

# Experimento de Processamento de Sinais de Áudio integrado com App Blynk

## Objetivos de Aprendizado (OA)

**OA1** - Após ler as informações e assistir aos vídeos do Passo 1, os alunos serão capazes de descrever o que são parâmetros Chroma, e como eles podem ser usados para diferenciar músicas.

**OA2** - Após a realização das atividades dos Passos 2 e 3, os alunos serão capazes de usar a biblioteca pyAudioAnalysis para processar áudios, demonstrando o processamento de áudios diferentes para professores e monitores, por meio de interação com o notebook Python.

**OA3** - Após a realização das atividades dos Passos 4, 5, 6 e 7 os alunos serão capazes de enviar os parâmetros obtidos com o processamento do áudio para seu celular, e obter esta mesma informação usando requisições HTTP em uma plataforma IoT.

## Passo 0: Antes de Começar

Requisitos:

- Celular Android ou iOS
- Acesso aos materiais do Drive compartilhado

Execute a célula abaixo para baixar os arquivos de música que serão utilizados (cada execução corresponde a um arquivo baixado). A seguir é apresentada uma lista com o nome do arquivo e um ID do drive para você poder baixar este arquivo. Altere os campos *id* e *output* da célula abaixo para baixar o arquivo correspondente. OBS: Este processo de download deverá demorar entre 20s e 1 min até o arquivo baixado constar na aba "Arquivos".

- blues.wav - 1myiFwDDP6Bu152ETta9S3iRIIPdQepR\_
- classical.wav - 1BaLneAoporANi6T0crxmBSQAoOIDVS61
- country.wav - 1fSKqAPIYYYCuQ7mpQGR0VmoZ5adB8g7I
- eletronic.wav - 1-6Kjtr0PGMHtoFMtjrkv6c3sWmmzbhj1
- hiphop.wav - 16NLp0VQgMJ6VWcKmfWforyo7YJ5Rf8pt
- jazz.wav - 1llrnus07x7t5vqDhhW5FA\_5-L8oXJ4Nb
- pop.wav - 1StGRqiAqAi8KkvTsT0Rb\_8C8PqIF7YmW
- rock.wav - 1hmmvCVMbwDlG996XVlvkswe3nIUCCsXS

```
import gdown
```

```
output = "blues.wav"
id = "1myiFwDDP6Bu152ETta9S3iRIIPdQepR_"
url = "https://drive.google.com/uc?id=" + id
gdown.download(url, output, quiet=False)
```

```
output = "classical.wav"
id = "1BaLneAoporANi6T0crxmBSQAoOIDVS61"
url = "https://drive.google.com/uc?id=" + id
gdown.download(url, output, quiet=False)
```

```
output = "country.wav"
id = "1fSKqAPIYYYCuQ7mpQGR0VmoZ5adB8g7I"
```

```
url = "https://drive.google.com/uc?id=" + id
gdown.download(url, output, quiet=False)
```

```
output = "eletronic.wav"
id = "1-6Kjtr0PGMHtoFMtjrkv6c3sWmmzbhj1"
url = "https://drive.google.com/uc?id=" + id
gdown.download(url, output, quiet=False)
```

```
output = "hiphop.wav"
id = "16NLp0VQgMJ6VWcKmfWforyo7YJ5Rf8pt"
url = "https://drive.google.com/uc?id=" + id
gdown.download(url, output, quiet=False)
```

```
output = "jazz.wav"
id = "1Ilrnus07x7t5vqDhhW5FA_5-L8oXJ4Nb"
url = "https://drive.google.com/uc?id=" + id
gdown.download(url, output, quiet=False)
```

```
output = "pop.wav"
id = "1StGRqiAqAi8KkvTsT0Rb_8C8PqIF7YmW"
url = "https://drive.google.com/uc?id=" + id
gdown.download(url, output, quiet=False)
```

```
output = "rock.wav"
id = "1hmmvCVMbWdlg996XVlvkswe3nIUCCsXS"
url = "https://drive.google.com/uc?id=" + id
gdown.download(url, output, quiet=False)
```

