GeckoBot ControlBoard Manual

Lars Schiller (AmP)

December 9, 2019

Contents

1 Quickstart

1.1 Power on the Board

- Check if all Potentionmeter are in zero position (turned left)
- Check if all small black switches are OFF (switched up)
- Check if 24V Switch is OFF (up)
- Check if pressure source is zero (throttle valve turned left, manometer shows 0 bar)
- Check if I2C Interface is connected to robot or pull-up circuit board (red face on red face!)
- Power on main switch of ControlBoard

1.2 Log into BBB

- You will need a computer with LAN access in the AmP network, and a terminal with ssh capabilities. Putty on Windows. Or standard Terminal on Linux.
- For Windows (Putty):

Hostname	Port
134.28.136.51	22

- Linux:

```
bianca@bianca:~ ssh root@134.28.136.51
```

- login: root   password: root

1.3 Running the Code

To run the geckobot code:

- on BBB as **su**

```
root@beaglebone:~# cd Git/GeckoBot/Code
root@beaglebone:~Git/GeckoBot/Code/# python3 main.py
```

If you want to run the GUI (to have the ability to save experimental data) run the the following piece of code on `bianca` with **python2**:

```
bianca@bianca:~ cd Git/GeckoBot/Code
bianca@bbianca:~Git/GeckoBot/Code/ python pc_liveplotter_main.py
```

- **Note** that you must start both programs more or less at the same time. (main on bbb is tries to connect to bianca. if there is no listing port at bianca, the gui wont start)
- In case there is an error related to the device tree, just run the code again.

1.4 Enable Pneumatic Power

When control program runs, you can enable the pneumatic energy sources:

- 24V Switch ON (down) (valves are enabled now)
- Pressure Source 1.2 bar (turn throttle valve right until manometer shows the 1.2bar)
- Plug in Vacuum Source if needed.

1.5 When your session is over

If you are done with your experiments, first disable the pneumatic power supply, then quit the control program.

- Pressure Source to 0bar
- 24V Switch OFF (up)
- All Potentiometer to zero
- All small black Switches OFF (up)
- Abort the `main.py` programm on BBB by hitting **Ctrl+C**

  ```
  root@beaglebone:~Git/GeckoBot/Code/# python3 main.py
  ... ^C [a lot of errors]
  root@beaglebone:~Git/GeckoBot/Code/#
  ```

  Note the last line in the listing above. This indicates that the terminal is ready again. This line **should** be there. Otherwise some process(es) are still running in the background... (You can use `htop` command and filter (F4) python programs to kill them. You have to use a different terminal for this purpose).

- Main Switch of ControlBoard off
- If it's friday or you know that nobody will use the ControlBoard in the next few days, you may shutdown `bianca` and `Dell Latitude`.

2 Setting Up the BBB

2.1 Install OS on BBB

The developers of BBB embedded linux systems decided to change the device tree structure from `kernel` overlay (till version 8.7), to `uboot` overlay (9.1+). (Don't ask me to explain). However, the PWM setup for all pins is only possible with `kernel` overlay (or at least I'm not able to configure it in version 9.1+). Therefore you have to use the following image:
`bone-debian-8.7-iot-armhf-2017-03-19-4gb.img` (Download: `http://beagleboard.org/latest-images`)
To install it on a `8GB` Micro-SD Card follow the instructions:

- You can use Etcher (`https://etcher.io/`).

OR (on debian):

- Instructions from: `http://derekmolloy.ie/write-a-new-image-to-the-beaglebone-black/`

  and from: `https://learn.adafruit.com/beaglebone-black-installing-operating-systems?view=all#copying-the-image-to-a-microsd`

- Decompress and write on SD card (need to be `su` and make sure the security locker of SD Adapter is in writing mode):

```
$ xz -d bone-debian-**.img.xz
$ dd if=./bone-debian-**.img of=/dev/sdX
```

  (Here, `sdX` is the mounted empty uSD Card. It can be found with multiple use of the command `mount` or `df`.)

- Obsolete:
  - In order to turn these images into eMMC flasher images, edit the `/boot/uEnv.txt` file on the BBB and remove the `#` on the line with

    `cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh`.

    Enabling this will cause booting the microSD card to flash the eMMC. Images are no longer provided here to avoid people accidentally overwriting their eMMC flash.
  - Insert the SD Card in the unpowered BBB, and power it by plugging in the USB or the 5VDC supply. Wait until all 4 LED have solid lights. This can take up to 45 minutes.
  - Flash MicroSD 4 with: `Debian 8.7 2017-03-19 4GB SD IoT` from `http://beagleboard.org/latest-images` (MicroSD 3 is weird ...).
  - Insert MicroSD in (unpowered) BBB, press the USER Button, and apply power.
  - It will take 30-45 minutes to flash the image onto the on-board chip. Once it is done, the bank of 4 LEDs to the right of the Ethernet port will all turn off. You can then power down your BBB.

