# Code Of Esp32 (Micro Controller):

```
#define MODEM_RST            5
#define MODEM_PWKEY          4
#define MODEM_POWER_ON       23
#define MODEM_TX             27
#define MODEM_RX             26
#define I2C_SDA              21
#define I2C_SCL              22
#define temprature           35
#define heartbeat            34
#define ALTITUDE             1655.0
#define IP5306_ADDR          0x75
#define IP5306_REG_SYS_CTL0  0x00
#define BLYNK_HEARTBEAT      5000
#define BLYNK_PRINT Serial
#define TINY_GSM_MODEM_SIM800
#define SerialMon Serial
#define SerialAT Serial1

#include <TinyGsmClient.h>
#include <BlynkSimpleSIM800.h>
#include <SPIFFS.h>
#include <SFE_BMP180.h>
#include <Wire.h>

SFE_BMP180 pressure;


const char apn[]  = "airtelgprs.com";
const char user[] = "";
const char pass[] = "";
const char auth[] = "kTvV_RtR3_RcFJYIcUz4lsRiRBWqEl6W";

TinyGsm modem(SerialAT);

void setup()
{
  SerialMon.begin(115200);
  delay(10);


  pinMode(MODEM_PWKEY, OUTPUT);
  pinMode(MODEM_RST, OUTPUT);
  pinMode(MODEM_POWER_ON, OUTPUT);
  pinMode(LED_BUILTIN,OUTPUT);
  digitalWrite(MODEM_PWKEY, LOW);
```

```
  digitalWrite(MODEM_RST, HIGH);
  digitalWrite(MODEM_POWER_ON, HIGH);

  // Set GSM module baud rate and UART pins
  SerialAT.begin(115200, SERIAL_8N1, MODEM_RX, MODEM_TX);
  delay(3000);

  SerialMon.println("Initializing modem...");
  modem.restart();

  String modemInfo = modem.getModemInfo();
  SerialMon.print("Modem: ");
  SerialMon.println(modemInfo);


  SerialMon.print("Waiting for network...");
  if (!modem.waitForNetwork(240000L)) {
   SerialMon.println(" fail");
   delay(10000);
   return;
  }
  SerialMon.println(" OK");

  if (modem.isNetworkConnected()) {
    SerialMon.println("Network connected");
  }

  SerialMon.print(F("Connecting to APN: "));
  SerialMon.print(apn);
  if (!modem.gprsConnect(apn, user, pass)) {
    SerialMon.println(" fail");
    delay(10000);
    return;
  }
  SerialMon.println("OK");

  if (pressure.begin())
    Serial.println("BMP180 init success");
  else
  {
    Serial.println("BMP180 init fail\n\n");
    while(1);
  }
  Blynk.begin(auth, modem, apn, user, pass);
}

void loop()
{
  int beat;
  int avg;
  int temp;
```

```
beat=analogRead(heartbeat);
avg=(beat*200)/4095;
avg=avg-15;
if(beat){
Serial.println(avg);
Blynk.virtualWrite(V0,avg);
}
else{
Serial.println("Beat not detected");
}
temp=analogRead(temprature);
if(temp){
 Serial.println("Temprature is:");
 temp=(temp-32);
 temp=((temp*5)/9)-5;
 Serial.println(temp);
 Blynk.virtualWrite(V1,temp);
}
else
{
 Serial.println("No Temprature:");
}
char status;
double T,P,p0,a;

Serial.println();
Serial.print("provided altitude: ");
Serial.print(ALTITUDE,0);
Serial.print(" meters, ");
Serial.print(ALTITUDE*3.28084,0);
Serial.println(" feet");

// If you want to measure altitude, and not pressure, you will instead need
// to provide a known baseline pressure. This is shown at the end of the sketch.

// You must first get a temperature measurement to perform a pressure reading.

// Start a temperature measurement:
// If request is successful, the number of ms to wait is returned.
// If request is unsuccessful, 0 is returned.

status = pressure.startTemperature();
if (status != 0)
{
  // Wait for the measurement to complete:
  delay(status);

  // Retrieve the completed temperature measurement:
  // Note that the measurement is stored in the variable T.
  // Function returns 1 if successful, 0 if failure.
```

```
    status = pressure.getTemperature(T);
    if (status != 0)
    {
      // Print out the measurement:
      Serial.print("temperature: ");
      Serial.print(T,2);
      Serial.print(" deg C, ");
      Serial.print((9.0/5.0)*T+32.0,2);
      Serial.println(" deg F");

      // Start a pressure measurement:
      // The parameter is the oversampling setting, from 0 to 3 (highest res, longest wait).
      // If request is successful, the number of ms to wait is returned.
      // If request is unsuccessful, 0 is returned.

      status = pressure.startPressure(3);
      if (status != 0)
      {
        // Wait for the measurement to complete:
        delay(status);

        // Retrieve the completed pressure measurement:
        // Note that the measurement is stored in the variable P.
        // Note also that the function requires the previous temperature measurement (T).
        // (If temperature is stable, you can do one temperature measurement for a number of
pressure measurements.)
        // Function returns 1 if successful, 0 if failure.

      status = pressure.getPressure(P,T);
      if (status != 0)
      {
         // Print out the measurement:
        Serial.print("absolute pressure: ");
        Serial.print(P,2);
        Serial.print(" mb, ");
        Serial.print(P*0.0295333727,2);
        Serial.println(" inHg");

          // The pressure sensor returns absolute pressure, which varies with altitude.
          // To remove the effects of altitude, use the sea level function and your current
altitude.
          // This number is commonly used in weather reports.
          // Parameters: P = absolute pressure in mb, ALTITUDE = current altitude in m.
          // Result: p0 = sea-level compensated pressure in mb

        p0 = pressure.sealevel(P,ALTITUDE); // we're at 1655 meters (Boulder, CO)
        Serial.print("relative (sea-level) pressure: ");
        Serial.print(p0,2);
        Serial.print(" mb, ");
        Serial.print(p0*0.0295333727,2);
        Serial.println(" inHg");
```

```
        // On the other hand, if you want to determine your altitude from the pressure reading,
        // use the altitude function along with a baseline pressure (sea-level or other).
        // Parameters: P = absolute pressure in mb, p0 = baseline pressure in mb.
        // Result: a = altitude in m.

      a = pressure.altitude(P,p0);
      Serial.print("computed altitude: ");
      Serial.print(a,0);
      Serial.print(" meters, ");
      Serial.print(a*3.28084,0);
      Serial.println(" feet");
      }
      else Serial.println("error retrieving pressure measurement\n");
      }
      else Serial.println("error starting pressure measurement\n");
      }
      else Serial.println("error retrieving temperature measurement\n");
      }
      else Serial.println("error starting temperature measurement\n");

      delay(5000);
      Blynk.run();
}
```