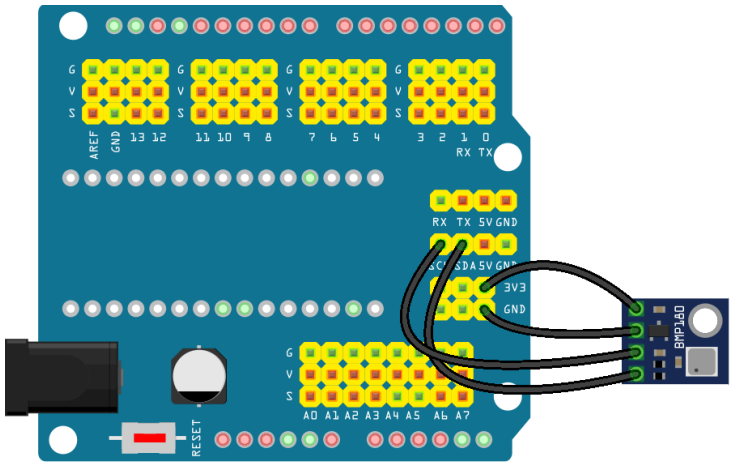
Task.1. Connect the circuit as shown in the picture.



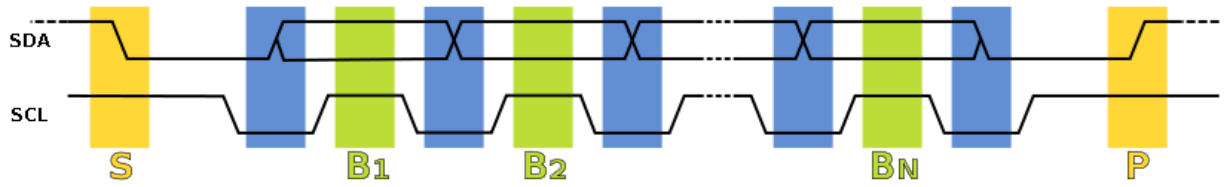
Arduino board	BMP180 board
+3.3V	**VIN**
GND	GND
SCL	SCL
SDA	SDA

<u>Attention: BMP-180 is powered from a 3,3V pin!</u>

BMP-180 is a barometric pressure sensor with an I²C (Two Wire Interface) interface. Technical specifications are as follows:

- **Vin - 3.3V DC.**
- Logic - 3 to 5V compliant.
- Pressure sensing range - 300-1100 hPa (9000m to -500m above sea level).
- Up to 0.03hPa / 0.25m resolution.
- -40 to +85°C operational range.
- +/- 2°C temperature accuracy.

- This board/chip uses I2C 7-bit address 0x77.
- Interface - I2C.



I²C requires just two wires for communication – one for a clock (SCL - **S**erial **CL**ock), and one for data (SDA - **S**erial **DA**ta).

Barometric pressure sensors measure the absolute pressure of the air around them. This pressure varies with both the weather and altitude. Depending on how you interpret the data, you can monitor changes in the weather, measure altitude, or any other tasks that require an accurate pressure reading.

```
#include <SFE_BMP180.h>
#include <Wire.h>
#define BAUDRATE 115200
#define ALTITUDE 37.0 // altitude of Bydgoszcz, in meters

SFE_BMP180 bmp180;
char status;
double temp,pressure,p0,a;

void setup() {
  Serial.begin(BAUDRATE);
  Serial.println("Sensor init");
  if (bmp180.begin())
    Serial.println("BMP180 init success");
   else {
    Serial.println("BMP180 init failure");
    while(1); }
}
```

```
void loop()
{
/*
1. You must first get a temperature measurement to perform a
pressure reading.
2. Start a temperature measurement:
If a request is successful, the number of ms to wait is
returned.
If a request is unsuccessful, 0 is returned.
*/
  status = bmp180.startTemperature();
  if (status != 0) {
   delay(status);

//3. The function returns 1 if successful, or 0 if failure
   status = bmp180.getTemperature(temp);
   if (status != 0)     {
    Serial.print("Temperature: ");
    Serial.print(temp,2);
    Serial.println(" deg C");
/*
4. Start a pressure measurement:
The parameter is the oversampling setting, from 0 to 3 (highest
res, longest wait). If a request is successful, the number of
ms to wait is returned. If a request is unsuccessful, 0 is
returned.
*/
    status = bmp180.startPressure(3);
     if (status != 0) {
      delay(status);
/*
5. Retrieve the completed pressure measurement:
The function requires the previous temperature measurement (T).
(If the temperature is stable, you can do one temperature
measurement for several pressure measurements.)
The function returns 1 if successful, 0 if failure.
*/
      status = bmp180.getPressure(pressure,temp);
      if (status != 0) {
       Serial.print("Absolute pressure: ");
       Serial.print(pressure,2);
       Serial.println(" mb");
```

