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

Analysis of Vehicle Activities and Live Streaming using IOT

Kandimalla Sandeep Kumar[1], S.Mahaboob Basha[2]
[1]P.G. Scholar, [2]Guide, Head of the Department
[1,2] Branch: Embedded systems, ECE, M.Tech
[1.2] Electronics and Communication Engineering Department
[1,2] Geethanjali College of Engineering & Technology, Kurnool
Email:[1] sandeep96424@gmail.com, [2]syedmahaboob45@gmail.com

Abstract

This paper presents analysis of vehicle activities and live streaming that helps to analysis the cause of vehicular accidents. The aim is to analysis accidents by tracking what occurs in vehicles. Use sensors to record the varied driving information like Alcohol sensor, vibration sensor. The system use external sensor such as GPS to collect location and display on maps. Here we can analyze the vehicle live position by using its latitude and longitude. DC motor to control the vehicle moments. we observe the activities of vehicle, such as moving or not. The proposed system has ability to control live video streaming over the internet local host.

Keywords: Raspberry Pi Zero device, Gps, Python3 software.

Introduction to Embedded Systems

An embedded system is a combination of computer hardware and software, fixed in capability or programmable, designed for a specific function or functions within a larger system. Industrial machines, agricultural and process industry devices, automobiles, medical equipment, cameras, household appliances, airplanes, vending machines and toys, as well as mobile devices, are possible locations for an embedded system.

Application Areas

Nearly 99 per cent of the processors manufactured end up in embedded systems. The embedded system market is one of the highest growth areas as these systems are used in very market segment- consumer electronics, office automation, industrial automation, biomedical engineering..etc

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

Industrial automation: Today a lot of industries use embedded systems for process control. These include pharmaceutical, cement, sugar, oil exploration, nuclear energy, electricity generation and transmission. The embedded systems for industrial use are designed to carry out specific tasks such as monitoring the temperature, pressure, humidity, voltage, current etc., and then take appropriate action based on the monitored levels to control other devices or to send information to a centralized monitoring station. In hazardous industrial

environment, where human presence has to be avoided, robots are used, which are programmed to do specific jobs. The robots are now becoming very powerful and carry out many interesting and complicated tasks such as hardware assembly.

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

Introduction to IoT:

The Internet of Things (IoT) has been envisioned as the next wave in the era of cyber technology, in which millions of smart devices (including various sensors and actuators) are wirelessly connected and integrated via the Internet. This emerging paradigm will fundamentally create and boost a number of new applications across many fields, including environmental monitoring, precision agriculture, smart grids, smart cities and e-health systems. It is of critical importance to provide the next generation of computer scientists and engineers an opportunity to not only understand the concepts and principles of IoT, but also to study the practical development of IoT solidly, so that students can learn how to apply theories to real applications. Many universities have started to introduce IoT courses into their undergraduate curriculum.

Real-world projects demand real resources. For example, wireless communication-enabled tiny computers are essential for students to program and experiment with in any IoT project. The invention of the Raspberry Pi, an inexpensive, tiny and relatively powerful computer board, not only provides a great building block to facilitate research and various IoT application developments, but also provides a desirable hardware platform for the project-based learning paradigm in computer science and engineering education. Educators have exploited Raspberry Pi either as a single device or as the basis of more sophisticated learning systems for their IoT education.

Proposed system:

The main aim in this system is to minimize the accident made by the vehicle, control the activities and prevent from the accident. In this system vibration sensor, MEMS sensor, alcohol sensor and camera are to be used. The accident is recorded and stored, after the accident happened the video can be retrieved and analysis accident. The live streaming can be done in this system and accident can be watched. The sensors are detecting the vehicle activities and give the alert message if any part in vehicle is not work properly.

Literature Survey

Vehicle Tracking System Global Positioning System (GPS) is most widely used for the tracking system. The most common ways to track vehicle is the Global Positioning System (GPS) and Global system for mobile communication or General Packet Radio Service (GSM/GPRS)

