Drive GUI

Note: pilot does NOT use this GUI. The Oculus and associated hardware is connected here but the person using this GUI is just monitoring drive systems.

contains options to save/load driver config.

File | About

Live Drive Data

PID goal: xxx m/s

FL:
speed: X m/s
slip:
: XXXX ma
stall:

virtual current graph

FR:
speed: X m/s
slip:
: XXXX ma
stall:

example of high current/stall warning.

stuck:

occurs if slip on all wheels.

BL:
speed: X m/s
slip:
: XXXX ma
stall:

BR:
speed: X m/s
slip:
: XXXX ma
stall:

Oculus preview

Oculus connection:

grow from approaching frame edge

Oculus Orientation data
Pitch:
YAW:
Roll:

Oculus distortion
param 1:
param 2:
param 3:

Inclinometer

Pitch:

Roll:

Warn:

warn:

Outgoing commands

->

LIVE

Pilot Preferences

Config: XXXXXX
Save config
Load config

left dead zone:
min: #
max: #

left sensitivity:
-4x -2x 1x 2x 4x

right dead zone:
min: #
max: #

right sensitivity:
-4x -2x 1x 2x 4x

Top speed:
0% 50% 100%

HUD

Connection to primary program

OK

Notes:

1. Oculus Preview

- The distorted images do not need (though it'd be cool if they did) to display the actual Oculus video feed. They simply show a visual representation of the drive-vision systems speed buffering system. As the driver turns their head and approaches the edge of the currently available frame, a glow forms at the periphery. If they try to move to a location the vision system cannot ever reach (i.e. straight up/down) a solid mass instead of a glow forms.
- The strange arrow with a tail represents the direction a suspected rock is. This data would come from logistics and possibly (depending on if we have logistics confirm CV reported rocks or if CV can directly send locations) come from the CV object recognition system.
- The distortion parameters are also saved when the config is saved. This allows for pilots with different preferences as well as, potentially, the use of an HMD other than the rift with different distortion parameters.

2. Pilot Preferences

- The LIVE button, when clicked, changes the pilot preferences area to a live graphical display of the controller. The LIVE button changes to Preferences and can be used to switch back.

3. Live Drive Data

- The primary purpose of this area is to ensure the pilot has control and that the drive system is responding. It should also help catch strange behavior in the drive system (low/no current, stalls, etc).
- If we can detect slip, we would be able to detect things like being stuck/high centered.

Engineering GUI

Error Stack

Class A	Class B	warnings/advisory
error/description		
error/description		

Rover Commands

Drive reset
blah
blah
blah

LAN data/Commands

This IP: xxx.xxx.x.xxx
Primary IP: xxx.xxx.x.xxx
Drive IP: xxx.xxx.x.xxx
Arm IP: xxx.xxx.x.xxx
Logistics IP: xxx.xxx.x.xxx
Rover IP: xxx.xxx.x.xxx
Video IP 1: xxx.xxx.x.xxx
Video IP 2: xxx.xxx.x.xxx
etc...

set
set
set
set
set

Pops up a message box asking for a new IP.

Selected Error Info

item removed when clicked. Could also launch event or perform associated delegate method. I.e. give logistics permissions to use LTE not WiMax...

Name	Class	source/system	? resolved?

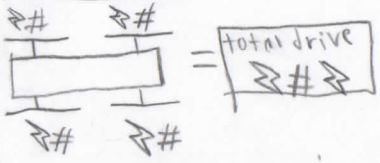
Description

Network Commands

reset stuff...

Rover data

Drive Power:



Other systems power:

blah #
blah #
blah #

Total Power usage:

#

Battery:

Network data

LTE
WiMAX

LTE
speed: xxx Mbps
strength:
Load: xxx%
Public IP: 111.111.1.111

WiMAX
speed: xxx Mbps
strength:
Load: xxx%
Public IP: 111.111.1.111

Video Status

Oculus feed:
Available
Connected

HD feed(s):
Available
Connected

Other Cams:
Cam 1
Cam 2
Cam 3
etc...

