



UNIVERSIDADE FEDERAL DE UBERLÂNDIA

**Engenharia de Controle e Automação / Engenharia Mecatrônica Sistemas
Embarcados II / Sistemas Digitais para Mecatrônica Prof. Éder Alves de
Moura**

Trabalho Final 02 – Casa Inteligente

Amanda Abigail Bonilla Trochez
Glênio Simião Ramalho
Ísis Germiniani Teixeira
Leonardo Mortari Borba
Neila Cristina Morais Silva

11711EMT027
11611EMT008
11811EMT019
11611EMT001
12011EAU015

Uberlândia-MG
Março/2022

SUMÁRIO

1	INTRODUÇÃO	3
2	OBJETIVO	4
3	BIBLIOTECAS UTILIZADAS	5
3.1	PYGAME	5
3.2	PIL (Python Imaging Library)	5
3.3	NumPy (Numerical Python)	5
3.4	Virtualenv	6
4	CÓDIGOS E FUNÇÕES	7
4.1	Acende_led	7
4.2	Style.css	8
4.3	Index.html	8
4.4	House_Control.ino	10
5	DIAGRAMA LÓGICO DE FUNCIONAMENTO DO SISTEMA	10
6	MONTAGEM E EXECUÇÃO	11
7	CONCLUSÃO	13
8	BIBLIOGRAFIA	14
9	ANEXO	15

1 INTRODUÇÃO

A Internet das Coisas ou *Internet of Things* (IoT) em inglês é uma referência à capacidade de diferentes objetos se conectarem com a internet, esses itens conseguem coletar e transmitir dados a partir da nuvem.

Atualmente, esses dispositivos IoT vêm sendo utilizados tanto em situações comuns da vida diária como no âmbito profissional. Desse modo, essas ferramentas tecnológicas estão contribuindo para o acontecimento da transformação digital no mundo.

Podemos encontrá-los em diversas situações como em nossas casas, com Smart TVs, luzes, tomadas e fechaduras inteligentes, na agricultura, com sensores que monitoram o solo e o clima e controlam a irrigação. Podemos até mesmo vesti-los como os coletes de monitoramento de atletas de alta performance ou os smartwatches.

Em um mundo cada vez mais conectado é essencial dominar os conhecimentos e conceitos necessários para implementar e operar um dispositivo com IoT.

2 OBJETIVO

O objetivo desse trabalho é desenvolver uma aplicação considerando o cenário de Internet das Coisas (Internet of Things – IoT), que é um mercado de trabalho em crescente demanda. Para isso, foi desenvolvido uma aplicação com Linux, Python, Django e Arduino.

Esse trabalho tem por objetivo representar o controle de uma casa automatizada, onde a aplicação é executada no ambiente virtual do Linux e a interface Homem-Máquina em Python e Django, permitindo controlar a função Liga/Desliga de equipamentos da casa. Essa interface é acessada em um navegador executando na máquina hospedeira e não na própria máquina virtual.

Para representar os equipamentos, foi utilizado leds (com as devidas resistências elétricas) nas portas do Arduino. O comando de liga desliga desses pinos é executado pela interface, via navegador. Os comandos são transmitidos para o Arduino via interface serial.

3 BIBLIOTECAS UTILIZADAS

Nesta seção serão apresentadas as bibliotecas utilizadas no código da simulação.

3.1 PySerial

Uma das primeiras interfaces para a conexão entre computadores e periféricos foi a RS232-C. Com o advento do USB, a comunicação serial se tornou extremamente mais simples. Cada sistema operacional expõe uma API serial ligeiramente diferente, de modo que a chamada direta desta API seria muito detalhada no desenvolvimento de aplicações multiplataforma. Mas, felizmente para os desenvolvedores, existem várias bibliotecas, nas diferentes linguagens, que procuram integrar estas diferentes API's numa única forma de chamada, independente do SO.

No caso da linguagem Python, uma biblioteca que faz exatamente isto é a pyserial. É uma biblioteca razoavelmente simples de ser utilizada.

3.2 Django

Django é um framework python de alto nível, que possibilita o desenvolvimento de sites seguros e de fácil manutenção.

