Raspberry Pi Solar Power Module User Manual

Table of Contents

Quick Start Guide

1. Disconnect all power sources. Micro USB power should never be connected to the raspberry PI, it will get it's power from the solar power module.
2. Connect the solar power module to the Raspberry Pi and secure it with stand offs and screws.
3. Connect Interface cables such as ethernet or Video out to the Raspberry Pi.
4. Connect a battery to the male terminals labeled BAT- and BAT+, the 5V power to the Raspberry Pi will soon turn on and the Pi it will start to power up.
5. Connect Solar Panel to the male terminals labeled GND and Panel +
6. Follow the software install section below.

Software Install

1. Download software from the product page on the website, copy it to any folder on the Raspberry Pi.
2. Unzip the .zip file, if working from command line use ***sudo unzip <file_name>***
3. Go into the unzipped install directory, ***cd dir_name/install***
4. Make install.sh executable with ***sudo chmod 777 install.sh***
5. Run ***sudo ./install.sh -install***
6. Run ***solar_pi.sh -i2c_find*** to verify communication.
7. Run ***solar_pi.sh -status*** to see the key information.
8. Run ***solar_pi.sh -help*** to see a full list of commands
9. The .zip file and unzipped folder are not used any more, feel free to delete them.
10. Samples scripts can be found in /opt/solar/scripts.

Software Un-Install

1. Run ***sudo /opt/solar/install.sh -uninstall***
2. If /etc/rc.local was modified to include start up scripts, these entries should be removed manually.

Note :

The default software install consists of the files stored inside /opt/solar folder as well as solar_pi.sh inside /usr/local/bin.

Selecting a Battery and Solar Panel

1. Solar panel voltage should be within the range specified in the datasheet, and at least 1V higher than the battery voltage.
2. The solar panels output power should be selected based on the on/sleep ratio of the system. Low power solar panels can still be used, it just means the system will spend more time in low power sleep mode.

3. Battery voltage should be within the range specified in the datasheet. The dc/dc is more efficient with a higher voltage battery, so if given a choice of batteries it's better to select a 12V over a 6V.
4. If battery charging is enabled, the battery chosen should be able so support at least 1A of charging current.

Setting Up The Battery Charger

1. With a text editor open the configuration file and update the FULL_BATTERY_V parameter. This voltage is used by the battery charger logic to know when to stop charging. Decimal places are allowed. ***sudo nano /opt/solar/solar_pi.conf*** is one way to do open the config file.
2. The battery charger is off by default, to turn it on run ***solar_pi.sh -charger_on***.
3. The charging current is automatically controlled depending on solar panel output power, battery voltage, and board temperature. It is limited to below 1A.
4. To extend battery life in demanding applications it's a good idea not to let the battery discharge to critically low levels and also not to charge it to full capacity. Keeping it between 30 and 80 % is a good rough estimate. To keep the battery from depleting too much put the system to sleep when the battery voltage starts to get low. Remember to make sure the charger is enabled before going to sleep.

Sleep Mode

1. Going to sleep turns off 5V power going to the Pi. If the battery charger is enabled, it will still charge the battery as long as the solar panel is producing some power.
2. There are 3 programmable wake up sources, 5V will be turned on when one or more are active.

   The wake up sources are as follows.

   a. solar panel voltage, when solar panel voltage reaches a certain level, turn on 5V

   b. battery voltage, when battery voltage is charged to a certain level, turn on 5V

   c. Real time clock, when time delay is up turn on 5V, time delay can be up to 1 year

Remote Operation

General recommendations for units operating out in the field

1. FULL_BATTERY_V variable inside the .conf file should be properly set to allow the battery charger to work.
2. Be sure to enable the battery charger on power up and before going to sleep.
3. Before going to sleep be sure to properly configure the wake up sources, otherwise the system may never wake up.
4. All the commands inside solar_pi.sh can be called from a user script, this allows relatively simple scripts to monitor battery/solar voltage and put the system to sleep if needed. For reference please see the contents of /opt/solar/scripts.

