Designed in UAE

المدرسة الإماراتية

28 COMPONENTS

Designed with passion and dedication in the United Arab Emirates

ibtibot Discovery Kit

Arduino Compatible

GUIDE

+13

SUITABLE AGE

ibtikar®
EDU TECH SOLUTIONS

While the advice and information in this book is believed to be true and accurate at the date of publication, neither the author nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Ibtikar Edu Tech Solutions, and its dealers and distributors will not be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Ibtikar Edu Tech Solutions has endeavored to provide trademark information of the companies and products mentioned in this book by the appropriate use of capitals. However, Ibtikar Edu Tech Solutions cannot guarantee the accuracy of this information.

ibtibot

Discovery Kit

Arduino Compatible

GUIDE

Contents

Introduction

Welcome to the World of Electronics. You will navigate this world of embedded systems, signals and modules using an incredible kit called the **Ibtikar Discovery Kit**. Designed for the beginner, this guide will take you on a journey of discovery stimulating curiosity through engaging activities and real-life applications of embedded systems.

Embedded systems control many devices in common use today. An **embedded system** is a specialised computer system with a specific function within a larger mechanical or electrical system. It is embedded as part of a complete device, often including hardware and mechanical parts.

A **controller** is an electronic unit that works as a computer, to manage the control process of a certain machine or device. A controller can be programmed to read the **input**, then based on its program, it will then control the **output**. A controller has three essential parts: memory, input peripherals and output peripherals. When these parts are contained together on a chip, you have a microcontroller.

The **microcontroller** is the main control unit of any automatic system or device. These systems and devices have the following units: an **input unit** for collecting signals or sensing the signals from an environment, a **control unit** for processing the received signals and an **output unit** for sending out signals or to control an output device.

One of the most known microcontrollers, is the **Arduino** board. The Arduino microcontroller is an easy-to-use board with many ports for input and output. The Arduino microcontroller can be programmed with a set of instructions (commands) to do a certain job. When you want to operate a machine to do a certain flow of tasks, you need to use different commands in a certain order. This process is called **programming**.

Arduino IDE (Integrated Development Environment) allows you to write a program and upload the program to an Arduino microcontroller board. Arduino IDE language is based on a programming language called C language. Arduino IDE calls the programs *sketches*, but to make it simple we will use *program* or *code*, rather than *sketch*.

The Discovery Kit

Now, you are going to use, the **Ibtikar Discovery kit**, also called the Input/Output Module Kit.

The kit contains the following components:

MODULES

- Ambient Light Sensor
- Potentiometer
- Vibration Sensor
- Sound Sensor
- Flame Sensor
- Gas Sensor
- Buzzer Module
- Digital PIR (Motion) Sensor
- Digital Push Button Module
- Capacitive Touch Sensor
- Digital 5A Relay Module
- Digital Vibration Sensor
- Rotary Encoder Module
- Digital IR Receiver Module
- Digital IR Transmitter Module
- LED Module
- Shift-out Module
- 7-Segment Module
- Temperature and Humidity Sensor
- Dual H-Bridge & Stepper Motor Driver

CABLES & SUPPORT

- 9V Battery Clip with Power Jack
- 9V Battery Clip with Cables
- Jumper Cables (F/F)
- Jumper Cables (M/F)
- 3-Pin Analog Cable x 4
- 3-Pin Digital Cable x 4
- 5-Pin Cable x 2
- Micro USB Cable
- Remote Control
- Screwdriver

MOTORS

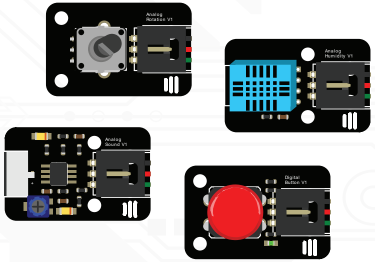
- DC Motor x 2
- Stepper Motor
- Servo Motor

MICRO & SHIELDS

- Ibtikar Arduino Leonardo
- Arduino IO Expansion Shield
- Arduino LCD Shield

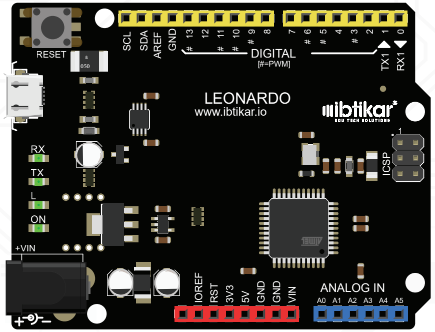
The general concept of this kit is to use the motors and the plug-and-play modules which can be input or output modules with the controller, to do certain tasks. Shields make the interface between these parts easier, while the cables do the actual connection.

Input



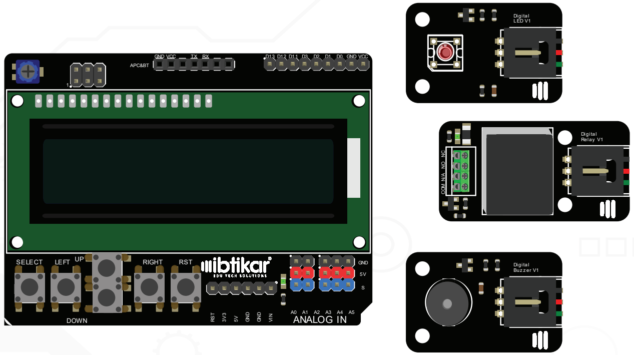
An **input** or a **sensor** is a unit that senses the characteristics of its environment and then sends this data to another device.

Controller



A **microcontroller** is the control unit. Think of it as the brain of your device.

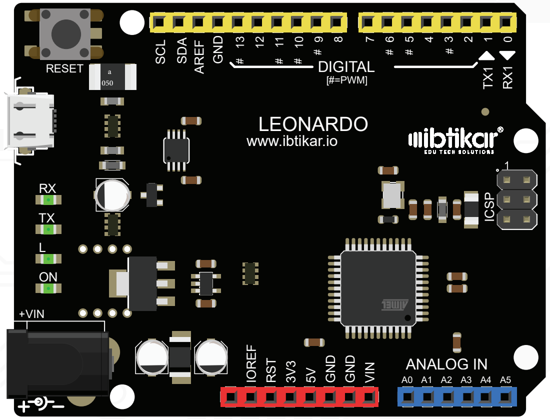
Output



An **output** or an **actuator** is a device that generates an action or moves a system or a mechanism.

You will start with simple activities, then move to more complex tasks and activities using different parts, ideas, maths, and commands. Let’s start with the brain of this kit - i.e. the Leonardo microcontroller.

Arduino Leonardo



The Arduino board in the Discovery kit is called the **Leonardo** board. It has an Atmel 32U4 microcontroller that reads the instructions of the program you write and follows them. You can think of it as the brain of the Arduino board.

The board has a USB connecter. This is where the USB cable connects the Arduino board to the computer. It allows you to access the Arduino board and upload your programs. It can be used to power up the Arduino board, without the need to use a separate power supply.

The board can also be powered through the power jack. The power jack has a voltage limitation, so you need to be careful when you connect an external power supply to the board. The recommended input voltage (Vin) to the Arduino is 7-12 volts. The limit of the input voltage is within 6-20 volts. If outside these limits, the Arduino will be operated at a risk and this could lead to a breakdown of the microcontroller!

A reset button and a set of light indicators are also on the board. The reset button resets the Arduino board which then resets the program you are running. The set of light indicators has 4 LEDs where each one has its own purpose:

- **RX-TX LEDs**: blink when the Arduino board receives or transmits data.
- **L LED**: used for test purposes and can be controlled by the user.
- **ON LED**: turns ON when the board is powered up.

Three types of sockets with different colours are on the board. The first type is the power socket coloured red. This socket contains 5 power pins and allows you to have access to different power options:

- **3.3V pin** which represents a node with a 3.3 voltage supply.
- **5V pin** which represents a node with a higher voltage supply, 5 volts.
- **2 GND pins** which are grounded, and their voltage is 0 volts.
- **Vin pin** which is similar to the power jack. It gives an option to power the Arduino board, using an external DC source.

The second types of sockets are the yellow ones. These are the digital sockets which allow you to access the Arduino digital pins. Arduino can read and write digital signals. A **digital signal** has a finite set of possible values, like a switch being either ON or OFF.

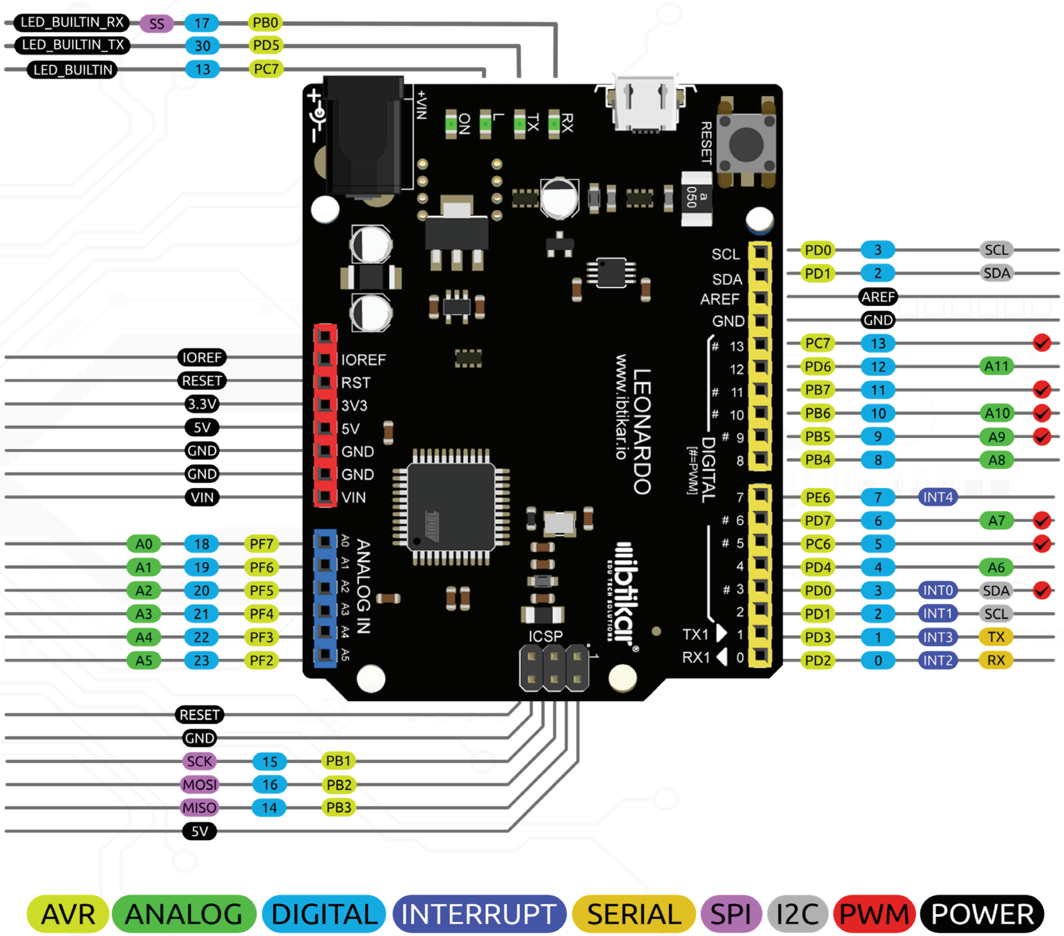
The third type is the blue socket. This is the analog input socket which is used to read analog signals from different sensors. An **analog signal** is a continuous signal with an infinite number of possibilities, like the infinite number of possible colour combinations.

The pinout of the board showing all pins and their purposes is shown.

Before you start programming your Arduino for the first time, you must set it up. You need the Arduino board, a USB cable, and a computer.

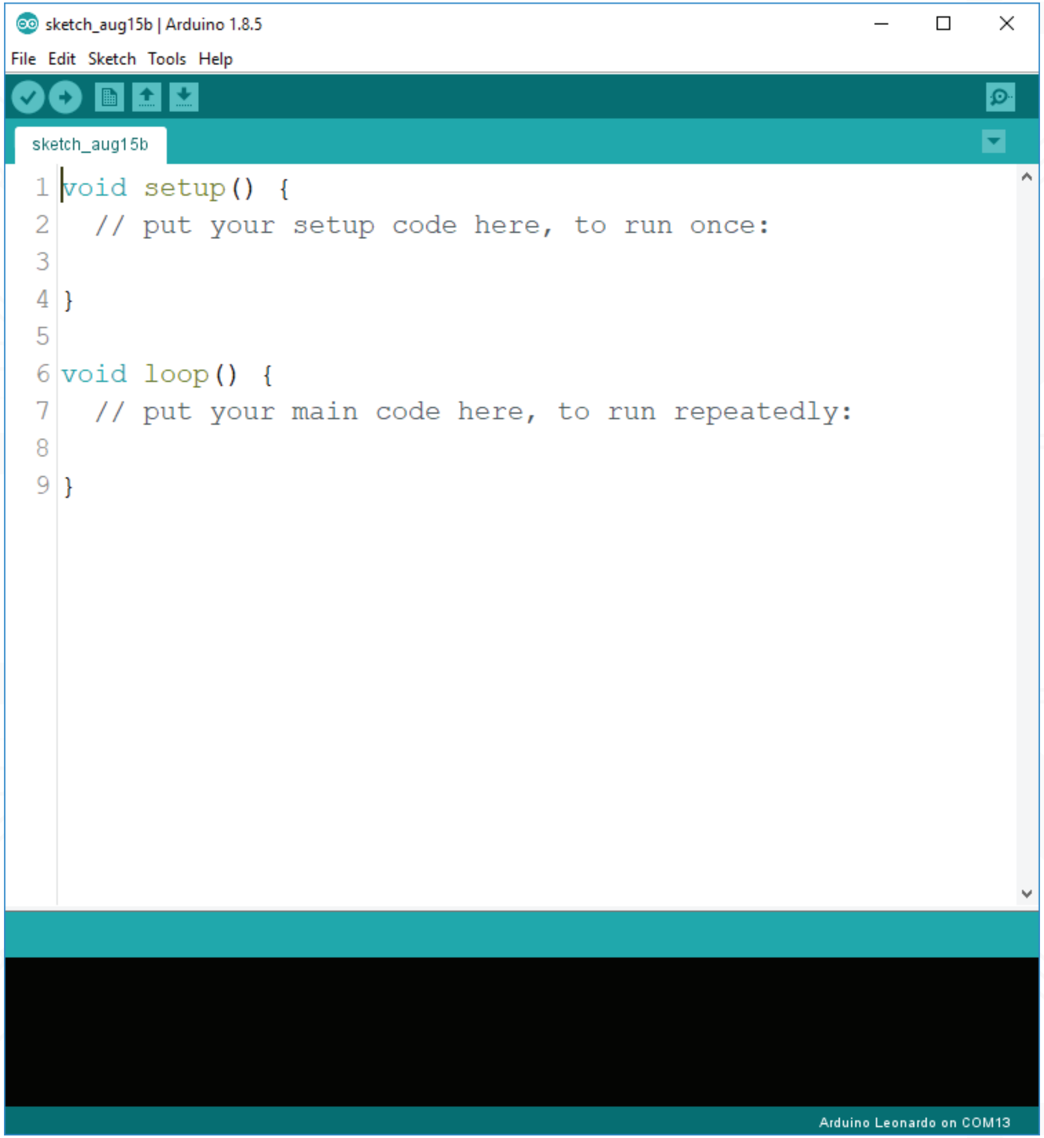
First, install the Arduino IDE from the Arduino official website. Choose the one that suits your machine and operating system. This software allows you to connect with the Arduino board, write a code using the Arduino language, verify it, and then upload it to the Arduino board.

Connect the Arduino board to the computer using the USB cable. Insert one end of the USB cable into the USB connecter and the other end into a USB socket on your computer. The power LED will turn ON.



A message might pop up saying that the “*Device driver software was not successfully installed*”. This means that the Arduino board and your computer cannot see each other. For your computer to recognise the Arduino board, it needs a proper introduction which is known as *Driver*. More details can be found in *Appendix: Arduino Driver Installation*.

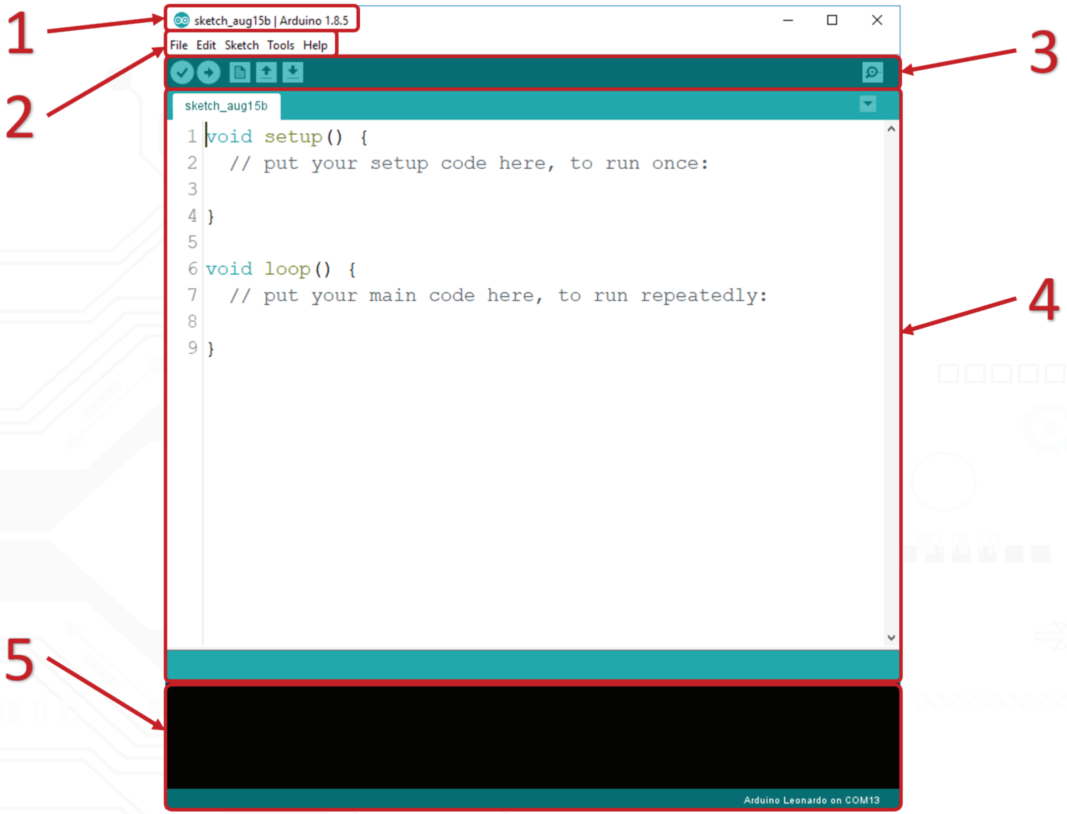
Launch the Arduino IDE by double-clicking on the Arduino icon on the desktop and wait until the software starts. You should see the following window:



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Arduino Leonardo on COM13

The IDE is split into five parts as shown:


1
2
3
4
5
sketch_aug15b | Arduino 1.8.5
File Edit Sketch Tools Help
sketch_aug15b
Arduino Leonardo on COM13

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

These 5 sections are:

1. The **file** information section which has two titles: the *filename* and the *Arduino IDE version*.

2. The **menu** bar which holds five drop down menus: *File*, *Edit*, *Sketch*, *Tools*, and Help.

3. The **toolbar** consists of six buttons; five on the left side and one on the far right. These buttons give you easy access to the most frequently used functions. These buttons are:

    - The **Verify** button, the first button to the left, is used to check the code and make sure that it is free of mistakes.

    - The **Upload** button, the second button to the left, is responsible for uploading the code in the sketch file to the connected Arduino board.

    - The **New** button, the button in the middle, will create a new blank sketch.

    - The **Open** button, the button with an arrow pointed up, will allow the user to open a stored sketch file.

    - The **Save** button, the button with an arrow pointed down, will allow the user to save the sketch file.

    - The **Serial Monitor**, the button on the far left, used to open the monitor display. The monitor will show all the serial data sent and received by the serial interface.

4. The **sketch** window with a tab on top, holding the sketch name. This is where you write your code.

5. The **message** window at the bottom of the program which shows the status and error messages to the user.

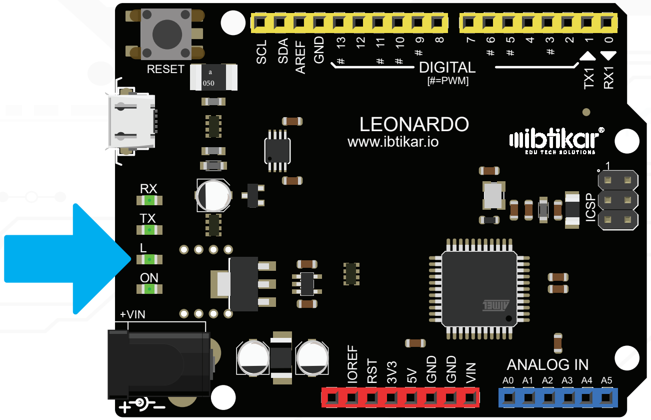
In the Arduino IDE window, you have two main sections where you write your code: the void setup() section and the void loop() section. The void setup() part of the code is executed when the device is initialised. It runs only once. It is used to declare the output and input of the device and other commands. The void loop() part of the code is executed after void setup(). Once initialised, it runs in a loop forever and it represents the actual job you need to do.

The Arduino files are called sketches, so the file name will start with the word *sketch* and be followed by the month and day. Arduino *sketches* are saved under the '**.ino**' extensions.

Now that you have installed the Arduino IDE, you are going to upload your first code.