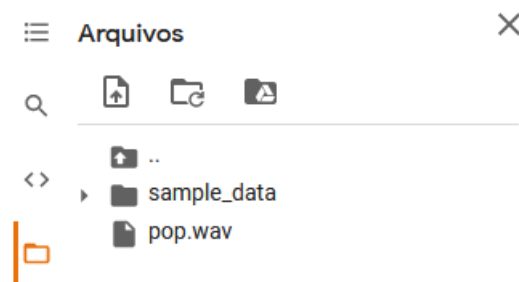
```
Downloading...
From: https://drive.google.com/uc?id=1myiFwDDP6Bu152ETta9S3iRlIPdQepR\_
To: /content/blues.wav
48.4MB [00:00, 90.0MB/s]
Downloading...
From: https://drive.google.com/uc?id=1BalneAoporANi6T0crxmBSQAo0IDVS61
To: /content/classical.wav
31.0MB [00:00, 80.0MB/s]
Downloading...
From: https://drive.google.com/uc?id=1fSKqAP1YYCuQ7mpQGR0VmoZ5adB8g7I
To: /content/country.wav
16.3MB [00:00, 58.1MB/s]
Downloading...
From: https://drive.google.com/uc?id=1-6Kjtr0PGMHtoFMtjrkv6c3sWmmzbhj1
To: /content/eletronic.wav
41.3MB [00:01, 32.6MB/s]
Downloading...
From: https://drive.google.com/uc?id=16NLp0VQgMJ6VWcKmfWforyo7YJ5Rf8pt
To: /content/hiphop.wav
44.6MB [00:00, 53.6MB/s]
Downloading...
From: https://drive.google.com/uc?id=1Ilrnus07x7t5vqDhhW5FA\_5-L8oXJ4Nb
To: /content/jazz.wav
47.4MB [00:00, 82.3MB/s]
Downloading...
From: https://drive.google.com/uc?id=1StGRqiAqAi8KkvTsT0Rb\_8C8PqIF7YmW
To: /content/pop.wav
28.9MB [00:00, 77.7MB/s]
Downloading...
From: https://drive.google.com/uc?id=1hmmvCVMbWdlg996XVlvkswe3nIUCCsXS
To: /content/rock.wav
28.7MB [00:00, 61.9MB/s]
'rock.wav'
```

```
import gdown
output = "pop.wav"
id = "1StGRqiAqAi8KkvTsT0Rb_8C8PqIF7YmW"
url = "https://drive.google.com/uc?id=" + id
gdown.download(url, output, quiet=False)
```

Downloading...

From: [https://drive.google.com/uc?id=1StGRqiAqAi8KkvTsT0Rb\\_8C8PqIF7YmW](https://drive.google.com/uc?id=1StGRqiAqAi8KkvTsT0Rb_8C8PqIF7YmW)

Observe que o arquivo deverá constar na aba lateral esquerda, conforme figura a seguir, onde foi baixado o arquivo "pop.wav"



## Instalação das bibliotecas necessárias

```
! pip install hmmlearn
! pip install simplejson
! pip install pydub
! pip install eyed3
```

```
from scipy import signal
```

```
! pip install pyAudioAnalysis
```

```
from pyAudioAnalysis.audioBasicIO import stereo_to_mono
from pyAudioAnalysis.audioBasicIO import read_audio_file
from pyAudioAnalysis.ShortTermFeatures import feature_extraction
```

Collecting hmmlearn

Downloading <https://files.pythonhosted.org/packages/4b/98/a2829aeb942b7146034d497afb3fc738a78a4fbd4797a03/>  
|████████████████████| 378kB 7.3MB/s

Requirement already satisfied: numpy>=1.10 in /usr/local/lib/python3.7/dist-packages (from hmmlearn) (1.19.1)

Requirement already satisfied: scipy>=0.19 in /usr/local/lib/python3.7/dist-packages (from hmmlearn) (1.4.1)

Requirement already satisfied: scikit-learn>=0.16 in /usr/local/lib/python3.7/dist-packages (from hmmlearn)

Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.16)

Installing collected packages: hmmlearn

Successfully installed hmmlearn-0.2.5

Collecting simplejson

Downloading <https://files.pythonhosted.org/packages/a8/04/377418ac1e530ce2a196b54c6552c018fd1fe776718053/>  
|████████████████████| 133kB 6.8MB/s

Installing collected packages: simplejson

Successfully installed simplejson-3.17.2

Collecting pydub

Downloading <https://files.pythonhosted.org/packages/a6/53/d78dc063216e62fc55f6b2eebb447f6a4b0a59f55c84063/>

Installing collected packages: pydub

Successfully installed pydub-0.25.1

Collecting eyed3

Downloading <https://files.pythonhosted.org/packages/e9/5e/9445f42ca86bc4832bac33fe37b506fc4db60dd9b68621e/>  
|████████████████████| 256kB 7.5MB/s

