INSTITUTO FEDERAL
Paraíba

Relatório Final de Projeto Sistemas Embarcados

Rafael Figueredo Guimarães
João Carlos Guerra de Abreu
Lucas Matheus Torres Costa

Monitoramento de temperatura e umidade por meio de sensores.

Diagrama de Blocos do Hardware

No diagrama de blocos abaixo na parte superior temos uma **fonte de 3.3v** responsável por alimentar o módulo ESP8266 bem como o sensor de temperatura e umidade DTH11. Do lado esquerdo temos o **ESP8266** responsável por receber o sinal e enviar os dados para o servidor web. Do lado direito temos o sensor de temperatura e umidade **DHT11** que coleta os dados e envia para esses dados para o ESP8266

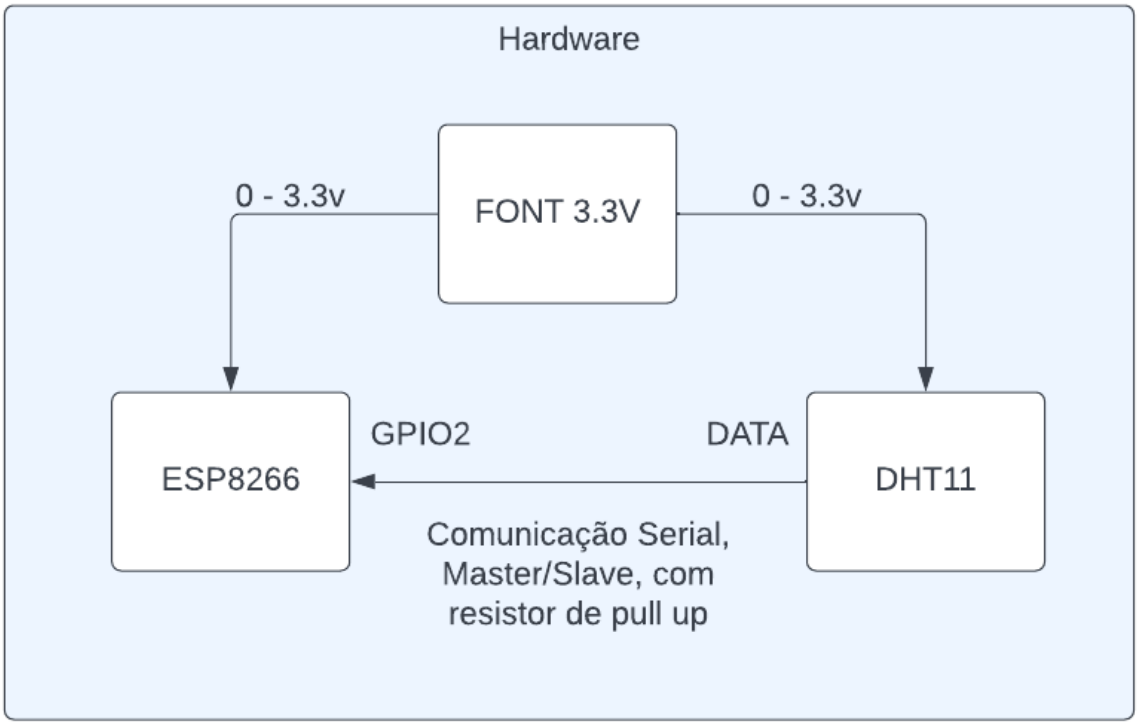
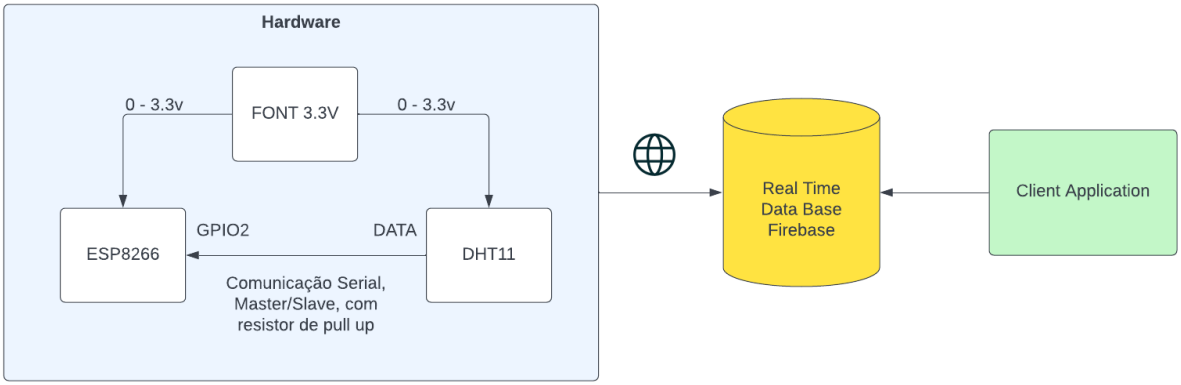


