**Contenido**

[Documentación y Buenas Prácticas 2](#_Toc163587397)

[1)Desarrollo de Alarmas 2](#_Toc163587398)

[2) Optimización del algoritmo de “send\_data\_to\_server” 4](#_Toc163587399)

[Gestión de Proyectos 6](#_Toc163587400)

[1)Listas de tareas 6](#_Toc163587401)

[2)Diagrama de Gantt 7](#_Toc163587402)

[3)Metromap 7](#_Toc163587403)

# Documentación y Buenas Prácticas

El siguiente documento tiene como objetivo desarrollar las funcionalidades realizadas en el algoritmo original presentado por Vemo para este Desafío.

Las funcionalidades se implementaron en lenguaje Python.

Las mismas fueron:

1. Desarrollos de Alarmas
2. Optimización del algoritmo de “send\_data\_to\_server”

## 1)Desarrollo de Alarmas

Se generó una lista “alarms setup” con las siguientes alarmas:

alarms\_setup = [

 "alarm Main Battery Voltage",

 "alarm Main Battery Current",

 "alarm Speed",

 "alarm Main Battery Temperature",

 "alarm Door Open",

 "alarm Tire Pressure"

]

Todas las alarmas tienen 2 estados: 0 apagadas y 1 encendidas

De las cuales se pueden clasificar en 2 tipo de categorías de alarmas:

Alarmas de Conducción Peligrosa:

* “alarm Speed”: la cual indica exceso de velocidad cuando se supera el valor de 100.

Alarmas de Problemas en el Vehículo:

* "alarm Main Battery Voltage": Se enciende cuando el valor de voltaje es inferior a los 360, inclusive. Esto de manera de contar con un margen para realizar la carga.
* "alarm Main Battery Current": Se activa cuando el valor de corriente es menor igual a 0, o si está por encima de los 40 A (exceso en consumo, puede ser un desperfecto en un componente).
* "alarm Main Battery Temperature": Esta alarma se activará al sobrepasar el valor de temperatura de 75.
* "alarm Door Open": Esta alarma se pone en “1” cuando una puerta esta abierta, es decir en “0”.
* "alarm Tire Pressure": La misma se activa cuando arroja el sensor el valor de 28 o de 34.

Para la creación de la lista de alarmas se utilizó la siguiente función (get\_alarm) y para asignarles valores (0 o 1) se reutilizó el “Class Signal” del programa original.

def get\_alarm(signals, alarmas\_signals):

 for j in range(len(signal\_dictionary)):

  id = j+1

  v = 0

  if (id == 1)and(signals[j].val <=360):

   v = 1

  elif (id == 2)and((signals[j].val <=0)or(signals[j].val >40)):

   v = 1

  elif (id == 3)and(signals[j].val >=100):

   v = 1

  elif (id == 4)and(signals[j].val > 75):

   v = 1

  if (id == 5)and(signals[j].val == 0):   # la alarma se pone en 1 si la puerta esta abierta(0)

   v = 1

  if (id == 6)and((signals[j].val == 28)or(signals[j].val == 34)):

   v = 1

  alarmas\_signals[j] = Signal(id, v)

class Signal:

 def \_\_init\_\_(self, id, val):

  self.id = id

  self.val = val

Con respecto a los resultados obtenidos, se hicieron las siguientes pruebas:

Cuando la opción debug es True, Estas alarmas son enviadas al servidor solo si están activas (estado 1), de lo contrario no se informarán.

A continuación, se muestra la salida, se puede apreciar que son discriminadas los dos tipos de categorías de alarmas, las de conducción peligrosa y las de problemas en el vehículo:

Sending data to server...

Data sent:

Signal: Main Battery Voltage, Value: 400, Time: 1712622274

Signal: Main Battery Current, Value: 12, Time: 1712622274

Signal: Speed, Value: 146, Time: 1712622274

Alarmas de Conduccion Peligrosa: alarm Speed, Value: 1, Time: 1712622274

Signal: Main Battery Temperature, Value: 64, Time: 1712622274

Signal: Door Open, Value: 0, Time: 1712622274

Alarmas de Problemas en el Vehiculo: alarm Door Open, Value: 1, Time: 1712622274

Cuando debug está en esta False, solo informará el ID de alarmas, el valor de las magnitudes sensada y los valores de alarmas correspondientes.

