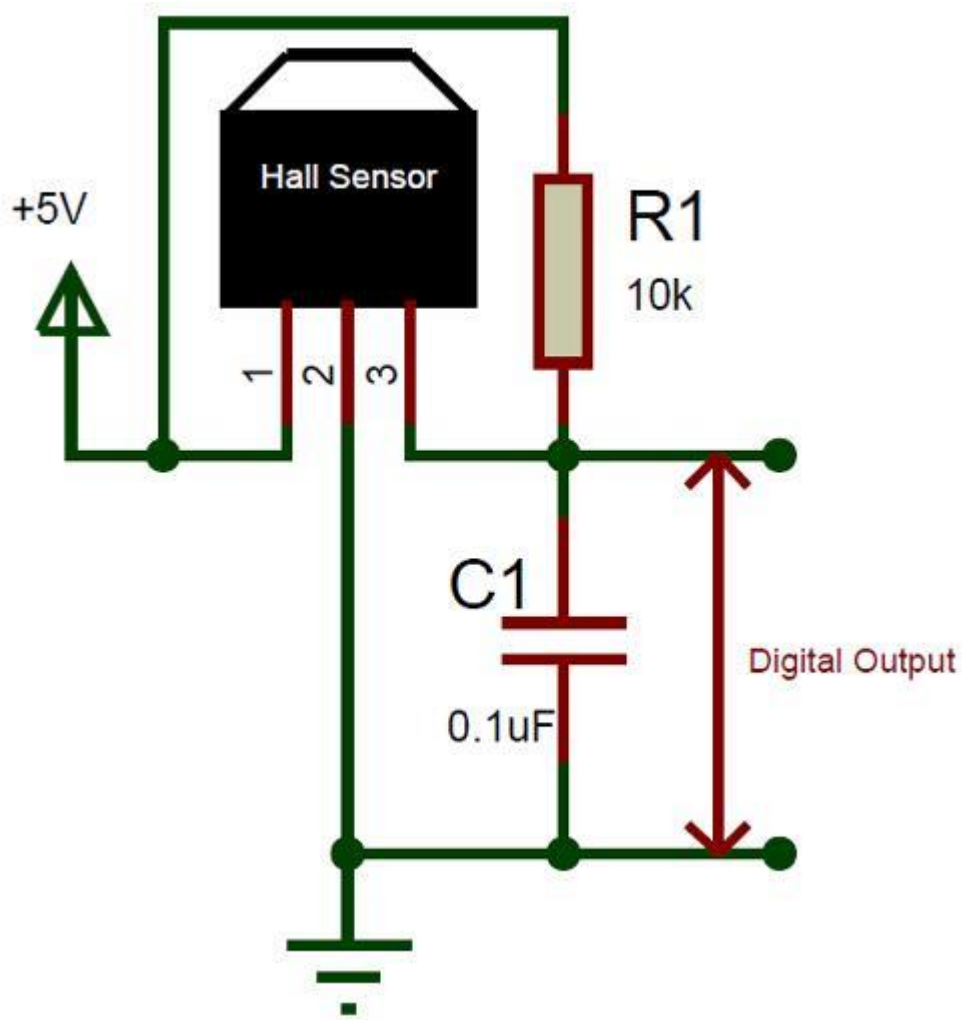
1. A3144 Magnetic Hall Sensor:

Code -

```
const int hallPin = 2 ;     // initializing a pin for the sensor output
10-11-202310-11-2023led. Arduino has built in led attached to pin 13

// variables will change
int hallState = 0 ;          // initializing a variable for storing the status of the hall sensor.
void setup ( ) {

  pinMode ( ledPin , OUTPUT ) ;      // This will initialize the LED pin as an output pin :

  pinMode ( hallPin , INPUT ) ;                          // This will initialize the hall effect sensor pin as an input pin to the Arduino :

  Serial.begin( 9600 ) ;

  Serial.println ("HALL SESNOR WITH ARDUINO") ;

  Serial.println ("Testing the analog hall sensor module:");

}

void loop ( ) {

  hallState = digitalRead ( hallPin ) ;
// reading from the sensor and storing the state of the hall effect sensor :

  if ( hallState == LOW ) {
// Checking whether the state of the module is high or low

    Serial.println ("The state of the analog hall module is high");

    digitalWrite ( ledPin , HIGH ) ;
// turn on the LED if he state of the module is high

  }
```

```
  else {

    digitalWrite ( ledPin , LOW ) ;
// otherwise turn off the LED :

    Serial.println ("The state of the analog hall module is
low ") ;
  }
```

