AUTOMATIC BEAM CONTROLLER AND DASH CAM

A project report submitted in partial fulfillment of the requirements for the award of the Degree

of

BACHELOR OF TECHNOLOGY

In

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted By

A.K.SRAVANTH (121710408003)

K.YOGEESWAR (121710408023)

M.SHALINI (121710408028)

S.S.BHARATH (121710408041)

Under the esteemed guidance of

Mr. V. RAJ KUMAR

Assistant Professor

DEPARTMENT OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING
GITAM INSTITUTE OF TECHNOLOGY
GITAM

(Deemed to be University)
(Estd. u/s 3 of UGC act 1956 & Accredited by NAAC with "A+" Grade)
VISAKHAPATNAM-530045

GITAM INSTITUTE OF TECHNOLOGY

GITAM (Deemed to be University)

CERTIFICATE

This is to certify that the project work entitled "**AUTOMATIC BEAM CONTROLLER AND DASH CAM**" is a bonafide work carried out by **A.K.SRAVANTH (121710408003), K.YOGEESWAR (121710408023), M.SHALINI (121710408028), S.S.BHARATH (121710408041)** submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in **Electronics and Communication Engineering**, GITAM Institute of Technology, GITAM (Deemed to be University), Visakhapatnam during the academic year 2020-2021.

PROJECT GUIDE
Mr. V. RAJ KUMAR
Assistant Professor
Dept. of EECE
GITAM Institute of Technology

HEAD OF THE DEPARTMENT
Dr. J.B. SEVENTLINE
Professor
Dept. of EECE
GITAM Institute of Technology

DEPARTMENT OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

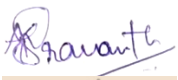
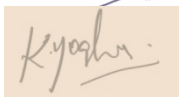
GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

DECLARATION

We hereby declare that the project work entitled "**AUTOMATIC BEAM CONTROLLER AND DASH CAM**" is an original work done in the Department of Electrical, Electronics and Communication Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of **B.Tech. In Electronics and Communication Engineering**. The work has not been submitted to any other college or university for the award of any degree or diploma.

Registration No.	Name	Signature
121710408003	**A.K.SRAVANTH**	
121710408023	**K.YOGEESWAR**	
121710408028	**M.SHALINI**	
121710408041	**S.S.BHARATH**	

ACKNOWLEDGEMENT

We would like to thank our project guide **Mr. V. Raj Kumar,** Assistant Professor Dept. of EECE for his stimulating guidance and profuse assistance. We shall always cherish our association with him/her for his/her guidance, encouragement and valuable suggestions throughout the progress of this work. We consider it a great privilege to work under his guidance and constant support.

We would like to thank our Project Coordinators **Dr. B. Udaya Kumar,** Associate Professor, Dept. of EECE and **Dr. L. L. Rajeswara Rao,** Assistant Professor Dept. of EECE in helping to complete the project by taking frequent reviews and for their valuable suggestions throughout the progress of this work.

We consider it as a privilege to express our deepest gratitude to **Dr. J. B. Seventline**, Head of the Department, Electrical, Electronics and Communication Engineering for her valuable suggestions and constant motivation that greatly helped us to successfully complete the project work.

Our sincere thanks to **Prof. C. Dharma Raj,** Principal, GITAM Institute of Technology, GITAM Deemed to be University for inspiring us to learn new technologies and tools.

Finally, we deem it a great pleasure to thank one and all who helped us directly and indirectly throughout this project.

A.K.SRAVANTH (121710408003)
K.YOGEESWAR (121710408023)
M.SHALINI (121710408028)
S.S.BHARATH (121710408041)

ABSTRACT

Driving in nights is bit difficult; driving in single lane roads during nights is very difficult due to the dazzling light problems and the frequent dipping of headlights by manual means that often causes irritation to the driver particularly at the time of peak traffic. So naturally to get rid of this problem, an automatic mechanism has to come up to dip the headlamp automatically whenever required. Simply, an automatic high beam controller is a unit, which can automatically judge when the headlight beam needs to be lowered, and which dip the headlamp from which beam to a dipped beam. Our work proposes an effective automatic control of the vehicle headlamps based on the detection of head lights and tail lights under night time road conditions.

Dashboard cameras (dash cam) become more and more popular due to decreased prices but with higher video qualities. With dash cam, many accidents cases are recorded and distributed as well as webcasted and highlighted to others to possibly learn the causes of accidents. Using dash cam for traffic and accident monitoring system, this work identifies some that were recorded accidents under different situations. From many explored video footages, it finds that dashboard camera is very effective traffic monitoring system. Learning from many video footages accidents, it is hoped that road drivers/users seeing these accident footages will get more awareness in increasing awareness/carefulness on the roadway.

CONTENTS

Chapter No		Description	Page No
Chapter 1		**Introduction**	**7**
Chapter 2		**Raspberry Pi Zero**	**8**
	2.1	History of Raspberry pi zero	8
	2.2	Introduction to Raspberry pi zero	9
	2.3	Setting up SD card	13
	2.4	Making an SD card- using Windows	14
	2.5	Making an SD card- using Mac	18
Chapter 3		**GPS Module**	**35**
	3.1	Functional Description	35
	3.2	Product Features	35
	3.3	Gps Performance	36
	3.4	Block Diagram	38
	3.5	Working and Information	39-56
Chapter 4		**Installing OpenCV in Raspberry pi zero**	**57**
Chapter 5		**Remotely Accessing Raspberry pi zero**	**66**
Chapter 6		**Making A Dash cam Using Raspberry Pi Zero**	**74**
	6.1	Making a Dash cam	74
	6.2	Interfacing GPS module	78
Chapter 7		**License Plate Detection Using Haarcascade**	**84**
	7.1	Introduction	84
	7.2	Method	85
Chapter 8		**Code and Conclusion**	**86**
	8.1	Code for Dash cam	86
	8.2	Code for Automatic beam controller	87
	8.3	Conclusion	88
	8.4	References	88

CHAPTER 1

INTRODUCTION

Modern automotive vehicles include a variety of different lamps to provide illumination under different operating conditions. Headlamps are typically controlled to alternately generate low beams and high beams. Low beams provide less illumination and are used at night to illuminate the forward path when other vehicles are present. High beams provide significantly more light and are used to illuminate the vehicle's forward path when other vehicles are not present. Daylight running lights have also begun to experience widespread acceptance. Vehicle dashboard cameras (i.e., dash cams) enable high-quality continuous recording of external views that provide evidence in case of unexpected traffic-related accidents and incidents.

This method comprises the following steps: the input images are obtained from the vision system using a Pi Camera which is mounted behind the windscreen inside the camera-assisted vehicle. In order to choose the optimal lens for this application, we did a geometrical study using a pin-hole model. We performed some experimental tests using Pi camera for obtaining the best option in an experimental way. The results of these experiments concluded that for the lens of 6 mm bright objects appear in the image very close to the horizon line and the variation in vertical pixels from the frame where the vehicle appears in the image to the frame where it leaves this, is smaller than using the 4.3 mm lens. Then, the number of frames where the vehicle is in the field of view of the camera is lower for the 6 mm than for the 4.3 mm focal distance. On the other hand, for uneven roads vehicles can be seen with a 4.3 mm lens before than using a 6 mm one due to its higher field of view. Finally, when a vehicle is overtaking the assisted car, with a 4.3 mm lens the vehicle is detected before than with a 6 mm one, minimizing the glare of the driver. As conclusion, the optical lens was set to 4.3 mm, because its field of view is closer to the human one and with this lens, vehicle detection performance is higher. In addition, with our camera settings we can use the same camera configuration for other driver assistance applications

Dash cams are relatively new recording devices, and though there is limited research directly related to dash cam video sharing context, increased research is being conducted related to vehicle-based sensor data and dash cam technology, such as anticipating accidents or moving object detection algorithms in dash cam videos. Along with recording, GPS location is also recorded in order to get location updated.

CHAPTER 2

RASPBERRY-PI ZERO

The Raspberry Pi Zero W extends the Pi Zero family and comes with added wireless LAN and Bluetooth connectivity.

2.1 HISTORY OF RASPBERRY PI ZERO:

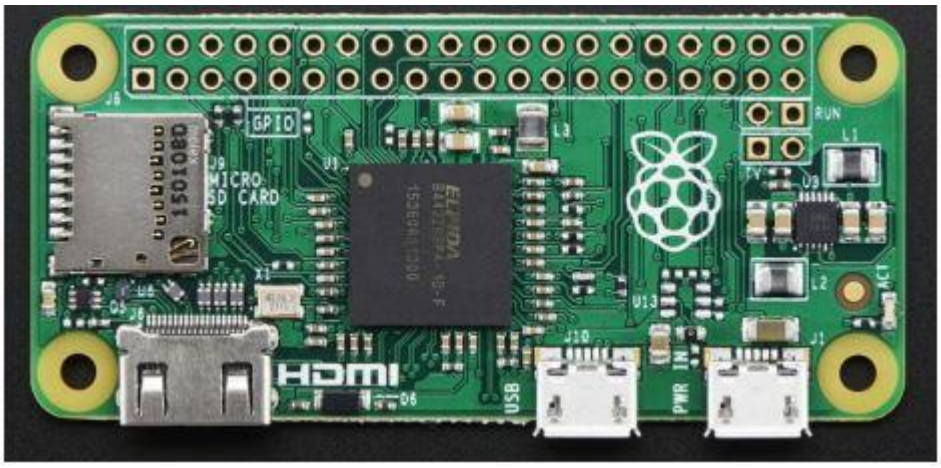
Raspberry Pi- A name which is now not a new technical buzzword for researchers in the area of Computer Science, Electronics & Embedded Systems engineering worldwide. Till date, 5 million pieces of various generations of Raspberry Pi ranging from Model A to Model B and even the latest Raspberry Pi Zero (Going Out of Stock in First week of Launch) have been sold and thousands of projects ranging from Robotics, Super Computer development, Gaming Consoles, Portable Tablets, Server based Implementations on lines of Linux and Cloud Computing, Drones and even Astro Pi has been proposed and implemented all over the world. The foundation seed for development for Raspberry Pi journey started in 2006, when researchers named: Eben Upton, Rob Mullins, Jack Lang and Alan Mycroft at University of Cambridge's Computer Laboratory became stunned to see the decline in the skill level of A Level students and students applying for computer science. The main idea behind their stepping stone development was to give kids tiny and affordable computer in the period where computers were expensive and programming practice among kids was not supported by parents of children in U.K. The team lead by Eben Upton developed several versions of working prototypes from year 2006 to 2008 and the final released version was named as "RASPBERRY Pi". In 2008, the processors used in mobile devices were becoming cheaper and powerful and have full potential to support and run multimedia and all sorts of programming. The project seemed to be of high potential to the team, so the members Eben Upton, Rob, Jack and Alan got the project joint-ventured by Pete Lomas, MD of Norcott Technologies (Hardware Design and Manufacture Company) and David Braben (Co-Author of Seminal BBC Micro Game Elite) to make Raspberry Pi Foundation. The Raspberry Pi was officially created in year 2012 (February 2012) and the developing stone was laid by Raspberry Pi Foundation and within 3 years, the model B entered mass level production with Element 14 and RS Electronics and within 2 years of official launch of Pi, 2 million pieces are sold till date.

Raspberry Pi, a complete PC in itself started a new movement of Portable and Low powerful computers. And taking Raspberry Pi into consideration various replica boards like Intel Galileo, Dwengo, Beaglebone, ORCID etc. have come up with their boards providing somewhat same or little bit more configuration as compared to Pi.

2.2 INTRODUCTION TO RASPBERRY PI ZERO

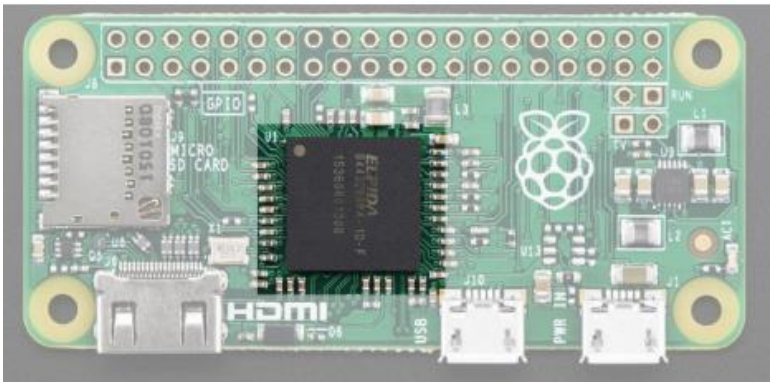
The Raspberry Pi- a credit card sized single board computer developed by Raspberry Pi Foundation, United Kingdom. The board is a miniature marvel, packs extreme computing power and capable to develop amazing projects. The computer costs ranging from $5 to $35 and is perfect to perform all sort of computing tasks and interfacing various sorts of devices via GPIO. The Raspberry Pi board contains Broadcom based ARM Processor, Graphics Chip, RAM, GPIO and other connectors for external devices. The operating procedure of Raspberry Pi is very similar as compared to PC and requires additional hardware like Keyboard, Mouse, Display Unit, Power Supply, SD Card with OS Installed (Acting like Hard Disk) for operation. Raspberry Pi also facilitates USB ports, Ethernet for Internet/Network-Peer to Peer Connectivity. Like any other computer, where Operating system acts as backbone for operation. Raspberry Pi, facilitates open source operating system's based on Linux. Till date more than 30 operating systems based on different flavors of Linux is being launched. Raspberry Pi foundation has also launched various accessories like Camera, Gertboard and Compute Model Kit for deploying add-on hardware modules.

A TOUR OF THE PI ZERO



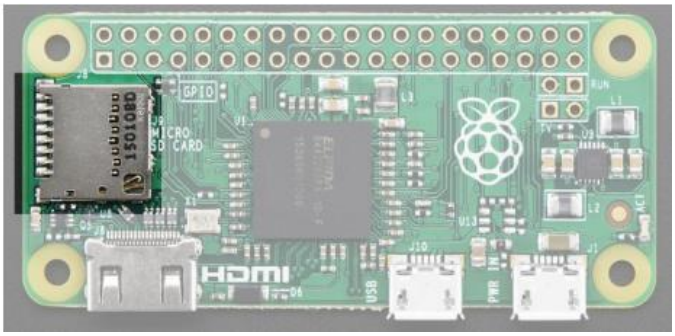
PROCESSOR AND SPEED

To keep the Pi Zero low cost, the processor and RAM are kept pretty basic. Instead of the Pi 2's zippy quad core ARM v7, we're back to a single-core 1GHz ARM (same processor in the Pi Model B+ and A+). We also have 512 MB of RAM with a 'package-on-package' setup.



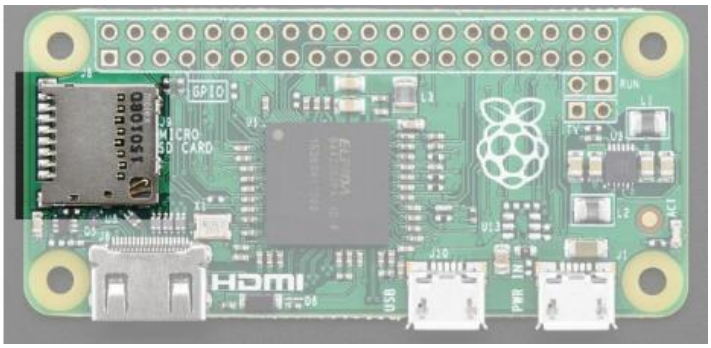
MICRO SD CARD HOLDER

Not much has changed here, we're still going with MicroSD for size and ease of use (they're the most common card size these days!) This time the card holder is up top and is push-pull style not push-push. Honestly, I prefer it this way since you wont accidentally 'push-pop' the card out



VIDEO OUT

HDMI Video is still available, you'll want to use a Mini to Standard HDMI adapter (http://adafru.it/2819) to connect an HDMI cable. There's no 3.5mm jack with composite out, however you can get PAL or NTSC out via two 0.1" pads. We've got a bigger write-up here about Pi Zero video outputs.(https://adafru.it/jEf)

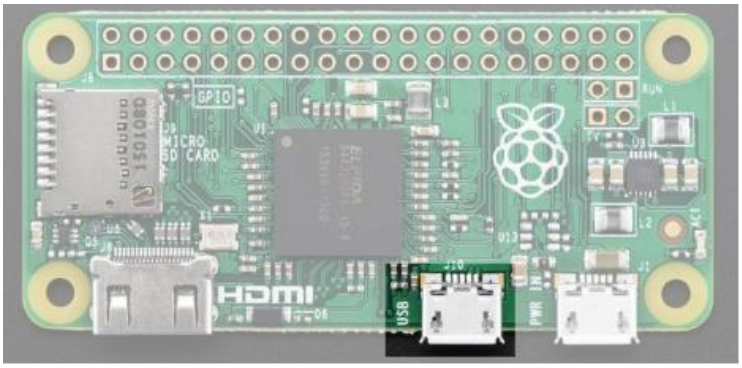


AUDIO OUT

No analog audio out, but if you connect HDMI to a monitor with speakers you will get HDMI digital audio. It's also possible to hack analog audio out with a few passive components, see our more detailed look at Pi Zero audio output options. (https://adafru.it/jEh)1

USB PORT

Like the Pi Model A+, the Pi Zero does not have a USB Hub built in which means you get one USB port! Moreover that USB port is not a standard type A port, instead it is a 'USB On-The-Go' port

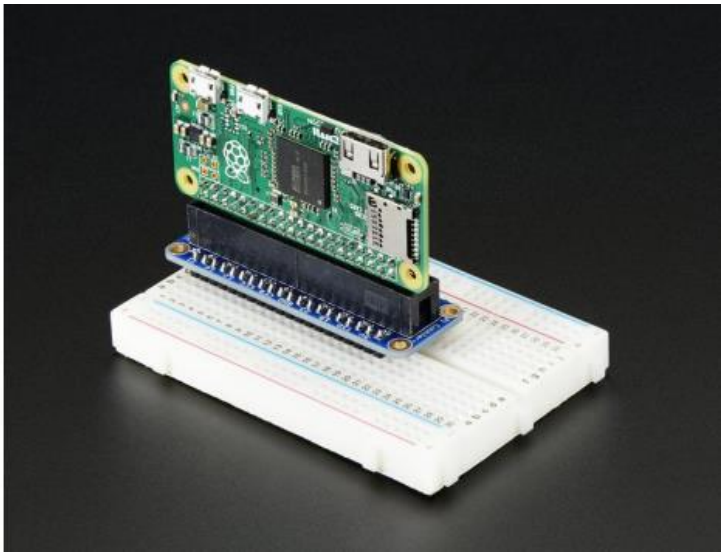


In order to connect a USB device (mouse, keyboard, WiFi) etc you'll need a USB OTG micro B to A cable (http://adafru.it/1099):

If you need to connect multiple USB devices, a simple USB hub will do what you need. A powered hub is even better (http://adafru.it/961), and will let you power high-current USB devices like WiFi adapters and even external USB hard-drives.

GPIO HEADER

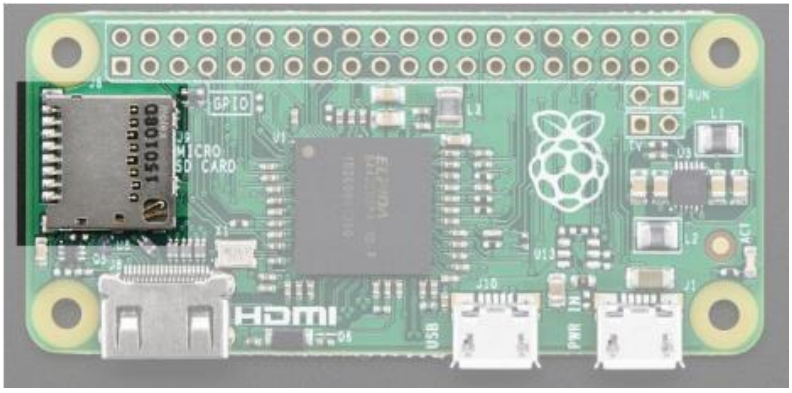
To keep the Zero as simple and small as possible, the 'normal' GPIO header spot has been left blank! Normally, a 2x20 male header is soldered in there (http://adafru.it/2822). While you could grab one of those and solder them in, the empty spot has a lot of potential. For example, you can solder in right-angle socket header, and turn the Pi Zero it a sort of 'daughter card'



2.3 SETTING UP YOUR SD CARD

Before you can power up your Pi Zero, you will need to program in the SD card with an Operating System Much like your computer has Windows, Mac OS X or Linux on it to make it run, the Raspberry Pi needs something to help it boot and run software. That software is Raspbian Linux (a flavor of Debian Linux)

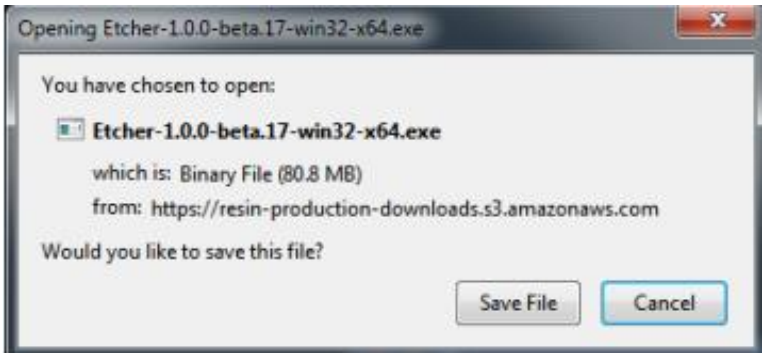
Once you're done, plug the micro SD card into the slot indicated. It will fit snugly in place but you won't hear or feel a 'click'



2.4 MAKING AN SD CARD – USING WINDOWS

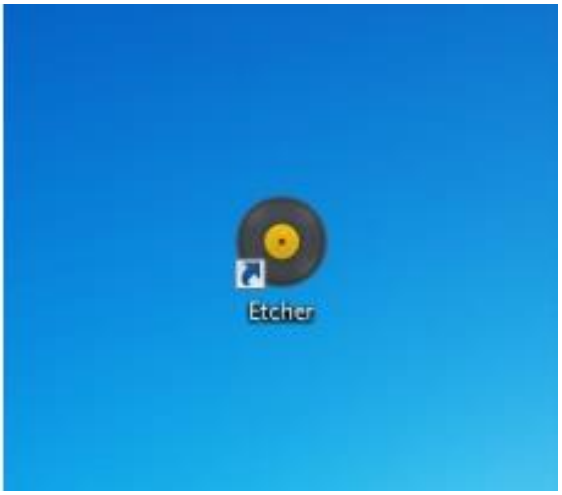
We really like using balenaEtcher for burning SD cards. Works great on any version of Windows, macOS and Linux. It will not over-write your backup disk drive, and can handle compressed images so you do not need to unzip them!

