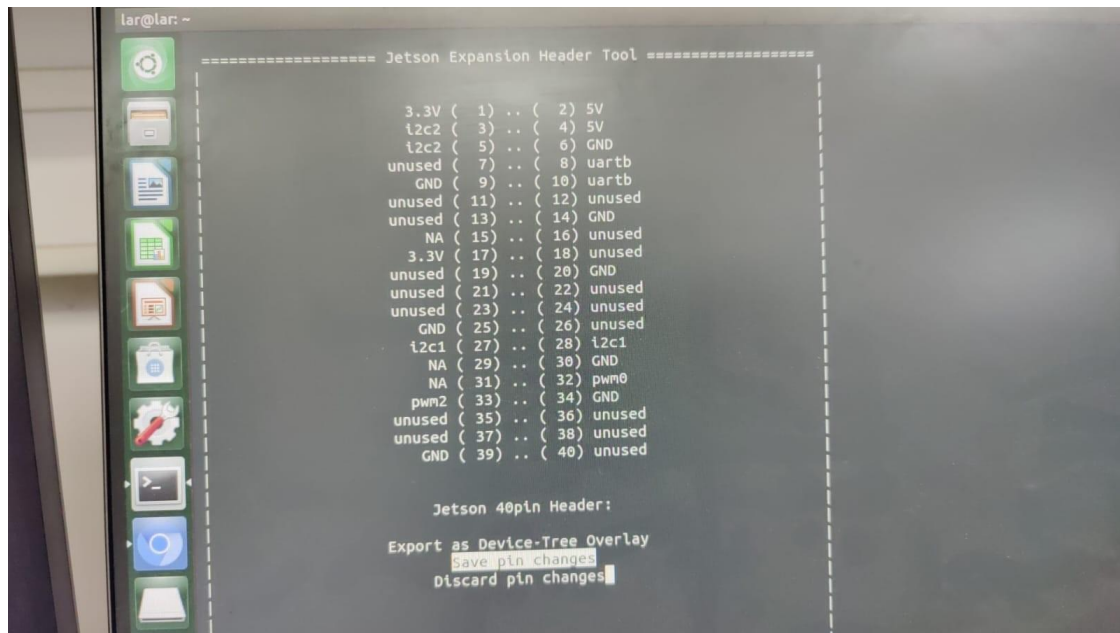
קינפוג פינים:

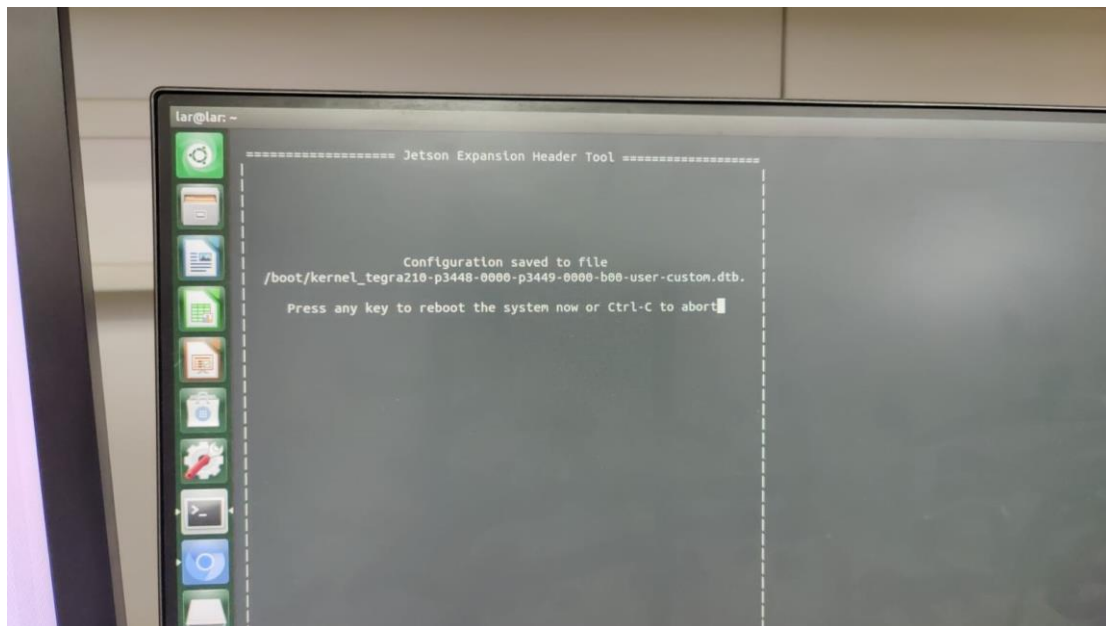
על מנת לקנפג את הפינים (למשל אפשור PWM):

```
lar@lar: ~  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
lar@lar:~$ nvcc --version  
bash: nvcc: command not found  
lar@lar:~$ sudo /opt/nvidia/jetson-io/jetson-io.py  
[sudo] password for lar:  
lar@lar:~$ sudo /opt/nvidia/jetson-io/jetson-io.py  
lar@lar:~$ sudo /opt/nvidia/jetson-io/jetson-io.py  
lar@lar:~$
```



