Mars Rover Experience
2018.08-
James H Phelan
Project Lead
Houston Robotics
renamed as of 4/'19 USAi Labs

Attribute	**Value [imperial]**	**Value [SI]**
Weight	25 [lbs]	11.34[kg]
Footprint	24x14 [in]	60.96x30.48 [cm]
Battery Capacity	5200 [mAh]	5200 [mAh]
Battery Discharge Rate	8 [A]	8 [A]
Nominal Current Draw	1.2 [A]	1.2 [A]
Operating time	5 [hrs] (continual use)	5 [hrs] (continual use)
Approximate Max speed	9.7 [in/s]	24.6 [cm/s]
Maximum 90 deg vertical scale	12 [in]	30.48 [cm]
Maximum height differential	14 [in]	35.56 [cm]
Communication (in this guide)	Bluetooth app (Android only) and Xbox Controller	
Cost	~ $2,500	
Individual parts	955+	
Line items	100+	
Vendors	13+	

This is, initially, a thrown-together collection of other Mars Rover comment files. Maybe eventually it may be made all pretty. Don't hold your breath!

From Mars Rover Timeline.rtf:

Mars Rover Timeline

2014.07.31 NASA/JPL announces plans for open source Mars rover.
https://scienceandtechnology.jpl.nasa.gov/build-your-own-rover

Introduction of NASA/JPL OpenSource plans at lunch
2018.08.11
Charles DeMontaigu
Roberto Pensotti
James Phelan
Carroll Vance
Diane Brown
Raymond Brown
Harsh Bhasin

GroupMe discussions
2018.08.11-25

Group Members
Link to NASA/JPL sources
Review of NASA/JPL documentation
2018.08.11-25
Group Members
Commit to project & funding
2018.08.25
$2500, $250/u
James Phelan 10-Xu
Roberto Pensotti 3u + in kind
Charles DeMontaigu 1u
Harsh Bhasin
Chuck Rosser
Review JPL Bill of Materials in prep for ordering
2018.08.25
James Phelan
Roberto Pensotti

Order Parts
2018.08.26
James Phelan
X-box deferred as of uncertain need
Thin-wall short Al tubing deferred to use add'l thicker wall tubing cut to size.
Raspberry Pi deferred, to be supplied by Roberto Pensotti
40-pin header deferred as no reference in specs
4-pin headers ordered per JPL BOM. (See error on BOM below)
Ordered from H.R. BOM through Digi-Key
2018.08.27
James Phelan
Ordered wheels from Amazon (JPL source sold out)
2018.08.29
James Phelan
Finished ordering parts
1/2" Bore, Face Thru-Hole Pillow Block on backorder from both:
https://www.servocity.com/0-500-1-2-bore-flat-bearing-mount
https://www.studica.com/us/en/RobotZone/1-4-bore-flat-bearing-mount.html

Order Printed Circuit Boards

2018.08.25

James Phelan, email OSH Park, receive ticket #

2018.08.27D

James Phelan,
Reply from OSH Park; instructions followed;
Order placed

2018.09.05

Order shipped USPS free shipping

Receive parts & inventory

2018.08.29 Delivery from Amazon 1,2/3

2018.08.30 Delivery from Amazon 3/3
Delivery from Digi-Key

James Phelan
Discovered when Digi-Key order arrived that master parts list called for 0.05" 4-headers for no apparent reason when 0.1" are on PCBs. Re-ordered correct headers. Not worth postage to return wrong headers.

2018.08.31 1" x 2' PVC Lowes
Delivery from ServoCity
Delivery from Adafruit

2018.09.09

Haven't kept up with multiple deliveries, sometimes 2 a day
Inventory maintained with each delivery, however.
All parts received EXCEPT: PCB, steering linkage, pillow blocks
Custom acrylic back plate needs laser cutting
3D printed head may need to be re-done to be in 1 color
Currently ½ red, ½ black (random filaments in printers)

Print 3D parts

Work w/ Roland HCC Stafford

2018.09.01

Initiated 3D printing

2018.09.08

Picked up 3D printed parts
Head is red/black, re-do?
Encoder mounts are bright yellow and look good

Print project documents for subassemblies

2018.09.08 done

FabLab color printer
2 sets – 1 me, 1 Roberto
Documents collated & placed in binder

Divide parts into subassembly kits

2018.09.08

Made attempt at this this evening and realized it was *overwhelming* for one person. Gathered zipper bags.

Put announcement in AI/ML44&IOT for Part Picking Party next Sat

2018.09.15

HCC/Houston Robotics Part Picking Party - many club members

Body assembly Quy & Derrick

Head assembly by Ziad

Al tubing cut by Duc

Wheel hub clamp groove extended by Duc

Body hole for differential drilled by Duc

[Off-center vs too small. Needs re-drilling

Done by Roland HCC FabLab 10/6/18 w/ Dremel tool]

NOTE FOR FUTURE:

label parts with their project reference

for ease of picking and assembly, i.e. “T11"

Build subassemblies

2018.09.22

First Assembly Party

Head - completed by Ziad

Body - completed by Quy

Rocker-Bogey - started by Derrick & Alfredo. Problems noted

Wheels - drilled by Omar w/ help of Fab-Lab adapter.

Custom acrylic end plate & pattern front plate potential fit / hole mis-match noted

PCBs populated & soldered by Roberto & testing begun

2018.09.29

Second assembly party

2018.10.06

Thirs assembly party

Rocker-bogies completed using stand-offs shortened to 3/16"

with help of Roland at HCC FabLab

Box of parts left behind but rescued by Usha Bhasin!

Testing procedure problems noted on Logic Shifter &

Voltage Divider PCBs, Roberto & Jim. Corrected & posted.

Wiring task taken home, Roberto.

2018.10.09

Differential Pivot assembled - James

2018.10.11

Corner Wheel Assembly

Roberto Pensotti & James Phelan at “Laboratorio Roberto” at his home

Roberto worked on corner wheel assembly without 0.25" bearing on back-order

James worked on body fit issues

Main shaft has to be worked under wiring on main electronics board

This was not mentioned in build for electronics board, but it is clear components were placed to allow room for this.
Corner brackets conflict with mounting for end panels not allowing inset.
Corner brackets moved in one large hole.
Electronics board redrilled to accommodate.
End plates mounted successfully.

2018.10.12

While not nearly complete, hoped to be able to display rover at Maker Faire at the GRB on 2018.10.13
0.5" Al rods are too wide to fit rocker-bogie & differential
but will fit hole in Al pattern plate & channels.
Bearings moved to neighboring holes to allow shafts to penetrate rocker-bogies
Wheel assemblies mounted to rocker-bogies - James
Rocker-bogies slid onto shaft but not mounted as box of parts left behind on

2018.10.12-13

Maker Faire at GRB. Presented Houston Robotics, the Mars Rover & our many robots (Ariel, Sonic, SneakerBot, GoPiGo, EduBot, Mars Rover, Poppy,...)
at a booth hosted by HCC Stafford Workforce Center
Generated a lot of interest in Houston Robotics

2018.10.20

Fourth assembly party
With help of Roland @ HCC FabLab & Omar at the lathe
& Alfredo disassembling the Differential Bar
Jim was able to lathe Al rods to w/in the 0.500" tolerance to fit the bearings.
Moved the rocker-bogey bearings back to home holes & fit rods together
still needed “shoebox of parts” to fully assemble the suspension.

2018.10.21

Jim rende-voused with Usha to get the “shoebox of parts”
Hard stop spacers mounted on corner steering assemblies.

2018.10.22

Much of above chronology documented with help of photo dates.

2018.10.23

Attempted to mount the lathed to fit main axle to rocker-bogies. The main axle is 15". The body is about 9" wide leaving 6" to stick out or about 3" on a side. We’re not talking rocket science here. Oh, wait, yes we are. So subtract 1/16" on each side for the width of the side panels. Mark off 3" less 1/6" from each end with a marker. Stick the axle through the body at the designated hole & **snake under the wires (there is no mention of this in the electronics board build nor the mechanical integraton!)** Center using your markings to stick out evenly on both sides. **I’m surprised NASA didn’t put those W3 slippery washers between the clamping collars and the body & rocker-bogies at this stage.** Slip the

innermost S22 0.5" collar clamp onto the axle on both sides and clamp just firm enough against the body to keep it from slipping. Tighten fully later when all assembled.

Slip the middle collar clamp on both sides. Slip the rocker-bogie assembly on each side being careful to gently align and penetrate both inner and outer bearings
NOTE: It is a VERY close fit through the bearings. If they are tightened down before the axle is inserted, it will almost certainly mis-align! Given the very tight working space in the U-channels, it is very difficult to grab the locking hex nuts. Also, given the tiny hex socket of the #6-23 x 1/4" B1 or 3/8" B2 button head screws it is VERY easy to strip out the socket unless the hex driver is well secured, and even then. The build calls for #180 locking hex nuts. With buying #200 we should have #20 left over. NASA BOM lists #300. The nuts are very easy to drop and roll away. Now I understand why they ordered #100 extra! Same for screws! Solution to the cramped work space - put the screws in from inside the U-channel wedged on the short arm of a hex key. Apply the lock nuts with a (manual) hex driver from the outside. Leave them loose until the axle is all the way through, then tighten. I used a spare piece of Al rod or the vertical Differential Bar rods to act as proxy for the axle & tightened the bearings in place under less awkward circumstances than wrestling with the body assembly. Then attempted to fit the main axle. Finally place the outermost clamping collar next to the rocker-bogie. I made the outer surface of the outermost collar flush with the end of the axle & tightened firmly. Snugging the rocker-bogie against it, I then placed the middle collar against the rocker-bogie & tightened. Lastly, being sure the body was centered, tightened the innermost collars against the body. Be sure the bearings, clamps and collars are firmly tightened all around.

2018.10.24

Re-positioning the axle. Comparing our rover to the build diagrams, I noticed I'd placed the axle two holes too far back on the rocker-bogies. I was counting holes from the wrong end! Wondered why the back of the body was hitting the rear corner wheel assemblies! Spent this evening redoing the above process but in the correct holes! (5 back from the elbow in front of it, 7 forward from the junction with the corner wheel assembly behind it.) Much better!

Differential bar turnbuckle and vertical post alignment. There is no mention in the build docs about leaving clearance between the main axle & the vertical differential bar posts. I left a mm or two to avoid friction. Also no mention of whether or how to adjust the turnbuckles. I just tightened them all the way.

Wheel treads. Someone (Omar, Dutch, ?) noticed at the last Meetup that

the tire treads weren't all pointing the same direction. Hadn't really thought about it before and dismissed it for the time. Tonight I noted 4 of the tires treaded forward and 2, 1 on either side, treaded reverse. Of course Traxxas would sell wheels in matched pairs so than when mounted on opposite sides of a vehicle they would tread the same direction. I swapped the two offending wheels to the opposite sides and all was right with the world!

Robot alive warning & emergency stop button. I believe, and I believe industry does too, that every robot capable of moving needs warning lights & sounds when it is active. The rover has no such feature (but then it was to be alone on Mars, what's the worry?) I also believe every moving robot needs an emergency stop big red button. Ditto.

2018.10.29

Replacement neck collar clamp, replacement & extra wheel hub clamps & extra B1 #6-32 x 14" button head screws and extra B11 #6-32 lock nuts arrive & installed where required. Now the only missing pieces are the 0.25" pillow bearing blocks. They're promised next week but don't trust that. Roberto is going to work with Roland/Lemme/Fred at HCC Stafford FabLab. Today Fred say's he's finished 10 prototypes in ABS. Roberto should have the bearings from Amazon tomorrow.

2018.10.31

Bearings OD 0.6" instead of 0.5". Fred redesigning to fit. Then to print in carbon fiber with thicker arms than steel ones for added strength.

Acrylic back plane redesigned with

a) smaller better fitting Rasbperry Pi window lower edge raised
b) hole for power switch (top & sides are awkward & wiring path odd)
c) holes for battery connectors (they won't fit through 0.5" pattern plate holes & would have to be cut & resoldered leaving battery stuck in body.)

2018.11.01

Head 16x32 LED Display Testing

The Adafruit project mentioned in the build is *obsolete* and not maintained.

LED display needs a **ribbon cable** to an appropriate connector on the Logic Shifter to avoid a "spaghetti farm" of jumper cables!

Will fall back to NASA/JPL display code.

Installing their code per Software Steps:

```
$ sudo apt-get install python-bluez
$ sudo apt-get install python-pip python-dev ipython
$ sudo apt-get install bluetooth libbluetooth-dev
$ sudo pip install pybluez
```

Changed $ sudo pip install pybluez to $ sudo pip**3** install pybluez

```
$ sudo systemctl start bluetooth

$ sudo nano /lib/systemd/system/bluetooth.service
```

Add "-C" after 'bluetoothd' on line 9

2.1.1 Naming the Bluetooth Device

[THIS ISN'T NECESSARY IF HOSTNAME IS CHANGED TO ROVER NAME "Fidelity"

bluetooth recognizes hostname "Fidelity" and it appears on other devices when bluetooth enabled & discoverable]

There is LED display python code under ~/osr/led:

Display.py Feature.py screen.py Updater.py

will just have to explore these to see what they do.

Meanwhile, figure out how to wire the LED display using the Logic Shifter....

Electrical Build - The LED Display

The 16x32 LED Display has a 8x2=16 pin keyed IDC connector included. Why didn't NASA/JPL make use of this in designing their Logic Shifter PCB and just plug the display side into that instead of this "spaghetti farm" of jumpers? Given their pics & description and the out-of-date Adafruit project, I'd say they just have an old version of the display. Or maybe the PCB designer knew nothing about it, just the connections necessary to the Pi?

I'll try to pick one up at EPO/Fry's/MicroCenter Saturday else order one from DigiKey/Jameco.

The neck wiring from Pi - Logic Shifter needs simmilar clean-up. A couple 10-conductor rainbow cables (stripping 2 conductors off each) & 4x 4x1 female pin connectors should do the job and fit through the opening in the head & 0.5" hole in body. Split them neatly in columns P1 + P3 on one cable and P2 + P4 on the other. Split out the spaghetti on the Pi side with individual pin headers.

2018.11.03

.25" pillow bearing blocks having been on back-order since the beginning of the project led us to try making our own. Chuck said it would take him a couple weeks to make them. Roberto ordered the bearing from Amazon:

https://www.amazon.com/FR4-ZZ-Flanged-Radial-Bearing-10pack/dp/B06X19Z696/ref=sr_1_2?ie=UTF8&qid=1539639434&sr=8-2&keywords=.25%22+ball+bearing+flange

and along with Frederic & Roland at the HCC FabLab 3D printed them in chopped carbon fiber.

Turns out the original part became available & arrived the day before the MeetUp. We decided to try our printed version to see how they'd do. Discovered that the 3D printed version is slightly thicker than the original part due to a recessed flange in the original part. This extra thickness caused the S23 4mm to 0.25" clamping shaft coupler to rise enough in the space for the clamping screw portion to strike the button heads above it. Possible solutions: 1) revert back to the original parts now in hand 2) reprint the blocks a little thinner or with a recess for the flange or 3) flip the pillow blocks over, so the thick part went down into the hole below. Solution 3 had one concern - the bearing could slip

down into the space below and would be difficult to access and correct. Solution - flip the other pillow block below the 3D printed Encoder Mounts so they fit face-to-face instead of back-to-back. They fit perfectly so that the bottom bearing kept the top bearing from falling down. While their being closer together could, theoretically, allow for more lateral wobble, the shaft seemed to fit perfectly without problem. Time will tell.

During this final Corner Steering assebbly, we noted one of the steering motors (left front) did not want to smoothly move passively, but rather stick. It seemed to run OK when powered but we are concerned the gear train may be compromised. We may have to replace it & get a spare also.

Despite the clever height staggering of the RoboClaw motor controllers, they are largely inaccessible when in the rover. Roberto will have to unscrew them one by one to test them or take the Electronics Board out of the rover for testing.

2018.11.04

Working on the LED display. The sample project quoted by NASA/JPL and Adafruit is no longer supported. I was unable to load it from Github although I may try again later as the rover's Pi is with Roberto this weekend as he works the power train & calibrates the motors. I did match up the LED Display input pins to the Logic Shifter Board output pins. What a "spaghetti farm"! See the NASA/JPL Github issue #59 below. I got the last 8x2 IDC connector from Fry's. I took 2 pretty 10-conductor rainbow ribbon cables, stripped 2 conductors off each, and clamped them into the IDC. I then soldered & clamped (wire too thin to clamp alone) a female pin connector to the other end of each conductor, avoiding putting them in housings until I saw how the pin assignments shook out. What a mess! What was NASA/JPL thinking when they made the Logic Shifter Board??

Electrical Wiring

2018.12.04 Team spent October finalizing the mechanical assembly once the pillow blocks arrived/were 3D printed. Then Jim & Roberto spent November getting the wiring accomplished. Much ambivalence surrounded the fact that there were 2 kinds of motor/encoder sets to wire: 1) the 4 Corner Steering Motors + Absolute Encoders and 2) the 6 drive motors with integrated encoders that had a 6 position female DuPont connector. The 30 AWG wires specified by NASA/JPL were fragile and near impossible to crimp into a DuPont connector even doubled over and broke easily. Thought was given to scrapping DuPont connectors in favor of RJ45 connectors and CAT5/6 cable. But then it was silly to re-wire the existing female DuPont connectors on the drive motors. The final solution was to use CAT5/6 cable to all the motors. As per Power over Ethernet protocol, 1 twisted pair (blue/blue-white) was dedicated to positive motor power, the other (brown/brown-white) to negative motor power. The other 4 (or 3) wires went to the encoders.

For the drive motor ends of the cable we needed 6 conductor DuPont connectors to mate with the female connectors on the motors. For power, the 2 wires had to be stripped longer than the crimp connector to allow 1 wire to penetrate the connector while the other wrapped around it near enough to be gripped by the insulation crimp,

leaving clearance so the insulation wouldn't prevent inserting the connector into the housing, but not so much as to cause a short. The 2 power pairs were inserted into a 2 place housing. Heat-shrink tubing ensured insulation when needed. The 4 encoder wires went into a 4 place housing. This allowed the two housings to fit through the ½" holes in the Al pattern plate which would not be the case if it were a single 6 place housing. They were bundled together using heat-shrink tubing for stability but which could easily be replaced if the two had to be separated.

For the corner motors 2 CAT5 wires were soldered to each lug (4 total) on the motor and reinforced with heat-shrink tubing. If to do over, would use quick-connect slip-on connectors instead of soldering to the lugs. The lugs are also fragile and one motor was almost lost because the lug broke off. Fortunately, enough of the plastic base could be scraped away to make a solder connection. Since the rover often has to be turned on its back for internal wiring, the solder lugs sticking up from the corner steering motors are prone to snagging or breaking. A temporary solution of Meccano cylinders protected them. The permanent solution was to attach 1.5" Al channels with channel connector plates around the motors for protection. A free-hanging RJ45 jack accepted the 2 pairs of power wires & the 3 encoder wires in its punch-down block. The jack was secured with cable ties in the channel. A CAT5/6 patch cable with RJ45 connectors on each end was cut in half. The plug end inserted in the jack by the motors. The raw end passed through the rocker-bogie legs to a hole in the body. The cable was secured with cable ties and given enough slack to accommodate movement of the rocker-bogies. The CAT5/6 cables were too wide all to fit through one body hole, so 2 were used - 1 for the 2 corner motors and 1 for the 3 drive motors. Different color cables were used for each right/left motor pair to aid in identification at the Electronics Board end and for esthetics.

At the Electronics Board end, the power pairs were simply screwed into the RoboClaws terminals. The encoder wires received single DuPont connectors to plug onto either the RoboClaw directly or the Voltage Divider board as appropriate.

Calibration
Name of the calibration program has changed to Basicicro Motion Studio

Programming & communication
Install a fresh Raspbian Stretch image from RaspberryPi.org
Configure to locale and taste.
Gitclone the NASA/JPL code per their directions.

```
$ cd /home/pi
$ git clone https://github.com/nasa-jpl/osr-rover-code osr
```

Update and Upgrade again
no updates since original install
Install packages:

```
$ sudo apt-get install python-bluez
$ sudo apt-get install python-pip python-dev ipython    (should this be python3- ?)
```

$ sudo apt-get install bluetooth libbluetooth-dev
$ sudo pip install pybluez
$ sudo systemctl start bluetooth

$ sudo nano /lib/systemd/system/bluetooth.service
    Add "-C" after 'bluetoothd' on line 9

$ sudo reboot

$ sudo sdptool add SP

2.2 Init Script

$ sudo nano /etc/rc.local
    sudo python /home/pi/osr/led/screen.py &
    sleep 10
    sudo python /home/pi/osr/rover/main.py -s **-x** &

$ sudo raspi-config
    - Interface Options .. > Serial
    - Would you like a login shell to be accessible over serial? .. > No
    - Would you like the serial port hardware to be enabled? .. > Yes
    - Would you like to reboot now? .. > Yes

$ ls -l /dev/serial*
    serial0 -> ttyS0

$ sudo nano /boot/cmdline.txt
    "console = ..." to read "console=tty1" (it already does...)

$ sudo reboot

3 Test files
3.1 Serial Test
use a jumper wire to connect the GPIO14 and GPIO15 pins (physical pins 8 &10)
$ python /home/pi/osr/rover/serialTest.py
    ...line 43 in main
    clientThread = threading.Thread(target=mythread.SendData, args=())
AttributeError: Threads instance has no attribute 'SendData'

Try:
$ python**3** /home/pi/osr/rover/serialTest.py
    ...line 29
    print 'Successful data received over serial!'
    SyntaxError: Missing parentheses in call to 'print'
but...fixing print () just goes back to above SendData error.

Stick with Python2

Github reports same error.
However, this script from the Mars Rover forum:
https://www.tapatalk.com/groups/jpl_opensource_rover/index.php
https://www.tapatalk.com/groups/jpl_opensource_rover/serialtest-py-error-t22.html

does work:

```
import time
import serial
ser = serial.Serial(
        port='/dev/ttyS0',
        baudrate=9600,
        parity=serial.PARITY_NONE,
        stopbits=serial.STOPBITS_ONE,
        bytesize=serial.EIGHTBITS,
        timeout=1
        )
counter = 0;
while 1:
        output = 'Output: ' + str(counter)
        print (output)
        ser.write(output.encode('utf-8'))
        time.sleep(1)
        counter += 1
        input = ser.readline()
        print (input)
```

output:

```
Output: 0
b ‘Output: 0'
Output: 1
b ‘Output: 1'
Output: 2
b ‘Output: 2'
Output: 3
b ‘Output: 3'
...
```

2018.12.08
See github calibration issue.
Roberto completed wiring of the drive motor encoders to the RoboClaws but had issues during calibration. See github Issue about the change in the calibration software name. At the meeting we worked on RoboClaws #1 and #2. Corrected some polarities. The

Logic Battery displayed on Motion Studio was 2.7V where it should be 5V. **We are still unclear as to why.** By changing the alarm setting to allow a Logic Battery of 2.7V the testing was able to proceed.
The PWM function worked ok allowing us to run the 4 associated motors forwards and backwards. However, at the Velocity settings, looking at the M1/M2 Speed boxes the two motors had widely different 8,000 vs 29,000 range and rapidly changing values. When out best estimate as to the speed at max PWM was plugged into the QPPS box and the Tune M1 box was clicked, the attached motor would slowly rotate back and forth a couple times as expected, then continue in the same direction forever until Motion Studio was ended by Task Manager and the motor stopped by the kill switch. *Next step is to investigate the Logic Battery voltage issue to see what effect that may have on this error.*

2018.12.11

See github LED Display Issue. The LED directions and sources were out of date and test programs unsuccessful. To the sure that the "spaghetti farm" of jumper wires from the LED display to the Logic Shifter to the Raspberry Pi wasn't at fault, wire management was addressed. On the custom ribbon cable from the display to the Logic Shifter the loose female connectors were grouped into three 4-position DuPont housings for P5, P6 and P7. The 4 ground wires combined into 1 connector of a 2-position DuPont housing for the Ground pin and its inert companion. (Why they didn't provide a 4-position mutually grounded header for P8 is beyond me.) The continuity of the connections were then painstakingly traced from the display input connector to the Logic Shifter output pins. One reversal on P5 of OE and LAT was discovered and corrected. The connections from the input pins of the Logic Shifter to the Raspberry Pi GPIO pins were similarly traced and adjusted if necessary. Reserving adjacent GPIO pins 4 & 6 for the 5v regulator pair, the Logic Shifter 5V and GND were relegated to pins 2 and 14. *Now to follow the rabbit of the LED software for testing.*

2018.12.18
RoboClaw Logic Battery voltage issue SOLVED. The connector from the 5V regulator to the Serial Splitter board was disconnected in the "spaghetti." Per Eric Junkins of JPL the RoboClaw has its own voltage regulator that provided the 2.7V measured. Replacing the connector brought the Logic Battery voltage to 4.7V.

However, we're getting erratic readings from the Motion Studio calibration app. The motor speed never reads constant. It always rapidly changes in a 200-300 point range. Sometimes the motors perform as expected with the encoder counts & speed varying directly with the motor control sliders. Sometimes a reverse response indicates the need for a switch in motor or encoder polarity, but otherwise normal. Other times the speed readings seem oddly defferent between the two motors. It also depends whether Sync Motors is checked or not: e.g. with Sync Motors OFF, RoboClaw #1 M1 speed (motor 5) reads ~5200 at max throttle while M2 (motor 6) reads ~4100. With Sync Motors ON, M1 reads ~5200 and M2 reads ~1400. I thought maybe the wheels were imbalanced causing unevenness in the readings. However, removing the wheels made

little difference. Could the tiny asymmetry in the hub clamps make that much difference? Without the hub clamps M1 (motor 5) reads ~5350 and M2 (motor 6) reads ~4150 +/- 50.

RoboClaw 1	M1 (5)		M2(6)	
Motor alone unsync	5300-5400		3900-4100	
Motor alone sync	10000-11000		1300-1700	
with hub clamp unsync	5200-5400	3500-3700	0-(-)3700 ?!?	
with hub clamp sync	3900-4200		900-1399	
with wheel unsync	5300-5400	4000-4200		
with wheel sync	5200-5500	1200-1500	(-)3800-(-)4200 ?!?	

Lesson #1: DON'T sync motors
Lesson #2: Readings can change dramatically between trials esp on M2 sync even into negative range. How will the rover work if one minute it thinks a motor is running forward, and another it thinks it's running backward?!
Velocity Settings
Let's take 5300 and 4000 for M1/motor 5 and M2/motor 6 respectively
and try to Autotune the motors.
M1/motor 5: P: 6541.70623 I: 1824.40640 D: 0.00000
M2/motor 6: P: crashes Motion Studio

Made a Y-adapter between the DuPont connectors at the motor end. Motor responded appropriately to slider. Without the motor attached, the voltmeter read proportional to the control slider up to the battery voltage. When the USB oscilloscope was applied across the power circuit, and the slider advanced, it melted the ground probe of the o'scope and Motion Studio crashed with an unexpected error. Reading input voltage at the RoboClaw, I see a small voltage even when the kill switch is off. When the kill switch is on, it reads battery voltage. When turned off, the voltage drops slowly back to almost, but not, zero. Looking at it, I suspect we have the kill switch wired on the ground side, not the supply side.

2018.12.30
Further adventures of motor calibration. On RoboClaw #1, Motor-2 (motor 5) kept getting erratic speed readings and wouldn't Autotune. Applying the voltmeter to the RoboClaw output without the motor attached showed appropriate positive & negative voltage to nearly battery voltage. Applying the Parallax USB digital Oscilloscope showed PWM in proportion to the slider at rapid sweep. At slower sweep the multiple microsecond pulses appeared to merge into larger but erratic pulses. Was this an artifact of a digital scope? Probably. Hard to believe that such precise pulses on a micro scale would be eratic on a macro scale. Attaching an analog oscilloscope couldn't reveal much at slow speed.

Then the slider was moved to reverse, there was a smell of burning plastic & the negative probe to only the USB oscilloscope started melting. The voltmeter & analog oscilloscope probes were ok.

After the meltdown, the motor would turn forward, but not reverse. The voltmeter read +14.97 vs15.29V battery max forward but only -5.7/15.29V at max reverse. The analog O'scope read appropriate square wave PWM forward. But in reverse showed a rounded negative slope with a sharp return to zero at low speed. At med-high speed the arcs jumbled together in a blurred mess. The behavior and readings were the same with a different motor and without the motor attached, so it wasn't a mechanical stall condition of the motor. Why would a RoboClaw fail on just one motor only in reverse? Go figure! New RoboClaw ordered (not in stock, notice requested).

