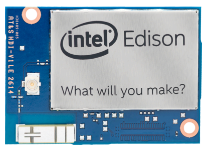


Intel® Edison Tutorial: IoT Machine Learning System Reference Design

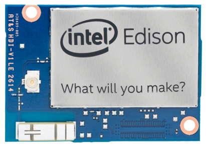
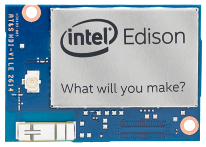


Table of Contents



Introduction

This Reference Design provides an introduction to the development of Machine Learning Systems on the Intel Edison platform. This includes instructions for installation of a Machine Learning Neural Network open source library. This also includes instructions for development of a complete sensing system that includes Neural Network construction and training.

The Neural Network Reference Design provides an introduction to Neural Network operation by enabling a sensor system that detects position using an Intel Edison platform as well as sensors available in the standard Intel Edison IoT Grove kit.

This provides instructions on how to use a provided software system to build a sensor network that will determine location of an object. This applies light sensors to detect shadows cast by an object. Once a shadow is detected, the system will classify the location of the shadow into one of four regions **in real time**. The key concepts used in this Reference Design are listed below.

1. Neural Network machine learning concepts
2. Sensor interfaces and sensor data acquisition
3. Modifying physical attributes of sensors to provide properly scaled signal level.

Users will first build and use the reference design. While building the reference design, the concepts of interfacing and modifying the sensors will be examined. Once the system is operational, users will learn more about machine learning and neural networks.

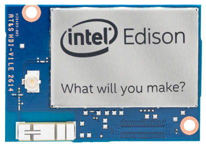
Prerequisite Tutorials

Users should ensure they are familiar with the documents listed below before proceeding.

1. Intel Edison Tutorial – Introduction, Linux Operating System Shell Access and SFTP
2. Intel Edison Tutorial – Introduction to Linux
3. Intel Edison Tutorial – Introduction to Vim
4. Intel Edison Tutorial – GPIO, Interrupts and I2C

List of Required Materials and Equipment

1. 1x Intel Edison Kit.
2. 2x USB 2.0 A-Male to Micro-B Cable (micro USB cable).
3. 1x powered USB hub **OR** 1x external power supply.
4. 1x Grove – Start kit for Arduino.
5. 1x Opaque Electrical Tape.
6. 3x Grove Compatible Analog Light Sensors.
7. 1x Personal Computer.
8. Access to a network with an internet connection.



Software Setup

1. Access the shell on the Intel Edison. For more information, refer to the document labelled ***Intel Edison Tutorial – Introduction, Shell Access and SFTP***.
2. Ensure that the Intel Edison has the Vim editor installed. For more information, refer to the document labelled ***Intel Edison Tutorial – Introduction to Vim***.
3. Ensure that the Intel Edison has Git. For more information, refer to the document labelled ***Intel Edison Tutorial – Introduction to OPKG***.
4. Ensure that the Intel Edison has internet access.
5. Navigate to the home directory.

   ```
   $ cd
   ```

6. Clone the source code by issuing the command shown below.

   ```
   $ git clone https://github.com/pban1993/edison_fann_shadows.git
   ```

7. Navigate to the directory **edison_fann_shadows**.

   ```
   $ cd ~/edison_fann_shadows
   ```

8. Inspect the contents of the directory. It contains two sub folders.

   The first subfolder contains shell scripts that enable access to the FANN library.

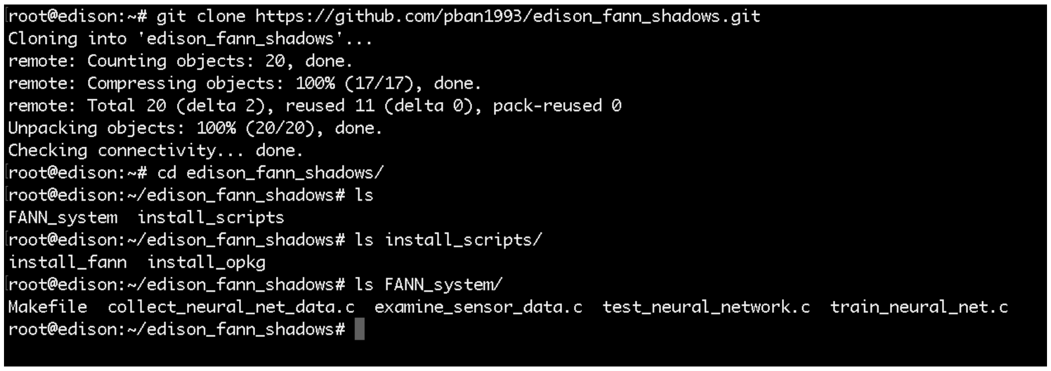
   install_scripts/install_fann.sh
   install_scripts/install_opkg.sh – only relevant if you have never updated opkg.

   The second subfolder contains C-code source files to examine sensor data, train a FANN system, and enable shadow detection.

   FANN_system/collect_neural_net_data.c
   FANN_system/collect_neural_net_data.c
   FANN_system/examine_sensor_data.c
   FANN_system/test_neural_network.c
   FANN_system/train_neural_network.c
   FANN_system/Makefile

   For more details about FANN, open the link below on a web-browser on a personal computer.

   http://leenissen.dk/



```
root@edison:~# git clone https://github.com/pban1993/edison_fann_shadows.git
Cloning into 'edison_fann_shadows'...
remote: Counting objects: 20, done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 20 (delta 2), reused 11 (delta 0), pack-reused 0
Unpacking objects: 100% (20/20), done.
Checking connectivity... done.
root@edison:~# cd edison_fann_shadows/
root@edison:~/edison_fann_shadows# ls
FANN_system  install_scripts
root@edison:~/edison_fann_shadows# ls install_scripts/
install_fann  install_opkg
root@edison:~/edison_fann_shadows# ls FANN_system/
Makefile  collect_neural_net_data.c  examine_sensor_data.c  test_neural_network.c  train_neural_net.c
root@edison:~/edison_fann_shadows#
```

Figure 1: Output after successfully cloning and inspecting the contents of the git repository.

9. Navigate to the **install_scripts** directory

   $ cd ~/edison_fann_shadows/install_scripts

10. Install **git** by issuing the below command. Skip this step if the Intel Edison already has **git** installed.

    $ sh install_opkg.sh

11. Install the **FANN library** by issuing the following command.

    $ sh install_fann.sh

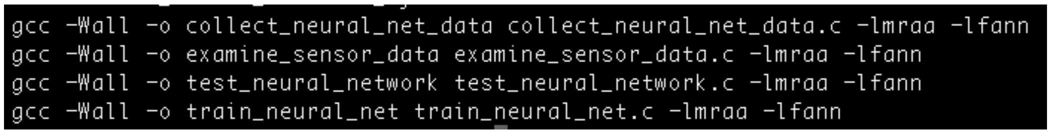
    This step may require about 10 minutes to successfully complete.

