SImbolo_M  **UNIVERSIDADE PRESBITERIANA MACKENZIE**

**Faculdade de Computação e Informática**

**Rastreador de bagagens em tempo real: um exemplo utilizando Arduino e Internet das Coisas**

**Bruna Amorim, Wallace Santana**

Universidade Presbiteriana Mackenzie (UPM)

Rua da Consolação, 930 Consolação, São Paulo-SP,01302-907-Brazil

[**brunacarlosamorim@outlook.com**](mailto:brunacarlosamorim@outlook.com)

**Abstract.** This work proposes a device to track back luggages using Arduino as hardware, sending data through internet of things frameworks and viewing results in a mobile application

**Resumo**. Este trabalho propõe a utilização de um dispositivo para rastreamento de bagagens utilizando Arduino como hardware e envio de dados através de Internet das coisas e disponibilização visual em um aplicativo para celular

**Palavras chaves:** Arduino, Internet das coisas, rastreamento bagagem

# Introdução

# Com alta temporada de férias e a retomada de turismo, o número de viagens teve um grande crescimento por esses meses entre junho, julho, dezembro e janeiro, muitas empresas de transporte aéreo vem enfrentando os desafios operacionais de grandes proporções.

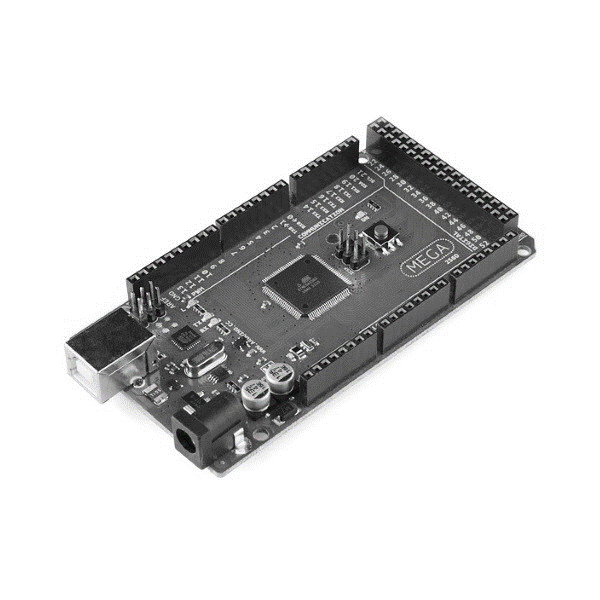
# Com isso um dos maiores problemas é o transporte de bagagem e o acompanhamento das mesmas pelos passageiros e funcionários dessas empresas, gerando dificuldades e custos na operação e acarretando em insatisfação dos clientes em caso de erro.

# Como muitas dessas empresas possuem aplicativos para check-in, emissão de cartões de embarque, reserva de passagens, o presente trabalho propõe a introdução de uma funcionalidade que permite o acompanhamento das bagagens pelo passageiro deste o momento do despacho, conexões e até a devolução no destino final. Utilizando rastreador, sensor biométricos e atuador de som, dificultando que sua bagagem seja extraviada.

# Materiais e Métodos

Para o desenvolvimento deste projeto foram utilizados diversos hardware e software, que serão apresentados nas seções seguintes.

**2.1. Modulo Arduino Mega, Sensor e Atuador**

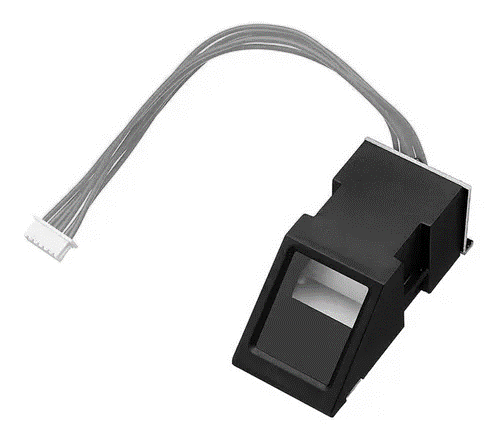


**Figura 1. Arduino Mega**

A Mega é baseada no microcontrolador ATmega2560 e possui 70 pinos – são múltiplas as opções de conexão e criação. Tem um número de entradas e saídas bem maior que a Uno R3, mas também é compatível com a maioria dos shields projetados para a placa irmã: basta conectar sua Mega a um computador com um cabo USB, uma fonte externa DC chaveada de 9 V com plug P4 ou bateria

