Open Embed
Embedded·IoT·Edge

EdgeBox-RPI5 User Manual V1.0

Raspberry PI CM5 Based Edge computer

Revision History

Revision	Date	Changes
1.0	24-10-2025	Created
1.1	18-06-2026	Add software sections

Contents

1. Introduction

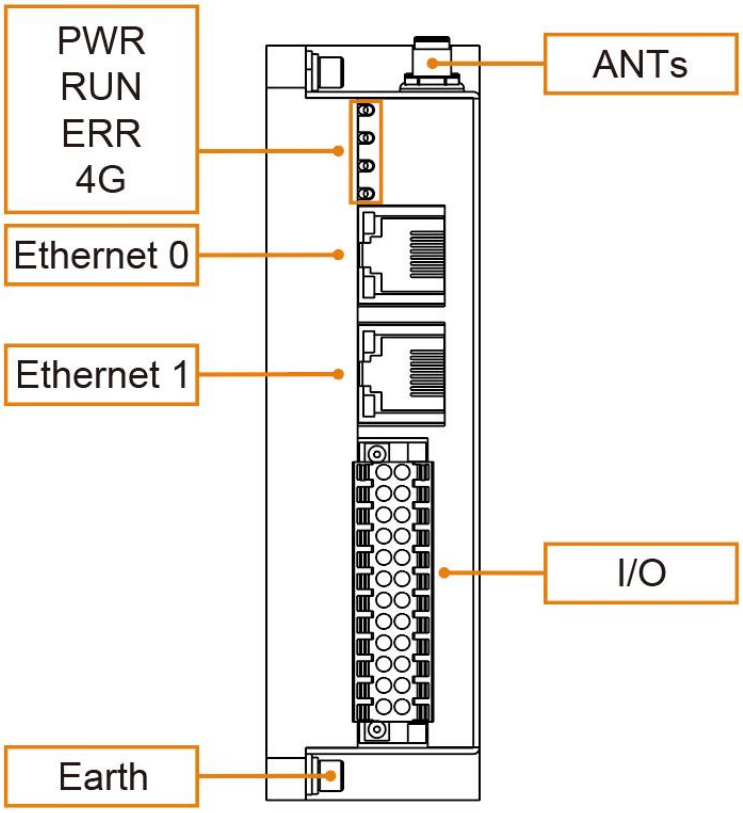
EdgeBox-RPI5 is a rugged fan-less Edge Computing Controller with Raspberry Pi Computer Module 5(CM5) for harsh industry environment.It can be used to connect the field networks with cloud or IoT applications. It is designed from the ground up to meet the challenges of rugged applications at competitive prices,ideal for small business or small order with scale milti-level demands.

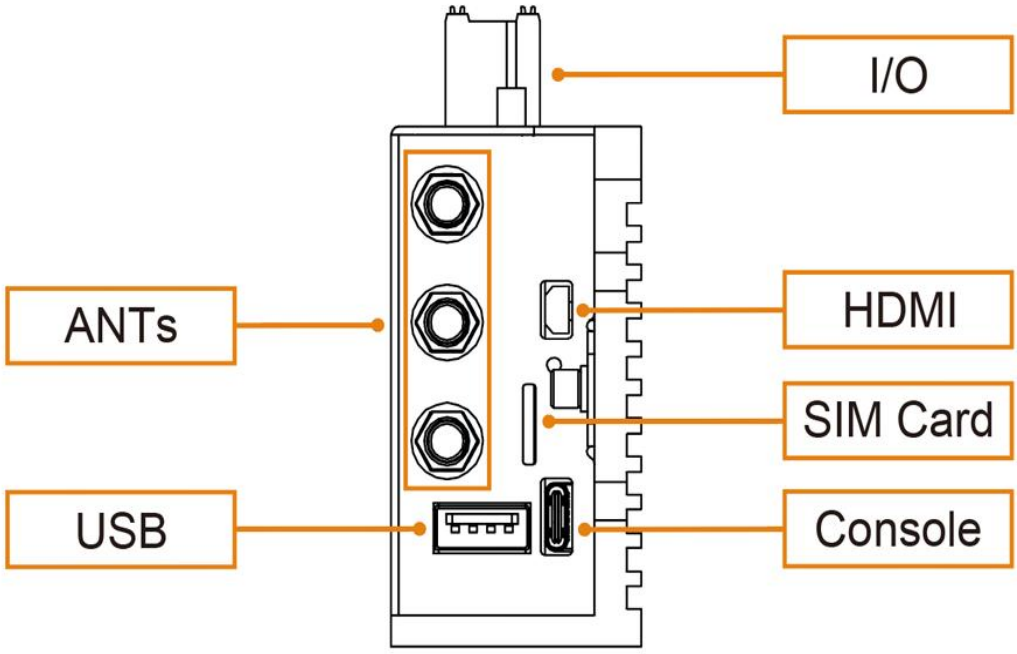
1.1 Features

- State-of-the-art Aluminium chassis for Harsh environment
- Integrated passive heat sink
- Built-in M.2 socket for RF module, such as 4G, WI-FI, Lora or Zigbee
- SMA antenna holes x3
- Encryption chip ATECC608B
- Hardware Watchdog
- RTC with Super Capacitor
- Isolated DI&DO terminal
- 35mm DIN Rail support
- Wide power supply from 9 to 36V DC

