International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 06 Issue 13
December 2019

Real-Time Monitoring Security System Integrated With Raspberry Pi And E-Mail Communication Link

K. Sudharani[1], S. Ashok Reddy[2], S. Mahaboob Basha[3],
[1]P.G. Scholar, [2]Assistant Professor, [3]Head of the Department
[1,2,3] Branch : Electronics and Communication Engineering(ES)
[1,2,3]Geethanjali College Of Engineering & Technology, Kurnool.
Email :[1]khatri.sudharani@gmail.Com, [2]singasaniashokreddy@gmail.com

ABSTRACT

Now-a-days, crime rates are increasing at an alarming rate due to theft cases. It is equally important to catch the thief to prevent theft. Most of the time, thief escapes due to delay in action taken by concerned authorities at that time. In this paper, we present the design of a security system which solves this problem. The proposed system is for Smart Door lock technique with Raspberry Pi using IoT and is done by integrating webcam and motion sensor with e-mail.

Raspberry Pi operates and controls motion sensor and webcam for sensing and surveillance. For instance, whenever any motion is detected, the webcam streaming starts and Raspberry Pi device alerts the owner through an e-mail on his registered mail-id. Whenever it is a wrong or random PIN entered, webcam captures the intruder's image and sends it over e-mail for real-time action to be taken by the owner to prevent theft.

Keywords: Raspberry Pi device, E-mail Communication Link.

INTRODUCTION

In this new era of technology the race is to provide a user friendly device which is cost effective. The capability of the device should be of such a nature that the cost of the item seems to be always less. Such a device is offered by Raspberry Pi systems. Its capability ranges from being a security system to a VPN server. Unlike other computers it has the capability to accept several program which includes "Python" language. Generally a security system will cost an approx of Rs 6000 to Rs 8000 where as the Raspberry Pi system will at max cost only Rs 4000. so who will be interested in buying a security system of Rs 6000 (approx) where in one can get a system in a price range of Rs 2000 and with added features of email notification. Apart from the price tag the next aspect is the user friendly nature of the system.

As we are all aware that in India not everyone is a tech savvy individual. There by making it more important to ensure that the user does has to rewind the entire system to view the problem. In this system the email notification feature helps the user to see what is wrong than to see the entire video to find the error. The Raspberry Pi system is not only user friendly it also enables a individual wit medium knowledge to assemble the system if the necessary raw material is available and by creation of some extra files to support the operating system to store the data. So they are not only a money saving project but also a efficient security system. With regards to our home, security is significant issue to the population. Presently, the additions of wrong doing associated with the house were numerous on reduced that extent this

concept gets the possibility to be notably attainable that extended the safety level of home. The sagacious game arrange is that create our home as innovative point the safety views. In earlier days, we've got one pet at our home for the safety, nonetheless the circumstance has modified lands up being gift days. individuals begin tolerating on the event to achieves some level of security in family. From these state of affairs, we have a tendency to ar driven to create such framework that will be created for society or organizations give security [3].

PROPOSED SYSTEM:

The proposed system has been designed to overcome the drawbacks of the previous security system and to improve the security, flexibility, efficiency whenever needed, having a security camera system may sometimes be impossible due to the exhaustive costs incurred during installation. The Raspberry Pi is a computer of a size of a credit card that has the capability to become a camera security system when its own camera board is used. It contains all the essential software to include motion detection which enables the Raspberry Pi's camera to detect motion and save the image as well as view a live streaming of the location from the camera. A python script, then directs the Raspberry Pi to send email notifications every time a motion is detected. With these components, a cost effective and efficient security camera system is made as reported here.

LITERATURE SURVEY

An Internet of Things Approach for Motion Detection using Raspberry Pi. The proposed system would detect and take snapshots and videos of the motion when detected and upload to an external server. The major use of the 'Motion Detection' is at homes, buildings and also for surveillance for security for example of server rooms. This system simplifies the use of motion detection and is user-friendly which alerts the user by sending notifications when motion is detected. The motion sensor detects the change of activity in the video by analyzing it with a python script, if there is any difference in the video from the most recent frames, it would alert the user by snapshots and video recordings. The sensor settings can be customized to the user requirements. Real-time data processing and analytics which may arise incomplete datasets in sensor settings should be considered along with clustering of datasets. The project makes use of Raspberry Pi, a low priced credit card sized computer which is capable enough of working as a normal computer and uses a triple-layered architecture where the first layer is the motion detection layer which shows the outcome by the python script. The second layer produces the triggered actions as per the configuration file. And in the third layer, the triggered actions are executed where the user is notified about the activity through emails or pictures saved on the ftp/sftp server.

In the first layer PIR sensor can detect infrared radiation which is emitted by particles. Generally, PIR can detect animal human movement in a requirement range, which is determined by the spec of the specific sensor. The detector itself does not emit any energy but passively receives it, detects infrared radiation from the environment. Once there is infrared radiation from the human body with temperature, focusing on the optical system causes the pyroelectric device to generate a sudden electrical signal and an alarm is issued. The passive infrared alarm does not radiate energy to space but relies on receiving infrared radiation from the human body to make an alarm. Any object with temperature is constantly radiating infrared rays to the outside world. The surface temperature of the human body is 36 – 37 ° C and most of its radiant energy is

concentrated in the wavelength range of 8 – 12 microns. Passive infrared alarms can be classified into infrared detectors (infrared probes) and alarm control sections. The most widely used infrared detector is a pyroelectric detector, which is used as a sensor for converting human infrared radiation into electricity. In the detection area, the infrared radiation energy of the human body through the clothing is received by the lens of the detector and focused on the pyroelectric sensor. When the human body (intruder) moves in this surveillance mode, it enters a certain field of view in sequence and then walks out. It is an enhanced alternative system for expensive security system being used in the present day. The sensor is easy to install and doesn't require any other special modifications for implementing.

The scripting language selected is Python which is easy to use and, with Raspberry Pi, lets you connect project to real world. The syntax is Clean, user friendly and uses simple English keywords.

SMTP works as a three-step process, using a client/server model. First, an e-mail server uses SMTP to send a message from an e-mail client, such as Outlook or Gmail, to an e-mail server. Second, the e-mail server uses SMTP as a relay service to send the e-mail to the receiving e-mail server. Third, the receiving server uses an e-mail client to download incoming mail via IMAP and place it in the inbox of the recipient.

Here in this project included IOT as well. By harnessing the power of IoT for security and surveillance solutions, we enable building owners, organization managers, and security professionals to:

- Manage and control surveillance devices remotely to monitor all aspects of a facility.
- Make smarter decisions about the best course of action to take based on real-time security conditions.
- Determine when there is a false alarm without having to physically inspect the location or unnecessarily dispatch law enforcement.
- Collect and analyze data to make important improvements to security processes and systems.

Perhaps the greatest benefit of using IoT solutions for your security system is the ability to prevent the loss of critical assets. IoT security solutions allow organizations to:

- Gain greater visibility over who enters and leaves a facility in real-time.
- Consistently and securely monitor facility conditions from any location with Wi-Fi access.
- Act quickly on important security alerts delivered right to their mobile device.

The main component of the project is Raspberry Pi, heart of the project.

The main advantage of the project is that the Raspberry Pi used, plays a 1080p resolution videos without lagging and is also used as a server for light traffic such as web traffic or DNS servers or NTP servers. The disadvantages found in the project are that the Raspberry Pi used cannot run X86 operating systems such as Windows and some Linux distributions and also applications, which require high CPU utilization.