Manual Telnet

IP: xxx.xxx.x.xxx:xxxx connect

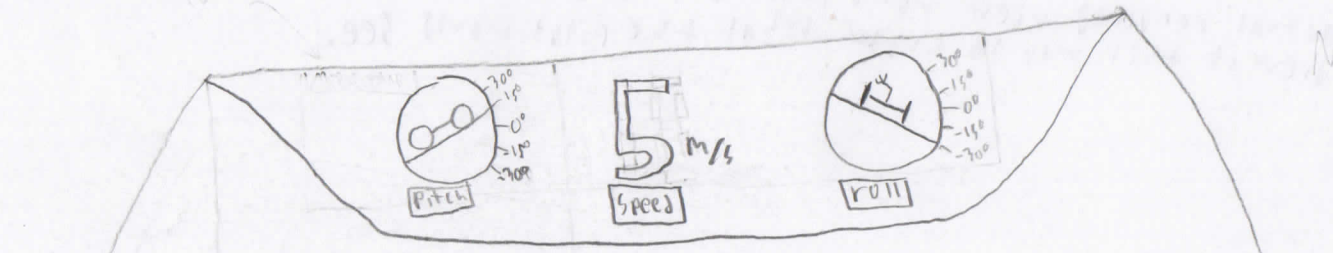
->

Input: send

Connection to primary program

OK

Oculus view



MAP

use bing maps API...

NE 42°

compass heading

Arrow used to point towards rocks pinned by logiitics. Arrow could be the color of the suspected rock.

Notes on back!!!

Notes:

General:

- The overlays can be controlled via the controller. For example, the map could be commanded to pop-up/vanish with the start button.
- The actual rendered view would involve two barrel distorted images, each taking half of the screen. I just drew it this way to show what the pilot would see.

Drive

Primary ~~Program~~ GUI

ARM

Logistics

MAP

Use bing map API...