Diagrama de Blocos da Aplicação

Nosso projeto tem como objetivo desenvolver uma aplicação focada na interface com usuário, para isso, temos no projeto um banco de dados que vai receber os dados em tempo real, a cada 5 segundos do nosso sistema embarcado, e uma aplicação Web que vai que vai mostrar esses dados de modo mais amigável para o usuário.

Nossa aplicação web (Client Application na figura abaixo) se conecta com o banco de dados, mostrando os dados para o usuário.



Sobre o DHT11 e Comunicação com ESP8266

Pinagem DHT11

1. VDD supply 3.3 ~ 5.5V DC 2.
2. DATA **serial data, single-bus**
3. NC NC (não utilizado)
4. GND grounding, power negative

O sensor digital de temperatura e umidade **DHT11** é uma saída de sinal digital calibrada do sensor combinado de temperatura e umidade. Ele usa uma tecnologia de

captura de módulos digitais dedicados e a tecnologia de sensor de temperatura e umidade para garantir produtos com alta confiabilidade e excelente estabilidade a longo prazo. O sensor inclui um elemento resistivo e com um microcontrolador de 8 bits de alto desempenho conectado .

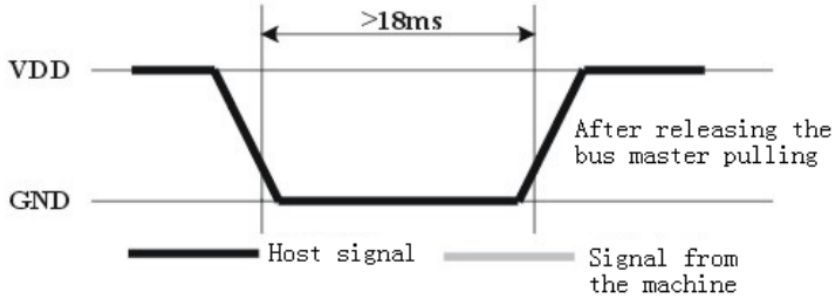
De acordo com o Datasheet o **DHT11** possui estabilidade a longo prazo, medição de umidade relativa e temperatura, excelente qualidade, resposta rápida, capacidade anti-interferência, transmissão de sinal de longa distância, saída de sinal digital, calibração precisa.

Comunicação DHT11 e ESP8266

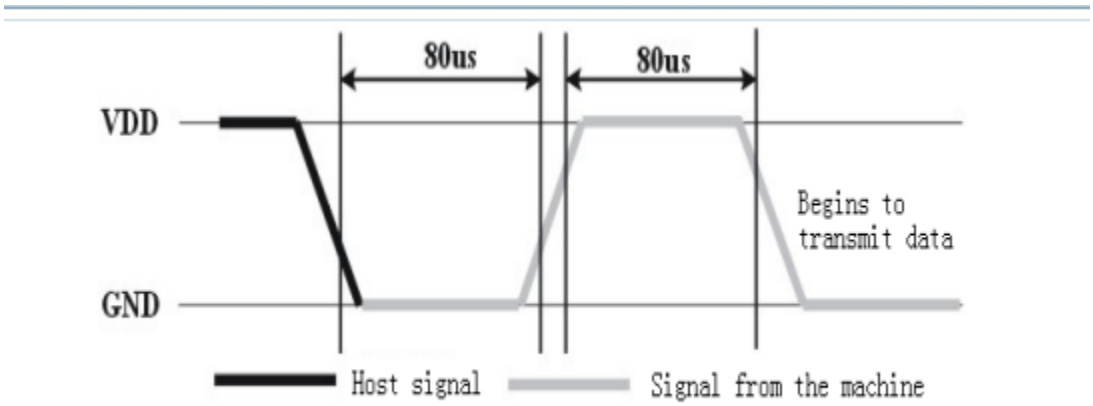
O dispositivo DHT11 usa uma comunicação simplificada de barramento único. Barramento único, ou seja, apenas uma linha de dados, Dispositivo (mestre ou escravo) através de uma porta de dreno aberto ou tri-state é conectado à linha de dados para permitir que o dispositivo envie dados quando não puder liberar o barramento e deixar outros dispositivos usarem o barramento; A comunicação entre mestre e escravo pode ser completada pelas seguintes etapas:

1. O DHT11 após alimentação, testa os dados de temperatura e umidade do ambiente e registrar dados enquanto as linhas de dados DATA do DHT11 são puxadas por um resistor pull-up e permanecem em nível lógico alto; O DHT11 desta vez o pino DATA é o estado de entrada, sempre detecta sinais externos.

2. A saída de I/O do microprocessador enquanto a saída está definida como baixa então a I/O do microprocessador é configurada para entrar no estado, devido ao resistor de pull-up, o microprocessador I/O e as linhas de dados do DHT11 também vão para o alto, esperando para atender os sinais DHT11 transmitidos como mostrado:



3. Quando o pino DATA do DHT11 detecta sinais externos em nível lógico baixo esperando por um sinal após um delay o pino DATA do DHT11 se torna um output com nível lógico baixo e 80 microsegundos de resposta ao sinal seguido de um output de 80 micro-segundos como mostrado abaixo:



Sobre o ESP8266

O ESP8266 nada mais do que um chip, um microcontrolador criado pela empresa Espressif System em 2014. Ele virou um sucesso pois o mesmo tem baixo custo, pequeno tamanho, além de possuir WiFi. A Figura abaixo mostra a arquitetura interna em diagrama de blocos

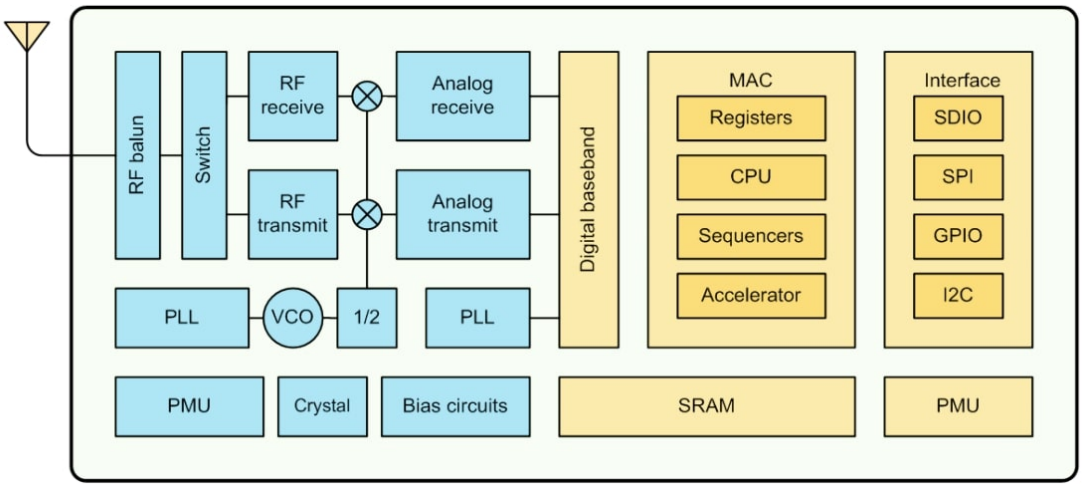


Tabela de Configuração do GPIO:

A tabela abaixo mostra o configuração de GPIO do ESP8266

GPIO Number	GPIO Name	Boot State	Precautions and information
0	D3	ALTO para inicialização BAIXO para programação serial	Resistor de pullup na maioria das placas
1			Usado como serial (TX) Pode ser usado, mas observe que o sinal pode piscar devido à atividade serial durante a inicialização.
2	D4	ALTO para inicialização ALTO para programação serial	Conectado ao LED integrado (baixo ativo) Usado como serial1 (TX1)
3			Usado como serial (RX) Pode ser usado, mas observe que o sinal pode piscar devido à atividade serial durante a inicialização.
4	D2		Geralmente usado como SDA (I2C)
5	D1		Geralmente usado como SCL (I2C)
6			Reservado para SPI + flash

7			Reservado para SPI + flash
8			Reservado para SPI + flash
9	D11		Reservado para SPI + flash
10	D12		Reservado para SPI + flash
11			Reservado para SPI + flash
12	D6		
13	D7		
14	D5		
15	D8	LOW for boot	Resistor pulldown na maioria das placas
16	D0		Sem PWM (contador não é possível). Nenhum pull up interno disponível. Usado no modo de suspensão para despertar

Ambiente de desenvolvimento expressif ESP8266 para freeRTOS

Para instalar o ambiente de desenvolvimento do freeRTOS para ESP8266 no ambiente linux basta seguirmos os seguintes passos:

1. criar uma pasta chamada esp dentro de `/home` e em seguida entrar na pasta:

```
$ mkdir esp
$ cd esp
```

2. Dentro da pasta ~/esp, clonar repositório oficial da express if:

```
$ git clone https://github.com/espressif/ESP8266_RTOS_SDK.git
```

3. Acessar arquivo .bashrc e colar o seguinte comando que o idf fique acessível de qualquer pasta do sistema.

```
$ cd ~/
$ code .bashrc
```

4. Quando o arquivo abrir cole o seguinte comando:

```
export IDF_PATH=~/esp/ESP8266_RTOS_SDK
```

5. Instale as bibliotecas necessárias usando python 2.7

```
python2.7 -m pip install --user -r $IDF_PATH/requirements.txt
```

Criando um projeto

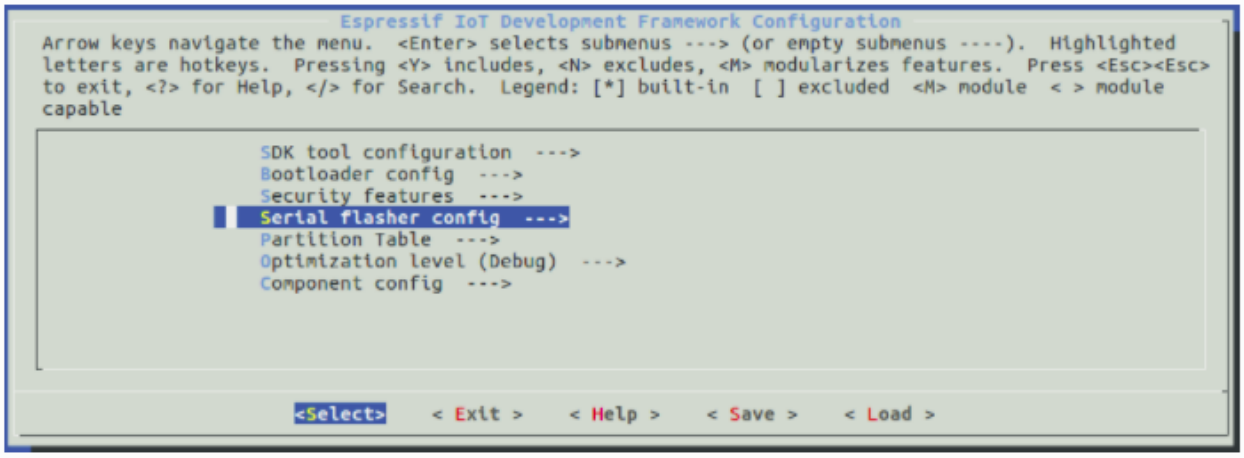
1. Acessar a pasta /esp e criar uma cópia do projeto template.

```
$ cd ~/esp
$ cp -r $IDF_PATH/examples/get-started/hello_world .
```