INTRODUCTION TO EMBEDDED SYSTEMS

Application Areas

Nearly 99 per cent of the processors manufactured end up in embedded systems. The embedded system market is one of the highest growth areas as these systems are used in very market segment- consumer electronics, office automation, industrial automation, biomedical engineering, wireless communication, data communication, telecommunications, transportation, military and so on.

Consumer appliances At home we use a number of embedded systems which include digital camera, digital diary, DVD player, electronic toys, microwave oven, remote controls for TV and air-conditioner, VCO player, video game consoles, video recorders etc. Today's high-tech car has about 20 embedded systems for transmission control, engine spark control, air-conditioning, navigation etc. Even wristwatches are now becoming embedded systems. The palmtops are powerful embedded systems using which we can carry out many general-purpose tasks such as playing games and word processing.

Office automation: The office automation products using em embedded systems are copying machine, fax machine, key telephone, modem, printer, scanner etc.

Industrial automation: Today a lot of industries use embedded systems for process control. These include pharmaceutical, cement, sugar, oil exploration, nuclear energy, electricity generation and transmission. The embedded systems for industrial use are designed to carry out specific tasks such as monitoring the temperature, pressure, humidity, voltage, current etc., and then take appropriate action based on the monitored levels to control other devices or to send information to a centralized monitoring station. In hazardous industrial environment, where human presence has to be avoided, robots are used, which are programmed to do specific jobs. The robots are now becoming very powerful and carry out many interesting and complicated tasks such as hardware assembly.

Medical electronics: Almost all medical equipment in the hospital is an embedded system. These equipments include diagnostic aids such as ECG, EEG, blood pressure measuring devices, X-ray scanners; equipment used in blood analysis, radiation, endoscopy etc. Developments in medical electronics have paved way for more accurate diagnosis of diseases.

Computer networking: Computer networking products such as bridges, routers, Integrated Services Digital Networks (ISDN), Asynchronous Transfer Mode (ATM), X.25 and frame relay switches are embedded systems which implement the necessary data communication protocols. For example, a router interconnects two networks. The two networks may be running different protocol stacks. The router's function is to obtain the data packets from incoming pores, analyze the packets and send them towards the destination after doing necessary protocol conversion. Most networking equipments, other than the end systems (desktop computers) we use to access the networks, are embedded systems

Telecommunications: In the field of telecommunications, the embedded systems can be categorized as subscriber terminals and network equipment. The subscriber terminals such as key telephones, ISDN phones, terminal adapters, web cameras are embedded systems. The network equipment includes multiplexers, multiple access systems, Packet Assemblers Dissemblers (PADs), satel1ite modems etc. IP phone, IP gateway, IP gatekeeper etc. are the latest embedded systems that provide very low-cost voice communication over the Internet.

Wireless technologies: Advances in mobile communications are paving way for many interesting applications using embedded systems. The mobile phone is one of the marvels of the last decade of the

20th century. It is a very powerful embedded system that provides voice communication while we are on the move. The Personal Digital Assistants and the palmtops can now be used to access multimedia services over the Internet. Mobile communication infrastructure such as base station controllers, mobile switching centers are also powerful embedded systems.

Insemination: Testing and measurement are the fundamental requirements in all scientific and engineering activities. The measuring equipment we use in laboratories to measure parameters such as weight, temperature, pressure, humidity, voltage, current etc. are all embedded systems. Test equipment such as oscilloscope, spectrum analyzer, logic analyzer, protocol analyzer, radio communication test set etc. are embedded systems built around powerful processors. Thank to miniaturization, the test and measuring equipment are now becoming portable facilitating easy testing and measurement in the field by field-personnel.

Security: Security of persons and information has always been a major issue. We need to protect our homes and offices; and also the information we transmit and store. Developing embedded systems for security applications is one of the most lucrative businesses nowadays. Security devices at homes, offices, airports etc. for authentication and verification are embedded systems. Encryption devices are nearly 99 per cent of the processors that are manufactured end up in~ embedded systems. Embedded systems find applications in.every industrial segment-consumer electronics, transportation, avionics, biomedical engineering, manufacturing, process control and industrial automation, data communication, telecommunication, defense, security etc. Used to encrypt the data/voice being transmitted on communication links such as telephone lines. Biometric systems using fingerprint and face recognition are now being extensively used for user authentication in banking applications as well as for access control in high security buildings.

Finance: Financial dealing through cash and cheques are now slowly paving way for transactions using smart cards and ATM (Automatic Teller Machine, also expanded as Any Time Money) machines. Smart card, of the size of a credit card, has a small micro-controller and memory; and it interacts with the smart card reader! ATM machine and acts as an electronic wallet. Smart card technology has the capability of ushering in a cashless society. Well, the list goes on. It is no exaggeration to say that eyes wherever we go, we can see, or at least feel, the work of an embedded system.

HARDWARE IMPLEMENTATION OF THE PROJECT

This chapter briefly explains about the Hardware Implementation of the project. It discusses the design and working of the design with the help of block diagram and circuit diagram and explanation of circuit diagram in detail. It explains the various modules used in this project.

PROJECT DESIGN

The implementation of the project design can be divided in two parts.

- Hardware implementation
- Firmware implementation

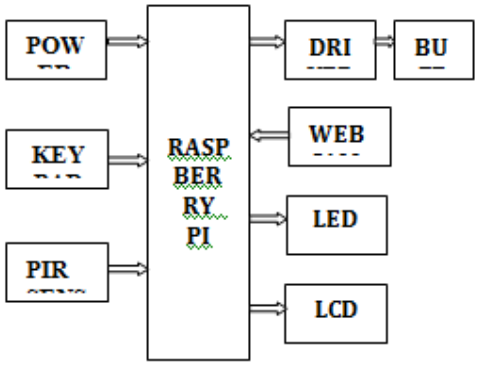
Hardware implementation deals in drawing the schematic on the plane paper according to the application, testing the schematic design over the breadboard using the various IC's to find if the design meets the objective, carrying out the PCB layout of the schematic tested on breadboard,

finally preparing the board and testing the designed hardware.

The project design and principle are explained in this chapter using the block diagram and circuit diagram. The block diagram discusses about the required components of the design and working condition is explained using circuit diagram and system wiring diagram.

BLOCK DIAGRAM OF THE PROPOSED SYSTEM

The block diagram of the design is as shown below. It consists of power supply unit, Raspberry pi and other modules



RASPBERRY PI

The Raspberry Pi is a credit-card sized computer that plugs into your TV and a keyboard. It is a capable little computer which can be used in electronics projects, and for many of the things that your desktop PC does, like spreadsheets, word-processing and games. It also plays high-definition video. We want to see it being used by kids all over the world to learn how computers work, how to manipulate the electronic world around them, and how to program.

The Raspberry Pi is a low cost, **credit-card sized computer** that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games. There are currently four Raspberry Pi models. They are the Model A, the Model B, the Model B+ and the Compute Module. All models use the same CPU, the BCM2835, but other hardware features differ.

RASPBERRY PI 3

Raspberry Pi 3 Model B+

Released in July 2014, the Model B+ is a updated revision of the Model B. It increases the number of USB ports to 4 and the number of pins on the GPIO header to 40. In addition, it has improved power circuitry which allows higher powered USB devices to be attached and now hotplugged. The full size composite video connector has been removed and the functionality moved to the 3.5mm audio/video jack. The full size SD card slot has also been replaced with a much more robust microSD slot.

The following list details some of the improvements over the Model B.

