平成15年度

筑波大学第三学群情報学類

卒業研究論文

題目
ユビキタス時代の柔らかいストレージの提案と試作

主専攻　　情報科学主専攻

著者　　岩淵 志学

指導教員　電子・情報工学系 田中二郎

要　旨

ユビキタス環境においては、多くのコンピュータが情報アプライアンスとして遍在し、人間の意識するところから離れて利用されると言われている。しかしながら、コンピュータがその姿を簡潔で暗黙的な存在に変化する一方で、デジタルデータを記録するためのストレージとそのメディアは未だに明示的な存在のままであり、従来から本質的に変化していない。また現在のコンピューティングにおいてストレージはいくつかのインタラクション上の負担を利用者に与えている。将来のユビキタスコンピューティングを想定しても、その負担が少なくなることが望ましい。本研究では、利用する際の負担が少なく、暗黙的に人々の生活に入り込むような “柔らかいストレージ” をユビキタス時代のストレージの姿として提案する。また、柔らかいストレージを実現するメディアとして、RFID を利用した FlipFloppy システムを試作し、その活用例について述べる。

目 次

図 目 次

表 目 次

第1章　はじめに

現在、ユビキタスコンピューティング [1] の時代が到来しようとしている。携帯電話は普及段階からすでに成熟段階に移行しており、携帯情報端末 (PDA) 等のデバイスも普及が進んでいる。また、情報アプライアンスと呼ばれる個々の分野を専門的に処理する道具としてのコンピュータが注目されている。ユビキタス環境においては、コンピュータは情報アプライアンスとして人の日常生活に入り込み、人はコンピュータを意識せずに生活を送るようになると言われている。D. Abowd らの唱える EverydayComputing[2] では、日常生活に限らず教育などの場面におけるインフォーマルなコンピューティングも望まれている。現在のコンピューティングは、コンピュータを明示的に利用するものであるが、今後は人の普段の生活とコンピューティングを明確に分けることができない暗黙的なコンピューティングが主流になると思われる。

さて、現在の明示的なコンピューティングにおいてデータを保持・移動するために利用されているストレージおよびそのメディアを無視することはできない。しかし、これらは利用者に様々な負担を与えている。今日コンピュータがその形を変え、人の生活に溶け込もうとする一方で、リムーバブルメディアは従来からデータを記録する物として明示的に存在したままである。ユビキタス時代のコンピューティングでは、人とコンピュータのインタラクションは自然で負担の少ないものになるだろう。その際には、現在のようなストレージおよびメディアの利用はそのようなインタラクションの大きな障害となる可能性がある。暗黙的なコンピューティングを害すことがないような理想的なストレージが望まれている。

本研究の目的

本研究は、「ユビキタスコンピューティングの到来と情報アプライアンスの普及を想定し、利用者の観点から負担の少ないストレージを実現する」ことを目的とする。

本論文の構成

本稿の構成について説明する。第 2 章では、将来のコンピュータに関するデザインとして、ユビキタスコンピューティング、情報アプライアンス、自然なインタフェースについて説明する。また、現在利用されているストレージとメディアについて分類し、それらが人に与えている負担を問題点としてまとめる。第 3 章では、それらの問題点を踏まえ、人にとって負担の少ない理想的なストレージである “柔らかいストレージ” を提案する。さらに、柔らかいスト

レージを実現するためのメディアの 1 つとして、RFID とネットワークを用いた FlipFloppy を考案する。第 4 章では、FlipFloppy の実装について詳しい説明をする。第 5 章では、FlipFloppy を実際に活用する例をいくつか挙げ、従来のメディアと比較してどのような利点があるかを考察する。第 6 章では、柔らかいストレージと FlipFloppy に関連する研究について比較と議論を行う。最後に、第 8 章で本稿をまとめる。

第2章　背景と問題点

本章では、近年注目されているコンピューティングおよびインタフェースのデザインの中から、ユビキタスコンピューティング・インビジブルコンピューティング・情報アプライアンス・自然なインタフェースについて説明する。また、現在一般的に利用されているストレージおよびそのメディアが持つ問題点について述べる。

2.1　ユビキタスコンピューティング

これまでの一般的なコンピューティングは、1 人が 1 ないし 2 台のデスクトップ PC やノートブック PC を利用するものであった。これに対して、生活の様々な場所にコンピュータが遍在し、1 人が日常生活の中で数多くのコンピュータを利用するようなコンピューティングを、**ユビキタスコンピューティング**と呼ぶ。ユビキタスコンピューティングは M. Weiser によって唱えられた。具体的に M. Weiser はコンピュータの大きさによって普及の規模に違いが生じると考え、1 部屋に大画面のコンピュータが 1 ないし 2 台、A4 用紙サイズの情報端末が 10 台、身につける小型の装置が 100 台程度の割合で利用されると予想した。これらの多数のコンピュータは互いにネットワークによって接続され連携して動作する。

2.2　情報アプライアンス

また、D. Norman は人の観点から見た際ににコンピュータが人の意識するところから消える、**インビジブルコンピューティング** [3] を唱えた。インビジブルコンピューティングの例として、次のような電気モータの話が良く言われる。

> 電気モータは当初、高価なものだったのでモータそのものが販売されていた。モータだけでは何もできないので、利用者はアタッチメント装置を別途に購入し、モータに取り付けることで様々な用途に使用した。しかし、モータを扱うためには専門的な知識が必要とされたこともあり、なかなか一般には広まらなかった。その後、モータは大量に生産されるようになってコストが下り、色々なものの一部として埋め込まれるようになった。現在では、人は道具の中にモータが入っているかどうかを意識することなく利用するようになった。

この話は、現在のコンピュータが汎用的で複雑になっていることに対応し、コンピュータの未来の姿は 1 つのタスクを処理するように特化された道具であることを示している。この

ような道具は**情報アプライアンス**と呼ばれる。

Digital Decor[4] は透明な存在のコンピュータによって強化された日用品である。ユビキタスコンピューティングではこのような情報アプライアンスの利用が一般的なものになると考えられている。

2.3 自然なインタフェース

従来からコンピュータの操作は、マウスとキーボードによるものがほとんどだった。キーボードやマウスは汎用性を重視した現在のコンピューティングに適応したインタフェースである。しかし、情報アプライアンスと化したコンピュータにとってこれらは必ずしも最適なものとは限らない。1 つのタスクに特化されたコンピュータに対しては、**自然なインタフェース**が搭載される。自然なインタフェースは、話す、見る、書く、物を掴むなどの人間の基本的な動作を利用するものである。それにより人は情報アプライアンスをより直感的に利用できるようになる。

2.4 ストレージとメディア

現在、我々はストレージを頻繁に利用しているが、ユビキタスコンピューティングにおいて人が多数の情報アプライアンスを利用する時代になっても、そのようなストレージを利用する機会は多いと考えられる。

ここで、ストレージとそのメディアについてまとめる。**ストレージ** (Storage) とは、デジタルデータを記憶するための論理的な場所であり、その大きさの単位には主にビットやバイトが用いられる。また、この論理的な概念であるストレージを物理的に実現するための媒体を**ストレージメディア**もしくは単に**メディア**と呼ぶ[1]。

