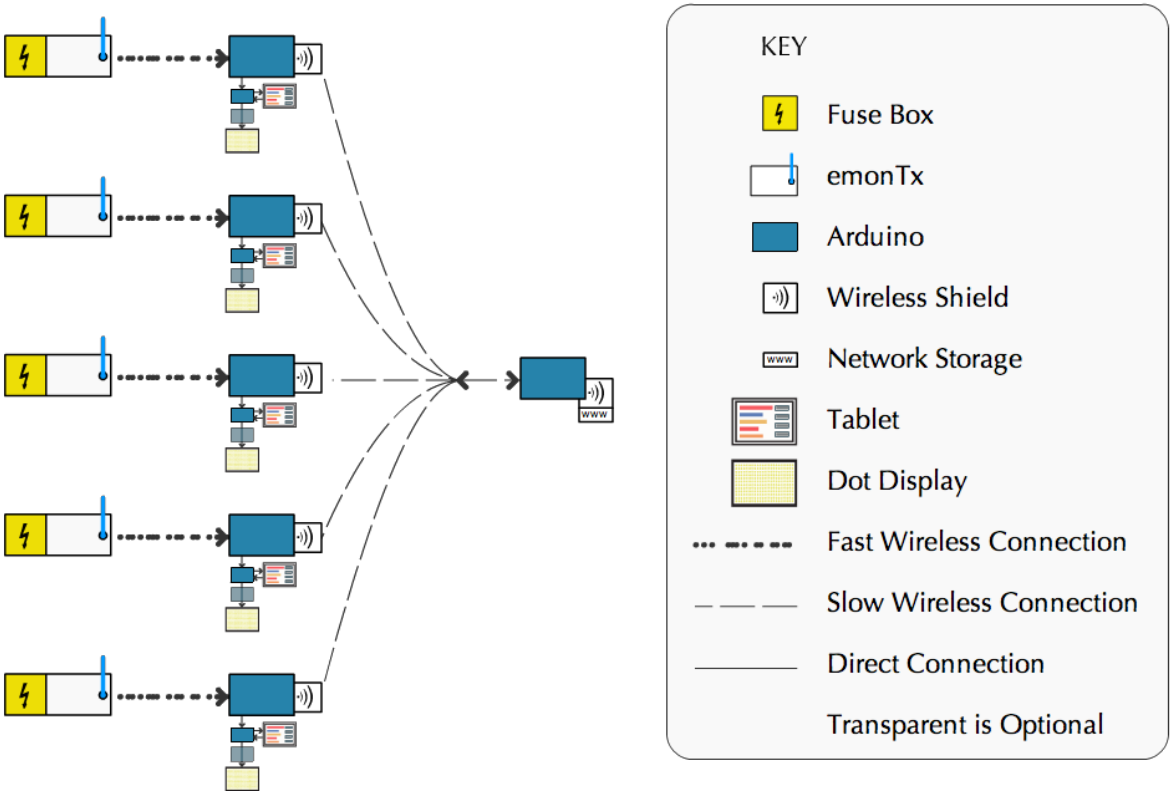
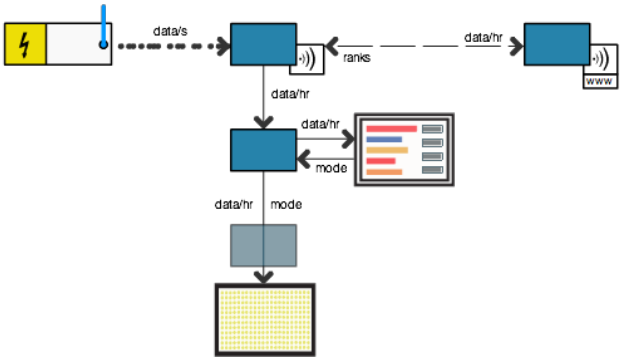
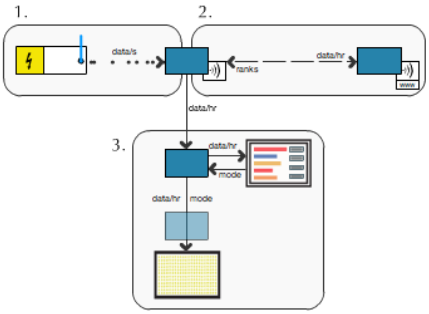
Network Diagram



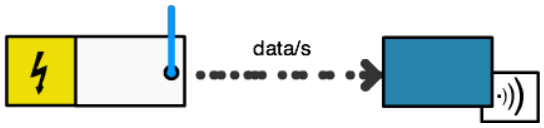
Node Diagram





- Throughout these components:
  - Time must be synchronoised, at least relatively..
  - TCP-like protocal communication.