Tem como vantagens:

- Gratuito
- Código aberto
- Comunidade ativa
- Boa documentação
- Versatilidade
- Segurança
- Portabilidade

3.3 Arduino

Os microcontroladores estão presentes em todos os equipamentos que

envolvem a eletrônica, por terem tamanho reduzido, excelente desempenho, baixo consumo de energia e um ótimo custo benefício. São placas que possuem um microprocessador para analisar as instruções pré-definidas e executar tarefas, memória para armazenar os dados, pinos de entrada e saída que tem o objetivo de comunicar o microcontrolador com os sensores ou atuadores, além de temporizadores, comunicação serial, resistores.

Um dos tipos de microcontroladores mais usados e estudados é o Arduino. Por se tratar de uma plataforma muito fácil de ser utilizada em projetos e por conter uma grande comunidade de usuários.

Um dos diferenciais dessa placa é a sua fonte aberta “open source” que é o conceito de que o software e o hardware são livres.

3.4 Virtualenv

Virtualenv é uma ferramenta para criar ambientes virtuais Python isolados. Desde o Python 3.3, um subconjunto dele foi integrado à biblioteca de módulos venv padrão, que é uma forma de isolar diferentes ambientes de desenvolvimento, permitindo assim que os programadores usem versões específicas de diferentes pacotes sem afetar a instalação de outros aplicativos ou sistemas.

4 CÓDIGOS E FUNÇÕES

Nesta seção serão apresentados os códigos e funções utilizadas no código e seu funcionamento.

4.1 acende_led

A função `acende_led` é responsável por passar os parâmetros para que os leds acendam. Inicialmente, a função recebe a porta e o comando, da função `ligaled`, então, este comando é convertido para binário, para posteriormente ser interpretado no controlador. Após a conversão, o arduino carrega a porta e escreve o comando. Foi optado também por fechar a conexão após o comando. Além disso, a função envia uma resposta para indicar o seu funcionamento ou não.

```
def acende_led(request):  
    if request.method == 'POST':  
        try:  
            port = request.POST.get('port')  
            #print('porta:', port)  
            ledCommand = request.POST.get('ledCommand')  
            # print(ledCommand)  
  
            arduino = serial.Serial(port, 9600, timeout=1)  
            command = '{}'.format(ledCommand).encode()  
            arduino.write(command)  
            arduino.close()  
            resposta = 'Led Aceso!'  
  
        except Exception as e:  
            # print('erro', e)  
            resposta = f'Ocorreu o erro, {e}'  
  
    return JsonResponse({'resposta':resposta})
```

Figura 1: Código `acende_led`.

4.2 Style.css

No código Style.css temos a referência do nosso desenvolvedor front-end que é responsável pela experiência do usuário dentro da aplicação, é ele quem vai desenhar e desenvolver as páginas com as quais, posteriormente, o usuário irá interagir e inclui elementos que determinam a identidade visual de um site ou aplicativo. Dentro desse código encontramos bootstrap, cores, botões e o footer.

4.3 Index.html

Código Index é o principal onde é desenvolvido o site, como podemos ver na Figura 1, que mostra a primeira parte do código temos todas as referências, incluindo o código Style que foi comentado acima.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no" />
  <meta name="description" content="" />
  <meta name="author" content="" />
  <title>GABIN</title>
  <link rel="icon" type="image/x-icon" href="assets/favicon.ico" />
  <!-- Font Awesome icons (free version)-->
  <script src="https://use.fontawesome.com/releases/v5.15.4/js/all.js" crossorigin="anonymous"></script>
  <!-- Google fonts-->
  <link href="https://fonts.googleapis.com/css?family=Montserrat:400,700" rel="stylesheet" type="text/css" />
  <link href="https://fonts.googleapis.com/css?family=Lato:400,700,400italic,700italic" rel="stylesheet" type="text/css" />
  <!-- Core theme CSS (includes Bootstrap)-->
  <link href="{% static 'styles.css' %}" rel="stylesheet" />
</head>
<body>
```

Figura 2: Código Index.html Linhas 1 a 20.

Na seguinte parte do código, Figura 3 tem nosso código principal, temos a parte principal do site, onde temos o nome do projeto e a parte do menu principal do site.


