Reactron Overdrive: System description

Overview:
Reactron Overdrive is named after the idea that simple machine units (*Reactrons*) can create complex and desirable results when their number and breadth of function hits a certain critical mass. The actual number of necessary nodes to produce real results is actually quite low. But as the number increases, it is the combinations of these nodes that can work together to create an ever-increasing number of outputs. Overdrive is reached when, in practice, the number of valuable machine outputs, on a time averaged basis, exceeds the number of manual human inputs, especially including all maintenance and setup tasks, as well as cost.

The idea is that in the future, small machines may become ubiquitous, and available to everyone to use as a service. Once the connected nodes are numerous enough, people will be able to interact in an augmented way with technology, rather than manual use and direct control of singular machines for singular purposes. There is some hint of this beginning with cloud-based services, but instead of a distributed synergistic system, we find these data channels to be singular, simply making the singular purpose remotely controllable. This is largely so that the channel provider can create revenue. This model will likely persist for quite some time, but at some point must give way, due to practical trends. There was a time when a person needed perhaps one password. Today it's hundreds, and each login interaction takes a finite period of time, which we tolerate. Some of us do things like make our browsers remember passwords automatically to hasten the process, thereby reducing the security for which the process was designed. When the number of authentications reaches thousands, and tens of thousands, a different paradigm simply must come about, because the energy and time cost of manually maintaining the old method is too high.

The Reactron network is designed to automate many of these things from a workflow point of view, working within the constraints of modern processes. There is still a lot of room for workflow optimization, allowing machines to do things in parallel to our human activity, so that the absolute minimum amount of our time is serialized away on manual interactions. It may not currently be able to handle login interactions, since companies use things like Captcha to specifically prevent you from using your time efficiently, instead forcing you into attention- and time-stealing procedures. A small amount of time, it seems; a necessary evil, perhaps, and we tolerate it. But when it becomes too many times per day, the amount of accumulated lifetime wasted in this way becomes significant.

However, we can certainly automate many physical things for which we normally wait, such as heating elements to come up to temperature, computers to boot up. We wait at the door after ringing a door bell. Ten seconds could be saved by having a doorbell Reactron node ring itself when you signal your intent before physically being in front of a button to manually press it. The efficiency of idea to outcome is increased. There are an unlimited number of potential uses. This project seeks to create a design whereby simple nodes with whatever singular purpose they may have, are integrable and connected, for the purpose of facilitating the idea to outcome process in the most time-efficient manner possible.

Simplicity of each node is a desired trait. The system as a whole becomes more robust, when each node is simple and low-cost, easily debuggable, and easily replaceable (and potentially easily backed up in a live manner).

Data persistence is a big goal of the system, since losing data and restoring data are huge efficiency hits to one's life's workflow. Data ownership is another important concept. Data should follow the user, and not be centralized on companies' remote servers. There is no central server in this system. Every node has a duty to back up a small portion of its nearest neighbors, and space is made to house "external swap data", encrypted, piecemeal data from outside sources. This ensures persistence as the data is distributed redundantly across the nodes. No one node has all the pieces, and as such any single node failure may contain only a fraction of a percent of a person's complete dataset. And with redundancy, a node failure does not mean data loss - the network also may detect the failure and re-allocate the redundancy across the remaining nodes - and to new arrivals. As we move into the future, the expectation is that modes will be added faster than they fail, thereby extending data persistence forever. As a person physically moves through the world of ubiquitous nodes, the nodes should recognize this and create a moving sphere of attention around the user, across the nodes' shared space - with remote access to the user's personal dataspace, which would be located on nodes they own, wherever they are in the world. Thus, a mobile connection between preferences and information could constantly exist, enhancing how the machine world may react to the user, and allowing for authentication using many coincident sets of credentials - not just a single password someone could steal or guess.