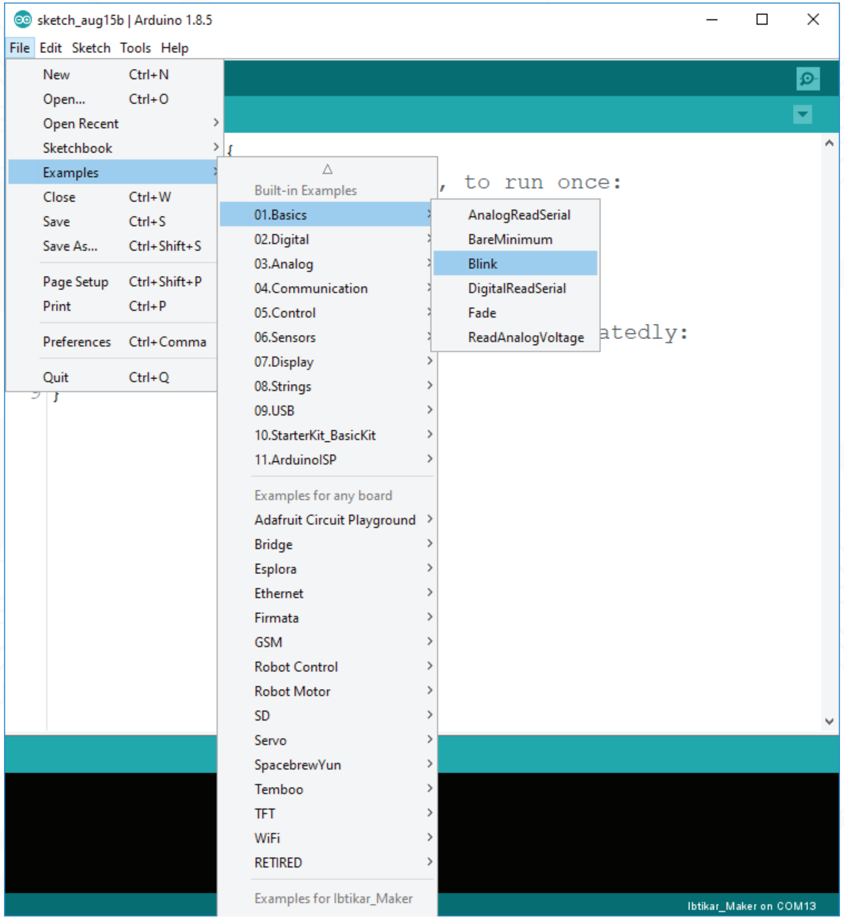
Your First Program

Here is a simple example of how you can use the Arduino microcontroller and the IDE to do an action. This will let you test the board to see if it works normally. In this test, the built-in LED in the board will blink using a ready-made example code from the Arduino IDE. The built-in LED is called the 'L' LED, as shown in the figure.

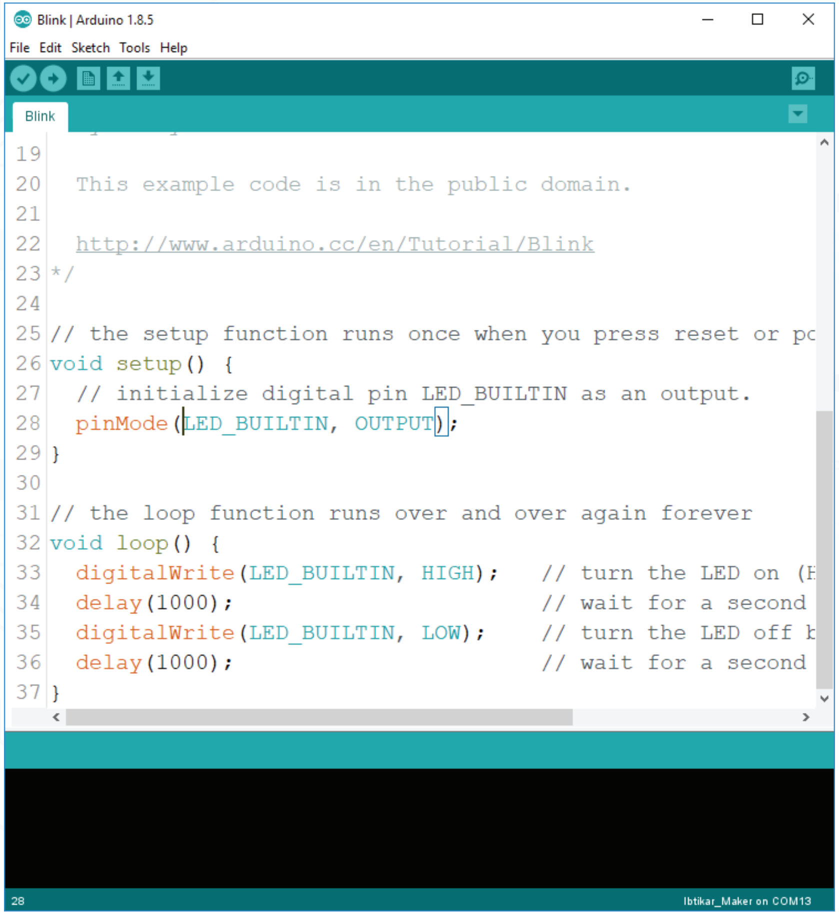


Follow these steps:

- Connect the board to your computer using the USB cable and your computer should recognise the board automatically
- Open the Arduino IDE
- Once it is open, go to File → Examples → 01.Basics → Blink. The Blink example will open as shown.

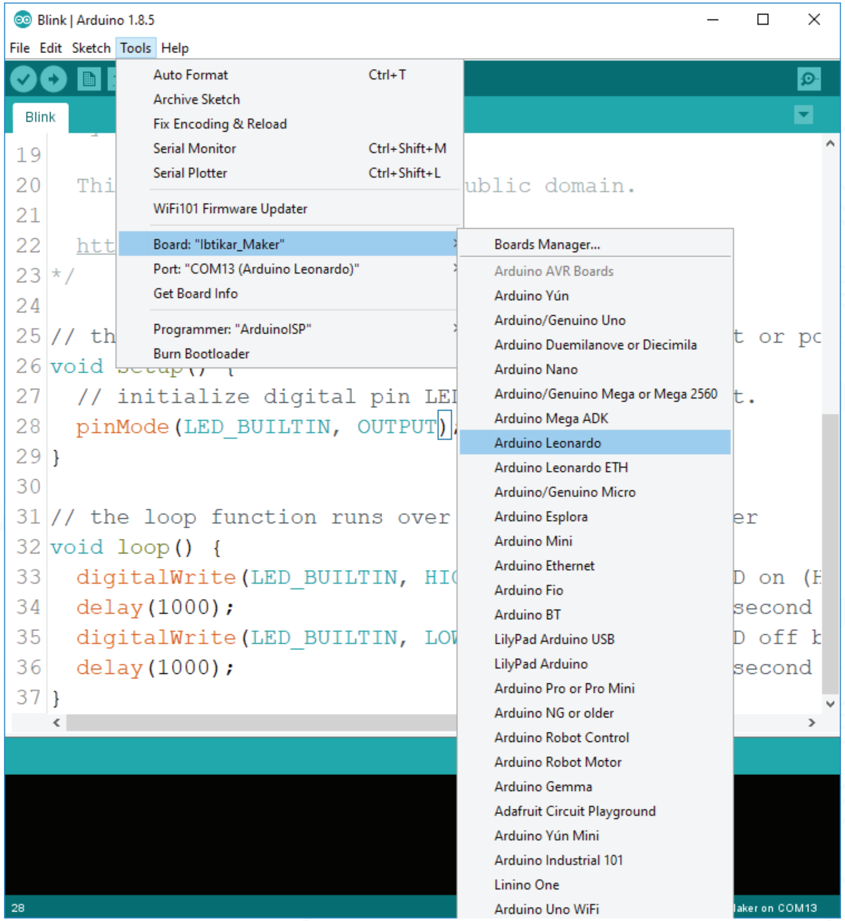


```
, to run once:

atedly:
```

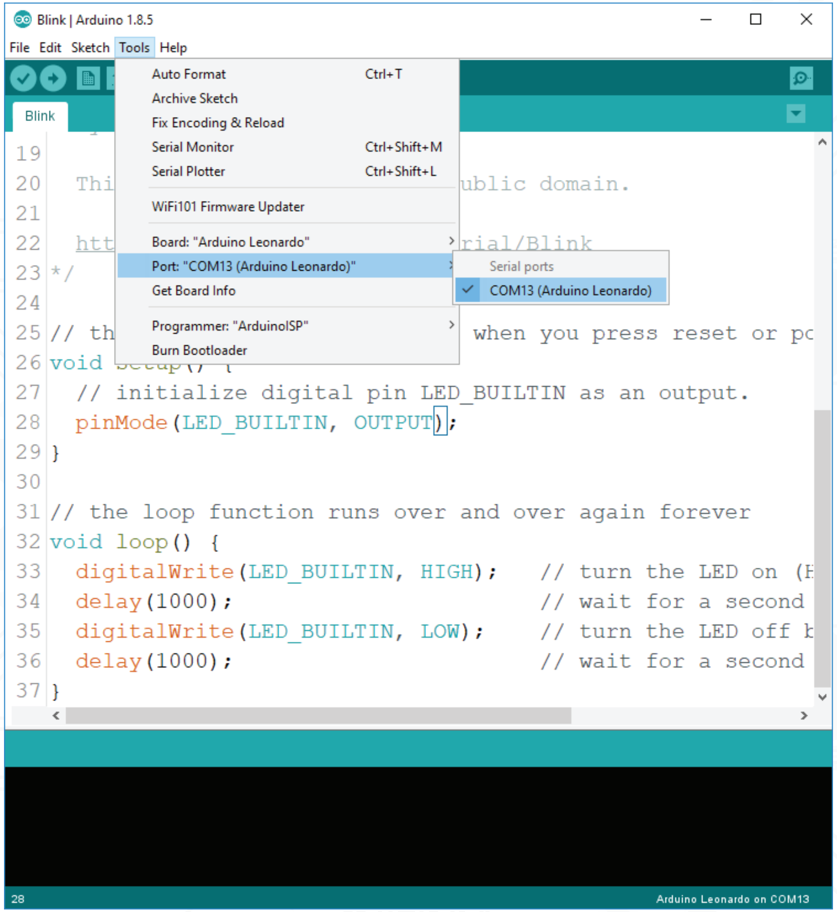


```
  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or pc
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (H
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off b
  delay(1000);                       // wait for a second
}
```

- Now, go to Tools and change the board to **Arduino Leonardo** and change the port whatever you board is connected to. In our case it is **COM13**.



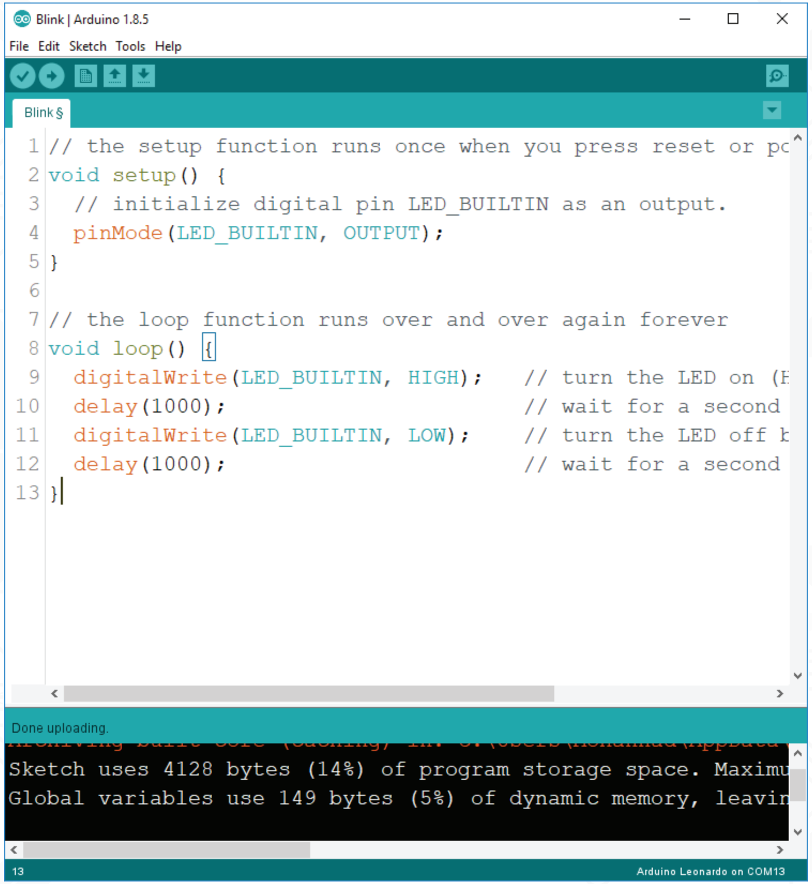
```
  Thi                                  ublic domain.

  htt
*/

// th                                  t or pc
void setup() {
  // initialize digital pin LEI        t.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over         er
void loop() {
  digitalWrite(LED_BUILTIN, HI(        D on (H
  delay(1000);                         second
  digitalWrite(LED_BUILTIN, LOW        D off b
  delay(1000);                         second
}
```


Blink | Arduino 1.8.5
File Edit Sketch Tools Help
Blink
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino Leonardo"
Port: "COM13 (Arduino Leonardo)"
Get Board Info
Programmer: "ArduinoISP"
Burn Bootloader
Serial ports
✓ COM13 (Arduino Leonardo)
28
Arduino Leonardo on COM13

```
  Thi                                  ublic domain.

  htt                                  rial/Blink
*/

// th                                  when you press reset or pc
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (H
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off b
  delay(1000);                       // wait for a second
}
```

- Click on Upload and wait a couple of seconds until the code is uploaded to your board. The message window should show "*Compiling sketch...*" and then it will change to "*Uploading*". The RX and TX LEDs will start blinking as they show that the sketch file is being transmitted from the IDE to the Arduino board.

- If you followed the steps correctly, you should see a message with “*Done uploading*” and the RX and TX LEDs will stop blinking.
- The ‘L’ LED will now start blinking once every second.

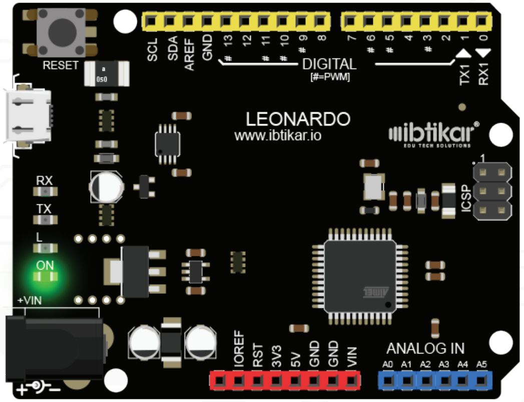


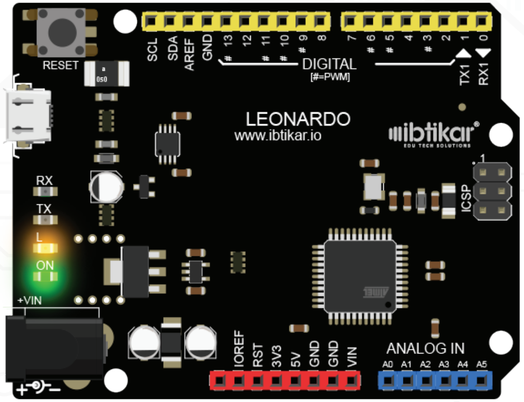
```
// the setup function runs once when you press reset or pc
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (H
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off b
  delay(1000);                       // wait for a second
}
```

Done uploading.

```
Sketch uses 4128 bytes (14%) of program storage space. Maximu
Global variables use 149 bytes (5%) of dynamic memory, leavin
```

13 Arduino Leonardo on COM13





The previous test is a basic test, but if you see the LED blinking, you are sure that your board is detected, and the board name and port are selected correctly.

Power Options

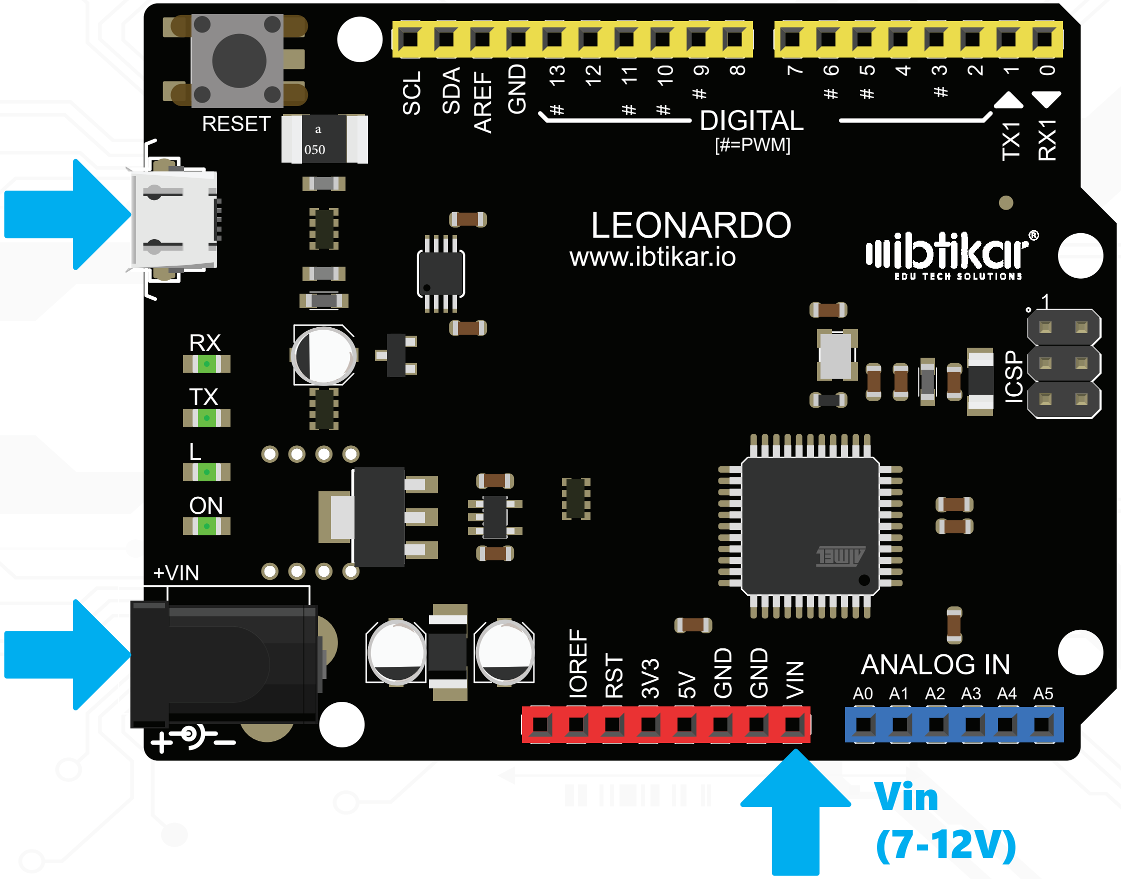
There are several ways to power the Arduino board such as, the USB cable, power jack, or the Vin pin. Choosing the power option depends on the application and whether you need to run the Arduino board in a stationary place or within a movable machine. In the Discovery Kit, there is a USB cable and two 9V battery clips one with a power jack and the other is with cables.

Needed Libraries

Some of the following examples need libraries to work properly. A **library** is a collection of programming instructions that a programmer can *'call'* when writing a code. With a library, there is no need to rewrite the code again. Some libraries are installed by default (Liquid Crystal, Servo and Stepper) and some need to be downloaded manually.

Make sure you have the following libraries (zip format) installed to the Arduino IDE.

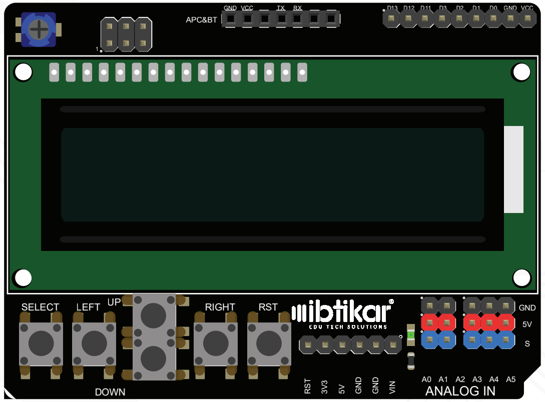
- DHT-lib.zip
- IRremote-2.2.3.zip



USB cable connector (fixed on 5V)

Power jack (7-12V)

Vin (7-12V)



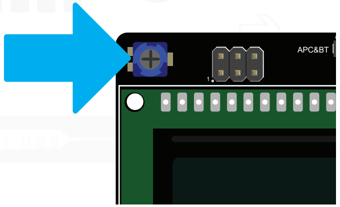
LCD Shield

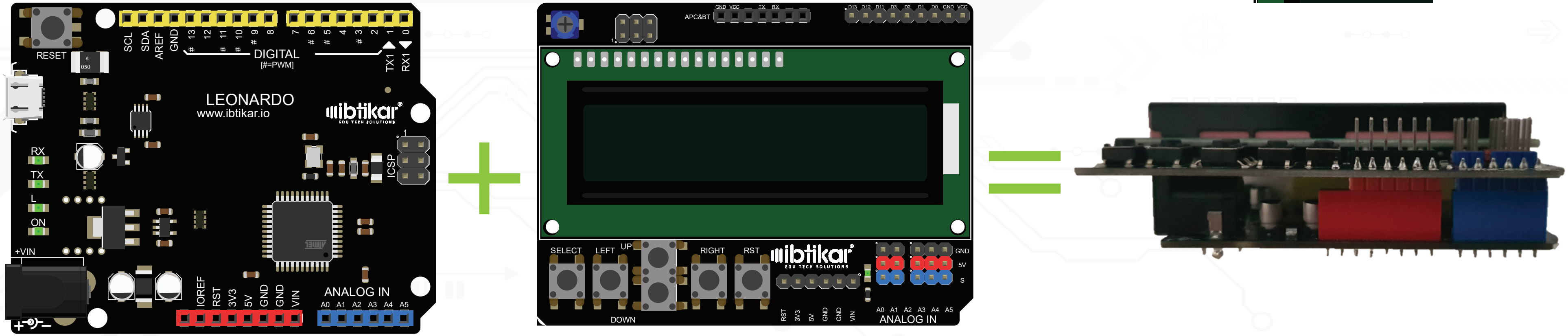
The LCD (Liquid Crystal Display) shield is a shield with a type of screen that uses a special type of substance - Liquid Crystals. The structure of liquid crystals can be changed by an electric current that changes the displayed data.

