PROJETO 02:

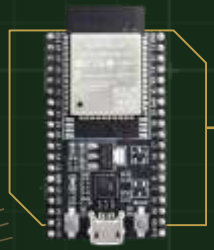
MONITORAMENTO DE BEBEDOURO

Grupo 04: Isabela Gomes, Jamilton Medeiros,
Matheus Vidal, Moisés Azevedo

DESCRIÇÃO DO PROJETO

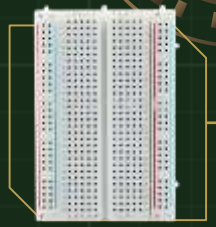
- Objetivo: monitorar um bebedouro de água dentro da UFRN;
- A Coisa (**bebedouro**) pode ser monitorada através de um microcontrolador **ESP32 DEVKIT V4** atrelado a dois sensores: **Vibração (SW-420) e Distância (HC-SR04)**;
- Detectar a troca de galão de água, e através do sensor de distância conseguimos detectar a presença de pessoas enchendo suas garrafas/copos;
- Dados transportados para internet via WiFi ao broker MQTT do AdaFruit;
- Raspberry foi utilizado para coletar os dados que foram enviados ao broker, e tratá-los, fazendo o cálculo da média de tempo em que o garrafão é trocado e enviando uma mensagem para o email do suporte do IMD;

