

**A REPORT
ON
VOLTAGE/CURRENT/POWER MONITORING USING
OPENBMC AND RASPBERRY PI 3**

BITS ZG628T: Dissertation

BY

**Pratik Arun Farkase
2020MT12443**

Dissertation work carried out

AT

**Sasken Technologies Limited, Sharayu Harmony Mhalunge, Baner Pune,
Maharashtra - 411045**



BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE

PILANI (RAJASTHAN)

APRIL 2022

**A REPORT
ON
VOLTAGE/CURRENT/POWER MONITORING USING
OPENBMC AND RASPBERRY PI 3**

BITS ZG628T: Dissertation

BY

**Pratik Arun Farkase
2020MT12443**

Prepared in partial fulfillment of **M.Tech Software Systems** Degree Programme

Dissertation work carried out

AT

**Sasken Technologies Limited, Sharayu Harmony Mhalunge, Baner Pune,
Maharashtra - 411045**



BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE

PILANI (RAJASTHAN)

APRIL 2022

ACKNOWLEDGEMENTS

I avail this opportunity to extend my Hearty indebtedness and Sincere Gratitude to my **Faculty Mentor cum Supervisor Mr. Sagar Purushottam Chopade** and **Additional Examiner Mr. Gopal Sunil Rathod** for their valuable guidance, constant encouragement and kind help at different stages for the execution of this work, for their ever encouraging and moral support.

I also express my Sincere Gratitude to **BITS Faculty Evaluator Mr. Tejasvi Alladi** for giving me necessary feedback and comments on the work done and also encouraging me to integrate new ideas into the project. I also thank other **BITS project faculties** for giving me an opportunity to execute this project and believing in me.

I sincerely thank to all academic and non-teaching staffs in **BITS Pilani, Rajasthan** who helped me. My sincere thanks to the author whose works I have consulted and quoted in this work.

Head of the organization : Rajiv C. Mody (CEO, SASKEN TECHNOLOGIES LTD)

Name of Supervisor : Sagar Purushottam Chopade

Name of Additional Examiner : Gopal Sunil Rathod

Professional expert / In-charge of the project : Pratik Arun Farkase

Faculty Mentor : Sagar Purushottam Chopade

ABSTRACT SHEET

A Successful working model of this project “Voltage/Current/Power Monitoring using OpenBMC and Raspberry Pi 3” was developed by me under the guidance of the Supervisor **Mr. Sagar Purushottam Chopade**. The following work was carried out throughout this project to get a working prototype :

- Designed and Developed a cutting edge device based on Raspberry Pi 3 and OpenBMC that can monitor the Current, Voltage and Power of various loads attached to it.
- Successfully Built OpenBMC image for Raspberry Pi 3 Board using Yocto build system and made sure that the image runs correctly.
- Interfaced and Integrated ina219 current/voltage/powerSensor from Texas Instruments to Raspberry Pi 3 with I2C protocol using Device Tree Overlays and Linux Device Driver integration.
- Successfully Mapped the sensor data to the OpenBMC Phosphor-webui and verified the correctness of the Current/Power/Voltage values received from the sensor.
- Performed Functional/Unit tests on the final product with proper validation procedures.

More in-depth knowledge of the process can be obtained in the upcoming parts of the project report.

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)**

WILP Division

Organization : Sasken Technologies Limited

Location : Pune

Duration : 36 Months

Date of Start : 15th April

Date of Submission : 25th April 2022

Title of the Project : Voltage/Current/Power Monitoring using OpenBMC and Raspberry Pi 3

Name of the Student : Pratik Arun Farkase

ID No. : 2020MT12443

Name of Supervisor : Sagar Purushottam Chopade

Designation : System Software Design and Development Engineer

Name of Additional Examiner : Gopal Sunil Rathod

Designation : System Software Design and Development Engineer

Name of the Faculty Mentor : Sagar Purushottam Chopade

Keywords : raspberry pi, openbmc, yocto project, ina219, i2c, device trees, linux kernel, servers, system integration, software development and management, agile development.

Project Areas : Software for Embedded System, Embedded Linux, Linux Kernel Development, Raspberry Pi 3 Development, Embedded Development in C/C++


Signature of the Student

Name : Pratik Arun Farkase

Date: 25/04/2022

Place: Nagpur


Signature of the Supervisor

Name: Sagar Purushottam Chopade

Date: 25/04/2022

Place: Nagpur

TABLE OF CONTENTS

PAGE NUMBER

I.	Cover	1
II.	Title Page	2
III.	Acknowledgements	3
IV.	Abstract Sheet	4
V.	Table of Contents	6
VI.	Introduction	7
VII.	Main Text	8
VIII.	Testing and Conclusions	20
IX.	References	21
X.	Glossary	22
XI.	Checklist	23

INTRODUCTION

High Performance and Advanced Linux Server nowadays continuously stream data to thousands of device running Android/IOS for example Netflix/Amazon servers. So monitoring the performance and health of these servers becomes essential. Most of these run OpenBMC and this OS gives us the framework to monitor the server statistics remotely. It's a world of simplicity. In this era of IOT and cloud based devices connected to the internet, we require devices that can stay out there in the field doing some sophisticated computing while we can monitor them and get the data sitting at the comfort of our home. We were motivated to develop a device that can measure the current, voltage and power of various loads attached to it. Let's say we have a person sitting in Amsterdam and he needs to monitor the fan speed of a Netflix server which is located in London. His main intention is to monitor the speeds with which the fan blades of the server are rotating so that the temperature of the server is under control. As these servers continuously stream data to millions of devices like Smart TV's/Android Phones they get pretty hot. OpenBMC OS provides a way to monitor such things so that a person can get the required data sitting at home comfortably. This is just one use case. But we were motivated to create a device for which the hardware is cheaply available and also leverage Open Source software.

