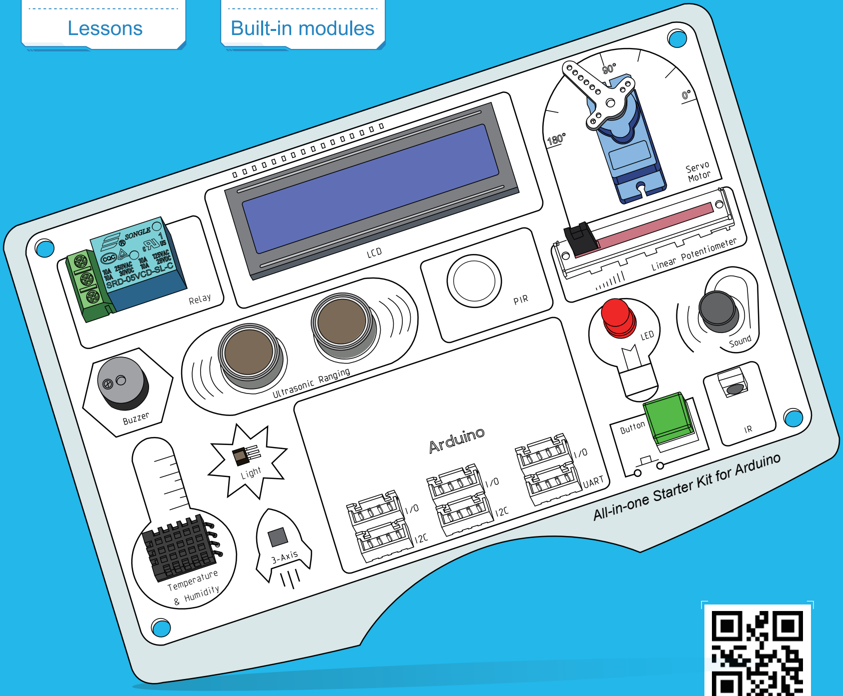
ELECROW

All-in-one Starter Kit for Arduino User Manual

- STEAM Education
- Open-source Hardware

20+
Lessons

15
Built-in modules



www.elecrow.com

Table of Contents

Introduction

Welcome to read the user manual of All-in-one Starter Kit for Arduino. Let's start learning about the All-in-one Starter Kit for Arduino development board and how to use the sensors now!

Don't worry, this development board comes with 21 easy-to-follow, fun and inspiring lessons that take you step by step through your knowledge. Here, you will be taken to familiarize yourself with the electronic modules, exercise your logical thinking, enhance your creative ideas, and program the functions of these electronic modules.

From the very beginning to understand the installation of Arduino software, then introduce the Arduino development board and various sensors, then learn the programming functions of these sensors and the programming language used, and finally how to realize the specific applications of these sensors. Each step of the explanation is very clear, even if you are a beginner, you can quickly master Arduino programming.

The all-in-one Starter Kit for Arduino product offers 15 electronic modules, each module has its own unique features and functionality, designed for beginners and ideal for getting started. In short, by learning this development board, you will learn the basics and principles of sensors, understand important concepts such as digital and analog signals, analog-to-digital conversion, programming logic, and learn how to use some of the complex electronic modules. Most importantly, you will start learning Arduino programming, which will help improve your logical thinking skills.

In the programming section, we will use Arduino software for programming.Arduino platform is one of the most popular open source platforms and easy to use. It contains global hardware resources including hardware (various models of Arduino boards) and software (Arduino IDE), which is one of the best choices for programming learning.

List of Sensors	
• Temperature& Humidity Sensor	x1
• Button	x1
• Ultrasonic Ranging Sensor	x1
• Light Sensor	x1
• Linear Potentiometer	x1
• LED	x1
• Buzzer	x1
• LCD	x1
• Infrared Remote Receiver	x1
• Relay	x1
• Servo motor	x1
• Sound Sensor	x1
• 3 Axis Accelerometer	x1
• PIR	x1
• Moisture Sensor	x1

Getting Started

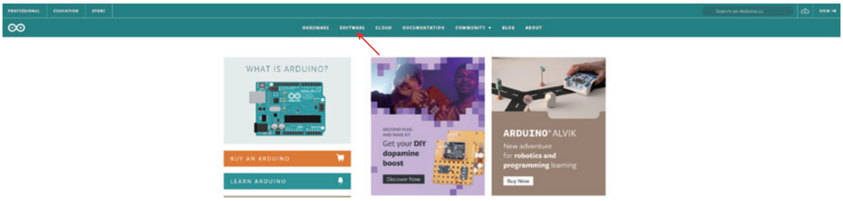
Installing Arduino IDE

Download Arduino in Windows system

STEP 1:

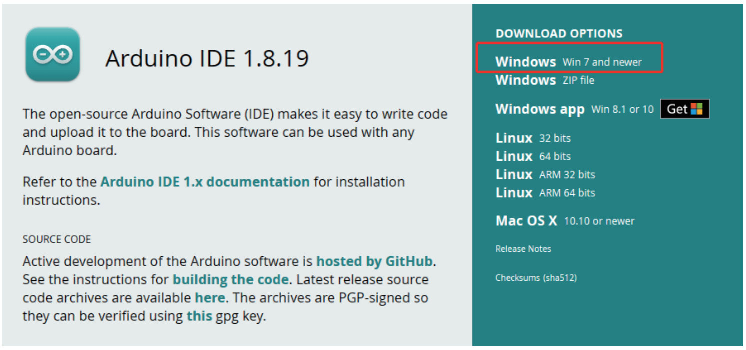
Login to Arduino official website, download Arduino, click SOFTWARE.

Arduino official website: https://www.arduino.cc/



STEP 2:

Select your computer's corresponding system to download, such as Window system.



Arduino IDE 1.8.19

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the **Arduino IDE 1.x documentation** for installation instructions.

SOURCE CODE

Active development of the Arduino software is **hosted by GitHub**. See the instructions for **building the code**. Latest release source code archives are available **here**. The archives are PGP-signed so they can be verified using **this** gpg key.

DOWNLOAD OPTIONS

Windows Win 7 and newer
Windows ZIP file

Windows app Win 8.1 or 10 Get

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

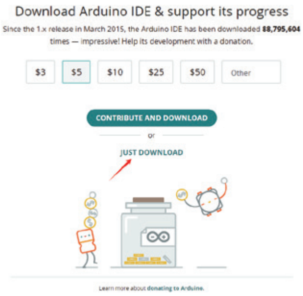
Mac OS X 10.10 or newer

Release Notes

Checksums (sha512)

STEP 3:

Click JUST DOWNLOAD and select the save location to start the download.



Download Arduino IDE & support its progress

Since the 1.x release in March 2015, the Arduino IDE has been downloaded **88,795,604** times — impressive! Help its development with a donation.

$3 $5 $10 $25 $50 Other

CONTRIBUTE AND DOWNLOAD

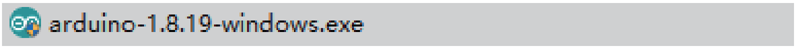
or

JUST DOWNLOAD

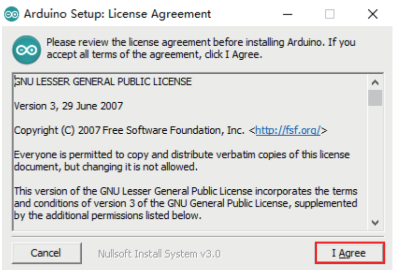
Learn more about **donating to Arduino.**

STEP 4:

1.When installing Arduino, please locate the executable file with the .exe extension within the folder where you previously saved, which is the Arduino installation package.


arduino-1.8.19-windows.exe

2. After double-clicking the installation package, this page will appear.
Click on "I Agree".


Arduino Setup: License Agreement

Please review the license agreement before installing Arduino. If you accept all terms of the agreement, click I Agree.

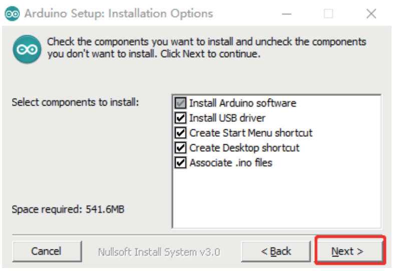
GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.


Arduino Setup: Installation Options

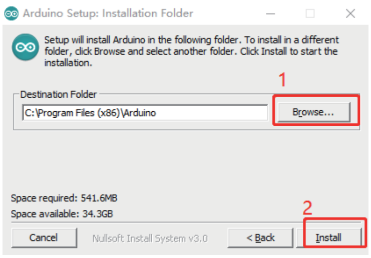
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:
- [x] Install Arduino software
- [x] Install USB driver
- [x] Create Start Menu shortcut
- [x] Create Desktop shortcut
- [x] Associate .ino files

Space required: 541.6MB

3.Check all options by default and click Next.

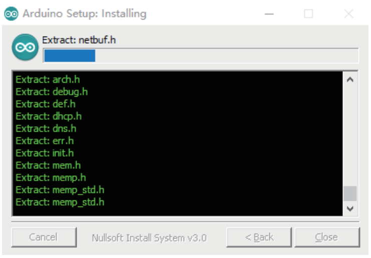
4.Click on 'Browse' to select the installation location, it is recommended to install it on any drive other than the C: drive. Then click 'Install'.


Arduino Setup: Installation Folder

Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files (x86)\Arduino

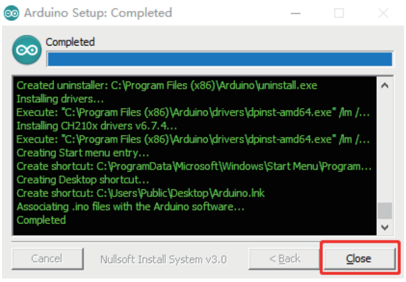
Space required: 541.6MB
Space available: 34.3GB


Arduino Setup: Installing

Extract: netbuf.h

```
Extract: arch.h
Extract: debug.h
Extract: def.h
Extract: dhcp.h
Extract: dns.h
Extract: err.h
Extract: init.h
Extract: mem.h
Extract: memp.h
Extract: memp_std.h
Extract: memp_std.h
```

5.Installation Complete


Arduino Setup: Completed

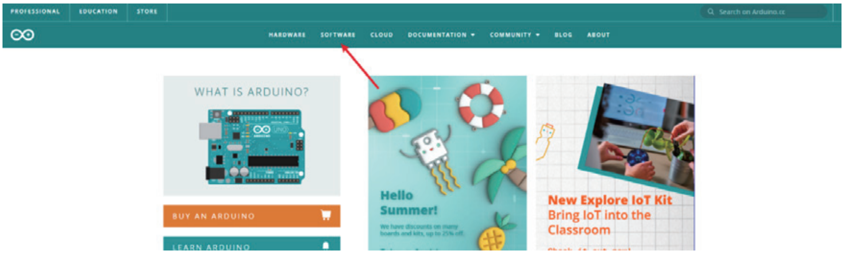
Completed

```
Created uninstaller: C:\Program Files (x86)\Arduino\uninstall.exe
Installing drivers...
Execute: "C:\Program Files (x86)\Arduino\drivers\dpinst-amd64.exe" /lm /...
Installing CH210x drivers v6.7.4...
Execute: "C:\Program Files (x86)\Arduino\drivers\dpinst-amd64.exe" /lm /...
Creating Start menu entry...
Create shortcut: C:\ProgramData\Microsoft\Windows\Start Menu\Program...
Creating Desktop shortcut...
Create shortcut: C:\Users\Public\Desktop\Arduino.lnk
Associating .ino files with the Arduino software...
Completed
```

Download Arduino in Linux system

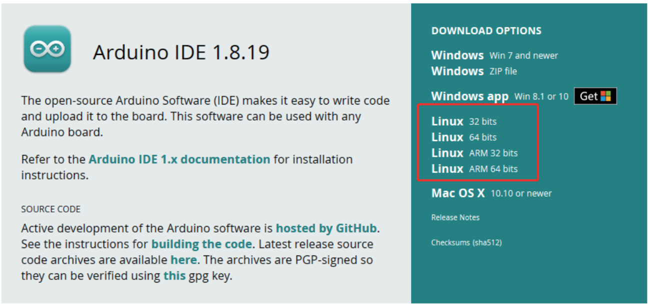
STEP 1:

Login to Arduino official website, download Arduino, click ‘SOFTWARE’.
Arduino official website:https://www.arduino.cc/



STEP 2:

Select the version of Linux that corresponds to your computer for download.


Arduino IDE 1.8.19

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the **Arduino IDE 1.x documentation** for installation instructions.

SOURCE CODE

Active development of the Arduino software is **hosted by GitHub**. See the instructions for **building the code**. Latest release source code archives are available **here**. The archives are PGP-signed so they can be verified using **this** gpg key.

DOWNLOAD OPTIONS

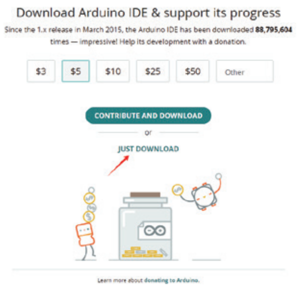
Windows Win 7 and newer
Windows ZIP file
Windows app Win 8.1 or 10 Get
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits
Mac OS X 10.10 or newer

Release Notes

Checksums (sha512)

STEP 3:

After clicking 'JUST DOWNLOAD', select the save location to begin the download process.



Download Arduino IDE & support its progress

Since the 1.x release in March 2015, the Arduino IDE has been downloaded **88,795,604** times — impressive! Help its development with a donation.

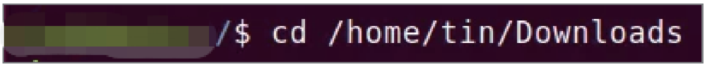
Learn more about donating to Arduino.

STEP 4:

Install Arduino IDE

You are now ready to install the Arduino IDE. Please navigate to the download folder where you saved the Arduino archive (assuming the directory is: /home/username/Downloads). You will need to extract the downloaded Arduino archive. You can do this by launching a terminal using the Ctrl+Alt+T shortcut and then running the following command to change to the Downloads folder:

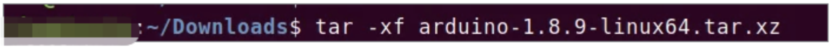
cd /home/username/Downloads



```
[redacted]/$ cd /home/tin/Downloads
```

To extract the archive folder, please run the following command in the terminal:

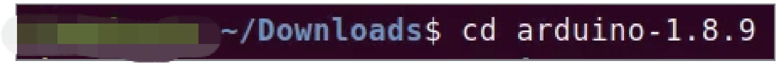
tar xf [Compressed-filename]



```
[redacted]:~/Downloads$ tar -xf arduino-1.8.9-linux64.tar.xz
```

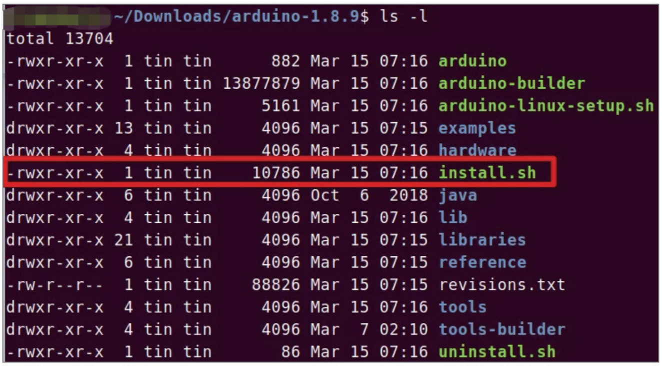
Then, in the terminal, run the following command to navigate to the newly extracted Arduino folder:

cd [Uncompressed-foldername]



```
[redacted]~/Downloads$ cd arduino-1.8.9
```

You can try executing the `ls -l` command in the terminal to view the file list in the Arduino folder, and then locate the install.sh installation script that we are about to install.

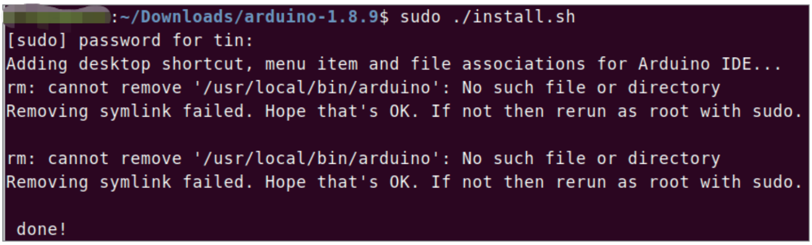


```
[redacted]~/Downloads/arduino-1.8.9$ ls -l
total 13704
-rwxr-xr-x  1 tin tin      882 Mar 15 07:16 arduino
-rwxr-xr-x  1 tin tin 13877879 Mar 15 07:16 arduino-builder
-rwxr-xr-x  1 tin tin     5161 Mar 15 07:16 arduino-linux-setup.sh
drwxr-xr-x 13 tin tin     4096 Mar 15 07:15 examples
drwxr-xr-x  4 tin tin     4096 Mar 15 07:16 hardware
-rwxr-xr-x  1 tin tin    10786 Mar 15 07:16 install.sh
drwxr-xr-x  6 tin tin     4096 Oct  6  2018 java
drwxr-xr-x  4 tin tin     4096 Mar 15 07:16 lib
drwxr-xr-x 21 tin tin     4096 Mar 15 07:15 libraries
drwxr-xr-x  6 tin tin     4096 Mar 15 07:15 reference
-rw-r--r--  1 tin tin    88826 Mar 15 07:15 revisions.txt
drwxr-xr-x  4 tin tin     4096 Mar 15 07:16 tools
drwxr-xr-x  4 tin tin     4096 Mar  7 02:10 tools-builder
-rwxr-xr-x  1 tin tin       86 Mar 15 07:16 uninstall.sh
```

sudo ./install.sh

We are now ready to install Arduino. Execute the following command in the terminal with sudo privileges to install Arduino:

Please wait for a period of time until the installation is complete.

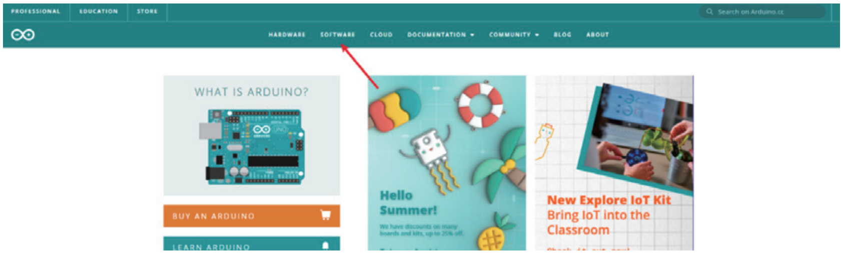


```
[redacted]:~/Downloads/arduino-1.8.9$ sudo ./install.sh
[sudo] password for tin:
Adding desktop shortcut, menu item and file associations for Arduino IDE...
rm: cannot remove '/usr/local/bin/arduino': No such file or directory
Removing symlink failed. Hope that's OK. If not then rerun as root with sudo.

rm: cannot remove '/usr/local/bin/arduino': No such file or directory
Removing symlink failed. Hope that's OK. If not then rerun as root with sudo.

 done!
```

Download Arduino in MacOS system

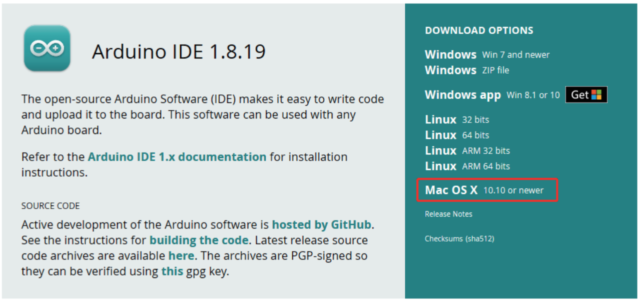
STEP 1:

Login to Arduino official website, download Arduino, click 'SOFTWARE'.
Arduino official website: https://www.arduino.cc/



STEP 2:

Select the Mac version of your computer to download.