Collecting filetype<2.0.0,>=1.0.7

Downloading <https://files.pythonhosted.org/packages/b4/6b/7bc015da1a576ac037582ae0c5ac675371de9e017e8609/>

Collecting coverage[toml]<6.0.0,>=5.3.1

Downloading <https://files.pythonhosted.org/packages/16/e0/fc9f7bd9b84e6b41d0aad1a113e36714aac0ca9b307aca/>  
|████████████████████| 245kB 13.4MB/s

Collecting deprecation<3.0.0,>=2.1.0

Downloading <https://files.pythonhosted.org/packages/02/c3/253a89ee03fc9b9682f1541728eb66db7db22148cd94f89/>

Requirement already satisfied: toml; extra == "toml" in /usr/local/lib/python3.7/dist-packages (from coverage[toml]<6.0.0,>=5.3.1)

Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (from deprecation<3.0.0,>=2.1.0)

Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging>=20.0)

ERROR: datascience 0.10.6 has requirement coverage==3.7.1, but you'll have coverage 5.5 which is incompatible

ERROR: datascience 0.10.6 has requirement folium==0.2.1, but you'll have folium 0.8.3 which is incompatible

ERROR: coveralls 0.5 has requirement coverage<3.999,>=3.6, but you'll have coverage 5.5 which is incompatible

Installing collected packages: filetype, coverage, deprecation, eyed3

Found existing installation: coverage 3.7.1

Uninstalling coverage-3.7.1:

Successfully uninstalled coverage-3.7.1

Successfully installed coverage-5.5 deprecation-2.1.0 eyed3-0.9.6 filetype-1.0.7

```
Collecting pyAudioAnalysis
  Downloading https://files.pythonhosted.org/packages/4a/8b/456ef9101ad5bffe60b84ed9f6efba7d5a6f62e7da16dea/
  |████████████████████████████████████████| 52.4MB 92kB/s
Building wheels for collected packages: pyAudioAnalysis
  Building wheel for pyAudioAnalysis (setup.py) ... done
  Created wheel for pyAudioAnalysis: filename=pyAudioAnalysis-0.3.7-cp37-none-any.whl size=52589873 sha256=
  Stored in directory: /root/.cache/pip/wheels/e0/da/f5/f8ab47859ae6e5f552d4fc289cc7af54879890d22bccd3d3cf
Successfully built pyAudioAnalysis
Installing collected packages: pyAudioAnalysis
Successfully installed pyAudioAnalysis-0.3.7
```

## ▼ Passo 1: Introdução aos Parâmetros Chroma

Assista ao vídeo abaixo sobre MQTT, e em seguida responda às perguntas dos professores/monitores.

```
from IPython.display import HTML
```

```
HTML('<iframe width="560" height="315" src="https://www.youtube.com/embed/iY243jku0UA" title="YouTube video playe
```



Veja a tabela abaixo com os parâmetros que a biblioteca pyAudioAnalysis nos fornece.

Se atente para o Chroma Vector!

Feature ID	Feature Name	Description
1	Zero Crossing Rate	The rate of sign-changes of the signal during the duration of a particular frame.
2	Energy	The sum of squares of the signal values, normalized by the respective frame length.
3	Entropy of Energy	The entropy of sub-frames' normalized energies. It can be interpreted as a measure of abrupt changes.
4	Spectral Centroid	The center of gravity of the spectrum.
5	Spectral Spread	The second central moment of the spectrum.
6	Spectral Entropy	Entropy of the normalized spectral energies for a set of sub-frames.
7	Spectral Flux	The squared difference between the normalized magnitudes of the spectra of the two successive frames.
8	Spectral Rolloff	The frequency below which 90% of the magnitude distribution of the spectrum is concentrated.
9-21	MFCCs	Mel Frequency Cepstral Coefficients form a cepstral representation where the frequency bands are not linear but distributed acco
22-33	Chroma Vector	A 12-element representation of the spectral energy where the bins represent the 12 equal-tempered pitch classes of western-ty
34	Chroma Deviation	The standard deviation of the 12 chroma coefficients.

