StarFive 40-Pin Header User Guide

Version: V1.1

Date: 2021-12-27

Shanghai StarFive Technology Co., Ltd.

Address:	Room 502, Building 2, No. 61 Shengxia Rd., China (Shanghai) Pilot Free Trade Zone, Shanghai, 201203, China
Website:	www.starfivetech.com
Email:	sales@starfivetech.com (sales)
	support@starfivetech.com (support)

About This Manual

Introduction

This document is intended to:

- introduce the 40-pin header.
- provide instructions to configure and debug GPIO, I2C, SPI, PWM, and UART.
- provide peripheral examples to use 40-pin header.

Revision History

Version	Released	Revision
V1	2021-12-08	The first official release.
V1.1	2021-12-27	• Updated the command and improved description in the *Generating dtb* section. • Added a note in the *GitHub Repository* section.

Table of Contents

1 Overview

The 40-pin header allows StarFive single board computers, including both StarLight and VisionFive, to interface with a variety of external components., which enabling users to create their projects. This document is intended to:

- introduce the 40-pin header as described in this chapter.
- provide instructions to configure and debug GPIO, I2C, SPI, and PWM, as described in *Preparation*, *GPIO Operations*, *I2C Operations*, *SPI Operations*, *PWM Operations,* and *UART Operations* chapters.
- provide peripheral examples to use 40-pin header, as described in *Peripheral Examples* chapter.

1.1 40-Pin Header Definition

The following figure shows the location of the 40-pin header. The VisionFive board is taken as an example:

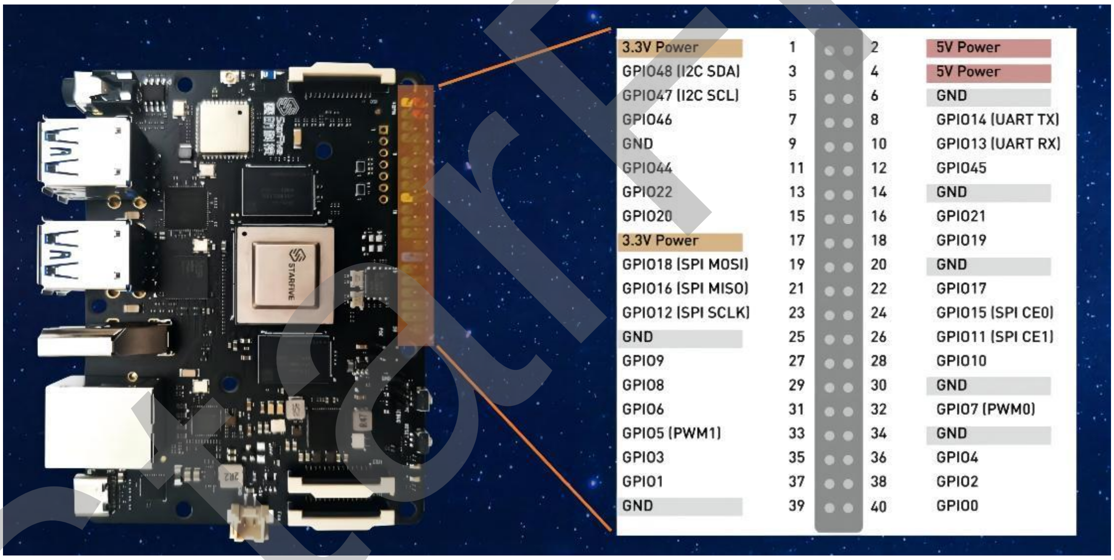


Figure 1-1 40-Pin Definition

1.2 GPIO Pinout

The following table describes the GPIO pinout, the map and explanation of what each pin can do:

Table 1-1 GPIO Pinout

dts	sys	Pin Name	Num	Num	Pin Name	sys	dts
		3.3V Power	1	2	5V Power		
i2c1	i2c-1	GPIO48 (I2C SDA)	3	4	5V Power		
i2c1	i2c-1	GPIO47 (I2C SCL)	5	6	GND		
	494	GPIO46	7	8	GPIO14 (UART TX)	ttyS0	uart3
		GND	9	10	GPIO13 (UART RX)	ttyS0	uart3
	492	GPIO44	11	12	GPIO45	PWM2	
	470	GPIO22	13	14	GND		
	468	GPIO20	15	16	GPIO21	469	
		3.3V Power	17	18	GPIO19	467	
spi2	spidev0.0	GPIO18 (SPI MOSI)	19	20	GND		
spi2	spidev0.0	GPIO16 (SPI MISO)	21	22	GPIO17	465	
spi2	spidev0.0	GPIO12 (SPI SCLK)	23	24	GPIO15 (SPI CE0)	spidev0.0	spi2
		GND	25	26	GPIO11 (SPI CE1)	spidev0.0	spi2
	457	GPIO9	27	28	GPIO10	458	
	456	GPIO8	29	30	GND		
	454	GPIO6	31	32	GPIO7 (PWM0)	PWM0	
	PWM1	GPIO5 (PWM1)	33	34	GND		
	451	GPIO3	35	36	GPIO4	452	
	449	GPIO1	37	38	GPIO2	450	
		GND	39	40	GPIO0	448	

2 Preparation

Before configuring and debugging the GPIOs, you need to prepare the follows:

2.1 Preparing Hardware

The following tale describes hardware items to be prepared if you want to configure, debug, and test this 40-pin header by following this guide:

Table 2-1 Hardware Preparation

Type	M/O	Item	Notes
General	M	A Single Board Computer	The following boards are applicable: • StarLight • VisionFIve
General	M	• 16GB (or more) micro-SD card • micro-SD card reader • Computer (PC/Mac/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 Fedora OS into a micro-SD card.
GPIO	O	An oscilloscope	The oscilloscope is used to measure the corresponding pin and check the PWM period and duty cycle.
I2C	O	• Sense Hat (B) • Dupont Line	-
SPI	O	• ADXL345 Module • Dupont Line	-

Type	M/O	Item	Notes
PWM	O	An oscilloscope	It is used to oscilloscope to measure the corresponding pin and check the PWM period and duty cycle.
Peripheral Example	O	• 2inch LCD Module • Dupont Line	-

*M/O: M (Mandatory)/ O (Optional)

2.2 Preparing Software

Before configuring the 40-pin header, the Fedora OS needs to be flashed into the Micro-SD card, and the `dtb` files need to be compiled and replaced. The following procedures are provided:

2.2.1 GitHub Repository

The following table describes the GitHub Repository addresses:

> **Note:**
>
> Make sure you have swtiched to the corresponding branch.

