平成 21 年度卒業論文
論文題目

μIP を用いたセンサネットワークの効率化

神奈川大学 工学部 電子情報フロンティア学科
学籍番号　200602835
上甲　薫

指導担当者　木下宏揚　教授

目 次

図 目 次

1 序論

今日の社会発展に伴い、セキュリティシステムも様々な形式が考案され、社会に浸透している。金銭はもちろん、重要な情報に対するセキュリティの重要性は年々増している。最近では、凶悪犯罪の増加に伴い、企業だけではなく、一般の家庭においてのセキュリティの重要性も再確認する必要性がある。セキュリティとは危険な状態から守り、安全を保障する事となっているが、大きく防犯と防災に分けられ、防犯は守る対象により以下の様に使い分けれらる場合がある。

- 警護 ···· 人
- 警備 ···· 場所、建物、企業、貴金属や人も含む
- 保安 ···· 航空、鉄道
- 治安 ···· 社会
- 安全保障 ···· 国家

この中の警備に着目する。一般的に、事務所などの警備業務対象施設に警備員が常駐する常駐警備が行われてきたが、夜勤業務となることが多いため、人件費の増大などから、現在では監視カメラなどを設置したセキュリティシステムなどの導入も増え、機械を使った警備が主流となっている。

しかし、防犯カメラによるセキュリティシステムにも、人によっては圧迫感を与えてしまう事や、カメラの台数や性能に大きく依存している事等の問題も考えられる。そこで、過去に本研究室では、低コストで導入できる、光の環境の変化を感知し、検知対象の検知及び動作検知を可能にするシステムが考案された。

この研究の課題として、データ取得効率の向上が挙げられている。センサのデータを取得するスピードを上げることにより、取得するデータの量が増え、今までの数値の並びからは検知不可能だった動作が発見できる可能性が出てくるためである。

そこで本研究では、μIP プロトコルスタック [2] の実装による、高速データ取得システムを提案する。低コストで導入できる超高輝度 LED をセンサとし [10]、μIP を実装した AVR チップ [3] と組み合わせ、検知対象の動作検知を可能にするシステムである。センサにより電圧の変化を感知し、感知したデータを、μIP を実装した AVR チップにより TCP/IP[1] で通信させ、システムを構築する。そして、μIP のソースコードのアプリケーション部を解析、変更し、センサのデータ取得スピードを上げることを目的としている。

2 章で本研究を行うにあたって得た知識と、必要と思われる基礎知識を述べる。3 章で使用した機器を示し、4 章で実験内容を記し、5 章で結論とする。

2 基礎知識

この章では、以下の本を特に参考した。
竹下 隆史・村山 公保・荒井 透・苅田 幸雄/共著：
”マスタリング TCP/IP” 入門編 第 2 版 オーム社 1998 年-5 月 [1]

2.1 μIP

μIP は、SICS[7] のネットワーク組み込みシステム開発グループの Adam Dunkels 氏によって、組み込み向けに開発された 8 ビットのマイクロコントローラ [3] で動作するように実装されている、フリーソフトで無料の TCP/IP スタックである。複数のヘッダファイル、アプリケーションファイルで構成されており、ユーザーは TCP/IP スタックの部分はそのままに、IP アドレスやアプリケーションを自分の開発環境に合わせて加工することができる。プログラムは完全に C 言語で書かれており、コンパイラは AVR-GCC と Imagecraft である。全部合わせても極めてサイズが小さく、実行コードの大きさは約 6 キロバイト程である。[2] 各ファイルのソースコードはコメントが行き届いており、組み込み用途のみならず、TCP/IP スタックの実装を学ぶ上でも有効なものである。[8]

2.1.1 特徴

μIP の特徴を以下にまとめる。[6]

- 各ファイルのソースコードは、文章とコメントで詳しく説明されている。
- 非常に小規模なコードサイズである。
- RAM の使用率が非常に低く、コンパイル時に変更することもできる。
- 対応プロトコルは、ARP、SLIP、UDP、ICMP(ping)、TCP/IP である。[4]
- Web サーバー、Web クライアント、SMTP クライアント、Telnet サーバー、DNS リゾルバなどのアプリケーション例が含まれている。
- 同時に複数の TCP 接続が可能であり、接続の最大数はコンパイル時に変更することもできる。
- 複数接続されていても受動的に受信できる、最大数はコンパイル時に変更することもできる。
- 商用、非商用どちらにおいても自由に利用できる。
- RFC に沿ったフロー制御、フラグメント再構成、再送信のタイムアウト判定を含む TCP/IP プロトコルが実装済である。

2.2 プロトコル

人間は知能、応用力、理解力を持っているため、ある程度ルールから外れていても、意思の疎通を図ることができる。また、とっさにルールを変更したり、拡張したりすることもできる。しかし、コンピュータは知能、応用力、理解力を持っていないため、物理的なレベルからソフトウェアのレベルまで、様々な部分で明確な規約を決めて、それを守るようにしなければ通信することができない。人間の場合は、途中の言葉を聞き逃しても、会話の前後などから意味を推測して、相手の言いたいことを理解することができる。しかし、コンピュータの場合はそうはいかず、通信途中で障害が発生した場合にどのように処理するかなど、通信中に起こりうる様々な問題を事前に想定しておかなければならない。そして、実際に障害が発生した場合に、お互いに決めておいた適切な処理をする必要がある。このような、コンピュータ同士の通信において、コンピュータ同士できめ細かく決められている規約がプロトコルである。

2.2.1 プロトコルの標準化

標準化は、異なるメーカーの製品同士でも、互換性を持って利用できるような規格を作り上げることである。例えばコンピュータ通信以外でも、鉛筆の大きさが違ったら鉛筆削りに差し込むことができない、電源コンセントの大きさが違ったら差し込むことができないなど、標準化が行われていないと困ることは容易に想像できる。コンピュータ通信においてプロトコルが標準化されていない場合、違うメーカーのパソコン同士では通信することができないため、拡張性に乏しく、不便でとても使いにくいネットワークとなる。そのために、プロトコルの標準化が唱えられるようになり、通信の国際標準、OSI(Open System Interconnection) プロトコルが誕生した。

2.2.2 OSI 参照モデル

OSI 参照モデルとは、プロトコルを 7 つの階層に分け機能を分割することで、複雑になりがちなネットワークプロトコルを単純化するためのモデルである。各階層は、下位層から特定のサービスを受け、上位層に特定のサービスを提供する。このサービスのやりとりをするときの規約をインターフェースと呼び、通信相手の同じ層とのサービスのやりとりをするときの規約をプロトコルと呼ぶ。プロトコルを階層化すると、各階層を独立なものとして扱うことができるという利点があり、とある階層を変更しても、その影響が全体へ及ばないため、拡張性・柔軟性に優れたシステムを構築することができる。欠点として、階層化を進めすぎると、処理が重くなったり、各階層で似たような処理をしなければならなくなることがあり、その結果、無駄が増えるという問題がある。

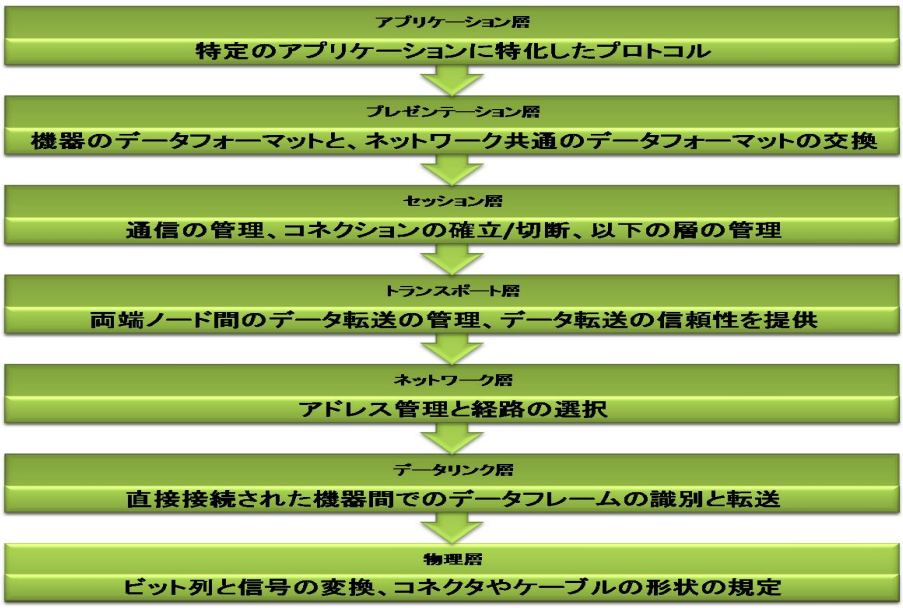

図 2.1: OSI 参照モデル

2.2.3 TCP/IP の誕生と普及

現在のコンピュータネットワークの世界では、TCP/IP が最も有名であり、かつ最もよく使われているプロトコルである。TCP/IP をサポートしていない OS はほとんど販売されていない。これほどまでに TCP/IP がサポートされるようになった経緯を、インターネット発達の歴史から考えていく。
1960 年代後半に、アメリカ合衆国の組織 DoD(the Delartment of Defense) を中心に、通信技術の研究開発や実験が行われていた。そこでは、通信は軍事的に非常に重要なものだと考えられ、パケット交換技術、パケット通信の必要性が唱えられるようになった。パケット通信を利用すれば、複数のユーザーで同時に 1 つの回線を共有し、回線の利用効率を向上させるというメリットがある。そこで、パケット交換技術の実用性を試験を行うため、大学と研究機関の 4 つのノードを結んだネットワークが開発された。そこに一般のユーザーを取り込んだ、当時としては非常に大きなネットワークである ARPANET が誕生した。そして、ARPANET 研究グループにより、ただのパケット交換だけでなく、信頼性の高い通信手段として開発されたのが TCP/IP である。この当時には既に大学や研究所で BSD UNIX が広く利用されており、この BSD UNIX が TCP/IP を実装することになった。そして、UNIX ワークステーションの急速な普及により、TCP/IP によるネットワークの構築が盛んになっていった。このようにして、TCP/IP は UNIX と密接な関係を持って発達し普及、TCP/IP は商用サービスにも耐えうる、成熟したプロトコルとなり、世の中にサポートされていくことになる。