ストレージメディアには**固定メディア** (Fixed Media) と**リムーバブルメディア** (Removable Media) がある。固定メディアは、物理的にコンピュータの中に取り付けられていて、人が自由に持ち運ぶことができないメディアである。比較的大容量で高速にアクセスできるのが特徴で、特にハードディスクは代表的な固定メディアである。リムーバブルメディアは持ち運び可能なメディアであり、今日では数多くの種類が利用されている (図 2.1)。

これらストレージを利用する目的は 2 つにわけられる。1 つはデータを保持することであり、もう 1 つの大きな利用方法は、あるコンピュータから他のコンピュータにデータを移動することである。これはデジタルカメラで保存した写真データをデスクトップコンピュータに移動したり、CD などのメディアを用いて音楽や映画などのデジタルコンテンツを販売することにあたる。この点において、データの移動が物理的なリムーバブルメディアによって行

[1]揮発性のメモリデバイスもストレージを実現している媒体であるので、この定義のなかに含まれるが、電流が流れていることでしか維持できない。本稿で言うストレージメディアは、電源が無い環境でもデータを維持できるような媒体を指すことにする。

図 2.1: 現在利用されているリムーバブルメディアの数々

なわれるリムーバブルストレージと、データの移動がネットワークを通して行なわれるネットワークストレージとに分けることができる。

リムーバブルストレージ

光・磁気・電気などを利用して情報を記録しているストレージであり、CD、DVD、フロッピーディスク、メモリカード、磁気テープなどがこれに当たる。これらのメディアは、物理的な特性を利用しているため、実体を持ち、大きさがある。また、読み取り・書き込みのためにドライブ装置を必要とする。このため、扱うにあたってドライブに挿入するなどの物理的な操作が必要とされ、人には**物理的負担**がかかる。

ネットワークストレージ

近年、インターネットや LAN などのネットワークが広く普及したことで、物理的なメディアを利用せずにデータのやりとりを行うケースが増えてきた。FTP や HTTP など、サーバ側のハードディスクの内容をネットワークを通じてダウンロードするものが多い。これはデータの移動がなされるという点で、物理的メディアを介さないストレージと言える。ネットワークを用いてデータが取り出せるので、利用者は物理的なメディアを扱う必要がなく、ドライブ装置も必要ない。このため物理的負担はほとんどない。しかし、HTTP であれば URL を覚えなくてはならない、FTP であれば大容量のストレージの中から目的のデータを取り出すのにディレクトリの構造を把握しなければならない、など、**心理的負担**が比較的大きい。

図 2.2 は横軸に物理的負担の大きさ、縦軸に心理的負担の大きさをとって、代表的なメディアの位置付けを表したものである。

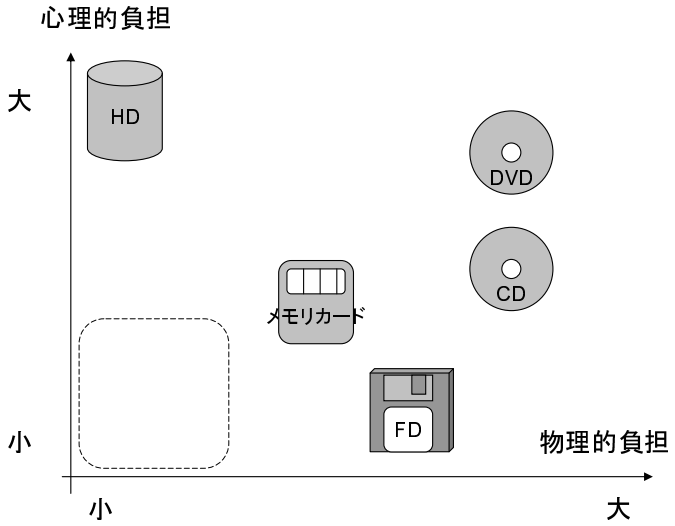

図 2.2: 負担によるストレージの分布

CD や DVD はメディアが大きく、自由に書き込むことが難しいストレージである。ハードディスクでは物理的な負担はないものの、容量が大きいことからファイルの位置を把握するのが心理的に負担になる。この点では容量が 4GB 程度ある DVD でも同じようなことが考えられる。それらに対してフロッピーディスクでは、何が記録されているか分かりやすい。またメモリカードは大きさもコンパクトなので物理的負担は軽い。

今まで述べたようなストレージ以外にも、データを移動する手法が提案されている。ペン型デバイスでデータを移動させる Pick-and-Drop[5] や、木のブロックで掴めるインタフェースを採用した mediaBlocks[6] なども、データをコンピュータから他のコンピュータに移動することができる。これらはペンでデータをすくうメタファを利用したり、オンライン上のデータを手に持つという直観的なインタフェースを備えていることから、心理的負担が少ない。

図中の点線の部分は、心理的負担と物理的負担の両方が少ない理想的なストレージであり、本研究が目指すところである。

2.5 ストレージの問題点

ここで、各種ストレージの負担について詳しくまとめる。表 2.1 に各ストレージの利用において考えられる負担の比較をまとめた。比較するストレージは CD、DVD、MO、メモリカード、Web、そしてハードディスクである。比較した点は、次の点において人がどのような負担

を受けるかである。

1. 認識 ... コンピュータにそのストレージを認識させる

2. 書き込み ... コンピュータからストレージにデータを書き込む

3. 管理 ... ストレージの内容を把握する、取り扱う

CD や DVD に限らず、ほとんどの物理的なメディアを介するストレージは、データにアクセスするためにメディアをドライブ装置に挿入しなくてはならない。アプリケーションからデータを読み込みたい時にはコンピュータを利用しているが、ストレージを利用するために機械的な作業が必要となることは手間になる。ハードディスクや機械的な作業の手間はない。Web の共有ストレージはプロトコルによってある程度の操作が必要となる (HTTP では URL を入力する、など)。

また、データの書き込みについては、特に CD-R や DVD-R などの規格では追記や書き込み速度に関する制約が強い。一般的に、OS は CD や DVD の書き込みをサポートしないので専用のソフトウェアが必要になっており、保存したい時にすぐ書き込むことは難しい。メモリカードなどは書き込み・読み込み双方に特別なプロセスは必要ない。Web によるネットワークストレージはプロトコルにより負担は異なるが、FTP では書き込みにはある程度のコマンド操作や GUI 操作が必要となる。

ストレージによっては、中に何が記録されているかを把握しやすいものとそうでないものがある。ハードディスクや FTP を扱う際に、目的のファイルがどこにあるのか忘れてしまうことはよくある。それに比べて、CD に記録した内容を忘れてしまうことは少なく、メモリカードの場合はその可能性はより低い。つまり、メディアが持つ記憶容量が大きい程、把握しにくくなる傾向にある。

表 2.1: ストレージとメディアの利用による負担 (コンピューティング)

	CD	DVD	MO	メモリカード	Web	HD
認識	△挿入	△挿入	△挿入	△挿入	×プロトコル	○
書き込み	×ソフト	×ソフト	○	○	△プロトコル	○
管理	△	×	○	○	×	×

