StarFive 赛昉科技

# Using VisionFive 2 GPIO to Detect Button in a Specified Time

with Python

Application Note

Version: 1.1

Date: 2023/06/08

Doc ID: VisionFive 2-ANEN-011

# 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 Technology 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 502, Building 2, No. 61 Shengxia Rd., China (Shanghai) Pilot Free Trade Zone, Shanghai, 201203, China Room 502, Building 2, No. 61 Shengxia Rd., China (Shanghai) Pilot Free Trade Zone, Shanghai, 201203, China

Website: http://www.starfivetech.com http://www.starfivetech.com

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

# Contents

List of Tables....4
List of Figures....5
Legal Statements....ii
Preface....vi
**1. Introduction....7**
1.1. 40-Pin GPIO Header Definition....7
**2. Preparation....8**
2.1. Environment Requirements....8
2.2. Preparing Hardware....8
2.2.1. Hardware Setup....10
2.3. Preparing Software....11
**3. Running Demo Codes....15**
**4. Demo Source Code....17**
**5. Resources....19**
**6. Buy Now....20**

# List of Tables

Table 0-1 Version History..... vi
Table 2-1 Hardware Preparation..... 8
Table 2-2 Connect button to the 40-Pin Header..... 10

# List of Figures

Figure 1-1 40-Pin GPIO Header Definition..... 7
Figure 2-1 Breadboard Overview..... 10
Figure 2-2 Connect the button to the 40-Pin Header.....11

# Preface

About this guide and technical support information.

## About this document

This application note provides steps to use VisionFive 2's GPIO pins to detect rising edge and falling edge in a specified time through an example program with Python. Detecting the rising or falling edge means detecting the signal of the button.

## Version History

**Table 0-1 Version History**

| Version | Released | Revision |
|---|---|---|
| 1.1 | 2023/06/08 | Updated the method for installing `VisionFive.gpio` package in Preparing Software *(on page 11)*. |
| 1.0 | 2023/05/31 | 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.

# 1. Introduction

This application note provides steps to use VisionFive 2's GPIO pins to detect rising edge and falling edge in a specified time through an example program with Python. Detecting the rising or falling edge means detecting the signal of the button.

