Witty Pi Mini

Realtime Clock and Power Management for Raspberry Pi

User Manual (revision 1.02)

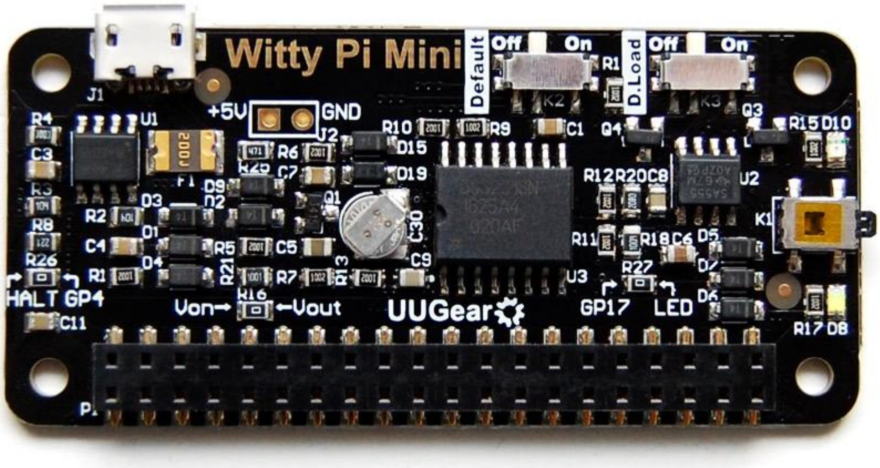


Table of Content

What is Witty Pi (Mini)?

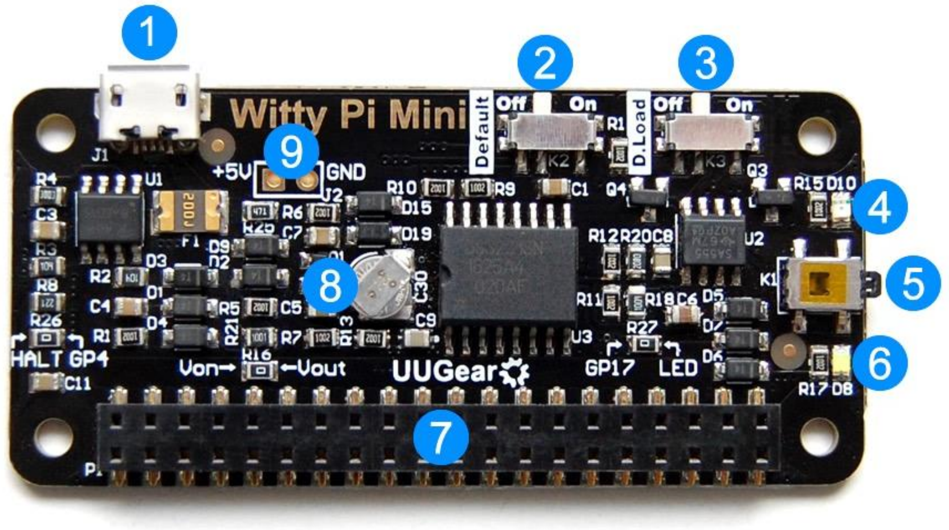
Witty Pi is small extension board that can add realtime clock and power management to your Raspberry Pi. After installing Witty Pi on your Raspberry Pi, you get some amazing new features:

- Gracefully turn on/off Raspberry Pi with single tap on the switch.
- Fully cuts power for Raspberry Pi and all its USB peripherals after shutdown.
- Raspberry Pi knows the correct time, even without accessing the Internet.
- You can schedule the startup/shutdown of your Raspberry Pi.
- You can even write a script to define complex ON/OFF sequence.
- When the OS loses response, you can long hold the switch to force power cut.

Witty Pi supports all Raspberry Pi models that has the 40-pin GPIO header, including A+, B+, 2B, Zero and 3B.

Witty Pi Mini is a new member in Witty Pi family. It has same features with Witty Pi Mini and is much smaller (pHAT shape). Comparing to Witty Pi Mini, Witty Pi Mini uses a super capacitor (0.08F, or 80000uF) for realtime clock off-power time keeping, and use two surface mounted switch to replace the red and blue jumpers in Witty Pi Mini. Thanks to the small dimension (65mm x 30mm x 4mm), Witty Pi Mini is very suitable to be used on Raspberry Pi Zero (V1.2, V1.3 or W), although it could also be used on other models.

The picture below shows how Witty Pi Mini looks like:



1) Micro USB connector as DC 5V power in

2) Configure if Raspberry Pi should be turned on when power connected

3) Configure if dummy pausing load should be turned on.

4) Red LED as power indicator, when Raspberry Pi is on.

5) Tact switch to turn on/off Raspberry Pi.

6) White LED as action indicator.

7) 2x20-pin header (connects to Raspberry Pi).

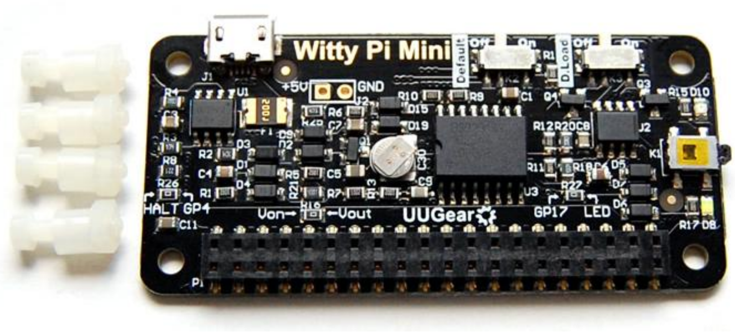
8) Super capacitor (0.08F, or 80000uF) for off-power time keeping.

9) Unpopulated connector as DC 5V alternative power input.

What is in the Package?

Each Witty Pi Mini package contains:

- Witty Pi Mini board x 1
- M2.5 x 10mm plastic screw x 4
- 4mm spacer x 4
- M2.5 plastic nut x 4


Witty Pi Mini
Default
Off
On
D.Load
+5V
GND
HALT
GP4
Von→
←Vout
UUGear
GP17
LED

Witty Pi Mini Specifications

Dimension:	65mm x 30mm x 4mm
Weight	9g (net weight)
Realtime Clock Chip	DS3231SN (datasheet)
LED Indicator	Red LED: lights up when Raspberry Pi gets powered. White LED: lights up for a few seconds when a shutdown command is received, or blinks when Witty Pi Mini is standing by.
Connector	20x2 surface-mounted header with 2.54 mm pitch and 2mm plastic height. Micro USB female connector as 5V DC input.
Time Keeping	Using 0.08F super capacitor as time keeping power source. Allows about 17 hours off-power time keeping.
Power In	DC 5V via micro USB, or via the alternative power input connector (unpopulated).
Output Current	Maximum 2A for Raspberry Pi and its peripherals
Standby Current	~ 5mA with dummy load turned off (default) ~7mA in average with dummy load turned on
Operating Temperature	-30℃~80℃ (-22°F~176°F)
Storage Temperature	-40℃~85℃ (-40°F~185°F)
Humidity	0~80%RH, no condensing

Before Installation - 1

If you have installed Raspbian Jessie by directly flash the OS image into SD card, you can skip reading this section.

You will need to installed the OS on the SD card first. **We recommend NOT to use NOOBS**, instead download the Raspbian image and directly flash it into your SD card (tutorial is here). The process could be faster and you will not have the problem caused by the NOOBS boot menu (read on for details).

The software of Witty Pi has been tested under **Raspbian Jessie and Raspbian Stretch**. Raspbian Wheezy should also work, but it is better to use Jessie or Stretch as it will be much easier to install Qt 5, which is required by the (optional) GUI of the software.

The software of Witty Pi might also work on other operating systems with or without modification. The software is written in BASH, so it could be easily modified by the customers to support other OS. However, at lest for now, only Raspbian Stretch, Jessie and Wheezy are officially tested. You should have mind preparation that tweaking might be needed when using Witty Pi Mini on other operating systems.

If you have installed the OS with NOOBS, please make sure to skip the boot menu. Otherwise the boot menu will postpone the booting long enough to make Witty Pi Mini think the system is down and then cut the power (so your Raspberry Pi will not boot).

Here is how to skip the NOOBS boot menu: you insert your SD card to your PC (use a SD card reader if your PC doesn’t have SD card slot), open the RECOVERY partition and create a text file named “autoboot.txt” there. Edit it and put this into the file:

```
boot_partition=6
```

It tells NOOBS to boot from the partition who is numbered 6. It is usually the case when you installed only one OS on the SD card (more details could be found here). If you have multiple OS installed, you can run “sudo fdisk –l” command to find the partition to boot with.

Remarks: there is a regression on NOOBS V2.2 and V2.3, so the trick above will not work on (at least) these two versions (details here). We hope it could be fixed soon, meanwhile you can use NOOBS V2.1 to avoid this problem (download from here), or even better, do not use NOOBS and directly flash the OS into the SD card.

Before Installation - 2

Before installing the software, please check whether you have 1-Wire interface enabled. This is important because 1-Wire interface uses GPIO-4 by default, which will conflict with Witty Pi. If you need to use 1-Wire interface, please assign it to a different GPIO pin.

You can assign different GPIO pin to 1-Wire interface in /boot/config.txt file, find the "dtoverlay=w1-gpio" text and replace it with:

```
dtoverlay=w1-gpio,gpiopin=18
```

If you don't need 1-Wire interface for now, you can disable it:

```
#dtoverlay=w1-gpio
```

If you have 1-Wire interface enabled on GPIO-4 and installed Witty Pi's software, you may not be able to login to your Raspberry Pi because it always shuts itself down before you get the chance to login. To solve this problem, you will need to take out the micro-SD card on your Raspberry Pi, and access its file system via a card reader. You need to edit the config.txt file (as mentioned above) in the "boot" volume to change the GPIO pin used by 1-Wire interface, or you can disable 1-Wire interface if you don't need it for now. After saving the file and put the micro SD card back to Raspberry Pi, you should be able to boot it again.

Software Installation

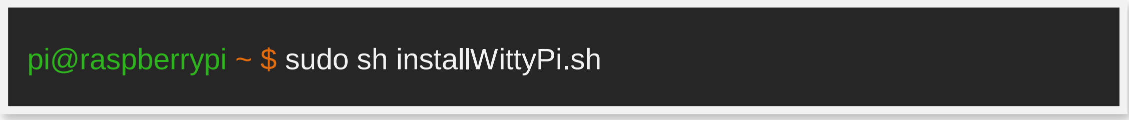
We strongly recommend to install the software for Witty Pi Mini **BEFORE** physically mount Witty Pi Mini on your Raspberry Pi.

Witty Pi Mini uses the same software with Witty Pi 2. You will need to have your Raspberry Pi connected to the Internet. The installation will be very simple if you run our installing script. The wiringPi utility is required by the software so the script will install it for you, if you don't have it installed yet.

First step is to run this command in your home directory:

```
pi@raspberrypi ~ $ wget http://www.uugear.com/repo/WittyPi2/installWittyPi.sh
```

If your Raspberry Pi has internet connection, it will immediately download the script from our website, and you will then see the "installWittyPi.sh" script in your home directory. Then you just need to run it with sudo:



```
pi@raspberrypi ~ $ sudo sh installWittyPi.sh
```

Please notice that **sudo** is necessary to run this script. This script will automatically do these tasks in sequence:

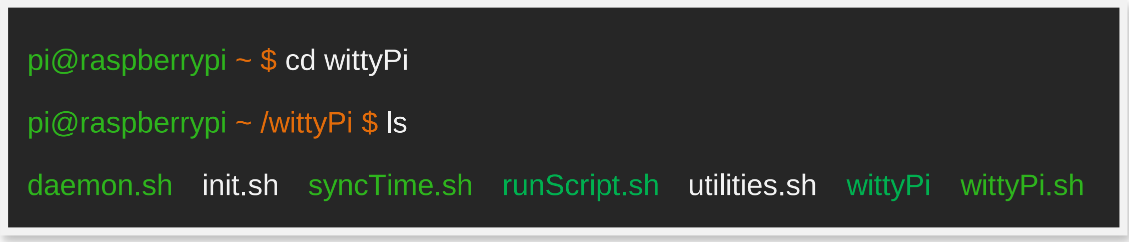
1. Enable I2C on your Raspberry Pi
2. Install i2c-tools, if it is not installed yet
3. Configure Bluetooth to use mini-UART (Raspberry Pi 3 only)
4. Install wiringPi, if it is not installed yet
5. Install Witty Pi programs, if they are not installed yet
6. Remove fake-hwclock and disable ntpd daemon
7. Install Qt 5, if it is not installed yet (it is optional, and is for Jessie only)

The fake-hwclock package is no longer needed as you have a real hardware clock now. Removing fake-hwclock package can avoid some potential problems (e.g. this one). Also the ntpd daemon should be disabled to avoid corrupting the RTC time. Disabling ntpd will not affect the NTP time synchronization as Witty Pi software will explicitly query NTP time and then update to system and RTC.

