UART Tools

Comprehensive Reference Guide

Serial Communication Toolkit for Linux Embedded Systems

Author: Kalpesh Solanki · github.com/xploitoverload/UART-Tools · MIT License

1. Overview

The UART Tools is a professional-grade, open-source toolkit of Bash shell utilities for embedded Linux developers working with serial communication interfaces. Every script is focused on a single task, hardened with strict mode (set -euo pipefail), and designed to compose cleanly with other shell tools.

1.1 Design Principles

- Single Responsibility — each script does one thing well
- Composability — tools connect via pipes and environment variables
- Fail-Fast — strict mode catches errors before they cascade
- Idempotency — operations are safe to repeat
- Minimal Dependencies — only POSIX/GNU coreutils plus optional lrzsz

1.2 Toolkit at a Glance

Script	Purpose	Key Dependency
time-sync.sh	Synchronise device clock over UART	stty, date
file-transfer.sh	Send/receive files via XMODEM/YMODEM/ZMODEM/raw	lrzsz (sz/rz)
rce.sh	Execute commands or scripts on a remote console	stty, read, timeout
fw-update.sh	Backup → flash → verify → reboot firmware workflow	file-transfer.sh
logger.sh	Capture, filter, colourize and analyse UART traffic	stty, grep, awk
AT.sh	Send AT commands to modems, GPS, ESP32/8266	stty, custom conf
menu.sh	Interactive TUI launcher for all of the above	All scripts

1.3 Supported Hardware

Category	Examples
Single-board computers	Raspberry Pi, BeagleBone
Microcontrollers	Arduino (AVR), STM32, ESP32/ESP8266
Industrial / automation	PLCs, SCADA interfaces, RS-485 gateways
Networking gear	Routers, managed switches with serial console
Cellular / GPS modems	SIM800, SIM7600, Quectel EC21/BG95, u-blox
Any UART/RS-232 device	RS-232↔USB adapters (FTDI, CP210x, CH340)

2. Installation

2.1 System Requirements

Requirement	Minimum	Notes
Operating System	Linux kernel 2.6+	Ubuntu, Debian, Fedora, Kali, Arch, Alpine
Shell	Bash 4.0+	Needed for associative arrays
Core utilities	GNU coreutils	stty, date, timeout, read, fuser
File transfer (optional)	lrzsz	Required for XMODEM/YMODEM/ZMODEM
Serial port access	dialout group	Or run with sudo

2.2 Clone & Install

```
git clone https://github.com/xploitoverload/UART-Tools.git
cd UART-Tools
chmod +x *.sh
```

2.3 Install Dependencies

Debian / Ubuntu / Kali

```
sudo apt-get update && sudo apt-get install -y bash coreutils psmisc lrzsz
```

RHEL / CentOS / Fedora

```
sudo dnf install -y bash coreutils psmisc lrzsz
```

Arch Linux

```
sudo pacman -S bash coreutils psmisc lrzsz
```

Alpine Linux

```
sudo apk add bash coreutils lrzsz
```

2.4 Serial Port Permissions

```
sudo usermod -a -G dialout $USER
newgrp dialout          # Or logout/login

# Verify
groups | grep dialout
ls -l /dev/ttyUSB0    # crw-rw---- root dialout
```

 Note: If you see "permission denied" when opening a port, the dialout group fix above is almost always the solution. You should not need to use sudo after this change.

2.5 Add to PATH (Optional)

```
# Per-user
mkdir -p ~/.local/bin && cp *.sh ~/.local/bin/
echo 'export PATH="$HOME/.local/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc

# System-wide
sudo cp *.sh /usr/local/bin/
```

3. Configuration

3.1 Environment Variables (Global)

Variable	Default	Description
UART_PORT	/dev/ttyUSB0	Serial port device path
BAUD_UART	115200	Baud rate
VERBOSE	0	Set 1 for verbose/debug logging
LOG_FILE	/var/log/uart.log	Log output file
TIMEOUT	per-script	Default operation timeout (seconds)

3.2 Config Files

/etc/UART-Tools.conf (global)

```
UART_PORT=/dev/ttyUSB0
BAUD_UART=115200
VERBOSE=0
LOG_FILE=/var/log/uart.log
DEFAULT_TIMEOUT=10
TRANSFER_TIMEOUT=60
FIRMWARE_TIMEOUT=300
```

~/.UART-Tools.conf (per-user override)

```
UART_PORT=/dev/ttyACM0
VERBOSE=1
```

3.3 Precedence (highest → lowest)

1. Command-line arguments (-p /dev/ttyACM0 -b 9600)
2. Inline env vars (UART_PORT=/dev/ttyACM0 ./time-sync.sh)
3. User config (~/.UART-Tools.conf)
4. Global config (/etc/UART-Tools.conf)
5. Built-in script defaults

4. time-sync.sh — Time Synchronisation

Synchronises the clock on a connected embedded device by sending a formatted date command over the serial console. Useful when the target has no RTC or NTP access.

