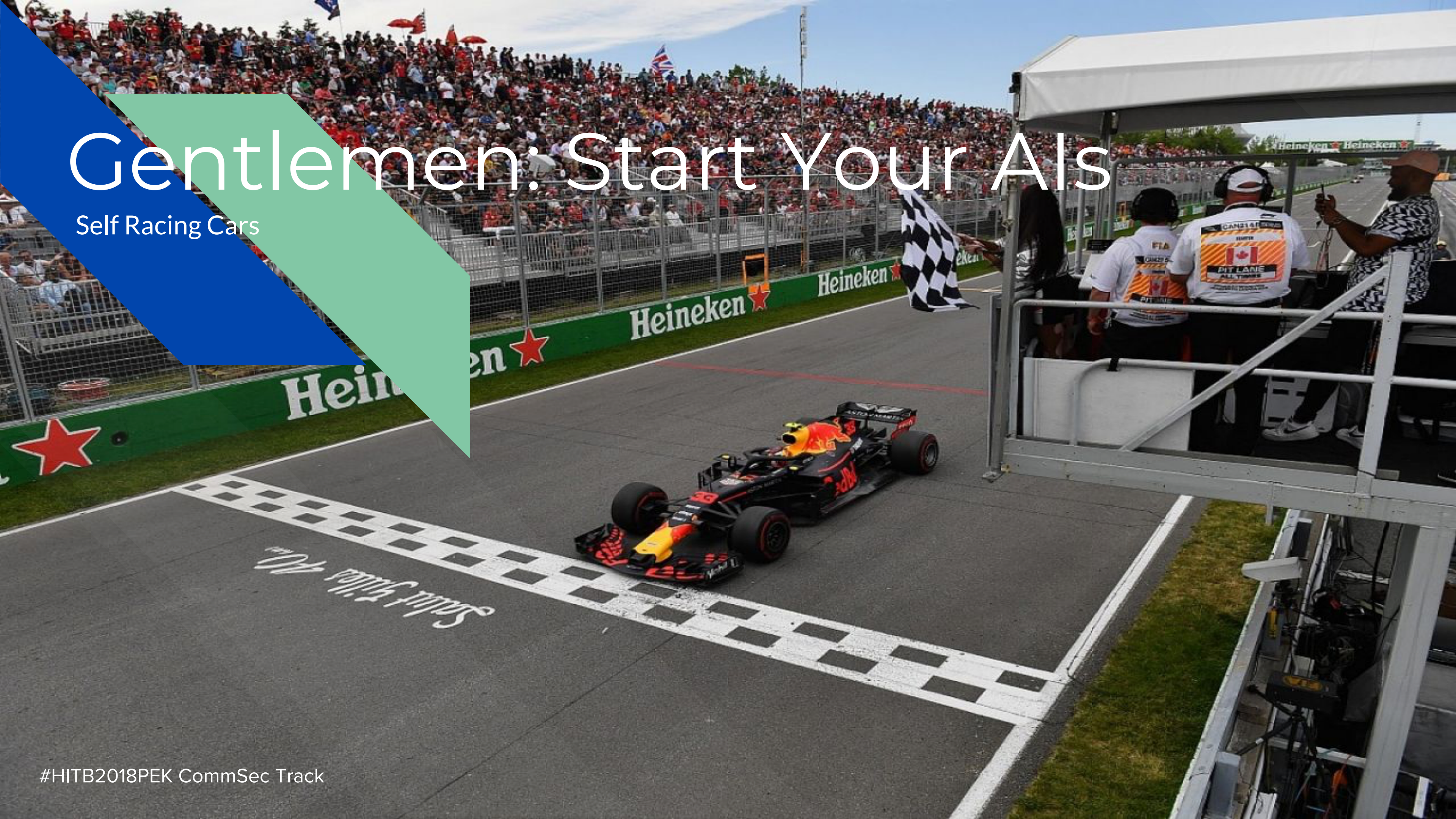
Gentlemen: Start Your AIs

Self Racing Cars

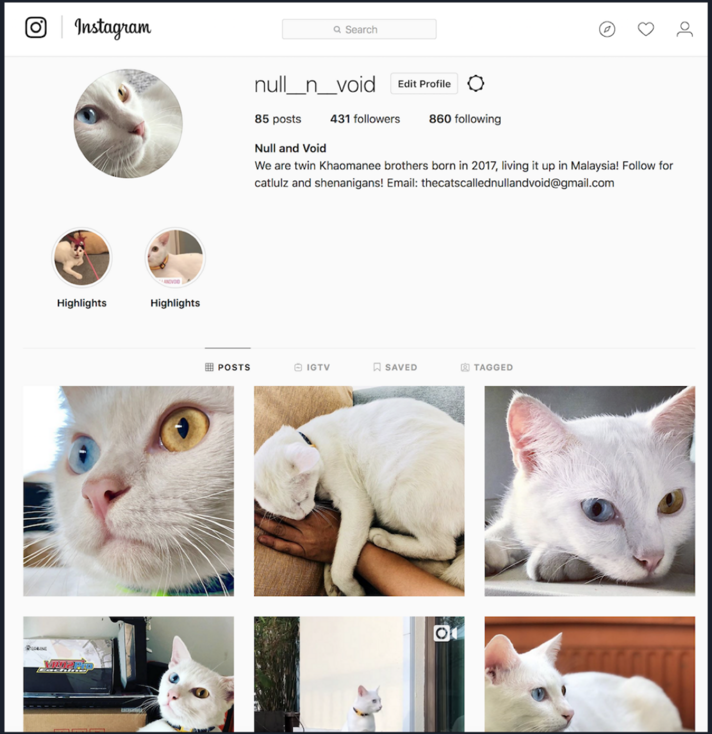


#whoami

- Founder / CEO @ Hack In The Box
- @L33tdawg on Twitter

#more

- Wrote code a long time ago
- Founder of Tumpang.la
- AI / ML Enthusiast
- I live in Malaysia with 2 ‘famous’ cats



#HITB2018PEK CommSec Track

#State of the AI(rt)

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

DeepRCar: An Autonomous Car Model

Bc. David Ungurean

Department of Applied Mathematics
Supervisor: Ing. Zdeněk Buk, Ph.d.