The Qt 5 installation is for the GUI only. If you don't plan to use it, you can skip this step for now, as Qt 5 installation will take a while.

You can also manually install these packages and make those configurations, if you prefer to. After the installation, please remember to **reboot your Raspberry Pi**, so the Realtime clock I2C hardware will be loaded correctly.

You will see a new "wittyPi" directory, and it contains 5 runnable files:



```
pi@raspberrypi ~ $ cd wittyPi
pi@raspberrypi ~ /wittyPi $ ls
daemon.sh  init.sh  syncTime.sh  runScript.sh  utilities.sh  wittyPi  wittyPi.sh
```

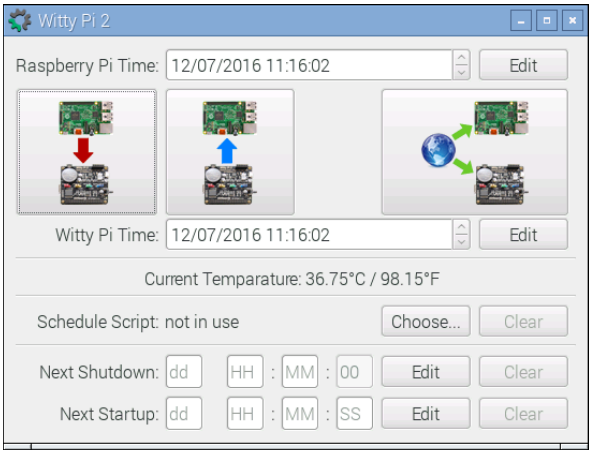
Although the **daemon.sh** is runnable, you should not run it manually. The installing script has registered it into /etc/init.d and it will run automatically after the start up.

The **syncTime.sh** script is not suppose to be manually run either, it will run automatically at about one minute after the start up. It will copy the time from Raspberry Pi system to RTC if you have never set the RTC time before. If RTC has correct time and your Raspberry Pi hasn't (because of the lacking of Internet), it will copy the RTC time to your Raspberry Pi system.

The **runScript.sh** script is the one who takes charge of the schedule script running. Usually you don't need to run it manually, as it will be executed after the system is up. If there is a schedule script in use, it will schedule the next shutdown and next startup, according to the schedule script.

The **wittyPi.sh** is the software that allows you to configure your Witty Pi interactively. You can use it to copy time between Realtime clock and the system, and schedule the time for auto shutdown and/or startup. Please see the "Software Usage" chapter for more information.

The **wittyPi** is the GUI version of wittyPi.sh, it requires desktop environment and Qt 5 to run.



Raspberry Pi Time: 12/07/2016 11:16:02

Witty Pi Time: 12/07/2016 11:16:02

Current Temparature: 36.75°C / 98.15°F

Schedule Script: not in use

Next Shutdown: dd HH : MM : 00

Next Startup: dd HH : MM : SS

Now the software has been installed, and you will need to physically mount Witty Pi Mini on your Raspberry Pi.

Software Update/Uninstallation

If you want to update the software to newer version, you don't have to uninstall it first. Just remove or rename your "wittyPi" directory and repeat the installing process, then you are all set.

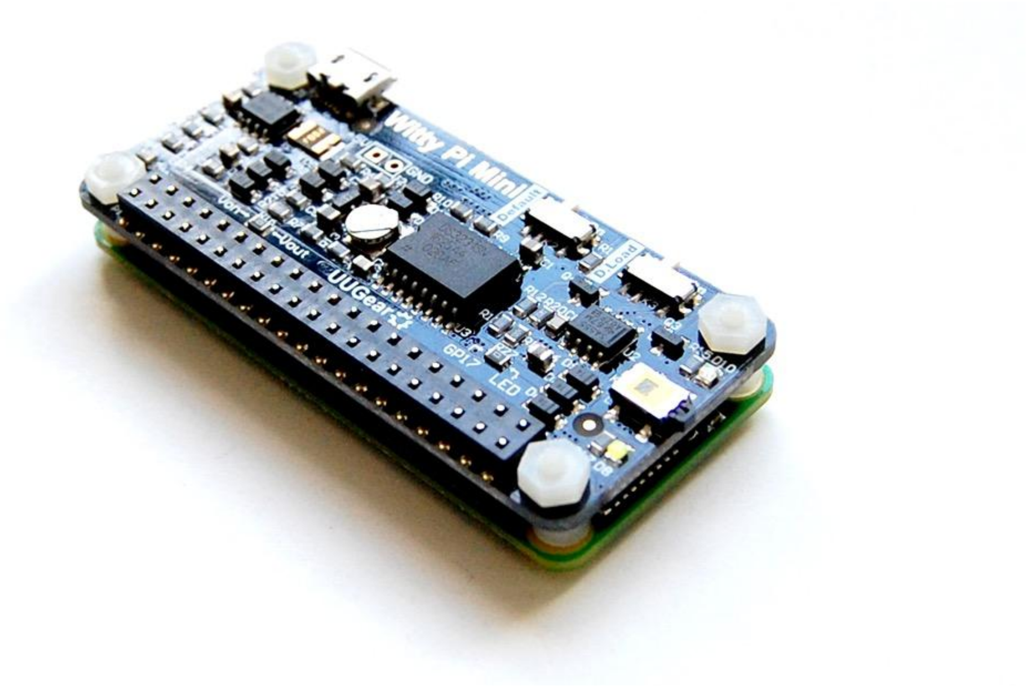
```
pi@raspberrypi ~ $ mv wittyPi wittyPi.bak

pi@raspberrypi ~ $ wget http://www.uugear.com/repo/WittyPi2/installWittyPi.sh

pi@raspberrypi ~ $ sudo sh installWittyPi.sh
```

If you prefer to completely remove the software, besides removing the “wittyPi” directory, you should also remove the “/etc/init.d/wittypi” file. There are some dependencies (such as wiringPi, i2c-tools etc.), who are installed during the software installation. In the major of cases you don’t have to remove them, but if you wish, you can check the content of “installWittyPi.sh” script and do the reverse.

Mounting Witty Pi Mini on Raspberry Pi Zero

If you want to use Witty Pi Mini on Raspberry Pi Zero (V1.2, V1.3 or W), you will need to solder the 20x2 pin male header on it first, or you can buy one with the header soldered already.

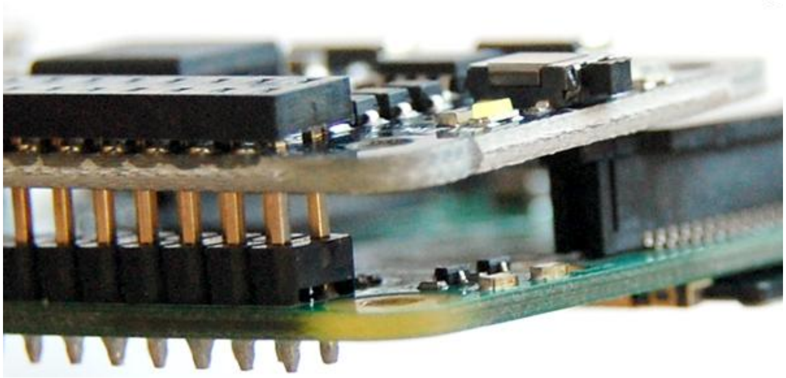
You can simply mount Witty Pi Mini on your Raspberry P Zero’s 40-pin header, and it can work just like that. However, if you wish, you can use the plastic screws, spacers and nuts in the package to tightly mount Witty Pi Mini on your Raspberry Pi Zero. Please make sure NOT to put Witty Pi Mini up side down, the 40 pins should connect to the female header via the holes underneath, as shown in figure below.



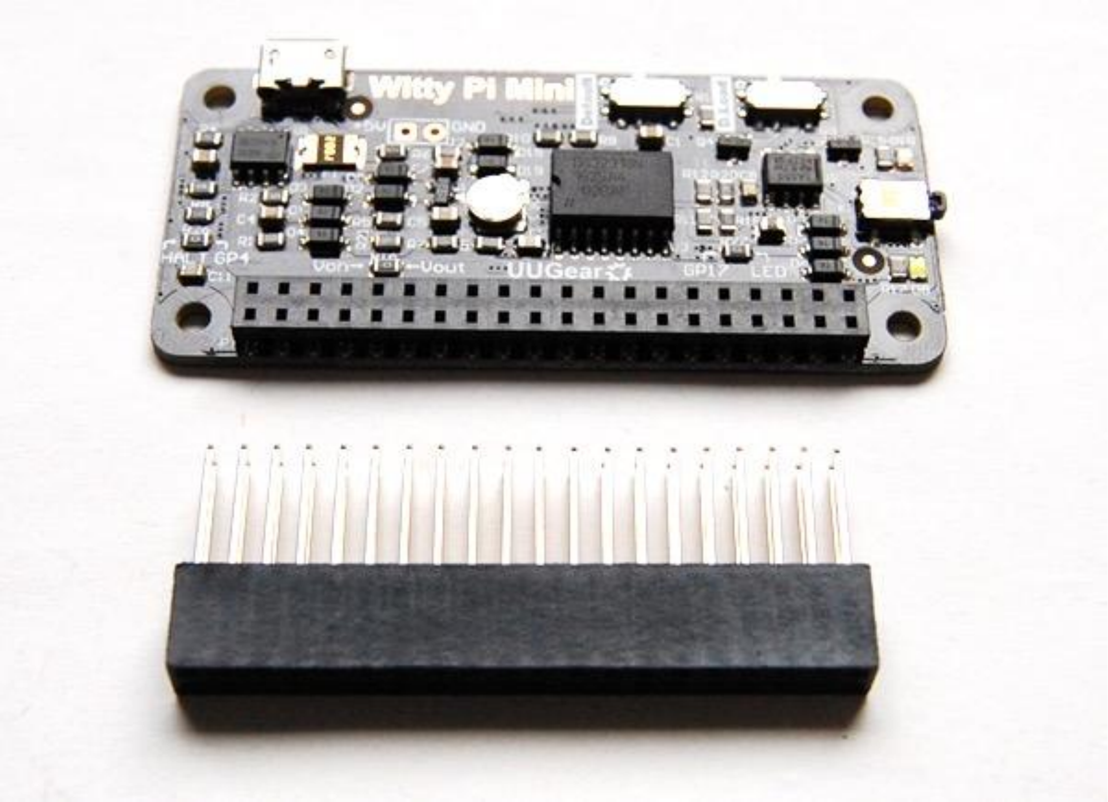
You will need to connect the power supply to the micro USB connector on Witty Pi Mini to make it work. Or you can power it via the (unpopulated) alternative power input connector.

