**Bitácora**

**Miércoles 28 de febrero de 2018**

El grupo de trabajo investigó el problema junto a su Dataset, deduciendo que el sistema podría ser modelado con 4 neuronas en la capa de entrada para obtener las características del motor, 1 neurona de salida para decir la clase de motor e inicialmente 10 nodos en la capa oculta para hacer el procesamiento de la red neurona.

Una vez se definió la anterior, se estudian los algoritmos desarrollados por los expertos José Luis Paniagua y Jesús López que brindan una clara base acerca de como abordar dicho problema, prestando particular atención en los temas de como adquirir y transformar los datos para que sean legibles para una red neuronal desde un archivo .csv, validación en problemas de regresión y gráfica de la función de perdida de la red además de sus datos de salida.

Por otra parte, el grupo indaga en la página de TensorFlow las diferentes funciones de activación (<https://www.tensorflow.org/api_guides/python/nn>) y los métodos de optimización (<https://www.tensorflow.org/api_guides/python/train>) que permitan variar el desempeño de la red. A partir de este punto se realizó un algoritmo en Python capaz de leer los archivos .csv, convertirlos en matrices legibles por la red, realizar el respectivo entrenamiento de la misma juntos a su validación y por último enviar los valores de los pesos sinápticos a un archivo .csv para realizar la posterior validación de la red en Arduino.

Las siguientes imágenes corresponden a resultados obtenidos con el código variando diferentes parámetros como la razón de aprendizaje, el momentum, número de nodos en la capa oculta o el optimizador a utilizar. Cabe destacar que la función de activación de la capa oculta se utilizó tipo sigmoide debido a que los datos están comprendidos en valores mayores a cero y la función de activación de la capa de salida es lineal ya que la salida se encuentra comprendida entre 0.1 y 7.1 aproximadamente.

**Prueba con el optimizador de Gradiente Descendente, con un Alpha=0.1, 10 nodos en la capa oculta y 10000 épocas.**

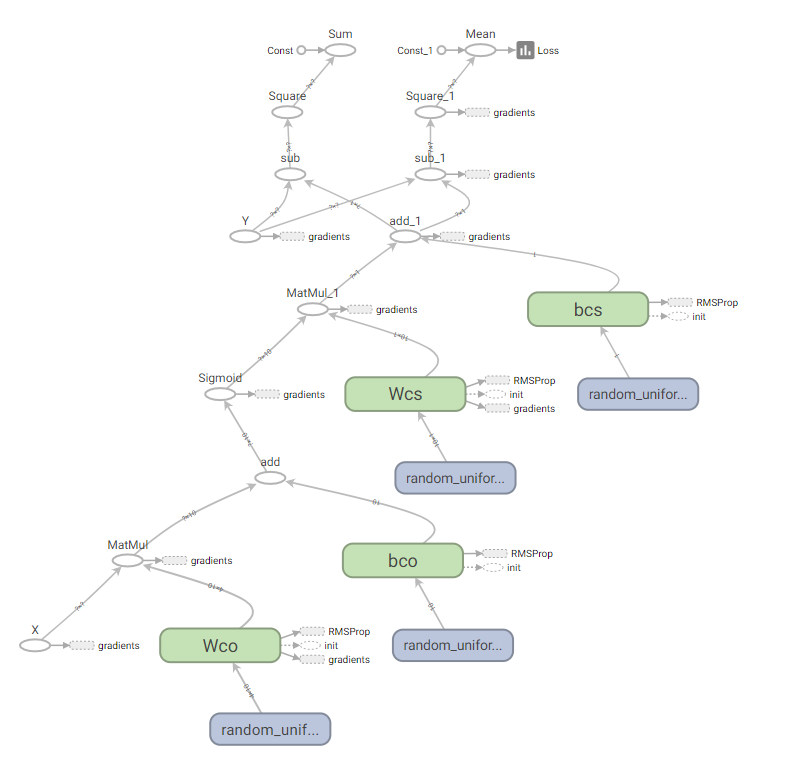


Fig. 1. Grafo de la red neuronal superficial.

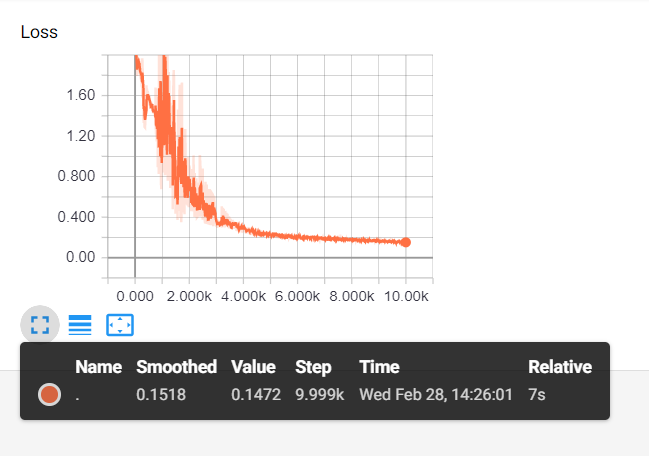


Fig. 2. Gráfica de la función de perdida de la red neuronal

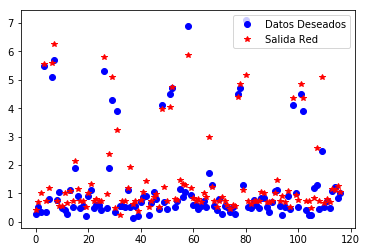


Fig. 3. Gráfica de los datos deseados contra la salida de la red, con un loss de 0.22874111

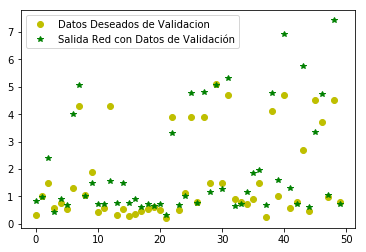


Fig. 4. Gráfica de los datos de validación deseados contra la salida de la red, con un loss de 0.9894685

**Prueba con el optimizador de Gradiente Descendente, con un Alpha=0.1, 15 nodos en la capa oculta y 10000 épocas.**

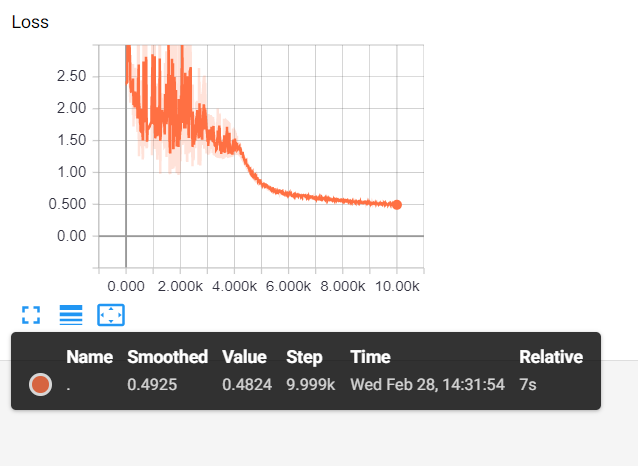


Fig. 5. Gráfica de la función de perdida de la red neuronal

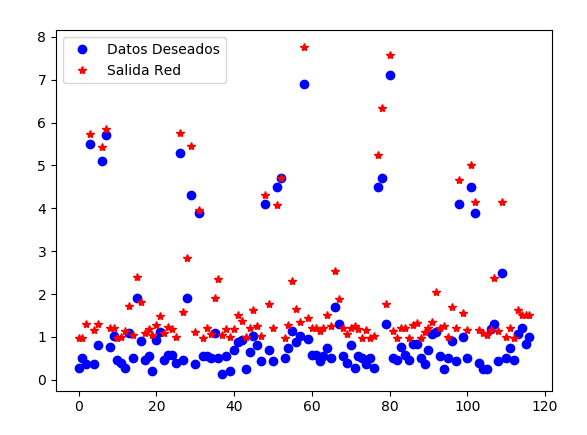


Fig. 6. Gráfica de los datos deseados contra la salida de la red, con un loss de 0.48243296

