UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
TECNOLOGIA EM SISTEMAS PARA INTERNET

CAMPOS, ESTER DE ARRUDA

PROJETO E GERENCIAMENTO DE UMA IOT

TRABALHO 4

GUARAPUAVA

2016

CAMPOS, ESTER DE ARRUDA

PROJETO E GERENCIAMENTO DE UMA IOT

Projeto apresentado ao professor Dr. Hermano Pereira como requisito parcial para composição de nota semestral de sua respectivas disciplina.

GUARAPUAVA

2016

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

1.1 JUSTIFICATIVA

1.2 OBJETIVO

O presente projeto tem por objetivo fazer o projeto e gerenciamento de uma rede da Internet das Coisas. Desenvolver uma aplicação remota para gerenciar as lâmpadas de uma planta arquitetonica da Internet das Coisas. Parametrizamos a elaboração deste trabalho nos pre-requisitos listados pelo docente da disciplina, conforme está definido no sumário deste.

1.3 ESTRUTURA DO PROJETO

O Capítulo 1 – Introdução, irá definir o que vem a ser o projeto e quais os objetivos a serem analisados. No Capítulo 2, estaremos apresentando as exigências Gerais definidas como requisito para elaboração do projeto. O Capítulo 3, apresentará respostas para as exigências específicas e seus devidos tópicos. Por fim, no Capítulo 4 apresentaremos alguns trechos de código.

Toda formatação para este texto incluindo a formatação para apresentação dos códigos elaborados foram feitos utilizando a linguagem *LaTeX*.

1.4 METODOLOGIA UTILIZADA

Utilizaremos o método de pesquisa aplicada cuja meta é contribuir para fins práticos buscando solução para problema concreto.

2 EXIGÊNCIAS GERAIS

2.1 CONTEÚDO

2.2 REQUISITOS

Neste projeto foram utilizados *TKINTER*, *Python* e *PHP*, além do *Cooja* e *Contiki*.

2.3 LAYOUT TKINTER

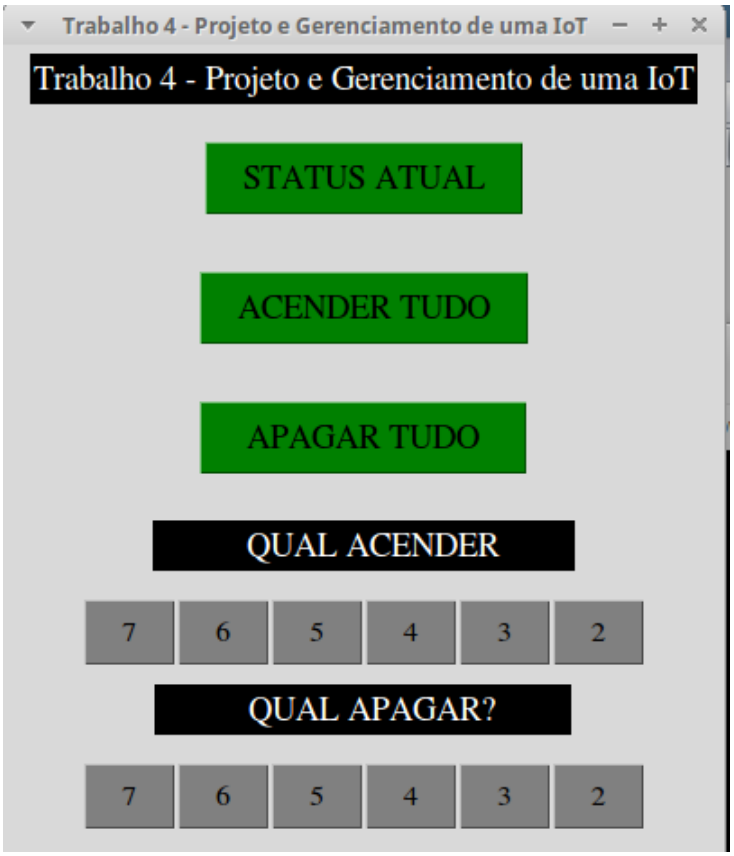

Figura 5: Tela principal

O modulo TKinter ou *TK interface* é uma biblioteca para interface gráfica que é padrão da liguagem *Python*. Seguindo a idéia da liguagem *Python* ela é bem simples de se

aprender e utilizar.

