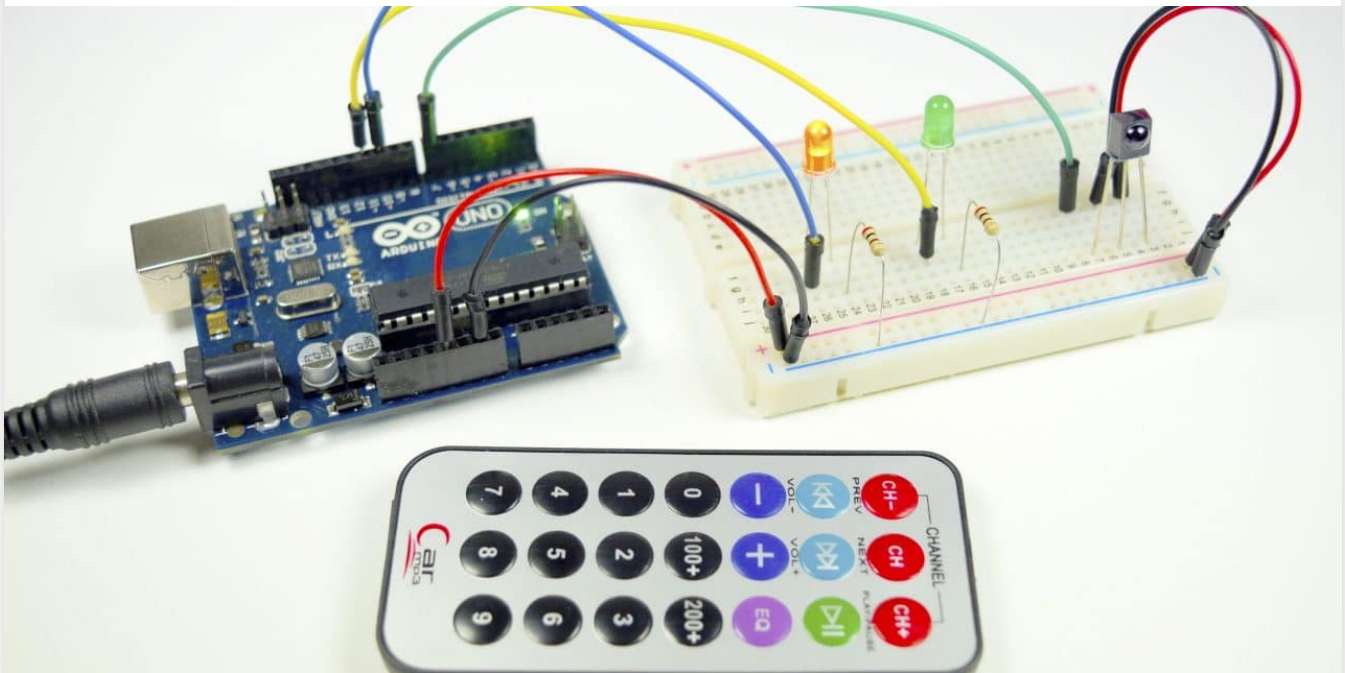


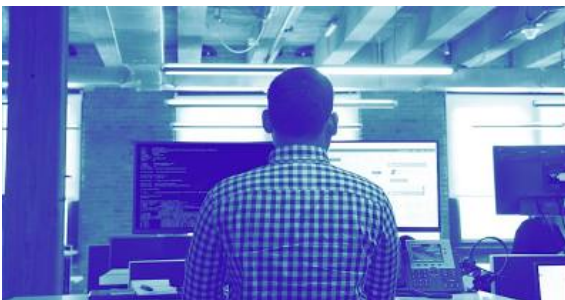
# HOW TO SET UP AN IR REMOTE AND RECEIVER ON AN ARDUINO

Posted by Krishna Pattabiraman | Arduino | 6 🗨



Infrared (IR) communication is a widely used and easy to implement wireless technology that has many useful applications. The most prominent examples in day to day life are TV/video remote controls, motion sensors, and infrared thermometers.

There are plenty of interesting Arduino projects that use IR communication too. With a simple IR transmitter and receiver, you can make remote controlled robots, distance sensors, heart rate monitors, DSLR camera remote controls, TV remote controls, and lots more.



Get unattended access.  
Help your customers solve  
computer issues remotely.



Ad GoToAssist

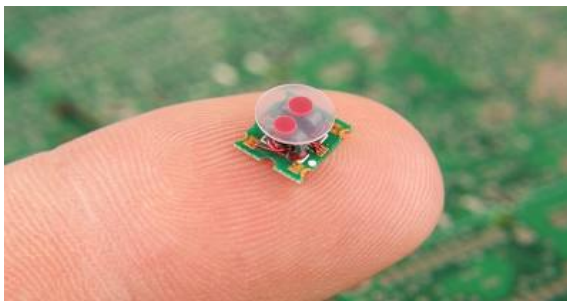
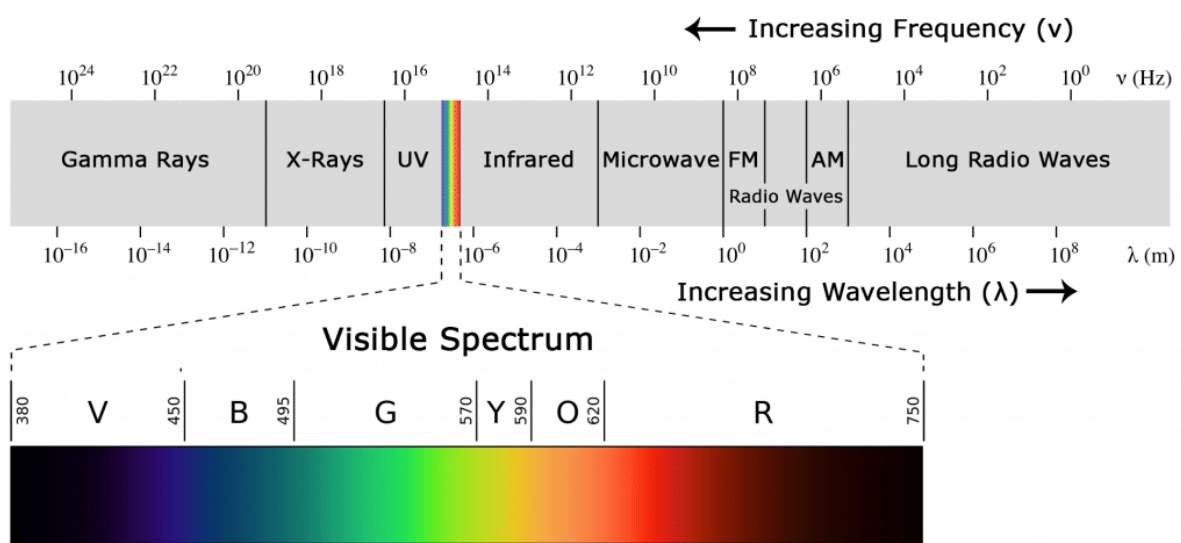
OPEN

