








# Lampiran

## 1. Link Google Drive demo alat dan datasheet modul :

<https://drive.google.com/drive/folders/1PEGCgNDJX0KC7IhX3UMFkHOFdpYhk1Mn?usp=sharing>

## 2. Datastream pada Blynk

<input type="checkbox"/>	Id	Name	Alias	Color	Pin	Data Type	Units	Is R
<input checked="" type="checkbox"/>	1	Beri Pakan	Beri Pakan		V0	Integer		fals
<input checked="" type="checkbox"/>	2	hari	hari		V1	String		fals
<input checked="" type="checkbox"/>	3	tanggal	tanggal		V2	String		fals
<input checked="" type="checkbox"/>	4	waktu	waktu		V3	String		fals
<input checked="" type="checkbox"/>	5	jarak	jarak		V4	Integer	cm	fals
<input type="checkbox"/>	6	suhu	suhu		V5	Double	°C	fa
<input checked="" type="checkbox"/>	8	Infrared	Infrared		V7	Integer		fa

## 3. Datasheet Modul



# **ESP8266EX Datasheet**

**Version 4.3**

Espressif Systems IOT Team

<http://bbs.espressif.com/>

Copyright © 2015



## Disclaimer and Copyright Notice

Information in this document, including URL references, is subject to change without notice. THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

The Wi-Fi Alliance Member Logo is a trademark of the WiFi Alliance.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.

Copyright © 2015 Espressif Systems. All rights reserved.



# Table of Contents

- 1. General Overview .....6
  - 1.1. Introduction .....6
  - 1.2. Features .....7
  - 1.3. Parameters .....7
  - 1.4. Ultra Low Power Technology .....9
  - 1.5. Major Applications.....9
- 2. Hardware Overview.....11
  - 2.1. Pin Definitions .....11
  - 2.2. Electrical Characteristics.....13
  - 2.3. Power Consumption.....13
  - 2.4. Receiver Sensitivity.....14
  - 2.5. MCU.....15
  - 2.6. Memory Organization.....15
    - 2.6.1. Internal SRAM and ROM.....15
    - 2.6.2. External SPI Flash.....15
  - 2.7. AHB and AHB Blocks.....16
- 3. Pins and Definitions.....17
  - 3.1. GPIO .....17
    - 3.1.1. General Purpose Input/Output Interface (GPIO) .....17



- 3.2. Secure Digital Input/Output Interface (SDIO) .....18
- 3.3. Serial Peripheral Interface (SPI/HSPI).....18
  - 3.3.1. General SPI (Master/Slave).....18
  - 3.3.2. SDIO / SPI (Slave).....19
  - 3.3.3. HSPI (Master/Slave) .....19
- 3.4. Inter-integrated Circuit Interface (I2C).....19
- 3.5. I2S .....20
- 3.6. Universal Asynchronous Receiver Transmitter (UART).....20
- 3.7. Pulse-Width Modulation (PWM) .....21
- 3.8. IR Remote Control .....22
- 3.9. ADC (Analog-to-digital Converter) .....22
- 3.10. LED Light and Button .....24
- 4. Firmware & Software Development Kit .....26
  - 4.1. Features.....26
- 5. Power Management .....27
- 6. Clock Management .....28
  - 6.1. High Frequency Clock.....28
  - 6.2. External Reference Requirements .....29
- 7. Radio.....29
  - 7.1. Channel Frequencies .....30
  - 7.2. 2.4 GHz Receiver .....30
  - 7.3. 2.4 GHz Transmitter .....30



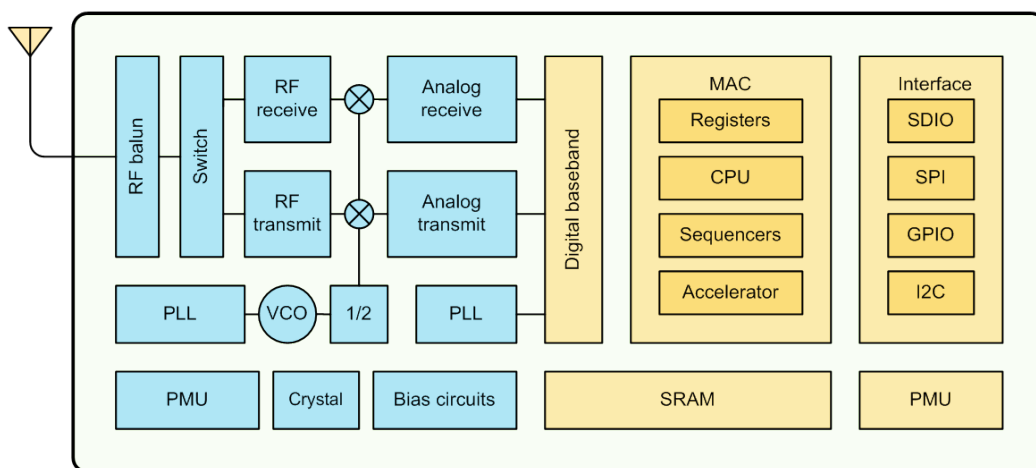
7.4. Clock Generator.....30

8. Appendix: QFN32 Package Size .....31

# 1. General Overview

## 1.1. Introduction

Espressif Systems' Smart Connectivity Platform (ESCP) is a set of high performance, high integration wireless SOCs, designed for space and power constrained mobile platform designers. It provides unsurpassed ability to embed WiFi capabilities within other systems, or to function as a standalone application, with the lowest cost, and minimal space requirement.