Arduino IDE 1.8.19

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the Arduino IDE 1.x documentation for installation instructions.

SOURCE CODE

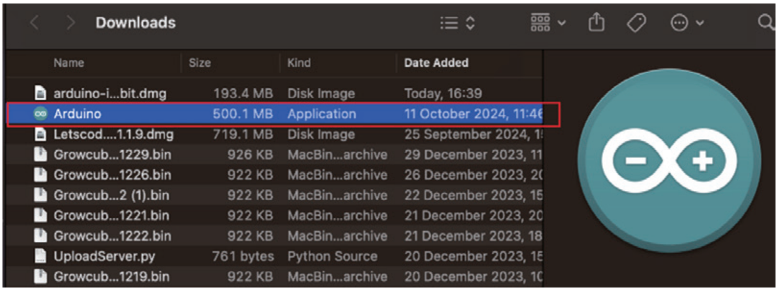
Active development of the Arduino software is hosted by GitHub. See the instructions for building the code. Latest release source code archives are available here. The archives are PGP-signed so they can be verified using this gpg key.

DOWNLOAD OPTIONS

Windows Win 7 and newer
Windows ZIP file
Windows app Win 8.1 or 10 Get
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits
Mac OS X 10.10 or newer
Release Notes
Checksums (sha512)

STEP 3:

After the download is successful, double-click to install.



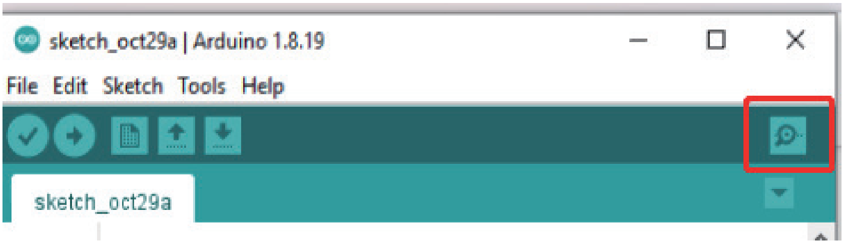
Downloads

Name	Size	Kind	Date Added
arduino-i...bit.dmg	193.4 MB	Disk Image	Today, 16:39
Arduino	500.1 MB	Application	11 October 2024, 11:46
Letscod....1.1.9.dmg	719.1 MB	Disk Image	25 September 2024, 1
Growcub...1229.bin	926 KB	MacBin...archive	29 December 2023, 11
Growcub...1226.bin	922 KB	MacBin...archive	26 December 2023, 2
Growcub...2 (1).bin	922 KB	MacBin...archive	22 December 2023, 15
Growcub...1221.bin	922 KB	MacBin...archive	21 December 2023, 20
Growcub...1222.bin	922 KB	MacBin...archive	21 December 2023, 18
UploadServer.py	761 bytes	Python Source	20 December 2023, 15
Growcub...1219.bin	922 KB	MacBin...archive	20 December 2023, 1

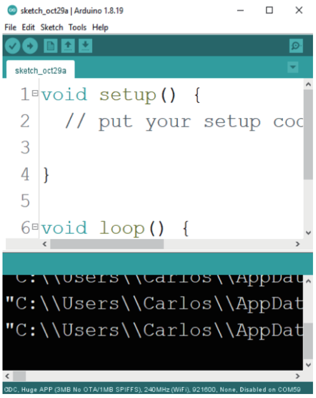
How to Use the Serial Monitor Tool

1.Open the Arduino IDE.

2.Click on the "Serial Monitor" button. Note that you need to have a serial device connected to open it; otherwise, the Serial Monitor tool will not be accessible.



sketch_oct29a | Arduino 1.8.19

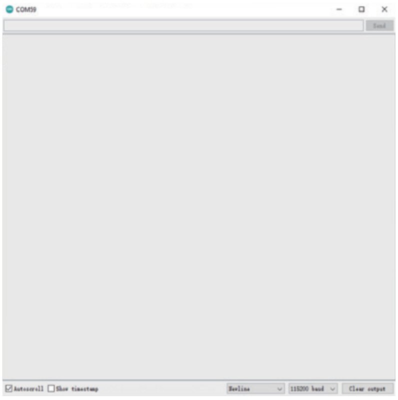

sketch_oct29a | Arduino 1.8.19

File Edit Sketch Tools Help

```
void setup() {
  // put your setup coc

}

void loop() {
```

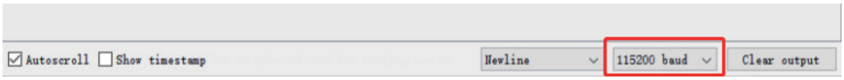
```
"C:\\Users\\Carlos\\AppDat
"C:\\Users\\Carlos\\AppDat
"C:\\Users\\Carlos\\AppDat
```

DC, Huge APP (3MB No OTA/1MB SPIFFS), 240MHz (WiFi), 921600, None, Disabled on COM59


COM59

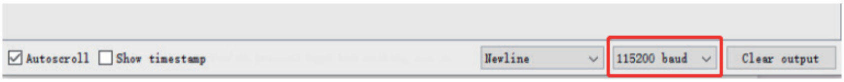
Autoscroll · Show timestamp · Newline · 115200 baud · Clear output

Setting the baud rate.


☑ Autoscroll ☐ Show timestamp · Newline · 115200 baud · Clear output

The baud rate setting should be the same as that initialized by the code to communicate properly.

```
Serial.begin(115200);
```

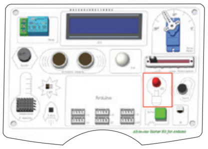

☑ Autoscroll ☐ Show timestamp · Newline · 115200 baud · Clear output

Lessons

Lesson 1 - LED Control

Introduction

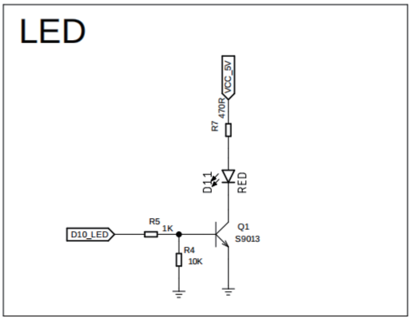
In this lesson, we will explore how to control LED lights. By programming the LEDs, we are able to make them blink alternately at certain intervals.



Hardware required

- All_in_one Starter Kit for Arduino x1
- USB Cable x1

Hardware schematic



What will you see

We will see that the LED will light up for a second and go out for a second, and the process will continue to cycle. If the LED does not work as expected, make sure the program is running correctly.

Complete Code

```
int LedPin = 10;
void setup() {
  // put your setup code here, to run once:
  pinMode(LedPin, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(LedPin, HIGH);
  delay(1000);
  digitalWrite(LedPin, LOW);
  delay(1000);
}
```

Main Code Explanation

int LedPin = 10;

Variables serve as placeholders for values that may change within the code. It is necessary to "declare" a variable before using it. In this case, we have declared a variable of type `LedPin` as an `Int` (integer) and assigned it the value of 8, thereby mapping the LED to pin 8. Please note that variable names are case-sensitive!

pinMode(LedPin, OUTPUT);

After confirming the pin, we need to configure the pin's mode. Pins can operate in either input mode or output mode; in this case, we are using output mode.

digitalWrite(LedPin, HIGH); digitalWrite(LedPin, LOW);

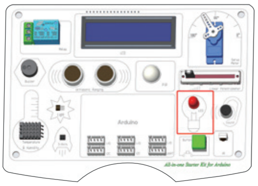
Having completed the preliminary operations to control the pin, you can now control the output voltage level of the specified pin using the digital write function. When set to a high level, the pin outputs a high voltage level, causing the LED to light up; when set to a low level, the pin outputs a low voltage level, causing the LED to turn off.

Lesson 2 - Button Control Led

Introduction

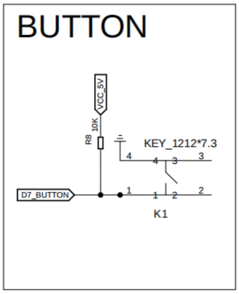
When a button is pressed, it displays a high signal level, and when released, it shows a low signal level. In this lesson, we will use the button to control the state of the LED light: the LED light turns on when the button is pressed and turns off when the button is released.

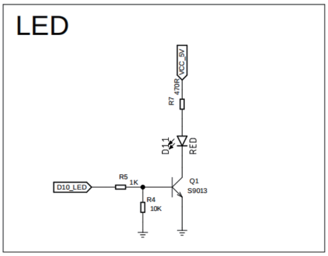
Hardware required



• All_in_one Starter Kit for Arduino x1
• USB Cable x1

Hardware schematic

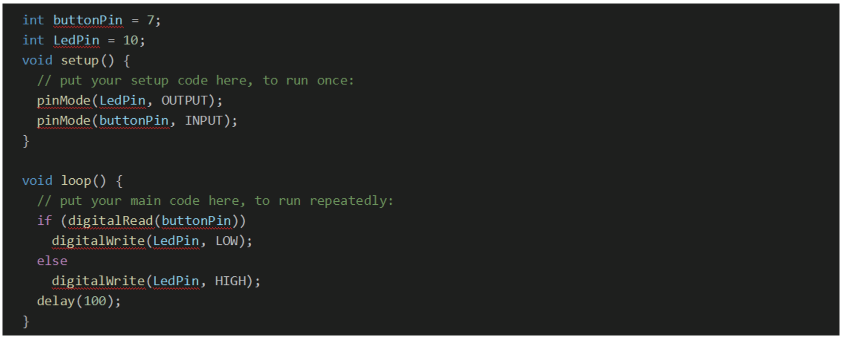




What will you see

When you hold down the button, the LED stays on, and when you release your hand, the LED goes out. If the LED is not working as expected, please ensure that the program is running correctly.

Complete Code



```
int buttonPin = 7;
int LedPin = 10;
void setup() {
  // put your setup code here, to run once:
  pinMode(LedPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  if (digitalRead(buttonPin))
    digitalWrite(LedPin, LOW);
  else
    digitalWrite(LedPin, HIGH);
  delay(100);
}
```

Main Code Explanation

int buttonPin = 7;int LedPin = 10;

Variables are placeholders for values that may change within the code. Variables must be introduced or "declared" before they are used. In this case, we have declared two variables, one of type `LedPin` and another of type `buttonPin`.
Assign integer values of type `int` to be 10 and 7, respectively, so that the LED corresponds to pin 10, and the button is connected to pin 8. Remember that variable names are case-sensitive!

pinMode(LedPin, OUTPUT);pinMode(buttonPin, INPUT);

Here, we use the `pinMode` function to assign modes to the LED pin and the button pin. Configure `LedPin` in output mode and `buttonPin` in input mode.

if (digitalRead(buttonPin))   digitalWrite(LedPin, LOW);   else   digitalWrite(LedPin, HIGH);

The `if` statement determines whether to turn the LED on and off based on the value returned by the `digitalRead` function when reading the `buttonPin` pin. When it returns `true`, `buttonPin` is in a high state, meaning it is not pressed. At this time, the `digitalWrite` function is used to pull the `LedPin` pin high, causing the LED to turn off. When it returns `false`, `buttonPin` is in a low state, meaning the button has been pressed. At this time, the `digitalWrite` function is used to pull the `LedPin` pin high, causing the LED to light up.

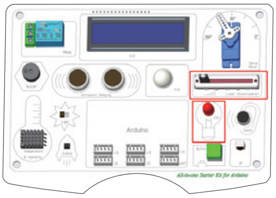
Lesson 3 - Breathing LED

Introduction

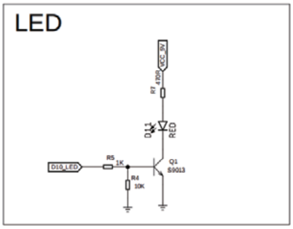
In this lesson, we will use a potentiometer with a maximum resistance of 10kΩ to create a breathing LED light effect. As you slide it from left to right, its output voltage will range from 0v to 5v (VCC). In this course section, we will adjust the LED light using the potentiometer to achieve a breathing light effect!

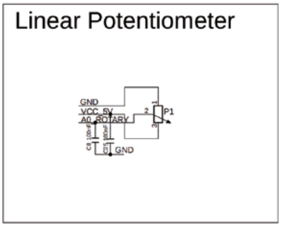
Hardware required

- All_in_one Starter Kit for Arduino x1
- USB Cable x1



Hardware schematic

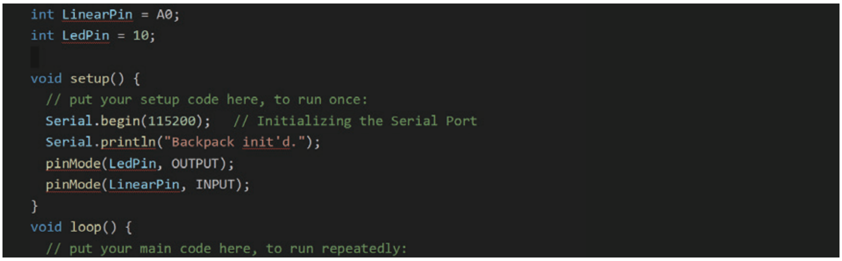




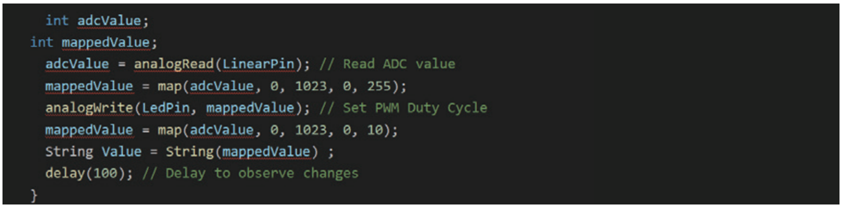
What will you see

As the slider of the potentiometer is moved, the brightness of the LED light will change accordingly. When slid to the right to its maximum position, the LED will be at its brightest; when slid to the left to its minimum, the LED will turn off. If the LED is not working as expected, please ensure that the program is running correctly.

Complete Code



```
int LinearPin = A0;
int LedPin = 10;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);   // Initializing the Serial Port
  Serial.println("Backpack init'd.");
  pinMode(LedPin, OUTPUT);
  pinMode(LinearPin, INPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
```



```
  int adcValue;
int mappedValue;
  adcValue = analogRead(LinearPin); // Read ADC value
  mappedValue = map(adcValue, 0, 1023, 0, 255);
  analogWrite(LedPin, mappedValue); // Set PWM Duty Cycle
  mappedValue = map(adcValue, 0, 1023, 0, 10);
  String Value = String(mappedValue) ;
  delay(100); // Delay to observe changes
}
```

Main Code Explanation

int LinearPin = A0;int LedPin = 10;

Variables are placeholders for values that may change within the code. Variables must be introduced or "declared" before they are used. In this context, we have declared two variables, one of type `LedPin` and another of type `LinearPin`.
Assign the values 10 and A0 to integer types (Int), with the LED corresponding to pin 10 and the sliding module to pin A0. Remember, variable names are case-sensitive!

pinMode(LedPin, OUTPUT);pinMode(LinearPin , INPUT);

We utilize the `pinMode` function to assign modes to the LED pin and the potentiometer pin. The `LedPin` is configured in output mode, while the `LinearPin` is set to input mode.

int adcValue;int mappedValue;

Two intermediate variables, `adcValue` and `mappedValue`, have been defined. The `adcValue` is used to store the value read from the ADC, while `mappedValue` is used to convert the value read from `adcValue` into a duty cycle.

int adcValue;int mappedValue;

Two intermediate variables, `adcValue` and `mappedValue`, have been defined. The `adcValue` is used to store the value read from the ADC, while `mappedValue` is used to convert the value read from `adcValue` into a duty cycle.

adcValue = analogRead(LinearPin); mappedValue = map(adcValue, 0, 1023, 0, 255);

The `analogRead` function is used to read the ADC value, and the `map` function is used to transform the read ADC value.

analogWrite(LedPin, mappedValue);

After the preliminary preparations, we need to apply the obtained duty cycle value to the LED, which can be accomplished by using the `analogWrite` function.

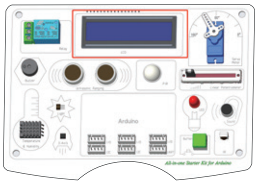
Lesson 4 - LCD Display

Introduction

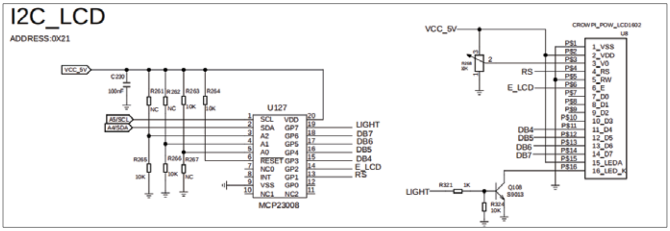
In this lesson, we will utilize the LCD module on the development board to display text. The LCD module is connected to the development board via the I2C communication protocol. This LCD module is easy to operate and user-friendly.

Hardware required

- All_in_one Starter Kit for Arduino x1
- USB Cable x1



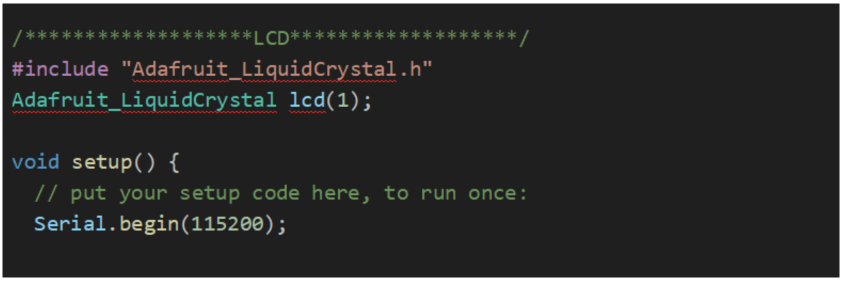
Hardware schematic



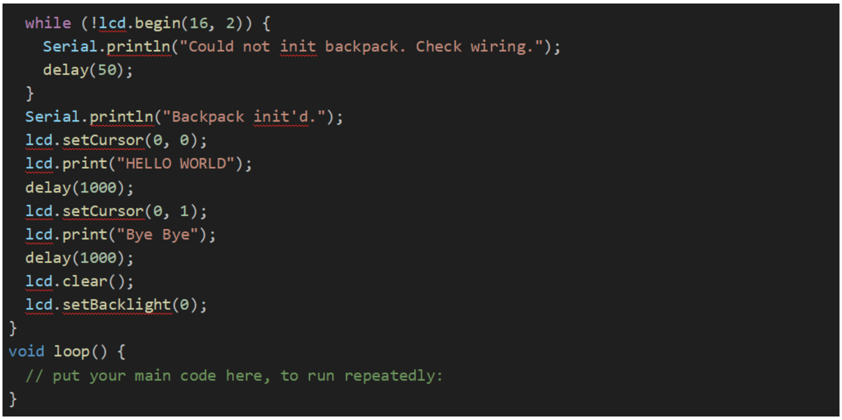
What will you see

You will see the LCD light up and display "HELLO WORLD!" on the first line. After a one-second interval, "Bye Bye" will appear on the second line of the LCD. Another second later, the LCD will clear the display and turn off the screen. If the LCD does not operate as expected, please ensure that the program is running correctly.

Complete Code



```
/******************LCD******************/
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
```



```
  while (!lcd.begin(16, 2)) {
    Serial.println("Could not init backpack. Check wiring.");
    delay(50);
  }
  Serial.println("Backpack init'd.");
  lcd.setCursor(0, 0);
  lcd.print("HELLO WORLD");
  delay(1000);
  lcd.setCursor(0, 1);
  lcd.print("Bye Bye");
  delay(1000);
  lcd.clear();
  lcd.setBacklight(0);
}
void loop() {
  // put your main code here, to run repeatedly:
}
```

Main Code Explanation

`#include "Adafruit_LiquidCrystal.h"`

Before using the LCD, we need to do some preparatory work. First, we need to import the relevant libraries.

