Universidade Federal do Maranhão

Centro de Ciência Exatas e Tecnologia

Identificação do Projeto:

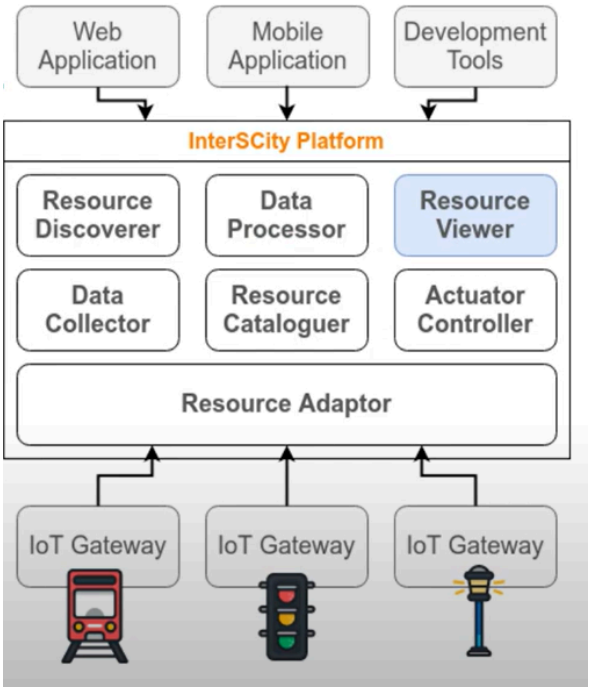
Sistema de Acompanhamento de Temperatura em Tempo Real com Alerta via Bluetooth e Wi-Fi

Discentes:

- Fernanda Sousa de Assunção Vale
- Isabel Silva de Araujo
- Leonardo Victor dos Santos Sá Menez
- Lucas Martins Campos Matos
- Vitor Ferreira Nunes

Estudo da plataforma InterSCity

Baseado em microserviços



1. Resource Adaptor

Responsavel pelo proxy do IOT

1. Registrar / atualizar recursos
2. Enivar dados pra plataforma

2. Resource Catalog

Responsavel por armazenar dados estáticos (metadados)

Criar UUIDs de cada recursos que vai ser utilizado

Notificação dos recursos na plataforma

3. Data Collector

Armazenar dados dos sensores

Permitir consultar os dados no atuais e anteriores

4. Resource Discoverer

Auxilia na descoberta de recursos através de filtros

Uso de filtros:

- Localização
- Tipo de capacidade
- Faixa de valores para uma dada capacidade

5. Actuator Controller

Responsável por gerenciar atuações

registra o webHook no sistema e a interSCity pode fazer o controle

Faz registros das atuações para auditoria

6. Data Processor

Analisar dados

Processamento em cluster

Uso de interface web para não usar Apache Spark

7. Resource Viewer

Apresentar visualizações dos dados

1. Tempo real
2. Históricos
3. Gráficos

Aplicação para nosso projeto

```
!pip install -q requests
```

```
import requests
import json
```

```
# Endereço para a api
api = 'https://cidadesinteligentes.lsdi.ufma.br/interscity_lh'
```

```
# teste de ip
!apt-get update
!apt-get install -y iputils-ping
```

Mostrar saída oculta

```
!ping -c 1 google.com
```

Mostrar saída oculta

```
!ping -c 1 playground.interscity.org
!ping -c 1 192.168.10.104
```

Mostrar saída oculta

Capacidade

```
# Cria uma 'capability'

# Playground - Resource Catalog - Post - Catalog capabilities

capability1_temp_json = {
  "name": "rooom/temperature",
  "description": "mede a temperatura em graus celsius (int)",
  "capability_type": "sensor"
}
r = requests.post(api+'/catalog/capabilities/', json=capability_temp_json)
if(r.status_code == 201):
  content = json.loads(r.text)
  print(json.dumps(content, indent=2, sort_keys=True))
else:
  print('Status code: '+str(r.status_code))
```

```
{
  "capability_type": "sensor",
  "description": "mede a temperatura em graus celsius (int)",
  "id": 30,
  "name": "rooom/temperature"
}
```

```
# Exibe as 'capabilities'

# Playground - Resource Catalog - Get - Catalog capabilities

r = requests.get(api+'/catalog/capabilities')
if(r.status_code == 200):
  content = json.loads(r.text)
  print(json.dumps(content, indent=2, sort_keys=True))
else:
  print('Status code: '+str(r.status_code))
```


```
{
  "capabilities": [
    {
      "description": "Capacidade de teste",
      "function": 0,
      "id": 9,
      "name": "teste1"
    },
    {
      "description": "Capacidade de teste",
      "function": 0,
      "id": 10,
      "name": "teste2"
    },
    {
      "description": "Vaga dispon\u00edvel ou ocupada",
      "function": 0,
      "id": 11,
      "name": "vagaA"
    },
    {
      "description": "Vaga dispon\u00edvel ou ocupada",
      "function": 0,
      "id": 12,
      "name": "vagaB"
    },
    {
      "description": "Vaga dispon\u00edvel ou ocupada",
      "function": 0,
      "id": 13,
      "name": "vagaC"
    },
    {
      "description": "Vaga dispon\u00edvel ou ocupada",
      "function": 0,
      "id": 14,
      "name": "vagaD"
    },
    {
      "description": "Mede qq coisa",
      "function": 0,
      "id": 15,
      "name": "NewVariable"
    },
    {
      "description": "Capacidade de teste",
      "function": 0,
      "id": 16,
      "name": "testeA"
    },
    {
      "description": "Capacidade de teste",
      "function": 0,
      "id": 17,
      "name": "testeB"
    },
    {
      "description": "Vaga dispon\u00edvel ou ocupada",
```


