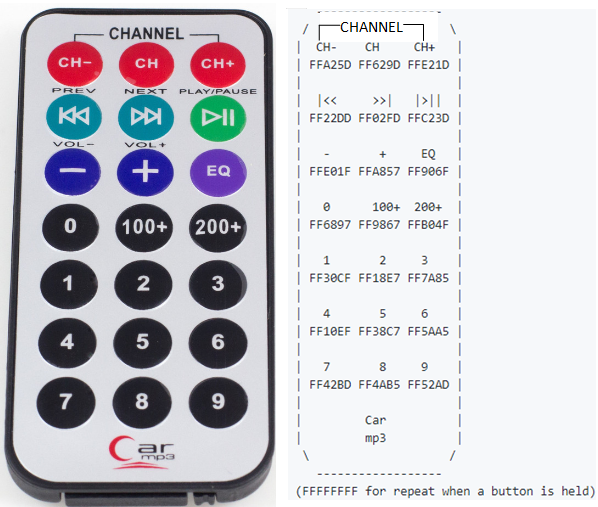
**Arduino IR Remote Description:**

With this cheap infrared controller, you can easily make remote control projects such as the Arduino IR Remote receiver thanks to the open-hardware community, there are also some [*IR*](https://roboelectrics.com/product-category/electronic-components/ir-leds/) libraries such the Joshua noble IR library for you to make everything in your house to be remote-control. This remote control offer 21 buttons, we suggest you use this remote control with the IR receiver.

Infrared IR Remote Control [Transmitter]:

**Infrared**[**IR remote**](https://roboelectrics.com/product-category/electronics-breakouts-and-modules/light-ir-radiation/)**control** is a remote control that sends out infrared light from an IR LED. Infrared is one of the frequencies or wavelengths found in the electromagnetic spectrum with a wavelength longer than that of the visible light (700nm-1mm), hence we cannot see most **infrared** radiations with our  eyes’ lenses, but, a smart phone camera’s lens can view it.

So, put your remote control under a smart phone camera and press a button, you will see the remote control **LED** blinking (at a frequency of 38,000 Hz or **38 KHz** per second). Information can be encoded into this wave at this frequency to make it unique and different from other *IR waves* in the environment, this process is called modulation, the information can then be sent to a receiver sensor which receives and demodulates the information for interpretation and usage with the help of a [***micro-controller***](https://roboelectrics.com/product-category/electronics-breakouts-and-modules/arduino-boards/).



[Arduino](https://roboelectrics.com/product-category/electronics-breakouts-and-modules/arduino-boards/) IR Remote Control Introduction

[](https://www.geeetech.com/wiki/index.php/File:Remote_control.jpg)

The new ultra-thin 38K **universal infrared remote control**, NEC encoding format 1-21-key remote control, USB port stereo, car MP3, foot bath, lighting, digital photo frame, microcontroller development board, learning board, etc..

Arduino IR Remote Control Feature

* **IR Remote control** distance: more than 8 meters
* Launch tube infrared wavelength: 940Nm
* Crystal: the oscillation frequency of 455 KHz
* IR carrier frequency: 38KHz
* Encoding: the encoding format of the NEC, upd6122 encoding scheme, the user code 00FF, key coding below picture.
* Size: 86 \* 40 \* 6mm
* Frequency: 38K
* *Power supply*: CR2025/160mAH
* Button: free height is less than 3mm, the force 200-350g, the life of more than 200 000

Arduino IR Remote Control Document

[IR Reomote library](http://www.geeetech.com/Documents/IRremote%20library.zip)

Usage

You need a **IR** receiving breakout to detect the **IR** signal and decodes it as HEX code, then display it on the serial monitor to read what the **remote control** send. Our price of Arduino IR Remote in Pakistan is low.

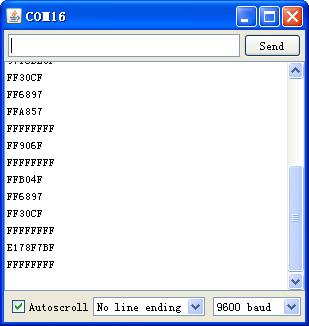
[](https://www.geeetech.com/wiki/index.php/File:IR_remote_control_2.jpg)

S -> D11

VCC -> 5V

GND -> GND

When you press any button on the remote control, serial monitor shows the hexadecimal code of that button.Every button on the remote control has a corresponding hexadecimal code. If you keep on pressing any button, it shows FFFFFFFF on the serial monitor. As well as the **IR remote control transmitter** and **receiver module**, keep the **IR LED** on the remote control opposite the **IR** detector for better signal reception.

