Conteo de Vehículos – Nota de Aplicación

Resumen

Conteo de vehículos es una aplicación que está totalmente resuelta con el modelo uRAD Industrial + Tracking Software. Te permite contar vehículos en múltiples carriles, medir la velocidad y clasificarlos, con gran precisión y mínima configuración. El sistema trabaja en la banda de 60 GHz, una banda de frecuencia para emisión disponible en todo el mundo, lo que facilita la certificación de cualquier producto.

Caso de Uso

El sistema es muy versátil y se puede usar en muchos escenarios de conteo:

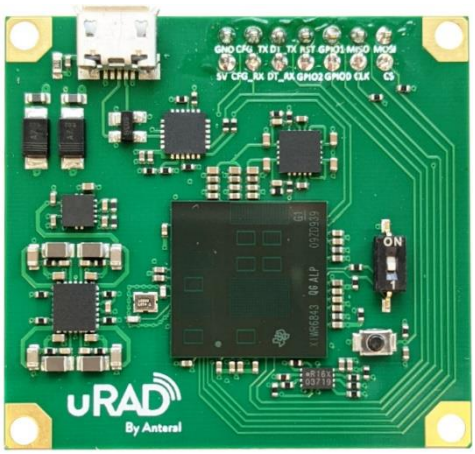
- Carreteras urbanas e interurbanas.
- Medida de velocidad hasta 160 km/h.
- Monitorización de hasta 6 carriles con un solo radar.
- Contar vehículos con velocidad positiva (alejándose) o negativa (acercándose) al mismo tiempo.
- Escenarios de tráfico densos o ligeros.

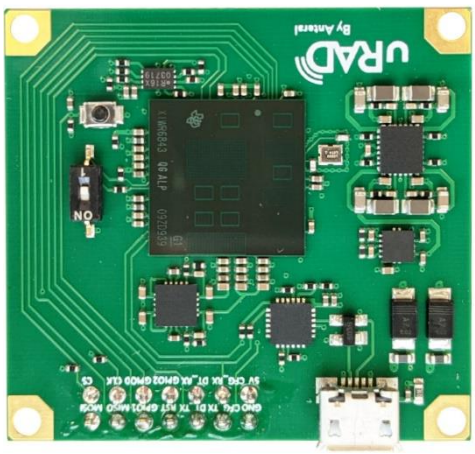
Montaje

Hay que tener en cuenta tres aspectos en el montaje del sistema de conteo de vehículos:

- **Orientación del radar**

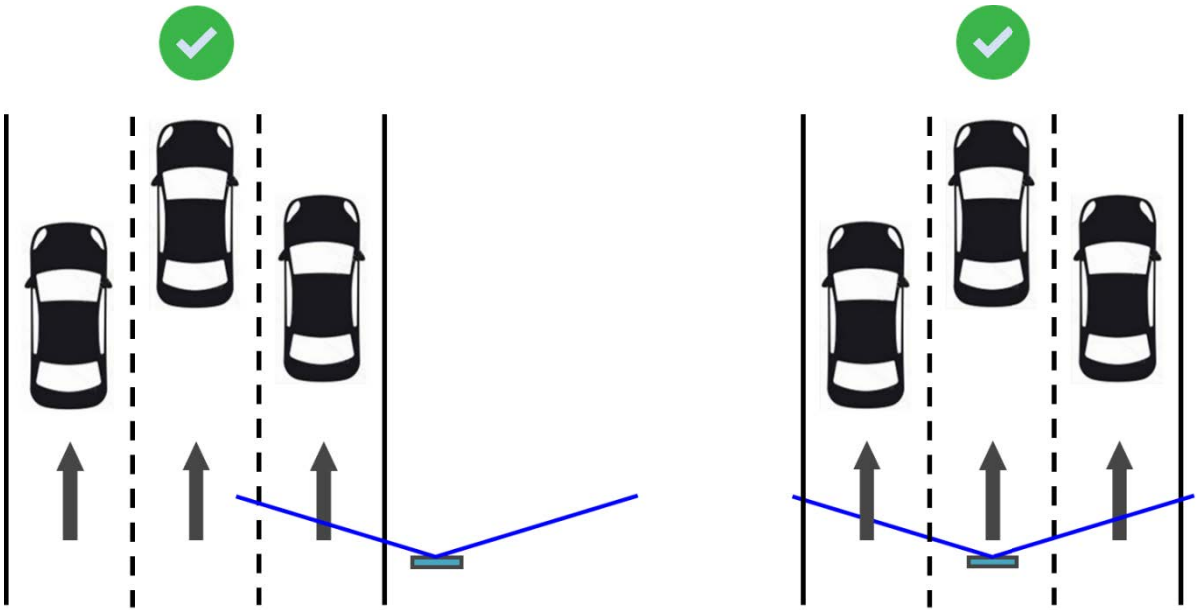
uRAD Industrial se tiene que montar de manera que el conector USB quede en la parte superior o inferior. Cualquiera de las dos opciones es correcta.