12. Navigate to the directory containing the C code source files and the Makefile.

    $ cd ~/edison_fann_shadows/FANN_system

13. Compile the code source files into executable binary files.

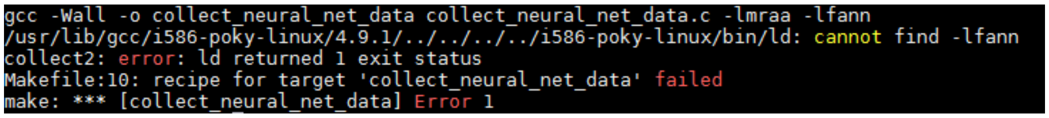
    $ make



```
gcc -Wall -o collect_neural_net_data collect_neural_net_data.c -lmraa -lfann
gcc -Wall -o examine_sensor_data examine_sensor_data.c -lmraa -lfann
gcc -Wall -o test_neural_network test_neural_network.c -lmraa -lfann
gcc -Wall -o train_neural_net train_neural_net.c -lmraa -lfann
```

14. **Figure 2: Output from gcc when binary files have been successfully compiled**

If **gcc** produces the below error, verify that your Intel Edison has internet access, and repeat steps 10 to 12.



```
gcc -Wall -o collect_neural_net_data collect_neural_net_data.c -lmraa -lfann
/usr/lib/gcc/i586-poky-linux/4.9.1/../../../../i586-poky-linux/bin/ld: cannot find -lfann
collect2: error: ld returned 1 exit status
Makefile:10: recipe for target 'collect_neural_net_data' failed
make: *** [collect_neural_net_data] Error 1
```

15. **Figure 3: Output from gcc when the FANN library has not been successfully installed**

If the problem persists, please contact your course instructor.

16. The software has now been correctly configured.

Hardware Setup

1. Power down the Intel Edison. Remove all USB and power supply cables.
2. Attach the Grove Shield to the Intel Edison board.
3. Ensure the Grove shield is operating at 5V by adjusting the switch.

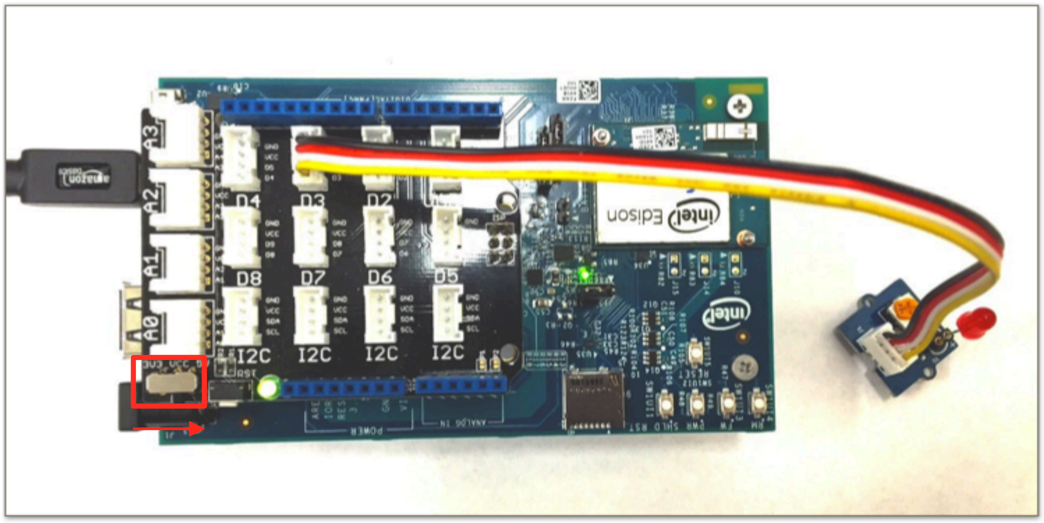


Figure 4: Correct hardware configuration to blink an LED

4. Print out the PDF document with this tutorial labelled **setup_page.pdf**.
5. Cut a small piece of rectangular electric tape from the roll.
6. Wrap the ends together with the adhesive side facing outside.
7. Attach the tape to one of the legs of the Intel Edison board.