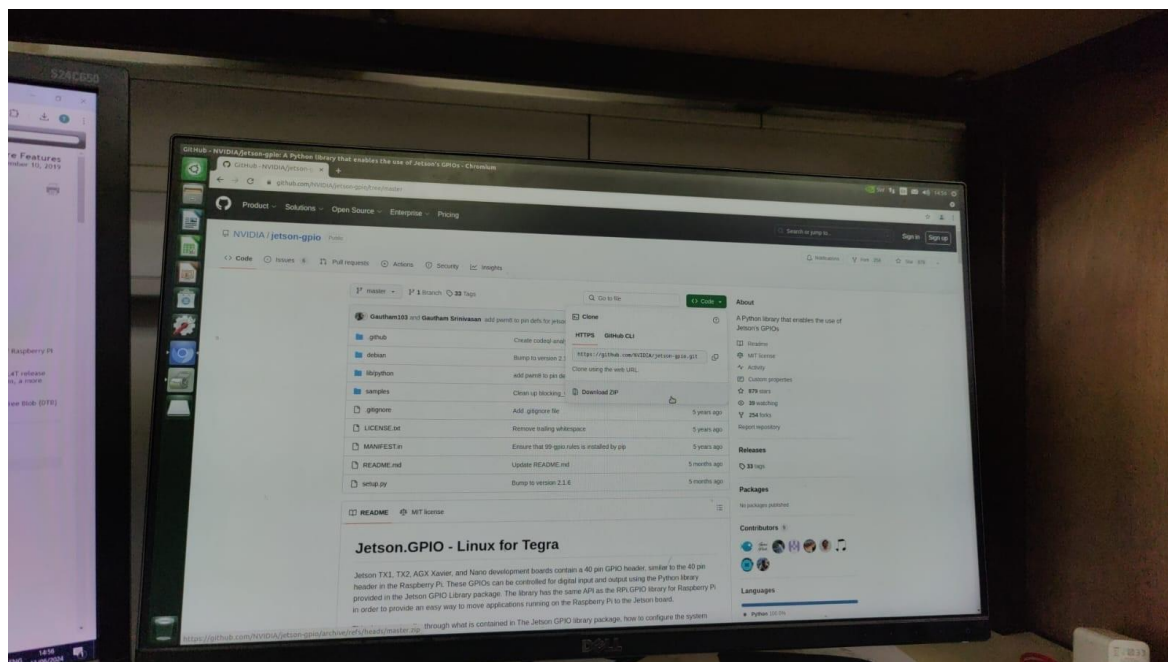






בשביל הרצת קובץ דוגמא להפעלת PWM ניגש לאתר:

<https://github.com/NVIDIA/jetson-gpio>



שינוי מיקום הזיכרון לכרטיס הזיכרון:

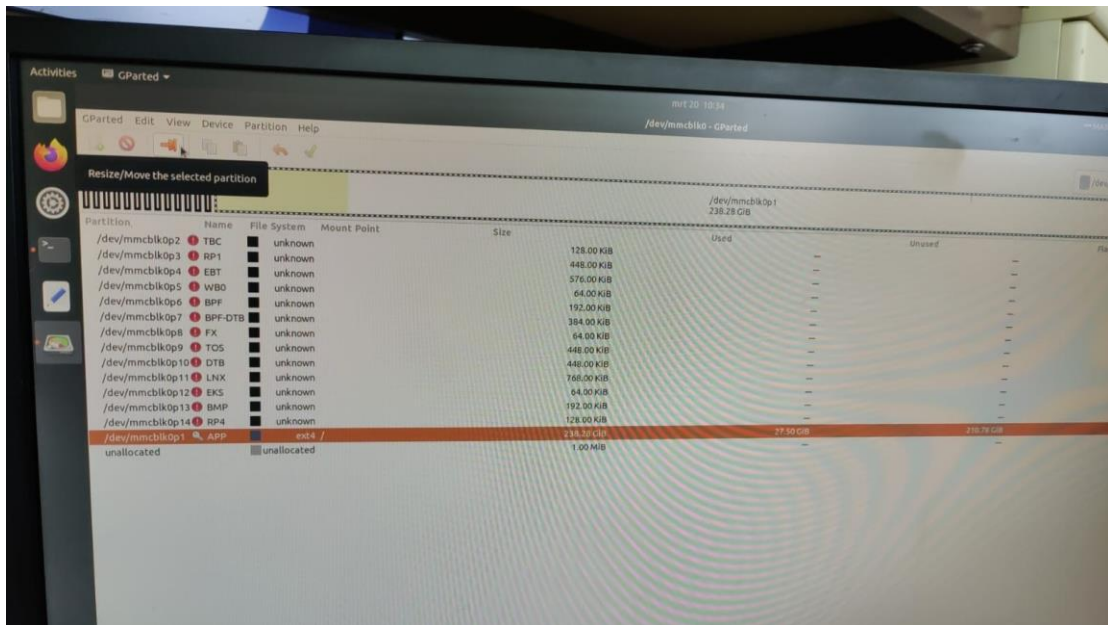
יש להוריד תוכנה שנקראת " gparted ".
יש לפתוח את הטרימינל ב-jetson ולרשום:

```
apt update sudo  
gparted apt install sudo
```

בשורות אלו בעצם ביצענו את ההורדה והתקנה של התוכנה שלנו.
לאחר מכן נפתח אותה על ידי:

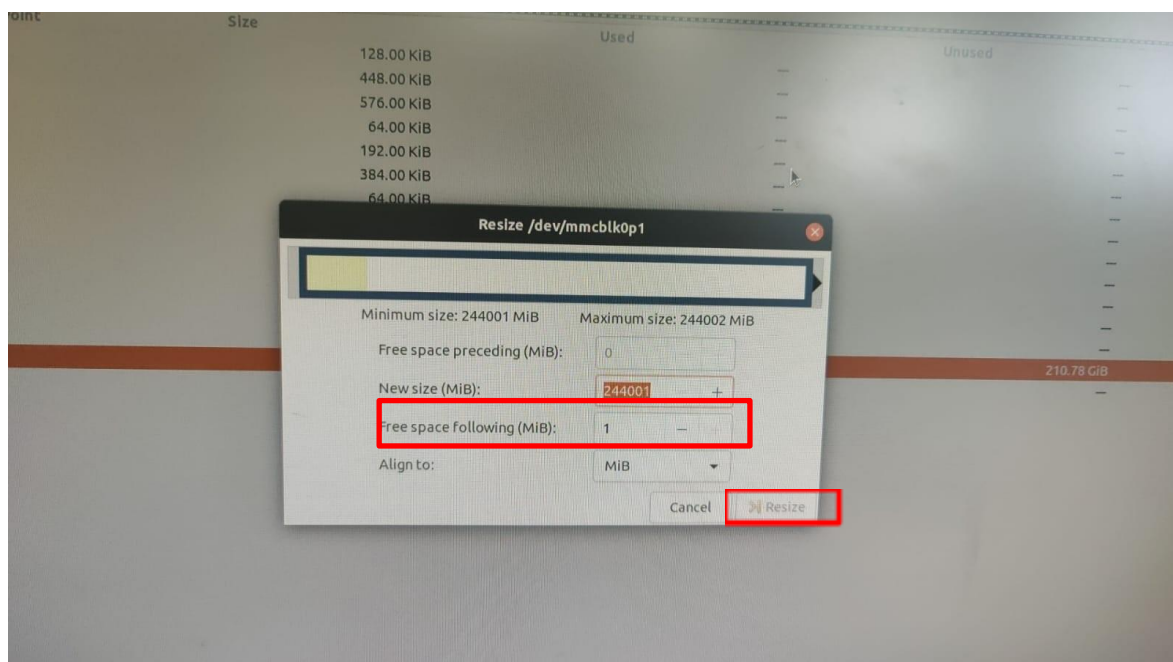
```
sudo gparted
```

התוכנה תיפתח ותראה ככה :



נלחץ על השורה שבה מופיע השדה /ext4 תחת השדה file system.

אחר מכן נלחץ על הכפתור עם החץ הכתום והקו. יפתח החלון הבא:



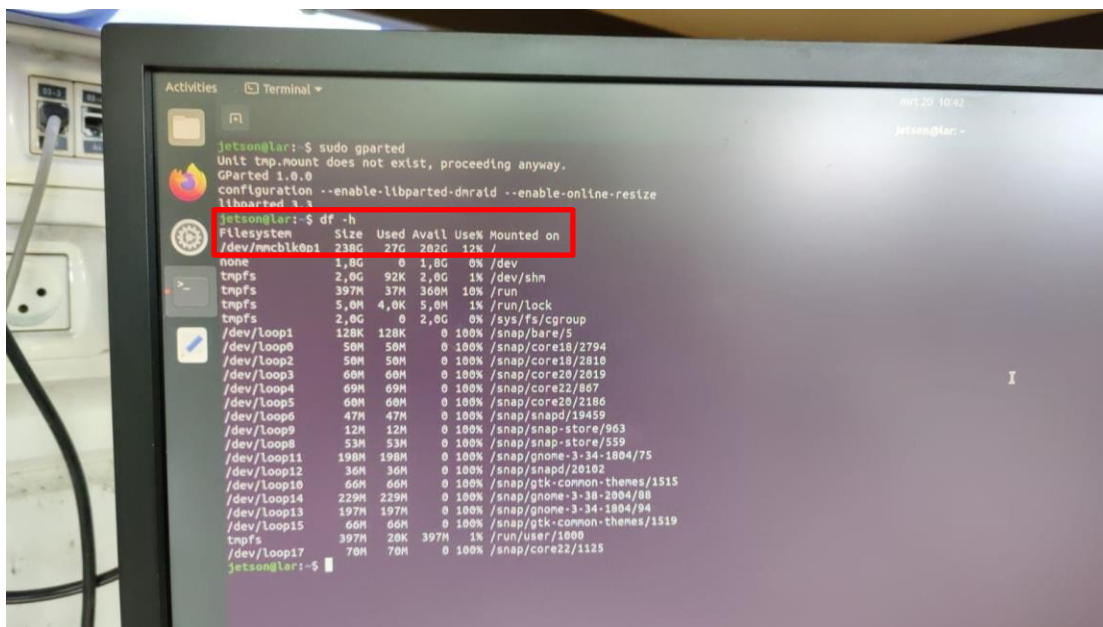
בשדה של free space following (MiB) נשנה לאפס (או כנראה כברירת מחדל הוא ישנה לאחד).

נלחץ .resize.

נסגור את כל החלונות, נוודא שאכן השתנה נפח האחסון על ידי הפקודה:

df -h

נצפה לראות משהו כזה:



```
jetson@lar:~$ sudo gparted
Unit tmp.mount does not exist, proceeding anyway.
GParted 1.0.0
configuration --enable-libparted-dmraid --enable-online-resize
libparted 1.4
jetson@lar:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mmcblk0p1 238G  27G  202G   12% /
none            1.0G  0  1.0G   0% /dev
tmpfs           2.0G  92K  2.0G   1% /dev/shm
tmpfs           397M  37M  360M  10% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           2.0G  0  2.0G   0% /sys/fs/cgroup
/dev/loop1      128K  128K  0  100% /snap/bare/5
/dev/loop0       50M   50M  0  100% /snap/core18/2794
/dev/loop2       50M   50M  0  100% /snap/core18/2810
/dev/loop3       60M   60M  0  100% /snap/core20/2019
/dev/loop4       60M   60M  0  100% /snap/core22/867
/dev/loop5       60M   60M  0  100% /snap/core20/2186
/dev/loop6       47M   47M  0  100% /snap/snapd/19459
/dev/loop9       12M   12M  0  100% /snap/snap-store/963
/dev/loop8       53M   53M  0  100% /snap/snap-store/559
/dev/loop11      198M  198M  0  100% /snap/gnome-3-34-1804/75
/dev/loop12       36M   36M  0  100% /snap/snapd/20102
/dev/loop10      66M   66M  0  100% /snap/gtk-common-themes/1515
/dev/loop14      229M  229M  0  100% /snap/gnome-3-38-2004/88
/dev/loop13      197M  197M  0  100% /snap/gnome-3-34-1804/94
/dev/loop15      66M   66M  0  100% /snap/gtk-common-themes/1519
tmpfs           397M  20K  397M   1% /run/user/1000
/dev/loop17      70M   70M  0  100% /snap/core22/1125
jetson@lar:~$
```

ולאחר מכן נעשה restart לjetson:

sudo reboot

כדי לשנות את התיקיות על הjetsonn ככה שיהיה אפשר להעתיק אליהן תוכן:

יש להגיע צעד אחד/ תיקיה אחת לפני התיקיה שאליה רוצים להגיע, כלומר הכוונה: אם רוצים את התיקיה ir0, יש לרשום:

calibration/eyerop/cd /opt

לאחר מכן יש להזין את הפקודות הבאות על מנת לשנות את ההרשאות של התיקייה:

```
ir0 u+w chmod sudo  
ir0 g+w chmod sudo  
ir0 o+w chmod sudo
```

כדי לוודא שמה שעשינו נכון נרשום (בתיקייה בה אנחנו מעוניינים):

ls

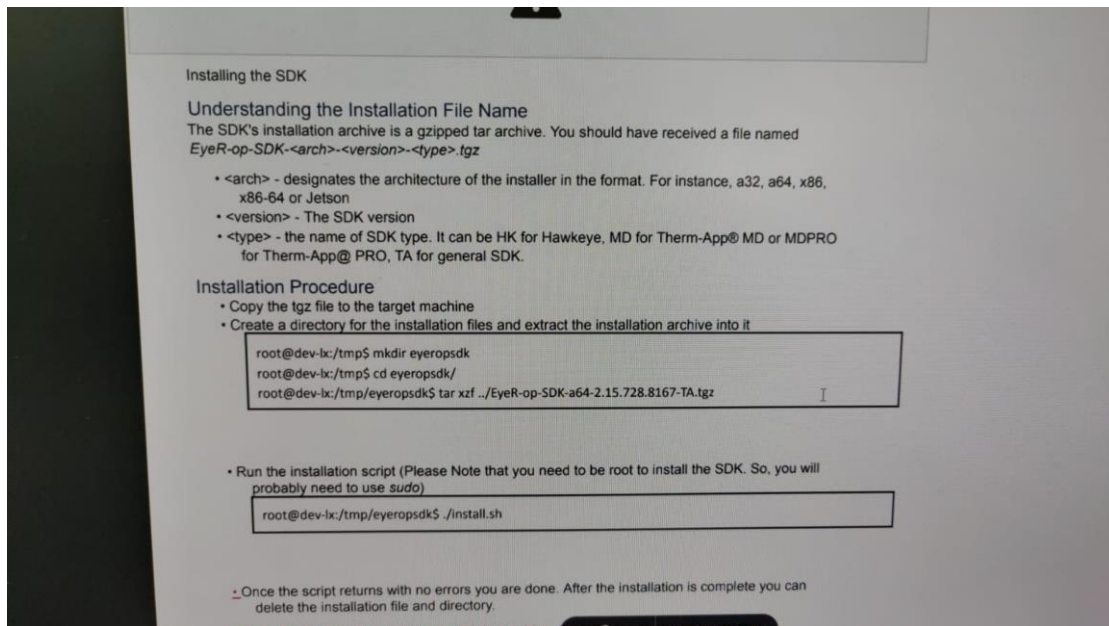
ls -l

נצפה לראות שרשום משהו מהצורה:

```
eyerop Mar 20 10:00 4096 root root 3 drwxrwxrwx
```

הפעלת המצלמה דרך jetson:

ראשית יש לוודא כי נמצאים ברשותינו קבצי הכיול המתאימים למצלמה וקבצי התקנה מתאימים לסוג המעבד שרוצים להשתמש בו.
במקרה שלנו (jetson nano) נוודא שקובץ ההתקנה מכיל את המילים a64 (גרסת המעבד של jetson).
לאחר מכן נפעל על פי ההוראות שרשומות תחת Installation procedure (ב- datasheet של המצלמה – עמוד 10):



בטרמינל נגיע לתיקייה tmp.
ניצור תיקייה ששמה eyeropsdk
ניכנס לתיקייה.
נשים את קובץ ההתקנה שם.
נחלץ על ידי הפקודה הנתונה.

לבסוף נריץ את קובץ install.
יש לשים לב כי ייתכן שנצטרך להשתמש ב-sudo לפני חלק מהפקודות.
לאחר מכן יש להעתיק את הקבצי הכיול לתיקייה המתאימה כפי שרשום תחת manual installation.
פקודה להרצת המצלמה:

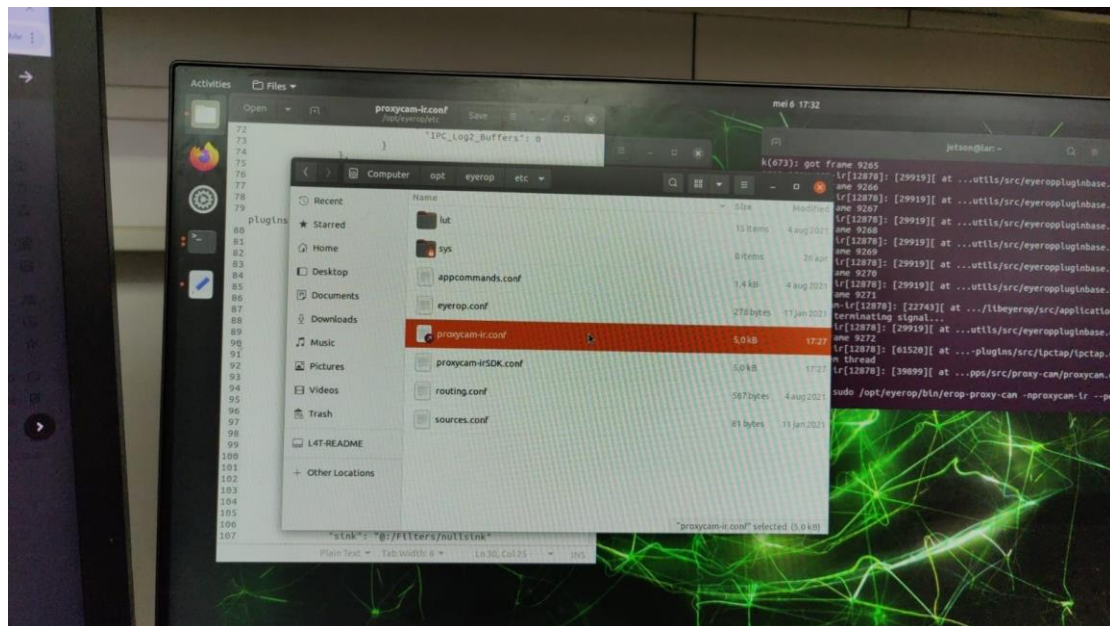
001=logmask-- perror-- nproxycam-ir- proxy-cam-erop/bin/eyerop/Sudo /opt