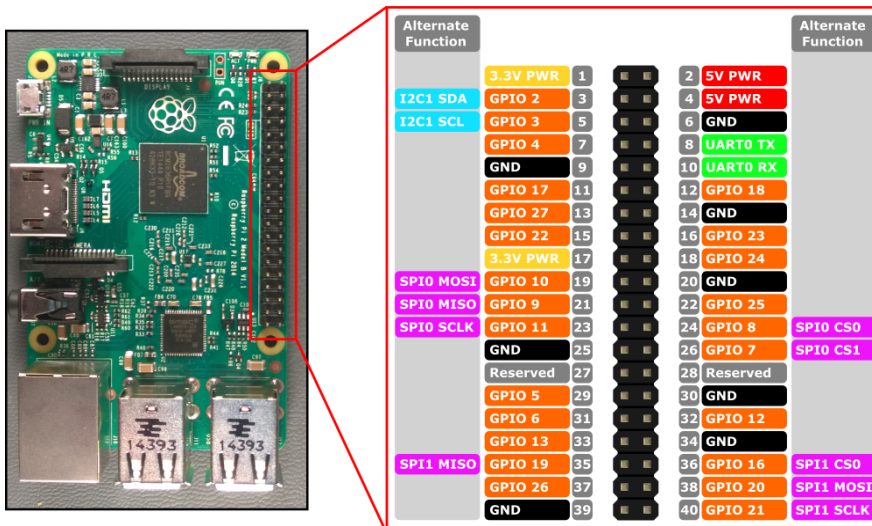
I am using the Raspberry Pi 3 model B here and Texas Instruments INA219 sensor here for the purpose and OpenBMC Dunfell branch. I was able to Design/Develop/Test a successful working prototype of the device under the constant supervision of my Supervisor. Though i have faced a lot of issues during the work, i was able to overcome the same with the help of my seniors and Mentors in the Organization.

MAIN TEXT

MODULES IN OBMC MULTIMETER

The main modules in OBMC MULTIMETER are Raspberry Pi 3 model B and Texas Instruments INA219 Current/Voltage/Power measurement sensor. On the Software side i am using OpenBMC Yocto Dunfell Branch.

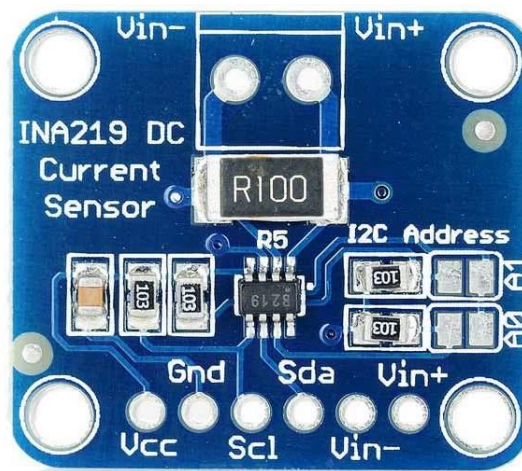
(a) Raspberry Pi 3 Model B :



Raspberry Pi 3 Model B is the earliest model of the third-generation Raspberry Pi. It replaced Raspberry Pi 2 Model B in February 2016, the latest product in the Raspberry Pi 3 range.

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 100 Base Ethernet
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

(b) Texas Instruments INA219

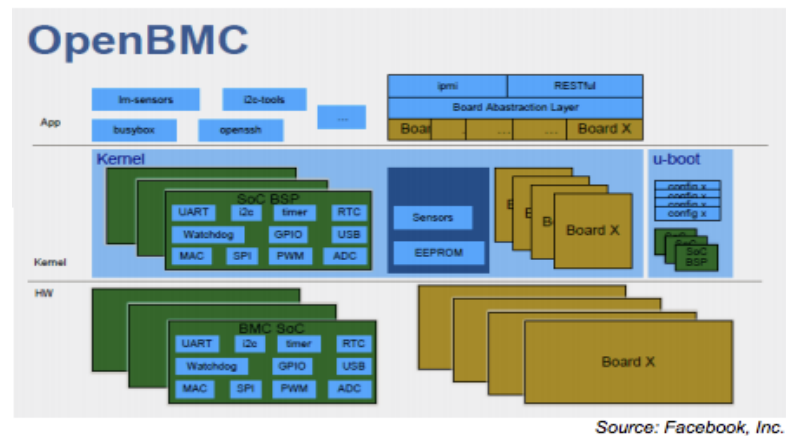


The INA219 is a current shunt and power monitor with an I2C- or SMBUS-compatible interface. The device monitors both shunt voltage drop and bus supply voltage, with programmable conversion times and filtering. A programmable calibration value, combined with an internal multiplier, enables direct readouts of current in amperes. An additional multiplying register calculates power in watts. The I2C- or SMBUS-compatible interface features 16 programmable addresses. The INA219 is available in two grades: A and B. The B grade version has higher accuracy and higher precision specifications. The INA219 senses across shunts on buses that can vary from 0 to 26 V. The device uses a single 3- to 5.5-V supply, drawing a maximum of 1 mA of supply current. The INA219 operates from -40°C to 125°C .

