1. USER LED

DESCRIPTION

The RuggedBoard Using Three SMD LEDs. SMD LEDs have an operating voltage between 1.8V to 3.6V. Led Connected with GPIO Pin. Gpio pin value 1 LED ON , GPIO Pin value 0 LED OFF.

The location of the LEDs.

LED No	SIGNAL NAME
LED_1 (D4)	PC13/GPIO_LED
LED_2 (D7)	PC17/GPIO_LED
LED_3 (D17)	PC19/GPIO_LED

README FILE

The script first exports GPIO 77 by writing the number 77 to the /sys/class/gpio/export file.

Then, it sets the direction of the pin to "out" by writing "out" to the /sys/class/gpio/PC13/direction file.

Next, the script reads the value of the GPIO pin by reading the contents of the /sys/class/gpio/PC13/value file into the "status" variable.

If the value of the GPIO pin is 0, the script sets the value of the GPIO pin to 0 by writing "0" to the /sys/class/gpio/PC13/value file, turns on the LED, and prints a "PASS" message.

If the value of the GPIO pin is not 0, the script prints a "FAIL" message.

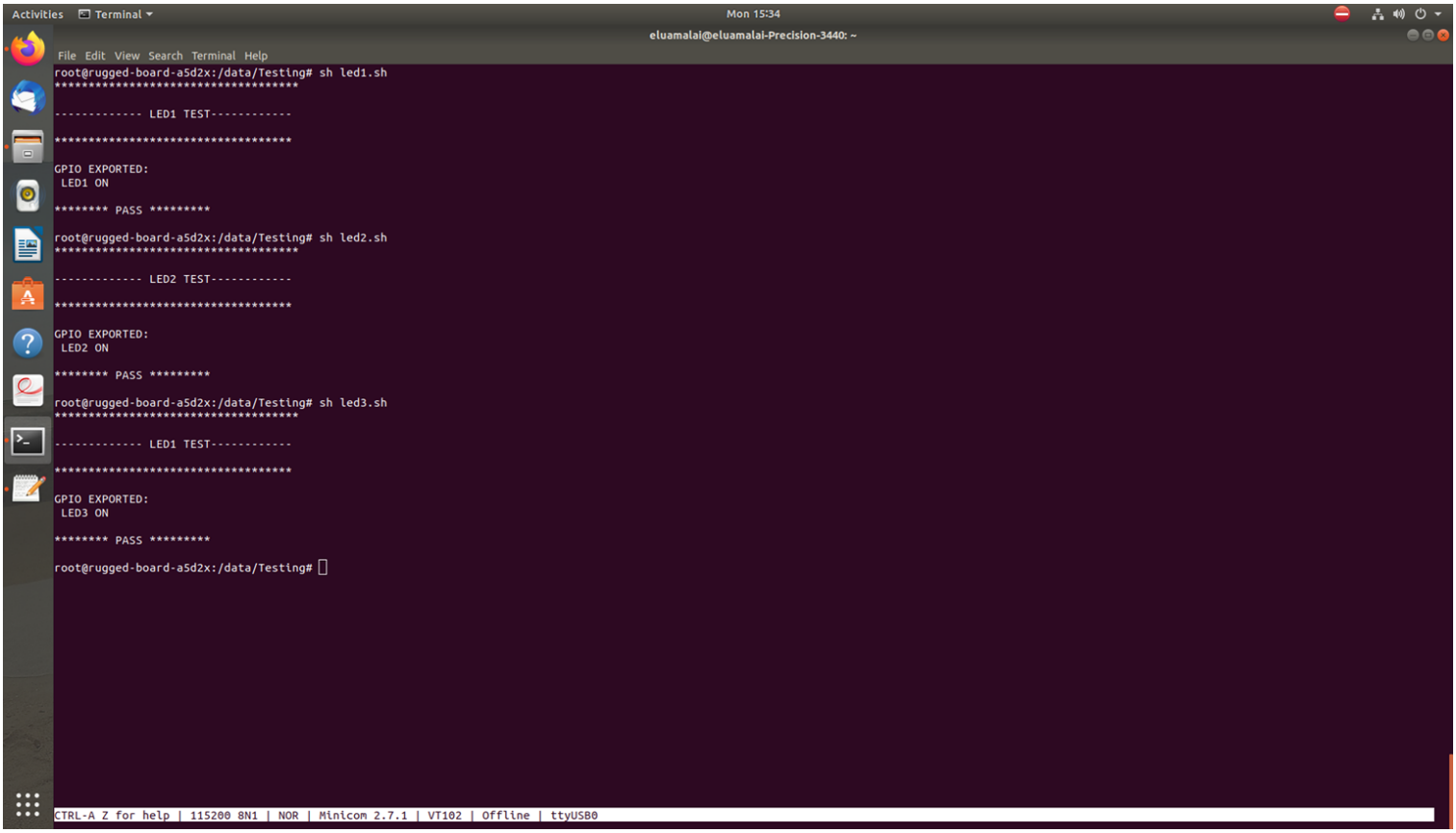
TESTING SCRIPT

```
#! /bin/bash
echo "**************************************"
echo
echo "---------- LED 1 TEST----------"
echo
echo "**************************************"
echo

echo 77 > /sys/class/gpio/export

echo "GPIO EXPORTED: "
echo out > /sys/class/gpio/PC13/direction
```

```
status=$(cat /sys/class/gpio/PC13/value)

if [ "$status" -eq 0 ]
 then
    echo 0 > /sys/class/gpio/PC13/value
    echo " LED1 ON "
    echo
    echo "******** PASS *********"
    echo
else
 echo "LED 1 NOT FOUND "
    echo
    echo "******** FAIL *********"
    echo
fi


#! /bin/bash
echo "**************************************"
echo
echo "---------- LED 2 TEST----------"
echo
echo "**************************************"
echo

echo 81 > /sys/class/gpio/export

echo "GPIO EXPORTED: "
echo out > /sys/class/gpio/PC17/direction
status=$(cat /sys/class/gpio/PC17/value)

if [ "$status" -eq 0 ]
 then
    echo 0 > /sys/class/gpio/PC17/value
    echo " LED2 ON "
    echo
    echo "******** PASS *********"
    echo
else
 echo "LED 2 NOT FOUND "
    echo
    echo "******** FAIL *********"
    echo
fi
```

```
echo "************************************"
echo
echo "---------- LED 3 TEST----------"
echo
echo "************************************"
echo

echo 83 > /sys/class/gpio/export

echo "GPIO EXPORTED: "
echo out > /sys/class/gpio/PC19/direction
status=$(cat /sys/class/gpio/PC19/value)

if [ "$status" -eq 0 ]
 then
    echo 0 > /sys/class/gpio/PC19/value
    echo " LED1 ON "
    echo
    echo "******** PASS *********"
    echo
else
 echo "LED 2 NOT FOUND "
    echo
    echo "******** FAIL *********"
    echo
fi
```

OUTPUT TESTING LOG



2. USER_SWITCH

DESCRIPTION

This button is used to flash new image to RuggedBoard. To enable flashing mode, press this button while connecting the micro USB cable.

The RuggedBoard Using Push-button SMD switches, have a DC voltage rating between 12V to 48V supporting

The location of the

Switch No SIGNAL NAME

SW1 PC12/GPIO_EN

README FILE

The GPIO pin 76 is exported using echo 76 > /sys/class/gpio/export to allow access to the pin.

