Restroom Monitor Mark II

Have you ever found yourself in need of answering nature's secondary call, walking across the office to heed it, only to find that all the stalls are in use? <sarcasm>Being situated at the far end of the office, this was a very serious issue</sarcasm>—or at least I could pretend it was to give myself enough of an excuse to do something about it. If only there was a way to know, before ever leaving your desk, if your call would be able to be answered in peace or not.

The concept is simple—determine the state of a restroom and put it somewhere that it can be checked before heading over. A similar system was installed many years ago using wireless, battery-powered magnetic switches updating a website—but the very obvious boxes were vandalized and required frequent maintenance. My objective was to make something completely invisible (and that wouldn't make people uncomfortable—as restroom tracking could easily do) and impervious to all but the most malicious sabotage—while providing a seamless way to check the state.

The restrooms in question are private rooms (not just stalls), with their own door and deadbolt—the door is always closed, and the occupancy is determined by the position of the deadbolt, which also has a red/green flag on the front of the door. One of the main concerns about this project was doing it in such a way that it wouldn't make anyone uncomfortable—which would rapidly kill the project—invasion of privacy lawsuits can be expensive. A wide variety of methods of determining state were considered and discarded:

- Motion sensor—too much like a camera and bad for long visits, tough to get a definitive state
- Infrared sensor—too much like a camera and visible
- Red/green color sensor looking at flag—too much like a camera
- Magnetic switch or door hinge rotation sensor—can't tell if the door is locked or not
- Deadbolt induction sensor—too fragile

- Switch connected to a Bluetooth dongle to communicate—sitting inside a metal door frame could have connectivity issues and the battery would have to be replaced

I eventually settled on a deadbolt switch designed specifically for commercial installation, wired through the door frame to above the dropped ceiling (isn't drilling holes in the office walls fun?). The switch sits inside the deadbolt pocket (so is not visible), is designed for industrial usage (so won't break with repeated use), and is wired so that connectivity is perfect and there are no batteries (so requires no maintenance). The switch I used was this deadbolt pocket switch.

See how the pocket looks blocked by a silver panel? That's the flipper of the switch.

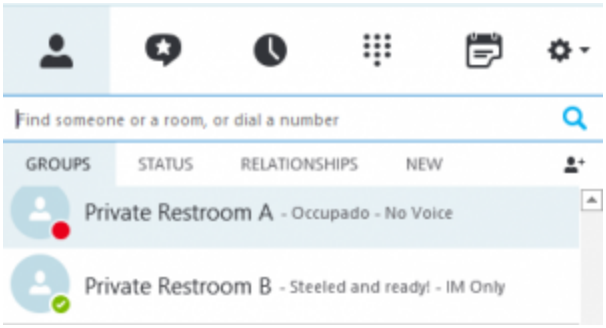
Once you have a way to determine the state of the restroom, the next step is to be able to read the state and send it somewhere. After working with a couple different microcontrollers, I decided to use the Spark Core because of the on-board WiFi and extremely easy development/deployment process. After working with it, I could not recommend it highly enough. After using the phone application to connect it to wifi and tie it to your account, you update the microcontroller by coding the application on their web IDE, then pushing the automatically verified and compiled code to the device over the public internet. It's one step short of pure magic—and a drastic and welcome change from the microcontrollers I've worked with in the past. All that aside, it's a simple task to have the Core

receive input from the switch when it changes, then POST notifications when it receives a changed state from the switch over WiFi to a listening service endpoint. The microcontroller is wired into power from a standard mains to USB power supply—again, removing any dependency on battery maintenance.

I did experiment with using USB battery packs to see what kind of battery life I could get—and ran into an interesting behavior. The battery packs that automatically turn on do so by monitoring the current dropped across the power pins. They also automatically turn off when too low a draw is detected—assuming that nothing is actually using the power. To conserve power draw, I disabled the WiFi on the microcontroller when not actively transmitting a changed switch state. While this did save energy, it also made the power draw low enough (<10 mA) that the battery pack automatically turned itself off thinking that nothing was plugged in. To get around this, I wired up a transistor circuit to put a 50 millisecond draw across the power pins (through a resistor) every 7 seconds (suggested by this article). This was effective in keeping the device on—but the biggest battery pack I could find (20,000 mAh) only lasted about a week. In the interest of a platform requiring zero maintenance, I instead decided to hook up a USB wall wart power supply and wire that in rather than relying on battery power.