Pin Configuration and Functions for INA219 :

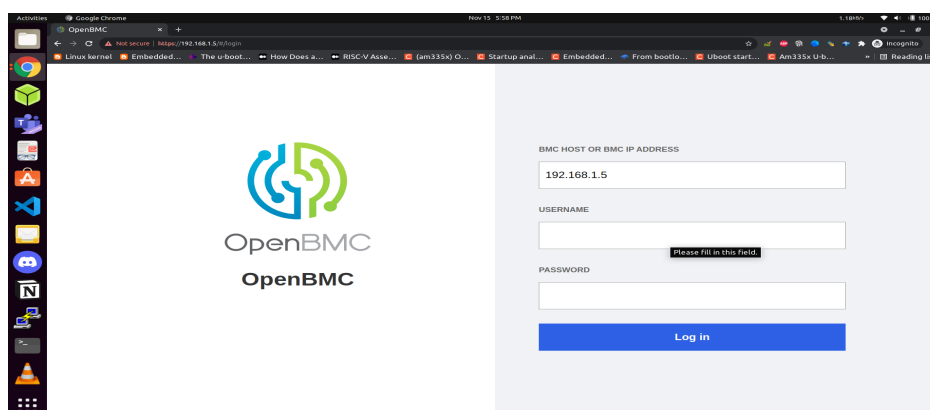
Pin Functions				
NAME	PIN		I/O	DESCRIPTION
	SOT-23	SOIC		
IN+	1	8	Analog Input	Positive differential shunt voltage. Connect to positive side of shunt resistor.
IN-	2	7	Analog Input	Negative differential shunt voltage. Connect to negative side of shunt resistor. Bus voltage is measured from this pin to ground.
GND	3	6	Analog	Ground
V _S	4	5	Analog	Power supply, 3 to 5.5 V
SCL	5	4	Digital Input	Serial bus clock line
SDA	6	3	Digital I/O	Serial bus data line
A0	7	2	Digital Input	Address pin. Table 1 shows pin settings and corresponding addresses.
A1	8	1	Digital Input	Address pin. Table 1 shows pin settings and corresponding addresses.

(c) OpenBMC :



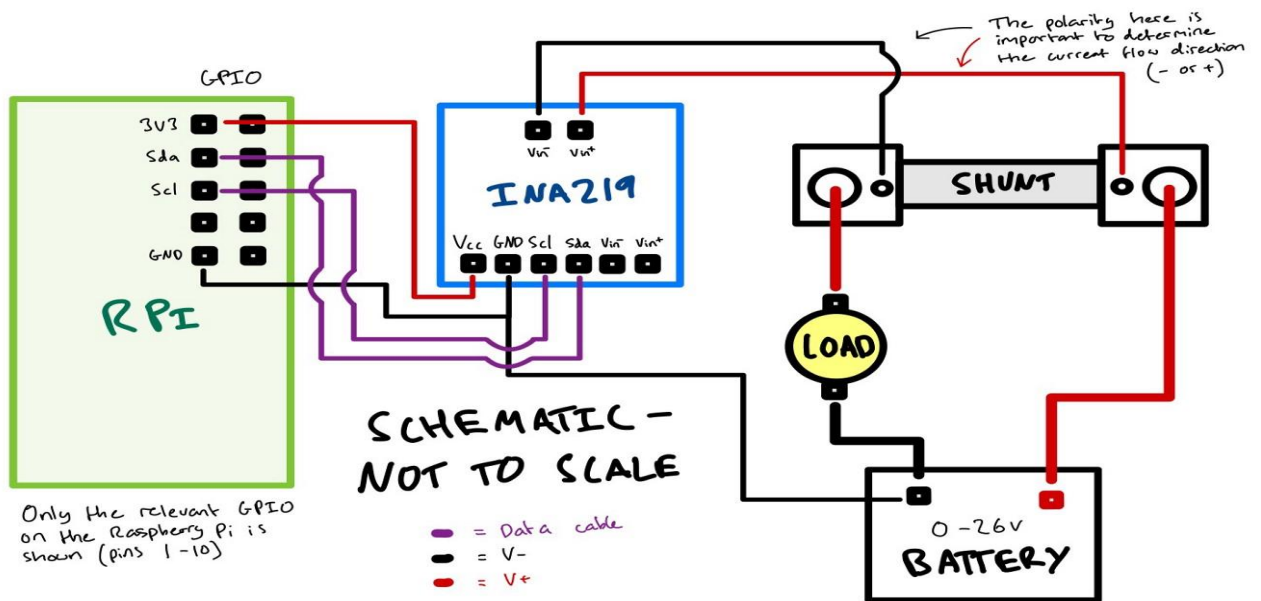
Source: Facebook, Inc.