4.1 Synopsis

```
time-sync.sh [OPTIONS]
```

4.2 Options

Flag	Default	Description
-p PORT	/dev/ttyUSB0	Serial port device
-b BAUD	115200	Baud rate
-t TIMEOUT	5	Timeout in seconds
-v	off	Verbose output
-h	—	Display help

4.3 Examples

```
# Basic usage
./time-sync.sh

# Custom port and baud
./time-sync.sh -p /dev/ttyACM0 -b 9600

# Verbose with longer timeout
./time-sync.sh -v -t 10

# Inline env override
UART_PORT=/dev/ttyUSB1 VERBOSE=1 ./time-sync.sh
```

4.4 Exit Codes

Code	Meaning
0	Success
1	Port not found/inaccessible
2	Configuration error
3	Timeout

4.5 Implementation Notes

- Configures port with stty before writing
- Sends: date -s "YYYY-MM-DD HH:MM:SS"
- Uses fuser to detect processes holding the port

- Registers cleanup trap on EXIT / INT / TERM

5. file-transfer.sh — File Transfer

Transfers files between the host and a serial device using XMODEM, YMODEM, ZMODEM, or raw binary. Protocol can be selected to trade speed against error-correction capability.

5.1 Synopsis

```
file-transfer.sh [OPTIONS] COMMAND FILE
```

5.2 Commands

Command	Description
send FILE	Send a file from host to device
receive FILE	Receive a file from device to host

5.3 Options

Flag	Default	Description
-p PORT	/dev/ttyUSB0	Serial port
-b BAUD	115200	Baud rate
-P PROTOCOL	zmodem	xmodem | ymodem | zmodem | raw
-t TIMEOUT	60	Transfer timeout (seconds)
-v	off	Verbose output

5.4 Protocol Comparison

Protocol	Speed	Error Correction	Resume	Best For
ZMODEM	Fast	CRC-32	Yes	Large files, unreliable links
YMODEM	Medium	CRC-16	No	Batch transfers
XMODEM	Slow	CRC/Checksum	No	Legacy or minimal devices
RAW	Fastest	None	No	Reliable links, direct binary dump

5.5 Examples

```
# Send firmware (ZMODEM default)
./file-transfer.sh send firmware.bin

# Receive a log file
./file-transfer.sh receive device.log

# XMODEM for a legacy bootloader
./file-transfer.sh -P xmodem send u-boot.bin

# Raw binary, 9600 baud
```

```
./file-transfer.sh -P raw -b 9600 send bootloader.bin

# Verbose, 5-minute timeout for large firmware
./file-transfer.sh -v -t 300 send big_firmware.bin
```

 Tip: ZMODEM is the recommended default. It auto-negotiates, supports resume on failure, and provides CRC-32 integrity checking. Drop to XMODEM only when the target bootloader requires it.

6. rce.sh — Remote Command Execution

Sends commands to a device via its serial console and reads back the response. Supports single commands, batch script execution, and an interactive shell session.

6.1 Synopsis

```
rce.sh [OPTIONS] COMMAND [ARGS]
```

6.2 Commands

Command	Description
exec "CMD"	Execute a single command on the device
script FILE	Run commands from a script file
shell	Interactive serial console session
monitor	Monitor output without sending input

6.3 Options

Flag	Default	Description
-p PORT	/dev/ttyUSB0	Serial port
-b BAUD	115200	Baud rate
-t TIMEOUT	10	Response timeout (seconds)
-n	off	No-wait: send and do not wait for response
-i	off	Interactive mode (alias for shell)
-v	off	Verbose output

6.4 Script File Format

```
# my_script.txt
cat /proc/cpuinfo
df -h
@sleep 2        # Pause 2 seconds
@wait 15        # Change response timeout to 15s
ip addr show
```

6.5 Examples

```
# Single command
./rce.sh exec "cat /etc/os-release"

# Fire-and-forget reboot
./rce.sh -n exec "reboot"

# Run a config script
```

```
./rce.sh script configure_device.txt

# Interactive session
./rce.sh shell

# Monitor only
./rce.sh monitor
```

7. fw-update.sh — Firmware Update

Manages the full firmware update lifecycle: bootloader entry, backup, flash, MD5 verification, and reboot. Backup and verify are enabled by default.

