

農家のロボットインストール説明書

本稿は、本書の 12 章と 13 章「農作業用ロボット製作」で使用するソフトウェアのインストール説明書です。この文書に従って、ソフトウェアのインストールをお願いいたします。

インストールおよび環境構築に必要な時間は、約2時間です。

インストールは、次の2つの手順で行います。

- ①Arch Linux と、ロボット開発・運用ソフトのインストール
- ②ロボット用ソフトウェアのインストール

Raspberry Pi の標準 OS は、Raspbian です。しかし、ロボットの場合には Raspberry Pi の性能を可能なかぎり引き出したいので、Arch Linux を使います。その OS 上にロボット開発・運用環境を構築します。

①(インストール)が終了した後に、本書で説明するロボットソフトウェアをインストールします。Raspberry Pi に対する操作は、最初の OS のインストール以外は、リモート操作で行います。したがって、操作側のコンピュータには、SSH 接続可能なターミナルエミュレータとファイル転送プログラムが必要です。

- ・Windows : Tera Term、PuTTY、WinSCP など
- ・Mac OS X : FileZilla など

SD カードは、Class10 8GB 以上を使ってください。また、すべての操作は root で行います。システム関連のファイルを操作しないように充分注意してください。

1. Arch Linux と、ロボット開発環境のインストール

1.1 aspberry Pi 起動用 SD カード作成

Raspberry Pi の公式 Web サイト(<http://www.raspberrypi.org/>)の[Downloads]より、「Arch Linux ARM」を PC などにダウンロードして解凍してください。後は、本書の Raspbian の SD カード作成 (3-3:「OS イメージを SD カードに書き込む」／20 ページ)と同じ手順で実施してください。

本書では、archlinux-hf-2013-02-11.img パッケージで確認しました。

1.2 Arch Linux の起動と SD カード拡張

上記の SD カードを Raspberry Pi に挿入後、電源を入れて起動してください。起動できなければ、再度 SD カード作成を行ってください。

作業はすべて root で行います(パスワードは root です)。最初の作業は、SD カードの全容量を使用可能にする設定です。16GB の SD カードを例に説明します。

以降の説明は、実際のコマンド入力と、Raspberry Pi からのメッセージを次の配色の組み合わせで記述します。

- ・**緑**: コマンドプロンプト(ex. **[root@alarmpi ~]#**)
- ・**黒**: キーボード入力(ex. **fdisk /dev/mmcblk0**)
- ・**青**: プログラムからの表示(ex. **Command (m for help):**)
- ・**赤**: 説明(ex. **このまま入力 パーティションの削除**)

```

[root@alarmpi ~]# fdisk /dev/mmcblk0
Welcome to fdisk (util-linux 2.21.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.
Command (m for help): p
Disk /dev/mmcblk0: 15.9 GB, 15931539456 bytes
4 heads, 16 sectors/track, 486192 cylinders, total 31116288 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000c21e5
Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1  *            2048     194559      96256    c   W95 FAT32 (LBA)
/dev/mmcblk0p2            194560     386252     183394    83   Linux
Command (m for help): d
Partition number (1-4): 2
Partition 2 is deleted
Command (m for help): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p):
Using default response p
Partition number (1-4, default 2):
Using default value 2
First sector (194560-31116287, default 194560):
Using default value 194560
Last sector, +sectors or +size{K,M,G} (194560-31116287, default 31116287):
Using default value 31116287
Partition 2 of type Linux and of size 14.8 GiB is set
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
[root@alarmpi ~]# reboot

[root@alarmpi ~]# resize2fs /dev/mmcblk0p2
resize2fs 1.42.5 (29-Jul-2012)
Filesystem at /dev/mmcblk0p2 is mounted on /; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 1
Performing an on-line resize of /dev/mmcblk0p2 to 3865216 (4k) blocks.
The filesystem on /dev/mmcblk0p2 is now 3865216 blocks long.
[root@alarmpi ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
rootfs          15G  405M   14G   3% /
/dev/root       15G  405M   14G   3% /

