Real time Road Space Rationing control using Jetson Nano

CHALLENGE : AI at the Edge Challenge with NVIDIA
AUTHOR : Cristian Lazo Quispe
EMAIL : clazoq@uni.pe

1. INSTALLATION OF ENVIRONMENT

1.1. Requirements

1. Jetson Nano Developed Kit
2. SD CARD 64 GB class 10
3. Good internet connection
4. Monitor
5. Keyboard
6. Mouse
7. Usb webcam

1.2. Download OS image:

It is necessary to use all the memory of the jetson nano, for this there are 2 alternatives to flash the operating system:

1. SDKMANAGER

   Problem with SDKMANAGER:
   According to the Tegra Linux Driver documentation in the 2.0 Known Issues part:
   https://docs.nvidia.com/jetson/archives/l4t-archived/l4t-321/pdf/Tegra_Linux_Driver_Package_Release_Notes_R32.1.pdf

   You can only burn an sd card with the size of 14 GB:

200488963	Nano `/dev/root` device size is restricted to 14 GB when using `flash.sh` to write to an SD card. User cannot create additional partitions on the root device, so any excess space is unavailable.

   That is why I have seen fit to use option 2.

2. Download image and flash with BalenaEtcher
   The available image https://developer.nvidia.com/embedded/downloads is used:

Be sure to use the one that comes with Opencv 3.* , because the plate recognition algorithm works with it.

username : dlinano
password : dlinano

By default, it consumes an average of 14 GB when it bounces the first time.

1.3. Preparing the system

Download the repository:

https://github.com/CristianLazoQuispe/Pico_y_placa.git

Save it in the documents folder

1.3.1. Test the image before you start

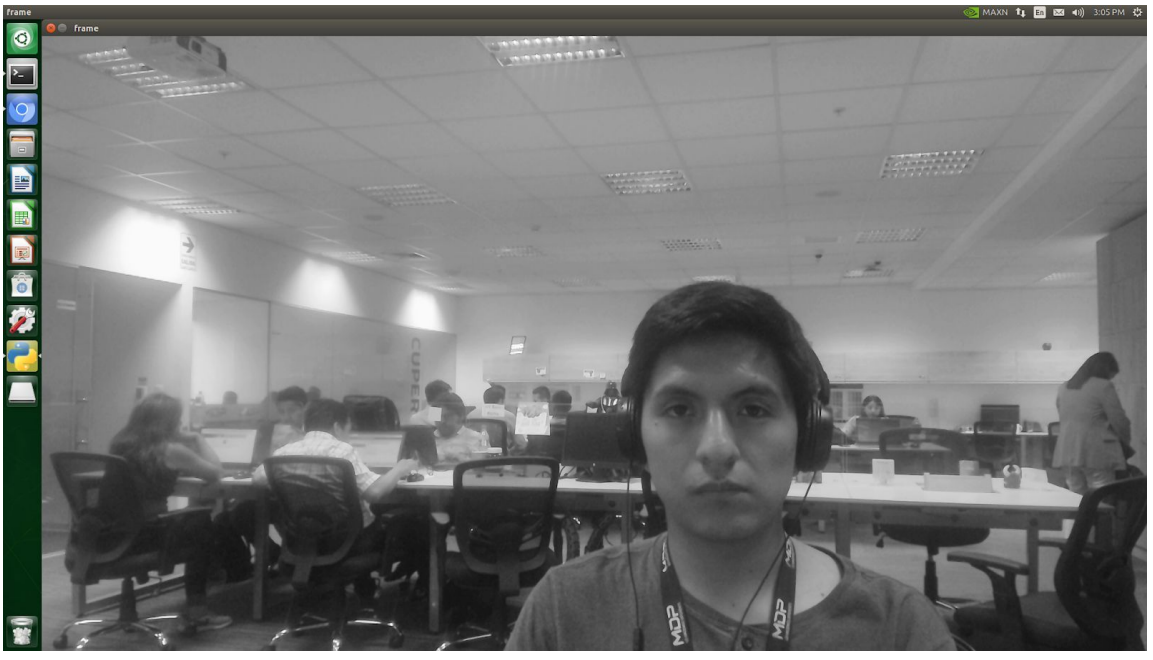
It is important to make sure that the image is working properly, to do this we must run the code read_camera_web.py:

Open terminal

Execute: https://github.com/CristianLazoQuispe/Pico_y_placa

python3 read_camera_web.py

If everything is correct you should see the camera reading in grayscale.