May 9, 2018

https://dspace.cvut.cz/bitstream/handle/10467/76316/F8-DP-2018-Ungurean-David-thesis.pdf

DeepPicar: A Low-cost Deep Neural Network-based Autonomous Car

Michael G. Bechtel†, Elise McEllhiney†, Minje Kim*, Heechul Yun†
† University of Kansas, USA. {mbechtel, elisemmc, heechul.yun}@ku.edu
* Indiana University, USA. minje@indiana.edu

***Abstract*—We present DeepPicar, a low-cost deep neural network based autonomous car platform. DeepPicar is a small scale replication of a real self-driving car called DAVE-2 by NVIDIA. DAVE-2 uses a deep convolutional neural network (CNN), which takes images from a front-facing camera as input and produces car steering angles as output. DeepPicar uses the same network architecture—9 layers, 27 million connections and 250K parameters—and can drive itself in real-time using a web camera and a Raspberry Pi 3 quad-core platform. Using DeepPicar, we analyze the Pi 3's computing capabilities to support end-to-end deep learning based real-time control of autonomous vehicles. We also systematically compare other contemporary embedded computing platforms using the DeepPicar's CNN-based real-time control workload.**

We find that all tested platforms, including the Pi 3, are capable of supporting the CNN-based real-time control, from 20 Hz up to 100 Hz, depending on hardware platform. However, we find that shared resource contention remains an important issue that must be considered in applying CNN models on shared memory based embedded computing platforms; we observe up to 11.6X execution time increase in the CNN based control loop due to shared resource contention. To protect the CNN workload, we also evaluate state-of-the-art cache partitioning and memory bandwidth throttling techniques on the Pi 3. We find that cache partitioning is ineffective, while memory bandwidth throttling is an effective solution.

***Keywords*-Real-time, Autonomous car, Convolutional neural network, Case study**

I. Introduction

Autonomous cars have been a topic of increasing interest in recent years as many companies are actively developing related hardware and software technologies toward fully autonomous driving capability with no human intervention. Deep neural networks (DNNs) have been successfully applied in various perception and control tasks in recent years. They are important workloads for autonomous vehicles as well. For example, Tesla Model S was known to use a specialized chip (MobileEye EyeQ), which used a vision-based real-time task may be directly linked to the safety of the vehicle. This requires a high computing capacity as well as the means to guaranteeing the timings. On the other hand, the computing hardware platform must also satisfy cost, size, weight, and power constraints, which require a highly efficient computing platform. These two conflicting requirements complicate the platform selection process as observed in [25].

To understand what kind of computing hardware is needed for AI workloads, we need a testbed and realistic workloads. While using a real car-based testbed would be most ideal, it is not only highly expensive, but also poses serious safety concerns that hinder development and exploration. Therefore, there is a need for safer and less costly testbeds.

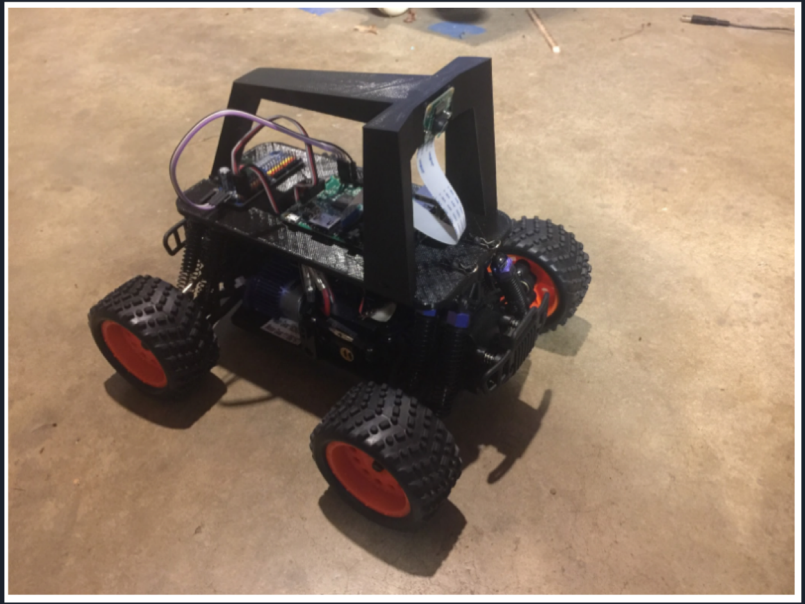
In this paper, we present DeepPicar, a low-cost autonomous car testbed for research. From a hardware perspective, DeepPicar is comprised of a Raspberry Pi 3 Model B quad-core computer, a web camera and a small RC car, all of which are affordable components (less than $100 in total). The DeepPicar, however, employs a state-of-the-art AI technology, namely end-to-end deep learning based real-time control, which utilizes a deep convolutional neural network (CNN). The CNN receives an image frame from a single forward looking camera as input and generates a predicted steering angle value as output at each control period in *real-time*. The CNN has 9 layers, about 27 million connections and 250 thousand parameters (weights). DeepPicar's CNN architecture is identical to that of NVIDIA's real-sized self-driving car, called DAVE-2 [5], which drove on public roads without human driver's intervention while only using the CNN.