7.1 Synopsis

```
fw-update.sh [OPTIONS] COMMAND [ARGS]
```

7.2 Commands

Command	Description
update FILE [BACKUP]	Full workflow: backup → flash → verify → reboot
flash FILE	Flash firmware only
backup FILE	Read and save current firmware
verify FILE	Compare flashed firmware via MD5
info	Print firmware version and device info
bootloader	Enter bootloader mode
reboot	Send reboot and monitor boot output

7.3 Options

Flag	Default	Description
-p PORT	/dev/ttyUSB0	Serial port
-b BAUD	115200	Baud rate
-t TIMEOUT	300	Operation timeout (seconds)
--no-verify	off	Skip MD5 verification after flashing
--no-backup	off	Skip firmware backup before flashing
-v	off	Verbose output

 Warning: Skipping --no-verify is strongly discouraged. A partial or corrupt flash can brick the device. Only use this flag when the device does not support readback and you have a hardware programmer as backup.

7.4 Examples

```
# Full safe update
./fw-update.sh update new_firmware.bin

# Backup before doing anything
./fw-update.sh backup firmware_backup_$(date +%Y%m%d).bin

# Verify an already-flashed image
```

```
./fw-update.sh verify expected_firmware.bin

# Check firmware version
./fw-update.sh info
```

8. logger.sh — Traffic Capture & Analysis

Captures raw UART traffic, applies automatic colour-coding for log levels, filters by regex, generates statistics, and analyses captured log files.

8.1 Synopsis

```
logger.sh [OPTIONS] COMMAND [ARGS]
```

8.2 Commands

Command	Description
monitor	Continuous live monitoring with colourized output
capture SECONDS	Capture for a fixed number of seconds
analyze FILE	Analyse a previously captured log file
stats [INTERVAL]	Live statistics, refreshed every INTERVAL seconds
tail	Follow UART output (like tail -f)
grep PATTERN	Filter live output to lines matching PATTERN

8.3 Options

Flag	Default	Description
-p PORT	/dev/ttyUSB0	Serial port
-b BAUD	115200	Baud rate
-o FILE	auto	Save to FILE
-f PATTERN	none	Regex filter — only matching lines shown
-x	off	Hex dump mode
-n	off	Suppress timestamps
-s	off	Enable per-session statistics

8.4 Colour Coding

Colour	Triggered By
Red	ERROR, error, fail, fatal, panic
Yellow	WARN, warning, alert
Green	success, ok, done, complete
Cyan	info, debug, trace

8.5 Examples

```
# Live monitoring
./logger.sh monitor

# Capture 60 seconds
./logger.sh capture 60 -o session.log

# Filter errors only
./logger.sh -f "ERROR|WARN" monitor

# Hex dump for binary protocols
./logger.sh -x monitor

# Post-session analysis
./logger.sh analyze session.log
```

9. AT.sh — AT Command Utility

A comprehensive interface for AT-command-capable devices: cellular modems (SIM800, SIM7600, Quectel), GPS/GNSS modules, and Wi-Fi chips (ESP32/ESP8266). Supports raw command injection, structured queries, SMS, GPS, and a custom command registry.

9.1 Synopsis

```
AT.sh [OPTIONS] COMMAND [ARGS]
```

9.2 Commands

Command	Description
test	Send AT — verify device responds with OK
info	Retrieve manufacturer, model, firmware, IMEI
send COMMAND	Send any raw AT command
list-commands	List all built-in and custom commands
add-custom NAME TYPE CMD	Register a custom command (TYPE: at | bash)
remove-custom NAME	Remove a registered custom command
list-custom	List all custom commands
sms-list [STATUS]	List SMS (ALL | RECEIVED | UNREAD | SENT)
sms-send NUMBER TEXT	Send an SMS
gps-on / gps-off	Enable/disable GPS engine
gps-info	Current fix: lat, lon, altitude, time
network-status	Network registration state
signal-quality	RSSI and BER (AT+CSQ)
operator	Current operator name and technology
list-operators	Scan all available network operators