Mounting Witty Pi Mini on Other Raspberry Pi Models

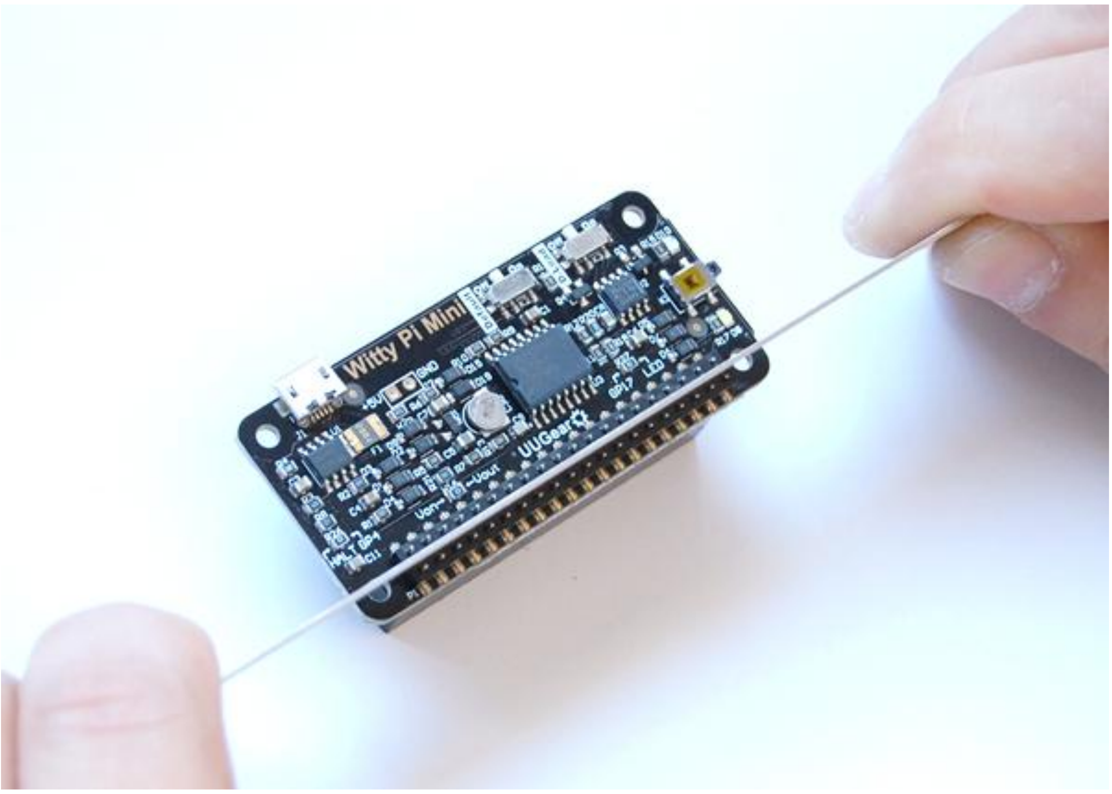
Other Raspberry Pi models (A+, B+, 2B and 3B) have the 40 pin header soldered already, so it is possible to directly mount Witty Pi Mini over them. However, because of the existence of display connector, you will not be able to firmly connect them.



In order to make reliable connection, we suggest to use a stacking pin header (not included in the package).



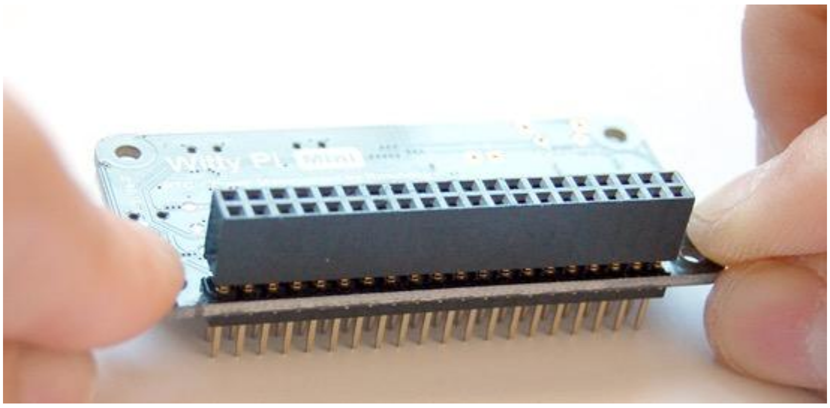
The 40 extra long pins on stacking header should reach the female header via the holes on the back of Witty Pi Mini. You can then place a steel ruler (or something similar) between the two rows of pins and push Witty Pi Mini until it reaches the plastic of the stacking header, as shown in figure below:



After mounting Witty Pi Mini on your Raspberry Pi, and connect the power supply to the micro USB connector on Witty Pi Mini, you can see **the white LED blinking, which means it is standing by**.

Now your Witty Pi Mini is ready to go.

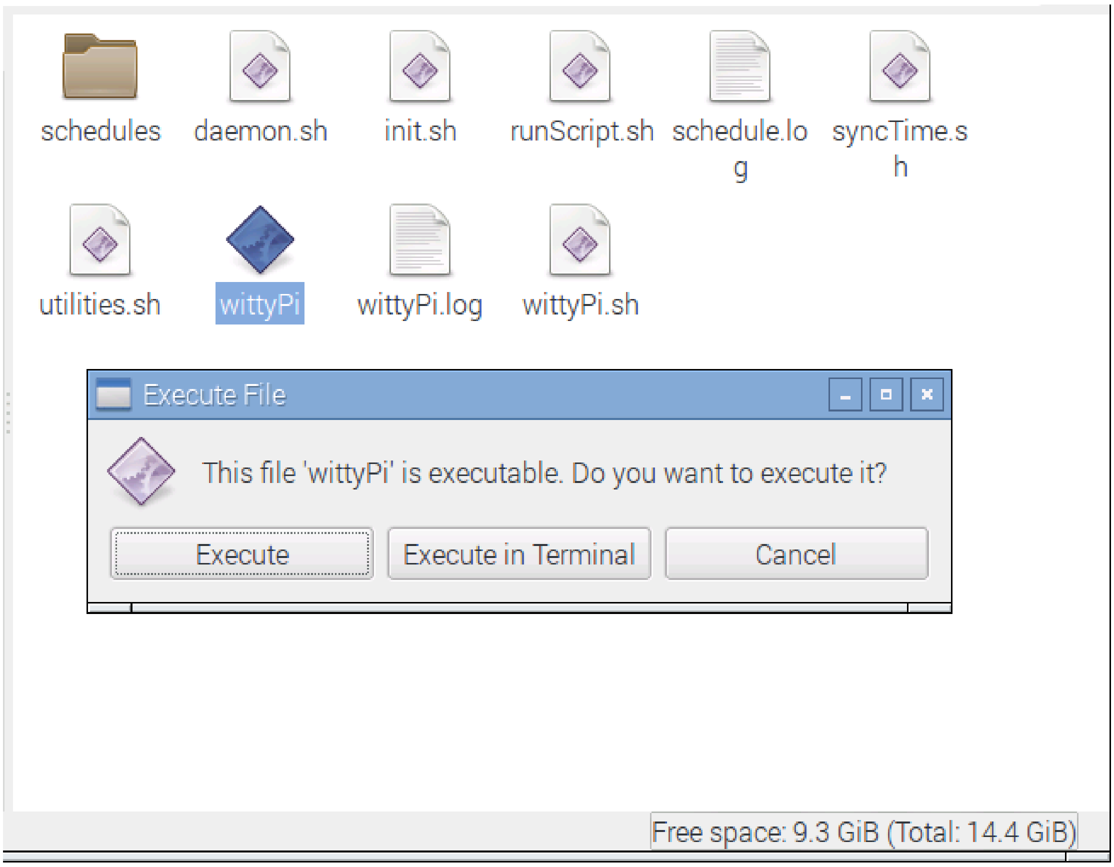
If you want to take off the stacking header from Witty Pi Mini, the proper way is to put your Witty Pi Mini upside down, let the 40 pins contact a flat surface, hold the two edges and press down. That way the stacking header will be taken off safely. See the figure below:



Software Usage

You can either run the wittyPi.sh or wittyPi (GUI executable), and they have (almost) the same functionality.

If you are running desktop environment, you can go to the “wittyPi” folder in your home directory, and double-click the wittyPi file. When you are asked whether to execute it, choose “Execute” to continue.



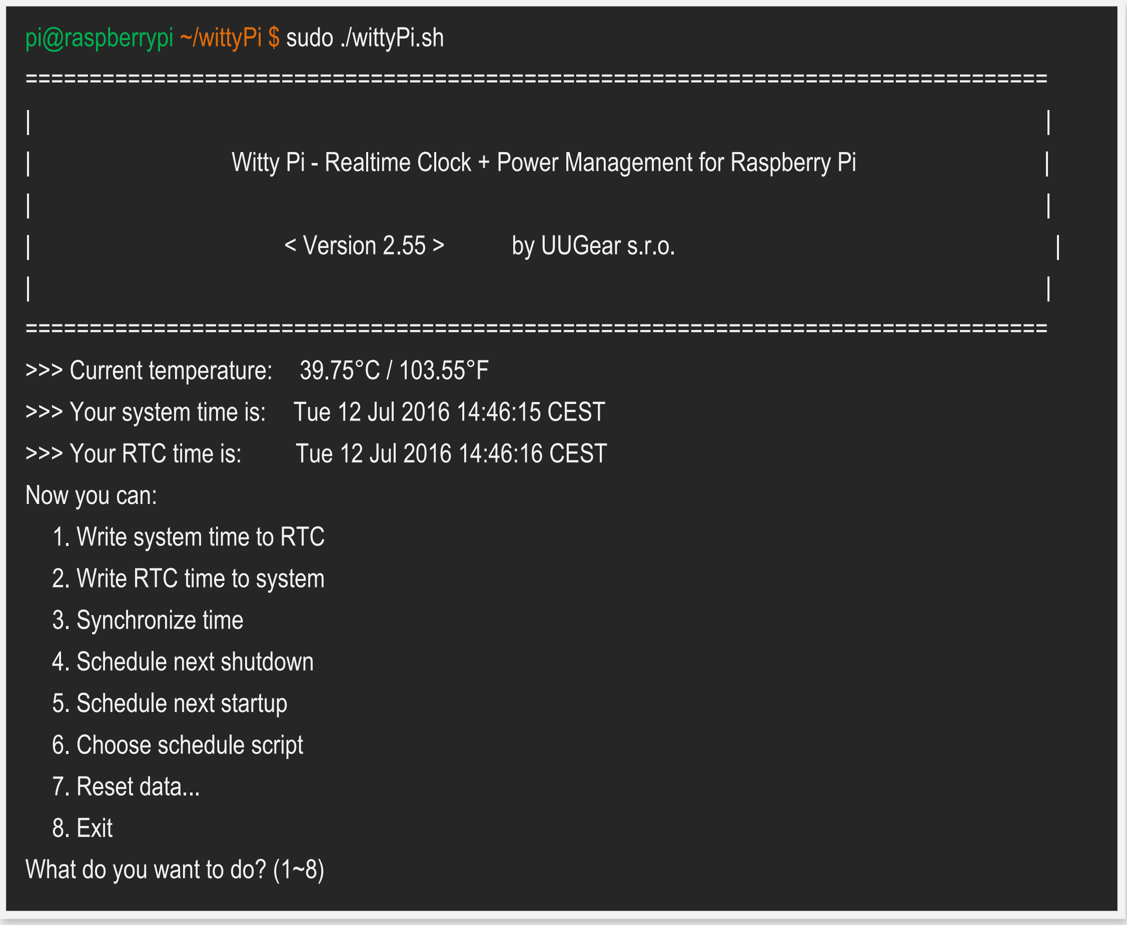
This file 'wittyPi' is executable. Do you want to execute it?

Free space: 9.3 GiB (Total: 14.4 GiB)

The wittyPi.sh is a bash script, and you can run it with:

```
pi@raspberrypi ~/wittyPi $ sudo ./wittyPi.sh
```

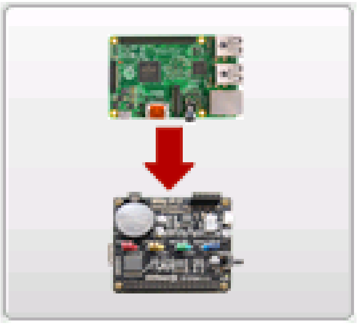
Please notice that sudo is required. Once the script is run, it will tell you the system time and RTC (Realtime clock) time, so you can decide how to copy the time. It also tells you the current temperature around your Raspberry Pi.



```
pi@raspberrypi ~/wittyPi $ sudo ./wittyPi.sh
================================================================================
|                                                                              |
|              Witty Pi - Realtime Clock + Power Management for Raspberry Pi   |
|                                                                              |
|                      < Version 2.55 >      by UUGear s.r.o.                  |
|                                                                              |
================================================================================
>>> Current temperature:   39.75°C / 103.55°F
>>> Your system time is:   Tue 12 Jul 2016 14:46:15 CEST
>>> Your RTC time is:      Tue 12 Jul 2016 14:46:16 CEST
Now you can:
  1. Write system time to RTC
  2. Write RTC time to system
  3. Synchronize time
  4. Schedule next shutdown
  5. Schedule next startup
  6. Choose schedule script
  7. Reset data...
  8. Exit
What do you want to do? (1~8)
```

The program gives you 8 options, and you can input the number and press ENTER to confirm.

