Ubicorder: A Mobile Device for Situated Interactions with Sensor Networks

Manas Mittal, *Member, IEEE,* and Joseph A. Paradiso, *Senior Member, IEEE*

***Abstract*—The Ubicorder is a mobile, location and orientation aware device for browsing and interacting with real-time sensor network data. In addition to browsing data, the Ubicorder also provides a graphical user interface (GUI) that users can use to define inference rules. These inference rules detect sensor data patterns, and translate them to higher-order events. Rules can also be recursively combined to form an expressive and robust vocabulary for detecting real-world phenomena, thus enabling users to script higher level and relevant responses to distributed sensor stimuli. The Ubicorder's mobile, handheld form-factor enables users to easily bring the device to the phenomena of interest, hence simultaneously observe or cause real-world stimuli and manipulate *in-situ* the event detection rules easily using its graphical interface. In a first-use user study, participants without any prior sensor network experience rated the Ubicorder highly for its usefulness and usability when interacting with a sensor network.**

I. Introduction

SIGNIFICANT research effort has been expended in building sensor network platforms targeted at Ubiquitous Computing applications [1]. These form a subset of a larger trend of deploying general purpose embedded platforms and instantiating specific applications on them [2]. Such platforms are generally designed for engineers and sensor network specialists. There has been very limited research directed toward lowering the threshold for interacting with such general purpose sensor networks. We believe there are significant advantages in enabling non-experts to explore, experiment with and utilize the facilities provided by a sensor network. Firstly, this will enable users of the instrumented environment to identify and develop their own applications that use the sensing capabilities available. Secondly, raw sensor data can often be useful, especially when examined in context of the location of particular sensor nodes, which could inspire users to design heuristic-driven inference rules operating on that data.

In this paper, we present the Ubicorder (Figure 1) [3], a mobile, location and orientation aware sensor network browser and inference tool. The Ubicorder lowers the threshold for users to observe, interact with, and draw inferences from real-time sensor network data. Consider how humans generally use sensor data. First, we *discover and identify the sensing capabilities* of an instrumented environment. Second, we draw heuristics to *translate sensor data into real-world events* of interest, and third, we *apply these heuristics to infer* such events in the future. The Ubicorder's subsystems parallel this process. The Ubicorder's *browser* enables discovery of sensing capabilities and lets users draw correlations between sensor data and physical phenomena. A second subsystem: *EDITY (Event Detection and Identification System)* enables users to formalize such heuristics into inference rules that automatically translate future data into real-world events.

For browsing, the Ubicorder provides a mobile location and orientation-aware interface to sensor networks. Users can use its point-and-browse affordance to discover and explore the facilities of a sensor network; the Ubicorder displays an interactive map of the sensor nodes available, showing the sensing modalities of each node and renders the real-time data generated by these sensors.

The Ubicorder's EDITY subsystem incorporates a graphical language for defining and combining inference rules, and an inference engine that detects and flags when such events occur. EDITY also addresses the scalability issues implied when viewing real time data from multiple sensors. Instead of showing raw sensor data limited to a few sensors, inference rule abstractions make it convenient to monitor an order-of-magnitude higher number of sensor data streams.

The browsing and event detection/definition functionalities complement each other. The Ubicorder's mobile, handheld form factor enables users to easily bring the device to the phenomena of interest, hence simultaneously observe or cause real-world stimuli and manipulate *in-situ* the event detection rules using EDITY's graphical interface. Creating a simple interface for enabling quick creation and iteration of event rules presents a challenge. Borrowing from the programming by example/demonstration communities [4], [5], the Ubicorder's EDITY (Event Definition and Identification System) allows users to define inference rules by *performing* or *observing* the action, correlating it with the corresponding sensor data, and designing a rule to detect such events in the future.

Our vision is that by facilitating the use of such networks by non-experts, we will promote a richer exploration of the application space for sensor networks and the data they generate. The Ubicorder, enabled by advances in mobile computational, sensing and communication capabilities, is a step in this direction. We envision a sensor network utility technician (here considered a "non-expert" in contrast to engineering expertise currently needed to deploy such systems) who will use the Ubicorder at installation to draw up the first few inference rules, which serve as building blocks and components for other such rules that are actively constructed and tweaked by users of the space.

Manas Mittal is with the MIT Media Lab, Cambridge, MA, 02139 USA e-mail: (manas@media.mit.edu)

Joseph A. Paradiso is with MIT Media Lab, Cambridge, MA 02139 USA email: (joep@media.mit.edu)

Manuscript received January 20, 2010; revised April 20, 2010.

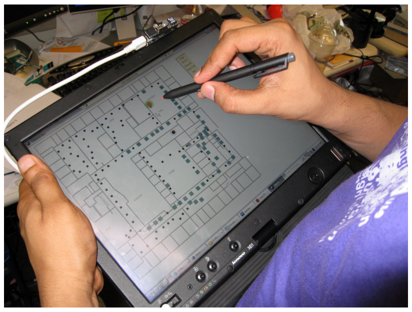


Fig. 1. Holding the Ubicorder. Custom hardware is mounted at the top of the tablet

The main contributions of this work are:

- A location and orientation aware sensor network browser that exposes the capabilities of a sensor network and graphically renders real-time sensor data.
- A trigger definition language and schema for inferring events based on sensor stimuli.
- An intuitive, mobile, visual, graphical interface to easily define, experiment and iterate over sensor rules, and an inference engine that detects events so defined.
- Techniques and models that support rapid exploration of inference rules through the application of the design-test-analyze [6], [7] paradigm on a much shorter timescale as compared to existing tools.

Next, we describe the Ubicorder's browsing functionality, followed by a description of EDITY. We then discuss the implications of such a system and describe our observations from a user study. We conclude with a discussion of related work.

II. Sensor Network Browser

The Ubicorder's browsing mode allows the user to browse real-time sensor data through a location and orientation-aware, map-based interface. Displaying the user's present location relative to the sensor nodes assists them in contextualizing their location. The browsing interface provides, at a glance, an overview of the current state of the monitored area.

A. System Components

Although not designed for a specific sensor network, the Ubicorder currently displays data from two sensor networks deployed on the third floor of the MIT Media Lab. The first is a network of 150 ceiling-mounted motion sensors [9]. The other sensor network was the 50 node "Spinner" sensor network [8], [10] that was in the process of being rolled out when the Ubicorder was developed. Each "Spinner" node contains a minimum of temperature, light, sound and vibration sensors, while also serving as radio (Zigbee) and Infrared (IR) beacons. In our implementation, both "Spinner" and motion sensors exposed their data over Wi-Fi gateways.