2019.01.12

Frustrating day. Roberto unable to get RC's to calibrate, particularly the corner motors. Confirmed that RC #1, motor 2 fails on reverse. Jim, Roberto, & Carroll applied voltmeter and logic analyzer to RC #4. Smelled hot & failed. Applying tests to Absolute encoder on corner motor #2 caused overheating for very unclear reasons. We failed to bench test the electronics before assembly. Looks like we'll have to take a backward step and do that. Roberto ordered 3 more RCs.

2019.01.15

Hello Jim,
I replaced the faulty RoboClaw and now the motors run correctly.
The Pololu 5V regulator was disconnected and when I connected it back, I notice smoke coming form a shorted wire marked "SIG from PI", which I replaced.
I had to replace the 5V power to RoboClaw 1:5/6 because the Dupont terminal would not stay in place.
I also disconnected power to the RoboClaw 4:1/2 because the chip on it was getting really hot.
I still have a problem with the calibration of RoboClaw 1
Somehow the encoder signal of one motor is unstable and always positive.
The Table under Fig 28 seem to indicate that the absolute encoders are pure analog and should return 0 to 5V on the blue wire when fed 5V (+5V Orange / GND Brown).
I have isolated the corresponding wires in the CAT6 test plug you created.
If you've been consistent in wiring the plug, this should allow us to test all 4 absolute encoders
+5V green/white
0-5V analog signal orange/white
GND green
The corresponding RoboClaws need an input from 0 to 2V and the Voltage Divider of Fig 29 should take care of that.
However, if I'm not confused, the absolute encoder should output a voltage varying from 0 to 5V.
When I manually test the absolute encoder No. 3 I get a constant voltage of 1V.
I'm not as patient as you and I believe you and I should methodically retest the wiring.
See you Thursday
Roberto

2019.01.30

RoboClaw calibration. The RoboClaws have been erratic. We replaced one that seemed defective. Tested each Absolute Encoder on the steering motors. 2 (1 & 3) tested ok giving outputs of 0-5V when attached to 5V & GND. 2 others (2 & 4) returned nothing. Requested Return Merchandise Authorization from USDigital & sent them back yesterday. Now I'm going to document each RoboClaw/motor system and their problems.

Default settings:

Setup:
- Packet Serial
- Sign Magnitude
- Enable Multi-Unit Mode

Serial:
- Serial Packet Address: 128, et seq.
- Baudrate 115200

Battery:
- Battery Cutoff: Use User Settings
- Max Main Battery: 17.5V
- Min Main Battery: 11.5V
- Max Logic Battery: 5.5V
- Min Logic Battery: 4.5V

Motors:
- M1 Max Current: 3.0A
- M2 Max Current: 3.0A

I/O:
- Encoder 1 Mode: Quadrature for 128, 129, 130, Absolute for 131, 132
- Encoder 2 Mode: Quadrature for 128, 129, 130, Absolute for 131, 132

RoboClaw #1
- motor # 5 (LR)
  - forward: doesn't run. M1 Over Current (OC1) signals red but current graph flat
  - backward: bumpy speed graph, max neg speed -5600
- motor # 6 (LM)
  - forward: wiggly speed graph, max pos speed 5500
  - backward: goes off chart negative, max neg speed -15,000

RoboClaw #2
- motor # 7 (LF)
  - forward: speed graph only negative, max fwd speed 4???
  - backward: speed graph only negative, off chart, max rev speed -53??
- motor # 8 (RF)
  - forward: smooth neg speed graph, max fwd speed -55??

backward: smooth pos speed graph, max rev speed +56??

RoboClaw #3
motor # 9 (RM)
forward: no movement, M1 Over Current (OC1) signals red
reverse: speed graph off chart positive, erratic, max rev speed +13,5??
motor # 10 (RR)
forward: minimal slow fwd movement, M2 Over Current (OC2) signals red
reverse: smooth in chart pos speed graph, max rev speed +5???

2019.02.02+
Isolation of RoboClaw problems
1) take virgin RoboClaw "#6"
[Noticed that some RoboClaw have none, one or three buttons but all say V5C]
2) use battery charger as 16V source
3) take 5V regulator from Fidelity & use in isolation w/ battery charger source for 5V Logic Battery power.
4) Motor 1: Motor 8; Motor 2: Motor 7
5) Reverse polarity of Motor 1 as contralateral
6) Reverse polarity of Encoder 1 A&B "
7) Parallax USB oscilloscope to Motor 1: Enc A&B; Motor 2: Enc A&B; Motor 1&2 Enc A.

Result: PWM tracings look reasonable on both motors
Trying to tune motors results in program crash.

2019.02.04
Took wheel off M2 which seemed to make M2 speed more stable.
I believe slight eccentricity and asymmetries of wheels make for slight "uphill", "downhill" stresses on motors. Motor tuning still crashes RoboClaw.

2019.02.14
Team,

Just had a very friendly and informative talk with tech support at BasicMicro re RoboClaw erratic reading issues.

1) the Pololu (and everybody else's) motors are Chinese made and ~~crap~~ "poorly designed". They often have a "snub" capacitor or two that feed motor noise into the encoder circuits.

He discussed a couple possible fixes using 0.01uF caps.

2) running the motor & encoder wires together, as suggested by JPL, is asking for noise. Our CAT5 twisted pairs should help alleviate the problem but he advised

keeping them separate.

Perhaps a 4-conductor shielded cable for the encoders? I'll be playing with the wiring and putting my O'scope to the situation.

My daughter has happily sold her Harvey ravaged & beautifully remodeled condo and will be moving home pending starting P.A. school next year.

I have to help her move the last (big) stuff out Saturday so I WON'T be at H.R. this Saturday. Take notes for me!

Jim

Since then, dis-assembled the mesh enclosed test cables and spread the power & encoder wires as far as possible. Also cut the shrink tubing on the motor leads to allow further separation on that then. Put the Parallax USB O'scope between the motor outputs and motor ground and got clean PWM output. Putting the scope between encoder EN1 and EN2 outputs and encoder minus revealed minimal noise at the base of the square waves. Not enough to spoof and encoder signal. The forward and reverse speeds were in the mid 5000s and were reasonably stable +/- 100 or less. Was able to autotune M1! M2 crashed with the red ERR LED on but no red indicators on MotionStudio, eg. OC2 (overcurrent on M2) as had sometimes appears on tests previously.

2019.02.19
Hello Jim,
This is mixed news.
I'm sorry you won't be able to join us Thursday, but I'm very happy that you're making progress with the Mars Rover.
I have now most of the hardware necessary to make Fidelity an autonomous vehicle
Robotic Arm with 6 degree of freedom (Still missing the gripper but the necessary Dynamixel servos are coming)
Latte Panda 2Gb SBC (a Raspberry Pi on steroid and turbocharged)
Intel RealSense stereo camera with distance sensing and imaging
A Movidius 2 Neural Stick for imitation learning and image recognition.
The paraphernalia for power management.
All we need is for Carroll to write the necessary code.
Starting next week, I'll have time to go back to the Rover.
You are the boss and I'm ready to do what is needed to finally make Fidelity show its stuff.
Ciao
Roberto

2019.02.19
To continue testing the "virgin" RoboClaw #6 (one from Roberto that has not been installed on the Electronics Board in the rover so not subject to possible power

mishaps). This time to test M2 again and try to autotune.

Further work on RoboClaw #6, using motor 7. Was able to autotune M2, but when tried to autotune M1 (having moved the connections to M1) it crashed. Maybe it needs 2 motors to autotune? Prepared the other cable, attached M2 to motor 8 & tried again. Also saved the RC configuration after tuning the first motor & reloading it for the second. Still crashed on M1.

2019.02.26

Found this on a Google search: “RoboClaw fails to auto tune”

Basicmicro Support 2017.12.29:
“Another possible cause of the problem is motor noise glitching the USB connection to the PC. You should use a USB cable with a ferrite choke on it(looks like a barrel near the end that connects to the Roboclaw).”

I don’t think I have a USB with a ferrite choke x- on a keyboard (attached).

Since last week trying to “divide and conquer” the RoboClaw motor tuning problem. I separated the cable wires. [Also thinking about getting 4-conductor shielded wire for the encoders & maybe even 2-conductor shielded wires for the power to minimize noise.] When put to the scope, I’m only seeing about 10% noise at baseline and less at the top of the encoder square waves. I see some noise on the charger power but I think it’s coming from the Parallax scope as it shows even if the ground clip is detached! I’ll have to try putting the RoboClaws on battery power which *should* eliminate AC noise. I tried the 0.01uF caps from each encoder to ground as suggested by BasicMicro. It reduced the top noise but increased the baseline noise. Trying a 0.001uF cap may have decreased the trigger level at which the square waves jumped around a little. But I didn’t see any noise spikes that exceeded a fourth of the square wave height. Doing a capacitor continuity check on one of the motors as suggested by BM showed conductivity on one but not the other. Shorted cap? My analog scope isn’t clear/bright enough to show the noise and my Parallax USB scope isn’t sensitive enough. Roberto’s Digilent Analog Discovery 2 works nicely. I’ll have to get one w/ the HCC student discount!

I’m testing each RoboClaw starting with the “virgin” #6, then #1-5. I attach it to each motor pair in turn 5/6, 7/8, 9/10. Run forward & reverse & note the speed. Then I try auto tune and note success or ERR. Then I reverse M1 and M2 and try again. I’m getting reasonable speeds in the 5000's and almost no 20,000's readings (just once and resolved on repeat). I got an “unknown error” pop-up window that turned out to be a reversed cable/motor connection. [Should change to the 6-position keyed Molex connectors I bought. Used DuPont connectors as the motors have them but, as meticulous as I am, I still managed to get one backwards.] I’m getting “success” on auto tune maybe 1/4 of the time. I think I should test only one motor at a time on just one channel to eliminate cross noise.

2019.03.03
RoboClaw test procedure:
Take each RoboClaw individually
Start with channel M1:
Attach batter power +/- to RoboClaw +/-
Attach cable motor power +/- to M1A/M1B
Attach +5V +/- from voltage regulator to logic battery LB IN +/-
Attach encoder power/gnd blue/green to first +/-
Attach encoder signal yellow/green to EN1 +/- (may need to reverse if speed polarity reversed)
Attach USB cable between RoboClaw and PC
Attach the test cable from the RoboClaw to the motor under test ensuring correct polarity -
be sure the red/black wires are on the same side else errors and damage may result
Open BasicMicro Motion Studio (formerly Ion Studio)
Apply 16V battery power to RoboClaw
When RoboClaw appears on MotionStudio, Connect Selected Unit to selected RoboClaw
Device Status: (default window) check for all green status boxes.
File / Load Settings / Fidelity.cfg
General Settings: check that NASA/JPL settings are in place (Baudrate 115200, etc)
PWM Settings:
Drag Motor1 slider to top.
Be sure the speed is a positive number. Else reverse the encoder signal wires (yellow/white)
Note the M1 Speed. It will vary +/- 50 points.
Pick a number the next multiple of 10 points higher that the largest number you see.
e.g. if you see 5630s, choose 5640 or even 5650 as the top speed.
This isn't critical as the top speed value for auto tune will be a multiple of 1000 above, e.g. 6000
Note this number on the spreadsheet for that RoboClaw, channel, and motor number as the forward speed.
Drag Motor1 slider to the bottom.
The speed should be negative.
Repeat the above procedure for the reverse speed.
STOP ALL
Velocity Settings:
In the Motor 1 QPPS (Quadrature Pulses Per Second) box, enter a number comfortably well above the M1 forward speed observed. I used 6000.
Click on Tune M1, click OK on the warning.
Observe the motor/wheel. It should rock back a forth a few times and stop.
The LEDs on the RoboClaw should all be green.
Calculations for the Velocity P, Velocity I values should appear. Copy to spreadsheet.

Velocity D always calulates to 0.0000. Ignore.
There should be a calculated/observed QPPS number. Copy to spreadsheet.
If test is successful enter “success” and highlight in green on spreadsheet.
If test faile, enter ERR and highlight in red on spreadsheet. There will be no P, I or QPPS.
Click Disconnect Selected Unit
Remove power from RoboClaw.

Move the test cable to the next motor to be tested.
Repeat above.

When all motors have been tested on Channel M1 switch to M2.
Move motor power wires to same positions on M2A/M2B
Move encoder power wires to next set of pins.
Move encoder signal wires to next set of pins.
Keep LB +/- the same.
Move cable to first motor to be tested
Repeat procedure above.

When all motors have been tested on M2, switch to the next RoboClaw to be tested.
Lather, rinse, repeat until all RoboClaws have been tested.

Project completion

Review project for errors, improvements and expansion
Create new BoM for Phase II

Lather, rinse, repeat

Naming of the robot

2018.10.19
Team,

Our rover needs a name.

So far it's only bubble-gummed together, but soon (perhaps in geologic time) it will be running. It needs a name.

In keeping with the currently active Mars rovers *Curiosity* and *Opportunity* with the -ity ending indicating a (preferably positive) condition or state of being, and in keeping with the bold advances that Houston Robotics is making,

I suggest we call it "*Audacity*."

Comments?

Wikipedia: Mars rover
Dictionary: -ity
Wikipedia: Audacity

Jim

==========
2018.10.28
Fellow robonuts,

I've been re-thinking our rover's name. I tossed out Audacity in the spirit of Houston Robotics which got a few amens. (Not to be confused with audio editing software.)

Chaz suggested Mobility, or Moby for short. (Not to be confused with a retired Houston radio DJ.) Mobility is certainly what the rover is about. But I think of mobility as an inherent property of robots, the exceptions proving the rule.

Omar's kids suggested Creativity. Certainly a noble attribute for a rover and what robotics should be about. What bothered me about that name was that this version of the rover wasn't about creativity. It was about trying to be as faithful as possible to the NASA/JPL recipe the first time, just to see if we could do it. Because of that, I think this rover should be named *Fidelity*.
The suggestions we all have made about how to correct, improve, expand, and put the Houston Robotics touch to the basic model tell me our *next* rover is the one that should deserve the name *Audacity*.
...
I'd like a reply-all from everybody on this list yea, nay, or indifferent and any thoughts or comments you may have.

Jim

Tom Atwood:
Hi Jim,

From my vantage point as former editor of Robot magazine, it is my opinion that your recommendations are well-reasoned and the names, themselves, ring true to the ear. They sound right.

I endorse the name Fidelity for the current rover, and I believe Audacity is an excellent title/moniker for the next one.

Thanks for the opportunity to vote! I look forward to posting news updates on these projects at www.The-NREF.org.

Best regards,

Tom

========
Omar H Gomez:
Hello to everyone, I trust you are doing great!!!,
I liked Audacity best, Creativity second since there is much to do still with the software and it is not only to what we have done to put together the rover but also the missions we want to accomplish.

There were my 2 cents...

Cheers!!!

Omar H Gomez

= = =

Roberto Pensotti:
Jim, Tom and friends,
My joke is that in view of the changes and workarounds it should be called "Remedy".
Seriously speaking, I think both names suggested by Jim are fine.
I'm still partial to "Audacity" though.
Roberto

Charles de Montaigu:
I am with Jim pick's: Fidelity/Creativity.
I will name my contraption Moby when it is in running shape.

= = =

David Williams:
Hi, All.

Fidelity is a beautiful word, and I use it often when comparing something with its closeness to its ideal. If the goal of the initial HCC mars rover build is to stay true to the NASA/JPL design, Fidelity fits and is a nice-sounding name. I like Audacity and Creativity as well, so I think any of those names will be great.

David
==========
Robot Safety:
2018.10.31
Team,

KILL SWITCH

We all know better than to trust version 1.0 software of any kind to work right the first time. Let alone on a mobile robot! Every live test of a robot (of sufficient size to be dangerous, I exclude SneakerBots & GoPiGo) should have a Robot Safety Officer whose only job is to have their thumb on the kill switch and their eyes and ears on the robot.

Here's a sample of some remote switches suitable for a kill switch (scroll down below the featured product):

https://www.aliexpress.com/item/Free-shipping-12v-1ch-rf-wireless-remote-control-power-switch-delay-Time-relay-is-adjustable-automation/32619835127.html

[Just check them for spy chips or find an alternative friendly source.] On the Stingray I put mine after the power take-off for the computer on the way to the motor controllers ('cause that S.O.B. can move!) If it had any major actuators like arms, I'd switch them too, but leave camera pan & tilt on. That way it doesn't kill the brain, but keeps the bot from running off or hitting you. Or, base case, just put it right after the main power switch and, in case of emergency, kill everything. This risks ungraceful system shutdown but we're talking safety first.
BOP STOP

I've searched for an inexpensive (<$20) panel mount red mushroom head emergency stop button but haven't found one I like. Ideally one that would fit the 1/2" holes on the rover's pattern plate top but willing to be flexible. Anybody see any suitable candidates?

TWO-STAGE POWER SWITCH

The Parallax Board of Education had a 3-position slide power switch: off/processor only/processor and servos. I thought this was a good idea as it allowed system

boot-up, stabilization, and testing before applying power to any actuators. I don't know how Ariel and Sonic are wired up but I'm thinking "the rover" (I have second thoughts on the name, more on that to follow) should have a second toggle switch for motor power. Can anybody think of a reason to have motor power without processor power, thus 3 switches - main, processor only, motor only? Should we have car-like key switches (off, aux, ignition) so not anybody can turn them on? All this in addition to a remote kill switch.

SAFETY MODULE

I know this can be done in software as a separate thread. But I think a simple, all-in-one hardware solution (again, not trusting v1.0 software) would be advantageous not only to our projects, but something we could provide to the greater robotics community. My electronics knowledge is pretty shallow. What is the minimal hardware that could turn a piezo buzzer & 4 quadrant yellow LEDs on for 1/2 second and off for 1/2 second? It should have its own voltage regulator so it could tap directly into a 5-24V power circuit without adaptation. Can this be done with just a transistor, capacitor and resistor? A 555 timer? Op-Amp? Help me here! An advanced version could include the emergency push button. The deluxe version adds the remote kill switch. We could make it as a naked PCB or in a nice 3D printed case.

BUMPERS

I don't know about you, but my version 1.0 object avoidance robots run into things! Sensors are cool and necessary. I haven't put this to the test, but I don't fully trust IR & US sensors to detect the dark fuzzy sweat pants I'm wearing now. Let alone the yet to be proven software to make the robot stop in time. Let's think about physical bumpers to protect our robots, their environment and ourselves. Something as simple as a pool noodle or an arc of garden hose around the front and rear of our beasts. 3D printed spring actuated bumpers? Go for it! Just not my shins!

Your thoughts & contributions encouraged!

Jim
= = =
Roberto Pensotti:
Nice thinking Jim,
I'll think about an elegant solution, but in these days of political correctness, we cannot call it a "KILL SWITCH".
We should call it: "Temporary disabling switch", "Instant paralyzer", "Gentle dissuader".....
Roberto

> R,
> Good point. Perhaps we could call it "Thorazine".
> J

Audacity - corrections, improvements & extensions from Fidelity:

Corrections (change things what were just wrong on Fidelity, incl documentation)
see Github Issues

3/16 spacers on rocker-bogies
screw bearings from inside on rocker-bogies
“spaghetti farm” PCB > “monolithic” board per JPL

Improvements (make things originally on Fidelity, but better, more, modified or moved)
3D printed jig for drilling wheels
Colorful anodized Al parts (red/white/blue per USAi Labs colors)
Acrylic back plane modified to better fit Raspberry Pi, add holes for switch & battery connectors. (Applied to Fidelity)
Laser engraved front plane w/ Houston Robotics & Houston Community College logos
Names of all HR members who worked on rover etched on front plane
LED lighting of front plane & interior
Lubricated washers (6) on main axle on each side of rocker-bogie & next to body.
Ribbon cable from LED display to Logic shifter board. Display already has 8x2=16 pin keyed IDC cable connector, why not use one on the Logic Shifter to avoid the jumper “spaghetti farm”? Also - make a GPIO break-out “hat” for the Pi. Have 2 cable connectors on it - one IDE 16 pin for the LED display in the head, one 4 pin for the motor controllers. Requires at least 2 feet from Pi to head; allow 3 feet for workability.
Put connectors between the body & the rocker-bogie cables and between rocker-bogie cables & motors/encoders to make dis-assembly easier.
Bumpers
Structural material / fabrication improvements: 3D printing, injection molding, PVC/Al pipes, roller bearings.
Higher torque motors

Extensions (add things that weren’t on Fidelity)
Change LED display to HD touch RPi display
Robot safety module - beeper, flashing lights, emergency stop button, kill switch
Sensors - US, IR, LIDAR
Camera
Speech
manipulator
AI
Solar panels
Radar (new TI chip)
Ruggedized waterproof body
IR / laser / LED headlights
Navigation lights like boats & planes

From Mars Rover Parts Comment.rtf:

Mars Rover parts odering
James H Phelan
2018.08.25

Funds committed from Houston Robotics members:

Roberto Pensotti	$750	+ Raspberry Pi + video camera & monitor + Xbox + etc.
Charles de Montagne	$250	
Andrew	$600	
Harsh Basan	working interest	
James Phelan	remainder	

printed circuit boards OSH Park email sent, support account initiated
https://support.oshpark.com/support/home

3-D printed parts HCC Stafford pretty sure we can do these here. Done!

cc Master BOM to Houston Robotics BOM
deleted titles & totals except for column heads to ease sorting
changed some links from part name to actual link in order to sort by source for ordering
sorted data by Link to source together

B11	#6-32 Locknuts	#80 needed	changed quant to buy from #200 to #100	
T1	#6 x 1/4 Spacer	#60 needed	#68 ordered by JPL? Why not #70 for spares?	
B1	#6-32 x 1/4 Button head Screw	#182 needed	#300 ordered by JPL? Why not just #200? CHANGED	
T8	#2-56 x 3/8 Threaded Standoff	#2 needed	#4 ordered by JPL? Why not just #2? CHANGED	
T7	M2.5 x 12mm Standoff	#6 needed	#10 ordered by JPL? CHANGED	
T3	#6-32 x 0.75 Standoff	#~~20~~ 16 needed	#16 ordered CONFIRMED #16	
S17	Dual Side mount A	#8 needed	16 ordered	CHANGED
n/a	0.1 Pitch Pin Header (40 pins)	? needed	#5 ordered WHY?	OMITTED
n/a	10x2 DIP socket for logic shift	2 needed	2 ordered	ADDED
n/a	4x2 DIP socket for OpAmps	2 needed	2 ordered	ADDED

From Mars Rover Problems.rtf:

E21 LED Board
serial splitter board missing from Electrical BOM, added from Master

no # Serial Splitter Board (for RPi to Motor Controllers)

E21 Logic Shifter Board (for RPi to LED display)
none Serial Splitter Board (for RPi to Motor Controllers)
none Voltage Divider Board (for Encoder / Motor Controller mismatch)

E20 refers to BOTH Switch and Encoder Voltage Divider Board.
B10 BOM M2.5 x **4mm** Electronic Board Parts List M2.5 x **11mm**
NEEDS TO BE 11mm (McMaster only has even, ie 10mm.
Should also be button head screw instead of cap head to match.
B13 BOM 2-56 x **1/4** button head Electronic Board Parts List 2-56 x **3/16**
S28 Electronics Board,

1) The mounting holes for the Serial Splitter Board are too small. They should fit a **4-40** screw.
2) The mounting holes for the Voltage Regulator E4 are too close together & wrong orientation. Should be **21mm** / 13/16" apart & in
3) Opposite diagonal orientation according to diagram & photos (although diag. & photo show facing opposite direction).
4) Above solved by drill press at home (done, successful).

Will seek screws locally.
Got M2.5 x 8mm screws from Ace Hardware. Success.

2018.09.16 Submitted above ISSUES to the NASA/JPL Mars Rover Github.
2018.09.18
RC Turnbuckles ordered from:
https://www.amazon.com/Vktech%C2%AE-Aluminum-Steering-Linkage-Upgrade/dp/B00MWQKPHS/ref=sr_1_1?ie=UTF8&qid=1535327690&sr=8-1&keywords=06048B
Package arrived today torn & empty.
Reordered from:
https://www.amazon.com/gp/product/B071HTBCMC/ref=oh_aui_detailpage_o01_s00?ie=UTF8&psc=1
New order appears to be 2 sets of 2 but reviews indicate quick & reliable delivery. We'll want a 2nd set for V2.0.

0.25" Pillow Bearing Block
On backorder from:
https://www.servocity.com/0-250-1-4-bore-flat-bearing-mount
Is STILL on backorder with none available elsewhere (except maybe England). Source, Actobotics, seems out of stock. No clue when will be available.
Update: 2018.10.18 call to Servocity: part due in next week!

Set screws on neck clamping hub & wheel hubs

https://www.servocity.com/1-315-pvc-clamp-hub-b
https://www.servocity.com/770-clamping-hubs
If the screws are put in the wrong way, ie not in the reamed out side the heads will break off leaving the screw irretrievably stuck in the hole

Aluminum tubing 0.5"

https://www.mcmaster.com/89965K364
Is about 0.001" too thick and needed to be lathed down to fit into bearings at all locations.

NASA/JPL GitHub Issues contributed:

4-pin headers 0.1 not 0.05" #34
The specs call for common 0.1" headers on the PCBs. The master parts list, however, has a Digi-Key link to 0.05" headers. (as well as 40 pin 0.1" headers which I see no reference to x- perhaps to break into 4 pin units?) I searched the docs and found no reference to 0.05" anything. Error? If so, will need to swap out with Digi-Key.

ericjunkins: There should only be 0.1 inch standard pin headers and connectors. I'll check through for the digikey parts and fix error if necessary

ericjunkins: Closed with commit ef91a15

Electronics Board holes for Serial Splitter too small; holes for 5V regulator misaligned. #41
On the laser cut Electronics Board, in the lower right corner, the holes for the Serial Splitter PCB are too small. They should accommodate the #4-40 screws needed for the #4-40 spacer, but they won't fit.
In the same corner, the holes for the 5V regulator should be 21mm or 13//16" apart (instead of 20mm). Also, according to both the diagram and the photo, the holes should be oriented on the opposite diagonal (NW-SE instead of NE-SW). Won't know until we get to wiring whether that's important.

ericjunkins: I do believe that might be correct about the 5V reg being flipped upside down. I will update the drawing to have the appropriate dimensions and be oriented correctly, sorry about that mistake.

As far as moving forward for you, I think there are a couple of options.

Match Drill the hole pattern in the acrylic yourself. A 2-56 screw is relatively small and you would run little risk of cracking acrylic if you choose to do this and are careful.

Install the header pins upside down on the 5V regulator and then make sure to connect things accordingly.

Make your own layout and design for the electronics board assembly and let us know!

.dxf files for Electronics Board and Back body plate corrupted? #45
Previously downloaded & laser cut successfully both pieces. Recently noted post from member unable to download. Tried myself - able to download both files, but got "unrecoverable error" in CorelDraw and "failed to insert dxf" in Fusion360. Perhaps the file got corrupted?
Jim Phelan, Mars Rover project lead, Houston Robotics

ericjunkins: @JHPHELAN I just checked and was able to open both of them in e-drawings, as well as upload them to sculpteo and the .dxfs were all recognized normally. Can you try just re-downloading them maybe your local version somehow got corrupted
Right-click and "save as" filename.dxf failed to open. But this worked:
Click on file, click on RAW, Control-A to select all, Control-C to copy all.
Open a new text file in WordPad, Control-V to paste, save as "filename.dxf", close file.
Open file with CorelDraw. BINGO! Save as "filename.cdr".
Don't know why simple "save target as" doesn't work!
Jim Phelan, Mars Rover project lead, Houston Robotics

ericjunkins: As a question, do you have the entire repository cloned locally? To test this I ran a git-clone on the repository and just took the file from there and it worked fine. If you have it cloned locally then all you would have to do is run a 'git pull' to get the latest updated files and that should have all worked well.

