WISSEN
TECHNIK
LEIDENSCHAFT

EWSN'24 Sustainability Competition

Competition format & How to use the E-Cube Testbed

Version 0.1

Dr. Markus Schuß

Low-power Embedded Networked Systems (LENS) group
Graz University of Technology, Austria

15.04.2024

Outline

- EWSN’24 competition
  - Organizers
  - Why this new competition?
  - Challenge at a glance
  - Awards & Timeline
  - Challenge in detail / Hashcash
  - Reference solution
- E-Cube testbed
  - Scheduling an experiment
  - Available energy traces
  - Monitoring experiments
  - Experiment results
- Contacts

EWSN'24 Sustainability Competition

Organizers

Nivedita Arora
(Northwestern University, US)
nivedita@northwestern.edu

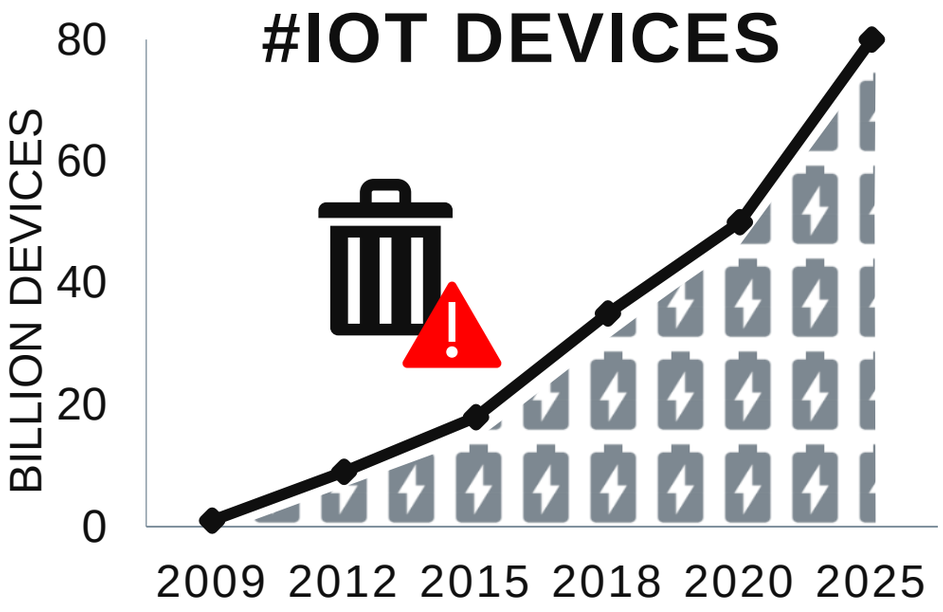
Przemysław Pawełczak
(TU Delft, NL)
p.pawelczak@tudelft.nl

Markus Schuss
(TU Graz, AT)
markus.schuss@tugraz.at

EWSN'24 Sustainability Competition

Why this new competition?

- By 2025, 78 million batteries used in IoT devices will be dumped globally each day [1]
- Several battery-free systems have been proposed by both academia and industry
  - → Performance rarely benchmarked under the same settings
  - → Let's bring together the community at EWSN to compare the performance of battery-free systems under the same settings
  - → Let's leverage D-Cube as infrastructure and modify it to benchmark battery-free systems



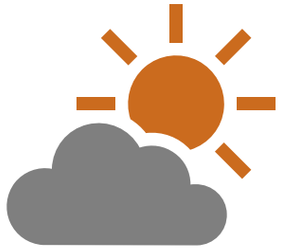
[1] https://cordis.europa.eu/article/id/430457-up-to-78-million-batteries-will-be-discarded-daily-by-2025-researchers-warn

EWSN'24 Sustainability Competition

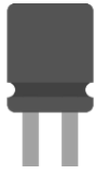
Challenge at a glance

Different real-world traces of a given energy source (e.g., solar panel) are replayed by the testbed in a reproducible way

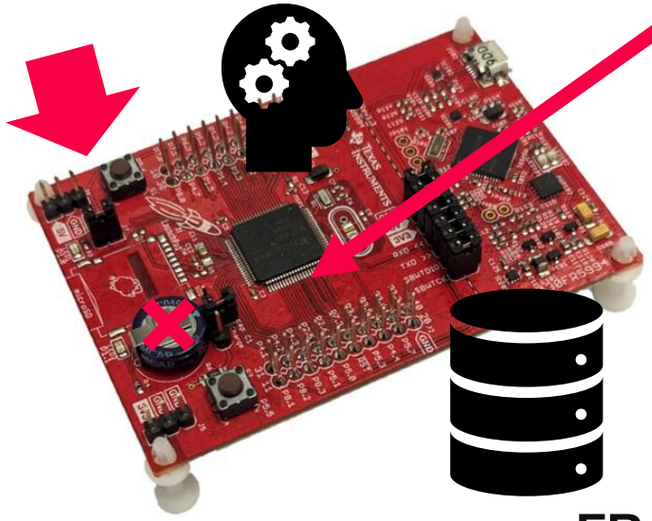
Trace 1
Trace 2
...



Simulated power supply charging a **small capacitor** powering the battery-free device (steered by the testbed)



Queue of Computation Tasks
(provided by the testbed, can be retrieved by the battery-free device at will)



Battery-free Device
MSP430FR5994

FRAM-Memory (internal)
for checkpoints

Contestant's firmware

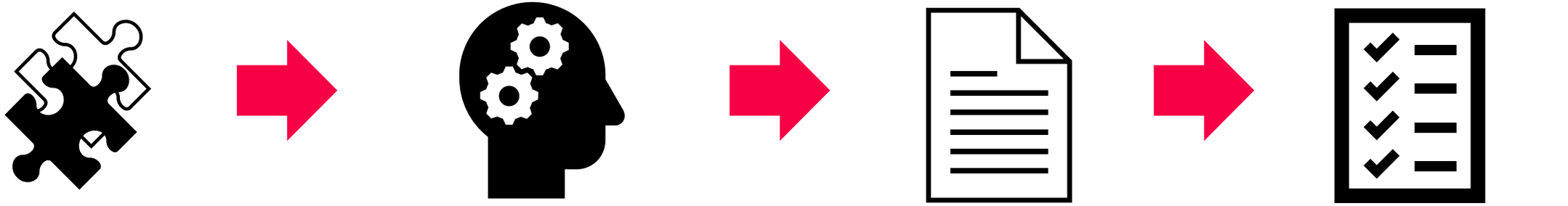
uploaded to the E-Cube testbed

Task: read a challenge from an **external** device (I2C FRAM task queue), perform the computation (which may take longer than the time available from the given energy supply!) and deliver proof of work to the **external** device

Proof of Work
(collected by the testbed)

EWSN'24 Sustainability Competition
Evaluation

- Solutions will be evaluated according to **proof of work** (Hashcash):
  - **At startup** an external FRAM holds N challenges (varying difficulty)
    → Compute the solution to a challenge and report the solution to FRAM (computation may require more energy than the capacitor can store!)
  - **At the end** the testbed reads back the solutions and awards points based on the number and difficulty of solved challenges
    → The current points on the website are a **placeholder** and subject to change! (the final metric as well as a leaderboard will be released during/after the informal testing phase)



Challenges
(challenge for Hashcash)

Compute the solution to (ideally) each challenge

Store solved challenges
(correct hash found for a given challenge in well-known format)

Award Points
(for each solutions based on difficulty)

EWSN'24 Sustainability Competition

Awards

- Top three teams will be awarded
  - First place: 6.000 €
  - Second place: 3.000 €
  - Third place: 1.500 €