2.2.4 TCP/IP プロトコル階層モデル

TCP/IP は TCP と IP という 2 つのプロトコルだけではなく、IP や ICMP、TCP や UDP、TELNET や FTP、HTTP など TCP や IP に深く関係するプロトコルが多く含まれており、インターネットを構築するうえでの必要なプロトコルがセットになっている。

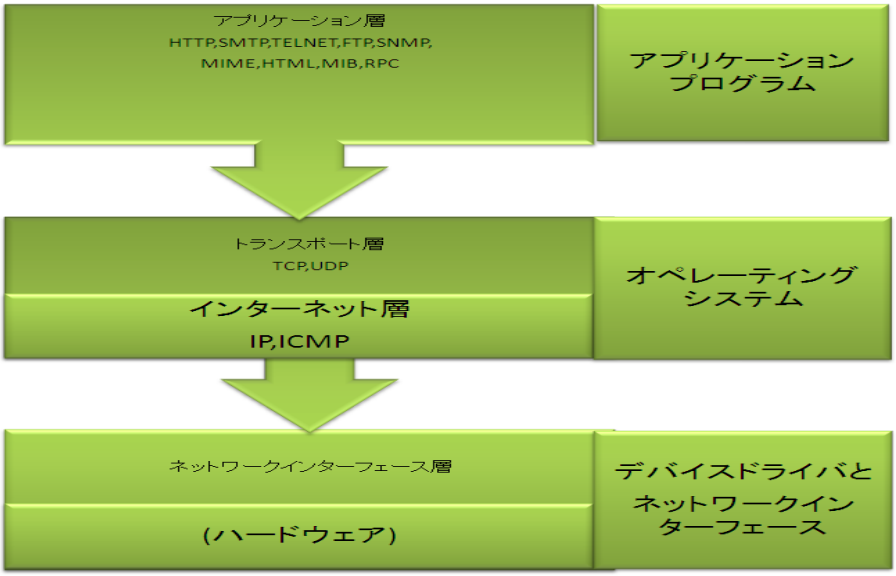

図 2.2: TCP/IP 階層モデル

2.2.5 OSI 参照モデルと TCP/IP の関係

ここでは OSI 参照モデルの機能分類に、TCP/IP の機能を当てはめて話を進める。TCP/IP の階層モデルは OSI 参照モデルとは違うが、TCP/IP に登場するプロトコルも、基本的には OSI 参照モデルに当てはめることができるうえ、OSI 参照モデルはプロトコルの学習に向いているためである。各プロトコルが OSI 参照モデルのどの層に該当するかが分かれば、そのプロトコルが何をするためのものなのかの見当がつく。二つの階層モデルが異なっている理由として、OSI 参照モデルが「通信プロトコルに必要な機能は何か」を考えてモデル化されているのに対し、TCP/IP の階層モデルは「プロトコルをコンピュータに実装するにはどのようにプログラミングしたらよいか」を考えてモデル化されているためである。また、OSI では階層を重視して無駄が生じることはやむを得ないことだと考えていることに対し、TCP/IP では無駄を少なくするために、階層を無視する場合がある。

図 2.3: OSI 参照モデルと TCP/IP の関係

2.2.6 ネットワークインターフェース層

ネットワークインターフェース層とは、一般に「デバイスドライバ」と呼ばれるものである。デバイスドライバとは、OS とハードウェアの橋渡しをするソフトウェアであり、これを利用するコンピュータの OS にインストールすることで、ネットワークインターフェースを利用できる環境が整う。コンピュータの周辺機器をコンピュータに接続しただけでは動作しない。必ずその周辺機器を利用するためのデバイスドライバが必要となる。だが、広く共通化が進んだハードウェアでは、OS 内部に標準ドライバが含まれている場合が多い。標準ドライバがサポートしないハードウェアに関しては、一般に、そのハードウェアを提供するメーカがデバイスドライバを製品に添付するか、あるいはインターネット上で配布する。

2.2.7 インターネット層

インターネット層では IP プロトコルが使われる。これは、OSI 参照モデルの第 3 層であるネットワーク層の役割を持っている。TCP/IP 階層モデルでは、このインターネット層とトランスポート層は、OS の内部に組み込まれている事を想定している。また、インターネットに接続されるルーターは、インターネット層を利用してパケットを転送する機能を実装しなければならない。インターネットに接続される機器でも、ブリッジ、リピーター、ハブの場合は、必ずしも TCP/IP を実装する必要はない。以下にインターネット層で使われるプロトコルを示す。

- IP(Internet Protocol)：パケットを送信するためのプロトコルであり、データリンクの特性を隠す役割も担う。つまり、通信したいホストの間の経路がどのようなデータリンクになっていようとも通信を可能にし、ネットワークの細かい構造が抽象化され、通信相手が雲のようにもやもやとしたネットワークの先に接続されているように見せる。
- ICMP(Internet Control Message Protocol)：IP パケットの配布中に何らかの異常が発生し, パケットを転送できなくなった場合に、パケットの送信元に異常を知らせるために使われるプロトコル。
- ARP(Address Resolution Protocol)：インターネット層とネットワークインターフェース層の間に位置し、IP アドレスと MAC アドレスの変換を行うプロトコル。

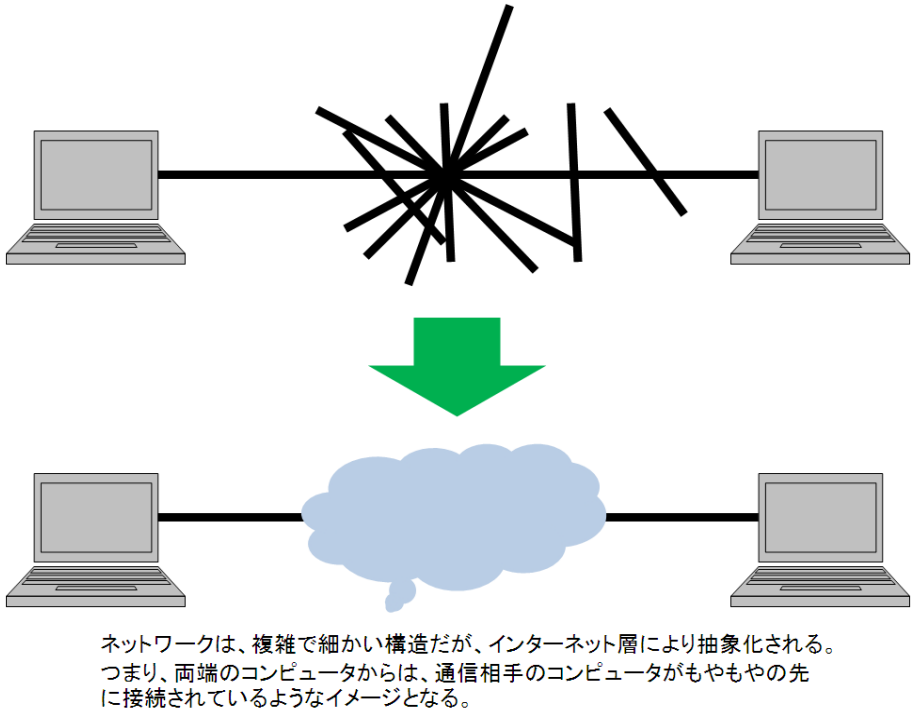

図 2.4: IP パケットの送信

2.2.8 トランスポート層

トランスポート層ではには TCP と UDP の 2 つのプロトコルがある、この層も、基本的には OSI 参照モデルのトランスポート層役割を持つ。また、プログラム間の通信を実現するのが、トランスポート層の役割である。さらに、複数のプログラムを同時に動作させているとき、どのプログラムとどのプログラムが通信して

いるかを識別するのも、トランスポート層の役割である。プログラムを識別するために、ポート番号というアドレスが使われる。TCP/IP 階層モデルでは、トランスポート層も OS の内部に組み込まれている事が想定されている。その理由としては、通信を行っているプログラムを識別し、プログラム間のデータの受け渡しをするためのサービスを提供するのは、OS の役割だと考えられるからである。

- TCP(Transmission Control Protocol)：データの到達を保証する、コネクション型の信頼性のあるプロトコル。経路の途中でデータがなくなったり、順番が入れ替わったとしても、解決するための工夫がなされている。また、ネットワークの帯域幅を有効に利用する工夫や混雑を和らげる仕組みもある。欠点として、コネクションの確立と切断だけで 6〜8 つものパケットが飛び交うため、データの総量が少ない場合には無駄が多くなる。また、様々な工夫・仕組みのため、音声・映像データのように一定間隔で決められた量のデータを送信するのには向いていない。
- UDP(User Datagram Protocol)：TCP とは逆の、コネクションレス型の信頼性のないプロトコル。相手に届かない場合や、相手がネットワークに接続されていないといったケースの場合、送信したデータが届いているかどうかのチェックは行わない。データの量が少ない場合や、ブロードキャストやマルチキャストの通信、音声・映像データなどのマルチメディア通信に向いている。