- Current monitors on the USB ports mean the B+ now supports hot plugging.
- Current limiter on the 5V for HDMI means HDMI cable powered VGA converters will now all work
- 14 more GPIO pins
- EEPROM readout support for the new HAT expansion boards
- Higher drive capacity for analog audio out, from a separate regulator, which means a better audio DAC quality.
- No more backpowering problems, due to the USB current limiters which also inhibit back flow, together with the "ideal power diode"
- Composite output moved to 3.5mm jack
- Connectors now moved to two sides of the board rather than the four of the original device.
- Ethernet LED's moved to the ethernet connector
- 4 squarely positioned mounting holes for more rigid attachment to cases etc.

The power circuit changes also means a reduction in power requirements of between 0.5W and 1W.

Product Description

The Raspberry Pi Model B+ incorporates a number of enhancements and new features. Improved power consumption, increased connectivity and greater IO are among the improvements to this Powerful, small and lightweight ARM based computer.

BCM 2835

The Broadcom chip used in the Raspberry Pi Model A, B and B+. The BCM2835 is a cost-optimized, full HD, multimedia applications processor for advanced mobile and embedded applications that require the highest levels of multimedia performance. Designed and optimized for power efficiency, BCM2835 uses Broadcom's VideoCore® IV technology to enable applications in media playback, imaging, camcorder, streaming media, graphics and 3D gaming.

Features:

- Low Power ARM1176JZ-F Applications Processor
- Dual Core VideoCore IV® Multimedia Co-Processor
- 1080p30 Full HD HP H.264 Video Encode/Decode
- Advanced Image Sensor Pipeline (ISP) for up to 20-megapixel cameras operating at up to 220 megapixels per second
- Low power, high performance OpenGL-ES® 1.1/2.0 VideoCore® GPU. 1 Gigapixel per second fill rate.
- High performance display outputs. Simultaneous high resolution LCD and HDMI with HDCP at 1080p60

Overview

BCM2835 contains the following peripherals which may safely be accessed by the ARM:

- Timers
- Interrupt controller
- GPIO
- USB
- PCM / I2S
- DMA controller
- I2C master
- I2C / SPI slave
- SPI0, SPI1, SPI2
- PWM
- UART0, UART1

The purpose of this datasheet is to provide documentation for these peripherals in sufficient detail to allow a developer to port an operating system to BCM2835. There are a number of peripherals which are intended to be controlled by the GPU. These are omitted from this datasheet. Accessing these peripherals from the ARM is not recommended.

General Purpose I/O (GPIO)

General Purpose Input/Output pins on the Raspberry Pi

Overview

This page expands on the technical features of the GPIO pins available on BCM2835 in general. For usage examples, see the GPIO Usage section. When reading this page, reference should be made to the BCM2835 ARM Peripherals Datasheet. GPIO pins can be configured as either general-purpose input, general-purpose output or as one of up to 6 special alternate settings, the functions of which are pin-dependant.

There are 3 GPIO banks on BCM2835.

Each of the 3 banks has its own VDD input pin. On Raspberry Pi, all GPIO banks are supplied from 3.3V. **Connection of a GPIO to a voltage higher than 3.3V will likely destroy the GPIO block within the SoC.**

A selection of pins from Bank 0 is available on the P1 header on Raspberry Pi.

GPIO PADS

The GPIO connections on the BCM2835 package are sometimes referred to in the peripherals datasheet as "pads" - a semiconductor design term meaning "chip connection to outside world". The pads are configurable CMOS push-pull output drivers/input buffers. Register-based control settings are available for

- Internal pull-up / pull-down enable/disable
- Output drive strength
- Input Schmitt-trigger filtering

Power-On States

All GPIOs revert to general-purpose inputs on power-on reset. The default pull states are also applied, which are detailed in the alternate function table in the ARM peripherals datasheet. Most GPIOs have a default pull applied.

Interrupts

Each GPIO pin, when configured as a general-purpose input, can be configured as an interrupt source to the ARM. Several interrupt generation sources are configurable:

- Level-sensitive (high/low)
- Rising/falling edge
- Asynchronous rising/falling edge

Level interrupts maintain the interrupt status until the level has been cleared by system software (e.g. by servicing the attached peripheral generating the interrupt).

The normal rising/falling edge detection has a small amount of synchronisation built into the detection. At the system clock frequency, the pin is sampled with the criteria for generation of an interrupt being a stable transition within a 3-cycle window, i.e. a record of "1 0 0" or "0 1 1". Asynchronous detection bypasses this synchronisation to enable the detection of very narrow events.

UART

The BCM2835 device has two UARTS. On mini UART and PL011 UART. This section describes the PL011 UART. For details of the mini UART see 2.2 Mini UART.

The PL011 UART is a Universal Asynchronous Receiver/Transmitter. This is the ARM UART (PL011) implementation. The UART performs serial-to-parallel conversion on data characters received from an external peripheral device or modem, and parallel-to-serial conversion on data characters received from the Advanced Peripheral Bus (APB).

The ARM PL011 UART has some optional functionality which can be included or left out.

The following functionality is not supported :

- Infrared Data Association (IrDA)
- Serial InfraRed (SIR) protocol Encoder/Decoder (ENDEC)
- Direct Memory Access (DMA).

The UART provides:

• Separate 16x8 transmit and 16x12 receive FIFO memory.

• Programmable baud rate generator.

• Standard asynchronous communication bits (start, stop and parity). These are added

prior to transmission and removed on reception.

- False start bit detection.
- Line break generation and detection.
- Support of the modem control functions CTS and RTS. However DCD, DSR, DTR, and RI are not supported.
- Programmable hardware flow control.
- Fully-programmable serial interface characteristics:
  - data can be 5, 6, 7, or 8 bits
  - even, odd, stick, or no-parity bit generation and detection
  - 1 or 2 stop bit generation
  - baud rate generation, dc up to UARTCLK/16

The UART clock source and associated dividers are controlled by the Clock Manager.

For the in-depth UART overview, please, refer to the ARM PrimeCell UART (PL011)

Troubleshooting UART Problems

The above code works (we've used it for TX and RX). If you can't get to to work for you and you've been through the steps to release the UART from being used for the console try the following:

Permissions

This command will set read and write access permissions for all users on the UART – it shouldn't be needed but can be used just to be sure there is not a permissions problem:

sudochmoda+rw /dev/ttyAMA0

Baud Rate Error

Try using a slower BAUD rate (or a single 0xFF byte which only has the start bit low) and see if it works. We had a problem using 115k2 baud rate where our microcontroller communicating with the RPi could hit 113636baud or 119047baud. 113636baud had the lowest error margin so we used it and TX from the RPi being received by the microcontroller worked fine. 754. Changing the microcontroller to use 119047baud caused RX to work. We then tested the RPi transmitting a byte of 0×00 and measured the low state on a scope we got 78uS, showing an actual baud rate of 115384 from the RPi (8bits + the start bit all low). This was odd as 113636baud still had to lower error margin but that was the finding.

POWER SUPPLY

The device is powered by a 5V micro USB supply. Exactly how much current (mA) the Raspberry Pi requires is dependent on what you connect to it. We have found that purchasing a 1.2A (1200mA) power supply from a reputable retailer will provide you with ample power to run your Raspberry Pi.