STEP 1. Download Etcher from: https://www.balena.io/etcher/ (https://adafru.it/EMc)

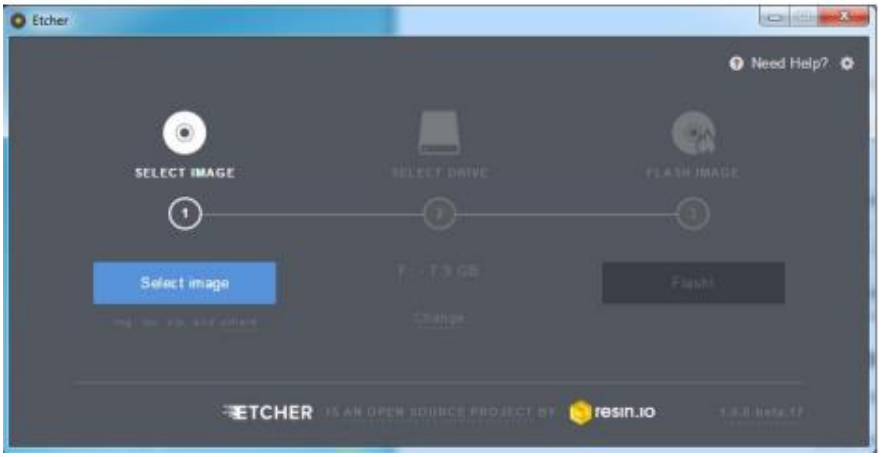


STEP 2. Run the downloaded app to install!

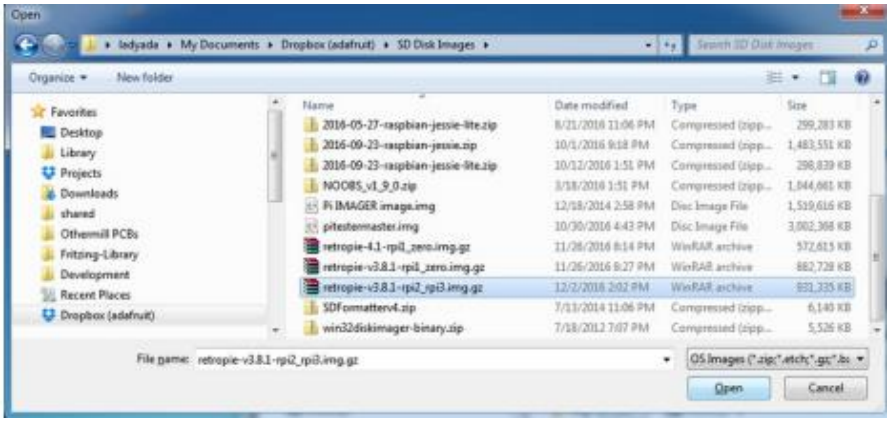
STEP 3. Eject any external storage devices such as USB flash drives and backup hard disks. This makes it easier to identify the SD card. Then insert the SD card into the slot on your computer or into the reader.



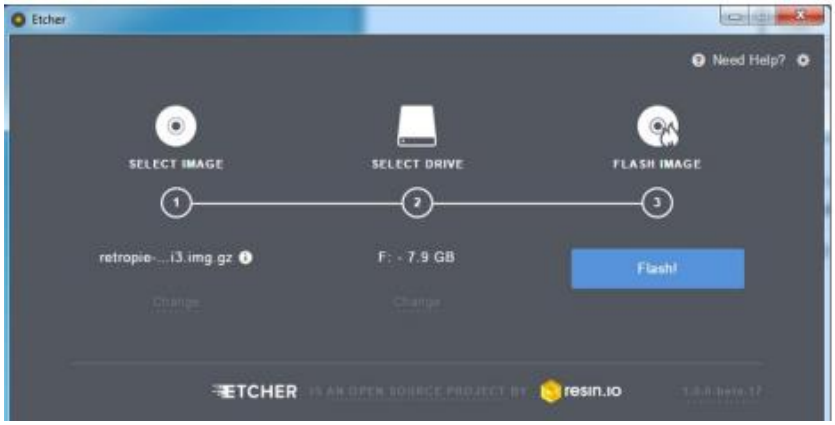
STEP 4. Run the Etcher program



STEP 5. Select the image file by clicking Select Image you can select a compressed file such as a .zip or .gz



STEP 6. Etcher will automatically try to detect the SD drive, check the size to make sure its the right one Then click Flash!



Check that you have the right device, as it will be reformatted, and then click Install. It will take a few minutes to install, but once the SD card is ready, you will see the following.

That's all there is to it. Your SD card is ready for use in your Raspberry Pi.

Faster writes

If you burn a lot of cards, speed it up by turning off Validate write on success

2.5 MAKING AN SD CARD – USING A MAC

We really like using balena Etcher for burning SD cards. Works great on Mac OS X 10.9 or later, won't over-write your backup disk drive, and can handle compressed images so you do not need to unzip them!

Mac OS Catalina Issues If you are having issues running Etcher on the Catalina release of Mac OS, see the links below for more information and some suggested workarounds.

Issue 2833 (https://adafru.it/GB3)

Issue 2911 (https://adafru.it/GB5)

Balena forum post (https://adafru.it/GB7)

Most success has been reported by simply running Etcher from the command line using

sudo: sudo /Applications/balenaEtcher.app/Contents/MacOS/balenaEtcher

STEP 1. Download Etcher from https://www.balena.io/etcher/ (https://adafru.it/EMc)

STEP 2. Open the downloaded disk image and drag the balenaEtcher application to the Applications folder. You can then eject the disk image.

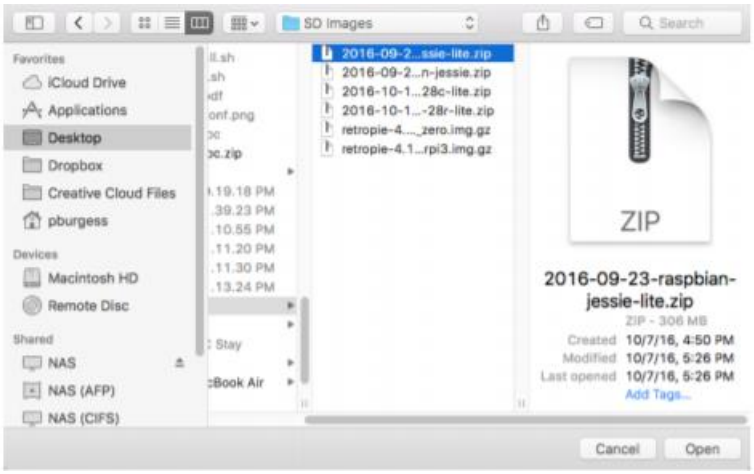
STEP 3. Eject any external storage devices such as USB flash drives and backup hard disks. This makes it easier to identify the SD card. Then insert the SD card into the slot on your computer or into the reader.

STEP 4. Run the Etcher application.

The first time you run Etcher you'll be asked to confirm the download. Click "Open" to continue.

This will launch the Etcher application…

STEP 5. Select the SD card image file by clicking Select Image. You can choose a compressed SD image file such as a .zip or .gz or an uncompressed .img, it's all good!



STEP 6. Etcher will automatically try to detect the SD drive. If you don't have an SD card currently inserted, you'll be prompted to connect one.

Check the disk size to make sure its the right one, that it's not overwriting your main drive or anything nasty. Then click Flash! A-ah!

Etcher will work for a few minutes to "burn" the SD image to the card. You'll see a progress bar as it works. This is about the time you'll wish you'd splurged on a high-speed card.

Once the SD card is ready, you will see the following:

The card will be unmounted automatically, so you can pull it out now and use it in your Raspberry Pi.

2.6 FASTER WRITES :

If you find yourself burning a lot of SD cards, you can speed things up by clicking the gear icon at the top-right, then turn off the “Validate write” option. I’ve written hundreds of cards and only had one fail validation.

VIDEO OUTPUTS :

The Raspberry Pi chipset was originally designed to be a HDMI/graphics co-processor for mobile devices. For that reason, it has quite a bit of 'HDMI horsepower' and can, despite it's small size, play 1080p video at full screen.

HDMI VIDEO OUT :

The easiest & fastest way to get video going is to connect up an HDMI display (https://adafru.it/jsb). We have a ton of options, and any HDMI display size from 640x480 up to 1920x1080 will work. The Mini HDMI port is conveniently labeled and shown below:

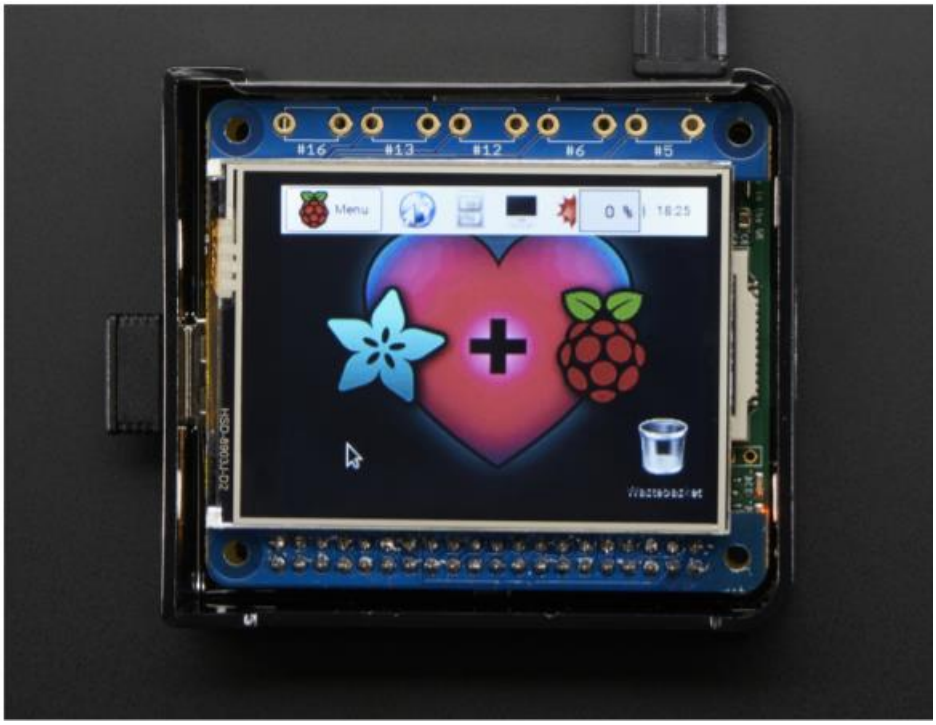
For example, our 5" HDMI touch backpack which is the smallest all-in-one display we carry can be powered from the Pi Zero's USB port and provide a touchscreen at the same time (http://adafru.it/2260)

(Shown here with a Pi 2 because, well, the Pi Zero wasn't out at the time)

To connect an HDMI device, you'll need 2 things, aMini HDMI to HDMI Adapter(http://adafru.it/2819) and an HDMI Cable (http://adafru.it/608) The HDMI cable is pretty straight-forward to understand, and you can get one anywhere. The HDMI adapter is required because the Pi Zero does not have a standard size HDMI port, instead the port is slimmer and smaller to keep the Zero petite. The adapter is pretty straight forward to use - plug it into the Pi Zero and the port is now large enough for any standard HDMI cable

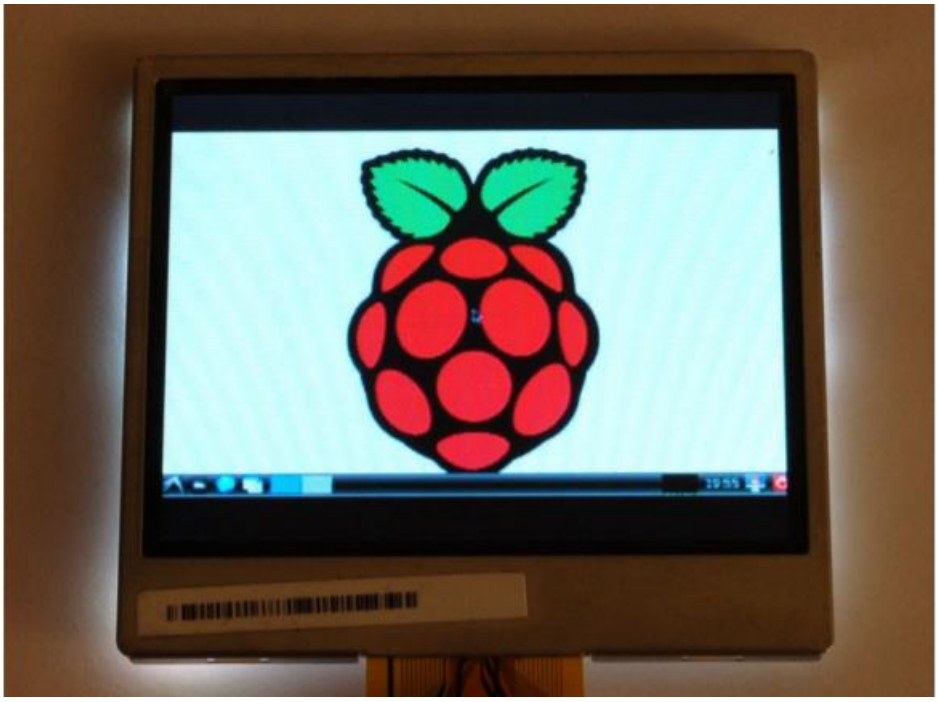
PITFT VIDEO :

Even though it is 'half size' of the A+, you can still use any of our PiTFT's on the Pi Zero (https://adafru.it/jE7) You can use any size from our 2.2" 320x240 PiTFT HAT, up to our 3.5" Touchscreen 480x320. Before you can plug in a HAT or PiTFTyou'll need to solder in the 2x20 male header (http://adafru.it/2822) Then follow the tutorial for the PiTFT of your choice! Be sure to pick the Jessie install image

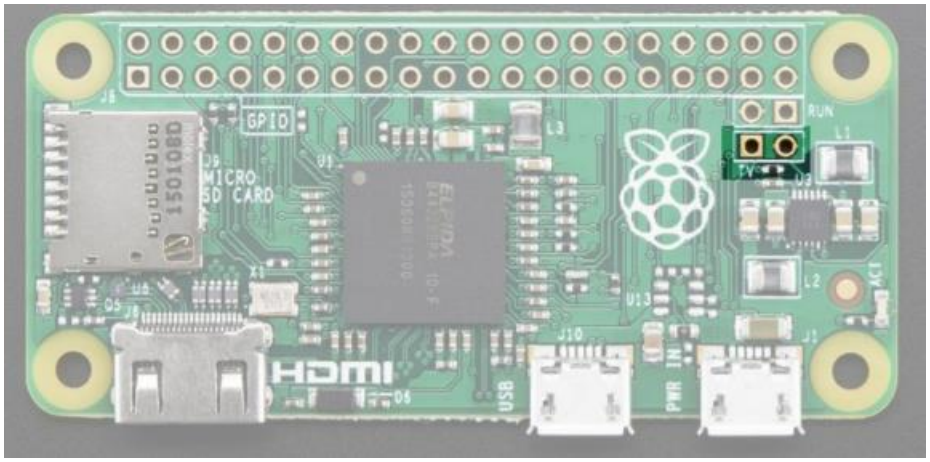


VGA Video Out This one is pretty easy, just use the HDMI adapter above, and anHDMI to VGA adapter (this also has the benefit of giving you an audio output) (http://adafru.it/1151)

NTSC/PAL Composite Video OK so you want TV video? Maybe for one of our very tiny composite video screens (https://adafru.it/jE8)?



Well, the quality is not going to be nearly as nice as with VGA or HDMI but you can do it. Find the two pads markedTV on the 'Zero



The hole on the left, nearest to the TV text, is the signal (+) line, the pin to the right of it is the ground (-) line. Solder two wires to these pads and connect them to an RCA Jack (http://adafru.it/2792) like this one

Make sure to not have HDMI plugged in, it should auto-switch to TV out. If you have somehow set your Pi for HDMI out only, plug your HDMI screen back in, or use a console cable to connect and log into the Pi. Then run sudoraspi-config at a command line to set video output to composite!You'll also want to tweak your Pi to use composite in the nicest resolution possible (https://adafru.it/diN)

Audio Outputs :

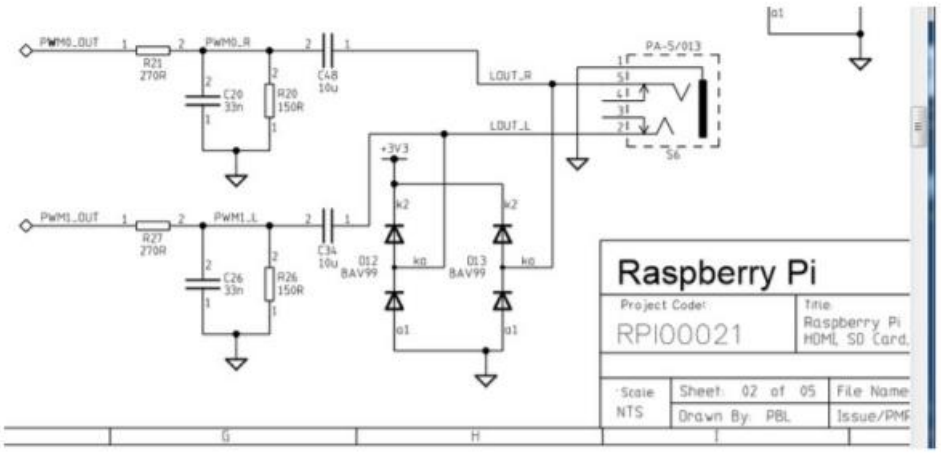
Uh, well, there aren't any! That's right, to keep the Pi Zero small and low cost, the headphone audio filter isn't included You can still get digital audio out via HDMI so if you plug it your Pi into a monitor with speakers, that will work fine.

How to Add Audio Outputs to your Pi Zero :

Hey, wanna do the below but with step-by-step instructions? We wrote a tutorial! Click here to do the thing @ https://learn.adafruit.com/adding-basic-audio-ouput-to-raspberry-pizero (https://adafru.it/jZD)

How Other Pi's Create Audio :

GPIO #18 is also known as PWM0 and in the original Pi was coupled with a very basic RC filter to create the audio output:



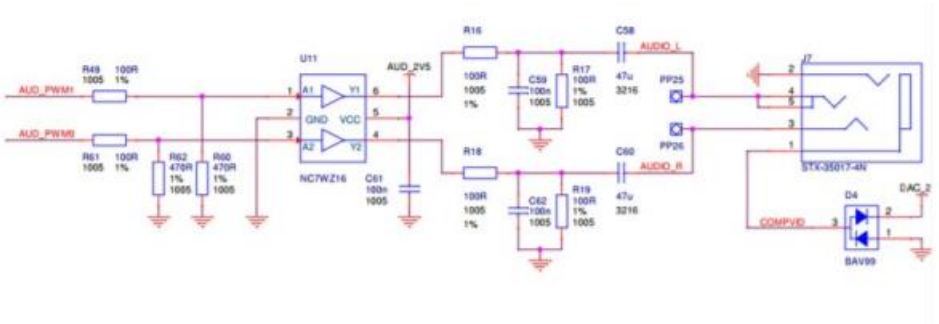
If you don't mind getting a few 150 and 270 ohm resistors, and two each of about 33nF (also known as 0.033uF) and 10uF capacitors, you can basically recreate those two filters.

Now all you need is access to PWM0_OUT and PWM1_OUT, which are...on GPIO #40 and #45 and are not brought out on the Pi Zero. Tragedy? Give up? No! You can get to PWM0 on GPIO #18 (ALT5) and PWM1 on GPIO #13 (ALT0) or GPIO #19 (ALT5) - see the full list of pins and alternate functions here (https://adafru.it/jEa)

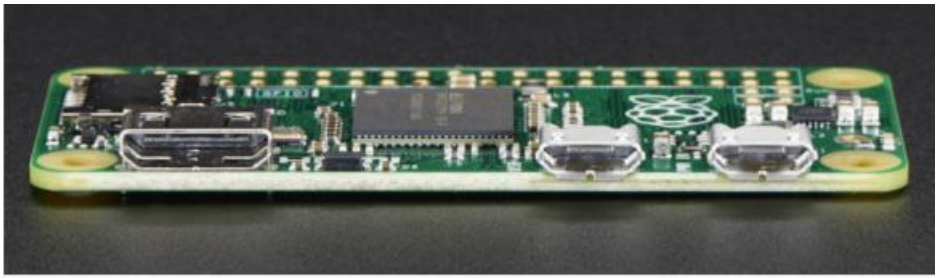
You can do that by adjusting the device tree overlay to change the PWM audio pins from pins #40 and #45 (which are not accessable) to pins #18 and #13 This very nice Pi forum thread will tell you how! (https://adafru.it/jEb)

See here for a program that will let you set the alt forms of GPIO pins (https://adafru.it/jEc)

If you want a higher quality audio output, the B+ and Pi 2 use this schematic - it has a driving buffer on the audio PWM lines for better current drive and it uses a cleaner 2.5V reference for better quality audio.



GPIO HEADER OPTIONS :



The most intriguing difference for hackers and makers is that the Pi Zero does not come with the soldered GPIO header. Partially this is to save cost, but it also allows the Pi Zero to be very thin and gives you the option of embedding it easily into a project box.

CONS:

You have to solder in the header to use Pi HATs and Pi toppers

PROS:

- You can practice your soldering!

