Managed Node Interface
or
How to Automate Quanto
• WiFi SSID: HOME

• Server IP: 192.168.1.103

• Login: userK; password: tutorialK; where k ∈ [1:7]

• If you have Linux/Mac OS X, forward your X11 connection:
  ssh -XYC userK@192.168.1.103

• Application files can be found in ~/public_html/

• You can access your files using your browser at
  http://192.168.1.103/~userK/
MNI

- Goal: Easy to use tools to manage a lot of nodes
- Written in Python
- Hooks for compilation and installation of code on a node.
- Contains special extensions for Quanto
MNI Structure

• Two main objects:
  • MNI: represents the whole network
  • Node: represents a node
Quanto Extensions

- QuantoTestbedMote
  - install: executes the install command
  - push_usr: presses the usr button on the node
  - release_usr: releases the usr button
  - reset: resets the node
  - start/stop: starts, stops a node
  - programming_mode/serial_mode: puts node into programming/serial mode
  - many others...
• QuantoMNI
• reset_all
• stop_all
• press_usr_all
• calibrate_all
• parse_quanto_log_all
• process_quanto_log_all
• get_energy_per_quanto_state_all
Useful Scripts

• In general, you won’t have to use the python modules directly
• many useful scripts that automate most of the tasks
• start.py, stop.py, reset.py, press_usr.py
• install.py, programming.py, serialmode.py
• energy_measurement.py
Configuring MNI

- A simple configuration script describes the network architecture (config.ini)

```
[Nodes]
numNodes: 2
type: QuantoTestbedMote
makeCmd: make epic quanto

[Node1] id: 1
ip: 192.168.1.200
serial: /dev/ttyqa00
installCmd: make epic quanto digi reinstall,1 bsl,/dev/ttyqa00

[Node2] id: 2
ip: 192.168.1.201
serial: /dev/ttyqb00
installCmd: make epic quanto digi reinstall,2 bsl,/dev/ttyqb00
```
Automating Installation

- install.py uses makeCmd and installCmd to compile and install a binary on a node
- Reports if install on some of the nodes failed
Calibrating Nodes

3.0V

io from epic
How Does Calibration Work?

• Cycle through each calibration resistor

• Measure the number of iCounts during $T$ for each resistor $\Rightarrow f$

• We know $V$, and $R$ for each step

• Thus, $E = V^2 R / f$ Joules per iCount

• At runtime, find the closest switching frequency, and interpolate to get the right energy values
Automating Calibration

• Every QuantoTestbedMote needs to be individually calibrated
• Calibration application can be found in apps/quantoApps/CalibrateQuanto
• Circulates through all the calibration registers, recording the iCount values for each of them
• Stores calibration in a calibration.ini file
Energy Metering

- energy_measurement.py
- Opens a serial connection to each node
- Listens for Quanto Log Messages
- Stores the messages in individual file
- After a limit (number of messages received), automatically decompresses, processes, and parses the log file
- Solves the equation system, depending on installed tools
Linear System vs. Optimization

- The LS can have negative solutions
- Better choice: Optimization Problem
- Constrain the individual powers to be positive
- cvxmod (for python) can easily solve such a system

```python
A = cvxmod.matrix(W*X)
b = cvxmod.matrix(W*Y.T)
x = cvxmod.optvar('x', cvxmod.size(A)[1])
p = cvxmod.problem(cvxmod.minimize(norm2(A*x - b)), [x >= 0])
p.solve()
x = x.value
```
Now What?

• Compile and install plain Blink
• Instrument Blink with Quanto
  • Compile, install, collect data
  • Run the Quanto tools
• Instrument RadioCountToLeds with Quanto
  • Each one picks a different period
  • See the impact of every other node in your energy expenditure
Equipping FTSP

• One application, one library
  • TestFTSP
  • FTSP lib

• TestFTSP receives beacon, timestamps it, and broadcasts the result

• FTSP lib runs FTSP in the background
FTSP

• Two main entry points
  • receive timestamp from neighbor
  • send out timestamp

• Your job:
  • Equip TestFTSP and FTSP lib with Quanto
  • receiving a timestamp should be colored by neighbor color
  • send out timestamp should be colored by node color
Now What?

- **Get TestFtsp:**
  ```
  cd ~/public_html
  cp -r /opt/tinyos-2.x/apps/tests/TestFtsp/ .
  ```

- **Get the FTSP library**
  ```
  cd TestFtsp
  cp /opt/tinyos-2.x/tos/lib/ftsp/* .
  ```

- **Edit Makefile and remove**
  ```
  -I$(TOSDIR)/lib/ftsp
  ```