The Ubicorder is composed of a touchscreen (pen sensitive) Tablet PC (IBM X60) with Wi-Fi and a Universal Serial Bus (USB) port. Additional hardware (Fig. 1) was necessary to acquire the location and orientation information. We use

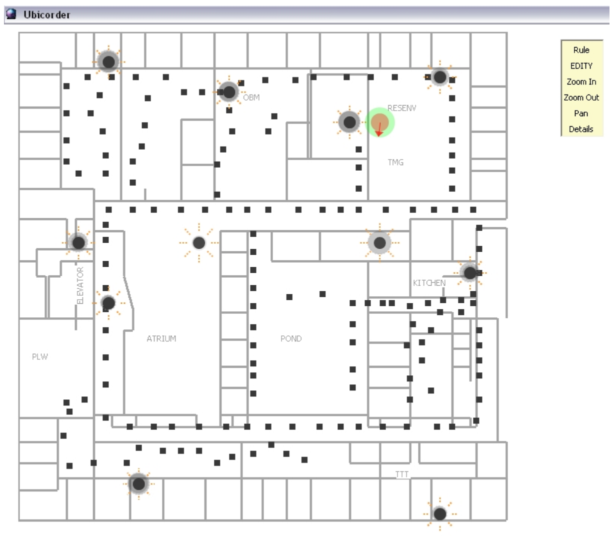


Fig. 2. Browsing Interface of the Ubicorder. Square icons represent movement sensors. Circular black icons correspond to "Spinner" [8] nodes. Icons jitter to convey vibration, change color to indicate temperature variations, change the size of their halo to represent sound level variations, and vary the length of emanated lines to indicate light levels. The square icons "pop" to indicate motion underneath. The pointed circular icon represents the user's location/orientation, and increases in size as the localization resolution grows coarser. More modalities are detailed in [3].

a three-axis, tilt-compensated digital compass for determining orientation (Honeywell HMC). We use the Ubicorder's wireless radio (Chipcon Zigbee) to provide coarse, room-level wireless localization. The Ubicorder also supports, where available, IR-based localization. Sensor nodes incorporate IR emitters and serve as IR beacons. The Ubicorder maintains a map of such beacons and uses this information to localize and point more accurately than afforded by the Zigbee radio and compass.

B. The Graphical Interface

On the Ubicorder screen, the browsing interface displays a floor plan overlaid with icons depicting sensor nodes. Icon shapes indicate the *type* of sensor node (Fig. 2).

Coarse grained, real-time sensor data is encoded in the icon's form. For example, a movement sensor temporarily "pops out" (i.e., grows larger) upon registering motion underneath. Fig. 3 shows a series of motion sensors "pop out" as a person walks underneath. Similarly, the circular multimodal "Spinner" icons change form to convey dynamic sensor data. Some of these variations are shown in Figure 4 and in [3]. The default color scheme (gray/black) of the map is chosen so that it looks dull when there is no activity or significant variation in sensor data.

C. Navigation and Context

The user's approximate physical location and orientation are conveyed as an icon on a map displayed on the screen. *Situating* the user simplifies navigation, both on the screen and in the physical space. On screen, we place a directional arrow (a circular "me" icon, Fig. 5) at the center of their icon

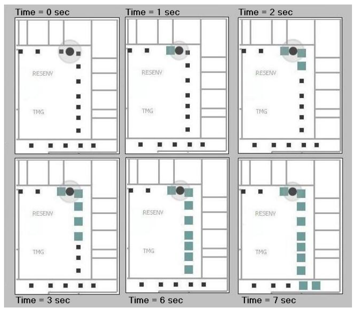


Fig. 3. The six images represent successive time slices, earliest (top left) to latest (bottom right). Black icons indicate ceiling-mounted motion sensors. Here, it is easy to infer that at least one person is walking underneath.

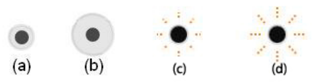


Fig. 4. Variations in sensor data (a) low-sound level (b) high-sound level, (c) low-light level (d) bright-light level

thus indicating the user's assumed orientation and location. The icon is dynamically responsive as the user walks around the building. The location accuracy is variable and depends upon the network's support for Zigbee or IR localization. We convey this locational uncertainty by increasing the diameter of the "me" icon (see Figure 5). Note that the whole map can rotate to align with the users view (like in our prior work, the Tricorder [11]), but some users found it easier to select a sensor node when the map remained static (i.e., did not reorient) and the "me" icon's arrow indicated orientation.

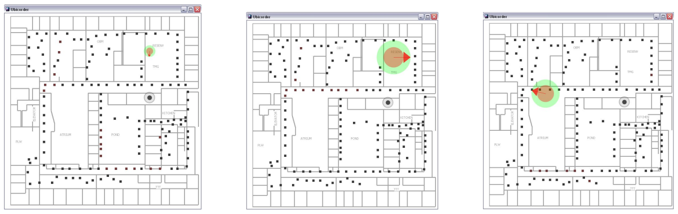


Fig. 5. The large (green and red) circular icon represents user's location. The icon's arrow indicates orientation, and the size of the icon changes with the localization accuracy.

III. Rule Scripting Interface: EDITY

While the browsing interface provides an overview of the current state of an area, it is difficult to display precise quantitative data from multiple nodes and multiple sensors in parallel via the browsing UI. However, in order to draw meaningful inferences, it is vital to be able to view such sensor data, including data from multiple nodes. Users, especially end users, are typically interested in the inferences that can be drawn from the data, rather than the data itself.

The Ubicorder's EDITY subsystem enables users to define, manipulate, and test simple inference rules that map sensor data to meaningful higher-level primitives. Commonly occurring sensor data patterns that correspond with physical phenomena of interest can be abstracted away as (higher-level) events. Users can design, experiment and iterate over such rules for identification of these events. The system can then apply these rules to detect such events upon receipt of future sensor data. Rules can be defined/combined in a piecewise and recursive fashion. The inference rules are organized as either simple rules or compound rules. Simple rules operate directly on sensor data, and monitor a single sensor stream. Compound rules combine multiple simple rules and/or compound rules to form new compound rules. In the following sections, we discuss in detail both the formal grammar and the interface for defining and experimenting with such rules.

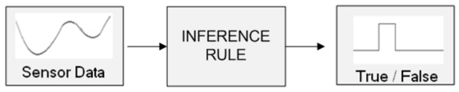


Fig. 6. Inference Rule, Input and Output

A. Definitions

We define an **event** as an occurrence of a physical phenomenon or action in the real world. Users (using EDITY) construct **rules** to infer events based on sensor data. A rule maps a stream of sensor values to Boolean values, testing a **condition** (Cd). One or more rules, when true, indicate the occurrence of an event. **Filters** (filters) may be applied to the sensor stream prior to testing for conditions.

Inference rules are of two types: simple and compound. **Simple rules** (Simple) test for conditions on a single sensor stream. **Compound rules** (Compound) are boolean, time-dependent combinations of simple rules, and/or compound rules. **Operators** (Oper) combine outputs of rules to form new compound rules. **Time Slack** (Ts) introduces time dependency for creating compound rules. If any (filtered) component rule is true within the time slack, the output of the component rule is considered true.

B. Simple Rules

Simple rules check for conditions on a single sensor data stream. Here we discuss the interface and mechanisms for defining such rules. The process of defining rules can be categorized into three steps: 1. Selecting a sensor, 2. Defining and manipulating decision rules, and 3. Linking actions and recording simple rules.

In keeping with the standard engineering metaphor of data flowing left to right, the input port, operation, and output port are placed left-to-right on the screen. The sensor/node selector forms the left pane (input), the rule definition, manipulation and detection subsystem forms the middle pane (operation) and the output, i.e., storing events and or linking actions (output) forms the right pane.

