

# SMRP - Sistema de Monitoramento Remoto de Paciente

## Monitoramento remoto de sinais vitais em ambiente hospitalar

Alexandre José da Silva Júnior

Engenharia Eletrônica - Universidade de Brasília  
Sistemas Embarcados  
Gama - DF  
alexandre\_zy@outlook.com

Itiane Thayná Batista Almeida

Engenharia Eletrônica - Universidade de Brasília  
Sistemas Embarcados  
Gama - DF  
itiane.batista@gmail.com

**Resumo** - O presente relatório visa descrever as práticas utilizadas para a construção de um dispositivo eletrônico que busca otimizar a interação médico-paciente realizando a captura de determinados sinais do paciente e os enviando ao médico cadastrado, de forma que o profissional da saúde tenha total percepção sobre o estado do seu paciente

**Palavras-chave**— relação médico-paciente, dispositivo eletrônico, banco de dados, monitoramento de paciente)

### I. INTRODUÇÃO

A monitoração de alguns sinais biológicos é de suma importância para a detecção de enfermidades ou o tratamento das mesmas. Um dos dados que é de grande significância é a determinação do nível de saturação de oxigênio (SpO<sub>2</sub>) no sangue arterial, também conhecida como oximetria.

A coleta desse dado faz parte da monitorização padrão em unidades de terapia intensiva, centros cirúrgicos, áreas de recuperação, ambulâncias, dentre outros. [1]

A monitorização da SpO<sub>2</sub> fornece informação acerca dos sistemas cardíaco e respiratório e do transporte de oxigênio no organismo. É amplamente utilizada por ser não-invasiva, monitorar de maneira contínua, além de ser uma técnica simples e indolor. [2]

O acompanhamento desse sinal em tempo real e principalmente o conhecimento de seu comportamento com relação ao tempo traria um grande benefício tanto para o médico quanto para o paciente. O uso da tecnologia pode ser de grande ajuda nesses casos. Através de determinados sensores utilizados, o paciente pode ser monitorado a todo momento.

Por meio do monitoramento em tempo real o médico responsável, poderia atender outros enfermos sem descuidar do paciente monitorado e após certo período de tempo poderá visualizar a evolução do paciente por meio dos dados coletados anteriormente.

Diante disso, o SMRP integra, através de uma infraestrutura de eletrônica embarcada, aspectos relevantes ao monitoramento remoto da saúde. Sua base de funcionamento se encontra na figura 1.

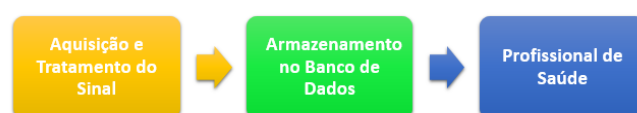


Figura 1: Processo do SMRP

Em suma o projeto consiste de um dispositivo que fará o monitoramento do paciente e enviará os dados coletados para um banco de dados onde o profissional conectado a mesma rede wireless do dispositivo e portando a senha terá acesso aos dados.

O banco de dados armazenará as informações coletadas, assim como as variações do sinal monitorado 24 horas por dia, e estará à disposição do profissional da saúde a todo momento por meio de qualquer celular ou computador conectado à mesma rede cabeada ou wireless que se encontra o aparelho, dessa forma o profissional poderá conduzir o tratamento mais adequado baseado no histórico do paciente.

### II. OBJETIVOS

O objetivo desse projeto é desenvolver um dispositivo que ofereça suporte no monitoramento de pacientes, aumentando sua qualidade de vida, uma vez que o médico terá todas as informações do monitoramento do paciente por meio de smartphone ou computador conectado à rede, e assim poderá obter um diagnóstico e tratamento mais preciso.