Table 2-2 GitHub Repository addresses

Type	Repository	Branch
Linux	Linux	visionfive
dts File under Linux Repo	• jh7100-common.dtsi • jh7100.dtsi	-
Uboot	Uboot	JH7100_upstream
OpenSBI	OpenSBI	master
Fedora image (Alpha version)	Fedora Image	-

2.2.2 Flashing Fedora OS to Micro-SD Card

Two methods are provided to flash images. One is for Mac/Linux, the other is for Windows. For detailed instructions, refer to Flashing Fedora OS to Micro-SD Card section in VisionFive Single Board Computer Quick Start Guide.

2.2.3 Generating dtb

To compile the device tree sources (`.dtsi` files) into device tree blobs (`.dtb` files) using device tree compiler (DTC), execute the following command under the root directory of Linux:

```
make <Configuration_File> ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu-
make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv dtbs
```

Information:

<Configuration_File>:

Both `starfive_jh7100_fedora_defconfig` and `visionfive_defconfig` are applicable.

The following is the example command:

```
make starfive_jh7100_fedora_defconfig ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu-
make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv dtbs
```

Different boards use different dtb files.

The following table describes the relationship:

Table 2-3 dtb Files

Board	File
StarLight	/linux/arch/riscv/boot/dts/starfive/jh7100-beaglev-starlight.dtb
VisionFive	/linux/arch/riscv/boot/dts/starfive/jh7100-starfive-visionfive-v1.dtb

2.2.4 Replacing dtb

The SD cards used to burn images identify the following directories:

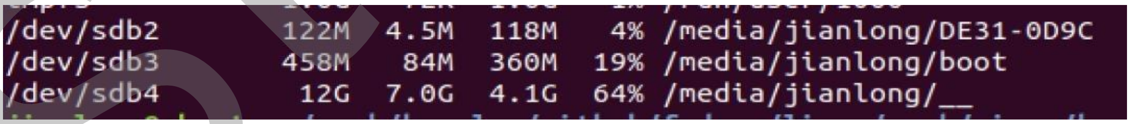


Figure 2-1 Identified Directories

Method 1: Directly Replacing dtb File

Execute the following command under the root directory of Linux to replace the dtb file:

```
sudo cp arch/riscv/boot/dts/starfive/<dtb_file> <Mount_Directory>/__boot/dtbs/<Kernel_Version>/starfive
```

Information:

- <dtb_file> refers to the dtb file name. Different boards use different dtb files. For more information, see *dtb Files* table in this document.

- <Mount_Directory> refers to the actual mount directory. For example, /media/jianlong.
- <Kernel_Version> refers to the kernel version number. For example, 5.14.0+.

Example Command:

```
sudo cp arch/riscv/boot/dts/starfive/jh7100-beaglev-starlight.dtb
/media/jianlong/__boot/dtbs/5.14.0+/starfive
```

Method 2: Adding Startup Item

To replace dtb file by adding startup item, perform the following:

Step 1 Execute the following commands under the root directory of Linux:

```
sudo cp arch/riscv/boot/dts/starfive/<dtb_file> <Mount_Di-
recotry>/boot/dtbs/
```

Information:

- *<dtb_file>* refers to the dtb file name. Different boards use different dtb files. For more information, see *dtb Files* table in this document.
- *<Mount_Directory>* refers to the actual mount directory. For example, /media/jianlong.

Example Command:

```
sudo cp arch/riscv/boot/dts/starfive/jh7100-beaglev-star-
light.dtb /media/jianlong/__boot/dtbs/
```

Step 2 Enter SD card mount directory:

```
cd <Mount_Direcotry>/__boot
```

Information:

<Mount_Directory> refers to the actual mount directory. For example, /media/jianlong.

Step 3 Update grub.cfg:

```
sudo gedit grub.cfg
```

Step 4 Add the following command lines, save and exit:

```
menuentry 'MY Fedora vmlinux-5.14.0+' {

linux /vmlinuz-5.14.0+ ro root=UUID=f852f7f6-aa4e-4404-8ea9-
439568b767a1 rhgb console=tty0 console=ttyS0,115200 earlycon
rootwait stmmaceth=chain_mode:1 selinux=0

LANG=en_US.UTF-8

 devicetree /dtbs/<dtb_File>

initrd /initramfs-<Kernel_Version>.img

}
```

Information:

In these command lines：

- <dtb_File > refers to the name of dtb file used by the board. For example, jh7100-beaglev-starlight.dtb. For the relationship between boards and dtb files, see *dtb Files* table in this document.
- <Kernel_Version> refers to the kernel version number. For example, 5.14.0+.
- MY Fedora vmlinux-5.14.0: Configurable menu item name.

Step 5 Select the menu item set in the previous step, for example, MY Fedora vmlinux-5.14.0+, during startup.

Notes:

Multiple startup items can be added according to the actual number of dtb files.

3 GPIO Operations

This section provides commands to configure GPIO:

3.1 Configuring GPIO

To configure GPIO, perform the following:

Step 1 Execute the following command to configure GPIO0:

```
cd /sys/class/gpio
echo 448 > export
```

Step 2 Locate to the GPIO0 directory:

```
cd gpio448
```

Step 3 Configure the direction of GPIO0 as in:

```
echo in > direction
```

Step 4 Alternatively, configure the direction of GPIO0 as out:

```
echo out > direction
```

Step 5 Configure the voltage level of GPIO0 as high:

```
echo 1 > value
```

You can use an oscilloscope to check the voltage level.

Step 6 Configure the voltage level of GPIO0 as low:

```
echo 0 > value
```

> **Information:**
>
> You can use an oscilloscope to check the voltage level.

Step 7 Connect the **3.3V Power** pin with the GPIO0, and check the voltage level of GPIO0:

```
cat value
```

Step 8 Connect the **GND** pin with the GPIO0, and check the voltage level of GPIO0:

```
cat value
```

4 I2C Operations

This chapter describe how to configure and debug I2C GPIO.

4.1 Configuring I2C GPIO

4 channels of I2C bus are supported: i2c0, i2c1, i2c2, and i2c3.

