Finalized Hardware & Software Needed as a whole:

- Jetson Nano (Dev Kit, 2Gb, 4Gb)
- Basler dart daA1440-220uc (CS-Mount)
- LIDAR Camera (find one)
- Power Supply / Battery
-

Central Embedded System System & Management

High-Level Description:

- **Import** information from various sensors
- Decide whether the central system processes information and performs action/the ecu itself
- Kill Switch (shutdown appropriate ECUs)
- Analyzes all sensor inputs, applies rules of the road and **operates** the steering, accelerator and brakes
- **Prioritize** actions based on information

Framework (Language & Libraries):

- Linux OS
- C#, Python

Hardware & Software Equipment Needed (Funding):

- Cameras (Decent Quality, **High frame rate** (Minimum 60fps))
- High Speed CAN Bus FD - 2 Wire
- **https://www.mouser.ca/datasheet/2/389/stm32h745i-disco-1602141.pdf**
    - **Possible Central Board?**
- **Jetson Nano**
- Orange Box
- GPS - Tachometers, Altimeters, Gyroscopes
- Ultrasonic Sensors (For detection of closer objects)
- Odometry sensors
- MCU - Microcontroller Unit

Plan on Communication with Other Systems:

- Ethernet

Obstacle Vision System

High-Level Description:

The group will design a deep neural network to recognize four types of obstacles - two types of traffic cones, barricades, and potholes - and performance tune the neural network to recognize these objects with high accuracy. Training the neural network will be done with mostly online images as well as specific self created images to refine the large training dataset.

Most of the work will be spent on designing a deep neural network. Other options considering extra inputs and functionalities can be added later in the project based on the work progress of the group, which will be outlined in the proposal.

Framework (Language & Libraries):

- Python
- Tensorflow & Keras
- Training neural network on linux based VM (jupyter)

Hardware & Software Equipment Needed (Funding):

- Raspberry Pi 3 Model B+
- Raspberry Pi Camera
- Google Cloud VM Credit (300$ free)

Plan on Communication with Central System:

Seems like the central group wants to use CAN bus

Intersection Control System

High-Level Description:

- Import obstacle map
- Create an occupancy map for driveable space
- Calculate trajectory to follow in a Global map and Local map.

We will focus more on how to calculate a trajectory to follow through an intersection given information about other vehicle positions, lanes, pedestrians locations, stop signs positions, the state of traffic lights, and occupancy map of driveable space for our vehicle.

Framework (Language & Libraries):

- Java, Python
- OpenCV

Hardware & Software Equipment Needed (Funding):

- Multiple Cameras (At Least 1 front camera and Front left and front right side Cameras)
- 360 degree Lidar

Plan on Communication with Central System:

- Input data: Needs obstacles, lanes, pedestrians from other group
- Output data: Trajectory Information of going through intersections

Lane Following System

High-Level Description:

- Apply distortion correction on frame
- Create thresholded binary image
- Apply perspective transform on binary image to create birds-eye view
- Detect lane and fit to find lane boundaries
- Determine curvature of lane to centre accordingly

Framework (Language & Libraries): Python & OpenCV, Linux system

Hardware & Software Equipment Needed (Funding):

- Camera
    - >= 60fps, at least 720p
- Lidar
- GPS/GNSS
- IMU/ARHS
- 2x backup lane/line detector signals (preferred, not required)
- Possible options:
    - Bosch multipurpose camera generation 1/2 (2048 x 1280 @ 45 fps)
    - See3CAM_CU27 (1937 x 1097 @ 100 fps)

Plan on Communication with Central System:

Undetermined at the moment, CAN bus might be easiest

Predicting Pedestrian System

Also using Agile Jira to track progress

High-Level Description:
- Import Video
- Second Layer is to get the video using bounding boxes
- Once bounding boxes are created we determine the box with pedestrian
- Create landmarks
- Use object tracking to give us position and extrapolate trajectory

Framework (Language & Libraries):
- Python
- OpenCV
    - YOLOv3 (NN, Finding people)
    - deepSORT (Kalman Filters, tracking people)
- Linux / UNIX based OS

Hardware & Software Equipment Needed (Funding):
- Minimum (not exactly, could just use a dataset but where is the fun in that?)
    - A camera able to see a person ~30m away at 30fps
        - Going to try out a GoPro5 this weekend. But it's not a live feed.
        - Sony IMX219 8MP (~$90)
    - Raspberry Pi
- Ideally
    - Sony IMX273 (1.58MP) (~$180)
    - A second camera for stereo imaging
    - 360* Lidar
    - Nvidia Jetson Nano
- Super
    - Sony IMX490 (5.4MP) (tesla cam) (~$750)
    - Thermal / Infrared camera
    - Flash lidar
    - Nvidia Jetson AXG Origin

Plan on Communication with Central System:
- pigeons