

ESP32 with HC-SR04 Ultrasonic Sensor&Buzzer with Arduino IDE(echipa 1011)

Am realizat un montaj Arduino compus dintr-o plăcuță ESP32, un senzor de mișcare și un buzzer, urmând pașii de mai jos:

1.Am achiziționat componentele necesare: Am cumpărat o plăcuță ESP32, un senzor de mișcare (PIR - Passive Infrared Sensor), precum și un buzzer și cabluri de conexiune.

2.Am studiat documentația: Am citit documentația și specificațiile tehnice pentru plăcuța ESP32, senzorul de mișcare și buzzer. Astfel, am înțeles pinii și funcționalitățile disponibile pentru fiecare componentă.

3.Am identificat pinii necesari: Am determinat pinii pe care trebuia să îi utilizez pentru conectarea senzorului de mișcare și a buzzerului la plăcuța ESP32, consultând schemele și documentația disponibilă.

4.Am conectat senzorul de mișcare: Am conectat pinii de ieșire ai senzorului de mișcare la pinii corespunzători de intrare ai plăcuței ESP32. Senzorul de mișcare avea trei pini: alimentare (VCC), masă (GND) și ieșire (OUT).

5.Am conectat buzzerul: Am conectat un pin digital al plăcuței ESP32 la pinul de control al buzzerului. Am asigurat că pinul era configurat ca ieșire în codul Arduino.

6.Am realizat inițializarea și configurarea: În funcția setup(), am configurat pinii de intrare/ieșire și am setat parametrii necesari pentru comunicarea cu senzorul de mișcare și controlul buzzerului.

7.Am implementat funcționalitatea: În funcția loop(), am citit valorile de la senzorul de mișcare și am luat decizii pe baza acestora. Dacă un obiect era detectat de senzorul de mișcare, am activat buzzerul pentru a emite un sunet.

8. Am testat și ajustat: Am rulat programul pe plăcuța ESP32 și am verificat funcționarea corectă a senzorului de mișcare și a buzzerului. Am făcut ajustări dacă erau necesare pentru a obține rezultatele dorite.

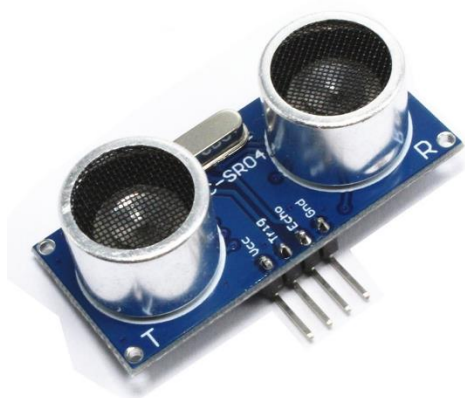
Componente utilizate:



ESP32



BUZZER



HC-SR04

Plăcuța ESP32 este o placă de dezvoltare avansată și puternică, care oferă o gamă largă de funcționalități și capacități tehnice remarcabile. Iată o descriere succintă a plăcuței ESP32 și a datelor tehnice relevante:

ESP32 este o placă de dezvoltare bazată pe cipul ESP-WROOM-32, care face parte din familia de cipuri ESP32 produse de Espressif Systems. Această placă oferă o soluție completă de dezvoltare și prototipare pentru proiecte de Internet of Things (IoT) și aplicații de conectivitate wireless.

Plăcuța ESP32 este dotată cu un procesor dual-core Tensilica Xtensa LX6, care rulează la frecvența de până la 240 MHz. Aceasta oferă o putere de calcul considerabilă și capacitatea de a executa sarcini complexe în timp real. În plus, plăcuța dispune de 520 KB de memorie SRAM și 4 MB de memorie flash, oferind spațiu suficient pentru stocarea programelor și a datelor.

Cipul ESP32 vine cu 48 de pini cu funcții multiple. Nu toți pinii sunt expuși în toate plăcile de dezvoltare ESP32, iar unii pini nu pot fi utilizați.

Figura de mai jos ilustrează configurarea pinilor pentru ESP-WROOM-32:

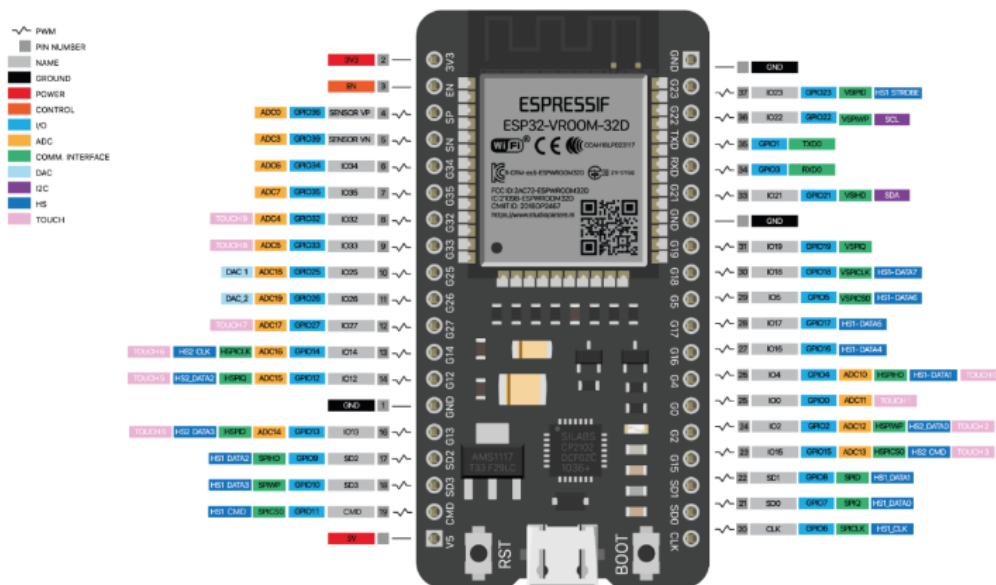
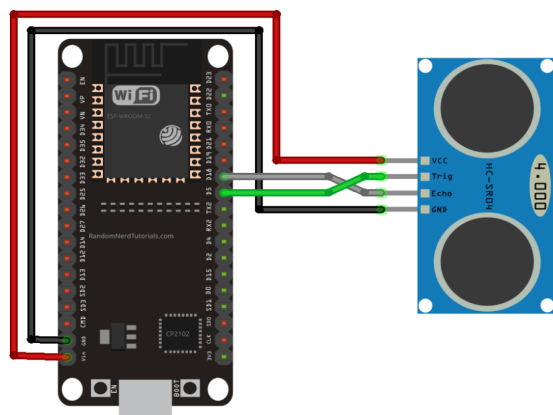


Fig. 1 - Diagrama pinilor plăcii de dezvoltare ESP32 – varianta cu 38 de pini [1]

ESP32 CONECTARE SENZOR:



Ultrasonic Sensor	ESP32
VCC	5V
Trig	G32
Echo	G33
GND	GND

Am realizat montarea corectă a senzorului utilizând următorii pași:

1. Am studiat documentația: Am citit cu atenție documentația specifică pentru senzorul pe care am dorit să-l utilizez. Astfel, am înțeles funcționalitățile acestuia, specificațiile tehnice și modul corect de conectare.

2. Am identificat pinii necesari: Consultând documentația senzorului și specificațiile tehnice, am determinat pinii de conexiune necesari pentru montaj. Am luat în considerare alimentarea, masă și pinii de semnal, în funcție de funcționalitatea dorită.

3. Am pregătit placa de dezvoltare: Am conectat placa de dezvoltare Arduino sau ESP32 la calculator și am deschis mediul de dezvoltare Arduino IDE.

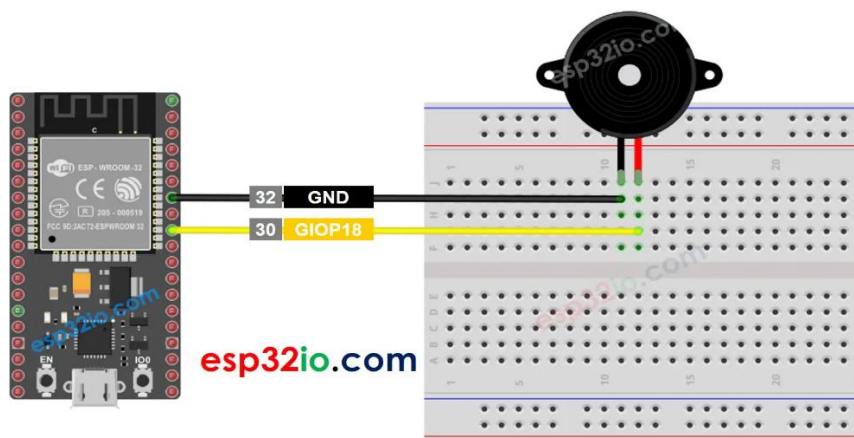
4. Am conectat senzorul: Utilizând cabluri de conexiune, am conectat pinii senzorului la pinii corespunzători ai plăcii de dezvoltare. Am asigurat că conexiunile erau corecte și strânse pentru a evita orice problemă de conexiune.

5. Am verificat alimentarea și masă: Am conectat pinul de alimentare al senzorului la pinul de alimentare adecvat de pe placa de dezvoltare (de obicei, pinul VCC). De asemenea, am conectat pinul de masă al senzorului la pinul de masă (GND) al plăcii de dezvoltare. Aceste conexiuni au furnizat energia necesară pentru funcționarea senzorului.