1. Write system time to RTC



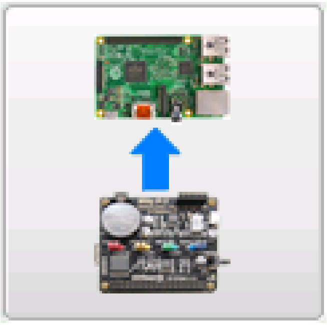
This option will copy the time from your Raspberry Pi system to the Realtime clock on Witty Pi. This option should be used when you find the system time is correct (may get synchronized from Internet) and RTC time is not.

If you are running the GUI, you can click the button shown on the right to finish the same task.

2. Write RTC time to system

This option will copy the time from the Realtime clock on Witty Pi Mini to your Raspberry Pi system. This option should be used when you find the RTC time is correct while the system time is not.

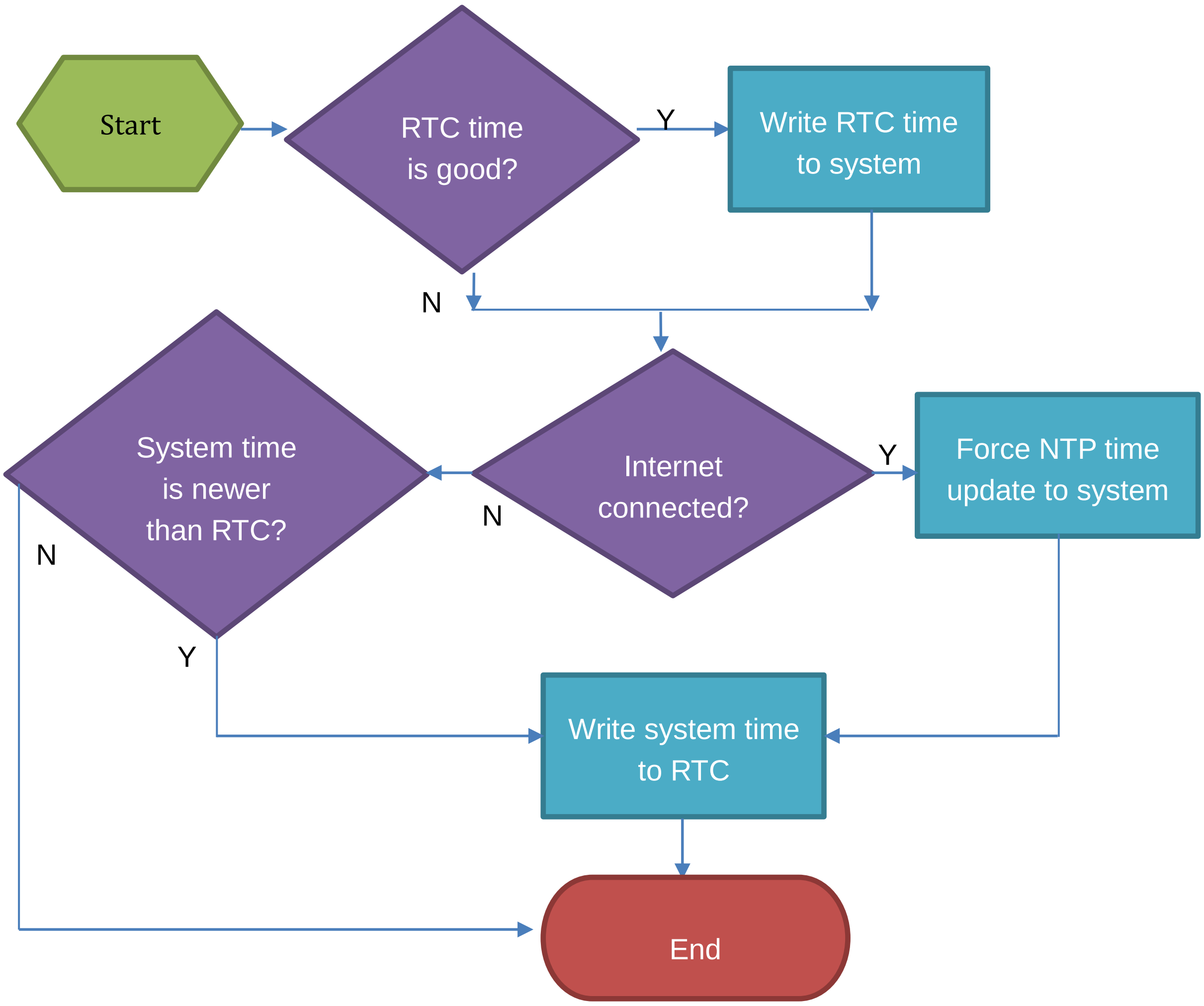
If you are running the GUI, you can click the button shown on the right to finish the same task.



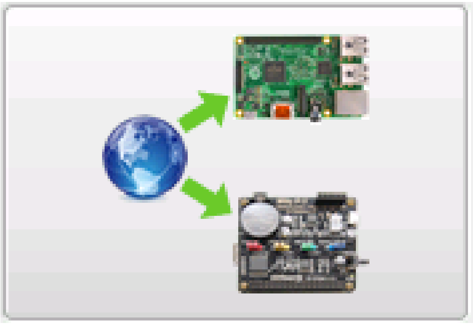
3. *Synchronize time*

If you choose this option, it will run the “syncTime.sh” script explicitly, which should have been executed once after the system is up.

This script will detect if Internet is connected, and apply NTP time to both system and RTC. The flow chart below shows what this script actually do:



If you are running the GUI, there is no such a button to do exactly the same. But if you want to write NTP time to both system and RTC, you can use this button:



4. *Schedule next shutdown*

This option allows you to specify when your Raspberry Pi should shutdown automatically.

Please notice the input format should be “DD HH:MM”. DD means the day in the month, HH is the hour, MM is the minute. All these should be 2 digits and 24-hour system is used. Here you can not specify the second. This is a hardware limitation on the RTC chip, and only day, hour and minute could be specified for scheduled shutdown.

You can use “??” as wildcard, which gives you the possibility to make a repeatable schedule. Please see the table below:

Repeatable Shutdown Schedule			
Day (dd)	Hour (HH)	Minute (MM)	Result
??	??	??	Minutely Schedule **(DON’T USE IT!)**
??	??	Number	Hourly Schedule
??	Number	Number	Daily Schedule

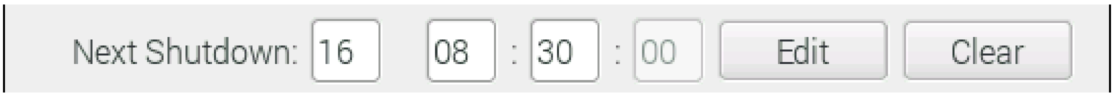
Please don’t use “?? ??:??” to schedule the next shutdown, or your Raspberry Pi will keep being shutdown and you hardly have a chance to change this setting (unless you remove the battery and force RTC to forget it).

According to the hardware limitation, not all patterns with wildcards are supported. The rule is: wildcards have to show up from let to right, and there is no number between two wildcards. So “?? ??:38” is OK, while “?? 16:??” is not supported.

Here are some examples of scheduling the shutdown:

- **15 21:45** means 9:45 at night, on 15th in this month.
- **?? 23:30** means 23:30 at night everyday (daily schedule)
- **?? ??:15** means the 15th minute every hour (hourly schedule)

If you are running the GUI, you can specify the next shutdown time by clicking the “Edit” button in the row:



Next Shutdown: 16 08 : 30 : 00

5. Schedule next startup

This option allows you to specify when your Raspberry Pi should startup automatically.

Please notice the input format should be “DD HH:MM:SS”, DD means the day in the month, HH is the hour, MM is the minute and SS is the second. All these should be 2 digits and 24-hour system is used. Different than the shutdown scheduling, you can specify the second here.

You can also use “??” as wildcard, which gives you the possibility to make a repeatable schedule. Please see the table below:

Repeatable Startup Schedule			
Day (dd)	Hour (HH)	Minute (MM)	Result
??	??	??	Minutely Schedule
??	??	Number	Hourly Schedule
??	Number	Number	Daily Schedule

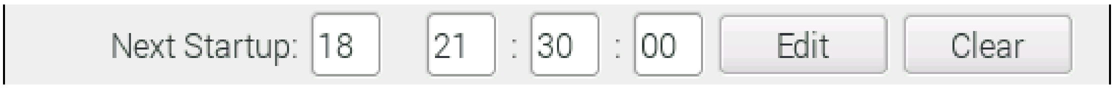
According to the hardware limitation, not all patterns with wildcards are supported. The rule is: wildcards have to show up from let to right, and there is no number between two wildcards. So “?? ??:??:12” is OK, while “?? 15:??:25” is not supported.

If you input an unsupported pattern, Witty Pi Mini will try to change it to the closest one that could be supported. You will see the message on the console.

Here are some examples of scheduling the startup:

- **15 07:30:00** means 7:30 in the morning, on 15th in this month.
- **?? 23:30:00** means 23:30:00 at night everyday (daily schedule)
- **?? ??:15:00** means the 15th minute every hour (hourly schedule)
- **?? ??:??:05** means the 5th second every minute (minutely schedule)

If you are running the GUI, you can specify the next startup time by clicking the “Edit” button in the row:


Next Startup: 18 21 : 30 : 00

6. Choose Schedule Script

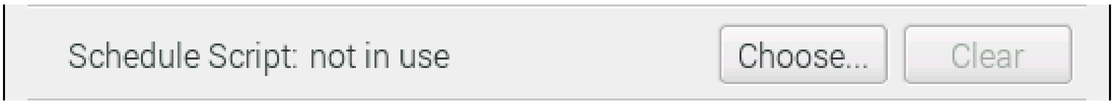
What if you want to define a complex ON/OFF sequence for your Raspberry Pi? The answer is “schedule script”.

A schedule script (.wpi file) defines a loop, with all states and their durations inside. By automatically running “runScript.sh” after booting, Witty Pi Mini will automatically schedule the next shutdown and next startup for you, and hence a complex ON/OFF sequence could be achieved.

After you select the “Choose schedule script” option, it will list all schedule scripts in the “schedules” folder. You can choose one, and then Witty Pi Mini will take care of the rest.

If you want to confirm what the script is doing, you can check the “schedule.log” file in the “~/wittyPi” directory, when your Raspberry Pi is on.

If you are running the GUI, you can click the “Choose…” button and select the schedule script you need.


Schedule Script: not in use

After selecting the schedule script, it will be displayed as “in use”, and you can hover your mouse cursor on it to see the content of current schedule script. You may also click the “Clear” button to terminate the usage of schedule script.

If you want to create your own schedule script, please read the “Making Schedule Script” chapter.

7. Reset Data…

If you want to erase the data you already set into Witty Pi Mini (scheduled startup time, scheduled shutdown time, currently used schedule script), you can choose this option.

Once you select this option, the software will display a sub menu, which allows you to:

- Clear auto startup time:
  The auto-startup time will be erased and Witty Pi Mini will not auto-start your Raspberry Pi.

- Clear auto shutdown time:
  The auto-shutdown time will be erased and Witty Pi Mini will not auto-shutdown your Raspberry Pi.
- Stop using schedule script:
  The “schedule.wpi” file will be removed.
- Perform all actions above:
  Clear all scheduled times and remove the “schedule.wpi” file.

If you are running the GUI, there is no such a button to clear all those data. However you can clear them one by one, by clicking the “Clear” button in specific row.