```

このまま入力 パーティション変更

このまま入力 現在の設定を表示

現在 2GB の使用であることがわかる

このまま入力 パーティションの削除

このまま入力 2番目を指定

このまま入力 新しいパーティション設定

Enter キー押下 (default 設定)

Enter キー押下 (default 設定)

Enter キー押下 (default 設定)

Enter キー押下 (default 設定)

このまま入力 (設定値の書き込み)

再起動

このまま入力 拡張処理 (2分くらい)

このまま入力 変更結果の確認

15GB に増えています

```
devtmpfs    92M  0 92M  0% /dev
tmpfs       92M  0 92M  0% /dev/shm
tmpfs       92M 256K 92M  1% /run
tmpfs       92M  0 92M  0% /sys/fs/cgroup
tmpfs       92M  0 92M  0% /tmp
/dev/mmcblk0p1 94M 37M 58M 39% /boot
```

1.3 Swap ファイル作成

```
[root@alarmpi ~]# dd if=/dev/zero of=/swapfile.img bs=1M count=512
512+0 records in                               Swap file 作成 512MB のファイル (1 分)
512+0 records out
536870912 bytes (537 MB) copied, 28.2801 s, 19.0 MB/s
[root@alarmpi ~]# mkswap /swapfile.img          このまま入力
Setting up swapspace version 1, size = 524284 KiB
no label, UUID=456776ef-d759-4e1e-92ec-65daa5fdc3c2
[root@alarmpi ~]# swapon /swapfile.img          このまま入力

[root@alarmpi ~]# cat /proc/swaps                swap file の確認
Filename    Type    Size    Used    Priority
/swapfile.img      file 524284 0 -1
```

```
[root@alarmpi ~]# vi /etc/fstab
起動時に mount するために/etc/fstab の最後に次の1行追加
/swapfile.img none swap sw 0 0
```

1.4 最初の Upgrade

ホスト名は、例です。ホスト名は、ご自分でお決めください。

```
[root@alarmpi ~]# pacman -Syu                  最初の upgrade (20 分くらい)
[root@alarmpi ~]# hostnamectl set-hostname rcmp-sv01 ホスト名変更
[root@alarmpi ~]# reboot
```

これで、SD card の全領域を使った、最新の Arch Linux だけ の環境が、構築できました。次は、ロボット開発・運用環境のインストールです。

1.5 無線 LAN 設定

筆者は、価格重視で、BUFFALO WLI-UC-GNM を使っています。これは、Ralink RT8070 チップを使っています。現在のインストールイメージには、Ralink RT8070 の firmware, driver とも入っていますので容易に使えます。Wi-Fi 設定の方法や使うコマンドは、他にもいろいろあります。このやり方だけが、すべてではありません。

Wi-Fi 子機を、USB 接続してください。

①wifi パッケージのインストール

```
[root@rcmp-sv01 ~]# pacman -S wpa_supplicant wireless_tools
```

②Wi-Fi 設定ファイルの編集

下の例は、家の LAN と Pocket Wi-Fi の2つの設定です。priority で同じ環境にあった場合の優先順位を決めています。優先順位は、自宅が高いです。このように、記述しておくで、自動的に切り替わります。その他は、標準的な WPAP2-PSK の設定方法であり、ssid と psk の部分を変更すれば、そのまま使えます。

```
[root@rcmp-sv01 ~]# vi /etc/wpa_supplicant/wpa_supplicant.conf
```

```
[root@rcmp-sv01 ~]# cat /etc/wpa_supplicant/wpa_supplicant.conf
```

```
# ----- add my block 19/08/2012 -----
```

```
network={
  ssid=" 自宅の ESSID"
  proto=RSN
  priority=1
  key_mgmt=WPA-PSK
  pairwise=CCMP TKIP
  group=CCMP TKIP
  psk=" 自宅用の KEY"
}
```

```
network={
  ssid=" Pocket Wi-fi の ESSID"
  proto=RSN
  priority=10
  key_mgmt=WPA-PSK
  pairwise=CCMP TKIP
  group=CCMP TKIP
  psk=" Pocket Wi-fi 用の KEY"
}
```