Perform the following to configure I2C:

4.1.1 Hardware Setup

Connect the Sense Hat (B) to the header as the following:

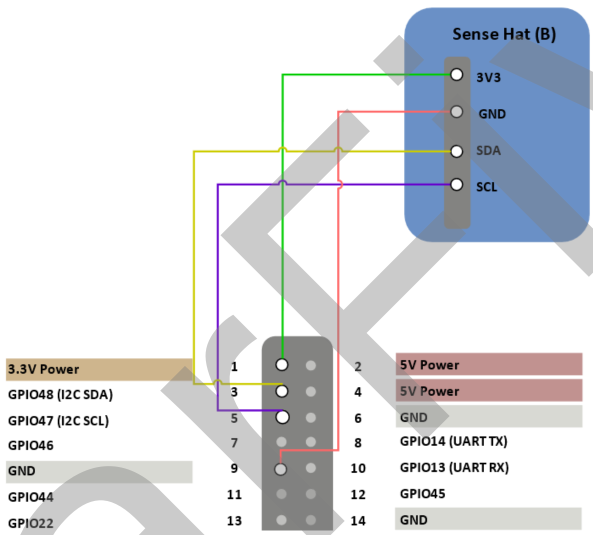


Figure 4-1 Connect the Sense Hat (B) to the header

4.1.2 Configuring dts File

Modify the file content of jh7100-common.dtsi under /linux/arch/riscv/boot/dts/starfive. The following is the default setting. You can configure the unoccupied GPIOs as required.

```
	i2c1_pins: i2c1-0 {
		i2c-pins {
			pinmux = <GPIOMUX(47, GPO_LOW,
				GPO_I2C1_PAD_SCK_OEN,
				GPI_I2C1_PAD_SCK_IN)>,
			       <GPIOMUX(48, GPO_LOW,
				GPO_I2C1_PAD_SDA_OEN,
				GPI_I2C1_PAD_SDA_IN)>;
			bias-pull-up;
			input-enable;
			input-schmitt-enable;
		};
	};
```

Figure 4-2 Example File Content

Information:

The I2C GPIO pin number is the number indicated in the Pin Name. For more details about the GPIO Pin Name, see the *GPIO Pinout* section in this document. The pin names of the I2C GPIO are listed as follows:

- GPIO48 (I2C SDA)
- GPIO47 (I2C SCL)

4.2 Debugging I2C

Perform the following steps to debug I2C:

Step 1 Execute the following command to scan bus:

```
i2cdetect -l
```

Result:

Figure 4-3 Example Output

Step 2 Execute the following command to detect device:

```
i2cdetect -y -r 1
```

Information:

1 is the I2C bus number.

Result:

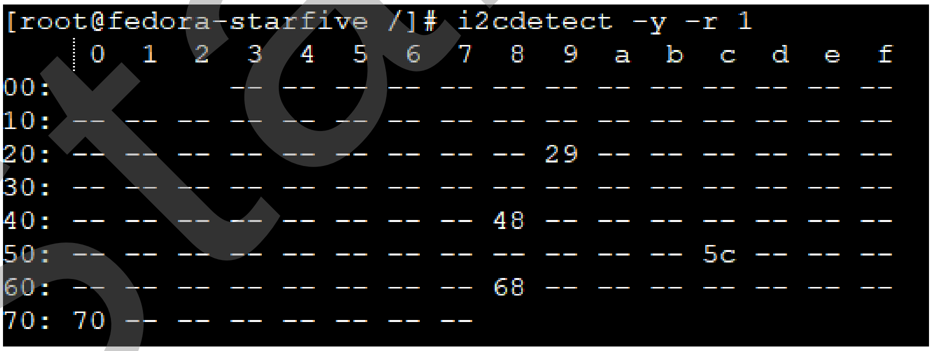


Figure 4-4 Example Output

In this figure, the detected devices are 0x29, 0x48, 0x5c, 0x68, and 0x70.

Step 3 Execute the following command to read register content:

```
i2cget -f -y 1 0x5c 0x0f
```

Information:

- 1: I2C bus number
- 0x5c: I2C device address

- 0x0f: Memory address

Result:

```
# i2cget -f -y 1 0x5c 0x0f
0xb1
#
```

The register content is `0xb1` in this output.

Step 4 Execute the following command to write register data:

```
i2cset -y 1 0x5c 0x11 0x10
```

Information:

- 1: I2C bus number.
- 0x5c: I2C device address.
- 0x11: Memory address.
- 0x10: The content to be written in the register.

Step 5 Execute the following to read all register values:

```
i2cdump -y 1 0x5c
```

Information:

- 1: I2C bus number
- 0x5c: I2C device address

Result:

```
# i2cdump -y 1 0x5c
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 b1    ...............?
10: 00 10 00 00 00 00 00 00 00 00 00 00 01 b1 3f 68    .?..........???h
20: 00 00 00 00 00 00 00 00 13 91 2f 00 00 00 00 00    ........??/.....
30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
40: 7a d4 03 20 86 0f 11 2d 00 06 8e 78 03 10 0b 48    z?? ???-.??x???H
50: 32 fb 6b 92 01 13 91 2f 06 03 14 08 b7 04 80 c0    2?k????/????????
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 b1    ...............?
90: 00 10 00 00 00 00 00 00 00 00 00 00 01 b1 3f 68    .?..........???h
a0: 00 00 00 00 00 00 00 00 13 91 2f 00 00 00 00 00    ........??/.....
b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
c0: 7a d4 03 20 86 0f 11 2d 00 06 8e 78 03 10 0b 48    z?? ???-.??x???H
d0: 32 fb 6b 92 01 13 91 2f 06 03 14 08 b7 04 80 c0    2?k????/????????
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
#
```

Figure 4-5 Example Output

5 SPI Operations

This chapter describe how to configure and debug SPI GPIO.

5.1 Configuring SPI GPIO

The configuration file, `jh7100-common.dtsi`, is under:

`/linux/arch/riscv/boot/dts/starfive`.

2 channels of SPI bus are supported: spi2 and spi3.

Modify Pins

The configured SPI GPIO number is the number indicated in the Pin Name. For more details about the GPIO Pin Name, see the *GPIO Pinout* section in this document. You can configure the unoccupied pins. The following is the default settings in the `jh7100-common.dtsi:`

```
spi2_pins: spi2-0 {
	mosi-pin {
		pinmux = <GPIOMUX(18, GPO_SPI2_PAD_TXD,
			  GPO_ENABLE, GPI_NONE)>;
		bias-disable;
		input-disable;
		input-schmitt-disable;
	};
	miso-pin {
		pinmux = <GPIOMUX(16, GPO_LOW, GPO_DISABLE,
			  GPI_SPI2_PAD_RXD)>;
		bias-pull-up;
		input-enable;
		input-schmitt-enable;
	};
	sck-pin {
		pinmux = <GPIOMUX(12, GPO_SPI2_PAD_SCK_OUT,
			  GPO_ENABLE, GPI_NONE)>;
		bias-disable;
		input-disable;
		input-schmitt-disable;
	};
	ss-pins {
		pinmux = <GPIOMUX(15, GPO_SPI2_PAD_SS_0_N,
			  GPO_ENABLE, GPI_NONE)>,
			 <GPIOMUX(11, GPO_SPI2_PAD_SS_1_N,
			  GPO_ENABLE, GPI_NONE)>;
		bias-disable;
		input-disable;
		input-schmitt-disable;
	};
};
```

Figure 5-1 Modify Pins

5.2 Debugging SPI GPIO

This section provides steps for loopback test and testing SPI with ADXL345 module.