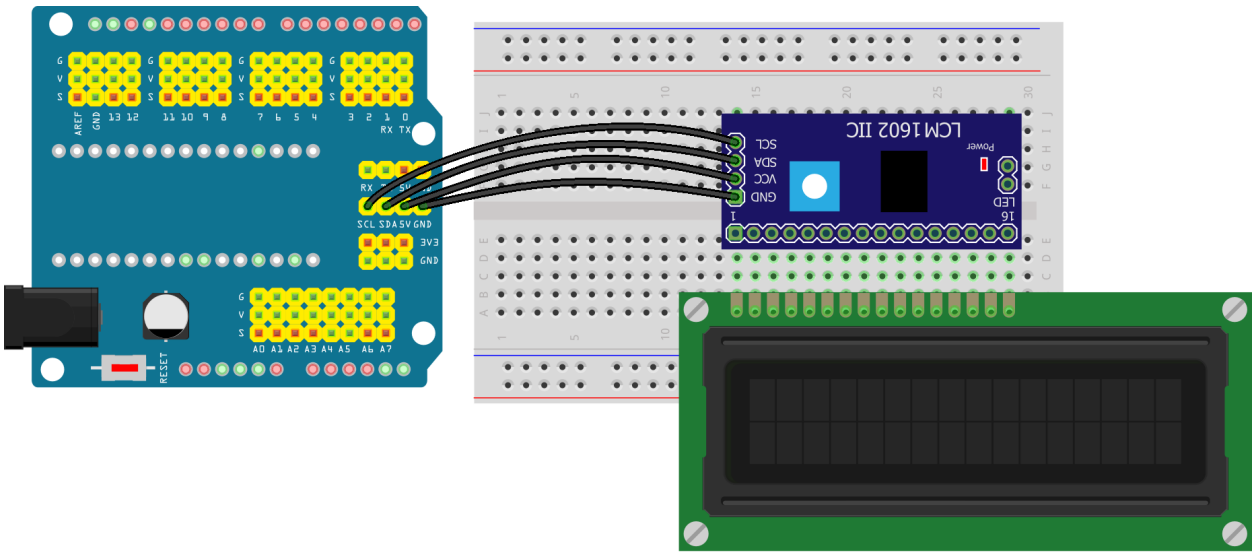
```
/*
6. The pressure sensor returns absolute pressure, which varies
with altitude. To remove the effects of altitude, use the sea
level function and your current altitude.
Parameters: P = absolute pressure in mb, ALTITUDE = current
altitude in m.
Result: p0 = sea-level compensated pressure in mb
*/
       p0 = bmp180.sealevel(pressure,ALTITUDE);
       Serial.print("Sea-level pressure: ");
       Serial.print(p0,2);
       Serial.println(" mb");
//7. If you want to determine your altitude from the pressure
//reading,
       a = bmp180.altitude(pressure,p0);
       Serial.print("Computed altitude: ");
       Serial.print(a,0);
       Serial.println(" meters");
      }
     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);
 }
```

