Control de Velocidad – Notas de Aplicación

Descripción general

Esta nota de aplicación describe como desarrollar un dispositivo de control de velocidad basado en las soluciones uRAD de 24 GHz.

La idea consiste en monitorizar con gran precisión la velocidad de vehículos que circulan por un vía urbana o interurbana. Se puede monitorizar la velocidad tanto de vehículos acercándose como alejándose del radar.

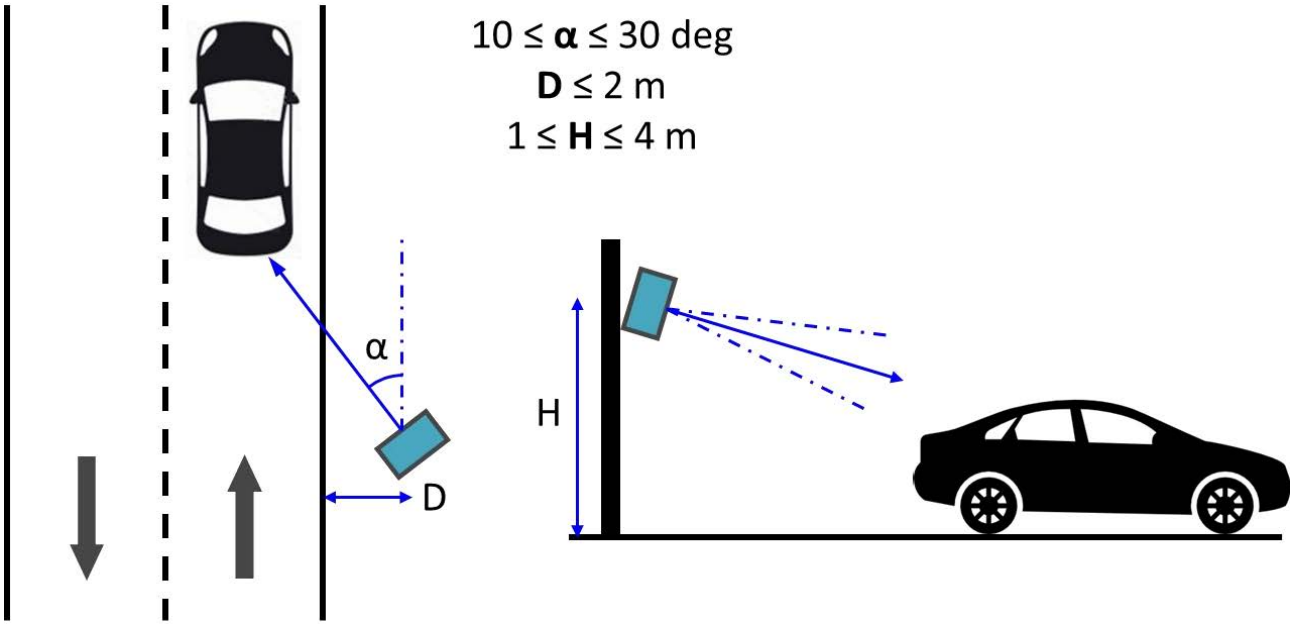
Dispositivos

Esta aplicación se puede desarrollar con cualquiera de las soluciones a 24 GHz de uRAD: uRAD Arduino, uRAD Raspberry Pi o uRAD USB. Elegimos estas soluciones por dos motivos principales en sus prestaciones:

- Capacidad de funcionar en modo de onda continua, lo que permite medir velocidades hasta 270 km/h con alta precisión.
- Ángulo de visión estrecho de 30 x 30 grados.

Montaje

El montaje sugerido consiste en colocar el radar a un lado de la carretera siguiendo estas recomendaciones:



- A una distancia **D** del borde de la calzada menor a 2 metros. Es preferible detectar a los vehículos por detrás, ya que, en general, la forma de la parte trasera de los vehículos favorece la reflexión de la onda radar.

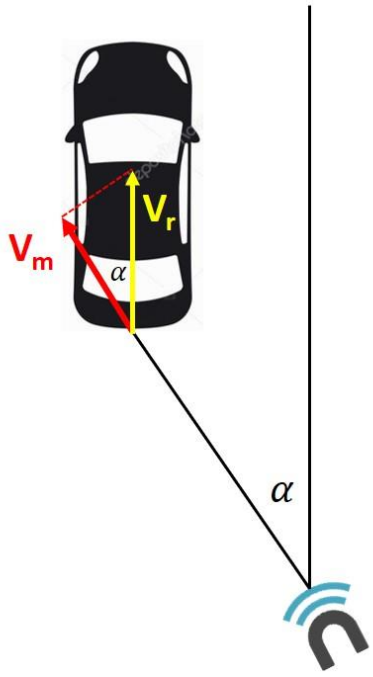
- Con un ángulo de visión α cercano a 10 y con un máximo de 30 grados. A mayor es el ángulo, mayor es la imprecisión cometida en la velocidad, como se describe en el punto siguiente.
- A una altura **H** entre 1 y 4 metros. En caso de colocarlo a una altura muy superior a la altura del vehículo, inclinar ligeramente el radar hacia abajo.