③手動起動

ifconfig を使っていますが、/sbin/ip link set dev wlan0 up でも同じです。お好みでどうぞ。

```
[root@rcmp-sv01 ~]# /sbin/ifconfig wlan0 up
```

```
[root@rcmp-sv01 ~]# /usr/sbin/wpa_supplicant -B -Dwext -iwlan0
-c/etc/wpa_supplicant/wpa_supplicant.conf
```

```
[root@rcmp-sv01 ~]# /usr/sbin/dhccpd wlan0
```

④接続状態の確認

```
[root@rcmp-sv01 system]# ifconfig wlan0
```

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 192.168.11.4 netmask 255.255.255.0 broadcast 192.168.11.255
  inet6 fe80::xxxx:xxxx:xxxx:95f4 prefixlen 64 scopeid 0x20<link>
  ether xx:xx:xx:xx:95:f4 txqueuelen 1000 (Ethernet)
  RX packets 304 bytes 36770 (35.9 KiB)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 21 bytes 2096 (2.0 KiB)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

1.6 IP アドレスの固定(無線、有線)

ロボット開発・運用環境は、固定 IP アドレスで行います。ご自分のネット環境に合わせて、Raspberry Pi に割当てる IP アドレスを決めてください。このインストール説明では、無線 LAN を 192.168.11.70、有線 LAN を 192.168.11.71 として説明します。ルータは、192.168.11.1 です。

① /etc/conf.d/network 作成

```
[root@rcmp-sv01 ~]# vi /etc/conf.d/network
[root@rcmp-sv01 ~]# cat /etc/conf.d/network
interface=wlan0
interface=eth0
addressw=192.168.11.70      環境に合わせて設定してください
address=192.168.11.71      環境に合わせて設定してください
netmask=24
broadcast=192.168.11.255    環境に合わせて設定してください
gateway=192.168.11.1        環境に合わせて設定してください
```

② /etc/systemd/system/network.service 作成

この設定は、無線 LAN を優先して設定し、有線 LAN を補助として使う設定方法です。したがって、無線 LAN が取り外れているときには、エラーになります。そのような運用に問題がある場合には、service をそれぞれ作ることが必要です。

```
[root@rcmp-sv01 ~]# vi /etc/systemd/system/network.service
[root@rcmp-sv01 ~]# cat /etc/systemd/system/network.service
[Unit]
Description=Network Connectivity (Static IP wlan0 eth0)
Wants=network.target
Before=network.target
[Service]
Type=oneshot
RemainAfterExit=yes
EnvironmentFile=/etc/conf.d/network
ExecStart=/sbin/ip link set dev ${interfacew} up
ExecStart=/sbin/ip link set dev ${interface} up
ExecStart=/usr/sbin/wpa_supplicant -B -D wext -i ${interfacew} -c
/etc/wpa_supplicant/wpa_supplicant.conf
ExecStart=/sbin/ip addr add ${addressw}/${netmask} broadcast ${broadcast} dev ${interfacew}
ExecStart=/sbin/ip addr add ${address}/${netmask} broadcast ${broadcast} dev ${interface}
ExecStart=/sbin/ip route add default via ${gateway}
ExecStop=/sbin/ip addr flush dev ${interfacew}
ExecStop=/sbin/ip addr flush dev ${interface}
ExecStop=/sbin/ip link set dev ${interfacew} down
ExecStop=/sbin/ip link set dev ${interface} down
[Install]
WantedBy=multi-user.target
```

③ systemctl による設定

```
[root@rcmp-sv01 ~]# systemctl disable dhcpcd@eth0.service
rm '/etc/systemd/system/multi-user.target.wants/dhcpcd@eth0.service'
[root@rcmp-sv01 ~]# systemctl disable dhcpcd
rm '/etc/systemd/system/multi-user.target.wants/dhcpcd.service'
[root@rcmp-sv01 ~]# systemctl enable network.service
ln -s '/etc/systemd/system/network.service' '/etc/systemd/system/multi-user.target.wants/network.service'
```

