Enhancement of Automation Testing System Using Yocto Project

Harita khandelwal
Department of Electronics & communication
G.H.Patel college of engineering & Technology
Vallabh vidyanagar,Anand,388120
harita.khandelwal@gmail.com

Parthesh Mankodi
Department of Electronics & communication
G.H.patel college of enginnering & Tech
Vallabh vidyanagar,Anand,388120
prmankodi@gmail.com

Ritesh prajapati
System Level Solutions pvt.ltd
Vallabh vidya nagar,Anand
ritesh.prajapati@slscorp.com

***Abstract*— nowadays, in industries, Testing has become one of the important tasks. Testing is necessary for an effective performance of product and software applications. When companies having a mass production of hardware boards, it is necessary to do testing of each and every module of hardware like SPI,UART,GPIO, PWM, ADC, USB, Ethernet .If we do testing manually then it will significantly take more time, which we can be reduced by automation testing. But the solution is not an automation testing because automation testing also requires the same amount of system, for example for 300 boards 300 system is required. The only difference in manual and automation testing is that in automation testing it will require less human effort. Now we are focusing more on developing a solution in which many tests can be done on one single system i.e. for 300 boards only one system is required for testing hence this problem can be solved by Yocto Project. Yocto project have build the tool named Bitbake which is written in Python language, which works on multithreading and scheduling so that simultaneously you can test more boards on a single system. In this paper we did automation testing of GPIO pins using Yocto project.**

Keywords— ***Automation, Yocto project, Open embedded, Bitbake***

I. INTRODUCTION

A. *Yocto project*

The Yocto Project[1] is an open source collaboration project that provides templates, tools, and methods to help you create custom Linux-based systems for embedded products regardless of hardware architecture

Yocto project architecture:

- User Configuration: Metadata for control the build process.
- Metadata Layers: layers that provides software's, BSP, and Metadata.
- Source Files: Source code from internet, local project
- Build System: Bitbake is build system for yocto project

Package Feeds: Package feeds to decide whether want to generate image, SDK

- Images: Images produced.
- Application Development SDK: Cross-development tools that are produced along with an image or separately with Bitbake

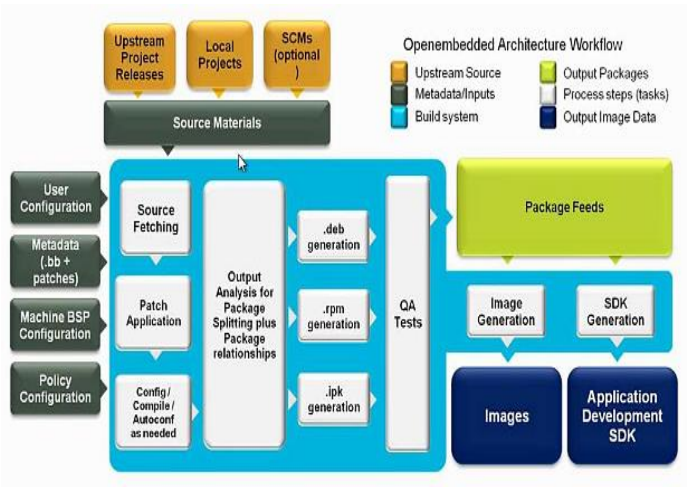


Fig 1 Yocto project architecture

B. *Bitbake*

Bitbake[1] is a build tool**.** Build tools are programs that automate the creation of executable applications from source code. Building incorporates compiling, linking and packaging the code into a usable or executable form.

Task of Bitbake:

Step1**:** Downloads data from upstream means from internet

Step2: extracts downloaded package

Step3: Do patching

Step4: configure the source

Step5: install to the staging area

Step6: install into package area

Step7: Creates packages like DEB, RPM or other packages

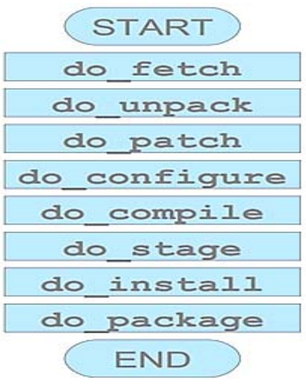


Fig 2 Flow Diagram of Bitbake

Fig 2 is the flow diagram of Bitbake that how it work.steps we explained in algorithm

Fig 3 This is a block diagram of Automation Testing System Using yocto project. Here as shown in the figure, there is one system on which bitbake is already installed. By using Bitbake Service file has been created. Service file helps us to enables the automation. E.g. one python script we created for testing purpose now we will call that python script in .service file. Now we will create image of .service file using bitbake build tool then after image creation, we will load that image on hardware.Due to .service file automation testing have been started and log Report will be generated that we can get on system, same way we can load image on more than one hardware and we can do testing of Multiple hardware.