## ▼ Passo 2: Processamento do Áudio de uma Música

Download de músicas em MP3 com licença Creative Commons do site: <https://freemusicarchive.org/>

- **Genre - Author - Music Name**

- blues - Pierce Murphy - Ashes Of Paradise
- classical - Crowander - Jerry's Back
- country - Lobo Loco - Verona - Intro (ID 1407)
- electronic - Xylo Ziko - Alternate
- hiphop - Eaters - Dogstarmegalazer
- jazz - Pierce Murphy - Among The Stars
- pop - Scott Holmes Music - We Are One
- rock - Blue Wave Theory - Jazz Hole

Foi necessária conversão de MP3 para WAV para utilizar a biblioteca pyAudioAnalysis

Escolha um dos arquivos de música que foram baixados:

```
file = "pop.wav"
```

Processamento do Áudio e visualização do tempo necessário (em segundos):

OBS: Esta etapa levará em média de 1 a 3 minutos, dependendo da música escolhida

```
import time

start = time.time()

[Fs, x] = read_audio_file(file);

signal = stereo_to_mono(x);

F = feature_extraction(signal, 44100, 1024, 128, deltas=False);

"""

    feature_extraction(signal, sampling_rate, window, step, deltas=True)

    This function implements the short-term windowing process.
    For each short-term window a set of features is extracted.
    This results to a sequence of feature vectors, stored in a np matrix.

    ARGUMENTS
        signal:          the input signal samples
        sampling_rate:    the sampling freq (in Hz)
        window:          the short-term window size (in samples)
        step:            the short-term window step (in samples)
        deltas:          (opt) True/False if delta features are to be
                        computed

    RETURNS
        features (numpy.ndarray):    contains features
                                    (n_feats x numOfShortTermWindows)
        feature_names (numpy.ndarray): contains feature names
                                    (n_feats x numOfShortTermWindows)
"""

features = F[0][21:33]

end = time.time()

print(end - start)

112.24778509140015
```

Exemplo de um dos 12 vetores de parâmetros obtidos:

```
features[0]
```

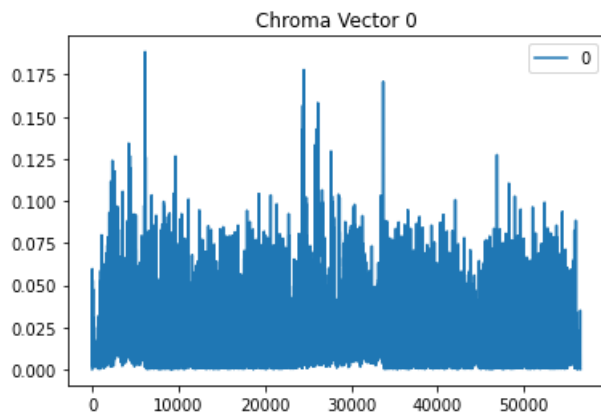
```
array([0.          , 0.0145498 , 0.00808195, ..., 0.          , 0.          ,  
       0.          ])
```

### ▼ Passo 3: Visualização dos Parâmetros Obtidos

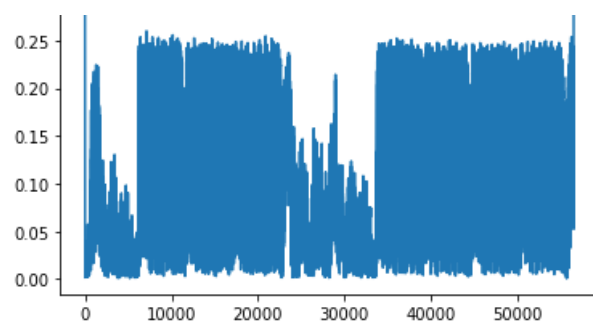
```
import matplotlib.pyplot as plt
```

```
def plot(features,i):  
    plt.title("Chroma Vector " + str(i))  
    plt.plot(features[i], label=i)  
    plt.legend(loc="upper right")  
    plt.show()
```