**Figure 1 ESP8266EX Block Diagram**

ESP8266EX offers a complete and self-contained WiFi networking solution; it can be used to host the application or to offload WiFi networking functions from another application processor.

When ESP8266EX hosts the application, it boots up directly from an external flash. It has integrated cache to improve the performance of the system in such applications.

Alternately, serving as a WiFi adapter, wireless internet access can be added to any micro controller-based design with simple connectivity (SPI/SDIO or I2C/UART interface).

ESP8266EX is among the most integrated WiFi chip in the industry; it integrates the antenna switches, RF balun, power amplifier, low noise receive amplifier, filters, power management modules, it requires minimal external circuitry, and the entire solution, including front-end module, is designed to occupy minimal PCB area.

ESP8266EX also integrates an enhanced version of Tensilica's L106 Diamond series 32-bit processor, with on-chip SRAM, besides the WiFi functionalities. ESP8266EX is often integrated with external sensors and other application specific devices through its GPIOs; sample codes for such applications are provided in the software development kit (SDK).



Espressif Systems' Smart Connectivity Platform (ESCP) demonstrates sophisticated system-level features include fast sleep/wake context switching for energy-efficient VoIP, adaptive radio biasing for low-power operation, advance signal processing, and spur cancellation and radio co-existence features for common cellular, Bluetooth, DDR, LVDS, LCD interference mitigation.

## 1.2. Features

- 802.11 b/g/n
- Integrated low power 32-bit MCU
- Integrated 10-bit ADC
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLL, regulators, and power management units
- Supports antenna diversity
- WiFi 2.4 GHz, support WPA/WPA2
- Support STA/AP/STA+AP operation modes
- Support Smart Link Function for both Android and iOS devices
- SDIO 2.0, (H) SPI, UART, I2C, I2S, IR Remote Control, PWM, GPIO
- STBC, 1x1 MIMO, 2x1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4s guard interval
- Deep sleep power <10uA, Power down leakage current < 5uA
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)
- +20 dBm output power in 802.11b mode
- Operating temperature range -40C ~ 125C
- FCC, CE, TELEC, WiFi Alliance, and SRRC certified

## 1.3. Parameters

Table 1 Parameters





Categories	Items	Values
WiFi Parameters	Certificates	FCC/CE/TELEC/SRRC
	WiFi Protocols	802.11 b/g/n
	Frequency Range	2.4G-2.5G (2400M-2483.5M)
	Tx Power	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx Sensitivity	802.11 b: -91 dbm (11 Mbps)
802.11 g: -75 dbm (54 Mbps)		
802.11 n: -72 dbm (MCS7)		
Types of Antenna	PCB Trace, External, IPEX Connector, Ceramic Chip	
Hardware Parameters	Peripheral Bus	UART/SDIO/SPI/I2C/I2S/IR Remote Control
		GPIO/PWM
	Operating Voltage	3.0~3.6V
	Operating Current	Average value: 80mA
	Operating Temperature Range	-40°~125°
	Ambient Temperature Range	Normal temperature
	Package Size	5x5mm
External Interface	N/A	
Software Parameters	WiFi mode	station/softAP/SoftAP+station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network)
	Software Development	Supports Cloud Server Development / SDK for custom firmware development
	Network Protocols	IPv4, TCP/UDP/HTTP/FTP



	User Configuration	AT Instruction Set, Cloud Server, Android/ iOS App
--	--------------------	---

### 1.4. Ultra Low Power Technology

ESP8266EX has been designed for mobile, wearable electronics and Internet of Things applications with the aim of achieving the lowest power consumption with a combination of several proprietary techniques. The power saving architecture operates mainly in 3 modes: active mode, sleep mode and deep sleep mode.

By using advance power management techniques and logic to power-down functions not required and to control switching between sleep and active modes, ESP8266EX consumes about than 60uA in deep sleep mode (with RTC clock still running) and less than 1.0mA (DTIM=3) or less than 0.5mA (DTIM=10) to stay connected to the access point.

When in sleep mode, only the calibrated real-time clock and watchdog remains active. The real-time clock can be programmed to wake up the ESP8266EX at any required interval.

The ESP8266EX can be programmed to wake up when a specified condition is detected. This minimal wake-up time feature of the ESP8266EX can be utilized by mobile device SOCs, allowing them to remain in the low-power standby mode until WiFi is needed.

In order to satisfy the power demand of mobile and wearable electronics, ESP8266EX can be programmed to reduce the output power of the PA to fit various application profiles, by trading off range for power consumption.