technology. The real time vehicle tracking system, used for many applications including vehicle security and fleet management. School bus tracking and monitoring system in which GPS is used to track the school bus which provides an accurate arrival time of the vehicle to a particular location or stop. Hence, the student can spend time for other activities rather than waiting for a school bus. The use of GSM and GPS technology together to allow the system to track the vehicles which provide most up-to-date information on the ongoing trips. The locations are reported by SMS message which takes an advantage of wireless technology in providing powerful transportation management engine. An algorithm for bus arrival system for individual stops along service route with GPS technology was developed. The algorithm is implemented in an intelligent system that can automatically detect the running route and direction of a bus and predict its arrival times at the downstream stops under any operating conditions. A system is developed which integrates GSM and Google map. The GSM modem at the control centre receives an SMS of the coordinates. This will update the main database and the position of the vehicle is displayed through Google map. Tracking of vehicle has been made simpler with the advent of GPS technology. Real time tracking of many vehicles on cloud platform was developed and it makes easy to track many vehicles at a time using the GPS and GSM. GPS positioning technology plays an important role in positioning, monitoring, and navigation. The combination of GPS and GSM technology enables the user to track/locate the vehicle in ease and convenient manner.

HARDWARE IMPLEMENTATION OF THE PROJECT

This chapter briefly explains about the Hardware Implementation of the project. It discusses the design and working of the design with the help of block diagram and circuit diagram and explanation of circuit diagram in detail. It explains the features, timer programming, serial communication, interrupts of atmega328 microcontroller. It also explains the various modules used in this project.

Design

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

- Hardware implementation
- Firmware implementation

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

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

Introduction to Microcontroller

Based on the Processor side Embedded Systems is mainly divided into 3 types

1. Micro Processor: - are for general purpose Eg: our personal computer

2. Micro Controller:- are for specific applications, because of cheaper cost we will go for these

3. DSP (Digital Signal Processor):- are for high and sensitive application purpose

Microcontroller versus Microprocessor

A system designer using a general-purpose microprocessor such as the Pentium or the 68040 must add RAM, ROM, I/O ports, and timers externally to make them functional. Although the addition of external RAM, ROM, and I/O ports makes these systems bulkier and much more expensive, they have the advantage of versatility such that the designer can decide on the amount of

RAM, ROM and I/O ports needed to fit the task at hand.

A Microcontroller has a CPU (a microprocessor) in addition to a fixed amount of RAM, ROM, I/O ports, and a timer all on a single chip. In other words, the processor, the RAM, ROM, I/O ports and the timer are all embedded together on one chip; therefore, the designer cannot add any external memory, I/O ports, or timer to it. The fixed amount of on-chip ROM, RAM, and number of I/O ports in Microcontrollers makes them ideal for many applications in which cost and space are critical.

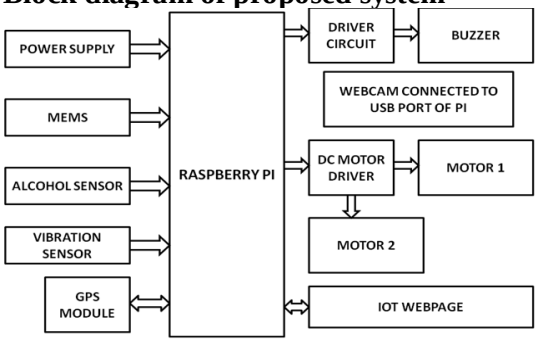
Microprocessor vs. Microcontroller	
Microprocessor	Microcontroller
CPU is stand alone RAM, RAM, I/O, timer are separate	CPU,RAM,ROM,I/O and timer are all on a single chip
Designer can decide on the amount of ROM, RAM and I/O ports.	Fix amount of on chip ROM,RAM, I/O Ports.
Expansive, Versatility	For applications in which cost, power and space are critical
General purpose	Single purpose

Table: Microprocessor versus Microcontroller

Block Diagram of the Project

The block diagram of the design is as shown in the figure below. The brief description about block diagram is given below.

Block diagram of proposed system



RASPBERRY PI

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

The Raspberry Pi is a low cost, **credit-card sized computer** that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

What's more, the Raspberry Pi has the ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras. We want to see the Raspberry Pi being used by kids all over the world to learn to program and understand how computers work.

Raspberry Pi Zero:

The Pi Zero is the smallest, most affordable Pi ever. Raspberry Pi Zero W is the Pi, but at a largely reduced size of only 65mm long by 30mm wide and at a very economical price. With the addition of wireless LAN and Bluetooth, the Raspberry Pi Zero W is ideal for making the embedded Internet of Things (IoT) projects. The Pi Zero W has been designed to be as flexible and compact as possible with mini connectors and an unpopulated 40-pin GPIO. The heart of the Raspberry Pi Zero W is a 1GHz BCM2835 single-core processor, the same as the B+ and A+, with 512MB RAM. Raspberry Pi Zero W is about four times faster than the

original Raspberry Pi and is only a fraction of the cost of the current RPi3.