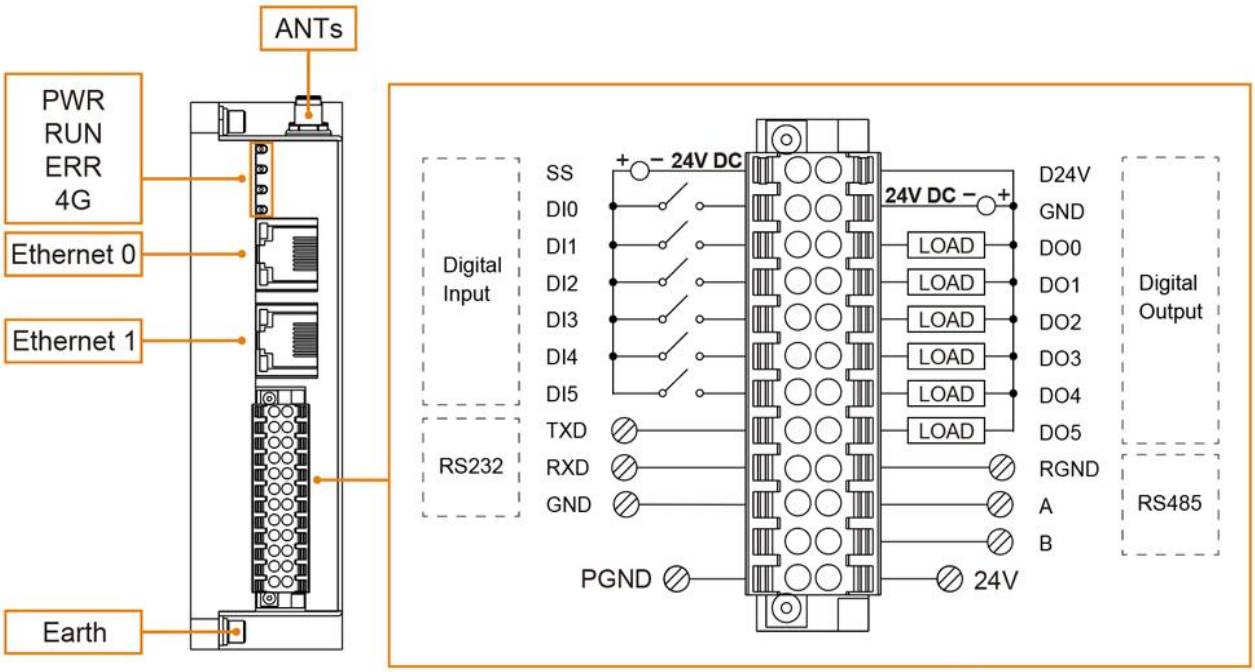
These features make the EdgeBox-RPI5 designed for easy setup and quick deployment for typical industrial applications, such as status monitoring, facility management,digital signage and remote control of public utilities.Furthermore, it is a user-friendly gateway solution with 4 cores ARM Cortex A76 and most industry protocols can save on total deployment costs including electrical power cabling cost and help reduce the product’s deployment time. Its ultra-lightweight and compact design is the answer for applications in space-constricting environments ensures it can operate reliably in a variety of extreme environments including in-vehicle applications.

1.2 Interfaces





Multi-Func phoenix connector



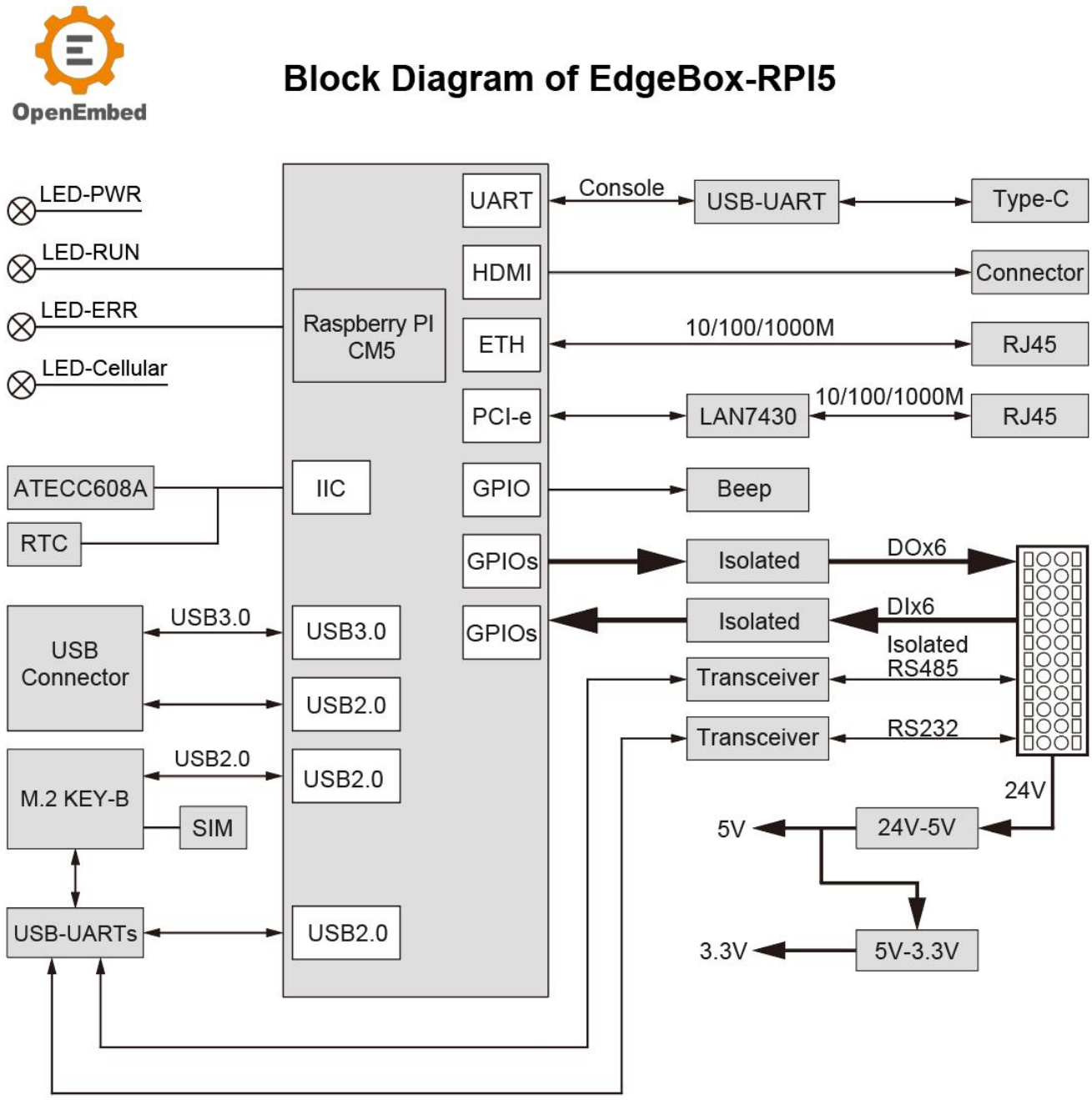
Note	Func name	PIN #	PIN#	Func name	Note
	SS	1	2	D24V	
	DI0	3	4	GND	GND only for D24V
	DI1	5	6	DO0	
	DI2	7	8	DO1	
	DI3	9	10	DO2	
	DI4	11	12	DO3	
	DI5	13	14	DO4	
	TXD	15	16	DO5	
	RXD	17	18	RGND	GND only for RS485
	GND	19	20	RS485_A	
		21	22	RS485_B	
Main GND	PGND	23	24	24V	Main power supply