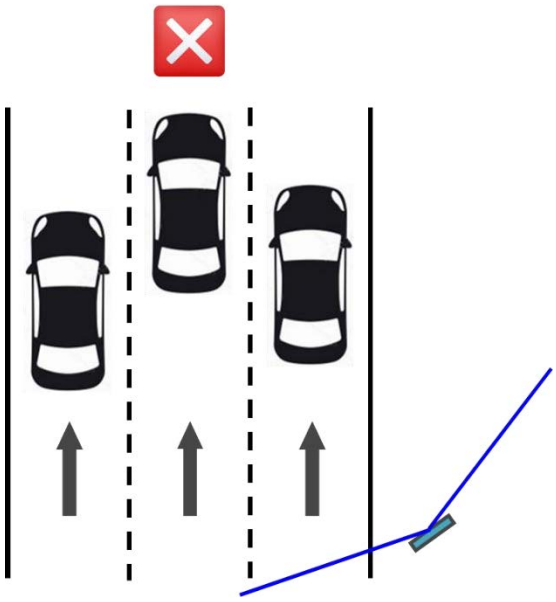


- **Montaje del radar respecto a la carretera**

uRAD Industrial se puede montar a un lado de la carretera o encima de ella. **Lo más importante es montarlo paralelo a la carretera.** Mire las siguientes imágenes:



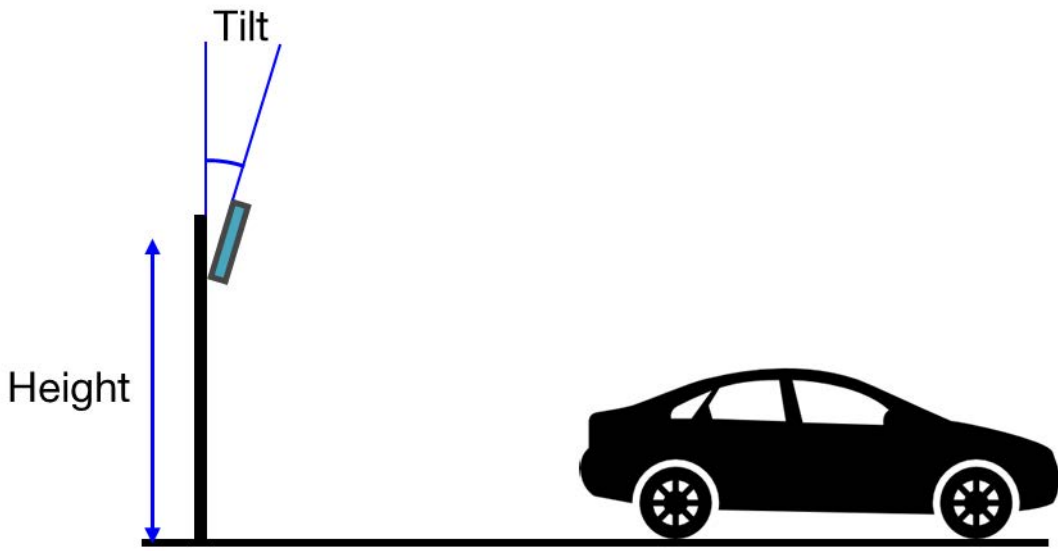
No gire el radar para apuntar al centro de la carretera. La precisión se verá comprometida. El campo de visión es suficiente para medir varios carriles con un montaje paralelo.



