DOCUMENT INFORMATION

TITLE: IoT devices and embedded systems with uLisp

AUTHOR: Max-Gerd Retzlaff

VERSION: extended version with new demo slides and source code

PUBLICATION DATE: 2022-03-21 (original version), 2022-05-16 (source code/demo version)

DESCRIPTION: Slides for the demo at the European Lisp Symposium 2022 in Porto, Portugal on March 21th, 2022, from 14:30 till 15:00. More information on the conference web site at `https://www.european-lisp-symposium.org/2022/`.

PUBLICATION LOCATION: The document is published on `http://www.defsystem.net` and can be referenced there. The direct link is `http://www.defsystem.net/id/74678c84-e7e3-400f-9558-e4c5f271f3a0`.

MAX-GERD RETZLAFF

IOT DEVICES AND EMBEDDED SYSTEMS WITH ULISP

M5STACK ESP32 BASIC CORE

~40 EUROS (INCL. VAT)

- 240 MHz 32-bit dual-core ESP32 processor (Espressif ESP32-D0WDQ6-V3),
- 520 KB SRAM in total for programs and data,
- 4 or 16 MB Flash memory,
- 2.4 GHz Wi-Fi and Bluetooth with built-in antennae,
- USB-C socket,
- 2 inch TFT LCD display with 320x240 pixels and 262 K colours,
- SD card slot,
- 150 mAh battery,
- 1 W Speaker and microphone,
- 3 buttons

[Retzlaff 2021d]

ESP32 M5STACK ATOM LITE ESP32

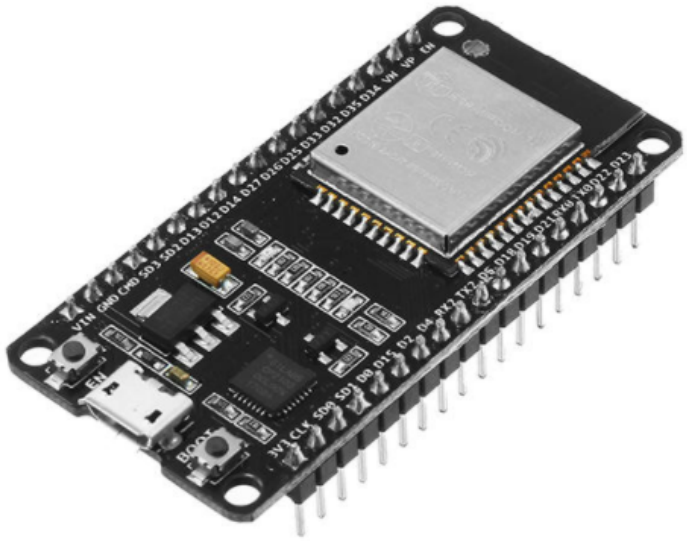
~10 EUROS (INCL. VAT)

- 240 MHz 32-bit dual-core ESP32 processor (Espressif ESP32-PICO-D4),
- 520 KB SRAM in total for programs and data,
- 4 MB Flash memory,
- 2.4 GHz Wi-Fi and Bluetooth with built-in antennae,
- USB-C socket,
- 1 RGB LED (compatible to Adafruit NeoPixel),
- Infrared Transmitter,
- 1 button

[M5Stack 2020]

DOIT ESP32 DEVKIT V1

~6-7 EUROS (INCL. VAT)



- 240 MHz 32-bit dual-core ESP32 processor (Espressif ESP-WROOM-32),
- 520 KB SRAM in total for programs and data,
- 4 MB Flash memory,
- 2.4 GHz Wi-Fi and Bluetooth with built-in antennae,
- USB-C socket

[DOIT 2018]

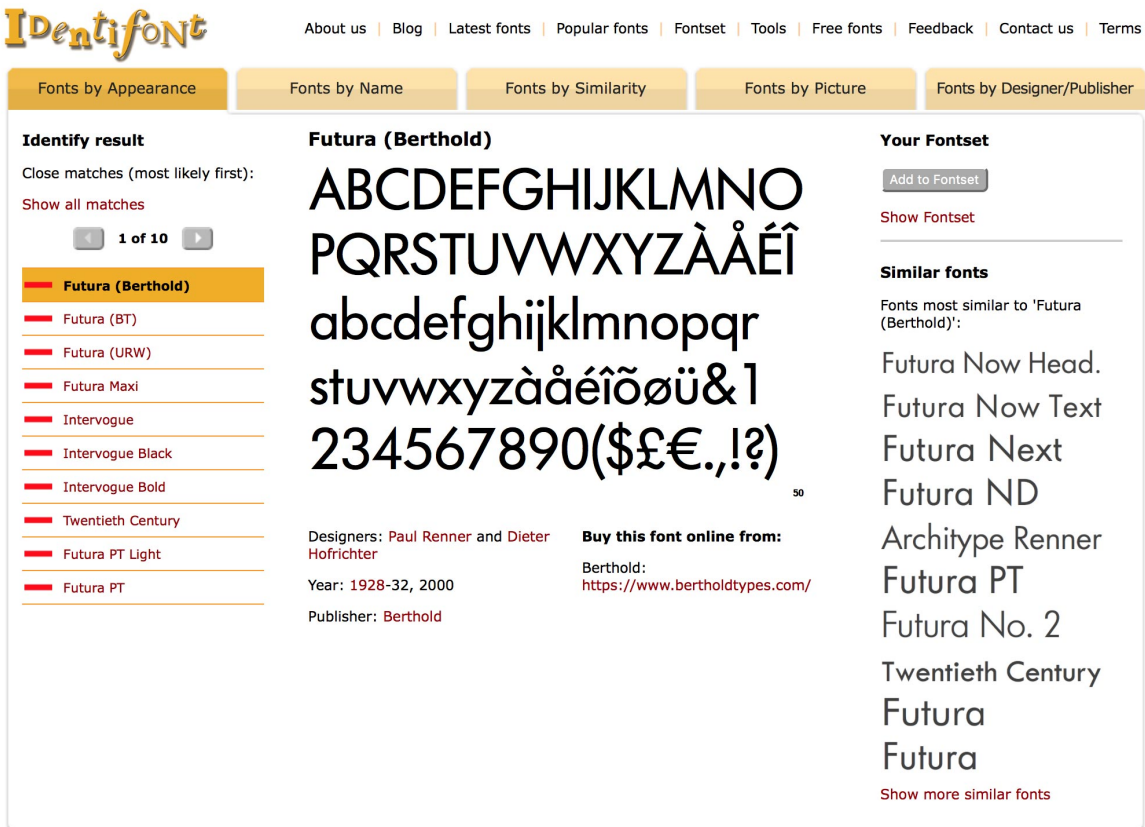
Authored by David Johnson-Davies from Cambridge, UK.

[Johnson-Davies 2021c]

IDENTIFONT

DAVID JOHNSON-DAVIES, 2000

A directory of digital fonts, including features to allow you to identify fonts by appearance, find fonts by name, find picture or symbol fonts, and find fonts by designer or publisher, at `www.identifont.com`.



FONTSCAPE

DAVID JOHNSON-DAVIES, 2000

The largest independent directory of typefaces organized into categories, at `www.fontscape.com`.

FONTIFIER

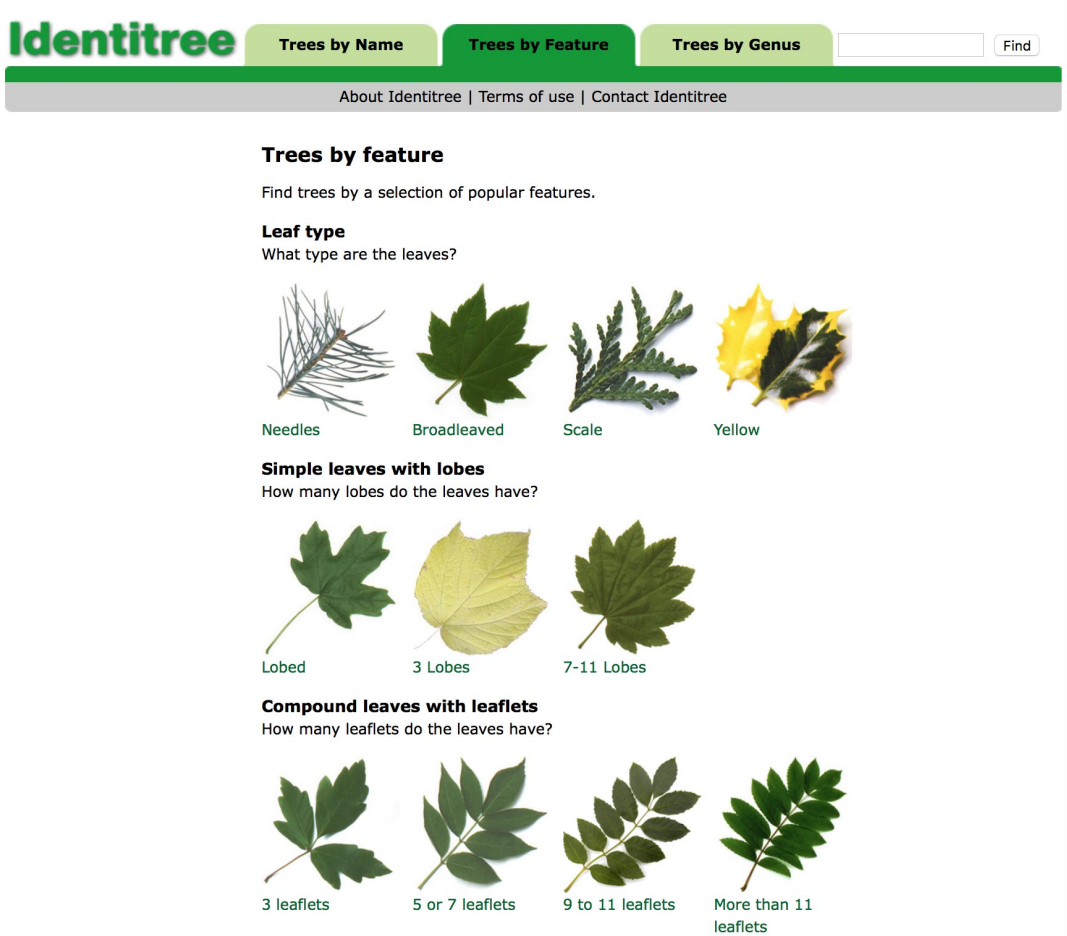
DAVID JOHNSON-DAVIES, 2003

Creates handwriting fonts from templates, at `www.fontifier.com`.

IDENTITREE

DAVID JOHNSON-DAVIES, 2015

An expert system written in Lisp designed to enable users to identify trees from their visual characteristics, available online at `www.identitree.com`.



CL-HTTP

The websites IDENTIFONT, FONTSCAPE, FONTIFIER, IDENTITREE, and also ULISP are all based on: CL-HTTP.

CL-HTTP is the Common Lisp Hypermedia Server, developed by John C. Mallery at the M.I.T. Computer Science & Artificial Intelligence Laboratory.

A CL-HTTP PRIMER by David Johnson-Davies is also served via CL-HTTP at `http://clhttp.plasticki.com/`.

ULISP NAME

"I decided to settle on uLisp, pronounced 'U'-lisp,
rather than μLisp, pronounced mu-lisp,
for two reasons:

- It's clearer what the URL for the website should be.
- The 'μ' character often doesn't look very good on-screen on a page of text.

Also, I was able to trademark "ULISP" as a word."

[Johnson-Davies 2021b]

ULISP

LISP FOR MICROCONTROLLERS

- specifically designed for limited RAM
- main challenge of writing uLisp:
  have it run on the ATmega328 with only 2 kB of RAM
  (and 32 kB of flash program memory)
- 8-, 16-, 32- and 64-bit platforms
- from the Arduino Uno based on the ATmega328 up to the Teensy 4.0/4.1

[Johnson-Davies 2014b]

ULISP

IMPLEMENTATION

- interpreter not compiler
- tail-optimization
  (efficient recursive functions)
- mark and sweep garbage collector
  (Under 1 msec on an Arduino Uno or under 3 msec on an Arduino Mega 2560.)

[Johnson-Davies 2014b]

ULISP

LANGUAGE

- The language is generally a subset of Common Lisp, this is: uLisp programs should also run under Common Lisp.
- major difference: It's a Lisp-1.
  (One namespace for functions and variables.)

[Johnson-Davies 2014b]

ULISP

ORIGINAL OBJECTIVE

"[. . .] a compact Lisp for small microcontrollers that would allow people to use a more advanced language than C for creating hardware-oriented projects. [. . .]

There's no point making uLisp into a full Common Lisp.
It won't make uLisp more likely to be adopted by microcontroller users, and

people with larger processor boards such as the Raspberry Pi can already get a full Common Lisp running under Unix (such as Clozure Common Lisp).

So for each proposed extension I have to think: will this make it more likely that people use uLisp for hardware projects? I think that the way to do this is to make it *appeal to people who haven't encountered Lisp before*, rather than trying to appeal to experienced Common Lisp users."

[Johnson-Davies 2021a]

ULISP VS. COMMON LISP

- major difference: It's a Lisp-1.
  One namespace for functions and variables.
- no macros
- no multiple-values,
  instead: use lists to represent multiple values.

  NOTHING – symbol with no value, equivalent to `(values)` in Common Lisp,
  to suppress output from functions
- no DESTRUCTURING-BIND
- no condition system or exceptions,
  error brings you back to the top-level
- ; (semicolon) – comments end not at the end of the line
  but *before the next opening bracket*

ULISP

GREAT WEB SITE

uLisp Home Reference Forum Search

Lisp for microcontrollers

Lisp for Arduino, Adafruit M0/M4, Micro:bit, ESP8266/32, RISC-V, and Teensy 4.x boards.

News!

New ARM Version 4.1

Like the AVR Version 4.1 of uLisp released earlier this month, the ARM version now adds a **register** function to allow you to read the values of 32-bit processor registers, or write values to the registers. It allows you to control the peripherals in the ARM processor from a Lisp program, or interactively experiment with the peripherals by giving commands at the uLisp prompt.

For more information see: ARM uLisp now also supports the register function.

uLisp® is a version of the Lisp programming language specifically designed to run on microcontrollers with a limited amount of RAM, from the Arduino Uno based on the ATmega328 up to the Teensy 4.0/4.1. You can use exactly the same uLisp program, irrespective of the platform.

Because uLisp is an interpreter you can type commands in, and see the effect immediately, without having to compile and upload your program. This makes it an ideal environment for learning to program, or for setting up simple electronic devices.

Lisp is also an ideal language for learning about fundamental programming concepts. It incorporates string handling, list processing, and garbage collection, and so is also an excellent language for expressing complex ideas, such as teaching a robot to solve mazes or finding the shortest route on a map. As well as supporting a core set of Lisp functions uLisp includes Arduino extensions, making it ideal as a control language for the Arduino.

You can download the current version of uLisp free from the Download uLisp page.

ULISP

GREAT WEB SITE

uLisp

Home

Search

Getting started

Lisp for microcontrollers
Performance
Using uLisp
Using uLisp from a terminal
Debugging in uLisp
Using the program editor
SD card interface
I2C and SPI serial interfaces
Lisp Library
Download uLisp
uLisp Sensor Library
Forum

Self-contained Lisp computers

Tiny Lisp Computer
Lisp Badge
Pocket Op Amp Lab

8/16-bit platforms

Arduino Uno
Arduino Mega 2560
ATmega1284
ATmega4809 boards
AVR DA and DB series boards

32/64-bit platforms

Arduino M0 Boards
Adafruit M0 boards
Adafruit QT-Py and Seeduino XIAO
Adafruit M4 boards
Adafruit PyGamer and PyBadge
Adafruit nRF52840 boards
BBC Micro:bit
Calliope mini
Maxim MAX32620FTHR
Teensy 4.0 and 4.1
RP2040 boards New!
ESP32 boards
ESP8266 boards
Sipeed MAiX RISC-V boards

Simple examples

Blinking primes
Tweetmaze
Mood light
LED 8x8 dot-matrix display
Simon game
I2C clock
Data logging
Ringing the changes
Driving DotStar RGB LEDs
LCD character display
DDS signal generator
Temperature sensor
Simple data plotter
Thermocouple interface
Benchmarks

Games

Animals
Eliza chatbot
Simple arcade game
Bulls & Cows game
Mini text adventure game

Larger examples

Route finder
Calculating with fractions
Infinite precision arithmetic
Scrolling text display
Dot-matrix clock
Graphics display interface in Lisp
Plotting to a colour TFT display
Ray tracing with uLisp
GPS mapping application
Query language
uLisp GSM server
A LoRaWAN node using uLisp
Solving resistor networks
Fast Fourier Transform
Sudoku solver
Prime number spiral
Wi-Fi examples
Automatic uLisp to C converter
Simple object system

Assemblers

AVR assembler overview
AVR assembler examples
AVR NeoPixel driver using assembler
ARM assembler overview
ARM assembler instructions
ARM assembler examples
ARM NeoPixel driver using assembler
RISC-V assembler overview
RISC-V assembler instructions
RISC-V assembler examples
Mandelbrot set using assembler
Returning a floating-point result

Tutorials

Getting started
Lists
Expressions
Defining functions
Variables
Strings
Testing a result
Manipulating lists
Processing items in a list
Recursion
Returning functions
Lisp for C programmers
Thinking in a Lispy way

Reference

Language reference
Floating-point extensions
Graphics extensions
Wi-Fi extensions
Programming AVR registers
Programming ARM registers
Error messages
Adding useful functions to uLisp

Implementation

Implementation
Objects
Symbols
Built-in symbols
Strings
Arrays
Read, Evaluate, Print
Garbage collection
Tail-call optimization
Saving and loading images
Assembler and defcode
Debugging uLisp
Adding your own functions
Converting between C and uLisp
Porting uLisp to a new platform
uLisp Builder
uLisp Zero

Other information

About me
Contact
Legal stuff

RSS Feed

uLisp news!