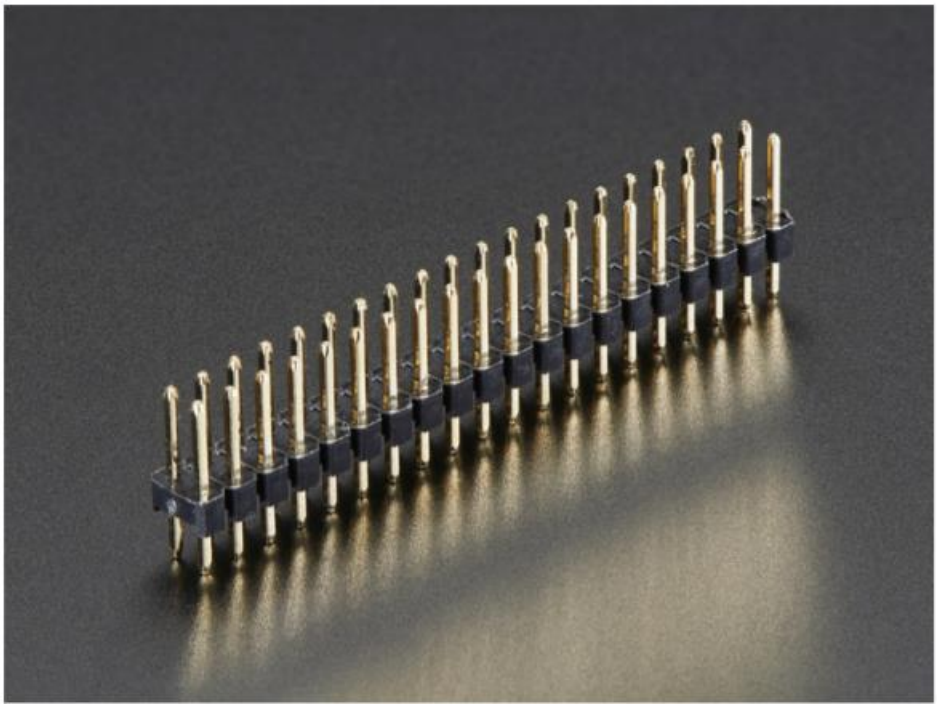
-Can skip the GPIO header to keep the Pi Zero super slim

-Solder wires directly into the GPIO pads, use only what you need

-Try different, exotic headers such as right angle or socket header

Go Classic with 2x20 Male Header:

Like blue jeans and Coca-Cola, the 2x20 male header is the classic option. (http://adafru.it/2822)

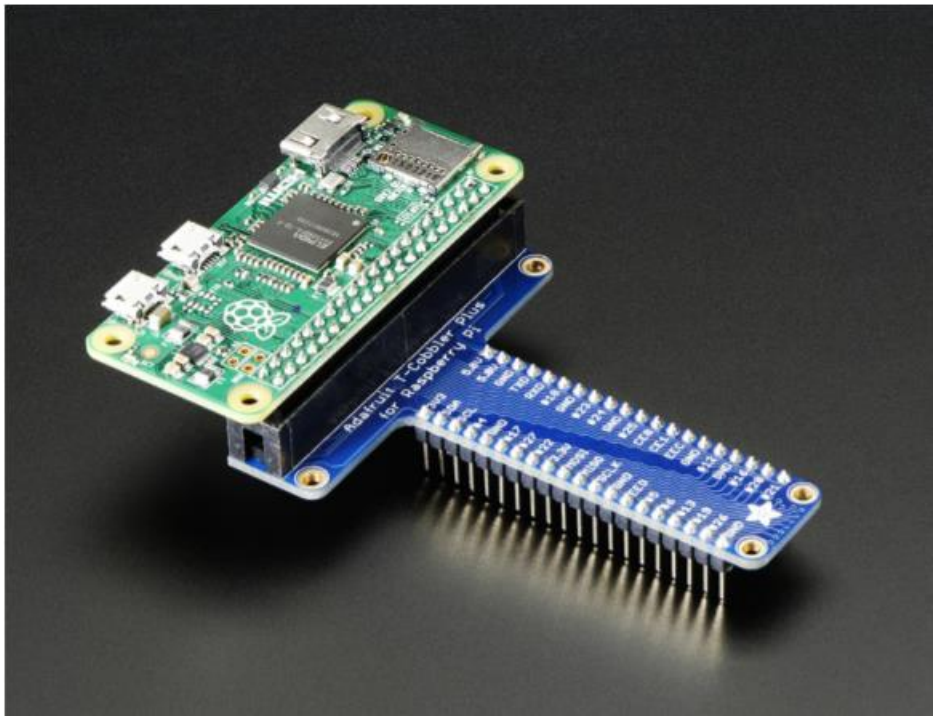


Once soldered in, you can plug in any HAT or topper. The pinout is completely identical to the 2x20 headers on the Pi 2 and Pi A+ & B+

OR 2X20 FEMALE SOCKET HEADER:

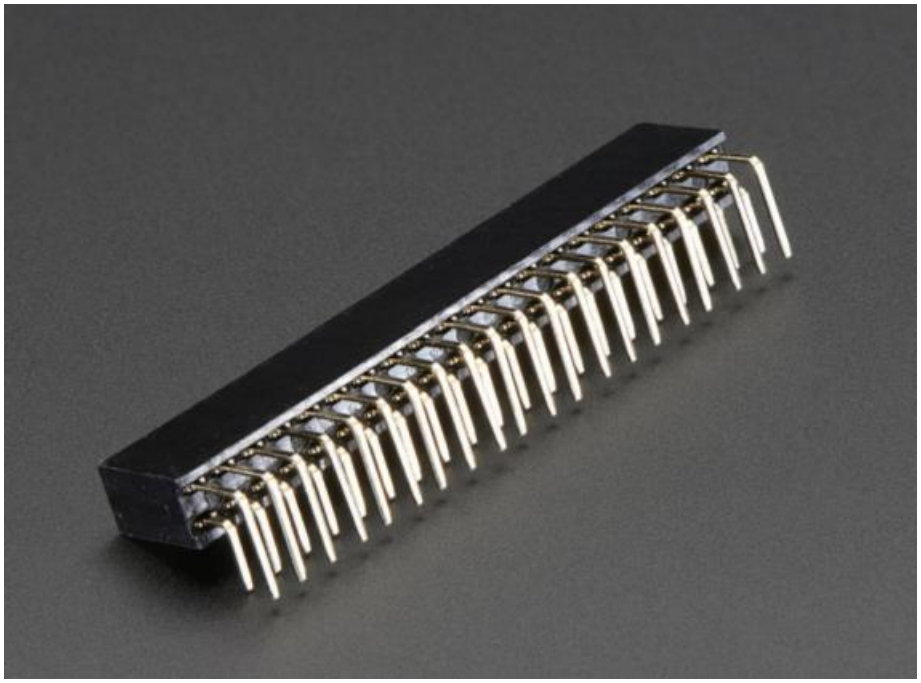
This one is interesting, if you solder in a 2x20 female socket header(http://adafru.it/2222)

and attach it upside down you can plug it right into a T-Cobbler!

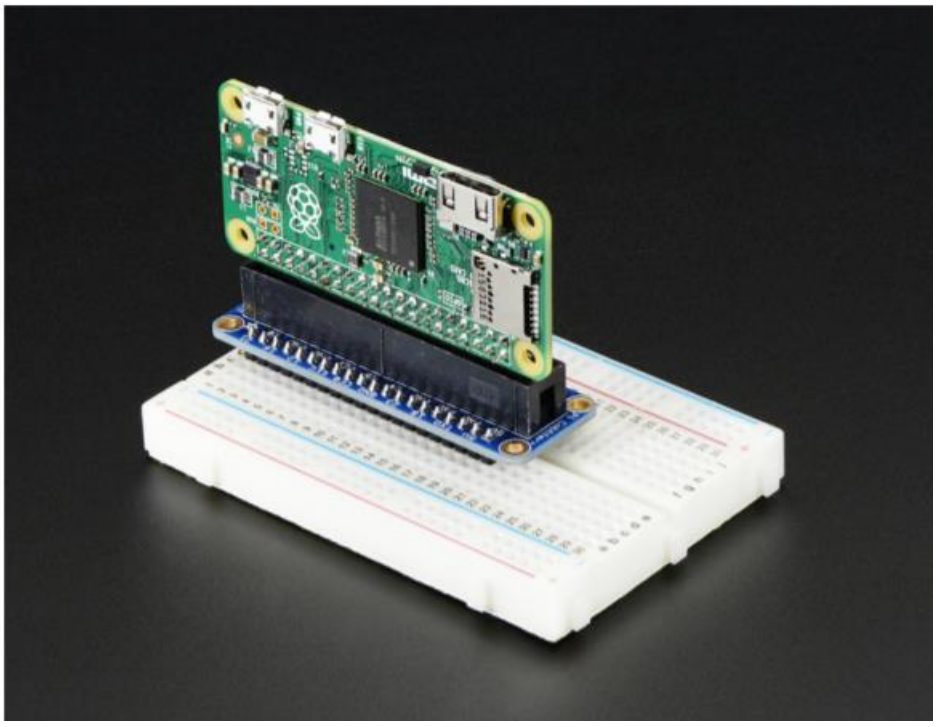


Advanced 2x20 Right-Angle Female Socket Header

Or, take it even more extreme with 2x20 right angle female header (http://adafru.it/2823)



Now you can stick it into a Cobbler or T-Cobbler and it will sit sort of like a computer daughtercard!



HOW TO CHECK IF THE RASPBERRY PI IS DEAD

The Pi Zero is so minimal, it can be tough to tell if its working at all. Here's how to do a quick check (from thissticky (https://adafru.it/upa))!

-Take your Zero, with nothing in any slot or socket(yes, no SD-card is needed or wanted to do this test!).

-Take a normal micro-USB to USB-A DATA SYNC cable (not a charge-only cable! make sure its a true data synccable!)

-Connect the USB cable to your PC, plugging the micro-USB into the Pi's USB, (not the PWR_IN).

-If the Zero is alive, your Windows PC will go ding for the presence of new hardware & you should see "BCM2708Boot" in Device Manager.

-Or on linux, run sudolsusb or run dmesg and look for a ID 0a5c:2763 Broadcom Corp message. If you see that,so far so good, you know the Zero's not dead.

I tested on Linux and here's my actual dmesg:

[226314.048026] usb 4-2: new full-speed USB device number 82 using uhci_hcd

[226314.213273] usb 4-2: New USB device found, idVendor=0a5c, idProduct=2763

[226314.213280] usb 4-2: New USB device strings: Mfr=1, Product=2, SerialNumber=0

[226314.213284] usb 4-2: Product: BCM2708 Boot

[226314.213288] usb 4-2: Manufacturer: Broadcom

but there will be NO LED LIGHT (looks so dead but its alive!)

Is There Even Life?

You can skip this section unless you have reason to believe your Pi Zero isn't alive.

The Pi Zero doesn't have much in the way of blinky LEDs to give you a warm fuzzy that it's doing anything or evenalive. And if the GPU doesn't find a valid OS image, it doesn't even turn on the green ACT LED and looks totally dead.Typically this just means something is up with the SD card. Bad card. Bad image. Out of date image. Whatever. It doesnot mean the Pi Zero is dead.

Here's how to run a sanity check to verify if the Pi Zero is OK.

(taken from here (https://adafru.it/upa)and also provided here (https://adafru.it/vIe))

-Take your Zero, with nothing in any slot or socket(yes, no SD-card is needed or wanted to do this test!).

-Take a normal micro-USB to USB-A DATA SYNC cable (not a charge-only cable! make sure its a true data synccable!)

-Connect the USB cable to your PC, plugging the micro-USB into the Pi's USB, (not the PWR_IN).

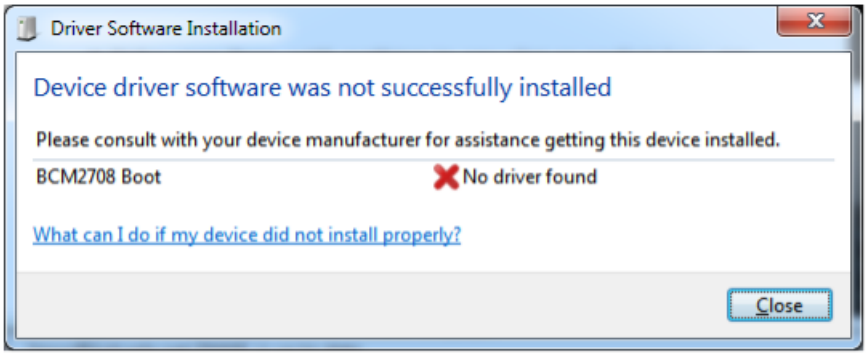
-If the Zero is alive, your Windows PC will go ding for the presence of new hardware & you should see "BCM2708Boot" in Device Manager.

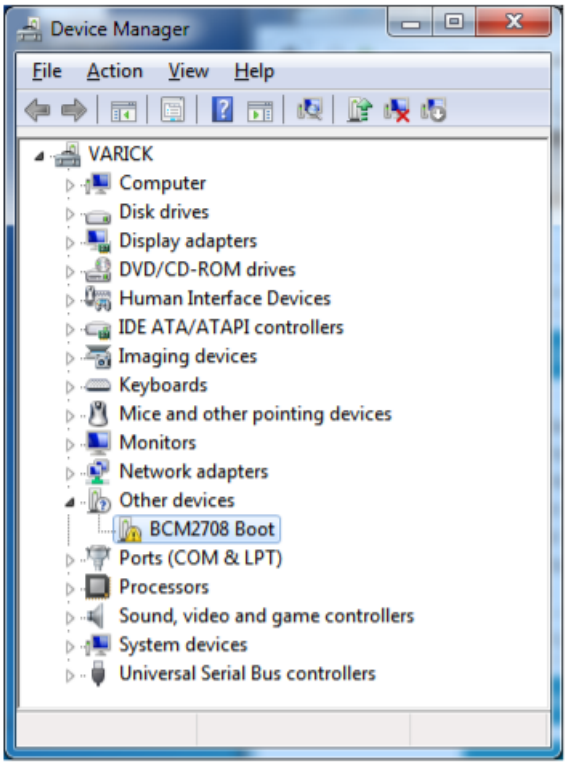
-Or on linux, run sudolsusb or run dmesg and look for a ID 0a5c:2763 Broadcom Corp message. If you see that,so far so good, you know the Zero's not dead.

Below is a Pi Zero connected to a Linux computer via a USB cable and the resulting dmesgoutput.Note: there is no

SD card installed, USB cable is in USB port, and there are no lights.

Here's what our Windows machine showed:


Driver Software Installation
Device driver software was not successfully installed
Please consult with your device manufacturer for assistance getting this device installed.
BCM2708 Boot
No driver found
What can I do if my device did not install properly?
Close


Device Manager
File Action View Help
VARICK
Computer
Disk drives
Display adapters
DVD/CD-ROM drives
Human Interface Devices
IDE ATA/ATAPI controllers
Imaging devices
Keyboards
Mice and other pointing devices
Monitors
Network adapters
Other devices
BCM2708 Boot
Ports (COM & LPT)
Processors
Sound, video and game controllers
System devices
Universal Serial Bus controllers

CHAPTER 3

GPS MODULE

3.1 FUNCTIONAL DESCRIPTION

OVERVIEW

The NEO-6 module series is a family of stand-alone GPS receivers featuring the high performance u-blox 6 positioningengine.Theseflexibleandcosteffectivereceiversoffernumerousconnectivityoptionsinaminiature 16x12.2x2.4mmpackage.Theircompactarchitectureandpowerandmemoryoptionsmake NEO-6modules idealforbatteryoperatedmobiledeviceswithverystrictcostandspaceconstraints.

The50-channelu-blox6positioningengineboastsaTime-To-First-Fix(TTFF)ofunder1second.Thededicated acquisition engine, with 2 million correlators, is capable of massive parallel time/frequency space searches, enabling it to find satellites instantly. Innovative design and technology suppresses jamming sources and mitigates multipath effects, giving NEO-6 GPS receivers excellent navigation performance even in the most challengingenvironments.

3.2 PRODUCT FEATURES

Model	Type					Supply		Interfaces				Features						
	GPS	PPP	Timing	Raw Data	Dead Reckoning	1.75 V - 2.0 V	2.7 V - 3.6 V	UART	USB	SPI	DDC (I²C compliant)	Programmable (Flash) FW update	TCXO	RTC crystal	Antenna supply and supervisor	Configuration pins	Timepulse	External interrupt/ Wakeup
NEO-6G	●					●		●	●	●	●		●	●	○	3	1	●
NEO-6Q	●						●	●	●	●	●		●	●	○	3	1	●
NEO-6M	●						●	●	●	●	●			●	○	3	1	●
NEO-6P	●	●		●			●	●	●	●	●			●	○	3	1	●
NEO-6V	●				●		●	●	●	●	●			●	○	3	1	●
NEO-6T	●		●	●			●	●	●	●	●		●	●	○	3	1	●

○ = Requires external components and integration on application processor

TABLE 1: FEATURES OF THE NEO-6 SERIES

- AllNEO-6 modules are based on GPS chips qualified according to AEC-Q100.

3.3 GPS PERFORMANCE

Parameter	**Specification**			
Receiver type	50 Channels GPS L1 frequency, C/A Code SBAS: WAAS, EGNOS, MSAS			
Time-To-First-Fix[1]		NEO-6G/Q/T	NEO-6M/V	NEO-6P
	Cold Start[2]	26 s	27 s	32 s
	Warm Start[2]	26 s	27 s	32 s
	Hot Start[2]	1 s	1 s	1 s
	Aided Starts[3]	1 s	<3 s	<3 s
Sensitivity[4]		NEO-6G/Q/T	NEO-6M/V	NEO-6P
	Tracking & Navigation	-162 dBm	-161 dBm	-160 dBm
	Reacquisition[5]	-160 dBm	-160 dBm	-160 dBm
	Cold Start (without aiding)	-148 dBm	-147 dBm	-146 dBm
	Hot Start	-157 dBm	-156 dBm	-155 dBm
Maximum Navigation update rate		NEO-6G/Q/M/T	NEO-6P/V	
		5Hz	1 Hz	
Horizontal position accuracy[6]	GPS	2.5 m		
	SBAS	2.0 m		

	SBAS + PPP[7]	< 1 m (2D, R50)[8)]		
	SBAS + PPP[7]	< 2 m (3D, R50)[8]		
Configurable Timepulse frequency range		NEO-6G/Q/M/P/V	NEO-6T	
		0.25 Hz to 1 kHz	0.25 Hz to 10 MHz	
Accuracy for Timepulse signal	RMS	30 ns		
	99%	<60 ns		
	Granularity	21 ns		
	Compensated[9]	15 ns		
Velocity accuracy[6]		0.1m/s		
Heading accuracy[6]		0.5 degrees		
Operational Limits	Dynamics	4 g		
	Altitude[10]	≤ 50,000 m		
	Velocity[10]	500 m/s		

TABLE 2: NEO-6 GPS PERFORMANCE

- Without aiding
- All satellites are -130 dB
- Dependent on aiding data connection speed and latency
- Demonstrated with a good active antenna
- For an outage duration 10s
- CEP, 50%, 24 hours static, -130dBm, SEP: <3.5m
- NEO-6P only
- Demonstrated under following conditions: 24hours, stationary, first 600 seconds of data discarded. HDOP<1.5duringmeasurement period, strong signals. Continuous availability of valid SBAS correction data during full test period.
- Quantization error information can be used with NEO-6T to compensate the granularity related error of the time pulse signal
- Assuming Airborne <4g platform

3.4 BLOCK DIAGRAM

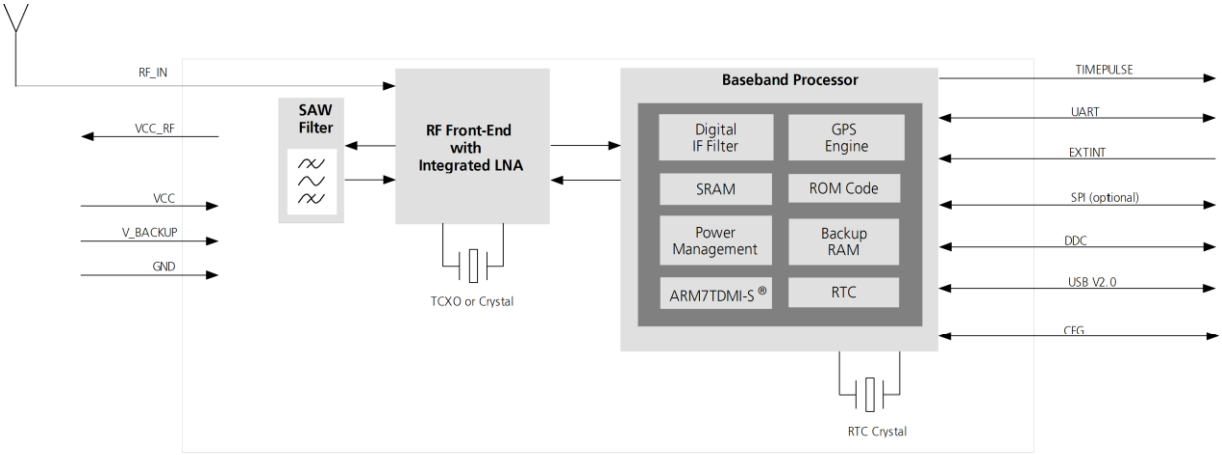


FIGURE 1: BLOCK DIAGRAM (FOR AVAILABLE OPTIONS REFER TO THE PRODUCT FEATURES TABLE IN SECTION 1.2).

ASSISTED GPS (A-GPS)

Supply of aiding information like ephemeris, almanac, rough last position and time and satellite status and an optional time synchronization signal will reduce time to first fix significantly and improve the acquisition sensitivity.AllNEO-6modulessupporttheu-bloxAssistNowOnlineandAssistNowOfflineA-GPSservices[11]and are OMA SUPLcompliant.

ASSIST NOW AUTONOMOUS

AssistNowAutonomousprovidesfunctionalitysimilartoAssisted-GPSwithouttheneedforahostorexternal networkconnection.Basedonpreviouslybroadcastsatelliteephemerisdatadownloadedtoandstoredbythe GPS receiver, Assist Now Autonomous automatically generates accurate satellite orbital data (“Assist Now Autonomous data”) that is usable for future GPS position fixes. Assist Now Autonomous data is reliable for up to 3 days after initial capture.

u-blox’ Assist Now

- Autonomous benefits are:
- Faster position fix
- No connectivity required

Complementary with Assist Now Online and Offline services No integration effort, calculations are done in the background

PRECISION TIMING

TIME MODE

NEO-6T provides a special Time Mode to provide higher timing accuracy. The NEO-6T is designed for use with stationary antenna setups. The Time Mode features three different settings described in Table 3: Disabled, Survey-In and Fixed Mode. For optimal performance entering the position of the antenna (when known) is recommended as potential source of errors will be reduced.