```

<!-- Navigation-->
<nav class="navbar navbar-expand-lg bg-secondary text-uppercase fixed-top" id="mainNav">
  <div class="container">
    <a class="navbar-brand" href="#page-top">Projeto GABIN</a>
    <button class="navbar-toggler text-uppercase font-weight-bold bg-primary text-white rounded" type="button" data-bs-toggle="collapse" data-bs-target="#navbarResponsive"
      <i class="fas fa-bars"></i>
    </button>
    <div class="collapse navbar-collapse">
      <ul class="navbar-nav ms-auto">
        <li class="nav-item mx-0 mx-lg-1"><a class="nav-link py-3 px-0 px-lg-3 rounded" href="#Casa">Casa</a></li>
        <li class="nav-item mx-0 mx-lg-1"><a class="nav-link py-3 px-0 px-lg-3 rounded" href="#Quarto">Quarto</a></li>
        <li class="nav-item mx-0 mx-lg-1"><a class="nav-link py-3 px-0 px-lg-3 rounded" href="#Cozinha">Cozinha</a></li>
      </ul>
    </div>
  </div>
</nav>

```

Figura 3: Código Index.html Linhas 21 a 37.

Para a próxima parte do código temos a construção do Header, Figura 4, que é um bloco de conteúdo que pode conter um ou mais elementos, campo de busca, elementos de navegação, um logo ou banner, uma introdução, um pequeno prefácio ou um índice em formato de lista. Normalmente trabalha como um agregador do conteúdo do cabeçalho de um documento ou de uma seção.

```

<!-- Masthead-->
<header class="masthead bg-primary text-white text-center">
  <div class="container d-flex align-items-center flex-column">
    <!-- Masthead Avatar Image-->
    
    <!-- Masthead Heading-->
    <h1 class="masthead-heading text-uppercase mb-0">GABIN - Casa Inteligente</h1>
    <!-- Icon Divider-->
    <div class="divider-custom divider-light">
      <div class="divider-custom-line"></div>
      <div class="divider-custom-icon"><i class="fas fa-star"></i></div>
      <div class="divider-custom-line"></div>
    </div>
    <!-- Masthead Subheading-->
    <p class="masthead-subheading font-weight-light mb-0">Glênio - Abhy - Borba - Isis - Neila</p>
  </div>
</header>

```

Figura 4: Código Index.html Linhas 38 a 54.

Após ter feito a pagina inicial e o menu podemos seguir com a construção dos códigos para cada um dos cômodos da casa, neles temos as imagens, botões de ligar e desligar que é onde a gente manda o comando e faz controle dos led no nosso circuito. Para finalizar o código é criado o footer que representa o rodapé do site, no rodapé temos as redes sociais, e mais informações do projeto como vemos na Figura 5.

```

<footer class="footer text-center">
  <div class="container">
    <div class="row">
      <!-- Footer Location-->
      <div class="col-lg-4 mb-5 mb-lg-0">
        <h4 class="text-uppercase mb-4">Endereço</h4>
        <p class="lead mb-0">
          Av. João Naves de Ávila, 2121.
          <br />
          Santa Mônica, Uberlândia - MG, 38408-100.
        </p>
      </div>
      <!-- Footer Social Icons-->
      <div class="col-lg-4 mb-5 mb-lg-0">
        <h4 class="text-uppercase mb-4">Redes Sociais</h4>
        <a class="btn btn-outline-light btn-social mx-1" href="#"><i class="fab fa-fw fa-facebook-f"></i></a>
        <a class="btn btn-outline-light btn-social mx-1" href="#"><i class="fab fa-fw fa-twitter"></i></a>
        <a class="btn btn-outline-light btn-social mx-1" href="#"><i class="fab fa-fw fa-linkedin-in"></i></a>
        <a class="btn btn-outline-light btn-social mx-1" href="#"><i class="fab fa-fw fa-dribbble"></i></a>
      </div>
      <!-- Footer Quarto Text-->
      <div class="col-lg-4">
        <h4 class="text-uppercase mb-4">Sobre</h4>
        <p class="lead mb-0">
          Projeto de IOT realizado para matéria de Sistemas Embarcados II.
        </p>
      </div>
    </div>
  </div>
</footer>

```

Figura 5: Código Index.html Linhas 137 a 171.

4.4 House_Control.ino

Como comentado anteriormente na seção dois, utilizou-se a biblioteca do arduino para inicialmente declarar os eletrodomésticos que correspondem a leds no circuito e os pinos onde foi conectado cada led. Em seguida, a comunicação serial é iniciada.

