

CAMOSUN COLLEGE
Department of Electronics

ELEX 235 Lab 8

Raspberry Pi One Wire

Part1:

Write a c program to read the contents of the w1 virtual files and display the number of one wire devices connected, their serial numbers, and the corresponding temperatures of any DS1820 family of devices. The program should update every minute.

Be prepared to demo the lab by the start of lab week9. Put a screenshot of your onewire temp data and the code in the D2L week8 dropbox.

The raspbian image has support for One Wire interfaces. The device driver makes the one wire bus available on **GPIO4** by default.

From the DS18S20 datasheet:

Figure 4. Supplying the Parasite-Powered DS18S20 During Temperature Conversions

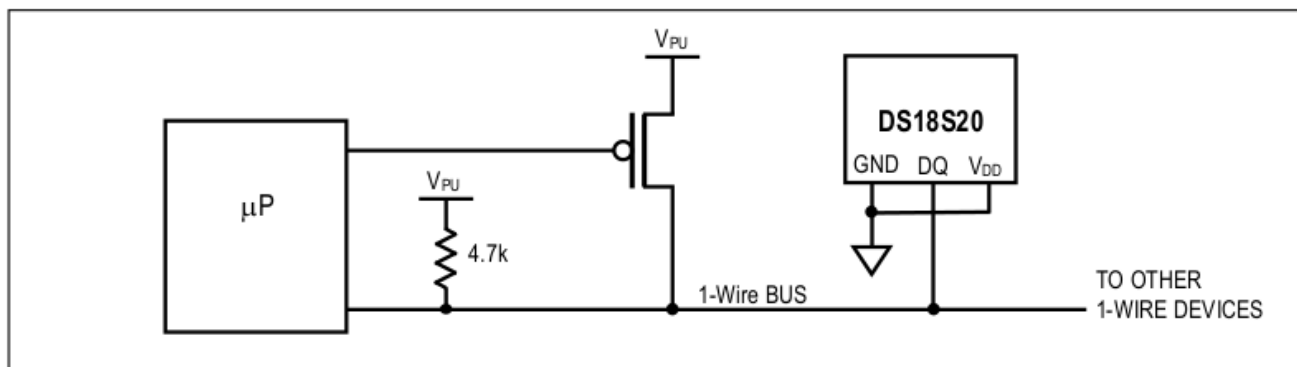
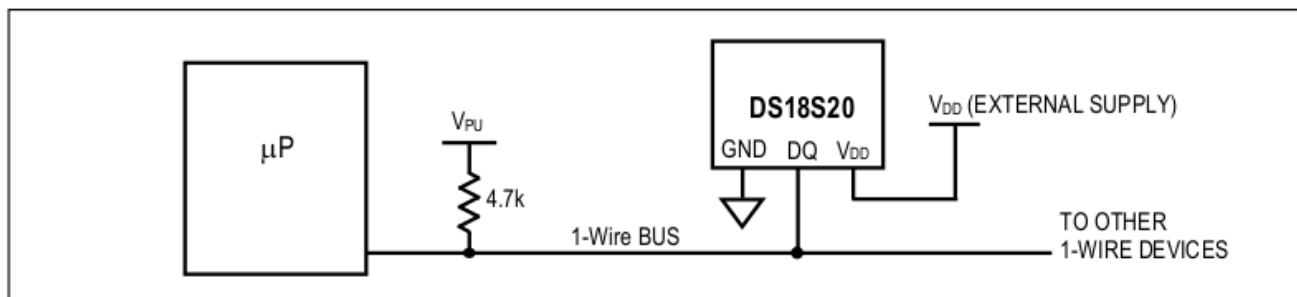


Figure 5. Powering the DS18S20 with an External Supply



Connect a DS18S20 to the raspberry pi.