Note - You also need:
◉ Arduino Leonardo

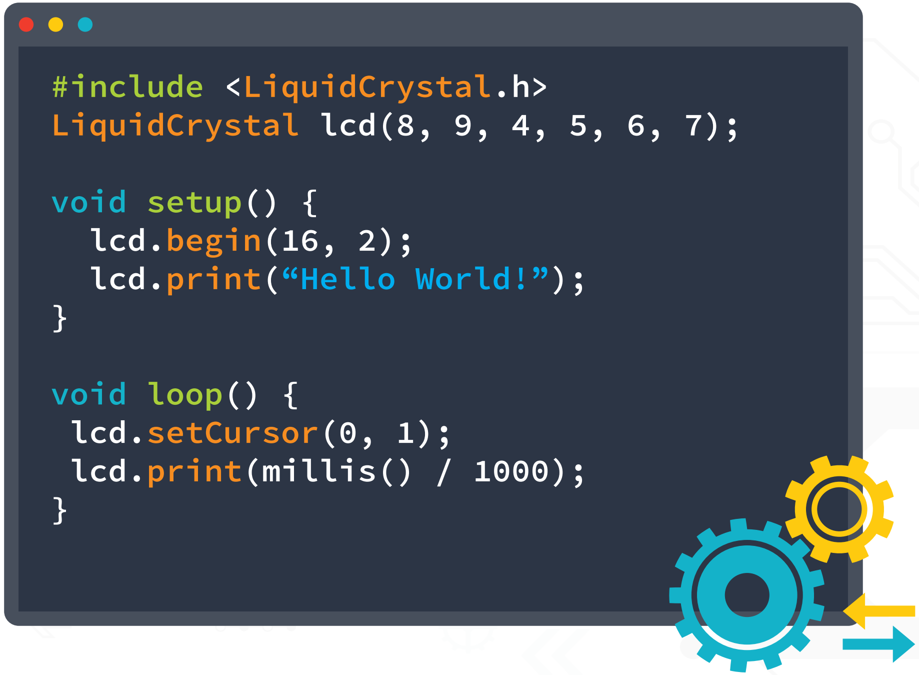
Don't Forget

This shield is attached to the Arduino board and stacks on top of it, "shielding" it.
You can change the brightness of the LCD display using the small screw driver in your kit to adjust the small knob on the LCD shield.





Sample Code – Part 1



```
#include <LiquidCrystal.h>
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

void setup() {
  lcd.begin(16, 2);
  lcd.print(“Hello World!”);
}

void loop() {
 lcd.setCursor(0, 1);
 lcd.print(millis() / 1000);
}
```

Result – Part 1

After uploading the code to Arduino, the LCD will display “Hello World!” in the first row. It will then set the screen cursor to column 0 and line 1. Line 1 is the second row since counting begins with 0. In the second row, the number of seconds since the last reset of the board will be printed on the LCD screen.

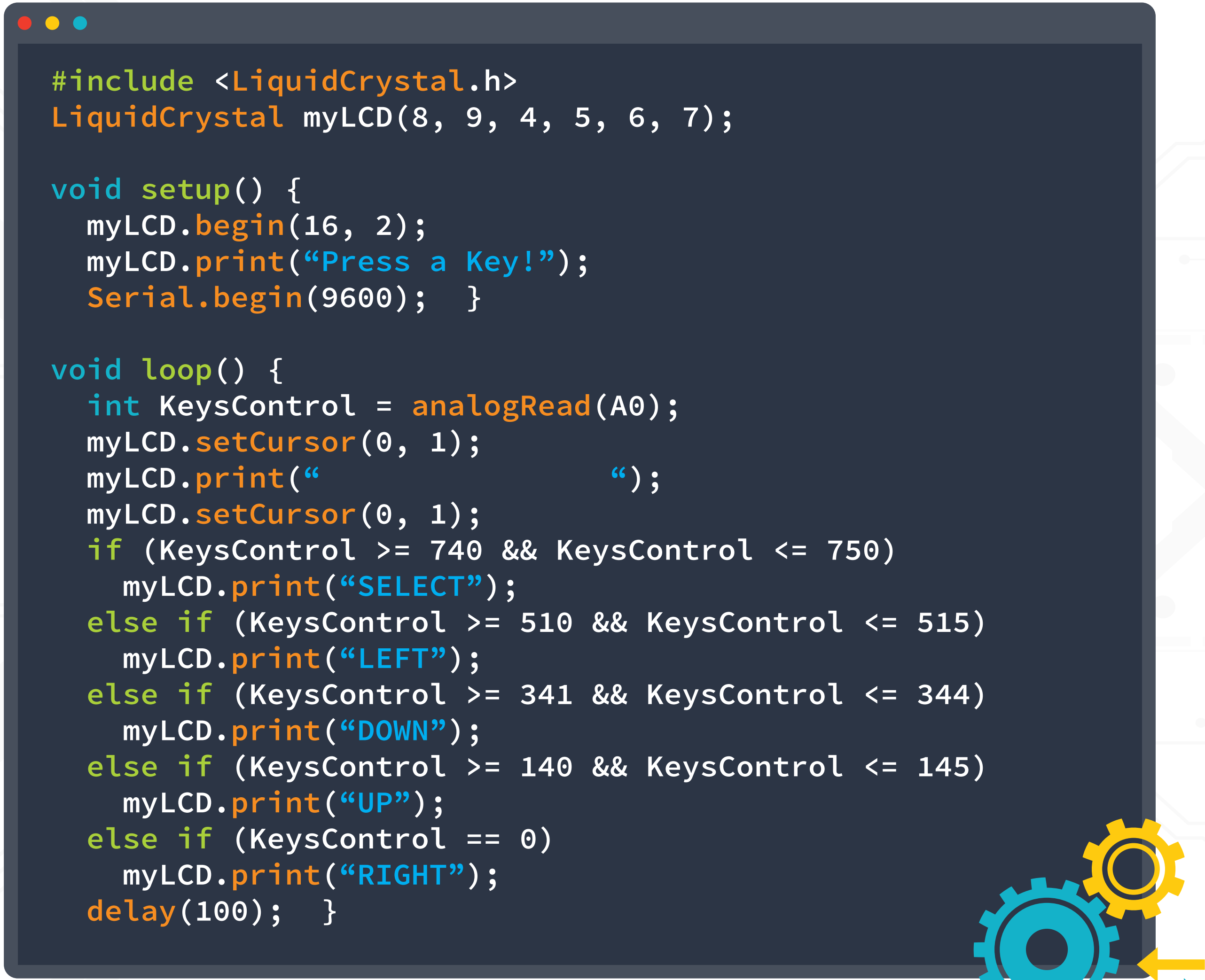
If you are using the LCD for the first time, make sure the brightness is adjusted using the screwdriver. Otherwise, you will not see the text on the screen.

There are six push buttons placed on the LCD screen shield: SELECT, LEFT, RIGHT, UP, DOWN, and RESET. We will use the first five keys, since the sixth button is the RESET which is used to reset the Arduino board. These keys are connected to the analog port A0, where each one on these keys has different analog values and can be used to trigger an event or action.

To check which key is pressed on the LCD shield, you need to read the current analog value of the pin A0, to compare it with the corresponding ranges in the following table.

Key	Analog Values Range
SELECT	740-750
LEFT	510-515
DOWN	341-344
UP	140-145
RIGHT	0

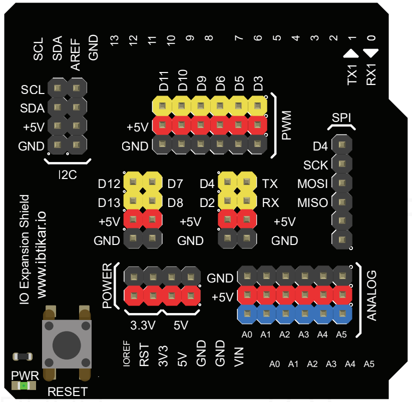
Sample Code – Part 2



```
#include <LiquidCrystal.h>
LiquidCrystal myLCD(8, 9, 4, 5, 6, 7);

void setup() {
  myLCD.begin(16, 2);
  myLCD.print(“Press a Key!”);
  Serial.begin(9600);  }

void loop() {
  int KeysControl = analogRead(A0);
  myLCD.setCursor(0, 1);
  myLCD.print(“                “);
  myLCD.setCursor(0, 1);
  if (KeysControl >= 740 && KeysControl <= 750)
    myLCD.print(“SELECT”);
  else if (KeysControl >= 510 && KeysControl <= 515)
    myLCD.print(“LEFT”);
  else if (KeysControl >= 341 && KeysControl <= 344)
    myLCD.print(“DOWN”);
  else if (KeysControl >= 140 && KeysControl <= 145)
    myLCD.print(“UP”);
  else if (KeysControl == 0)
    myLCD.print(“RIGHT”);
  delay(100);  }
```

Result – Part 2

After uploading the code to Arduino, the user will be asked to press a key as the LCD will display the message “Press a Key!” in the first row. It will then read the analog pin A0, set the screen cursor to column 0 and line 1, clear line 1 (second row) only, set the screen cursor again to column 0 and line 1 and compare the analog value. The name of the pressed button will be displayed on the LCD screen based on the read value.



Digital Pins	+5V Pins	Ground Pins	Analog Pins

I/O Expansion Shield

While building any electronic system, you might need to add many input and output modules to your system.

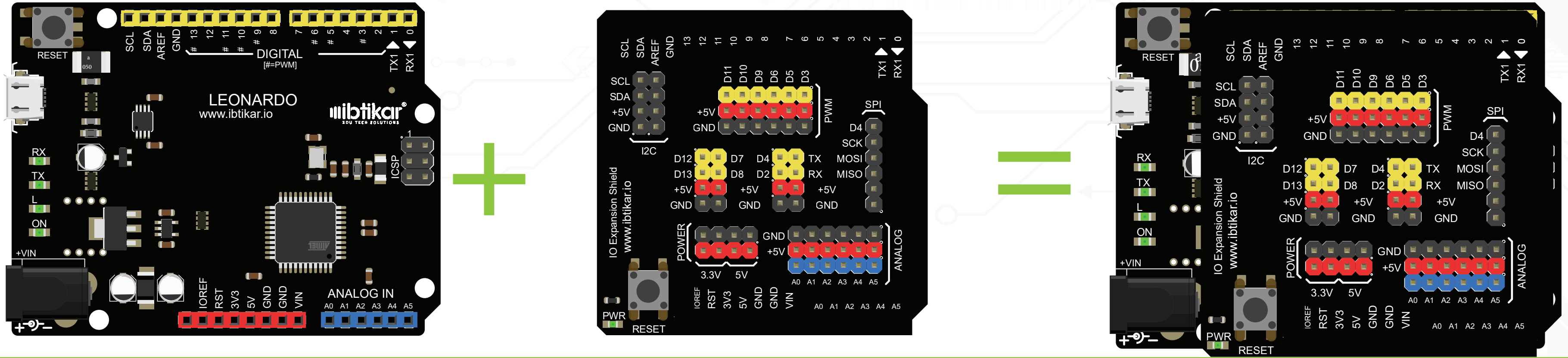
Note - You also need:
⦿ Arduino Leonardo

Don't Forget

To make it easy to connect these modules to the Arduino microcontroller, you need to use the I/O Expansion Shield. This shield works as a bridge between the Arduino and the other modules, using coloured wires to connect.

On the shield, there are different pin categories: the yellow pins stand for the digital modules (sensors and actuators) and the blue pins stand for the analog modules (sensors). Analog and digital pins are marked on the expansion shield. The area with (A) is for analog pins, and the area with (D) is for digital pins. When you connect any module to the shield you need to match the colours of the cable with the pin headers located on the shield.

After connecting the I/O Expansion Shield onto the Arduino board you can use the other modules easily (input and output). Input is when the microcontroller receives a signal from an input module (a sensor), and output is when the microcontroller sends a signal to an output module (an actuator).



There are other pins used for different applications, each category name is written on the shield. This table shows the name of each category and its usage:

Pin Category	Usage
I2C	Inter-Integrated Circuit (I2C), or Two-Wire Interface. Using this protocol, many I2C devices can be connected on the same two wires.
SPI	Serial Peripheral Interface (SPI), this communication protocol links the microcontroller with different devices for sending/receiving data. It has a separate clock line for data synchronization, two data lines, and one servant selector line for each slave device.
PWM	Pulse Width Modulation (PWM) is a technique used to gain analog results with digital means by switching signals between HIGH and LOW, thousands of times a second. Using this, you can simulate voltages between 0 and 5 volts.
Power	In this category there are two different voltages: 5 and 3.3 volts. If you have an external I/O device that needs 3.3 volts as a maximum voltage, you must use the 3.3V pins. All the modules included in the Discovery Kit operate on 5 volts.

Dealing with Digital Signals

All the pins named with (D) and (A) on the shield, D0-D13 and A0-A5, can be used as digital I/O pins based on the defined pin mode in the code, input or output. Pins D0-D13 can be defined directly using their numbers in the IDE code as input or output, but the analog pins A0-A5 follow a different order; they should be defined as follows:

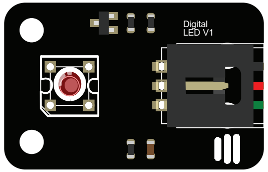
A0 = D18	A1 = D19	A2 = D20	A3 = D21	A4 = D22	A5 = D23

For example: A0 = D18 means that the analog pin A0 can be defined in the code as digital pin D18, and so on. This feature is only for the digital pins.

Dealing with Analog Signals

For the analog I/O, there are specific pins used to send signals and others used to receive. unlike the digital pins, the same pin cannot be used for sending or receiving analog signals.

The pins A0-A5 are used to read analog signals, and the pins included within the PWM category are used to send a signal that gives an analog behaviour to the actuator.

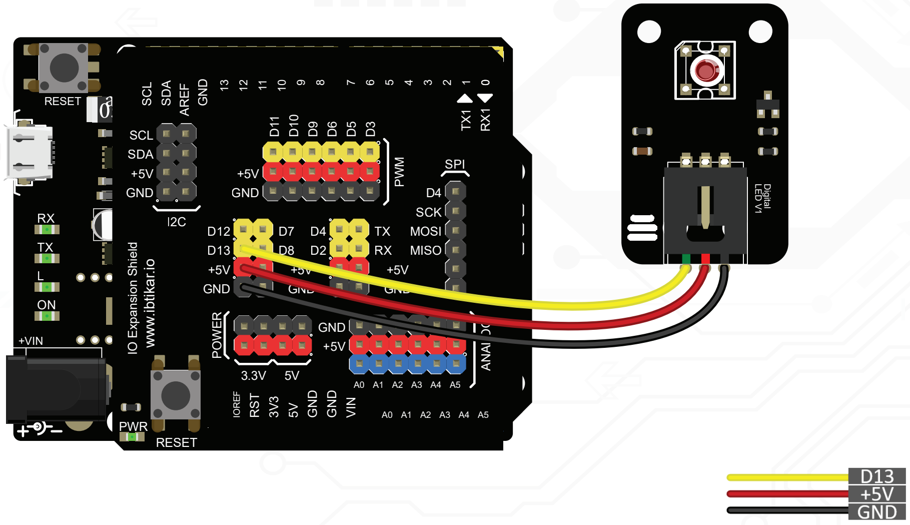


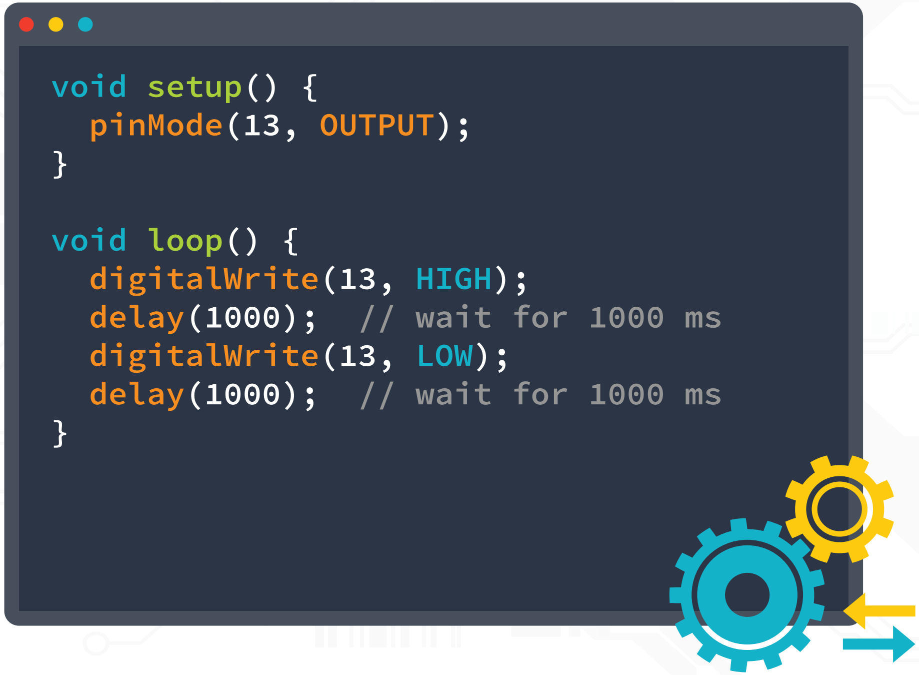
LED Module

LED stands for Light Emitting Diode which is a two-terminal electronic component that allows current to flow only in one direction. LEDs emit light when current passes through them.

Note - You also need:
- Arduino Leonardo
- I/O Expansion Shield
- Cable

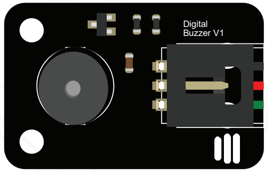
Don't Forget





```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);  // wait for 1000 ms
  digitalWrite(13, LOW);
  delay(1000);  // wait for 1000 ms
}
```

The LED will start blinking with a delay of 1 second.

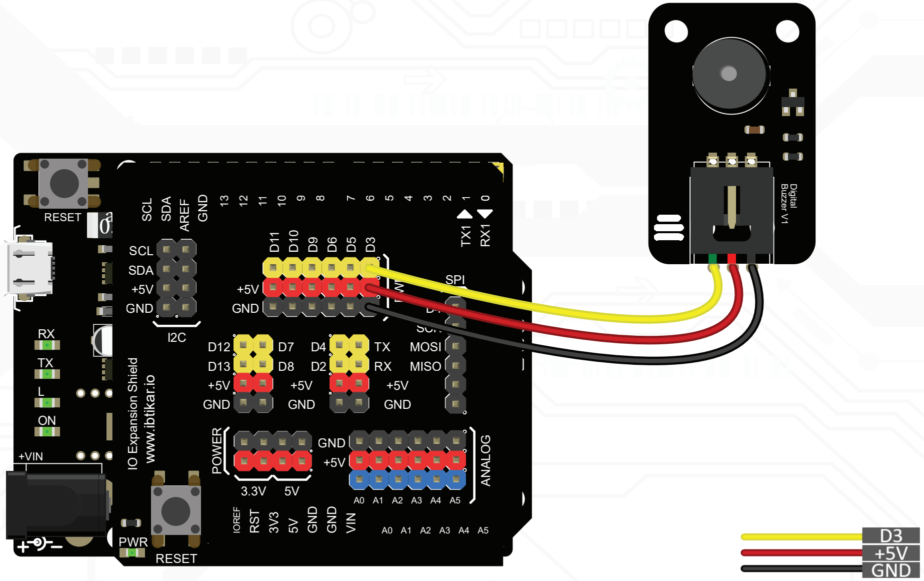


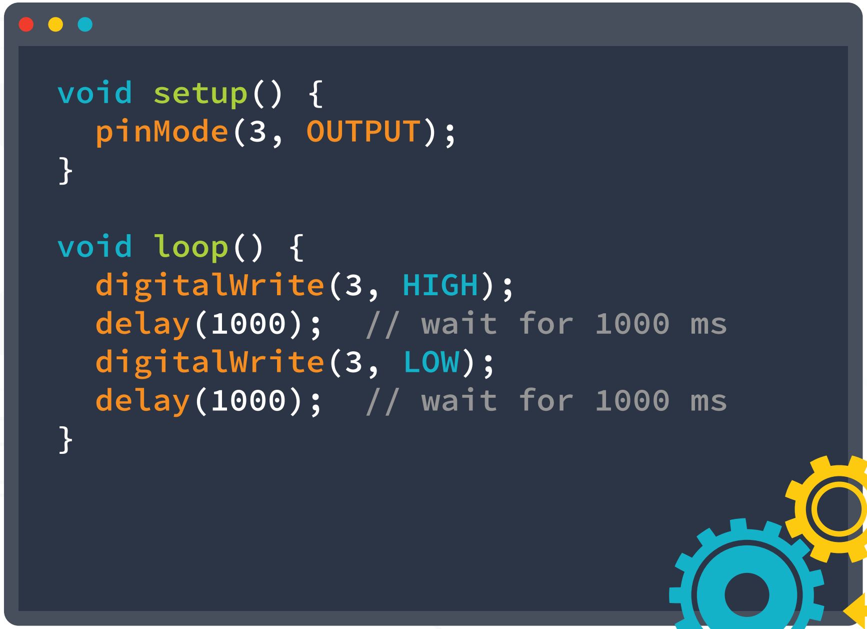
Buzzer

The digital buzzer module is an audio signalling device which is a common piezoelectric actuator that generates sound by vibrations.

Note - You also need:
- Arduino Leonardo
- I/O Expansion Shield
- Cable

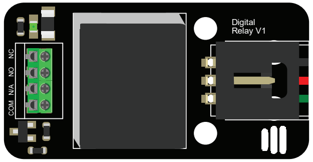
Don't Forget





```
void setup() {
  pinMode(3, OUTPUT);
}

void loop() {
  digitalWrite(3, HIGH);
  delay(1000);  // wait for 1000 ms
  digitalWrite(3, LOW);
  delay(1000);  // wait for 1000 ms
}
```

The buzzer will beep for 1 second and stop for another second. The beeping will repeat without stopping.



