International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8, Issue-1C, May 2019

Development of Raspbian kernel Customization for Automatic Railway Level Crossing Application

Sathish Pasika, D. Krishna Reddy, N. Alivelu Manga

Abstract*: *In the recent years, the usage of the linux Operating System (OS) becomes very important for the real-time monitoring applications. The performance of embedded application depends on the important factors such as response time, memory size and power consumption. Among these parameters, memory size plays an vital role in kernel implementation. Customizing a general purpose OS to an application-specific OS is a challenging task for real time environments. Raspbian OS is the most recommended, open-source linux based OS for Raspberry pi board. In this paper, the customization of the Raspbian OS for automatic railway level crossing application is discussed. The novelty of this paper is to develop various algorithms for the customization of Raspbian OS and implementation of the application. The application is implemented by using Raspberry pi 3 board, IR sensors, DC motor, LED and buzzer. The railway gate is controlled by using IR sensors and DC motor interfaced through pi board. An IoT based application is to be developed for real time monitoring of the status of train and railway gate. The memory size of the Raspbian OS kernel is reduced by 42.71% after the customization.

.

***Index Terms*: Raspbian OS, Customization, Web server, Internet of Things**

I. INTRODUCTION

Linux is an open-source operating system in which the source code of the kernel is freely available and can be customized for various applications based on their specifications.The significance of customization of the kernel is removing the unnecessary modules in order to minimize the memory size and increase the application response time [8,9]. The development of the embedded OS is very important for the IoT (Internet of things) based applications. Raspberry pi board is an OS based board which was developed by Raspberry pi foundation. It has a microSD card support mounted on it. The OS is ported into SD card ported on it.In this paper, to develop automatic railway level crossing application by using raspberry pi the necessary modules of the kernel are considered. The process of implementation of the entire setup is divided into two steps.

Revised Manuscript Received on May 22, 2019.

Sathish Pasika Assistant professor, Dept. of ECE Chaitanya Bharathi Institute of Technology, Osmania University Hyderabad, Telangana, India, e-mail- satish35ece@gmail.com
Krishna Reddy: Professor and Head, Dept. of ECE Chaitanya Bharathi Institute of Technology, Osmania University Hyderabad, Telangana, India.
Alivelu Ranga Associate Professor, Dept. of ECE

The first step is customization of Raspbian kernel and second step is development of IoT based application for automatic railway level crossing [1,2].In the first step, the raspbian OS kernel is cloned from the git repository and is customized by removing unnecessary modules. Acustomized raspbian kernel image is created and is ported into the microSD card mounted on raspberry pi board. In the second step, the raspberry pi board with customized raspbian kernel is interfaced with various components to develop automatic railway level crossing application [5,6,7]. Python language is used for the source code development of the application. An Apache web server and HTML are used for the IoT application development been accepted, prepare it in two-column format, including figures and tables.

II. RASPBIAN FILE STRUCTURE

The file structure of the Raspbian OS needs to be considered for the kernel customization process. The various directories and its importance are listed below:

bin - a standard subdirectory that contains executable programs and essential user command binaries
/boot - contains static files of boot loader that are used in booting the OS.

/dev - contains device files. All the devices in the linux are represented by files, including disk partitions of any attached hard drives.

/etc - contains configuration files that are used to control the operation of a program.

/home - serves as a repository for user's personal files, directories and programs.

/root - serves as home folder for root user

/lib - contains shared libraries that are used by different applications on pi.

/proc - contains process information represented as text files.

/run - used by running applications to store data.

Retrieval Number: A10340581C19/19©BEIESP
Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

