PRESENTER TRACKING FOR VIDEO RECORDING ON INTEL GALILEO BOARD

By

NMDJANDOD PATRICK
BERIMAN

14175

Dissertation submitted in partial fulfillment of

the requirements for the

Bachelor of Engineering (Hons)

(Electrical and Electronics)

JANUARY 2016

Univeristi Teknologi PETRONAS
Bandar Sri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Presenter Tracking for Video Recording on Intel Galileo Board

By

Nmdjandod Patrick Beriman
14175

A project dissertation submitted to the
Electrical and Electronic Engineering
Programme Universiti Teknologi
PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL & ELECTRONICS)

Approved by:

_____

AP. Dr. Fawnizu Azmadi Hussin

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
JANUARY 2016

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or individuals.

_____

Nmdjandod Patrick Beriman

ABSTRACT

This paper proposes an intelligent video recording system based on Intel Galileo Gen2 board. The system is able to perform image processing; more precisely face detection and detect a presenter face using OpenCV libraries. Face detection was used to track the presenter's position with the goal to effectively and autonomously make video recording. Processing images usually require a computer with a higher processing capability. But its application on smaller scale microcontroller can play an important role in the fields of science and technology with application in areas such as in television, video surveillance systems, robotics and industrial inspection.

This project successfully integrated a C++ based face detection code with an Arduino sketch for servo motor control on Intel Galileo Gen2. Face detection is achieved by capturing images extract and process single frame to detect face. The Arduino sketch is used to track the detected face by steady control of servo motors to which Logitech C270 720p 3-MP Widescreen HD Webcam is attached. The two processes inter-communicate using a shared memory.

ACKNOWLEDGEMENT

This project would not have been possible without the help and assistance received from so many people in so different ways.

The author extends his deepest gratitude to the project supervisor AP Dr Fawnizu Azmadi Hussin for his valuable guidance, support, patience and more importantly his willingness to put up with my numerous mistakes. All this contributed to helping the author remain focus on the more important tasks.

The author would like to extend his appreciation also to Mr. Abu Bakar Sayuti for the directions and suggestions provided. They help to greatly narrow down the work to be done.

The author would like to extend a sincere gratitude and appreciation to the author's family for both the mental and physical support. This helps the author to overcome difficult challenges. Besides, a special thanks goes to many of my friends for their endless support.

Last but not least, the author would like to thank all parties who have helped in this project and making it a success.

TABLE OF CONTENTS

Table of Figures:

CHAPTER I

INTRODUCTION

1.1 Background

Video and image processing are proving to be providing effective solutions in the multimedia information world. However processing videos and images usually require a computer with a higher processing capability. With resent influx of embedded processors such as Intel Galileo, raspberry Pi, beaglebone and many more, image and video processing have taken a new dimension. Their size and cost made them suitable for several applications.

Embedded platforms are receiving a continuous improvement on their computing capabilities making them more reliable.

The purpose of this project is to implement face detection and tracking on Intel Galileo. A model application of the system will be to detect, track and record a presenter for video conference.

1.2 Problem Statement

Presenter detection and tracking is made by recording a video using camera, capture single frames and process it to extract facial features used to localize a human face. Once detected, the face is tracked using servo motors attached to the camera, allowing a continuous recording of the presenter. This must be done autonomously using Intel Galileo Board. This was implemented by a former Universiti Teknologi Petronas student. Although the system worked on a PC running OpenCV, its implementation on Intel Galileo did not work properly. This is due to firmware bugs and the programing language used to develop the SD image.

This project addresses the challenges in the implementation of face detection and tracking on Intel Galileo board mainly building and booting a Linux based SD image.

1.3 Objectives and Scope of Study

The objectives of this project are:

a) Optimization of the existing SD image for implementation of embedded Linux
b) Proper interfacing of the the embedded platform (Intel Galileo) with camera for data acquisition (Driver development may be required).
c) Proper interfacing of the embedded platform with Servo motors for presenter tracking
d) Implementation of the developed SD Image on the embedded platform.

The scope of this project is to focus on the implementation of the developed face detection algorithm and tracking on Intel Galileo board. This project does not focus on the development of new face detection and tracking algorithm.

CHAPTER II

LITERATURE REVIEW

2.1 Insight into Face detection algorithm, challenges and application

An insight into our developed face detection algorithm will help us understand the challenges of its implementation on embedded platform. In fact, the objective of face detection is to find and locate a face in a given image or video frame. The technique used to develop our algorithm is the one proposed by Viola and Jones [1]. It has the advantage of offering efficiency in detection rate and fast processing speed. The challenge of face detection on embedded systems is circuit size and processing speed. The technique has been successfully used in personal computer using OpenCV [2]. This is due to the fact that face detection use Haar-like features classifiers that check every single pixel in a given image. However the computing power of embedded systems processors is less powerful in comparison with those in PCs. Even so, face detection can be optimized on embedded system with the addition of coprocessor acceleration for image processing [3].

Face detection algorithm was introduced for the first time in to intel chips by Lienhart and Maydt [4]. It was later used to build OpenCV library [2] His experiment with an image frame of 176 X 144 size on a 2 GHz Pentium 4 processor resulted in processing 15 Frames per second (fps).[3] explained that if the experiment is run on a 200 Mhz ARM processor, the speed of detection will only be 7 fps which is far from helping in real time image processing. Cho et al. [5] proposed an architecture that made used of multiple classifiers to detect face. Later Hiromoto et al. [6] propose a hybrid model that use parallel and sequential modules.

Gao et Lu [7], presented a model that uses FPGA to accelerate the Haar-like features processing. However they were able to use only classifiers on the chip. The integral image was done in a computer processor.