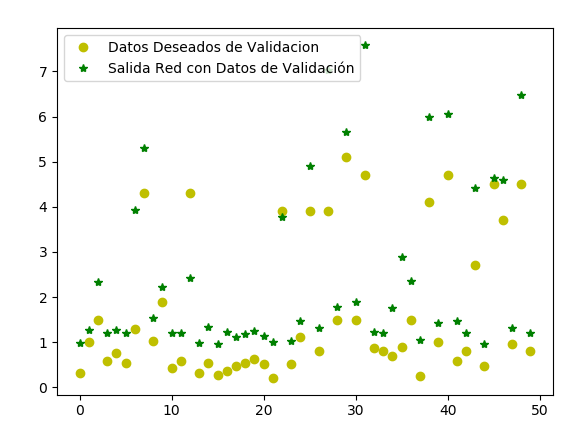


Fig. 7. Gráfica de los datos de validación deseados contra la salida de la red, con un loss de 1.2293872

**Prueba con el optimizador de Gradiente Descendente, con un Alpha=0.01, 10 nodos en la capa oculta y 10000 épocas.**

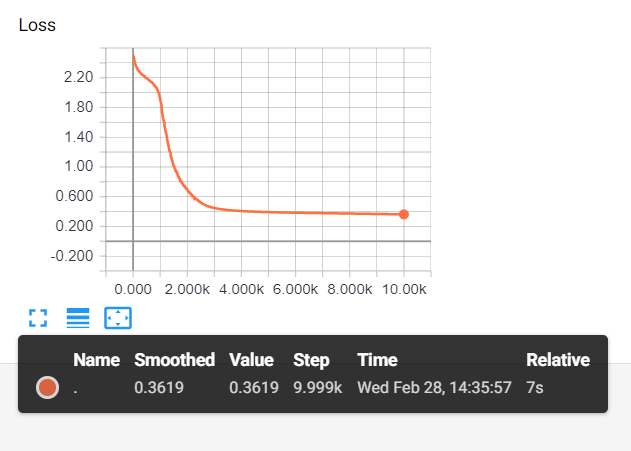


Fig. 8. Gráfica de la función de perdida de la red neuronal

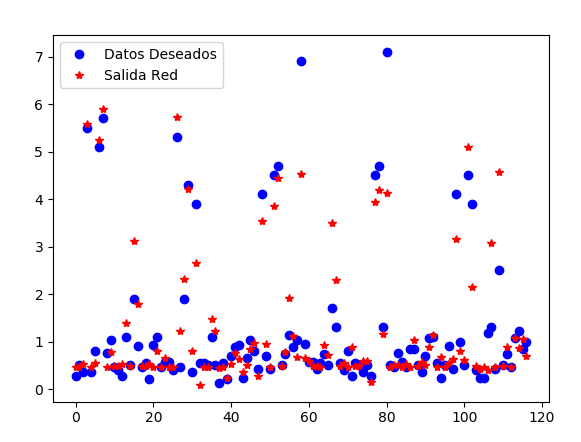


Fig. 9. Gráfica de los datos deseados contra la salida de la red, con un loss de 0.36186016

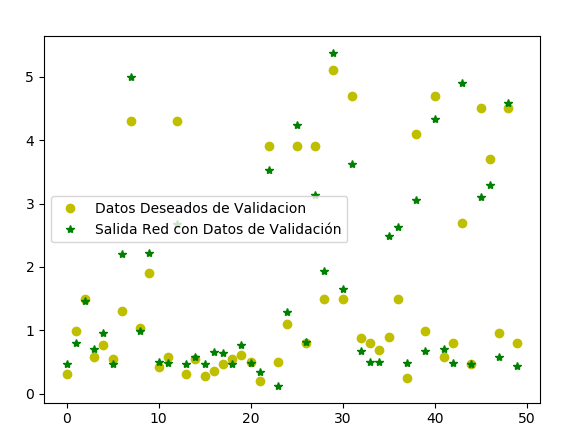


Fig. 10. Gráfica de los datos de validación deseados contra la salida de la red, con un loss de 0.39108586

**Prueba con el optimizador de Momentum, con un Alpha=0.01, un Momentum=0.9, 10 nodos en la capa oculta y 10000 épocas.**

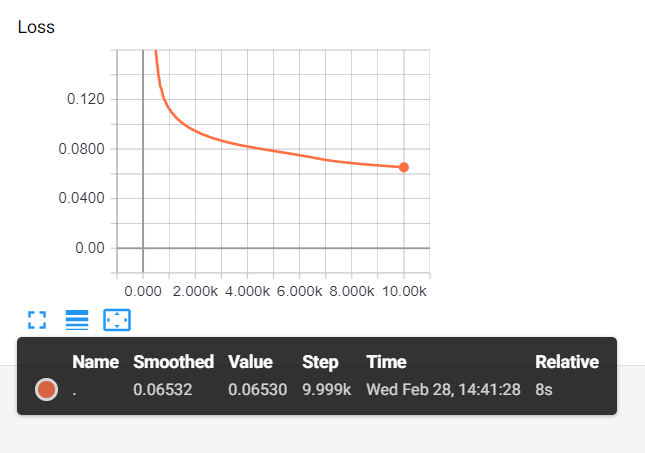


Fig. 11. Gráfica de la función de perdida de la red neuronal

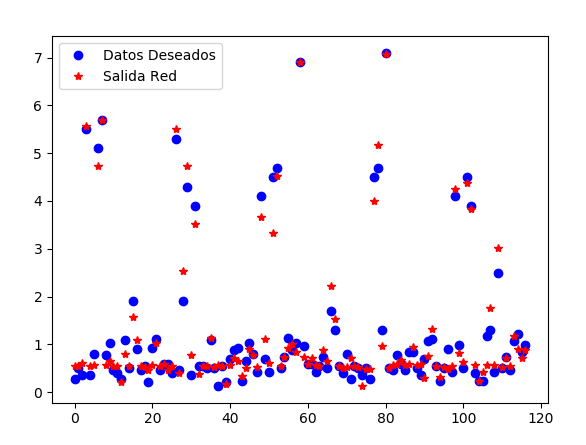


Fig. 12. Gráfica de los datos deseados contra la salida de la red, con un loss de 0.065301985

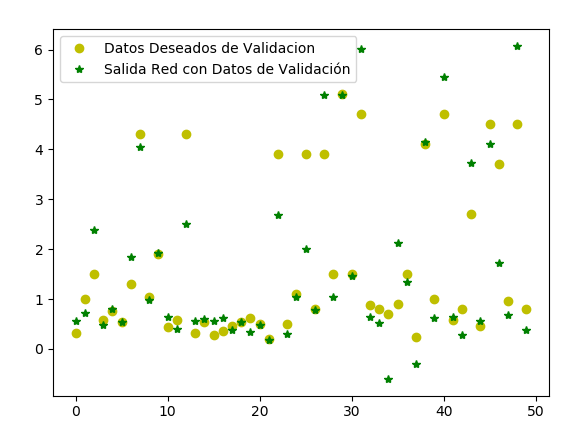


Fig. 13. Gráfica de los datos de validación deseados contra la salida de la red, con un loss de 0.5166969