DISPOSITIVOS



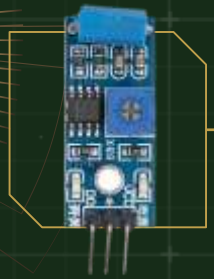
MICROCONTROLADOR

ESP32 DEVKIT V4



PROTOBOARD

PLACA PARA MONTAGEM DO CIRCUITO DO PROJETO



SENSOR DE VIBRAÇÃO

SENSOR SW-420 PARA DETECTAR A TROCA DE GALÃO

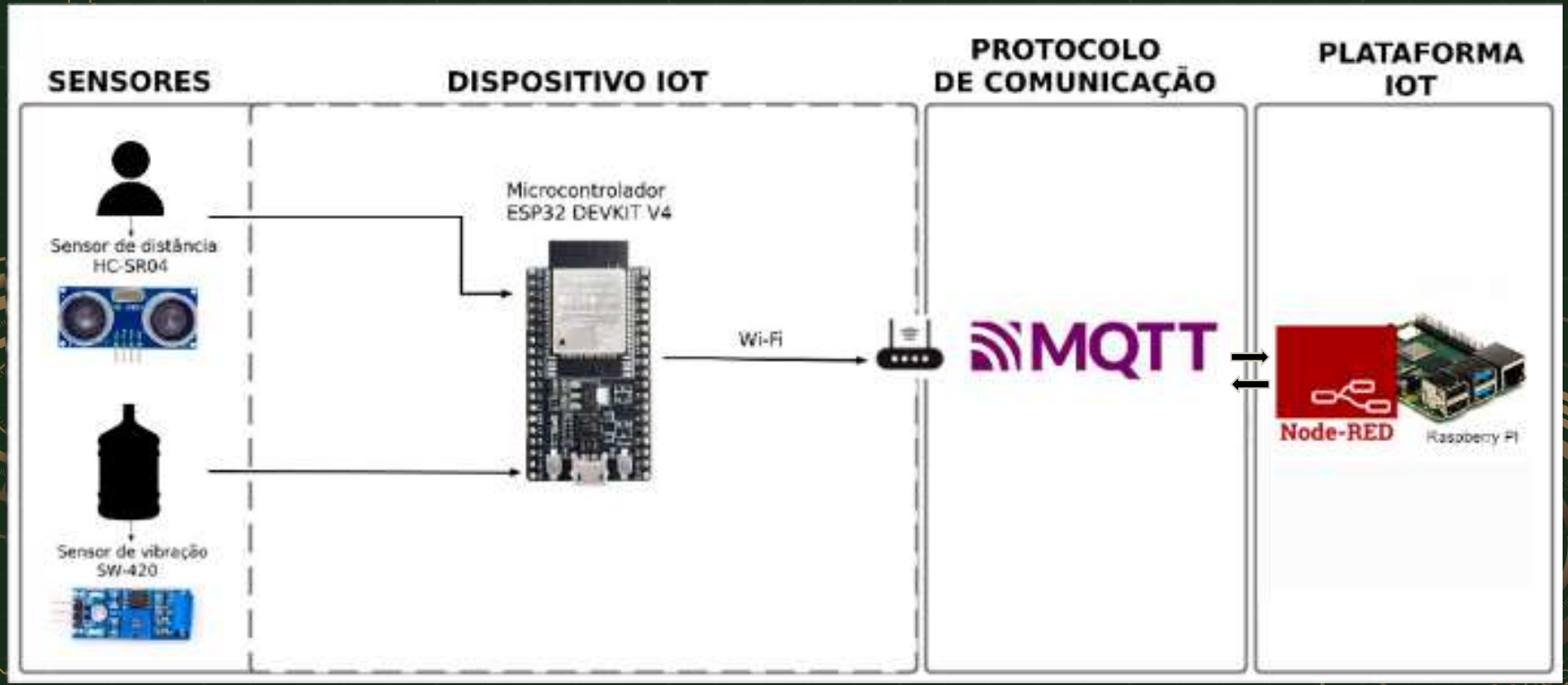
SENSOR DE DISTÂNCIA

SENSOR HC-SR04 PARA DETECTAR A PRESENÇA DE PESSOAS

RASPBERRY PI

MICRO-COMPUTADOR PARA RECEBER E ENVIAR DADOS

ARQUITETURA



COMUNICAÇÃO E ARMAZENAMENTO

ADAFRUIT

USO DO BROKER MQTT PARA COLETA DE INFORMAÇÕES E APRESENTAÇÃO DE DADOS NA PLATAFORMA

```
//Bibliotecas do projeto
#include <WiFi.h>
#include <WiFiClient.h>
#include <PubSubClient.h>
#include "AdafruitIO_WiFi.h"

WiFiClient ESPWiFiClient;
PubSubClient mqtt_client(ESPWiFiClient);


//Variáveis do WiFi e do MQTT
const char* WIFI_SSID = "projeto";
const char* WIFI_PASS = "projeto1";
int wifi_timeout = 200000;
const char* mqtt_broker = "io.adafruit.com";
const int mqtt_port = 1883;
int mqtt_timeout = 10000;

//Variáveis do AdaFruit
const char* mqtt_usernameAdafruitIO = "matheusvidal";
const char* mqtt_keyAdafruitIO = "aio_neqO81YtmxobWEabqYbmhdlVq0la";

//Configuração do AdaFruit
#if defined(USE_AIRLIFT) || defined(ADAFRUIT_METRO_M4_AIRLIFT_LITE) ||
    defined(ADAFRUIT_PYPORTAL)
// Configure the pins used for the ESP32 connection
#if !defined(SPIWIFI_SS) // if the wifi definition isnt in the board va
// Don't change the names of these #define's! they match the variant or
#define SPIWIFI SPI
#define SPIWIFI_SS 10 // Chip select pin
#define NINA_ACK 9    // a.k.a BUSY or READY pin
```

```
int tringPin = 13;
int echoPin = 12;
int pinoVibracao = 34;

long duracaoPulso;
float distanciaCm;
#define SOUND_SPEED 0.034

//Conectando o MQTT
void connectMQTT() {
  unsigned long tempoInicial = millis();
  while   (!mqtt_client.connected() && (millis() - tempoInicial < mqtt_ti
    if (WiFi.status() != WL_CONNECTED) {
      conecteWiFi();
    }
    Serial.print("Conectando ao MQTT Broker..");

    if (mqtt_client.connect("ESP32Client", mqtt_usernameAdafruitIO, mqtt_
      Serial.println();
      Serial.print("Conectado ao broker MQTT!\n");
    } else {
      Serial.println();
      Serial.print("Conexão com o broker MQTT falhou!");
      delay(1000);
    }
  }
  Serial.println();
}
```

```
void conecteWiFi(){
  WiFi.mode(WIFI_STA); //"station mode": permite o ESP32 ser um cliente da re
