Reminding about Tagged Objects using Passive RFIDs

Abstract. People often misplace objects they care about. We present a system for generating reminders about objects left behind by tagging those objects with passive RFID tags. Readers positioned in the environment frequented by users read tags and broadcast the tags' IDs over a short-range wireless medium. A user's personal server collects the read events in real-time and processes them to determine if a reminder is warranted or not. The reminders are delivered to a wristwatch-sized device through a combination of text messages and audible beeps. We believe this leads to a practical and scalable approach in terms of system architecture and user experience as well as being more amenable to maintaining user privacy than previous approaches. We present results that demonstrate that current RFID tag technology is appropriate for this application when integrated with calendar information.

1 Introduction

One of the promises of ubiquitous computing is that it will make our information systems proactive, that is, information will be made available, as we need it, rather than having to request it explicitly. To accomplish this, it is important to have a sense of the user's context which can be defined quite broadly, including such disparate elements as: location, ambient temperature, heart-rate, sound level, other people that may be nearby, the activity the user is engaged in, the task they are trying to complete, etc. [13, 14].

In this paper, we describe our work in implementing a simple proactive application that reminds us of objects we may have mistakenly left behind as we go about our day. It uses the user's location, calendar, and the objects they are carrying as the context. The reminding engine takes as input a set of static and dynamic rules and checks, that the objects that should be with a person at a given time are, in fact, present. When they are not present the system provides an appropriate alert to the user so that he can retrieve the objects left behind while it is still relatively easy to do so. An example is a lawyer reminded to return a legal brief to their law firm when leaving home in the morning. If the papers are left behind, the alert needs to be delivered to the user before they are too far from home and it is still efficient for them to go back and retrieve the brief.

Reminding is an application that has wide appeal [18, 19]. Recently, as part of a class on context-aware computing we conducted a short survey of approximately 30 members of our department (graduate students and faculty) and found that 75% lose things occasionally [1]. The items most commonly left behind included notebooks, pens, and water bottles with over 50% of respondents saying they forgot to take one or more of the things they needed for a meeting or a class at least once a month. Keys and cell phones were next in line as the most common items left behind. Laptops and PDAs were not as commonly forgotten as the other smaller and less expensive items. Respondents were also concerned about how they would be reminded, as over 75% preferred non-intrusive methods such as e-mail rather than pages or cell phone calls. However, they were not asked if they would want an immediate reminder such as our system provided.

From this and previous work in reminder systems, we find that the following are requirements for a proactive reminder system:

1. it should have the ability to keep track of a large number of small and inexpensive items,
2. reminders should not be too disruptive and must be kept to a minimum,
3. knowing where the user is headed, as opposed to only knowing where they are, is an important element of context needed to keep false reminders to a minimum,
4. it should be relatively straightforward for a user to add reminder rules to their application with minimal programming required for common situations, and
5. the system needs to be incrementally deployable and easy to maintain.

We used these five requirements to formulate our approach and the design of our system.

To address the first requirement, we use passive RFID tags that are inexpensive (currently on the order of less than $.50 per tag with the price dropping precipitously as RFID tags are proliferating in supply-chain management) and can be easily affixed to the objects we care about tracking. They do not require batteries; making them virtually free of on-going maintenance costs.

For the second requirement, we use a wristwatch to display two levels of reminders: warnings and alerts. A warning simply adds an item to a list displayed on the screen of the watch. These are intended to be items the user can easily keep in her consciousness by simply glancing down at the wristwatch. Think of these as the equivalent of tying a string around a finger. When a reminder is more certain, the watch will audibly beep to get the user's attention and display the specific items that appear to be missing. These more forceful reminders are more appropriate when a user is about to leave a location and the cost of leaving something behind is high.

For the third requirement, we provide our reminder system with calendar information to better determine the user's possible destinations when they are leaving a location. In the future, we will rely less on calendar information and integrate our system with a wide-area positioning system that can provide definite information about whether a user is entering or exiting an area rather than just being in the presence of an RFID reader.

For the fourth requirement, we have chosen to implement the concept of an auto-tag, a tag with predefined usage. For example, a user could tag an item at work with a

special auto-tag so that if it's taken home, the system reminds him the next morning to bring it back to work. Auto-tags have a predefined home base so that they will automatically activate when leaving a location. Additionally, auto-tags are registered to a user so that they are not confused with other users' tags.

Finally, for the fifth requirement, we have the RFID readers broadcast the tags they read over short-range radio for nearby users to hear. One important advantage of this feature is the freedom from connecting the readers to the networking infrastructure and a central database. Instead, the user's personal server receives the transmissions from nearby readers and maintains its own local database with no need to discover a database service or deal with inter-domain authentication and security issues. Additional readers are easy to deploy as only their location needs to be recorded (we will describe later how we plan to make that automatic as well) and the overall system is scalable as there is no central coordination or registration. Because of these properties, it is also a system that can be deployed by a single consumer who can install RFID readers in the places they commonly frequent (e.g., home, work, car, etc.). Although prices for the RFID readers we used are about the same as a PDA or laptop, these are expected to fall dramatically with economies of scale.

In summary, our approach has the following combination of features that distinguish it from related work:

- passive RFID tags that require no maintenance,
- broadcasting RFID readers that make the system incrementally deployable,
- an unobtrusive user interface that minimally distracts the user, and
- auto-tags to program common behaviors.