ULISP

GREAT WEB SITE

uLisp

Home

Search

Getting started

Lisp for microcontrollers
Performance
Using uLisp
Using uLisp from a terminal
Debugging in uLisp
Using the program editor
SD card interface
I2C and SPI serial interfaces
Lisp Library
Download uLisp
uLisp Sensor Library
Forum

Tutorials

Getting started
Lists
Expressions
Defining functions
Variables
Strings
Testing a result
Manipulating lists
Processing items in a list
Recursion
Returning functions
Lisp for C programmers
Thinking in a Lispy way

Reference

Language reference
Floating-point extensions
Graphics extensions
Wi-Fi extensions
Programming AVR registers
Programming ARM registers
Error messages
Adding useful functions to uLisp

Implementation

Implementation
Objects
Symbols
Built-in symbols
Strings
Arrays
Read, Evaluate, Print
Garbage collection
Tail-call optimization
Saving and loading images
Assembler and defcode
Debugging uLisp
Adding your own functions
Converting between C and uLisp
Porting uLisp to a new platform
uLisp Builder
uLisp Zero

ULISP

GREAT WEB SITE

uLisp

Home

Search

8/16-bit platforms

Arduino Uno
Arduino Mega 2560
ATmega1284
ATmega4809 boards
AVR DA and DB series boards

32/64-bit platforms

Arduino M0 Boards
Adafruit M0 boards
Adafruit QT-Py and Seeduino XIAO
Adafruit M4 boards
Adafruit PyGamer and PyBadge
Adafruit nRF52840 boards
BBC Micro:bit
Calliope mini
Maxim MAX32620FTHR
Teensy 4.0 and 4.1
RP2040 boards New!
ESP32 boards
ESP8266 boards
Sipeed MAiX RISC-V boards

Assemblers

AVR assembler overview
AVR assembler examples
AVR NeoPixel driver using assembler
ARM assembler overview
ARM assembler instructions
ARM assembler examples
ARM NeoPixel driver using assembler
RISC-V assembler overview
RISC-V assembler instructions
RISC-V assembler examples
Mandelbrot set using assembler
Returning a floating-point result

ULISP

GREAT WEB SITE

uLisp

Home

Search

Simple examples

Blinking primes
Tweetmaze
Mood light
LED 8x8 dot-matrix display
Simon game
I2C clock
Data logging
Ringing the changes
Driving DotStar RGB LEDs
LCD character display
DDS signal generator
Temperature sensor
Simple data plotter
Thermocouple interface
Benchmarks

Larger examples

Route finder
Calculating with fractions
Infinite precision arithmetic
Scrolling text display
Dot-matrix clock
Graphics display interface in Lisp
Plotting to a colour TFT display
Ray tracing with uLisp
GPS mapping application
Query language
uLisp GSM server
A LoRaWAN node using uLisp
Solving resistor networks
Fast Fourier Transform
Sudoku solver
Prime number spiral
Wi-Fi examples
Automatic uLisp to C converter
Simple object system

Self-contained Lisp computers

Tiny Lisp Computer
Lisp Badge
Pocket Op Amp Lab

Games

Animals
Eliza chatbot
Simple arcade game
Bulls & Cows game
Mini text adventure game

Other information

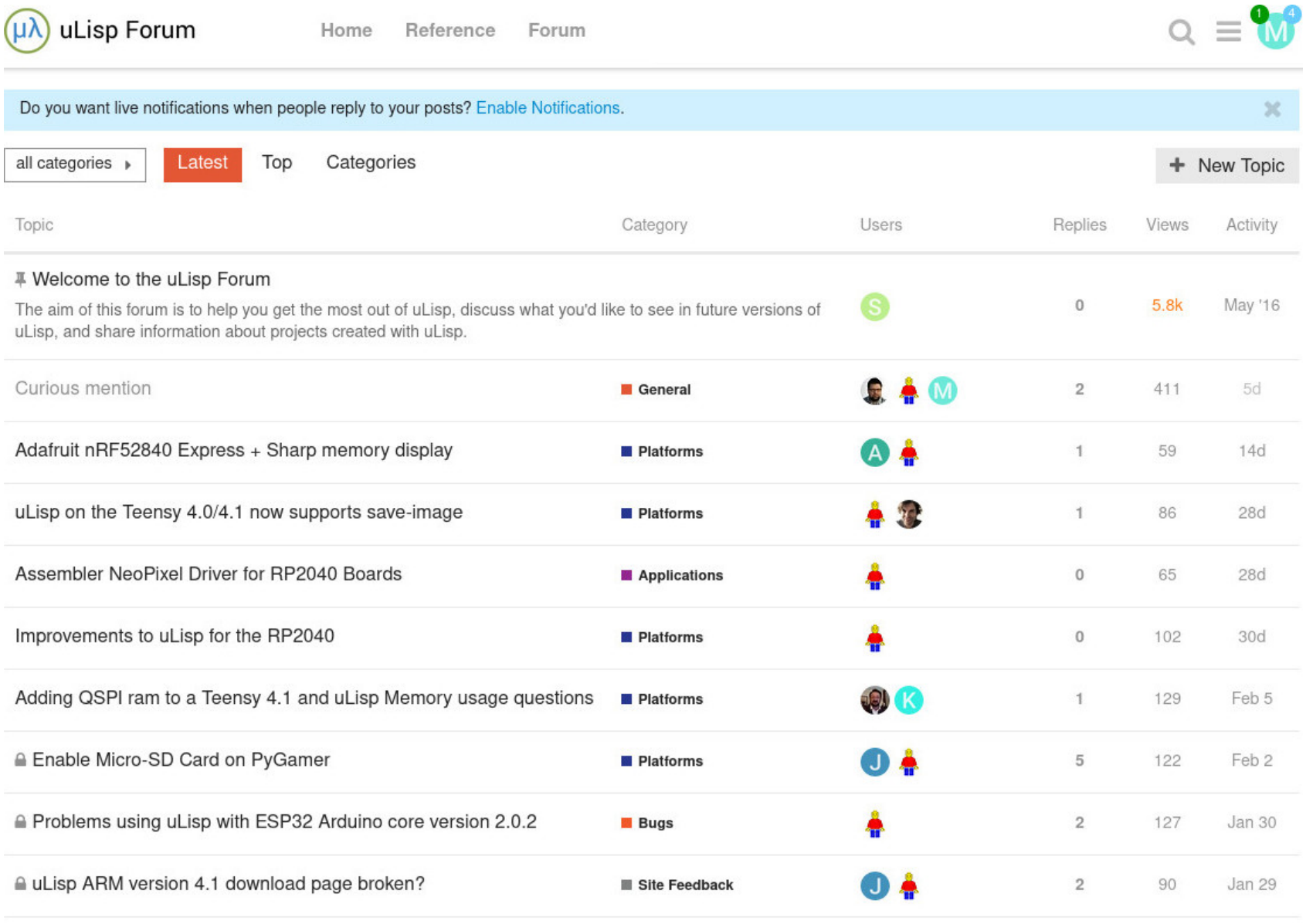
About me
Contact
Legal stuff

RSS Feed

uLisp news!

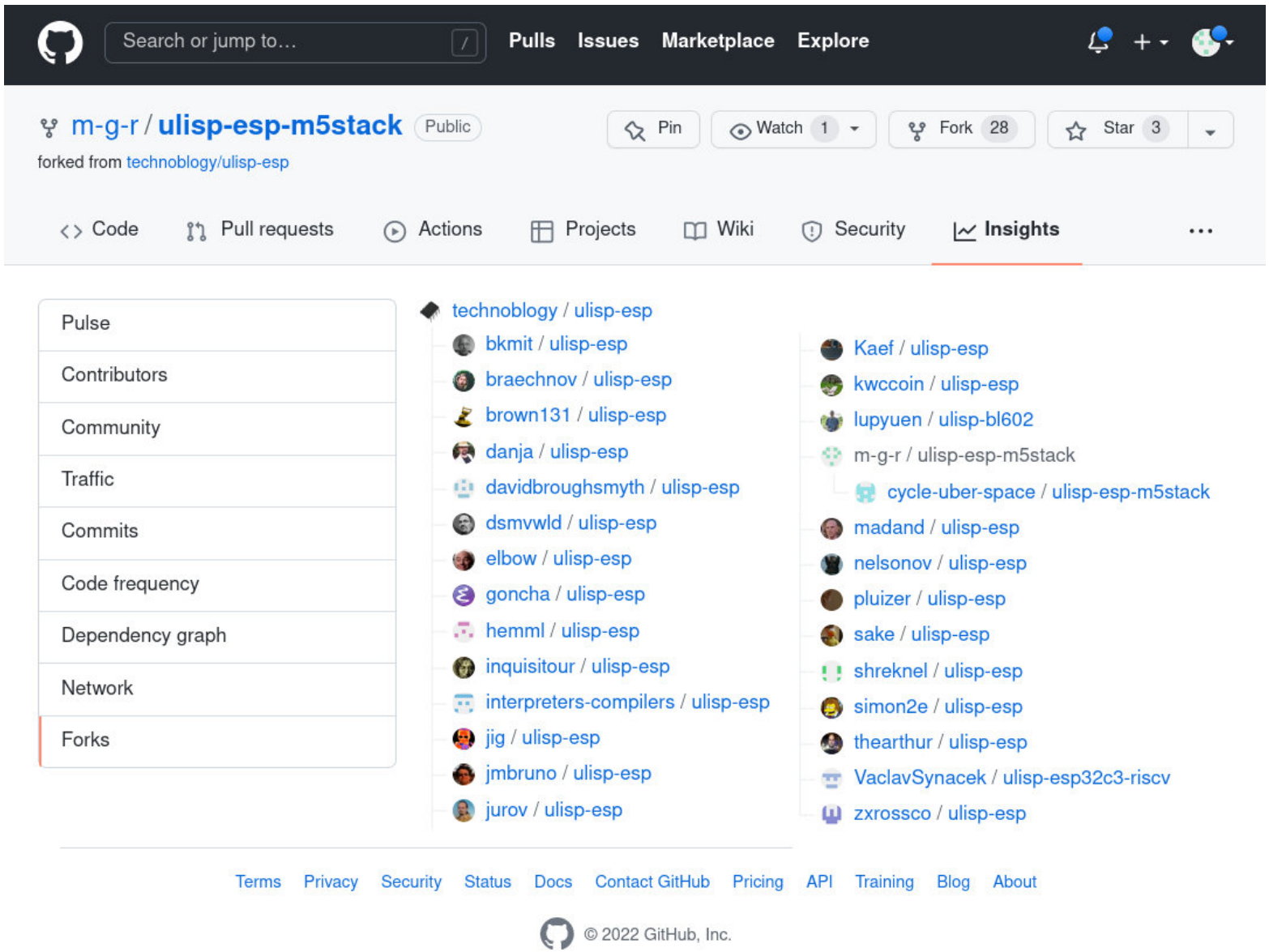
ULISP

ACTIVE USER FORUM



ULISP

ACTIVE GITHUB



CONNECT TO ULISP

Linux with screen:

```
screen /dev/ttyUSB0 115200
```

macOS:

```
screen /dev/tty.usbserial-55D2D232BE 115200
```

Emacs:

```
(defun ulisp-terminal-start ()
  (serial-term "/dev/ttyUSB0" 115200)
  (term-line-mode))

M-: (ulisp-terminal-start)
```

D E M O T I M E

ULISP DEMO

SIMPLE INPUT

```
6306> 23
23

6306> (+ 23 42)
65

6306> (defun fun () 'hello)
fun

6295> (fun)
hello

6295> fun
(lambda nil (quote hello))
```

ULISP DEMO

```
6295> (fun)
hello

6295> fun
(lambda nil (quote helo))

6295> (save-image)
2710

6295> (reboot)
Explicit reboot.
```

SAVE-IMAGE + LOAD-IMAGE

```
ets Jun  8 2016 00:22:57
rst:0x1 (POWERON_RESET),
boot:0x17 (SPI_FAST_FLASH_BOOT)
...
entry 0x400806b4
uLisp 3.6

6306> (load-image)
2710

6295> (fun)
hello
```

ULISP DEMO

DIGITAL INPUT + BUTTONS

```
#| Max-Gerd Retzlaff <mgr@defsystem.net> - www.defsystem.net - unreleased |#
#| (c) Copyright Max-Gerd Retzlaff, 2021. All rights reserved. |#

(defvar *button-1* 39)
(defvar *button-2* 38)
(defvar *button-3* 37)

(defun init-buttons ()
  (pinmode *button-1* :input)
  (pinmode *button-2* :input)
  (pinmode *button-3* :input))

(defun init-atom-button ()
  (pinmode *button-1* :input))

(defun button-pressed-p (button)
  (not (digitalread button)))

(defun atom-button-pressed-p ()
  (button-pressed-p *button-1*))
```

ULISP DEMO

DIGITAL INPUT + BUTTONS

```
6295> (init-buttons)
nil

6295> (button-pressed-p *button-1*)
nil

6295> (button-pressed-p *button-1*)
t
```

ULISP DEMO

ANALOG INPUT + CHEAP PH SENSOR

```
#| Max-Gerd Retzlaff <mgr@defsystem.net> - www.defsystem.net - unreleased |#
#| (c) Copyright Max-Gerd Retzlaff, 2021. All rights reserved. |#

(defvar *ph-pin* 35)

(defun raw-phv ()
  #| "single raw phv read" |#
  (/ (* (analogread *ph-pin*) 3.3) 4096)) #| 3.3 V and 12 bit |#

(defun get-phv (&optional (times 64))
  #| "phv as mean over multiple raw reads to get a stable value" |#
  (let ((v 0))
    (dotimes (i times)
      (let ((ph (raw-phv)))
        (incf v ph)))
    (/ v times)))
```

ULISP DEMO

CHEAP PH SENSOR + LINEAR INTERPOLATION

```
#| Max-Gerd Retzlaff <mgr@defsystem.net> - www.defsystem.net - unreleased |#
#| (c) Copyright Max-Gerd Retzlaff, 2021. All rights reserved. |#

(defvar *ph4* 1.891216)
(defvar *ph7* 1.380262)

(defvar *slope* nil)
(defvar *intercept* nil)

(defun calc-calibration ()
  (setf *slope* (/ (- 7 4)
                   (- *ph7* *ph4*))
        *intercept* (- 7 (* *slope* *ph7*)))
  (list *slope* *intercept*))

(defun get-ph (&optional (volt (get-phv)))
  (unless (and *slope* *intercept*)
    (calc-calibration))
  (+ (* volt *slope*)
     *intercept*))
```

ULISP DEMO

ANALOG READ + LINEAR INTERPOLATION

```
6233> (raw-phv)
0.100708

6233> (get-phv)
0.118558

6157> (get-ph)
14.4092
```

(Note: This was done with an unconnected sensor which explains the out of range value.)

ULISP DEMO EXTENDING ULISP IN C++ WITH ARDUINO LIBRARIES

```
// add switch to enable your feature

#define enable_dallastemp

// ...

// include libraries, define constants and so on

#if defined(enable_dallastemp)
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 16
#define DEFAULT_TEMPERATURE_PRECISION 12
// setup one wire instance to communicate with any OneWire devices
OneWire oneWire(ONE_WIRE_BUS);
// configre Dallas Temperature to use one wire instance
DallasTemperature temp_sensors(&oneWire);
#endif # enable_dallastemp

// ...
```

ULISP DEMO EXTENDING ULISP WITH ARDUINO LIBRARIES

```
// ...

// add your symbols to enum function

enum function { NIL, TEE, NOTHING, OPTIONAL, INITIALELEMENT, ELEMENTTYPE, // ...
// ...
#if defined(enable_dallastemp)
INITTEMP, GETTEMP, SETTEMPRESOLUTION, GETTEMPDEVICESCOUNT,
#endif # enable_dallastemp
// ...
ENDFUNCTIONS };

// ...
```

ULISP DEMO

EXTENDING ULISP WITH ARDUINO LIBRARIES

```
// ...

// add your C++ functions

// dallas temp
#if defined(enable_dallastemp)

void printOneWireAddress(DeviceAddress deviceAddress) {
  // function to print a device address
  for (uint8_t i = 0; i < 8; i++) {
    // zero pad the address if necessary
    if (deviceAddress[i] < 16) pserial('0');
    pintbase(deviceAddress[i], 4, pserial);
    if (i < 7) pserial(':');
  }
  pln(pserial);
}
```

ULISP DEMO EXTENDING ULISP WITH ARDUINO LIBRARIES

```
object *fn_inittemp(object *args, object *env) {
  /* Function init-temp
   *
   * Syntax:
   *    init-temp
   *      => result-list
   *
   * Arguments and values:
   *    result-list----a list of devices addresses; each address being
   * a list of 8 integer values specifying a device address.
   *
   * Description:
   *    Detects all supported temperature sensors connected via one wire
   * bus to the pin ONE_WIRE_BUS and returns the list of the sensors'
   * device addresses.
   *
   *    All sensors are configured to use the resolution specified by
   * default DEFAULT_TEMPERATURE_PRECISION via a broadcast. Note that
   * a sensor might choose a different resolution if the desired resolution
   * is not supported. See also: set-temp-resolution.
   */
// ...
```

ULISP DEMO EXTENDING ULISP WITH ARDUINO LIBRARIES

```
object *fn_gettemp(object *args, object *env) {
  /* Function get-temp
   *
   * Syntax:
   *    get-temp address
   *      => temperature
   *
   * Arguments and values:
   *    address---a list of 8 integer values specifying a device address.
   *
   *    temperature---an integer value; the measured temperature in Celsius.
   *
   * Description:
   *    Requests the sensor specified by address to measure and compute a new
   * temperature reading, retrieves the value from the sensor device and
   * returns the temperature in Celsius.
   */

// ...
```

ULISP DEMO EXTENDING ULISP WITH ARDUINO LIBRARIES

```
object *fn_settempresolution(object *args, object *env) {
  /* Function set-temp-resolution
   *
   * Syntax:
   *    set-temp-resolution address [resolution]
   *      => actual-resolution
   *
   * Arguments and values:
   *    address---a list of 8 integer values specifying a device address.
   *
   *    resolution---an integer value.
   *
   *    actual-resolution---an integer value.
   *

   ...
```

ULISP DEMO EXTENDING ULISP WITH ARDUINO LIBRARIES

```
  /* Function set-temp-resolution

   ...

   * Description:
   *    Tries to configure the sensor specified by address to use the given
   * resolution and returns the actual resolution that the devices is set to
   * after the attempt.
   *
   *    Note that a sensor might choose a different resolution if the desired
   * resolution is not supported. In this case, the returned actual-resolution
   * differs from the argument resolution.
   *
   * If the argument resolution is missing, instead the default given by
   * DEFAULT_TEMPERATURE_PRECISION is used.
   */

// ...
```