The value of the GPIO pin is read using status=$(cat /sys/class/gpio/PC12/value) and stored in the status variable.

An if statement checks if the value of status is equal to 0, indicating that the button is pressed.

If it is, the script prints "Button pressed", "PASS"

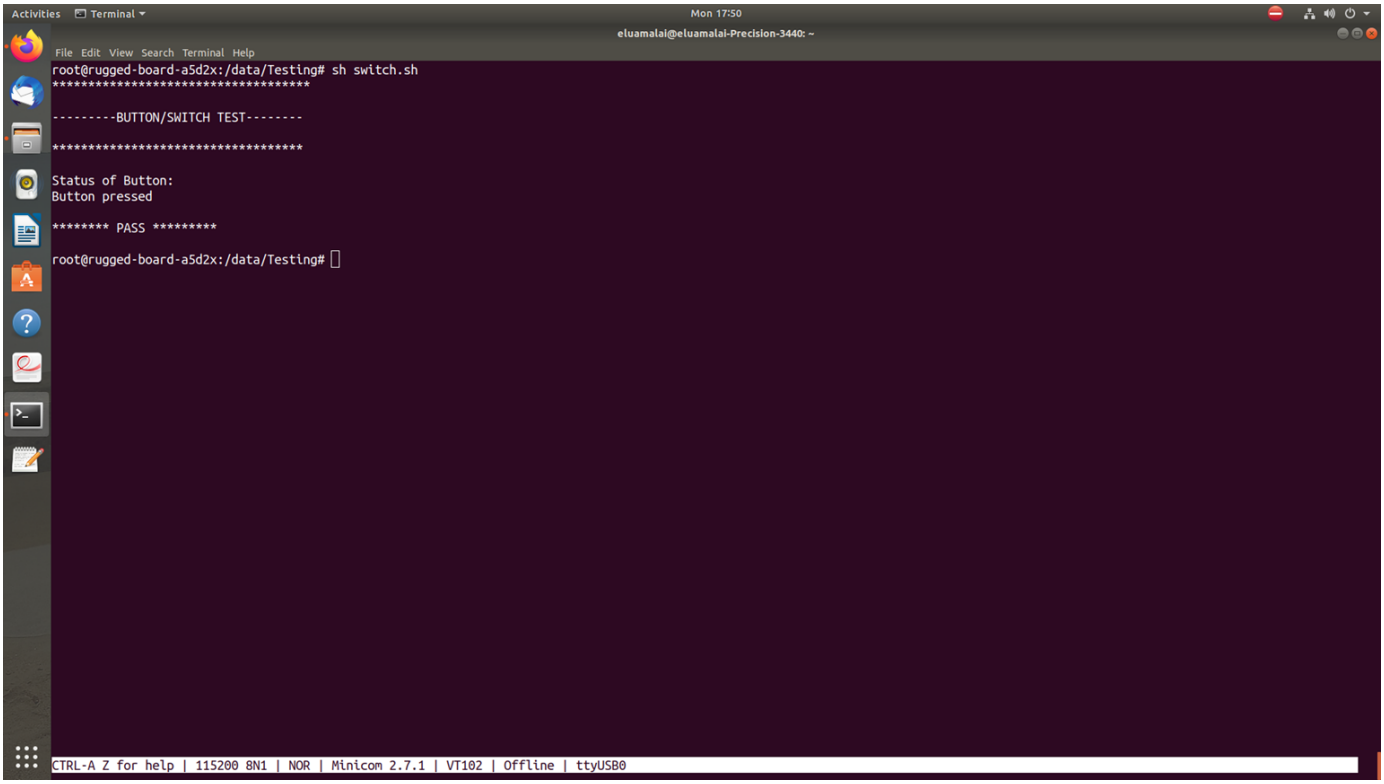
If not, it prints "Button not pressed", "FAIL"

TESTING SCRIPT

```
#! /bin/bash
echo "**************************************"
echo
echo "---------BUTTON/SWITCH TEST--------"
echo
echo "**************************************"
echo

# Export GPIO pin 76
echo 76 > /sys/class/gpio/export

echo "Status of Button: "

# Read the value of the GPIO pin
status=$(cat /sys/class/gpio/PC12/value)

if [ "$status" -eq 0 ]
 then
    echo "Button pressed"
    echo
    echo "******** PASS *********"
    echo
 echo
else
    echo "Button pressed"
    echo
    echo "******** FAIL *********"
```

```
    echo
fi
```

OUTPUT TESTING LOG



3 ADC (Analog-to-Digital Converter)

DESCRIPTION

Electronic circuit that measures a real-world signal (such as temperature, pressure, acceleration, and speed) and converts it to a digital representation of the signal

The ADC is based on a 12-bit Analog-to-Digital Converter (ADC) managed by an ADC Controller providing enhanced resolution up to 14 bits

it also integrates a 12-to-1 analog multiplexer, making possible the analog-to-digital conversions of 12 analog lines

Conversion results are reported in a common register for all channels, as well as in a channel-dedicated register.

README FILE

The if statement is checking if the ADC node exists at /sys/bus/iio/devices/iio\:device0/in_voltage12_raw.

This location may vary depending on the system configuration and ADC driver being used.

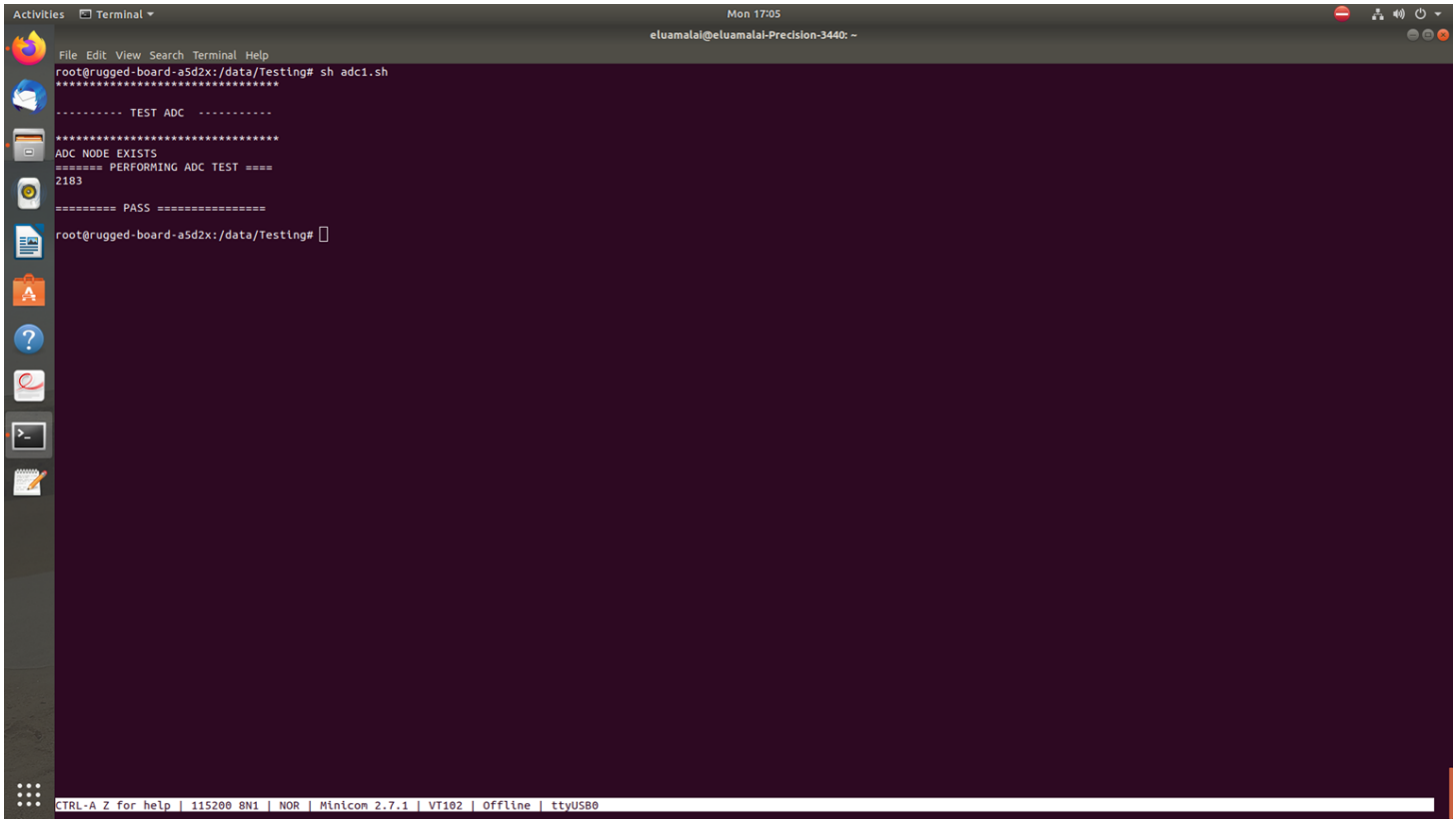
- If the ADC node exists, the script reads the raw value of the voltage being measured using the cat command and stores it in the variable var.
- The script then prints out the value of var, indicating that the ADC test has been performed and the value has been successfully read.
- Finally, the script prints a "PASS" message to indicate that the ADC test has been successful.
- If the ADC node does not exist, the script prints a message indicating that the ADC node could not be found.