NOTE: 24awg to 16awg cable are suggested

NOTE: All DOx(x=0...5)signals are wet contact

1.3 Block Diagram

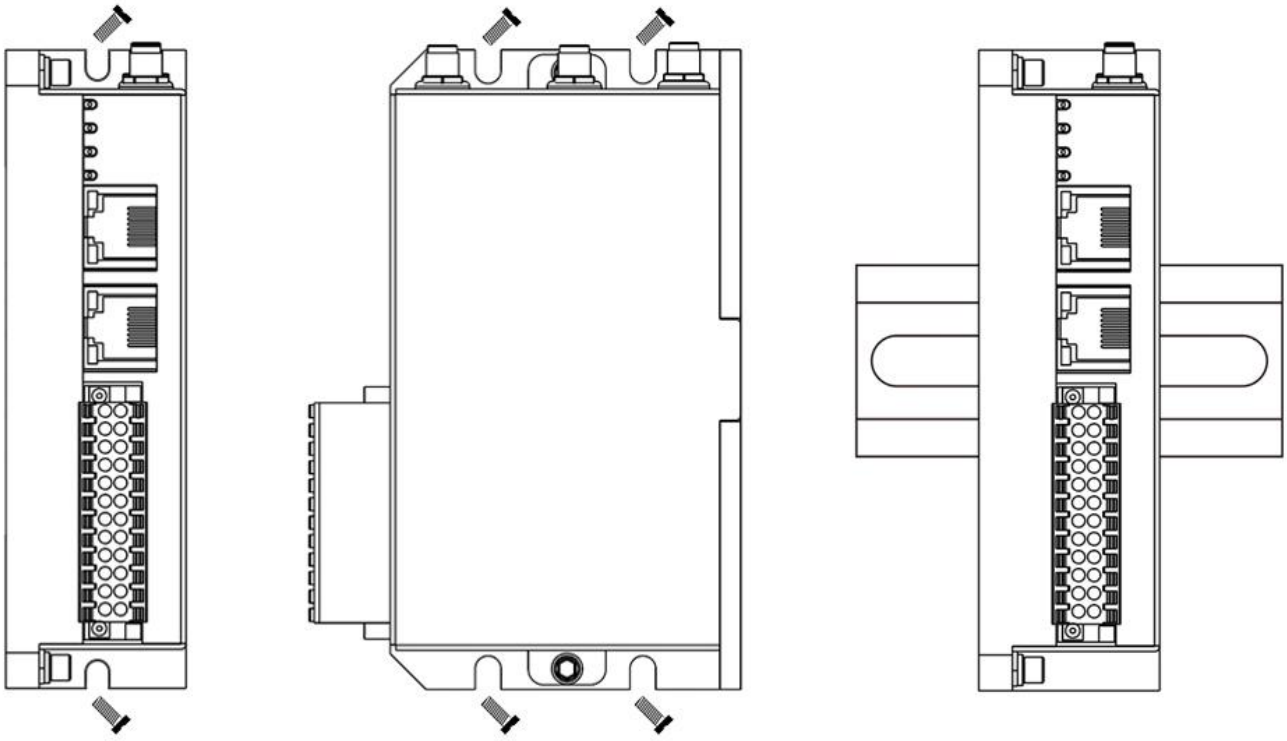
The processing core of the EdgeBox-RPI5 is a Raspberry CM5 board. An OpenEmbed specific base board implements the specific features. Refer to next figure for the block diagram.



2. Installation

2.1 Mounting

The EdgeBox-RPI5 is intended for two wall mounts, as well one with 35mm DIN-rail . Refer to next figure for the recommended mounting orientation.



2.2 Connectors and Interfaces

2.2.1 Power supply

Pin#	Signal	Description
1	24V	Mian power supply DC 9-36V
2	PGND	Ground (Reference potential)

The PE signal is optional. If there is no EMI present ,the PE connection can left open.

2.2.2 Serial Port (RS232 and RS485)

Pin#	Signal	Description
15	TXD	RS232 transmit line
17	RXD	RS232 receive line
19	GND	Ground (Reference potential)

Pin#	Signal	Description
20	RS485_A	RS485 difference line high
22	RS485_B	RS485 difference line low
18	RGND	RS485 Ground (isolated from GND)

The RGND signal is isolated with “GND” signal. If a shielded twisted pair wire is used ,the RGND is connected to the shield.

NOTE: The 120 Ohm termination resistor for RS485 has been installed inside.

2.2.3 DI&DO

Pin#	Signal of terminal	PIN Level of active	PIN of GPIO from BCM2712	NOTE
3	DI0	HIGH	GPIO7	
5	DI1	HIGH	GPIO12	
7	DI2	HIGH	GPIO13	
9	DI3	HIGH	GPIO14	
11	DI4	HIGH	GPIO15	
13	DI5	HIGH	GPIO22	

Pin#	Signal of terminal	PIN Level of active	PIN of GPIO from BCM2712	NOTE
6	DO0	HIGH	GPIO17	
8	DO1	HIGH	GPIO27	
10	DO2	HIGH	GPIO23	
12	DO3	HIGH	GPIO24	
14	DO4	HIGH	GPIO25	
16	DO5	HIGH	GPIO26	

NOTE:

1. DC voltage for input is 24V(+- 10%).
2. DC voltage for output should be under 60V ,the current capacity is 500ma.

2.2.4 HDMI

A micro HDMI connector is directly connected to the Raspberry PI CM5 board with TVS array.

2.2.5 Ethernet

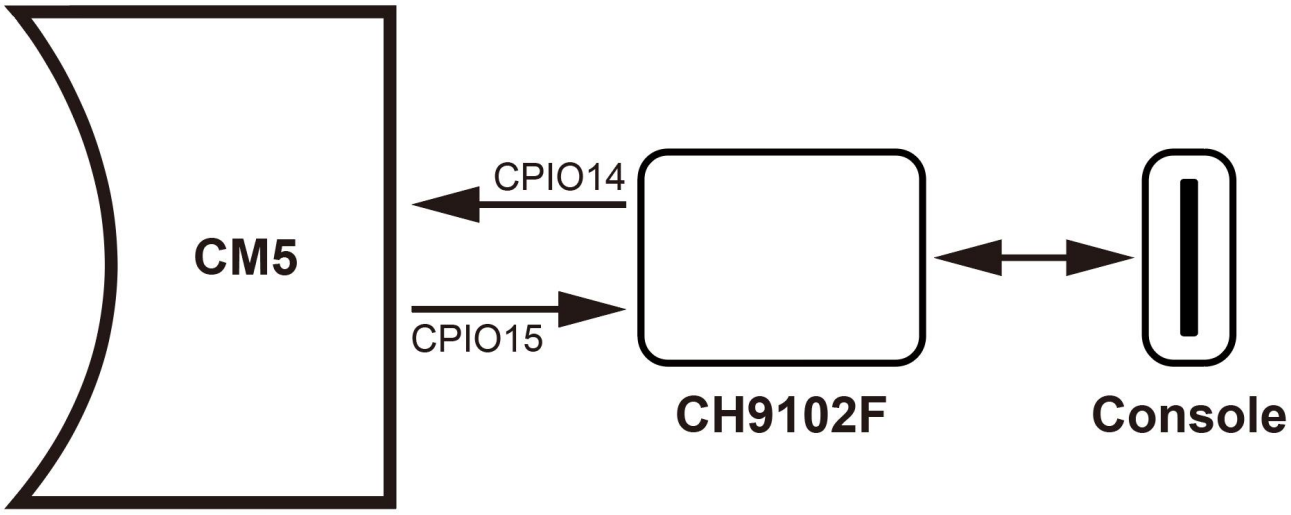
The Ethernet 0 interface is directly comes from Raspberry PI CM5,10/100/1000-BaseT supported, available through the shielded modular jack. Twisted pair cable or shielded twisted pair cable can be used to connect to this port. And Ethernet 1 port is converted from the PCIe signals of CM5.