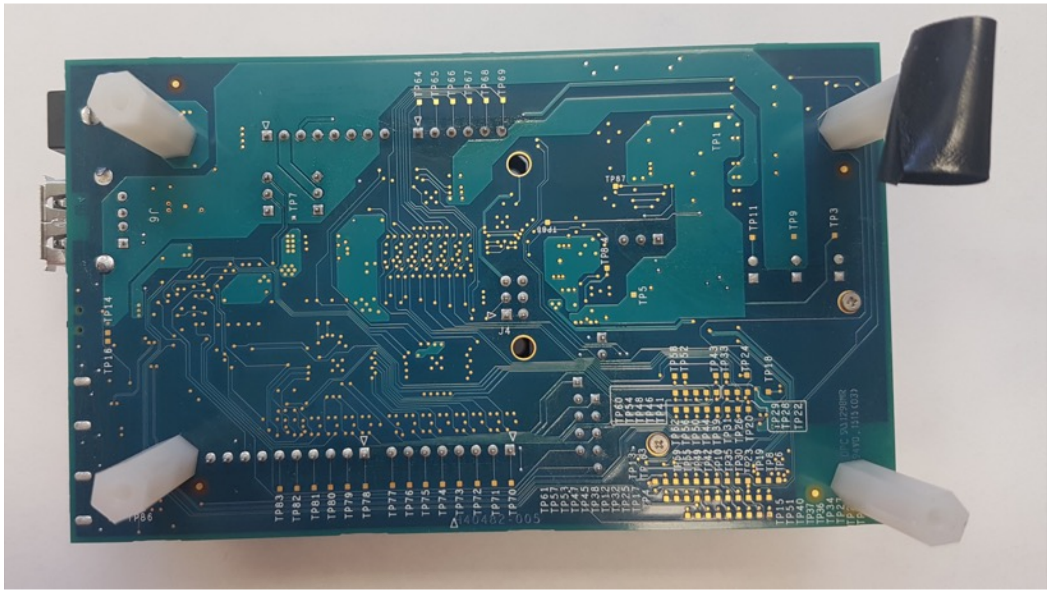


Figure 5: Attaching the tape to one leg of the Intel Edison board

8. Repeat steps 4-6 until a piece of electric tape is attached to each leg.

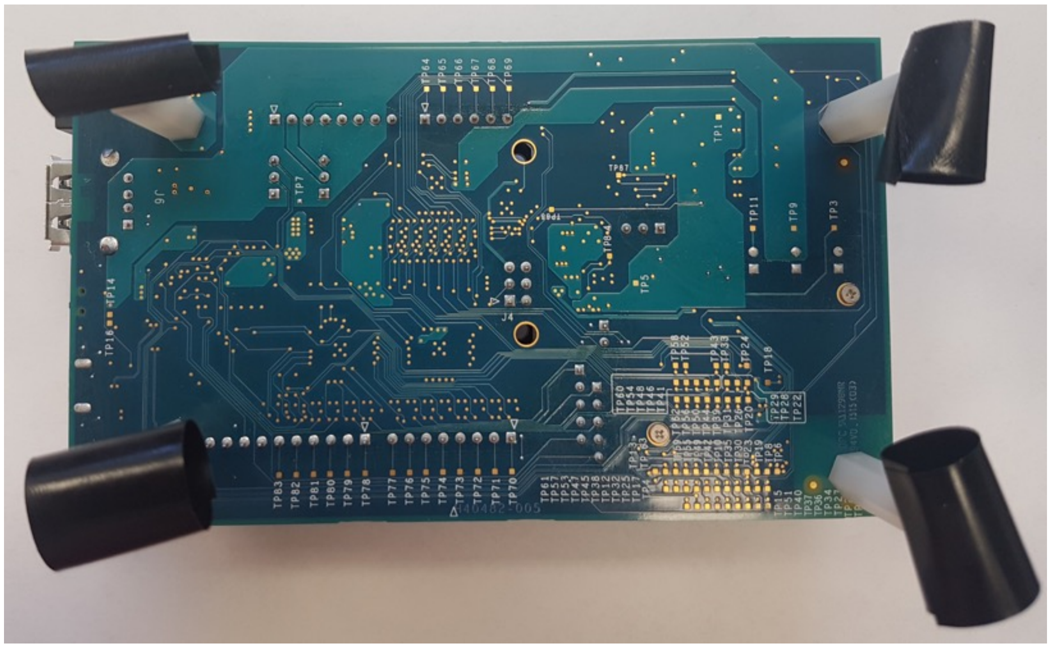


Figure 6: Attaching tape to the other three legs of the Intel Edison board

9. Place the Intel Edison board with the legs facing downwards to the printout from step 1 in the large rectangular area. Press down on the Edison Board for Arduino.

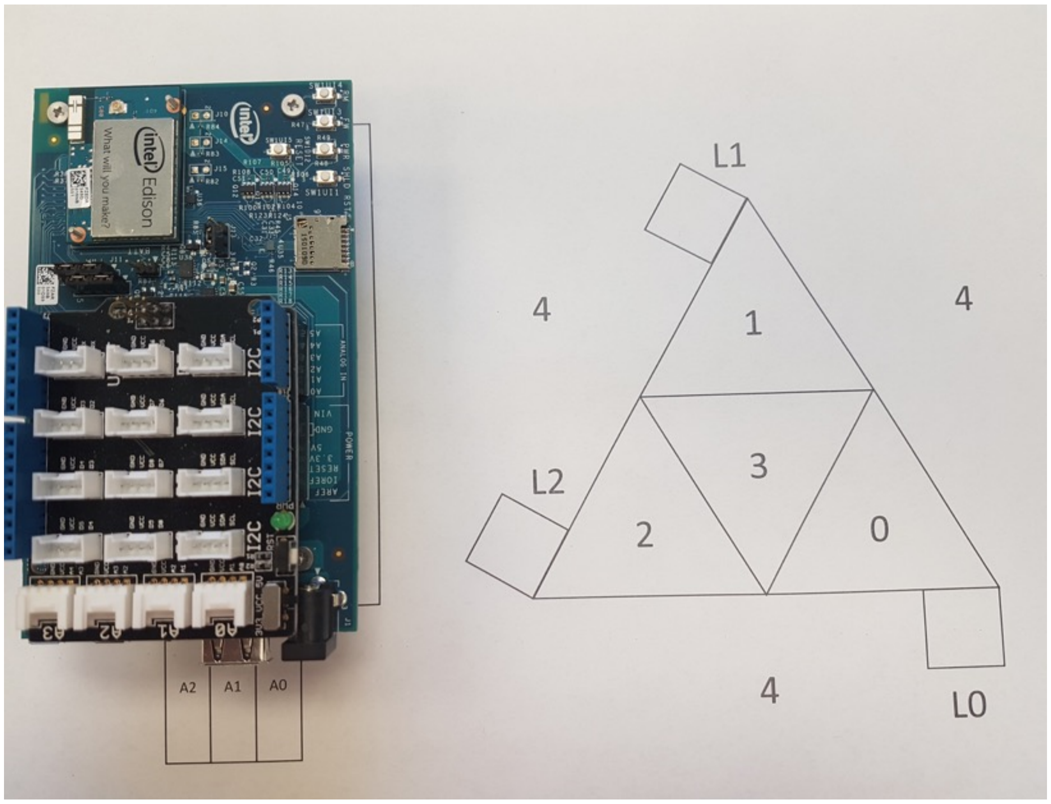


Figure 7: Ensure the orientation of the Intel Edison kit matches the above figure

10. Cut a small piece of rectangular black electric tape from the roll.
11. Wrap the ends together with the adhesive side facing outside.
12. Attach the tape to the **bottom** of the sensor (this is the side that **does not include the light sensor)**.

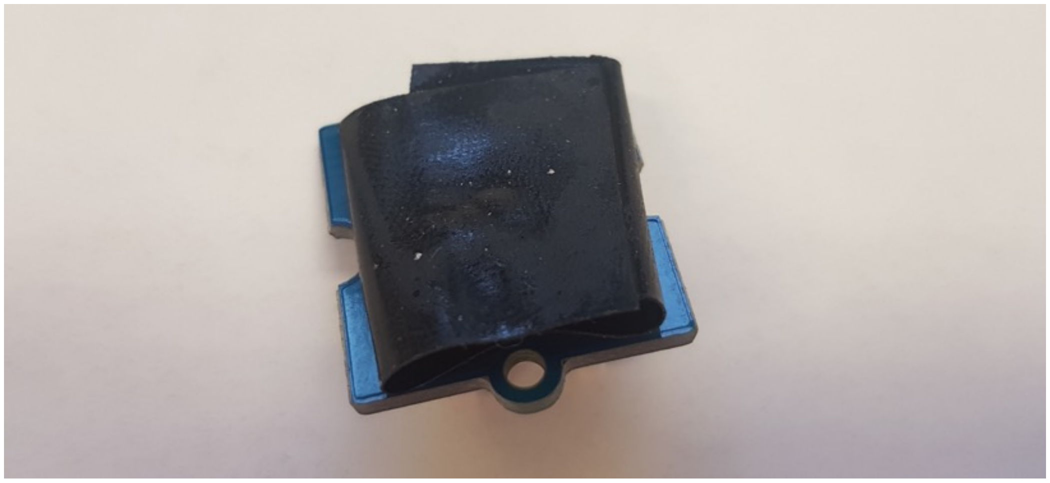


Figure 8: Attaching tape on the bottom of the sensor

13. Examine and take careful note of how the light sensors are oriented in Figure 9. Attach the sensors one by one to match this figure. Deviation from this orientation may cause the system to function incorrectly.