A user switches from the browsing mode to the rule creation mode by clicking a toolbar button, then is presented with the "Simple Rule Interface" (Figure 7). The left pane of this interface allows the user to choose the sensor stream of interest - by either tapping on the node's icon or by physically pointing the Ubicorder toward a node (if the node has an IR beacon).

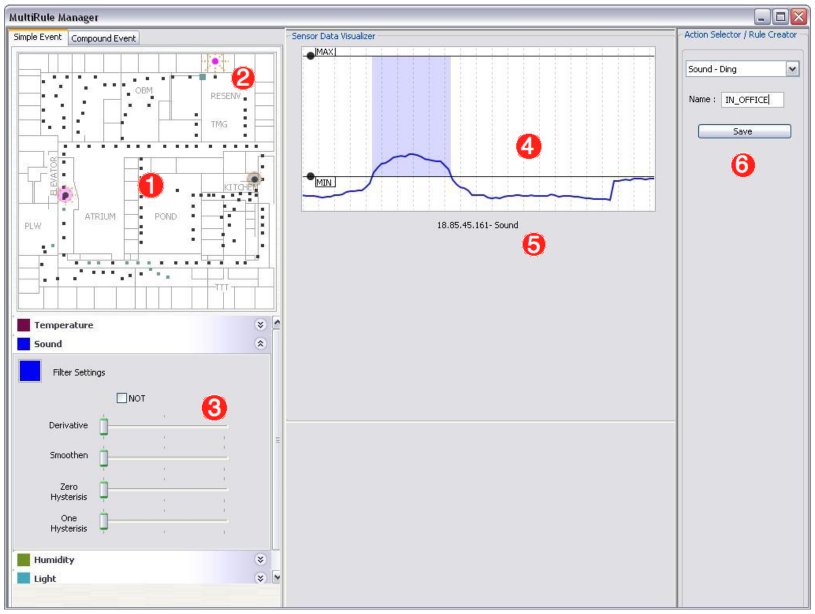


Fig. 7. Screen-shot of the Simple Rule Interface (1) Map indicating location/type of sensor nodes (2) Selected Sensor Node (3) Sensors on Selected Node (4) Data Stream on Selected Sensor, shaded region indicates sensor samples satisfying thresholds, (5) Node, Sensor name, (6) Action: Name of the Rule (IN_OFFICE)and actuation (Sound - Ding)

The selected node is highlighted on the map, and a list of sensors exposed by the node is displayed in the lower half of the left pane (Figure 7(3)). Tapping on a sensor name selects it, i.e., expands the sensor name box to display filters that may be applied to it.

While our interface and model are not directly tied to particular filters, we support five basic filters - Not(N), Smoothing(S), Derivative(D), Positive(Ph) and Negative Hysterisis(Nh), Debouncing(Db) and Deglitching(Dg). Filters expose a tweakable *"k"* parameter that varies their sampling window size, or in the case of hysterisis and de-glitching filters, holds the signal for k samples. Ideally, for a completely expressive interface, these filters should be stackable in a user-defined order. However, in our experience, we found the filters to be intuitively useful in the following (default) order - N, Dg, S, Ph, Nh. The filter model is extensible, and additional filters may be added later. In terms of formal grammar, this can be expressed as:

$$Ss \overset{Fs}{\rightarrow} Ss$$

where Ss indicates Simple Stream, Fs indicates a filter.

The real-time sensor signal (post filter application) is displayed in the middle pane, as a scrolling strip chart (Figure 7(4)). The user can also freeze the strip chart, or scroll back to view past data. The user defines and experiments with the conditions that specify the rule. We currently support upper and lower threshold conditions. A pair of user-draggable horizontal lines signify the upper and lower thresholds being set. As the user moves the horizontal lines, the section of the data stream satisfying the constraints are highlighted. Further, the user can "look-back"into the data stream to examine the time segments where the constraints are satisfied.

Events may triggered when rule conditions are met. The right pane enables the user to link the rule with an action and save it. Currently supported actions are selecting or coloring the node (in the map on the left pane), playing an audio file, or sending a keystroke to an external program (through the Operating System, using the Java Robots API [12]). The last option allows the user to control external programs. In the interface, a drop down "action" menu lets the user choose pre-defined actions. The action is either edge triggered or level triggered, based on user choice. Formally, this can be expressed as:

$$Ss \overset{Cd}{\rightarrow} Simple$$

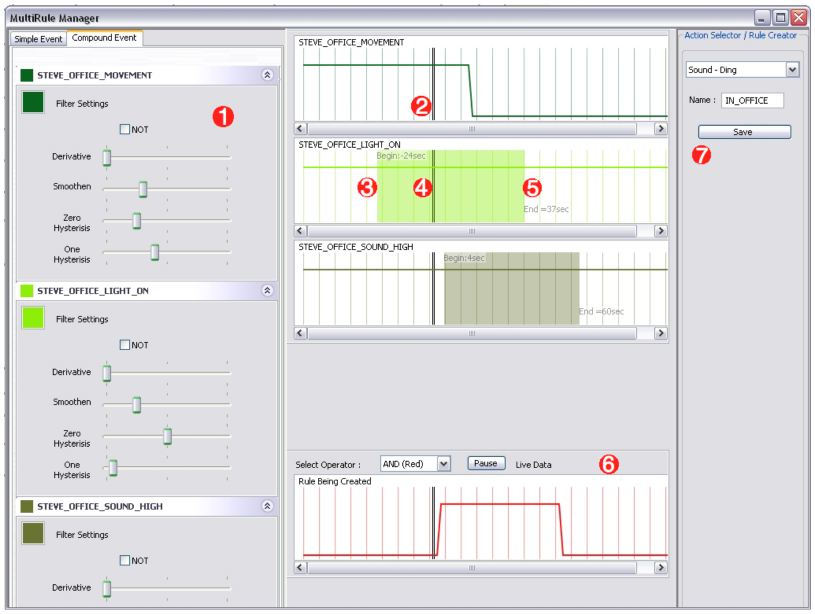
Where Ss indicates Simple Stream, Cd indicates Condition and *Simple* indicates Simple event.

Additionally, the user can "save" the rule and use it as a component for further rules. Saving implies that the rule (as defined by the selected node, the sensor on the node, the filters applied, and the thresholds) will be evaluated whenever new data is received from the selected sensor. A binary output stream is exposed by such a saved rule. These simple rules form the building blocks for *compound rules*, discussed in the next section.

C. Compound Rules

While the simple rules provide a simple and effective mechanism to set threshold events for a sensor stream, real events of interest can be better inferred by observing multiple sensor streams simultaneously.