- Announcement of the results
  - During the main conference track

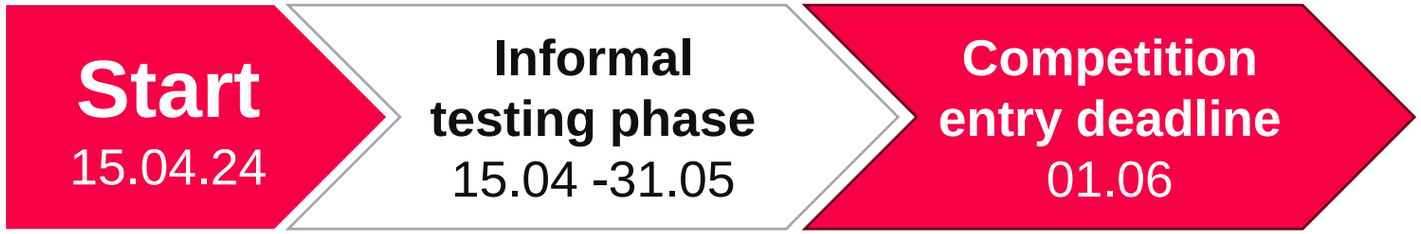
→ The top 3 teams will be given the opportunity to **present their solution** as a lightning talk

→ All participants will present their approach during a dedicated **poster session**

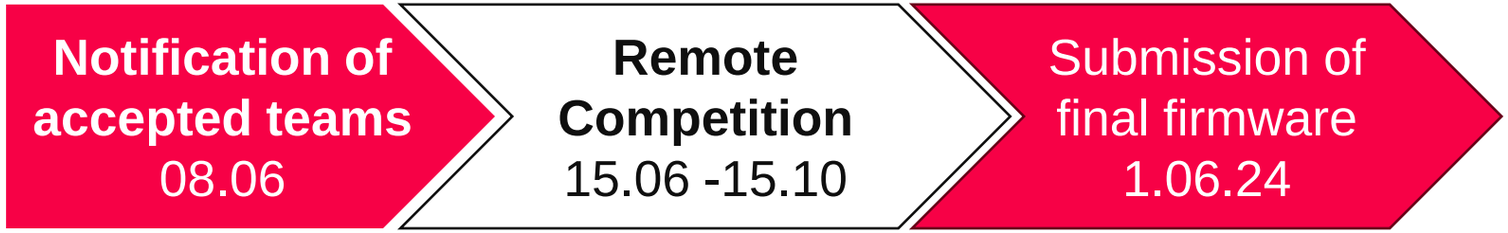
EWSN'24 Sustainability Competition

Roadmap & Timeline

- To allow contestants to test their solutions without commitment, the E-Cube testbed is opened early with a reduced feature set
  (this also allows us to stress test E-Cube and scale the number of nodes)

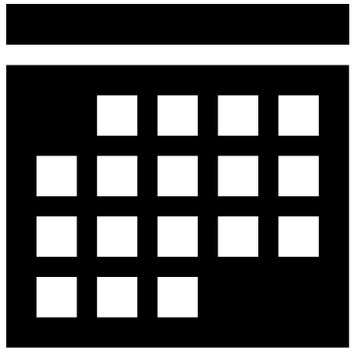


- Before the remote competition starts, all features will be enabled
  (leaderboard, more energy traces, final evaluation metric, etc.)



Timeline
Informal testing phase

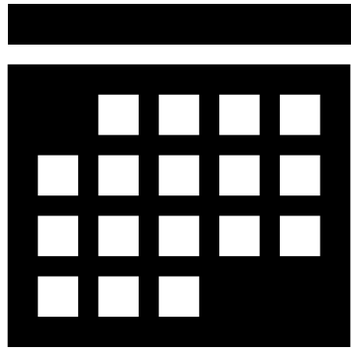
- The testbed infrastructure will be available remotely for 6 months
  - Informal testing phase: **April 15, 2024**
    - Teams can get familiar with the infrastructure
    - Open to “everyone” → **no** commitment to participate to the competition
    - Number and duration of energy traces limited
    - Allows potential contestants to tests their solution and as well as the competition infrastructure
    - This is meant also as a **stress test** for E-Cube, do not hesitate to contact us if something seems wrong or if you'd like additional features! (see last slide for e-mail contact and Discord group)



Timeline
Remote competition

- The testbed infrastructure will be available remotely for 6 months
  - Remote competition: **June 15 – October 15, 2024**
    - Access limited to the final teams selected as contestants
    - Over this timespan, we can beef up the challenge with
    - New energy traces
      (will be available for download along the original traces)
    - An increased difficulty of challenges
      (Hashcash allows for scaling of the difficulty for each individual challenge)
    - Goal: push the performance of your solutions to the limit!

→ “Holding back” your solution just means you may be surprised later!
(we will try different traces/challenges for the final experiments)



Timeline

Evaluation of the final firmware

- After the remote competition phase, a final firmware has to be submitted
- Deadline is in **October 15, 2024**
- Using this firmware, the organizers will perform a final evaluations
- Includes unreleased energy traces to avoid engineered solutions
  - Will use different challenge seeds!
  - Will include longer experiments!

- Results will be kept **secret** until the conference (**December 2024**)

→ The goal is to find the best performing general solution, not the one best tailored to the provided traces!

EWSN'24 Sustainability Competition
Challenge in detail

Goal: With the limited amount of (intermittent) energy, perform as much work as possible

- To verifiably prove that a certain amount of computations were performed, we employ a proof-of-work algorithm called Hashcash [3]
  - Based on the SHA1 hash algorithm
  - A given challenge contains a resource, e.g., "`EWSN2024`" and a difficulty in bits, e.g., `16`: Find as string s which contains the resource in a **well-known** format with the hash h(s) having its most significant 16 bits (as per difficulty) zero

[3] https://en.wikipedia.org/wiki/Hashcash

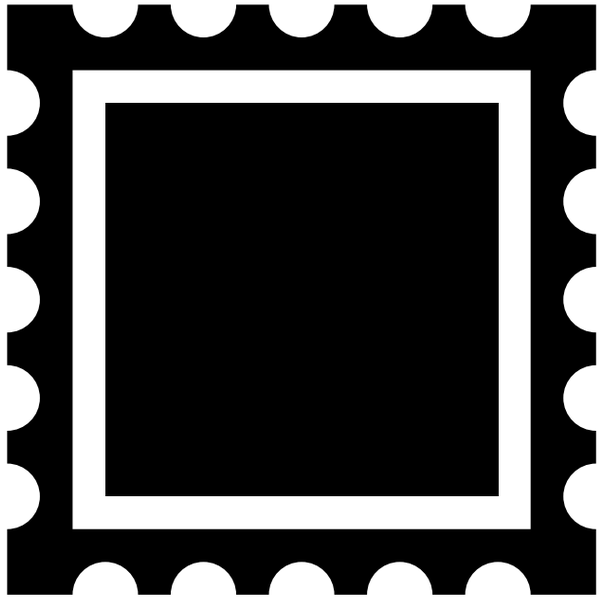
Hashcash
Well-known Format

Fixed for the competition
Provided as challenge
Up to the user

Well-known format [4] of a Hashcash **solution** (called “stamp”)

`ver:bits:date:resource:ext:rand:counter`

- **ver**: version (always 1)
- **bits**: difficulty (provided alongside challenge)
- **date**: **fixed** (use 240415 as the date)
- **resource**: challenge string
- **ext**: always empty
- **rand**: user-chosen random string (mitigate multiple spending)
- **counter**: a user-generated counter



[4] http://www.hashcash.org/docs/hashcash.html#stamp_format__version_1_

Hashcash

Solving the EWSN2024 Challenge

To solve the **challenge** “EWSN2024” with a difficulty of 16 bits:

- Create the string s = “`1:16:240403:EWSN2024::WXhnFeDleN:1`”
- Compute it’s hash
  h(s) = `20db26cb6b1e17a2079fc3daf05fd01a7ed08cb5`
  - Only the **two** most significant bits are zero (0x20=**00**100000 in binary)
  - (A SHA1 reference implementation is provided, others can be used)
- As per difficulty (16) this hash is not the solution, we increment the counter and retry hashing the new string until we find one with 16 leading zeros ...