### 1.5. Major Applications

Major fields of ESP8266EX applications to Internet-of-Things include:

- Home Appliances
- Home Automation
- Smart Plug and lights
- Mesh Network
- Industrial Wireless Control
- Baby Monitors
- IP Cameras
- Sensor Networks
- Wearable Electronics

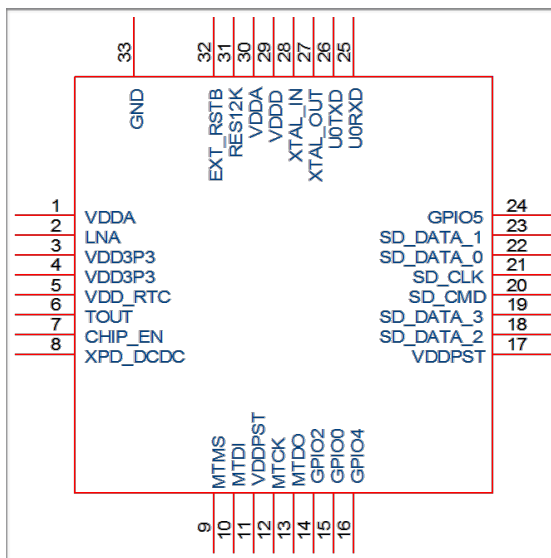


- WiFi Location-aware Devices
- Security ID Tags
- WiFi Position System Beacons

## 2. Hardware Overview

### 2.1. Pin Definitions

The pin assignments for 32-pin QFN package is illustrated in Fig.2.



**Figure 2 Pin Assignments**

Table 2 below presents an overview on the general pin attributes and the functions of each pin.

**Table 2 Pin Definitions**

Pin	Name	Type	Function
1	VDDA	P	Analog Power 3.0 ~3.6V
2	LNA	I/O	RF Antenna Interface. Chip Output Impedance=50Ω No matching required but we recommend that the n-type matching network is retained.
3	VDD3P3	P	Amplifier Power 3.0~3.6V
4	VDD3P3	P	Amplifier Power 3.0~3.6V
5	VDD_RTC	P	NC (1.1V)



6	TOUT	I	ADC Pin (note: an internal pin of the chip) can be used to check the power voltage of VDD3P3 (Pin 3 and Pin4) or the input voltage of TOUT (Pin 6). These two functions cannot be used simultaneously.
7	CHIP_EN	I	Chip Enable. High: On, chip works properly; Low: Off, small current
8	XPD_DCDC	I/O	Deep-Sleep Wakeup; GPIO16
9	MTMS	I/O	GPIO14; HSPI_CLK
10	MTDI	I/O	GPIO12; HSPI_MISO
11	VDDPST	P	Digital/IO Power Supply (1.8V~3.3V)
12	MTCK	I/O	GPIO13; HSPI_MOSI; UART0_CTS
13	MTDO	I/O	GPIO15; HSPI_CS; UART0_RTS
14	GPIO2	I/O	UART Tx during flash programming; GPIO2
15	GPIO0	I/O	GPIO0; SPI_CS2
16	GPIO4	I/O	GPIO4
17	VDDPST	P	Digital/IO Power Supply (1.8V~3.3V)
18	SDIO_DATA_2	I/O	Connect to SD_D2 (Series R: 200Ω); SPIHD; HSPIHD; GPIO9
19	SDIO_DATA_3	I/O	Connect to SD_D3 (Series R: 200Ω); SPIWP; HSPIWP; GPIO10
20	SDIO_CMD	I/O	Connect to SD_CMD (Series R: 200Ω); SPI_CS0; GPIO11
21	SDIO_CLK	I/O	Connect to SD_CLK (Series R: 200Ω); SPL_CLK; GPIO6
22	SDIO_DATA_0	I/O	Connect to SD_D0 (Series R: 200Ω); SPI_MSIO; GPIO7
23	SDIO_DATA_1	I/O	Connect to SD_D1 (Series R: 200Ω); SPI_MOSI; GPIO8
24	GPIO5	I/O	GPIO5
25	U0RXD	I/O	UART Rx during flash programming; GPIO3
26	U0TXD	I/O	UART Tx during flash programming; GPIO1; SPI_CS1
27	XTAL_OUT	I/O	Connect to crystal oscillator output, can be used to provide BT clock input
28	XTAL_IN	I/O	Connect to crystal oscillator input
29	VDDD	P	Analog Power 3.0V~3.6V
30	VDDA	P	Analog Power 3.0V~3.6V
31	RES12K	I	Serial connection with a 12 kΩ resistor and connect to the ground
32	EXT_RSTB	I	External reset signal (Low voltage level: Active)



**Note:** GPIO2, GPIO0, MTDO can be configurable as 3-bit SDIO mode.

## 2.2. Electrical Characteristics

Table 3 ESP8266EX Electrical Characteristics

Parameters		Conditions	Min	Typical	Max	Unit
Storage Temperature Range			-40	Normal	125	°C
Maximum Soldering Temperature		IPC/JEDEC J-STD-020			260	°C
Working Voltage Value			3.0	3.3	3.6	V
I/O	$V_{IL}/V_{IH}$		$-0.3/0.75V_{IO}$		$0.25V_{IO}/3.6$	V
	$V_{OL}/V_{OH}$		$N/0.8V_{IO}$		$0.1V_{IO}/N$	
	$I_{MAX}$				12	mA
Electrostatic Discharge (HBM)		TAMB=25°C			2	KV
Electrostatic Discharge (CDM)		TAMB=25°C			0.5	KV

## 2.3. Power Consumption

The following current consumption is based on 3.3V supply, and 25°C ambient, using internal regulators. Measurements are done at antenna port without SAW filter. All the transmitter's measurements are based on 90% duty cycle, continuous transmit mode.

