VEMO

Enabling Clean Mobility.

Ingeniero Electrónico IoT

Contenido

Desafío de Entrevista: Ingeniero Electrónico IoT

Descripción del Desafío

Estamos buscando evaluar tus habilidades y experiencia en relación con las responsabilidades del puesto. Este desafío consiste en una serie de preguntas y ejercicios prácticos relacionados con tus conocimientos técnicos, habilidades de resolución de problemas y capacidad para trabajar en un entorno de desarrollo y operaciones de IoT.

Parte 1: Conocimientos Técnicos y Resolución de Problemas

Contexto

Se desea mejorar un software implementado en un dispositivo de telemetría vehicular para vehículos Tesla Model 3. El software base y sus SDK se encuentran adjuntos. Este software se utiliza para recopilar datos del vehículo mediante una variedad de sensores y enviar esta información a un servidor central para su análisis y gestión.

Con el objetivo de mejorar la seguridad y la eficiencia de la conducción, se requieren dos funcionalidades clave:

Alarmas de Conducción Peligrosa:

Para mejorar la performance de los conductores, se necesitan alarmas que alerten sobre situaciones de conducción peligrosa.

Alarmas de Problemas en el Vehículo:

Para mitigar problemas en el vehículo, se necesitan alarmas que identifiquen y notifiquen sobre posibles anomalías en el funcionamiento del vehículo.

Además, se busca optimizar la transmisión de datos para reducir costos y mejorar la eficiencia operativa.

Desafío

Tu tarea consiste en implementar las siguientes funcionalidades:

Alarmas de Conducción Peligrosa:

Analizar, diseñar e implementar alarmas que detecten situaciones de conducción peligrosa.

Alarmas de Problemas en el Vehículo:

Diseñar e implementar alarmas que identifiquen posibles problemas en el vehículo.

Optimización de señales:

Diseñar e implementar un algoritmo para optimizar la transmisión de datos al servidor, evitando el envío repetitivo de señales que no registran cambios.

Las alarmas que diseñes deberán ser incorporadas al diccionario de señales para ser transmitidas al servidor. Utilizarás las funciones getDataRow() y sendDataToServer() para obtener las señales del vehículo y enviarlas al servidor, respectivamente. Adjunto encontrarás un código para cada lenguaje, Elige el que tú quieras. Crea un proyecto en GIT con el enunciado y el código de tu elección. Vela por las buenas prácticas. Solo se pide desarrollar alarmas de tipo instantáneas, no es necesario desarrollar alarmas que utilicen datos históricos, en caso de diseñar alguna de este tipo, exponer la lógica.

Parte 2: Documentación y Buenas Prácticas

Después de completar el desarrollo de las funcionalidades, te solicitaremos que prepares un informe muy corto detallando los resultados obtenidos, incluyendo:

Resultados del algoritmo de optimización, criterios y categorías de las alarmas y evidencia de sus resultados.

Parte 3: Gestión de Proyectos

Por último, te pediremos que planifiques desde la solicitud de un cliente en implementar estas mejoras hasta la ejecución y validación del proyecto mediante:

1. Lista de tareas para verificar el correcto funcionamiento del desarrollo.
2. Diagrama de Gantt para visualizar la secuencia y duración estimada de las actividades.
3. Metromap para identificar las áreas involucradas en el proyecto y sus relaciones.

Notas Adicionales:

Durante la entrevista, estaremos observando tu capacidad para comunicar claramente tus ideas, tu enfoque para abordar los problemas técnicos y tu capacidad para trabajar de manera efectiva en equipo.

Siéntete libre de utilizar pizarras, papel o herramientas de colaboración en línea para ayudarte en la resolución de los problemas prácticos.

No te preocupes si no puedes completar todos los ejercicios en el tiempo asignado. Queremos evaluar tu proceso de pensamiento y enfoque para resolver problemas más que tu capacidad para completar tareas específicas en un corto período de tiempo.

¡Buena suerte!