TESTING SCRIPT

```
#! /bin/bash
echo "**********************************"
echo
echo "---------- TEST ADC  -----------"
echo
echo "**********************************"

if [ -e /sys/bus/iio/devices/iio\:device0/in_voltage12_raw ]
 then
  echo "ADC NODE EXISTS"
  var=$(cat /sys/bus/iio/devices/iio\:device0/in_voltage12_raw)
  echo "======= PERFORMING ADC TEST ===="
  echo $var
```

```
  echo
  echo "========== PASS ================"
  echo
else
  echo "ADC NODE NOT FOUND"
fi
```

OUTPUT TESTING LOG



4 ETHERNET

DESCRIPTION

The Ethernet MAC (GMAC) module implements a 10/100 Mbps Ethernet MAC compatible with the IEEE 802.3 standard.
The GMAC can operate in either half or full duplex mode at all supported speeds
Eth signals from the SOM are connected to a RJ45 MagJack.

README FILE

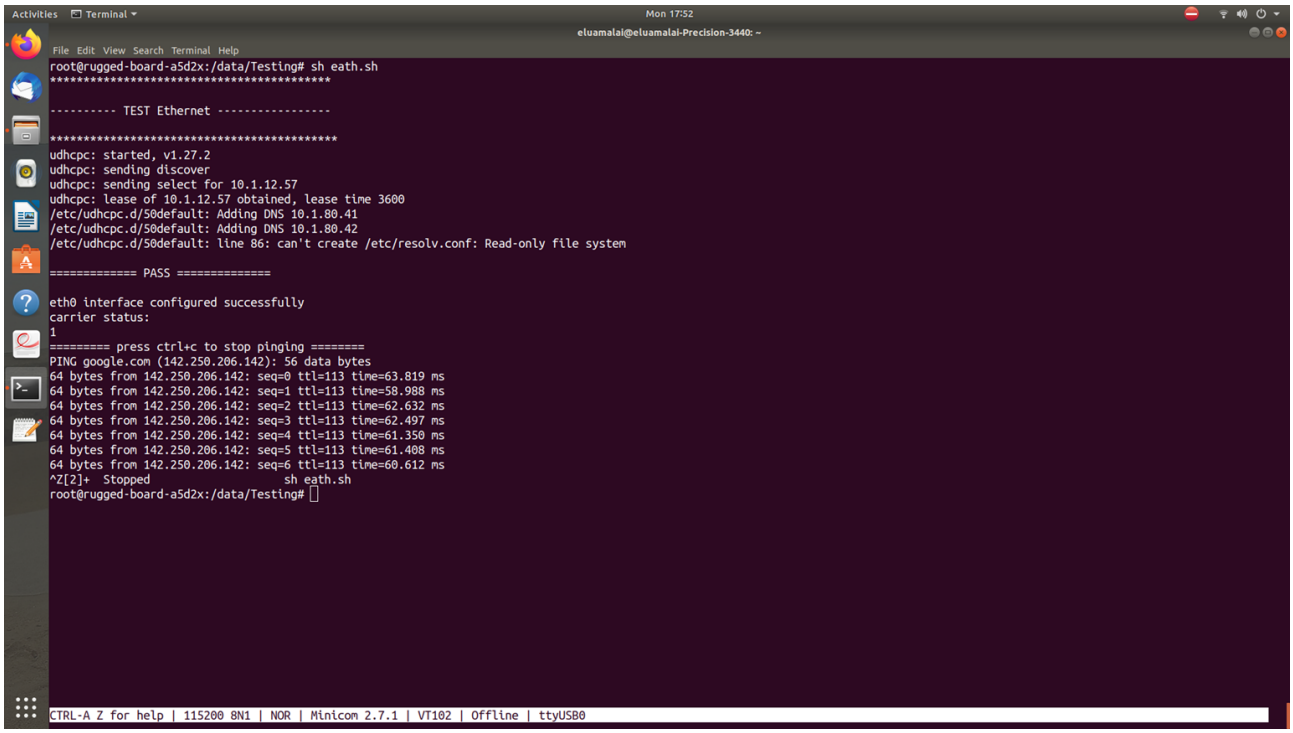
The script tries to configure the eth0 interface using the "udhcpc" command. If the configuration is successful, the script prints out "PASS" and a message indicating that the eth0 interface has been configured successfully.

- The script prints out the carrier status of the eth0 interface by reading the value of the /sys/class/net/eth0/carrier file.
- The script instructs the user to press "ctrl+c" to stop pinging and then pings "google.com" to test the Ethernet connection.
- If the configuration of the eth0 interface is not successful, the script prints out "FAIL".

TESTING SCRIPT

```
#!/bin/bash

echo "*****************************************"
echo
echo "---------- TEST Ethernet ----------------"
echo
echo "*****************************************"

if udhcpc -i eth0 ;
 then
    echo
    echo "============= PASS =============="
    echo
    echo "eth0 interface configured successfully"
    echo "carrier status:"
    cat /sys/class/net/eth0/carrier
    echo "========= press ctrl+c to stop pinging ========"

    ping google.com
```

```
else
    echo "Failed to configure eth0 interface"
    echo "============== FAIL =============="
    echo
fi
```



5 USB

DESCRIPTION

In RuggedBoard there are two stacked USB2.0 Host Ports. Both USB1 &USB2 are configured
as Host. USB2 signal are also used for the mPCIe port (P8).

The switching happens through USB mux switch configuration.