No, I don't. I should really do that. I've git-cloned once on Raspbian, but never on Windows-10. I'll look into that!
Jim Phelan, Mars Rover project lead, Houston Robotics

Done! Git for Windows install asks a lot of questions. Defaulted everything except changed the text editor from Vim to Notepad++ which required another download (to which it conveniently linked but watch out for add-on downloads like WinZip updater). Opened Windows Git GUI and selected clone existing repository. Then cc the Git-hub Mars rover clone link to clipboard, pasted into the Windows git-clone source line. The target line was confusing as I created a git-clone folder in the Mars rover project folder and it stopped saying "folder already exists". So I just backed out to my project folder, deleted the git-clone folder and put "git-clone" as the target folder. Git then created & populated the folder. Clicked on the .dxf file and it popped open (well s-l-o-w-l-y blossomed) in CorelDraw. Sweet! Appreciate y'all taking my nit-picking Issues seriously and so quickly updating the project! Houston Robotics is really excited about this project (naturally, this is Space City after all) and looking forward to further collaboration as we progress with the build.
Jim Phelan, Mars Rover project lead, Houston Robotics

ericjunkins: *Glad to hear it worked :)*

B10 screw incorrect #39
The Master Parts List lists B10 as a M2.5 x 4mm cap head screw.
The Electronics Board parts list shows 11mm.
For consistency (gasp!) it should be a button head screw to match the others on the board.
McMaster only has even sizes in M2.5 button head hex screws, so 10mm:
https://www.mcmaster.com/92095a460

On further exploration, the T7 M2.5 standoff is listed as 11mm long in the build but 12mm long on the Master Parts List and in the McMaster link. Longer is better. But the B10 M2.5 screw is listed as 11mm on the built but links to 4mm on the Master Parts List & Link, above. The standoff is 12mm and had to accommodate a screw from both ends, leaving 6mm for each screw plus the thickness of the board on either end, 3mm for the Electronics Board and 2mm for the Raspberry Pi. So the screws should be no longer than 8mm. I bought some m2.5 x 8mm screws at Ace Hardware & they work OK. M2.5 x 6mm hex cap head to match the others would probably be better: https://www.mcmaster.com/92095a458

B13 links to wrong size screw #40
B12 #2-56 x 3/16" Button head screw has a link to McMaster 1/4" screw. Don't think it makes any difference, but....

accidentalmeme: The "# 2" links to issue 2.

I believe the #2 refers to the screw size: #2-56.
My concern is that the screw length is listed as 3/16" but the source link refers to a 1/4" screw.
Will have a better sense of which is right as we get into the build.
Jim Phelan, Mars Rover project lead, Houston Robotics

E20 refers to BOTH Switch and Encoder Voltage Divider Board. #42
E20 on the Master Parts List refers to BOTH Switch and Encoder Voltage Divider Board.

Electronics Board: shortest RoboClaw needs 8 x #4 washers to make tight #43
The 5th RoboClaw motor controller on the Electronic board is supported by #4-40 1/4" stand-offs and 4-40 1/4" screws. 1/4" stand-off shared by 2 screws = 1/8" each. The Electronics Board is 1/8" acrylic allowing its screw to extend 1/8". The RoboClaw board is only 1/16" leaving a gap of 1/16" play when its screw meets its partner in the stand-off. A couple #4 washers does the trick and there's plenty left over. The purpose of the staggered elevation of the stand-offs is supposedly to allow room to access the USB port on the RoboClaw for testing and calibration. Seems this could have been accomplished as easily by rotation or translation of the RoboClaw but we won't know til wiring time what the reality of that situation is.
Jim Phelan, Mars Rover project lead, Houston Robotics

ericjunkins: I"ll look into these comments on screw lengths and standoff lengths. It is true rotation could have been done to have access to micro USB ports, I chose to keep

the wire routing for the Power to the motor controllers as simple as possible. I certainly would be open to suggestions as to other possible configurations of electronics boards that could make things easier! :)

Helpful jig for drilling clamping hub mounting holes in Traxx wheels #46
When the Houston Robotics Mars Rover team began to address the drilling of the mounting holes in the Traxxas wheels, we pondered how we might more accurately accomplish this. Initially we considered a vertical lathed spindle 12mm at the bottom to fit the wheel and 4mm at the top to fit the hub. Rotational orientation would still be by hand and there was concern that using the hub itself as a guide may damage the threads. We consulted the Houston Community College Stafford manufacturing campus Fab Lab supervisor Roland Fields. He suggested a more modern high tech approach of creating a 3D printed jig to fit the wheel hub to include appropriately spaced holes to guide pilot hole drilling. He directed us to lab assistant Frederic Lemme who whipped up a prototype on SolidWorks. After tweaking for fit, a further modification adding a hex nut shape to fit into the wheel ensured fixed alignment of the pilot hole guides to the target area. Houston Robotics' Omar Gomez and son accomplished the final drilling. The screws to attach the wheel to the hub seemed excessively long, but that's an issue for further along the build. ATTACHED is a drawing of the wheel adapter and zip file of the .SLDPRT and .STL files.
[attached .png of part and .SLDPRT CAD and .STL 3D print files.]

ericjunkins: Hi @JHPHELAN, yeah actually we have had a couple teams tell us they used some sort of 3D printed jig as a drilling guide to help with this. The main reason I am hesitant to include something like this in the documentation is because of the Traxxas wheels themselves. As they only stock a relatively small number of each wheels it is likely groups have to buy a wheel that is not exactly the same Part suggested. The hubs used the wheels differ from wheel set, however there doesn't seem to be any information on Traxxas as to which exact hub you get on each wheel, meaning one jig likely does not work for everyone.

If we can come up with a way to ensure that either A). Everyone is getting the same wheel hub, or at least know which wheel hub they are getting, or B). Make a jig that could work with all the wheel hubs, I think that would be a great optional thing to print to help teams easier to drilling into the wheels.

Overall though I like your design and idea. It has appropriate centering and clocking features to make the match drilling an easy task

I have added a note in the Wheel assembly build document to link to this issue so others can access and try using your jig! Thanks for the addition, I hope it is helpful for other teams as well :).

Glad we could be of help. Would love to see what other groups made to solve the same issue. Consistency of supplies/suppliers is always an issue. The last remaining par to acquire, one of the pillow blocks, is still on backorder! Fortunately in skilled 3D CAD hands (not mine!) a custom jig was designed in less than an hour and simplified the job greatly.

T2 spacer/standoff - should it be 3/16 instead of 3/8?
In the Rocker-Bogie Assembly the build parts list does not include **T2 the 6-32 3/8" standoff** that is mentioned in the build. Two are shown attaching an F Pattern Bracket to each side of the 2nd modified 3" U channel with two B3 6-32 x 1/2" screws. There is no mention of using a nut, but the 3D pdf shows the screws penetrating the F Pattern Bracket, standoffs, U channel, hex nuts and extending into the U channel. (Washers are neither mentioned nor shown.) Using a 1/2" screw as listed, the screw barely penetrates the F Pattern Bracket and the 3/8" spacer. Trying a 1/4" spacer the screw penetrates the F Pattern Bracket, spacer and U channel but not sufficiently for a nut to grab on, let alone penetrate the nut and beyond. Perhaps what was meant was a 3/16" standoff or spacer? That should allow the nut to attach but not be penetrated. A 3/8" or even 1/4" spacer seems to give more room than necessary between the F pattern bracket and the U channel. The 3D pdf suggests the space is tighter. **Was a 3/16" spacer intended instead?** And/or was a longer screw than 1/2" intended (5/8 or 3/4")?

Issue discovered by **Derrick & Alfredo**, Rocker-Bogie build team, Mars Rover project, **Houston Robotics**

Yet another nit-pick from
Jim Phelan, Mars Rover project lead, **Houston Robotics** ;-))

Logic Shifter board testing confusion - put DIR & OE to GND instead of GPIO
In testing the Logic Shifter Board section 4 and Figure 22 indicate that the DIR and OE pins must be LOW (0V or GND).
DIR (pin 1, upper left) can be seen to connect on the schematic (via pathway "11" and demonstrated by following the PCB trace and proven by continuity check with a voltmeter) to header pin P4-3.
OE (pin 19, second right) connects (via pathway "10") to header pin P1-2.
These pins are shown connected by grey and white jumpers to Raspbery Pi GPIO pins 10 and 12.

However, unless there is a script running on the Pi to cause these pins to go LOW, they will "float" and be neither HIGH nor LOW preventing the chip from functioning properly. There is no mention of running a script on the pi for this test. Instead, we connected them to two of many spare GND pins on the RPi such as physical pins 9 and 14 to force them LOW.

+5V (P1-1) red wire to RPi +5V (pin 2). Check.

GND (P4-4) black wire to RPi GND (pin 6). Check.

During testing at each step the +3.3V purple wire to RPi +3.3V (pin 1) needs to march down the logic shifter headers P1-3 through P4-2. At each step confirm the +3.3V on the corresponding Bx pin on the right side of the shifter (according to the schematic) and look for the +5V on its partner Ax on the left side one pin up.

Thanks to Roberto Pensotti, Electronics build & testing team, Mars Rover Project, Houston Robotics

Jim Phelan, Mars Rover project lead & chief fault finder ;-) , Houston Robotics

Voltage Divider Board testing reversed

Houston Robotics, Mars Rover project, Electronics build team member **Roberto Pensotti** beautifully populated & soldered the three PCBs. While testing the Voltage Divider all checked out according to the directions up until step 9 at which point it failed. Repeating the test together during our weekly MeetUp failed again as did a fresh attempt at home the next morning. Instead of measuring just under 2V, it measured 0.2V or less. IC chip placement, resistor values & soldering technique all passed inspection.

Tracing the directions back from the pin number to the header designation to the schematic to the encoder function revealed that +5V signal voltage was being applied to the ENCn OUT pin and measured at the ENCn IN pin which seemed reversed. Reversing the directions, placing +5V at the ENCn IN pin & measuring at the ENCn OUT pin measured consistently about 1.6V. Test passed!

Jim Phelan, Mars Rover project lead, **Houston Robotics**

2018.10.30
Why is rover body tilted forward?

Why is rover body tilted forward? I don't see that the actual Mars rovers had a forward tilt. Is it just a happenstance of the choice/necessity of placement of the Differential Pivot and Rocker-Bogies? I find no mention of it in the build documents. I can reduce the tilt from ~14o to ~10o if I lengthen the turnbuckles from their minimum length to as long as I dare without them coming apart. There is no mention in the build as to whether or how to adjust them.

By the way, there is no mention of leaving any space between the main axle on the rocker-bogies and the ends of the differential pivot to avoid friction. I left a mm or two. I

noticed there no lubricant washers next to the collar clamps along the main axle at the rocker-bogie arms & the body.

The last critical mechanical component, the 0.25" pillow bearing blocks are STILL on back-order in an apparent world-wide shortage! Though promised next week, we're going to try carbon-fiber 3D printing the blocks and inserting our own bearings. Will keep you posted how that goes.

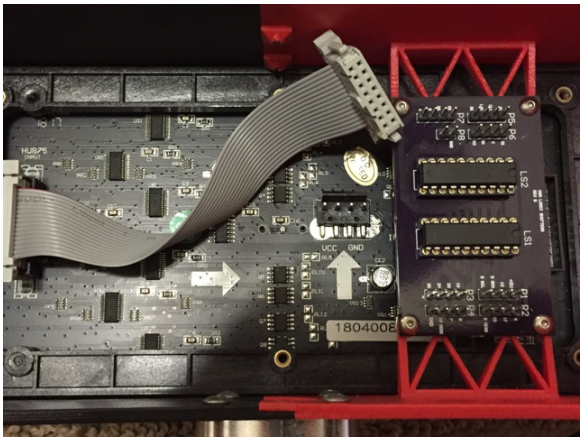
Jim Phelan
Mars rover project lead
Houston Robotics

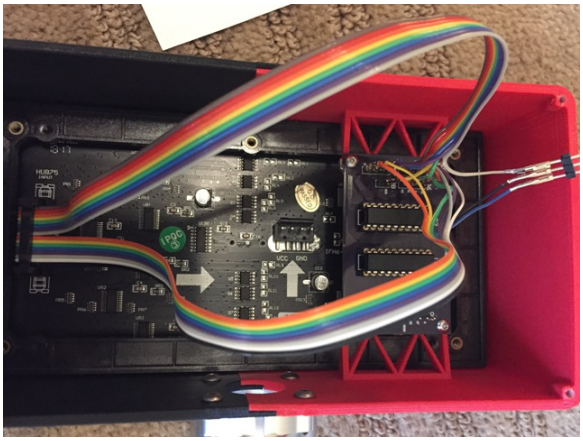
2018.11.04
Logic Shifter output pins scrambled vs LED display input pins

In Fig 31 "Raspberry Pi connections to LED matrix", the LED MATRIX input pins are in the same configuration as the 8x2=16 pin IDC connector jack on the LED display. However, on the Logic Shifter Board the output pins in P5, P6, P7 & P8 have no relation to the LED display input pins creating a chaotic "spaghetti farm" of jumper wires. P8 has one ground pin and one unconnected pin. There are 4 ground pins on the LED display. Adafruit recommends grounding all of them (although they are connected internally in the LED display). P8 could easily have made a place for all 4 of them.

Even better, however, would have been to replace P5-P8 with an 8x2 header on the lower edge of the Logic Shifter Board (or moved the board closer to the LED input in the head) so that the provided ribbon cable (or home made if not provided with LED display) could be plugged directly into the Logic Shifter Board with the Logic Shifter ouput / LED display input routed logically to the connector.

A similar 16 pin connector could also replace P1-P4 connecting between the Logic Shifter Board input and the Raspberry Pi (although the 1/2" neck hole in the body would have to be enlarged) and sort out the connections on the Raspberry Pi end. Better still, put the Logic Shifter on a GPIO break-out "hat" on the Raspberry Pi and just send a single 16 conductor ribbon cable directly to the LED Display input in the head (plus the power cable). The break-out board should also include a 4-pin connector to connect to the Serial Splitter Board.
...





James Phelan
Mars rover project lead
Houston Robotics

2018.11.06
Electrical Build Errors

Electrical Build, Page 22, blue text box, “In addition, Figure 19 shows how we routed the wires through the rocker-bogie, and then from the rocker-bogie into the main body.” Figure 19 should read Figure 21.

Electrical Build, Page 23, table, “2x 24AWG & 4x 30AWG”. No 24AWG wire is specified in the parts list, only 16, 20 and 30AWG. Presume to use 20AWG in place of 24AWG in this build.

James Phelan
Mars rover project lead
Houston Robotics

2018.12.10
add MotorCap.stl file to 3D printed folder in Corner Steering #72

Protection for corner steering motors.
The rock climbing picture shows them. There is discussion in the Open Forum about posting the 3D print file, but I can't find it.

We discovered the need for corner motor protection when inverting the rover into "dead cockroach" position to work on the electronics board. The solder lugs of the steering motors were vulnerable, got caught in the workbench rug and one broke off. Fortunately was able to bare enough pad to re-solder!

One interim solution was some Meccano cylinders in Roberto's collection. See photo.

A permanent solution "in keeping" with the rover was the addition of 1.5" channels with a pair channel connectors arching over each corner motor. (same Servo City source). See photo.

Would like to try the third option of 3D printed motor caps to match the encoder mounts.

James Phelan
Mars rover project lead
Houston Robotics

LED Display directions out of date
Electrical Build, Secton 4, Connecting the LED Matrix, refers to an Adafruit tutorial:
https://learn.adafruit.com/connecting-a-16x32-rgb-led-matrix-panel-to-a-raspberry-pi/overview
which, in turn, refers to a Henner Zeller github:
https://github.com/Boomerific/rpi-rgb-led-matrix
Unfortunately, the Zeller github has changed and the Adafruit directions no longer work. Fortunately, trying to run the obsolete Zeller code results in an error message that directs you to another Demo folder. We're still chasing that rabbit. If NASA/JPL or any of the teams that have crossed this particular bridge ahead of us could post a clear path to testing the LED display we would all appreciate it!

Jim Phelan
Mars rover project lead
Houston Robotics

Calibration update & issue

The Calibration document refers to a program Ion Studio at
http://www.ionmc.com/downloads.
This link redirects to
http://www.basicmicro.com/downloads
and the program is now called BasicMicro Motion Studio found under RoboClaw General Downloads.

Aside from that the program looks the same. Installation and General Settings no problem EXCEPT the Logic battery would read 2.7V instead of 4.5V. Otherwise it seemed to run ok. We changed the General Settings alerts to permit 2.7V. We've yet to figure out that discrepancy.

PWM settings were no problem and indicated the occasional need to swap connections. We were careful to zero the encoder values before starting each trial.

Under Velocity Settings at max speed the M1/M2 Speed boxes showed rapidly changing values. Sometimes in a fairly narrow range of, say, 7,000-8,000 where we

could estimate an average of 7,500. The other motor, however, may range from 27,000-29,000 which seemed wildly different!

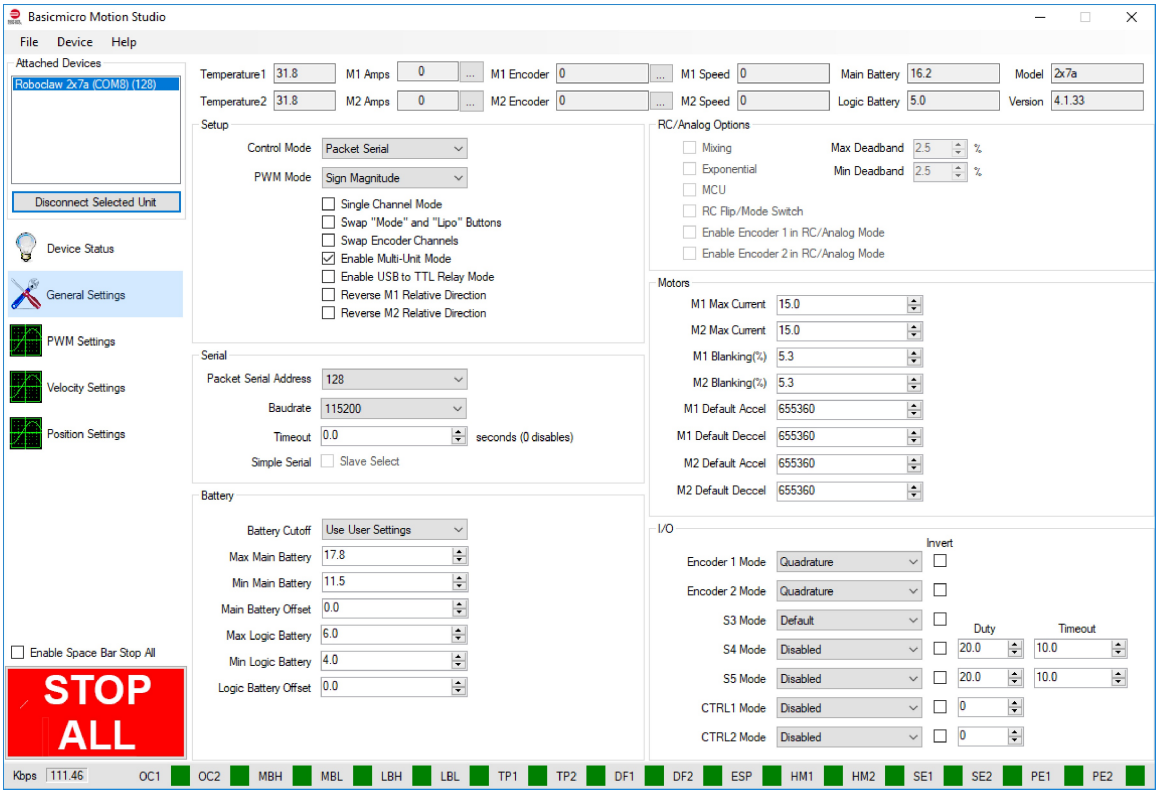
Entering the estimated max speed into the QPPS box and clicking Tune M1 causes the motor to slowly rotate back and forth a couple times as mentioned. It then continues to rotate in the same direction forever. No values are populated in the P, I, D boxes. Motion Studio has to be stopped via Task Manager while the motor continues to run at its last commanded speed until power is removed by the kill switch. Trouble shooting is still in progress, but if anyone else has insight into this problem it would be appreciated.

Will also post to the Forum

Jim Phelan
Mars rover project lead
Houston Robotics

April 28, 2019
Ok. A LOT has happened since my last report!

Re: Drive Motor RoboClaw Calibration issue:



See the snip of the MotionStudio General Settings screen above.
The JPL directions indicated to set the M1 Max Current and M2 Max Current to 3A.

But during calibration, the giggling of the motor frequently caused current flows higher than this setting, causing the calibration to stop and the program to freeze and the motor to continue in its last speed and direction. Only a reboot of the program and RoboClaw would solve the issue. Discussion with Nathan Scherdin at BasicMicro suggested to change the M1 & M2 Max Current to 15A for calibration, then to the stall current of our motors for running. Also, with the newest versions of the MotionStudio software, guessing the QPPM is unnecessary. Just click on the Tune button in the PWM settings window and autotune does the rest. After doing this, the calibration worked flawlessly!

Re: Corner Motor RoboClaw Calibration issue: Even though the absolute encoders run on 5V and the RoboClaw itself runs on 5V Logic Battery external power, the chip that reads the encoder only reads 0-3V. Therefore the Voltage Divider MUST be in place for the calibration to work! Also, these encoders are VERY fragile to over current. We blew out two of them and they're expensive! Nathan also suggested tuning with just PD setting instead of PID as the integral value is very sensitive to the instability of the encoder potentiometer and leads to oscillation.

Re: Installing ROS on Raspberry Pi:
Github post 2019.04.09:

> In the README.md instructions at
> https://github.com/nasa-jpl/osr-rover-code/tree/osr-ROS
> after installing ROS on the RPi, creating and moving to the ~/osr_ws/src directory it says to
> git clone https://github.com/nasa-jpl/osr-rover-code/tree/osr-ROS/ROS
> however this creates a directory/file not found error.
> Clicking the green Clone or download button and saving to clipboard produces this url, which works:
> https://github.com/nasa-jpl/osr-rover-code.git
> so it should be -
> git clone https://github.com/nasa-jpl/osr-rover-code.git
>
> [footnote: see 2019.05.27 recent Github post
> 5/18/2019, 4:59 PM
> Ian Reasor <notifications@github.com>
> Re: [nasa-jpl/osr-rover-code] Wrong URL for git-clone of ROS code (#23)
>
> You'll actually need to run:
>
> $ git clone https://github.com/nasa-jpl/osr-rover-code.git
> $ git checkout osr-ROS
> $ git pull
>
> This will get you the source code from the correct branch. Just checking out osr-rover-code.git will get you the master branch. I ran into the same

issue when trying to get the Arduino code. The README instructs you to run:

$ git clone
https://github.com/nasa-jpl/osr-rover-code/tree/osr-ROS/Arduino
OsrScreen

which results in: "fatal: repository
'https://github.com/nasa-jpl/osr-rover-code/tree/osr-ROS/Arduino/' not found"

I was able to download the code by checking out the repository and selecting the correct branch.

I'm unclear on what you are referring to in the ROS/RPi installation instructions, but it sounds like this may be a different issue. It would probably be better to track this as such?]

Also, in the initial ROS/RPi installation instructions in:
http://wiki.ros.org/ROSberryPi/Installing%20ROS%20Kinetic%20on%20the%20Raspberry%20Pi
if you go so far as section 4.1 Updating the Workspace which refers you to Ubuntu source install instructions:
http://wiki.ros.org/kinetic/Installation/Source
they refer to the "desktop-full" variant. Our instructions call for the "ros_comm" variant so
replace "desktop-full" with "ros_comm" as in -
$ mv -i kinetic-desktop-full-wet.rosinstall kinetic-desktop-full-wet.rosinstall.old
becomes
$ mv -i kinetic-ros_comm-wet.rosinstall kinetic-ros_comm-wet.rosinstall.old
note where hyphen "-" vs underscore "_"

Jim Phelan
Mars Rover project lead
Houston Robotics / now renamed USAi Labs

Re: Test Code for Raspberry Pi:
When trying to test the Raspberry Pi, you get error messages if it's not connected to the rover as it's expecting to connect to peripherals. However *pabloriveracelerity* posted mock peripherals to allow testing on his Github:
https://github.com/pabloriveracelerity/osr-rover-code/tree/pr/issue-12-init-tests/rover/tests

However, trying to run it as directed [very sparse directions and no e-mail link to ask] I get error messages:

```
pi@raspberrypi:~/osr/rover/tests $ python -m unittest discover
Eusage: python -m unittest [-h] [-x] [-s] [-b]
python -m unittest: error: unrecognized arguments: discover
Eusage: python -m unittest [-h] [-x] [-s] [-b]
python -m unittest: error: unrecognized arguments: discover
Eusage: python -m unittest [-h] [-x] [-s] [-b]
python -m unittest: error: unrecognized arguments: discover
EE
======================================================================
ERROR: test_connections (unittest.loader.ModuleImportFailure)
----------------------------------------------------------------------
ImportError: Failed to import test module: test_connections
Traceback (most recent call last):
  File "/usr/lib/python2.7/unittest/loader.py", line 254, in _find_tests
    module = self._get_module_from_name(name)
  File "/usr/lib/python2.7/unittest/loader.py", line 232, in _get_module_from_name
    __import__(name)
  File "/home/pi/osr/rover/tests/test_connections.py", line 15, in <module>
    from unittest.mock import patch
ImportError: No module named mock


======================================================================
ERROR: test_bluetooth_argument_is_false_by_default
(test_arguments.TestArguments)
Is the bluetooth argument False by default?
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/pi/osr/rover/tests/test_arguments.py", line 21, in setUp
    self.arguments = Arguments()
  File "/home/pi/osr/rover/arguments.py", line 19, in __init__
    self.bt_flag = parser.parse_args().bt
  File "/usr/lib/python2.7/argparse.py", line 1704, in parse_args
    self.error(msg % ' '.join(argv))
  File "/usr/lib/python2.7/argparse.py", line 2374, in error
    self.exit(2, _('%s: error: %s\n') % (self.prog, message))
  File "/usr/lib/python2.7/argparse.py", line 2362, in exit
    _sys.exit(status)
SystemExit: 2

======================================================================
ERROR: test_unix_argument_is_false_by_default (test_arguments.TestArguments)
Is the unix argument False by default?
```

```
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/pi/osr/rover/tests/test_arguments.py", line 21, in setUp
    self.arguments = Arguments()
  File "/home/pi/osr/rover/arguments.py", line 19, in __init__
    self.bt_flag = parser.parse_args().bt
  File "/usr/lib/python2.7/argparse.py", line 1704, in parse_args
    self.error(msg % ' '.join(argv))
  File "/usr/lib/python2.7/argparse.py", line 2374, in error
    self.exit(2, _('%s: error: %s\n') % (self.prog, message))
  File "/usr/lib/python2.7/argparse.py", line 2362, in exit
    _sys.exit(status)
SystemExit: 2

======================================================================
ERROR: test_xbox_argument_is_false_by_default (test_arguments.TestArguments)
Is the xbox argument False by default?
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/pi/osr/rover/tests/test_arguments.py", line 21, in setUp
    self.arguments = Arguments()
  File "/home/pi/osr/rover/arguments.py", line 19, in __init__
    self.bt_flag = parser.parse_args().bt
  File "/usr/lib/python2.7/argparse.py", line 1704, in parse_args
    self.error(msg % ' '.join(argv))
  File "/usr/lib/python2.7/argparse.py", line 2374, in error
    self.exit(2, _('%s: error: %s\n') % (self.prog, message))
  File "/usr/lib/python2.7/argparse.py", line 2362, in exit
    _sys.exit(status)
SystemExit: 2

======================================================================
ERROR: test_config (unittest.loader.ModuleImportFailure)
----------------------------------------------------------------------
ImportError: Failed to import test module: test_config
Traceback (most recent call last):
  File "/usr/lib/python2.7/unittest/loader.py", line 254, in _find_tests
    module = self._get_module_from_name(name)
  File "/usr/lib/python2.7/unittest/loader.py", line 232, in _get_module_from_name
    __import__(name)
  File "/home/pi/osr/rover/tests/test_config.py", line 15
    config_path = f'{dir_path}{os.path.sep}{filename}'
                                          ^
SyntaxError: invalid syntax
```

```
----------------------------------------------------------------------
Ran 5 tests in 0.039s

FAILED (errors=5)
pi@raspberrypi:~/osr/rover/tests $ python3 -m unittest discover
usage: python3 -m unittest [-h] [-x] [-s] [-b]
python3 -m unittest: error: unrecognized arguments: discover
Eusage: python3 -m unittest [-h] [-x] [-s] [-b]
python3 -m unittest: error: unrecognized arguments: discover
Eusage: python3 -m unittest [-h] [-x] [-s] [-b]
python3 -m unittest: error: unrecognized arguments: discover
EE.....
======================================================================
ERROR: test_bluetooth_argument_is_false_by_default
(test_arguments.TestArguments)
Is the bluetooth argument False by default?
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/pi/osr/rover/tests/test_arguments.py", line 21, in setUp
    self.arguments = Arguments()
  File "/home/pi/osr/rover/arguments.py", line 19, in __init__
    self.bt_flag = parser.parse_args().bt
  File "/usr/lib/python3.5/argparse.py", line 1738, in parse_args
    self.error(msg % ' '.join(argv))
  File "/usr/lib/python3.5/argparse.py", line 2394, in error
    self.exit(2, _('%(prog)s: error: %(message)s\n') % args)
  File "/usr/lib/python3.5/argparse.py", line 2381, in exit
    _sys.exit(status)
SystemExit: 2

======================================================================
ERROR: test_unix_argument_is_false_by_default (test_arguments.TestArguments)
Is the unix argument False by default?
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/pi/osr/rover/tests/test_arguments.py", line 21, in setUp
    self.arguments = Arguments()
  File "/home/pi/osr/rover/arguments.py", line 19, in __init__
    self.bt_flag = parser.parse_args().bt
  File "/usr/lib/python3.5/argparse.py", line 1738, in parse_args
    self.error(msg % ' '.join(argv))
  File "/usr/lib/python3.5/argparse.py", line 2394, in error
    self.exit(2, _('%(prog)s: error: %(message)s\n') % args)
```

```
  File "/usr/lib/python3.5/argparse.py", line 2381, in exit
    _sys.exit(status)
SystemExit: 2

======================================================================
ERROR: test_xbox_argument_is_false_by_default (test_arguments.TestArguments)
Is the xbox argument False by default?
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/pi/osr/rover/tests/test_arguments.py", line 21, in setUp
    self.arguments = Arguments()
  File "/home/pi/osr/rover/arguments.py", line 19, in __init__
    self.bt_flag = parser.parse_args().bt
  File "/usr/lib/python3.5/argparse.py", line 1738, in parse_args
    self.error(msg % ' '.join(argv))
  File "/usr/lib/python3.5/argparse.py", line 2394, in error
    self.exit(2, _('%(prog)s: error: %(message)s\n') % args)
  File "/usr/lib/python3.5/argparse.py", line 2381, in exit
    _sys.exit(status)
SystemExit: 2

======================================================================
ERROR: test_config (unittest.loader._FailedTest)
----------------------------------------------------------------------
ImportError: Failed to import test module: test_config
Traceback (most recent call last):
  File "/usr/lib/python3.5/unittest/loader.py", line 428, in _find_test_path
    module = self._get_module_from_name(name)
  File "/usr/lib/python3.5/unittest/loader.py", line 369, in _get_module_from_name
    __import__(name)
  File "/home/pi/osr/rover/tests/test_config.py", line 15
    config_path = f'{dir_path}{os.path.sep}{filename}'
                                                     ^
SyntaxError: invalid syntax


----------------------------------------------------------------------
Ran 16 tests in 0.261s

FAILED (errors=4)
```

2019.04.20
Google search for “python unittest” found:
https://docs.python.org/3/library/unittest.html

...but I didn't find it particularly helpful. I don't understand how to do testing.]