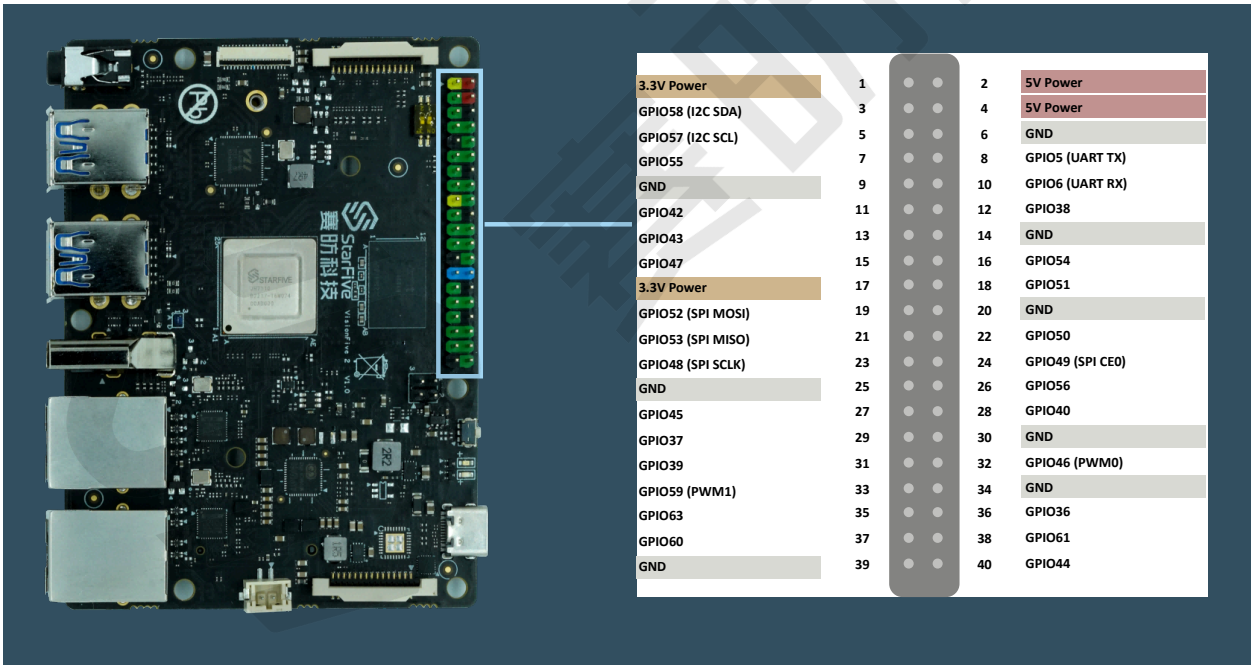
**Tip:**

`VisonFive.gpio` is compatible with the `RPi.GPIO` command, which means that the `RPi.GPIO` python demo can be run on VisionFive 2. In addition, the callback function of API `add_event_detect()` has been optimized compared to `RPi.GPIO`, which adds a *edge_type* parameter in callback function. Therefore, the python demo of `RPi.GPIO` related to callback functions needs to be modified manually by adding *edge_type* parameter.

## 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 5.15
- OS: Debian 12
- SBC: VisionFive 2
- SoC: JH7110

## 2.2. Preparing Hardware

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

**Table 2-1 Hardware Preparation**

| Type | M/O* | Item | Notes |
|---|---|---|---|
| General | M | VisionFive 2 Board | - |
| General | M | • 32 GB (or more) micro-SD card<br>• Micro-SD card reader<br>• Computer (Windows/Mac OS/Linux)<br>• USB to serial converter (3.3 V I/O)<br>• Ethernet cable<br>• Power adapter (5 V / 3 A)<br>• USB Type-C Cable | These items are used for flashing Fedora OS into a micro-SD card. |
| Button detection demo | M | • An LED<br>• A Breadboard<br>• Two Male-Female jumper wires<br>• A 4-pin button | • LED stands for Light Emitting Diode, and glows when electricity is passed through it. The longer leg (known as the 'anode'), is always connected to the positive supply of the cir- |

**Table 2-1 Hardware Preparation (continued)**

| Type | M/O* | Item | Notes |
|---|---|---|---|
| | | | cuit. The shorter leg (known as the 'cathode') is connected to the negative side of the power supply, known as 'ground'.<br>• Breadboard: Refer to the introduction below. |

