Code:

```
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
#include <Wire.h>
#include <SFE_BMP180.h>

SFE_BMP180 bmp180;

void setup() {
  Serial.begin(9600);
   Serial.println("Initializing..");
  dht.begin();

 bool success = bmp180.begin();

  if (success) {
    Serial.println("BMP180 init success");
    Serial.println("DHT11 init success");
  }
}

void loop() {
  Serial.flush();
  delay(2000);

  // Reading temperature or humidity takes about 250
milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a
very slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit
  float f = dht.readTemperature(true);
  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("Failed to read from DHT sensor!");
```

```
    return;
  }
  // Compute heat index
  // Must send in temp in Fahrenheit!
  float hi = dht.computeHeatIndex(f, h);

   char status;
  double T, P;
bool success = false;

  status = bmp180.startTemperature();

  if (status != 0) {
    delay(1000);
    status = bmp180.getTemperature(T);

    if (status != 0) {
      status = bmp180.startPressure(3);

      if (status != 0) {
        delay(status);
        status = bmp180.getPressure(P, T);

        if (status != 0) {
          Serial.print("Pressure: ");
          Serial.print(P);
          Serial.println(" hPa");

        }
      }
    }
  }

  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.println(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C ");
  Serial.print(f);
  Serial.println(" *F\t");
```

```
  Serial.print("Heat index: ");
  Serial.print(hi);
  Serial.println(" *F");
}
```

Explaination:

First we create an object called `bmp180`:

```
SFE_BMP180 bmp180;
```

To initialize the BMP180 sensor and download the calibration coefficients, we need to call the `begin()` method. On success it returns a non-zero value:

```
bool success = bmp180.begin();
```

Following the flow diagram shown earlier, we first use the `startTemperature()` method to start a temperature measurement. On success it also returns a non-zero value:

```
status = bmp180.startTemperature();
```

Then we wait for at least 4.5 milliseconds, and use `getTemperature(T)` to receive the value and store it in the variable `T`:

```
status = bmp180.getTemperature(T);
```

The `startPressure()` method sends the command to start the measurement of pressure. We provide an oversampling value as parameter, which can be between 0 to 3. A value of 3 provides a high resolution, but also a longer delay

between measurements. A value of 0 provides a lower resolution, but is faster. The function returns the number of milliseconds the Arduino needs to wait before reading the pressure value from the sensor:

```
status = bmp180.startPressure(3);
```

Then we use the `getPressure()` method to read the pressure value and store it in the variable `P`:

```
status = bmp180.getPressure(P, T);
```

Notice that we also pass it the variable `T`, since the pressure calculation is dependent on the temperature.