9.3 Options

Flag	Default	Description
-p PORT	/dev/ttyUSB0	Serial port
-b BAUD	4800	Baud rate (modems often default to 4800–9600)
--data-bits N	8	Data bits: 7 or 8
--stop-bits N	1	Stop bits: 1 or 2
--parity TYPE	none	none | even | odd
--flow-control	off	Enable RTS/CTS hardware flow control
-t TIMEOUT	2	Response timeout (seconds)

Flag	Default	Description
-w DELAY	1000	Wait time after sending command (ms)
--line-ending	crlf	crlf | lf
-v, --verbose	off	Show raw send/receive bytes

9.4 Environment Variables

Variable	Default	Description
UART_PORT	/dev/ttyUSB0	Serial port
BAUD_UART	4800	Baud rate
TIMEOUT	2	Command timeout (s)
RESPONSE_WAIT	1000	Response wait (ms)
LINE_ENDING	crlf	CR+LF or LF only
VERBOSE	0	1 for verbose mode
CUSTOM_COMMANDS_FILE	~/.uart-tools/custom_commands.conf	Custom command store
LOG_FILE	/var/log/uart.log	Log file path

9.5 AT Command Reference

Basic / Connection

Command	Function
AT	Test — device should respond OK
ATI	Device identification string
ATE0 / ATE1	Disable / enable echo
ATZ	Reset to stored profile
AT&F	Factory reset
AT&W	Write settings to non-volatile storage

Device Identification

Command	Returns
AT+GMI	Manufacturer name
AT+GMM	Model name
AT+GMR	Firmware / revision
AT+GSN	Serial number (IMEI)
AT+GCAP	Capability list

Network & Cellular

Command	Returns
AT+CPIN?	SIM status (READY / PIN / PUK)
AT+CSQ	Signal quality: RSSI, BER
AT+CREG?	Network registration status
AT+COPS?	Current operator
AT+COPS=?	Scan all available operators

SMS

Command	Function
AT+CMGF=1	Set text mode (1=text, 0=PDU)
AT+CMGL="ALL"	List all messages
AT+CMGD=<idx>	Delete message at index
AT+CMGS="+1234567890"	Send SMS (follow with text then Ctrl+Z)

GPS / GNSS

Command	Function
AT+CGPS=1	Enable GPS engine
AT+CGPS=0	Disable GPS engine
AT+CGPSINFO	Get fix: lat, lon, altitude, time

Wi-Fi (ESP32 / ESP8266 AT firmware)

Command	Function
AT+CWMODE=1	Set Station mode
AT+CWLAP	Scan nearby Wi-Fi networks
AT+CWJAP="SSID","pass"	Connect to a network

9.6 Custom Commands

Register shortcuts for frequently used AT sequences or local shell commands. Persisted in ~/.uart-tools/custom_commands.conf.

```
# Register an AT shortcut
./AT.sh add-custom "check_signal" "at" "AT+CSQ"

# Register a bash shortcut
./AT.sh add-custom "get_epoch" "bash" "date +%s"

# List custom commands
./AT.sh list-custom

# Run by name
./AT.sh send check_signal
```

```
# Remove
./AT.sh remove-custom check_signal
```

Config File Format (~/.uart-tools/custom_commands.conf)

```
# name|type|command
check_battery|at|AT+CBC
get_epoch|bash|date +%s
imei|at|AT+GSN
```

9.7 Usage Examples

```
# Test connection
./AT.sh test

# Full device info
./AT.sh info

# Signal quality
./AT.sh signal-quality

# SMS operations
./AT.sh sms-list UNREAD
./AT.sh sms-send "+91XXXXXXXXXX" "Hello from UART-Tools"

# GPS workflow
./AT.sh gps-on
./AT.sh gps-info
./AT.sh gps-off

# Baud rate auto-discovery
for baud in 4800 9600 19200 115200; do
  ./AT.sh -b $baud test && echo "Found at $baud" && break
done

# Environment variable override
UART_PORT=/dev/ttyACM0 BAUD_UART=115200 ./AT.sh test
```

9.8 Exit Codes

Code	Meaning
0	Success
1	Port error
2	Invalid command/syntax
3	Timeout
4	Device returned ERROR
5	Command not supported

10. file-editor.sh — File Editor

Line-by-line file editing utility with append, prepend, insert, replace, delete, find-replace, and automatic backup/undo. Works on any plain-text file: config files, scripts, logs, etc. Can be used non-interactively (single command) or through a full interactive TUI menu.