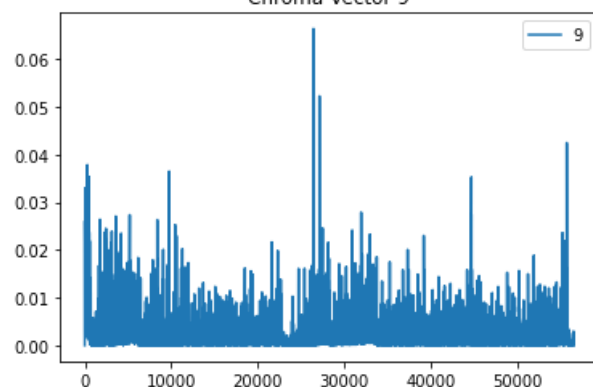
```
plot(features,0)
```



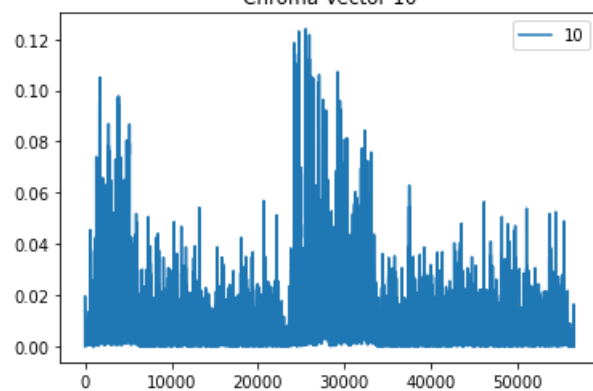
```
plot(features,0)  
plot(features,1)  
plot(features,2)  
plot(features,3)  
plot(features,4)  
plot(features,5)  
plot(features,6)  
plot(features,7)  
plot(features,8)  
plot(features,9)  
plot(features,10)  
plot(features,11)
```



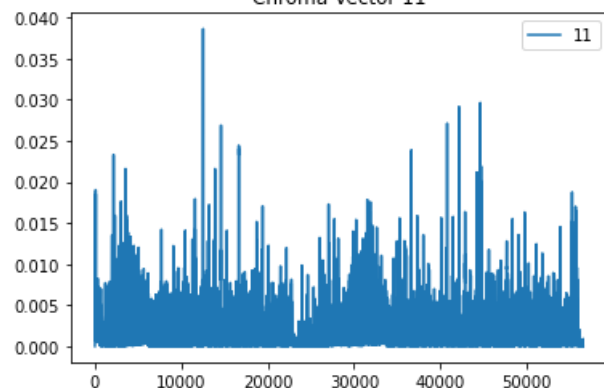
Chroma Vector 9



Chroma Vector 10



Chroma Vector 11







#### ▼ Passo 4: Instalar e Criar Conta no Aplicativo Blynk

Procure por "Blynk" na Play Store, e o instale no seu sistema Android.

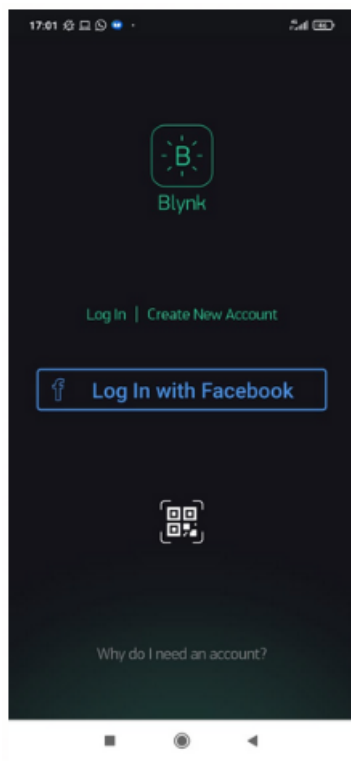
[https://play.google.com/store/apps/details?id=cc.blynk&hl=pt\\_BR&gl=US](https://play.google.com/store/apps/details?id=cc.blynk&hl=pt_BR&gl=US)

Ele também está disponível na App Store para usuários iOS

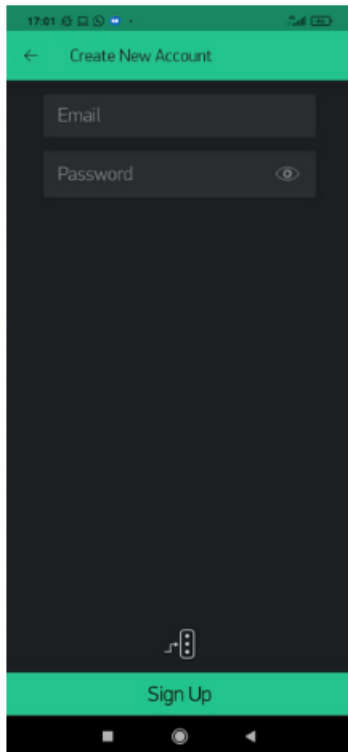
<https://apps.apple.com/br/app/blynk-iot-for-arduino-esp32/id808760481>



Ao abrir o Blynk você verá a seguinte tela:

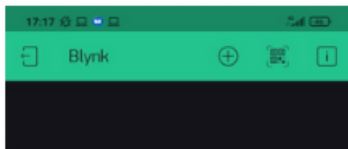


Clique em *Create New Account*, onde você deverá informar um email e uma senha para criar sua conta. Após clicar em *Sign Up* sua conta será criada e você receberá um email de confirmação. Após a criação da conta já é feito automaticamente o *Log In* no aplicativo

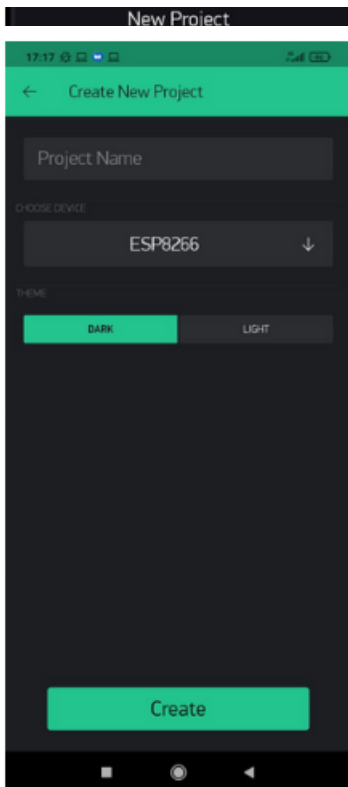


## ▼ Passo 5: Criando um Projeto no Blynk

Ao realizar *login* no Blynk a seguinte tela é apresentada



Clique em *New Project*, para criar um novo projeto, mostrando a seguinte tela

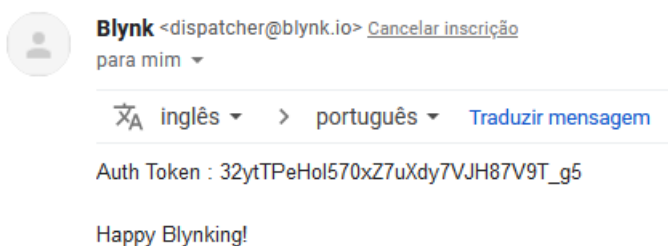


Escolha as seguintes opções:

- **Project Name:** Projeto
- **Choose Device:** ESP8266
- **Conection Type:** Wi-Fi

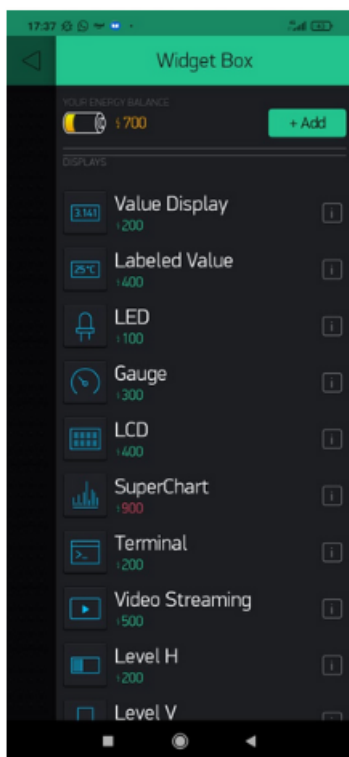
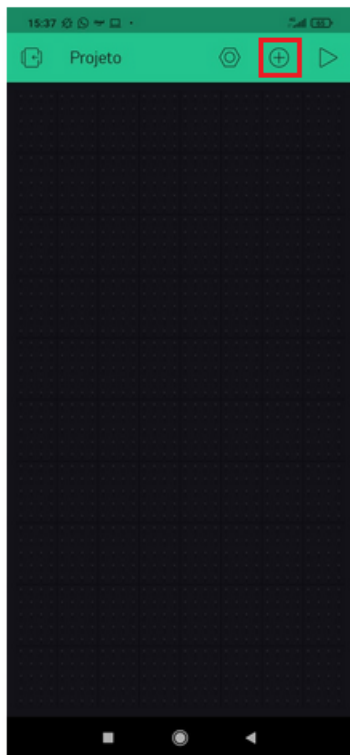
OBS: Neste projeto não iremos usar um dispositivo em específico, nos limitando apenas a comunicação entre a CPU e o Blynk, desta forma, a escolha do "ESP8266" foi arbitrária.

Após a criação do projeto você irá receber um email contendo o **token** do projeto. Este **token** é necessário para realizar a comunicação via HTTP Request na nuvem do Blynk e posteriormente será configurado neste notebook.



