Welcome!

Thank you for purchasing our *AZ-Delivery KY-005 Infrared LED Module.* On the following pages, we will introduce you to how to use and set up this handy device.

Have fun!

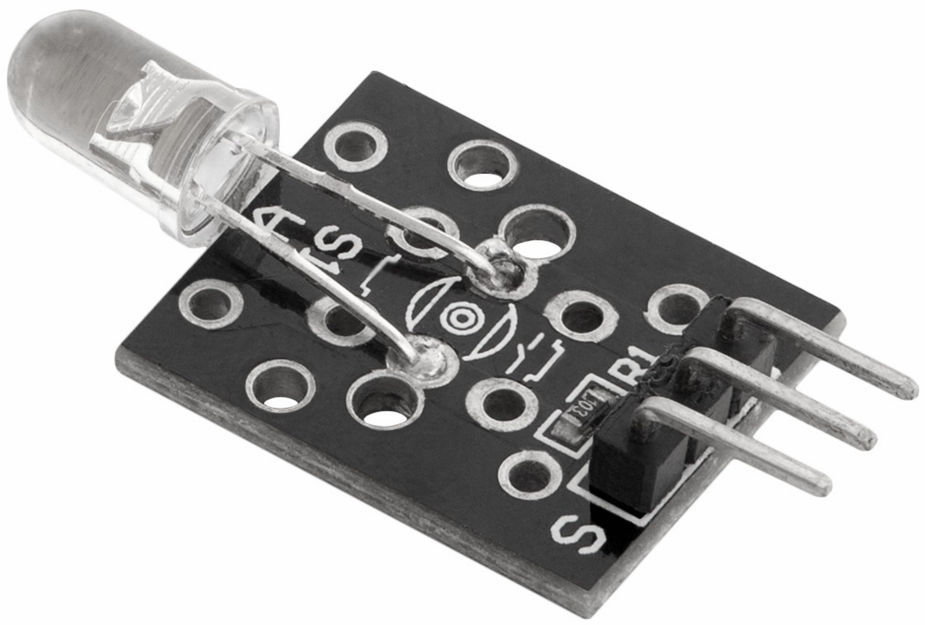


Table of Contents

Introduction

The KY-005 Infrared LED module emits infrared light at the frequency of *38kHz*. The module consists just of a *5mm* infrared LED. The module, together with the KY-022 Infrared receiver module is great for creating infrared interfaces.

Infrared light-emitting diode (IR LED) is a special purpose LED that emits infrared rays ranging from *700nm* to *1um* wavelength. Different infrared LEDs may produce infrared light of differing wavelengths, just like different LEDs produce different colours light. Infrared LEDs are usually made of gallium arsenide or aluminum gallium arsenide. In complement with infrared receivers, these are commonly used as sensors.