2.4 LAYOUT PHP

Abaixo listamos algumas imagens que apresentam o layout para página. Mais testes podem ser realizados durante a apresentação deste projeto.

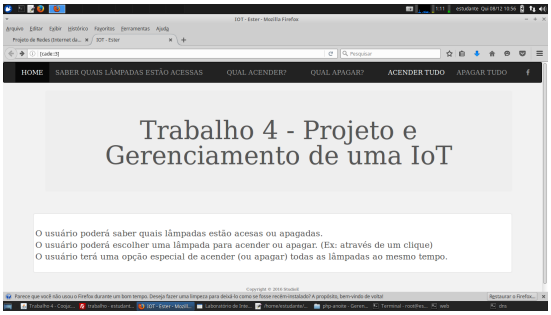

Figura 6: Tela inicial

Figura 7: Escolhendo

3 EXIGÊNCIAS ESPECÍFICAS

3.1 PROJETO DA REDE

3.2 GERENCIAMENTO DA REDE

Abaixo uma idéia da estrutura da rede.

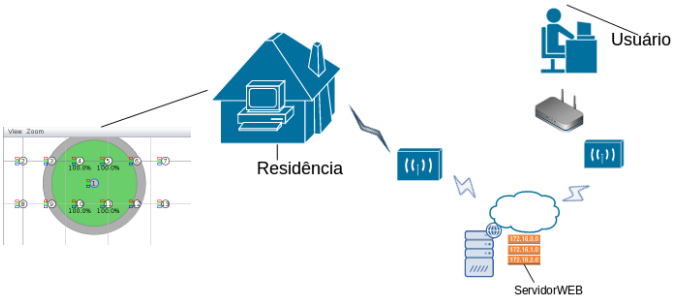

Figura 9: Para o modelo PHP

3.2.1 PLANTA BAIXA LOCAL

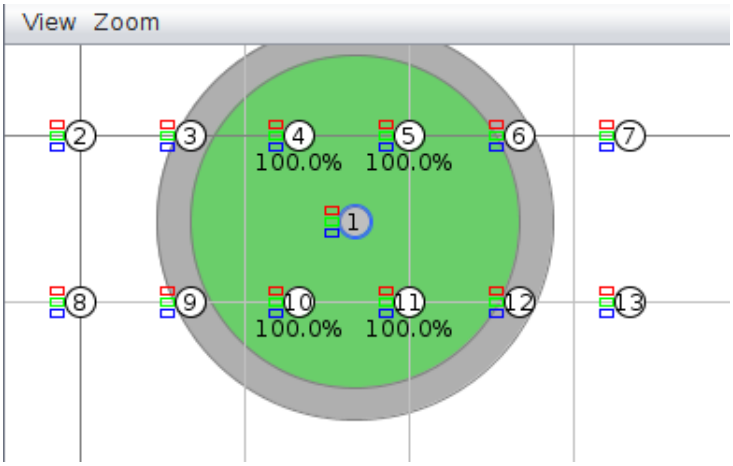

Figura 10:

Na figura acima considere que os nós 2, 3, 4, 5, 6 e 7 representam as lâmpadas e o nós 8, 9, 10, 11, 12 e 13 representam os interruptores. Os interruptores acionam as respectivas lâmpadas na sequência $8 -> 2$, $9 -> 3$ e sucessivamente.

3.3 CONFIGURAÇÕES RÁPIDAS

Dicas e etapas que as vezes esquecemos:

No terminal entrar na pasta do contiki e executar:

```
# ./tunslip6 -a 127.0.0.1 -p 60001 cafe::1/64
```

Iniciar a simulação no *cooja* que deve ter o *border router*

No navegador é bom ter instalado o plugin *Copper*.

