Linux組み込みボードゼミ

室蘭工業大学　ロボットアリーナW．G．
しくみ情報　倉重健太郎

目 次

図 目 次

第1章　Linux組み込みボード基本設定

1.1　準備

1.1.1　ネットワークの申請

室蘭工業大学内でのネットワークを使用する場合、情報メディアセンターへ IP 申請を行う必要がある.

1.1.2　物品確認

・Armadillo-300 一式
「はじめにお読みください」に一覧があるので内容物を確認.

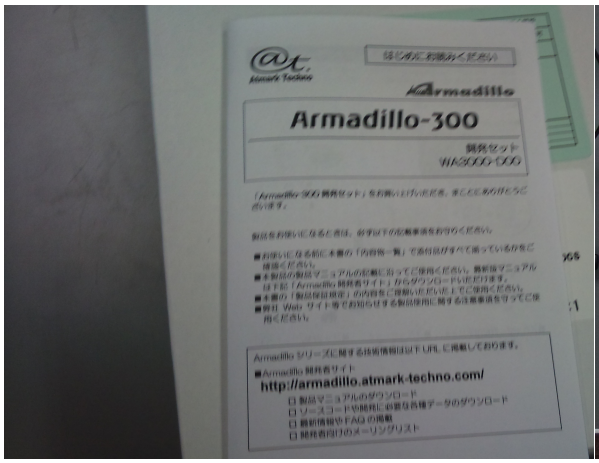

図 1.1: はじめにお読みください

図 1.2: Armadillo300 内容物

・USB-シリアル変換ケーブル

図 1.3: USB-シリアル変換ケーブル

・サーボ・センサコントローラ

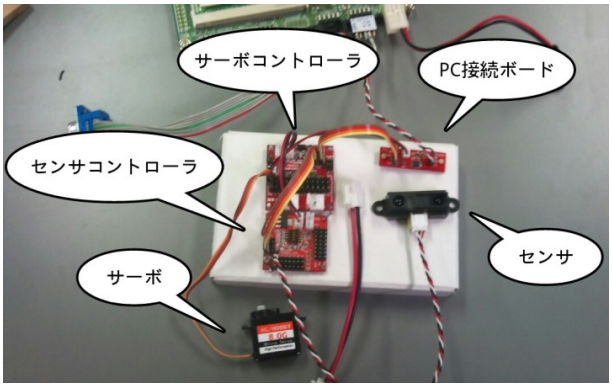

図 1.4: サーボ・センサコントローラ

表 1.1: 物品一覧

商品名	型番	会社名	個数	備考
AC-DC アダプター 5V			1	
電源接続ケーブル			1	AC-DC アダプターと サーボコントローラを接続
RC サーボコントローラ	AGB65-RSC	浅草ギ研	1	付属品：AGB65 電源ケーブル, AGB65 接続ケーブル, ジャンパピン x2
センサー入力ボード	AGB65-ADC	浅草ギ研	1	付属品：AGB65 電源ケーブル, 電源延長ケーブル, 通信ケーブル
PC 接続ボード	AGB65-232C	浅草ギ研	1	付属品：AGB65 通信ケーブル, 5V 接続ケーブル, PC ケーブル
RC サーボモータ	R/C HOBBY 8.0G	R/C HOBBY	1	
PSD 距離センサー	AS-PSD	浅草ギ研	1	距離レンジは 10cm〜80cm
シリアル接続ケーブル			1	Armadillo-300 のシリアルポートと AGB65-232C を接続

1.1.3 使用ソフト

・通信用ソフト
  ・TeraTerm(Windows)
    ・窓の杜
      `http://www.forest.impress.co.jp/lib/inet/servernt/remote/utf8teraterm.html`
      等から DL，インストール.
  ・minicom(Linux)
    ・apt-get でインストール (Debian).

・USB-シリアル変換ケーブルドライバ
  ・付属の CD からインストール.
  ・最初に USB-シリアル変換ケーブルドライバの USB を PC に接続.
  ・ドライバを求められるので，付属 CD を PC に入れ，インストール.

1.2　初期接続設定

・写真のように接続する.

図 1.5: 接続図

・特に注意するコネクタ.

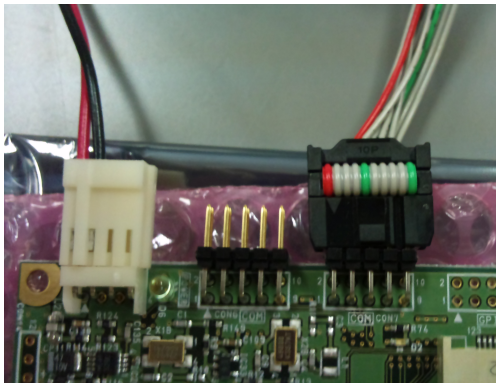

図 1.6: 注意するコネクタ

1.2.1　TeraTerm で起動確認 (Windows)

TeraTerm の設定

・TeraTerm 起動.
・「TeraTer:新しい接続」の画面にてキャンセル.
・設定 (S)-シリアルを選択し，下記の画面.

図 1.7: TeraTerm 設定

・以下で設定.

シリアルポート設定

ポート	：USB-シリアル変換ケーブルで使用しているポート
転送レート	：115,200bps
データ長	：8bit
ストップビット	：1bit
パリティ	：なし
フロー制御	：なし

・リセットボタンを押して起動確認.

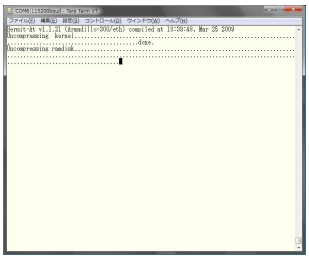

図 1.8: 起動確認

・リセットボタンの位置

図 1.9: リセットボタンの位置

・起動確認できない場合

windows のデバイスマネージャーから USB-シリアル変換ケーブルのプロパティを選択し，詳細設定からポート番号を変更してみる.

1.2.2 minicom で起動確認 (Linux)

minicom の設定

・端末に以下のコマンドを入力して [設定] 画面を表示

```
$ minicom -s
```

・「シリアルポート」を選択

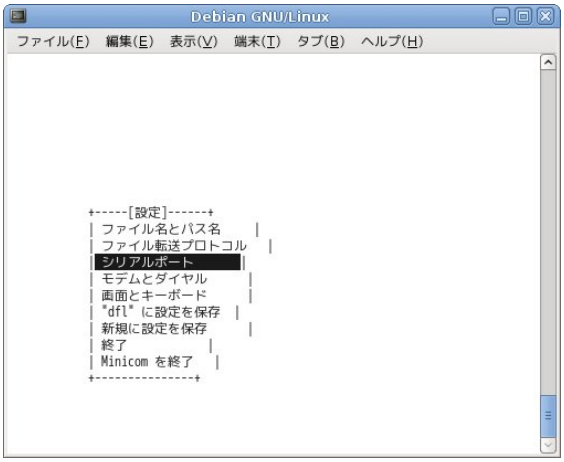

図 1.10: minicom 設定画面

・以下で設定 (表示がおかしい場合は 1 項目をすべて消してから書き直す)

シリアルポート設定

A - シリアルデバイス : /dev/ttyUSB0（ここは USB-シリアル変換ケーブルで使用しているポート）
B - ロックファイルの位置 : /var/lock
C - Callin Program :
D - Callout Program :
E - 速度/パリティ/ビット : 115200 8N1
F - ハードウェア流れ制御 : いいえ
G - ソフトウェア流れ制御 : いいえ

・[設定] 画面に戻ったら、「”dfl”に設定を保存」を選択
・「minicom を終了」

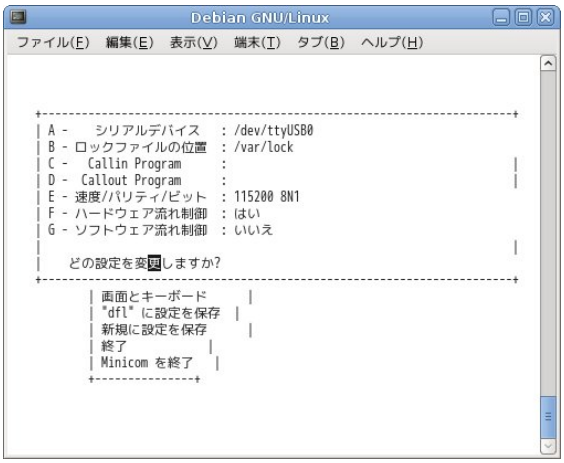

図 1.11: シリアルポート選択

・以下のコマンドを入力して minicom を起動

```
$ minicom
```

・リセットボタンを押して起動確認

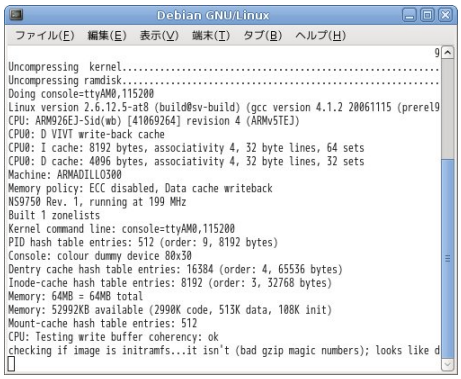

図 1.12: 起動確認

・起動確認できない場合

・以下のメッセージが出る場合

minicom: /dev/ttyUSB? をオープンできません : 許可がありません

・対処１：su コマンドでスーパーユーザーになって minicom を起動する.