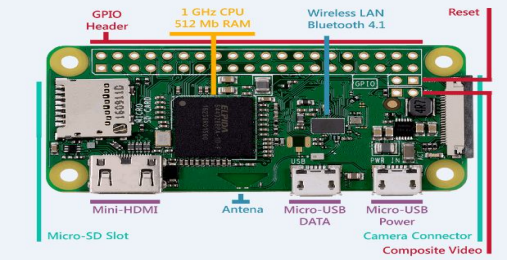
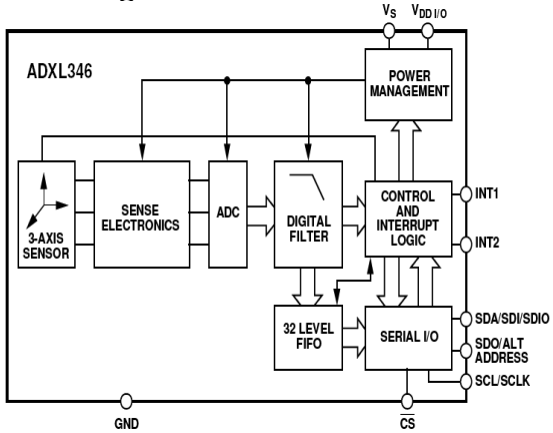


Fig: Raspberry Pi Zero

The Raspberry Pi Zero Case has been designed to fit both the Pi Zero and the Pi Zero W. These official cases consist of several parts and protect the Raspberry Pi Zero from things like rogue wires that might short it out while still allowing full access to the board. Simply snap the RPi into the bottom half of the enclosure, then snap on the desired top. No tools required.

Each case has a standard base featuring a cut-out to allow access to the GPIO, and a choice of three lids: a standard lid, a GPIO lid (allowing access to the GPIO pins from above) and a camera lid (which, when used with the short camera cable supplied, allows the Raspberry Pi camera to be fitted neatly inside it). Additionally, each case includes a 4cm long CSI cable and four rubber feet.

Block Diagram of ADXL346



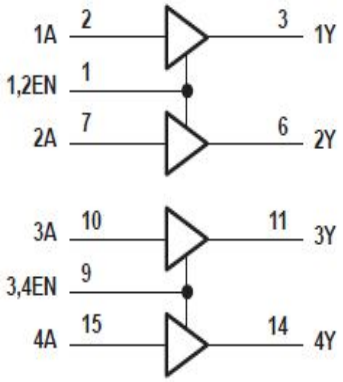
The ADXL345 is a small, thin, low power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 g. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I2 C digital interface. The ADXL345 is well suited for mobile device applications. It measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock.

FUNCTION TABLE
(each driver)

INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high level, L = low level, X = irrelevant,
Z = high impedance (off)
† In the thermal shutdown mode, the output is in the high-impedance state, regardless of the input levels.

Logic Diagram of Motor



This chip contains 4 enable pins. Each enable pin corresponds to 2 inputs. Based on the input values given, the device connected to this IC works accordingly.

GPS Technology:

The Global Positioning System (GPS) is a satellite based navigation system that sends and receives radio signals. A GPS receiver acquires these signals and provides the user with information. Using GPS technology, one can determine location, velocity and time, 24 hours a day, in any weather conditions anywhere in the world for free.

GPS was formally known as the NAVSTAR (Navigation Satellite Timing and Ranging). Global Positioning System was originally developed for military. Because of its popular navigation capabilities and because GPS technology can be accessed using small, inexpensive equipment, the government made the system available for civilian use. The USA owns GPS technology and the Department of Defense maintains it.

A Brief History of Navigation Systems

Navigation can be defined as going from one place to another while logging your position periodically and as necessary. In older times, our ancestors looked at the sky and made many calculations to determine their location on earth. For hundreds of years, seamen used celestial objects and this continued until the 1940s. Then, various navigation systems emerged like DECCA, LORAN and OMEGA. However, all these required special charts and the positions calculated were not pinpoint positions in many cases.

The GPS concept originated during the race to space between Russia and the United States. U.S. scientists realized that they could monitor Sputnik's transmissions and determine its position in the sky by measuring the Doppler distortion of the signal's frequency between the satellite and their known position on earth. They realized that the converse would also be true that if the satellites position was known then they could determine a particular location on earth. GPS satellites were first launched over 20 years ago in 1978, paid for by the American taxpayer. However, it was not until 1993, when a full constellation of 24 satellites were deployed, that it was considered fully operational. Early commercial applications in 1984 were ascertaining position fixes on offshore oil rigs, and surveying, when GPS equipment was very expensive ($150K) as well as large and unwieldy.

