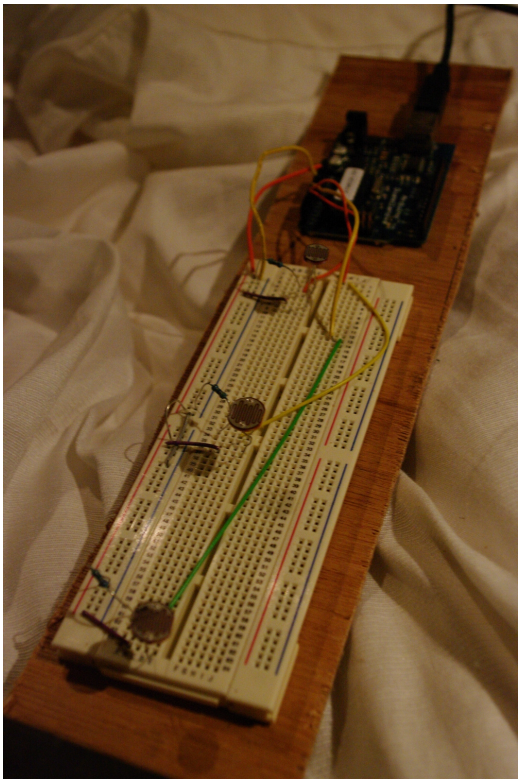
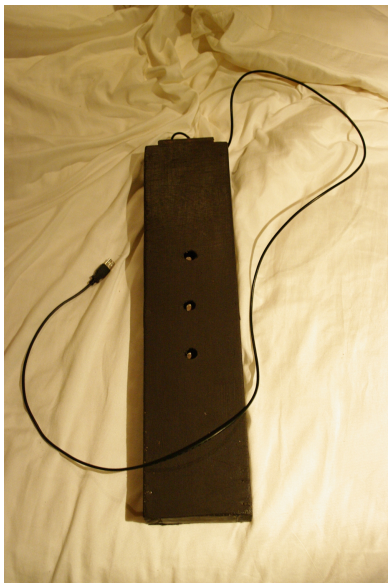
Working with Photo-resistor input

Photo-resistors detect variations in light and communicate this as variations in voltage resistance in a circuit. We can capture these sensor values, restrict them to byte values (0 – 255) and pass them via serial communication through to the Scala-based graphics engine (Loom). These values can then modify any specific program parameters that you wish.

Here is a basic handmade input device – a parallel circuit of 3 photo-resistors embedded in a black box with three holes.





What could this control? Any number of things. You could make it so that the lowest value from any particular photo-resistor determined the rendering mode (points, lines or filled) or selected a different image loop. It's up to the production groups to consider the various possibilities.

Here are some step by step instructions to get you up and running.

1. Begin by checking out the Loom engine. It is a bit different than the old Scala drawing/animation engine. You need to read the Loom.pdf and consult the API documentation.

2. This should give you the skills necessary to run the two sample programs - HeightMap and ImageLoop. HeightMap now includes a spinning ground plane, a semi-transparent scull image and some depth dependent sound. It demonstrates how 3D display, images and sound can be linked. The other sketch is a new version of ImageLoop. In default mode it just shows

an endlessly repeated squiggle, but later in serial input mode it can swap to different image sequence animations. Having looked at these two examples you need to decide as a group what you would like to do with either photo-resistor input. We will discuss the possibilities in class.

3. Having come up with your groups creative concept the next step is to create an Arduino circuit based on photo-resistor input. See the weekly notes for week 10 on eLearning. Note that you can chain together a set of photo-resistors if you need multiple sensor readings (just make sure to arrange in a parallel circuit). Please speak to our technical support officer, Glenn Alexander, if you need assistance putting together your circuit. Here are Glenn's contact details: (extn: 5847, email: glenn_alexander@uow.edu.au). He is based on the IC campus so you may need to travel over there for help. Alternatively, he will be attending class over the next few weeks.

4. Having created the circuit you need to test it with the photo-resistor based Arduino sketch provided in the Loom directory. You may need to adapt the code depending upon the number of photo-resistors in your circuit – my sketch includes 3. You should then compile and upload the sketch to your circuit and see how it works. Worth turning on Serial Monitor to check that the circuit is producing appropriate values. Individual values must fall in the range 0 – 255.

5. Having built and tested your circuit the next step is to link it up to the Loom graphics engine. Go into the sketches directory. You will find versions of the HeightMap and ImageLoop sketches. Select the one that you'd like to work with/adapt. The HeightMap example uses photo-resistor input to modify the rendering mode. The program works out the lowest photo-resistor value (the one that is being, perhaps, covered by a user's hand) and then changes the rendering mode to the one that is linked to that index. The ImageLoop sketch does something similar but instead of adjusting the renderer it switches the image sequence currently displayed. Make sure that your circuit is running and attached via USB to the computer. Then go to the config directory in your selected sketch. There should be three files:

    - config_default.xml (plays without serial input)
    - config_photoresistor.xml (works with photo-resistor circuit)
    - config_rfid.xml (works with RFID circuit).

6. You need the second one. Open up the photo-resistor configuration file and check the settings. It may need editing in terms of the number of photo-resistor sensor readings. This is listed in the final parameter – quantity. For 3 photo-resistors you need to set a 'quantity' of 4 (Bytes). The first Byte is a default -5 value (the start Byte) and the remaining 3 Bytes store your sensor readings as values between 0 and 255.

6. Go into your sketch directory and open up the MySketch.scala file. Read it thoroughly to get an understanding of how it works. Pay particular heed to the update() method. You will find examples of dealing with byte based sensor readings. The latter are accessible as an array of values in the serialByteReadings property (variable). This variable gets filled with new readings as fast as they can come in via serial communication. The array contains one less value than the number you specified in 'quantity' (in the config file) – so serialByteReadings contains 3 values in my example, corresponding to the three Byte values coming in from my 3 photo-resistors.

7. Now you need to modify the MySketch code to do whatever you want to accomplish. You also need to build a compelling interaction framework. A black box is a bit dull and obvious. What can you design that creates an interesting relationship between user interaction and screen display?

8. Now copy the photo-resistor build files in your sketch directory into the main loom directory. Use these to build and test your project. Consult the Loom pdf for more information.

9. PLEASE NOTE THAT YOU CANNOT RUN THE SERIAL VERSIONS OF HEIGHTMAP AND IMAGELOOP IF YOU DO NOT HAVE SERIAL DEVICES CONNECTED AND SET TO THE CORRECT PORT – MISTAKES HERE WILL CAUSE A CASCADE OF ERROR MESSAGES.