In [3] Kim and Kim proposed an architecture that is based on Adaboost learning algorithm with Haar-like features. The goal was to apply face detection to a low cost FPGA applicable to analogue video. The performance of their architecture was compared with detection result using Open CV. The result showed that the processing speed for face detection they have designed was 3 time faster than previous work for low cost face detection.

2.2 Intel Galileo and Servo Motors

For this project, servo motors are used to perform face detection. More information on servo working principle can be found in [10].

Figure 1 shows the servo movements according to the pulses at 0, 90 and 180 degrees.

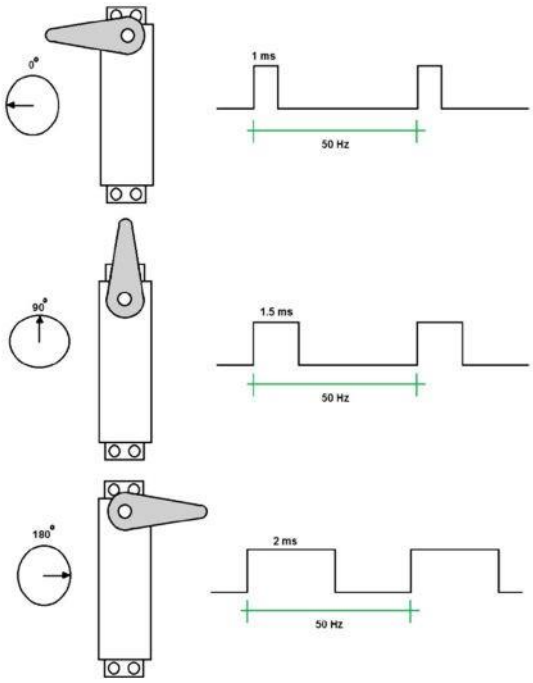


Figure 1: Servo motor rotatinal motion and pulse widh modulation

Communication between Intel Galileo board and the servo motors is made through a GPIO expander produced by a third party. It uses I2C communication protocol. This hardly contributes to obtain a servo motor working on a 1 degree increment at 50Hz frequency. This explained the problem that the previous student was having with the servo motors.

Figure 2 below show to interface 2 servo motors with intel Galileo 2. Both VCC are connected to 5V and both grounds to ground. One of the servos pulse is connected to pin 9 and the other to pin 3.

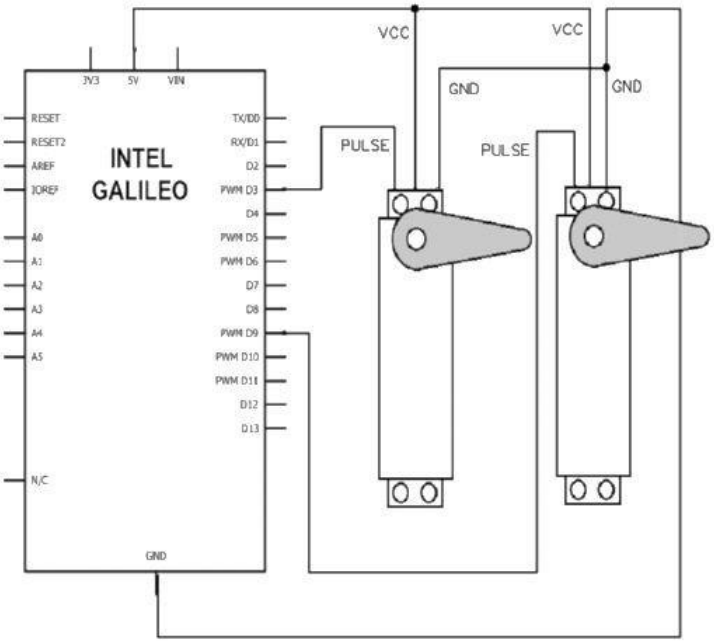


Figure 2: Interfacing two servos with Intel Galileo

It is worth nothing that Intel Galileo and Intel Galileo Gen 2 do not use microcontrollers to interface with the motors. This makes implementation easier even though other boards that run Linux (such as Beaglebones) do use a microcontroller to interface with pins and all have good performance.

CHAP III

METHODOLOGY

3.1 Hardware Architecture

There are three main components in this design.

The first component is the camera. For this project, Logitech 270P webcam is used due to its compatibility with linux OS. It is used to record video and sent to Intel Galileo board. The video is processed by the embedded platform to detect the presenter's face and determine it location within a predefined window. If the presenter moves, the board will provides PMW pulses to the Servo Motors to move the camera to follow the presenter on its new location.

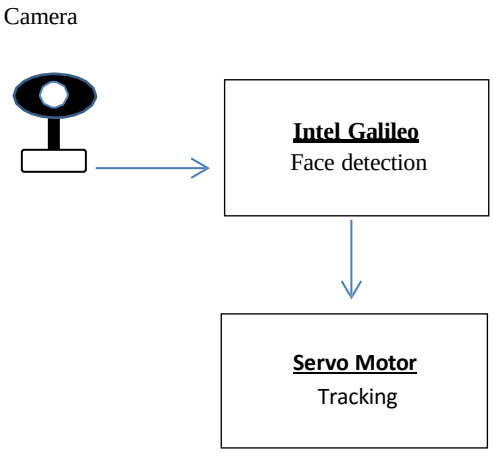


Figure 3: Hardware architecture

3.2 Implementation Task

To successfully carry out this project, some specific activities need to be defined and carried out.

- Define clear product requirements
- Review related works and identify suitable theory to optimize the system
- Test existing prototype to acquire information on the algorithm working principles, hardware and software processing speed and pitfalls
- Design a new system to be implemented on Intel Galileo Board
- Break system in to small subsystems that include Camera interface system, Video processing system, and Servo motor control system.
- Test and debag subsystems
- Integrate subsystems to produce a working prototype.
- Create necessary reports and documentation

3.2.1 Product requirements:

- Perform face detection and tracking using camera
- Must be a standalone device
- Must run on an embedded OS (Intel Galileo Board)

3.2.2 Test of existing prototype

This work was previously done by Suraya Nafila from Universiti Teknoogi Petronas.