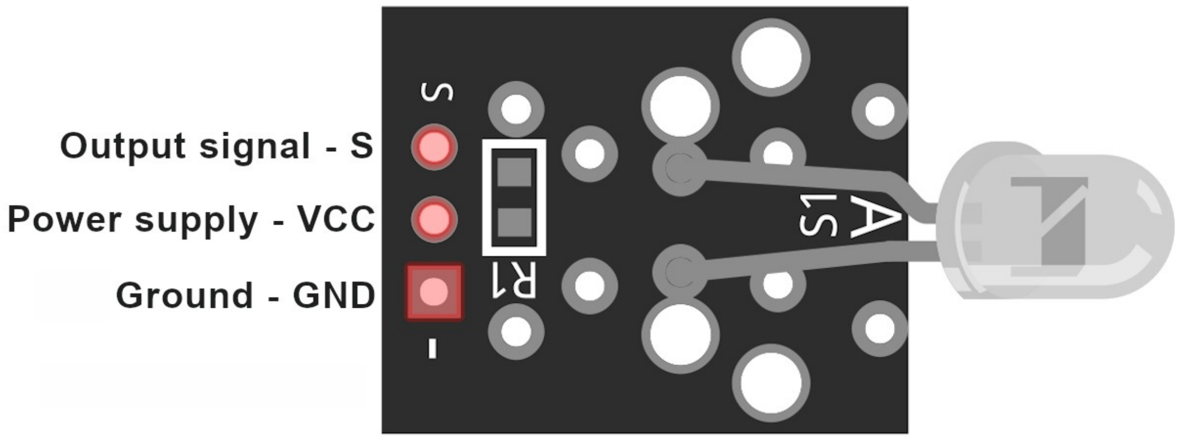
Specifications

»	Operating voltage range:	from 3.3V to 5V DC
»	Forward current:	30 ~ 60 mA
»	Power consumption:	90mW
»	Operating temperature:	-25°C to 80°C [-13°F to 176°F]
»	Dimensions:	19 x 15mm [0.73 x 0.6in]

The appearance of the infrared LED is the same as a common LED. Infrared LEDs are mostly used in the TV remotes. Since the human eye cannot see the infrared radiation. A person can not identify if an infrared LED is working or not. But a camera on mobile phone solves this problem. The rays from the infrared LED are shown on the following image:

The pinout

The KY-005 infrared LED module has three pins. The pinout diagram is shown on the following image:



Working principle of TV remotes

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

In the infrared signal modulation, an encoder on the infrared remote converts a binary signal into a modulated electrical signal. This electrical signal is sent to the infrared LED. The LED converts the modulated electrical signal into a modulated infrared light signal. The infrared receiver or photodiode then demodulates the infrared light signal and converts it back to electroc (binary), before passing the information to a microcontroller. The modulated infrared signal is a series of infrared light pulses, switched *ON*/*OFF* at a high frequency, known as the carrier frequency. The carrier frequency used by most transmitters is *38kHz* because it is rare in nature and it can be distinguished from ambient noise. This way the infrared receiver knows that the *38kHz* signal was sent from the transmitter and not picked up from the surrounding environment.

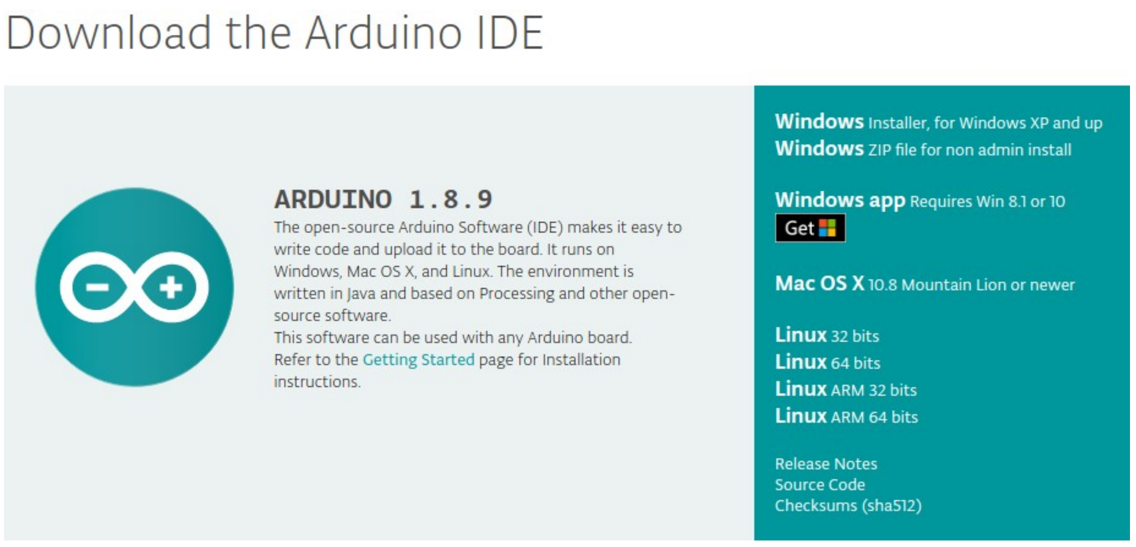
The receiver diode detects all frequencies of infrared light, but it has a band pass filter and only lets through infrared light at *38kHz* frequency. It then amplifies the modulated signal with a pre-amplifier and converts it to a binary signal, before sending it to a microcontroller. The pattern in which the modulated infrared signal is converted to binary is defined by a transmission protocol.