Especificações do Arduino Mega 2560 R3:

* Microcontrolador: ATmega2560
* Velocidade do Clock: 16 MHz
* Pinos I/O Digitais: 70 (15 podem ser usadas como PWM)
* Portas Analógica: 16
* Tensão de Operação: 5 V
* Tensão de Alimentação:  7-12 V
* Corrente Máxima Pinos I/O: 40 mA
* Memória Flash: 256 KB (8KB usado no bootloader)
* SRAM: 8 KB
* EEPROM: 4 KB
* Dimensões: 101,6 mm x 53,34 mm
  1. **Sensor Biométrico**



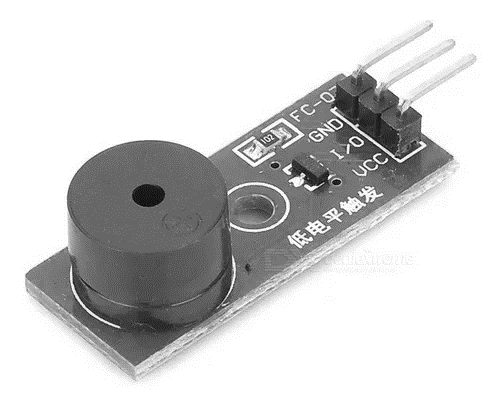
**Figura 2. Leitor Biométrico**

O Leitor Biométrico DY50 é um sensor de impressão digital óptico desenvolvido especialmente para a criação de projetos de automação residencial e sistema de controle de acesso.

Para que entre em operação é necessário integrar o Leitor Biométrico em uma plataforma de prototipagem, entre elas, o Arduino ou Raspberry Pi. Quando em funcionamento o Leitor Biométrico possui a capacidade de salvar uma diversidade muito grande de impressões dentro de sua memória flash onboard, chegando a gravar até 162 impressões digitais.

Especificações:

* Tensão de alimentação: DC 3.3 - 5V
* Corrente de alimentação: < 120mA
* Pico de corrente: <140mA
* Tempo de imagem da impressão digital: <1.0 segundos
* Dimensões da janela: 14 x 18 mm
* Modo de correspondência: modo de correspondência (1:1)
* Modo de pesquisa (1:N)
* Arquivo principal: 256 bytes
* Arquivo de modelos: 512 bytes
* Capacidade de armazenamento: 150
* Nível de segurança: cinco (do mais baixo para o mais alto: 1, 2, 3, 4, 5)
* Taxa de falsa aceitação (FAR): < 0,001% (nível de segurança 3)
* Taxa de falsa rejeição (FRR): < 1,0% (nível de segurança 3)
* Tempo de pesquisa: < 1,0 segundos (1:500, a média)
* Interface PC: UART (nível lógico TTL) ou USB2.0/USB1.1
* Comunicação baud rate (UART): (9600 x N) bps onde N = 1 ~ 12 (valor padrão N = 6, ie 57600bps)
* Ambiente de funcionamento:  
  Temperatura: -20°C - +50°C  
  Humidade relativa: 40% RH - 85% RH (sem condensação)
* Ambiente de armazenamento:  
  Temperatura: -40°C - +85°C  
  Humidade relativa: < 85% H (sem condensação)
* Dimensões (C x L x A): 56 x 20 x 21.5mm
  1. **Módulo Buzzer Passivo 5V**



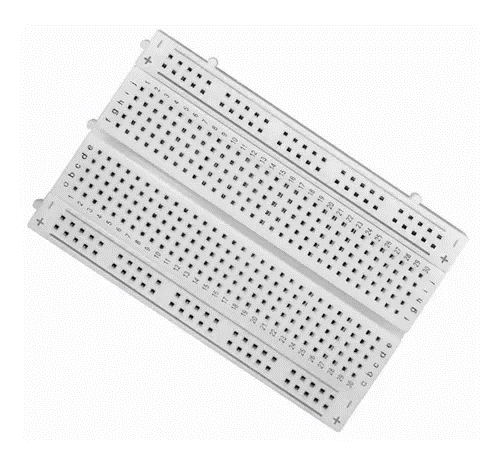
**Figura 3. Módulo Buzzer Passivo 5V**

O Módulo de Buzzer Passivo é um pequeno alto-falante destinado a emitir sinais sonoros a partir do oferecimento de diferentes frequências.O Buzzer é uma estrutura simplificada e integrada de transdutores eletrônicos, muito utilizados em alarmes, impressoras, computadores e projetos robóticos.