In this tutorial I'll first explain what infrared is and how it works. Then I'll show you how to set up an [IR receiver and remote](#) on an [Arduino](#). I'll also show you how to use virtually any IR remote (like the one for your TV) to control things connected to the Arduino.

Now let's get into the details...

## WHAT IS INFRARED?

Infrared radiation is a form of light similar to the light we see all around us. The only difference between IR light and visible light is the frequency and wavelength. Infrared radiation lies outside the range of visible light, so humans can't see it:



### 11,000% Bigger Than BTC?



**Ad** Meet the company taking on an \$11.2 trillion dollar market. Don't miss this one.

Decentric

OPEN

Because IR is a type of light, IR communication requires a direct line of sight from the receiver to the transmitter. It can't transmit through walls or other materials like WiFi or Bluetooth.

## HOW IR REMOTES AND RECEIVERS WORK

A typical infrared communication system requires an IR transmitter and an IR receiver. The transmitter looks just like a standard LED, except it produces light in the

IR spectrum instead of the visible spectrum. If you have a look at the front of a TV remote, you'll see the IR transmitter LED:



Get unattended access.  
Help your customers solve  
computer issues remotely.



Ad GoToAssist

OPEN

The same type of LED is used in IR transmitter breakout boards for the Arduino. You can see it at the front of this Keyes IR transmitter:





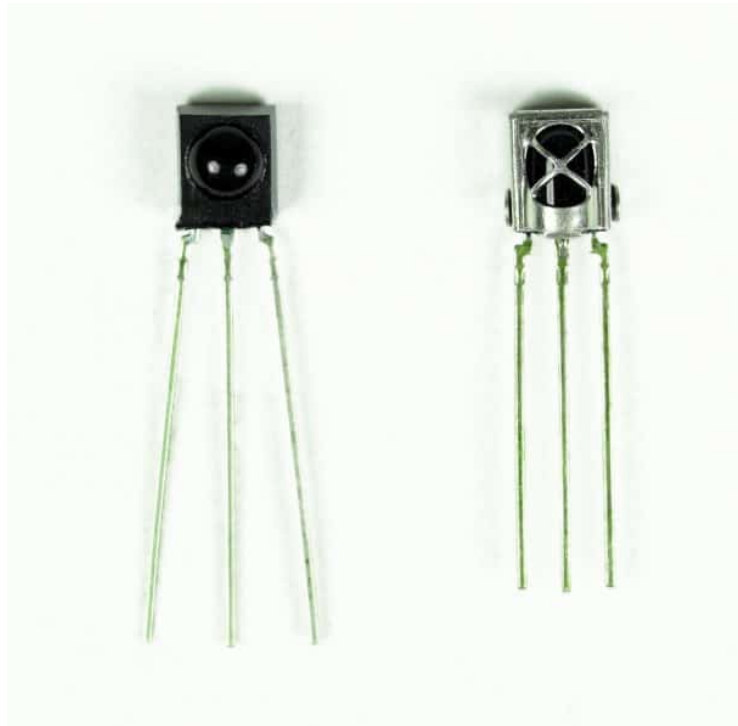
**Bid Now for Fast Delivery on all Remaining Clearance Buildings**

**Ad** We're Overstocked! All 30-50' Wide Buildings to be Sold. Make an Offer

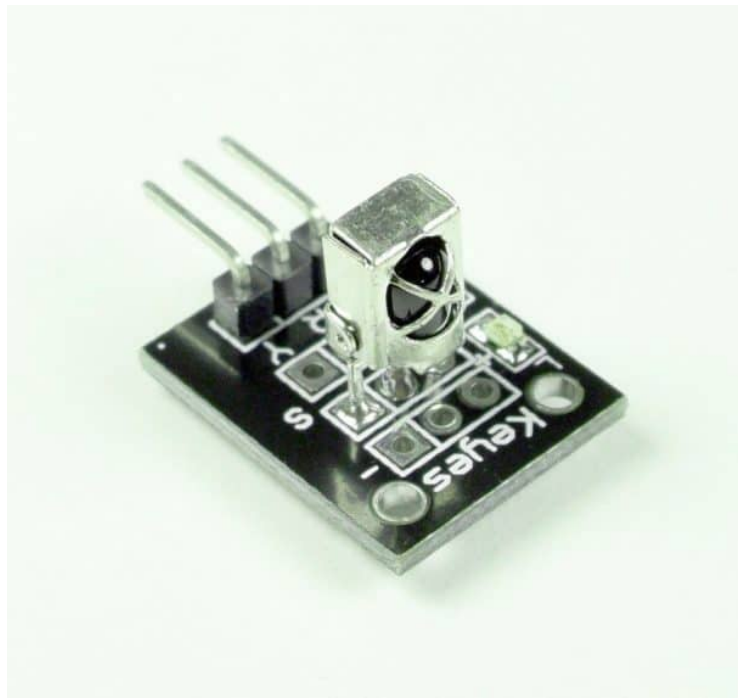
Future Buildings

[OPEN](#)

