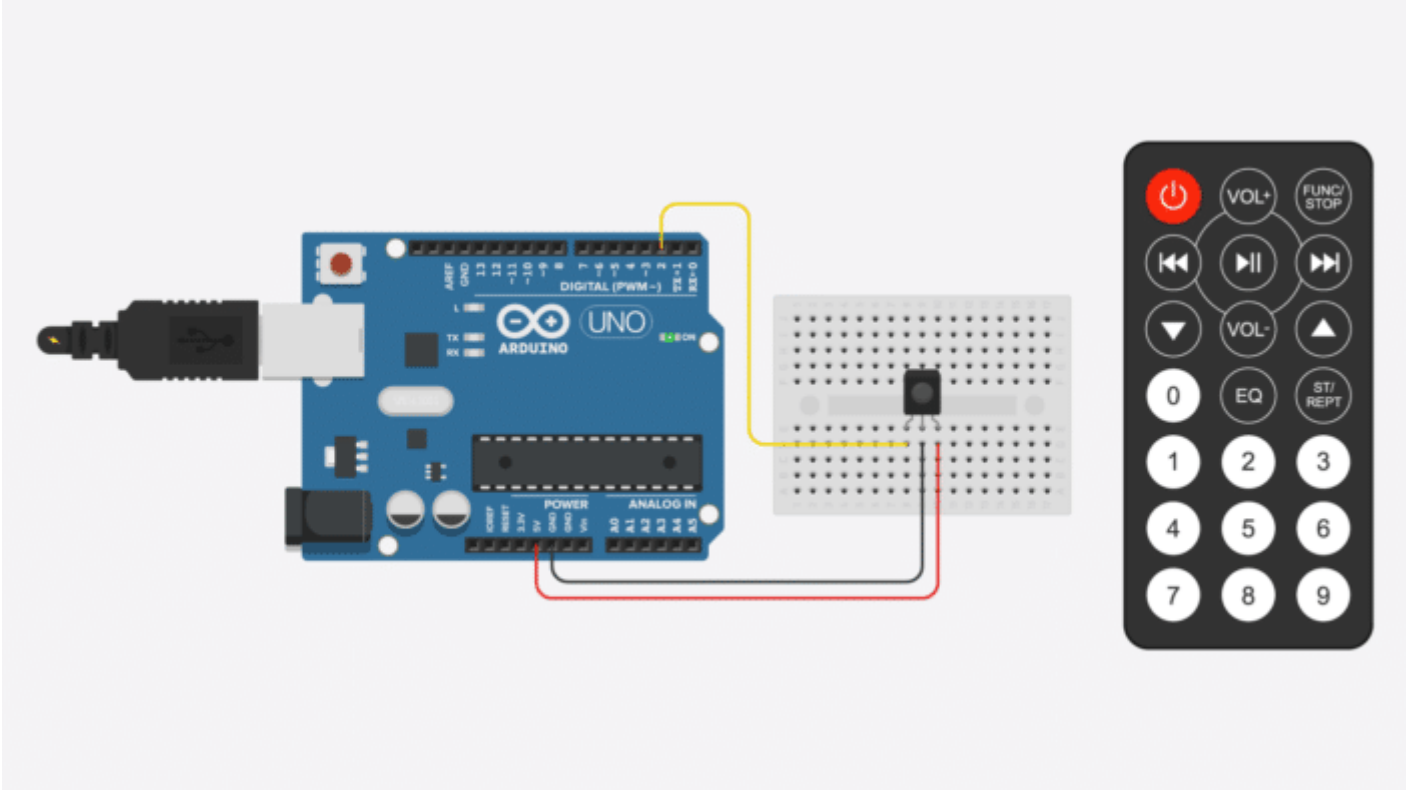
Makerguides.com

How to use an IR receiver and remote with Arduino

Written by Benne de Bakker (https://www.makerguides.com/author/benne-de-bakker/)



In this tutorial, you will learn how to use an IR remote and receiver (https://s.click.aliexpress.com/e/_dUFMHcC) with the Arduino. I have included wiring diagrams and several example codes to help you get started.

By following the instructions in this tutorial you will be able to use virtually any IR remote (like the one from your TV) to control things connected to the Arduino.

In the code examples below, we will be using the **IRremote** Arduino library . This library is fairly easy to use and supports many different IR communication protocols. With the first two examples, you can identify the IR protocol of your remote and determine which

code it sends when you press on a key/button. Next, I will show you how to map the received code to the key values and display these in the Serial Monitor or on an LCD. Lastly, we will look at controlling the outputs of the Arduino with an IR remote and receiver.

If you have any questions, please leave a comment below.