2.2 Log in BBB for the first time

Assuming you are called `bianca` and your PC is also called `bianca`, your BBB is called `beaglebone` and the default user on BBB is called `debian`, then the following sythax is correct.

- Connect your PC with a MicroUSB cable to the BBB.
- Open a terminal and ssh into BBB as `debian` and then get superuser to configure the Board.

```
bianca@bianca:~ ssh debian@192.168.7.2
temppwd
debian@beaglebone:~ su
root
root@beaglebone:~#
```

- Note that the default passwords are:

temppwd	for `debian`
root	for `root`

2.3 Change static IP of USB port

https://stackoverflow.com/questions/23805457/changing-the-static-ip-of-beagle-bone-black-usb0

- To change the static ip of BBB's usb0 interface from default 192.168.7.2 to ...5.2:

```
root@bbb:~# nano /etc/network/interfaces

iface usb0 inet static
        address 192.168.5.2
        netmask 255.255.255.0
        network 192.168.5.0
        gateway 192.169.5.1
```

- (I also edited the file /opt/scripts/boot/am335_evm.sh. Maybe it had an effect...)

2.4 Set LAN connection on BBB at AmP

This is from:
https://groups.google.com/forum/#!msg/beaglebone/AS2US9rtNd4/8y0mZ3LxAwAJ

- You have to configure eth0 like this:

address	134.28.136.51 (ask administrator for your personal IP)
netmask	255.255.255.0
dns-nameservers	134.28.205.14
gateway	134.28.136.1

- Plug in LAN cable.
- Get the name of the LAN connection:

```
su
root@beaglebone:/etc/network# connmanctl services
*Ac Wired                ethernet_689e19b50543_cable
```

- Using the appropriate ethernet service, tell connman to setup a static IP address for this service.

  Syntax:

```
connmanctl config <service> --ipv4 manual <ip_addr> <netmask> <gateway> --nameservers <
    dns_server>
```

  In our case:

```
connmanctl config ethernet_689e19b50543_cable --ipv4 manual 134.28.136.51 255.255.255.0
    134.28.136.1 --nameservers 134.28.205.14
```

- Reboot and you are done.
- You can revert back to a DHCP configuration simply as follows:

```
$ sudo connmanctl config ethernet_689e19b50543_cable --ipv4 dhcp
```

2.5 Configure SSH Connection to BBB

- Source: https://askubuntu.com/questions/115151/how-to-set-up-passwordless-ssh-access-for-root-user
- If your Board crashed, and you were forced to reinstall the OS, there already exist a ssh-key. This you have to remove first (this is for USB cable):

```
bianca@bianca:~ ssh-keygen -f "/home/bianca/.ssh/known_hosts" -R 192.168.7.2
```

- Generate a new key:

```
bianca@bianca:~ ssh-keygen -f "/home/bianca/.ssh/key_bianca"
```

  When you are prompted for a password, just hit the enter key and you will generate a key with no password.

- Allow to log in as root with a password on the server, in aim to transfer the created key to it:

```
root@beaglebone:# nano /etc/ssh/sshd_config
```

  Make sure you allow root to log in with the following syntax

```
PermitRootLogin yes
PasswordAuthentication yes
```

  Restart the ssh-server:

```
root@beaglebone:# service ssh restart
```

- Now you are able to transfer the key to the server:

```
bianca@bianca:~ ssh-copy-id -i /home/bianca/.ssh/key_bianca root@192.168.7.2
```

- Check if its work:

```
bianca@bianca:~ ssh root@192.168.7.2
```

- Now disable root login with password on server (for saftey):

```
root@beaglebone:# nano /etc/ssh/sshd_config
```

  And modify the Line:

```
PermitRootLogin without-password
PasswordAuthentication yes
```

  This will allow to login as root with valid key, but not with a password. All other users can further login with a password. Restart the ssh-server and you are done:

```
root@beaglebone:# service ssh restart
```

2.6 Configure BBB Device Tree

In order to enable `P9.28` as pwm pin, you have to load `cape-universala`. This you gonna do in `/boot/uEnv.txt`:

- source: `https://groups.google.com/forum/#!topic/beagleboard/EYSwmyxYjdM`
- `/boot/uEnv.txt` should be looking something like this:

```
root@beaglebone:# cat /boot/uEnv.txt | grep -v "#"

uname_r=4.4.54-ti-r93
cmdline=coherent_pool=1M quiet cape_universal=enable
```