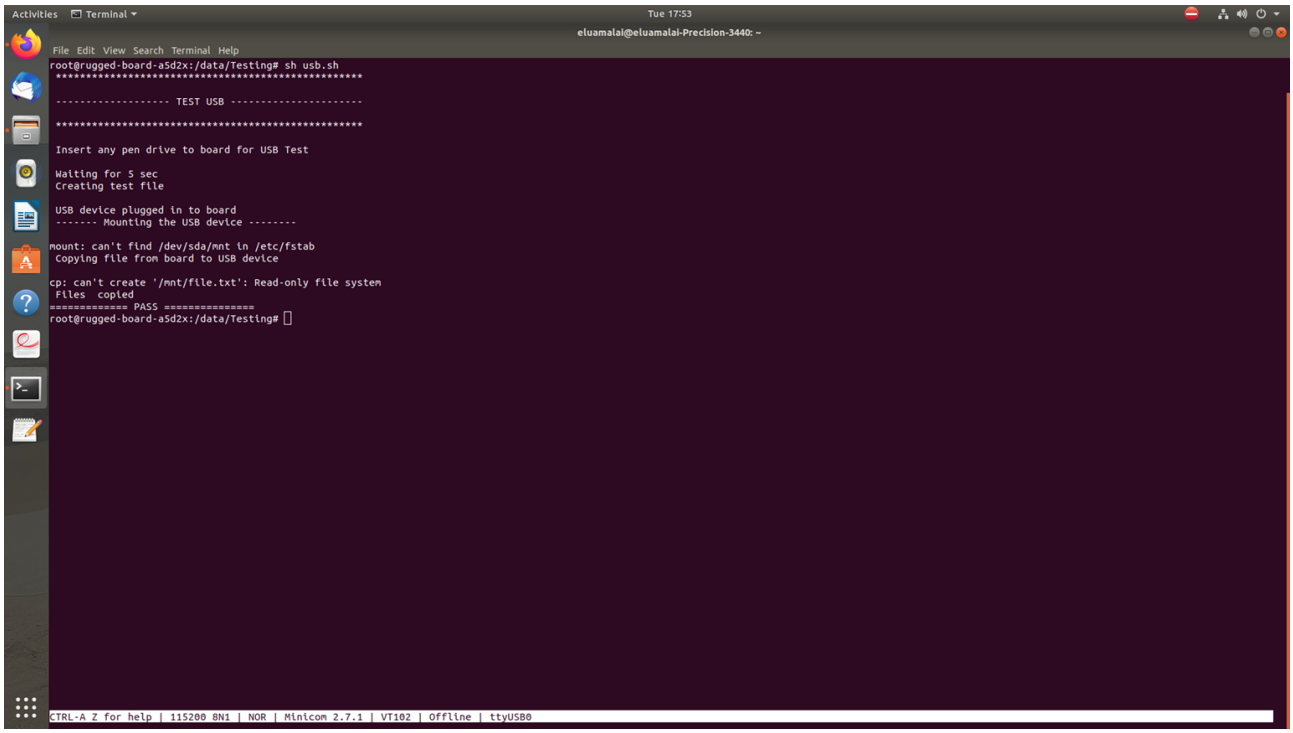
For software configuration DNM the resistor R65, R66, R70 and R71. Then mount R68 and R69 with signal.

TESTING SCRIPT

```
#!/bin/bash
echo " ************************************************** "
echo
echo " ---------- TEST USB ---------- "
```

```
echo
echo " ************************************************** "
echo
echo " Insert any pen drive to board for USB Test "
echo
echo " Waiting for 5 sec "
sleep 5
echo " Creating test file "
touch /data/Testing/file.txt
echo " Welcome on phytec " > file.txt

if [ -e "/dev/sda" ] || [ -e "/dev/sdb" ]
then
   echo
   echo " USB device plugged in to board "
   echo " ------- Mounting the USB device -------- "
   echo
   mount  /dev/sda/mnt
   cd /mnt || exit
echo " Copying file from board to USB device "
   echo
   cp /data/Testing/file.txt  /mnt
   cd / || exit

   if [ -e "/mnt/file.txt" ]
   then
      echo " Copying file from board to USB device successful "
      echo
      echo "============= PASS ==============="
      echo
   else
      echo " Files not copied "
      echo "============= FAIL ==============="
   fi
else
   echo " USB device not found "
   echo "============= FAIL ==============="
fi
```

OUTPUT LOG



6 SD_CARD

DESCRIPTION

The Secure Digital MultiMedia Card Controller (SDMMC) supports the embedded MultiMedia Card (e.MMC) Specification V4.51, the SD Memory Card Specification V3.0, and the SDIO V3.0 specification.
It is compliant with the SD Host Controller Standard V3.0 specification.
The SDMMC includes the register set defined in the “SD Host Controller Simplified Specification V3.00” and additional registers to man- age e.MMC devices, sampling tuning procedure, PAD calibration and enhanced features.

README FILE

STEP1
creates an empty file named "data.txt" in the "/data/Testing" directory.
and writes the text " Welcome on phytec " to the "data.txt" file.

STEP2
checks if satement the device file "/dev/mmcblk1p2" exists or not.

STEP 3
copying file from board to sd card"

STEP 4

copying of file from board to sd card successful means PASS

STEP 5

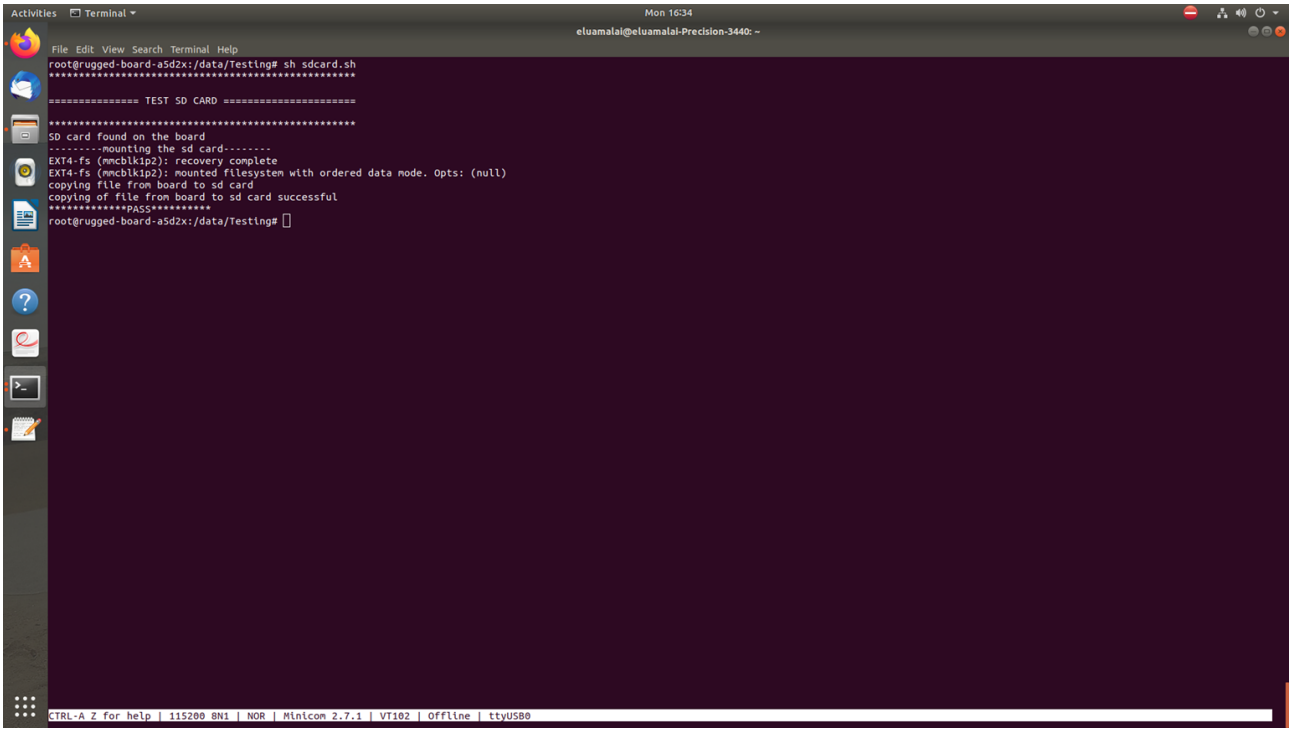
copying of file from board to sd card Not successful means FAIL