2.2.6 USB HOST

There is an USB3.0 interface at the connector panel and an electronic fuse is used to connected between the 5.0V and main 5.0V main power.

NOTE: Max current for both ports is limited to 1000ma.

2.2.7 Console(USB typeC)



The design of console used a USB-UART converter, most OS of the computer have the driver.This port is used as a Linux console default.You can log into the OS use the settings of 115200,8n1(Bits: 8, Parity: None, Stop Bits: 1, Flow Control: None).A terminal program such as putty is needed,too.

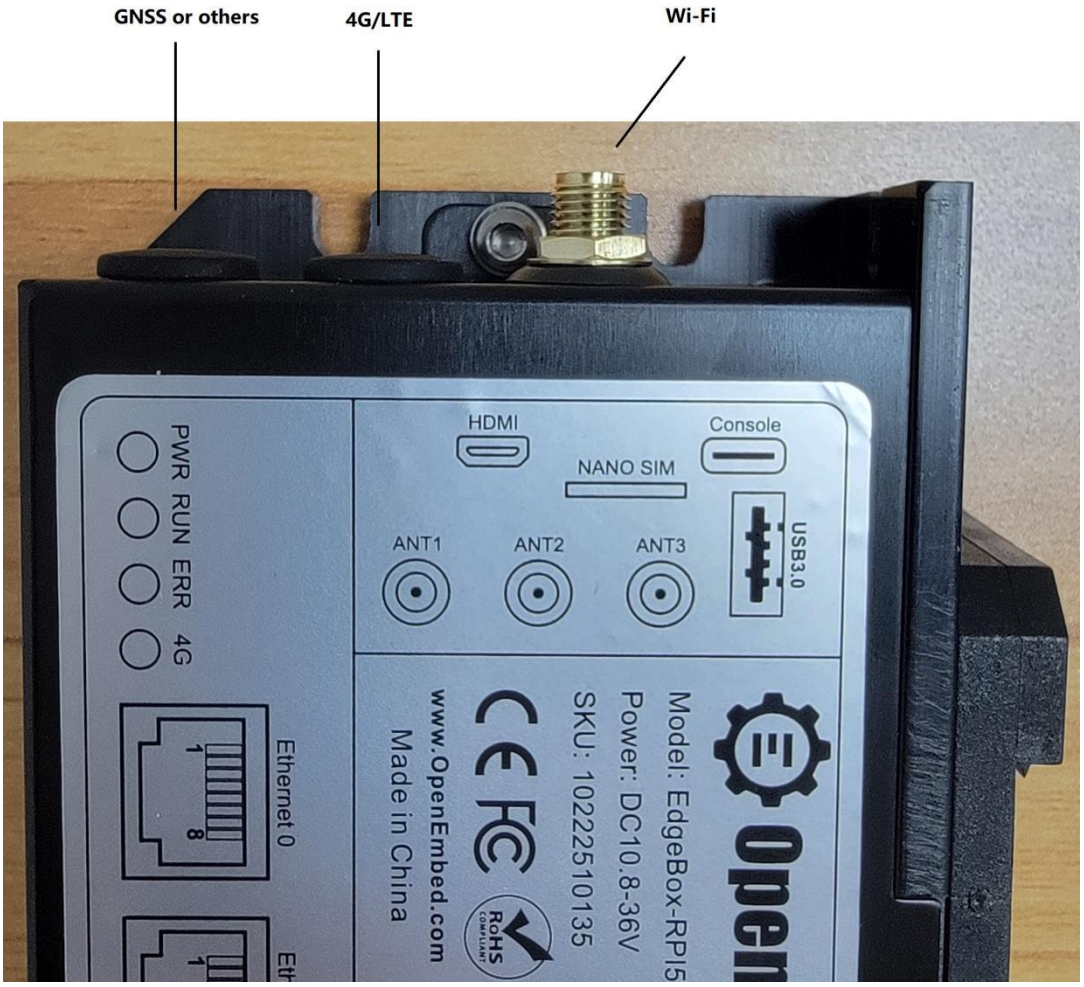
NOTE: The default username is pi and password is raspberry.

2.2.8 LED

EdgeBox-RPI5 use four green/red colour LED as outside indicators.

2.2.9 SMA Connector

There are three SMA Connector holes for antennas. The antenna types are very depend on what modules fitted into the M.2 socket. The ANT in the right is default used for for Internal WI-FI signal from CM5 module, and the middle mostly used for 4G/LTE module from the M.2 socket . Gnss/GPS ANT. is also a option in the left.



NOTES:

1. The functions of the antennas are not fixed,maybe adjusted to cover other usage.

2.2.10 NANO SIM card slot

The sim card is only needed in cellular(4G,LTE or others based on cellular technology) mode.