  Edit it with:

```
root@beaglebone:# nano /boot/uEnv.txt
```

  Add the following lines, such that `/boot/uEnv.txt` looks like:

```
root@beaglebone:# cat /boot/uEnv.txt | grep -v "#"

uname_r=4.4.54-ti-r93
dtb=am335x-boneblack-overlay.dtb
cmdline=coherent_pool=1M quiet cape_universal=enable
cape_enable=bone_capemgr.enable_partno=cape-universala
```

- Reboot and you should be able to configure with:

```
root@beaglebone:# config-pin P9_28 pwm
```

Note:

- In `debian-elinux-version-9.1+` the `/boot/uEnv.txt` looks like:

```
root@beaglebone:# cat /boot/uEnv.txt | grep -v "#"

uname_r=4.9.82-ti-r102
enable_uboot_overlays=1
enable_uboot_cape_universal=1
cmdline=coherent_pool=1M net.ifnames=0 quiet
```

  If you see this, you may want to find a way to enable all the pins. I failed.
  Robert C Nelson seems to be the only one, who has an idea whats going on... `https://elinux.org/Beagleboard:BeagleBoneBlack_Debian#U-Boot_Overlays`

Debian 9 / Kernel v4.14.71-ti-r80:

- Note: you might need to disable HDMI with `disable_uboot_overlay_video=1` in `/boot/uEnv.txt` if the pins are already in use.
- update bootloader (`https://elinux.org/Beagleboard:BeagleBoneBlack_Debian#U-Boot_Overlays`)
  Check version (19-08-07):

```
root@beaglebone:~$ cd /opt/scripts/tools/
root@beaglebone:/opt/scripts/tools$ git pull
root@beaglebone:/opt/scripts/tools$ ./version.sh | grep bootloader
bootloader:[eMMC-(default)]:[/dev/mmcblk1]:[U-Boot 2016.01-00001-g4eb802e]:[location: dd
    MBR]
```

  To upgrade your version of U-Boot:

```
root@beaglebone:~$ cd /opt/scripts/tools/developers/
root@beaglebone:/opt/scripts/tools/developers$ ./update_bootloader.sh
root@beaglebone:/opt/scripts/tools/developers$ reboot

...

root@beaglebone:/opt/scripts/tools$ ./version.sh | grep bootloader
bootloader:[microSD-(push-button)]:[/dev/mmcblk0]:[U-Boot 2019.04-00002-gbb4af0f50f]:[
    location: dd MBR]
bootloader:[eMMC-(default)]:[/dev/mmcblk1]:[U-Boot 2016.01-00001-g4eb802e]:[location: dd
    MBR]
```

  Delete the old version:

```
root@beaglebone:/opt/scripts/tools$ dd if=/dev/zero of=/dev/mmcblk1 bs=1M count=10
```

  also make sure the bb-cape-overlays package is upto date

```
apt update
apt install --only-upgrade bb-cape-overlays
```

2.7 Set I2C Bus to FastMode (400kHz)

- **Kernel version 4.14.xx**
- For Kernel version < 4.4.xx replace `am335x-boneblack.dtb` with `am335x-boneblack-overlay.dtb`
- Backup the original `.dtb`:

```
root@beaglebone: /boot/dtbs/4.14.71-ti-r80# cp am335x-boneblack.dtb am335x-boneblack.dtb
    .orig
```

- Generate source device tree (`.dts`) from binary block device tree (`.dtb`) with device tree compiler (`dtc`):

```
root@beaglebone: /boot/dtbs/4.14.71-ti-r80# dtc -I dtb -O dts -o am335x-boneblack.dts
    am335x-boneblack.dtb
```

- There are 3 diffrent i2c-buses in the `.dts`:
  - i2c0: 0x44E0B000 (Not available as Pins)
  - i2c1: 0x4802A000 (Not enabled by default)
  - i2c2: 0x4819C000 (The actual one for configured i2c-1 in Linux-Debian, although the register name/-expansion port is i2c2)

  We want to increase the speed of the `i2c2` bus. Therefore modify the `.dts` with nano:

```
i2c@4819c000 {
compatible = "ti,omap4-i2c";
#address-cells = <0x1>;
#size-cells = <0x0>;
ti,hwmods = "i2c3";
reg = <0x4819c000 0x1000>;
interrupts = <0x1e>;
status = "okay";
pinctrl-names = "default";
pinctrl-0 = <0x35>;

#clock-frequency = <0x186a0>;
clock-frequency = <0x61a80>;

linux,phandle = <0xa1>;
phandle = <0xa1>;
```

  The `clock-frequency = <0x186a0>` is the frequency, `0x186a0 = 100000 = 100kHz` here is the default i2c-1 (Expansion port i2c2) frequency for stock beaglebone black image. `0x61a80 = 400000 = 400kHz` is the highest frequency possible for i2c-devices. This we gonna use.