**Prueba con el optimizador de Momentum, con un Alpha=0.01, un Momentum=0.9, 10 nodos en la capa oculta y 100000 épocas.**

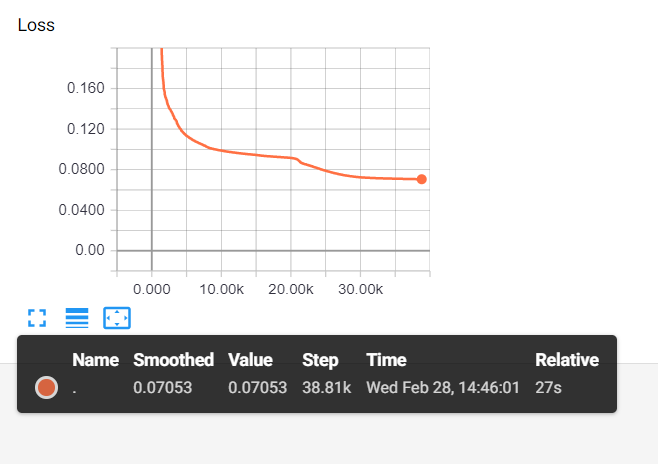


Fig. 14. Gráfica de la función de perdida de la red neuronal

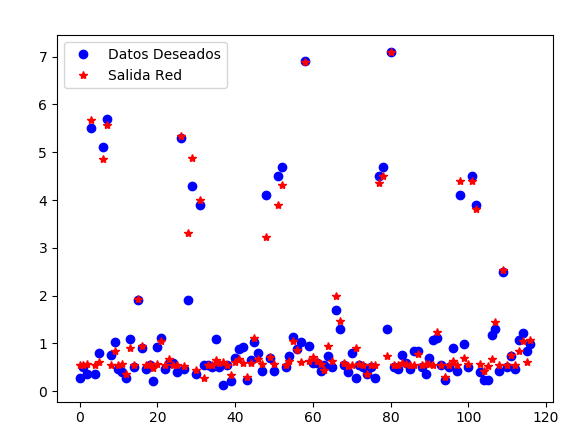


Fig. 15. Gráfica de los datos deseados contra la salida de la red, con un loss de 0.06739829

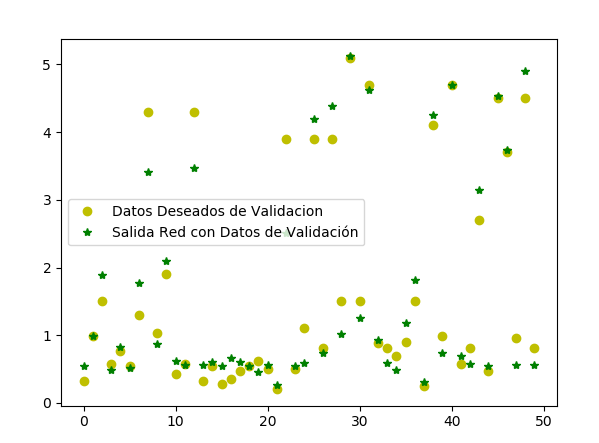


Fig. 16. Gráfica de los datos de validación deseados contra la salida de la red, con un loss de 0.12279792

**Prueba con el optimizador de Momentum Nesterov, con un Alpha=0.01, un Momentum=0.9, 10 nodos en la capa oculta y 10000 épocas.**

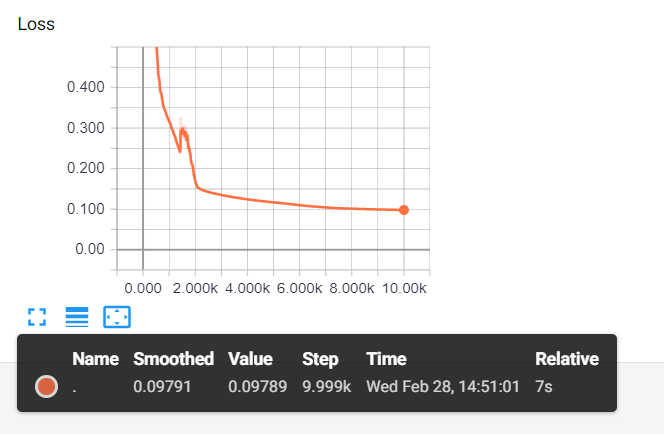


Fig. 17. Gráfica de la función de perdida de la red neuronal

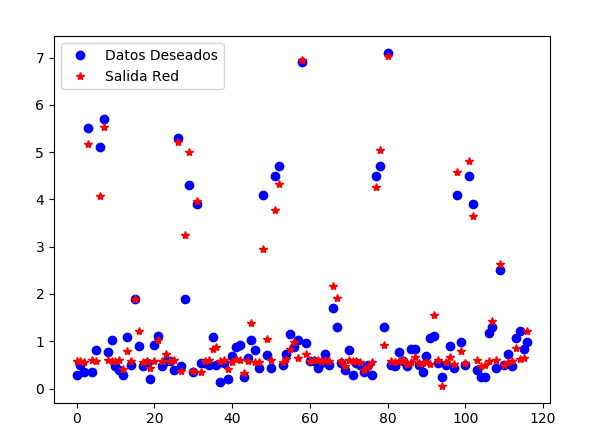


Fig. 18. Gráfica de los datos deseados contra la salida de la red, con un loss de 0.09789137

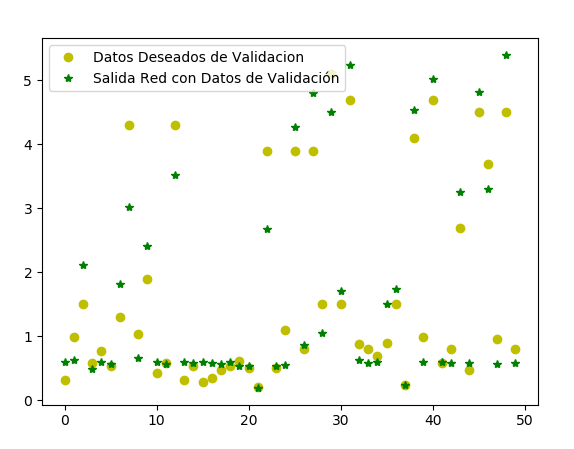


Fig. 19. Gráfica de los datos de validación deseados contra la salida de la red, con un loss de 0.2012761

**Prueba con el optimizador de Adagrad, con un Alpha=0.1, 10 nodos en la capa oculta y 10000 épocas.**

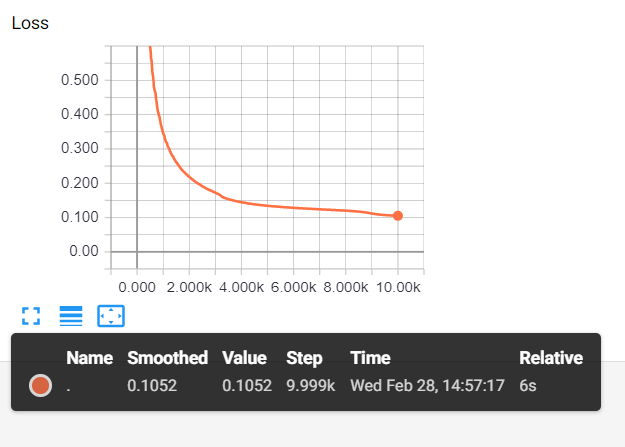


Fig. 20. Gráfica de la función de perdida de la red neuronal

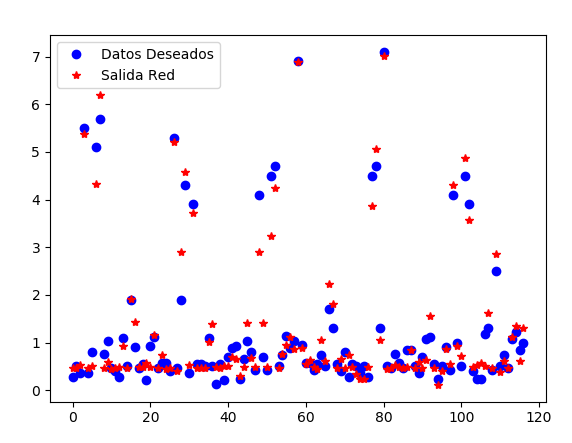


Fig. 21. Gráfica de los datos deseados contra la salida de la red, con un loss de 0.106791645