There are many infrared transmission protocols: *Sony*, *Matsushita*, *NEC*, and *RC5* are some of the more common protocols. The *NEC* protocol is also the most common type in embedded projects, so it is used in this eBook as an example to show how the transiver converts the modulated infrared signal to a binary one. Logical *1* starts with a *562.5μs* long *HIGH* pulse at *38kHz* infrared light followed by a *1687.5μs* long *LOW* pulse. Logical *0* is transmitted with a *562.5μs* long *HIGH* pulse followed by a *562.5μs* long *LOW* pulse. Other protocols differ only in the duration of the individual *HIGH* and *LOW* pulses.

Each time a button on the remote control is pressed, a unique hexadecimal code is generated. This is the information that is modulated and sent over infrared light to the receiver. 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 the code generated for each key on the particular remote has to be determined. It can be found in the datasheet (the infrared key codes should be listed in datasheet). There is a simple sketch that reads most of the popular remote controls and display the hexadecimal codes to the Serial Monitor when a key is pressed (later in the text).

How to set-up Arduino IDE

If the Arduino IDE is not installed, follow the *link* and download the installation file for the operating system of choice.



Download the Arduino IDE

ARDUINO 1.8.9

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
Get

Mac OS X 10.8 Mountain Lion or newer

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

Release Notes
Source Code
Checksums (sha512)

For `Windows` users, double click on the downloaded *`.exe`* file and follow the instructions in the installation window.

For *Linux* users, download a file with the extension *.tar.xz*, which has to be extracted. When it is extracted, go to the extracted directory and open the terminal in that directory. Two *.sh* scripts have to be executed, the first called *arduino-linux-setup.sh* and the second called *install.sh*.

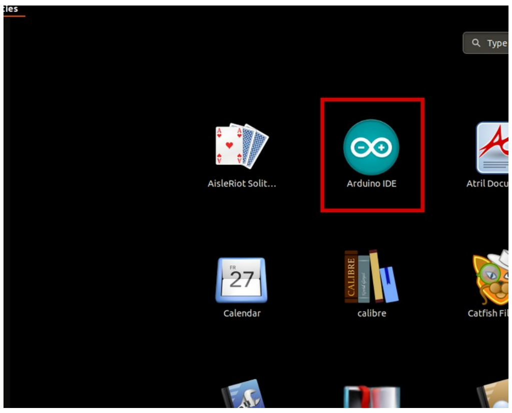
To run the first script in the terminal, open the terminal in the extracted directory and run the following command:

sh arduino-linux-setup.sh user_name

user_name - is the name of a superuser in the Linux operating system. A password for the superuser has to be entered when the command is started. Wait for a few minutes for the script to complete everything.