Rules involving multiple sensors (or simple rules) are called "compound rules" (*Compound*). A compound rule is defined by combining the output of multiple simple or (existing) compound rules. When forming a compound rule as a combination of simple (*simple*) rules, this effectively translates as directly applying conditions on two sensor signals. Simple rules may



Select Operator : AND (Red)

Sound - Ding

Name : IN_OFFICE

Fig. 8. Screenshot of the Compound Rule Interface. The left panel (1) has a list of existing rules, three of which are selected and have their filters set. The middle panel displays strip charts for the rules (in order of selection). The time dependency is defined by the black double line(2,4), and by the highlighted time slack in the strip charts(3,5). The bottom middle panel (6) shows the result of the rule that the user is currently designing(color image)

be combined with Boolean operators (*oper*) such as *and, or, and xor*. Rules may also be linked temporally, using the concept of *time slack* discussed later (*Ts*). Formally, this can be expressed as:

$$(TsSimple)Oper(TsSimple) \rightarrow (Compound)$$

The Compound rule so defined may be combined further with previously defined simple rules. The same operators and time slack concepts are applicable. Compound rules provide the flexibility to define more and more complex rules that might be necessary to model a real world phenomenon. This can be expressed as:

$$(TsCompound)Oper(TsSimple) \rightarrow Compound$$
$$(TsCompound)Oper(TsCompound) \rightarrow Compound$$

For simplicity in describing the workings of the system, we refer any rule that contributes to a compound rule as a *component rule*. Internally, a compound rule is represented as a node in a directed acyclic graph (DAG), with the component rules forming the children of a node depicting the current rule. The simple rules form the leaves of such a graph.

In the UI (Figure 7), clicking on the Compound Event Tab changes the view to the compound panel (Figure 8). The left pane contains a list of all existing simple/compound rules. Clicking on a rule name (Figure 8 (1)) toggles the inclusion of the rule as a component rule. Selecting a rule for inclusion also brings up a filter panel (same as that for a simple sensor). Note that for a compound rule, more than one component rule can be selected at a time.

In the middle pane, a new strip chart is created for every (included) component rule, and shows the near-real-time output of this rule. The strip charts are color-coded (lines are drawn with the color of a component rule as specified in the left pane).

One of the challenges of defining compound rules is to describe the temporal connection between the different component rule events that define a compound rule.

EDITY supports specification of time dependency in the form of "*Event Y happens between (p,q) seconds after Event X*". That is, event Y happened at least p seconds after X and that event Y happened at most q seconds after X. Note that p and q can be negative numbers.

The concept of *time slack* is introduced. Time slack implies that a component rule, if it is *ever* true within a specified time window, will be construed as being true when evaluating the compound rule it participates in. The beginning and end of the time slack window are specified with respect to the first rule. The first rule, by definition then, has no time slack (instead, it sets the zero point). Graphically, the time slack is set by dragging two vertical lines on the strip chart. The zero line (black vertical line in the strip chart) is set by default in the first rule, i.e., the rule first selected, and therefore displayed at the top of the strip chart pane (Figure 8(2) shows this line). This zero line is then synchronized across all the other strip charts (for example, Figure 8(4) shows one such line). Two additional time slack lines are also drawn in all but the first strip chart, corresponding to the time slack for each component rule.

At this point, three lines are visible on a strip chart (except on the strip chart corresponding to the first selected sensor, where only the "zero point" thin black double line is visible) - Figure 8(4). The second line, (Figure 8(3)) drawn in the color of the current strip chart, indicates the lower bound of the time slack, i.e., the event must happen at least *p* seconds after the first component is found to be true. The third line (Figure 8(5)) specifies the maximum time slack, that is, the event must happen by the end of this time period. Setting the time slack highlights the corresponding region on the graph.

A drop down menu at the bottom of the middle pane allows selection of Boolean operators (And, Or, Xor) for combining the component rules. The output of the rule is displayed in a strip chart at the bottom of the middle pane (Figure 8(6)).

The right pane is the same as in the simple rule interface. The selected rule can be saved, and used recursively as a component rule for the design of another compound rule. Being able to recursively define rules is crucial to provide the flexibility to express real-world phenomena. The user study (discussed later) describes one such scenario in detail.

IV. Discussion

We discuss four important threads about the EDITY system:

A. In-situ *Creation of rules*

The Ubicorder is a mobile device. Being mobile allows the user, when defining the rule, to be physically present at the location of the phenomena, thereby allowing them to correlate the real event (*cause*) with the observed sensor data (*effect*). Such a configuration has three advantages. First, it simplifies understanding the sensor behavior, as the user can learn about the sensors and their data streams by actuating a change in the real world and observing the corresponding sensor signature. Second, it facilitates quick testing and tweaking of the rule being designed. Third, allowing the user to see the correlation between sensor signals, real phenomena, and the output from the rule thus designed, exposes to the user to the limitations of the sensing infrastructure, and could prompt the installation of new sensors or relocation of old ones.

B. *Expressiveness of the rules*

While EDITY only supports simple amplitude threshold based conditions for simple rules, the general model of using filters and visualizing thresholds is generic and extensible. More complex conditions can be implemented using well-designed filters. For example, consider implementing a correlation condition. A correlation filter can be added that translates a sensor signal to a correlation coefficient with respect to another signal (specified in the filter), upon which thresholding can be applied. Similarly, more expressive rule combiners can be used when forming compound rules.

C. *Boolean Operators*

Users are known to face problems when designing database queries that involve Boolean operators, the so called "Boolean bottleneck" [13], [14]. Some of the reasons behind this difficulty are (1) difficulty in the use of parenthesis and order of evaluation when specifying queries, and (2) confusing the Boolean operators AND, and OR with their counterparts in common English language. The EDITY system parenthesizes every component rule, i.e., it evaluates a component rule completely before plugging its Boolean value into its compound rule. Thereon, every rule that the user designs is implicitly parenthesized.

The EDITY system addresses the second problem, i.e., confusion between logical AND/OR with normal English usage of "and", and "or". First, we argue that our model of AND/OR operators closely parallels their equivalent English usage. In our context, such terms define linkages among the truth stage of component rules. Each component rule itself would generally map to some observable phenomena. For example, a compound rule to detect if someone is in the office might consist of one component rule for increased light level, and another for increased sound level. In this case, the condition will be defined in usual English as "A person is in the office if the light is switched on *and* the sound level is high", similar to its usage in EDITY. Secondly, by displaying to the user the graphical output of the rule while she devises it, we encourage the user to experiment rather than analyze; the interface lets the user quickly see the result of the rule she has so far created. Therefore, the user matches the output of the rule to the desired output.

TABLE I
Expressiveness of Rules

Expressiveness		
Event Type:	**Equivalent EDITY construction**	**Example**
On-Off (Binary) Events	EDITY event defined by a threshold-based Simple Rule	If Light Level is above threshold, Floor Lamp is swiched on
(1-2-3-4 ..) (Discrete State) Events	EDITY event defined by Compound Rule comprising of multiple Threshold-based Simple rules.	Detect if a user is 'busy', i.e., study lamp is switched on and sound level is below a threshold.
Events matching a sensor pattern	Using a correlation, or dynamic time-warping filter	Accelerometer based sensing
X after Y events (Temporally Linked)	Using the concept of time slack	Trigger a series of motion sensors to detect a person walking. Link a series of binary simple rules.