A continuación, se muestra la salida:

[Running] python -u "c:\Users\Diaz.R155\OneDrive - BGH SA\Escritorio\vemo\Desafio Vemo\_Rodrigo Diaz Martucci\_08042024.py"

1712622683,1,402,0

1712622683,2,-48,1

1712622683,3,100,1

1712622683,4,41,0

1712622683,5,0,1

1712622683,6,28,1

1712622684,1,409,0

1712622684,2,47,0

1712622684,3,78,0

1712622684,4,96,1

1712622684,6,29,0

1712622685,1,401,0

1712622685,2,37,0

1712622685,3,58,0

1712622685,4,87,1

1712622685,5,1,0

1712622685,6,34,1

## 2) Optimización del algoritmo de “send\_data\_to\_server”

Para realizar esta solicitud, la idea fue mantener la estructura original. La manera de resolver este punto fue agregando una lista auxiliar “aux” global, la cual esta inicializada en “None” cada elemento. Estos se comparan con los valores de “signals” los cuales en la primera corrida van a ser distintos y por lo tanto se enviarán los datos al servidor.

Para cuando transcurra 1 segundo posterior e ingresen los nuevos valores de los sensores, estos nuevos datos se compararán con cada elemento de la lista aux, los cuales serán los anteriores a estos nuevos. Así si son igual no serán enviados al servidor, de lo contrario sí serán transmitidos al mismo.

def send\_data\_to\_server(signals, timestamp, alarms\_signals):

 global debug

 global aux   # aux es una lista auxiliar para guardar los valores de signals

 if debug:

  print("\nSending data to server...")

  print("Data sent:")

 for i in range(len(signal\_dictionary)):

  if debug:

   ### Optimizacion del algoritmo ###

   if aux[i]!=signals[i].val:   # Evalua que el valor anterior (1 seg antes) sea distinto al presente.

    print(f"Signal: {signal\_dictionary[i]}, Value: {signals[i].val}, Time: {timestamp}")

    if alarms\_signals[i].val != 0:

     if i != 2:   # solo envia las alarmas activadas (en 1)

      print(f"Alarmas de Problemas en el Vehiculo: {alarms\_setup[i]}, Value: {alarms\_signals[i].val}, Time: {timestamp}")  # Alarmas por falla en el vehiculo

     else:

      print(f"Alarmas de Conduccion Peligrosa: {alarms\_setup[i]}, Value: {alarms\_signals[i].val}, Time: {timestamp}")  # Alarmas por manejo inapropiado

    aux[i] = signals[i].val   # Le asigno a la lista auxiliar los valores de signals para ser comparados en el proximo periodo de tiempo

  else:   # Modo Debug False, solo envia el timestamp, ID y valores de sensores y alarmas. No repite data al server

    for i in range(len(signal\_dictionary)):

     if aux[i]!=signals[i].val:

      print(f"{timestamp},{signals[i].id},{signals[i].val},{alarms\_signals[i].val}")

      aux[i] = signals[i].val

A continuación, se puede observar una salida donde el valor de Door Open (valor = 0) será el mismo en 2 periodos de tiempo seguidos, por lo cual solo es enviado el primero.

Sending data to server...

Data sent:

Signal: Main Battery Voltage, Value: 445, Time: 1712622266

Signal: Main Battery Current, Value: 25, Time: 1712622266

Signal: Speed, Value: 21, Time: 1712622266

Signal: Main Battery Temperature, Value: 75, Time: 1712622266

Signal: Door Open, Value: 0, Time: 1712622266

Alarmas de Problemas en el Vehiculo: alarm Door Open, Value: 1, Time: 1712622266

Signal: Tire Pressure, Value: 33, Time: 1712622266

Sending data to server...

Data sent:

Signal: Main Battery Voltage, Value: 381, Time: 1712622267

Signal: Main Battery Current, Value: 32, Time: 1712622267

Signal: Speed, Value: 15, Time: 1712622267

Signal: Main Battery Temperature, Value: 42, Time: 1712622267

Signal: Tire Pressure, Value: 34, Time: 1712622267

Alarmas de Problemas en el Vehiculo: alarm Tire Pressure, Value: 1, Time: 1712622267

# Gestión de Proyectos

## 1)Listas de tareas

A continuación, se detallan las listas de tareas para llevar adelante este desarrollo, teniendo en cuenta que el mismo fue solicitado por un cliente.

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente con confianza media

## 2)Diagrama de Gantt

El diagrama de Gantt se adjunta en el mail, para que se pueda apreciar mejor su visualización (Diagrama de Gantt\_Rodrigo Díaz Martucci.pdf)



## 3)Metromap

A continuación, se puede observar el Metromap, para identificar las áreas involucradas en la concepción de este producto/proyecto y sus relaciones.

Diagrama

Descripción generada automáticamente