We can see if the driver is enabled by looking at the virtual directory /sys . In particular we want to look at what bus interfaces are active so:

ls /sys/bus

```
pi@piRTC: ~  
pi@piRTC ~ $ ls /sys/bus/  
amba          clocksource  event_source i2c          mmc          scsi  spi  workqueue  
clockevents   cpu          hid          iscsi_flashnode platform  sdio  usb  
pi@piRTC ~ $
```

The one wire bus will show up as w1. By default in the raspbian image, the one wire bus is supported by kernel modules. As the one wire bus entry is not there, we need to look for the kernel modules.

lsmod

```
pi@piRTC ~ $ lsmod  
Module                Size  Used by  
snd_soc_wm8804         7833  0  
snd_soc_pcm512x        8909  0  
i2c_bcm2708            4719  0  
snd_bcm2835            18169 0  
snd_soc_bcm2708_i2s    5486  0  
regmap_mmio            2818  1 snd_soc_bcm2708_i2s  
snd_soc_core          128166 3 snd_soc_pcm512x,snd_soc_wm8804,snd_soc_bcm2708_i2s  
regmap_spi             1913  3 snd_soc_pcm512x,snd_soc_wm8804,snd_soc_core  
snd_pcm_dmaengine      5481  1 snd_soc_core  
snd_pcm               81518 3 snd_bcm2835,snd_soc_core,snd_pcm_dmaengine  
snd_page_alloc         5168  1 snd_pcm  
regmap_i2c             1657  3 snd_soc_pcm512x,snd_soc_wm8804,snd_soc_core  
snd_compress           8136  1 snd_soc_core  
snd_seq               54581 0  
snd_timer              20353 2 snd_pcm,snd_seq  
snd_seq_device         6485  1 snd_seq  
8192cu                 551136 0  
leds_gpio              2055  0  
led_class              4119  1 leds_gpio  
snd                    61518 7 snd_bcm2835,snd_soc_core,snd_timer,snd_pcm,snd_seq,snd_seq_device,snd_compress  
pi@piRTC ~ $
```

These are not installed by default, so we need to install the following modules

w1_therm

w1_gpio

```
sudo modprobe w1_therm
sudo modprobe w1_gpio
```

lsmod

```
pi@piRTC: ~
pi@piRTC ~ $ sudo modprobe w1_therm w1_gpio
pi@piRTC ~ $ lsmod
Module                Size  Used by
w1_therm              2870   0
wire                  25249  1 w1_therm
cn                     4795   1 wire
snd_soc_wm8804         7833   0
snd_soc_pcm512x        8909   0
i2c_bcm2708            4719   0
snd_bcm2835            18169  0
snd_soc_bcm2708_i2s    5486   0
regmap_mmio           2818   1 snd_soc_bcm2708_i2s
snd_soc_core          128166  3 snd_soc_pcm512x,snd_soc_wm8804,snd_soc_bcm2708_i2s
regmap_spi            1913   3 snd_soc_pcm512x,snd_soc_wm8804,snd_soc_core
snd_pcm_dmaengine      5481   1 snd_soc_core
snd_pcm               81518  3 snd_bcm2835,snd_soc_core,snd_pcm_dmaengine
snd_page_alloc         5168   1 snd_pcm
regmap_i2c             1657   3 snd_soc_pcm512x,snd_soc_wm8804,snd_soc_core
snd_compress           8136   1 snd_soc_core
snd_seq                54581   0
snd_timer              20353  2 snd_pcm,snd_seq
snd_seq_device         6485   1 snd_seq
8192cu                 551136  0
leds_gpio              2055   0
led_class              4119   1 leds_gpio
snd                    61518  7 snd_bcm2835,snd_soc_core,snd_timer,snd_pcm,snd_seq,snd_seq_device,snd_compress
pi@piRTC ~ $
```

Now the one wire bus should be found in /sys/bus

ls /sys/bus

```
pi@piRTC: ~
pi@piRTC ~ $ ls /sys/bus/
amba          clocksource  event_source  i2c          mmc          scsi  spi  w1
clockevents   cpu          hid           iscsi_flashnode platform      sdio  usb  workqueue
pi@piRTC ~ $
```

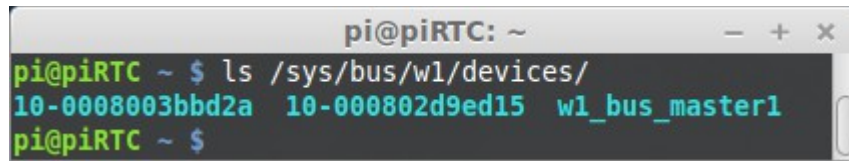
ls /sys/bus/w1

```
pi@piRTC: ~
pi@piRTC ~ $ ls /sys/bus/w1
devices  drivers  drivers_autoprobe  drivers_probe  uevent
pi@piRTC ~ $
```

