GARAGE DOOR MONITOR

Component Testing

1) Camera and Bonnet Hardware. Device assembled (including firmware/OS updates) and tested using the pre-loaded Joy program. In the Joy program, the system blinks differently depending on whether the device believes the user is smiling or frowning.
Apply Test Data: Test data was human tester smiling/frowning.
Observe Test Results: Tester's facial movements caused button to blink appropriately. Network testing of device achieved via remote login to device and observation of image files created in ~/Pictures directory.
Conclusion: Camera and Bonnet Hardware passed component-level testing.

2) ML Software. Script created to collect data of garage door conditions (inadvertently open vs securely closed) and data collection is in progress. Images will be hand labeled and feed into TensorFlow for training. Expect sufficient quantity of images to be collected by 7/15/20.
Apply Test Data: Dataset of images will be assigned to ML training set, ML validation set (to tune hyperparameter choices and model fiit) and test set (evaluates model fit on previously unseen data).
Observe Test Results: Based on accuracy scores from the validation set, ML training parameters will be adjusted until final model created. Based on accuracy scores from test set on final model and previously unseen data, model will be accepted or rejected.

Integration Testing

After an acceptable ML model created and compiled on Bonnet Compiler, compiled graph will be loaded on Raspberry PI IoT device.
Applying and Observing Test Data: Load compiled graph (initial ML model) on Garage Door Monitor Raspberry PI to provide classification results for human review and validation.
Long Term Testing: Scheduled reviews of ML model on monthly basis to determine if ML model needs retraining. Raspberry PI system logs to be reviewed monthly to detect anomalies.

Garage Door Monitor 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 to be created to instruct the Raspberry PI to take a picture r egularly; these images will be saved to the ~/Pictures directory on the PI and uploaded to a larger GP U-enabled machine for training. Estimate that the process of collecting images will take several wee ks, 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 on to Raspberry PI IoT device.
8) Test and re-train if necessary.