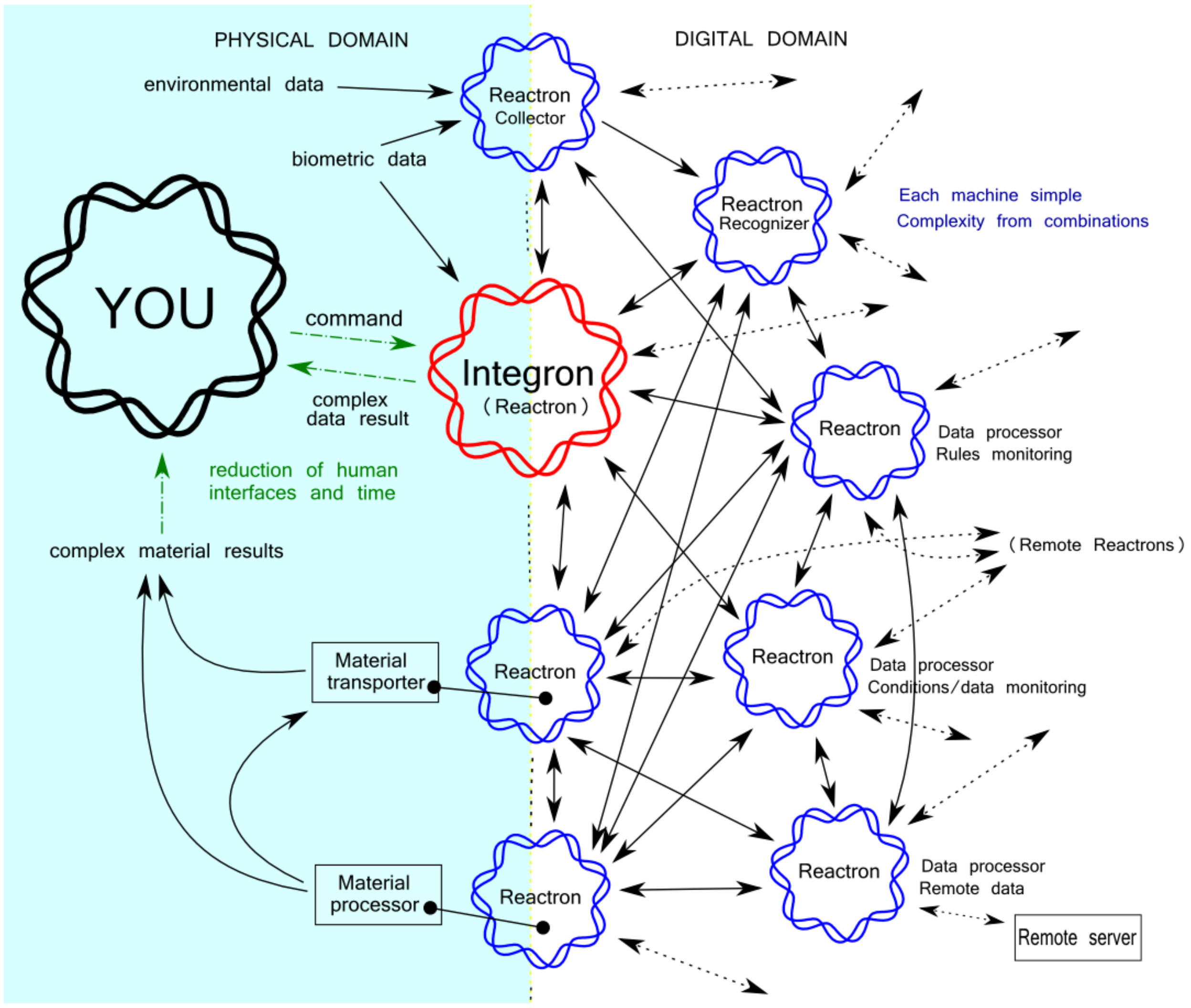
In a future version, the goal is to be able to mount these distributed data domains as personal file systems, in a secure way, from any node built for that interfacial purpose. For now, we will concentrate on rules-based outcomes based on preferences. Some examples may serve to illustrate:

- Open this door when I am within 6 feet of it (make sure it is me, and not someone else).
- Unlock this cabinet when any adult (sensed by height) is within two feet of it, otherwise keep it locked at all times.
- Unlock my car (authenticate physiognomically) if I am approaching it.
- Start my coffeemaker when I am within 100 feet of the coffeemaker, after being outside that scope for more than one hour.
- Shut off the coffeemaker after five minutes of inactivity.
- Shut off the soldering iron after thirty minutes of inactivity.

