TIZEN BASED REMOTE CONTROLLER CAR USING RASPBERRY PI2

Pintu Kumar (pintu.k@samsung.com, pintu_agarwal@yahoo.com)
Samsung Research India – Bangalore : Tizen Kernel/BSP Team
INTRODUCTION

RASPBERRY PI2 OVERVIEW

TIZEN OVERVIEW

HARDWARE & SOFTWARE REQUIREMENTS

SOFTWARE CUSTOMIZATION

SOFTWARE SETUP & INTERFACING

HARDWARE INTERFACING & CONNECTIONS

ROBOT CONTROL MECHANISM

SOME RESULTS

CONCLUSION

REFERENCES
This talk is about designing a remote controller robot (toy car) using the raspberry pi2 hardware, pi2 Linux Kernel and Tizen OS as platform.

In this presentation, first we will see how to replace and boot Tizen OS on Raspberry Pi using the pre-built Tizen images. Then we will see how to setup Bluetooth, Wi-Fi on Tizen and finally see how to control a robot remotely using Tizen smart phone application.
RASPBERRY PI2 - OVERVIEW

- 40 Pin GPIO Headers
- DSI Display Connector
- SD Card Slot (Back side of the board)
- 1 GB RAM BCM2836 900MHZ
- 5V Micro USB
- HDMI Port
- CSI Camera Connector
- 4 USB Ports
- Ethernet
- Audio/Video Jack
Raspberry Pi2 Features

- Broadcom BCM2836 900MHz Quad Core ARM Cortex-A7 CPU
- 1GB RAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot
- Video Core IV 3D graphics core
<table>
<thead>
<tr>
<th>GPIO#</th>
<th>2nd func</th>
<th>2nd func</th>
<th>GPIO#</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>+3V3</td>
<td>+5V</td>
<td>N/A</td>
</tr>
<tr>
<td>GPIO2</td>
<td>SDA1 (I2C)</td>
<td>+5V</td>
<td>N/A</td>
</tr>
<tr>
<td>GPIO3</td>
<td>SCL1 (I2C)</td>
<td>GND</td>
<td>N/A</td>
</tr>
<tr>
<td>GPIO4</td>
<td>GCLK</td>
<td>TXD0 (UART)</td>
<td>GPIO14</td>
</tr>
<tr>
<td></td>
<td>GND</td>
<td>RXD0 (UART)</td>
<td>GPIO15</td>
</tr>
<tr>
<td>GPIO17</td>
<td>GEN0</td>
<td>GEN1</td>
<td>GPIO18</td>
</tr>
<tr>
<td>GPIO27</td>
<td>GEN2</td>
<td>GEN1</td>
<td>N/A</td>
</tr>
<tr>
<td>GPIO22</td>
<td>GEN3</td>
<td>GEN4</td>
<td>GPIO23</td>
</tr>
<tr>
<td></td>
<td>+3V3</td>
<td>GEN5</td>
<td>GPIO24</td>
</tr>
<tr>
<td>GPIO10</td>
<td>MOSI (SPI)</td>
<td>GND</td>
<td>N/A</td>
</tr>
<tr>
<td>GPIO9</td>
<td>MISO (SPI)</td>
<td>GEN6</td>
<td>GPIO25</td>
</tr>
<tr>
<td>GPIO11</td>
<td>SCLK (SPI)</td>
<td>CEO_N (SPI)</td>
<td>GPIO8</td>
</tr>
<tr>
<td>N/A</td>
<td>GEN5</td>
<td>CEO1_N (SPI)</td>
<td>GPIO10</td>
</tr>
<tr>
<td>EEPROM</td>
<td>ID_SD</td>
<td>ID_SC</td>
<td>EEPROM</td>
</tr>
<tr>
<td>GPIO5</td>
<td>N/A</td>
<td>GND</td>
<td>N/A</td>
</tr>
<tr>
<td>GPIO6</td>
<td>N/A</td>
<td>-</td>
<td>GPIO12</td>
</tr>
<tr>
<td>GPIO13</td>
<td>N/A</td>
<td>GND</td>
<td>N/A</td>
</tr>
<tr>
<td>GPIO19</td>
<td>N/A</td>
<td>N/A</td>
<td>GPIO16</td>
</tr>
<tr>
<td>GPIO26</td>
<td>N/A</td>
<td>N/A</td>
<td>GPIO20</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>N/A</td>
<td>GPIO21</td>
</tr>
</tbody>
</table>
**TIZEN Profiles**