ULISP DEMO EXTENDING ULISP WITH ARDUINO LIBRARIES

```
object *fn_gettempdevicescount(object *args, object *env) {
  /* Function get-temp-devices-count
   *
   * Syntax:
   *    get-temp-devices-count
   *      => count
   *
   * Arguments and values:
   *    count---an integer value; the number of detected supported temperature
   * sensors.
   *
   * Description:
   *    Returns the number of temperature sensors supported by this interface
   * that were detected by the last call to INIT-TEMP. Note that this might not
   * be the correct current count if sensors were removed or added since the
   * last call to INIT-TEMP.
   */
  return number(temp_sensors.getDS18Count());
}
#endif # enable_dallastemp
// ...
```

ULISP DEMO

EXTENDING ULISP WITH ARDUINO LIBRARIES

```
// ...

// add strings for the Lisp symbols
// (Note: I used randomized string names here but that's optional.)

#if defined(enable_dallastemp)
const char userc65fb817287f56327cb653a23ff1[] PROGMEM = "init-temp";
const char user338df9807a7c0fb38c6d3b67b09d[] PROGMEM = "get-temp";
const char usera1220cc63a1b191e09349963756c[] PROGMEM = "set-temp-resolution";
const char user46625f7d60ca8651d4a2b9150c7d[] PROGMEM = "get-temp-devices-count";
#endif # enable_dallastemp

// ...
```

ULISP DEMO EXTENDING ULISP WITH ARDUINO LIBRARIES

```
// ...

// add string and function reference to the lookup table
// the byte specifies the minimum and maximum number of arguments

const tbl_entry_t lookup_table[] PROGMEM = {
// ...
#if defined(enable_dallastemp)
  { userc65fb817287f56327cb653a23ff1, fn_inittemp, 0x00 },
  { user338df9807a7c0fb38c6d3b67b09d, fn_gettemp, 0x11 },
  { usera1220cc63a1b191e09349963756c, fn_settempresolution, 0x12 },
  { user46625f7d60ca8651d4a2b9150c7d, fn_gettempdevicescount, 0x00 },
#endif # enable_dallastemp
// ...
};

// ...
```

That’s it. This code is included in my version of ulisp-esp-m5stack at:
https://github.com/m-g-r/ulisp-esp-m5stack/blob/master/ulisp-esp.ino

ULISP DEMO

TEMPERATURE SENSOR VIA ARDUINO LIBRARY

```
6155> (init-temp)
Found Dallas Temp: 28:7f:77:07:d6:01:3c:09
 -> resolution actually set to: 12
((40 127 119 7 214 1 60 9))

6155> (get-temp)
Error: 'get-temp' too few arguments

6155> (get-temp '(40 127 119 7 214 1 60 9))
21.9375
```

ULISP DEMO

I2C + M5STACK FACES KEYBOARD #1

```
#| Max-Gerd Retzlaff <mgr@defsystem.net> - www.defsystem.net - unreleased |#
#| (c) Copyright Max-Gerd Retzlaff, 2021. All rights reserved. |#

(defvar *faces-kbd-addr* #x08)
(defvar *faces-kbd-pin* 5)

(defun init-kbd ()
  (pinmode *faces-kbd-pin* :input)
  (digitalwrite *faces-kbd-pin* :high))

(defun kbd-pressed-p ()
  (null (digitalread *faces-kbd-pin*)))

(defun raw-read-kbd ()
  (with-i2c (str *FACES-KBD-ADDR* 1)
    (read-byte str)))

#| ... |#
```

ULISP DEMO

I2C + M5STACK FACES KEYBOARD #1

```
7530> (kbd-pressed-p)
nil

7530> (kbd-pressed-p)
t

7530> (raw-read-kbd)
104
```

ULISP DEMO

I2C + M5STACK FACES KEYBOARD #2

```
#| Max-Gerd Retzlaff <mgr@defsystem.net> - www.defsystem.net - unreleased |#
#| (c) Copyright Max-Gerd Retzlaff, 2021. All rights reserved. |#

#| ... |#

(defun decode-char (code)
  (if (< code 128)
      (code-char code)
      code))

(defun read-kbd ()
  (when (kbd-pressed-p)
    (let ((raw (raw-read-kbd)))
      (when raw ;; might be nil if i2c unsuccessful
                ;; as the kbd got disconnected
        (decode-char raw)))))
```

ULISP DEMO

I2C + M5STACK FACES KEYBOARD #2

```
7530> (read-kbd)
nil

7530> (read-kbd)
#\h
```

M5STACK STAND-ALONE ULISP COMPUTER MAX-GERD RETZLAFF, MARCH 2021

Batteries are included, which makes it self-contained and take-along.

[Retzlaff 2021e]

GPS-BASED SPEEDOMETER AND CLOCK

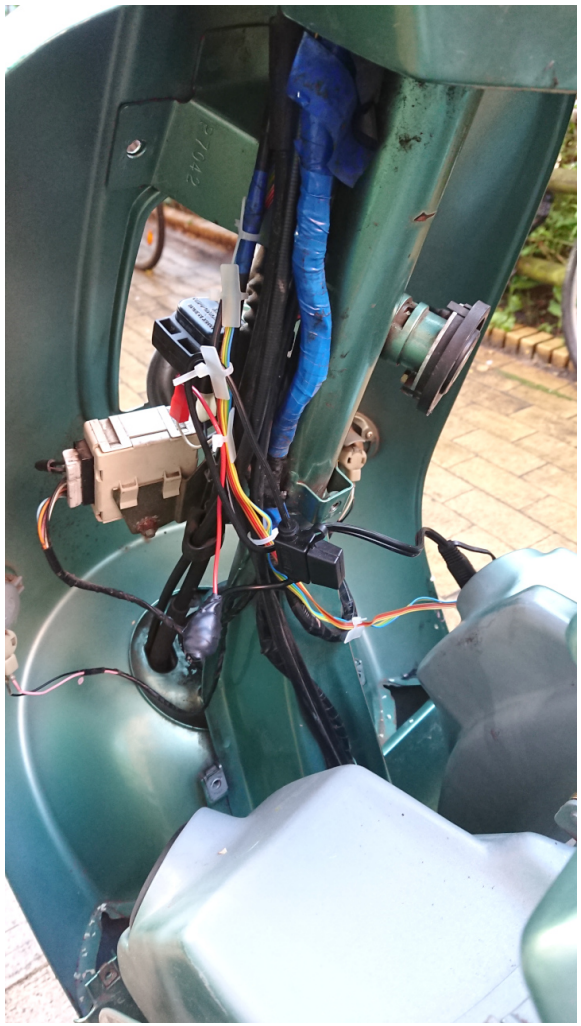
MAX-GERD RETZLAFF, MAY 2021

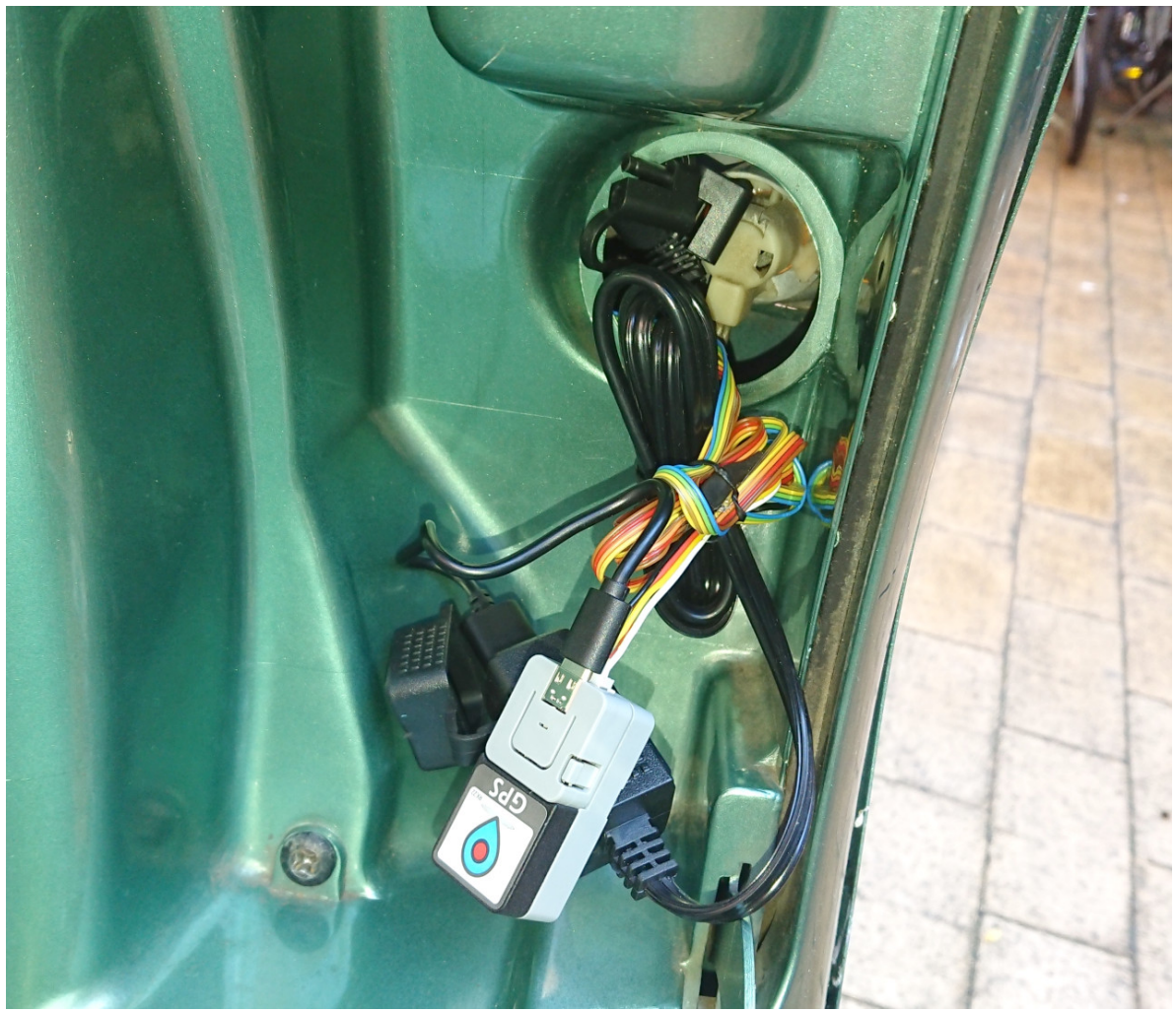
A replacement for the speedometer clock of my Vespa ET4
which often fails as it just uses a tiny battery.

[Retzlaff 2021f]

GPS-BASED SPEEDOMETER AND CLOCK

MAX-GERD RETZLAFF, MAY 2021

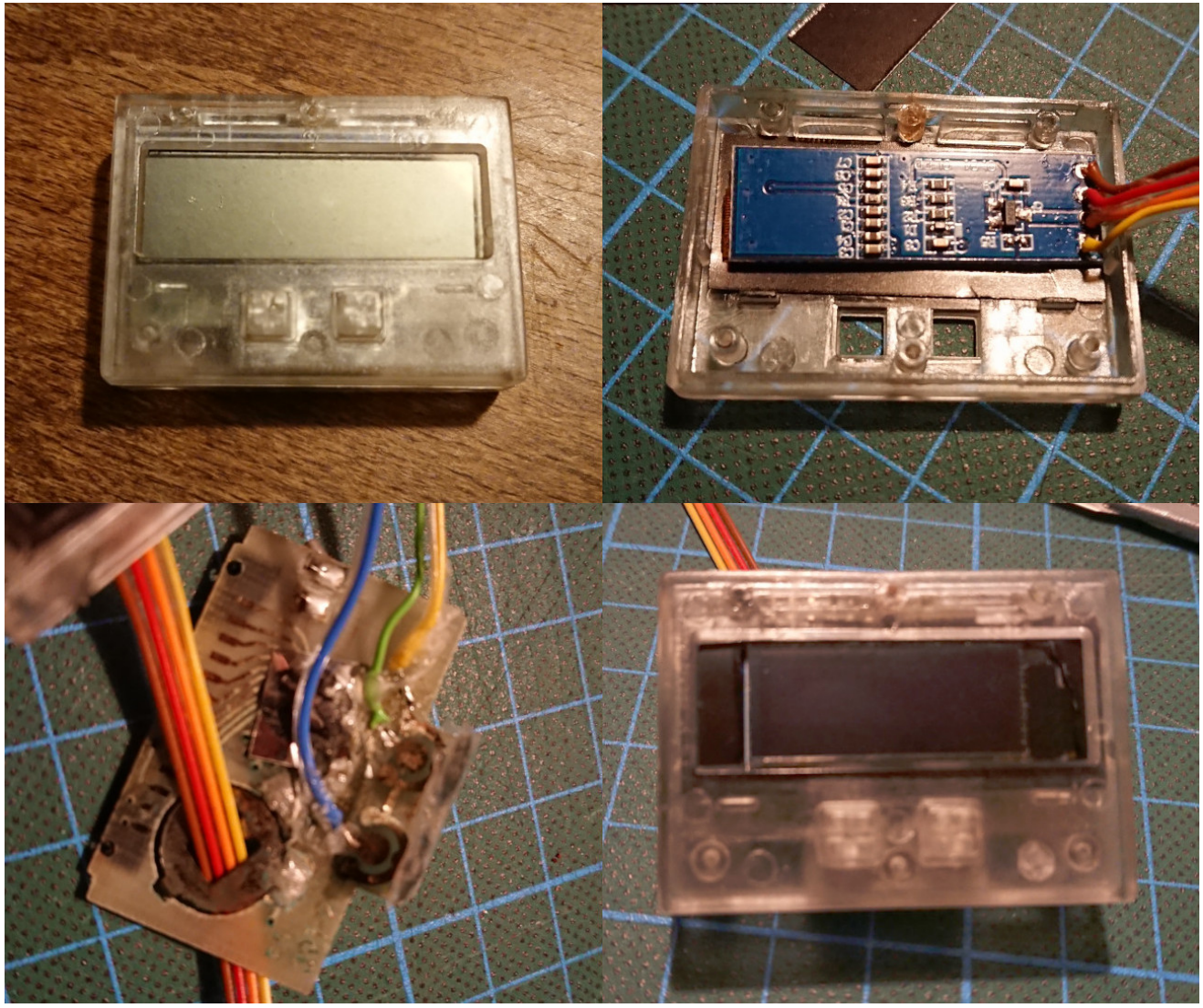




[Retzlaff 2021c]

GPS-BASED SPEEDOMETER AND CLOCK

MAX-GERD RETZLAFF, MAY 2021



GPS-BASED SPEEDOMETER AND CLOCK

MAX-GERD RETZLAFF, MAY 2021

[Retzlaff 2021g]

ULISP DEMO

WI-FI + WEB CLIENT

```
#| uLisp Wi-Fi examples by David Johnson-Davies |#
#| from http://www.ulisp.com/show?2B22 |#

(wifi-connect "Falafel" "fd2f12c66225")

(defun println (x s)
  (princ x s)
  (princ #\return s)
  (princ #\newline s))

(defun web-client ()
  (with-client (s "www.ulisp.com" 80)
    (println "GET /rss?1DQ5 HTTP/1.0" s)
    (println "Host: www.ulisp.com" s)
    (println "Connection: close" s)
    (println "" s)
    (loop (unless (zerop (available s)) (return)))
    (loop
     (when (zerop (available s)) (return))
     (princ (read-line s))
     (terpri))))
```

ULISP DEMO

WI-FI + WEB CLIENT

```
6306> (wifi-connect "Falafel" "fd2f12c66225")
Error: 'wifi-connect' connection failed

6306> (wifi-connect "Falafel" "fd2f12c66225")
"192.168.43.153"
```

ULISP DEMO

WI-FI + WEB CLIENT

```
6196> (web-client)
HTTP/1.1 200 OK
Date: Mon, 21 Mar 2022 13:26:08 GMT
Server: CL-HTTP/70.217 (LispWorks; 2.1.7)
Content-Type: text/xml; character-set=utf-8

<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
<title>uLisp</title>
<link>http://www.ulisp.com/show?3J</link>
<description><![CDATA[]]></description>
<pubDate>Thu, 30 Dec 2021 00:00:00 GMT</pubDate>
<lastBuildDate>Thu, 30 Dec 2021 00:00:00 GMT</lastBuildDate>
<language>en</language>
<image>
<url>http://www.ulisp.com/pictures/3J/icon-72.png</url>
<title>uLisp</title>
<link>http://www.ulisp.com/show?3J</link>
...
nil
```

ULISP DEMO

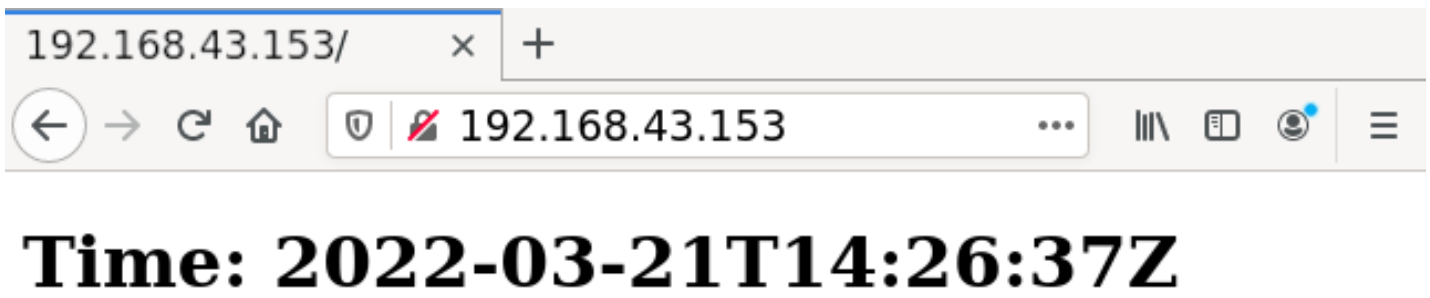
WI-FI + WEB SERVER

```
#| uLisp Wi-Fi examples by David Johnson-Davies |#
#| from http://www.ulisp.com/show?2B22 |#

(wifi-server)

(defun web-server ()
  (init-ntp)
  (loop
    (with-client (s)
      (loop
       (let ((line (read-line s)))
         (when (null line) (return))
         (print line)))
      (println "HTTP/1.1 200 OK" s)
      (println "Content-Type: text/html" s)
      (println "Connection: close" s)
      (println "" s)
      (princ "<!DOCTYPE HTML><html><body><h1>Time: " s)
      (princ (get-time) s)
      (println "</h1></body></html>" s)
      (println "" s))
    (delay 100)))
```

