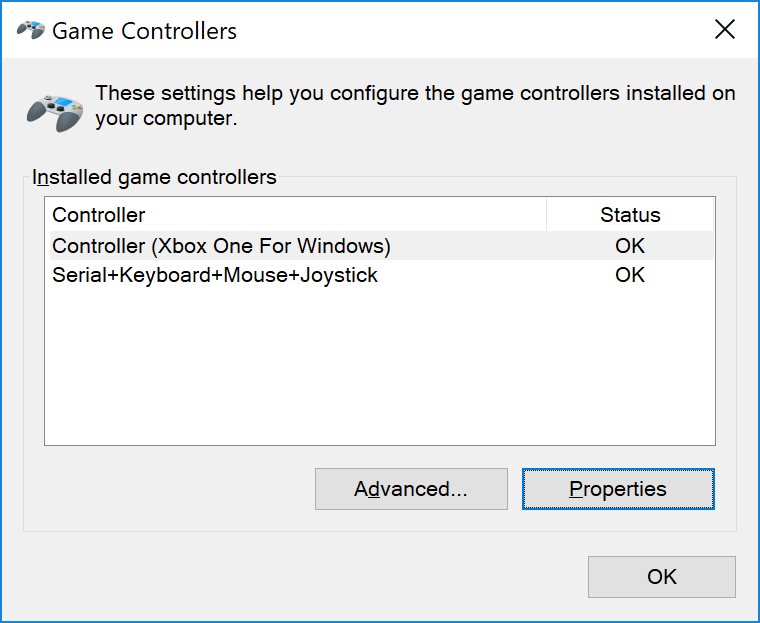
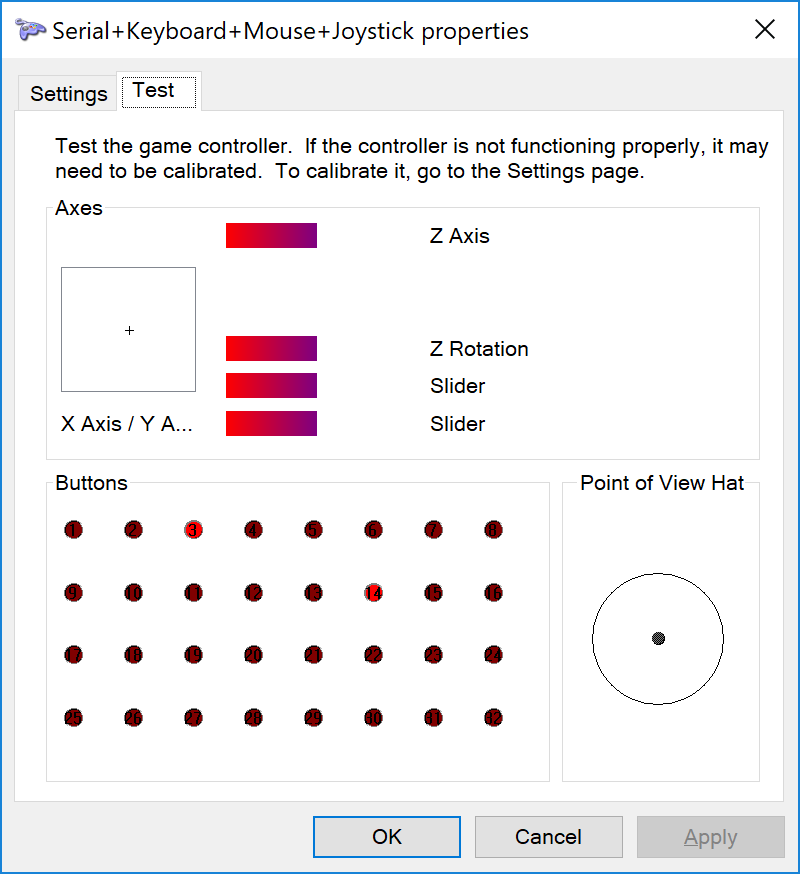
Design of the DeepSpaceButtonBox