`Adafruit_LiquidCrystal lcd(1);`

Define a global variable for the LCD, and then we will use this variable to manipulate the LCD display.

while (!lcd.begin(16, 2)) {Serial.println("Could not init backpack. Check wiring.");delay(50);}

A variable named `LCD` is used to initialize the LCD module. If the LCD initialization fails, an error message will be repeatedly sent to the serial port until initialization is successful.

```
lcd.setCursor(0, 0);lcd.print("HELLO WORLD");delay(1000);
lcd.setCursor(0, 1);lcd.print("Bye Bye");
```

Let's familiarize ourselves with the operation process of the LCD.
First, we use the `setCursor` function to position the cursor at the 0th row and 0th column, which determines the starting position for the next display. This function is included in the `Adafruit_LiquidCrystal` class. After setting the cursor position, we can display text. We call the `lcd.print` function to show the text we wish to display. In this example, "HELLO WORLD" is displayed on the 0th row, while "Bye Bye" is displayed on the 1st row.

`lcd.clear();lcd.setBacklight(0);`

After the text is displayed, it is necessary to clear the LCD screen and turn off the backlight. We use the `lcd.clear()` function to clear the screen, followed by the `lcd.setBacklight(0)` function to turn off the backlight.

Lesson 5 - Moisture Monitor

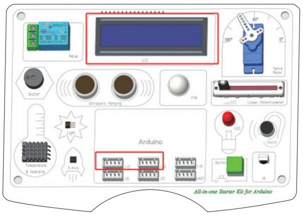
Introduction

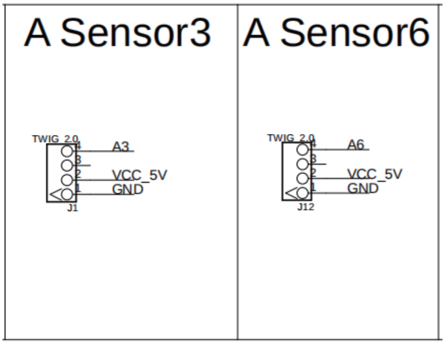
In this lesson, we will use an LCD module and a soil moisture sensor to check the soil's moisture level. When the soil moisture sensor is inserted into the soil, the LCD will continuously display the moisture values read from the soil. We can use these numerical values to assess the condition of the soil.

Hardware required

- All_in_one Starter Kit for Arduino x1
- Crowtail - Moisture Sensor x1
- USB Cable x1

Note: Ensure that the interface A of the Crowtail - Moisture Sensor module matches the one specified in the code. In this case, we are using A Sensor3.

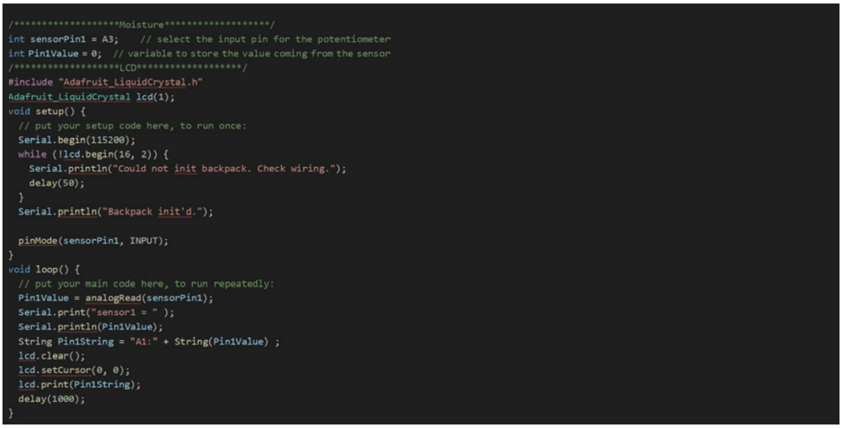




What will you see

You will observe the LCD continuously displaying the values obtained from the soil moisture sensor. If this does not occur on the LCD, please ensure that the program is running correctly.

Complete Code



```
/*******************Moisture*******************/
int sensorPin1 = A3;    // select the input pin for the potentiometer
int Pin1Value = 0;  // variable to store the value coming from the sensor
/*******************LCD*******************/
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  while (!lcd.begin(16, 2)) {
    Serial.println("Could not init backpack. Check wiring.");
    delay(50);
  }
  Serial.println("Backpack init'd.");

  pinMode(sensorPin1, INPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
  Pin1Value = analogRead(sensorPin1);
  Serial.print("sensor1 = " );
  Serial.println(Pin1Value);
  String Pin1String = "A1:" + String(Pin1Value) ;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(Pin1String);
  delay(1000);
}
```

Main Code Explanation

```
int sensorPin1 = A3; int Pin1Value = 0; #include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);
```

Before we start using the module, we need to do some preparatory work. First, we import the relevant libraries, then define the LCD variable, and specify the pin used by the soil moisture sensor module.We use the variable `sensorPin1` and assign pin A3 to it. We also define an intermediate variable `Pin1Value` to store the value read from the soil moisture sensor module.

```
while (!lcd.begin(16, 2)) {Serial.println("Could not init backpack. Check wiring.");delay(50);
  }Serial.println("Backpack init'd.");pinMode(sensorPin1, INPUT);
```

After the preparatory work is completed, we can begin initializing the LCD and the pins. Using the statements mentioned above, we can initialize the LCD and configure the `sensorPin1` pin as an input mode.

```
Pin1Value = analogRead(sensorPin1);String
Pin1String = "A3:" + String(Pin1Value) ;lcd.clear();
lcd.setCursor(0, 0);lcd.print(Pin1String);
```

After initialization is complete, you can read the value from the soil moisture sensor using an analog function and display it on the screen with the `lcd.print` function .Before displaying the information, you need to set the cursor position with `lcd.setCursor(0, 0)` and clear the screen by calling `lcd.clear()` to prevent previous data from interfering with the current display.

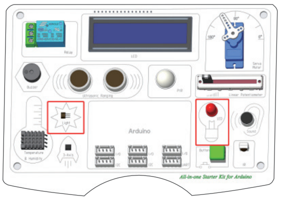
Lesson 6 - Intelligent Street Light

Introduction

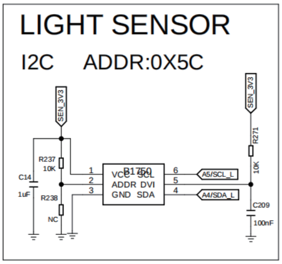
In this lesson, you will learn how to obtain light intensity information from a light sensor module and how to control the LED light's on/off state based on this data. By determining different levels of brightness, you can achieve intelligent control over the LED's operation, smartly managing the LED's activation and deactivation to prevent unnecessary energy consumption and achieve energy-saving goals.

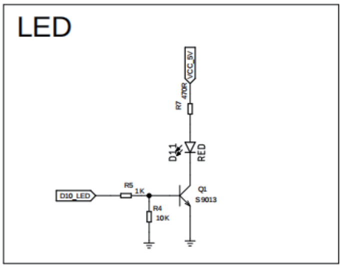
Hardware required

- All_in_one Starter Kit for Arduino x1
- USB Cable x1



Hardware required

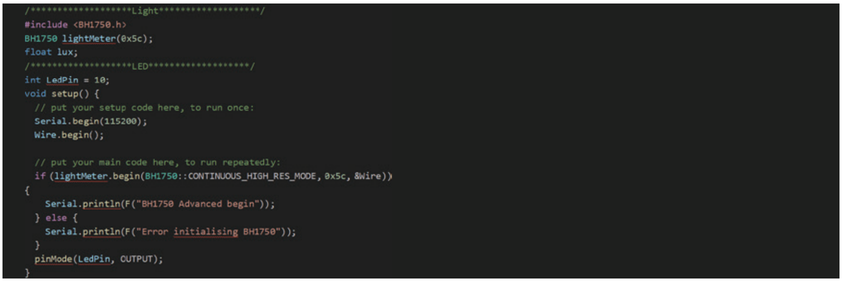




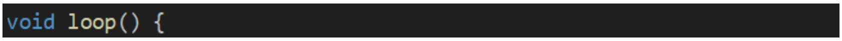
What will you see

When you cover the top of the light sensor to simulate a light-deprived environment, you will see the LED light up. When you remove your hand to simulate a lighted environment, the LED will turn off. If this does not happen, please check to ensure that the program is executing correctly.

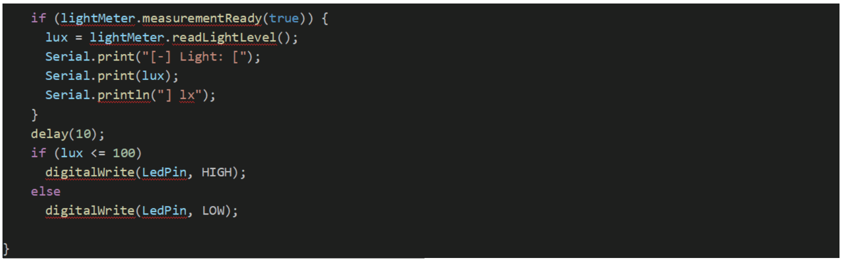
Complete Code



```
/*******************Light*******************/
#include <BH1750.h>
BH1750 lightMeter(0x5c);
float lux;
/*******************LED*******************/
int LedPin = 10;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Wire.begin();

  // put your main code here, to run repeatedly:
  if (lightMeter.begin(BH1750::CONTINUOUS_HIGH_RES_MODE, 0x5c, &Wire))
{
    Serial.println(F("BH1750 Advanced begin"));
  } else {
    Serial.println(F("Error initialising BH1750"));
  }
  pinMode(LedPin, OUTPUT);
}
```



```
void loop() {
```



```
  if (lightMeter.measurementReady(true)) {
    lux = lightMeter.readLightLevel();
    Serial.print("[-] Light: [");
    Serial.print(lux);
    Serial.println("] lx");
  }
  delay(10);
  if (lux <= 100)
    digitalWrite(LedPin, HIGH);
  else
    digitalWrite(LedPin, LOW);

}
```

Main Code Explanation

```
#include <BH1750.h>BH1750 lightMeter(0x5c);float lux;int LedPin = 10;
```

Building upon the introduction of the relevant libraries, we have defined a variable for the light-sensing module, "photometer", and stored the luminance data in the variable "lux". Concurrently, we have also defined a pin variable for controlling the LED, "LedPin".

```
Wire.begin(); if (lightMeter.begin(BH1750::CONTINUOUS_HIGH_RES_MODE, 0x5c, &
Wire)) {
Serial.println(F("BH1750 Advanced begin")); } else {Serial.println(F("Error initialising
BH1750"));}
```

Before utilizing the Light module, it is necessary to initialize the IIC (I2C) communication protocol and the Light module itself. Once the initialization is complete, we can proceed to read data from the Light module via the IIC interface.

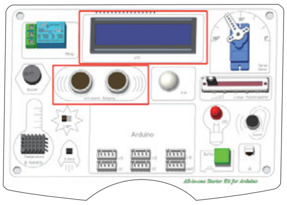
```
if (lightMeter.measurementReady(true)) {lux = lightMeter.readLightLevel();}
if (lux <= 100)digitalWrite(LedPin, HIGH);else digitalWrite(LedPin, LOW);
```

We initially employ an if statement to ascertain whether the return value of the function "lightMeter.measurementReady(true)" indicates that data can be read. If data is available for reading, we then use the variable `lux` to store the data retrieved by the function "lightMeter.readLightLevel()". Following this, we utilize another if statement to evaluate the magnitude of the reading. Generally, readings below 100 are considered to represent an environment with insufficient or no light, in which case we set the LED pin to a high level to turn on the LED; conversely, readings above 100 are deemed to indicate a well-lit environment, and in such cases, we set the LED pin to a low level to extinguish the LED.

Lesson 7 - Ultrasonic Ranging Display

Introduction

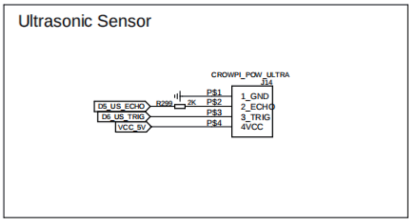
In this lesson, we will learn how to utilize an ultrasonic module. With the help of this module, we can measure the distance between a flat surface in front of the module and the module itself. We can create an ultrasonic distance meter that will display the measured distance on an LCD module.

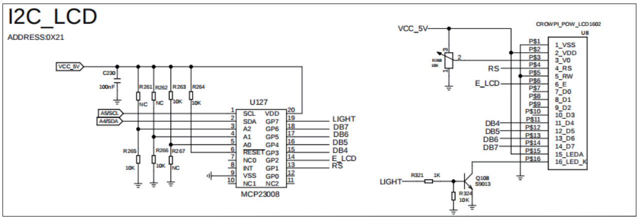


Hardware required

- All_in_one Starter Kit for Arduino x1
- USB Cable x1

Hardware schematic



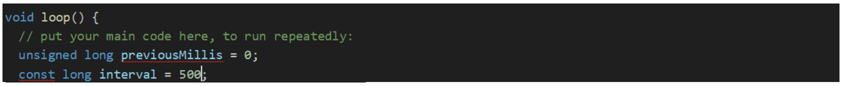


What will you see

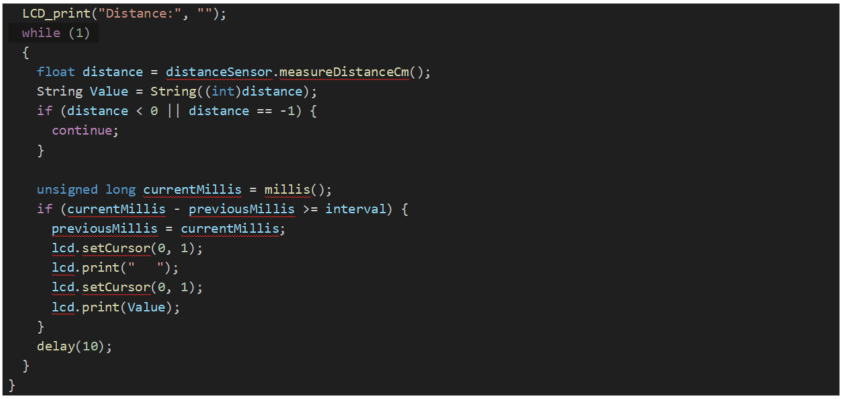
You will observe the LCD screen continuously updating with distance data measured by the ultrasonic sensor. As the flat surface in front of the ultrasonic module moves, the measured distance values will also change accordingly. If this phenomenon does not occur, please check to ensure that the program is executing correctly.

Complete Code

```
/******************LCD******************/
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);

String NULL_TXT = "                ";
void LCD_print(String txt1, String txt2)
{
  lcd.setCursor(0, 0);
  lcd.print("                ");
  lcd.setCursor(0, 1);
  lcd.print("                ");
  lcd.setCursor(0, 0);
  lcd.print(txt1);
  lcd.setCursor(0, 1);
  lcd.print(txt2);
}
/******************Ultrasonic******************/
#include <HCSR04.h>
// Initialize sensor that uses digital pins 13 and 12.
const byte triggerPin = 6;
const byte echoPin = 5;
UltraSonicDistanceSensor distanceSensor(triggerPin, echoPin);
int Cursor_position = 0;
int flag = 0;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  while (!lcd.begin(16, 2)) {
    Serial.println("Could not init backpack. Check wiring.");
    delay(50);
  }
  Serial.println("Backpack init'd.");
}
```



```
void loop() {
  // put your main code here, to run repeatedly:
  unsigned long previousMillis = 0;
  const long interval = 500;
```



```
  LCD_print("Distance:", "");
  while (1)
  {
    float distance = distanceSensor.measureDistanceCm();
    String Value = String((int)distance);
    if (distance < 0 || distance == -1) {
      continue;
    }

    unsigned long currentMillis = millis();
    if (currentMillis - previousMillis >= interval) {
      previousMillis = currentMillis;
      lcd.setCursor(0, 1);
      lcd.print("   ");
      lcd.setCursor(0, 1);
      lcd.print(Value);
    }
    delay(10);
  }
}
```

Main Code Explanation

#include "Adafruit_LiquidCrystal.h"Adafruit_LiquidCrystal lcd(1);

#include <HCSR04.h>UltraSonicDistanceSensor distanceSensor(triggerPin, echoPin);

Firstly, since we will be utilizing both the ultrasonic module and the LCD module, it is necessary to initialize these two modules before use.

float distance = distanceSensor.measureDistanceCm();

After initializing the modules, we can proceed to read data from them. Here, we use the function `distanceSensor.measureDistanceCm()` to read the measured distance and store it in an intermediate variable called `distance`.

lcd.setCursor(0, 1);lcd.print(Value);

Once we have obtained the distance value, we can then display it on the LCD screen. By calling the function `lcd.setCursor(0, 1)`, we position the cursor at the 0th column and the 1st row. Subsequently, we use the `lcd.print(Value)` function to display the distance value on the screen.

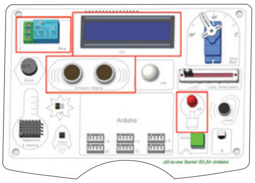
Lesson 8 - Obstacle Close Range Alarm

Introduction

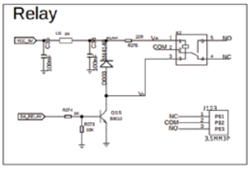
In this section, we will delve deeper into the ultrasonic module and learn how to coordinate its operation with other modules. We will use the distance data obtained from the ultrasonic module to control the activation and deactivation of the relay module and the LED module, thereby implementing the function of ultrasonic obstacle avoidance.

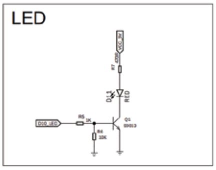
Hardware required

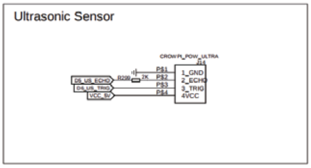
- All_in_one Starter Kit for Arduino x1
- USB Cable x1

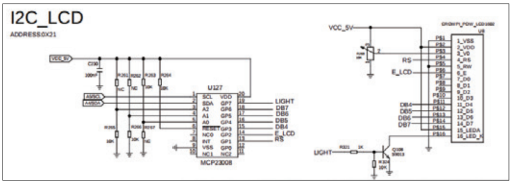


Hardware schematic





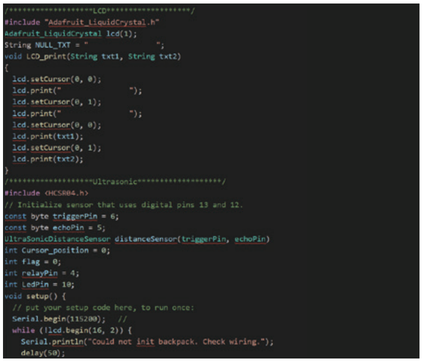




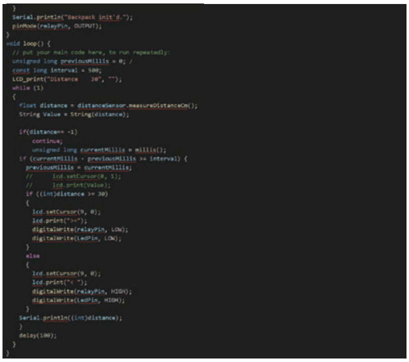
What will you see

You will observe that as the distance measured by the ultrasonic module changes, when the distance is less than 30 centimeters, both the relay and the LED light will activate simultaneously, indicating the detection of an obstacle. If the distance reaches or exceeds 30 centimeters, the relay and LED light will turn off, signifying that the surrounding environment is safe and there are no obstacles. If this process does not occur, please verify that the program is being executed correctly.