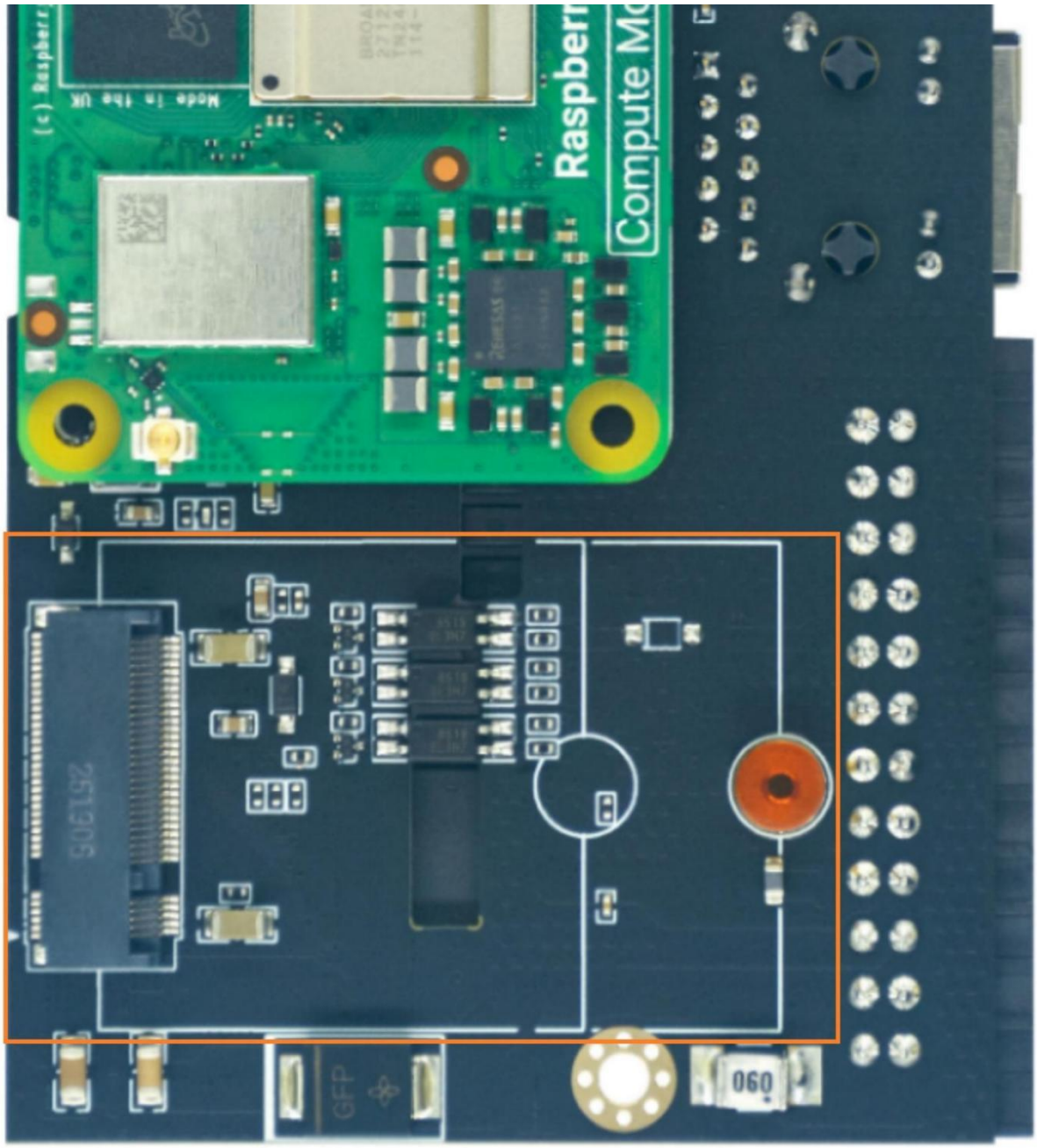
NOTES:

1.Only NANO Sim card is accepted,pay attention to the card size.

2.The NANO sim card is inserted with chip side top.

2.2.11 M.2

The orange area is the rough m.2 add-on card position,only one m2x5 screw is needed.



The table below show all the signals of normal M.2 card.

Signals of M.2 slot:

Signal	PIN#	PIN#	Signal
	1	2	PWR
GND	3	4	PWR
GND	5	6	
USB_DP	7	8	
USB_DM	9	10	LED#
GND	11	12	
	21	22	
	23	24	
	25	26	GND
GND	27	28	
	29	30	USIM_RESET
	31	32	USIM_CLK
GND	33	34	USIM_DATA
	35	36	USIM_PWR
	37	38	
GND	39	40	GND
	41	42	SPI0_SCK
	43	44	SPI0_MISO
GND	45	46	SPI0_MOSI
	47	48	SPI0_CS#
	49	50	GND
GND	51	52	
	53	54	
	55	56	
GND	57	58	
	59	60	
	61	62	UART_PCIE_TXD
	63	64	UART_PCIE_TXD
	65	66	
PERST#	67	68	
	69	70	PWR
GND	71	72	PWR
GND	73	74	PWR
	75		

NOTE 1: All blank signals are NC(no connect).
NOTE 2: 3042 size, B KEY card is supported .
NOTE 3: PWR is the individual power supply for M.2 card .

2.3 GPIO Multiplex

Overview of the GPIO usage from Edgelogix1145,most of the GPIO have the fixed function as list.

Name	IO of CM5	Type	Function
RUN_LED	GPIO 0	Output, Low active	Green LED of LED RUN
M2_RST	GPIO 1	Output, High active	Reset for M.2 card
I2C_SDA	GPIO 2		
I2C_SCL	GPIO 3		
ERR_LED	GPIO 4	Output, Low active	Green LED of LED ERR
WDI_OUT	GPIO 5	Output	GPIO for WDT
GPIO_BT0	GPIO 6	Input,Low active	Button for user in left of the Enclosure,mostly used for software system reset
DI0	GPIO 7		
SPI0_CS0#	GPIO 8		SPI0 for M.2 card
SPI0_MISO	GPIO 9		
SPI0_MOSI	GPIO 10		
SPI0_SCK	GPIO 11		
DI1	GPIO 12		
DI2	GPIO 13		
DI3	GPIO 14		
DI4	GPIO 15		
BEEP	GPIO 16	Output ,high active	
DO0	GPIO 17		
SPI1_CS0#	GPIO 18		SPI1 for SPI FLASH
SPI1_MISO	GPIO 19		
SPI1_MOSI	GPIO 20		
SPI1_SCK	GPIO 21		
DI5	GPIO 22		
DO2	GPIO 23		
DO3	GPIO 24		
DO4	GPIO 25		
DO5	GPIO 26		
DO1	GPIO 27		

3. Drivers and Programming Interfaces

All drivers have been packaged into system before shipping.We will show you she interface necessary for programming.And all these source is opened to our customers.

3.1 LED and GPIO

```
# GPIO Pin definitions
DI_PINS = [7, 12, 13, 14, 15, 22]    # DI0-DI5
DO_PINS = [17, 27, 23, 24, 25, 26]    # DO0-DO5
RUN_LED = 0
ERROR_LED = 4
BUZZER = 16

end_time = time.time() + 1
                        while time.time() < end_time:
                            values = [0] * len(indicator_pins)
                            values[error_idx] = 1
                            gpio_lines['indicators'].set_values(values)
                            time.sleep(0.1)
                            values[error_idx] = 0
                            gpio_lines['indicators'].set_values(values)
                            time.sleep(0.1)
except Exception as e:
            print(f"Alert error: {e}")
```