・対処２：スーパーユーザーになって chmod コマンドで USB ポート使用の許可を与える.

1.2.3 ログイン

・login プロンプトが出たらログイン

・ユーザ：root

・パスワード：root

1.3 システム設定

1.3.1 設定準備・vi の基礎知識

vi とは

・vi とは unix 付属のテキストエディタである．他のエディタ (emacs，メモ帳，秀丸など) と異なるのは，カーソルの移動やコピー＆ペーストを行うコマンドモードと文字入力を行う入力モードに分かれていることである．つまり，いつでも文字を入力できるのではない．

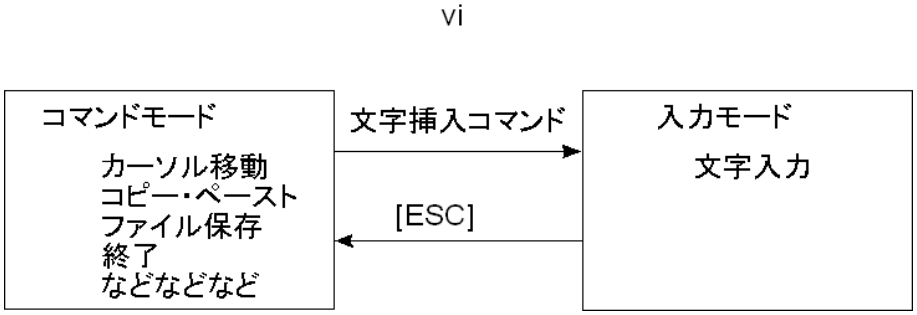

図 1.13: vi におけるモード選択

起動

・ vi ファイル名
  ・例：vi test.txt

文字入力

・コマンドモードの時，文字挿入コマンドを押して入力モードにする．
  ・コマンドモードか入力モードか分からない時には，[ESC] を押してコマンドモードにしておく
  ・文字挿入コマンド
    ・i，a，o，I，A，O

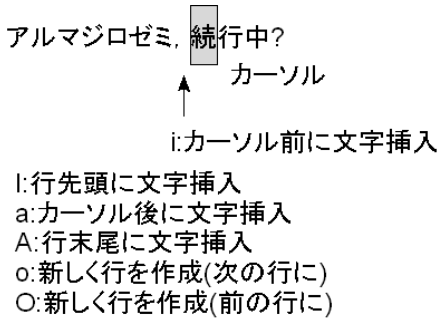

図 1.14: 文字挿入

カーソル移動

・コマンドモード時にカーソルを移動するコマンド

- h：左
- l：右
- k：上
- j：下

文字・コピー＆ペースト

・コマンドモード時に文字を消す (入力モードでバックスペースでも大丈夫)
  ・x：1 文字削除
    ・3x など前に数字をつけると 3 文字 (その数字分) 削除
  ・dd：行削除．2dd などで 2 行削除などできる．
  ・yy：行コピー．2yy などで 2 行コピーなどできる．
  ・p：貼り付け．
  ・u：操作のやりなおし．

保存・終了

・コマンドモード時にファイルへの保存，vi の終了
  ・: q
    ・終了
  ・: w
    ・保存

1.3.2　ネットワーク設定

有線 LAN の設定 (有線 LAN：eth0)

コンピュータ名の設定

・vi /etc/config/HOSTNAME
  ・コンピュータの名前を入れる

/etc/config/HOSTNAME

```
armadillo300.csse.muroran-it.ac.jp
```

IP 設定 (DHCP)

・vi /etc/config/interfaces

/etc/config/interfaces

```
auto lo eth0
iface lo inet loopback
iface eth0 inet dhcp
```

IP 設定 (固定 IP)

・vi /etc/config/interfaces

/etc/config/interfaces

```
auto lo eth0
iface lo inet loopback
iface eth0 inet static
   adress ***.***.***.***
   netmask ***.***.***.***
   network ***.***.***.***
   broadcast ***.***.***.***
   gateway ***.***.***.***
```

DNS 設定

・vi /etc/config/resolv.conf

/etc/config/resolv.conf

```
nameserver ***.***.***.***
```

プロキシ設定

・学内で使用する場合は必須

- vi /root/.profile

/root/.profile

```
export http_proxy="http://proxy.muroran-it.ac.jp:8080"
```

1.3.3 ネットワーク設定確認

・再起動
  ・reboot

・設定確認
  ・ifconfig -a
  ・eth0 などの項目を見て，IP アドレスが割り振られていることを確認

1.3.4 情報書き込み

・flatfsd -s
  ・/etc/config 以下の変更を有効に

・flatfsd -w
  ・初期化する
    ・/etc/default/で初期化

第2章　Linux組み込みボードへのDebianインストール

2.1　準備

2.1.1　物品確認

・CF（コンパクトフラッシュカード）：

図 2.1: CF（コンパクトフラッシュカード）

2.1.2　注意 1

・赤 LED 点滅時には，決して，電源を切らないように.

2.1.3　注意 2

・Debian インストール後は，以下のコマンドで終了すること.
  ・shutdown -h now

2.2　ファイルシステムの初期化

・CF 上に ext2 ファイルシステムを作成して Debian をコピーできる状態にします.
・Debian を導入するだけで 350MB ぐらいの容量が必要となります.

2.2.1 手順

・CF を Linux 組み込みボード に挿入

図 2.2: Armadillo300 への取り付け

・Linux ボード電源投入
・root でログイン
・パーティション作成 (◎の行でキー入力が必要)

```
◎ [armadillo ~]# fdisk /dev/hda
    ◎ Command (m for help): d
    Selected partition 1
    ◎ Command (m for help): n
    Command action
    e extended
    p primary partition (1-4)
    ◎ p
    ◎ Partition number (1-4): 1
    ◎ First cylinder (1-1946, default 1): 1
    ◎ Last cylinder or +size or +sizeM or +sizeK (1-1946, default 1946): 1946
    ◎ Command (m for help): w
    The partition table has been altered!
    Calling ioctl() to re-read partition table.
    Syncing disks.
```

・ファイルシステムの作成

```
◎ [armadillo ~]# mke2fs -O -filetype /dev/hda1
```

2.3 Debain のインストール

・Debian の分割されたアーカイブファイルを Armadillo に ftp 転送して CF にコピーします.

2.3.1 CF と転送ファイル用の ramfs をマウント

```
[armadillo ~]# mount /dev/hda1 /mnt
[armadillo ~]# mount -t ramfs none /home/ftp/pub
[armadillo ~]# chmod 777 /home/ftp/pub
```

2.3.2 ftp でアーカイブファイルを転送

```
[armadillo ~]# cd /mnt
[armadillo ~]# export http_proxy="http://proxy.muroran-it.ac.jp:8080"
[armadillo ~]# wget http://download.atmark-techno.com/armadillo-300
                                                /images/linux-a300-1.07.bin.gz
[armadillo ~]# wget http://download.atmark-techno.com/armadillo-300
                                                /debian/debian-etch-a300-1.tgz
[armadillo ~]# wget http://download.atmark-techno.com/armadillo-300
                                                /debian/debian-etch-a300-2.tgz
[armadillo ~]# wget http://download.atmark-techno.com/armadillo-300
                                                /debian/debian-etch-a300-3.tgz
[armadillo ~]# wget http://download.atmark-techno.com/armadillo-300
                                                /debian/debian-etch-a300-4.tgz
[armadillo ~]# wget http://download.atmark-techno.com/armadillo-300
                                                /debian/debian-etch-a300-5.tgz
```

2.3.3 CF にアーカイブファイルを展開

```
[armadillo ~]# tar zxvf /mnt/debian-etch-a300-1.tgz -C /mnt
  debian-etch-a300-1 から debian-etch-a300-5 まで繰り返し行う
[armadillo ~]# rm -f /mnt/debian-etch-a300-?.tgz
[armadillo ~]# mv /mnt/linux-a300-1.07.bin.gz /mnt/boot/Image.gz
[armadillo ~]# sync
```

2.4 起動設定

・CF にインストールした Debian を起動するように設定

2.4.1 Debian 起動設定

・Linux ボードの電源を切る
・ジャンパピンの設定
  ・JP1:2-3
  ・JP2:1-2

図 2.3: Armadillo300 上のジャンパピンとジャンパピン番号

・Linux ボードの電源投入する.
・起動確認.