Hashcash

Solving the EWSN2024 Challenge

To solve the **challenge** “EWSN2024” with a difficulty of 16 bits:

- ...
- After 96620 attempts
  s=“`1:16:240403:EWSN2024::WXhnFeDleN:96620`” has a hash
  h(s)=`0000babe195c81d00ecec4cd8f0dc1572ebd46d4`
  - This hash finally has the required 16 leading zeros (most significant bits of zero)

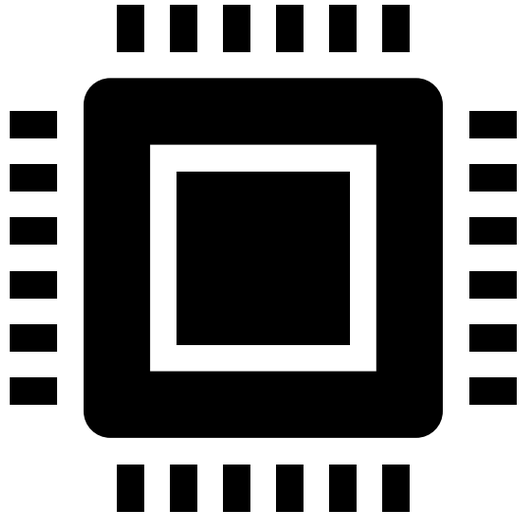
  → “`1:16:240403:EWSN2024::WXhnFeDleN:96620`” is a **solution** to the challenge “`EWSN2024`” with a difficulty of 16 (others exist as well!)

Hashcash

Reading Challenges and reporting Solutions

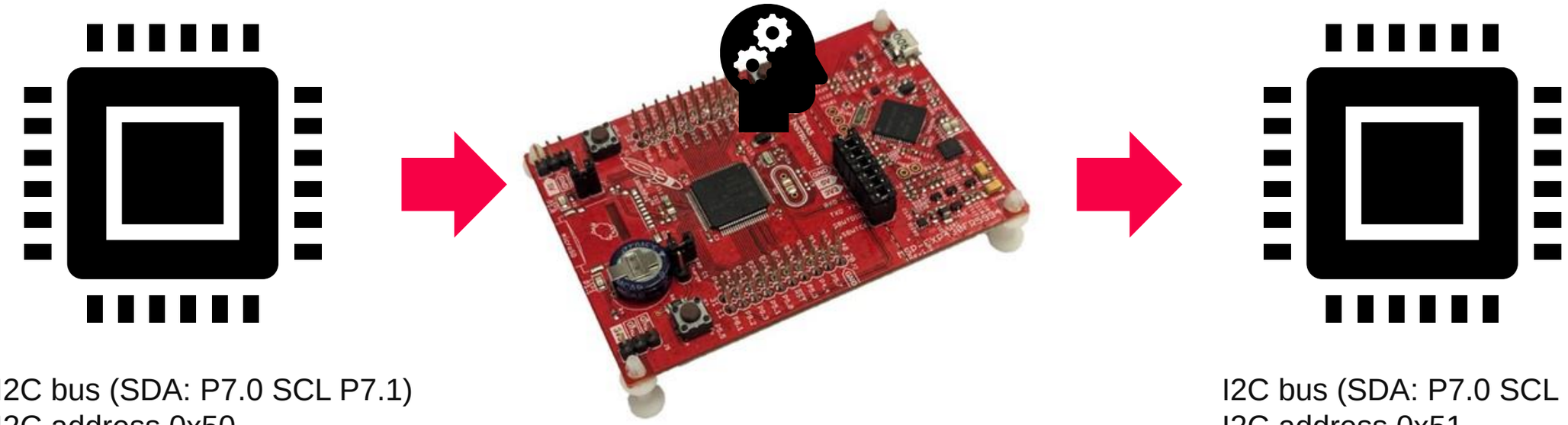
External I2C FRAM chip (MR44V100A) (128kB **split** into 2x64kB parts)

- A number of **challenges** are stored in the first part at address I2C 0x50 before startup by E-Cube → 64kB *challenge image*
- Each challenge has a fixed length: 1 byte of difficulty, 15 byte challenge
- Unused slots are zeroed (\0) out and indicate the end of the challenges (termination)
- At runtime the resulting **solutions** are stored on the second address (0x51) by the contestant's firmware
- At the end, E-Cube retrieves the entire FRAM content → 64kB *solutions image*



Competition Scenario
Detailed Version

Read a challenge from the I2C FRAM (0x50), compute a valid solution using Hashcash (accounting for the intermittent energy supply), and write it to the solution as string I2C FRAM (0x51, as null aka \0 terminated string)



I2C bus (SDA: P7.0 SCL P7.1)
I2C address 0x50
16 bit register addresses
(0-65535) for 64k of memory

I2C bus (SDA: P7.0 SCL P7.1)
I2C address 0x51
16 bit register addresses
(0-65535) for 64k of memory

Competition Scenario

Challenge Image Format

Upon start, the FRAM contains 16 byte challenges, with the first byte being the difficulty (e.g., 4) and the remaining 15 bytes the actual challenge string

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text	
00000000	04	46	46	63	53	68	43	34	79	73	37	61	63	61	49	6F	.FFcShC4ys7acaIo	1
00000010	04	31	48	49	6F	53	48	54	6A	66	66	4C	6C	47	34	73	.1HIoSHTjffLlG4s	2
00000020	04	33	33	59	59	34	53	68	45	52	47	41	43	34	33	35	.33YY4ShERGAC435	
00000030	04	58	34	4D	33	42	6A	56	66	64	34	66	6A	44	59	35	.X4M3BjVfd4fjDY5	
00000040	04	4D	57	62	73	4D	67	36	69	4D	56	75	6A	36	48	74	.MWbsMg6iMVuj6Ht	
00000050	04	30	36	76	67	72	44	50	69	6D	34	6F	49	78	77	62	.06vgrDPim4oIxwb	
00000060	04	44	55	78	51	32	79	51	4D	66	45	70	34	65	52	4A	.DUxQ2yQMfEp4eRJ	
00000070	04	32	64	37	4E	52	70	45	6A	64	4F	6D	43	4B	38	57	.2d7NRpEjdOmCK8W	
00000080	04	36	63	70	63	54	36	73	61	6A	74	55	73	62	57	30	.6cpcT6sajtUsbW0	

Note: While the challenges could be read in any order, solutions must be written in the same order as the challenges are in the FRAM - **skipping challenges is not allowed!**

Competition Scenario
Solution Image Format

Upon computing a valid solution write the solution to the FRAM (second address 0x51, not 0x50!) using the well-known format, terminated by \0