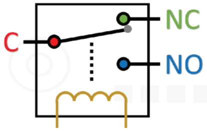
5A Relay Module

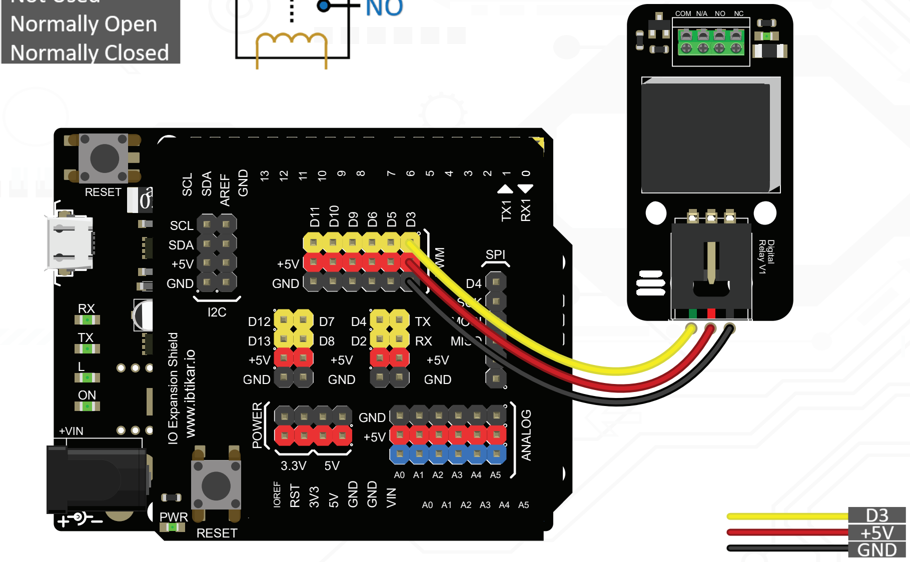
The relay module is basically an electromagnetic switch that has an internal magnetic switch that can be enabled by a small electrical signal. It is used when there is a need to use a low power control signal to operate a larger load. This module can handle a load up to 5A on its terminals. Do not test the relay with real electrical loads without supervision of an electrician.

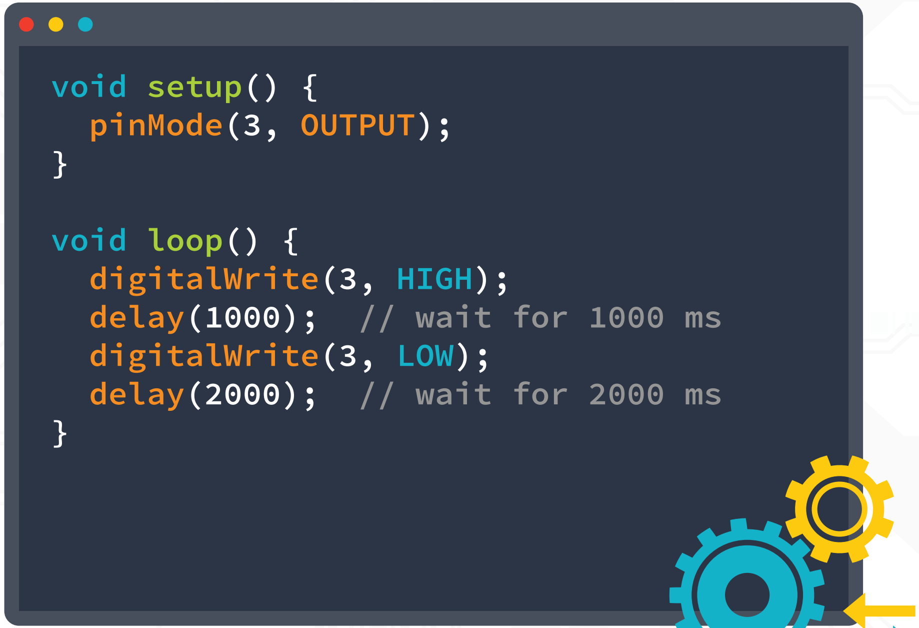
Note - You also need:
- ⦿ Arduino Leonardo
- ⦿ I/O Expansion Shield
- ⦿ Cable

Don't Forget

🟩	COM	Common
🟩	N/A	Not Used
🟩	NO	Normally Open
🟩	NC	Normally Closed

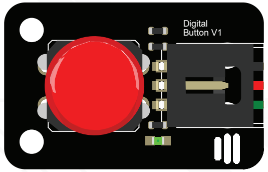






```
void setup() {
  pinMode(3, OUTPUT);
}

void loop() {
  digitalWrite(3, HIGH);
  delay(1000);  // wait for 1000 ms
  digitalWrite(3, LOW);
  delay(2000);  // wait for 2000 ms
}
```

After uploading the code to Arduino, you will see the LED on the relay board blinking and you will start hearing clicks from the relay module. This sound is because of the opening and closing of the mechanical switch inside.



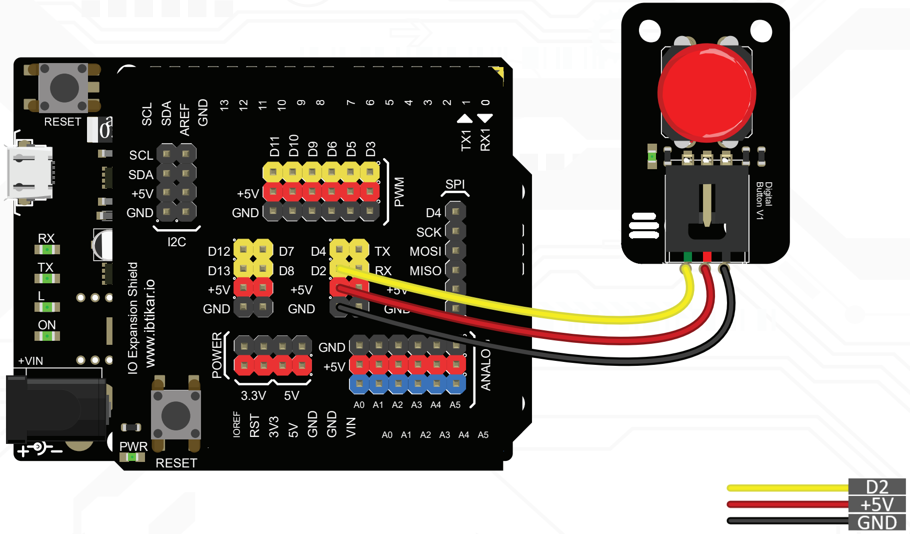
Push Button

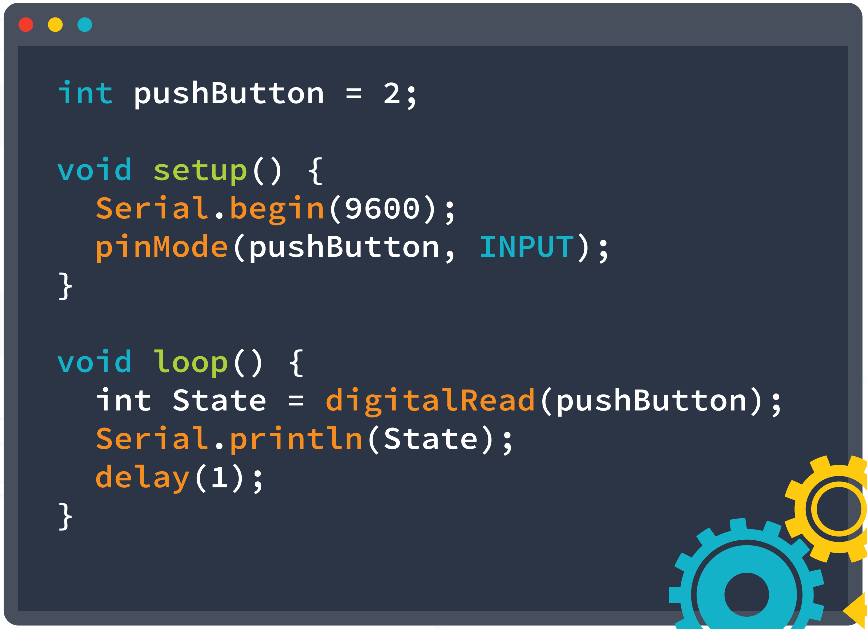
The push button module is one of the most common user-interface devices that has two states, either pressed or released. Pressed means 1 and released means 0.

Note - You also need:

- ⦿ Arduino Leonardo
- ⦿ I/O Expansion Shield
- ⦿ Cable

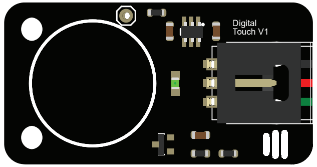
Don't Forget





```
int pushButton = 2;

void setup() {
  Serial.begin(9600);
  pinMode(pushButton, INPUT);
}

void loop() {
  int State = digitalRead(pushButton);
  Serial.println(State);
  delay(1);
}
```

After uploading the code to Arduino, open the Serial Monitor. Now press and release the button and read the change in the values between 1 and 0 on the Serial Monitor.

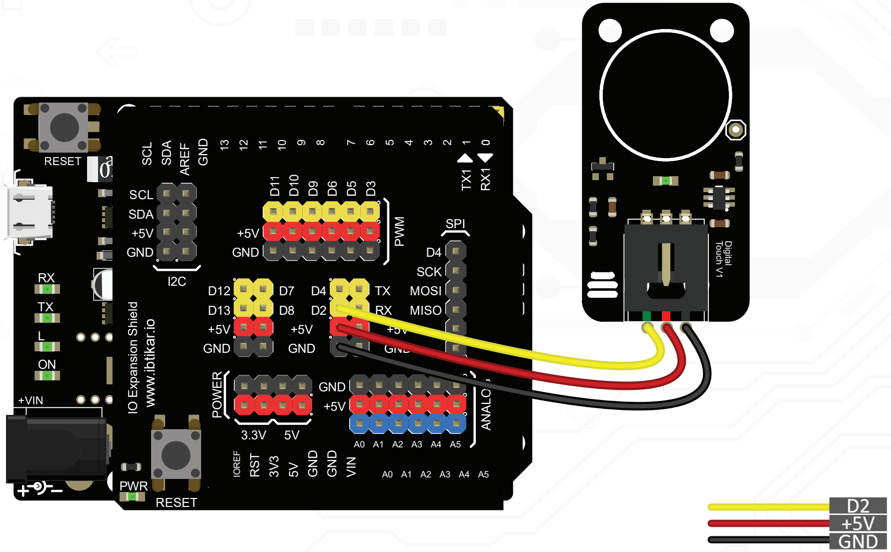


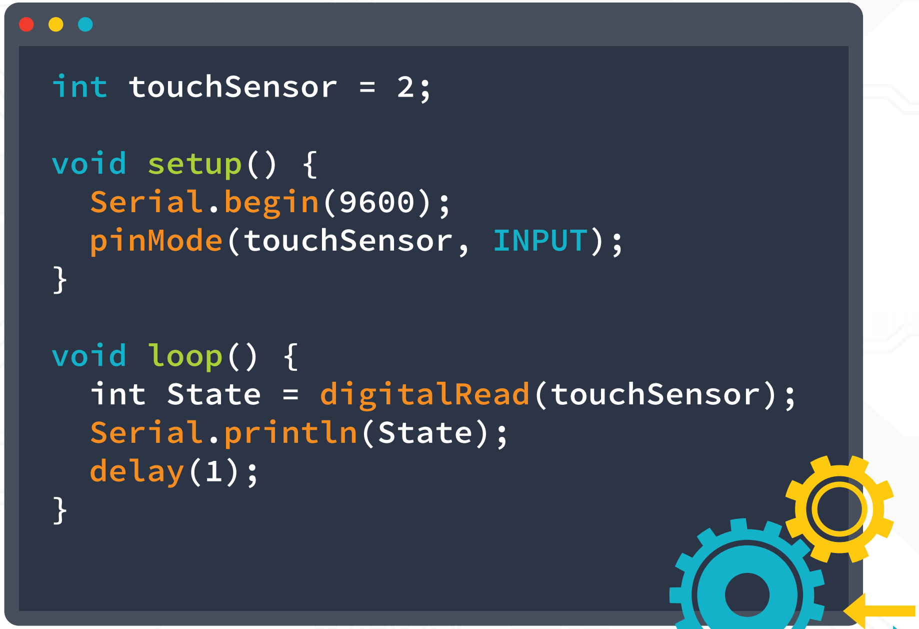
Capacitive Touch Sensor

An electrical capacitance-based sensor that has an electronic chip which senses the change in the electrical capacitance on the board where the human finger acts like an electrical capacitor.

Note - You also need:
- ⦿ Arduino Leonardo
- ⦿ I/O Expansion Shield
- ⦿ Cable

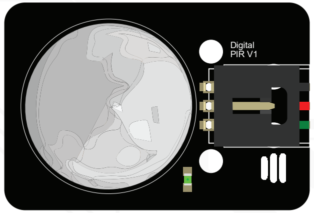
Don't Forget





```
int touchSensor = 2;

void setup() {
  Serial.begin(9600);
  pinMode(touchSensor, INPUT);
}

void loop() {
  int State = digitalRead(touchSensor);
  Serial.println(State);
  delay(1);
}
```

After uploading the code to Arduino, open the Serial Monitor. Now touch and untouch the sensor and read the change in the values between 1 and 0 on the Serial Monitor.

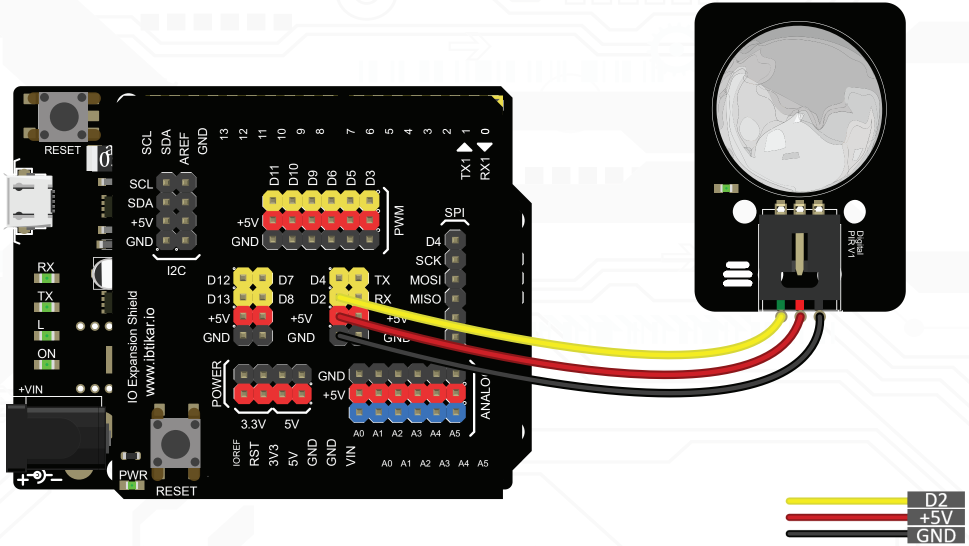


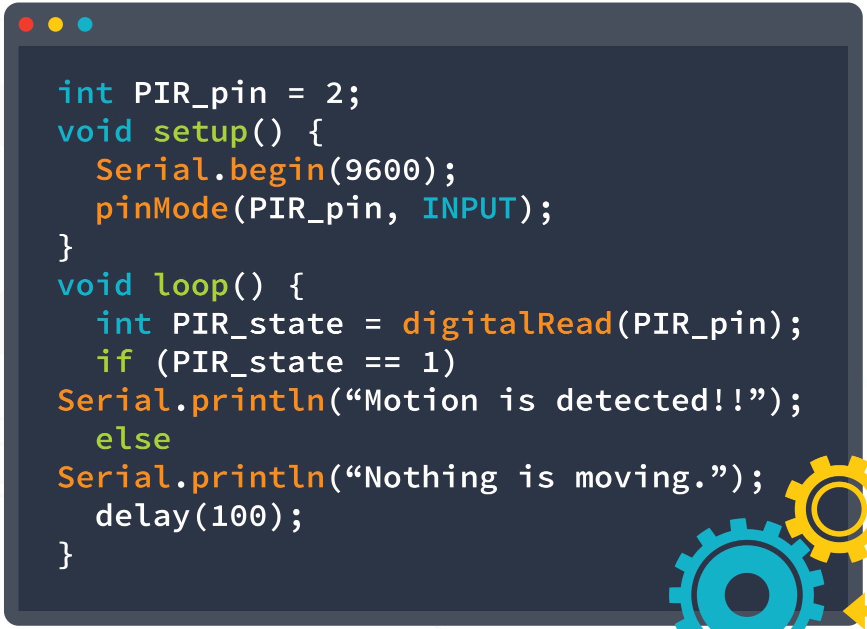
Infrared Motion Sensor

A PIR sensor (Passive Infrared) gets activated when it is exposed to heat from bodies that emit energy, like humans and animals. The PIR sensor compares the new signal with the previous signal. A change between the readings, means there is motion.

Note - You also need:
- Arduino Leonardo
- I/O Expansion Shield
- Cable

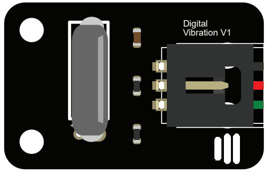
Don't Forget





```
int PIR_pin = 2;
void setup() {
  Serial.begin(9600);
  pinMode(PIR_pin, INPUT);
}
void loop() {
  int PIR_state = digitalRead(PIR_pin);
  if (PIR_state == 1)
Serial.println(“Motion is detected!!”);
  else
Serial.println(“Nothing is moving.”);
  delay(100);
}
```

Test the motion sensor by passing your hand over the top of the sensor (do not touch the sensor). The PIR sensor updates its output signal based on the variations in the infrared signal from the objects. Now keep your hand stationary over the sensor.



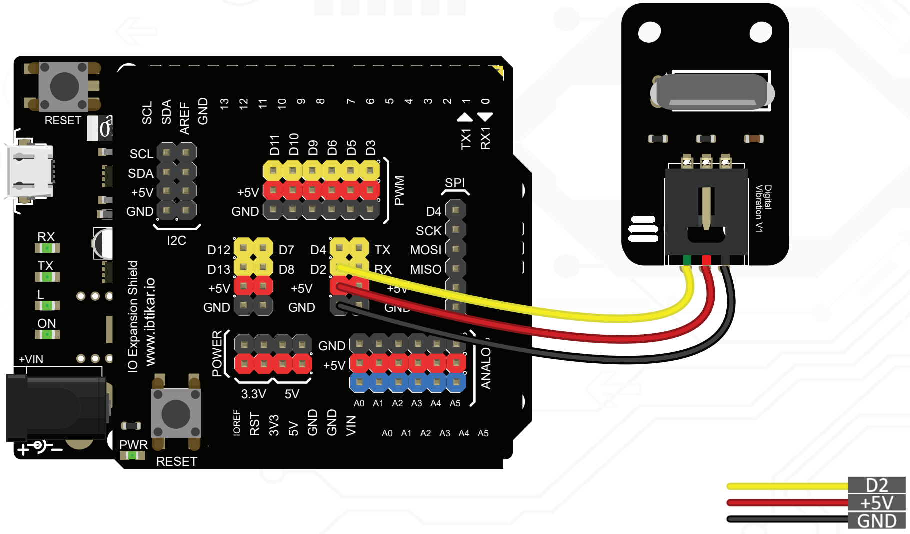
Digital Vibration Sensor

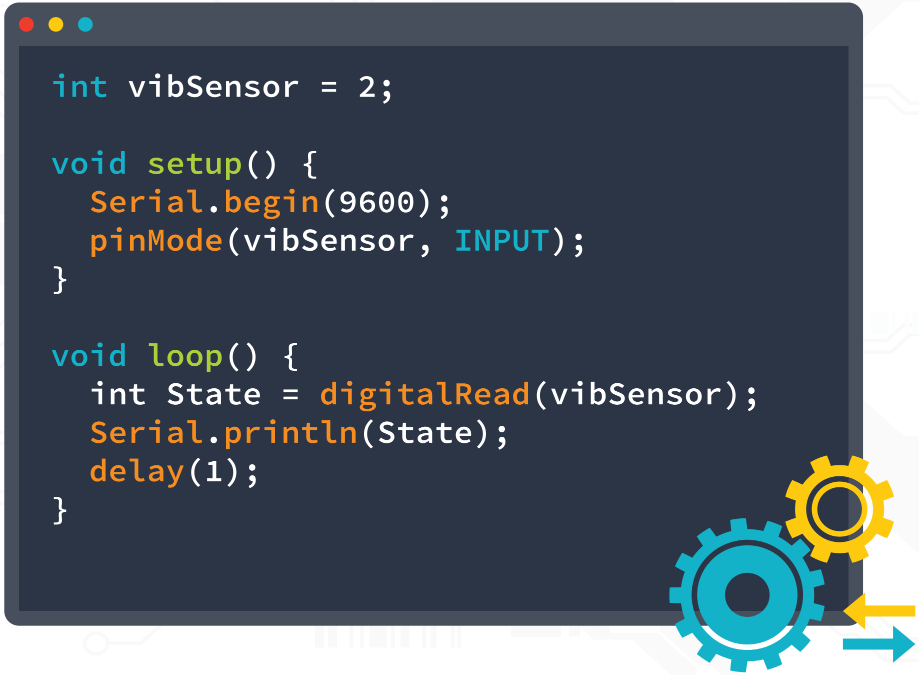
A digital vibration sensor is an electromechanical sensor that detects vibration and acts as an electrical switch. This digital vibration module has a pull-up resistor configuration, where you must vibrate the sensor to get a logic 0, otherwise it has a logic 1 as a default value.