2.4.2 旧情報

・現在，”Hermit 起動モード設定”を行わなくても起動するので，必要がなければ”Hermit 起動モード設定”を行わなくてよい.

Hermit 起動モード設定

・Linux ボードの電源を切る
・CF を抜く.
・ジャンパピンの設定 (工場出荷の初期状態でも可能).
  ・JP1:2-3
  ・JP2:1-2

図 2.4: Armadillo300 上のジャンパピンとジャンパピン番号

・Linux ボードの電源投入する.
・起動パラメータでルートファイルシステムを CF に指定する.

  ・`hermit> clearenv`
  ・`hermit> setenv console-ttyAM0,115200 root=/dev/hda1 noinitrd`

・Linux ボードの電源を切る．
・CF を挿入する．
・Linux ボードの電源投入する．
・起動確認．

2.5 初期設定

2.5.1 root でのログイン確認

・root でのログインを行う．初期設定では以下のようになっている．
  ・ユーザ名:root
  ・パスワード:なし

2.5.2 root のパスワード設定

・root のパスワードを設定する．設定しないと，ssh 等でログインができない．
  ・パスワード設定コマンド
    ・passwd
  ・設定パスワード
    ・ユーザ名:root
    ・パスワード:root

2.6 ネットワーク設定

・注意事項
  ・Armadillo300 には有線 LAN，無線 LAN が存在する．
  ・同一セグメントで動作させる場合，以下の状態にするとネットワークの動作が保障されない．
    ・片方を固定 IP で，もう片方を DHCP で IP を設定する．
    ・両方固定 IP の場合に，両方に gateway を設定する．
  ・デバイス名
    ・有線 LAN:eth0
    ・無線 LAN:ath0

2.6.1 必須ソフトウェアのインストール

・無線 LAN 設定のためのソフトウェア
  ・wlanconfig
  ・iwconfig
・これらは初期状態ではインストールされていない．
・また，ネットワーク経由で Armadillo300 にインストールするためのソフトウェア
  ・ssh
  も初期状態ではインストールされていない．
・そこで，Debian で使用可能なソフトウェアパッケージ (ソフトウェアをダウンロード・インストール等行う管理システム) である apt を用いて無線 LAN の設定ツールをインストールする．まず，apt の設定を行い，次に apt を用いたソフトウェアのインストールを行う．

apt の設定

・apt で用いるソフトウェアパッケージリストの更新

・vi /etc/apt/sources.list
・以下を追加
・deb http://ftp.riken.jp/Linux/debian/debian-archive etch main contrib non-free
・deb http://ftp.riken.jp/Linux/debian/debian-non-US etch/non-US main contrib non-free
・編集終了 ([ESC]-:q)

・リストのアップデート
・apt-get update

無線 LAN 設定ツールのインストール

・apt-get install wireless-tools madwifi-tools
・途中で容量確認等のプロンプトが出るので，全部 y を入力する．

・注意：無線 LAN の MAC アドレスは全部設定し終わるまで取得できない．そこで MAC アドレスが必要な場合は，まず以下の手順で適当に設定し，MAC アドレスを取得する．

・wlanconfig ath0 destroy
・wlanconfig ath0 create wlandev wifi0 wlanmode sta
・ifconfig ath0
・これで MAC アドレス取得
・HWaddr の項目が MAC アドレス

ssh のインストール

・apt-get install ssh

ssh の設定

・vi /etc/ssh/sshd_config
・PermitRootLogin yes
・root でログインするのを許可する．
・本来，一般ユーザを登録するので，基本的には PermitRootLogin no で．
yes にする場合は，セキュリティ的に問題があることを認識し，熟考すること．

2.6.2 有線・無線 LAN 共通設定

コンピュータ名の設定

・vi /etc/hostname
・コンピュータの名前を入れる
・armadillo300.csse.muroran-it.ac.jp

プロキシ設定

・学内で使用する場合は必須

・vi /root/.bashrc
・export http_proxy="http://proxy.muroran-it.ac.jp:8080"

2.6.3 固定 IP の設定

・ここでは，有線 (eth0)・無線 LAN(ath0) に固定 IP を付与する．
どちらかのみを設定したい場合は，該当部分を読み飛ばすこと．

/etc/network/interfaces の設定

・vi /etc/network/interfaces (/etc/network/interface を編集する．)
（ ）の行は書かない．

/etc/network/interfaces

```
auto lo eth0 ath0
iface lo inet loopback
iface eth0 inet static
  address ***.***.***.***
          (***.***.***.*** はシステム管理者によって与えられた IP)
  netmask ***.***.***.***
          (***.***.***.*** はシステム管理者によって与えられたネットマスク)
  network ***.***.***.***
          (設定可能項目として記述．実際には記述しなくてよい)
  broadcast ***.***.***.***
          (設定可能項目として記述．実際には記述しなくてよい)
  gateway ***.***.***.***
          (***.***.***.*** はシステム管理者によって与えられたゲートウェイ)
iface ath0 inet static
  address ***.***.***.***
          (***.***.***.*** はシステム管理者によって与えられた IP)
  netmask ***.***.***.***
          (***.***.***.*** はシステム管理者によって与えられたネットマスク)
    pre-up wlanconfig ath0 destroy
    pre-up wlanconfig ath0 create wlandev wifi0 wlanmode sta
    wireless-essid ***
          (*** はシステム管理者によって与えられた無線 LAN の ESSID)
```

DNS 設定 (有線・無線 LAN 共通)

・vi /etc/resolv.conf

/etc/resolv.conf

```
    ・nameserver ***.***.***.***
```

2.6.4 DHCP による IP 設定

・ここでは DHCP サーバにより有線・無線 LAN に IP を自動割付することを考える．有線・無線のどちらかのみ設定したい場合は，該当部分を読み飛ばすこと．

・DHCP サーバが適切に設定されていれば，DNS の設定も自動的に行ってくれるため，ここでは割愛する．必要があれば，固定 IP の項目の DNS 設定を参照のこと．

/etc/network/interfaces の設定

・vi /etc/network/interfaces

/etc/network/interfaces

```
auto lo eth0 ath0
iface lo inet loopback
iface eth0 inet dhcp
iface ath0 inet dhcp
```

2.6.5 ネットワーク設定確認

・ネットワークの再起動
・/etc/init.d/networking restart
・ログアウト・ログイン (プロキシを更新)
・設定確認
・ifconfig -a
・eth0 などの項目を見て，IP アドレスが割り振られていることを確認.
・設定確認 2
・TeraTerm 等で割り当てた IP に SSH でログインできることを確認.

2.6.6 その他情報

無線 LAN のコマンドによる手動設定

・wlanconfig ath0 destroy
・wlanconfig ath0 create wlandev wifi0 wlanmode ***
・***:通信モード．以下のモードが指定可能
・sta: 通信モード Managed，アクセスポイントを介しての通信方式.
・ap: 通信モード Master，Linux 組み込みボード自体がアクセスポイントとして機能.
・adhoc: 通信モード Ad-Hoc，アクセスポイントを介さず他の機器と 1 対 1 で通信を行う方式.
・iwconfig ath0 essid ***
・***:ESS-ID
・IP の設定
・IP を手動で設定する場合
・ifconfig ath0 ***.***.***.*** up
・(***.***.***.***: IP アドレス)
・IP を DHCP で設定する場合
・dhclient ath0

OS 起動時における無線 LAN の自動起動

・ただし，通常は/etc/network/interfaces の設定で十分なので以下は不要.

・vi /etc/rc0.d/S35networking
・wlanconfig ath0 destroy
・... # "コマンドによる手動設定"の手順を全部書く.
・exit 0 #ファイルの最後に exit 0 を書くことを忘れないように.