Instale as *libs aiocaop* para utilizar no *Python*:

```
$ sudo apt-get update
$ sudo apt-get install python3-aiocoap
```

4 CÓDIGO

4.1 IMPLEMENTAÇÃO COM TKINTER

Apresentamos abaixo o arquivo principal que foi utilizado na implementação. Utilizamos o *TKinter*. Neste arquivo foram realizadas configurações para que seja criado uma janela gráfica com dimensões 400*x*450, nesta janela estão dispostos 3 botões com as respectivas funções mostradas nos subitens abaixo.

```
1  import os
2  import subprocess
3  import tkMessageBox
4  from Tkinter import *
5  class Janela:
6      def __init__(self,toplevel):
7          print "Inicio"
8          toplevel.title('Trabalho 4 - Projeto e Gerenciamento de uma IoT
               ')
9          toplevel.geometry("400x450")
10         self.fr2 = Frame(toplevel, pady=5)
11         self.fr2.pack()
12         self.botao12 = Label(self.fr2, text='Trabalho 4 - Projeto e
               Gerenciamento de uma IoT', background='black', fg='white',
               font=('Times','14'))
13         self.botao12.pack(side=LEFT)
14         self.fr2 = Frame(toplevel, pady=15)
15         self.fr2.pack()
16         self.botao2 = Button(self.fr2, text='STATUS ATUAL', background=
               'green', font=('Times','14'), command=clicado)
17         self.botao2.pack(side=RIGHT)
18         self.fr2 = Frame(toplevel, pady=15)
19         self.fr2.pack()
20         self.botao2 = Button(self.fr2, text='ACENDER TUDO', background=
               'green', font=('Times','14'), command=acender)
21         self.botao2.pack(side=LEFT)
```

```
22          self.fr2 = Frame(toplevel, pady=15)
23          self.fr2.pack()
24          self.botao2 = Button(self.fr2, text=' APAGAR TUDO ',
               background='green', font=('Times','14'), command=apagar)
25          self.botao2.pack(side=LEFT)
26          self.fr2 = Frame(toplevel, pady=10)
27          self.fr2.pack()
28          self.botao2 = Label(self.fr2, text='          QUAL ACENDER
                    ', background='BLACK', fg='white', font=('Times
               ','14'))
29          self.botao2.pack(side=LEFT)
30          self.fr2 = Frame(toplevel, pady=5)
31          self.fr2.pack()
32          self.botao2 = Button(self.fr2, text='2', background='gray',
               font=('Times','12'), command=lampada1)
33          self.botao2.pack(side=RIGHT)
34          self.botao3 = Button(self.fr2, text='3', background='gray',
               font=('Times','12'), command=lampada2)
35          self.botao3.pack(side=RIGHT)
36          self.botao4 = Button(self.fr2, text='4', background='gray',
               font=('Times','12'), command=lampada3)
37          self.botao4.pack(side=RIGHT)
38          self.botao5 = Button(self.fr2, text='5', background='gray',
               font=('Times','12'), command=lampada4)
39          self.botao5.pack(side=RIGHT)
40          self.botao6 = Button(self.fr2, text='6', background='gray',
               font=('Times','12'), command=lampada5)
41          self.botao6.pack(side=RIGHT)
42          self.botao7 = Button(self.fr2, text='7', background='gray',
               font=('Times','12'), command=lampada6)
43          self.botao7.pack(side=RIGHT)
44          self.fr2 = Frame(toplevel, pady=5)
45          self.fr2.pack()
46          self.botao2 = Label(self.fr2, text='          QUAL APAGAR?
                    ', background='black', fg='white', font=('Times
               ','14'))
47          self.botao2.pack(side=LEFT)
48          self.fr2 = Frame(toplevel, pady=10)
49          self.fr2.pack()
50          self.botao2 = Button(self.fr2, text='2', background='gray',
               font=('Times','12'), command=lampada11)
51          self.botao2.pack(side=RIGHT)
```

```
52          self.botao3 = Button(self.fr2, text='3', background='gray',
                font=('Times','12'), command=lampada21)
53          self.botao3.pack(side=RIGHT)
54          self.botao4 = Button(self.fr2, text='4', background='gray',