We believe this combination represents important steps toward a practical and usable reminder application.

The remainder of the paper is structured as follows. In the next section, we describe related work in reminder systems and compare them to our approach. In section 3, we present the properties of RFID tags and tag readers and discuss some of their limitations and specifically focus on the issues of broadcasting read events. Section 4 describes our reminder application and its user interface in detail. Section 5 describes our experiments in validating the use of RFID tags as well as a complete reminder scenario. Section 6 provides a discussion of the issues we've uncovered and outlines our future work to resolve these issues and create a truly usable reminder system. Finally, Section 7 summarizes the paper and draws some preliminary conclusions.

2 Related Work

Technology aids for helping people remember what they need to do have been around for quite some time (see [2] for a good survey and [18, 19] for specific applications). In the era of personal computing, most of the work has led to a variety of desktop and palmtop applications that focus on the *when* of a reminder. User interfaces for specifying all forms of recurring events are now standard on PDAs and popular email programs.

More recently, as embedded systems consisting of wireless sensors and wearable devices [17] have become more practical, attention has begun to shift toward reminders for physical objects, or the *what* of a reminder. An early example of this is the CybreMinder system [3]. In CyberMinder, user context (defined in terms of location, username, sound level, or, even, stock price) is used to trigger a reminder message to be delivered to the user. Several delivery methods are supported including wearable user interfaces and printing of paper to-do lists. The CyberMinder system selects the most appropriate method to use. Although CyberMinder added direct context sensing to the desktop reminder applications, it was still a reminder system focusing on information rather than physical objects.

In line with the current work in integrating the physical and virtual worlds, SPECs take reminders a step further by making it possible to remind people about physical objects as well [4]. With SPECs, each object of interest is tagged with a bi-directional infrared sensor package. SPECs beacon their ID over the IR medium to be picked up by other SPECs and SPEC base stations. By collecting this information, reminders can now be based on the presence or absence of particular objects. Discrimination based on location is easy to achieve by simply affixing a SPEC at a particular location. One of the main issues with SPECs is the requirement of line-of-sight between IR transceivers. This makes it impossible to place a tagged object in a bag or pocket and could cause many proximity events to be missed because the IR transceivers on two SPECs do not line up.

Wireless sensor nodes have been used to construct a radio-frequency version of SPECs [5]. As with SPECs, two objects are said to be near each other if they can communicate, not necessarily bi-directionally. Each sensor node records the times at which it hears RF packets from other nodes. This data can then be mined by a base station to generate reminders or study work-flow patterns. Because proximity events are recorded in parallel by all nodes, the clocks of the nodes need to be kept synchronized to an appropriate accuracy for human motion.

The problem with both SPECs and their radio counterparts is that they are active devices. They have batteries that may last months but will eventually need to be replaced. Their cost consists of a small microcontroller and memory, a radio or IR transceiver, and a battery; although this cost is admittedly small, it is still prohibitive for tagging large numbers of objects. Moreover, if too many are in a small space they may saturate the limited communication bandwidth available.

Another issue with these approaches is the lack of a central repository where the proximity information can be processed to determine if a reminder must be issued. Proximity events are stored in each of the nodes and only occasionally connected by base stations that then need to forward it to the appropriate repository and user interface.

Our approach is distinguished from these in several aspects:

- we use passive RFID tags to tag objects because they are cheap and will soon even be printable onto paper [16],
- RFID tags do not require batteries and are spared associated maintenance issues,
- RFID readers can act as efficient sensors for a large number of tags and broadcast the information to all interested parties in parallel, and

- a single repository, the user's personal server [9], can collect and process the information relevant to each user.

In addition to these differences with prior work, our approach is as amenable to privacy protection and can benefit from the same reminder management and user interface. SPECs provide a language for specifying reminders. However, it requires the user to spell out the rules for each object very explicitly making for a laborious process each time one wants to add a new object to the collection being tracked. In contrast, we introduce the concept of an auto-tag, where the tag itself specifies the reminding behavior for its associated object.

3 RFID Tags and Readers

Radio-frequency identification tags are a rapidly evolving technology. They were developed to make supply chain management more efficient. In the basic technology, a reader antenna induces enough power to operate a tag's integrated circuit through its antenna. The tag electronics radiate back to the reader and modulate the signal corresponding to the identification number of the tag [6].

In recent years, many advances have been made in RFID technology. Among these are: the ability to read multiple tags within range of the reader antenna, longer range tags through better antenna design and lower-power tag circuitry, increased storage capacity in the tag to represent other information besides ID, and the ability to write the tag memory with new information. Our work leverages all of these advances.

The ability to read multiple tags is essential as we expect users to carry many tagged devices at any one time. Longer read range is important so that readers can be placed in entryways and corridors and read the tags that pass by without requiring a conscious action on the part of the user. Large storage capacity in the tag allows us to add more identifiers and, eventually, will allow us to include code for the reminder system within the tag itself (we call this code a *taglet*). The ability to write all or part of the tag's data allow us to build a more privacy-friendly system with capabilities that can be more easily personalized to each user.