### III. DESENVOLVIMENTO

#### A. Justificativa

O projeto busca auxiliar o profissional de saúde no monitoramento dos pacientes, aumentando a eficiência do diagnóstico e tratamento, podendo monitorar um maior número de pacientes simultaneamente com qualidade. Uma vez que os oxímetros presentes no mercado não possuem o sistema de armazenamento de dados coletados, como também não possuem sistema de comunicação com outros dispositivos, como ilustrado na figura 2.

O SMRP traz ao profissional de saúde uma maior segurança no diagnóstico e tratamento do paciente, devido ao monitoramento em tempo real dos sinais, ofertando conforto e eficiência para ambas as partes.



Figura 2: Oxímetros comumente utilizados.

### B. Requisitos

Para o bom desenvolvimento do projeto são necessárias três grandes frentes, considerando a dependência entre as mesmas.

A primeira grande frente é a aquisição dos dados por meio de sistema de sensores que irão estar a todo tempo com o paciente em monitoramento.

A segunda grande frente é a comunicação entre sensores e um microcontrolador que fará a aquisição de dados e utilizando uma conexão I2C, se comunicará com a raspberry pi.

A terceira grande frente é a comunicação com a Raspberry pi. A Raspberry fará a aquisição dos dados do microcontrolador continuamente, 24 horas por dia e armazenará os dados coletados no servidor de banco de dados, que será a própria raspberry, onde ficará registrado o dado proveniente do sensor e a hora que o mesmo foi coletado.

O servidor de banco de dados será local e poderá ser acessado por meio de qualquer dispositivo conectado a mesma rede que o aparelho de monitoramento se encontra, contudo, o acesso possuirá senha, restringindo a entrada apenas aos seus portadores, garantindo maior autonomia dos serviços e maior segurança das informações pessoais do paciente.

### C. Hardware

Para a construção do circuito de oximetria e para o banco de dados com comunicação wireless serão necessários:

- 1 Diodo Emissor de Luz Vermelha (Vermelho)
  - 1 Diodo Emissor de Infravermelho
  - 2 LDR's
  - Resistores
  - Capacitores
  - 1 Amplificador Operacional LM324
  - 1 microcontrolador/microprocessador (Arduino)
  - Raspberry pi Zero
- LED (Diodo Emissor de Luz): O LED é um dispositivo capaz de emitir luz de forma eficiente e econômica. Será necessário um LED na cor vermelha (660nm) para

que se possa verificar absorvância do sangue com esse comprimento de onda.

- LDR (Light Dependent Resistor): Um resistor dependente de luz nada mais é que um resistor variável, cuja variação depende de da intensidade de luz que incide sobre ele. Neste projeto será utilizado apenas 1 LDR com o objetivo de captar a variação da absorção da luz vermelha pelo sangue.
- Resistores: componentes que têm por finalidade oferecer uma oposição à passagem de corrente elétrica, através de seu material. Aqui serão usados resistores para proteção do LED, LDR como também para a configuração de ganho do amplificador.
- Capacitor: Dispositivo composto por um conjunto de dois ou mais condutores isolados entre si por meio de dielétricos e que tem como função armazenar carga e energia elétrica no campo eletrostático que se estabelece entre os condutores. Nesse projeto seu uso se dará na configuração do filtro amplificador.
- Amplificador LM 324: Circuito integrado amplificador, com um ganho elevado, seu diferencial é possuir duas entradas, uma inversora negativa e outra não inversora positiva, a tensão de sua única saída é o resultado da diferença entre as entradas multiplicadas pelo ganho. O CI LM324 é composto por quatro amplificadores operacionais independentes de alto ganho destinados a operar a partir de uma única fonte de alimentação com uma ampla gama de tensões, ele será usado tanto na filtragem do sinal quanto na amplificação do mesmo.
- 1 microcontrolador/microprocessador: Define-se como um circuito integrado constituído por unidade de controle, registradores e unidade aritmética e lógica, capaz de obedecer a um conjunto predeterminado de instruções e de ser utilizado como unidade central de processamento. Nesse projeto será utilizado o Arduino, que é uma plataforma de prototipagem eletrônica de hardware livre e de placa única, projetada com um microcontrolador Atmel AVR. No SMRP, ele será utilizado para realizar a aquisição do sinal e prepará-lo para transmiti-lo.
- Raspberry Pi Zero: É um mini-microcomputador que abriga processador ARM, processador gráfico, slot para cartões de memória, interface USB, HDMI e seus respectivos controladores. Além disso, ele também apresenta memória RAM, entrada de energia e barramentos de expansão. O Raspberry será a peça chave para esse projeto, pois por meio dele se fará a recepção dos dados, sua alocação no banco de dados e a comunicação via wireless.