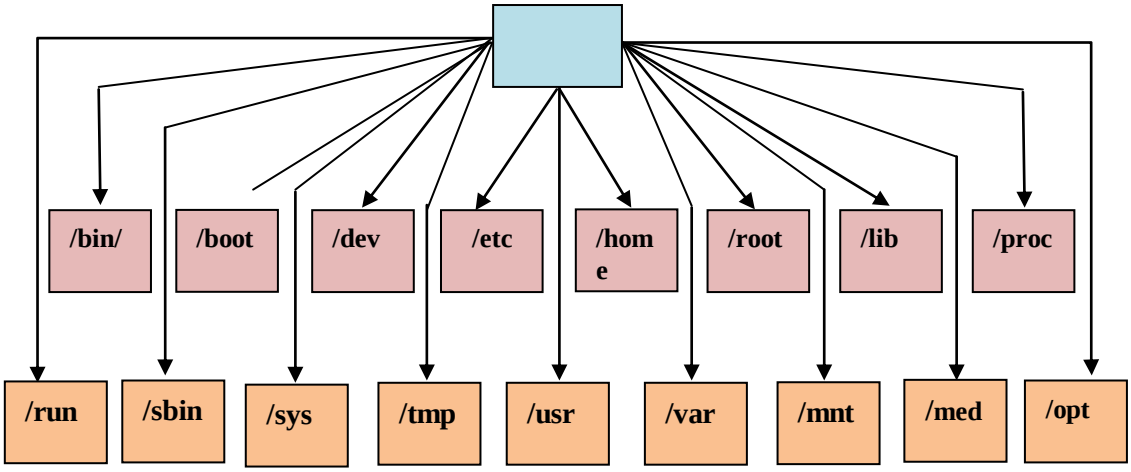
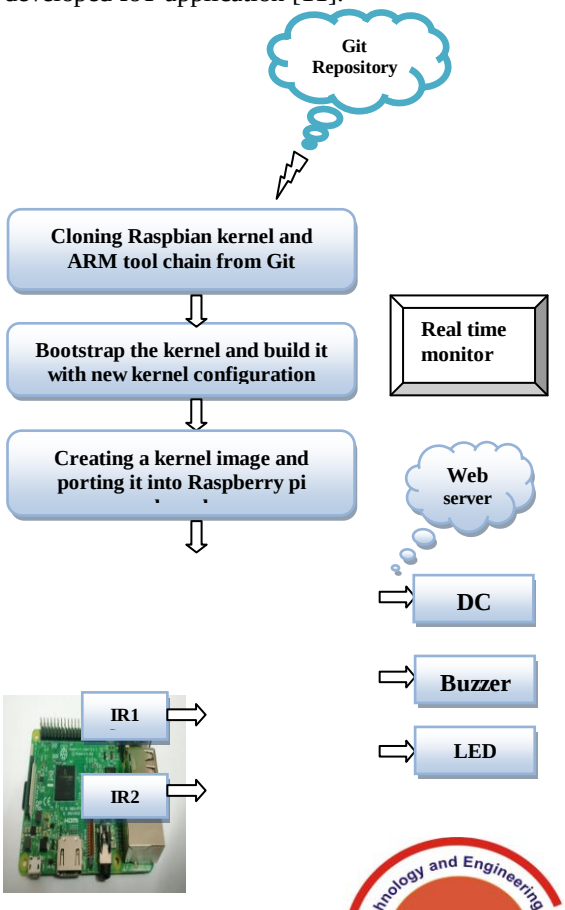


Fig 1: Raspbian file system structure

/sbin - contains system binaries. These are the applications that are mostly run by root user.
/sys - contains information and statistics about devices on the pi like display and sound.
/tmp - contains temporary files used by the operating system
/usr – contains user binaries, header files and libraries.
/var – another directory that OS writes files to during normal operation
/mnt – mount points for any temporary file system
/media – mount points for removable media. When USB hard drives are mounted on the pi board, they are accessed using media directory.
/opt – contains the applications that are installed on the pi.

III. PROPOSED SYSTEM BLOCK DIAGRAM

The customization of the Raspbian kernel is implemented by using ARM tool chain. Raspbian kernel source which is present in git repository is cloned into host machine. A git client is installed in the host machine to fetch the files from the git repository. The kernel source and ARM tool chain are cloned from the git repository and are stored in "Linux" and "Tools" sub directories. A specific tool chain having Linaro patches is selected from several tool chains present in the tools directory and an environment variable is set up to that particular tool chain. The kernel source is customized by configuring the ".config" file and it is built with new configuration. An image file of customized kernel (kernel.img) is created in the host machine. The customized kernel image and its associated modules are moved to '/boot' directory of Raspberry pi. Then the Pi board is restarted to boot the new kernel configuration. An automatic railway level crossing is developed by interfacing IR sensors, DC motor, Buzzer and LED with the raspberry pi board ported with customized OS[3,4]. In this Application, the arrival and departure of the trains are detected using IR sensors. The two sensors i.e., IR sensor 1 and 2 are placed on the either sides of railway level crossing. The sensed information is sent to the Raspberry pi board and based on the information the DC motor is operated in clockwise and counter clockwise to open and close the railway gate. The Buzzer which is placed near the railway gate is used as an audio alarming for the road passengers. An LED indicator is interfaced with the Raspberry pi board which is used for signalling at the railway tracks. Buzzer and LED are also used for signaling purpose in the 'collision case' to avoid the collision of two trains coming towards each other on the same track[10].
A web server is developed by installing "Apache" package in Raspberry pi. By default an HTML file is installed in the web folder of raspberry pi. The status of the railway gate is continuously updated and monitored using web server in the developed IoT application [11].