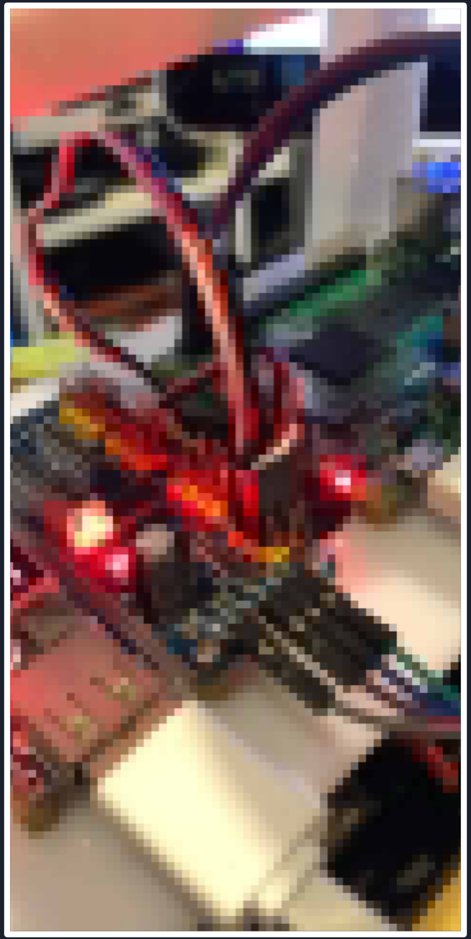
Using DeepPicar, we systematically analyze its real-time capabilities in the context of end-to-end deep-learning based real-time control, especially on real-time *inferencing* of the CNN. We also evaluate other, more powerful, embedded computing platforms to better understand achievable real-time performance of DeepPicar's CNN based control system

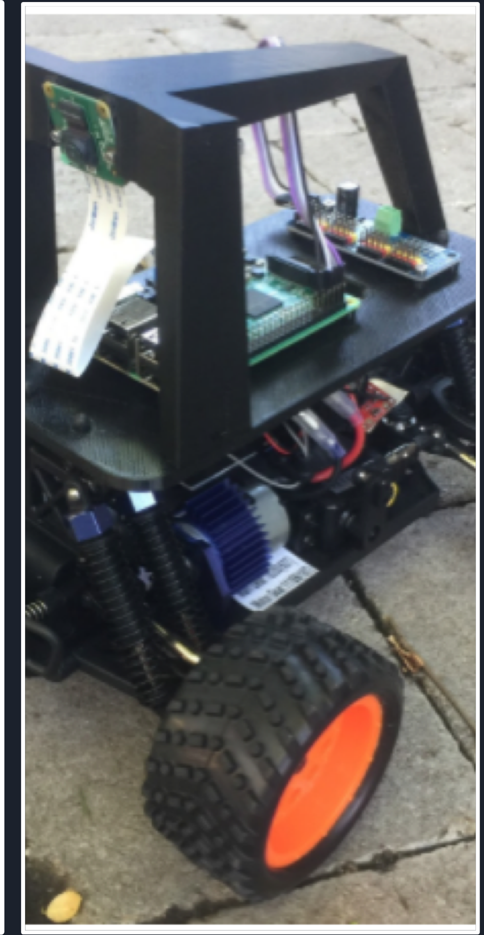
https://arxiv.org/pdf/1712.08644.pdf

#HITB2018PEK CommSec Track

#Donkey What?