また、ストレージはコンピュータがない状態でも日常生活の物から “記録されるもの” として独立して存在している。紙の書類を整理したり移動したりするように、メディアも物として管理しなくてはならない。表 2.2 にコンピュータが無い状況での負担についてまとめた。比較したのは次の点である。

1. 大きさ ... メディアの物理的な大きさ。運びやすさ。

2. 耐性 ... 物理的な耐性。取り扱いやすさ。

3. 譲渡 ... メディアそのものを他人に譲渡できるか。

メディアの大きさは、持ち運ぶ時の負担になる。CD や DVD はケースも含め、12cm 四方以上あり、他のメディアと比べても持ち運びの負担が多い。Web 共有を行う場合は持ち運ぶ必要がないので利用者に負担がない。耐性が弱いメディアも持ち運ぶ際に気をつけなくてはならないので負担になる。また、コンピュータが無い状態でデータだけを他人に渡したい場合を考えたとき、そのメディアの価値 (価格) が高い場合には所有者に強く帰属するので、譲渡できない。特に USB メモリなどはハードウェアなので価格が高くなり、DVD などでは値段だけでなく渡したくないデータも記録される場合が多いので、譲渡するのが困難になり易い。Web 共有の場合、メディアの価値に制約を受けない。HTTP では URL を教えることでデータを他人に渡すことができる。

表 2.2: ストレージとメディアの利用による負担 (日常生活)

	CD	DVD	MO	メモリカード	Web	HD
大きさ	大きい	大きい	中	小さい	-	-
衝撃耐性	弱い	弱い	強い	強い	-	弱い
譲渡	△	×	×	×	○	×

このように、従来メディアにはインタラクション上の問題点が多い。新たなメディアが登場しているが、それはフロッピーディスクの世代から根本的に同じであって、MO やメモリカードなどは大容量のフロッピーディスクと変わらないのが現状である。

ここで、これらの問題が人とコンピュータのインタラクションの中で発生したときの影響を例を挙げて考えてみる。デジタルカメラで撮った写真に対して、それをセピア色に加工する簡単な情報アプライアンスがあるとする。そのアプライアンスには、写真データを与えなくてはならない。そこでメモリカードを用いるとすれば、カメラからカードを取り出し、加工するアプライアンスに挿入する操作が必要となる。このときに利用者が本来行いたいことは、写真をセピア色に加工することである。しかしその目的に到達するまでの間に、メディアの操作が必要となる。これは利用者とコンピュータのインタラクションにメディアが干渉していると言える。挿入だけに限らずメディアが要求する操作が複雑であれば、それだけ干渉が強くなり利用者はより多くの負担を受けることになる。

多くの情報アプライアンスが自然なインタフェースを備えて遍在することが予想される中で、このような従来メディアの持つ問題点は改善されるべきであると言えよう。

第3章　柔らかいストレージ

前章では、現在主に利用されているストレージおよびメディアとその問題点について触れた。本章では、人への負担が少ない “柔らかいストレージ” について述べる。また、柔らかいストレージを実現するメディアの 1 つとして、FlipFloppy を考案する。

3.1　柔らかいストレージの提唱

重さや大きさ、アクセスに対する準備など、メディア側の都合による負担が限りなく小さく、インタラクションに干渉しないような理想的なメディアを考えよう。このストレージを利用すれば、利用者はストレージに煩わされることなく本来行いたいタスクに集中することができる。コンピュータが自然なインタフェースを備えているときには、それを害すこともない。

また、コンピュータを扱わないときには実世界の物に溶け込むことも考えよう。普段明示的に存在しないようなメディアであれば、必要以外の時は人の意識するところから消えることができる。インビジブルコンピューティングと合わせてストレージも消えることで、人は形式的なコンピューティングの束縛から解放されるであろう。

このような理想的なストレージの概念を “柔らかいストレージ” として提唱する。具体的に、柔らかいストレージは**暗黙的存在性**、**操作の簡潔性**、**賢い振る舞い**の 3 つの特徴を持つ。次にそれぞれの特徴について説明する。

3.1.1　暗黙的存在性

M. Weiser によって提唱された Calm Technology[7] は、**ペリフェリ**という概念で説明されている。ペリフェリとは人の意識の中心から外れたものである。車の運転で言えば、道路のセンターラインや道路標識は意識の中心にあるもので、エンジン音や走行の振動など、意識しないものがペリフェリである。Calm Technology とは普段はペリフェリにあって、必要なときにだけ意識の中心に現れるような技術を言う。

柔らかいストレージは Calm Technology の特徴を持ち、実世界に暗黙的に存在する。明示的なメディアは利用者の意識に現れることで負担になるので、コンピューティングの時以外には意識しないで済む方が良い。もし日用品をストレージとして用いることができれば、利用者は普段の生活ではそれがストレージであることを意識しないで、単に日用品として利用する。つまり、その日用品が持つストレージはペリフェリに存在して人々の意識には現れな

い。しかし、その日用品をコンピューティングに利用する時には、日用品ではなく、ストレージとして意識の中に現れる。

3.1.2 操作の簡潔性

柔らかいストレージは、自然なインタフェースを備えており、人が簡単に行なえるような行動 (置く、握る、見るなど) がキーとして扱われる。現在使われている CD ではストレージにアクセスするために、CD ドライブの挿入ボタンを押す、トレイが開く、メディアを挿入する、というようなかなり長いプロセスを必要とする。この一連の操作はメディアの都合によるものであって、利用者が本来行いたい処理を妨げる。柔らかいストレージではそのようなメディア主体の操作ではなく、人を主体とした簡単な動作をキーとする。

3.1.3 賢い振る舞い

ハードディスクや DVD、ネットワークストレージなどの大容量メディアを扱う際には、どこにどんなデータが記録されているのか把握できなくなることがよくある。コンピュータの世界では、1 つの意味を持ったデジタルデータを一般にファイルと呼んでいるが、このような問題は、1 つのストレージが持つファイルの数が個人の管理できる限界を越えた時に起こる。物とデジタルデータの対応にはバランスがあり、このバランスが崩れるときに負担が生じると考えられる。ここで図 3.1 に物理的なメディアと仮想世界のファイルとの関連付けの種類を示す。

(A) は現在のハードディスクの利用方法であり、ファイルがまとまってディレクトリを形成し、それが多数存在している。持つ意味が異なるファイルやディレクトリが遍在するため、人が管理できる限界を越えてストレージ上のデータを把握できなくなる。(B) は CD などの利用方法に当たる。1 つのストレージ上にファイルが複数存在するが、ある程度の意味の抽象度 (同じアーティスト、同じ製品) においてまとまっている。(C) は、(B) よりも抽象度が低く、メディア 1 つに対してファイルが 1 つ割り当てられている。(D) はファイルの中の単語やバイトごとにメディアが割り当てられている状態である。

(C) のような 1 対 1 対応は、1 つのファイルが 1 つの物として見えるので人はデータを直観的に認知しやすい。しかし、管理したいファイルが 1 つ増えると、物理的にも 1 つのメディアを管理しなくてはならないので、ファイル数が多くなれば物理的なメディアを管理する限界に達する。このとき、ハードディスクの中でファイルが行方不明になるのと同じ事が実世界で起こってしまう。また FTP などネットワークによるストレージはハードディスクと同じ (A) に当たり、人にとっては直感的に理解しにくい。