We connect the DI and DO signals together:

```
try:
        # Test each DO-DI pair
        for i in range(6):
            print(f"Testing DO{i} -> DI{i}")

            # Set all outputs to low
            output_values = [0] * 6
            gpio_lines['outputs'].set_values(output_values)
            time.sleep(0.02)    # Optimized: reduced delay

            # Set current output to high
            output_values[i] = 1
            gpio_lines['outputs'].set_values(output_values)
            time.sleep(0.02)    # Optimized: reduced delay
```

```
            # Read inputs
            input_values = gpio_lines['inputs'].get_values()

            if input_values[i] == 1:
                result.add_pass(f"GPIO{i} Loopback Test (DO{i}->DI{i})")
            else:
                result.add_fail(f"GPIO{i} Loopback Test", f"DO{i} output high level, but DI{i} not detected")

            # Reset output
            output_values[i] = 0
            gpio_lines['outputs'].set_values(output_values)
            time.sleep(0.02)    # Optimized: reduced delay

    except Exception as e:
        result.add_fail("GPIO Loopback Test", str(e))
```

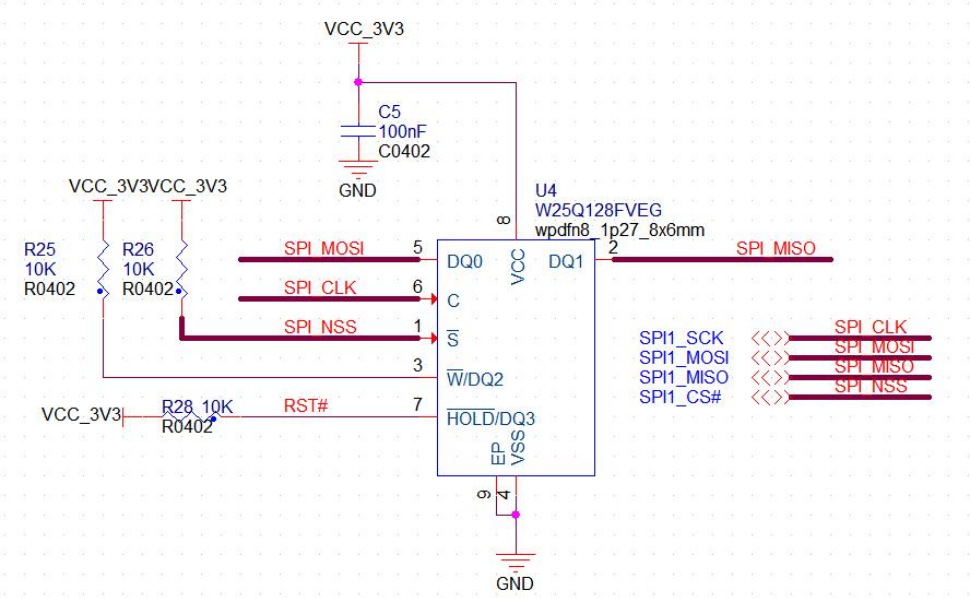
3.2 Serial Port (RS232 and RS485)

```
# RS485 send, RS232 receive test( A RS232----RS485 converter is needed in hardware)
        print("Testing RS485 -> RS232")
        try:
            with serial.Serial(rs485_port, baudrate, timeout=1) as rs485, \
                 serial.Serial(rs232_port, baudrate, timeout=1) as rs232:

                rs232.flushInput()
                rs485.write(test_data)
                time.sleep(0.1)    # Optimized: reduced wait time

                received = rs232.read(len(test_data))
                if received == test_data:
                    result.add_pass("RS485->RS232 Communication Test")
                else:
                    result.add_fail("RS485->RS232 Communication Test",
                                    f"Sent: {test_data}, Received: {received}")

        except Exception as e:
            result.add_fail("RS485->RS232 Communication Test", str(e))

        # RS232 send, RS485 receive test
        print("Testing RS232 -> RS485")
```

```
        try:
            with serial.Serial(rs485_port, baudrate, timeout=1) as rs485, \
                 serial.Serial(rs232_port, baudrate, timeout=1) as rs232:

                rs485.flushInput()
                rs232.write(test_data)
                time.sleep(0.1)    # Optimized: reduced wait time

                received = rs485.read(len(test_data))
                if received == test_data:
                    result.add_pass("RS232->RS485 Communication Test")
                else:
                    result.add_fail("RS232->RS485 Communication Test",
                                    f"Sent: {test_data}, Received: {received}")

        except Exception as e:
            result.add_fail("RS232->RS485 Communication Test", str(e))

    except Exception as e:
        result.add_fail("Serial Communication Test", str(e))
```