Recommended articles

- How to control a character I2C LCD with Arduino (https://www.makerguides.com/character-i2c-lcd-arduino-tutorial/)
- How to use a 16×2 character LCD with Arduino (https://www.makerguides.com/character-lcd-arduino-tutorial/)
- TB6600 Stepper Motor Driver with Arduino Tutorial (https://www.makerguides.com/tb6600-stepper-motor-driver-arduino-tutorial/)
- How to control servo motors with Arduino (https://www.makerguides.com/servo-arduino-tutorial/)

Supplies

Hardware components

IR remote and receiver (https://s.click.aliexpress.com/e/_dUFMHcC)	× 1	Amazon (AliExpres
Arduino Uno (https://amzn.to/374aJjX)	× 1	Amazon (
Jumper wires (https://amzn.to/2EG9wDc)	× 15	Amazon (AliExpres
16×2 character LCD display (https://www.amazon.com/HiLetgo-Display-Backlight-Controller-Character/dp/B00HJ6AFW6/ref=as_li_ss_tl?keywords=lcd+16x2&qid=1562768812&s=gateway&sr=8-3&linkCode=ll1&tag=makerguides-20&linkId=560296518db01cb992dfe0ac9cc8e4d7&language=en_US)	× 1	Amazon (Controller keywords 3&linkCo 20&linkId AliExpres
Breadboard (https://amzn.to/2sZTxNA)	× 1	Amazon (
10 kΩ potentiometer (https://amzn.to/2Yq66Ph) (breadboard type)	× 1	Amazon (
Resistor assortment (https://amzn.to/2X4liy0)	× 1	Amazon (
LED assortment (https://amzn.to/2RIVQgu)	× 1	Amazon (
USB cable type A/B (https://amzn.to/34SBuXf)	× 1	Amazon (

Software

Arduino IDE (https://www.arduino.cc/en/Main/Software)

How does an infrared (IR) remote and receiver work?

An IR remote and receiver communicate with each other by transmitting and decoding a signal in the form of pulsed IR radiation.

Sent and detected signal by IR transmitter (left) and receiver (right) (Source: SB-Projects (https://www.sbprojects.net/))

Infrared radiation (IR), or infrared light, is a type of electromagnetic radiation with wavelengths ranging from 700 nm to 1 mm. Because humans can only see light with wavelengths of roughly 400 (violet) to 700 (red) nanometers, IR radiation is invisible to the human eye.

Electromagnetic spectrum with visible light highlighted (Source: Wikipedia (https://en.wikipedia.org/wiki/Electromagnetic_radiation))

Since IR transmission is a wireless protocol based on a type of light, it requires a clear line of sight between the transmitter (the remote) and the receiver. This means it can't transmit through walls or ceilings, unlike WiFi or Bluetooth.

IR communication basics

Unfortunately, the IR LED in your remote is not the only source of IR radiation. Any object that has a temperature also radiates in the infrared spectrum. This phenomenon is also used by thermal cameras to detect heat.

IR LED in the end of a remote (Source: SparkFun (https://learn.sparkfun.com/tutorials/ir-communication/all))

All this ambient IR can interfere with the communication between the remote and the receiver. So how does the receiver only detect the IR signal coming from the remote? The answer is signal modulation.

With signal modulation, the IR light source at the end of the remote is blinked with a specific frequency. In consumer electronics, this carrier frequency is usually around 38 kHz.

This specific frequency is used for commercial IR transmission because it is rare in nature and, therefore, it can be distinguished from the ambient IR.

The receiver is built in such a way that it only lets IR through that is coming in at 38 kHz. This is done using a bandpass filter and amplifier. The demodulated binary signal is then sent to the microcontroller (the Arduino) where it is decoded.

So what does an IR signal actually look like?

NEC protocol (Source: SB-Projects (https://www.sbprojects.net/))

In the image above, the vertical axis can be seen as the voltage going to the IR LED in the remote and the horizontal axis is time. So when the LED is on, it is blinked (modulated) at 38 kHz and when it is off, no voltage is applied.

What is important is the amount of time that the IR LED is held high or low (on or off). In the NEC protocol, which is one of the most popular IR transmission protocols, the bits ("1" or "0") are represented as follows:

Each bit consists of a 560 µs long 38 kHz carrier burst (about 21 cycles) followed by a pause. A logical "1" has a total transmission time of 2.25 ms, while a logical "0" only 1.125 ms.

The amount of time the signal stays high or low and the number of bits that are sent for each command is different for all of the IR protocols. In the NEC protocol, the total message usually consists of four 8-bit bytes.

Types of IR receivers

IR receivers, sometimes called IR sensors or IR detection diodes, usually come in two different form factors. You can either buy the diodes separately or mounted on a small breakout board.

IR receiver diode (left) and receiver mounted on a breakout board (right)

They work exactly the same, so it doesn't matter which one you use. The only difference is that the breakout board often contains a small LED that blinks every time the receiver detects a signal which can be handy for debugging.

Connecting an IR receiver to the Arduino

It is very easy to hook up an IR receiver to the Arduino as you only need to connect three wires. The output wire can be connected to any of the digital pins of the Arduino. In this case, I connected it to pin 2 for the first examples below.