5.2.1 Loopback Test

The following steps are provided for loopback test:

Step 1 Wiring: Connect pin 18 and 16 as the following:

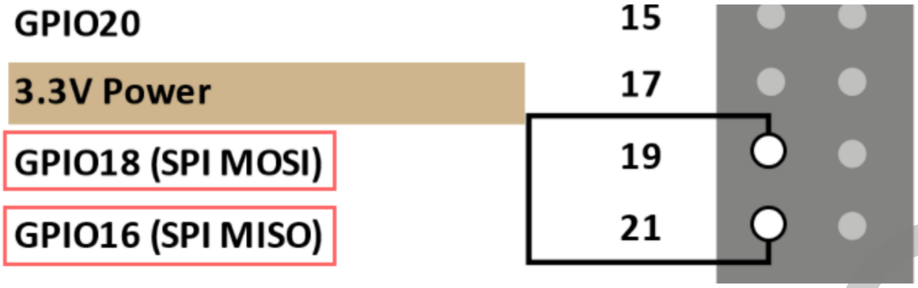


Figure 5-2 Connect Pin 18 and 16

Step 2 Locate to the following path for test tool, `spidev_test.c`:

```
cd /linux/tools/spi
```

Step 3 Execute the following command under the test tool directory:

```
make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv
```

Result:

The output file is spidev_test in the same directory.

Step 4 Upload `spidev_test` to the board, for example, Starlight, and change the execution permission by executing the following:

```
chmod +x spidev_test
```

Step 5 Confirm the SPI device.

```
ls /dev/spidev*
```

Example:

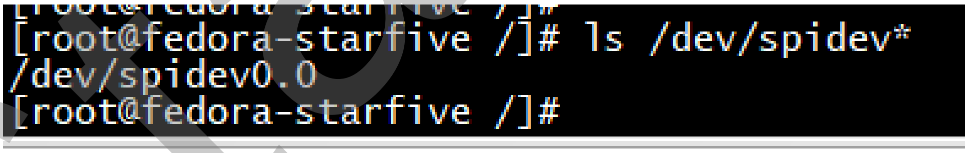


Figure 5-3 Example Output

In this output, spidev0.0 is the device name.

Step 6 Execute the following command to perform the test:

> Notes:
>
> spidev0.0 is the device name got from the previous step.

```
./spidev_test -D /dev/spidev0.0 -v -p string_to_send
```

Result:

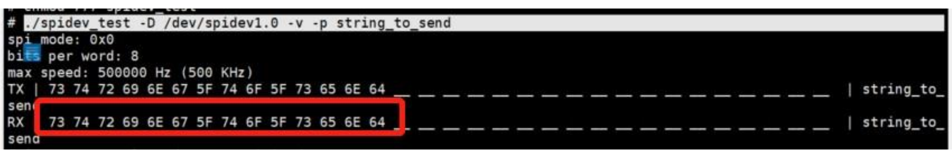


Figure 5-4 Example Output

In this figure, the highlighted part indicates the test is successful.

5.2.2 Testing SPI with ADXL345 Module

Perform the following steps to test SPI with ADXL345 module:

Step 1 Connect the ADXL345 module to the 40-pin header as the following:

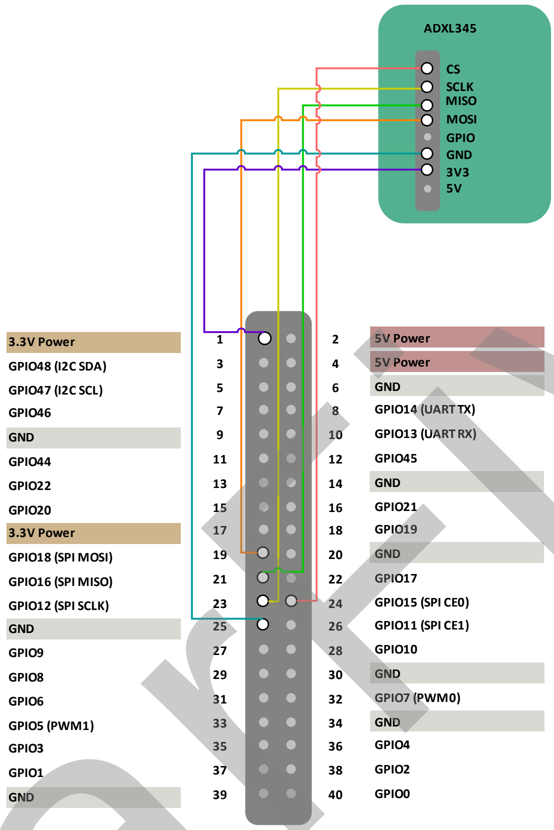


Figure 5-5 Connect ADXL345 Module to the Header

Step 2 Locate to the following path for test tool, `spidev_test.c`:

```
cd /linux/tools/spi
```

Step 3 Execute the following command under the test tool directory:

```
make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv
```

Result:

The output file is spidev_test in the same directory.

Step 4 Upload `spidev_test` to the board, for example, StarLight, and change the execution permission by executing the following:

```
chmod +x spidev_test
```

Step 5 Confirm the SPI device.

```
ls /dev/spidev*
```

Example:

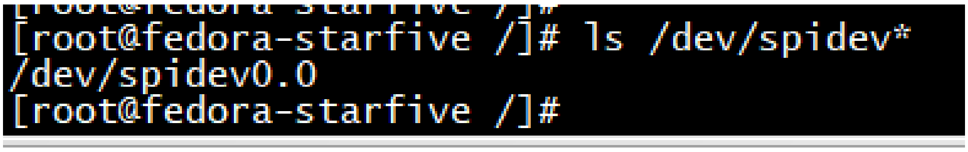


Figure 5-6 Example Output

In this output, spidev0.0 is the device name.