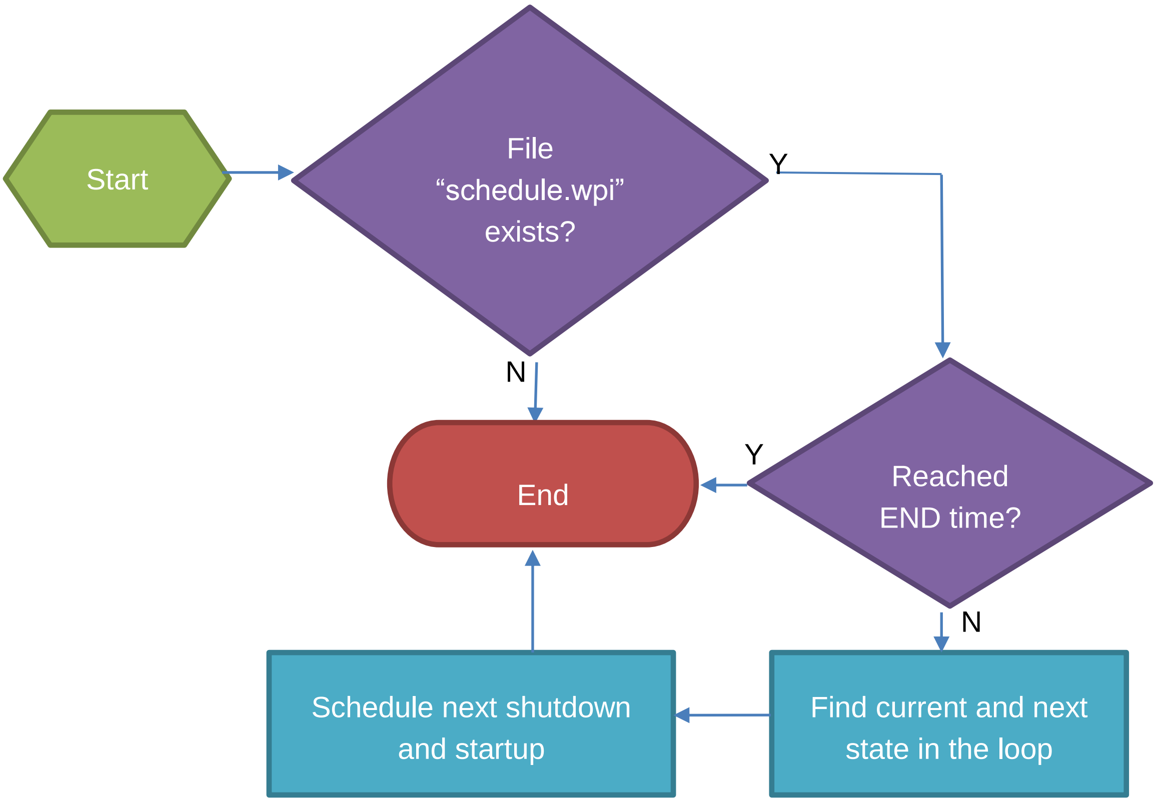
8. Exit

Selecting this option will simply exit the software and return to the console.

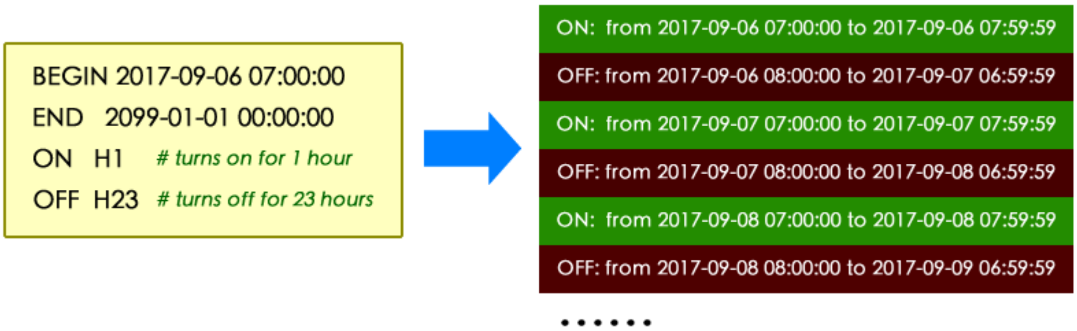
If you are running the GUI, just close the GUI window to exit.

How Schedule Script Works?

A schedule script defines a serial of ON/OFF states and specify the duration of each state. At the end of each state, there should be a scheduled shutdown (for ON state) or startup (for OFF state).
All states in the schedule script will be executed in sequence and repeatedly, until the END time is reached.


Start
File “schedule.wpi” exists?
Y
N
End
Y
Reached END time?
N
Schedule next shutdown and startup
Find current and next state in the loop

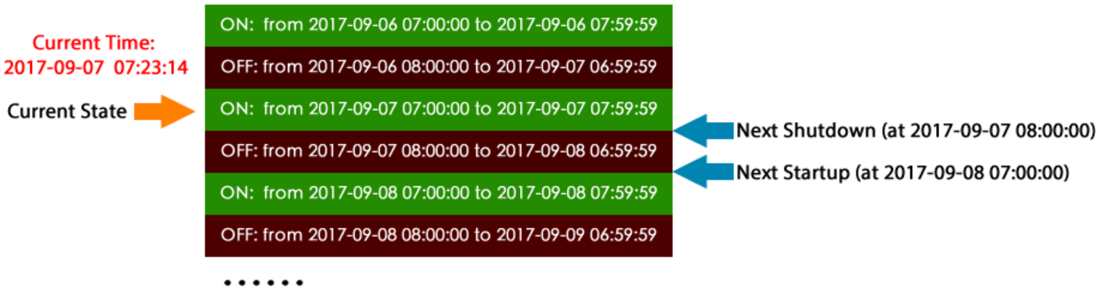
Every time your Raspberry Pi wakes up, it has a chance to run the runScript.sh file, which loads the schedule script (“schedule.wpi” file). If the current time doesn’t reach the END time defined by the schedule script, the next shutdown and next startup will be scheduled automatically. When your Raspberry Pi is wakened at scheduled startup time, it will repeat this process and schedule the next shutdown and startup. Although a schedule script only defines a few ON/OFF states, they could become many ON/OFF states in reality.



```
BEGIN 2017-09-06 07:00:00
END   2099-01-01 00:00:00
ON    H1     # turns on for 1 hour
OFF   H23    # turns off for 23 hours
```

ON: from 2017-09-06 07:00:00 to 2017-09-06 07:59:59
OFF: from 2017-09-06 08:00:00 to 2017-09-07 06:59:59
ON: from 2017-09-07 07:00:00 to 2017-09-07 07:59:59
OFF: from 2017-09-07 08:00:00 to 2017-09-08 06:59:59
ON: from 2017-09-08 07:00:00 to 2017-09-08 07:59:59
OFF: from 2017-09-08 08:00:00 to 2017-09-09 06:59:59
……

Please keep in mind that, running the same schedule script at different moment may get different result, as the “runScript.sh” will search and find the proper state according to current time.



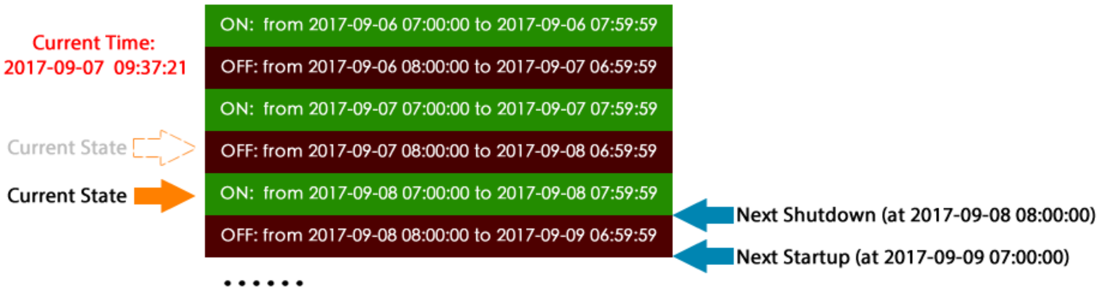
Current Time:
2017-09-07 07:23:14

Current State

ON: from 2017-09-06 07:00:00 to 2017-09-06 07:59:59
OFF: from 2017-09-06 08:00:00 to 2017-09-07 06:59:59
ON: from 2017-09-07 07:00:00 to 2017-09-07 07:59:59
OFF: from 2017-09-07 08:00:00 to 2017-09-08 06:59:59
ON: from 2017-09-08 07:00:00 to 2017-09-08 07:59:59
OFF: from 2017-09-08 08:00:00 to 2017-09-09 06:59:59
……

Next Shutdown (at 2017-09-07 08:00:00)
Next Startup (at 2017-09-08 07:00:00)

When the “runScript.sh” is executed, if the current time is located at an “OFF” state instead, it will take the next “ON” state as the current state, as it knows Raspberry Pi is currently ON.



Current Time:
2017-09-07 09:37:21

Current State

Current State

ON: from 2017-09-06 07:00:00 to 2017-09-06 07:59:59
OFF: from 2017-09-06 08:00:00 to 2017-09-07 06:59:59
ON: from 2017-09-07 07:00:00 to 2017-09-07 07:59:59
OFF: from 2017-09-07 08:00:00 to 2017-09-08 06:59:59
ON: from 2017-09-08 07:00:00 to 2017-09-08 07:59:59
OFF: from 2017-09-08 08:00:00 to 2017-09-09 06:59:59
……

Next Shutdown (at 2017-09-08 08:00:00)
Next Startup (at 2017-09-09 07:00:00)

Make Schedule Script

A schedule script is a text file with. wpi file extension. You can use any text editor to create and edit it. In the major of cases, using “nano” will be very convenient.

Below is a very simple schedule script and it will keep your Raspberry Pi on for 5 minutes in every 20 minutes.

```
# Turn on Raspberry Pi for 5 minutes, in every 20 minutes
BEGIN   2015-08-01 00:00:00
END     2025-07-31 23:59:59
ON      M5      # keep ON state for 5 minutes
OFF     M15     # keep OFF state for 15 minutes
```

Like many other scripting languages, Witty Pi’s schedule script also uses “#” to make single line comment.

The first two lines define the time range for executing the script. Please make sure to use the correct time format (YYYY-mm-DD HH:MM:SS). You can use one or more white characters (space or tab) between BEGIN/END and the time string.

The rest of the script defines the states in the loop. It could be “ON” or “OFF”, and you should define at least one “ON” and one “OFF” states in the loop. The ON and OFF states are used in pair.

You should also specify the duration of each state. You can do so by putting one or more parameters after ON/OFF text, separated by space or tab. Each parameter starts with a capital letter and follows by a number, where the capital letter is the unit of time:

- D = Days (D2 means 2 days)
- H=Hours (H3 means 3 hours)
- M=Minutes (M15 means 15 minutes)
- S=Seconds (S30 means 30 seconds)

For example, if you wish to define an ON state for one and a half hour, you can write:

```
ON      H1  M30
```

When the script engine executes this line, it will actually schedule a shutdown at the end of the ON state.

If you wish to define an OFF state for two days, you can write:

OFF D2

When this line gets executed, a startup will be scheduled at the end of the OFF state.

Sometimes you may want to skip certain scheduling of shutdown/startup, and let your own program to do the job. This can be achieved by using the WAIT syntax. For example:

ON M15 WAIT

This will keep your Raspberry Pi ON and no shutdown will be scheduled after 15 minutes, because there is a WAIT at the end of the line. The parameter M15 is here only to make sure the next OFF state can be calculated correctly and next shutdown can be scheduled properly. Once you use WAIT in the ON state, you are responsible for the shutdown of your Raspberry Pi. Also if you use WAIT in the OFF state, you will need to turn on your Raspberry Pi (manually or via external electronic switch).

After installing the software, there are some schedule scripts in the “schedules” directory, and they all have comments inside to explain themselves. You can take them as example to learn how to create the Witty Pi schedule script.