ULISP DEMO

WI-FI + WEB SERVER

```
6085> (web-server)

"GET / HTTP/1.1
"Host: 192.168.43.153
"User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:88.0) Gecko/201001"  Firefox/88.0
"Accept: text/html,application/xhtml+xml,application/xml;q=0.9,ima" /webp,*/*;q=0
"
"Accept-Encoding: gzip, deflate
"Connection: keep-alive
"Upgrade-Insecure-Requests: 1
```



ULISP DEMO

HTTP(S) GET/POST/PUT FUNCTION

```
object *fn_http (object *args, object *env) {
  /* Function http
   *
   * Syntax:
   *    http url &key verbose
   *                  (https t)
   *                  auth
   *                  (user default_username)
   *                  (password default_password)
   *                  accept
   *                  content-type
   *                  (method :get)
   *                  data
   *      => result-string
   *
   * Arguments and values:
   *    verbose---t, or nil (the default); affects also debug output of the
   * argument decoding itself and should be put in first position in a call
   * for full effect.
   *
```

ULISP DEMO HTTP(S) GET/POST/PUT FUNCTION

```
*    https---t (the default), nil, or a certificate as string; uses default
* certificate in C string root_ca if true; url needs to fit: "http://..."
* for true and and "https://..." for false.
*
*    auth---t, or nil (the default).
*
*    user---a string, or nil (the default); uses default value in C string
* default_username if nil; only used if :auth t.
*
*    password---a string, or nil (the default); uses default value in C
* string default_password if nil; only used if :auth t.
*
*    accept---nil (the default), or a string.
*
*    content-type---nil (the default), or a string.
*
*    method---:get (the default), :put, or :post.
*
*    data---nil (the default), or a string; only necessary in case of
* :method :put or :method :post; error for :method :get.
```

ULISP DEMO

HTTP(S) GET/POST/PUT FUNCTION

```
   * Examples:
   *    ;; HTTP GET:
   *    (http "http://192.168.179.41:2342" :https nil)
   *
   *    ;; HTTP PUT:
   *    (http "http://192.168.179.41:2342"
   *          :https nil
   *          :accept "application/n-quads"
   *          :content-type "application/n-quads"
   *          :auth t :user "foo" :password "bar"
   *          :method :put
   *          :data (format nil "<http://example.com/button>
   *                             <http://example.com/pressed>
   *                             \"~a\" .~%"
   *                            (get-time)))
   */
// ...
```

This code is included in my version of ulisp-esp-m5stack at:
https://github.com/m-g-r/ulisp-esp-m5stack/blob/master/ulisp-esp.ino

ULISP DEMO

PRESENTER DEVICE #1

```
#| Max-Gerd Retzlaff <mgr@defsystem.net> - www.defsystem.net - unreleased |#
#| (c) Copyright Max-Gerd Retzlaff, 2022. All rights reserved. |#

(defun send-next ()
  (ignore-errors
   (http "http://192.168.43.67:2342/present/next"
         :https nil)))

#| ... |#


6085> (http "http://192.168.43.67:2342/present/next" :https nil)
HTTP status: 200
""

6085> (http "http://192.168.43.67:2342/present/next" :https nil)
HTTP status: 200
""
```

ULISP DEMO

PRESENTER DEVICE #2

```
#| Max-Gerd Retzlaff <mgr@defsystem.net> - www.defsystem.net - unreleased |#
#| (c) Copyright Max-Gerd Retzlaff, 2022. All rights reserved. |#

#| ... |#

(defun led-ack (&optional (color #x00ff00) (times 2) (wait 200) (off #x000000))
  "flash sequences of the LED to communicate acknowledgement"
  (dotimes (i times)
    (atom-led color)
    (delay wait)
    (atom-led off)
    (delay wait)))

#| ... |#
```

ULISP DEMO

PRESENTER DEVICE #3

```
#| Max-Gerd Retzlaff <mgr@defsystem.net> - www.defsystem.net - unreleased |#
#| (c) Copyright Max-Gerd Retzlaff, 2022. All rights reserved. |#

(defun present-loop (&optional (wait 300) test)
  (init-atom-button)
  (led-ack #xff00ff 3 500 #x0000ff)
  (atom-led #x000000)
  (if (or test (not (atom-button-pressed-p)))
      (progn
        (led-ack)
        (loop
          (when (atom-button-pressed-p)
            (format t "~&send next")
            (atom-led #xffff00) (delay 100)
            (send-next)
            (atom-led #x00ff00) (delay 100)
            (atom-led #x000000))
          (format t "~&sleep ~a ms" wait)
          (delay wait)))
      (progn #| aborted |#
        (led-ack #xff0000))))
```

ULISP DEMO

PRESENTER DEVICE #4

```
#| Max-Gerd Retzlaff <mgr@defsystem.net> - www.defsystem.net - unreleased |#
#| (c) Copyright Max-Gerd Retzlaff, 2022. All rights reserved. |#

#!/bin/bash
while true; do
  echo -e "HTTP/1.1 200 OK\n" | nc -l -p 2342 -N | grep next
  if [ $? -eq 0 ]; then
    echo next
    xdotool key Page_Down
  else
    echo unknown
  fi
done
```

PRESENTER FOR "NEXT PAGE"'

MAX-GERD RETZLAFF, MARCH 2022

On the plane trip to ELS 2022 … :-)

[Retzlaff 2022b]

ULISP WITH ANDROID

MARCH 2022

[Retzlaff 2022d]

ULISP DEMO

GRAPH STORE HTTP PROTOCOL – PUT

```
#| Max-Gerd Retzlaff <mgr@defsystem.net> - www.defsystem.net - unreleased |#
#| (c) Copyright Max-Gerd Retzlaff, 2021. All rights reserved. |#

(defun graph-store-put (&optional (time (get-time)))
  (http "https://de8.dydra.com/max/test/service"
        :accept "application/n-quads"
        :content-type "application/n-quads"
        :auth t
        :method :put
        :data (unparse-ntriples
               (list
                (list "<http://example.com/button>"
                      "<http://example.com/pressed>"
                      (cons time
                            "<http://www.w3.org/2001/XMLSchema#dateTime>"))))))
```

ULISP DEMO

GRAPH STORE HTTP PROTOCOL – PUT

```
5994> (graph-store-put)
HTTP status: 200
"<urn:uuid:D8629134-A926-11EC-8FCA-A3BFF8D080F9> <http://www.w3.org/ns/acti↩
↪vitystreams#id> <urn:uuid:D8629134-A926-11EC-8FCA-A3BFF8D080F9> .
<urn:uuid:D8629134-A926-11EC-8FCA-A3BFF8D080F9> <http://www.w3.org/1999/02/↩
↪22-rdf-syntax-ns#type> <http://www.w3.org/ns/activitystreams#Update> .
<urn:uuid:D8629134-A926-11EC-8FCA-A3BFF8D080F9> <http://www.w3.org/1999/02/↩
↪22-rdf-syntax-ns#type> <http://www.w3.org/2011/http-methods#PUT> .
<urn:uuid:D8629134-A926-11EC-8FCA-A3BFF8D080F9> <http://www.w3.org/ns/prov#↩
↪startedAtTime> \"2022-03-21T14:54:44Z\"^^<http://www.w3.org/2001/XMLSchem↩
↪a#dateTime> .
<urn:uuid:D8629134-A926-11EC-8FCA-A3BFF8D080F9> <http://www.w3.org/ns/prov#↩
↪endedAtTime> \"2022-03-21T14:54:44.761398Z\"^^<http://www.w3.org/2001/XML↩
↪Schema#dateTime> .
<urn:uuid:D8629134-A926-11EC-8FCA-A3BFF8D080F9> <http://www.w3.org/ns/activ↩
↪itystreams#actor> <http://dydra.com/users/max> .
<urn:uuid:D8629134-A926-11EC-8FCA-A3BFF8D080F9> <http://www.w3.org/ns/activ↩
↪itystreams#object> <http://dydra.com/max/test?revision=E7258D0E-B093-FC41↩
↪-8962-33E838E26726> .
"
```

ULISP DEMO

PARSE RDF 1.1 N-TRIPLES

```
5994> (parse-ntriples (graph-store-put))
HTTP status: 200
(("<urn:uuid:E860071A-A926-11EC-8FCA-A3BFF8D080F9>" "<http://www.w3.org/ns/↩
↪activitystreams#id>" "<urn:uuid:E860071A-A926-11EC-8FCA-A3BFF8D080F9>") (↩
↪"<urn:uuid:E860071A-A926-11EC-8FCA-A3BFF8D080F9>" "<http://www.w3.org/199↩
↪9/02/22-rdf-syntax-ns#type>" "<http://www.w3.org/ns/activitystreams#Updat↩
↪e>") ("<urn:uuid:E860071A-A926-11EC-8FCA-A3BFF8D080F9>" "<http://www.w3.o↩
↪rg/1999/02/22-rdf-syntax-ns#type>" "<http://www.w3.org/2011/http-methods#↩
↪PUT>") ("<urn:uuid:E860071A-A926-11EC-8FCA-A3BFF8D080F9>" "<http://www.w3↩
↪.org/ns/prov#startedAtTime>" ("2022-03-21T14:55:11Z" . "<http://www.w3.or↩
↪g/2001/XMLSchema#dateTime>")) ("<urn:uuid:E860071A-A926-11EC-8FCA-A3BFF8D↩
↪080F9>" "<http://www.w3.org/ns/prov#endedAtTime>" ("2022-03-21T14:55:11.5↩
↪7763Z" . "<http://www.w3.org/2001/XMLSchema#dateTime>")) ("<urn:uuid:E860↩
↪071A-A926-11EC-8FCA-A3BFF8D080F9>" "<http://www.w3.org/ns/activitystreams↩
↪#actor>" "<http://dydra.com/users/max>") ("<urn:uuid:E860071A-A926-11EC-8↩
↪FCA-A3BFF8D080F9>" "<http://www.w3.org/ns/activitystreams#object>" "<http↩
↪://dydra.com/max/test?revision=EA707E5D-977D-CF4D-8782-C692E35EDEE6>"))
```

ULISP DEMO

PARSE RDF 1.1 N-TRIPLES

```
5994> (mapcar (lambda (x) (format t "~&~a" x)) (parse-ntriples (graph-store-put)))
HTTP status: 200
(<urn:uuid:F68B3BD4-A926-11EC-8FCA-A3BFF8D080F9>↩
↪<http://www.w3.org/ns/activitystreams#id>↩
↪<urn:uuid:F68B3BD4-A926-11EC-8FCA-A3BFF8D080F9>)
(<urn:uuid:F68B3BD4-A926-11EC-8FCA-A3BFF8D080F9>↩
↪<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>↩
↪<http://www.w3.org/ns/activitystreams#Update>)
(<urn:uuid:F68B3BD4-A926-11EC-8FCA-A3BFF8D080F9>↩
↪<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>↩
↪<http://www.w3.org/2011/http-methods#PUT>)
(<urn:uuid:F68B3BD4-A926-11EC-8FCA-A3BFF8D080F9>↩
↪<http://www.w3.org/ns/prov#startedAtTime>↩
↪(2022-03-21T14:55:35Z . <http://www.w3.org/2001/XMLSchema#dateTime>))
(<urn:uuid:F68B3BD4-A926-11EC-8FCA-A3BFF8D080F9>↩
↪<http://www.w3.org/ns/prov#endedAtTime>↩
↪(2022-03-21T14:55:35.35146Z . <http://www.w3.org/2001/XMLSchema#dateTime>))
(<urn:uuid:F68B3BD4-A926-11EC-8FCA-A3BFF8D080F9>↩
↪<http://www.w3.org/ns/activitystreams#actor>↩
↪<http://dydra.com/users/max>)
(<urn:uuid:F68B3BD4-A926-11EC-8FCA-A3BFF8D080F9>↩
↪<http://www.w3.org/ns/activitystreams#object>↩
↪<http://dydra.com/max/test?revision=D20792B5-AD3D-8445-9013-08F14C3A7123>)
(nil nil nil nil nil nil nil)
```

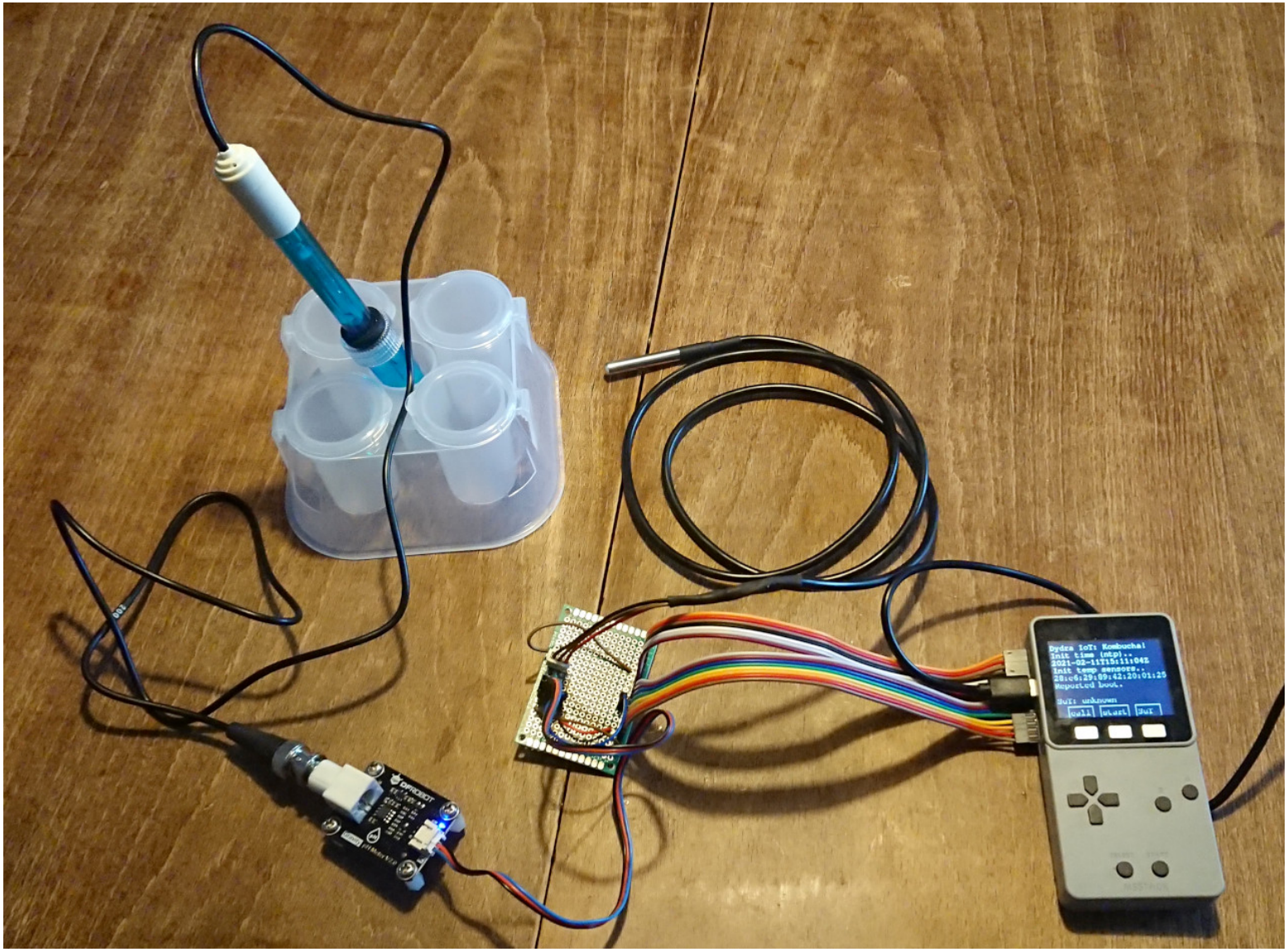
ULISP DEMO

GRAPH STORE HTTP PROTOCOL – GET

```
#| Max-Gerd Retzlaff <mgr@defsystem.net> - www.defsystem.net - unreleased |#
#| (c) Copyright Max-Gerd Retzlaff, 2022. All rights reserved. |#

(defun graph-store-get ()
  (http "https://de8.dydra.com/max/test/service"
        :accept "application/n-quads"
        :auth t))

#| ... |#


5964> (graph-store-get)
HTTP status: 200
"<http://example.com/button> <http://example.com/pressed> \"2022-03-21T15:↩
↪55:32Z\"^^<http://www.w3.org/2001/XMLSchema#dateTime> .
"

5964> (car (caddr (first (parse-ntriples (graph-store-get)))))
HTTP status: 200
"2022-03-21T15:55:32Z"
```

KOMBUCHA SENSOR DEVICE

MAX-GERD RETZLAFF, FEB. 2021



[Retzlaff 2021a]

KOMBUCHA SENSOR DEVICE

MAX-GERD RETZLAFF, FEB 2021

KOMBUCHA SENSOR DEVICE

MAX-GERD RETZLAFF, FEB 2021

[Retzlaff 2021k]

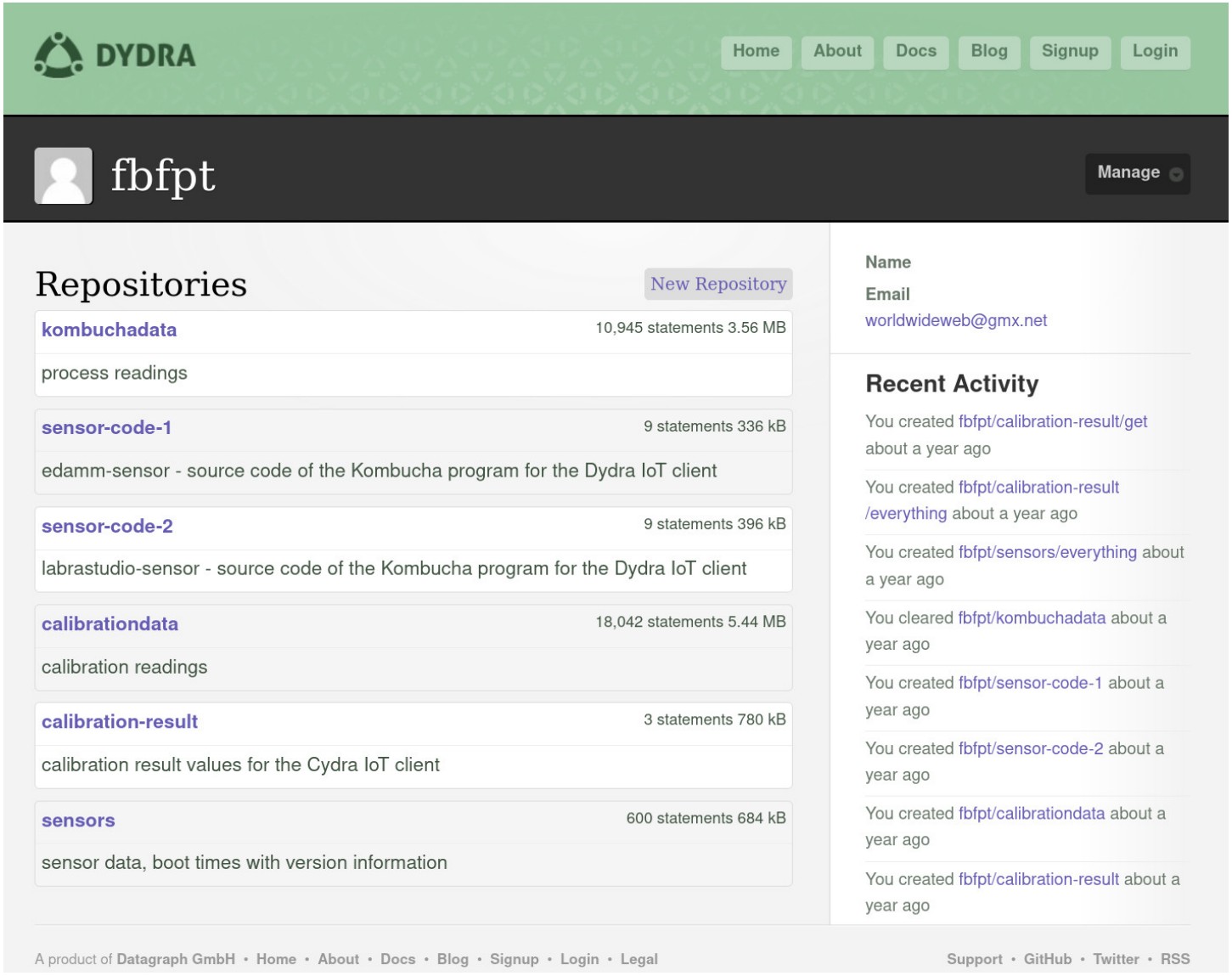
KOMBUCHA SENSOR DEVICE

MAX-GERD RETZLAFF, FEB 2021

[Retzlaff 2021k]

KOMBUCHA SENSOR DEVICE

MAX-GERD RETZLAFF, FEB. 2021



[Retzlaff 2021j]

KOMBUCHA SENSOR DEVICE

MAX-GERD RETZLAFF, FEB. 2021

datagenous · labra.studio

Share...

