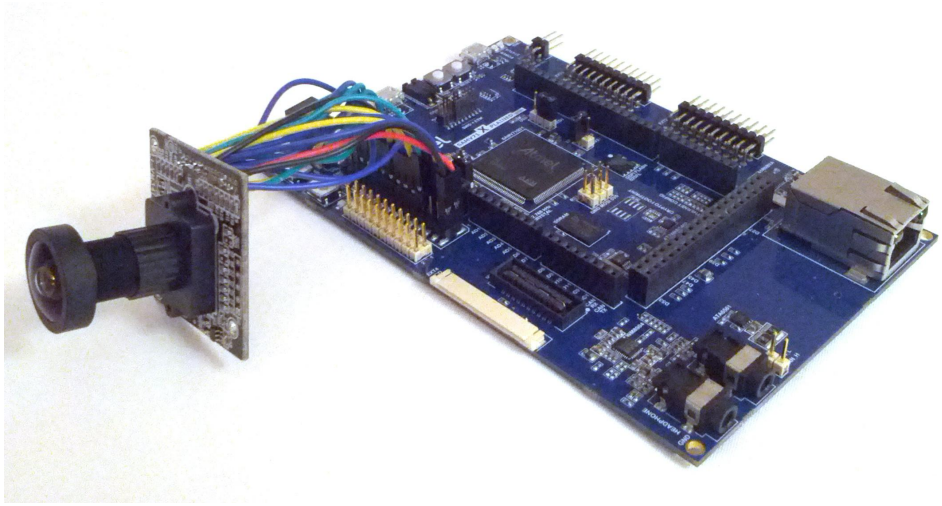
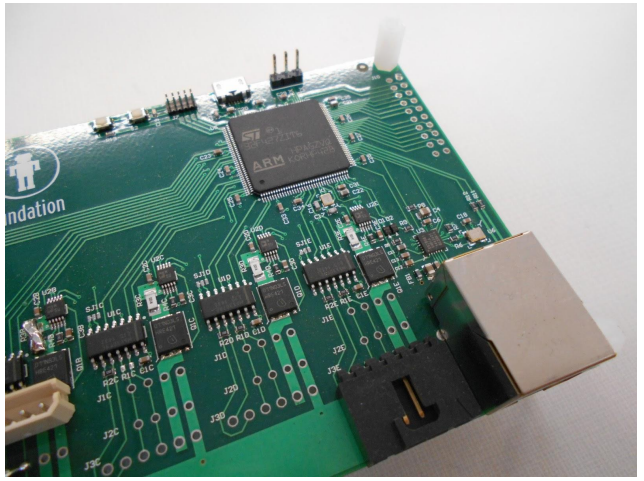
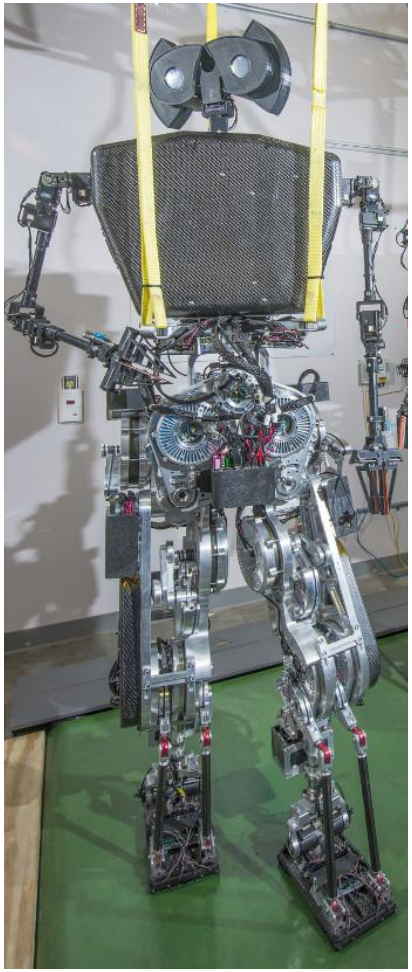