Complete Code



```
/*******************LCD*******************/
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);
String NULL_TXT = "                ";
void LCD_print(String txt1, String txt2)
{
  lcd.setCursor(0, 0);
  lcd.print("                ");
  lcd.setCursor(0, 1);
  lcd.print("                ");
  lcd.setCursor(0, 0);
  lcd.print(txt1);
  lcd.setCursor(0, 1);
  lcd.print(txt2);
}
/*******************Ultrasonic*******************/
#include <HCSR04.h>
// Initialize sensor that uses digital pins 13 and 12.
const byte triggerPin = 6;
const byte echoPin = 5;
UltraSonicDistanceSensor distanceSensor(triggerPin, echoPin)
int Cursor_position = 0;
int flag = 0;
int relayPin = 4;
int LedPin = 10;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);   //
  while (!lcd.begin(16, 2)) {
    Serial.println("Could not init backpack. Check wiring.");
    delay(50);
```



```
  }
  Serial.println("Backpack init'd.");
  pinMode(relayPin, OUTPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
  unsigned long previousMillis = 0; /
  const long interval = 500;
  LCD_print("Distance    30", "");
  while (1)
  {
    float distance = distanceSensor.measureDistanceCm();
    String Value = String(distance);

    if(distance== -1)
        continue;
        unsigned long currentMillis = millis();
    if (currentMillis - previousMillis >= interval) {
      previousMillis = currentMillis;
      //      lcd.setCursor(0, 1);
      //      lcd.print(Value);
      if ((int)distance >= 30)
      {
        lcd.setCursor(9, 0);
        lcd.print(">=");
        digitalWrite(relayPin, LOW);
        digitalWrite(LedPin, LOW);
      }
      else
      {
        lcd.setCursor(9, 0);
        lcd.print("< ");
        digitalWrite(relayPin, HIGH);
        digitalWrite(LedPin, HIGH);
      }
    Serial.println((int)distance);
    }
    delay(100);
  }
}
```

Main Code Explanation

#include "Adafruit_LiquidCrystal.h"Adafruit_LiquidCrystal lcd(1);
#include <HCSR04.h>UltraSonicDistanceSensor distanceSensor(triggerPin, echoPin);
To begin with, we will be utilizing both an ultrasonic module and an LCD module, which necessitates initializing these two modules before we can use them.

float distance = distanceSensor.measureDistanceCm();
After initializing the modules, we can proceed to read data from them. We use the `measureDistanceCm()` function of the distance sensor to read the measured distance, and then we store this value in an intermediate variable called `distance`.
if ((int)distance >= 30){lcd.setCursor(9, 0);lcd.print(">=");digitalWrite(relayPin, LOW);
digitalWrite(LedPin, LOW);}else{lcd.setCursor(9, 0);lcd.print("< ");digitalWrite(relayPin, HIGH);
}

The `if` statement is utilized to ascertain whether the relay and LED need to be activated. When the distance is less than 30 centimeters, the `digitalWrite(relayPin, HIGH);` and `digitalWrite(LedPin, HIGH);` functions are employed to activate the relay and LED, respectively. Conversely, when the distance is greater than or equal to 30 centimeters, the `digitalWrite(relayPin, LOW);` and `digitalWrite(LedPin, LOW);` functions are used to deactivate the relay and LED. Additionally, the results are displayed on the LCD.

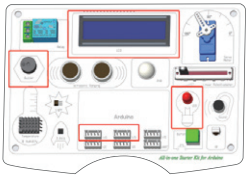
Lesson 9 - Plant Watering Reminder System

Introduction

In this lesson, we will learn how to monitor soil moisture changes using a soil moisture sensor. When the moisture level drops to 10%, a buzzer will sound an alarm to alert you that it's time to water the plants; when the moisture is between 10% and 20%, an LED light will flash, indicating that the plants are slightly water-deficient and reminding you to water them as soon as possible; when the moisture level is greater than 20%, it signifies that the plants are in good condition and no additional watering is required.

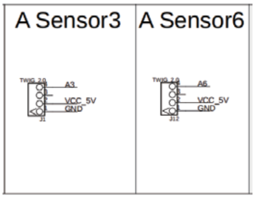
Hardware required

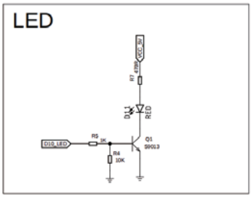
- All_in_one Starter Kit for Arduino x1
- USB Cable x1
- Crowtail - Moisture Sensor x1

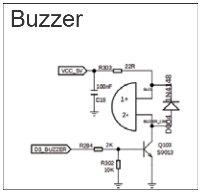


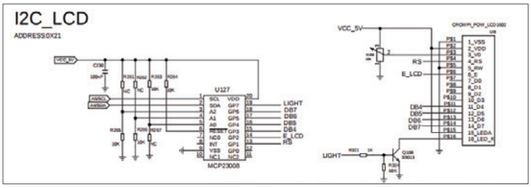
Note: The interface to which the Crowtail - Moisture Sensor module is connected must be consistent with the code. In this case, we are using the A Sensor3 interface for the connection.

Hardware schematic





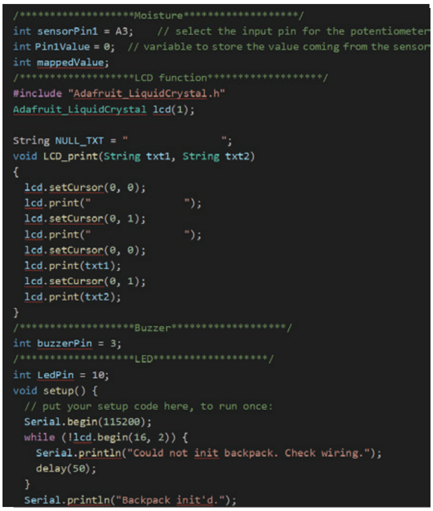




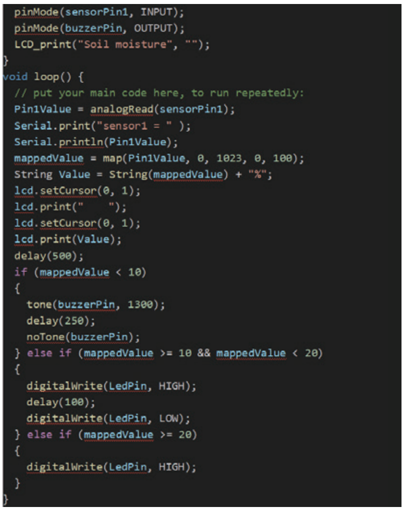
What will you see

You will observe that the LED and buzzer respond to changes in soil moisture levels. When the buzzer sounds an alarm, it indicates that the plants are severely water-deficient and require immediate watering; when the LED flashes, it signifies that the plants are slightly water-deficient and need to be watered promptly; when the LED stays on continuously, it suggests that the plants have adequate moisture and are thriving.

Complete Code



```
/*******************Moisture*******************/
int sensorPin1 = A3;    // select the input pin for the potentiometer
int Pin1Value = 0;  // variable to store the value coming from the sensor
int mappedValue;
/*******************LCD function******************/
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);

String NULL_TXT = "                ";
void LCD_print(String txt1, String txt2)
{
  lcd.setCursor(0, 0);
  lcd.print("                ");
  lcd.setCursor(0, 1);
  lcd.print("                ");
  lcd.setCursor(0, 0);
  lcd.print(txt1);
  lcd.setCursor(0, 1);
  lcd.print(txt2);
}
/*******************Buzzer*******************/
int buzzerPin = 3;
/*******************LED*******************/
int LedPin = 10;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  while (!lcd.begin(16, 2)) {
    Serial.println("Could not init backpack. Check wiring.");
    delay(50);
  }
  Serial.println("Backpack init'd.");
```



```
  pinMode(sensorPin1, INPUT);
  pinMode(buzzerPin, OUTPUT);
  LCD_print("Soil moisture", "");
}
void loop() {
  // put your main code here, to run repeatedly:
  Pin1Value = analogRead(sensorPin1);
  Serial.print("sensor1 = " );
  Serial.println(Pin1Value);
  mappedValue = map(Pin1Value, 0, 1023, 0, 100);
  String Value = String(mappedValue) + "%";
  lcd.setCursor(0, 1);
  lcd.print("    ");
  lcd.setCursor(0, 1);
  lcd.print(Value);
  delay(500);
  if (mappedValue < 10)
  {
    tone(buzzerPin, 1300);
    delay(250);
    noTone(buzzerPin);
  } else if (mappedValue >= 10 && mappedValue < 20)
  {
    digitalWrite(LedPin, HIGH);
    delay(100);
    digitalWrite(LedPin, LOW);
  } else if (mappedValue >= 20)
  {
    digitalWrite(LedPin, HIGH);
  }
}
```

Main Code Explanation

```
/******************Moisture******************/
int sensorPin1 = A3;
int Pin1Value = 0;
int mappedValue;
/******************LCD function******************/
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);
/******************Buzzer******************/
int buzzerPin = 3;

/******************LED******************/
int LedPin = 10;
```

Before we begin coding our plant watering reminder system, we need to define the relevant variables. The variable `sensorPin1` represents the soil moisture sensor connected to pin A3. `pin1Value` and `mappedValue` are intermediate variables that store the values before and after transformation, respectively. Subsequently, we define an `Adafruit_LiquidCrystal` class variable `lcd`, passing the argument 1 to indicate the selection of the first address. Lastly, we define two variables: `buzzerPin` to specify the pin connected to the buzzer, and `LedPin` for the pin connected to the LED light.

```
lcd.begin(16, 2);  pinMode(sensorPin1, INPUT);
pinMode(LedPin, OUTPUT);pinMode(buzzerPin, OUTPUT);
```

Initialize the LCD, buzzer, LED, and soil moisture sensor pins. The LCD input parameters indicate a 16-column by 2-row LCD display. Initialize the buzzer using `pinMode`, set the LED pin to output mode, and the soil moisture sensor pin to input mode.

```
Pin1Value = analogRead(sensorPin1);mappedValue = map(Pin1Value, 0, 1023, 0, 100);
lcd.print(Value);if (mappedValue < 10);tone(buzzerPin, 1300);digitalWrite(LedPin, HIGH);
```

After initializing all sensors, obtain the soil moisture sensor value using the `analogRead` function, and display the converted value on the LCD using the `map` function. Use `if` statements to evaluate the values to determine whether to use the `tone` function to make the buzzer sound, and use the `digitalWrite` function to manipulate the LED light.

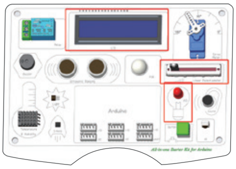
Lesson 10 - Brightness Display

Introduction

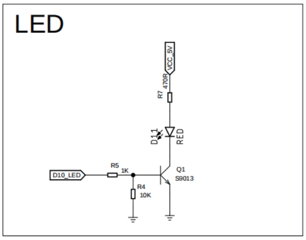
In this lesson, we will learn how to control the brightness of an LED using a slide sensor, allowing us to manually adjust the LED's brightness to our desired level.

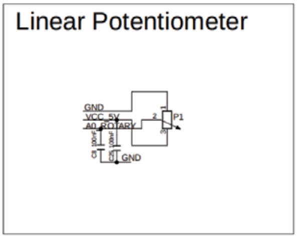
Hardware required

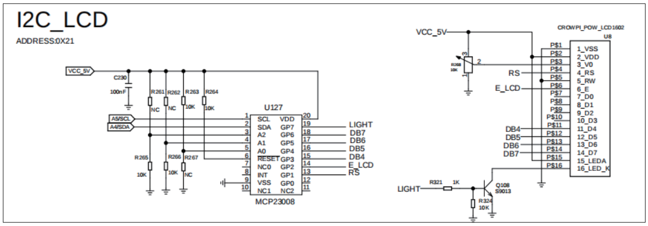
- All_in_one Starter Kit for Arduino x1
- USB Cable x1



Hardware schematic





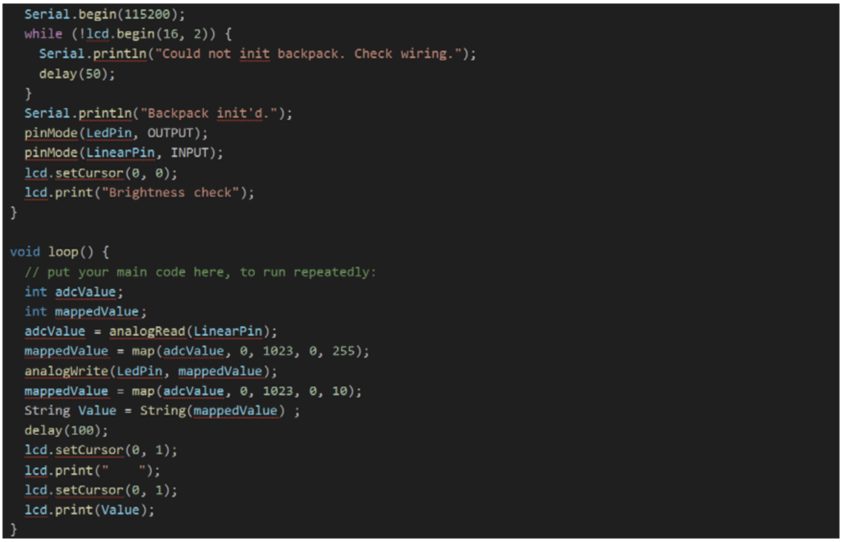


What will you see

We will learn how to control the brightness of an LED using a sliding sensor, and display the brightness level, ranging from 1 to 10, on an LCD to achieve the effect of controlling the brightness.

Complete Code

```
int LinearPin = A3;
int LedPin = 10;
/******************LCD******************/
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);
void setup() {
  // put your setup code here, to run once:
```



```
  Serial.begin(115200);
  while (!lcd.begin(16, 2)) {
    Serial.println("Could not init backpack. Check wiring.");
    delay(50);
  }
  Serial.println("Backpack init'd.");
  pinMode(LedPin, OUTPUT);
  pinMode(LinearPin, INPUT);
  lcd.setCursor(0, 0);
  lcd.print("Brightness check");
}

void loop() {
  // put your main code here, to run repeatedly:
  int adcValue;
  int mappedValue;
  adcValue = analogRead(LinearPin);
  mappedValue = map(adcValue, 0, 1023, 0, 255);
  analogWrite(LedPin, mappedValue);
  mappedValue = map(adcValue, 0, 1023, 0, 10);
  String Value = String(mappedValue) ;
  delay(100);
  lcd.setCursor(0, 1);
  lcd.print("    ");
  lcd.setCursor(0, 1);
  lcd.print(Value);
}
```

Main Code Explanation

int LinearPin = A3;int LedPin = 10;Adafruit_LiquidCrystal lcd(1);

We assign pin numbers to the sliding rheostat and define two variables, `LinearPin` and `LedPin`, to facilitate subsequent operations on the two sensors. Lastly, do not forget to create the LCD variable.

lcd.begin(16, 2);pinMode(LedPin, OUTPUT);pinMode(LinearPin, INPUT);

First, we initialize the LCD. Then, by using the `pinMode` function, we configure the `LedPin` as an output mode and the `LinearPin` as an input mode. These modes are set according to the way the sensors are used. The sliding rheostat is set to input mode because it needs to read the varying resistance, while the LED is set to output mode because the microcontroller needs to change its state.

int adcValue;int mappedValue;adcValue = analogRead(LinearPin);
mappedValue = map(adcValue, 0, 1023, 0, 255);analogWrite(LedPin, mappedValue);

Once the preparatory work is complete, we can begin to operate the sensors. Let's start by defining two intermediate variables, `adcValue` and `mappedValue`. The `adcValue` variable is used to store the raw value read from the sliding module, while the `mappedValue` variable is used to store the value after being transformed using the `map` function. Since the raw ADC value cannot be used directly, the `map` function is employed for transformation. Finally, based on the transformed values, we adjust the voltage of the LED pin through analogWrite to achieve the effect of controlling the brightness.

```
mappedValue = map(adcValue, 0, 1023, 0, 10);String Value = String(mappedValue) ;
lcd.setCursor(0, 1);lcd.print(Value);
```

After controlling the LED's output voltage, remember that we also need to display the brightness value on the LCD screen. Before displaying, we need to convert the raw ADC values. Then, we use `lcd.setCursor` to position the cursor, and finally, we output the brightness value through `lcd.print`.

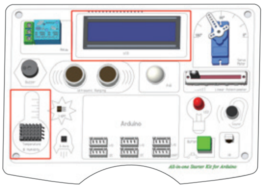
Lesson 11 - Temperature&Humidity Detecting System

Introduction

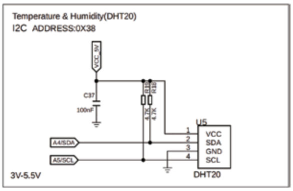
In this lesson, we will introduce how to use a temperature and humidity module to acquire temperature and humidity data and display it on the screen. With this module, we can obtain real-time local temperature and humidity data, which will be beneficial for our subsequent research.

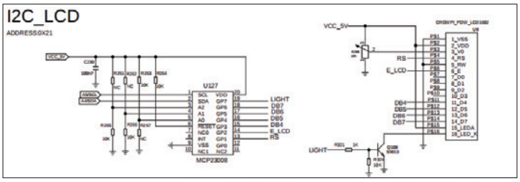
Hardware required

- All_in_one Starter Kit for Arduino x1
- USB Cable x1



Hardware schematic

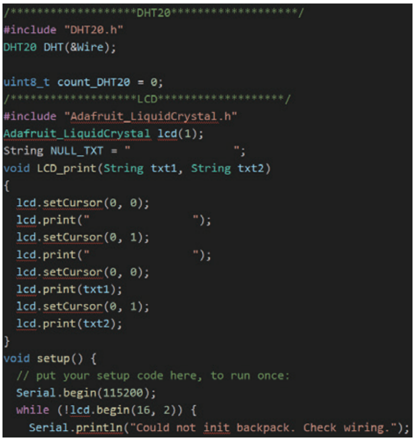




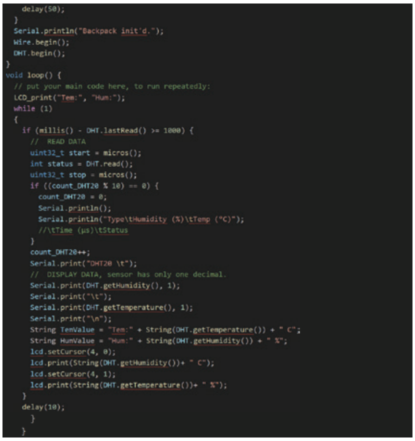
What will you see

You will observe that the temperature and humidity data from the sensors are updated on the LCD screen at regular intervals. If the updates do not occur, please check whether the firmware has been programmed correctly and whether the hardware connections are functioning properly.

Complete Code



```
/******************DHT20******************/
#include "DHT20.h"
DHT20 DHT(&Wire);

uint8_t count_DHT20 = 0;
/******************LCD******************/
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);
String NULL_TXT = "                ";
void LCD_print(String txt1, String txt2)
{
  lcd.setCursor(0, 0);
  lcd.print("                ");
  lcd.setCursor(0, 1);
  lcd.print("                ");
  lcd.setCursor(0, 0);
  lcd.print(txt1);
  lcd.setCursor(0, 1);
  lcd.print(txt2);
}
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  while (!lcd.begin(16, 2)) {
    Serial.println("Could not init backpack. Check wiring.");
```



```
    delay(50);
  }
  Serial.println("Backpack init'd.");
  Wire.begin();
  DHT.begin();
}
void loop() {
  // put your main code here, to run repeatedly:
  LCD_print("Tem:", "Hum:");
  while (1)
  {
    if (millis() - DHT.lastRead() >= 1000) {
      //  READ DATA
      uint32_t start = micros();
      int status = DHT.read();
      uint32_t stop = micros();
      if ((count_DHT20 % 10) == 0) {
        count_DHT20 = 0;
        Serial.println();
        Serial.println("Type\tHumidity (%)\tTemp (°C)");
        //\tTime (µs)\tStatus
      }
      count_DHT20++;
      Serial.print("DHT20 \t");
      //  DISPLAY DATA, sensor has only one decimal.
      Serial.print(DHT.getHumidity(), 1);
      Serial.print("\t");
      Serial.print(DHT.getTemperature(), 1);
      Serial.print("\n");
      String TemValue = "Tem:" + String(DHT.getTemperature()) + " C";
      String HumValue = "Hum:" + String(DHT.getHumidity()) + " %";
      lcd.setCursor(4, 0);
      lcd.print(String(DHT.getHumidity())+ " C");
      lcd.setCursor(4, 1);
      lcd.print(String(DHT.getTemperature())+ " %");
    }
    delay(10);
      }
    }
```

Main Code Explanation

/******************DHT20******************/
#include "DHT20.h"
DHT20 DHT(&Wire);
/******************LCD******************/
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);

Before we start working with the sensors, we need to make some preliminary definitions to simplify our subsequent operations with the sensors. We begin by including the necessary libraries, and then we define variables for the DHT20 class and the LCD class.