Re: Monolithic Printed Circuit Board (PCB):
The original PCB + Serial Splitter + Voltage Divider + Voltage Regulator (plus kill switch & power grid from battery/switch/voltmeter was a royal "spaghetti farm" of jumper wires and cables from the motors & encoders. So JPL designed a "monolithic" PCB that included 5V & 12V regulators, the serial splitter, voltage divider, plugs & terminals for the RoboClaws, ribbon cable to the R Pi and to a new Arduino for the LED display in the head. It did not include a kill switch, but I'll but that between the voltmeter & the PCB, I think. Otherwise we may alter the PCB with the help of a local fab. As of today (2019.04.28) the final instructions & schematics are not posted although the Bill of Materials & the Gerber files to order the PCB are. I'm not ready to order anything until I see the final versions. There is another Builders' Call on Wed 5/8/19 and the documents are supposed to be available the day before.

Re Wiring:
Thinking the Cat5 cable and noise was responsible for the failed calibrations [it wasn't, see above] I ripped out the Cat5 cable and replaced it with one two-conductor shielded power cable and one four-conductor shielded sensor cable for each motor [10 in all]. I used 6-position DuPont connectors in keeping with the existing drive motor connectors and rubber grommets in the U-channel holes instead of the awkward plastic net sleeve. NASA/JPL used and expensive custom D-sub cable connector on each side so they could easily remove the legs on each side for transport. I don't find removing the legs that convenient or helpful as it requires removing the two outer clamping collars on the main axle and loosening the clamping hubs for each differential bar arm. The disassembled rover is no more convenient to store and transport than the intact rover. So I will just push the cables through the body through one or more grommet holes.

Re Head:
With the addition of the Arduino for managing the LED display in the head, JPL created a new 3D printed head in just one piece plus backplane which I'll probably do in acrylic for viewing interest into the head.

2019.05.04 (Star Wars day "May the Fourth Be With You!")
Been working on the parts list for the reworked PCB and Arduino shield. Some of the major parts we already have from the old electronics board, but most of the minor parts & hardware are new. Ambivalent about adding the USB hubs for the RoboClaws and the Raspberry Pi and the required remake of front & back panels. The link to the PC fab house is still missing but I think I can do it. Finished wiring all the motors with the shielded cables last night. Put wheels back on rover at USAi Robotics (formerly Houston Robotics) to make it presentable for ComicPalooza next weekend. Starting printing the new one piece head and laser cut the head back plate & Arduino carrier in clear acrylic in the FabLab today. Put one of the new USAi Labs logos on the head back plate. ORDERED PARTS FROM DigiKey, McMaster-Carr, Pololu & PCBs from JLPCB! Expect all w/in a week.

2019.05.06
(to present to E. Junkins today before the call on 5/8/19)
Wish list for Open Source Rover Builders' Call and/or Github documentation:

Consolidate the documentation for the new electronics. Seems I keep looking all over and can't find documents that I've seen before. Seem to only find from outside links but not from root of OSR Github.

Finish documentation for assembly of monolithic board & Arduino shield.
Integration of monolithic board into chassis w/ batteries & USB hubs.
Integration of Arduino into new head.

PowerPoles for battery charging? (I am familiar w/ PowerPoles but see no reference in BOM or build docs. Battery already has plugs for charging, so?)

Why two 40-pin GPIO connectors on PCB? One connects the R Pi. The other?

Arduino code install. I've setup & installed my first Arduino sketch, but which rover code goes on Arduino? Setup?

How about test code for each subset of the rover, especially:
X-box controller to R Pi communication.
R Pi to Arduino communication.
Arduino LED test pattern.
R Pi to RoboClaw test code.
Simple drive motor system test - drive forward x seconds, stop, reverse, stop.
Simple steering motor system test - straight, right, straight, left, straight, stop.
ROS communication tests to demonstrate that each subsystem is working.

2019.05.13
The new monolithic & Arduino shield circuit boards arrived today from JLCPCB from China! The other parts arrived last week from McMaster-Carr, DigiKey and Pololu. The new 3D head and laser cut panels were done last weekend. This weekend Fidelity made an appearance in the MakerSpace section of ComicPalooze at the George R Brown Convention Center in Houston. Generated quite a bit of interest although not as much as the cosplay everywhere! Stayed up late Friday night putting the old circuit board back in for show. Spent this evening taking it back out & disassembling it and the old head. I now see that for sure I need the heat set screw posts for the head which I had deferred as the part # wasn't clear. I'm thinking it's the 4-40 version but need to clarify.

2019.05.18
JPL says the heat set inserts are the *tapered* ones and 6-32. I bought 6-32 & 4-40 types and they arrived yesterday. Looks to me like the 4-40s fit the 3D printed head not the 6-32s. Meanwhile, spent this week populating & soldering the new “monolithic” PCB. It’s done and beautiful. Posted some documentation errors I found of Tapatalk as I had difficulty creating an issue on GitHub. I posted 43 step-by-step photos on Tapatalk and got positive feedback from JPL. Next is PCB power testing. Then do the Arduino shield. For now there are no instructions but the thing itself speaks to me and the path seems self-evident, but I must confirm. I need fresh solder flux. My old gel flux is going dry & I prefer pen flux. Micro Center or Fry’s on the way home from robotics today.

2019.05.25
Last week was able to populate and solder the Arduino shield with the help of some PCB diagrams from JPL. Again the silkscreen labels were reversed. I tried using KiCAD to view the PCB connections but only got a slew of errors. I was able to completely wire up the “monolithic” Control Board to all the motor/encoders and tune all the RoboClaw motor controllers. Again, the silkscreen was reversed labeling the right motor connections left and vice-versa. Sat at the Meetup for USAi Labs (Houston Robotics) I expected to be frustrated as the next step was to download the code for the LED display head for the Arduino. The JPL instructions for this were, as yet, nonexistant. I pretty much know how to clone the github code into the R Pi. I’d also loaded a sample LED flash program into the Arduino using the Windows IDE via USB cable. But I didn’t know how to load multiple programs into the Arduino if that was even possible as it seemed to only load and run one program. Perhaps it compiled from several sources to then load. Had to await proper directions. Meanwhile I thought I’d try the sample display from the Adafruit web site for that display. Was able to follow their directions and eventually load it into the Arduino.

The next step was to power the head from the Control Board. Doing so failed to power the Arduino, but if I plugged in the USB from the laptop, it lit. So, not adeq. power from the Control Board. The LED display failed to light. The terminal block to the LED display read only millivolts. So, remembering the LED display was fairly high current, I bypassed the Arduino shield and hooked the display directly to the 12V out from the Control Board [this was a mistake, the LED display is high current, ie 2 Amps, but only 5v.] Doing this, the LED display lit up in a delightful rainbow of colors as intended!

Next problem was to solve why there was not adeq power at the Arduino shield. Tracing back to the JST connector on the Arduino shield, it appeared to be backwards with the power comming from the wrong end. Also, it read 7V where it should have read 12V. Removing the jump directly from the 12V out from the Control Board fixed this and it then read 12V. Desoldering the JST connector from the shield was a tedious process using desoldering braid and effectively destroyed the JST connector. Fortunately I had spares (knowing these were fragile). But also in the process it appeared the GND pad connection was destroyed with its connection to the 5V out terminal GND. This was fixed by a jumper wire between the JST pin and the 5V out

GND pin. We also noticed, but didn't fully appreciate, on the Control Board, the 5V trace to the JST connector had fried. The fix was to jumper to the GND pin of a nearby capacitor. But that had to wait til I got home.

When I got home I jumped the fried 5v trace between the 5v JST pin and the near pin of the nearly capacitor (C7). [I later discovered it was the WRONG end of the capacitor. It beeped continuity only briefly (until the capacitor charged) then stopped. The OTHER lead beeped continuously. I switched the jumper to the far end.] Trying again to re-create the display I smelled hot plastic & immediately cut power! Probing around I found the 5V connections to the LED display were shorted. Further probing revealed it to be at the display power connector itself. The shield and Control Board were ok separate from the display. I went so far as to unsolder the power pins from the LED display with a heat gun, but no apparent short between them. It had to be inside somewhere. Removing the front mask by removing 18 tiny screws only revealed the naked LEDs. No apparent way to access the other side of the LED PCB. It was fried. In retrospect, by the mis-applied 12V back when. That fried the Control Boad JST GND connection and may have contributed to the bad GND on the shield. Unfortunately, Adafruit is OUT OF STOCK. Subscribed to restock email. There is one through Amazon for almost twice the price I ordered but won't be here for a couple weeks!

Meanwhile, what to do? Forget the head. Let's see if we can get the rover to move. Since all the motors are wired and RoboClaw controllers tuned, maybe we can get the R Pi programmed and running this thing. Previous attempts failed. A post on the Open Source Rover (OSR) Github or Tapatalk forums suggested the way to get the current ROS (Robot Operating System) code installed to the Pi. Worth another try. An attempt to just run the main.py program on the Pi revealed a series of PRINT errors due to missing ()s. Apparently written for Python 2 instead of 3. I feel I need to start from scratch on the Pi which means a clean SD card and loading the latest Raspbian (vs Ubuntu?) OS. It will be a royal hemorrhoid, but should save conflict down the road, I hope! (If not create new once because of missing dependencies with the latest code!).

Starting w/ loading ROS on to the R Pi I took the easy route and loaded Ubuntu with ROS image from: https://downloads.ubiquityrobotics.com/pi.html using Win32 Disk Imager then moved the SD card to the R Pi.

Booting up with HDMI, keyboard & mouse "ubuntu" is the password.
Since this isn't Raspbian, I'm not as familiar with the OS and options.

As a second alternative, as I like Raspbian, I will first try the Raspbian Stretch Lite, ROS Kinetic and OpenCV 3.4 image recommended by fellow member David Williams:
https://docs.google.com/forms/d/e/1FAIpQLSeq5Sq8QyBLBJ5ncH46fNSh4MTQbexsWpMBKhSUpJjba1MIZQ/viewform
After downloading, unzipping, cc image to SD card & booting Pi, it comes up without a desktop and requests login. The standard pi/raspberry doesn't work. No email. No hint in the download. "rosbots!" is the password! Discovered a link that gave directions:

https://github.com/ROSbots/rosbots_setup_tools#use-our-existing-rosbots-raspbianroso pencv-image-after-youve-downloaded-it

2019.05.30
Tried working above ROSbots Raspbian Lite + ROS Kinetic + OpenCV and adding a desktop and other features I liked from regular Raspbian but without all the games and trash. I had it on a 16GB card and ran out of room in some memory space. Amazon had a sale on 200GB cards (limit 2) so bought a couple which arrived today. The other will be the Ubuntu + ROS but I'm not familiar/comfortable with Ubuntu so prefer Raspbian.

On the ROSbots Raspbian Lite + ROS Kinetic + OpenCV card working direct:
- $ sudo apt-get update
- $ sudo apt-get upgrade

Switch to SSH per the above like to more easily cut/paste commands.
- $ update_rosbots_git_repos
  - Latest bash source for rosbots updated!
- $ initialize_rosbots_image
  - Set up a new password
  - Changing password for pi.
  - (current) UNIX password: "*******"
  - New password: "**********"
- raspi-config window pops up
  - 1 Change User Password
    - password already changed
  - 2 Network options
    - N1 Hostname: "Fidelity"
    - N2 SSID: "*SSID-NAME*" / *wifi password*
    - N3 Network interface names
      - Would you like to enable predictable network interface names? <Yes>
  - 3 Boot Options
    - *No change*

4 Localization Options
I1 Change Locale
[*] en_US.UTF-8 UTF-8
I2 Change Timezone
US / Central
I3 Change Keyboard Layout
Generic 105 Intel / Other / English (US) /
default / No compose key
I4 Change Wi-fi Country
US United States
5 Interfacing Options
P1 Camera
P2 SSH
P3 VNC (*installation follows*)
P4 SPI
P5 I2C
P6 Serial
6 Overclock **NO**
7 Advanced Options
A1 Expand Filesystem
Root partition has been resized.
The filesystem will be enlarged upon the next reboot
A2 Overscan **NO**
A3 Memory Split
How much memory (MB) should the GPU have?
e.g. 16/32/64/128/256 **256**
A4 Audio
Choose the audio output
0 Auto
1 Force 3.5mm ('headphone') jack
2 Force HDMI
A5 Resolution **No change**
A6 Pixel Doubling **No change**
A7 GL Driver **No change**
Finish

$ sudo reboot

ROSCORE already running...
For all slaves, "export ROS_MASTER_URI=http://192.168.1.14:11311"

pi@Fidelity:~ $ rosnode list
/rosout
[this is the end of ROSbots instructions except for their modules]

Strange.... I tried installing lightDM to have a desktop. But I could SSH into the Pi and login with pi/Audacity1Now! successfully but directly with keyboard, mouse &

HDMI it accepts the login, goes blank for several seconds then comes back to the login screen. If I use a wrong password, it shows a red warning window in the login. I was unable to Google a solution. I Googled installing the regular desktop over Raspbian Lite but decided that for the rover, I really didn't need a desktop if I could SSH in. So, I reformatted the SD card. Rewrote the image and will reconfigure tomorrow. It will be much faster since I don't have to document every step again! Omitted this wasteful step above.

From JPL:
On your Raspberry Pi run the following commands in a terminal [this has errors]

```
mkdir -p ~/osr_ws/src
cd ~/osr_ws/src
git clone https://github.com/nasa-jpl/osr-rover-code/tree/osr-ROS/ROS
```

From Ian Reasor, OSR Github forum:

```
You'll actually need to run:
$ git clone https://github.com/nasa-jpl/osr-rover-code.git
$ git checkout osr-ROS
[this results in:
pi@Fidelity:~/osr_ws/src $ git checkout osr-ROS
fatal: Not a git repository (or any of the parent directories): .git]ls
$ git pull
```

Github post 2019.05.31:
@ireasor
Step 1 runs ok:

```
pi@Fidelity: /osr_ws/src $ git clone https://github.com/nasa-jpl/osr-rover-code.git
Cloning into 'osr-rover-code'...
remote: Enumerating objects: 206, done.
remote: Total 206 (delta 0), reused 0 (delta 0), pack-reused 206
Receiving objects: 100% (206/206), 206.63 KiB | 0 bytes/s, done.
Resolving deltas: 100% (67/67), done.
```

When I do step 2 above I get:

```
pi@Fidelity: /osr_ws/src $ git checkout osr-ROS
fatal: Not a git repository (or any of the parent directories): .gitn
```

Now what?

I also got the same error from the original instructions:

```
pi@Fidelity: /osr_ws/src $ git clone
https://github.com/nasa-jpl/osr-rover-code/tree/osr-ROS/ROS
Cloning into 'ROS'...
fatal: repository 'https://github.com/nasa-jpl/osr-rover-code/tree/osr-ROS/ROS/' not
found
```

Once the ROS code is eventually installed, what do I do so I can push which buttons on the Xbox controller to see the wheels move?

JHP

Meanwhile, moving on to the Arduino setup:
from: https://github.com/ireasor/osr-rover-code/tree/osr-ROS

Installing

On your development machine (not the Raspberry Pi), you need to install Arduino IDE from https://www.arduino.cc/en/Main/Software

You will also need to install a couple open source libraries, provided by Adafruit, that we use to manage the LED panel.

In the Arduino IDE open the Library Manager (on a Mac, it is in Tools > Manage Libraries)

In the Library Manager, search for and install the latest version of the "RGB Matrix Panel" library from Adafruit.

Then search for and install the latest version of the "Adafruit GFX" library.

Clone the OSR code repo
git clone https://github.com/nasa-jpl/osr-rover-code.git
git checkout osr-ROS [THIS STEP FAILS]
git pull

Build our custom library
Select the downloaded Arduino folder and create a ZIP out of it
Rename the zip file to OsrScreen.zip

Load the sketch onto the Arduino
Connect the Arduino to your development machine with a USB cable
Open the Arduino IDE
Select Sketch -> Include Library -> Add .ZIP Library
[Specified folder does not contain a valid library]
Select the OsrScreen.zip file
Go to File -> Examples -> Arduino -> Osr_Screen
Click the Upload button in the Sketch window

Arduino: 1.8.9 (Windows 10), Board: "Arduino/Genuino Uno"

In file included from C:\Users\Me\Documents\Arduino\libraries\Arduino\examples\Osr_Screen\Osr_Sc

C:\Users\Me\Documents\Arduino\libraries\Arduino/OsrScreen.h:4:55: fatal error: ../RGB-matrix-Panel-master/RGBmatrixPanel.h: No such file or directory

compilation terminated.

exit status 1
Error compiling for board Arduino/Genuino Uno.

This report would have more information with
"Show verbose output during compilation"
option enabled in File -> Preferences.

cc the RGBmatrixPanel.h into the complaining directory didn't fix it.

2019.06.01
Github reply from Ireasor:

If you take a look at the changes I made, you'll see that I had to modify OsrScreen.h and OsrScreen.cpp to remove "-master" from the path. Hopefully it works the same for you on Windows:

> Arduino/OsrScreen.cpp and
> Arduino/OsrScreen.h
>
> CHANGE: #include "../RGB-matrix-Panel-master/RGBmatrixPanel.h"
> TO: #include "../RGB_matrix_Panel/RGBmatrixPanel.h"

That helped, but then:

@ireasor Ian - REALLY appreciate your guidance as I'm a total Arduino newbie! Ok, I took out the -master part and reloaded the zip library. Now, even though the RGBmatrixPanel.h is in the very next tab on the Arduino IDE, I get the following error message near the bottom (Maybe I should turn off 'verbose'?):
Thanks again,
JHP

...In file included from
C:\Users\Me\Documents\Arduino\libraries\Arduino\examples\Osr_Screen\Osr_Screen.ino:12:0:

C:\Users\Me\Documents\Arduino\libraries\Arduino/OsrScreen.h:4:48: fatal error: ../RGB-matrix-Panel/RGBmatrixPanel.h: No such file or directory

2019.06.01

Shifting back to hardware. Since the Control Board was mounted on the top and the axle and side panels get in the way working from the open bottom, I put the bottom in and removed the top. Mounted the Control Board in the top. Attached it uprig ht in the back plane area to make it accessible for attaching all those wires. Routed & zip-tied cables in the U-channels, fed them through grommets in the lower side plate holes for esthetics and minimizing chaffing. (See pic on following page)

2019.06.02
With hardware at a standstill until software in installed, returned attention to the Arduino install problem. Looked again at the error message:

../**RGB-matrix-Panel**/RGBmatrixPanel.h: No such file or directory

Looking in the folder - C:\Users\Me\Documents\Arduino\libraries\Arduino there IS a folder **RGB_matrix_Panel** and it DOES contain RGBmatrixPanel.h. Oh! Wait!

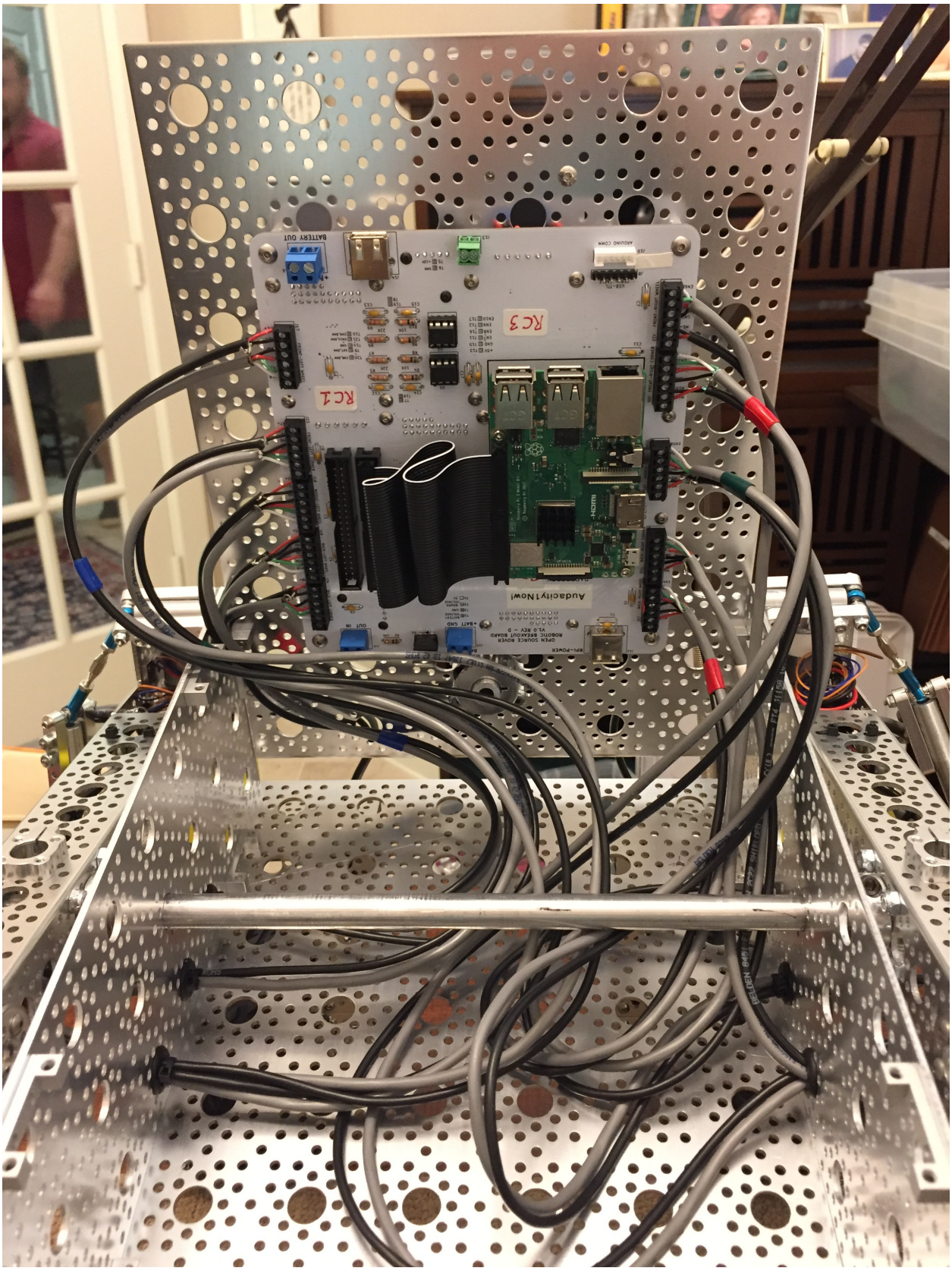
It's looking for **RGB-matrix-Panel** but it's finding **RGB_matrix_Panel**. A case of '-' vs '_'!
Renaming '_' to '-' and trying again.... This time I get:

Osr_Screen:10:28: error: RGBmatrixPanel.h: No such file or directory

but there IS RGBmatrixPanel.h in the

C:\Users\Me\Documents\Arduino\libraries\Arduino\RGB-matrix-Panel

folder!


BATTERY OUT
RC3
RC1
Audacity1Now!

2019.06.02
Need to learn Github. Trying tutorials:
https://www.youtube.com/watch?v=0fKg7e37bQE

and

https://www.youtube.com/watch?v=BCQHnlnPusY et seq.

Designed new Back Plate for rover. Pi slot no longer needed. Holes moved for symmetry. USAi Labs / Houston, Texas added.

Used Dual Side Mounts to create a hinge for the top plate for easier access.

Separated Power Switch from panel Multimeter and pondering location for Kill Switch.

Need to write an appropriate "Blessing of the Robot."

2019.06.08
With the help of Carlito did proper download of ROS code. Do this:

mkdir -p ~/osr_ws/src

cd ~/osr_ws/src

git clone https://github.com/nasa-jpl/osr-rover-code.git

To get the latest code go to ROS subdirectory -

cd ~/osr_ws/src/osr-rover-code

git checkout osr-ROS or git fetch

then

git pull

Running
There is a roslaunch file that will start all of the nodes in the ROS system, to run it:

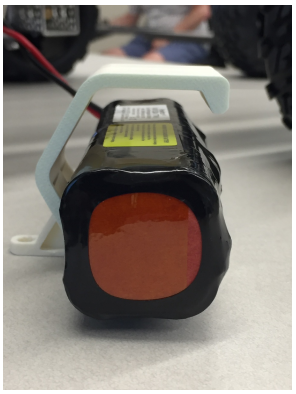
roslaunch osr_bringup osr.launch

Gives error message:

[osr.launch] is neither a launch file in package [ors_bringup] nor is [osr_bringup] a launch file name.

Suspect that some build needs to be done but not sure what....

3D printed Battery Holder only to find it too big:
Post from ejunkins NASA/JPL:



oh, I forgot that I have a slightly higher capacity battery in mine so it is a bigger battery. Can you measure the dimensions of the battery and I can give you an updated part for it?
@ericjunkins
https://www.batteryspace.com/li-ion-battery-14-8v-5-2ah-77wh-8a-rate-for-diving-light---un-38-3-passed.aspx
"Dimensions 1.6"(43mm) H x 1.6"(43mm) W x 5.5 "L(139mm)"
I measure from the 2 flat sides, avoiding the bumpy wire, side 1-15/32" a little less than the specs.
If the top of the holder were lowered to an inside jaws clearance of 1-15/32" it would be good. JHP

Download NASA & JPL logos.