One real-world limitation of RFID readers is that they can occasionally fail to read a tag that is present. This occurs because of interference from other objects, especially human bodies with their high water content, that absorb RF energy. Another reason for missed reads is that the tag reading protocol has trouble with tags that have their antennas very close to each other. Therefore, our system must be designed with false negatives in mind. False positives, where the reader reports a tag as being read when it is not really present, are nearly impossible and we therefore do not consider them. Of course, an attacker could spoof a tag and have it respond with another tag's ID. However, the space of tag IDs is huge (often 128 bits but only 64 bits in our case) making this possibility quite remote.

In our implementation, we use Alien tags and readers [7]. More specifically, we chose the ALL-9250 Alien tag (S1800002-001), also known as the I2-tag and shown in Figure 1 that operates at 915MHz. We found these tags to have a read range of 5-10 feet although they are quite sensitive to proximity to human bodies. However, they were the best (84% success rate) of the four types of Alien tags (the worst was 68%).

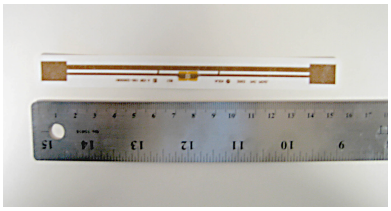


Figure 1. Example of an Alien ALL-9250 915MHz long-range RFID tag (the I2-tag). The tag intergrated circuit is at the center of the tag and connected to an antenna that is approximately 15cm long.

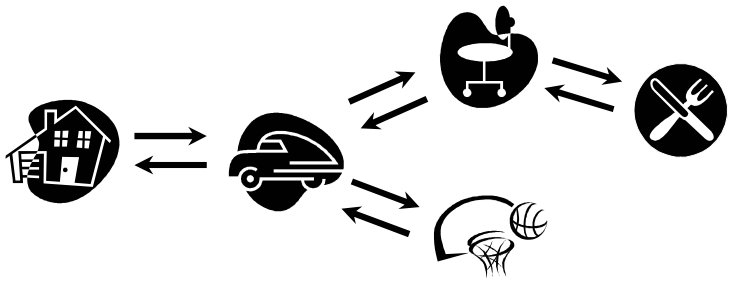
Our RFID readers broadcast the tags they read to anyone in the vicinity (much in the spirit of [15]). We use UC Berkeley sensor motes [8] operating at 433MHz to realize this broadcast. There is a mote attached to each reader and a mote attached to each user's personal server to receive the broadcast. The motes use a low-power connection-less radio protocol that transmits data at only 19.2kbps but does not require the several seconds of discovery time required of such protocols as Bluetooth. However, this is very limiting bandwidth that we will eventually replace with WiFi, but for our purposes, dozens of tags' IDs can be broadcast in less than a second. The range of the radio is an order of magnitude greater than the tag reader's range and consistently reaches 10-15 meters and often much more. This longer range allows ample time to ensure that the personal server of the user receives the transmission from the reader.

Each tag ID consists of 64 bits. Assuming that the reader reads up to 25 tags at a time, it will take a mote less than 1 second to transmit this data. With human motion usually not exceeding 3m/sec, this translates into the user being no more than 3 meters away from the reader, which is considerably less than the range of the mote radios. Because the data is broadcast, we are not concerned with the number of users that need to receive the data but, rather, only with the total number of tag IDs that need to transmitted. This makes the approach quite scalable with RFID reader technology.

4 The Reminder Application and its User Interface

We begin the detailed description of our system with a typical reminder scenario. Our example subject is one of the authors, a male with a 9-5 job Monday through Friday. His typical work day follows the pattern of going from home to work, work to lunch, back to work, work to home, home to the gym, and back home on most days. We tagged all items he deemed important with appropriate tags and took a day and a half long period from his week as the focus for the experiments we report in the next section. RFID readers are present at each of the locations. Readers for home and car could be purchased as appliances. Readers at work may be purchased by the employer and placed at entrances to the building and/or purchased by employees and installed at their desks. Readers at local restaurants and gyms are likely to be

purchased by those businesses to help extend an additional convenience service to their customers. Figure 2 has a schematic for these five locations and lists the items needed at each as specified by the user.



8:00		Leave for work: **keys, wallet, phone, jacket, backpack, keycard**
8:00	- 8:30	In car: **keys, wallet, phone, jacket, backpack, keycard**
8:30		Arrive at work, park car
8:33		Enter building: **keys, wallet, phone, jacket, backpack, keycard**
8:35		Arrive at desk: **keys, wallet, phone, jacket, backpack, keycard**
8:35	- 12:00	At work: **keycard**
12:00	- 13:00	At lunch: **keycard, phone, wallet**
13:00	- 17:00	At work: **keycard**
17:00		Leave work: **keys, wallet, phone, jacket, backpack, keycard**
17:00	- 17:45	In car: **keys, wallet, phone, jacket, backpack, keycard, docs (auto-tag)**
17:45		Arrive home: **keys, wallet, phone, jacket, backpack, keycard, docs (auto-tag)**
19:00		Leave for basketball
19:00	- 19:15	In car: **keys, wallet, phone, gym bag**
19:15		Arrive at gym: **keys, wallet, phone, gym bag**
21:00		Leave for home
21:02		In car: **keys, wallet, phone, gym bag**
21:17		Arrive home: **keys, wallet, phone, gym bag**

Figure 2. Usage scenario for a typical day for our test subject. RFID readers are present at each of the five locations represented by the icons at the top of the figure. Each time interval is detailed with the items carried between the locations and the most likely position of the items on the person.

