Hands on session 04: Touchless Dispenser

Session Introduction: In this session we will make a touchless dispenser using ultrasonic sensor, Relay & Micro Submersible Water Pump.

Session Objectives

- **To understand working of ultrasonic sensor**
- **To understand working of relay**
- **To construct a touchless dispenser**

Ultrasonic Sensor

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. It comes complete with ultrasonic transmitter and receiver modules.

Features

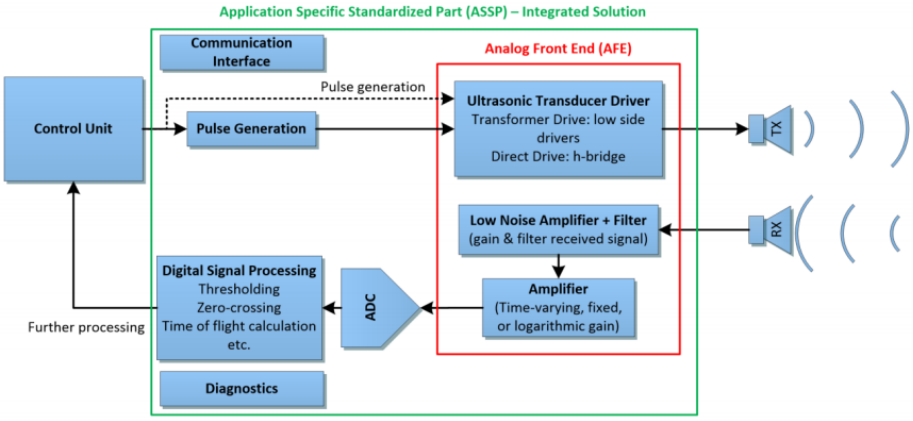
Here's a list of some of the HC-SR04 ultrasonic sensor features and specs:

- Power Supply: +5V DC
- Quiescent Current : <2mA
- Working Current: 15mA
- Effectual Angle: <15°
- Ranging Distance: 2cm – 400 cm/1" – 13ft
- Resolution: 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm

Quiescent Current can be defined as the amount of current used by an IC when in a Quiescent state. The Quiescent state being any period of time when the IC is in either a no load or non-switching condition, however is still enabled.

Effectual Angle is the maximum angle at which an object can be placed from the centre of the sensor in order to deliver accurate reading.

Resolution is the minimum change in distance that can be measured by the sensor when the target moves relative to it. the lower the frequency of the sensor, the longer the range of detection, while a higher frequency sensor will have greater measurement resolution and less susceptibility to the background noise.

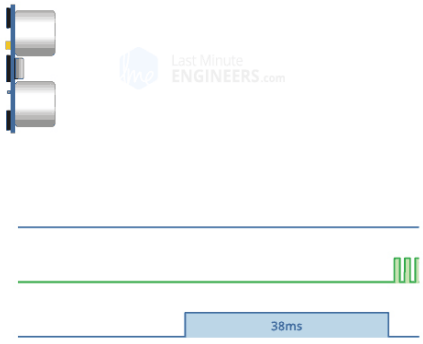


Working Principle

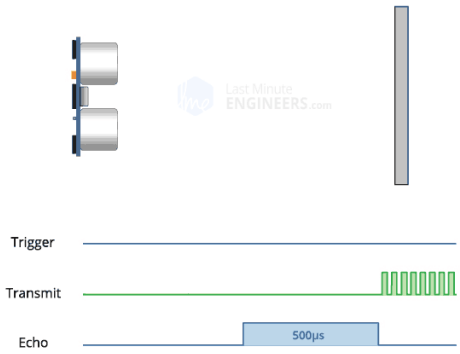
It all starts, when a pulse of at least 10 µS (10 microseconds) in duration is applied to the Trigger pin. In response to that the sensor transmits a sonic burst of eight pulses at 40 KHz. This 8-pulse pattern makes the "ultrasonic signature" from the device unique, allowing the receiver to differentiate the transmitted pattern from the ambient ultrasonic noise.

The eight ultrasonic pulses travel through the air away from the transmitter. Meanwhile the Echo pin goes HIGH to start forming the beginning of the echo-back signal.

In case, If those pulses are not reflected back then the Echo signal will timeout after 38 mS (38 milliseconds) and return low. Thus a 38 mS pulse indicates no obstruction within the range of the sensor.



If those pulses are reflected back the Echo pin goes low as soon as the signal is received. This produces a pulse whose width varies between 150 µS to 25 mS, depending upon the time it took for the signal to be received.



The width of the received pulse is then used to calculate the distance to the reflected object. This can be worked out using simple distance-speed-time equation, we learned in High school.

Let's take an example to make it clearer. Suppose we have an object in front of the sensor at an unknown distance and we received a pulse of width 500 µS on the Echo pin. Now let's calculate how far the object from the sensor is. We will use the below equation.

Distance = Speed x Time

Here, we have the value of Time i.e., 500 µs and we know the speed. What speed do we have? The speed of sound, of course! Its 340 m/s. We have to convert the speed of sound into cm/µs in order to calculate the distance. A quick Google search for "speed of sound in centimetres per microsecond" will say that it is 0.034 cm/µs. You could do the math, but searching it is easier. Anyway, with that information, we can calculate the distance!

Distance = 0.034 cm/µs x 500 µs

But this is not done! Remember that the pulse indicates the time it took for the signal to be sent out and reflected back so to get the distance so, you'll need to divide your result in half.

Distance = (0.034 cm/µs x 500 µs) / 2

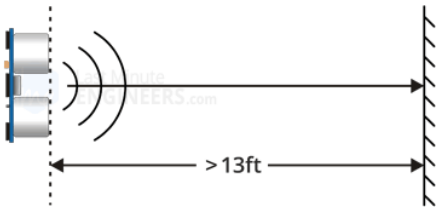
Distance = 8.5 cm

So, now we know that the object is 8.5 centimetres away from the sensor.

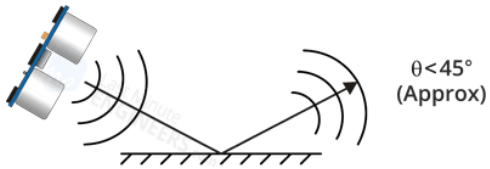
What are the limitations?

In terms of accuracy and overall usefulness, HC-SR04 ultrasonic distance sensor is really great, especially compared to other low-cost distance detection sensors. That doesn't mean that the HC-SR04 sensor is capable of measuring "everything". Following diagrams shows a few situations that the HC-SR04 is not designed to measure:

a) The distance between the sensor and the object/obstacle is more than 13 feet.

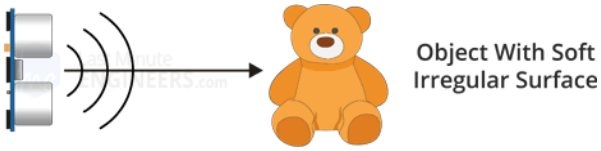


b) The object has its reflective surface at a shallow angle so that sound will not be reflected back towards the sensor.



c) The object is too small to reflect enough sound back to the sensor. In addition, if your HC-SR04 sensor is mounted low on your device, you may detect sound reflecting off of the floor.

d) While experimenting with the sensor, we discovered that some objects with soft, irregular surfaces (such as stuffed animals) absorb rather than reflect sound and therefore can be difficult for the HC-SR04 sensor to detect.



Effect of Temperature on Distance Measurement

Though the HC-SR04 is reasonably accurate for most of our projects such as intruder detection or proximity alarms; But there are times you might want to design a device that is to be used outdoors or in an unusually hot or cold environment. If this is the case, you might want to take into account the fact that the speed of sound in air varies with temperature, air pressure and humidity.

Since the speed of sound factors into our HC-SR04 distance calculation this could affect our readings. If the temperature (°C) and Humidity is already known, consider the below formula:

$$\text{Speed of sound m/s} = 331.4 + (0.606 * \text{Temp}) + (0.0124 * \text{Humidity})$$

PulseIn()

Description

Reads a pulse (either HIGH or LOW) on a pin. For example, if value is HIGH, pulseIn() waits for the pin to go from LOW to HIGH, starts timing, then waits for the pin to go LOW and stops timing. Returns the length of the pulse in microseconds or gives up and returns 0 if no complete pulse was received within the timeout.

Syntax

pulseIn(pin, value)
pulseIn(pin, value, timeout)

Parameters

pin: the number of the Arduino pin on which you want to read the pulse. Allowed data types: int.
value: type of pulse to read: either HIGH or LOW. Allowed data types: int.
timeout (optional): the number of microseconds to wait for the pulse to start; default is one second. Allowed data types: unsigned long.

https://www.arduino.cc/reference/en/language/functions/advanced-io/pulsein/

Ultrasonic connections:

- **VCC -> 5V**
- **Trig -> GPIO 13**
- **Echo -> GPIO 12**
- **GND -> GND**

Relay

A Relay is an electromechanical device that can be used to make or break an electrical connection. It consists of a flexible moving mechanical part which can be controlled electronically through an electromagnet, basically, a relay is just like a mechanical switch but you can control it with an electronic signal instead of manually turning it on or off.