Time Mode Settings	Description
Disabled	Standard PVT operation
Survey-In	The GPS receiver computes the average position over an extended time period until a predefined maximum standarddeviationhasbeenreached.Afterwardsthereceiverwillbeautomatically setto Fixed Mode and the timing features will be activated.
Fixed Mode	Inthismode,afixed3Dpositionandknownstandarddeviationisassumedandthetimingfeaturesare activated.FixedModecaneitherbeactivateddirectlybyfeedingpre-definedpositioncoordinates(ECEF - Earth Center Earth Fixed format) or by performing a Survey-In. InFixedmode,thetimingerrorsintheTIMEPULSEsignalwhichotherwiseresultfrompositioningerrors areeliminated.Single-satelliteoperationissupported.Fordetails,pleaserefertothe*u-blox6Receiver Description including Protocol Specification* [2].

TABLE 3: TIME MODE SETTINGS

Time pulse and frequency reference

NEO-6T comes with a time pulse output which can be configured from 0.25Hz upto 10MHz. The time pulse can either be used for time synchronization (i.e.1pulsepersecond) or as a reference frequency in the MHz range. A time pulse in the MHz range provides excellent long-term frequency accuracy and stability.

Time mark

NEO-6T can be used for precise time measurements with sub-micro second resolution

using the external interrupt (EXTINT0). Rising and falling edges of these signals are time-stamped to the GPS or UTC time and counted. The Time Mark functionality can be enabled with the UBX-CFG-TM2 message

Raw data

Raw data output is supported at an update rate of 5 Hz on the NEO-6T and NEO-6P. The UBX-RXM-RAW message includes carrier phase with half-cycle ambiguity resolved, code phase and Doppler measurements, which can be used in external applications that offer precision positioning, real-time kinematics (RTK) and attitude sensing.

Automotive DeadReckoning

Automotive Dead Reckoning (ADR) isu-blox'industry proven off-the-shelf Dead Reckoning solution for tier-one automotive customers. u-blox' ADR solution combines GPS and sensor digital data using a tightly coupled Kalmanfilter.Thisimprovespositionaccuracyduringperiodsofnoordegraded GPSsignal.

TheNEO-6V provides ADR functionality over its software sensor interface. Avariety of sensors (such as wheel ticks and gyroscope) are supported, with the sensor data received via UBX messages from the application processor. This allows for easy integration and a simple hardware interface, lowering costs. By using digital sensor data available on the vehicle bus, hardware costs are minimized since no extra sensors are required for Dead Reckoning functionality. ADR is designed for simple integration and easy configuration of different sensor options (e.g.withorwithoutgyroscope) and vehicle variants, and is completely self-calibrating.

Precise PointPositioning

u-blox' industry proven PPP algorithm provides extremely high levels of position accuracy in static and slow movingapplications,andmakestheNEO-6Panidealsolutionforavarietyofhighprecisionapplicationssuchas surveying, mapping, marine, agriculture or leisureactivities.

Ionospheric corrections such as those received from local SBAS[12] geostationary satellites (WAAS, EGNOS, MSAS) or from GPS enable the highest positioning accuracy with the PPP algorithm. The maximum improvement of positioning accuracy is reached with PPP+SBA Sand can only be expected in an environment with unobstructed sky view during a period in the order of minutes.

Oscillators

NEO-6 GPS modules are available in Crystal and TCXO versions. The TCXO allows accelerated weak signal acquisition, enabling faster start and reacquisition times.

Protocols and interfaces

Protocol	Type
NMEA	Input/output, ASCII, 0183, 2.3 (compatible to 3.0)
UBX	Input/output, binary, u-blox proprietary
RTCM	Input, 2.3

Table 4: Available protocols

UART

NEO-6 modules include one configurable UART interface for serial communication (for information about configuration see section 1.15).

USB

NEO-6modulesprovideaUSBversion2.0FS(FullSpeed,12Mbit/s)interfaceasanalternativetotheUART.The pull-upresistoronUSB_DPisintegratedtosignalafull-speeddevicetothehost.TheVDDUSBpinsuppliesthe USBinterface.u-bloxprovidesaMicrosoft®certifiedUSBdriverforWindowsXP,WindowsVistaandWindows 7 operatingsystems.

Serial Peripheral Interface(SPI)

The SPI interface allows for the connection of external devices with a serial interface, e.g. serial flash to save configurationandAssistNowOfflineA-GPSdataortointerfacetoahostCPU.Theinterfacecanbeoperatedin masterorslavemode.Inmastermode,onechipselectsignalisavailabletoselectexternalslaves.In slavemodea single chip select signal enables communication with thehost.

- The maximum bandwidth is 100kbit/s.

Display Data Channel(DDC)

They C compatible DDC interface can be used either to access external devices with a serial interface EEPROM or to interface with a host CPU. It is capable of master and slave operation. The DDC interface is IC Standard Mode compliant. For timing parameters consult the I^2C standard.

- The DDC Interface supports serial communication with ublox wireless modules. See the specification of the applicable wireless module to confirm compatibility.
- The maximum bandwidth is 100kbit/s.

1.1.1.1 External serial EEPROM

NEO-6 modules allow an optional external serial EEPROM to be connected to the DDC interface. This can be used to store Configurations permanently.

For more information see the *LEA-6/NEO-6/MAX-6 Hardware Integration Manual* [1].

- **Use caution when implementing since forward compatibility is not guaranteed.**

Antenna

NEO-6 modules are designed for use with passive and active[13] antennas.

Parameter	Specification	
Antenna Type		Passive and active antenna
Active Antenna Recommendations	Minimum gain Maximum gain Maximum noise figure	15dB(tocompensatesignallossinRFcable) 50dB 1.5 dB

Table 5: Antenna Specifications for all NEO-6 modules

PowerManagement

u-blox receivers support different power modes. These modes represent strategies of how to control the acquisitionandtrackingenginesinordertoachieveeitherthebestpossibleperformanceorgoodperformance with reduced powerconsumption.

Maximum PerformanceMode

During a Cold start, a re ceiver in Maximum Performance Mode continuously deploys the acquisition engine to search for all satellites. Once the receiver has a position fix (or if pre-positioning information is available), the acquisition engine continues to be used to search for all visible at ellites that are not being tracked.

EcoMode

During a Cold start, a receiver in Eco Mode works exactly as in Maximum Performance Mode. Once a position can be calculated and a sufficient number of satellites are being tracked, the acquisition engine is powered off resulting in significant power savings. The tracking engine continuously tracks acquired satellites and acquires other available or emerging satellites.

- Note that even if the acquisition engine is powered off, satellites continue to be acquired.

Power SaveMode

Power Save Mode (PSM) allows a reduction in system power consumption by selectively switching parts of the receiver on and off.

Configuration

Boot-time configuration

NEO-6 modules provide configuration pins for boot-time configuration. These become effective immediately after start-up. Once the module has started, the configuration settings can be modified with UBX configuration messages. The modified settings remain effective until power-down or reset. If these settings have been stored in battery-backup RAM,then the modified configuration will be retained, as long as the backup battery supply is not interrupted.

NEO-6 modules include both **CFG_COM0** and **CFG_COM1** pins and can be configured as seen in Table 6. Default settings in bold.

CFG_COM1	**CFG_COM0**	**Protocol**	**Messages**	**UART Baud rate**	**USB power**
1	**1**	**NMEA**	**GSV, RMC, GSA, GGA, GLL, VTG, TXT**	**9600**	**BUS Powered**
1	0	NMEA	GSV, RMC, GSA, GGA, GLL, VTG, TXT	38400	Self Powered
0	1	NMEA	GSV[14], RMC, GSA, GGA, VTG, TXT	4800	BUS Powered
0	0	UBX	NAV-SOL, NAV-STATUS, NAV-SVINFO, NAV-CLOCK, INF, MON-EXCEPT, AID-ALPSERV	57600	BUS Powered

Table 6: Supported COM settings

NEO-6modules include a **CFG_GPS0** pin, which enables the boot-time configuration of the power mode. These settings are described in Table 7. Default settings in bold.

CFG_GPS0	**Power Mode**

0	Eco Mode
1	**Maximum Performance Mode**

Table 7: Supported CFG_GPS0 settings

- Static activation of the CFG_COM and CFG_GPS pins is not compatible with use of the SPI interface.

Design-in

In order to obtain the necessary information to conduct a proper design-in, u-blox strongly recommends consulting the *LEA-6/NEO-6/MAX-6 Hardware Integration Manual* [1].

PinDefinition

Pinassignment

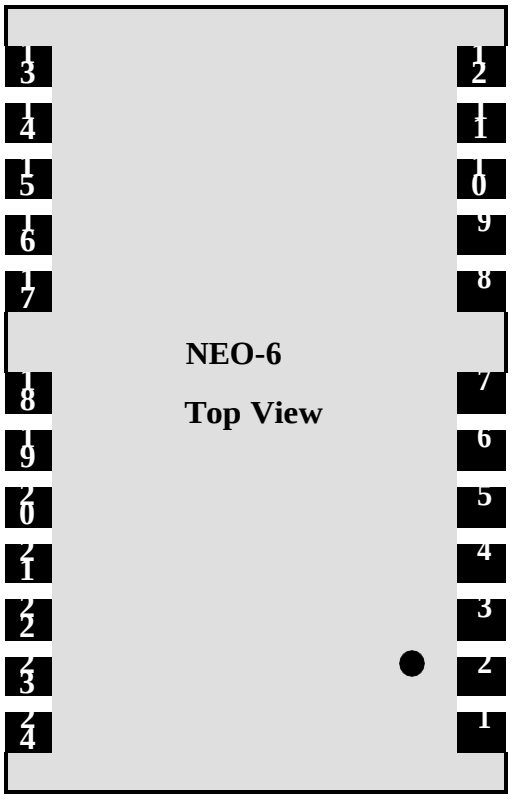


Figure 2 Pin Assignment

No	Module	Name	I/O	Description
1	All	Reserved	I	Reserved
2	All	SS_N	I	SPI Slave Select
3	All	TIMEPULSE	O	Timepulse (1PPS)
4	All	EXTINT0	I	External Interrupt Pin
5	All	USB_DM	I/O	USB Data
6	All	USB_DP	I/O	USB Data
7	All	VDDUSB	I	USB Supply
8	All	Reserved		See Hardware Integration Manual Pin 8 and 9 must be connected together.
9	All	VCC_RF	O	Output Voltage RF section
				Pin 8 and 9 must be connected together.
10	All	GND	I	Ground
11	All	RF_IN	I	GPS signal input
12	All	GND	I	Ground
13	All	GND	I	Ground
14	All	MOSI/CFG_COM0	O/I	SPI MOSI / ConfigurationPin. Leave open if not used.
15	All	MISO/CFG_COM1	I	SPI MISO / ConfigurationPin. Leave open if not used.
16	All	CFG_GPS0/SCK	I	PowerModeConfigurationPin/SPIClock. Leave open if notused.
1	All	Reserve	I	Reserved

7		d		
18	All	SDA2	I/O	DDC Data
19	All	SCL2	I/O	DDC Clock
20	All	TxD1	O	Serial Port 1
21	All	RxD1	I	Serial Port 1

No	**Module**	**Name**	**I/O**	**Description**
22	All	V_BCKP	I	Backup voltage supply
23	All	VCC	I	Supply voltage
24	All	GND	I	Ground

Table 8: Pin out

Electrical specifications

Absolute maximum ratings

Parameter	**Symbol**	**Module**	**Min**	**Max**	**Units**	**Condition**
Power supply voltage	VCC	NEO-6G	-0.5	2.0	V	
		NEO-6Q, 6M, 6P, 6V, 6T	-0.5	3.6	V	
Backup battery voltage	V_BCKP	All	-0.5	3.6	V	
USB supply voltage	VDDUSB	All	-0.	3.6	V	

			5			
Input pin voltage	Vin	All	-0.5	3.6	V	
	Vin_usb	All	-0.5	VDDUSB	V	
DC current trough any digital I/O pin (except supplies)	Ipin			10	mA	
VCC_RF output current	ICC_RF	All		100	mA	
Input power at RF_IN	Prfin	NEO-6Q, 6M, 6G, 6V, 6T		15	dBm	source impedance = 50Ω, continuous wave
		NEO-6P		-5	dBm	
Storage temperature	Tstg	All	-40	85	°C	

Table 9: Absolute maximum ratings

Operating conditions

All specifications are at an ambient temperature of25°C.

Parameter	**Symbol**	**Module**	**Min**	**Typ**	**Max**	**Units**	**Condition**
Power supply voltage	VCC	NEO-6G	1.75	1.8	1.95	V	
		NEO-	2.7	3.0	3.6	V	

		6Q/M NEO-6P/V/T					
Supply voltage USB	VDDUSB	All	3.0	3.3	3.6	V	
Backup battery voltage	V_BCKP	All	1.4		3.6	V	
Backup battery current	I_BCKP	All		22		µA	V_BCKP = 1.8 V, VCC = 0V
Input pin voltage range	Vin	All	0		VCC	V	
Digital IO Pin Low level input voltage	Vil	All	0		0.2*VCC	V	
Digital IO Pin High level input voltage	Vih	All	0.7*VCC		VCC	V	
Digital IO Pin Low level output voltage	Vol	All			0.4	V	Iol=4mA
Digital IO Pin High level output voltage	Voh	All	VCC-0.4			V	Ioh=4mA
USB_DM, USB_DP	VinU	All	Compatible with USB with 22 Ohms series resistance				

VCC_RF voltage	VCC_RF	All		VCC-0.1		V	
VCC_RF output current	ICC_RF	All			50	mA	
Antenna gain	Gant	All			50	dB	
Receiver Chain Noise Figure	NFtot	All		3.0		dB	
Operating temperature	Topr	All	-40		85	°C	

Table 10: Operating conditions

- Operation beyond the specified operating conditions can affect device reliability.

Indicative power requirements

Table 11 lists examples of the total system supply current for a possible application.

Parameter	Symbol	Module	Min	Type	Max	Units	Condition
Max. supply current [15]	Iccp	All			67	mA	VCC = 3.6 V[16] / 1.95 V[17]
	Icc Acquisition	All		47[19]		mA	
	Icc Tracking (Max Performance mode)	NEO-6G/Q/T		40[20]		mA	
		NEO-6M/P/V		39[20]		mA	
Average supply current[18]	Icc Tracking (Eco mode)	NEO-6G/Q/T		38[20]		mA	VCC = 3.0 V[16] / 1.8 V[17]
		NEO-6M/P/V		37[20]		mA	
	Icc Tracking (Power Save mode / 1 Hz)	NEO-6G/Q		12[20]		mA	
		NEO-6M		11[20]		mA	

Table 11: Indicative power requirements

Values in Table 11 are provided for customer information only as an example of typical power requirements. Values are characterized on samples, actual power requirements can vary depending on FW version used, external circuitry, number of SVs tracked, signal strength, type of start as well as time, duration and conditions of test.

SPI timing diagrams

In order to avoid a faulty usage of the SPI, the user needs to comply with certain timing conditions. The following signals need to be considered for timing constraints:

Symbol	Description
SS_N	Slave Select signal
SCK	Slave Clock signal

Table 12: Symbol description

Figure 3: SPI timing diagram

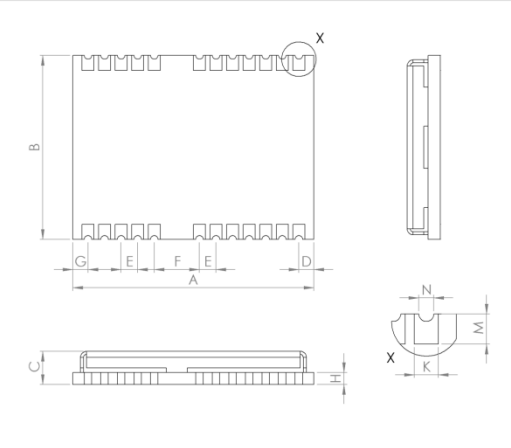
Timing recommendations

Parameter	Description	Recommendation
t_{INIT}	Initialization Time	500 s
t_{DES}	Deselect Time	1 ms
Bitrate		100 kbit/s

Table 13: SPI timing recommendations

- The maximum bandwidth is 100kbit/s[21].

Mechanical specifications



Symbol	Min. (mm)	Typ. (mm)	Max. (mm)
A	15.9	16.0	16.6
B	12.1	12.2	12.3
C	2.2	2.4	2.6
D	0.9	1.0	1.3
E	1.0	1.1	1.2
F	2.9	3.0	3.1
G	0.9	1.0	1.3
H		0.82	
K	0.7	0.8	0.9
M	0.8	0.9	1.0
N	0.4	0.5	0.6
Weight		1.6g	

Figure 4: Dimensions

Product handling &soldering

Packaging

NEO-6 modules are delivered as hermetically sealed, reeled tapes in order to enable efficient production, production lot set-up and tear-down. For more information about packaging, see the *u-blox Package Information Guide* [4].

Figure 5: Reeled u-blox 6 modules

Reels

Parameter	Specification
Reel Type	B
Delivery Quantity	250

Table 14: Reel information for NEO-6 modules

Tapes

Figure6showsthepositionandorientationofNEO-6modulesastheyaredeliveredontape.Thedimensionsof the tapes are specified in Figure7.

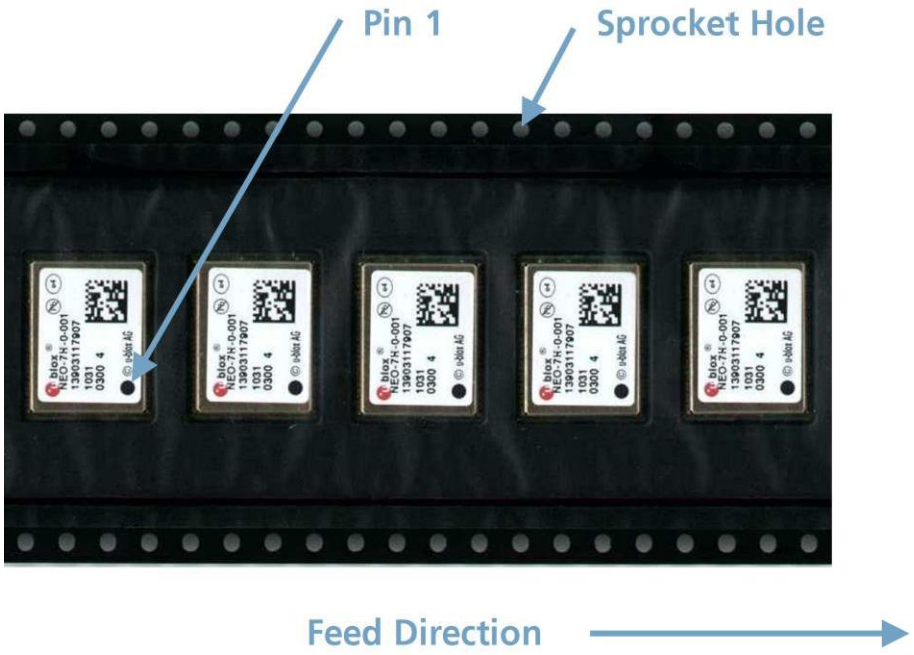


Figure 6: Orientation for NEO-6 modules on tape

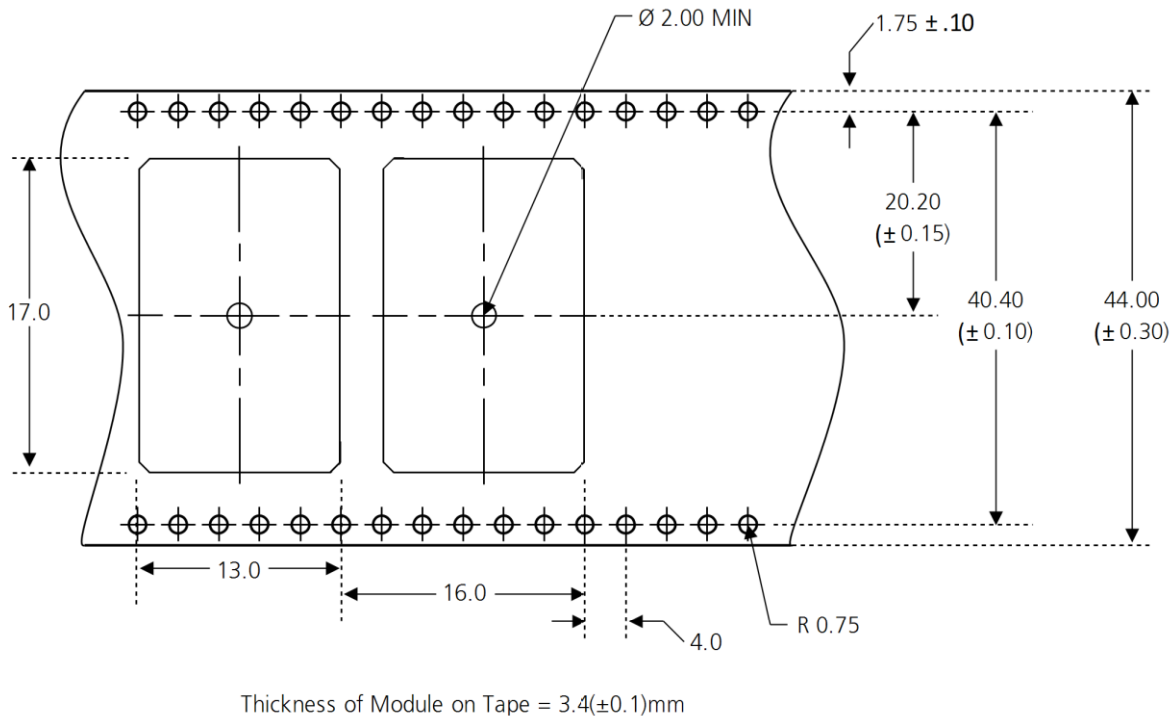


Figure 7: NEO tape dimensions (mm)

Moisture Sensitivity Levels

NEO-6 modules are Moisture Sensitive Devices (MSD) in accordance to the IPC/JEDEC specification.

NEO-6 modules are rated at MSL level 4. For more information regarding moisture sensitivity levels, labeling, storage and drying see the *u-blox Package Information Guide* [4].

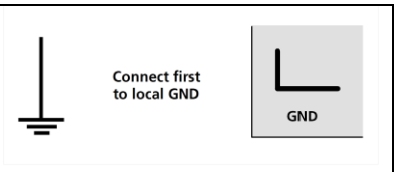
Reflow soldering

Reflow profiles are to be selected according to u-blox recommendations (see *LEA-6/NEO-6/MAX-6Hardware Integration Manual* [1]).

ESD handling precautions

NEO-6 modules contain highly sensitive electronic circuitry and are Electrostatic Sensitive Devices (ESD). Observe precautions for handling! Failure to observe these precautions can result in severe damage to the GPS receiver!