Os LED's, emissores de luz vermelha e infravermelha emitem comprimentos de onda na faixa de 660 a 940nm, tais diodos ao seres alocados em uma superfície com um bom fluxo sanguíneo promovem a passagem de luz que será mais ou menos absorvida pelo ldr. O microcontrolador Arduino irá funcionar como conversor analógico digital, capturando esse dado e o enviando pela porta serial para a

raspberry. O esquemático e a montagem em protoboard do circuito se encontra no Anexo I.

Implementado também um circuito de alerta, onde um LED piscará alertando os profissionais de saúde que os dados obtidos da oximetria não se encontram no intervalo aceitável.

O circuito de medição de oximetria em esquemático, projetado e testado no software Proteus está presente na figura de anexo I, e a figura onde se é mostrado o circuito implementado na pratica, também se encontra no anexo I.

#### D. Software

Parte do funcionamento do dispositivo se dá pelo processamento dos sinais adquiridos, esse processo será efetuado por um microcontrolador/microprocessador, no caso o Arduino.

A ligação do mesmo com a raspberry se dará pela porta serial, de modo que as informações cheguem rapidamente e sem perdas.

O Código do Arduino, utilizado no processamento dos sinais, se encontra no Anexo II.

A Raspberry irá receber os dados da entrada serial por meio de um código em C, que funciona como uma pasta, capturando os dados, criando um arquivo txt com os mesmos e os alocando em um banco de dados, que nesse caso é um servidor local feito com a própria raspberry.

Para inserção dos dados no banco de dados, foi utilizado o script do Linux que captura as informações que estão presentes na pasta /dev/ttyACM0 e o guarda em um arquivo .txt que possui a data no formato yyyy/mm/dd para melhor controle das informações. Como uma boa prática de banco de dados, os mesmos arquivos salvos no banco local, são sincronizados com o Google Drive, para que se tenha um backup em nuvem das informações recebidas, além de verificar os arquivos mais antigos e os apagar do arquivo local, evitando um possível esgotamento da memória.

Para utilização do banco de dados, usaremos Json, que transforma os dados recebidos em um objeto que pode é lido pelo código em JavaScript, HTML para o corpo da aplicação web e Javascript para a implementação do gráfico e valor atual dos dados do paciente, e informará o histórico do paciente.

Para o script foi utilizado as funções “cat”, “tee”, “mv”, “find” que concatenam, captura as saídas do terminal, renomeiam o arquivo e procuram o arquivo, respectivamente. O script também utiliza o pipe “|” que facilita a troca de informação de dois processos filhos, também é utilizado um script Gdrive para o update dos arquivos para o Google Drive.

Para o código em Json, é utilizado um modulo que está em sua primeira versão, o modulo SerialPort, um modulo que não se tem para o sistema operacional Raspbian, mas com algumas alterações, podemos utilizá-lo.

Para o código em javascript é utilizado a biblioteca Chart.js que implementa diversos gráficos em aplicações

web, juntamente com a biblioteca Socket.IO que permite a comunicação bidirecional entre cliente e servidor, com alta taxa de verificação de dados, apresentando os dados em tempo real.

Para a implementação em HTML foi utilizado um código simples, apenas para mostrar os resultados em JavaScript, o código ainda será melhorado para aumentar a segurança dos dados e para mostrar diferentes usuários cadastrados, o sistema simplificado está mostrado a seguir.