10.1 Synopsis

```
file-editor.sh [OPTIONS] [COMMAND [ARGS...]]
```

10.2 Options

Flag	Description
-f FILE	Target file to edit
-v	Verbose output
-h	Show help

10.3 Commands (Non-Interactive)

Command	Description
view	View file with line numbers
view START END	View lines START to END
append "text"	Append a line to end of file
prepend "text"	Prepend a line to start of file
insert-after N "text"	Insert new line after line N
insert-before N "text"	Insert new line before line N
replace N "text"	Replace line N with new text
delete N	Delete line N
delete-range S E	Delete lines S through E (inclusive)
append-to N "text"	Append text inline to end of line N
prepend-to N "text"	Prepend text inline to start of line N
find-replace OLD NEW	Global literal find-and-replace
find-replace-regex PAT REP	Global regex find-and-replace
undo	Restore previous backup (one-step undo)

10.4 Interactive Menu

Key	Action	Key	Action
s	Select target file	n	Create new file
v	View all lines	r	View line range

Key	Action	Key	Action
a	Append to end	p	Prepend to start
ia	Insert after line N	ib	Insert before line N
d	Delete line	dr	Delete range
rl	Replace line	al	Append to line N
pl	Prepend to line N	f	Find & replace (literal)
fr	Find & replace (regex)	u	Undo last change

10.5 Safety Features

- Automatic timestamped backup before every write operation
- One-step undo — restores previous backup instantly
- View command shows line numbers for precise targeting
- Verbose mode logs every operation with before/after context

10.6 Examples

```
# View file with line numbers
./file-editor.sh -f /etc/hosts view

# View lines 10-20 only
./file-editor.sh -f app.conf view 10 20

# Append a line to the end
./file-editor.sh -f /etc/hosts append "192.168.1.10 mydevice.local"

# Prepend a line to the start
./file-editor.sh -f config.txt prepend "# Auto-generated"

# Insert after line 5
./file-editor.sh -f config.txt insert-after 5 "debug=true"

# Replace line 15 entirely
./file-editor.sh -f sshd_config replace 15 "PermitRootLogin no"

# Delete a single line
./file-editor.sh -f logfile.txt delete 47

# Delete lines 10-25
./file-editor.sh -f logfile.txt delete-range 10 25

# Append text inline to line 1
./file-editor.sh -f /etc/hosts append-to 1 " # added"

# Literal find-and-replace
./file-editor.sh -f config.yaml find-replace "localhost" "192.168.1.100"

# Regex find-and-replace
./file-editor.sh -f app.conf find-replace-regex "port=[0-9]+" "port=8080"

# Undo last change
./file-editor.sh -f config.txt undo
```

```
# Launch interactive TUI
./file-editor.sh
```

10.7 Exit Codes

Code	Meaning
0	Success
1	File not found or not readable
2	Invalid arguments
3	Write permission denied
4	Backup failed

11. menu.sh — Interactive Menu

An interactive TUI that ties all tools together. Ideal for ad-hoc work without remembering flags.

```
./menu.sh

  [1] Time Synchronisation    - Sync device clock
  [2] File Transfer           - Send / receive files
  [3] Remote Command          - Execute commands
  [4] Firmware Update         - Flash new firmware
  [5] Traffic Monitor         - Log UART traffic
  [6] AT Command Control      - Modem / GPS / cellular
  [7] File Editor             - Line-by-line file editing
  [d] Device Discovery        - Detect serial ports
  [c] Configure               - Set port and baud rate
  [q] Quit
```

Device Discovery (d) scans /dev/ttyUSB* and /dev/ttyACM* and shows USB VID/PID where available. Configuration (c) sets UART_PORT and BAUD_UART for the session.

12. API Reference

15.1 Universal Exit Codes

Code	Meaning
0	Success
1	General error
2	Invalid arguments
3	Permission denied
4	Port not found
5	Timeout
6	Transfer error
7	Verification failed

15.2 Signal Handling

All scripts register cleanup traps for SIGINT (Ctrl+C), SIGTERM, and EXIT:

- Closes file descriptors to the serial port
- Kills background monitoring processes
- Restores stty / terminal state
- Flushes write buffers before exit

13. Troubleshooting

15.1 Permission Denied

```
[ERROR] Cannot access /dev/ttyUSB0 (permission denied)

sudo usermod -a -G dialout $USER
newgrp dialout
groups | grep dialout
```

15.2 Port Not Found

```
[ERROR] UART port /dev/ttyUSB0 not found
```

```
ls -l /dev/ttyUSB* /dev/ttyACM*
lsusb
dmesg | grep tty | tail -20
```