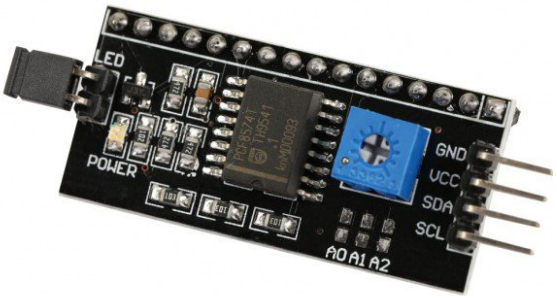
To run the code example You have to download the BMP180 library:

github.com/sparkfun/BMP180_Breakout_Arduino_Library/archive/master.zip

To install the library, open your Arduino IDE, and from the menu, choose *Sketch → Include library → Add .ZIP Library*. A file requester will open.

Task.2. Connect the circuit as shown in the picture:





Arduino board	Converter
GND	GND
+5V	Vcc
SDA	SDA
SCL	SCL

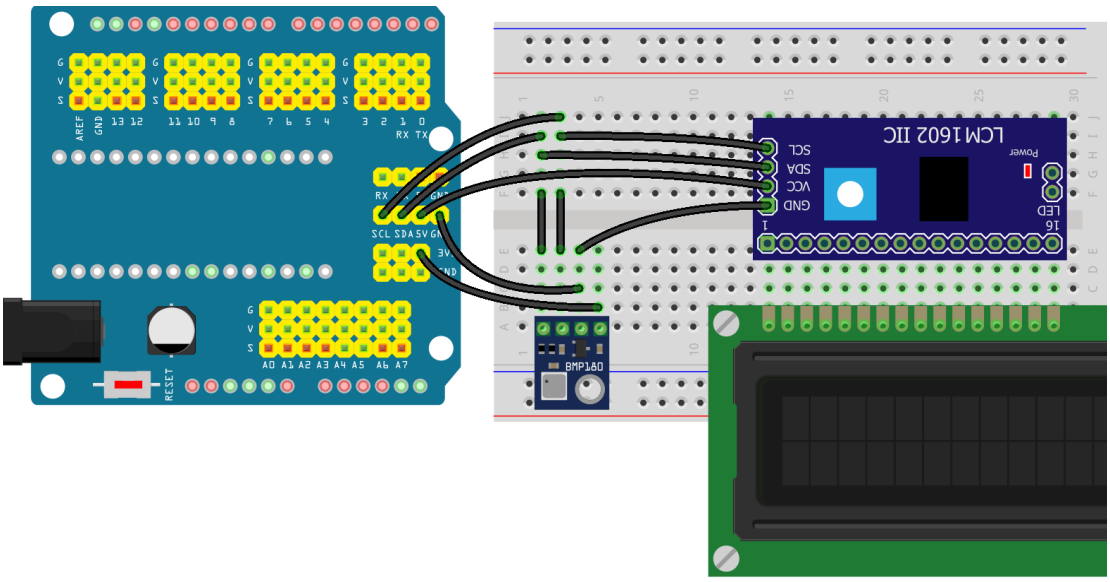
The LM1602 module is based on the PCF8574 chip. It connects an LCD module with HD44780 driver to any programmable unit via an I2C bus. Instead of 7 lines (D4 - D7, E, RS, R/W), we will use only 2. It is also not necessary to connect the potentiometer to adjust the contrast. LM1602 also gives the user control over the display backlight with the appropriate library functions.

```
#include <Wire.h>
#include <LiquidCrystal_PCF8574.h>
#define BAUDRATE 115200
```

```
// set the LCD address to 0x27 for a 16 chars and 2 line
display
LiquidCrystal_PCF8574 lcd(0x27);
int show=0;

void setup() {
  int error;
  Serial.begin(BAUDRATE);
  Serial.println("LCD...");

  Wire.begin();
  Wire.beginTransmission(0x27);
  error = Wire.endTransmission();
  Serial.print("Error: ");
  Serial.println(error);
  if (error == 0)
    Serial.println("LCD found.");
   else {
    Serial.println("LCD not found.");
    while(1);
  }
  // initialize the lcd
  lcd.begin(16, 2);
  show = 0;
}

void loop() {
  if (show == 0) {
    lcd.setBacklight(255);
    lcd.home(); lcd.clear();
    lcd.print("Hello LCD");
    delay(1000);

    lcd.setBacklight(0);
    delay(400);
    lcd.setBacklight(255);

  } else if (show == 1) {
    lcd.clear();
    lcd.print("Cursor On");
    lcd.cursor();
```

```
  } else if (show == 2) {
    lcd.clear();
    lcd.print("Cursor Blink");
    lcd.blink();

  } else if (show == 3) {
    lcd.clear();
    lcd.print("Cursor OFF");
    lcd.noBlink();
    lcd.noCursor();

  } else if (show == 4) {
    lcd.clear();
    lcd.print("Display Off");
    lcd.noDisplay();

  } else if (show == 5) {
    lcd.clear();
    lcd.print("Display On");
    lcd.display();

  } else if (show == 7) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("*** first line.");
    lcd.setCursor(0, 1);
    lcd.print("*** second line.");

  } else if (show == 8) {
    lcd.scrollDisplayLeft();
  } else if (show == 9) {
    lcd.scrollDisplayLeft();
  } else if (show == 10) {
    lcd.scrollDisplayLeft();
  } else if (show == 11) {
    lcd.scrollDisplayRight();
  }

  delay(2000);
  show = (++show)%12;
}
```