Step 6 Execute the following to read the device ID:

```
./spidev_test -H -O -D /dev/spidev0.0 -v -p \\x80\\x00
```

Step 7 Execute the following to read the value for multiple registers:

```
./spidev_test -H -O -D /dev/spidev0.0 -v -p
\\xec\\x00\\x00\\x00\\x00\\x00\\x00
```

Step 8 Execute the following to read:

```
./spidev_test -H -O -D /dev/spidev0.0 -v -p \\x9e\\x00
```

Step 9 Execute the following to write:

```
./spidev_test -H -O -D /dev/spidev0.0 -v -p \\x1e\\xaa
```

Step 10 Execute the following to read the verification:

```
./spidev_test -H -O -D /dev/spidev0.0 -v -p \\x9e\\x00
```

6 PWM Operation

This chapter describes how to configure and debug PWM GPIO:

6.1 Configuring PWM GPIO

The configuration file, `jh7100-common.dtsi`, is located under:

`/linux/arch/riscv/boot/dts/starfive`

8 channels of PWM are supported at the most.

Modify Pin

The following figure shows the example file content to modify pin:

```
    pwm_pins: pwm_pins-0 {
        ptc-pins {
            pinmux = <GPIOMUX(7, GPO_PWM_PAD_OUT_BIT0, GPO_PWM_PAD_OE_N_BIT0, GPI_NONE)>,
                     <GPIOMUX(5, GPO_PWM_PAD_OUT_BIT1, GPO_PWM_PAD_OE_N_BIT1, GPI_NONE)>,
                     <GPIOMUX(45, GPO_PWM_PAD_OUT_BIT2, GPO_PWM_PAD_OE_N_BIT2, GPI_NONE)>;
            bias-disable;
            drive-strength = <35>;
            input-disable;
            input-schmitt-disable;
            slew-rate = <0>;
        };
    };
};
```

Figure 6-1 Example File Content

The configured PWM GPIO number is the number contained in the **Pin Name**. For more details about the GPIO Pin Name, see the *GPIO Pinout* section in this document.

PWM and Pin Name Mapping

The following table describes the PWM and pin name mapping:

Table 6-1 PWM and Pin Name Mapping

PWM	GPIO (Pin Name)
PWM0	GPIO7
PWM1	GPIO5
PWM2	GPIO45

6.2 Debugging PWM GPIO

This section describes how to debug PWM GPIO:

Step 1 Execute the following to configure PWM lane:

```
cd /sys/class/pwm/pwmchip0
```

```
echo 0 > export
```

Step 2 Execute the following to configure PWM period:

```
cd pwm0
echo 5000000 > period
```

Step 3 Execute the following to configure PWM duty cycle:

```
echo 1000000 > duty_cycle
```

Step 4 Use an oscilloscope to measure the corresponding pin and check the PWM period and duty cycle.

7 UART Operations

This chapter describes how to configure and debug UART GPIO:

7.1 Configuring UART GPIO

The configuration file, jh7100-common.dtsi, is located under:

`/linux/arch/riscv/boot/dts/starfive`

4 channels of UART are supported at the most:

- Uart3 is for Debug.
- Uart0 is for Bluetooth.
- Uart1 and Uart2 can be used.

The configured UART GPIO number is the number contained in the Pin Name. For more details about the GPIO Pin Name, see the *GPIO Pinout* section in this document.

7.1.1 Modifying dts

To modify dts file, perform the following steps:

Step 1 Add aliases of uart1 or uart2 on the aliases node. The following is an example:

```
/ {
        aliases {
                mshc0 = &sdio0;
                mshc1 = &sdio1;
                serial0 = &uart3;
                serial1 = &uart0;
                serial2 = &uart1;
                serial3 = &uart2;
        };

        chosen {
                stdout-path = "serial0:115200n8";
        };

        cpus {
                timebase-frequency = <6250000>;
        };
```

Figure 7-1 Example Configuration

Step 2 Add uart1 or uart2 node on the dts. The following is an example:

```
&uart3 {
        pinctrl-names = "default";
        pinctrl-0 = <&uart3_pins>;
        status = "okay";
};

&uart1 {
        pinctrl-names = "default";
        pinctrl-0 = <&uart1_pins>;
        status = "okay";
};

&uart2 {
        pinctrl-names = "default";
        pinctrl-0 = <&uart2_pins>;
        status = "okay";
};
```

Figure 7-2 Example Configuration

Step 3 Add uart1_pins or uart2_pins node on the &gpio node

The configured UART GPIO number is the number contained in the **Pin Name**. You can configure the unoccupied pins. For more details about the GPIO Pin Name, see the *GPIO Pinout* section in this document.

```
                };
        };

        uart1_pins: uart1-0 {
                rx-pins {
                        pinmux = <GPIOMUX(1, GPO_LOW, GPO_DISABLE,
                                  GPI_UART1_PAD_SIN)>;
                        bias-pull-up;
                        drive-strength = <14>;
                        input-enable;
                        input-schmitt-enable;
                };
                tx-pins {
                        pinmux = <GPIOMUX(3, GPO_UART1_PAD_SOUT,
                                  GPO_ENABLE, GPI_NONE)>;
                        bias-disable;
                        drive-strength = <35>;
                        input-disable;
                        input-schmitt-disable;
                };
        };

        uart2_pins: uart2-0 {
                rx-pins {
                        pinmux = <GPIOMUX(0, GPO_LOW, GPO_DISABLE,
                                  GPI_UART2_PAD_SIN)>;
                        bias-pull-up;
                        drive-strength = <14>;
                        input-enable;
                        input-schmitt-enable;
                };
                tx-pins {
                        pinmux = <GPIOMUX(2, GPO_UART2_PAD_SOUT,
                                  GPO_ENABLE, GPI_NONE)>;
                        bias-disable;
                        drive-strength = <35>;
                        input-disable;
                        input-schmitt-disable;
                };
        };
};

&i2c0 {
        clock-frequency = <100000>;
```

Figure 7-3 Example Configuration

UART and DEV Mapping

The following table describes the UART and DEV mapping:

Table 7-1 UART and DEV Mapping

UART	DEV
UART1	/dev/ttyS2
UART2	/dev/ttyS3

7.2 Debugging UART GPIO

7.2.1 Hardware Setup

To set up the hardware, perform the following steps:

Steps:

Step 1 Connect the jumper wires from the USB-to-Serial Converter to the 40-Pin GPIO header of the VisionFive as follows.