- TIZEN is the OS of everything.
- Tizen is a multi-device OS which can support many types of profiles.
- The current profile that are supported are:
  - Mobile
  - Wearable
  - IVI
  - Common
- The new profiles can be easily derived using the minimal common profile.
TIZEN Features

- Tizen is truly open source. Almost all components are based on open source packages.
- Uses mainline Linux Kernel
- Uses systemd for booting
- Uses dbus for IPC communication
- Uses DRM/X11/Wayland for Display & Graphics
- Uses Gstreamer for multimedia framework
- Uses SMACK for platform security
- Uses EFL (Enlightenment Foundation Libraries) for UI framework
- Provides SDB (Smart Development Bridge) for developers.
- Uses HTML5 for WebApps development
- And many more....
HARDWARE COMPONENTS

- Raspberry pi 2 hardware
- Linux PC – Ubuntu 14.04
- Micro SD Card (8 GB)
- Robot Chassis platform (with 2 DC motors, 2 wheels, 1 Castor wheels)
- L293D Driver Board (1 number)
- USB Power Bank (1 number)
- AA size batteries (8 numbers, 12V)
- Battery holder/case (1 number)
- Wi-Fi USB Dongle (1 number)
- Bluetooth USB dongle (1 number)
- USB Web Cam (1 number)
- A Monitor Screen (for Display purpose)
- HDMI Cable (1 number)
- USB keyboard & mouse
- A Tizen Smart Phone with Tizen 2.4
- Screws, Blots, Spacer, jumper wires etc.
SOFTWARE COMPONENTS

- Raspberry Pi – NOOBS image
- Tizen 3.0 common pre-built images (alternatively Tizen pi2 pre-built image).
- Raspberry Pi Linux Kernel 4.1.16
- GCC ARM tool chain (arm-linux-gnueabihf-gcc)
- Tizen Yocto setup (Or, Tizen GBS Build setup)
- Tizen 2.4 SDK software
- Ubuntu 14.04
• Download Raspberry pi software from:
• Extract it and install it on the SD card.
• Boot the raspberry pi using this SD card.
• Install the Raspbian OS and boot it till desktop.
• At this time verify that all functionalities are working fine on Raspberry pi image.
• Download Tizen images from:
  – https://download.tizen.org/
• Choose any one type of image from the below repo.

<table>
<thead>
<tr>
<th>Name</th>
<th>Last_modified</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>docker/</td>
<td>2015-09-11 04:36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iris/</td>
<td>2014-09-03 03:31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lecture/</td>
<td>2015-07-19 23:02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>live/</td>
<td>2015-04-13 13:33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>misc/</td>
<td>2015-03-04 14:35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>prerelease/</td>
<td>2014-07-02 13:31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>releases/</td>
<td>2015-10-28 01:19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sdk/</td>
<td>2016-02-03 09:12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>services/</td>
<td>2015-04-29 11:41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>snapshots/</td>
<td>2015-11-26 22:22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tct/</td>
<td>2015-10-28 05:35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tools/</td>
<td>2015-09-02 06:48</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• If you want to try latest release mobile profile, you can use this:
  – https://download.tizen.org/releases/2.4/2.4-mobile/tizen-2.4-mobile_20151030.1/images/
• If you want to use common profile, you can use this:
  – https://download.tizen.org/snapshots/tizen/common/latest/images/
**TIZEN PI2 Images**

- **Tizen raspberry pi 2 pre-built images:**
  - [https://files.s-osg.org/tizen-on-rpi2/](https://files.s-osg.org/tizen-on-rpi2/)