The two grey wires go through the wall, down the frame, and to the two door switches (via Molex connectors for easier maintenance), the lower black cable is power from a USB wall wart, and the upper black cable goes to the indicator lights—more on that later)

On the other side of the POST request, I have a Windows Service running on a server, which is self-hosting both an HTTP endpoint for the microcontroller to call with switch state changes, as well as a Skype for Business platform and user endpoints representing each restroom. Since everyone at Clarity is on our internal IM client all day (Lync/Skype for Business), it's logical that we'd look there for the state of the restrooms. Since Skype for Business endpoints already have a presence state associated with them that shows red/green, it's an absolutely perfect fit to have endpoints for each restroom that can be Available or Busy, according to the status of the switch on the physical room. Welcome to the internet of things (or places)!

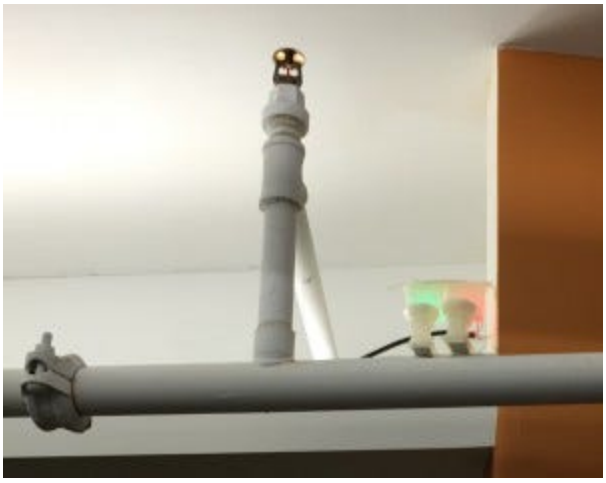


Don't ask how a restroom can be 'Steeled'. You don't want to know.

So that's all nice and dandy! Indicators on our computers getting the state of the restroom—that seems good enough. Yea, I wasn't happy with "good enough" either—it just wasn't quite over-the-top enough yet. Clearly, more was needed.

I printed 3D models of toilets (can I just say how much I love the previous 6 words?) on the office 3D printer (MakerBot 2) using clear plastic filament, and embedded LED lights into the back of them (hot glue to the rescue). Since the microcontroller already knows the state of the switches and is able to put out a very convenient 5 volt current that can drive the LEDs, it was a simple task to wire up the translucent toilets to lit LEDs indicating their respective state—functioning as remote physical indicators for the rooms that could be glanced at before heading down the hallway to the doors themselves. As a side note, I used an Ethernet cable to go from the microcontroller to the RGB LEDs—3 power sinks per light plus one shared voltage source needed 7 wires to run to the models, and Ethernet cables are a very convenient 8 strands, and are easily