1.3.2. Install libraries

Installing the libraries is a very tedious job because of compatibility issues and installation time.

```
$ sudo apt-get update
$ sudo apt-get install -y python3-dev python3-numpy python3-py python3-pytest
$ sudo apt-get install git cmake
$ sudo apt-get install libatlas-base-dev gfortran
$ sudo apt-get install libhdf5-serial-dev hdf5-tools libhdf5-dev zlib1g-dev zip libjpeg8-dev
```

1.3.2.1. Install pip

```
$ sudo python3 -m pip uninstall pip setuptools wheel
$ sudo apt-get --reinstall install  python3-setuptools python3-wheel python3-pip
```

1.3.2.2. Install Tensorflow gpu

```
$ sudo pip3 install -U pip testresources setuptools
$ sudo pip3 install -U numpy==1.16.4 future==0.17.1 mock==3.0.5 h5py==2.9.0
keras_preprocessing==1.0.5 keras_applications==1.0.8 gast==0.2.2 enum34 futures protobuf
$ sudo pip3 install --extra-index-url
https://developer.download.nvidia.com/compute/redist/jp/v42
tensorflow-gpu==1.13.1+nv19.3
```

1.3.2.3. Install pycuda

First it is necessary to find the address of cuda in the system, you can find the path using the command:

```
$ find / -type d -name cuda 2>/dev/null
```

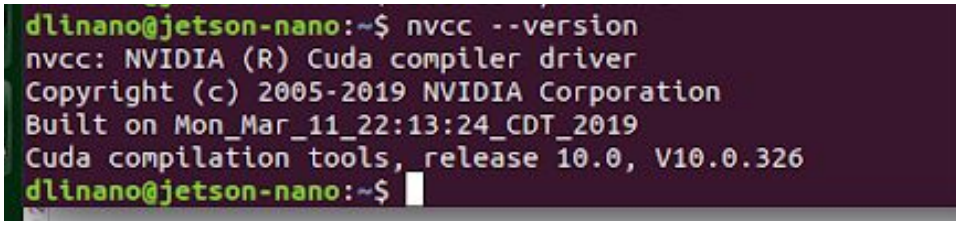
In my case:

```
/usr/local/cuda-10.0
```

Execute this commands:

```
$ export PATH=$PATH:/usr/local/cuda-10.0/bin

$ export CPATH=$CPATH:/usr/local/cuda-10.0/targets/aarch64-linux/include

$ export LIBRARY_PATH=$LIBRARY_PATH:/usr/local/cuda-10.0/targets/aarch64-linux/lib

$ source ~/.bashrc

$ nvcc --version
```

In my case:



Download the version of pycuda pycuda 2019.1.2 using the link below

https://pypi.org/project/pycuda/#files

Execute the following commands for the pycuda installation

```
$ cd pycuda-VERSION
$ python3 configure.py --cuda-root=/usr/local/cuda-10.0
$ sudo make install
```

1.3.2.4. Install complements

```
$ sudo apt-get install python3-scipy
$  sudo pip3 install imutils
$  sudo pip3 install keras
$  sudo apt-get install python3-matplotlib
$  sudo apt-get install libpcap-dev libpq-dev
$ sudo pip3 install cython
```