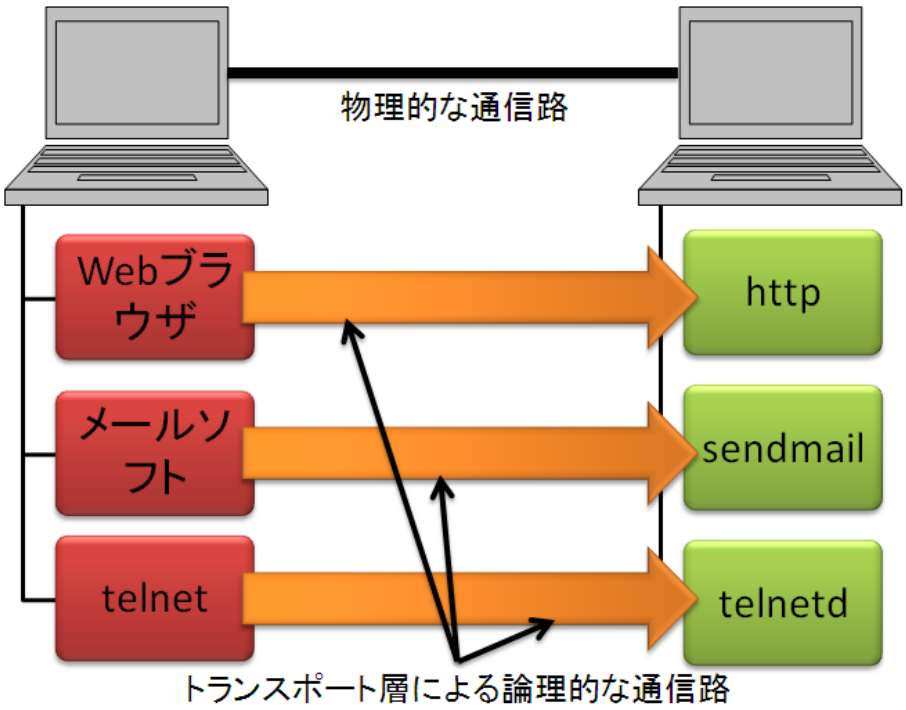

図 2.5: プログラム間での通信

2.2.9 アプリケーション層

TCP/IP 階層モデルでは、OSI 参照モデルのセッション層、プレゼンテーション層、アプリケーション層は、全てアプリケーションプログラムの中で実現されていると考えている。ほとんどの TCP/IP のアプリケーションはクライアント・サーバーモデルで構成されている。サービスを提供するプログラムがサーバーで、サービスを受けるプログラムがクライアントである。この通信モデルでは、いつクライアントから要求が来ても対応できる様に、サービスを提供するサーバプログラムは、あらかじめホスト上で動作させておかなければならない。クライアントは、いつでも好きなときにサービスを要求することができる。もしサーバーが動作していなかったり、要求が集中してサービスが受けられなかった場合は、しばらく待ってから、再度サービスを要求する。

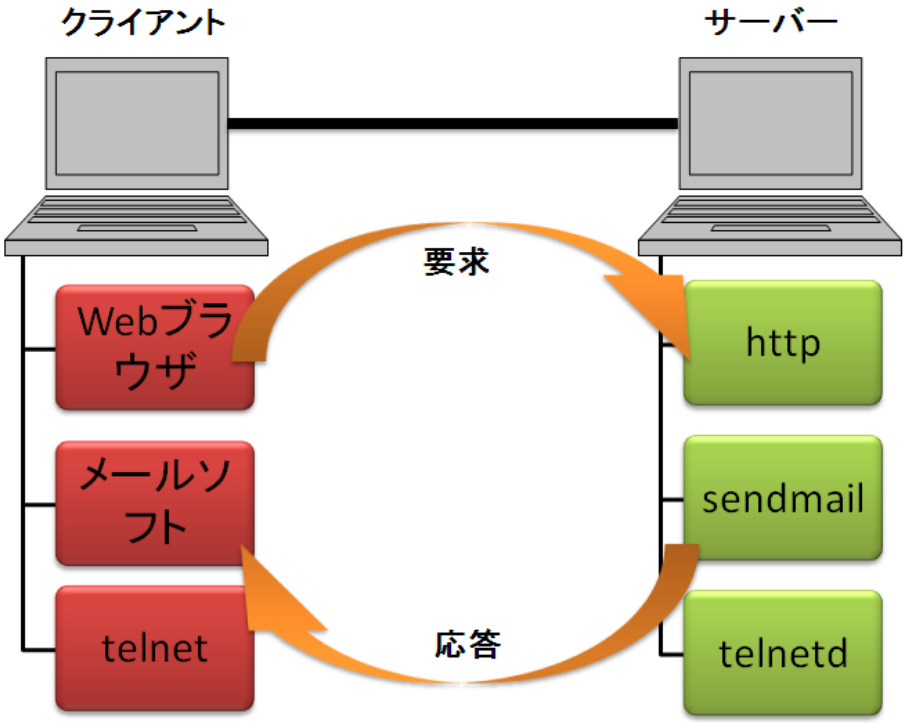

図 2.6: クライアント/サーバーモデル

2.2.10 TCP/IP 通信

TCP/IP はどのように通信を行っているのかを、アプリケーション層から物理媒体までのデータと処理の流れを見ていく。各階層では、送信されるデータにヘッダと呼ばれる情報が付加される。ヘッダにはその層で必要とされる情報、つまり送信元や宛先の情報や通信するデータに関する情報が入っている。下位層から見

ると上位層から受け取るものは全て単なる 1 つのデータとして認識される。以下に TCP/IP 階層モデルの通信例の図を示し、処理の流れを見ていく。

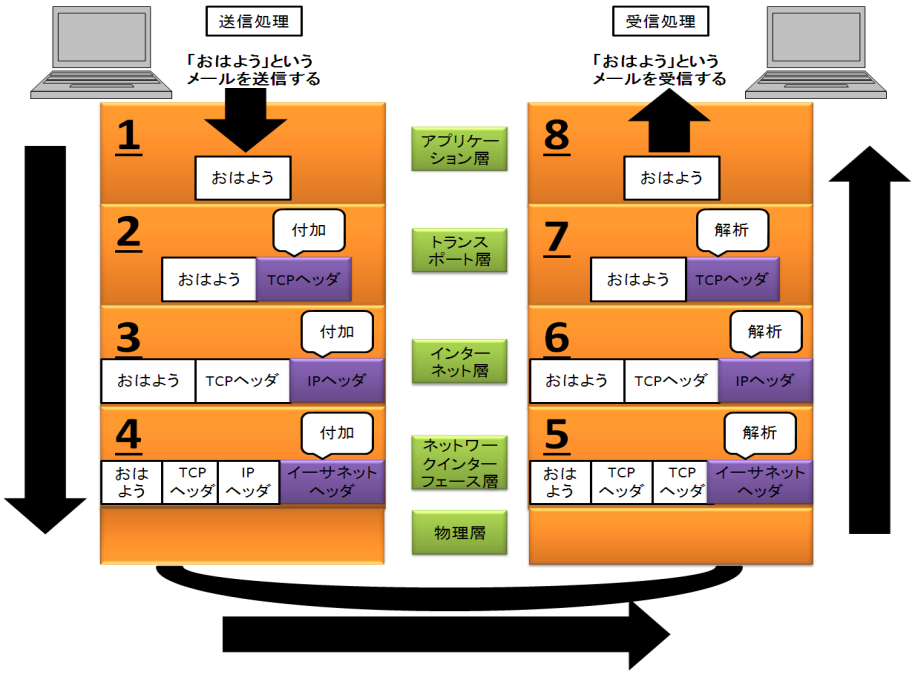

図 2.7: TCP/IP 階層モデルによる通信

送信処理

1. アプリケーション層の処理
   アプリケーションプログラムによりメールを作成し、符号化処理を行う。そして、メールを送信する際に TCP にコネクションの確立を指示する。TCP のコネクションが確立されたら、それを利用してデータを送信する。作成したデータは下位層の TCP に渡され、実際の転送処理が行われる。

2. トランスポート層、TCP モジュールの処理
   TCP は、アプリケーション層の指示によりコネクションを確立/切断したり、データを確実に相手に届けるために、信頼性のあるデータ転送を提供する。ここでは、送られてきたアプリケーションのデータの前に TCP ヘッダを付け、下位層の IP へ送る。TCP のヘッダには、送信ホストと受信ホストのアプリケーションを識別するためのポート番号、パケットのデータが何バイト目のデータなのかを示すシーケンス番号、データが壊れていないことを保障するチェックサムなどが含まれる。

3. インターネット層、IP モジュールの処理
   IP では、TCP から渡された TCP ヘッダとデータを 1 つのデータとして扱う。ここでも、送られてきたデータの前に IP ヘッダを付け、下位層へ送信する。

IP のヘッダには、送信元と宛先の IP アドレス、送られてきたデータが TCP なのか UDP なのかを示す情報が含まれる。送信すべきデータが完成したら、送信すべきルーターやホストを決定し、その機器が接続されているネットワークインターフェースのドライバに、データ渡して、実際に送信処理を行わせる。送信先の MAC アドレスが分からない場合は、ARP を利用して MAC アドレスが調べられる。

4. ネットワークインターフェース層の処理
ここでも、IP から渡されたデータと IP ヘッダは、すべて 1 つのデータとして扱う。このデータにイーサネットヘッダを付け加え、物理層を経由し、相手先へ運ばれる。イーサネットヘッダには、送信元と宛先の MAC アドレス、IP から渡されたデータ群のプロトコルを示すイーサネットタイプが書き込まれる。

受信処理

5. ネットワークインターフェース層の処理
送られてきたデータのイーサネットヘッダの宛先 MAC アドレスが自分宛のものか調べ、自分宛でない場合にはそのフレームは捨てられる。次に、イーサネットプロトコルが運んでいるデータを調べる。ここでは、イーサネットヘッダの前に IP ヘッダを付けていたので、IP を処理するルーチン、つまりインターネット層へデータを渡す。また、処理できないプロトコルの値が送られてきたデータに入っていた場合、データは捨てられる。