Table 4 Description on Power Consumption

Parameters	Min	Typical	Max	Unit
Tx802.11b, CCK 11Mbps, P OUT=+17dBm		170		mA
Tx 802.11g, OFDM 54Mbps, P OUT =+15dBm		140		mA
Tx 802.11n, MCS7, P OUT =+13dBm		120		mA
Rx 802.11b, 1024 bytes packet length , -80dBm		50		mA
Rx 802.11g, 1024 bytes packet length, -70dBm		56		mA
Rx 802.11n, 1024 bytes packet length, -65dBm		56		mA
Modem-Sleep①		15		mA
Light-Sleep②		0.9		mA
Deep-Sleep③		10		uA
Power Off		0.5		uA



①: Modem-Sleep requires the CPU to be working, as in PWM or I2S applications. According to 802.11 standards (like U-APSD), it saves power to shut down the WiFi Modem circuit while maintaining a WiFi connection with no data transmission. E.g. in DTIM3, to maintain a sleep 300ms-wake 3ms cycle to receive AP's Beacon packages, the current is about 15mA

②: During Light-Sleep, the CPU may be suspended in applications like WiFi switch. Without data transmission, the WiFi Modem circuit can be turned off and CPU suspended to save power according to the 802.11 standard (U-APSD). E.g. in DTIM3, to maintain a sleep 300ms-wake 3ms cycle to receive AP's Beacon packages, the current is about 0.9mA.

③: Deep-Sleep does not require WiFi connection to be maintained. For application with long time lags between data transmission, e.g. a temperature sensor that checks the temperature every 100s, sleep 300s and waking up to connect to the AP (taking about 0.3~1s), the overall average current is less than 1mA.

## 2.4. Receiver Sensitivity

The following are measured under room temperature conditions with 3.3V and 1.1V power supplies.

Table 5 Receiver Sensitivity

Parameters	Min	Typical	Max	Unit
Input frequency	2412		2484	MHz
Input impedance		50		$\Omega$
Input reflection			-10	dB
Output power of PA for 72.2Mbps	15.5	16.5	17.5	dBm
Output power of PA for 11b mode	19.5	20.5	21.5	dBm
Sensitivity				
DSSS, 1Mbps		-98		dBm
CCK, 11Mbps		-91		dBm
6Mbps (1/2 BPSK)		-93		dBm
54Mbps (3/4 64-QAM)		-75		dBm
HT20, MCS7 (65Mbps, 72.2Mbps)		-72		dBm
<b>Adjacent Channel Rejection</b>				
OFDM, 6Mbps		37		dB
OFDM, 54Mbps		21		dB
HT20, MCS0		37		dB
HT20, MCS7		20		dB



## 2.5. MCU

ESP8266EX is embedded with Tensilica L106 32-bit micro controller (MCU), which features extra low power consumption and 16-bit RSIC. The CPU clock speed is 80MHz. It can also reach a maximum value of 160MHz. Real Time Operation System (RTOS) is enabled. Currently, only 20% of MIPS has been occupied by the WiFi stack, the rest can all be used for user application programming and development. The following interfaces can be used to connect to the MCU embedded in ESP8266EX:

- Programmable RAM/ROM interfaces (iBus), which can be connected with memory controller, and can also be used to visit external flash;
- Data RAM interface (dBus), which can connected with memory controller;
- AHB interface, can be used to visit the register.

## 2.6. Memory Organization

### 2.6.1. Internal SRAM and ROM

ESP8266EX WiFi SoC is embedded with memory controller, including SRAM and ROM. MCU can visit the memory units through iBus, dBus, and AHB interfaces. All memory units can be visited upon request, while a memory arbiter will decide the running sequence according to the time when these requests are received by the processor.

According to our current version of SDK provided, SRAM space that is available to users is assigned as below:

- **RAM size < 36kB**, that is to say, when ESP8266EX is working under the station mode and is connected to the router, programmable space accessible to user in heap and data section is around 36kB.)
- There is no programmable ROM in the SoC, therefore, user program must be stored in an external SPI flash.

### 2.6.2. External SPI Flash

An external SPI flash is used together with ESP8266EX to store user programs. Theoretically speaking, up to 16 Mbyte memory capacity can be supported.