There are few things to note in this scenario. While at work, the only item needed on the subject at all times is the keycard. Every other item can stay in the office. When leaving a location (for lunch, home, or anywhere) the wallet and phone will always be needed. The subject will likely pass a reader in his office many times throughout the day, so a simple way to tell if he is leaving work is to have a reader near the exit of his building. The concept of such location boundaries introduces a need for determination of user direction. Thus, integration with a wide-area positioning system may be important. We'll revisit this issue in Section 6.

In our scenario, the subject needs to return items to the locations from which they were originally taken. For instance, items taken from home in the morning (like a jacket) should return home at the end of the day. Similarly, items brought home from work will probably need to be brought back the next day. An instance of this concept

is captured in our scenario when the subject tags a legal document with an auto-tag in order to remind himself to bring it back to work on the following day.

There may, of course, be many variations to the routine presented here. For example, in the morning he may bring his gym bag with him so that he can go straight to the gym after work. In another example, the subject brings his own lunch, so he doesn't have to leave the building for a restaurant. On those days that he brings his lunch, the system should remind him to return his lunch container back home at the end of the day. Figure 3 shows an example of a user arriving at a location.

Figure 3. A user walking through a doorway with several tagged objects. An RFID reader is visible to the left (white box on black stand); tags are visible on the notebooks in his hand; his personal server is in his front left pants pocket; and, our wristwatch UI is on his left wrist (images of the personal server and wristwatch are in Figure 4).

Reminder Specification Language

We wrote a custom language to describe reminders. In the future, we plan to expand our reminder creation system with a graphical UI that enables users to query, sort, and edit their reminders. This graphical UI would simplify configuration with desktop RFID reader integration. Users could then easily scan tags and manipulate reminder semantics without having to type tag IDs.

We have developed a very simple, yet powerful, language to help users express reminders. The language is built around two basic constructs. First, the keyword 'object' allows the user to specify symbolic names for objects rather than forcing the user to remember specific tag IDs. The syntax is 'object name(tagID)', where 'name' is a user-entered name, and 'tagID' is the RFID tag ID of the associated object. Once the user has specified his objects, he can begin to employ the second language construct, 'reminders'. An example follows:

```
reminder (day=tuesday, destination=work, location=home, starttime=12:00,
           endtime=1:00, items=[keys, wallet])
```

With the exception of 'items', all of the keywords for the 'reminder' construct are optional. Each unspecified option means "always." This lets us build a wide variety of expressions with fairly little effort. Since our system uses predictive algorithms, users do not need to enter specific times or even ranges of times, but they can do so if they prefer to be more precise. Leaving out the 'day' keyword indicates every day. By including location but not destination, we indicate "from location L to anywhere." When we leave out location but include destination, we indicate "from anywhere to destination D." If we leave out both location and destination, we indicate "always present." Furthermore, if both location and destination are the same, it means "always present at the specified location." Choosing to not specify a start time and including an end time indicates "before this time." Similarly, by leaving out an end time and including a start time, we indicate "after this time." The user can employ all of these semantics in order to represent all possible situations for which our system presents reminders. An example set of reminders for our target scenario follows:

```
#items to bring to work
reminder (destination=work, items=[phone, keys, wallet, backpack,
           jacket, keycard])

#items I need at lunch
reminder (location=work, starttime=11:00, endtime=2:00, items=[keycard,
           phone, wallet])

#items I always need with me at work
reminder (location=work, destination=work items=[keycard])

#items to bring home from work
reminder (location=work, destination=home, items=[phone, keys, wallet,
           backpack, jacket, keycard])

#items I need any time I go to the gym
reminder (destination=gym, items=[gymbag, wallet, phone, keys])
reminder (location=gym, destination=home, items=[gymbag, wallet, phone,
           keys])

#auto-tag associated reminder
reminder (destination=work, items=[autoTag1])
```

The last reminder in the list above is automatically generated by the presence of the auto-tag. As the auto-tag has been pre-registered by the user to be associated with his work location, it will simply trigger a reminder if the item is not returned to work the next time the user goes there. Auto-tags are pre-registered by a procedure that uses an RFID reader at home or at work to write the reminder into the tag itself and register the tag ID in a database on the user's personal server.

Reminder Application Implementation

Our reminder software runs on the user's personal server [9], shown on the right in Figure 4. The personal server is a small embedded server that runs the Linux operating system, our application software, and stores relevant user information. Therefore, it can interact with a wide range of other devices throughout the day (e.g., a personal calendar). The reminding software consists of several concurrent, communicating modules. A reminder control module constantly evaluates the set of reminder rules against the current context. A location module infers the current location from various input sources and maintains a list of possible user destinations. Finally, a radio module listens for beaconing RFID readers and wirelessly communicates warnings and alerts to the wristwatch user interface (see the left half of Figure 4). Each component accesses a SQL database on the personal server to load programmable rules and parameters as well as store persistent state (e.g., past RFID tag read events).

When the radio module receives a packet broadcast from a nearby RFID reader, it parses the packet and sends both the reminder control and location module the list of tags, if any, seen by the reader. While the reminder control component is interested in both which reader was broadcasting and which tags were seen, the location component cares only about which reader. This is why readers periodically broadcast independently of whether they have read any tags or not. This helps the application keep track of locations even in the presence of false negative reads. The radio module also manages outgoing communication to the wristwatch.