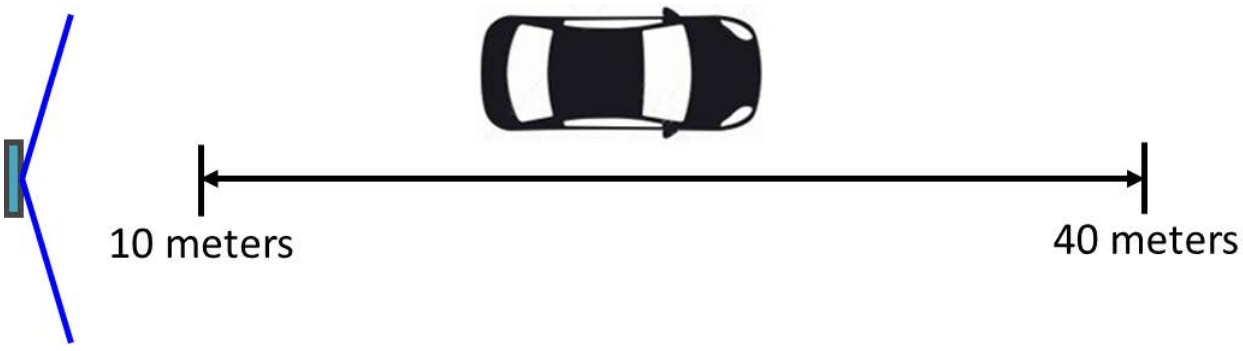
- **Altura del radar e inclinación**

Para tener una visión adecuada de los vehículos y, por lo tanto, evitar que un vehículo bloquee a otro, el radar se debe colocar a una altura de 3 metros o más. Dependiendo de la altura de montaje, el radar se debe inclinar hacia abajo, pero solo un poco. Siga estas recomendaciones.

ALTURA (height)	INCLINACIÓN (tilt)
2 m to 3 m	0 grados
3 m to 4 m	5 grados
4 m to 5 m	10 grados
5 m to 6 m	15 grados



La distancia óptima de detección de los vehículos son 20 metros. Por eso es recomendable instalar el radar en vías donde se pueda monitorizar el tráfico, **sin que este se detenga,** entre una distancia de 10 a 40 metros desde el radar.



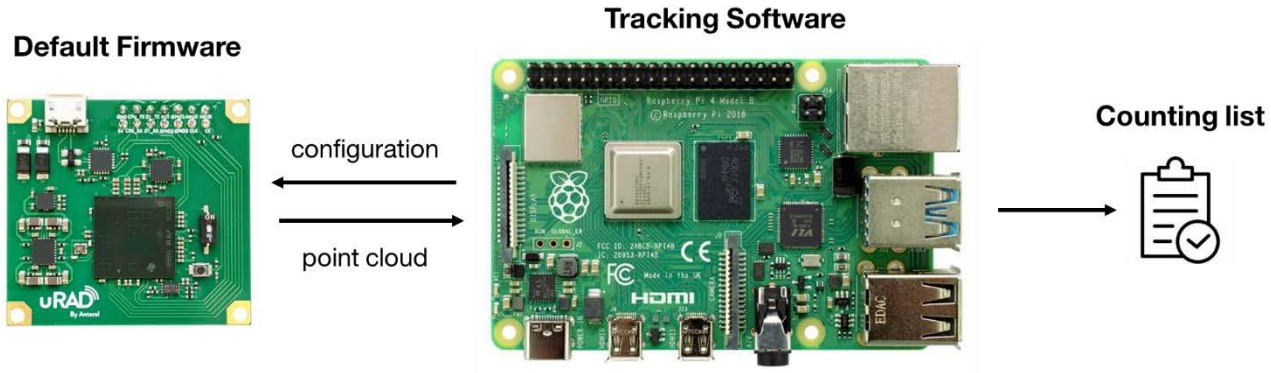
Tracking Software

El Tracking Software consiste en un programa de Python que se debe ejecutar en el dispositivo maestro que controla uRAD Industrial. Este software, junto con el dispositivo maestro:

- Envía a uRAD Industrial los correspondientes parámetros de configuración.
- Recibe de uRAD Industrial la nube de puntos 3D con las coordenadas espaciales X, Y, Z, velocidad y SNR (Signal to Noise Ratio).
- Procesa la nube de puntos para identificar vehículos, extrae su velocidad y los clasifica.
- Guarda la lista de conteo con la información relevante.

El tipo de objetivo identifica, en primera instancia, peatones, vehículos normales y vehículos de gran longitud (camiones grandes). Además, la clasificación se puede refinar para incluir bicicletas.

Cualquier dispositivo que pueda correr Python y tenga comunicación UART/USB puede usarse como dispositivo maestro. Un ejemplo típico de dispositivo que se puede integrar fácilmente con uRAD Industrial es una Raspberry Pi.