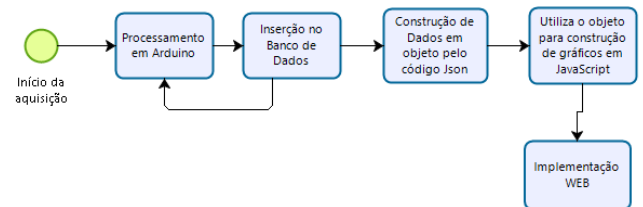


Figura 3: Diagrama de blocos simplificado do sistema.

#### E. Benefícios

Durante o monitoramento de um paciente, é importante para os médicos e enfermeiros estarem atentos as diversas funções corporais, para se certificar de que o paciente está seguro em todos os momentos. Esse dispositivo permite que os profissionais da saúde garantam que o nível de oxigênio do paciente não se torne demasiado baixo criando uma situação de risco.

Devido aos os pulsos de luz que são extremamente rápidos a leitura dos sinais é feita sem demora. Isso é importantíssimo em momentos críticos como em situação de emergência médica.

Além disso, o médico pode constatar o gráfico do histórico do paciente e verificar se a sua oxigenação do mesmo tem acontecido com qualidade ou se vem decaindo com o tempo.

## IV. RESULTADOS

Os resultados do Hardware podem ser vistos na figura comparativa de número 4, onde se tem a forma de onda esperada de um aparelho de oximetria industrial e a forma de onda que foi recebida pelo circuito.

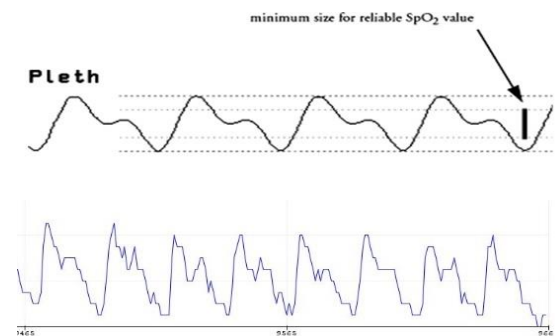
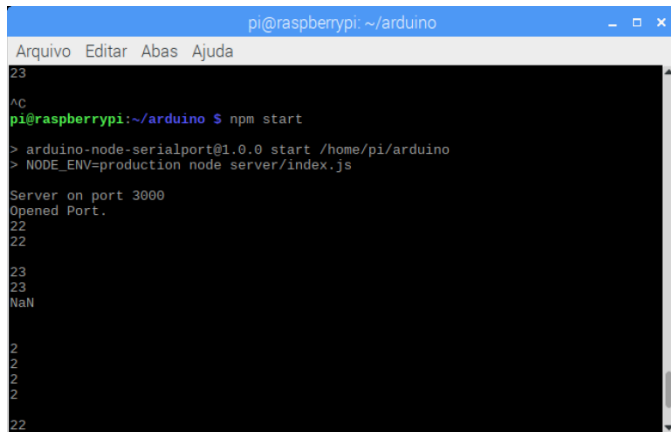


Figure 4: Formatos de ondas, comparativos.

Os resultados obtidos a partir dos métodos e aplicações utilizados podem ser vistos nas figuras 5 e 6, da aquisição de dados em tempo real e dos gráficos também em tempo real, respectivamente, a figuras estão a seguir.



```
pi@raspberrypi: ~/arduino
Arquivo Editar Abas Ajuda
23
Ac
pi@raspberrypi:~/arduino $ npm start
> arduino-node-serialport@1.0.0 start /home/pi/arduino
> NODE_ENV=production node server/index.js
Server on port 3000
Opened Port.
22
22
23
23
Null
2
2
2
2
22
```