The IR receiver is a [photodiode](#) and pre-amplifier that converts the IR light into an electrical signal. [IR receiver diodes](#) typically look like this:



Some may come on a breakout board like this:

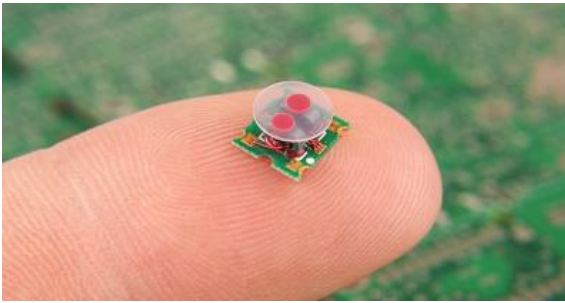


## IR SIGNAL MODULATION

IR light is emitted by the sun, light bulbs, and anything else that produces heat. That means there is a lot of IR light noise all around us. To prevent this noise from interfering with the IR signal, a signal modulation technique is used.

In IR signal modulation, an encoder on the IR remote converts a binary signal into a modulated electrical signal. This electrical signal is sent to the transmitting LED. The transmitting LED converts the modulated electrical signal into a modulated IR light signal. The IR receiver then demodulates the IR light signal and converts it back to binary before passing on the information to a microcontroller:





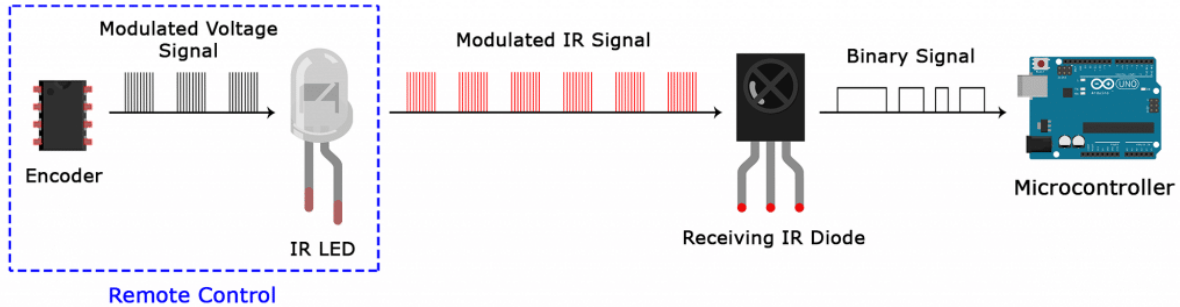
## 11,000% Bigger Than BTC?



**Ad** Meet the company taking on an \$11.2 trillion dollar market. Don't miss this one.

Decentric

OPEN



## Learn Best Practices for Modernizing & Consolidating Datacenter & Platforms



**Ad** HPE

Visit Site

The modulated IR signal is a series of IR light pulses switched on and off at a high frequency known as the carrier frequency. The carrier frequency used by most transmitters is 38 kHz, because it is rare in nature and thus can be distinguished from ambient noise. This way the IR receiver will know that the 38 kHz signal was sent from the transmitter and not picked up from the surrounding environment.

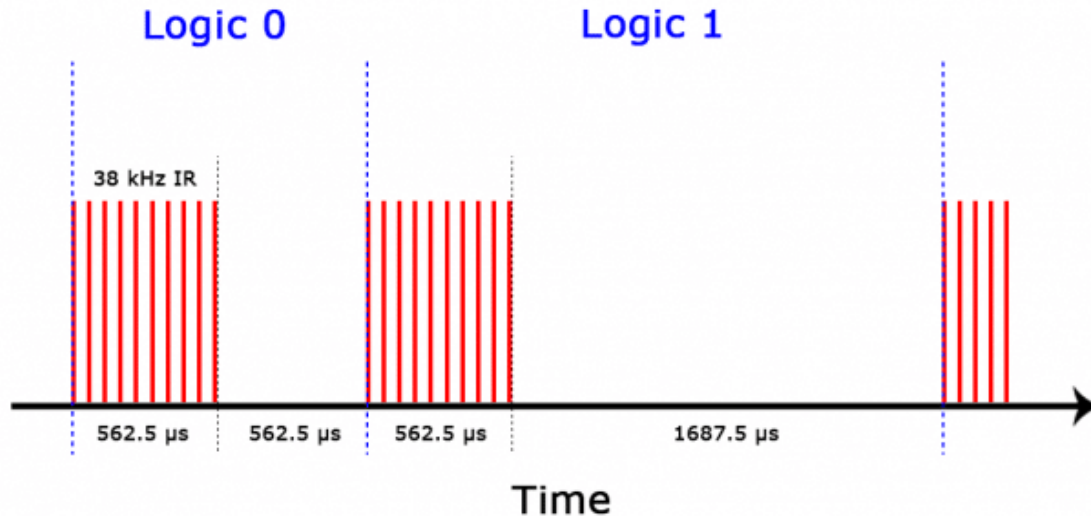
The receiver diode detects all frequencies of IR light, but it has a band-pass filter and only lets through IR at 38 kHz. It then amplifies the modulated signal with a pre-amplifier and converts it to a binary signal before sending it to a microcontroller.

