Relatório Técnico

Nº Grupo: 04
Nome dos integrantes: Ezequiel Ferreira, Guilherme Toledo, Camila Vitória, Matheus Zorzete, João Dorl, Daniel Costa
Turma: 1ADSB

Tema do projeto: Monitoramento de temperatura e umidade em criadouros de répteis
Sensor: LM35 (Temperatura) e DHT11 (temperatura e umidade)

Introdução

O projeto de medição de temperatura e umidade em criadouros de répteis tem como foco disponibilizar estas informações para as empresas que atuam nesta área, a fim de manter estes animais em condições saudáveis, diminuindo a mortalidade e conservando os níveis de fertilidade, reduzindo prejuízos financeiros.

Para fazer esta medição, utilizamos os sensores DHT11 – de temperatura e umidade – e LM35 – de temperatura. Visto que o sensor LM35 tem uma maior precisão, só utilizaremos o DHT11 para extração da umidade.

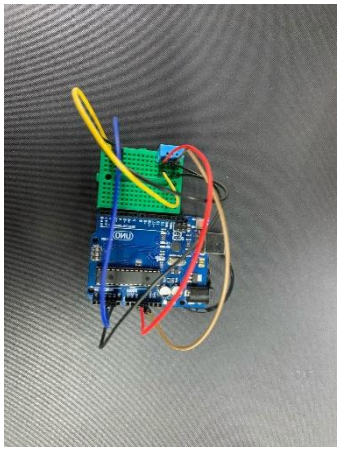
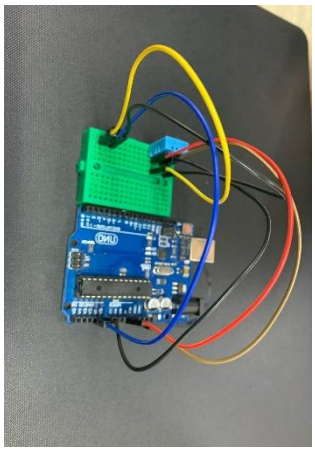
Por fim, as informações captadas serão tratadas e disponibilizadas em um dashboard para monitoramento dos usuários. Abaixo estão detalhes referentes à montagem do sensor e outras informações.

Arquitetura de Montagem do Sensor

No LM35, o primeiro pino da esquerda para direita é conectado diretamente ao slot de conexão de alimentação. O segundo pino é conectado à porta e por consequência o terceiro é conectado ao aterramento.

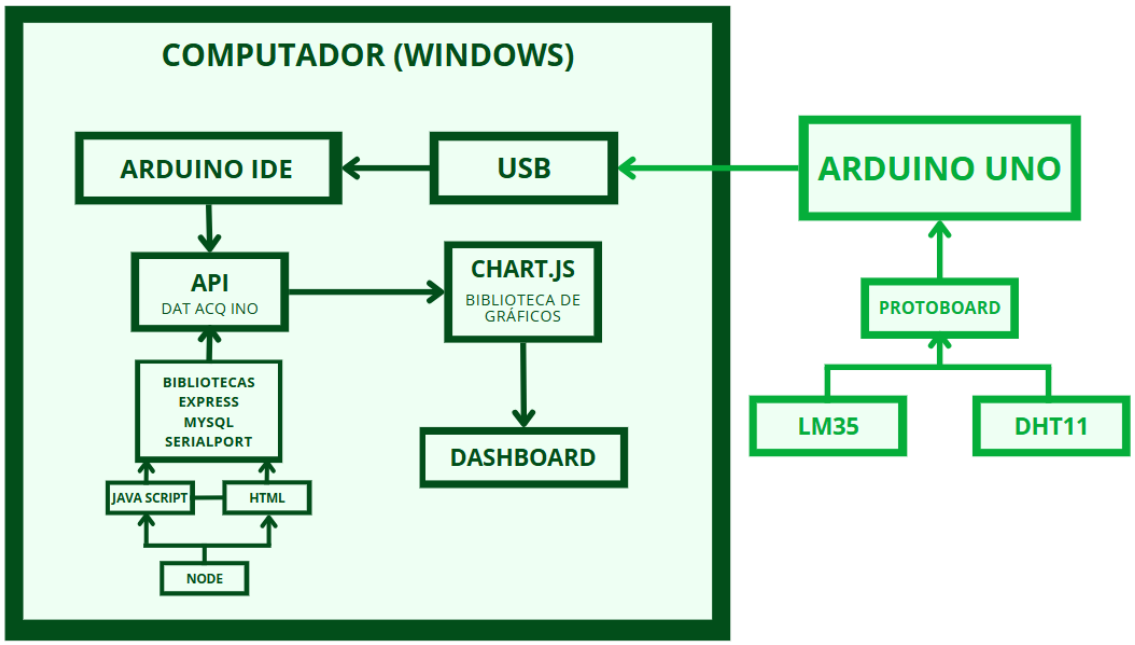
Da mesma forma, o DHT11 tem seu primeiro pino conectado a alimentação, o segundo na porta analógica, enquanto o terceiro não possui conexão e o quarto está conectado ao aterramento.