Add the following line to /boot/config.txt

dtoverlay=w1-gpio

ls /sys/bus/w1/devices



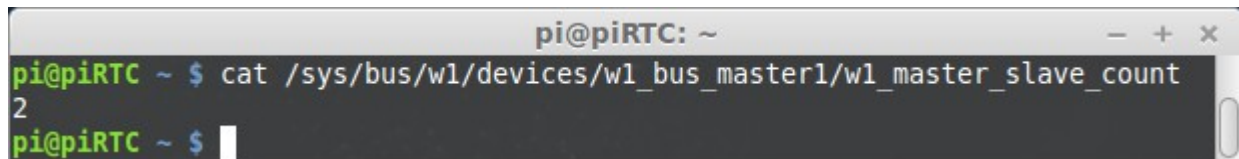
```
pi@piRTC: ~  
pi@piRTC ~ $ ls /sys/bus/w1/devices/  
10-0008003bbd2a 10-000802d9ed15 w1_bus_master1  
pi@piRTC ~ $
```

We have 2 one wire devices attached to this bus. Each device will appear under a directory entry named with its serial number. Here we have 10-0008003bbd2a and 10-000802d9ed15. The 10- indicates these are DS1820 or DS18S20 one wire temperature devices. It would be 28- for DS18B20 devices.

Under the /sys/bus/w1/devices/w1_bus_master1 directory, there are a number of file entries to provide information.

Looking at the contents of /sys/bus/w1/devices/w1_bus_master1/w1_master_slave_count

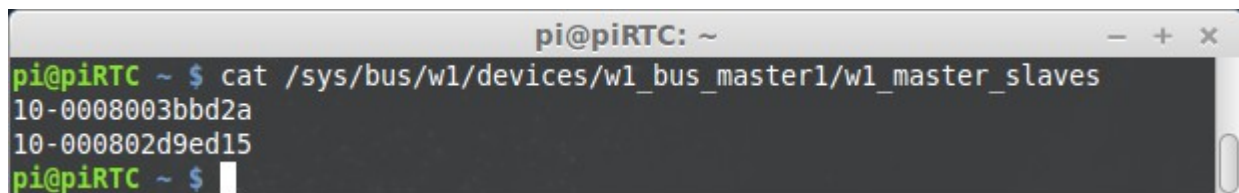
cat /sys/bus/w1/devices/w1_bus_master1/w1_master_slave_count



```
pi@piRTC: ~  
pi@piRTC ~ $ cat /sys/bus/w1/devices/w1_bus_master1/w1_master_slave_count  
2  
pi@piRTC ~ $
```

We see a response of 2 devices.

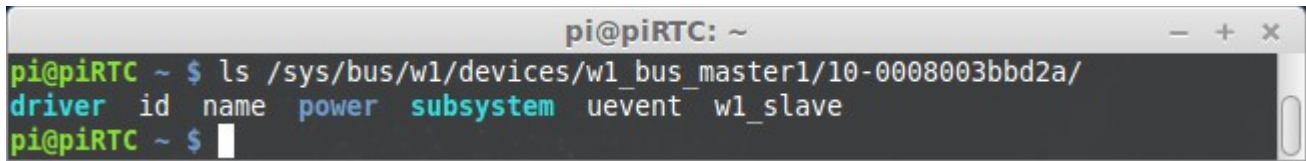
cat /sys/bus/w1/devices/w1_bus_master1/w1_master_slaves



```
pi@piRTC: ~  
pi@piRTC ~ $ cat /sys/bus/w1/devices/w1_bus_master1/w1_master_slaves  
10-0008003bbd2a  
10-000802d9ed15  
pi@piRTC ~ $
```

We display the contents of a file containing the serial numbers of the one wire devices found.

