tass
TECHNOLOGY SOLUTIONS

picoTCP

The reference TCP/IP stack for the Internet of Things

FOSDEM 2014

tass TSS TECHNOLOGY SOLUTIONS

A leading **embedded** technology company, located in Leuven and Gent, providing **Consultancy**, **Training**, **Projects** and **Outsourced Services**

We ensure that our clients in **co-creation** with TASS can launch innovative and better **quality** products with a shorter **time to market**.

We believe "**Connected devices**" improve the quality of our lives in an innovative way.

That’s why we made this...

tass
TECHNOLOGY SOLUTIONS

picoTCP

A fully featured, highly portable **TCP/IP stack** designed for **small footprint** **embedded systems.**

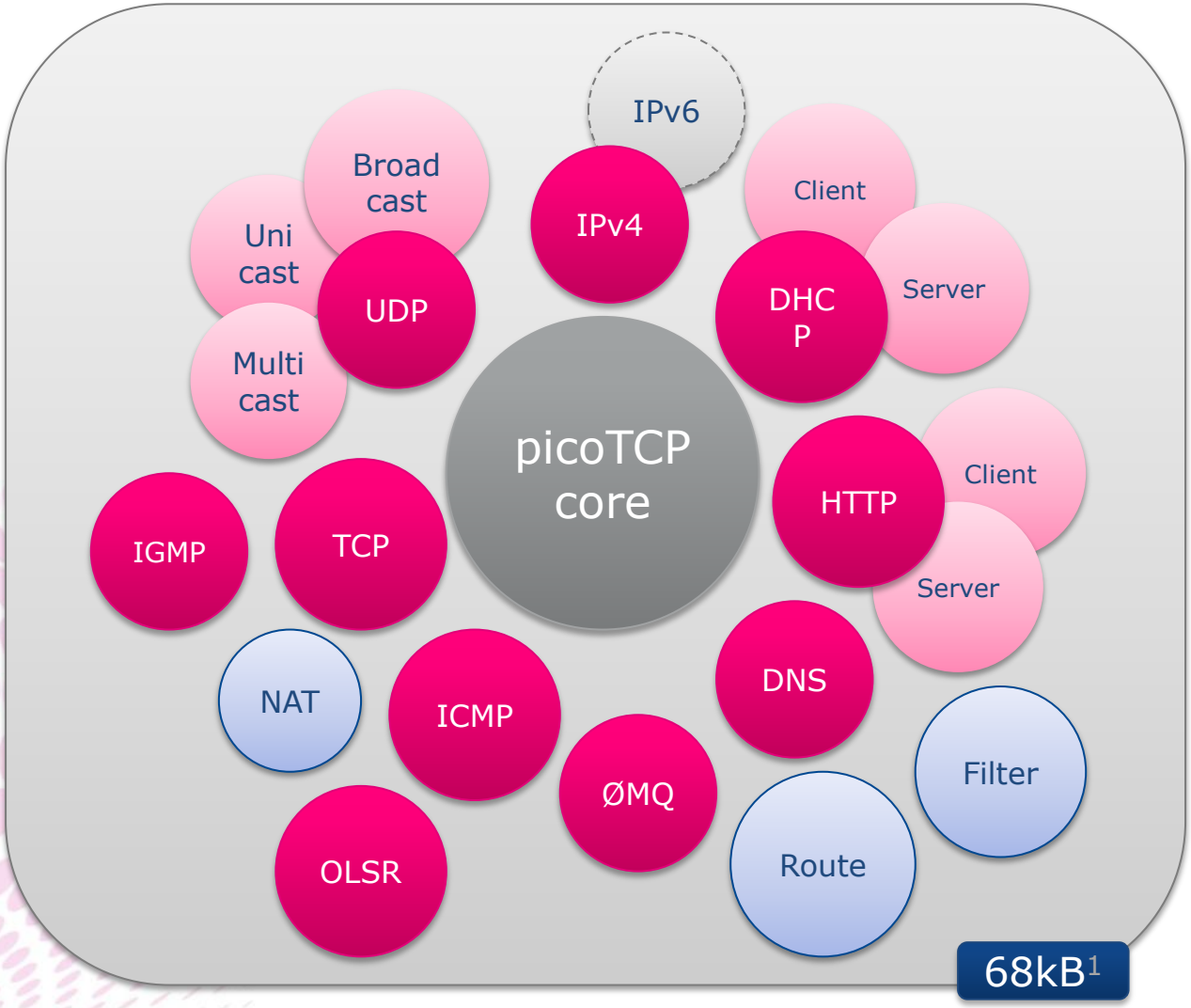
The reference TCP/IP stack for the Internet of Things

Our focus areas:

Free as in free speech

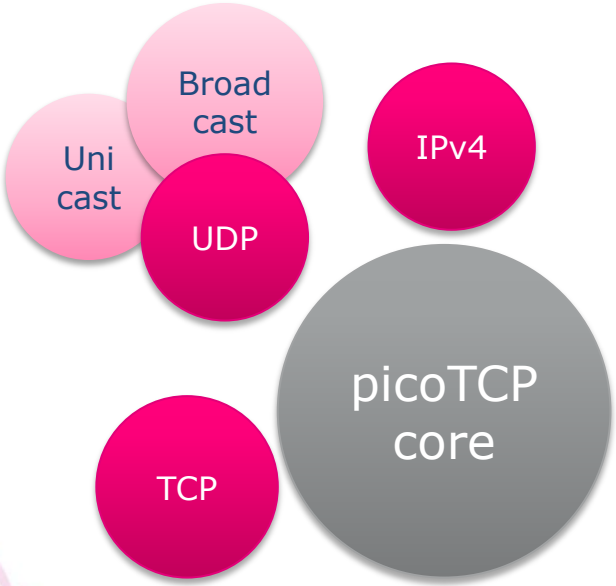
- Dual licensing policy in place
  - Freely distributed under the **GNU General Public License v2** for the benefit of the community
  - **Proprietary license available** at user's option, when the platform code can't fully comply with the terms of GPL
  - **Full copyright is owned by TASS:** different licensing agreements are possible for special cases

Modularity



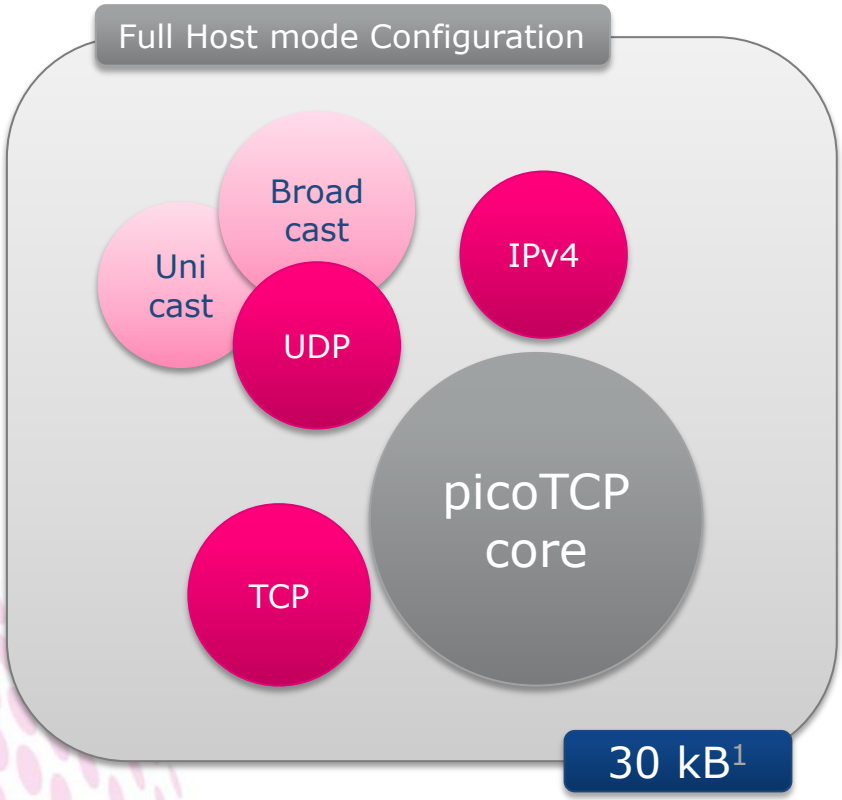
[1] Figures for ARM Cortex-M3, gcc compiler

Modularity – Full Host Mode Config



[1] Figures for ARM

Modularity– Full Host Mode Config



[1] Figures for ARM

Modularity


```
$> make ARCH=stm32 CROSS_COMPILE=arm-none-eabi- \
                TCP=1 UDP=1 IPV4=1 IPFRAG=0 NAT=0 ICMP4=0 \
                MCAST=0 DEVLOOP=0 PING=0 DHCP_CLIENT=0 \
                DHCP_SERVER=0 DNS_CLIENT=0 IPFILTER=0 \
                CRC=0 HTTP_CLIENT=0 HTTP_SERVER=0 ZMQ=0 \
                OLSR=0 SLAACV4=0 STRIP=1 DEBUG=0
  [CC] ...
  [CC] ...
  [AR] ./build/lib/libpicotcp.a
  [RANLIB] ./build/lib/libpicotcp.a
  [STRIP] ./build/lib/libpicotcp.a
  [LIBSIZE] 30508         ./build/lib/libpicotcp.a
```

Portability

- CPU Architecture independent
- 8, 16, 32 & 64 bit
- Big or Little endian
- Bare Metal / Embedded OS / OS / RTOS
- 10+ different platforms already supported
- (RT)OS easily added, because of our OS Abstraction Layer

Portability

Supported architectures

- 32 bit
  - ARM variants:
    - LPC1768 (Cortex-M3)
    - LPC4357 (Cortex-M0 + Cortex-M4)
    - STM32 variants
- 16 bit
  - MSP430
  - PIC24Fxxxx
- 8 bit
  - AVR ATmega128

Portability

Supported Communication channels/chips

- BCM43362 (IEEE 802.11)
- MRF24WG (IEEE 802.11)
- LPC Ethernet ENET/EMAC (IEEE 802.3)
- Stellaris Ethernet (IEEE 802.3)
- Wiznet W5100 (IEEE 802.3)
- USB CDC-ECM (CDC1.2)
- Virtual drivers
  - TUN/TAP
  - VDE
  - libpcap

Portability

Supported RTOSes

- No OS / Bare metal
- FreeRTOS
- mbed-RTOS (CMSIS-RTOS compliant)
- Linux / POSIX

Our OS Abstraction Layer provides:

- Abstraction from mutexes/signals
- Abstraction from threads/processes
- Blocking socket interface

Performance

Performance measurements on MBED NXP LPC176x @ 100 MHz

Throughput

- TCP Tx: 10,09 Mbit/s
- TCP Rx: 20,23 Mbit/s

CPU time

- Idle state: 45 µs / loop
- Max throughput: 832 µs / loop

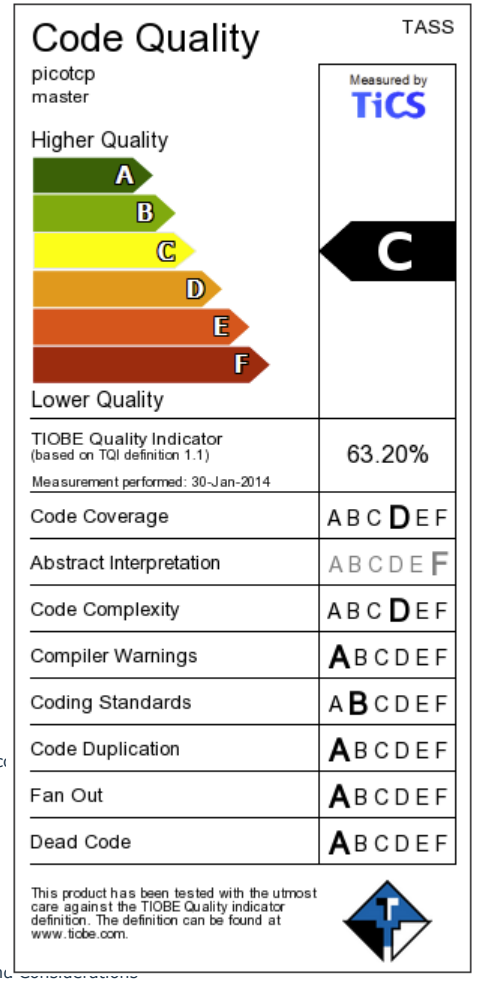
RAM Usage

- Dyn. Memory: Min. 1kB Max. 8,3kB Avg. 4,7kB
- Static Memory: 16kB
- Stack: 2kB

Quality

- Quality oriented development environment
  - Agile development (SCRUM)
  - Test Driven Development
  - Continuous Integration & Automated testing (Virtual testing is possible)
  - Code Quality check: Tiobe Tics Continuous improvement

Project: picotcp



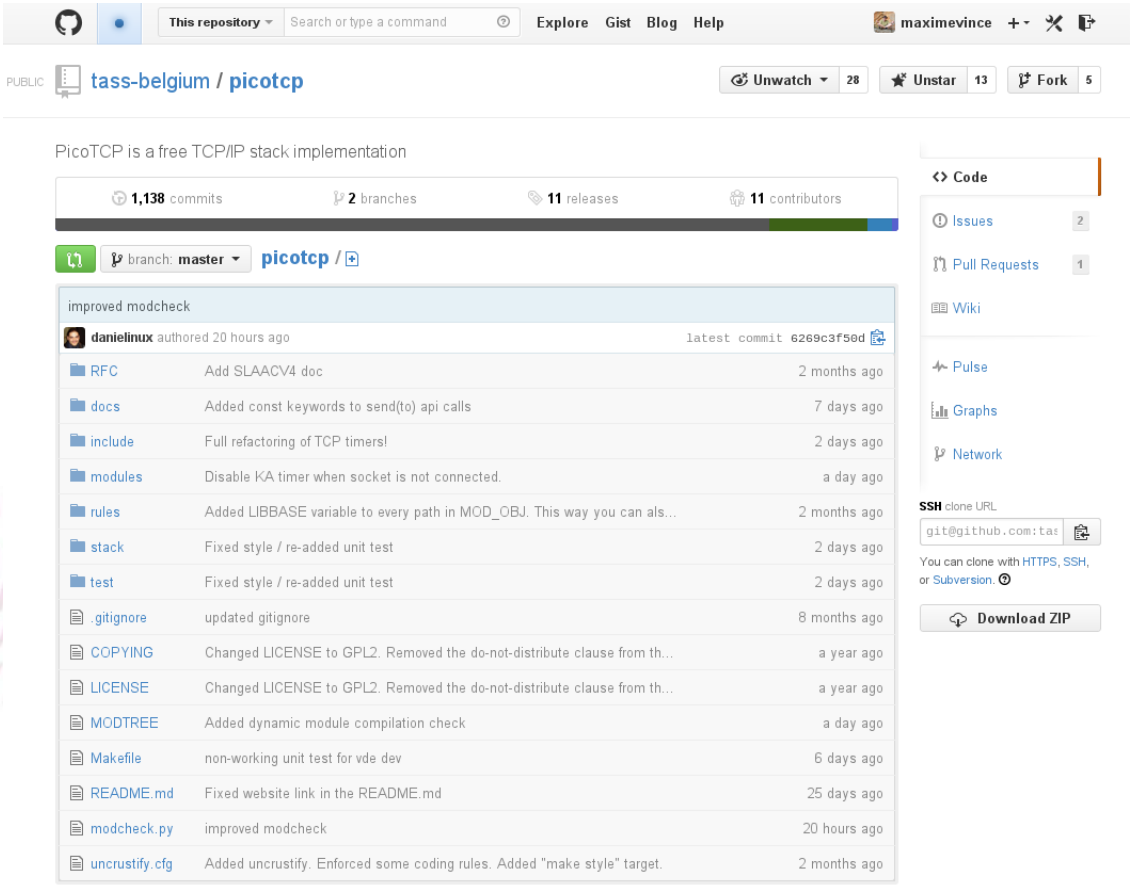
- Full compliance with IETF TCP/IP standards

RFC 768 User Datagram Protocol (UDP)
RFC 791 Internet Protocol (IP)
RFC 792 Internet Control Message Protocol (ICMP)
RFC 793 Transmission Control Protocol (TCP)
RFC 816 Fault Isolation and Recovery
RFC 826 Address Resolution Protocol (ARP)
RFC 879 The TCP Maximum Segment Size and Related Topics
RFC 894 IP over Ethernet
RFC 896 Congestion Control in IP/TCP Internetworks
RFC 919 Broadcasting Internet Datagrams
RFC 922 Broadcasting Internet Datagrams in the Presence of Subnets
RFC 950 Internet Standard Sub-netting Procedure
RFC 1009 Requirements for Internet Gateways
RFC 1034 Domain Names Concepts and Facilities
RFC 1035 Domain Names Implementation and Specification
RFC 1071 Computing the Internet Checksum
RFC 1112 Internet Group Management Protocol (IGMP)
RFC 1122 Requirements for Internet Hosts Communication Layers
RFC 1191 Path MTU Discovery
RFC 1323 TCP Extensions for High Performance
RFC 1337 TIME-WAIT Assassination Hazards in TCP

RFC 1534 Interoperation Between DHCP and BOOTP
RFC 1542 Clarifications and Extensions for the Bootstrap Protoc
RFC 1812 Requirements for IP Version 4 Routers
RFC 1878 Variable Length Subnet Table For IPv4
RFC 1886 DNS Extensions to Support IP Version 6 (1)
RFC 2018 TCP Selective Acknowledgment Options
RFC 2131 Dynamic Host Conguration Protocol (DHCP)
RFC 2132 DHCP Options and BOOTP Vendor Extensions
RFC 2236 Internet Group Management Protocol, Version 2
RFC 2460 Internet Protocol, Version 6 (IPv6) Specification (1)
RFC 2581 TCP Congestion Control
RFC 2616 Hypertext Transfer Protocol { HTTP/1.1
RFC 2663 IP Network Address Translator (NAT) Terminology and
RFC 3042 Enhancing TCP's Loss Recovery Using Limited Transmit
RFC 3315 Dynamic Host Configuration Protocol for IPv6 (DHCPv6) (1)
RFC 3376 Internet Group Management Protocol, Version 3 (2)
RFC 3517 A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP
RFC 3782 The NewReno Modification to TCP's Fast Recovery Algorithm
RFC 4291 IP Version 6 Addressing Architecture (1)
RFC 6691 TCP Options and Maximum Segment Size (MSS)

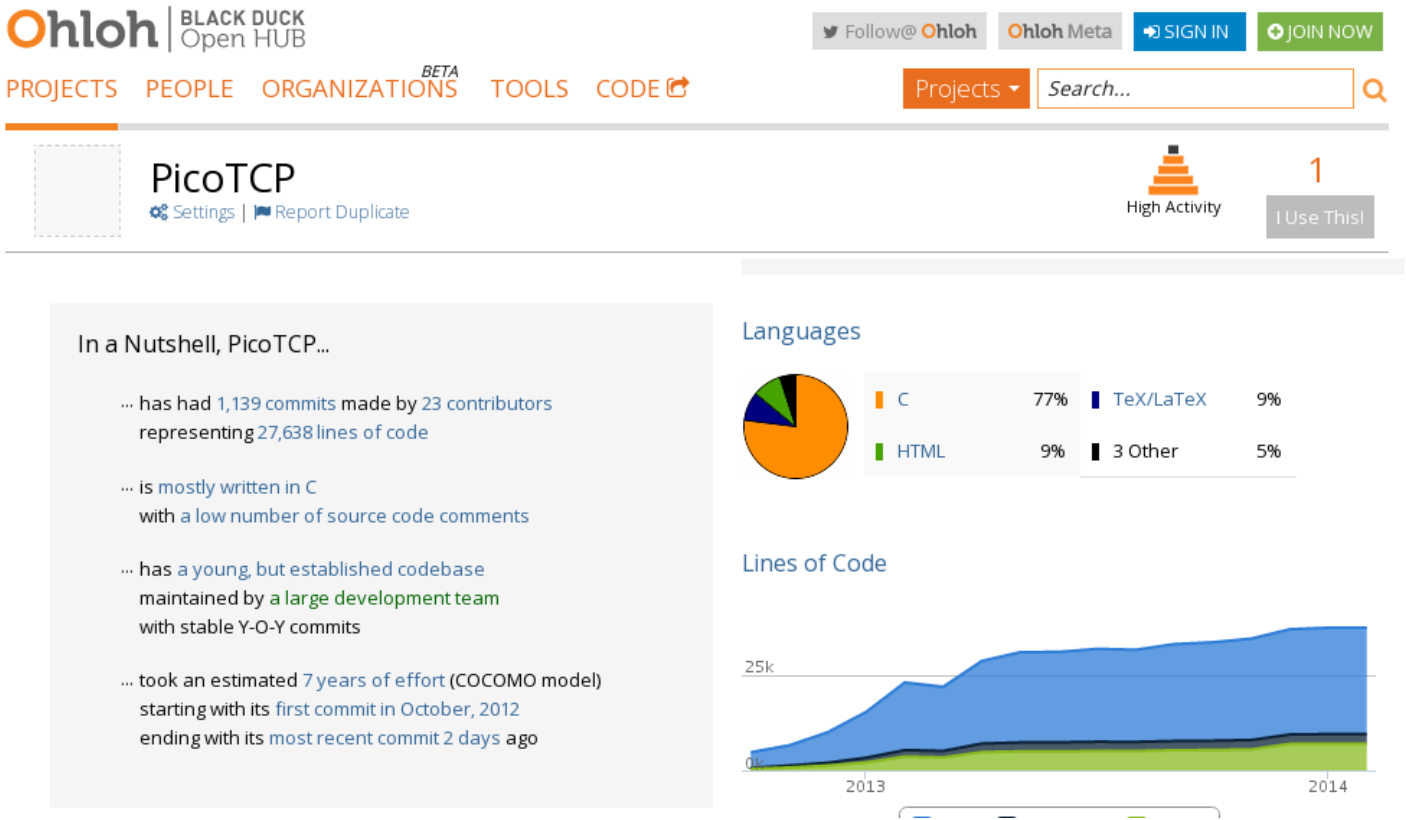
Community driven

- Community driven: We're on GitHub
  - Full source code freely available
  - Public issue-tracking system
  - GPLv2 licensed
  - Dual licensing possible



Community driven

- Ohloh says PicoTCP …
  - has a young, but established codebase
  - is maintained by a large development team
  - with stable Y-O-Y commits
  - took an estimated 7 years of effort (COCOMO model)
  - starting with its first commit in October, 2012



How to port to platform XYZ?

- Easy to port!
- You need:
  - **Memory management:**
    - Malloc/free functions
  - **Timing functions:**
    - Returning the amount of elapsed milliseconds / seconds
  - **A driver for your hardware**
    - Your way to the physical layer

include/arch header

- Memory management:
  - use malloc() and free() from your C-lib
  - Use memory allocation from your RTOS
  - Or our own memory manager which is on it's way