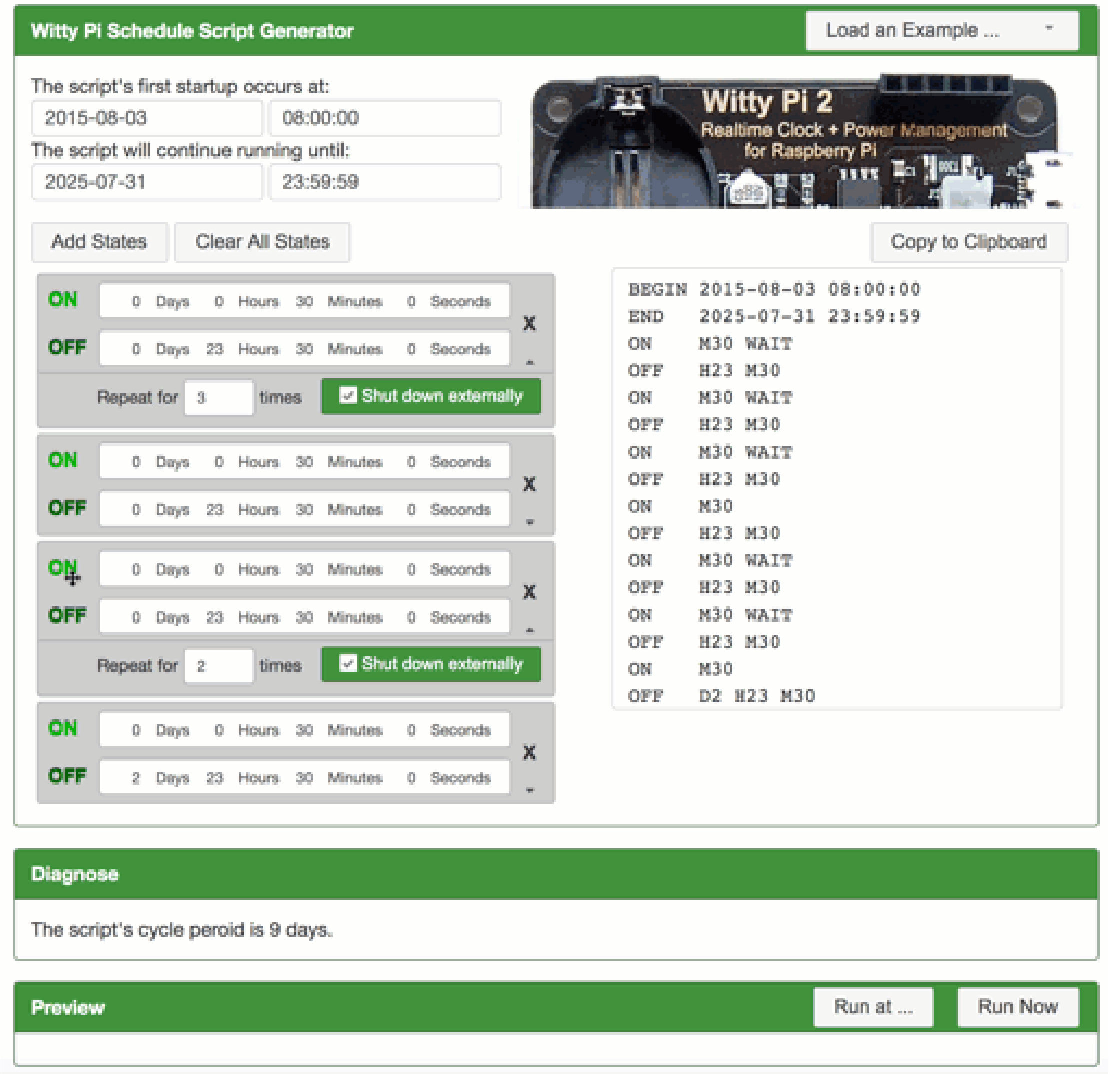
Using Schedule Script Generator

You can use our web application to create your schedule script. Just simply open this URL in your web browser and you are ready to go:

http://www.uugear.com/app/wittypi-scriptgen/

This web application allows you to visually create the schedule script, it immediately generate the final schedule script (on the right).

You can also click the “Run” button to preview how the schedule script will work. Alternatively, you can click the “Run at…” button and specify the moment to run the script. With this preview engine, you can verify your schedule script without waiting for hours or even days to see how your Raspberry Pi will be turned on and off.



The script's first startup occurs at: 2015-08-03 08:00:00

The script will continue running until: 2025-07-31 23:59:59

ON 0 Days 0 Hours 30 Minutes 0 Seconds
OFF 0 Days 23 Hours 30 Minutes 0 Seconds
Repeat for 3 times ☑ Shut down externally

ON 0 Days 0 Hours 30 Minutes 0 Seconds
OFF 0 Days 23 Hours 30 Minutes 0 Seconds

ON 0 Days 0 Hours 30 Minutes 0 Seconds
OFF 0 Days 23 Hours 30 Minutes 0 Seconds
Repeat for 2 times ☑ Shut down externally

ON 0 Days 0 Hours 30 Minutes 0 Seconds
OFF 2 Days 23 Hours 30 Minutes 0 Seconds

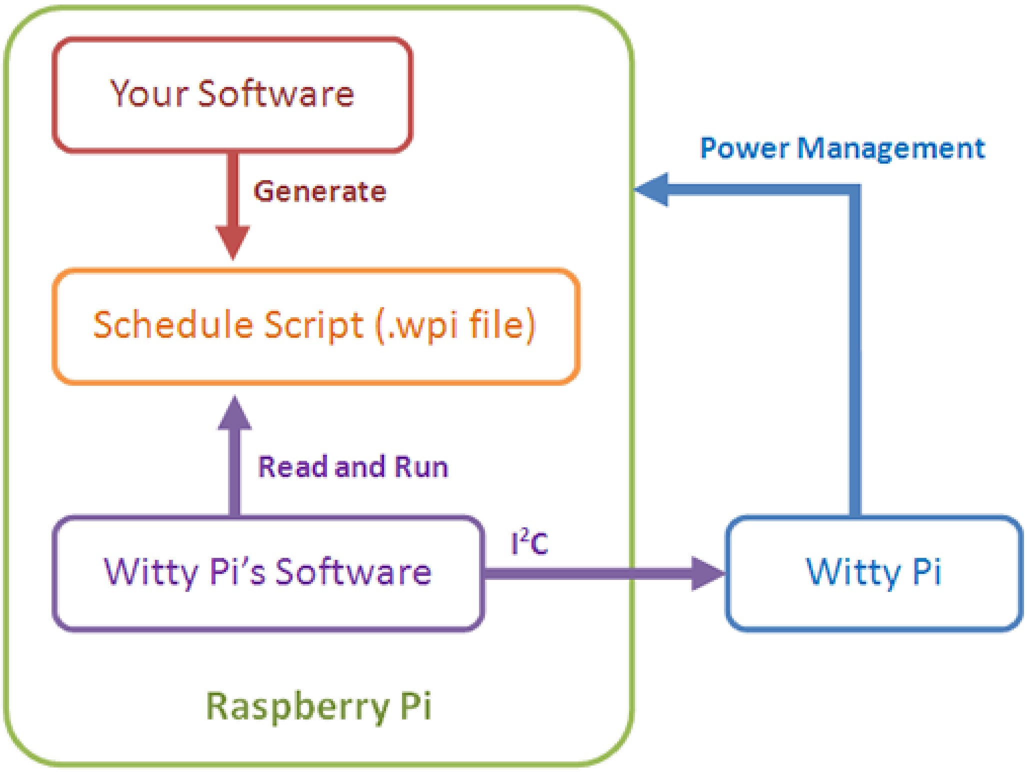
```
BEGIN 2015-08-03 08:00:00
END   2025-07-31 23:59:59
ON    M30 WAIT
OFF   H23 M30
ON    M30 WAIT
OFF   H23 M30
ON    M30 WAIT
OFF   H23 M30
ON    M30
OFF   H23 M30
ON    M30 WAIT
OFF   H23 M30
ON    M30 WAIT
OFF   H23 M30
ON    M30
OFF   D2 H23 M30
```

The script's cycle peroid is 9 days.

Although the schedule script doesn’t support looping, this web application allows you to repeat the specific ON/OFF states for certain times, and it will generate the schedule script with those ON/OFF states duplicated for certain times.

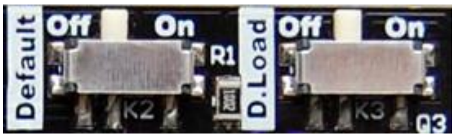
Advanced Usage of Schedule Script

Although the schedule script can be chosen by wittyPi.sh, you can use it without the help from wittyPi.sh. Just copy the schedule script file to “~/wittyPi/schedule.wpi” and then run “sudo ./runScript.sh” in the “~/wittyPi” directory, the script will start to work. This allows you to use schedule script as an interface, to integrate other tools with Witty Pi Mini together. For example, you can create your own tool to visually create a schedule script, or remotely generate the schedule script via a web interface.



Hardware Configuration and Hackings

There are two small switches on the board, and they allow you to make some customization on your Witty Pi Mini.



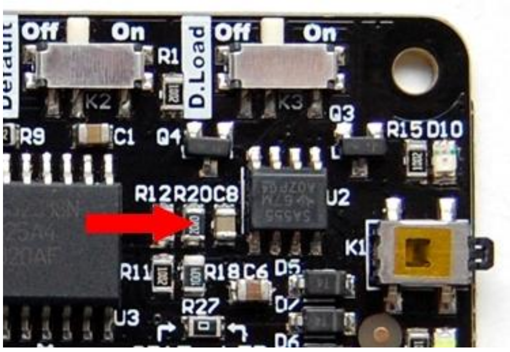
Default On/Off

The switch on the left has label “Default” can decide if you Raspberry Pi gets powered immediately when you connect the 5V power supply to Witty Pi Mini. By default, this switch is set to “Off”, so you have to tap the button once to power on your Raspberry Pi.

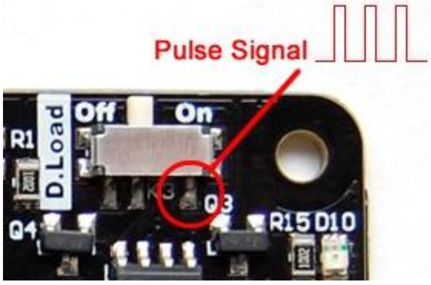
Pulsing Dummy Load

The switch on the right has label “D.Load”, witch stands for “Dummy Load”. This switch is to turn on/off the pulsing dummy load. The dummy load may be useful if you are using power bank to power your Witty Pi Mini + Raspberry Pi. When dummy load is on, it will draw certain current (~200mA) for very short period (~50ms), with a fixed interval (~5 seconds), which could be useful to keep your power bank alive with low average current consumption.

Please notice that, different power bank may have different current thresholds to keep them on. Witty Pi Mini’s pulsing dummy load may not necessarily meet all their requirements. If you find out that your power bank needs higher peak current to keep alive, you can change the resistor R20 (20 Ohms) to a smaller one.

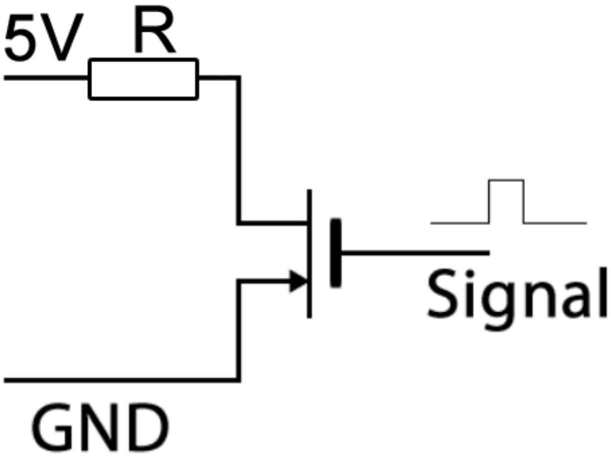


How the pulsing dummy load works? You may ask. The idea behind is quite straightforward. The SA555 chip will generate the pulsing signal, which is routed to the right lead of the switch K3, as shown below:



By default, the dummy load is off. If you turn the switch to the right side, the pulsing signal will be connected to a MOSFET, and controls the on/off of load (R20).

When dummy load switch is on, the pulsing signal connects to the gate of a N-MOSFET. When the signal goes high, the MOSFET is conducted and the current will be consumed (via the resistor R20).