II. Block diagram of Automation Testing

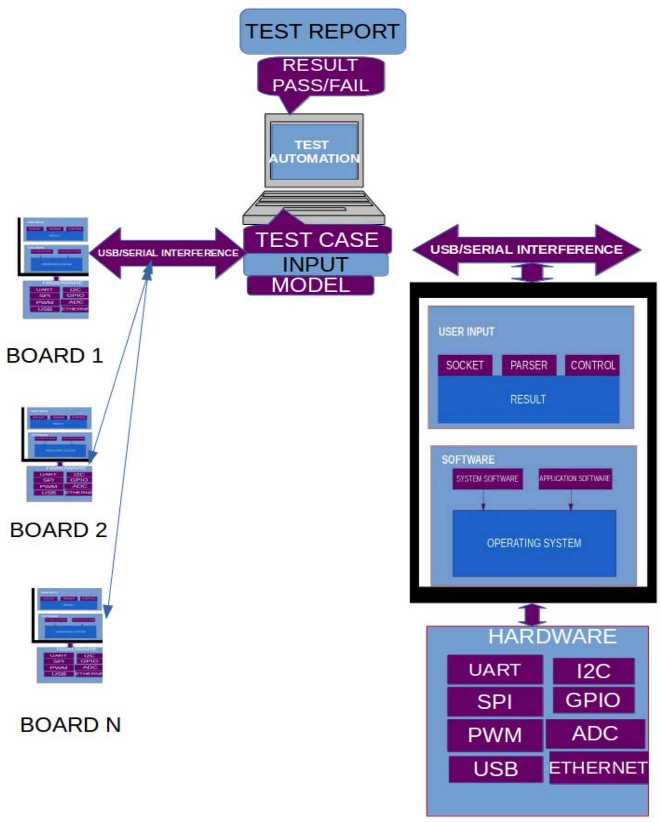


Fig 3 Block diagram of automation testing

III. Testing of GPIO pins

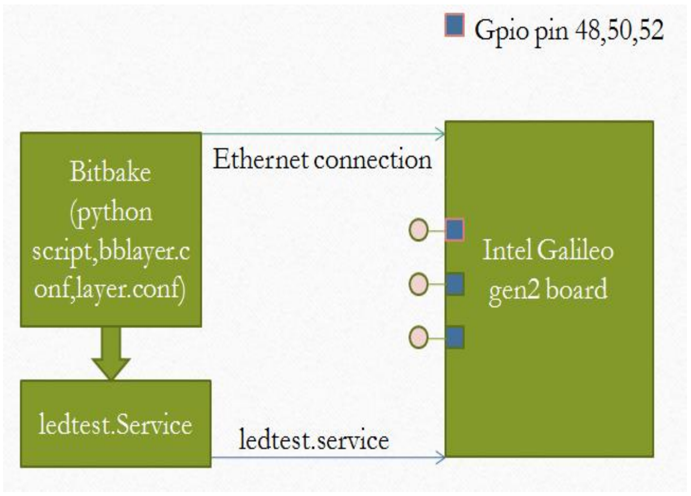


FIG 4 Testing of GPIO pins

We want to test GPIO pins 48, 50,52 so for that we are attaching LED's on 48,50,52 port of Intel Galileo generation2 board. Here we did Ethernet connection for connecting to the hardware. Firstly we create one python script which ON led for 10 seconds and which OFF led for 10 seconds .After creation of python script we call that python script in .service file, We are using .service file because .service file enables things automatically so that when a board is up it will start to testing automatically of GPIO pins.