---

### Index of /tizen-on-rpi2

<table>
<thead>
<tr>
<th>Name</th>
<th>Last_modified</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Directory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>README.txt</td>
<td>12-Aug-2015 10:42</td>
<td>1.2K</td>
</tr>
<tr>
<td>bblayers.conf</td>
<td>18-Aug-2015 02:24</td>
<td>1.3K</td>
</tr>
<tr>
<td>local.conf</td>
<td>06-Aug-2015 15:41</td>
<td>4.5K</td>
</tr>
<tr>
<td>local.conf.sd_accel_vc</td>
<td>03-Sep-2015 07:10</td>
<td>3.3K</td>
</tr>
<tr>
<td>md5sum.txt</td>
<td>12-Aug-2015 10:58</td>
<td>57</td>
</tr>
<tr>
<td>rpm/</td>
<td>14-Aug-2015 10:24</td>
<td>-</td>
</tr>
<tr>
<td>tizen-common-core-image-crosswalk-dev-raspberrypi2-20150806223520_rootfs.rpi-sdimg</td>
<td>06-Aug-2015 06:18</td>
<td>1.2G</td>
</tr>
<tr>
<td>tizen-common-core-image-crosswalk-dev-raspberrypi2.rpi-sdimg-2015-08-06</td>
<td>06-Aug-2015 18:18</td>
<td>1.2G</td>
</tr>
<tr>
<td>tizen-common-core-image-crosswalk-dev.sdimg-2015-08-06</td>
<td>06-Aug-2015 16:33</td>
<td>1.2G</td>
</tr>
<tr>
<td>tizen.rpi-sdimg.2015-04-22</td>
<td>22-Apr-2015 14:18</td>
<td>596M</td>
</tr>
<tr>
<td>tizen.rpi-sdimg.2015-07-14</td>
<td>14-Jul-2015 08:13</td>
<td>620M</td>
</tr>
<tr>
<td>tizen.rpi-sdimg.2015-08-04</td>
<td>04-Aug-2015 06:22</td>
<td>856M</td>
</tr>
<tr>
<td>tizen.rpi-sdimg.2015-08-12</td>
<td>12-Aug-2015 10:35</td>
<td>728M</td>
</tr>
<tr>
<td>tizen.rpi-sdimg.LATEST</td>
<td>12-Aug-2015 10:35</td>
<td>728M</td>
</tr>
</tbody>
</table>
TIZEN Build Setup

• GBS Build System:

• YOCTO Build System:
  – https://wiki.tizen.org/wiki/Build_Tizen_with_Yocto_Project
  – https://wiki.tizen.org/wiki/Tizen_on_Yocto_Project

Tizen 3.0 common pre-built rpms:

https://download.tizen.org/snapshots/tizen/common/latest/repos/arm-wayland/packages/armv7l/
• Extract the Tizen common image on the Linux PC. It will contain 3 images:
  – rootfs.img (root file system)
  – system-data.img (system partition: /opt)
  – user.img (user partition: /opt/usr)

• Now, check the size of each image using the “du –h” command.
  – # du –h *.img
    • 864M  rootfs.img
    • 49M   system-data.img
    • 97M   user.img
• Use Gparted on Ubuntu to create new partitions for Tizen images, on the SD card.
• First erase the raspberry pi OS root partition. Do not disturb the SETTINGS and boot partitions.
• Then create the new partitions as follows:
• Make sure to run `resize2fs` command to resize all the partitions.
  – `sudo resize2fs /dev/sdb7 [rootfs partition]`
  – `sudo resize2fs /dev/sdb8 [system-data partition]`
  – `sudo resize2fs /dev/sdb9 [user partition]`
• Now, use “`dd`” commands in Linux to write the actual Tizen images to the respective partitions on SD card.
  – `sudo dd if=rootfs.img of=/dev/sdb7 bs=4M`
  – `sudo dd if=system-data.img of=/dev/sdb8 bs=4M`
  – `sudo dd if=user.img of=/dev/sdb9 bs=4M`
• Then, remount the SD card on the Linux PC.
• Now, using the “df –h” command, we should be able to see all the partitions as follows:

<table>
<thead>
<tr>
<th>Partition</th>
<th>Size</th>
<th>Used</th>
<th>Free</th>
<th>Usage</th>
<th>Mount Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/sdb6</td>
<td>63M</td>
<td>20M</td>
<td>44M</td>
<td>32%</td>
<td>/media/boot</td>
</tr>
<tr>
<td>/dev/sdb8</td>
<td>196M</td>
<td>17M</td>
<td>180M</td>
<td>9%</td>
<td>/media/system-data</td>
</tr>
<tr>
<td>/dev/sdb5</td>
<td>31M</td>
<td>1.4M</td>
<td>28M</td>
<td>5%</td>
<td>/media/SETTINGS</td>
</tr>
<tr>
<td>/dev/sdb9</td>
<td>5.0G</td>
<td>72M</td>
<td>4.9G</td>
<td>2%</td>
<td>/media/user</td>
</tr>
<tr>
<td>/dev/sdb7</td>
<td>1009M</td>
<td>869M</td>
<td>132M</td>
<td>87%</td>
<td>/media/rootfs</td>
</tr>
</tbody>
</table>