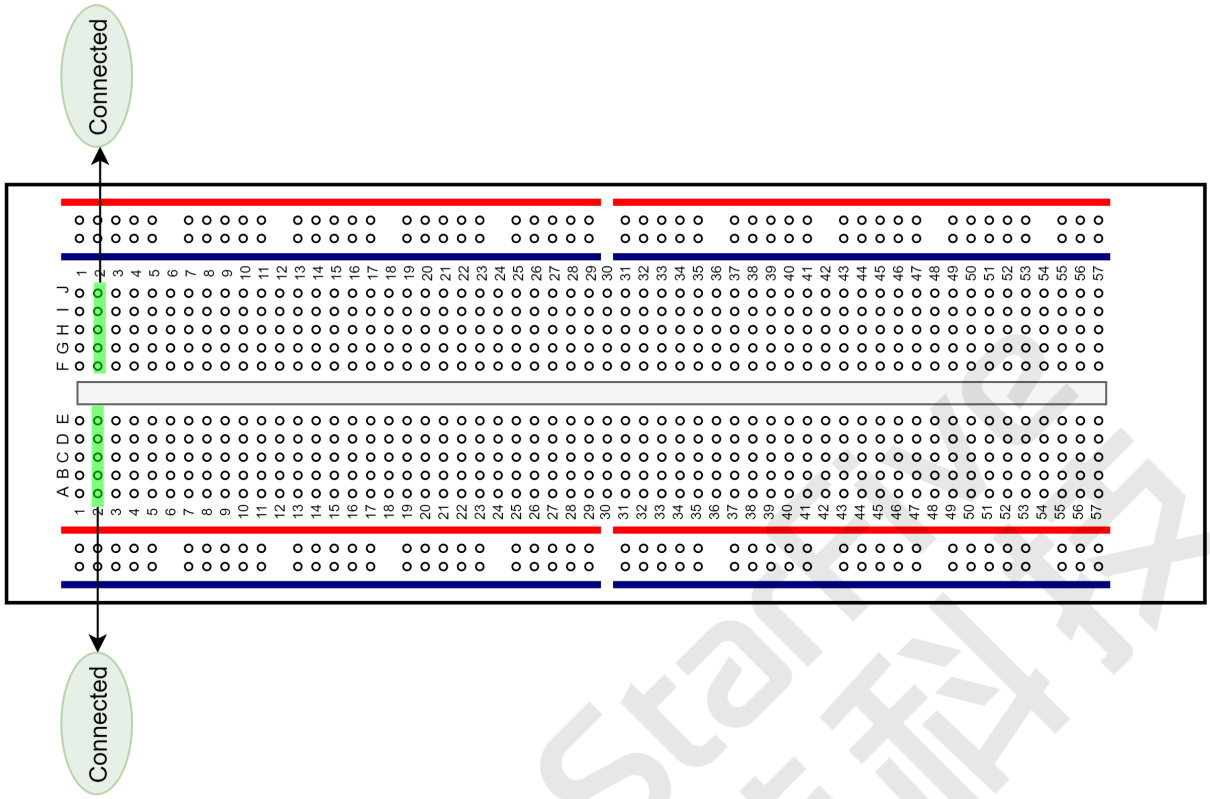
**Note:**
*: M: Mandatory, O: Optional

## Breadboard Introduction

The breadboard is a way of connecting electronic components to each other without having to solder them together. They are often used to test a circuit design before creating a Printed Circuit Board (PCB). As shown in the following figure, there are two lines at the top and the bottom respectively of the breadboard. These two lines are used for power connection: the blue line is for negative and the red line is for positive. Besides, they are divided into two sections, and the holes in each section are connected.

In each column (from A to E, and F to J), holes are connected electrically. In each row (from 1 to 57), holes are not connected.