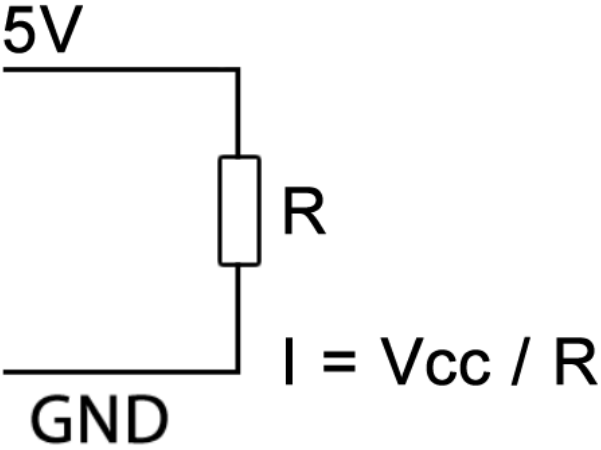


The peak current can be calculated with this formula:

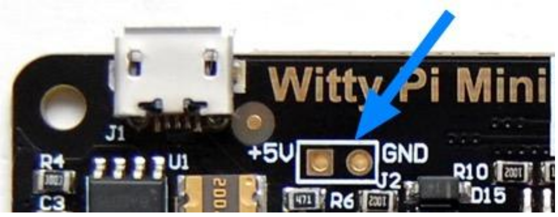
$$I_{max} = V_{cc} / (R + R_{ds})$$

Where Vcc is 5V, R is the load resistor (R20) and R_{ds} is the resister between the drain and source of the N-MOSFET, which is a few Ohms.

Some power banks have relatively complex algorism to decide whether they should cut off the power. Some of them even calculate the integration of current to make sure certain current should be consumed when the power is on. For those power banks, this “pulsing dummy load” trick may not work well. In this case you might want to use a simple resistor to be a real load to drain enough current and keep the power bank alive.

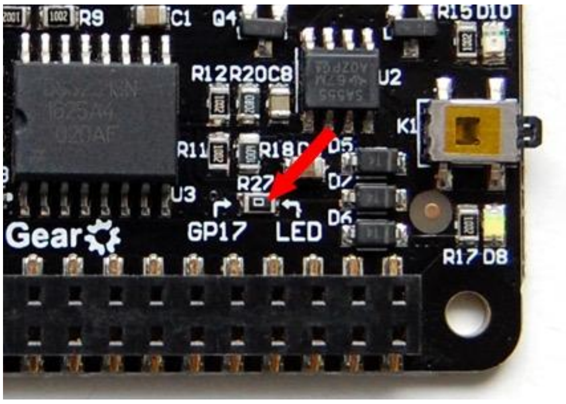


You can connect the resistor between the 5V and the ground, and you can make use of the unpopulated alternative power input connector.



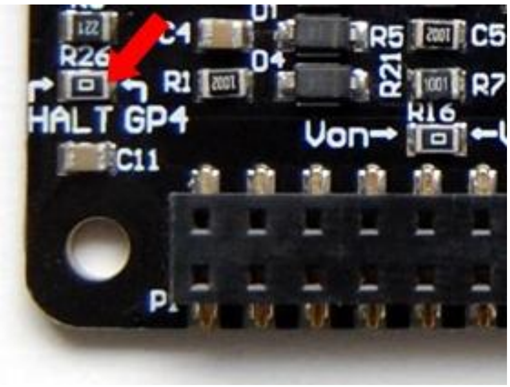
White LED Indicator

By default, the white LED indicator is driven by GPIO-17 pin. If you want to change that, you will need to make some hackings with your soldering iron (at your own risk). You can remove the resistor R27 (0 Ohm) and then connect the pad with “LED” label to the GPIO pin you want.



Halt Pin

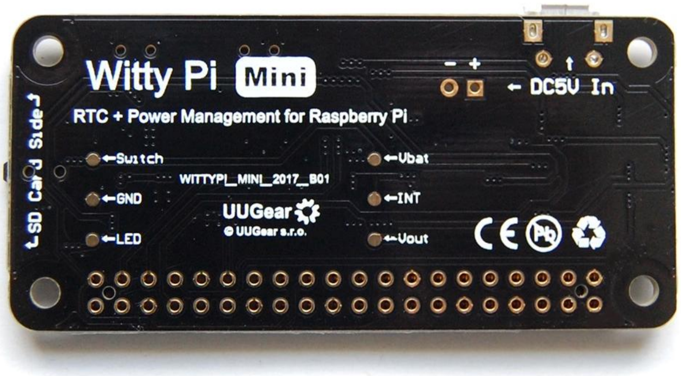
By default, GPIO-4 pin is the pin that receives signal to shutdown your Raspberry Pi. If you want to change that, you will need to make some hackings again (at your own risk). You can remove the resistor R26 (0 Ohm) and then connect the pad with “HALT” label to the GPIO pin you want.



You will also need to modify the software a little bit to use different GPIO pins. More details here: http://www.uugear.com/portfolio/change-the-pin-that-used-by-witty-pi/

Testing Pads on the Back

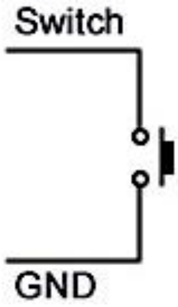
On the back of Witty Pi Mini board, there are 6 testing pads, and each of them has a label beside.



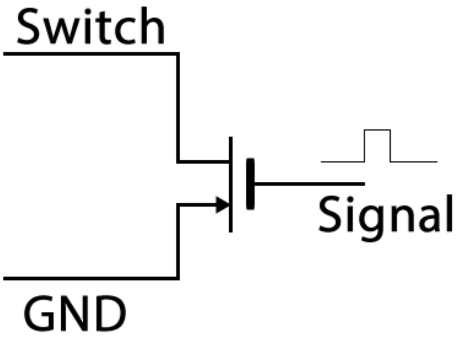
These testing pads were for product testing, but you can solder wires on them and them for extension and integration purpose.

Switch

It is the signal line that connects to the switch (button) on Witty Pi Mini. If you want to connect your own (2-lead) switch, just wire the two leads to Switch and GND pads.



Alternatively, if you wish to trigger Witty Pi Mini with external signal, you can use a N-channel MOSFET to achieve this:



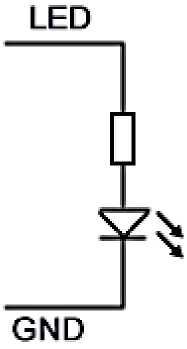
The signal should be a positive pulse, and the pulse length should be longer than 300ms. Processing a pulse will be equal to taping the switch once, so it will turn on your Raspberry Pi if your Pi is off, or turn off your Raspberry Pi if your Pi is on.

GND

It is the ground of the circuit board and all voltages are based on it.

LED

It is connected to the anode of the white LED. You can use this test pad to connect your own LED, but don’t forget to put a 1K resistor in serial.

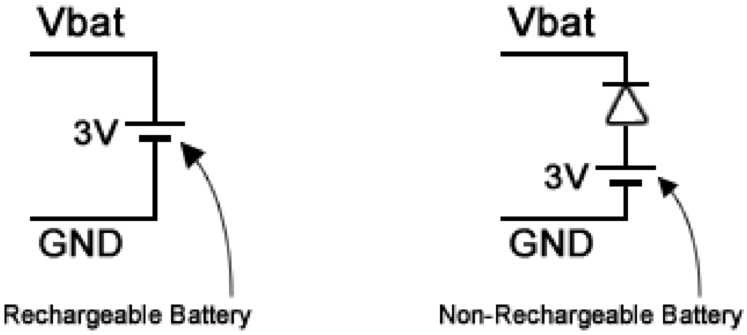


Vbat

This pin directly connects to the "+" terminal of the super capacitor. By measuring the voltage of this pin, you know if the capacitor is fully charged (3.3V means full). If you want to have a (much) longer off-power time keeping duration, you can connect your own 3V battery to this pad.

If you use the rechargeable battery, you just need to connect it directly to Vbat and GND pads. Witty Pi Mini will charge this battery with trickle current, when the power supply is connected.

If you use non-rechargeable battery, you will need to use a diode to make sure the battery will be discharged only.



INT

It is the interrupt signal that generated by the RTC alarm. It is in 5V level and has HIGH state (5V) by default. If any alarm occurs (scheduled startup or shutdown), it goes to LOW state (0V), and this state will be cleared once Witty Pi Mini's software detects and process it.

If the alarm occurs when Witty Pi Mini is not powered, the INT pad will stay in LOW state. When power is back online, your Raspberry Pi will boot immediately.

Vout

This pad is actually connected to the +5V pin on Raspberry Pi, which is the output voltage of Witty Pi Mini board. By detecting its voltage, you will know if your Raspberry Pi is in ON state.

The red LED indicator on board is also connected to this pad (via a resistor).

Witty Pi Mini Log Files

In the directory that you install your Witty Pi software, you can find two log files: **schedule.log** and **wittyPi.log**. If you need our help for solving a problem, please kindly put the log files in email attachment too. This will help us to help you better.

The “schedule.log” file contains the history of schedule script executions. Here you can see how the next shutdown and startup get scheduled. If you saw unexpected schedule script behavior, this log should be the first file to check.

The “wittyPi.log” file records the history of all Witty Pi activities. If you think your Witty Pi doesn’t behave right, this log file might provide more information for debugging.

The time-stamp in the log file is always the system time, instead of the RTC time. During the booting, the first log written by Witty Pi's software is "Witty Pi 2 daemon (v2.55) is started.", but its time-stamp might not be correct as the system time has not been updated by RTC yet (or if it is correct already, maybe the ntpd daemon updates the time earlier, when Internet is connected). You will see the correct time after the RTC time has been written to the system. For example:

```
[2017-05-09 11:10:14] Witty Pi 2 daemon (v2.55) is started.

[2017-05-09 11:10:16] Synchronizing time between system and Witty Pi...

[2017-05-09 11:10:16] Writing RTC time to system...

[2017-05-09 14:05:13] Done :-)
```

This correct time-stamp shows up in the line for "Done :-)", so this startup happened at 14:05.

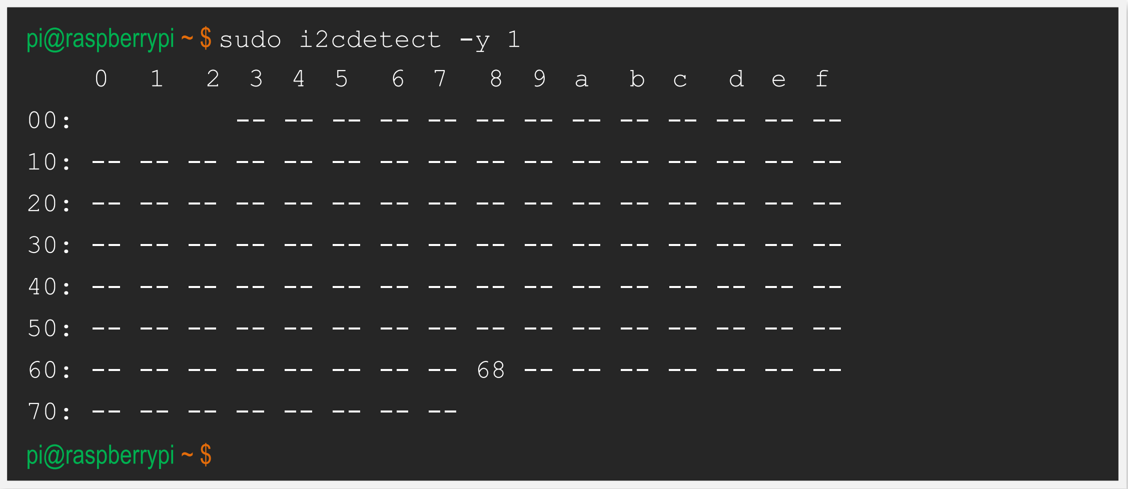
When the booting is done, the last log written by Witty Pi software is "Pending for incoming shutdown command...". If there is Internet connection, the NTP time will be applied to the system and RTC later.

Frequently Asked Questions (FAQ)

What I2C Address is Used by Witty Pi Mini?

Raspberry Pi communicates with the RTC chip (DS3231) on Witty Pi Mini via I^{2}C protocol. The DS3231 chip has a fixed I^{2}C address: **0x68**.

If you have Witty Pi Mini connected to Raspberry Pi and run “sudo i2cdetect -y 1” in the console, you will see this:



```
pi@raspberrypi ~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi ~ $
```

This address is fixed and you can never change it. If you want to use other I^{2}C devices on your Raspberry Pi, please make sure they have different I^{2}C addresses.

What GPIO Pins Are Used by Witty Pi Mini?

The GPIO pins used by Witty Pi Mini are marked with **green** color in the table below.