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text	
00000000	31	3A	34	3A	32	34	30	33	32	36	3A	46	46	63	53	68	1:4:240326:FFcSh	1
00000010	43	34	79	73	37	61	63	61	49	6F	3A	3A	38	58	6B	45	C4ys7acaIo::8XkE	
00000020	3A	32	00	31	3A	34	3A	32	34	30	33	32	36	3A	31	48	:2.1:4:240326:1H	2
00000030	49	6F	53	48	54	6A	66	66	4C	6C	47	34	73	3A	3A	65	IoSHTjffLlG4s::e	
00000040	36	2B	55	3A	32	32	00	31	3A	34	3A	32	34	30	33	32	6+U:22.1:4:24032	
00000050	36	3A	33	33	59	59	34	53	68	45	52	47	41	43	34	33	6:33YY4ShERGAC43	
00000060	35	3A	3A	5A	6A	6A	57	3A	38	00	31	3A	34	3A	32	34	5::ZjjW:8.1:4:24	
00000070	30	33	32	36	3A	58	34	4D	33	42	6A	56	66	64	34	66	0326:X4M3BjVfd4f	
00000080	6A	44	59	35	3A	3A	58	58	4A	41	3A	31	39	00	31	3A	jDY5::XXJA:19.1:	

Note: While the challenges have a fixed width (16 bytes) the solutions do not and thus \0 termination is **required**

Reference Solution

- We provide a reference solution based the Arduino fork for the MSP430: Energia (https://energia.nu/)
  - https://iti-ecube.tugraz.at/wiki/images/b/b0/Hashcash_example.zip
    (builds with software serial enabled, change '`#define DEBUG 1`' to 0 to disable)
  - Does not include any awareness for the intermittent energy supply
  - Includes a register level I2C driver and SHA1 implementation
- This solution (while computing correct Hashcash solutions) serves to illustrate the task, it is not necessary (or expected) to use it **as is**!
  (e.g., you are free to take the parts that are useful to you and integrate them in your own solution)

Reference Solution

- The code implements the three steps to solve the task outlined in the competition scenario (read challenge, solve challenge, write solution)

1. It reads a new challenge from the FRAM (starting at register address 0x0000)
   - The provided I2C driver is taken from TI's examples found at (https://dev.ti.com/)
   - As the read call is non-blocking, we busy wait until it is finished (this is a good point to improve upon in your design)

```
127   I2C_Mode rx = I2C_ReadReg(0x50, challenge_counter*CHALLENGE_SIZE, CHALLENGE_SIZE);
128   while(ControllerMode!=IDLE_MODE){
129     delay(10);
130   }
```

Reference Solution

- The code implements the three steps to solve the task outlined in the competition scenario (read challenge, solve challenge, write solution)

2. Using Hashcash we generate a solution to the challenge we have just read
   - Uses two functions, `generate` and `validate` to implement Hashcash

     `generate` creates the string using the well known format and keeps track of the counter

     `verify` computes the hash and checks if the difficulty is satisfied
   - The `generate` call repeatedly calls `verify` with an ever increasing counter until a valid solution is found, which is then copied to the provided buffer (third argument)

```
156   generate(challenge_bits, challenge_string,solution_string);
```

Reference Solution

- The code implements the three steps to solve the task outlined in the competition scenario (read challenge, solve challenge, write solution)

3. Using the same I2C driver we write to solution back to the FRAM (on the second I2C address, not the one we read it from!)
   - Due to limitations in the driver, we need to write in blocks/chunks of 32 byte
   - Before each write, we check if there is still space left in the FRAM (up to 64kB)
   - We remember the last write position as well as the now incremented number of solved challenges to go back to step 1. and fetch the next challenge

```
188     I2C_WriteReg(0x51, eeprom_last_write+offset, (uint8_t*)solution_string+offset, chunk);
189     while(ControllerMode!=IDLE_MODE){
190       delay(10);
191     }
```

Outline

- EWSN’24 competition
  - Organizers
  - Why this new competition?
  - Challenge at a glance
  - Awards & Timeline
  - Challenge in detail / Hashcash
  - Reference solution
- E-Cube testbed
  - Scheduling an experiment
  - Available energy traces
  - Monitoring experiments
  - Experiment results
- Contacts

Competition Infrastructure: E-Cube

Build upon D-Cube

- E-Cube is a new benchmarking infrastructure created specifically for the EWSN'24 competition (accessible via https://iti-ecube.tugraz.at/)

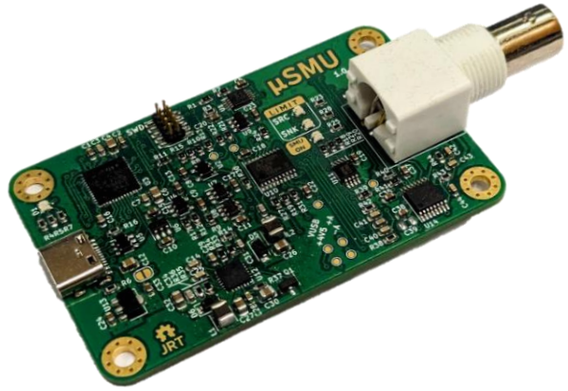
- The goal is to provide a low barrier of entry to those wanting to explore and evaluate different solutions for intermittent computing
  - A key focus of E-Cube is on automation
    (including automatic setup, execution, and evaluation of solutions)
  - Built upon D-Cube (https://iti-testbed.tugraz.at)
    (the benchmarking facility used in the EWSN Dependability Competition series)

E-Cube

Hardware Overview

- Observer modules
  - *Raspberry Pi 4*
  - *Connected to target battery-free device and SMU providing power*
  - *Uses analog switches to control SBW (programming), UART (logging)*
- Target battery-free device
  - *TI MSP430FR5994 Launchpad*
  - *16MHz MSP430 CPU, 8KB SRAM*
  - *256KB of embedded FRAM*
- Power
  - *Open-source SMU called uSMU (https://github.com/joeltroughton/uSMU/tree/main)*
  - *Energy Traces will be distributed freely*

TI MSP430FR5994 Launchpad



Open Hardware by Joel Troughton

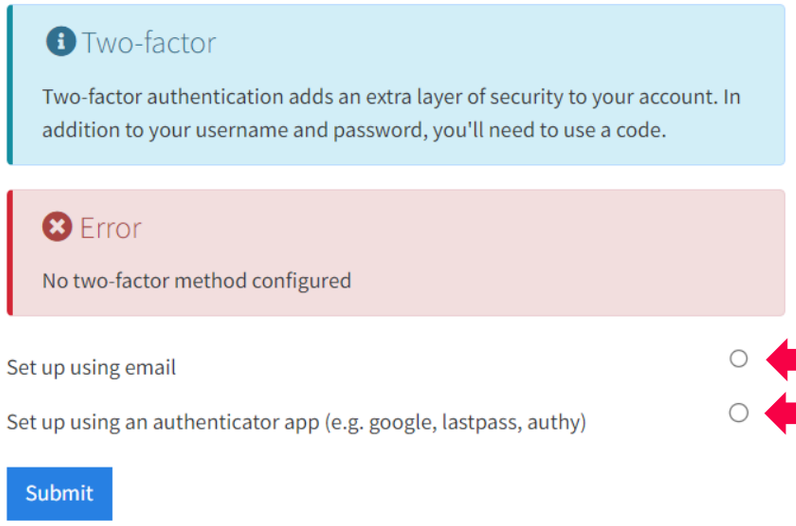
E-Cube
Features

- Web-based user interface and several features:
  - Schedule new experiments (with selectable energy trace)
  - Watch measurements (voltage and current) during experiments live
  - Automatic evaluation after an experiment has finished
  - Retrieve FRAM images (64kB binary of challenges and solutions)
  - Download detailed evaluation results as PDF
  - Download serial logs (9600 baud UART RX:P2.1 TX:P2.0)
    (note: enabling serial logs affects the energy trace, use carefully!)

