LAB: Arduino Gamecontroller

Apparatus

Computer
Anaconda
Jupyter Notebook
VPython (for Jupyter)
Arduino IDE programming editor and compiler
Arduino Uno microprocessor
breadboard
2-axis potentiometer (joystick)
wiring kit
LED
pushbutton SPST switch

Goal

The purpose of this activity is to build a game controller and to use the game controller to operate the thruster in the Lunar Lander game. You will install a number of software packages to make this possible. The project requires: (1) biulding

Procedure

There are four steps:

1. Install software
2. Build the game controller
3. Upload an Arduino program to the Arduino Uno
4. Run Lunar Lander with the Arduino game controller

Install Software

1. Go to our course web site where you will find links to download and install software.
2. Install the Arduino IDE for writing and compiling Arduino programs and uploading to an Arduino board.
3. Install the Anaconda distribution of the Python 2.7 programming language and scientific packages. **IMPORTANT – DOWNLOAD THE INSTALLER FOR PYTHON 2.7.** (not Python 3.5)
4. After you install Anaconda, open a terminal window (also called the command line). On a Mac, you will do this by opening Applications→Utilities→Terminal. It should look similar to the following terminal window.
5. To verify that Anaconda is installed property, at the command line, type the following:

   ```
   which conda
   ```

   The command `which` returns the filepath to the location of the conda program.

Figure 1: Terminal window.

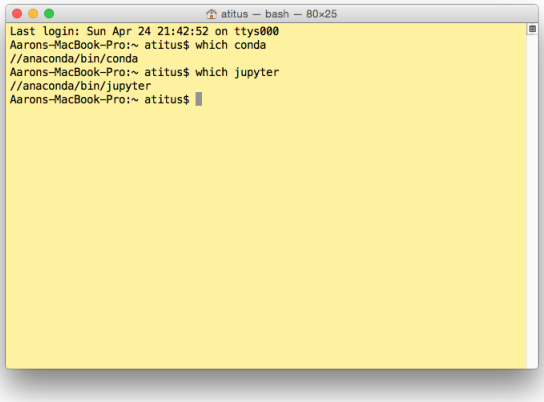


Figure 2: Terminal window showing path to conda and path to jupyter.

6. Now, type `which jupyter` at the command line. It should return the path to the jupyter program as shown in Figure 25.2.

   If this command does not return a path to jupyter, then you need to install Jupyter. In this case, type the command:

   ```
   conda install jupyter
   ```

7. Install the `vpython` package by typing:

   ```
   pip install vpython
   ```

8. Install the `pyserial` package by typing:

   ```
   pip install pyserial
   ```

   To test your software installation, we will open a Jupyter Notebook, import vpython and pyserial packages, and create a 3D object. If it is successful and produces no error messages, then we are confident that we will be able to develop programs that use our game controller.

9. At the command line, type

   ```
   jupyter notebook
   ```

   A Jupyter window will open in your web browser showing your files and folders in your home directory (or whatever directory you are in when you launch jupyter notebook), as shown in Figure 25.3.

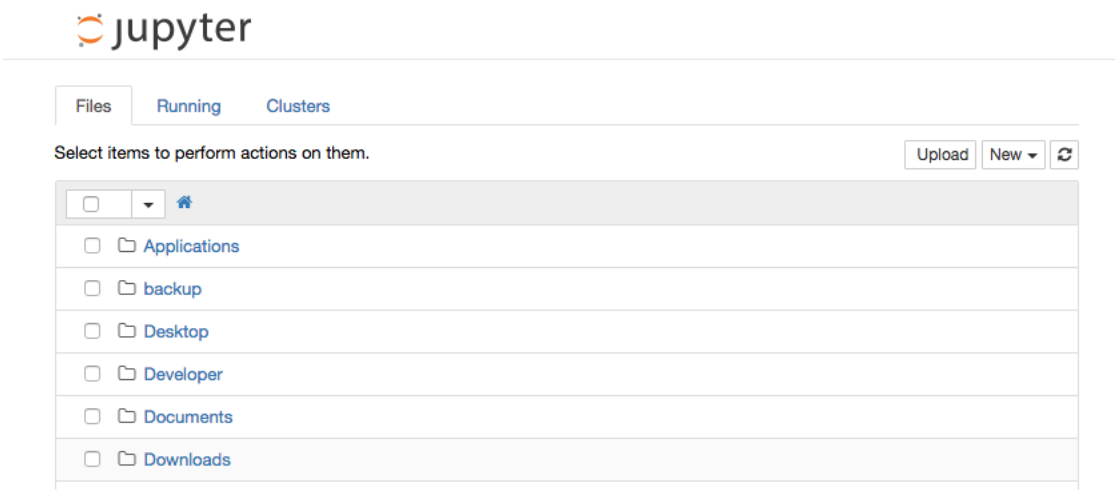


Figure 3: A Jupyter window.