                font=('Times','12'), command=lampada31)
55          self.botao4.pack(side=RIGHT)
56          self.botao5 = Button(self.fr2, text='5', background='gray',
                font=('Times','12'), command=lampada41)
57          self.botao5.pack(side=RIGHT)
58          self.botao6 = Button(self.fr2, text='6', background='gray',
                font=('Times','12'), command=lampada51)
59          self.botao6.pack(side=RIGHT)
60          self.botao7 = Button(self.fr2, text='7', background='gray',
                font=('Times','12'), command=lampada61)
61          self.botao7.pack(side=RIGHT)
62  def clicado():
63      pastaDoScript = os.path.dirname(os.path.realpath(__file__))
64      x = subprocess.Popen(['python3', pastaDoScript + '/ledstatus-all.py
            ', '3'], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
65      retornoScript, erros = x.communicate()
66      if erros:
67          print "Ocorreu algum erro.\nErro:" + erros
68      print retornoScript
69      tkMessageBox.showinfo('----> Status de todas as luzes <-----',
            retornoScript)
70  def acender():
71      pastaDoScript = os.path.dirname(os.path.realpath(__file__))
72      x = subprocess.Popen(['python3', pastaDoScript + '/
            toggle2-lightup.py', '3'], stdout=subprocess.PIPE,
            stderr=subprocess.PIPE)
73      retornoScript, erros = x.communicate()
74      if erros:
75          print "Ocorreu algum erro.\nErro:" + erros
76      print retornoScript
77      tkMessageBox.showinfo('Ligando todas as luzes', retornoScript)
78  def apagar():
79      pastaDoScript = os.path.dirname(os.path.realpath(__file__))
80      x = subprocess.Popen(['python3', pastaDoScript + '/
            toggle2-lightdown.py', '3'], stdout=subprocess.PIPE,
            stderr=subprocess.PIPE)
81      retornoScript, erros = x.communicate()
82      if erros:
83          print "Ocorreu algum erro.\nErro:" + erros
```

```
84      print retornoScript
85      tkMessageBox.showinfo('Desligando todas as luzes', retornoScript)
86  def escolher():
87      pastaDoScript = os.path.dirname(os.path.realpath(__file__))
88      x = subprocess.Popen(['python3', pastaDoScript + '/teste4.py',
            '3'], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
89      retornoScript, erros = x.communicate()
90      if erros:
91          print "Ocorreu algum erro.\nErro:" + erros
92      print retornoScript
93      tkMessageBox.showinfo('Escolhendo qual luz ligar', retornoScript)
94  def lampada1():
95      pastaDoScript = os.path.dirname(os.path.realpath(__file__))
96      x = subprocess.Popen(['python3', pastaDoScript + '/ligand.py',
            '2'], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
97      retornoScript, erros = x.communicate()
98      if erros:
99          print "Ocorreu algum erro.\nErro:" + erros
100     print retornoScript
101     tkMessageBox.showinfo('----> Ligando <-----', retornoScript)
102 def lampada2():
103     pastaDoScript = os.path.dirname(os.path.realpath(__file__))
104     x = subprocess.Popen(['python3', pastaDoScript + '/ligand.py',
            '3'], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
105     retornoScript, erros = x.communicate()
106     if erros:
107         print "Ocorreu algum erro.\nErro:" + erros
108     print retornoScript
109     tkMessageBox.showinfo('----> Ligando <-----', retornoScript)
110 def lampada3():
111     pastaDoScript = os.path.dirname(os.path.realpath(__file__))
112     x = subprocess.Popen(['python3', pastaDoScript + '/ligand.py',
            '4'], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
113     retornoScript, erros = x.communicate()
114     if erros:
115         print "Ocorreu algum erro.\nErro:" + erros
116     print retornoScript
117     tkMessageBox.showinfo('----> Ligando <-----', retornoScript)
118 def lampada4():
119     pastaDoScript = os.path.dirname(os.path.realpath(__file__))
120     x = subprocess.Popen(['python3', pastaDoScript + '/ligand.py',