Circuit -



2. DS18B20 Temp sensor :

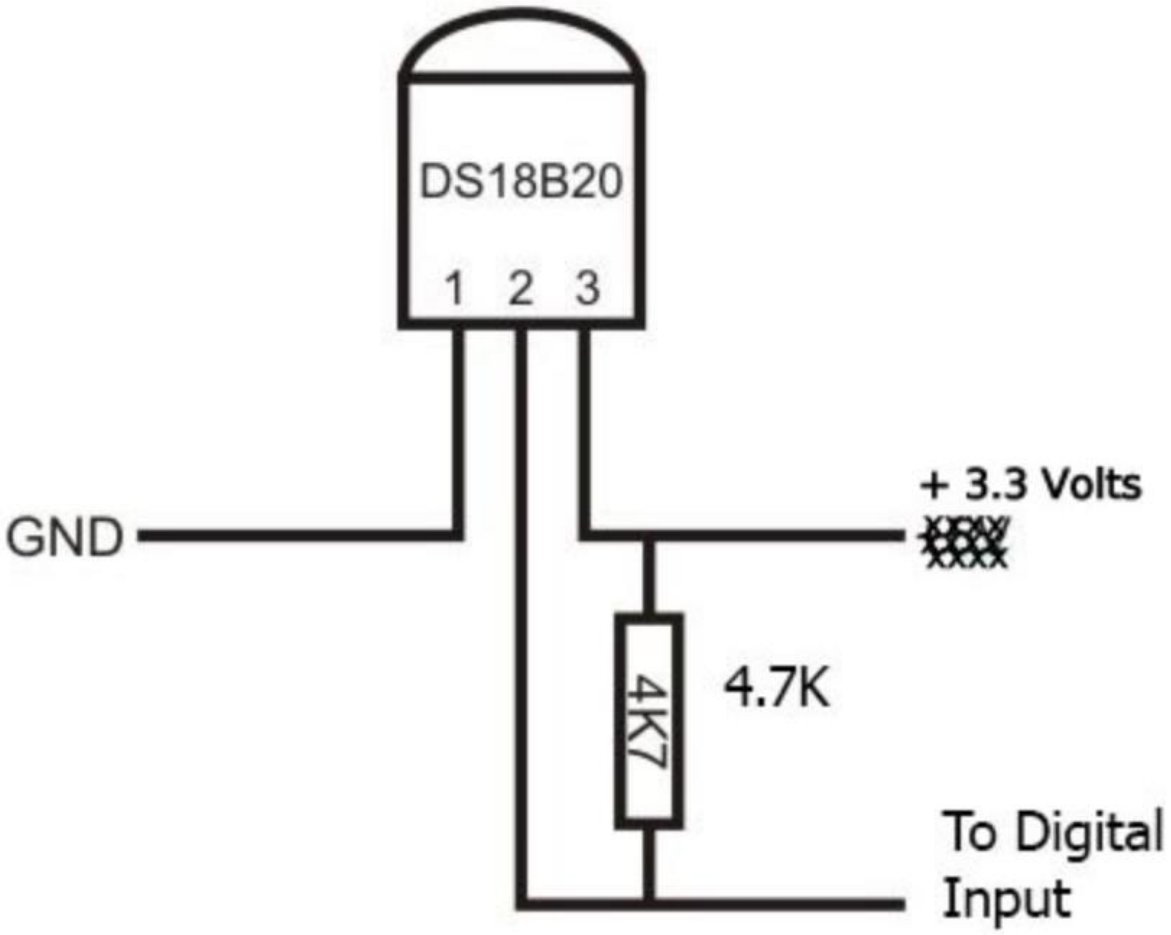
Code -

```
// Include the required Arduino libraries:

#include

#include



// Define to which pin of the Arduino the 1-Wire bus is
connected:

#define ONE_WIRE_BUS 2



// Create a new instance of the oneWire class to
communicate with any OneWire device:

OneWire oneWire(ONE_WIRE_BUS);



// Pass the oneWire reference to DallasTemperature
library:

DallasTemperature sensors(&oneWire);



void setup() {

  // Begin serial communication at a baud rate of 9600:

  Serial.begin(9600);

  // Start up the library:
```

```
  sensors.begin();

}



void loop() {

  // Send the command for all devices on the bus to
perform a temperature conversion:

  sensors.requestTemperatures();



  // Fetch the temperature in degrees Celsius for device
index:

  float tempC = sensors.getTempCByIndex(0); // the index
0 refers to the first device

  // Fetch the temperature in degrees Fahrenheit for
device index:

  float tempF = sensors.getTempFByIndex(0);



  // Print the temperature in Celsius in the Serial
Monitor:

  Serial.print("Temperature: ");

  Serial.print(tempC);

  Serial.print(" \xC2\xB0"); // shows degree symbol

  Serial.print("C  |  ");



  // Print the temperature in Fahrenheit
```

```
  Serial.print(tempF);
  Serial.print(" \xC2\xB0"); // shows degree symbol
  Serial.println("F");


  // Wait 1 second:
  delay(1000);
}
```