10. You can clink the folder links to navigate to the folder of your choice. Then, create a new notebook by clicking the New button in the Jupyter toolbar. In the menu, select the `VPython` notebook, as shown in Figure 25.4.

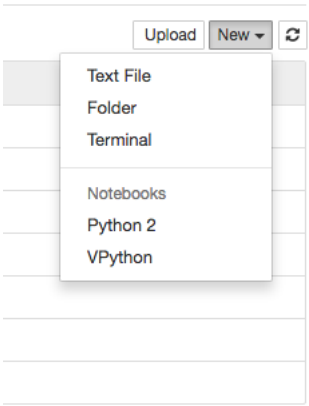


Figure 4: Creating a new notebook

   This creates a new notebook file as shown in Figure 25.5.

11. Click the name "Untitled" and change the name to something more appropriate like "notebook-test" or something like that. I suggest that you do not use blanks or characters other than a hypen or underscore in filenames.

12. In the first cell, type

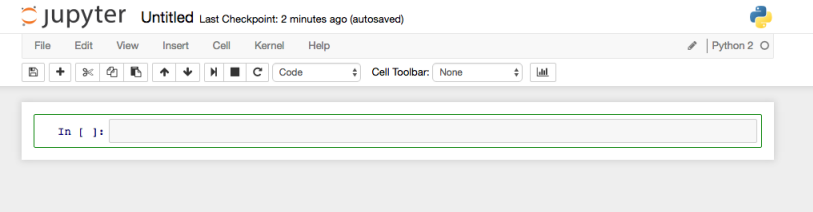


Figure 5: A newly created Jupyter notebook

```
from vpython import *
```

Use shift-RETURN to run the code in the cell.

After running, the cell will receive the number 1 and will be labeled `In [1]:` . The numbering system shows you the sequence that cells were run.

13. In the second cell, type

```
from serial import *
```

Again, use shift-RETURN to run the cell. (Do this for every cell in order to run it.) At this point, there should be no error messages.

14. Now, we have to create a canvas (i.e. scene) and a 3D object. In the third cell, type and run the following code:

```
scene=canvas(title="3D scene")
sphere()
```

You should see a sphere like the example in Figure 25.6.

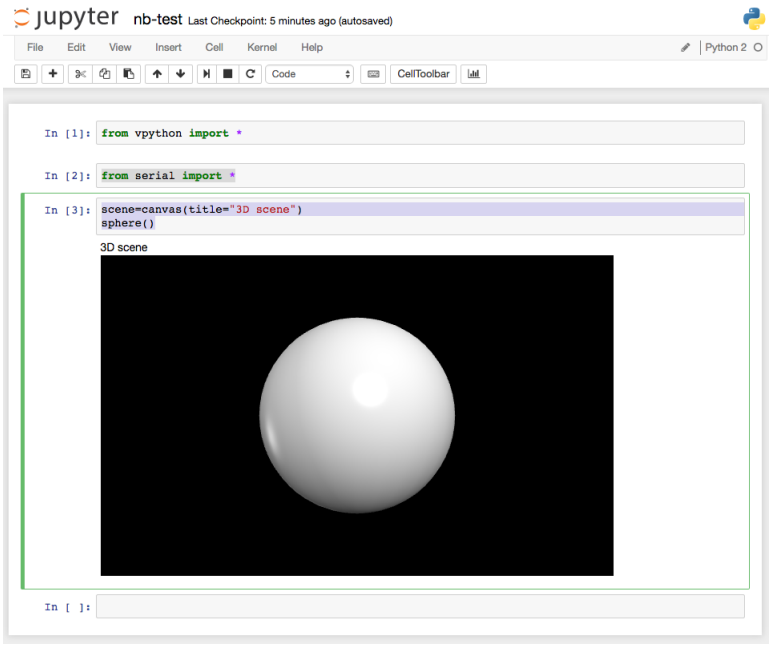


Figure 6: A VPython program running in a Jupyter notebook.