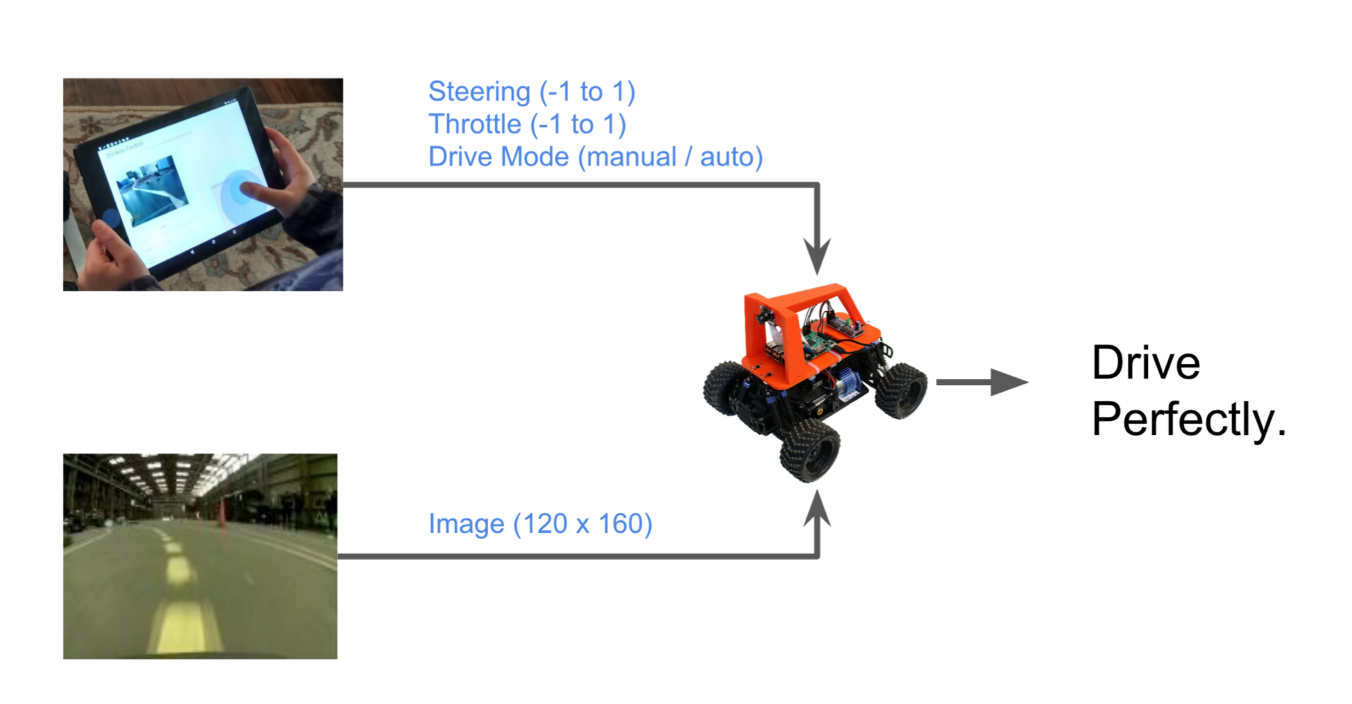




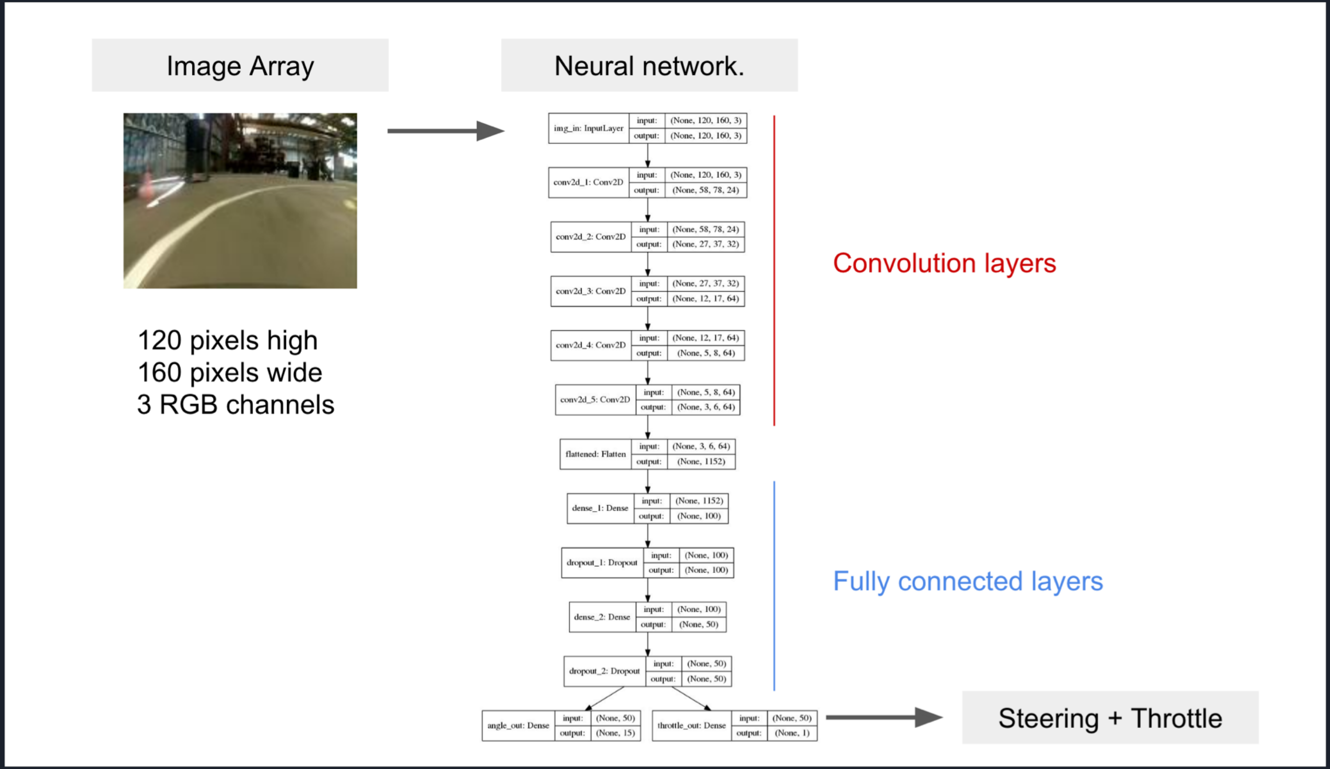


#HITB2018PEK CommSec Track

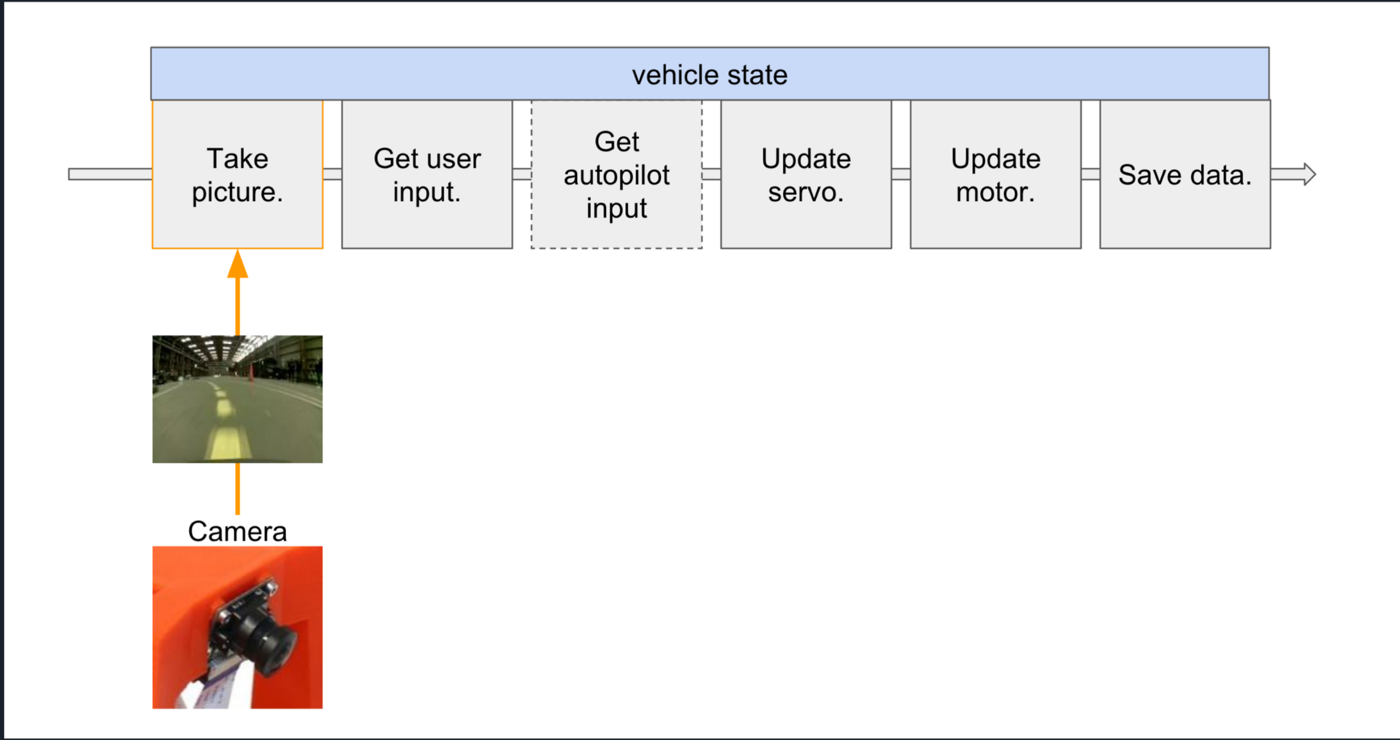
#How Does It Work?