There are different levels of authentication in play (me, any adult, anyone) and they are accomplished by sensors and statistics (Recognizers). A Recognizer's sole purpose is to assemble statistically whatever sensor data is available for a known given spatial scope, and determine things that are important to the other nodes within the scope. How many people are within the scope, how many are adults (based on height, voiceprint analysis, gait), how close a speaking human is to an interface node (versus other interface nodes, to localize the interaction), has a person raised their voice to an “emergency volume”, has any person or a specific person made a certain hand gesture - these are all examples of things to be recognized. Recognizers will tally their “understanding” of the physical situation continuously, as best as can be done with the sensors at its disposal, and based on the nodes that may ask it for its assessment.

The unattended, rules-based mechanism is designed to create workflow efficiency based on normal or routine behaviors identified by the user in his own space. In addition to physical outcomes, the Reactron system is designed to create workflow efficiency with respect to information retrieval, which can often take a long time if done manually.

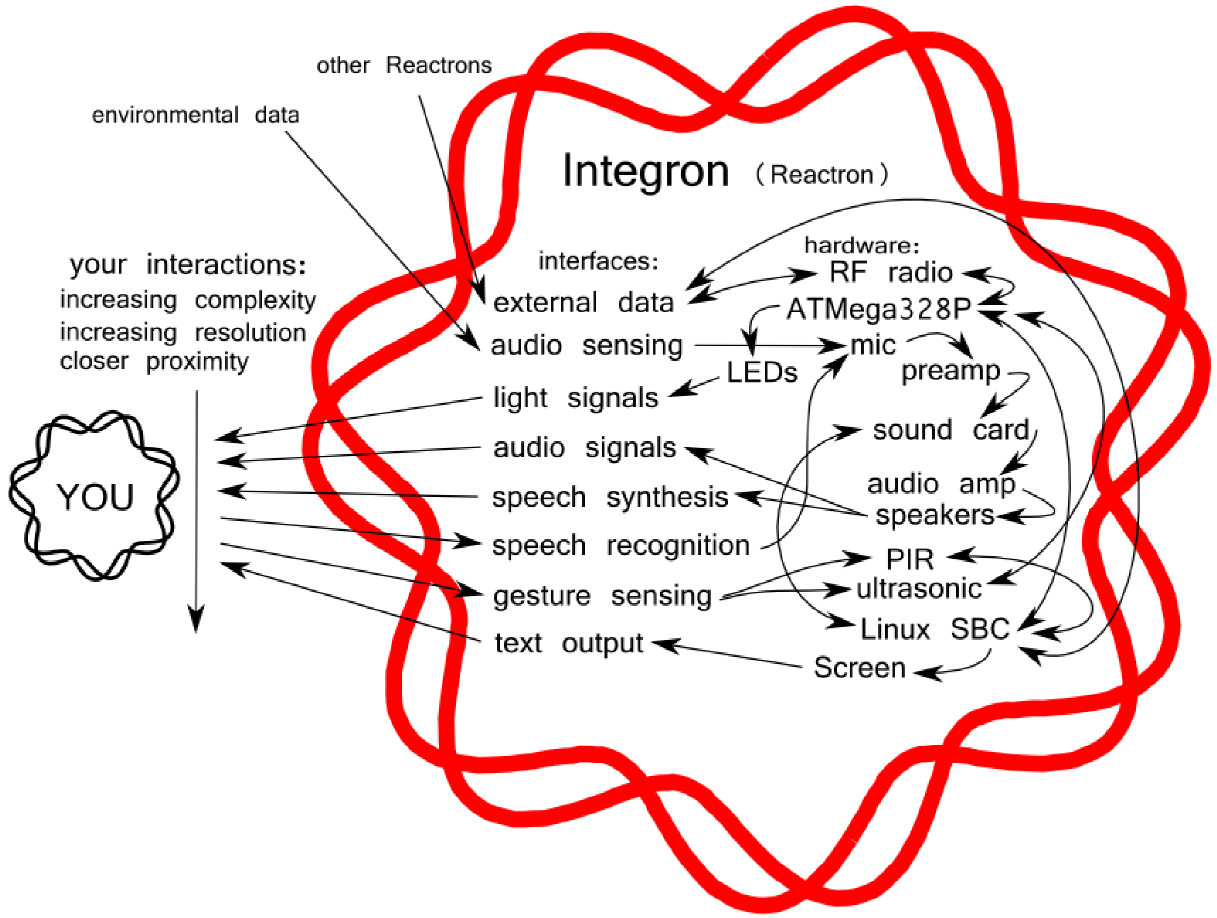
This diagram illustrates the interaction of human and machine domains:



Electronic information furnished to a user requires an interface. Interfaces vary in quality, but often the most complex interfaces, capable of arbitrary functions, are not so good at executing very specific functions. For instance, the laptop is an excellent, general-purpose human-machine interface. It is of course a machine in its own right, but as a portal to Internet-based information, it is an interface. One can open a browser, point it to an information source, retrieve that information, and view it. But that is a lot of steps.

The Reactron Overdrive project defines a new kind of interface node, called an "*Integron*". It is an "Integrating Reactron", one that attempts to facilitate an easy interface and integration between the human world, with dominant physical workflow considerations, and the machine world, where digital and power considerations are significant.

The following diagram summarizes the intended levels of human interaction and correlating internal hardware:



The Integron unit is a human-interface device, but more than that it is a human culture-machine "culture" integration device. An ambassador between worlds. It does not seek to replace the laptop, or the smartphone, but it occupies a new niche in human existence, that of the ubiquitous interface device. We actually already have these, such as power outlets on every wall, a thermostat on the wall, lights in every room for illumination of the space. This object is designed to allow human interaction with the Reactron network that is managing the aforementioned workflow.

Interaction is on several levels. Already described is the no-interaction level, rules-based management of devices. The next level of interaction is the passive signal. The Integron is a visible device, and as such one can glance at it for afar to get a gross idea of things that are important to the user. Several LEDs exist within the casing to cause the entire object to be softly illuminated. The colors and behavior of the lights can be configured to the user's desire. This creates an annunciator-type system of non-intrusive signal lights. This is, effectively, automated information retrieval, at a low- to medium- resolution. Again, some examples may serve to illustrate this passive interaction:

- Primary status light is red but not blinking: An error condition designated as important requires some eventual attention, further information is available.
- Primary status light is red and blinking rapidly: An error condition designated as critical and requires immediate attention, further information is available.
- Primary status light is blinking white: someone is at the front door.
- Primary status light is pulsing orange: you asked to be reminded of some things at this time.
- Secondary status light is yellow: the stock market is down
- Secondary status light is blue: the stock market is up
- Secondary status light is blue and blinking: the stock market is way up (consider selling!)
- Tertiary status light is green: one week has elapsed, and it's not raining (Consider mowing the lawn today - or building that mow-bot to do it for you.)

The actual meanings of the lights can be anything, but the point is, they serve various kinds of information in an efficient way by creating a passive signal. The datasets behind each signal are specific to the user, and the user may define certain Integrons to be set up one way, and others a different way. For instance, only show the "mow the lawn" signal on the unit by the back door, and only show "stock market" signals on the one on the desk in the office, but show reminders on all units mounted at a door threshold, so they may be checked when changing spaces. Display error conditions on all units. When "further information is available" that information may be obtained by closer "higher-resolution" interactions with the device, or by simply accessing another device, such as laptop or tablet or smartphone.

