Running ROS and basic commands:
Start running ROS - this is starts your OS and must remain running while using system
$ roscore

Create a catkin workspace:
$ mkdir -p ~/catkin_ws/src
$ catkin_make

Create a new package in catkin_ws:
$ catkin_create_pkg <name> <depends1> <depends2> …

Show list of active nodes:
$ rosnode list

Show list of active topics:
$ rostopic list

Run a package and name:
$ rosrun <package_name> <node_name>

Run a script:
$ roslaunch <script>

Print path to ROS info - includes distort
$ printenv | grep ROS

Payload Sensing Commands:

The names of the cameras we have are published as:
mv_26803584 (the color camera)
mv_30000337 (the grayscale camera / IR camera)

The cameras use two sets of drivers to work with ROS. The first is the mvBlueFOX USB 2.0 drives. These can be
installed by a script on the manufacturers website or by the instructions in the MatrixVisionBluefox folder. The second is the Bluefox2 ROS package.
To launch the cameras with ROS you need to start roscore and then launch the camera individually:

$ cd ~/catkin_ws/src/bluefox2/launch
$ roslaunch single_node.launch device:=30000337
$ roslaunch single_node.launch device:=26803584

To view the camera feed ROS provides a nice display. To launch an instance of the display:
$ rviz

Using ROSbag:
Rosbag allows us to record the published topics which includes the display from the camera. Rosbag does not produce a video file. You will output a rosbag that can be played back which includes the images aka video feed from the camera.

Record a bag file with the contents of specified topics
$ rosbag record <topic_names>

Record all topics:
$ rosbag record -a

Displays a summary of the contents of the bag file- Useful to figure out the name of the topics you need to play back:
$ rosbag info <bag_name.bag>

Determine whether or not a bag is playable in the current system:
$ rosbag check <bag_name.bag>

Play a bag:
$ rosbag play <bag_name.bag>

***Specifically to watch back the bag files that we record with our cameras:**
$ rviz
$ rosbag play <bag_name.bag>
The video feed from both cameras should be visible and playing

The cameras provide many adjustable parameters, many of which need to be adjusted to get a nice video.
There is an Ros page on this and it has a few commands but its not that helpful and doesn't explain the purpose of each parameter or the full name
To get the parameters:
$ rosin dynamic_reconfigure dynparam list

Get particular param:
$ rosrun dynamic_reconfigure dynparam get <param>

To set a particular parameter:
$ rosrun dynamic_reconfigure dynparam set <camera name> <param> <value>
i.e. rosrun dynamic_reconfigure dynparam set /mv_26803584 web 4

some helpful parameters abbr meanings that we were able to figure out:
wbp: controls white balance. needs an integer value. default is 1 but has been changed to 4

- -1 - wbp_unavailable
- 0 ~ 5 - wpb_tungsten and friends
- 6 - wbp_user1

7 - wbp_calibrate, calibrate next time for white balance
aec: enables automatic exposure control. needs an integer value. Default is 0
0 - aec_off, fixed exposure time
1 - aec_on, auto control by driver
2 - aec_fix, auto determined by driver and set to a fixed value
3 - aec_clamp, auto control by driver, but clamped to expose time set by used in expose_us
fps: frames per second. needs a double. default is 20. camera dependent
cbm: camera binning mode. needs boolean. default false , true uses BinningHV which is horizontal and vertical binning. changing distorts image
ctm: camera trigger mode. 0 or 1 integer value. 0 is ctm_continuous, 1 is ctm_on_demand. recommend 1
dcfm: dark current filter mode. default 1. needs int
0 - dcfm_off
1 - dcfm_on
2 - dcfm_calibrate
3 - correction_image
hdr: high dynamic range. 0 or 1 integer value.
expose_us: exposure time. The number of microseconds each frame is exposed for. Default is 5000. changed to 23336
gain_db: adjust camera gain. artificially increases brightness of image. needs a double. default is 0.0
r_gain: adjust red camera gain. needs a double
b_gain: adjust blue camera gain. needs a double
mm: mirror mode . can only be adjusted in launch file, not dynamically. 0 is default.
0 - mm_off, no mirroring
1 - mm_topdown, resulting image will be flipped around a horizontal axis
2 - mm_leftright, resulting image will be flipped around a vertical axis
3 - resulting image will be flipped around a horizontal axis and a vertical axis

List of Open Items:

1. Test and refine code for detection of how far the trap is from the center of the frame

There is currently code that can detect the trap using the IR leds and a red trap. This code can output a radius from the trap to the center of the frame. We did not get as far as we wanted with testing it to see how accurate it is. This code can be found under the Colored Blob Detect folder. There is an additional element that was added to the code that we were just starting to test, which was when the UAV came close to the trap and was out of view of the cameras. There are some markers on the outer edges that helped us orient ourselves.

2. Make this data usable and relevant

Currently the data is just being printed out to us and shown through a GUI but we need to convert it to a distance or a velocity. We were figuring out how it to use it via mavros and whether it would be best to publish the data or use another method.

3. Control the UAV movement/position based on data through mavros commands

There are mavros plugins available to us that could be potentially useful in controlling the UAV based on the camera data. We were looking at the position and velocity plugins and weighing which one would be easier for use to provide more accurate control.

4. Adjust parameters on camera i.e. b_gain, r_gain, gain etc to provide better video

We were still having a bit of trouble getting the color camera to produce video with realistic color. The parameters may be a bit off and it differs greatly from indoors to outdoors. Need to adjust and record various setting for very sunny days at midday, afternoon and overcast days. The parameters i.e. exposure time will vary with these factors.

Requirements:

1. Locate Trap Precisely : Detection and Tracking of Payload

code for detection and tracking is on github. a few refinements may be needed for tracking but otherwise complete

2. Locate Trap Precisely : Communication of sensing via mavros

In progress. using mavros plugins to communicate with pixhawk