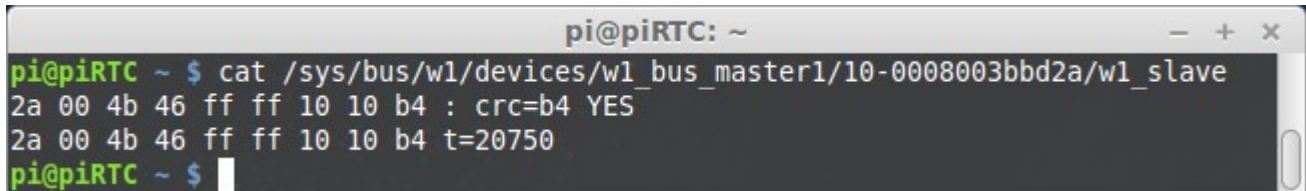
```
ls /sys/bus/w1/devices/w1_bus_master1/10-0008003bbd2a
```



```
pi@piRTC: ~  
pi@piRTC ~ $ ls /sys/bus/w1/devices/w1_bus_master1/10-0008003bbd2a/  
driver id name power subsystem uevent w1_slave  
pi@piRTC ~ $
```

shows the files and directories and shortcut links available for this device.

```
cat /sys/bus/w1/devices/w1_bus_master1/10-0008003bbd2a/w1_slave
```



```
pi@piRTC: ~  
pi@piRTC ~ $ cat /sys/bus/w1/devices/w1_bus_master1/10-0008003bbd2a/w1_slave  
2a 00 4b 46 ff ff 10 10 b4 : crc=b4 YES  
2a 00 4b 46 ff ff 10 10 b4 t=20750  
pi@piRTC ~ $
```

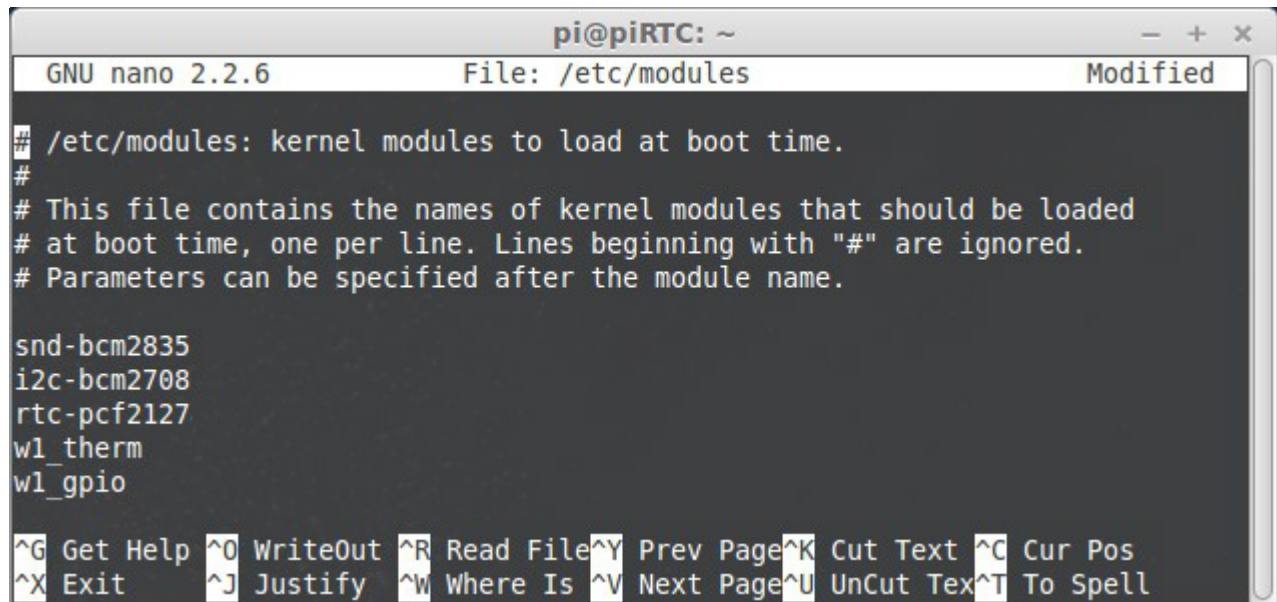
```
cat /sys/bus/w1/devices/10-0008003bbd2a/w1_slave
```



```
pi@piRTC: ~  
pi@piRTC ~ $ cat /sys/bus/w1/devices/10-0008003bbd2a/w1_slave  
29 00 4b 46 ff ff 04 10 a6 : crc=a6 YES  
29 00 4b 46 ff ff 04 10 a6 t=20500  
pi@piRTC ~ $
```

The devices show up under the w1_bus_master1 directory or the devices directory. Listing either locations w1_slave file contents will display the DS1820 eeprom contents.