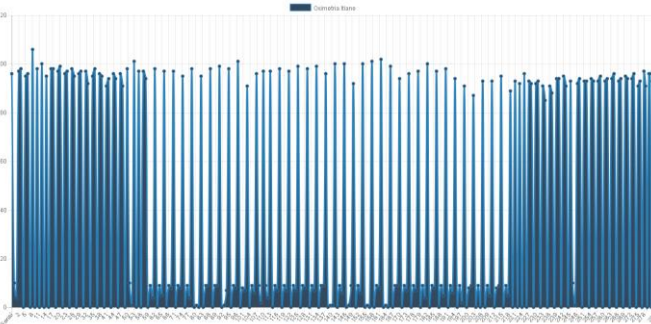


Figura 5: Aquisição de dados em tempo real.

Figura 6: Gráficos obtidos em tempo real.

Os dados obtidos a partir do sensor de oximetria dentro do padrão esperado, pois, o circuito ainda está em sua versão final, o processamento analógico foi feito com amplificador com alto fator de dessensibilidade e filtros analógicos.

## V. CONCLUSÃO

Em um conceito geral o sistema funcionou ao que lhe era esperado, pois o objetivo do mesmo é a análise remota dos dados, e esta análise já pode ser feita por pessoas que possuem o link do mesmo, caso a aplicação web possua um servidor ligado a um domínio pago. O problema que persistia na aquisição e processamento do sensor foi reparado e os resultados dos valores de oximetria foram verificados dentro do padrão internacional de saúde.

O projeto foi de grande ganho para o entendimento da disciplina, aprender e aplicar conceitos como pipes, sockets, processos entre outros e utilizar os conhecimentos para algo palpável que pode ser até mesmo utilizado comercialmente mostrou que não se trata de apenas mais créditos para a formação, mas sim de uma disciplina de caráter profissional.