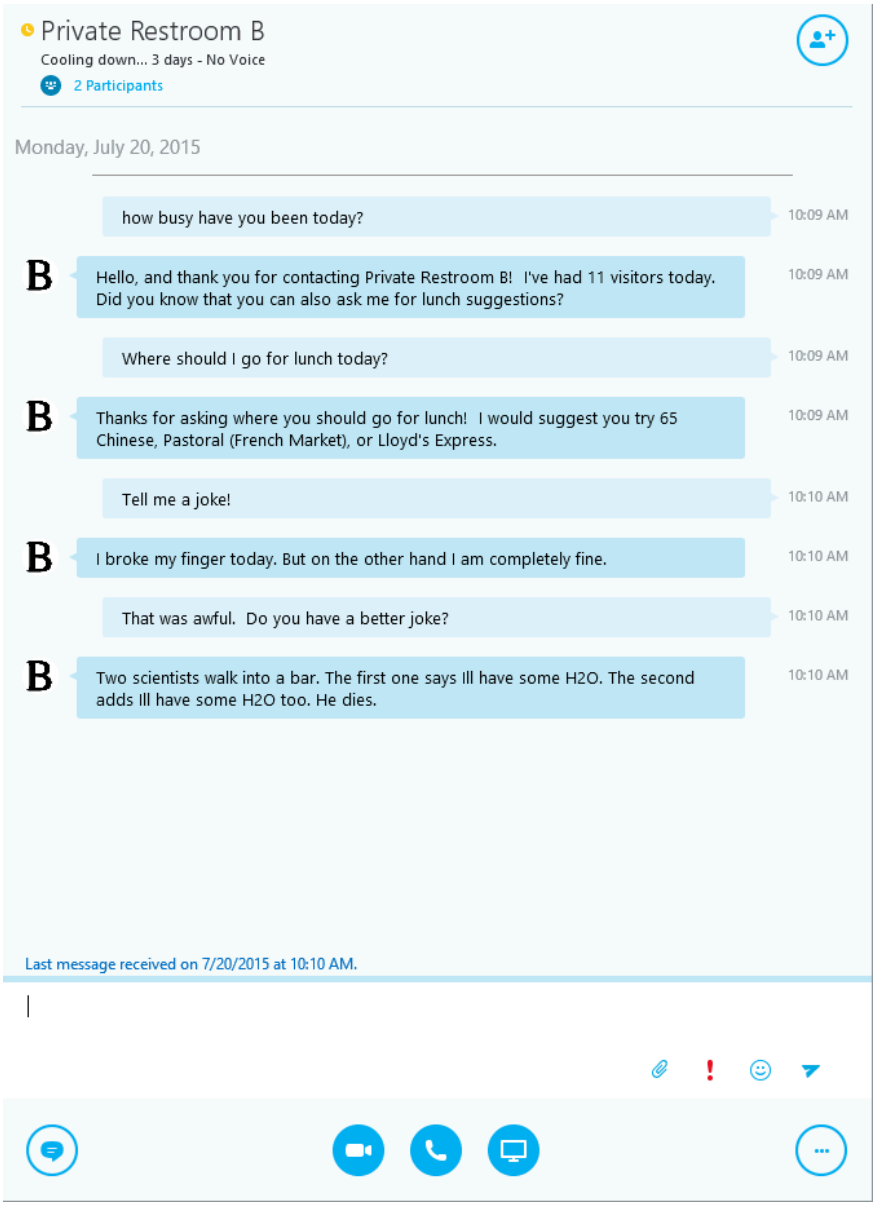
available in an office. I wired up female connectors from Ethernet ‘extension’ cable to both ends, so that the light is easy to disconnect and can use standard Ethernet cables of whatever length is needed to run the distance without having to re-solder the pins. In the picture above, it’s the upper black wire that I said I would mention later.



Pay no mind to the colorful spaghetti behind the toilet bowls...

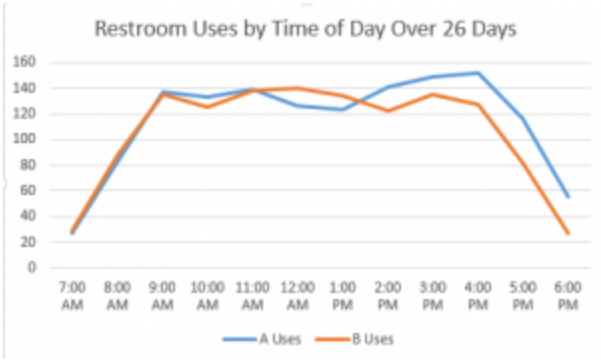
Next, since the Skype for Business endpoints that were showing the presence for the restrooms already support IM very easily (and were coming in to an application that I controlled), why not allow the restroom endpoints to have conversations? I added the ability for the endpoints to respond to inquiries about usage for the day with some basic statistics (from the aforementioned state change data), suggest places for lunch (randomized, suggesting 3 different cuisines from a database of over 50 places in the immediate vicinity), and tell jokes (all of them awful—from a database collection of several thousand).

Some people seem to have a thing about using a restroom when the seat still carries warmth from the last occupant. To accommodate those folks, I added in a ‘cool-down’ period, based on how long it eas occupied. Y’know, because it was absolutely necessary.



You have no idea how many terrible jokes there are out there. So. Many.

Finally, since data was getting sent to the service anyway with each switch change, I set up a database that the historical changes could be written to. This way, we can compile all sorts of utterly useless statistics about restroom usage, preference between the two, peak times of day, etc. What better information is there to offer at quarterly meetings?



There's got to be some justifiable reason to do this. I just don't know what it is.

And with that, the Pooper Snooper Mark II (er... Restroom Monitor) was born.

, permalink

. . .

Originally published at blogs.claritycon.com on July 20, 2015.