```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <ESP8266WiFi.h>
#include <ThingSpeak.h>

Adafruit_SSD1306 display(128, 64, &Wire);

// -------- WiFi --------
const char* ssid = "ESP_TEST";
const char* password = "123456789";

// -------- ThingSpeak --------
WiFiClient client;
unsigned long channelID = 3263806;
const char* writeAPIKey = "76P4WTT1W9CR9Y9K";

// -------- Sensor --------
const int sensorPin = A0;
const int ledPin = D8;

// -------- Thresholds --------
int threshold = 580;
int resetThreshold = 550;

// -------- Timing --------
unsigned long lastSampleTime = 0;
unsigned long measureStartTime = 0;
unsigned long lastBeatTime = 0;
unsigned long restartTimer = 0;

const unsigned long sampleInterval = 50;
const unsigned long measureDuration = 20000;   // 20 sec
const unsigned long restartDelay = 60000;      // 1 minute
const unsigned long minBeatInterval = 300;

// -------- Variables --------
int sensorValue;
bool pulseDetected = false;
bool measurementDone = false;

int beatIntervals[5];
int beatIndex = 0;
int bpm = 0;

void setup() {
  Serial.begin(115200);
  pinMode(ledPin, OUTPUT);

  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextColor(WHITE);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) delay(300);

  ThingSpeak.begin(client);

  startMeasurement();
}

void startMeasurement() {
  measureStartTime = millis();
  lastBeatTime = 0;
  beatIndex = 0;
  bpm = 0;
  measurementDone = false;

  display.clearDisplay();
  display.setTextSize(2);
  display.setCursor(0, 20);
  display.println("Measuring");
  display.display();
}

void loop() {
  unsigned long currentMillis = millis();

  // -------- AUTO RESTART AFTER 1 MIN --------
  if (measurementDone && (currentMillis - restartTimer >= restartDelay)) {
    startMeasurement();
  }

  if (measurementDone) return;

  // -------- SENSOR SAMPLING --------
  if (currentMillis - lastSampleTime >= sampleInterval) {
    lastSampleTime = currentMillis;
    sensorValue = analogRead(sensorPin);

    if (sensorValue > threshold && !pulseDetected &&
        (currentMillis - lastBeatTime > minBeatInterval)) {

      pulseDetected = true;
      digitalWrite(ledPin, HIGH);
      if (lastBeatTime > 0 && beatIndex < 5) {
        beatIntervals[beatIndex++] = currentMillis - lastBeatTime;
      }
      lastBeatTime = currentMillis;
    }

    if (sensorValue < resetThreshold) {
      pulseDetected = false;
      digitalWrite(ledPin, LOW);
    }
  }

  // -------- PROGRESS BAR --------
  int progress = map(currentMillis - measureStartTime, 0, measureDuration, 0, 128);
  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0, 0);
  display.println("Measuring...");
  display.drawRect(0, 20, 128, 10, WHITE);
  display.fillRect(0, 20, progress, 10, WHITE);
  display.display();

  // -------- BPM CALCULATION --------
  if (currentMillis - measureStartTime >= measureDuration) {

    long sum = 0;
    for (int i = 0; i < beatIndex; i++) sum += beatIntervals[i];

    if (beatIndex > 0) {
      long avgInterval = sum / beatIndex;
      bpm = 60000 / avgInterval;
    }

    display.clearDisplay();
    display.setTextSize(2);
    display.setCursor(0, 0);
    display.println("Heart Rate");
    display.setCursor(0, 30);
    display.print("BPM: ");
    display.print(bpm);
    display.display();

    ThingSpeak.writeField(channelID, 1, bpm, writeAPIKey);

    Serial.print("Stable BPM: ");
    Serial.println(bpm);
```

```
    measurementDone = true;
    restartTimer = currentMillis;
  }
}
```