- Generate the `.dtb` from this modified .dts:

```
root@beaglebone: /boot/dtbs/4.14.71-ti-r80# dtc -I dts -O dtb -o am335x-boneblack.dtb
    am335x-boneblack.dts
```

- reboot and check:

```
root@beaglebone:# dmesg | grep i2c
```

  Something like

```
...
omap/i2c@4819c000 is enabled at 400kHz
...
```

  should be the output.

2.8 Installing Software on BBB

In order to run the `GeckoBot` software on the BBB install following packages:

- **python3**: on BBB as `su`

```
root@beaglebone:# apt-get update
root@beaglebone:# apt-get install ntpdate
root@beaglebone:# ntpdate pool.ntp.org
root@beaglebone:# apt-get install build-essential python3-pip python3-scipy python3-
    numpy -y

root@beaglebone:# pip3.5 install Adafruit_BBIO Adafruit_GPIO board Adafruit-Blinka
    adafruit-circuitpython-charlcd


root@beaglebone:~# mkdir Git
root@beaglebone:~# cd Git
root@beaglebone:~/Git/# git clone https://github.com/larslevity/GeckoBot.git
```

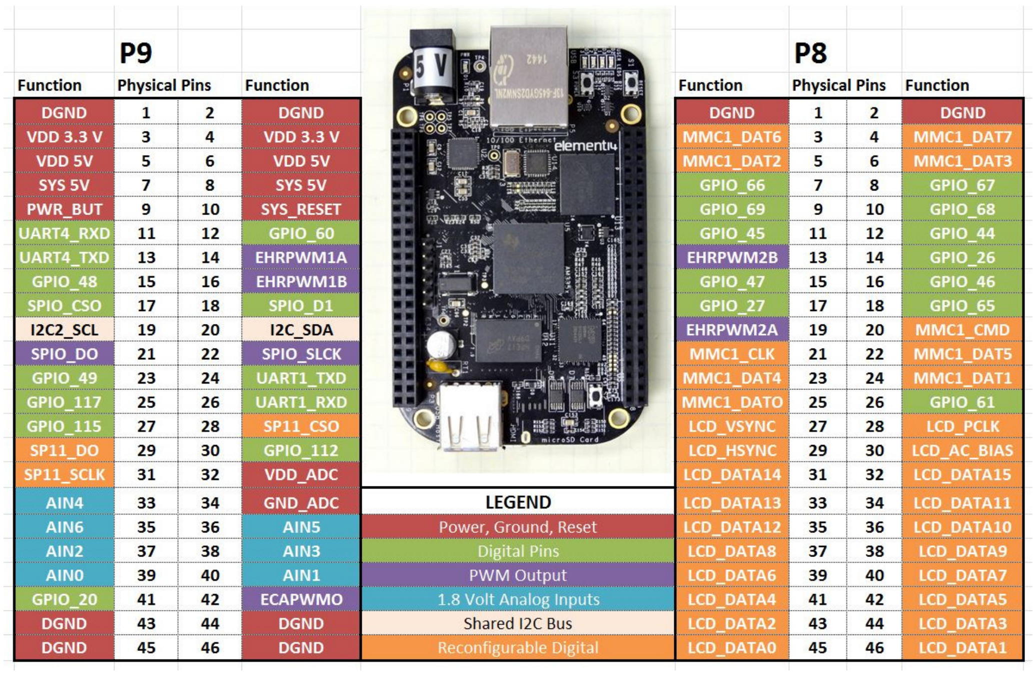
- **python2**: on BBB as `su`

```
root@beaglebone:# apt-get update
root@beaglebone:# apt-get install ntpdate
root@beaglebone:# ntpdate pool.ntp.org
root@beaglebone:# apt-get install build-essential python-dev python-pip -y
root@beaglebone:# pip install --upgrade pip
root@beaglebone:# pip install Adafruit_BBIO
root@beaglebone:# pip install Adafruit_GPIO
root@beaglebone:# pip install termcolor
root@beaglebone:# pip install numpy

root@beaglebone:~# mkdir Git
root@beaglebone:~# cd Git
root@beaglebone:~/Git/# git clone https://github.com/larslevity/GeckoBot.git
```

3 Pin Layout

Figure **??** shows all available pins and there functions of the Beaglebone Board Black and which of these pins are used and for what purpose.