E-Cube

Creating an Account

1. Request an account by filling out the form
   https://iti-ecube.tugraz.at/wiki/index.php/Testbed_Access
2. Log in with the provided credentials
   https://iti-ecube.tugraz.at/auth/login
3. Follow the instructions to set up two-factor authentication



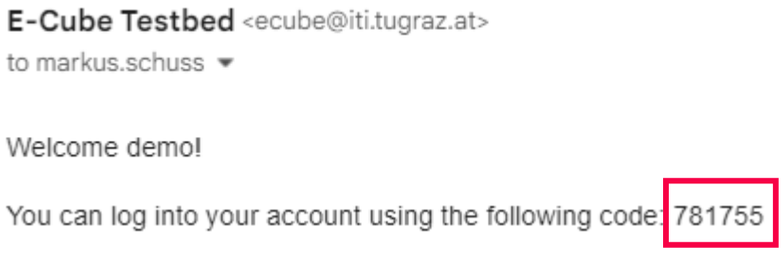
An email with a one-time password (OTP) will be sent to the email address you registered with

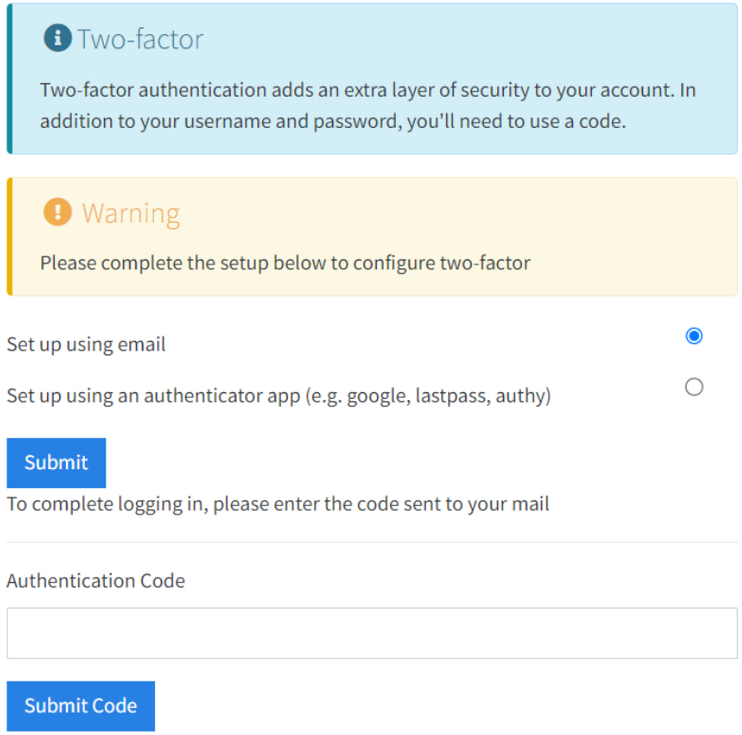
Use your smartphone or a desktop equivalent to generate the OTP
Google Authenticator and privacyIDEA are tested, others should work

E-Cube

Creating an Account

- To verify the selected option is working, enter the OTP you received via email or generated via the app
- For the app, you will be shown a QR code to scan with the app, email fallback can still be used





E-Cube

Scheduling an Experiment

- Compile your solution for the MSP430FR5994 in the form of a **.hex** file (*pure .bin/binary or .elf files are not supported by E-Cube*)
- We provide a reference solution based the Arduino fork for the MSP430: Energia (https://energia.nu/)
  - https://iti-ecube.tugraz.at/wiki/images/b/b0/Hashcash_example.zip (builds with software serial enabled, change '`#define DEBUG 1`' to 0 to disable)
  - Does not include any awareness for the intermittent energy supply
  - Includes a register level I2C driver and SHA1 implementation
- Under File → Preference: check "Show verbose output during compilation" ☑
  - The compiled firmware will be typically be placed at `%TEMP%\arduino_build_xxxxxx` (on Windows)

E-Cube

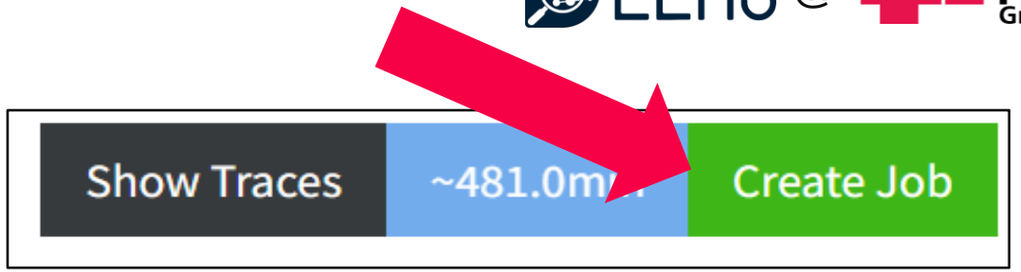
Scheduling an Experiment

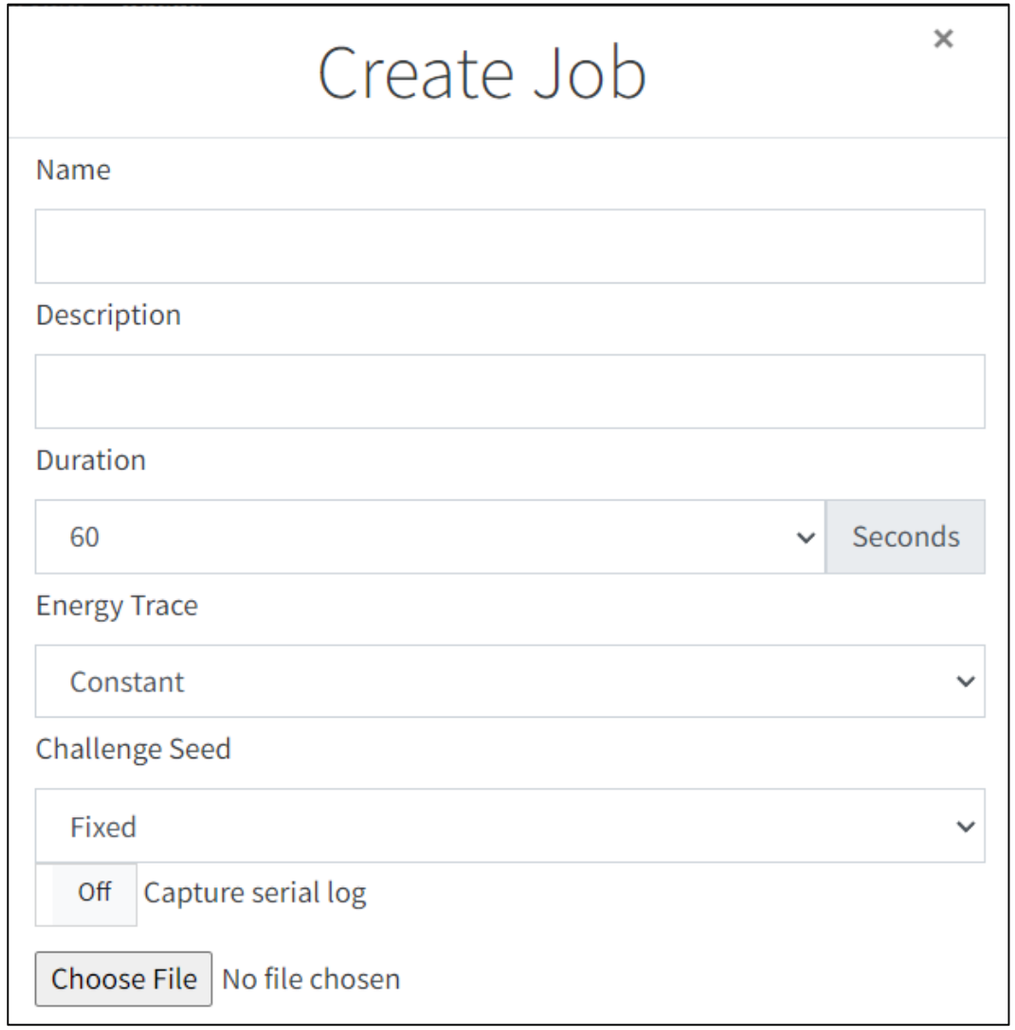
- On the E-Cube site under "Testbed Access", open "Access E-Cube" (https://iti-ecube.tugraz.at/overview)
- The testbed’s main page (Jobs Overview) shows all experiments run on the testbed so far: as such, all results are **public**
  - The identity of groups is not published by the organizers, please do not disclose your group number either!
  - The detailed evaluation report as well as images are only available to the owner of the experiment, but performance metrics and coarse evaluation (Difficulty Breakdown) are visible to anyone
- Navigate to the "Queue Tab" on the top-bar of the testbed (not the Wiki!)