**Suggested SPI Flash memory capacity:**

- OTA is disabled: the minimum flash memory that can be supported is 512 kByte;
- OTA is enabled: the minimum flash memory that can be supported is 1 Mbyte.

Several SPI modes can be supported, including Standard SPI, Dual SPI, DIO SPI, QIO SPI, and Quad SPI.





Therefore, please choose the correct SPI mode when you are downloading into the flash, otherwise firmwares/programs that you downloaded may not work in the right way.

## 2.7. AHB and APB Blocks

The AHB block performs the function of an arbiter, controls the AHB interfaces from the MAC, SDIO (host) and CPU. Depending on the address, the AHB data requests can go into one of the two slaves: APB block, or

flash controller (usually for standalone applications).

Data requests to the memory controller are usually high speed requests, and requests to the APB block are usually register access.

The APB block acts as a decoder. It is meant only for access to programmable registers within ESP8266's main blocks. Depending on the address, the APB request can go to the radio, SI/SPI, SDIO (host), GPIO, UART, real-time clock (RTC), MAC or digital baseband.

# HC-SR04 Ultrasonic Sensor

Elijah J. Morgan

Nov. 16 2014

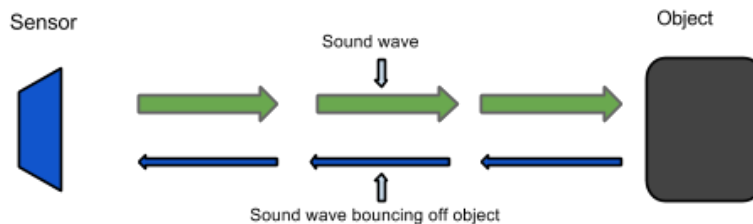
The purpose of this file is to explain how the HC-SR04 works. It will give a brief explanation of how ultrasonic sensors work in general. It will also explain how to wire the sensor up to a microcontroller and how to take/interpret readings. It will also discuss some sources of errors and bad readings.

1. How Ultrasonic Sensors Work
2. HC-SR04 Specifications
3. Timing chart, Pin explanations and Taking Distance Measurements
4. Wiring HC-SR04 with a microcontroller
5. Errors and Bad Readings



## 1. How Ultrasonic Sensors Work

Ultrasonic sensors use sound to determine the distance between the sensor and the closest object in its path. How do ultrasonic sensors do this? Ultrasonic sensors are essentially sound sensors, but they operate at a frequency above human hearing.



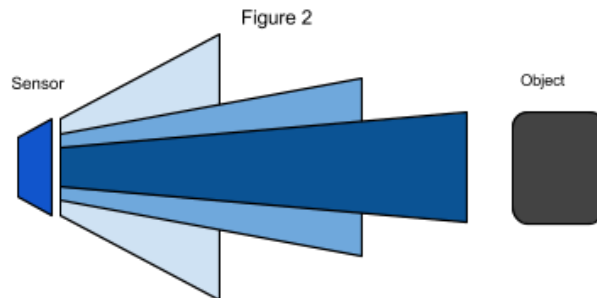
The sensor sends out a sound wave at a specific frequency. It then listens for that specific sound wave to bounce off of an object and come back (Figure 1). The sensor keeps track of the time between sending the sound wave and the sound wave returning. If you know how fast something is going and how long it is traveling you can find the distance traveled with equation 1.

**Equation 1.**  $d = v \times t$

The speed of sound can be calculated based on the a variety of atmospheric conditions, including temperature, humidity and pressure. Actually calculating the distance will be shown later on in this document.

It should be noted that ultrasonic sensors have a cone of detection, the angle of this cone varies with distance, Figure 2 show this relation. The ability of a sensor to

detect an object also depends on the objects orientation to the sensor. If an object doesn't present a flat surface to the sensor then it is possible the sound wave will bounce off the object in a way that it does not return to the sensor.



## 2. HC-SR04 Specifications

The sensor chosen for the Firefighting Drone Project was the HC-SR04. This section contains the specifications and why they are important to the sensor module. The sensor modules requirements are as follows.

- Cost
- Weight
- Community of hobbyists and support
- Accuracy of object detection
- Probability of working in a smoky environment
- Ease of use

The HC-SR04 Specifications are listed below. These specifications are from the Cytron Technologies HC-SR04 User's Manual (source 1).

- Power Supply: +5V DC
- Quiescent Current: <2mA
- Working current: 15mA
- Effectual Angle: <15°
- Ranging Distance: 2-400 cm
- Resolution: 0.3 cm
- Measuring Angle: 30°
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm
- Weight: approx. 10 g

The HC-SR04's best selling point is its price; it can be purchased at around \$2 per unit.

