```
# WSMainSend.py
# Ben Arnett
# 05/12/2025

import time
import board
import busio
import analogio
import adafruit_bme680
import digitalio
from digitalio import DigitalInOut, Direction, Pull
from adafruit_pm25.i2c import PM25_I2C


reset_pin = None

# Setup IO pin to turn on and off PMSA03i
pwr = digitalio.DigitalInOut(board.GP15)
pwr.direction = digitalio.Direction.OUTPUT
pwr.value = True


# Create library object, use 'slow' 100KHz frequency!
i2c = busio.I2C(board.GP9, board.GP8, frequency=100000)

# Connect to a PMSA003i over I2C
pms = PM25_I2C(i2c, reset_pin)
print("Found PMSA003i")

# Connect to BME688 over I2C
# note: BME688 uses the bme680 library
sensor = adafruit_bme680.Adafruit_BME680_I2C(i2c)
sensor.seaLevelhPa = 1011
print("Found BME688")

# Init UART bus for RYLR993 lite
uart = busio.UART(board.GP0, board.GP1, baudrate=9600)
print("UART bus enabled")

# PhotoTransistor setup
pt = analogio.AnalogIn(board.GP26)
print("Photo Transistor Found")

# Address of Rylr module to send to
address = 6

# Function to send LoRa messages with the RYLR
def rylr_send(message):
    length = len(message)
    send = 'AT+SEND={},{},{}\r\n'.format(address, length, message)
    uart.write(send.encode("ascii"))
    print('Data sent to {}'.format(address))

# Function to apply AQI correction calculations
def correct_pm25(pm25, humidity, temperature):
    # correction calculations from ChatGPT
    # Humidity correction (EPA formula)
    humidity_correction = 1 + 0.487 * (2.718 ** (0.059 * humidity))
    pm25_corrected = pm25 / humidity_correction

    # Temperature correction
    temp_correction_factor = 1 - 0.02 * (temperature - 25)
    pm25_corrected *= temp_correction_factor

    return pm25_corrected

# Get Light sensor voltage
def ptvoltage(adcin):
    steps = 0
    i = 0
    # get average over 4 seconds
    while i < 20:
        steps += adcin.value
        time.sleep(0.2)
        i += 1

    steps = steps/20

    # Pico ADC is 12 bits, so need to scale to 16 bits to work with CircuitPython's API
    return (steps * 3.3) / 65536

# Main Loop
while True:
    # Turn on PMSA, give 30s to warmup
    pwr.value = True
    print("Turned on PMSA, 30s warmup")
    time.sleep(30)
    print("Reading Sensors")

    try:
        aqdata = pms.read()
        # print(aqdata)
    except RuntimeError:
        print("Unable to read from sensor, retrying...")
        continue

    # get raw PM 2.5 concentration and apply corrections

    pms_data = pms.read()
    pm25_raw = pms_data["pm25 standard"]
    pm25_corrected = correct_pm25(pm25_raw, humidity, temperature)

    # get BME688 sensor data
    humidity = sensor.relative_humidity
    temperature = sensor.temperature
    pressure = sensor.pressure
    altitude = sensor.altitude

    # get PhotoTransistor Voltage
    ptvolt = ptvoltage(pt)

    # Apply significant figures
    pm25_corrected = round(pm25_corrected)
    humidity = "{:2.1f}".format(humidity)
    temperature = round(temperature) # 3 Digits
    pressure = round(pressure) #
    altitude = round(altitude)
    ptvolt = "{:1.2f}".format(ptvolt) # 3 dig. 4 char

    # format to be parse-able
    rylrmsg =
'AQI:{0};T:{1};RH:{2};hPa:{3};Alt:{4};L:{5}'.format(pm2
5_corrected, temperature, humidity, pressure,
altitude, ptvolt)
    rylr_send(rylrmsg)
    print(rylrmsg)

    # Turn off PMSA
    pwr.value = False
    print("Turned off PMSA, light sleeping")
    # Light sleep till next reading
    time.sleep(866) # Sleep 14m 26s (15 mins
between data transmissions)
                              # ^ account for 30s PMSA on time
& 4s light volt reading
```

```
# ModelComplete.py
# Ben Arnett
# 05/12/2025

# UART = GP 0,1
# I2C = SDA-GP8, SCL-GP9
# Led PWM = GP5
# Servo PWM = GP28

import board
import busio
import digitalio
import time
import pwmio
from lcd1602 import LCD1602
from adafruit_motor import servo

# Initialize Devices
rylr = busio.UART(board.GP0, board.GP1,
baudrate=9600)
i2c0 = busio.I2C(board.GP9, board.GP8,
frequency=100000)

lcd = LCD1602(i2c0)
lcd.clear()
lcd.write('  Initializing')

whtled = pwmio.PWMOut(board.GP3,
frequency=5000, duty_cycle=0)
servo_a_pin = pwmio.PWMOut(board.GP28,
frequency=50)
servo_a = servo.Servo(servo_a_pin,
min_pulse=1000, max_pulse=2000)

# 'Breathe' leds once, modulate servo to verify
function
for i in range(100):
        if i < 50:
            whtled.duty_cycle = int(i * 2 * 65535 / 100)  #
Up
        else:
            whtled.duty_cycle = 65535 - int((i - 50) * 2 *
65535 / 100)  # Down
        time.sleep(0.025)

whtled.duty_cycle = 0

servo_a.angle = 0
time.sleep(0.5)
servo_a.angle = 20
time.sleep(0.5)
servo_a.angle = 40
time.sleep(0.5)
servo_a.angle = 60
time.sleep(0.5)
servo_a.angle = 70
time.sleep(0.5)

lcd.clear()
lcd.set_cursor(2, 0)
lcd.write('Waiting for')
lcd.set_cursor(2, 1)
lcd.write('Weather Data')
print("Init complete, waiting for data")

recieved = False

# Function to parse new recieved data
# Modified for circuitpython from
https://github.com/TimHanewich/MicroPython-Collec
tion/blob/master/REYAX-RYLR998/
def parse_sensor_data(data: str) -> dict:
    # Extract the weather data from entire recieved
byte data
    result = {}
    address:int = None # the address of the
transmitter it came from
    length:int = None # the length (number of bytes)
of the data payload
    msg:bytes = None # the payload data itself
    RSSI:int = None # Received signal strength
indicator
    SNR:int = None # Signal-to-noise ratio

    try:
            # find landmarkers that will help with parsing
            i_equal:int = data.find("=")
            i_comma1:int = data.find(",")
            i_comma2:int = data.find(",", i_comma1 + 1)
            i_comma4:int = data.rfind(",") # search from end
            i_comma3:int = data.rfind(",", 0, i_comma4-1) # search for a comma from right, starting at 0 and ending at the last comma (or right before it)
           # i_linebreak:int = data.find("\r\n")

            # extract
            ReceivedMessage = data
            address = int(data[i_equal + 1:i_comma1])
            length = int(data[i_comma1 + 1:i_comma2])
            msg = data[i_comma2 + 1:i_comma3]
            RSSI = int(data[i_comma3 + 1:i_comma4])
            #SNR = int(data[i_comma4 + 1:i_linebreak])
    except Exception as e:
            raise Exception("Unable to parse line '" + str(data) + "' as a ReceivedMessage! Exception message: " + str(e))

    # Take the weather data string and turn into a dictionary
    # Modified from ChatGPT generated code
    pairs = msg.split(";")  # Split the string into key-value pairs
    for pair in pairs:
        if ":" in pair:
            key, value = pair.split(":")  # Separate key and value
            # Try to convert value to int or float if possible
            if value.replace(".", "", 1).isdigit():
                value = float(value) if "." in value else int(value)
            result[key] = value
    return result # returns a dictionary

# Function to change interior conditions
def interior(d):
    light = d['L']
    # If its pretty bright, open blinds and turn off lights
    if light > 3:
        whtled.duty_cycle = 0
        servo_a.angle = 15
        return

    involt = 3 - light
    lightduty = round(involt * 18000) # inverted scaling
    servangle = round(18 * involt) + 15 # regular scaling

    print('lightduty = {0} servangle = {1}'.format(lightduty, servangle))

    whtled.duty_cycle = lightduty
    servo_a.angle = servangle
    return
```

```
# Function to update stats on display
def lcddisplay(d):
    if recieved == False:
        lcd.clear()
        lcd.write('T:  C Light:   %')
        lcd.set_cursor(0, 1)
        lcd.write('AQI:    RH:   %')

    lightperc = round(d['L'] / 3.2 * 100)

    lcd.set_cursor(2, 0)
    if d['T'] < 10:
        lcd.write(" ")
    lcd.write(str(d['T']))
    lcd.set_cursor(12, 0)
    lcd.write(lcdform(lightperc))
    lcd.set_cursor(4, 1)
    lcd.write(lcdform(d['AQI']))
    lcd.set_cursor(11, 1)
    lcd.write(str(lcdform(round(d['RH']))))
    return
    # T:00C Light:00%
    # _AQI:000 RH:00%

# Format data to write properly
def lcdform(number):
    stringout = ""
    if number < 10:
        stringout = stringout + "  "
    elif number < 100:
        stringout = stringout + " "
    stringout = stringout + str(number)
    return stringout

# Main Loop
while True:
    data = rylr.read()

    if data is not None:
        # Make a string with incoming data, then put it through the parser
        data_string = ''.join([chr(b) for b in data])
        print(data_string, end="")
        parsed = parse_sensor_data(data_string)
        print(parsed)

        # Update display and Interior settings
        lcddisplay(parsed)
        interior(parsed)
        recieved = True

    else:
        if recieved == False:
            time.sleep(0.05)
```

See Github for Test programs