Node 1. emonTx to base

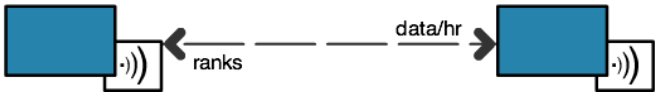


Pseudocode:

- emonTx
  - sends data (Power) once every few seconds

- Uniform Storage Mechanism - Pseudocode. Occuring at the Base.
  - on initiation:
    - int[24] avgHourValues = {-1,-1,…}
    - int[7] avgDayValues = {-1,-1,…}
    - int[15] avgWeekValues = {-1,-1,…} , ~15 weeks - duration of test
  - recieves data.
    - data is an integer value representing Power. It is sent every few seconds.
    - adds data to accumulatedHourValue
    - adds 1 count to accumulatedHourSignals
  - ———————————————————————————————————————
  - *(exists on the base)*
  - *at every hour:
    - calculate avgHourValue
    - store avgHourValue to int[currentHour] avgHourValues
    - add value to accumulatedDayValue
    - add 1 count to accumulatedDaySignals
    - **<< base: output data >>**
  - ———————————————————————————————————————
  - *(exists on tablet, on storage-medium)*
  - *at every day:
    - calculate averageDayValue
    - store avgDayValue to int[currentDay] avgDayValues
    - add value to accumulateWeekValues
    - add 1 count to accumulatedWeekSignals
  - at every week:
    - calculate averageWeekValue
    - store avgWeekValue to int[currentWeek] avgWeekValues
  - *Notes:*
    - *(data can also be initially accumulated as minuteValues)*
    - *(52 weeks is a bounding state in this code.)*

Node 2. base to parent



Asynchronous Communication Protocals. (Uniform)

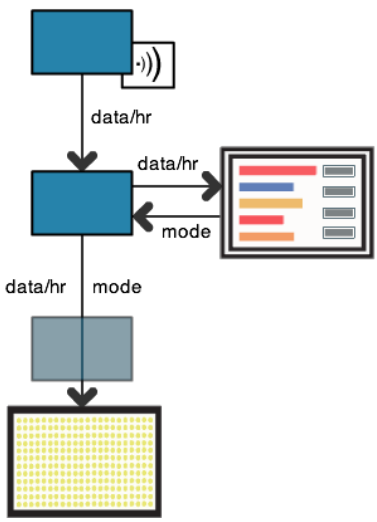
- Synchronized clocks.
    - can be ensured by maintaining a single clock at the parent, which stores the data, and pushing that out to it's base nodes along with any other messages.
    - clocks can be relatively synchronised and not absolute.
- Communication Protocols.
    - Will have to logistically measure the stability and speed at which a structure can switch between sending and recieving modes.
    - TCP-Like in nature, with checks that a package has been recieved. (e.g. If A sends a message to B, B pings back that a message has been recieved, if B doesn't ping back, A tries sending the message again at a random time within allocated time ranges.)
    - Maintain queues (may have to be pre-scheduled, e.g. listen every half a minute), so communication is asynchronous and signals will not be lost.

- Base sends:
    - unique id, accumulatedHourValue, check digit at the end (full stop).
- Parent sends back:
    - ping, (time), that messaged is received.
    - hourly rank - every hour.
    - daily rank, total rank - every day.
    - weekly rank - every week.
    - *(The parent could potentially push stored hourly,daily,weekly data once a day, disabling any input from the user at that time (e.g. 3am)).*

- Data Storage Mechanism - Parent:
    - Data is uploaded to the server, which calculates resulting data.
    - The hourly data is treated independently, but in the same manner as the Uniform Storage Mechanism, following every hour*.