The OpenBMC project is a Linux Foundation collaborative open-source project whose goal is to produce an open source implementation of the Baseboard Management Controllers (BMC) Firmware Stack. OpenBMC is a Linux distribution for BMCs meant to work across heterogeneous systems that include enterprise, high-performance computing (HPC), telecommunications, and cloud-scale data centers. OpenBMC uses the Yocto Project as the underlying building and distribution generation framework. OpenBMC uses D-Bus as an inter-process communication (IPC). OpenBMC includes a web application for interacting with the firmware stack. OpenBMC added Redfish support for hardware management. Founding organizations of the OpenBMC project are Microsoft, Intel, IBM, Google, and Facebook. OpenBMC WEBUI :



FUNCTIONAL BLOCK DIAGRAM/HARDWARE CONNECTIONS/APPROACH

The functional block diagram for OBMC Multimeter is fairly simple and illustrated below:



a) Figure : INA219 Connections with Raspberry Pi and load to test

INA219 sensor supports I2C protocol and Raspberry Pi 3 has I2C1 bus which is used to interface the sensor. This required us to enable the I2C interface in the OpenBMC for the Pi. Now for the OS to correctly discover and initialize the INA219 sensor, we require 2 things- A Linux kernel Driver for the sensor and a way for Pi to discover the sensor while booting. For this we created a device tree overlay (.dts) for INA219 with correct interfacing of the same to I2C. It was made sure that the compatible property in the device tree overlay and the driver code was matching for the OS to load the driver correctly for INA219 while booting. Final step was to select the driver for INA219 in the menuconfig for Raspberry Pi 3 kernel. Some modifications were made in the OpenBMC software stack to integrate all of these planned steps so that we get a final device which can perform the work it is supposed to do.

SOFTWARE SETUP/CONFIGURE/BUILD/FLASH

Building a working OpenBMC image for Raspberry Pi 3 model B.

This required me to get the latest OpenBMC source code from the github repo. Since OpenBMC is based on the Yocto project, the bitbake commands commonly used for building Yocto images were used here. Some custom configurations were added in the build for adding the following configuration in build/conf/local.conf for enabling wifi, phosphor-webui, redfish ipmitool and enabling i2c interface for ina219 sensor and UART. During the build process some errors were encountered and they fixed them. Finally generated vanilla OpenBMC image for raspberry pi 3 model was flashed to the sdcard using balena etcher tool and the booting was successful. Following steps were performed on Ubuntu 20.04:

i) Getting the essential libraries and dependencies needed to build OpenBMC were installed by this command:

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc multilib
```

```
build-essential chrpath socat libssl-dev xterm
```

ii) Cloning the OpenBMC github repo to local build machine:

```
$ git clone https://github.com/openbmc/openbmc.git
```

iii) Auto generating the local.conf and bblayers.conf configuration file for building image for Raspberry Pi 3 board. Configuration files bblayers.conf and local.conf will be generated in the build directory :

```
$ cd openbmc/
```

```
$ TEMPLATECONF=meta-evb/meta-evb-raspberrypi/conf . openbmc-env
```

iv) Adding the following configuration in **build/conf/local.conf** for enabling wifi, phosphor-webui, redfish ipmitool and enabling i2c interface for ina219 sensor and UART

```
GPU_MEM = "16"
```

```
DISTRO_FEATURES:append = " systemd"
```

```
VIRTUAL-RUNTIME_init_manager = "systemd"
```

DISTRO_FEATURES_BACKFILL_CONSIDERED = "sysvinit" 8

VIRTUAL-RUNTIME_initscripts = ""

IMAGE_FSTYPES += " rpi-sdimg" // raspberry pi bootable filesystem type

ENABLE_I2C = "1" // enable i2c bus on the pi

KERNEL_MODULE_AUTOLOAD:rpi += "i2c-dev i2c-bcm2708" // loading the necessary i2c drivers

ENABLE_UART = "1" // enable UART for serial debugging

IMAGE_INSTALL:append="vim ipmitool wpa-supPLICANT phosphor-webui" // necessary packages installation

KERNEL_DEVICETREE:append="overlays/ina219.dtbo" // append device tree blob for ina219 to the obmc image.

```
GPU_MEM = "16"
DISTRO_FEATURES:append = " systemd"
VIRTUAL-RUNTIME_init_manager = "systemd"
DISTRO_FEATURES_BACKFILL_CONSIDERED = "sysvinit"
VIRTUAL-RUNTIME_initscripts = ""
IMAGE_FSTYPES += " rpi-sdimg"
ENABLE_I2C = "1"
KERNEL_MODULE_AUTOLOAD:rpi += "i2c-dev i2c-bcm2708"

IMAGE_INSTALL:append = " vim ipmitool wpa-supPLICANT webui-vue"

KERNEL_DEVICETREE:append = " overlays/mpu6050.dtbo overlays/ina219.dtbo"
```

v) Building the OpenBMC image :

\$ bitbake obmc-phosphor-image

The final generated image “**obmc-phosphor-image-raspberrypi3.rpi-sdimg**” will be on the path “**openbmc/build/tmp/deploy/images/raspberrypi3**”. Flash the same onto a sdcard using balena etcher tool and boot the Pi 3 till you get the root shell.

