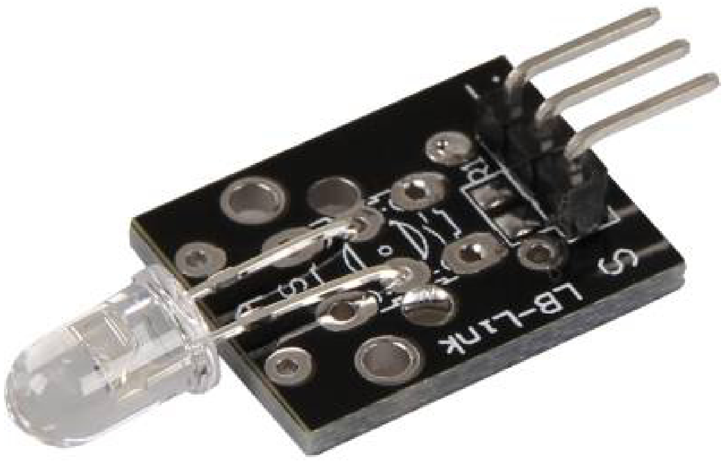
INFRARED EMITTER MODULE - HR0034

Specifications		
Function		IR Emitter
Model		KY-005
Forward Voltage		1.1 V
Forward Current		20 mA
Reception angle		± 45 °
Infrared Wavelength		940 nm
Dimensions		20 x 15 mm

KY-005 Infrared Transmitter module

Contents

Picture



Technical data / Short description

A LED which emits infrared light. A resistor might be necessary for some voltages.

Vf= 1,1V

If= 20mA

emitted wavelength: 940nm *(not visible for the human eye)*

Pre-resistor:

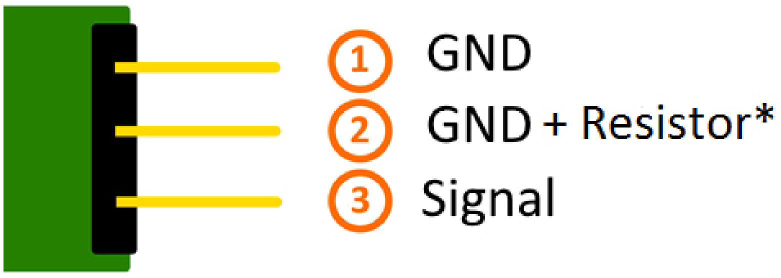
Rf (3,3V) = 120Ω

[used with ARM CPU-Core based microcontroller]

Rf (5V) = 220Ω

[used with Atmel Atmega based microcontroller]

Pinout



- You can directly solder a resistor to the circuit board. In that case, the central pin (2), which includes the resistor, can be used.

Code example Arduino

With both sensor modules, KY-005 and KY-022, you can build an infrared remote + infrared receiver system.
In order to do this, you will need the two sensor modules as well as two Arduinos.
The first one will handle the receiver system and the second one will handle the transmitter system.

An additional library is needed for this code example:

-[Arduino-IRremote] from Ken Shirriff | published under LGPL

The library is in the package and has to be copied before the start into the library folder.

You can find your Arduino Library folder at: C:\User\[UserName]\Documents\Arduino\libraries

There are different infrared protocolls to send data. In this example we use the RC5 protocol.
The used library "Arduino-IRremote" converts the data independently.
The library has additional protocolls, they are marked in this documentation.