2.7 ユーザ登録

2.7.1 一般情報

・ユーザ・グループの追加
  ・新規に作成する場合，グループの追加・ユーザの追加の順に行う.
    (ユーザの追加時にどのグループに属するかを指定しなければならないため．)
  ・以下の条件で例を示す.

> ユーザ名:ken
> ユーザ番号:10000
> グループ名:kura
> グループ番号:10001
> 名前とか:ken kura
> ホームディレクトリ:/home/ken-kura
> シェル:/bin/sh
> 個人用設定ファイルのありか:/etc/skel/
> 初期パスワード:hoge

・グループの追加 (コマンド)
  ・groupadd
    ・man groupadd で使い方をみるべし
    ・例 1:groupadd kura
    ・例 2:groupadd -g 10001 kura
・グループの追加 (手動)
  ・/etc/group を編集
    ・例:`echo "kura:*:10001:" >> /etc/group`
・ユーザの追加 (コマンド)
  ・useradd
    ・man useradd で使い方をみるべし
    ・例 1:useradd -m -c ”ken kura” -d /home/ken-kura -g kura -k /etc/skel -p hoge -s /bin/sh -u 10000 ken
・ユーザの追加 (手動)
  ・vipw
    ・ユーザの追加・編集
    ・例:ken::10000:10001::0:0:ken kura:/home/ken-kura:/bin/sh
  ・mkdir /home/ken
  ・chmod 775 /home/ken-kura
  ・chown ken:kura /home/ken-kura
  ・cp /etc/skel/* /home/ken-kura/

2.7.2 具体事例

・groupadd -g 1000 kentarou
・useradd -m -c ”Kentarou Kurashige” -d /home/kentarou -g kentarou -k /etc/skel -p hoge -s /bin/bash -u 1000 kentarou
・vi /etc/group
  ・root:x:0:kentarou
    ・登録したユーザ kentarou が su などで root になれるようにしておく.

2.8　終了

・shutdown -h now
　・”Power down”のメッセージがでたら電源を切っても大丈夫.
・もしくは
　・halt

第3章　Linux組み込みボードでのプログラミング

3.1　概要

・Linux 組み込みボード上でシリアル通信を使ったプログラムを行う．これまで Linux 組み込みボード上のシリアルポート (CON7) を介して PC と繋げ，ログイン・各種設定を行ってきた．このように Linux 組み込みボード上のシリアルポート (CON7) はシステムが使用する．そこで Linux 組み込みボード上のシリアルポート (CON6) を使用して，Linux 組み込みボード上のプログラムの出力を PC に表示するプログラムの作成を行う．その為，システムへは ssh(TCP/IP ネットワーク) によってログイン等を行い，PC と繋げていたシリアルケーブルは Linux 組み込みボード上のシリアルポート (CON6) に繋げなおして使用する．

・前回まで

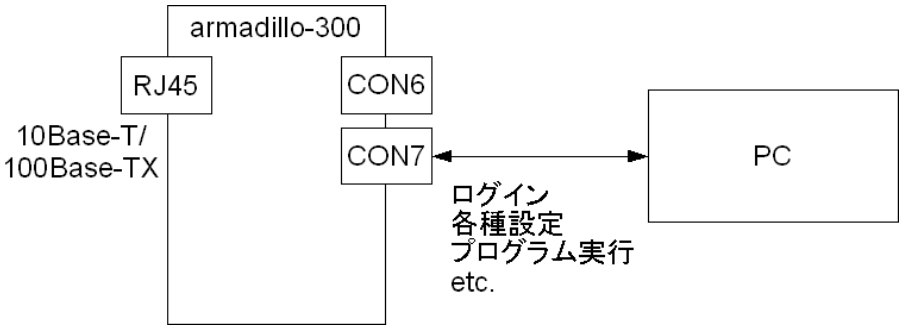

図 3.1: シリアルポート経由でのログイン

・今回から

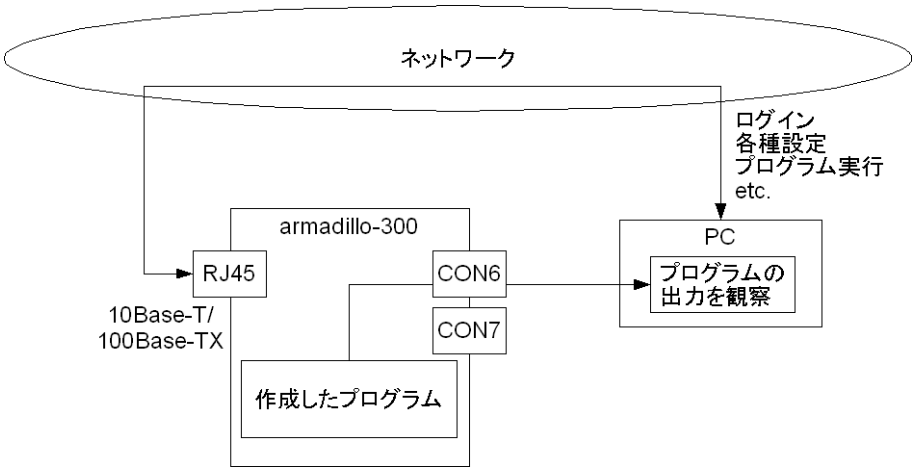

図 3.2: ネットワーク経由でのログイン

3.2 準備

3.2.1 開発環境のインストール

・apt-get を用いて Linux 組み込みボード上でプログラムをコンパイルする環境を整える．

・apt-get install binutils-dev gcc g++ libpopt-dev make patch

3.2.2 SSH 経由でログインできることを確認

・TeraTerm,minicom などで，SSH 経由でログインできることを確認する．

・SSH 経由でのログインの場合，ホスト名/アドレスが必要になる．分からない場合，シリアル経由で Armadillo 300 にログインし，以下のコマンドで IP アドレスを確認しておく．

・有線 LAN 使用の場合：ifconfig eth0

・無線 LAN 使用の場合：ifconfig ath0

3.2.3 配線の変更

・con7 のケーブルを con6 へ挿す

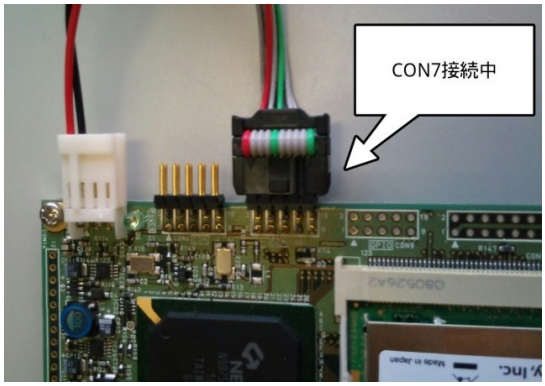

図 3.3: con7 に挿さっている状態

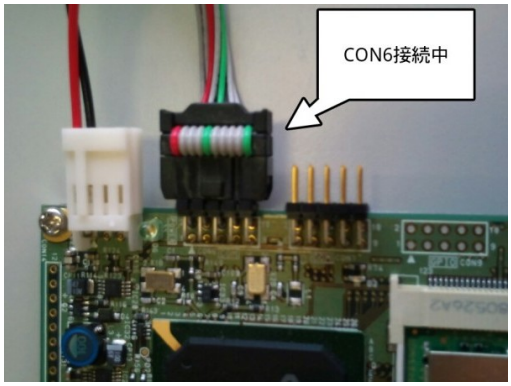

図 3.4: con6 に挿さっている状態

3.3 プログラミング

3.3.1 Hello World!を出力

・下記のプログラムを入力

filesample.c

```
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <termios.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* シリアルインターフェースに対応するデバイスファイル */
#define SERIAL_PORT     "/dev/ttyAM1"

int main(){

  /* シリアルインターフェースのオープン */
  /* O_RDWR: 入出力用にオープン */
```

```
18    /* O_NOCTTY: ノイズ等による不意のctrl-cを防ぐため，tty制御なし */
19    /* シリアルインターフェースをint型変数"fd"の名前で扱えるように */
20    int fd;
21    fd = open(SERIAL_PORT, O_RDWR | O_NOCTTY);
22    if(fd < 0){
23      printf("%s doesn't open it\n",SERIAL_PORT);
24      return -1;
25    }
26
27    /* シリアルポートの設定を行う変数を宣言 */
28    struct termios oldtio, newtio;
29    /* 現在の設定を oldtio に保存 */
30    tcgetattr(fd, &oldtio);
31    /* 今回使用する設定 newtio に現在の設定 oldtio をコピー */
32    newtio = oldtio;
33
34    /* 入出力スピードの設定 */
35    /* 以下の方法1または方法2で設定．どちらかでよい */
36
37    /* 方法1 */
38    cfsetispeed(&newtio, B115200); /* 入力スピード設定 */
39    cfsetospeed(&newtio, B115200); /* 出力スピード設定 */
40
41    /* 方法2 */
42    cfsetspeed(&newtio, B115200); /* 入出力スピード設定 */
43
44    /* 27行目-42行目までの設定を有効にする */
45    tcflush(fd, TCIFLUSH);
46    tcsetattr(fd, TCSANOW, &newtio); /* 設定を有効に */
47
48    /* 通信処理:シリアルデバイスに出力 */
49    char buf[255];
50    strcpy(buf,"hello");
51    write(fd, buf , sizeof(buf)); /* write関数:送信処理 */
52
53    /* デバイスの設定を戻す */
54    tcsetattr(fd, TCSANOW, &oldtio);
55    close(fd);
56    return 0;
57  }
```

