ANEXO 1

~\Documents\PIP\Rutinas LoRa.py

```
#----------------------------
#       METODOS DEL LoRa
#----------------------------
# Los únicos métodos que utiliza el programa principal son:

# LoRa.comunicacion_establecida()
# LoRa.trama(tipo_de_dato, pesos, Trama_GPS)
# LoRa.enviar_datos(datos)

# Todas los demás son llamados internamente por estos


# Toma la herencia de la clase conexion_generica para crear los objetos para el LoRa
class conexion_lora(conexion_generica):
    def __init__(self, P, BR, TO):
        super().__init__(P, BR, TO)

        self.__trama = ""
        self.__infoTrama = ""

    def power(self):    #Establece la potencia para la transmisión
        msj=bytes('radio set pwr 20\r\n','utf-8')
        print("Enviado: ",msj)
        self.limpiar()
        self.enviar(msj)
        time.sleep(0.1)
        resp = self.recibir()
        print("MENSAJE: ", resp)
        itera = 5
        resp = ''
        while resp == '' and itera > 0:
            resp =  self.recibir()
            itera -=1
            print(itera)
            time.sleep(0.5)
            print(resp)
            if resp.find('ok') == 0: return True
        time.sleep(1)
        return False


    def aceptado(self):     #Envía las instrucciones para establecer el join y devuelve true si es aceptado
        msj=bytes('mac join otaa\r\n','utf-8')
        print("Enviado: ",msj)
        self.limpiar()
        self.enviar(msj)
        time.sleep(0.1)
        resp = self.recibir()
        print("MENSAJE: ", resp)
        if resp.find('ok') == 0:
            print("Recibió ok")
            itera = 10
            resp = ''
            while resp == '' and itera > 0:
                resp =  self.recibir()
```

```
                itera -=1
                print(itera)
                time.sleep(0.5)
            print(resp)
            if resp.find('accepted') == 0: return True
        time.sleep(1)
        return False

    def join(self, iteraciones):    #Itera para establecer el join mediante el método aceptado()
        self.power()                            # Aca debe setear la potencia
        resp = False
        while resp == False and iteraciones > 0:
            resp = self.aceptado()
            print(resp)
            iteraciones -=1
        return resp

    def comunicacion_establecida(self):     #Establece comunicación
        iteraciones = 10
        comunicados = False
        while comunicados == False and iteraciones > 0:
            msj=bytes('mac get status\r\n','utf-8')
            print("Enviado: ",msj)
            self.limpiar()
            self.enviar(msj)
            time.sleep(0.1)
            status=self.recibir()
            print("Recibió: ", status)
            print(status[3])
            print(type(status[3]))
            if status[3] == "1":
                print("Status = 1")
                comunicados = True
            else:
                comunicados = self.join(10)
            iteraciones -=1
        return comunicados

    def enviar_datos(self, datos):  #Envía los datos préviamente preparados por trama()
        iteraciones = 5
        while iteraciones > 0:
            time.sleep(1)
            msj=bytes('mac set dr 1\r\n','utf-8')
            print("Enviado: ",msj)
            self.limpiar()
            self.enviar(msj)
            time.sleep(0.1)

            msj=bytes('mac tx cnf 13 '+datos+'\r\n','utf-8')
            print("Enviado: ",msj)
            self.limpiar()
            self.enviar(msj)
            time.sleep(0.1)

            resp = self.recibir()
            print("Recibió: ", resp)

            if resp.find('ok') == 0:
                print("Recibió ok")
```

```
                itera = 10
                resp = ''
                while resp == '' and itera > 0:
                    resp =  self.recibir()
                    itera -=1
                    print(itera)
                    time.sleep(1)
                print(resp)
                if resp.find('mac_tx_ok') == 0: return True
            iteraciones -= 1
        return False


    def convertir(self, nro, digitos): # convierte el formato de los datos numéricos
        convertido = str(hex(nro))
        convertido = convertido.replace('0x','')
        convertido = convertido.rjust(digitos, "0")
        return convertido

    def trama(self,tipo_dato,pesos,trama_GPS):  # Prepara la trama de datos que se debe enviar a partir de los datos del GPS y las Balanzas
        tipo_dato = tipo_dato[0].encode('utf-8')
        tipo = tipo_dato.hex()
        sig1 = "2b"
        nro = str(pesos[0])
        nro = nro.split(".")
        nro = int(nro[0])
        if nro < 0:
            nro = -nro
            sig1 = "2d"
        print(nro)
        blz1 = self.convertir(nro,4)
        sig2 = "2b"
        if len(pesos) == 2:
            nro = str(pesos[1])
            nro = nro.split(".")
            nro = int(nro[0])
            if nro < 0:
                nro = -nro
                sig2 = "2d"
        else:
            nro=0
        blz2 = self.convertir(nro,4)

        trama = trama_GPS.split(",")

        lat_g = self.convertir(int(trama[3][0:2]),2)
        lat_m = self.convertir(int(trama[3][2:4]),2)
        lat_d = self.convertir(int(trama[3][5:10]),6)
        lat_p = trama[4].encode('utf-8')
        lat_p = lat_p.hex()

        lon_g = self.convertir(int(trama[5][0:3]),2)
        lon_m = self.convertir(int(trama[5][3:5]),2)
        lon_d = self.convertir(int(trama[5][6:11]),6)
        lon_p = trama[6].encode('utf-8')
        lon_p = lon_p.hex()

        enviar = tipo+sig1+blz1+sig2+blz2+lat_g+lat_m+lat_d+lat_p+lon_g+lon_m+lon_d+lon_p
        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
```