2019.06.14
Closing github post:

@ericjunkins SOLVED - hardware issue.
Thanks for the schematics and PCB outlines!
[what version of KiCAD are y'all using? I suspect 4 and I have the latest 5 and there are compatibility issues which is why I have been unable to open them]

I tested TX and RX. They were not shorted to each other and connected to the appropriate RPi and Arduino pins.

However, during the initial build of the shield, I had the serial connector backwards and had to rework it.

In the process I damaged the board as the GND line from the Pi did not connect to the GND for the LED display but was successfully jumped for a functioning display. However, the 12V line from the Pi did not check as continuous with the TP1/12V nor the VIN pin of the Arduino. Jumping that failed to solve the upload problem.

So, I rebuilt the board using one of the excess PCBs. I didn't have a spare 16 pin connector so just used naked header pins and marked the side for the key slot. Fortunately I DID have a spare 6 pin connector.
BINGO!! I can now upload to the Ardino without disconnecting the shield!

JHP

response from ericjunkins:

@JHPHELAN I'm using 5.0.2

image

If you are having KiCAD problems can you open an issue for that so we can track that? Want to make sure we don't miss something.

Inorder to solve the Arduino download issue & the KiCAD issue I have uninstalled both Arduino IDE & KiCAD & will re-install them and try again following updated (but still incomplete?) instructions.

https://www.arduino.cc/en/Main/Software
Choose Windows installer for XP and up
download
run as administrator

Instructions from:
https://github.com/nasa-jpl/osr-rover-code/tree/osr-ROS

Arduino
Installing
On your development machine (not the Raspberry Pi), you need to install Arduino IDE from https://www.arduino.cc/en/Main/Software

You will also need to install a couple open source libraries, provided by Adafruit, that we use to manage the LED panel:

In the Arduino IDE open the Library Manager
(on a PC it is in Tools > Manage -Libraries)
In the Library Manager, search for and install the latest version of the "RGB Matrix Panel" library from Adafruit.

Then search for and install the latest version of the "Adafruit GFX" library.

Clone the OSR code repo:

[do following or download git files as a .zip which I did
https://github.com/nasa-jpl/osr-rover-code/tree/osr-ROS]

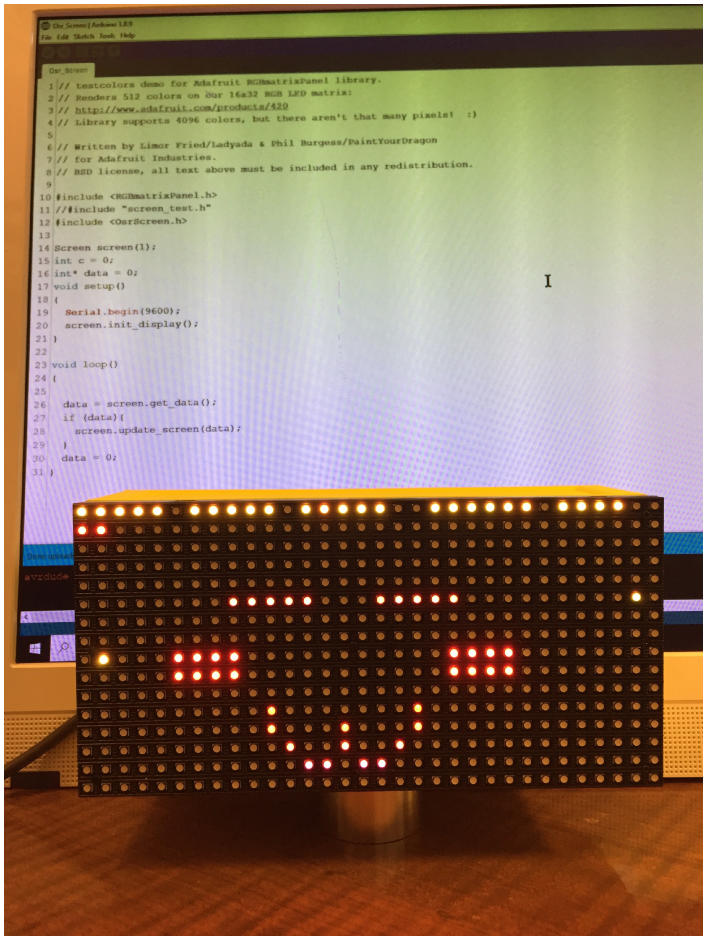
git clone https://github.com/nasa-jpl/osr-rover-code.git
git checkout osr-ROS
git pull

Build our custom library

Select the downloaded Arduino folder and create a ZIP out of it
Rename the zip file to OsrScreen.zip

Load the sketch onto the Arduino

Connect the Arduino to your development machine with a USB cable
Open the Arduino IDE
Select Sketch -> Include Library -> Add .ZIP Library
Select the OsrScreen.zip file
Go to File -> Examples -> Arduino -> Osr_Screen
Click the Upload button in the Sketch window



Discovered the LED display doesn't light when plugged into the Control Board and the battery is connected. From the 6-pin connector on the Control Board to the head, the 12V pins read 12V, but the 5V pins don't read 5V. Suspect board damage from prev 12V to the LED display. Will have to trace the Control Board & perhaps jump the 5V from elsewhere. Just pray I don't have to rebuild the whole Control Board as I had to the Arduino shield!!

2019.06.15
Remarkably productive weekend: Arduino IDE reinstalled & code successfully uploaded & run. LED presents a sort-of face & other lines and dots. KiCAD reinstalled & rover PCB files successfully opened. Electrical problems on Control Board corrected with help of KiCAD files. (Jumper for GND moved from 5V to GND where it belongs on JLC connector on Control Board. Jumper added to 5V for the JLC connector. The PCB at the JLC connector got fried when I applied 12V to the LED instead of 5V during testing. Source for 5V & GND were opposite ends of nearby capacitor). Front plate milled to fit & installed. NASA & JPL logos added to USAi Labs logo on head back plate. All electrical systems working!

Now, just to figure out how to get the ROS code to install & run.... Hoping for 4th of July parade!!

2019.06.16
Working on learning ROS. Suspect there's something that needs compiling/building in the ROS code supplied by NASA/JPL which is still a work in progress and poorly documented.

Following A Gentle Introduction to ROS by JasonM. O'Kane.

> *p11: Adding the ROS repository As root, create a file called*
> */etc/apt/sources.list.d/ros-latest.list*
> *containing a single line:*
> *deb http://packages.ros.org/ros/ubuntu trusty main*

As ROS is already installed I checkd to see if this was already there.

> $ cat /etc/apt/sources.list.d/ros-latest.list
> $ deb http://packages.ros.org/ros/ubuntu stretch main

so it's there but a newer version.

> *If you are unsure of which Ubuntu version you're using, you can find out using this*
> *command:*
> *lsb_release -a*
> *The output should show both a codename and a release number.*

pi@Fidelity:~ $ lsb_release -a
No LSB modules are available.
Distributor ID: Raspbian
Description: Raspbian GNU/Linux 9.9 (stretch)
Release: 9.9
Codename: stretch

OK, ROS is installed, but I am running Raspbian instead of Ubuntu. Is this a problem?

Let's try my ubuntu/ROS image
Login: ubuntu Password: ubuntu

On your Raspberry Pi run the following commands in a terminal
mkdir -p ~/osr_ws/src
cd ~/osr_ws/src
git clone https://github.com/nasa-jpl/osr-rover-code/tree/osr-ROS/ROS

```
ubuntu@ubiquityrobot:~$ ls
catkin_ws  Documents  Music    Public    Videos
Desktop    Downloads  Pictures  Templates
```

So ~/osr_ws/src is not already there. Let's see what is...

```
ubuntu@ubiquityrobot:~$ cd catkin_ws/
ubuntu@ubiquityrobot:~/catkin_ws$ ls
build  devel  src
ubuntu@ubiquityrobot:~/catkin_ws$ cd src
ubuntu@ubiquityrobot:~/catkin_ws/src$ ls
CMakeLists.txt  demos
ubuntu@ubiquityrobot:~/catkin_ws/src$ cd demos
ubuntu@ubiquityrobot:~/catkin_ws/src/demos$ ls
dnn_rotate  fiducial_follow  move_demo  partybot  README.md
```

...ok, demos could be interesting for another day. However, back to business...

```
ubuntu@ubiquityrobot:~$ mkdir -p ~/osr_ws/src
ubuntu@ubiquityrobot:~$ ls
catkin_ws  Documents  Music   Pictures  Templates
Desktop    Downloads  osr_ws  Public    Videos
ubuntu@ubiquityrobot:~$ cd osr_ws/
ubuntu@ubiquityrobot:~/osr_ws$ ls
src
ubuntu@ubiquityrobot:~/osr_ws$ cd src
ubuntu@ubiquityrobot:~/osr_ws/src$ ls
```

nothing there

ubuntu@ubiquityrobot:~/osr_ws/src$ **git clone**
https://github.com/nasa-jpl/osr-rover-code/tree/osr-ROS/ROS
Cloning into 'ROS'...
fatal: repository 'https://github.com/nasa-jpl/osr-rover-code/tree/osr-ROS/ROS/' not found

Same error as before. Per advice from ireasor fellow OSR builder:

You'll actually need to run:

$ git clone https://github.com/nasa-jpl/osr-rover-code.git
$ git checkout osr-ROS
$ git pull

ubuntu@ubiquityrobot:~/osr_ws/src$ **git clone**
https://github.com/nasa-jpl/osr-rover-code.git
Cloning into 'osr-rover-code'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 213 (delta 0), reused 2 (delta 0), pack-reused 206
Receiving objects: 100% (213/213), 216.07 KiB | 0 bytes/s, done.
Resolving deltas: 100% (67/67), done.
Checking connectivity... done.

ubuntu@ubiquityrobot:~/osr_ws/src$ **git checkout osr-ROS**
fatal: Not a git repository (or any of the parent directories): .git
ubuntu@ubiquityrobot:~/osr_ws/src$

As I recall from before, per Github thread:

@ireasor With the help of fellow USAi Labs member Charles, we were able to figure to
cd ~/osr_ws/src/osr-rover-code
then do either
git checkout osr-ROS
or
git fetch
then
git pull

But the next step (way down the documentation after the explanation of the various components):
"Running
There is a roslaunch file that will start all of the nodes in the ROS system, to run it:
roslaunch osr_bringup osr.launch"

Running this gives error message (regardless of which directory we run it from):
[osr.launch] is neither a launch file in package [ors_bringup] nor is [osr_bringup] a launch file name
so we suspect that some ROS build needs to be done, but aren't sure what.

JHP

```
ubuntu@ubiquityrobot:~/osr_ws/src$ ls
osr-rover-code

ubuntu@ubiquityrobot:~/osr_ws/src$ cd osr-rover-code/
ubuntu@ubiquityrobot:~/osr_ws/src/osr-rover-code$ ls
led  LICENSE.txt  README.md  requirements.txt  rover

ubuntu@ubiquityrobot:~/osr_ws/src/osr-rover-code$ cd rover
ubuntu@ubiquityrobot:~/osr_ws/src/osr-rover-code/rover$ ls
arguments.py    killAll.py          motorTest.py  robot.py       xbox.py
config.json     main.py             osr.py        rover.py
connections.py  motor_controller.py  roboclaw.py   serialTest.py
```

This looks like OLD python code. Need to find the ROS code.

Get back to directory - ~/osr_ws/src/osr-rover-code

```
ubuntu@ubiquityrobot:~/osr_ws/src/osr-rover-code$ git checkout osr-ROS
Branch osr-ROS set up to track remote branch osr-ROS from origin.
Switched to a new branch 'osr-ROS'
ubuntu@ubiquityrobot:~/osr_ws/src/osr-rover-code$ git pull
Already up-to-date.
ubuntu@ubiquityrobot:~/osr_ws/src/osr-rover-code$ git fetch
```

supposedly we're up to date..?
This time -

```
ubuntu@ubiquityrobot:~/osr_ws/src/osr-rover-code$ ls
Arduino  img  LICENSE.txt  README.md  ROS
```

instead of

```
ubuntu@ubiquityrobot:~/osr_ws/src/osr-rover-code$ ls
led  LICENSE.txt  README.md  requirements.txt  rover
```

Progress!?
Now what?

Post to Github issues:

@ireasor @ericjunkins @vssystemluba
[The current directions fail...]
On your Raspberry Pi run the following commands in a terminal
mkdir -p ~/osr_ws/src
cd ~/osr_ws/src
git clone https://github.com/nasa-jpl/osr-rover-code/tree/osr-ROS/ROS
[...with the the error:]
Cloning into 'ROS'...
fatal: repository 'https://github.com/nasa-jpl/osr-rover-code/tree/osr-ROS/ROS/' not found

[ireasor's directions work:]
$ mkdir -p ~/osr_ws/src
$ cd ~/osr_ws/src
$ git clone https://github.com/nasa-jpl/osr-rover-code.git
[the missing step here is to drill down from the home directory into the osr-rover-code directory:]
$ cd ~/osr_ws/src/osr-rover-code
[then switch branches with:]
$ git checkout osr-ROS
$ git pull
[this then shows successful listing:]
~/osr_ws/src/osr-rover-code$ ls
Arduino img LICENSE.txt README.md ROS
[which when drilling down to ROS gives:]
~/osr_ws/src/osr-rover-code/ROS$ ls
led_screen osr osr_bringup osr_msgs

[In later directions:]
Running
There is a roslaunch file that will start all of the nodes in the ROS system, to run it:
roslaunch osr_bringup osr.launch
[gives the error:]
$ **roslaunch osr_bringup osr.launch**
[osr.launch] is neither a launch file in package [osr_bringup] nor is [osr_bringup] a launch file name
The traceback for the exception was written to the log file

[Here my ignorance of ROS comes out. I SUSPECT that at this point we need to do some sort of build or make or source command but I'm still learning. Any guidance here??]

Jim Phelan
Mars rover project lead / USAi Labs (formerly Houston Robotics)

Reply from ericjunkins/JPL:

> Yeah there is a little bit to Setup the ROS system etc. We'll try and get these done before the builders call this coming week

Back to following A Gentle Introduction to ROS by JasonM. O'Kane for further clues:

p13
Setting up rosdep systemwide After installing the ROS packages, you'll need to execute this command:

> sudo rosdep init

This is a one-time initialization step; once ROS is working correctly, many users will not need to revisit rosdep.

> ubuntu@ubiquityrobot:~/osr_ws/src/osr-rover-code/ROS$ **sudo rosdep init**
> [sudo] password for ubuntu:
> ERROR: default sources list file already exists:
> /etc/ros/rosdep/sources.list.d/20-default.list
> Please delete if you wish to re-initialize

OK, apparently that's already been done.

> 2.2 Configuring your account
> Whether you install ROS yourself or use a computer on which ROS is already installed, there are two important configuration steps thatmust be done within the account of every user that wants to use ROS.
> Setting up rosdep in a user account First, you must initialize the rosdep system in your account, using this command:
>
> > rosdep update
>
> This command stores some files in your home directory, in a subdirectory called .ros. It generally only needs to be done once.

Go to home directory:

> $ **cd**
> $ **rosdep update**
>
> ubuntu@ubiquityrobot:~$ **rosdep update**
> reading in sources list data from /etc/ros/rosdep/sources.list.d
> Hit
> https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/osx-homebrew.yaml

Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/base.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/python.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/ruby.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/releases/fuerte.yaml
Hit
https://raw.githubusercontent.com/UbiquityRobotics/rosdep/master/raspberry-pi.yaml
Query rosdistro index
https://raw.githubusercontent.com/ros/rosdistro/master/index-v4.yaml
Skip end-of-life distro "ardent"
Add distro "bouncy"
Add distro "crystal"
Add distro "dashing"
Skip end-of-life distro "groovy"
Skip end-of-life distro "hydro"
Skip end-of-life distro "indigo"
Skip end-of-life distro "jade"
Add distro "kinetic"
Add distro "lunar"
Add distro "melodic"
updated cache in /home/ubuntu/.ros/rosdep/sources.cache

Not sure all these obsolete and future ROS distros are necessary.

> Setting environment variables ROS relies on a few environment variables to locate the files it needs. To set these environment variables, you'll need to execute the setup.bash script that ROS provides, using this command:
>
> source /opt/ros/indigo/setup.bash

but in my case indigo > kinetic

> **source /opt/ros/kinetic/setup.bash**

You can then confirm that the environment variables are set correctly using a command like this:

> **export | grep ROS**

ubuntu@ubiquityrobot:~$ export grep | ROS
ROS: command not found

[this may have worked if I had the "|" in the right place, but I went ahead and...]

ubuntu@ubiquityrobot:~$ source /opt/ros/kinetic/setup.bash
ubuntu@ubiquityrobot:~$ export | grep ROS

```
declare -x OLDPWD="/home/ubuntu/osr_ws/src/osr-rover-code/ROS"
declare -x ROSLISP_PACKAGE_DIRECTORIES=""
declare -x ROS_DISTRO="kinetic"
declare -x ROS_ETC_DIR="/opt/ros/kinetic/etc/ros"
declare -x ROS_HOSTNAME="ubiquityrobot.local"
declare -x ROS_MASTER_URI="http://ubiquityrobot.local:11311"
declare -x ROS_PACKAGE_PATH="/opt/ros/kinetic/share"
declare -x ROS_PARALLEL_JOBS="-j2"
declare -x ROS_ROOT="/opt/ros/kinetic/share/ros"
declare -x ROS_VERSION="1"
```

So that step is / has been done.

> If you get “command not found” errors from the ROS commands introduced later in this chapter, the most likely reason is that setup.bash has not been run in your current shell.
>
> Note, however, that the steps listed above apply only to the current shell. It would work perfectly well to simply execute that source command each time you start a new shell in which you’d like to execute ROS commands. However, this is both annoying and remarkably easy to forget, especially when you consider that the modular design of many ROS systems often calls for several different commands to execute concurrently, each in a separate terminal.
>
> Thus, you’llwant to configure your account to run the setup.bash script automatically, each time you start a new shell.
>
> To do this, edit the file named .bashrc in your home directory, and put the above source command at the bottom.
>
> ~$ **cat .bashrc**
> ...
> source /opt/ros/kinetic/setup.bash
> source /home/ubuntu/catkin_ws/devel/setup.bash
> source /etc/ubiquity/env.sh
> export ROS_PARALLEL_JOBS=-j2 # Limit the number of compile threads due to memory limits

So that’s already been done.

> $ **roslaunch osr_bringup osr.launch**

still fails.

Now book wants to do examples with turtlesim. Let’s see if we have it in this image:

ubuntu@ubiquityrobot:~$ **sudo find / -name turtle***
[sudo] password for ubuntu:
/opt/ros/kinetic/include/turtlesim
...and many others

Ok, so we've got the turtle!

Starting turtlesim In **three separate terminals**, execute these three commands:

roscore
rosrun turtlesim turtlesim_node
rosrun turtlesim turtle_teleop_key

ubuntu@ubiquityrobot:~$ **roscore**
... logging to
/home/ubuntu/.ros/log/70de2ede-d0dc-11e5-a130-0f3179f626c1/roslaunch-ubiqu
ityrobot-3380.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ubiquityrobot.local:40619/
ros_comm version 1.12.14

SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.14

NODES

roscore cannot run as another roscore/master is already running.
Please kill other roscore/master processes before relaunching.
The ROS_MASTER_URI is http://ubiquityrobot.local:11311/
The traceback for the exception was written to the log file

ubuntu@ubiquityrobot:~$ rosrun turtlesim turtlesim_node
QXcbConnection: Could not connect to display
Aborted

Oops, forgot to open SECOND terminal!
Can't run on Putty terminals:

ubuntu@ubiquityrobot:~$ **rosrun turtlesim turtlesim_node**
QXcbConnection: Could not connect to display
Aborted

Can't connect to R Pi w/ VNC or Remote Desktop.

Running from desktop on Pi

roscore	skip as already running
[open second terminal]	
rosrun turtlesim turtlesim_node	opens turtle window
[open third terminal]	
rosrun turtlesim turtle_teleop_key	opend keypad control window

drive turtle around in a pretty pattern.

2.4 Packages
All ROS software is organized into packages. A ROS package is a coherent collection of files, generally including both executables and supporting files, that serves a specific purpose. In the example, we used two executables called turtlesim_node and turtle_teleop_key, both of which aremembers of the turtlesim package.

It is not an overstatement to say that all ROS software is part of one package or another. Importantly, this includes newprograms that you create. We'll see how to create new packages in Section 3.1. In the meantime, ROS provides several commands for interacting with installed packages.

You can obtain a list of all of the installed ROS packages
using this command:

rospack list

ubuntu@ubiquityrobot:~$ **rospack list**
actionlib /opt/ros/kinetic/share/actionlib
actionlib_msgs /opt/ros/kinetic/share/actionlib_msgs
actionlib_tutorials /opt/ros/kinetic/share/actionlib_tutorials
...a VERY long list of packages...

An important exception is that, for most packages—specifically, those that have been updated to use the new **catkin build system**—compiled executables are not stored in the package directory, but in a separate standardized directory hierarchy.

Ah ha!!

For packages installed by apt-get, this hierarchy is rooted at */opt/ros/kinetic*
Executables are stored in the *lib* subdirectory under this root

no mention of osr-rover-code there.

automatically generated include files are stored inside the *include* subdirectory

no mention of osr-rover-code there.

To find the directory of a single package, use the rospack find command:

rospack find package-name

ubuntu@ubiquityrobot:~$ **rospack find osr-rover-code**
[rospack] Error: package 'osr-rover-code' not found

So clearly the package has not been built/made/sourced yet.

Of course, there may be times when you don't know (or can't remember) the complete name of the package that you're interested in. In these cases, it's quite convenient that rospack supports tab completion for package names. For example, you could type

rospack find turtle

and, **before pressing Enter, press the Tab key twice [quickly]** to see a list of all of the installed ROS packages whose names start with turtle.

$ **rospack find osr**[tab][tab]

comes up with nothing

Inspecting a package To view the files in a package directory, use a command like this:

rosls *package-name*

~$ **rosls turtle**[tab][tab]
turtle_actionlib/ turtlesim/ turtle_tf/ turtle_tf2/

ubuntu@ubiquityrobot:~$ **rosls turtlesim**
cmake images msg package.xml srv

If you'd like to "go to" a package directory, you can change the current directory to a particular package, using a command like this:

roscd *package-name*

ubuntu@ubiquityrobot:~$ **roscd turtlesim**
ubuntu@ubiquityrobot:/opt/ros/kinetic/share/turtlesim$

2.5 The master

So far we've talked primarily about files, and how they are organized into packages. Let's shift gears and talk now about how to actually execute some ROS software. One of the basic goals of ROS is to enable roboticists to design software as a collection of small,mostly independent programs called nodes that all run at the same time. For this to work, those nodes must be able to communicate with one another. The part of ROS that facilitates this communication is called the ROSmaster. To start the master, use this command:

roscore

We've already seen this in action in the turtlesim example. For once, there is no additional complexity to worry about: roscore does not take any arguments, and does not need to be configured. You should allow the master to continue running for the entire time that you're using ROS. One reasonable workflow is to **start roscore in one terminal, then open other terminals for your "real" work**. There are not many reasons to stop roscore, except when you've finished working with ROS. When you reach that point, **you can stop the master by typing Ctrl-C in its terminal**.

Most ROS nodes connect to the master when they start up, and do not attempt to reconnect if that connection fails later on. Therefore, if you stop roscore, any other nodes running at the time will be unable to establish new connections, even if you restart roscore later

The **roscore** command shown here is used to explicitly start the ROS master. In Chapter 6, we'll learn about a tool called **roslaunch** whose purpose is to start many nodes at once; this tool is smart enough to start a master if none is running, but will also happily use an existing master if there is one.

Starting nodes

The basic command to create a node (also known as "running a ROS program") is rosrun:

rosrun *package-name executable-name*

There's nothing "magical" about rosrun: It's just a shell script that understands ROS's file organization well enough to know where to look for executables by their package names. Once it finds the program you ask for, rosrun executes that program normally. For example, if you really wanted to, you could execute turtlesim_node directly, just like any other program:

/opt/ros/indigo/lib/turtlesim/turtlesim_node

The work of registering with the master to become a ROS node happens inside the program, not in rosrun.

Listing nodes ROS provides a few ways to get information about the nodes that are running at any particular time. To get a list of running nodes, try this command:

rosnode list

ubuntu@ubiquityrobot:~$ **rosnode list**
/controller_spawner
/joint_state_publisher
/joy_node
/motor_node
/robot_state_publisher
/rosbridge_ws/rosapi
/rosbridge_ws/rosbridge_websocket
/rosbridge_wss/rosapi
/rosbridge_wss/rosbridge_websocket
/rosout
/teleop_twist_joy
/tf2_web_republisher
/turtlesim

If you do this after executing the commands from Section 2.3, you'll see a list of three nodes:

/rosout
/teleop_turtle [note difference from my system]
/turtlesim

The /rosout node is a special node that is started automatically by roscore. Its purpose is somewhat similar to the standard output

Inspecting a node You can get some information about a particular node using this command:

rosnode info *node-name*

ubuntu@ubiquityrobot:~$ **rosnode info turtlesim**

Node [/turtlesim]
Publications:
 * /rosout [rosgraph_msgs/Log]
 * /statistics [rosgraph_msgs/TopicStatistics]
 * /turtle1/color_sensor [turtlesim/Color]
 * /turtle1/pose [turtlesim/Pose]

Subscriptions:
 * /turtle1/cmd_vel [unknown type]

Services:

```
 * /clear
 * /kill
 * /reset
 * /spawn
 * /turtle1/set_pen
 * /turtle1/teleport_absolute
 * /turtle1/teleport_relative
 * /turtlesim/get_loggers
 * /turtlesim/set_logger_level

contacting node http://ubiquityrobot.local:33215/ ...
ERROR: Communication with node[http://ubiquityrobot.local:33215/] failed!
```

Error may be because turtlesim isn't running?

```
ubuntu@ubiquityrobot:~$ rosnode info turtlesim
--------------------------------------------------------------------------------
Node [/turtlesim]
Publications:
 * /rosout [rosgraph_msgs/Log]
 * /statistics [rosgraph_msgs/TopicStatistics]
 * /turtle1/color_sensor [turtlesim/Color]
 * /turtle1/pose [turtlesim/Pose]

Subscriptions:
 * /turtle1/cmd_vel [geometry_msgs/Twist]

Services:
 * /clear
 * /kill
 * /reset
 * /spawn
 * /turtle1/set_pen
 * /turtle1/teleport_absolute
 * /turtle1/teleport_relative
 * /turtlesim/get_loggers
 * /turtlesim/set_logger_level


contacting node http://ubiquityrobot.local:34637/ ...
Pid: 4773
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound
    * transport: TCPROS
```

* topic: /turtle1/cmd_vel
  * to: /teleop_turtle (http://ubiquityrobot.local:41453/)
  * direction: inbound
  * transport: TCPROS

Killing a node
To kill a node you can use this command:

rosnode kill *node-name*

Unlike killing and restarting the master, killing and restarting a node usually does not have a major impact on other nodes; even for nodes that are exchanging messages, those connections would be dropped when the node is killed and reestablished when the node restarts.

You can also kill a node using the usual Ctrl-C technique. However, that method may not give the node a chance to unregister itself from the master. A symptom of this problem is that the killed node may still be listed by rosnode list for a while. This is harmless, butmightmake itmore difficult to tell what's going on. To remove dead nodes fromthe list, you can use this command:

rosnode cleanup

2.7 Topics andmessages
In our turtlesim example, it's clear that the teleoperation node and the simulator node must be talking to each other somehow. Otherwise, how would the turtle, which lives in the latter node, know when to move in response to your key presses, which are collected by the former node? The primary mechanism that ROS nodes use to communicate is to send **messages**. Messages in ROS are organized into named **topics**. The idea is that a node that wants to share information will **publish** messages on the appropriate topic or topics; a node that wants to receive information will **subscribe** to the topic or topics that it's interested in. The ROSmaster takes care of ensuring that publishers and subscribers can find each other; the messages themselves are sent directly frompublisher to subscriber.

2.7.1 Viewing the graph
This idea is probably easiest to see graphically, and the easiestway to visualize the publishsubscribe relationships between ROS nodes is to use this command:

rqt_graph

In this name, the r is for ROS, and the qt refers to the Qt GUI

Have to display on R Pi desktop, not w/ Putty as is a graphic.