Note - You also need:

- Arduino Leonardo
- I/O Expansion Shield
- Cable

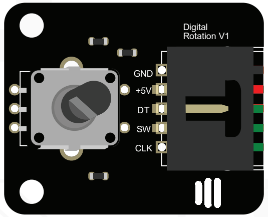
Don't Forget





```
int vibSensor = 2;

void setup() {
  Serial.begin(9600);
  pinMode(vibSensor, INPUT);
}

void loop() {
  int State = digitalRead(vibSensor);
  Serial.println(State);
  delay(1);
}
```

The value of the digital vibration module can be displayed on the screen, this can be done by opening the Serial Plotter tool from the *Tools* tab in the Arduino IDE window. Now shake the sensor and see the change in the values between 1 and 0 on the Serial Plotter.

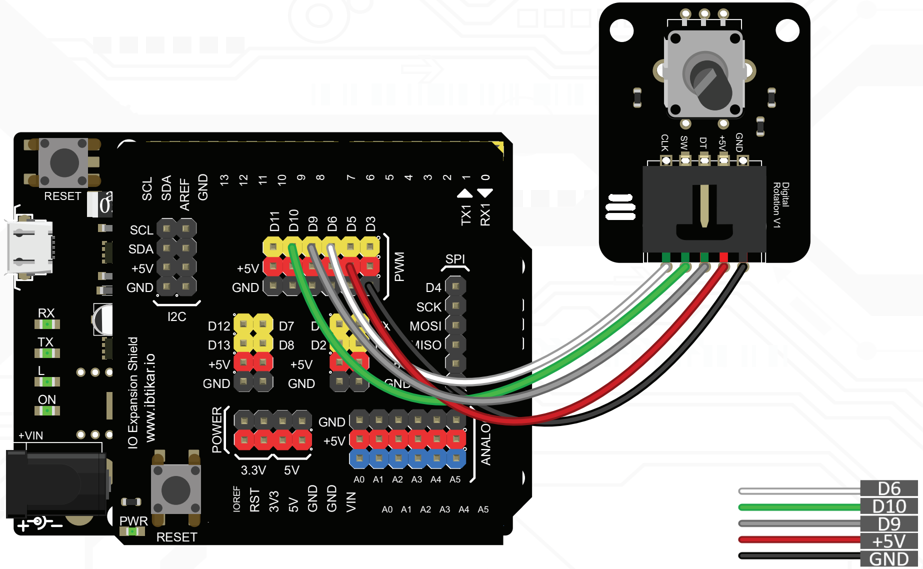


Note - You also need:
- ⦿ Arduino Leonardo
- ⦿ I/O Expansion Shield
- ⦿ Cable

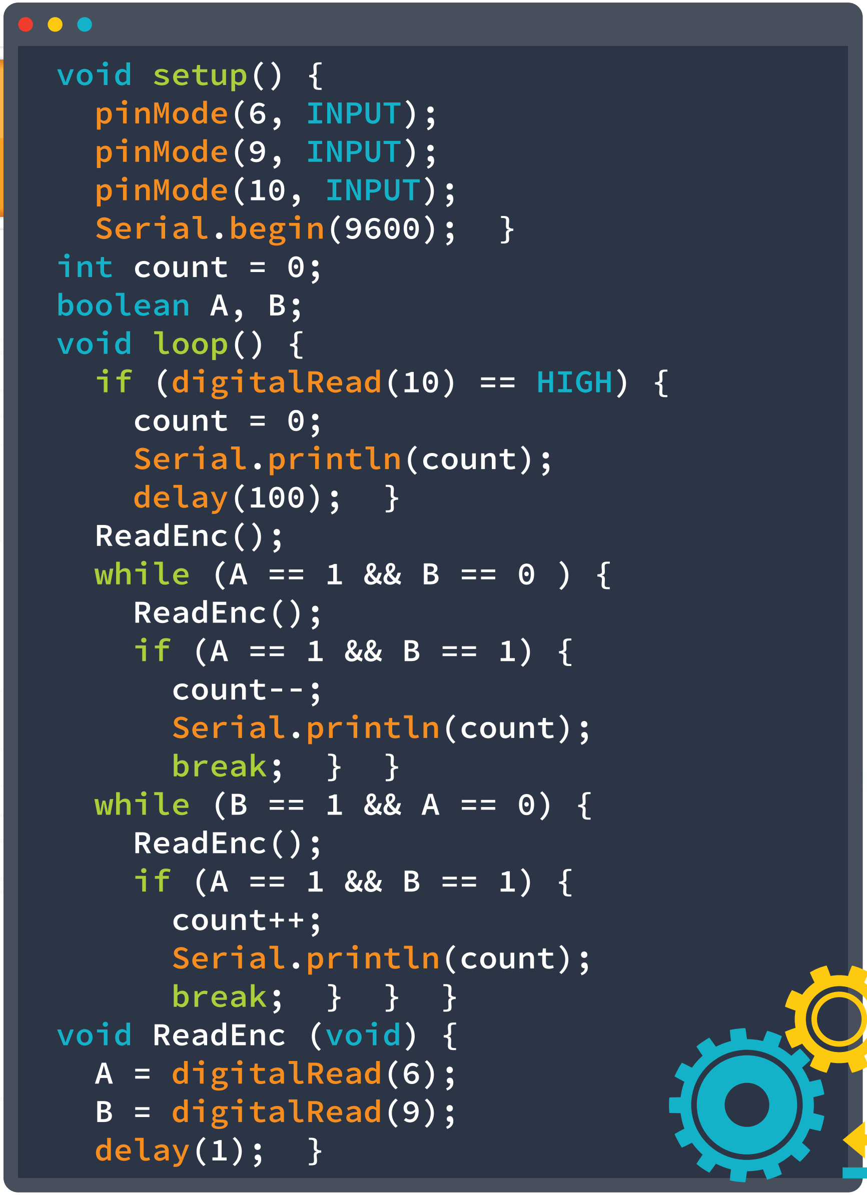
Don't Forget

Digital Rotation Sensor

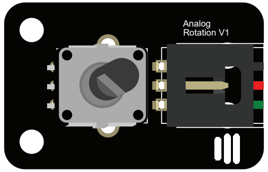
A digital rotation sensor (or an encoder) is an electro-mechanical device that senses the rotary motion (position and speed for instance) from an axis and then send an electrical signal to the microcontroller.



After uploading the code to your Arduino, open the Serial Monitor, then rotate the digital rotation sensor clockwise and counter clockwise. The Serial Monitor displays the value of the rotational sensor. You can reset the value of the counter to 0 by pressing the knob itself.



```
void setup() {
  pinMode(6, INPUT);
  pinMode(9, INPUT);
  pinMode(10, INPUT);
  Serial.begin(9600);   }
int count = 0;
boolean A, B;
void loop() {
  if (digitalRead(10) == HIGH) {
    count = 0;
    Serial.println(count);
    delay(100);   }
  ReadEnc();
  while (A == 1 && B == 0 ) {
    ReadEnc();
    if (A == 1 && B == 1) {
      count--;
      Serial.println(count);
      break;  }  }
  while (B == 1 && A == 0) {
    ReadEnc();
    if (A == 1 && B == 1) {
      count++;
      Serial.println(count);
      break;  }  }  }
void ReadEnc (void) {
  A = digitalRead(6);
  B = digitalRead(9);
  delay(1);  }
```

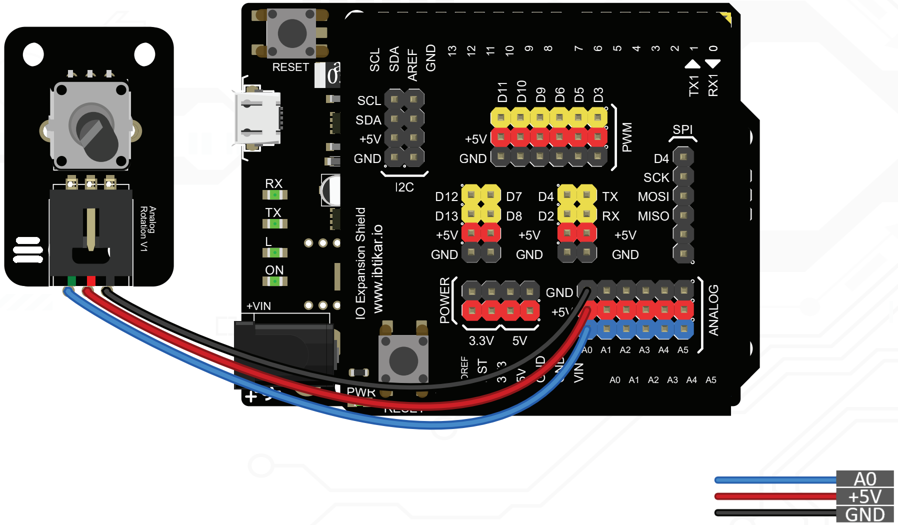


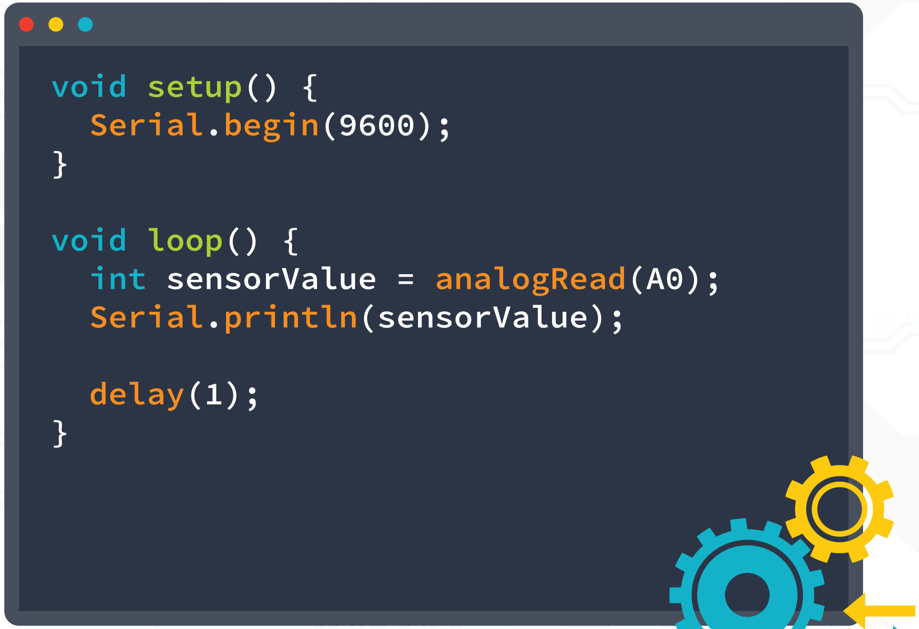
Analog Rotation Sensor

An analog rotation sensor (or a potentiometer) consists of a fixed value resistor and a rotating wiper. Manipulating the wiper will divide the fixed value resistor into two parts. Its main use is as a variable resistor or voltage divider.

Note - You also need:
- Arduino Leonardo
- I/O Expansion Shield
- Cable

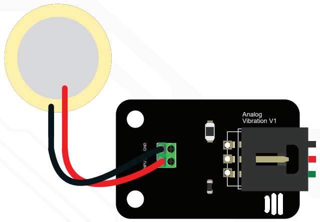
Don't Forget





```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);

  delay(1);
}
```

After uploading the code to your Arduino, open the Serial Monitor, then rotate the analog rotation sensor. The Serial Monitor displays the value of the rotational sensor; it varies between 0 and 1023.

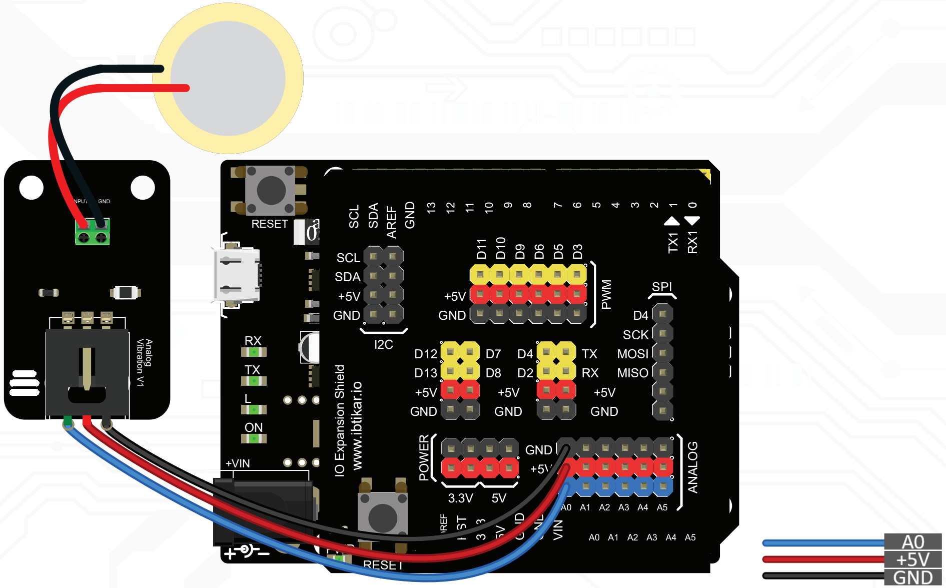


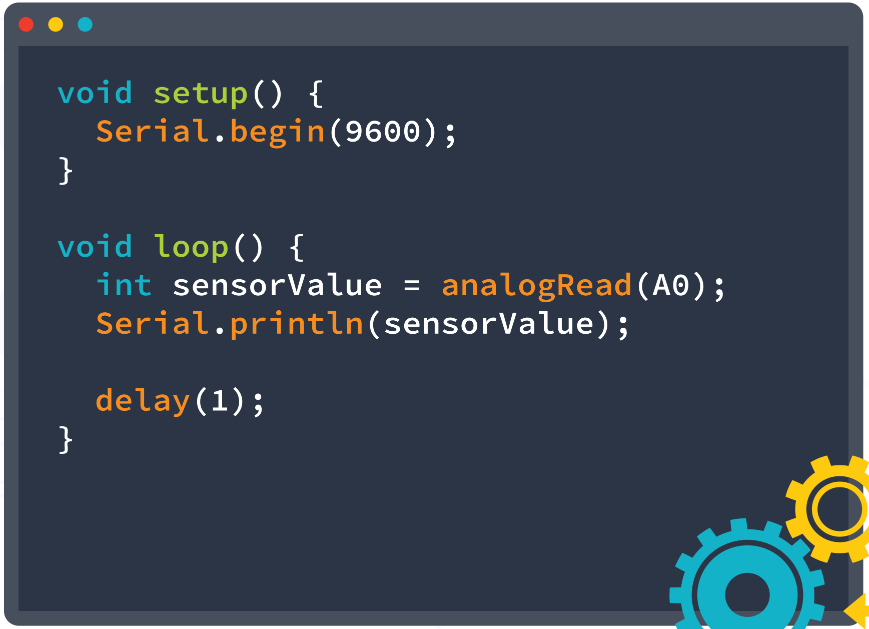
Analog Vibration Sensor

An analog vibration sensor converts mechanical energy into electrical energy. It generates a variable DC voltage, between 0 and 5V when an external vibration is applied on the piezo circular disc.

Note - You also need:
- ⦿ Arduino Leonardo
- ⦿ I/O Expansion Shield
- ⦿ Cable

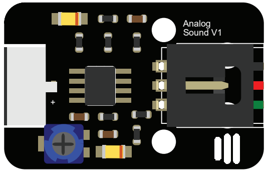
Don't Forget





```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);

  delay(1);
}
```

After uploading the code to your Arduino, open the Serial Plotter. Put your phone on the sensor. Change the vibration level from your phone and notice the change in the Serial Plotter. The value of the analog vibration sensor varies between 0 and 1023.

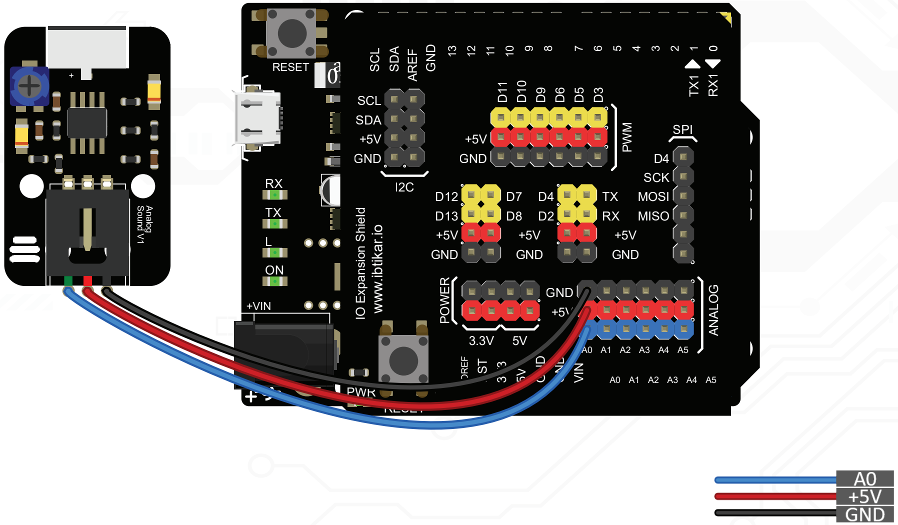


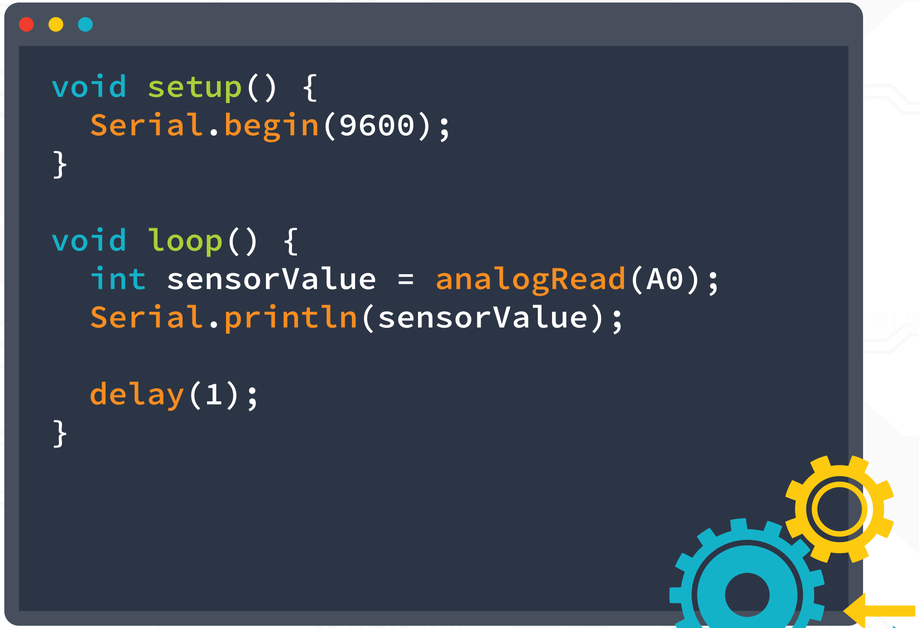
Sound Sensor

The analog sound module is an electronic sensor that measures the analog signal of a sound wave applied to the sensor, to generate an output voltage, proportional to the received sound wave. You can change the sensitivity of the sensor by tuning the knob located on the sensor, depending on the application.

Note - You also need:
- ⦿ Arduino Leonardo
- ⦿ I/O Expansion Shield
- ⦿ Cable

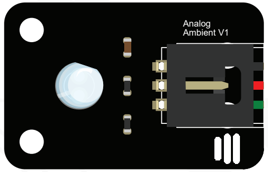
Don't Forget





```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);

  delay(1);
}
```

After uploading the code to your Arduino, open the Serial Monitor. The Serial Monitor displays analog value which varies between 0 and 1023. The values can also be plotted and displayed on the screen while the Arduino board is running, this can be done by opening the Serial Plotter tool from the *Tools* tab in the Arduino IDE.



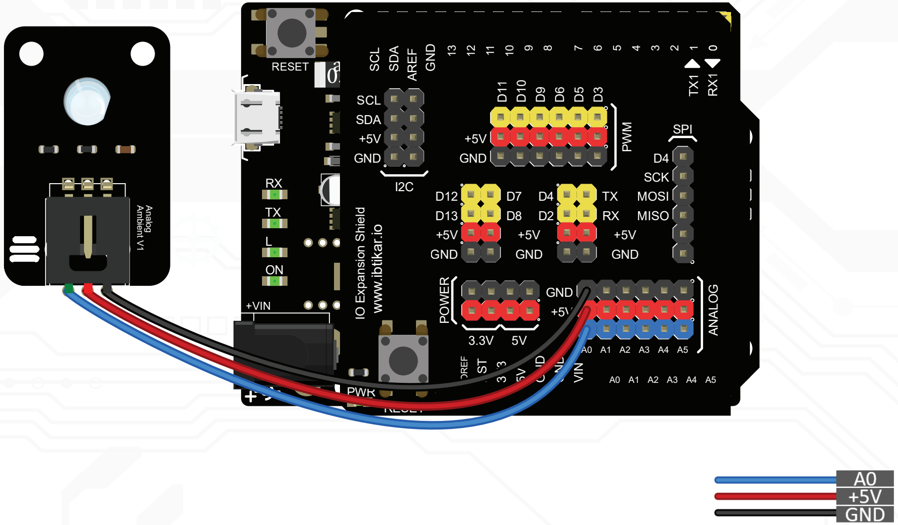
Ambient Light Sensor

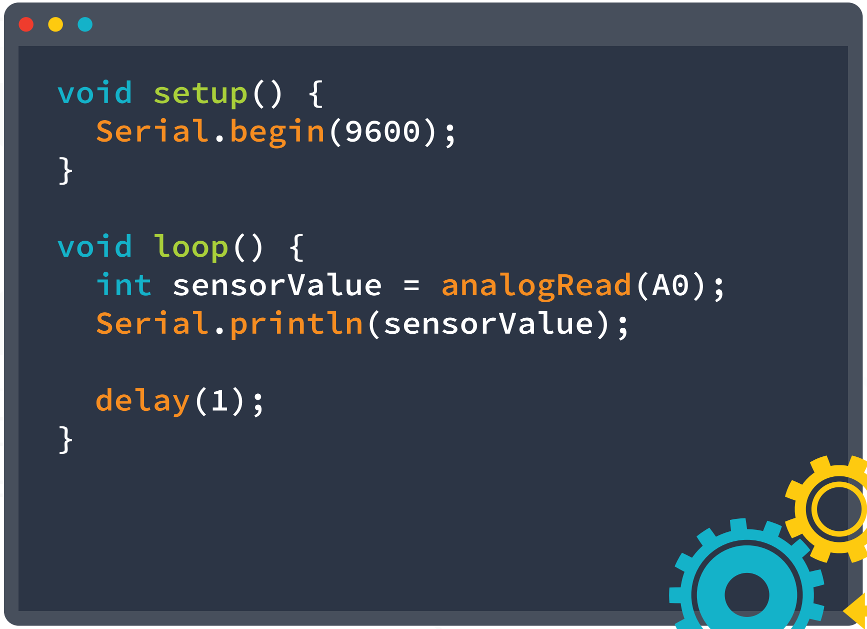
A light sensor is a variable electrical resistor that changes its electrical resistance value according to the amount of light that falls on the sensor.

