Tutorial 8: Smart Gas Detection Sensors

1. Introduction

1.1 What am I learning here and why?

Gas detection sensors play a crucial role in ensuring safety within enclosed spaces. Having a smart gas detection system provides a timely and automated response to potential gas leaks, enhancing overall safety measures. This system is designed to alert users to the presence of gas by triggering a buzzer alarm. This not only offers a quick response in case of emergencies but also allows for remote monitoring, providing peace of mind knowing that potential gas threats can be addressed promptly.

1.2 Learning objectives

In this tutorial, you will explore the implementation of a gas detection system using sensors and alarms. The scenario is based on the detection of gas within a specific environment, where the buzzer is activated upon sensing the presence of gas. The tutorial comprises three installation steps to guide you through the setup.

Upon completing the tutorial, you will:

- Understand the functionality of gas detection sensors
- Be familiar with buzzer alarms
- Be able to create an automated gas detection system

1.3 What do I need?

Software

So that you can carry out the installations shown in this tutorial you should have downloaded the Thonny programming environment on your device. Also, you need to have installed the firmware of MicroPython on your Raspberry Pi Pico. The extended modifications (see p 42 in manual) including the extra components and their connectivity must also be made on the breadboard.

Electrical Hardware

- 1x Raspberry Pi Pico
- 1x Full size breadboard
- 1x Micro-USB cable
- 1x MQ-135 Air Quality Sensor
- 1x Buzzer
- 5 x male-to-female jumper cables (20 cm)
    - 3 to connect gas detector
    - 2 to connect buzzer
- 11 x male-to- male jumper cables (10 cm)

To attach components at SmartHome4Seniors house model

- 2x Bolts
- 2x Nuts

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Ability

When it comes to using your hands, you need to be able to count the holes on the breadboard and put the parts into them correctly.

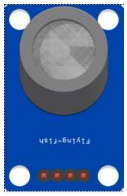
2 Learning content

2.3 Theoretical background

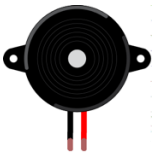
To understand the content of this tutorial well, you will now get an introduction to the most important terms and contexts.

Definitions

- **Air Quality Sensor:** This sensor is helpful in monitoring the air in your surroundings to ensure it's safe to breathe. It can detect various gases in the air, such as ammonia, methane, carbon dioxide, and other harmful substances. This is important for maintaining a healthy and safe environment, especially indoors.



- **Buzzer:** The buzzer is like an electronic beeper that makes noise. It's a simple but useful component that can be programmed to play different sounds or beeps. You can use it to create all sorts of sounds – from simple beeps to melodies. You can program the buzzer to make a sound when something specific happens, like receiving a message or when a particular event occurs. In our case it would be, that it makes a sound when it detects gas in the air.



2.4 Step-by-step guide

Let's proceed with the implementation of the scenario by referring to the SmartHome4Senior model or consulting the provided instructions and recordings for the gas detection example.

Two installation steps are required for our case:

1. Using the Air Quality Sensor to try to detect gas
2. Making the buzzer work, once it detects gas

To make the sensors work, you have to

- connect the electronics
- write a program code
- double check and run the program

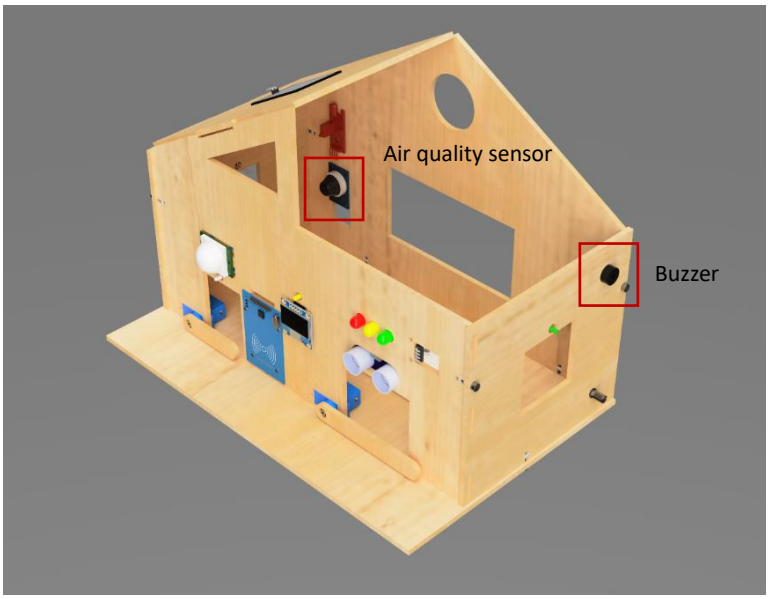
Additional Instructions:

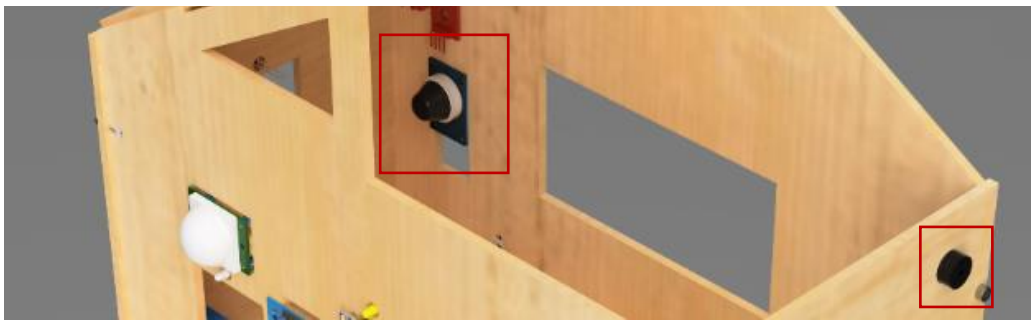
This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

- Materials: Gather all necessary materials mentioned in the respective tutorials for the installation and programming of the gas detection system.
- Software Installation: If not done previously, install the Thonny computer program on your computer. Additionally, ensure the Raspberry Pi Pico firmware is installed (refer to pp. 13-14 of the SmartHome4Seniors manual).
- Connection: Connect the Raspberry Pi Pico microcontroller to the breadboard. Subsequently, establish a connection between the Raspberry Pi Pico microcontroller and your computer/laptop using the USB cable.

2.2.1 Attach the components to the SmartHome4Seniors house model

1. Place the **air quality sensor** into the house from the exterior. Secure it by utilizing two bolts and two nuts through the top left and top right mounting holes.
2. Insert the **buzzer** to the upper-right corner slot (make sure its pins are on the inside of the house model) and friction will keep it in place.





2.2.2 Connect the electronics

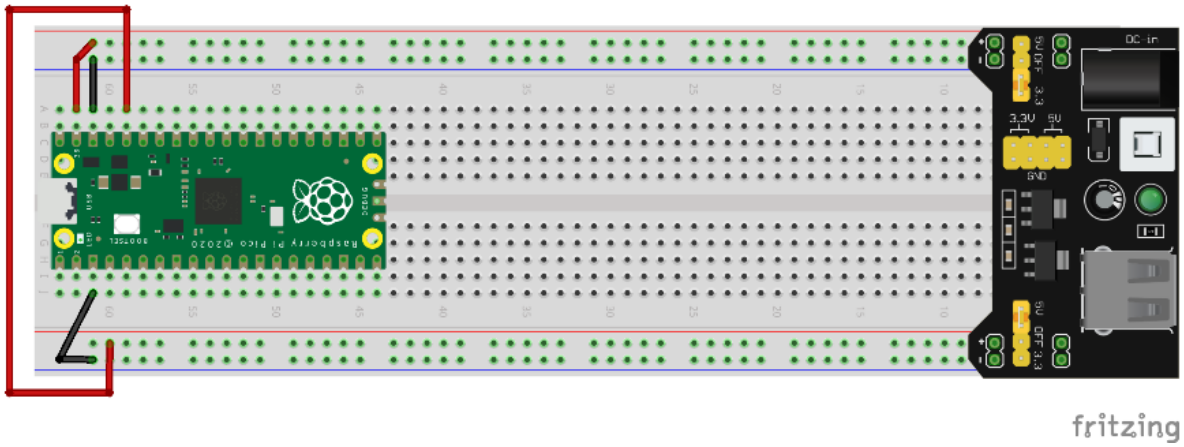
Now we have to connect the sensors with cables with our Raspberry Pi.

This is the base we need to start with:

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

– top side connections: VSYS 5V ((+) red) and GND ((-) black)

– bottom side connections: 3V3 ((+) red) and GND ((-) black)

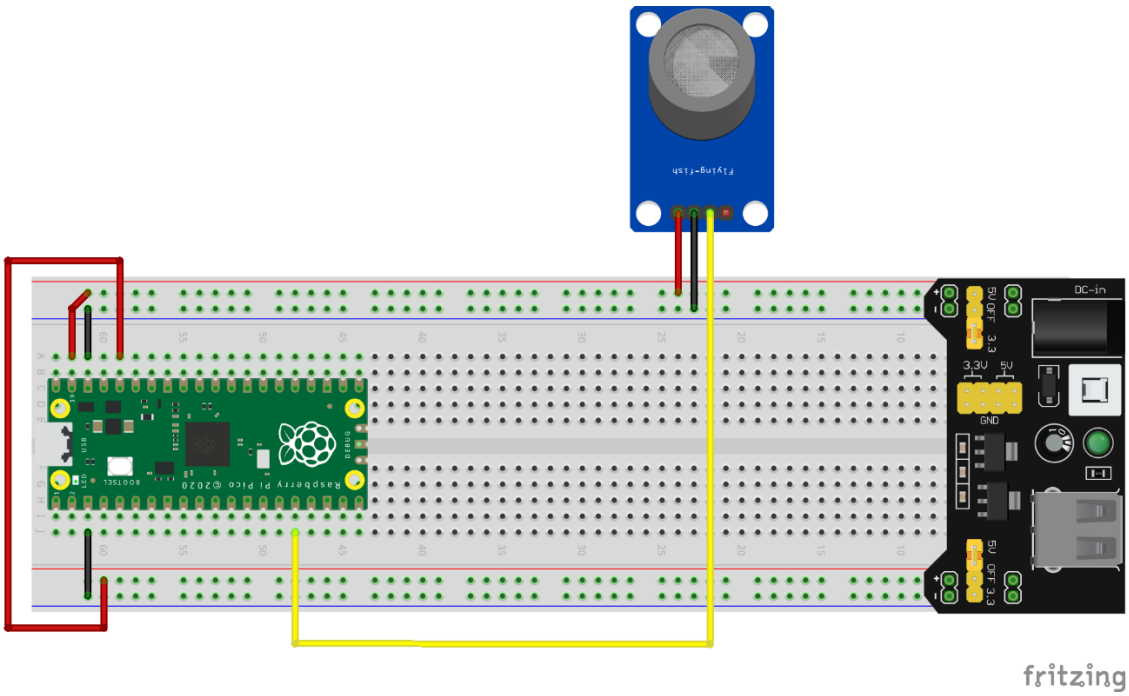


For installing the gas sensor:

– connect the red cable (VCC) to 5V rail (+)

– connect the black cable (GND) to GND rail (-)

– connect the yellow cable (DO/OUT) to GPIO12 pin

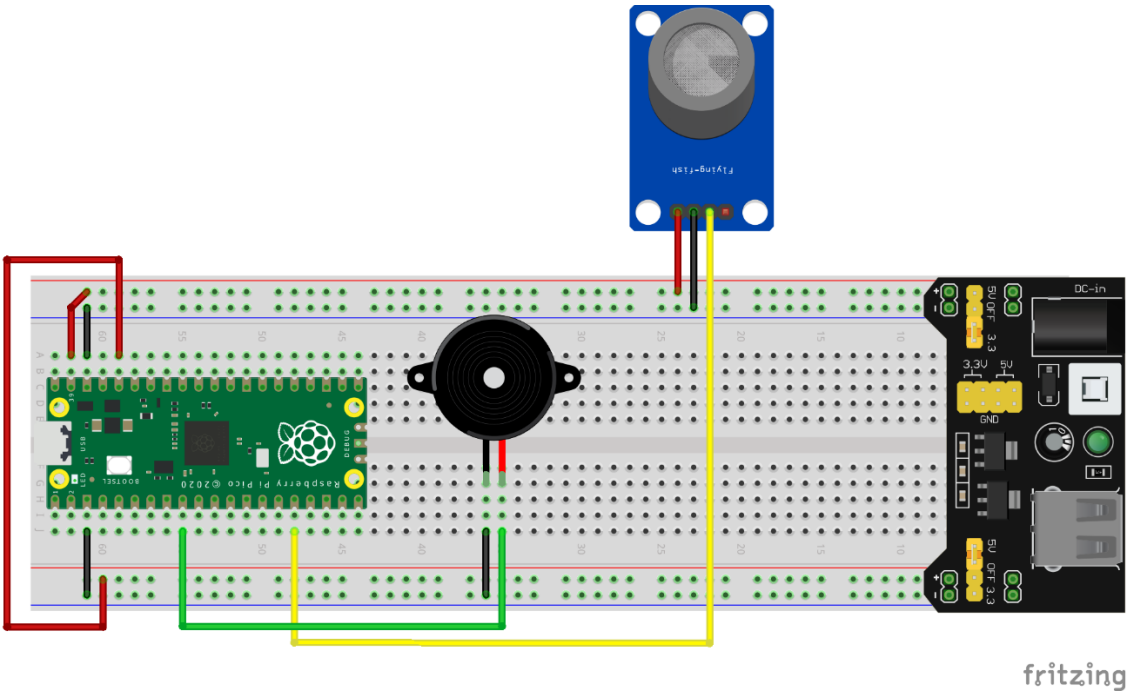


For connecting the buzzer:

– connect the longer end (+) of the buzzer to GPIO6 pin

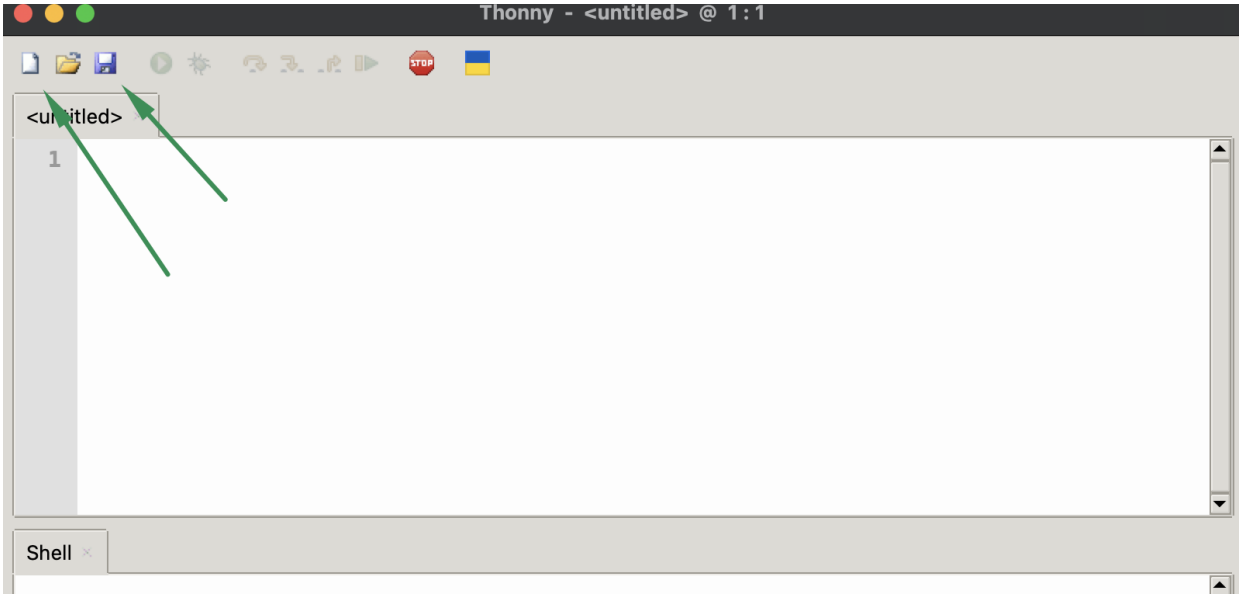
– connect the shorter end (-) of the buzzer to a GND rail

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



2.2.3 Generate program code

To enter the code, let's first open Thonny and create a new file. Before typing the code, it's advisable to save the file immediately as gasdetection.py (be sure not to forget the .py extension). Then, we can proceed with the code. Simply copy the code and read below for an explanation of what this code actually does.



This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
from machine import Pin
from time import sleep

#Define pins for each component
PIN_BUZZER = 6
PIN_GAS = 12

#Setup input and output
gas_sensor = Pin(PIN_GAS, Pin.IN)
buzzer = Pin(PIN_BUZZER, Pin.OUT)

while True:
    if gas_sensor() == 0:
        print("Gas Detected")
        #Activate the buzzer when gas is detected
        buzzer.value(1)
        sleep(0.5)
        #Turn off the buzzer after the delay
        buzzer.value(0)
    else:
        print("Gas Not Detected")
        sleep(1.5)
```

Here is a description, what the code actually does:

1. **Setting Up:** The code starts by importing necessary functions for handling pins and time.
2. **Defining Pins:** It defines two pins using Pin(). One is for the gas sensor (connected to GPIO pin 12), and the other is for the buzzer (connected to GPIO pin 6).
3. **Initialisation:** Import necessary modules: machine for GPIO pin control; sleep for introducing delays
4. **Main Loop:** The code enters a continuous loop (while True:) to keep checking for gas continuously.
5. **Gas Detection:** Inside the loop, it checks the value of the gas pin (gas_pin.value()). If it's 0, it means gas is detected.
    - If the gas sensor value is 0, it indicates gas detection.
      The buzzer is activated (buzzer.value(1)) when gas is detected.
    - A brief delay of 0.5 seconds is introduced (sleep(0.5)).
      The buzzer is turned off (buzzer.value(0)) after the delay.
    - If no gas is detected (gas sensor value is not 0), a message is printed, and a longer delay of 1.5 seconds is introduced.

2.2.4 Application

Now that you have completed the setup, you can perform a practical test by introducing a lighter in front of the gas detector. This test will help you evaluate how well your code responds to the introduction of gas, simultaneously checking if the buzzer activates to alert you. This hands-on demonstration allows you to validate the functionality of your code and the responsiveness of the gas detection system. Pay close attention to the system's reaction to the introduced gas source and ensure that the audible alarm is triggered, providing both a

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

visual and auditory indication of the gas detection.

3 Summary

In this tutorial, you learned how to use a gas sensor and a buzzer with a microcontroller, likely Raspberry Pi Pico, through Python code.

- The Python program continuously checks the gas sensor for detection, and upon sensing gas, it triggers the buzzer, providing an audible alert.
- This setup forms the basis for a simple gas detection system, which can be crucial for alerting users to potential gas leaks or the presence of harmful substances in the air.
- The tutorial demonstrates fundamental concepts, such as reading sensor values, making decisions based on those values, and controlling an output device (buzzer) accordingly, laying the groundwork for more advanced environmental monitoring and safety projects.

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.