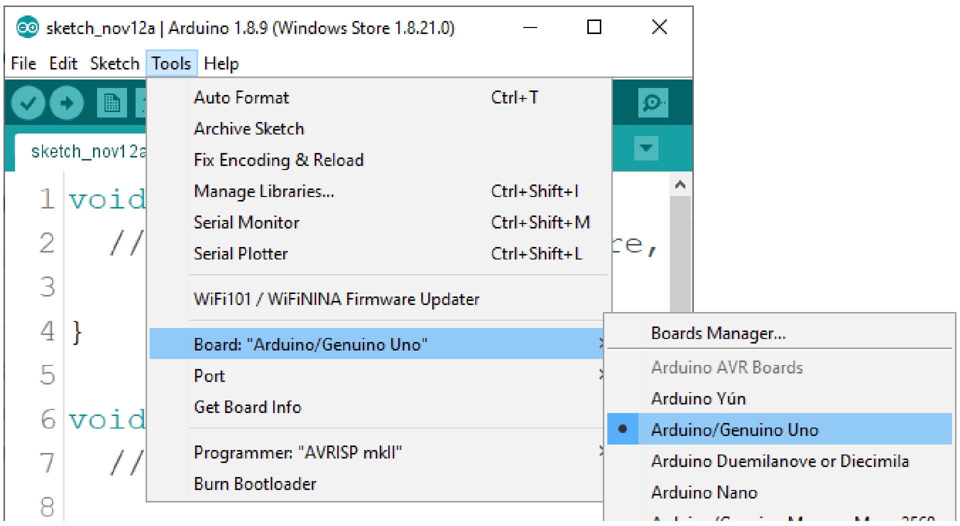
The second script called *install.sh* script has to be used after installation of the first script. Run the following command in the terminal (extracted directory): **sh install.sh**

After the installation of these scripts, go to the *All Apps*, where the *Arduino IDE* is installed.



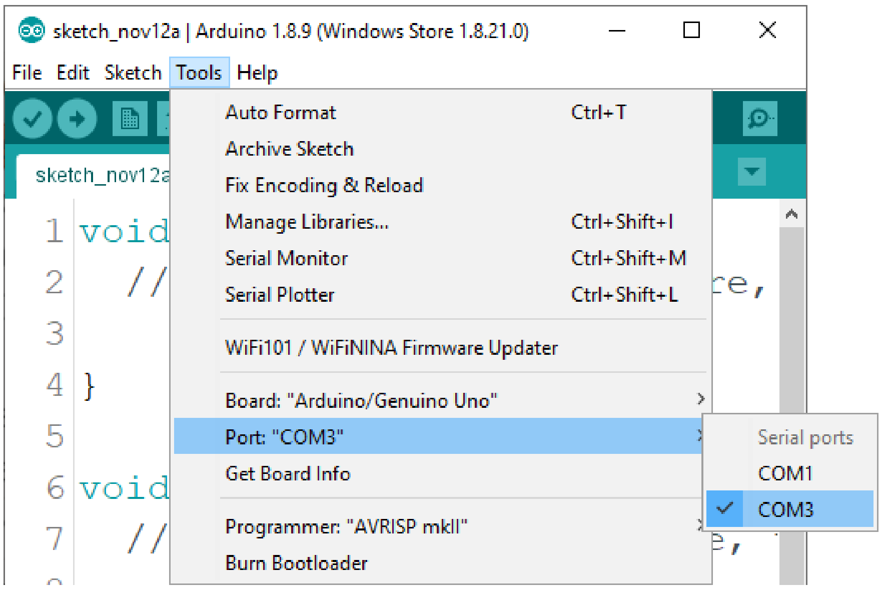
Almost all operating systems come with a text editor preinstalled (for example, *`Windows`* comes with *`Notepad`*, *`Linux Ubuntu`* comes with *`Gedit`*, *`Linux Raspbian`* comes with *`Leafpad`*, etc.). All of these text editors are perfectly fine for the purpose of the eBook.

Next thing is to check if your PC can detect an Arduino board. Open freshly installed Arduino IDE, and go to:
`Tools > Board > {your board name here}`
`{your board name here}` should be the *`Arduino/Genuino Uno`*, as it can be seen on the following image:


sketch_nov12a | Arduino 1.8.9 (Windows Store 1.8.21.0)
File Edit Sketch Tools Help
sketch_nov12a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino/Genuino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
Arduino Yún
Arduino/Genuino Uno
Arduino Duemilanove or Diecimila
Arduino Nano

The port to which the Arduino board is connected has to be selected. Go to:
`Tools > Port > {port name goes here}`
and when the Arduino board is connected to the USB port, the port name can be seen in the drop-down menu on the previous image.