### 3. Timing Chart and Pin Explanations

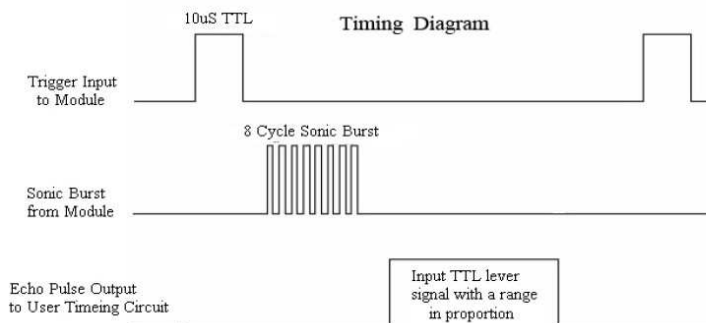
The HC-SR04 has four pins, VCC, GND, TRIG and ECHO; these pins all have different functions. The VCC and GND pins are the simplest -- they power the HC-SR04. These pins need to be attached to a +5 volt source and ground respectively. There is a single control pin: the TRIG pin. The TRIG pin is responsible for sending the ultrasonic burst. This pin should be set to HIGH for 10  $\mu$ s, at which point the HC-SR04 will send out an eight cycle sonic burst at 40 kHz. After a sonic burst has been sent the ECHO pin will go HIGH. The ECHO pin is the data pin -- it is used in taking distance measurements. After an ultrasonic burst is sent the pin will go HIGH, it will stay high until an ultrasonic burst is detected back, at which point it will go LOW.

#### Taking Distance Measurements

The HC-SR04 can be triggered to send out an ultrasonic burst by setting the TRIG pin to HIGH. Once the burst is sent the ECHO pin will automatically go HIGH. This pin will remain HIGH until the the burst hits the sensor again. You can calculate the distance to the object by keeping track of how long the ECHO pin stays HIGH. The time ECHO stays HIGH is the time the burst spent traveling. Using this measurement in equation 1 along with the speed of sound will yield the distance travelled. A summary of this is listed below, along with a visual representation in Figure 2.

1. Set TRIG to HIGH
2. Set a timer when ECHO goes to HIGH
3. Keep the timer running until ECHO goes to LOW
4. Save that time
5. Use equation 1 to determine the distance travelled

**Figure 3**  
**Source 2**



Source 2

To interpret the time reading into a distance you need to change equation 1. The clock on the device you are using will probably count in microseconds or smaller. To use equation 1 the speed of sound needs to be determined, which is 343 meters per second at standard temperature and pressure. To convert this into more useful form use equation 2 to change from meters per second to microseconds per centimeter. Then equation 3 can be used to easily compute the distance in centimeters.

$$\text{Equation 2. Distance} = \frac{\text{Speed}}{170.15 \text{ m}} \times \frac{\text{Meters}}{100 \text{ cm}} \times \frac{1e6 \mu\text{S}}{170.15 \text{ m}} \times \frac{58.772 \mu\text{S}}{\text{cm}}$$

$$\text{Equation 3. Distance} = \frac{\text{time}}{58} = \frac{\mu\text{s}}{\mu\text{s/cm}} = \text{cm}$$

#### 4. Wiring the HC-SR04 to a Microcontroller

This section only covers the hardware side. For information on how to integrate the software side, look at one of the links below or look into the specific microcontroller you are using.

The HC-SR04 has 4 pins: VCC, GND, TRIG and ECHO.

1. VCC is a 5v power supply. This should come from the microcontroller
2. GND is a ground pin. Attach to ground on the microcontroller.
3. TRIG should be attached to a GPIO pin that can be set to HIGH
4. ECHO is a little more difficult. The HC-SR04 outputs 5v, which could destroy many microcontroller GPIO pins (the maximum allowed voltage varies). In order to step down the voltage use a single resistor or a voltage divider circuit. Once again this depends on the specific microcontroller you are using, you will need to find out its GPIO maximum voltage and make sure you are below that.

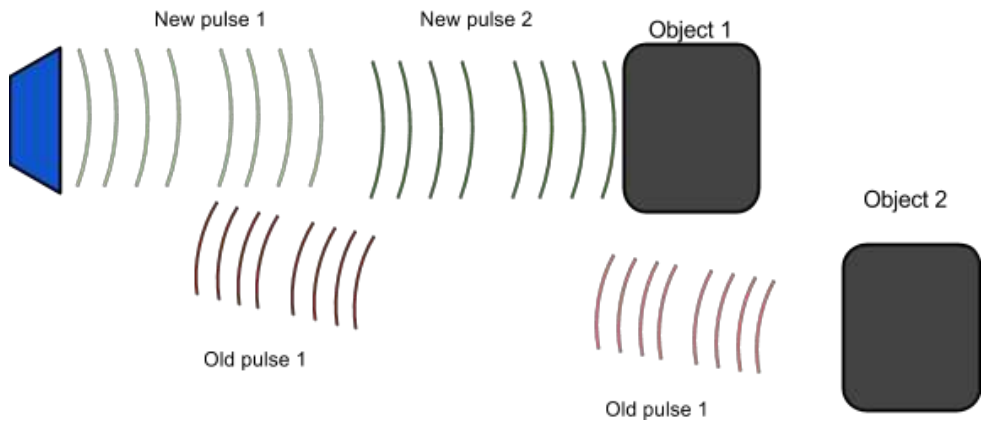


Figure 4

#### 5. Errors and Bad Readings

Ultrasonic sensors are great sensors -- they work well for many applications where other types of sensors fall short. Unfortunately, they do have weaknesses. These weaknesses can be mitigated and worked around, but first they must be understood. The

