```
#*-*coding:utf-8*-*
#*-*coding:utf-8*-*
# Estemódulo realiza el monitoreo de los datos del bluetooth
# Esto es software libre, licencia GPL3
# Diego Alberto Parra Garzón
# Bogotá D.C., Colombia
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
#qpy:kivy
from kivy.uix.relativelayout import RelativeLayout
from kivy.uix.label import Label
from kivy.clock import Clock
from kivy.app import App
from kivy.uix.image import Image
from kivy.uix.camera import Camera
from kivy.utils import platform
if platform == "linux":
    print "hola mundo Linux"
    import sys
    sys.path.append('GPSLinux/')
    from Ardutooth import ArduinoBluetooth
if platform == "android":
    print "hola mundo Linux"
    import sys
    sys.path.append('GPSAndroid/')
    from Ardutooth import ArduinoBluetooth


class ScreenLogin(RelativeLayout):
#class ScreenLogin(App):
    def build(self):
        self.opc = False
        self.opc1 = False
        self.cont1 = 0


        self.rl = RelativeLayout()
        self.lb  = Label(font_size="20 sp",pos_hint={'center_x': .5,
'center_y': .1} , markup=True)

#================================================================================
#esta parte es para agregar  la camaraen el layout
#================================================================================
#________________________________________________________________________________
        cam =Camera(resolution=(640, 480), size_hint_x=1.2, size_hint_y=1,
pos_hint={'center_x': .5, 'center_y': .5})
        self.rl.add_widget(cam)
#================================================================================


#================================================================================
#esta parte es para agregar una imagen en el layout
#================================================================================
#________________________________________________________________________________
#         self.img = Image()
#         self.img.source = "Imagenes/Screen1.png"
#         self.img.pos=(0,0)
#         self.rl.add_widget(self.img)
#________________________________________________________________________________
#================================================================================

        self.rl.add_widget(self.lb)
        self.Procesos_Bluetooth()
        self.mensaje1 = "[color=ff3333] Coloque su llave en el lector."
        self.blueRequest = "Nada"
```

```
63          Clock.schedule_interval(self.Cont1, 0.8)
64          Clock.schedule_interval(self.IDENTIFICADOR_P, 0.3)
65          return self.rl
66
67
68      # FUNCIÓN TIPO RELOJ PARA COLOCAR Y QUITAR LOS AVISOS
69      def Cont1(self, dt):
70          print self.cont1
71          Mensaje = "a \r\n"
72          try:
73              ArduinoB.Escribir(Mensaje)
74          except:
75              pass
76
77          try:
78              self.blueRequest =  ArduinoB.LeerCADENA() # LECTURA DEL BLUETOOTH
    ALMACENDA EN UNA VARIABLE
79          except:
80  #            self.blueRequest = " 123142 "
81              pass
82          print "Desde Screen1, la variable es: "+ self.blueRequest + "La longitud
    es: " + str(len(self.blueRequest))
83          el = self.blueRequest.split("\r")
84          self.blueRequest = el[0]
85          if ((self.blueRequest== "Nada")and(len(self.blueRequest)<5)):
86
87              if (self.cont1 <= 2):
88                  print "Llamando el aviso"
89                  self.AvisON()
90                  self.cont1 = self.cont1 + 1
91
92              if ((self.cont1 > 2)or(self.cont1 == 4)):
93                  print "Quitando el aviso"
94                  self.AvisOFF()
95                  self.cont1 = self.cont1 + 1
96
97              if (self.cont1 > 4):
98                  self.cont1 = 0
99                  self.opc = False
100                 self.opc1 = False
101
102
103         if ((len(self.blueRequest) >3) and (self.blueRequest!= "Nada")):
104             print "============> la longitud de la puta variable es: " + str(len
    (self.blueRequest))
105             print "NUID encontrada, Deteneiendo el hilo"
106             Clock.unschedule(self.Cont1)
107             print type(self.blueRequest)
108             self.opc = False
109             dataBlue = []
110             dataBlue = self.blueRequest.split(" ")
111             print dataBlue[0]
112             print len(dataBlue)
113             mensaje = ""
114             for i in range (1, len(dataBlue), 1):
115                 mensaje = mensaje + str(dataBlue[i])
116             print mensaje
117             self.mensaje1 = " [color=ff3333] NUID: " + str(mensaje)
118             self.AvisON()
119             self.blueRequest = str(mensaje)
120             self.bufferLectura()
121
122     # DEFINO UN BUFFER PARA LA LECTURA DEL BLUETOOTH
123     def bufferLectura(self):
```

```
124             lectura_del_Bluetooth = str(self.blueRequest)
125             return lectura_del_Bluetooth
126 
127         # FUNCIÓN PARA ENCENDER LA LECTURA DE LA TARJETA NUEVAMENTE
128         def ReiniciarLectura(self):
129             self.blueRequest = "Nada"
130             self.mensaje1 = "[color=ff3333] Coloque su llave en el lector."