The supply power pin is connected to 5 V and the middle ground pin to GND. If you are using a receiver that is mounted on a breakout board, check the labels on the PCB as the order of the pins can be different!

The output pin of the IR receiver is connected to pin 2

The connections are also given in the table below

IR receiver connections

IR receiver	Arduino
OUT (left)	Pin 2
GND (middle)	GND
Vcc (right)	5 V

Installing the IRremote Arduino library

For this tutorial, we will be using the popular **IRremote** library written by Rafi Khan and others. This library is fairly easy to use and supports many different types of IR remotes. You can find the source code of this library here on GitHub (https://github.com/z3t0/Arduino-IRremote).

To install the library, go to Tools > Manage Libraries (Ctrl + Shift + I on Windows) in the Arduino IDE (https://www.arduino.cc/en/main/software). The Library Manager will open and update the list of installed libraries.

You can search for ‘IRremote’ and look for the library by shirriff and z3to. Select the latest version and then click Install.

Determine the IR protocol used by your remote

This section is optional

NEC is probably the best known and most widespread IR transmission protocol as it is used by the vast majority of Japanese-manufactured consumer electronics. However, many other types of protocols exist. It can be useful to known what type of IR protocol your remote is using if you want to work on more advanced projects. Or you might just be curious

The IRremote library currently supports 18 different IR protocols:

- Aiwa
- BoseWave
- Denon
- Dish

- JVC
- Lego
- LG
- MagiQuest
- Mitsubishi
- NEC
- Panasonic
- Philips RC5
- Philips RC6,
- Samsung
- Sanyo
- Sharp
- Sony
- Whynter

Although the library is quite old, new protocols are still being added (see the GitHub (https://github.com/z3t0/Arduino-IRremote/tree/master/src)).

With the code below you can identify which protocol your remote is using. You can also try some other remotes that you have in your house and see if it can detect the protocol.

IR remote protocol finder

```
#include <IRremote.h> // include the IRremote library

#define RECEIVER_PIN 2 // define the IR receiver pin
IRrecv receiver(RECEIVER_PIN); // create a receiver object of the IRrecv class
decode_results results; // create a results object of the decode_results class
unsigned long key_value = 0; // variable to store the pressed key value

void setup() {
  Serial.begin(9600); // begin serial communication with a baud rate of 9600
  receiver.enableIRIn(); // enable the receiver
  receiver.blink13(true); // enable blinking of the built-in LED when an IR signal is received
}

void loop() {
  if (receiver.decode(&results)) {
    if (results.value == 0XFFFFFFFF) {
```

```
        results.value = key_value;
      }
      Serial.println(results.value, HEX);
      switch (results.decode_type) {
        case NEC:
          Serial.println("NEC");
          break;
        case SONY:
          Serial.println("SONY");
          break;
        case RC5:
          Serial.println("RC5");
          break;
        case RC6:
          Serial.println("RC6");
          break;
        case DISH:
          Serial.println("DISH");
          break;
        case SHARP:
          Serial.println("SHARP");
          break;
        case JVC:
          Serial.println("JVC");
          break;
        case SANYO:
          Serial.println("SANYO");
          break;
        case MITSUBISHI:
          Serial.println("MITSUBISHI");
          break;
        case SAMSUNG:
          Serial.println("SAMSUNG");
          break;
        case LG:
          Serial.println("LG");
          break ;
        case WHYNTER:
          Serial.println("WHYNTER");
          break;
        case AIWA_RC_T501:
          Serial.println("AIWA_RC_T501");
          break;
        case PANASONIC:
          Serial.println("PANASONIC");
          break;
        case DENON:
          Serial.println("DENON");
          break;
        case BOSEWAVE:
          Serial.println("BOSEWAVE");
          break;
        case LEGO_PF:
          Serial.println("LEGO_PF");
          break;
        case MAGIQUEST:
```

```
          Serial.println("MAGIQUEST");
          break;
        default:
        case UNKNOWN:
          Serial.println("UNKNOWN");
          break ;
      }
      key_value = results.value;
      receiver.resume();
    }
  }
```

I experimented with some of the remotes in my house and was able to detect the following protocols:

Detected IR protocols from several remotes

Finding the key codes for your remote

Because there are many different types of remotes on the market (different number of keys and values printed on the keys), we need to determine which received signal corresponds to which key.

The IRremote library will read the signal and output a specific code in the form of a hexadecimal number (https://www.arduino.cc/reference/en/language/variables/constants/integerconstants/) depending on which key is pressed.

By printing this output in the Serial Monitor, we can create a conversion table.