3.3 SPI flash memory

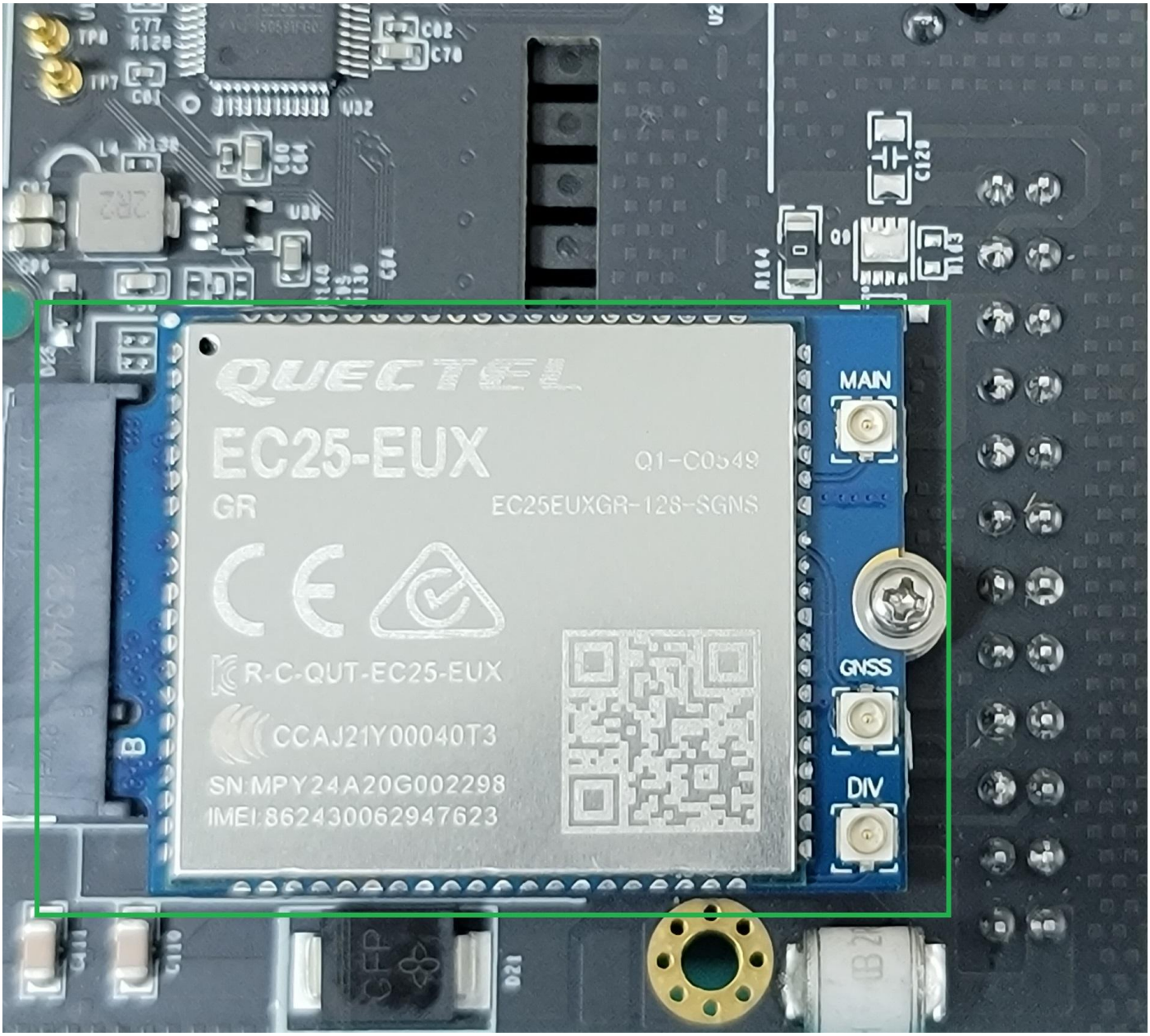
The flash chip is connected of the CM5 .



```
def test_spi_flash(result):
    """Test SPI1 Flash chip - W25Q128FVEG"""
    print(f"\n{Colors.CYAN}=== SPI Flash Test (W25Q128FVEG) ==={Colors.END}")
    try:
        spi_devices = ['/dev/spidev1.0', '/dev/spidev1.1']
        flash_found = False

        for spi_dev in spi_devices:
            if os.path.exists(spi_dev):
                print(f"Detected SPI device: {spi_dev}")
                try:
                    # Use spidev library for reliable SPI communication
                    import spidev
                    spi = spidev.SpiDev()
                    bus = int(spi_dev.split('/')[-1].split('.')[0][-1])    # Extract bus number
                    device = int(spi_dev.split('/')[-1].split('.')[1])     # Extract device number

                    spi.open(bus, device)
                    spi.max_speed_hz = 1000000    # 1MHz
                    spi.mode = 0

                    # Send JEDEC ID command (0x9F)
                    jedec_response = spi.xfer2([0x9F, 0x00, 0x00, 0x00])
                    spi.close()

                    if len(jedec_response) >= 4:
                        manufacturer_id = jedec_response[1]
                        memory_type = jedec_response[2]
                        capacity_code = jedec_response[3]

                        print(f"JEDEC ID: {jedec_response[1]:02X} {jedec_response[2]:02X} {jedec_response[3]:02X}")

                        # W25Q128FVEG JEDEC ID: EF 40 18
                        # EF = Winbond, 40 = Q Series, 18 = 128Mbit
                        if manufacturer_id == 0xEF and memory_type == 0x40 and capacity_code == 0x18:
                            capacity = "128Mbit (16MB)"
                            chip_model = "W25Q128FVEG"
                            result.add_pass(f"SPI Flash Detection: {chip_model}, Capacity={capacity}")
                            flash_found = True
                            break
                        elif manufacturer_id == 0xEF:    # Winbond chip but different model
                            # Calculate capacity based on capacity code
                            capacity_mb = 2**(capacity_code - 13)    # W25Q series capacity calculation
                            result.add_pass(f"SPI Flash Detection: Winbond chip, Capacity={capacity_mb}MB
(Code:{capacity_code:02X})")
```

```
                                print(f"Warning: Expected W25Q128FVEG (EF4018), actually detected {manufacturer_id:02X}{memory_type:02X}{capacity_code:02X}")
                                flash_found = True
                                break
                            else:
                                print(f"Detected other Flash chip: {manufacturer_id:02X}{memory_type:02X}{capacity_code:02X}")
                                result.add_fail("SPI Flash Model Verification", f"Expected W25Q128FVEG (EF4018), actual {manufacturer_id:02X}{memory_type:02X}{capacity_code:02X}")
                                flash_found = True
                                break
                        else:
                            print(f"{spi_dev}: No valid JEDEC response")

                    except ImportError:
                        print("Warning: spidev library not installed, using fallback method")
                        # Fallback method: directly read device file
                        try:
                            with open(spi_dev, 'w+b') as spi_file:
                                id_cmd = bytes([0x9F, 0x00, 0x00, 0x00])
                                spi_file.write(id_cmd)
                                spi_file.seek(0)
                                response = spi_file.read(4)

                                if len(response) >= 4 and response[1] == 0xEF:
                                    result.add_pass(f"SPI Flash Detection (fallback method): Detected Winbond chip")
                                    flash_found = True
                                    break
                        except Exception as e:
                            print(f"{spi_dev}: Fallback method also failed - {e}")

                    except Exception as e:
                        print(f"{spi_dev}: Read failed - {e}")

            if not flash_found:
                result.add_fail("SPI Flash Detection", "W25Q128FVEG Flash chip not found")

        except Exception as e:
            result.add_fail("SPI Flash Test", str(e))
```

3.4 4G/LTE



1. **Insert the EC20 into M.2 socket and NANO sim card in related slot, connect the antenna.**
2. **Log in the system via console use pi/raspberry.**
3. **Turn on the power of M.2 socket and release the reset signal.**

```
$ sudo -i      #enable root account privileges
$ cd /sys/class/gpio
$ echo 1 > export    #GPIO1 which is POW_ON signal

$ cd gpio1
```

$ echo out > direction
$ echo 1 > value # turn off the power of M.2
$ echo 0 > value # turn on the power of M.2

NOTE: Only 3042 size card is supported.
NOTE: Insert micro sim card first.
NOTE: Check your country area, make sure it mach with the 4G module you insert.
NOTE: Check the LED of 4G in front panel, it should start to flash.