            '5'], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
121     retornoScript, erros = x.communicate()
```

```
122     if erros:
123         print "Ocorreu algum erro.\nErro:" + erros
124     print retornoScript
125     tkMessageBox.showinfo('----> Ligando <-----', retornoScript)
126 
127 def lampada5():
128     pastaDoScript = os.path.dirname(os.path.realpath(__file__))
129     x = subprocess.Popen(['python3', pastaDoScript + '/ligand.py',
            '6'], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
130     retornoScript, erros = x.communicate()
131     if erros:
132         print "Ocorreu algum erro.\nErro:" + erros
133     print retornoScript
134     tkMessageBox.showinfo('----> Ligando <-----', retornoScript)
135 def lampada6():
136     pastaDoScript = os.path.dirname(os.path.realpath(__file__))
137     x = subprocess.Popen(['python3', pastaDoScript + '/ligand.py',
            '7'], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
138     retornoScript, erros = x.communicate()
139     if erros:
140         print "Ocorreu algum erro.\nErro:" + erros
141     print retornoScript
142     tkMessageBox.showinfo('----> Ligando <-----', retornoScript)
143 def lampada11():
144     pastaDoScript = os.path.dirname(os.path.realpath(__file__))
145     x = subprocess.Popen(['python3', pastaDoScript + '/desligand.py',
            '2'], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
146     retornoScript, erros = x.communicate()
147     if erros:
148         print "Ocorreu algum erro.\nErro:" + erros
149     print retornoScript
150     tkMessageBox.showinfo('----> desLigando <-----', retornoScript)
151 
152 def lampada21():
153     pastaDoScript = os.path.dirname(os.path.realpath(__file__))
154     x = subprocess.Popen(['python3', pastaDoScript + '/desligand.py',
            '3'], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
155     retornoScript, erros = x.communicate()
156     if erros:
157         print "Ocorreu algum erro.\nErro:" + erros
158     print retornoScript
159     tkMessageBox.showinfo('----> desLigando <-----', retornoScript)
160 def lampada31():
```

```
161     pastaDoScript = os.path.dirname(os.path.realpath(__file__))
162     x = subprocess.Popen(['python3', pastaDoScript + '/desligand.py',
            '4'], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
163     retornoScript, erros = x.communicate()
164     if erros:
165         print "Ocorreu algum erro.\nErro:" + erros
166     print retornoScript
167     tkMessageBox.showinfo('----> desLigando <-----', retornoScript)
168 def lampada41():
169     pastaDoScript = os.path.dirname(os.path.realpath(__file__))
170     x = subprocess.Popen(['python3', pastaDoScript + '/desligand.py',
            '5'], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
171     retornoScript, erros = x.communicate()
172     if erros:
173         print "Ocorreu algum erro.\nErro:" + erros
174     print retornoScript
175     tkMessageBox.showinfo('----> desLigando <-----', retornoScript)
176 def lampada51():
177     pastaDoScript = os.path.dirname(os.path.realpath(__file__))
178     x = subprocess.Popen(['python3', pastaDoScript + '/desligand.py',
            '6'], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
179     retornoScript, erros = x.communicate()
180     if erros:
181         print "Ocorreu algum erro.\nErro:" + erros
182     print retornoScript
183     tkMessageBox.showinfo('----> desLigando <-----', retornoScript)
184 def lampada61():
185     pastaDoScript = os.path.dirname(os.path.realpath(__file__))
186     x = subprocess.Popen(['python3', pastaDoScript + '/desligand.py',
            '7'], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
187     retornoScript, erros = x.communicate()
188     if erros:
189         print "Ocorreu algum erro.\nErro:" + erros
190     print retornoScript
191     tkMessageBox.showinfo('----> desLigando <-----', retornoScript)
192 raiz=Tk()
193 Janela(raiz)
194 raiz.mainloop()
```