2019.06.20
Attempting build & launch of updated ROS code from:
https://github.com/nasa-jpl/open-source-rover/blob/ros-rework/Software/Software%20Steps.pdf
[NOTE: in prev & later directions the generic “catkin_ws” is called “osr_ws”]
1.4 Downloading the Rover Code
✔
1.5 Building the Rover Code to work with your ROS installation
✔Move to the directory containing the main OSR logic:
cd /home/pi/osr/ROS
✔Create a catkin workspace directory: (read more here):
/home/pi/osr/ROS $ **mkdir /home/pi/catkin_ws**
✗Copy the source files to your catkin workspace:
/home/pi/osr/ROS $ **mv led_screen/ osr/ osr_bringup/ osr_msgs/ src**
mv: target 'scr' is not a directory
✔Need to specify target as catkin_ws/src:
/home/pi/osr/ROS $ **mv led_screen/ osr/ osr_bringup/ osr_msgs/ ~/catkin_ws/src**
✔Move to your catkin workspace:
cd /home/pi/catkin_ws
✔Run a "catkin make" of the OSR workspace:
/home/pi/catkin_ws $ **catkin make**
...very long build list. Success.

✔Move to your built catkin workspace: [You’re already there?]
cd /home/pi/catkin_ws/
✗ Run the roslaunch command
(read more about roslaunch at http://wiki.ros.org/roslaunch):
~/catkin_ws$ **roslaunch src/osr_bringup/launch/osr.launch**
This should read:
~/catkin_ws$ **roslaunch osr_bringup osr.launch**
... logging to /home/ubuntu/.ros/log/7033a20c-d0dc-11e5-9d60-b750b66e2ac7/roslaunch-ubiquityrobot-3086.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ubiquityrobot.local:45643/

SUMMARY
========

PARAMETERS
* /accel: 0,0,0,0
* /battery_high: 18

```
 * /battery_low: 11
 * /baud_rate: 115200
 * /enc_max: 1480,1755,1240,1230
 * /enc_min: 550,420,360,325
 * /joy_node/autorepeat_rate: 1.0
 * /joy_node/coalesce_interval: 0.05
 * /mech_dist: 7.254,10.5,10.5,1...
 * /motor_controller_addresses: 128,129,130,131,132
 * /motor_controller_device: /dev/serial0
 * /qpps: 0,0,0,0
 * /rosdistro: kinetic
 * /rosversion: 1.12.14

NODES
  /
    joy_node (joy/joy_node)
    joystick (osr/joystick.py)
    motor_controller (osr/motor_controller.py)
    rover (osr/rover.py)

ROS_MASTER_URI=http://ubiquityrobot.local:11311

ERROR: cannot launch node of type [osr/motor_controller.py]: can't locate node
[motor_controller.py] in package [osr]
ERROR: cannot launch node of type [osr/joystick.py]: can't locate node [joystick.py] in
package [osr]
ERROR: cannot launch node of type [osr/rover.py]: can't locate node [rover.py] in
package [osr]
process[joy_node-4]: started with pid [3103]
[ERROR] [1561070410.392290674]: Couldn't open joystick /dev/input/js0. Will retry
every second.

^C[joy_node-4] killing on exit
shutting down processing monitor...
... shutting down processing monitor complete
done
```

~/catkin_ws$ **find ~/ -name motor_controller.py**
/home/ubuntu/catkin_ws/src/osr/scripts/motor_controller.py
/home/ubuntu/osr_ws/src/osr-rover-code/ROS/osr/scripts/motor_controller.py

~/catkin_ws$ **find ~/ -name joystick.py**
/home/ubuntu/catkin_ws/src/osr/scripts/joystick.py
/home/ubuntu/osr_ws/src/osr-rover-code/ROS/osr/scripts/joystick.py

~/catkin_ws$ **find ~/ -name rover.py**

/home/ubuntu/catkin_ws/src/osr/scripts/rover.py
/home/ubuntu/osr_ws/src/osr-rover-code/ROS/osr/scripts/rover.py

So, the *.py programs appear to be there, but maybe in the wrong directory or not set up as nodes?

2019.06.21
Response from NASA/JPL Github issue post:

> *I suspect it is because the instructions currently omit the step to make the files executable as well.*
>
> *Navigate to each of the files and do the following*
> *sudo chmod +x <filename>*
>
> *@vssystemluba I notice a few other things that seem a little off. There should be '_' in the package names not spaces. For instance 'osr_bringup' not 'osr bringup'. This is ROS convention to use underscore for package names. The roslaunch command should also be:* [this is a failure of cut/paste to carry the "_" char]
>
> *roslaunch osr_bringup osr.launch*
>
> *If the catkin_ws is properly sourced then it will find the packages and launch files*
>
> *The following error:*
> *[ERROR] [1561070410.392290674]: Couldn't open joystick /dev/input/js0. Will retry every second.*
> *Makes sense unless you currently have a controller paired to the system and it is registered as /dev/input/js0. As soon as you connect the xbox controller this will go away*

Not sure which .py should be chmod'ed: catkin_ws or osr_ws. Will try catkin_ws first.

```
ubuntu@ubiquityrobot:~/catkin_ws/src/osr/scripts$ sudo chmod +x joystick.py
ubuntu@ubiquityrobot:~/catkin_ws/src/osr/scripts$ sudo chmod +x motor_controller.py
ubuntu@ubiquityrobot:~/catkin_ws/src/osr/scripts$ sudo chmod +x roboclaw.py
ubuntu@ubiquityrobot:~/catkin_ws/src/osr/scripts$ sudo chmod +x
        roboclaw_wrapper.py
ubuntu@ubiquityrobot:~/catkin_ws/src/osr/scripts$ sudo chmod +x robot.py
ubuntu@ubiquityrobot:~/catkin_ws/src/osr/scripts$ sudo chmod +x rover.py
ubuntu@ubiquityrobot:~/catkin_ws/src/osr/scripts$ ls -l
total 56
-rwxrwxr-x 1 ubuntu ubuntu  2621 Jun 20 21:31 joystick.py
-rwxrwxr-x 1 ubuntu ubuntu  1823 Jun 20 21:31 motor_controller.py
-rwxrwxr-x 1 ubuntu ubuntu 26630 Jun 20 21:31 roboclaw.py
-rwxrwxr-x 1 ubuntu ubuntu  7651 Jun 20 21:31 roboclaw_wrapper.py
```

-rwxrwxr-x 1 ubuntu ubuntu 6734 Jun 20 21:31 robot.py
-rwxrwxr-x 1 ubuntu ubuntu 1029 Jun 20 21:31 rover.py

Github post yesterday:

> *When I follow the steps re serial communication with the Pi I get:*
> *ubuntu@ubiquityrobot:~$ ls -l /dev/serial**
> *lrwxrwxrwx 1 root root 7 Feb 11 2016 /dev/serial0 -> ttyAMA0*
> *lrwxrwxrwx 1 root root 5 Feb 11 2016 /dev/serial1 -> ttyS0*
>
> *Does it matter that ttyS0 is on serial1 or does it HAVE to be serial0?*
>
> *JHP*

Response from JPL:

Hm, that's interesting. Are you using a Pi 3 for sure, or did you grab a Pi 2? I believe the ttyAMA0 and ttyS0 swapped between Pi 2 and Pi 3. [It's a Pi 3B+]

The second thing to check is that your sudo raspi-config steps match those outlined in the Software Steps.pdf document. Specifically, make sure your login shell is NOT available over serial (set to "No"), and that the serial port hardware is enabled (set to "Yes"). [set correctly]

The short story is that you need to use /dev/ttyS0 (so long as ttyS0 is mapped to GPIO pins 14 and 15 and not the bluetooth device). The above /dev/serial0 and /dev/serial1 are aliases to the actual hardware. If you've verified that the raspi-config configuration is correct and you've added the console=tty1 to /boot/cmdline.txt, you can go ahead and try to do the serial comms. If something fails, you may need to revisit this issue.

It's a Raspberry Pi 3 Model B+
The serial login shell is disabled
The serial interface is enabled
building on Ubuntu makes it ttyS1, using Raspbian it's ttyS0

2019.06.23

I *thought* the R Pi acted as the Xbox receiver via Bluetooth, but JPL said I need the USB receiver. Yes, it is in the parts list. Had forgotten that when Charles provided the Xbox controller. So yesterday ordered my own controller & receiver. Should get the controller today and the receiver by 2019.07.05. Noticed today there is a micro USB connector on the controller. Plugging it in to the R Pi made the 'X' light up. Moving the controls did nothing. Checked the freshly booted Pi and so no rover nodes. Launched the rover:

> ubuntu@ubiquityrobot:~$ **roslaunch osr_bringup osr.launch**

```
[ERROR] [1561317379.361845]: bad callback: <function joy_callback at
0x7638bfb0>
Traceback (most recent call last):
  File "/opt/ros/kinetic/lib/python2.7/dist-packages/rospy/topics.py", line 750, in
_invoke_callback
    cb(msg)
  File "/home/ubuntu/catkin_ws/src/osr/scripts/rover.py", line 16, in joy_callback
    cmds = MotorCommands()
NameError: global name 'MotorCommands' is not defined
```

let's see if the prev mentioned js0 for the Xbox controller/receiver is there:

```
ubuntu@ubiquityrobot:~$ ls -s /dev/input
total 0
0 by-id  0 by-path  0 event0  0 event1  0 event2  0 js0  0 mice
```

by adding a mouse I get:

```
ubuntu@ubiquityrobot:~$ ls -s /dev/input
total 0
0 by-id    0 event0  0 event2  0 js0   0 mouse0
0 by-path  0 event1  0 event3  0 mice
```

then adding a keyboard I get:

```
ubuntu@ubiquityrobot:~$ ls -s /dev/input
total 0
0 by-id    0 event0  0 event2  0 event4  0 mice
0 by-path  0 event1  0 event3  0 js0     0 mouse0
```

removing the xbox controller I get:

```
ubuntu@ubiquityrobot:~$ ls -s /dev/input
total 0
0 by-id  0 by-path  0 event0  0 event1  0 event3  0 event4  0 mice  0 mouse0
```

so it apparently recognized js0 as the xbox controller
now it's just a matter of the unrecognized software.
Let's try installing exactly the way NASA/JPL did it....

2019.06.23
Install by NASA/JPL Raspbian/ROS build pathway
(instead of Ubuntu/ROS image or Raspbian/ROS/OpenCV image)

Get latest NOOBS image: https://www.raspberrypi.org/downloads/noobs/
Unzip latest version to:
C:\Users\Me\Documents\My Downloads\Robotics\Raspberry Pi\NOOBS_v3_0_1

Format 64GB card w/ SD Card Formatter, XC format won't format to FAT32!
So go here:
https://www.raspberrypi.org/documentation/installation/sdxc_formatting.md
reformat BLANK using SD Card Formatter
then reformat to FAT32 using FAT32FORMAT / guiformat.exe

cc unzipped files to "NASAJPLROS" 64GB SD card.
Move SD card to R Pi & boot.

Reconfigure to taste following:
Raspberry Pi MNT Droid Project Documentation CONFIGURATION.pdf
enable SSH to allow upload via Filezilla to downloads
of USAi Labs round OSR logo.png

"Once the Pi reboots, open up a terminal again and look at the serial devices:
pi@Fidelity: ~ $ **ls -l /dev/serial***
lrwxrwxrwx 1 root root 5 Jun 23 21:11 /dev/serial0 -> ttyS0
lrwxrwxrwx 1 root root 7 Jun 23 21:11 /dev/serial1 -> ttyAMA0

Make sure that this shows serial0 -> ttyS0 . If it does not, ensure that you have followed every step in this tutorial in order."

It worked differently under Ubuntu the serial0 and serial1 were reversed.

"Next, edit the /boot/cmdline.txt
pi@Fidelity: ~ $ **sudo nano /boot/cmdline.txt**
Change ONLY the part with "console = ..." to read "console=tty1" and remove any other instance where it references console."

Already correct.

SAVE IMAGE OF SETUP BEFORE LOADING ROS AS A RESET POINT

Win32 Disk Imager SD Card > NASAJPLROS 2019.06.24.img

Next, we will install ROS (and speci cally, the version called 'Kinetic'). To install ROS, follow the installation instructions at:
http://wiki.ros.org/ROSberryPi/Installing%20ROS%20Kinetic%20on%20the%20Raspberry%20Pi

ROSberryPi/ Installing ROS Kinetic on the Raspberry Pi

/!\ Note: If you're using the Raspberry Pi 2 or 3 it is faster and easier to use the standard ARM installation instructions here:
NO, this is Ubuntu installation, not Raspbian. Don't use.

Raspbian Stretch:
Note: Instructions are similar to Jessie, but you must first install dirmngr:
pi@Fidelity: ~ $ **sudo apt-get install dirmngr**

pi@Fidelity: ~ $ **sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'**

$ **sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116**

$ **sudo apt-get update**
$ **sudo apt-get upgrade**

$ **sudo apt-get install -y python-rosdep python-rosinstall-generator python-wstool python-rosinstall build-essential cmake**
WARNING: The following packages cannot be authenticated!
python-catkin-pkg-modules python-catkin-pkg python-rospkg-modules python-rospkg python-rosdistro-modules python-rosdistro python-rosdep python-vcstools python-wstool python-rosinstall python-rosinstall-generator
E: There were unauthenticated packages and -y was used without --allow-unauthenticated

Try again with...
$ **sudo apt-get install -y python-rosdep python-rosinstall-generator python-wstool python-rosinstall build-essential cmake --allow-unauthenticated**

$ **sudo rosdep init**
Wrote /etc/ros/rosdep/sources.list.d/20-default.list
Recommended: please run

rosdep update

$ **rosdep update**

$ **mkdir -p ~/ros_catkin_ws**
$ **cd ~/ros_catkin_ws**

ROS-Comm: (recommended) ROS package, build, and communication libraries. No GUI tools.

$ **rosinstall_generator ros_comm --rosdistro kinetic --deps --wet-only --tar > kinetic-ros_comm-wet.rosinstall**
$ **wstool init src kinetic-ros_comm-wet.rosinstall**

3.2.1 Unavailable Dependencies

mkdir -p ~/ros_catkin_ws/external_src
cd ~/ros_catkin_ws/external_src
wget
http://sourceforge.net/projects/assimp/files/assimp-3.1/assimp-3.1.1_no_test_models.zip/download -O assimp-3.1.1_no_test_models.zip
unzip assimp-3.1.1_no_test_models.zip
cd assimp-3.1.1
cmake .
make
sudo make install

3.2.2 Resolving Dependencies with rosdep

$ cd ~/ros_catkin_ws
$ rosdep install -y --from-paths src --ignore-src --rosdistro kinetic -r --os=debian:stretch

3.3 Building the catkin Workspace

$ **sudo ./src/catkin/bin/catkin_make_isolated --install -DCMAKE_BUILD_TYPE=Release --install-space /opt/ros/kinetic**

Very long build, then it freezes here:

...
[43%] Building CXX object
CMakeFiles/roscpp.dir/src/libros/single_subscriber_publisher.cpp.o
Eventually the battery died overnight.

It is highly likely that the compilation will fail with an "internal compiler error" caused by memory exhaustion. A quick fix for this is to add swap space to the Pi

and recompile. If the error persists try building with the -j2 option instead of the default -j4 option:

Well, no “internal compiler error” but let’s add swap space:

http://raspberrypimaker.com/adding-swap-to-the-raspberrypi/

... or maybe not:

Let me put this right up front. Using swap space on SD cards and USB flash drives is not a good idea. It creates more problems than it solves. To start, it will be too slow to be useful. Second, and more importantly, Linux swapping can cause many write cycles which will tax your SD/flash drive quickly causing failures. I only use swap when I have a USB hard disk

If the error persists try building with the -j2 option instead of the default -j4 option:

sudo ./src/catkin/bin/catkin_make_isolated --install
-DCMAKE_BUILD_TYPE=Release --install-space /opt/ros/kinetic -j2

<== Finished processing package [52 of 52]: 'rosbag'
pi@Fidelity:~/ros_catkin_ws $

Now ROS should be installed! Remember to source the new installation:
$ source /opt/ros/kinetic/setup.bash
Or optionally source the setup.bash in the ~/.bashrc, so that ROS environment variables are automatically added to your bash session every time a new shell is launched:
$ echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc

I like #2

2019.06.25

4. Maintaining a Source Checkout
4.1 Updating the Workspace
See the Ubuntu source install instructions for steps on updating the ros_catkin_ws workspace. The same steps should apply to the Raspberry Pi.
http://wiki.ros.org/kinetic/Installation/Source

*To update your workspace, first move your existing rosinstall file so that it doesn't get overwritten, and generate an updated version. For simplicity, we will cover the *destop-full* variant. For other variants, update the filenames and rosinstall_generator arguments appropriately.*

“desktop” >> “ros_comm”

“kinetic-desktop” >> “kinetic-ros_comm”

*$ mv -i kinetic-**desktop-full**-wet.rosinstall kinetic-**desktop-full**-wet.rosinstall.old*
becomes
$ mv -I kinetic-**ros_comm**-wet.rosinstall kinetic-**ros_comm**-wet.rosinstall.old

$ *rosinstall_generator **desktop_full** --rosdistro kinetic --deps --wet-only --tar >*
*kinetic-**desktop-full**-wet.rosinstall*
becomes
$ rosinstall_generator **ros_comm** --rosdistro kinetic --deps --wet-only --tar >
kinetic-**ros_comm**-wet.rosinstall

Then, compare the new rosinstall file to the old version to see which packages will be updated:

*$ diff -u kinetic-**desktop-full**-wet.rosinstall kinetic-**desktop-full**-wet.rosinstall.old*
becomes
$ diff -u kinetic-**ros_com**-wet.rosinstall kinetic-**ros_comm**-wet.rosinstall.old
...apparently no changes since was just freshly installed.

If you're satisfied with these changes, incorporate the new rosinstall file into the workspace and update your workspace:

$ wstool merge -t src kinetic-desktop-full-wet.rosinstall
becomes
$ **wstool merge -t src kinetic-ros_comm-wet.rosinstall**
Merge caused no change, no new elements found
$ **wstool update -t src**
...Done

3.2 Rebuild your workspace
Now that the workspace is up to date with the latest sources, rebuild it:

$ **./src/catkin/bin/catkin_make_isolated --install**
permission denied
$ **sudo ./src/catkin/bin/catkin_make_isolated --install**
...long install of 52 packages...
<== Finished processing package [52 of 52]: 'rosbag'

Once your workspace has been rebuilt, you should source the setup files again:
$ source ~/ros_catkin_ws/install_isolated/setup.bash

1.4 Downloading the Rover Code

Move to your home directory:

pi@Fidelity: ~ $ **cd /home/pi**

Clone the rover code repository:

pi@Fidelity: ~ $ **git clone https://github.com/nasa-jpl/osr-rover-code osr**

Cloning into 'osr'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 216 (delta 2), reused 2 (delta 0), pack-reused 206
Receiving objects: 100% (216/216), 216.84 KiB | 0 bytes/s, done.
Resolving deltas: 100% (69/69), done.

Move into the directory you just cloned:

pi@Fidelity: ~ $ **cd osr**

Check out the osr-ROS branch:

pi@Fidelity: ~ $ **git checkout osr-ROS**

Branch osr-ROS set up to track remote branch osr-ROS from origin.
Switched to a new branch 'osr-ROS'

Pull to make sure you have the latest code:

pi@Fidelity: ~ $ **git pull**

Already up-to-date.

1.5 Building the Rover Code to work with your ROS installation

Move to the directory containing the main OSR logic:

pi@Fidelity: ~ $ **cd /home/pi/osr/ROS**

Create a catkin workspace directory:

pi@Fidelity: ~ $ **mkdir -p /home/pi/catkin_ws/src**

Copy the source files to your catkin workspace:

pi@Fidelity: ~ $ **mv led_screen/ osr/ osr_bringup/ osr_msgs/**
/home/pi/catkin_ws/src

Move to your catkin workspace:

pi@Fidelity: ~ $ **cd /home/pi/catkin_ws**

Run a "catkin make" of the OSR workspace:

pi@Fidelity: ~/catkin_ws $ **catkin_make**

```
CMake Warning at
/home/pi/ros_catkin_ws/install_isolated/share/catkin/cmake/catkinConfig.cmake:
76 (find_package):
  Could not find a package configuration file provided by "sensor_msgs" with
  any of the following names:

    sensor_msgsConfig.cmake
    sensor_msgs-config.cmake

  Add the installation prefix of "sensor_msgs" to CMAKE_PREFIX_PATH or set
  "sensor_msgs_DIR" to a directory containing one of the above files.  If
  "sensor_msgs" provides a separate development package or SDK, be sure it
  has been installed.
Call Stack (most recent call first):
  osr_msgs/CMakeLists.txt:10 (find_package)


-- Could not find the required component 'sensor_msgs'. The following CMake
error indicates that you either need to install the package with the same name or
change your environment so that it can be found.
CMake Error at
/home/pi/ros_catkin_ws/install_isolated/share/catkin/cmake/catkinConfig.cmake:
83 (find_package):
  Could not find a package configuration file provided by "sensor_msgs" with
  any of the following names:

    sensor_msgsConfig.cmake
    sensor_msgs-config.cmake

  Add the installation prefix of "sensor_msgs" to CMAKE_PREFIX_PATH or set
  "sensor_msgs_DIR" to a directory containing one of the above files.  If
  "sensor_msgs" provides a separate development package or SDK, be sure it
  has been installed.
Call Stack (most recent call first):
  osr_msgs/CMakeLists.txt:10 (find_package)


-- Configuring incomplete, errors occurred!
See also "/home/pi/catkin_ws/build/CMakeFiles/CMakeOutput.log".
See also "/home/pi/catkin_ws/build/CMakeFiles/CMakeError.log".
Invoking "cmake" failed
```

pi@Fidelity:~/catkin_ws $ **find ~/ -name sensor_msgs***
/home/pi/ros_catkin_ws/src/genmsg/test/files/sensor_msgs

so clearly something in the sensor_msgs world exists.
Posted issure to NASA/JPL OSR Github 2019.06.25

ericjunkins reply:

> *I'm seeing a little bit of inconsistencies in path stuff here:*
>
> *~/catkin_ws $ catkin_make*
>
> */home/pi/ros_catkin_ws/install_isolated/share/catkin/cmake/catkinConfig.cmake: 76 (find_package):*
>
> */home/pi/ros_catkin_ws/src/genmsg/test/files/sensor_msgs*
>
> *Some of these have a "ros_catkin_ws" and some have a "catkin_ws".*
>
> *Sensor_msgs is a default package of ROS and if it's failing that it is because there is something not set right with the path of your system. can you verify path and folder names for me*

Let me try:

Move to your catkin workspace:
pi@Fidelity: ~ $ **cd /home/pi/ros_catkin_ws**
Run a "catkin make" of the OSR workspace:
pi@Fidelity: ~/ros_catkin_ws $ **catkin_make**

virtual memory exhausted: Cannot allocate memory
ros_comm/roscpp/CMakeFiles/roscpp.dir/build.make:662: recipe for target 'ros_comm/roscpp/CMakeFiles/roscpp.dir/src/libros/service_client.cpp.o' failed
make[2]: *** [ros_comm/roscpp/CMakeFiles/roscpp.dir/src/libros/service_client.cpp.o] Error 1
make[2]: *** Waiting for unfinished jobs....
CMakeFiles/Makefile2:10511: recipe for target 'ros_comm/roscpp/CMakeFiles/roscpp.dir/all' failed
make[1]: *** [ros_comm/roscpp/CMakeFiles/roscpp.dir/all] Error 2
Makefile:138: recipe for target 'all' failed
make: *** [all] Error 2
Invoking "make -j4 -l4" failed
Let's try:
pi@Fidelity: ~/ros_catkin_ws $ **catkin_make -j2**
...
[100%] Built target record

Let's see if we can launch now:

pi@Fidelity:~ $ **roslaunch osr_bringup osr.launch**
[osr.launch] is neither a launch file in package [osr_bringup] nor is [osr_bringup] a launch file name
The traceback for the exception was written to the log file

Maybe we need to "source" something i.e.:

Once your workspace has been rebuilt, you should source the setup files again:

$ source ~/ros_catkin_ws/install_isolated/setup.bash

pi@Fidelity:~/ros_catkin_ws $ **source ~/ros_catkin_ws/install_isolated/setup.bash**

Then try:

pi@Fidelity:~ $ **roslaunch osr_bringup osr.launch**
[osr.launch] is neither a launch file in package [osr_bringup] nor is [osr_bringup] a launch file name
The traceback for the exception was written to the log file

2019.06.26
Posted to Github:

The directions say go to the "catkin_ws". Should it be "ros_catkin_ws"?
So tried:
pi@Fidelity: ~/ros_catkin_ws $ catkin_make
but got:
virtual memory exhausted: Cannot allocate memory
...
Invoking "make -j4 -l4" failed
So tried:
pi@Fidelity: ~/ros_catkin_ws $ catkin_make -j2
and got:
...
[100%] Built target record
So suggest updating directions to "ros_catkin_ws" and "-j2"

JHP

Then added:

... but when I try again (regardless from which directory I try):
$ roslaunch osr_bringup osr.launch
I still get the same error:

[osr.launch] is neither a launch file in package [osr_bringup] nor is [osr_bring up] a launch file name
The traceback for the exception was written to the log file

If I look for osr.launch I get:

pi@Fidelity:~/ros_catkin_ws $ find ~/ -name osr.launch
/home/pi/catkin_ws/src/osr_bringup/launch/osr.launch

but if I go to that directory I still get the same error.
What am I missing?
JHP

ericjunkins responded:

Truthfully the workspace can be called whatever you want it to be as long as you are consistent with the name. I usually name mine '< project>_ws', for example 'osr_ws'. But for simplicity's sake it should be 'catkin_ws'. I don't think it says anywhere in the docs to put the 'ros' in there. The '-j' option just specifies how many threads to use for the compilation, it should make no difference in the command working or not.

The roslaunch command is not specific to the directory, it is looking for that specific launch package in the current ros workspace. For instance in our case:

command: roslaunch
ros package: osr_bringup
launch file: osr.launch
This will work for any directory on the machine because it is looking for the osr_bringup package in what is set as the current $ROS_PATH. It then looks for the osr.launch launch file in that package. Your error is coming from not having the ros path looking in the correct place. Did you do this above comment? This is what sets when you open a new terminal to have the ros path look at the workspace you set up

"sourcing" the workspace should be done as part of the workspace setup as well.

http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment

2019.06.27
I responded:

I believe the catkin_ws was created by the ROS install.
Then the ros-catkin-ws was created by the OSR install.
(although it could be vice-versa, I'd have to recheck the instructions)
The -j2 was because of the "virtual memory exhausted: Cannot allocate memory" error.
Looking for ROS_PATH I get:
pi@Fidelity:~ $ ROS_PATH

-bash: ROS_PATH: command not found
pi@Fidelity:~ $ ros_path
-bash: ros_path: command not found
pi@Fidelity:~ $ ROS_path
-bash: ROS_path: command not found
pi@Fidelity:~ $ rospath
-bash: rospath: command not found
pi@Fidelity:~ $ ROSPATH
-bash: ROSPATH: command not found
pi@Fidelity:~ $ ROSpath
-bash: ROSpath: command not found
so somehow its not being created

JHP

ericjunkins responded:

okay let's be explicit step-by-step here and @vssystemluba can update the instructions to coincide with this. Let's start over with creating the workspace. I did these exact steps this morning on a new workspace I created.

mkdir -p ~/osr_ws/src

cd osr_ws

catkin_make

Copy the packages into the '~/osr_ws/src' folder. To be clear they should be
- osr_bringup
- osr_msgs
- osr
- led_screen

Add the following to the bottom of the bashrc script: sudo nano ~/.bashrc
- source /opt/ros/kinetic/setup.bash
- source ~/osr_ws/devel/setup.bash
- source ~/osr_ws/devel/setup.sh

Open a new terminal. echo $ROS_PACKAGE_PATH. It should be the following: /home/< youruser>/osr_ws/src:/opt/ros/kinetic/share

cd ~/osr_ws/src/osr/scripts

chmod +x motor_controller.py robot.py rover.py joystick.py

roslaunch osr_bringup osr.launch

ROS guide on creating catkin_ws:
http://wiki.ros.org/catkin/Tutorials/create_a_workspace
Udemy Tutorial ROS for Beginners (I HIGHLY recommend spending the $11 to take this course and learn ROS, it covers all the basics that you will need):
https://www.udemy.com/ros-for-beginners/

So, following the above:

pi@Fidelity:~ $ **mkdir -p ~/osr_ws/src**
pi@Fidelity:~ $ **cd osr_ws**
pi@Fidelity:~/osr_ws $ **catkin_make**
Base path: /home/pi/osr_ws
...
-- Build files have been written to: /home/pi/osr_ws/build
####
Running command: "make -j4 -l4" in "/home/pi/osr_ws/build"
####

pi@Fidelity:~/osr_ws $ **ls**
build devel src

success so far.