A principal finalidade do Módulo de Buzzer é a emissão de sinais sonoros como forma de alerta para que o operador fique informado que algo esta ocorrendo. O acionamento do buzzer dá-se através da placa microcontroladora, que deverá estar programada para diante de determinado acontecimento oferecer diferentes frequências ao buzzer, que dará sinais de aviso ao operador.

O Módulo de Buzzer Passivo é compatível com a maioria dos sistemas microcontroladores, dentre estes, Arduino, AVR, PIC, AMR. etc. Indicado para utilização por estudantes e profissionais o Módulo Buzzer é de fácil aplicação, atuando em conjunto com a placa microcontroladora, sendo muito confiável e eficiente.

* 1. **Breadboard Protoboard 400 Pontos Furos Pinos**



**Figura 4. Breadboard Protoboard 400 Pontos Furos Pinos**

O protoboard permite que você monte seu circuito de uma maneira mais rápida, prática e fácil. O produto contém em sua superfície 400 furos e do outro lado um adesivo que permite colá-lo em qualquer lugar.

Especificações:

* Quantidade de pontos: 400
* Material base: ABS
* Material conexão: Bronze banhado com níquel
* Diâmetro do furo: 0,8mm²
* Possui 2 barramentos laterais interligados
* Dimensões: 83 x 55 x 10 mm
  1. **Protocolo MQTT**

O protocolo MQTT é principalmente utilizado em aplicações de IOT, devido a sua simplicidade e facilidade de implementação. Além de aplicações de IOT, alguns usos muito comuns são para a obtenção de dados em tempo real. Ele é um protocolo de troca de mensagens entre maquinas. Utilizarei linguagem Java essa linguagem tem biblioteca MQTT que é criada pelo mosquito.

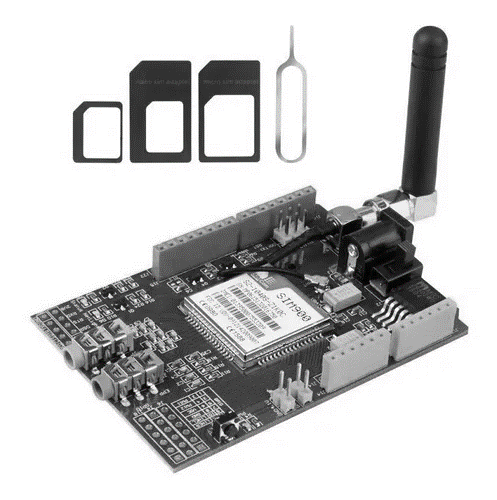
* 1. **Node-RED**

Node-RED é uma ferramenta voltada para o desenvolvimento de aplicações relacionadas ao conceito de Internet de Coisas (IOT) simplifica o desenvolvimento ligando o bloco de notas.

Este instrumento de programação foi desenvolvido para conectar gadgets de equipamentos (Hardware), APIs e administrações online de maneiras novas e intrigantes.

* 1. **Modulo GSM**

O módulo GSM habilita conexão via rede GSM ao módulo Arduino, permite o envio de SMS, MMS, GPRS e Áudio via UART através do envio de comandos AT.

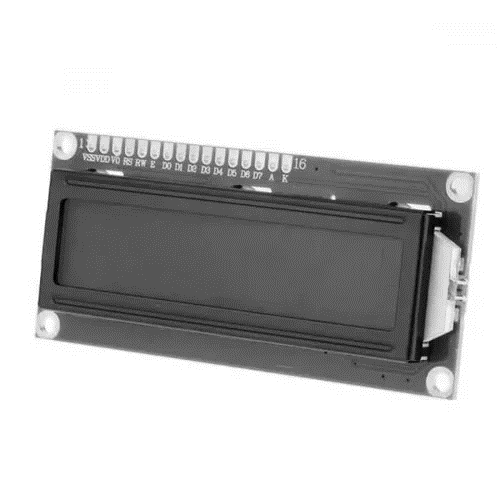
****

**Figura 5. Módulo GSM SIM**

Especificações:

* Tensão de operação: 5V
* Peso: 70 gramas
  1. **Tela LCD**

A Tela LCD exibe as informações e permite acompanhar o estágio de execução do programa Arduino.