Retrieval Number: A10340581C19/19©BEIESP
Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Fig 2: Block diagram of automatic railway level crossing application using Raspberry pi board

IV. CUSTOMIZATION OF RASPBIAN KERNEL

The Raspbian kernel which is to be customized an ARM tool chains are cloned into build machine from git repository. The kernel source is then cleaned to remove all the previous configurations from configuration file. After cleaning the kernel source, the configuration file is pulled from the Linux sub directory. Then the kernel is bootstrapped with new kernel configuration.The Raspbian kernel source file is built with new configuration. The newly built kernel source is then verified for symbols. After successful verification of the kernel file, it is ready for porting into raspberry pi. An image file is created and copied into '/tmp' directory of host machine. Along with the kernel image file, a tar archive for modules is created and copied into 'tmp' directory of host machine. Then both the kernel image and tar archive are copied from 'tmp' directory of host machine to 'boot' directory of target machine into which the OS is ported. Then the target board is made to restart to boot the customized kernel.

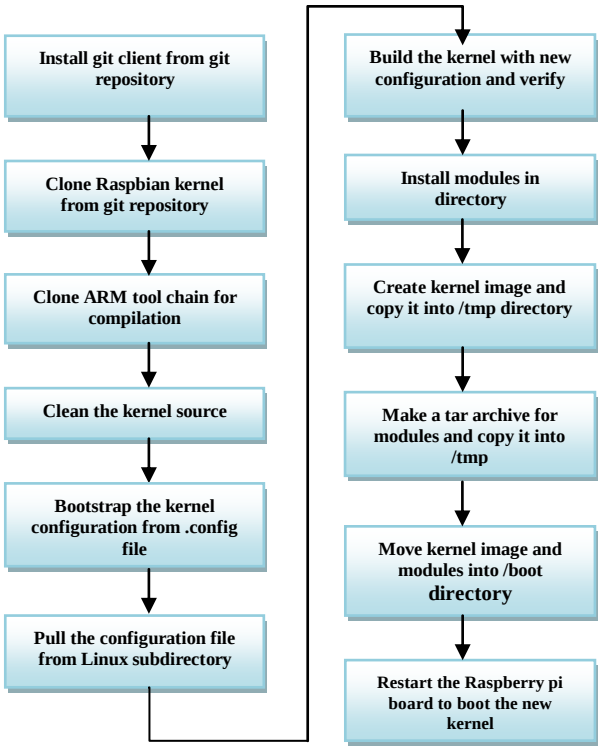


Fig 3: Flowchart for customization of kernel

V. AUTOMATIC RAILWAY LEVEL CROSSING APPLICATION

The development of the application includes hardware components such as IR sensors, DC motor, buzzer and LED are interfaced with the raspberry pi board to develop the hardware setup of automatic railway level crossing application. Python language is used in developing source code for the hardware setup of this application. Initially the Raspbian GPIO libraries and timing libraries are imported. The GPIO pins that are connected to hardware components are configured as inputs and outputs accordingly. Two Flags (Flag 1 and Flag 2) are considered and are initialized to Zero (0). These two flags are associated with two IR sensors (Flag 1 with IR sensor 1 and Flag 2 with IR sensor 2). The change in the state of sensor leads to change in the flag value (0 to 1 or 1 to 0). The flag values play a crucial role in identifying the event in the application i.e., arrival of train, departure of train.