5 V
10/100 Ethernet
element14
microSD Card

P9

Function	Physical Pins		Function
DGND	1	2	DGND
VDD 3.3 V	3	4	VDD 3.3 V
VDD 5V	5	6	VDD 5V
SYS 5V	7	8	SYS 5V
PWR_BUT	9	10	SYS_RESET
UART4_RXD	11	12	GPIO_60
UART4_TXD	13	14	EHRPWM1A
GPIO_48	15	16	EHRPWM1B
SPIO_CSO	17	18	SPIO_D1
I2C2_SCL	19	20	I2C_SDA
SPIO_DO	21	22	SPIO_SLCK
GPIO_49	23	24	UART1_TXD
GPIO_117	25	26	UART1_RXD
GPIO_115	27	28	SP11_CSO
SP11_DO	29	30	GPIO_112
SP11_SCLK	31	32	VDD_ADC
AIN4	33	34	GND_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	ECAPWMO
DGND	43	44	DGND
DGND	45	46	DGND

P8

Function	Physical Pins		Function
DGND	1	2	DGND
MMC1_DAT6	3	4	MMC1_DAT7
MMC1_DAT2	5	6	MMC1_DAT3
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
EHRPWM2B	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
EHRPWM2A	19	20	MMC1_CMD
MMC1_CLK	21	22	MMC1_DAT5
MMC1_DAT4	23	24	MMC1_DAT1
MMC1_DATO	25	26	GPIO_61
LCD_VSYNC	27	28	LCD_PCLK
LCD_HSYNC	29	30	LCD_AC_BIAS
LCD_DATA14	31	32	LCD_DATA15
LCD_DATA13	33	34	LCD_DATA11
LCD_DATA12	35	36	LCD_DATA10
LCD_DATA8	37	38	LCD_DATA9
LCD_DATA6	39	40	LCD_DATA7
LCD_DATA4	41	42	LCD_DATA5
LCD_DATA2	43	44	LCD_DATA3
LCD_DATA0	45	46	LCD_DATA1

LEGEND
Power, Ground, Reset
Digital Pins
PWM Output
1.8 Volt Analog Inputs
Shared I2C Bus
Reconfigurable Digital

P9

DGND	1	2	
Btn PWR 3.3V	3	4	
Sens PWR 5V	5	6	
	7	8	
	9	10	
Dvalve 0 Ref	11	12	
Dvalve 3 Ref	13	14	PValve 4 Signal
Dvalve 2 Ref	15	16	PValve 5 Signal
Dvalve 1 Ref	17	18	
I2C Clock	19	20	I2C Data
PValve 2 Signal	21	22	PValve 0 Signal
PBtn Mode 1	23	24	PBtn Fun 1
	25	26	PBtn Fun 2
PBtn Mode 2	27	28	PValve 6 Signal
	29	30	PBtn Mode 3
	31	32	ADC PWR 1.8V
PValve 2 Ref	33	34	ADC GND
PValve 5 Ref	35	36	PValve 4 Ref
PValve 6 Ref	37	38	PValve 1 Ref
PValve 3 Ref	39	40	PValve 0 Ref
	41	42	PValve 7 Signal
	43	44	
	45	46	

P8

	1	2	
	3	4	
	5	6	
Dvalve 1 Signal	7	8	Dvalve 2 Signal
Dvalve 3 Signal	9	10	Dvalve 0 Signal
	11	12	
PValve 3 Signal	13	14	LED Mode 2
LED Mode 1	15	16	LED Fun 1
LED Mode 3	17	18	LED Fun 1
PValve 1 Signal	19	20	
	21	22	
	23	24	
	25	26	
	27	28	
	29	30	
	31	32	
	33	34	
	35	36	
	37	38	
	39	40	
	41	42	
	43	44	
	45	46	

Legend:

pwr
i2c
gpio
pwm
ain

from 28. Juni 2018

Figure 1: Pin layout of BBB

4 Wiring the Hardware

4.1 The User Interface

Figure **??** shows the circuit of the User Interface. It consists of:

- 5 Push-Buttons (3 of them are used as reference for different operating modes, and 2 are used to enable/disable different functions inside these operating modes)
- 5 light emitting diodes, which indicates the actual status of programme, i.e. the operating mode.
- 8 potentiometer, which are used to read the reference signal for the proportional valves.
- 4 switches, which are used to read the reference signal for the discrete valves.
- 9 pull-down resistors, which pull the reference signal for discrete valves and operating modes down again, after activation.