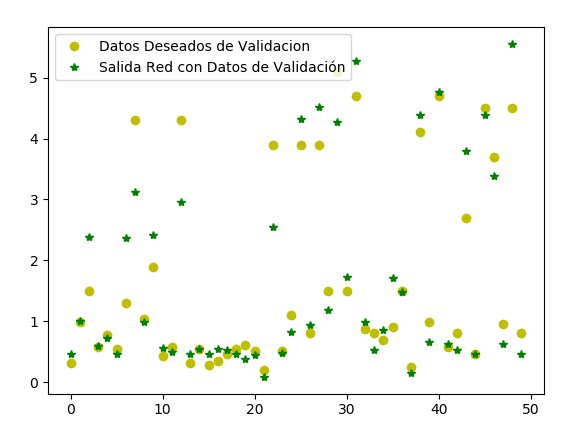


Fig. 22. Gráfica de los datos de validación deseados contra la salida de la red, con un loss de 0.2598414

**Prueba con el optimizador de Adagrad, con un Alpha=0.3, 10 nodos en la capa oculta y 10000 épocas.**

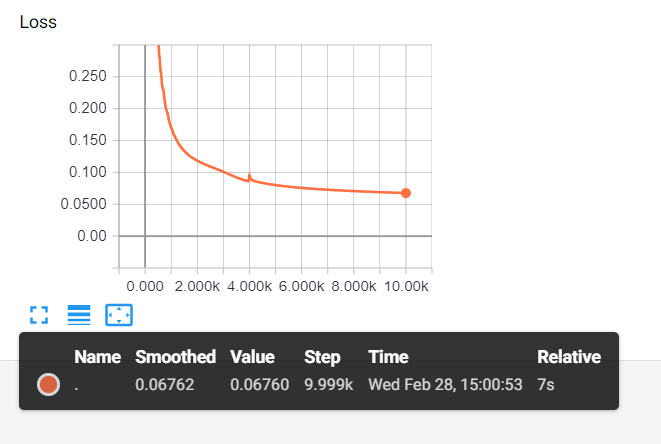


Fig. 23. Gráfica de la función de perdida de la red neuronal

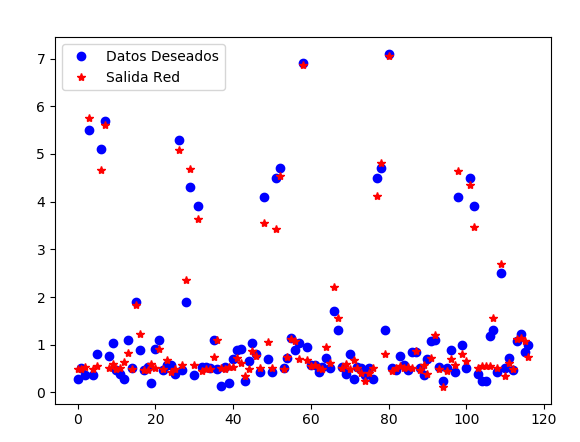


Fig. 24. Gráfica de los datos deseados contra la salida de la red, con un loss de 0.0675995

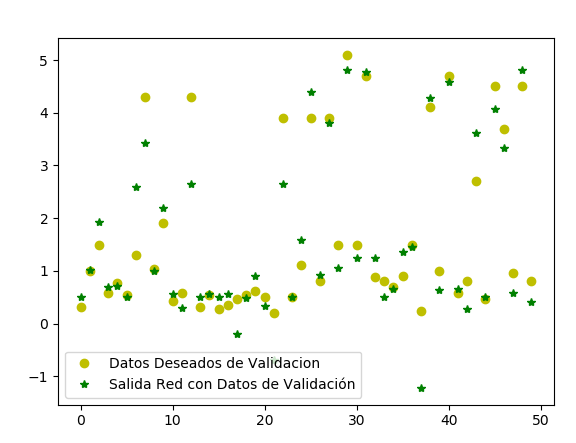


Fig. 22. Gráfica de los datos de validación deseados contra la salida de la red, con un loss de 0.2818412

**Prueba con el optimizador RMS, con un Alpha=0.01, un decay=0.9 10 nodos en la capa oculta y 10000 épocas.**

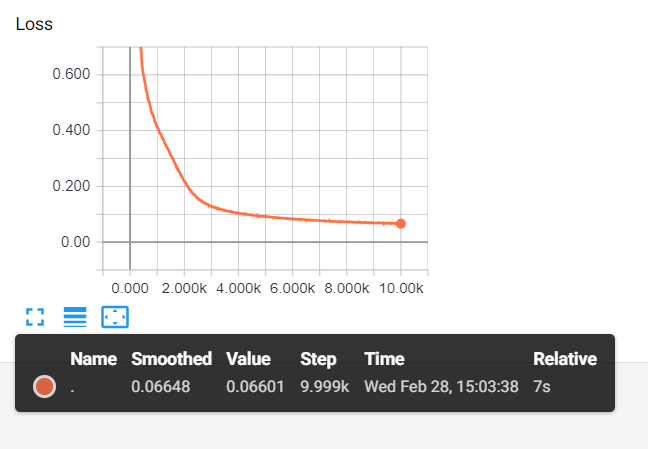


Fig. 26. Gráfica de la función de perdida de la red neuronal

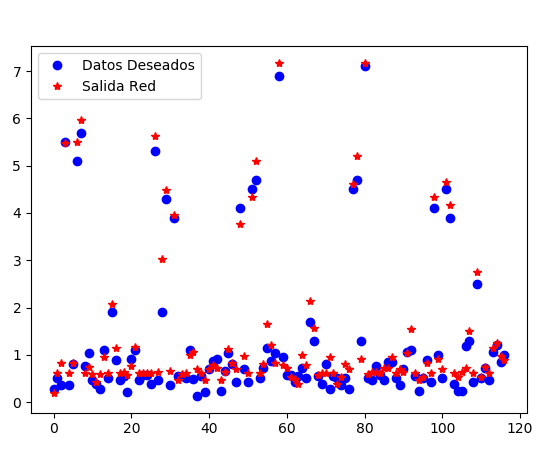


Fig. 27. Gráfica de los datos deseados contra la salida de la red, con un loss de 0.06600659

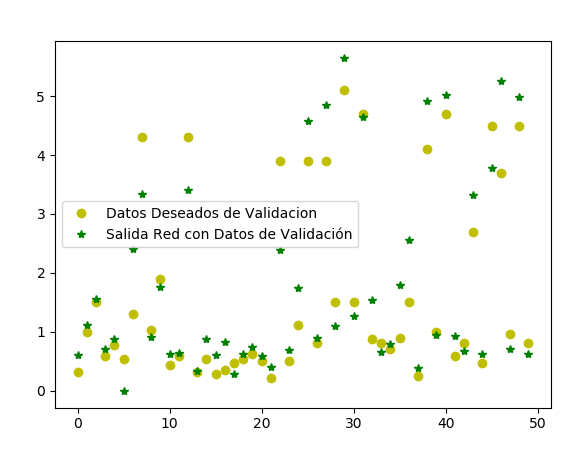


Fig. 28. Gráfica de los datos de validación deseados contra la salida de la red, con un loss de 0.31352174