・gcc -o sample1 sample.c
  ・プログラム sample.c をコンパイル
  ・コンパイル後の実行ファイルの名前を sample1 として保存
・./sample1
  ・プログラムを実行
  ・Hello World!と表示されることを確認

3.3.2 プログラムの説明

- 通信のための準備・終了処理
  - 16 行目から 46 行目までがシリアル通信を行うための準備の部分である．基本的にプログラムによってあまり変更がないので，丸覚えでもかまわない．詳しくは，3.4.1「参考のシリアル通信に関する重要な部分の説明」を参照のこと．
  - 53 行目から 56 行目までがシリアル通信を終了させるための後始末の部分である．基本的にプログラムによって変更はないので，丸覚えでもかまわない．
- メインプログラム
  - 48 行目から 51 行目までがシリアル通信を使ったプログラムの部分である．ここで，自分の考えた処理を行わせる．特に 51 行目が重要であり，write 関数によって接続機器に情報を送信している．
    ・read 関数:受信 (接続機器 から Armadillo-300 に情報伝達)
    ・write 関数:送信 (Armadillo-300 から 接続機器に情報伝達)

3.4 参考

3.4.1 シリアル通信に関する重要な部分の説明

- 12 行目
  `#define SERIAL_PORT "/dev/ttyAM1"`
  — 今回使用するシリアルインターフェース

- 21 行目
  `fd = open(SERIAL_PORT, O_RDWR | O_NOCTTY);`
  — シリアルインターフェースを”fd”の名前で使えるようにする．

- 28-32 行目
  `struct termios oldtio, newtio`

  – 非同期通信ポートを制御するための汎用ターミナルインタフェース (ここではシリアルポート) の設定を司る変数．詳しくは後ほど説明．

  `tcgetattr(fd, &oldtio);`

  – シリアルインターフェース”fd”から，現在の設定を読み込み oldtio に保存．

  `newtio = oldtio;`

  – 現在の設定を newtio にコピー．以後，newtio の設定を変え，tcsetattr(fd,&newtio) などでシリアルインターフェース”fd”の設定変更を行う．最後に tcsetattr(fd, TCSANOW, &oldtio) にてシリアルインターフェース”fd”を初期の設定に戻す．大まかな流れは後ほど説明．

- 38-39 行目
  `cfsetispeed(&newtio, B115200);`
  — newtio の中の入力に関する通信速度の設定
  `cfsetospeed(&newtio, B115200);`
  — newtio の中の出力に関する通信速度の設定

- 42 行目
  `cfsetspeed(&newtio, B115200);`
  — 36-38 行目と同じことをしている．こちらは入出力を一度に設定している．

- 45-46 行目
  `tcflush(fd, TCIFLUSH);`
  — シリアルインターフェースの送受信データをクリア
  ・tcsetattr の実行のために必要．
  `tcsetattr(fd, TCSANOW, \&newtio);`
  — newtio を用いてシリアルインターフェース”fd”を設定．

- 51 行目
  `write(fd, buf , sizeof(buf));`
  — シリアルインターフェース”fd”に size(buf) 分のデータ buf を送信

- 54-55 行目
  `tcsetattr(fd, TCSANOW, \&oldtio);`
  — シリアルインターフェース”fd”を初期の設定 (oldtio) に戻す
  `close(fd);`
  — シリアルインターフェースと名前”fd”の繋がりを開放．

3.4.2 struct termios の説明

・struct temios は構造体であり，以下のようになっている．

struct temios

```
struct termios {
     tcflag_t c_iflag; /* 入力フラグ */
     tcflag_t c_oflag; /* 出力フラグ */
     tcflag_t c_cflag; /* 制御フラグ */
     tcflag_t c_lflag; /* ローカルフラグ */
     cc_t c_cc[NCCS]; /* 特殊制御文字の設定 */
};
```

入力フラグ

・端末 (この場合 armadillo) への入力に関する設定である．特に文字入力の仕方 (入力の 8 ビット目を落とす，パリティ検査を行うなど) を制御する．

出力フラグ

・端末 (この場合 armadillo) からの出力に関する設定である．特に出力処理の実行の仕方 (改行を CR/LF に変更するなど) を制御する．

制御フラグ

・端末 (この場合 armadillo) のハードウェアに関する設定である．RS-232 のシリアルライン (モデムの状態信号の無視，文字ごとのストップビット数など) に影響する．

ローカルフラグ

・端末 (この場合 armadillo) のその他の設定である．ドライバとユーザのインターフェース (エコーのオン/オフ，削除した文字の表示方法，端末生成シグナルの有無，バックグラウンドからの出力を止めるジョブ制御シグナルなど) に影響する．

・`tcflag_t c_lflag;`

3.4.3 struct termios の使い方

・tcgetattr にてプログラム開始時の初期設定を読み込む
・new = old などとして初期設定と同じ新しい設定変数を作成し，以後新しい設定変数を使う
・sfsetspeed など必要に応じて必要な箇所のみの変更を行う
・通信処理
・プログラム終了時に初期設定に戻す

3.4.4 シリアルデバイスにおける入力処理の概念

入力方式

シリアル通信では，シリアルデバイスを介して外部からデータが入力される．プログラム上では，read 関数を用いて外部から入力されたデータを読み取る．この読み取りに関して，シリアルデバイスでは大きく分けて 2 種類の入力処理がある．

- カノニカル入力処理 (デフォルト)
  - 外部からの全ての入力は，行単位で処理される．つまり，read によって得られるデータは 1 行全体である．よってデータは以下のいずれかで終わる．
    * NL(ASCII の LF)
    * ファイル終端
    * 行終端文字
    * 注意点としては，標準の設定では CR(DOS/Windows のデフォルトの行終端文字) は行終端とはならない．
- 非カノニカル入力処理
  - read 関数を呼び出す際に読み込むデータの大きさを指定する方法である．特に，プログラムが決まった文字数のキャラクタを読み込む時や，接続したデバイスが大量の文字を送ってくる場合に使用する．

処理方式

- カノニカル入力処理/非カノニカル入力処理においては外部から入力されるデータは read 関数にて読み込む．このとき，read 関数の処理の仕方が大別すると 2 種類ある．
  - 同期処理
    * read 関数にてデータ読み込みが始まると，データを読み終えるまでプログラムの処理は中断される．つまりデータが読み込まれるまで待つ．
  - 非同期処理
    * read 関数にてデータ読み込みが始まると，プログラム自体は read 関数の次の行から処理が再開される．つまり，データが読み終えるまで待たずに，次の処理が続行される．データ読み込みが終わるとシステムがプログラムにシグナル (信号・合図) を送るしくみとなる．

Armadillo での処理

・シリアルに接続されたセンサなどを想定しているので非カノニカル入力処理・非同期処理が好ましい．

第4章　Linux組み込みボードでのサーボの取り扱い

4.1　概要

Linux 組み込みボードのシリアルデバイス経由でプログラムによりサーボを動作させる．ここで，Linux 組み込みボードのシリアルデバイスに接続する機器として以下のものがある．

- Linux 組み込みボードのシリアルデバイス
  - RS-232C レベルの信号を扱うシリアルデバイスである．具体的には，-15V〜15V の範囲を扱い，論理 0 が+3V〜+15V，論理 1 が-15V〜-3V である．
- AGB65-RSC
  - シリアルポート経由で受けた指令を元にサーボを制御する機器である．この機器が扱うシリアルデバイスは TTL レベル信号であり，論理 0 が 0V，論理 1 が 5V である．
- AGB65-232C
  - RS-232C レベルの信号と TTL レベルの信号には電圧互換がないため，変換処理が必要となる．AGB65-232C は RS-232C レベルの信号と TTL レベルの信号の変換処理を行う機器である．
- サーボモータ
  - 物体の位置などを制御量として目標値に追従するように自動で作動するモータ.