1.3.2.5. Install llvm

```
$ wget http://releases.llvm.org/7.0.1/llvm-7.0.1.src.tar.xz
$ tar -xvf llvm-7.0.1.src.tar.xz
$ cd llvm-7.0.1.src
$ mkdir llvm_build_dir
$ cd llvm_build_dir/
$ cmake ../ -DCMAKE_BUILD_TYPE=Release
-DLLVM_TARGETS_TO_BUILD="ARM;X86;AArch64"
$ make -j4
```

```
$ sudo make install

$ cd bin/

$ echo "export LLVM_CONFIG=\""`pwd`"/llvm-config\"" >> ~/.bashrc

$ echo "alias llvm='"`pwd`"/llvm-lit'" >> ~/.bashrc

$ source ~/.bashrc

$ sudo pip3 install llvmlite==0.30.0
```

1.3.2.6. Install numba

Before proceeding to the Numba installation, you need to perform the LLVM installation above first since Numba is heavily rely on LLVM installation.

```
$ sudo pip3 install numba==0.34.0
```

1.3.2.7. Install scikit learn

```
$ sudo apt-get install gfortran

$ sudo pip3 install https://github.com/scikit-learn/scikit-learn/archive/0.20.1.tar.gz
```

1.3.2.8. Install scikit image

```
$ sudo apt-get install python3-skimage

$ sudo pip3 install -U scikit-image
```

1.3.2.9. Install scipy

```
$ sudo apt remove python3-scipy

$ sudo apt install --reinstall python\*-decorator

$ sudo apt install -f

$ sudo apt install python3-scipy

$ sudo apt-get update
```

1.3.2.10. GPIO JETSON NANO

First you need to download this repository:

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

```
$ sudo python3 setup.py install
```

```
$ sudo groupadd -f -r gpio

$ sudo usermod -a -G gpio dlinano

$ sudo cp /opt/nvidia/jetson-gpio/etc/99-gpio.rules /etc/udev/rules.d/

$ sudo udevadm control --reload-rules && sudo udevadm trigger
```

1.3.2.11. LCD DISPLAY

```
$ sudo pip3 install RPLCD==1.2.2

$ sudo pip3 install filterpy
```

1.3.2.12. INSTALL OPENALPR

```
$ sudo add-apt-repository ppa:alex-p/tesseract-ocr
$ sudo apt-get update
$ wget http://www.leptonica.org/source/leptonica-1.78.0.tar.gz
$ tar xvf leptonica-1.78.0.tar.gz leptonica-1.78.0
$ cd leptonica-1.78.0
$ ./configure
$ make -j $(($(nproc)-1))
$ sudo make install
$ cd ~
$ git clone https://github.com/tesseract-ocr/tesseract.git
$ cd tesseract
$ ./autogen.sh
$ ./configure
$ make -j $(($(nproc)-1))
$ sudo make install
$ sudo ldconfig
$ cd ~
```