6. Am conectat pinii de semnal: Am conectat pinii de semnal ai senzorului la pinii corespunzători ai plăcii de dezvoltare. Aceste conexiuni au permis transferul datelor și semnalelor între senzor și placa de dezvoltare.

7. Am verificat și testat conexiunile: După finalizarea conexiunilor, am verificat fiecare conexiune pentru a ne asigura că nu existau erori sau probleme de conectare. Am utilizat un multimetru sau am efectuat teste de funcționalitate pentru a confirma că senzorul era conectat corect și funcționa în mod corespunzător.

ESP32 CONECTARE BUZZER:



Buzzer	Esp32
POZ	G18
GND	GND

Am realizat montarea corectă a unui buzzer utilizând următorii pași:

1.Am studiat documentația: Am citit cu atenție documentația specifică pentru buzzerul pe care am dorit să-l utilizez. Astfel, am înțeles funcționalitățile acestuia, specificațiile tehnice și modul corect de conectare.

2.Am pregătit placa de dezvoltare: Am conectat placa de dezvoltare Arduino sau ESP32 la calculator și am deschis mediul de dezvoltare Arduino IDE.

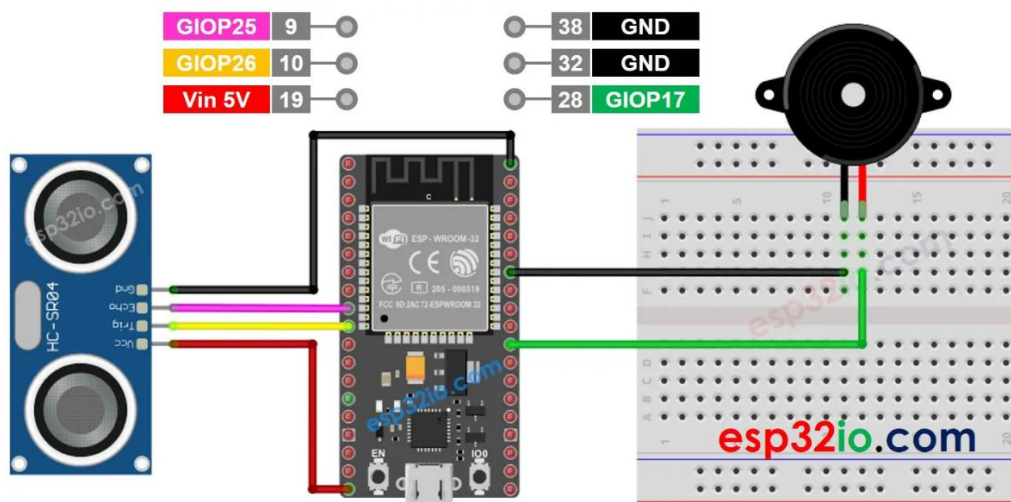
3.Am identificat pinii necesari: Consultând documentația buzzerului și specificațiile tehnice, am determinat pinul de control al buzzerului pe care trebuia să-l utilizăm pentru a controla funcționalitatea acestuia.

4.Am conectat buzzerul: Utilizând cabluri de conexiune, am conectat pinul de control al buzzerului la pinul corespunzător al plăcii de dezvoltare. Am asigurat că conexiunea era corectă și strânsă pentru a evita orice problemă de conexiune.

5. Am verificat alimentarea și masă: Dacă buzzerul necesită o sursă de alimentare separată, am conectat pinul de alimentare al buzzerului la o sursă de alimentare adecvată, cum ar fi o baterie sau o sursă de curent continuu. De asemenea, am conectat pinul de masă al buzzerului la pinul de masă (GND) al plăcii de dezvoltare. Aceste conexiuni au furnizat energia necesară pentru funcționarea corectă a buzzerului.

6. Am verificat și testat conexiunile: După finalizarea conexiunilor, am verificat fiecare conexiune pentru a ne asigura că nu existau erori sau probleme de conectare. Am utilizat un multimetru sau am efectuat teste de funcționalitate pentru a confirma că buzzerul era conectat corect și funcționa în mod corespunzător.

