This is a custom XBee-based remote control. The remote uses a 7-segment display, 7 buttons (plus a reset button), and 6 analog inputs as a user interface, and a 6-pin FTDI header for reprogramming through the Arduino IDE. Default code includes a program that prints the current state of the buttons to the XBee and a program that uses a switch statement to select what to send based on the buttons being pressed.

I originally designed this board for the Summer@Brown education program. In a class for this program, students design, construct, and program an aluminum-framed robot, and compete with it in a basic navigational challenge. Students have the option of controlling their robots remotely for part of the exercise. Previously, the control systems of these robots were created using discrete gates on 7400 series logic chips, and the remote control functionality was provided by expensive RF demo boards from Texas Instruments. I decided to upgrade the systems by implementing an Arduino and XBee based control system on the robots. That way, the robots could be controlled by anything that could speak over a serial connection. I had the students build their own remote controls on a breadboard, but quickly found that the wires would pop out and the systems would break. To fix the problem, I designed this remote control. Students would still solder it together and program it, but it would be easier to hold, have more buttons, and be more reliable. Further, it would be expandable: If they outstripped the 7 switches, they could always attach more switches, potentiometers, or I2C communications devices through the analog lines.

While I designed this board to be paired with a control system mounted on a robot, it can be used to interface with anything that communicates over an XBee serial connection. As such, it could work as both a remote control and a remote monitoring system. The Atmega328P runs off 5 volts, while the XBee runs of 3.3 volts. To convert between the two, a low drop out regulator is used, along with a 74AHC125 buffer. However, the '125 has another useful feature. Using its active-low Output Enable pin, its outputs switched to High Impedance at will. Since the buffer is used on both the RX and TX lines, the Atmega328P can disconnect itself from the XBee.

Typically, when Arduinos and XBees are combined, the XBee cannot be connected to the Arduino during serial bootloading. To solve this problem, designers usually put in a DPDT switch to manually transition between connecting the Atmega328 to the computer and connecting it to the outside world. However, since the Atmega I/O lines are high impedance during reset and bootloading, I simply connect the ‘125 Output Enable pin to the Atmega, and install a pull-up resistor on the line. This means that during bootloading, the Atmega is disconnected from the XBee. When the program is loaded and the Atmega is ready to communicate wirelessly, it simply pulls the Output Enable pin low.