If the Arduino IDE is used on Windows, port names are as follows:


sketch_nov12a | Arduino 1.8.9 (Windows Store 1.8.21.0)
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM3"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1
✓ COM3

For *Linux* users, for example port name is */dev/ttyUSBx*, where *x* represents integer number between *0* and *9*.

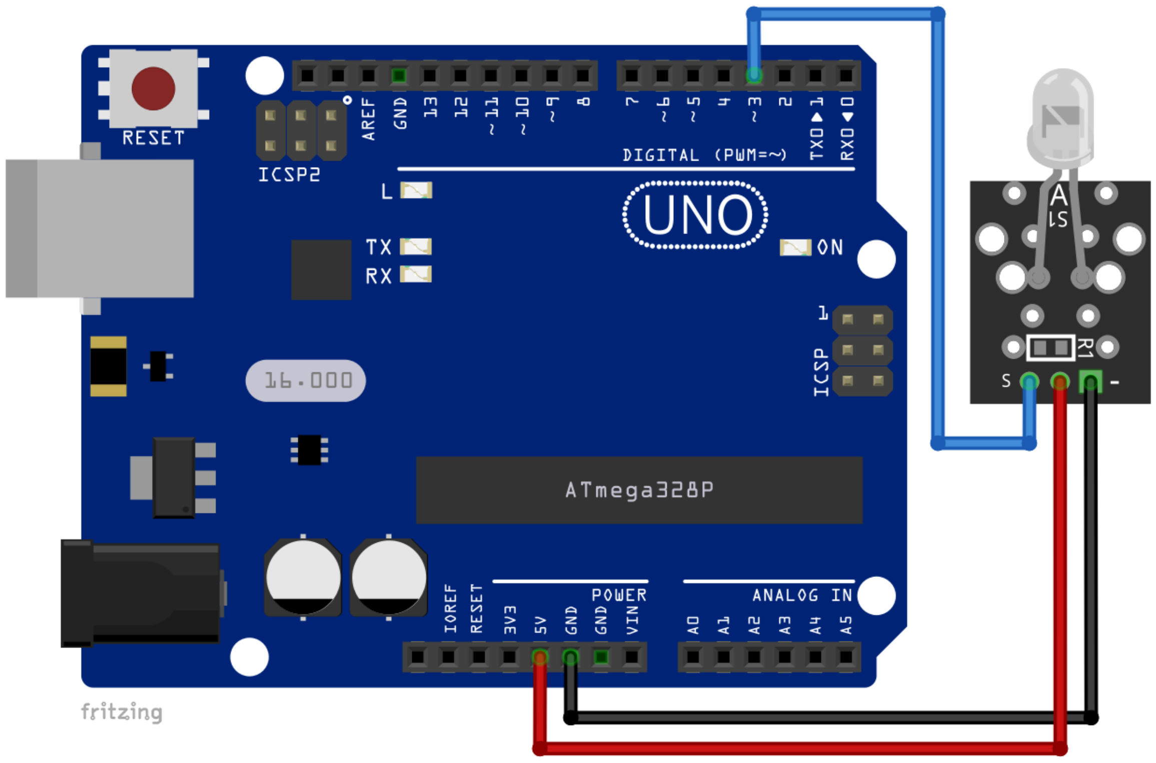
How to set-up the Raspberry Pi and Python

For the Raspberry Pi, first the operating system has to be installed, then everything has to be set-up so that it can be used in the *Headless* mode. The *Headless* mode enables remote connection to the Raspberry Pi, without the need for a *PC* screen Monitor, mouse or keyboard. The only things that are used in this mode are the Raspberry Pi itself, power supply and internet connection. All of this is explained minutely in the free eBook:
Raspberry Pi Quick Startup Guide

The *Raspbian* operating system comes with *Python* preinstalled.