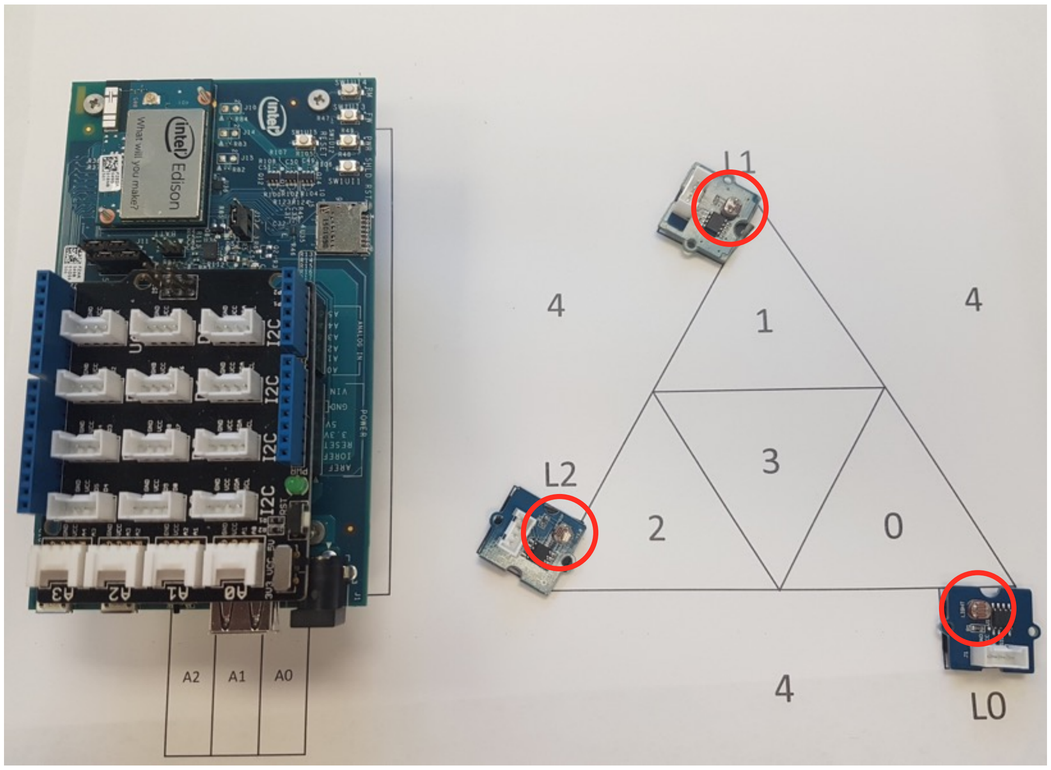
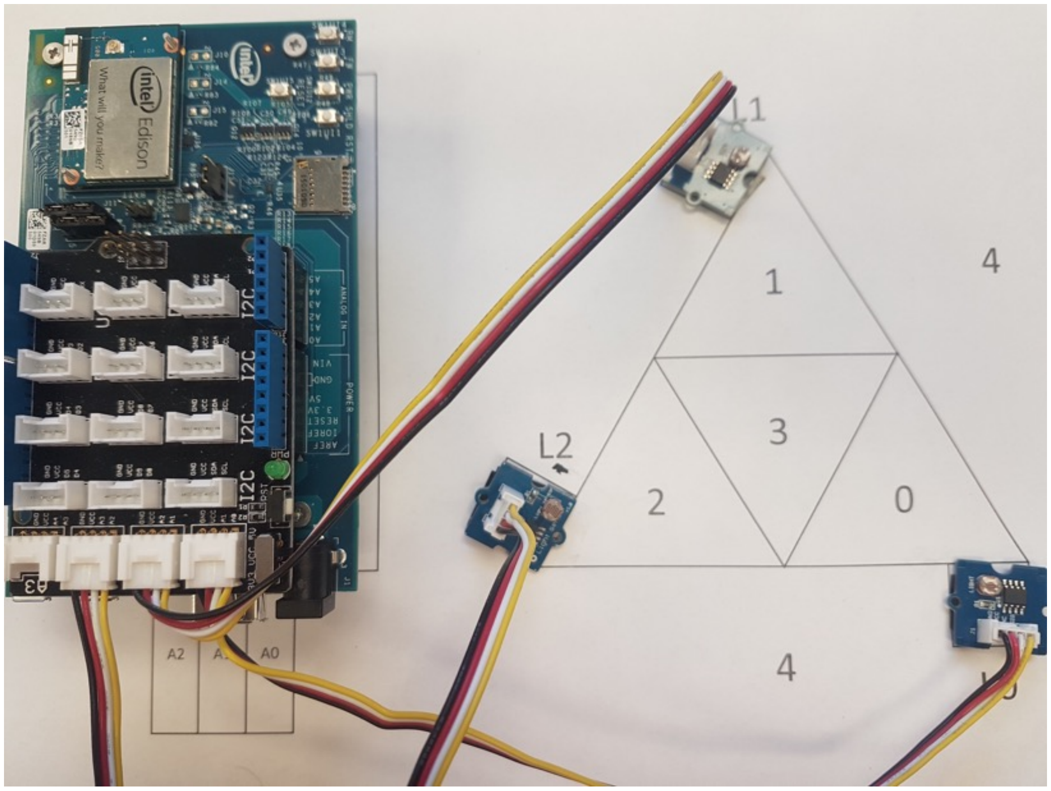


Figure 9: Correct sensor orientation and configuration for optimal system performance

14. Use the connectors to establish an interface between each sensor and the Intel Edison. Create bends in the connectors if they are creating a force which is creating curvature in the paper. These connectors can be found under the plastic, vacuum molded component holder in the Grove – Starter kit for Arduino.