STEP 6
SD_CARD not insert output will be "No sd card found on the board"

TESTING SCRIPT

```
#!/bin/sh

echo "*****************************************************"
echo
echo "=============== TEST SD CARD ======================="
echo
echo "*****************************************************"
touch /data/Testing/data.txt
echo " Welcome on phytec " > data.txt

if [ -e "/dev/mmcblk1p2" ]
then
  echo "SD card found on the board"
  echo "---------mounting the sd card--------"
  mount /dev/mmcblk1p2 /mnt/
  cd /mnt/
  echo "copying file from board to sd card"
  cp /data/Testing/data.txt /mnt/
  if [ -e "/mnt/data.txt" ]
 then
    echo "copying of file from board to sd card successful"
    echo "**************PASS***********"
  else
 echo "file not copied to sd card"
    echo "***********FAIL*************"
  fi
else
  echo "No sd card found on the board"
fi
```

OUTPUT LOG



7 RS232 UART

DESCRIPTION

The Universal Asynchronous Receiver Transmitter (UART) features a two-pin UART that can be used for communication and trace pur-poses and offers an ideal medium for in-situ programming solutions.

The RS-232 level signals are available at the connector P4.

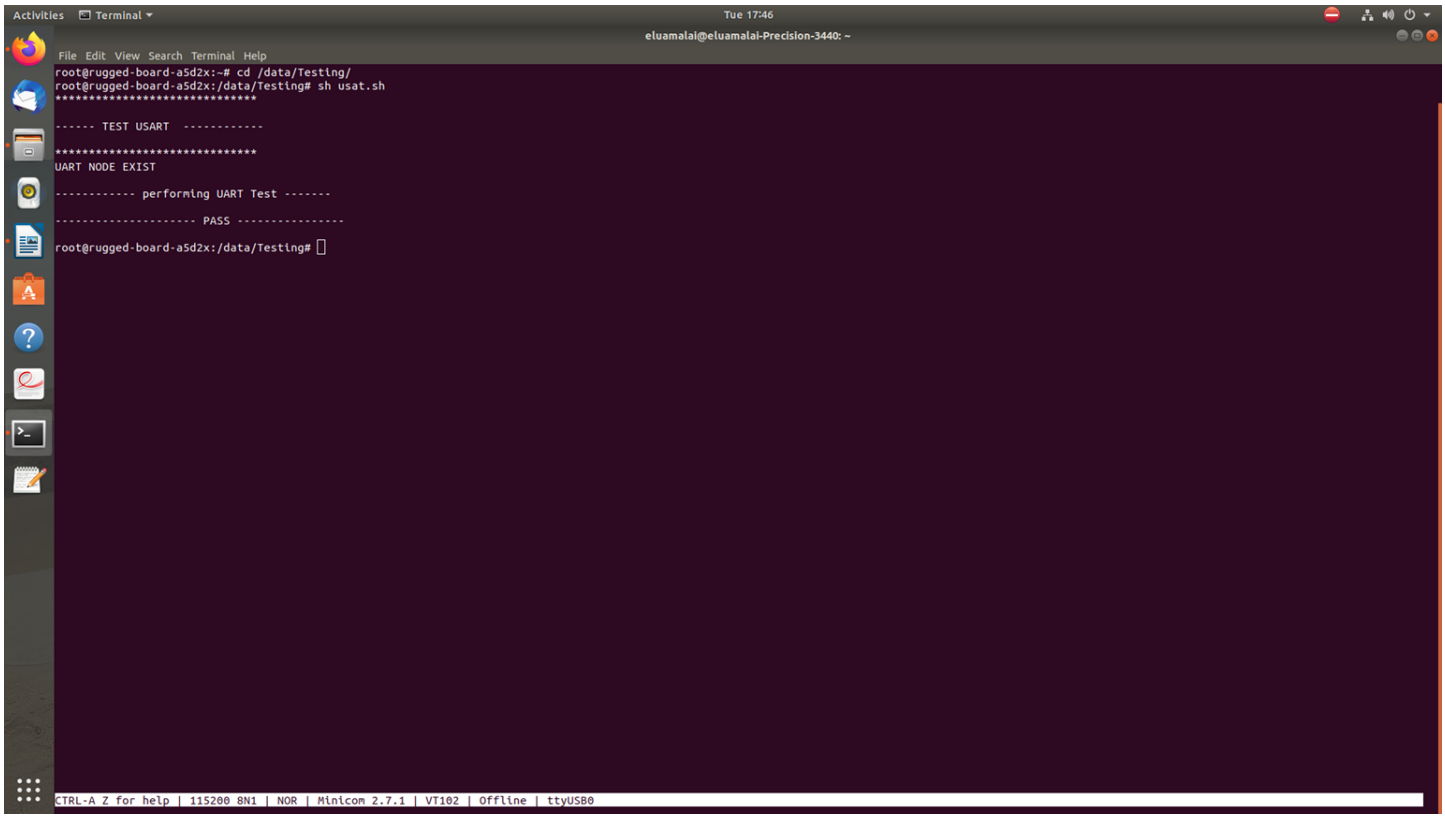
PIN DESCRIPTION	SIGNAL NAME	Software Node
RS232_TX_1	PB27_UTXD0/LCDDAT16	/dev/ttyS4
RS232_RX_1	PB26_URXD0/LCDDAT15	/dev/ttyS4
GND	Ground	

RS232_TX_2 PB4_UTXD4 /dev/ttyS1
RS232_RX_2 PB3_URXD4 /dev/ttyS1

TESTING SCRIPT

```
#! /bin/bash

echo  "********************************"

echo

echo "------ TEST USART  ------------"

echo

echo  "********************************"

if [ -e "/dev/ttyS1" ]

then

echo "ADC NODE EXIST"

echo

echo "------------ performing UART Test ------- "

echo

echo "---------------------- PASS ----------------"

echo

else

echo "USART Node NOT Found "

echo "---------------------- FAIL ----------------"

fi
```

OUTPUT TESTING LOG



8 EEPROM

DESCRIPTION

The Inter-IC Sound Controller (I2SC) provides a 5-wire, bidirectional, synchronous, digital audio link to external audio devices: I2SC_DI,I2SC_DO, I2SC_WS, I2SC_CK, and I2SC_MCK pins.
The I2SC is compliant with the Inter-IC Sound (I 2S) bus specification.

The I2SC consists of a receiver, a transmitter and a common clock generator that can be enabled separately to provide Master, Slave or Controller modes with receiver and/or transmitter active.