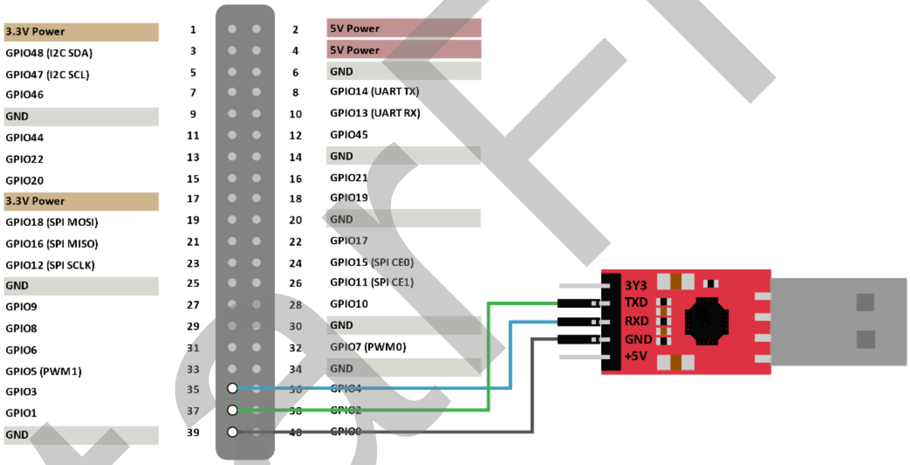


Figure 7-4 Connect the Converter to the Header

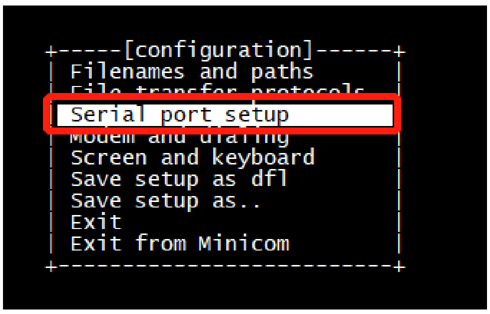
Step 2 Connect the other end of the USB-to-Serial Converter to your device (PC/Ubuntu).

7.2.2 Debugging UART Send and Receive Functions

Step 1 Configure Visionfive minicom

```
sudo minicom -s
```

Step 2 Select **Serial port setup**, and configure minicom as follows:



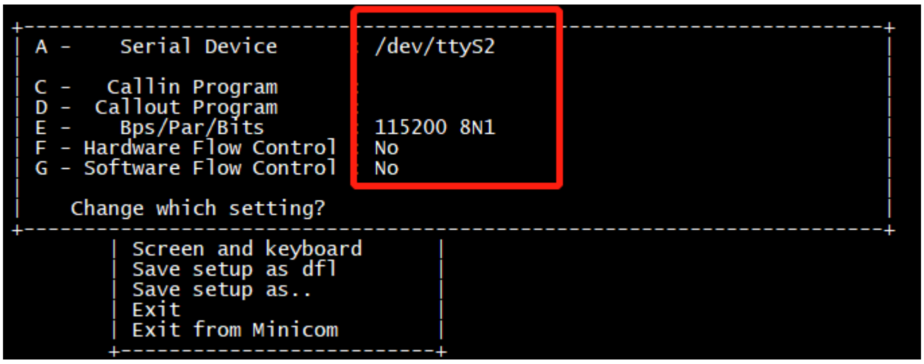


Figure 7-5 Example Configuration

Step 3 Start Visionfive minicom by typing the following command:

```
minicom -o -D /dev/ttyS2
```

Result:

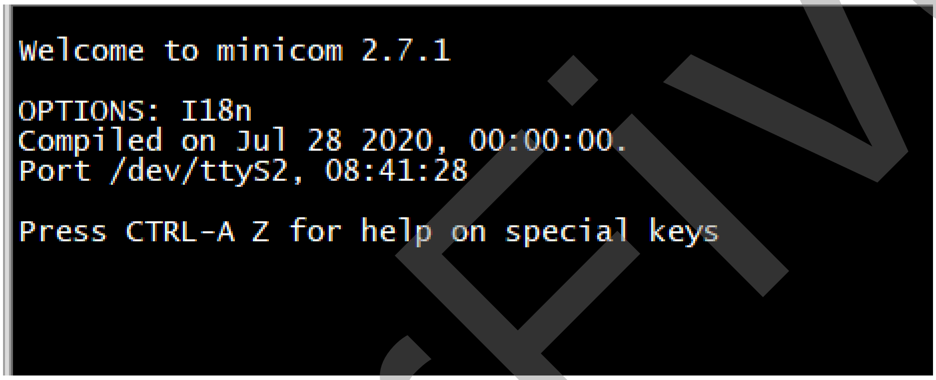


Figure 7-6 Example Output

Step 4 Configure Ubuntu minicom by typing the following:

```
sudo minicom -s
```

Step 5 Select Serial port setup, and configure minicom as follows:

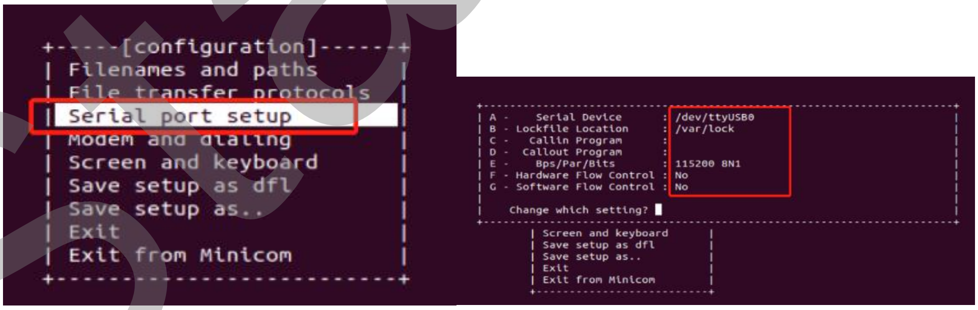


Figure 7-7 Example Configuration

Information:

Serial Device can be detected by command `dmesg | grep tty` on Ubuntu

Start Ubuntu minicom, you can see as follows:

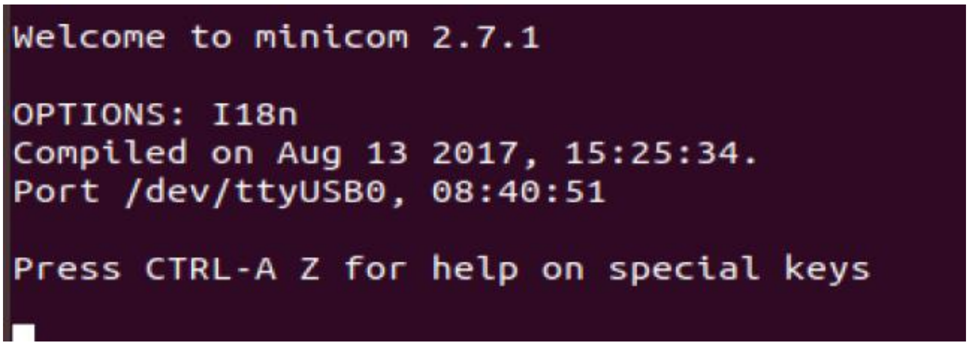
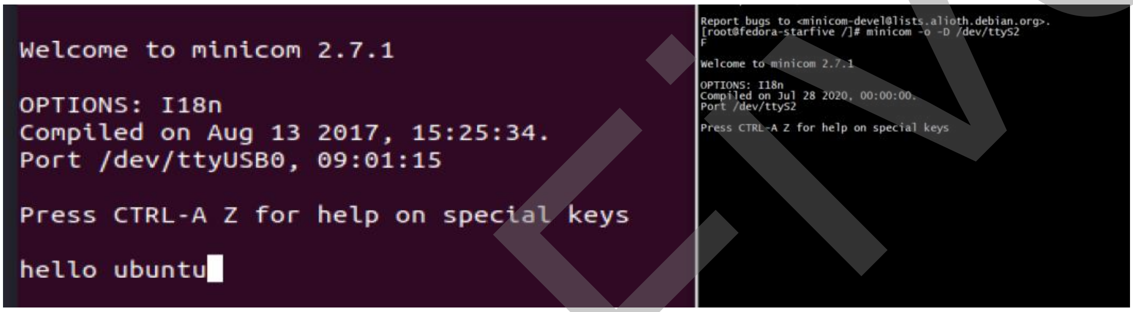


Figure 7-8 Example Output

Test UART Send:

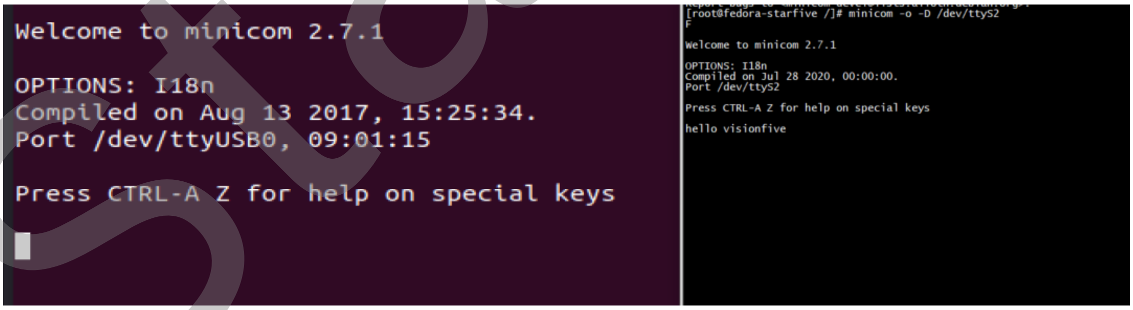
To test UART send function, you can input characters, such as **hello ubuntu**, on the VisionFive minicom. Then you wil see the character are outputted on the Ubuntu minicom as the following:



- Figure on the Left: Ubuntu minicom interface
- Figure on the Right: VisionFive minicom interface

Test UART Receive:

To test UART receive, you can input characters, such as hello visionfive on the Ubuntu minicom. Then you will see the characters are outputted on the Visionfive minicom:



- Figure on the Left: Ubuntu minicom interface
- Figure on the Right: VisionFive minicom interface

8 Peripheral Examples

In this demo, Sense Hat (B) is used. For the detailed specifications, refer to https://www.waveshare.com/wiki/Sense_HAT_(B) .

Notes:

The official libraries of BCM2835, Python, and wiringPi are not supported, and we use the system call instead. The examples are required to be modified.

8.1 Sense Hat (B) Example

8.1.1 Hardware Setup

The following table and figure describe how to connect Sense HAT to the 40-pin header:

Table 8-1 Connect Sense Hat (B) and the 40-Pin Header

Sense HAT (B)	Pin Number
3V3	1
GND	9
SDA	3
SCL	5

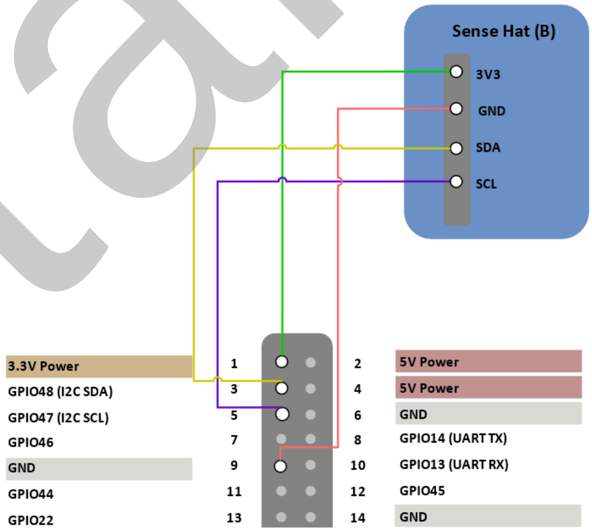


Figure 8-1 Connect Sense Hat (B) and the 40-Pin Header

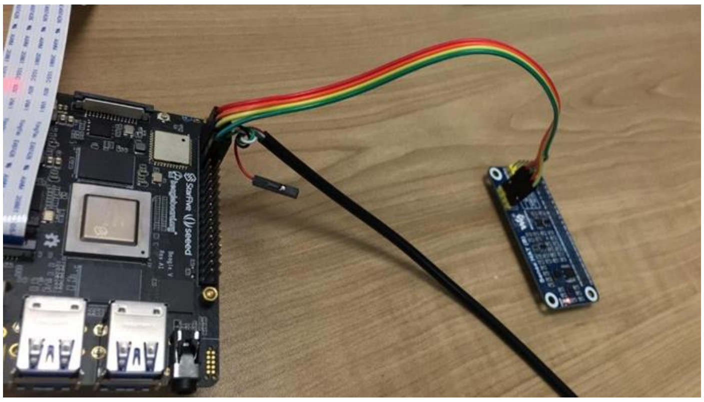


Figure 8-2 Connect Sense Hat (B) and the 40-Pin Header

8.1.2 Examples

Take SHTC3 sensor as an example:

Step 1 Download the source code from: SHTC3_dev.c

Step 2 (Optional) Install the tool to compile. The following is an example to install:

```
sudo apt-get install gcc-riscv64-linux-gnu
```

Notes:

This step can be skipped if the tool has been installed.