15.3 Garbled Output (Baud Mismatch)

```
for baud in 9600 19200 38400 57600 115200; do
    echo "Testing $baud..." && ./time-sync.sh -b $baud -v
    sleep 2
done
```

15.4 Transfer Failures

```
# Switch to a more robust protocol
./file-transfer.sh -P xmodem send file.bin

# Increase timeout
./file-transfer.sh -t 300 send large_firmware.bin

# Reduce baud rate
./file-transfer.sh -b 9600 send file.bin
```

15.5 Debug Mode

```
# Full Bash trace
bash -x ./time-sync.sh -v 2>&1 | tee debug.log

# Verbose flag
VERBOSE=1 ./rce.sh exec "uname -a"

# Capture raw bytes
./logger.sh -x monitor > traffic_dump.hex
```

14. Advanced Topics

15.1 systemd Service

```
# /etc/systemd/system/uart-monitor.service
[Unit]
Description=UART Monitoring Service
After=network.target
[Service]
Type=simple
User=uart
Group=dialout
Environment="UART_PORT=/dev/ttyUSB0"
ExecStart=/usr/local/bin/logger.sh monitor
Restart=always
RestartSec=10
[Install]
WantedBy=multi-user.target

sudo systemctl daemon-reload
sudo systemctl enable uart-monitor.service
sudo systemctl start uart-monitor.service
```

15.2 udev Rules

```
# /etc/udev/rules.d/99-UART-Tools.rules
ACTION=="add", SUBSYSTEM=="tty", ATTRS{idVendor}=="0403",\
  ATTRS{idProduct}=="6001",\
  RUN+="/usr/local/bin/time-sync.sh"

SUBSYSTEM=="tty", ATTRS{idVendor}=="1234",\
  ATTRS{idProduct}=="5678", MODE="0666", GROUP="dialout"
```

```
sudo udevadm control --reload-rules
sudo udevadm trigger
```

15.3 Docker

```
FROM ubuntu:22.04
RUN apt-get update && apt-get install -y bash coreutils lrzsz socat
COPY *.sh /usr/local/bin/
RUN chmod +x /usr/local/bin/*.sh
RUN useradd -m uart && usermod -a -G dialout uart
USER uart
ENTRYPOINT ["/usr/local/bin/menu.sh"]

docker build -t uart-tools . && docker run --device=/dev/ttyUSB0 -it uart-tools
```

15.4 Performance Reference

Protocol	Time (1 MB)	Throughput	Notes
ZMODEM	~12 s	~85 KB/s	CRC-32, compression, resume
YMODEM	~15 s	~68 KB/s	CRC-16, batch transfers
XMODEM	~25 s	~41 KB/s	Legacy compatibility

Protocol	Time (1 MB)	Throughput	Notes
RAW	~8 s	~128 KB/s	No error correction

15. Quick Reference Card

15.1 Common One-Liners

Task	Command
Test AT connection	./AT.sh test
Get device info	./AT.sh info
Signal quality	./AT.sh signal-quality
List unread SMS	./AT.sh sms-list UNREAD
Send SMS	./AT.sh sms-send "+91XXXXXXXXXX" "Hello"
Enable GPS	./AT.sh gps-on && ./AT.sh gps-info
Sync device time	./time-sync.sh
Send firmware	./file-transfer.sh send firmware.bin
Receive file	./file-transfer.sh receive output.bin
Execute command	./rce.sh exec "cat /proc/cpuinfo"
Run script	./rce.sh script setup.txt
Interactive shell	./rce.sh shell
Monitor UART	./logger.sh monitor
Capture 60s	./logger.sh capture 60 -o session.log
Filter errors	./logger.sh -f "ERROR" monitor
Full firmware update	./fw-update.sh update firmware.bin
Backup firmware	./fw-update.sh backup backup.bin
Open menu	./menu.sh

15.2 GPS Polling Loop

```
./AT.sh gps-on
while true; do
    ./AT.sh gps-info
    sleep 10
done
```

15.3 Baud Rate Auto-Detection

```
for baud in 4800 9600 19200 115200; do
    echo "Trying $baud..." && ./AT.sh -b $baud test && break
    sleep 1
done
```

15.4 Project Links

Resource	URL / Info
GitHub	https://github.com/xploitoverload/UART-Tools
Issues	https://github.com/xploitoverload/UART-Tools/issues
Author	Kalpesh Solanki · owner@kalpeshsolanki.me
License	MIT — free for commercial and personal use