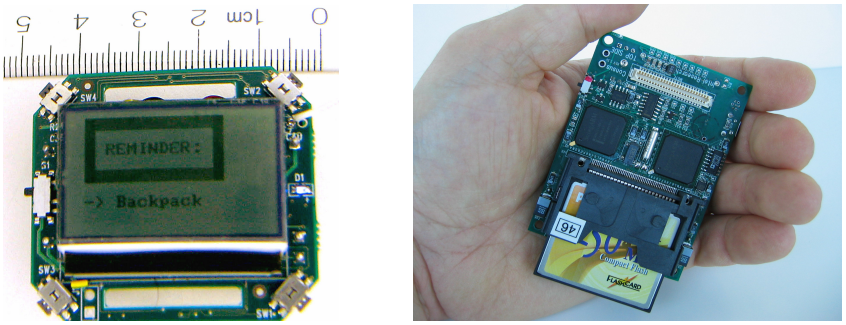


Figure 4. Larger views of our wristwatch (showing a reminder alert) and personal server. Note that the personal server has WiFi communication capability in addition to the mote radio.

The location inference component determines the current location of the user from known locations of the RFID readers 'seen' most recently (later we'll explain how we plan to generalize this). The module then cross-references this determination with the user's calendar data in order to construct a list of possible future destinations that will be visited before returning to the current location. Together, the destination list and current location are known as the user's 'location context.' Whenever this context changes, the location module informs the reminder control unit.

Finally, we come to the core of the reminding software: the control component. Within this module, there are three important tasks to be managed: maintenance of the

user's 'item context,' which consists of those items currently with the user; reminder rule generation and evaluation, which may lead to the generation of warnings or alerts; and personal item tracking.

At this time, the implementation of item context is very simple. If a reader has seen an item, then it is part of the context. If we pass a reader that does not see the item, the item is lost from the context. To better cope with false negatives, deployments with numerous redundant RFID readers would be advantageous. In such situations, more complex implementations might be able to "smooth" the item context over multiple, successive reads by accounting for the probability of false negatives. At this time, however, we assume a sparse set of readers, so such smoothing is not included in our prototype.

The control module constantly reevaluates all relevant rules as the user's item and location contexts change. If it is clear from the context that a rule has been violated, the component will trigger a reminder. Depending upon the severity of the consequences of a missed reminder, the reminder either takes the form of an audible alert or a non-intrusive, inaudible warning. Currently, severity is simply a function of whether the location context has changed, and audible alerts are raised when this is the case. Otherwise, the item in question is simply added to the top of the wristwatch's 'consciousness list' interface, an alternate display on the wristwatch.

Finally, the control component tracks the last known location of tagged items. This is done both as an independent service to the user (i.e. "where did I leave my keys?") and as an important input for intelligent decision-making within the rule evaluation system. An example will assist in understanding the importance of the latter: our user usually leaves work and heads home and, after picking up his gym bag, leaves home for the gym. When the user leaves work, the gym is in the user's possible destination list, but the user's gym bag is at home. Since the reminding software last saw the gym bag at home, it is able to infer that the user is probably going to pick up the gym bag at home. Even if this prediction is incorrect, delaying an alert until the user leaves home is both practical and intelligent; after all, the user cannot even remedy the missing item until he reaches home.

We also use the concept of a safe zone to limit the issuance of an alert or warning until the time nears that a user is likely to leave a destination. Basically, users can create personal zones, such as home, car, or office, where any item can be left without warnings being issued.

5 Experiments

We conducted experiments on our underlying RFID technology as well as the reminder application itself.

Initially, we wanted to determine which of the several tags available to us would yield the best results. We configured our reader in a test setup that would minimize the effects of the subject's direction and orientation by placing the antennas on both sides of the likely walking path at slightly different angles as shown in Figure 5. We found this configuration to be the most reliable of those tested. Note that both antennas are connected to a single reader that multiplexes between the two so there is

only a small incremental cost to the two-antenna configuration (antennas are much cheaper than readers).

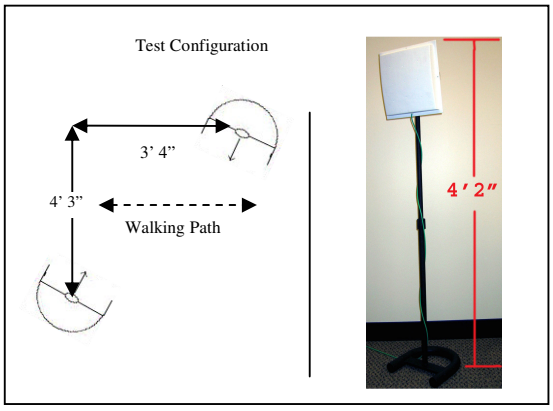


Figure 5. Larger views of an RFID reader antenna on a stand. The white box that houses the antenna is approximately 30cm square. Two antennas are arranged as show around a walking path to minimize missed tag reads when the user moves in either direction.

