“Old” RFCs, New Horizons

IoT in the age of SIP

Maksym Sobolyev <sobomax@sippysoft.com>

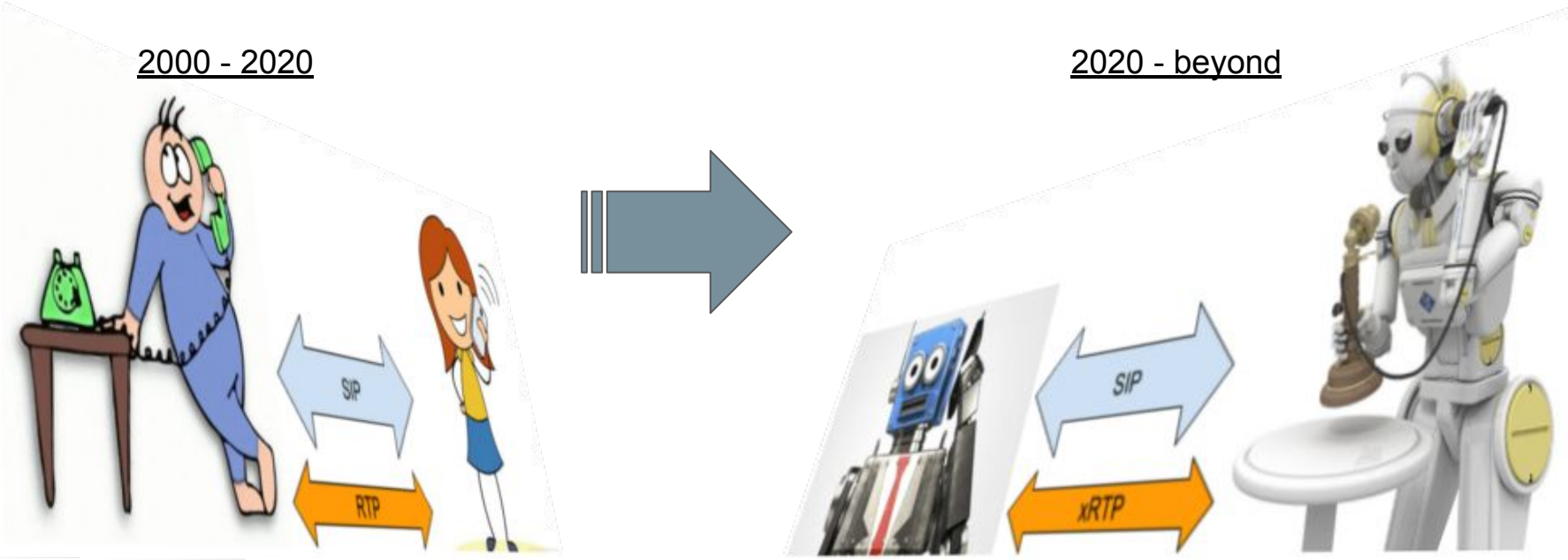
Who are we?

Sippy Sofware: Pushing OpenSIPS since dawn of time (aka SER 0.9, circa 2006) into the world along with Sippy B2BUA, RTPproxy, Asterisk and Yate for fun and profit.

Sippy Labs: Few open-source projects of our own: RTPproxy, Sippy Python B2BUA & SIP Stack, Sippy Go B2BUA and SIP Stack, RTP Cluster, Voiptests

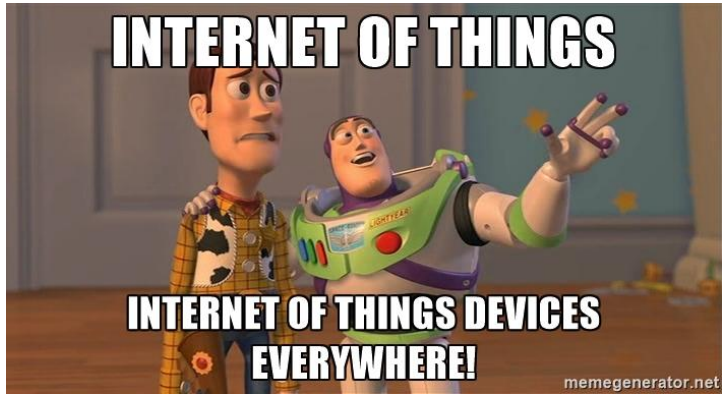
Outline

What this talk is about? The Future!



