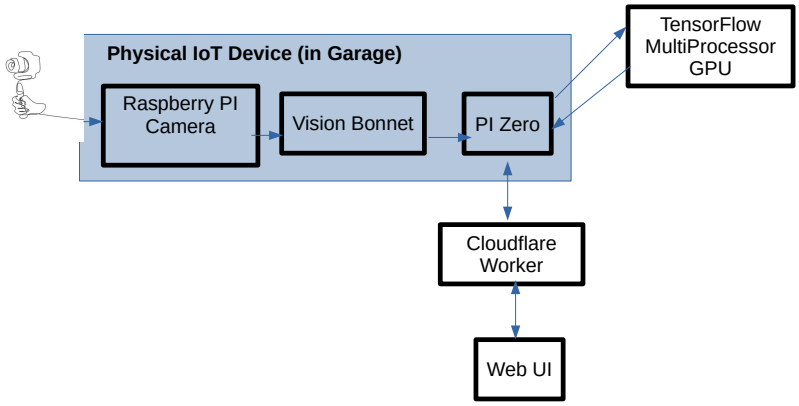

GARAGE DOOR MONITOR

Goal: Capture images and correctly identify state of garage door (inadvertently open = needyGarage; closed = happyGarage) as cheaply and easily as possible. Respond to Internet requests for status.

Design Options:
Hardware Needs: Microprocessor, camera, ML model to classify image, network interface to communicate. GPU enabled CPUs to train model. Cost and efficiency paramount.
Options: Google AIY Image Processing System has all the components, interfaces, and stubs for software development, and can be obtained for a cost of $49, including shipping. The cost to purchase a Raspberry PI, power supply and case is over twice the cost of an AIY Vision kit, even before adding the cost of the camera or the ML model. The AIY Vision kit satisfies both goals. - cost and ease. Have free access to GPU-enabled CPUs that are running TensorFlow and other ML tools.

Software Needs:
AIY Vision system comes with several pre-trained models. Detectors: Face/Joy and Dog/Cat/Human. Classifiers: Food Classifier and Image Classifier, which return a confidence score. Since this software accompanies the purchased hardware kit and there is expertise to run TensorFlorw, will use TensorFlow's Object Detection API to train model.

Network Needs:
A Raspberry PI out in the wild is fairly vulnerable. All Internet access to the Raspberry PI will occur through an Argo Tunnel provided by a (free) Cloudflare worker. Multiple vendors provide free edge devices for personal users. Cloudflare was chosen due to the ease of setup of the Argo Tunnel and the availability of expertise with this product.

Process Steps
1) Obtain a multitude of images of the two classes. For the initial iteration, happyGarage will be a garage with garage door closed. needyGarage will be a garage with a door that is not shut that needs a human to do something for it. Script will instruct the Raspberry PI to take a picture regularly; these images will be saved to the ~/Pictures directory on the PI and uploaded to a larger GPU-enabled machine for training. Estimate that the process of collecting images will take several weeks, during which time the rest of the programs can be developed.
2) Images to be hand-labeled as needyGarage or happyGarage.
3) Subdivide the images into training set, testing set and validation set.
4) Define parameters in object detection configuration file, embedded_ssd_mobilenet_v1_coco.config.
5) Using GPU-enabled TensorFlow machine, train model with TensorFlow to produce graph.
6) Freeze graph and compile using Ubuntu's Bonnet Compiler.
7) Load compiled graph onto Raspberry PI IoT device.