```
pratik@compute-ci:/media/pratik/openbmc/build$ bitbake obmc-phosphor-image
Loading cache: 100% [#####] Time: 0:00:00
Loaded 4076 entries from dependency cache.
Parsing recipes: 100% [#####] Time: 0:00:00
Parsing of 2675 .bb files complete (2674 cached, 1 parsed). 4077 targets, 374 skipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION           = "1.51.1"
BUILD_SYS            = "x86_64-linux"
NATIVELSBSTRING      = "ubuntu-18.04"
TARGET_SYS           = "arm-openbmc-linux-gnueabi"
MACHINE              = "raspberrypi3"
DISTRO                = "openbmc-phosphor"
DISTRO_VERSION        = "nodistro-0"
TUNE_FEATURES         = "arm vfp cortexa7 neon vfpv4 thumb callconvention-hard"
TARGET_FPU            = "hard"
meta
meta-poky
meta-oe
meta-networking
meta-perl
meta-python
meta-webserver
meta-security
meta-phosphor
meta-raspberrypi     = "master:58ac4343a47e69e6c23f7f3128b6da1b9d922a91"
```

Once flashed and booted, connect the pi to wlan by modifying the /etc/wpa_supplicant.conf file and adding the following :

```
ctrl_interface=/var/run/wpa_supplicant
```

```
ctrl_interface_group=0
```

```
update_config=1
```

```
network={
```

```
ssid="yourHiddenSSID"
```

```
scan_ssid=1
```

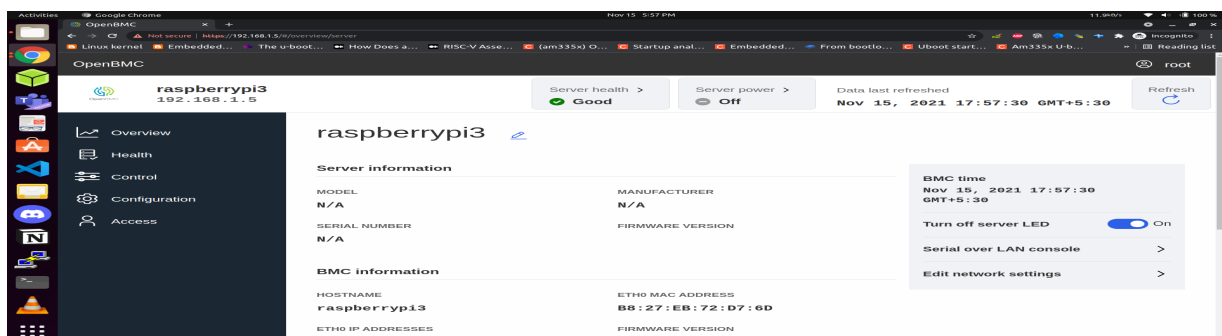
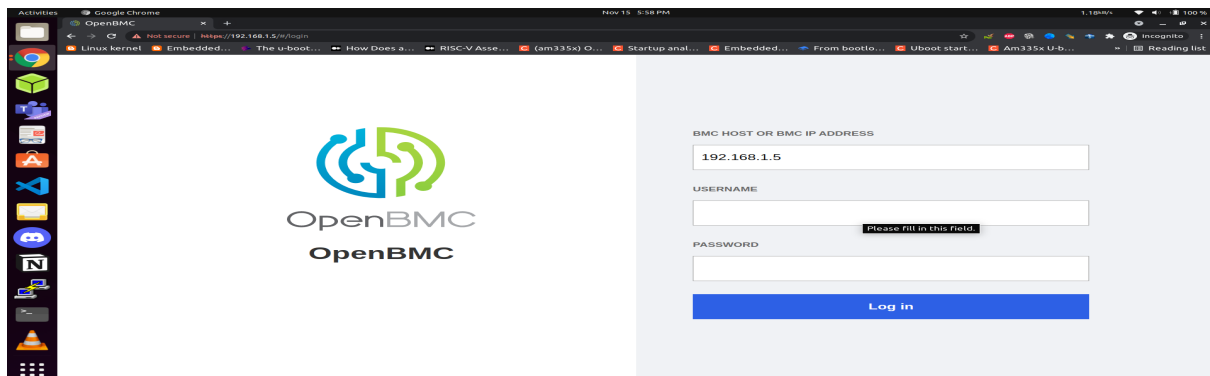
```
psk=Your_wifi_password
```