DMA Controller channels, separate for the receiver and for the transmitter, allow a continuous high bit rate data transfer without processor

README FILE

- The script uses the "i2cdetect" command to detect devices on the I2C bus.
- The "-y" option indicates that the user should not be prompted to confirm device connections, and the "-r 0" option specifies that the test should be run on I2C bus 0.
- The script then checks for the existence of an EEPROM (Electrically Erasable Programmable Read-Only Memory) device connected to the I2C bus using the "if [-e /sys/class/i2c-adapter/i2c-0/0-0050/eeprom]" condition.

If the device is present, the script prints out a message indicating that it is reading the value of the EEPROM.

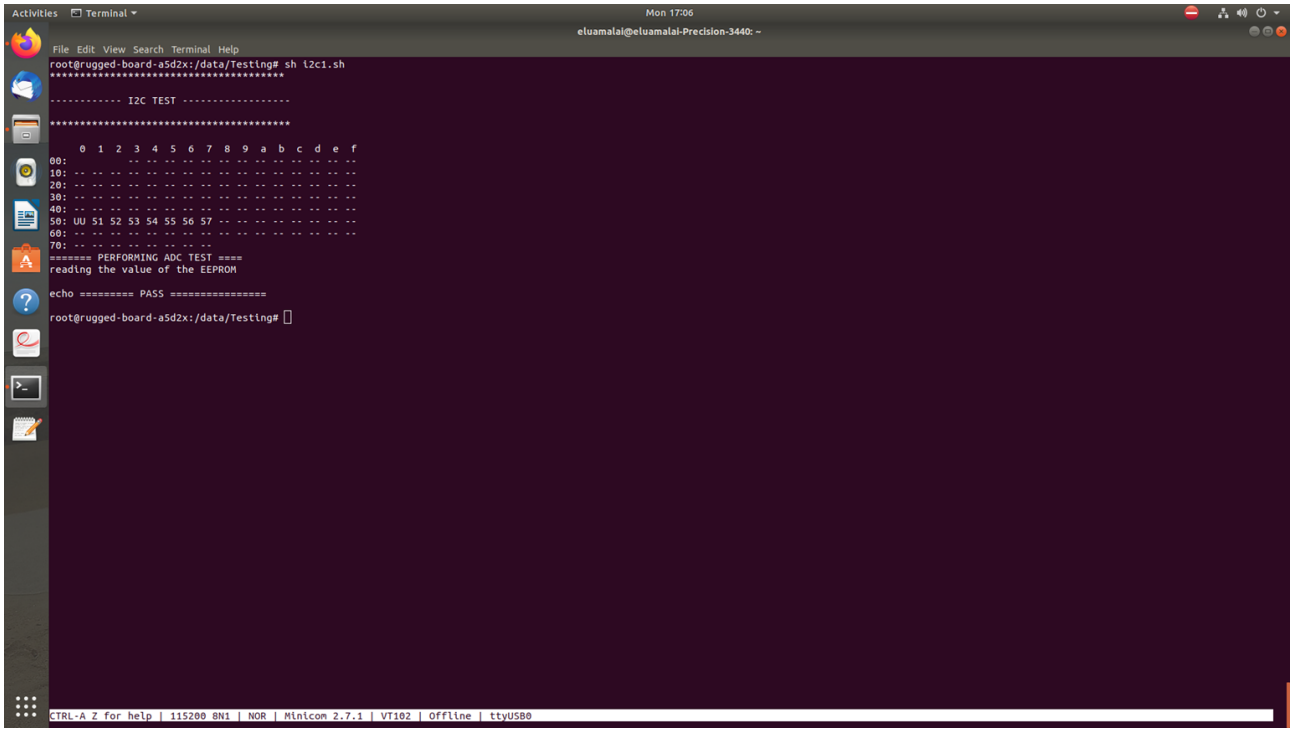
- If the EEPROM is present, the script reads the value of the EEPROM using the "cat" command and stores it in the "value" variable (which is currently commented out).
- The script prints out "PASS" if the EEPROM is present and the value can be read, and "FAIL" if the EEPROM is not present.

TESTING SCRIPT

```
#! /bin/bash
echo
echo "------------ I2C TEST ------------------"
echo
echo "*****************************************"
echo

i2cdetect -y -r 0
echo "======= PERFORMING ADC TEST ===="

if [ -e /sys/class/i2c-adapter/i2c-0/0-0050/eeprom ]
then
echo "reading the value of the EEPROM"
#value=$(cat /sys/class/i2c-adapter/i2c-0/0-0050/eeprom)
```

#echo $value
echo
echo echo "========== PASS ================"
echo
else
echo
echo "*************** FAIL **************"
echo
fi

OUTPUT TESTING LOG



9 DIN (DIGITAL INPUT)

DESCRIPTION

• To test the digital output pins on carrier boardTo test the digital output pins on carrier board

Required Hardware

• phyCORE-A5d2x System on Module
• phyCORE-A5d2 Rugged Board
• USB cable
• External LED

READMEFILE

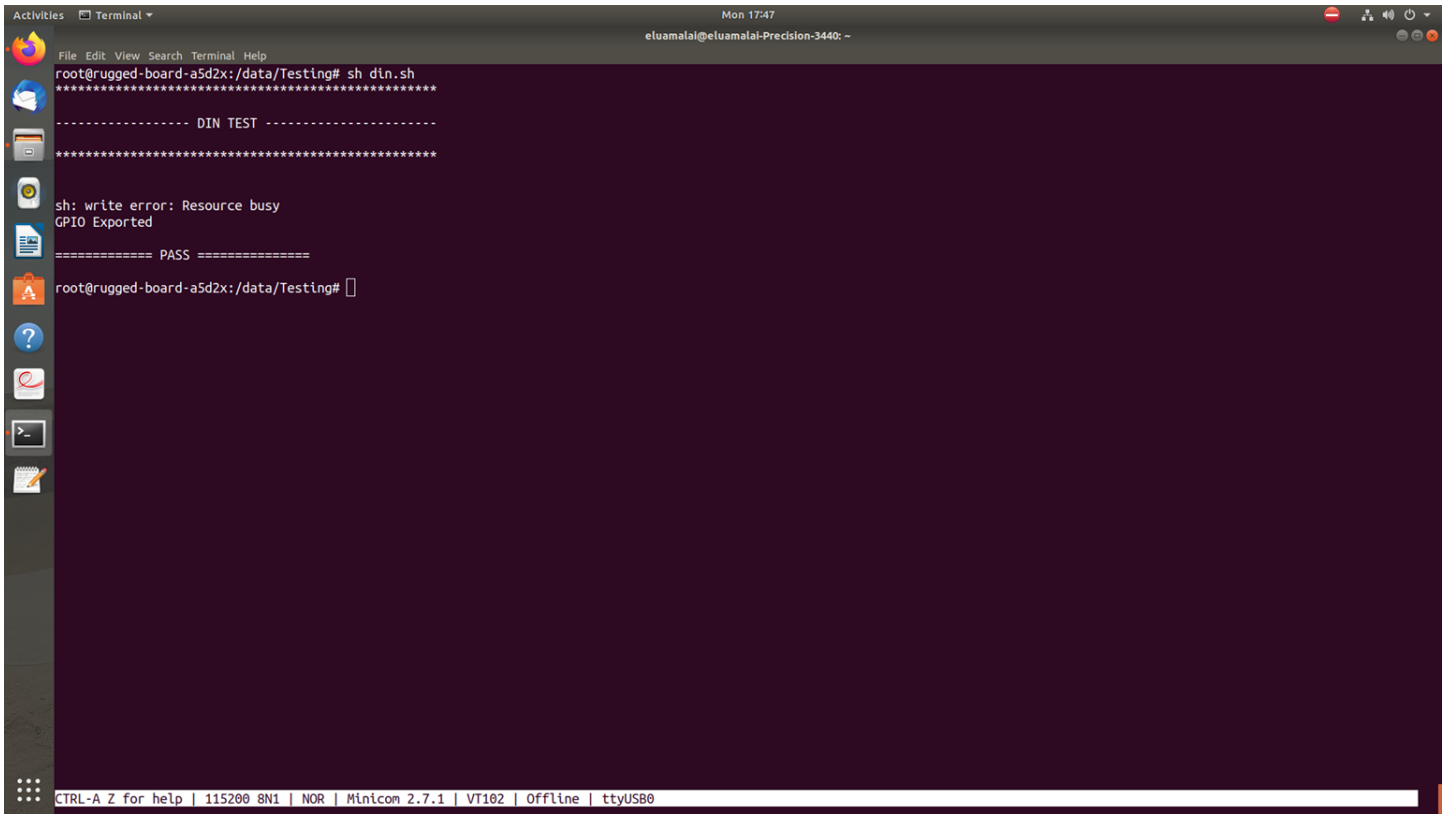
1. The first few lines of the script print out a banner indicating that this is a test script for the GPIO pin.
2. The script exports GPIO pin 84.
3. The script checks if the PC20 directory exists. If it does, it prints out "GPIO Exported." If it doesn't, it prints out "GPIO Export Failed."
4. The script sets the direction of the GPIO pin to "in".
5. The script sleeps for 2 seconds.
6. The script reads the value of the GPIO pin and stores it in the gpio5_value variable.
7. The script checks if the gpio5_value is equal to 1. If it is, the script prints out "PASS." If it isn't, the script prints out "FAIL."