To determine the efficacy of the RFID technology and the rate of false negatives that we could expect, we tagged all the objects in the scenario with the most successful I2-tag and performed many trials under different conditions to see whether the reader would correctly detect the tag IDs. All tags were placed directly on the items' surface with the exception of the laptop. Due to the high metallic content of the laptop, we found that the success rate improved dramatically when we placed a 0.125-inch thick piece of packing foam between the surface of the laptop and the tag (not an unreasonable modification, moreover there are specific tags designed to work on metal objects [10]). Table 1 shows the percentage of the time that each object's tag was successfully recognized by the RFID reader. The different trials included many passes by the reader's antennae (5 times in each direction for a total of 10 trials for each scenario) with the object placed in a plausible carrying position (e.g., notebook in a backpack or cell phone in a pocket). Attempts were made to minimize the effect of item location in this experiment. For instance, no items were placed directly against the subject's skin (for reasons explained later), nor were any items placed within an extremely close proximity to one another. The number of test scenarios in which each item was tested is listed in the 'Number of Scenarios' column.

We also performed experiments to assess tag reliability when items are carried in different positions or arrangements. Table 2 shows the percentages of successful (I2-tag) detections for a variety of cases. Items were placed on the subject as dictated by our practical test scenarios. The success rates for each category were calculated using the cumulative averages across all of these scenarios. The number of test scenarios for which each location was tested is included in the 'Number of Scenarios' column. The 'Person' category includes tagged items on a jacket, a gym bag, a cloth bag, and a

backpack. It is important to recognize that for 'Person', a substantial amount of material separated the subject's skin from the object's tag. A more indicative reflection of success rates when tags are touching the skin can be found in the 'In Hand' category.

Table 1. Success rates for reading tags affixed to the various items in our example.

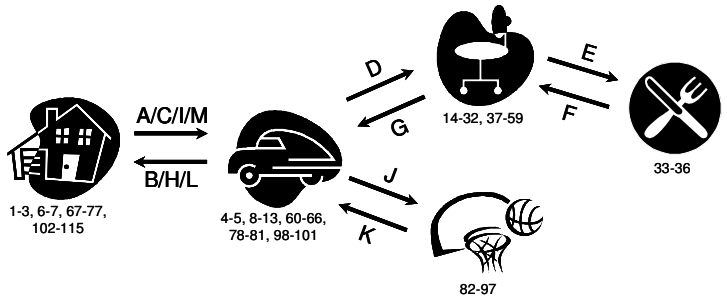
Item	Success Rate (cumulative)	Number Of Scenarios	Number Of Trials
Jacket	100%	1	10
Backpack	100%	1	10
Keys	65%	3	30
Cell phone	47%	3	30
Wallet	93%	3	30
Keycard	55%	3	30
Laptop	80%	1	10
Papers	100%	1	10
Gym Bag	100%	1	10
Cloth Bag	100%	1	10
Lunch container	100%	2	20

Table 2. Success rates for reading tags when the objects are carried in different positions.

Item Location	Success Rate (Cumulative)	Number Of Scenarios	Number Of Trials
Person	100%	5	50
Jacket Pocket	95%	2	20
Jean Pocket (back)	100%	2	20
Jean Pocket (front)	38%	5	50
Backpack	93%	3	30
In Hand	23%	1	10
Gym Bag	73%	3	30
Cloth Bag	100%	1	10

The second set of experiments focused on our reminder software implementation. The first experiment tested our reminding system by giving it our target scenario as input (see Figure 2 and the reminder code of section 4).

Figure 6 summarizes our scenario simulation results (we use only a partial trace log due to space limitations). The location transitions are alphabetically labeled by simulation chronology, and the likewise ascending number ranges associated with each location correspond to atomic groups of read events (e.g., 1-3 represents 3 reads performed by the reader at home, each consisting of multiple tag IDs (refer to the simulation log for details). One may follow the user throughout his day by starting at home, following the transition A to the car, returning to home on B, and so on.



Current location: home – Possible destinations: car
Read #2: keys, wallet, jacket, backpack, and keycard
Current location: car – Possible destinations: work, lunch (transition A)
****Alert issued for phone as location was changing.**
Current location: home – Possible destinations: work, lunch, car (transition B)
Read #7: phone
Current location: car – Possible destinations: work, lunch (transition C)
Current location: work – Possible destinations: lunch (transition D)
Read #21: no keycard
****Alert issued for keycard.**
Read #22: keycard
Read #30: no jacket, no keys, no backpack
Current location: lunch – Possible destinations: work, car, home, gym (transition E)
Current location: work – Possible destinations: car, home, gym (transition F)
Read #41: keys, jacket, and backpack
Read #57: auto1-tag
Current location: car – Possible destinations: home (transition G)
Current location: home – Possible destinations: car, gym (transition H)
Read #70: no jacket, no auto1, no keys, no backpack, no keycard, no wallet, no phone
Read #74: keys, wallet, phone, gym bag
Current location: car – Possible destinations: gym (transition I)
Current location: gym – Possible destinations: car, home, work, lunch (transition J)
Read #84: no gym bag
****Warning issued for gym bag.**
Read #94: gym bag
Current location: car – Possible destinations: home (transition K)
Current location: home – Possible destinations: car, work, lunch (transition L)
****Alert issued for auto1 as location was changing.**
Read #104: no keys, no gym bag, no wallet, no phone
Read #109: keys, wallet, phone, jacket, backpack, keycard
Current location: car – Possible destinations: work, lunch (transition M)
****Alert issued for auto1 as location was changing.**

Figure 6. A partial trace log for a typical day of our usage scenario. The numbers under the location icons correspond to the read events recorded at each location. The transitions are labeled in the figure for cross-reference to the log.