```
}
```

And then, connect to internet using the following command

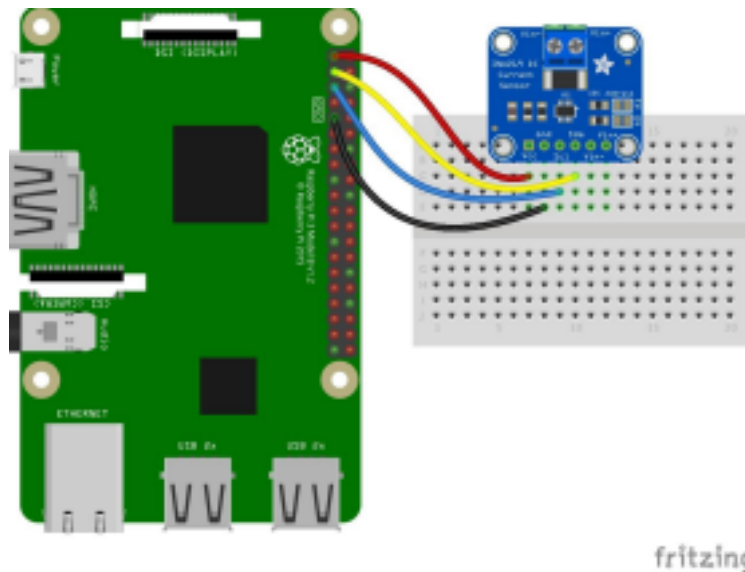
```
$ wpa_supplicant -iwlan0 -c /etc/wpa_supplicant.conf
```

If wlan is not available, you can still connect ethernet to the pi to get an IP address via dhcp. Using `$ifconfig` note down the IP address of the pi and in the browser, open the URL - <https://<ip address of the pi>>. You will be redirected to login page of OpenBMC running on the Pi. Something like this :



INA219 SENSOR DETECTION AND INTEGRATION

Now that we got the OpenBMC build successfully booted on the Raspberry Pi 3, next thing is the INA219 sensor connection, discovery and integration. As we saw from INA219 Specs, the sensor supports I2C protocol connection and Raspberry Pi 3 has I2C1 bus which is available freely to use and connect any device that supports I2C interface. The connection of I2C1 on Pi with INA219 is shown as below :



This is just a hardware connection. But we need a methodology and mechanism to get this sensor working on OpenBMC which is just a modified flavour version of Linux. We do this via what we called a "**Device Tree**" which has it's own syntax and also a compiler to convert this syntax into a blob which is used by Linux while booting to discover the hardware which is not discoverable by Linux like SPI/I2C/PWM/IOMUX buses. We first check if INA219 is getting recognized by the I2C1 on OpenBMC. For this we use i2c-tools installed on the OpenBMC image and use command on the Pi:

```
$ i2cdetect -y 1
```

We get something like 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: 40 -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

Where “40” is the I2C address of the INA219. This means the connection to the sensor is correct. Now create an empty file **ina219.dts** and add the following content in it :

```
/* Definitions for I2C based current sensor INA219 */

/dts-v1/;

/plugin/;

/ {
    compatible = "brcm,bcm2708 brcm,bcm2835";

    fragment@0 {
        target = <&i2c1>;

        __overlay__ {
            #address-cells = <1>;

            #size-cells = <0>;

            ina219@40 {
                status = "okay";

                compatible = "ti,ina219";

                reg = <0x40>;

                shunt-resistor = <100000>; // R100
            };
        };
    };
};
```

On the build machine install the device tree compile using “**sudo apt-get install device-tree-compiler**” and compile our dts with the following command:

```
$ dtc -@ -I dts -O dtb -o ina219.dtbo ina219.dts
```


We will have generated a file named “**ina219.dtbo**” with the following command which we need to copy into the sdcard in “**/boot/overlays**” folder. Then add the following lines in “/boot/config.txt” file :

dtoverlay=ina219

Thus, after booting the Pi we can see the driver getting loaded by dmesg command as seen in the following pic :

```
root@raspberrypi3:~# dmesg | grep ina
[ 3.067624] hid: raw HID events driver (C) Jiri Kosina
[ 3.125239] ina2xx_1-0040: power monitor ina219 (Rshunt = 100000 uOhm)
```

This is how you know, that the sensor is discovered by the driver, initialized and integrated correctly. You can also see the corresponding sysfs entries in the /sys/class path to confirm :

```
root@raspberrypi3:/sys/class/hwmon/hwmon0# ls
curr1_input  in0_input    name         power        shunt_resistor uevent
device       in1_input    of_node      power1_input subsystem
```

Where,

in0_input	Shunt voltage(mV) channel
in1_input	Bus Voltage(mV) channel
curr1_input	Current(mA) measurement channel
power1_input	Power(uW) measurement channel
shunt_resistor	Shunt resistance(uOhm) channel

These sysfs entries by the driver are the ones which will give us the Voltage, Current and Power values of the connected load thus acting as a multimeter device. Note that these values are in the terms of milliVolts, milliAmperes, microWatts and microOhms.

So these values have to be multiplied by 1000 to get the actual raw values. The system will automatically convert it during calculation. Next thing would be to populate these values onto the phosphor webui. This will be explained in the next section. So the default sensor information on our OpenBMC image will be :

Sensors (2)	Low critical	Low warning	Current	High warning	High critical
Utilization CPU			0.1295703098615081	80	90
Utilization Memory			12.458672334920834	80	85

MAPPING THE INA219 SENSOR VALUES TO PHOSPHOR WEB-UI

Now, that the Linux Device Driver has initialized the sensor and sysfs entries are created, we will further look on how we can populate and correctly map these values on the webui. OpenBMC provides a method to do the same. One OpenBMC utility called **phosphor-hwmon-readd** is used to continuously read the sensor configuration file which we will create soon and reads the sysfs entries to give us the current/voltage/power values of the connected load on the web-ui. Create a configuration file called **ina219@40.conf** under the OpenBMC build path :

`openbmc/meta-raspberrypi/recipes-phosphor/sensors/phosphor-hwmon/obmc/hwmon/soc/i2c@7e804000/ina219@40.conf` and add the following content :