D. *Modularity*

Combining multiple sensor streams provides a more expressive language and improves recognition accuracy. Recursively building rules (using component rules) brings the advantages of modularity to the rule making process. "High-level" sensor outputs can be re-used as components for several compound rules. The modular approach ensures that simpler rules can be defined and debugged completely before more complex rules are defined. Further, rules can denote physical phenomena (for example, "Joe in office" and "Joe in car") and can be combined to imply higher order phenomena ("Joe not at home if in the car or office"). Additionally, this modularity lends itself well to sharing of rules. A repository of rules describing standard states of a given space can be incorporated. A subset of these rules can be designed by domain experts or even through common-sense databases [15] . Non-expert users can then incorporate these rules in the compound rules they create.

V. Evaluation

We conducted a controlled first-use user study to evaluate the utility and the usability of the Ubicorder and its EDITY system. The study group was comprised of ten participants. They came from a variety of educational backgrounds: four from Electrical Engineering/Computer Science, three from other engineering/science fields, and three from design. Five participants were graduate students, three were undergraduates, and two were post-doctoral researchers. An additional two participants served as pilot testers, hence their observations are not included in the final analysis.

All participants had at least some programming experience. Half the participants had no knowledge of sensor networks. Of the remaining five, three had some experience with sensor networks and two were experts, with experience both in designing and deploying sensor networks.

Half of the participants also had some experience interpreting data from electronic sensors by simple eyeballing techniques. Two had written computer programs to analyze and interpret such data. Almost all participants (9) were aware of, and comfortable with Boolean operators. None of the participants had any functional knowledge of the deployed sensor networks in the building, and only four were aware of the existence of any sensor network in the building. This was mildly surprising, since most participants worked in the same building.

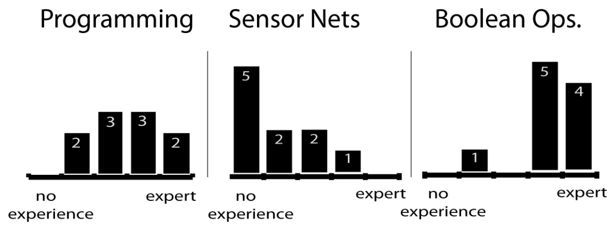


Fig. 9. Prior experience of our study participants (5 point scale)

A. Study Protocol

Two sensor networks were deployed around the Media Lab's third floor. One was the 150-node ceiling mounted motion sensor system from MERL [9]. The other was our own "Spinner" sensor network, which had three nodes deployed in our lab area, and one in an office nearby.

Each trial lasted for an hour and fifteen minutes. The study started with a pre-survey questionnaire, followed by a brief (10 minute) introduction. The user was then asked to perform a task. The task was divided into two smaller subtasks:

1. *Warm Up Task, Task A*: Use the Ubicorder's browser to locate a "Spinner" node in a dark room. Then author a rule to detect if the room is occupied. Multiple sensor nodes (with light, motion, humidity and sound sensors) were deployed in different rooms. The user was expected to design a simple single-modality rule.

2. *Main Task, Task B*: The participant was asked to design rules to detect if a lab workbench (with a soldering station) was in use. An actor portrayed a typical use-case scenario, i.e., the spotlight and (audible) fume exhaust fan were turned on. There was a second ambient light that may or may not be turned on while the unit was being used. A multimodal sensor node was placed on the workstation. The user was instructed that they may move this sensor node if needed, and could freely turn on or off the lights, exhausts etc.

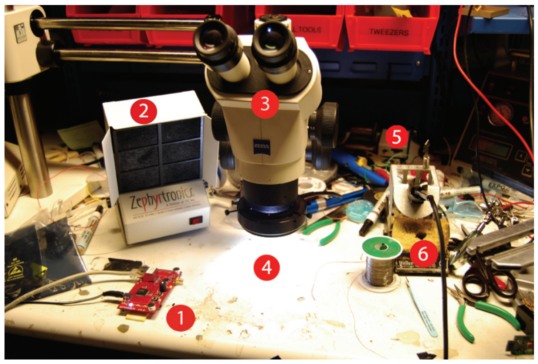


Fig. 10. Zoomed in view of the Spinner Node deployed on the microscope/soldering workbench. (1) Spinner Node, (2) Fume Exhaust Fan, (3) microscope, (4) Area under the microscope, illuminated by a LED spotlight, (5) LED spotlight switch, (6) Soldering Iron

This task was chosen because it provides an opportunity for the participant to design a straightforward (but non-trivial) compound rule involving sound and light sensors. This setup also allowed participants to create more intricate rules involving additional sensors. Around half of our participants were familiar with the workbench, and so it therefore introduced invariance with regards to prior knowledge of the task. The participant was instructed to program an alert or actuation to be triggered when the soldering area was in use. For example, participants could set audio alerts ("ding" sound) or visual alerts (box drawn around a node, changing the color of the node, etc.). The study concluded with a post-task questionnaire, which contained both Likert scale and essay type questions.

B. Observations

Participants were able to successfully make inference rules that detected the phenomena with varying degrees of accuracy. Participants took a variety of approaches in defining rules, some of which had not been originally envisioned by the study designer when the user study was originally formulated. For the warm up task, participants always designed a rule that used light levels as the sensed modality. For the main task of detecting the soldering workbench in use, participants reused the first rule (light level), applied to a different sensor stream and experimented with additional sensing modalities that were available. Some of our key findings were:

- **Exploration of sensor facilities:** Participants were able to discover the location of sensor nodes and the sensing capabilities they provided. Participants cited the ability to identify their location and orientation with regards to sensor nodes as crucial in locating the nodes. Once they found the sensor nodes, participants experimented by changing stimuli and seeing the corresponding effect on the sensor data streams. Participants without any experience working with sensor nodes were able to understand the stimuli-sensor reading relationship and then used it to draw rules. For example, Task A, participants identified where the dark room was and walked to it, and then once there, switched the ambient light on and off to see the effect on the light sensor output. In Task B, participants experimented with different sensors such as light and audio (e.g., switching on the ambient light, switching on the spotlight, noticing the sound level due to the exhaust).

- **Iteration and experimentation:**
  Participants also experimented with the various sensing modalities available. Most users designed rules using light levels (Task A) and light and sound levels (Task B). Participants experimented with different modalities and combinations (light, motion, light and motion, sound, sound and light, etc). Two participants designed a simple rule that used data from the Spinner node's motion sensor. One of these subjects used this rule in addition to the Light and Sound rule, while the other used it in place of a rule for sound level.
- **Rule reuse:** Eight participants reused the light detection rule that they designed for Task A in Task B as well. In Task B, participants first designed simple rules for light, motion, and sound, then combined them. These observations indicated that most users felt comfortable reusing rules.