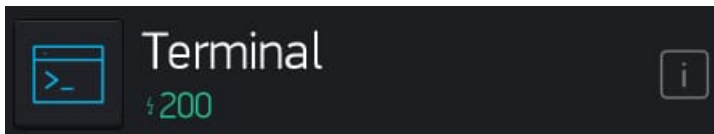
Iremos agora adicionar um *Widget* em nosso projeto para receber os dados correspondentes aos parâmetros extraídos do áudio.

Com a tela do projeto aberta, clique no botão contendo um + no canto superior direito para abrir a *Widget Box*, onde é possível selecionar *Widgets* para o seu projeto

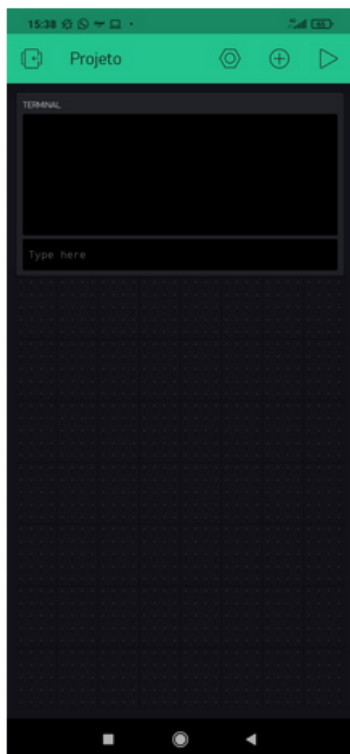


Aqui é importante destacar o conceito de "energia" do Blynk. Toda conta gratuita como a aqui criada possui 2000 pontos de energia para uso no Blynk. A adição de *Widgets* no projeto consome esses pontos de energia, sendo que os *Widgets* podem ser posteriormente reciclados (excluídos) para recuperar estes pontos.

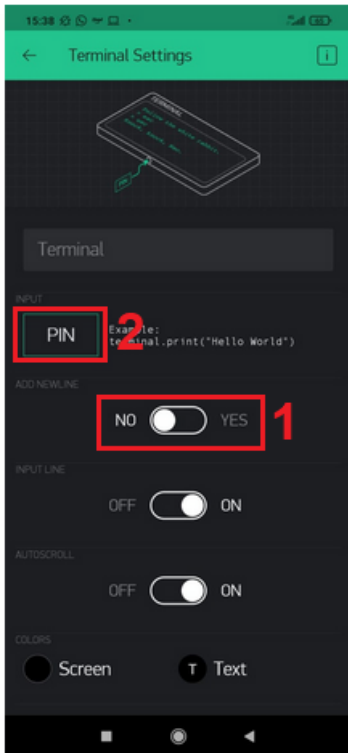
Role para baixo até a seção *Displays* e crie um *Widget* do tipo *Terminal*



A tela do projeto então ficará da seguinte forma:



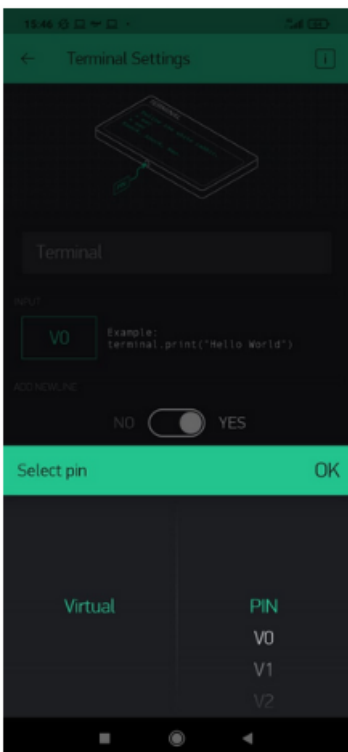
Será necessário agora configurar o terminal. Clique nele, fazendo com que abra uma tela de *Settings* dele.