In this application, an example is considered for a railway level crossing in between Secunderabad (sec-bad) and Warangal (wgl). The cases that occur are train moving from Secunderabad to Warangal, train moving from Warangal to Secunderabad and collision case. In case of train moving from Secunderabad to Warangal, the train initially passes through IR sensor 1. It checks with the state of flag 2 and in this case as flag 2 is in low state, it is considered as train is arriving from Secunderabad. The flag1 is then updated to high state (1). As the same train passes through IR sensor 2, it checks with the state of flag 1. As flag 1 is in high state, it is considered as departure of train from Secunderabad to Warangal. The railway gate is operated to open and close using DC motor. The same scenario is with the case of train moving from Warangal to Sec-bad. If both the flags (Flag1 and Flag2) become high at same time, it is considered as collision case i.e., two trains are coming towards each other. A warning alert is given using Buzzer and LED in this case.

VI. RESULTS AND DISCUSSIONS

(i) Customization of Raspbian kernel

There are various steps need to be performed for the customization of Raspbian kernel. The step by step execution procedure along with the results is explained.

Step 1: Installing git client using command to install the git client is "**sudo apt-get install git"**.

Step 2: Cloning Raspbian kernel

To store Raspberry pi related files, a directory on the name 'custom_raspbian' is created. In this step, the Raspbian kernel source present in the git repository is cloned into the host machine. The command used to clone the kernel source is **"git clone https://github.com/raspberrypi/linux.git"**

Retrieval Number: A10340581C19/19©BEIESP
Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

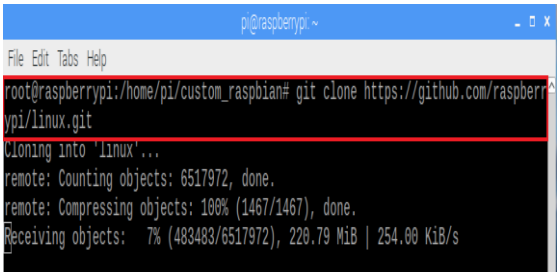


Fig 4: Cloning of Raspbian kernel source

The Raspberry pi kernel source will be downloaded into "Linux" subdirectory of "custom_raspbian". The path of the Raspbian kernel source is given an environment variable "KERNEL_SRC". In further steps KERNEL_SRC is used in using the Raspbian kernel instead of using entire path.

Step 3: Cloning of Arm Tool chain. ARM tool chains from git repository are downloaded in the host machine. The command used to download ARM tool chain is **"git clone https://github.com/raspberrypi/tools"**. The tool chains are downloaded in 'tools' subdirectory. A tool chain is selected from several tool chain versions and is given an environment variable "CCPREFIX". In further steps CCPREFIX is used instead of using entire path for ARM tool chain.

Step 4: Cleaning the kernel source. This is one of the important step before configuring and compiling the kernel. The command used to clean the kernel source is **"make mrproper".** This command makes the Raspbian kernel clean by removing the most used directories and previous configuration in '.config' file is also removed.

Step 5: Pulling the configuration file. The command used to pull the configuration file (.config) file is **"zcat /proc/config.gz > running.config"**. The configuration file is pulled from the kernel source and is used for configuration. 'zcat' in the command is used for reading the '.config' file from a zipped file. If the configuration file is not located, the command used to locate it is **"sudo modprobe configs"**.

Step 6: Bootstrapping the kernel source. To bootstrap the new kernel configuration from the '.config' file, the command used is **"ARCH=arm CROSS_COMPILE=${CCPREFIX} make oldconfig"**. 'ARCH' in this command is specified as 'arm' else the kernel will be configured using 'x86' configuration.

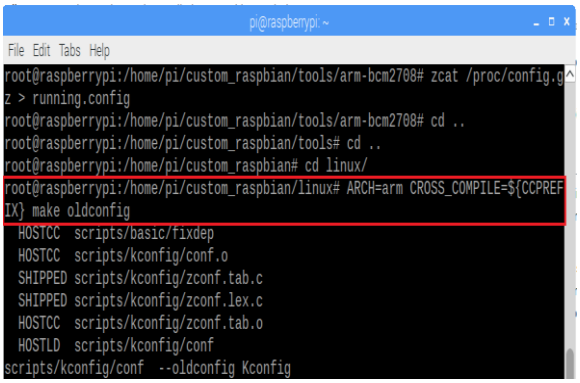


Fig 5: Bootstrapping the kernel source

Step 7: modifying the new kernel. The command used to modify the new kernel configuration is **"ARCH=arm CROSS_COMPILE={CCPREFIX} make menuconfig"**. 'make menuconfig' is a menu-driven user interface which allows to choose the features of kernel source that needs to be compiled.