Los requisitos del sistema no son muy exigentes. Contáctenos para evaluar su dispositivo previsto.

• Archivos de Python

El Tracking Software está compuesto por un programa principal de Python y una librería compilada.

El programa principal, nombrado ***SDK_tracking_XX.py***, es de código abierto y es el que se debe lanzar. La librería, nombrada ***uRAD_Tracking_vX.X.py***, está compilada y es la que se encarga de procesar la nube de puntos.

Proporcionamos dos programas principales, uno para UART asumiendo que se usa una Raspberry Pi, ***SDK_tracking_RPi.py***, y otro para USB, ***SDK_tracking_USB.py***. El de Raspberry Pi tiene una particularidad con el pin RESET que se comenta después.

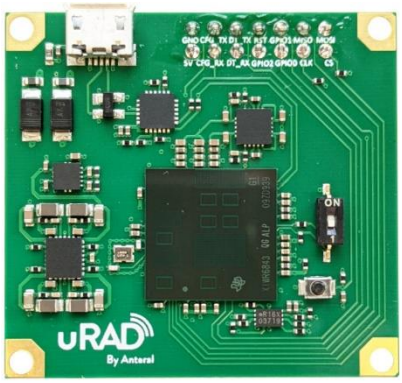
- **Parámetros de configuración**

Solo es necesario introducir unos pocos parámetros de entrada en ***SDK_tracking_XX.py***. Al comienzo del programa, hay unas líneas, entre la 9 y la 15, que hay que configurar:

```
8  #### CONFIGURATION PARAMETERS ####
9   usbConnector_upward = False      # Define the radar orientation
10  pitch_angle = 5                  # Tilt angle with the vertical
11  velocity_positive = True         # Counting vehicles driving away
12  velocity_negative = True         # Counting vehicles approaching
13  frequency_channel = 1            # Frequency channel within the 60-64 GHz band
14  x_min = -20                      # Minimum distance horizontal direction
15  x_max = 20                       # Maximum distance horizontal direction
```

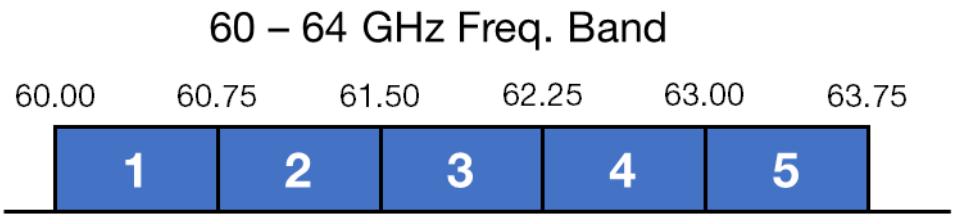
- **usbConnector_upward:** define la orientación del radar mediante la posición del conector USB. Introduce esta variable de acuerdo a la siguiente imagen.

