OVERVIEW

O teste consiste na implementação de um sistema em python que fará:

- Leitura de sensores
- Acionamentos
- Comunicação

Para este, será considerado a implementação fake destes sensores. Portanto, o desenvolvedor não terá esses sensores para a prova, portanto, deverá criar mecanismo de simulação dos mesmos.

A IMPLEMENTAÇÃO

O desenvolvedor precisará implementar:

- Leitura de sensores:

- Luminosidade
- Distância
- GPS
- Sensor de bateria 12V

- Acionamento:

- Ligar iluminação, quando luminosidade baixar;
- Desligar o sistema quando a bateria baixar de 10V;

- Comunicação:

- Estabelecer a conexão GPRS
- Ler dados do GPS e enviar a cada 30s a posição
- Enviar a posição, bateria, distância e luminosidade via REST;

- API:

- Criar API de comunicação

- Backend para Interface Local:

- Criar backend para as funcionalidades:
  * Ajuste de data e hora;
  * Cadastro do endereço/porta servidor para envio de informações
  * Busca de informações locais dos sensores

DETALHAMENTO TÉCNICO

Sensor de luminosidade: consiste de um sensor I2C que tem o seguinte protocolo
| ID | VALOR_LIDO | CHECKSUM |

Onde:
ID : um byte hexadecimal
VALOR_LIDO : um byte hexadecimal
CHECKSUM : valor hexadecimal, que é a soma do ID e VALOR_LIDO, se maior 255, fica apenas o byte inferior da soma;

Aqui o desenvolvedor deve criar array de mensagens, que simulam a o resultado da leitura (de 0 a 100) desse sensor, este array deve ser lido de forma crescente periodicamente;

Se o valor for menor que 40, deverá acionar o iluminador, que é uma GPIO do sistema.

Sensor de distância: consiste de um sensor I2C que tem o seguinte protocolo
| ID | VALOR_LIDO | CHECKSUM |

Onde:
ID : um byte hexadecimal
VALOR_LIDO : um byte hexadecimal
CHECKSUM : valor hexadecimal, que é a soma do ID e VALOR_LIDO, se maior 255, fica apenas o byte inferior da soma;

Aqui o desenvolvedor deve criar array de mensagens, que simulam a o resultado da leitura desse sensor, este array deve ser lido de forma crescente periodicamente;

GPS: é o sensor de posicionamento, possui comunicação serial e protocolo de comando AT.

Aqui o desenvolvedor precisará criar uma array das seguintes mensagens:

```
$GPRMC,140602.00,A,2736.12493,S,04834.61709,W,0.234,,010920,,,A*7E
$GPVTG,,,,,,,,,N*30
$GPGGA,140601.00,,,,,0,00,99.99,,,,,,*64
$GPGSV,3,1,10,05,34,266,,07,37,125,,08,14,133,18,09,34,049,33*79
$GPGLL,,,,,140601.00,V,N*48
```

A posição é dada no comando GPRMC nos parâmetros 2736.12493,S e 04834.61709,W. O desenvolvedor deverá extrair essas informações do pacote recebido, ou seja, lido aleatoriamente do array e enviar para o servidor;

Sensor de bateria: consiste de um sensor I2C que tem o seguinte protocolo
| ID | VALOR_LIDO | CHECKSUM |

Onde:
ID : um byte hexadecimal
VALOR_LIDO : um byte hexadecimal
CHECKSUM : valor hexadecimal, que é a soma do ID e VALOR_LIDO, se maior 255, fica apenas o byte inferior da soma;

Aqui o desenvolvedor deve criar array de mensagens, que simulam a o resultado da leitura desse sensor, este array deve ser lido de forma crescente periodicamente;

Caso o valor lido seja menor que 10V, o sistema deverá se desligar.

Modem GPRS: necessário para estabelecer a comunicação. Utiliza a comunicação serial. Para cada comando enviado, o sistema responde com OK ou ERROR. O desenvolvedor deverá enviar os comando, na ordem, de estabelecimento de conexão e verificar se foi OK.

```
ATE1
AT+COPS=2
AT+COPS=0
AT+CGDCONT=1,"IP","tim.com.br","",0,0
```

Startup do software: na inicialização do software, o mesmo deverá ler um arquivo, que conterá as configurações de portas seriais, endereço e porta de servidor.

AVALIAÇÃO

Será avaliado:
- Qualidade do código
- Commits pontuais
- Nomes de variáveis e funções
- Reutilização de código

- Compromisso com a entrega

DÚVIDAS:

Em caso de dúvida, fazer via issues.

SUGESTÕES

A DBA já utiliza algumas tecnologias, e que sugere ao desenvolvedor:

- Python3-twisted
- Python3-serial
- Python3-tornado