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  Serial.print("Conectando à rede WiFi .. ");

  unsigned long startTime = millis();

  while(WiFi.status() != WL_CONNECTED && (millis() - startTime < wifi_timeout
    Serial.print(".");
    delay(100);
  }
  Serial.println();

  if(WiFi.status() != WL_CONNECTED){
    Serial.println("Falhou!");
  }else{
    Serial.print("Conectado com o IP: ");
    Serial.println(WiFi.localIP());
  }
}


//Configurando o setup
void setup() {
  Serial.begin(115200);
  pinMode(tringPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(pinoVibracao, INPUT);
  conecteWiFi();
  if(WiFi.status() == WL_CONNECTED){
    Serial.println("Conectando ao broker MQTT...\n");
    mqtt_client.setServer(mqtt_broker, mqtt_port);
  }
```

```
void loop() {
  if (!mqtt_client.connected()){
    connectMQTT();
  }

  //Recebendo o valor do sensor de vibração
  long measurement = vibration();

  //Envio de sinal de tring
  digitalWrite(tringPin, LOW);
  delayMicroseconds(2);
  //Seta o pino Tring alto por 10ms
  digitalWrite(tringPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(tringPin, LOW);

  //Calcular o tempo que o pino fica alto em MS
  duracaoPulso = pulseIn(echoPin, HIGH);

  //Calcular a distância
  distanciaCm = (duracaoPulso * SOUND_SPEED) / 2;
  Serial.print("A distância é:" + String(distanciaCm));
  Serial.println("\n");

  if (mqtt_client.connected()){
    mqtt_client.loop();

    //Publicação da presença
    mqtt_client.publish("matheusvidal/feeds/presenca", String(distanciaCm).c_str(), t
    Serial.println("Publicou o dado da presença: " + String(distanciaCm) + "\n");

    if(measurement > 0){
```

```
140
141       if(measurement > 0){
142         Serial.println("Está vibrando em: " + String(measurement));
143       }
144       else{
145         Serial.println("Não está vibrando");
146       }
147
148       //Publicação da vibração
149       mqtt_client.publish("matheusvidal/feeds/vibracao", String(measurement).c_str(), true + "\n");
150       Serial.println("Publicou o dado da vibração: " + String(measurement));
151       delay(900);
152     }
153     delay(900);
154   }
155
156   //Função para receber a vibração
157   long vibration(){
158     long measurement = pulseIn (pinoVibracao, HIGH);
159     return measurement;
160   }
```