Anexo

Código en Java

```
import java.util.Random;

public class Main {
  static final String[] signalDictionary = {
    "Main Battery Voltage",
    "Main Battery Current",
    "Speed",
    "Main Battery Temperature",
    "Door Open",
    "Tire Pressure"
  };

  static class Signal {
    int id;
    int val;

    Signal(int id, int val) {
      this.id = id;
      this.val = val;
    }
  }

  static final boolean DEBUG = true; // Set to true for debug mode, false for regular mode

  static void getDataRow(Signal[] signals) {
    Random rand = new Random();
    for (int i = 0; i < signalDictionary.length; i++) {
      int signalId = i + 1;
      int value;
      switch (signalId) {
        case 1:
          value = rand.nextInt(100) + 350;
          break;
        case 2:
```

```
            value = rand.nextInt(100) - 50;
            break;
          case 3:
            value = rand.nextInt(150);
            break;
          case 4:
            value = rand.nextInt(80) + 20;
            break;
          case 5:
            value = rand.nextInt(2);
            break;
          case 6:
            value = rand.nextInt(7) + 28;
            break;
          default:
            value = -1;
            break;
        }
        signals[i] = new Signal(signalId, value);
      }
    }

    static void sendDataToServer(Signal[] signals, long timestamp) {
      if (DEBUG) {
        System.out.println("\nSending data to server...");
        System.out.println("Data sent:");
      }
      for (int i = 0; i < signalDictionary.length; i++) {
        if (DEBUG) {
          System.out.printf("Signal: %s, Value: %d, Time: %d%n", signalDictionary[i],
signals[i].val, timestamp);
        } else {
          System.out.printf("%d,%d,%d%n", timestamp, signals[i].id, signals[i].val);
        }
      }
    }

    public static void main(String[] args) throws InterruptedException {
      final int DATA_PERIOD = 1; // 1 second
      while (true) {
        Signal[] signals = new Signal[signalDictionary.length];
```

```
        long timestamp = System.currentTimeMillis() / 1000;
        getDataRow(signals);
        sendDataToServer(signals, timestamp);
        Thread.sleep(DATA_PERIOD * 1000);
      }
    }
}
```

Código en C

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>

#define DEBUG 1 // Set to 1 for debug mode, 0 for regular mode

struct Signal {
  int id;
  int val;
};

const char *signalDictionary[] = {
  "Main Battery Voltage",
  "Main Battery Current",
  "Speed",
  "Main Battery Temperature",
  "Door Open",
  "Tire Pressure"
};

void getDataRow(struct Signal *signals) {
  srand(time(NULL));
  for (int i = 0; i < sizeof(signalDictionary) / sizeof(signalDictionary[0]); i++) {
    int signalId = i + 1;
    int value;
    switch (signalId) {
      case 1: value = rand() % 100 + 350; break;
      case 2: value = rand() % 100 - 50; break;
      case 3: value = rand() % 150; break;
      case 4: value = rand() % 80 + 20; break;
      case 5: value = rand() % 2; break;
      case 6: value = rand() % 7 + 28; break;
      default: value = -1; break;
    }
    signals[i].id = signalId;
```

```
      signals[i].val = value;
    }
}

void sendDataToServer(struct Signal *signals, long timestamp) {
    if (DEBUG){
      printf("\nSending data to server...\n");
      printf("Data sent:\n");
    }
    for (int i = 0; i < sizeof(signalDictionary) / sizeof(signalDictionary[0]); i++) {
      if (DEBUG) {
        printf("Signal: %s, Value: %d, Time: %ld\n", signalDictionary[i], signals[i].val,
timestamp);
      } else {
        printf("%ld,%d,%d\n", timestamp, signals[i].id, signals[i].val);
      }
    }
}

int main() {
    const int DATA_PERIOD = 1; // 1 second
    while (1) {
      struct Signal signals[sizeof(signalDictionary) / sizeof(signalDictionary[0])];
      long timestamp = time(NULL);
      getDataRow(signals);
      sendDataToServer(signals, timestamp);
      sleep(DATA_PERIOD);
    }
    return 0;
}
```

Código en C++