4.2　準備

4.2.1　接続

1. Armadillo-300 と AGB65-232C を接続する．(接続ケーブル:ピンソケット-3 ピン 2mm ピッチコネクタ)
2. AGB65-232C と AGB65-RSC を接続する．(接続ケーブル:付属の 4 ピンケーブル・電源ケーブル)
3. AGB65-RSC の HV ピンをジャンパピンで短絡させる．
4. サーボを接続する．
5. サーボ用電源にバッテリーを接続する．

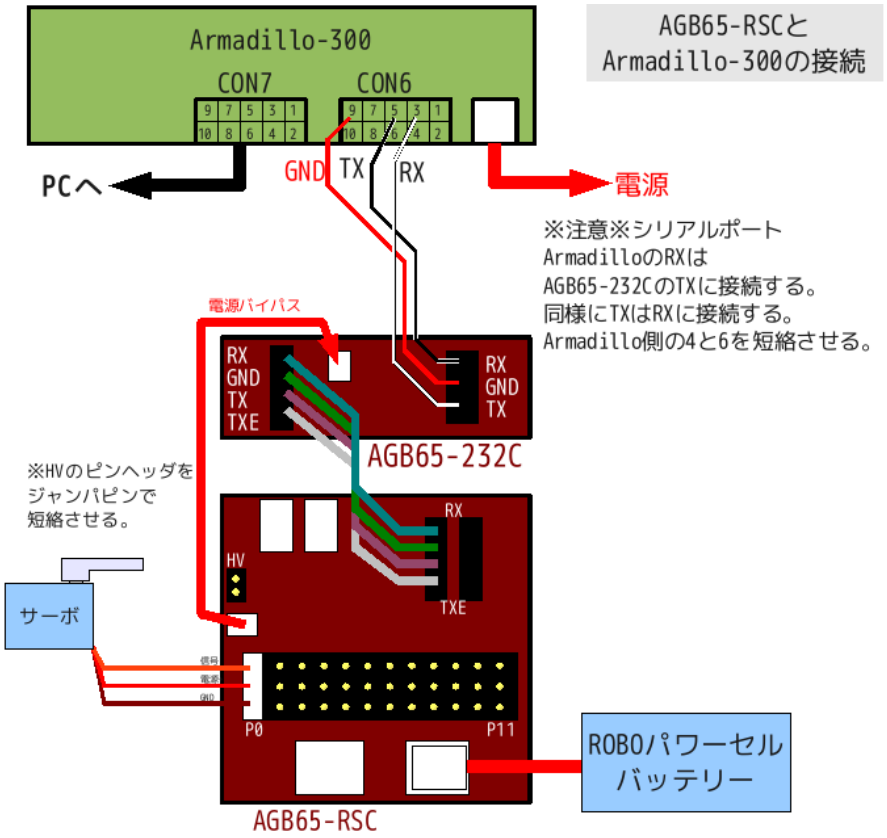

図 4.1: サーボ接続図

・Armadillo-300 と AGB65-RSC のシリアル信号は電圧レベルが違うため，
電圧変換のために AGB65-232C を使用する．
  ・Armadillo-300 は RS-232C レベル
  ・AGB65-RSC は TTL レベル

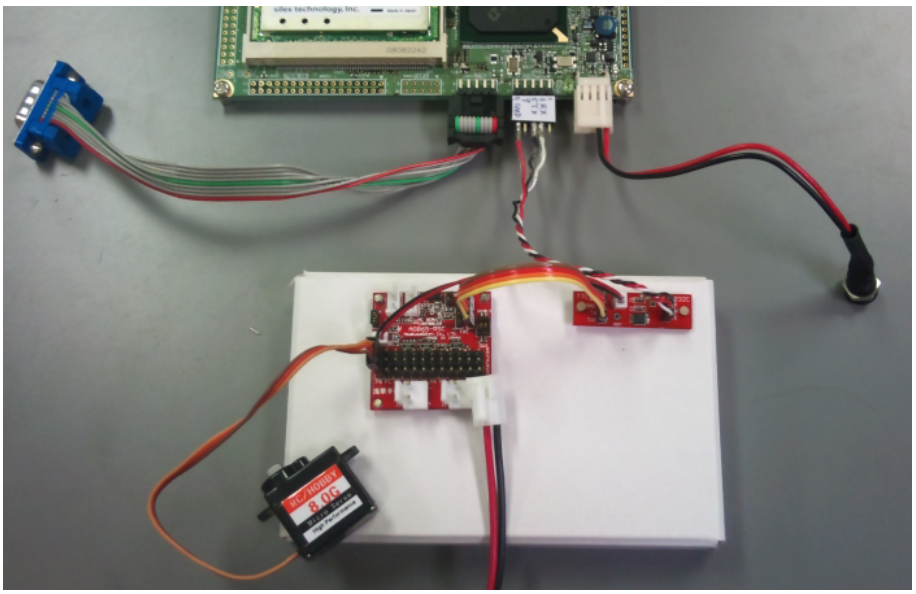

図 4.2: 部品配置・接続

4.3 プログラム1

4.3.1 AGB65-RSC の簡単な使い方

・以下のように配列を定義して値を代入し、シリアル送信する.

unsigned char output[7] = 255,3,4,2,0,127,20;

表 4.1: AGB65-RSC へ送信するデータ形式

シンクロバイト (255)	ＲＳＣにデータの通信開始を知らせるデータで，常に「２５５」で始まる.
ＩＤ (3)	ＲＳＣに設定された固有のＩＤ (AGB65 シリーズを複数接続したときの判別用. ＲＳＣの場合，出荷時のＩＤは「３」.)
送信バイト数 (4)	送信される命令の（バイト）数. シンクロバイト，ＩＤ，送信バイト数は数えない.
命令 1(2)	個別サーボ駆動設定
命令 2(0)	サーボ番号. 0 は RSC 基板上の P0 に対応
命令 3(127)	角度 (min:0，max:255 で，絶対位置を指定)
命令 4(2)	指定した角度まで移動する速度 (x15ms)

4.3.2 AGB65-RSC への命令値とサーボ角度の関係について

・命令値は 0 から 255 まで.

・命令値に対するサーボの可動角は 0 °から 180 °まで. (角度はサーボによる.)

4.3.3 対話的サーボ制御プログラムの作成

・下記のプログラムを入力

fileservo_control.c

・0 から 255 までの値を入力すると、その値に応じてサーボモータの角度が変わる.

・0-255 以外の値を入力するとプログラムは終了する.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <termios.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define SERIAL_PORT       "/dev/ttyAM1"

int main(){
  int fd;
  fd = open(SERIAL_PORT, O_RDWR | O_NOCTTY);s
  if(fd < 0){
    printf("%s doesn't open it\n",SERIAL_PORT);
    return -1;
  }

  struct termios oldtio, newtio;
  tcgetattr(fd, &oldtio);
  newtio = oldtio;