By m-g-r You Edited Mar 20 Fork of ph sensor calibration runs

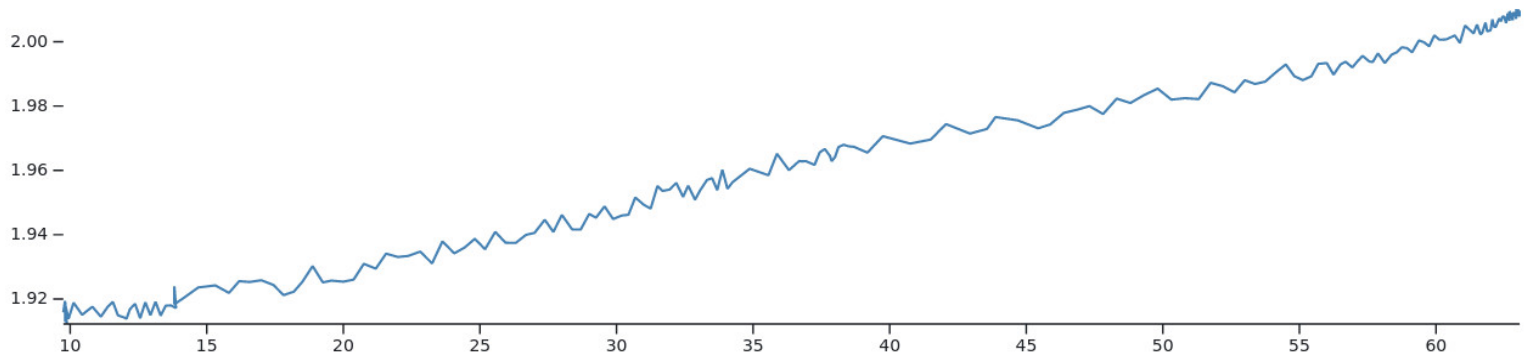
ph sensor calibration runs

All data is stored in: https://nl4.dydra.com/fbfpt/calibration See query:
https://nl4.dydra.com/fbfpt/calibration/@query#calibration_all_get

interactive visualization

```
showfile = ▶ FileAttachment {name: "ph4.01-red@1.csv"}
```

```
// Apera Instruments calibration solutions:
//showfile = ph7g
showfile = ph4r
//showfile = ph10b
//showfile = ph7g2
// dfrobot calibration solutions:
//showfile = ph4
//showfile = ph7
```



[Retzlaff 2021i]

ULISP DEMO

WI-FI PROVISIONING VIA BLUETOOTH BLE

Our uLisp systems on the M5Stack Atom Lite ESP32 run a version of m5stack-ulisp-esp that has Wi-Fi provisioning activated.

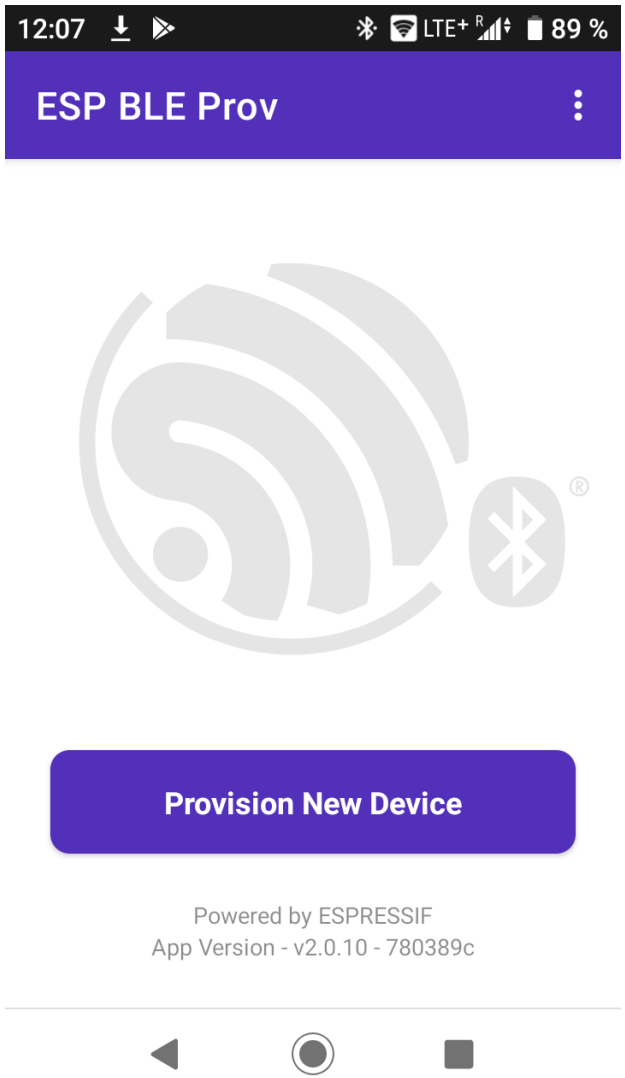
We have implemented it to use Espressif's Unified Provisioning using Bluetooth Low Energy (BLE) as the transport.

Unified Provisioning is an extensible mechanism to transfer Wi-Fi credentials and also other custom configuration.

The provisioning is initiated from another device, a smartphone or a computer.

More on: `http://www.defsystem.net/Wi-Fi-provisioning-via-Bluetooth-BLE.xml`

ULISP DEMO



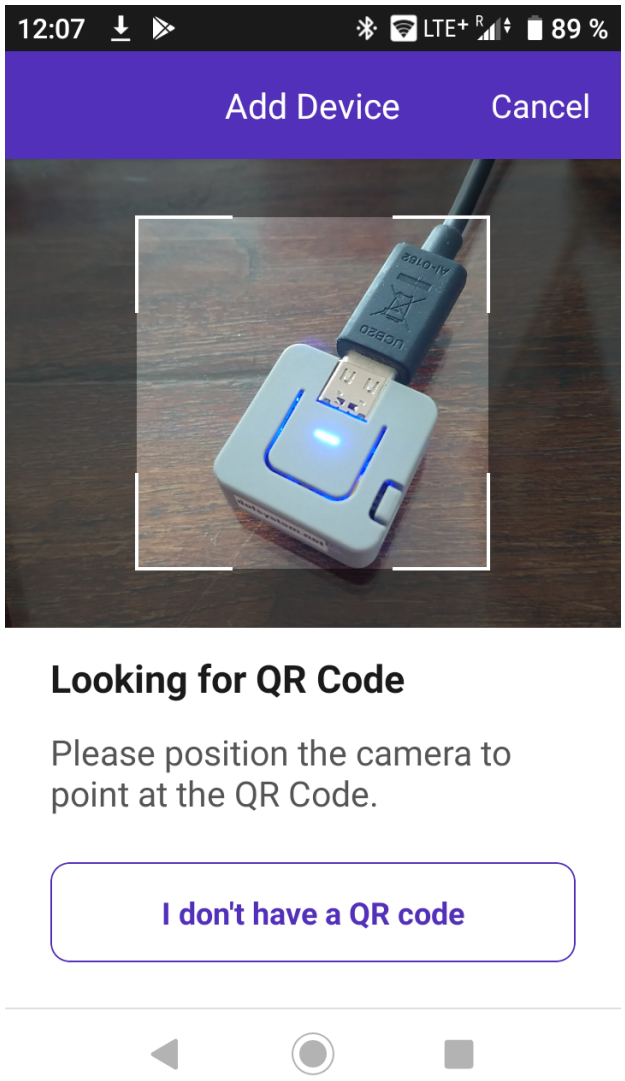
ESP BLE PROVISIONING – STEP BY STEP

1.

This is the initial screen of the provisioning app. Just click on the button “Provision New Device”.

[Retzlaff 2022a]

ULISP DEMO



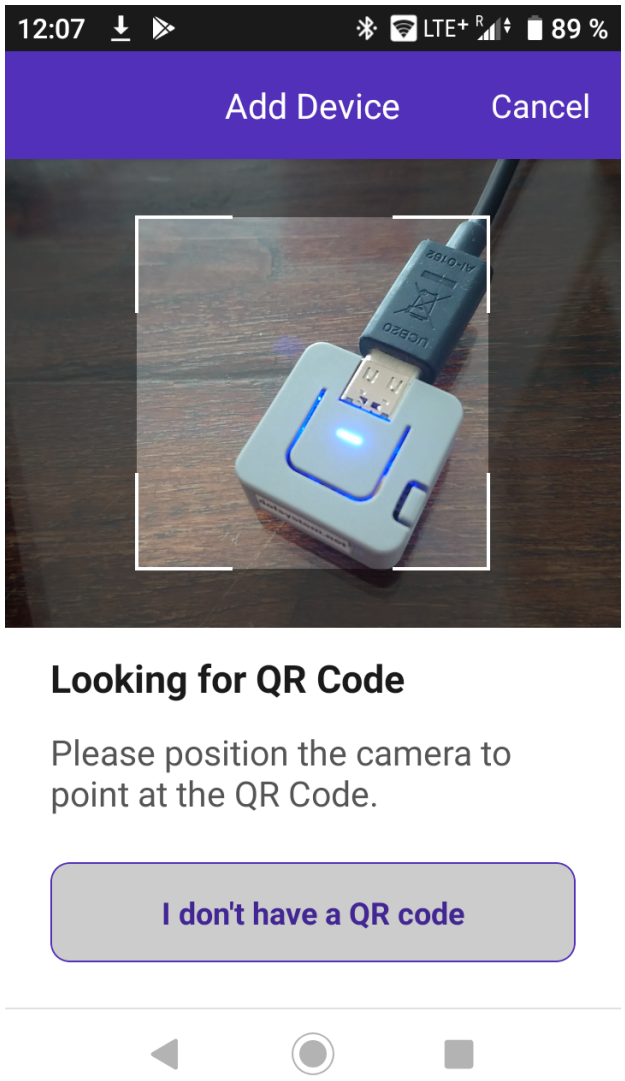
ESP BLE PROVISIONING – STEP BY STEP

2.

QR codes can be used to make provisioning easier . . .

[Retzlaff 2022a]

ULISP DEMO



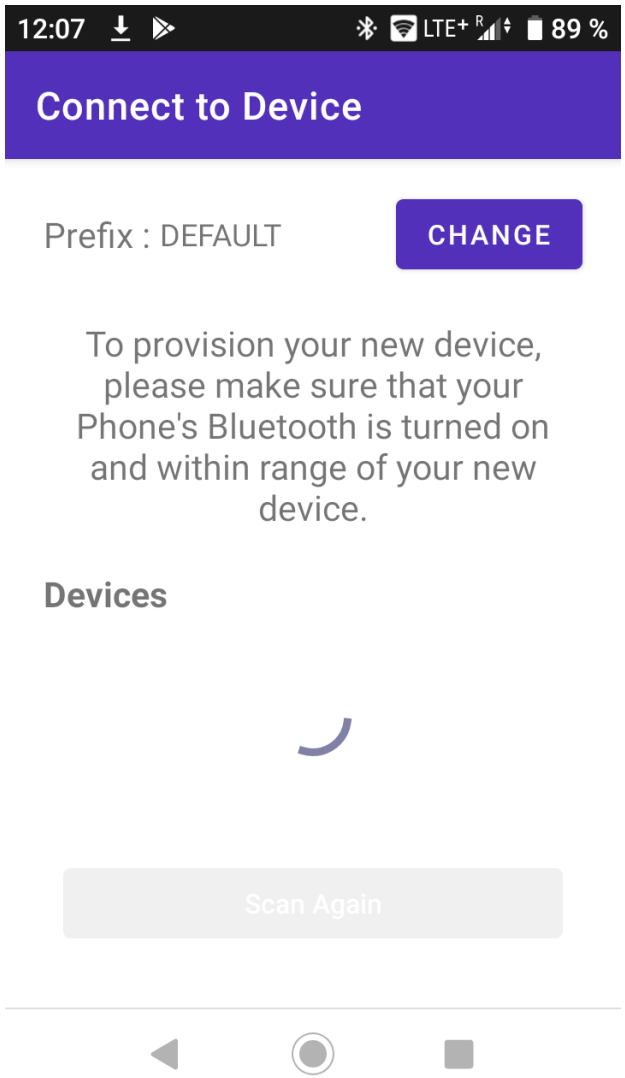
ESP BLE PROVISIONING – STEP BY STEP

3.

. . . but as there is no QR code on the defsystem device, just click on the button “I don’t have a QR code”.

[Retzlaff 2022a]

ULISP DEMO



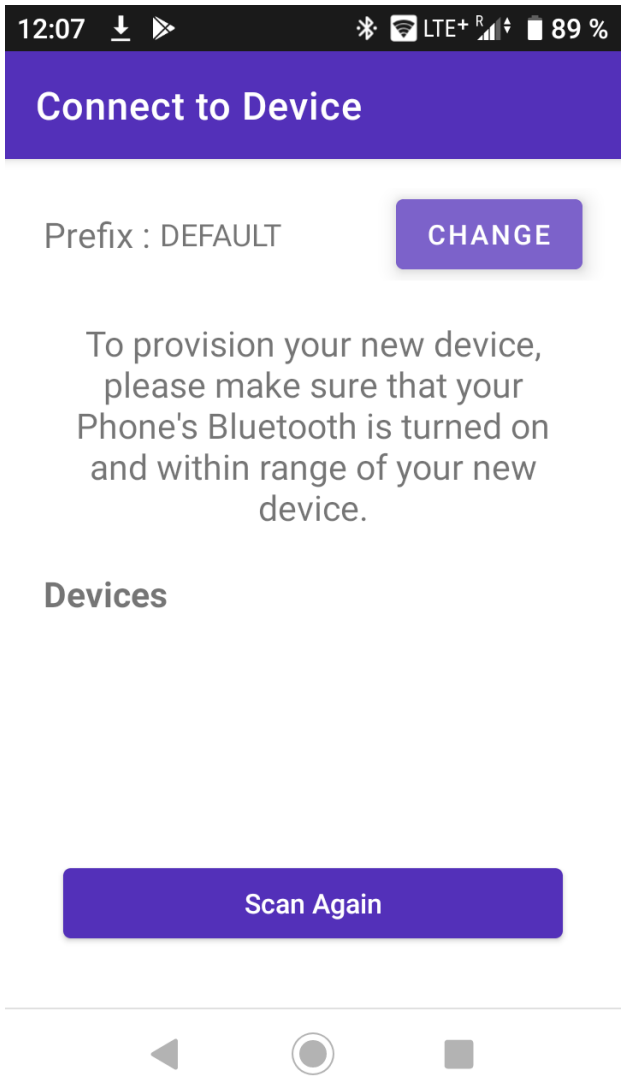
ESP BLE PROVISIONING – STEP BY STEP

4.

The app looks for devices to provision . . .