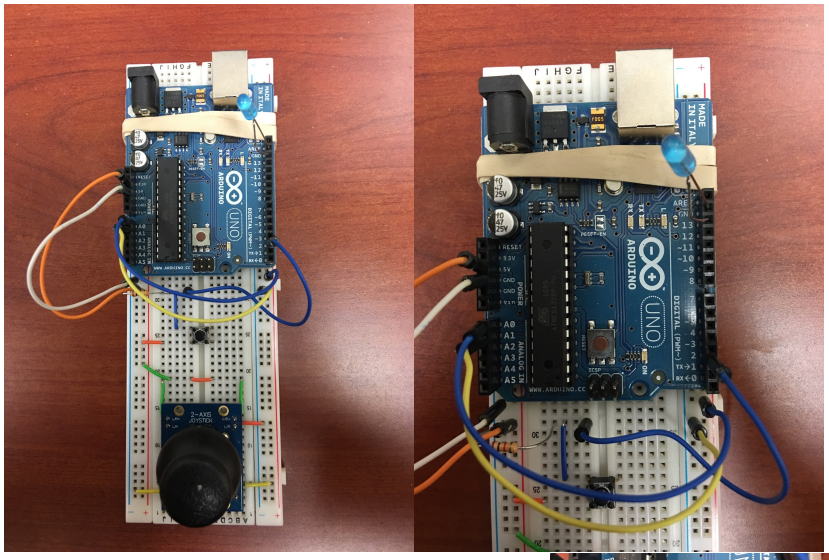
If you receive any errors up to this point, please get help. This program must work before you continue.

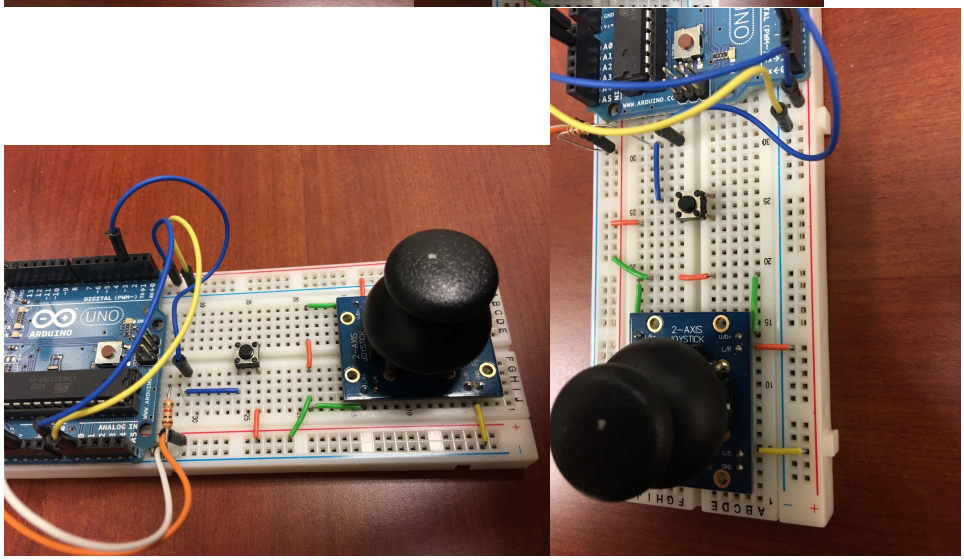
Build the game controller

Here are the parts:

(a) an Arduino Uno microprocessor
(b) a breadboard
(c) a LED
(d) a push button SPST switch
(e) a 1000 Ω resistor
(f) a Parallax 2-axis joystick
(g) a wiring kit
(h) a USB cable
(i) a rubber band

Here are photos that show the wiring for the game controller.





15. Attach the Arduino to the breadboard with the rubber band.

16. Place the LED in pins 13 and GND. The LED has a long lead and a short lead. The long lead goes in pin 13 and the short lead goes in GND.

17. Run wires from GND (white wire) and +5 V (orange wire) pins on the left side of the Arduino to the – and + columns on the left side of the breadboard, respectively. GND is wired to the – column and 5 V is wired to the + column.

18. Run wires from pins A0 and A1 on the left side of the Arduino to the + and – columns on the right side of the breadboard, respectively. A0 is wired to the +column an dA1 is wired to the – column.

19. Attach the joystick to the breadboard.

20. Wire the L/R+ and U/D+ pins on the joystick to +5 V (the + column on the left side of the breadboard).

21. Wire the GND pin on the joystick to GND (the – column on the left side of the breadboard).

22. Wire the U/D pin on the joystick to pin A0 of the Arduino (the + column on the right side of the breadboard).

23. Wire the L/R pin on the joystick to pin A1 of the Arduino (the – column on the right side of the breadboard).

24. The pushbutton switch, resistor, and the wire from pin 3 of the Arduino are not needed for the Lunar Lander game. You can add those parts later if you wish to fire bullets in Asteroids, for example.

Upload an Arduino program to the Arduino Uno

We must upload a program to the Arduino that reads the voltage across each axis of the joystick and passed it to VPython when requested.