• Now, we need to make Tizen specific changes in raspberry pi kernel and Tizen platform to boot the image successfully on Pi2.
- Download Raspberry Pi Kernel (4.1.16) repo from the following:
  - `git clone --depth=1 git://github.com/raspberrypi/linux`

- Build the kernel:
  - `make ARCH=arm -j8 CROSS_COMPILE=arm-linux-gnueabibcm2709_defconfig`
  - `make ARCH=arm -j8 CROSS_COMPILE=arm-linux-gnueabibcm2709_defconfig`

- Create a new defconfig for Tizen:
  - `# cp -f .config arch/arm/configs/tizen_pi2_defconfig`

- Enable Tizen specific kernel configurations (one by one), using:
  - `make ARCH=arm menuconfig`

- Each time you change the configuration, make sure to sync with the `tizen_pi2_defconfig`. 
• Fortunately, in Raspberry Pi Kernel (bcm2709_defconfig), most of the Tizen configs are already enabled.

• However, we still need to enable few once as below:

```
CONFIG_SECURITYFS=y
CONFIG_SECURITY_SMACK=y
CONFIG_AUDIT=y
CONFIG_DRM=y
CONFIG_MEMCG=y
CONFIG_MEMCG_SWAP=y
CONFIG_ZRAM=y
CONFIG_CGROUP_DEBUG=y
CONFIG_PM_SLEEP=y
CONFIG_PM_AUTOSLEEP=y
```

• After enabling these configs, make sure to sync the .config with the default `tizen_pi2_defconfig` again.
• Build the final Kernel image:
  – make ARCH=arm -j8 CROSS_COMPILE=arm-linux-gnueabibizen_pi2_defconfig
  – make ARCH=arm -j8 CROSS_COMPILE=arm-linux-gnueabi-
• Generate the device tree image:
  – ./scripts/mkknlimg arch/arm/boot/zImage kernel7.img
• Copy the kernel images to the SD card boot partition.
  – cp -f kernel7.img /media/boot/
  – cp -f arch/arm/boot/dts/bcm2709-rpi-2-b.dtb /media/boot/
  – cp -f arch/arm/boot/dts/overlays/* .dtb
    /media/boot/overlays/
• Install the modules built by kernel:
  – make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf -j8
    INSTALL_MOD_PATH=../modules modules_install

• Copy the modules & firmware to SD card rootfs folder:
  – sudo cp -rf modules/lib/modules/4.1.16-v7+
    /media/rootfs/lib/modules/
  – sudo cp -rf modules/lib/firmware/*
    /media/rootfs/lib/firmware/

• Copy the original pi2 firmware from NOOBs root folder
to the SD card Tizen rootfs folder:
  – sudo cp -rf <pi2 noobs root>/lib/firmware/*
    /media/rootfs/lib/firmware/
• Update rootfs device node under Kernel command line in SD card partition: /media/boot/cmdline.txt
  – dwc_otg.lpm_enable=0 console=ttyAMA0,115200
  console=tty1 root=/dev/mmcblk0p7 rootfstype=ext4
  elevator=deadline fsck.repair=yes rootwait

• Tizen uses systemd services for booting, so customize systemd services as per your needs, under:
  – /usr/lib/systemd/system/*

• For example to boot the system till command prompt, you can set the default.target to multi-user.target

• At this time, you can also perform the clean up of unnecessary services by simply deleting it and removing it dependencies.
• In Tizen, the getty and console services are disabled by default.
• We need to enable these services to get the login prompt on the terminal.
• To enable these services we need to modify the following file:
  – /media/rootfs/usr/lib/systemd/system-preset/90-systemd.preset
    • enable console-getty.service
    • enable console-shell.service
• To get login prompt on Virtual Terminal (tty1), we need to create tty1 service file:
  – cd /media/rootfs/usr/lib/systemd/system/multi-user.target.wants
  – sudo ln –s ..;/getty@.service getty@tty1.service
- Plug the SD card on the PI2 hardware.
- Plug other required peripherals as shown below and power on the raspberry pi.
• You will be able to see the console messages flowing on the monitor screen.
• If Tizen platform images are mounted successfully, you will be able to see the following on the terminal.

Detected architecture 'arm'.

Welcome to Tizen 3.0.0 (Tizen3/Common)!

