Implementing a Smart Door Bell with PIN Code and Automatic Face Recognition (EECS 4221 Project)

Ege Cakmak
Christopher Yip

Table of Content

1. Introduction 2

2. Overview of Debian 3

3. Features of Debian 3

4. Overview of BeagleBone Black 4

5. Features of BeagleBone Black 4

6. Set up BeagleBone Black with Debian Linux or Angstrom Linux 6
   - Download a Debian or Angstrom software image for BeagleBone Black 6
   - Write the download image to a microSD card 6
   - Boot Your BeagleBone Black off the SD card 7
   - Check if the setup was successful 7
   - Advanced Setup 8
     - Browse to Beagle Board 8
     - Expand Storage 8
     - Connect to Internet 8
     - Install new applications 9
     - Setting up x11 forwarding over SSH 9

7. Developing Software on BeagleBone Black 9
   - Cloud9 IDE 9
   - Qt Creator IDE 10
   - JavaScript and BoneScript Library 10

8. Developing an Application on BeagleBone Black (Smart Door Bell) 11
   - Setup Phase for the Smart Door Bell 11
   - Deploying the Smart Door Bell Application 19

9. References 25

Introduction

In this project manual we will be implementing a Smart Door Bell with two ways to authenticate the user, these are PIN Code and Automatic Face Recognition respectively. Firstly, we will talk about the specifics of BeagleBone Black and the operating system we will be using for this project, Debian. Secondly, we will show how to install Debian on BeagleBone Black and how to set it up. Lastly, we will mention about our project and how it can be set up to work on this board.

Our Smart Door Bell Project consists of two ways to unlock the door as the title suggests. The PIN Code authentication will be done via the buttons we will be connecting to our board using a bread board. We will have a C++ program to detect the button inputs and blink the leds on the board to inform the user. This program will also have a simple UI made with Qt that will be showing the status of the door.

The Automatic Face Recognition will be done with the help of the computer vision library OpenCV. This method will store registered users' pictures in a folder with each user's name respectively. We will use these pictures to train our face recognition machine learning model to be able to recognize faces. This method also supports a separate remote Web UI developed in Python and Flask which shows a live feed of the camera and it will also unlock the door automatically if the face recognized is authorized to unlock the door. Further, this UI also allows users to unlock the door manually and allows them to configure who can open the door automatically or not.

Overview of Debian

Debian is a UNIX like open source operating system developed by “The Debian Project”. The initial release of Debian was 25 years ago [10] and it is still being updated regularly. [1] It supports many platforms such as amd64 and arm. Further, it supports 75 languages and uses APT and dpkg for managing packages. In addition, by default it comes with GNOME and XFCE desktop environments, however these can be changed easily by the user. Lastly, Debian’s kernel consists of two parts, these are monolithic and microkernel respectively. More information can be found at www.debian.org.

Features of Debian

- Debian has access to repositories that contain over 51000 packages and this makes it the largest collection of software in the whole World. [11]
- Debian is a popular linux distro for servers. [12]
- Debian has a both monolithic and microkernel kernel structure. [14]
- Debian has 9 main releases so far. [13]
- Debian is free. [13]
- Many Linux distros such as Ubuntu and Linux Mint are based on Debian. [15]

Overview of BeagleBone Black

The BeagleBone Black (BBB) is one of the low-cost single-board computers designed by a non-profit corporation located in Michigan, USA named BeagleBoard.org Foundation [1]. The board was released in 2012 with objective for developers to learn and collaborate together in the development of open-source software and hardware in embedded computing. Moreover, the corporation provides an online forum allowing owners of the board to share ideas and experience as well as provide support for those in help.

Compared to the numerous boards of the BeagleBone family, BBB is at even lower cost with focus on high hardware expansion and provides exclusive features that will be discussed later. BBB is now shipped with Debian Linux, but Ubuntu or Angstrom OS can also be installed by the user.

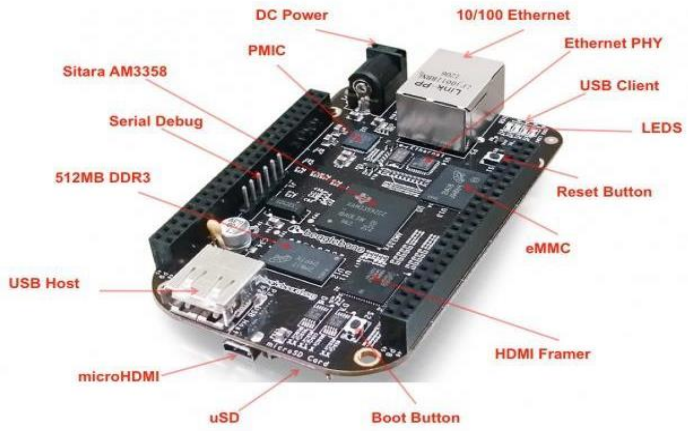
Features of BeagleBone Black

BeagleBone Black is classified as a single-board computer [2], meaning that it contains, in a single circuit board, all the minimum hardware required for a functional computer. Moreover, processor, memory and graphics acceleration chips are present on the board [3]. However, its size is significantly smaller than a regular desktop machine but still provide the same expandability.

BBB has a 1GHz ARM-based processor (Sitara), as well as 512MB of RAM [2]. Moreover, BBB provides HDMI output (up to 24fps video output and stereo audio output) whereas the regular BeagleBone does not. On the other side, BBB removes the on-board USB JTAG and serial adapters.

A great feature of the BBB is the 4GB 8bit embedded Multi-Media Controller (eMMC) it possesses in addition to the MicroSD slot [3]. This allows the OS to be flashed on the eMMC and the microSD to be used as storage or the possibility of switching OS images rapidly.

Furthermore, BBB is equipped with a mini USB port and a DC jack which can be used to power supply the board. In addition, it possesses an ethernet port for internet connection (which can also be provided via USB to virtual Ethernet as discussed later) and 4 user-controllable LEDs.



The table below provides a summary of the specifications of BBB [2].

	Feature	
Processor	Sitara AM3358BZCZ100 1GHz, 2000 MIPS	
Graphics Engine	SGX530 3D, 20M Polygons/S	
SDRAM Memory	512MB DDR3L 800MHZ	
Onboard Flash	4GB, 8bit Embedded MMC	
PMIC	TPS65217C PMIC regulator and one additional LDO.	
Debug Support	Optional Onboard 20-pin CTI JTAG, Serial Header	
Power Source	miniUSB USB or DC Jack	5VDC External Via Expansion Header
PCB	3.4" x 2.1"	6 layers
Indicators	1-Power, 2-Ethernet, 4-User Controllable LEDs	
HS USB 2.0 Client Port	Access to USB0, Client mode via miniUSB	
HS USB 2.0 Host Port	Access to USB1, Type A Socket, 500mA LS/FS/HS	
Serial Port	UART0 access via 6 pin 3.3V TTL Header. Header is populated	
Ethernet	10/100, RJ45	
SD/MMC Connector	microSD , 3.3V	
User Input	Reset Button Boot Button Power Button	
Video Out	16b HDMI, 1280x1024 (MAX) 1024x768,1280x720,1440x900 ,1920x1080@24Hz w/EDID Support	
Audio	Via HDMI Interface, Stereo	
Expansion Connectors	Power 5V, 3.3V , VDD_ADC(1.8V) 3.3V I/O on all signals McASP0, SPI1, I2C, GPIO(69 max), LCD, GPMC, MMC1, MMC2, 7 AIN(1.8V MAX), 4 Timers, 4 Serial Ports, CAN0, EHRPWM(0,2),XDMA Interrupt, Power button, Expansion Board ID (Up to 4 can be stacked)	
Weight	1.4 oz (39.68 grams)	

Set up BeagleBone Black with Debian Linux or Angstrom Linux

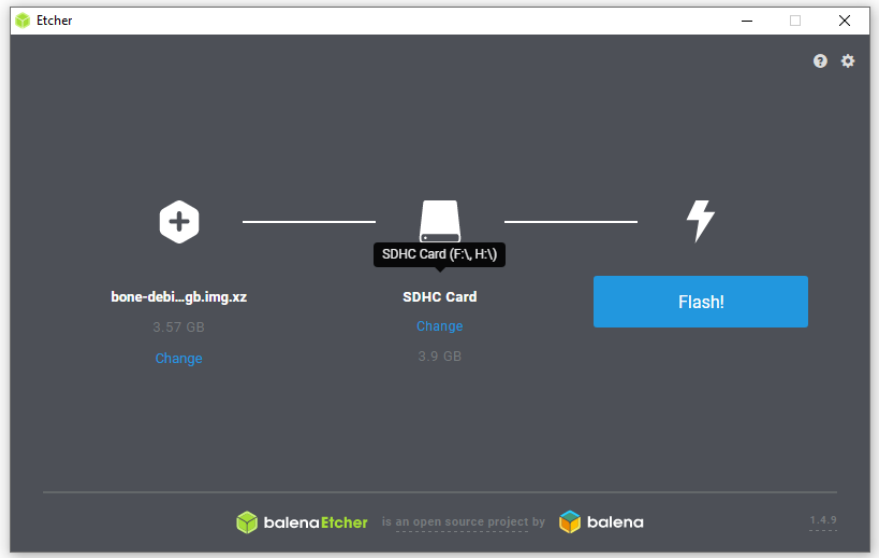
The following steps will help you to install the latest available Linux OS (Debian or Angstrom) on BBB [3].

1. Download a Debian or Angstrom software image for BeagleBone Black

- Official images are available on the webpage: https://beagleboard.org/latest-images
- Several versions of firmware images are available but the latest version is recommended.
- For Debian, choose LXQT version if you are going to connect the board to a screen. The IoT version is however available and provides more disk space at the expense of the GUI.
- For Angstrom, you have the choice between eMMC flasher version which allows you to flash the image to the eMMC of the board (can also be done for Debian as explained in step number 3) or microSD version to boot from SD card.

2. Write the download image to a microSD card

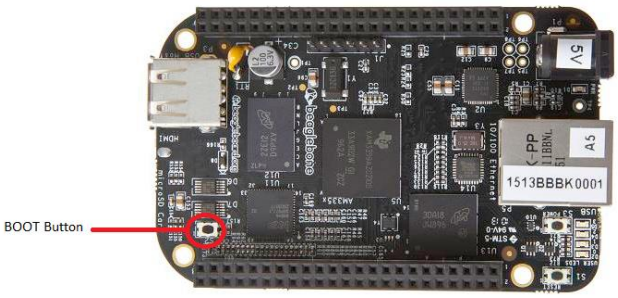
- You can use Etcher (from: https://www.balena.io/etcher/) or any other image writer to write the image onto the card. (Note: you may need to extract the image if you are using a different image writer)
- Your SD card needs to be at least 4GB to hold the image.
- **Warning**: Carefully select the appropriate drive when writing the image as the content of the drive will be overwritten.



Etcher Intuitive User Interface

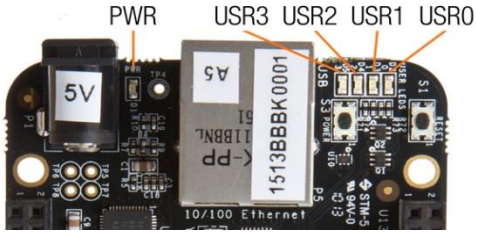
3. Boot Your BeagleBone Black off the SD card

- After ejecting the SD card from your computer, insert it into the powered-down board.
- Because BBB may include a programmed on-board flash, you need to press the BOOT button to boot from the SD card.
- Apply power (via USB cable or a 5V adapter) while pressing the BOOT button for 5 seconds.

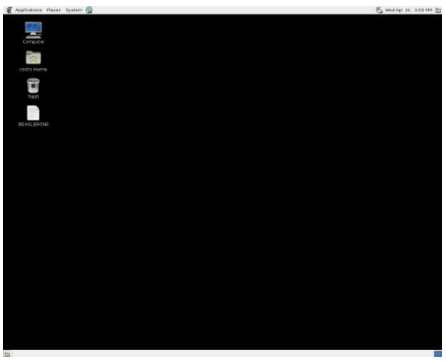


- For Debian, if you wish to flash the image onto the eMMC of the board [5], after logging as debian (default password: temppwd), edit the file /boot/uEnv.txt with root access, locate and uncomment the line below (remove '#'):
  #cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh
  and reboot the board to start flashing the eMMC.
- For Angstrom, if you download an eMMC flasher image, the flashing will start upon the first boot.
- Flashing to the eMMC may take up to 45 minutes and the USER LEDs will be steady on when it is done. You may now power off the board and remove the SD card before powering on again.
- **Warning**: if you flash the eMMC, make sure to remove the SD card before powering on again as it will keep re-flashing the eMMC.

4. Check if the setup was successful



- When power on, the power LED should light steadily and after a few minutes, the 4 other LEDs should blink in their default configuration:
  - ❖ USR0 blink in a heartbeat pattern
  - ❖ USR1 lights during SD card accesses
  - ❖ USR2 lights during CPU activity
  - ❖ USR3 lights during eMMC accesses
- If you are using BBB via HDMI, the screen should appear as on the left:



- If BBB is connected via USB to your computer, you should now be able to successfully ping beaglebone.local as the board will be running a DHCP server.
  Note:
  You may need to eject BEAGLE_BONE drive to start the network.
  For Angstrom, you may need to install the appropriate driver on your computer (available on: http://beagleboard.org/getting-started)
  If you are using Windows and you encounter an error during the installation of the driver, it may be because the driver is unsigned [6]. Install driver from:
  https://github.com/beagleboard/beaglebone-getting-started/tree/master/Drivers/Windows

5. Advanced Setup

5.1 Browse to Beagle Board

If the board is connected to a computer via USB, a network adapter should show up on your computer.

If you connected the board via or USB to virtual Ethernet or to a router via ethernet, you may now browse to the web server running on the board.

Using Chrome or Firefox browser, go to http://beaglebone.local/ which will provide further information on how the board can be used. Moreover, you will now have access to Cloud9 IDE (on http://beaglebone.local:3000/) which can be used to access command line or develop application on your board (to be further discussed later).

You can also ssh to your board using username 'debian' and default password 'temppwd' for Debian and username 'root' and empty default password for Angstrom.

5.2 Expand Storage

If you are booting from SD card and have available space on the card, it will not show up on the system. To expand storage, you may run the provided script from https://github.com/RobertCNelson/boot-scripts/blob/master/tools/grow_partition.sh and on reboot, you should have access to the extra storage on the SD card [7].

5.3 Connect to Internet

Connecting the board to a router with internet connection via ethernet should be enough to provide internet access to your board. However, you can also do the same using the USB to virtual Ethernet feature [8].

On Windows, you need to setup a network configuration to share the current network access.

1. From control panel, go to "Network and Internet" -> "Change adapter settings"
2. Right click on your current internet access network and click on "properties" -> "sharing"
3. Tick the checkbox "Allow other network users to connect through this computer's connection" and select your Beagle board adapter. Click "OK" to save.
4. Select the Beagle board adapter and go to its properties. Select "Internet Protocol Version 4" and click on "Properties".
5. Select "Use the following IP address" and enter "192.168.7.1" with subnet mask "255.255.255.0" and click "OK" to save.
6. On your Beagle board, you need to issue a route command to specify how to connect to the web by updating the IP routing table. Run the command "sudo /sbin/route/ add default gw 192.167.7.1"
7. To configure a nameserver, edit /etc/resolv.conf and add the following lines:

   domain localdomain
   search localdomain
   nameserver 8.8.8.8
   nameserver 8.8.4.4

8. You should now be able to access internet! However, every time you power on the board, you need to perform steps 6 and 7.

5.4 Install new applications

To install new application, first you need to run the command “sudo apt-get update” once. After that, you can install any application package available using “sudo apt-get install <name of package>”.
Recommended applications for C programming/debugging are gdb and gdbserver (to be used later in the guide).

5.5 Setting up x11 forwarding over SSH

X11 forwarding over SSH allows a graphical user interface when using SSH. It can be done by installing “xorg” on the beagle board and “XMing” on the computer [9]. Then using putty, you should be able to securely access graphical applications over the Internet.

Developing Software on BeagleBone Black

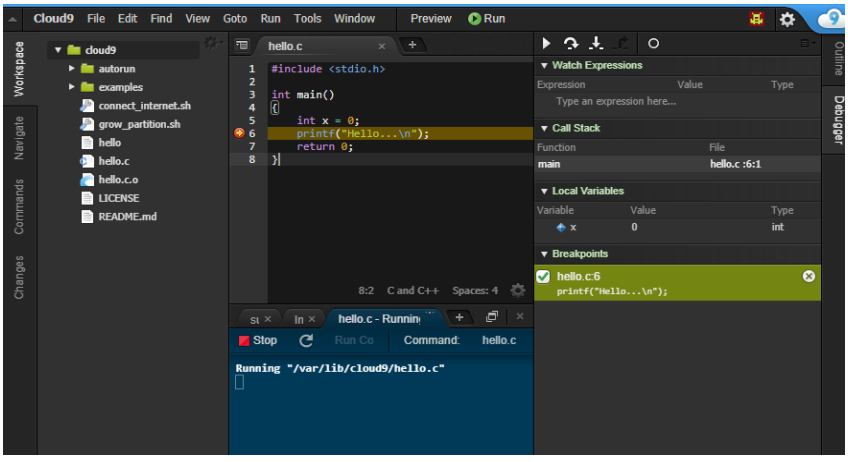
Cloud9 IDE

When you browse into your BeagleBone Black, you can access cloud9 IDE through the ip address http://beaglebone.local:3000. However, the functionalities differ depending on the OS image you are using.

On Angstrom, you are limited to only JavaScript programming, including the BoneScript library which will be discussed later.

However, with Debian Stretch image, you are able to do much more. You can compile and debug numerous languages such as C/C++, Java, Python, Go, etc. Moreover, you can also run command line script to navigate and/or use the full abilities of your board.

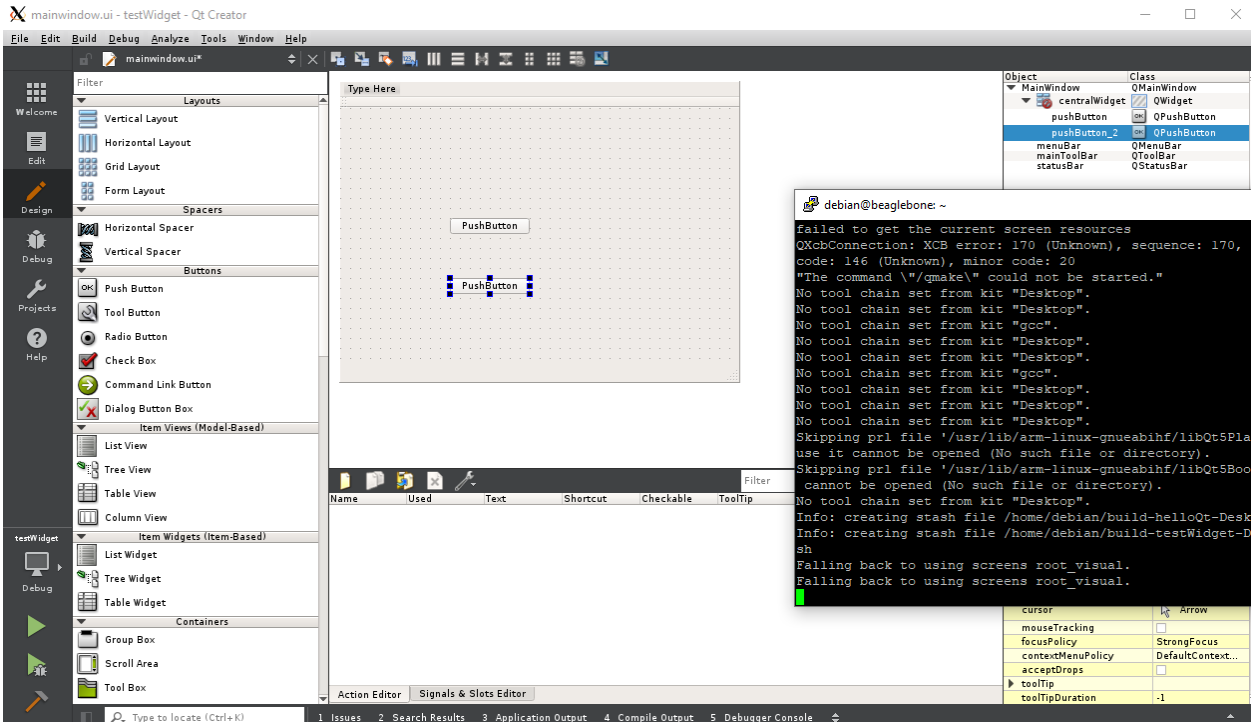
If you wish to debug your C/C++ program, you can do so on Cloud9 by installing gdb and gdbserver. Simply run “sudo apt-get install gdb” and “sudo apt-get install gdbserver”.



Qt Creator IDE

Alternatively, I would recommend to also install Qt creator which allows you to code in C/C++ and has a rich library for Graphical User Interface under GPL and LGPLv3 open source licenses. You can do so by simply running “sudo apt-get install qtcreator” and “sudo apt-get install qt5-default”
Together with x11 forwarding over SSH, you can use its user-friendly interface. However, one disadvantage compared to Cloud9 is that you will need extra space on your SD card.

The screenshot below shows Qt Creator running through ssh on BBB.



JavaScript and BoneScript Library

Using Cloud9 and the BoneScript library (already part of the provided image), you can control your board with more freedom such as controlling the USER LEDs. It provides a vast number of function exclusive to BBB that you can run immediately.

To use BoneScript library, use the "require('bonescript')" function call which will return an object containing all the functions exported by the library.

From here, you can use function such as digitalWrite(pin, value) which will turn on or off the USER LED indicated as “pin” depending on “value” (1 for on and 0 for off).
pinMode(pin, direction) allows you to select the pin/LED you wish to perform an action on and have to be use prior to digitalWrite.

You can also use getPlatform() which allows you to obtain information about your board as well as the BoneScript version you are using.

Below is a simple example provided by beaglebone.org to turn on all the 4 USER LEDs for 2 seconds.

```
var b = require('bonescript');
b.pinMode('USR0', b.OUTPUT);
b.pinMode('USR1', b.OUTPUT);
b.pinMode('USR2', b.OUTPUT);
b.pinMode('USR3', b.OUTPUT);
b.digitalWrite('USR0', b.HIGH);
b.digitalWrite('USR1', b.HIGH);
b.digitalWrite('USR2', b.HIGH);
b.digitalWrite('USR3', b.HIGH);
setTimeout(restore, 2000);
```

Developing an Application on BeagleBone Black (Smart Door Bell)

In this part, as we mentioned we have a guide on how to run the app we developed on the BeagleBone. As we have mentioned, we developed a Smart Door Bell with PIN Code and Automatic Face Recognition, we will now show how to set it up on a fresh installation of a BeagleBoard. Before we continue however, we need to install some required programs.

Setup Phase for the Smart Door Bell:

The following guide will help you install the required programs and configure your board. It also shows how to use a Android Device camera as webcam on the board as well. Please keep in mind that you should use a SD card that is at least 8 GBs.

1. Preparing for Setup:
   a) Increasing the Primary Partition Size:
   The default Debian image for BeagleBone Black only comes with a 4 GB partition. This means that you will only have 4 GB of space usable even if your SD card has storage greater than 4 GBs, unless you follow the below instructions.

   Available space in each partition can be checked using the following command.

   ```
   df -h
   ```

   If you have the BeagleBone Debian distribution with the desktop environment, you should notice that your primary partition is only 4 GBs and almost full.

To resize the primary partition, we will use the fdisk utility. To start this program, copy the following command and paste it to your terminal.

sudo fdisk /dev/mmcblk0

You will notice the application has started and waiting for your input. Type in “p” and hit enter. This will print out information about your primary partition. The output should look like the following. Notice the table at the bottom of the output and make sure you note the starting sector for partition /dev/mmcblk0. In our case it is 8192.

```
Command (m for help): p

Disk /dev/mmcblk0: 7.4 GiB, 7948206080 bytes, 15523840 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x18e8111b

Device         Boot Start     End Sectors  Size Id Type
/dev/mmcblk0p1       8192 7667712 7659521  3.7G 83 Linux

Command (m for help):
```

Now we will delete this partition. Proceed with typing “d” and hitting enter. Right after deleting the primary partition we will create a bigger one. To do so, type “n” and hit enter again. For the first questions, leave them blank and proceed by just pressing enter. However, for the starting sector make sure that you type in the starting sector we noted in the previous step. In our case, we type in “8192” carefully and press enter. For the last question just proceed with pressing enter.

If you did everything correctly now your size of the primary partition should increase to the size of the whole card. Type in “w” and press enter to commit the changes we have just made. After this step the program will exit. Before we are done there is an extra step we need to follow, however for this we need to reboot our board. To reboot type in the below command to your terminal and press enter.

sudo reboot

For the final step, you need to run the following command.

sudo resize2fs /dev/mmcblk0p1

Finally, you should have your primary partition as big as the the size of your SD card if you followed the above steps carefully.

b) Creating a Swap File:
Our board has only 512 MB of RAM, which might cause us to run out of memory while installing programs.

The following command will create this swap file.

sudo fallocate -l 1G /swapfile

We also need to set the permissions for this file.

sudo chmod 600 /swapfile

To enable the swap file, use the following commands.

sudo mkswap /swapfile

sudo swapon /swapfile

The above series of commands create a swap file of size 1 GB. Make sure that you have enough storage.

c) Adding a Required Repository:
One of the programs we will be installing is only available on an Ubuntu archive repository. This repository can be added as follows.

echo "deb http://us.archive.ubuntu.com/ubuntu/ yakkety universe" | sudo tee -a /etc/apt/sources.list

d) Updating the Repository Cache and Upgrading Existing Software:
Before we get started installing the dependencies, we need to update our repository cache and upgrade the software already installed on the board. This can be done as follows.

sudo apt-get update & sudo apt-get upgrade

e) Installing Dependencies and Required Programs:

a) Installing Linux Headers:

The default Debian image for BeagleBone Black does not come with linux headers. Headers are basically an interface between internal kernel components and also between userspace and the kernel. [16]

It can be installed as follows. This step is mandatory as linux-headers are required by one of our dependencies.

sudo apt-get install linux-headers-`uname -r`

b) Installing gstreamer:
The gstreamer app will transmit the video outputted by our Android device to a video device in our Linux environment.

Use the following command to install gstreamer.

sudo apt-get install libgstreamer1.0-0 gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-bad gstreamer1.0-plugins-ugly gstreamer1.0-libav gstreamer1.0-doc gstreamer1.0-tools gstreamer1.0-x gstreamer1.0-alsa gstreamer1.0-pulseaudio

c) Installing v4l2loopback:
This application will help us create a loopback device which will be taking the stream provided by gstreamer. We will need to follow a different path to install v4l2loopback.

First, we will create a folder called “v4l2loopback”. This is where we will store the installation files for this application. Make sure you are in your home directory by simply running “cd” before running the following command.

mkdir v4l2loopback

After creating the folder navigate to this directory using the below command.

cd v4l2loopback

Next, download the required installation files using the below command.

git clone https://github.com/umlaeute/v4l2loopback.git

This command will create another folder including the installation files with the name v4l2loopback inside our already existing v4l2loopback folder. Simply run the following command to navigate into this folder.

cd v4l2loopback

Once you navigate into the folder, simply run the following commands to complete the installation.

make && sudo make install

sudo depmod -a

d) Installing Some More Python Modules and Dependencies:
Simply run the following commands to install these.

sudo apt-get install v4l2loopback-utils ffmpeg libjasper-dev

sudo pip3 install -U numpy

sudo apt-get install python-scipy python3-scipy

sudo apt-get install python-sklearn python3-sklearn

sudo pip3 install -U argparse cv2 imutils time flask camera

e) Installing the pyfakewebcam Module:
This Python module will allow us to create a fake webcam and stream the frames we will be getting from OpenCV to a video device.

We will now get started installing pyfakewebcam. We will be following a series of commands very similar to the ones we used to install v4l2loopback.

Create a folder named pyfakewebcam.

mkdir pyfakewebcam

Download the required installation files.

git clone https://github.com/jremmons/pyfakewebcam.git

Navigate into the nested pyfakewebcam folder.

cd pyfakewebcam

Since we already installed numpy we can tell the installer for pyfakewebcam to omit installing numpy. To do so edit the setup.py file using the below command and remove lines 11,12 and 13.

nano setup.py

Once you are done editing do “CTRL + X”, type “Y” and press enter to save your changes.

Finally run the following command to install pyfakewebcam module.

sudo python setup.py install

f) Installing OpenCV:
We will be using OpenCV in this project to recognize faces. To install OpenCV follow the below steps.

Firstly, we will install some prerequisites for OpenCV.

```
sudo apt-get install build-essential cmake pkg-config
sudo apt-get install libtiff5-dev libjpeg-dev libpng-dev
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
```

Secondly and finally, we will download and build OpenCV for our device. Run the following commands in order to do so.

Download OpenCV using the below command.

```
git clone https://github.com/opencv/opencv.git
```

Navigate into the folder downloaded.

```
cd opencv
```

Create the folder to store the build files and navigate into it.

```
mkdir build && cd build
```

Setup some parameters for building OpenCV.

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D
CMAKE_INSTALL_PREFIX=/usr/local -D WITH_CUDA=OFF -D
WITH_CUFFT=OFF -D WITH_CUBLAS=OFF -D WITH_NVCUVID=OFF -D
WITH_OPENCL=OFF -D WITH_OPENCLAMDFFT=OFF -D
WITH_OPENCLAMDBLAS=OFF -D BUILD_opencv_apps=OFF -D
BUILD_DOCS=OFF -D BUILD_PERF_TESTS=OFF -D BUILD_TESTS=OFF -D
ENABLE_NEON=on
```

Build OpenCV. BE WARNED THAT THIS STEP COULD TAKE OVER COUPLE OF HOURS.

```
make
```

Once done building OpenCV run the following commands to install it.

```
make
sudo make install
sudo ldconfig
```

2. Connecting an Android Device Camera as a Video Device in Linux:
   In this project, we will be using an Android Device's camera to transmit video to our BeagleBone, as finding a high-quality webcam might be challenging. Now that we have all the required software, we can proceed with setting up our camera.

   We will be using an application called "IP Webcam" to stream video from the cameras of our Android Device to our board. This application will simply stream the video from the cameras of our Android Device to other devices in its local network via HTTP.
   Install this application on your device via the following link.

   https://play.google.com/store/apps/details?id=com.pas.webcam&hl=en_CA

   Once the application is downloaded, navigate to the Video Preferences menu and do the following changes.

   Set Video Resolution to 640x480.
   Set Photo Resolution to 640x480.
   Set Quality to some value near 50.
   Set the Main camera to the front camera.

   Once you are done setting up the application, turn off the WiFi and Cellular on your device, connect it to your BeagleBone Black via USB, and turn on USB Tethering on your device. Make sure you have the 5V adapter connected to your board, in order for it to be able to charge your device.

   When you are done with the above steps, simply launch the application and click the start button on the bottom. If you did everything correctly you should see an IP address on the bottom of the screen. Make sure you note this IP address. In our case, the IP address we get is as the following.

   192.168.42.129

   After you are done setting up your Android Device, switch back to your board, ssh into it and run the following commands.

   Below command creates two loopback video devices which will receive and broadcast video feeds.

   sudo modprobe v4l2loopback devices=2 (This command needs to be run after each time Beagle Board is rebooted.)

   Below command streams the video feed from the Android device to video0 device on our board in background. Make sure to change the IP address to the one you noted back in the IP Webcam application.

```
gst-launch-1.0 -vt souphttpsrc location='http://192.168.42.129:8080/videofeed' is-
live=true ! multipartdemux ! decodebin ! videoconvert ! v4l2sink device=/dev/video0 &
```

If you did everything correctly, you should see that there is now a Video Connection on your Android Device on the bottom of the screen in the IP Webcam application. To test we can run the following command on the board to take a picture from the camera.

```
fswebcam -r 640x480 --jpeg 100 -D 1 web-cam-shot.jpg
```

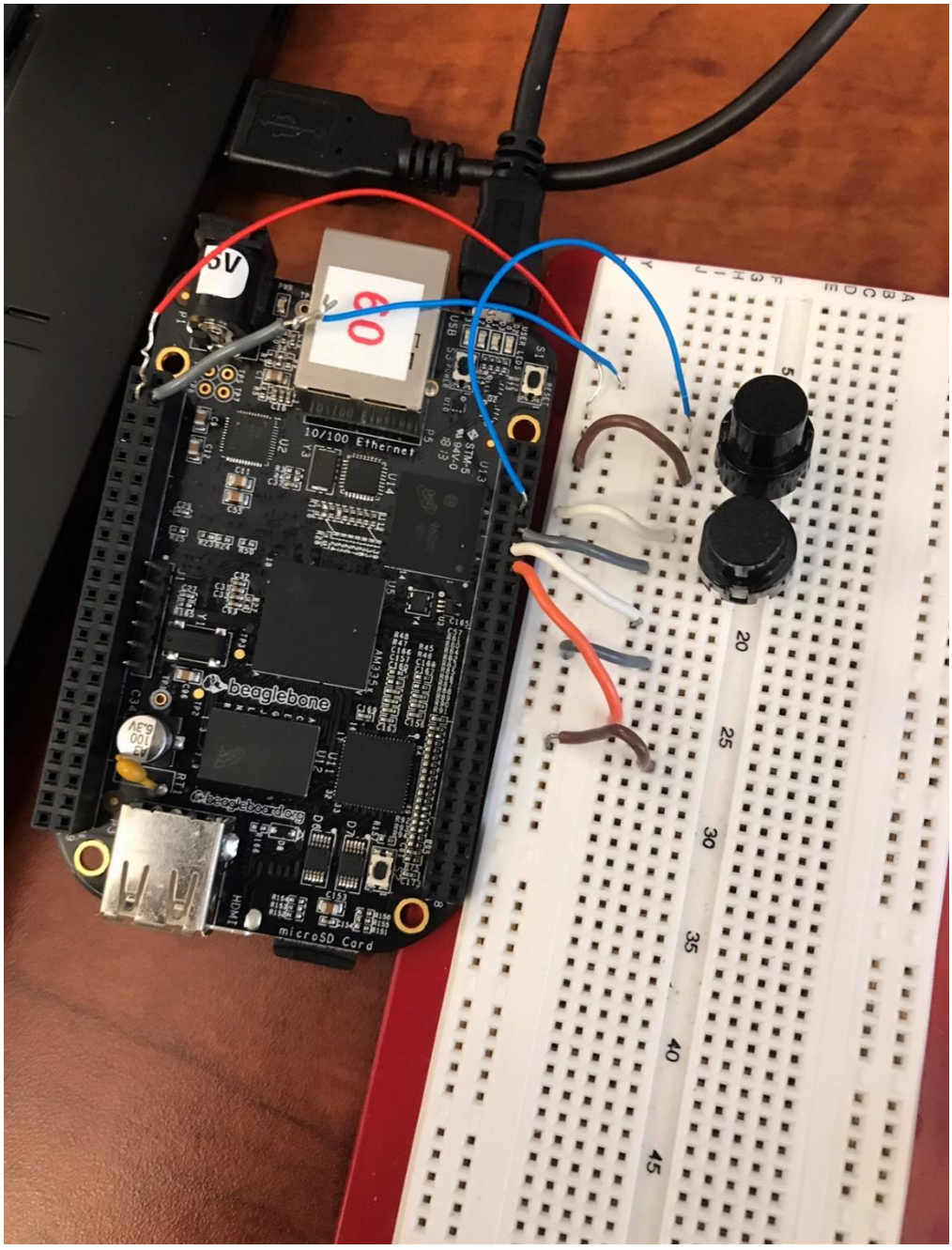
If fswebcam is not installed on your board, you can install it using the below command.

```
sudo apt-get install fswebcam
```

Deploying the Smart Door Bell Application:

Now that we have a camera set up and all software installed, we can proceed with deploying our app.

We will start with setting up the PIN Code Authentication. For this part, we will use couple of buttons, and to attach these to the board we will need some cables and a bread board as we have mentioned earlier. To get started make sure you have the same equipment and do the connections to your board as in the picture below.



Now that we have the PIN Code method working, we will now deploy the Face Recognition method. Make sure that, your Android Device is connected to the board with tethering on and "IP Webcam" application started. We will use the below two commands to start receiving video from the device as we have done earlier. Please note that the first command needs to be run after each reboot and the second command needs to be started and put in the background whenever we want to start the video stream. To get started, enter the following commands to your ssh terminal.

```
sudo modprobe v4l2loopback devices=2
```

```
gst-launch-1.0 -vt souphttpsrc location='http://192.168.42.129:8080/videofeed' is-live=true ! multipartdemux ! decodebin ! videoconvert ! v4l2sink device=/dev/video0 &
```

The first command creates a loopback device which can receive a video feed and distribute it through a linux device, where the second grabs the video stream that is being broadcasted by the Android Device through http and redirects it to the linux device "video0".

Once we have the video working, we will proceed with inspecting the face_recognition folder and its contents. Its content is as follows. Also, make sure you copy this folder to your board.

- dataset
- face_detection_model
- output
- static
- templates
- authorizations.txt
- authorizationweb.py
- camera.py
- extract_embeddings.py
- openface_nn4.small2.v1.t7
- recognize_video.py
- train_model.py
- web.py

Let's talk briefly about what each one of these folders or files are for.

- Dataset:
  This folder will store the pictures of each person. Currently it has only 6 people in it. For each person you would like the door bell to recognize, simply create a folder with that person's name put their pictures in it.

- Face_detection_model:
  This folder contains a pre-trained deep learning model provided by OpenCV to detect faces.
- Output:
  There are 3 files worth explaining in this folder.
  - Embeddings.pickle: Embeddings stored for each face is stored in this file.
  - Le.pickle: Contains the name labels for the persons we added to our recognizer.
  - Recognizer.pickle: This is the machine learning model responsible for recognizing faces.
- Static:
  Folder for static files for the Web UI.
  - ajax.js : A basic script written in javascript and jquery to get the recognized person's name and insert into the Web UI.
  - jquery-3.3.1.min.js: Library file for jquery.
- Templates:
  Folder to store the Web UI design templates.
- Authorizations.txt:
  A file that keeps tracks of the authorized users who can unlock the door automatically.
- Authorizationweb.py:
  A Python script that can add or remove authorized persons from authorizations.txt and it can also check if a person is authorized given a name.
- Camera.py
  A Python file including a VideoCamera class that can get frames from video0 device and output it as a jpeg image file.
- Extract_embeddings.py
  A Python script that extracts face embeddings from the pictures we will be putting in the folder dataset.
- openface_nn4.small2.v1.t7
  A Torch deep learning model which helps produce the face embeddings.
- Recognize_video.py
  A Python script to recognize the person facing the camera. It returns person's name or unknown.
- Train_model.py
  A python script to train our model for the recognizer using the embeddings extracted by extract_embeddings.py.
- Web.py
  A Python script that starts a Flask Web server which starts the Web UI for the Smart Door Bell.

Now that we have looked at the files, let's get started to do recognition.
Before we start the recognizer, we need to place pictures of the people we want in a separate folder for each person and copy those folders into the folder dataset.
In our case, we already have 6 people in the dataset file.

Once we have copied the pictures, we will first run the extract_embeddings.py to extract the face embeddings in the pictures we placed in the dataset folder. Then, we will call train_model.py to train our recognize model with the embeddings we just extracted. These two scripts can be called as follows. This operation will take some time depending on the number of pictures placed in the dataset folder. It should take around 3-4 minutes.

python3 extract_embeddings.py

python3 train_model.py

Note: These python scripts(extract_embeddings.py, train_model.py, recognize_video.py) were created following the guide in the given link in the references [17], however they were modified to optimize for BeagleBone and to make it work with our WebUI.

Now we can, start our WebUI and start our recognizer!

Simply start the Web UI by typing in the following command.

python3 web.py

It will take around 15 seconds to start the Web UI, meanwhile you will be able to see its status in your terminal. Once it is loaded, simply go to your favorite web browser on your computer that is connected to the BeagleBone and navigate to the following address.

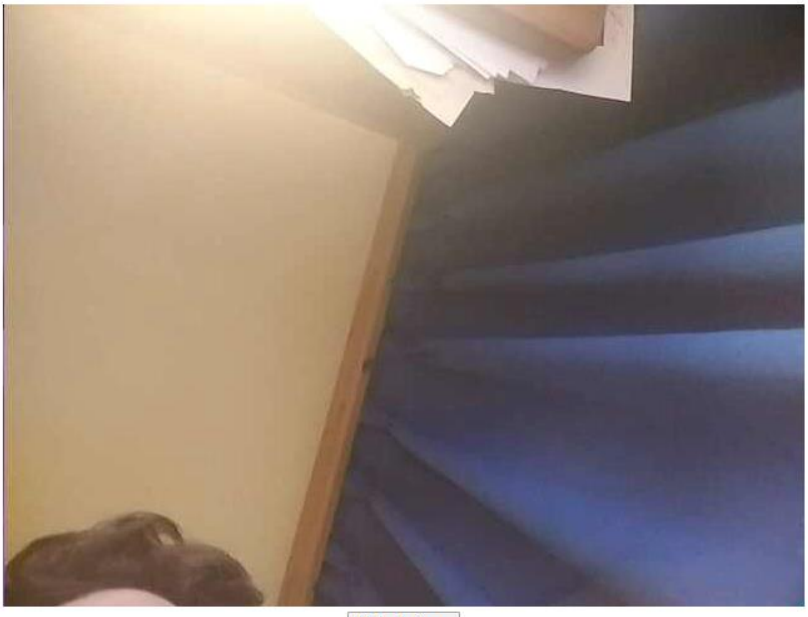
192.168.7.2:5000

You should now be able to see the Web UI! Here you can see a live video feed of the camera, unlock the door manually and see the result if the person facing the camera is recognized. Also, you can authorize/deauthorize who can unlock the door automatically with their face.

Let's try the recognizer now. Point the camera to your face and hold it around 15 seconds, the UI should update and print out the name of the person that is recognized. Further it should also unlock the door if the door you are authorized. When the door is unlocked the UI will also print a message for that. If you haven't authorized your face to unlock the door automatically yet, you can do so by clicking "Click here to authorize a person.". In the authorization page, just find your username and check if you are authorized, if not you can simply type in your username in the form at the bottom of the list and click "Authorize User" to authorize your face. Now whenever the program recognizes your face it will unlock the door automatically and print out a message. If you want to unlock the door manually you can use the "Unlock Door" button. If you want to deauthorize a user's face, you can use the second link for it.

A small demo video with the name demo.mp4 can be found attached with this project documentation.

Smart Doorbell UI



Unlock Door

The person at the door:

Click here to authorize a person.

Smart Doorbell UI

Username	Authorized
s2	False
tanseli	True
s1	False
ege	True
unknown	False
mustafa	True

Please type in the name of the user you would like to authorize and click "Authorize User".

ege Authorize User

References

Below is a list of references used when writing this guide. More information about BeagleBone Black is available on their website and the links below.

[1] http://beagleboard.org/about

[2] https://www.teachmemicro.com/beaglebone-black-hardware/

[3] https://www.tested.com/art/makers/459278-everything-you-need-know-about-beaglebone-black/

[4] http://beagleboard.org/getting-started

[5] https://elinux.org/Beagleboard:BeagleBoneBlack_Debian#Flashing_eMMC

[6] https://groups.google.com/forum/#!topic/beagleboard/_y-fIgg-J7I

[7] https://github.com/RobertCNelson/boot-scripts/blob/master/tools/grow_partition.sh

[8] https://ofitselfso.com/BeagleNotes/HowToConnectBeagleboneBlackToTheInternetViaUSB.php

[9] https://www.vultr.com/docs/setup-x11-forwarding-over-ssh-on-debian-wheezy

[10] http://www.ibiblio.org/pub/historic-linux/distributions/debian-0.91/ChangeLog

[11] https://www.researchgate.net/publication/229014230_Measuring_Lenny_the_size_of_Debian_50

[12]https://www.pcworld.com/article/247845/debian_linux_named_most_popular_distro_for_web_servers.html

[13] https://www.debian.org

[14] https://en.wikipedia.org/wiki/Debian

[15] https://www.ubuntu.com/community/debian

[16] https://wiki.gentoo.org/wiki/Linux-headers

[17] https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/