Step 8: Building the new kernel. After configuring the '.config' file, the kernel is built. The command used to build the kernel source is **"ARCH=arm CROSS_COMPILE={CCPREFIX} make"**. The kernel which is customized with new configuration is built. To make the building process faster '-j<amount of cores>' is added at the end of the command such that the building process is distributed. If the cpu has 4 cores, -j<4-6> is acceptable.

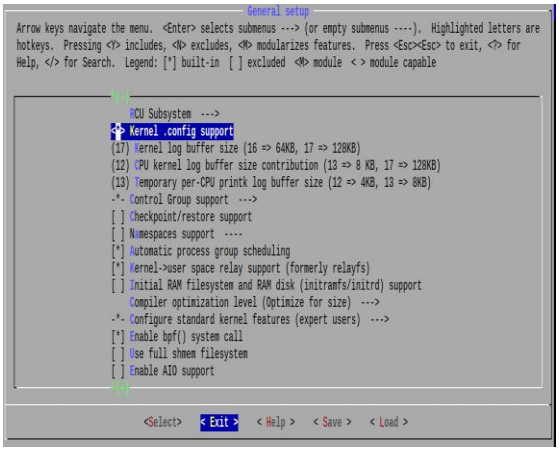


Fig 6: Modifying the kernel

Step 9: make modules_install**.** After building the kernel, the modules associated with it are installed. Before installing a directory on the name 'modules' is created in the directory where all raspberry pi related files are loaded. The command used to install the modules is **"ARCH=arm**

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication
Retrieval Number: A10340581C19/19©BEIESP

CROSS_COMPILE={CCPREFIX} INSTALL_MOD_PATH=${MODULES_TEMP} make modules_install"

Step 10: Creation of kernel image. An uncompressed kernel image is created using an image tool using the command **"python\./imagetool-uncompressed.py ${KERNEL_SRC}/arch/arm/boot/zImage"**. After created it is uploaded into the 'tmp' directory. The command used here is **"scp kernel.img pi@raspberrypi:/tmp"**.

Directory name	Memory size in original kernel	Memory size after customization	Reduced memory in %
/bin	6.0 MiB	6.0 MiB	0%
/boot	20.8 MiB	20.8 MiB	0%
/dev	1.7 KiB	1.22 KiB	28%
/etc	6.7 MiB	4.55 MiB	32%
/home	334.4 MiB	334.4 MiB	0%
/lib	182.2 MiB	115.7 MiB	36.5%
/proc	234.4 KiB	234.4 KiB	0%
/run	12 MiB	12 MiB	0%
/sbin	7.3 MiB	7.3 MiB	0%
/sys	36.9 MiB	32.47 MiB	12%
/tmp	48.0 KiB	48.0 KiB	0%
/usr	3.1 GiB	1.86 GiB	40%
/var	475.4 MiB	342.2 MiB	28%
/opt	918.4 MiB	183.6 MiB	80%
Total size	**5174.78 MiB**	**2964.13 MiB**	**42.71%**

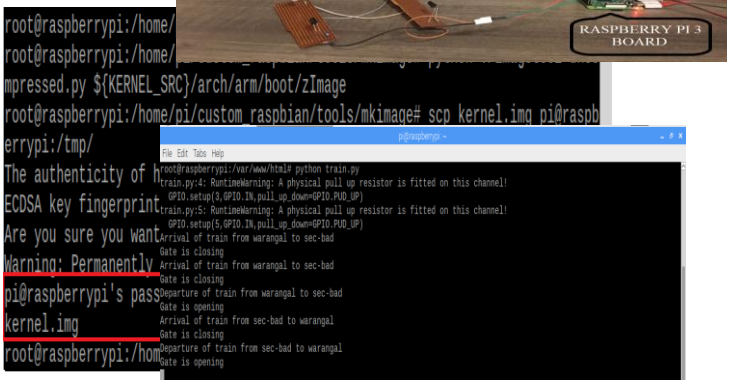


Fig 7: Creating kernel image file

Step 11: Making tar archive for modules. A tar archive containing the kernel modules is created and is uploaded it into the /tmp directory of raspberry pi. The command used to create a tar is **"tar modules.tgz *".** The command used to upload the tar archive containing modules into '/tmp' directory is **"scp modules.tgz pi@raspberrypi:/tmp"**

Step 12: Uploading kernel.img and modules into Raspberry pi. Kernel.img and modules are uploaded into the '/boot' directory of raspberry pi. The command used to upload the kernel.img into '/boot' directory is **"sudo mv /tmp/kernel.img /boot/"** and the command used to upload the modules is **"sudo tar xzf /tmp/modules.tgz"**.