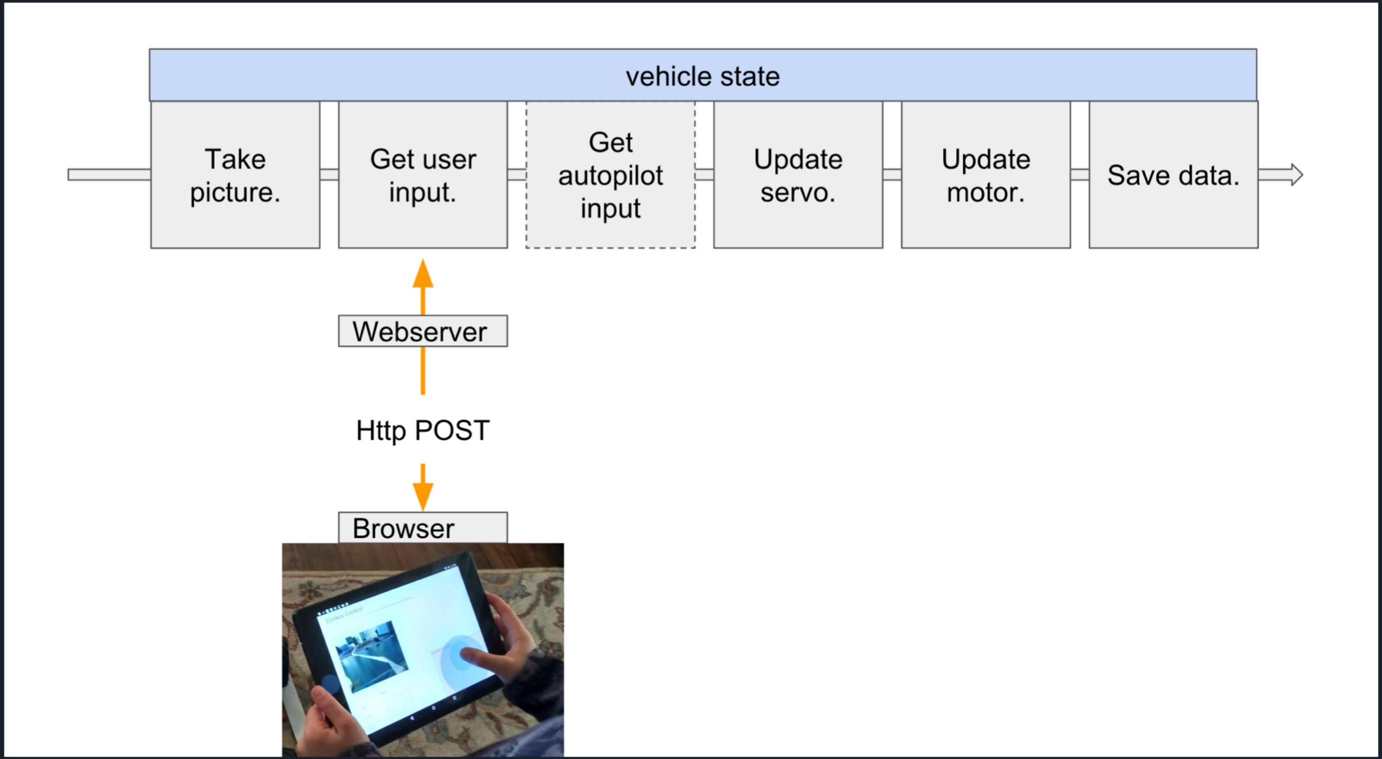
#How Does It Work?



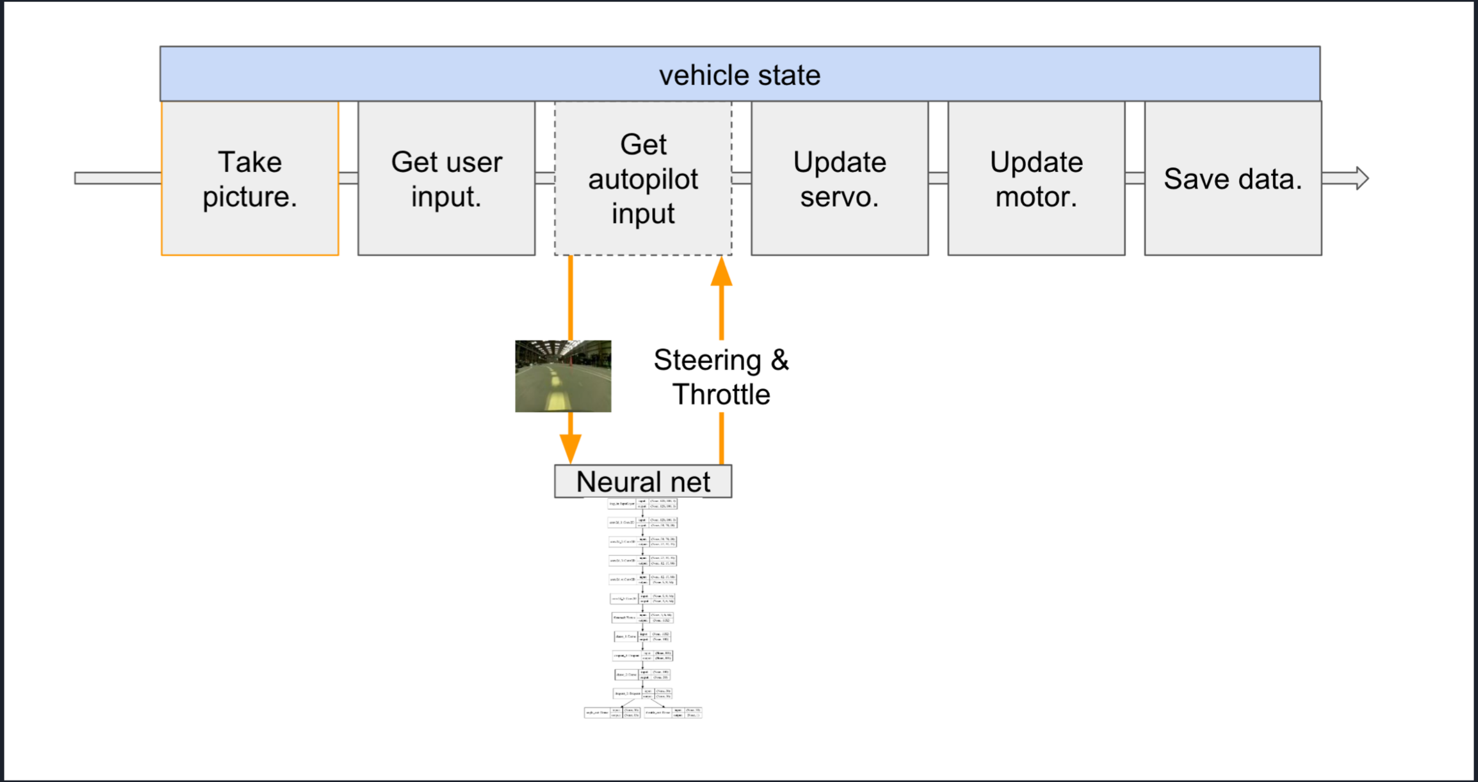
#How Does It Work?



#How Does It Work?

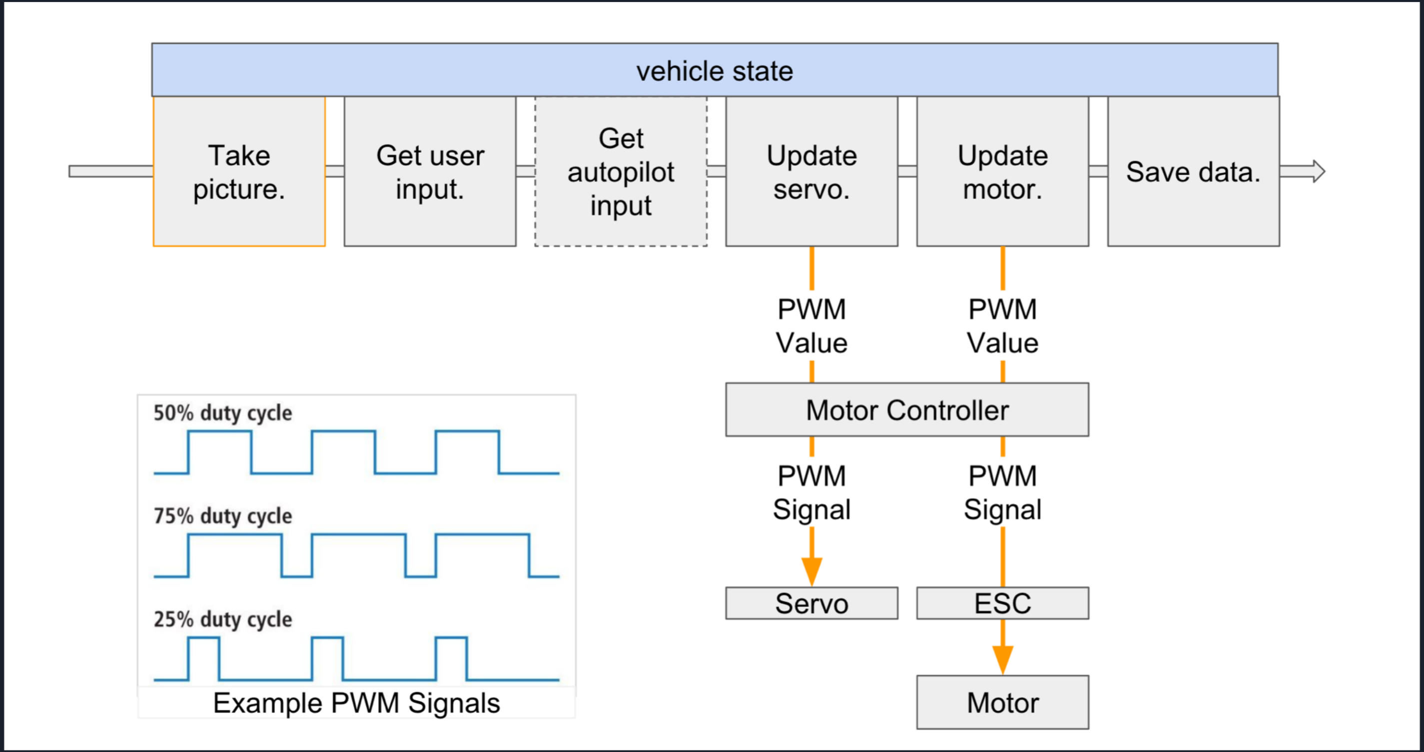


#How Does It Work?

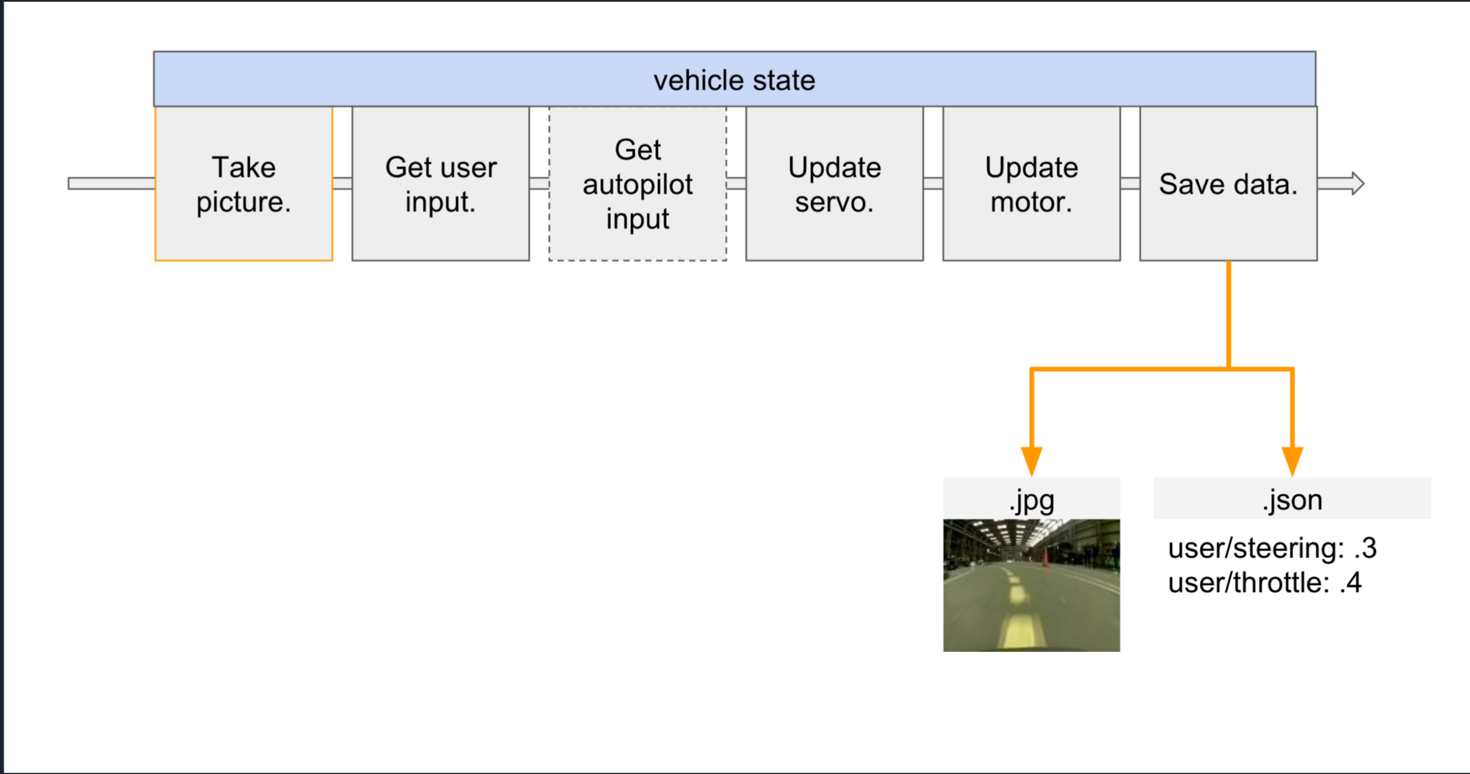


#HITB2018PEK CommSec Track

#How Does It Work?



#How Does It Work?



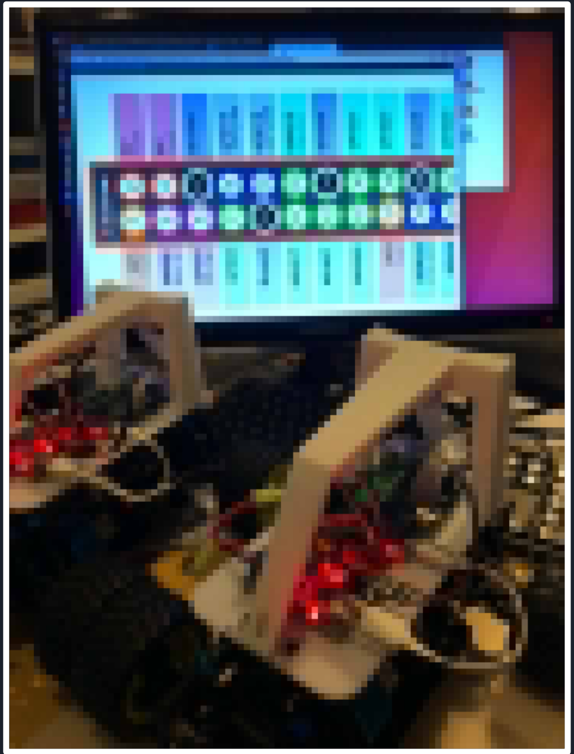
#How Does It Work?

Keras / Tensorflow Autopilots

```
img_in = Input(shape=(120, 160, 3), name='img_in')
x = img_in
x = Convolution2D(24, (5,5), strides=(2,2), activation='relu')(x)
x = Convolution2D(32, (5,5), strides=(2,2), activation='relu')(x)
x = Convolution2D(64, (5,5), strides=(2,2), activation='relu')(x)
x = Convolution2D(64, (3,3), strides=(2,2), activation='relu')(x)
x = Convolution2D(64, (3,3), strides=(1,1), activation='relu')(x)

x = Flatten(name='flattened')(x)

x = Dense(100, activation='relu')(x)
x = Dropout(.1)(x)
x = Dense(50, activation='relu')(x)
x = Dropout(.1)(x)

#categorical output of the angle
angle_out = Dense(15, activation='softmax', name='angle_out')(x)

#continous output of throttle
throttle_out = Dense(1, activation='relu', name='throttle_out')(x)
```