Code for the receiver:

```
// Arduino-IRremote library will be added
#include <IRremote.h>
#include <IRremoteInt.h>

// You can declare the input pin for the signal output of the KY-022 here
int RECV_PIN = 11;

// Arduino-IRremote library will be initialized
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Infrared receiver will start
}

// main program loop
void loop() {

  // It will be checked if the receiver has gotten a signal.
  if (irrecv.decode(&results)) {
    //At signal input, the received and decoded signal will show via serial console.
    Serial.println(results.value, HEX);
    irrecv.resume();
  }
}
```

Code for the transmitter:

```
//Arduino-IRremote library will be added
#include <IRremote.h>
#include <IRremoteInt.h>

//...and here initialized
IRsend irsend;

// The configuration of the output pin will be made by the library
// The output pin is a different one for different arduinos
// Arduino UNO:  Output = D3
// Arduino MEGA: Output = D9
// You will find a full list of output pins on the website:
// http://z3t0.github.io/Arduino-IRremote/
void setup()
{
}

// main program loop
void loop() {
        // The transmitter sends the signal A90 (hex. dezimal form) in the encoding "RC5"
        // It will be transmitted 3 times after that it will make a 5 second break
    for (int i = 0; i < 3; i++) {
        irsend.sendRC5(0xA90, 12); //[12] Bit-length signal (hex A90=1010 1001 0000)
        delay(40);
    }
    delay(5000); // 5 second break between the sending impulses
}
```

Example program download:

KY-005_KY-022_Infrared-Modules_ARD

Connections Arduino 1 [Receiver]:

KY-022

Signal	=	[Pin 11]
+V	=	[Pin 5V]
GND	=	[Pin GND]

Connections Arduino 2 [Transmitter]:

KY-005

Signal	= [Pin 3 (Arduino Uno) | Pin 9 (Arduino Mega)]
GND+resistor	= [Pin GND*]
GND	= [Pin GND]

- * Only if resistor was soldered to the circuit board.

Code example Raspberry Pi

Code example remote

Because of its progressive processor architecture, the Raspberry Pi has a big advantage, compared to the Arduino.

It can run a full Linux OS.
With help of an infrared-receiver, it is not only able to transmit simple data signals, furthermore it can control complete programs via remote.

To setup an infrared control system, we recommend to use the Linux software "lirc" (published under the LGPL-Website).
In the following section, we show you how to use lirc and how the remotely send the learned signals via infrared.

On this purpose, the module KY-005 will be used as an infrared-transmitter and the KY-022 will be used as an infrared-receiver.

Connections Raspberry Pi:

KY-005

Signal	= GPIO17	[Pin 11]
GND+resistor	= GND*	[Pin 9]
GND	= GND	[Pin 6]

- * Only if a resistor was soldered to the module

KY-022

Signal	=	GPI18	[Pin 12]
+V	=	3,3V	[Pin 17]
GND	=	GND	[Pin 25]

Lirc Installation

Open a terminal at the desktop or use SSH to log into your Raspberry Pi. To install lirc, enter the following command:

```
sudo apt-get install lirc -y
```

[For this the Raspberry Pi has to be connected to the internet]

To use the lirc module immediately after starting the OS, you have to add the following line to the end of the file "/boot/config.txt":

```
    dtoverlay=lirc-rpi,gpio_in_pin=18,gpio_out_pin=17,gpio_in_pull=up
```

The "gpio_in_pin=18" will be defined as an input pin of the IR-receiver and the "gpio_out_pin=17" as an output pin of the IR-transmitter.

The file can be edited by entering the command:

```
sudo nano /boot/config.txt
```

You can save and close the file via the key sequence [ctrl+x -> y -> enter]

You will also have to modify the file "/etc/lirc/hardware.conf" by entering the command:

```
sudo nano /etc/lirc/hardware.conf
```

In this file you have to change following lines:

```
    DRIVER="UNCONFIGURED"
    --->>
    DRIVER="default"
    DEVICE=""
    --->>
    DEVICE="/dev/lirc0"
    MODULES=""
    --->>
    MODULES="lirc_rpi"
```