**Figure 2-1 Breadboard Overview**



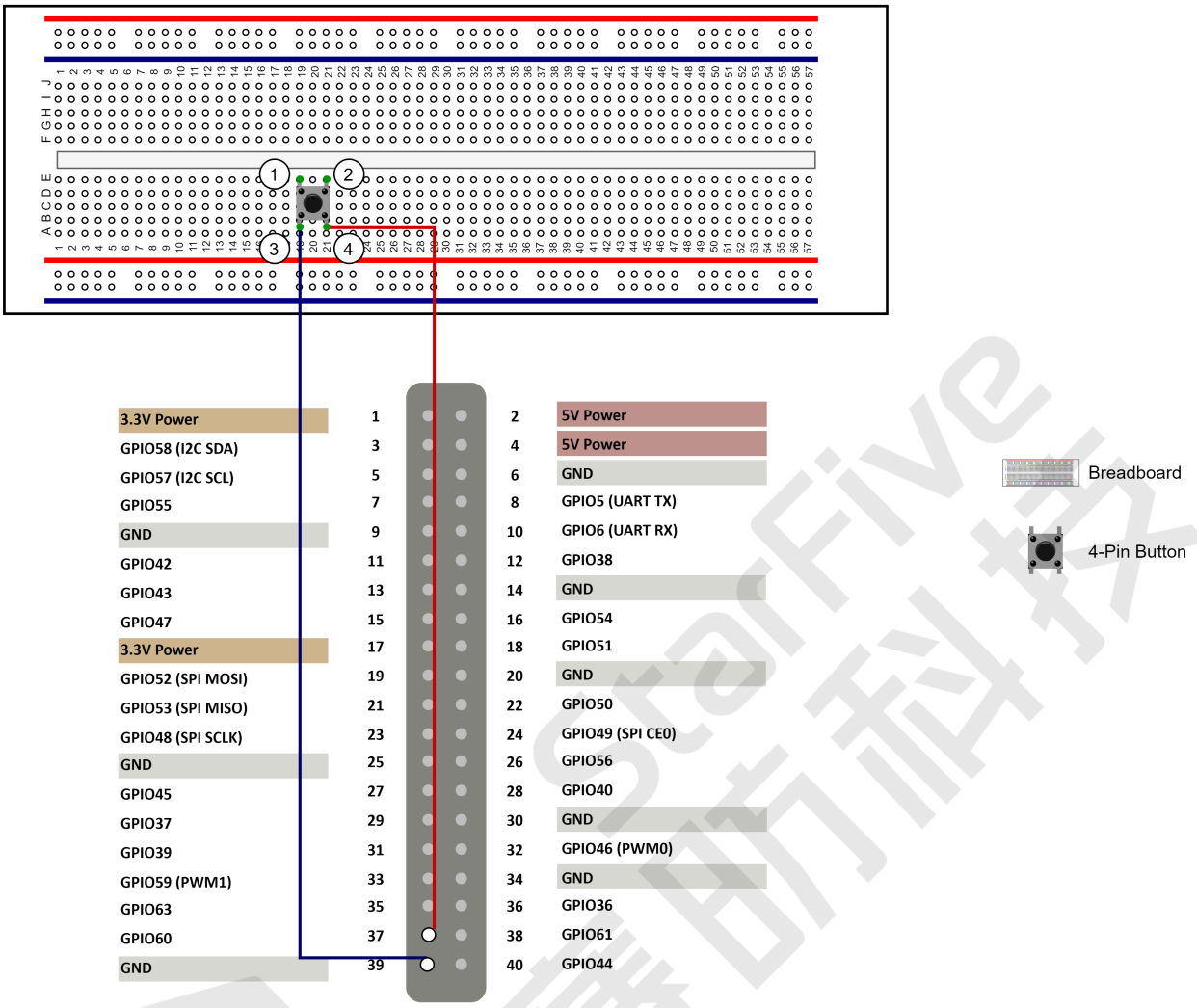

## 2.2.1. Hardware Setup

To setup hardware, connect buttons pin ① and pin ② to the breadboard first, and then connect pin ③ and pin ④ to VisionFive 2. The following table and figure describe how to connect button to the 40-pin GPIO Header:

**Table 2-2 Connect button to the 40-Pin Header**

| Button | 40-Pin GPIO Header | |
|---|---|---|
| | Pin Number | Pin Name |
| Pin ④ | 37 | GPIO60 |
| Pin ③ | 39 | GND |

**Figure 2-2 Connect the button to the 40-Pin Header**




**Tip:**

Inside the button, the pins ① and ③ are connected while the pins ② and ④ are connected.

## 2.3. Preparing Software

Make sure the following procedures are performed:

**Note:**

The python project, `VisionFive.gpio`, is applicable for VisionFive, VisionFive 2 and JH7110 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 PIP on Debian:

```
apt-get install python3-pip
```

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 `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:

```
apt install libxml2-dev libxslt-dev
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):
    req = requests.get(url=link_addr)
    req.encoding = "utf-8"
    html=req.text
    soup = BeautifulSoup(req.text,features="html.parser")
    search_data = soup.find(class_type,class_=key_str)
    search_data_2 = search_data.find("a")
    dl_link_get = search_data_2.get("href")
    return dl_link_get

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".for
mat(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())
```

# 3. Running Demo Codes

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

1. Locate to the directory where the test code, `edge_detection_with_waiting_time.py`, exists:

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

   ```
   pip show VisionFive.gpio
   ```

   **Result:**

   ```
   Location: /usr/local/lib64/python3.9/site-packages
   ```

   **Note:**

   The actual output depends on how the application is installed.