Aspectos teóricos

El principal aspecto teórico a tener en cuenta es que, la velocidad medida, es siempre la velocidad radial, es decir, la componente de velocidad que cae sobre la recta que une el radar y el vehículo.

En la siguiente imagen, la velocidad que mide uRAD es $\mathbf{V_m}$, sin embargo, la velocidad real del vehículo es $\mathbf{V_r}$. La velocidad real se extrae dividiendo por el coseno del ángulo.

$$Velocity_{real} = Velocity_{measured} / \cos \alpha$$



Si el ángulo de colocación es bajo, el error cometido es pequeño. Por ejemplo, para un coche circulando a 100 km/h, con α = 10 grados, uRAD mide 98.48 km/h. Para un ángulo de 30 grados, uRAD mide 86.60 km/h, lo que resulta en un error mayor.

Por lo tanto, cuando un coche pase por delante del radar, su velocidad será medida con diferentes ángulos α. Cuando el vehículo está cerca de uRAD, el ángulo será mayor y la velocidad medida más imprecisa, y conforme se aleja, el ángulo será menor y la velocidad medida más precisa. Esto lo veremos claramente un poco más adelante en el ejemplo con datos reales.

Configuración

La configuración inicial sugerida es la misma que en el ejemplo *uRAD_velocity_meter*, y es válida para cualquiera de los modelos de uRAD de 24 GHz. En este caso el software/firmware del radar debe ser el SDK 1.1.

- **mode = 1**: modo Doppler de onda continua. El mejor modo para medir solo velocidad.
- **f0 = 125**: emitir a 24.125 GHz, en el centro de la banda.
- **BW = 240**: parámetro irrelevante en el modo 1 ya que se emite solo a una frecuencia.
- **Ns = 200**: 200 muestras para tener la mejor precisión.
- **Ntar = 3**: detectar la velocidad de hasta tres objetivos.
- **Vmax = 75**: el valor de 75 corresponde con buscar en todo el rango de velocidades disponibles (75 m/s = 270 km/h).
- **MTI = 0**: parámetro irrelevante en el modo 1.
- **Mth = 0**: irrelevante ya que no queremos información booleana de movimiento.
- **Alpha = 20**: definimos la sensibilidad inicial Alpha en 20.
- **distance_true = False**: no es posible recibir información de distancia en el modo 1.
- **velocity_true = True**: recibir información de velocidad.
- **SNR_true = True**: recibir información de la relación señal a ruido.
- **I_true = False**: no nos interesa los datos RAW de la componente en fase de la señal IF.
- **Q_true = False**: no nos interesa los datos RAW de la componente en cuadratura de la señal IF.
- **movement_true = False**: no nos interesa el indicador booleano de si se ha detectado movimiento o no.

Esta configuración inicial es solo una recomendación, ya que los parámetros se deben ajustar de acuerdo a su escenario particular. En el manual de usuario se describen todos los parámetros de configuración en detalle. Los parámetros más relevantes a ajustar son:

- **Alpha** ajusta la sensibilidad. El mínimo es 3 y el máximo 25. Un valor menor aumenta la sensibilidad, pero también las detecciones fantasmas.
- **Ntar** selecciona el número de targets a buscar. Los resultados de velocidad se ordenan de mayor a menor SNR detectado. Empezar con hasta 3 targets es una buena opción para ver las detecciones reales y también detecciones fantasmas si las hubiese. También si se quieren ver al mismo tiempo vehículos en ambos sentidos.

Programación de un ejemplo

El código siguiente también se basa en el script de ejemplo proporcionado con la compra *uRAD_velocity_meter*. En este caso, además de mostrar los resultados por pantalla, lo que hacemos es guardar los resultados de velocidad y SNR obtenidos junto con su marca temporal en un archivo de texto, para luego dibujarlos en una gráfica a lo largo del tiempo.