Handheld units arrived on the scene in 1989 and their purchase price was $3,000, still not in the price range for casual hobbyist users. However, by 1995 this barrier was crossed when handhelds came down to $200 a unit, making it feasible for hunters, fishermen and hikers. Now GPS is integrated into the cell phones, even though in most cases it has not been activated by the service provider. CDMA cell sites use GPS for network synchronization. As a result of the current price point of GPS receivers and increasing accuracy, GPS is showing up in many new personal and business applications.

For national security reasons, the civilian signalled was originally deliberately injected with an error factor, referred to as **selective availability** (SA). In May 2000, the government turned off SA. It should be noted that the military can jam GPS signals over a particular geographic region if necessary for national security purposes.

FIRMWARE IMPLEMENTATION OF THE PROJECT DESIGN

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

How to Install Raspbian on The Raspberry Pi

Installing Raspbian on the Raspberry Pi is pretty straightforward. We'll be downloading Raspbian and writing the disc image to a microSD card, then booting the Raspberry Pi to that microSD card. For this project, you'll need a microSD card (go with at least 8 GB), a computer with a slot for it, and, of course, a Raspberry Pi and basic peripherals (a mouse, keyboard, screen, and power source). This isn't the

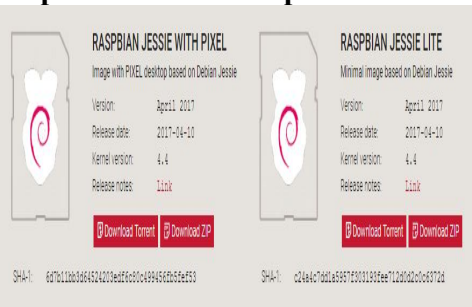
only method for installing Raspbian (more on that in a moment), but it's a useful technique to learn because it can also be used to install so many other operating systems on the Raspberry Pi. Once you know how to write a disc image to a microSD card, you open up a lot of options for fun Raspberry Pi projects.

A word about NOOBS

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

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

Step 1: Download Raspbian



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

Step 2: Unzip the file

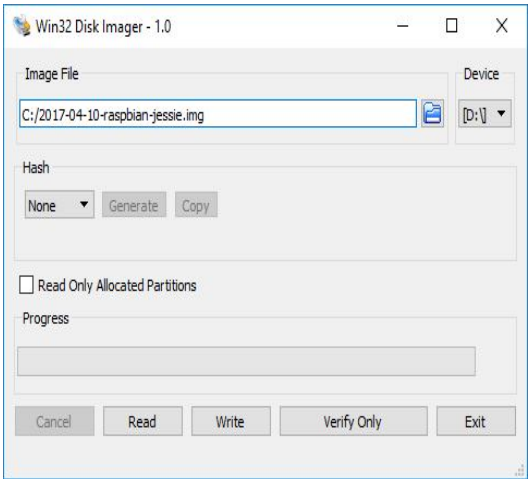
The Raspbian disc image is compressed, so you'll need to unzip it. The file uses the ZIP64 format, so depending on how current your built-in utilities are, you need to use certain programs to unzip it. If you have any trouble, try these programs recommended by the Raspberry Pi Foundation:

Windows users: you'll want **7-Zip.**

Mac users: **The Unarchiver** is your best bet.

Linux users: will use the appropriately named **Unzip**.

Step 3: Write the disc image to your microSD card



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

Windows users: your answer is **Win32 Disk Imager**.

Mac users: you can use the disk utility that's already on your machine.

Linux people: **Etcher**, which also works on Mac and Windows is what the Raspberry Pi Foundation recommends.

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

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

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

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

OPENCV INSTALLATION ON RASPBERRY PI:

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

- Physical access to your Raspberry Pi 3 so that you can open up a terminal and execute commands
- Remote access via SSH or VNC.

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

Can't SSH?