```

House_Control
const int allPins[] = {13, 12, 11, 10};
const int geladeiraPin = allPins[0];
const int fogaoPin = allPins[1];
const int ventPin = allPins[2];
const int tvPin = allPins[3];
int incomingByte;      // a variable to read incoming serial data into

void setup() {
  // initialize serial communication:
  Serial.begin(9600);
  // initialize the LED pin as an output:
  pinMode(geladeiraPin, OUTPUT);
}

```

Figura 6: Código House_Control.ino. Linhas 1 à 14.

Em seguida, um loop checa constantemente se há algum novo dado serial. Caso encontre, o último byte do buffer serial analisado e determina quais ações serão realizadas. Se o *byte* representar uma letra maiúscula, um comando de “ligar” será executado, caso o este represente uma letra minúscula, “desligar” entrará em ação. O que determina qual eletrodoméstico será ativado ou desativado é o byte correspondente a sua letra inicial.

5 DIAGRAMA LÓGICO DE FUNCIONAMENTO DO SISTEMA

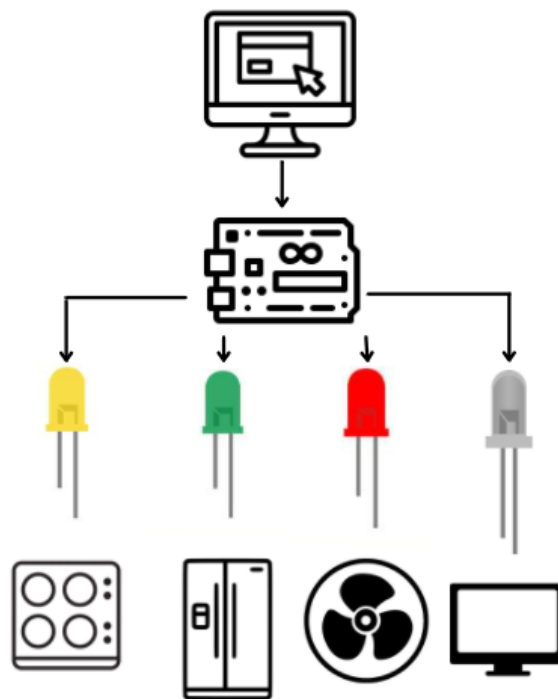


Figura 7: Diagrama lógico de funcionamento do sistema..