  /* 通信方式の設定・非カノニカル入力処理選択 */
  newtio.c_iflag = IGNPAR;
  newtio.c_lflag = 0;

```

```
29    /* read関数での文字待ちうけの設定 */
30    newtio.c_cc[VTIME] = 0; /* 値x0.1秒待つ */
31    newtio.c_cc[VMIN] = 1; /* 値の文字分だけ入力されるまで待つ*/
32
33    /* 通信速度の設定 */
34    cfsetspeed(&newtio, B9600);
35
36    /* シリアルデバイス初期化処理 */
37    tcflush(fd, TCIFLUSH);
38    tcsetattr(fd, TCSANOW, &newtio);
39
40    /* サーボ指令の雛形 */
41    /* 個別サーボ駆動モード, サーボコントローラID:3, サーボ番号:0 */
42    unsigned char output[7] = {255,3,4,2,0,127,20};
43
44    /* input(入力):サーボへの角度指令:有効範囲0(0°)-255(180°) */
45    /* 入力が有効範囲内の間, サーボをコントロール */
46    int input;
47    while(1){
48      printf("input value from 0 to 255[-1:end]:");
49      scanf("%d", &input);
50      if(input < 0 || input > 255){ /* サーボコントロール終了 */
51        break;
52      }
53      /* output[5]がサーボへの角度指令 */
54      output[5] = (unsigned char)input;
55      write(fd, output, sizeof(output));
56    }
57
58    /* シリアルデバイス終了処理 */
59    tcsetattr(fd, TCSANOW, &oldtio);
60    close(fd);
61    return 0;
62  }
```

4.4 プログラム2

4.4.1 お題

・数値を入力するとメーター上の対応する値を指すようにサーボモータを制御するプログラムの作成.
→メーターに合うように値を変換する必要がある.

4.5 参考

4.5.1 AGB65-RSC 命令仕様

・マスタ (Armadillo-300) から RSC への指示は，４～１７バイトの数値データで送る．
・バイトサイズ（８ビット）なので，表される数値は０～２５５の２５６種類．
・数値は１０進数．
・注）データは文字データではなく，数値（バイナリ）データで送る．

基本形

・データの基本形は次の通り．[　] 内は１バイトを表す．() 内は送りえる数値の範囲．

[シンクロバイト (255)][ＩＤ (0-3)][送信バイト数 (1-14)][命令１][命令２][命令３]　...

表 4.2: 各項目の説明

シンクロバイト	ＲＳＣにデータの通信開始を知らせるデータで，常に「２５５」で始まる．
ＩＤ	ＲＳＣに設定された固有のＩＤ (AGB65 シリーズを複数接続したときの判別用．ＲＳＣの場合，出荷時のＩＤは「3」．)
送信バイト数	送信される命令の（バイト）数．シンクロバイト，ＩＤ，送信バイト数は数えない．
命令	ＲＳＣに動作させたい命令．詳細は以下の表．

命令の説明

表 4.3: 命令の説明

命令値	動作	方向	フォーマット
1	全サーボ駆動	送信	[255][ID][長 (14)][命令 (1)][P0][P1]...[P10][P11][Speed]
2	個別サーボ駆動	送信	[255][ID][長 (4)][命令 (2)][サーボ番号][サーボ位置][Speed]
3	全サーボパルス停止	送信	[255][ID][長 (1)][命令 (3)]
4	指定サーボパルス停止	送信	[255][ID][長 (2)][命令 (4)][サーボ番号]
5	180 °モード	送信	[255][ID][長 (1)][命令 (5)]
6	255 解像度モード (default)	送信	[255][ID][長 (1)][命令 (6)]

（＊）[Speed] は指定したサーボ位置までにかかる時間．[Speed] × 15ms.

第5章　Linux組み込みボードでのセンサの取り扱い

5.1　概要

Linux 組み込みボードのシリアルデバイス経由でプログラムにより距離センサから信号を受け取る．「Linux 組み込みボードでのサーボの取り扱い」の項で接続した機器に追加して接続する．

- AGB65-ADC
  - シリアルポート経由で受けた指令を元にセンサからの信号を数値化・整形して返す機器である．この機器が扱うシリアルデバイスは TTL レベル信号であり，論理 0 が 0V，論理 1 が 5V である．
- 距離センサ
  - 物体との距離を電圧で表すデバイスである．測定できる距離はデバイスによって異なる．

5.2　準備

5.2.1　接続

1. AGB65-RSC に AGB65-ADC を接続する．(接続ケーブル:付属の 4 ピンケーブル・電源ケーブル)
2. AGB65-ADC にセンサを接続する．

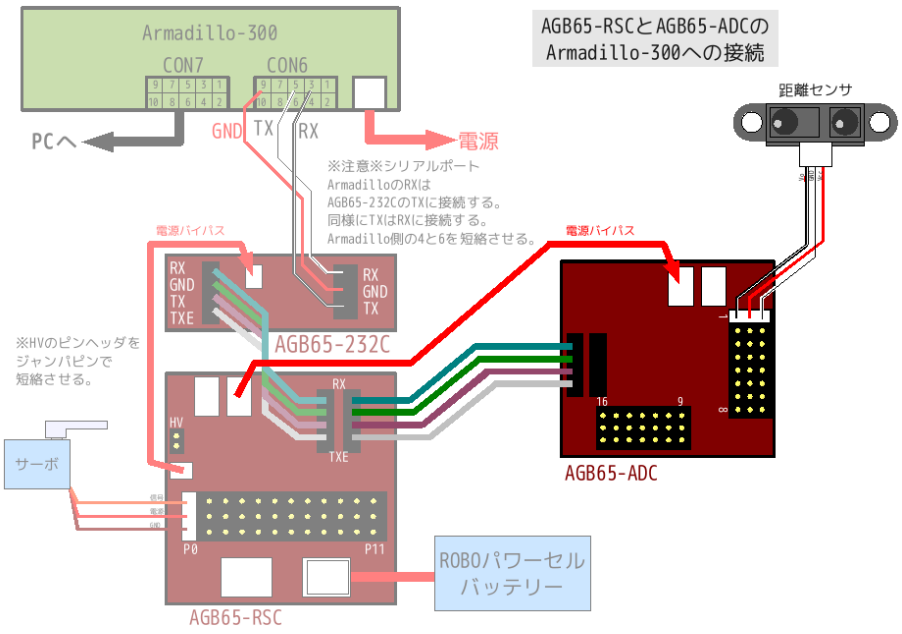

図 5.1: センサ接続図

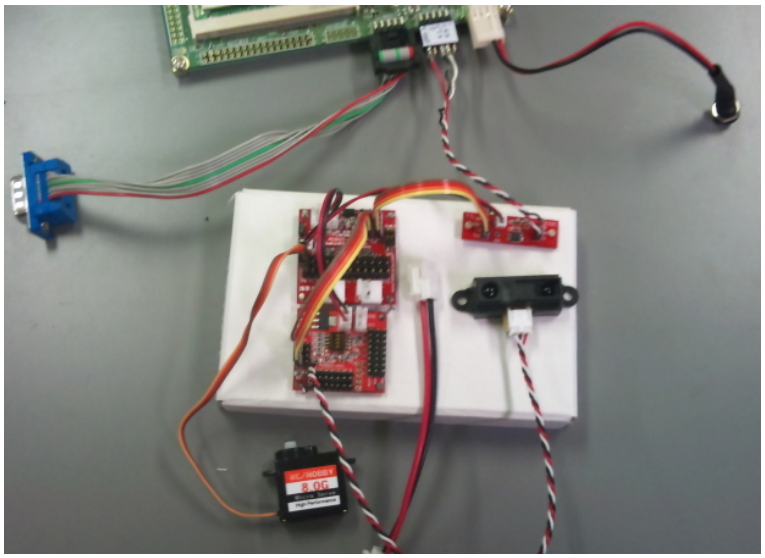

図 5.2: 部品配置・接続

5.3　プログラム 1

5.3.1　AGB65-ADC の簡単な使い方

- AGB65-ADC へシリアル信号で個別の命令を送ることで，その命令に合った形式で結果が返ってくる．
- AGB65-RSC と基本的な使い方は同じであるが，返ってくる値を読み取る必要がある．

5.3.2　AGB65-ADC からのセンサ値と実際の距離の関係について

・電源投入後、約４０ｍＳ後から信号線に距離に応じて出力される．
・この出力は２８～４８ｍＳ毎にアップデートされる．
・出力電圧と距離の関係は次のようになる．

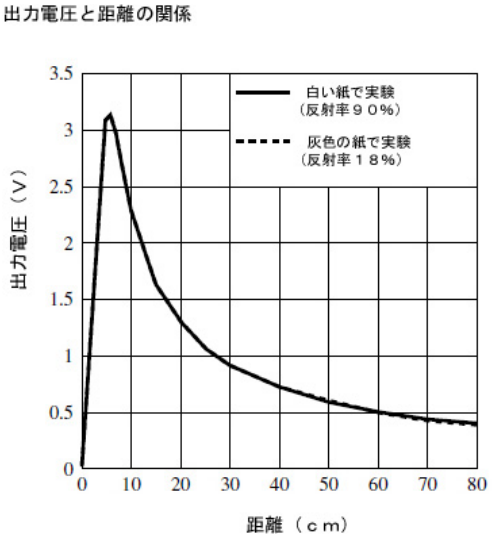

図 5.3: 出力電圧と距離の関係図

5.3.3 センサ値読み取りプログラムの作成

・下記のプログラムを入力

filesensing.c

・センサの値を読み取り，画面に出力.

・センサの値は，デバイスに特化した値.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <termios.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/time.h>

#define SERIAL_PORT       "/dev/ttyAM1"

int main(){
  int fd;
  fd = open(SERIAL_PORT, O_RDWR | O_NOCTTY);
  if(fd < 0){
    printf("%s doesn't open it\n",SERIAL_PORT);
    return -1;
  }

  struct termios oldtio, newtio;
  tcgetattr(fd, &oldtio);
  newtio = oldtio;

  newtio.c_iflag = IGNPAR;
  /* 非カノニカル入力処理 */
  newtio.c_lflag = 0;

  /* read関数での文字待ちうけの設定 */
  newtio.c_cc[VTIME] = 0; /* 値x0.1秒待つ */
  newtio.c_cc[VMIN] = 1; /* 値の文字分だけ入力されるまで待つ*/

  /* バッファをクリア */
  tcflush(fd, TCIFLUSH);

  cfsetspeed(&newtio, B9600);
  tcsetattr(fd, TCSANOW, &newtio);

  /* ボードの設定 */
  int id=120;

  int i, j, k;
  unsigned char command[4] = {255, id, 1, 1};
  unsigned char data[20];

  for(i=1 ; i<=10 ; i++){
    printf("Getting data : %d\n", i);

    /* センサボードにデータ要請 */
    write(fd, command, sizeof(command));
    sleep(1); /* 1s待つ */

    /* 1文字目データ受信 */
    read(fd, &data[0], 1);
    if(data[0] == 255){
      read(fd, &data[1], 1);
      if(data[1] == id){
        read(fd, &data[2], 1);
        if(data[2] == 17){
          read(fd, &data[3], 1);
          if(data[3] == 1){
            for(k=4 ; k<20 ; k++){
              read(fd, &data[k], 1);
              printf("port %2d : data = %d\n", k-3, data[k]);
            }
          }
```

```
68              }
69            }
70          }
71
72        }
73
74        tcsetattr(fd, TCSANOW, &oldtio);
75        close(fd);
76        return 0;
77    }
```

