Integration of a microcontroller-based power measurement device

Manuela Beckert, Janosch Hirsch

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

January 21, 2015

Outline

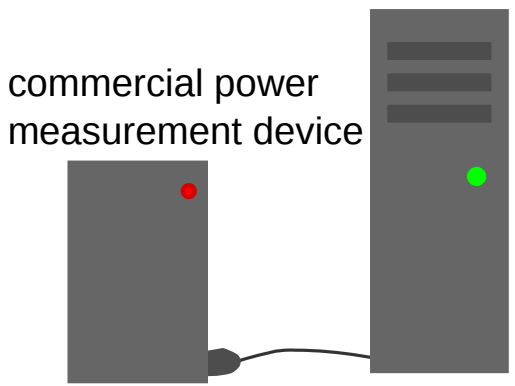
Outline

Motivation

Objective: Precise and time dependent power-consumption measurement of code running on a computer

Available solutions: commercial devices

- expensive
- hard to integrate into hardware
  - laboratory use conditions - not intended for permanent integration
- hard to integrate into software
  - intermediate software required
  - e.g. only for Windows
- not scalable

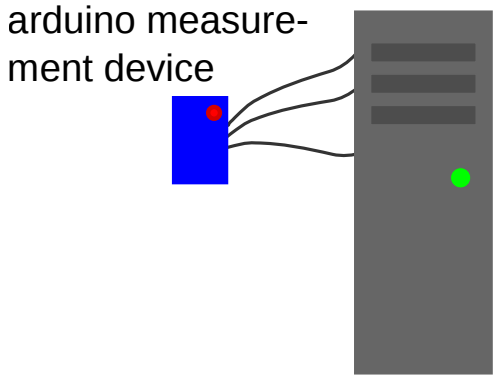


Motivation

Objective: Precise and time dependent power-consumption measurement of code running on a computer

Project goal:

- observation of all mainboard, GPU and HDD power cords
- microcontroller based
- integration in the power measurement tool pmlib
- cheap, small (easy to integrate) and scalable



Outline

Arduino Mega 2560

- Microcontroller Board
- On board bidirectional USB Interface
- Integrated ADC (Analog-to-Digital Converter)
  - 16 Channels with 10 Bit Resolution
- C like Programming language
- Many more Features like General Purpose Input/Output Pins
- Additional boards (called shields) can simply plugged on top
- Easy to use, open source, well documented, large community
- Cheap (around 40 Euro)

Arduino Mega 2560

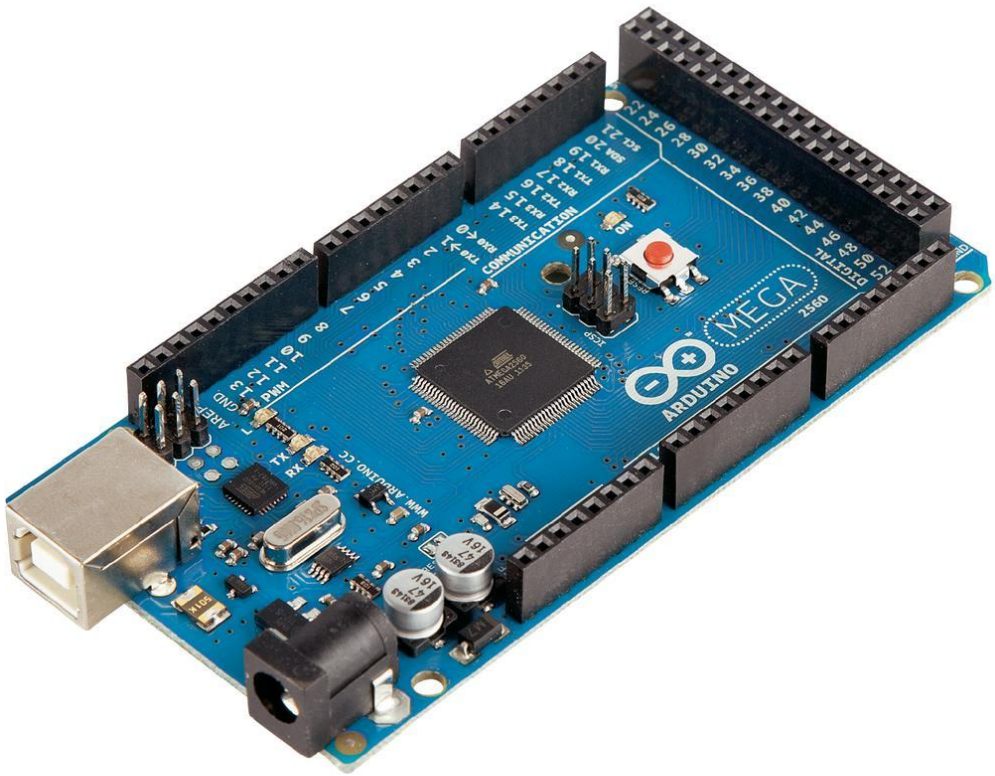


Figure : Arduino Mega 2560 [1]

Arduino IDE

Arduinos provide their own simple IDE

- Code can easily be uploaded to the board by pressing one button
- selection of Arduino model and desired serial port from the Tools menu
- too crude for a real programmer

Arduino IDE

Arduino example code:

```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on
                             // (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making
                             // the voltage LOW
  delay(1000);               // wait for a second
}
```

Measurement shield

- Custom designed board for current measurement
- Consists out of 16 hall-effect sensor IC's (Integrated circuits)
- Can be directly plugged onto an Arduino Mega
- Cheap and simple