Note - You also need:

- ⦿ Arduino Leonardo
- ⦿ I/O Expansion Shield
- ⦿ Cable

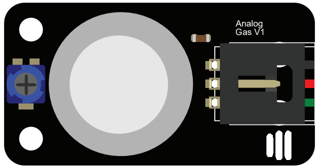
Don't Forget





```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);

  delay(1);
}
```

After uploading the code to your Arduino, open the Serial Monitor. The Serial Monitor displays analog value which varies between 0 and 1023 which represents the ambient light intensity.



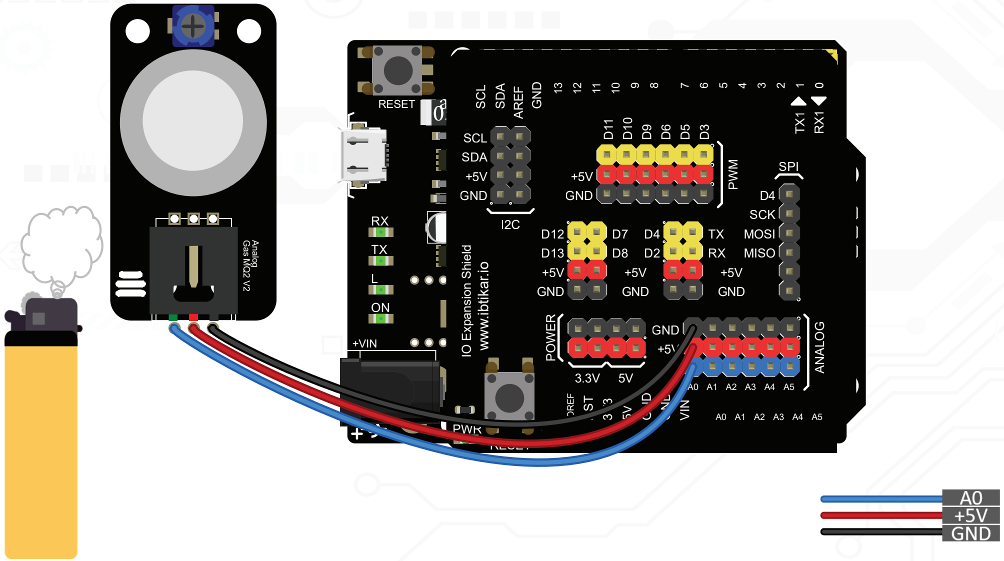
Gas Sensor

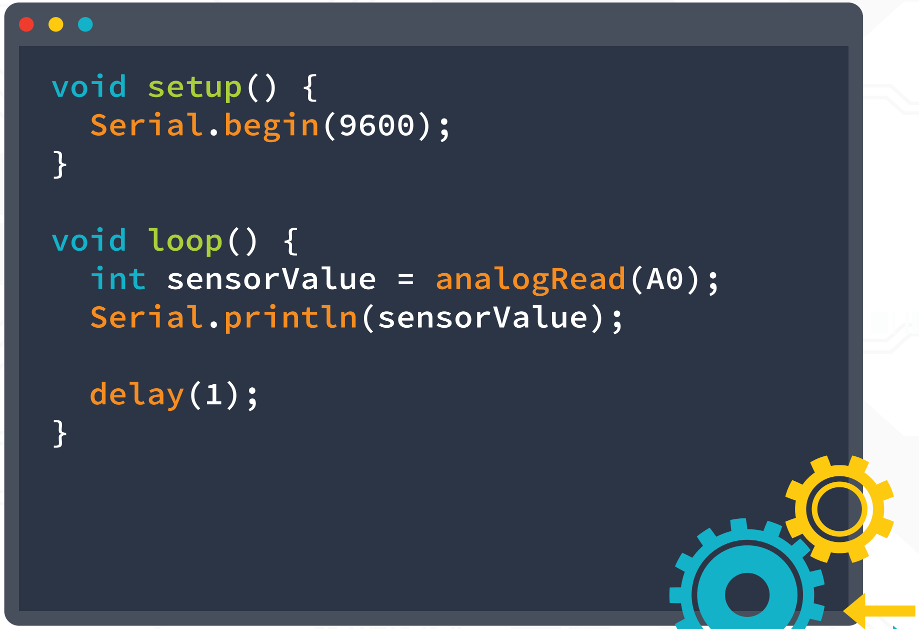
This sensor indicates a gas leakage and sends an electrical signal to the controller to be processed. This gas sensor detects the following gases: butane, propane, methane, hydrogen, and smoke. The sensitivity of this sensor can be tuned using a small knob located on the top of the module.

Note - You also need:

- Arduino Leonardo
- I/O Expansion Shield
- Cable

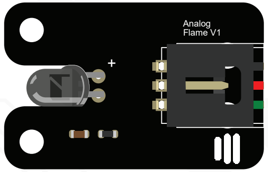
Don't Forget





```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);

  delay(1);
}
```

After uploading the code to your Arduino, open the Serial Monitor. The Serial Monitor displays analog value which varies between 0 and 1023. To test this sensor, you need a gas source like butane or smoke. You can use a lighter to release butane gas near the sensor by pressing the button down.

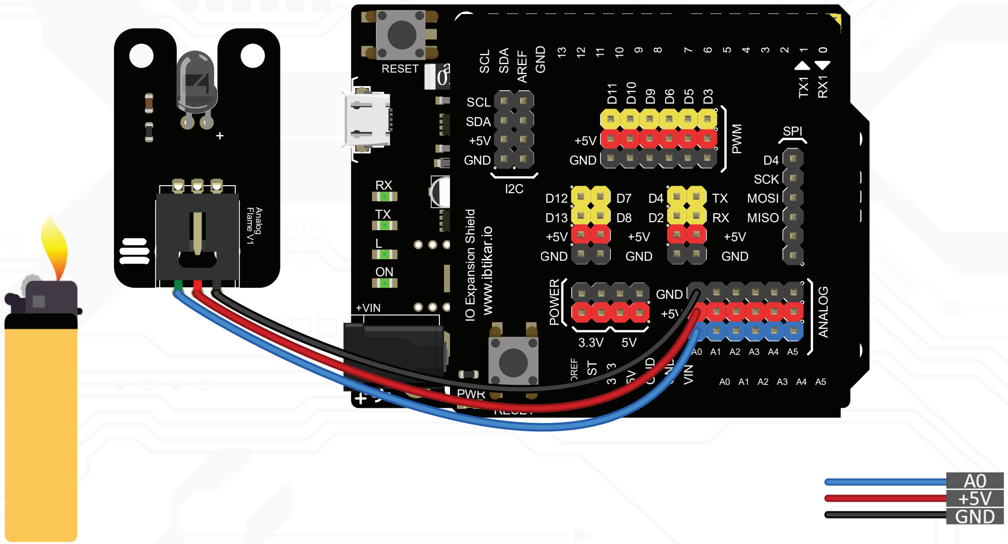


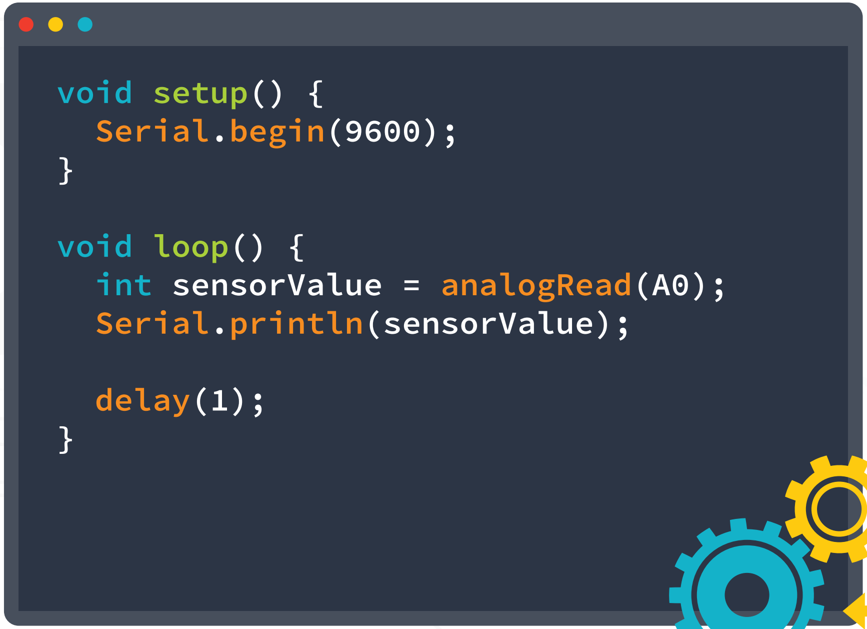
Flame Sensor

This sensor indicates if there is fire or any light within a wavelength from 760 nm ~ 1100 nm.

Note - You also need:
- Arduino Leonardo
- I/O Expansion Shield
- Cable

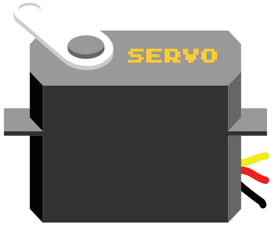
Don't Forget





```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);

  delay(1);
}
```

After uploading the code to your Arduino, open the Serial Monitor. The Serial Monitor displays analog value which varies between 0 and 1023. To test this sensor, you need a fire source. Test using a lighter. Do not let the fire touch the sensor, to avoid damage!



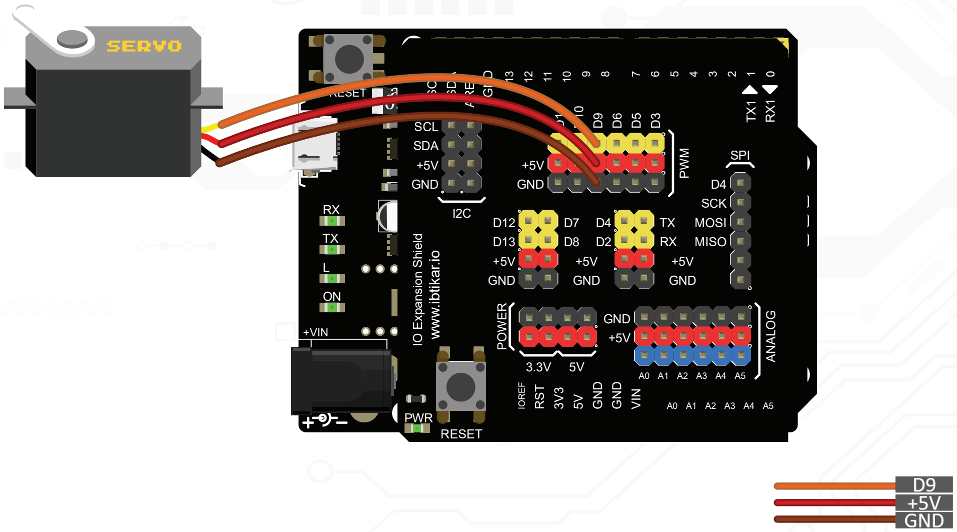
9g Micro Servomotor

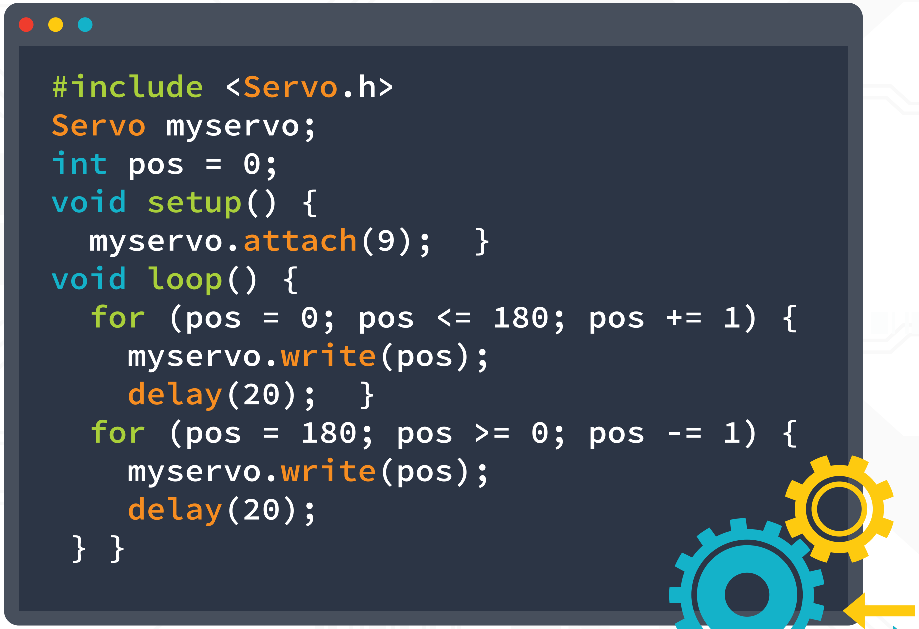
A servomotor is a rotary actuator that allows for exact rotary positioning. It consists of an electric motor with a feedback system to detect the position. The servo motor in the Discovery Kit has a limited range of 0-180 degrees of rotation.

Note - You also need:
- ⦿ Arduino Leonardo
- ⦿ I/O Expansion Shield

Don't Forget

36

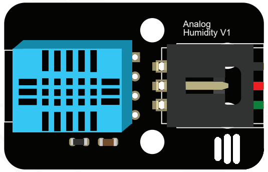




```
#include <Servo.h>
Servo myservo;
int pos = 0;
void setup() {
  myservo.attach(9);  }
void loop() {
  for (pos = 0; pos <= 180; pos += 1) {
    myservo.write(pos);
    delay(20);  }
  for (pos = 180; pos >= 0; pos -= 1) {
    myservo.write(pos);
    delay(20);
 } }
```

After uploading the code to your Arduino, the servo motor will start rotating from 0-180 degrees and then go back to 0 degrees.

Temperature and Humidity Sensor (DHT 11)

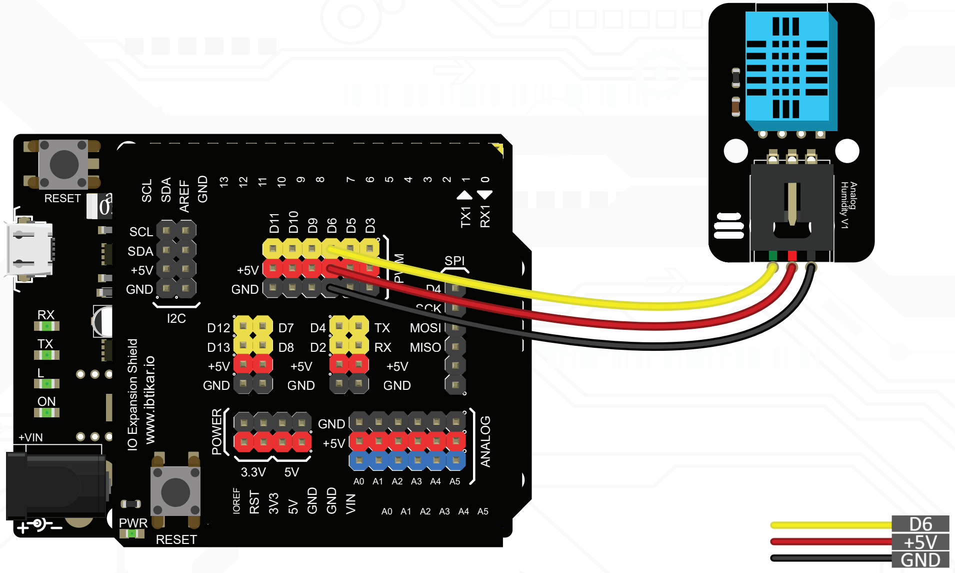


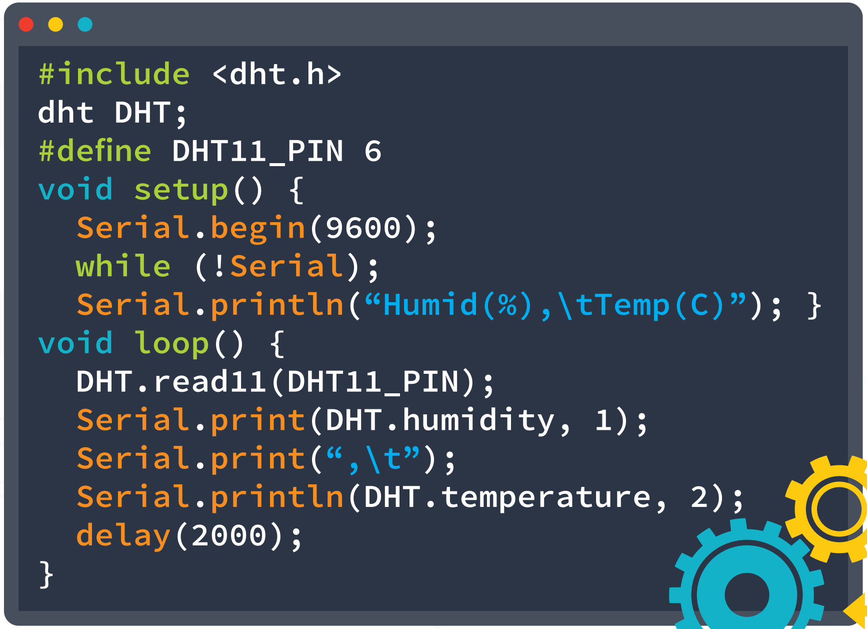
This sensor is used for measuring the temperature and the amount of the humidity in the air. The sensor sends the two values at the same time.

Before using this module, you need to add its library. Without the library, you cannot use this sensor as it is not included by default in the Arduino IDE.

Note - You also need:
- ⦿ Arduino Leonardo
- ⦿ I/O Expansion Shield
- ⦿ Cable

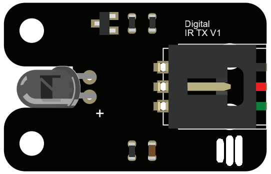
Don't Forget



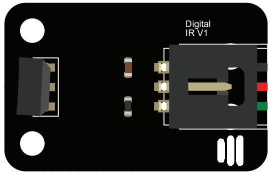


```
#include <dht.h>
dht DHT;
#define DHT11_PIN 6
void setup() {
  Serial.begin(9600);
  while (!Serial);
  Serial.println(“Humid(%),\tTemp(C)”); }
void loop() {
  DHT.read11(DHT11_PIN);
  Serial.print(DHT.humidity, 1);
  Serial.print(“,\t”);
  Serial.println(DHT.temperature, 2);
  delay(2000);
}
```

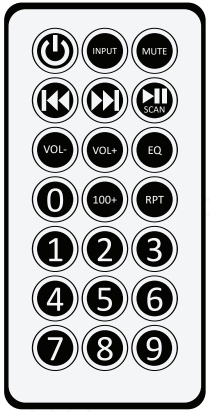
Once you download this code to your Arduino, open the Serial Monitor to read the temperature and humidity values. Blow air on the sensor and notice the change in the values.



IR Transmitter



IR Receiver



Remote Control

Infrared (IR) Kit

IR stands for Infra-Red: It is a light that cannot be seen by the human's eye; it has a wavelength higher than our eyes' ability to detect. This wavelength can be detected by an IR sensor/receiver that decodes the light pulses into digital patterns of 0's and 1's.

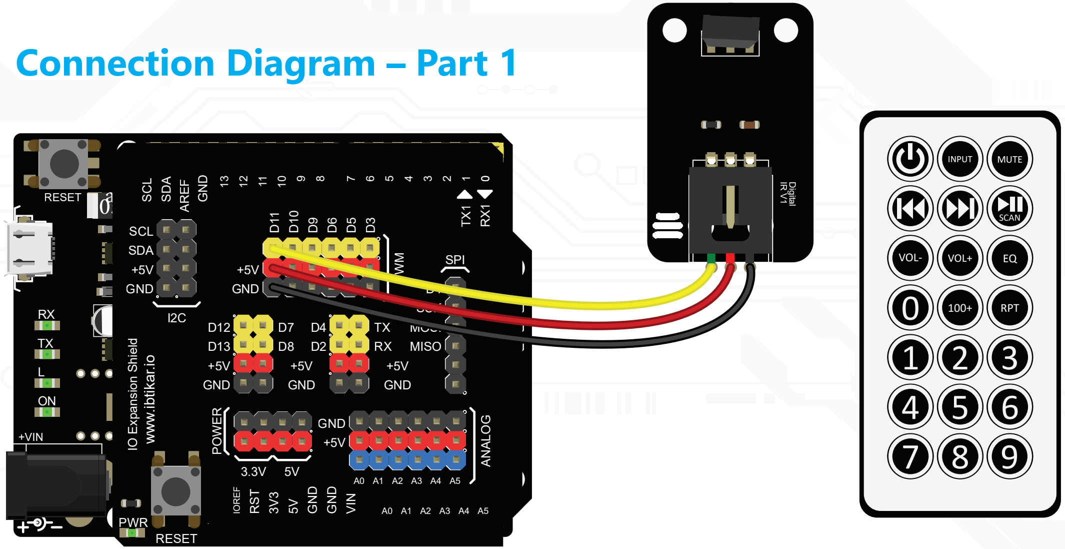
Note - You also need:
- ◉ Arduino Leonardo
- ◉ I/O Expansion Shield
- ◉ Cable

Don't Forget

This kit consists of three main items; the transmitter, the receiver and the remote control. The transmitter has an IR LED light, that emits a series of IR digital pulses as a set of Highs and Lows at a certain frequency.

