

ANEXO 1

~\Documents\PIP\Rutinas LoRa.py

```
1  #-----
2  #      METODOS DEL LoRa
3  #-----
4  # Los únicos métodos que utiliza el programa principal son:
5
6  # LoRa.comunicacion_establecida()
7  # LoRa.trama(tipo_de_dato, pesos, Trama_GPS)
8  # LoRa.enviar_datos(datos)
9
10 # Todas los demás son llamados internamente por estos
11
12
13 # Toma la herencia de la clase conexion_generica para crear los objetos para el LoRa
14 class conexion_lora(conexion_generica):
15     def __init__(self, P, BR, TO):
16         super().__init__(P, BR, TO)
17
18         self.__trama = ""
19         self.__infoTrama = ""
20
21     def power(self):    #Establece la potencia para la transmisión
22         msj=bytes('radio set pwr 20\r\n','utf-8')
23         print("Enviado: ",msj)
24         self.limpiar()
25         self.enviar(msj)
26         time.sleep(0.1)
27         resp = self.recibir()
28         print("MENSAJE: ", resp)
29         itera = 5
30         resp = ''
31         while resp == '' and itera > 0:
32             resp = self.recibir()
33             itera -=1
34             print(itera)
35             time.sleep(0.5)
36             print(resp)
37             if resp.find('ok') == 0: return True
38         time.sleep(1)
39         return False
40
41
42     def aceptado(self):    #Envía las instrucciones para establecer el join y devuelve
43                             true si es aceptado
44         msj=bytes('mac join otaa\r\n','utf-8')
45         print("Enviado: ",msj)
46         self.limpiar()
47         self.enviar(msj)
48         time.sleep(0.1)
49         resp = self.recibir()
50         print("MENSAJE: ", resp)
51         if resp.find('ok') == 0:
52             print("Recibió ok")
53             itera = 10
54             resp = ''
55             while resp == '' and itera > 0:
56                 resp = self.recibir()
```

```

56         itera -=1
57         print(itera)
58         time.sleep(0.5)
59         print(resp)
60         if resp.find('accepted') == 0: return True
61     time.sleep(1)
62     return False
63
64     def join(self, iteraciones):    #Itera para establecer el join mediante el método
aceptado()
65         self.power()                # Aca debe setear la potencia
66         resp = False
67         while resp == False and iteraciones > 0:
68             resp = self.aceptado()
69             print(resp)
70             iteraciones -=1
71         return resp
72
73     def comunicacion_establecida(self):    #Establece comunicación
74         iteraciones = 10
75         comunicados = False
76         while comunicados == False and iteraciones > 0:
77             msj=bytes('mac get status\r\n','utf-8')
78             print("Enviado: ",msj)
79             self.limpiar()
80             self.enviar(msj)
81             time.sleep(0.1)
82             status=self.recibir()
83             print("Recibió: ", status)
84             print(status[3])
85             print(type(status[3]))
86             if status[3] == "1":
87                 print("Status = 1")
88                 comunicados = True
89             else:
90                 comunicados = self.join(10)
91             iteraciones -=1
92         return comunicados
93
94     def enviar_datos(self, datos):    #Envía los datos previamente preparados por trama()
95         iteraciones = 5
96         while iteraciones > 0:
97             time.sleep(1)
98             msj=bytes('mac set dr 1\r\n','utf-8')
99             print("Enviado: ",msj)
100             self.limpiar()
101             self.enviar(msj)
102             time.sleep(0.1)
103
104             msj=bytes('mac tx cnf 13 '+datos+'\r\n','utf-8')
105             print("Enviado: ",msj)
106             self.limpiar()
107             self.enviar(msj)
108             time.sleep(0.1)
109
110             resp = self.recibir()
111             print("Recibió: ", resp)
112
113             if resp.find('ok') == 0:
114                 print("Recibió ok")