Measurement shield

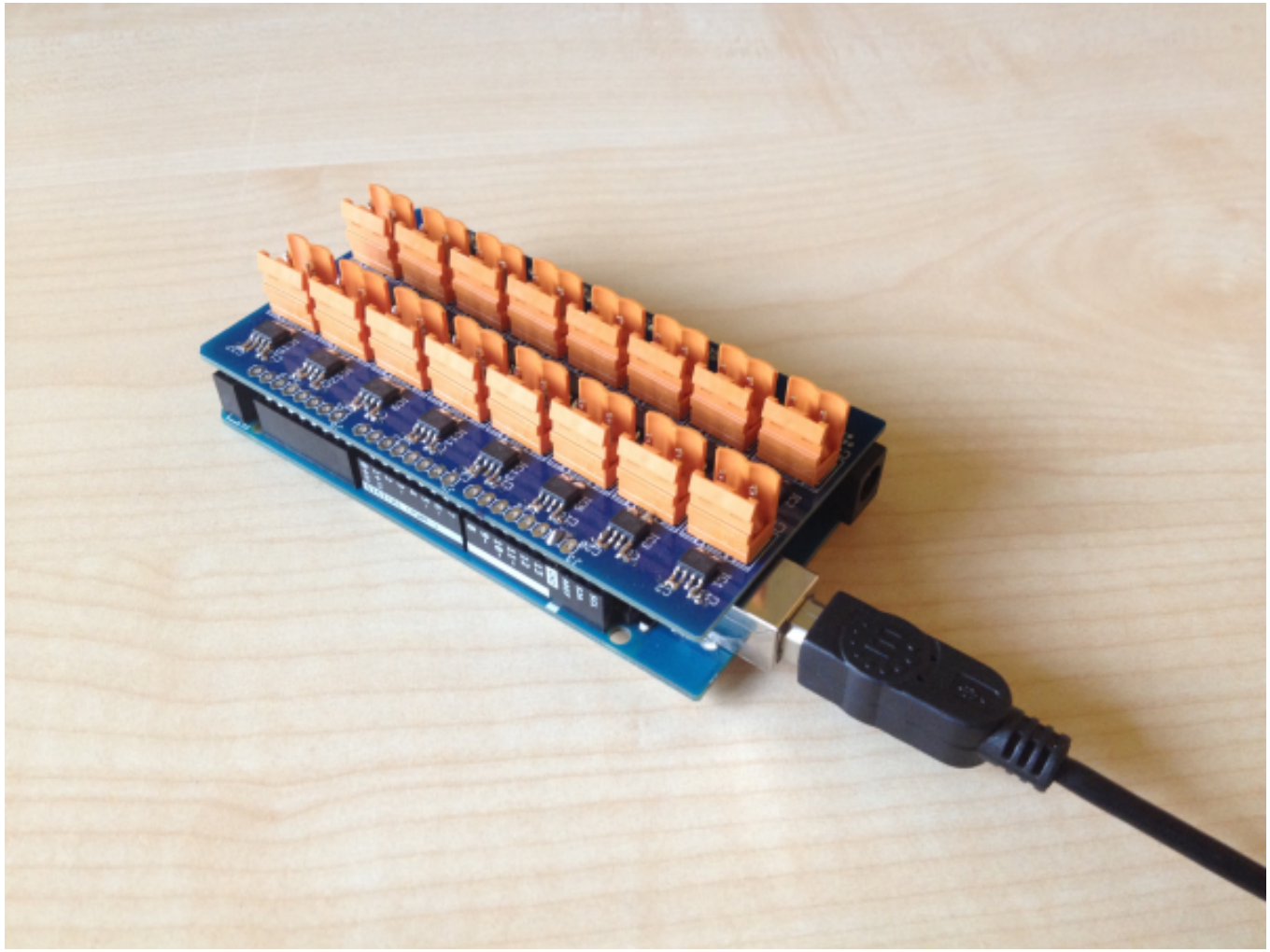


Figure : Arduino with halleffect sensor measurement shield

Hall-Effect Sensor

- Can be used to measure current
- Working principle is based on the Lorentz force
- Outputs a voltage proportional to the current put through