④ /etc/resolv.conf の修正と再起動

固定アドレスを設定した場合には、/etc/resolv.conf に DNS Server を設定します。下記の例は、ルータと Google's primary DNS(8.8.8.8)を設定しています。通常は、ルータだけで大丈夫です。

```
[root@rcmp-sv01 ~]# ping -c 3 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=46 time=46.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=46 time=50.1 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=46 time=50.2 ms
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 46.446/48.952/50.259/1.791 ms
[root@rcmp-sv01 ~]# vi /etc/resolv.conf
[root@rcmp-sv01 ~]# cat /etc/resolv.conf
# Generated by dhcpcd
# /etc/resolv.conf.head can replace this line
# /etc/resolv.conf.tail can replace this line
nameserver 192.168.11.1          # your DNS
nameserver 8.8.8.8              # 8.8.8.8 ping OK
nameserver 8.8.8.4
```

```
[root@rcmp-sv01 ~]# reboot
```

reboot 後に、/etc/resolv.conf が変わっていない場合があります。確認してください。そのときは再編集し reboot してください。dhcpcd が停止時に書き換えるようです。

⑤ 確認

```
[root@rcmp-sv01 ~]# dmesg | grep eth0
[ 2.326344] smsc95xx 1-1.1:1.0: eth0: register 'smc95xx' at usb-bcm2708_usb-1.1, smc95xx
USB 2.0 Ethernet, b8:27:eb:7a:da:c6
[ 17.464573] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[ 18.971807] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 18.974373] smsc95xx 1-1.1:1.0: eth0: link up, 100Mbps, full-duplex, lpa 0x45E1
[root@rcmp-sv01 ~]# dmesg | grep wlan0
[ 17.471293] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[ 20.137110] wlan0: authenticate with xx:xx:xx:xx:44:74
[ 20.282721] wlan0: send auth to xx:xx:xx:xx:44:74 (try 1/3)
[ 20.284445] wlan0: authenticated
[ 20.361233] wlan0: associate with xx:xx:xx:xx:44:74 (try 1/3)
[ 20.365241] wlan0: RX AssocResp from xx:xx:xx:xx:44:74 (capab=0x431 status=0 aid=7)
```



```
[ 20.381284] wlan0: associated
[ 20.381465] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
[root@rcmp-sv01 ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN    link/loopback
00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host valid_lft forever preferred_lft forever
4: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether xx:xx:xx:xx:da:c6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.71/24 brd 192.168.11.255 scope global eth0
    inet6 fe80::ba27:ebff:fe7a:dac6/64 scope link valid_lft forever preferred_lft forever
5: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 100
    link/ether xx:xx:xx:xx:95:f4 brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.70/24 brd 192.168.11.255 scope global wlan0
    inet6 fe80::126f:3fff:fe5b:95f4/64 scope link valid_lft forever preferred_lft forever
```

1.7 timezone の変更

```
[root@rcmp-sv01 ~]# timedatectl set-timezone Japan    または
[root@rcmp-sv01 ~]# timedatectl set-timezone Asia/Tokyo
```

1.8 ロボット開発用ソフトウェア

pacman を使って、下記のソフトウェアインストールをしてください。

```
[root@rcmp-sv01 ~]# pacman -S gcc make cmake pkg-config
[root@rcmp-sv01 ~]# pacman -S libtool
[root@rcmp-sv01 ~]# pacman -S git
[root@rcmp-sv01 ~]# pacman -S python2
[root@rcmp-sv01 ~]# pacman -S glut
[root@rcmp-sv01 ~]# pacman -S doxygen graphviz
[root@rcmp-sv01 ~]# pacman -S opencv
[root@rcmp-sv01 ~]# pacman -S ttf-bitstream-vera ttf-dejavu
[root@rcmp-sv01 ~]# pacman -S fswebcam
[root@rcmp-sv01 ~]# pacman -S monkey unzip
```

1.9 HTTP Server インストール

ロボットの起動、停止、データ確認は、Web で行います。HTTP Server の設定を行います。Monkey は組み込み用機器でも動く軽量の HTTP Server です。Apache と同じ考え方で設定可能です。