C. Results

TABLE II
POST-EXPERIMENT QUESTIONNAIRE RESULTS: USEFULNESS. (LIKERT SCALE, 5: STRONGLY AGREE, 1: STRONGLY DISAGREE)

Usefulness :		
Question:	μ	σ
Made me aware of the sensor network and its facilities	4.2	0.42
Taught me how I could use sensors to infer events in the real world	4.2	0.63
I like being able to remotely observe events using a sensor network	4.1	5.6
Events helped in inferring sensor data	3.6	1.0
I prefer defining rules and observing events rather than view raw sensor data	4.5	0.7
The Ubicorder encourages me to experiment with inferences I can make with sensor network data	4.1	0.73
I would prefer to have someone else define rules for me	2.0	1.0
Gave me ideas for additional sensors to be deployed to sense real-life events of interest	3.9	1.1
I will be more forgiving when events are incorrectly detected because I can modify and tweak the rules by myself.	3.6	0.51

All participants were able to use the system to design rules and define events. Participants with no prior sensor network experience were able to use the network for useful tasks. The rules that they designed spanned a range of solutions. Further, once the participants were familiar with the basic interface of the Ubicorder, the majority of their remaining time was spent exploring the sensor modalities, defining rules and experimenting with rule combinations. Users typically experimented with at least three modalities, often choosing a subset for their final rule. This rapid iteration enabled the participants to define robust rules.

TABLE III
POST-EXPERIMENT QUESTIONNAIRE RESULTS: USABILITY. (LIKERT SCALE, 5: STRONGLY AGREE, 1: STRONGLY DISAGREE)

Usability		
Question:	μ	σ
The interface for viewing sensor data was easy to use	4.3	0.96
The interface for creating and defining events was easy to use	4.1	0.56
The idea of defining events by creating simple and compound rules was intuitive	4.5	0.52
Making compound rules from simple rules was easy	4.8	0.48
Tying multiple sensor values together through compound rules was easy to do	4.6	0.7
I liked being able to see real world events and the corresponding sensor signal at the same time	4.8	0.42

The Ubicorder successfully enabled participants with no prior sensor network experience to discover and explore the deployed sensor networks (μ = 4.2, σ = 0.42, on a 5 point Likert scale). Further, participants, including those without any sensor network experience, were able to learn how to use the sensors to infer real world events (μ=4.2, σ=0.62). Participants also liked the idea of using higher-level rules (μ=4.1, σ=0.7), and found our interface to implement them easy to use (μ=4.1, σ=0.56). In general, participants found it easy to create compound rules from simple rules (μ=4.8, σ=0.48). Participants also liked that the Ubicorder was portable, and that they could design rules *in-situ* (μ=4.8, σ=0.42). Being able to see the real world event and the sensor data at the same time was often cited as the most appreciated feature. The Ubicorder also gave participants ideas about new sensors to add to the sensor network (μ=4.8, σ=0.42) to infer events they might be interested in. From this data, it can be implied that users would take a more proactive role with regards to the sensor network if such a system were to be deployed.

The study was less conclusive about whether participants were more forgiving about false event detections, because they could tweak the rules (μ=3.6, σ=0.51). The question asking if "events helped in inferring sensor data" was polarizing (μ=3.6, σ=1.0). Participants having significant experience with sensor networks and analyzing sensor data wanted to be able to define probabilistic models for inferring events. On the other hand, novices were happy with the current Ubicorder system. Detailed results are presented in Table 1 and Table 2 and are discussed in [3].

D. Limitations And Future Work

Participants identified some key areas for improvement. The Ubicorder currently supports only deterministic combinations of rules, i.e., rules can not be combined "fuzzily". Users typically employ fuzzy rules when using heuristics (equivalent to the 'sometimes' clauses when thinking about sensor rules), thus the lack of this capability limits the expressivity of rules in EDITY.

Participants pointed out the need for categorization of the (previously made) component rules. Currently, all previously rules are displayed in the order in which they were created. One approach would be to categorize rules based on participating sensor nodes, or sensor types. Future versions of the Ubicorder will incorporate this.

There is also a two-second latency (i.e., a delay between when the event happens and subsequently "seeing" resultant changes in sensor data) in the currently implemented system. Most of this delay was from inherent latencies in the data acquisition. For future work, we would like to quantify such delays, and present them to the user in a meaningful way.

We would also like to improve the point-and-browse interface of the Ubicorder. Based on our user studies, we find that users would like some way to definitely ascertain that the node they have physically pointed the device at is indeed the one depicted by the system. We propose using some form of feedback on the sensor node, for example, the user can touch a virtual node and see the corresponding real node light up (LED's are pretty common on sensor nodes). Or the user can make deliberate stimulation of a real node (e.g., shine a flashlight at it) and see the virtual node change.

VI. Related Work

This work was originally inspired by the "Starfleet Tricorder," a fictional device from the popular science-fiction TV series, *Star Trek*. The Tricorder is a handheld device that, when pointed in a direction, scans that area for virtually *any* information [16], [17], then interprets and displays the data. Typical applications included farfetched scanning for "novel life forms" and "energy sources", as well as more mundane readings for radiation levels and atmospheric composition. However, unlike the Star Trek Tricorder, our Ubicorder does not contain all the sensing abilities within the handheld, but instead gleans such data from a locally available sensor network. The Ubicorder is more directly related to several current areas of research, discussed below.

A. Query Languages and Stream Processing

There has been significant research in the systems community in building new query languages and stream processing engines for sensor network applications [18]–[20]. This work centers around themes such as reducing latency, increasing computational and power efficiency, and addressing scalability and robustness concerns. The emphasis of our work lies in presenting a graphical user interface to such a system and thereby making sensor networks more accessible, hence it complements the previous work in this community. The Ubicorder can serve as a front-end for accessing sensor infrastructure. For example, stream processing engines such as Aurora [20] are well suited to processing incoming data and detecting events. The EDITY interface will enable users to easily devise such event conditions (rules).

The Ubicorder needs to acquire sensor data from a network. Acquisition of sensor data through networks is a complex task, with multiple opportunities for optimizing for power and latency. Mueller et al. present SwissQM [21], a virtual machine that presents the sensor network via a single gateway. The system also allows "event rules" to be pushed further down into the network, and optimizes the gateway for a given set of event rules.

B. Sensor Scripting

The idea of scripting simple sensor rules based on a combination of sensor values is well established. First-order logic is traditionally used in the artificial intelligence community for specifying "if-then-else" constructs. Researchers in the context-aware computing space have built several such rule-based systems and associated infrastructure [22]–[24]. In the gesture recognition community, there have been several projects [25], [26] that incorporate a scripting system. These scripting systems typically allow the user to write a text script defining sensor conditions, and combinations thereof. In [26], the authors describe a sensor network scripting system for home automation applications. While the above systems rely on a text based markup language, EDITY, in contrast, presents a graphical user interface on a mobile device, which provides an iterative design-test-analyze functionality. This is particularly valuable, since scripting systems are typically used at first by users who view the data and then, guessing the thresholds, test the rule.

Researchers have previously built graphical sensor scripting sysems, specifically for location-aware applications. In particular, Li et al.'s Topiary [27] allows users to prototype location-aware applications by creating storyboards that describe the interaction sequences (involving people, places and things) on a map. More recently, the Panaromic system [28] also employs a storyboard metaphor for specifying location-based events. Unlike the general-purpose nature of EDITY, such systems target a particular sensing modality, i.e. location. EDITY, on the other hand, is a general-purpose system. We believe that for specific scenarios, a system such as Panoromic might indeed be more convenient for defining very high-level rules, at the cost of generality and expressiveness. We can imagine a scenario where a technically proficient "sensor utility worker" designs primitives using EDITY that are then employed by end-users in a Panaromic-like interface.