The remote control has an IR LED as well but with buttons. Each button on the remote control sends a different digital pattern of 0's and 1's that can be generated by an integrated circuit.

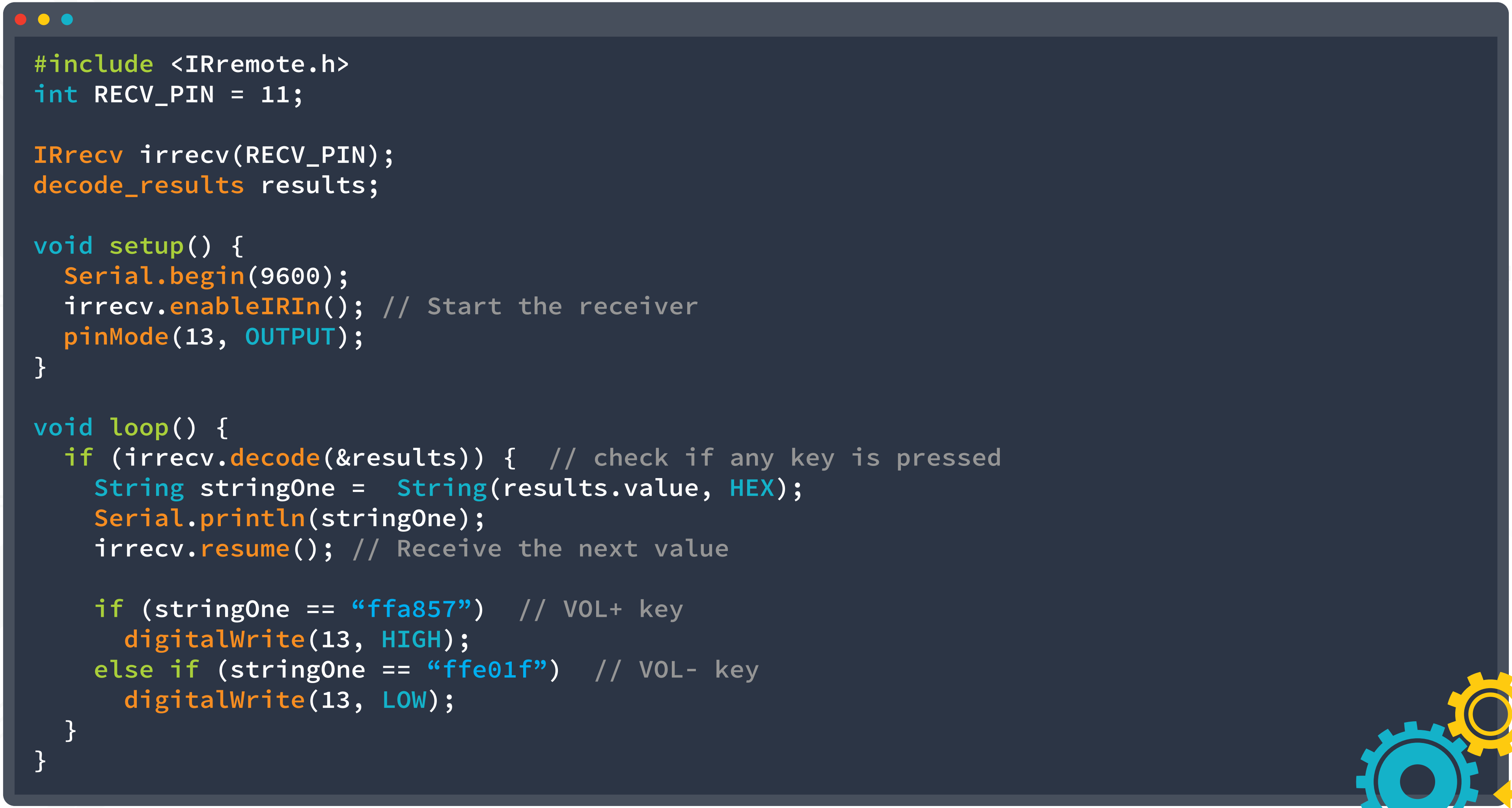
Connection Diagram – Part 1



The IR receiver module decodes the pulses received from the IR transmitter (either the transmitter module itself or the remote control), into a digital series of 0's and 1's which can be used to perform a certain task.

Before using these modules, you need to add its library. Without the library, you cannot use these modules as it is not included by default into the Arduino IDE.

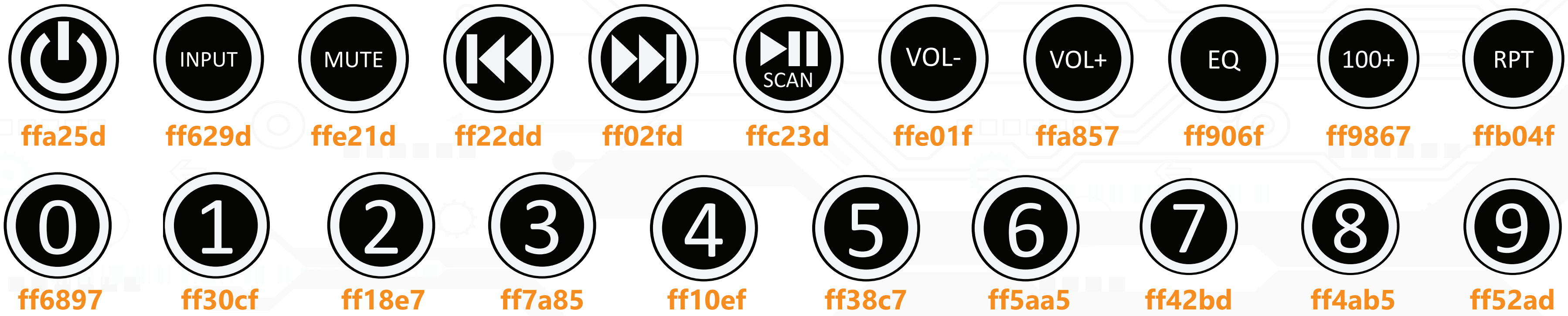
Sample Code – Part 1



```
#include <IRremote.h>
int RECV_PIN = 11;

IRrecv irrecv(RECV_PIN);
decode_results results;

void setup() {
  Serial.begin(9600);
  irrecv.enableIRIn(); // Start the receiver
  pinMode(13, OUTPUT);
}

void loop() {
  if (irrecv.decode(&results)) {  // check if any key is pressed
    String stringOne =  String(results.value, HEX);
    Serial.println(stringOne);
    irrecv.resume(); // Receive the next value

    if (stringOne == “ffa857”)  // VOL+ key
      digitalWrite(13, HIGH);
    else if (stringOne == “ffe01f”)  // VOL- key
      digitalWrite(13, LOW);
  }
}
```

Result – Part 1

After uploading the code to your Arduino, open the Serial Monitor, then press the keys of the remote control while you are pointing it toward the IR receiver. You should see HEX values similar to what you see below. If you are using a different remote control, you will see different values.



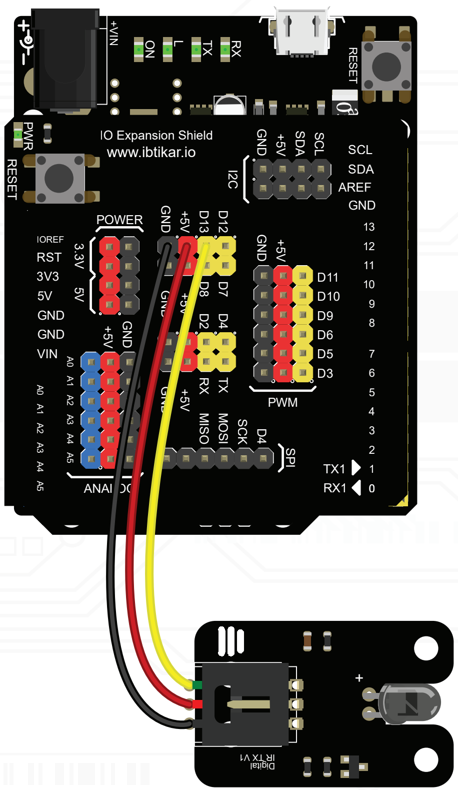
Once you click the **VOL+** button, the built-in LED on the Leonardo board will be ON. Click the **VOL-** button to turn it OFF. Similarly, you can use any other key to do different tasks by checking its HEX value.

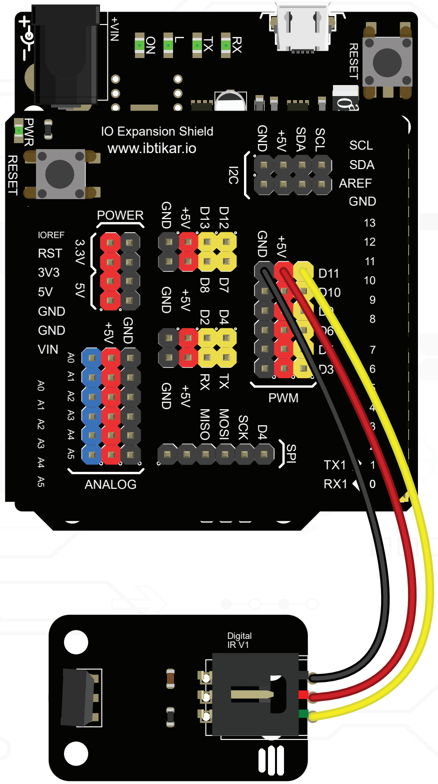
If you are using the remote control for the first time, make sure you remove the small plastic piece that is attached to the battery compartment.

To test the transmitter and receiver, you will need two Arduino boards. The first Arduino has the exact same code from part 1. The second Arduino will have a code that sends the HEX values which turn the LED ON and OFF.

The IR sender module must be connected to pin 13 as it is internally defined in the IR library for Arduino Leonardo. If you are using an Arduino Uno, then you must connect the module to pin 3.

Connection Diagram – Part 2

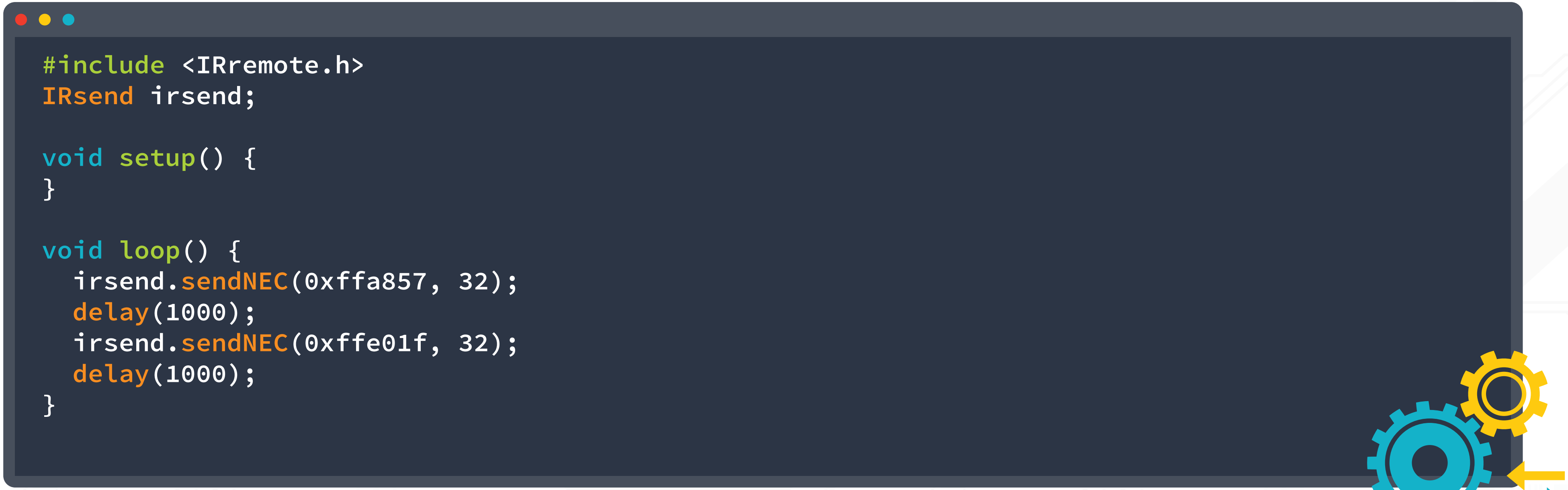




🟨 D13
🟥 +5V
⬛ GND

🟨 D11
🟥 +5V
⬛ GND

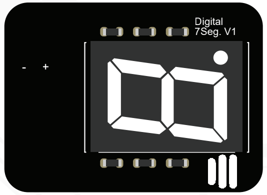
Sample Code – Part 2



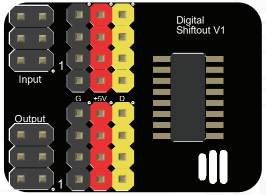
```
#include <IRremote.h>
IRsend irsend;

void setup() {
}

void loop() {
  irsend.sendNEC(0xffa857, 32);
  delay(1000);
  irsend.sendNEC(0xffe01f, 32);
  delay(1000);
}
```

Result – Part 2

After uploading the codes to your Arduino boards, the built-in LED will turn ON and OFF each second. If it is not blinking, make sure the modules are facing each other. Since the same HEX values from the remote control are used, you can use the remote control as well. Cover the IR sender module with your hand. This will make the LED stop blinking. You can now control it manually using the remote control as in part 1. If you click the **VOL+** button, the built-in LED will be ON and if you click the **VOL-** button it will turn it OFF.



7-Segment



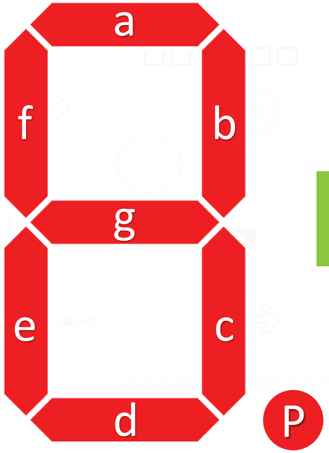
Shift-Out

7-Segment & Shift-Out Modules

The 7-segment display has 8 embedded LEDs combined into one package as one digital digit; 7 for the segments and 1 for the decimal point. To control all LEDs, you need 8 pins from the Arduino. Luckily, you have the shift-out module in your kit.

The shift-out module has an electronic chip that controls a certain number of parallel outputs using binary signals. It can control many devices with less input wires. The module in the Discovery Kit can control up to 8 outputs using 3 inputs only which makes it an ideal solution to control the 7-segment display with 3 wires instead of 8.
The shift-out module outputs a Byte of data (8 Bits), where each bit can either be 0 or 1.

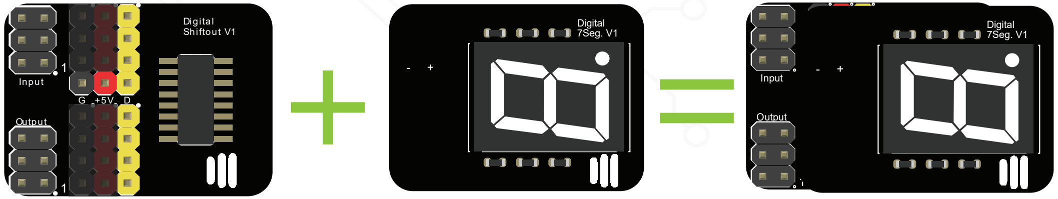
The shift-out module has an input socket and an output socket. The input socket will be connected to Arduino. If it happens that you have other 7-segment and shift-out modules, you can drive them using the same pins by connecting the output socket of the first one to the input socket of the second one.

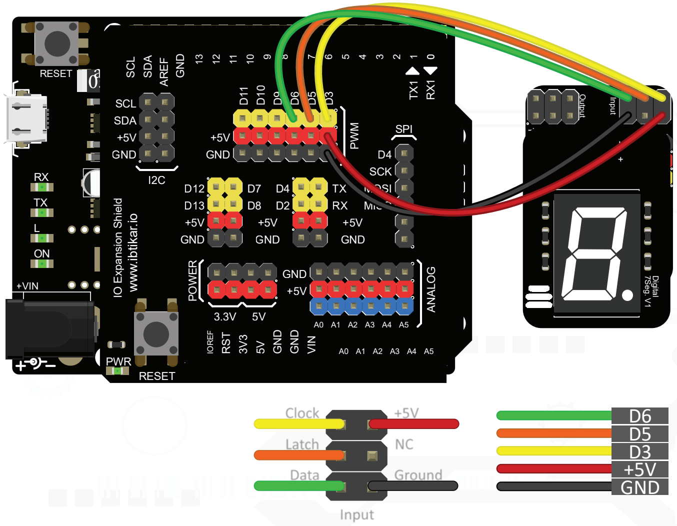


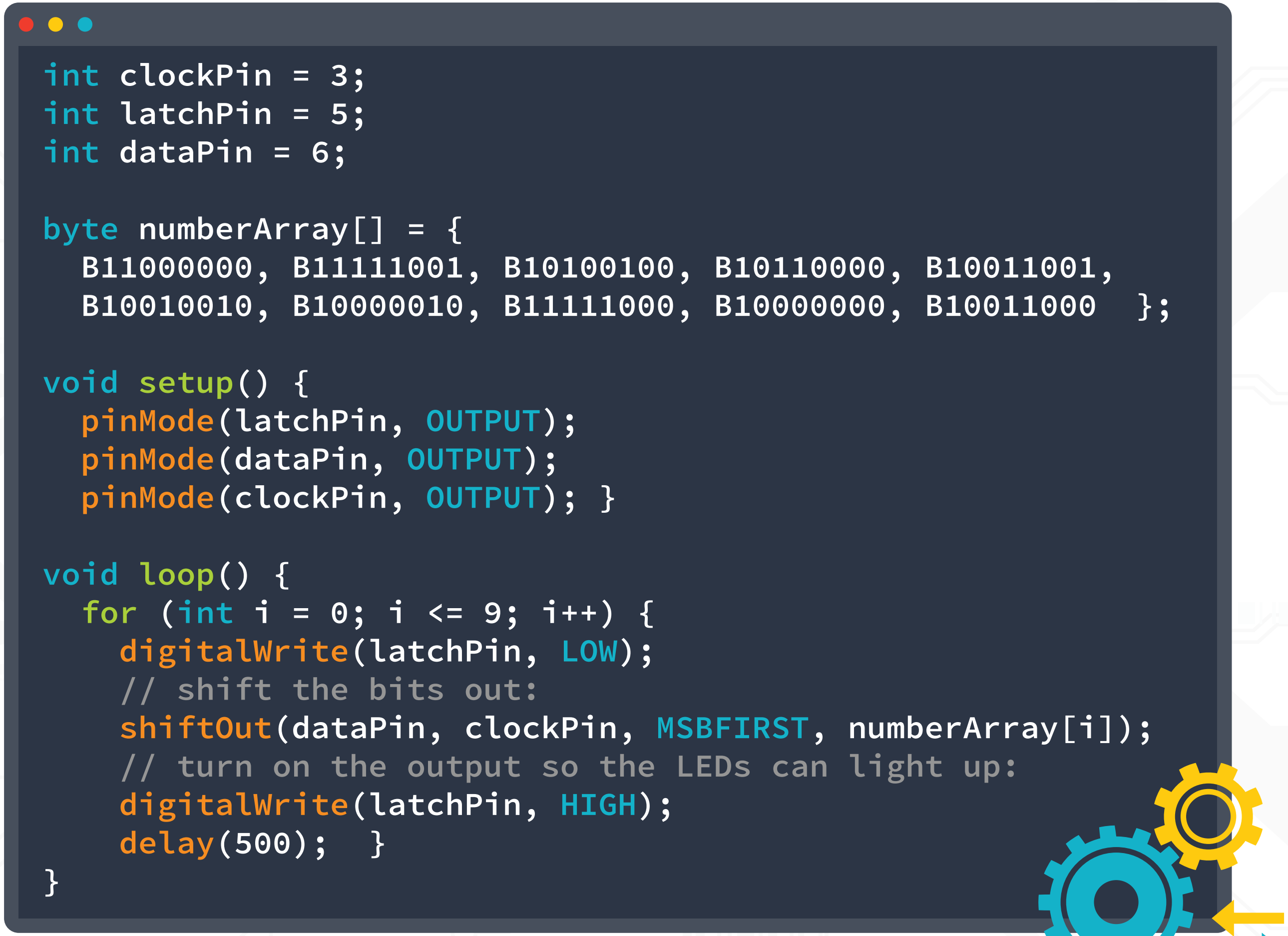
In the 7-segment display, each LED receives its signal from the shift-out module in a binary form 0 or 1. For example, if the displayed number is 1 it means there are two LEDs only which are ON since the number 1 has two segments only (segments b and c).

The 7-segment in the Discovery Kit has a special hardware connection that makes the 8 embedded LEDs work when sending logic 0 to them not 1. This connection is called the Common Anode. This means if number 1 is to be displayed, the binary representation will be **11111001** since the order of the 8 segments which needs to be sent are **Pgfedcba**.

Attach the 7-segment module on top of the shift-out module. The 7-segment module pins must match the coloured pins of the shift-out module which is shown.