או

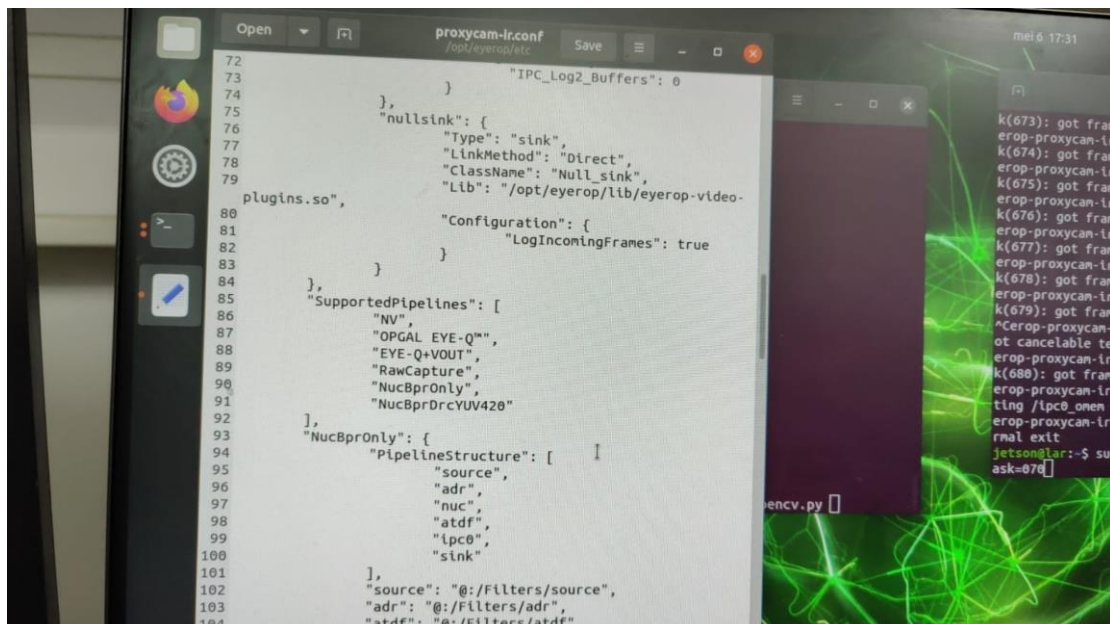
070=logmask-- perror-- nproxycam-ir- proxy-cam-erop/bin/eyerop/Sudo /opt

על מנת לבדוק מה התמונה שהמצלמה קולטת ניתן להריץ את קובץ ה-capture_opencv.py.

על מנת שהתמונה של המצלמה תהיה מכוילת נכון, ניגש ל: `proxycam-/etc/eyerop/opt` `ir.conf`.



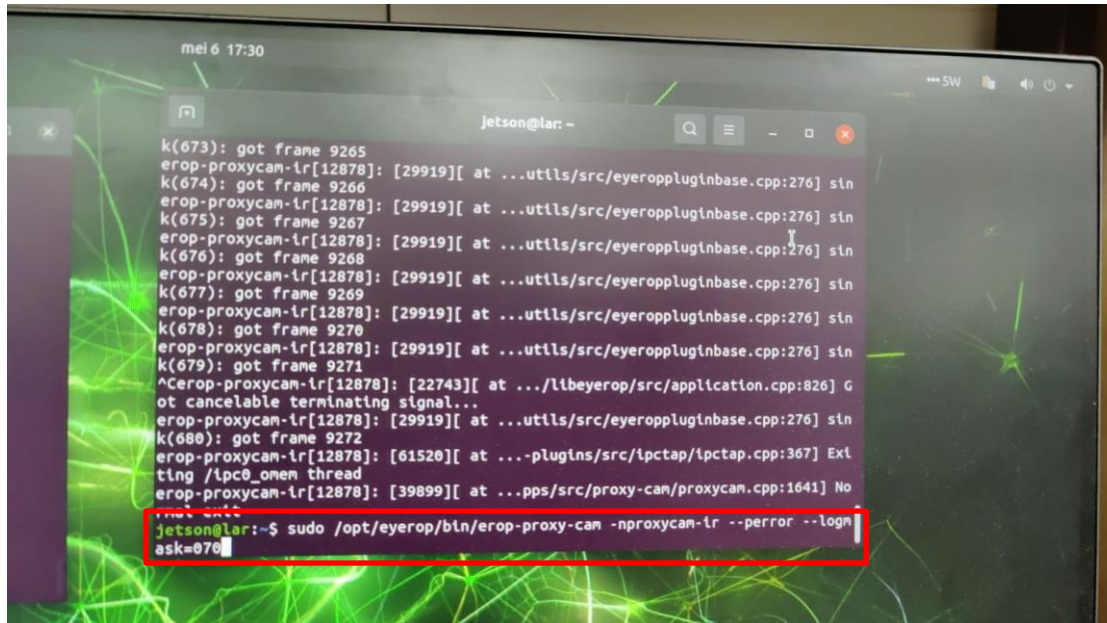
בקבוצ שיפתח תחת "ActivePipeline". נשנה ל:
`"OPGAL EYE-QTMOPGAL EYE-QTM"`
 (נשים לב שבמעבדים שונים יש צורך בהגדרות אחרות, לדוגמא במעבד UP נרשום בשדה זה "NV").



