Limux

"Soda" here refers to anything in a form factor similar to the 0.5l PET bottle, glass bottles are not an impossibility (Club Mate comes in such) but we'd like to avoid those due to much higher likelyhood of breakage.

A soda bottle vending machine we got from Nixu. "does not cool" was the diagnosis, Rambo&Dist fixed this problem (the fan that was supposed to cool down the hot-side of heat exchanger was completely stuck and the fan that was supposed to spread cool air on the cool-side was making horrible noises. Both are now fixed).

Log at end of file, last entry @ 2014-08-10 20:53.

Next steps:

1. Reverse-engineer all the internal wiring (which is 220V relay logic)

This requires two to three people, two that can be trusted to not get themselves or anyone else hurt with the 220V and if there is a third one his job is to write things down as the two make measurements.

My (rambo) current plan is to put **everything** to a graph database (like Neo4j) as nodes (as in each individual switch or relay contact gets a node, in addition to the switch/relay itself) and then do shortest path analysis and whatnot to figure out the schema. This plan might sound stupid to you, and it might be but I have a gut feeling that trying to draw the schema as we go is not going to work...

Naturally everything needs to be photographed and labelled as we go along regardles of any other intermediate documentation methods to attain the goal of proper schema.

Easy version (done): treat it as black box, here is how it is controlled: https://gist.github.com/tanelikaivola/1753dd5f55003650c98c The "Aktivoi" relay needs to be pulsed first, then we can pulse the relay for the correct lane, the pulses don't need to be long, exact lenght requirement is not known (the lane timings etc are handled mechanically) with cam wheels and switches.

2. Figure out whether to replace some parts of it (like all the sensing switches...) with something safer (though since some of it **has** to stay 220V [the motors for example] it might be better to keep any false notions of safety far away).

This can be though about **after** we have a schema

3. Make software/hardware driver that only cares about the state of the machine

Parts:

- Barcode reader
- Embedded linux machine (Beagleboard or something)
- Relay board
    - There's a huge one in there already, a smaller one will do just fine and will be simpler to use
        - We destroyed the huge one, replaced with easier-to-use DX ones
- optoisolated taps to the lane bottle-detect switches
    - probably we want a small MCU doing the debouncing on these since we're not going to be running a proper realtime kernel.

- optical gate to detect when bottle has actually been dispensed
- OPTIONAL: Temperature sensor
- OPTIONAL: bypass the internal thermostat for better temperature control
- OPTIONAL: water sensor (to detect broken bottles earlier than next refill)

Quick specs

- Keeps track of stacks of barcodes (==bottles) in each lane
- Can be queried for the status (returns stacks of barcodes for each lane)
- Can be instructed to release a bottle on lane X
- Keeps track of events (vend/refill) for historical analysis by barcode.
    - Can be queried for slices of history by time
- Use special barcodes (printed on the machine itself) for entering/exiting refill mode so there no need for use of a separate computer (or keyboard+display) when refilling.
- Automatic backup of the history database to somewhere safe
- RESTful interface (with strong[ish] authentication)

Keep is small and simple, make it robust. Crush feature creep mercilessly.

4. Make software that can be used to buy soda

This is a different piece of software, probably running on a completely different piece of embedded linux HW.

- Keep track of everyones "bar tab"
- Show what's up next on each lane (Map barcodes to product names)
- Show general fill status
- Show fill status by product name
- Show product popularity curves (how quickly brand X sells after refill etc).
- OPTIONAL: Take requests, make polls to measure request popularity

Log entry @ 2014-08-10 20:53:

Last night soldered level shifters to breakout board (https://github.com/HelsinkiHacklab/ledmatrix/blob/master/max33xxE_breakout_sch.pdf) and connected them to a BeagleBoard (c4). We can now control the relays from the BB, a potential problem is that the lines are pulled on boot before we can assign them to outputs, so at least the activation relay control needs to be inverted.

- rambo

Log entry @ 2014-07-07 02:21:

Relays documented in falstad-format: https://gist.github.com/tanelikaivola/1753dd5f55003650c98c

Old harder to drive relay board was malfunctioning. Relay board isnow replaced with two other relay boards (4x relays each). 5 relays are used.

One is for activation (money inserted, all lanes unlocked), 4 are for each lane (trigger for activating a motor

in a lane, switches keep the motor spinning until it completes half a rotation).

- dist & rambo