202x: Smart “Everything” on increasing scale

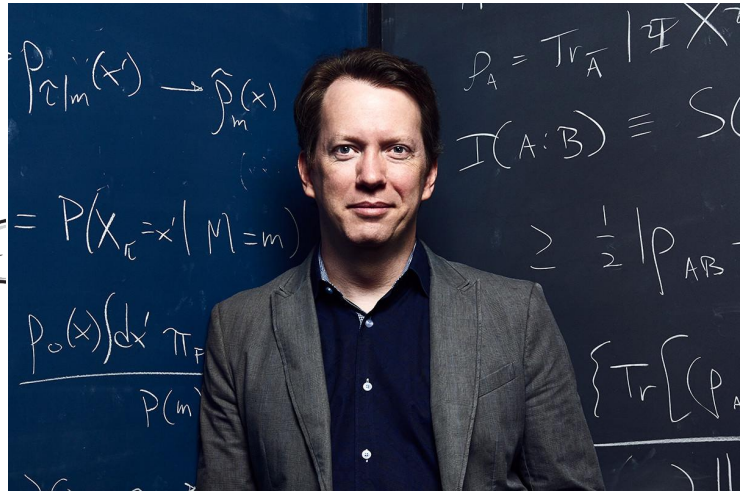
Homes
Businesses
Cars
Roads
Factories
Remote Areas
Cities
To infinity… and beyond
Countries (203x)?
Planet(s)?? #whoknows



Sensors Everywhere, literally many billions of them.

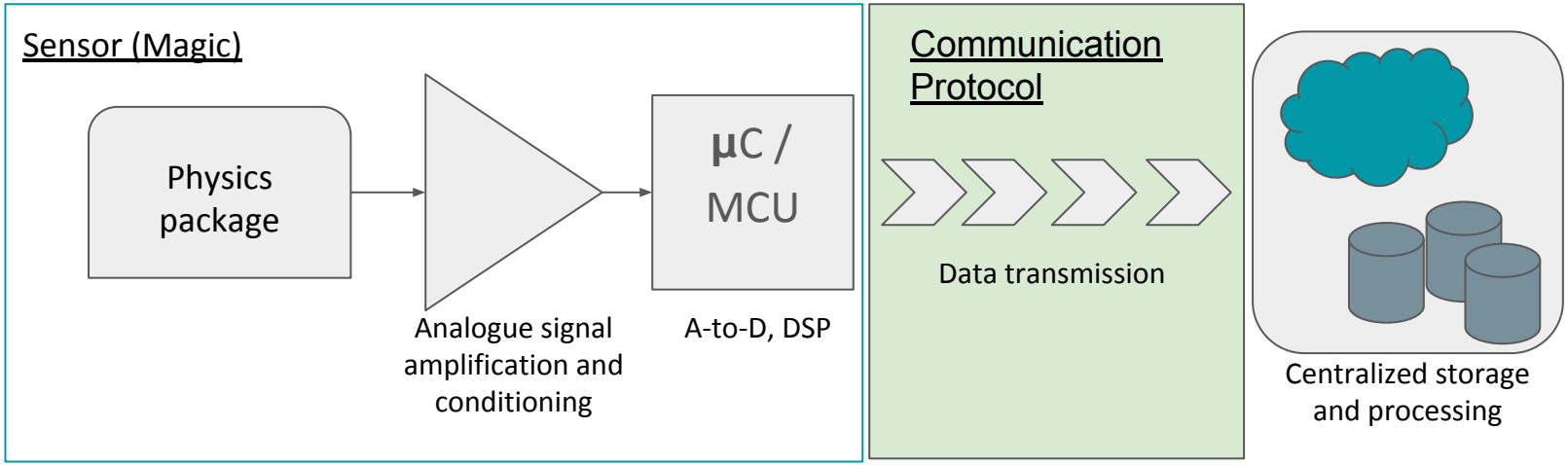
Collecting **real-time** data 24/7/365, feeding it back to the System

Sensors: Science or Magic?