E-Cube
Scheduling an Experiment

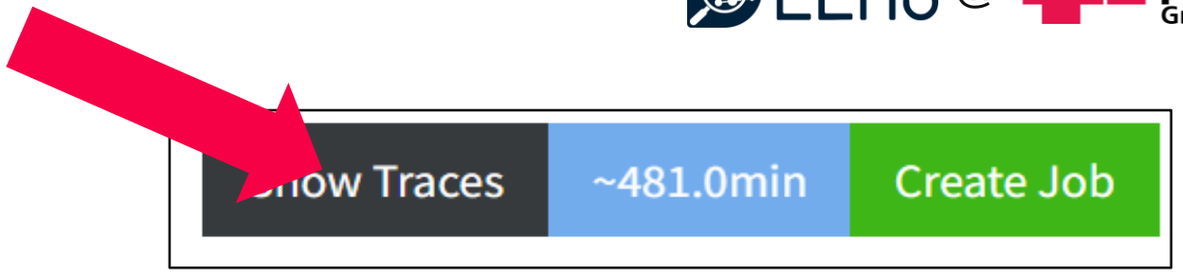
- To create a Job:
  - Provide a name and a short description (e.g., testing with parameter X=30)
  - Select a job duration in seconds
  - Select the energy trace (from a list)
    - The list will be updated continuously
  - Select the challenge image's seed
    - Fixed (0), Random, or custom (user)
  - Specify whether to log the serial output ≣ *(note: this affects energy traces!)*

  → Wait for the testbed to run ▶ and evaluate ✔ your experiment

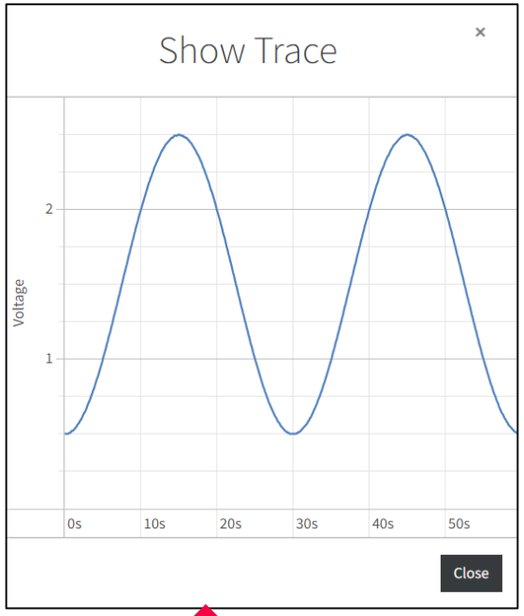




E-Cube
Available Energy Traces



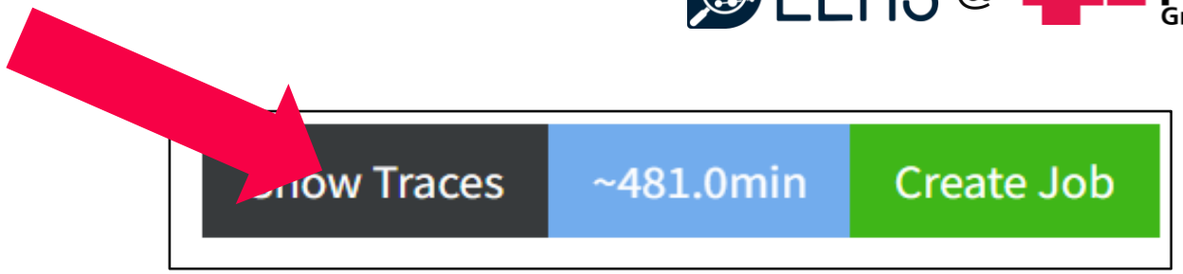
- A description as well as graphical representation are available for all current energy traces
  - You may download them in CSV format for local experiments
  - Over time, more traces will be added
  - For final evaluation, a new – not seen before – set of traces will be added
  - All traces are set to loop during an experiment (e.g., a 60s during a 600s will loop ten times)



Traces

ID	Name	Description	Filename	Actions
1	Constant	Trace for a constant 3.0V supply voltage (60s)	trace_1.csv	
2	Linear	Trace for a linear increasing supply voltage from 0V to 3.0V (60s)	trace_2.csv	

E-Cube
Available Energy Traces



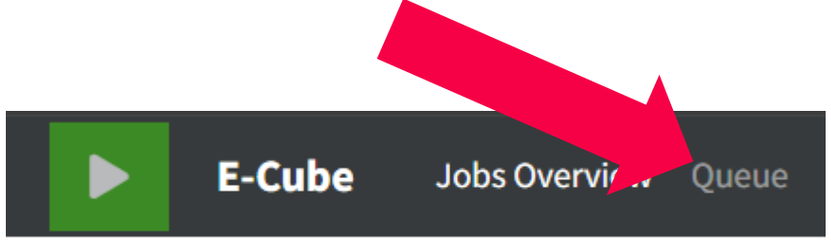
- The traces available during the informal testing phase are fairly “nice”...

- Which traces to expect during the actual remote competition phase?
  - While we strive to limit ourselves to real-world traces such as https://github.com/TUDSSL/WARio/tree/master/traces we are constrained by the uSMU’s replay capabilities (~40 samples per second, varying the full range of the MSP430 of 0-3.6V)
  - → As such, a theoretical “nightmare” trace would only pass brief bursts of power for 25ms at a time

→ Should you have a database of additional real-world traces (or just individual ones) we would be **happy** to integrate them into E-Cube
(although we will not guarantee that they are used for the evaluation of the final firmware)

E-Cube

Monitoring Experiments



The Queue UI contains all the information about current, previous, and scheduled experiments

Team name/number

Jobs for team organizers

« 1 2 3 4 »

Experiment currently in progress

Show status

Live measurements

#	Name	Description	Trace	Scheduled	Duration [s]	Flags	Actions
183	Testrun	hashcash debug s=1	Constant	14.04.24 11:59	28800	▶ ≣	? 👁
182	Testrun	hashcash debug s=1	Constant	12.04.24 19:47	28800	✓ ≣	🔍 ? 👁 ≣
181	Testrun	hashcash debug	Constant	12.04.24 11:51	28800	✓ ≣	🔍 ? 👁 ≣
180	Testrun	hashcash debug	Constant	12.04.24 11:51	60	✓ ≣	🔍 ? 👁 ≣