Typically, the model B uses between 700-1000mA depending on what peripherals are connected; the model A can use as little as 500mA with no peripherals attached. The maximum power the Raspberry Pi can use is 1 Amp. If you need to connect a USB device that will take the power requirements above 1 Amp, then you must connect it to an externally-powered USB hub. The power requirements of the Raspberry Pi increase as you make use of the various interfaces on the Raspberry Pi. The GPIO pins can draw 50mA safely, distributed across all the pins; an individual GPIO pin can only safely draw 16mA. The HDMI port uses 50mA, the camera module requires 250mA, and keyboards and mice can take as little as 100mA or over 1000mA! Check the power rating of the devices you plan to connect to the Pi and purchase a power supply accordingly.

BACKPOWERING

Backpowering occurs when USB hubs do not provide a diode to stop the hub from powering against the host computer. Other hubs will provide as much power as you want out each port. Please also be aware that some hubs will backfeed the Raspberry Pi. This means that the hubs will power the Raspberry Pi through its USB cable input cable, without the need for a separate micro-

USB power cable, and bypass the voltage protection. If you are using a hub that backfeeds to the Raspberry Pi and the hub experiences a power surge, your Raspberry Pi could potentially be damaged.

USB

Overview

The Raspberry Pi Model B is equipped with two USB2.0 ports. These are connected to the LAN9512 combo hub/Ethernet chip IC3, which is itself a USB device connected to the single upstream USB port on BCM2835. On the Model A, the single USB2.0 port is directly wired to BCM2835. The USB ports enable the attachment of peripherals such as keyboards, mice, webcams that provide the Pi with additional functionality. There are some differences between the USB hardware on the Raspberry Pi and the USB hardware on desktop computers or laptop/tablet devices.

The USB host port inside the Pi is an On-The-Go (OTG) host as the application processor powering the Pi, BCM2835, was originally intended to be used in the mobile market: i.e. as the single USB port on a phone for connection to a PC, or to a single device. In essence, the OTG hardware is simpler than the equivalent hardware on a PC.

OTG in general supports communication to all types of USB device, but to provide an adequate level of functionality for most of the USB devices that one might plug into a Pi, the system software has to do more work.

SUPPORTED DEVICES

In general, every device supported by Linux is possible to use with the Pi, subject to a few caveats detailed further down. Linux has probably the most comprehensive driver database for legacy hardware of any operating system (it can lag behind for modern device support as it requires open-source drivers for Linux to recognise the device by default).

If you have a device and wish to use it with a Pi, then plug it in. Chances are that it'll "just work". If you are running in a graphical interface (such as the LXDE desktop environment in Raspbian), then it's likely that an icon or similar will pop up announcing the new device.

If the device doesn't appear to work, then refer to the Troubleshooting section.

FIRMWARE IMPLEMENTATION OF THE PROJECT DESIGN

This chapter briefly explains about the firmware implementation of the project. The required software tools are discussed below

FIRMWARE IMPLEMENTATION

Raspbian is a competent and versatile operating system that gives your Raspberry Pi all the comforts of a PC: a command line, a browser, and tons of other programs. You can use a Raspberry Pi running Raspbian as a cheap and effective home computer, or you can use it as a springboard and turn your Raspberry Pi into any of countless other functional devices, from wireless access points to retro gaming machines. Here's how to install Raspbian on the Raspberry Pi.

How to install Raspbian on the Raspberry Pi

Installing Raspbian on the Raspberry Pi is pretty straightforward. We'll be downloading Raspbian and writing the disc image to a microSD card, then booting the Raspberry Pi to that microSD card. For this project, you'll need a microSD card (go with at least 8 GB), a computer with a slot for it, and, of course, a Raspberry Pi and basic peripherals (a mouse, keyboard, screen, and power source). This isn't the only method for installing Raspbian (more on that in a moment), but it's a useful technique to learn because it can also be used to install so many other operating systems on the Raspberry Pi. Once you

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 06 Issue 13
December 2019

know how to write a disc image to a microSD card, you open up a lot of options for fun Raspberry Pi projects.

A word about NOOBS

It's worth noting that the method described here isn't your only option for installing Raspbian. You can also opt to use NOOBS, an operating system installation manager that makes it easy to install Raspbian, as well as a few other operating systems.

If you really want to make things easy, you can even buy SD cards that come pre-loaded with NOOBS.

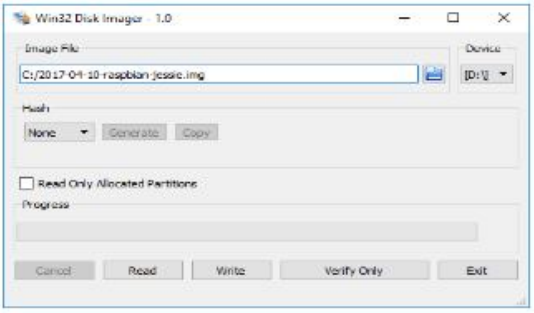
Step 1: Download Raspbian

First things first: hop onto your computer (Mac and PC are both fine) and download the Raspbian disc image. You can find the latest version of Raspbian on the Raspberry Pi Foundation's website here.

Step 2: Unzip the file

The Raspbian disc image is compressed, so you'll need to unzip it. The file uses the ZIP64 format, so depending on how current your built-in utilities are, you need to use certain programs to unzip it. If you have any trouble, try these programs recommended by the Raspberry Pi Foundation:

- Windows users, you'll want 7-Zip.
- Mac users: The Unarchiver is your best bet.
- Linux users will use the appropriately named Unzip.



Step 3: Write the disc image to your microSD card

Next, pop your microSD card into your computer and write the disc image to it. You'll need a specific program to do this:

Windows users, your answer is Win32 Disk Imager. Mac users, you can use the disk utility that's already on your machine.

Linux people, Etcher, which also works on Mac and Windows is what the Raspberry Pi Foundation recommends. The process of actually writing the image will be slightly different across these programs, but it's pretty self-explanatory no matter what you're using. Each of these programs will have you select the destination (make sure you've picked your microSD card!) and the disc image (the unzipped Raspbian file). Choose, double-check, and then hit the button to write.

Step 4: Put the microSD card in your Pi and boot up

Once the disc image has been written to the microSD card, you're ready to go. Put the card into your Rasberry Pi, plug in the peripherals and power source. The current edition to Raspbian will boot directly to the desktop. Your default credentials are username pi and password raspberry.

OPENCV INSTALLATION ON RASPBERRY PI:

Assuming that your OS is up to date, you'll need one of the following for the remainder of this post:

Physical access to your Raspberry Pi 3 so that you can open up a terminal and execute commands

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 06 Issue 13
December 2019

Remote access via SSH or VNC.

I'll be doing the majority of this tutorial via SSH, but as long as you have access to a terminal, you can easily follow along.

Can't SSH? If you see your Pi on your network, but can't ssh to it, you may need to enable SSH. This can easily be done via the Raspberry Pi desktop preferences menu (you'll need an HDMI cable and a keyboard/mouse) or running sudo service ssh start from the command line of your Pi.

After you have changed the setting and rebooted, you can test SSH directly on the Pi with the localhost address. Open a terminal and type ssh pi@127.0.0.1 to see if it is working.

Keyboard layout giving you problems? Change your keyboard layout by going to the Raspberry Pi desktop preferences menu. I use the standard US Keyboard layout, but you'll want to select the one appropriate for your keyboard or desire (any Dvorkac users out there?).

Installing OpenCV 3 on a Raspberry Pi 3 running Raspbian Stretch

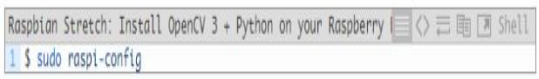
If you've ever installed OpenCV on a Raspberry Pi (or any other platform before), you know that the process can be quite time consuming with many dependencies and pre-requisites that have to be installed. The goal of this tutorial is to thus guide you step-by-step through the compile and installation process.

