# Controlling LED via Terminal Only

## The ”No-Code” DTR Hack on Arch Linux

Your Name

February 12, 2026

## How It Works (The DTR Trick)

Standard USB-to-Serial adapters have pins meant for communication flow control:

- **RX/TX:** For data (requires code/firmware).
- **DTR (Data Terminal Ready):** A hardware line we can toggle directly!
- **RTS (Request To Send):** Another toggleable line.

By connecting the LED to the **DTR pin** instead of a standard GPIO pin, we can control it purely via Linux device settings.

## Modified Wiring

Instead of a digital pin (like D13), connect the LED here:

1. **Anode (+):** Connect to the **DTR** (or RTS) pin on your board/adapter.
2. **Cathode (-):** Connect to Resistor $\rightarrow$ GND.

*Note: On many USB chips, DTR is "Active Low," meaning 0V when active and 3.3V/5V when inactive.*

## The Magic Command: stty

We use the standard Linux tool `stty` (Set Teletype) to manipulate the serial device file.
Open your Arch terminal:

### 1. Turn LED ON (Clear DTR)

```
$ stty -F /dev/ttyUSB0 hupcl
```

### 2. Turn LED OFF (Set DTR)

```
$ stty -F /dev/ttyUSB0 -hupcl
```

*Replace `/dev/ttyUSB0` with `/dev/ttyACM0` if using an Arduino Uno/Mega.

## Bash Scripting Example

Since these are just terminal commands, you can script a blink effect directly in Bash:

```
#!/bin/bash
DEVICE="/dev/ttyUSB0"

echo "Blinking␣LED␣on␣$DEVICE..."

while true; do
    # LED ON
    stty -F $DEVICE hupcl
    sleep 0.5

    # LED OFF
    stty -F $DEVICE -hupcl
    sleep 0.5
done
```

## Conclusion

- We bypassed the microcontroller's CPU entirely.
- We treated the USB adapter as a simple switch.
- **Result:** Hardware control achieved using only standard Linux system tools.