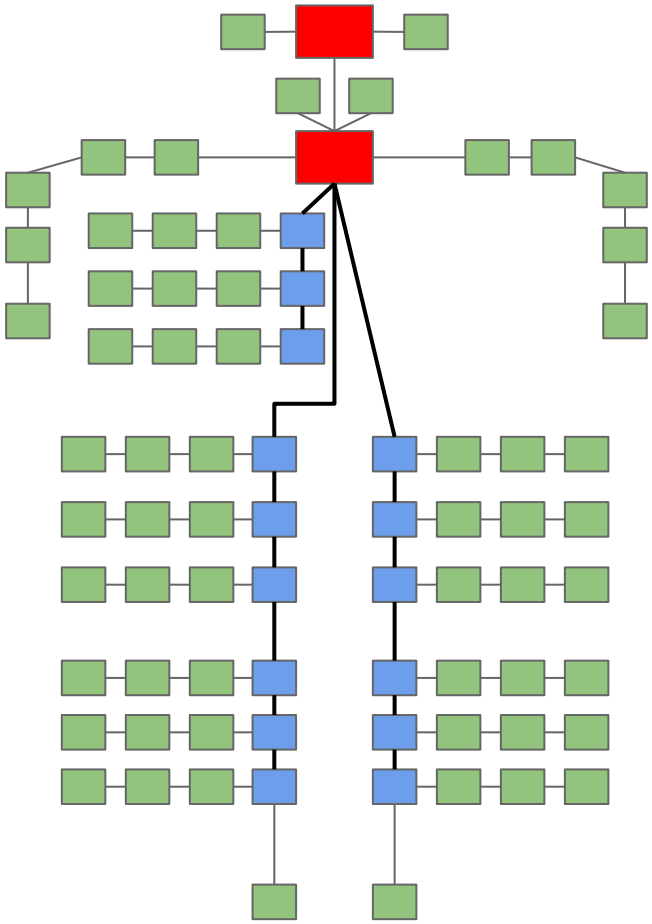


ROS2 on "small" embedded systems

Morgan Quigley
OSRF

Small embedded systems are everywhere!

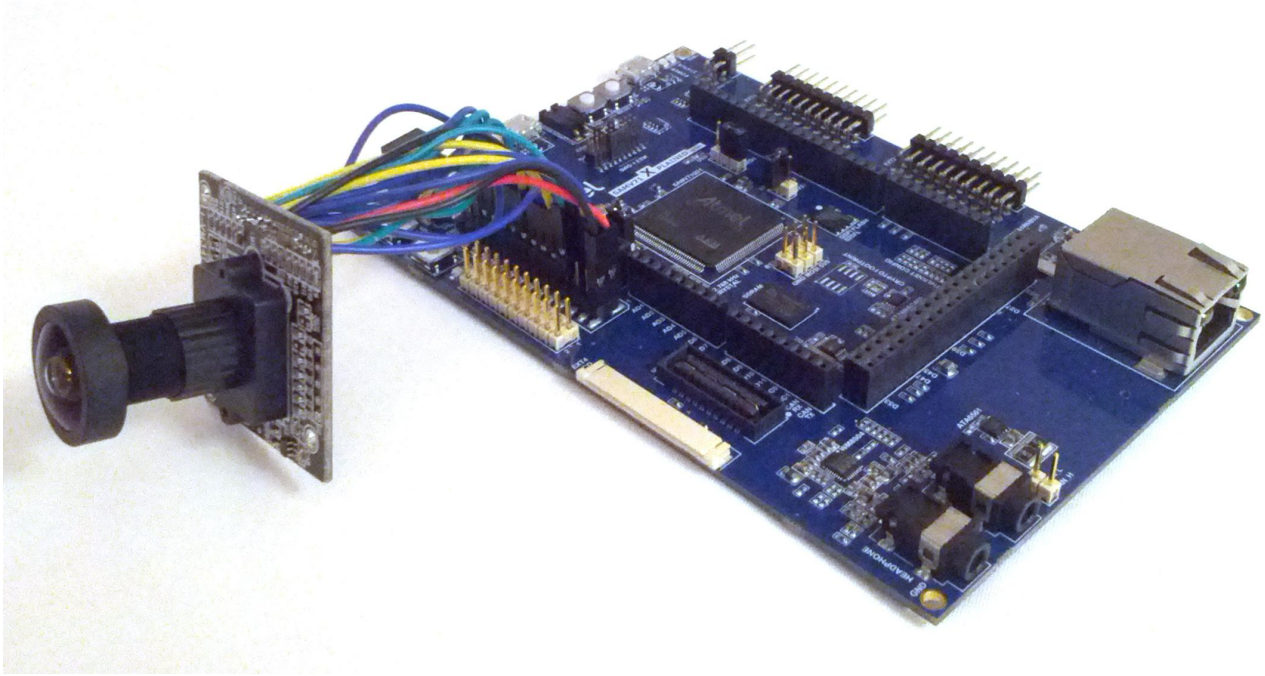




Red: x86
Green: MCU's
Blue: FPGA's

Goal #1: standardize wire protocols to allow generic tooling

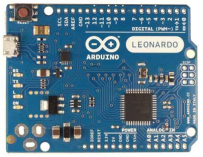
ROS2 can fit in sensors!