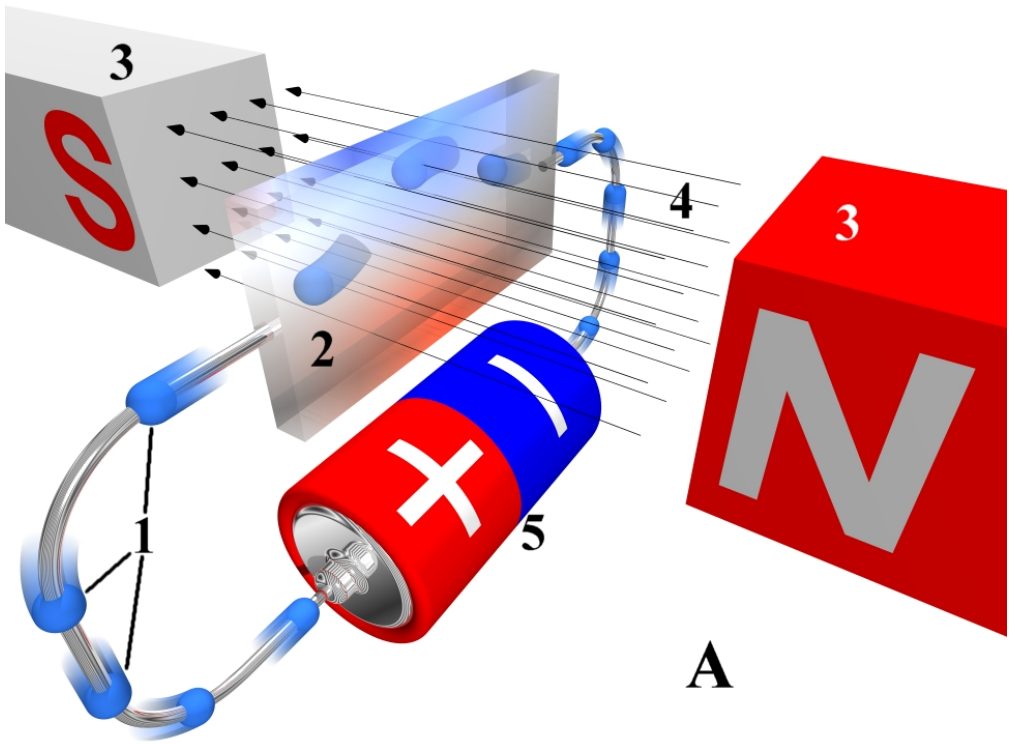


Figure : Halleffect principle [3]

Hall-Effect Sensor

- Can be used to measure current
- Working principle is based on the Lorentz force
- Outputs a voltage proportional to the current put through

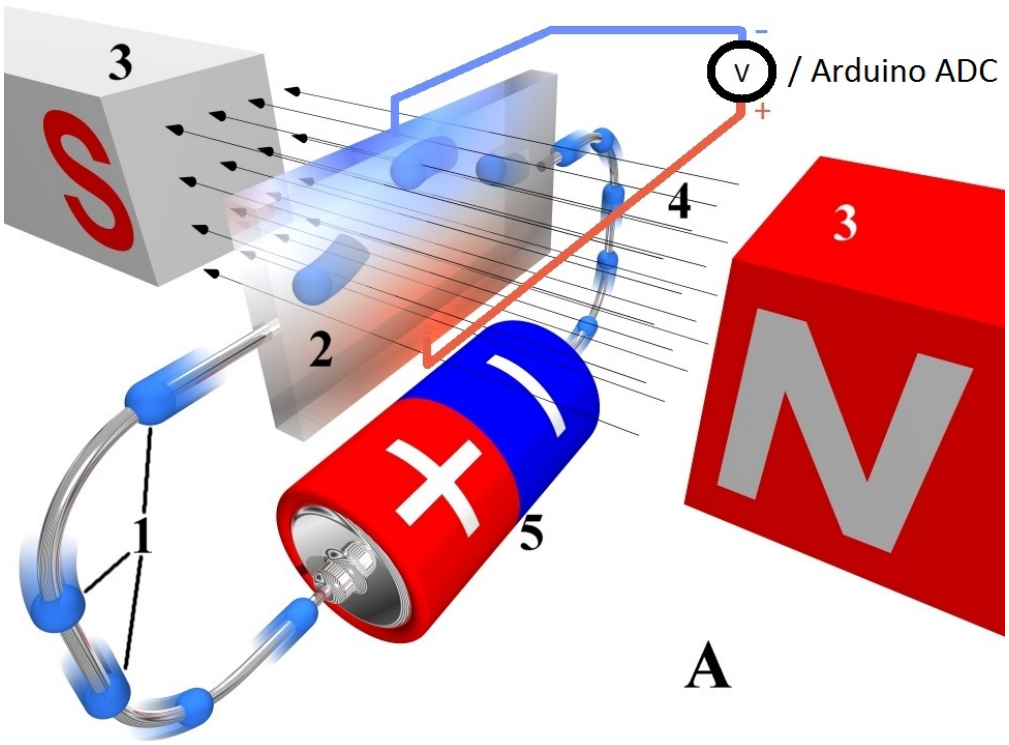


Figure : Halleffect principle [3]

ADC (Analog-to-Digital Converter)

- Converts analog signals to a digital representation
  - Transforms continuous values into discrete ones through sampling and quantization
  - Has a sampling rate and n bits for quantization

