2023

1. DIY Car backup sensor

Project number: **2021-1-FR01-KA220-SCH-000031617**

Co-funded by the European Union

SCRAPY Partnership

31/05/2023

Table of Contents

Experiment 1: DIY Car backup sensor

Short Description

Create a car reversing system with Raspberry Pi Pico and ultrasonic sensor.

Extended Description

Do you find reversing your car a bit tricky? Do you worry about hitting something or someone while backing up? Well, worry no more! With the help of the Raspberry Pi Pico board and an HC-SR04 ultrasonic sensor, you can build your very own reversing radar system that will make parking your car a breeze.

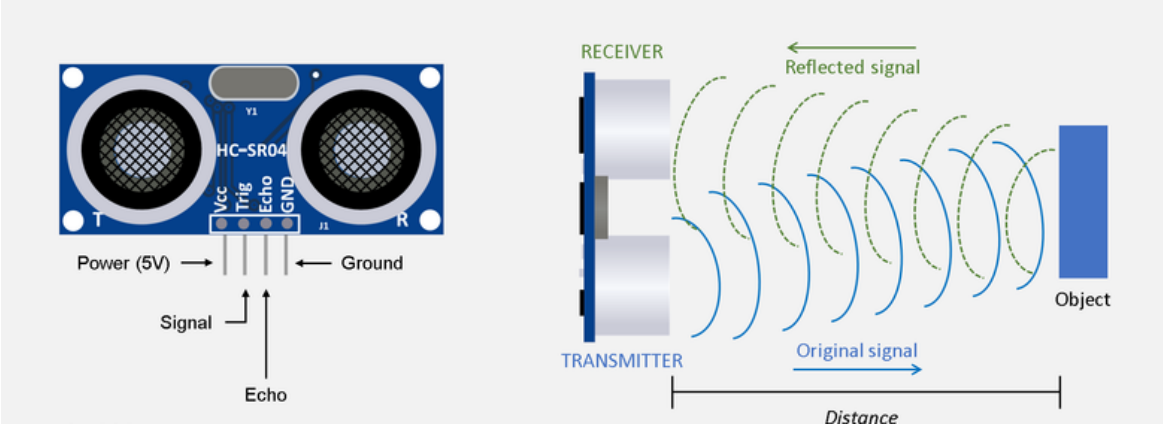
- This DIY project is perfect for anyone interested in electronics and engineering, and it is suitable for all skill levels. This reversing radar is sure to challenge and delight you.
- In addition to the Raspberry Pi Pico and HC-SR04 sensor, this project also requires a buzzer, three LEDs (green, yellow, and red), three 220-ohm resistors, a rapid prototyping board, and connection wires. With these components, you'll be able to create a reliable and accurate reversing radar that will help you park your car with ease.
- By following the step-by-step instructions, you'll learn how to wire the components together, program the Raspberry Pi Pico board, and test the system to ensure it works correctly. You'll also learn about the physics of ultrasonic sensors and how they can be used to measure distance.

The operational principles of the HC-SR04 ultrasonic sensor?

The HC-SR04 ultrasonic sensor works by emitting high-frequency sound waves that are inaudible to human ears. These sound waves travel through the air and bounce off of objects in their path. When the sound waves hit an object, they are reflected back to the sensor, which measures the time it takes for the waves to bounce back. By knowing the speed of sound and the time it takes for the waves to travel to the object and back, the sensor can calculate the distance to the object.

Here are the main steps in the operational principles of the HC-SR04 ultrasonic sensor:

- The sensor sends a high-frequency sound wave (typically around 40 kHz) from its transmitter.
- The sound wave travels through the air and bounces off of an object.
- The reflected sound wave is detected by the sensor's receiver.
- The sensor measures the time it takes for the sound wave to travel to the object and back.
- The sensor calculates the distance to the object based on the time it took for the sound wave to travel to the object and back.



Overall, the HC-SR04 ultrasonic sensor is a reliable and accurate way to measure distances, and it is commonly used in applications such as robotics, automation, and distance sensing.