4.1.1 BOTÃO STATUS ATUAL

O botão *STATUS ATUAL* aciona/chama o script *Python− ledstatus − all.py* que por consequência irá consultar o estado de cada lâmpada na simulação e posteriormente mostrará ao usuário uma janela onde são mostrados os estados *ON* ou *OFF* das lâmpadas,

```
import logging
import sys
import asyncio
from aiocoap import *
logging.basicConfig(level=logging.INFO)
@asyncio.coroutine
def main():
  for num in range(2,8):
        protocol = yield from Context.create_client_context()
        request = Message(code=GET)
        address = "coap://[cafe::c30c:0:0:" + str(num) + "]/actuators/
           ledstatus"
        request.set_request_uri(address)
        try:
            response = yield from protocol.request(request).response
        except Exception as e:
            print('Failed to fetch resource:')
            print(e)
        else:
            print('Result: %s[barra-n]%r'%(response.code,
               response.payload))
            variavel = response.payload
            variavel = str(variavel)
            print (variavel[2:4])
if __name__ == "__main__":
   asyncio.get_event_loop().run_until_complete(main())
```

4.1.2 BOTÃO ACENDER TUDO

O botão *ACENDER TUDO* aciona/chama o script *Python− toggle2 − lightup.py* que por consequência irá consultar o estado de cada lâmpada na simulação, caso o estado da lâmpada seja *OFF* então no script foi adicionado a função de alternar o status da lâmpada para *ON*. Após a execução do script *Python− toggle2 − lightdown.py* uma janela de mensagem é mostrada ao usuário com o status de cada lâmpada da simulação.

```
import logging
```

```
2  import sys
3  import asyncio
4  from aiocoap import *
5  logging.basicConfig(level=logging.INFO)
6  @asyncio.coroutine
7  def main():
8    for num in range(2,8):
9          protocol = yield from Context.create_client_context()
10         request = Message(code=GET)
11         address = "coap://[cafe::c30c:0:0:" + str(num) + "]/actuators/
               ledstatus"
12         request.set_request_uri(address)
13         try:
14             response = yield from protocol.request(request).response
15         except Exception as e:
16             print('Failed to fetch resource:')
17             print(e)
18         else:
19             print('Result: %s[barra-n]%r'%(response.code,
                   response.payload))
20             variavel = response.payload
21             variavel = str(variavel)
22             print (variavel[2:5])
23             if variavel[2:5] == 'OFF':
24               address = "coap://[cafe::c30c:0:0:" + str(num) + "]/
                     actuators/toggle"
25               request = Message(code=POST)
26               request.set_request_uri(address)
27               try:
28                 response = yield from protocol.request(request).
                       response
29               except Exception as e:
30                 print('Failed to fetch resource:')
31                 print(e)
32               else:
33                 print('Ligando tudo')
34 if __name__ == "__main__":
35    asyncio.get_event_loop().run_until_complete(main())
```

4.1.3 BOTÃO APAGAR TUDO

O botão *APAGAR TUDO* aciona/chama o script *Python− toggle2 − lightdown.py* que por consequência irá consultar o estado de cada lâmpada na simulação, caso o estado da lâmpada seja *ON* então o script irá chamar a função de alternar o status e a lâmpada apagará mudando o status para *OFF*. Após a execução do script *Python− toggle2 − lightdown.py* uma janela de mensagem é mostrada ao usuário com o status de cada lâmpada da simulação.

```
1  import logging
2  import sys
3  import asyncio
4  from aiocoap import *
5  logging.basicConfig(level=logging.INFO)
6  @asyncio.coroutine
7  def main():
8    for num in range(2,8):
9          protocol = yield from Context.create_client_context()
10         request = Message(code=GET)
11         address = "coap://[cafe::c30c:0:0:" + str(num) + "]/actuators/
               ledstatus"
12         request.set_request_uri(address)
13         try:
14             response = yield from protocol.request(request).response
15         except Exception as e:
16             print('Failed to fetch resource:')
17             print(e)
18         else:
19             print('Result: %s[barra-n]%r'%(response.code,
                   response.payload))
20             variavel = response.payload
21             variavel = str(variavel)
22             print (variavel[2:4])
23             if variavel[2:4] == 'ON':
24               address = "coap://[cafe::c30c:0:0:" + str(num) + "]/
                     actuators/toggle"
25               request = Message(code=POST)
26               request.set_request_uri(address)
27               try:
28                 response = yield from protocol.request(request).
                       response
29               except Exception as e:
30                 print('Failed to fetch resource:')
31                 print(e)
```

```
32                else:
33                  print('Desligando tudo')
34 if __name__ == "__main__":
35    asyncio.get_event_loop().run_until_complete(main())
```

4.1.4 OPÇÃO QUAL ACENDER - APAGAR

Comentaremos sobre a opção *QUAL ACENDER*? por analogia a opção *QUAL APAGAR*? é semelhante, mudando apenas o nome de alguns arquivos. Na opção *QUAL ACENDER*? consideramos a apresentação de 6 botões, estes botões representam as seis lâmpadas que dispomos na simulação. Cada botões, que representa uma lâmpada terá no arquivo *teste*3.*py* uma configuração semelhante a mostrada abaixo:

```
1         self.botao2 = Button(self.fr2, text='2', background='gray',
             font=('Times','12'), command=lampada1)
2         self.botao2.pack(side=RIGHT)
```