    A0 -> L0
    A1 -> L1
    A2 -> L2



15. Access the shell on your Intel Edison using SSH.

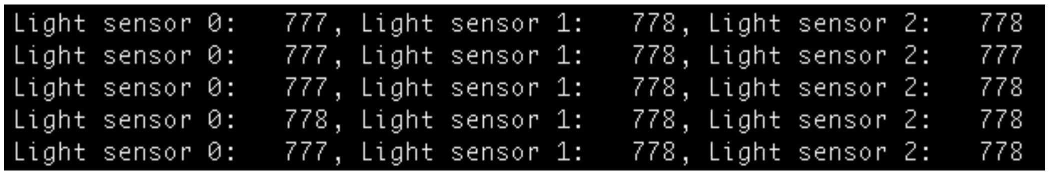
16. Navigate to the **edison_fann_shadows/FANN_system** directory.

    $ cd ~/edison_fann_shadows/FANN_system

17. Run the **examine_sensor_data** executable binary file.

    $ ./examine_sensor_data

    Notice how the digital approximation of the analog data points that the sensors are producing are near 700 when there are no shadows being cast on the light sensors.


```
Light sensor 0:   777, Light sensor 1:   778, Light sensor 2:   778
Light sensor 0:   777, Light sensor 1:   778, Light sensor 2:   777
Light sensor 0:   777, Light sensor 1:   778, Light sensor 2:   778
Light sensor 0:   778, Light sensor 1:   778, Light sensor 2:   778
Light sensor 0:   777, Light sensor 1:   778, Light sensor 2:   778
```
Figure 10: Digital approximation of light intensity values as provided analog output from light sensors in Grove – Starter kit for Arduino

Note: Do not terminate the **examine_sensor_data** process, it is required to examine the output while performing the steps below.

18. The light sensor system in your Intel Edison IoT Kit is a photoconductor sensor. This device exhibits a change in conductance when light intensity is received by the sensor. An interface circuit is integrated with the sensor providing a voltage signal to the Edison platform.
19. The sensor system value ranges from zero in complete lack of illumination to a maximum value of greater than 700.
20. The sensor systems now require adjustment in order to ensure that changes in light intensity, resulting from casting shadows on the sensors, yields a change in measured light intensity.
21. A typical room environment provides an illumination level that *saturates* the light intensity sensor output at its maximum value.
22. Therefore, the sensors must be modified to limit the received light intensity such that the light intensity value is approximately 100 to 500 in a room with typical illumination.

    Follow the below instructions to make this physical modification.

23. Please note the image of the photoconductor sensor below.

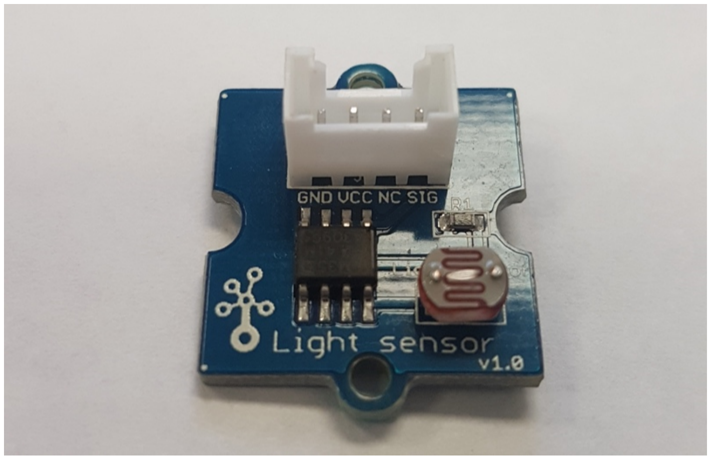


Figure 11: The photoconductor light intensity sensor system. The photoconductor sensor is the device at lower right on this printed circuit board.

24. Cut four small strips of black electrical tape and place these over the sensor element as in the schematic drawing below.
25. Place the tape strips on the sensor such that they create a small, restricted area region as shown in the Figure 10, below. This restricted area region admits a reduced light intensity to the sensor element.
26. Adjust the tape separation and therefore the aperture size such that the sensor output should lie between 100-500 with no shadows being cast upon it.

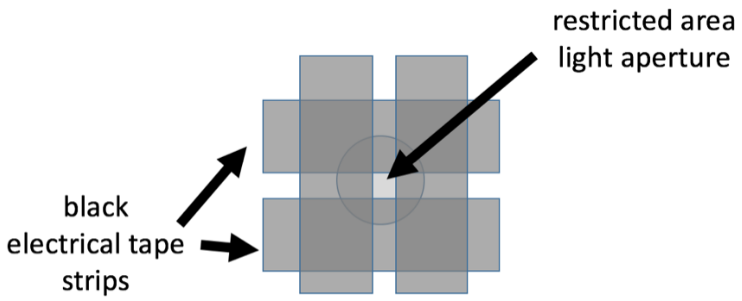


Figure 12: Creating a restricted area light aperture by application of four black electrical tape strips to the sensor element. Note that there is a region of about 0.5 to 1mm in height and width that remains clear, admitting light via a restricted area to the sensor element.