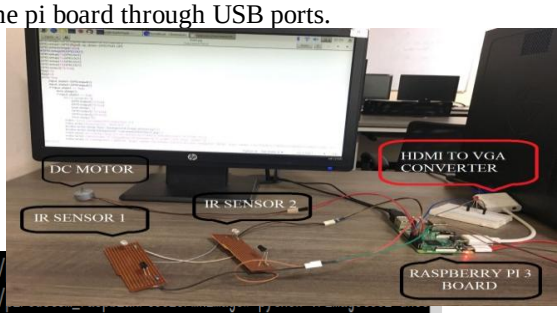
Finally the Raspberry pi board is restarted to boot the new kernel. The command to restart the Pi board is **"sudo shutdown –r now"**. The Raspberry pi board ported with customized Raspbian kernel is then used in developing the application. The memory size of the essential directories in the customized kernel is shown in Table I.

A. Memory size of various directories

Table 1 lists the memory size of various directories of raspbian kernel before and after customization. The memory size of all the directories in the original kernel file is amounted to 5174.78 MiB (Mebibytes). After removing unnecessary files from various directories of original kernel, the memory size of all the directories amounted to 2964.13 MiB which is a reduction of approximately 42.71% in the memory size of original kernel.

B. Automatic Railway level crossing application

The complete hardware setup is developed by interfacing IR sensors, DC motor, buzzer and LED with Raspberry pi 3 board as shown in Fig 11. Keyboard and mouse are connected to the pi board through USB ports.



(b)

Fig 7: (a) Hardware setup of Automatic railway level crossing application (b) Raspbian terminal output

C. Setting up of Apache web server

The application is further implemented using Apache web server. The command to install the apache web server is **"sudo apt-get install apache2"**. After installing the web server, by default an html file is placed in web folder of Raspberry pi. It can be accessed using **https://localhost/** or by using Pi's IP Address. The html file is used in the developed python code in such a way that for every

Retrieval Number: A10340581C19/19©BEIESP
Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

event in the application the status of train and railway gate is updated.

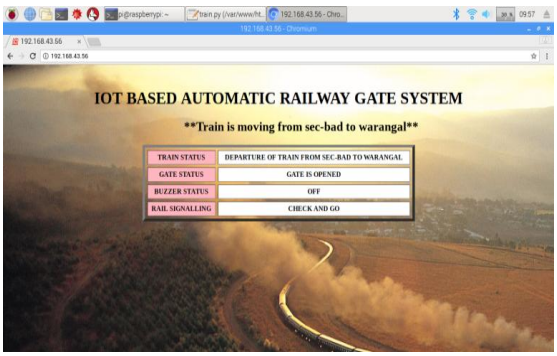


Fig 8: IoT based Application monitoring for railway level crossing

A condition where two trains are coming towards each other is called as collision case. It can be controlled using anti-collision technique. If the trains are coming from both the stations towards each other on a same track, a signal is sent to the Raspberry pi board indicating the collision. Based on this condition, the railway gate is closed, buzzer rung and led blinks continuously indicating the collision to the road passengers and loco pilot. With the continuous warning alerts, the trains are slowed down and made to halt before they approach each other.

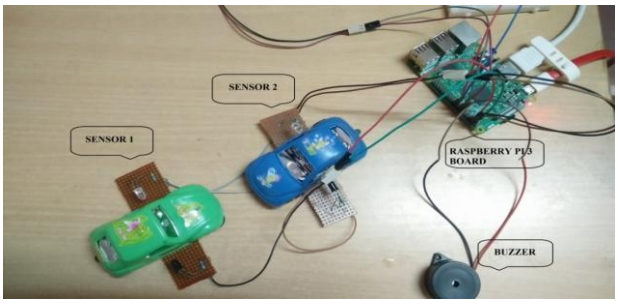


Fig 9: Hardware setup for Collision case

VII. CONCLUSIONS AND FUTURE SCOPE

The customization of Raspbian kernel structure is developed for Automatic railway level crossing application. By including the required modules, the image size of the kernel has been reduced by 42.7% compared to its original image size. After porting of Raspbian OS, a python code is developed for the application. A hardware setup is developed by interfacing IR sensors, DC motor, Buzzer and LED with the raspberry pi 3 board. Real time monitoring of the application is achieved by developing a web server such that the status of the railway level crossing is continuously updated. By implementing this system in a mobile application, the status of it can be monitored using smart phones.