つまり、ストレージはデジタルデータが持つ意味のまとまりと物理的なメディアのバランスが重要となる。柔らかいストレージはこの問題に対して柔軟な情報管理ができる。具体的にはストレージを利用しているユーザやコンピュータの状況などが持つコンテクストを用いて、状況に応じた処理をすることなどが考えられる。

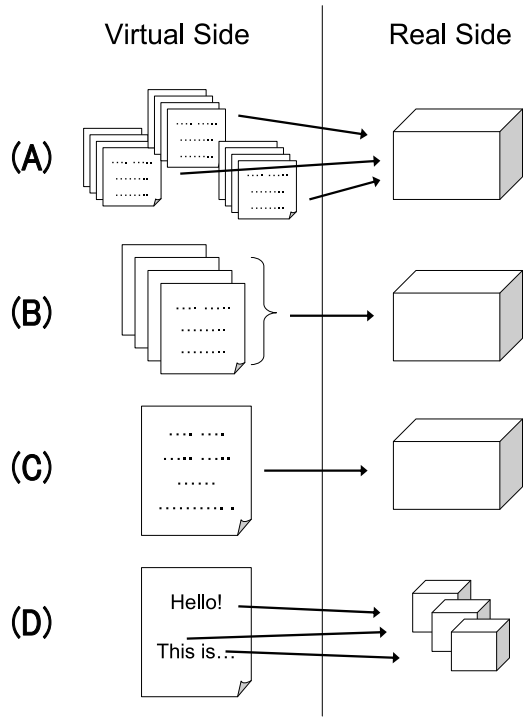

図 3.1: 実世界の意味と仮想世界のデータを対応付ける

3.2 柔らかいストレージの実現

理想的な概念である柔らかいストレージを実現するためにの方法を検討していくことにする。

ストレージを実現するには、データを物に直接記録する方法と、データをオンライン上に配置してそのポインタを物に記録する方法がある。前者は現在の CD のようなメディアの実現方式である。この方式は実用的なデータ容量を持たせるために物に対して非常に精密にデータを記録しなくてはならず、結果としてデータを記録するための (つまり明示的な) 物になってしまう。後者の方法は、物に ID を割り当てて、その ID とオンラインのストレージを結び付けることで実現でき、ポインタに必要なデータ量が小さくて済むというメリットがある。実世界の物に ID を付与するだけならば、その物の見た目や形を保つことができる。また、オンラインのデータは動的に変化するものであっても構わないので、ライブ映像などのデータも記録することができる。我々は柔らかいストレージの実現には後者の方法が有用と判断した。

なお、ID の指すストレージにアクセスするためにはコンピュータがネットワークに接続されている必要があり、また ID に対応するストレージの容量も十分に用意する必要がある。最近ではインターネットプロトコルの IPv6 化が進み、無線や電灯線を利用したネットワークによって家庭内のモバイル器機もネットワーク化されると思われる。さらにハードディスクなどのストレージの容量は年々増化している。これらネットワークの普及とストレージ容量の

増化の関係は、ユビキタスコンピューティングやモバイルコンピューティングが発展する上で注目されている [8]。

3.3 FlipFloppy の考案

物の認識を行なう ID にはバーコード、QR コードなど紙に印刷するもの、RFID のように無線で IC チップと通信するものから、指紋、虹彩などの生体の特徴を利用したものまで、様々な手法がある。我々は柔らかいストレージを実現するためのメディアとして、RFID(無線 ID）を採用することにした。RFID とネットワークによって実現されるメディアを **FlipFloppy** と呼ぶことにする。FlipFloppy において RFID を採用した理由は次の通りである。

1. **物理的な柔軟性** RFID タグは IC チップにデータを記録するので、一般に小型で軽い。柔軟に物に取り付けることができ、見た目や大きさが変わらないので実世界の物と同化することができる。
2. **非接触でアクセスできる** ID を認識させるために読み取り装置に明示的に提示する必要がない。バーコードは利用者が読み取り装置でスキャンしなくてはならないが、RFID タグは RFID ドライブの近くにあるだけで良い。
3. **メディア単価が安い** 物理的な価値の高いメディア (分かりやすく言えば、値段が高いメディア) は、譲渡することが難しい。RFID は現在数 10 円単位であり、今後はさらに安価になると予想される。安価なメディアを採用することによって、データを渡したい場合にはメディアそのものを渡すことが可能となる。
4. **データの書き換えが容易である** バーコードは一度印刷してしまえばデータを書き換えることができず、新しいバーコードを印刷しなくてはならない。RFID はタグのデータを自由に書き換えることができる。

このように、RFID の単に ID としての能力が高いだけではなく、物理的な観点からも実世界に溶け込む能力を有する。ネットワークストレージの代表である HTTP では、URL を ID として利用しているが、実体がなく URL を覚える必要がある。それに対して RFID は、物理的なメディアそのものが ID を持つので、タグを持つことがデータを持つことになる。

また、3.1.3 で述べたような情報管理の問題について、柔らかいストレージを実現するためには ID とストレージの対応付けについても工夫が必要と考えられる。柔らかいストレージを実現するには、バランスが従来メディアのように崩れないような工夫が必要である。そこで、FlipFloppy では 1 つのメディアに対して複数ファイルを対応させることにした。ディレクトリの構築は、先ほど述べたようなバランスの問題を考慮して意図的に許可しないことにする。(3.1 での (B) に相当する)

第4章　FlipFloppyの実装

FlipFloppy のプロトタイプを製作した。ソフトウェアのプログラミングには Microsoft Visual C#.NET、Visual C++、Perl を用いた。実行環境および実装に使用した装置は以下である。

コンピュータ CPU:PentiumII 500MHz, Memory 256MB

OS Microsoft Windows XP

RFID リーダ Texas Instruments Ti-RFid S6350

RFID タグ Texas Instruments Tag-it

4.1　システムの概要

FlipFloppy システムは、1 つのコンピュータ上で動作するローカルシステムを単位とする。ローカルシステムは 1 つの RFID タグと 1 つの記憶領域を対応づけ、RFID タグにアクセスする機能を持つ。記憶領域はストレージであれば何でもよいが、主にハードディスクのディレクトリが適当であろう。また、全体として複数のローカルシステムがネットワークで結ばれてることで、タグと記憶領域の対応付けがネットワークを越えて行われる (図 4.1)。

4.2　ローカルシステム

ローカルシステムは、タグと記憶領域の対応を管理する。また、ネットワークを越えたアクセスを提供する。試作では FlipFloppy のローカルシステムを 2 つのプログラムで構成した。

データサーバ タグと対応づけられた記憶領域を管理する Perl プログラムであり、HTTP プロトコルを利用した CGI プログラムとして動作する。タグのシリアル番号が与えられると、対応したディレクトリのファイル一覧を返す (インデクサ機能)。また、シリアル番号とファイル名を入力としてファイルのデータ本体を返す機能や、逆にファイルのデータを格納する機能などがある。機能の詳細は付録 A に添付した。