27. An image of the sensor with four tape strips applied in sequence is shown below.

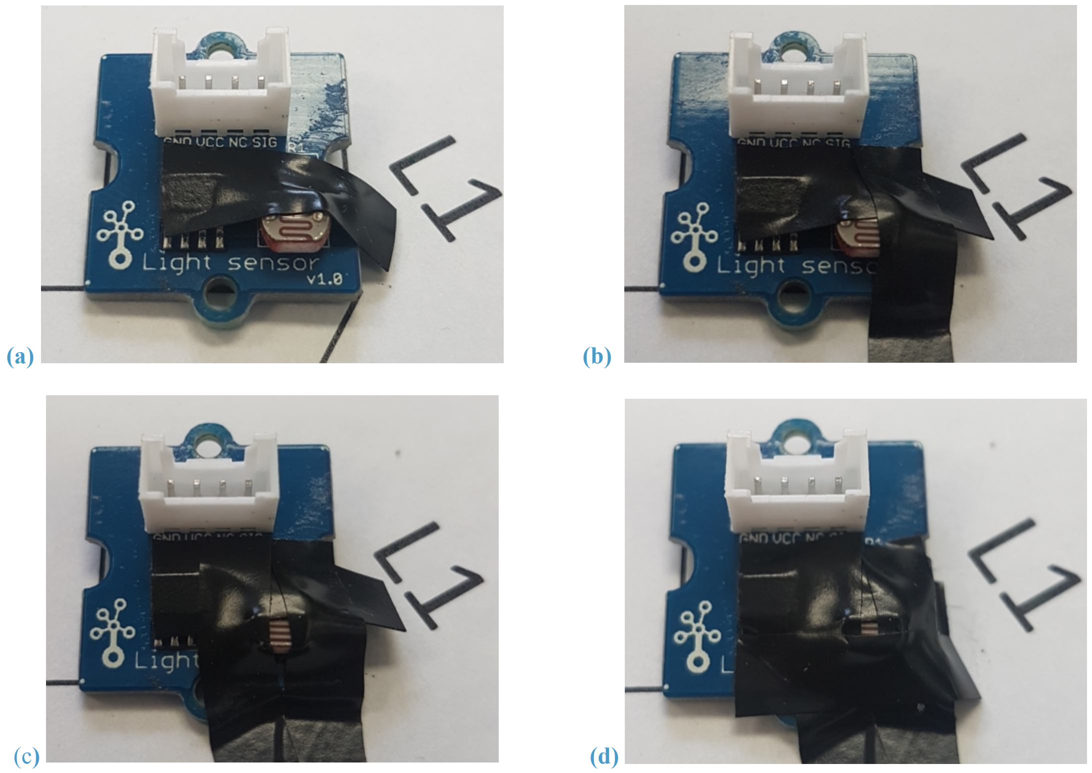


Figure 13: Image of the photoconductor sensor with tape strips applied in sequence of four steps.

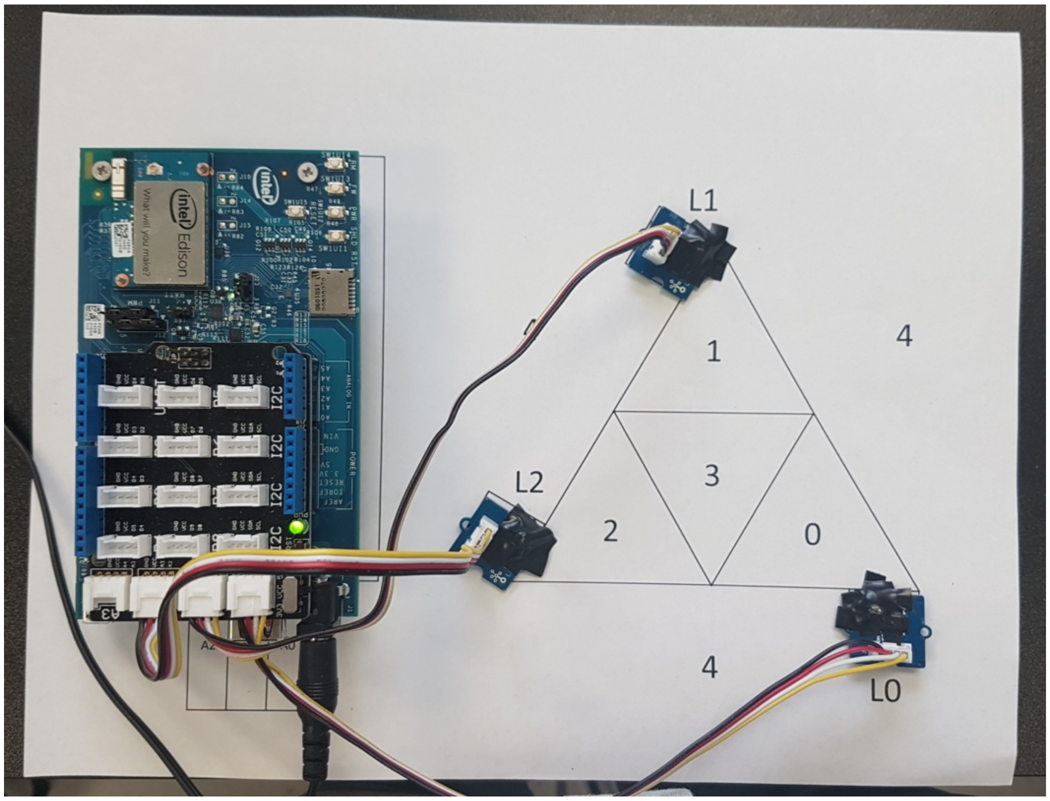
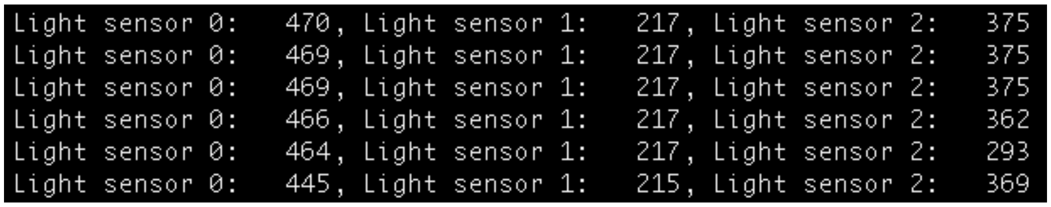


Figure 14. Final configuration of Intel Edison and Light Intensity Sensors

28. Repeat steps 18-20 until the value digital representation of the data point generated by the sensor lies between 100 and 500. The sensor values can vary between sensors (as shown by below screenshot). It is not required that all sensors display the same response. However, all sensor outputs should lie in the range of 100 to 500.



```
Light sensor 0:   470, Light sensor 1:   217, Light sensor 2:   375
Light sensor 0:   469, Light sensor 1:   217, Light sensor 2:   375
Light sensor 0:   469, Light sensor 1:   217, Light sensor 2:   375
Light sensor 0:   466, Light sensor 1:   217, Light sensor 2:   362
Light sensor 0:   464, Light sensor 1:   217, Light sensor 2:   293
Light sensor 0:   445, Light sensor 1:   215, Light sensor 2:   369
```

Figure 15: Digital approximation of analog data points after modification of light sensor

