Complete Documentation: Temperature Sensor Linux Kernel Module

From Build to Execution - Step by Step Guide

This comprehensive documentation covers every step from building the project to successful execution, including troubleshooting and verification steps.

📋 Table of Contents

🛠️ Pre-Build Preparation

Step 1: System Requirements Check

First, verify your system meets all requirements:

```bash
# Check Linux kernel version
uname -r

# Check if you have root access
sudo whoami

# Verify GCC installation
gcc --version

# Check make utility
make --version
```

Expected Output Example:

```
5.15.0-91-generic
root
gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
GNU Make 4.3
```

Step 2: Install Required Dependencies

```bash
# Update package list
sudo apt update

# Install essential packages
sudo apt install -y build-essential linux-headers-$(uname -r) libncurses5-dev

# Verify installation
dpkg -l | grep linux-headers-$(uname -r)
dpkg -l | grep libncurses5-dev
```

Step 3: Directory Setup

```bash
# Create project directory
mkdir Temperature_Sensor_Project
cd Temperature_Sensor_Project

# Verify you have all three files:
# - Temperature_Sensor.c
# - tempsensor_ncurses.c
# - Makefile

ls -la
```

Expected Output:

```
total 20
drwxrwxr-x 2 user user 4096 Aug 14 10:30 .
drwxrwxr-x 3 user user 4096 Aug 14 10:29 ..
-rw-rw-r-- 1 user user  412 Aug 14 10:30 Makefile
-rw-rw-r-- 1 user user 3845 Aug 14 10:30 Temperature_Sensor.c
-rw-rw-r-- 1 user user 2834 Aug 14 10:30 tempsensor_ncurses.c
```

🏗️ Build Process

Step 1: Execute Build Command

```bash
make all
```

What happens during build:

1. **Kernel Module Compilation:**
   - Invokes kernel build system
   - Compiles `Temperature_Sensor.c` into `Temperature_Sensor.ko`
   - Creates additional files (.o, .mod, .symvers, etc.)
2. **User Application Compilation:**
   - Compiles `tempsensor_ncurses.c` with ncurses library
   - Creates executable `tempsensor_ui`

Expected Successful Output:

```bash
make -C /lib/modules/5.15.0-91-generic/build M=/home/user/Temperature_Sensor_Project modules
make[1]: Entering directory '/usr/src/linux-headers-5.15.0-91-generic'
  CC [M]  /home/user/Temperature_Sensor_Project/Temperature_Sensor.o
  MODPOST /home/user/Temperature_Sensor_Project/Module.symvers
  CC [M]  /home/user/Temperature_Sensor_Project/Temperature_Sensor.mod.o
  LD [M]  /home/user/Temperature_Sensor_Project/Temperature_Sensor.ko
  BTF [M] /home/user/Temperature_Sensor_Project/Temperature_Sensor.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.15.0-91-generic'
gcc -o tempsensor_ui tempsensor_ncurses.c -lncurses
```

Step 2: Verify Build Success

```bash
# Check generated files
ls -la

# Verify kernel module
file Temperature_Sensor.ko

# Verify user application
file tempsensor_ui
./tempsensor_ui --help 2>/dev/null; echo "Exit code: $?"
```

Expected Output:

```
-rw-rw-r-- 1 user user  1234 Aug 14 10:35 Temperature_Sensor.ko
-rwxrwxr-x 1 user user  8760 Aug 14 10:35 tempsensor_ui

Temperature_Sensor.ko: ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), BuildID[sha1]=..., not stripped
tempsensor_ui: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=..., for GNU/Linux 3.2.0, not stripped
```

📦 Module Loading

Step 1: Load the Kernel Module

```bash
sudo insmod Temperature_Sensor.ko
```

Step 2: Verify Module Loading

```bash
# Check if module is loaded
lsmod | grep Temperature_Sensor

# Check kernel messages
dmesg | tail -5

# Check module details
modinfo Temperature_Sensor.ko
```

Expected Output:

```bash
# lsmod output
Temperature_Sensor    16384  0

# dmesg output
[12345.678901] Temperature Sensor: Module loaded with major number 240

# modinfo output
filename:       /home/user/Temperature_Sensor_Project/Temperature_Sensor.ko
version:        1.0
description:    Temperature Sensor Simulator with Weather Symbols
author:         Tharun_Kumar_M
license:        GPL
srcversion:     ABC123DEF456GHI789
depends:
retpoline:      Y
name:           Temperature_Sensor
vermagic:       5.15.0-91-generic SMP mod_unload modversions
```

Step 3: Note the Major Number

```bash
# Extract major number from dmesg
MAJOR_NUM=$(dmesg | grep "Temperature Sensor: Module loaded" | tail -1 | sed 's/.*major number \([0-9]*\).*/\1/')
echo "Major Number: $MAJOR_NUM"
```

🔧 Device Setup

Step 1: Check for Auto-Created Device

```bash
# Check if device was auto-created
ls -l /dev/Temperature_Sensor

# Check device class
ls -l /sys/class/temp/
```