Choosing a FAN

1. The fan is powered from the battery, so the fan's maximum voltage needs to be taken into account. For example a 5V fan would be damaged if driven from a 12V battery.
2. The maximum current of the fan should not greatly exceed 1A.

Firmware Update

1. Download the new firmware from website
2. unzip it, ***sudo unzip <file_name>***
3. run ***solar_pi.sh -fw_update <file_name>***

Firmware Recovery

If something like a power loss during a firmware update "bricked" the solar power module, follow the steps below to recover it.

1. Disconnect all power sources, solar panel and battery.
2. Press and hold push button.
3. Connect a battery, and keep holding the push button until the power LEDs on the Pi come on, then release the push button. LEDs should come on no later than 15 seconds after battery is connected.
4. Wait until he Pi boots up, then run ***solar_pi.sh -i2c_find,*** to verify connection to the bootloader
5. update firmware with ***solar_pi.sh -fw_update <file_name>***

Push Button Hardware Reset

Pressing and holding the push button for around 15 seconds will power cycle the 5V. Hold the push button until the LEDs turn off, then release the push button

List of Commands

-help

Result : Prints a list of commands.

Example :
solar_pi.sh -help

-board_info

Result : Prints the PCB version, firmware version, software version, and I2C address.

Example :
solar_pi.sh -board info

-status

Result : Prints some general information such as voltages, currents and temparature

Example :
solar_pi.sh -status

-battery_v

Result : Prints the battery voltage

Example :
solar_pi.sh -battery_v

-solar_v

Result : Prints the solar panel voltage

Example :
solar_pi.sh -solar_v

-current_5v

Result : Prints the 5V current going to the PI in amps.

Example :
solar_pi.sh -current_5v

-current_solar

Result : Prints the solar panel current in amps.

Example :
solar_pi.sh -current_solar

-temp

Result : Returns the reading of the controller's internal temperature sensor in degrees Celsius. The accuracy of this reading can vary and shouldn't be relied upon to make important decisions.

Example :
solar_pi.sh -temp

-charger_<on/off>

Result : Enables or disables the battery charger.

Example :
solar_pi.sh -charger_off
solar_pi.sh -charger_on

-io_read_inputs

Result : Prints the 10 bit A/D(analog to digital) readings of all the I/O pins. The reference voltage for the A/D converter is 3.3V, so to convert the reading to volts the following formula can be used.
Volts = (AD_VAL * 3.3)/1023, For example if AD_VAL is 310, that equals to 1V.

Note 1 : The voltage range is limited to 3.3V, for higher voltages please place a resistor divider to bring down

the voltage.

Note 2 : When not connected to anything, inputs will tend to float. This is generally not a big deal, however best practice would be to configure all unused I/Os as outputs. Floating inputs may also increase sleep current.

Example :
solar_pi.sh -io_read_inputs

-io_out <io_num> <val>

Result : Drives one of the output pins high or low, val must be 0 or 1

Note: The pin must first be configured as an output, otherwise this will have no effect. To configure a pin as an output see -io_set_direction command.

Example : drive IO_1 high
solar_pi.sh -io_out 1 1

Example : drive IO_5 low
solar_pi.sh -io_out 5 0

-io_read_direction

Result : Shows whether the IO pins are configured as outputs or inputs, use -io_set_direction to change the direction of a pin .

Example :
solar_pi.sh -io_read_direction

-io_set_direction <io_num> <dir>

Result : Configures an IO pin as an input or output.
<io_num> must be {1,2,3,4,5} and <dir> bust be {0,1} where 0 equals out and 1 equals in.