In short, the user leaves home but forgets their cell phone, receives an alert about the missing phone as he reaches the car, and returns home to retrieve the missing item. Now carrying all his important items, he returns to the car and drives to work. While working, he accidentally leaves his keycard behind and receives a prompt alert

to that effect at the next reader he passes. The user retrieves the keycard and continues to work until lunch, at which point he intentionally leaves his jacket, keys, and backpack in his office and, on foot, leaves the building for lunch. When he arrives at lunch, the reminding software does not remind him of the missing jacket, keys, or backpack because it infers that he will return to work before going home without the items.

After his meal, he returns to work and eventually places a 'Return To Work' auto-tag on a paper that he will revise at home that evening but must remember to return the next day. When he leaves work for home, the auto-tag is recognized and its corresponding code is activated in the reminding software. He then spends some time at home, drops a few unnecessary items, picks up his gym bag, and heads to the gym. At the gym, he leaves his gym bag in a locker, and the reminding software immediately adds a warning about the gym bag to the wristwatch's consciousness list. Afterward, he picks up the gym bag, heads home, and when he arrives, the software recognizes that work will again soon be a valid user destination and reminds him about the papers that must return to work (in the future, we hope to do without this reminder and only issue it on the next day that the user will be heading to work). To further illustrate the auto-tag mechanism, our scenario includes the very start of the following day. The user leaves home in the morning with everything but the tagged papers. As a result, an alert is issued as the user gets into his car, thus concluding our scenario.

To further understand the effect of missed reads, we created a simulation sequence based on our test scenario where all the tags were present at the correct times and location (unlike our previous example where the user did leave some things behind). Our simulator randomly deleted individual tags from the read events to simulate the effect of false negatives. The averaged results for 30 independent sets of data are show below in Table 3. Notice that there is one warning issued in the 0% drop case. This warning occurs when the subject leaves his gym bag in the locker at the gym (a non-safe zone). According to the reminders that were setup, the user will need to take the gym bag home so the system is issuing a warning in case the user accidentally left it behind.

Table 3 shows that our reminder implementation works reasonably well even when 10% of possible tag reads are missed. From our experiments we believe that the most likely value of missed reads will be somewhere between 5% and 10% for the RFID technology we are using. The number of warnings quickly reaches a plateau as the number of false negatives increases. Again, the warnings do not interrupt the user but simply add items to the wristwatch's consciousness list. The number of alerts is more significant and this appears to grow linearly with the drop rate. Even though the system is detecting possible problems, the great majority cause warnings rather than alerts to be issued. Alerts are only issued when it is important that the user notices the current situation as there may be a larger cost associated with retrieving the item. The results in Table 3 were generated with no smoothing algorithms; however, initial testing with smoothing gives us confidence that we can significantly decrease the number of warnings and alerts that are issued mistakenly even further.

Table 3. The number of warnings and alerts increase with an increasing number of false negatives, but only moderately so.

Drop Rate	Warnings	Std. Dev	Alarms	Std. Dev
0%	**1**	0.0	**0**	0.00
5%	**13**	3.6	**3**	1.8
10%	**24**	3.8	**6**	2.2
15%	**33**	4.8	**10**	2.8
20%	**41**	5.3	**12**	2.9
25%	**48**	5.5	**15**	2.8
30%	**54**	4.7	**17**	2.7
35%	**58**	4.7	**19**	2.4
40%	**61**	4.1	**21**	3.4

6 Discussion

In constructing our reminder application, we encountered several important issues that merit further discussion as they can limit the practicality of our system.

Our first set of challenges revolves around the use of location information. Ideally, you would like the system to issue an alert only when the user is about to take an action that will increase the cost to go back and retrieve the item. For the reminder to be issued at the appropriate time, the system needs to know when the user's location is changing. One possible solution is to specify which reader is near a doorway and if detected the system knows the user is at the boundary and is about to change locations. However, detection of a boundary reader does not always accurately indicate a location change because if the user is only walking by the door and the reader picks up the user, the system may erroneously assume the user is leaving and issue an unwanted reminder. For example, you might walk outside to get something off your porch and walk back inside. This might look the same to the system as you are leaving for work. Ideally, you would want the system to recognize when you are entering or exiting a location and issue the proper reminders only when it is sure you are changing locations. Due to the system's inability to distinguish entrance from exit, reminder rules must be evaluated in terms of more general location-change events.

To remedy this problem, we plan to integrate Place Lab [11] WiFi-based localization into our system. This will enable us to detect when the user is leaving a location because we can localize the user's personal server to within 30m. We can even apply learning algorithms to automatically discern locations where the user spends considerable time and then allow the user to label them for use with the reminders [12]. Localization to this granularity promises to resolve many of the problems in generating timely alerts and minimizing problems due to false negative reads.

A second set of challenges is presented by the interaction with other users' tags. Of course, a user's own object tags can be registered with their personal server so that

any other tag ID is simply ignored (it may simply mean another user was near a reader at the same time). This is why it was necessary to register a user's auto-tags as these would otherwise could be construed to be another user's tags the first time they are seen. Another issue related to this, is the ability to track another user's movements by keeping track of all the tags broadcast by readers. A remedy for this is for a user's home reader to automatically randomize tag IDs every evening. Writable tags with password protection make this feasible [10]. The large space of possible of IDs (64 to 128 bits) makes it unlikely that an ID will collide with another. This ability to re-write tag IDs and having the application run completely on the user's own client, their personal server, make this system much more privacy-friendly than infrastructure-based approaches. Furthermore it lends itself to incremental deployment.