Step 2: Manual Device Creation (if needed)

If the device wasn't auto-created:

```bash
# Get major number from previous step
MAJOR_NUM=$(dmesg | grep "Temperature Sensor: Module loaded" | tail -1 | sed 's/.*major number \([0-9]*\).*/\1/')

# Create device node manually
sudo mknod /dev/Temperature_Sensor c $MAJOR_NUM 0

# Set proper permissions
sudo chmod 666 /dev/Temperature_Sensor

# Verify creation
ls -l /dev/Temperature_Sensor
```

Expected Output:

```
crw-rw-rw- 1 root root 240, 0 Aug 14 10:40 /dev/Temperature_Sensor
```

Step 3: Test Device Communication

```bash
# Test basic read operation
cat /dev/Temperature_Sensor
# Should output something like: "23 Warm"

# Test multiple reads
for i in {1..3}; do echo "Read $i:"; cat /dev/Temperature_Sensor; done
```

Expected Output:

```
Read 1:
15 Cloudy
Read 2:
28 Sunny
Read 3:
-5 Freezing
```

🚀 Application Execution

Step 1: Pre-Execution Checks

```bash
# Verify terminal supports colors
tput colors

# Check terminal size
tput cols
tput lines

# Test ncurses
echo "Testing ncurses support..."
```

Step 2: Run the Application

```bash
# Standard execution
./tempsensor_ui

# Alternative: Run with specific terminal settings
TERM=xterm-256color ./tempsensor_ui

# If permission issues
sudo ./tempsensor_ui
```

Step 3: Application Interface

When running successfully, you'll see:

```
==============================
   Temperature Monitor
==============================

Temperature: 25°C
Condition  : Sunny

==============================

Press Ctrl+C to exit...
```

Color Coding:

- **Red**: Extreme Hot (≥35°C)
- **Yellow**: Sunny (25-34°C)
- **White**: Warm (18-24°C)
- **Cyan**: Cloudy (10-17°C)
- **Green**: Cool (5-9°C)
- **Blue**: Cold (0-4°C)
- **Magenta**: Freezing (<0°C)

Step 4: Normal Operation

- Temperature updates every 2 seconds
- Colors change based on temperature range
- Press Ctrl+C to exit cleanly

✅ Verification Steps

System-Level Verification

```bash
# 1. Check kernel module status
lsmod | grep Temperature_Sensor

# 2. Verify device node exists
ls -l /dev/Temperature_Sensor

# 3. Check device major/minor numbers
ls -l /dev/Temperature_Sensor | awk '{print "Major:", int($5/256), "Minor:", $5%256}'

# 4. Verify module information
cat /proc/devices | grep Temperature

# 5. Check system resources
cat /proc/modules | grep Temperature_Sensor
```

Functional Verification

```bash
# 1. Test device read functionality
echo "Testing device reads:"
for i in {1..5}; do
  echo "Read $i: $(cat /dev/Temperature_Sensor)"
  sleep 1
done

# 2. Test application startup
timeout 10 ./tempsensor_ui &
PID=$!
sleep 5
kill $PID 2>/dev/null
echo "Application test completed"

# 3. Verify memory usage
ps aux | grep tempsensor_ui
```

Performance Verification

```bash
# 1. Check module memory usage
cat /proc/slabinfo | grep -i temp

# 2. Monitor system logs
tail -f /var/log/kern.log &
LOG_PID=$!
sleep 10
kill $LOG_PID

# 3. Test under load
echo "Stress testing device..."
for i in {1..100}; do cat /dev/Temperature_Sensor >/dev/null; done
echo "Stress test completed"
```

🐛 Troubleshooting

Build Issues

Issue: "No rule to make target"

```bash
# Solution:
make clean
make all

# If still failing:
which gcc
which make
sudo apt install --reinstall build-essential
```

Issue: "fatal error: linux/module.h: No such file"

```bash
# Solution:
sudo apt install linux-headers-$(uname -r)
ls /lib/modules/$(uname -r)/build
```

Issue: "cannot find -lncurses"

```bash
# Solution:
sudo apt install libncurses5-dev libncurses5-dev-dbg
pkg-config --libs ncurses
```

Module Loading Issues

Issue: "Operation not permitted"

```bash
# Check secure boot status
mokutil --sb-state

# If secure boot is enabled:
# Option 1: Disable secure boot in BIOS
# Option 2: Sign the module (advanced)
sudo modprobe configs
```

Issue: "Invalid module format"

```bash
# Rebuild module for current kernel
make clean
make all
uname -r
ls /lib/modules/$(uname -r)/build
```

Issue: "Module already exists"

```bash
# Remove existing module
sudo rmmod Temperature_Sensor
# Then reload
sudo insmod Temperature_Sensor.ko
```