[Retzlaff 2022a]

ULISP DEMO



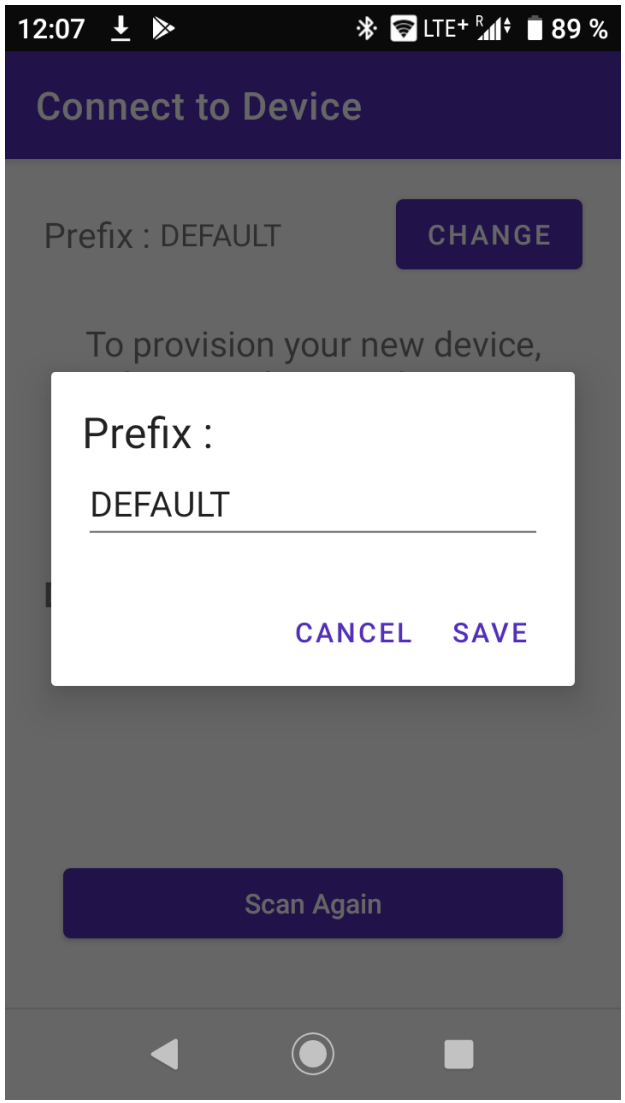
ESP BLE PROVISIONING – STEP BY STEP

5.

. . . but as the search prefix is set to “DEFAULT”, the app does not find any devices. So click on the button “CHANGE”.

[Retzlaff 2022a]

ULISP DEMO



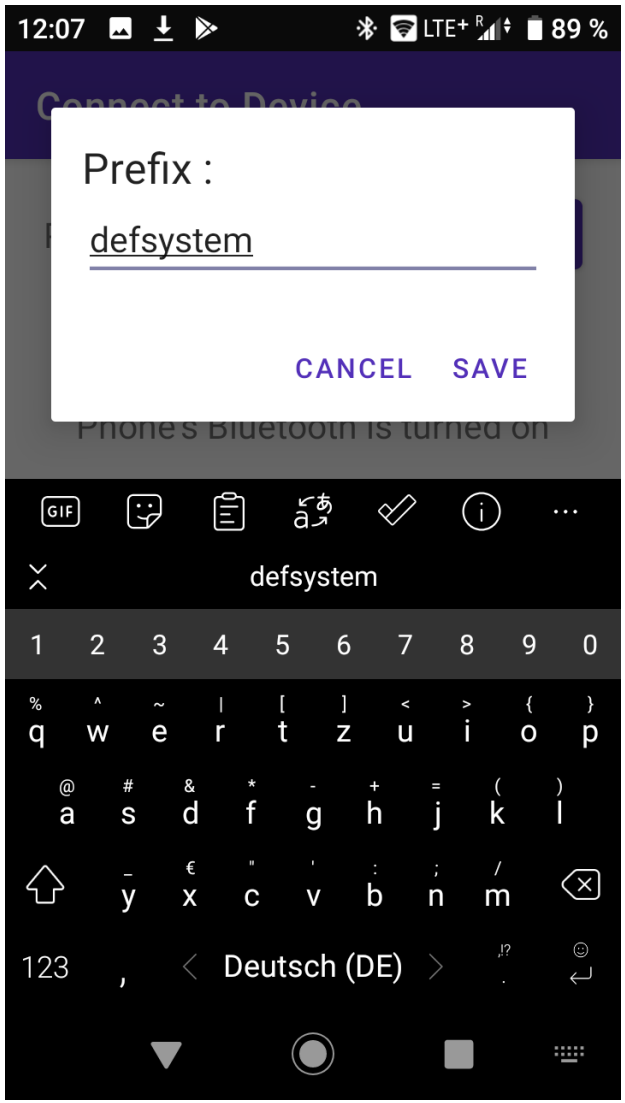
ESP BLE PROVISIONING – STEP BY STEP

6.

Select the input field that shows “DEFAULT” . . .

[Retzlaff 2022a]

ULISP DEMO

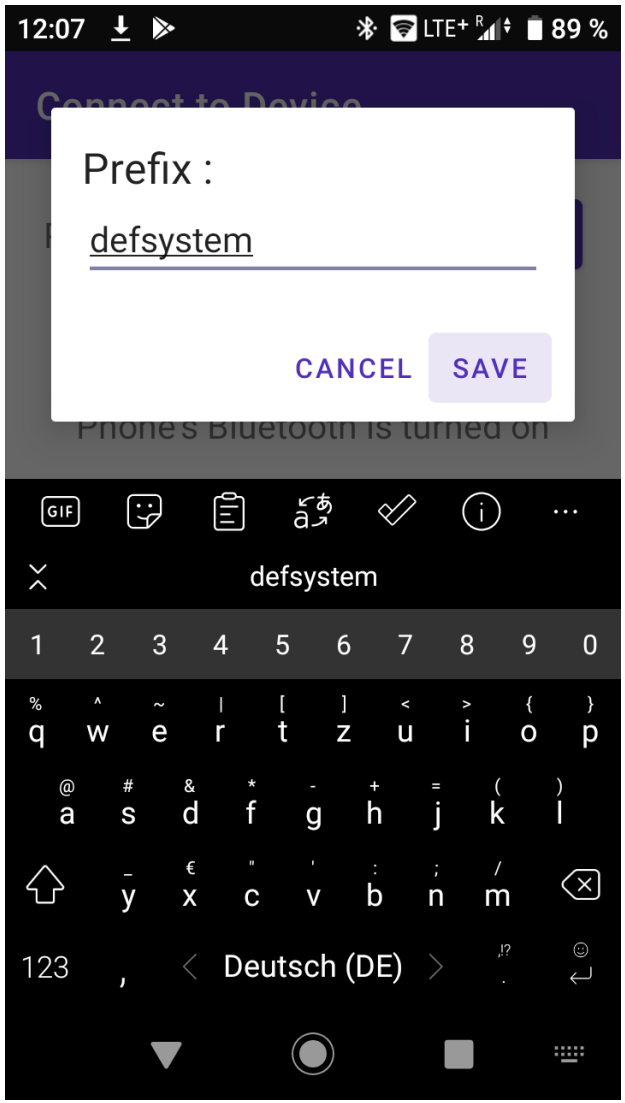


ESP BLE PROVISIONING – STEP BY STEP

7.

. . . and type in “defsystem” instead . . .

ULISP DEMO

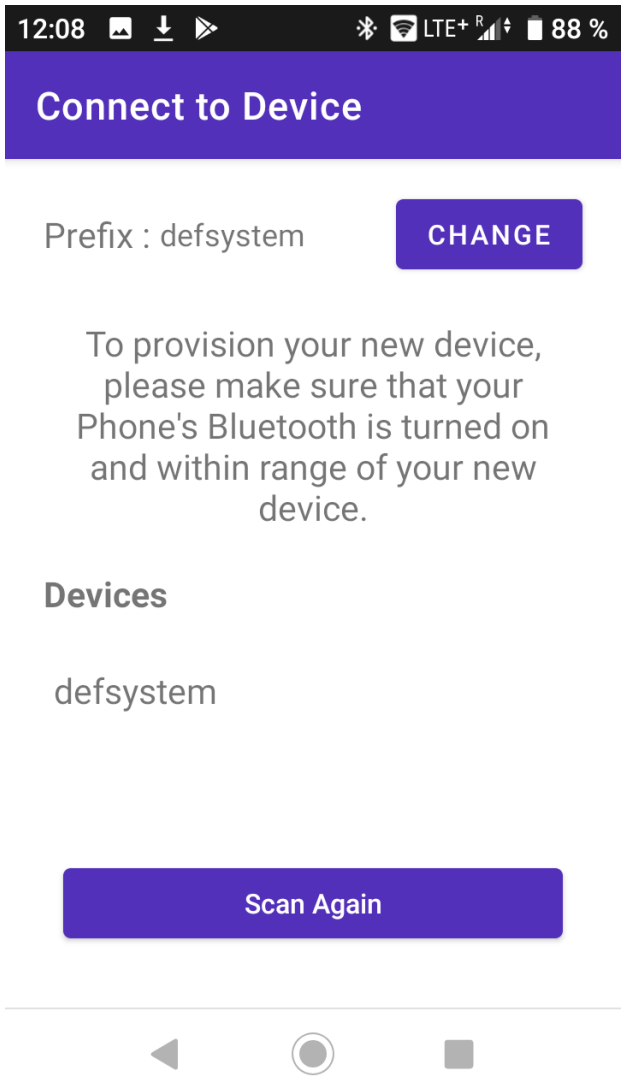


ESP BLE PROVISIONING – STEP BY STEP

8.

. . . and click on the button “SAVE”.

ULISP DEMO

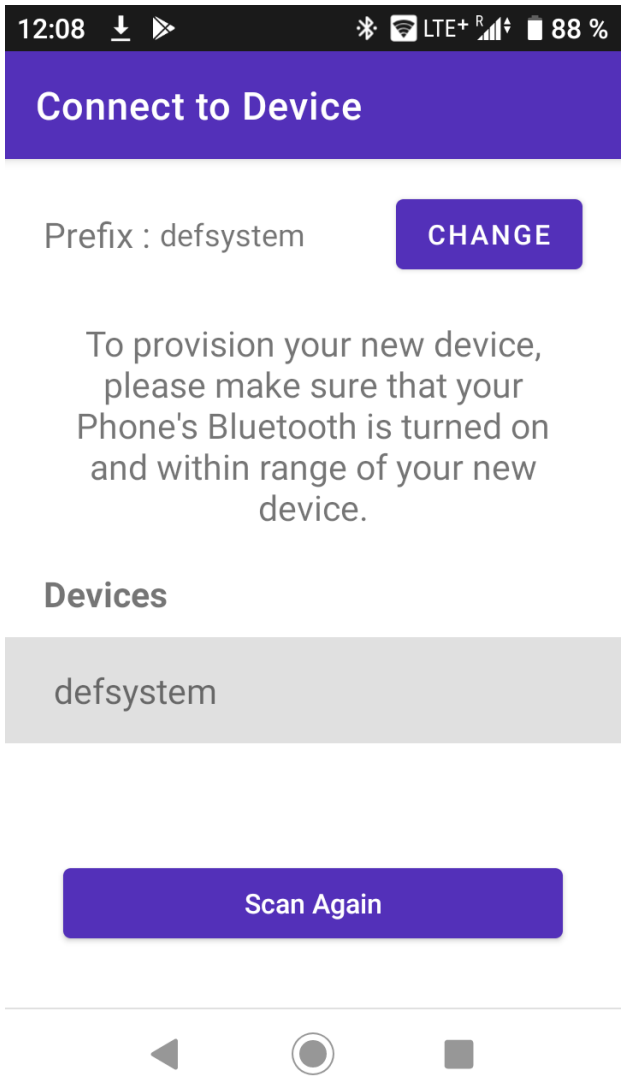


ESP BLE PROVISIONING – STEP BY STEP

9.

The application should now list your device with name “defsystem” or “defsystem-xx” where “xx” is the two digit number that you should find on the bottom side of your device.

ULISP DEMO



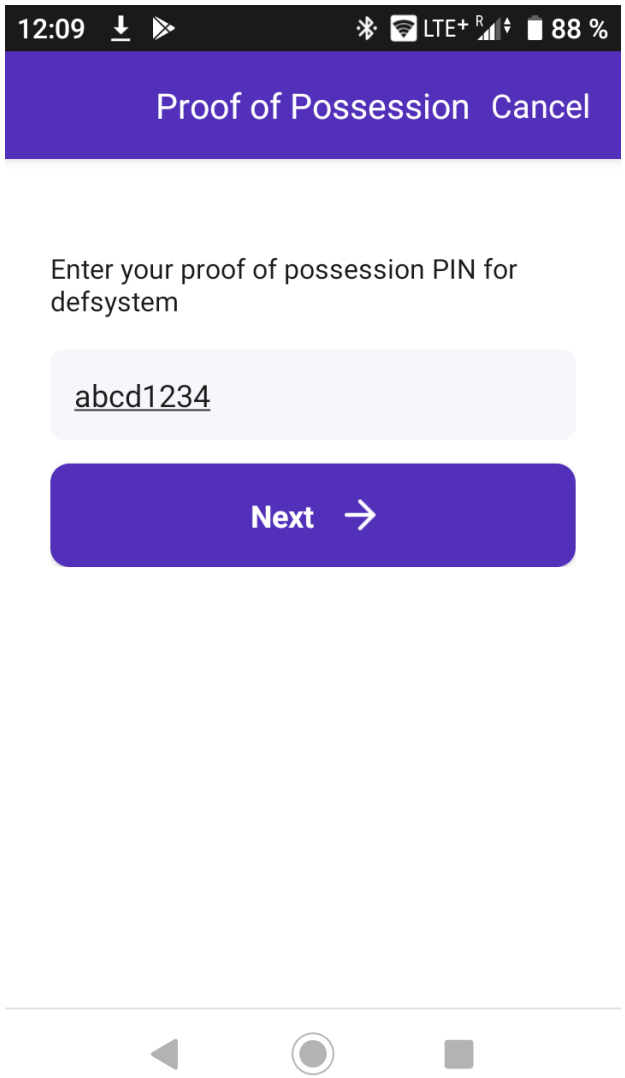
ESP BLE PROVISIONING – STEP BY STEP

10.

Select your device by clicking on its name.

[Retzlaff 2022a]

ULISP DEMO

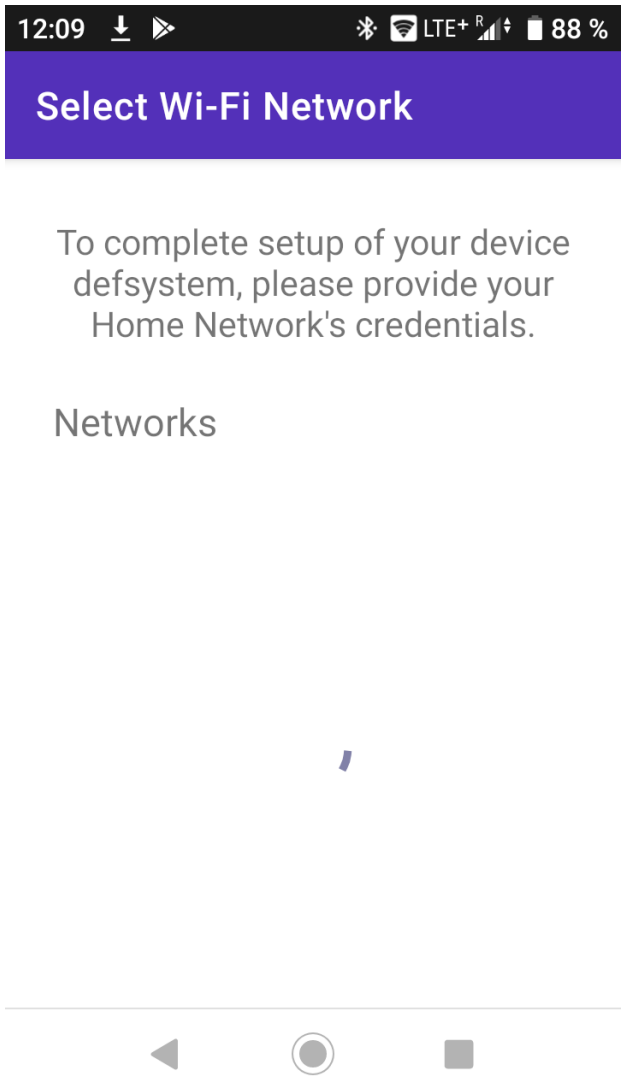


ESP BLE PROVISIONING – STEP BY STEP

11.

The app asks for a “PIN” code as “Proof of Possession”. As we have used the application’s default code “abcd1234” you can leave this as it is and just click on the button “Next →”.

ULISP DEMO



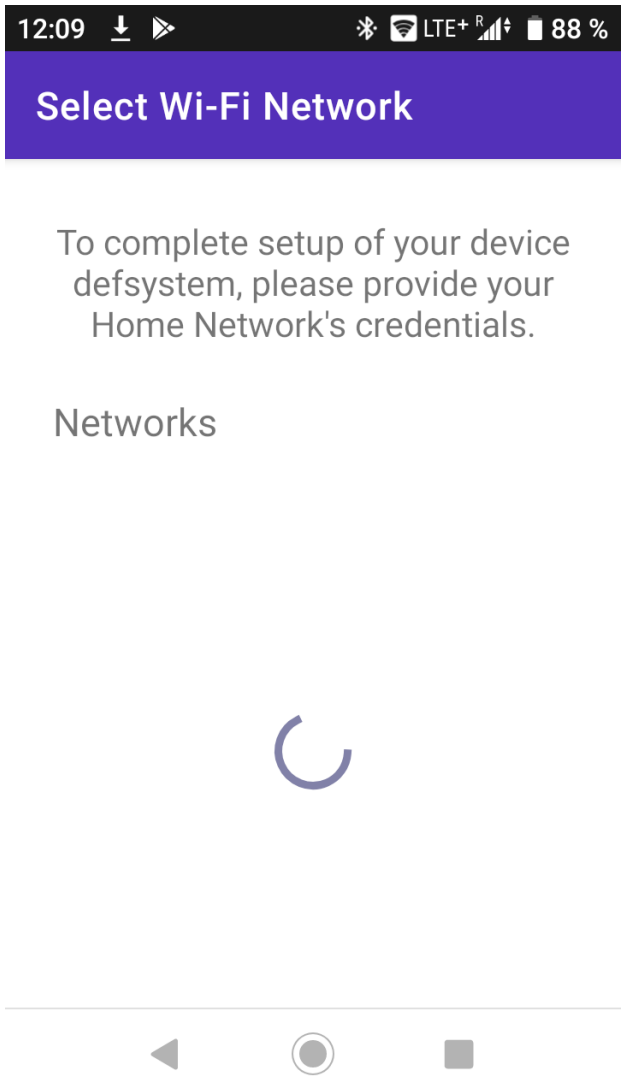
ESP BLE PROVISIONING – STEP BY STEP

12.

The app should now talk to your device via Bluetooth BLE and make it search for Wi-Fi networks.