In order to make the installation process go more smoothly, I've included timings for each step so you know when to take a break, grab a cup of coffee, and checkup on email while the Pi compiles OpenCV.

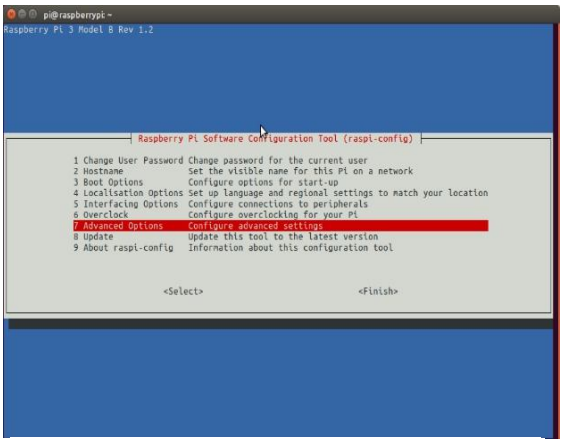
Step #1: Expand filesystem

Are you using a brand new install of Raspbian Stretch?

If so, the first thing you should do is expand your filesystem to include all available space on your micro-SD card:

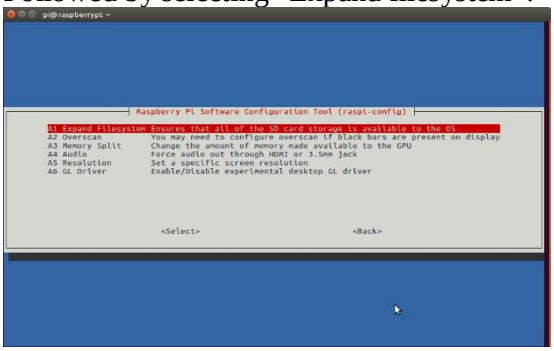


And then select the "Advanced Options" menu item:

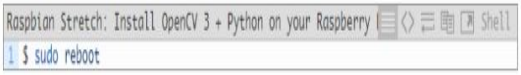


Followed by selecting "Expand filesystem":

Followed by selecting "Expand filesystem":



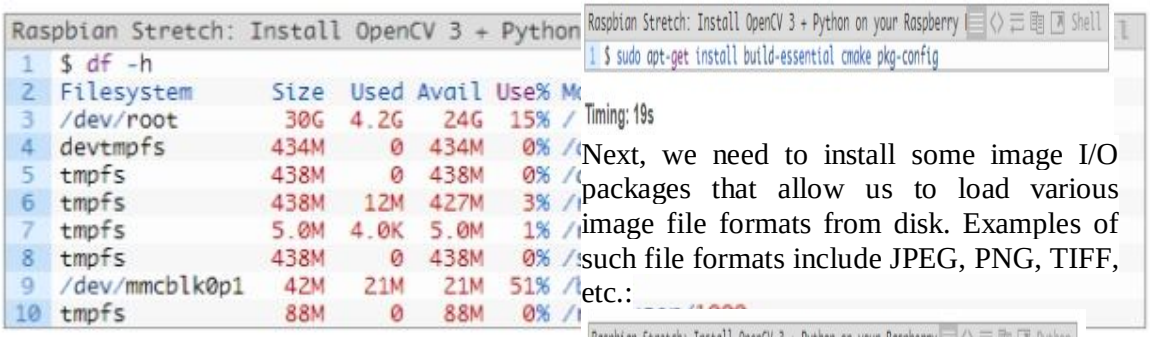
Once prompted, you should select the first option, "A1. Expand File System", hit Enter on your keyboard, arrow down to the "<Finish>" button, and then reboot your Pi — you may be prompted to reboot, but if you aren't you can execute:



After rebooting, your file system should have been expanded to include all available space on your micro-SD card. You can

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 06 Issue 13
December 2019

verify that the disk has been expanded by executing *df -h* and examining the output:



As you can see, my Raspbian filesystem has been expanded to include all 32GB of the micro-SD card.

However, even with my filesystem expanded, I have already used 15% of my 32GB card.

If you are using an 8GB card you may be using close to 50% of the available space, so one simple thing to do is to delete both LibreOffice and Wolfram engine to free up some space on your Pi:

```
$ sudo apt-get purge wolfram-engine
$ sudo apt-get purge libreoffice*
$ sudo apt-get clean
$ sudo apt-get autoremove
```

After removing the Wolfram Engine and LibreOffice, you can reclaim almost 1GB!

Step #2: Install dependencies

I've also included the amount of time it takes to execute each command (some depend on your Internet speed) so you can plan your OpenCV + Raspberry Pi 3 install accordingly (OpenCV itself takes approximately 4 hours to compile — more on this later).

The first step is to update and upgrade any existing packages:

```
$ sudo apt-get update && sudo apt-get upgrade
```

Timing: 2m 14s

We then need to install some developer tools, including CMake, which helps us configure the OpenCV build process:

```
$ sudo apt-get install build-essential cmake pkg-config
```

Timing: 19s

Next, we need to install some image I/O packages that allow us to load various image file formats from disk. Examples of such file formats include JPEG, PNG, TIFF, etc.:

```
$ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
```

Timing: 21s

Just as we need image I/O packages, we also need video I/O packages. These libraries allow us to read various video file formats from disk as well as work directly with video streams:

```
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
$ sudo apt-get install libxvidcore-dev libx264-dev
```

Timing: 32s

The OpenCV library comes with a sub-module named *highgui* which is used to display images to our screen and build basic GUIs. In order to compile the *highgui* module, we need to install the GTK development library:

```
$ sudo apt-get install libgtk2.0-dev libgtk-3-dev
```

Timing: 1m 36s

Many operations inside of OpenCV (namely matrix operations) can be optimized further by installing a few extra dependencies:

```
$ sudo apt-get install libatlas-base-dev gfortran
```

Timing: 23s

These optimization libraries are **especially important** for resource constrained devices such as the Raspberry Pi.

Lastly, let's install both the Python 2.7 and Python 3 header files so we can compile OpenCV with Python bindings:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ sudo apt-get install python2.7-dev python3-dev
```

Timing: 45s

If you're working with a fresh install of the OS, it is possible that these versions of Python are already at the newest version (you'll see a terminal message stating this).

If you skip this step, you may notice an error related to the *Python.h* header file not being found when running *make* to compile OpenCV.

Step #3: Download the OpenCV source code

Now that we have our dependencies installed, let's grab the 3.3.0 archive of OpenCV from the official OpenCV repository. This version includes the *dnn* module which we discussed in a previous post where we did Deep Learning with OpenCV (Note: As future versions of openCV are released, you can replace 3.3.0 with the latest version number):

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ cd ~
2 $ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.3.0.zip
3 $ unzip opencv.zip
```

Timing: 41s

We'll want the **full install** of OpenCV 3 (to have access to features such as SIFT and SURF, for instance), so we also need to grab the opencv_contrib repository as well:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspber
1 $ wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.3.0
2 $ unzip opencv_contrib.zip
```

Timing: 37s

You might need to expand the command above using the "<=>" button during your copy and paste. The *.zip* in the 3.3.0.zip may appear to be cutoff in some browsers. The full URL of the OpenCV 3.3.0 archive is:

https://github.com/Itseez/opencv_contrib/archive/3.3.0.zip

Note: **Make sure your** *opencv* **and** *opencv_contrib* **versions are the same (in this case,**3.3.0**). If the versions numbers do not match up, then you'll likely run into either compile-time or runtime errors.**

1.1.1. Step #4: Python 2.7 or Python 3?

Before we can start compiling OpenCV on our Raspberry Pi 3, we first need to install *pip* , a Python package manager:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ wget https://bootstrap.pypa.io/get-pip.py
2 $ sudo python get-pip.py
3 $ sudo python3 get-pip.py
```

Timing: 33s

You may get a message that pip is already up to date when issuing these commands, but it is best not to skip this step.