GPS receivers are Electrostatic Sensitive Devices (ESD) and require special precautions when handling. Particular care must be exercised when handling patch antennas, due to the risk of electrostatic charges. In addition to standard ESD safety practices, the following measures should be taken into account whenever handling the receiver:

• Unless there is a galvanic coupling between the local GND (i.e. the worktable) and the PCB GND, then the first point of contact when handling the PCB must always be between the local GND and PCB GND. • Before mounting antenna patch, connect ground of the device	
• When handling the RF pin, do not come into contact with any charged capacitors and be careful when contacting materials that can develop charges (e.g. patch antenna ~10pF, co ax cable ~50- 80pF/m, soldering iron,…)	
• To prevent electrostatic discharge through the RF input, do not touch any exposed antenna area. If there is any risk that such exposed antenna area is touched in non ESD protected work area, implement proper ESD protection measures in the design.	
• When soldering RF connectors and patch antennas to the receiver's RF pin, make sure to use an ESD safe soldering iron (tip).	

Defaultsettings

Interface	**Settings**
Serial Port 1 Output	9600 Baud, 8 bits, no parity bit, 1 stop bit Configured to transmit both NMEA and UBX protocols, but only following NMEA and no UBX messages have been activated at start-up: **GGA, GLL, GSA, GSV, RMC, VTG, TXT** (In addition to the 6 standard NMEA messages the NEO-6T includes **ZDA**).
USB Output	ConfiguredtotransmitbothNMEAandUBXprotocols,butonlyfollowingNMEAandno UBX messages have been activated at start-up: **GGA, GLL, GSA, GSV, RMC, VTG, TXT**

	(Inadditiontothe6standardNMEAmessagestheNEO-6Tincludes**ZDA**). USB Power Mode: Bus-Powered
Serial Port 1 Input	9600 Baud, 8 bits, no parity bit, 1 stop bit Automatically accepts following protocols without need of explicit configuration: **UBX, NMEA** The GPS receiver supports interleaved UBX and NMEA messages.
USB Input	Automatically accepts following protocols without need of explicit configuration: **UBX, NMEA** The GPS receiver supports inter leaved UBX and NMEA messages. USB Power Mode: Bus-Powered
TIMEPULSE (1Hz Nav)	1 pulse per second, synchronized at rising edge, pulse length 100ms
Power Mode	Maximum Performance mode
Assist Now Autonomous	Disabled.

Table 15: Default settings

Labeling and ordering information

Product labeling

The labeling of u-blox6 GPS modules includes important product information. The location of the product type number is shown

Explanation of codes

3 different product code formats are used. The **Product Name** is used in document at ion such as this data sheet and identifies all u-blox6 products, independent of packaging and quality grade. The **Ordering Code** includes options and quality, while the **Type Number** includes the hardware and firmware versions. Table 16 below details these 3 different formats:

Format	Structure
Product Name	PPP-GV
Ordering Code	PPP-GV-T
Type Number	PPP-GV-T-XXX

Table 16: Product Code Formats

The parts of the product code are explained in Table 17.

Code	Meaning	Example
PPP	Product Family	NEO
G	Product Generation	6 = u-blox6
V	Variant	T = Timing, R = DR, etc.
T	Option / Quality Grade	Describes standardized functional element or quality grade such as Flash size, automotive grade etc.
XXX	Product Detail	Describes product details or options such as hard-and software revision, cable length, etc.

Table 17: part identification code

CHAPTER 4

INSTALLING OPEN CV IN RASPBERRY PI

Step #1: Expand file system

If you're using a brand new install of Raspbian Jessie, then the first thing you should do is ensure your file system has been expanded to include all available space on your micro-SD card:

Installing OpenCV on your Raspberry Pi Zero

$ sudo raspi-config

Select the first option "1. Expand File system", arrow down to "Finish", and reboot you're Pi:

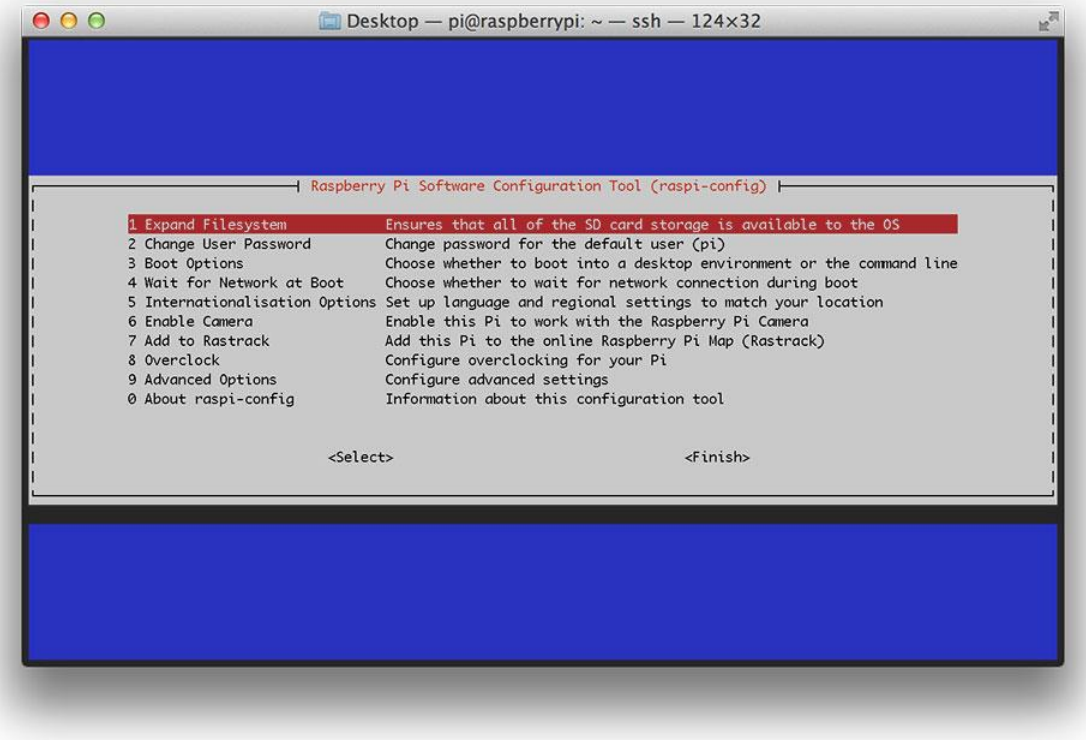


Figure 1: Expanding the file system on your Raspberry Pi Zero.

After rebooting, your file system will have be expanded to include all available space on your micro-SD card.

Step #2: Install dependencies

I've discussed each of these dependencies are in previous posts, so I'll just provide a brief description, the command(s) themselves, along with the amount of time it takes to execute each command so you can plan your OpenCV install accordingly (the compilation of OpenCV alone takes 9+ hours).

First, we need to update and upgrade our existing packages:

Installing OpenCV on your Raspberry Pi Zero

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

Timing: 2m 29s

Install our developer tools:

Installing OpenCV on your Raspberry Pi Zero

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

Timing: 49s

Let's grab the image I/O packages and install them:

Installing OpenCV on your Raspberry Pi Zero

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

Timing: 36s

Along with some video I/O packages (although it's unlikely that you'll be doing a lot of video processing with the Raspberry Pi Zero):

Installing OpenCV on your Raspberry Pi Zero

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

Timing: 36s

We'll need to install the GTK development library for OpenCV's GUI interface:

Installing OpenCV on your Raspberry Pi Zero

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

Timing: 2m 57s

Let's also pull down a couple routine optimization packages leveraged by OpenCV:

Installing OpenCV on your Raspberry Pi Zero

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

Timing: 52s

Lastly, let's install the Python 2.7 headers so wen can compile our OpenCV + Python bindings:

Installing OpenCV on your Raspberry Pi Zero

```
$ sudo apt-get install python2.7-dev
```

Timings: 55s

Note: I'll only be covering how to install OpenCV 3 with Python 2.7 bindings in this post. If you would like to install OpenCV 3 with Python 3 bindings

Step #3: Grab the OpenCV source

At this point, all of our dependences have been installed, so let's grab the

3.0.0

release of OpenCV from GitHub and pull it down:

Installing OpenCV on your Raspberry Pi Zero

```
$ cd ~
```

```
$ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.0.0.zip
```

```
$ unzip opencv.zip
```

Timing: 1m 58s

Let’s also grab the OpenCV repository as well:

Installing OpenCV on your Raspberry Pi Zero

```
$ wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.0.0.zip
```

```
$ unzip opencv_contrib.zip
```

Timing: 1m 5s

It’s especially important to grab the

`opencv_contrib`

repo if you want access to SIFT and SURF, both of which have been removed from the default install of OpenCV.

Now that

`opencv.zip`

and

`opencv_contrib.zip`

have been expanded, let’s delete them to save space:

Installing OpenCV on your Raspberry Pi Zero

```
$ rm opencv.zip opencv_contrib.zip
```

Step #4: Setup Python

The first step in setting up Python for the OpenCV build is to install `pip` , a Python package manager:

```
$ wget https://bootstrap.pypa.io/get-pip.py
```

```
$ sudo python get-pip.py
```

Timing: 49s

Let’s also install

virtualenv

and

virtualenvwarpper

, allowing us to create separate, isolated Python environments for each of our future projects:

$ sudo pip install virtualenvvirtualenvwrapper

$ sudo rm -rf ~/.cache/pip

Timing: 30s

To complete the install of virtualenv and virtual env wrapper , open up your

~./profile

$ nano ~/.profile

And append the following lines to the bottom of the file:

virtualenv and virtualenvwrapper

export WORKON_HOME=$HOME/.virtualenvs

source /usr/local/bin/virtualenvwrapper.sh

Now,

source

your

~/.profile

file to reload the changes:

$ source ~/.profile

Let’s create a new Python virtual environment appropriately named

cv

:

$ mkvirtualenv cv

Timing: 31s

The only requirement to build Python + OpenCV bindings is to have NumPy installed, so let's use

pip

to install NumPy for us:

$ pip install numpy

Timing: 35m 4s

Step #5: Compile and install OpenCV for the Raspberry Pi Zero

We are now ready to compile and install OpenCV. Make sure you are in the

cv

virtual environment by using the

workon

command:

$ workon cv

And then setup the build using CMake:

$ cd ~/opencv-3.0.0/

$ mkdirbuild

$ cd build

$ cmake -D CMAKE_BUILD_TYPE=RELEASE \

-D CMAKE_INSTALL_PREFIX=/usr/local \

```
-D INSTALL_C_EXAMPLES=ON \
-D INSTALL_PYTHON_EXAMPLES=ON \
-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.0.0/modules \
-D BUILD_EXAMPLES=ON ..
```

Timing: 4m 29s

Now that the build is all setup, run

```
make
```

to start the compilation process (this is going to take awhile, so you might want to let this run overnight):

```
$ make
```

Timing: 9h 42m

Assuming OpenCV compiled without error, you can install it on your Raspberry Pi Zero using:

```
$ sudo make install
$ sudoldconfig
```

Timing: 2m 31s

Step #6: Finishing the install

Provided you completed Step #5 without an error, your OpenCV bindings should now be installed in

```
/usr/local/lib/python2.7/site-packages
```

:

```
$ ls -l /usr/local/lib/python2.7/site-packages
```

```
total 1640
-rw-r--r-- 1 root staff 1677024 Dec 208:34 cv2.so
```

All we need to do now is sym-link the

`cv2.so`

file (which are our actual Python + OpenCV bindings) into the

`site-packages`

directory of the

`cv`

virtual environment:

```
$ cd ~/.virtualenvs/cv/lib/python2.7/site-packages/
$ ln -s /usr/local/lib/python2.7/site-packages/cv2.so cv2.so
```

Step #7: Verifying your OpenCV install

All that's left to do now is verify that OpenCV has been correctly installed on your Raspberry Pi Zero.

Whenever you want to use OpenCV, first make sure you are in the

`cv`

virtual environment:

```
$ workon cv
```

And from there you can fire up a Python shell and import the OpenCV bindings:

```
$ workon cv
$ python
>>> import cv2
>>> cv2.__version__
```

'3.0.0'

```
>>>
```

Or you can execute a Python script that imports OpenCV.

Once OpenCV has been installed, you can remove both the

`opencv-3.0.0`

and

`opencv_contrib-3.0.0`

directories, freeing up a bunch of space on your filesystem:

```
$ rm -rf opencv-3.0.0 opencv_contrib-3.0.0
```

But be cautious before you run this command! Make sure OpenCV has been properly installed on your system before blowing away these directories, otherwise you will have to start the (long, 9+ hour) compile all over again!

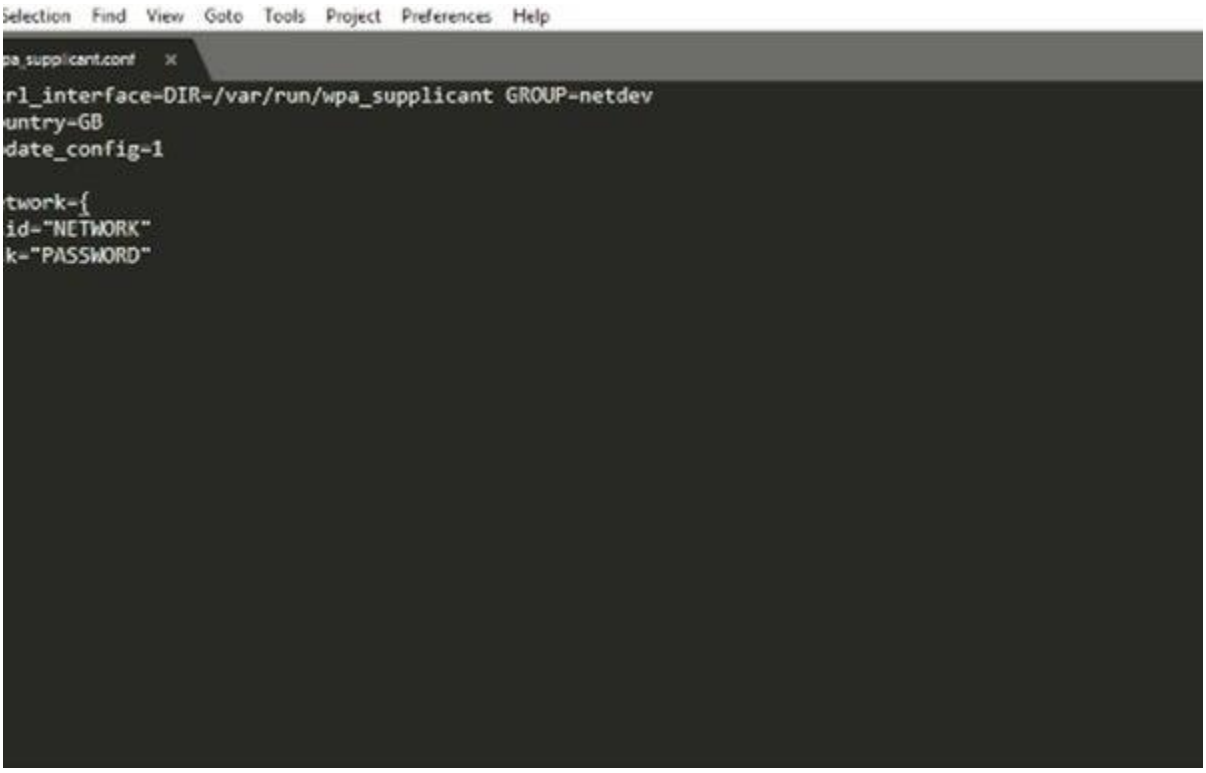
CHAPTER 5

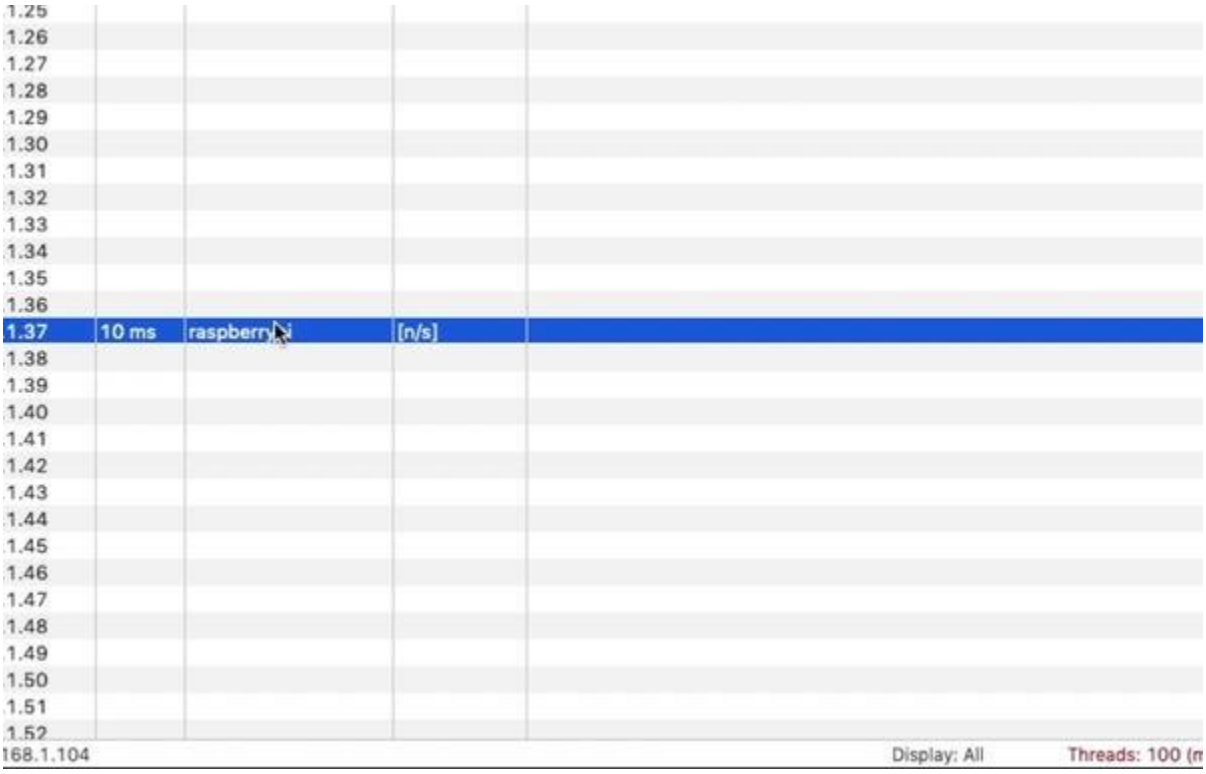
REMOTELY ACCESSING RASPBERRY PI

There are 3 different methods by which you can remotely access the Raspberry Pi to make it a bit easier to work with it. The first one is SSH, which will allow you to remotely access the terminal. The second one is a remote desktop connection, which will allow you to access the Raspberry Pi desktop for times when you need to interact with the UI. The 3rd one will allow you to directly access the files and folders so that you can easily obtain or transfer files between the Raspberry Pi and your computer.

The video above goes into further detail for each one of them and I recommend watching it first to get an understanding of the different methods.

Step 1: Connect to Your Local Network





For all three methods to work, the Raspberry Pi has to be connected to the same network as the computer from which you want to remote access into it. There are several ways in which you can do this.

Wired Connection:

If you plan on using a wired connection then simply plug in the ethernet cable in the board and plug in the other end of the cable into your home router. The board should automatically connect to the network.

Wireless (with display/keyboard/mouse):

If you plan on using a wireless connection then you simply need to connect to the network by clicking the wireless network icon in the taskbar, entering the password and clicking OK. The board should automatically connect to the network. Please watch the video for an example of this.

Wireless (No display, headless mode):

You can also connect the board to the network by creating a wpa_supplicant.conf file in the boot directory of the microSD card. The Raspberry Pi operating system checks this file when it first boots up and if it is present, then it will use the network details contained within it to connect to the network. You can download the template file from the link below and update it with your country code, network name and password. It is recommended to use a text editor

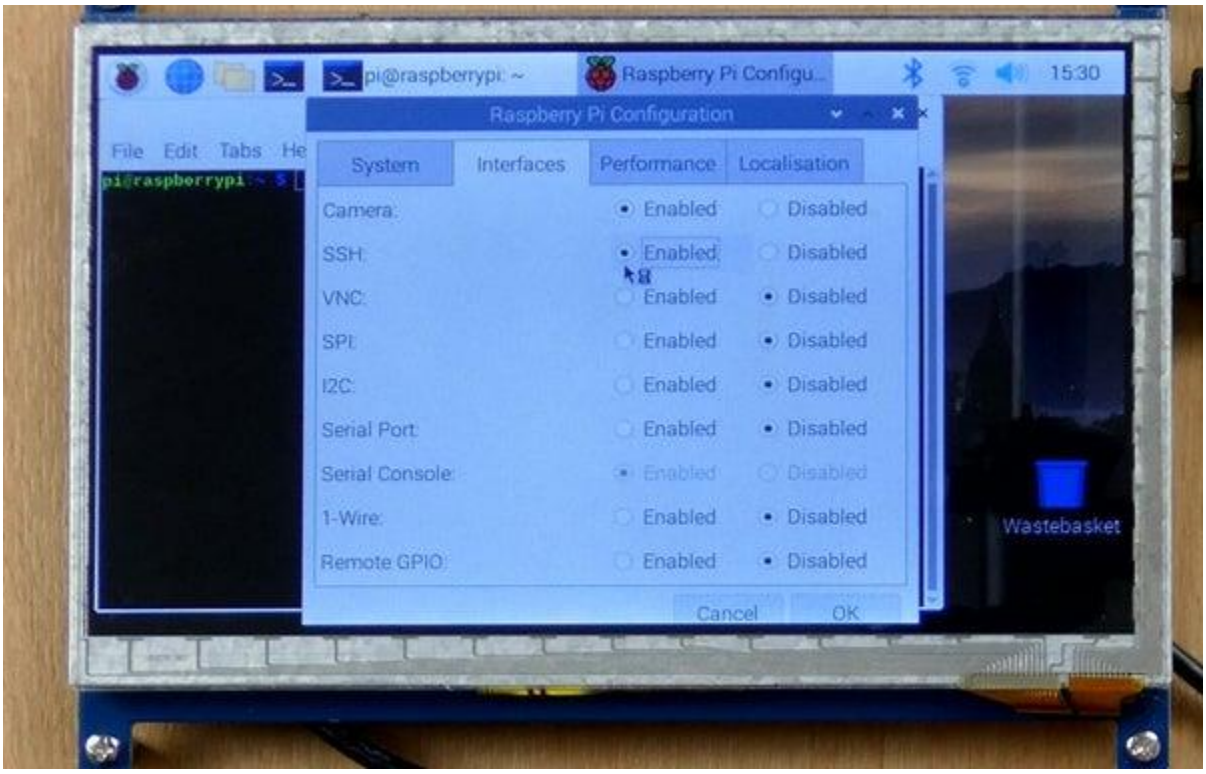
like Notepad++ or Sublime Text 3 to create the file. Once done, simply copy it over to the boot drive as soon as you finish flashing the image, but before you boot the board for the first time.