Finally, probabilistic methods and machine learning have much to offer reminding applications. As false negatives are a reality, it is imperative that our reminder system be tolerant of possible dropped reads and consider the location of objects probabilistically until it receives incontrovertible evidence. For example, a gym bag might have last been seen at home, but it may have made its way home without being detected by the home reader. The system must maintain a probability that the gym bag is back at the gym or at home. Over time, it may learn what is most likely given past experience. Learning may be extended further to help come up with basic reminder rules when a user first purchases the application. This may minimize the need for creating reminders explicitly or at least create starting templates that already include regularities such as weekends. Most importantly, learning where the user is likely to go next when they leave a location can greatly mitigate the need for calendar information which we currently use [12].

7 Conclusion

We have built a working prototype of an RFID-based object reminder system. It uses a novel combination of broadcasting RFID readers, a personal server which runs the software, and a wristwatch user interface to deliver the two levels of reminders. Our initial results based on a complete user scenario indicate that the system works well even in the face of a large number of missed RFID tag reads. We are confident that our approach is practical and scalable to a large number of users.

Our next steps are: to integrate our system with a WiFi-based location estimation system that can help us better determine when a user is leaving a location and thereby issue more definitive and timely reminders; create a graphical UI for creating and maintaining reminders; augment the wristwatch interface with the ability to help a user track down a lost object by returning to its last known location and/or retracing steps. A full user study is also planned so that we can better study the interactions between multiple users and their ability to easily adjust their reminders.

References

1. [redacted]
2. Silvermam, B.G.: Computer reminders and alerts. IEEE Computer , Vol. 30 , No. 1 , Jan. 1997, 42 – 49.
3. Dey, A., Abowd, G.: Cybreminder: A context-aware system for supporting reminders. 2nd International Symposium on Handheld and Ubiquitous Computing, Bristol, UK, Sep. 2000, 172-186.
4. Lamming, M., Bohm, D.: SPECs: Another Approach to Human Context and Activity Sensing Research Using Tiny Peer-to-Peer Wireless Computers. 5th International Conference on Ubiquitous Computing, Seattle, WA, Oct. 2003, 191-199.
5. Brunette, W., Hartung, C., Nordstrom, B., Borriello, G.: Proximity Interactions between Wireless Sensors and their Applications. 2nd ACM International Workshop on Wireless Sensor Networks & Applications, San Diego, CA, Sep. 2003, 30-37.
6. Want, R.: RFID: A Key to Automating Everything. Scientific American. Jan. 2004, 56-65.
7. Alien Technology RFID Readers and Tags. http://www.alientechnology.com/
8. Hill, J., et al.: System architecture directions for network sensors. 9th International Conference on Architectural Support for Programming Languages and Operating Systems, Cambridge, MA, Nov. 2000, 93-104.
9. Want, R., Pering, T., Danneels, G., Kumar, M., Sundar, M., Light, J.: The Personal Server: changing the way we think about ubiquitous computing. 4th International Conference on Ubiquitous Computing, Goteborg Sweden, Oct. 2002, 194-209.
10. Intermec Technologies Coporation: Guide to RFID Tag Selection, Sep. 2003 (http://epsfiles.intermec.com/eps_files/eps_brochure/RFIDTagSelectionGuide_brochure_web.pdf)
11. Schilit, B., et. al.: Challenge: Ubiquitous Location-Aware Computing and the Place Lab Initiative. 1st ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots, San Diego, CA, Sep. 2003, 29-35.
12. Ashbrook, D., Starner, T.: Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5), 2003, 275–286.
13. P.J. Brown, J.D. Bovey and X. Chen. Context-Aware Applications: from the Laboratory to the Marketplace. IEEE Personal Communications, No. 5, Vol. 4, 1997, pp 58-64.
14. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, Oct. 1994, 85- 90.
15. Siegemund, F., Flörkemeier, C.: Interaction in Pervasive Computing Settings using Bluetooth-Enabled Active Tags and Passive RFID Technology together with Mobile Phones. IEEE Conference on Pervasive Computing and Communications, Dallas, TX, Mar. 2003, 378 - 387.
16. Want, R., Fishkin, K., Gujar, A., Harrison, B.: Bridging Physical and Virtual Worlds with Electronic Tags. ACM Conference on Human Factors in Computing Systems, Pittsburgh, PA, May 1999, 370-377.
17. Toney, A., Mulley, B., Thomas, B., Piekarski, W.: Minimal Social Weight User Interactions for Wearable Computers in Business Suits. IEEE 6th International Symposium on Wearable Computing, Seattle, WA, Oct. 2002, 57-64.
18. Lamming, M., Flynn, M.: "Forget-me-not" - Intimate Computing in Support of Human Memory. Invited keynote paper for FRIEND21 Symposium on Next Generation Human Interface, Feb. 1994.
19. Rhodes, B., Starner, T.: The Remembrance Agent: A continuously running automated information retrieval system. 1st International Conference on The Practical Application of Intelligent Agents and Multi Agent Technology, London, UK, Apr. 1996, 487-495.