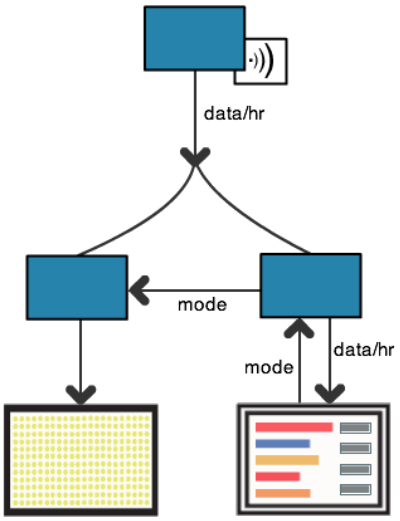
Node 3. base to interface and display

3a.



OR

3b.



Description.

- **3a.** The base sends data to an arduino that is talking back and forth with an Android Tablet. This arduino forwards data to a flip dot display which may or may not require an independent arduino to control. The below text assumes 3a.
- 3b. Same as 3a, but definitely requires an additional arduino for the display, and it receives two independent data streams dictating mode and data, as opposed to one.

Communication between base and interface:

- Same Asynchronous Communication Protocals. (above)
    - *(synchronised clocks, queuing, TCP-like)*

Interface Code:

- The ADK method will be used above Bluetooth and ADB methodologies.

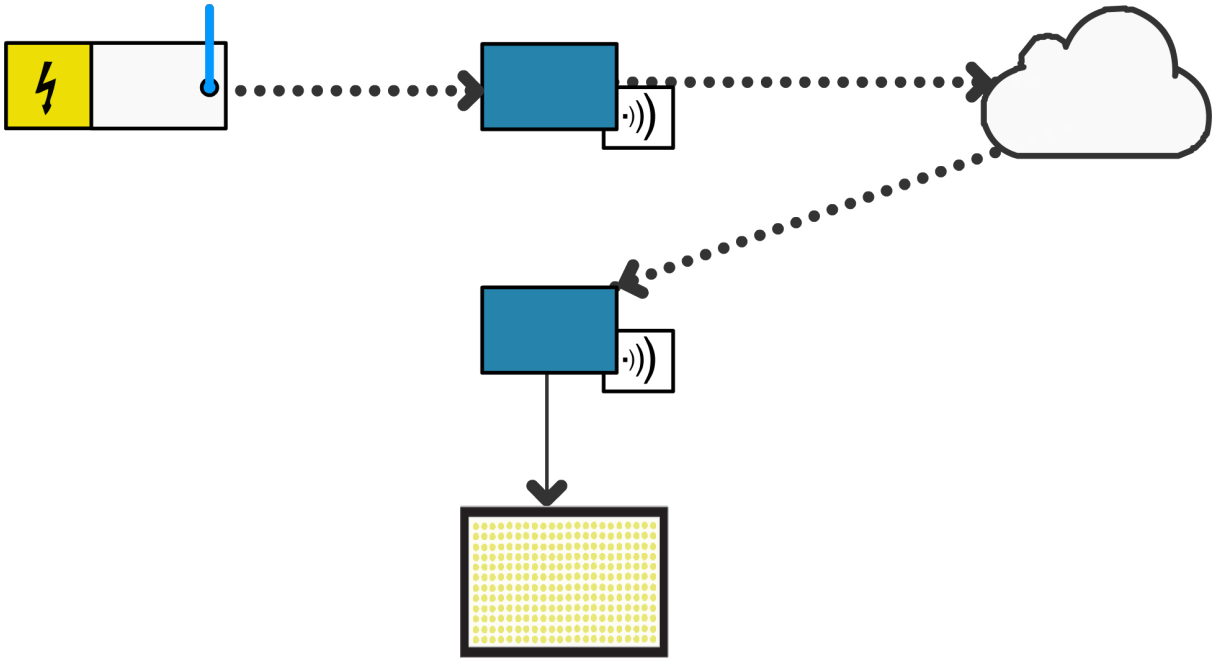
Data Storage Mechanism - Interface, Dot Display:

- The hourly data is received and treated independent of other devices, but in the same manner as the Uniform Storage Mechanism, following every hour*.

New Notes:

- For online data store, GAE (Google App Engine) is sufficient.

Current System



(The two arduinos could potentially be combined further on, but for now this is a good way of thinking of them.)