**Prueba con el optimizador RMS, con un Alpha=0.01, un decay=0.5 10 nodos en la capa oculta y 10000 épocas.**

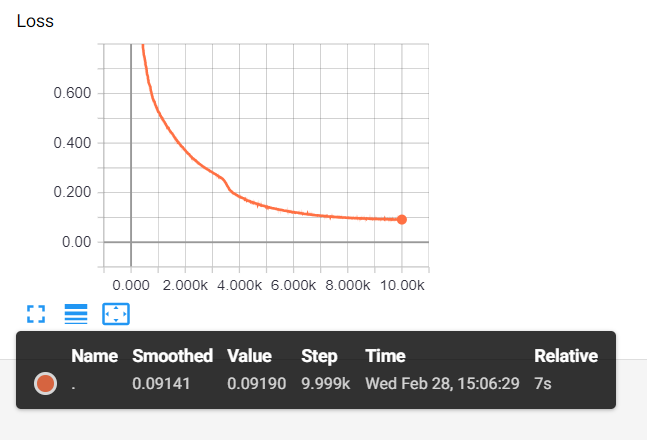


Fig. 29. Gráfica de la función de perdida de la red neuronal

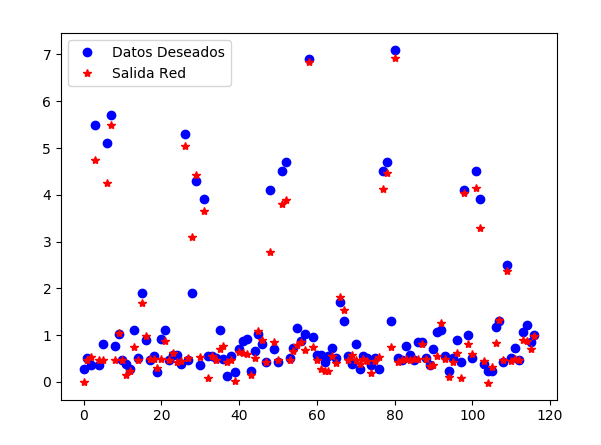


Fig. 30. Gráfica de los datos deseados contra la salida de la red, con un loss de 0.09189847

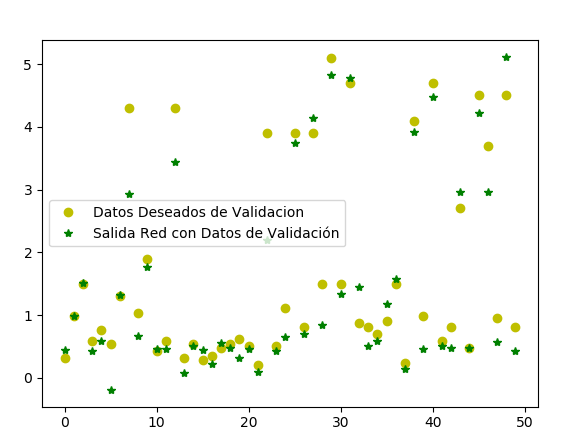


Fig. 31. Gráfica de los datos de validación deseados contra la salida de la red, con un loss de 0.19475706

A partir de estas primeras pruebas realizadas, se concluye que la red es capaz de aprender de manera aceptable la función, no obstante, se decide hacer una consulta para ver si es posible bajar más el loss de la validación.

**Jueves 1 de marzo de 2018**

Se hizo una reunión con el experto Jose Luis Paniagua, en la cual se consultó el por qué el loss en la validación era mucho mayor al loss en el entrenamiento, dándonos por respuesta que este comportamiento era normal debido a la no linealidad del sistema y que también podría ser un efecto de la dispersión de los datos de entrada. Nos da el consejo de esperar hasta la clase del viernes donde se enseñará a normalizar los datos de entrada para obtener mejores resultados en la red.

**Viernes 2 de marzo de 2018**

Se observa la clase del profesor Paniagua donde enseña a realizar la normalización de los datos de entrada, permitiendo a la red operar de mejor manera ya que los datos se encontrarán comprendidos en valores entre -1 y 1, disminuyendo altas variaciones en los pesos sinápticos y permitiendo un mejor aprendizaje de la red ante el problema de regresión propuesto.

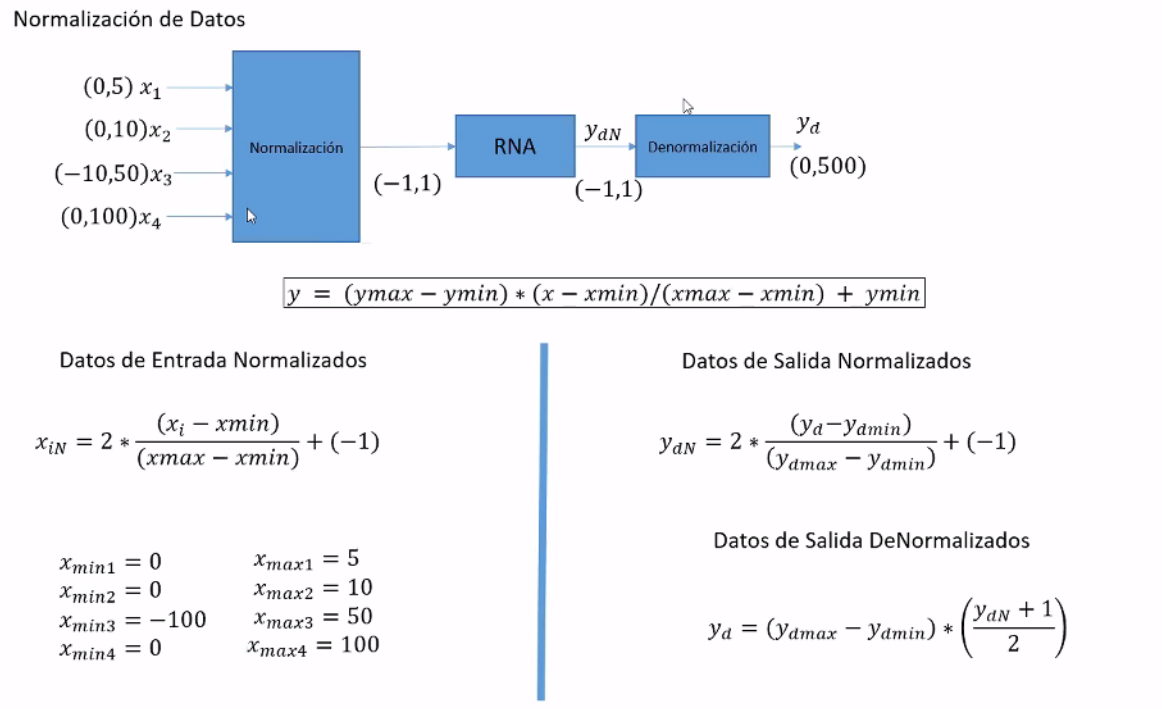


Fig. 32. Como normalizar datos, presentación realizada por Jose Luis Paniagua

Se realiza otro código igual al anterior, con la diferencia que ahora se normalizan los datos de entrada para que la red trabaje y una vez ha dado una salida, estos datos son desnormalizados para ver la respuesta final del sistema. Los resultados obtenidos en las pruebas fueron los siguientes.