No diagrama, Figura 7, temos o Diagrama Lógico de Funcionamento que representa o fluxo do nosso projeto, começando por uma chamado feito pelo site, no botão de desligar e ligar, esse pedido passa para o Arduino UNO, que está ligado ao nosso computador, e nele mesmo temos os leds que acendem na ordem quando ligamos algum dos aparelhos da casa. Podemos ver que cada aparelho tem um led de cor diferente que representa ele, amarelo (fogão), verde (geladeira), vermelho (ventilador) e branco (televisão).

6 MONTAGEM E EXECUÇÃO

Para testar o sistema deve-se realizar a montagem do circuito conforme representado pelas figuras 8 e 9 utilizando um kit Arduino.

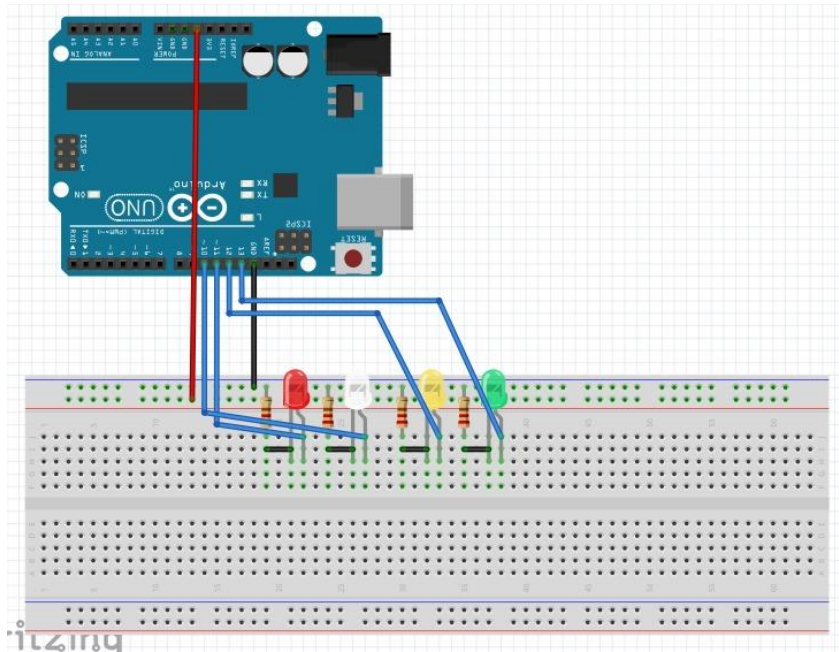


Figura 8: Representação do circuito a ser montado utilizando um kit Arduino.

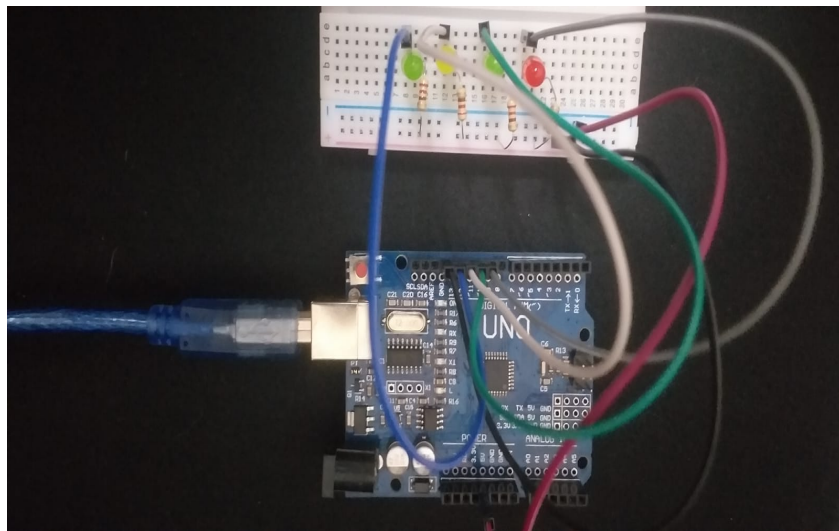


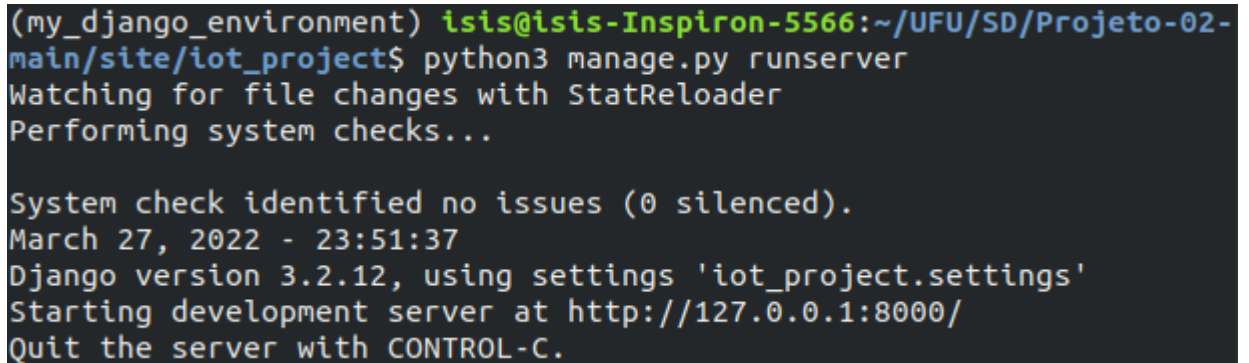
Figura 9: Montagem experimental do circuito utilizando um kit Arduino..

Em seguida, é necessário entrar no ambiente virtual do Virtualenv, com as bibliotecas Django e PySerial previamente instaladas, inserindo o seguinte comando no terminal Linux:

```
$ mkvirtualenv my_django_environment
```

Com os comandos `cd` e `ls`, navegue até a pasta do projeto em seu computador em que se encontra o código `manage.py`.

`$ python3 manage.py runserver`

A terminal window with a dark background and light-colored text. The prompt is '(my_django_environment) isis@isis-Inspiron-5566:~/UFU/SD/Projeto-02-main/site/iot_project\$'. The command 'python3 manage.py runserver' has been entered. The output shows 'Watching for file changes with StatReloader', 'Performing system checks...', 'System check identified no issues (0 silenced).', the date and time 'March 27, 2022 - 23:51:37', 'Django version 3.2.12, using settings \'iot_project.settings\'', 'Starting development server at http://127.0.0.1:8000/', and 'Quit the server with CONTROL-C.'

```
(my_django_environment) isis@isis-Inspiron-5566:~/UFU/SD/Projeto-02-main/site/iot_project$ python3 manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
March 27, 2022 - 23:51:37
Django version 3.2.12, using settings 'iot_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Figura 10: Resultado do terminal após executar o código `manage.py`.

Clique na url gerada <http://127.0.0.1:8000> para acessar o GABIN - Casa Inteligente.

Clique [aqui](#) para acessar o vídeo que contém uma demonstração do funcionamento do sistema, o trecho encontra-se a partir dos 8min11 de vídeo.

7 CONCLUSÃO

Conclui-se que o objetivo de representar o controle de uma casa automatizada, onde a aplicação é executada no ambiente virtual do Linux e a interface Homem-Máquina em Python e Django, permitindo controlar a função Liga/Desliga de equipamentos da casa foi alcançado com sucesso.

Durante o desenvolvimento surgiram problemas que foram superado com sucesso, como a dificuldade de reconhecimento das imagens css pelo Django ou o não reconhecimento da entrada serial pelo arduino.

Além disso, foi possível construir uma base a respeito de equipamentos e objetos com IoT que somou à formação dos autores.

8 BIBLIOGRAFIA

- [1] Django Documentation. Disponível em:
<<https://docs.djangoproject.com/en/4.0/>>. Acesso em: 20 de março de 2022.
- [2] MOURA, Éder Alves de. Trabalho Final 2 - Roteiro, UFU, 2022.
- [3] Arduino. Disponível em: <<https://www.arduino.cc/>>. Acesso em: 20 de março de 2022.

9 ANEXO

9.1 Manage.py

```
"""Django's command-line utility for
administrative tasks."""
import os

import sys


def main():

    """Run administrative tasks."""

    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'iot_project.settings')
    try:

        from django.core.management import
execute_from_command_line
    except ImportError as exc:

        raise ImportError(

            "Couldn't import Django. Are you sure
it's installed and "

            "available on your PYTHONPATH
environment variable? Did you "

            "forget to activate a virtual
environment?"

        ) from exc

    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```


9.2 Index.html

```
<html lang="en">

  <head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width,
initial-scale=1.0, shrink-to-fit=no" />
    <meta name="description" content="" />
    <meta name="author" content="" />

    <title>GABIN</title>

    <link rel="icon" type="image/x-icon" href="assets/favicon.ico" />

    <!-- Font Awesome icons (free version)-->

    <script src="https://use.fontawesome.com/releases/v5.15.4/js/all.js"
crossorigin="anonymous"></script>

    <!-- Google fonts-->

    <link
href="https://fonts.googleapis.com/css?family=Montserrat:400,700"
rel="stylesheet" type="text/css" />

    <link
href="https://fonts.googleapis.com/css?family=Lato:400,700,400italic,70
0italic" rel="stylesheet" type="text/css" />
    <!-- Core theme CSS (includes Bootstrap)-->

    {% load static %}

    <link href="{% static 'styles.css' %}" rel="stylesheet" />

  </head>

  <body>

    <!-- Navigation-->

    <nav class="navbar navbar-expand-lg bg-secondary text-uppercase
fixed-top" id="mainNav">
      <div class="container">

        <a class="navbar-brand" href="#page-top">Projeto GABIN</a>

        <button class="navbar-toggler text-uppercase
font-weight-bold bg-primary text-white rounded" type="button"
data-bs-toggle="collapse" data-bs-target="#navbarResponsive"
aria-controls="navbarResponsive" aria-expanded="false"
aria-label="Toggle navigation">
          Menu
```

```

        <i class="fas fa-bars"></i>

    </button>

    <div class="collapse navbar-collapse">

        <ul class="navbar-nav ms-auto">

            <li class="nav-item mx-0 mx-lg-1"><a class="nav-link
py-3 px-0 px-lg-3 rounded" href="#Casa">Casa</a></li>
            <li class="nav-item mx-0 mx-lg-1"><a class="nav-link
py-3 px-0 px-lg-3 rounded" href="#Quarto">Quarto</a></li>
            <li class="nav-item mx-0 mx-lg-1"><a class="nav-link
py-3 px-0 px-lg-3 rounded" href="#Cozinha">Cozinha</a></li>
        </ul>