Copy the packages into the '~/osr_ws/src' folder. To be clear they should be
- osr_bringup
- osr_msgs
- osr
- led_screen

Previous instructions were:
Move to the directory containing the main OSR logic:
pi@Fidelity: ~ $ **cd /home/pi/osr/ROS**
Create a catkin workspace directory:
pi@Fidelity: ~/osr/ROS $ **mkdir -p /home/pi/catkin_ws/src**
Copy the source files to your catkin workspace:
pi@Fidelity: ~/osr/ROS $ **mv led_screen/ osr/ osr_bringup/ osr_msgs/ /home/pi/catkin_ws/src**

so now they would be:

Move to the directory containing the main OSR logic:
pi@Fidelity: ~ $ **cd /home/pi/osr/ROS**
Create a catkin workspace directory:
pi@Fidelity: ~/osr/ROS $ **mkdir -p /home/pi/osr_ws/src**

Copy the source files to your catkin workspace:
pi@Fidelity: ~/osr/ROS $

*** RATHER THAN REMAKE THE WHOLE SD CARD,
I TRANSLATED THE NEW DIRECTIONS BACK TO THE OLD DIRECTIONS
WITH THE APPROPRIATE ADDITIONS***

1 mkdir -p ~/osr_ws/src
use existing catkin_ws/src instead

2 cd osr_ws
move to catkin_ws insetead

3 catkin_make
already done in catkin_ws

4 Copy the packages into the '~/osr_ws/src' folder. To be clear they should be
- osr_bringup
- osr_msgs
- osr
- led_screen
already done in catkin_ws

5 Add the following to the bottom of the bashrc script: ***sudo nano ~/.bashrc***
- source /opt/ros/kinetic/setup.bash
- source ~/osr_ws/devel/setup.bash
- source ~/osr_ws/devel/setup.sh
Only the 1st one was in .bashrc. The other two added.

6 Open a new terminal. ***echo $ROS_PACKAGE_PATH****. It should be the following:*
/home/< youruser>/osr_ws/src:/opt/ros/kinetic/share

In mine I get:
pi@Fidelity:~ $ **echo $ROS_PACKAGE_PATH**
/home/pi/catkin_ws/src:/home/pi/ros_catkin_ws/install_isolated/share

7 cd ~/osr_ws/src/osr/scripts

8 chmod +x motor_controller.py robot.py rover.py joystick.py
To find my *.py I did:
pi@Fidelity:~ $ **find ~/ -name motor_controller.py**
/home/pi/catkin_ws/src/osr/scripts/motor_controller.py
So in /home/pi/catkin_ws/src/osr/scripts I did the chmod's

9 roslaunch osr_bringup osr.launch
and I get:

pi@Fidelity:~ $ **roslaunch osr_bringup osr.launch**
... logging to
/home/pi/.ros/log/6a327474-9941-11e9-aea8-b827eb2b674d/roslaunch-Fidelity-2263.lo
g
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://Fidelity:36053/

SUMMARY
========

PARAMETERS
 * /accel: 0,0,0,0
 * /battery_high: 18
 * /battery_low: 11
 * /baud_rate: 115200
 * /enc_max: 1480,1755,1240,1230
 * /enc_min: 550,420,360,325
 * /joy_node/autorepeat_rate: 1.0
 * /joy_node/coalesce_interval: 0.05
 * /mech_dist: 7.254,10.5,10.5,1...
 * /motor_controller_addresses: 128,129,130,131,132
 * /motor_controller_device: /dev/serial0
 * /qpps: 0,0,0,0
 * /rosdistro: kinetic
 * /rosversion: 1.12.14

NODES
  /
    joy_node (joy/joy_node)
    joystick (osr/joystick.py)
    motor_controller (osr/motor_controller.py)
    rover (osr/rover.py)

auto-starting new master
process[master]: started with pid [2273]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 6a327474-9941-11e9-aea8-b827eb2b674d
process[rosout-1]: started with pid [2286]
started core service [/rosout]

process[motor_controller-2]: started with pid [2293]
process[joystick-3]: started with pid [2301]
process[rover-4]: started with pid [2303]
ERROR: cannot launch node of type [joy/joy_node]: joy
ROS path [0]=/home/pi/ros_catkin_ws/install_isolated/share/ros
ROS path [1]=/home/pi/catkin_ws/src
ROS path [2]=/home/pi/ros_catkin_ws/install_isolated/share
Traceback (most recent call last):
  File "/home/pi/catkin_ws/src/osr/scripts/rover.py", line 3, in <module>
    from osr_msgs.msg import Joystick, Commands, Encoder
ImportError: No module named osr_msgs.msg

Looking for osr_msgs.msg:

pi@Fidelity:~ $ **sudo find / -name osr_msgs.msg**
comes up blank

It seems osr_msgs.msg is missing.

Posted above to Github this evening.

> @ericjunkins
>
> Is your main directory 'catkin_ws' or 'osr_ws'.
> It's catkin_ws as per Software Steps. There's no mention of osr_ws until this thread.
> In the rewrite, is it going to be osr_ws? Given this thread I'll assume osr_ws.
>
> I'm pretty sure the ros_catkin_ws came from the ROS install directions referenced in the Software Steps.
>
> I'm going to re-install from scratch, merging these additional instructions, and translate from the generic catkin_ws to the specific osr_ws.
>
> Additional confusion arises as the Raspberry Pi directions advise using the leaner ros-comm version of ROS while the ROS install directions assume desktop-full and translations have to be done there. I'll skip the update workspace part of the ROS install as nothing was updated after a clean install.
>
> I'll continue to document my progress step-by-step and share to assist in the Software Steps rewrite.
>
> JHP

2019.06.28

Following instructions in: 2019.06.20 Software Steps.pdf
1.1 Getting the Raspberry Pi running

Starting over from the saved configured Raspbian image:
NASAJPLROS 2019.06.24.img:

Created following:
https://projects.raspberrypi.org/en/projects/raspberry-pi-getting-started
Loading Raspbian NOOBS.

Pre-configured to the locale, and per my personal preferences listed in:
Raspberry Pi MNT Droid Project Documentation CONFIGURATION.pdf

1.2 Installing ROS
Even tho recently created, will update/upgrade
✔pi@Fidelity:~ $ **sudo apt-get update**
✔pi@Fidelity:~ $ **sudo apt-get upgrade**

follow the installation instructions at:
http://wiki.ros.org/ROSberryPi/Installing%20ROS%20Kinetic%20on%20the%20Raspberry%20Pi

2.1 Setup ROS Repositories
✔pi@Fidelity:~ $ **sudo apt-get install dirmngr**
✔pi@Fidelity:~ $ **sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'**
✔pi@Fidelity:~ $ **sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116**

Now, make sure your Debian package index is up-to-date:
$ sudo apt-get update
$ sudo apt-get upgrade
This was done earlier.

2.2 Install Bootstrap Dependencies

✔pi@Fidelity:~ $ **sudo apt-get install -y python-rosdep python-rosinstall-generator python-wstool python-rosinstall build-essential cmake**

2.3 Initializing rosdep
✔pi@Fidelity:~ $ **sudo rosdep init**
✔pi@Fidelity:~ $ **rosdep update**

3.1 Create a catkin Workspace

✔pi@Fidelity:~ $ **mkdir -p ~/ros_catkin_ws**

✔pi@Fidelity:~ $ **cd ~/ros_catkin_ws**

Here we have a choice of installs:

a) ROS-Comm: (recommended) ROS package, build, and communication libraries. No GUI tools.

b) Desktop: ROS, rqt, rviz, and robot-generic libraries

Initially I chose a) because of recommendation. But b) has "robot-genereid libraries which *may be* what was missing in the last build. This time choose b) which will make future instructions simpler as they assume desktop.

✗pi@Fidelity:~/ros_catkin_ws $ **rosinstall_generator desktop --rosdistro kinetic --deps --wet-only --tar > kinetic-desktop-wet.rosinstall**

This was interrupted mid download & Putty disconnected. Try again:

✔pi@Fidelity:~/ros_catkin_ws $ **rosinstall_generator desktop --rosdistro kinetic --deps --wet-only --tar > kinetic-desktop-wet.rosinstall**

This will add all of the catkin or wet packages in the given variant and then fetch the sources into the ~/ros_catkin_ws/src directory. The command will take a few minutes to download all of the core ROS packages into the src folder. The -j8 option downloads 8 packages in parallel.

Unclear what "This" refers to. There is no -J8 mentioned (default?). Prev this or something like it failed @ lack of swap memory. Perhaps better to use -j4 option from the beginning?

✔pi@Fidelity:~/ros_catkin_ws $ **wstool init src kinetic-desktop-wet.rosinstall -j4**

So far, only these two variants have been tested on the Raspberry Pi in Kinetic; however, more are defined in REP 131 such as robot, perception, etc. Just change the package path to the one you want, e.g., for robot do:

Since these variants contain robot apps, they should probably be installed also as they may be the source of the prev. missing apps.

✔pi@Fidelity:~/ros_catkin_ws $ **rosinstall_generator robot --rosdistro kinetic --deps --wet-only --tar > kinetic-robot-wet.rosinstall**

✗pi@Fidelity:~/ros_catkin_ws $ **wstool init src kinetic-robot-wet.rosinstall**
Error: There already is a workspace config file .rosinstall at "src". Use wstool install/modify.
Best guess as to what the error message means:
✗pi@Fidelity:~ $ **wstool install/modify src kinetic-robot-wet.rosinstall**

Error: unknown command: install/modify
wstool is a command to manipulate ROS workspaces. wstool replaces its predecessor rosws.
Official usage:
  wstool CMD [ARGS] [OPTIONS]
wstool will try to infer install path from context
Type 'wstool help' for usage.
help provide help for commands
init set up a directory as workspace
set add or changes one entry from your workspace config
merge merges your workspace with another config set
remove (rm) remove an entry from your workspace config, without deleting files
scrape interactively add all found unmanaged VCS subfolders to workspace
update (up) update or check out some of your config elements
info Overview of some entries
status (st) print the change status of files in some SCM controlled entries
diff (di) print a diff over some SCM controlled entries
foreach run shell command in given entries

So let's try:
✗pi@Fidelity:~/ros_catkin_ws $ **wstool init src kinetic-robot-wet.rosinstall /modify**

So there are some misdirections here. Skip it for now.

3.2 Resolve Dependencies
Before you can build your catkin workspace, you need to make sure that you have all the required dependencies. We use the rosdep tool for this, however, a couple of dependencies are not available in the repositories. They must be manually built first.

3.2.1 Unavailable Dependencies
Compilation of collada_urdf will fail per this issue.
You can provide a compatible version of Assimp (Open Asset Import Library) to fix this link error:
network error. Putty connection lost. Reopen.

✔pi@Fidelity:~ $ **mkdir -p ~/ros_catkin_ws/external_src**
✔pi@Fidelity:~ $ **cd ~/ros_catkin_ws/external_src**
✔pi@Fidelity:~/ros_catkin_ws/external_src $ **wget http://sourceforge.net/projects/assimp/files/assimp-3.1/assimp-3.1.1_no_test_models.zip/download -O assimp-3.1.1_no_test_models.zip**

✔pi@Fidelity:~/ros_catkin_ws/external_src $ **unzip assimp-3.1.1_no_test_models.zip**
✔pi@Fidelity:~/ros_catkin_ws/external_src $ **cd assimp-3.1.1**
✔pi@Fidelity:~/ros_catkin_ws/external_src/assimp-3.1.1 $ **cmake .**
✔pi@Fidelity:~/ros_catkin_ws/external_src/assimp-3.1.1 $ **make**
✔pi@Fidelity:~/ros_catkin_ws/external_src/assimp-3.1.1 $ **sudo make install**

Lost Putty connection. Reopened.

3.2.2 Resolving Dependencies with rosdep
The remaining dependencies should be resolved by running rosdep:
Raspbian Stretch:

✔pi@Fidelity:~ $ **cd ~/ros_catkin_ws**
✔pi@Fidelity:~/ros_catkin_ws $ **rosdep install -y --from-paths src --ignore-src --rosdistro kinetic -r --os=debian:stretch**
ERROR: the following packages/stacks could not have their rosdep keys resolved
to system dependencies:
opencv3: No definition of [ffmpeg] for OS version [stretch]
Continuing to install resolvable dependencies...
.....
#All required rosdeps installed successfully

3.3 Building the catkin Workspace
Once you have completed downloading the packages and have resolved the dependencies, you are ready to build the catkin packages.
Invoke catkin_make_isolated:

✗pi@Fidelity:~/ros_catkin_ws $ **sudo ./src/catkin/bin/catkin_make_isolated --install -DCMAKE_BUILD_TYPE=Release --install-space /opt/ros/kinetic**
Traceback (most recent call last):
  File "./src/catkin/bin/catkin_make_isolated", line 12, in <module>
    from catkin.builder import build_workspace_isolated
  File "./src/catkin/bin/../python/catkin/builder.py", line 66, in <module>
    from catkin_pkg.terminal_color import ansi, disable_ANSI_colors, fmt, sanitize
ImportError: No module named terminal_color

Is this because of the failure of install of opencv3 above??

Now ROS should be installed! Remember to source the new installation:
$ **source /opt/ros/kinetic/setup.bash**
Or optionally source the setup.bash in the ~/.bashrc, so that ROS environment variables are automatically added to your bash session every time a new shell is launched:
$ **echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc**
The second option seems better.

✔pi@Fidelity:~/ros_catkin_ws $ **echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc**

Let's look at .bashrc:
pi@Fidelity:~/ros_catkin_ws $ **cat ~/.bashrc**

...
source /opt/ros/kinetic/setup.bash

4. Maintaining a Source Checkout
4.1 Updating the Workspace
Skip this as it is a fresh install

4.2 Adding Released Packages
You may add additional packages to the installed ros workspace that have been released into the ros ecosystem. First, a new rosinstall file must be created including the new packages (Note, this can also be done at the initial install). For example, if we have installed ros_comm, but want to add ros_control and joystick_drivers, the command would be:

This could explain why the joystick didn't work before as it was a ros_comm install. This time it's a desktop install so *should* contain these packages? Skip.

Now to go back to the NASA/JPL Software Steps.pdr directions, updated per Github thread.

Let's see where we are:

pi@Fidelity:~/ros_catkin_ws $ echo $ROS_PACKAGE_PATH
blank
Clearly there's work to be done.

1.3 Setting up serial communication
go home:
pi@Fidelity:~/ros_catkin_ws $ **cd**
pi@Fidelity:~ $

✔pi@Fidelity:~ $ **sudo raspi-config**
Already done in initial setup for
login shell over serial no
serial port hardware enabled yes

✔pi@Fidelity:~ $ **ls -l /dev/serial***
lrwxrwxrwx 1 root root 5 Jun 24 18:57 /dev/serial0 -> ttyS0
lrwxrwxrwx 1 root root 7 Jun 24 18:57 /dev/serial1 -> ttyAMA0

✔pi@Fidelity:~ $ sudo nano /boot/cmdline.txt
Correct

1.4 Downloading the Rover Code
Move to your home directory: cd /home/pi
✔pi@Fidelity:~ $
already there
Clone the rover code repository: git clone https://github.com/nasa-jpl/osr-rover-code osr
✔pi@Fidelity:~ $ **git clone https://github.com/nasa-jpl/osr-rover-code osr**
Move into the directory you just cloned: cd osr
✔pi@Fidelity:~ $ **cd osr**
pi@Fidelity:~/osr $
Check out the osr-ROS branch: git checkout osr-ROS
✔pi@Fidelity:~/osr $ **git checkout osr-ROS**
Branch osr-ROS set up to track remote branch osr-ROS from origin.
Switched to a new branch 'osr-ROS'
Pull to make sure you have the latest code: git pull
✔pi@Fidelity:~/osr $ **git pull**
Already up-to-date.

1.5 Building the Rover Code to work with your ROS installation
Now, you must build the rover code packages so that ROS can run them on your Raspberry Pi. To do this, run the following commands in a terminal:
Move to the directory containing the main OSR logic: cd /home/pi/osr/ROS

✔pi@Fidelity:~/osr $ **cd /home/pi/osr/ROS**
pi@Fidelity:~/osr/ROS $
HOWEVER (this discovered below)
pi@Fidelity:~/osr/ROS $ **ls**
blank

Create a catkin workspace directory: mkdir -p /home/pi/catkin ws/src
But the new thread directions say: mkdir -p ~/osr_ws/src

pi@Fidelity:~/osr/ROS $ **mkdir -p ~/osr_ws/src**

Move to your catkin workspace: cd /home/pi/catkin_ws
New thread directions say: cd osr_ws

✗pi@Fidelity:~/osr/ROS $ **cd osr_ws**
-bash: cd: osr_ws: No such file or directory

Directions should read: cd ~/osr_ws

✔pi@Fidelity:~ $ **cd ~/osr_ws**
pi@Fidelity:~/osr_ws $

Copy the source files to your catkin workspace:
mv led screen/ osr/ osr bringup/ osr msgs/ /home/pi/catkin ws/src
But new directions say do this AFTER catkin_make.

catkin_make

pi@Fidelity:~/osr_ws $ **catkin_make**
-bash: catkin_make: command not found

So let's try moving the packages first...
*mv led screen/ osr/ osr bringup/ osr msgs/ /home/pi/**catkin ws/src***
but change the destination to **osr_ws/src** per new directions:

pi@Fidelity:~/osr_ws $ **mv led screen/ osr/ osr bringup/ osr msgs/ /home/pi/osr_ws/src**
mv: cannot stat 'led': No such file or directory
mv: cannot stat 'screen/': No such file or directory
mv: cannot stat 'osr/': No such file or directory
mv: cannot stat 'osr': No such file or directory
mv: cannot stat 'bringup/': No such file or directory
mv: cannot stat 'osr': No such file or directory
mv: cannot stat 'msgs/': No such file or directory

That's because we left the directory containing the main OSR logic: cd /home/pi/osr/ROS

pi@Fidelity:~/osr_ws $ **cd /home/pi/osr/ROS**

NOW cc over the packages:

pi@Fidelity:~/osr/ROS $ **mv led screen/ osr/ osr bringup/ osr msgs/ /home/pi/osr_ws/src**
pi@Fidelity:~/osr/ROS $ mv led screen/ osr/ osr bringup/ osr msgs/ /home/pi/osr_ws/src
mv: cannot stat 'led': No such file or directory
mv: cannot stat 'screen/': No such file or directory
mv: cannot stat 'osr': No such file or directory
mv: cannot stat 'bringup/': No such file or directory
mv: cannot stat 'osr': No such file or directory
mv: cannot stat 'msgs/': No such file or directory

Error because " _ " got lost in the copy/paste of the command. Edited to

pi@Fidelity:~/osr/ROS $ **mv led_screen/ osr/ osr_bringup/ osr_msgs/ /home/pi/osr_ws/src**

mv: cannot stat 'osr/': No such file or directory
pi@Fidelity:~/osr/ROS $ **ls**
blank

Error because packages are ALREADY in:

pi@Fidelity:~/osr_ws/src $ **ls**
led_screen osr osr_bringup osr_msgs

Run a "catkin make" of the OSR workspace: catkin make
New directions agree - catkin_make

pi@Fidelity:~/osr_ws/src $ **catkin_make**
-bash: catkin_make: command not found

Error because being run from ~/osr_ws/src instead of ~/osr_ws

pi@Fidelity:~/osr_ws/src $ **cd ..**
pi@Fidelity:~/osr_ws $

NOW catkin_make:
pi@Fidelity:~/osr_ws $ **catkin_make**
-bash: catkin_make: command not found

I'M STUMPED! THE PACKAGES ARE IN THE ~/osr_ws/src directory
BUT catkin_make fails as "command not found" ?!?!?!

2019.07.06

After discussing w/ David Williams, he ran into the same problem with missing 3 *.py files at the end. He did both NOOBS per the OSR instructions and the Ubuntu image option per the ROS install instructions.

A NEW version of Raspbian, Buster, just came out a week ago to make use of the additional features of the new Raspberry Pi 4:

https://www.raspberrypi.org/blog/buster-the-new-version-of-raspbian/

https://www.raspberrypi.org/blog/raspberry-pi-4-on-sale-now-from-35/

We decided to go with
1) Raspbian Buster via NOOBS offline & network install via .zip
2) ROS kinetic Desktop-Full
3) the OSR code

https://www.raspberrypi.org/downloads/
https://www.raspberrypi.org/downloads/noobs/
NOOBS_v3_1_1.zip
extract to C:\Users\Me\Documents\My Downloads\Robotics\Raspberry Pi\
NOOBS_v3_1_1
cc extracted files to reformatted 64GB SD card
insert in Raspberry Pi 3B+
boot
desktop appears
Resizing FAT partition
Please wait while NOOBS initializes
✖select Raspbian Full [RECOMMENDED]
Install (I),
Confirm YES
long install...
OS(es) Installed Successfully [OK]
reboots
desktop
Welcome to Raspberry Pi [Next]
Set Country
Country United States
Language Amereican English
Timezone Chicago
[Next]....
Change Password
new password
[Next]....

Set Up Screen
✔This screen shows a black border around the desktop
[Next]....
Select WiFi Network
NUTHOUSE 5G
[Next]....
Enter WiFi Password
****
[Next]....
Update Software
[Next]....
System is up to date [OK]
[Restart]

Raspberry Pi Configuration
System
Hostname: Fidelity
Interfaces
Enable: SSH, VNC, SPI, 12C, Serial Port
Disable: Camera, Serial Console, 1-wire, Remote GPIO
Performance
GPU Memory 64 > 128
Localisation
no change
[OK]
reboot

Note IP address of R Pi 192.168.1.13
Filezilla from PC > R Pi
C:\Users\Me\Documents\My Downloads\Robotics\USAi Labs\Logos\
Round Robotics Open Source Rover WHITE.png
>>
/home/pi/Downloads
Desktop - Right-click
Desktop
Layout Fit image onto screen
Picture pi/Downloads/Round Robotics Open Source Rover WHITE.png
Colour white #FFFFFF
Text Colour black #000000
[OK]

Desktop - Right-click
Create New, Empty File, “SYSTEM.txt”
open SYSTEM.txt
describe system
Raspbian NOOBS 3.1.1 Buster full,
ROS kinetic Desktop Full (pending install)
NASA/JPL Open Source Rover ROS (pending install)

Test VNC Viewer into R Pi
IP Address; Login; Password
exit

PuTTY into R Pi
IP Address; Login; Password
remain logged in to perform further installs via PuTTY
for ease of copy/paste documentation

save cc of image file of this SD card to shorten future installs in case of failure:
shutdown, power-off
remove SD card
> SD card adapter > PC
ignore all PC efforts to reformat
Win32 Disk Imager read SD card to “4 BusKinOSR 2019.07.06.img”
Will take ~55min
done
Pausing for tonight as this is a good place to stop....

Once the Pi reboots, open up a terminal again and look at the serial devices:

✔ls -l /dev/serial*

Make sure that this shows serial0 -> ttyS0 . If it does not, ensure that you have followed every step in this tutorial in order.

```
pi@Fidelity:~ $ ls -l /dev/serial*
lrwxrwxrwx 1 root root 5 Jul  6 20:13 /dev/serial0 -> ttyS0
lrwxrwxrwx 1 root root 7 Jul  6 20:13 /dev/serial1 -> ttyAMA0
```

Next, edit the /boot/cmdline.txt le:

✔sudo nano /boot/cmdline.txt

Change ONLY the part with "console = ..." to read "console=tty1" and remove any other instance where it references console.
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p7 rootfstype=ext4 elevator=$....

2019.07.07

ROS install plan based on
http://wiki.ros.org/ROSberryPi/Installing%20ROS%20Kinetic%20on%20the%20Raspberry%20Pi

Pass on the easy route via Ubuntu image as the OSR is based on Raspbian (although Stretch as opposed to the latest Buster)

It's not clear if this is required for Buster, but it is for Stretch and shouldn't do any harm if not needed?

2.1 Setup ROS Repositories
Raspbian Stretch:
Note: Instructions are similar to Jessie, but you must first install dirmngr:

> ✔**sudo apt-get install dirmngr**
>
> pi@Fidelity:~ $ sudo apt-get install dirmngr
> Reading package lists... Done
> Building dependency tree
> Reading state information... Done
> dirmngr is already the newest version (2.2.12-1+rpi1).
> The following packages were automatically installed and are no longer required:
> libboost-system1.62.0 libboost-thread1.62.0 libreoffice-gtk2
> Use 'sudo apt autoremove' to remove them.
> 0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
>
> pi@Fidelity:~ $ **sudo apt autoremove**
> Reading package lists... Done
> Building dependency tree
> Reading state information... Done
> The following packages will be REMOVED:
> libboost-system1.62.0 libboost-thread1.62.0 libreoffice-gtk2
> 0 upgraded, 0 newly installed, 3 to remove and 0 not upgraded.
> After this operation, 796 kB disk space will be freed.
> Do you want to continue? [Y/n] Y
> (Reading database ... 141634 files and directories currently installed.)
> Removing libboost-thread1.62.0:armhf (1.62.0+dfsg-10+b3) ...
> Removing libboost-system1.62.0:armhf (1.62.0+dfsg-10+b3) ...
> Removing libreoffice-gtk2 (1:6.1.5-3+rpi1) ...
> Processing triggers for libreoffice-common (1:6.1.5-3+rpi1) ...
> Processing triggers for libc-bin (2.28-10+rpi1) ...

Raspbian Buster:
Note: Instructions are same as Jessie unless mentioned otherwise
Raspbian Jessie:

✔sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

pi@Fidelity:~ $ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

✔sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116

pi@Fidelity:~ $ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
Executing: /tmp/apt-key-gpghome.2Nw2SMKeUO/gpg.1.sh --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
gpg: key 5523BAEEB01FA116: public key "ROS Builder <rosbuild@ros.org>" imported
gpg: Total number processed: 1
gpg: imported: 1

Now, make sure your Debian package index is up-to-date:
Skip this as it was just done yesterday:

~~sudo apt-get update~~
~~sudo apt-get upgrade~~

sudo apt-get install -y python-rosdep python-rosinstall-generator python-wstool python-rosinstall build-essential cmake

2.2 Install Bootstrap Dependencies
Raspbian Jessie:

✔sudo apt-get install -y python-rosdep python-rosinstall-generator python-wstool python-rosinstall build-essential cmake

pi@Fidelity:~ $ sudo apt autoremove
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  libboost-system1.62.0 libboost-thread1.62.0 libreoffice-gtk2
0 upgraded, 0 newly installed, 3 to remove and 0 not upgraded.
After this operation, 796 kB disk space will be freed.

Do you want to continue? [Y/n] Y....