You can copy the code below by clicking in the top right corner of the code field.

```
/* Finding the key codes for your remote. More info: https://www.makerguides.com */

#include <IRremote.h> // include the IRremote library

#define RECEIVER_PIN 2 // define the IR receiver pin
IRrecv receiver(RECEIVER_PIN); // create a receiver object of the IRrecv class
decode_results results; // create a results object of the decode_results class

void setup() {
  Serial.begin(9600); // begin serial communication with a baud rate of 9600
  receiver.enableIRIn(); // enable the receiver
  receiver.blink13(true); // enable blinking of the built-in LED when an IR signal is received
}

void loop() {
  if (receiver.decode(&results)) { // decode the received signal and store it in results
    Serial.println(results.value, HEX); // print the values in the Serial Monitor
    receiver.resume(); // reset the receiver for the next code
  }
}
```

After you have uploaded the code, open the Serial Monitor (Ctrl + Shift + M on Windows). Now press each key on the remote and record the corresponding hexadecimal value that you see in the Serial Monitor.

Serial Monitor output

Note that you will see the code **FFFFFFFF** when you press on a key continuously. This is the repeat code send by the remote.

For my remote I got the following conversion table:

(https://s.click.aliexpress.com/e/_dTmSQpQ)

Key	Code
POWER	0xFD00FF
VOL+	0xFD807F
FUNC/STOP	0xFD40BF
|◄◄	0xFD20DF

Key	Code
▶||	0xFDA05F
►►|	0xFD609F
▼	0xFD10EF
VOL-	0xFD906F
▲	0xFD50AF
0	0xFD30CF
EQ	0xFDB04F
ST/REPT	0xFD708F
1	0xFD08F7
2	0xFD8877
3	0xFD48B7
4	0xFD28D7
5	0xFDA857
6	0xFD6897
7	0xFD18E7

Key	Code
8	0xFD9867
9	0xFD58A7

As you can see in the table, hexadecimal values are indicated by the prefix “0x”.

Note that your table will probably look different! You will have to create your own to use in the rest of the code examples below.

IR remote and receiver Arduino example code – Print key values in the Serial Monitor

Now that we know which code (hexadecimal value) corresponds to which keypress, we can modify the code to print the value of the pressed key in the Serial Monitor.

For this, we will be using a switch case (https://www.arduino.cc/reference/en/language/structure/control-structure/switchcase/) control structure. This lets us execute a different piece of code depending on which key is pressed.

The code example below prints the key value in the Serial Monitor instead of the hexadecimal value like we did in the previous example. After uploading the code you can read the explanation below to learn how the code works.

```
 /* IR remote and receiver Arduino example code. Print key values in the Serial Monitor. More info: https://www.makerguides.com */

#include <IRremote.h> // include the IRremote library

#define RECEIVER_PIN 2 // define the IR receiver pin
IRrecv receiver(RECEIVER_PIN); // create a receiver object of the IRrecv class
decode_results results; // create a results object of the decode_results class
unsigned long key_value = 0; // variable to store the key value

void setup() {
  Serial.begin(9600); // begin serial communication with a baud rate of 9600
  receiver.enableIRIn(); // enable the receiver
  receiver.blink13(true); // enable blinking of the built-in LED when an IR signal is received
}

void loop() {
  if (receiver.decode(&results)) { // decode the received signal and store it in results
    if (results.value == 0xFFFFFFFF) { // if the value is equal to 0xFFFFFFFF
      results.value = key_value; // set the value to the key value
    }
    switch (results.value) { // compare the value to the following cases
      case 0xFD00FF: // if the value is equal to 0xFD00FF
        Serial.println("POWER"); // print "POWER" in the Serial Monitor
        break;
      case 0xFD807F:
        Serial.println("VOL+");
        break;
      case 0xFD40BF:
        Serial.println("FUNC/STOP");
        break;
      case 0xFD20DF:
        Serial.println("|<<");
        break;
      case 0xFDA05F:
        Serial.println(">||");
        break ;
      case 0xFD609F:
        Serial.println(">>|");
        break ;
```

```
        case 0xFD10EF:
          Serial.println("DOWN");
          break ;
        case 0xFD906F:
          Serial.println("VOL-");
          break ;
        case 0xFD50AF:
          Serial.println("UP");
          break ;
        case 0xFD30CF:
          Serial.println("0");
          break ;
        case 0xFDB04F:
          Serial.println("EQ");
          break ;
        case 0xFD708F:
          Serial.println("ST/REPT");
          break ;
        case 0xFD08F7:
          Serial.println("1");
          break ;
        case 0xFD8877:
          Serial.println("2");
          break ;
        case 0xFD48B7:
          Serial.println("3");
          break ;
        case 0xFD28D7:
          Serial.println("4");
          break ;
        case 0xFDA857:
          Serial.println("5");
          break ;
        case 0xFD6897:
          Serial.println("6");
          break ;
        case 0xFD18E7:
          Serial.println("7");
          break ;
        case 0xFD9867:
          Serial.println("8");
          break ;
        case 0xFD58A7:
          Serial.println("9");
          break ;
      }
      key_value = results.value; // store the value as key_value
      receiver.resume(); // reset the receiver for the next code
    }
  }
```

Serial Monitor output

If your remote sends different key codes than the ones shown in the table above, simply replace the hex and key value in each of the lines in the switch case statement that look like this:

```
22.   case 0xFD00FF: // if the value is equal to 0xFD00FF
23.   Serial.println("POWER"); // print "POWER" in the Serial Monitor
```

These lines translate to: when results.value is equal to 0xFD00FF, print “POWER” in the Serial Monitor. Again notice that you need to add “0x” before the values that you saw in the Serial Monitor in the previous example.