プログラムは Perl によって実装され約 450 行である。

ストレージマネージャ RFID ドライブを監視してタグを検出し、データサーバに接続してタグに対応する記憶領域をローカルマシンにダウンロードする。ユーザーインタフェイスも搭載しており、タグの初期化や内容の閲覧、コピーなどを行う機能がある。

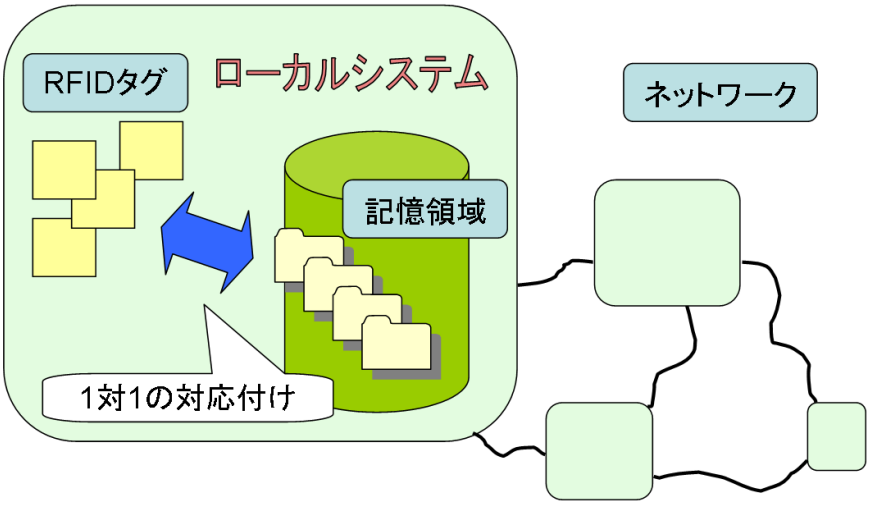

図 4.1: システム概念図

プログラムは C#によって実装され約 1000 行である。

これら 2 種類のプログラムを、RFID ドライブの接続されたマシン上で動作させて 1 つのローカルシステムとする。

プロトコルの流れ

タグの検出から排出までの一連の動作を図 4.2、図 4.3、図 4.4 に示す。

ストレージマネージャは、RFID ドライブからタグを検出すると、タグから 1. シリアル番号 2. ホストマシンの IP アドレス を読み込む。次に、IP アドレスに基づいてデータサーバプログラムに HTTP 経由で接続を行い、タグが持つファイルのインデックス一覧をダウンロードする。

インデックスのダウンロードが終了すると、ファイル本体をローカルマシンのディレクトリに全てダウンロードする。試作システムでは、ディレクトリを次のように設定した。

```
C:\flipfloppy\(タグのシリアル番号)
```

検出後、ストレージマネージャは検出されたタグのシリアル番号を保持しており、一定時間ごとに RFID ドライブを通じてそのシリアル番号を持つタグが存在しているかどうかを調べる。もし存在しているならば特に処理は行わないが、存在していない場合にはタグが排出されたと見なす。排出された時には、ローカルマシンに構築された一時的なストレージの内容がホストマシンと異なっている可能性がある。整合性を保つために、ストレージマネージャはデータサーバに接続し、挿入から排出までの間に変更されたファイルを調べて更新を行なう。

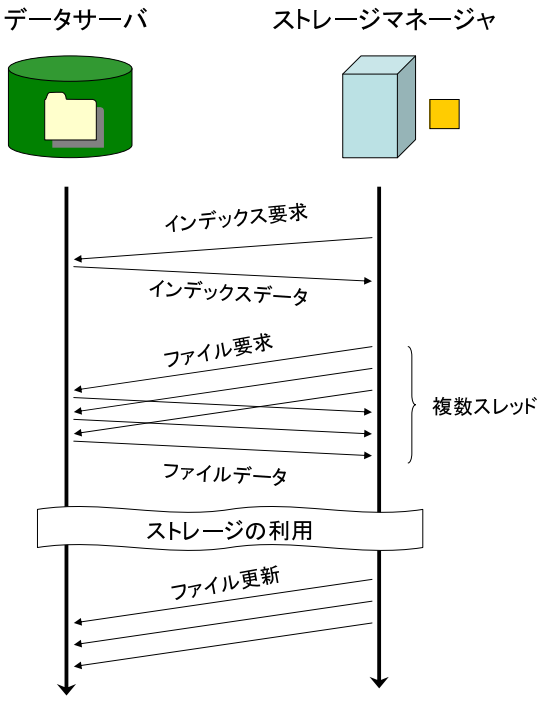

図 4.2: プロトコルの流れ

4.3 ハードウェアインタフェース

試作に用いた RFID ドライブ (TI-RFid S6350) は、15cm x 20cm x 3cm 程度の装置である (図 4.5)。装置の上面の空間に RFID タグがあればその情報にアクセスすることができる。検出できる範囲はタグが持つアンテナの大きさに依存するが、おおよそ〜30cm までである。13.56 メガヘルツの周波数帯で RFID タグと通信する。コンピュータとのインタフェースは RS232 であり、専用のプロトコルで通信する。試作ではストレージマネージャが RFID ドライブにアクセスできるよう、プロトコルに従って RS232C との通信をまとめた API セットを Microsoft 社の Visual C++で作成した (付録 B)。

FlipFloppy システムのメディアとなるタグ (Tag-it RFID トランスポンダ) は、数センチ四方のフィルムの形をしている (図 4.6)。小型かつ柔軟であるので、様々な物に取りつけることができる (図 4.7)(図 4.8)。

1 つのタグは、4 バイトのデータを 1 ブロックとして、8 ブロック分 (32 バイト＝ 256 ビット) のデータを記憶できる。データは書き換え可能である。また、タグごとに固有のシリアル番号が出荷時に記録されている。シリアル番号は 4 バイトであり、書き換え不能である。FlipFloppy システムは、メディアのホストマシンの IP アドレスを格納するのに第 2 ブロック

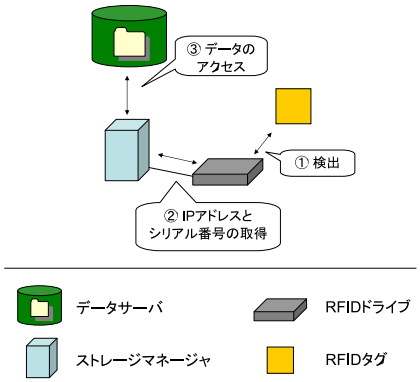

図 4.3: ローカルマシンがホストの場合の流れ

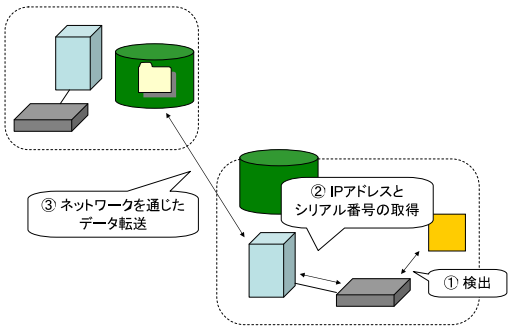

図 4.4: リモートマシンがホストの場合の流れ

図 4.5: TI-RFid S6350 RFID リーダライタ

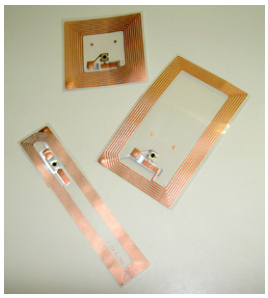

図 4.6: Tag-it RFID トランスポンダ

を使用する。それ以外のブロックは利用しない (図 4.9)。

4.4 ソフトウェアインタフェース

ストレージマネージャを起動するとメインウィンドウが開かれる (図 4.10)。

ストレージマネージャが RFID ドライブを通じてタグを検出すると、メインウィンドウにストレージアイコンが表示される。最初、ストレージアイコンは“？”マークである。タグが持つデータがデータサーバからローカルマシンにダウンロードされるまではそのまま変化しない。ダウンロードが完了するとアイコンは FlipFloppy アイコンに変化し、そのストレージが利用可能な状態となったことを示す。ネットワークの不具合や、初期化されていないメディアの場合などにはアイコンは“？”マークのままとなり、そのストレージを利用することはできない。