- Timing functions:
  - Timer ISR every millisecond, incrementing a 64-bit integer
  - PICO_TIME_MS() returning this integer as uint64_t
  - PICO_TIME() returning this integer / 1000 as uint64_t

```
extern volatile uint32_t lpc_tick;
extern volatile pico_time full_tick;

#define pico_free(x) free(x)

static inline void *pico_zalloc(size_t size)
{
    void *ptr = malloc(size);
    if(ptr)
        memset(ptr, 0u, size);
    return ptr;
}

static inline pico_time PICO_TIME_MS(void)
{
    if ((full_tick & 0xFFFFFFFF) > lpc_tick) {
        full_tick +=  0x100000000ULL;
    }
    full_tick = (full_tick & 0xFFFFFFFF00000000ULL) + lpc_tick;
    return full_tick;
}

static inline pico_time PICO_TIME(void)
{
    return PICO_TIME_MS() / 1000;
}

static inline void PICO_IDLE(void)
{
    uint32_t now = lpc_tick;
    while(now == lpc_tick);
}
```

Device driver

- Generic device driver API to PicoTCP:
  - struct pico_device * **pico_dev_create**(void)
  - struct **pico_device**
    ->send = pico_dev_send;
    ->poll = pico_dev_poll;
    ->destroy = pico_dev_destroy;
  - int **pico_dev_send**(struct pico_device *dev, void *buf, int len)
  - int **pico_dev_poll**(struct pico_device *dev, int loop_score)
  - void **pico_dev_destroy**(struct pico_device *dev)

```c
#ifndef PICO_DEV_MBED_H
#define PICO_DEV_MBED_H

#include "pico_config.h"
#include "pico_device.h"

void pico_mbed_destroy(struct pico_device *vde);
struct pico_device * pico_mbed_create(char *name);

#endif  /* PICO_DEV_MBED */
```

```
struct pico_device * pico_mbed_create(char *name)
{
    uint8_t mac[PICO_SIZE_ETH];
    struct pico_device_mbed *mb =
                    pico_zalloc(sizeof(struct pico_device_mbed));
    if (!mb)
      return NULL;

    ethernet_address(mac);

    if( 0 != pico_device_init((struct pico_device *)mb, name, mac))
    {
      dbg ("Loop init failed.\n");
      pico_loop_destroy(mb);
      return NULL;
    }

    mb->dev.send = pico_mbed_send;
    mb->dev.poll = pico_mbed_poll;
    mb->dev.destroy = pico_mbed_destroy;
    mb->bytes_left_in_frame = 0;

    if(ethernet_init() != 0) {
      dbg("Failed to initialize hardware.\n");
      pico_device_destroy((struct pico_device *)mb);
      return NULL;
   }

   dbg("Device %s created.\n", mb->dev.name);
   return (struct pico_device *)mb;
}
```