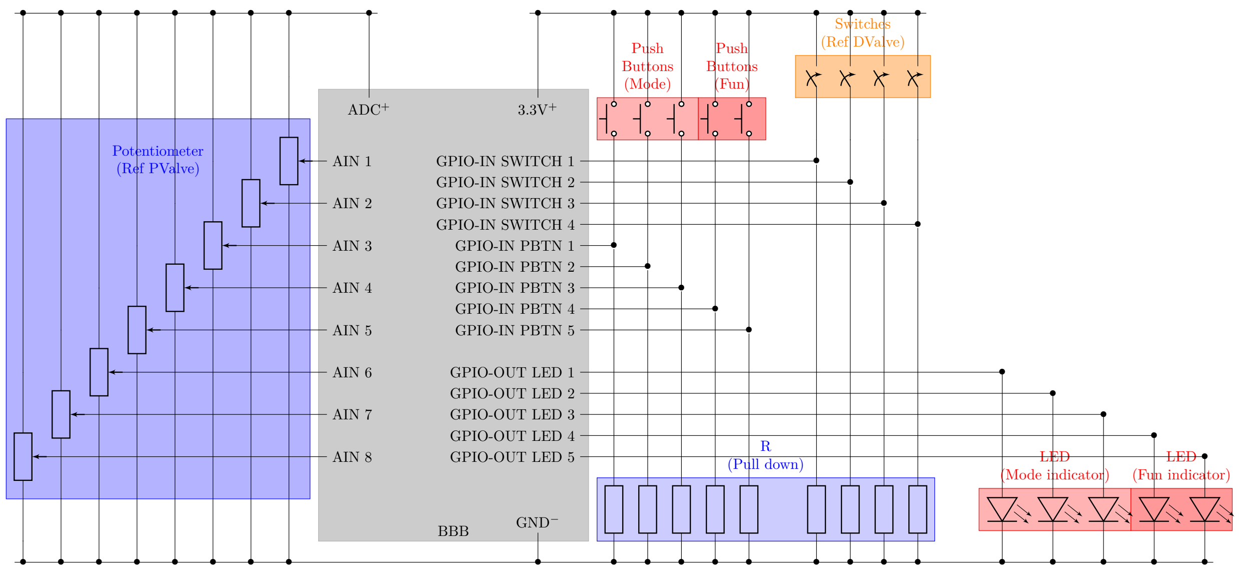
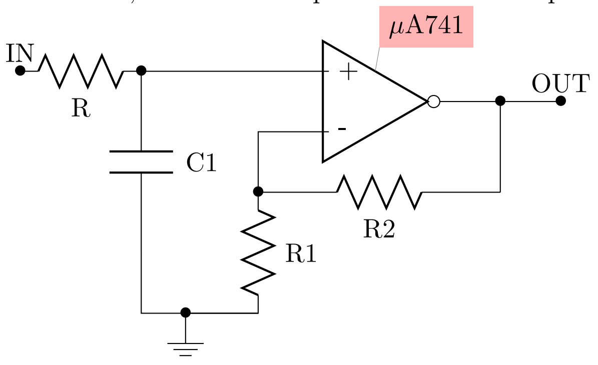


Figure 2: User Interface Wiring

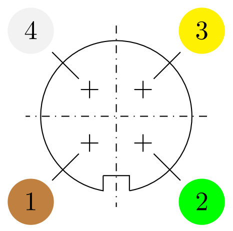
4.2 Proportional Valves

To generate the control signal for the proportional valves, `pwm` is used. Since the pwm-signal oscillating and its level is 3.3V, it must be lowpass-filtered and amplified. Therefore the following circuit is used:



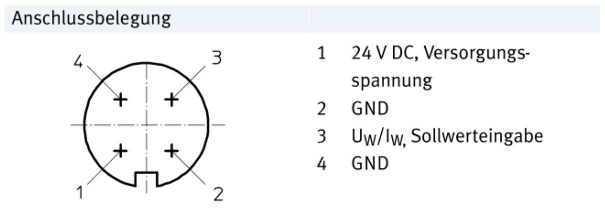
R	510 Ω
R1	11 kΩ
R2	22 kΩ
C	220 μF

For the proportional valves, the used cable (status: 28.6.18) has the following color scheme (accordingly to the data sheet[?, p. 9]):



1	24VDC (brown)
2	GND (green)
3	U_W (yellow)
4	GND (gray!10)

Potentiometer	10	kOhm
Pull Down Resistor	22 (?)	kOhm



Anschlussbelegung

1	24 V DC, Versorgungs-spannung
2	GND
3	U_W/I_W, Sollwerteingabe
4	GND

4.3 Discrete Valves

The discrete valves are controlled directly via a GPIO. The signal controls a mosfet [?]. A ready-to-use Arduino module is available:

4.4 Pressure Sensors

The following table is from [?, p. 30]. It shows the PinOut of the used pressure Sens. The figure below shows the numbering scheme of the pressure sensors [?, p. 19].