6. インターネット層、IP モジュールの処理
送られてきたデータの宛先 IP アドレスを調べ、自分のホストの IP アドレスであれば、そのまま受信し、上位層へ送る。TCP の場合なら TCP ルーチンに、UDP の場合であれば UDP の処理ルーチンにデータを送る。自分宛でない場合は、次に送るホストやルーターを調べて転送処理を行う。

7. トランスポート層、TCP モジュールの処理
チェックサムを計算し、データが壊れていないか、順番通りに受信しているかを確認する。データが問題なく届いたと判断された場合、送信ホストにデータが届いたことを確認するために「確認応答」を返す。この確認応答が、データを送信したホストに届かない場合には、届くまでデータを繰り返し送信する。そして、ポート番号を調べて、通信を行っているアプリケーションプログラムを特定し、データを渡す。

8. アプリケーション層の処理
受信したデータを解析し、自分宛のメールであることを確認し、ハードディスクにメッセージを格納する。格納し終えたら、処理が正常に終了したこと

を、送信元のアプリケーションに伝える。メッセージを格納できなかった場合は、異常終了のメッセージを送信する。以上全ての処理を経て、ディスプレイ上に「おはよう」と表示される。

プロトコルのパケットは、プロトコルが利用するヘッダと、そのプロトコルの上位層が利用するデータから構成されている。ヘッダの構造には、上位プロトコルを識別するフィールドの位置やビット数、チェックサム計算法やどこのフィールドに入れるかなどが決められている。もし、通信する双方のコンピュータでプロトコルの識別番号やチェックサムの計算方法が違えば通信は不可能になる。このように、パケットのヘッダは、プロトコルの仕様が目に見える形で存在している。

2.3 遠隔ログイン

遠隔ログインは、TSS のような環境を実現するアプリケーションである。TSS では、中央に処理能力の高いコンピュータが存在し、そのコンピュータに複数の端末が、端末専用の通信回線で接続されていた。このような関係を、自分の使用しているコンピュータとネットワークの先に接続されているコンピュータの間で実現したものが、TELNET プロトコルである。TELNET を利用することで、ネットワーク上にある全てのコンピュータにログインすることが可能であり、それにはログインするコンピュータに、自分のログイン名とパスワードが登録されている必要がある。TELNET による通信は、ネットワーク上の全てのコンピュータとの間に、通信回線を仮想的に敷設したようにイメージすることができる。

2.3.1 TELNET

TELNET は、1 つの TCP コネクションを利用し、これを通して、相手のコンピュータにコマンドが文字列として送信され、相手のコンピュータで実行される。これは、自分のキーボードとディスプレイが相手のコンピュータの内部で動作している「シェル」に接続されているイメージである。シェルとは、キーボードやマウスからユーザーのコマンドを解釈して、それを OS に実行させるインターフェースのプログラムである。TELNET のサービスは、ネットワーク仮想端末 (NVT) の機能と、オプションのやり取りをする 2 つのサービスに分けることができる。[13]

2.3.2 ネットワーク仮想端末

ネットワーク仮想端末 (NVT: Network Virtual Terminal) とは、どのような種類のコンピュータの組み合わせでも、画面表示に異常が起こらないようにする機能である。TELNET では、NVT を定義し、端末に合わせて画面描画の制御を行う。端末により、画面の文字コードや、画面を消去したりスクロールさせたりするコードが異なる。しかし、TELNET を使用するときには、NVT に従わせることで、どの端末でも画面が乱れることなく作業するこができる。NVT を定義したことにより、現在存在する端末も、今後登場する新たな端末も、動作を保証することができる。

2.3.3 オプション

TELNET では、ユーザーが入力した文字以外にも、オプションをやり取り機能が用意されている。NVT を実現するための画面制御情報は、このオプションの機能を利用して送信される。また、TELNET には行モードと透過モードの 2 つモードがあり、これらの設定も、オプションの機能を利用して設定される。行モードでは、1 行分のデータをまとめて送る。透過モードでは、1 文字毎に送信する。

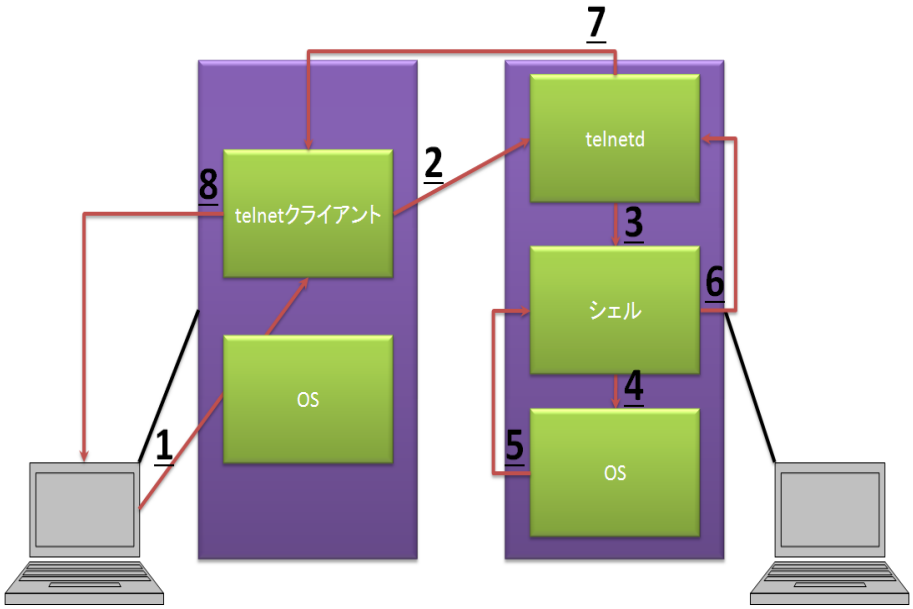

図 2.8: TELNET の仕組み

厳密には、すべてのプロセスで OS を経由する。

1. キーボードから文字列が入力される。
2. 入力された文字列を、行モードや透過モードの処理をして telnetd へ送信する。
3. シェルに文字列を送信する。
4. シェルから文字列を解釈してプログラムを実行。
5. シェルへ実行結果を渡す。
6. telnetd へ実行結果を送信。
7. 実行結果を、行モードや透過モードの処理をして telnet クライアントに送信する
8. NVT の設定に従い実行結果を画面へ出力する。

2.4 シリアルバス

シリアルバスは、コンピュータ内部で使われるデバイス同士を接続するバスである。パラレルバスに比べて接続端子数が少なくて済み、信号線は 4 本で構成されており、比較的低速なデータ転送を行うデバイスに利用される。それほど速度を必要とされないデバイスに関しては小型化できることから、少ないピン数で接続できる形態が望まれた。このような背景から、いくつかのシリアルバス規格が提唱された。シリアルバスの一例を以下に示す。

- SPI
- I2C
- MicroWire

2.4.1 SPI

SPI(Serial Peripheral Interface) は、双方向シリアル通信方式のひとつであり、シリアル通信の中では高速で安定しており、最も使いやすい。USB 通信の非同期式通信と違って同期式通信である。非同期式通信の場合は、それぞれのデバイスが独自のクロック信号を持っているので、安定した非同期通信を実現するためには、使われているクロック周波数は高精度である必要がある。以下に SPI の信号線を示す。[2]

- SI(serial input)
- SO(serial output)
- SCK(serial clock)
- CS(ship select)

この内、SCK と CS を制御するデバイスをマスターと呼び、もう片方のデバイスをスレーブと呼ぶ。SPI では、マスターとスレーブ間で一対一通信が行われ、同期のために 1 つの 1 つのクロック信号を用いる。また、そのクロック信号は常に、マスター側からスレーブ側へ供給する。SPI は双方向シリアル通信なので、シリアル入力と出力があるが、すべてのデータは、マスターが供給するクロック信号に従って、スレーブへの入出力を実現する。1 つのマスターから複数のスレーブへの通信を実現するために、スレーブ側にはチップセレクトと呼ばれる信号が用意してある。これの選択信号によって、マスターは複数のスレーブの中から 1 つのスレーブを選ぶことができる。[11]

2.5 組み込みシステム

組み込みシステム (Embedded system) とは、特定の機能を実現するために、電子機器に組み込まれるコンピュータシステムのことである。現在では、電子制御を必要とするほとんどの製品に用いられている。パソコンは汎用性を持つので、組み込みシステムとは対比される。組み込みシステムが発達する以前の電子制御は、主に電子回路や機械的なハードウェアによる、直接的な機能により実現していた。そこに組み込みシステムの登場により、ソフトウェアの利用によって、複雑な機能も実現できるようになり、回路の変更を最小限に抑えられるようになり、全体的なコストが低減できることから、広範囲の製品に導入されることになった。[5]

2.5.1 特徴

- 汎用性のあるパソコンと比べて最終製品が多岐に亘るため、非常に数と種類が多い。
- 専用のハードウェアを開発することが多く、対応するデバイスドライバも作る必要がある。
- 汎用システムとは異なり、制御を行う際にリアルタイム制御が重要となる。
- 大量生産される製品の場合には、コストが非常に重要となるので、必要最低限のメモリと安価な CPU で動作する必要がある。
- 専用のハードウェアに専用のソフトウェアが搭載されて製品となるので、テスト工程でハードウェアとソフトウェア、両方の検証が重要になる。
- 特に低資源の環境では OS を使用しないことが多い