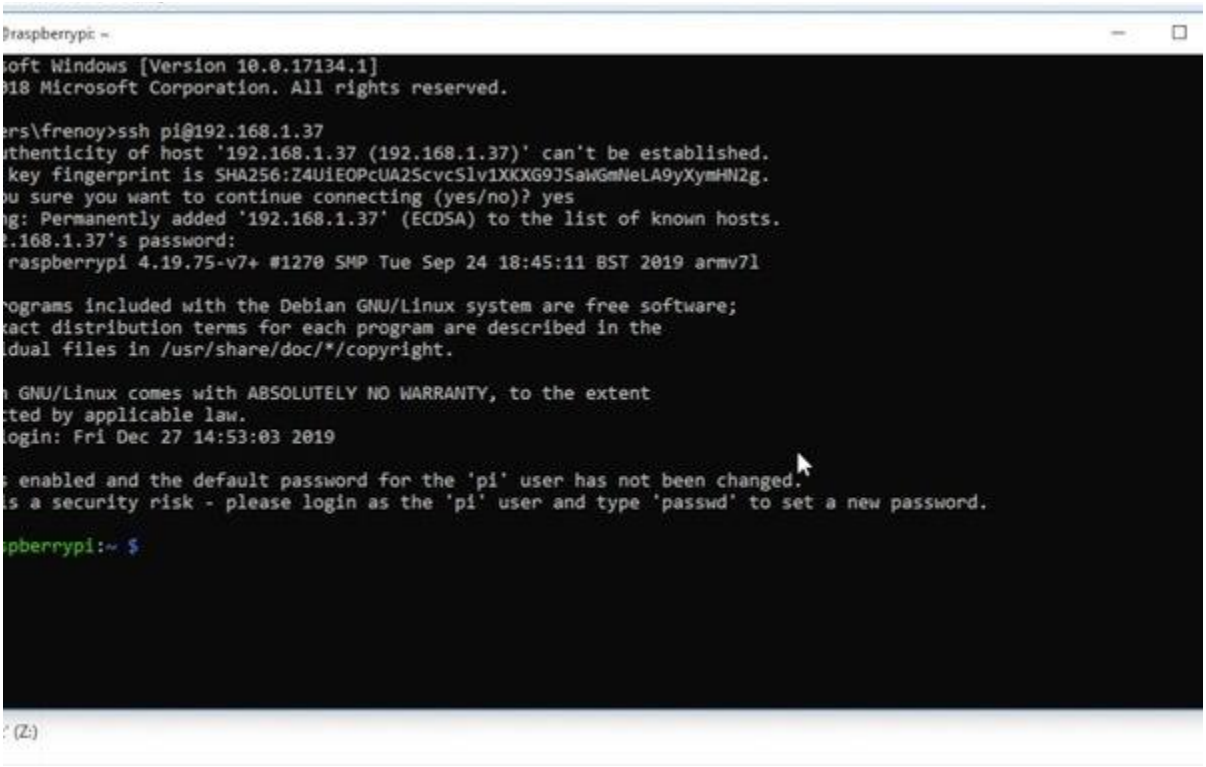
WPA template file: https://www.bitsnblobs.com/wp-content/uploads/2020/05/wpa_supplicant.txt

Use the following link for a list of country codes: https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2

Once connected to the network, we need to obtain the board's IP address. You can do this in many ways, but one convenient method is using software like Angry IP Scanner. Please watch the video if you need to see it in action. The software simply scans your local network and lists all the active devices along with their IP addresses. The IP address for my board is 192.168.1.37 and if your board shows up then you can be sure that it has successfully connected to your network.

Step 2: SSH Into Your Board





If you've used the Raspberry Pi before then you will have used the terminal window at some point. Using the terminal allows you to easily create scripts, update software and so on. By default, you can access and use the terminal by connecting a display and keyboard to the board, but this is not always convenient, particularly if you are using the Pi Zero - which does not have a full-sized USB port nor a full-sized HDMI port. SSH allows you to access the terminal without the need for a display or keyboard, which makes it very convenient. More importantly, you can also copy/paste commands and scripts from your main computer instead of typing everything. SSH stands for Secure SHell and it allows you to securely communicate to a device over an unsecured connection. There are two main ways by which you can enable SSH.

With Display/Keyboard/Mouse:

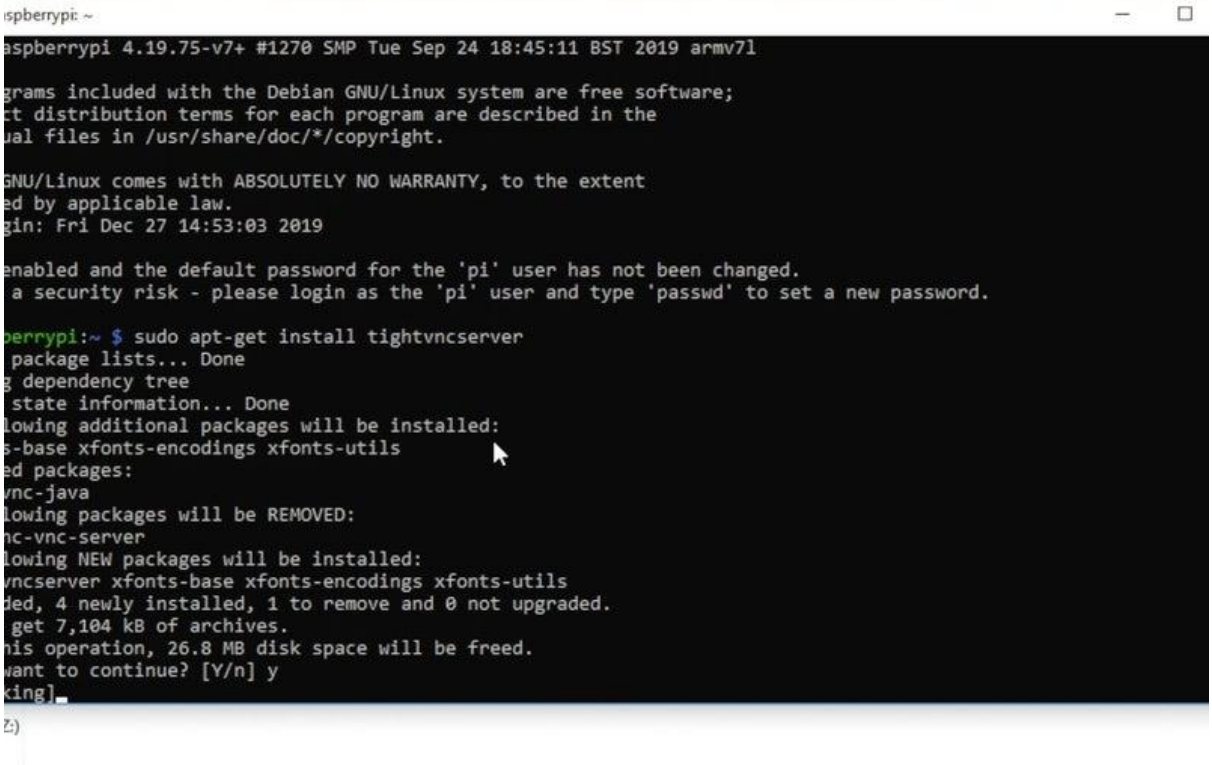
If you have a display, keyboard and mouse connected then you can simply open up the "Raspberry Pi Configuration" window from the "Preferences" menu and then switch to the "Interfaces" tab. You can then simply click the radio button next to SSH which says "Enabled" and then hit OK. You can watch the video to see this being done.

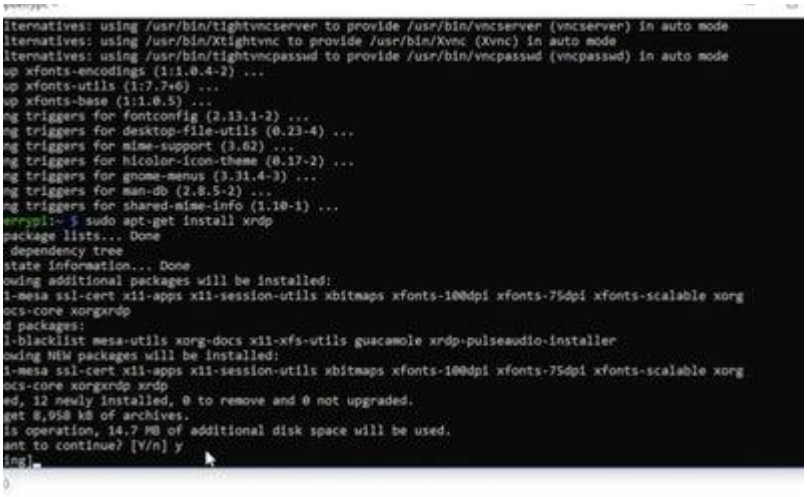
No Display, Headless Mode:

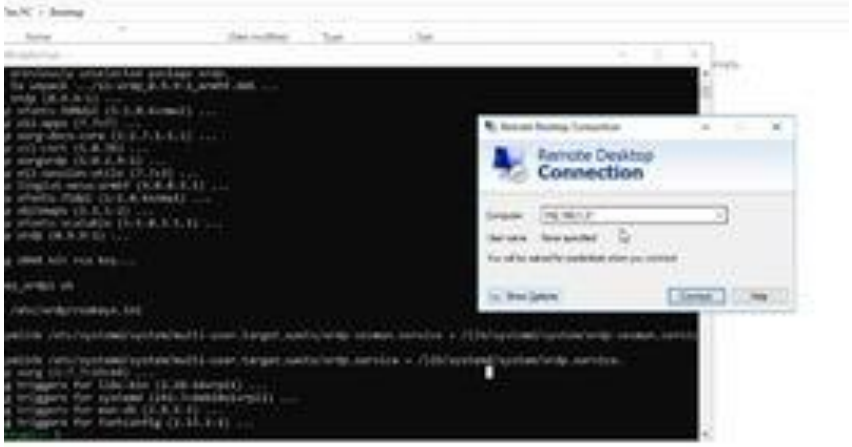
If you don't have access to a display then you can simply create an empty file with the name "ssh" and copy this to the boot drive. Do not add an extension to the file. It is recommended to use a text editor like Notepad++ or Sublime Text 3 to create the file. Copy this file soon after you flash the image but before you boot it for the first time. This will enable SSH for you.

Once done, you simply need to open up a terminal window on your computer (Command Prompt for Windows and Terminal for Mac). Once done, simply type in "ssh pi@192.168.1.37" and hit enter. Please be sure to update your IP address in that command. It will then ask you if you want to remember the host and you can type in yes, followed by pressing the enter key. It will then ask you for a password and by default, this is "raspberry" without the quotation marks. ONce you enter the password, you will log into the board and you can then access the terminal and run commands as if you were directly connected to the board using a keyboard.

Step 3: Remote Desktop Connection







2 More Images

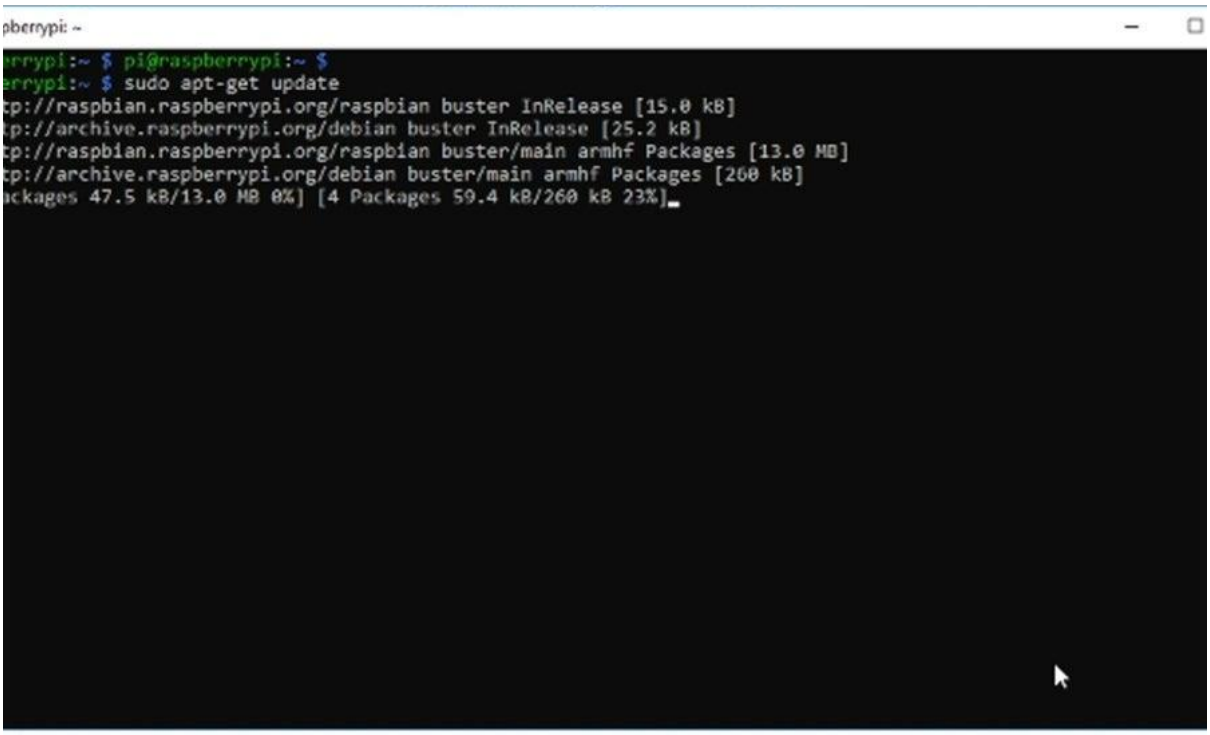
SSH is useful when you want to execute text-based commands and scripts. However, sometimes you will need to access and interact with the graphical user interface or GUI and in times like these, a remote desktop connection is useful. Gaining remote access is not at all difficult. You simply need to run two commands which will install the remote connection server on the Pi.

These commands can be run directly on the Pi terminal or it can be run by first logging into the board using SSH. Once at the terminal, simply type in "sudo apt-get install tightvncserver" and enter y to confirm the install. This will install tightvncserver for us. The next command we need to run is "sudo apt-get install xrdp" and enter y to confirm the install. This will install xrdp which will enable remote access.

All that's needed to be done now is access the desktop. To do this, open up the "Remote Desktop Connection" application on Windows and enter the board's IP address. It will take you to a login screen where you will have to enter the default username which is "pi' and the default password, which is "raspberry". Once done, you will be taken to the Raspberry Pi Desktop where you can interact with the board remotely and do everything as if you are physically connected to the board using a display, keyboard and mouse.

If you are using a Mac, then you will need to first install the "Microsoft Remote Connection" app from the app store. You can then create a new connection by typing in the IP address, username, password and connection name. Finally, double click the connection name to start the connection and you will be taken to the Raspberry Pi Desktop. Please watch the video as we demonstrate this for both a PC and a Mac.

Step 4: Enabling FTP


pberrypi: ~
errypi:~ $ pi@raspberrypi:~ $
errypi:~ $ sudo apt-get update
tp://raspbian.raspberrypi.org/raspbian buster InRelease [15.0 kB]
tp://archive.raspberrypi.org/debian buster InRelease [25.2 kB]
tp://raspbian.raspberrypi.org/raspbian buster/main armhf Packages [13.0 MB]
tp://archive.raspberrypi.org/debian buster/main armhf Packages [260 kB]
ackages 47.5 kB/13.0 MB 0%] [4 Packages 59.4 kB/260 kB 23%]


NOTE: Save file by
Pressing CTRL+X
Then Y
Then ENTER

3 More Images

Using a remote desktop connection is useful but you cannot directly copy files between your PC and the Pi Desktop by using it. Luckily, there is an easy way for us to do this remotely and that is by using FTP or the File Transfer Protocol.

Setting up FTP is simple as well. We simply need to first update the package information by running "sudo apt-get update". Then, we need to run the "sudo apt install proftpd" command which will install the FTP server for us. And that's all you need to do. By default, there is no restriction on the directories which can be accessed by using FTP. If you have multiple users then it would make sense to limit user access to only their directory which is /home/user. You will need to update the configuration file to do this and for that, you will need to run the "sudo nano /etc/proftpd/proftpd.conf" command which will open up the config file in a text editor. Scroll down to the "#DefaultRoot" line and uncomment the "#" which will enable this. Once done, simply save the file by pressing "CTRL+X" then "y", then "ENTER". You will then need to reload the service by running the "sudo service proftpd reload" command. This will put the new configuration into effect and we will only be able to access the /home/pi directory.

Accessing the files and folders is just as easy. You can open up a web browser and type in "ftp://192.168.1.37" and then log in with the default username which is "pi" and default password which is "raspberry". You will then be able to view the files and even download them. It will keep asking you to authenticate the session for security purposes. This is not entirely convenient and the recommended way is by using something called an FTP client like FileZilla. Simply download and install it and then enter the connection details in the top bar, as shown in the image - IP address, username, password, and port which is 21. Once done, click the "Quickconnect" button and you will be able to connect to the board. The Raspberry Pi files and folders will be shown on the right half and the file system of your computer will be on the left. You can drag files across to enable transfers. This way, you can easily retrieve the files you need and manage the file system.

That's how easy it is to remote access your Raspberry Pi using three different methods. If you like helpful posts like this, then please consider subscribing to our YouTube channel as it helps tremendously.

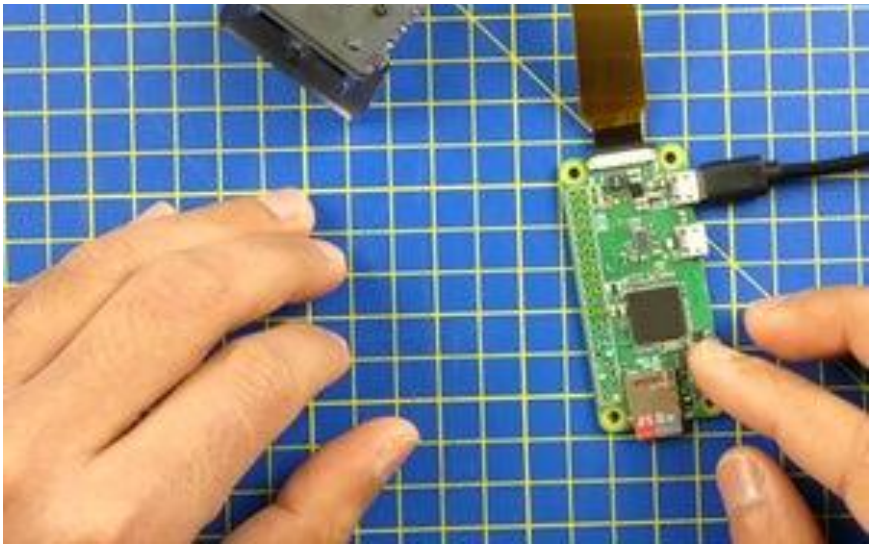
CHAPTER 6

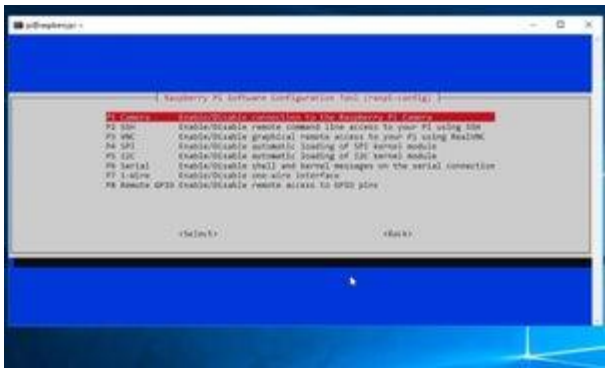
MAKING A DASHCAM USING RASPBERRY PI ZERO

This chapter involves capturing video from the Pi camera and storing it to the microSD card. We also take care of file numbering, detecting remaining storage, stopping the script if it is low and we also add the FTP functionality to access the files remotely.

The video takes a closer look at how everything comes together and we also create some test scripts which help understand the final script. I would strongly recommend watching the video first to get a better understanding of the project.

Step 1: Preparing the Raspberry Pi





The first thing we need to do is download the Raspbian operating system for the Raspberry Pi. I am using the Pi Zero for this project as it is a compact board. I will also not be using the desktop and that's the reason why I downloaded the lite version. You then need to flash this image to a microSD card but before we can boot the Pi with it, we need to copy the wpa_supplicant.conf and ssh files to the boot drive. These files are needed to allow us to control the Raspberry Pi remotely. Another option is using a wired connection and connecting a keyboard/mouse/display to control the Pi. The choice is yours and here's a link to a previous post which explains this in more detail along with remote access and Here's a summary on how to get this configured:

The wpa_supplicant.conf file provides the network details which allows the Pi to connect to your WiFi network - this is needed to install software and also control it for now. A WiFi connection is only needed during the initial setup and we do not need it to record video while in the car. You can use the link below to download a template file for this. Simply update your country code, network name and password and then copy it across to the microSD card. You can use a text editor like notepad++, Sublime Text or Atom to update the file.

WPA template file: https://www.bitsnblobs.com/wp-content/uploads/2020/05/wpa_supplicant.txt

Use the following link for a list of country codes: https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2

We then need to create an empty file with the name ssh using your preferred text editor. Do not add any extension to the file and simply copy it over to the board. Once done, connect the camera, insert the microSD card and then power on the board. The board will take about a minute to connect to your network for the first time.

Before we can proceed, we need to obtain the IP address for the board. One way to do this is by using software called AngryIP scanner. Install and run that and you will then be able to obtain the board's IP address.