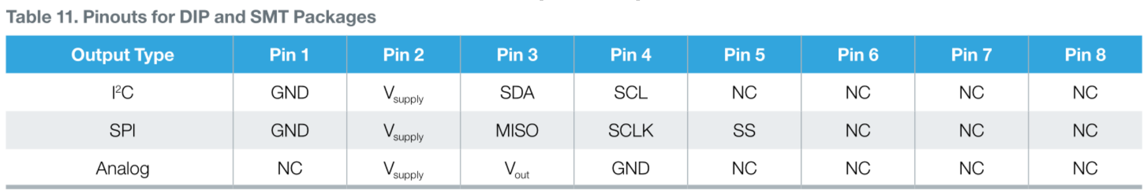
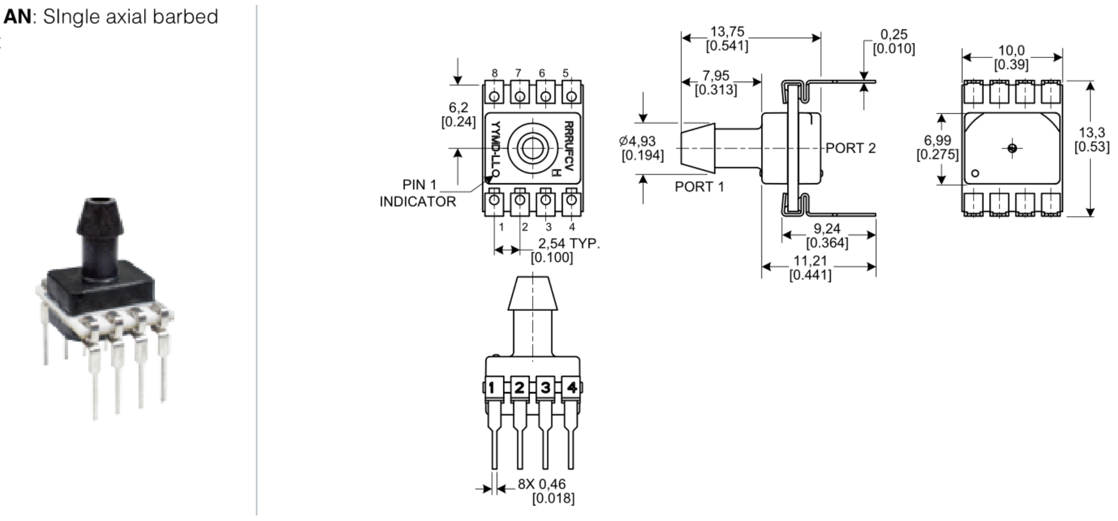


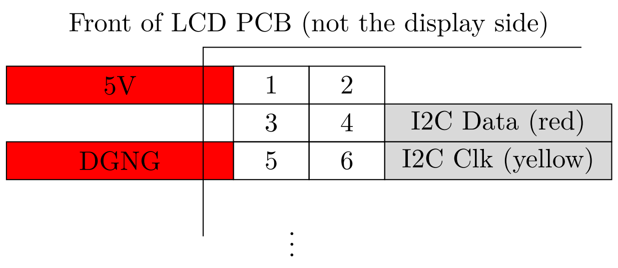
Table 11. Pinouts for DIP and SMT Packages

Output Type	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8
I²C	GND	V_{supply}	SDA	SCL	NC	NC	NC	NC
SPI	GND	V_{supply}	MISO	SCLK	SS	NC	NC	NC
Analog	NC	V_{supply}	V_{out}	GND	NC	NC	NC	NC

DIP AN: SIngle axial barbed port



4.5 LCD-Display



LCD-Display is connected to I2C.

5 Auxilary

5.1 IP-Addresses in AmP

Subnet	255	.	255	.	255	.	0
Route / Gateway	134	.	28	.	136	.	1
DNS	134	.	28	.	202	.	14
alt. DNS	134	.	28	.	205	.	14
Main	134	.	28	.	136	.	30
BBB CBoard	134	.	28	.	136	.	51
VR - Mond / T500-Schiller	134	.	28	.	136	.	129
VR - Bianca	134	.	28	.	136	.	131
RaspPi IMUCam	134	.	28	.	136	.	49
RaspPi GeckoCam	134	.	28	.	136	.	118
DellLat CBoard	134	.	28	.	136	.	70
BBB Rohat	-	.	-	.	-	.	110
BBB experiment	-	.	-	.	-	.	55

5.2 Formatting SD Card with debian

- Source: `https://www.techwalla.com/articles/how-to-format-an-sd-card-in-debian-linux`
- Determine location of SDCard (in the following called: `/dev/mmcblk0p2`) and directory where it is mounted (in the following called: `/media/SDCard`):

```
su
df
```

- Unmount, format, and remount:

```
umount /dev/mmcblk0p2
mkdosfs /dev/mmcblk0p2 -F16
mount /dev/mmcblk0p2 /media/SDCard
```

- For formatting SD with more than one partition, use:

```
cfdisk /dev/mmcblk0
```

  and follow the instructions.