To make the one wire drivers load at startup we need to add the modules to the bottom of the /etc/modules file



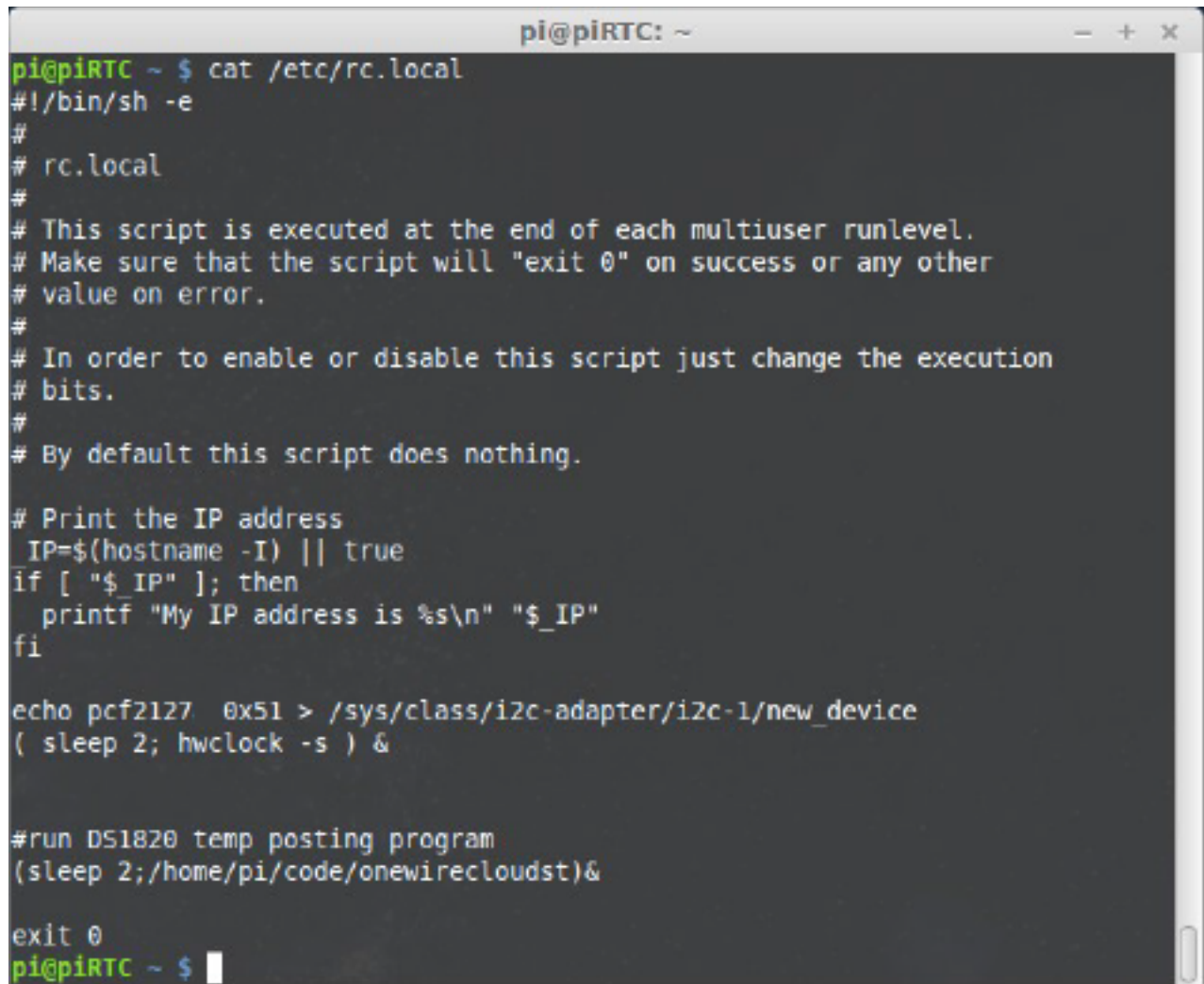
The image shows a terminal window titled "pi@piRTC: ~" with standard window controls. The terminal is running GNU nano 2.2.6, editing the file /etc/modules. The file's content is as follows:

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.

snd-bcm2835
i2c-bcm2708
rtc-pcf2127
wl_therm
wl_gpio
```

At the bottom of the window, a status bar displays various keyboard shortcuts for nano, such as ^G Get Help, ^O WriteOut, ^R Read File, ^Y Prev Page, ^K Cut Text, ^C Cur Pos, ^X Exit, ^J Justify, ^W Where Is, ^V Next Page, ^U UnCut Tex, and ^T To Spell.

To make your program run at startup, edit the /etc/rc.local file.

A terminal window titled 'pi@piRTC: ~' with standard window controls. It displays the contents of the /etc/rc.local file. The script starts with a shebang, followed by comments explaining its purpose and how to enable/disable it. It then prints the IP address, adds a new I2C device, and runs a background task for a DS18B20 temperature sensor. The script ends with an 'exit 0' command.