**Prueba con el optimizador por Momentum, con un Alpha=0.01, un Momentum=0.9, 10 nodos en la capa oculta y 25000 épocas.**

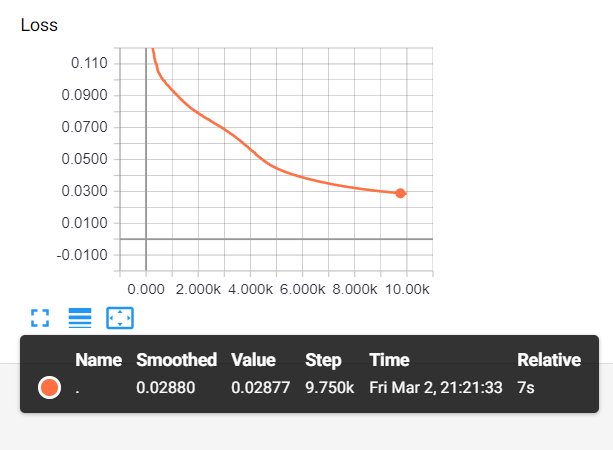


Fig. 33. Gráfica de la función de perdida de la red neuronal

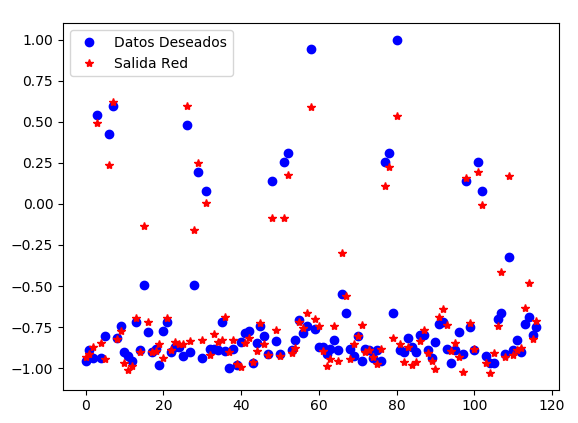


Fig. 34. Gráfica de los datos deseados contra la salida de la red, con un loss de 0.015535408

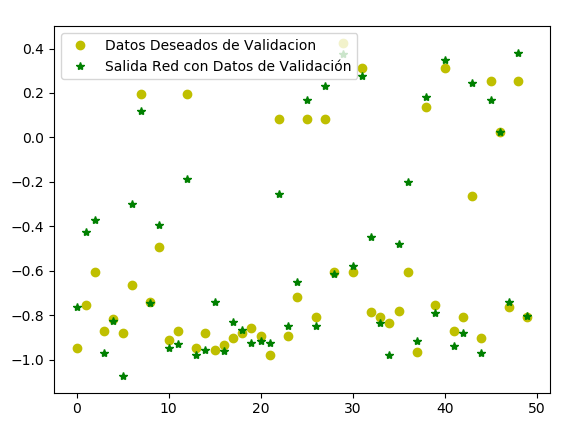


Fig. 35. Gráfica de los datos de validación deseados contra la salida de la red, con un loss de 0.029186273

**Prueba con el optimizador Adagrad, con un Alpha=0.1, 10 nodos en la capa oculta y 10000 épocas.**

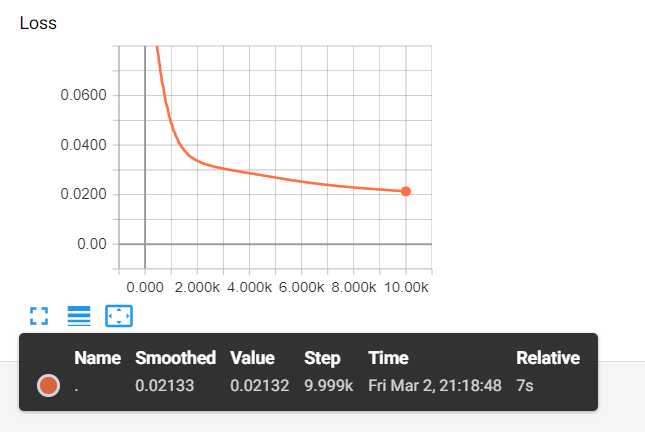


Fig. 36. Gráfica de la función de perdida de la red neuronal

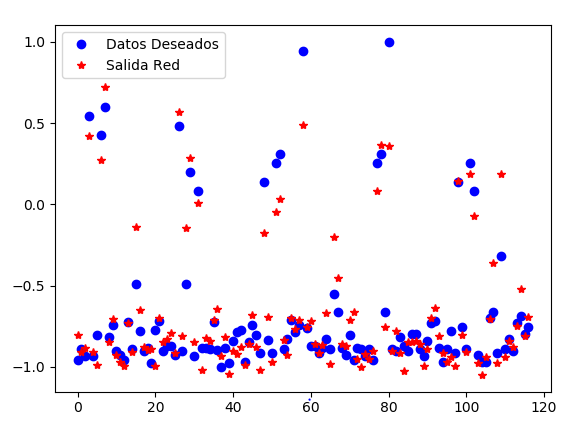


Fig. 37. Gráfica de los datos deseados contra la salida de la red, con un loss de 0.021320907

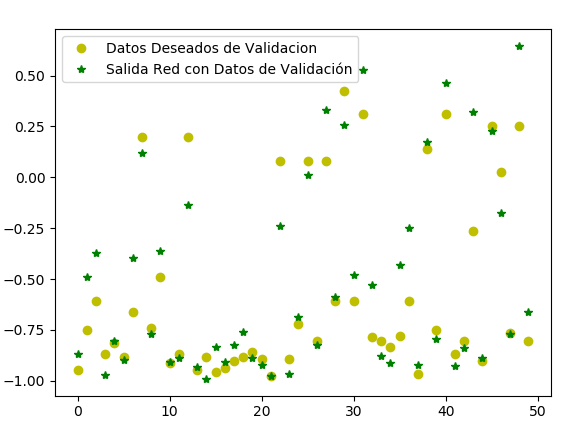


Fig. 38. Gráfica de los datos de validación deseados contra la salida de la red, con un loss de 0.03168855

**Prueba con el optimizador RMS, con un Alpha=0.001, un Decay=0.9, 10 nodos en la capa oculta y 10000 épocas.**

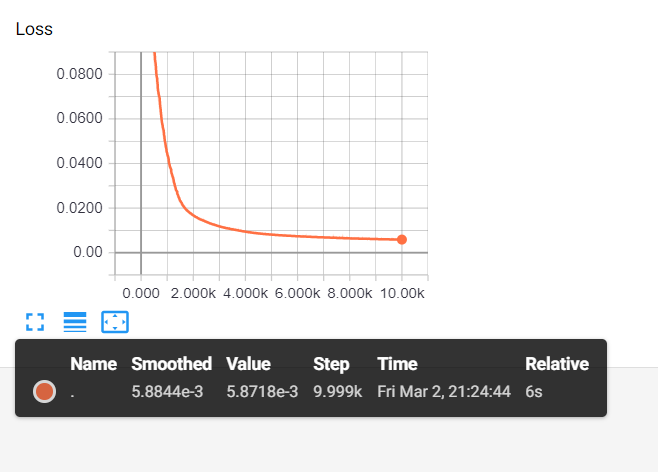


Fig. 39. Gráfica de la función de perdida de la red neuronal

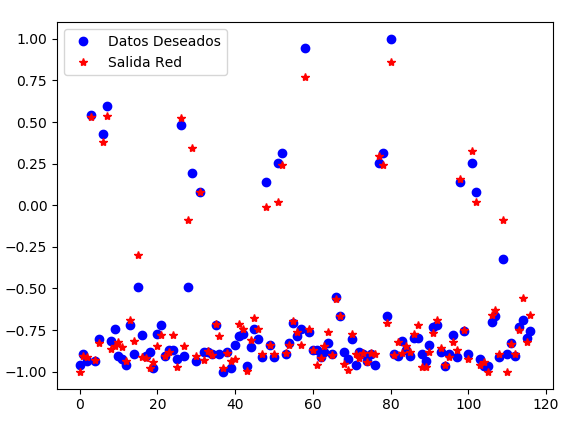


Fig. 40. Gráfica de los datos deseados contra la salida de la red, con un loss de 0.005871836