**Figura 6. Tela LCD 16X2**

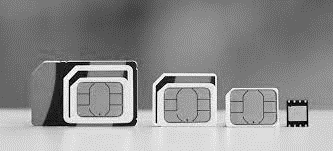
Especificações:

* LCD 16X2
* Cor de luz de fundo: azul
* Quantidade de linhas: 2
* Quantidade de caracteres: 32
* Altura total: 8 mm
* Largura total: 36 mm
* Comprimento total: 80 mm
  1. **Rede GSM**

A rede GSM dispõe de uma arquitetura abrangente que permite a integração de componentes de diversos fabricantes com o intuito de disponibilizar redes para aplicações implementadas em dispositivos móveis como celulares.

Ela facilita a criação de soluções versáteis com baixo custo e de aderência aos meios de comunicação mais utilizados no momento, simplificando a adoção de soluções como a apresentada no presente artigo.

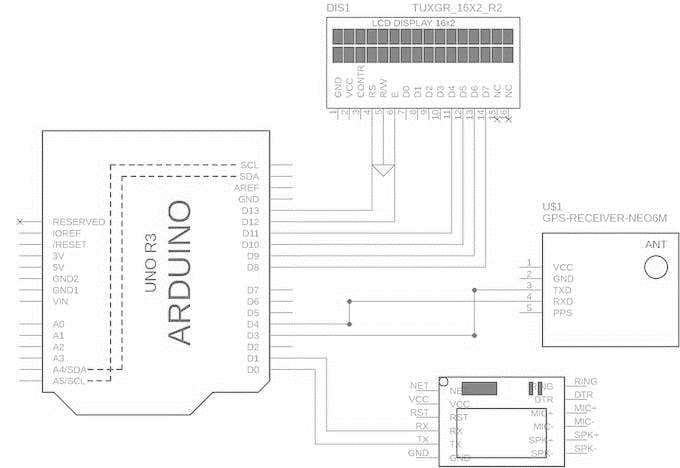
O emprego desta rede foi realizado por meio do módulo GSM através de um cartão SIM, também conhecido como chip, que é um circuito impresso e inteligente que permite identificar, controlar e armazenar dados de celulares que utilizem a rede GSM.



**Figura 7. Cartão SIM**

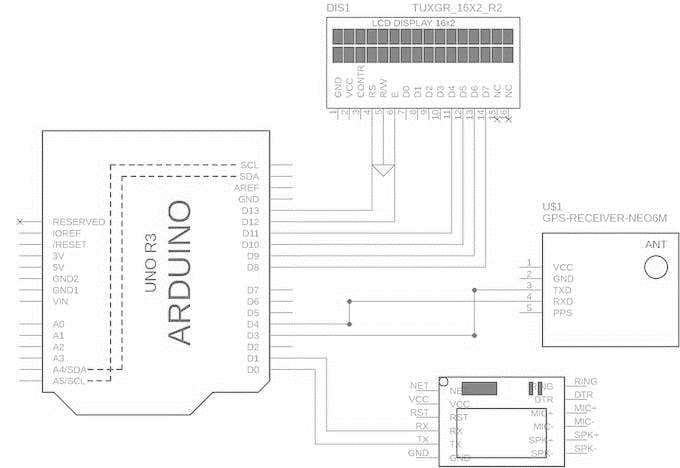
* 1. **Protótipo e Funcionamento**

O módulo Arduino pode ser conectado ao cartão SIM para garantir a comunicação com o sistema de posicionamento global (GPS) e enviar a localização via protocolo MQTT. Utilizou-se um software de série para estabelecer a comunicação com o GPS e conectar o TX do DPS ao D4 da placa Arduino.



**Figura 7. Esquema de conexão do GPS ao Arduino**

Para estabelecer conexão do Arduino com a tela LCD foi utilizado esquema representado a seguir, conectando o LCD ao display D8 e D13.



**Figura 8. Esquema de conexão LCD com módulo Arduino**

Para programação das funcionalidades descritas utilizou-se a linguagem programação própria do Arduino importando a biblioteca tinypgs que auxilia na interação com as funcionalidades do sistema GPS.

Quando o hardware estiver habilitado, basta enviar mensagem utilizando o módulo SIM enviando mensagem via MQTT ao sistema de processamento de controle das bagagens que armazenará as coordenadas da localização do objeto rastreado.

Uma vez que os registros estejam o aplicativo de celular demonstrará, utilizando google maps, a localização mais recente da bagagem, bem como distância relativa ao usuário comparando com a localização do celular.

Os metadados, dados sobre o registro enviado via MQTT, que indicam a data da atualização permitem demonstrar também ao usuário quando foi o último mapeando de localização realizado.