Device Issues

Issue: "No such device"

```bash
# Check if module is loaded
lsmod | grep Temperature_Sensor

# Check major number
dmesg | grep "Temperature Sensor"

# Manually create device
MAJOR_NUM=240  # Use your actual major number
sudo mknod /dev/Temperature_Sensor c $MAJOR_NUM 0
sudo chmod 666 /dev/Temperature_Sensor
```

Issue: "Permission denied"

```bash
# Fix permissions
sudo chmod 666 /dev/Temperature_Sensor

# Or run as root
sudo ./tempsensor_ui

# Check SELinux (if applicable)
getenforce
```

Application Issues

Issue: "Failed to open device"

```bash
# Verify device exists
ls -l /dev/Temperature_Sensor

# Test device manually
cat /dev/Temperature_Sensor

# Check permissions
stat /dev/Temperature_Sensor
```

Issue: "Terminal display problems"

```bash
# Reset terminal
reset
clear

# Check terminal capabilities
echo $TERM
tput colors

# Try different terminal
TERM=xterm-256color ./tempsensor_ui
```

Issue: "Segmentation fault"

```bash
# Run with debugger
gdb ./tempsensor_ui
(gdb) run
(gdb) backtrace

# Check for memory issues
valgrind ./tempsensor_ui
```

🧹 Cleanup Process

Step 1: Stop Application

```bash
# If running in background
pkill -f tempsensor_ui

# Or use Ctrl+C if running in foreground
```

Step 2: Remove Module

```bash
# Check if module is in use
lsmod | grep Temperature_Sensor

# Remove module
sudo rmmod Temperature_Sensor

# Verify removal
lsmod | grep Temperature_Sensor
```

Step 3: Clean Device Node

```bash
# Remove device node (if manually created)
sudo rm -f /dev/Temperature_Sensor

# Verify removal
ls -l /dev/Temperature_Sensor
```

Step 4: Clean Build Files

```bash
# Remove compiled files
make clean

# Verify cleanup
ls -la
```

Step 5: Complete Cleanup Verification

```bash
# Check no traces remain
lsmod | grep Temperature
ls -l /dev/Temperature*
dmesg | tail -5
```

📊 Complete Execution Log Example

Here's what a successful complete execution looks like:

```
bash

user@ubuntu:~/Temperature_Sensor_Project$ make all
make -C /lib/modules/5.15.0-91-generic/build M=/home/user/Temperature_Sensor_Project modules
make[1]: Entering directory '/usr/src/linux-headers-5.15.0-91-generic'
  CC [M]  /home/user/Temperature_Sensor_Project/Temperature_Sensor.o
  MODPOST /home/user/Temperature_Sensor_Project/Module.symvers
  CC [M]  /home/user/Temperature_Sensor_Project/Temperature_Sensor.mod.o
  LD [M]  /home/user/Temperature_Sensor_Project/Temperature_Sensor.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.15.0-91-generic'
gcc -o tempsensor_ui tempsensor_ncurses.c -lncurses

user@ubuntu:~/Temperature_Sensor_Project$ sudo insmod Temperature_Sensor.ko

user@ubuntu:~/Temperature_Sensor_Project$ dmesg | tail -1
[12345.678901] Temperature Sensor: Module loaded with major number 240

user@ubuntu:~/Temperature_Sensor_Project$ ls -l /dev/Temperature_Sensor
crw-rw-rw- 1 root root 240, 0 Aug 14 10:45 /dev/Temperature_Sensor

user@ubuntu:~/Temperature_Sensor_Project$ cat /dev/Temperature_Sensor
32 Sunny

user@ubuntu:~/Temperature_Sensor_Project$ ./tempsensor_ui
[Colorful interface displays with temperature readings updating every 2 seconds]
^C

user@ubuntu:~/Temperature_Sensor_Project$ sudo rmmod Temperature_Sensor

user@ubuntu:~/Temperature_Sensor_Project$ dmesg | tail -1
[12346.789012] Temperature Sensor: Module unloaded

user@ubuntu:~/Temperature_Sensor_Project$ make clean
make -C /lib/modules/5.15.0-91-generic/build M=/home/user/Temperature_Sensor_Project clean
make[1]: Entering directory '/usr/src/linux-headers-5.15.0-91-generic'
  CLEAN   /home/user/Temperature_Sensor_Project/Module.symvers
make[1]: Leaving directory '/usr/src/linux-headers-5.15.0-91-generic'
rm -f tempsensor_ui
```

📞 Support and Additional Help

If you encounter issues not covered in this documentation:

1. **Check system logs**: `journalctl -f` or `tail -f /var/log/kern.log`
2. **Review dmesg output**: `dmesg | grep -i error`
3. **Verify system resources**: `free -h`, `df -h`
4. **Test in safe mode**: Boot with minimal kernel modules
5. **Create issue report**: Include full error messages and system information

System Information Collection:

```bash
uname -a > system_info.txt
lsb_release -a >> system_info.txt
gcc --version >> system_info.txt
make --version >> system_info.txt
dmesg | tail -20 >> system_info.txt
```

This comprehensive documentation should guide anyone through the complete process from building to execution successfully.