## IR TRANSMISSION PROTOCOLS

The pattern in which the modulated IR signal is converted to binary is defined by a transmission protocol. There are many IR transmission protocols. Sony, Matsushita, NEC, and RC5 are some of the more common protocols.

The NEC protocol is also the most common type in Arduino projects, so I'll use it as an example to show you how the receiver converts the modulated IR signal to a binary one.

Logical '1' starts with a 562.5  $\mu$ s long HIGH pulse of 38 kHz IR followed by a 1,687.5  $\mu$ s long LOW pulse. Logical '0' is transmitted with a 562.5  $\mu$ s long HIGH pulse followed by a 562.5  $\mu$ s long LOW pulse:



## Your Apps Have Gone Serverless. Has Your Security?



**Ad** Discover the Threats and Solutions. Get Protego's Serverless Security eBook ...

Protego

Download

This is how the NEC protocol encodes and decodes the binary data into a modulated signal. Other protocols differ only in the duration of the individual HIGH and LOW pulses.

## IR CODES

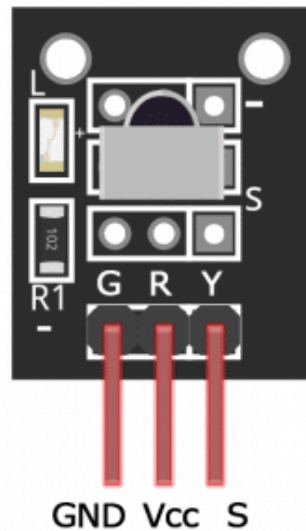
Each time you press a button on the remote control, a unique hexadecimal code is generated. This is the information that is modulated and sent over IR to the receiver. In order to decipher which key is pressed, the receiving microcontroller needs to know which code corresponds to each key on the remote.

Different remotes send different codes for the keypresses, so you'll need to determine the code generated for each key on your particular remote. If you can find the datasheet, the IR key codes should be listed. If not though, there is a simple Arduino sketch that will read most of the popular remote controls and print the hexadecimal codes to the serial monitor when you press a key. I'll show you how to set that up in a minute, but first we need to connect the receiver to the Arduino...

# HOW TO CONNECT AN IR RECEIVER TO THE ARDUINO

There are several different types of IR receivers, some are stand-alone, and some are mounted on a breakout board. Check the datasheet for your particular IR receiver since the pins might be arranged differently than the [HX1838 IR receiver and remote set](#) I am using here. However, all IR receivers will have three pins: signal, ground, and Vcc.

Lets get started with the hardware connections. The pin layout on most breakout boards looks like this:



## PROPRICER: Proposal Pricing Suite for Federal Contractors [▶](#) [×](#)

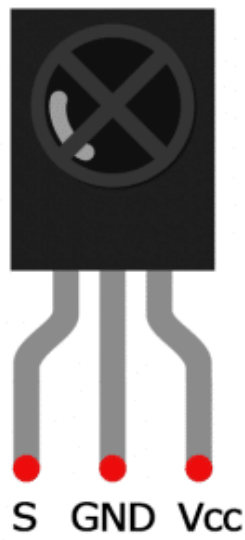
**Ad** The fastest way for teams to estimate, price, & bid on FAR/CAS-covered federal ...

PROPRICER

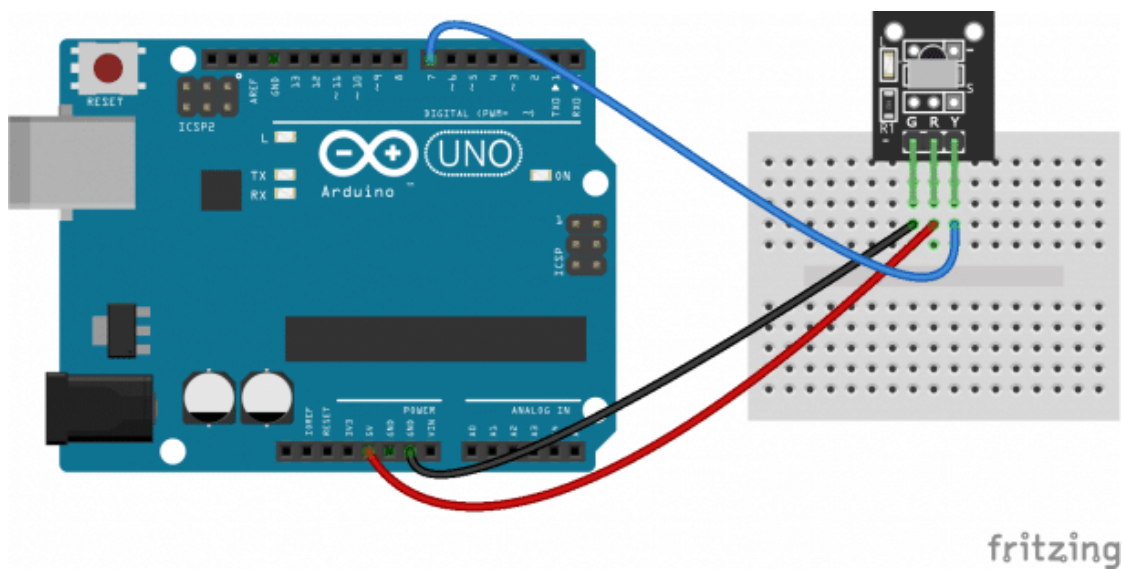
[Learn More](#)

The pinout of most stand-alone diodes is like this:

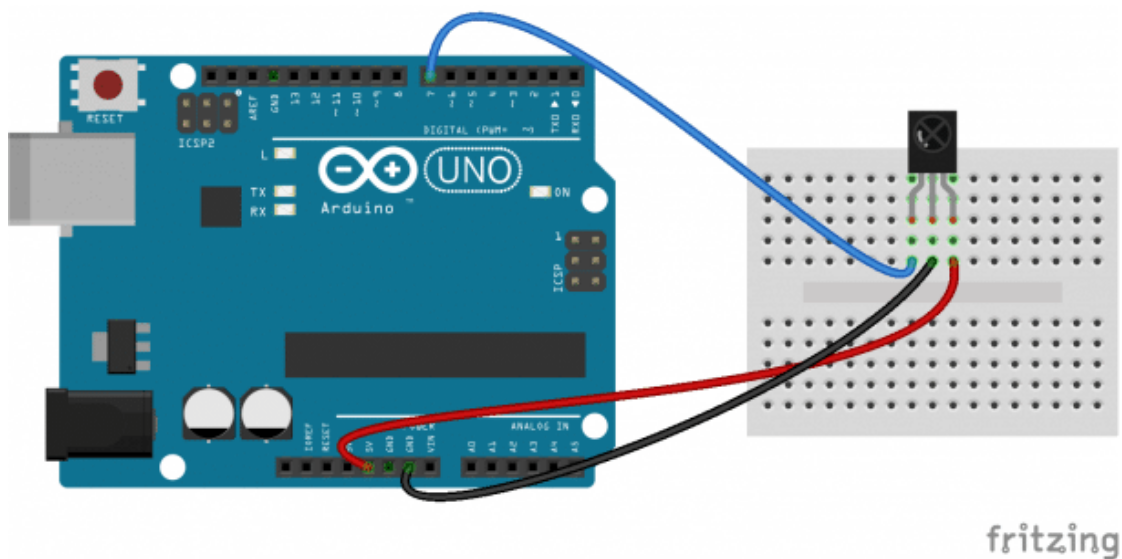




To connect a breakout board mounted IR receiver, hook it up to the Arduino like this:



To connect a stand-alone receiver diode, wire it like this:



## PROGRAMMING THE IR RECEIVER

Once you have the receiver connected, we can install the Arduino library and start programming. In the examples below, I'll show you how to find the codes sent by your remote, how to find the IR protocol used by your remote, how to print key presses to the serial monitor or an LCD, and finally, how to control the Arduino's output pins with a remote.

### INSTALL THE IRREMOTE LIBRARY

We'll be using the IRremote library for all of the code examples below. You can download a ZIP file of the library from [here](#).

To install the library from the ZIP file, open up the Arduino IDE, then go to Sketch > Include Library > Add .ZIP Library, then select the IRremote ZIP file that you downloaded from the link above.

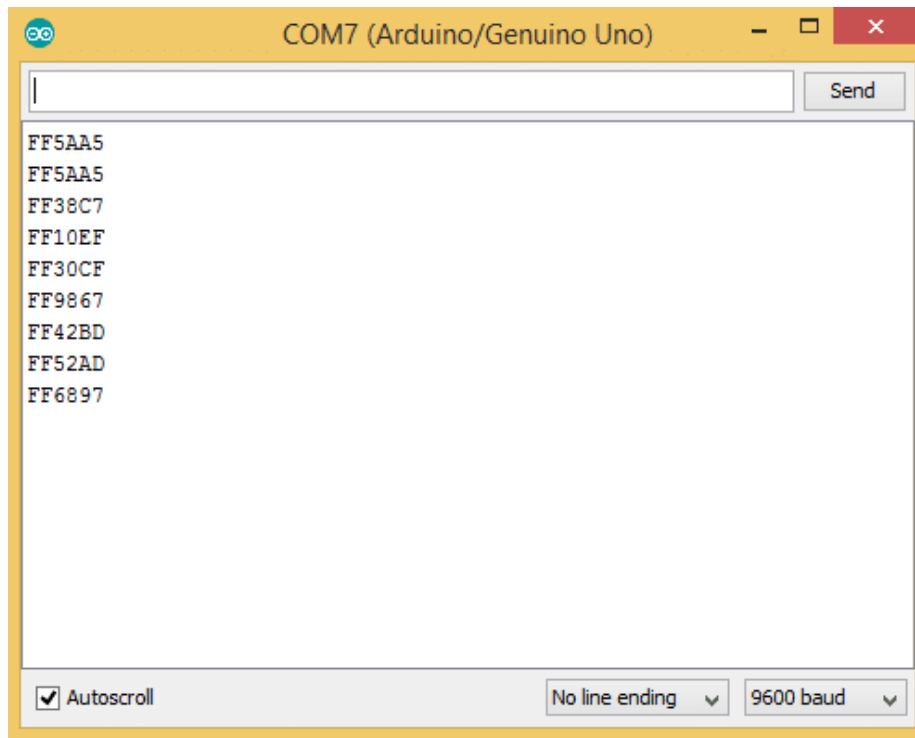
### FIND THE CODES FOR YOUR REMOTE

To find the key codes for your remote control, upload this code to your Arduino and open the serial monitor:



```
1 #include <IRremote.h>
2
3 const int RECV_PIN = 7;
4 IRrecv irrecv(RECV_PIN);
5 decode_results results;
6
7 void setup(){
8   Serial.begin(9600);
9   irrecv.enableIRIn();
10  irrecv.blink13(true);
```

Now press each key on your remote and record the hexadecimal code printed for each key press.



Using the program above, I derived a table of keys and their corresponding codes from the remote that came with my [HX1838 IR receiver and remote set](#). Note that you will receive a 0xFFFFFFFF code when you press a key continuously.