Circuit –



3. DHT22 Temp sensor :

Code -

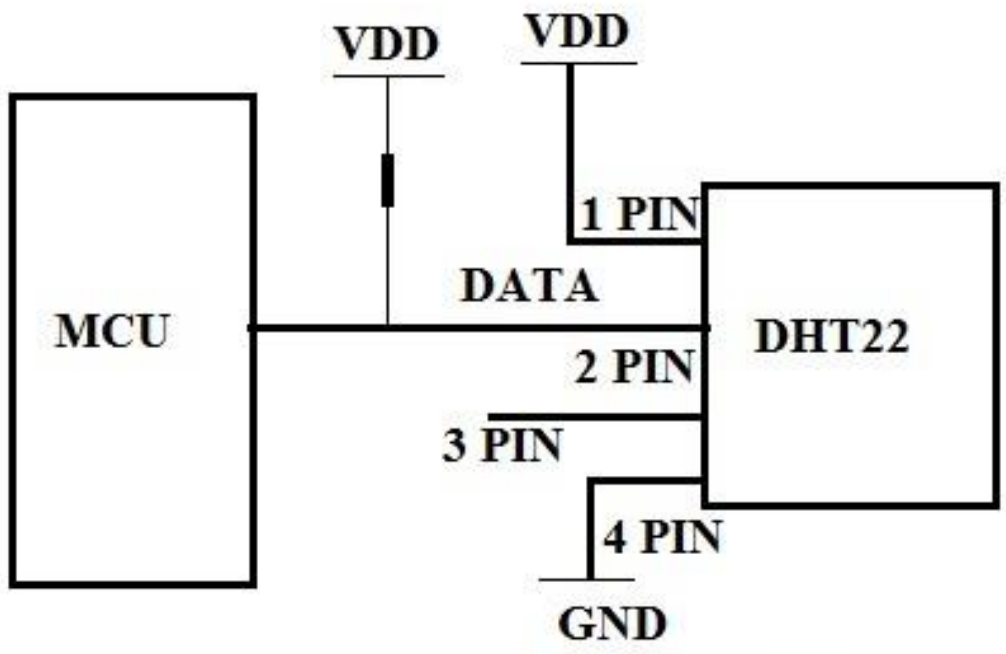
```
/* Arduino example code for DHT11, DHT22/AM2302 and DHT21/AM2301 temperature and humidity sensors. More info: www.www.makerguides.com */


// Include the libraries:

#include

#include


// Set DHT pin:

#define DHTPIN 2


// Set DHT type, uncomment whatever type you're using!

#define DHTTYPE DHT11   // DHT 11

//#define DHTTYPE DHT22   // DHT 22  (AM2302)

//#define DHTTYPE DHT21   // DHT 21 (AM2301)


// Initialize DHT sensor for normal 16mhz Arduino:

DHT dht = DHT(DHTPIN, DHTTYPE);


void setup() {
```

```
  // Begin serial communication at a baud rate of 9600:
  Serial.begin(9600);


  // Setup sensor:
  dht.begin();
}


void loop() {
  // Wait a few seconds between measurements:
  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)


  // Read the humidity in %:
  float h = dht.readHumidity();
  // Read the temperature as Celsius:
  float t = dht.readTemperature();
  // Read the 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 in Fahrenheit (default):
  float hif = dht.computeHeatIndex(f, h);
  // Compute heat index in Celsius:
  float hic = dht.computeHeatIndex(t, h, false);



  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" % ");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" \xC2\xB0");
  Serial.print("C | ");
  Serial.print(f);
```

```
  Serial.print(" \xC2\xB0");

  Serial.print("F ");

  Serial.print("Heat index: ");

  Serial.print(hic);

  Serial.print(" \xC2\xB0");

  Serial.print("C | ");

  Serial.print(hif);

  Serial.print(" \xC2\xB0");

  Serial.println("F");

}
```

Circuit –



10.GY-30 BHI750FVI :

Code –

The library comes with five examples. The best one to get started on is the BH1750Test.

```
#include <Wire.h>
#include <BH1750.h>

BH1750 lightMeter;

void setup(){

  Serial.begin(9600);

  Wire.begin();

  lightMeter.configure(BH1750::ONE_TIME_HIGH_RES_MODE);
  lightMeter.begin();

  Serial.println(F("BH1750 Test begin"));

}
```

Reading the lux values is fairly easy. The first step is to create an object from the BH1750 class which is usable after including the library:

```
#include <BH1750.h>


BH1750 lightMeter;
```

Then, invoke *begin()* function then the *readLightLevel()* function which returns a float value:

```
lightMeter.begin();


float lux = lightMeter.readLightLevel()
```

In the *BH1750Test* sketch above, the lux levels are printed in the Serial Monitor.