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

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

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

Installing OpenCV 3 on a Raspberry Pi 3 running Raspbian Stretch

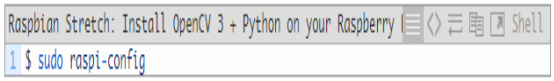
If you've ever installed OpenCV on a Raspberry Pi (or any other platform before), you know that the process can be quite time consuming with many dependencies and pre-requisites that have to be installed. **The goal of this tutorial is to thus guide you step-by-step through the compile and installation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

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

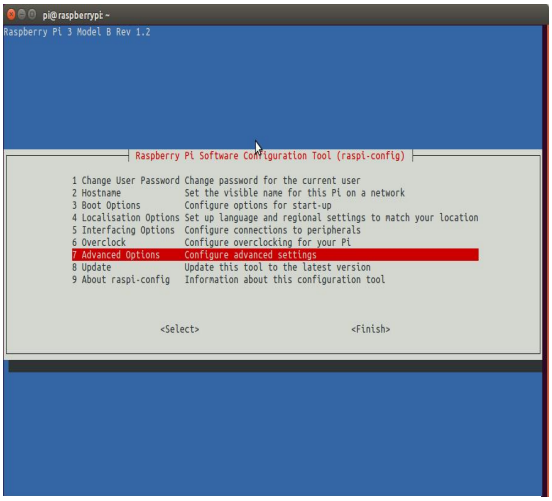
Step #1: Expand filesystem

Are you using a brand new install of Raspbian Stretch?

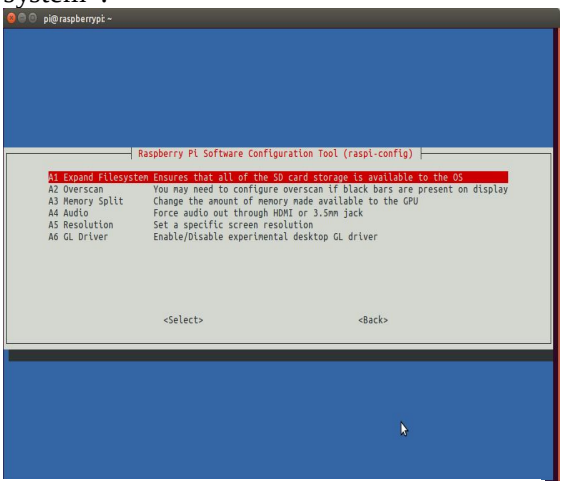
If so, the first thing you should do is expand your filesystem to include all available space on your micro-SD card:



And then select the "Advanced Options" menu item:



Followed by selecting "Expand file system":



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

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ sudo reboot
```

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ sudo apt-get update && sudo apt-get upgrade
```

After rebooting, your file system should have been expanded to include all available space on your micro-SD card. You can verify that the disk has been expanded by executing df -h and examining the output:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ df -h
2 Filesystem     Size  Used Avail Use% Mounted on
3 /dev/root       30G  4.2G   24G  15% /
4 devtmpfs       434M     0  434M   0% /dev
5 tmpfs          438M     0  438M   0% /dev/shm
6 tmpfs          438M   12M  427M   3% /run
7 tmpfs          5.0M  4.0K  5.0M   1% /run/lock
8 tmpfs          438M     0  438M   0% /sys/fs/cgroup
9 /dev/mmcblk0p1  42M   21M   21M  51% /boot
10 tmpfs          88M     0   88M   0% /run/user/1000
```

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

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

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

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ sudo apt-get purge wolfram-engine
2 $ sudo apt-get purge libreoffice*
3 $ sudo apt-get clean
4 $ sudo apt-get autoremove
```

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

Step #2: Install dependencies

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

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

Timing: 2m 14s

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

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ sudo apt-get install build-essential cmake pkg-config
```

Timing: 19s

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

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
```

Timing: 21s

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

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
2 $ sudo apt-get install libxvidcore-dev libx264-dev
```

Timing: 32s

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

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ sudo apt-get install libgtk2.0-dev libgtk-3-dev
```