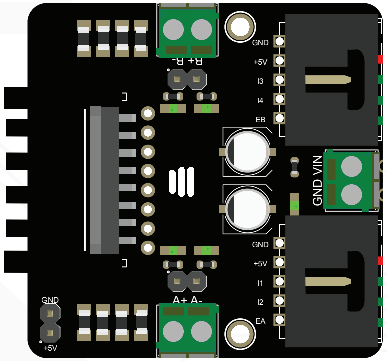

```
int clockPin = 3;
int latchPin = 5;
int dataPin = 6;

byte numberArray[] = {
  B11000000, B11111001, B10100100, B10110000, B10011001,
  B10010010, B10000010, B11111000, B10000000, B10011000  };

void setup() {
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT); }

void loop() {
  for (int i = 0; i <= 9; i++) {
    digitalWrite(latchPin, LOW);
    // shift the bits out:
    shiftOut(dataPin, clockPin, MSBFIRST, numberArray[i]);
    // turn on the output so the LEDs can light up:
    digitalWrite(latchPin, HIGH);
    delay(500);  }
}
```

After uploading the code to your Arduino, the 7-Segment module will display the numbers from 0 to 9 with a delay of half a second. This will repeat forever.

Note - You also need:

- Arduino Leonardo
- I/O Expansion Shield
- Jumper Cable (F/F) x 5

Don't Forget



L298N Dual H-Bridge & Stepper Motor Driver

Motors require high current sources to operate. Unfortunately, this current is much higher than what the Arduino can handle. This motor driver can be used to drive 2 DC motors or 1 stepper motor and control their speed and direction. To control the speed, the Arduino outputs an analog signal.

Note - You also need:

- ⦿ Arduino Leonardo
- ⦿ I/O Expansion Shield
- ⦿ 5-Pin Cable x 2
- ⦿ 9V Battery Clip with Cables
- ⦿ DC Motor x 2
- ⦿ Stepper Motor x 1

Don't Forget

Pulse Width Modulation (PWM) is a technique for generating an analog signal using digital means. PWM creates a square wave (digital signal) by switching between HIGH (ON) and LOW (OFF) states. The Analog signal generated by PWM which ranges between 0 and 255 can be used to control the speed of the motor.

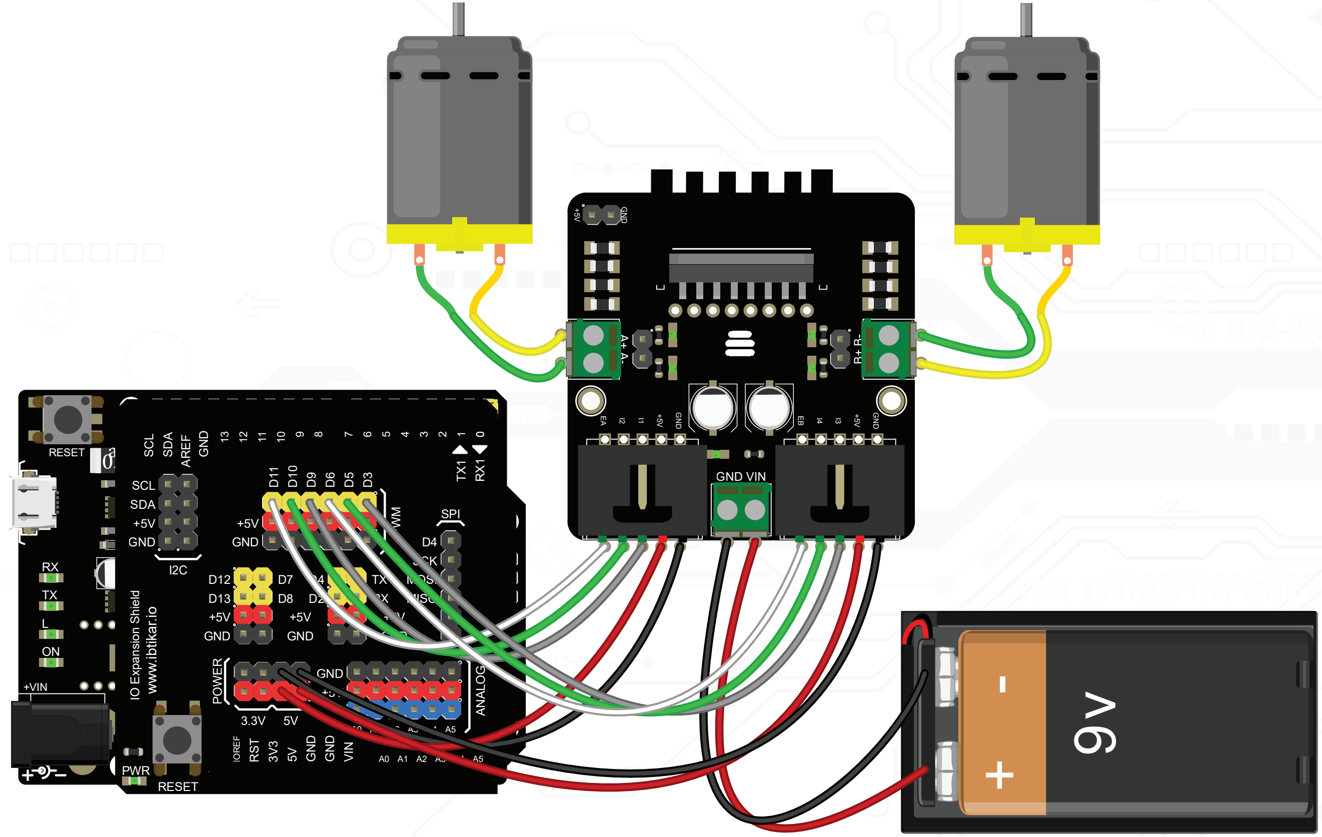
The motor driver needs a dedicated power supply other than the one for the Arduino board. The L298N H-bridge module can be used with motors that have a voltage between 5 and 35V DC. In the Discovery Kit, you have two yellow DC motors and one stepper motor.

DC motors have two leads, positive and negative. When a DC motor is supplied with power, it will start spinning continuously until that power is removed. The DC motor usually runs continuously at high RPM (revolutions per minute) proportional to the supplied voltage. The speed revolutions of the motor shaft can be controlled by controlling the supplied voltage.

Steppers have several phases which if energized with the correct sequence of pulses can rotate in discrete steps. Since the sequence to move one step at a time is known, the position of the motor can be known all the time which means you can control it precisely without the need for rotation sensor.

You must always ensure the motors you use are compatibale with the motor driver module in terms of voltage and current. Read the rated voltage of each motor from its datasheet.

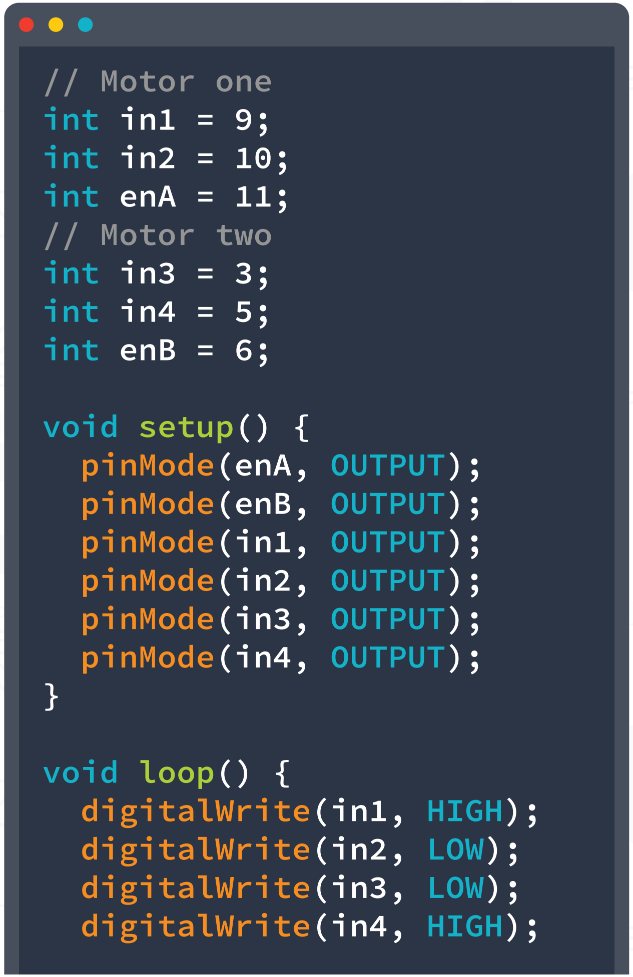
Connection Diagram – Part 1



EA	D11
I2	D10
I1	D9
+5V	+5V
GND	GND

EB	D6
I4	D5
I3	D3
+5V	+5V
GND	GND

Sample Code – Part 1

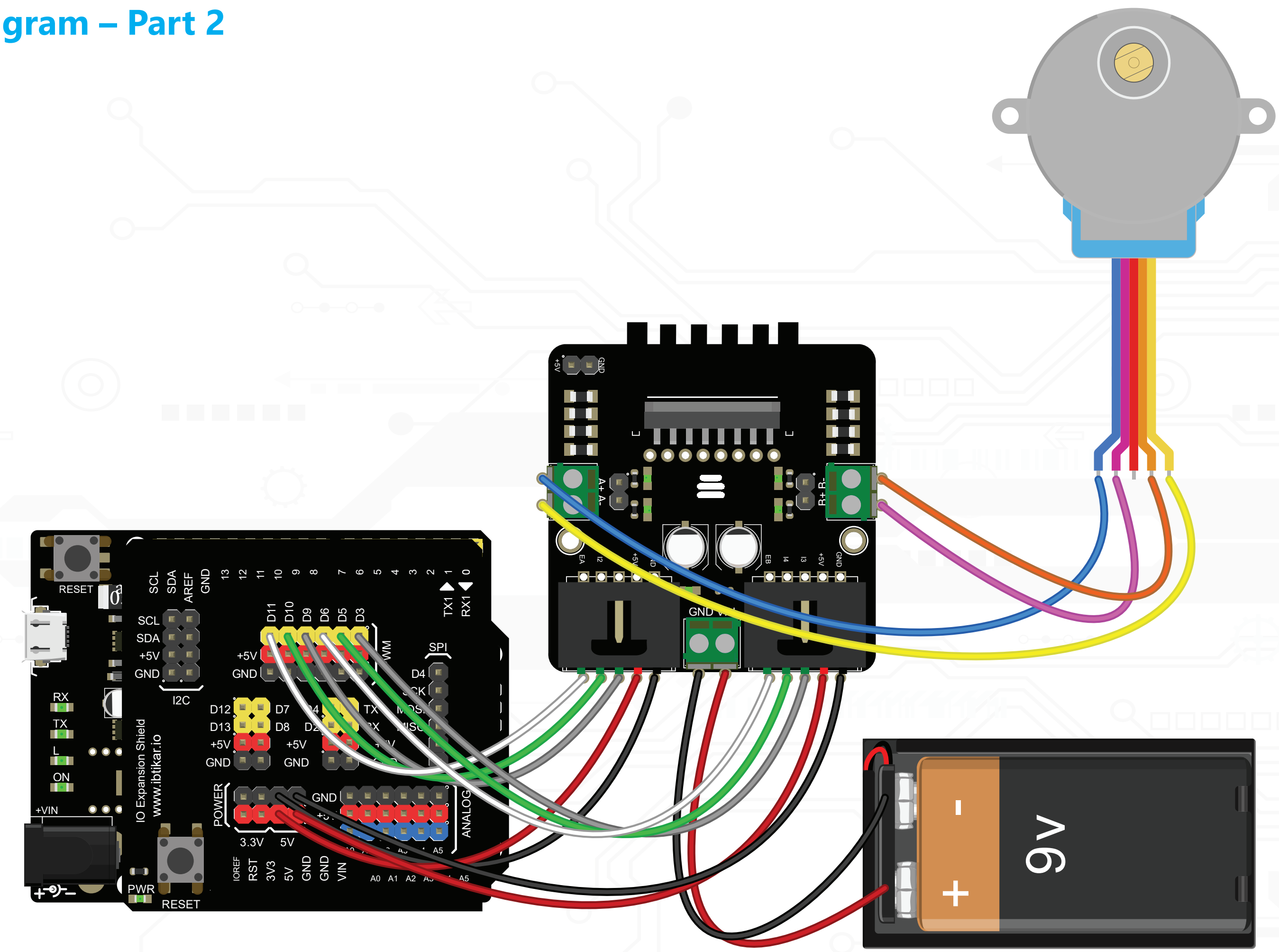


```
// Motor one
int in1 = 9;
int in2 = 10;
int enA = 11;
// Motor two
int in3 = 3;
int in4 = 5;
int enB = 6;

void setup() {
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
}

void loop() {
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  // Accelerate from zero to maximum speed
  for (int i = 0; i < 256; i++) {
    analogWrite(enA, i);
    analogWrite(enB, i);
    delay(20);
  }
  // Decelerate from maximum speed to zero
  for (int i = 255; i > 0; i--) {
    analogWrite(enA, i);
    analogWrite(enB, i);
    delay(20);
  }

  // Turn off the motors
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  delay(1000);
}
```

Result – Part 1

Once you download this code to your Arduino, the 2 DC motors will start moving opposite to each other. Both will accelerate from zero to the maximum speed, then decelerate from the maximum speed to zero and finally stop for 1 second.

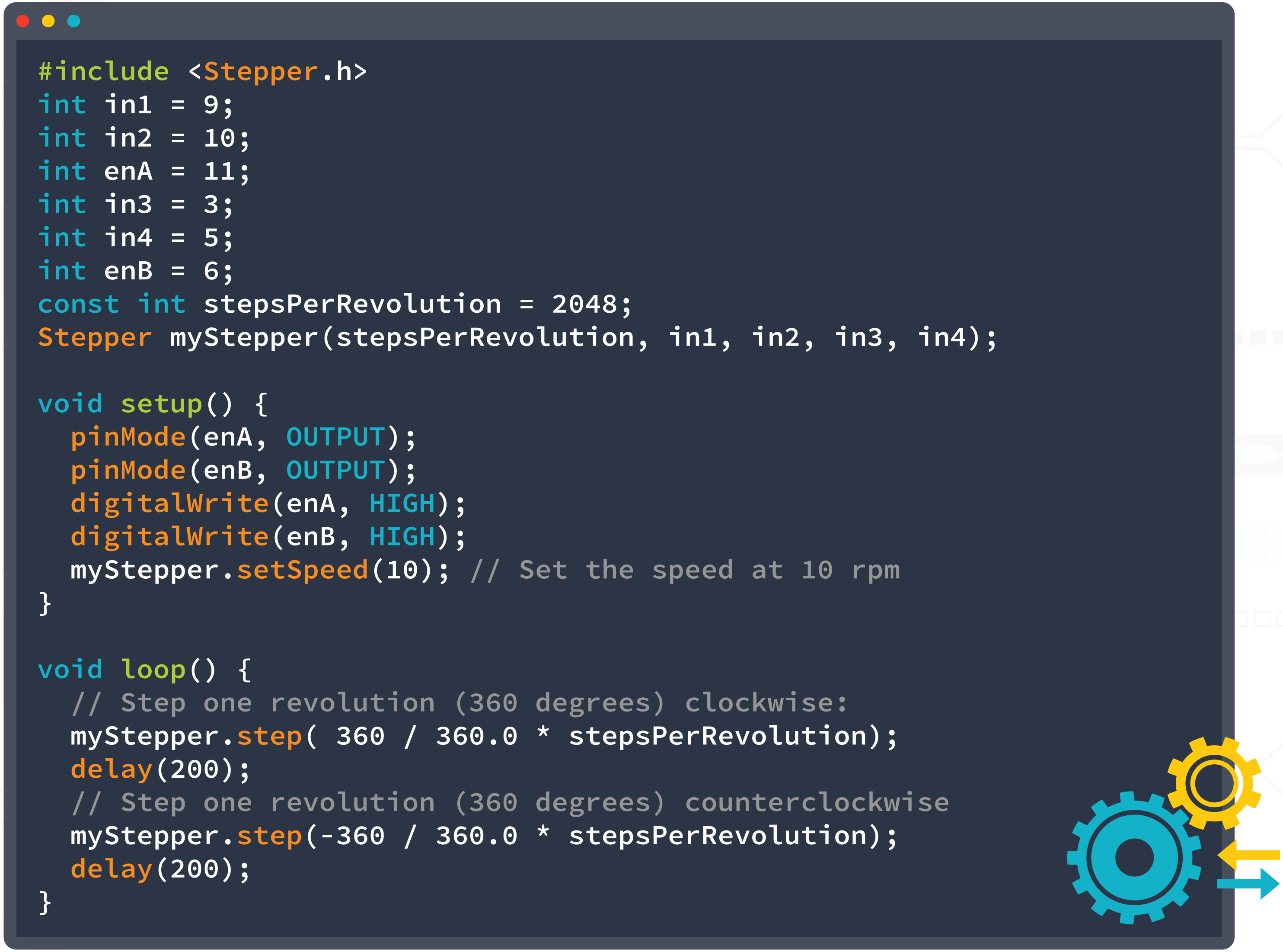
Connection Diagram – Part 2



EA		D11
I2		D10
I1		D9
+5V		+5V
GND		GND

EB		D6
I4		D5
I3		D3
+5V		+5V
GND		GND

Sample Code – Part 2



```
#include <Stepper.h>
int in1 = 9;
int in2 = 10;
int enA = 11;
int in3 = 3;
int in4 = 5;
int enB = 6;
const int stepsPerRevolution = 2048;
Stepper myStepper(stepsPerRevolution, in1, in2, in3, in4);

void setup() {
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  digitalWrite(enA, HIGH);
  digitalWrite(enB, HIGH);
  myStepper.setSpeed(10); // Set the speed at 10 rpm
}

void loop() {
  // Step one revolution (360 degrees) clockwise:
  myStepper.step( 360 / 360.0 * stepsPerRevolution);
  delay(200);
  // Step one revolution (360 degrees) counterclockwise
  myStepper.step(-360 / 360.0 * stepsPerRevolution);
  delay(200);
}
```

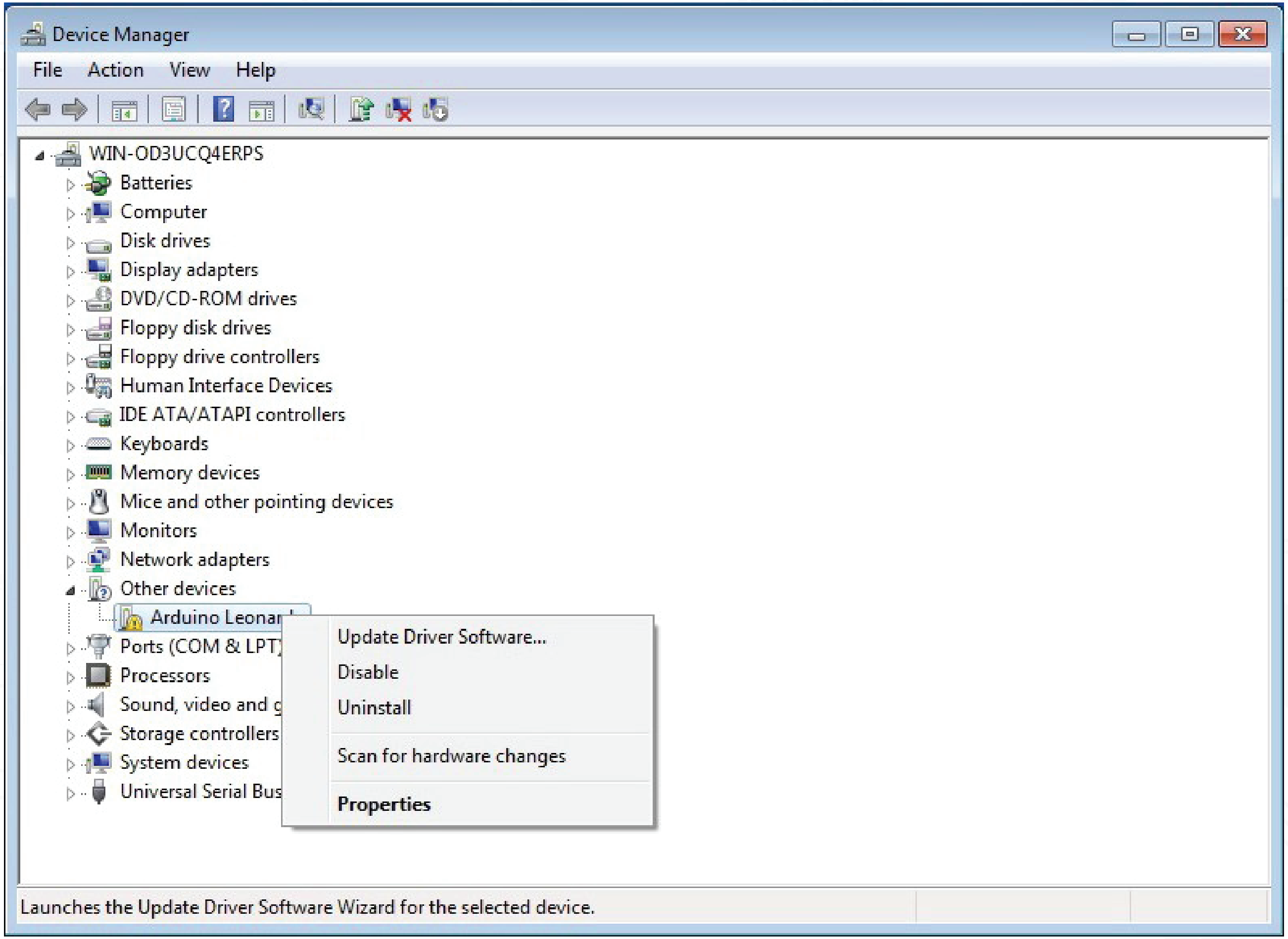
Result – Part 2

Once you download this code to your Arduino, the stepper motor will start rotating one full rotation clockwise and counter-clockwise.

Appendix: Arduino Driver Installation

To install the Arduino board driver to the computer, follow these steps:

- Click on the *Start Menu* and then write *Control Panel*.
- Open it and then navigate to *System* and *Security*.
- Click on *System* and on the left panel click on *Device Manager*.
- A window will pop up.
- Under *Other Devices*, you will see an icon with a yellow hazard sign named *Arduino Leonardo*.
- Right click on the icon named *Arduino Leonardo*.
- Then click *Update Driver Software*.
- Browse to the driver folder where the Arduino IDE is installed and click *next*.
- Your computer will install a proper *Driver* for your board.



Launches the Update Driver Software Wizard for the selected device.