El código corresponde con el script programad en Python, pero un ejemplo alternativo está disponible también para Arduino. Se muestra solo la parte de código más relevante.

```
# switch ON uRAD
return_code = uRAD_USB_SDK11.turnON(ser)
if (return_code != 0):
    closeProgram()

if (not usb_communication):
    sleep(timeSleep)

# loadConfiguration uRAD
return_code = uRAD_USB_SDK11.loadConfiguration(ser, mode, f0, BW, Ns, Ntar, Vmax, M
TI, Mth, Alpha, distance_true, velocity_true, SNR_true, I_true, Q_true, movement_tr
ue)
if (return_code != 0):
    closeProgram()

if (not usb_communication):
    sleep(timeSleep)

resultsFileName = 'velocity.txt'
fileResults = open(resultsFileName, 'a')

# infinite detection loop
while True:

    # target detection request
    return_code, results, raw_results = uRAD_USB_SDK11.detection(ser)
    if (return_code != 0):
        closeProgram()

    # Extract results from outputs
    NtarDetected = results[0]
    velocity = results[2]
    SNR = results[3]

    t_i = time()

    velocity_string = ''
    # Iterate through desired targets
```

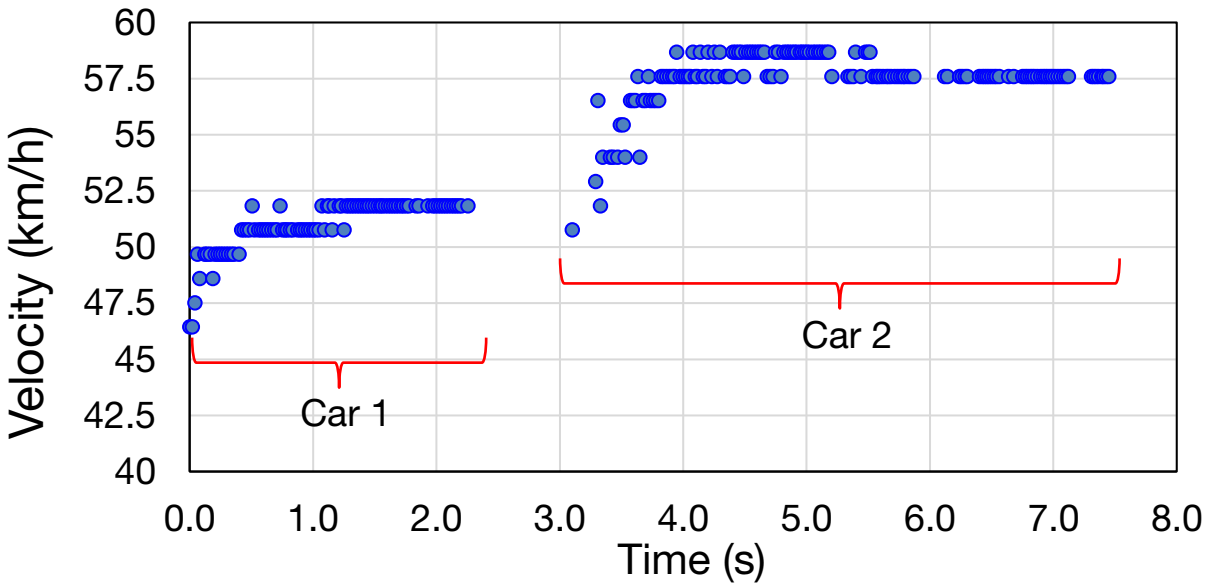
```
    for i in range(NtarDetected):
        # If SNR is big enough
        if (SNR[i] > 0):
            # Prints target information
            print("Target: %d, Velocity: %1.1f m/s, SNR: %1.1f dB" % (i+1, velocity
[i], SNR[i]))

            # Save target information
            velocity_string += ('%1.1f %1.1f ' % (velocity[i], SNR[i]))

    if (NtarDetected > 0):
        fileResults.write(velocity_string + '%1.3f\n' % t_i)
        print(" ")
```

Medidas reales

A continuación, se muestran resultados representativos de medidas reales.

El radar está colocado como en la imagen siguiente, a una altura de unos 90 cm, a un metro de la calzada y con un ángulo aproximado de 10 grados.

La siguiente gráfica muestra los resultados de velocidad capturados por el radar de dos coches que han pasado con una diferencia algo menor a un segundo entre ellos.



Se observa en ambos coches como las primeras muestras se miden con un error mayor debido al ángulo relativo entre el radar y el vehículo, y conforme el vehículo se va a alejando la velocidad se estabiliza hasta su valor real, 51.84 km/h para el primer coche y 57.6 km/h para el segundo.

La precisión en la medida es de ±1.08 km/h (±0.3 m/s).

En este caso, la velocidad es positiva, e indica que los coches se alejan del radar. Se puede monitorizar al mismo tiempo coches que se alejan y acercan del radar por lo que sería válido para un montaje de dos carriles con distinto sentido de circulación.