2. Agora precisamos configurar o projeto usando make, no meu caso faltaram algumas bibliotecas e deu erro nas primeiras tentativas. Então o ideal é visualizar os logs e buscar no google como instalar as bibliotecas que não estão disponíveis na sua máquina.

```
$ cd ~/esp/hello_world
$ make menuconfig
```

3. Se tudo ocorrer bem vai aparecer a seguinte tela no terminal, clique em



Project configuration - Home window

4. Selecione a opção SERIAL FLASHER CONFIG, e selecione a porta que está conectada ao NodeMCU ou ESP8266.

5. Agora basta rodar o flash para enviar o código hello world para seu dispositivo

```
$ make flash
```

6. Para visualizar o serial monitor digite:

```
$ make monitor
```

Ambiente de desenvolvimento no arduino IDE

1. Para o ambiente do arduino IDE é necessário instalar a biblioteca ESP-8266

2. É necessário instalar essa biblioteca baixando seu arquivo como .zip e adicionar através da IDE do arduino disponível no link abaixo: (github.com/FirebaseExtended/firebase-arduino)

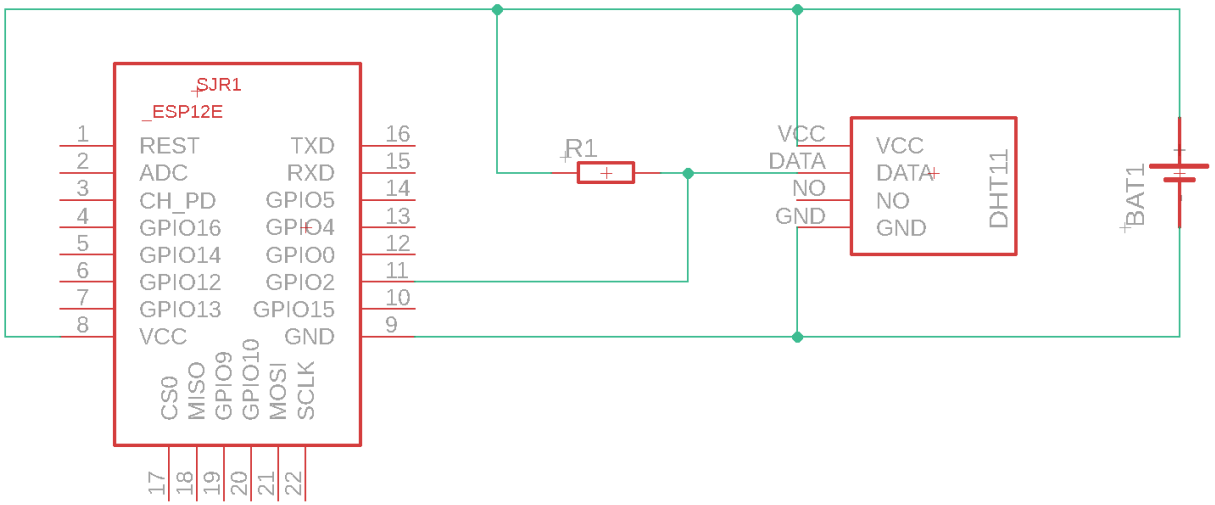
3. Instalar biblioteca ArduinoJson **versão 2.7**

Criação do Banco de dados e conexão com o banco de dados:

Para esse caso, decidimos gravar um vídeo e disponibilizar aqui a documentação.

[Vídeo Youtube]