If you're a longtime PyImageSearch reader, then you'll know that I'm a **huge fan** of both virtualenv and virtualenvwrapper. Installing these packages is not a requirement and you can **absolutely** get OpenCV installed without them, but that said, I highly recommend you install them as other existing PyImageSearch tutorials (as well as future tutorials) also leverage Python virtual environments. I'll also be assuming that you have both *virtualenv* and*virtualenvwrapper* installed throughout the remainder of this guide.

So, given that, what's the point of using *virtualenv* and *virtualenvwrapper* ?

First, it's important to understand that a virtual environment is a **special tool** used to keep the dependencies required by different projects in separate places by creating **isolated, independent** Python environments for each of them.

In short, it solves the **"Project X depends on version 1.x, but Project Y needs 4.x"** dilemma. It also keeps your global *site-packages* neat, tidy, and free from clutter.

If you would like a full explanation on why Python virtual environments are good

practice, absolutely give this excellent blog post on RealPython a read.

It's standard practice in the Python community to be using virtual environments of some sort, so I **highly recommend** that you do the same:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ sudo pip install virtualenv virtualenvwrapper
2 $ sudo rm -rf ~/.cache/pip
```

Timing: 35s

Now that both *virtualenv* and *virtualenvwrapper* have been installed, we need to update our ~/*.profile* file to include the following lines at the **bottom** of the file:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 # virtualenv and virtualenvwrapper
2 export WORKON_HOME=$HOME/.virtualenvs
3 export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
4 source /usr/local/bin/virtualenvwrapper.sh
```

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.profile
2 $ echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.profile
3 $ echo "export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3" >> ~/.profile
4 $ echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.profile
```

Now that we have our ~/.profile updated, we need to reload it to make sure the changes take affect. You can force a reload of your ~/.profile file by:

Logging out and then logging back in.

Closing a terminal instance and opening up a new one

Or my personal favorite, **just use the** source **command:**

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ source ~/.profile
```

Note: **I recommend running the** *source* ~/*.profile* **file** each time **you open up a new terminal to ensure your system variables have been setup correctly.**

Creating your Python virtual environment

Next, let's create the Python virtual environment that we'll use for computer vision development:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ mkvirtualenv cv -p python2
```

This command will create a new Python virtual environment named *cv* using Python 2.7.

If you instead want to use Python 3, you'll want to use this command instead:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ mkvirtualenv cv -p python3
```

Timing: 24s

How to check if you're in the "cv" virtual environment

If you ever reboot your Raspberry Pi; log out and log back in; or open up a new terminal, you'll need to use the *workon* command to re-access the *cv* virtual environment. In previous blog posts, I've seen readers use the *mkvirtualenv* command — this is entirely unneeded!The *mkvirtualenv* command is meant to be executed only once: to actually **create** the virtual environment.

After that, you can use *workon* and you'll be dropped down into your virtual environment:

Now that we have our ~/.profile updated, we need to reload it to make sure the changes take affect. You can force a reload of your ~/.profile file by:

Logging out and then logging back in.

Closing a terminal instance and opening up a new one

Or my personal favorite, **just use the** source **command:**

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ source ~/.profile
```

Note: **I recommend running the** *source* ~/*.profile* **file** each time **you open up a new terminal to ensure your system variables have been setup correctly.**

Creating your Python virtual environment

Next, let's create the Python virtual environment that we'll use for computer vision development:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ mkvirtualenv cv -p python2
```

This command will create a new Python virtual environment named *cv* using Python 2.7.

If you instead want to use Python 3, you'll want to use this command instead:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ mkvirtualenv cv -p python3
```

Timing: 24s

How to check if you're in the "cv" virtual environment

If you ever reboot your Raspberry Pi; log out and log back in; or open up a new terminal, you'll need to use the *workon* command to re-access the *cv* virtual environment. In previous blog posts,

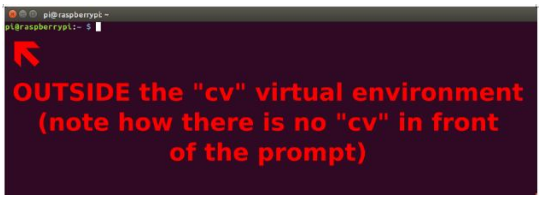
I've seen readers use the mkvirtualenv command — **this is entirely unneeded!**The mkvirtualenv command is meant to be executed only once: to actually *create* the virtual environment.

After that, you can use workon and you'll be dropped down into your virtual environment:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ source ~/.profile
2 $ workon cv
```

To validate and ensure you are in the cv virtual environment, examine your command line — *if you see the text* (cv) *preceding your prompt, then you* **are** *in the* cv *virtual environment:*

Otherwise, if you do not see the (cv) text, then you are not in the cv virtual environment:



To fix this, simply execute the **source** and **workon** commands mentioned above.

Installing NumPy on your Raspberry Pi

Assuming you've made it this far, you should now be in the **cv** virtual environment (which you should stay in for the rest of this tutorial). Our only Python dependency is NumPy, a Python package used for numerical processing:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ pip install numpy
```

Timing: 11m 12s

Step #5: Compile and Install OpenCV

We are now ready to compile and install OpenCV! Double-check that you are in the cvvirtual environment by examining your prompt (you should see the **(**cv**)** text preceding it), and if not, simply execute workon :

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ workon cv
```

Once you have ensured you are in the cv virtual environment, we can setup our build using CMake:

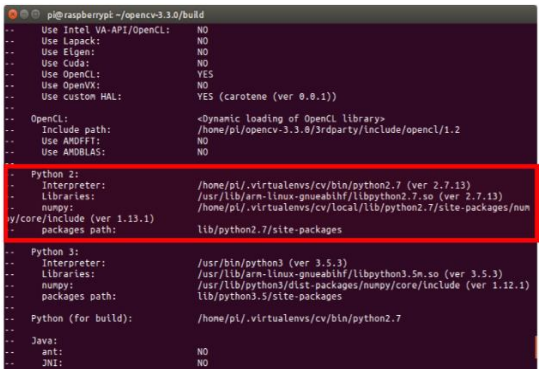
```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ cd ~/opencv-3.3.0/
2 $ mkdir build
3 $ cd build
4 $ cmake -D CMAKE_BUILD_TYPE=RELEASE \
5     -D CMAKE_INSTALL_PREFIX=/usr/local \
6     -D INSTALL_PYTHON_EXAMPLES=ON \
7     -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.3.0/modules \
8     -D BUILD_EXAMPLES=ON ..
```

Timing: 2m 56s

Now, before we move on to the actual compilation step, **make sure you examine the output of CMake!**

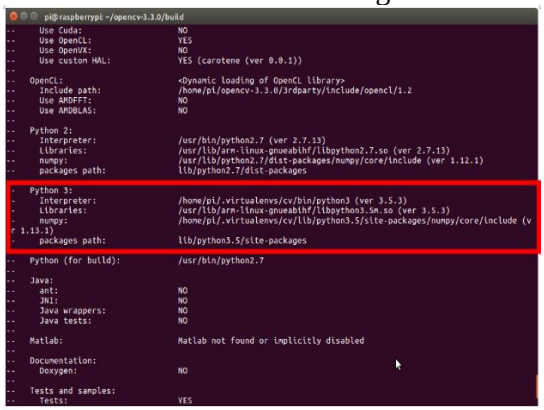
Start by scrolling down the section titled Python 2 and Python 3 .