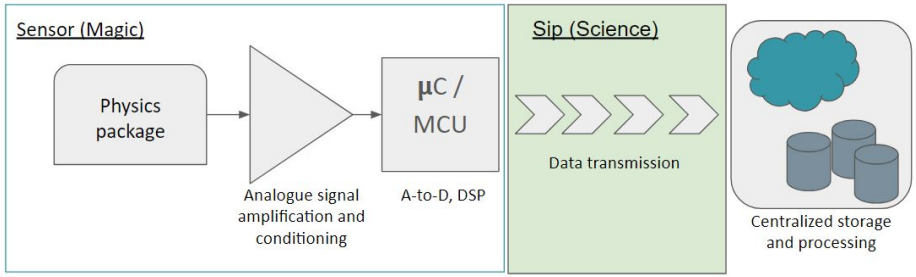
Sensors 101

Beepin’ sensors, how do they work?



SIP in a Nutshell: Quick Recap

- ➔ Endpoint Registration
- ➔ Endpoint Location
- ➔ RTP Session Setup
- ➔ RTP Session Parameters Negotiation
- ➔ RTP Session Parameters Update(s)
- ➔ Yada Yada (we are at OpenSIPS summit after all, you guys all know what the SIP/RTP is!)



SIP vs. HTTP

	HTTP	SIP + RTP
Supported network protocols	Stream-based (TCP, TLS)	Stream-based (TCP, TLS), datagram-based (UDP)
Model	Client-server	Endpoint-network-endpoint
Persistent Session(s)	No, transaction-based. Persistency is ad-hoc	Yes, both transaction and session support is built in
Registration and Location	No	Yes
PoC implementation complexity	Low, single thread linear, dozens of implementations both client and server	Medium, single thread FSM. Few implementations esp on low-end
Overall system implementation complexity	High	Low (relatively), OpenSIPS Cluster to help :)

Proof of Concept: Goals & Objectives

Focus on the software side

Use our own production SIP software (OpenSIPS included) to handle registration/location and mediate session between sensor (UAC) and data collector (UAS)

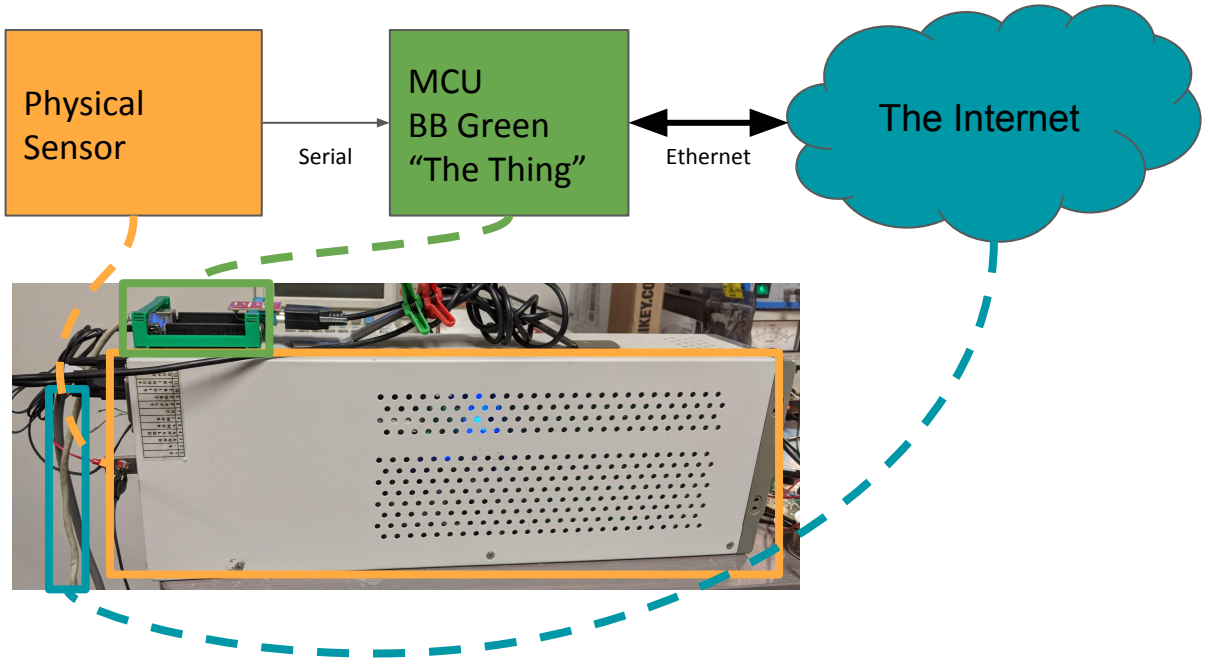
Use off-the-shelf IoT device to implement SIP part