No hostname configured.  
Set hostname to <localhost>.

localhost login:  

Welcome to Tizen  
root@localhost:~#  

User name: {root, guest}  
Password: tizen

Note:  
User name and password can be found from .ks file.  
tizen-common_20160315.2_common-wayland-3parts-armv7l-odroidu3.ks
In raspberry pi, camera interfacing can be done in 2 ways:

- Using CSI Camera Slot
- Using the USB Web Cam

CSI Camera:

- In CSI camera slot we can directly plug a raspberry pi 5MP Camera module using the ribbon cable.
- It is directly controlled by GPU and it is faster. But it requires around 128MB of system RAM reserved memory.
- However, we can convert this memory to CMA if memory saving is important.

• **USB Web Camera:**
  – In one of the USB slot plug a webcam (Logitech Webcam)
  – It will create a node : /dev/video0 through which we can access it using V4L2 calls.
  – It does not need any reserved memory but the processing could be little slower compared to CSI camera.
  – Using USB you can connect any number of web cams.
• On Tizen, we can perform camera capture using the following:
  – Using the standard **Gstreamer** commands
  – Using a simple **V4L2** application
  – Using the **mm_test_suite** for the Tizen source code repo.
  – Using **launch_cam.sh** (only for web cam) [For Tizen 3.0]

• **Gstreamer** command for single frame capture:
  – `gst-launch-1.0 v4l2src num-buffers=1 ! video/x-raw, format=I420, width=640, height=480, framerate=30/1 ! filesink location=/opt/usr/media/file.yuv`

• **Other sample applications are available under Tizen source:**
  – [https://review.tizen.org/git/?p=platform/core/api/camera.git;a=tree](https://review.tizen.org/git/?p=platform/core/api/camera.git;a=tree)

• **MM Camcoder test suite is available under:**
  – [https://review.tizen.org/git/?p=framework/multimedia/libmm-camcorder.git;a=tree;f=test;h=49ee5fc53d4fcb99e37a1cd5c554f0c95974f365;hb=HEAD](https://review.tizen.org/git/?p=framework/multimedia/libmm-camcorder.git;a=tree;f=test;h=49ee5fc53d4fcb99e37a1cd5c554f0c95974f365;hb=HEAD)
• Tizen uses DRM (Direct Rendering Manager) & X11/Wayland based display system.
• Both DSI display connector or HDMI display interface can be used.
• The DRM support for Raspberry Pi graphics controller VC4 is already available from Linux Kernel 4.1.16.
  – Linux/driver/gpu/drm/vc4/...
• Tizen graphics port for Raspberry Pi is already available as part of Tizen Yocto project repo for PI2.
Bluetooth Setup

- To setup Bluetooth on Tizen, insert the Bluetooth USB Dongle and perform the following steps:
  a) `root@localhost:~# hciconfig hci0 up`
  b) `root@localhost:~# bluetoothctl`
  c) `[bluetooth]# power on`
  d) `[bluetooth]# agent on`
  e) `[bluetooth]# scan on`
  f) `[bluetooth]# pair <scanned device MAC_ID>`
  g) `[bluetooth]# connect <MAC_ID>`
  h) `[bluetooth]# exit`

For more information please visit:

https://wiki.tizen.org/wiki/Connecting_to_a_Smartphone_with_Bluetooth_and_Making_Phone_Calls
• In Tizen 3.0, we can configure Wi-Fi from the command prompt using the following steps:
  
a) `root@localhost:~# ifconfig wlan0 up`
  
b) `root@localhost:~# wpa_supplicant -u -t -B -d -Dwext -f/var/log/wpa_supplicant.log`
  
c) `root@localhost:~# connmanctl`
  
d) `connmanctl> enable wifi`
  
e) `connmanctl> agent on`
  
f) `connmanctl> services`
  
  o `wifi_<wlan0_MAC_ID>_<XXXXX>_managed_psk`
  
g) `connmanctl> connect wifi_<XXXXX>_managed_psk
  [Enter the passphrase here] xxxxxxxxx`
  
h) `connmanctl> exit`
• If Wi-Fi is connected properly, an IP Address would be assigned to wlan0 interface:

root@localhost:~: # ifconfig
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.43.91 netmask 255.255.255.0 broadcast 192.168.43.255
inet6 fe80::2c1:41ff:fe29:9c80 prefixlen 64 scopeid 0x20<link>
ether 00:c1:41:29:9c:80 txqueuelen 1000 (Ethernet)
RX packets 24 bytes 2789 (2.7 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 37 bytes 5470 (5.3 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

