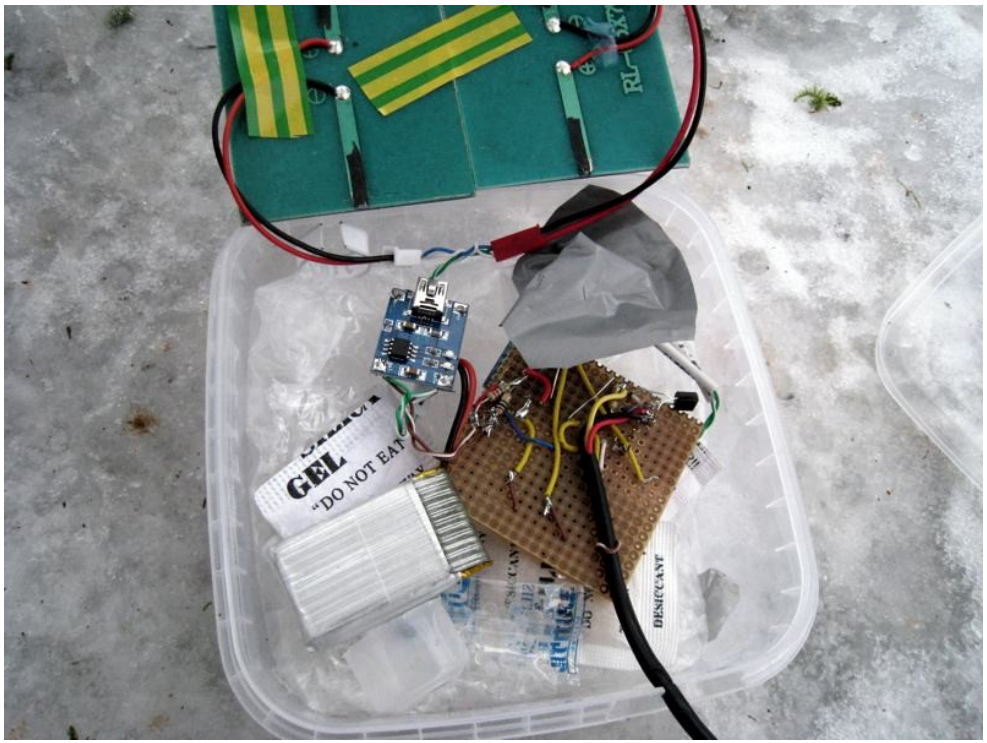
Part 1

Motivation (too much DIY hurts)



The uMesh project: Motivation (too much DIY hurts)

- Availability of cheap, single-chip sub-1GHz transceivers (Semtech, Silabs, Microchip)
- China made modules available for $3 (HopeRF)
- Consumer IoT devices
- IoT development in the age of startups and crowdfunding in amateur conditions
- IoT and radio-enabled devices are made/copied by common, unskilled people (by China too)
- Overcrowded unlicensed bands (SRD, ISM)

The uMesh project: Motivation (too much DIY hurts)

- IoT, IoM, Industry 4.0, Cloud buzzwords
- A solution to common problems becomes more and more centralized
- Dependency on service providers (data, computing power, storage)
- Loosing control of your own things
- Networks for IoT are usually simple in features
- Last-mile connectivity only (LTE-M, LoRa, SIGFOX)
- Usually there is no real security at all
- That's not how decentralized and distributed modern world should look like

- On the other hand - academic work on distributed networks (eg. WSN)
- Few implementations of these concepts exist - mesh networking (BT4-LE, ZigBee, a more general 802.15.4 MAC, B.A.T.M.A.N, OLSR, ...)
- Not very common (not cool enough)
- Many approaches and working protocols available (reactive, proactive, hybrid, data-centric, geographic, probabilistic, constraint-based, energy-consumption-aware, multipath, self-healing)
- Much research done in the field of AODV-based, reactive routing protocols
- Very well hidden in academic papers and no modern, easily usable and open implementation available

The uMesh project: Motivation (too much DIY hurts)

- Distributed nature breaks the dependency on service providers
- Networks can be community deployed, owned and used
- Usability in critical and disaster recovery applications
- Availability of a well maintained software allows higher transport security, code auditability (does your mobile service provider show you the internals of their baseband software?)
- No direct fees for data transmission (network node maintenance is needed)

The uMesh project: Motivation (too much DIY hurts)

- Professional IoT/IoM possible (higher redundancy, higher control)
- Amateur DIY possible too
- Commercial and service provider deployment possible
- Community deployment possible
- In case of emergency, a robust network coverage is readily available

Part 2

The Goal

The Goal is to Keep the Goal the Goal.
(https://www.bikejames.com/strength/the-goal-is-to-keep-the-goal-the-goal/)

The uMesh project: The Goal

- To design a set of **experimental** communication protocols and software
- ...to be run on many different hardware platforms
- ...for low data rate, hierarchical, self-healing, large scale wireless mesh networks
- With high level of security, proven crypto algorithms and protocols
- For community-built networks, amateur and DIY use, professional use
- Environmental monitoring, tracking, data acquisition, machine control
- Messaging, paging, voice communication
- Critical and disaster recovery applications in the future
- **Not** intended and suitable for consumer-grade IoT devices (no fridge-is-empty messaging)