```

```

115         itera = 10
116         resp = ''
117         while resp == '' and itera > 0:
118             resp = self.recibir()
119             itera -= 1
120             print(itera)
121             time.sleep(1)
122             print(resp)
123             if resp.find('mac_tx_ok') == 0: return True
124         iteraciones -= 1
125     return False
126
127
128     def convertir(self, nro, digitos): # convierte el formato de los datos numéricos
129         convertido = str(hex(nro))
130         convertido = convertido.replace('0x', '')
131         convertido = convertido.rjust(digitos, "0")
132         return convertido
133
134     def trama(self, tipo_dato, pesos, trama_GPS): # Prepara la trama de datos que se debe
135 enviar a partir de los datos del GPS y las Balanzas
136         tipo_dato = tipo_dato[0].encode('utf-8')
137         tipo = tipo_dato.hex()
138         sig1 = "2b"
139         nro = str(pesos[0])
140         nro = nro.split(".")
141         nro = int(nro[0])
142         if nro < 0:
143             nro = -nro
144             sig1 = "2d"
145         print(nro)
146         blz1 = self.convertir(nro, 4)
147         sig2 = "2b"
148         if len(pesos) == 2:
149             nro = str(pesos[1])
150             nro = nro.split(".")
151             nro = int(nro[0])
152             if nro < 0:
153                 nro = -nro
154                 sig2 = "2d"
155             else:
156                 nro = 0
157         blz2 = self.convertir(nro, 4)
158
159         trama = trama_GPS.split(",")
160
161         lat_g = self.convertir(int(trama[3][0:2]), 2)
162         lat_m = self.convertir(int(trama[3][2:4]), 2)
163         lat_d = self.convertir(int(trama[3][5:10]), 6)
164         lat_p = trama[4].encode('utf-8')
165         lat_p = lat_p.hex()
166
167         lon_g = self.convertir(int(trama[5][0:3]), 2)
168         lon_m = self.convertir(int(trama[5][3:5]), 2)
169         lon_d = self.convertir(int(trama[5][6:11]), 6)
170         lon_p = trama[6].encode('utf-8')
171         lon_p = lon_p.hex()
172
173         enviar = tipo+sig1+blz1+sig2+blz2+lat_g+lat_m+lat_d+lat_p+lon_g+lon_m+lon_d+lon_p
174         return enviar

```

```

174
175 #-----
176 #   FIN METODOS DEL LoRa
177 #-----
178
179
180 # RUTINAS DENTRO DEL PROGRAMA PRINCIPAL
181
182 # Rutina para establecer comunicación con LoRa por primera vez
183     print("ESTABLECER COMUNICACION:")
184     comunicados = LoRa.comunicacion_establecida()
185     if comunicados:
186         enviados = False
187         datos = "542d07cf2b01f32639000d165344030019f257" # Solo envía una trama genérica
para establecer la comunicación
188         enviados = LoRa.enviar_datos(datos)
189         msj = "Todos los módulos han sido detectados.\n\nComunicación LoRa establecida"
190         btn_iniciar.set_function(2)
191         GLib.idle_add(actualiza_ini, msj, lab_bot)           # Actualiza el texto del botón
"INICIAR"
192     else:
193         msj = "Todos los módulos han sido detectados.\n\nComunicación LoRa NO establecida"
194         lab_bot = lab_iniciar
195         btn_iniciar.set_function(1)
196         GLib.idle_add(actualiza_ini, msj, lab_bot)           # Actualiza el texto del botón
"INICIAR"
197         GPIO.output(led_iniciar, True)
198
199 # Rutina para cuando debe enviar datos
200     contador_LoRa = 20
201     datos = LoRa.trama(tipo_de_dato, pesos, Trama_GPS)
202     comunicados = LoRa.comunicacion_establecida() # Chequea si la comunicación se
encuentra establecida
203     if comunicados:
204         enviados = False
205         enviados = LoRa.enviar_datos(datos)
206         if enviados == False:
207             print("perdida de comunicación")
208     else:
209         msj = "Registrando datos de peso y coordenadas...\n\nSin comunicación con LoRa"
210         GLib.idle_add(actualiza_msj, msj)
211

```