With annunciator lights being a low-resolution presentation of information, another higher-resolution but still passive (from the human side) interaction is via sound. The amount of information able to be represented with three or so lights, with distinct colors and luminosity patterns is less that is possible with sound. The Integron object shall be able to make sounds to alert the user to asynchronous conditions. The sounds may take the form of wave files of arbitrary contents, or they may be contextualized, synthesized speech (currently using *festival*).

Whereas the lights are good at displaying the status of things that are continuous conditions, sounds are more suited toward alerts, such as:

- “You have just received an e-mail from <this important person>”, who was previously identified as temporarily important for a specified temporary time period.
- “It’s six o’clock” (you asked to be alerted).
- “A vehicle is approaching” (driveway sensors).
- <the four famous notes of Beethoven’s fifth> (A new item concerning <specific hacker of interest> has appeared in your Hackaday feed)

The last example is one where arbitrary meaning of a sound is assigned by the user. The speech examples are short explanations of specific information. A purple light could just as easily blink six times to indicate “It’s six o’clock”, but it should be obvious that the speech-synthesized presentation of the data has higher resolution or specificity.

All of these interactions are passive, in that they are based on rules set up in advance, resulting in machine-initiated outputs or signals.

The next level of interaction is active, where the human initiates the local interaction (in other words, setting up configurations and rules is not considered an initiation of an interaction). Just as it is possible to see lights or hear sounds from a distance from the integron object, it shall be possible to actively interact with the object from a distance. The current Integron model described in the project uses only sound initiation via speech recognition. At a later time, a camera may be added to use machine vision as well. However, in keeping with the objective of simplicity, it may be that a separate “Collector” Reactron, with a camera, is used to obtain the vision data, and a separate Recognizer may be set up to then cause interaction initialization at the Integron. For now, we will discuss just the on-board speech recognition, facilitated by a simple microphone as the input.

This level of interaction allows for arbitrary commands. The Integron has a list of capabilities of its local network, and can trigger anything that the rules based nodes can, by simply requesting it. In a previous example, a door could open based on proximity to it. That same output may be obtained by manual request “Open the door” directed at the Integron. Of course specifiers may be added to make the command unambiguous, but when a not-fully argumented command is received, the Integron may query the Recognizer for context. In this case, it may cause the nearest door to open. If all of the possible combinations are too low in probability at the Recognizer, the Integron may simply ask for specificity using speech synthesis.

Commands may be directed at the Integron, but also questions, such as “How many unread e-mails do I have?” or even “What is the melting temperature of Polyethylene?” (Google search and read with text to speech - not perfect, but often useful and fast enough).

This level of interaction allows the human to interact with the machine network without having to physically be located in a particular location, such as at a laptop. There is some overlap with the utility of an on-the-person smartphone, but without the requirement to have eyes and hands completely involved with the interface.

For higher-resolution presentation of the data, the Integron unit has a small screen to display text information that is faster read than heard. The text information may further contextualize the meaning of the lights, or may simply display some default information, such as the time and weather, when no lights are on. The Integron unit will have a small ultrasonic distance detector to sense when a human is nearby, and four PIR sensors to detect hand gestures. This will allow the human to “scroll” through the different statuses of the lights, and read different pages of text on the screen, without having to touch the device. Gestures can be large and imprecise, not finely controlled such as you need for a smartphone. People will not have to be forced to be accurate by this interface, when they may be limited by eyesight or arthritis. The interface is subject to the human - the human is not required to be subject to the limitations of the interface.

This is an important issue that will become more and more important in the future, as machine interfaces become more numerous and richly featured. Increasing complexity creates this niche, a need for the ubiquitous human integration device, that enables the human to access and interact with his machine network using natural and intuitive human workflows, with minimal training.