Open-source and auditable software

- Community driven development
- Fast release cycle
- Quick response to issues, bugs, vulnerabilities
- Independent auditing and penetration/vulnerability testing
- Different licensing schemes (invasive - GPL, liberal - BSD) would allow the protocol stack to be used even in proprietary and closed applications

Open and well-documented hardware

- Open-source, documented and fully replicable hardware to be built by enthusiasts and amateurs
- Many common development platforms for DIY and learning
- Commercially available platforms for professional use
- More demanding software and protocols would require use of a more powerful hardware
- ...thus a higher level of capabilities and knowledge would be needed to build a compatible hardware platform
- No arduino, sry

HIgh level of security

- The software would be designed to withstand many common local attacks by unskilled and medium-skilled attackers (when a physical access is possible)
- ... to prevent remote attacks by skilled attackers and organizations (communication is secured with strong crypto)
- But it is still hard to prevent attacks by skilled attackers with physical access to the device
- ... in this case a special hardware (cryptograhical coprocessors, SAM modules) can help, but not 100%

HIgh level of security - how?

- Secure bootloader: firmware integrity checking, authentication
- MAC layer level security
- Routing protocol security
- Transport layer security
- Assymetric elliptic crypto, only proven algorithms allowed
- Stream and block symmetric encryption and authentication
- No fear to break legacy algorithms and rules
- Country and politics independent crypto algorithms with formal proofs (Ed25519, X25519, ChaCha20, Poly1305, 3DH elliptic based key exchange, axolotl ratchet key management)

Open, well documented, future-proof protocols

- Proactive mode for optimized latency
- Reactive mode for large-scale route discovery
- A hybrid mode can be used to tune between the two
- Mobile agent routing generalization (ant based routing)
- Ant colony optimization (allows self-healing, multipath, optimized routes)
- Hierarchical routing for easier target discovery
- Worlds and realms
- Geo-hints, direction-hints, realm-hints
- Route discovery based on constraints

Simple (CSMA/CA) or TDMA/FDMA slotted MAC

- Distributed time slot and channel/frequency allocation (SAS - slot allocation scheme, OFDMA-like)
- Distributed modulation and error-correction selection (MCS - message coding scheme)
- Distributed interference detection and avoidance
- Cognitive-radio ready
- Compatible with low-cost FSK transceivers (gaussian modes GFSK, GMSK, QPSK), PSK transceivers, EN 300 220
- Compatible with SDR transceivers
- Can be used over Ethernet, RS-485 or other similar buses

Part 3 What's done

The Framework

- RTOS kernel (FreeRTOS)
- C language, object-oriented (uh)
- Firmware modules provide services with interfaces
- Dependency injection of services
- Alternative service-locator design pattern
- Very loose coupling between modules
- Ready for microkernel architecture (services in userspace, including device drivers, IPC communication between services)
- MMU/MPU ready

Services and interfaces

- HAL (hardware abstraction layer) with interfaces
- libopencm3 library is used for low level hardware access
- Interfaces for Flash, Stream, Power device, Sensor device, Radio device, MAC, Filesystems
- SFFS flash filesystem
- Drivers for temperature sensors, voltage and current measurement (solar powered devices)

The Hardware

- Designed for ARM Cortex cores
- Tested on a development platform with STM32 Cortex-M4 microcontroller
- The most recent full implementation can run on STM32F401 with 64K of SRAM and 256K of flash memory, 84MHz clock
- HopeRF RFM69 modules tested

Command line interface

- The firmware can be easily controlled with a serial terminal emulator
- Interactive command line interface, command completion, history
- Tree-structured commands (hierarchical)

Protocols

- Simple CSMA MAC
- MAC layer encryption and data authentication
- MAC layer Golay FEC
- Neighbor discovery protocol
- Neighbor authentication protocol (3DH)
- L2 file transfer and stream protocol
- SIGMA authentication protocol tested (obsolete)
- Basic AODV protocol tested (obsolete)

A base for many other projects...

- Terminal graphing library
- Lineedit library
- TreeCLI library
- SFFS filesystem
- Refactored crypto libraries
- Circular logging library

Part 4 Possibilities

- A modular platform in development (2015 - ...), different form factors boards (80x100, 80x50, 80x25), stackable. SPI over M-LVDS communication
- Different modules can be built for data acquisition and machine control
- Solar/wind powered nodes
- Multicast support would allow efficient message broadcasting
- Strong security allows many uncommon applications

Testing with HopeRF RFM69 module (Semtech SX1231 transceiver)

- Testing with 0dBm EIRP on 433MHz SRD/ISM band (to comply with EN 300 220), quarter-wave whip antennas
- 100kbps GMSK (BT=0.5)over-the-air data rate easily achievable for distances up to ~1km line of sight, FEC required to overcome single bit errors
- 50kbps achieved with GMSK, Golay FEC, 300m in sparse vegetation, neighbors authenticate under 5s
- UKHAS tests show ~100km ranges (narrowband, very low data rates)

Broad range of compatible radio transceivers

- Texas Instrumets: CC1101, CC1110, CC..., CC...
- Microchip: MRF89
- STM: Spirit1
- SiLabs: Si4455, **Si4460/1/3/4**, Si4432, Si4467, Si4438
- On Semiconductor: AX5043, **AX5243 (PSK!)**
- Maxim: MAX7030
- Analog Devices: ADF7024
- Semtech: SX1211, XE1205, **SX1231**, LoRa transceivers
- SDR: Lime Microsystems (LMS6002D, LMS7002M)