• Assemble the Robot DIY kit (that includes: chassis, DC motors, rubber wheels, castor wheels).
• Attach the Raspberry Pi2 to the chassis.
• Attach the power bank under the chassis and connect the USB cable to it (*Do not connect to RPi2 now*).
• Attach the L293D driver board to the chassis.
• Stick the battery case on the chassis (*Do not put the battery*).
• Connect the motor driver as shown in the next slide.
• Connect the Wi-Fi, Bluetooth, Webcam to the RPi2 USB slot.
• Temporarily connect the keyboard and monitor to do the initial configuration and setup (*Remove it once done*).
• Now, connect the power bank USB cable to the RPi2 (to power on and boot the Pi2).
• Finally, connect the battery to the battery case.
Motor Connection with PI2

Raspberry Pi 2 - > GPIO Pins

4 AA Battery Case < 12V

L293D Driver Board

DC Motor 1

DC Motor 2

Embedded Linux Conference – 06th April/2016
Tizen Smart Phone (Samsung Z1)

<table>
<thead>
<tr>
<th></th>
<th>Forward (F)</th>
<th>Back (B)</th>
<th>Left (L)</th>
<th>Right (R)</th>
<th>Stop (S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIO17</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>GPIO18</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GPIO22</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GPIO23</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

F => Forward  L => Left
B => Back     R => Right
S => Stop the robot

DC Motor 1
DC Motor 2

GPIO17
GPIO18
GPIO22
GPIO23

BT USB Dongle

Embedded Linux Conference – 06th April/2016
• The devices should be already paired and connected.

• Use Tizen Mobile 2.4 SDK to develop RFCOMM Client App.
  – Reference: https://developer.tizen.org/dev-guide/2.4/org.tizen.native.mobile.apireference/group__CAPI__NETWORK__BLUETOOTH__SOCKET__MODULE.html

• Use Tizen CAPI to develop RFCOMM server that runs as a Daemon to receive data from client and take action.
  – Reference: https://review.tizen.org/git/?p=framework/api/bluetooth.git;a=tree;f=test;h=e7732ccffdc87b0ae64c55e5486581a4b5956653;hb=HEAD

• To control the motor, we can simple write \{1,0\} to the respective GPIOs as shown in the table, using the GPIO sysfs entries.
  – echo 1 > /sys/class/gpio/gpio17/value
  – echo 0 > /sys/class/gpio/gpio18/value
  – echo 1 > /sys/class/gpio/gpio22/value
  – echo 0 > /sys/class/gpio/gpio23/value

  [Forward]
## RAM Usage

- RAM memory usage just after boot-up with Wi-Fi, Bluetooth connected (without display)

<table>
<thead>
<tr>
<th>Mem:</th>
<th>total</th>
<th>used</th>
<th>free</th>
<th>shared</th>
<th>buffers</th>
<th>cached</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>973</td>
<td>137</td>
<td>835</td>
<td>12</td>
<td>6</td>
<td>75</td>
</tr>
<tr>
<td>-/+ buffers/cache:</td>
<td></td>
<td></td>
<td>56</td>
<td>916</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Swap:</td>
<td>255</td>
<td>0</td>
<td>255</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>1229</td>
<td>137</td>
<td>1091</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- RAM: 1GB (1024 MB)
- Reserved Memory: (1024 – 973) = 51 MB
- Used during boot-up: 137 MB
- Total Used: 51 + 137 = 188 MB
- Swap (ZRAM) = 256 MB (1/4\(^{th}\) for 1GB)

[Change zram size in: 
/etc/resourced/swap.conf]

**Note:**

To get process wise memory usage, we can use a special command in Tizen:

# memps –a
**Kernel Code Size:**

```
# size -t vmlinux

<table>
<thead>
<tr>
<th>text</th>
<th>data</th>
<th>bss</th>
<th>dec</th>
<th>hex</th>
<th>filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>8314042</td>
<td>690396</td>
<td>784004</td>
<td>9788442</td>
<td>955c1a</td>
<td>vmlinux</td>
</tr>
<tr>
<td>8314042</td>
<td>690396</td>
<td>784004</td>
<td>9788442</td>
<td>955c1a</td>
<td>(TOTALS)</td>
</tr>
</tbody>
</table>
```

**Kernel Reserved:**

```
# dmesg | grep -i memory
Memory: 988016K/1015808K available (6123K kernel code, 527K rwdata, 1688K rodata, 448K init, 757K bss, 19600K reserved, 8192K cma-reserved)
```

Total RAM visible to Kernel = 1015808K = **992MB**
Reserved for GPU = **16MB** (cat /media/boot/config.txt : gpu_mem=16)
Reserved memory others = **16MB** (????)
Kernel Reserved = 19600 = **19.14MB** (includes kernel code & data structures)

- Kernel code size can be reduced below 5MB.
- Platform memory can be optimized further by analyzing memps report.
ROM Details

- ROM memory details with Tizen common 3.0 profile.