* 1. **Código Arduino**

O código abaixo implementa as funcionalidades de envio das informações descritas no tópico anterior.

|  |
| --- |
| #include <TinyGPS++.h>  #include <SoftwareSerial.h>  #include<LiquidCrystal.h>  LiquidCrystal lcd(13, 12, 11, 10, 9, 8);  static const int RXPin = 4, TXPin = 3;  static const uint32\_t GPSBaud = 9600;  TinyGPSPlus gps;  int temp=0,i;  SoftwareSerial ss(RXPin, TXPin);  String stringVal = "";  void setup(){    Serial.begin(9600);    ss.begin(GPSBaud);    lcd.begin(16,2);    pinMode(13,OUTPUT);    digitalWrite(13,LOW);    lcd.print("Mapeando bagagem");    lcd.setCursor(0,1);    lcd.print("    System      ");    delay(2000);    gsm\_init();    lcd.clear();    Serial.println("AT+CNMI=2,2,0,0,0");    lcd.print("GPS Initializing");    lcd.setCursor(0,1);    lcd.print("  Sem sinal GPS  ");    delay(2000);    lcd.clear();    lcd.print("GPS Encontrado");    lcd.setCursor(0,1);    lcd.print("GPS pronto");    delay(2000);    lcd.clear();    temp=0;  }  void loop()  {  serialEvent();        while(temp)        {          while (ss.available() > 0)          {            gps.encode(ss.read());                  if (gps.location.isUpdated())                  {                   temp=0;                   digitalWrite(13,HIGH);                   tracking();                  }            if(!temp)            break;          }        }        digitalWrite(13,LOW);  }  void serialEvent()  {    while(Serial.available()>0)    {      if(Serial.find("Bagagem"))      {        temp=1;        break;      }      else      {      temp=0;      }    }  }  void gsm\_init()  {    lcd.clear();    boolean at\_flag=1;    while(at\_flag)    {      Serial.println("AT");      delay(1);      while(Serial.available()>0)      {        if(Serial.find("OK"))        at\_flag=0;      }        delay(1000);    }    lcd.clear();    lcd.print("Modulo Connectado..");    delay(1000);    lcd.clear();    boolean echo\_flag=1;    while(echo\_flag)    {      Serial.println("ATE0");      while(Serial.available()>0)      {        if(Serial.find("OK"))        echo\_flag=0;      }      delay(1000);    }    lcd.clear();      delay(1000);    lcd.clear();    lcd.print("Procurando Rede..");    boolean net\_flag=1;    while(net\_flag)    {      Serial.println("AT+CPIN?");      while(Serial.available()>0)      {        if(Serial.find("+CPIN: READY"))        net\_flag=0;      }      delay(1000);    }    lcd.clear();    lcd.print("Rede Encontrada..");      delay(1000);    lcd.clear();  }  void init\_sms()  {    Serial.println("AT+CMGF=1");    delay(400);    Serial.println("AT+CMGS=\"8825737586\"");    delay(400);  }  void send\_data(String message)  {    Serial.print(message);   delay(200);  }  void send\_sms()  {    Serial.write(26);  }  void lcd\_status()  {    lcd.clear();    lcd.print("Mensagem enviada");    delay(2000);    lcd.clear();    lcd.print("Sistema pronto");    return;  }  void tracking()  {      init\_sms();      Serial.print("Latitude: ");      Serial.print(gps.location.lat(), 6);      Serial.print("\n Longitude: ");      Serial.println(gps.location.lng(), 6);      send\_sms();      delay(2000);      lcd\_status();  } |

# Referências

NERI, Renan; LOMBA, Matheus; BULHÕES, Gabriel. **MQTT.** 2019. Disponível em: <https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/mqtt/>. Acesso em: 10 set. 2022.

CELESTRINI, Jordano. **Desenvolvimento de aplicações integrando serviços web, fontes de dados e dispositivos IoT com o uso do Node-RED.** Disponível em: <http://inf.ufes.br/~zegonc/material/Redes%20de%20Sensores%20sem%20Fio/Minicurso%20Node-RED.pdf>. Acesso em: 10 set. 2022.

SANTOS, Ricardo. **REDES GSM, GPRS, EDGE e UMTS.** Disponível em: < https://www.gta.ufrj.br/ensino/eel879/trabalhos\_vf\_2008\_2/ricardo/2.html#:~:text=A%20rede%20GSM%20%C3%A9%20formada,o%20pre%C3%A7o%20para%20o%20usu%C3%A1rio.>. Acesso em: 20 out. 2022.