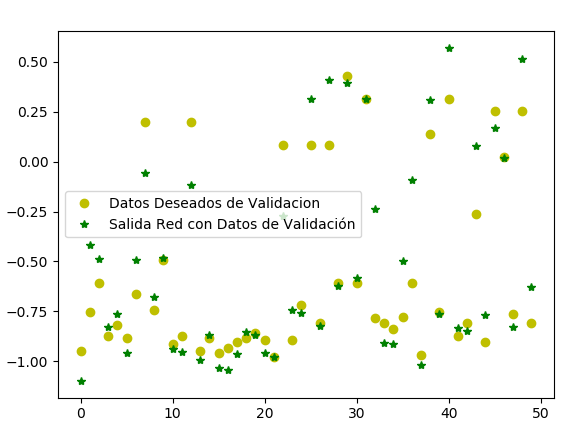


Fig. 41. Gráfica de los datos de validación deseados contra la salida de la red, con un loss de 0.034245055

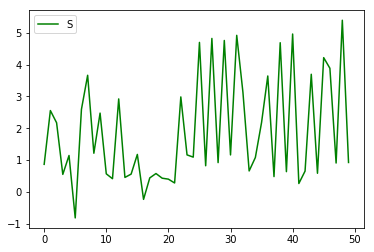


Fig. 42. Gráfica de los datos de validación desnormalizados

Concluidas estas pruebas, se puede observar que los resultados son mejores en términos de el Loss tanto en la fase de entrenamiento como en la fase de validación, pero se observó que a la hora de comparar la fidelidad con que la red seguía los datos de validación con los datos normalizados y los datos no normalizados, se obtienen mejores resultados con la red no normalizada a pesar de que el Loss de validación sea mucho mayor. En base a lo anterior se decide trabajar con la red no normalizada para realizar la validación de la red en Arduino.

No obstante, se decide realizar la implementación de la red normalizada en Arduino con ayuda de la extracción de los pesos realizados en los archivos .csv.

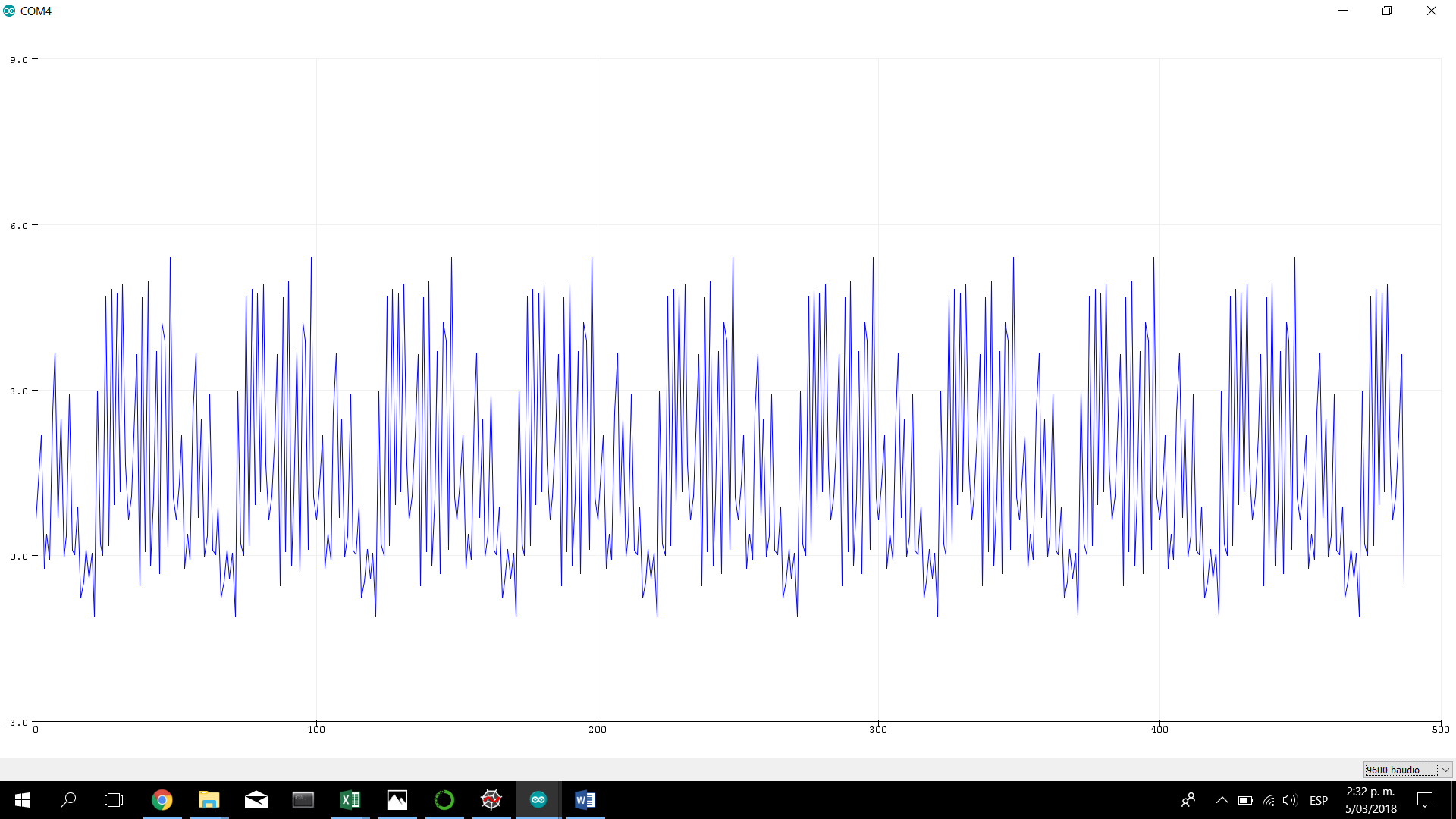


Fig. 43. Gráfica de la red neuronal validada en Arduino con ayuda del Serial Plotter

Se puede apreciar que los resultados obtenidos por el Serial Plotter de Arduino son muy similares a los de la Fig. 42, llevándonos a deducir que la implementación fue realizada de manera adecuada. Por otra parte, la respuesta oscila muy cerca a cero, lo cual no es cierto según los valores deseados de la red, se concluye que tiene algo de error esta prueba.

**Domingo 4 de marzo de 2018**

Finalmente, el grupo de trabajo decidió realizar la implementación de la red utilizando un optimizador por Momentum sin normalizar la red, el cual arrojó resultados muy satisfactorios en términos de error y la fidelidad con que la red seguía los datos de validación.