Download serial logs

Show details (see **Evaluation Details** later on)

Experiment Results

Automatic Evaluation of the Competition Scenario

E-Cube automatic evaluates each experiment once completed and returns results in different forms

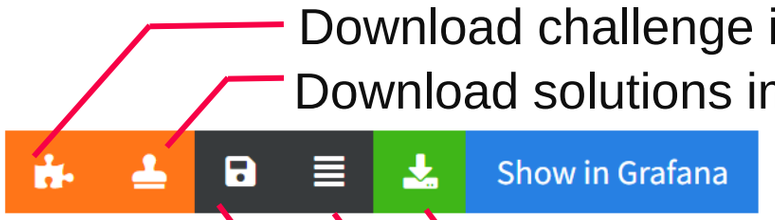
Job number

Details for job 182

Information

Parameter	Value
Team	organizers
Name	Testrun
Firmware	debug_hashcash_example.ino.hex
Description	hashcash debug s=1
Duration	28800s

Job details, provided during experiment creation, start and end time of the experiment



Evaluation

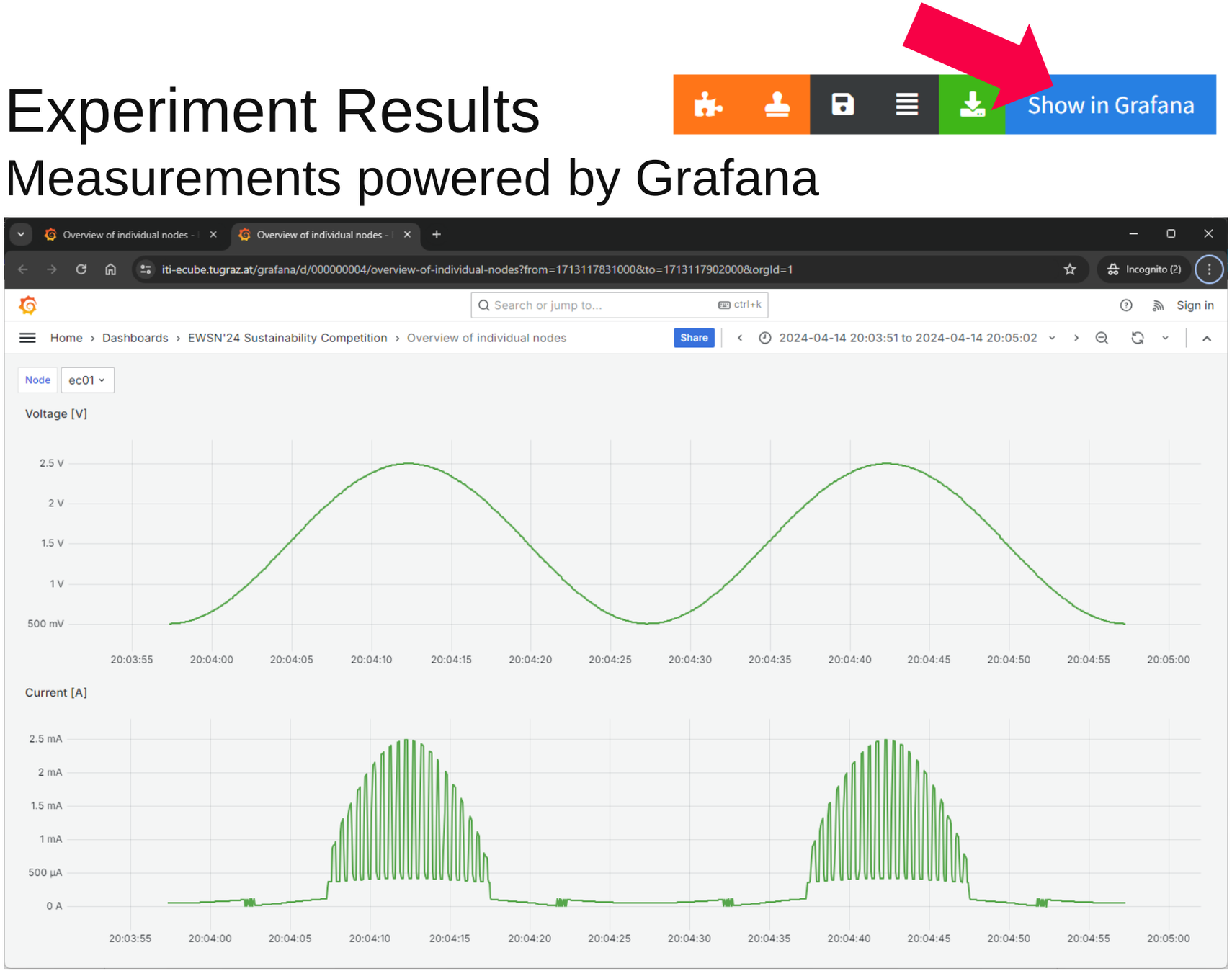
Difficulty Breakdown

Difficulty 4 bits	
Correct	157
Incorrect	-
Difficulty 5 bits	
Correct	157

Breakdown of correct and incorrect solutions based on the difficulty (only showing attempted difficulties, higher difficulties may be omitted)

Experiment Results

Measurements powered by Grafana



Example: A simple LED-blink program that, as soon as sufficient voltage to start is available, will begin to toggle both the red and green LED on the board

Voltage according to the energy trace
A Schottky 1n5817 diode acts as “harvester”, so the real voltage is ~0.2V less at the target

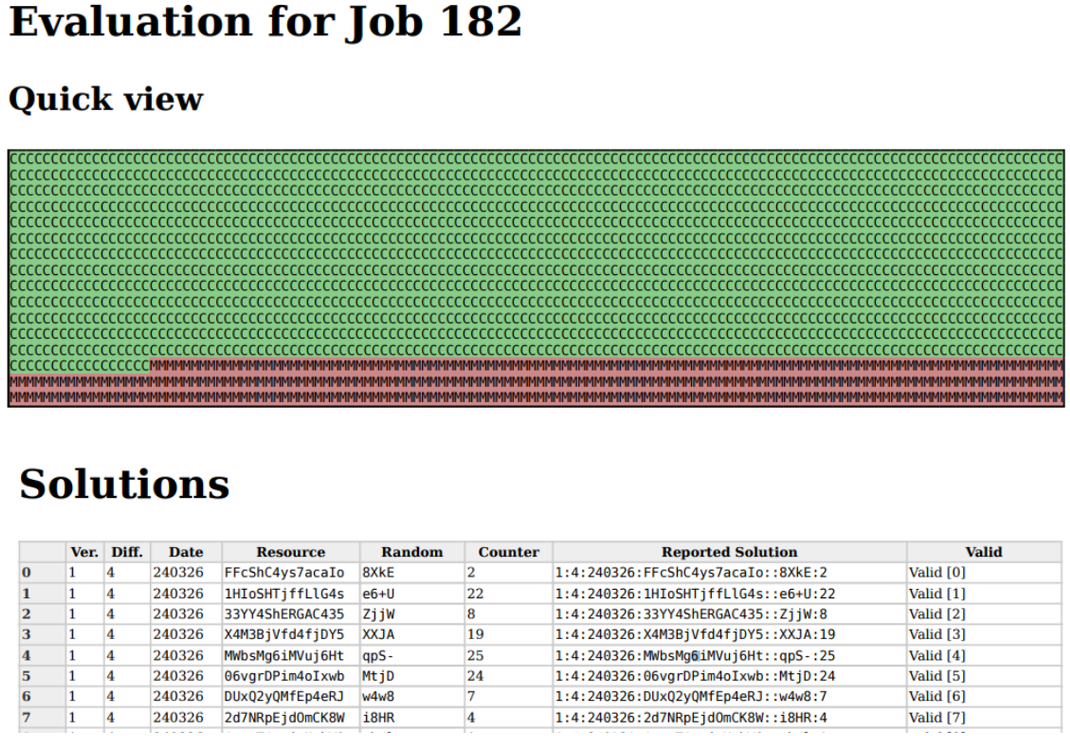
Current drawn by the MSP430 from the uSMU

Disclaimer: Current measurements during Energy Traces are on a best effort and may not be 100% accurate

Experiment Results
Detailed PDF Report

E-Cube returns a downloadable PDF containing all detailed (per challenge) results of the automatic evaluation

1. Quick view
   - To review the results at a glance
   - Colors to highlight correct & missed
2. Correctness of the solved challenges
   - Detailed breakdown per field
   - Showing which solution were valid, invalid, out of order, etc.