```
LABEL_curr1 = "INA219"
```

```
LABEL_power1 = "INA219"
```

```
LABEL_in1 = "INA219"
```

Also create file named **phosphor-hwmon_%.bbappend** with the following content :

```
FILESEXTRAPATHS:prepend := "${THISDIR}/${PN}:"
```

```
CHIPS = "i2c@7e804000/ina219@40"
```

```
ITEMSFMT = "soc/{0}.conf"
```

```
ITEMS = "${@compose_list(d, 'ITEMSFMT', 'CHIPS')}
```

```
ENVS = "obmc/hwmon/{0}"
```

```
SYSTEMD_ENVIRONMENT_FILE:${PN} += "${@compose_list(d, 'ENVS',
```

```
'ITEMS'))}" 14
```


Clean and Recompile the image using the following command :

```
$ bitbake obmc-phosphor-image -c clean && bitbake obmc-phosphor-image
```

Flash the new compiled image onto Raspberry Pi and boot the board. Login to the webui and go to sensors sub-section in health section. This time we can see our sensor correctly recognized, initialized and installed in the webui and that's what we intended!

Sensor values Without Load connected :

OpenBMC



raspberrypi3

192.168.0.104

Server health >
Good

Server power >
Off

Data last refreshed

Mar 13, 2022 23:57:14 GMT+5:30

Refresh

Overview

Health

Event log

Hardware status

Sensors

Control

Configuration

Access

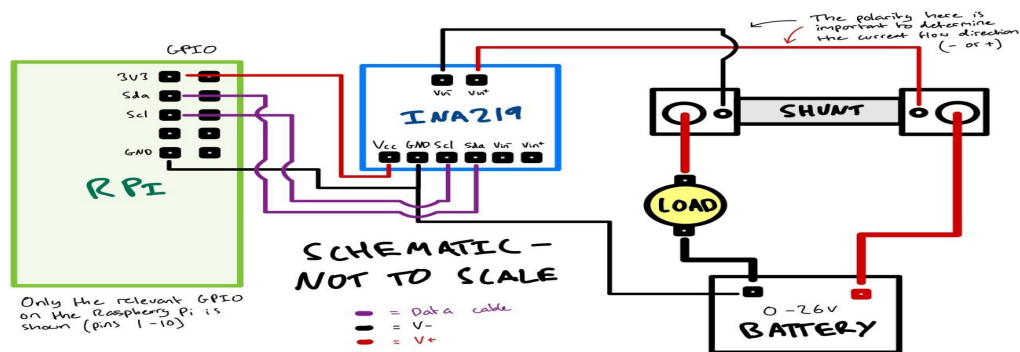
FILTER BY SEVERITY

AllCriticalWarningNormal

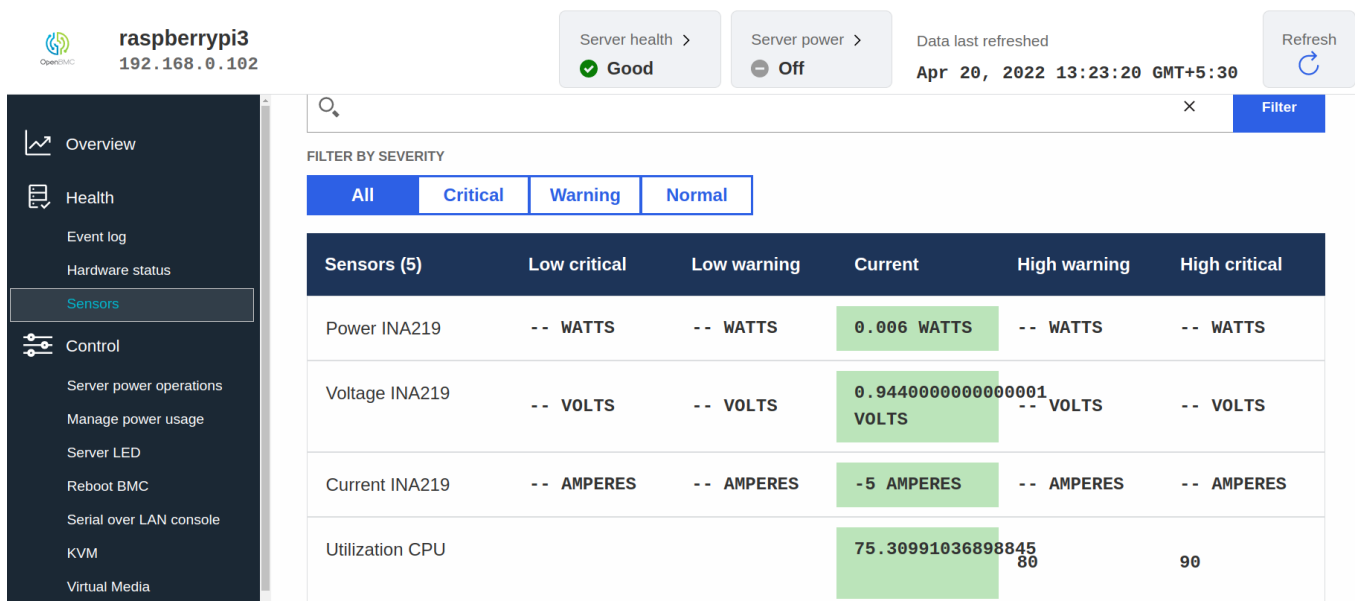
Sensors (5)	Low critical	Low warning	Current	High warning	High critical
Power INA219	-- WATTS	-- WATTS	0 WATTS	-- WATTS	-- WATTS
Voltage INA219	-- VOLTS	-- VOLTS	0.884 VOLTS	-- VOLTS	-- VOLTS
Current INA219	-- AMPERES	-- AMPERES	0 AMPERES	-- AMPERES	-- AMPERES
Utilization CPU			0.36181619476688914		90
Utilization Memory			12.12357907845244		85

TESTING OF THE PRODUCT AND CONCLUSION

Now that the sensor values are available on the Web-ui, we are ready to do some validation and testing. For the test purpose, i have chosen to connect simple loads across the sensor. Different colour LED's with different Current/Voltage/Power draw values were chosen. The connection of loads is illustrated in the diagram down below :



The INA219 sensor has built-in shunt resistor. The Load/LED has to be connected where we see the yellow circle in the diagram. For the test purpose, i have connected the blue LED and the result are as follows :



Hence we can see that the power drawn by the LED is getting reflected on the Web-UI and hence we conclude the same.

REFERENCES

- [1] Raspberry Pi 3: Complete Programming Guide With Step by Step
Raspberry Pi 3 Projects for Beginners: Complete Programming Guide with
Step by Step ... Beginners
- [2] Exploring Raspberry Pi: Interfacing Real World with Embedded Linux
- [3] The I2C Bus: From Theory to Practice
- [4] ina219 datasheet from Texas instruments