Goal #2: eliminate driver-nodes and push some real-time processing out to the sensors

- **Modern MCU's are fast!**
- **Real-time can be easier on small systems!**

Scope



	8/16-bit MCU	32-bit MCU	ARM A-class smartphone without screen	SFF x86 laptop without screen
Example Chip	Atmel AVR	STM32	Samsung Exynos	Intel Core i5
Example System	Arduino Leonardo	Pixhawk PX4	ODROID	Intel NUC
MIPS	10's	100's	1000's	10000's
RAM	1-32 KB	4-256 KB	a few GB (off-chip)	2-16 GB (SODIMM)
Max power	10's of mW	100's of mW	1000's of mW	10000's of mW
Comms peripherals	UART, USB FS, ...	USB HS, Ethernet	Gigabit Ethernet	USB SS, PCIe

Future work

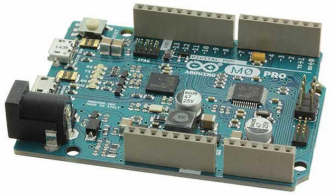
This talk

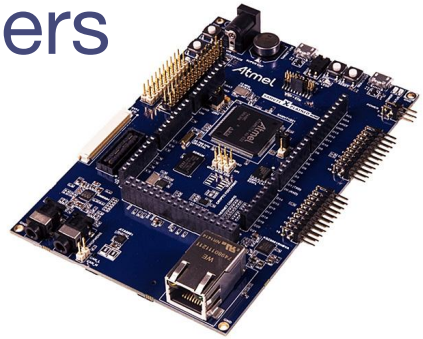
"normal" ROS2

Open Source Robotics Foundation

Single-chip 32-bit microcontrollers

Large price/performance tradeoff, even within this category





	"small" 32-bit MCU	"big" 32-bit MCU
Core	ARM Cortex-M0	ARM Cortex-M7
Speed	48 Mhz	300 Mhz
RAM	32 KB	384 KB
Flash	256 KB	2048 KB
Cost @ 1K units	$2	$10
Comms	USB FS	Ethernet, USB HS

Future work

This talk

ROS1

- startup sequencing
- XML-RPC discovery
  - parse XML trees
- TCP data streams
- UDPROS not complete
  - no multicast
  - no fragment retries
  - no "latched" topics

embedding becomes very difficult!

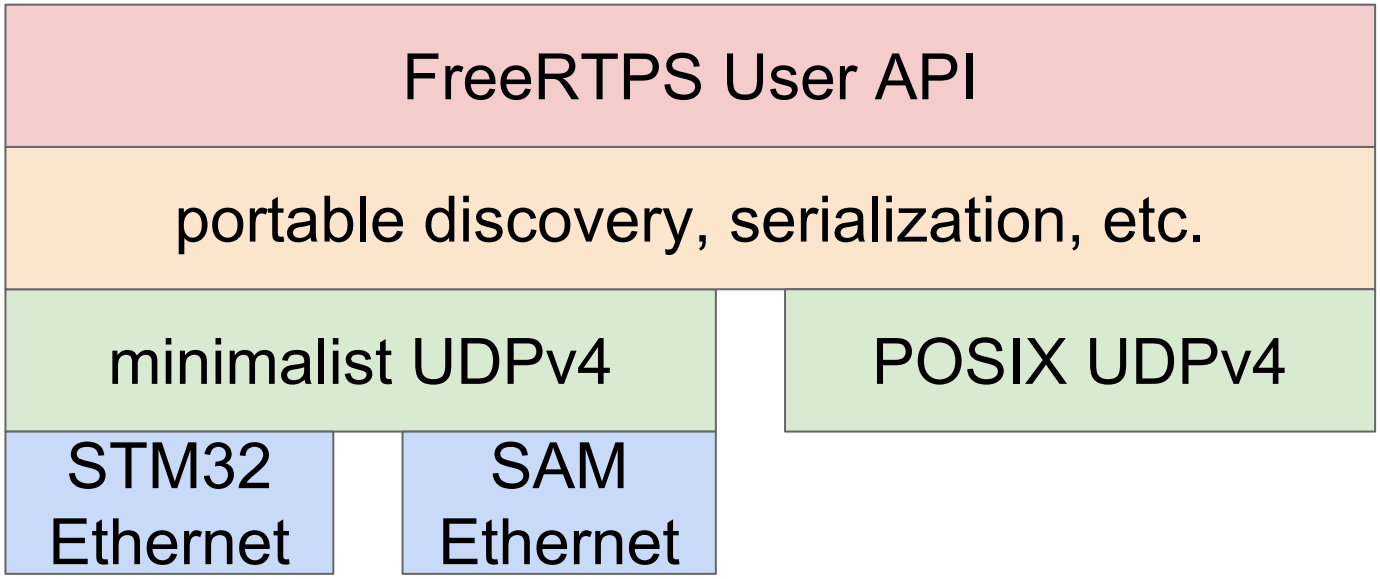
ROS2 (on RTPS)