Wire.begin();DHT.begin();lcd.begin(16, 2);

We utilize predefined variables to commence the initialization of the DHT20 temperature and humidity sensor. Note that before initializing the DHT20 sensor, we must first ensure that the `Wire.begin()` function initializes the I2C communication, otherwise the DHT20 sensor will not be able to initialize. Following this, we will proceed to initialize the LCD.

if (millis() - DHT.lastRead() >= 1000)

Due to the potential for unknown errors in the frequent and rapid collection of DHT20 data, we set an interval of 1 second and collect data using a non-blocking delay method. Each time data is updated, we check if the last data update was within the last second.

```
lcd.setCursor(4, 0);lcd.print(String(DHT.getHumidity())+ " C");
lcd.setCursor(4, 1);lcd.print(String(DHT.getTemperature())+ " %");
```

When the time interval has elapsed, we can retrieve data from the DHT20 and display it on the LCD. We can obtain the temperature and humidity by using `DHT.getHumidity()` and `DHT.getTemperature()`, respectively.

Lesson 12 - Servo Control

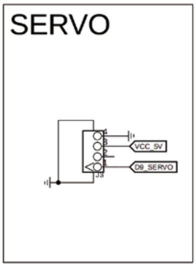
Introduction

In this lesson, we will learn the basic use of a servo motor. We will rotate the servo from 0 degrees to 180 degrees, and then back from 180 degrees to 0 degrees.

Hardware required

- Crowtail Starter All In One x1
- servo x 1

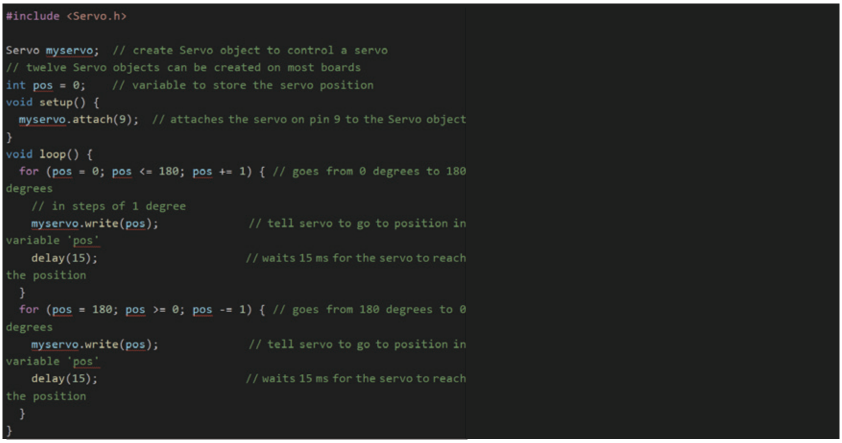
Hardware schematic



What will you see

You will observe the servo motor rotating from 0 degrees to 180 degrees, and then from 180 degrees back to 0 degrees. If this process does not occur, please check to ensure that the firmware has been correctly flashed and that the hardware connections are error-free.

Complete Code


```
#include <Servo.h>

Servo myservo;  // create Servo object to control a servo
// twelve Servo objects can be created on most boards
int pos = 0;    // variable to store the servo position
void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the Servo object
}
void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180
degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in
variable 'pos'
    delay(15);                       // waits 15 ms for the servo to reach
the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0
degrees
    myservo.write(pos);              // tell servo to go to position in
variable 'pos'
    delay(15);                       // waits 15 ms for the servo to reach
the position
  }
}
```

Complete Code

```
#include <Servo.h>
Servo myservo;
int pos = 0;
```

First, we import the library for operating the servo motor, and then we define the relevant variables for the servo motor to facilitate subsequent operations on it.

```
myservo.attach(9);
```

The servo is initialized with pin 9 designated as the pin to which the servo is connected.

```
for (pos = 0; pos <= 180; pos += 1) {


    myservo.write(pos);
    delay(15);
  }
  for (pos = 180; pos >= 0; pos -= 1) {
myservo.write(pos);
    delay(15);
  }
```

Create two loops: The first loop rotates the servo motor from 0 degrees to 180 degrees, incrementing by 1 degree each time;the second loop rotates the servo motor from 180 degrees back to 0 degrees, decrementing by 1 degree each time.

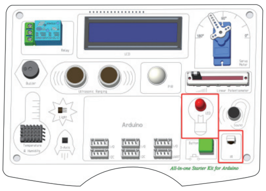
Lesson 13 - IR Control LED

Introduction

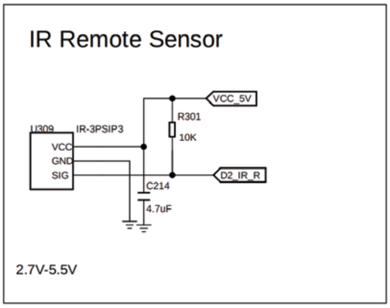
In this section of the course, we will delve into some advanced applications of infrared remote control technology. We can control the status of an LED using infrared remote control by recognizing different button signals. When button 1 is pressed, the LED remains on; when button 2 is pressed, the LED flashes; and when button 3 is pressed, the LED turns off.

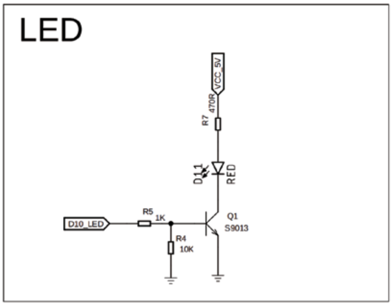
Hardware required

- All_in_one Starter Kit for Arduino x1
- USB Cable x1
- Crowtail- IR Emitter x1



Hardware schematic

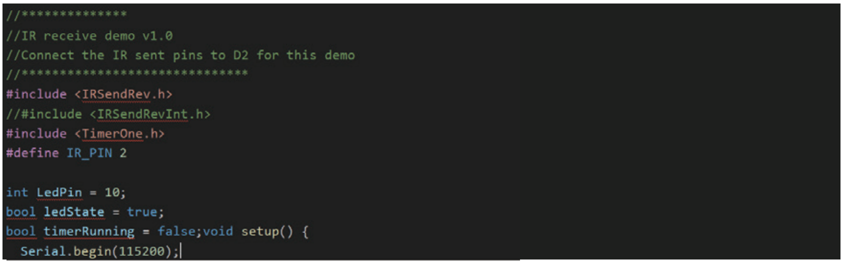




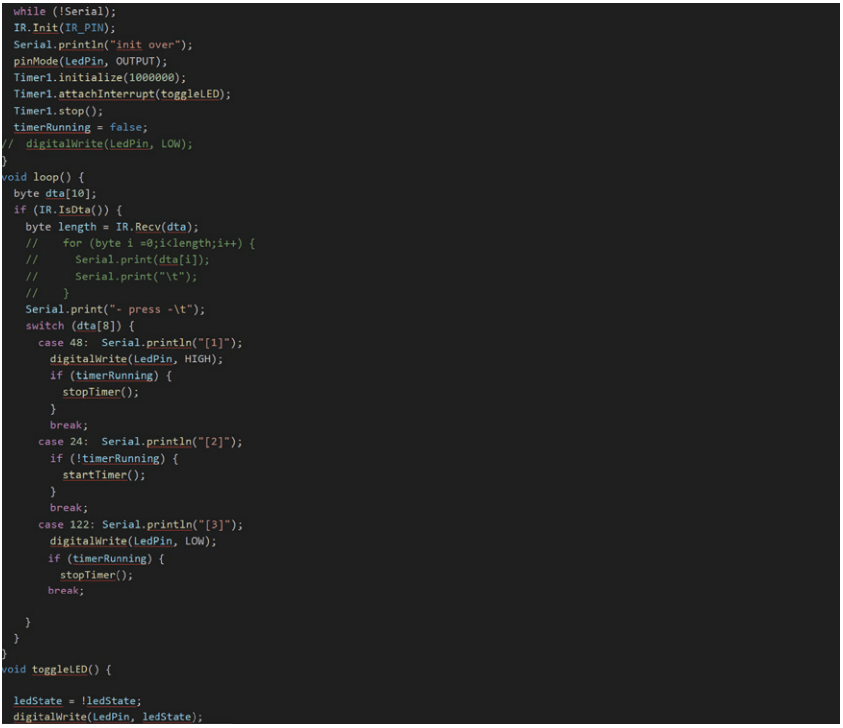
What will you see

You will observe that when infrared remote control button 1 is pressed, the LED stays on continuously; when button 2 is pressed, the LED flashes; and when button 3 is pressed, the LED turns off. If this process does not occur, please check to ensure that the firmware has been programmed correctly and that all hardware connections are functioning properly.

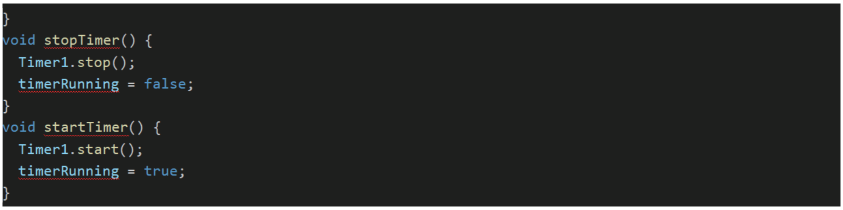
Complete Code



```
//**************
//IR receive demo v1.0
//Connect the IR sent pins to D2 for this demo
//******************************
#include <IRSendRev.h>
//#include <IRSendRevInt.h>
#include <TimerOne.h>
#define IR_PIN 2

int LedPin = 10;
bool ledState = true;
bool timerRunning = false;void setup() {
  Serial.begin(115200);
```



```
  while (!Serial);
  IR.Init(IR_PIN);
  Serial.println("init over");
  pinMode(LedPin, OUTPUT);
  Timer1.initialize(1000000);
  Timer1.attachInterrupt(toggleLED);
  Timer1.stop();
  timerRunning = false;
//  digitalWrite(LedPin, LOW);
}
void loop() {
  byte dta[10];
  if (IR.IsDta()) {
    byte length = IR.Recv(dta);
    //    for (byte i =0;i<length;i++) {
    //      Serial.print(dta[i]);
    //      Serial.print("\t");
    //    }
    Serial.print("- press -\t");
    switch (dta[8]) {
      case 48:  Serial.println("[1]");
        digitalWrite(LedPin, HIGH);
        if (timerRunning) {
          stopTimer();
        }
        break;
      case 24:  Serial.println("[2]");
        if (!timerRunning) {
          startTimer();
        }
        break;
      case 122: Serial.println("[3]");
        digitalWrite(LedPin, LOW);
        if (timerRunning) {
          stopTimer();
        break;

    }
  }
}
void toggleLED() {

  ledState = !ledState;
  digitalWrite(LedPin, ledState);
```



```
}
void stopTimer() {
  Timer1.stop();
  timerRunning = false;
}
void startTimer() {
  Timer1.start();
  timerRunning = true;
}
```

Main Code Explanation

```
#include <TimerOne.h>
#include <IRSendRev.h>
```

Introduce a timer library named `TimerOne`, and incorporate an infrared remote control library `IRSendRev.h`.

```
IR.Init(IR_PIN);
Timer1.attachInterrupt(toggleLED);
Timer1.stop();
```

Initialize the infrared remote control sensor, with `IR_PIN` being the pin connected to the sensor.
Initialize Timer1 to control the blinking of the LED, and stop the timer in its initial state.

```
if (IR.IsDta())
```

Use an if statement to check whether the return value of the function IR.IsDta() is true; if data is received, it returns true.

```
switch (dta[8]) {
      case 48:  Serial.println("[1]");
        digitalWrite(LedPin, HIGH);
        if (timerRunning) {
          stopTimer();
        }
        break;
      case 24:  Serial.println("[2]");
        if (!timerRunning) {
          startTimer();
        }
        break;
      case 122: Serial.println("[3]");
        digitalWrite(LedPin, LOW);
        if (timerRunning) {
          stopTimer();
        }
        break;
}
```

Upon receiving data, a switch statement is utilized to discern the type of data. If button 1 is pressed, the LED light will illuminate, and if the LED blinking timer is active, it will be paused. If button 2 is pressed, it will check whether the LED blinking timer is inactive; if so, it will be activated. If button 3 is pressed, the LED light will be turned off, and it will determine whether the LED blinking timer should continue to run, pausing it if necessary.

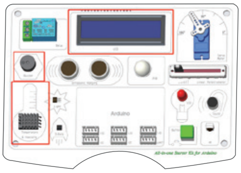
```
void toggleLED() {
  ledState = !ledState;
  digitalWrite(LedPin, ledState);
}
void stopTimer() {
  Timer1.stop();  timerRunning = false;
}
void startTimer() {
  Timer1.start();
  timerRunning = true;
}
```

The function's purpose is to set the state of the LED and to update the `ledState` flag to record the current status of the LED.
The `stopTimer()` function halts the timer and updates the status to `timerRunning`. The `timerRunning` variable is used to track the state of the timer.
The `startTimer()` function initiates the timer and updates the status to `timerRunning`. The `timerRunning` variable is used to track the state of the timer.

Lesson 14 - Weather Reminder

Introduction

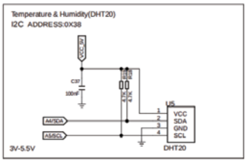
In this section of the course, we will explore how to adjust the information displayed on the LCD screen, the on/off state of the LED, and the alarm status of the buzzer based on the temperature and humidity data detected by the DHT20 sensor.

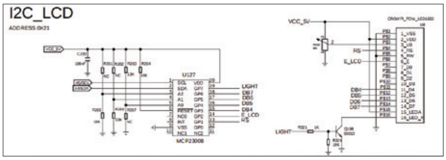


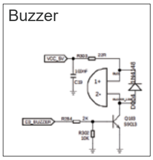
Hardware required

- All_in_one Starter Kit for Arduino x1
- USB Cable x1

Hardware schematic



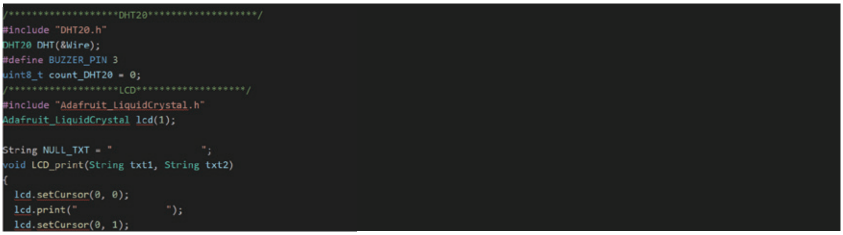




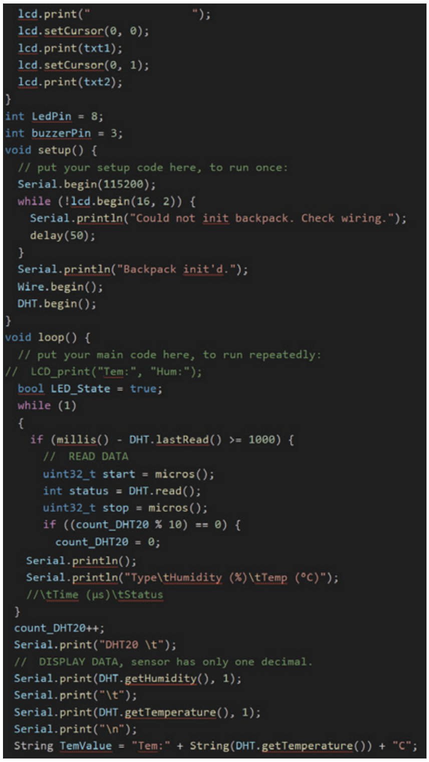
What will you see

The LCD will display temperature and humidity information. If the temperature exceeds 25 degrees Celsius, a red light will illuminate, and the LCD will display a "High Temperature" warning. If the temperature exceeds 30 degrees Celsius, the red light will flash, and the LCD will display a "Hot Temperature" warning. When the humidity falls below 40%, a buzzer will sound, and the LCD will display a "Dry Air" warning.

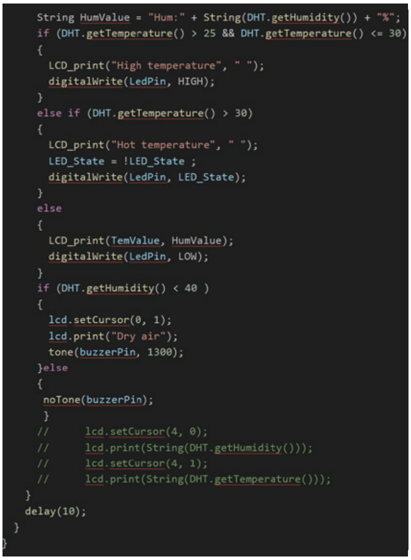
Complete Code



```
/*******************DHT20*******************/
#include "DHT20.h"
DHT20 DHT(&Wire);
#define BUZZER_PIN 3
uint8_t count_DHT20 = 0;
/*******************LCD*******************/
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);

String NULL_TXT = "                ";
void LCD_print(String txt1, String txt2)
{
  lcd.setCursor(0, 0);
  lcd.print("                ");
  lcd.setCursor(0, 1);
```



```
  lcd.print("                ");
  lcd.setCursor(0, 0);
  lcd.print(txt1);
  lcd.setCursor(0, 1);
  lcd.print(txt2);
}
int LedPin = 8;
int buzzerPin = 3;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  while (!lcd.begin(16, 2)) {
    Serial.println("Could not init backpack. Check wiring.");
    delay(50);
  }
  Serial.println("Backpack init'd.");
  Wire.begin();
  DHT.begin();
}
void loop() {
  // put your main code here, to run repeatedly:
//  LCD_print("Tem:", "Hum:");
  bool LED_State = true;
  while (1)
  {
    if (millis() - DHT.lastRead() >= 1000) {
      //  READ DATA
      uint32_t start = micros();
      int status = DHT.read();
      uint32_t stop = micros();
      if ((count_DHT20 % 10) == 0) {
        count_DHT20 = 0;
     Serial.println();
     Serial.println("Type\tHumidity (%)\tTemp (°C)");
     //\tTime (µs)\tStatus
   }
   count_DHT20++;
   Serial.print("DHT20 \t");
   //  DISPLAY DATA, sensor has only one decimal.
   Serial.print(DHT.getHumidity(), 1);
   Serial.print("\t");
   Serial.print(DHT.getTemperature(), 1);
   Serial.print("\n");
   String TemValue = "Tem:" + String(DHT.getTemperature()) + "C";
```



```
     String HumValue = "Hum:" + String(DHT.getHumidity()) + "%";
     if (DHT.getTemperature() > 25 && DHT.getTemperature() <= 30)
     {
       LCD_print("High temperature", " ");
       digitalWrite(LedPin, HIGH);
     }
     else if (DHT.getTemperature() > 30)
     {
       LCD_print("Hot temperature", " ");
       LED_State = !LED_State ;
       digitalWrite(LedPin, LED_State);
     }
     else
     {
       LCD_print(TemValue, HumValue);
       digitalWrite(LedPin, LOW);
     }
     if (DHT.getHumidity() < 40 )
     {
       lcd.setCursor(0, 1);
       lcd.print("Dry air");
       tone(buzzerPin, 1300);
     }else
     {
      noTone(buzzerPin);
      }
     //      lcd.setCursor(4, 0);
     //      lcd.print(String(DHT.getHumidity()));
     //      lcd.setCursor(4, 1);
     //      lcd.print(String(DHT.getTemperature()));
   }
   delay(10);
 }
}
```