If you are compiling OpenCV 3 for Python 2.7, then make sure your Python 2 section includes valid paths to the Interpreter , Libraries , numpy and packages path , similar to my screenshot below:



Notice how the Interpreter points to our python2.7 binary located in the cv virtual environment. The numpy variable also points to the NumPy installation in the cvenvironment.

Similarly, **if you're compiling OpenCV for Python 3**, make sure the Python 3 section looks like the figure below:



Again, the Interpreter points to our python3.5 binary located in the cv virtual environment while numpy points to our NumPy install.

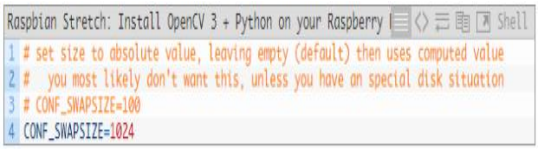
In either case, if you ***do not*** see the cv virtual environment in these variables paths, **it's almost certainly because you are NOT in the** cv **virtual environment prior to running CMake!**

If this is the case, access the cv virtual environment using **workon** cv and re-run the cmakecommand outlined above.

Configure your swap space size before compiling

Before you start the compile process, you should **increase your swap space size**. This enables OpenCV to **compile with all four cores** of the Raspberry PI without the compile hanging due to memory problems.

Open your /etc/dphys-swapfile and then edit the CONF_SWAPSIZE variable:



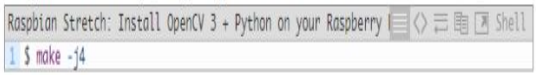
Notice that I've commented out the 100MB line and added a 1024MB line. This is the secret to getting compiling with multiple cores on the Raspbian Stretch.

If you skip this step, OpenCV might not compile.

To activate the new swap space, restart the swap service:

Note: It is possible to burn out the Raspberry Pi microSD card because flash memory has a limited number of writes until the card won't work. It is highly recommended that you change this setting back to the default when you are done compiling and testing the install (see below). To read more about swap sizes corrupting memory, see this page.

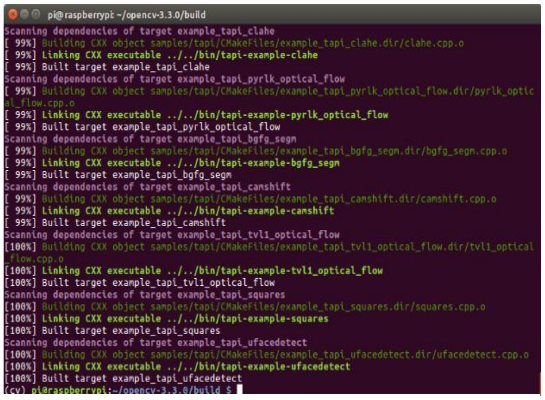
Finally, we are now ready to compile OpenCV:



Timing: 1h 30m

Once OpenCV 3 has finished compiling, your output should look similar to mine

below:



From there, all you need to do is install OpenCV 3 on your Raspberry Pi 3:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ sudo make install
2 $ sudo ldconfig
```

Timing: 52s

Step #6: Finish installing OpenCV on your Pi

We're almost done — just a few more steps to go and you'll be ready to use your Raspberry Pi 3 with OpenCV 3 on Raspbian Stretch.

For Python 2.7:

Provided your Step #5 finished without error, OpenCV should now be installed in/usr/local/lib/python2.7/site-pacakges .

You can verify this using the ls command:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ ls -l /usr/local/lib/python2.7/site-packages/
2 total 1852
3 -rw-r--r-- 1 root staff 1895772 Mar 20 20:00 cv2.so
```

Note: In some cases, OpenCV can be installed in /usr/local/lib/python2.7/dist-packages(note the dist-packages rather than site-packages). If you do not find the cv2.so bindings in site-packages , we be sure to check dist-packages . Our final step is to sym-link the OpenCV bindings into our cv virtual environment for Python 2.7:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ cd ~/.virtualenvs/cv/lib/python2.7/site-packages/
2 $ ln -s /usr/local/lib/python2.7/site-packages/cv2.so cv2.so
```

For Python 3:

After running make install , your OpenCV + Python bindings should be installed in/usr/local/lib/python3.5/site-packages .

Again, you can verify this with the ls command:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ ls -l /usr/local/lib/python3.5/site-packages/
2 total 1852
3 -rw-r--r-- 1 root staff 1895932 Mar 20 21:51 cv2.cpython-34m.so
```

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ cd /usr/local/lib/python3.5/site-packages/
2 $ sudo mv cv2.cpython-35m-arm-linux-gnueabihf.so cv2.so
```

After renaming to cv2.so , we can sym-link our OpenCV bindings into the cv virtual environment for Python 3.5:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ cd ~/.virtualenvs/cv/lib/python3.5/site-packages/
2 $ ln -s /usr/local/lib/python3.5/site-packages/cv2.so cv2.so
```

Step #7: Testing your OpenCV 3 install

Congratulations, you now have OpenCV 3 installed on your Raspberry Pi 3 running Raspbian Stretch!

But before we pop the champagne and get drunk on our victory, let's first verify that your OpenCV installation is working properly.