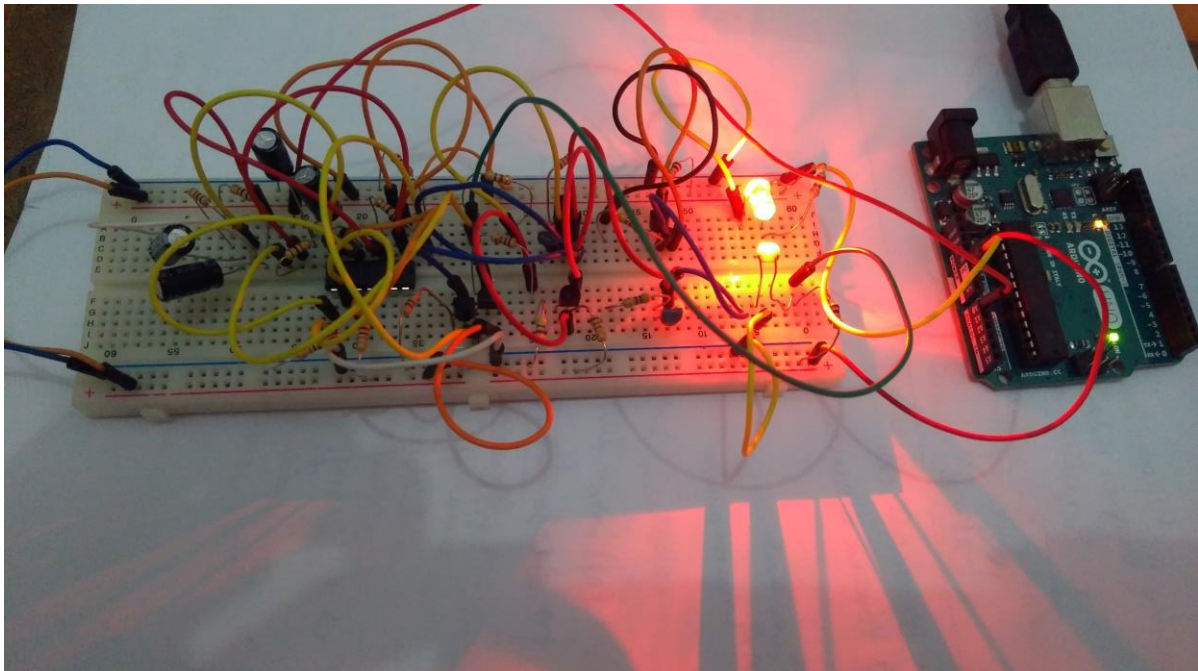
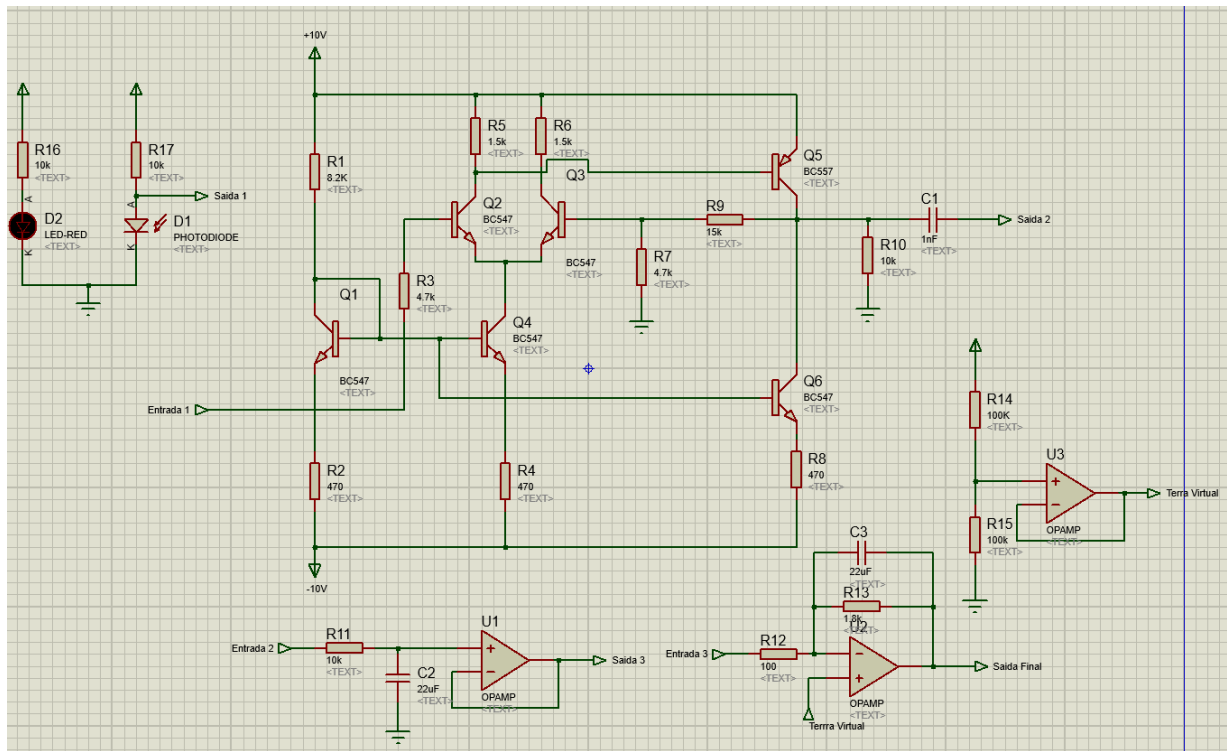
## REFERENCES

- [1] Alves, A. M. M, et al. OXIMETRIA DE PULSO: PRINCÍPIOS DE FUNCIONAMENTO E APLICAÇÕES. Revista Univap. São José dos Campos. 2016.
- [2] Carrara, D. et al. Oximetria de Pulso Arterial. Conselho Regional de Enfermagem de São Paulo. São Paulo, 2009.
- [3] Sergio T. Carvalho, et al. Monitoramento Remoto de Pacientes em Ambiente Domiciliar. Niteroi – RJ – Brasil.