```
pi@piRTC ~ $ cat /etc/rc.local
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
IP=$(hostname -I) || true
if [ "$IP" ]; then
    printf "My IP address is %s\n" "$IP"
fi

echo pcf2127 0x51 > /sys/class/i2c-adapter/i2c-1/new_device
( sleep 2; hwclock -s ) &

#run DS1820 temp posting program
(sleep 2;/home/pi/code/onewirecloudst)&

exit 0
pi@piRTC ~ $
```

We have added a comment line:

#run DS1820 temp posting program

The line below runs as a background task. It sleeps 2 seconds and runs the onewirecloudst app.

(sleep 2;/home/pi/code/onewirecloudst) &

Make sure the lines are entered above the exit 0 call.

To run a program at specific times, we can utilize the cron daemon.

crontab -e will allow you to edit our cron jobs.

From <http://www.thegeekstuff.com/2009/06/15-practical-crontab-examples/>

Linux Crontab Format

```
MIN HOUR DOM MON DOW CMD
```

Table: Crontab Fields and Allowed Ranges (Linux Crontab Syntax)

| Field | Description | Allowed Value |
|-------|--------------|-----------------------------|
| MIN | Minute field | 0 to 59 |
| HOUR | Hour field | 0 to 23 |
| DOM | Day of Month | 1-31 |
| MON | Month field | 1-12 |
| DOW | Day Of Week | 0-6 |
| CMD | Command | Any command to be executed. |

1. Scheduling a Job For a Specific Time

The basic usage of cron is to execute a job in a specific time as shown below. This will execute the Full backup shell script (full-backup) on **10th June 08:30 AM**.

Please note that the time field uses 24 hours format. So, for 8 AM use 8, and for 8 PM use 20.

```
30 08 10 06 * /home/ramesh/full-backup
```

- **30** – 30th Minute
- **08** – 08 AM
- **10** – 10th Day
- **06** – 6th Month (June)
- ***** – Every day of the week

2. Schedule a Job For More Than One Instance (e.g. Twice a Day)

The following script take a incremental backup twice a day every day.

This example executes the specified incremental backup shell script (incremental-backup) at 11:00 and 16:00 on every day. The comma separated value in a field specifies that the command needs to be executed in all the mentioned time.

```
00 11,16 * * * /home/ramesh/bin/incremental-backup
```

- **00** – 0th Minute (Top of the hour)
- **11,16** – 11 AM and 4 PM
- ***** – Every day
- ***** – Every month
- ***** – Every day of the week

3. Schedule a Job for Specific Range of Time (e.g. Only on Weekdays)

If you wanted a job to be scheduled for every hour with in a specific range of time then use the following.

Cron Job everyday during working hours

This example checks the status of the database everyday (including weekends) during the working hours 9 a.m – 6 p.m

```
00 09-18 * * * /home/ramesh/bin/check-db-status
```

- **00** – 0th Minute (Top of the hour)
- **09-18** – 9 am, 10 am, 11 am, 12 am, 1 pm, 2 pm, 3 pm, 4 pm, 5 pm, 6 pm
- ***** – Every day
- ***** – Every month
- ***** – Every day of the week

Cron Job every weekday during working hours

This example checks the status of the database every weekday (i.e excluding Sat and Sun) during the working hours 9 a.m – 6 p.m.

```
00 09-18 * * 1-5 /home/ramesh/bin/check-db-status
```

- **00** – 0th Minute (Top of the hour)
- **09-18** – 9 am, 10 am, 11 am, 12 am, 1 pm, 2 pm, 3 pm, 4 pm, 5 pm, 6 pm
- ***** – Every day
- ***** – Every month
- **1-5** - Mon, Tue, Wed, Thu and Fri (Every Weekday)