The modified file should now look like:

```
# /etc/lirc/hardware.conf
#
# Arguments which will be used when launching lircd
LIRCD_ARGS=""

#Don't start lircmd even if there seems to be a good config file
#START_LIRCMD=false

#Don't start irexec, even if a good config file seems to exist.
#START_IREXEC=false

#Try to load appropriate kernel modules
LOAD_MODULES=true

# Run "lircd --driver=help" for a list of supported drivers.
DRIVER="default"
# usually /dev/lirc0 is the correct setting for systems using udev
DEVICE="/dev/lirc0"
MODULES="lirc_rpi"

# Default configuration files for your hardware if any
LIRCD_CONF=""
LIRCMD_CONF=""
```

After that we reboot the Raspberry Pi with the following command:

```
sudo reboot
```

IR-Receiver Test

To test the connected receiver, you have to close lirc first with the following command:

```
sudo /etc/init.d/lirc stop
```

After that, you can test if signals could be detected on the Raspberry Pi by using the following command:

```
mode2 -d /dev/lirc0
```

and by pressing random keys on an infrared remote. You should see numbers in the following form:

```
space 95253
pulse 9022
space 2210
pulse 604
space 95246
pulse 9019
space 2211
pulse 601
space 95252
pulse 9019
space 2210
```

```
pulse 603
space 95239
pulse 9020
space 2208
pulse 603
...
```

You can restart lirc with the following command:

```
sudo /etc/init.d/lirc start
```

Remote teach

To register an infrared-remote at the lirc system, you have to configure the file "/etc/lirc"lircd.conf".

In this file, all command assignments of the infrared codes are saved.

To get a good formatted lircd.conf, use the lirc assistant software which creates the file automatically.

To start this process you have to stop lirc first by using the command:

```
sudo /etc/init.d/lirc stop
```

With the following command, we can start the assistant:

```
irrecord -d /dev/lirc0 ~/MeineFernbedienung.conf
```

The assistant will start an initialization of the remote, in this initialization you have to press a few keys so that the lirc system is able to learn the encoding of the remote. For that, please follow the instructions of the assistant. After the initialization, the assistant asks for the name of the key which should get a new infrared code. You can choose your key from the following file:

FernbedienungsCodes.txt

You have to type these into the assistant and need to confirm with enter. After this, the recording of the infrared code for the chosen key will start.

Example: type in [KEY_0] - - -> confirm with enter - - -> press key 0 of the remote - - -> waiting for the assistant to confirm the recording.

If no more keys need to be configured, you can close the assistant by pressing the enter key. After this, the configuration file is created, but you have to choose a name for the recorded remote. For this we have to open the file with the editor:

```
sudo nano ~/MeineFernbedienung.conf
```

Here you have to change the line:

```
    name  /home/pi/MeineFernbedienung.conf
```

to

```
    name  MeineFernbedienung
```

Please don't use any spaces or additional characters in the name.

You can save and close the file with the key sequence [ctrl+x ---> y ---> enter].

After creating the configuration, you can make a backup for original lircd.conf with the following command:

```
sudo mv /etc/lirc/lircd.conf /etc/lirc/lircd.conf.bak
```

With the command

```
sudo cp ~/MeineFernbedienung.conf /etc/lirc/lircd.conf
```

you can use the before created file for the lirc system.

Now you can start the lirc system again with the command:

```
sudo /etc/init.d/lirc start
```

From now on, the remote is known and can be used with the right software. Alternatively you can use the following command to test the functions:

```
irw
```

Sending a command via Infrared Transmitter

If you want to control devices, like your Television, via Raspberry Pi, you can now send the learned commands with the infrared transmitter. With that you can build a software controlled infrared controller or you can use the internet or the network to switch single devices on and off.

First we check with the following command:

```
irsend LIST MeineFernbedienung ""
```

which assigments are available for the remote.

Now we can send the command [KEY_0] with the command:

```
irsend SEND_ONCE MeineFernbedienung KEY_0
```

You can have the example above in other variations like , instead of sending the signal only once , it will be send multiple times.

```
irsend SEND_START MeineFernbedienung KEY_0
```

After this, the code [KEY_0] will be repeatly send out until we end it with the following command:

```
irsend SEND_STOP MeineFernbedienung KEY_0
```