25. Download lunarLander.zip from our course web site. Here is the code.

```
#define UD A0
#define LR A1
int received;
char buffer[10];                    // input buffer
int N;                              // how many measurements to make
boolean done = false;


void setup() {
  Serial.begin(9600);
}

void loop() {
        received = 0;
        buffer[received] = '\0';
        done = false;

        // Check input on serial line.
        while (!done) {
                if (Serial.available()) {       // Something is in the
                    buffer
                        N = Serial.read();      // so get the number byte
                        done = true;
                }
```

```
24          }
25
26          int LRval = analogRead(LR);
27          int UDval = analogRead(UD);
28          Serial.print(LRval, DEC);
29          Serial.print('\t');
30          Serial.print(UDval, DEC);
31          Serial.print('\n');
32          delay(10);
33      }
```

26. Open `lunarLander.ino` with the Arduino software. The program looks like Figure 25.7.

Figure 7: An Arduino program running on the microprocessor.

27. Go to the **Tools** menu and select **Port** . One of these ports corresponds to the Arduino. Make sure the correct one is checked, as shown in Figure 25.8. This usually occurs by default.

    There are two important buttons in the top left corner of the menu bar. One is a checkmark and one is a right arrow, as shown in Figure 25.9.

28. The checkmark button is used to compile the Arduino program. It will tell you if there are any programming errors. Click the checkmark button to compile.

29. If there are no errors, then you are ready to upload the program to the microprocessor. Click the right arrow button to upload the program to the mircoprocessor.

    The program should now be running on the microprocessor. It runs continuously in an infinite loop as long as it has power.

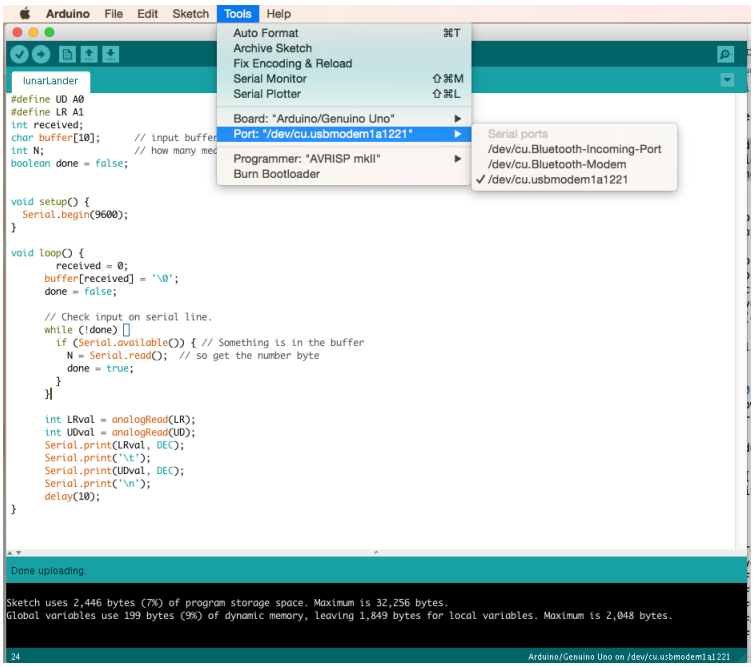


Figure 8: Select the serial port for your Arduiono

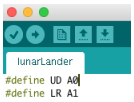


Figure 9: Select the serial port for your Arduiono

Running Lunar Lander with the Arduino game controller

30. I have already created a Jupyter notebook that you can use as a template. Download the file `lunar-lander-nb.ipynb` from our course web site. Save the file (or move the file) into the folder where you stored your first Jupyter notebook file that you used to test the software.

31. Go to the Jupyter browser window that shows your files and folders. Click on the file you just downloaded. It should open as a VPython notebook (as indicated in the top, right corner).

32. Run the first cell. (It imports packages.)

33. The second cell has the code necessary to communicate with the Arduino. You need to get the serial port for your Arduino. Go to the Arduino software, click the **Tools** menu and select **Port** . Right down the name of the serial port that your Arduino is connected to. In your VPython notebook, edit the `port` variable to match the name of the port used by your Arduino. Mine is:

```
#serial port for the Arduino; get the name for the port from the Arduino
    software
port = '/dev/cu.usbmodem1a1221'
```

Make sure the name of the port is contained within quotes since it is a string.

34. Run this cell and verify that there are no errors.

35. Run the third cell that contains the code for the game. You should see the lander, and you should be able to control it with your game controller.

Analysis

B Complete this exercise and do the following.

1. Upload a working notebook file produced at the end of this activity.

A Do everything for **B** with the following modifications and additions.

1. Write a new game that uses the game controller and runs in a notebook.