利用者は、ストレージに対する操作をすべてストレージアイコンを介して行う。ストレージアイコンをクリックすると、Windows のエクスプローラが起動し、ダウンロードしたファ

図 4.7: 手帳に取り付けられた RFID タグ

図 4.8: 財布に取り付けられた RFID タグ

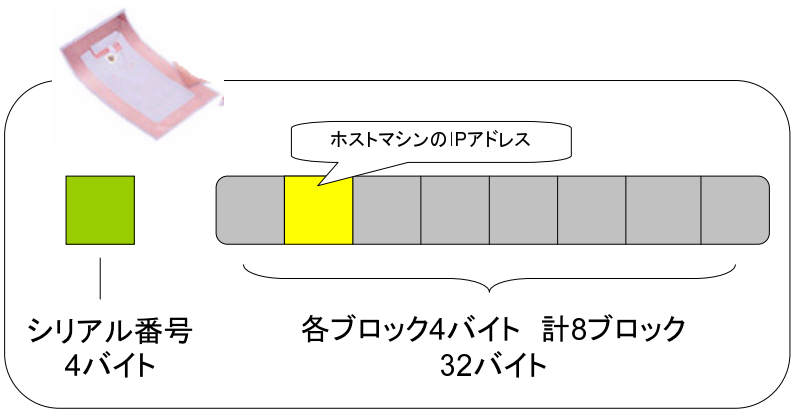

図 4.9: タグに記録されている内容

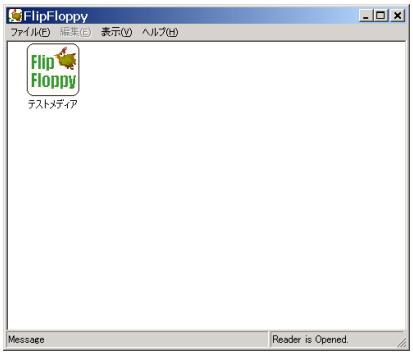

図 4.10: メインウィンドウ

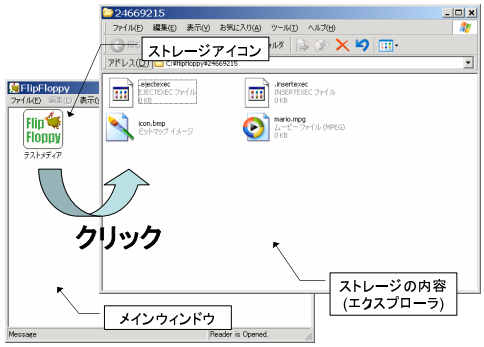

図 4.11: ストレージアイコンのクリック

イルを表示する (図 4.11)。利用者の視点からは、タグに記録されているデータを開いたように見える。

4.5　主な機能

4.5.1　自動挿入スクリプト

ストレージマネージャはメディアが検出された際に、メディアのインデックスから `inser-texec.bat` というシェルスクリプトを検索する。そして、もし見つかった場合にはそれを実行する。これにより CD の自動挿入に対応するような処理ができるが、RFID を用いているのでその用途の幅は広い。

4.5.2 自動排出スクリプト

排出時にも処理を行うことができる。メディアの排出の際にインデックスから`ejectexec.bat`を探し、見つかった場合にはスクリプトを実行する。排出後でもデータはネットワーク上に存在するのでアクセスすることができる。物理的なメディアを用いるストレージでは排出後にデータにアクセスすることは不可能である。また、ネットワークストレージでは挿入、排出の概念がないので不可能である。

4.5.3 メディアの初期化

FlipFloppy のメディアを利用するためには、初期化をしてデータサーバにメディアとデータの対応を作成しなければならない。初期化されていないメディアがストレージマネージャで検出されると “?” アイコンとなるので、ここで右クリックをしてメニューを開き、初期化を選択する。初期化ダイアログボックスが表示されるので、必要な項目を設定する。初期化ボタンをクリックすると初期化が行なわれる。

第5章　活用例

本章では、FlipFloppy システムの有用性を活用例を用いて説明する。

5.1　メールボックスに利用する

メールボックスは目に見えないネットワークストレージの典型的な例である。多くの一般のユーザはプロバイダーから提供されるメールボックスを利用している。ある話を例として挙げよう。

> 大学 4 年生の P 君は、FlipFloppy を手帳に 2 枚挟み込んでメールボックスとして利用している。朝、研究室に登校した P 君はノートパソコンと手帳をカバンから取り出してパソコンの前に置いた。これは彼のいつもの習慣である。パソコンの画面に、3 件のメールが届いたことを示すメッセージが表示された。1 件は先生からのメールで、残りの 2 件はスパムメールのようだが、ソフトウェアが自動的にメールを振り分けて保存した。P 君はそのまま研究をした。午後、研究室の先輩が、今朝先生から P 君に届いたメールが見たいと言った。P 君は手帳から 1 枚の FlipFloppy メディアを取り出して、先輩に渡した。用が済んでメディアを返しにきた先輩が、スパムが一つも無いことに驚いていた。

この話で注意したい点は 2 つある。まずこのメールボックスの特徴は、先生からのメールとスパムメールが振り分けられた際に、物理的に異なるストレージに格納される点である。先輩から先生のメールを見たいと頼まれた P 君は、片方の FlipFloppy メディアを渡すことで、スパムメールを除去することができた。

また、FlipFloppy を用いたメールボックスのもう一つの大きな特徴は、メールボックスを持ち運ぶことを意識しなくても良い点である。もしフロッピーディスクをこのようにメールボックスとして用いる場合には、利用者は家を出る際に手帳を持って行こうと意識するだけでなく、“フロッピーディスクを持っていこう” と、明示的に意識しなくてはならない。これに対して手帳そのものがメールボックスと化している FlipFloppy の場合は、メールボックスを意識する必要がない。

5.2 手ぶらでデータを持つ

> Q さんは日曜日に街へ出かけた。Q さんの洋服には両手のそでの部分に FlipFloppy が埋め込まれている。散歩をするときには身軽でいたい Q さんは、携帯情報端末などの明示的なコンピュータを特に身につけていない。音楽ショップに入った Q さんは、新曲をいくつか視聴した。気に入った曲が見つかったところサンプルデータがダウンロードできるようなので、Q さんは視聴する装置の上に手を一瞬かざした。すぐに曲のサンプルデータが FlipFloppy に保存された。帰宅して音楽再生プレーヤの上に手をかざすと、サンプルの再生が始まった。