#HITB2018PEK CommSec Track

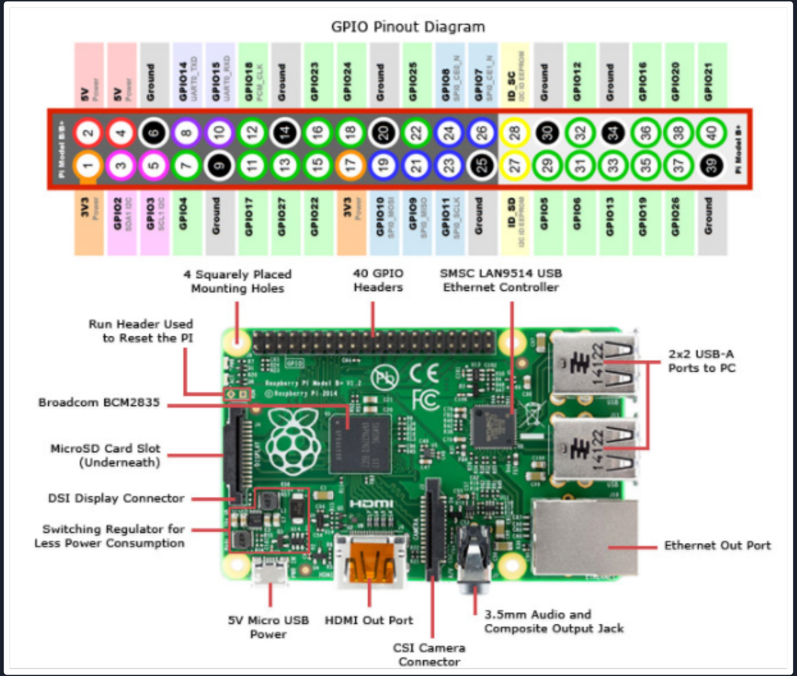
#Hardware



Magnet Car (Red / Blue) or alternative $92
M2x6 screws (4) $6
M2.5x12 screws (8) $5
M2.5 nuts (8) $6
M2.5 washers (8) $7
USB Battery with microUSB cable $17
Raspberry Pi 3 $38
MicroSD Card $20
Wide Angle Raspberry Pi Camera $25
Female to Female Jumper Wire $7
Servo Driver PCA 9685 $12
3D Printed roll cage and top plate. $45

TOTAL:

#Software



Raspberry Pi

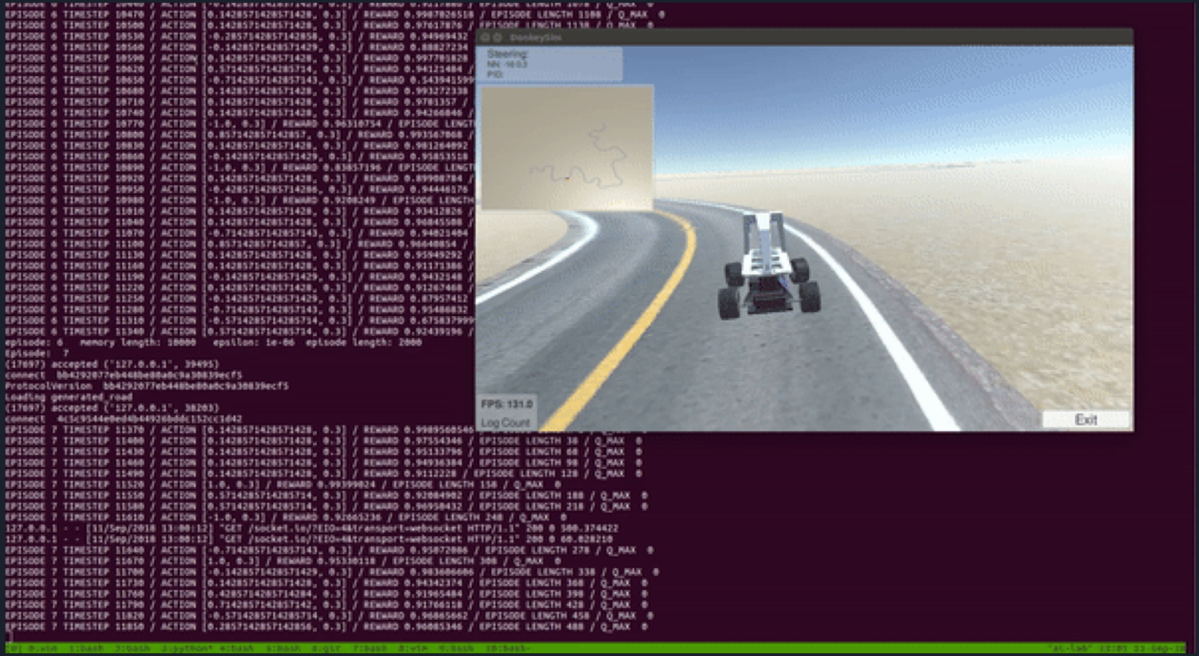
1. Download prebuilt zipped disk image (1.1GB)
2. Flash it
3. *git clone https://github.com/wroscoe/donkey*
4. *pip install -e .[pi]*

Linux / Host Machine

1. *sudo apt-get install virtualenv build-essential python3-dev gfortran libhdf5-dev*
2. *virtualenv env -p python3*
3. *source env/bin/activate*
4. *pip install tensorflow*
5. *git clone https://github.com/wroscoe/donkey*
6. *pip install -e .*

#HITB2018PEK CommSec Track

#Unity Simulator



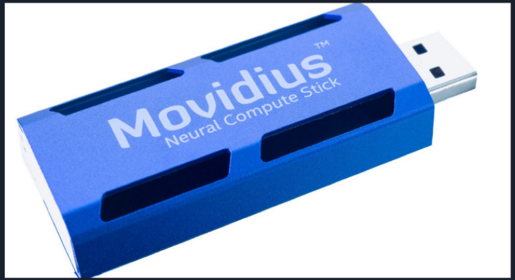
Download: https://docs.donkeycar.com/guide/simulator/

#HITB2018PEK CommSec Track

#HITB2019AMS - Gentlemen Start Your AIs

May 6th - 10th 2019

Hackerspaces / Individuals



- 1/16 scale Donkeycar - All hardware provided
- 2 batteries per team
- 3 days free practice (6 / 7 / 8th May)
- Qualifying - 9th May (2 sessions)
- Top 10 teams move to race day (10th May)

Finalists will be given Intel Movidius Neural Compute Stick - Go harder, go faster, be better!

Professional Teams

- Bring your own car - 1/10 scale
- No limit on hardware sensors
- Limited to 3 batteries per team
- 3 days free practice (6 / 7 / 8th May)
- Qualifying - 9th May (2 sessions)
- Race Day - 15 min race time

Get building, get racing, and see you in Amsterdam!

Questions?

l33tdawg@hitb.org // @L33tdawg