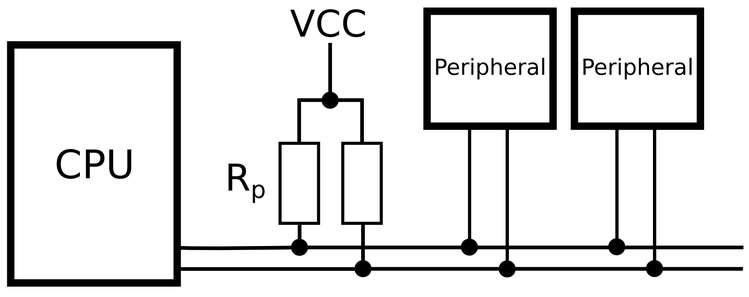
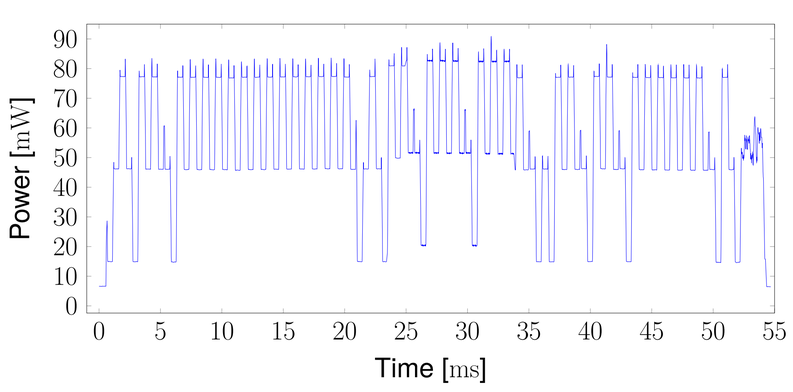
# I²C Considered Wasteful

## Saving Energy with Host-Controlled Pull-Up Resistors

**Birte Friesel, Olaf Spinczyk**
Embedded Software Systems, Universität Osnabrück
{birte.friesel,olaf.spinczyk}@uos.de




**Issue:** Always-on pull-up resistors waste energy for each “0” signal.

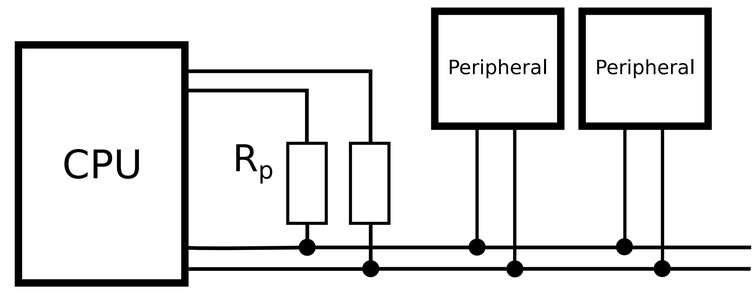
**Possible Remedy:** Connect pull-ups to bus master, disable them when sending “0”

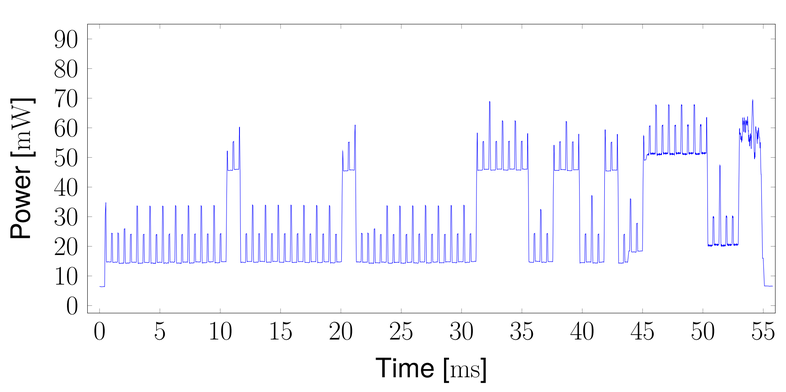
**Complications:** Pull-up control also consumes energy → is it worth it?

**Preliminary Results:**

- Using Software I²C: Yes
- Using Hardware I²C: Not for end-users

**Conclusion:** It pays off to analyze *all* system components – even decade-old protocols may still have room for improvement.




## I²C Basics

“Inter-Integrated Circuit bus”: popular interface for communication between system components

- Two bi-directional wires: SDA (data), SCL (clock)
- Standard mode: up to $100\,\mathrm{kHz}$ provided by bus master
- Transmission: 7-bit address + R/$\overline{\mathrm{W}}$ bit + $n$ data bytes
- Ninth clock cycle after each byte: Ack/Nak
- Pull-up resistors: “1” $\widehat{=}$ do nothing, “0” $\widehat{=}$ connect to 0V