Re-use existing code wherever it makes sense

Minimize number of external dependencies

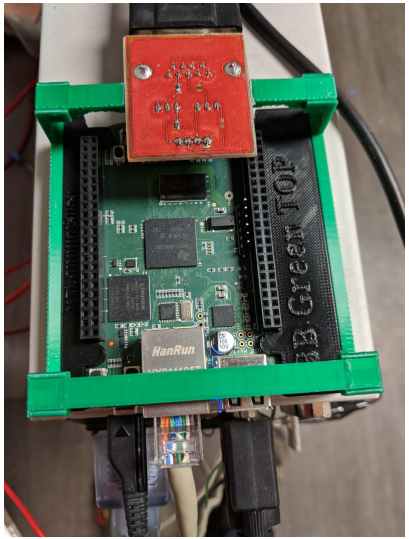
Functionality first, performance is secondary

Proof of Concept: Hardware Magic



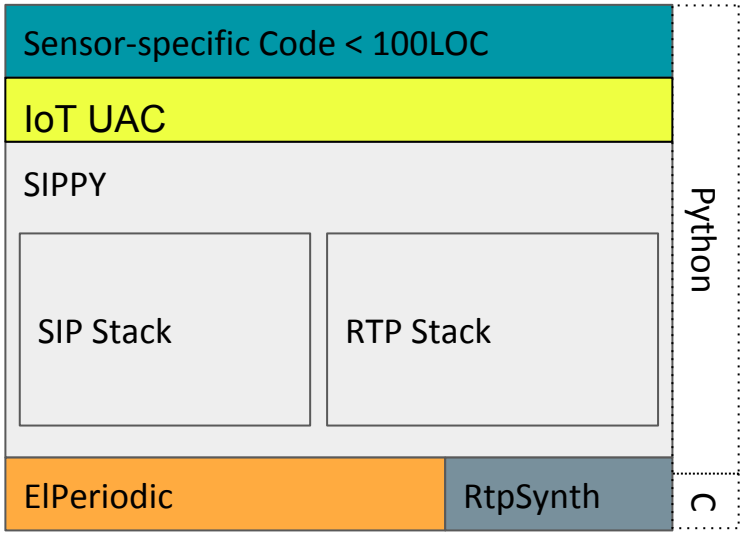
MCU

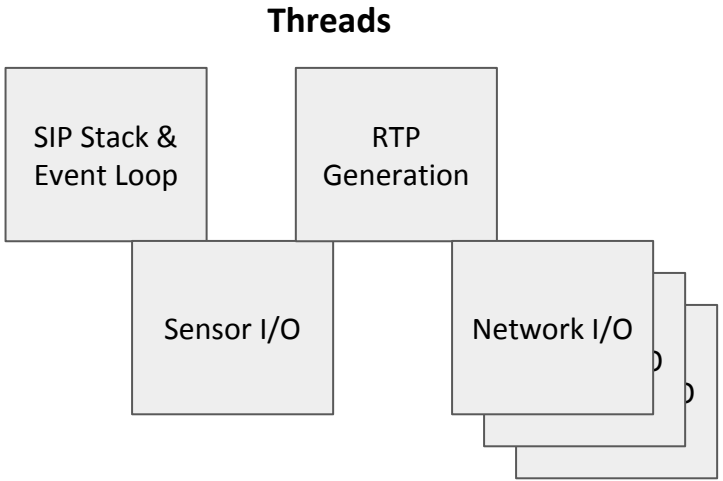
CPU: ARM Cortex-A8 1GHz
RAM: 512MB
MMC: 4GB
Power: ~1.75W (0.35A @ 5V)