Timing: 1m 36s

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

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ sudo apt-get install libatlas-base-dev gfortran
```

Timing: 23s

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

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

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

Timing: 45s

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

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

Step #3: Download the OpenCV source code

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

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

Timing: 41s

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

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

Timing: 37s

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

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

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

Step #4: Python 2.7 or Python 3?

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

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

Timing: 33s

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

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

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

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

In short, it solves the "Project X depends on version 1.x, but Project Y needs

4.x" dilemma. It also keeps your global site-packages neat, tidy, and free from clutter.

If you would like a full explanation on why Python virtual environments are good practice, absolutely give this excellent blog post on RealPython a read.

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

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

Timing: 35s

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

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

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

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

1. Logging out and then logging back in.
2. Closing a terminal instance and opening up a new one
3. Or my personal favorite, **just use the** source **command:**

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

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

Creating your Python virtual environment

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

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

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

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

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

Timing: 24s

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

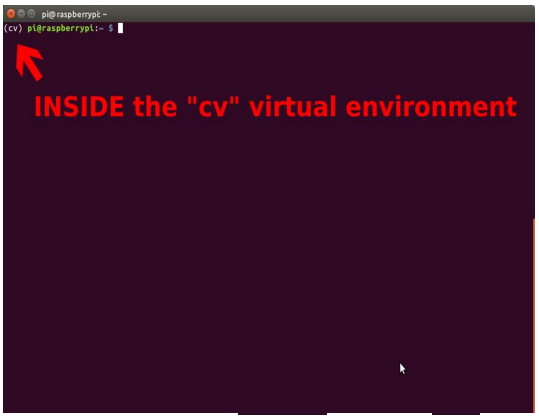
If you ever reboot your Raspberry Pi; log out and log back in; or open up a new terminal, you'll need to use the workon command to re-access the cv virtual environment. In previous blog posts, I've seen readers use the mkvirtualenv command — **this is entirely unneeded!**The mkvirtualenv command is meant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

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

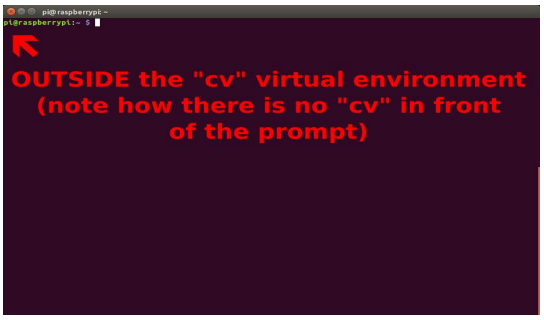
```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry
1 $ source ~/.profile
2 $ workon cv
```

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

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



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



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

Installing NumPy on your Raspberry Pi

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

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

Timing: 11m 12s

Step #5: Compile and Install OpenCV

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

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

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

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

Timing: 2m 56s

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

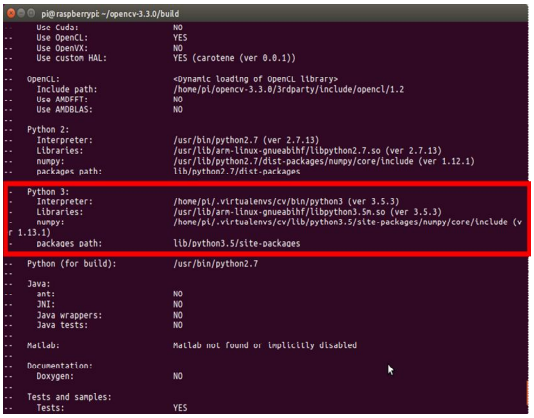
Start by scrolling down the section titled Python 2 and Python 3 .

If you are compiling OpenCV 3 for Python 2.7, then make sure your Python 2 section includes valid paths to

the Interpreter , Libraries , numpy and packa ges path , similar to my screenshot below:

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

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



Again, the Interpreter points to our python3.5 binary located in the cv virtual environment while numpy points to our NumPy install.
In either case, if you do not see the cv virtual environment in these variables paths, **it's almost certainly because you are NOT in the** cv **virtual environment prior to running CMake!**
If this is the case, access the cv virtual environment using workon cv and re-run the cmake command outlined above.

Configure your swap space size before compiling

Before you start the compile process, you should **increase your swap space size**. This enables OpenCV to **compile with all four cores** of the Raspberry PI without the compile hanging due to memory problems.
Open your /etc/dphys-swapfile and then edit the CONF_SWAPSIZE variable:

```
# set size to absolute value, leaving empty (default) then uses computed value
#   you most likely don't want this, unless you have an special disk situation
# CONF_SWAPSIZE=100
CONF_SWAPSIZE=1024
```

Notice that I've commented out the 100MB line and added a 1024MB line. This is the secret to getting compiling with multiple cores on the Raspbian Stretch.
If you skip this step, OpenCV might not compile.
To acivate the new swap space, restart the swap service:

```
$ sudo /etc/init.d/dphys-swapfile stop
$ sudo /etc/init.d/dphys-swapfile start
```

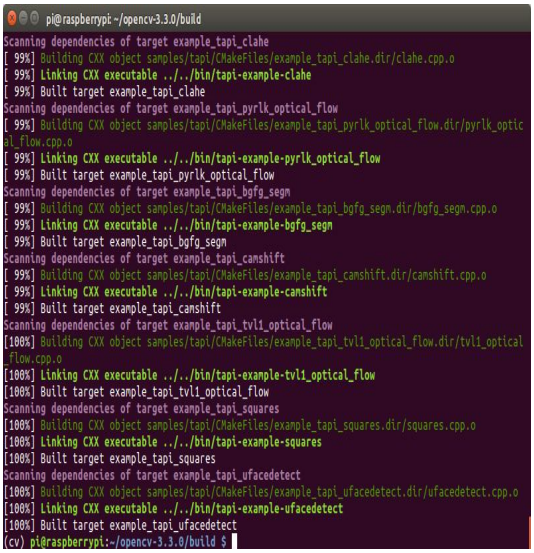
Note: It is possible to burn out the Raspberry Pi microSD card because flash memory has a limited number of writes until the card won't work. It is **highly recommended** that you change this setting back to the default when you are done compiling and testing the install (see below). To read more about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

Finally, we are now ready to compile OpenCV:

```
$ make -j4
```

Timing: 1h 30m

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



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

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

Timing: 52s

Step #6: Finish installing OpenCV on your Pi

We're almost done — just a few more steps to go and you'll be ready to use your Raspberry Pi 3 with OpenCV 3 on Raspbian Stretch. Provided your **Step #5** finished without error, OpenCV should now be installed in/usr/local/lib/python2.7/site-pacakges . You can verify this using the ls command:

```
$ ls -l /usr/local/lib/python2.7/site-packages/
total 1852
-rw-r--r-- 1 root staff 1895772 Mar 20 20:00 cv2.so
```

Note: In some cases, OpenCV can be installed in /usr/local/lib/python2.7/dist-packages(note the dist-packages rather than site-packages). If you do not find the cv2.so bindings in site-packages , we be sure to check dist-packages .

Our final step is to sym-link the OpenCV bindings into our cv virtual environment for Python 2.7:

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

For Python 3:

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

Again, you can verify this with the ls command:

```
$ ls -l /usr/local/lib/python3.5/site-packages/
total 1852
-rw-r--r-- 1 root staff 1895932 Mar 20 21:51 cv2.cpython-34m.so
```

```
$ cd /usr/local/lib/python3.5/site-packages/
$ sudo mv cv2.cpython-35m-arm-linux-gnueabihf.so cv2.so
```

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

```
$ cd ~/.virtualenvs/cv/lib/python3.5/site-packages/
$ ln -s /usr/local/lib/python3.5/site-packages/cv2.so cv2.so
```

Step #7: Testing your OpenCV 3 install

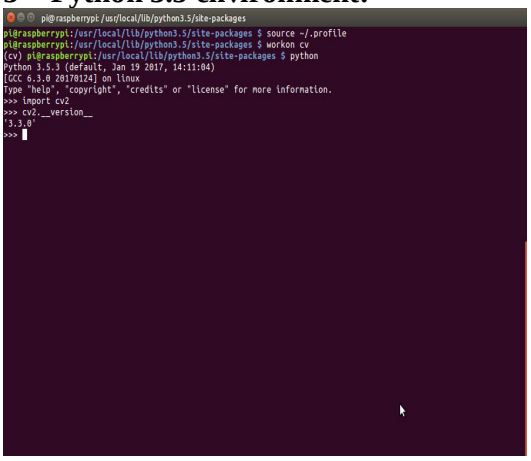
Congratulations, you now have OpenCV 3 installed on your Raspberry Pi 3 running Raspbian Stretch!

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

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