Searchers connected: XX

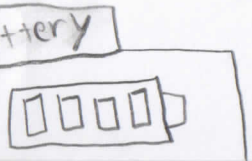
Engineering

Notices

Info

Primary (this) IP: XXX.XXX.X.XXX

Battery



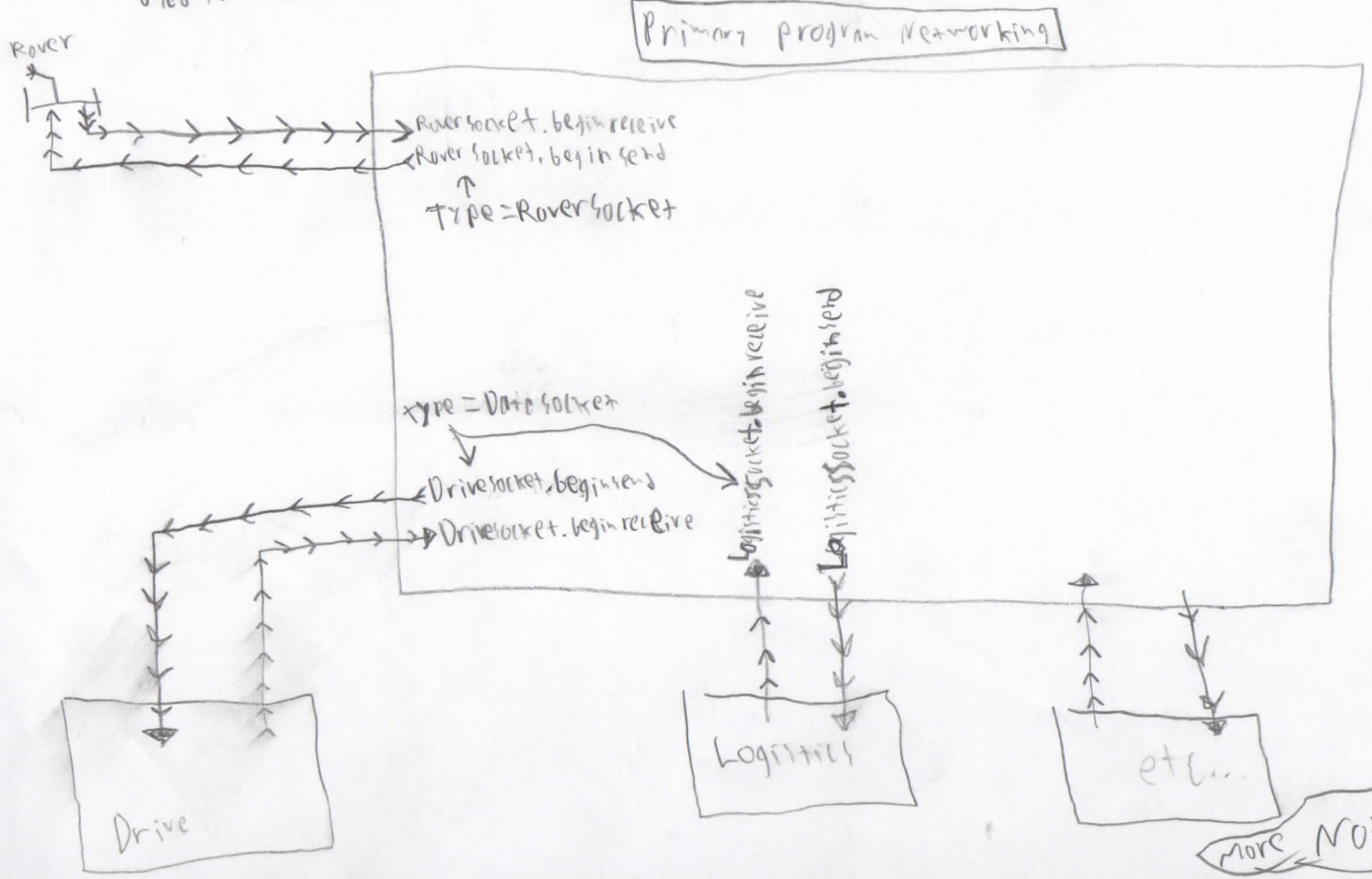
1655

827.5

Mission Control Sockets

Types

1. Data Socket:
   - Used to move relevant data from 1 station to another. The station that wants to send the data sends it to the primary program. Like a string socket in the boggle server the primary program's begin receive, running on a unique socket for each system (sort of like a match in boggle?), would be able to tell where the message came from and where it is trying to go (destination would be in the message, just like WORD/PLAY/STOP commands in boggle). The command program would then send the data to the appropriate station. It is also possible the data could be intended for the primary program itself, such as a command from engineering. The data could also have the rover as the destination
2. Rover Socket:
   - Used to talk to/receive from the rover. Should be optimized for low bandwidth. Other than that IDK yet...



Note: In boggle whenever two connections are established, in "boggle server" they are put into a match. This would be similar in that there would be a server but there are 2 differences. The first is that the server would not simply accept any connection, they would need to be checked for security (maybe only accept from a list of IPs?) Then, instead of being put into a new match, connections would be added to an already existing object that would basically "be" the primary program.

more Notes on back →

- Data socket:
  - Constructor could require address of primary program [which would be asked for at each systems' startup and displayed by the primary program].

Logistics GUI

Capture Controls

Cam set 1 Cam set 2
Cam set 3 Cam set 4 } Settings

Size (% of available):

0% 50% 100%

Destination:

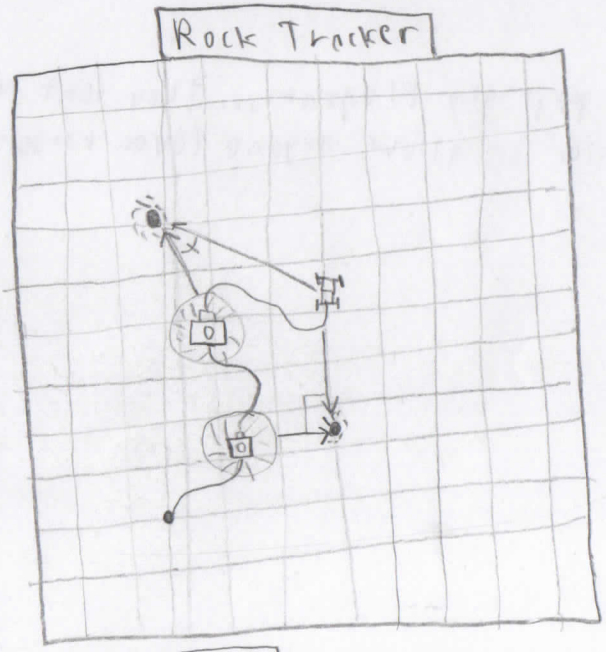
Searcher 1 ← drop down showing available searchers

Source:

Cam set 3 ← drop down showing available camera sources

Capture & Send!

Rock Tracker



Network Data

LTE
WiMAX

LTE

Speed: XXX mbps
Strength:
Load: XXX%
Public IP: III.III.I.III

WiMAX

Same as LTE

Capture Queue

Cam set 2 searchers | Captured retrieved | Size: 1.5MB | Cancel ← example capture on queue...

CV Queue

Cam set 3 | Size: 2MB | Color of rock: Red | Send to: Searcher 3 | Ignore ← example CV object recognized awaiting review...

MAP

Use bing maps API... It looks easy/pretty

GPS

Lock
Lat:
Long:
Accuracy:

CV variables

Sensitivity: 0 100

Run object recognition On:

- [x] cam set 1
- [x] cam set 2
- [x] cam set 3
- [ ] cam set 4

Logistics IP: XXX.XXX.X.XXX ← This IP can be told to potential searchers so they can become searchers.