REZULTAT FINAL:



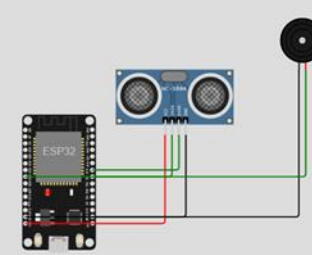
WOKWI

sketch.ino • diagram.json • Library Manager

```
1 const int trigPin = 5;
2 const int echoPin = 18;
3 const int buzzerPin = 25; // Define the pin for the buzzer
4
5 // Define sound velocity in cm/uS
6 #define SOUND_VELOCITY 0.034
7 #define CM_TO_INCH 0.393701
8
9 long duration;
10 float distanceCm;
11 float distanceInch;
12
13 void setup() {
14   Serial.begin(115200); // Starts the serial communication
15   pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
16   pinMode(echoPin, INPUT); // Sets the echoPin as an Input
17   pinMode(buzzerPin, OUTPUT); // Sets the buzzerPin as an Output
18 }
19
20 void loop() {
21   // Clears the trigPin
22   digitalWrite(trigPin, LOW);
23   delayMicroseconds(2);
24   // Sets the trigPin on HIGH state for 10 microseconds
25   digitalWrite(trigPin, HIGH);
26   delayMicroseconds(10);
27   digitalWrite(trigPin, LOW);
28
29   // Reads the echoPin, returns the sound wave travel time in microseconds
30   duration = pulseIn(echoPin, HIGH);
31
32   // Calculate the distance
33   distanceCm = duration * SOUND_VELOCITY / 2;
34
35   // Convert to inches
36   distanceInch = distanceCm * CM_TO_INCH;
37
38   // Prints the distance on the Serial Monitor
39   Serial.print("Distance (cm): ");
40   Serial.println(distanceCm);
41   Serial.print("Distance (inch): ");
42   Serial.println(distanceInch);
43
44   // Check if a new distance is discovered
45   if (distanceCm < 1) {
```

Simulation

Restart the simulation

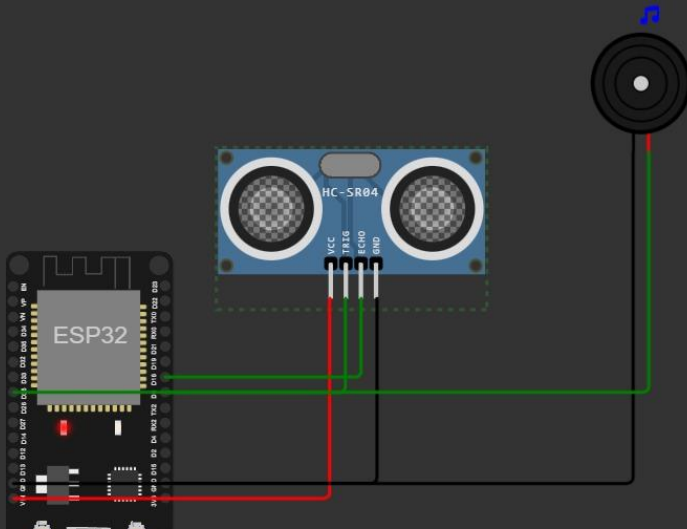


Distance (inch): 157.46
Distance (cm): 399.94
Distance (inch): 157.46
Distance (cm): 399.94
Distance (inch): 157.46
Distance (cm): 399.94
Distance (inch): 157.46
Distance (cm): 399.94

00:04.333 99%

Editing Ultrasonic Distance Sensor

Distance: 10cm



Distance (inch): 3.92
Distance (cm): 9.96
Distance (inch): 3.92
Distance (cm): 9.96
Distance (inch): 3.92
Distance (cm): 9.96
Distance (inch): 3.92

00:31.003 100%

<https://wokwi.com/projects/365353261479464961>

CODUL FOLOSIT:

```
const int trigPin = 32;

const int echoPin = 33;

const int buzzerPin = 18; // Define the pin for the buzzer


// Define sound velocity in cm/uS
#define SOUND_VELOCITY 0.034
#define CM_TO_INCH 0.393701


long duration;

float distanceCm;

float distanceInch;


void setup() {
  Serial.begin(115200); // Starts the serial communication
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  pinMode(buzzerPin, OUTPUT); // Sets the buzzerPin as an Output
}


void loop() {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);

  delayMicroseconds(2);

  // Sets the trigPin on HIGH state for 10 microseconds
```

```
digitalWrite(trigPin, HIGH);  
delayMicroseconds(10);  
digitalWrite(trigPin, LOW);  
  
// Reads the echoPin, returns the sound wave travel time in microseconds  
duration = pulseIn(echoPin, HIGH);  
  
// Calculate the distance  
distanceCm = duration * SOUND_VELOCITY / 2;  
  
// Convert to inches  
distanceInch = distanceCm * CM_TO_INCH;  
  
// Prints the distance on the Serial Monitor  
Serial.print("Distance (cm): ");  
Serial.println(distanceCm);  
Serial.print("Distance (inch): ");  
Serial.println(distanceInch);  
  
// Check if a new distance is discovered  
if (distanceCm < 50) {  
    // Sound the buzzer  
    digitalWrite(buzzerPin, HIGH);  
    delay(100); // Buzz duration  
    digitalWrite(buzzerPin, LOW);  
}  
  
delay(1000);  
}
```