## ANEXO I

### Esquemático do circuito de Oximetria



## **ANEXO II**

Código utilizado no microcontrolador Arduino, para aquisição dos sinais:

```
int leitura=A0;
int sinal=0;

void setup() {
  Serial.begin(9600);
  pinMode(leitura, INPUT);
}

void loop() {
  sinal=analogRead(leitura);
  Serial.println(sinal);
  if(sinal >= 700)
  {
    digitalWrite(9,HIGH);
  }
  if(sinal<=699)
  {
    digitalWrite(9,LOW);
  }
  delay(300);
}
```

### ANEXO III

Código em Json utilizado para transformar os dados captados em objetos:

```
{
  "name": "arduino-node-serialport",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "NODE_ENV=production node server/index.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.16.2",
    "serialport": "^6.0.4",
    "socket.io": "^2.0.4"
  }
}
```

## ANEXO IV

Código utilizado em JavaScript para utilizar o objeto criado e manipula-lo:

```
const path = require('path');

const express = require('express');
const http = require('http');
const SocketIo = require('socket.io');

const app = express();
const server = http.createServer(app);
const io = SocketIo.listen(server);

// settings
// routes
app.get('/', (req, res) => {
  res.sendFile(__dirname + '/index.html');
});

// static files
app.use(express.static(path.join(__dirname, 'public')));

const SerialPort = require('serialport');
const Readline = SerialPort.parsers.Readline;
const parser = new Readline();

const mySerial = new SerialPort('/dev/ttyACM0', {
  baudRate: 9600
});

mySerial.pipe(parser);

mySerial.on('open', function () {
  console.log('Opened Port.');
```

```
});
  console.log(parseInt(data));
  console.log(data.toString());
  io.emit('arduino:data', {
    value: data.toString()
  });
});
mySerial.on('err', function (data) {
  console.log(err.message);
});
server.listen(3000, () => {
  console.log('Server on port 3000');
```

```
});
```



## **ANEXO V**

Script utilizado para salvar os dados em arquivo local e na nuvem e utilizar a limpeza de arquivos antigos a cada 120 dias:

```
#!/bin/bash
# Pipe |
#Renomeia o arquivo para a data atual
#Remove arquivos com mais de 120 dias
#Adiciona os arquivos ao Drive
cat /dev/ttyACM0 | tee -a /home/pi/arduino/Banco/Dados.txt ;
mv Dados.txt Dados`date +%Y%m%d`.txt ;
find /home/pi/arduino/Banco -mtime +120 -exec rm {} \;
./gdrive-linux-rpi upload - /home/pi/arduino/Banco;
```

## ANEXO VI

Código utilizado em HTML para utilizar o código em JS para apresentá-lo em uma página web:

```
<!DOCTYPE
html>

<html>
  <head>
    <meta charset="utf-8">
    <title>Arduino Nodejs</title>
  </head>
  <body>
    <canvas id="myChart"></canvas>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.4.0/Chart.min.js"></script>
    <script src="/socket.io/socket.io.js" charset="utf-8"></script>
    <script type="text/javascript">
      const socket = io();
      var ctx = document.getElementById('myChart').getContext('2d');
      var chart = new Chart(ctx, {
        // The type of chart we want to create
        type: 'line',
        // The data for our dataset
        data: {
          labels: ["Serial"],
          datasets: [{
            label: "Serial Data from Arduino",
            backgroundColor: 'rgb(52, 73, 94)',
            borderColor: 'rgb(41, 128, 185)',
            data: [],
          }]
        },
        // Configuration options go here
        options: {}
      });
      let counter = 0 ;
      socket.on('arduino:data', function (dataSerial) {
        // console.log(dataSerial);
        chart.data.labels.push(counter);
        chart.data.datasets.forEach(dataset => {
          dataset.data.push(dataSerial.value);
        });
        counter++;
        chart.update();
      });
    </script>
  </body>
</html>
```