    </div>

</div>

</nav>

<!-- Masthead-->

<header class="masthead bg-primary text-white text-center">

    <div class="container d-flex align-items-center flex-column">

        <!-- Masthead Avatar Image-->

        <!-- Masthead Heading-->

            <h1 class="masthead-heading text-uppercase mb-0">GABIN -
Casa Inteligente</h1>

            <!-- Icon Divider-->

            <div class="divider-custom divider-light">

                <div class="divider-custom-line"></div>

                <div class="divider-custom-icon"><i class="fas
fa-star"></i></div>

                <div class="divider-custom-line"></div>

            </div>

            <!-- Masthead Subheading-->

            <p class="masthead-subheading font-weight-light mb-0">Glênio
- Abhy - Borba - Isis - Neila</p>

        </div>

    </header>

    <!-- Casa Section-->

    <section class="page-section Casa" id="Casa">

        <div class="container">

            <!-- Casa Section Heading-->

```

```

        <h2 class="page-section-heading text-center text-uppercase
text-secondary mb-0">Casa</h2>

        Seleccione a porta arduino: <input id="port" type="text"
name="port" placeholder="Arduino Port here">

        <hr>

        <!-- Icon Divider-->

        <div class="divider-custom">

            <div class="divider-custom-line"></div>

            <div class="divider-custom-icon"><i class="fas
fa-star"></i></div>

            <div class="divider-custom-line"></div>

        </div>

        <button class="buttonon" type="button"
onclick="LigaLed('H')">Ligar</button>

        <button class="buttonoff" type="button"
onclick="LigaLed('l')">Desligar</button>

    </div>

</section>

<!-- Quarto Section-->

    <section class="page-section bg-primary text-white mb-0"
id="Quarto">

        <div class="container">

            <!-- Quarto Section Heading-->

            <h2 class="page-section-heading text-center text-uppercase
text-white">Quarto</h2>

            <!-- Icon Divider-->

            <div class="divider-custom divider-light">

                <div class="divider-custom-line"></div>

                <div class="divider-custom-icon"><i class="fas
fa-star"></i></div>

                <div class="divider-custom-line"></div>

            </div>

            <div class="row justify-content-center">

                <!-- Quarto Item 1-->

                <div class="col-md-6 col-lg-4 mb-5">

                    <div class="Casa-item mx-auto">

```

```

        </div>

        <button class="buttonon1" type="button"
onclick="Ligaled('T')">Ligar</button>
        <button class="buttonoff1" type="button"
onclick="Ligaled('t')">Desligar</button>
    </div>

    <!-- Quarto Item 2-->
    <div class="col-md-6 col-lg-4 mb-5">

        <div class="Casa-item mx-auto" >

            
        </div>

        <button class="buttonon1" type="button"
onclick="Ligaled('V')">Ligar</button>
        <button class="buttonoff1" type="button"
onclick="Ligaled('v')">Desligar</button>
    </div>

</div>

</div>

</section>

<!-- Cozinha Section-->
<section class="page-section" id="Cozinha">
    <div class="container">

        <!-- Cozinha Section Heading-->
        <h2 class="page-section-heading text-center text-uppercase
text-secondary mb-0">Cozinha</h2>
        <!-- Icon Divider-->
        <div class="divider-custom">

            <div class="divider-custom-line"></div>

            <div class="divider-custom-icon"><i class="fas
fa-star"></i></div>

            <div class="divider-custom-line"></div>

        </div>

        <div class="row justify-content-center">

            <!-- Cozinha Item 1-->
            <div class="col-md-6 col-lg-4 mb-5">

                <div class="Casa-item mx-auto">

```



```

href="#"><i class="fab fa-fw fa-facebook-f"></i></a>
        <a class="btn btn-outline-light btn-social mx-1"
href="#"><i class="fab fa-fw fa-twitter"></i></a>
        <a class="btn btn-outline-light btn-social mx-1"
href="#"><i class="fab fa-fw fa-linkedin-in"></i></a>
        <a class="btn btn-outline-light btn-social mx-1"
href="#"><i class="fab fa-fw fa-dribbble"></i></a>
    </div>

    <!-- Footer Quarto Text-->

    <div class="col-lg-4">

        <h4 class="text-uppercase mb-4">Sobre</h4>

        <p class="lead mb-0">

            Projeto de IOT realizado para matéria de Sistemas
Embarcados II.