2.5.2 ISP

ISP(In-System Programming) とは、プログラム可能な電子部品において、事前にプログラムを書き込んでからシステムに組み込むのではなく、組み込み済みの状態でプログラムを書き込むことである。この機能の利点は、システムの組み立て前に書き込み段階を別途に設ける必要がなく、電子機器の製造者がプログラムの書き込みとテストを 1 つの工程で行えることである。また、製造者がシステムの製造ラインでチップへの書き込みができるので、製造期間の途中でもコードや設計の変更が可能である。一般的に ISP をサポートしたチップは、ライタとはシリアルプロトコルで通信する。

2.6 AVR

AVR とは、Atmel 社が製造と販売を行っている、マイクロコントローラのシリーズ名である。AVR マイコンは 8 ビットであり、ここでいう 8 ビットとは、マイコンの中にあるチップのバスが 8 ビットという意味である。チップとは、記憶や演算、入出力用の回路を半導体の小さな坂にしたものである。このチップが一つだけなので、ワンチップマイコンに分類される。ワンチップマイコンは、比較的小型で価格も安く、多くの製品に使われている。また、マイコンを機器に搭載することを、組み込みと呼ぶ。チップには、以下の機能が搭載されており、書き込まれたプログラムにより制御される。[3]

- CPU
- SRAM
- EEPROM
- フラッシュROM
- I/O ポート
- クロックジェネレータ
- A/D コンバータ
- タイマ

本研究では、megaAVR シリーズの Atmega168 を使用している。

2.6.1 特徴

- 1 つの命令を CPU クロック 1 回分の時間で実行させることができる。例えば、CPU クロックが 20MHz の AVR マイコンでは、1 つの計算を 50n 秒でこなすことができる。
- ISP に対応しており、非常にプログラミングしやすい。
- 内部に発振子を搭載しているため、クロックを外部から取り入れなくても動作させることができる。使用する発振子は内部/外部を切り替えることができる。
- C 言語でのプログラミングを意識しており、アセンブラを含んだ開発環境の「AVR Studio」が無償で提供されている。
- AVR Studio に WinAVR を組み合わせて使用することにより, 楽に開発環境を整えることができる。

- AVR という名前は、チップを設計した Alf Egil Bogen 氏と、Vegard Wollan 氏の名前と、RISC(Reduced Instruction Set Computer) から取られている。

2.7 割り込み処理

3.2.1 で述べた通り、組み込みシステムにおいては、リアルタイム制御が重要となる。メイン処理をプログラムで常にチェックし、必要な時に処理を実行する方法もあるが、ソフトウェアは様々な処理を行っており、リアルタイム性を保証することはできず、また、常にチェックするのも効率が良くない。[9] そこで、割り込み機能を導入する。割り込みとは、通常のプログラムの流れから、強制的に別のプラグラムを実行することのできる仕組みである。[5] 割り込み機能は、多くのマイコンに用意されている。割り込みを使用することで、リアルタイムでの処理が可能となり、信頼性・効率性を上げることができる。例えば、割り込み機能を使うと、メインルーチンでは定期的にスイッチのチェックをすることなく、指定したピンに繋がれたスイッチが押されると、メイン処理を中断して、割り込み処理を実行させることができる。割り込み処理後は、すぐにメイン処理の続きが再開される。非常に便利な割り込みだが、使い方を間違えると、タイミングが絡む難解な問題となりやすいところであり、十分な設計と検証が必要となる。[12]

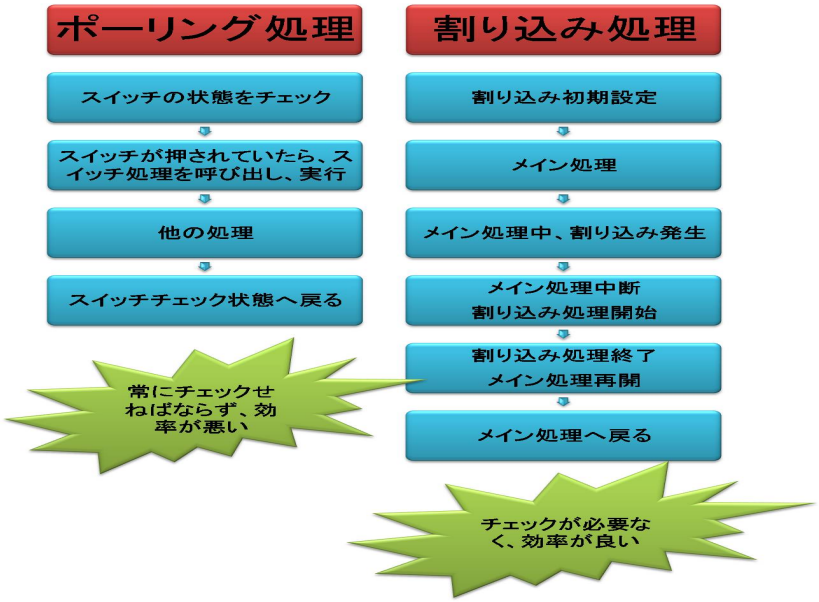

図 2.9: 処理の違い

2.8 ダイオード

発光ダイオードは、LED(Light Emittiong Diode) と呼ばれ、半導体を用いた pn 接合で作られている。[14] LED に電流を流すと、LED のアノードとカノード間に順方向電圧が生じ発光する。発光する明るさには mcd(ミリカンデラ) という単位が使われる。明るさは数 mcd から数万 mcd まであり、数 100mcd 以上のものを高輝度 LED、1000mcd 以上のものを超高輝度 LED と呼ぶ。超高輝度 LED とフォトダイオードを比較すると、照明用の LED がそのまま測定用として使える、超高輝度 LED の方が入力光に対する電圧の変化が大きい等の理由により、本研究に適している。[10]

2.8.1 受光スペクトル

過去に本研究室において、超高輝度 LED の受光スペクトルの実験が行われた。光源から分光器を通過して単色となった光を、超高輝度 LED に照射、波長を 400 700mm で変化させ、それに伴って変化する電圧値を測定した。実験が行われた結果、赤色超高輝度 LED が最も高い電圧値を示すとわかった。その結果を踏まえ、過去の研究では赤色超高輝度 LED が使われた。

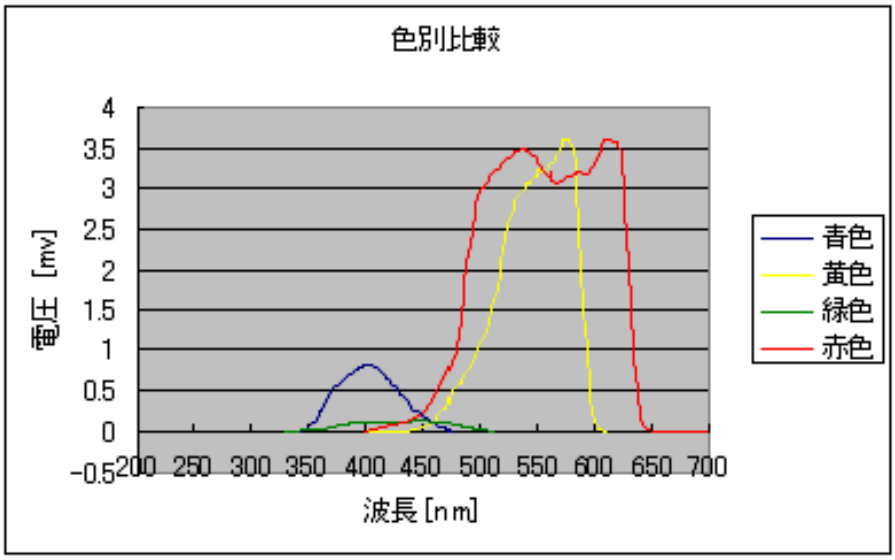

図 2.10: 受光スペクトル

3 提案

前項2.8において、照明用のLEDがそのまま測定用として使えると記述した。これは、超高輝度LEDは電流を流し発光させることもできるが、光を当てた際に起電力を発生させることができる、ということである。本研究では、超高輝度LEDの発光よりも、起電力の発生に注目した。超高輝度LEDに当てる光の明るさを変化させることにより、生じる起電力に変化が起きる。この特性を利用し、光の環境の変化を感知することにより、動作検知を可能にするセキュリティシステムを構築する。

3.1 μIP とセンサ

システムの構成は、超高輝度LEDをセンサとして用いて、μIPをATmega168に実装することにより、電圧取得アプリケーションプログラムと、TCP/IPによる通信を構成、取得したデータをホストコンピュータへ送信させる。人の動作により変化した影を、光の明るさの変化とし、ホストコンピュータで継続的に送られてくる電圧値を表示し、電圧値に変化があった場合は、動作の検知となるような構成である。

3.1.1 開発環境

開発環境は以下の通りである。

- AVR Studio 4(統合開発)
- WinAVR(Cコンパイラ)
- ATMEGAマイコンボード(書き込み用ライタ)

3.1.2 開発の流れ

1. μIPのソースコード内のアプリケーション部を、超高輝度LEDの電圧変化を取得させるプログラムに変更する。
2. コンパイルして生成したHEXファイルのμIPのソースコードを、ATMEGAマイコンボードを用いて、Atmega168に書き込む。ATMEGAマイコンボードとAtmega168はSPIで接続させ、Atmega168は取り外しが容易である。
3. μIPを実装したAtmega168を、SPIによる接続を用いてイーサネットコントローラと接続し、Atmega168に超高輝度LEDを接続し、実験装置を作製。
4. 実験装置とホストコンピュータを、ルーターを介して接続する。
5. TELNETにより、実験装置にアプリケーションを実行させる。