לאחר מכן נקבל תמונה מכוילת כראוי.

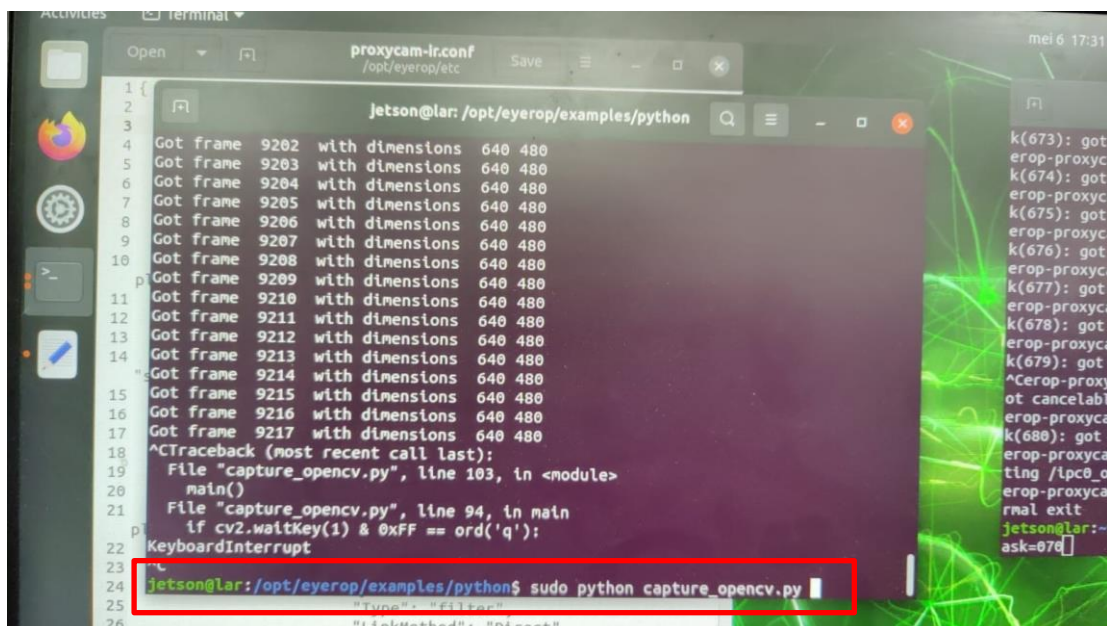
דוגמא להרצה:

בחלון טרמינל נרשום את הפקודה להפעלת המצלמה (capture):



```
mei 6 17:30
jetson@lar: ~
k(673): got frame 9265
erop-proxycam-ir[12878]: [29919][ at ...utils/src/eyeroppluginbase.cpp:276] sin
k(674): got frame 9266
erop-proxycam-ir[12878]: [29919][ at ...utils/src/eyeroppluginbase.cpp:276] sin
k(675): got frame 9267
erop-proxycam-ir[12878]: [29919][ at ...utils/src/eyeroppluginbase.cpp:276] sin
k(676): got frame 9268
erop-proxycam-ir[12878]: [29919][ at ...utils/src/eyeroppluginbase.cpp:276] sin
k(677): got frame 9269
erop-proxycam-ir[12878]: [29919][ at ...utils/src/eyeroppluginbase.cpp:276] sin
k(678): got frame 9270
erop-proxycam-ir[12878]: [29919][ at ...utils/src/eyeroppluginbase.cpp:276] sin
k(679): got frame 9271
^Cerop-proxycam-ir[12878]: [22743][ at ...libeyerop/src/application.cpp:826] G
ot cancelable terminating signal...
erop-proxycam-ir[12878]: [29919][ at ...utils/src/eyeroppluginbase.cpp:276] sin
k(680): got frame 9272
erop-proxycam-ir[12878]: [61520][ at ...plugins/src/lpctap/lpctap.cpp:367] Ext
ing /ipc0_omen thread
erop-proxycam-ir[12878]: [39899][ at ...pps/src/proxy-cam/proxycam.cpp:1641] No
rmal exit
jetson@lar:~$ sudo /opt/eyerop/bin/erop-proxycam -nproxycam-ir --perror --logn
ask=070
```

בטרמינל נוסף, נריץ את הפקודה:



```
Activities
Terminal
Open
proxycam-ir.conf
/opt/eyerop/etc
Save
Jetson@lar: /opt/eyerop/examples/python
1 {
2
3
4 Got frame 9202 with dimensions 640 480
5 Got frame 9203 with dimensions 640 480
6 Got frame 9204 with dimensions 640 480
7 Got frame 9205 with dimensions 640 480
8 Got frame 9206 with dimensions 640 480
9 Got frame 9207 with dimensions 640 480
10 Got frame 9208 with dimensions 640 480
11 Got frame 9209 with dimensions 640 480
12 Got frame 9210 with dimensions 640 480
13 Got frame 9211 with dimensions 640 480
14 Got frame 9212 with dimensions 640 480
15 Got frame 9213 with dimensions 640 480
16 Got frame 9214 with dimensions 640 480
17 Got frame 9215 with dimensions 640 480
18 Got frame 9216 with dimensions 640 480
19 Got frame 9217 with dimensions 640 480
20 ^CTraceback (most recent call last):
21   File "capture_opencv.py", line 103, in <module>
22     main()
23   File "capture_opencv.py", line 94, in main
24     if cv2.waitKey(1) & 0xFF == ord('q'):
25     KeyboardInterrupt
26
jetson@lar:/opt/eyerop/examples/python$ sudo python capture_opencv.py
```

קיימים מספר קבצי פייתון אשר ניתן להריץ על מנת לקבל וידאו של מה שהמצלמה משדרת:

Capture_opencv - קובץ פייתון מובנה של opgal מגיע ביחד עם ההתקנה של הקבצים.

Main.py – קובץ פייתון שמבצע עיבוד תמונה תוך שימוש בgpu.