- no master
- multicast UDP Discovery
  - parse parameter lists
- RTPS/UDP data streams
- extensive QoS on UDP
  - TCP-like
  - "fire and forget" UDP
  - everything in between

goal: show these benefits with free, portable, small code.

FreeRTPS: https://github.com/ros2/freertps

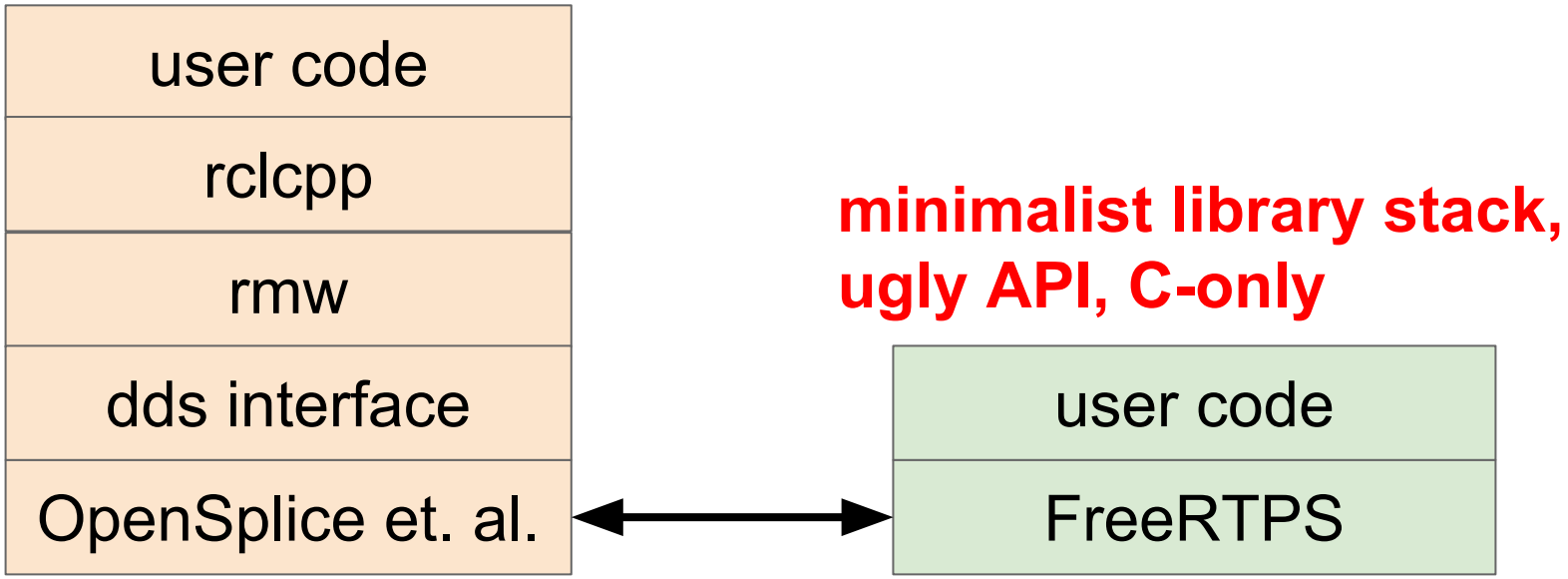
- Apache2 License
- RTPS (transport) and CDR (serialization)
- work in progress! **WARNING WARNING**
- can use on MCU's and on Linux