usbConnector_upward = True



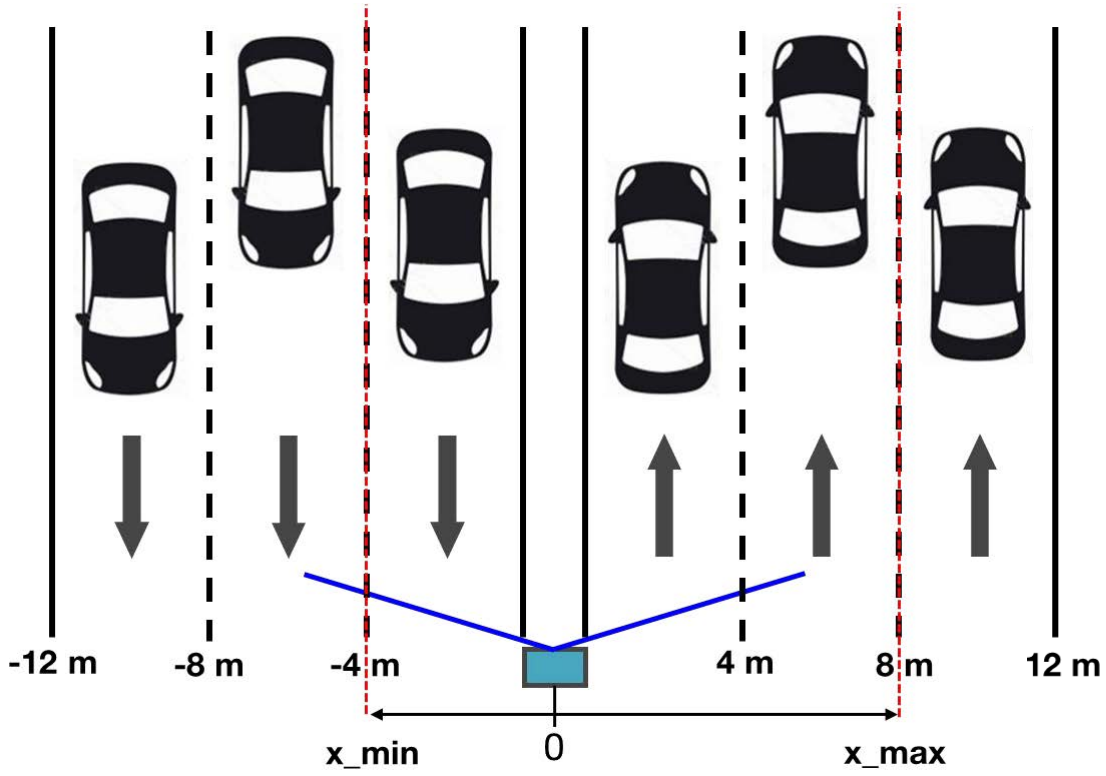
usbConnector_upward = False

- **pitch_angle**: define el ángulo de inclinación del radar con la vertical (en grados). Introduce este parámetro de acuerdo a tu montaje.
- **velocity_positive**: introduce True/False para contar/descartar vehículos alejándose del radar.
- **velocity_negative**: introduce True/False para contar/descartar vehículos aproximándose al radar.
- **frequency_channel**: uRAD Industrial puede funcionar en cinco canales diferentes en la banda de 60-64 GHz. Esto es útil, por ejemplo, para cumplir con las regulaciones específicas de cada región o para limitar la interferencia entre dispositivos cercanos. Introduce este parámetro de 1 a 5 según la siguiente imagen.



- **x_min**: define la mínima distancia en metros en la dirección horizontal que quiere considerar para contar vehículos. Esto es útil para limitar el número de carriles a monitorizar.
- **x_max**: define la máxima distancia en metros en la dirección horizontal que quiere considerar para contar vehículos. Esto es útil para limitar el número de carriles a monitorizar.

Imagina, con la siguiente imagen, que quiere medir solo los dos carriles a la derecha del radar y un carril a la izquierda del radar. Debes introducei x_min = -4 y x_max = 8.



- **Resultados de salida**

El usuario puede seleccionar tres tipos de resultados:

```
#### OUTPUT RESULTS ####
saveResults = True                # Save counting results (Vehicle_results.txt)
saveVehicleFeatures = True        # Save vehicle features (Vehicle_features.txt)
saveRawData = True                # Save the full point cloud (PointCloud.txt)
output_filename = './Vehicle_results.txt'
output_datetimeFormat = 0         # 0 for datetime (yyyy/mm/dd HH:MM:SS),
                                    1 for timestamp (UNIX)
```

- **saveResults:** crea un .txt nombrado **Vehicle_results.txt** con la información más relevante. Cada línea de este archivo de texto corresponde con un vehículo detectado. La información de cada columna es la siguiente:

`Marca_temporal  Velocidad  Distancia_x  Tipo_vehiculo`

`Marca_temporal`: fecha y hora (yyyy/mm/dd HH:MM:SS) o marca temporal (UNIX) en la que el vehículo es detectado. Se toma del sistema del dispositivo, la Raspberry Pi en este caso.

`Velocidad`: velocidad en km/h del vehículo. Velocidad positiva significa vehículo alejándose y negativa vehículo acercándose.

`Distancia_x`: estimación de la distancia horizontal en metros del vehículo. Útil para identificación de carril.

`Tipo_vehículo`: identificación del tipo de vehículo. 1 = vehículo normal, 2 = vehículo largo, 3 = peatón, 4 = bicicleta.

- **saveVehicleFeatures:** crea un .txt nombrado **Vehicle_features.txt** con información adicional para cada vehículo. Esta información es especialmente útil para el equipo uRAD para comprobar la clasificación y mejorarla para cada escenario particular.