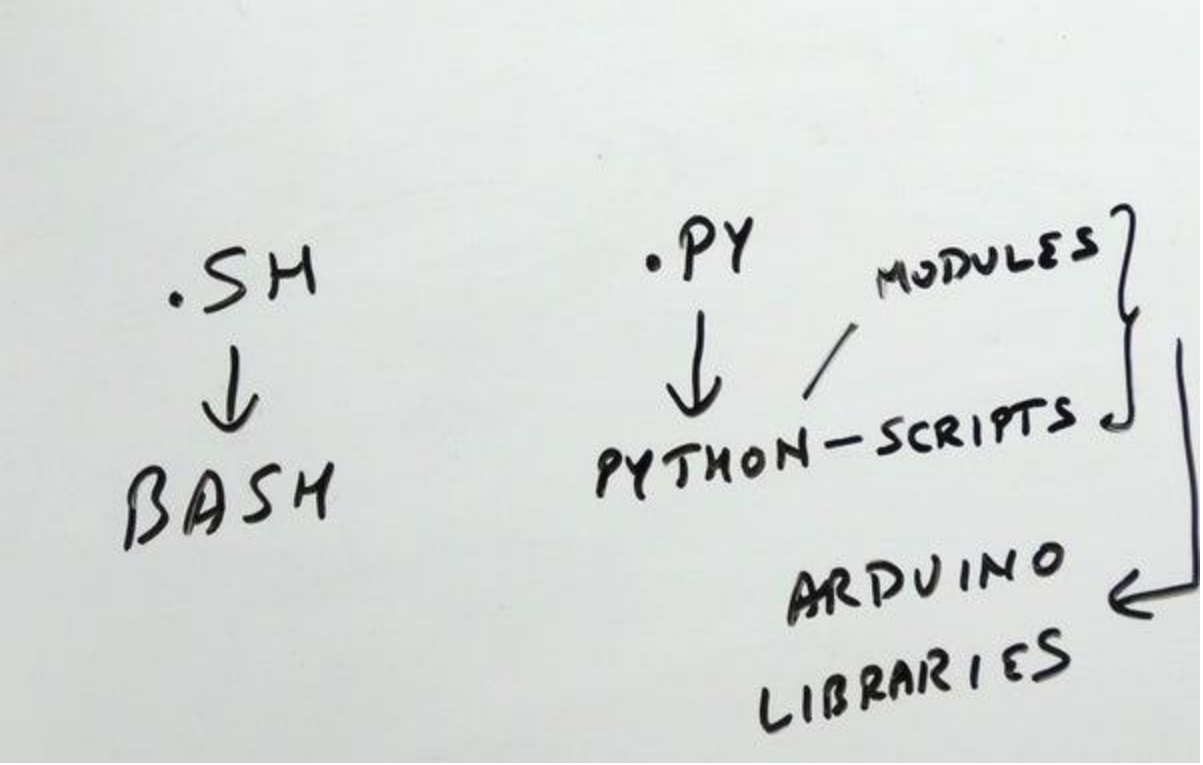
We then need to log into the board by using a terminal (or command prompt in Windows). Simply type in "ssh pi@192.168.1.35" and hit enter. Be sure to use your board's IP address. Type in the default password which is raspberry and you will then have access to the board.

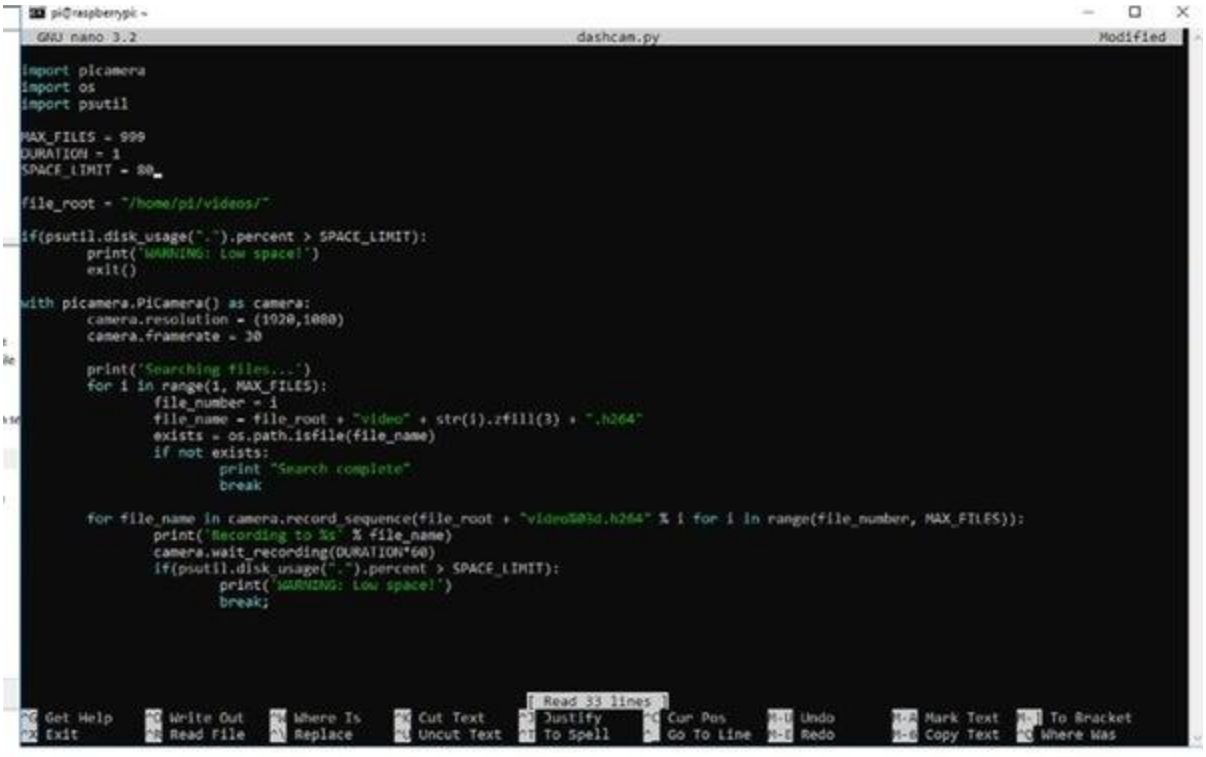
We now need to do a few things before we can start with the script. The first thing is enabling the camera which can be done by running the "sudoraspi-config" command. You will have to navigate to "Interfacing options", then "Camera" and enable it. Use the tab key to select the "Finish" option and then reboot the board. Give the board a minute and then log back into it.

We then need to update the Raspbian operating system and this can be done by running the "sudo apt update && full-upgrade -y" command. Then, install proftpd by running the "sudo apt install proftpd" command. This installs the FTP software that allows us to access the files on the microSD card. You can then open up FileZilla, and connect to the board which will allow you to access the files.

Add TipAskQuestionCommentDownload

Step 2: Create the Script



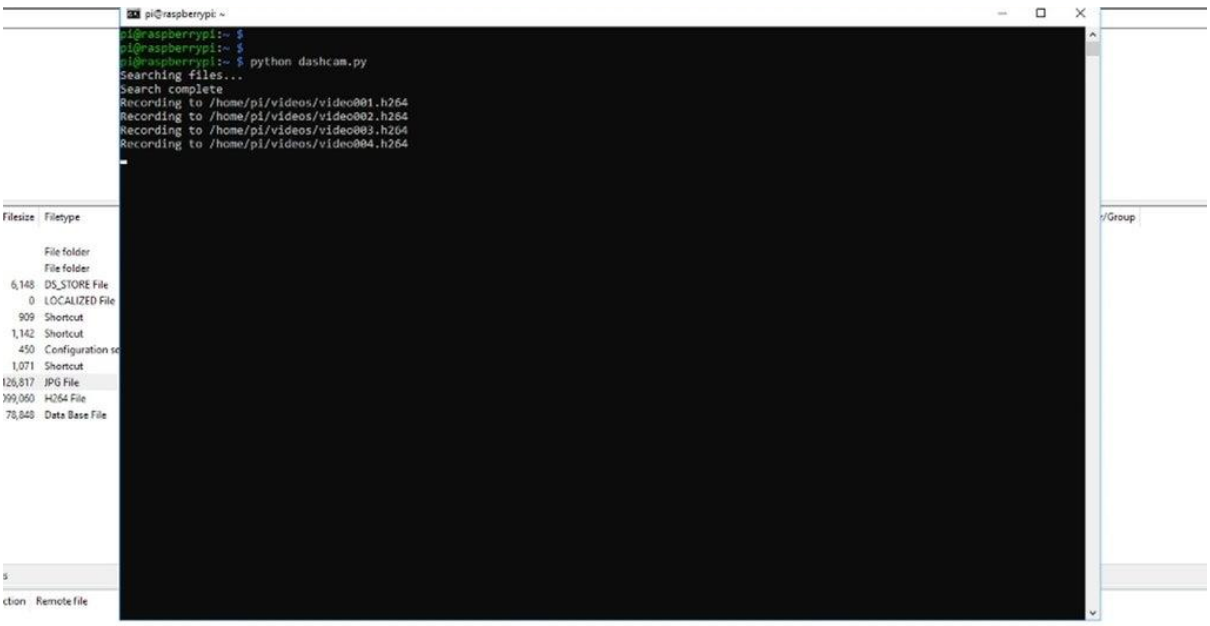


We have previously created script files that had the .sh extension, these were bash scripts. For this project, we will be creating a python script which will have a .py extension. Using python gives us access to a lot of modules, libraries and scripts which makes it easier for us to create projects.

We first need to install the picamera module so that we can use the camera in our scripts. This can be done by running the "sudo apt install python-picamera" command. Once that is completed, simply create a python script by running "sudo nano dashcam.py

You can also copy/paste the contents which is one benefit of using SSH. You can save the file by typing in "CTRL+X", then Y, then ENTER. Please watch the video for a full explanation of how it all works together. We also created two additional scripts before this to get a better understanding.

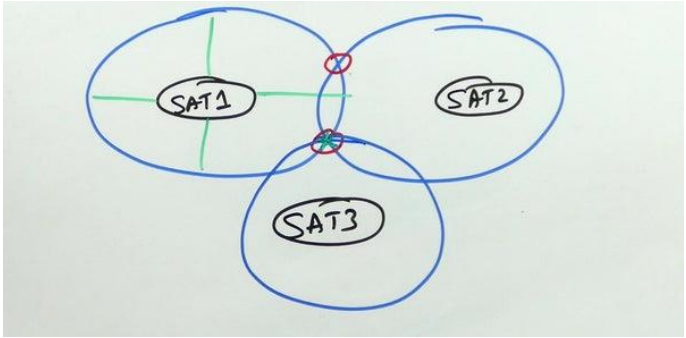
Step 3: Test the Script



Before we can run the script, we need to install the psutil module as we use that to obtain the disk usage. In order to install psutil, we first need to install something called pip which is a package installer for python. This can be done by running the "sudo apt install python-pip" command, followed by the "pip install psutil" command which will install psutil.

Once done, create a new folder by running the "mkdir videos" command and we can then simply run the "python dashcam.py" command which will run the script. You should be able to view the files being created as shown in the image. You can also obtain them using FileZilla and play them back using VLC. You can stop the script by typing "CTRL+C" and it is recommended to then reboot the board.

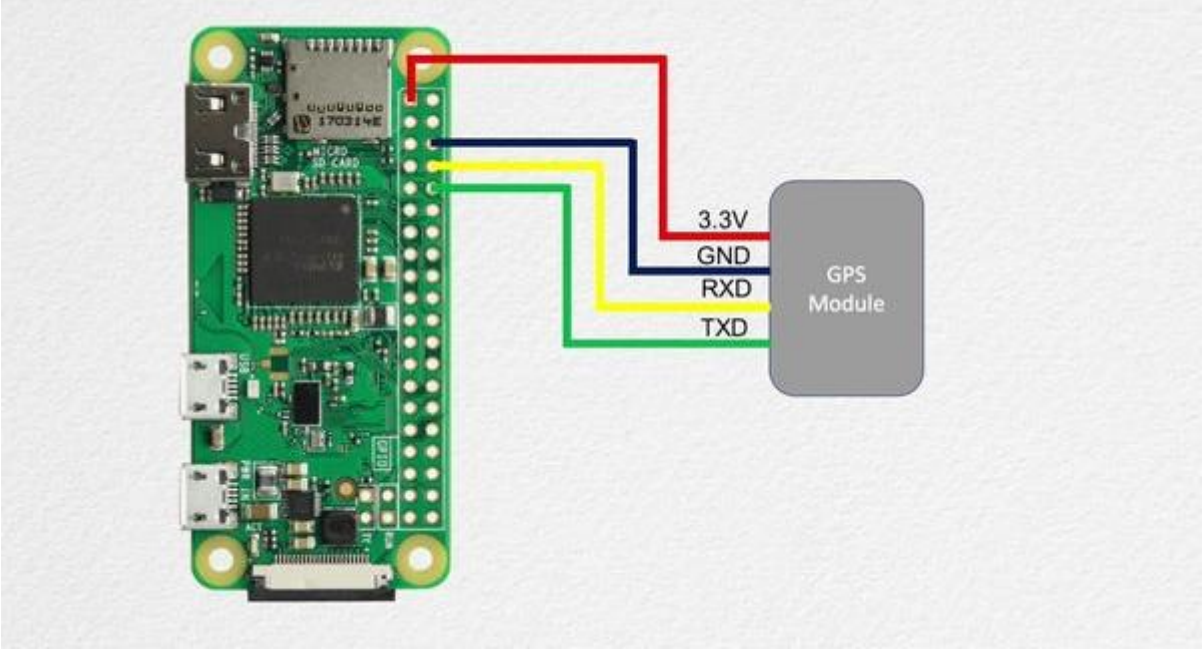
6.2 INTERFACING A GPS MODULE WITH THE RASPBERRY PI: DASHCAM

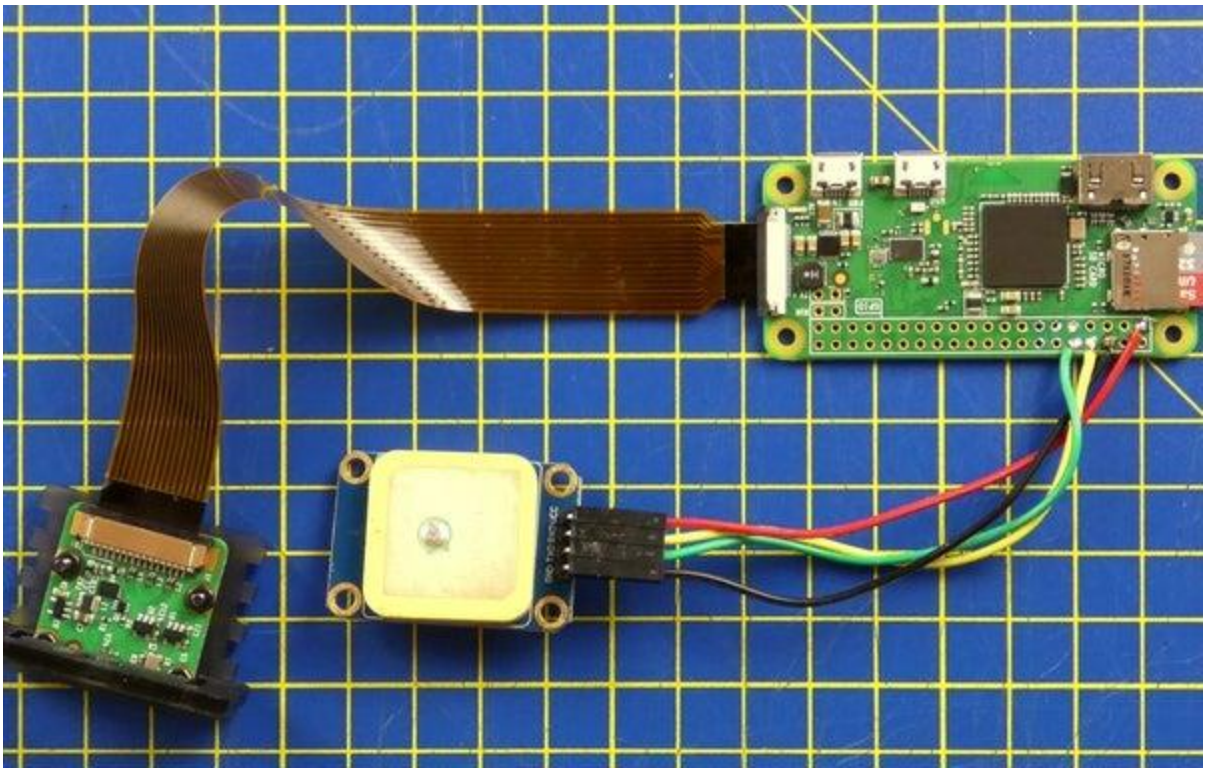


Lets interface a GPS module to the Raspberry Pi. We will then use the GPS data and add it to the video as a text overlay. https://www.instructables.com/id/Making-a-DashCam-Using-the-Raspberry-Pi-Zero-pt1/

The video below gives you more information about GPS modules, communication along with an overview of how everything comes together. I would recommend watching it first before continuing.

Step 1: Enable Serial Port





We first need to SSH into the board and then enable the serial port. This can be done by running the following command:

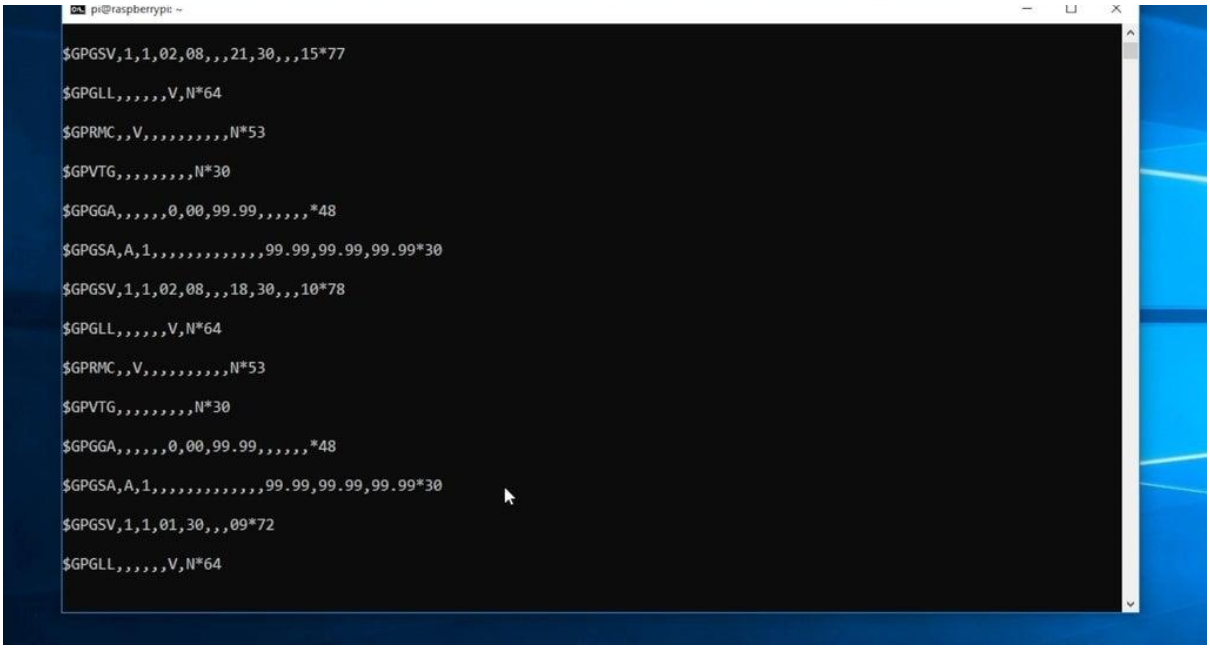
- sudoraspi-config

This will open up the configuration tool and you can use the arrows to navigate to the interfacing options, and then serial. Select NO to the console and then YES to the hardware serial port. Use the TAB key to navigate to the Finish option and then select NO when it asks you if you want to reboot. This will take you back to the terminal. Enter the following command so that we can shut down the board:

- sudo shutdown -h now

Once the board has shut down, we can connect the serial port to the board using the reference image.

Step 2: Test the GPS Module



We will now test the GPS module to make sure it is working correctly. That can be done by running the following command:

- sudo cat /dev/serial0

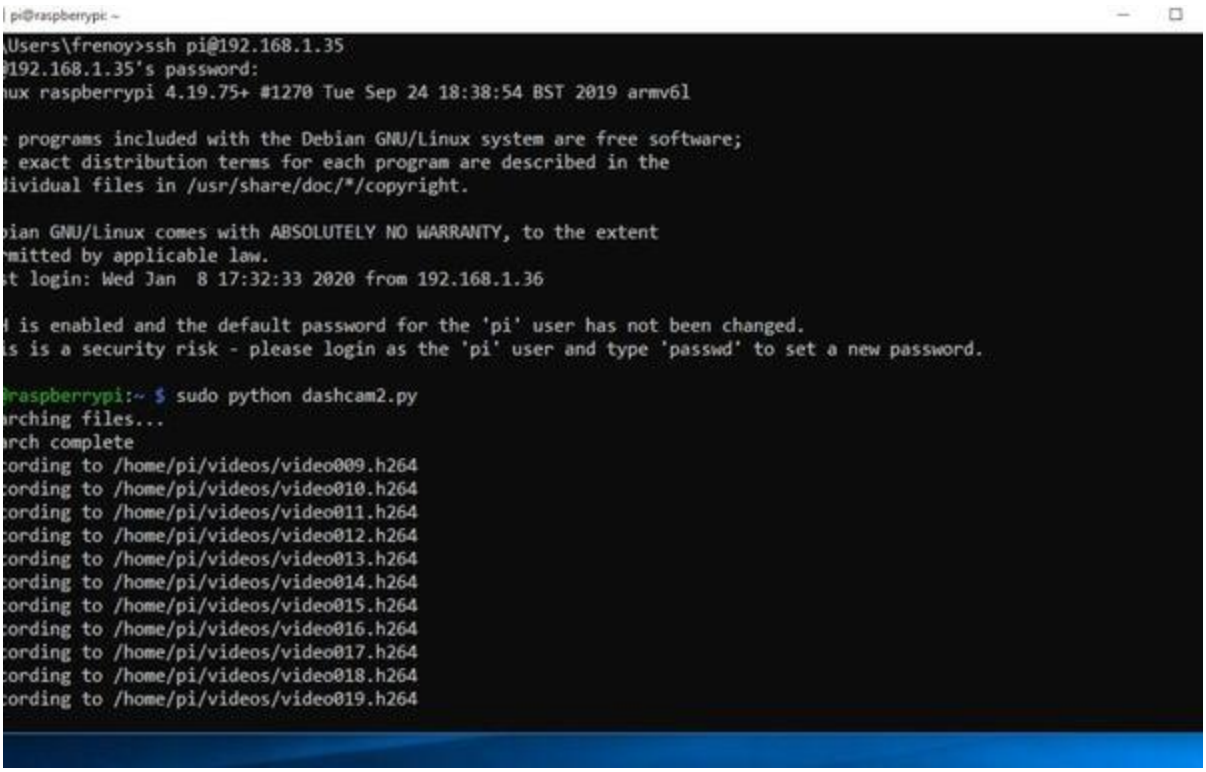
You should be able to see some text output starting with "$GP...." as seen in the image. This is data from the GPS module and it means that the serial communication is working as it should. You can press "CTRL+Z" to stop the output.