```
#include <iostream>
#include <ctime>
#include <unistd.h>

#define DEBUG 1 // Set to 1 for debug mode, 0 for regular mode

struct Signal {
  int id;
  int val;
};

const char *signalDictionary[] = {
  "Main Battery Voltage",
  "Main Battery Current",
  "Speed",
  "Main Battery Temperature",
  "Door Open",
  "Tire Pressure"
};

void getDataRow(Signal *signals) {
  srand(time(NULL));
  for (int i = 0; i < sizeof(signalDictionary) / sizeof(signalDictionary[0]); i++) {
    int signalId = i + 1;
    int value;
    switch (signalId) {
      case 1: value = rand() % 100 + 350; break;
      case 2: value = rand() % 100 - 50; break;
      case 3: value = rand() % 150; break;
      case 4: value = rand() % 80 + 20; break;
      case 5: value = rand() % 2; break;
      case 6: value = rand() % 7 + 28; break;
      default: value = -1; break;
    }
    signals[i].id = signalId;
    signals[i].val = value;
```

```
    }
}

void sendDataToServer(Signal *signals, long timestamp) {
    if (DEBUG){
        std::cout << "\nSending data to server...\n";
        std::cout << "Data sent:\n";
    }
    for (int i = 0; i < sizeof(signalDictionary) / sizeof(signalDictionary[0]); i++) {
        if (DEBUG) {
            std::cout << "Signal: " << signalDictionary[i] << ", Value: " << signals[i].val << ",
Time: " << timestamp << std::endl;
        } else {
            std::cout << timestamp << "," << signals[i].id << "," << signals[i].val << std::endl;
        }
    }
}

int main() {
    const int DATA_PERIOD = 1; // 1 second
    while (true) {
        Signal signals[sizeof(signalDictionary) / sizeof(signalDictionary[0])];
        long timestamp = time(NULL);
        getDataRow(signals);
        sendDataToServer(signals, timestamp);
        sleep(DATA_PERIOD);
    }
    return 0;
}
```

Código en Python

```
import random
import time
from time import sleep

signal_dictionary = [
  "Main Battery Voltage",
  "Main Battery Current",
  "Speed",
  "Main Battery Temperature",
  "Door Open",
  "Tire Pressure"
]

class Signal:
  def __init__(self, id, val):
    self.id = id
    self.val = val

debug = True  # Set to True for debug mode, False for regular mode

def get_data_row(signals):
  for i in range(len(signal_dictionary)):
    signal_id = i + 1
      value = 0
    if signal_id == 1:
      value = random.randint(350, 449)
    elif signal_id == 2:
      value = random.randint(-50, 49)
    elif signal_id == 3:
      value = random.randint(0, 149)
    elif signal_id == 4:
      value = random.randint(20, 99)
    elif signal_id == 5:
      value = random.randint(0, 1)
    elif signal_id == 6:
      value = random.randint(28, 34)
    signals[i] = Signal(signal_id, value)

def send_data_to_server(signals, timestamp):
```

```
    global debug
    if debug:
        print("\nSending data to server...")
        print("Data sent:")
    for i in range(len(signal_dictionary)):
        if debug:
            print(f"Signal: {signal_dictionary[i]}, Value: {signals[i].val}, Time: {timestamp}")
        else:
            print(f"{timestamp},{signals[i].id},{signals[i].val}")

def main():
    global debug
    data_period = 1  # 1 second
    while True:
        signals = [None] * len(signal_dictionary)
        timestamp = int(time.time())
        get_data_row(signals)
        send_data_to_server(signals, timestamp)
        sleep(data_period)

if __name__ == "__main__":
    main()
```

Technology

CONTÁCTANOS

Juan B. Alberdi 431, Olivos Buenos Aires - Argentina

E-mail: juan.cozzi@vemo.global

www.vemovilidad.com/