Main_thread.py - קובץ פייתון שמבצע עיבוד תמונה תוך שימוש בgpu וגם ב-threads=2.

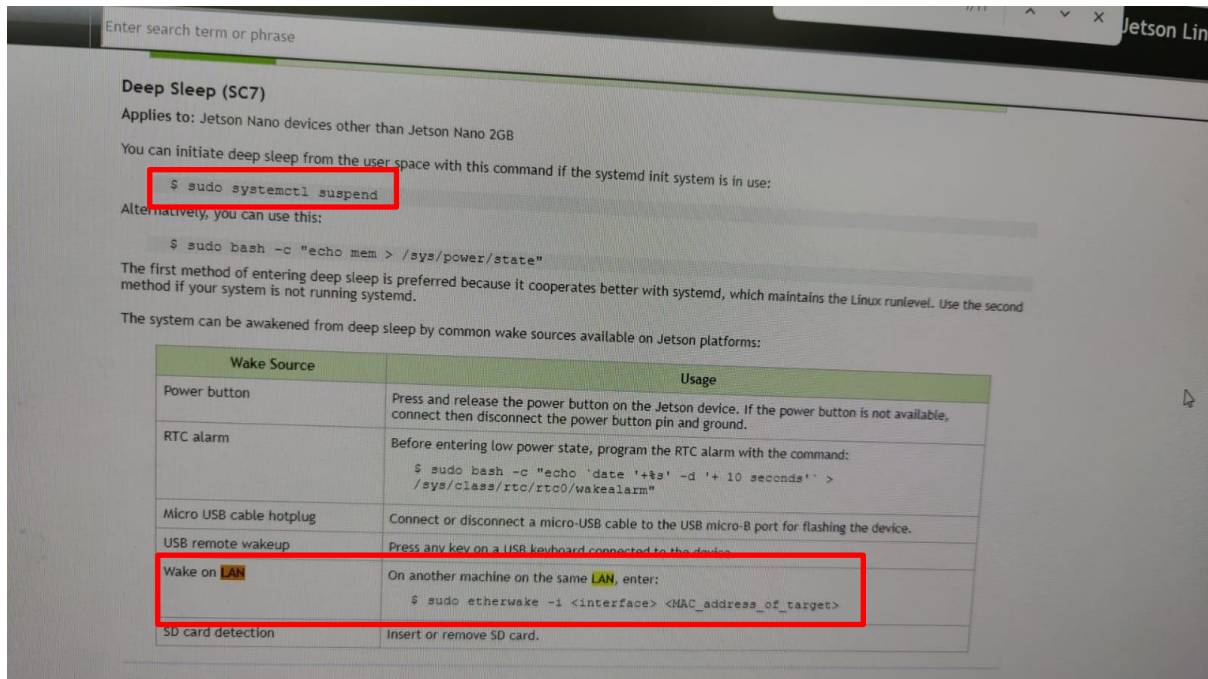
Main_thread_comp.py - קובץ פייתון שמבצע עיבוד תמונה תוך שימוש בgpu וגם ב-threads=2 וגם מבצע דחיסה ושולח את התמונה (נרצה להשתמש בקוד זה בטייגר).

```
lar@lar:~$ cd /opt/eyerop
lar@lar:/opt/eyerop$ ls
bin calibration etc examples lib
lar@lar:/opt/eyerop$ cd examples/
lar@lar:/opt/eyerop/examples$ cd python/
lar@lar:/opt/eyerop/examples/python$ ls
capture_opencv.py  cuda_try.py  __pycache__  switchcolormap.py
capture.py          main.ov      pyeyerop.py  switchpipeline.py
compressed_stream.py  main_thread_comp.py  querysource.py  toggleblackhot.py
cpyeyerop3.so        main_thread.py  re_capture.py
cpyeyerop.so         opgal-pylib    servo_try_33.py
lar@lar:/opt/eyerop/examples/python$ cd ..
lar@lar:/opt/eyerop/examples$ cd ..
lar@lar:/opt/eyerop$ cd calibration/ir0/
lar@lar:/opt/eyerop/calibration/ir0$ cd ..
lar@lar:/opt/eyerop/calibration$ cd ..
lar@lar:/opt/eyerop$ cd etc
lar@lar:/opt/eyerop/etc$ ls
appcommands.conf  lut  proxycam-irSDK.conf  sources.conf
eyerop.conf        proxycam-ir.conf  routing.conf          sys
lar@lar:/opt/eyerop/etc$ cd
lar@lar:~$
```

Deep sleep

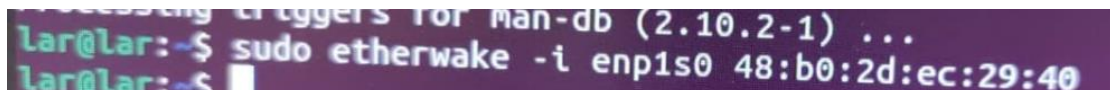
על מנת להכניס את המעבד למצב של שינה נרשום את הפקודה:

```
sudo systemctl suspend
```



בשביל להעיר את הגטסון נלחץ על מקש כלשהו במקלדת או לחילופין נרשום את הפקודה הבאה במעבד אחר שמתקשר עם הגטסון דרך ETHERNET:

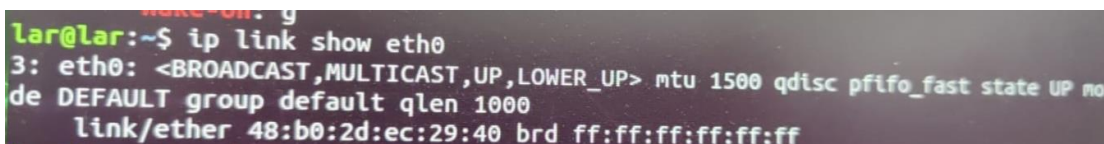
```
sudo etherwake -i enp1s0 48:b0:2d:ec:29:40
```