131             try:
132                 Clock.unschedule(self.Cont1)
133             except:
134                 pass
135             try:
136                 Clock.schedule_interval(self.Cont1, 0.8)
137             except:
138                 pass
139 
140 
141 
142         #PONER EL AVISO
143         def AvisON(self):
144             if (self.opc == False):
145                 self.lb.text = str(self.mensaje1)
146                 self.opc = True
147 
148             if (self.opc == True):
149                 pass
150 
151 
152         # QUITAR EL AVISO
153         def AvisOFF(self):
154             if (self.opc1 == False):
155                 self.lb.text = " "
156                 self.opc1 = True
157 
158             if (self.opc1 == True):
159                 pass
160 
161 
162         #IDENTIFICADOR DE PROCESOS DEL BLUETOOTH
163         def Procesos_Bluetooth(self):
164             try:
165                 global ArduinoB
166                 ArduinoB = ArduinoBluetooth()
167                 Mensaje = "Procesos Bluetooth Activados "
168                 print Mensaje
169                 self.EncenderBluetooth()
170             except:
171                 Mensaje = "Fallo al activar los procesos Arduino"
172                 print Mensaje
173             pass
174 
175         #FUNCIÓN PARA ENCENDER EL BLUETOOTH
176         def EncenderBluetooth(self, *args):
177             print "Llamado a prender"
178             Dispo1 = "HC-05" #MODULO BLUETOOTH
179             Dispo2 = "HC-06" #MODULO BLUETOOTH
180             try:
181                 ArduinoB.obtenerCorrienteEnchufe(Dispo1)
182                 Mensaje = "Dispositivo conectado"
183                 print Mensaje
184             except:
185                 Mensaje = "Dispositivo HC-05 NO ENCONTRADO PROBANDO EL HC-06 "
186                 print Mensaje
187                 #            ArduinoB.obtenerCorrienteEnchufe(Dispo1)
```

```
188             #            pass
189             try:
190 
191                 ArduinoB.obtenerCorrienteEnchufe(Dispo2)
192                 Mensaje = "Dispositivo conectado"
193                 print Mensaje
194             except:
195                 Mensaje = "Dispositivo HC-06 NO ENCONTRADO PASANDO A MODO
    AUTONOMO "
196                 print Mensaje
197                 #           ArduinoB.obtenerCorrienteEnchufe("HC-06")
198                 pass
199 
200 
201 
202     def ApagarBluetooth(self):
203         try:
204             ArduinoB.Cerrar()
205         except:
206             print "No se pudo cerrar el bluetooth."
207         try:
208             ArduinoB.__del__()
209         except:
210             pass
211 
212 
213     #DECIDE CUANDO DEJAR O QUITAR LOS PROCESOS DEL SCREEN
214     def IDENTIFICADOR_P(self, dt):
215         #ABRO UN ARCHIVO Y LO ALMACENO EN UNA VARIABLE
216         var = open("Datos/SCREEN.text", "r")
217         self.lec = var.readline()
218         print self.lec
219         var.close()
220 
221 
222         #PERMITE QUE EL SCREEN PERMANEZCA VIVO
223         if self.lec == "True\n":
224             print "Lectura verdadera desde El escreen de verificacion de llaves"
225             pass
226 
227         if(self.lec == "False\n"):
228             print "Lectura Falsa"
229             try:
230                 self.blueRequest = "Nada"
231                 self.bufferLectura()
232             except:
233                 pass
234             try:
235                 #QUITO EL HILO QUE DECIDE CUANDO DEJAR O QUITAR LOS PROCESOS DEL
    SCREEN
236                 Clock.unschedule(self.IDENTIFICADOR_P)
237             except:
238                 pass
239             try:
240                 #QUITO EL HILO DE LA FUNCIÓN TIPO RELOJ PARA COLOCAR Y QUITAR
    LOS AVISOS
241                 Clock.unschedule(self.Cont1)
242             except:
243                 pass
244         #     try:
245          #        ArduinoB.__del__()
246           #   except:
247            #      pass
248 
```

```
249
250
251  class ScreenLogin1(RelativeLayout):
252      def build(self):
253          return  ScreenLogin()
254
255
256  #if __name__ =="__main__":
257   #   ScreenLogin().run()
```