Connecting the module with Uno

Connect the KY-005 module with the Uno as shown on the following connection diagram:



KY-005 pin	**>**	**Uno pin**	
S	>	D3	**Blue wire**
Middle pin (VCC)	>	5V	**Red wire**
- (GND)	>	GND	**Black wire**

Basic sketch example

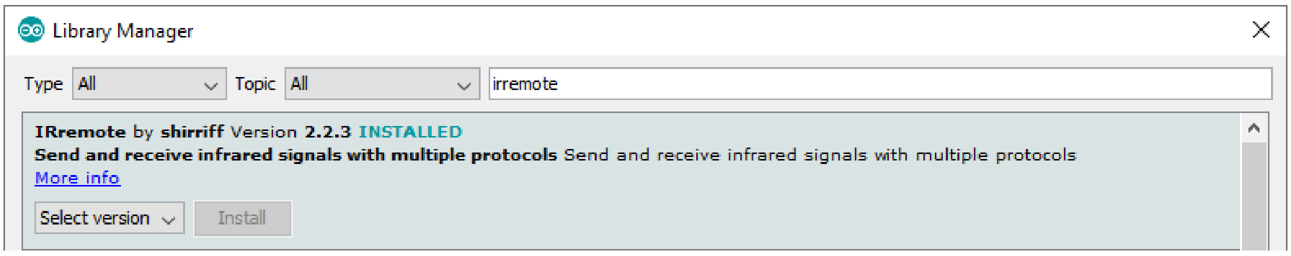
```
#define LED 3 # digital I/O pin with PWM feature “~”
uint8_t brightness = 200;

void setup() {
  pinMode(LED, OUTPUT);
  analogWrite(LED, brightness);
}

void loop() { }
```

The sketch is used to turn *ON* the infrared LED on-board the KY-005 module.

Library for Arduino IDE

To use the KY-005 module with Uno as TV remote it is recommended to install an external library. Open Arduino IDE and go to:
`Tools > Manage Libraries`

When a new window opens, type *`IRremote`* (two R's) in the search box and install a library called *`IRremote`* made by *`shirriff`*, as shown on the following image:



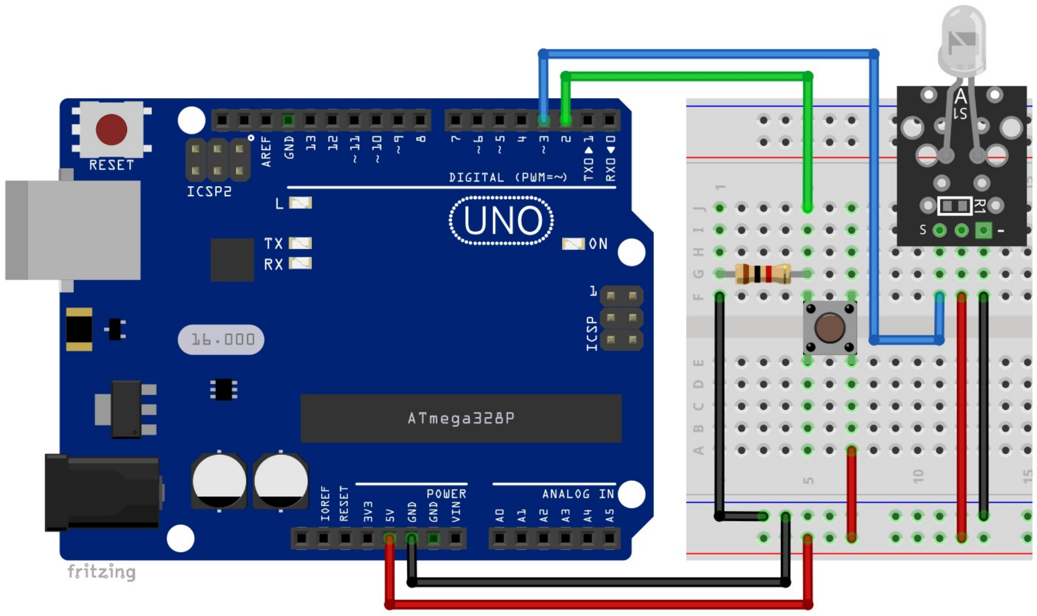
Library Manager

Type: All | Topic: All | irremote

IRremote by **shirriff** Version **2.2.3** **INSTALLED**
Send and receive infrared signals with multiple protocols Send and receive infrared signals with multiple protocols
More info

TV remote with the KY-005 (Uno)

In this chapter the TV remote is simulated using the KY-005 infrared LED module. Connect the KY-005 module with the Uno as shown on the following connection diagram:



KY-005 pin	**>**	**Uno pin**	
S	>	D3	**Blue wire**
Middle pin (VCC)	>	5V	**Red wire**
- (GND)	>	GND	**Black wire**

Button pin	**>**	**Uno pin**	
Bottom right pin	>	5V	**Red wire**
Top left pin	>	D2	**Green wire**
Top left pin (via resistor*)	>	GND	**Black wire**

* Resistor resistance value is *1k*Ω

Sketch code

```
#include <IRremote.h>
#define BUTTON_PIN 2
uint8_t button_state = 0, toggle_state = 0;
IRsend transmitter;
void setup() {
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT);
}
void loop() {
  button_state = digitalRead(BUTTON_PIN);
  Serial.println(button_state);
  if (button_state == HIGH) {
    if (toggle_state == 0){
       # ON/OFF button code of some TV remote
      transmitter.sendNEC(0x807F00FF, 32);
      digitalWrite(LED_BUILTIN, HIGH);
      toggle_state = 1;
    }
    else { // if you hold the button
      transmitter.sendNEC(0xFFFFFFFF, 32);
    }
  }
  else {
    digitalWrite(LED_BUILTIN, LOW);
    toggle_state = 0;
  }
  delay(200);
}
```

At the beginning of the sketch the *IRremote.h* library is included. Next, one macro and two variables are created, where the macro represents the digital I/O pin number to which the push button is connected (*2*). The variables are used to detect the state of the push button. After that, the *transmitter* object is created.

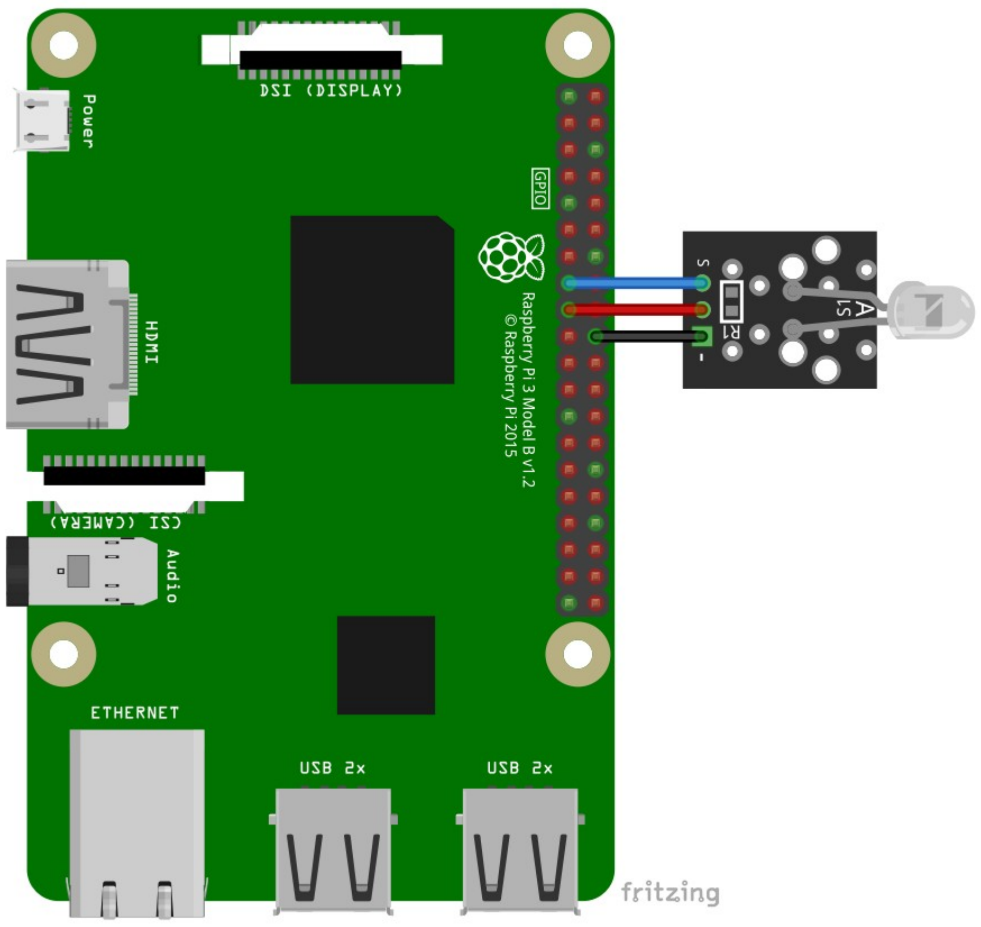
In the *setup()* function, the serial communication is started with baud rate of *9600bps*. The pin modes for digital pin *13* is set to *OUTPUT* and for the button pin (*2*) mode is set to *INPUT*.