Reactron structure:

- A Reactron is a machine with at least one method of remote communication, such as TCP/IP or an RF transceiver.
- A communications protocol may be abstracted differently for each type of transport, however the underlying structure shall be the same.
- Each Reactron is a small server in its own right, thar gives responses to connecting clients on any of its communication channels.
- A Reactron must support certain queries, even if the response is null.
    - one example is a list of voice commands that the node can manage
    - another is a list of output “abilities” (activate pump, energize relay, etc.)
    - physical location (arbitrary coordinate system, possibly user defined)
- A Reactron must maintain a small amount of shared space for other Reactrons to use. The amount may be configurable.
- A Reactron searches for other Reactrons. The search method may vary by communication transport type. In the case of broadcast-type communications (like RF) a Reactron listens to the channel and may determine other nodes based on common traffic. A specific confirmatory command establishing a “connection” may optionally be issued.
- A Reactron keeps an internal list of “nearby” Reactrons and their abilities in order to quickly parse requests and have the state of the network at the ready. A Reactron may

request the list of a nearby Reactron to obtain second-order connections. This method may be continued for a configurable number of hops, and/or may be limited by resources. All communication is with first-order connections. User overrides can connect critical path remote devices, with a specified (i.e. not searched and discovered) communications channel.

Integron structure:

- An Integron is a type of Reactron, with all of the traits above, and comprised of several Collectors (sensors, such as ultrasonic distance and PIR, with a software interface to publish results if requested) and human interaction hardware such as microphone, speakers, lights, and screen.
- The current model runs on BeagleBone Black rev C, running Debian.
- Speech recognition is done locally via Pocketsphinx
- Speech synthesis is done locally via Festival
- Text-based responses from other Reactrons are parsed by Phonetisaurus, to create lexicons tailored to the abilities of those remote nodes.
- Fixed-space Integrons typically will have a TCP/IP connection as well as an RF connection, managed by an attached Moteino (Arduino clone of small size, with integrated HopeRF radio and Flash memory chip).
- Automobile and other mobile-based Integrons, where WiFi is not continuously reliable, will use PPP over GPRS as the main connection method. (Currently investigating Adafruit FONA module.)
- A USB sound card is used for both mic input and speaker output.
- Speaker output is amplified using a small class D amplifier. (Currently investigating several Adafruit models, plus other generic modules based on PAM ICs.)

Integron physical design:
Fixed-space Integrons:

- General design is desired to be pleasant and unobtrusive, yet visible and accessible.
- Current tabletop design is a tapered cylinder, with top circular surface containing a small screen for text-based output.
- The bottom circular platform is the support structure for the unit, and has holes to admit power and Ethernet cables.
- The cylinder itself is made of acoustically transparent white speaker grille cloth. This allows sound to be collected by the internal microphone as well as sound from the speaker to be emitted. The cloth also provides a light-diffusing surface for the status LEDs to illuminate.
- Internally there is a BeagleBone Black with a cape containing the USB sound card, Moteino with Hope RF transceiver, sound amplifier, a microphone, and an HC-SR04 ultrasonic sensor. Separately mounted are four PIR sensors and a minimum of three NeoPixels. (These are separately mounted due to their critical location and orientation.)

- There is an internal baffle to act as a cone for incoming sound to the microphone. This baffle also creates a support structure and correct orientation for the PIR sensors. The PIR sensors have been confirmed as operable through the acoustically-transparent mesh.
- The ultrasonic sensor requires further testing, and an alternate plan if that testing is unsuccessful.
- Plans for ceiling-mount and wall-mount integrons are TBD.