TESTING SCRIPT

```
#!/bin/bash
echo "*******************************************************"
echo
echo "------------------ DIN TEST ----------------------"
echo
echo "*******************************************************"
echo
echo
echo 84 > /sys/class/gpio/export
if [ -d "/sys/class/gpio/PC20" ]
then
   echo "GPIO Exported"
else
   echo "GPIO Export Failed"
fi
echo in > /sys/class/gpio/PC20/direction
sleep 2
gpio5_value=$(cat /sys/class/gpio/PC20/value)
if [ "$gpio5_value" == 1 ]
then
   echo
   echo "============= PASS ==============="
   echo
else
   echo
```

```
    echo "============== FAIL ================"
    echo
fi
```

OUTPUT LOG



10 DOUT (DIGITAL OUTPUT)

DESCRIPTION

• To test the digital input pin on the carrier board.To test the digital input pin on the carrier board.

• **Required Hardware**

• phyCORE-A5d2x System on Module
• phyCORE-A5d2 Rugged Board
• USB cable
• Patch card

README FILE

The script exports GPIO pin 17.

The script checks if the PA17 directory exists. If it does, it prints out "GPIO Exported." If it doesn't, it prints out "GPIO Exported Failed."

The script sets the direction of the GPIO pin to "out".

The script sets the value of the GPIO pin to 1.

The script sleeps for 2 seconds.

The script sets the value of the GPIO pin to 0.

The script sleeps for 2 seconds.

The script reads the value of the GPIO pin and stores it in the gpio_value variable.

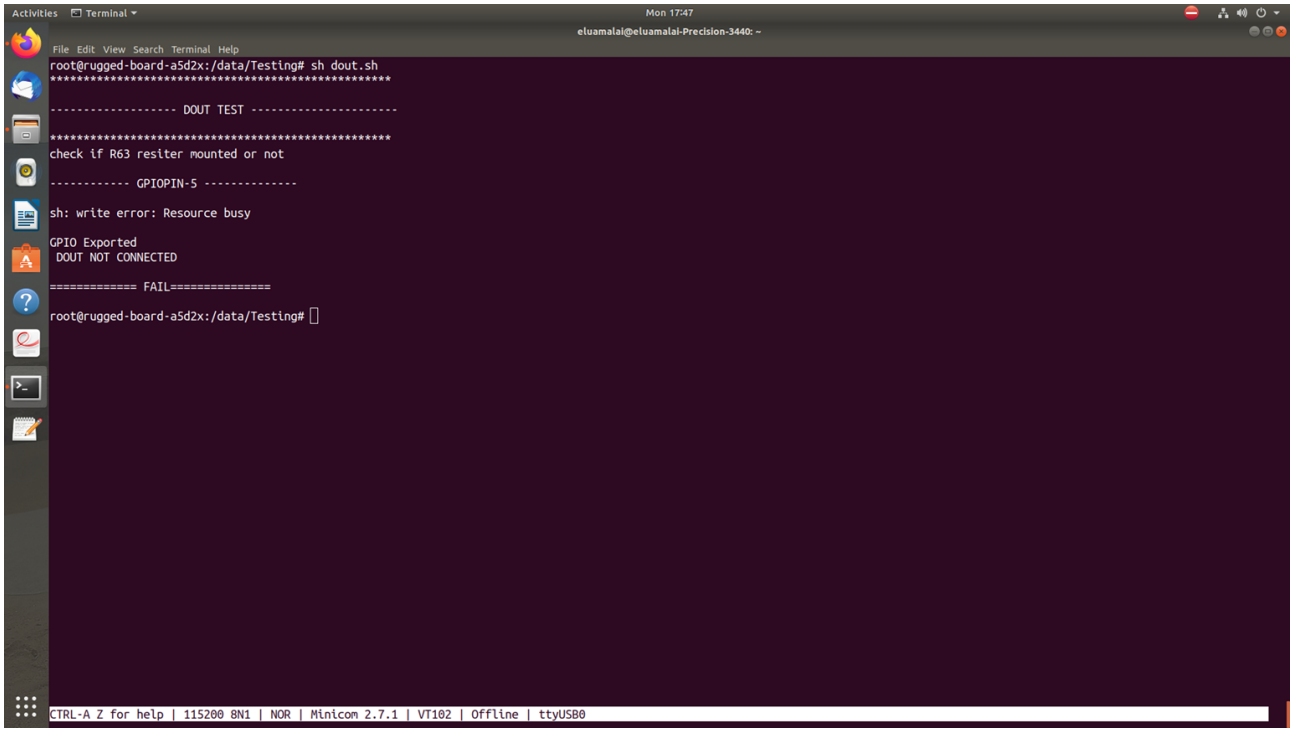
The script checks if the gpio_value is equal to 1. If it is, the script prints out "DOUT CONNECTED" and "PASS." If it isn't, the script prints out "DOUT NOT CONNECTED" and "FAIL."

TESTING SCRIPT

```
#! /bin/bash
echo  "*****************************************************"
echo
echo "------------------- DOUT TEST ----------------------"
echo
echo  "*****************************************************"
echo "check if R63 resiter mounted or not "
echo
echo "------------ GPIOPIN-5 --------------"
echo
echo 17 > /sys/class/gpio/export
echo
if [ -d "/sys/class/gpio/PA17" ]
then
   echo "GPIO Exported"
else
   echo "GPIO Exported Failed"
fi
echo out > /sys/class/gpio/PA17/direction
echo 1 > /sys/class/gpio/PA17/value
sleep 2
```

```
echo 0 > /sys/class/gpio/PA17/value
sleep 2
gpio_value=$(cat /sys/class/gpio/PA17/value)
if [ "$gpio_value" == 1 ]
then
    echo " DOUT CONNECTED "
    echo
    echo "============= PASS ==============="
    echo
else
    echo " DOUT NOT CONNECTED "
    echo
    echo "============= FAIL==============="
    echo
fi
```

OUTPUT LOG



11 PWM (PULSE WIDTH MODULATION)

DESCRIPTION

The Pulse Width Modulation Controller (PWM) generates output pulses on 4 channels independently according to parameters defined per channel.