Proof of Concept: Software

UAC Architecture





Results: Overall

Performance: ~10% CPU usage on target MCU, split between threads:

- Network I/O: 6.4%
- RTP Generation: 2.3%
- Sensor I/O: 0.5%
- SIP & Events: 0.8%

These only run when RTP session is established

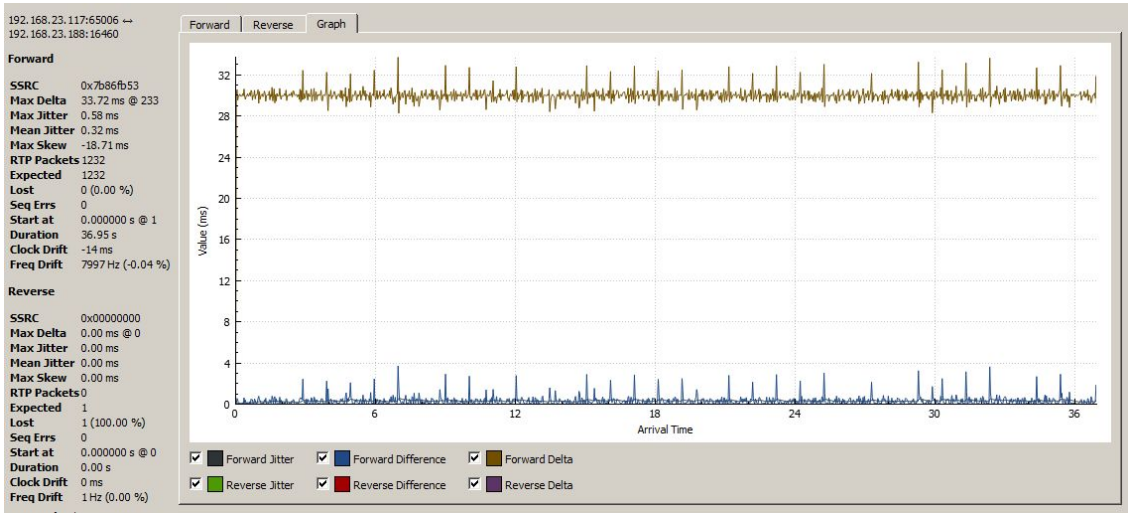
Memory usage: ~22MB

1 Man-Month worth of effort < 500 lines of new code

Fully functional prototype, very little device-specific code, 100% BSD licensed

Results: RTP Quality

Reasonable RTP timing achieved given 95% code being Python & general purpose OS with preemptive scheduler. Could further be improved by using elevated priority for critical threads. And more C code in key areas.



Conclusions

SIP/RTP might have application outside of its current major use case that could potentially outshadow its main use to date in just a handful of years.

Future Work

Standards

Real-world Deployments

Raising SIP awareness within “smart” communities

Small-scale SIP/RTP implementations optimized for memory / power / CPU consumption:

- µC-class, e.g. ESP32, low-end Atmel (aka Arduino) is probably too shallow
- FPGA / ASIC (cuz why not)?

Acknowledgements & Credits

Scientific Research and Experimental Development Tax Incentive Program, Government of Canada: for financially supporting this effort

Olle E Johansson <oej@edvina.net>: for allowing me to bounce some early version of these ideas off him during SipIT'16

Sippy Software, Inc. Staff: for helping out with presentation and PoC

ICP, "Miracles": https://youtu.be/8GyVx28R9-s for funny picture, music and inspiration

Sean Carroll, "Particle at the End of Universe": https://youtu.be/RwdY7Eqyguo?t=8m

Q&A

Anything you might have wanted to ask but were too shy to interrupt.

Contacts & Links

E-Mail: Maksym Sobolyev <sobomax@sippysoft.com>

GitHub: https://github.com/sippy

Patreon: https://www.patreon.com/sippylabs