The Ubicorder's integrated browsing and EDITY interface simplifies and streamlines this process by visually overlaying the thresholds directly over the sensor data. Additionally, EDITY supports application of rules retroactively, i.e., it allows looking back in time, thereby helping the user drill down to the right rules and thresholds quickly. The Ubicorder's portability means that the user can be physically present at location of the event and observe the ground truth. This makes it significantly easier to correlate sensor data with real world events of interest.

C. Programming by Demonstration Systems

While the Ubicorder does not directly offer a programming by demonstration system [4], [5], it borrows the notion of "events", and of explicitly aiding the user in iteratively crafting rules based on observation (physical phenomena) and effect(sensor data pattern).

The Ubicorder's EDITY system is inspired in part by Hartmann et al.'s Exemplar system [29]. Exemplar is a programming by demonstration system for prototyping sensor interactions for ubiquitous computing systems. Exemplar allows users to perform an action (such as "titling an accelerometer"), then observe and tweak the corresponding sensor (accelerometer) pattern to detect this action in the future. The Ubicorder, like Exemplar, allows user to define events based on sensor signals, but offers significant architectural contributions. First, Exemplar is designed for a single-sensor paradigm, where the sensor node is tethered to a desktop computer with Exemplar's software running. Second, Exemplar does not have any notion of compound rules, component rules and simple rules that allow users to assemble complex rules easily. Third, Exemplar does not support specifying time offsets for combining sensor patterns. Sensor networks encompass the idea that multiple, geographically diverse sensors will enable the specification of features not possible with single isolated sensors, therefore linking sensors across time and space domains is a crucial capability. Fourth, Exemplar limits the number of sensors that can be viewed (eight sensors are displayed in a small-multiples configuration). This does not scale well to sensor networks where large numbers of nodes have to be monitored in parallel. To address this, the Ubicorder includes a sensor network browser. Furthermore, the browser displays data mapped directly onto the corresponding node icons, in contrast to Exemplar's small multiple strip charts.

The context-aware community has a rich tradition of designing tools for programming and prototyping context-aware interactions [23], [24], [30], [31]. In Stick-e Note [23], the concept of *trigger-condition* is discussed - for example, virtual stick-e notes appear based on a users location. iCAP [31] was an early tool for programming context-aware applications, and "a CAPpella" [30] enables users to program such interactions by demonstration. Such systems inspire and motivate our work. First, in [30], the authors make a strong case for empowering end-users to create context-aware applications, the chief suggestion being to utilizing the user's implicit understanding of the environment. The Ubicorder's notion of empowering users to explore and use sensor network data is grounded on a similar argument. Second, [30], [31] states that such capabilities underscore the importance of an *in-situ* system for creating and editing the context-deriving rules. The Ubicorder is also implemented as a mobile device for similar reasons. On the other hand, the Ubicorder is significantly different from these systems. First, EDITY, unlike [23], [30], [31] provides a real-time interactive interface for defining rules. This allows users to quickly establish correlations between sensor data and real-world actions. Second, EDITY has better support for continuous sensor data streams. Third, while [30] and [31] abstract away the raw sensor signals, EDITY allows the user to get "under the hood" of the sensor network by explicitly exposing the raw sensor signals and involving the user in defining the rules.

D. Browsing Sensor Networks

Mobile platforms are often used as sensor network configuration tools. For example, the Great Duck Island project [32] for sensor network habitat monitoring was one of the first systems to use a handheld Personal Digital Assistant (PDA) as a network management tool. Going beyond network management to actual sensor data, in [33] the authors use a PDA to display the availability of nearby conference rooms. Similarly, Maroti et al.'s [34] sniper localization system uses a handheld as an output device, i.e., to display the sniper's location as computed by the system. Although smart mobile phones are beginning to be able to run limited, GPS/vision-anchored augmented reality applications, they generally don't integrate with real-time sensor networks and render mainly cached or web-based data [35]. The Ubicorder's sensor network browsing functionality has directly evolved from the Responsive Environments (ResEnv) Tricorder [11], [36], [37] built earlier by our research team. This is a location, orientation, and network-aware hand-held device used to interface in real-time to a wireless sensor network embedded in a surrounding domestic or occupational environment. Unlike the Ubicorder, the ResEnv Tricorder does not incorporate an inference rule design system such as EDITY. Additionally, the Ubicorder has an improved localization system (using IR) that enables new affordances, such as selecting sensor nodes by pointing at them. Finally, the ResEnv Tricorder was strongly tied to a particular sensor network. The Ubicorder is designed ground-up for heterogeneous sensor networks and supports common metaphors for indicating sensor values.

VII. Conclusions

In this paper, we presented the Ubicorder, a device that lowers the difficulty threshold for users to interact with, customize and utilize the facilities offered by sensor networks. The Ubicorder includes a sensor network browser and the EDITY subsystem to enable users to graphically define higher-level events and script their dependence on sensor network data. In a first-use user-study, both experienced and novice participants rated the Ubicorder highly for its usefulness and usability. Future work will explore porting Ubicorder functions to lighter platforms (e.g., mobile phones and PDA's). We will also explore hybrid applications, where mobile users modify and adjust classifiers that are derived or suggested by machine learning systems running on sensor network data.

Acknowledgment

This work was supported by the Things That Think (TTT) Consortium and other sponsors of the MIT Media Lab. We thank our colleagues in the Responsive Environments Group for their technical help.

References

[1] J. Lifton, M. Feldmeier, Y. Ono, C. Lewis, and J. A. Paradiso, "A Platform for Ubiquitous Sensor Deployment in Occupational and Domestic Environments," in *Proceedings of the Sixth International Symposium on Information Processing in Sensor Networks (IPSN)*, April 2007, pp. 119–127.

[2] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich, "Mobiscopes for human spaces," *IEEE Pervasive Computing*, vol. 6, no. 2, pp. 20–29, 2007.

[3] M. Mittal, "Ubicorder: A Mobile Interface to Sensor Networks," S.M. Dissertation, Massachusetts Institute of Technology, Media Arts and Sciences, Aug. 2008.

[4] A. Cypher, Ed., *Watch What I Do: Programming by Demonstration*. MIT Press, 1993.

[5] H. Lieberman, Ed., *Your Wish is My Command: Giving Users the Power to Instruct their Software*. Morgan Kaufmann, 2000.

[6] B. Hartmann, S. R. Klemmer, M. Bernstein, L. Abdulla, B. Burr, A. Robinson-Mosher, and J. Gee, "Reflective physical prototyping through integrated design, test, and analysis," in *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*. New York, NY, USA: ACM, 2006, pp. 299–308.

[7] S. R. Klemmer, A. K. Sinha, J. Chen, J. A. Landay, N. Aboobaker, and A. Wang, "Suede: a wizard of oz prototyping tool for speech user interfaces," in *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*. New York, NY, USA: ACM, 2000, pp. 1–10.