* Code for the Teensy++ 2.0 USB microcontroller. This will look like a joystick to the Windows 10 PC, with will then be compatible with the Driver Station software. The RoboRIO robot code software will need to interpret the XboxController object’s “buttons” that it can poll.
* You must install the Arduino Sketch IDE with the Windows Installer, does not work with the Windows App version. Then install the Teensyduino software from PJRC. (<https://www.pjrc.com/teensy/teensyduino.html>)
  + It is annoying that the first time you compile (verify in Arduino terms), the Teensyduino library has several warnings in it.
* There are L,M,R and L,C,R to indicate the azimuth angle of attack the robot needs, to support semi-autonomous robot approach. We eliminated that kind of redundancy for the cargo ship.
* the data structure needed on the RoboRIO to apply these button signals is as follows: a matrix of cells indexed by [BaySelected, Hatch/CargoSelected]. Each cell is a record { ElevatorHeightInEncoderCounts, ElevatorHeightInLidarMeasure, RequiredRobotAzimuthAngleOfAttack } and any other targets to be commanded, then close loop measured until achieved.
* <https://www.pjrc.com/teensy/pinout.html>

A screenshot of a cell phone

Description automatically generated





Note that joystick button zero seems to be working, even though this property sheet doesn’t show it (DIO D0 is not on this dialog, but DIO D1 can be shown to toggle the 1 red circle.) Will need to check whether the RoboRIO does indeed see DIO D0, and if not, shift it to a higher pin, such as pin 41 (F3)

**Collected Serial Monitor output:**

Setup Complete

Setup Complete

Elevator Button Pressed: Arcade DIO#0, Joystick Button #0.

Elevator Button Pressed: Arcade DIO#1, Joystick Button #1.

Elevator Button Pressed: Arcade DIO#2, Joystick Button #2.

Elevator Button Pressed: Arcade DIO#3, Joystick Button #3.

Elevator Button Pressed: Arcade DIO#4, Joystick Button #4.

Elevator Button Pressed: Arcade DIO#5, Joystick Button #5.

Elevator Button Pressed: Arcade DIO#7, Joystick Button #7.

Elevator Button Pressed: Arcade DIO#8, Joystick Button #8.

Elevator Button Pressed: Arcade DIO#9, Joystick Button #9.

Elevator Button Pressed: Arcade DIO#10, Joystick Button #10.

Elevator Button Pressed: Arcade DIO#11, Joystick Button #11.

Elevator Button Pressed: Arcade DIO#12, Joystick Button #12.

Elevator Button Pressed: Arcade DIO#13, Joystick Button #13.

Cargo/Hatch Button Pressed, commanded: 1

Elevator Button Pressed: Arcade DIO#41, Joystick Button #6.

Elevator Button Pressed: Arcade DIO#4, Joystick Button #4.

Cargo/Hatch Button Pressed, commanded: 0

Cargo/Hatch Button Pressed, commanded: 1

Cargo/Hatch Button Pressed, commanded: 0

Cargo/Hatch Button Pressed, commanded: 1

Elevator Button Pressed: Arcade DIO#2, Joystick Button #2.

Cargo/Hatch Button Pressed, commanded: 0

Cargo/Hatch Button Pressed, commanded: 1

Cargo/Hatch Button Pressed, commanded: 0

Elevator Button Pressed: Arcade DIO#3, Joystick Button #3.

Elevator Button Pressed: Arcade DIO#2, Joystick Button #2.

Open issues:

* However, we had a question if we're allowed unlimited inputs which we should find out. I.e. whether or not we can read more I/Os than on a standard Xbox controller. If it's the former, the good is good as-is. If it's the latter, it's still not too hard to get around -- we'd have a few options as to how to solve things there. One option could be to encode all of our buttons as binary options. Directly, this makes an assumption button presses are mutually exclusive, but you can slightly modify the design to get around this. Another would be to take advantage of the analog options we have and discretize or even encode information in them.
  + My xbox one controller on a windows 10 pc: opening the Game Controller properties it shows (and I can demonstrate) 10 buttons, 8 PoV HAT "buttons". So far we need 14 buttons, so it looks like we have enough to do a 1:1 signalling by splitting across the two types. Still, it would be easier if we just communicate them all uniformly as buttons, so please do check whether we can just drive 14+ buttons.
* I do not see a way for the RoboRIO to communicate back to our USB device. I was hoping a rumble feature would be supported and we could use that for the RoboRIO to send feedback to what its state tracking of cargo/hatch setting and bay selection was, possibly even more info. The Teensy can also look like serial port, keyboard, etc. to the PC, but not sure how the RoboRIO can see them.