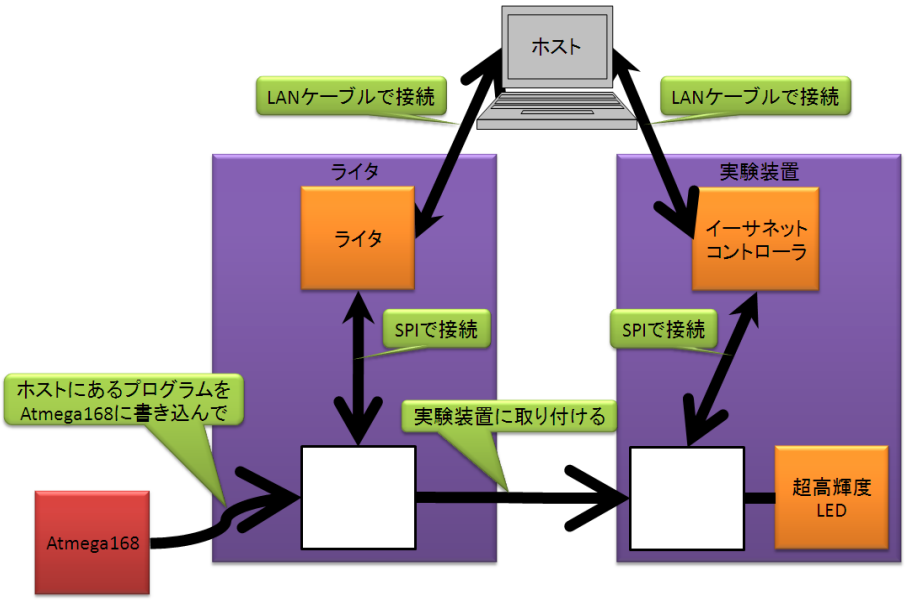

図 3.1: 開発の流れ

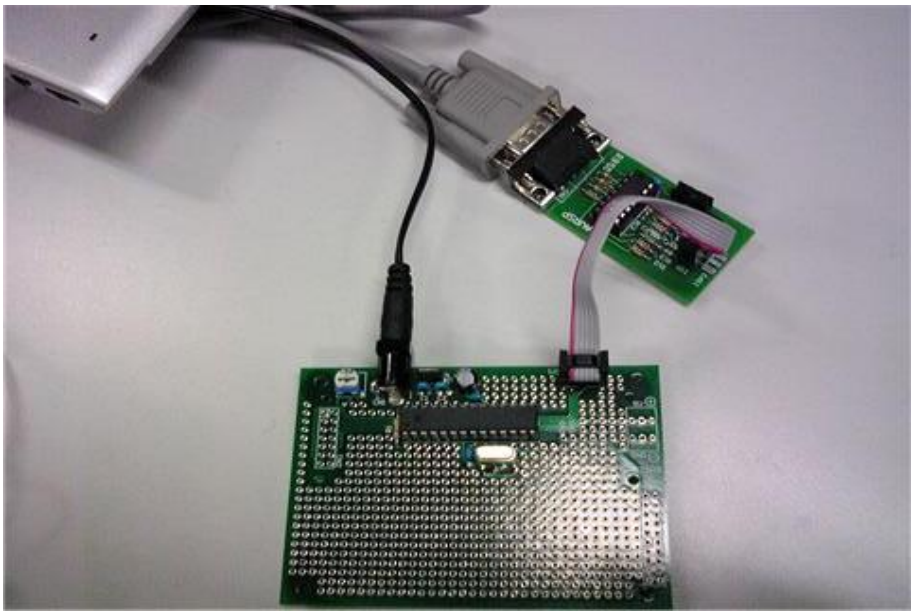

図 3.2: ATMEGA マイコンボード

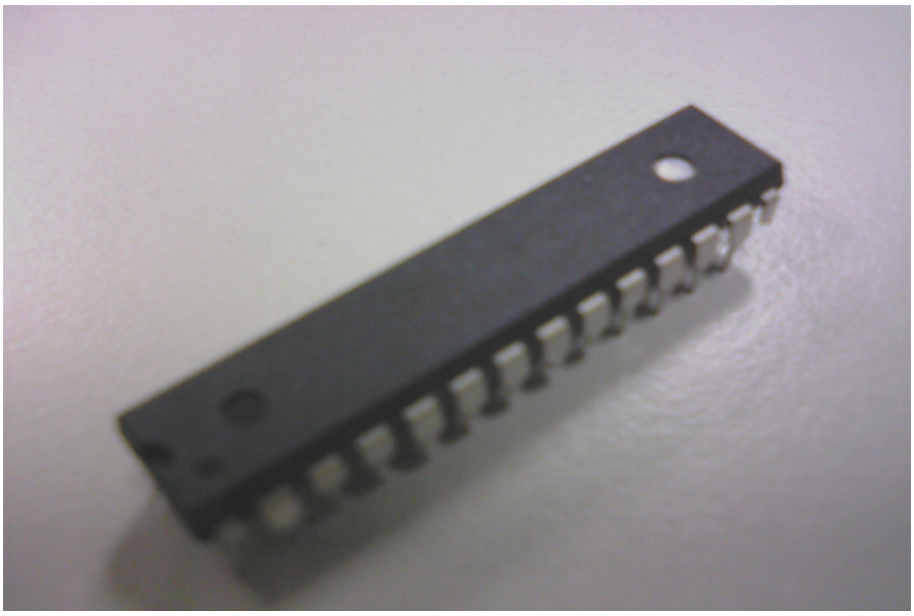

図 3.3: Atmega168

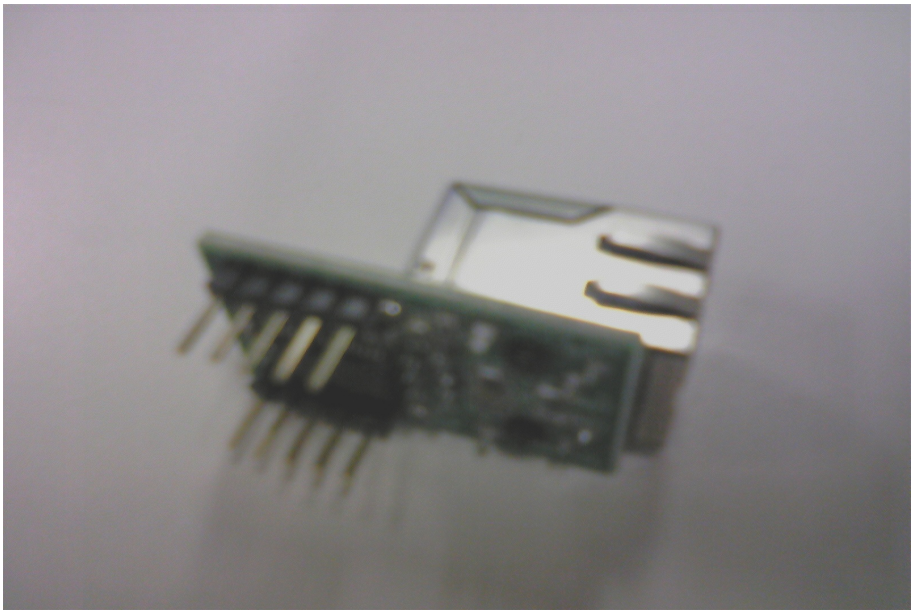

図 3.4: イーサネットコントローラ

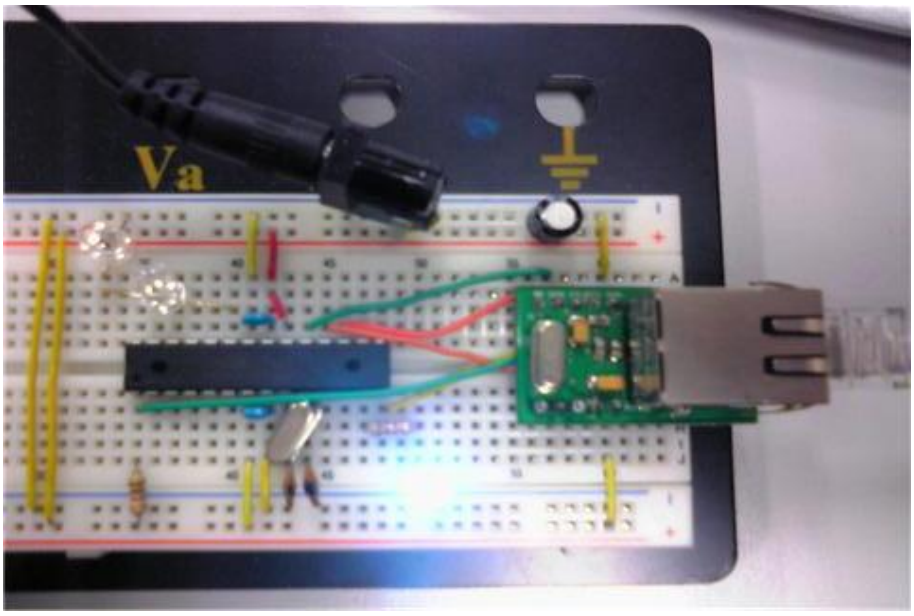

図 3.5: 実験回路

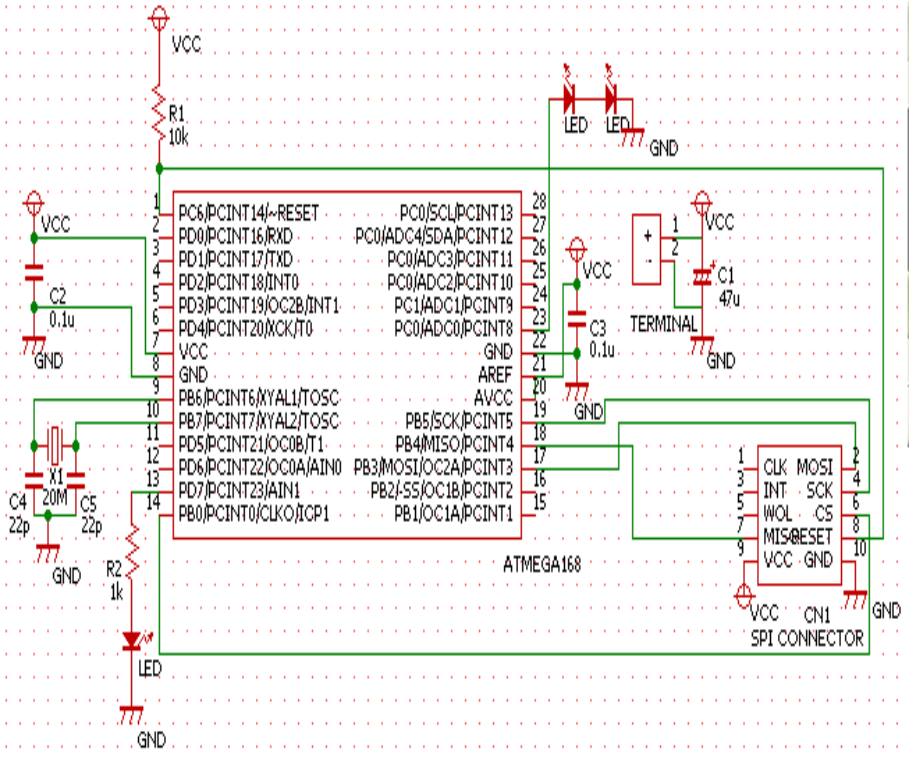

図 3.6: 実験回路図

3.2 センサネット

図3.5の実験装置を複数作製し、パソコンとSPI接続することにより、センサネットを構築した。センサからの取得した電圧値の変化を読み取り、AD変換したデータをパソコンへ送信するシステムである。センサに対して光の遮断を行い、取得した電圧値の変化をより素早く大量に送信することを目的としている。なお、実験装置のIPアドレス、MACアドレスは、実験装置ごとにμIPの設定ファイルを変更し、再コンパイルした後に各Atmega168にプログラムの書き込みを行った。今回は実験装置を2つ作製した。

3.3 システムの処理

ここでは、センサネットシステムの処理の流れについて記述する。

1. 全ての装置に対し電圧の取得命令を送信する
2. センサの電圧の変化を比較するために、センサの電圧の初期値を取得する
3. 再び全てのセンサに電圧取得命令を送信する
4. 初期値の電圧と比較し、電圧に変化があればホストコンピュータへ電圧の値を送信する
5. 装置から送信されてきた電圧を取得し、環境の変化が起こったか判定する
6. 環境に変化があれば装置の位置を特定し、表示する
7. 再び全てのセンサに電圧取得命令を送信する

4 実験と考察

4.1 超高輝度 LED の選択