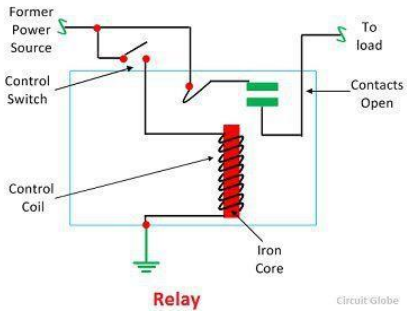
Relay

Working Principle of Relay

It works on the principle of an electromagnetic attraction. When the circuit of the relay senses the fault current, it energises the electromagnetic field which produces the temporary magnetic field.

This magnetic field moves the relay armature for opening or closing the connections. The small power relay has only one contacts, and the high-power relay has two contacts for opening the switch.

The inner section of the relay is shown in the figure below. It has an iron core which is wound by a control coil. The power supply is given to the coil through the contacts of the load and the control switch. The current flows through the coil produces the magnetic field around it.



Due to this magnetic field, the upper arm of the magnet attracts the lower arm. Hence close the circuit, which makes the current flow through the load. If the contact is already closed, then it moves oppositely and hence open the contacts.

Relay Configurations:

COM: connect the current you want to control (mains voltage).

NC (Normally Closed): the normally closed configuration is used when you want the relay to be closed by default. The NC are COM pins are connected, meaning the current is flowing unless you send a signal from the ESP32 to the relay module to open the circuit and stop the current flow.

NO (Normally Open): the normally open configuration works the other way around: there is no connection between the NO and COM pins, so the circuit is broken unless you send a signal from the ESP32 to close the circuit.

Normally Closed configuration (NC):

- HIGH signal – current is flowing
- LOW signal – current is not flowing

Normally Open configuration (NO):

- HIGH signal – current is not flowing

- LOW signal – current in flowing

You should use a normally closed configuration when the current should be flowing most of the times, and you only want to stop it occasionally.

Use a normally open configuration when you want the current to flow occasionally (for example, turn on a lamp occasionally).

Relay Connections:

- **Relay Out -> GPIO 25**

Touchless Dispenser

The touchless dispenser is made by combining functionalities of Ultrasonic sensor and relay module to turn on & off a submersible pump which can be immersed in soap solution to make it work.

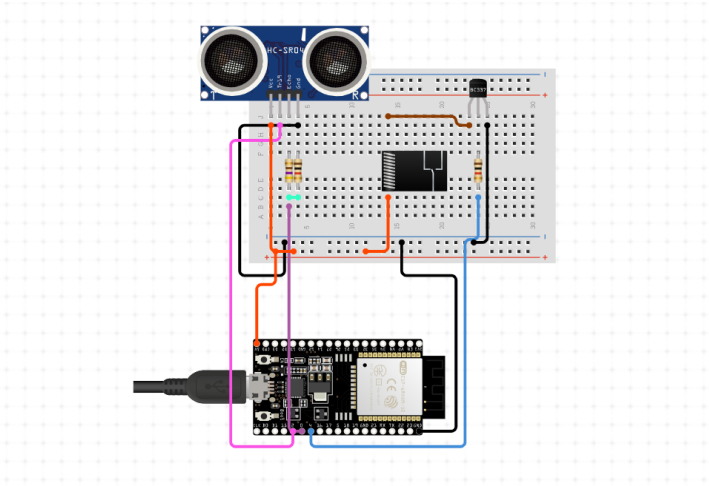
Connect all the components in the following order and create a code when a hand is detected in the range it turns on relay which turns on the pump for some milliseconds.

Ultrasonic connections:

- **VCC -> 5V**
- **Trig -> GPIO 13**
- **Echo -> GPIO 12**
- **GND -> GND**

Relay Connections:

- **Relay Out -> GPIO 25**



HCSR-04 Library

https://github.com/Martinsos/arduino-lib-hc-sr04

Ultrasonic sensor

https://www.ti.com/lit/an/slaa907c/slaa907c.pdf?ts=1615370081325