Monkey 自体の問題ではありませんが、注意点が 1 つあります。pacman でアップグレードすると設定ファイルが上書きされます。必ず、/etc/monkey の設定はコピーしておいてください。ただし、upgrade により設定方法の変更があるかもしれませんので、上書きされた内容の確認は必要です(執筆時点)。

① 起動設定ファイルの編集

```
[root@rcmp-sv01 ~]# vi /etc/monkey/monkey.conf
"User nobody" to "User root"    change your root group user

[root@rcmp-sv01 ~]# vi /etc/monkey/plugins.load
# CGI
===
# This plugin enable the CGI support in Monkey. CGI is an old
```

```
# interface and not recommended for production enviroments, due
# to it nature, it lack of performance.
#
Load /usr/lib/monkey/monkey-cgi.so          enable
```

```
[root@rcmp-sv01 ~]# vi /etc/monkey/sites/default
[HOST]
# DocumentRoot :
DocumentRoot /var/http                      enable & set
[CGI]                                       enable
# Per-vhost CGI matching, same rules as with the global match
Match /cgi-bin/*.cgi                       enable
[root@rcmp-sv01 ~]# vi /etc/monkey/plugins/cgi/cgi.conf
[CGI]
Match /cgi-bin/*.cgi.*.sh.*.pl             add extensions
```

② ファイル作成をアクセス権の変更

```
[root@rcmp-sv01 ~]# touch /var/log/monkey/access.log
[root@rcmp-sv01 ~]# touch /var/log/monkey/error.log
[root@rcmp-sv01 ~]# chmod +w /var/log/monkey/access.log
[root@rcmp-sv01 ~]# chmod +w /var/log/monkey/error.log
[root@rcmp-sv01 ~]# mv /srv/http /var
[root@rcmp-sv01 ~]# mkdir /var/http/cgi-bin
[root@rcmp-sv01 ~]# rm -r /srv
```

③ 手動起動による確認

```
[root@rcmp-r03 ~]# monkey -D
Monkey HTTP Daemon 1.1.1
Built : Dec 14 2012 14:16:18 (gcc 4.7.2)
Home : http://monkey-project.com
* Process ID is 1677
* Server socket listening on Port 80
* 5 threads, 101 client connections per thread, total 505
* Transport layer by liana in http mode
[2013/01/22 12:55:33] [ Info] HTTP Server started
```

④ 自動起動登録

```
[root@rcmp-r03 ~]# vi /etc/systemd/system/monkey.service
[root@rcmp-r03 ~]# cat /etc/systemd/system/monkey.service
[Unit]
Description=Monkey http server deamon
Required=network.target
After=network.target
[Service]
```



```
Type=forking
ExecStart=/usr/bin/monkey -D
ExecStop=/bin/kill $MAINPID
ExecReload=/bin/kill $MAINPID ; /usr/bin/monkey -D
PIDFile=/var/run/monkey.pid.80
[Install]
WantedBy=multi-user.target
[root@rcmp-r03 ~]# systemctl enable monkey.service
ln -s '/etc/systemd/system/monkey.service' '/etc/systemd/system/multi-
user.target.wants/monkey.service'
[root@rcmp-r03 ~]# reboot
```

再起動後に、設定した HTTP Server にアクセスして下記のイメージが表示されれば稼動しています。



2. ロボット用ソフトウェアのインストール

2.1 ダウンロード と設定

本書のサポートページより「nouka-robot.zip」をダウンロードします。ダウンロードしたファイルは、Raspberry Pi の /root において解凍します。

```
[root@rcmp-sv01 ~]# unzip nouka-robot.zip
```

次に、ロボットの Web 環境をコピーします。

```
[root@rcmp-sv01 ~]# cp -r ./http /var/http
```

再度、HTTP Server にアクセスして下記の画面が現れれば設定完了です。



All it needs is courage, imagination, and a little dough

[Tomato Farmer's robot.robotEye](#)

[Tomato Farmer's robot](#)

[Robot creation by tomato farmer](#)

すべてのインストールと設定が終了しました。