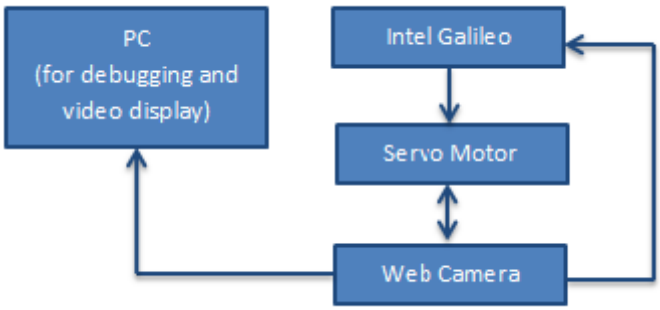


Figure 4: Previous architecture

The student managed to succeed detection and tracking using openCV on a computer. However its implementation on Intel Galileo did not work properly. The board could not establish communication with the webcam even though the webcam used (Logitech 270 P) is supported on linux. There are few reasons to that:

- The then existing firmware has a bug that prevented Linux from booting from the SD card. [8]
- In the used SD image, the Python-OpenCV module does not work unless a LSB linux image is built.[9]
- Since C/C++ programing was used, the codes could only be compiled on a host computer because the existing binaries works with difficulties unless build tools are installed on the image too. [9]

As a result, few actions are recommended:

- The firmware must be updated to the latest release.
- Build a new SD image using Poky a build tool provided by the Yocto project.

3.3 New Design implementation

The objective of the new design as shown in figure 5 is to have the software run on Intel Galileo Board. To achieve that there a need to solve problems faced by the previous student by updating the firmware and by building a new SD image based on the yocto project.

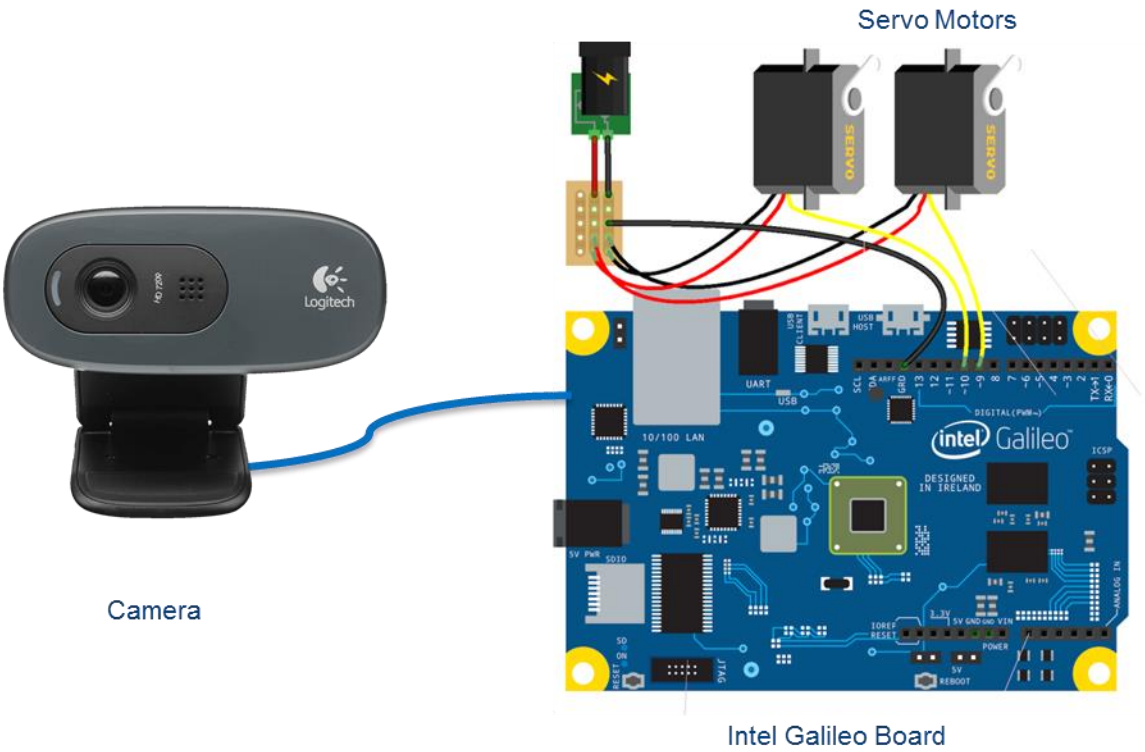


Figure 5: hardware system design

3.3.1 Building SD Image

3.3.1.1 Requirement:

- At least a 2 GB SD card
- A Linux terminal

3.3.1.2 About the Yocto project

. The new firmware allows the webcam to be detected but the program is still not running. This is due the programing language used on the SD image. Therefore a there maybe need to rebuild the SD image.

Yocto was used to build the linux image.

The Yocto Project is an open-source collaboration project focused on embedded Linux developers. Among other things, the Yocto Project uses a build system based on the OpenEmbedded (OE) project, which uses the BitBake tool, to construct complete Linux images. The BitBake and OE components are combined together to form Poky, a reference build system. It was created to be a flexible and customizable platform for different hardware architectures and code. The Yocto project brings a series of tools, methods, and code that allows choosing the target CPU (the software and hardware components) and the footprint size to build a software release based in the Linux operating system. Among the CPU supported are the Intel Architecture (IA), ARM, PowerPC, and MIPS.

Poky is the name given to the build system in a Yocto project. Poky depends on a task executor and scheduler called the bitbake tool.