REFERENCES

1. Bharti S.Dhande, Utkarsha S.Pancharaney, "Unmanned level crossing controller and Rail tack broken detection system using IR sensors and Internet of things technology", International conference on incentive communication and computational technologies(ICICCT), October, 2017.
2. G.Hemanth kumar, Ramesh G.P, "Intelligent gateway for real time train tracking and Railway crossing including emergency path using DD communication", International conference on Information, Communication and Embedded systems(ICICES), February, 2017.
3. A.Indhuja, C.Ranjani, "In-Service Rail track monitoring and fault reporting", International conference on innovations in Information, Embedded and communications systems(ICIIECS), 2017.
4. Ujjwal Kohli, Anmol Agarwal, "Smart Unmanned Level Crossing System in Indian Railways", International Journal of Mechanical and Production Engineering, ISSN: 2320-2092, Volume-4, Issue-10, Oct 2016.
5. N.R.Darga, G.R.Gidveer, "Wireless Web Server for Industrial Data Acquisition and Control Using Raspberry Pi", International Journal of Advanced Research in Computer and Communication Engineering, Vol.5, Issue 8, August 2016.
6. Randeep Kushwaha, Brij Bihari Chaubey, Jyotindra Kumar Singh, Prashant Kumar Dubey, Rahul Jaiswal, "Automatic Railway Gate Control System", International Journal Of Engineering and Computer Science, ISSN: 2319-7242, Volume 5, Issue 5, May 2016.
7. B.Brailson Mansingh, K.S.Selvakumar, S.R.Vignesh Kumar, "Automation in Unmanned Railaway Level Crossing", IEEE 9th International Conference on Intelligence Systems and Control (ISCO), 2015.
8. S.Pravin Kumar, G.Pradeep, G.Nantha Kumar, C.Dhivya Devi, "Developing an ARM Based GNU/LINUX Operating System for Single-Board Computer, Cubietruck", International Journal of Engineering and Technology (IJET), Vol.6, Jan 2015.
9. Chi-Tai Lee, Jim-Min Lin, Zeng-Wei Hong, Wei-Tsong Lee, "An Application Oriented Linux Kernel Customization For Embedded Systems", Journal of Information Science and Engineering, 20 June 2014.
10. [Amit Kumarand, M.N.Pandey, "Study of Safety Awareness at Railway Level Crossing", International Journal of Engineering Research and Technology (IJERT), ISSN: 2278-0181, Vol.2, Issue 12, December 2013.
11. Bhargav Goradiya, H.N.Pandya, "Real Time Monitoring and Data Logging System using ARM Architecture of Raspberry Pi and Arduino UNO", International Journal of VLSI and Embedded Systems-IJVES, Vol. 04, Article 0611; July 2013.

AUTHORS PROFILE

Sathish Pasika completed his B.Tech and M.Tech from JNTU University Hyderabad in 2005 and 2008 respectively. He is presently working as Assistant professor in ECE Dept., Chaitanya Bharathi Institute of Technology, Hyderabad. Having 11 years of experience in teaching and his research interest includes Embedded System Design and GNSS/NavIC applications.

Prof.. D. Krishna Reddy has obtained B.E. degree in ECE from Andhra University, Waltair in 1990 with distinction. In 1995, he obtained his M.E. degree. Awarded Ph.D degree in 2008 from Osmania University, Hyderabad. 26 years of experience in teaching and R&D organizations. Presently he is head & professor in ECE dept. of CBIT. He has about 67 publications to his credit both in national and international journals; His areas of research interest include Navigation, 4G and mobile communications.

Dr. N. Alivelu Manga completed her B. Tech from VNRVJIET in 2001 and received M. Tech from JNTUH in 2008. She obtained Ph. D from JNTUH in 2017. Currently working as Associate. Professor in the Dept of ECE CBIT. She published 7 international journals and 10 international conference papers. Having 18 years of experience in teaching and her research interest includes VLSI System Design, GNSS/ NavIC Applications.

Retrieval Number: A10340581C19/19©BEIESP

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication