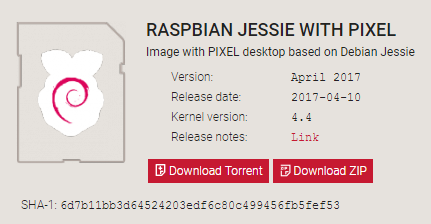
Smart Parking via Distributed Vision Version 3.0 11/28/2017

This project utilizes a camera-based system to provide an effective solution for general parking area/garages. The project combines the physical sensors, microcontrollers, smartphones and dedicated software to demonstrate the fundamental idea of IoT (Internet of Things). Our goal is to monitor the available parking space and update the real-time data on the Amazon Web Service (AWS) to allow the users to access the information on their smartphone.

**Installing Raspbian on Raspberry Pi:**

* 1. **Download** Image from <https://www.raspberrypi.org/downloads/>
  2. **Unzip** .zip downloads to get the image file (.img) to write to a SD card.
  3. **Download** Etcher and install the etcher packages. https://etcher.io/
  4. **Connect** an SD card reader with the SD card inside.
  5. **Choose 'Flash!'** to begin writing data to the SD card.
  6. **NOOBS** is easy to implement. The link is on below. <https://www.raspberrypi.org/documentation/installation/noobs.md>

**Connect Raspberry Pi by PuTTY and Ethernet cable:**

1. **Download** PuTTY from <http://www.putty.org/>
2. **Download** Advance IP Scanner <https://www.advanced-ip-scanner.com/>
3. **Switch off** the Raspberry Pi board
4. **Disconnect** the motors and the batteries.
5. **Make sure** the SD card is in the Raspberry Pi.
6. **Plug in** my Wi-Fi dongle to a USB port on the Raspberry Pi.
7. **Connect** my Ethernet cable to my computer and to the Raspberry Pi.
8. **Plug in** a power adapter into the Raspberry Pi.
9. **Power on** the Raspberry Pi.
10. **Go** Control Panel.
11. **Click** Network and Internet.
12. **Click** Network Connections.
13. **Click** the internet source to share network in sharing properties.
14. **Go** Ethernet and **click** packets.
15. **Open** Advance IP Scanner search Raspberry Pi.
16. **Open** Putty and **enter** the Host Name as the IP address of Raspberry Pi
17. **Login** PuTTY and
18. **Turn On** VNC server or Teamviewer on the Raspberry Pi
19. **Go** VNC viewer or Teamviewer on my laptop.
20. **Type** IP and port to connect Raspberry Pi.

**Hardware components & Drivers:**

Raspberry Pi 3 Model B Starter kit (including Raspberry Pi 3 Model B, power adaptor, 32G micro SD card

<https://www.amazon.com/CanaKit-Raspberry-Complete-Starter-Kit/dp/B0778CZ97B/ref=sr_1_1_sspa?s=pc&ie=UTF8&qid=1511920261&sr=1-1-spons&keywords=raspberry+pi+3&psc=1>

Raspberry Pi Camera Board, Version 2.1

<http://www.newark.com/raspberry-pi/rpi-8mp-camera-board/camera-board-8-mp-raspberry-pi/dp/77Y6521>

Camera Lens to replace the original Raspberry Pi Camera lens

<http://www.m12lenses.com/16-0mm-F2-0-Mega-Pixel-CCTV-Board-Lens-p/pt-1620bmp.htm>

Lidar-Lite V3 <https://www.sparkfun.com/products/14032?gclid=CjwKCAiAr_TQBRB5EiwAC_QCq-xb02z_Mkq-JKdIUzdnb67QI5LNe9PftPCVMAV9vD1Ck940zg1RgBoCssgQAvD_BwE>

HS-422 Servo Motor

<http://www.robotshop.com/ca/en/hitec-hs422-servo-motor.html>

Servo HAT for Raspberry Pi

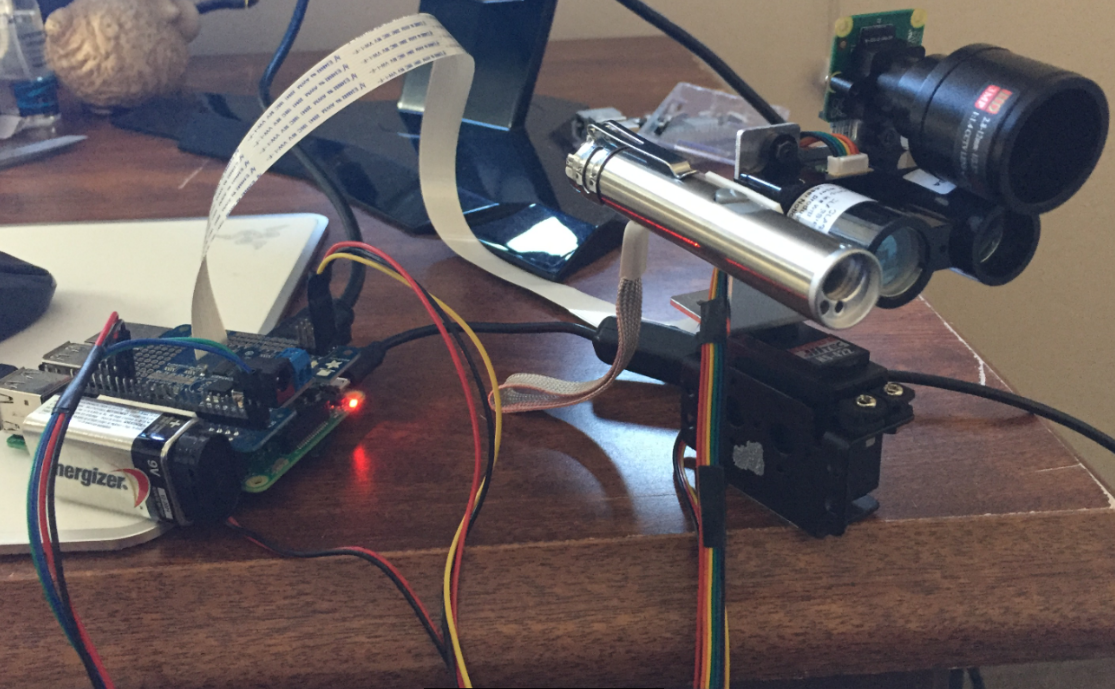
<https://www.adafruit.com/product/2327>

Bracket and frames to hold Lidar

<https://www.superdroidrobots.com/shop/item.aspx/lidar-lite-mounting-bracket/2324/>

Assemble the Hardware components

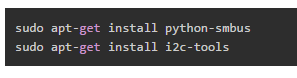
1. Install the bracket for holding the Lidar on the Servo Motor
2. The Servo HAT can be mounted onto the Raspberry Pi module.
3. Connect the Lidar sensor and Pi using I2C interface
4. Connect the Servo motor and Pi using PWM pin on the Servo HAT (A 9v battery is needed to power the Servo HAT)
5. Replace the original Raspberry Pi Camera with new lens. Tutorial Link: <http://wiki.raspberrytorte.com/index.php?title=Camera_Module_Lens_Modifcation>
6. Connect Raspberry Pi Camera to the Raspberry Pi module



Software required in the Raspberry Pi

I2C interface for the Raspberry Pi and Servo HAT.

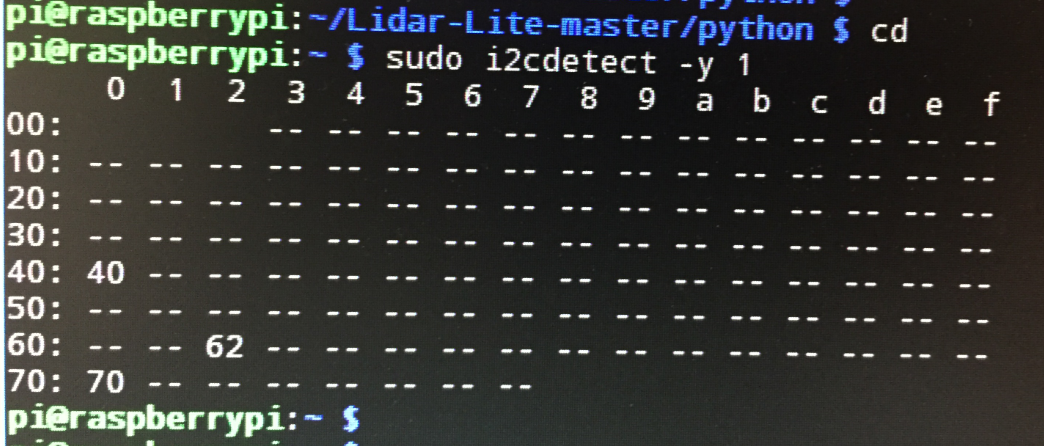
<https://learn.adafruit.com/adafruit-16-channel-pwm-servo-hat-for-raspberry-pi/>



when the Servo HAT is connected, you should see the following. The device address is 0x40



I2C interface for the Raspberry Pi and Lidar Lite V3. The device address is 0x62

.

Library for Lidar Sensor: <https://github.com/Sanderi44/Lidar-Lite>

**Camera Installation, Image IO packages, OpenCV & OpenALPR:**

**Raspberry Pi camera installation**

* 1. **Connect** the camera module into the Raspberry Pi’s camera port.
  2. Reference: <https://www.youtube.com/watch?v=GImeVqHQzsE>
  3. **Enable** my camera module by using the Raspberry Pi command.

$ sudo raspi-config

* 1. **Reboot** my Raspberry Pi
  2. **Test** out the camera module

$ raspistill -o output.jpg

**Developer tools & Image I/O packages**

* 1. All of the instructions are followed by the website <https://www.pyimagesearch.com/2016/04/18/install-guide-raspberry-pi-3-raspbian-jessie-opencv-3/>
  2. **Command** terminal shown below to update the Raspberry Pi



* 1. **Install** the developer tools and packages such as Cmake.



* 1. **Install** the image I/O packages such as jpeg, tiff, jasper, and png.



* 1. **Install** the GTK development library



* 1. **Install** the video I/O packages such as avcodec and avformat.



* 1. **Install** the libraries used to optimize the operations within OpenCV.



* 1. **Install** pip.



* 1. **Install** virtualenv and virtualenvwrapper (Optional)



**Update Raspberry Pi profile after installing Virtual Environment (Optional)**

* 1. Update Profile



* 1. Reload Profile



* 1. Create computer vision virtual environment named cv



**OpenCV Installation**

* 1. **Download** OpenCV and unpack packets



From <http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.10/opencv-2.4.10.zip/download>

However, our team has used the latest versions of OpenCV (Version 3.2)

Therefore change 2.4.10.zip into 3.2.0.zip and download from <https://github.com/Itseez/opencv/archive/3.2.0.zip>

* 1. **Setup** the build



* 1. **Command**:

$ cmake -D CMAKE\_BUILD\_TYPE=RELEASE -D CMAKE\_INSTALL\_PREFIX=/usr/local -D BUILD\_NEW\_PYTHON\_SUPPORT=ON -D INSTALL\_C\_EXAMPLES=ON -D INSTALL\_PYTHON\_EXAMPLES=ON  -D BUILD\_EXAMPLES=ON ..

* 1. **Install** OpenCV
     1. **Compile and install** OpenCV into the Raspberry Pi with 4 CPUs



* + 1. **Compile and install** OpenCV into the Raspberry Pi with a simple CPU





* 1. **Locate** the OpenCV in /usr/local/lib/python2.7/site-packages



However, our team uses python 3.4. Therefore, replace the python2.7 to python3.4 because they are not sharing the same packages.

* 1. **Test** the installation of OpenCV by using Python



* 1. Example of using OpenCV
     1. Import a OpenCV library into the Python program
     2. Call the functions included in OpenCV
        1. For example, cv2.cvtColor.



**Installing PiCamera Library**

* 1. **Activate** the virtual environment (Optional)



* 1. **install** PiCamera by utilizing pip



* 1. Example of using PiCamera



**Installing NumPy on my Raspberry Pi**

* 1. Using pip command to install numpy packa



**OpenALPR Implementation**

Our team has followed the compilation instructions of the OpenALPR installation. The following link shows how the installation can be processed in Linux or Raspbian OS.

<https://github.com/openalpr/openalpr/wiki/Compilation-instructions-(Ubuntu-Linux)>

* 1. **Install** prerequisites of OpenALPR installation such as libopencv-dev and libtesseract-dev.

$ sudo apt-get install libopencv-dev libtesseract-dev git cmake build-essential libleptonica-dev

$ sudo apt-get install liblog4cplus-dev libcurl3-dev

* 1. **Install** beanstalkd library in the Raspberry Pi

$ sudo apt-get install beanstalkd

* 1. **Get** the latest code of OpenALPR from GitHub

$ git clone <https://github.com/openalpr/openalpr.git>

* 1. **Setup** a compile environment for OpenALPR installation

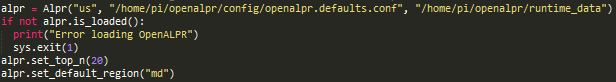
$ cmake -DCMAKE\_INSTALL\_PREFIX:PATH=/usr -DCMAKE\_INSTALL\_SYSCONFDIR:PATH=/etc ..

* 1. **Compile** the library

$ make –j4

$ sudo make install

* 1. Example of using OpenAlpr library on Python



**AWS IoT and DynamoDB Setup:**

1. Set Up AWS IoT. Follow the official developing guide from Amazon. <http://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html>
2. Setup an AWS account and AWS IoT service.
3. Register the things (Cameras) and set the rules <http://docs.aws.amazon.com/iot/latest/developerguide/register-device.html>
4. Set a rule that atomically pushed the message from IoT device to DynamoDB, REMEMBER to use DynamoDBv2.
5. Create a table on DynamoDB follow the instruction or the guideline from the AWS IoT console. <http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SettingUp.DynamoWebService.html>
6. Try the MQTT simulator on AWS IoT console to publish some fake sensor data or license data to your own assigned topic and double check if the data is being stored in the database. <http://docs.aws.amazon.com/iot/latest/developerguide/config-and-test-rules.html>

**AWS IoT SDK and Run code on Raspberry Pi:**

1. Prerequisite: Install OpenCV and other dependences; setup all hardware drivers and libraries.
2. Download the AWS IoT SDK from Amazon and follow the official guide to set the SDK on Pi <http://docs.aws.amazon.com/iot/latest/developerguide/iot-sdk-setup.html> & <https://github.com/aws/aws-iot-device-sdk-python/blob/master/README.rst>
3. After that, please save the credentials from AWS to the Pi.
4. Replace the credentials or reset the default credentials in sample code.
5. Run the code on CMD by typing Final.py. Note please enter the relevant filed

python3 FileName.py -endpoint –root\_CA -key -cert –topic

1. Note: Please change the parameters and constants in the code such as positions, distances, and angels based on your measurement of your own testing environment. You might also need to change or add control flows of the program as well to get the best result in different situation.