To prevent unwanted double clicks, you could add a short delay (e.g. 500 ms) at the end of the loop.

How the code works

The first step is to include the IRremote library. If you get the error message “IRremote.h: No such file or directory”, you have probably forgotten to install the library.

```
3.  #include <IRremote.h> // include the IRremote library
```

Next, I defined to which Arduino pin the receiver output is connected. The statement `#define` is used to give a name to a constant value. The compiler will replace any references to this constant with the defined value when the program is compiled. So everywhere you mention `RECEIVER_PIN`, the compiler will replace it with the value 2 when the program is compiled.

```
5.  #define RECEIVER_PIN 2 // define the IR receiver pin
```

After that, we need to create an object of the `IRrecv` class and link it to the receiver output pin (pin 2). In this case, I called the object 'receiver' but you can use other names as well. This object takes care of the IR protocol and processing of the receiver signal.

```
6.  IRrecv receiver(RECEIVER_PIN); // create a receiver object of the IRrecv class
```

Next, an object called `results` of the `decode_results` class is created. This object stores the decoded values from the receiver object. We will access these values in the rest of the code.

```
7.  decode_results results; // create a results object of the decode_results class
```

In the last line before the setup section we create a variable to store the key value.

```
8.  unsigned long key_value = 0; // variable to store the key value
```

In the setup section of the code, we first set the data rate for serial communication in bits per second (baud rate) to 9600. Make sure that the Serial Monitor is also set to the same baud rate.

Next, we initialize the receiver with the function `enableIRIn()`. The next line enables the blinking of the built-in LED of the Arduino. The `blink13()` function will blink the LED connected to digital pin 13 every time the receiver gets a signal from the remote control. This can be very handy for debugging.

```
10.  void setup() {
11.    Serial.begin(9600); // begin serial communication with a baud rate of 9600
12.    receiver.enableIRIn(); // enable the receiver
13.    receiver.blink13(true); // enable blinking of the built-in LED when an IR signal is
     received
14.  }
```

The loop section of the code starts with an if statement (https://www.arduino.cc/reference/en/language/structure/control-structure/if/) checks for a condition and executes the proceeding statement or set of statements if the condition is 'true'. When a code is received, the condition `receiver.decode(&results)` returns true and the rest of the code is executed.

As I mentioned before, when you continuously press a key we start receiving 0xFFFFFFFF from the remote. This means a repetition of the previous key. Because I want to keep printing the pressed key value I added the following lines to the code:

```
18.  if (results.value == 0xFFFFFFFF) { // if the value is equal to 0xFFFFFFFF
19.    results.value = key_value; // set the value to the key value
20.  }
```

and

```
86.  key_value = results.value; // store the value as key_value
```

at the end of the loop. When the repeat code 0xFFFFFFFF is received, we replace it with the previous key value which we stored as `key_value` in line 86.

After that, the switch case statement is used to check the received code against the different key codes that we recorded in the previous example. The corresponding key values are then printed in the Serial Monitor on a new line with the function `Serial.println()`.

At the end of the loop section, the function `resume()` is called, which resets the receiver and prepares it to receive the next code.

```
87.       receiver.resume(); // reset the receiver for the next code
```

IR remote and receiver with Arduino and LCD example

The following example can be used to print the pressed key values on a character LCD (https://s.click.aliexpress.com/e/_d77i1Hw). I have written a detailed tutorial on using character LCD displays which you can find here:

- How to use a 16×2 character LCD with Arduino (https://www.makerguides.com/character-lcd-arduino-tutorial/)

If you prefer to use an I2C LCD which needs fewer connections, see the tutorial below:

- How to control a character I2C LCD with Arduino (https://www.makerguides.com/character-i2c-lcd-arduino-tutorial/)

For this example, I connected the output of the IR receiver to digital pin 8 instead of 2. The connections for the character LCD display are shown in the wiring diagram below. Note that you also need a 10 kΩ potentiometer (https://amzn.to/2Yq66Ph) to set the contrast of the display and a 220 Ω resistor (https://amzn.to/2X4liy0) to control the brightness of the backlight.

IR remote and receiver with Arduino Uno and 16×2 character LCD wiring diagram

The connections are also given in the table below. The leftmost pin of the LCD is pin 1 (GND).