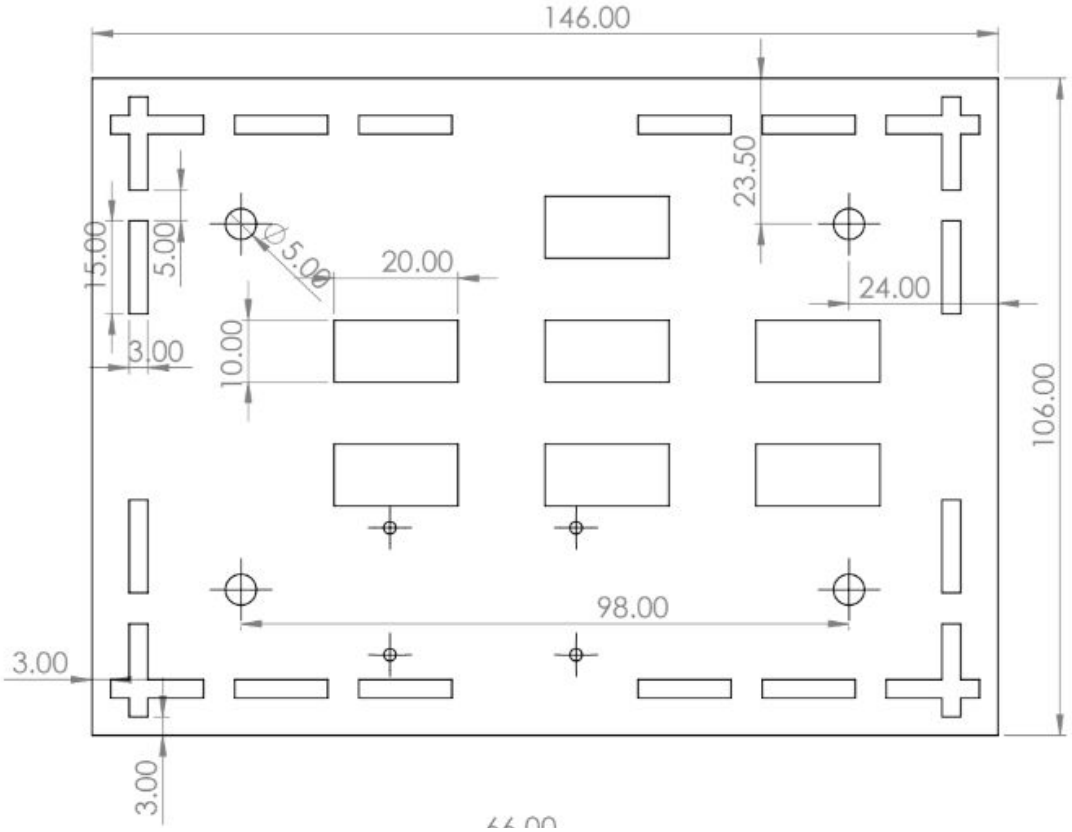
2. CASE DESIGN

Materials:

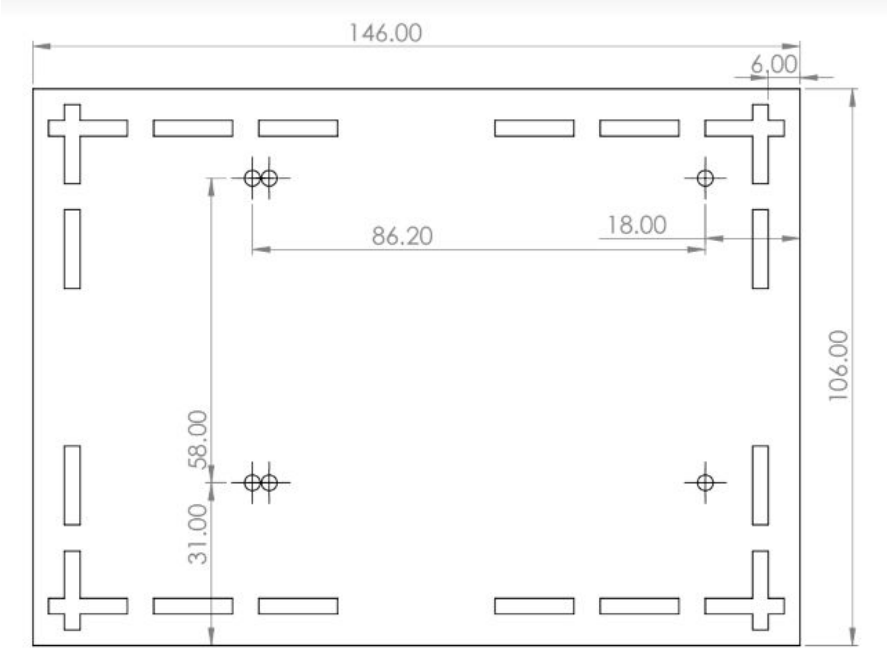
1. Medium-density fibreboard (MDF) 3 mm A3 sheet
2. Screws m3 x 8 mm (to fix the support plate to the box)
3. Screws m3 x 20 mm (for fixing the fan)
4. 5V fan

A proprietary design is used for the nano jetson case consisting of 6 pieces that are assembled to form the complete structure.