By default, the library uses continuous high-resolution mode. To change this, pass the parameter either to begin() or to another function configure(). The possible parameters are:

```
BH1750::CONTINUOUS_LOW_RES_MODE

BH1750::CONTINUOUS_HIGH_RES_MODE

BH1750::CONTINUOUS_HIGH_RES_MODE_2

BH1750::ONE_TIME_LOW_RES_MODE

BH1750::ONE_TIME_HIGH_RES_MODE

BH1750::ONE_TIME_HIGH_RES_MODE_2
```

For example, if you want to use one shot, high-resolution mode, then:

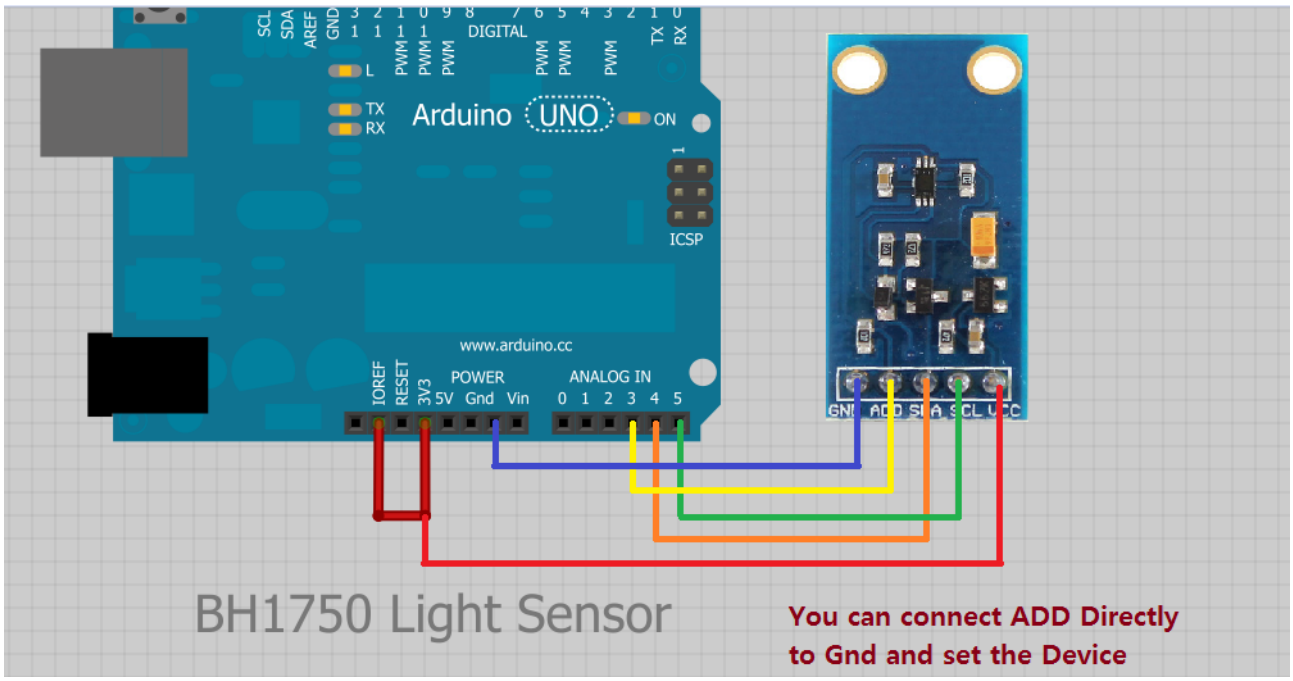
```
lightMeter.begin(BH1750::ONE_TIME_HIGH_RES_MODE)
```

Or

```
lightMeter.configure(BH1750::ONE_TIME_HIGH_RES_MODE)
```

Normally, the sensor reverts to the default mode after data is sent using one-shot mode. However, the library automatically uses the same mode even if the previous mode is one shot.