LCD and IR receiver connections

Pin	Connection
LCD pin 1 (GND)	Arduino GND
LCD pin 2 (VCC)	Arduino 5 V
LCD pin 3 (VO)	Middle pin potentiometer
Left pin potentiometer	Arduino 5 V
Right pin potentiometer	Arduino GND
LCD pin 4 (RS)	Arduino pin 2
LCD pin 5 (RW)	Arduino GND
LCD pin 6 (E)	Arduino pin 3
LCD pin 11 (DB4)	Arduino pin 4
LCD pin 12 (DB5)	Arduino pin 5
LCD pin 13 (DB6)	Arduino pin 6
LCD pin 14 (DB7)	Arduino pin 7
LCD pin 15 (LED-anode)	Arduino 5 V via 220Ω resistor
LCD pin 16 (LED-kathode)	Arduino GND
IR receiver OUT (left)	Arduino pin 8

Pin	Connection
IR receiver GND (middle)	Arduino GND
IR receiver Vcc (right)	Arduino 5 V

If you encounter any problems with the LCD display, see this tutorial (https://www.makerguides.com/character-lcd-arduino-tutorial/) for more details.

With the example code below, you can print the pressed key value on the LCD. I have programmed the power button so that it clears the display.

Again, if your remote sends different codes, just replace the hexadecimal values and corresponding key values in the switch case statement.

IR remote and receiver with Arduino and LCD display example code

You can copy the code by clicking on the button in the top right corner of the code field.

```
 /* IR remote and receiver with Arduino and character LCD example code. More info: https://www.makerguides.com/ */

#include <IRremote.h> // include the IRremote library
#include <LiquidCrystal.h> // include the LiquidCrystal library


LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7); //create an LCD object with pin parameters (RS, E, D4, D5, D6, D7)

#define RECEIVER_PIN 8 // define the IR receiver pin
IRrecv receiver(RECEIVER_PIN); // create a receiver object of the IRrecv class
decode_results results; // create a results object of the decode_results class
unsigned long key_value = 0; // variable to store the key value

void setup() {
  lcd.begin(16, 2); // enable LCD with 16 colums and 2 rows
  receiver.enableIRIn(); // enable the receiver
  receiver.blink13(true); // enable blinking of the built-in LED when an IR signal is received
 }
```

```
19.
20.  void loop() {
21.    if (receiver.decode(&results)) { // decode the received signal and store it in results
22.      if (results.value == 0xFFFFFFFF) { // if the value is equal to 0xFFFFFFFF
23.        results.value = key_value; // set the value to the key value
24.      }
25.      switch (results.value) { // compare the value to the following cases
26.        case 0xFD00FF: // if the value is equal to 0xFD00FF
27.          lcd.clear(); // clear the display
28.          break;
29.        case 0xFD807F:
30.          lcd.print("VOL+");
31.          break;
32.        case 0xFD40BF:
33.          lcd.print("FUNC/STOP");
34.          break;
35.        case 0xFD20DF:
36.          lcd.print("|<<");
37.          break;
38.        case 0xFDA05F:
39.          lcd.print(">||");
40.          break ;
41.        case 0xFD609F:
42.          lcd.print(">>|");
43.          break ;
44.        case 0xFD10EF:
45.          lcd.print("DOWN");
46.          break ;
47.        case 0xFD906F:
48.          lcd.print("VOL-");
49.          break ;
50.        case 0xFD50AF:
51.          lcd.print("UP");
52.          break ;
53.        case 0xFD30CF:
54.          lcd.print("0");
55.          break ;
56.        case 0xFDB04F:
57.          lcd.print("EQ");
58.          break ;
59.        case 0xFD708F:
60.          lcd.print("ST/REPT");
61.          break ;
62.        case 0xFD08F7:
63.          lcd.print("1");
64.          break ;
65.        case 0xFD8877:
66.          lcd.print("2");
67.          break ;
68.        case 0xFD48B7:
69.          lcd.print("3");
70.          break ;
71.        case 0xFD28D7:
72.          lcd.print("4");
73.          break ;
74.        case 0xFDA857:
```

```
          lcd.print("5");
          break ;
        case 0xFD6897:
          lcd.print("6");
          break ;
        case 0xFD18E7:
          lcd.print("7");
          break ;
        case 0xFD9867:
          lcd.print("8");
          break ;
        case 0xFD58A7:
          lcd.print("9");
          break ;
      }
      key_value = results.value; // store the value as key_value
      receiver.resume(); // reset the receiver for the next code
    }
  }
```

You should see the following output on the LCD if you press the VOL+ key. Note that you might have to adjust the potentiometer to increase the contrast of the display.