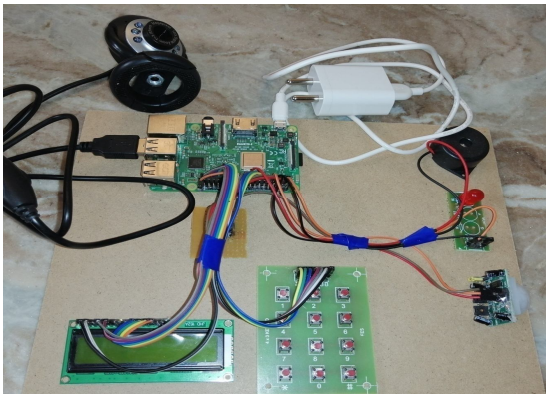
Open up a new terminal, execute the source and workon commands, and then finally attempt to import the Python + OpenCV bindings:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ source ~/.profile
2 $ workon cv
3 $ python
4 >>> import cv2
5 >>> cv2.__version__
6 '3.3.0'
7 >>>
```

As you can see from the screenshot of my own terminal, **OpenCV 3 has been successfully installed on my Raspberry Pi 3 + Python 3.5 environment:**

Working procedure:

Real-Time Monitoring Security System is a security device designed to provide security to homes, offices or industries. Raspberry Pi is used to implement this task.



The main reasons why we have chosen Raspberry pi as specific element are the high processing capacity, relatively low price, and its ability to adapt in different programming modes. The device uses Linux as an operating system, which has access to a large number of libraries and applications compatible with it. Raspberry Pi has an Ethernet port allowing us a network connection, as long as we are in the same subnet with the device we want to access and manage, 4 USB ports used to connect devices like a keyboard, mouse, camera, and other devices that connect through a USB port, and an HDMI port giving us access to the interface of the operating system installed, and can also be used the first time while installing the devices. It has 40 pins that allow us to receive and send signals. They are divided in half into two groups: the 3V, and the 5V group. Therefore, one side of the microcontroller gives a voltage of 3V, and the other 5V. Besides the 40 voltage pins, it has pins that are used to receive signals, which in our case was used to connect the button, that will send the signal for the face identification. Raspberry Pi does not have an operating system previously installed, but that can be downloaded from the Raspberry website, and transferred to an SD card.

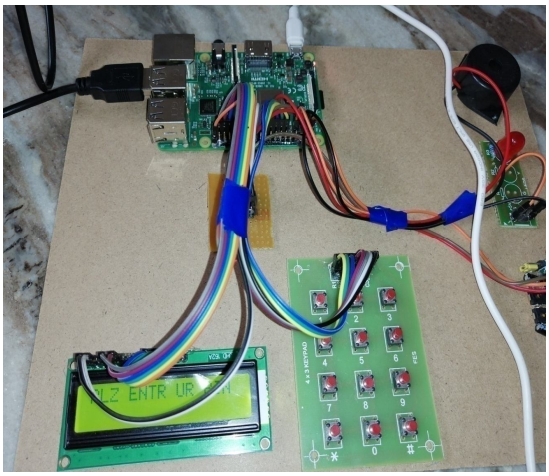
This design consists of PIR sensor for the human detection, keypad for entering password, camera to capture the image, LCD to display the introductory and security messages, LED and buzzer for light and audio indication if the entered person is not an authorized one. The system continuously reads the PIR sensor status. If the human is detected, the PIR sensor will trigger and Raspberry Pi detects this and allows the user to enter the password using the keypad. If the entered password is correct or not, the camera captures the image and this image will be sent to email by Raspberry Pi. If the entered password is wrong, then the buzzer will be alerted immediately and also LED glows.

This design consists of PIR sensor for the human detection, keypad for entering password, camera to capture the image, LCD to display the introductory and security messages, LED and buzzer for light and audio indication if the entered person is not an authorized one.

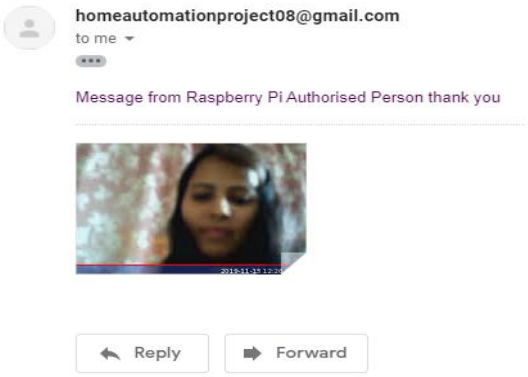
The system continuously reads the PIR sensor status. If the human is detected, the PIR sensor will trigger and Raspberry Pi detects this and allows the user to enter the password using the keypad. If the entered password is correct or not, the camera captures the image and this image will be sent to email by Raspberry Pi. If the entered password is wrong, then the buzzer will be alerted immediately and also LED glows.

SNAPSHOTS

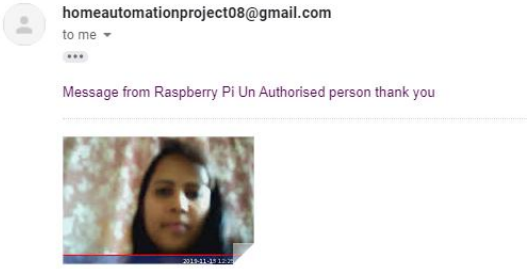
Here in this project we aim to moniter security system and get notified through email. Firstly as soon as the person comes close to the security system the buzzer is alerted, image is captured through webcam and asks for the password to be entered as below



Once the password is entered then password is verified against the saved ones and if the entered password is right then email would be sent to admin with the alert message "Authorised person" and image captured as below



If the person enters the wrong password and system checks if the password entered is correct, since the password didn't match and so the buzzer will be alerted and LED bulb glows indicating that an unauthorized person entered. In addition to it the email is triggered with the person image captured at the security system as shown below.



APPLICATIONS

- Security puposes in office, industries, home etc.
- Security in military and defence.

CONCLUSION AND FUTURE SCOPE

The proposed system of Keypad Lock developed on Raspberry Pi is very much flexible and customizable. This System not only makes any restricted area secure but also aids in catching the intruder. The feature of Live Video Monitoring is an extremely powerful tool to unfold the intruder's motive. It is also power efficient as the webcam turns ON only when the PIR sensor detects motion, thus saving a lot of power than usual CCTVs which continuously consumes power. This also increases lifespan of components. The main advantage of this system is that it is developed on Raspberry Pi which provides much flexibility in terms of embedding more sensors and other hardware components to

the system. Raspberry Pi has enough computational power to support multiple components connected to it. It used work over the Internet but neither it need not to host it anywhere on the net nor using any Cloud reduces the chances of hacking into the system and thus makes it more reliable. Hence, it is highly efficient with least chances of failure and real-time monitoring makes it quite an effective system to be implemented practically.

FUTURE ENHANCEMENTS

Since the proposed system is a very flexible system, it can be implemented in houses, multi-story buildings,farmhouse and industrial offices etc to provide security.The system proposed here is a one-step authentication process i.e. there is already a PIN lock present but it can be modified into multiple level or multi-step process by adding Biometrics or RFID. This can make the system more secure. Also, Face Identification can be used for the image captured by the webcam in order to identify the intruder from a prerecorded

database. Hence, it leaves a lot much potential to improvise this system more.

REFERENCES

1. Sharma, Rupam Kumar, et al. "Android interface based GSM home security system", Issues and Challenges in Intelligent Computing Techniques (ICICT), Proc. IEEE Conf., 2014.
2. D. Luca, Gabriele, et al. "The use of NFC and Android technologies to enable a KNX-based smart home", Software, Telecommunications and Computer Networks (SoftCOM), Proc. 21st IEEE Conf., 2013.
3. Gu, Yi, et al. "Design and Implementation of UPnP-Based Surveillance Camera System for Home Security", Information Science and Applications (ICISA), Proc. IEEE Conf., 2013.
4. Van Thanh Trung, Bui, and Nguyen Van Cuong. "Monitoring and controlling devices system by GPRS on FPGA platform", Advanced Technologies for Communications (ATC), Proc. IEEE Conf., 2013.
5. Karia, Deepak, et al. "Performance analysis of ZigBee based Load Control and power monitoring system", Advances in Computing, Communications and Informatics (ICACCI), Proc. IEEE Conf., 2013.
6. Ryu, Yeonghyeon, Jeakyu Yoo, and Youngroc Kim. "Cloud services based Mobile monitoring for Photovoltaic Systems", Cloud Computing Technology and Science (CloudCom), Proc. IEEE Conf., 2012.
7. Robson, Clyde, et al. "High performance web applications for secure system monitoring and control", Proc. IEEE Conf., Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2012.
8. Han, Jinsoo, et al. "User-friendly home automation based on 3D virtual world", IEEE Trans. Consumer Electronics, 56(3), 2010, 1843- 1847.
9. Xu, Lingshan, et al. "A Cloud-based monitoring framework for Smart Home."Cloud Computing Technology and Science (CloudCom), Proc. 4th IEEE Conf., 2012.
10. Bajorek, Marcin, and Jedrzej Nowak. "The role of a mobile device in a home monitoring healthcare system." Computer Science and Information Systems (FedCSIS), Proc. Federated IEEE Conf., 2011.