本研究室にある超高輝度 LED の性能の評価を行った。様々な色・輝度があるため、最も本実験に適した超高輝度 LED を特定させるためである。実験方法は、電圧値を 1 回取得する毎に、実験装置の超高輝度 LED を付け替え、それぞれの超高輝度 LED で、電圧値の変化を確認する。超高輝度 LED には遮断物を近付けず、取得できる最大の電圧値を比較する。データは 8 ビットで構成され、取得できる電圧値の最大値は 255 であり、電圧取得画面の 255 は 3V に相当する。

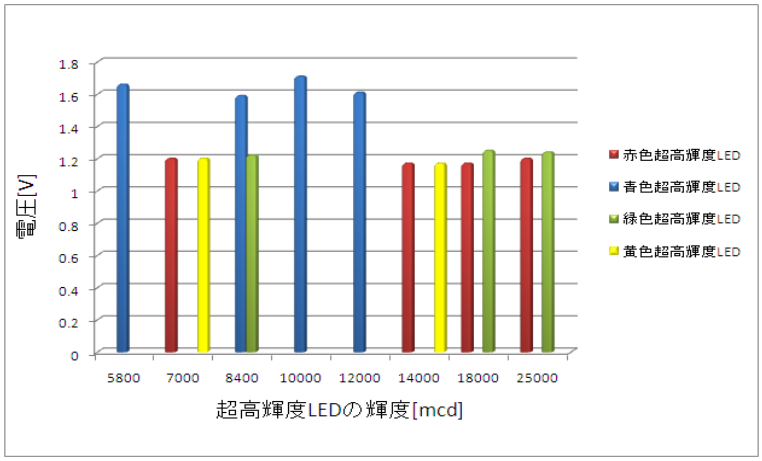

図 4.1: 超高輝度 LED の最大取得電圧値

実験を行った結果、青色 LED が最も高い電圧値を取得できた。また、青色 LED を二つ使った場合のみ、取得できる電圧値の最大値 255 に到達することができるため、本研究の実験装置のセンサには、青色超高輝度 LED を二つ、直列に繋いで、センサとして使用した。

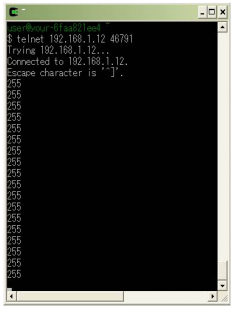

図 4.2: 青色超高輝度 LED2 個の最大取得電圧値

4.2 実験装置の使用

遮断物を近付け、光の環境に影響があった場合、以下の様な結果となる。

図 4.3: 青色超高輝度 LED2 個の電圧取得値

また、図 3.5 に示した様に、光の環境の変化を装置で視覚化するために、電圧値に変化があった場合は発光するようにした。これは、電圧値が 255 未満だった場合発光するという仕様である。これにより、電圧値が 255 未満になった場合、移動体の検知をしたことになる。取得した数値を見なくても、電圧値が 255 未満の時に発光するため、発光した場合も、移動体の検知をしたことになる。また、電圧値の取得は 0.2 秒に 1 回行われている。

4.3 データのバッファリング処理

以上の通り、電圧値の取得と送信は、0.2 秒に 1 回行われる。このスピードを高めるため、取得データのバッファリングを行った。1 データ毎に送信しているため、オーバヘッドが増加し、十分なサンプルを得ることができていないと考えたためである。
μIP のアプリケーション部に for 文を用いて、データをバッファリングし、規定の値 (送信したいデータの数) までデータがたまったら、まとめて送信する処理である。しかし、1 つのデータを、既定の数 (送信したいデータの数) に増やして送信するという結果になってしまい、想定されていたシステムとはならなかった。