Key	Code
CH-	0xFFA25D
CH	0xFF629D
CH+	0xFFE21D
<<	0xFF22DD
>>	0xFF02FD
>	0xFFC23D
-	0xFFE01F
+	0xFFA857
EQ	0xFF906F
100+	0xFF9867

200+	0xFFB04F
0	0xFF6897
1	0xFF30CF
2	0xFF18E7
3	0xFF7A85
4	0xFF10EF
5	0xFF38C7
6	0xFF5AA5
7	0xFF42BD
8	0xFF4AB5
9	0xFF52AD

## FIND THE PROTOCOL USED BY YOUR REMOTE

Knowing which protocol your remote uses can be useful if you want to work on some more advanced projects. Or you might just be curious. The program below will identify the protocol used by your remote. It even works on most remote controls around your house.

```

1 #include <IRremote.h>
2
3 const int RECV_PIN = 7;
4 IRrecv irrecv(RECV_PIN);
5 decode_results results;
6
7 void setup(){
8   Serial.begin(9600);
9   irrecv.enableIRIn();
10  irrecv.blink13(true);

```

## PRINT KEYS TO THE SERIAL MONITOR

I extended the code above to print the key value instead of the hexadecimal code:

```

1 #include <IRremote.h>
2
3 const int RECV_PIN = 7;
4 IRrecv irrecv(RECV_PIN);
5 decode_results results;

```

```

6 unsigned long key_value = 0;
7
8 void setup(){
9   Serial.begin(9600);
10  }

```

If your remote sends different codes than the ones in the table above, just replace the hex code in each line where it says:

```

case 0xFFA25D:
  Serial.println("CH-");

```

In these lines, when the hex code 0xFFA25D is received, the Arduino prints "CH-".

## HOW THE CODE WORKS

For any IR communication using the IRremote library, first we need to create an object called `irrecv` and specify the pin number where the IR receiver is connected (line 3). This object will take care of the protocol and processing of the information from the receiver.

The next step is to create an object called `results`, from the `decode_results` class, which will be used by the `irrecv` object to share the decoded information with our application (line 5).

In the `void setup()` block, first we configure the serial monitor baud rate. Next we start the IR receiver by calling the `IRrecv` member function `enableIRIn()` (line 10).

The `irrecv.blink13(true)` function on line 11 will blink the Arduino's on board LED every time the receiver gets a signal from the remote control, which is useful for debugging.

In the `void loop()` block, the function `irrecv.decode` will return true if a code is received and the program will execute the code in the if statement. The received code is stored in `results.value`. Then I used a switch to handle each IR code and print the corresponding key value.

Before the switch block starts there is a conditional block:

```

if (results.value == 0xFFFFFFFF)
  results.value = key_value;

```

If we receive 0xFFFFFFFF from the remote, it means a repetition of the previous key. So in order to handle the repeat key pattern, I am storing the hex code in a global



variable `key_value` every time a code is received:

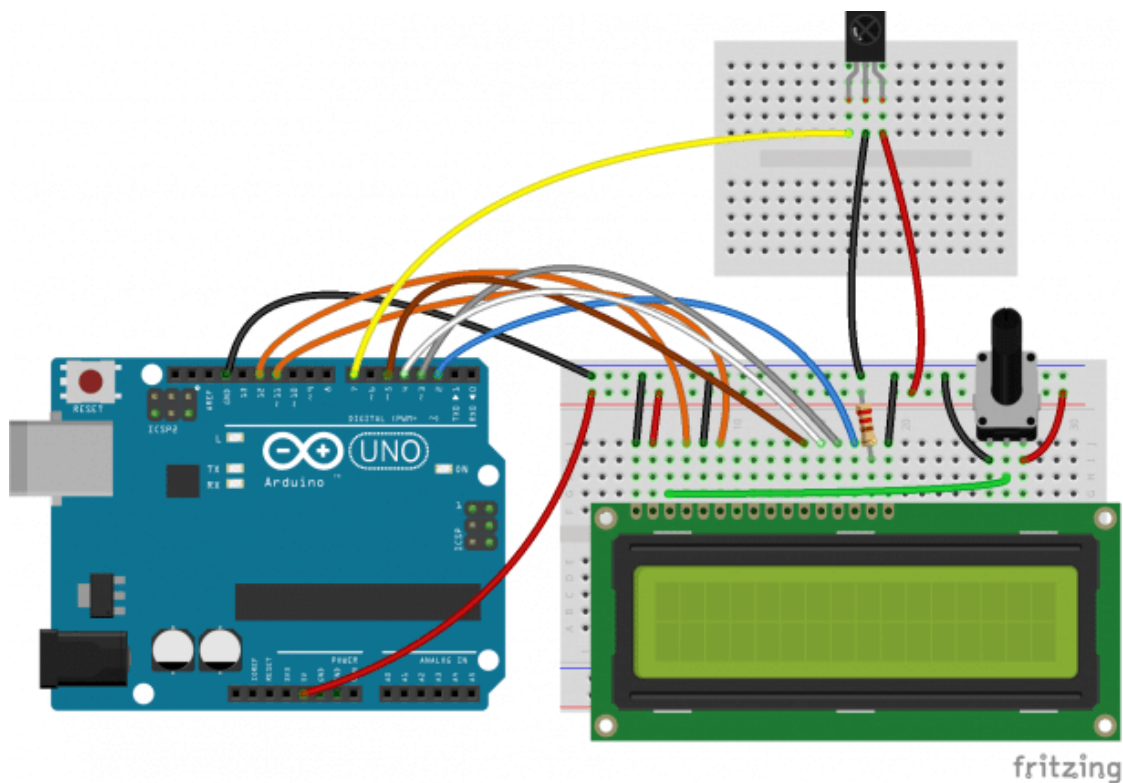
```
key_value = results.value;
```

When you receive a repeat pattern, then the previously stored value is used as the current key press.

At the end of the `void loop()` section, we call `irrecv.resume()` to reset the receiver and prepare it to receive the next code.