Step 3 Execute the following to compile:

```
riscv64-linux-gnu-gcc SHTC3_dev.c -o shtc3
```

Result:

The output file is shtc3 in the same directory.

Step 4 Copy the executable codes from the `shtc3` file to the board, and change the execution permission by execute the following command:

```
chmod +x shtc3
```

Step 5 Execute the following command to run:

```
./shtc3
```

Result:

The following output indicates the execution is successful:

```
[root@fedora-starfive riscv]# ./shtc3
SHTC3 Sensor Test Program ...
Fopen : /dev/i2c-1
Temperature =   75.61°C , Humidity =  68.55
Temperature =   27.40°C , Humidity =  68.54
Temperature =   27.40°C , Humidity =  68.55
Temperature =   27.40°C , Humidity =  68.54
```

```
Temperature =   27.39°C , Humidity =  68.54
```

8.2 2inch LCD Module Example

2inch LCD Module is used in this example. For the detailed specifications, refer to the following: https://www.waveshare.com/wiki/2inch_LCD_Module.

Notes:

The official examples are required to be modified for this demo.

8.2.1 Hardware Setup

The following table and figure describe how to connect the 2inch LCD module and the 40-pin header:

Table 8-2 Connect 2inch LCD and 40-pin Header

2 2inch LCD Module	Pin Number
VCC	17
GND	39
DIN	19
CLK	23
CS	28
DC	22
RST	13
BL	18

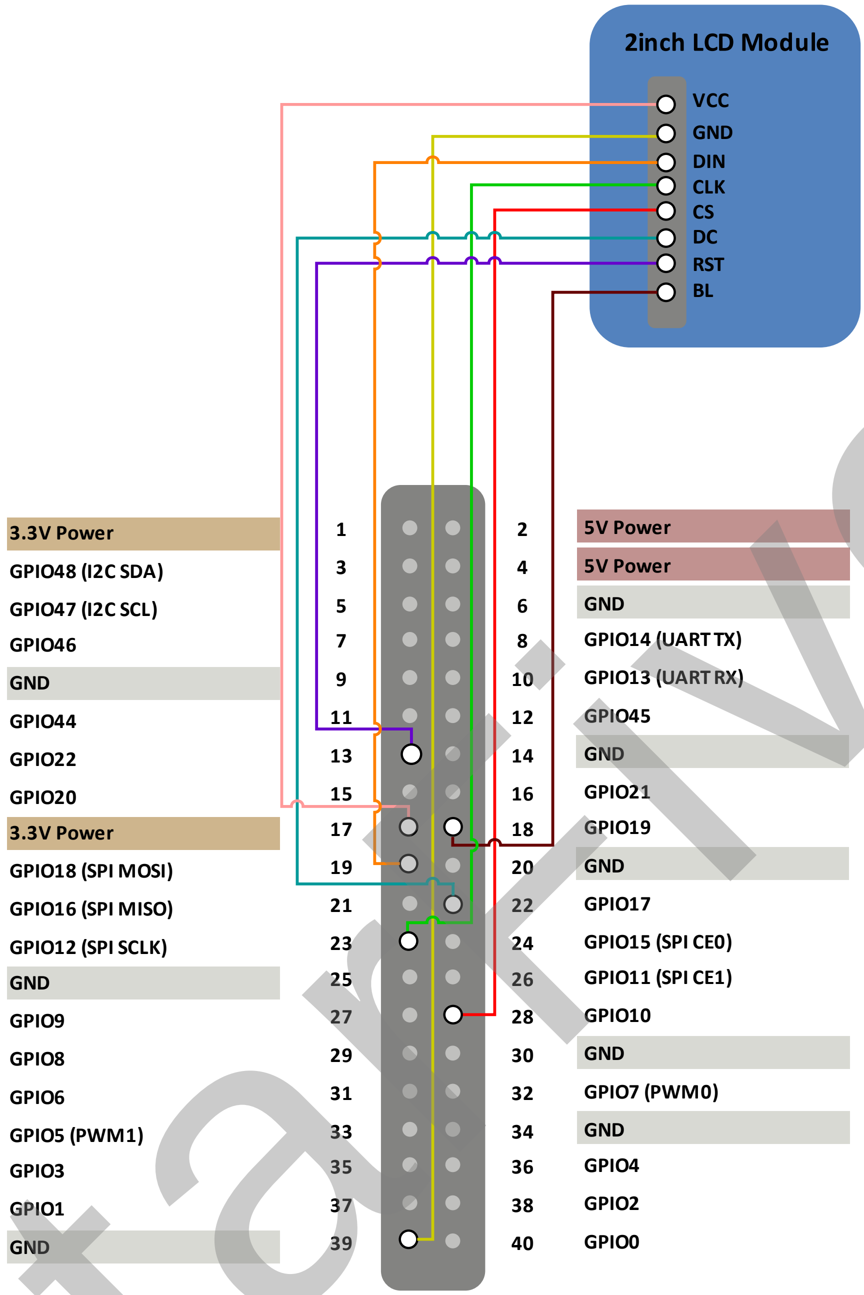


Figure 8-3 Connect 2inch LCD and 40-pin Header

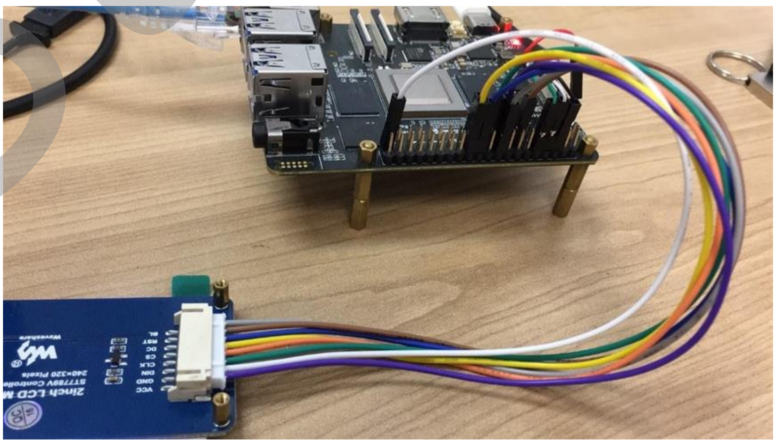


Figure 8-4 Connect 2inch LCD and 40-pin Header

8.2.2 Executing Example

Perform the following steps to execute the example:

Step 1 Download the source code from starlight-0916.tar.gz.

Step 2 Execute the following command to copy the codes to the board. For example, StarLight.

```
tar -xvf starlight-0916.tar.gz
cd starlight/
./main 2
```