このストーリーでは、洋服という日常生活に密着したものがストレージとして利用されている。子供は複雑なコンピュータを扱えないので持ち運ばないし、少しの物理的な重さや大きさが負担になる場合も多いので、コンピュータやメディアを持ち歩きたくない場合も多いが、FlipFloppy を活用することで明示的なコンピュータやメディアを持たずともデジタルデータを扱える。さらに、データをダウンロードする時まではストレージの存在を意識しなくても良い。

また、サンプルの曲データは実際には FlipFloppy が指すネットワークストレージ上に転送されるので、装置はホストマシンのアドレスとタグのシリアル番号を得ることができれば良い。実際のデータ転送は Q さんが手をかざした直後から行なわれる。大きなデータになると転送時間がかかるが、利用者の視点からは装置に一瞬手をかざすだけの軽い操作で転送が行なわれたように感じられる。

5.3 情報アプライアンスと連携する

ちょっとした休憩中の会話を後で思い出したい場合はよくある。FlipFloppy を情報アプライアンスとともに活用することで、意識しなくとも物に音声を吹き込むようなことも可能である。例えばテーブルの下に音声を録音する情報アプライアンスを取り付けておき、利用者は鍵などのテーブルに置くようなものに FlipFloppy を取り付けておく。すると、座っていた時間だけの音声をその鍵に記録することができる。RFID タグは明示的に認識させなくても良いので、利用者は特に記録しようと意識しなくても良い。あとから「そういえば...」と思った時に、鍵を連想のキーとしてストレージの存在を意識すれば良い。

記憶の連想に関して、河村らは UbiquitousMemories[9] のコンセプトを提案し、ウェアラブルコンピュータを用いて実世界の物を通じた記憶の拡張を行なっている。UbiquitousMemories ではコンピュータを持ち歩くことでどんな場所でも記憶を想起されることが可能であるが、コンピュータを装着する負担がある。また、記憶を結びつけたい場合には明示的な操作が必要とされている。FlipFloppy と遍在する情報アプライアンスの活用では、その場で記憶を閲覧することはできないが、暗黙的に音声などの記憶を物に格納することができる。

第6章　関連研究

B.Ullmer らは実世界アイコン (Phicon) として mediaBlocks を実装している [6]。mediaBlocks ではオンライン上のデータの扱いについて問題点を挙げ、リムーバブルメディアの有用性に着目している。本研究ではオンラインデータだけでなく現在のリムーバブルメディアの問題点を克服しようとしており、目的はより幅広い。mediaBlocks は木でできた小型のブロックを実世界のアイコンとしている。これに対して、FlipFloppy では RFID を取り付けた身の回りの日用品をメディアとして利用する。ブロックをメディアとして利用した場合、オンラインメディアをより直観的に持ち運んだり操作することが可能である。しかし、ブロックは明示的なメディアであって、従来の CD のようなメディアと同様 “記録される物” として明示的に存在することになる。FlipFloppy は暗黙的存在性から、必要な時以外は人の意識から消えるストレージを実現できる。また、mediaBlocks が採用している掴めるユーザインタフェース (TUI = Tangible User Interface)[10] は、FlipFloppy を取り付けた日用品に対しても適用することができる。この点で本研究は medeiaBlocks の考えを発展させたものである。

実世界の物に ID を取り付け、コンピュータ上のデータを結び付けるシステムに、IconSticker[11] が挙げられる。IconSticker はバーコードプリンターを用いて、デスクトップ上のアイコンをメタファーとした紙アイコンを実世界に取り出す。バーコードは明示的にスキャンしなくてはならないので、ストレージとして利用する上で負担になる。これに対して FlipFloppy は非接触アクセスができる RFID を利用しているので、明示的な提示の必要がないことから、利用者が無意識に行なう動作をキーとしてメディアのデータを扱うことが可能である。

また、IconSticker で作成した ID はバーコードなので一度作成するととそのポインタを書き換えることができない。FlipFloppy は ID に RFID タグを使用しており、物理的にはそのままでデータの書き換えが可能であることから、再利用性が高い。また、IconSticker ではデスクトップ上のアイコンのエイリアスを 1 つの物理的 ID とする。しかし 3.1.3 で述べたように、データと物理的な ID の 1 対 1 対応は直感的な理解を促すが、物理的な ID が大量になる可能性がある。FlipFloppy は 1 つの ID に複数ファイルを対応させることができるので、デジタルデータが持つ意味のまとまりに応じて 1 つのメディアに収めておくことができ、1 対 1 対応よりも柔軟である。

tranStick[12] は、動的なデータの物理的な媒体であるケーブルを仮想化したものである。物理的なケーブルを使うと、機器同士の接続の関係は明確にわかるが、絡まってしまうなど煩わしい物理的な問題がある。tranStick はメモリカードをペアにして、仮想的につながった見

えないケーブルの両端として利用している。データの移動という点において FlipFloppy と同じメディアであるが、tranStick はケーブルという動的にデータが移動するものに焦点をあてている。FlipFloppy は静的なデータの移動を行なうことを目的としている。

PaperLink[13] では、紙に描かれた文字列を ID としてデジタルコンテンツへリンクさせる。これは VideoPen と呼ばれるペンと一体型になったカメラで紙面を撮影し、画像を解析して紙面の文字を認識することで関連したコンテンツにアクセスする。このシステムもまた、実世界の ID とコンピュータのデータをリンクさせている。紙に書かれた文字列を ID を利用してデータのやりとりを行なう場合には、文字列が長くなると URL のように覚えにくくなってしまう。FlipFloppy では物にデータが記録されるので、物を移動することでデータの移動ができる。

暦本による Pick-and-Drop[5] は、ペン型デバイスを用いてコンピュータからデータをすくい上げ、他のコンピュータに落とすメタファを用いたシステムである。Pick-and-Drop は近くにあるコンピュータ同士のデータの移動を直観的に行なうことができる。しかし、メディアとなるペン型デバイスにはコンピュータが必要となるので、データを保持した状態のデバイスを移動させなければならない。FlipFloppy ではメディアを移動させることがデータを移動させることになるので、コンピュータを必要としない。

第7章　まとめ

本稿ではまず、ユビキタス時代が到来することを背景として述べ、現在主流のストレージおよびメディアが利用者に負担を与えていることを述べた。次に、自然なインタフェースやインビジブルコンピューティングのデザインに沿った “柔らかいストレージ” を提案した。そして柔らかいストレージを実現する 1 つの方法として、RFID とネットワークを用いた FlipFloppy システムを考案した。また、システムの試作を行って、その活用例を述べた。

今後の展望としては、情報アプライアンスと FlipFloppy の連携させたシステムの試作や、RFID 以外の ID を利用したメディアの開発を進める予定である。

謝辞

本稿の執筆にあたり、筑波大学電子・情報工学系 田中二郎教授には丁寧なご指導と適切なご助言を頂きました。ここに深く感謝いたします。また、有益な情報を与えて下さった筑波大学電子・情報工学系 志築文太郎講師ならびに三浦元喜助手に心から感謝いたします。筑波大学電子・情報工学系 田中研究室のメンバーの方々にも大変お世話になりました。この場を借りてお礼申し上げます。

参考文献

[1] Mark Weiser. The computer for the twenty-first century. In *Scientific American*, pp. 94–104. Scientific American, September 1991.