GPIO (BCM)	Name	Physical		Name	GPIO (BCM)
	3.3V	**1**	**2**	5V	
2	SDA 1	**3**	**4**	5V	
3	SCL 1	**5**	**6**	GND	
4	GPIO 7	**7**	**8**	TXD	14
	GND	**9**	**10**	RXD	15
17	GPIO 0	**11**	**12**	GPIO 1	18
27	GPIO 2	**13**	**14**	GND	
22	GPIO 3	**15**	**16**	GPIO 4	23
	3.3V	**17**	**18**	GPIO 5	24
10	MOSI	**19**	**20**	GND	
9	MISO	**21**	**22**	GPIO 6	25
11	SCLK	**23**	**24**	CE0	8
	GND	**25**	**26**	CE1	7
0	SDA 0	**27**	**28**	SCL 0	1
5	GPIO 21	**29**	**30**	GND	
6	GPIO 22	**31**	**32**	GPIO 26	12
13	GPIO 23	**33**	**34**	GND	
19	GPIO 24	**35**	**36**	GPIO 27	16
26	GPIO 25	**37**	**38**	GPIO 28	20
	GND	**39**	**40**	GPIO 29	21

As you can see in the table, Witty Pi Mini uses **GPIO-4**, **GPIO-17**, **GPIO-2 (SDA 1)** and **GPIO-3 (SCL 1).**

The usage of GPIO-4 and GPIO-17 are customizable. If you want to use other GPIO pins to do their job, you can follow this tutorial:

http://www.uugear.com/portfolio/change-the-pin-that-used-by-witty-pi/

GPIO-2 and GPIO-3 are for I^{2}C communication between Raspberry Pi and the RTC chip (DS3231). I^{2}C devices are identified by I^{2}C address, and they can share the I^{2}C pins as long as they have different I^{2}C addresses.

Witty Pi Mini doesn't actually use the TXD pin (it only monitors it). So using serial port for data transaction will not be a problem.

Is Witty Pi Mini Compatible with "Other Hardware"?

Please understand that we might not have the hardware you have on hand, and even if we have, it is difficult for us to make tests on all these hardware with Witty Pi Mini. Fortunately, it is not that difficult to figure out the answer by yourself. Basically you just need to consider the I^{2}C address and GPIO pins used by the "Other Hardware".

If the "Other Hardware" uses 0x68 I^{2}C address, it will be a confliction with Witty Pi's RTC chip and you can not use it with Witty Pi together.

If the "Other Hardware" doesn't use any GPIO pin that used by Witty Pi, and it doesn't use 0x68 I^{2}C address, then it should be compatible with Witty Pi.

If you have no idea which I^{2}C address (if applicable) or GPIO pins are used by the "Other Hardware", please contact its developer, as they certainly know their hardware and can provide you accurate information about it.

Witty Pi Mini dose not boot?

Some customers meet the situation that Witty Pi Mini only boot Raspberry Pi for a few seconds, and then shutdown Raspberry Pi with or without lighting the white LED first.

Remarks: you may need to disconnect Witty Pi Mini from your Raspberry Pi, and power on Raspberry Pi only for troubleshooting.

There are some possible reasons that can cause this kind of problem:

1. 1-Wire interface is enabled on GPIO-4, which conflicts with Witty Pi

If you have 1-Wire interface enabled, by default it will use GPIO-4. Because Witty Pi uses GPIO-4 pin to receive shutdown command, this will bring confliction and Witty Pi will shut down your Raspberry Pi after every boot. You won't even have chance to login the system.

In order avoid this problem, 1-Wire interface should be disabled, or assigned to a different GPIO pin before installation of Witty Pi's software. If the problem already happens, you need to take out the micro SD card from Raspberry Pi and access its file system via a card reader.

You need to edit the config.txt file in the "boot" volume to change the GPIO pin used by 1-Wire interface, or you can disable 1-Wire interface if you don't need it for now. You need to find something like "dtoverlay=w1-gpio" in the file.

If you want 1-Wire to use GPIO-18, just replace "dtoverlay=w1-gpio" with:

```
dtoverlay=w1-gpio,gpiopin=18
```

If you want to disable 1-Wire interface, just comment it out:

```
#dtoverlay=w1-gpio
```

Save the file and eject your micro SD card, and put it back to your Raspberry Pi. Now your Raspberry Pi should be able to boot normally.

2. The OS was installed via NOOBS, and boot menu was not skipped

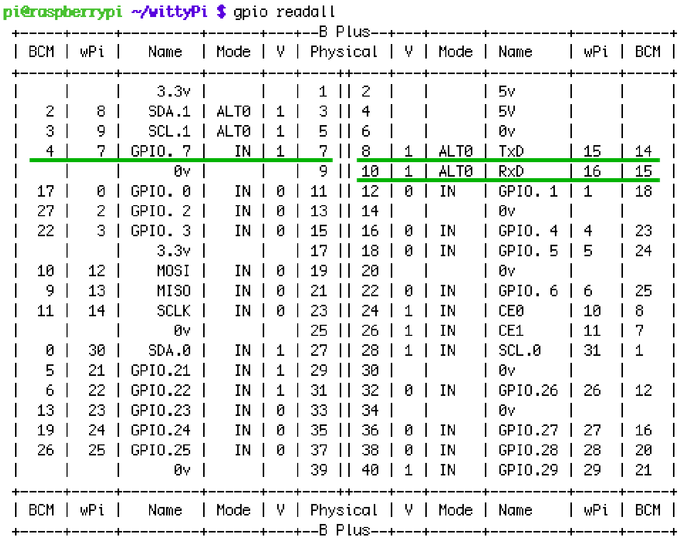
If you have a display connected to your Raspberry Pi during the booting, and you have not saw the Raspberry Pi logo before Witty Pi Mini cuts the power, then it is the case.

Please read the "Before Installation" section first, and make sure to skip the NOOBS boot menu. Otherwise the boot menu will delay the booting, and hence the TX pin doesn't raise fast enough, so Witty Pi Mini thinks the OS is down and cuts the power directly.

3. Serial port is disabled or misconfigured

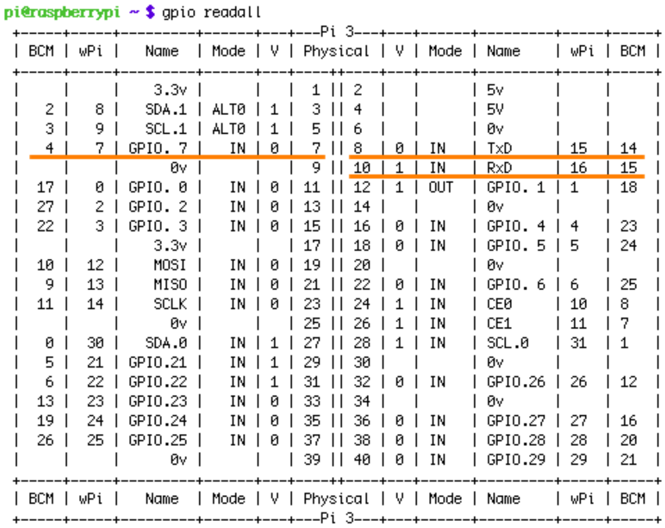
The TX pin in serial port should quickly go to ~3.3V when system is up. If for any reason, that TX pin has not realized to ~3.3V in given time frame (a few seconds), Witty Pi Mini will think the OS is down and will cut the power directly.

If disconnecting Witty Pi Mini and powering Raspberry Pi directly can allow you to enter the system, you can run "gpio readall" to check the pin state. The normal pin state should look like this:



```
pi@raspberrypi ~/wittyPi $ gpio readall
 +-----+-----+---------+------+---+--B Plus--+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 | ALT0 | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   SCL.1 | ALT0 | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | ALT0 | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALT0 | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 1 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+--B Plus--+---+------+---------+-----+-----+
```

If the serial port is disabled or misconfigured, you might see something like this:



```
pi@raspberrypi ~ $ gpio readall
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 | ALT0 | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   SCL.1 | ALT0 | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 0 |  7 || 8  | 0 | IN   | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | IN   | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 1 | OUT  | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
```

If you are not using Raspberry Pi 3, it is most probably because the serial port is disabled. If in the “/boot/config.txt” file, you can find “enable_uart=0”, please change it to “**enable_uart=1**” and reboot.

If you are using Raspberry Pi 3, you will need to move the Bluetooth to mini-UART(ttyS0) and restore UART0/ttyAMA0 over GPIO 14 and 15. You can achieve this by adding “**dtoverlay=pi3-miniuart-bt**” to the “/boot/config.txt” file. You will also need to add “**core_freq=250**” to the “/boot/config.txt” file to make sure Bluetooth can work properly. Don’t forget to reboot to let the changes take effect.

4. Software does not run automatically after Raspberry Pi system is on

After installing the software, you will have “daemon.sh”, “syncTime.sh” and “runScript.sh” scripts in the directory that has software installed. These scripts should be executed automatically after the operating system is up.

If these scripts are not automatically executed for any reason, Witty Pi Mini will cut the power of Raspberry Pi, **without lighting up the white LED**. In this case, the first place to check is the “/etc/init.d/wittypi” file, and there is possibility that it contains wrong paths to these scripts.

If you are running a special distribution of OS, make sure to check if the commands in the “/etc/init.d/wittypi” file do exist in the system, or it will fail silently. For example, Minibian doesn’t have “sudo” command.

5. Witty Pi Mini is unexpectedly woken up by shutdown alarm

This could happen, after you scheduled a shutdown in the future, and then Witty Pi Mini lost power connection. Raspberry Pi is off ungracefully and later the power connection is resume. If the shutdown alarm occurs when Raspberry Pi is still off, it will wake it up, which is unexpected. The software will detect this situation and shutdown your Raspberry Pi immediately, and you will see the white LED lights up before the shutdown.

Next time when you turn on your Raspberry Pi, it will be normal again, and you can find this message in the log, for previous shutdown:

```
Seems I was unexpectedly woken up by shutdown alarm, must go back to sleep...
```

Sometimes the same case can happen when you tap the button to turn on your Raspberry Pi, if the shutdown alarm occurs during the time that no power supply was connected.

You may ask, how comes the shutdown alarm could act like a startup alarm? This is specific to Witty Pi 2 and Mini. Witty Pi 2 and Mini uses the same RTC chip (DS3231), which is way better than the one used in on Witty Pi (DS1337). The downside however, is the better RTC chip merges two alarm pins into one. Hence without turning on your Raspberry Pi and check the RTC registers, there is no way to know if the current alarm is a shutdown alarm or not.

6. GPIO-4 pin doesn’t reach a stable status in given time

After the system is on, GPIO-4 pin should be in input mode and gets internally pulled up. However, during the startup the GPIO-4 pin could be unstable. In the **daemon.sh** script, the GPIO-4 listener will be started once the pin state hasn’t changed for 10 seconds. Once the GPIO-4 listener is started, any action that pulls down GPIO-4 will be regarded as a shutdown command. So if GPIO-4 pin doesn’t really get stable after the given 10 seconds, Witty Pi will take it as a shutdown command, **lights up the white LED** and then shutdown the system.

There are many factors that might cause the GPIO pin unstable, and **the most**

common one is the power supply. If your power supply is not strong enough, during the booting its voltage may drop from time to time, which may also make the GPIO pin voltage drop, and trigger the GPIO-4 listener to shutdown your Raspberry Pi.

If it is the case, you don’t have to replace the power supply. Just try to delay the starting of GPIO-4 listener, and in the major of cases it will help. You can modify the "**daemon.sh**" script, in line 66:

```
while [ $counter -lt 5 ]; do
```

Try to change the number 10 to 25. The bigger number you use, the later the GPIO-4 listener will be started.

This modification may workaround the problem. If it doesn’t, it means your GPIO-4 is really pulled down (by software or hardware), and you can confirm that by measuring the voltage on GPIO-4 with a multimeter. By default, the GPIO-4 should be internally pulled up. If it gets pulled down, try to find out who does this and don’t let it do this again, or you can use another pin to replace GPIO-4.

Revision History

Revision	Date	Description
1.00	2017.08.10	Initial revision
1.01	2017.09.08	Add more content to explain schedule script
1.02	2019.07.28	Add information about confliction with 1-Wire interface.