```
import time
import RPi.GPIO as GPIO
import Ports
import pio

def peripheral_setup () :
 pio.terminal=Ports.SerialTerminal (9600)


def main():
    dht11 = 24
    peripheral_setup()
    instance = DHT11(pin=dht11)
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BCM)
    while True:
        temperature, humidity = instance.read()
        pio.terminal.println("Temperature: %-3.1f C" % temperature)
        pio.terminal.println("Humidity: %-3.1f %%" % humidity)
        pio.terminal.println("-----------------------------------------")
        time.sleep(6)

class DHT11:

    def __init__(self, pin):
        self.pin = pin
        self.temperature = None
        self.humidity = None

    def read(self):
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(self.pin, GPIO.OUT)
        # HIGH
        GPIO.output(self.pin, GPIO.HIGH)
        time.sleep(0.05)
        # LOW
        GPIO.output(self.pin, GPIO.LOW)
        time.sleep(0.02)

        # chuyển nó sang input và pull_up
        GPIO.setup(self.pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

        # thu thập dữ liệu
        count = 0
        last = -1
        data = []
        while True:
            current = GPIO.input(self.pin)
            data.append(current)
            if last != current:
                count = 0
                last = current
            else:
                count += 1
                if count > 100:
                    break

        # phân tích độ dài của dữ liệu
        pull_up_lengths = self.parse_data_pull_up_lengths(data)

        if len(pull_up_lengths) != 40:
            return (0, 0)

        # tính toán các bit
        bits = self.calculate_bits(pull_up_lengths)

        # khi có bit, tính toán các bye
```

```
        the_bytes = self.bits_to_bytes(bits)

        # ý nghĩa của các giá trị cảm biến được trả về
        # the_bytes[0]: humidity int
        # the_bytes[1]: humidity decimal
        # the_bytes[2]: temperature int
        # the_bytes[3]: temperature decimal

        self.temperature = the_bytes[2] + float(the_bytes[3]) / 10
        self.humidity = the_bytes[0] + float(the_bytes[1]) / 10

        return self.temperature, self.humidity  # trar về kết quả

    def parse_data_pull_up_lengths(self, data):
        STATE_INIT_PULL_DOWN = 1
        STATE_INIT_PULL_UP = 2
        STATE_DATA_FIRST_PULL_DOWN = 3
        STATE_DATA_PULL_UP = 4
        STATE_DATA_PULL_DOWN = 5

        state = STATE_INIT_PULL_DOWN

        lengths = []  # nó sẽ chứa độ dài dữ liệu trước
        current_length = 0  # nó sẽ chứa độ dài dữ liệu sau

        for i in range(len(data)):

            current = data[i]
            current_length += 1

            if state == STATE_INIT_PULL_DOWN:
                if current == GPIO.LOW:
                    # pull down pin
                    state = STATE_INIT_PULL_UP
                    continue
                else:
                    continue
            if state == STATE_INIT_PULL_UP:
                if current == GPIO.HIGH:
                    # pull upp pin
                    state = STATE_DATA_FIRST_PULL_DOWN
                    continue
                else:
                    continue
            if state == STATE_DATA_FIRST_PULL_DOWN:
                if current == GPIO.LOW:
                    # pull down pin
                    state = STATE_DATA_PULL_UP
                    continue
                else:
                    continue
            if state == STATE_DATA_PULL_UP:
                if current == GPIO.HIGH:
                    # data pulled up, độ dài của pull up sẽ quyết định đâu là 0 hoặc 1
                    current_length = 0
                    state = STATE_DATA_PULL_DOWN
                    continue
                else:
                    continue
            if state == STATE_DATA_PULL_DOWN:
                if current == GPIO.LOW:
                    # pulled down, chúng ta lưu trữ độ dài của pull up trước
                    lengths.append(current_length)
                    state = STATE_DATA_PULL_UP
                    continue
                else:
                    continue
```

```
135
136             return lengths
137
138         def calculate_bits(self, pull_up_lengths):
139             # tìm khoảng thời gian ngắn nhất và dài nhất
140             shortest_pull_up = 1000
141             longest_pull_up = 0
142
143             for i in range(0, len(pull_up_lengths)):
144                 length = pull_up_lengths[i]
145                 if length < shortest_pull_up:
146                     shortest_pull_up = length
147                 if length > longest_pull_up:
148                     longest_pull_up = length
149
150             # sử dụng haflway để xác định xem
151             # khoảng thời gian đó là dài hay ngắn
152             halfway = shortest_pull_up + (longest_pull_up - shortest_pull_up) / 2
153             bits = []
154
155             for i in range(0, len(pull_up_lengths)):
156                 bit = False
157                 if pull_up_lengths[i] > halfway:
158                     bit = True
159                 bits.append(bit)
160
161             return bits
162
163         def bits_to_bytes(self, bits):
164             the_bytes = []
165             byte = 0
166
167             for i in range(0, len(bits)):
168                 byte = byte << 1
169                 if (bits[i]):
170                     byte = byte | 1
171                 else:
172                     byte = byte | 0
173                 if ((i + 1) % 8 == 0):
174                     the_bytes.append(byte)
175                     byte = 0
176
177             return the_bytes
178
```