Check the device:

```
#!/bin/bash

# DEVICE
DEVICE="/dev/ttyUSB2"
TEMP_FILE="/tmp/response.txt"
PING_HOST="www.baidu.com"
NET_INTERFACE="usb0" #maybe changed to actual interface
PING_COUNT=3              # pact cnt
PING_TIMEOUT=2            # time out seconds

# get_network_info
get_network_info() {
    echo "getting network."
    ip addr show "$NET_INTERFACE"
    echo
}

# checking the network
check_network() {
    echo "checking: $NET_INTERFACE"
    if ping -I "$NET_INTERFACE" -c "$PING_COUNT" -W "$PING_TIMEOUT" "$PING_HOST" > /dev/null
  2>&1; then
        echo " It is working   $PING_HOST。"
        exit 0
    else
        echo "Can not ping $PING_HOST。"
    fi
}

# 检查网卡接口是否存在，并测试连接
if ip link show "$NET_INTERFACE" > /dev/null 2>&1; then
    get_network_info
    check_network
fi
```

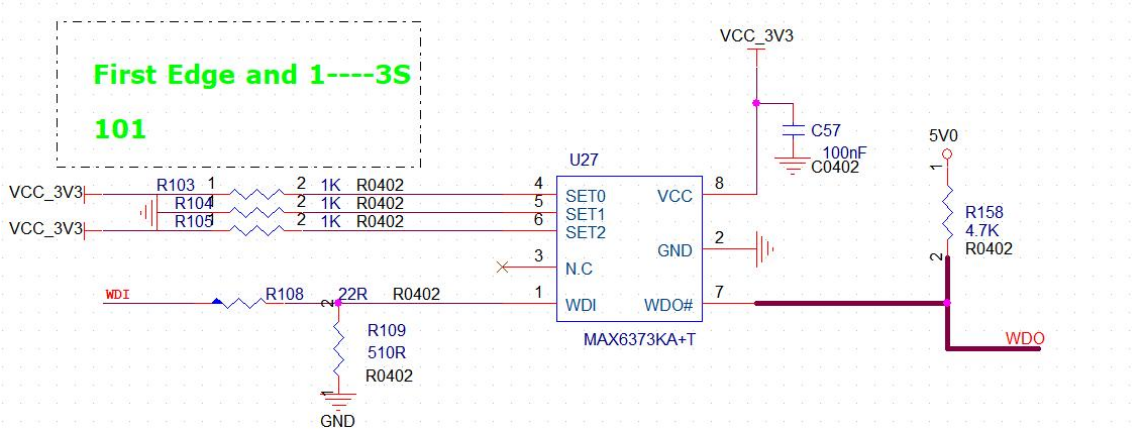
```
# 确保 ModemManager 未干扰配置
if systemctl is-active --quiet ModemManager; then
    echo "ModemManager 正在运行，停止并禁用它..."
    sudo systemctl stop ModemManager
    sudo systemctl disable ModemManager
fi

# 配置串口设备
if [ ! -e "$DEVICE" ]; then
    echo "设备 $DEVICE 未找到!"
    exit 1
fi

# 配置串口（假设波特率 115200）
stty -F "$DEVICE" 115200 cs8 -cstopb -parenb

# 发送配置命令
echo "配置 USB 网络..."
echo -e "AT+QCFG=\"usbnet\",3\r\n" > "$DEVICE"
sleep 1
cat "$DEVICE" > "$TEMP_FILE" &
sleep 2

# 检查配置响应
RESPONSE=$(cat "$TEMP_FILE")
if echo "$RESPONSE" | grep -q "OK"; then
    echo "USB 网络配置成功"
else
    echo "USB 网络配置失败: $RESPONSE"
    exit 1
fi

# 使能功能并重启模块
echo "重启模块..."
echo -e "AT+CFUN=1,1\r\n" > "$DEVICE"
sleep 35

# 函数：检查 USB 设备
check_usb_device() {
    echo "检查 USB 设备..."
    lsusb
    echo
}

# 函数：检查 ttyUSB2 是否存在
```

```
check_ttyusb2() {
    echo "检查 /dev/ttyUSB2 设备节点..."
    if [ -e "$DEVICE" ]; then
        echo "$DEVICE 存在"
        return 0
    else
        echo "$DEVICE 不存在"
        return 1
    fi
    echo
}

# 函数：等待设备重新连接
wait_for_device() {
    echo "等待 $WAIT_TIME 秒..."
    sleep "$WAIT_TIME"
}

# 再次获取网卡信息并测试网络连接
echo "再次检查网络连接..."
get_network_info
if ping -I "$NET_INTERFACE" -c "$PING_COUNT" -W "$PING_TIMEOUT" "$PING_HOST" > /dev/null
  2>&1; then
    echo "配置完成，能够 ping 通 $PING_HOST。"
    exit 0
else
    echo "配置失败，不能 ping 通 $PING_HOST。"
    exit 1
fi
```

3.5 WDT

This is a WDT chip from Analog devices named MAX6373.

3.5.1 Block Diagram of WDT



3.5.2 How it works

During normal operation, the microprocessor should repeatedly toggle the watchdog input (WDI) before the selected watchdog timeout period elapses to demonstrate that the system is processing code properly. If the CM5 does not provide a valid watchdog input transition before the timeout period expires, the supervisor asserts a watchdog (WDO) output to signal that the system is not executing the desired instructions within the expected time frame.

The watchdog output pulse can be used to reset the CM5. The MAX6373 are flexible watchdog timer supervisors that can increase system reliability through notification of code execution errors. The family offers several pin-selectable watchdog timing options to match a wide range of system timing applications:

- Watchdog startup delay: provides an initial delay before the watchdog timer is started.
  In this system , First Edge of pulse from WDI signal is used to active the WDT.
- Watchdog timeout period: **1 second** is selected of watchdog timeout period after the initial startup delay.

Key part of testing code:

```
pulse_count = 0
    test_duration = 10    # 测试 10 秒
    pulse_interval = 1.0    # 每秒喂狗一次

    start_time = time.time()

    try:
```

```
        while (time.time() - start_time) < test_duration and running:
            # 发送喂狗信号
            if send_watchdog_pulse():
                pulse_count += 1
                current_time = datetime.now().strftime('%H:%M:%S.%f')[:-3]
                print(f"[{current_time}] {Colors.GREEN}✓{Colors.END} 发送第 {pulse_count} 次喂狗信号")
            else:
                print(f"{Colors.RED}✗ 喂狗信号发送失败{Colors.END}")
                return False

            time.sleep(pulse_interval)

        print(f"\n{Colors.GREEN}✓ 基本测试完成: 成功发送 {pulse_count} 次喂狗信号{Colors.END}")
        return True

    except Exception as err:
        print(f"{Colors.RED}测试出错: {e}{Colors.END}")
        return False
```

4. Electrical specifications

4.1 Power consumption

The power consumption of the EdgeBox-RPI5 strongly depends on the application, the mode of operation and the peripheral devices connected. The given values have to be seen as approximate values. The following table shows power consumption parameters of the EdgeBox-RPI5:

Note: On condition of power supply 24V, no add-on card in sockets and no USB devices.

Mode of operation	Current(ma)	Power	Remark
Idle	81		
Stress test	172		stress -c 4 -t 10m -v &

5. Mechanical Drawings

Unit: mm