Task.3. Create a program for the Arduino board that allows You to display ambient temperature and pressure on the LCD. **Attention: BMP-180 is powered from a 3,3V pin!**



Hint:

```
#include <Adafruit_BMP085.h>
#define seaLevelPressure_hPa 992.0

Adafruit_BMP085 bmp;

void setup() {
 Serial.begin(BAUDRATE);
 if (!bmp.begin()) {
  Serial.println("Could not find a valid sensor.");
  while(1); }
}

#define DELAY 2000

void loop() {
 Serial.print("Temperature = ");
 Serial.print(bmp.readTemperature());
 Serial.println(" *C");
```

```
 Serial.print("Pressure = ");
 Serial.print(bmp.readPressure());
 Serial.println(" Pa");

 Serial.print("Altitude = ");
 Serial.print(bmp.readAltitude());
 Serial.println(" meters");

 Serial.print("Pressure at sealevel (calculated) = ");
 Serial.print(bmp.readSealevelPressure());
 Serial.println(" Pa");

 Serial.print("Real altitude = ");
 Serial.print(bmp.readAltitude(seaLevelPressure_hPa * 100));
 Serial.println(" meters");

 Serial.println();
 delay(DELAY);
}
```

Task.4. Prepare a program for the Arduino board according to the following requirements:

1. After sending 'P' to the Arduino board, it should display the result of the pressure measurement on the LCD.
2. After sending 'T' to the Arduino board, it should display the result of the temperature measurement on the LCD.
3. After sending 'A' to the Arduino board, it should display the result of the altitude calculation on the LCD.

Hint:

```
// Remember to select "No line ending" in Serial Monitor
#define BAUDRATE 115200

char buff;

void setup() {
 Serial.begin(BAUDRATE); }
```

```
void loop() {
while(Serial.available()==0);
if(Serial.available() > 0)
  buff = Serial.read();
switch(buff) {
  case 'P':
   Serial.println("P was pressed");
  break;

  case 'T':
   Serial.println("T was pressed");
  break;

  case 'A':
   Serial.println("A was pressed");
  break;

  default:
   Serial.println("Unrecognized command");
  break;
}
}
```

Task.5. Present real-time temperature and pressure values on the Node-RED dashboard.

For those interested:

1. BMP180 Barometric Pressure Sensor Hookup:

   learn.sparkfun.com/tutorials/bmp180-barometric-pressure-sensor-hookup-

2. Random Nerds tutorial on BMP-180:

   randomnerdtutorials.com/guide-for-bmp180-barometric-sensor-with-arduino

3. Sparkfun I2C tutorial:

   learn.sparkfun.com/tutorials/i2c