<table>
<thead>
<tr>
<th>Filesystem</th>
<th>Size</th>
<th>Used</th>
<th>Avail</th>
<th>Use%</th>
<th>Mounted on</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/root</td>
<td>945M</td>
<td>806M</td>
<td>114M</td>
<td>88%</td>
<td>/</td>
</tr>
<tr>
<td>devtmpfs</td>
<td>483M</td>
<td>4.0K</td>
<td>483M</td>
<td>1%</td>
<td>/dev</td>
</tr>
<tr>
<td>tmpfs</td>
<td>487M</td>
<td>4.0K</td>
<td>487M</td>
<td>1%</td>
<td>/dev/shm</td>
</tr>
<tr>
<td>tmpfs</td>
<td>487M</td>
<td>13M</td>
<td>475M</td>
<td>3%</td>
<td>/run</td>
</tr>
<tr>
<td>tmpfs</td>
<td>487M</td>
<td>8.0K</td>
<td>487M</td>
<td>1%</td>
<td>/tmp</td>
</tr>
<tr>
<td>/dev/mmcblk0p8</td>
<td>180M</td>
<td>384K</td>
<td>175M</td>
<td>1%</td>
<td>/opt</td>
</tr>
<tr>
<td>/dev/mmcblk0p9</td>
<td>4.9G</td>
<td>14M</td>
<td>4.8G</td>
<td>1%</td>
<td>/opt/usr</td>
</tr>
<tr>
<td>tmpfs</td>
<td>487M</td>
<td>0</td>
<td>487M</td>
<td>0%</td>
<td>/opt/usr/share/crash/temp</td>
</tr>
<tr>
<td>tmpfs</td>
<td>98M</td>
<td>0</td>
<td>98M</td>
<td>0%</td>
<td>/run/user/5001</td>
</tr>
</tbody>
</table>