Premlinary components list for desktop/tabletop-morphology Integron

- 1 BeagleBone Black (http://beagleboard.org/black)
- 1 Adafruit Proto Cape Kit (https://www.adafruit.com/products/572)
- 1 USB sound card (http://www.amazon.com/gp/product/B000N35A0Y/)
- 1 electret mic (https://www.adafruit.com/products/1064)
- 1 audio amp (https://www.adafruit.com/products/987)
- 1 speaker of pair (https://www.adafruit.com/products/1669)
- 1 TFT screen (http://www.adafruit.com/products/1651)
- 10 NeoPixels 9https://www.adafruit.com/products/1558)
- 1 R4 Moteino (http://www.lowpowerlab.com) w/ HopeRF radio and 4Mbit Flash memory option
- 1 HC-SR04 ultrasonic ranging sensor (https://lowpowerlab.com/shop/index.php?_route_=HC-SR04)
- 4 generic PIR modules
- cast acrylic sheet for casing, sound baffle, and light diffusers
- sound transparent speaker grille material
- PVC tubing for PIR housing

Currently working (20 Aug 2014):

- Audio (USB sound card, mic, audio amp, speaker (mono).
- Speech recognition (though slower than desired)
- Speech synthesis (though slow than desired)
- Signal audio file playing (ALSA)
- LED signals (NeoPixel)
- PIR tests through acoustically transparent sleeve
- Ultrasonic sensor standalone, unobstructed
- RF communications
- ATMega328P board (Moteino)
- TCPIP connectivity (wired)

Currently still in development (20 Aug 2014):

- Better audio (mic, preamp?, amp)
- Better CPU performance on BBB (overclocking?)
- Ultrasonic sensor tests through acoustically transparent sleeve
- Coarse gesture sensing with PIRs and Ultrasonic together as a subsystem
- Screen output coordinated with Reactron network data
- Gesture navigation of screen data
- Configuration structure for mapping annunciations to external datasets
- Better communications encryption
- Wireless TCPIP connectivity option (requires USB hub)
- USB hub option
- Internal baffle structure configuration
- BBB-Moteino serial communication
- Assessment on screen existence requirement

Automotive Integrons:

- Automobile itself is the "casing"
- Single status LED, two locations (door sill and dashboard). One for viewing when driving, one for viewing when outside the car, at the driver's door - essentially two different interfaces. No screen-based or gesture interface; it is passive, and active-voice only.
- NeoPixels with light pipes used to drive the status LEDs.
- No other visible components.
- Microphone and speaker behind small grille in ceiling compartment.
- Antenna locations TBD, since roof is metal.
- 12VDC operation.
- No PIR or ultrasonic sensors - future model may include camera for gesture sensing
- Integrated GPS module for location coordination with fixed-base Reactron network.

Premlinary components list for automobile Integron

- 1 BeagleBone Black (http://beagleboard.org/black)
- 1 Adafruit Proto Cape Kit (https://www.adafruit.com/products/572)
- 1 USB sound card (http://www.amazon.com/gp/product/B000N35A0Y/)
- 1 electret mic (https://www.adafruit.com/products/1064)
- 1 audio amp (https://www.adafruit.com/products/987)
- 1 speaker of pair (https://www.adafruit.com/products/1669)
- 2 NeoPixels (https://www.adafruit.com/products/1558)
- 1 GPRS module (https://www.adafruit.com/products/1963)
- 1 GSM/Cellular Quad-Band Antenna (https://www.adafruit.com/products/1858)
- 1 GPS module (http://www.adafruit.com/products/746)

- 1 R4 Moteino (http://www.lowpowerlab.com) w/ HopeRF radio and 4Mbit Flash memory option
- 1 SIM card
- Clear acrylic light pipe material

Currently working (20 Aug 2014):

- Audio (USB sound card, mic, audio amp, speaker (mono).
- Speech recognition (though slower than desired)
- Speech synthesis (though slow than desired)
- Signal audio file playing (ALSA)
- LED signals (NeoPixel)
- RF communications
- ATMega328P board (Moteino)
- Standalone GSM/GPRS connectivity from FONA module

Currently still in development (20 Aug 2014):

- Better audio (mic, preamp?, amp)
- Better CPU performance on BBB (overclocking?)
- GPS module integration
- GPRS communications integration
- BBB-Moteino serial communication
- Better communications encryption
- Installation in automobile!