Full DDS stack is not strictly necessary

- Comms with ROS2 only needs RTPS !
- Don't need all of DDS, nor every possible QoS

flexible library stack,
elegant API via C++

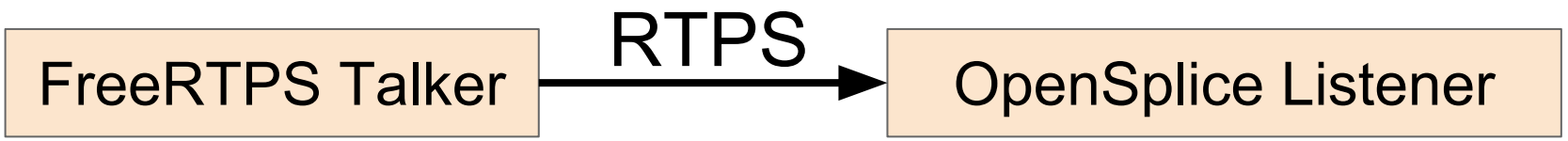


Static FreeRTPS nodes

- Buffer needs for RTPS discovery and communications are predictable
- All-static (compile-time) allocation is possible!
  - Node discovery buffers
  - Topic discovery buffers
  - Message send/receive buffers
- FreeRTPS currently does not use the heap
  - Tradeoff: finite number of nodes can be discovered

FreeRTPS talker example

- Minimal proof of concept: sending RTPS strings
  - same as ROS2 std_msgs::msg::String
- Node is all-static (no malloc, etc.)
  - RAM: ~100 KB
  - Flash: ~60 KB
- many potential size reductions are possible



Runs on POSIX, STM32F4,
STM32F7, Atmel SAM V71

```c
#include <stdio.h>
#include "freertps/freertps.h"
#include "std_msgs/string.h"

struct std_msgs__string msg;
char data_buf[64];
uint8_t cdr[68];

void main(int argc, char **argv)
{
  freertps_init();
  freertps_pub_t *pub = freertps_create_pub("chatter", std_msgs__string__type.typename);
  freertps_start();
  msg.data = data_buf;
  int pub_count = 0;
  while (true)
  {
     freertps_listen(500000);
     freertps_disco_tick();
     snprintf(msg.data, sizeof(data_buf), "Hello, world! %d", pub_count++);
     int cdr_len = serialize_std_msgs__string(&msg, cdr, sizeof(cdr));
     freertps_publish(pub, cdr, cdr_len);
     printf("sending: [%s]\r\n", data_buf);
  }
}
```

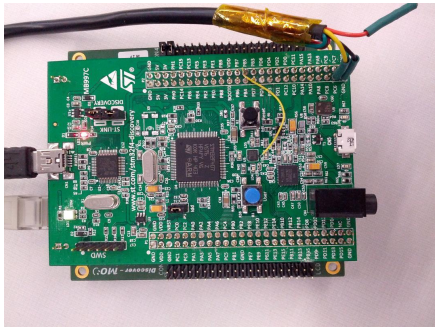
Static buffers for userland message and its serialization

tick the discovery machinery periodically

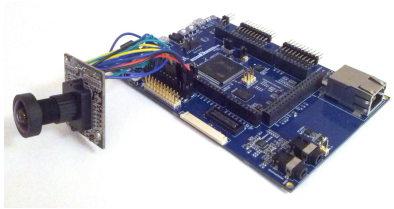
stuff, serialize, and send the message

Edge-node examples

- IMU demo via STM32F4-Discovery:
  - sends sensor_msgs::Imu @ 1 KHz
  - 100 KB RAM, 60 KB Flash



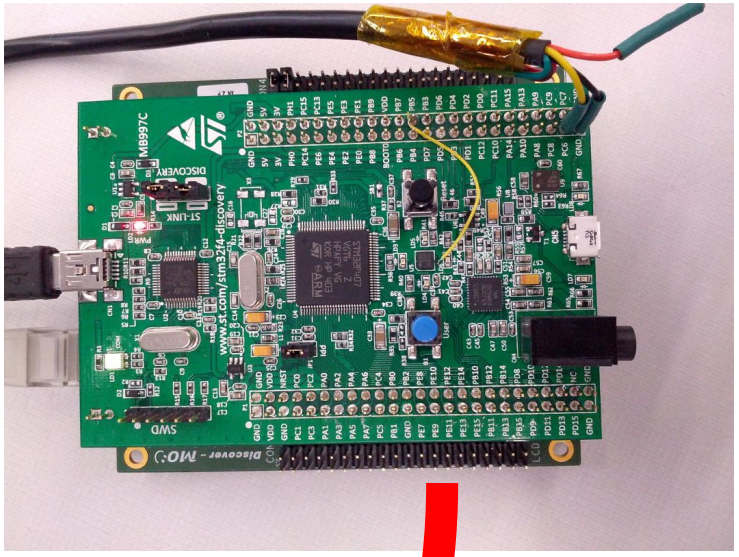
- Camera demo via Atmel SAM V71 Xplained:
  - sends sensor_msgs::Image @ 30 Hz
  - 380 KB RAM (framebuffer), 60 KB Flash