        </p>

    </div>

</div>

</div>

</footer>

<!-- Copyright Section-->

<div class="copyright py-4 text-center text-white">

    <div class="container"><small>Copyright &copy; www.projetogabin.com
</small></div>

</div>

<script>

    function LigaLed(ledCommand){

        var xhr = new XMLHttpRequest();

        var formData = new FormData();

        port = document.getElementById('port').value

        formData.append('ledCommand', ledCommand);

        formData.append('port', port);

        xhr.open("POST", "{% url 'acende_led' %}", true);

        xhr.setRequestHeader('X-CSRFToken', '{{ csrf_token }}');

        // xhr.onreadystatechange = function() {

        //     if(xhr.readyState == 4 && xhr.status == 200) {

        //         alert(xhr.response)

        //     }

```

```
        // }

        xhr.send(formData);

    }

</script>

</html>
```

9.3 House_Control.ino

```
const int allPins[] = {13, 12, 11, 10};
const int geladeiraPin = allPins[0];
const int fogaoPin = allPins[1];
const int ventPin = allPins[2];
const int tvPin = allPins[3];
int incomingByte;    // a variable to read incoming serial data into

void setup() {
    // initialize serial communication:
    Serial.begin(9600);
    // initialize the LED pin as an output:
    pinMode(geladeiraPin, OUTPUT);
}

void loop() {
    // see if there's incoming serial data:
    if (Serial.available() > 0) {
        // read the oldest byte in the serial buffer:
        incomingByte = Serial.read();

        // Comando liga-desliga para geladeira (G para on e g para off)
        if (incomingByte == 'G' | incomingByte == 'g'){
            acende_desliga(incomingByte, geladeiraPin);
        }

        // Comando liga-desliga para fogao (F para on e f para off)
```

```

    if (incomingByte == 'F' | incomingByte == 'f'){
        acende_desliga(incomingByte, fogaoPin);
    }

    // Comando liga-desliga para ventilador (V para on e v para off)
    if (incomingByte == 'V' | incomingByte == 'v'){
        acende_desliga(incomingByte, ventPin);
    }

    // Comando liga-desliga para TV (T para on e t para off)
    if (incomingByte == 'T' | incomingByte == 't'){
        acende_desliga(incomingByte, tvPin);
    }

    if (incomingByte == 'H' | incomingByte == 'h'){
        for (int i = 0; i < sizeof(allPins)/sizeof(int); i++){
            acende_desliga(incomingByte, allPins[i]);
        }
    }
}

void acende_desliga(int comando, const int pin){
    if (isUpperCase(comando) == true){
        digitalWrite(pin, HIGH);
    }
    else{
        digitalWrite(pin, LOW);
    }
}

```

9.4 views.py

```

from django.shortcuts import render
from django.http import JsonResponse

```



```

import serial

# Create your views here.
def index(request):
    #Essa função juntará as informações iniciais e renderizará de acordo
    com o que for passado
    titulo = "Bem vindo ao sistema de controle inteligente GABIN"
    return render(request, 'html/index.html', context={
        'titulo':titulo,
    })

def acende_led(request):
    if request.method == 'POST':
        try:
            port = request.POST.get('port')
            print('porta:', port)
            ledCommand = request.POST.get('ledCommand')
            print(ledCommand)
            #arduino = serial.Serial(f'{port}', 9600, timeout=1)
            arduino = serial.Serial('/dev/ttyUSB0', 9600, timeout=1)
            command = '{}'.format(ledCommand).encode()
            arduino.write(command)
            arduino.close()
            resposta = 'Led Aceso!'
            print(resposta)
        except Exception as e:
            print('erro', e)
            resposta = f'Ocorreu o erro, {e}'

    return JsonResponse({'resposta':resposta})

```

9.5 apps.py

```

from django.apps import AppConfig

```

```
class HouseControlConfig(AppConfig):  
    default_auto_field = 'django.db.models.BigAutoField'  
    name = 'house_control'
```