Recurso

```
# Cria um 'resource'

# Playground - Resource Catalog - Post - Catalog resource

resource_termometro01_json = {
  "data": {
    "description": "Sensor de Temperatura 01",
    "capabilities": ["rooom/temperature"],
    "status": "active",
    "collect_interval": 30,
    "lat": -3.559616,
    "lon": -6.731386
  }
}
r = requests.post(api+'/catalog/resources', json=resource_termometro01_json)
uuid = ''
if(r.status_code == 201):
  resource = json.loads(r.text)
  uuid = resource['data']['uuid']
  print(json.dumps(resource, indent=2))
else:
  print('Status code: '+str(r.status_code))
```

```
{
  "data": {
    "id": 7,
    "uri": null,
    "created_at": "2025-01-09T20:26:53.708Z",
    "updated_at": "2025-01-09T20:26:53.708Z",
    "lat": -3.559616,
    "lon": -6.731386,
    "status": "active",
    "collect_interval": 30,
    "description": "Sensor de Temperatura 01",
    "uuid": "31fdb4e8-8fdc-4704-9f83-b5d31ff0a444",
    "city": null,
    "neighborhood": null,
    "state": null,
    "postal_code": null,
    "country": null,
    "capabilities": [
      "rooom/temperature"
    ]
  }
}
```


```
# Exibe os 'resources'

# Playground - Resource Catalog - Get - Catalog resource

r = requests.get(api+'/catalog/resources')
if(r.status_code == 200):
  content = json.loads(r.text)
  print(json.dumps(content, indent=2, sort_keys=True))
else:
  print('Status code: '+str(r.status_code))
```

```
Status code: 404
```

Criando dados

```
# Adiciona dado da 'capability' ao 'resource'

# Playground - Resource Adaptor - Post - Adaptor resources

capability_data_json = {
  "data": [
    {
      "room/temperature": 25,
      "timestamp": "2024-03-26T23:03:23.609Z"
    },
    {
      "room/temperature": 26,
      "timestamp": "2020-11-26T17:32:25.428Z"
    }
  ]
}
r = requests.post(api+'/adaptor/resources/'+"31fdb4e8-8fdc-4704-9f83-b5d31ff0a444"+'/data/environment_monitoring', json=capability_data_json)
if(r.status_code == 201):
  print('Ok')
else:
  print('Status code: '+str(r.status_code))
```

```
Ok
```