5.3 Set WiFi connection

- Order WiFi Antenna `TP-LINK WLAN LITEN HI.G USB ADA. WN722N` from somewhere.
- Complete this tuturial ...

5.4 Setup for analog inputs

- https://groups.google.com/forum/#!topic/beagleboard/Lk3vWNIExiQ
- Insert in command line on BBB:

```
su apt-get install bb-cape-overlays

cd /opt/source/bb.org-overlays

./dtc-overlay.sh

./install.sh

sudo sh -c "echo 'BB-ADC' > /sys/devices/platform/bone_capemgr/slots"
```

- Reboot.
- For readout the ADC input Pins from python: https://learn.adafruit.com/setting-up-io-python-library-on-beaglebone-black/adc

5.5 Autorun

In order to autorun this script after booting the BBB use crontab like this:

```
root@beaglebone:# crontab -e -u root
```

adding the following lines to the cron boot jobs:

```
@reboot config-pin P9_28 pwm
@reboot python /home/debian/Git/GeckoBot/Code/server_hardware_controlled.py &
```

NOTE: Dont forget the & at the end. Otherwise it will block the console. And you wont be able to ssh into it. But with the & it will run as background process and will be able to ssh into the BBB.
Ending Background Processes
Since the python script will run in the background, we need to find it and end it manually. Enter this to find the processing running off the file we wrote earlier.

```
ps aux | grep home/debian/GeckoBot/Code/server_hardware_controlled.py
```

You will get something like this:

```
    root    873    0.1    0.6    7260    3264    ?   S    22:19    0:01  python home/debian/
        GeckoBot/Code/server_hardware_controlled.py
```

The number 873 is the process ID. Then, just use the process ID and kill the process.

```
root@beaglebone:# kill 873
```

Ref: https://billwaa.wordpress.com/2014/10/03/beaglebone-black-launch-python-script-at-boot-like-arduino-sketch/

Okay, cron gives error: try with daemontools - Ref: http://samliu.github.io/2017/01/10/daemontools-cheatsheet.html
– This is super weird! starting the script every time a error occurs again.

To see what happens in crontab, create a Crontab Logger:

```
crontab -e:
    @reboot /home/debian/Git/GeckoBot/boot_autorun_test/ssh_hack.sh 2>&1 |
        /home/debian/Git/GeckoBot/boot_autorun_test/timestamp.sh >>
        /home/debian/Git/GeckoBot/boot_autorun_test/log/cronlog.log
```

ssh Hack: For some reason the BBIO.PWM module needs a terminal (tty) to initialize. A Job, started by crontab does not have a tty. There is simply no tty. Therefore we ssh into the device from the device itself. So we create a virtual tty. To do so run the ssh_hack.sh script. it will automatically run the start script. But you must enable a ssh-login as root without password. 2 Steps:
1. disable root pw (to clear the password):

```
passwd -d root
```

editing

```
nano /etc/pam.d/common-auth
```

Find the pam_unix.so line and add nullok to the end if its not there or change nullok_secure to be just nullok if yours says nullok_secure.
2. allow ssh to root login without password: Ref: https://askubuntu.com/questions/115151/how-to-set-up-passwordless-ssh-access-for-root-user
Basically, we have to create a public key for root and copy it to the BBB itself. Just follow the instructions on Ref above. But dont set PasswordAuthentication to no! Since than nobody can login with a password anymore, only with a public key. Which is not yet created anywhere else except on the BBB itself.

```
        nano /etc/ssh/sshd_config
            PermitRootLogin without-password
```

3. restart ssh service:

```
service ssh restart
```

4. disable requiretty for root:

```
visudo
```

and add Defaults: root !requiretty
5. spawn a shell:
Ref:https://netsec.ws/?p=337
Error: stdin is no tty: https://michaelseiler.net/2013/04/25/cron-jobs-and-ssh-errors-tty-and-sudo/
https://sachinpradeeplinux.wordpress.com/2012/09/28/stdin-is-not-a-tty-error/
On the destination server, edit /root/.bashrc file and comment out the mesg y line.
If it is no there, please add the following line to .bashrc file .

```
if `tty -s`; then
 mesg n
fi
```

5.6 Disable Webserver on BBB

Webserver

```
root@BBB: systemctl stop apache2.service
root@BBB: systemctl disable apache2.service
```

NodeJS:

```
root@BBB: systemctl stop bonescript-autorun.service
root@BBB: systemctl disable bonescript-autorun.service
```