# Development Notes for the Raspberry Pi I2C Bit Banging

# Not Used: https://www.raspberrypi.org/forums/viewtopic.php?t=40743

# https://www.raspberrypi.org/forums/viewtopic.php?f=44&t=129678&p=866625

# http://abyz.co.uk/rpi/pigpio/

# PIGPIO Example in C: <https://github.com/LeoWhite/OptimusPi/blob/master/src/PiWars/I2C.cpp>

Pay attention to the log files to see if they are rotated and compressed.

Log file location is /var/log

REF: <https://www.raspberrypi.org/forums/viewtopic.php?f=63&t=134971#p898539>

The messages file in the log folder has the same repeating message.

rsyslogd-2007: action 'action 17' suspended, next retry is

## Hardware Used

* QTY 1 - Raspberry Pi Model B (Original Version)
* QTY 1 - WiFi adapter (ebay [150Mbps USB Wifi Wireless Adapter with 8188 cus realtek chip For Raspberry Pi B](http://www.ebay.com/itm/361463422872))
* QTY 1 - 16 GB SD Card ([Microcenter](http://www.microcenter.com/product/366176/16GB_microSDHC_Class_10_Flash_Memory_Card))
* QTY 1 – HDMI Monitor
* QTY 1 – Keyboard
* QTY 1 – Mouse
* QTY 1 to 9 – 4 conductor 3.5mm audio cable(s)
* QTY 1 to 9 – Attiny85 I2C Temperature, Humidity, and Light sensor(s)
* QTY 1 - Custom Made Raspberry Pi Hat (Need to document)

## Windows Applications to be used

* Win32DiskImager <https://sourceforge.net/projects/win32diskimager/>
* PuTTY <http://www.putty.org/>
* WinSCP <https://winscp.net/>

## Steps

1. Write Raspbian Image to SD Card
   1. Download latest Raspbian Image (Debian Jessie) from <https://www.raspberrypi.org/downloads/raspbian/>
      * Version: March 2016
      * Release date: 2016-03-18
      * Kernel version: 4.1
      * SHA-1: db41f2a8c6236c0ca9150fe4db2017c09e7871fb
   2. Unzip Image
   3. Write to SD Card using Win32DiskImager <https://sourceforge.net/projects/win32diskimager/>
2. Boot up the Raspberry Pi with the prepared SD Card
3. Make a few configuration changes
   1. Open a terminal window
   2. Type the following command  
      sudo raspi-config
   3. Unless you are in the UK you may want to change the “5 Internalisation Options”  
      ***NOTE:*** *These are the settings I chose. Your choices may be different.*
      * “I1 Change Local”  
        Removed “en\_GB.UTF-8 UTF-8”  
        Added “en\_US.UTF-8 UTF-8”  
        On next screen, I left default local to “None”
      * “I2 Change Timezone”  
        Geographic area: US  
        Time zone: Eastern
      * “I3 Change Keyboard Layout”  
        Left as is since I do have a UK Keyboard
      * “I4 Change Wi-fi Country”  
        US United States
   4. Most likely you will want to choose “1 Expand Filesystem” to use the remaining space on your SD Card
      * After the filesystem has been expanded, exit raspi-config and reboot the Raspberry Pi
4. Connect the Raspberry Pi to the Wi-Fi network if you are using wireless by clicking on the network icon on the top right of the screen  
   BTW: If you did not perform the step above to change Wi-Fi country, you may have issues with the Wi-Fi being unreliable. I had this issue earlier and did not realize that was the reason why. It did work most of the time so I did not think much of it.
5. Find the assigned IP Address so we can use PuTTY and WinSCP
   1. Open a terminal window
   2. Type the following command  
      ifconfig
   3. Take note of the wlan0 IP Address

|  |
| --- |
| wlan0 Link encap:Ethernet HWaddr xx:xx:xx:xx:xx:xx  inet addr:192.168.1.178 Bcast:192.168.1.255 Mask:255.255.255.0  inet6 addr: xxxx::xxxx:xxxx:xxx:xxxx/64 Scope:Link  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  RX packets:407 errors:0 dropped:203 overruns:0 frame:0  TX packets:94 errors:0 dropped:4 overruns:0 carrier:0  collisions:0 txqueuelen:1000  RX bytes:139109 (135.8 KiB) TX bytes:42920 (41.9 KiB) |

* 1. Now we can use PuTTY to issue commands to the Raspberry Pi. From now on when I refer to opening a terminal, I will be using PuTTY but you are free to use the terminal on the Raspberry Pi.

1. Update and Upgrade to get the latest software changes  
   REF: <https://www.raspberrypi.org/documentation/raspbian/updating.md>
   1. Open a terminal window
   2. Type the following command  
      sudo apt-get update
   3. Type the following command  
      sudo apt-get upgrade
   4. Optionally, you may free up space used in the archives by issuing the following command *(I did not apply)*   
      sudo apt-get clean  
        