Escolha a seguinte opção:

- **Add Newline:** Yes

Em seguida clique em PIN para configurar qual o pino que este botão irá controlar, o que irá apresentar uma aba

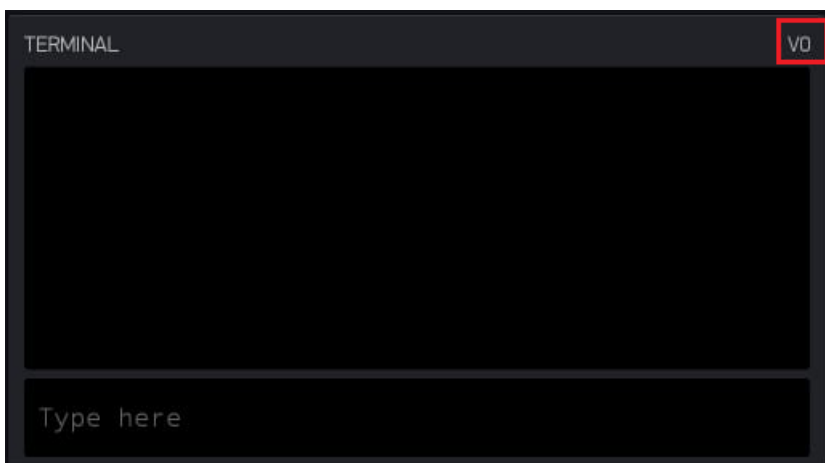
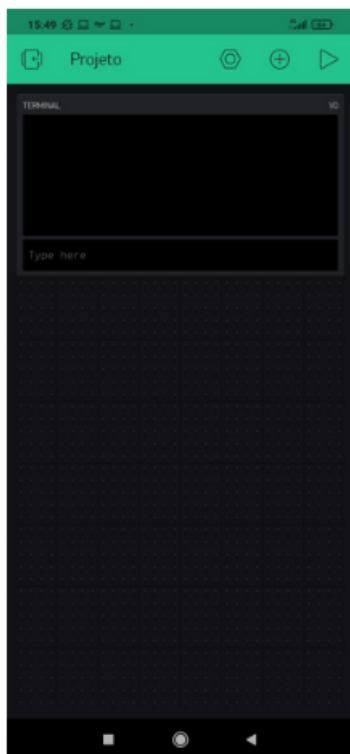


O terminal só possui a opção de ser mapeado em um pino virtual. Os **pinos virtuais** são uma abstração desenvolvida pelo Blynk possíveis graças a infraestrutura em nuvem. Aqui não existe um mapeamento físico do pino, mas sim um mapeamento **virtual**, onde determinado valor é escrito/lido em uma região da nuvem do Blynk.

Maiores informações sobre o funcionamento dos pinos virtuais podem ser obtidas no link a seguir:

<http://help.blynk.cc/en/articles/512061-what-is-virtual-pins>

Nesta aba selecione o pino **V0**. Em seguida clique em OK e na seta (←) localizada no canto superior esquerdo da tela para voltar ao menu do projeto. Perceba que o mapeamento do terminal já é apresentado



Clique no botão de *play* localizado no canto superior direito da tela para iniciar o seu projeto, permitindo com que o seu terminal receba os dados.

OBS: Após clicar neste botão ele é substituído por um botão de *stop*, que pode ser utilizado para parar a execução do projeto. Quando o projeto está parado, o terminal irá parar de receber eventuais dados.



## ▼ Passo 6: Configuração do Aplicativo Blynk

Token de Autenticação do Projeto do Blynk recebido no email cadastrado

```
#auth_token = "1s4QAm2L2Lj_I1NDEfyPQu5AO_Ftt4mL"  
#auth_token = "32ytTPeHoI570xZ7uXdy7VJH87V9T_g5"  
auth_token = "Xrt0vyJ787ho3Q0KpZq-LXBhXgMc46Ts"
```

Pino Virtual onde foi mapeado o Terminal Virtual no App Blynk

```
virtual_pin = "V0"
```

Teste Inicial com Requisição HTTP pelo próprio navegador. Clique no link que será gerado para enviar a string "oi" para o terminal do seu projeto.

```
print('http://blynk-cloud.com/'+auth_token+'/update/'+virtual_pin+'?value=0i')
```

<http://blynk-cloud.com/Xrt0vyJ787ho3Q0KpZq-LXBhXgMc46Ts/update/V0?value=0i>



## ▼ Passo 7: Interação com a Plataforma IoT por meio de Requisição HTTP

Biblioteca para requisição HTTP:

```
!pip install requests
```

```
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (2.23.0)  
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests)  
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests)  
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests) (2.10)  
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests)
```

Exemplo de requisição para escrever 'Ola' no Aplicativo Blynk

```
import requests  
  
blynk_server = '45.55.96.146'  
  
r = requests.get('http://'+blynk_server+'/'+auth_token+'/update/'+virtual_pin+'?value=0la')
```