5.4 プログラム1

5.4.1 お題

- センサから障害物 (手のひらなど) までの距離を測る．サーボモータによって，計測した距離 (cm) をメータで示すプログラムの作成.

5.5 参考

5.5.1 AGB65-ADC 命令・返値仕様

・マスタ (Armadillo-300) から ADC への指示は, ４～１７バイトの数値データで送る.
・バイトサイズ（８ビット）なので, 表される数値は０～２５５の２５６種類.
・数値は１０進数.
　注）データは文字データではなく, 数値（バイナリ）データで送る.

基本形

・データの基本形は次の通り．[　] 内は１バイトを表す. () 内は送りえる数値の範囲.

[シンクロバイト (255)][ＩＤ (0-3)］[送信バイト数 (1-14)］[命令１][命令２][命令３]...

表 5.1: 各項目の説明

シンクロバイト	ＲＳＣにデータの通信開始を知らせるデータで, 常に「２５５」で始まる.
ＩＤ	ＲＳＣに設定された固有のＩＤ (AGB65 シリーズを複数接続したときの判別用．ADC の場合, 出荷時のＩＤは「120」．)
送信バイト数	送信される命令の（バイト）数．シンクロバイト, ＩＤ, 送信バイト数は数えない.
命令	ADC に動作させたい命令．詳細は以下の表.

命令の説明

表 5.2: 各項目の説明

命令値	動作	方向	フォーマット
1	全ポート読取 (8bit)	送信	[255][ID(120-127)][バイト長 (1)][命令 (1)]
		受信	[255][ID(120-127)][バイト長 (17)][命令 (1)][P1 結果][P2 結果]…[P16 結果]
2	1 ポート読取 (8bit)	送信	[255][ID(120-127)][バイト長 (2)][命令 (2)][ポート番号 (1-16)]
		受信	[255][ID(120-127)][バイト長 (3)][命令 (2)][ポート番号][結果]
１１	全ポート読取 (12bit)	送信	[255][ID(120-127)][バイト長 (1)][命令 (11)]
		受信	[255][ID(120-127)][バイト長 (33)][命令 (11)][P1_H][P1_L]...[P16_H][P16_L]
１２	1 ポート読取 (12bit)	送信	[255][ID(120-127)][バイト長 (2)][命令 (12)][ポート番号 (1-16)]
		受信	[255][ID(120-127)][バイト長 (4)][命令 (12)][P?_H][P?_L]
２５４	セルフチェック	送信	[255][ID(120-127)][バイト長 (1)][254]
		受信	[255][ID(120-127)][バイト長 (1)][254]
上以外	無視される.		

命令の処理時間

- ADC が送信フォーマットを受け取ってから返信フォーマットを返し終わって待機状態になるまでの時間
  - 8bit,1 ポート読み取り
    - 9600bps の場合：約 12ms
    - 115200bps の場合：約 1.4ms
  - 8bit, 全 16 ポート読み取り
    - 9600bps の場合：約 25ms
    - 115200bps の場合：約 2.6ms
  - 12bit,1 ポート読み取り
    - 9600bps の場合：約 13ms
    - 115200bps の場合：約 1.5ms
  - 12bit, 全 16 ポート読み取り
    - 9600bps の場合：約 42ms
    - 115200bps の場合：約 4ms

第6章　Linux組み込みボードでのサーボ・センサ

6.1　概要

・これまで扱ったサーボ・センサを両方使ってプログラムを作成する.

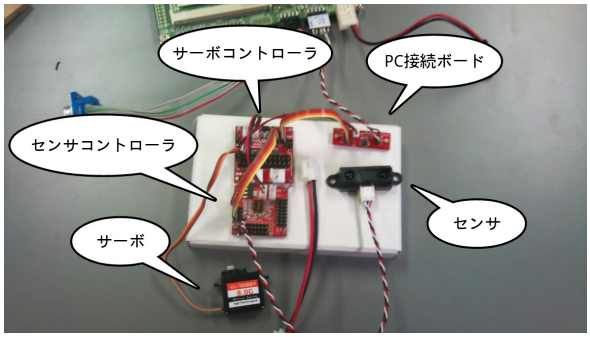

図 6.1: サーボ・センサコントローラ

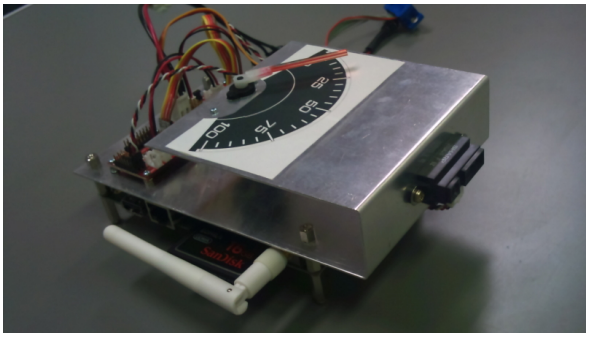

図 6.2: 組み上げ前方図

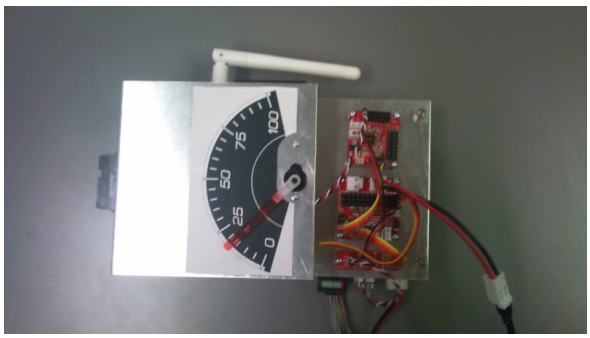

図 6.3: 組み上げ上方図

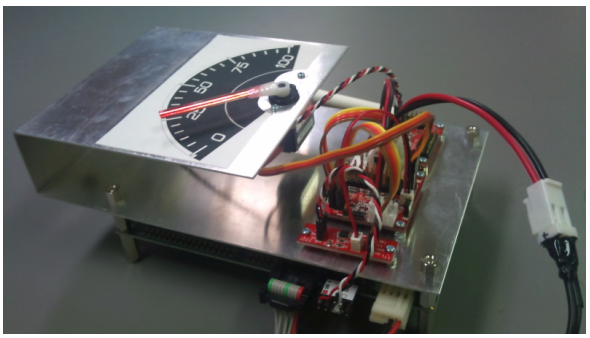

図 6.4: 組み上げ後方図

6.2　プログラム

6.2.1　お題

- 数値を入力するとメータ上の対応する値をさすようにサーボモータを制御するプログラムの作成.
  - メーターに合うように値を変換する必要がある.