   b. Execute the following to enter the directory, for example, `/usr/local/lib64/python3.9/site-packages` as indicated in the previous step output:

   ```
   cd /usr/local/lib64/python3.9/site-packages
   ```

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

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

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

   ```
   sudo python edge_detection_with_waiting_time.py
   ```

   Alternatively, you can execute the following command:

   ```
   sudo python3 edge_detection_with_waiting_time.py
   ```

   **Result:**

   - The terminal displays as follows:

   ```
   # python3 edge_detection_with_waiting_time.py
   *----------------------Case 1-----------------------------------
   -----------------*
   Note: don't press the key on pin 37 once within 5 seconds !!!
   ```

   - After waiting for 5 seconds, the terminal displays as follows:

```
Edge hasn't been detected within 5 seconds  while setting 5
 seconds to timeout.
The return value of GPIO.event_detected(37) within 5 seconds:
 False

*----------------------Case 2-----------------------------------
----------------------------------*
Please press the key on pin 37 once at any time !!!
```

- Finally press the button, and the terminal displays as follows:

```
Edge has detected while setting -1 to timeout,
timeout -1 means waiting until edge is detected.

The return value (True) of GPIO.event_detected(37) should be True.
The return value (False) of GPIO.event_detected(37) should be
 False, because of the secondly reading.
```

# 4. Demo Source Code

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

`edge_detection_with_waiting_time.py`:

```
'''
Please make sure the buttion is connected to the correct pins.
The following table describes how to connect the button to the 40-pin
 header.
-----------------------------------------
______button_____Pin Number_____Pin Name
    one end          37          GPIO60
  The other end      39            GND
-----------------------------------------
'''

import VisionFive.gpio as GPIO
import sys
import time

key_pin = 37

def main():
    # Configure the direction of key_pin as input.
    GPIO.setup(key_pin, GPIO.IN)

    print("*----------------------Case
 1-----------------------------------------------------*")
    print("Note: don't press the key on pin {} once within 5
 seconds !!!".format(key_pin))
    print()
    # edge falling can be detected, also set bouncetime(unit: millisecond)
 to avoid jitter.
    # timeout(unit: millisecond), it means if edge will be detected within
 timeout time.
    # timeout -1 means  waiting until edge is detected.
    edge_detected = GPIO.wait_for_edge(key_pin, GPIO.FALLING, bouncetime=2,
 timeout=5000)


    if edge_detected == key_pin:
        print("Edge has detected within 5 seconds while setting 5 seconds to
 timeout.")
    else:
        print("Edge hasn't been detected within 5 seconds  while setting 5
 seconds to timeout.")
```

```
    # query if edge event happens.
    edge_detected_flag = GPIO.event_detected(key_pin)

    print("The return value of GPIO.event_detected({}) within 5 seconds:
{}".format(key_pin, edge_detected_flag))

    print()
    print("*----------------------Case
2----------------------------------------------------------------------*")
    print("Please press the key on pin {} once at any
time !!!".format(key_pin))

    # edge rising can be detected, also set bouncetime(unit: millisecond) to
avoid jitter.
    # the default timeout is -1, meaning that waiting until edge is
detected.
    edge_detected = GPIO.wait_for_edge(key_pin, GPIO.RISING, bouncetime=2)

    if edge_detected == key_pin:
        print("Edge has detected while setting -1 to timeout,")
        print("timeout -1 means waiting until edge is detected.")
    else:
        print("Edge hasn't been detected while setting -1 to timeout,")
        print("timeout -1 means waiting until edge is detected.")

    # query if edge event happens.
    edge_detected_flag = GPIO.event_detected(key_pin)

    print()
    print("The return value ({}) of GPIO.event_detected({}) should be
True.".format(edge_detected_flag, key_pin))

    # query if edge event happens.
    edge_detected_flag = GPIO.event_detected(key_pin)

    print("The return value ({}) of GPIO.event_detected({}) should be False,
because of the secondly reading.".format(edge_detected_flag, key_pin))

    print()



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

# 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 to VisionFive 2.

- Buy Accessory