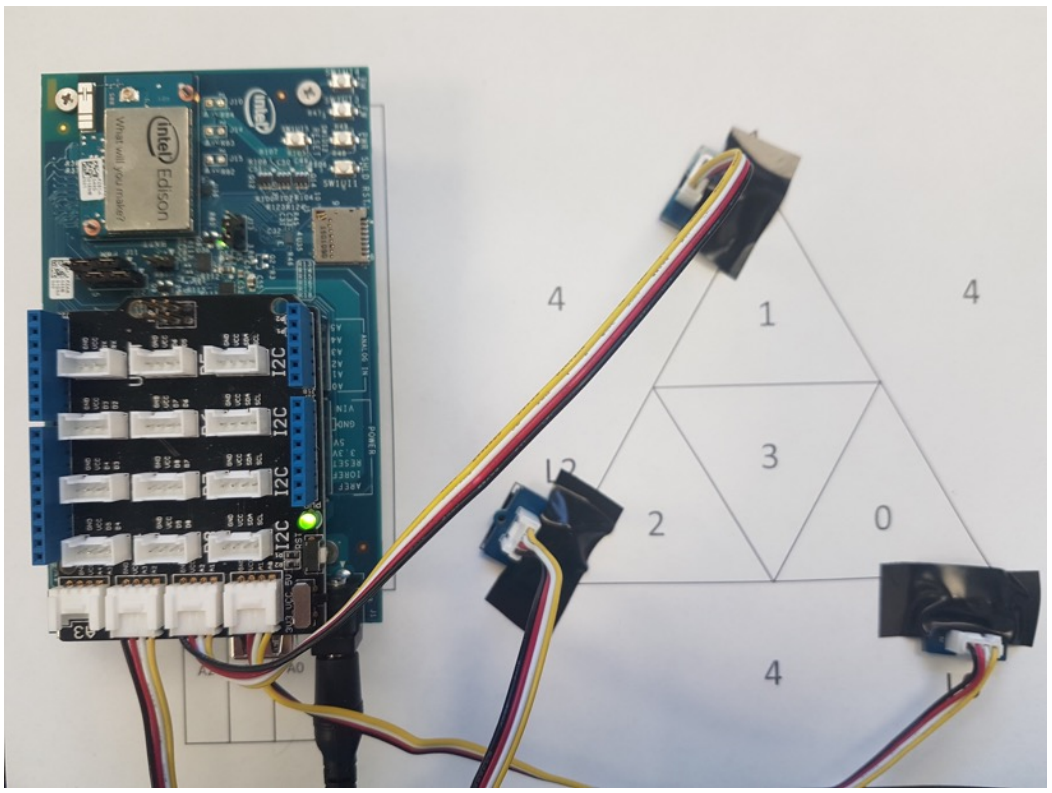


Figure 16: Shadow region detection system correctly configured.

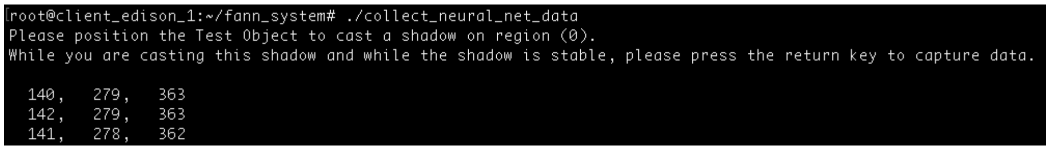
29. Terminate the **examine_sensor_data** process by pressing **ctrl-C**.

Execution of the Neural Network System

1. Now, run the **collect_neural_net_data** executable binary file.

   To cast a shadow, hold a small object (such as a playing card or smart phone) over the light sensor indicated by the onscreen instructions. For region 3, cast a shadow over all three light sensors. For region 4, do not cast a shadow.

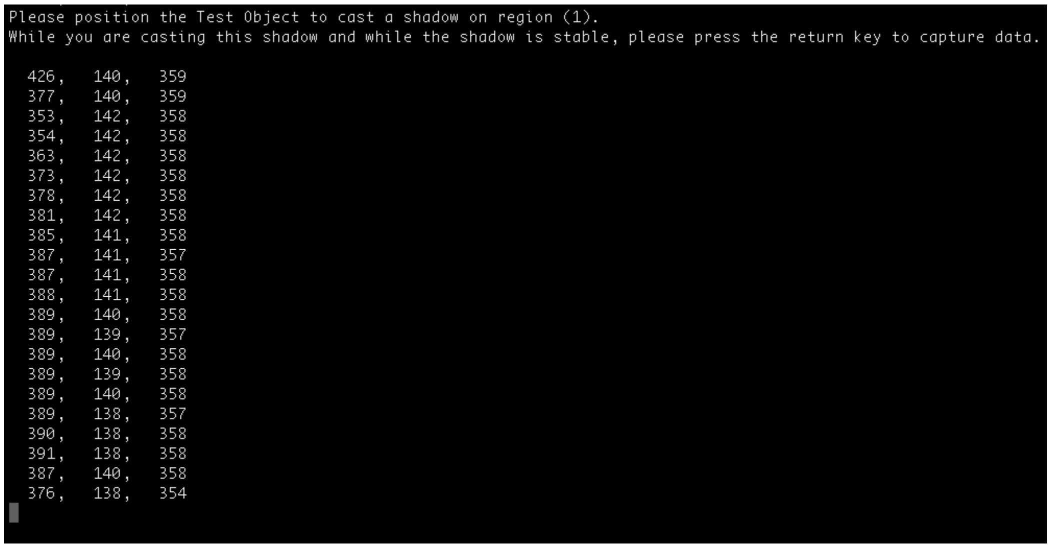
   $ ./collect_neural_net_data



```
root@client_edison_1:~/fann_system# ./collect_neural_net_data
Please position the Test Object to cast a shadow on region (0).
While you are casting this shadow and while the shadow is stable, please press the return key to capture data.

  140,   279,   363
  142,   279,   363
  141,   278,   362
```

Figure 17: Collecting neural network data for training purposes

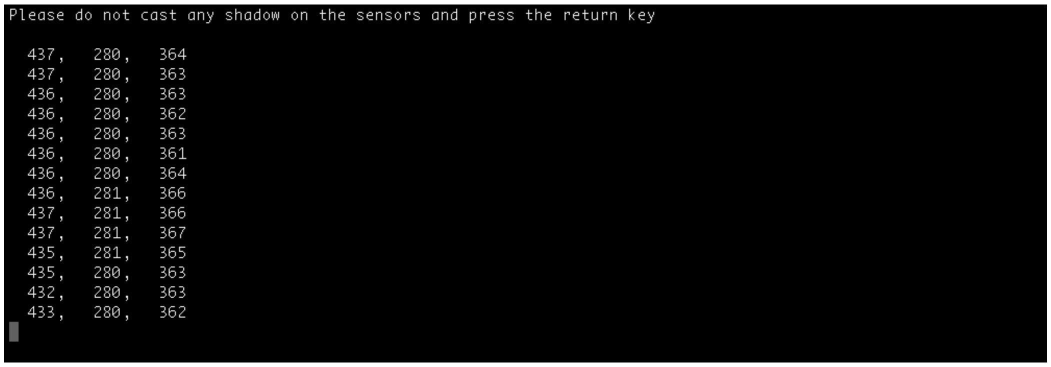


```
Please position the Test Object to cast a shadow on region (1).
While you are casting this shadow and while the shadow is stable, please press the return key to capture data.

  426,   140,   359
  377,   140,   359
  353,   142,   358
  354,   142,   358
  363,   142,   358
  373,   142,   358
  378,   142,   358
  381,   142,   358
  385,   141,   358
  387,   141,   357
  387,   141,   358
  388,   141,   358
  389,   140,   358
  389,   139,   357
  389,   140,   358
  389,   139,   358
  389,   140,   358
  389,   138,   357
  390,   138,   358
  391,   138,   358
  387,   140,   358
  376,   138,   354
```