## PRINT KEYS TO AN LCD

Instead of printing the key values to the serial monitor, you can also display the information on an LCD. Check out our article on [setting up and programming an LCD on the Arduino](#) for more information on programming the LCD, but the basic setup looks like this:



The resistor sets the LCD's backlight brightness. It can be anything from 200 ohms to about 2K ohms. The potentiometer sets the character contrast. I normally use a 10K ohm potentiometer for this one.

Once everything is connected, upload this code to the Arduino:

```
1 #include <IRremote.h>
2 #include <LiquidCrystal.h>
3
4 const int RECV_PIN = 7;
```

```

5 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
6 IRrecv irrecv(RECV_PIN);
7 decode_results results;
8 unsigned long key_value = 0;
9
10 void setup(){

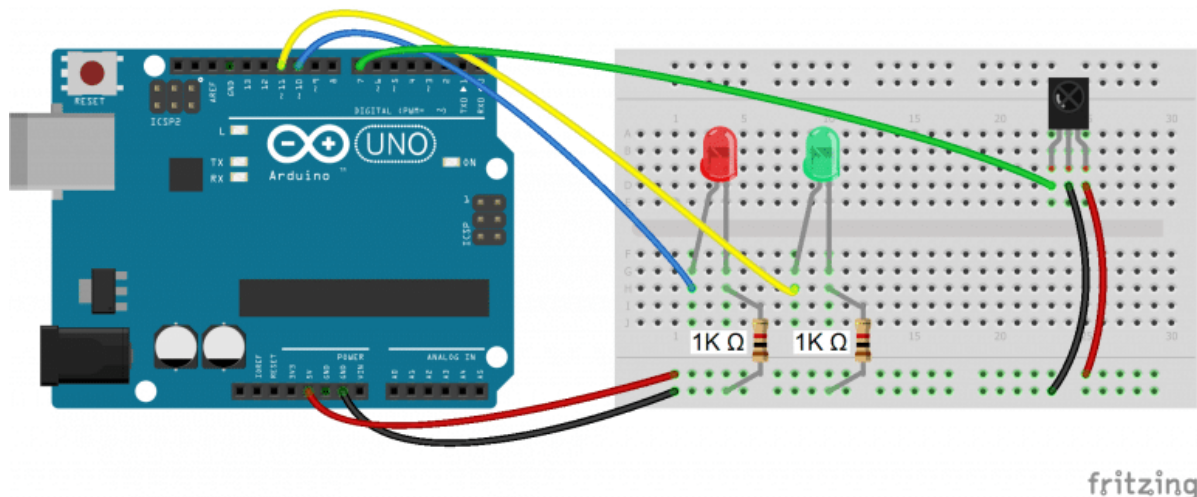
```

Again, if the hex codes don't match the codes output by your remote, just replace them for each character where it says `0XXXXXXXXX`;

## USING THE IR REMOTE TO CONTROL THINGS

Now I'll show you a simple demonstration of how you can use the IR remote to control the Arduino's output pins. In this example, we will light up an LED when a particular button is pressed. You can easily modify the code to do things like control servo motors, or activate relays with any button press from the remote.

The example circuit has the IR receiver connected to the Arduino, with a red LED connected to pin 10 and a green LED connected to pin 11:



The code below will write digital pin 10 HIGH for 2 seconds when the "5" button is pressed, and write digital pin 11 HIGH for 2 seconds when the "2" button is pressed:

```

1 #include <IRremote.h>
2
3 const int RECV_PIN = 7;
4 IRrecv irrecv(RECV_PIN);
5 decode_results results;
6 const int redPin = 10;
7 const int greenPin = 11;
8
9
10 void setup(){

```

So far we have covered the properties of infrared radiation and how communication happens between the transmitter and receiver. We saw how to identify the IR key codes for a given remote control. We learned how to display key presses on serial monitor and on an LCD screen. Finally I showed you how to control the Arduino's

output with the remote. Have fun playing with this and be sure to let us know in the comments if you have any questions or trouble setting this up!

*Krishna Pattabiraman is a frequent guest writer on Circuit Basics and the founder of [www.codeelectron.com](http://www.codeelectron.com).*

## JLCPCB - Prototype 10 PCBs for \$2 + 2 days Lead Time

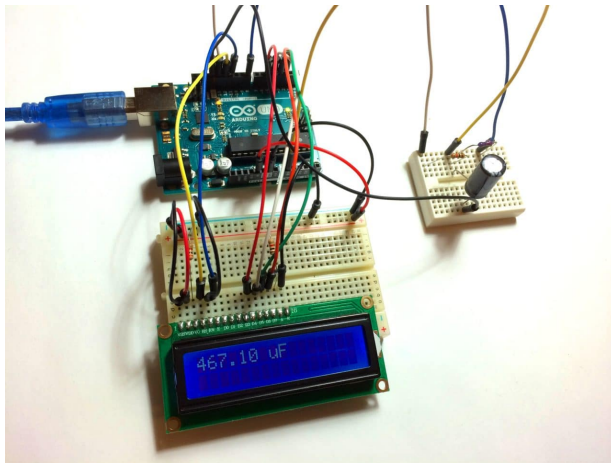
China's Largest PCB Prototype Enterprise, 300,000+ Customers & 10,000+ Online Orders Per Day

Get quote online easily: <https://jlcpcb.com/quote>

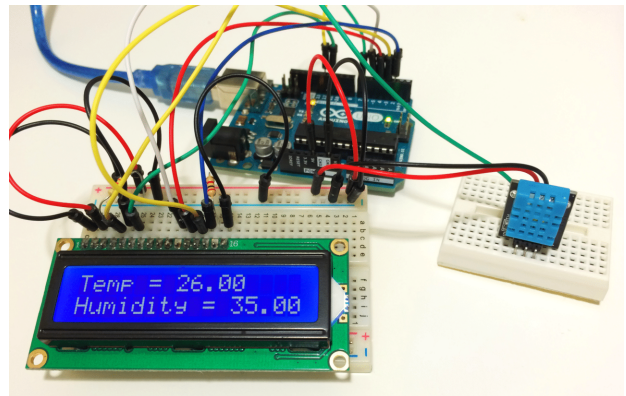
SHARE:



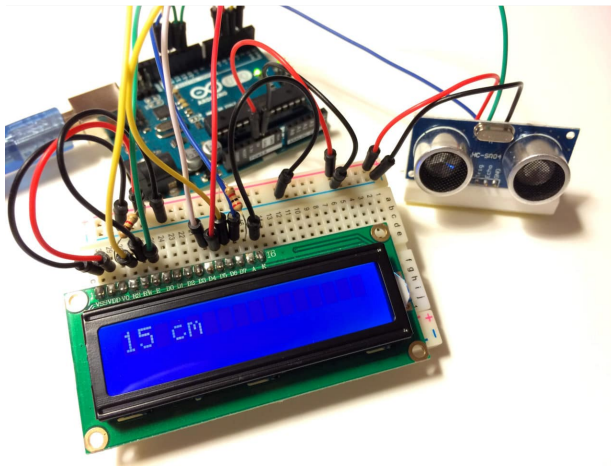
## RELATED POSTS



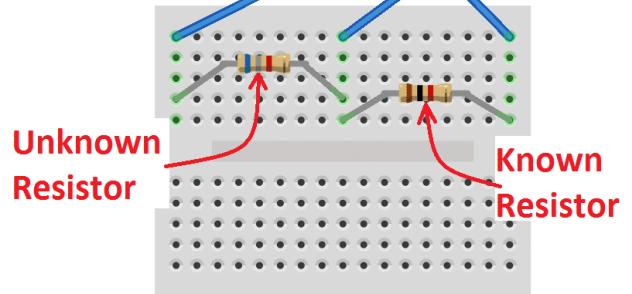
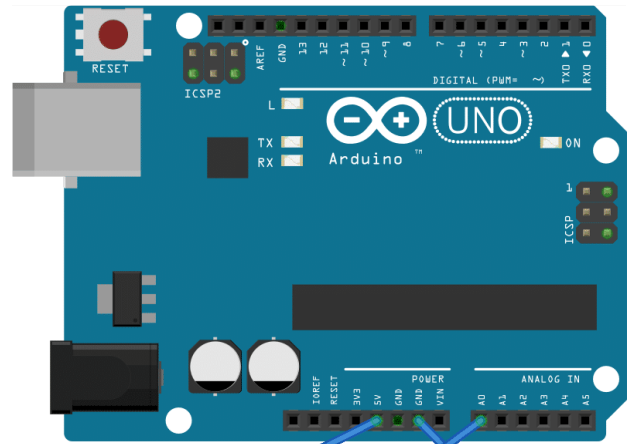
[How to Make an Arduino Capacitance Meter](#)



[How to Set Up the DHT11 Humidity Sensor on an Arduino](#)



How to Set Up an Ultrasonic Range Finder on an Arduino



fritzing

How to Make an Arduino Ohm Meter

● Cheap Laptop ▶	● Infrared ▶
● IR Remote ▶	● Ir Remote ▶
● Samsung ▶	● Universal ▶
● Programming ▶	● Pocket PC ▶
● PC Remote ▶	● IR Receiver ▶

## 6 COMMENTS

**StEw Lindenberger** on November 2, 2017 at 4:41 pm

Thank you.

A well presented, informative and useful overview including specific examples for implementation. Bravo.

REPLY

**Jeremy Borg** on November 3, 2017 at 10:18 am

A very well written and informative article. One thing I would have liked to learn more about is how to choose the IR emitter and receiver. My local store stocks several options of each, does it matter which one I choose?

<https://www.fabian.com.mt/en/products/webshop/bycategory/843/name/asc/18/1/infrared-uv-emitters-and-receivers.htm>

REPLY

**Stan** on November 3, 2017 at 3:26 pm

Well written, clear and concise. Keep it up.

REPLY

**Shekhar** on November 17, 2017 at 11:03 am

Are the codes complete?

REPLY

**Sajjad** on December 12, 2017 at 6:06 pm

Hello hope that you all are fine. my Ir reciever giving me continous values on serial monitor although i am sending no signals to it. kindly reply.

Thanks

REPLY

**peter obasa** on January 6, 2018 at 1:41 pm

Hi, please i try loading the code on uno and nano board this is the error message (

Build options changed, rebuilding all

C:\Program Files (x86)\Arduino\libraries\RobotIRremote\src\IRremoteTools.cpp:5:16:  
error: 'TKD2' was not declared in this scope



error: TKD2 was not declared in this scope

```
int RECV_PIN = TKD2; // the pin the IR receiver is connected to
```

```
^
```

```
exit status 1
```

```
Error compiling for board Arduino/Genuino Uno.
```

```
)
```

please what should i do , thanks in advance

REPLY

## LEAVE A REPLY

Your email address will not be published. Required fields are marked \*

COMMENT

NAME \*

EMAIL \*

WEBSITE

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

POST COMMENT

SEARCH ...

FOLLOW US





## SUBSCRIBE

Subscribe to get new tutorials sent straight to your inbox!

EMAIL ADDRESS

SUBSCRIBE

**J@LC JLCPCB**  
Low Cost Quick-Turn PCB  
Only **\$2** / 10pcs  
48 Hours Quick Turn >> **JLCPCB.COM**

Copyright **Circuit Basics**

[Raspberry Pi](#) [Arduino](#) [DIY Electronics](#) [Programming](#) [Videos](#) [Resources](#) [About](#) [Contact Us](#)

[Privacy Policy](#)