first weakness is that they use sound. There is a limit to how fast ultrasonic sensors can get distance measurements. The longer the distance, the slower they are at reporting the distance. The second weakness comes from the way sound bounces off of objects. In enclosed spaces it is possible, if not probable that there will be unintended echos. The echos can very easily cause false short readings. In Figure 2 a pulse was sent out. It bounced off of object 1 and returned to the sensor. The distance was recorded and then a new pulse was sent. There was another object farther away, so that when the new pulse reaches object 1, the first signal will reach the sensor. This will cause the sensor to think that there is an object closer than is actually true. The old pulse is smaller than the new pulse because it has grown weaker. The longer the pulse exists the weaker it grows until it is negligible. If multiple sensors are being used, the number of echos will increase along with the number of errors. There are two main ways to reduce the number of errors. The first is to provide shielding around the sensor. This prevents echos coming in from angle outside what the sensor should actually pick up. The second is to reduce the frequency at which pulses are sent out. This gives more time for the echos to dissipate.



## Works Cited

Source 1.

“HC-SR04 User's\_Manual.” *docs.google*. Cytron Technologies, May 2013 Web. 5 Dec. 2009.

<[https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL\\_pfa39RsB-x2qR4vP8saG73rE/edit](https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit)>

Source 2.

“Attiny2313 Ultrasonic distance (HR-SR04) example.” *CircuitDB*. n.a. 7 Sept. 2014 Web. 5 Dec. 2014. <<http://www.circuitdb.com/?p=1162>>

## Links

These are not formatted; you will need to copy and paste them into your web browser.

Want to learn about Ultrasonic Sensors in general?

<http://www.sensorsmag.com/sensors/acoustic-ultrasound/choosing-ultrasonic-sensor-proximity-or-distance-measurement-825>

All about the HC-SR04

- <http://www.circuitdb.com/?p=1162>
- <http://www.micropik.com/PDF/HCSR04.pdf>
- <http://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/>
- <http://www.ezdenki.com/ultrasonic.php>  
(^fantastic tutorial, explains a lot of stuff)
- <http://www.elecrow.com/hcsr04-ultrasonic-ranging-sensor-p-316.html>  
(^ this one has some cool charts)

## KY-032 Obstacle-detect module

### Contents

1 Picture .....	1
2 Technical data / Short description .....	1
3 Code example Arduino .....	3
4 Code example Raspberry Pi .....	4

### Picture



### Technical data / Short description

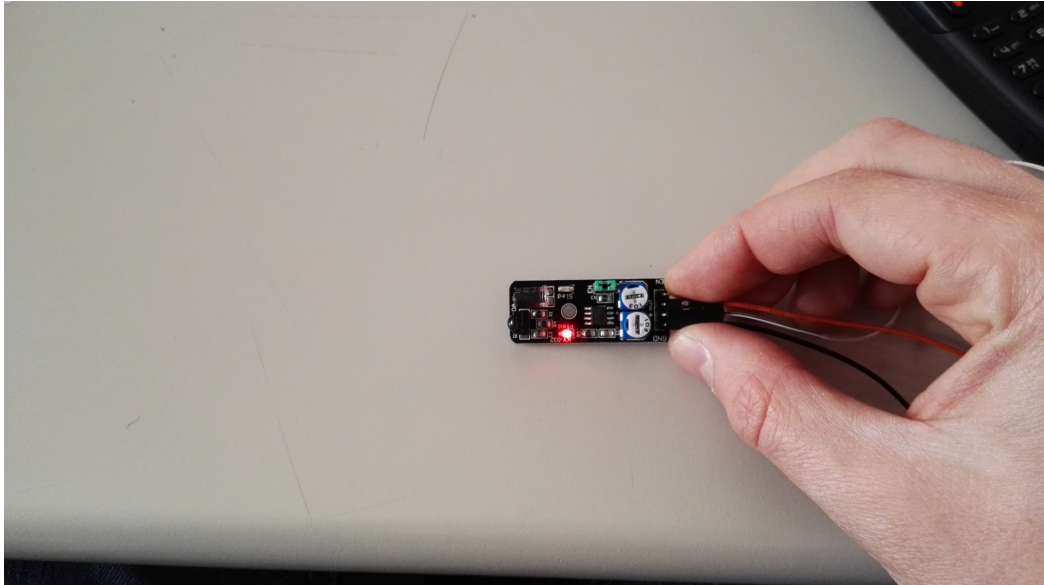
If the sended infrared light hits an obstacle, the light will be reflected and detected by the photodiode. The detection range can be configured by the 2 controllers.

The behaviour of this can be used for Controllers, for example to stop a robot automatically which drives straight to an obstacle.

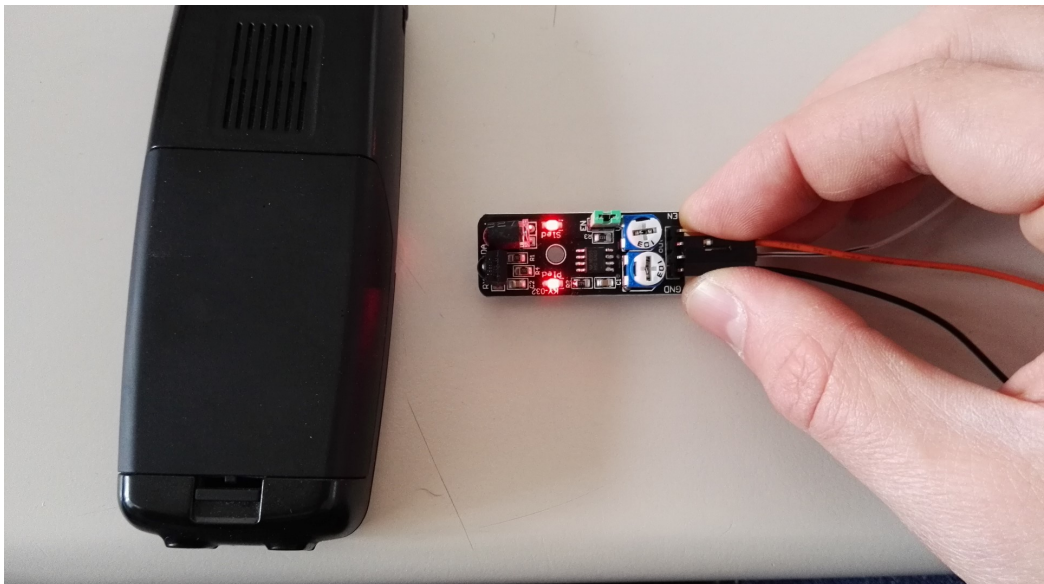
**Condition 1:** There is no obstacle in front of the detector [LED on the module: Off] [Sensor signal= Digital On]



KY-032 Obstacle-detect module

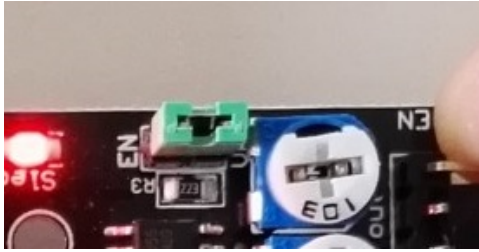


**Condition 2:** Detector detects an obstacle [LED on the module: ON] [Sensor Signal= Digital Off]



This sensor allows you to activate and deactivate the obstacle detection by manipulating the enable-pin. It is activated by default which means that the sensor is in detection mode by default. If you don't want that, you need to remove the jumper with the label "EN" and put a control signal on the enable pin.

## KY-032 Obstacle-detect module



## Code example Arduino

This program reads the status of the sensor pin and prints it to the serial terminal if an obstacle was detected.

```
int Sensor = 10; // Declaration of the sensor input pin
void setup ()
{
  Serial.begin(9600); // Initialization serial output
  pinMode (Sensor, INPUT) ; // Initialization sensor pin
}
// The program reads the current status of the sensor pin
// shows via serial console if the sensor detects a obstacle or not
void loop ()
{
  bool val = digitalRead (Sensor) ; // The current signal at the sensor will be read
  if (val == HIGH) // If a signal is detected, the LED will light up.
  {
    Serial.println("No obstacle");
  }
  else
  {
    Serial.println("Obstacle detected");
  }
  Serial.println("-----");
  delay(500); // Break of 500ms between each measurement
}
```

### Connections Arduino:

Sensor enable	=	[N.C. (jumper plugged in)]
Sensor signal	=	[Pin 10]
Sensor +V	=	[Pin 5V]
Sensor GND	=	[Pin GND]

### Example program download

[KY-032\\_Obstacle-detection](#)

## Code example Raspberry Pi

```
# Needed modules will be imported and configured
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# Declaration of the input pin which is connected with the sensor.
GPIO_PIN = 24
GPIO.setup(GPIO_PIN, GPIO.IN, pull_up_down = GPIO.PUD_UP)

# Break between the results will be defined here (in seconds)
delayTime = 0.5

print "Sensor-Test [press ctrl+c to end]"

# main program loop
try:
    while True:
        if GPIO.input(GPIO_PIN) == True:
            print "No obstacle"
        else:
            print "Obstacle detected"
            print "-----"

            # Reset + Delay
            time.sleep(delayTime)

# Scavenging work after the end of the program
except KeyboardInterrupt:
    GPIO.cleanup()
```

### Connections Raspberry Pi:

Enable	= -	[N.C. (jumper plugged in)]
Signal	= GPIO24	[Pin 18]
+V	= 3,3V	[Pin 1]
GND	= GND	[Pin 6]

### Example program download

[KY-032\\_Obstacle-detection\\_RPi](#)

To start, enter the command:

```
sudo python KY-032_Obstacle-detection_RPi.py
```