[8] M. Laibowitz, "Creating Cohesive Video with the Narrative-Informed use of Ubiquitous Wearable and Imaging Sensor Networks," Ph.D. Dissertation, Massachusetts Institute of Technology, Media Arts and Sciences, Jan. 2010.

[9] C. R. Wren and R. Srinivasa, "Self-configuring, lightweight sensor networks for ubiquitous computing," in *International Conference on Ubiquitous Computing (UBICOMP 2003)*, 2003, pp. 205–206.

[10] J. Lifton, M. Laibowitz, D. Harry, N.-W. Gong, M. Mittal, and J. A. Paradiso, "Metaphor and manifestation: Cross-reality with ubiquitous sensor/actuator networks," *IEEE Pervasive Computing*, vol. 8, pp. 24–33, 2009.

[11] J. Lifton, M. Mittal, M. Lapinksi, and J. A. Paradiso, "Tricorder: A mobile sensor network browser," in *Mobile Spatial Interaction Workshop, CHI 2007*, April 2007.

[12] "java.awt class robot."

[13] C. R. Hildreth, *Intelligent Interfaces and Retrieval Methods for Subject Searching in Bibliographic Retrieval Systems.* Cataloging Distribution Service, Library of Congress, Washington, DC 20541., 1989.

[14] D. Young and B. Shneiderman, "A graphical filter/flow representation of boolean queries: A prototype implementation and evaluation," *J. American Society for Information Science*, vol. 44, pp. 327–339, 1993.

[15] B. Morgan and P. Singh, "Elaborating sensor data using temporal and spatial commonsense reasoning," *International Workshop on Wearable and Implantable Body Sensor Networks*, pp. 187–190, 2006.

[16] K. Kleiner, "The star trek tricorder," *New Scientist Blogs*, 2007, accessed: 08/17/2008. [Online]. Available: http://www.newscientist.com/blog/technology/2007/08/star-trek-like-tricoders-in-works.html

[17] Wikipedia, "Tricorder, From Wikipedia, the free encyclopedia," http://en.wikipedia.org/wiki/Tricorder, July 2008, accessed: 07/14/2008. [Online]. Available: http://en.wikipedia.org/wiki/Tricorder

[18] S. Madden, "The design and evaluation of a query processing architecture for sensor networks," Ph.D. dissertation, University of California, Berkeley, 2003.

[19] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tinydb: an acquisitional query processing system for sensor networks," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 122–173, March 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1061322

[20] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik, "Aurora: a new model and architecture for data stream management," *The VLDB Journal*, vol. 12, no. 2, pp. 120–139, 2003.

[21] R. Mueller, G. Alonso, and D. Kossmann, "SwissQM: Next Generation Data Processing in Sensor Networks," in *Third Biennial Conference on Innovative Data Systems Research (CIDR 2007)*, Asilomar, CA, January 2007.

[22] B. N. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Proceedings of the workshop on mobile computing systems and applications*. IEEE Computer Society, 1994, pp. 85–90.

[23] J. Pascoe, "The stick-e note architecture: extending the interface beyond the user," in *IUI '97: Proceedings of the 2nd international conference on Intelligent user interfaces*. New York, NY, USA: ACM, 1997, pp. 261–264.

[24] D. Salber, A. K. Dey, and G. D. Abowd, "The context toolkit: aiding the development of context-enabled applications," in *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 1999, pp. 434–441.

[25] A. Y. Benbasat and J. A. Paradiso, "Compact, configurable inertial gesture recognition," in *Gesture Workshop*, 2001.

[26] T. Haenselmann, T. King, M. Busse, W. Effelsberg, and M. Fuchs, *Emerging Directions in Embedded and Ubiquitous Computing*. Springer Berlin / Heidelberg, 2007, ch. Scriptable Sensor Network Based Home-Automation, pp. 579–591.

[27] Y. Li, J. I. Hong, and J. A. Landay, "Topiary: a tool for prototyping location-enhanced applications," in *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*. New York, NY, USA: ACM, 2004, pp. 217–226.

[28] E. Welbourne, M. Balazinska, G. Borriello, and J. Fogarty, "Specification and verification of complex location events with panoramic," in *Pervasive*, ser. Lecture Notes in Computer Science, vol. 6030. Springer, 2010, pp. 57–75.

[29] B. Hartmann, L. Abdulla, M. Mittal, and S. R. Klemmer, "Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition," in *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 2007, pp. 145–154.

[30] A. K. Dey, R. Hamid, C. Beckmann, I. Li, and D. Hsu, "a cappella: programming by demonstration of context-aware applications," in *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 2004, pp. 33–40.

[31] T. Sohn and A. Dey, "icap: an informal tool for interactive prototyping of context-aware applications," in *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*. New York, NY, USA: ACM, 2003, pp. 974–975.

[32] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. New York, NY, USA: ACM, 2002, pp. 88–97.

[33] W. S. Conner, L. Krishnamurthy, and R. Want, "Making everyday life easier using dense sensor networks," in *UbiComp '01: Proceedings of the 3rd international conference on Ubiquitous Computing*. London, UK: Springer-Verlag, 2001, pp. 49–55.

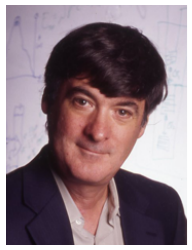
[34] M. Maroti, G. Simon, A. Ledeczi, and J. Sztipanovits, "Shooter localization in urban terrain," *Computer*, vol. 37, no. 8, pp. 60–61, 2004.

[35] A. L. Liu, H. Hile, H. Kautz, G. Borriello, P. A. Brown, M. Harniss, and K. Johnson, "Indoor wayfinding:: developing a functional interface for individuals with cognitive impairments," in *Assets '06: Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility*. New York, NY, USA: ACM, 2006, pp. 95–102.

[36] J. Lifton, "Dual Reality: An Emerging Medium," Ph.D. Dissertation, Massachusetts Institute of Technology, Media Arts and Sciences, Sept. 2007.

[37] J. Lifton and J. A. Paradiso, "Dual Reality: Merging the Real and Virtual," in *Proceedings of the First International ICST Conference on Facets of Virtual Environments (FaVE)*, July 2009.

Manas Mittal Manas Mittal is a software engineer at Intuit Inc (Mint.com). His research interests include sensor networks, ubiquitous computing and HCI. He has a SM from the Massachusetts Institute of Technology Media Laboratory and a BE from NSIT at Delhi University. Contact him at manas@media.mit.edu



Joseph Paradiso Joseph Paradiso is an Associate Professor of Media Arts and Sciences at the MIT Media Laboratory, where he directs the Responsive Environments group, which explores how sensor networks augment and mediate human experience, interaction, and perception. In addition, he co-directs the Things That Think Consortium, a group of industrial partners and Media Lab researchers who explore the extreme fringe of embedded computation, communication, and sensing. After receiving a BS in electrical engineering and physics from Tufts University, Paradiso became a K.T. Compton fellow at the Lab for Nuclear Science at MIT, receiving his PhD in physics there for research conducted at CERN in Geneva. He is a senior member of the IEEE and AIAA, and a member of the APS, ACM, and Sigma Xi.