2.3 Initializing rosdep

✔sudo rosdep init

pi@Fidelity:~ $ sudo rosdep init
Wrote /etc/ros/rosdep/sources.list.d/20-default.list
Recommended: please run

rosdep update

✔rosdep update

pi@Fidelity:~ $ rosdep update
reading in sources list data from /etc/ros/rosdep/sources.list.d
Hit
https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/osx-homebrew.ya
ml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/base.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/python.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/ruby.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/releases/fuerte.yaml
Query rosdistro index
https://raw.githubusercontent.com/ros/rosdistro/master/index-v4.yaml
Skip end-of-life distro "ardent"
Add distro "bouncy"
Add distro "crystal"
Add distro "dashing"
Add distro "eloquent"
Skip end-of-life distro "groovy"
Skip end-of-life distro "hydro"
Skip end-of-life distro "indigo"
Skip end-of-life distro "jade"
Add distro "kinetic"
Skip end-of-life distro "lunar"
Add distro "melodic"
updated cache in /home/pi/.ros/rosdep/sources.cache

3. Installation
Now, we will download and build ROS Kinetic.

3.1 Create a catkin Workspace
In order to build the core packages, you will need a catkin workspace. Create one now:

✔mkdir -p ~/ros_catkin_ws

✔cd ~/ros_catkin_ws

pi@Fidelity:~ $ cd ~/ros_catkin_ws
pi@Fidelity:~/ros_catkin_ws $

So far, only these two variants [ros_comm & desktop] have been tested on the Raspberry Pi in Kinetic; however, more are defined in REP 131 such as robot, perception, etc. Just change the package path to the one you want

- desktop:
  extends: [ros-full, robot, viz]
  stacks: [ros_tutorials, common_tutorials, geometry_tutorials, visualization_tutorials]
- desktop-full:
  extends: [desktop, mobile, perception, simulators]

Based on above, desktop-full seems to have all the desired features even tho it has NOT been "tested and approved". Therefore will [try to] install desktop-full

The Ubuntu directions say use apt-get:

✗sudo apt-get install ros-kinetic-desktop-full

which seems much easier. Let's try that first.

pi@Fidelity:~/ros_catkin_ws $ sudo apt-get install ros-kinetic-desktop-full
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package ros-kinetic-desktop-full

The Raspbian directions say use rosinstall_generator and wstool: [-full added]

✗rosinstall_generator desktop-full --rosdistro kinetic --deps --wet-only --tar > kinetic-desktop-wet.rosinstall

pi@Fidelity:~/ros_catkin_ws $ rosinstall_generator desktop**-full** --rosdistro kinetic --deps --wet-only --tar > kinetic-desktop-wet.rosinstall
The following not released packages/stacks will be ignored: desktop-full
No packages/stacks left after ignoring not released

Forgot to add -full on the second half.

pi@Fidelity:~/ros_catkin_ws $ rosinstall_generator desktop**-full** --rosdistro kinetic --deps --wet-only --tar > kinetic-desktop**-full**-wet.rosinstall
The following not released packages/stacks will be ignored: desktop-full
No packages/stacks left after ignoring not released

OK, let's try desktop plain...

✔rosinstall_generator desktop --rosdistro kinetic --deps --wet-only --tar > kinetic-desktop-wet.rosinstall

pi@Fidelity:~/ros_catkin_ws $ ls
kinetic-desktop-full-wet.rosinstall kinetic-desktop-wet.rosinstall

✔wstool init src kinetic-desktop-full-wet.rosinstall

pi@Fidelity:~/ros_catkin_ws $ wstool init src kinetic-desktop-wet.rosinstall
Using initial elements from: kinetic-desktop-wet.rosinstall
Writing /home/pi/ros_catkin_ws/src/.rosinstall...
...update complete.

–I'll go with that.

If wstool init fails or is interrupted, you can resume the download by running:

wstool update -j4 -t src

- didn't happen.

3.2 Resolve Dependencies
Before you can build your catkin workspace, you need to make sure that you have all the required dependencies. We use the rosdep tool for this, however, a couple of dependencies are not available in the repositories. They must be manually built first.

3.2.1 Unavailable Dependencies
Compilation of collada_urdf will fail per this issue.
[this may not be trus as David Williams did not see a collada_urdf failure. We will see]

No collada failure noted.

3.2.2 Resolving Dependencies with rosdep
The remaining dependencies should be resolved by running rosdep:
Raspbian Buster:

✔cd ~/ros_catkin_ws

already there

rosdep install -y --from-paths src --ignore-src --rosdistro kinetic -r --os=debian:buster

pi@Fidelity:~/ros_catkin_ws $ rosdep install -y --from-paths src --ignore-src --rosdistro kinetic -r --os=debian:buster
executing command [sudo -H apt-get install -y python-matplotlib]
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:..
[...VERY long build...]
[PuTTY crashed mid build
re-do above...]
pi@Fidelity:~/ros_catkin_ws $ **rosdep install -y --from-paths src --ignore-src --rosdistro kinetic -r --os=debian:buster**
executing command [sudo -H apt-get install -y libboost-all-dev]
E: Could not get lock /var/lib/dpkg/lock-frontend - open (11: Resource temporarily unavailable)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), is another process using it?
[Probably still going on in lost terminal. Will have to reboot]
✔cd ~/ros_catkin_ws
✔rosdep install -y --from-paths src --ignore-src --rosdistro kinetic -r --os=debian:buster
...
#All required rosdeps installed successfully

pi@Fidelity:~/ros_catkin_ws $ ls
kinetic-desktop-full-wet.rosinstall kinetic-desktop-wet.rosinstall src
[src is new]
pi@Fidelity:~/ros_catkin_ws/src $ ls
[lengthy list]

3.3 Building the catkin Workspace
Once you have completed downloading the packages and have resolved the dependencies, you are ready to build the catkin packages.
Invoke catkin_make_isolated:

It is highly likely that the compilation will fail with an "internal compiler error" caused by memory exhaustion. A quick fix for this is to add swap space to the Pi and recompile. If the error persists try building with the -j2 option instead of the default -j4 option [Let's just do -j2 from the start]:

sudo ./src/catkin/bin/catkin_make_isolated --install -DCMAKE_BUILD_TYPE=Release --install-space /opt/ros/kinetic -j2

pi@Fidelity:~/ros_catkin_ws $ sudo ./src/catkin/bin/catkin_make_isolated --install -DCMAKE_BUILD_TYPE=Release --install-space /opt/ros/kinetic -j2
Base path: /home/pi/ros_catkin_ws
Source space: /home/pi/ros_catkin_ws/src
Build space: /home/pi/ros_catkin_ws/build_isolated
Devel space: /home/pi/ros_catkin_ws/devel_isolated
Install space: /opt/ros/kinetic
Additional CMake Arguments: -DCMAKE_BUILD_TYPE=Release
Additional make Arguments: -j2
~~~~~~
~~ traversing 191 packages in topological order:

[extensive build...]
-- Tesseract: NO
-- xfeatures2d/boostdesc: Download: boostdesc_bgm.i
-- xfeatures2d/boostdesc: Download: boostdesc_bgm_bi.i
-- xfeatures2d/boostdesc: Download: boostdesc_bgm_hd.i
[PuTTY crashed, again, at this point]
[reboot]
✔✗pi@Fidelity:~/ros_catkin_ws $ **sudo ./src/catkin/bin/catkin_make_isolated --install -DCMAKE_BUILD_TYPE=Release --install-space /opt/ros/kinetic -j2**
[loooong build]
...
[several of this type of error:]
/home/pi/ros_catkin_ws/build_isolated/opencv3/install/modules/python_bindings_generator/pyopencv_generated_ns_reg.h:2515:38: warning: cast between incompatible function types from 'PyObject* (*)(PyObject*, PyObject*, PyObject*)' {aka '_object* (*)(_object*, _object*, _object*)'} to 'PyCFunction' {aka '_object* (*)(_object*, _object*)'} [-Wcast-function-type]

make[1]: *** Waiting for unfinished jobs....
[100%] Linking CXX shared module ../../lib/cv2.so
[100%] Built target opencv_python2
~~~~~~

make: *** [Makefile:163: all] Error 2
<== Failed to process package 'opencv3':
  Command '['/opt/ros/kinetic/env.sh', 'make', '-j2']' returned non-zero exit status 2

Reproduce this error by running:
==> cd /home/pi/ros_catkin_ws/build_isolated/opencv3 && /opt/ros/kinetic/env.sh make -j2

Command failed, exiting.

pi@Fidelity:~/ros_catkin_ws/build_isolated/opencv3 $ **cd /home/pi/ros_catkin_ws/build_isolated/opencv3 && /opt/ros/kinetic/env.sh make -j2**
make: *** No targets specified and no makefile found. Stop.

[but if I do this:]

pi@Fidelity:~/ros_catkin_ws/build_isolated/opencv3/install $ **make**
[100%] Built target gen-pkgconfig
...
I get similar errors as predicted

CMake may have trouble finding FindEigen3.cmake. See this discussion for workarounds. [no such problem reported]

On Raspbian Buster the compilation may fail with "'boost/tr1/unordered_set.hpp' file not found". This is because rospack version used in Kinetic is dependent on boost 1.58. To fix this error try installing boost 1.58 manually. [But it doesn't say how. Cross bridge if..] [no such problem reported]

...Or optionally source the setup.bash in the ~/.bashrc, so that ROS environment variables are automatically added to your bash session every time a new shell is launched:

✔echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc

At this point ROS should be installed. Next stop is to install OSR code.

2019.07.08
Download & Install OSR code

Taking the directions here:

https://github.com/nasa-jpl/open-source-rover/blob/ros-rework/Software/Software%20Steps.pdf

and this Github issue thread:

https://github.com/nasa-jpl/open-source-rover/issues/133

the plan is:

✔pi@Fidelity:~ $ **ls -l /dev/serial***
lrwxrwxrwx 1 root root 5 Jul 7 20:07 /dev/serial0 -> ttyS0
lrwxrwxrwx 1 root root 7 Jul 7 20:07 /dev/serial1 -> ttyAMA0

✔**sudo nano /boot/cmdline.txt**
"console = ..." to read "console=tty1"

1.4 Downloading the Rover Code

Move to your home directory:
✔pi@Fidelity:~ $ **cd /home/pi** [already there]
Clone the rover code repository:
✔pi@Fidelity:~ $ **git clone https://github.com/nasa-jpl/osr-rover-code osr**
Cloning into 'osr'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 216 (delta 2), reused 2 (delta 0), pack-reused 206
Receiving objects: 100% (216/216), 216.84 KiB | 1.12 MiB/s, done.
Resolving deltas: 100% (69/69), done.
Move into the directory you just cloned:
✔pi@Fidelity:~ $ **cd osr**
pi@Fidelity:~/osr $
Check out the osr-ROS branch:
✔pi@Fidelity:~/osr $ **git checkout osr-ROS**
Branch 'osr-ROS' set up to track remote branch 'osr-ROS' from 'origin'.
Switched to a new branch 'osr-ROS'
Pull to make sure you have the latest code:
✔pi@Fidelity:~/osr $ **git pull**
Already up to date.

1.5 Building the Rover Code to work with your ROS installation

pi@Fidelity:~/osr $ **ls**
Arduino img LICENSE.txt README.md ROS

Move to the directory containing the main OSR logic:
✔pi@Fidelity:~/osr $ **cd /home/pi/osr/ROS**
pi@Fidelity:~/osr/ROS $ **ls**
led_screen osr osr_bringup osr_msgs

Create a catkin workspace directory:
mkdir -p /home/pi/catkin ws/src
NO! There is already a catkin_ws:
pi@Fidelity:~/osr/ROS $ **cd**
pi@Fidelity:~ $ **ls**
Desktop Documents Downloads MagPi Music osr Pictures Public
ros_catkin_ws Templates ttyS0 Videos
Copy the source files to your catkin workspace:
mv led_screen/ osr/ osr_bringup/ osr_msgs/ /home/pi/catkin_ws/src
change this to
mv led_screen/ osr/ osr_bringup/ osr_msgs/ /home/pi/**ros_catkin_ws**/src

Move (back) to the directory containing the main OSR logic:
pi@Fidelity:~ $ **cd /home/pi/osr/ROS**
pi@Fidelity:~/osr/ROS $ **ls**
led_screen osr osr_bringup osr_msgs

pi@Fidelity:~/osr/ROS $ **mv led_screen/ osr/ osr_bringup/ osr_msgs/ /home/pi/ros_catkin_ws/src**
Now let's see what all we moved there:
pi@Fidelity:~/osr/ROS $ **cd /home/pi/ros_catkin_ws/src**
pi@Fidelity:~/ros_catkin_ws/src $ **ls -1**
actionlib
angles
bond_core
catkin
class_loader
cmake_modules
collada_urdf
common_msgs
common_tutorials
control_msgs
diagnostics
dynamic_reconfigure
eigen_stl_containers
executive_smach

filters
gencpp
geneus
genlisp
genmsg
gennodejs
genpy
geometric_shapes
geometry
geometry2
geometry_tutorials
gl_dependency
image_common
interactive_markers
joint_state_publisher
kdl_parser
laser_geometry
led_screen
media_export
message_generation
message_runtime
metapackages
navigation_msgs
nodelet_core
octomap
opencv3
orocos_kinematics_dynamics
osr
osr_bringup
osr_msgs
pluginlib
python_qt_binding
qt_gui_core
qwt_dependency
random_numbers
resource_retriever
robot_model
robot_state_publisher
ros
rosbag_migration_rule
ros_comm
ros_comm_msgs
rosconsole_bridge
roscpp_core
ros_environment
roslint

roslisp
rospack
ros_tutorials
rqt
rqt_action
rqt_bag
rqt_common_plugins
rqt_console
rqt_dep
rqt_graph
rqt_image_view
rqt_launch
rqt_logger_level
rqt_moveit
rqt_msg
rqt_nav_view
rqt_plot
rqt_pose_view
rqt_publisher
rqt_py_console
rqt_reconfigure
rqt_robot_dashboard
rqt_robot_monitor
rqt_robot_plugins
rqt_robot_steering
rqt_runtime_monitor
rqt_rviz
rqt_service_caller
rqt_shell
rqt_srv
rqt_tf_tree
rqt_top
rqt_topic
rqt_web
rviz
std_msgs
urdf
vision_opencv
visualization_tutorials
webkit_dependency
xacro

Move to your catkin workspace:

cd /home/pi/catkin_ws

change to

cd /home/pi/ros_catkin_ws

pi@Fidelity:~/ros_catkin_ws/src $ **cd /home/pi/ros_catkin_ws**
pi@Fidelity:~/ros_catkin_ws $ **ls**
build_isolated devel_isolated kinetic-desktop-full-wet.rosinstall
kinetic-desktop-wet.rosinstall src

Run a "catkin make" of the OSR workspace:

catkin_make

pi@Fidelity:~/ros_catkin_ws $ **catkin_make**
Base path: /home/pi/ros_catkin_ws
Source space: /home/pi/ros_catkin_ws/src
Build space: /home/pi/ros_catkin_ws/build
Devel space: /home/pi/ros_catkin_ws/devel
Install space: /home/pi/ros_catkin_ws/install
Creating symlink "/home/pi/ros_catkin_ws/src/CMakeLists.txt" pointing to "catkin/cmake/toplevel.cmake"
...long list of build commands...
CMake Error at catkin/cmake/catkin_workspace.cmake:95 (message):
This workspace contains non-catkin packages in it, and catkin cannot build a non-homogeneous workspace without isolation. **Try the 'catkin_make_isolated' command instead.**
Call Stack (most recent call first):
CMakeLists.txt:67 (catkin_workspace)

-- Configuring incomplete, errors occurred!
See also "/home/pi/ros_catkin_ws/build/CMakeFiles/CMakeOutput.log".
See also "/home/pi/ros_catkin_ws/build/CMakeFiles/CMakeError.log".
Invoking "cmake" failed
pi@Fidelity:~/ros_catkin_ws $ pi@Fidelity:~/ros_catkin_ws $ catkin_make
bash: pi@Fidelity:~/ros_catkin_ws: No such file or directory
pi@Fidelity:~/ros_catkin_ws $ Base path: /home/pi/ros_catkin_ws
bash: Base: command not found
pi@Fidelity:~/ros_catkin_ws $ Source space: /home/pi/ros_catkin_ws/src
bash: Source: command not found
pi@Fidelity:~/ros_catkin_ws $ Build space: /home/pi/ros_catkin_ws/build
bash: Build: command not found
pi@Fidelity:~/ros_catkin_ws $ Devel space: /home/pi/ros_catkin_ws/devel
bash: Devel: command not found
pi@Fidelity:~/ros_catkin_ws $ Install space: /home/pi/ros_catkin_ws/install
bash: Install: command not found

Let’s try the above advice:

```
pi@Fidelity:~/ros_catkin_ws $ catkin_make_isolated
Base path: /home/pi/ros_catkin_ws
Source space: /home/pi/ros_catkin_ws/src
Build space: /home/pi/ros_catkin_ws/build_isolated
Traceback (most recent call last):
  File "/opt/ros/kinetic/bin/catkin_make_isolated", line 162, in <module>
    main()
  File "/opt/ros/kinetic/bin/catkin_make_isolated", line 158, in main
    override_build_tool_check=opts.override_build_tool_check,
  File "/opt/ros/kinetic/lib/python2.7/dist-packages/catkin/builder.py", line 897, in
build_workspace_isolated
    mark_space_as_built_by(buildspace, 'catkin_make_isolated')
  File "/usr/lib/python2.7/dist-packages/catkin_pkg/tool_detection.py", line 78, in
mark_space_as_built_by
    with open(marker_path, 'w') as f:
IOError: [Errno 13] Permission denied:
'/home/pi/ros_catkin_ws/build_isolated/.built_by'
```

Permission denied usually means run with sudo ...

```
pi@Fidelity:~/ros_catkin_ws $ sudo catkin_make_isolated
sudo: catkin_make_isolated: command not found
```

2019.07.13
Added the HCC logo to the rover back plate. Messed up reversing the logo so had to re-cut/etch the whole thing. Lots of visiting w/ newcomers. No OSR SD card image upload yet to started toward the end of the day a new install plan trying to build on past successes & avoid prior failures while still not really knowing what I'm doing. See OSR Software Install Plan.pdf this date.

2019.07.16
Tried the New, new Software Steps from JPL but it crashed at this step:

3.3 Building the catkin Workspace
Once you have completed downloading the packages and have resolved the dependencies, you are ready to build the catkin packages.
Invoke catkin_make_isolated:

```
        pi@Fidelity:~/ros_catkin_ws $ sudo ./src/catkin/bin/catkin_make_isolated --install
-DCMAKE_BUILD_TYPE=Release --install-space /opt/ros/kinetic
        Traceback (most recent call last):
  File "./src/catkin/bin/catkin_make_isolated", line 12, in <module>
    from catkin.builder import build_workspace_isolated
  File "./src/catkin/bin/../python/catkin/builder.py", line 66, in <module>
    from catkin_pkg.terminal_color import ansi, disable_ANSI_colors, fmt, sanitize
ImportError: No module named terminal_color
```

Posted error msg to Github.

2019.07.17
Will try this time with Raspbian Buster per JPL notes:
        raspbian-buster-installation-notes.docx
See "OSR Software Install Plan 2019.07.17.wpd/.pdf"

See OSR Software Install Plan.wpd

RETURN from OSR Software Install Plan.wpn

2019.08.19 Stafford
"First steps" of Fidelity. Caught on video by HCC video.
Search YouTube for "HCC beat usai":
https://www.youtube.com/watch?v=2Wdgss0q63s

2019.08.24 West Loop
"Blessing of the Robot"

2019.10.26 West Loop
Presentation of:

C:\Users\Me\Documents\My Downloads\Robotics\USAi Labs\
Mars Rover\Documentation**Fidelity to Audacity.pptx**

C:\Users\Me\Documents\My Downloads\Robotics\USAi Labs\
Robot Safety Module**Robot Safety Module.pptx**

2019.11.16 HCC West Loop
Extension of work last time on making the ROS boot on startup. A fellow member, Antonio, discovered an error in JPLs code in the "Software Steps" Init Script section & fixed it. See comments below. I now get an error if I try to enable launch on power-on.

Issue posted to Github 2019.11.17:

Software Steps 2.7 Init Scipts.
This section says to:
cd /home/pi/osr/Initn Scripts
sudo cp LaunchOSR.sh /usr/bin/LaunchOSR.sh
sudo chmod +x /usr/bin/LaunchOSR.sh
sudo cp osr startup.service /etc/systemd/system/osr_startup.service
sudo chmod 644 /etc/systemd/system/osr_startup.service

/usr/bin/LaunchOSR.sh
#!/bin/bash
bash -c ". /home/pi/osr_ws/devel/setup.sh"
bash -c ". /home/pi/osr_ws/devel/setup.bash"
bash -c ". /opt/ros/kinetic/setup.sh"
bash -c ". /opt/ros/kinetic/setup.bash"
bash -i -c "roslaunch osr_bringup osr.launch"

osr_startup.service example from OSR Github:
[Unit]
Description=OSR service
After=network.target

[Service]

User=ubuntu
Group=ubuntu
WorkingDirectory=/home/ubuntu/LaunchOSR.sh
ExecStart=/home/ubuntu/
ExecReload=/bin/kill -HUP $MAINPID
Restart=always

RestartSec=3

[Install]
Wantedby=multi-user.target

Translating this from Ubuntu to Raspbian:
/etc/systemd/system/osr_startup.service
[Unit]
Description=OSR service
After=network.target
[Service]
User=pi
Group=pi
#WorkingDirectory=/usr/pi/LaunchOSR.sh
#This line is wrong. It doesn't point to a directory but to a file. It should be:
WorkingDirectory=/home/pi/
#Same for below. The filename was missing:
ExecStart=/usr/bin/LaunchOSR.sh
ExecReload=/bin/kill -HUP $MAINPID
Restart=always
RestartSec=3
[Install]
Wantedby=multi-user.target

This configuration works fine (after correcting the error above thanks to Antonio from our group)
for start, stop, and status of osr_startup.service
BUT if I try
sudo systemctl enable osr startup.service
in order for the system to launch on power-on
I get the following error:
pi@Fidelity:~ $ sudo systemctl enable osr_startup.service
The unit files have no installation config (WantedBy, RequiredBy, Also, Alias
settings in the [Install] section, and DefaultInstance for template units).
This means they are not meant to be enabled using systemctl.
Possible reasons for having this kind of units are:

A unit may be statically enabled by being symlinked from another unit's
.wants/ or .requires/ directory.

A unit's purpose may be to act as a helper for some other unit which has
a requirement dependency on it.
A unit may be started when needed via activation (socket, path, timer,
D-Bus, udev, scripted systemctl call, ...).
In case of template units, the unit is meant to be enabled with some
instance name specified.
the WantedBy file referenced does exist in the directory.

The Linode.com link provides only a rudimentary example or a useless encyclopedia of documentation.
I don't see any mention of this problem here or in the TapaTalk.
Perhaps no one else has tried launch on power-up?

Jim Phelan
USAi Labs

2019.11.19
Antonio Hernandez <antonio@mlmusings.com>
Hi Jim,

The only issue I see, and it makes sense that it does not want to be enabled is the last line of the systemd file,

"Wanted**b**y=multi-user.target" should be "Wanted**B**y=multi-user.target"

Basically WantedBy tells Linux if you load the multi-user.target also load this script/service.

Let me know if that helps.

Antonio

pi@AUDACITY:/etc/systemd/system $ **sudo nano osr_startup.service**
        capitalized the B in WantedBy
pi@AUDACITY:/etc/systemd/system $ **sudo systemctl enable osr_startup.service**
Created symlink /etc/systemd/system/multi-user.target.wants/osr_startup.service → /etc/systemd/system/osr_startup.service.
pi@AUDACITY:/etc/systemd/system $ **sudo systemctl status osr_startup.service**
● osr_startup.service - OSR service
  Loaded: loaded (/etc/systemd/system/osr_startup.service; enabled; vendor pres
  Active: inactive (dead)

pi@AUDACITY:/etc/systemd/system $ **sudo systemctl start osr_startup.service**
pi@AUDACITY:/etc/systemd/system $ **sudo systemctl status osr_startup.service**
● osr_startup.service - OSR service
  Loaded: loaded (/etc/systemd/system/osr_startup.service; enabled; vendor pres

```
   Active: active (running) since Tue 2019-11-19 19:27:12 CST; 3s ago
 Main PID: 1238 (LaunchOSR.sh)
    Tasks: 2 (limit: 4915)
```

pi@AUDACITY:~ $ **sudo systemctl status osr_startup.service**

```
● osr_startup.service - OSR service
   Loaded: loaded (/etc/systemd/system/osr_startup.service; enabled; vendor preset:
enabled)
   Active: active (running) since Tue 2019-11-19 19:33:25 CST; 4min 3s ago
 Main PID: 475 (LaunchOSR.sh)
    Tasks: 47 (limit: 4915)
   CGroup: /system.slice/osr_startup.service
           ├─ 475 /bin/bash /usr/bin/LaunchOSR.sh
           ├─ 605 python /home/pi/osr_ws/devel/bin/roslaunch osr_bringup osr.launch
           ├─1063 python /home/pi/osr_ws/devel/bin/rosmaster --core -p 11311 -w 3
__log:=/home/pi/.ros/log/c53bb8c4-0b35-11ea-8f3c-b827eb2b674d/ma
           ├─1076 /home/pi/osr_ws/devel/lib/rosout/rosout __name:=rosout
__log:=/home/pi/.ros/log/c53bb8c4-0b35-11ea-8f3c-b827eb2b674d/rosout-1.lo
           ├─1079 python /home/pi/osr_ws/src/osr/scripts/motor_controller.py
__name:=motor_controller __log:=/home/pi/.ros/log/c53bb8c4-0b35-11ea-
           ├─1080 python /home/pi/osr_ws/src/osr/scripts/joystick.py __name:=joystick
__log:=/home/pi/.ros/log/c53bb8c4-0b35-11ea-8f3c-b827eb2b674
           ├─1081 python /home/pi/osr_ws/src/osr/scripts/rover.py __name:=rover
__log:=/home/pi/.ros/log/c53bb8c4-0b35-11ea-8f3c-b827eb2b674d/rove
           ├─1082 python /home/pi/osr_ws/src/led_screen/scripts/arduino_comm.py
__name:=led_screen __log:=/home/pi/.ros/log/c53bb8c4-0b35-11ea-8f3
           └─1083 /opt/ros/kinetic/lib/joy/joy_node __name:=joy_node
__log:=/home/pi/.ros/log/c53bb8c4-0b35-11ea-8f3c-b827eb2b674d/joy_node-6.log

Nov 19 19:33:25 AUDACITY systemd[1]: Started OSR service.
Nov 19 19:33:28 AUDACITY LaunchOSR.sh[475]: bash: cannot set terminal process
group (475): Inappropriate ioctl for device
Nov 19 19:33:28 AUDACITY LaunchOSR.sh[475]: bash: no job control in this shell
Nov 19 19:33:45 AUDACITY LaunchOSR.sh[475]: [ERROR] [1574213625.142923949]:
Couldn't open joystick /dev/input/js0. Will retry every second.
```

Joystick WORKS to drive & turn. Force-feedback error is expected.

2019.12.02 Home

TapaTalk new forum post:
https://www.tapatalk.com/groups/usailabs/viewtopic.php?f=2&t=3

This topic is to address improvements in the NASA/JPL Open Source Rover. Specifically the drive train. A separate topic will address the steering motors. Other topics for display, sensors, etc.... Sometimes it's tough to stick to topic in a forum, but let's do our best so the info we need to share is in one place ;-)
In a follow-up post I'll list what we have now, some of the problems, some alternatives other OSR groups have used and suggestions from our group comparing specifications.

I'd like a discussion of---
torque: how much torque is needed to climb stairs and mountains? Can we agree on a standard unit e.g N/m vs lbs/ft, oz/in, kg/m? All other considerations follow from this.
types of motors: brushed, brushless, hub, stepper, etc. and their advantages, disadvantages, costs, applicability to our project.
gears: gear trains, planetary gears, worm gears, etc. with same considerations.
drive train support: current wheel attached directly to motor hub. Puts undue lateral stress on motor.
coupling: hub clamps are prone to loosening and slippage. The hex shaft of the REV Robotics motors is appealing, but not sure about the rest of it.
wheels: how large and what kind of wheels to we need to climb stairs and mountains?
Would really appreciate everybody's thoughts on this as we all have different areas of expertise and perspectives. My ignorance in this area is immense!
JHP

James Phelan (Co-Organizer) sent a message to the Artificial Intelligence and Machine Learning for Robotics mailing list
Fidelity to AUDACITY Open Source Rover upgrade forum

Having finished "Fidelity" our NASA/JPL Open Source Mars Rover this Fall it's time to put the USAi Labs (Houston Robotics + AI/ML for Robotics & Iot) magic on it and upgrade it to "AUDACITY" a fully autonomous and improved rover. We need EVERYBODY's expertise on this starting with the drive train, but soon to expand to other areas. You'll need to join TapaTalk to participate (it's free):

https://www.tapatalk.com/groups/usailabs/viewtopic.php?f=2&t=3

Here's the original OSR Github page:

https://github.com/nasa-jpl/open-source-rover

And their TapaTalk forum:

https://www.tapatalk.com/groups/jpl_opensource_rover/index.php

Here's a video from HCC about USAi labs including the "first steps" of Fidelity.

https://www.youtube.com/watch?v=2Wdgss0q63s

If you're a member but haven't been to a meeting in a while (or ever) it's time to come check out the exciting variety of projects, people, and talents this group has to offer for all ages!

Jim Phelan