Example : make IO_1 an output
solar_pi.sh -io_set_direction 1 0

Example : make IO_5 an input
solar_pi.sh -io_set_direction 5 1

-fan_speed <0/50/75/100>

Result : Sets up the fan speed.
Argument must be one of {0,50,75,100}
Note: The fan is driven by the battery, so the batteries voltage should not exceed the fan's voltage rating. For example driving a 5V fan by a 12V battery is not advised.

Example : turns off the fan
solar_pi.sh -fan_speed 0

Example : turn on the fan to 100%
solar_pi.sh -fan_speed 100

-sleep <sec>

Result : Solar panel module will wait <sec> seconds then turn off 5V. Maximum delay is 255 seconds.After calling this command the Rapsberry pi should shutdown(sudo poweroff) asap.

Note: Before going to sleep the wake up sources should be enabled/disable with the -set_wake_up_sources command.

Example : go to sleep in 30 seconds
solar_pi.sh -sleep 30

-power_cycle <sec>

Result : Solar panel module will wait <sec> seconds then turn off 5V, then wait a couple more seconds then turn 5V on again. Rapsberry pi should shutdown(sudo poweroff) asap after calling this command. Maximum delay <sec> is 255 seconds.

Example : power cycle in 30 seconds
solar_pi.sh - power_cycle 30

-read_wake_up_sources

Result : Shows which of the wake up sources are enabled.

Example :
solar_pi.sh -read_wake_up_sources

-set_wake_up_sources <solar_v> <bat_v> <rtc_sec>

Result : Enables or disables the wake up sources that could bring the system out of sleep mode. Decimals are allowed, 0 means disable this wake up source. This command should be called before calling the -sleep command.

<solar_v> is the solar panel voltage
<bat_v> is the battery voltage
<rtc_sec> is the time in seconds, up to 1 year

Example : wake up when the solar panel voltage is higher than 17.7V
solar_pi.sh -set_wake_up_sources 17.7 0 0

Example : wake up when the battery voltage is higher than 11.2V
solar_pi.sh -set_wake_up_sources 0 11.2 0

Example : wake up when either the solar panel voltage is higher than 15.1 or battery voltage is higher than 12.2 or 1 hour has passed
solar_pi.sh -set_wake_up_sources 15.1 12.2 3600

-rtc_status

Result : Shows the real time clock time, and alarm setting.
Note : Alarm is set by the -set_wake_up_sources command, and the time is set by the -rtc_set_time_from_pi command.

Example :
solar_pi.sh -rtc_status

-rtc_set_time_from_pi

Result : Copies Raspberry Pi's time to the real time clock.
Note: Real time clock time will be kept as long as there's power.

Example :
solar_pi.sh -rtc_set_time_from_pi

-rtc_copy_time_to_pi

Result : Copies the real time clock's time to the Pi. Useful in the field when the Pi wakes up from sleep and doesn't have a network to retrieve time from.

Example :
solar_pi.sh -rtc_copy_time_to_pi

-i2c_find

Result : Finds the solar power module on the I2C bus, and updates I2C_ADDR variable inside /opt/solar/solar_pi.conf. This command is normally only used once when setting up the system for the first time.

Example :
solar_pi.sh -i2c_find

-i2c_change_addr <new_addr>

Result : Changes the I2C address, and updates I2C_ADDR variable inside /opt/solar/solar_pi.conf. Changing the I2C address allows the solar power module to coexist with other daughter cards.

Example : Change I2C address to 50(Hex)
solar_pi.sh -i2c_change_addr 50

-sw_rev

Result : Reads the Linux software revision

Example :
solar_pi.sh -sw_rev

-fw_update <file_name>

Result : Updates the firmware.

Note: There is also a firmware recovery procedure described in other parts of the user manual. The recovery procedure should be used for “bricked” devices.

Example :
solar_pi.sh -fw_update file.bin

Document Revisions

Rev 1.2 :

- Added -current_solar command.

Rev 2.0 :

- Removed the -fw_update_recover command, -fw_update should be used instead
- Added -sw_rev command