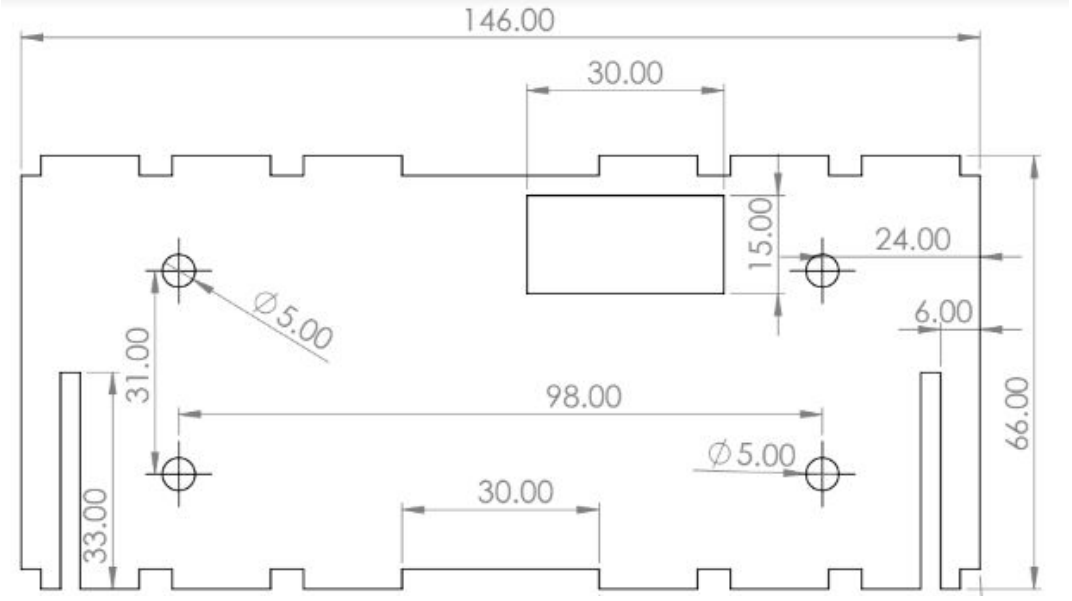
2.1. Main base:

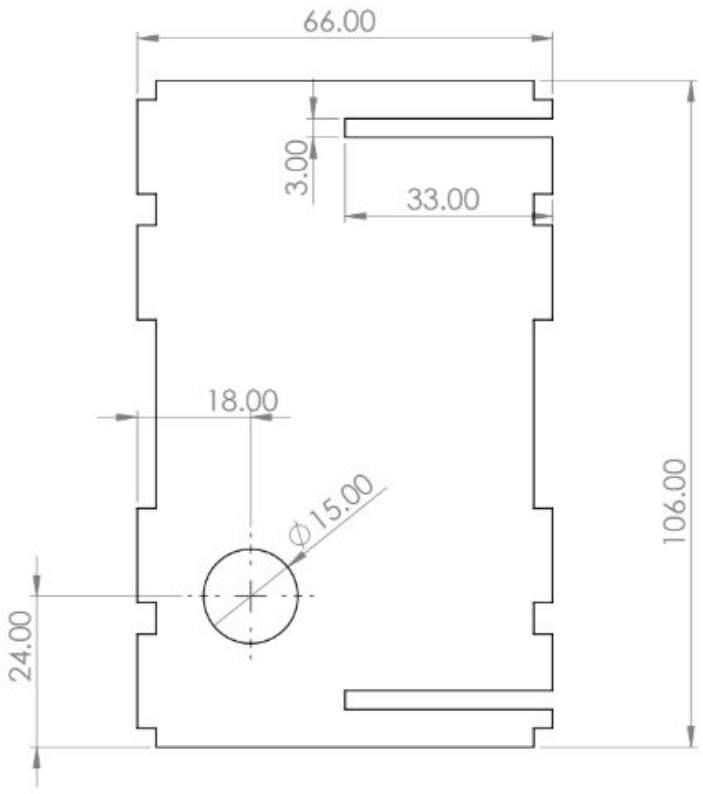


2.2. Top cover:

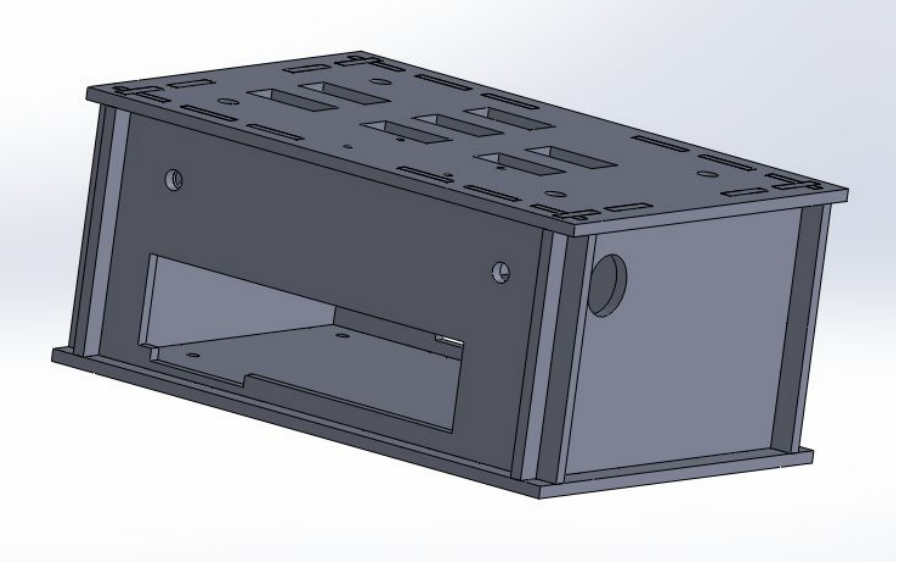


2.3. Side faces :





2.4. 3D Assembly:



3. CIRCUIT DESIGN

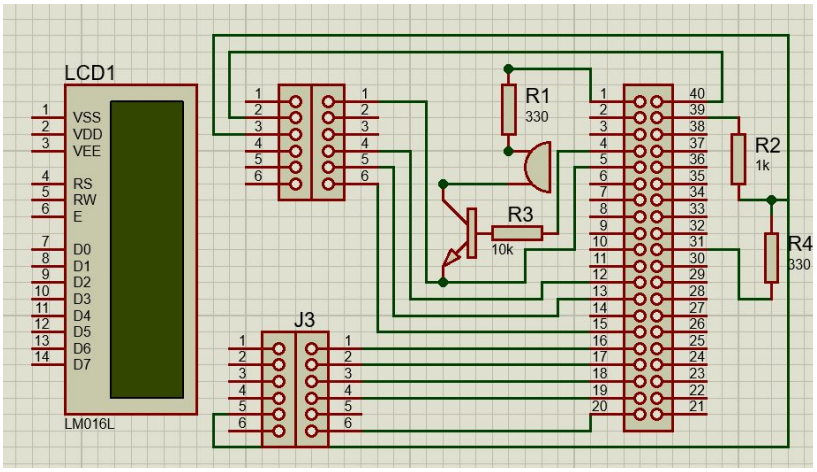
3.1. Materials:

1. 2 Resistor 330 ohms
2. 1 Resistor 10k ohms
3. 1 Resistor 1k ohms
4. 1 Buzzer

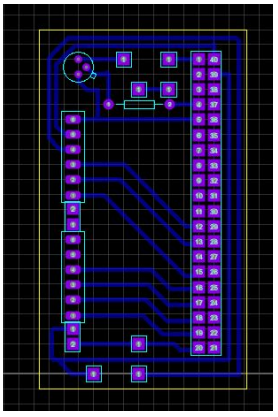
3.2. Design

This circuit was inspired by this repository:
https://www.jetsonhacks.com/2019/06/07/jetson-nano-gpio/

Custom design on Proteus:

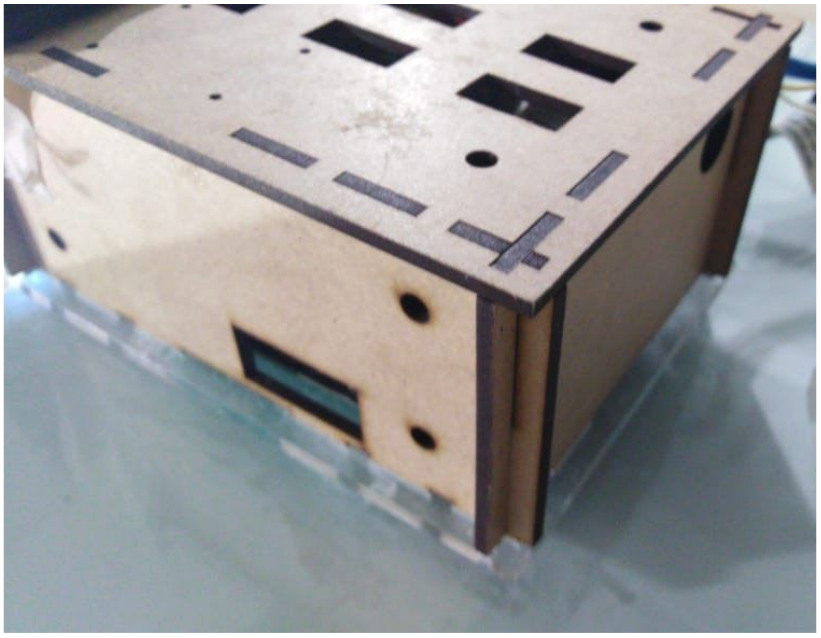


PCB design:

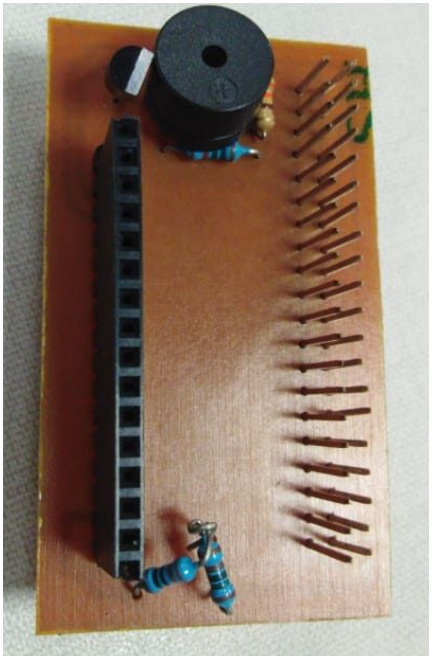


4. Hardware implemented

4.1. Case



4.2. Circuit



5. Algorithm

Real Time Road Space Rationing control using Jetson Nano

5.1. GETTING STARTED

5.1.1. Using ISO

- Download the image of jetson nano with all libraries and code implemented :

    www.google.com

- Open folder Pico_placa_SSD :

    cd home/dlinano/Documents/Pico_placa_SSD

- Run code pico_placa.py

    python3 pico_placa.py

- If you want to use special hardware of image capture:

  - webcam :

        self.video = cv2.VideoCapture(0)
  - video :

        self.video = cv2.VideoCapture('PATH_OF_VIDEO')
  - picam :

        self.video = cv2.VideoCapture(gstreamer_pipeline(flip_method=0),cv2.CAP_GSTREAMER)

5.1.2. Step by step

Download the repository:

    https://github.com/CristianLazoQuispe/Pico_y_placa.git

Save it in the documents folder

- Follow the instructions of installation on pdf:

Pico_y_placa.pdf

5.1.2.1. SSD MobileNet

This project use the model od SSD MobileNet on TensorRT
We use the model of the repository tensorrt_demos:

```
git clone https://github.com/jkjung-avt/tensorrt_demos.git
```

Download and implement the TensorRT model from SSD Mobilenet and copy it to the ssd folder

5.1.2.2. OpenALPR

Download the repository and copy the runtime_data folder to Pico_y_placa:

```
git clone https://github.com/openalpr/openalpr.git
```