במעבד האחר יש לוודא כי מותקן ethtool ואם לא להריץ:

```
sudo apt install ethtool
```

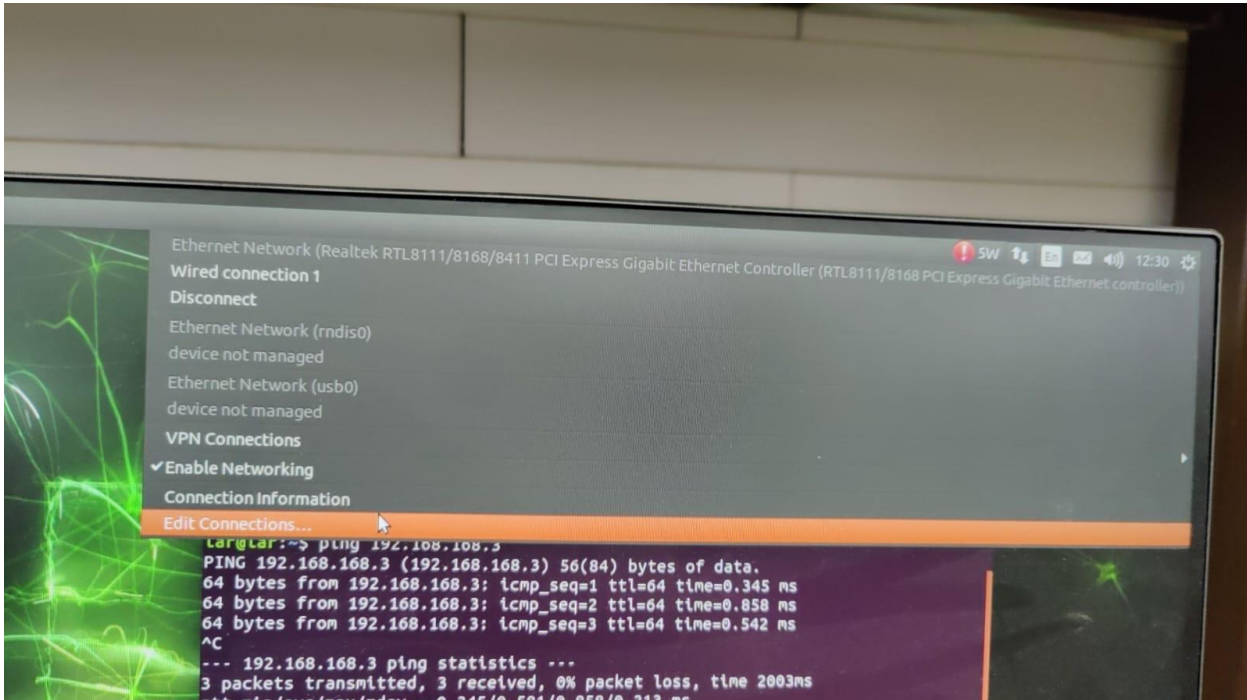
כתובת mac של הגטסון(ספציפי לכל גטסון):



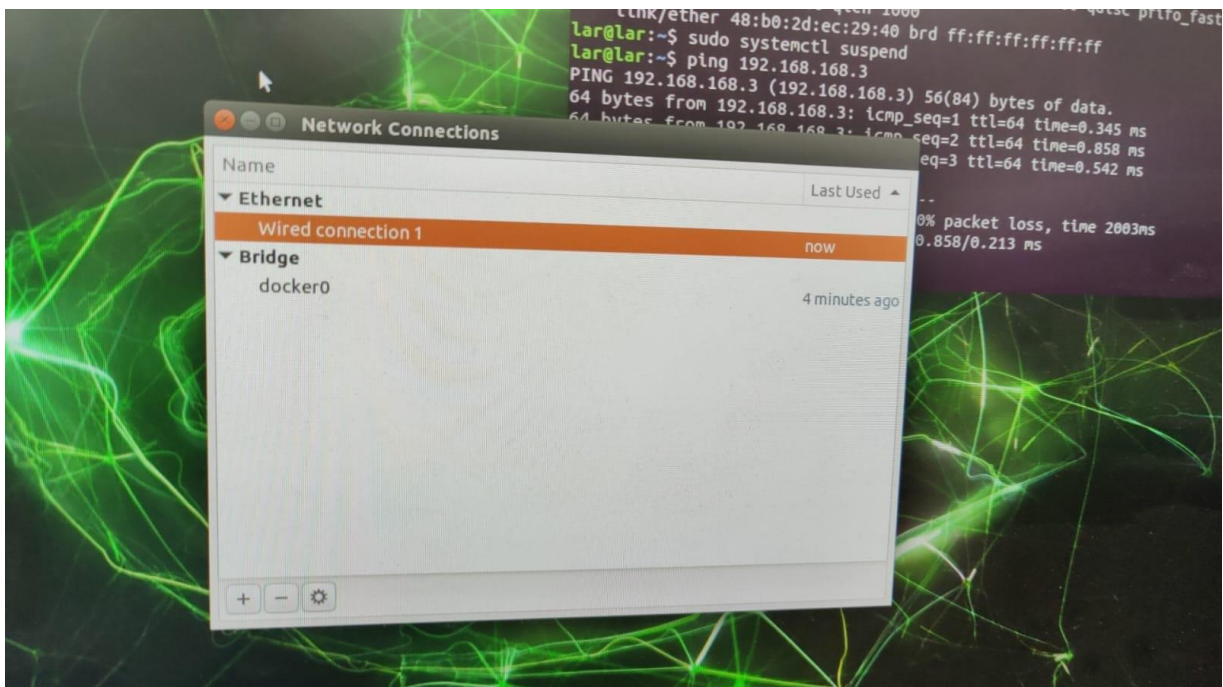
הגדרת ip

על מנת להגדיר את כתובת ה-ip של הגטסון.

נלחץ על הסמל של שתי החצים ונבחר edit connections:



לאחר מכן נבחר wired connection 1:



ונגדיר:

