StarFive 赛昉科技

Using VisionFive 2 GPIO to Make A Buzzer Beep

with Python

Application Note

Version: 1.2

Date: 2025/08/06

Doc ID: VisionFive2-ANEN-001

Legal Statements

Important legal notice before reading this documentation.

PROPRIETARY NOTICE

Information in this document is provided "as is," with all faults. Contents may be periodically updated or revised due to the product development. Shanghai StarFive Semiconductor Co., Ltd. (hereinafter "StarFive") reserves the right to make changes without further notice to any products herein.

StarFive expressly disclaims all warranties, representations, and conditions of any kind, whether express or implied, including, but not limited to, the implied warranties or conditions of merchantability, fitness for a particular purpose and non-infringement.

StarFive does not assume any liability rising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation indirect, incidental, special, exemplary, or consequential damages.

Contact Us

Address: Room 506, Building 2, No. 61 Shengxia Rd., China (Shanghai) Pilot Free Trade Zone, Shanghai, 201203, China

Website: http://www.starfivetech.com

Email: sales@starfivetech.com(sales) , support@starfivetech.com(support)

Preface

About this guide and technical support information.

About this document

This application note provides steps to use VisionFive 2's GPIO pins to make a buzzer beep through an example program with Python.

Revision History

Table 0-1 Revision History

Version	Released	Revision
1.2	2025/08/06	Updated the Linux and OS version in Environment Requirements *(on page 8)*. Updated the steps in Preparing Software *(on page 9)*. Updated the steps in Running Demo Code *(on page 12)*.
1.1	2023/06/08	• Added a note in 40-Pin GPIO Header Definition *(on page 7)*. • Updated the method for installing `VisionFive.gpio` package in Preparing Software *(on page 9)*. • Added Resources *(on page 15)* and Buy Now *(on page 16)* chapters.
1.0	2022/11/30	The first official release.

Notes and notices

The following notes and notices might appear in this guide:

- **Tip:**
  Suggests how to apply the information in a topic or step.
- **Note:**
  Explains a special case or expands on an important point.
- **Important:**
  Points out critical information concerning a topic or step.
- **CAUTION:**
  Indicates that an action or step can cause loss of data, security problems, or performance issues.
- **Warning:**
  Indicates that an action or step can result in physical harm or cause damage to hardware.

Contents

List of Tables

List of Figures

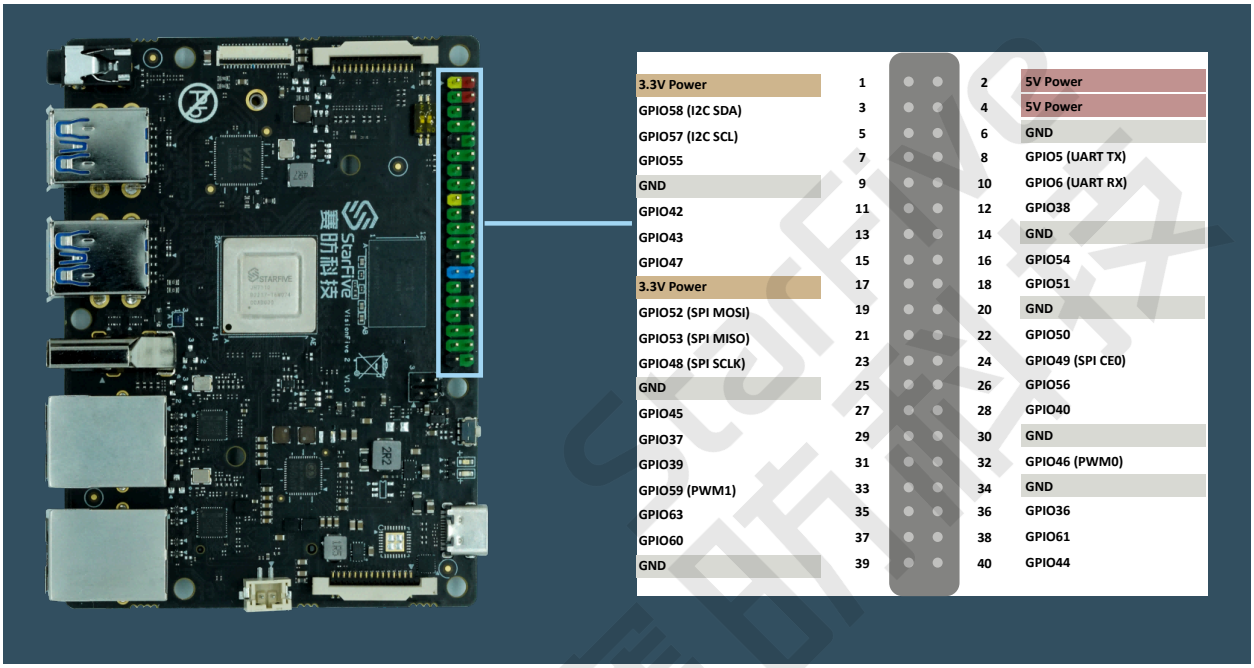
1. Introduction

This application note provides steps to use VisionFive 2's GPIO pins to make a buzzer beep through an example program with Python.

1.1. 40-Pin GPIO Header Definition

The following figure shows the location of the 40-pin header on VisionFive 2.

Figure 1-1 40-Pin GPIO Header Definition



Note:
The multiplexed pin has been initialized and cannot be used as a general GPIO.

2. Preparation

Before executing the demo program, make sure you prepare the following:

2.1. Environment Requirements

The environment requirements are as follows:

- Linux Kernel: Linux 6.6
- OS: Debian 13
- SBC: VisionFive 2
- SoC: JH-7110

2.2. Preparing Hardware

Prepare the following hardware items before running the demo code:

Table 2-1 Hardware Preparation

Type	M/O*	Item	Notes
General	M	VisionFive 2 Board	-
General	M	• 32 GB (or more) micro-SD card • Micro-SD card reader • Computer (Windows/Mac OS/Linux) • USB to serial converter (3.3 V I/O) • Ethernet cable • Power adapter (5 V / 3 A) • USB Type-C Cable	These items are used for flashing Debian OS into a Micro-SD card.
GPIO Demo (Buzzer)	M	A passive buzzer	-

Note:
*: M: Mandatory, O: Optional

2.2.1. Hardware Setup

The following table and figure describe how to connect the buzzer to the 40-pin GPIO header:

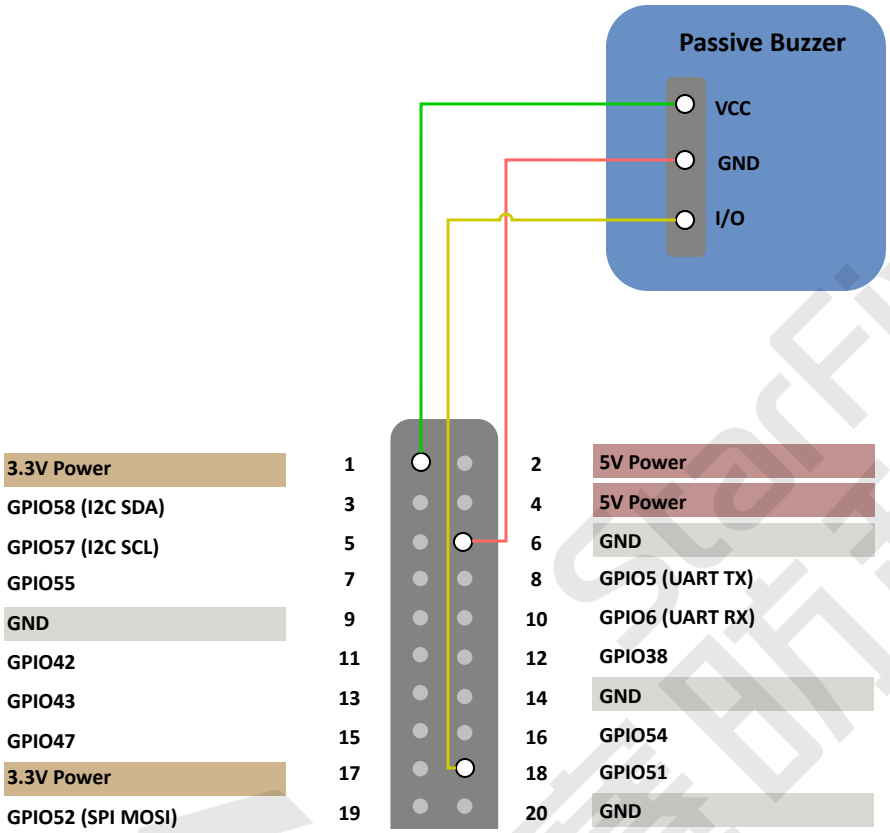
Table 2-2 Connect Buzzer to the 40-Pin Header

Passive Buzzer	40-Pin GPIO Header	
	Pin Number	Pin Name
VCC	1	3.3V Power
GND	6	GND

Table 2-2 Connect Buzzer to the 40-Pin Header (continued)

Passive Buzzer	40-Pin GPIO Header	
	Pin Number	Pin Name
I/O	18	GPIO51

Figure 2-1 Connect Buzzer to the 40-Pin GPIO Header



2.3. Preparing Software

Make sure the following procedures are performed:

Note:
The python project, `VisionFive.gpio`, is applicable for VisionFive, VisionFive 2 and JH-7110 EVB.

1. Flash Debian OS into a Micro-SD card as described in the *Flashing Fedora OS to a Micro-SD Card* section in the *VisionFive 2 Single Board Computer Quick Start Guide*.
2. Log into the Debian and make sure VisionFive 2 is connected to the Internet. For detailed instructions, refer to the Using SSH over Ethernet or *Using a USB to Serial Converter* section in the *VisionFive 2 Single Board Computer Quick Start Guide*.
3. Extend the partition on Debian as described in *Extend Partition* in the *VisionFive 2 Single Board Computer Quick Start Guide*.
4. Execute the following command to install and create a Python3 Virtual Environment on Debian:

```
sudo apt install python3-venv
python3 -m venv myvenv
```

Note:

You may rename “myvenv” according to your preference.

5. Execute the `pip` command on VisionFive 2 Debian to install the `VisionFive.gpio` package:

Note:

Due to the fact that `pypi.org` official website does not yet support uploading `whl` installation packages for the RISC-V platform, so it cannot directly execute `python3 -m pip install VisionFive.gpio` command to install online.

Please follow the steps below to install the `VisionFive.gpio` package.

a. Execute the following command to install dependent package within the newly created virtual environment:

```
sudo apt install libxml2-dev libxslt-dev
source ./myvenv/bin/activate
python3 -m pip install requests wget bs4
```

b. Execute the following command to run the installation script `Install_VisionFive_gpio.py`:

```
python3 Install_VisionFive_gpio.py
```

The installation script codes are as follows:

```
import requests
import wget
import sys
import os
from bs4 import BeautifulSoup

def parse_data(link_addr, class_type, key_str):
    req = requests.get(url=link_addr)
    req.encoding = "utf-8"
    html = req.text
    soup = BeautifulSoup(req.text, features="html.parser")
    package_version = soup.find(class_type, class_=key_str)
    dd = package_version.text.strip()
    data = dd.split()
    return data

def parse_link(link_addr, class_type, key_str):
    version_list = []
    req = requests.get(url=link_addr)
    req.encoding = "utf-8"
    html = req.text
    soup = BeautifulSoup(req.text, features="html.parser")
    search_data = soup.find_all(class_type, class_=key_str)
    for i in range(0, len(search_data)):
        search_data[i] = search_data[i].find("a").get("href)
        version_list.append(search_data[i].split("cp")[-1].split("-")[0])

    python_version = sys.version
    python_version = python_version.split(".")[0] + python_version.split(".")[1]

    for i in range(0, len(search_data)):
        if python_version == version_list[i]:
            return search_data[i]

    return search_data[0]

def get_dl_addr_page():
    link_address = "https://pypi.org/project/VisionFive.gpio/#history"
    key_str = "release version"
    class_key = "p"
    data_get = parse_data(link_address, class_key, key_str)
    latest_version = data_get[0]
```

```
    dl_addr_page
 = "https://pypi.org/project/VisionFive.gpio/{}/#files".format(latest_version)
    return dl_addr_page

def get_dl_addr_of_latest_version(link_addr):
    key_str = "card file card"
    class_key = "div"
    addr_get = parse_link(link_addr, class_key, key_str)

    return addr_get

def main():
    dl_addr_p = get_dl_addr_page()
    whl_dl_addr = get_dl_addr_of_latest_version(dl_addr_p)

    whl_name = whl_dl_addr.split("/")[-1]
    whl_name_suffix = os.path.splitext(whl_name)[-1]
    whl_name_prefix = os.path.splitext(whl_name)[0]
    whl_name_prefix_no_platform = whl_name_prefix[0: len(whl_name_prefix) - 3]
    new_platform = "linux_riscv64"

    rename_whl_name = "{}{}{}".format(whl_name_prefix_no_platform, new_platform,
 whl_name_suffix)

    wget.download(whl_dl_addr, out=rename_whl_name)

    os.system("pip install " + rename_whl_name)
    os.system("rm -rf " + rename_whl_name)

if __name__ == '__main__':
    sys.exit(main())
```

c. (Optional) Exit the Python3 Virtual Environment.

```
deactivate
```

3. Running Demo Code

To run the demo code, perform the following on VisionFive 2 Debian:

1. Locate to the directory where the code, `buzzer.py`, exists:
    a. Source into the Python3 Virtual Environment:

    ```
    source ./myvenv/bin/activate
    ```

    b. Execute the following command to install dependency:

    ```
    python3 -m pip install pillow
    ```

    c. Execute the following command to get the directory where `VisionFive.gpio` exists:

    ```
    python3 -m pip show VisionFive.gpio
    ```

    Result:

    ```
    Location: /home/user/myvenv/lib/python3.11/site-packages
    ```

    Note:

    The actual output depends on how the application is installed.

    d. Execute the following to enter the directory, for example, `/home/user/myvenv/lib/python3.11/site-packages` as indicated in the previous step output:

    ```
    cd /home/user/myvenv/lib/python3.11/site-packages
    ```

    e. Execute the following command to enter the `sample-code` directory:

    ```
    cd ./VisionFive/sample-code/
    ```

2. Under the `sample-code` directory, execute the following command:

    ```
    sudo python buzzer.py
    ```

    Alternatively, you can execute the following command:

    ```
    sudo python3 buzzer.py
    ```

3. Enter the values as prompted to configure the pitch and the lasting time of the beeps as prompted:
    - **Enter Pitch (200 to 20000)**: The frequency (Range: 200 to 20,000) of the buzzer. Unit: Hz. For example, `400`.

      Note:
      If the value is out of the range (200 to 20,000 Hz) , a warning message will display, and you need to re-enter the pitch value.

    - **Enter Cycle (seconds)**: The time in seconds when the buzzer keeps beeping. For example, `100`.

    Example:

    ```
    [riscv@fedora-starfive sample-code]$ sudo python3 buzzer.py
    Enter Pitch (200 to 20000): 400
    Enter Cycle (seconds): 100
    ```

    Result:

    The buzzer keeps beeping for 100 seconds at 400 Hz.

4. (Optional) Exit the Python3 Virtual Environment.

    ```
    deactivate
    ```

4. Demo Source Code

The Python source code of this demo is provided for reference purposes only.

`buzzer.py`:

```
'''
Please make sure the buzzer is connected to the correct pins.
The following table describes how to connect the buzzer to the 40-pin header.
-----------------------------------------
Passive Buzzer___Pin Number_____Pin Name
    VCC             1         3.3V Power
    GND             6           GND
    I/O             18          GPIO51
-----------------------------------------
'''

import VisionFive.gpio as GPIO
import time

buzz_pin = 18
ErrOutOfRange = 0

def setup():
    #Configure the direction of buzz_pin as out.
    GPIO.setup(buzz_pin, GPIO.OUT)
    #Configure the voltage level of buzz_pin as high.
    GPIO.output(buzz_pin, GPIO.HIGH)

def pitch_in_check():
    val_in = input('Enter Pitch (200 to 20000): ')
    val = float(val_in)

    if 200 <= val <= 20000:
        return val
    else:
        print('The input data is out of range (200 to 20,000 Hz). Please re-enter.')
        return ErrOutOfRange

def loop(pitch, cycle):
    delay = 1.0 / pitch
    cycle = int((cycle * pitch)/2)

    #Buzzer beeps.
    while cycle >= 0:
        GPIO.output(buzz_pin, GPIO.LOW)
        time.sleep(delay)
        GPIO.output(buzz_pin, GPIO.HIGH)
        time.sleep(delay)

        cycle = cycle - 1

def destroy():
    GPIO.output(buzz_pin, GPIO.HIGH)
    GPIO.cleanup()

if __name__ == '__main__':
    setup()
    try:
        #Input value of pitch (200 to 20,000 Hz).
        pitch = pitch_in_check()
        while pitch == 0:
            pitch = pitch_in_check()

        #Input value of cycle time (seconds).
        cycle_in = input("Enter Cycle (seconds): ")
        cycle = int(cycle_in)

        #The buzzer beeps with the specified pitch and cycle.
        loop(pitch, cycle)
```

```
    finally:
        destroy()
```

5. Resources

Click on this tab to find all SBC relevant resources.

StarFive provides the following resources to guide you through an extraordinary experience on using the VisionFive 2 SBC.

- RVspace Wiki
- Application Center
- Documentation Center
- Technical Forum
- VisionFive 2 GitHub Repository
- VisionFive 2 Debian OS Download
- Code download
- View All PDF Documents

6. Buy Now

Click on this tab to find all the online shops and compatible accessories.

Buy SBC

Use the following page to find your nearest sales channel or the global channels for purchasing a VisionFive 2 Single Board Computer (SBC).

- Buy VisionFive 2

Buy Parts

Use the following page to find the parts that are tested as compatible with VisionFive 2.

- Buy Accessory