      You may run the following command to determine if you need to delete the archives.  
      df -h  
      Look at the free space left in /dev/root.
2. Install PIGPIO  
   REF: <http://abyz.co.uk/rpi/pigpio/download.html>
   1. From a terminal window, issue the following commands
      * wget abyz.co.uk/rpi/pigpio/pigpio.zip
      * unzip pigpio.zip
      * cd PIGPIO
      * make -j4
      * sudo make install
   2. Validate that the library installed correctly by running the following command  
      *Initially I saw a failed test on TEST 6.1. I removed the Pi Hat so that nothing was connected to pin 22 and reran the tests. This time all of the tests passed.*
      * sudo ./x\_pigpio # check C I/F
3. Shut down the Raspberry Pi, plug in the Pi Hat, and boot the Raspberry Pi
4. If you installed 9 LEDs on the Pi Hat, you may notice that one of the LEDs is on when the Raspberry Pi boots. Let’s correct that issue by having a script run when the Raspberry Pi boots to put all of LED GPIO Pins in output mode with a low value.  
   *NOTE: In my case, GPIO14 (TXD) was high so LED 8 was in the on state*
   1. Install wiringPi  
      REF: <https://projects.drogon.net/raspberry-pi/wiringpi/download-and-install>  
      From a terminal window, issue the following commands
      * git clone git://git.drogon.net/wiringPi
      * cd wiringPi
      * git pull origin
      * ./build

|  |
| --- |
| NOTE: To compile programs with wiringPi, you need to add:  -lwiringPi  to your compile line(s) To use the Gertboard, MaxDetect, etc.  code (the devLib), you need to also add:  -lwiringPiDev  to your compile line(s). |

* 1. Use WinSCP to connect to the Raspberry Pi and copy setupgpio.sh from the Raspberry Pi folder on the PC to the /home/pi/ folder on the Raspberry Pi
  2. Make setupgpio.sh executable by running the following command from the /home/pi directory  
     *HINT: You may need to change to the proper folder first. cd ..*  
     chmod +x setupgpio.sh
  3. Run the script to make certain that all of the LEDs turn off  
     ./setupgpio.sh
  4. Run setupgpio.sh on every boot
     + In a terminal window, issue the following command  
       sudo nano /etc/rc.local
     + Add the following lines at the end of the file but before the line “exit 0”  
       # setup the gpio pins for gate control  
       /home/pi/setupgpio.sh
     + Press <Ctrl> O to save the file
     + Press <Ctrl> X to exit nano editor
     + Reboot the Raspberry Pi and verify that the LED(s) go out on boot

1. We need to have the pigpio daemon start when the Raspberry Pi boots  
   REF: <https://www.raspberrypi.org/forums/viewtopic.php?f=32&t=103752>
   1. Use WinSCP to connect to the Raspberry Pi and copy pigpiod from the Raspberry Pi folder on the PC to the /home/pi/ folder on the Raspberry Pi
   2. From the terminal window, run the following commands
      * sudo chmod 755 pigpiod
      * sudo cp pigpiod /etc/init.d/pigpiod
      * sudo update-rc.d pigpiod defaults
      * sudo update-rc.d pigpiod enable
   3. Reboot the Raspberry Pi
   4. When the Raspberry Pi boots back up, verify that the pigpio daemon has started by typing the following command in a terminal window
      * ps aux | grep "pigpiod"
      * The output should be similar to the following

|  |
| --- |
| pi@raspberrypi:~ $ ps aux | grep "pigpiod"  root 377 11.1 0.3 9856 1580 ? SLsl 21:02 0:06 /usr/local/bin/pigpiod |

1. Now, get the program loaded and running every 5 minutes
   1. Use WinSCP to connect to the Raspberry Pi and copy the following files from the Raspberry Pi folder on the PC to the /home/pi/ folder on the Raspberry Pi
      * getdevval\_bb
      * thl.config
      * thl.py
   2. Edit the thl.config file with your settings
      * SmtpServer – The name of the SMTP Server such as smtp.gmail.com.
      * SmtpPort – The port for the SMTP Server. This is typically 25 or 587.
      * SmtpUserName – The Username for your SMTP server. (Email account to send email.)
      * SmtpPassword – The password for your SMTP server. (Email account to send email.)
      * EmailAddresses – You may add email addresses to send alerts to when there is an error reading a sensor. If you use more than one email address, separate them with a semicolon.
      * AlertSubject – You may leave the default setting or change it. This is the subject of the email that will be sent if there is an error reading one of the sensors.
      * AdafruitIoKey – When you go to io.adafruit.com and setup an account, click on “Settings” > “VIEW AIO KEYS”. Copy the value shown and paste it into the config file.
      * You may also modify the sensor sections to match the hardware that you have and any names that you would like to use for the sensors and feeds. The only restriction on the section names is that they must start with the word “Sensor” in order to be picked up by the Python Script.
        + I2CAddress – This is the decimal representation of the I2C address of the sensor. The valid values are 8 to 119.  
          REF: <http://www.totalphase.com/support/articles/200349176-7-bit-8-bit-and-10-bit-I2C-Slave-Addressing>
        + SensorName – This is a name used for information messages such as emails and logs.
        + AIOFeed\_Temperature – A name to use for a feed on Adafuit.IO for the temperature.
        + AIOFeed\_Humidity – A name to use for a feed on Adafuit.IO for the humidity.
        + AIOFeed\_Light – A name to use for a feed on Adafuit.IO for the light level.
        + DataFile – The name of a local file to use to collect data for this sensor. This is useful in the event of connection issues with Adafruit.IO.
        + LogFile - The name of a local file to use to collect error messages for this sensor.
        + ModelNumber – This is the model number expected from the sensor. It is used as a final check to ensure that data was properly read.
        + LED\_Pin – This is the pin number not the GPIO number of the pin which has an LED connected. The LED will light when the sensor is being read.
        + IsActive – Valid values are True and False. If this is set to True, the Python Script will attempt to read from the sensor.
   3. Setup the cron job to run the Python Script once every 5 minutes.
      * From a terminal window, issue the following command.  
        crontab –e
      * Select an editor from the list
      * Add the following line to the file  
        \*/5 \* \* \* \* /home/pi/thl.py
      * Save the file by pressing <Ctrl> O
      * Exit by pressing <Ctrl> X
   4. Wait for 6 minutes and check if the data and log files were written and contain information. Also check Adafruit.IO and see if you now have the feeds listed with data.