Abaixo está uma foto da arquitetura de montagem do projeto na mini protoboard, a imagem mostra como os sensores LM35 E DHT11 foram conectados ao Arduino Uno R3:

Arquitetura do Sistema

<ESCREVA AQUI NODE E BD>
<FOTOS E PRINTS AQUI>



Para captação e exibição dos dados no modelo atual, utilizamos os sensores LM35 e DHT11, conectados no microcontrolador Arduino Uno através de um protoboard. O Arduino Uno é conectado à porta USB do computador, e utilizados pela Arduino IDE. Na IDE, fazemos o tratamento dos dados utilizando a biblioteca do sensor DHT11, e tratando os dados para que fiquem no formato que precisamos (“umidade;temperatura”).

Estes dados serão utilizados na API DAT ACQ INO, que contém as bibliotecas Express (servidor web e expor dados), MySQL (comunicação com o banco de dados) e SerialPort (comunicação com a porta usb), gerenciadas pelo node.js, e que, através do Chart.JS, serão disponibilizadas em gráficos no navegador.

Código do Projeto

<ESCREVA AQUI DO ARDUINO E NODE>

<FOTOS E PRINTS AQUI>

Abaixo estão os prints do código utilizado para recebimento dos dados e criação do dashboard através do Chart.js. A licença para uso do código pertence à SPTech, devendo ser utilizado somente para fins de aprendizado:

main.js > ...

```
// Importar bibliotecas para fazer a conexão com o banco de dados, o
// arduino e a interface como constantes
const serialport = require('serialport');
const express = require('express');
const mysql = require('mysql2');

// Constantes para conexão e transmissão de dados
const SERIAL_BAUD_RATE = 9600;
const SERVIDOR_PORTA = 3300;

// habilita ou desabilita a inserção de dados no banco de dados (Deve
// ficar desbilitado até termos o banco de dados)
const HABILITAR_OPERACAO_INSERIR = false;

// função para comunicação serial, que busca funções criadas mais pra
// frente no código
const serial = async (
    valoresSensorAnalogico,
    valoresSensorDigital,
) => {

    // conexão com o banco de dados MySQL, sendo o principal a porta
    // correta do banco (Ainda não funciona devido a não ter o Banco de dados)
    let poolBancoDados = mysql.createPool(
        {
            host: 'HOST_DO_BANCO', //Colocar o nome do host (Conexão) do banco de dados
            user: 'USUARIO_DO_BANCO', //Colocar o usuário do banco de dados
            password: 'SENHA_DO_BANCO', //Colocar a senha (Caso tenha)
            database: 'DATABASE_DO_BANCO', //Dizer qual é o banco de dados
            port: 3306 //Porta do seu banco de dados
        }
    ).promise();

    // lista as portas seriais disponíveis e procura pelo Arduino
    const portas = await serialport.SerialPort.list();
    const portaArduino = portas.find((porta) => porta.vendorId == 2341 && porta.productId == 43);
    if (!portaArduino) {
        throw new Error('O arduino não foi encontrado em nenhuma porta serial');
    }
```

```
    // configura a porta serial com o baud rate especificado
    // E o atribuindo a constante arduino
    const arduino = new serialport.SerialPort(
        {
            path: portaArduino.path,
            baudRate: SERIAL_BAUD_RATE
        }
    );

    // evento quando a porta serial é aberta
    // E mostra no console a porta e o baud rate
    arduino.on('open', () => {
        console.log(`A leitura do arduino foi iniciada na porta ${portaArduino.path} utilizando Baud Rate de ${SERIAL_BAUD_RATE}`);
    });

    // processa os dados recebidos do Arduino
    // Separando os dois dados por um ;
    arduino.pipe(new serialport.ReadlineParser({ delimiter: '\r\n' })).on('data', async (data) => {
        console.log(data);
        const valores = data.split(';');
        const sensorDigital = parseInt(valores[0]);
        const sensorAnalogico = parseFloat(valores[1]);

        // armazena os valores dos sensores nos arrays correspondentes
        // E faz a ação de envia-lor para o terminal
        valoresSensorAnalogico.push(sensorAnalogico);
        valoresSensorDigital.push(sensorDigital);

        // insere os dados no banco de dados (se habilitado)
        // Dizendo que SE a constante abaixo for verdadeira (true) irá executar a ação abaixo
        if (HABILITAR_OPERACAO_INSERIR) {

            // este insert irá inserir os dados na tabela "medida"
            // É um código de mysql para insersão de dados
            await poolBancoDados.execute(
                'INSERT INTO medida (sensor_analogico, sensor_digital) VALUES (?, ?)',
                [sensorAnalogico, sensorDigital]
            );
            // Dizendo o que vai mostrar no console caso os valores sejam inseridos
            console.log("valores inseridos no banco: ", sensorAnalogico + ", " + sensorDigital);

        }

    });
    // evento para lidar com erros na comunicação serial
    // Mensagem vai aparecer no console
    arduino.on('error', (mensagem) => {
        console.error(`Erro no arduino (Mensagem: ${mensagem}`)
    });
}

// função para criar e configurar o servidor web
const servidor = (
    valoresSensorAnalogico,
    valoresSensorDigital
) => {
    const app = express();

    // configurações de requisição e resposta
    app.use((request, response, next) => {
        response.header('Access-Control-Allow-Origin', '*');
        response.header('Access-Control-Allow-Headers', 'Origin, Content-Type, Accept');
        next();
    });

    // inicia o servidor na porta especificada
    // Definida na variável anetrior SERVIDOR_PORTA
    app.listen(SERVIDOR_PORTA, () => {
        console.log(`API executada com sucesso na porta ${SERVIDOR_PORTA}`);
    });

    // define os endpoints da API para cada tipo de sensor
    app.get('/sensores/analogico', (_, response) => {
        return response.json(valoresSensorAnalogico);
    });
    app.get('/sensores/digital', (_, response) => {
        return response.json(valoresSensorDigital);
    });
}

// função principal assíncrona para iniciar a comunicação serial e o servidor web
// É "puxada" lá na parte de cima do código
(async () => {
    // arrays para armazenar os valores dos sensores
    const valoresSensorAnalogico = [];
    const valoresSensorDigital = [];
```

```
128
129     // inicia a comunicação serial
130     await serial(
131         valoresSensorAnalogico,
132         valoresSensorDigital
133     );
134
135     // inicia o servidor web
136     servidor(
137         valoresSensorAnalogico,
138         valoresSensorDigital
139     );
140 })();
141
```

Resultados Iniciais

A partir do tempo disponibilizado em aula para execução da atividade, pudemos entender o funcionamento da API, bem como da utilização do node para executar o código javascript sem precisar de um navegador. A experiência funcionou conforme esperado, e conseguimos receber os dados de captura obtidos através dos sensores. Entretanto, no momento de execução da atividade, não realizamos a conexão com o banco de dados – que também é possível através da API.

Durante a aula de Introdução a Sistemas Operacionais, prosseguimos no uso da API integrando os dados diretamente no banco de dados escolhido, o que também funcionou como esperado.

As duas aulas foram fundamentais para o entendimento do conceito de API, bem como entender a importância das bibliotecas, que abstraem códigos complexos que levariam tempo para serem criados. Portanto, os resultados foram satisfatórios, e o conteúdo aprendido foi de extrema importância para o desenvolvimento dos projetos das sprints 2 e 3.