Evaluation for Job 182

Quick view

Solutions

	Ver.	Diff.	Date	Resource	Random	Counter	Reported Solution	Valid
0	1	4	240326	FFcShC4ys7acaIo	8XkE	2	1:4:240326:FFcShC4ys7acaIo::8XkE:2	Valid [0]
1	1	4	240326	1HIoSHTjffLlG4s	e6+U	22	1:4:240326:1HIoSHTjffLlG4s::e6+U:22	Valid [1]
2	1	4	240326	33YY4ShERGAC435	ZjjW	8	1:4:240326:33YY4ShERGAC435::ZjjW:8	Valid [2]
3	1	4	240326	X4M3BjVfd4fjDY5	XXJA	19	1:4:240326:X4M3BjVfd4fjDY5::XXJA:19	Valid [3]
4	1	4	240326	MWbsMg6iMVuj6Ht	qpS-	25	1:4:240326:MWbsMg6iMVuj6Ht::qpS-:25	Valid [4]
5	1	4	240326	06vgrDPim4oIxwb	MtjD	24	1:4:240326:06vgrDPim4oIxwb::MtjD:24	Valid [5]
6	1	4	240326	DUxQ2yQMfEp4eRJ	w4w8	7	1:4:240326:DUxQ2yQMfEp4eRJ::w4w8:7	Valid [6]
7	1	4	240326	2d7NRpEjdOmCK8W	i8HR	4	1:4:240326:2d7NRpEjdOmCK8W::i8HR:4	Valid [7]

Experiment Results
Performance Metrics

For each solved challenge, points p are awarded based on the difficulty:

$$p = \sum_i p_i \;\; with \;\; p_i = 2^{bits-1}$$

So a challenge of difficulty 1 bit = 1 point, 2 bit = 2 points, 3 bit = 4 points...

Disclaimer: Currently incorrect challenges do not provide a penalty, this will change during later stages!

Performance Metrics

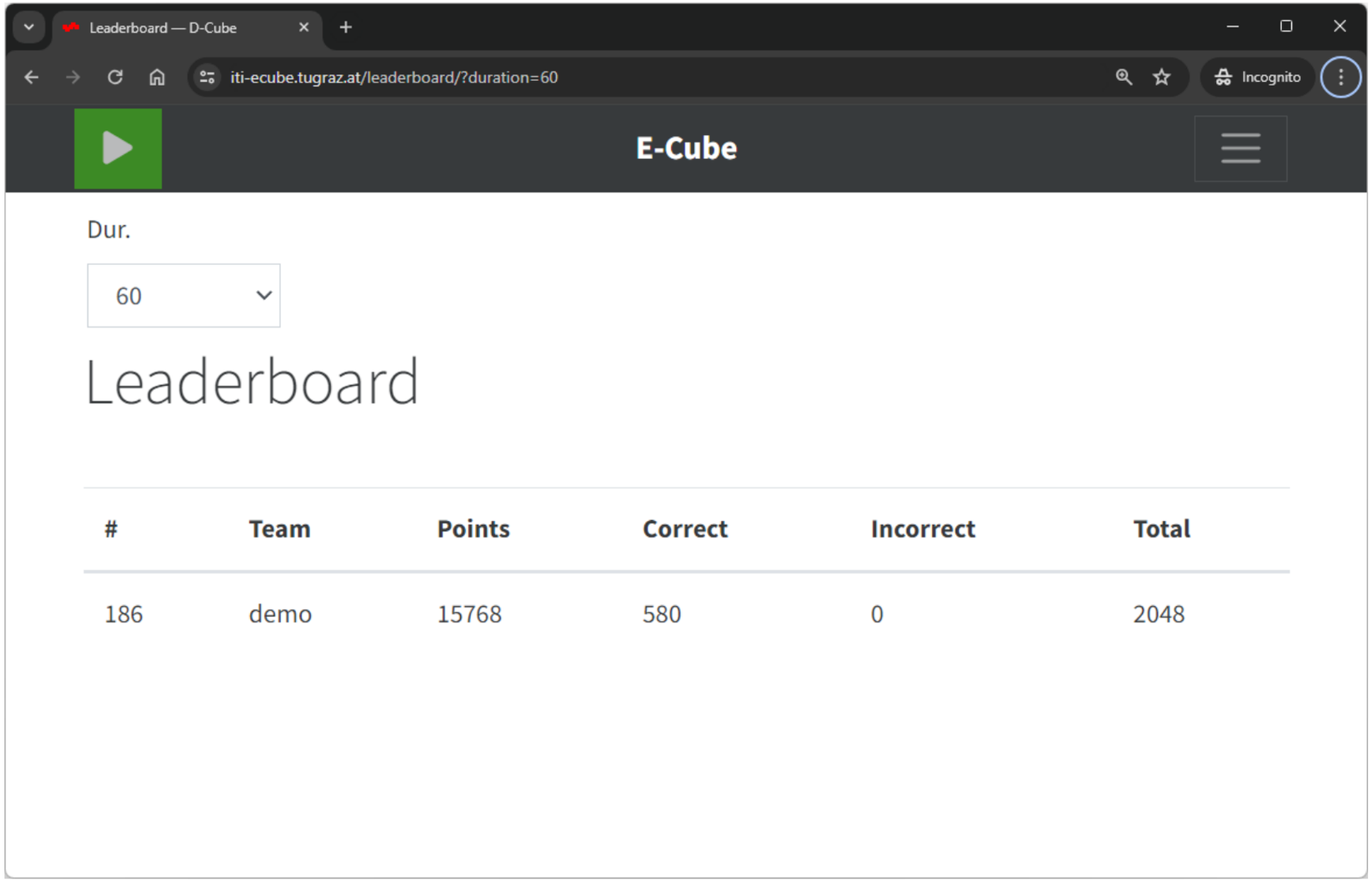
Metric	Result
Points	2194200
Correct	1681/2048
Incorrect	0

This is a **placeholder** using the above equation without any penalties (e.g., wrong solutions)

Number of correct solutions / total challenges

Number of incorrect solutions

Experiment Results
Leaderboard



Publicly visible leaderboard will be added in the “Remote Competition” phase **after** the informal testing has finished and the number of competing teams is fixed.

Lists the current best run for each team given a duration (and if needed other filters)

Contacts

- E-Cube is being improved and upgraded on-the-go!
- We look forward to interact with you!
- Questions? Feedback?
  - https://discord.gg/baBP2GbUvb
    Feel free to hang out and exchange ideas!
  - markus.schuss@tugraz.at
- Other general inquiries?
  - ecube@iti.tugraz.at