3.3.1.3 Linux image Build steps:

The image is built using a 64 bit Linux system version 14.04.4. The following steps were executed:

- Step 1: insure all development tools are available mainly a GCC compiler and a 7Zip utilities. The current system comes with a GCC compiler.

  sudo apt-get install build-essential 7zip-full

  The above code was used to install 7zip.
- Step 2: 'sdimage' directory was created for the build process
- Step 3: the build support packages was downloaded from Intel The right BSP was required. Since Intel Gen 2 used runs the Intel Quartz processor, the right BSP can be downloaded from https://downloadcenter.intel.com/download/23197/Intel-Quark-BSP.
- Step 4: unzip the file then *meta-clanton_v0.7.5.tar.gz* and finaly change

directory to *meta-clanton_v0.7.5.tar.gz*

- Step 5: The following was run *./setup.sh*

- Step 6: bitbake image-full

  This the last code to use to build the image after which the image is copied to the SD card.

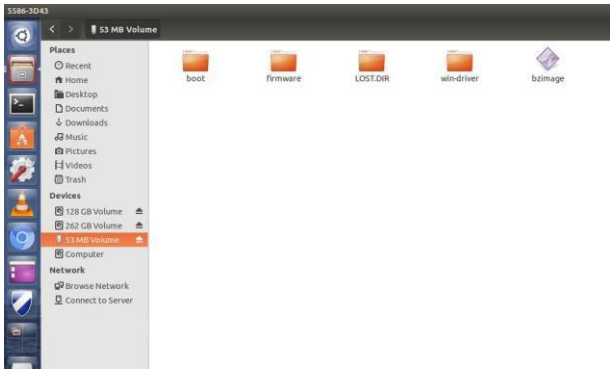


Figure 6: SD Image result

Fig 7 above shows the result of SD card after successfully building the image.

3.3.2 Development settings

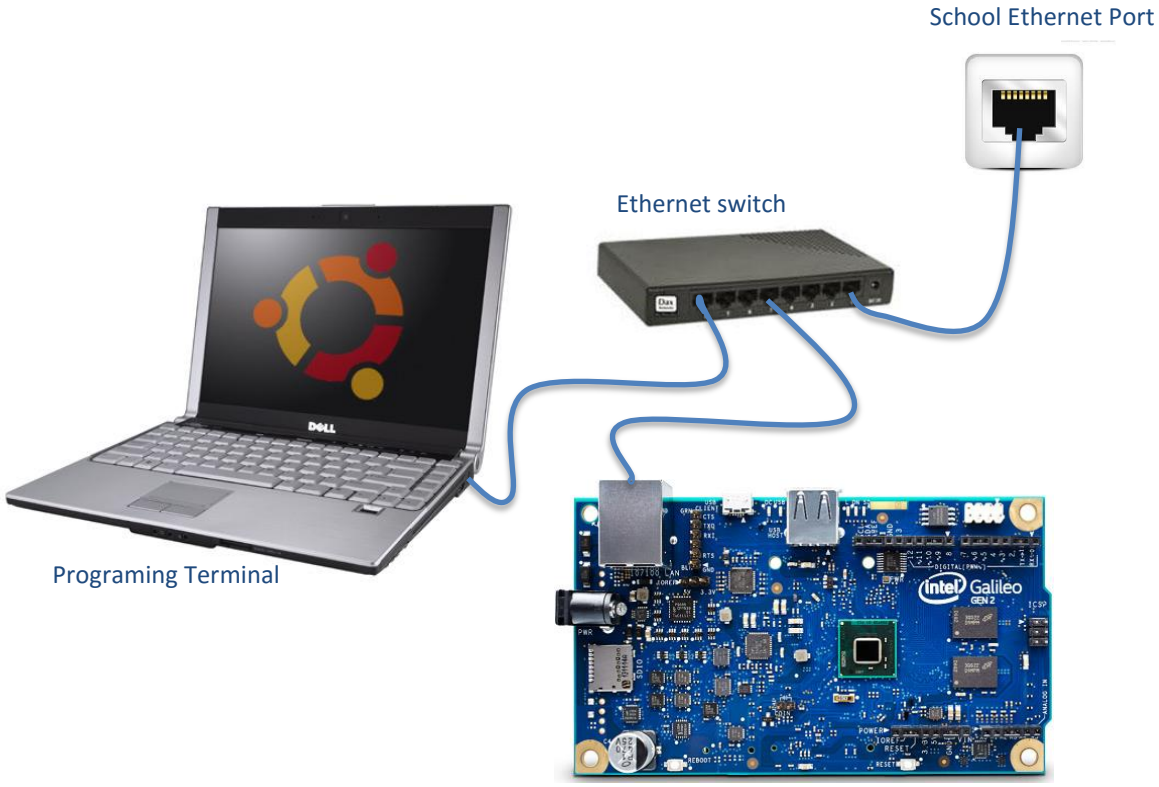


Figure 7: Development set-up

3.3.2.1 Requirements

- Laptop with Linux OS. There are many Linux distributions out there. The one used for this project is Ubuntu 14.04
- Ethernet switch (optional): the switch in not necessary needed. The connection to the board can be established by connecting both the board and the pc to the local Area Network (LAN).
- Galileo Board: of course the development board
- A LAN port: for this project Universiti Teknologi Petronas LAN is used.

3.3.2.2 Set-up:

- Connect the switch to the LAN port using an Ethernet cable
- Connect the Pc to the Switch and the Boards to the switch through its Ethernet port.
- Insert the SD card to the board
- Power-up the board. Since Linux OS is a little slow, wait for about 2 minutes for the Linux image to boot.

3.3.2.3 Logging into the board:

3.3.2.3.1 Discovering the board's IP address

To discover the board's IP address ARP packet scanner was used. The ARP scan tool, also called MAC shows every active IPv4 device on the subnet even if they have firewalls. It works best on the Local Area Network. Below are the steps to getting the boards IP:

3.3.2.3.2 Install arp-scan:

Open a terminal and type:

```
sudo apt-get update
sudo apt-get install arp-scan
```

These two codes will update the list of available packages and their versions and install arp-scan

3.3.2.3.3 Determine Intel boards MAC address.

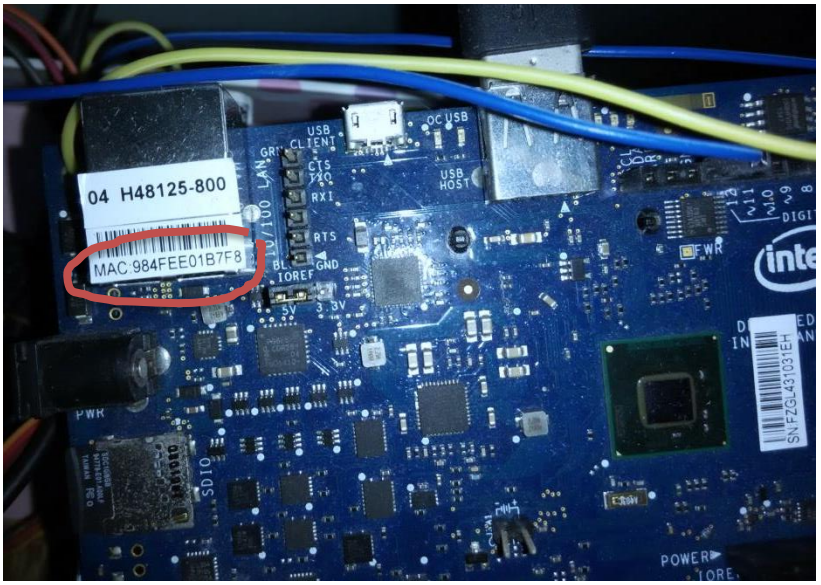


Figure 8: Mac address

The MAC address for the Board I am using is 98:4f:ee:01:b7:f8 as shown in the picture above.

3.3.2.3.4 Scan for all hosts in the network using the following codes

```
sudo arp-scan --interface=eth0 --localnet
```

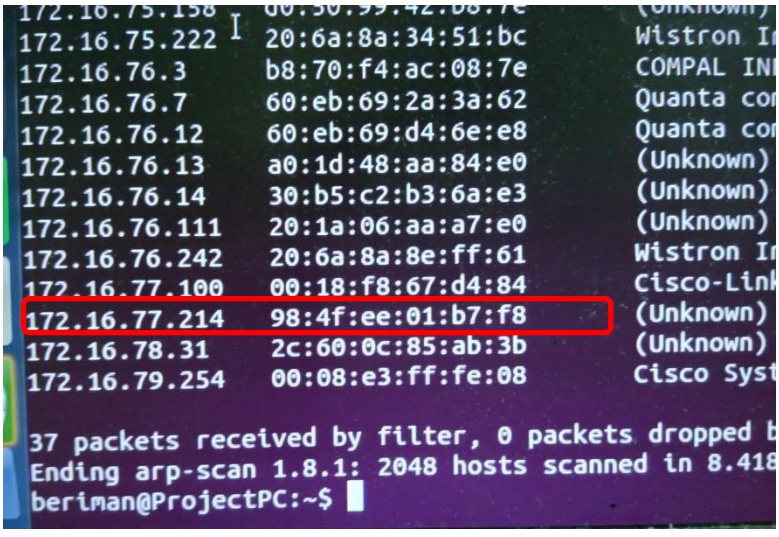


Figure 9: IP address

As the result shows, the board with the MAC address 98:4f:ee:01:b7:f8 was assigned an IP address 172.16.77.214.

3.3.2.3.5 Access the Board:

Accessing the linux image on the SD card give us an opportunity to do many this. We can access the board console, write and/or run scripts directly from the board. To do that, type *sudo ssh @my_device_IP_address.* Since the current device's IP address in this case is 172.16.77.214 as shown in the picture, we can use

```
sudo ssh @172.16.77.214
```

This landed us directly to the home page of the board SD linux OS.

From here we can navigate through the system using normal linux commands sush as ls to list files and folders in the current directory or cd to change directory. For instance, we can use *cd /* to access the root directory.

3.3.3 Connecting the board to arduino IDE

- Connect the board to the PC using a micro USB cable
- See appendices A 'Installing the arduino IDE' to install and start working with arduino. Alternatively, the instruction provided on Intel developer zone at https://software.intel.com/en-us/iot/library/galileo-getting-starte can also be useful to get started with arduino IDE.
-

3.3.4 Run facetracker.

Facetracker is an arduino sketch. A sketch is the name that Arduino uses for a program. It's the unit of code that is uploaded to and run on an Arduino board. Intel Galileo was made to be compatible with arduino board allowing arduino sketch to be executed on te board.

Facetracker sketch is provided in project folder. Access the project folder and open the folder named arduino and another folder called facetracker. Facetracker folder contains 2 files: facetracker.ino which is the main program and fdmap.h which is the header file used to communicate with facedetect.

- Load facetracker.ino, compile and upload to Intel Board.

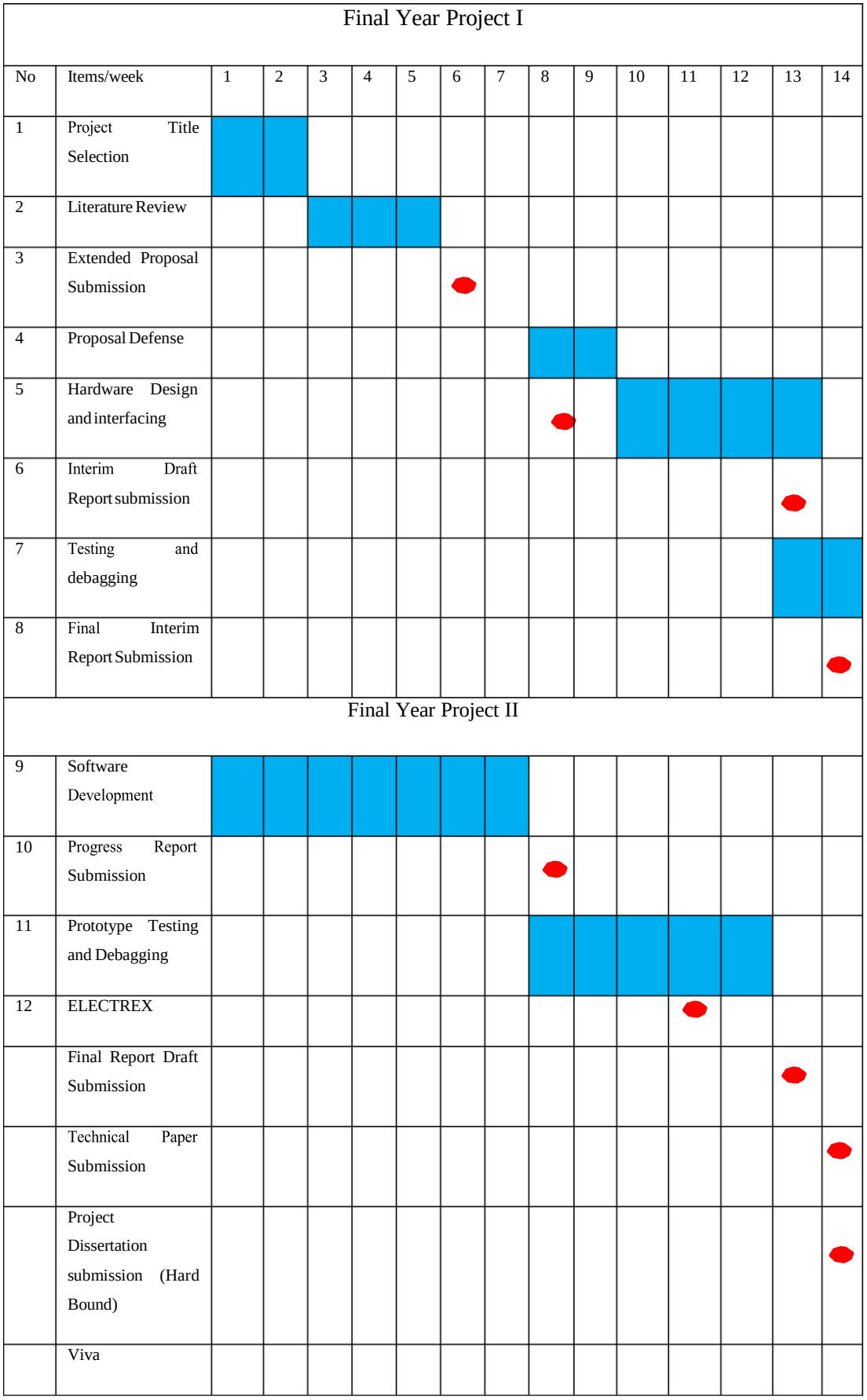
3.3.5 Run face detection software

Nano was used as the editor to write face detection code. GNU Nano is a small and friendly text editor. Besides basic text editing, nano offers many extra features like an interactive search and replace, go to line and column number, etc…
Once edited, the code is built using OpenCV and pthread libraries to produce an executable file.

To execute facedetection,

- SSH into the board using ssh @my_device_IP_address
- Navigate to facedection software directory using cd /fyp/galileo/run/
- Execute the script named build.sh using command ./build.sh
- Sit 1 meter away and face the webcam.
- Stay still for about 50 seconds, allowing the camera to initialize its pan and tilt movements. This is called auto calibration.
- Move head up or down in a slow motion while making sure the face is detected.
- enjoy

3.4 Gantt chart



CHAPTER IV

RESULT AND DISCUSSION

4.1 Face detection

4.1.1 Capture and process single frames

As of now, the SD image can run on the board. Capturing video from the camera is a little buggy but single frame capture is successful. After applying face extraction features to the image frame, the following result is obtained with a processing speed set to 1 fps (frame per second):

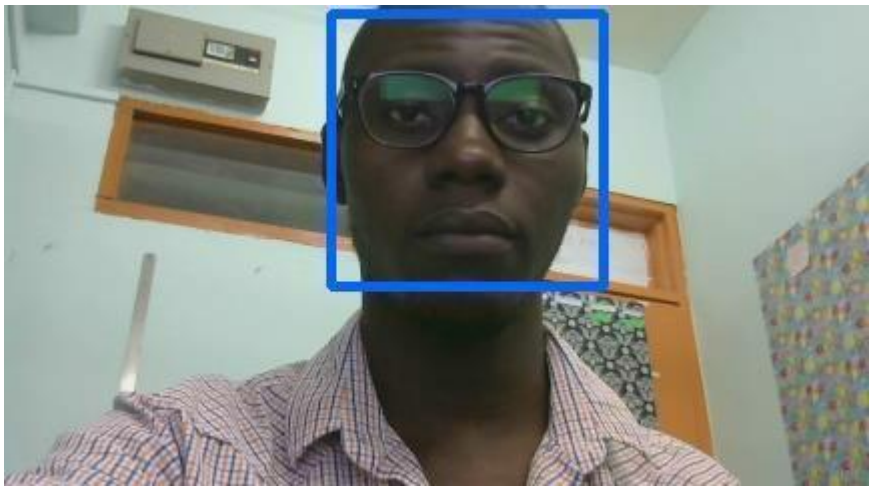


Figure 10: Straight face position

After applying face extraction features to the image frame, the result showed in fig.1. is obtained with an execution time of 1100 Ms.

To detect the face from the frame however, the camera must be placed at close proximity to the presenter and the presenter's eyes must be directed to the camera. There are few cases in which face detection runs into complications.

4.1.2 Face detection problems

Detecting face using our device becomes buggy when the presenter is not looking into the camera or when the presenter is far from the camera. Fig.8 shows that when the face is in profile position, no detection is made. In this case the frame is considered empty. To solve this issue without affecting further the processing speed of 1 fps, this particular image is considered to be containing no human face. The frame is then discarded and a new frame is captured and processed.

Figure 11: Profile face position

In fig. 9 below, the person here representing our presenter is placed 2m away from the camera. No face is detected.

Figure 12: Far distance face detection

4.1.3 Detection precision and accuracy of multiple frame processing

To help determine the accuracy and precision of our face detection on Intel Galileo, we have considered 3 elements that affect our result: Glasses, distance and luminosity.

Table 1: Rate (%) of face detection of a presenter without glasses

Distance (Cm)	100	200	300	400	500
High luminosity	80	60	30	10	0
Low Luminosity	50	20	0	0	0

Table 2: Rate (%) of face detection of a presenter with glasses

Distance (Cm)	100	200	300	400	500
High luminosity	40	40	10	0	0
Low Luminosity	10	0	0	0	0

4.1.4 Continuous detection and latency

To allow tracking the presenter face using servo motor, a continuous face detection and face center position update is required. An infinite loop is added to the process. As a result of that, the processing time of the single frame is increased to 1600 MS when running with a windows scale of 1.5; resulting in a processing speed of 0.625 fps.

Once the center position of the face is determined, the values are written on a memory map which is then charred with the Arduino sketch for motor control. The whole process result in an average latency of approximately 8000ms – that is it, the time required to process 5 frames.

To understand latency, assume the webcam is capturing an empty window with no face. Now when a face shows up at location A at time t0 and move to location B at time t1, it will take 8000ms for values of the center position of the face at position A to be updated and another 8000ms for the face at position B.

4.2 Face tracking

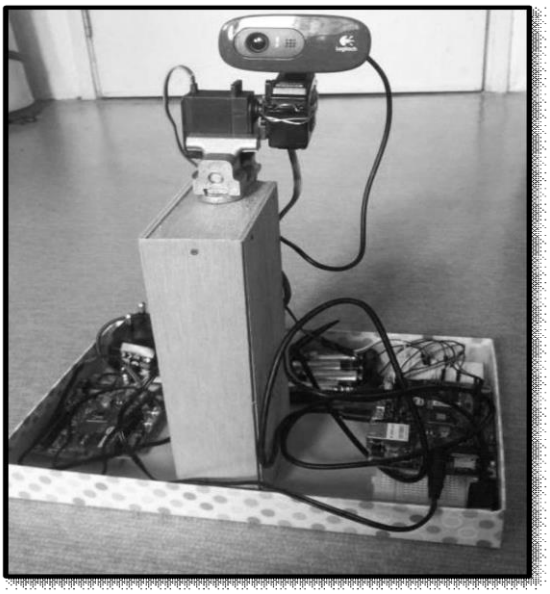


Figure 13: Hardware Prototype

Controlling servo is the final output of this whole project. The Arduino sketch is nicely integrated with the face detection code. This means that the values of the center position of the face can be accessed by the sketch to move the motors accordingly. A delay of 8 seconds can be observed due to latency between the two processes.

CHAPTER V

CONCLUSION AND RECOMANDATIONS

The implementation of a real time face detection and tracking on personal computers has seen tremendous successes over the years. This is mainly due the high computing power of the computer processor on which they run. Implementing a face detection system on an embedded platform though can be quite challenging due the amount of load task that is required by the widely used algorithm and the limited processing power of embedded platform. Nevertheless, real time face detection systems are being implemented on FPGAs and since conferences are being held at this moment on the topic, we hope to be able to break through the system on Intel Galileo Board.

REFERENCES

1. Viola, P. and M. Jones, *Robust Real-Time Face Detection.* International Journal of Computer Vision, 2004. 57(2): p. 137-154.

2. Corp, I., *Intel OpenCV Library.*

3. Kim, M. and K.-Y. Kim, *Hardware Architecture for Real-Time Face Detection on Embedded Analog Video Cameras*, in *Computer Science and its Applications*, J.J. Park, et al., Editors. 2015, Springer Berlin Heidelberg. p. 311-316.

4. Lienhart, R. and J. Maydt. *An extended set of Haar-like features for rapid object detection*. in *Image Processing. 2002. Proceedings. 2002 International Conference on*. 2002.

5. Cho, J., et al. *FPGA-Based face detection system haar classifiers*. in *Proceedings of the 7th ACM SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA'09*. 2009.

6. Hiromoto, M., H. Sugano, and R. Miyamoto, *Partially Parallel Architecture for AdaBoost-Based Detection With Haar-Like Features.* Circuits and Systems for Video Technology, IEEE Transactions on, 2009. 19(1): p. 41-52.

7. Changjian, G. and L. Shih-Lien. *Novel FPGA based Haar classifier face detection algorithm acceleration*. in *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*. 2008.

8. corp, I. *Intel Galileo - Building Linux Image*. [cited 2015 23 july 2015]; Available from: http://www.malinov.com/Home/sergey-s-blog/intelgalileo-buildinglinuximage.

9. corp, I. *OpenCV projects*. 23 July 2015]; 2015:[Available from: https://communities.intel.com/message/220707.

10. Intel® Galileo and Intel® Galileo Gen 2: API Features and Arduino Projects for Linux Programmers, Manoel Carlos Ramon, Apress Media, 2014.

APENDICES

A- Installing the Arduino* IDE

This guide contains steps to install the Arduino IDE on a system with Linux*.

Requirements

- You have connected your board to your computer and gathered any required components.

1. To check if you have Java installed, open a terminal and enter the command:

   java

2. If you see the above message, you do not have Java installed and you will need to install it. To install the Java package, enter the command:

   sudo apt-get install default.jre

   You may be prompted to enter your user password.

3. Download the Arduino IDE from the Arduino Software page. Be sure to download the version for your operating system. Newer versions of Linux will use a .txz rather than a .tgz file type.

4. Navigate to the folder where you copied the Arduino IDE .tgz or .txz file and double-click it to open the archive.

   You can decompress the file from the command line. To do so, use xz by entering the following command:

   unxz *filename*

   Where *filename* is the name of the file to decompress. For example: unxz IntelArduino-1.6.0-Linux64.txz

   If you don't have xz installed, install it by entering one of the following commands:

   - For Ubuntu or other Debian-based machines:

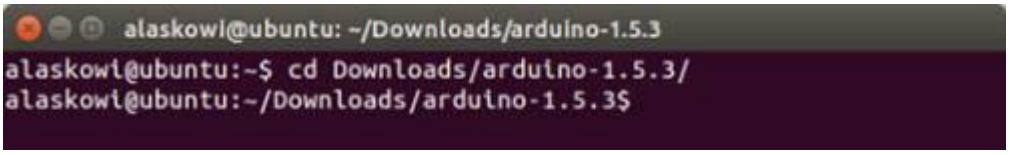
   sudo apt-get install xz-utils

5. Click Extract and navigate to the directory where you would like to unzip the Arduino IDE. In this example, we will leave it in the Download directory. Click Extract.

6. The extracted folder should contain a file named arduino, as well as several folders.

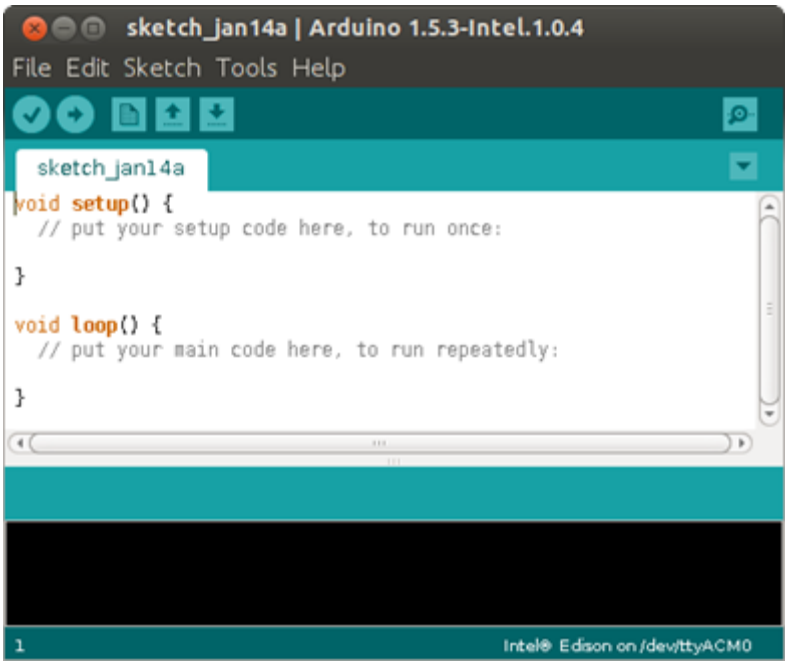
7. Open up a new Terminal window.
8. Navigate to the Arduino IDE folder. In this example, the command will be cd Downloads/arduino-*x.x.x*/, where *x.x.x* is the Arduino IDE version number you downloaded.

   Note: When you start typing in cd Downloads/arduino, you can press Tab to auto-complete the folder path.



9. To run Arduino with administrator privileges, enter the command: sudo ./arduino.

   If prompted, enter your password. The Arduino IDE opens.



10. In a Terminal window on your host machine, check the availability of the /ttyACM* port by entering the following command:

    ls /dev/ttyACM*

    Note: If the /ttyACM* port is not available, here are several possible reasons why:

    - The modem manager is using the port. When the port becomes active, the modem manager can claim the port, blocking the IDE's access to the port. The exact command to remove it will depend on your Linux distribution. For example, the command

    sudo apt-get remove modemmanager

    may work.

    - The /ttyACM port was not created automatically when you plugged in your board. To add the port, do the following:
        a. Create a file: etc/udev/rules.d/50-arduino.rules
        b. Add the following to the file:

    KERNEL=="ttyACM[0-9]*", MODE="0666"

        c. Restart udev by entering the following command:

    sudo service udev restart

    If you are using a virtual machine (VM), you may need to reboot Linux

within the VM.

- If you still are not able to see the port in the IDE, it may be because your user hasn't been added to the dialout group. Add yourself to the dialout group by entering the following command:
- sudo adduser your_user_name dialout

Then restart the IDE and try again.

11. Choose Tools > Port, then verify that your port and device are selected. The entry should be similar to:/dev/ttyACM0 (Intel® Edison).
12. Choose Tools > Board > Boards Manager. The Boards Manager opens.
13. In the list of boards, select your board type:

- For the Intel® Edison board, select Intel i686 Boards.
- For the Intel® Galileo board, select Intel i586 Boards. From the Select version drop-down list, select1.6.2+1.0.

14. Click Install to install the board definition package for your board.
15. When the installation process is finished, click OK.

APENDIX B – FACEDETECTION CODE