```
$ source ~/.profile
$ workon cv
$ python
>>> import cv2
>>> cv2.__version__
'3.3.0'
>>>
```

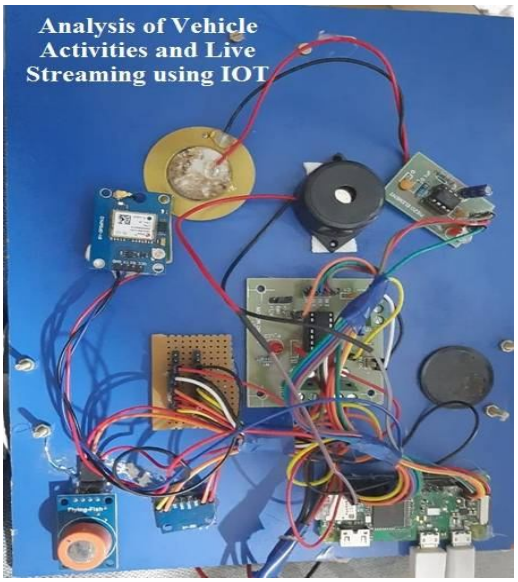
As you can see from the screenshot of my own terminal, **OpenCV 3 has been successfully installed on my Raspberry Pi 3 + Python 3.5 environment:**



Working Procedure and Result:

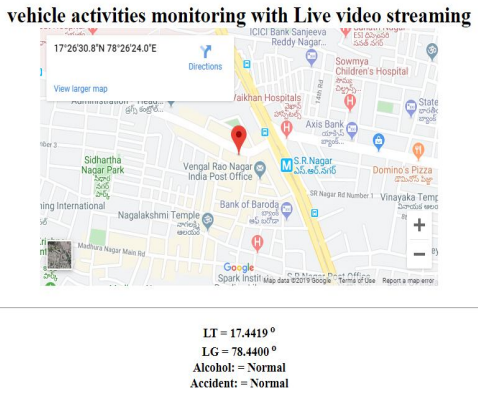
Even after the government has made many rules to avoid road accidents by not wearing the helmet and driving the bike after consuming alcohol, the accidents are not reducing. In order to overcome this, Analysis of vehicle activities is proposed.

This system uses MEMS sensor to determine vehicle position coordinates, accident sensor to detect the accident of the vehicle, alcohol sensor to detect whether the user has consumed alcohol or not, camera for live video streaming of the vehicle, GPS to calculate the vehicle location coordinates, buzzer for audio indication and motors to start or stop the engine.



Top-view of proposed system

The system working starts as: The system detects the vehicle position and also the alcohol present in rider's breath. An alcohol sensor is placed near mouth of the rider. The alcohol sensor detects the presence of alcohol in rider's breath. The data of the detection of vehicle position using MEMS sensor, accident sensor and alcohol is continuously read by Raspberry Pi and if any of the sensors is triggered, the system will stop the engine by turning off the motors.



Ouput of Proposed System

The camera will take the video of the vehicle and this video will be uploaded to the Motion server where the live video streaming of the vehicle can be seen. GPS module will upload the vehicle location coordinates to the web server. Thus, the vehicle met with an accident can be located easily and immediate treatment can be started.

Applications:

Easy to observe the vehicle condition.
Live streaming of video will help in theft.
Easy to find the vehicle location with GPS.

Future scope

This project offers a lot of scope for adding newer features. Since all image processing is done remotely, there are no resource constraints apart from the bandwidth of the network. Implementing of email sending and message alerts to users when accident occur. Sending alerts to nearest Police station and Hospital. It can integrate in all automations like Industry and military with live video.

Conclusion

A system was design method for capture pictures. The recognition and exploit digital computers range of other computer. The chance of utilization computers to support the digital computer in face recognition would speed up process, however the price is to distribute the information on totally different machines, and also the security between these machines ought to be thought of once the size issue will increase, the typical error rate increase, as a result of image size reduction can decrease the standard of the image and consequently increase the error. Surveillance system is out there with varied options. Choice is predicated on numerous factors like price, quality. The planned system price effective also they are user friendly. It's application in numerous fields like military, defenses, house, workplace and setting observance.

References:

1. Gartner Says 6.4 Billion Connected Things Will Be in Use in 2016, Up 30 Percent From 2015. Available online: http://www.gartner.com/newsroom/id/3165317 (accessed on 7 September 2016).
2. Navarro, M.; Tyler,W.D.; Villalba, G.; Li, Y.; Zhong, X.; Erratt, N.; Liang, X.; Liang, Y. Towards Long-Term Multi-Hop WSN Deployments for Environmental Monitoring: An Experimental Network Evaluation. J. Sens. Actuator Netw. **2014**, 4, 297–330. [CrossRef]
3. Vellidis, G.; Tucker, M.; Perry, C.; Kvien, C.; Bednarz, C.A Real-time Wireless Smart Sensor Array for Scheduling Irrigation. Comput. Electron. Agric. **2008**, 61, 44–50. [CrossRef]
4. Karnouskos, S. Cyber-Physical Systems in the SmartGrid. In Proceedings of the Industrial Informatics, Lisbon, Portugal, 26–29 July 2011.
5. Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. Internet of Things for Smart Cities. IEEE Internet Things J. **2014**, 1, 22–32. [CrossRef].