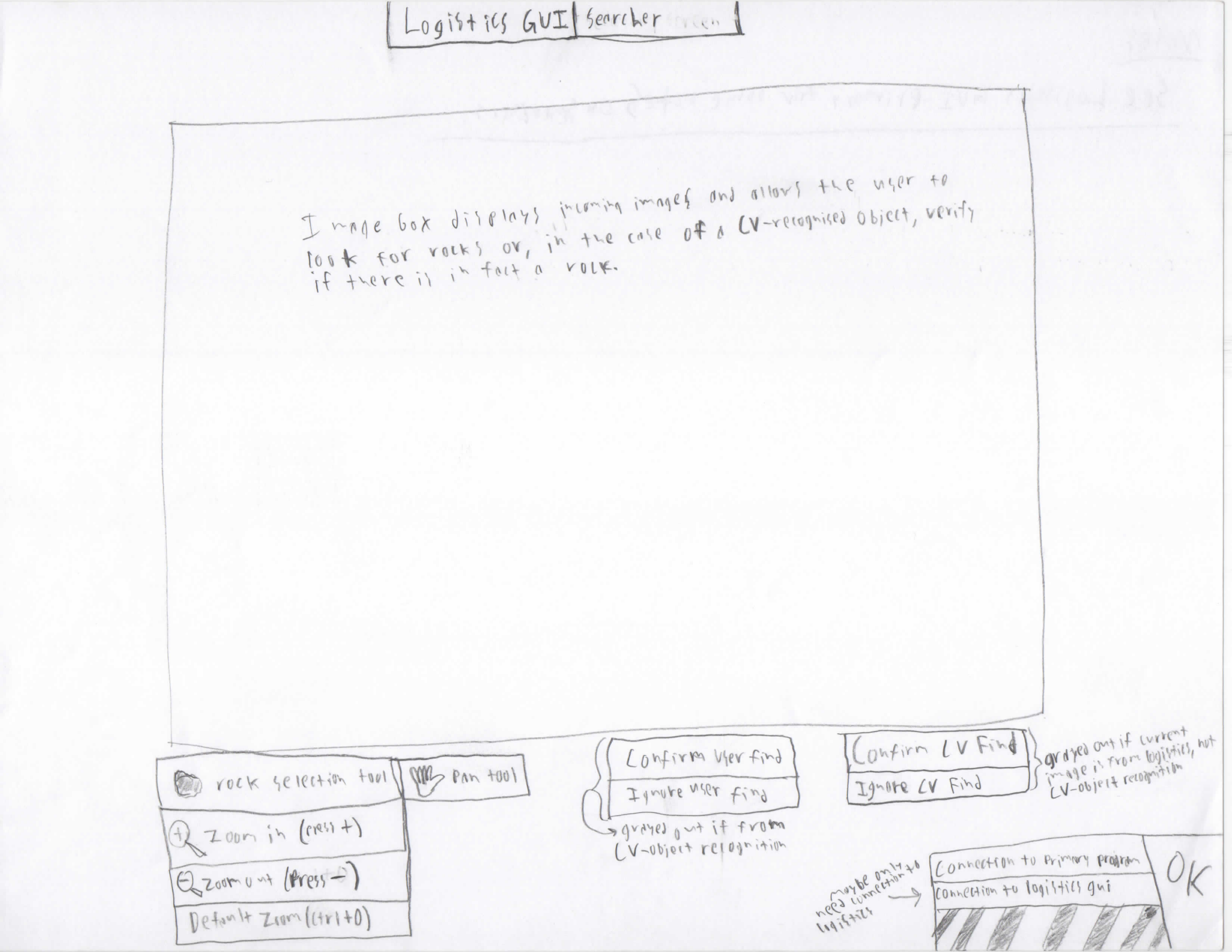
Connection to primary program

OK

Notes:

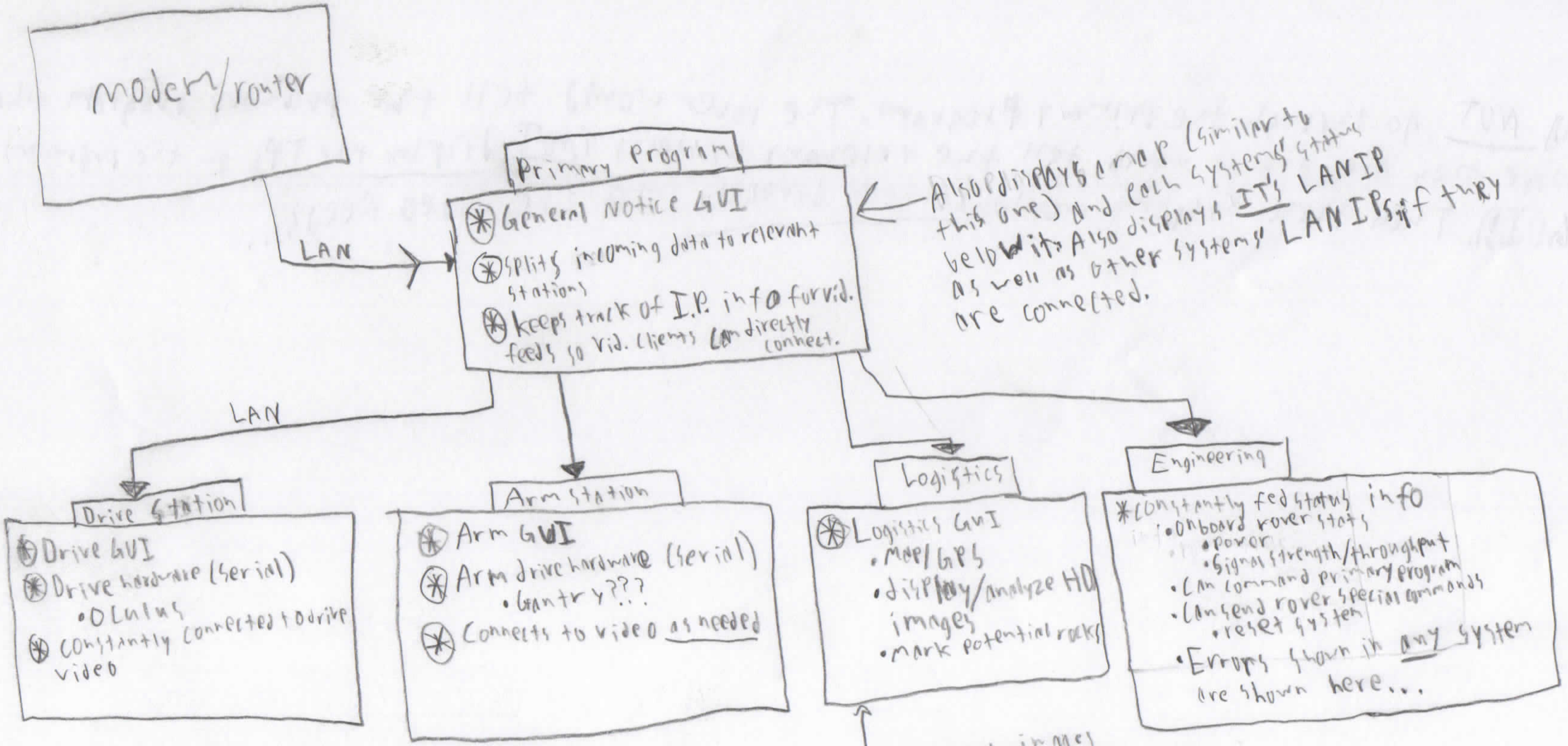
Researchers:

- researchers do NOT go through the primary program... they dont need high security and letting logistics manage them will make it easier to allow anyone (even random people with a laptop) to join as researchers.


Logistics GUI
Searcher
Image box displays incoming images and allows the user to look for rocks or, in the case of a CV-recognised object, verify if there is in fact a rock.
rock selection tool
Pan tool
Zoom in (press +)
Zoom out (press −)
Default Zoom (ctrl+0)
Confirm User find
Ignore user find
grayed out if from CV-object recognition
Confirm CV Find
Ignore CV Find
grayed out if current image is from logistics, not CV-object recognition
maybe only to need connection to logistics
Connection to Primary program
Connection to logistics gui
OK

Notes

See Logistics GUI-Primary for some notes on searchers.



Would search images for rocks and report their positions relative to the robot when they found them. I.E. they can "click" a diagram saying approximately where the rock was. (12 o'clock, 3 o'clock, etc). They could also review rocks reported by the CV system. They can also remove reported rocks from the list when they are investigated/retrieved. This system would also contain the rock tracking system, which is basically a program that knows where the robot was when an HD image was taken then keeps track of the robot (encoders & GPS). This way when a rock is reported its position relative to the rover's current position is known. To do this with high precision we would need RGBD data on all cameras... Is it possible to have 2 camera mounts everywhere? for depth data. The OpenCL software can handle 64!

TODO:

- draw Oculus output
  - mini-map?
  - inclinometer(s)?
  - speed data?
  - potential rock locations
- draw Arm Gui

Notes on back

Notes:

Video:

- video would NOT go through the primary program. The rover would tell the primary program the IPs of where they are and it would tell the relevant systems (NOT display the IPs on the primary program's GUI). Then these systems would connect directly to the video feeds...