  Cada línea del archivo de texto corresponde con un vehículo detectado. La información de cada columna es la siguiente:

  `Marca_temporal Velocidad Longitud_vehículo Densidad Puntos Amplitud`

  `Marca_temporal`: fecha y hora (yyyy/mm/dd HH:MM:SS) o marca temporal (UNIX) en la que el vehículo es detectado. Se toma del sistema del dispositivo, la Raspberry Pi en este caso.

  `Velocity`: velocidad en km/h del vehículo. Velocidad positiva significa vehículo alejándose y negativa vehículo acercándose.

  `Vehicle_length`: cuando un vehículo es susceptible de ser un vehículo largo, se indica una estimación de su longitud. 0 significa que este vehículo ha sido clasificado directamente como un vehículo normal.

  `Densidad`: cuando un vehículo es susceptible de ser un vehículo largo, la densidad de puntos asociada a este vehículo.

  `Points`: número de puntos de la nube de puntos 3D asociados al vehículo. Es diferente a la densidad.

  `Amplitud`: amplitud media de los puntos asociados de la nube de puntos 3D.

- **saveRawData:** crea dos .txt nombrados PointCloud.txt y stats.txt. Esta información es útil para el equipo de uRAD para comprobar el correcto funcionamiento del radar.

PointCloud.txt contiene la nube de puntos 3D complete. Cada línea corresponde con una trama del radar. Cada línea contiene (X,Y,Z,velocidad,SNR,ruido) de todos los puntos con la marca temporal al final de la línea.

stats.txt contiene un registro que asocial cada trama del radar con su marca temporal.

- **output_filename:** para especificar la ruta donde quieres que se guarde el archivo Vehicle_results.txt.
- **output_datetimeFormat:** para elegir el formato del campo **`Marca_temporal`** en los archivos .txt. 0 para fecha y hora (yyyy/mm/dd HH:MM:SS), 1 para marca temporal (UNIX).

La ruta de los archivos Vehicle_features.txt, stats.txt y PointCloud.txt se pueden elegir en las líneas 38 a 40.

```
38  stats_filename = './stats.txt'
39  pointCloud_filename = './PointCloud.txt'
40  features_filename = './Vehicle_features.txt'
```

- ## Puerto UART

Cuando se usa ***SDK_tracking_RPi.py***, hay que definir el nombre del Puerto UART del dispositivo maestro. En el caso de Raspberry Pi, que solo tiene un puerto UART es:

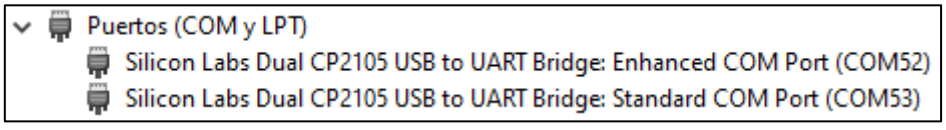
```
22  #### UART PORT ####
23  Port_name = '/dev/serial0'
```

- ## USB port

Si el puerto usado es el USB en lugar de UART, con ***SDK_tracking_USB.py***, hay que definir el nombre del puerto USB del dispotivio maestro.

```
#### USB PORT ####
configPort_name = 'COM1'
dataPort_name = 'COM2'
```

Cuando uRAD Industrial se conecta por USB, se reconocen dos puertos COM. **configPort** es el que se identifica como **Enhanced** y **dataPort** es el **Standard.**



- **Reset**

uRAD Industrial tiene un pin de reset en el conector de pines que se puede controlar con un GPIO. Es recomendado usarlo para reseterar el radar cada vez que se ejecuta el programa principal. Para hacer un reset, hay que enviar un flanco de subida al pin reset.

Si no tiene el pin reset conectados, introduzca la variable **`reset = False`** en la línea 27.

En el programa principal, hemos incluido este proceso de reset considerando que utilizamos una Raspberry Pi. En este caso, asumimos que el pin reset de uRAD está conectado al pin GPIO número de 18 de la Raspberry Pi.

```
25  #### RESET ####
26  Reset_pin_number = 18
27  reset = True
```

Con esta simple sección `if`, fijamos a nivel bajo el GPIO18 de la Raspberry Pi y luego, después de 10 ms, lo fijamo alto para ejecutar el reset.

```
122     if (reset):
123         from gpiozero import OutputDevice
124         PinReset = OutputDevice(Reset_pin_number)
125         PinReset.off()
126         sleep(10e-3)
127         PinReset.on()
128         sleep(1)
```