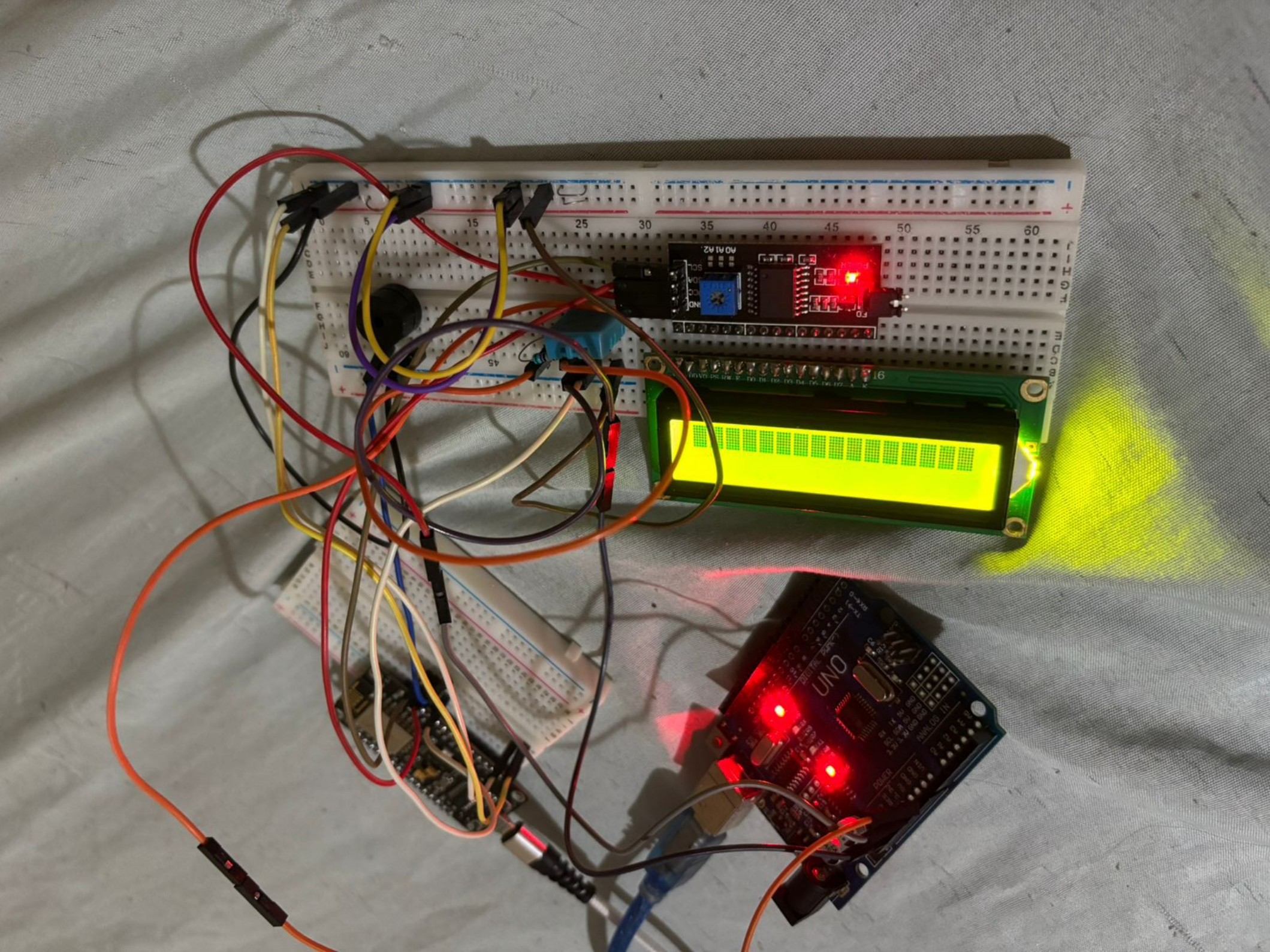
```
# Exibe dados do 'resource'

# Playground - Data collector - Post - Resources data

r = requests.post(api+'/collector/resources/'+"31fdb4e8-8fdc-4704-9f83-b5d31ff0a444"+'/data')
if(r.status_code == 200):
  content = json.loads(r.text)
  print(json.dumps(content, indent=2, sort_keys=True))
else:
  print('Status code: '+str(r.status_code))
```


```
{
  "resources": [
    {
      "capabilities": {
        "environment_monitoring": [
          {
            "date": "2024-03-26T23:03:23.609Z",
            "room/temperature": 25
          },
          {
            "date": "2020-11-26T17:32:25.428Z",
            "room/temperature": 26
          }
        ]
      },
      "uuid": "31fdb4e8-8fdc-4704-9f83-b5d31ff0a444"
    }
  ]
}
```

Após o teste inicial com dois dados, faremos o envio da série de dados rotulados completa através da API do InterSCity.


UNO

prototipo_copy_20250107150800.ino

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>


#define DHTPIN D4
#define DHTTYPE DHT11
#define BUZZER_PIN D3

DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  pinMode(BUZZER_PIN, OUTPUT);
  digitalWrite(BUZZER_PIN, LOW);

  Serial.begin(115200);
  dht.begin();

  lcd.init();
  lcd.backlight();
  lcd.print("Inicializando...");
  delay(2000);
  lcd.clear();
}

void loop() {
  float temperatura = dht.readTemperature();
  float umidade = dht.readHumidity();
```

```
30
31  if (isnan(temperatura) || isnan(umidade)) {
32    Serial.println("Falha na leitura do sensor!");
33    lcd.clear();
34    lcd.setCursor(0, 0);
35    lcd.print("Erro no sensor!");
36    delay(2000);
37    return;
38  }
39
40  Serial.print("Temp: ");
41  Serial.print(temperatura);
42  Serial.print(" C  Umidade: ");
43  Serial.print(umidade);
44  Serial.println(" %");
45
46  lcd.clear();
47  lcd.setCursor(0, 0);
48  lcd.print("Temp: ");
49  lcd.print(temperatura);
50  lcd.print(" C");
51
52  lcd.setCursor(0, 1);
53  lcd.print("Umid: ");
54  lcd.print(umidade);
55  lcd.print(" %");
```

```
51
52    lcd.setCursor(0, 1);
53    lcd.print("Umid: ");
54    lcd.print(umidade);
55    lcd.print(" %");
56
57    if (temperatura < 15 || temperatura >320) {
58      Serial.println("ALERTA! Temperatura fora do intervalo!");
59      lcd.setCursor(0, 1);
60      lcd.print("ALERTA!         ");
61      digitalWrite(BUZZER_PIN, HIGH);
62      delay(1000);
63      digitalWrite(BUZZER_PIN, LOW);
64    }
65
66    delay(5000);
67  }
```