[2] Gregory D. Abowd and Elizabeth D. Mynatt. Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction*, Vol. 7, No. 1, pp. 29–58, March 2000.

[3] D.A.Norman. パソコンを隠せ、アナログ発想でいこう！, 情報アプライアンスの世界. 新曜社, July 2000.

[4] 椎尾一郎, Jim Rowan, 美馬のゆり, Elizabeth Mynatt. Digital Decor: 日用品コンピューティング. In *The 10th Workshop on Interactive Systems and Software*, 2002.

[5] Jun Rekimoto. Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments. In *UIST'97*, pp. 31–39, 1997.

[6] Brygg Ullmer, Hiroshi Ishii, and Dylan Glas. mediaBlocks: Physical Containers, Transports, and Controls for Online Media. In *SIGGRAPH'98*, Computer Graphics Proceedings. ACM, July 1998.

[7] Mark Weiser, John Seely Brown. Designing Calm Technology. December, 1995. `URL:<http://www.ubiq.com/hypertext/weiser/calmtech/calmtech.htm>`.

[8] Roy Want and Trevor Pering. New horizons for mobile computing. In *IEEE International Conference on Pervasive Computing and Communications*, 2003.

[9] Tatsuyuki Kawamura, Tomohiro Fukuhara, Hideaki Takeda, Yasuyuki Kono, and Masatsugu Kidode. Ubiquitous memories: Wearable interface for computational augmentation of human memory based on real world objects. In *The International Conference on Cognitive Science*, June 2003.

[10] H. Ishii and B. Ullmer. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. In *CHI*. ACM, March 22-27 1997.

[11] 椎尾一郎, 美馬義亮. コンピュータソフトウェア, 第 16 巻, IconSticker: 紙アイコンによる情報整理, pp. 24–32. 岩波書店, November 1999.

[12] 綾塚祐二, 暦本純一. tranStick: 空間を越えて仮想的に繋がったメディア. In *The 11th Workshop on Interactive Systems and Software*, 2003.

[13] T. Arai, D. Aust, and S. Hudson. PaperLink: A Technique for Hyperlinking from Real Paper to Electronic Content. In Proc. of ACM CHI'97, Conference on Human Factors in Computing Systems, March 1997.
URL:<http://www.ubiq.com/hypertext/weiser/calmtech/calmtech.htm>.

付 録A　データサーバ CGI 仕様

FlipFloppy システムの試作で用いたデータサーバは、ID に対応付けられたファイル集合を管理する CGI プログラムである。HTTP プロトコル上で動作する。データサーバはファンクションと呼ばれるファイル操作機能をいくつか持っており、それぞれが異なった入力・出力を行う。

A.1　ファンクション一覧

表 A.1: ファンクションと機能の説明

mode の値	sn の値	fn の値	機能の説明
indexer	シリアル番号	-	インデックスを得ます
newmedia	シリアル番号	-	インデックスを作成します
delmedia	シリアル番号	-	インデックスとファイルを削除します
update	シリアル番号	ファイル名	ファイルを更新します
delete	シリアル番号	ファイル名	ファイルを削除します
create	シリアル番号	ファイル名	ファイルを作成します

A.2　インデックスファイル 仕様

1 つの FlipFloppy が持つファイルの情報を記録しているのがインデックスファイルである。インデックスファイルは~/serverdata/(シリアル番号) ディレクトリの直下に置かれる。内容は、行区切りのデータで、最初にヘッダ、そしてファイル名が続く。ヘッダの種類を表 A.2 に示す。

インデックスファイルの例を以下に示す。

```
NAME: テストメディア
QUOTA: 256
INDEX:
\icon.bmp
\mario.mpg
\insertexec.bat
\ejectexec.bat
```

このインデックスファイルの例は、テストメディアという名前の FlipFloppy を表す。容量制限は 256 メガバイトである。ストレージには 4 つのファイルが記録されている。

表 A.2: インデックスファイルのヘッダ

ヘッダ名	値	説明
NAME:	文字列	メディアの名前
QUOTA:	数値	容量の制限値をメガバイト単位で指定する (0 ならば制限しない)
INDEX:	-	次の行からファイル名が続くことを宣言する

付 録B　Ti-RFID S6350 COM リファレンス

Texas Instruments TI-RFID S6350 はシリアルポートを経由してコンピュータと通信を行う。汎用性を考慮し、Windows の COM(Common Object Module) を作成した。COM モジュールで S6350 の機能をラップすることで、Visual Basic や Visual C#などの様々な言語から RFID ドライブを扱うことができる。開発は Microsoft Visual C++ 6.0 で行った。

B.1　配布の形式

`TiRFID.dll` という名前のダイナミックリンクライブラリで提供される。Ti-RFid S6350 COM オブジェクトを使用するには、`TiRFID.dll` がパスの通った場所になくてはならない。

B.2　インストール方法

開発に利用するには dll ファイルを COM サーバへ登録する必要がある。この操作は、エンドユーザの環境では必要ない。登録を行なうには、ウィンドウズのコマンドラインから以下のように入力する。

```
> regsrv32.exe (dll の存在するパス)\TiRFID.dll
```

B.3　公開 API 一覧

Ti-RFID S6350 COM が提供する API の一覧を表 B.1 に示す。

表 B.1: Ti-RFID S6350 COM API 一覧

API 定義	**引数**	**機能**
Open(long nPort)	nPort=シリアルポート番号	TI-RFid S6350 との通信を開きます。nPort に 0 を指定すると COM ポートではなくパイプを開きます。
Close()	なし	通信を閉じます。
GetReaderVersion(long *maj, long *min, long *t)	maj=メジャーバージョンを格納する変数へのポインタ，min=マイナーバージョンを格納する変数へのポインタ, t=戻り値	S6350 のファームウェアのバージョンを得ます。
ReadBlock(long bnum, long *val)	bnum=読み込むブロック番号, val=値を格納する変数へのポインタ	タグから 1 ブロック読み込みます。
WriteBlock(long bnum, long val, long *res)	bnum=書き込むブロック番号, val=書き込む値, res=結果を格納する変数へのポインタ	タグにデータを 1 ブロック書き込みます。
GetTagSerial(long *snum)	snum=シリアル番号	タグ固有のシリアル番号を取得します。
ReadBlockWithAddress(long bnum, long *val, long addr)	bnum=書き込むブロック番号, val=値を格納する変数へのポインタ, addr=シリアル番号	シリアル番号で指定したタグのみからデータを 1 ブロック読み込みます。
WriteBlockWithAddress(long bnum, long val, long *res, long addr)	bnum=書き込むブロック番号, val=書き込む値, res=結果を格納する変数へのポインタ, addr=シリアル番号	シリアル番号で指定したタグのみにデータを 1 ブロック書き込みます。