**Prueba con el optimizador por Momentum, con un Alpha=0.01, un Momentum=0.9, 10 nodos en la capa oculta y 25000 épocas.**

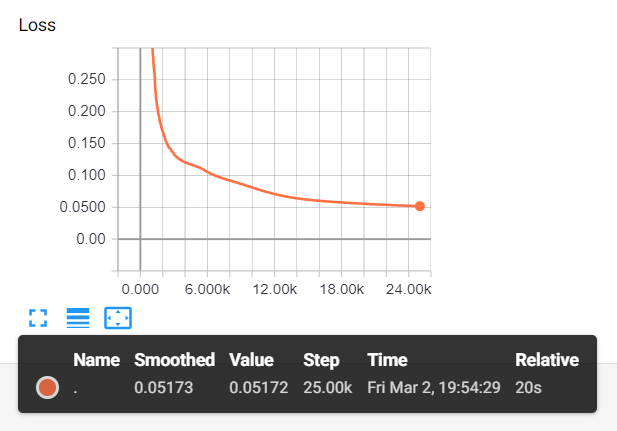


Fig. 44. Gráfica de la función de perdida de la red neuronal

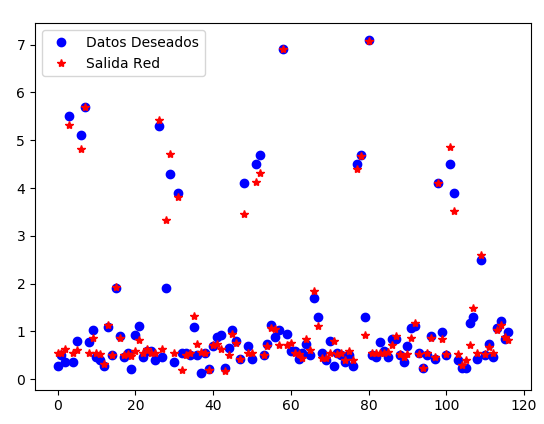


Fig. 45. Gráfica de los datos deseados contra la salida de la red, con un loss de 0.05171554

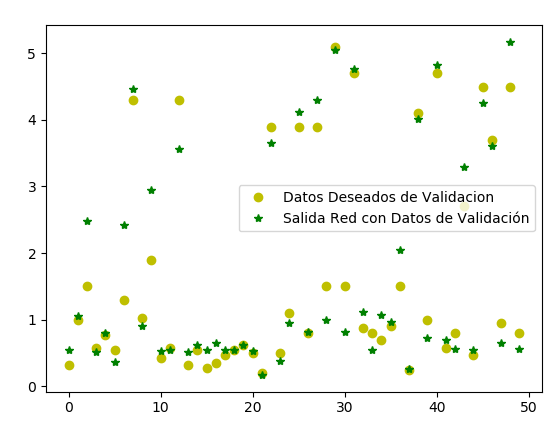


Fig. 46. Gráfica de los datos de validación deseados contra la salida de la red, con un loss de 0.1398349

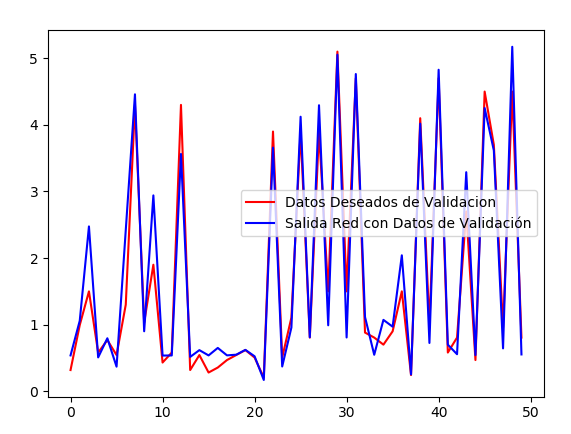


Fig. 47. Gráfica de los datos de validación deseados contra la salida de la red graficados de manera continua

Finalizada la etapa de simulación en tensorflow, se realiza la extracción de los pesos sinápticos de la capa oculta y la capa de salida para realizar su implementación y validación en Arduino.

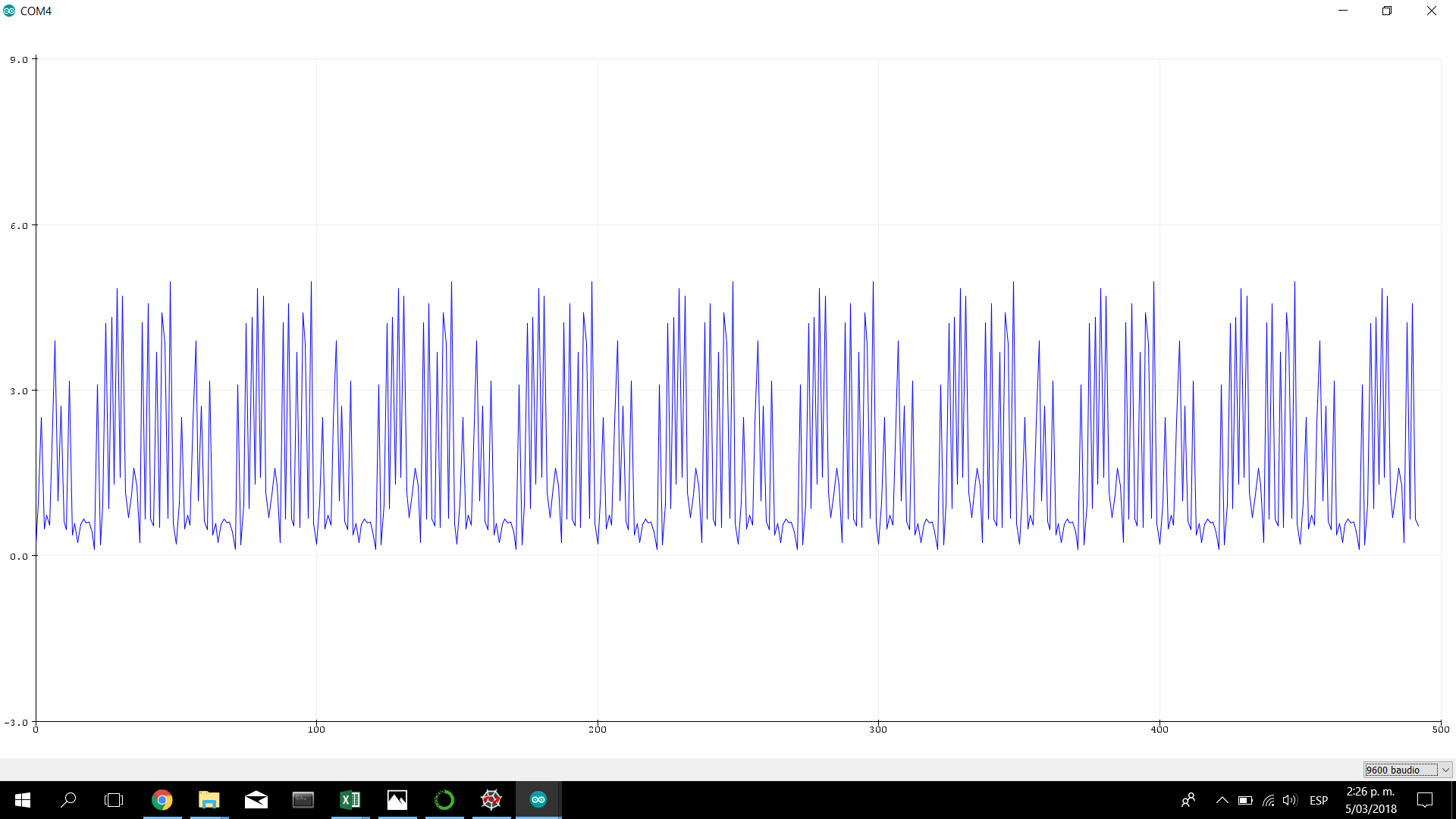


Fig. 48. Gráfica de la red neuronal validada en Arduino con ayuda del Serial Plotter

Se concluye que los datos son satisfactorios utilizando esta arquitectura de red, ya que Arduino es capaz de simular la red de manera adecuada, obteniendo un resultado muy similar al arrojado por Tensorflow.

**Lunes 5 de marzo de 2018**

Se realiza el informe del trabajo.