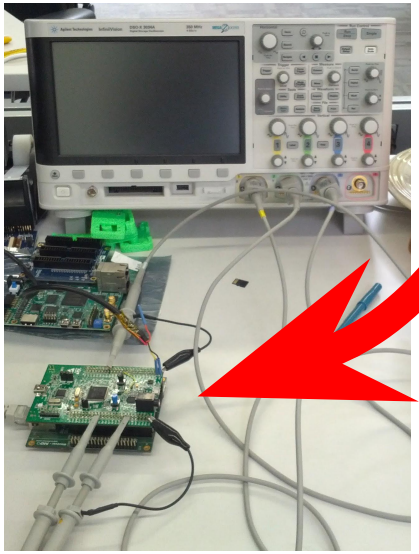
- Actuator (just an LED) on various boards:
  - subscribes to std_msgs::Bool

IMU Demo on STM32F4: Measuring Jitter

- STM32F4-Discovery stack: $55
- Slightly modified to use both Ethernet PHY and IMU
- Goal: measure FreeRTPS jitter



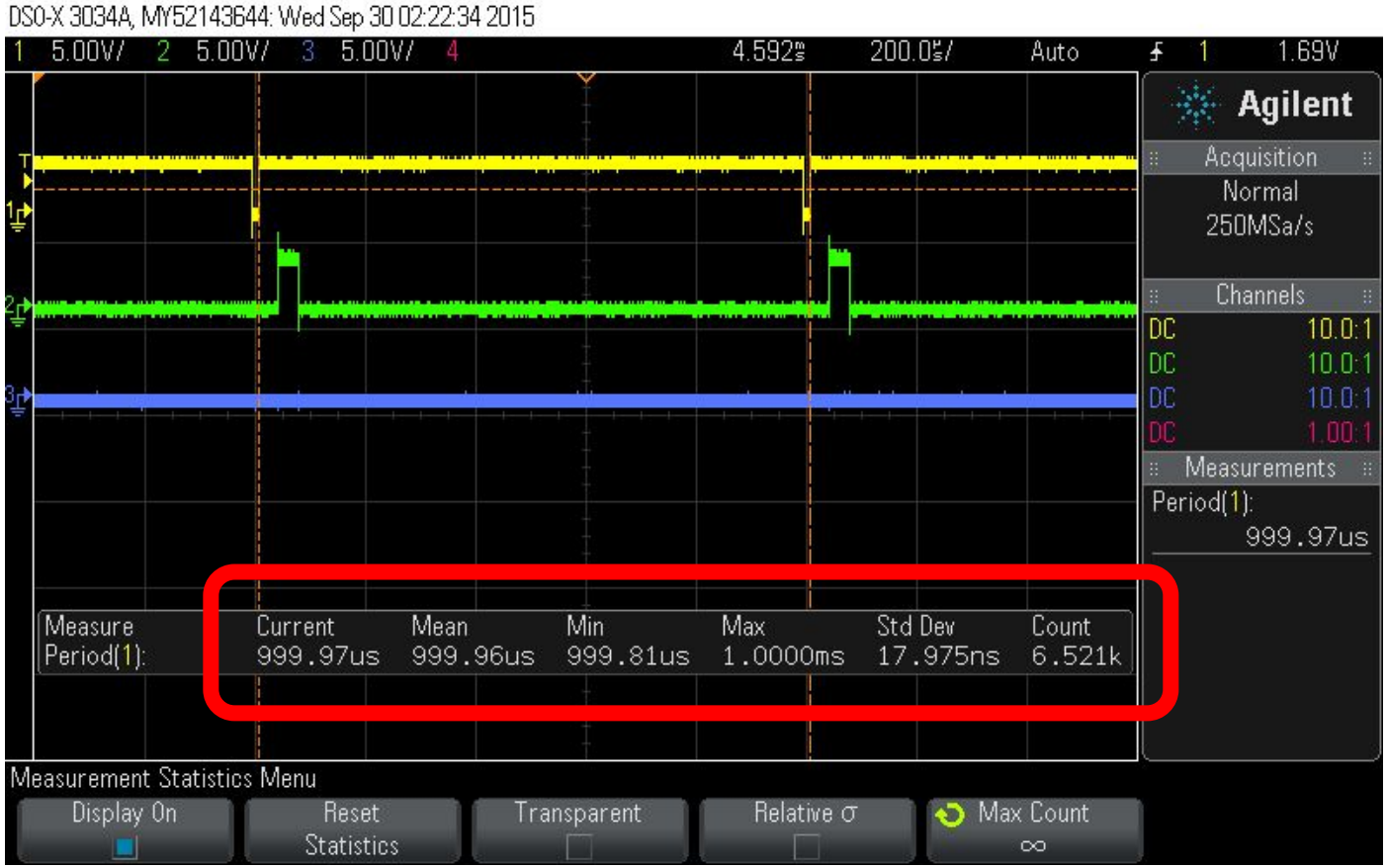
- Accelerometer CS and Ethernet TXEN signals to Agilent DSO-X 3034A