[Retzlaff 2022a]

ULISP DEMO



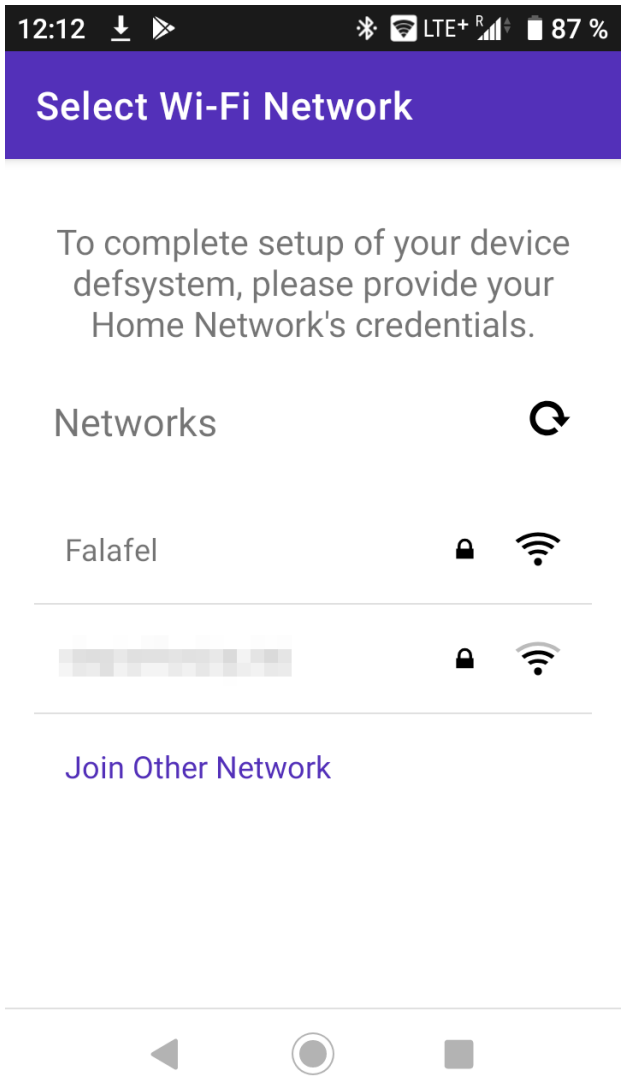
ESP BLE PROVISIONING – STEP BY STEP

13.

This may take a moment while the app shows a waiting animation.

[Retzlaff 2022a]

ULISP DEMO



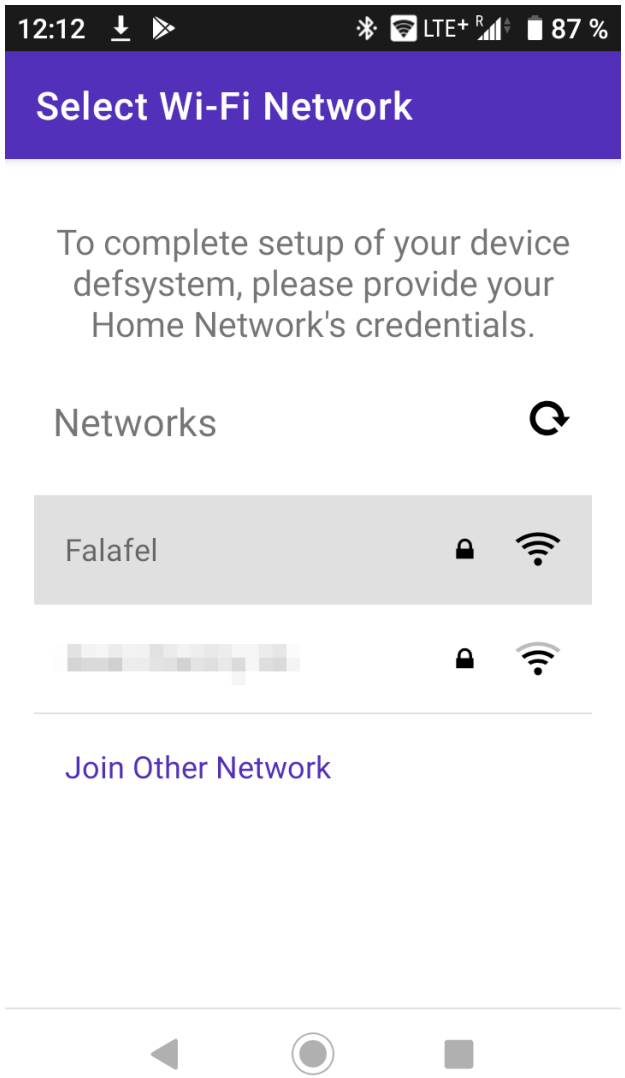
ESP BLE PROVISIONING – STEP BY STEP

14.

Hopefully after a short wait the app presents a list of networks that the defsystem device has discovered.

[Retzlaff 2022a]

ULISP DEMO



ESP BLE PROVISIONING – STEP BY STEP

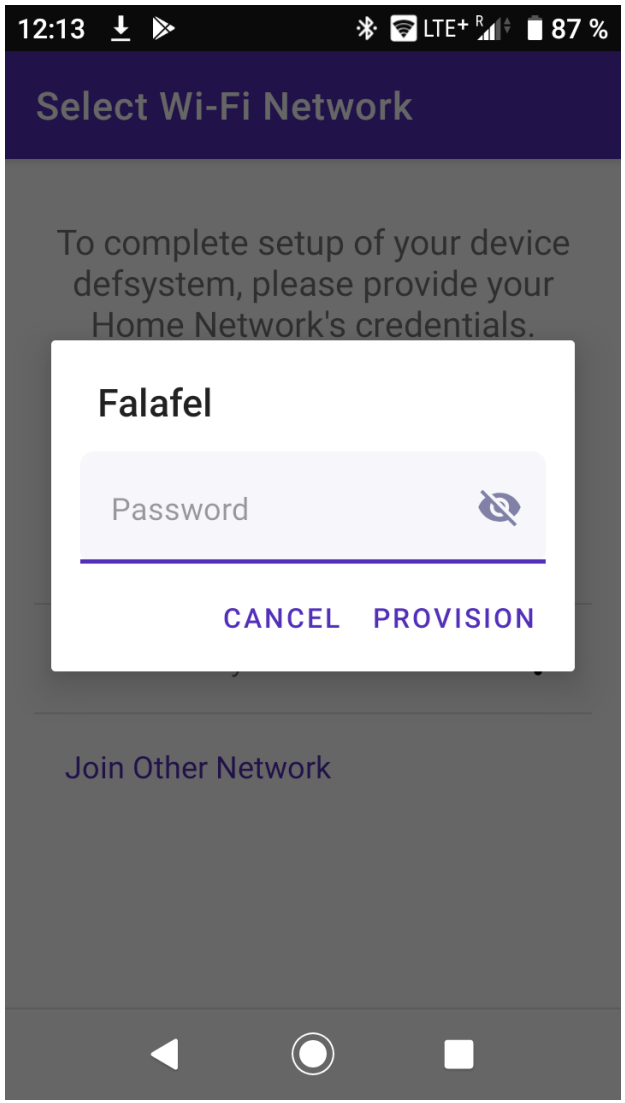
15.

In this demonstration we select “Falafel” as our example network.

[Retzlaff 2022a]

ULISP DEMO

ESP BLE PROVISIONING – STEP BY STEP

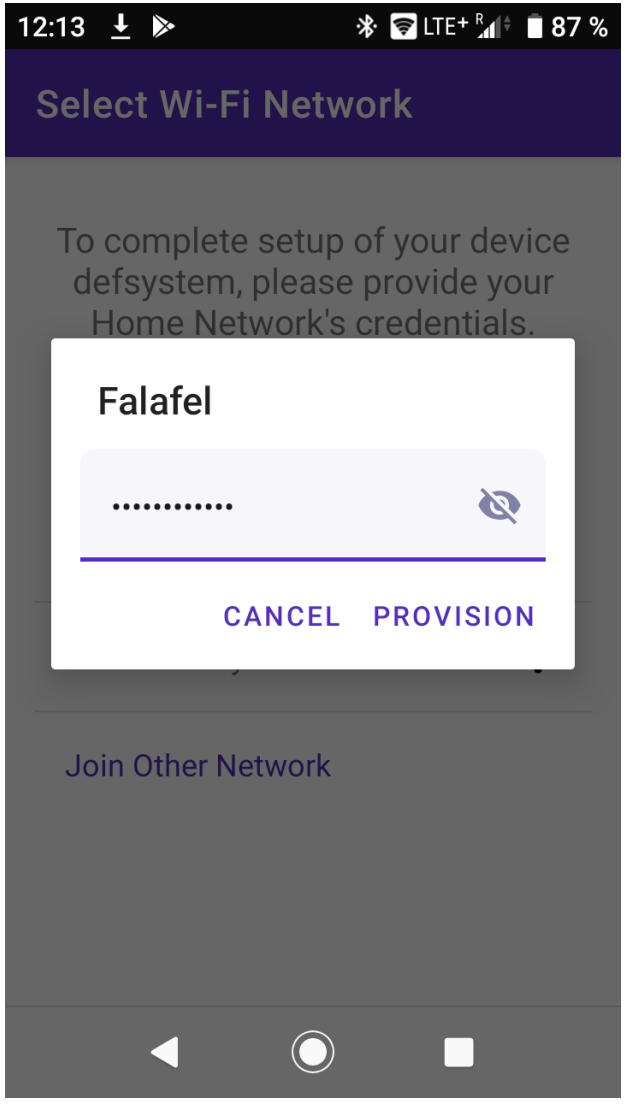


16.

Enter the password for the network . . .

[Retzlaff 2022a]

ULISP DEMO

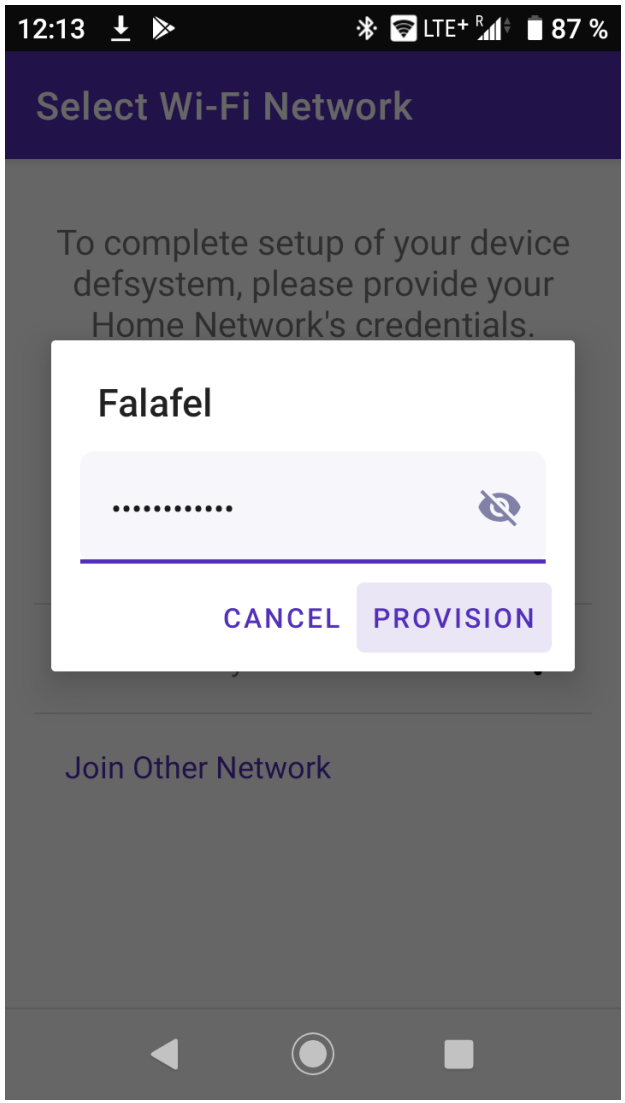


ESP BLE PROVISIONING – STEP BY STEP

. . .

[Retzlaff 2022a]

ULISP DEMO



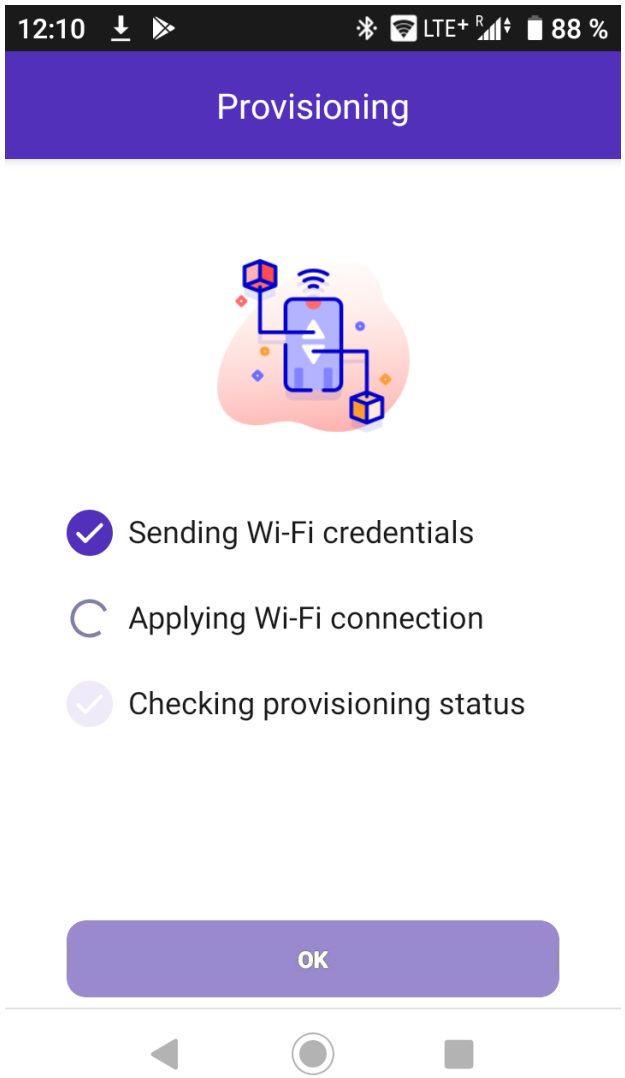
ESP BLE PROVISIONING – STEP BY STEP

17.

. . . and click on the button “PROVISION”.

[Retzlaff 2022a]

ULISP DEMO

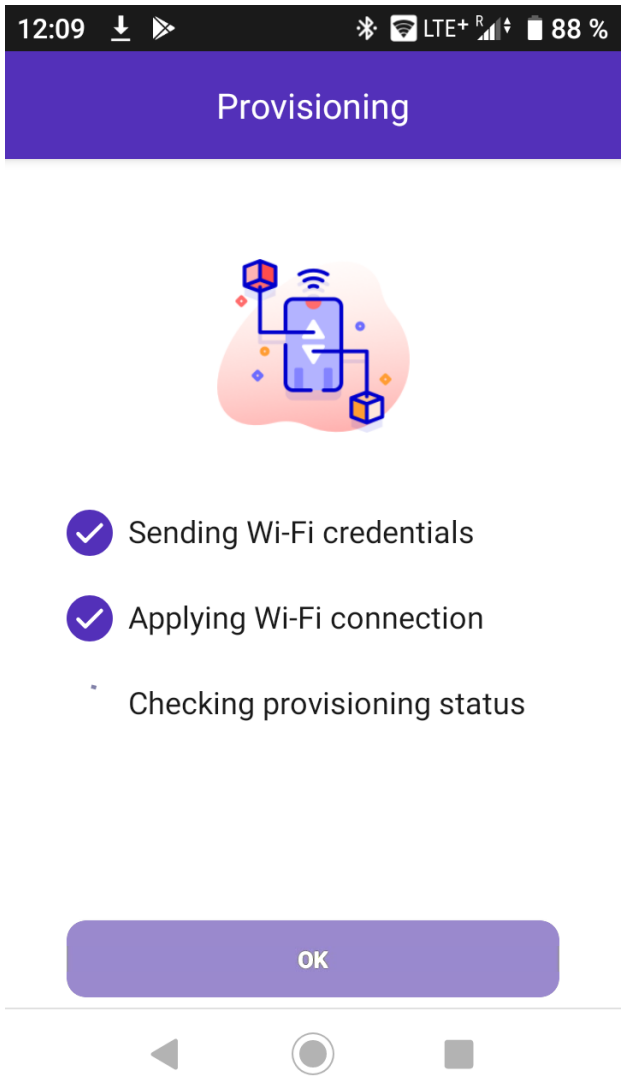


ESP BLE PROVISIONING – STEP BY STEP

18.

The actual provisioning starts. As a first step the app sends the Wi-Fi credentials . . .

ULISP DEMO



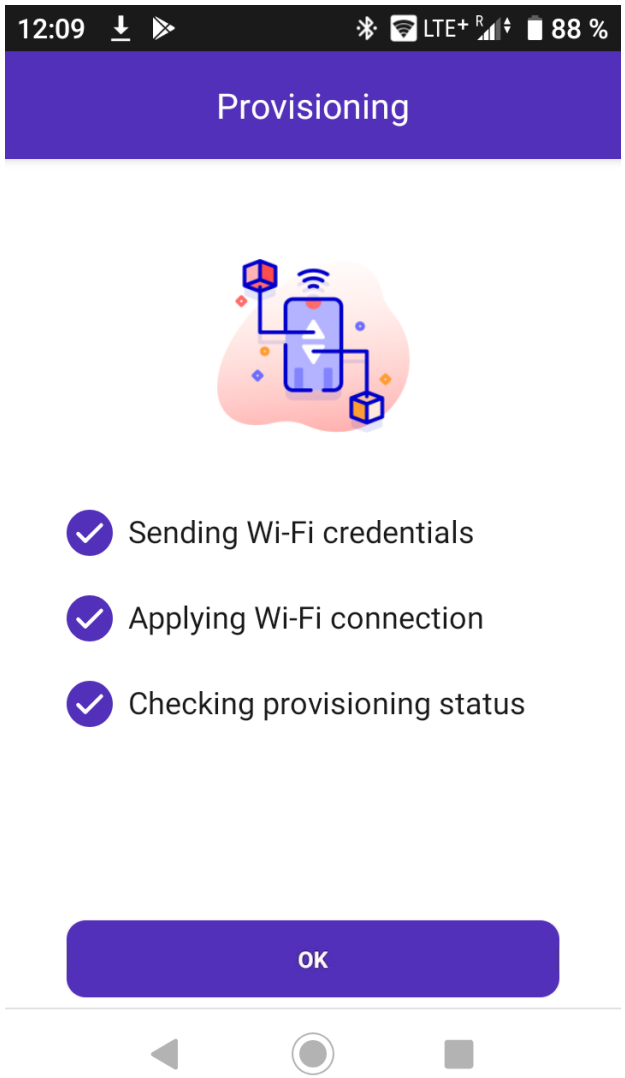
ESP BLE PROVISIONING – STEP BY STEP

19.