→ Each “0” wastes energy in pull-up

## Transmission Cost

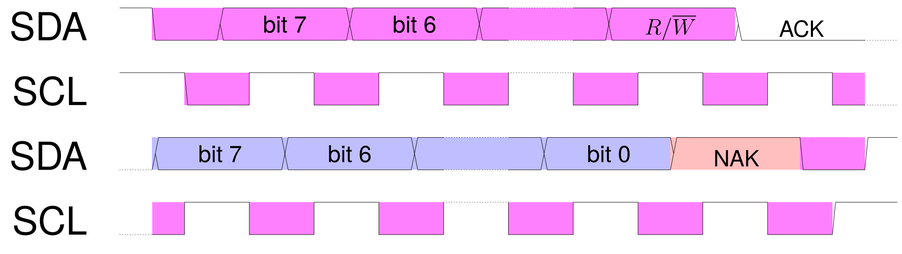
Pull-up energy per clock cycle depends on resistance $R_p$, bus voltage $U$, bus frequency $f$:

$$E_{SCL,*} = \frac{U^2}{2R_p f} \qquad E_{SDA,0} = \frac{U^2}{R_p f}$$

Transmission cost varies with direction and number of 0-bits:

$$E_{SCL} = (9n + 9) \cdot E_{SCL,*}$$
$$n \cdot E_{SDA,0} \leq E_{SDA} \leq (9n + 8) \cdot E_{SDA,0}$$

## Saving Energy




Master-controlled periods during an I²C write (blue/magenta) and read (red/magenta)

- SCL is controlled by bus master for $\approx 100\,\%$ of the time
- SDA for up to $89\,\%$ (write) / $50\,\%$ (read)

→ Connect pull-ups to bus master

→ Disable pull-up while sending low signal on corresponding line

- Pull-up control is transparent to slave devices
- Does not violate I²C specification

Requirements:

- Two additional master output pins
- Adjusted I²C implementation

## I²C Implementations

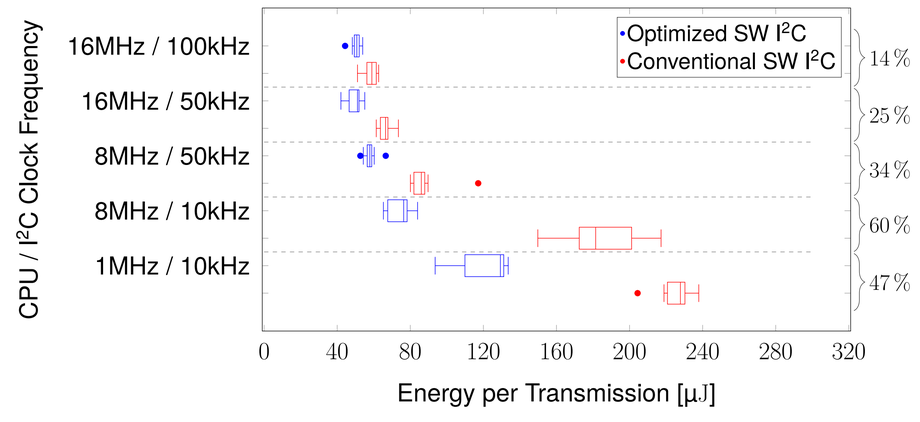
**Software I²C:**

- Microcontroller sets SDA/SCL manually
- $\geq 18$ wake-ups per data byte
- Expensive (wakeup / CPU active time costs energy)

**Hardware I²C:**

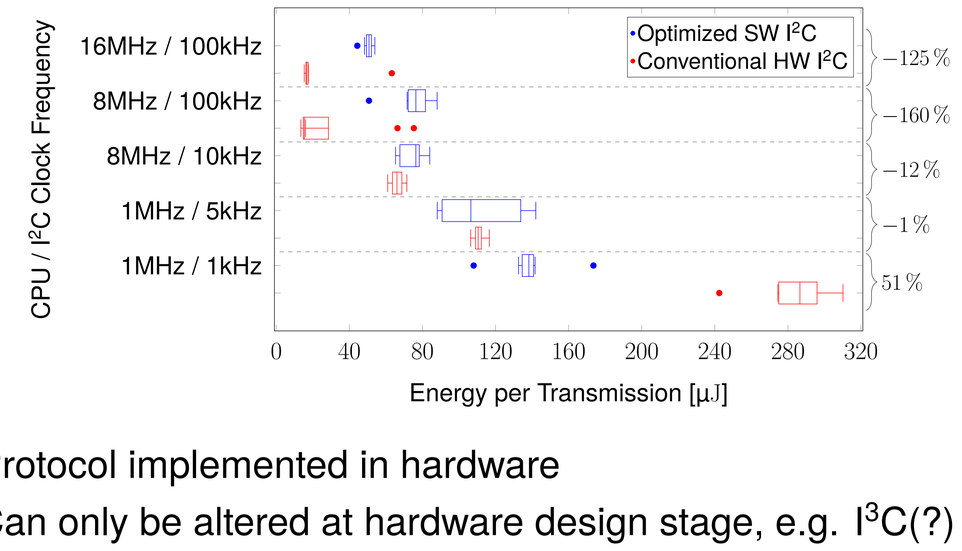
- Hardware module handles protocol autonomously
- $\approx 1$ wakeup per data byte
- More efficient, but not always available

## Evaluation: Software I²C




- Simple adjustment: Toggle pull-up when toggling SDA/SCL
- Overhead: few CPU cycles
- On MSP430FR5969 + I²C temperature sensor: whole-system energy per transmission reduced by $\geq 14\,\%$

## Evaluation: Hardware I²C




- Protocol implemented in hardware
- Can only be altered at hardware design stage, e.g. I³C(?)
- Conventional HW I²C is more efficient than optimized SW I²C
- Exception: Bus clock below 3 to $10\,\mathrm{kHz}$

UNIVERSITÄT OSNABRÜCK

ess.cs.uos.de

embedded software systems