Objectives:

Through this activity, the user will experiment on building a reversing radar system using the Raspberry Pi Pico board and an HC-SR04 ultrasonic sensor. The user will acquire knowledge on:

- The physics of ultrasonic waves and how they can be used to measure distance.
- The basics of programming in Python and how to write code to control the Raspberry Pi Pico board.
- The principles of circuit design and how to wire components together on a rapid prototyping board to create a functional reversing radar system.

By completing this project, the user will gain a deeper understanding of electronics, engineering, and programming. They will also have a practical and useful device that they can use to make parking their car safer and more convenient.

Materials to be used:

- 1 x Raspberry Pi Pico
- 1 x Pico breadboard kit
- 1 x Full size breadboard
- 1 x HC-SR04 ultrasonic sensor
- 1 x Buzzer
- 3 x LEDs (green, yellow, and red)
- 3 x 220-ohm resistors
- Jumper wires

Steps to be followed:

The main steps to realize the reversing radar experiment with the Raspberry Pi Pico board and HC-SR04 ultrasonic sensor:

1. Connect the HC-SR04 ultrasonic sensor to the Raspberry Pi Pico board using connection wires.
2. Connect the buzzer and LEDs to the Raspberry Pi Pico board using connection wires and the 220-ohm resistors to limit the current flow.
3. Write a Python program to control the Raspberry Pi Pico board and use the HC-SR04 sensor to measure distances.
4. Program the Raspberry Pi Pico board to turn on the green LED when there is no obstacle, the yellow LED when the obstacle is within a certain range, and the red LED when the obstacle is too close.
5. Program the Raspberry Pi Pico board to activate the buzzer when the obstacle is too close.
6. Test the reversing radar system by placing obstacles at various distances and angles behind the radar and ensure that the LEDs and buzzer provide the appropriate feedback.

For Step 4: To study the operation of the system, it is necessary to program its behavior. The distance values can be adjusted according to specific requirements. The programming should follow the following rules:

- The green LED should be continuously on if the distance between the sensor and any obstacle is greater than 20cm (1 meter).
- The orange LED should be on if the distance between the sensor and any obstacle is between 20cm and 5cm.
- The red LED should be on if the distance between the sensor and any obstacle is less than 5cm.

By programming the system to follow these rules, it will provide clear visual feedback of the distance between the sensor and any obstacle.

These rules can be adjusted as needed to suit specific requirements or use cases.

Wiring diagram

Here is a wiring schematic for the reversing radar project with Raspberry Pi Pico board, HC-SR04 ultrasonic sensor, buzzer, and three LEDs:

Raspberry Pi Pico Board:

- GP15: Trigger pin of the HC-SR04 sensor
- GP14: Echo pin of the HC-SR04 sensor
- GP10: Positive pin of the green LED
- GP11: Positive pin of the orange LED
- GP12: Positive pin of the red LED

- GP2: Positive pin of the buzzer
- GND: Ground pin of the board

HC-SR04 Sensor:

- VCC: Connect to 5V power source
- GND: Connect to GND of Raspberry Pi Pico board
- Trig: Connect to GP15 of Raspberry Pi Pico board
- Echo: Connect to GP14 of Raspberry Pi Pico board

Green LED:

- Positive leg: Connect to GP10 of Raspberry Pi Pico board via a 220-ohm resistor
- Negative leg: Connect to GND of Raspberry Pi Pico board

Orange LED:

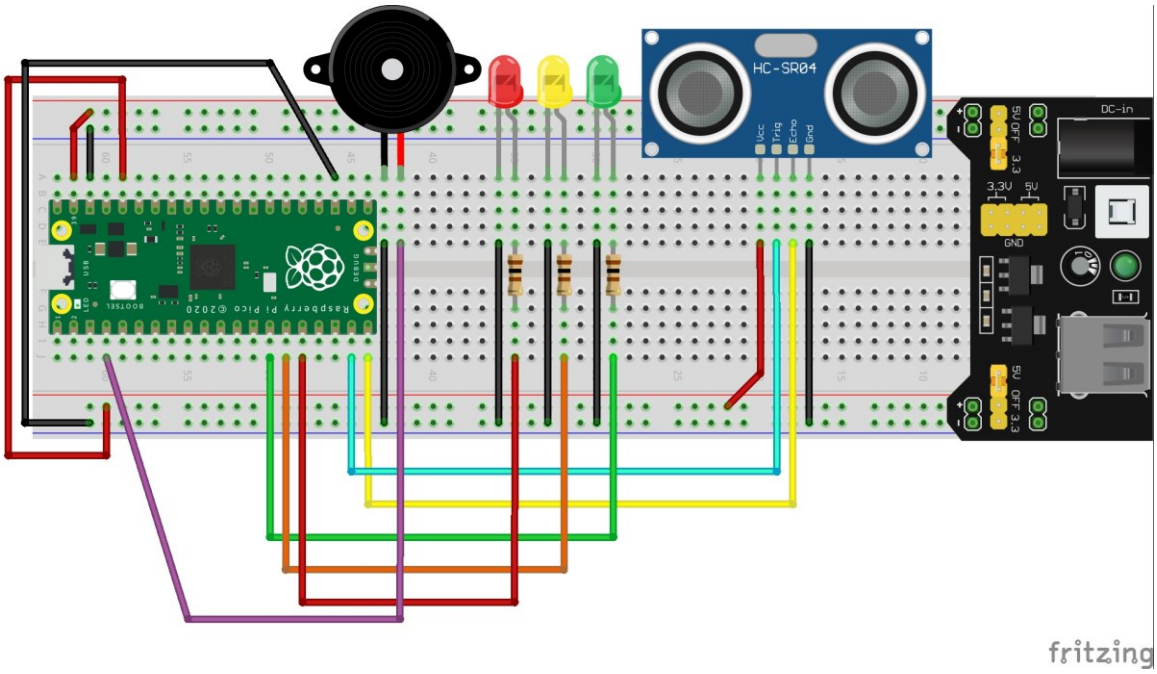
- Positive leg: Connect to GP11 of Raspberry Pi Pico board via a 220-ohm resistor
- Negative leg: Connect to GND of Raspberry Pi Pico board

Red LED:

- Positive leg: Connect to GP12 of Raspberry Pi Pico board via a 220-ohm resistor
- Negative leg: Connect to GND of Raspberry Pi Pico board

Buzzer:

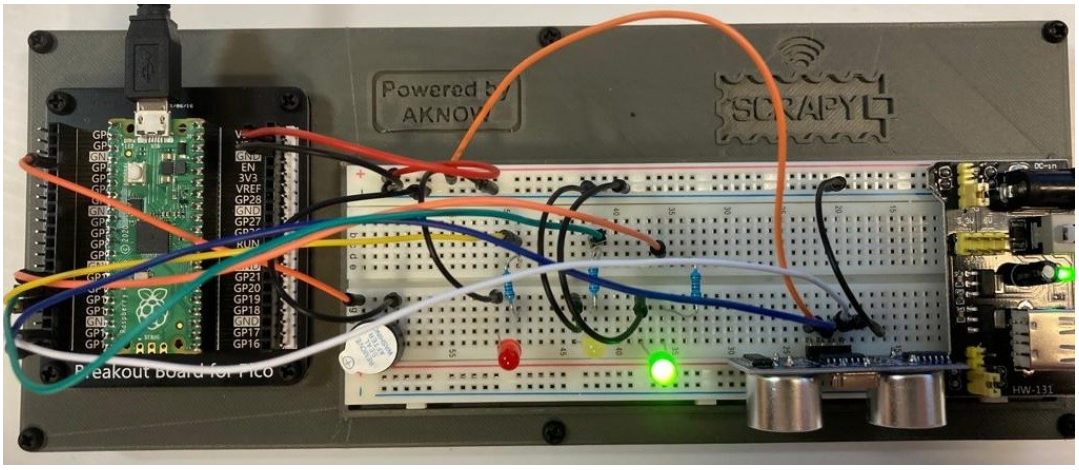
- Positive leg: Connect to GP2 of Raspberry Pi Pico board
- Negative leg: Connect to GND of Raspberry Pi Pico board

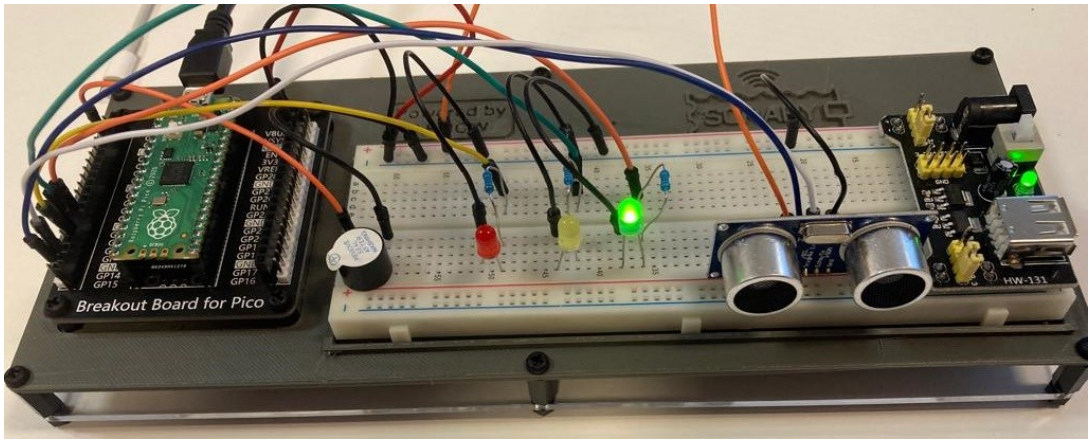


Code

```
# Import required libraries
from machine import Pin, time_pulse_us
import time
# Define pins for the components
trigger_pin = Pin(15, Pin.OUT)
echo_pin = Pin(14, Pin.IN)
green_led = Pin(10, Pin.OUT)
orange_led = Pin(11, Pin.OUT)
red_led = Pin(12, Pin.OUT)
buzzer = Pin(2, Pin.OUT)
# Define the distance thresholds for each LED
green_threshold = 20  # cm
orange_threshold = 5 # cm
# Define the function to calculate the distance from the HC-SR04 sensor
def get_distance():
    # Send a 10us pulse to trigger the sensor
    trigger_pin.low()
    time.sleep_us(2)
    trigger_pin.high()
    time.sleep_us(10)
    trigger_pin.low()
    # Measure the duration of the echo signal
    duration = time_pulse_us(echo_pin, 1, 10000)
    # Calculate the distance from the duration using the speed of sound (343
m/s)
    distance = duration / 2 / 1000000 * 343 * 100
    return distance
# Define the main loop to read the distance and control the LEDs and buzzer
while True:
    # Get the distance from the sensor and print it
    distance = get_distance()
    print(f"Distance : {distance} cm")
    # Turn on the green LED if the distance is greater than the threshold
    if distance > green_threshold:
        green_led.on()
        orange_led.off()
        red_led.off()
        buzzer.off()
    # Turn on the orange LED if the distance is between the thresholds
    elif distance > orange_threshold:
        green_led.off()
        orange_led.on()
        red_led.off()
        buzzer.off()
    # Turn on the red LED and buzzer if the distance is less than the threshold
    else:
        green_led.off()
        orange_led.off()
        red_led.on()
        buzzer.on()
        time.sleep(0.5)  # Buzz for 0.5 seconds
    # Wait for 0.1 second before the next measurement
          time.sleep(0.1)
```

Example pictures





Conclusion

In conclusion, this project involved using a Raspberry Pi Pico board, an HC-SR04 ultrasonic sensor, and some additional components to create a reversing radar for cars. We learned about the operating principles of the HC-SR04 sensor, how to wire the components together, and how to program the behavior of the system using MicroPython. By following the steps in this tutorial, we were able to create a system that can detect obstacles and provide visual and audible feedback to the driver.

As for further exploration, there are many ways to expand this project. Some suggestions include:

- Adding more LEDs or a display to provide more detailed distance feedback.
- Using machine learning algorithms to improve obstacle detection and accuracy.
- Creating a wireless interface to allow the system to communicate with a mobile device or other external device.
- Incorporating additional sensors or components to create a more comprehensive vehicle safety system.

Overall, this project provides a great introduction to the world of electronics and programming and serves as a starting point for further exploration and experimentation.