(https://s.click.aliexpress.com/e/_d9D5r6G)

LCD display output

Control LEDs (Arduino outputs) with IR remote and receiver

Although it's fun to see the key values in the Serial Monitor or on an LCD display, you will probably want to use the remote for something more useful, i.e. actual control something. In the following example, I will show you how to toggle LEDs on and off with the remote. You can also use this type of code to control relays, lights, and motors.

The code looks mostly the same as the previous examples but I added an extra functions to control the LEDs.

You will have to create the following circuit to control the LEDs:

Wiring diagram to control LEDs with IR remote, receiver and Arduino

The LEDs are connected to pins 2 to 5 of the Arduino and the output of the IR receiver to pin 6. I used some 470 Ω resistors to limit the current going through the LEDs. The usual operating current of 3 and 5 mm LEDs is around 20 mA. You can determine what value resistor you need with an online calculator like this one (https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-led-series-resistor). I like to buy these resistor assortments on Amazon (https://amzn.to/3l9sG8D) so I always have the right value on hand.

```
 /* Example code to control LEDs with an IR remote and receiver with Arduino. More info: https://www.makerguides.com/ */

#include <IRremote.h>

#define RECEIVER_PIN 6 // define the connections
#define RED_LED_PIN 2
#define YELLOW_LED_PIN 3
#define GREEN_LED_PIN 4
#define BLUE_LED_PIN 5

IRrecv receiver(RECEIVER_PIN); // create a receiver object of the IRrecv class
```

```
12.  decode_results results; // create a results object of the decode_results class
13.
14.  unsigned long key_value = 0;
15.
16.  void setup() {
17.    Serial.begin(9600); // begin serial communication at a baud rate of 9600
18.    receiver.enableIRIn(); // enable the receiver
19.    receiver.blink13(true); // enable blinking of the built-in LED when an IR signal is
     received
20.    pinMode(RED_LED_PIN, OUTPUT); // set the LED pins as output
21.    pinMode(YELLOW_LED_PIN, OUTPUT);
22.    pinMode(GREEN_LED_PIN, OUTPUT);
23.    pinMode(BLUE_LED_PIN, OUTPUT);
24.    digitalWrite(RED_LED_PIN, LOW); // turn all the LEDs off
25.    digitalWrite(YELLOW_LED_PIN, LOW);
26.    digitalWrite(GREEN_LED_PIN, LOW);
27.    digitalWrite(BLUE_LED_PIN, LOW);
28.  }
29.
30.  void loop() {
31.    if (receiver.decode(&results)) {
32.      if (results.value == 0xFFFFFFFF) {
33.        results.value = key_value;
34.      }
35.
36.      switch (results.value) {
37.        case 0xFD00FF:
38.          Serial.println("POWER");
39.          break;
40.        case 0xFD807F:
41.          Serial.println("VOL+");
42.          break;
43.        case 0xFD40BF:
44.          Serial.println("FUNC/STOP");
45.          break;
46.        case 0xFD20DF:
47.          Serial.println("|<<");
48.          break;
49.        case 0xFDA05F:
50.          Serial.println(">||");
51.          break ;
52.        case 0xFD609F:
53.          Serial.println(">>|");
54.          break ;
55.        case 0xFD10EF:
56.          Serial.println("DOWN");
57.          break ;
58.        case 0xFD906F:
59.          Serial.println("VOL-");
60.          break ;
61.        case 0xFD50AF:
62.          Serial.println("UP");
63.          break ;
64.        case 0xFD30CF:
65.          Serial.println("0");
66.          break ;
```

```
        case 0xFDB04F:
          Serial.println("EQ");
          break ;
        case 0xFD708F:
          Serial.println("ST/REPT");
          break ;
        case 0xFD08F7:
          Serial.println("1");
          toggleLED(RED_LED_PIN); // run the toggle LED function with the red LED pin as
input
          break ;
        case 0xFD8877:
          Serial.println("2");
          toggleLED(YELLOW_LED_PIN);
          break ;
        case 0xFD48B7:
          Serial.println("3");
          toggleLED(GREEN_LED_PIN);;
          break ;
        case 0xFD28D7:
          Serial.println("4");
          toggleLED(BLUE_LED_PIN);
          break ;
        case 0xFDA857:
          Serial.println("5");
          break ;
        case 0xFD6897:
          Serial.println("6");
          break ;
        case 0xFD18E7:
          Serial.println("7");
          break ;
        case 0xFD9867:
          Serial.println("8");
          break ;
        case 0xFD58A7:
          Serial.println("9");
          break ;
      }

      key_value = results.value;
      receiver.resume();
    }
  }

  void toggleLED(int pin) { // function to toggle the LED on and off
    if (digitalRead(pin) == HIGH) { // if the LED is on
      digitalWrite(pin, LOW); // turn it off
    }
    else { // else
      digitalWrite(pin, HIGH); // turn it on
    }
  }
```

If you press the buttons 1 – 4 on the remote, you can turn the corresponding LEDs on and off.