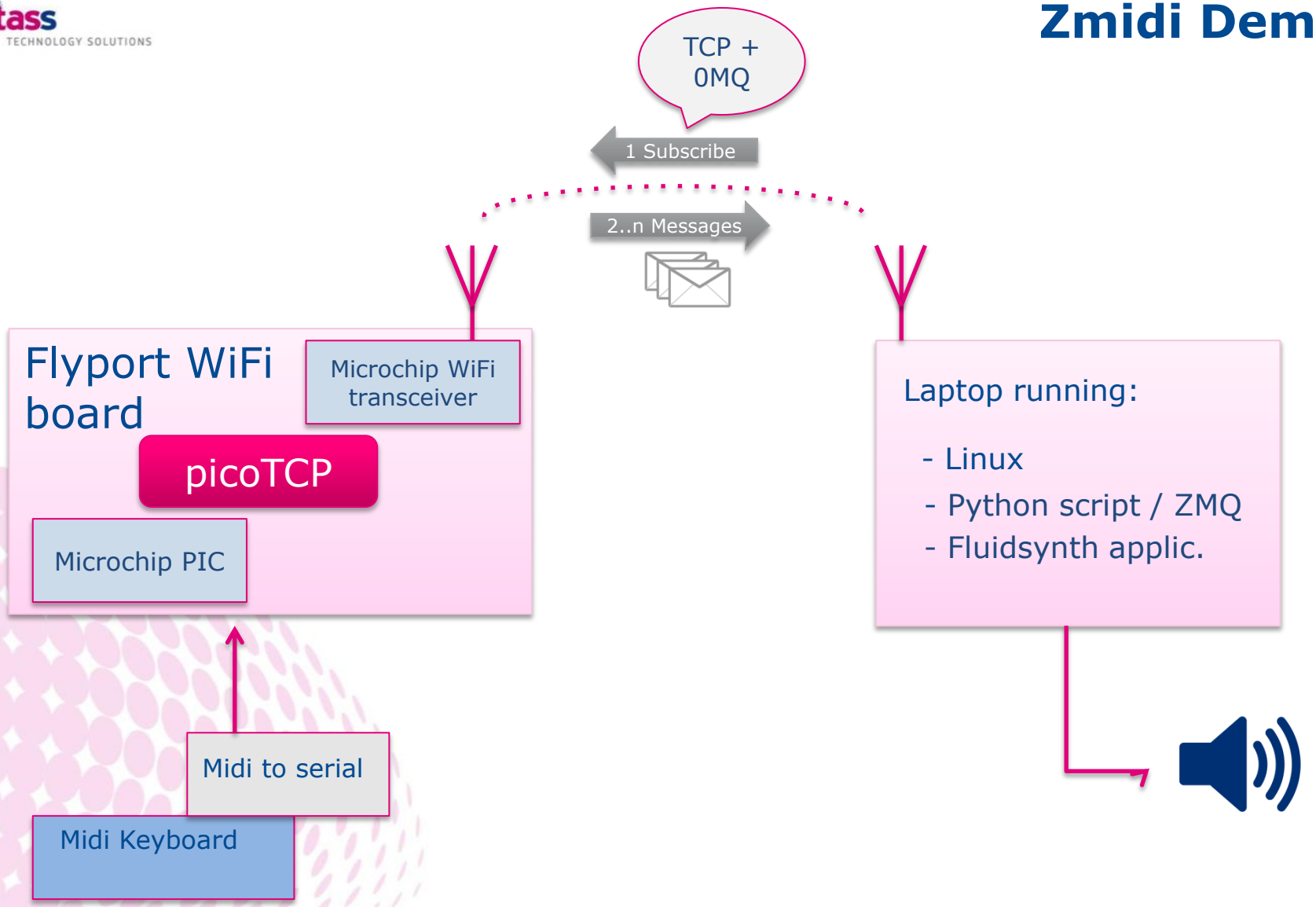
```c
static int pico_mbed_poll(struct pico_device *dev, int loop_score)
{
  int len;
  struct pico_device_mbed *mb = (struct pico_device_mbed *) dev;

  while(loop_score != 0)
  {    /* check for a new frame */

    if(mb->bytes_left_in_frame == 0)
    {
      if((len = ethernet_receive()) > 0)
        mb->bytes_left_in_frame = len;
    }

    if(mb->bytes_left_in_frame == 0)
      return loop_score;     /* read a frame */

    len = ethernet_read((char *)buf), ETH_MAX_FLEN);

    if(len != mb->bytes_left_in_frame)
      return -1;

    dbg("MBed received %u new bytes\n",len);
    pico_stack_recv(dev, buf, len);

    loop_score--;
  }
  return loop_score;
}
```

```c
static int pico_mbed_send(struct pico_device *dev, void *buf, int len)
{
  int ret, sent;
  struct pico_device_mbed *mb = (struct pico_device_mbed *) dev;

  if (len > ETH_MAX_FLEN)
    return -1;

  dbg("MBed sending %d bytes ...\n",len);

  /* Write buf content to dev and return amount written */
  ret = ethernet_write((const char *)buf, len);
  sent = ethernet_send();
  dbg("MBed sent %d bytes\n",ret);

  if (len != ret || sent != ret)
    return -1;
  else
    return ret;
}
```

Demo Time!

Zmidi Demo

Zmidi Demo


TCP +
0MQ
1 Subscribe
2..n Messages
Flyport WiFi board
Microchip WiFi transceiver
picoTCP
Microchip PIC
Midi to serial
Midi Keyboard
Laptop running:
- Linux
- Python script / ZMQ
- Fluidsynth applic.

Check us out!

picoTCP

Check us out on GitHub:

https://github.com/tass-belgium/picotcp.git