Figure 18: Follow the onscreen instructions to generate data for neural network training

…

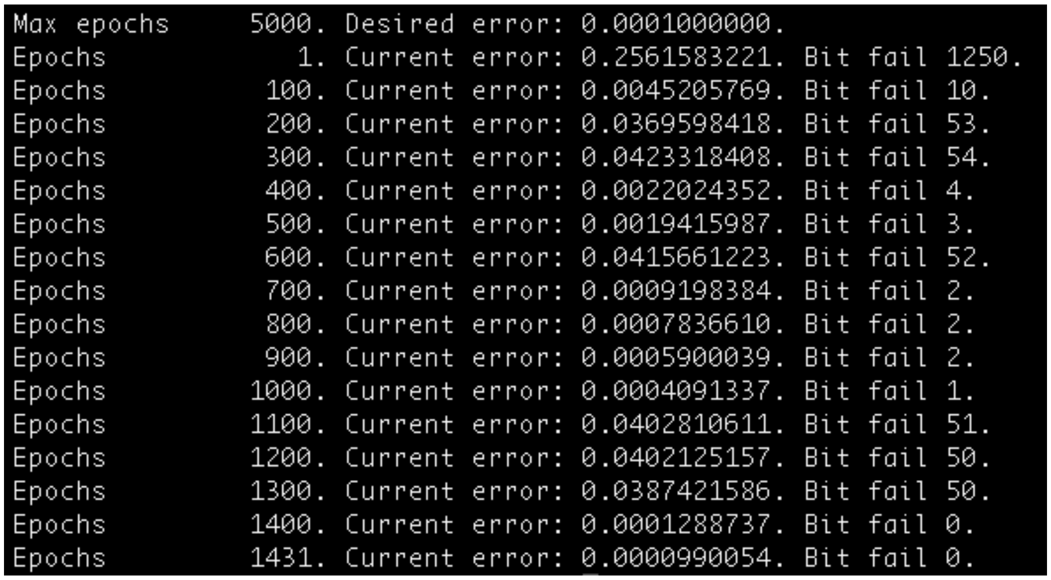


```
Please do not cast any shadow on the sensors and press the return key

  437,   280,   364
  437,   280,   363
  436,   280,   363
  436,   280,   362
  436,   280,   363
  436,   280,   361
  436,   280,   364
  436,   281,   366
  437,   281,   366
  437,   281,   367
  435,   281,   365
  435,   280,   363
  432,   280,   363
  433,   280,   362
```

Figure 19: If no light is being cast on the sensors, it will classify the shadow as being cast outside of the triangle

2. Run the **train_neural_network** executable binary file.
   $ ./train_neural_network



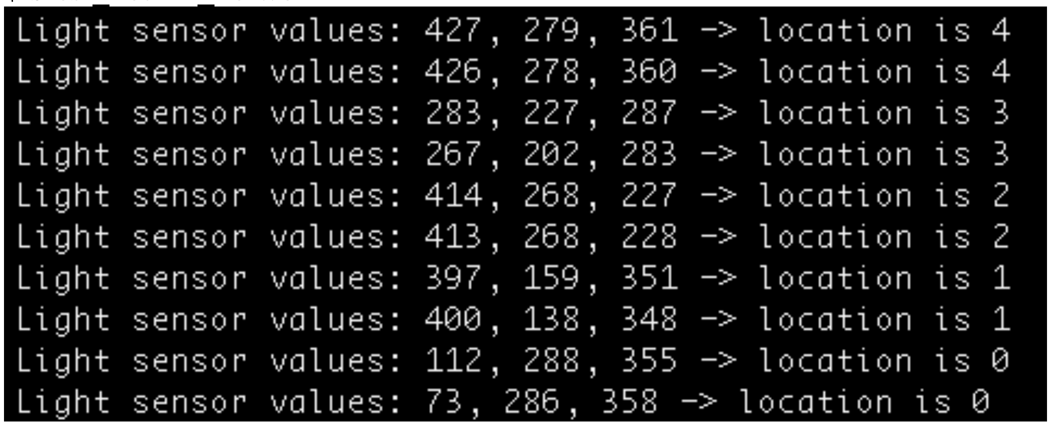
```
Max epochs     5000. Desired error: 0.0001000000.
Epochs            1. Current error: 0.2561583221. Bit fail 1250.
Epochs          100. Current error: 0.0045205769. Bit fail 10.
Epochs          200. Current error: 0.0369598418. Bit fail 53.
Epochs          300. Current error: 0.0423318408. Bit fail 54.
Epochs          400. Current error: 0.0022024352. Bit fail 4.
Epochs          500. Current error: 0.0019415987. Bit fail 3.
Epochs          600. Current error: 0.0415661223. Bit fail 52.
Epochs          700. Current error: 0.0009198384. Bit fail 2.
Epochs          800. Current error: 0.0007836610. Bit fail 2.
Epochs          900. Current error: 0.0005900039. Bit fail 2.
Epochs         1000. Current error: 0.0004091337. Bit fail 1.
Epochs         1100. Current error: 0.0402810611. Bit fail 51.
Epochs         1200. Current error: 0.0402125157. Bit fail 50.
Epochs         1300. Current error: 0.0387421586. Bit fail 50.
Epochs         1400. Current error: 0.0001288737. Bit fail 0.
Epochs         1431. Current error: 0.0000990054. Bit fail 0.
```

Figure 20: Standard output while training a neural network using the FANN library

The **Bit fail** metric is a measure of classification accuracy by reporting Neural Network neurons whose output value does not meet the training objective. If the **Bit fail** metric does not fall to values less than 20, the neural network may not accurately classify regions in which the shadow is being cast.

3. Run the **test_neural_network** executable binary file. While it is running, cast a shadow over various locations and see how the classification algorithm performs.

   $ ./test_neural_network



```
Light sensor values: 427, 279, 361 -> location is 4
Light sensor values: 426, 278, 360 -> location is 4
Light sensor values: 283, 227, 287 -> location is 3
Light sensor values: 267, 202, 283 -> location is 3
Light sensor values: 414, 268, 227 -> location is 2
Light sensor values: 413, 268, 228 -> location is 2
Light sensor values: 397, 159, 351 -> location is 1
Light sensor values: 400, 138, 348 -> location is 1
Light sensor values: 112, 288, 355 -> location is 0
Light sensor values: 73, 286, 358 -> location is 0
```

Figure 21: Neural network classifying the region in which the shadow is being cast

4. If the Neural Network is not classifying the shadows accurately, retrain the system by repeating steps 1-3.