Main Code Explanation

```
#include "DHT20.h"
DHT20 DHT(&Wire);
#define BUZZER_PIN 3
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);
```

To facilitate subsequent operations on these two types of sensors, we will import the DHT20 library and the LCD library. To enhance the readability of the code, we have defined a variable BUZZER_PIN to represent the buzzer's pin 3.

```
Wire.begin();
DHT.begin();
while (!lcd.begin(16, 2)) {
  Serial.println("Could not init backpack. Check wiring.");
  delay(50);
}
```

Initialize the DHT20 sensor and the LCD sensor. Please note that before the DHT20 can be initialized, the I2C bus needs to be initialized.

```
if (millis() - DHT.lastRead() >= 1000)
```

The `millis()` function subtracts the last time the DHT20 data was read by the `DHT.lastRead()` function from the current device's runtime to determine if a 1-second interval has been reached.

```
String TemValue = "Tem:" + String(DHT.getTemperature()) + "C";
String HumValue = "Hum:" + String(DHT.getHumidity()) + "%";
```

Temperature data is obtained through the `DHT.getTemperature()` function, and humidity data is acquired via the `DHT.getHumidity()` function. Subsequently, these values are concatenated into strings and assigned to the variables `TemValue` and `HumValue`. The temperature and humidity data are conveniently displayed on the LCD for easy reading.

```
if (DHT.getTemperature() > 25 && DHT.getTemperature() <= 30)
{
   LCD_print("High temperature", " ");
   digitalWrite(LedPin, HIGH);
}
```

The temperature is checked to see if it lies within the range of 25 to 30 degrees Celsius. If the temperature is above the upper limit, the LED light is turned on.

```
else if (DHT.getTemperature() > 30)
{
   LCD_print("Hot temperature", " ");
   LED_State = !LED_State ;
```

```
  digitalWrite(LedPin, LED_State);
}
```

The temperature is evaluated to determine if it exceeds 30 degrees Celsius. If it is higher, the LCD displays 'Hot' to indicate the temperature, and the LED light is inverted to achieve a blinking effect.

```
else
{
  LCD_print(TemValue, HumValue);
  digitalWrite(LedPin, LOW);
}
```

If the temperature does not fall into the aforementioned two scenarios, then the current temperature and humidity will be displayed on the LCD, and the LED light will be turned off.

```
if (DHT.getHumidity() < 40 )
    {
     lcd.setCursor(0, 1);
     lcd.print("Dry air");
     tone(buzzerPin, 1300);
    }else
    {
     noTone(buzzerPin);
 }
```

The `DHT.getHumidity()` function is used to determine if the current humidity is below 40%. If it is, the LCD displays 'Dry Air' and the buzzer sounds; otherwise, the buzzer is turned off.

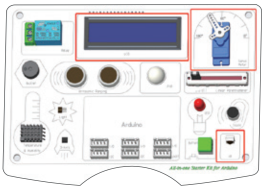
Lesson 15 - Servo Angle Control

Introduction

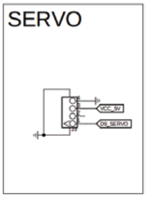
In this lesson, we will explore how to use an infrared remote control to adjust the rotation angle of a servo motor and display the angle in degrees on an LCD screen.

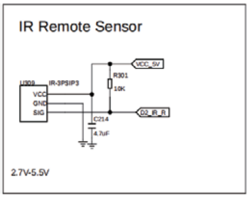
Hardware required

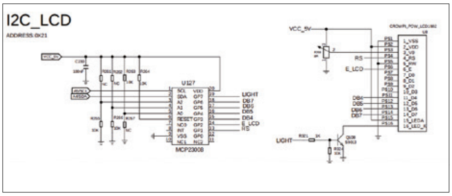
- All_in_one Starter Kit for Arduino x1
- USB Cable x1
- servo x 1
- Crowtail- IR Emitter x1



Hardware schematic



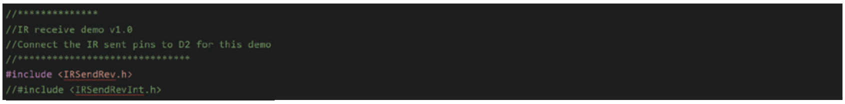




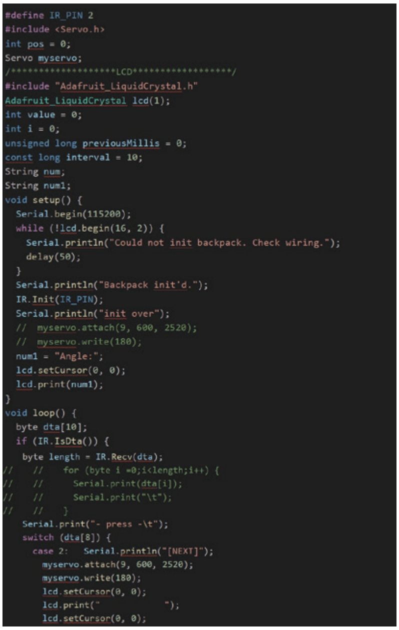
What will you see

You will observe that the LCD displays the corresponding degrees after the angle is input via the remote control. Upon pressing the confirm button, the servo motor rotates to the specified angle, and the current angle is displayed on the LCD screen. When the stop button is pressed, the servo motor ceases operation, and when the start button is pressed, the servo motor begins to function.

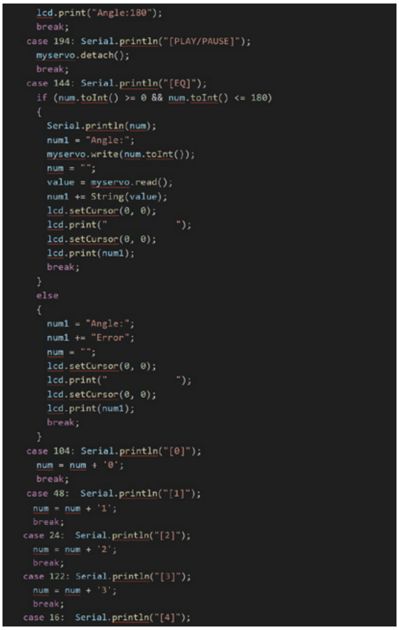
Complete Code



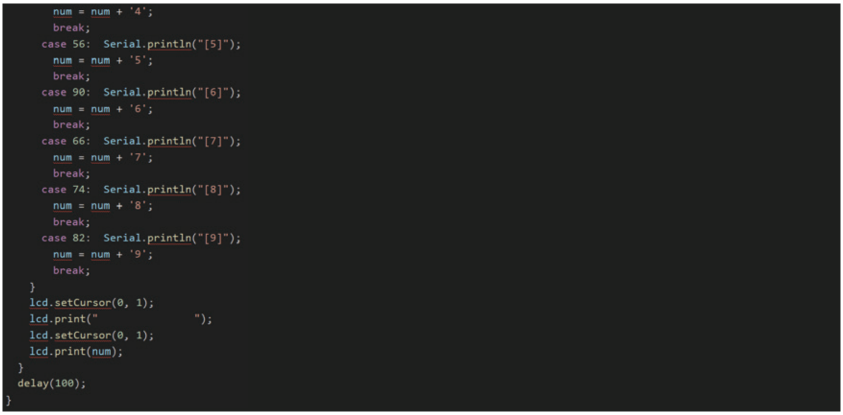
```
//**************
//IR receive demo v1.0
//Connect the IR sent pins to D2 for this demo
//******************************
#include <IRSendRev.h>
//#include <IRSendRevInt.h>
```



```
#define IR_PIN 2
#include <Servo.h>
int pos = 0;
Servo myservo;
/******************LCD*****************/
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);
int value = 0;
int i = 0;
unsigned long previousMillis = 0;
const long interval = 10;
String num;
String num1;
void setup() {
  Serial.begin(115200);
  while (!lcd.begin(16, 2)) {
    Serial.println("Could not init backpack. Check wiring.");
    delay(50);
  }
  Serial.println("Backpack init'd.");
  IR.Init(IR_PIN);
  Serial.println("init over");
  //  myservo.attach(9, 600, 2520);
  //  myservo.write(180);
  num1 = "Angle:";
  lcd.setCursor(0, 0);
  lcd.print(num1);
}
void loop() {
  byte dta[10];
  if (IR.IsDta()) {
   byte length = IR.Recv(dta);
//   //   for (byte i =0;i<length;i++) {
//   //     Serial.print(dta[i]);
//   //     Serial.print("\t");
//   //   }
   Serial.print("- press -\t");
   switch (dta[8]) {
     case 2:   Serial.println("[NEXT]");
       myservo.attach(9, 600, 2520);
       myservo.write(180);
       lcd.setCursor(0, 0);
       lcd.print("                ");
       lcd.setCursor(0, 0);
```



```
       lcd.print("Angle:180");
       break;
     case 194: Serial.println("[PLAY/PAUSE]");
       myservo.detach();
       break;
     case 144: Serial.println("[EQ]");
       if (num.toInt() >= 0 && num.toInt() <= 180)
       {
         Serial.println(num);
         num1 = "Angle:";
         myservo.write(num.toInt());
         num = "";
         value = myservo.read();
         num1 += String(value);
         lcd.setCursor(0, 0);
         lcd.print("                ");
         lcd.setCursor(0, 0);
         lcd.print(num1);
         break;
       }
       else
       {
         num1 = "Angle:";
         num1 += "Error";
         num = "";
         lcd.setCursor(0, 0);
         lcd.print("                ");
         lcd.setCursor(0, 0);
         lcd.print(num1);
         break;
       }
     case 104: Serial.println("[0]");
       num = num + '0';
       break;
     case 48:  Serial.println("[1]");
      num = num + '1';
      break;
    case 24:  Serial.println("[2]");
      num = num + '2';
      break;
    case 122: Serial.println("[3]");
      num = num + '3';
      break;
    case 16:  Serial.println("[4]");
```



```
        num = num + '4';
        break;
      case 56:  Serial.println("[5]");
        num = num + '5';
        break;
      case 90:  Serial.println("[6]");
        num = num + '6';
        break;
      case 66:  Serial.println("[7]");
        num = num + '7';
        break;
      case 74:  Serial.println("[8]");
        num = num + '8';
        break;
      case 82:  Serial.println("[9]");
        num = num + '9';
        break;
    }
    lcd.setCursor(0, 1);
    lcd.print("                ");
    lcd.setCursor(0, 1);
    lcd.print(num);
  }
  delay(100);
}
```

Main Code Explanation

```
#include <IRSendRev.h>
#include <Servo.h>
#include "Adafruit_LiquidCrystal.h"
#define IR_PIN 2
Servo myservo;
Adafruit_LiquidCrystal lcd(1);
```

For ease of subsequent operations, we have defined two class variables to control the servo and liquid crystal sensors, respectively, and have included the infrared remote control library, servo library, and LCD library. The infrared sensor is connected to the IR_PIN pin.

```
while (!lcd.begin(16, 2)) {
    Serial.println("Could not init backpack. Check wiring.");
    delay(50);
  }
IR.Init(IR_PIN);
```

Initialize the LCD and infrared remote control sensor.

```
if (IR.IsDta())
```

Check for the arrival of an infrared signal.

```
byte length = IR.Recv(dta); switch (dta[8])
```

If an infrared signal is received, save the information and then use a switch statement to determine the type of infrared signal.

```
case 2:   Serial.println("[NEXT]");
        myservo.attach(9, 600, 2520);
        myservo.write(180);
        lcd.setCursor(0, 0);
        lcd.print("             ");
        lcd.setCursor(0, 0);
        lcd.print("Angle:180");
        break;
      case 194: Serial.println("[PLAY/PAUSE]");
        myservo.detach();
        break;
      case 144: Serial.println("[EQ]");
        if (num.toInt() >= 0 && num.toInt() <= 180)
        {
          Serial.println(num);
          num1 = "Angle:";
          myservo.write(num.toInt());
num = "";
          value = myservo.read();
          num1 += String(value);
          lcd.setCursor(0, 0);
          lcd.print("             ");
          lcd.setCursor(0, 0);
          lcd.print(num1);
          break;
        }
        else
        {
          num1 = "Angle:";
          num1 += "Error";
          num = "";
          lcd.setCursor(0, 0);
          lcd.print("             ");
          lcd.setCursor(0, 0);
          lcd.print(num1);
          break;
        }
```

case 2: Represents pressing the NEXT button; if this button is pressed, the motor is turned on.

case 194: Represents pressing the PLAY/PAUSE button; if this button is pressed, the motor is turned off.
case 144: Represents pressing the EQ button; if this button is pressed, it indicates confirmation to rotate to the specified angle; if the angle is invalid, an error is displayed on the LCD.

```
case 104: Serial.println("[0]");
        num = num + '0';
        break;
      case 48:  Serial.println("[1]");
        num = num + '1';
        break;
      case 24:  Serial.println("[2]");
        num = num + '2';
        break;
      case 122: Serial.println("[3]");
        num = num + '3';
        break;
      case 16:  Serial.println("[4]");
        num = num + '4';
        break;
      case 56:  Serial.println("[5]");
        num = num + '5';
        break;
      case 90:  Serial.println("[6]");
        num = num + '6';
        break;
      case 66:  Serial.println("[7]");
        num = num + '7';
        break;
      case 74:  Serial.println("[8]");
        num = num + '8';
        break;
      case 82:  Serial.println("[9]");
        num = num + '9';
        break;
```

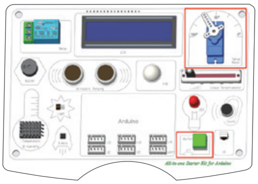
The code segment indicates that when the numeric buttons 0-9 are pressed, the pressed numbers are concatenated into a string to facilitate the transmission of rotation angles to the motor.

Lesson 16 - Polite Automatic Door

Introduction

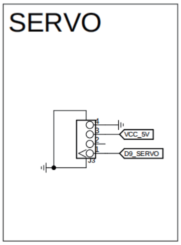
In this lesson, we will learn about advanced operations of servo motors. We will control the rotation of the servo motor using buttons to achieve the effect of opening the door when the button is pressed and closing the door when the button is released.

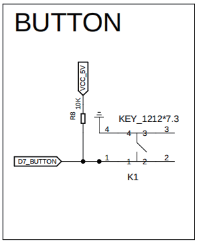
Hardware required



- All_in_one Starter Kit for Arduino x1
- USB Cable x1
- servo x 1

Hardware schematic

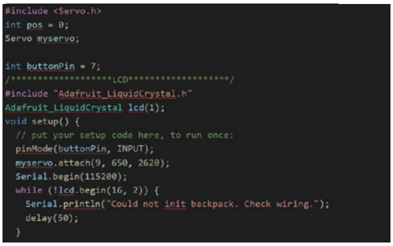




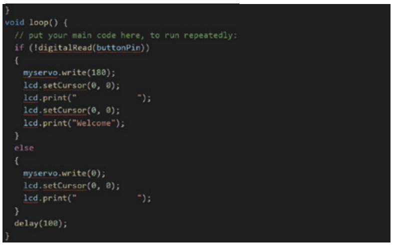
What will you see

We will observe that when the Button button is pressed, the servo motor rotates 180 degrees to indicate that the gate is open, and when the button is released, the servo motor rotates back 180 degrees to indicate that the gate is closed.

Complete Code



```
#include <Servo.h>
int pos = 0;
Servo myservo;

int buttonPin = 7;
/*******************LCD*******************/
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);
void setup() {
  // put your setup code here, to run once:
  pinMode(buttonPin, INPUT);
  myservo.attach(9, 650, 2620);
  Serial.begin(115200);
  while (!lcd.begin(16, 2)) {
    Serial.println("Could not init backpack. Check wiring.");
    delay(50);
  }
```



```
}
void loop() {
  // put your main code here, to run repeatedly:
  if (!digitalRead(buttonPin))
  {
    myservo.write(180);
    lcd.setCursor(0, 0);
    lcd.print("                ");
    lcd.setCursor(0, 0);
    lcd.print("Welcome");
  }
  else
  {
    myservo.write(0);
    lcd.setCursor(0, 0);
    lcd.print("                ");
  }
  delay(100);
}
```

Main Code Explanation

```
#include <Servo.h>
Servo myservo;
int buttonPin = 7;
/******************LCD******************/
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);
```

Firstly, the libraries servo.h and Adafruit_LiquidCrystal.h are included. Subsequently, a variable named myservo is declared for the subsequent control of the servo motor. Similarly, a variable named lcd is defined for operating the LCD display. Lastly, a variable named buttonPin is established to detect the state of the button.

```
myservo.attach(9, 650, 2620);
```

To initialize and calibrate the servo motor, the first parameter specifies the pin to which the motor is connected, the second parameter indicates the minimum pulse width, and the third parameter denotes the maximum pulse width.

```
 while (!lcd.begin(16, 2)) {
    Serial.println("Could not init backpack. Check wiring.");
    delay(50);
  }
```

Initialize the LCD, which is a necessary operation for using the LCD display.

```
if (!digitalRead(buttonPin))
```

Use an `if` statement to determine whether the button has been pressed.

```
  {
    myservo.write(180);
    lcd.setCursor(0, 0);
    lcd.print("                ");
    lcd.setCursor(0, 0);
    lcd.print("Welcome");
  }
```

If the button is pressed, then rotate the motor by 180 degrees and display "Welcome" on the LCD.

```
  else
  {
    myservo.write(0);
    lcd.setCursor(0, 0);
    lcd.print("                ");

  }
```

If The Button Is Not Pressed, The Motor Should Rotate To 0 Degrees And The LCD Should Be Cleared.

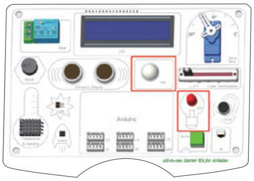
Lesson 17 - PIR Control Light

Introduction

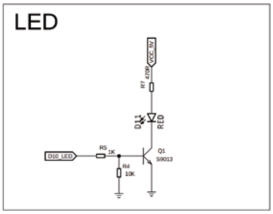
In this lesson, we will delve into advanced operations with PIR motion sensors. We will learn how to control the illumination of an LED based on the detection from the PIR motion sensor.

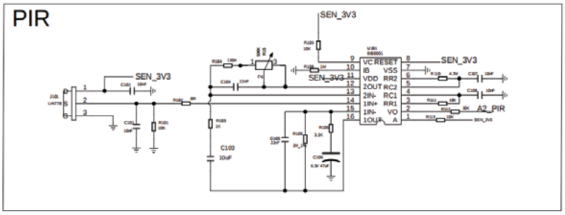
Hardware required

- All_in_one Starter Kit for Arduino x1
- USB Cable x1



Hardware schematic

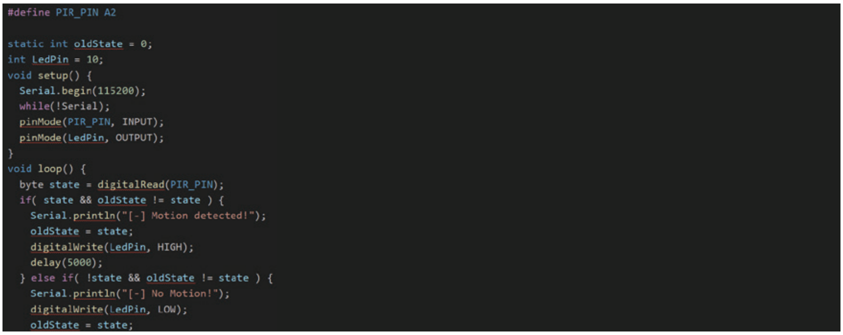




What will you see

You will observe that when you wave or walk within the range of the PIR motion sensor, the LED will light up for 5 seconds. If there is no movement detected within the sensing range of the PIR motion sensor, the LED will not illuminate. If this process does not change as described, please check to ensure that the program has been successfully uploaded to the device.