[](https://www.geeetech.com/wiki/index.php/File:IR_remote_control_3.jpg)

Example Code

#include <IRremote.h>

int RECV\_PIN = 11; //define input pin on Arduino

IRrecv irrecv(RECV\_PIN);

decode\_results results;

void setup()

{

Serial.begin(9600);

irrecv.enableIRIn(); // Start the receiver

}

void loop() {

if (irrecv.decode(&results)) {

Serial.println(results.value, HEX);

irrecv.resume(); // Receive the next value

}

}

**IR, or infrared, communication** is a common, inexpensive, and easy to use wireless communication technology. IR light is very similar to visible light, except that it has a slightly longer wavelength. This means **IR** is undetectable to the human eye – perfect for wireless communication. For example, when you hit a button on your ***TV remote***, an *IR LED* repeatedly turns on and off, 38,000 time a second, to transmit information (like volume or channel control) to an *IR* photo sensor on your TV.

This tutorial will first explain the inner workings of common **IR** communication protocols. Then we will show how to use the micro bit with the *IR controller* to control the *RGB module*

# NEC Infrared Transmission Protocol

Frozen Content

Modified by Admin on Sep 13, 2017

The NEC IR transmission protocol uses pulse distance encoding of the message bits. Each pulse burst (mark – RC transmitter ON) is 562.5µs in length, at a carrier frequency of 38kHz (26.3µs). Logical bits are transmitted as follows:

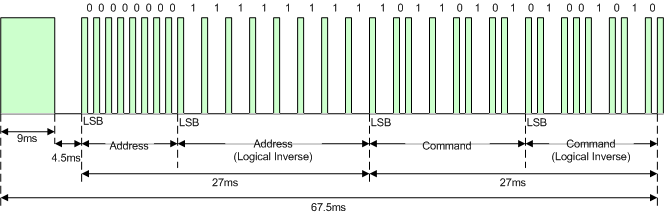
* Logical '0' – a 562.5µs pulse burst followed by a 562.5µs space, with a total transmit time of 1.125ms
* Logical '1' – a 562.5µs pulse burst followed by a 1.6875ms space, with a total transmit time of 2.25ms

When transmitting or receiving remote control codes using the NEC IR transmission protocol, the WB\_IRRC performs optimally when the carrier frequency (used for modulation/demodulation) is set to 38.222kHz.

When a key is pressed on the remote controller, the message transmitted consists of the following, in order:

* a 9ms leading pulse burst (16 times the pulse burst length used for a logical data bit)
* a 4.5ms space
* the 8-bit address for the receiving device
* the 8-bit logical inverse of the address
* the 8-bit command
* the 8-bit logical inverse of the command
* a final 562.5µs pulse burst to signify the end of message transmission.

The four bytes of data bits are each sent least significant bit first. Figure 1 illustrates the format of an NEC IR transmission frame, for an address of 00h (00000000b) and a command of ADh (10101101b).

  
Figure 1. Example message frame using the NEC IR transmission protocol.

Notice from Figure 1 that it takes:

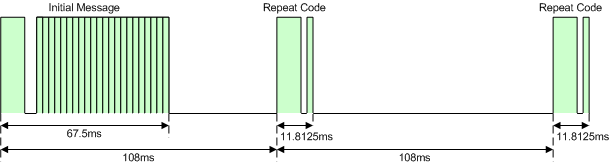
* 27ms to transmit both the 16 bits for the address (address + inverse) and the 16 bits for the command (command + inverse). This comes from each of the 16 bit blocks ultimately containing eight '0's and eight '1's - giving (8 \* 1.125ms) + (8 \* 2.25ms).
* 67.5ms to fully transmit the message frame (discounting the final 562.5µs pulse burst that signifies the end of message).

## REPEAT CODES

If the key on the remote controller is kept depressed, a repeat code will be issued, typically around 40ms after the pulse burst that signified the end of the message. A repeat code will continue to be sent out at 108ms intervals, until the key is finally released. The repeat code consists of the following, in order:

* a 9ms leading pulse burst
* a 2.25ms space
* a 562.5µs pulse burst to mark the end of the space (and hence end of the transmitted repeat code).

Figure 2 illustrates the transmission of two repeat codes after an initial message frame is sent.

  
Figure 2. Example repeat codes sent for a key held down on the transmitting remote controller.