Circuit –



11. GY-BME280 Precision Altimeter Atmospheric Pressure Sensor :

Code –

After installing the BME280 library, and the Adafruit_Sensor library, open the Arduino IDE and, go to **File** > **Examples** > **Adafruit BME280 library** > **bme280 test.**

```
/*
   BME280 Weather Station application

   Uses LCD2004 with I2C interface for display.
   Connect I2C interface of both LCD display and BME280
        SCL connects to A5 or dedicated SCL pin
        SDA connects to A4 or dedicated SDA pin
   Connect LCD Vcc to 5V and GND to ground
   Connect BME280 Vcc to 3.3V and GND to ground
   Need to install library LiquidCrystal_I2C
   Need to install library Adafruit_BME280
   Need to manually install library Adafruit_Sensor
*/
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

float temperature;
float humidity;
float pressure;
```

```
float altitude;


float const ALTITUDE = 62.0;                // Altitude at
my location in meters
float const SEA_LEVEL_PRESSURE = 1013.25;  // Pressure at
 sea level


Adafruit_BME280 bme; // I2C


LiquidCrystal_I2C lcd(0x27, 20, 4);   // I2C address, 20
char x 4 lines
//=========================================================
=======================
//  Initialization
//=========================================================
=======================
void setup(void) {


  lcd.begin();
  lcd.clear();           // Clear display
  lcd.backlight();       // Make sure backlight is on
  lcd.print("Reading sensor");


  bool status;


  // default settings
  status = bme.begin(0x76);  // The I2C address of the se
nsor is 0x76
  if (!status) {             // Loop if sensor not found
    lcd.clear();
    lcd.print("Error. Check");
    lcd.setCursor(0, 1);
    lcd.print("connections");
```

```
    while (1);
  }
  // Print non-changing info on LCD once
  lcd.clear();          // Clear display
  lcd.setCursor(0, 0);  //Set cursor to character 0 on li
ne 0
  lcd.print("Temperature: ");
  lcd.setCursor(0, 1);  //Set cursor to line 1
  lcd.print("Humidity: ");
  lcd.setCursor(0, 2); // Set cursor to line 2
  lcd.print("Pressure: ");
  lcd.setCursor(0, 3); // Set cursor to line 3
  lcd.print("Altitude:   ");
}
//=========================================================
=======================
//  Main
//=========================================================
=======================
void loop() {

  getPressure();   // Get sensor data and print to LCD
  getHumidity();
  getTemperature();
  getAltitude();

  delay(2000);     // Update readings every 2 seconds
}
//=========================================================
=======================
//  getTemperature - Subroutine to get and print temperat
ure
```

```
//=========================================================
========================
void getTemperature()
{
  temperature = bme.readTemperature();
  temperature = temperature * 9 / 5 + 32; // Convert C to
 F
  String temperatureString = String(temperature, 1); // O
ne decimal position
  lcd.setCursor(13, 0);            // Move to start of rea
ding
  lcd.print("       ");           // Clear old reading
  lcd.setCursor(13, 0);            // Reset cursor locatio
n
  lcd.print(temperatureString);    // Write new reading
  lcd.print((char)223);            // Degree symbol
  lcd.print("F ");
}
//=========================================================
========================
//  getHumidity - Subroutine to get and print humidity
//=========================================================
========================
void getHumidity()
{
  humidity = bme.readHumidity();
  String humidityString = String(humidity, 0);
  lcd.setCursor(13, 1);
  lcd.print("       ");
  lcd.setCursor(13, 1);
  lcd.print(humidityString);
  lcd.print("%");
}
```

```
//=========================================================
========================
//  getPressure - Subroutine to get and print pressure
//=========================================================
========================
void getPressure()
{
  pressure = bme.readPressure();
  pressure = bme.seaLevelForAltitude(ALTITUDE, pressure);
  pressure = pressure / 3386.39;    // Convert hPa to in/
Hg
  lcd.setCursor(13, 2);
  lcd.print("       ");
  lcd.setCursor(13, 2);
  String pressureString = String(pressure, 2);
  lcd.print(pressureString);
  lcd.print("""");
}
//=========================================================
========================
//  getAltitude - Subroutine to get and print temperature
//=========================================================
========================
void getAltitude()
{
  altitude = bme.readAltitude(SEA_LEVEL_PRESSURE);
  altitude = altitude * 3.28084;  // Convert meters to fe
et
  lcd.setCursor(13, 3);
  lcd.print("      ");
  lcd.setCursor(13, 3);
  String altitudeString = String(altitude, 0);
  lcd.print(altitudeString);
```

```
  lcd.print(" ft");
}
```

Circuit –