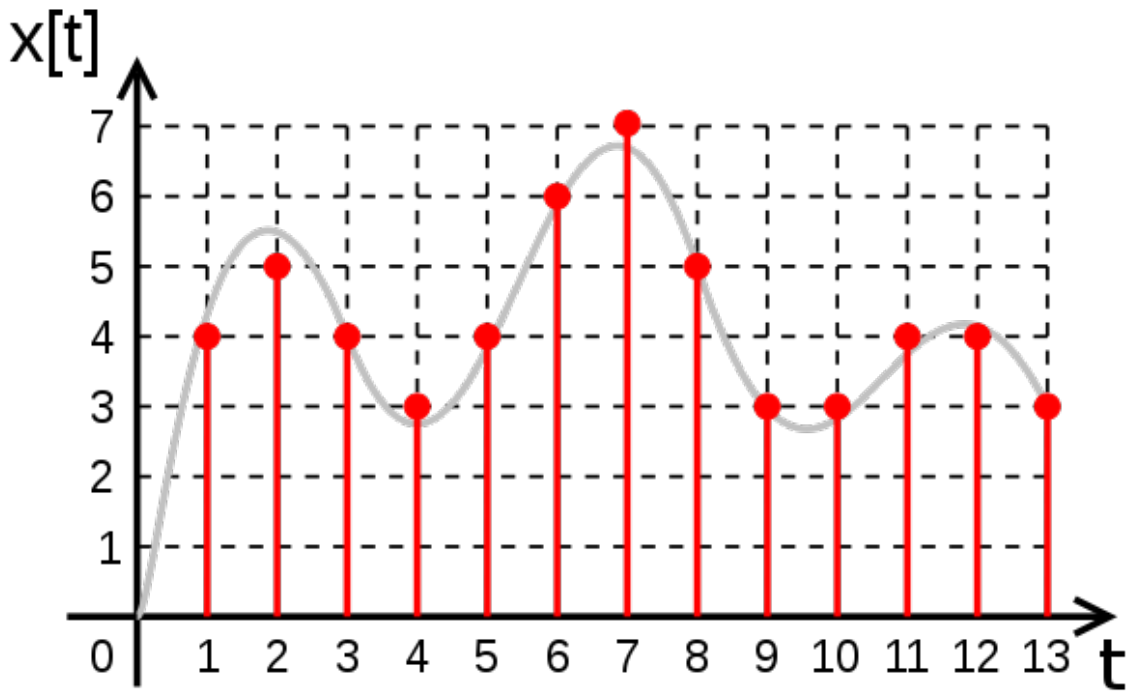


Figure : Sampling and quantization [2]

What is pmlib?

Power **M**easurement **L**ibrary

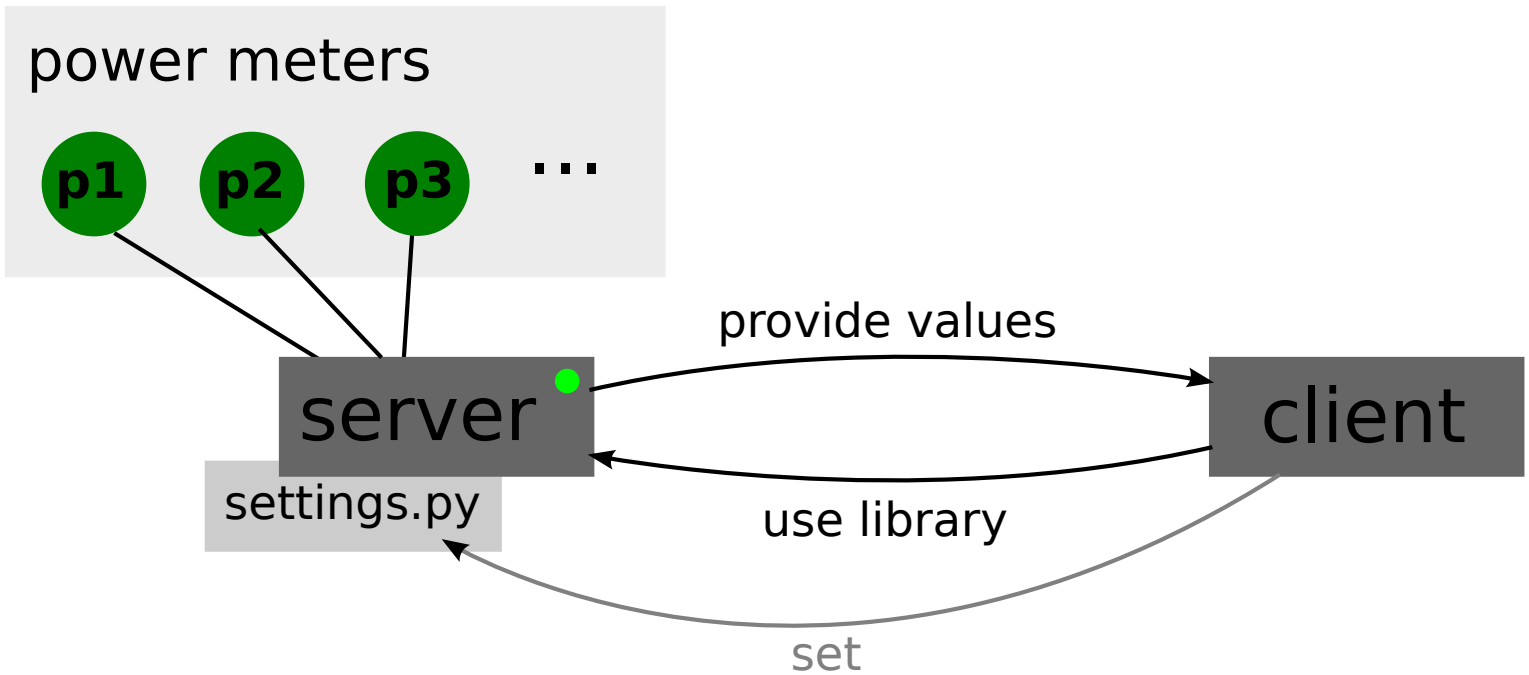
Server daemon

- has access to power measurement devices
- collects data and sends it to clients

Client library

- enables communication with server
- allows clients to measure the power consumption of their programs

pmlib


power meters
p1
p2
p3
...
provide values
server
client
settings.py
use library
set

pmlib Server

- one class for every device attached
- handles how information from the device shall be interpreted
  - has power measurement devices attached
  - if client asks for measurement, startes collecting data
  - reads from file `settings.py`
  - computes power from read values if needed
  - stops when client asks to stop
  - data will be stored until client asks to receive the data

pmlib Client

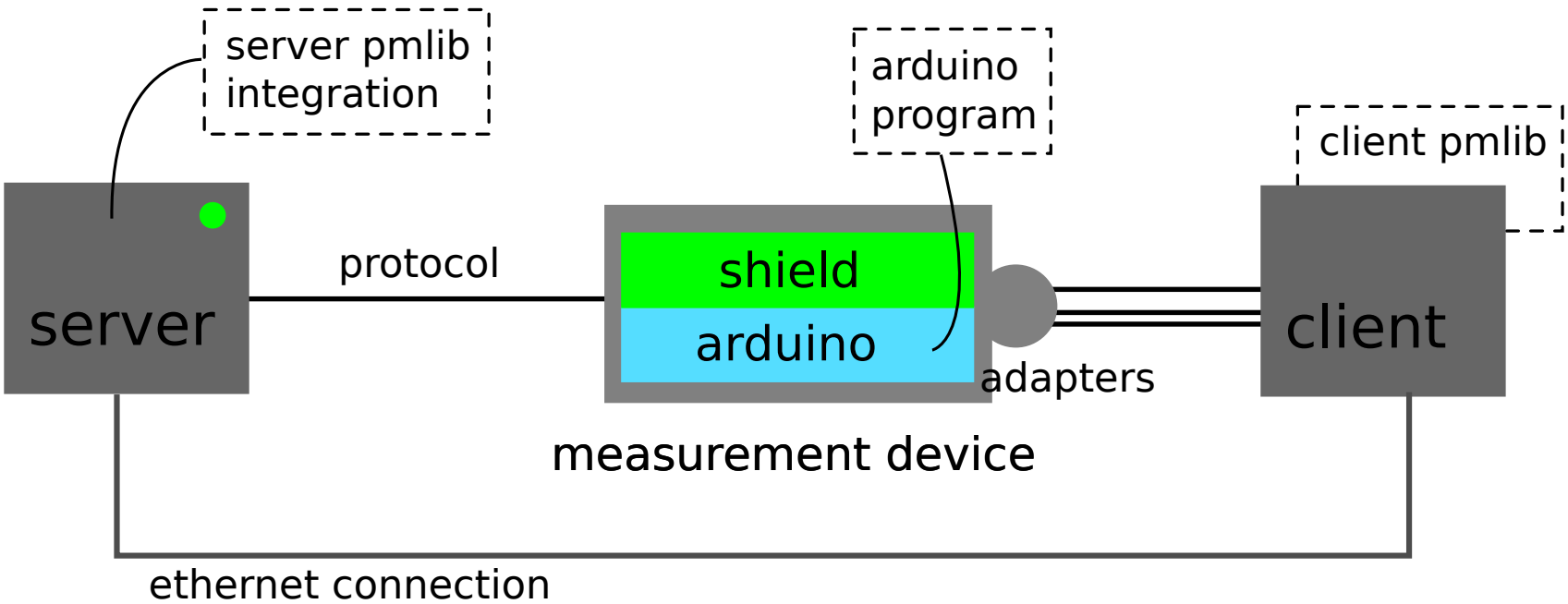
Usage for a client:

- set needed information in file `settings.py`
  - which power measurement devices
  - which lines/channels to read from the devices
- in program source code: use library methods to start measuring
  - server is written in Python, but there is a library for C

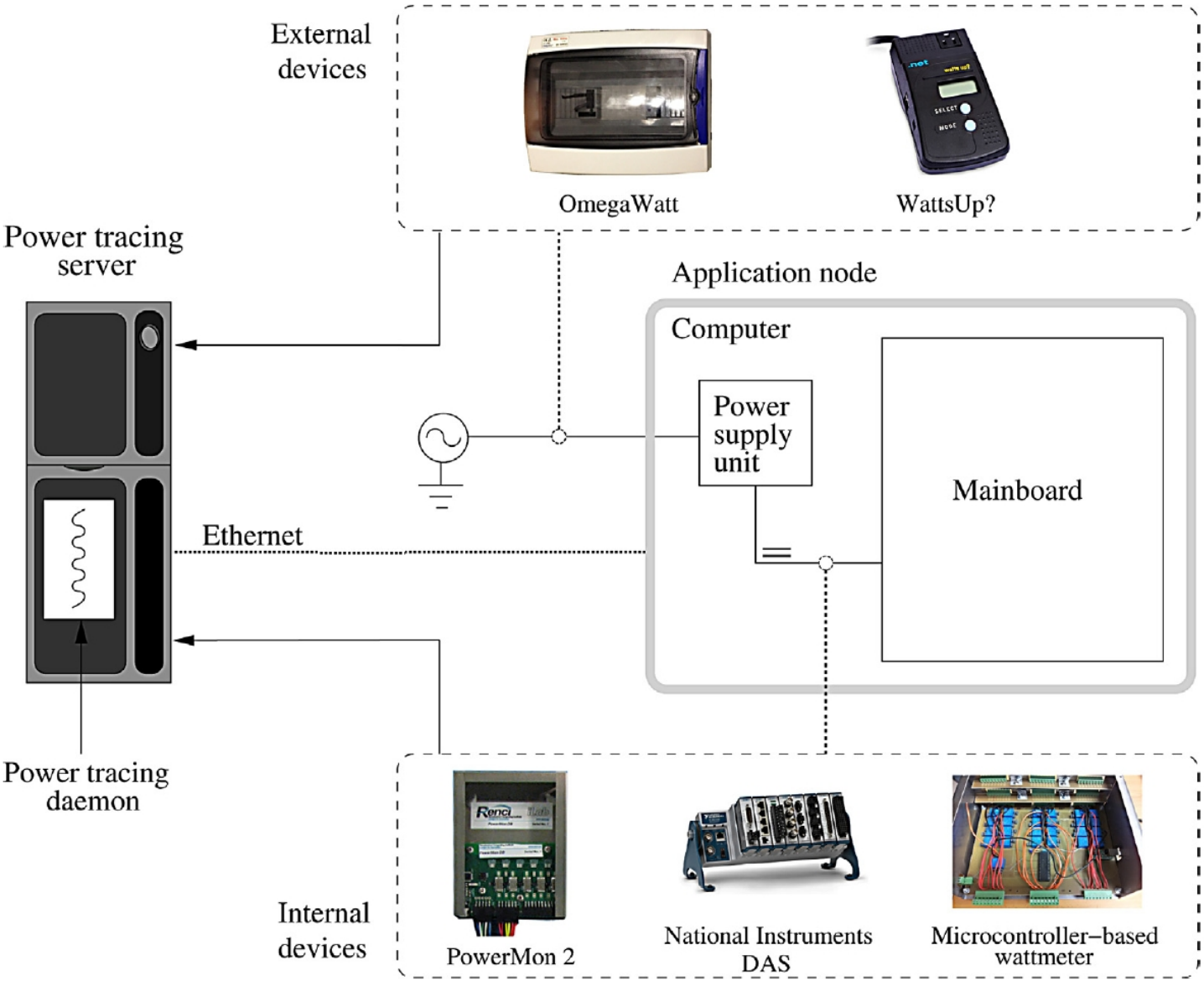
```
pm_create_counter("DCM1",...)
pm_start_counter(...)
[Code to measure]
pm_stop_counter(...)
pm_get_counter_data(...)
```

Outline

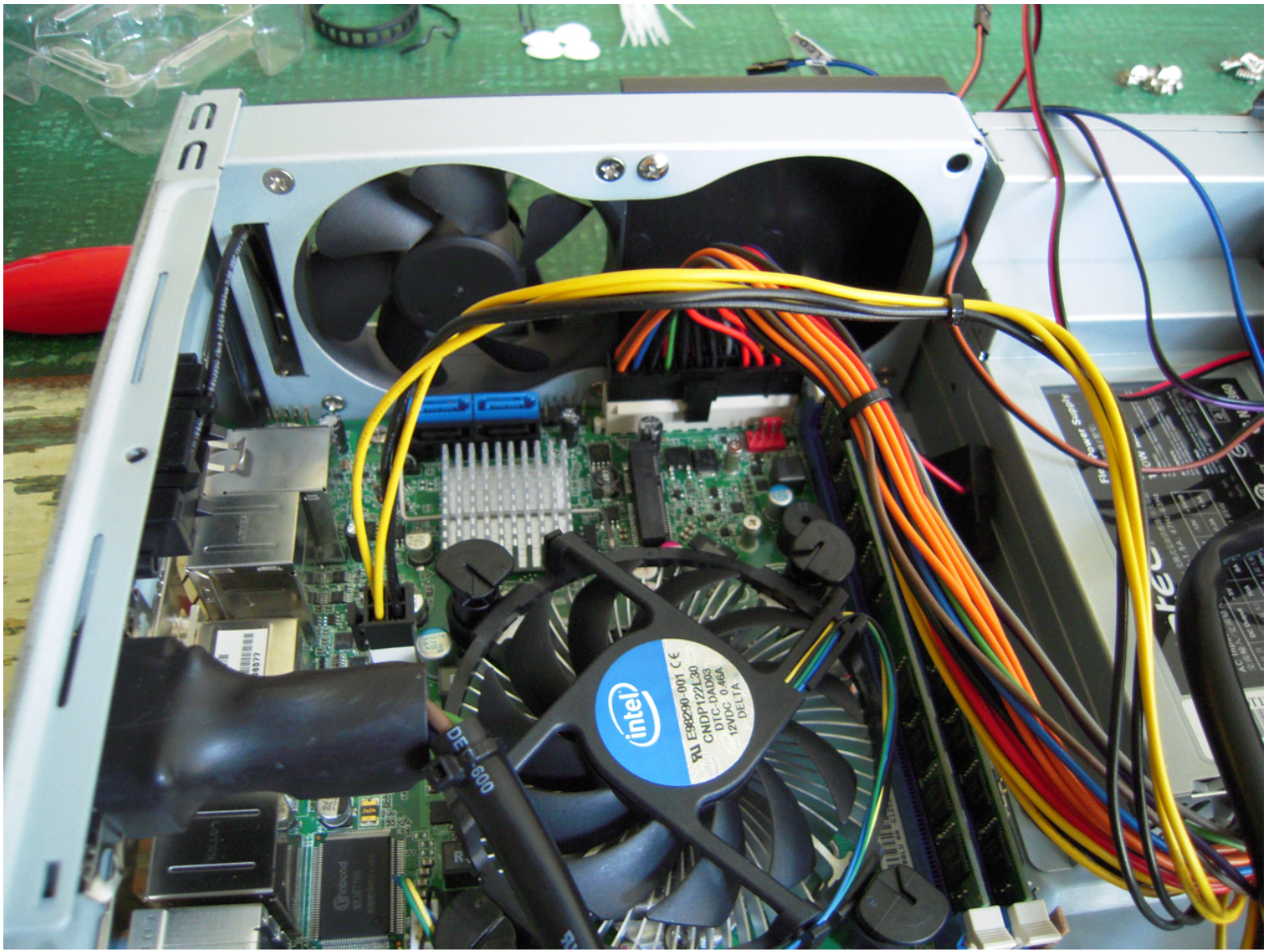
Setup


server pmlib
integration
arduino
program
client pmlib
server
protocol
shield
arduino
client
adapters
measurement device
ethernet connection

Setup


External devices
OmegaWatt
WattsUp?
Power tracing server
Application node
Computer
Power supply unit
Mainboard
Ethernet
Power tracing daemon
Internal devices
PowerMon 2
National Instruments DAS
Microcontroller–based wattmeter

Setup



Setup

- the arduino-blackbox is attached into a computer
- the computer can be closed again
- only a usb cable to the pmlib server is needed
- highly scalable
  - you can put one in every computer of whole cluster

Outline

Serial Protocol

- used for communication between microcontroller and USB interface chip on the arduino board
- usb-to-serial is handled by the operating system

Serial Protocol

- used for communication between microcontroller and USB interface chip on the arduino board
- usb-to-serial is handled by the operating system

pmlib $\rightarrow$ **arduino:** selecting channels for data retrieval

Protocol:

- 2 byte long, every bit represents a channel
- 1 means the channel shall be send
  - Example for channel 0, 1 and 3:

```
00000000 00001011
```

Serial Protocol

arduino $\rightarrow$ **pmlib:** sending values

Format used:

```
0CCCCVVV 1VVVVVVV
```

- the first bit indicates whether we are reading the first or second byte
  - might be useful for robustness later
- 4 bit for the channel (C)
- 10 bit for the value (V)

Serial Protocol

arduino $\rightarrow$ **pmlib:** sending values

Format used:

```
0CCCCVVV 1VVVVVVV
```

- the first bit indicates whether we are reading the first or second byte
  - might be useful for robustness later
- 4 bit for the channel (C)
- 10 bit for the value (V)

- we may use a different protocol with less overhead and higher sample data rate
  - Packing: A - first value, B - second value etc.

```
AAAAAAAA AABBBBBB BBBBCCCC ...
```

Arduino Code and pmlib Integration

- we'll show you some code
- and a live presentation

Calibration process

- Measured current vs. bit representation of the hall-effect sensor voltage from the Arduino ADC
- Difficulties:
  - Noise
  - Hall-effect sensor offset voltage
  - Selection of a ADC reference voltage for higher precision
  - Calibrate on channel or shield level?
- Calibration through linear regression
- Make a script for fast calibration

Calibration process

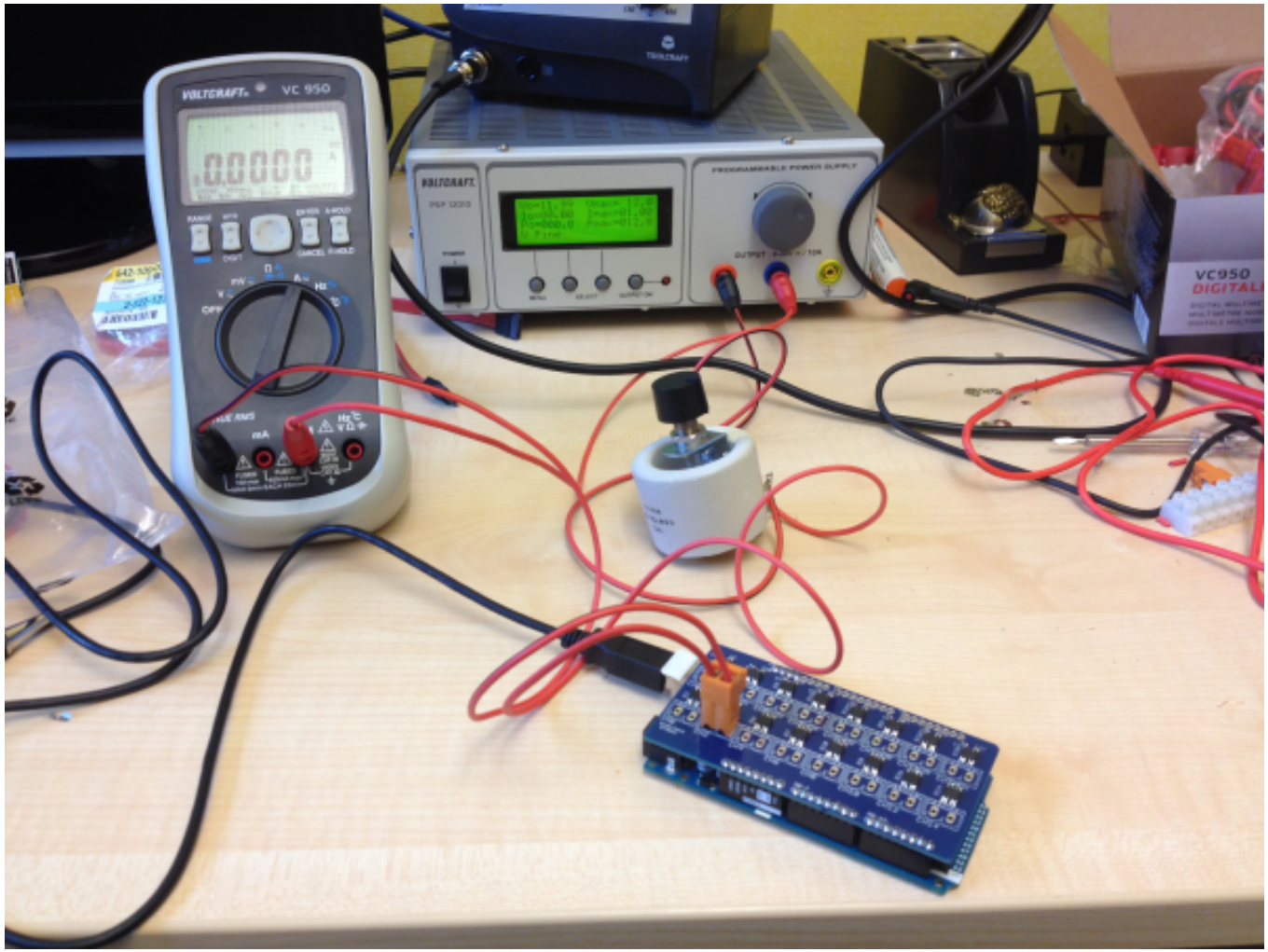


Figure : Calibration arrangement

Calibration process

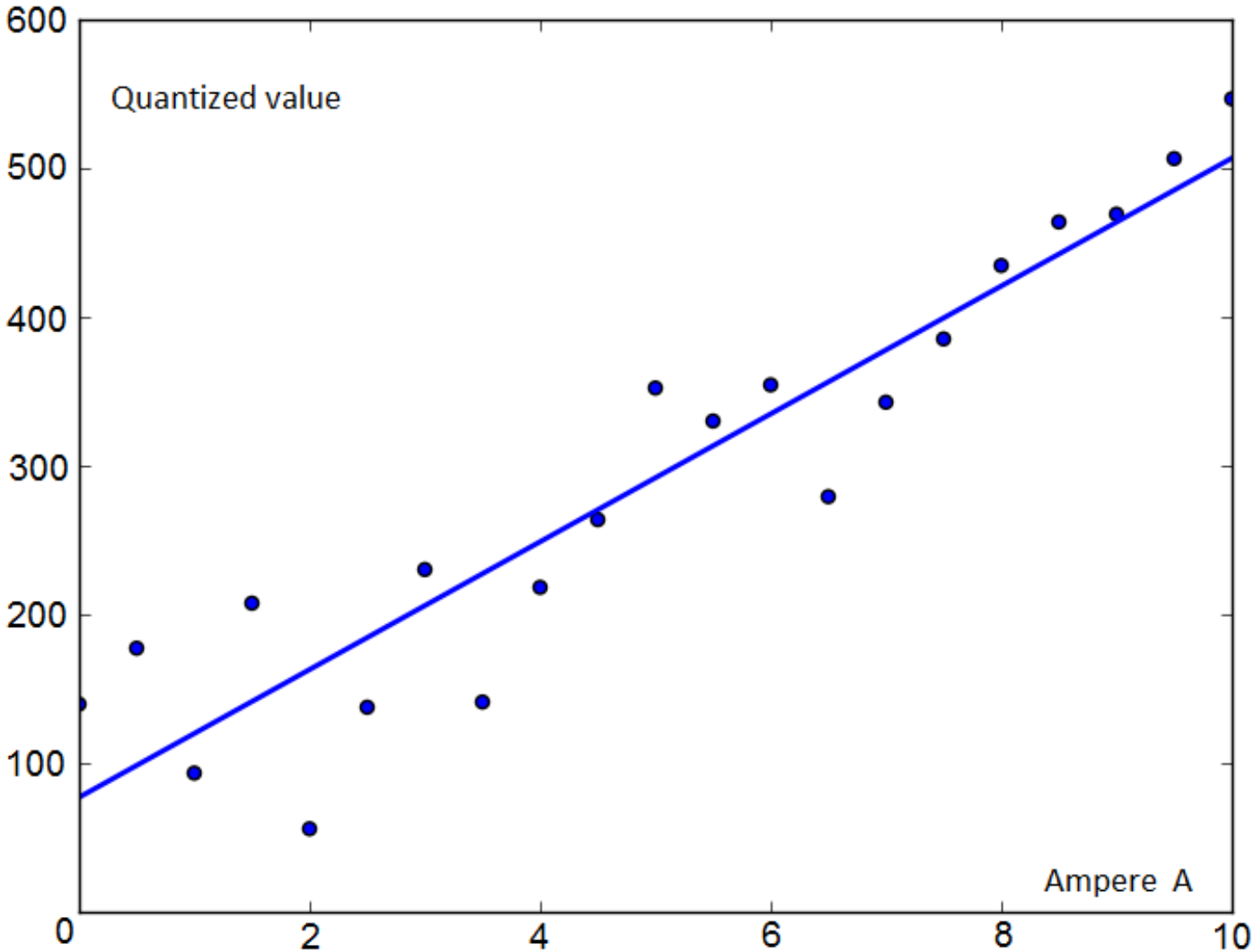


Figure : Linear regression graph

Outline

Current State of the Project

Done:

- prototype of the device
- code for Arduino
- code for pmlib
- working serial protocol

Todo:

- consider using a different protocol for better sample data rate
- Measuring speed and performance of the code and hardware
- Calibrate the device
- Provide a script for automatized calibration
- Install the device in a server environment

Outline

Literature

- Arduino homepage: http://arduino.cc
- pmlib Github: https://redmine.wr.informatik.uni-hamburg.de/git/pmlib
- ADC: http://en.wikipedia.org/wiki/Analog-to-digital_converter
- Hall-effect: http://en.wikipedia.org/wiki/Hall_effect

1 http://www.conrad.de/medias/global/ce/1000_1999/1900/1910/1917/191790_BB_00_FB.EPS_1000.jpg

2 http://upload.wikimedia.org/wikipedia/commons/thumb/0/04/Digital.signal.discret.svg/600px-Digital.signal.discret.svg.png

3 http://upload.wikimedia.org/wikipedia/commons/0/01/Hall_effect.png

Questions

Are there any questions?

Thank you

Thank you for your attention