In the *loop()* function, the state of the push button is read and displayed in the Serial Monitor. Next, the state of the push button is checked if it is in the *HIGH* state. If it is, the state of a *toggle_state* variable is toggled, the *NEC* code is sent via an infrared LED and the on-board LED of the Uno is turned *ON*. If the push button is pressed for too long, the state of the *toggle_state* variable is not toggled anymore, and the *NEC* code *0xFFFFFFFF* is sent to simulate TV remote behaviour. The remote outputs number *0xFFFFFFFF* when the specific button is pressed and held (an indication that the button is held). If the state of the push button is *LOW*, the on-board LED of the Uno is turned *OFF* and the state of the *toggle_state* variable is set to zero value. At the end of the *loop()* function, the delay of *200* milliseconds is set.

The code that is sent is for turning ON/OFF one specific TV. To determine this code, the KY-022 module is used. Read about this in the eBook for the KY-022 module.

Connecting the module with Raspberry Pi

Connect the KY-005 module with the Raspberry Pi as shown on the following connection diagram:



KY-005 pin	**>**	**Raspberry Pi pin**		
S	>	GPIO22	[pin 15]	**Blue wire**
Middle pin (VCC)	>	3V3	[pin 17]	**Red wire**
- (GND)	>	GND	[pin 20]	**Black wire**

Python script

```
import RPi.GPIO as GPIO
from time import sleep
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
# Pin setup for the module
IRLED = 22
GPIO.setup(IRLED, GPIO.OUT)
print('[Press CTRL + C to end the script!]')
try: # Main program loop
     while True:
          # PWM 500Hz, duty cucle 50%
          GPIO.output(IRLED, GPIO.HIGH)
          sleep(0.001)
          GPIO.output(IRLED, GPIO.LOW)
          sleep(0.001)
except KeyboardInterrupt:
     print('\nScript End!')
finally:
     GPIO.cleanup()
```

The script just turns *ON* the infrared LED on-board the KY-005 module (using PWM). Save the script by the name *ky005.py*. To run the script, open the terminal in the directory and run the following command:

```
python3 ky005.py
```

To end the script press *CTRL + C* on the keyboard.

Now it is the time to learn and make your own projects. You can do that with the help of many example scripts and other tutorials, which can be found on the internet.

If you are looking for the high quality products for Arduino and Raspberry Pi, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.

https://az-delivery.de

Have Fun!

Impressum

https://az-delivery.de/pages/about-us