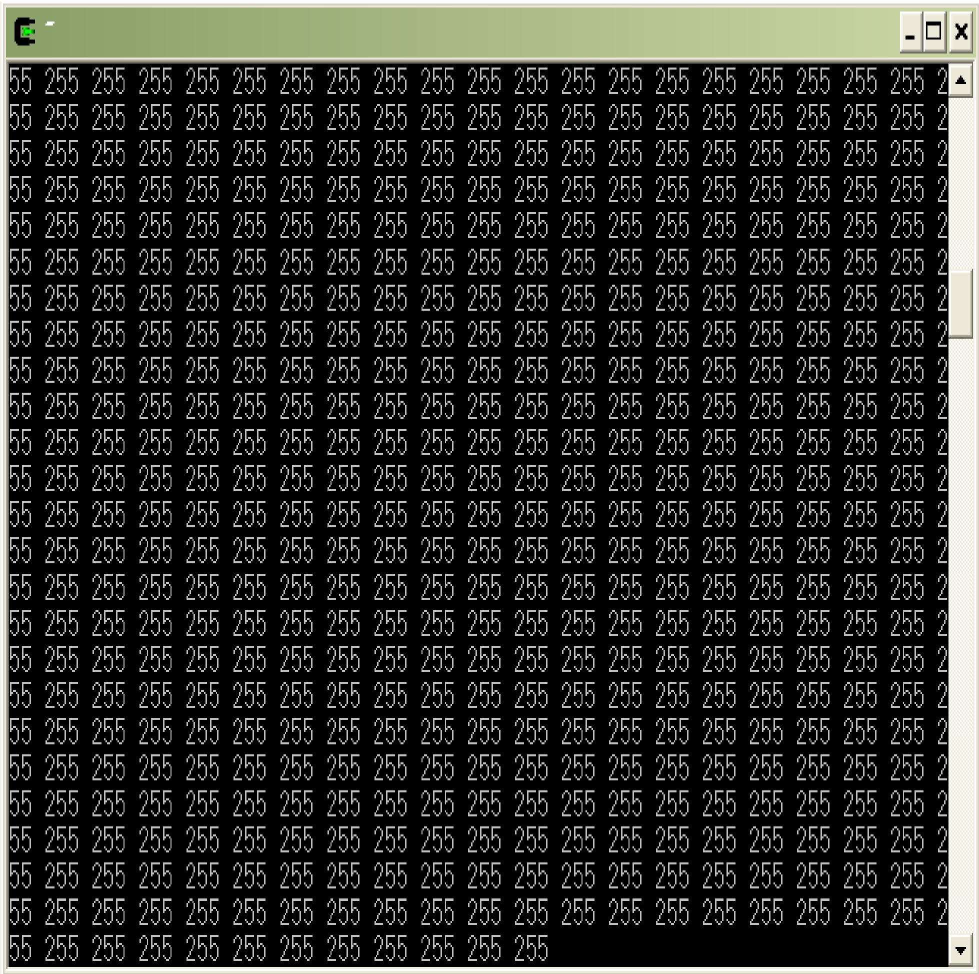

図 4.4: データのバッファリング処理

4.4 割り込みによるデータのバッファリング処理

次に、割り込み処理を用いてデータのバッファリングを行った。

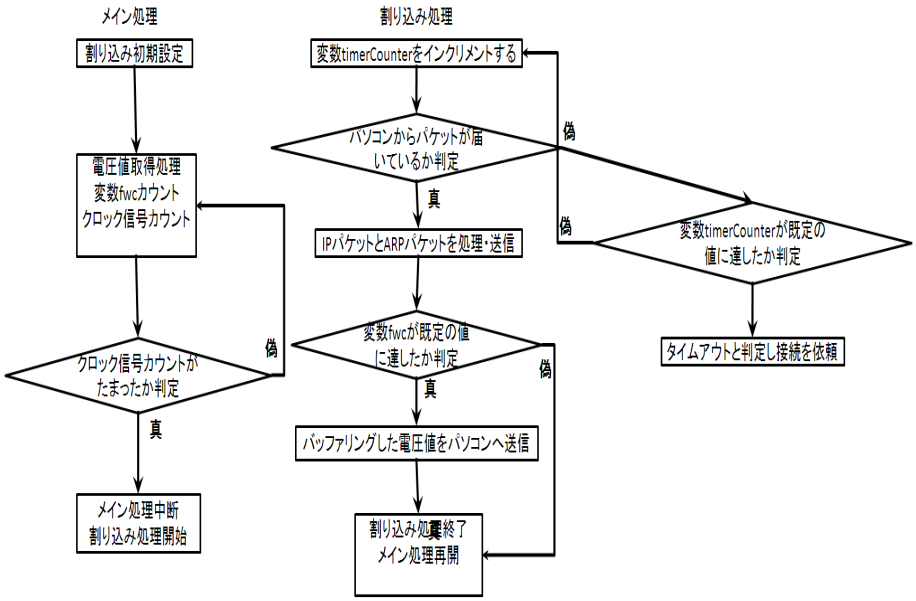

図 4.5: 割り込み処理の流れ

まず、割り込み処理を行う際、初期設定が必要となる。ここでは、グローバル変数 fwc を定義、レジスタの初期値設定、使用するピンを選択、プリスケーラ設定 (クロックの分周を選択) である。初期設定が終わったら、メイン処理が始まる。電圧値を取得する毎に変数をインクリメントしていき、それとは別の周期でクロック信号をカウントしていく。クロック信号が、プリスケーラ設定で選択した値まで到達したら、現在の処理、レジスタの状態を保存し、メイン処理を中断、割り込み処理が始まる。

割り込み処理が始まったらまず、変数 timerCounter の処理を行う。これにより、接続が継続されているかを判定している。接続が問題ないと判定された場合、メイン処理でインクリメントされ続けていた、変数 fwc が既定の値 (送信したいデータの数) に達しているかを判定する。もしデータ数が十分に溜まっている場合は、そのデータの集まりをパソコンへ送信する。データ数が十分に溜まっていない場合は、割り込み処理を中断し、メイン処理を再開する。この割り込み処理により、リアルタイムでの処理が可能となり、データ取得効率を上げることができる。

実験を行った結果、5 秒で 25 個のデータを送信するといったように、0.2 秒で 1 個取得していることには変化がなかった。

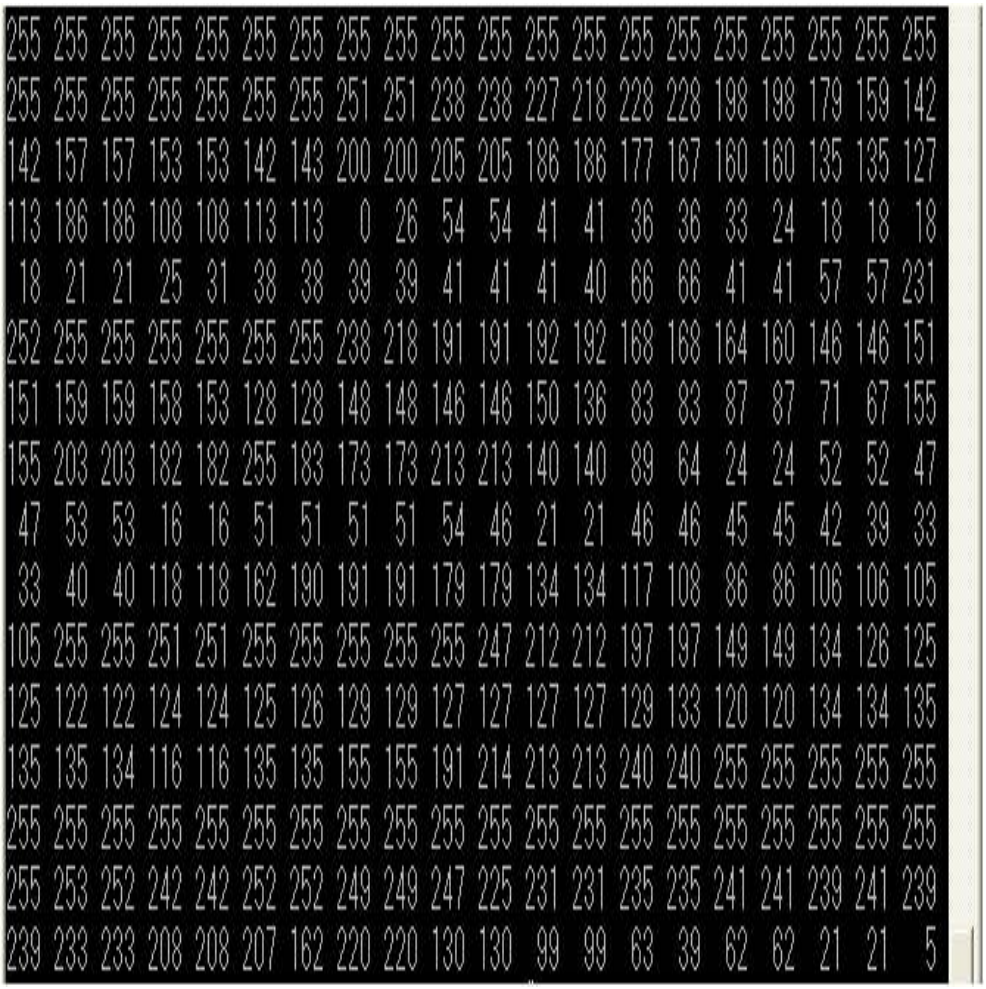

図 4.6: 割り込みによるデータのバッファリング処理

5 結論

本研究では、μIP のアプリケーションプログラムを変更し、Atmega168 に書き込み、超高輝度 LED をセンサとして組み合わせ、センサネットを構築し、光の遮断により電圧値を変化させ、パソコンへ送信した。だが、1 データ取得する毎に送信しているため、オーバヘッドが増加し、十分なサンプルを得ることができなかった。そこで μIP を解析し、割り込み処理を用いてデータを一定数バッファリングすることにより、データ数を増やすシステムを考案した。しかし、μIP では、TCP/IP の処理も割り込み処理により実現している。一方、センサの計測を一定間隔で行うためにも割り込み処理が必要となる。これを実現するためには多重割り込みを行う必要があるが、何れの処理も同程度の優先度が要求され、リアルタイム性を損なうことなく、非同期でこれらの処理を行うことは ATmega の能力では困難になる。そこで、μIP の処理のすべての割り込み処理のタイミングに同期して計測を行い、データをバッファリングする。データの送信はポーリング処理により、バッファのデータ量を監視し、一定量蓄積された時点で送信処理を行う。すべての割り込み処理のタイミングを同期させることができれば、処理間の衝突を防ぐことができ、センサネットワークの効率化が実現できる可能性がある。そのためには、今後はより μIP の解析を進め、各処理のタイミングを操作していく必要がある。

謝辞

本研究を行うにあたり、終止熱心にご指導して頂いた木下宏揚教授と、問題に対する的確な助言をして頂いた鈴木一弘氏、ご多忙の折研究室に足を運び、様々な面で有益なご助言をして頂いた森住哲也氏に、深く感謝いたします。さらに、公私にわたり良き研究生活送らせて頂いた木下研究室の方々に感謝いたします。

参考文献

[1] 竹下 隆史・村山 公保・荒井 透・苅田 幸雄/共著：”マスタリング TCP/IP” 入門編 第 2 版
オーム社 1998 年-5 月

[2] 武藤 佳恭/著：”インターネット・ガジェット設計”
オーム社/雑誌局 2008 年-4 月

[3] 松原 拓也/著：”AVR マイコン活用ブック”
電波新聞社 2007 年-1 月

[4] 笠野 英松/監修, マルチメディア通信研究会/編：”インターネット RFC 辞典”
アスキー出版局 1998 年-10 月

[5] 土井 滋貴/著：”試しながら学ぶ AVR 入門”
CQ 出版社 2008 年-3 月

[6] Main Page - uIP
http://www.sics.se/~adam/uip/index.php/Main_Page

[7] SICS
http://www.sics.se/

[8] LA Skater - uIP-AVR
http://www.laskater.com/projects/uipAVR.htm

[9] Getting Started Notes
http://avrwiki.jpn.ph/wiki.cgi?page=GettingStarted+Notes

[10] Using led as solar cell
http://homepage3.nifty.com/s-danjo/electro/led/led.htm

[11] http://moon.gmobb.jp/hero/PDF/mega88.pdf
http://moon.gmobb.jp/hero/PDF/mega88.pdf

[12] 技術レポート「マイコン用ソフトウェアの割り込み処理」
http://www.softech.co.jp/mm_090513_firm.htm

[13] telnet の使い方 - PukiWiki
http://zenno.org/pukiwiki/index.php?telnet%A4%CE%BB%C8%A4%A4%CA%FD

[14] ダイオードは一方通行？
http://part.freelab.jp/p_diode.html#photo

質疑応答

Q. 使用しているチップを変更すれば取得効率は上がるのか?

A. 本研究では高機能を実現するため、CPU クロックではなく、外部発振子を用いた。この発振子を更に高機能の物へ変更すれば、取得効率を上げることができる可能性がある。