Performance Measurements: IMU demo

IMU CS

Ethernet TXEN

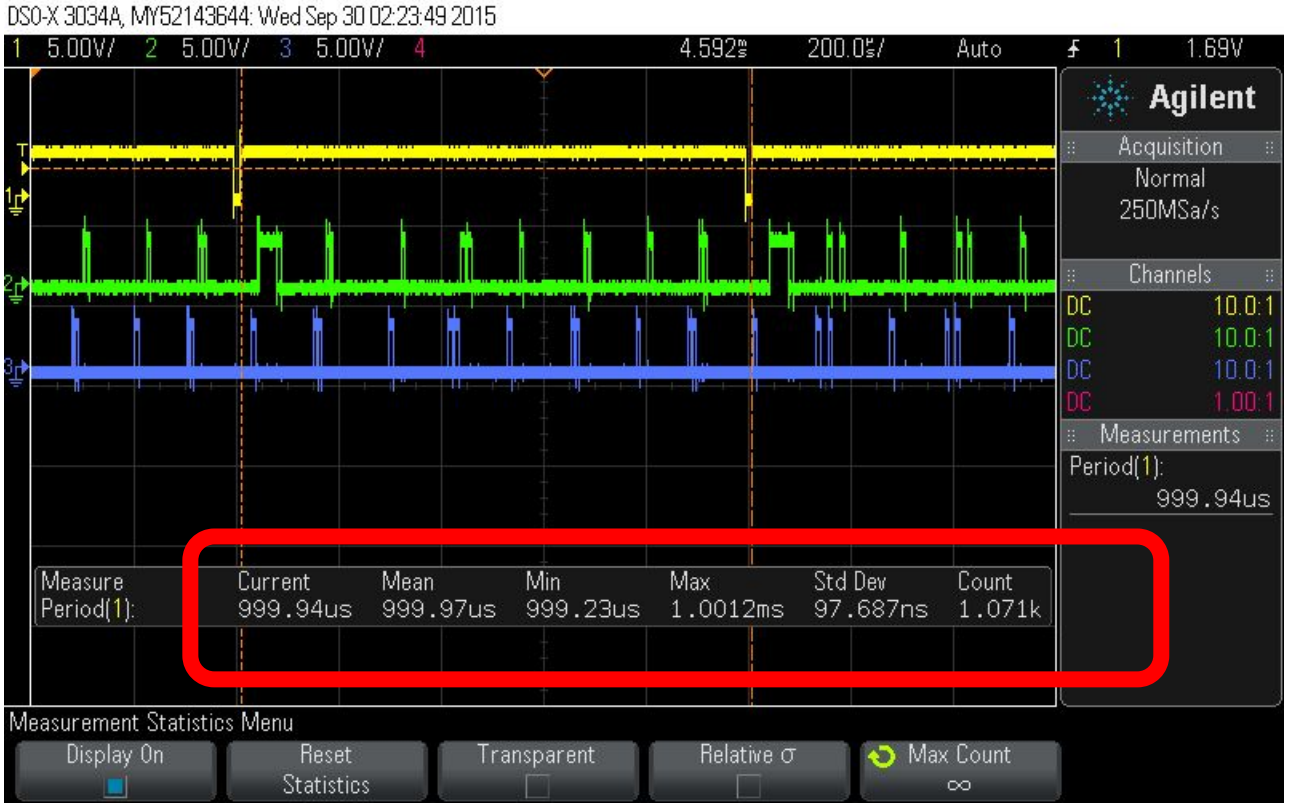


Performance Measurements: IMU demo

IMU CS

Ethernet TXEN

Ethernet RXDV



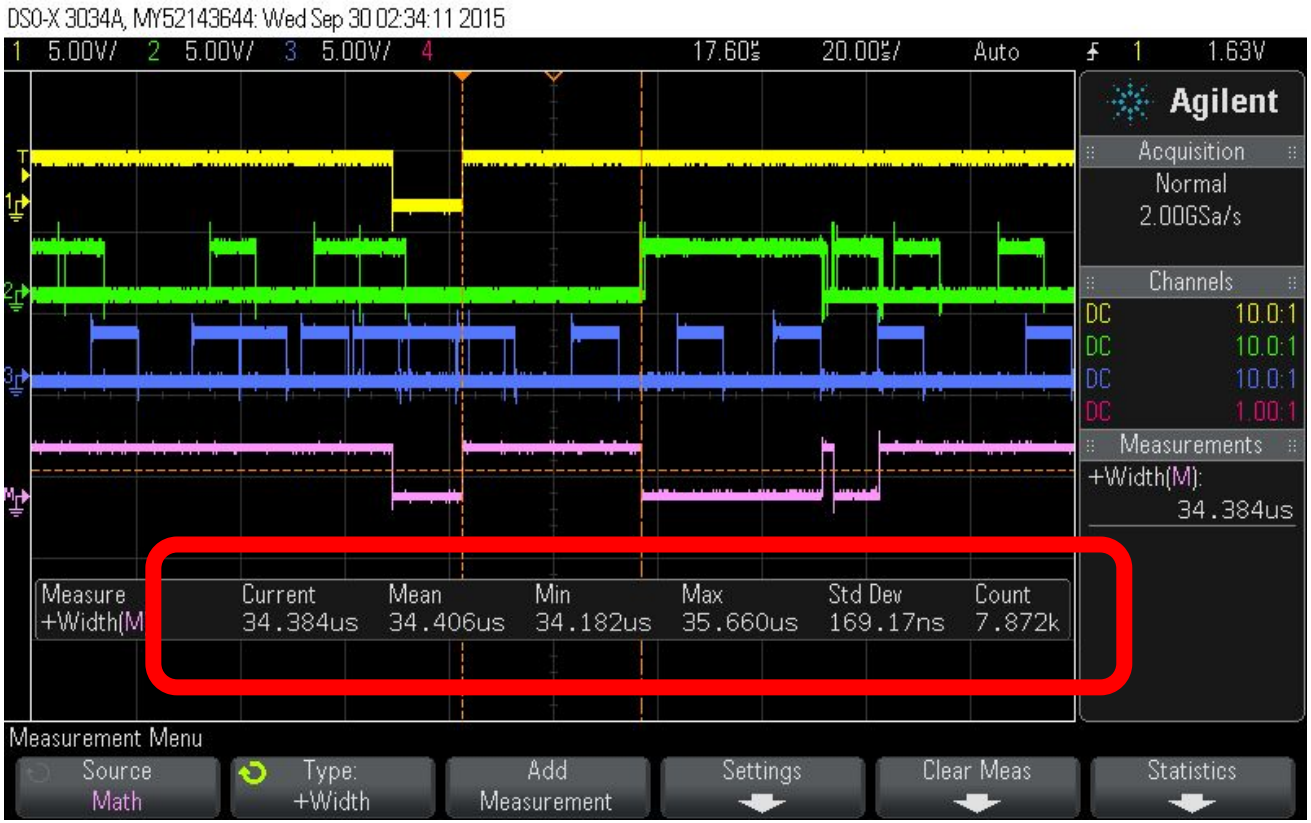
Performance Measurements: IMU demo

IMU CS

Ethernet TXEN

Ethernet RXDV

time calculation



Summary

ROS2 / DDS / RTPS is much more embedded-friendly than the ROS1 protocols

Future Work

- more MCU's, especially smaller ones!
- other physical layers (via standardized gateways), using abbreviated net/RTPS headers:
  - USB: FS, HS, SS
  - RS485, TTL UARTs of various bitrates
  - 802.15.4 and other wireless radios