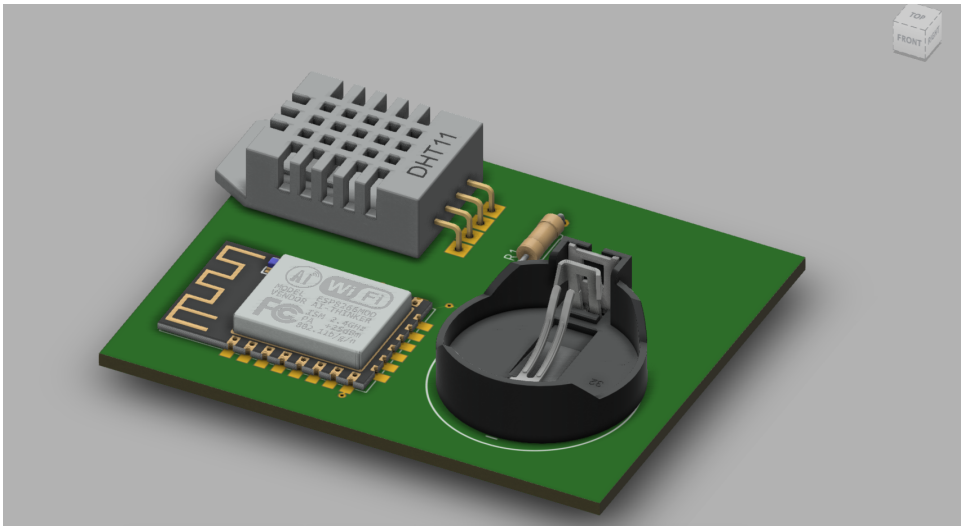
Esquemático



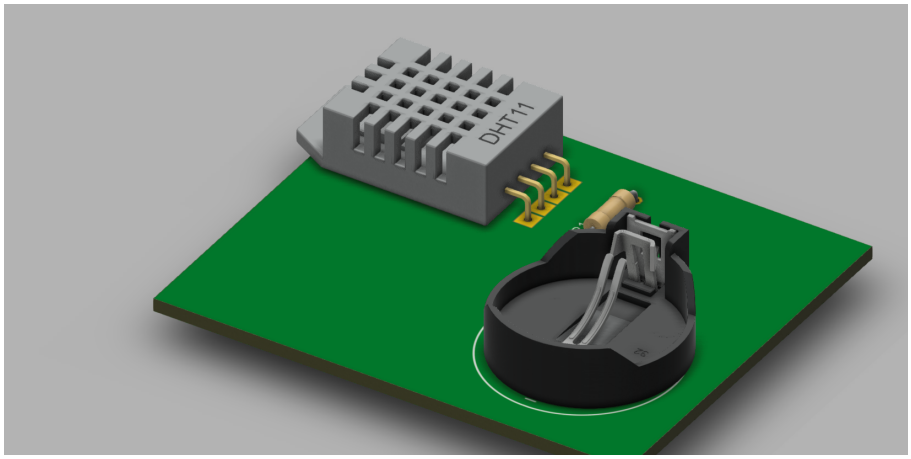
A figura acima mostra o esquema criado que representa o que foi construído para o funcionamento do nosso sistema embarcado utilizando o ESP8266 e sensor de umidade e temperatura Dht11.

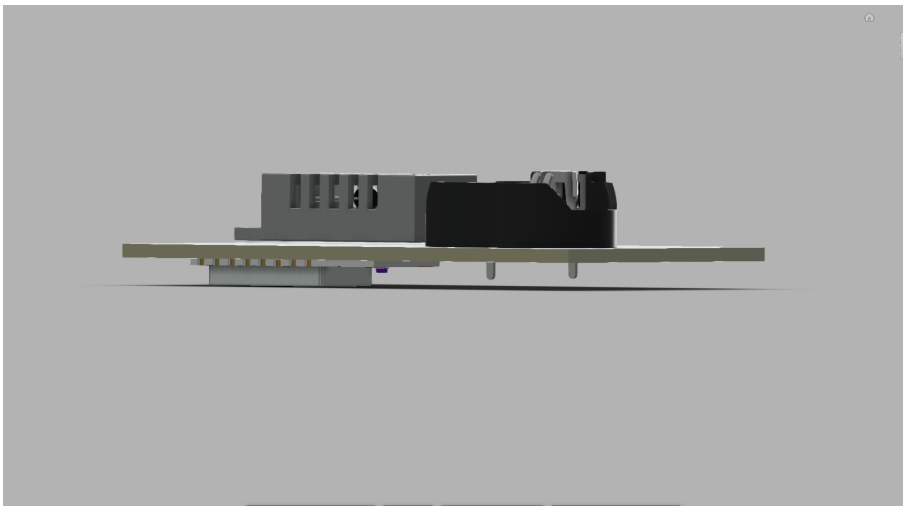
Placa 3D

Placa 1:



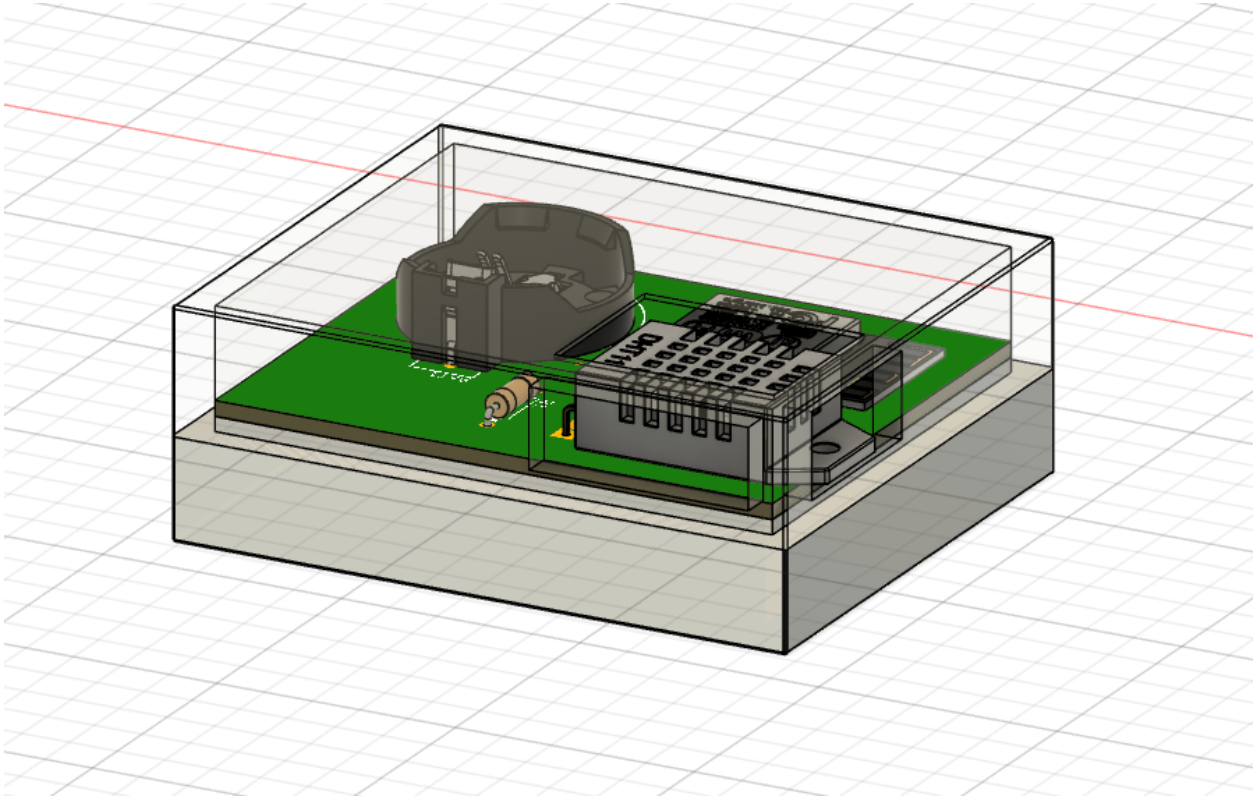
Placa 2:.





Estamos mostrando duas versões da placa porque o modelo desejado por nossa equipe e que acreditamos que seria mais adequado seria com todos os componentes na área superior da placa, que é representada pela figura placa, porém para isso precisamos utilizar rotas de ligação do tipo bottom e top. O segundo modelo, representado pela figura placa 2, foi possível fazer as ligações das rotas utilizando apenas o mapeador com o bottom, porém para isso foi preciso alterar a posição do ESP8266, onde ele precisou ficar na parte inferior da borda.

Case



A figura acima representa o protótipo 3D da case utilizada para proteger a placa 3D do projeto. A *case* conta com dois corpos estruturais, sendo uma tampa superior, transparente. A base da *case*, possui uma estrutura com apoio para a placa, para a mesma ter uma boa visibilidade na visão superior, não cair para o fundo ao ser inserida e possui também um bom material, rígido, para a proteção contra quedas ou impactos. O propósito das bordas arredondadas é evitar maiores danos em eventuais quedas ou impactos.