Each channel controls two complementary square output waveforms.

README FILE

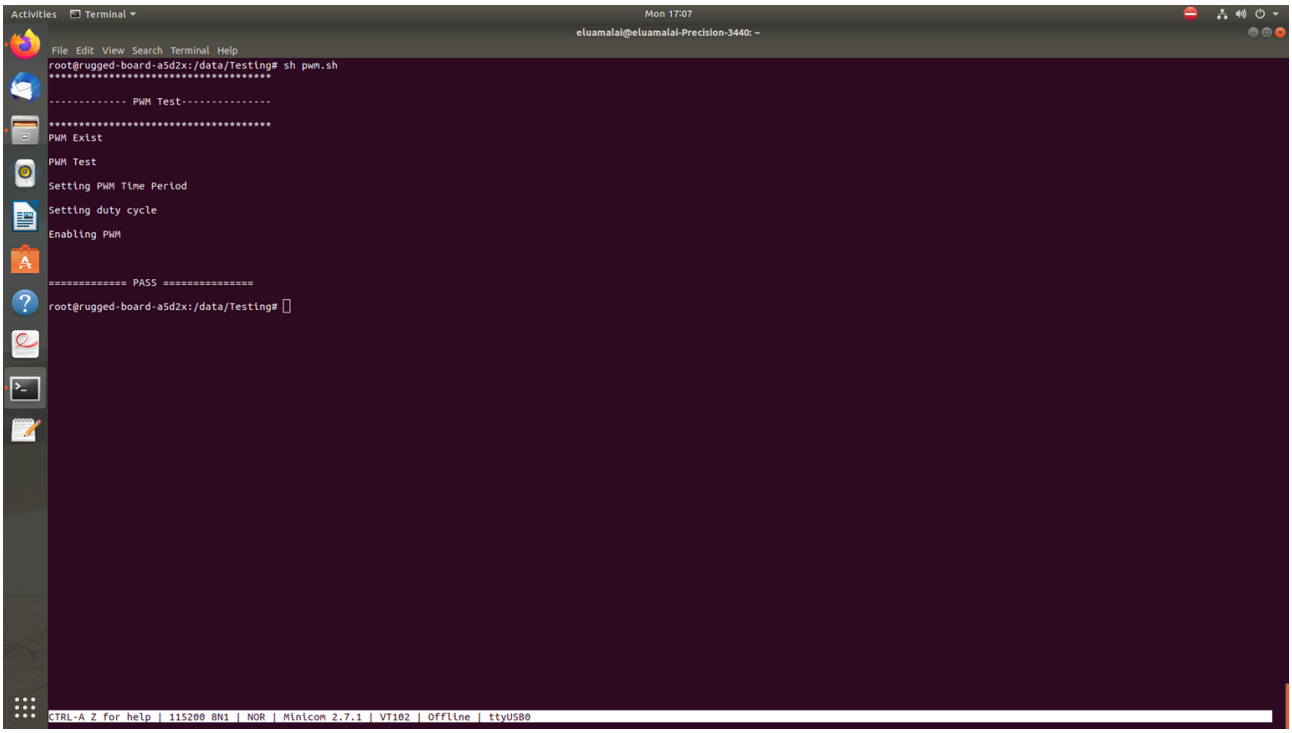
- checks whether a directory exists at "/sys/class/pwm/pwmchip0/" to determine whether the device has a PWM chip.
- If a PWM chip exists, the script proceeds with the test by exporting the PWM, setting the PWM time period to 50000, and setting the duty cycle to 2500.
- The script then waits for 3 seconds, changes the duty cycle to 1200, and enables the PWM.
- After waiting for another 3 seconds, the script checks the enable status of the PWM to determine whether the test has passed or failed.
- If a PWM chip does not exist, the script prints an error message and fails the test.

TESTING SCRIPT

```
#!/bin/bash
echo "**************************************"
echo
echo "------------- PWM Test--------------"
echo
echo "**************************************"
if [ -d "/sys/class/pwm/pwmchip0/" ]
then
   echo "PWM Exist"
   echo
   echo 0 > /sys/class/pwm/pwmchip0/export
   echo "PWM Test"
   echo
   sleep 3
   echo "Setting PWM Time Period"
   echo 50000 > /sys/class/pwm/pwmchip0/pwm0/period
   echo
   echo "Setting duty cycle"
   echo 2500 > /sys/class/pwm/pwmchip0/pwm0/duty_cycle
   echo
   sleep 3
   echo 1200 > /sys/class/pwm/pwmchip0/pwm0/duty_cycle
   echo "Enabling PWM"
echo "Enabling PWM"
   echo 0 > /sys/class/pwm/pwmchip0/pwm0/enable
#    cat /sys/kernel/debug/pwm
   echo
   echo 1 > /sys/class/pwm/pwmchip0/pwm0/enable
```

```
    sleep 3
    echo
    status=$(cat /sys/class/pwm/pwmchip0/pwm0/enable)
    if [ "$status" == "1" ]
    then
        echo
        echo "============= PASS ==============="
        echo
    else
        echo
        echo "============= FAIL ==============="
        echo
    fi
else
    echo "No PWM chip found"
    echo "============= FAIL ==============="
fi
```

OUTPUT LOG



12 RTC (REAL TIME CLOCK)

DESCRIPTION

The Real-time Clock (RTC) peripheral is designed for very low power consumption. For optimal functionality, the RTC requires an accurate external 32.768 kHz clock, which can be provided by a crystal oscillator.
It combines a complete time-of-day clock with alarm and a Gregorian or Persian calendar, complemented by a programmable periodic interrupt.

README FILE

- This is a shell script that performs the following tasks:Checks if the RTC (Real-Time Clock) device is available in the system.
- Restarts the chronyd service.
- Sets the timezone to Asia/Kolkata.
- Prints the current date and time.
- Performs a simple test by comparing the output of the "date" command with itself.
- The script is commented out the line that restarts the chronyd service, so it won't actually restart the service if executed. Also, the if statement in the test doesn't seem to make sense, as it compares the variable with itself.
- Assuming that the missing comparison in the if statement was a typo, the script appears to be a basic system check script that verifies the RTC device and the system's timezone, and checks if the "date" command is working correctly.

TESTING SCRIPT

```
#!/bin/sh
echo
echo "----------- RTC TEST -----------"
echo

if [ -e "/dev/rtc0" ]
then
   echo "RTC device found: /dev/rtc0"
else
   echo "No RTC device found"
fi
else
   echo "No RTC device found"
fi

echo
echo "Restarting chronyd service..."
#systemctl restart chronyd
echo
echo "Setting timezone to Asia/Calcutta..."
```

```
ln -sf /usr/share/zoneinfo/Asia/Kolkata /etc/localtime

echo
echo "Current date and time:"
date

echo
var=$(date)
if [ "$var" == "$var" ]
then
   echo "* PASS **"
else
   echo "* FAIL **"
fi
```

OUTPUT LOG