We then need to disable the "getty" service as it might interfere with the serial communication. This can be done by running the following commands.

- sudo systemctl stop serial-getty@ttyS0.service

- sudo systemctl disable serial-getty@ttyS0.service

Step 3: Write the Final Script



Before we write the final script, we need to run a few commands. First, we need to install the python-serial module which can be done by running the following command:

- sudo apt install python-serial

We will be using the pynmea2 module to make sense of the GPS data and that can be installed by running the following command:

- sudo pip install pynmea2

We also need to install psutil for the root user and this can be done by running the following command:

- sudo pip install psutil

Finally, we can create a new script file using the following command:

- sudo nano dashcam2.py

You can then copy the contents from the following file and paste it in the script as seen in the video.

https://github.com/bnbe-club/rpi-dashcam-p2-diy-29

Once this is done, save the file by typing "CTRL+X", then Y, then ENTER. You can then test the script by running the following command:

- sudo python dashcam2.py

The script will then run as it should and you can use FileZilla to obtain the video files like last time. The video files will contain an overlay with the GPS data along with the CPU usage.

CHAPTER 7

LICENCE PLATE DETECTION USING HAARCASCADE

License plate recognition is a core module for intelligent transportation systems, while license plate location is an important part of it. Haar-like cascade classifier is good for face detection, but its application to license plate localization largely depends on selection of positive and negative samples. In this paper we studied on how to choose good samples for Haar-like cascade classifiers and image post- processing methods to achieve good location results. It is hoped that the study could be useful to guide sample preparation for other object detection using Haar-like cascade classifiers.

7.1 INTRODUCTION

License plate (LP) has been widely used to validate cars, therefore LP recognition (LPR) is one of the important parts in intelligent transportation, used in traffic monitoring, stolen car detection, portal controlling and so on. As such, many researchers have paid close attention to this field. The main style of Chinese LPs is blue plate. Its width is 440mm and height is 140mm. It usually has 7 characters and a dot between the second and third characters from left. Although the geometry of an LP is stable, the complex environment of image acquisition can greatly change its image appearance. Many problems have to be solved to develop a robust LPR system, some of which are listed below:

- Plate variations: there are more than 5 styles of LPs in China. They have different colors, sizes, characterlayouts, and adornments.
- Environment variations: different illuminations and weather conditions may change the quality of input image greatly and plate-like backgrounds can make localization difficult.
- Image variations: image collecting devices can produce images with different properties, such as different resolutions and different noises.

7.2 METHOD

In the past decades, many methods have been proposed for LP location. In this section, we will briefly review such methods. Approaches for detecting LPs from images can be classified into two categories: image processing based methods and machine learning based methods.

A. Image processing methods

1) Edge based method: One of the significant features in plates is that they always have rich edge information. Techniques based on edge features [1] would have good results if the input images have high quality and do not have complex background like plates.

2) Color based method: Some methods based on color information have been considered [2] as LPs only have a few specified colors. These methods can have good results if illumination is desirable and the quality of images is high. The techniques based on image processing are easy to implement, but they are sensitive to changes in environment.

Therefore a robust LP location algorithm should not be merely based on the techniques described above.

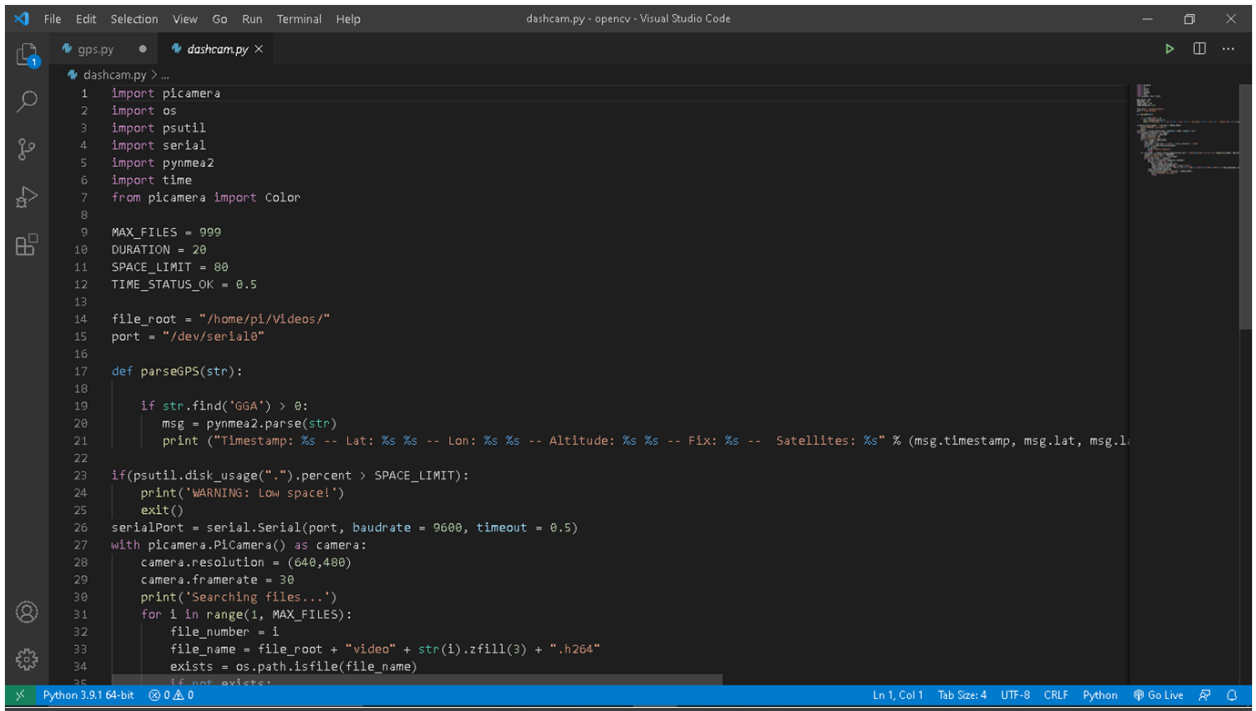
B. Machine learning methods

1) Haar-like feature : the method is originated from face detection and has already been successful in some other object detection fields. Some researchers have already used Haar-like cascade classifiers to detect LPs. In [4], Sun et al compared the performance of Discrete AdaBoost, Real AdaBoost, and Gentle AdaBoost on LP location. In [5], Wu et al used edge information to remove large number of areas in images, then put the rest of areas into classifiers to produce candidate regions. The method showed an encouraging performance but having problems of choosing a good threshold for edge images with different qualities. In [6], Dlagnekov used 1500 images for both training and testing, using enhanced Haar-like features (derivative and variance of the image instead of the original image itself).

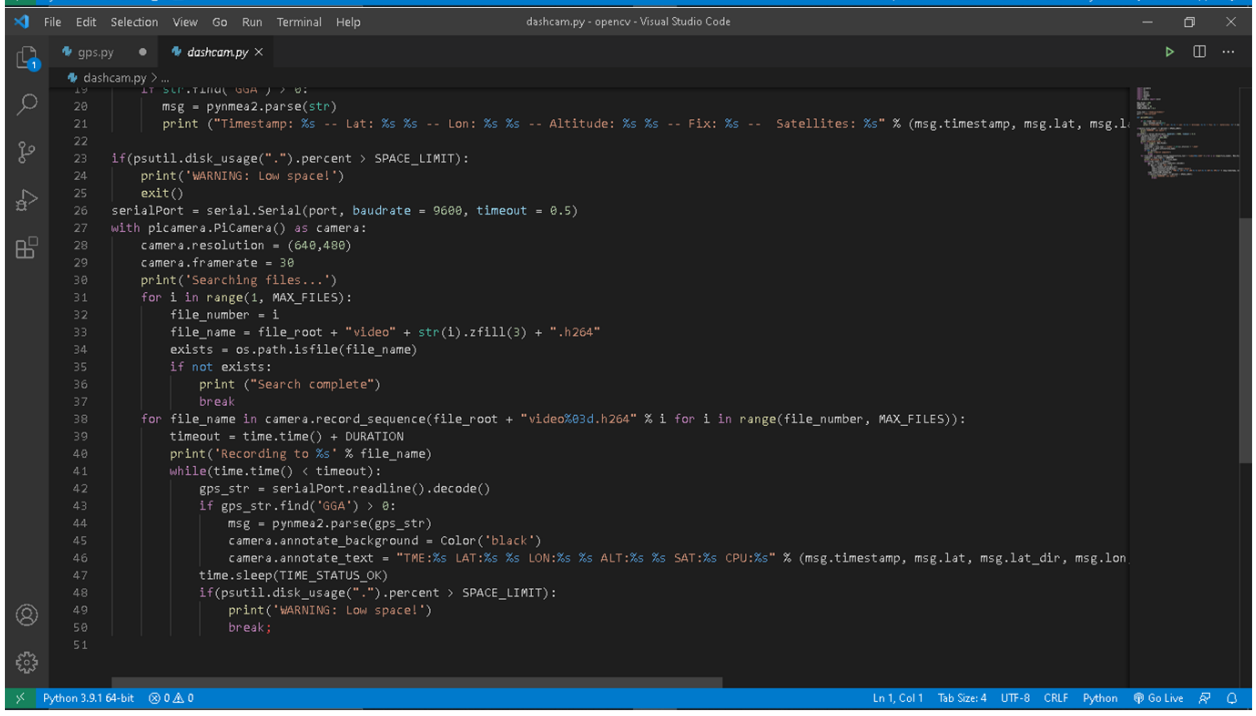
2) Color-texture feature + support vector machine (SVM): in [7] color textural patterns are fed directly to the SVM to derive a classifier for LPs which is then used to generate an image where each pixel represents the possibility of being part of a plate region.

CHAPTER 8

8.1 CODE FOR DASH CAM

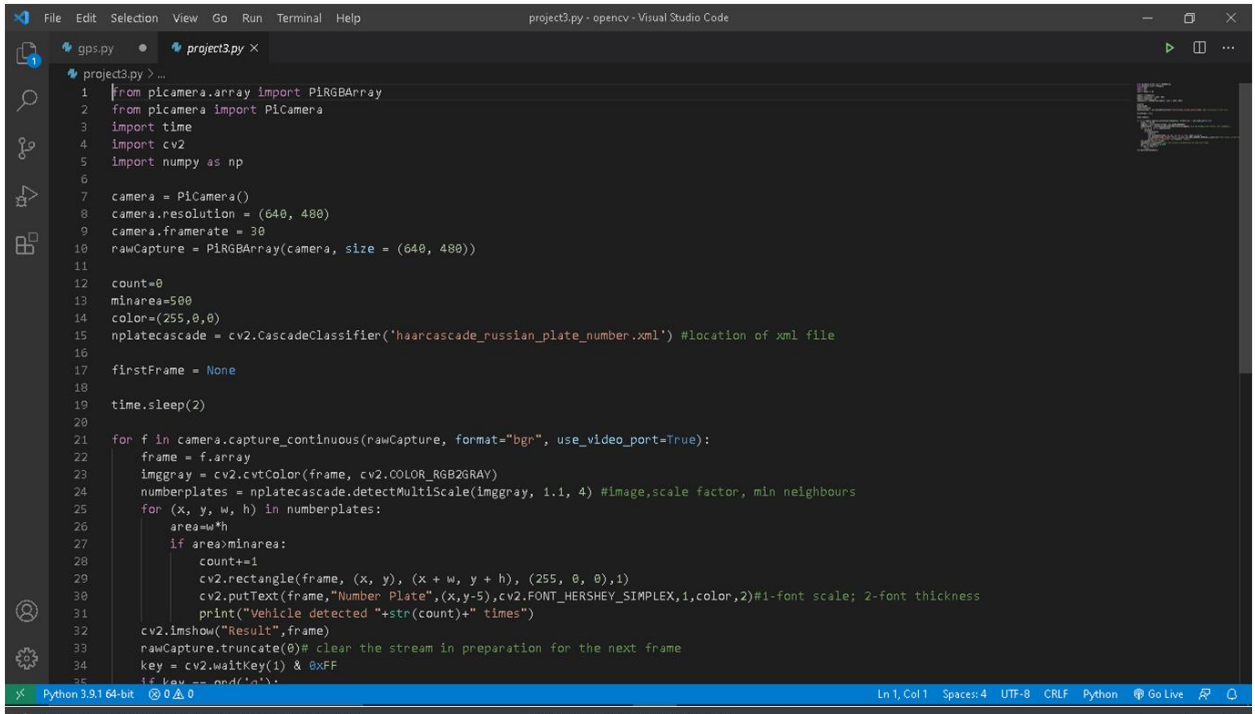


```
import picamera
import os
import psutil
import serial
import pynmea2
import time
from picamera import Color

MAX_FILES = 999
DURATION = 20
SPACE_LIMIT = 80
TIME_STATUS_OK = 0.5

file_root = "/home/pi/Videos/"
port = "/dev/serial0"

def parseGPS(str):

    if str.find('GGA') > 0:
        msg = pynmea2.parse(str)
        print ("Timestamp: %s -- Lat: %s %s -- Lon: %s %s -- Altitude: %s %s -- Fix: %s --  Satellites: %s" % (msg.timestamp, msg.lat, msg.l

if(psutil.disk_usage(".").percent > SPACE_LIMIT):
    print('WARNING: Low space!')
    exit()
serialPort = serial.Serial(port, baudrate = 9600, timeout = 0.5)
with picamera.PiCamera() as camera:
    camera.resolution = (640,480)
    camera.framerate = 30
    print('Searching files...')
    for i in range(1, MAX_FILES):
        file_number = i
        file_name = file_root + "video" + str(i).zfill(3) + ".h264"
        exists = os.path.isfile(file_name)
```

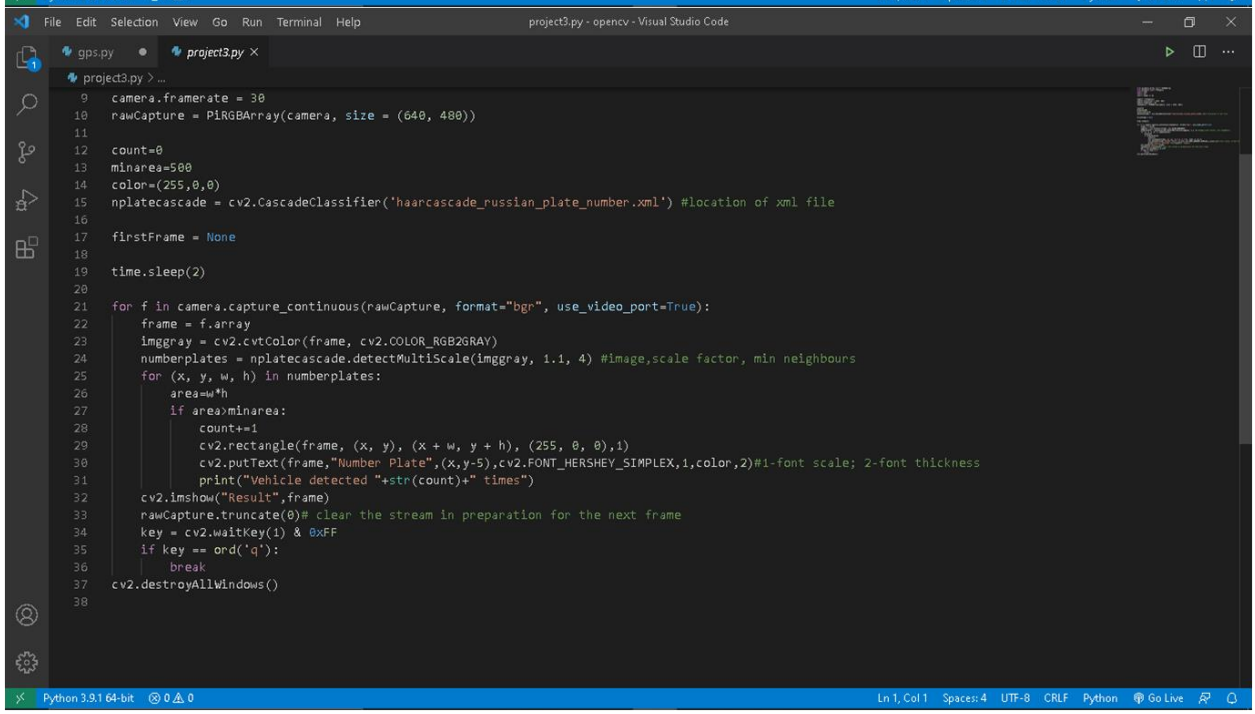


```
        msg = pynmea2.parse(str)
        print ("Timestamp: %s -- Lat: %s %s -- Lon: %s %s -- Altitude: %s %s -- Fix: %s --  Satellites: %s" % (msg.timestamp, msg.lat, msg.l

if(psutil.disk_usage(".").percent > SPACE_LIMIT):
    print('WARNING: Low space!')
    exit()
serialPort = serial.Serial(port, baudrate = 9600, timeout = 0.5)
with picamera.PiCamera() as camera:
    camera.resolution = (640,480)
    camera.framerate = 30
    print('Searching files...')
    for i in range(1, MAX_FILES):
        file_number = i
        file_name = file_root + "video" + str(i).zfill(3) + ".h264"
        exists = os.path.isfile(file_name)
        if not exists:
            print ("Search complete")
            break
    for file_name in camera.record_sequence(file_root + "video%03d.h264" % i for i in range(file_number, MAX_FILES)):
        timeout = time.time() + DURATION
        print('Recording to %s' % file_name)
        while(time.time() < timeout):
            gps_str = serialPort.readline().decode()
            if gps_str.find('GGA') > 0:
                msg = pynmea2.parse(gps_str)
                camera.annotate_background = Color('black')
                camera.annotate_text = "TME:%s LAT:%s %s LON:%s %s ALT:%s %s SAT:%s CPU:%s" % (msg.timestamp, msg.lat, msg.lat_dir, msg.lon
            time.sleep(TIME_STATUS_OK)
            if(psutil.disk_usage(".").percent > SPACE_LIMIT):
                print('WARNING: Low space!')
                break;
```

8.2 CODE FOR AUTOMATIC BEAM CONTROLLER




```python
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
import numpy as np

camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 30
rawCapture = PiRGBArray(camera, size = (640, 480))

count=0
minarea=500
color=(255,0,0)
nplatecascade = cv2.CascadeClassifier('haarcascade_russian_plate_number.xml') #location of xml file

firstFrame = None

time.sleep(2)

for f in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    frame = f.array
    imggray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
    numberplates = nplatecascade.detectMultiScale(imggray, 1.1, 4) #image,scale factor, min neighbours
    for (x, y, w, h) in numberplates:
        area=w*h
        if area>minarea:
            count+=1
            cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0),1)
            cv2.putText(frame,"Number Plate",(x,y-5),cv2.FONT_HERSHEY_SIMPLEX,1,color,2)#1-font scale; 2-font thickness
            print("Vehicle detected "+str(count)+" times")
    cv2.imshow("Result",frame)
    rawCapture.truncate(0)# clear the stream in preparation for the next frame
    key = cv2.waitKey(1) & 0xFF
```




```python
camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 30
rawCapture = PiRGBArray(camera, size = (640, 480))

count=0
minarea=500
color=(255,0,0)
nplatecascade = cv2.CascadeClassifier('haarcascade_russian_plate_number.xml') #location of xml file

firstFrame = None

time.sleep(2)

for f in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    frame = f.array
    imggray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
    numberplates = nplatecascade.detectMultiScale(imggray, 1.1, 4) #image,scale factor, min neighbours
    for (x, y, w, h) in numberplates:
        area=w*h
        if area>minarea:
            count+=1
            cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0),1)
            cv2.putText(frame,"Number Plate",(x,y-5),cv2.FONT_HERSHEY_SIMPLEX,1,color,2)#1-font scale; 2-font thickness
            print("Vehicle detected "+str(count)+" times")
    cv2.imshow("Result",frame)
    rawCapture.truncate(0)# clear the stream in preparation for the next frame
    key = cv2.waitKey(1) & 0xFF
    if key == ord('q'):
        break
cv2.destroyAllWindows()
```

8.3 CONCLUSION

Above system would help in automatically converting the high beam to low beam when required which could decrease accidents mainly in hilly areas and also record footage in emergency along with position and system parameters.

8.4 REFERENCES:-

1. Raspberry Pi Blog by Eben Upton, 26th Nov 2015

https://www.raspberrypi.org/blog/raspberry-pi-zero/

2. Raspberry Pi Camera Module:
https://www.raspberrypi.org/products/camera-module-v2/

3. Dash cam Witness: Video sharing motives and privacy concerns across different nations by joohyun kim sangkeun park and uichin lee
https://www.researchgate.net/publication/342139954_Dashcam_Witness_Video_Sharing_Motives_and_Privacy_Concerns_Across_Different_Nations

4. Automatic Light Beam Controller for driver assistance September 2011 by P. F. Alcantarilla·L. M. Bergasa·P. Jiménez·I. Parra·D. F. Llorca·M. A. Sotelo·S. S. Mayoral

https://www.researchgate.net/publication/225715555_Automatic_LightBeam_Controller_for_driver_assistance

5. Vehicle Detection ASSIST

https://www.murtazahassan.com/courses/learn-opencv-in-3-hours/

6. Dash camera ASSIST
https://www.instructables.com/Making-a-DashCam-Using-the-Raspberry-Pi-Zero-pt1/

7. Interfacing GPS module to Dash camera ASSIST
https://www.instructables.com/Interfacing-a-GPS-Module-With-the-Raspberry-Pi-Das/

8. NEO6MV2 GPS Module:
https://www.engineersgarage.com/microcontroller-projects/articles-raspberry-pi-neo-6m-gps-module-interfacing/