JeeBoot

Jean-Claude Wippler
jeelabs.org

Re-flashing an ATmega

Until now:

JeeNode is like Arduino: USB serial upload

Wireless nodes don't have USB (cost)

FTDI-serial adapter, 6-pin "FTDI" header

Remote nodes

Don’t bring the JeeNode to the PC, dummy!

Bring the PC to the JeeNode? Nah...

Upload to the JeeNode over the air!

Based on a central “boot server node”

Demo

Fake server: JeeLink with fast & slow blinks

Boot loaders are a bit tricky to debug

No “normal” context, no Arduino libs

Test as “sketch” first, without re-flash calls

Current code fits in less than 2 KByte

Let me show it to you ...

Over-the-air uploads

Upload can be up to a few dozen kilobytes

No way to store and **then** re-flash

Once uploading starts, the node is unusable

Need to retry forever, there is no fallback

What if the "boot server" node is off?

Saving power

Can't just retry all the time

Exponential back-off, down to one / min

Remember those ULP scenario's #1 .. #3 ?

Turn the logic around:

Remote nodes drive the entire process

The boot server can be state-less

Security

Wicked neighbour could re-flash **my** nodes!

Power-cycling won't trigger an upload check

Node only checks for new upload on reset

You need to be physically near the node

Or: current code could start a boot-check

Future option: encrypted data, replay-safe

Other concerns

No serial upload fallback - it **has** to work

Installed base: change boot loader in the field

... and a way to revert to the standard one

If everything happens remotely,

how do you identify a particular node?

What's the new workflow - is it convenient?

Where to go from here

Implement a decent boot server node

... could be JeeLink, or Linux (Flukso)

Test all the possible failure modes

Add some basic power optimisations

Figure out a **convenient** upload workflow

Suggestions?