. . . the second is to wait for the device to actually connect to that network and get a Wi-Fi connection.

[Retzlaff 2022a]

ULISP DEMO



ESP BLE PROVISIONING – STEP BY STEP

20.

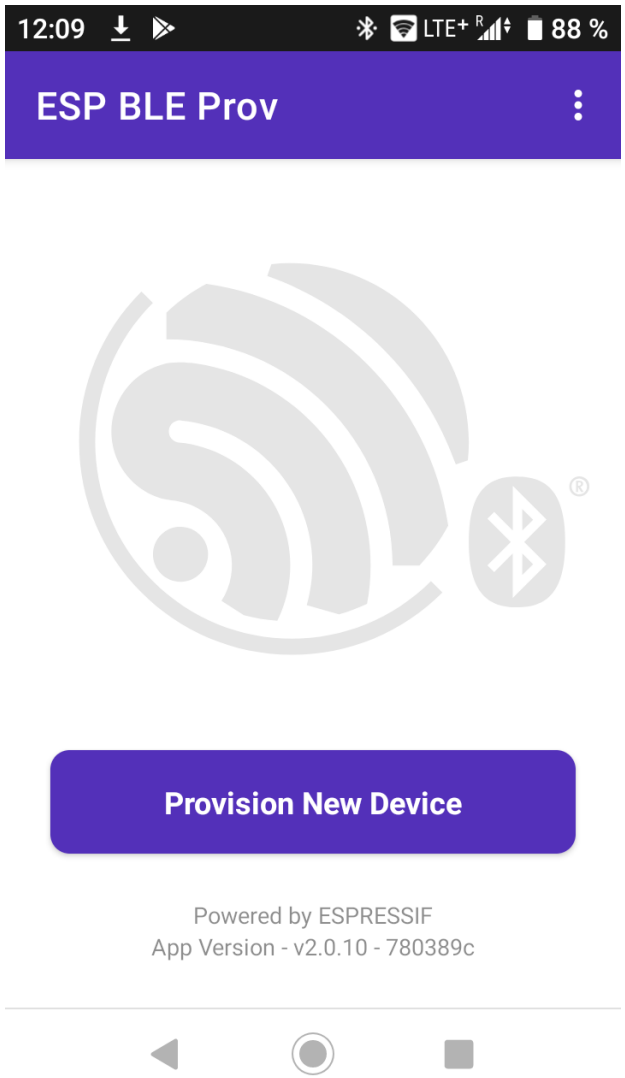
. . . and the last step is to check for a confirmation by the device that provisioning was successful.

The device should reboot and connect to the configured Wi-Fi network. While connecting the device's LED shines GREEN.

When it stops being GREEN, that means, it successfully connected. If it continues to shine GREEN, it means it does not find the network or cannot connect.

If you click on the button "OK" . . .

ULISP DEMO



ESP BLE PROVISIONING – STEP BY STEP

21.

. . . you get back to the starting screen of the provisioning app. You can end the app, or start over provisioning another device.

[Retzlaff 2022a]

ULISP DEMO WI-FI PROVISIONING VIA BLUETOOTH BLE

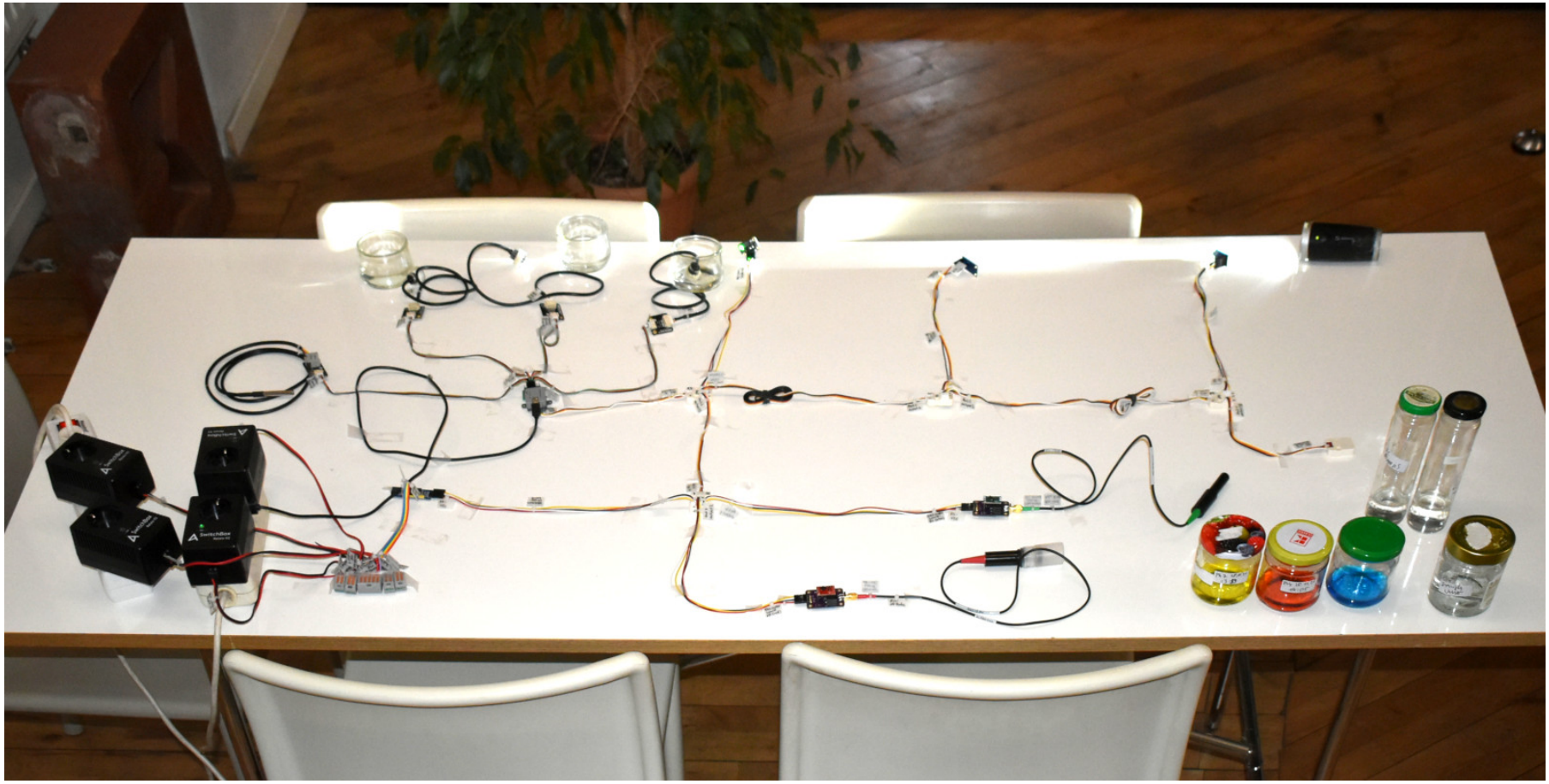
```
ets Jun  8 2016 00:22:57
rst:0x1 (POWERON_RESET)
Provisioning started: Use app to supply credentials of your access point
Waiting for provisioning to end

Received Wi-Fi credentials
   SSID : Falafel
   Password : fd2f12c66225

Connected IP address : 192.168.43.100
Provisioning Successful
Provisioning Ends
Provisioning Deinit
Waiting for provisioning to deinit
Waiting for WiFi

WiFi Connected:
   SSID : Falafel
   Password : <hidden>
Explicit reboot.
```

ULISP DEMO

WI-FI PROVISIONING VIA BLUETOOTH BLE

```
ets Jun  8 2016 00:22:57
rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
...
entry 0x400806a8

Provisioning Deinit
Waiting for provisioning to end
Waiting for provisioning to deinit
Waiting for WiFi
.
Connected IP address : 192.168.43.100

WiFi Connected:
   SSID : Falafel
   Password : <hidden>
uLisp 3.6
12127> (init-ntp)
nil

12127> (get-time)
"2022-03-21T15:00:06Z"
```

SENSOR DEVICE FOR AN AUTOMATED IOT DEVICE MAX-GERD RETZLAFF, NOV. 2021



[Retzlaff 2021b]

SENSOR DEVICE FOR AN AUTOMATED IOT DEVICE MAX-GERD RETZLAFF, NOV. 2021

- ten environmental sensors
- four controllable power sockets to activate environmental control measures
- Wi-Fi provisioned via Bluetooth (BLE)
- communication with a REST backend over Wi-Fi via JSON and HTTPs
- JSON protocol based on RelaxNG schemas (json-rnc)
- sends sensor readings to backend
- retrieves commands from a controlling smartphone application (from a different party) to control the actors or to calibrate the more complicated sensors
- liquid temperature sensor connected via one wire bus
- most sensors and actors connected via I2C bus
  with libraries completely written in uLisp

D I S C U S S I O N

[Retzlaff 2022c]

T H A N K Y O U

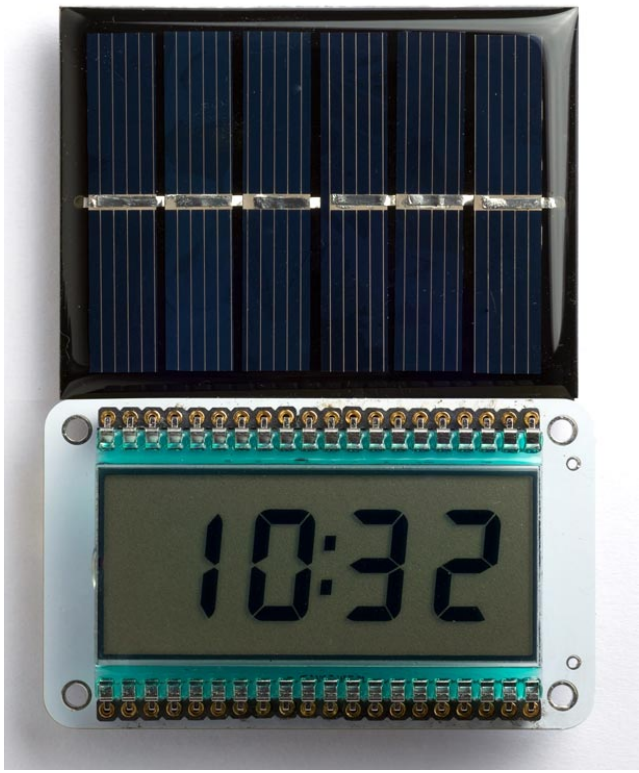

defsystem
Defining connected systems
Max-Gerd Retzlaff, Diplom-Informatiker

A P P E N D I X

SOLAR CLOCK

DAVID JOHNSON-DAVIES, 2021

A low-power LCD clock, with an average current consumption as about 9μA at 3.3V, designed to run from a solar cell allowing indefinite off-grid operation.



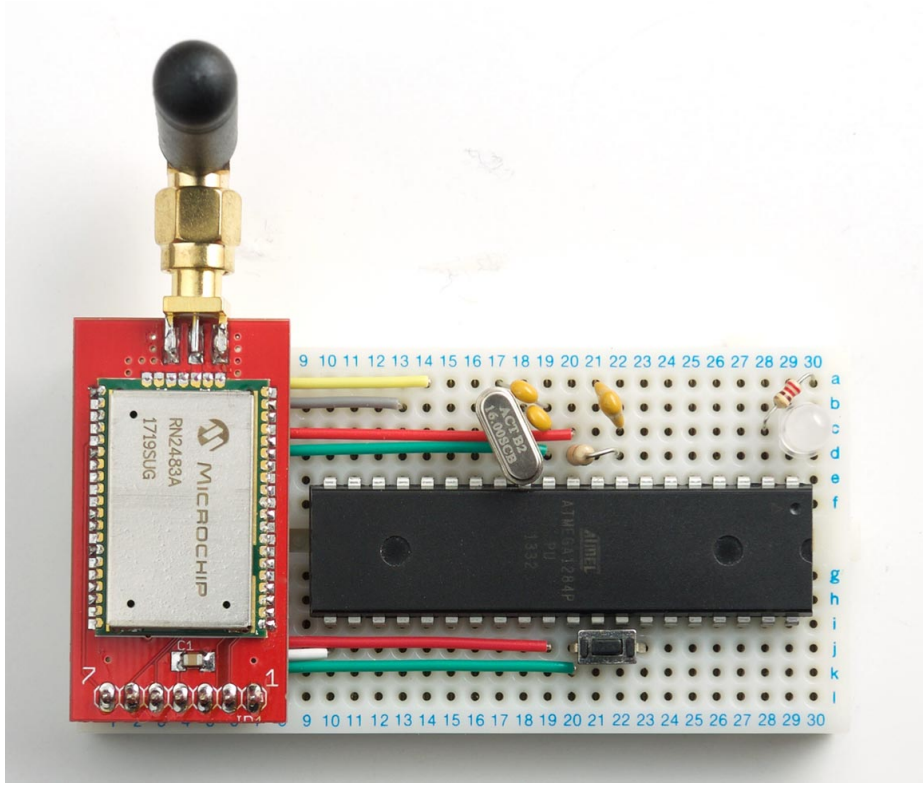
[Johnson-Davies 2021d]

LORAWAN NODE USING ULISP

DAVID JOHNSON-DAVIES, 2017

A LoRaWAN node, based on a Microchip RN2483A interfaced to an ATmega1284, and controlled by a program written in uLisp, capable of sending data to The Things Network.

See also `http://www.ulisp.com/show?1XWL`.



LISP BADGE

DAVID JOHNSON-DAVIES, 2019

A self-contained computer with its own display and keyboard, based on an ATmega1284 microcontroller, that can be programmed in uLisp, a subset of Common Lisp.

See also `http://www.ulisp.com/show?2L0C`.

[Johnson-Davies 2019]

Nurullah Akkaya's article "A micro-manual for LISP Implemented in C"
in early stages of designing the interpreter.

Bob Nystrom's article "Baby's First Garbage Collector"
for the garbage collector.

Section "A Properly Tail-Recursive Interpreter"
in Peter Norvig's classic book "Paradigms of Artificial Intelligence Programming"
for help with making the interpreter tail-recursive.

User clawson on AVR Freak (`https://www.avrfreaks.net/`)
when designing a procedure name lookup table in program memory.

B I B L I O G R A P H Y

REFERENCES

DOIT 2018
DOIT, Shenzhen Sibo Zhilian Technology Co Ltd: *Picture of DOIT ESP32 DEVKIT V1*. Promotional picture, 2018. `http://doit.am`.

Johnson-Davies 2014a
JOHNSON-DAVIES, David: *uLisp implementation page*. Web page, 2014-2022. `http://www.ulisp.com/show?1AWG`.

Johnson-Davies 2014b
JOHNSON-DAVIES, David: *uLisp specification page*. Web page, 2014-2022. `http://www.ulisp.com`.

Johnson-Davies 2017
JOHNSON-DAVIES, David: *Picture of a LoRaWAN Node using uLisp by David Johnson-Davies*. Photography, 2017. http://www.ulisp.com/show?1XWL.

Johnson-Davies 2019
JOHNSON-DAVIES, David: *Picture of the Lisp Badgeby David Johnson-Davies*. Photography, 2019. http://www.ulisp.com/show?2L0C.

Johnson-Davies 2021a
JOHNSON-DAVIES, David: *Design objective of uLisp*. E-mail, 2021. Personal correspondence, Feb 5th, 2021.

Johnson-Davies 2021b
JOHNSON-DAVIES, David: *On the name of uLisp*. E-mail, 2021. Personal correspondence, May 6th, 2021.

Johnson-Davies 2021c
JOHNSON-DAVIES, David: *Picture of David Johnson-Davies*. Photography, 2021. Personal correspondence.

Johnson-Davies 2021d
JOHNSON-DAVIES, David: *Picture of the Solar Clock by David Johnson-Davies*. Photography, 2021. Personal correspondence.

M5Stack 2020
M5STACK: *Picture of M5Stack Atom Lite ESP32*. Promotional picture, 2020.

Retzlaff 2021a
RETZLAFF, Max-Gerd: *Picture of a Kombucha sensor device*. Photography, 2021.

Retzlaff 2021b
RETZLAFF, Max-Gerd: *Picture of a sensor device for an automated IoT device*. Photography, 2021.

Retzlaff 2021c

RETZLAFF, Max-Gerd: *Picture of assembly of the Scooter Lisp computer*. Photography, 2021.

Retzlaff 2021d

RETZLAFF, Max-Gerd: *Picture of M5Stack ESP32 Basic Core*. Photography, 2021.

Retzlaff 2021e

RETZLAFF, Max-Gerd: *Picture of M5Stack stand-alone uLisp computer*. Photography, 2021.

Retzlaff 2021f

RETZLAFF, Max-Gerd: *Picture of replaced speedometer clock with Scooter Lisp*. Photography, 2021.

Retzlaff 2021g

RETZLAFF, Max-Gerd: *Picture of replaced speedometer clock with Scooter Lisp #2*. Photography, 2021.

Retzlaff 2021h
RETZLAFF, Max-Gerd: *Picture of the replacement of the display of the speedometer clock*. Photography, 2021.

Retzlaff 2021i
RETZLAFF, Max-Gerd: *Screenshot of an Observable notebook by Datagenous*. Screenshot picture, 2021. `http://corp.datageno.us`.

Retzlaff 2021j
RETZLAFF, Max-Gerd: *Screenshot of repository overview of the Dydra graph database by Datagraph*. Screenshot picture, 2021. `http://www.dydra.com`.

Retzlaff 2021k
RETZLAFF, Max-Gerd: *Six pictures of the display of the Kombucha sensor device*. Photography, 2021.

Retzlaff 2022a

RETZLAFF, Max-Gerd: *21 screenshots of the ESP BLE Prov Android app by Espressif*. Screenshot picture, 2022.

Retzlaff 2022b

RETZLAFF, Max-Gerd: *Picture of a presenter device*. Photography, 2022.

Retzlaff 2022c

RETZLAFF, Max-Gerd: *Picture of twelve defsystem atoms*. Photography, 2022.

Retzlaff 2022d

RETZLAFF, Max-Gerd: *Picture of using an Android smartphone to connect to an M5Stack Atom Lite ESP32 (while on an air plane)*. Photography, 2022.

– End of Document –