.Service files are provided by Bitbake which makes Automation testing very easy, just we need to create any python or c file and that files we need to can call in .service file

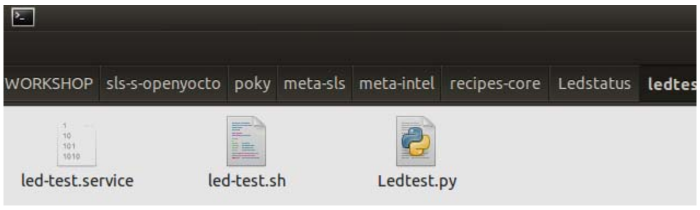


Fig 5 meta-ledtest

Here as shown in this fig 5 Ledtest.py is a python script which we are calling in led-test. service file for automation testing purpose. Python script directly can't be called by .service file so we creating .sh file which will read python file and that .sh file we will call in .service file

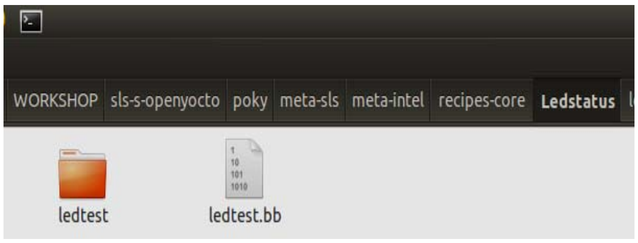


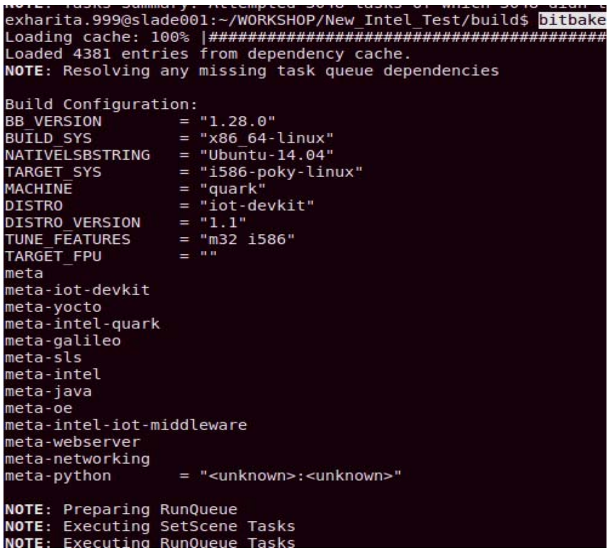
Fig 6 bb file in meta-ledtest

As shown in Fig 6 Ledtest.bb file is Bitbake file which is core file to generate image or binary file. We want to load image of Ledtest.py and led-test. service file on hardware, so for that we need to call Ledtest.py ,led-test.sh and led-test. Service in ledtest.bb file .So when you run following command it will generate image in build folder and that you can load on hardware

Command:

bitbake core-image-minimal (To Build whole build)

bitbake example.bb (example.bb file in sample bitbake layer file)



```
exharita.999@slade001:~/WORKSHOP/New_Intel_Test/build$ bitbake
Loading cache: 100% |##############################################
Loaded 4381 entries from dependency cache.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION          = "1.28.0"
BUILD_SYS           = "x86_64-linux"
NATIVELSBSTRING     = "Ubuntu-14.04"
TARGET_SYS          = "i586-poky-linux"
MACHINE             = "quark"
DISTRO              = "iot-devkit"
DISTRO_VERSION      = "1.1"
TUNE_FEATURES       = "m32 i586"
TARGET_FPU          = ""
meta
meta-iot-devkit
meta-yocto
meta-intel-quark
meta-galileo
meta-sls
meta-intel
meta-java
meta-oe
meta-intel-iot-middleware
meta-webserver
meta-networking
meta-python         = "<unknown>:<unknown>"

NOTE: Preparing RunQueue
NOTE: Executing SetScene Tasks
NOTE: Executing RunQueue Tasks
```

Fig 6 Build is created

As shown in figure build is created when we run the first command

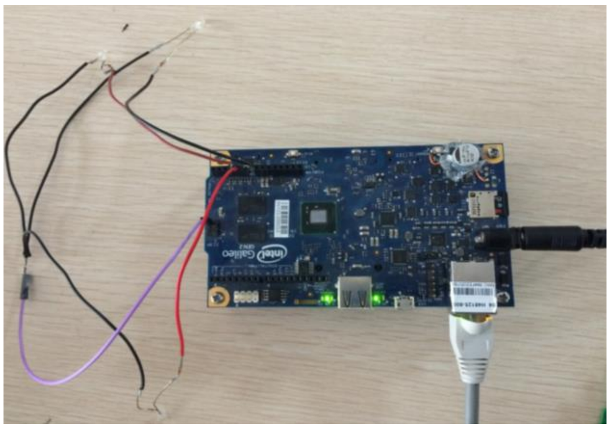


Fig 7 Three gpio pin automation Testing system using Bitbake

As shown in Fig 7 we are using Intel Galileo generation 2 boards for testing purpose. Here we selected pin 48, 50, 52 GPIO pin for testing, here we used three led's for testing .we attached led's on port 48, 50, 52 initial stage it is in off state