Complete Code



```
#define PIR_PIN A2

static int oldState = 0;
int LedPin = 10;
void setup() {
  Serial.begin(115200);
  while(!Serial);
  pinMode(PIR_PIN, INPUT);
  pinMode(LedPin, OUTPUT);
}
void loop() {
  byte state = digitalRead(PIR_PIN);
  if( state && oldState != state ) {
    Serial.println("[-] Motion detected!");
    oldState = state;
    digitalWrite(LedPin, HIGH);
    delay(5000);
  } else if( !state && oldState != state ) {
    Serial.println("[-] No Motion!");
    digitalWrite(LedPin, LOW);
    oldState = state;
```

```
    }
    delay(20);
  }
```

Main Code Explanation

```
#define PIR_PIN A2
static int oldState = 0;
int LedPin = 10;
```

Before we begin using the sensor, it is necessary to define some variables that will facilitate our subsequent operations. We will use a macro definition to assign PIR_PIN to represent pin A2, which makes it more understandable. We also define a variable named oldState to record the state of the PIR sensor before any change. Additionally, we define a variable LedPin and assign it pin number 10.

```
pinMode(PIR_PIN, INPUT);
pinMode(LedPin, OUTPUT);
```

Initialize the PIR sensor and the LED pin. Set the PIR_PIN to input mode. Assign the LedPin to output mode. The assigned modes are related to their subsequent use.

```
byte state = digitalRead(PIR_PIN);
  if( state && oldState != state ) {
    Serial.println("[-] Motion detected!");
    oldState = state;
    digitalWrite(LedPin, HIGH);
    delay(5000);
  } else if( !state && oldState != state ) {
    Serial.println("[-] No Motion!");
    digitalWrite(LedPin, LOW);
    oldState = state;
  }
```

Define a variable `state` to read the level of the PIR sensor. If the current level is high and the previous state was not high, then record the current state to `oldState`, turn on the LED light, and delay for 5 seconds. If the current level is low and the previous state was not equal to the current level, then turn off the LED light and record the current state of the PIR sensor.

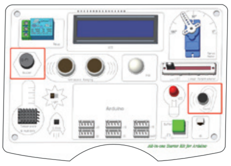
Lesson 18 - Sound Reminder

Introduction

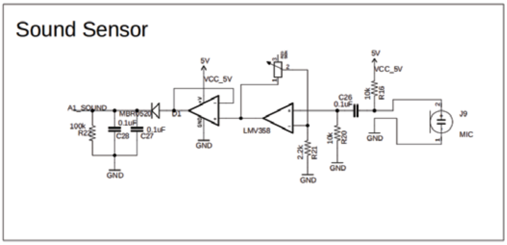
In this lesson, we will explore advanced operations with a sound sensor. We will use the sound sensor to detect the ambient noise level. When the sensor detects sound, it will trigger a buzzer to provide an audible alert.

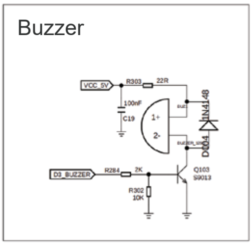
Hardware required

- All_in_one Starter Kit for Arduino x1
- USB Cable x1



Hardware schematic

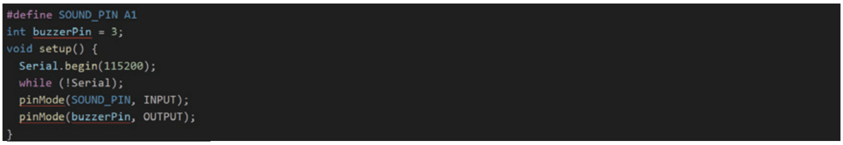




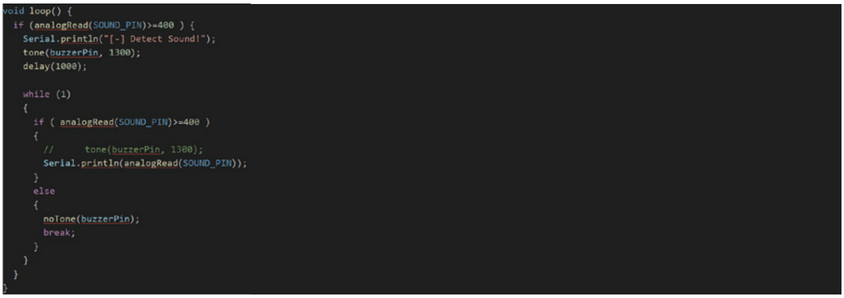
What will you see

You will observe that if the surrounding noise is too loud or you shout at the sound sensor, the buzzer will emit a one-second beep to remind you to lower the volume. If the noise remains high, the buzzer will continue to emit beeps. If there is no sound, the buzzer will cease to produce beeps.

Complete Code



```
#define SOUND_PIN A1
int buzzerPin = 3;
void setup() {
  Serial.begin(115200);
  while (!Serial);
  pinMode(SOUND_PIN, INPUT);
  pinMode(buzzerPin, OUTPUT);
}
```



```
void loop() {
  if (analogRead(SOUND_PIN)>=400 ) {
    Serial.println("[-] Detect Sound!");
    tone(buzzerPin, 1300);
    delay(1000);

    while (1)
    {
      if ( analogRead(SOUND_PIN)>=400 )
      {
        //      tone(buzzerPin, 1300);
        Serial.println(analogRead(SOUND_PIN));
      }
      else
      {
        noTone(buzzerPin);
        break;
      }
    }
  }
}
```

Main Code Explanation

```
#define SOUND_PIN A1
int buzzerPin = 3;
```

Define a macro named `SOUND_PIN` to represent pin A1 for better understanding. Assign pin number 3 to the variable `buzzerPin` for the buzzer.

```
pinMode(SOUND_PIN, INPUT);
pinMode(buzzerPin, OUTPUT);
```

Initialize the mode of the sound sensor and buzzer pins. Configure the `SOUND_PIN` pin as an input mode, which outputs a high level when sound is detected. Assign the `buzzerPin` pin as an output mode for controlling the buzzer.

```
if (analogRead(SOUND_PIN)>=400 ) {
   Serial.println("[-] Detect Sound!");
   tone(buzzerPin, 1300);
   delay(1000);
while (1)
   {
     if ( analogRead(SOUND_PIN)>=400 )
     {
       //      tone(buzzerPin, 1300);
       Serial.println(analogRead(SOUND_PIN));
     }
     else
     {
       noTone(buzzerPin);
       break;
     }
   }
}
```

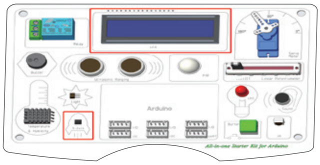
When the `analogRead` function detects that the analog value output by the `SOUND_PIN` pin exceeds 400, use the `tone` function to sound the buzzer. Continuously monitor whether the analog value of the `SOUND_PIN` pin falls below 400; if the output analog value is below 400, it indicates that the detected sound level has not reached the threshold, and the buzzer should be turned off and the loop should exit. If the sound level is consistently above 400, the buzzer should continue to sound.

Lesson 19 - Calculation of Acceleration

Introduction

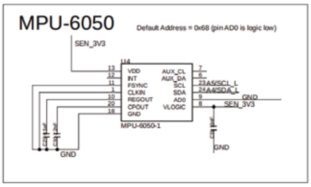
In this lesson, we will learn how to operate the MPU6050 sensor. By moving the board, the MPU6050 sensor will detect and calculate different accelerations in various directions.

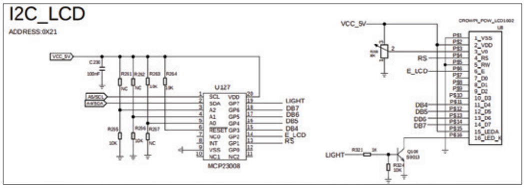
Hardware required



- All_in_one Starter Kit for Arduino x1
- USB Cable x1

Hardware schematic

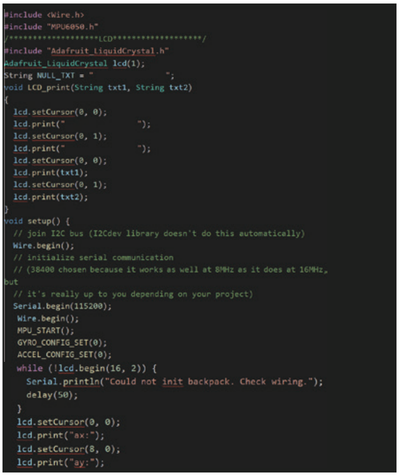




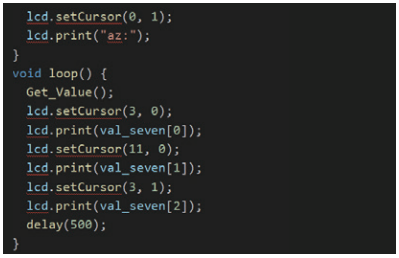
What will you see

When you run the program, you will see the LCD display the accelerometer values for the X, Y, and Z axes. As you quickly move the accelerometer along the axis, you will observe the accelerometer values for that axis change accordingly.

Complete Code



```
#include <Wire.h>
#include "MPU6050.h"
/*******************LCD*******************/
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);
String NULL_TXT = "                ";
void LCD_print(String txt1, String txt2)
{
  lcd.setCursor(0, 0);
  lcd.print("                ");
  lcd.setCursor(0, 1);
  lcd.print("                ");
  lcd.setCursor(0, 0);
  lcd.print(txt1);
  lcd.setCursor(0, 1);
  lcd.print(txt2);
}
void setup() {
  // join I2C bus (I2Cdev library doesn't do this automatically)
  Wire.begin();
  // initialize serial communication
  // (38400 chosen because it works as well at 8MHz as it does at 16MHz,
but
  // it's really up to you depending on your project)
  Serial.begin(115200);
   Wire.begin();
   MPU_START();
   GYRO_CONFIG_SET(0);
   ACCEL_CONFIG_SET(0);
   while (!lcd.begin(16, 2)) {
     Serial.println("Could not init backpack. Check wiring.");
     delay(50);
   }
   lcd.setCursor(0, 0);
   lcd.print("ax:");
   lcd.setCursor(8, 0);
   lcd.print("ay:");
```



```
   lcd.setCursor(0, 1);
   lcd.print("az:");
}
void loop() {
   Get_Value();
   lcd.setCursor(3, 0);
   lcd.print(val_seven[0]);
   lcd.setCursor(11, 0);
   lcd.print(val_seven[1]);
   lcd.setCursor(3, 1);
   lcd.print(val_seven[2]);
   delay(500);
}
```

Main Code Explanation

```
#include <Wire.h>
#include "MPU6050.h"
/******************LCD******************/
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);
```

Introduce the relevant libraries: `MPU6050.h` is used for operating the MPU6050 sensor, and `Adafruit_LiquidCrystal.h` is used for operating the LCD. Both communicate via I2C, so you also need to include `Wire.h`.

```
Wire.begin();
MPU_START();
GYRO_CONFIG_SET(0);
ACCEL_CONFIG_SET(0);
while (!lcd.begin(16, 2)) {
   Serial.println("Could not init backpack. Check wiring.");
   delay(50);
}
```

After initializing the I2C interface with Wire.begin(), proceed to initialize the MPU6050, followed by activating it with MPU_START(). Then, configure the device's gyroscope sensitivity using GYRO_CONFIG_SET(0), and set the accelerometer sensitivity with ACCEL_CONFIG_SET(0). Finally, complete the initialization of the LCD.

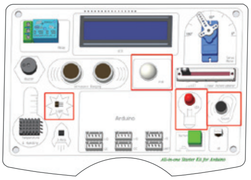
```
Get_Value();
lcd.setCursor(3, 0);
lcd.print(val_seven[0]);
lcd.setCursor(11, 0);
lcd.print(val_seven[1]);
lcd.setCursor(3, 1);
lcd.print(val_seven[2]);
```

The acceleration values are obtained through Get_Value() and are ensured to be stored in the global array variable val_seven. Subsequently, the cursor position is set using lcd.setCursor, and the corresponding values are displayed with lcd.print. The displayed ax represents the acceleration along the x-axis, ay represents the acceleration along the y-axis, and az represents the acceleration along the z-axis.

Lesson 20 - Smart Corridor Light

Introduction

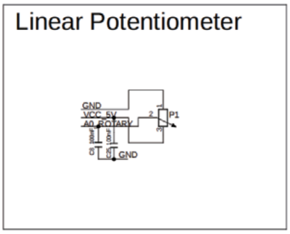
In this lesson, we will learn about the advanced operations of linking the Sound Sensor, PIR, Light Sensor, and LED sensors. By coordinating between the sensors, we will achieve the effect of a smart corridor light.

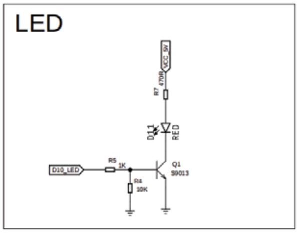


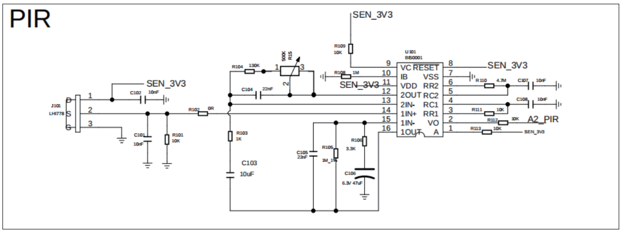
Hardware required

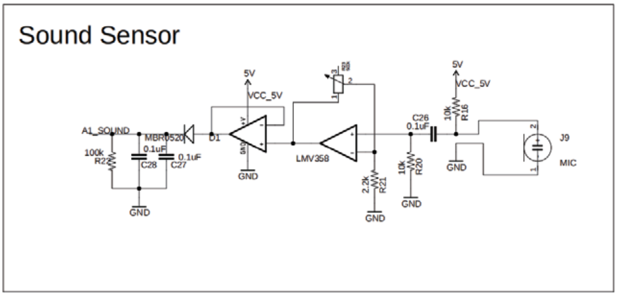
• All_in_one Starter Kit for Arduino x1
• USB Cable x1

Hardware schematic





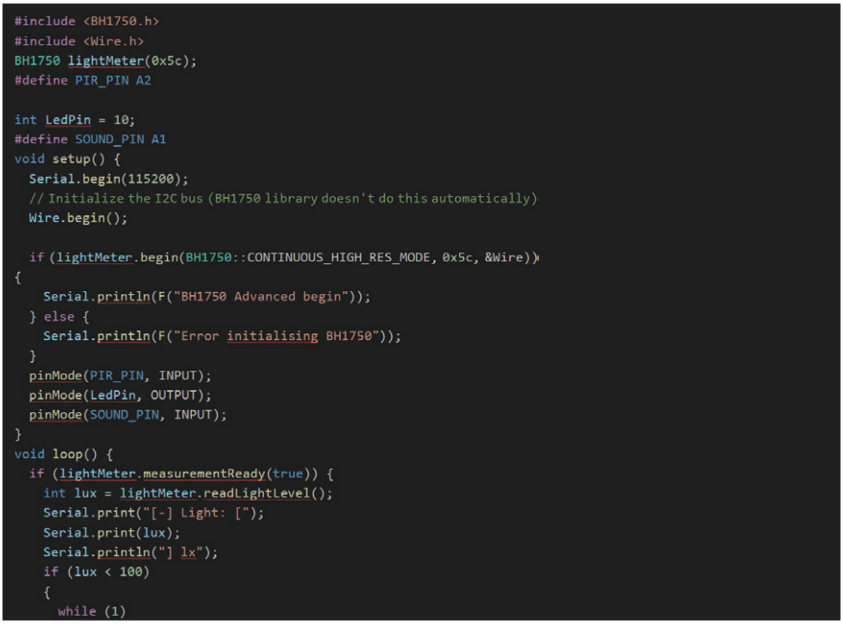




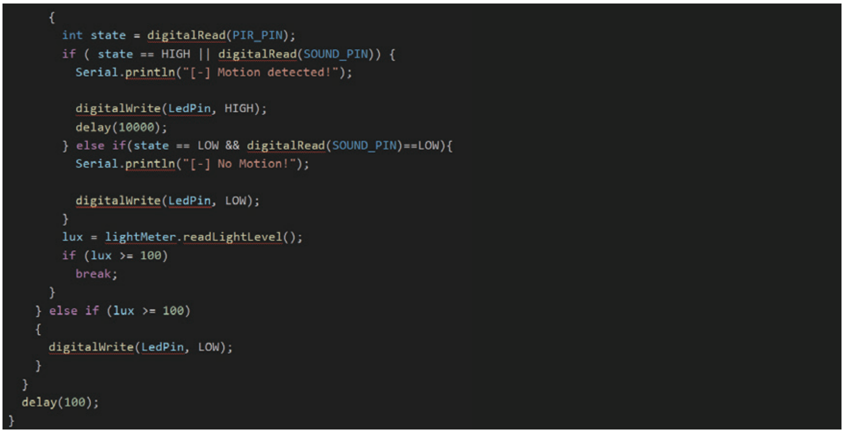
What will you see

When the ambient light is very strong, the LED will not light up regardless of movement or sound. When the ambient light is dim or the light sensor is blocked by hand, the LED is set to be off by default. At this point, if there is movement or sound, the LED will light up for 10 seconds and then turn off. If there is continuous movement or continuous sound, the LED will remain on until there is no activity, after which it will stay on for another 10 seconds before turning off.

Complete Code



```
#include <BH1750.h>
#include <Wire.h>
BH1750 lightMeter(0x5c);
#define PIR_PIN A2

int LedPin = 10;
#define SOUND_PIN A1
void setup() {
  Serial.begin(115200);
  // Initialize the I2C bus (BH1750 library doesn't do this automatically)
  Wire.begin();

  if (lightMeter.begin(BH1750::CONTINUOUS_HIGH_RES_MODE, 0x5c, &Wire))
{
    Serial.println(F("BH1750 Advanced begin"));
  } else {
    Serial.println(F("Error initialising BH1750"));
  }
  pinMode(PIR_PIN, INPUT);
  pinMode(LedPin, OUTPUT);
  pinMode(SOUND_PIN, INPUT);
}
void loop() {
  if (lightMeter.measurementReady(true)) {
    int lux = lightMeter.readLightLevel();
    Serial.print("[-] Light: [");
    Serial.print(lux);
    Serial.println("] lx");
    if (lux < 100)
    {
      while (1)
```



```
      {
        int state = digitalRead(PIR_PIN);
        if ( state == HIGH || digitalRead(SOUND_PIN)) {
          Serial.println("[-] Motion detected!");

          digitalWrite(LedPin, HIGH);
          delay(10000);
        } else if(state == LOW && digitalRead(SOUND_PIN)==LOW){
          Serial.println("[-] No Motion!");

          digitalWrite(LedPin, LOW);
        }
        lux = lightMeter.readLightLevel();
        if (lux >= 100)
          break;
      }
    } else if (lux >= 100)
    {
      digitalWrite(LedPin, LOW);
    }
  }
  delay(100);
}
```

Main Code Explanation

```
#include <BH1750.h>
#include <Wire.h>
BH1750 lightMeter(0x5c);
#define PIR_PIN A2
int LedPin = 10;
#define SOUND_PIN A1
```

Before commencing the utilization of sensors, it is imperative to include the relevant library files. Initially, to employ the light sensor, we must incorporate the BH1750.h library. Given that this photosensitive sensor communicates via the I2C protocol, it is also necessary to include the Wire.h library. Once the library files are incorporated, we can proceed to define variables that will facilitate subsequent sensor operations. Specifically, we need to declare a variable named `lightMeter` and assign it the address `0x5c`. Furthermore, we utilize a macro definition `PIR_PIN` to denote pin A2, and we define a variable named `LedPin`, assigning it to pin 10. Similarly, we employ a macro definition `SOUND_PIN` to represent pin A1.

```
Wire.begin();
if (lightMeter.begin(BH1750::CONTINUOUS_HIGH_RES_MODE, 0x5c, &Wire)) {
  Serial.println(F("BH1750 Advanced begin"));
} else {
  Serial.println(F("Error initialising BH1750"));
}
pinMode(PIR_PIN, INPUT);
pinMode(LedPin, OUTPUT);
pinMode(SOUND_PIN, INPUT);
```

After initializing the I2C with Wire.begin(), you can then initialize the light sensor with: lightMeter.begin(BH1750::CONTINUOUS_HIGH_RES_MODE, 0x5c, &Wire)

When initializing the modes for various pins, where INPUT represents input mode and OUTPUT represents output mode.

```
if (lightMeter.measurementReady(true))
```

Use an if statement to determine whether data can be obtained from the light sensor.

```
int lux = lightMeter.readLightLevel();
```

Store the acquired brightness data in a variable named `lux` for subsequent processing.

```
if (lux < 100)
```

When the brightness data is less than 100, it indicates that the natural light is very dim, necessitating the activation of the PIR and Sound sensors to determine if there is any movement or sound indicative of a person passing by.

```
int state = digitalRead(PIR_PIN);
```

```
if ( state == HIGH || digitalRead(SOUND_PIN))
{
digitalWrite(LedPin, HIGH);
delay(10000);
}
```

Read the PIR sensor level and the sound sensor level; when there is movement or sound detected, the LED light will turn on and a delay of 10 seconds will be implemented.

```
else if(state == LOW && digitalRead(SOUND_PIN)==LOW){
     Serial.println("[-] No Motion!");
     digitalWrite(LedPin, LOW);
    }
```

When the PIR sensor and Sound sensor both read a low level, turn off the LED.

```
lux = lightMeter.readLightLevel();
if (lux >= 100)
break;
```

When reading brightness data, if the brightness is greater than or equal to 100, it indicates that the natural light is already bright enough. In this case, the loop will be exited, and no further detection will be performed using the PIR sensor and Sound sensor.

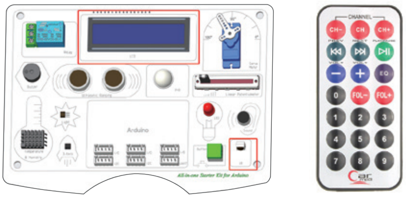
Lesson 21 - Simple Calculator

Introduction

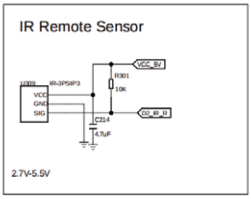
In this lesson, we will explore advanced methods of using IR remote controls. We will learn how to input numerical values and expressions via an IR remote control and display them on the screen. After pressing the confirmation button, the system will calculate the result and display it on the screen.

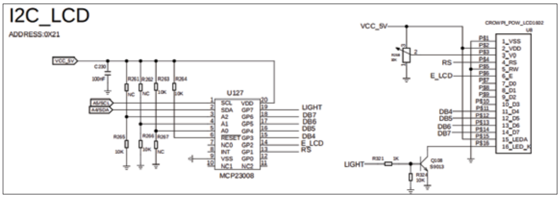
Hardware required

- All_in_one Starter Kit for Arduino x1
- USB Cable x1
- Crowtail-IR Emitter x1



Hardware schematic



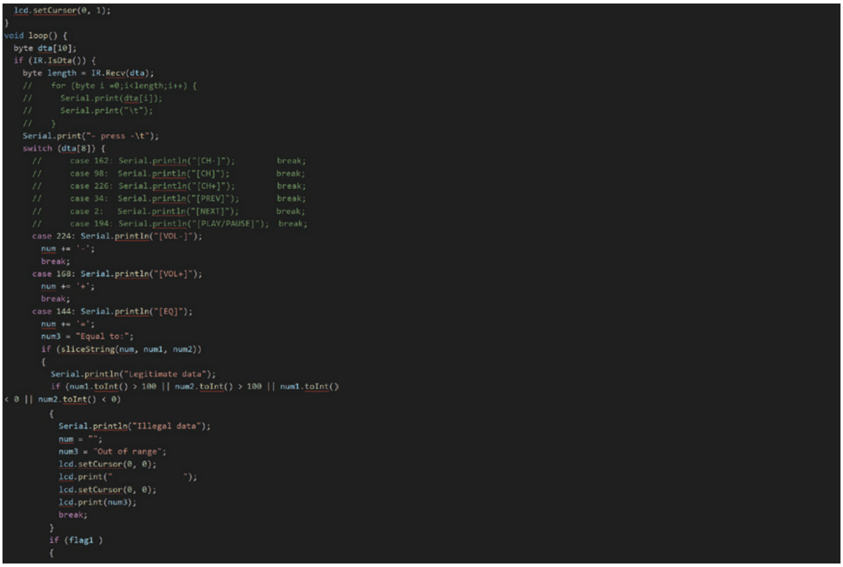


What will you see

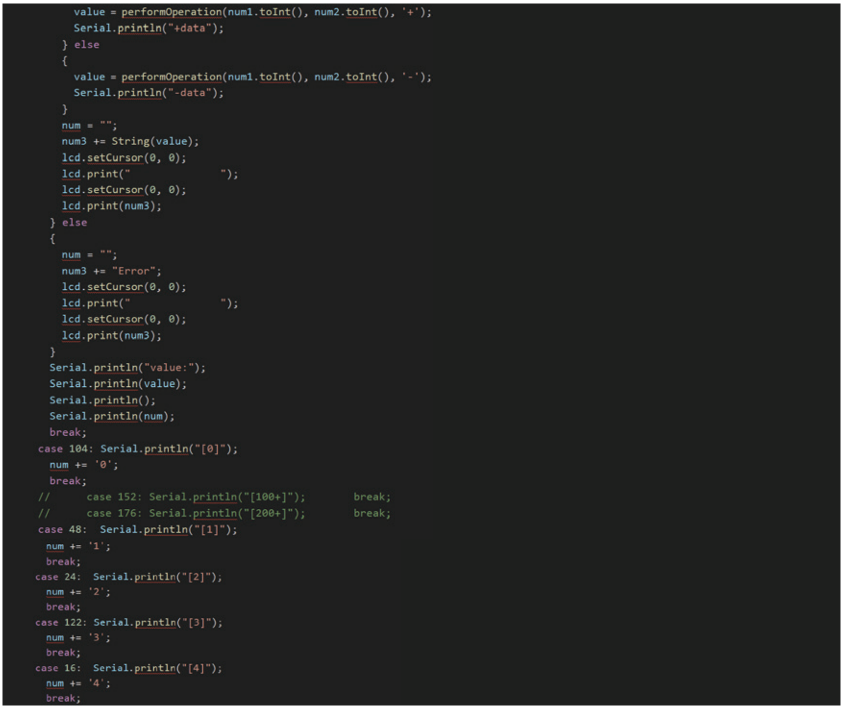
An IR remote control serves as the input device, while an LCD acts as the calculator display. Five non-numeric buttons on the remote control are designated as: "＋, -". Upon running the program, the LCD displays "Please enter:", prompting the user to input numerical values, which are then synchronized and displayed on the LCD in real-time. After inputting the values, pressing the equals sign causes the LCD to display the result. If the calculation exceeds the range, the LCD shows "Out of calculation range"; if the input format is incorrect, it displays "Error".

Complete Code

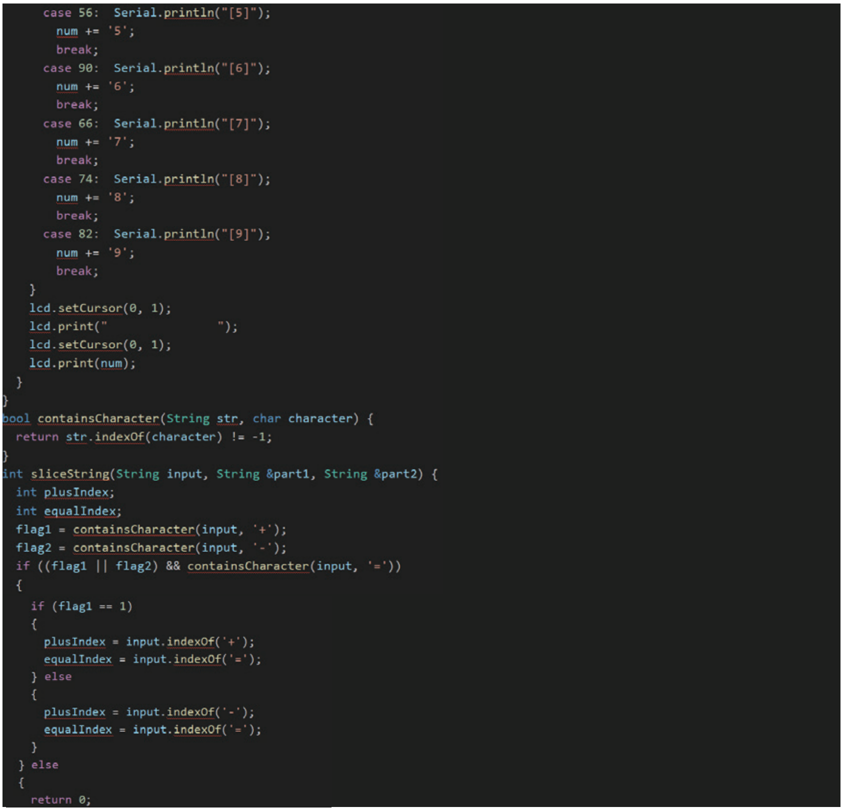
```
//**************
//IR receive demo v1.0
//Connect the IR sent pins to D2 for this demo
//******************************
#include <IRSendRev.h>
//#include <IRSendRevInt.h>

#define IR_PIN 2
/*******************LCD*******************/
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);
String num;
String num1;
String num2;
String num3;
int flag1 = 0;
int flag2 = 0;
int value = 0;
void setup() {
  Serial.begin(115200);
  while (!Serial);
  IR.Init(IR_PIN);
  Serial.println("init over");
  while (!lcd.begin(16, 2)) {
    Serial.println("Could not init backpack. Check wiring.");
    delay(50);
  }
  lcd.setCursor(0, 0);
  lcd.print("Equal to:");
```



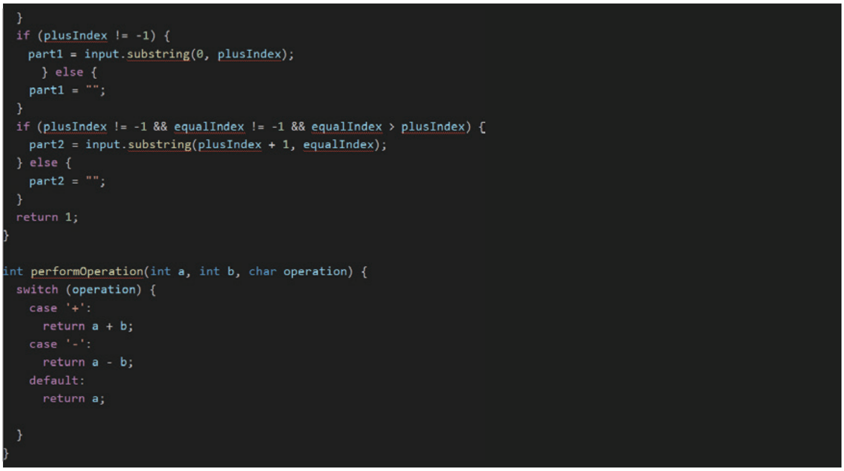
```
  lcd.setCursor(0, 1);
}
void loop() {
  byte dta[10];
  if (IR.IsDta()) {
    byte length = IR.Recv(dta);
    //    for (byte i =0;i<length;i++) {
    //      Serial.print(dta[i]);
    //      Serial.print("\t");
    //    }
    Serial.print("- press -\t");
    switch (dta[8]) {
      //      case 162: Serial.println("[CH-]");          break;
      //      case 98:  Serial.println("[CH]");           break;
      //      case 226: Serial.println("[CH+]");          break;
      //      case 34:  Serial.println("[PREV]");         break;
      //      case 2:   Serial.println("[NEXT]");         break;
      //      case 194: Serial.println("[PLAY/PAUSE]");  break;
      case 224: Serial.println("[VOL-]");
        num += '-';
        break;
      case 168: Serial.println("[VOL+]");
        num += '+';
        break;
      case 144: Serial.println("[EQ]");
        num += '=';
        num3 = "Equal to:";
        if (sliceString(num, num1, num2))
        {
          Serial.println("Legitimate data");
          if (num1.toInt() > 100 || num2.toInt() > 100 || num1.toInt()
< 0 || num2.toInt() < 0)
          {
            Serial.println("Illegal data");
            num = "";
            num3 = "Out of range";
            lcd.setCursor(0, 0);
            lcd.print("                ");
            lcd.setCursor(0, 0);
            lcd.print(num3);
            break;
          }
          if (flag1 )
          {
```



```
            value = performOperation(num1.toInt(), num2.toInt(), '+');
            Serial.println("+data");
          } else
          {
            value = performOperation(num1.toInt(), num2.toInt(), '-');
            Serial.println("-data");
          }
          num = "";
          num3 += String(value);
          lcd.setCursor(0, 0);
          lcd.print("                ");
          lcd.setCursor(0, 0);
          lcd.print(num3);
        } else
        {
          num = "";
          num3 += "Error";
          lcd.setCursor(0, 0);
          lcd.print("                ");
          lcd.setCursor(0, 0);
          lcd.print(num3);
        }
        Serial.println("value:");
        Serial.println(value);
        Serial.println();
        Serial.println(num);
        break;
      case 104: Serial.println("[0]");
        num += '0';
        break;
      //      case 152: Serial.println("[100+]");        break;
      //      case 176: Serial.println("[200+]");        break;
      case 48:  Serial.println("[1]");
        num += '1';
        break;
      case 24:  Serial.println("[2]");
        num += '2';
        break;
      case 122: Serial.println("[3]");
        num += '3';
        break;
      case 16:  Serial.println("[4]");
        num += '4';
        break;
```



```
      case 56:  Serial.println("[5]");
        num += '5';
        break;
      case 90:  Serial.println("[6]");
        num += '6';
        break;
      case 66:  Serial.println("[7]");
        num += '7';
        break;
      case 74:  Serial.println("[8]");
        num += '8';
        break;
      case 82:  Serial.println("[9]");
        num += '9';
        break;
    }
    lcd.setCursor(0, 1);
    lcd.print("                ");
    lcd.setCursor(0, 1);
    lcd.print(num);
  }
}
bool containsCharacter(String str, char character) {
  return str.indexOf(character) != -1;
}
int sliceString(String input, String &part1, String &part2) {
  int plusIndex;
  int equalIndex;
  flag1 = containsCharacter(input, '+');
  flag2 = containsCharacter(input, '-');
  if ((flag1 || flag2) && containsCharacter(input, '='))
  {
    if (flag1 == 1)
    {
      plusIndex = input.indexOf('+');
      equalIndex = input.indexOf('=');
    } else
    {
      plusIndex = input.indexOf('-');
      equalIndex = input.indexOf('=');
    }
  } else
  {
    return 0;
```



```
  }
  if (plusIndex != -1) {
    part1 = input.substring(0, plusIndex);
      } else {
    part1 = "";
  }
  if (plusIndex != -1 && equalIndex != -1 && equalIndex > plusIndex) {
    part2 = input.substring(plusIndex + 1, equalIndex);
  } else {
    part2 = "";
  }
  return 1;
}

int performOperation(int a, int b, char operation) {
  switch (operation) {
    case '+':
      return a + b;
    case '-':
      return a - b;
    default:
      return a;

  }
}
```

Main Code Explanation

```
#include <IRSendRev.h>
//#include <IRSendRevInt.h>
#define IR_PIN 2
/******************LCD******************/
#include "Adafruit_LiquidCrystal.h"
Adafruit_LiquidCrystal lcd(1);
```

Before utilizing the LCD and IR sensor, we must first include the IRSendRev.h and Adafruit_LiquidCrystal.h libraries. To enhance code readability, we use a macro definition IR_PIN to represent pin 2. We define the lcd variable for subsequent operations on the LCD.

```
IR.Init(IR_PIN);
  Serial.println("init over");
  while (!lcd.begin(16, 2)) {
    Serial.println("Could not init backpack. Check wiring.");
    delay(50);
  }
```

Initialization IR and LCD

```
  byte dta[10];
  if (IR.IsDta())
```

Define an array `dta` for storing received data. Use an if statement to check if there is any infrared data to be received; if there is data, the function `IR.IsDta()` will return true.

```
byte length = IR.Recv(dta);
    Serial.print("- press -\t");
switch (dta[8])
```

Utilize the `IR.Recv(dta)` function to read data sent by the IR transmitter, and then employ a switch statement to determine the type of data received.

```
ase 224: Serial.println("[VOL-]");
        num += '-';
        break;
 case 168: Serial.println("[VOL+]");
        num += '+';
        break;
case 104: Serial.println("[0]");
        num += '0';
        break;
case 48:  Serial.println("[1]");
        num += '1';
break;
```

```
case 24:  Serial.println("[2]");
        num += '2';
        break;
case 122: Serial.println("[3]");
        num += '3';
        break;
case 16:  Serial.println("[4]");
        num += '4';
        break;
case 56:  Serial.println("[5]");
        num += '5';
        break;
case 90:  Serial.println("[6]");
        num += '6';
        break;
case 66:  Serial.println("[7]");
        num += '7';
        break;
case 74:  Serial.println("[8]");
        num += '8';
        break;
case 82:  Serial.println("[9]");
        num += '9';
        break;
```

The selection statement above is used to identify the data received from the IR transmission. Once the keys 0-9 and the symbols '+“and”-' are recognized, these characters are appended to the `num` variable for subsequent calculations.

```
case 144: Serial.println("[EQ]");
        num += '=';
        num3 = "Equal to:";
        if (sliceString(num, num1, num2))
        {
          Serial.println("Legal data");
          if (num1.toInt() > 100 || num2.toInt() > 100 || num1.toInt() < 0 || num2.toInt() < 0)
          {
            Serial.println("Illegal data");
num = "";
            num3 = "Out of range";
            lcd.setCursor(0, 0);
            lcd.print("                ");
            lcd.setCursor(0, 0);
            lcd.print(num3);
            break;
          }
          if (flag1 )
          {
            value = performOperation(num1.toInt(), num2.toInt(), '+');
            Serial.println("+data");
          } else
          {
            value = performOperation(num1.toInt(), num2.toInt(), '-');
            Serial.println("-data");
          }
          num = "";
          num3 += String(value);
          lcd.setCursor(0, 0);
          lcd.print("                ");
          lcd.setCursor(0, 0);
          lcd.print(num3);
        } else
        {
          num = "";
          num3 += "Error";
          lcd.setCursor(0, 0);
          lcd.print("                ");
          lcd.setCursor(0, 0);
          lcd.print(num3);

        }
        Serial.println("value:");
        Serial.println(value);
        Serial.println();
        Serial.println(num);

        break;
```

Upon pressing the EQ button, defined as ‘=’, it signifies the completion of equation input and initiates the calculation process. Initially, the custom function sliceString is employed to segment the string. Subsequently, the legality of the numerical values is assessed through the statement if (num1.toInt() > 100 || num2.toInt() > 100 || num1.toInt() < 0 || num2.toInt() < 0). Should the input expression be valid, an if (flag1) check is then conducted to determine whether the operation used within the expression is addition ‘+’ or subtraction ‘-’. Finally, the custom function performOperation is utilized to compute the result, which is stored in the variable value. After the calculation, the cursor position is set using lcd.setCursor(0, 0), and the calculated result is displayed by lcd.print(num3). In the event that the input expression is invalid, an “Error” message is displayed on the LCD.

ELECROW

MAKE YOUR MAKING EASIER