Neste primeiro quadro foi definido o botão que irá chamar o método lampada1, mostrado no quadro abaixo:

```
1 def lampada1():
2     pastaDoScript = os.path.dirname(os.path.realpath(__file__))
3     x = subprocess.Popen(['python3', pastaDoScript + '/ligand.py',
          '2'], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
4     retornoScript, erros = x.communicate()
5     if erros:
6         print "Ocorreu algum erro.\nErro:" + erros
7     print retornoScript
8     tkMessageBox.showinfo('----> Ligando <-----', retornoScript)
```

Neste quadro acima, podemos verificar que o script *ligand.py* é chamado com o parâmatro 2. É nesse script que iremos realizar a ação solicitada, que é ligar a lâmpada 2. Abaixo o scritp *ligand.py*, veja que na linha 23 estamos fazendo um teste para verificar se a lâmpada 2 já está ligada. Se ela estiver ligada será mostrada apenas a mensagem da linha 22.

```
1 import logging
2 import sys
3 import asyncio
4 from aiocoap import *
5 logging.basicConfig(level=logging.INFO)
6 @asyncio.coroutine
7 def main():
```

```
8       num = str(sys.argv[1])
9       protocol = yield from Context.create_client_context()
10      request = Message(code=GET)
11      address = "coap://[cafe::c30c:0:0:" + str(num) + "]/actuators/
            ledstatus"
12      request.set_request_uri(address)
13      try:
14          response = yield from protocol.request(request).response
15      except Exception as e:
16          print('Failed to fetch resource:')
17          print(e)
18      else:
19          variavel = response.payload
20          variavel = str(variavel)
21          print('\n')
22          print ('Status lÃ¢mpada ' + str(num) + ' Ã©: ' + variavel[2:5])
23          if variavel[2:5] == 'OFF':
24            address = "coap://[cafe::c30c:0:0:" + str(num) + "]/actuators
                /toggle"
25            request = Message(code=POST)
26            request.set_request_uri(address)
27            try:
28              response = yield from protocol.request(request).response
29            except Exception as e:
30              print('Failed to fetch resource:')
31              print(e)
32            else:
33              print('Ligando lÃ¢mpada ' + str(num))
34  if __name__ == "__main__":
35  asyncio.get_event_loop().run_until_complete(main())
```

4.2 TELAS TKINTER

Desconsidere o terminal mostrado na figura acima, pois está mostrando o *log* de uma excução anterior. Mas a tela principal do aplicativo proposto é a janela mostrada ao centro com os botões verdes.

4.2.0.1 TELA BOTÃO STATUS ATUAL E ACENDER TUDO

O primeiro botão mostrará o status atual de cada lâmpada, conforme o exemplo mostrado na imagem acima. No segundo botão é possível acender todas as lâmpadas.