- Rootfs size is less than 1GB. Further reduction is possible.
- Still we have lots of space in usr partition.
- For our use case, even 2GB storage should be enough.
- User files can be stored in /usr/share/media/ folders.
<table>
<thead>
<tr>
<th>Module</th>
<th>Size</th>
<th>Used by</th>
</tr>
</thead>
<tbody>
<tr>
<td>rfcomm</td>
<td>33992</td>
<td>2</td>
</tr>
<tr>
<td>btusb</td>
<td>29353</td>
<td>0</td>
</tr>
<tr>
<td>bnep</td>
<td>10479</td>
<td>2</td>
</tr>
<tr>
<td>btintel</td>
<td>1369</td>
<td>1 btusb</td>
</tr>
<tr>
<td>btbcm</td>
<td>4490</td>
<td>1 btusb</td>
</tr>
<tr>
<td>xt_connmark</td>
<td>1735</td>
<td>0</td>
</tr>
<tr>
<td>iptable_nat</td>
<td>1646</td>
<td>0</td>
</tr>
<tr>
<td>nf_conntrack_ipv4</td>
<td>13237</td>
<td>1</td>
</tr>
<tr>
<td>nf_defrag_ipv4</td>
<td>1321</td>
<td>1 nf_conntrack_ipv4</td>
</tr>
<tr>
<td>nf_nat_ipv4</td>
<td>4891</td>
<td>1 iptable_nat</td>
</tr>
<tr>
<td>nf_nat</td>
<td>12207</td>
<td>1 nf_nat_ipv4</td>
</tr>
<tr>
<td>nf_conntrack</td>
<td>76946</td>
<td>4 nf_nat,nf_nat_ipv4,xt_connmark,nf_conntrack_ipv4</td>
</tr>
<tr>
<td>xt_mark</td>
<td>998</td>
<td>0</td>
</tr>
<tr>
<td>iptable_filter</td>
<td>1275</td>
<td>0</td>
</tr>
<tr>
<td>iptable_mangle</td>
<td>1379</td>
<td>1</td>
</tr>
<tr>
<td>ip_tables</td>
<td>11439</td>
<td>3 iptable_filter,iptable_mangle,iptable_nat</td>
</tr>
<tr>
<td>x_tables</td>
<td>13353</td>
<td>5 xt_mark,ip_tables,iptable_filter,xt_connmark,iptable_mangle</td>
</tr>
<tr>
<td>bluetooth</td>
<td>324803</td>
<td>26 bnep,btbcm,btusb,rfcomm,btintel</td>
</tr>
<tr>
<td>bcm2835_gpiomem</td>
<td>2973</td>
<td>0</td>
</tr>
<tr>
<td>bcm2835_rng</td>
<td>1770</td>
<td>0</td>
</tr>
<tr>
<td>arc4</td>
<td>1778</td>
<td>2</td>
</tr>
<tr>
<td>rt2800usb</td>
<td>17476</td>
<td>0</td>
</tr>
<tr>
<td>rt2800lib</td>
<td>71877</td>
<td>1 rt2800usb</td>
</tr>
<tr>
<td>crc_ccitt</td>
<td>1149</td>
<td>1 rt2800lib</td>
</tr>
<tr>
<td>rt2x00usb</td>
<td>8539</td>
<td>1 rt2800usb</td>
</tr>
<tr>
<td>rt2x00lib</td>
<td>36483</td>
<td>3 rt2x00usb,rt2800lib,rt2800usb</td>
</tr>
<tr>
<td>mac80211</td>
<td>523380</td>
<td>3 rt2x00lib,rt2800usb,rt2800usb</td>
</tr>
<tr>
<td>snd_bcm2835</td>
<td>19620</td>
<td>0</td>
</tr>
<tr>
<td>snd_pcm</td>
<td>74535</td>
<td>1 snd_bcm2835</td>
</tr>
<tr>
<td>snd_timer</td>
<td>18419</td>
<td>1 snd_pcm</td>
</tr>
<tr>
<td>snd</td>
<td>52151</td>
<td>3 snd_bcm2835,snd_timer,snd_pcm</td>
</tr>
<tr>
<td>cfg80211</td>
<td>403784</td>
<td>2 mac80211,rt2x00lib</td>
</tr>
<tr>
<td>uio_pdrv_genirq</td>
<td>2997</td>
<td>0</td>
</tr>
<tr>
<td>uio</td>
<td>7880</td>
<td>1 uio_pdrv_genirq</td>
</tr>
<tr>
<td>rfskill</td>
<td>16398</td>
<td>4 cfg80211,bluetooth</td>
</tr>
<tr>
<td>joydev</td>
<td>9213</td>
<td>0</td>
</tr>
<tr>
<td>evdev</td>
<td>10421</td>
<td>5</td>
</tr>
<tr>
<td>sch_fq_codel</td>
<td>7858</td>
<td>2</td>
</tr>
<tr>
<td>ipv6</td>
<td>341361</td>
<td>20</td>
</tr>
</tbody>
</table>
**BENEFITS OF USING TIZEN**

- Tizen is truly a open source platform. Every component used in Tizen is derived from open source. So we have huge flexibility to customize as per our needs.
- Tizen uses profile concept to support new devices. So new use case can be easily derived using one of the profile.
- As we have seen, it is very easy to create new profile to support future technologies.
- With Tizen is easy to create bare minimal functionalities with lesser foot prints.
- Because of it’s multi-device capabilities it is possible to create device convergence and derive new communication mechanism.
- Finally, using the power of open h/w and open source OS, it is easy to perform various experiments before deriving actual products.
FUTURE WORK

- Perform various clean-ups and create a simple Robotics profile.
- Touch screen display bring-up using DSI connector.
- CSI camera bring-up.
- SDB bring-up and integration in Raspberry Pi Kernel.
- Various sensors interfacing with the robot.
- Power consumption analysis while robot is in operation.
- Setting up the web server and controlling robot using Wi-Fi.
- Getting camera preview remotely on smart phone.
- Contribute all the changes to upstream and update in Tizen wiki.

Others who like to contribute can join:
  - https://wiki.tizen.org/wiki/How_to_contribute_to_Tizen_on_Yocto_Project
  - Community:
    - https://www.tizen.org/community/mailing-lists
REFERENCES

- https://www.tizen.org/ko?langredirect=1
- https://review.tizen.org/git/
- https://wiki.tizen.org/wiki/Tizen_on_Yocto_Project
- https://blogs.s-osg.org/tizen-on-rpi2/
- https://people.csail.mit.edu/albert/bluez-intro/
THANKS