Code explanation

Below I will quickly highlight the differences in the code compared to the previous examples.

In the first part of the code, I defined the Arduino pins to which the LEDs and IR receiver are connected. In the setup section, I set the LED pins as output and turned all the LEDs off with `digitalWrite(pin, value)` .

```
18.     pinMode(RED_LED_PIN, OUTPUT); // set the LED pins as output
19.     pinMode(YELLOW_LED_PIN, OUTPUT);
20.     pinMode(GREEN_LED_PIN, OUTPUT);
21.     pinMode(BLUE_LED_PIN, OUTPUT);
22.     digitalWrite(RED_LED_PIN, LOW); // turn all the LEDs off
23.     digitalWrite(YELLOW_LED_PIN, LOW);
24.     digitalWrite(GREEN_LED_PIN, LOW);
25.     digitalWrite(BLUE_LED_PIN, LOW);
```

The loop section of the code is mostly the same as before, but now instead of just printing the key values in the Serial Monitor, the `toggleLED(pin)` function is called if you press on keys 0 to 4.

This function takes the specific LED pin as input and toggles the LED on or off if you press on the key. Note that if you continuously press on a key, the LED will just keep turning on and off.

```
109.  void toggleLED(int pin) { // function to toggle the LED on and off
110.    if (digitalRead(pin) == HIGH) { // if the LED is on
111.      digitalWrite(pin, LOW); // turn it off
112.    }
113.    else { // else
114.      digitalWrite(pin, HIGH); // turn it on
115.    }
```

Since you are basically just controlling the 4 output pins of the Arduino, you can also use this function to turn relays on and off.

Now if you simply write a different function and call it when a specific button is pressed, you can basically control anything you want. I would love to know what things you are controlling with your remote and it would be great if you could share with the other readers how you implemented it.

Conclusion

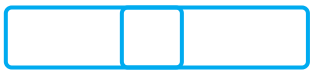
In this tutorial, I have shown you how to use an IR remote and receiver with Arduino. We looked at several different code examples for determining the IR protocol and identifying the IR key codes for your remote. We then looked at displaying the key/button values in the Serial Monitor and on a 16×2 character LCD. Lastly, I showed you how to control the outputs of the Arduino with the remote to toggle some LEDs on and off. There are many more applications for IR receivers and remotes, so be sure to leave some suggestions in the comments.

I hope you found this article useful and informative. If you did, **please share it with a friend** that also likes electronics and making things!

I would love to know what projects you plan on building (or have already built) with an IR remote and receiver. If you have any questions, suggestions, or if you think that things are missing in this tutorial, **please leave a comment below**.

Note that comments are held for moderation to prevent spam.

Beginner


What to read next?

LM35 analog temperature sensor with Arduino tutorial (https://www.makerguides.com/lm35-arduino-tutorial/)

TMP36 analog temperature sensor with Arduino tutorial (https://www.makerguides.com/tmp36-arduino-tutorial/)

Arduino Nano Board Guide (Pinout, Specifications, Comparison) (https://www.makerguides.com/arduino-nano/)

The complete guide for DS18B20 digital temperature sensors with Arduino (https://www.makerguides.com/ds18b20-arduino-tutorial/)

How to use an IR receiver and remote with Arduino (https://www.makerguides.com/ir-receiver-remote-arduino-tutorial/)

Comments

kurt ifanger (http://www.kifanger.ch/) says
March 9, 2021 at 9:09 pm (https://www.makerguides.com/ir-receiver-remote-arduino-tutorial/#comment-7482)

bei mir funktioniert der code nicht

The function decode(&results)) is deprecated and may not work as expected! Just use decode() – without any parameter.

haben Sie eine lösung
danke
kurt

Reply

Socks says
January 31, 2021 at 10:57 am (https://www.makerguides.com/ir-receiver-remote-arduino-tutorial/#comment-6336)

I think this code is out of date as the Library has been updated it would be helpful if you could update the code pls they wrote about how to update to the new 3.0 library on the GIT hub.

Reply

Benne de Bakker says
March 2, 2021 at 9:01 am (https://www.makerguides.com/ir-receiver-remote-arduino-tutorial/#comment-7297)

Hi Socks,

Thanks for letting me know, I will try to update the tutorial when I have time.
Benne

Reply

(https://www.makerguides.com/dht11-dht22-arduino-tutorial/)

How to use DHT11 and DHT22 Sensors with Arduino (https://www.makerguides.com/dht11-dht22-arduino-tutorial/)

(https://www.makerguides.com/arduino-motor-shield-stepper-motor-tutorial/)

How to control a Stepper Motor with Arduino Motor Shield Rev3 (https://www.makerguides.com/arduino-motor-shield-stepper-motor-tutorial/)

(https://www.makerguides.com/servo-arduino-tutorial/)

How to control servo motors with Arduino (https://www.makerguides.com/servo-arduino-tutorial/)