MONITORAMENTO

DADOS COLETADOS

1

Troca de galão às 21h47

90

Pessoas que tomaram água

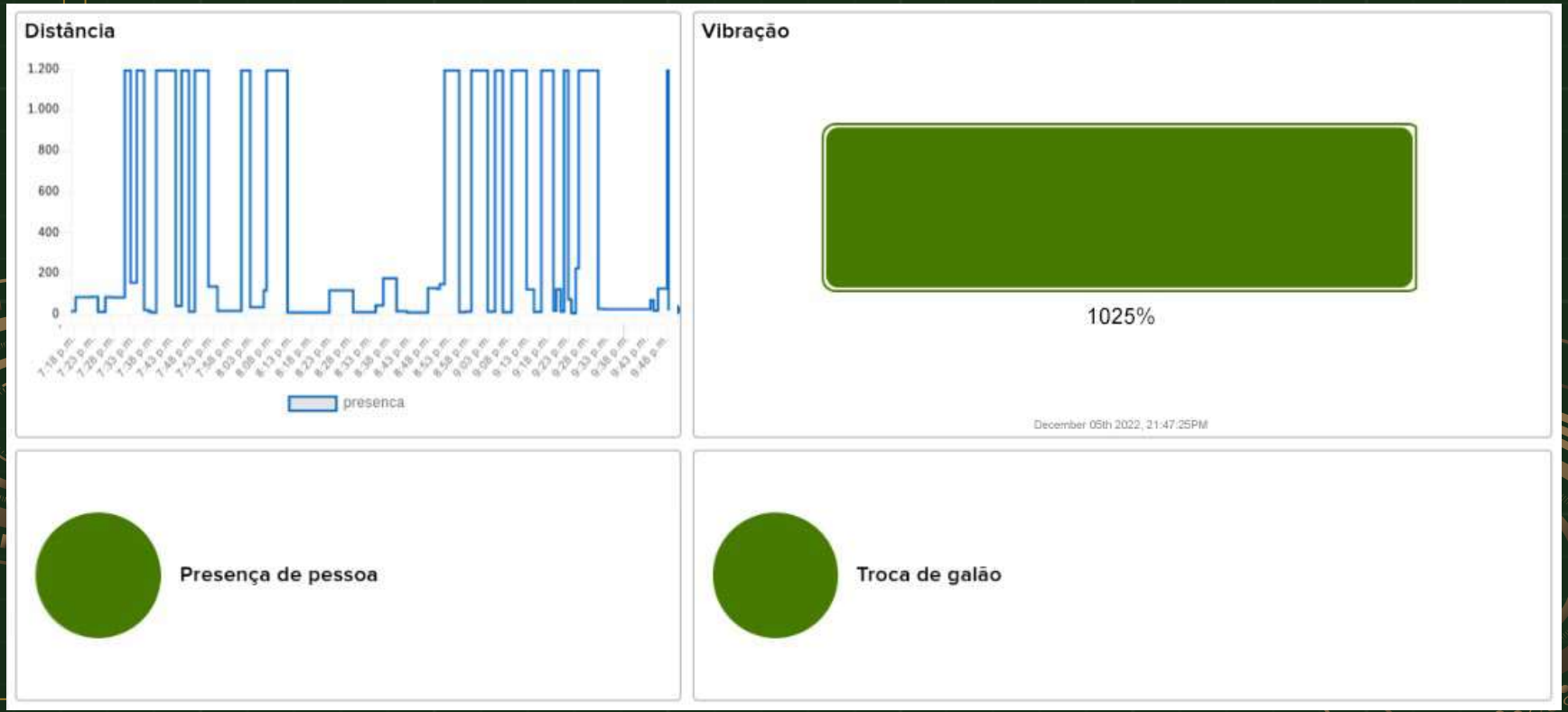
Pico

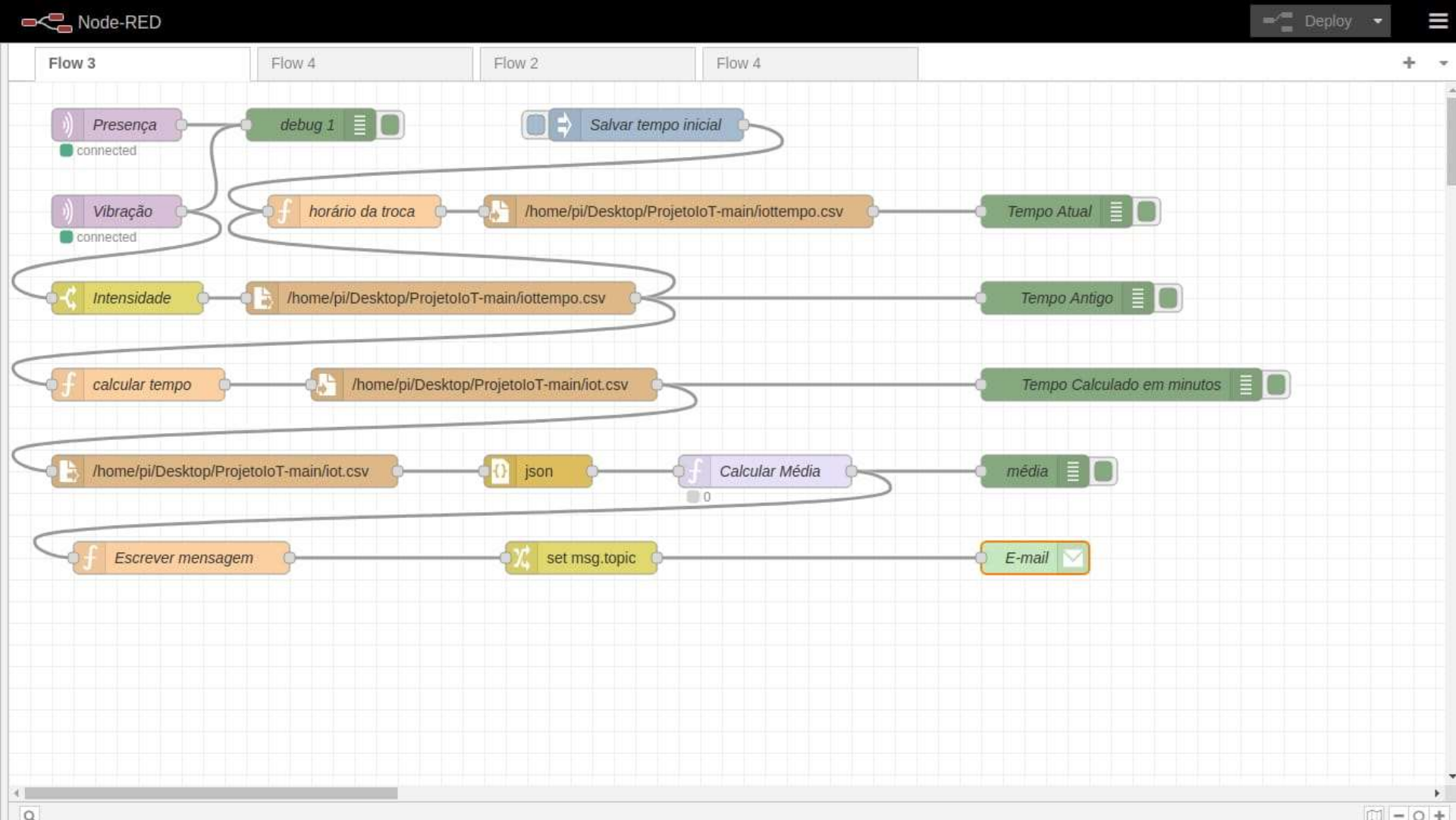
20h10 às 20h35

90 min

Média de tempo para trocar o galão

ADAFRUIT IO


Distância
1.200
1.000
800
600
400
200
0
presenca
Vibração
1025%
December 05th 2022, 21:47:25PM
Presença de pessoa
Troca de galão


Node-RED
Deploy
Flow 3
Flow 4
Flow 2
Flow 4
Presença
connected
debug 1
Salvar tempo inicial
Vibração
connected
horário da troca
/home/pi/Desktop/ProjetoIoT-main/iottempo.csv
Tempo Atual
Intensidade
/home/pi/Desktop/ProjetoIoT-main/iottempo.csv
Tempo Antigo
calcular tempo
/home/pi/Desktop/ProjetoIoT-main/iot.csv
Tempo Calculado em minutos
/home/pi/Desktop/ProjetoIoT-main/iot.csv
json
Calcular Média
0
média
Escrever mensagem
set msg.topic
E-mail

Setup | On Start | **On Message**

```
1  var now = new Date().toLocaleString("br-BR");
2  return { payload: now };
```

Setup | On Start | **On Message** | On Stop

```
1  var tempoPassado = new Date(msg.payload).toLocaleString("br-BR");
2  var tempoAtual = new Date().toLocaleString("br-BR");
3  let tempoTotal = getMinutes(tempoAtual, tempoPassado);
4
5  return { payload: tempoTotal};
6
7  function getMinutes(data1, data2) {
8      data1 = new Date(data1);
9      data2 = new Date(data2);
10     let segundos1 = data1.getSeconds();
11     let minutos1 = data1.getMinutes();
12     let horas1 = data1.getHours();
13
14     let segundos2 = data2.getSeconds();
15     let minutos2 = data2.getMinutes();
16     let horas2 = data2.getHours();
17
18     let tempo1 = (horas1*60)+(minutos1)+(segundos1/60);
19     let tempo2 = (horas2*60)+(minutos2)+(segundos2/60);
20
21     let total = tempo1 - tempo2;
22     return total
23 }
```

Name: Escrever mensagem

Setup | On Start | **On Message** | On Stop

```
1
2  let tempo = msg.payload;
3
4  let mensagem = "tempo";
5
6  return { payload: ("O próximo garrafão de água mineral deverá ser trocado em " + tempo + " minutos.")};
7
8
```

Enabled

MENSAGEM NO EMAIL

Dúvidas?

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**