- [5] <https://www.kernel.org/doc/html/v4.14/driver-api/i2c.html>
- [6] <https://www.kernel.org/doc/html/v5.12/hwmon/ina2xx.html>
- [7] <https://github.com/openbmc/openbmc>

GLOSSARY

Terminologies used in the final Report :

OBMC	OpenBMC
DTC	Device Tree Compiler
I2C	Inter-Integrated Circuit
OS	Operating System
IOT	Internet of Things

Checklist of Items for the Final Dissertation / Project / Project Work Report

This checklist is to be attached as the last page of the final report.

This checklist is to be duly completed, verified and signed by the student.

1.	Is the final report neatly formatted with all the elements required for a technical Report?	Yes
2.	Is the Cover page in proper format as given in Annexure A?	Yes
3.	Is the Title page (Inner cover page) in proper format?	Yes
4.	(a) Is the Certificate from the Supervisor in proper format? (b) Has it been signed by the Supervisor?	Yes Yes
5.	Is the Abstract included in the report properly written within one page? Have the technical keywords been specified properly?	Yes Yes
6.	Is the title of your report appropriate? The title should be adequately descriptive, precise and must reflect scope of the actual work done. Uncommon abbreviations / Acronyms should not be used in the title	Yes
7.	Have you included the List of abbreviations / Acronyms?	Yes
8.	Does the Report contain a summary of the literature survey?	Yes
9.	Does the Table of Contents include page numbers? (i). Are the Pages numbered properly? (Ch. 1 should start on Page # 1) (ii). Are the Figures numbered properly? (Figure Numbers and Figure Titles should be at the bottom of the figures) (iii). Are the Tables numbered properly? (Table Numbers and Table Titles should be at the top of the tables) (iv). Are the Captions for the Figures and Tables proper? (v). Are the Appendices numbered properly? Are their titles appropriate	Yes Yes Yes Yes Yes Yes
10.	Is the conclusion of the Report based on discussion of the work?	Yes
11.	Are References or Bibliography given at the end of the Report? Have the References been cited properly inside the text of the Report? Are all the references cited in the body of the report	Yes Yes Yes
12.	Is the report format and content according to the guidelines? The report should not be a mere printout of a Power Point Presentation, or a user manual. Source code of software need not be included in the report.	Yes

Declaration by Student:

I certify that I have properly verified all the items in this checklist and ensure that the report is in proper format as specified in the course handout.

Place: Nagpur

Date: 20/04/1994

Name: Pratik Arun Farkase

ID No.: 2020MT12443

Signature of the Student : 