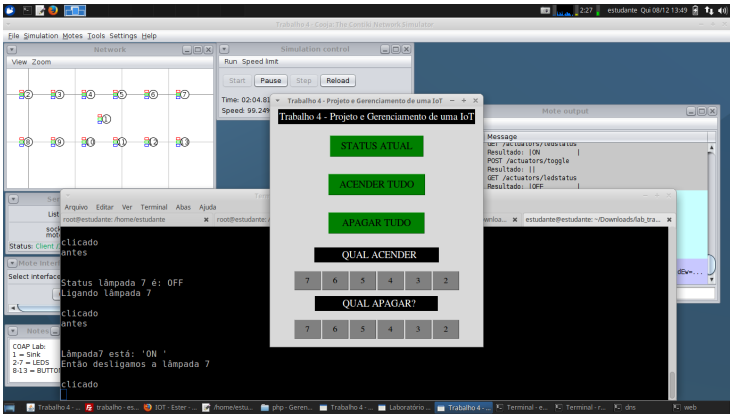

Figura 12: Tela principal

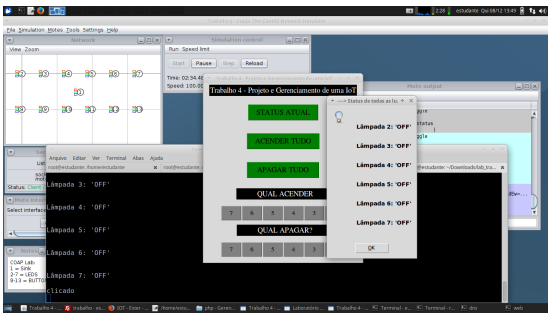

Figura 13: Status atual

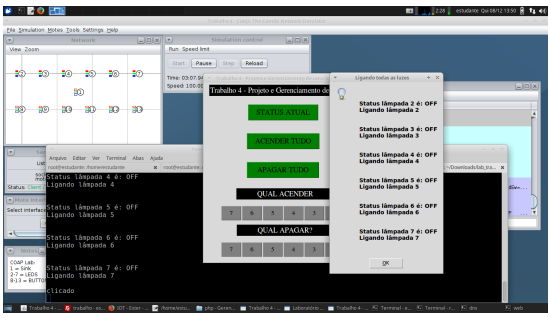

Figura 14: Escolhendo

4.2.0.2 TELA APAGAR TUDO

No terceiro botão é possível apagar todas as lâmpadas.

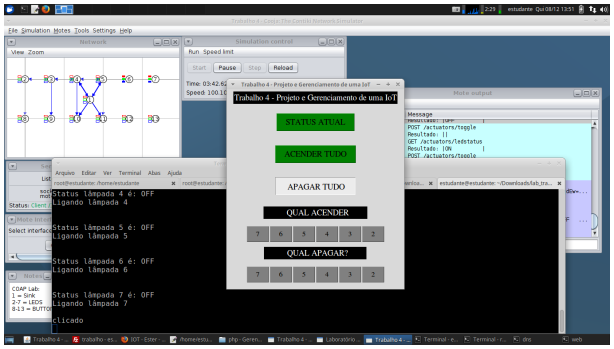

Figura 15: Apagar tudo

4.2.0.3 TELA OPÇÕES LIGAR/APAGAR INDIVIDUALMENTE

E finalmente na quarta opção temos a possiblidade de acender cada uma das lâmpadas individualmente como mostra a figura acima.

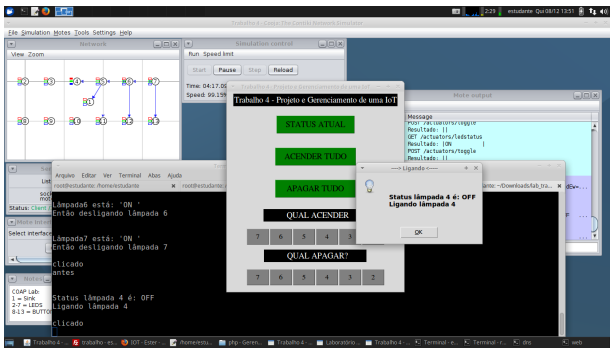

Figura 16: Tela inicial

4.3 IMPLEMENTAÇÃO COM PHP

Para esta implementação utilizamos o *bootstrap* para layout da página fizemos utilização do *lab.py* que foi disponibilizado pelo professor. Esta implementação não atingiu 100% do objetivo, visto que ela nem sempre interage com a simulação. Tentamos integrar a mesma com a implementação realizada acima, contudo o resultado esperado não agradou. Resolvemos manter esta implementação no trabalho visando uma posterior correção e analise para melhoria e também para demostrar nosso empenho em a utilizá-la. Foram criados os scripts php:

```
1 php/acender-tudo.php
2 php/apagar-tudo.php
3 php/desligand.php
4 php/ligand.php
5 php/quais-acesas.php
6 php/python.php
```

Cada um deles segue a ideia que o nome representa, como exemplo veja abaixo o código *php/acender − tudo.php*

```
1 <?php
2   echo '<p>Trabalho de IOT</p>';
3   print('acendendo todas as lÃ¢mpadas: ');
4   exec("/usr/bin/python3 /var/www/html/toggle2-lightup.py");
5 ?>
```

Já o código *php/python.py*, chamaria o arquivo *teste*3.*py*

```
1 <?php
2   exec('/usr/bin/python3 /var/www/html/teste3.py');
3 ?>
```

5 REFERÊNCIAS

Rich Bradshaw. Cross fading images, November de 2016.

Rich Bradshaw. Cross fading images, Nov. 2016.