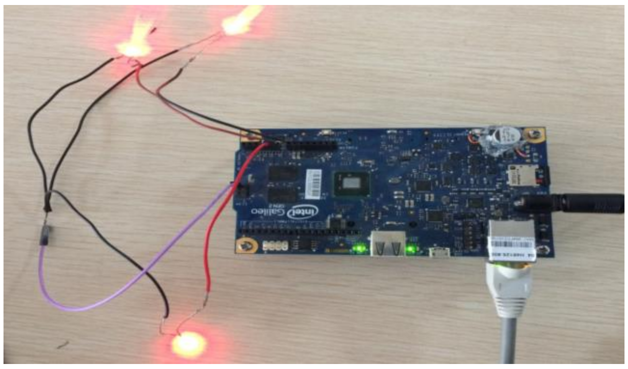


Fig 8 Three Gpio pin automation Testing system using Bitbake success

After loading an image it starts testing because GPIO pins are working fine Led's blinking as per image we loaded on Hardware.
So by that we can say that GPIO pins are ok, Test is done and that Log report we can get on system

FUTURE WORK

1. SPI testing
2. I2c testing
3. UART testing
4. PWM testing
5. USB testing
6. Ethernet testing

In Future we are going to test above mentioned testing using Bitbake on multiple hardware at single time

Licences Required To Use Yocto Project

1.**MITLicence:**The **MIT License** is a permissive free software license originating at the Massachusetts Institute of Technology (MIT) As a permissive license, it puts only very limited restriction on reuse and has, therefore, an excellent license compatibility

2.**GPL licence: General Public License** is a widely used free software **license**, which guarantees end users the freedom to run, study, share and modify the software.

3.**AGPL Licence**: The Affero General Public License. This provision requires that the full source code be made available to any network user of the AGPL-licensed work, typically a web application.

CONCLUSION

The purpose of this paper is to describe how automation testing can be done using Yocto project. This paper shows how we can create .service file which helps to do automation. Bitbake helps to create image that we can load on any hardware and we can do automation testing and we are also able to create test report.

ACKNOWLEDGEMENTS

Authors are grateful to G. H. Patel College of Engineering and Technology and System Level solutions Pvt.Ltd and various researchers and their work, which in turn helped to complete this study.

REFERENCES

[1] Harita Khandelwal,Parthesh Mankodi and Ritesh Prajapati "A Survey: Improve Automation Testing system using Yocto Project, International Conference on Intelligent Systems and Signal Processing (ISSP- 2017)" on 23-25 March 2017

[2]Arttu Leppakoski Erno Salminen and Timo D. Hamalainen on Framework for Industrial Embedded System Product Development and Management System on Chip (SoC), 2013 International Symposium on 23-24 Oct. 2013

[3]J. Jocklin, "Koneenohjausjarjestelman ylHipito" (in Finnish), Master's thesis, Aalto University, 2011

[4]R. Domer, A. Gertslauer, W. MOiler, "Introduction to Hardware-dependent Software design," [n Asia and South Pacific Design Automation Conference (ASP-DAC), Jan 2009, pp. 290-292.

[5] Linux Foundation, "Yocto Project I Open Source embedded Linux build system, package metadata and SDK generator,"Online, oct3-2016, https://www.Yoctoproject.org

[6]Montavista, MontaVista embedded Linux software and development tools for intelligent devices and embedded systems, Online, Cited: oct 4-2016, http://www.mvista.com.

[7]H. Elemo, "Defining Software Configuration Management for Product Development," Helsinki University of Technology, Master's thesis, 2008, 92 pages

[8]Galileo and Minnov board projects,online,cited oct 4-2016, https://www.Yoctoproject.org

[9] P. Mishra and N. Dutt, "Architecture description languages for programmable embedded systems," IEE Proceedings-Computers andDigital Techniques, vol. 152, no. 3, pp. 285–297, 2005.

[10] A. Mashtizadeh, "PHDL: A Python hardware design framework," M.S.thesis, ECE Dept. MIT, 2007

[11] "System Python," 2015, http://cgi.di.uoa.gr/_ evlog/syspy.html

[12] P. Haglund, O. Mencer, W. Luk, and B. Tai, "PyHDL: Hardware scripting with Python," in *Proc. International Conference on Field Programmable Logic (FPL)*, 2003, pp. 1040–1043