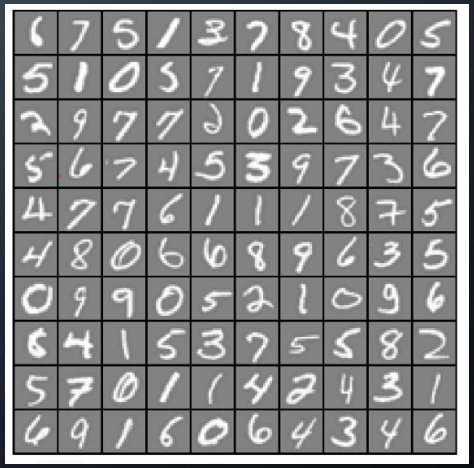
RECOGNIZE HANDWRITTEN DIGITS USING TRAINED MODEL.



OVERVIEW

- Using the STM32 NUCLEO-L4P5ZG and OLED screen to print numbers, We will compute the numbers 0-9 predictions using numbers images of 20x20 pixels and gradient descent values previously trained with the machine learning techniques specifically logistic regression with Multi-class Classification.
- The gradient (theta) optimal values was calculated with a functions in octave, Now our work is compute in the STM32 system the function h(z).

$$h_\theta(x) = g(\theta^T x)$$

$$z = \theta^T x$$

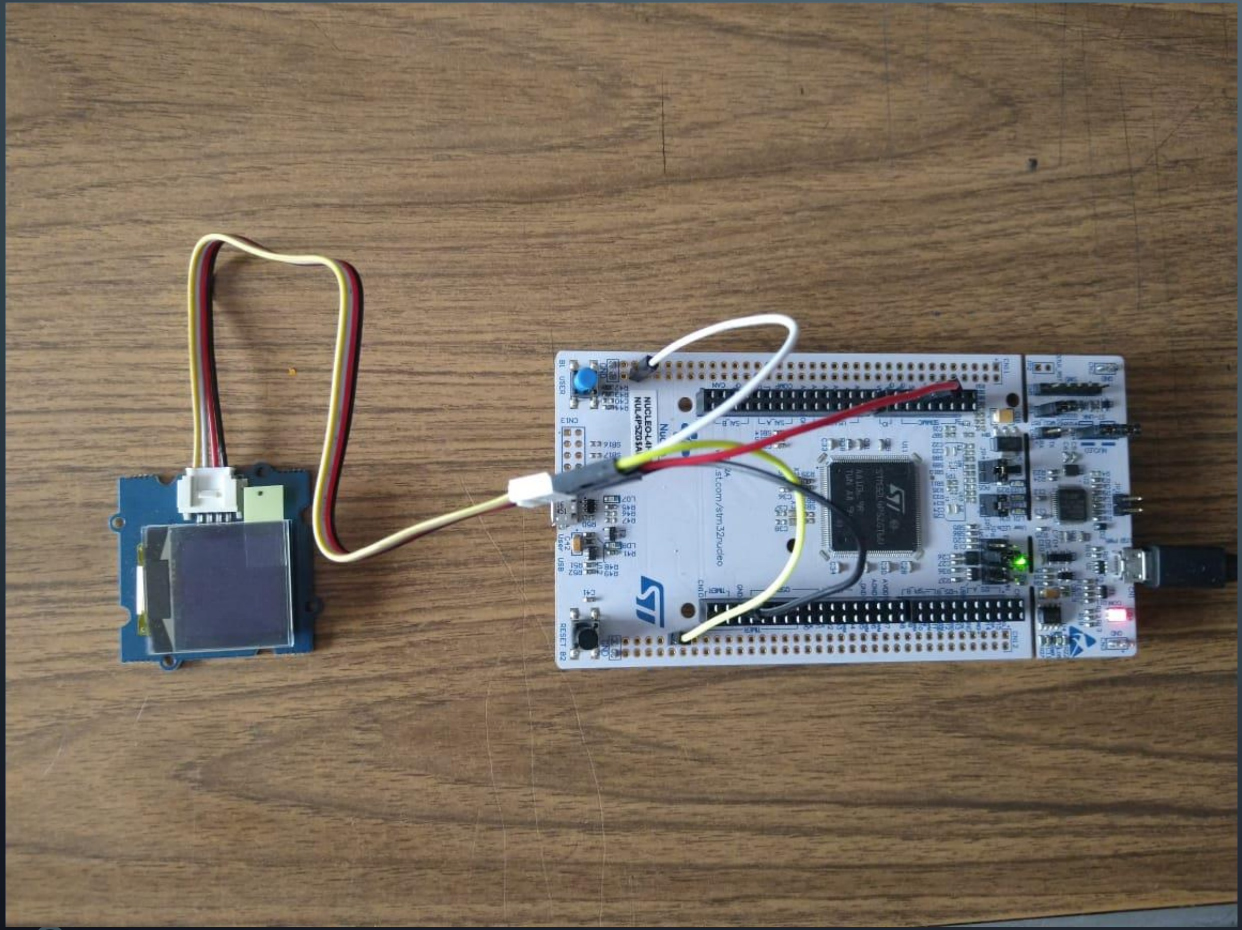
$$g(z) = \frac{1}{1+e^{-z}}$$

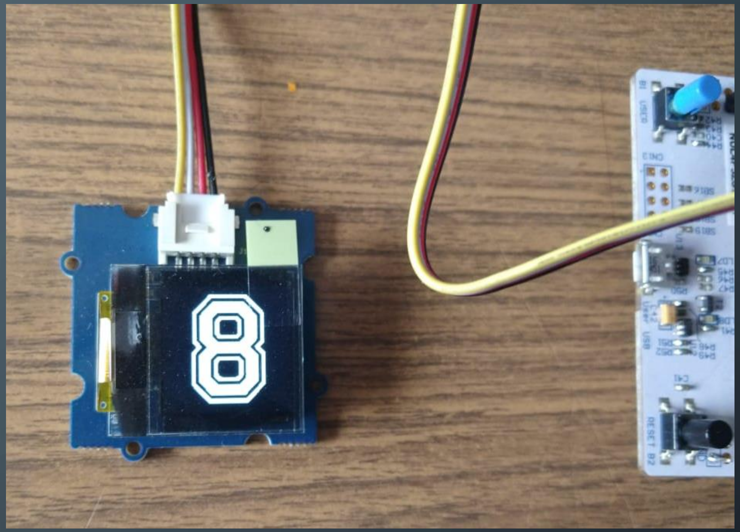
PROGRAM FUNCTIONALITY

- The main function of the Project is converter the data from octave in arrays(mxn) to compute array of predictions using operations like transposed of a matrix and product of matrix, after compute the sigmoid value of each element decide with other function which is the likelihood major to be 0-9 options for each example and save in an array of size m examples. Finally using the I2C protocol the prediction Will be printed in the OLED display and compared with the real digit value and check if the prediction was right. Finally we print a message to inform to the user if the prediction was correct or not.

LINKS TO SEE FUNCTIONALITY

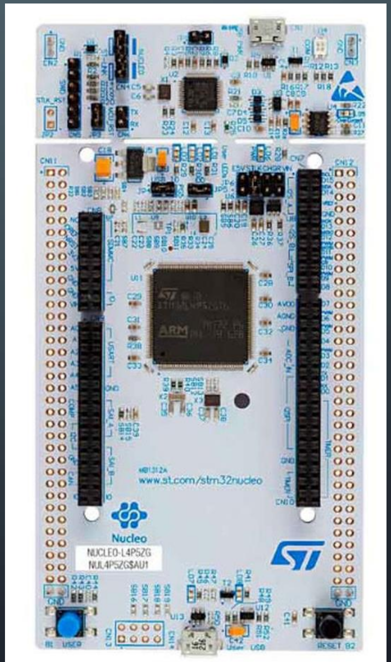
Link: https://www.youtube.com/watch?v=IHskfSiJGI4





TOOLS AND IDE

- IDE: STM32CubeIDE
- Board: STM32 Core L4P5ZG
- Grove Oled Display 1.12”
  - 128x128 dot Matrix
  - SH1107G driver IC
  - Interface I2C
  - Supply Voltage: 3.3V/5V



1.- MATHEMATICAL ANALYSIS

- In this part we will review the function that you can see in the figure 1.
- The result of compute this is a Matrix(arrays) with the likelihoods of the each case (0-9) for m number of examples, our work is decide which likelihood is major and after print in the OLED screen the number.

$$h_\theta(x) = g(\theta^T x)$$

$$z = \theta^T x$$

$$g(z) = \frac{1}{1+e^{-z}}$$

Figure 1.

h_θ will give us the **probability** that our output is 1. For example, $h_\theta(x) = 0.7$ gives us the probability of 70% that our output is 1.

$$h_\theta(x) = P(y=1|x;\theta) = 1 - P(y=0|x;\theta)$$
$$P(y=0|x;\theta) + P(y=1|x;\theta) = 1$$

MULTICLASS CLASSIFICATION: ONE-VS-ALL

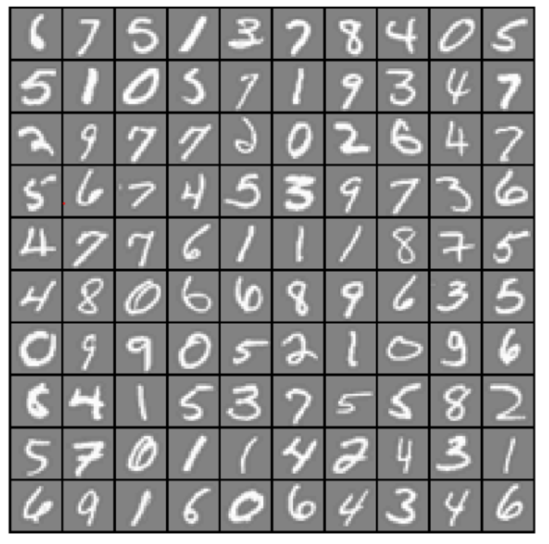
- Now we will approach the classification of data into more than two categories. Instead of y = {0,1} we will expand our definition so that y = {0,1…n}.
- In this case we divide our problem into n+1 (+1 because the index starts at 0) binary classification problems; in each one, we predict the probability that 'y' is a member of one of our classes.

$$
\begin{aligned}
&y \in \{0, 1 \ldots n\} \\
&h_\theta^{(0)}(x) = P(y = 0 | x; \theta) \\
&h_\theta^{(1)}(x) = P(y = 1 | x; \theta) \\
&\cdots \\
&h_\theta^{(n)}(x) = P(y = n | x; \theta) \\
&\mathrm{prediction} = \max_i (h_\theta^{(i)}(x))
\end{aligned}
$$

- We are basically choosing one class and then lumping all the others into a single second class. We do this repeatedly, applying binary logistic regression to each case, and then use the hypothesis that returned the highest value as our prediction.

DATA

- The data was obtained from file with 5000 examples of handwritten digits but we only take 30 and save in a data.h file to use in the project.
- We have the header data.h that contain the trained values of the logistic regression in the array theta, the other data is the data from 30 digits contained in X_test variable, we added other dataset of X called X_test2 to have different options.
- Finally we have the array y_test and y_test2 these arrays contain the real value of handwritten digit to compare the result of predictions with this data.



FILE DATA.H

```c
#ifndef _THETA_H_
#define _THETA_H_

/*Optimal gradient descent values to compute the predictions */
const float theta [] = {
-3.30005310017198,0,0,8.791529837476473e-05,-0.0008123901750976241,-0.0008853045953538524,0.001428618062894698,-0.01758020550147633,-0.001088875879211381,0.000560577825393251,0.0004734436940974181,6.951489091315579e-05,0.0004032822473572637,0.0017974316543
-3.253285444169741,0,0,-2.894181600355599e-05,0.0003974636171005364,-0.0009163074875424686,-0.0164035895152176,-0.009124738730518875,-0.01319638287008011,-0.001588142210849465,-0.008600708074360868,-0.003240929728402239,0.004022322574056749,0.0054557940467
-5.28132474801942,0,0,-1.006017876447379e-05,-0.0001762446501916476,0.004639586297640164,0.01908270287160198,-0.02654316814437686,-0.006956727578357109,-0.0001524731031678227,-0.01025351982956108,-0.003127013388772 97,-0.004426487819265862,-0.01623997049085
-2.386353876689085,0,0,-5.762916151460591e-06,-2.090611969670613e-05,0.001477890584540638,-0.0009269984678858945,0.001146754444491442,0.05649919670410364,0.04307423742900966,0.07303627010477248,0.02219692013034429,-0.003346071068902866,0.004960436630257668
0.6102923286889931,0,0,-9.053863111271268e-07,7.753976925201637e-05,-0.0008254286920365317,-0.007404711829827821,-0.003161178182219375,-0.01520530254262676,-0.009125745089175285,-0.001383449469909062,0.000106098745304292,0.001756308316441491,0.008307940457
-3.637958794074102,0,0,-1.334548565625279e-06,2.183731639789797e-05,-0.0001305406977533 57,-0.0002472104289291366,-0.001440489124299679,-0.001062064557689807,-0.0002033645419190502,-0.0004207795848396177,-0.0001407641320925945,8.151739422003263e-05,9.095770
-1.960668575786727,0,0,-2.323694490710678e-05,0.0001702943146758314,0.0005438375834083683,0.004080557905226817,0.02853201393543729,-0.0002316217866861026,-0.001236070342153798,-0.005104129987236462,-0.001899105378499407,5.827259080981301e-05,0.000190541390
-8.141079355019954,0,0,-4.721129787403584e-06,4.527134294076244e-05,8.143671377199165e-05,-0.0001771041924657688,-0.006970193959034875,-0.005993524236521598,0.0005137713152314442,7.448352622441044e-05,8.707552320612387e-06,4.29638055762685e-05,0.0001427102
-5.007798003690331,0,0,-1.71398203511519e-06,1.896698474127433e-05,0.0003460758387826118,-0.005225624565635333,-0.003663096685467552,-0.0006464449454267426,-0.01541316462138955,-0.008566181825803942,-0.0002407313243006554,0.0009938251922158 39,0.00154844019
-6.201348631592574,0,0,-4.727959088613198e-07,-5.500580044116281e-06,0.0001272370302800 26,0.0009267491458017927,4.451042788772974e-05,-0.0004943166776705338,-0.0008587679278671186,-0.001665285529364029,-0.0003380008312121332,0.000350956598796869 1,0.0005961
};

/*30 examples to compute and predict which number is */
const float X_test [] = {
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.01102941176474e-05,4.345383986928473e-05,-0.001581699346405286,-0.003153000285353337,-0.001655075571895485,0.00012
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.0001392463235294133,-0.002121783088235369,-0.005617749183006734,-0.004438725490196227,-0.0004851514410338088,0.0003
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4.503676470587928e-05,0.0001835171568627423,-0.01041023284313725,-0
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.0001455269607843233,-0.001572712418300691,-0.
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4.68750000000016e-05,-0.0003170445261438014,-0.001694546568627509,-0.0002977864742570724,3.88071895424851
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.633986928104636e-05,3.668827198240358e-06,-0.00143601919934647,-0.008205729166666884,-0.01964225898692844,-0.0354821282
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8.152949329420012e-06,0.0008420759307392706,-0.007719620072561339,-0.02466185642656253,-0.03350209938445264,-0.03770739064856744,-0.0218288862045743,-0.000330194
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.00105442551327352,-0.007784028372263707,-0.04415229709347395,-0.04018384900737877,-0.00271697036402925,0.0005174495574397831,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-0.00794667971
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5.259395424836796e-05,0.0001429313929313947,-0.006181832107843283,-0.02631709558823569,-0.01737351919934665,0.0007073631535947549,3
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4.595588235293962e-07,3.477328431372664e-05,-0.0003945567810457663,-0.0006505310457516555,6.420447596918853e-06,5.106
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.0002242061065590666,-0.0006948699618737477,-0.0
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.0001320777791366038,0.0004435204435204716,-0.01992091639150483,-0.02997383513465234,-0.002028046145693234,0.000234397293220824,0,0,0,0,0,0,0,0,0,0,0,0,
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7.826831356242657e-05,-0.0008699196934491173,-0.00937385349150064,-0.0007280583751171988,0.0001269215895607015,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3.574346405228891e-0
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.222942399412916e-05,0.0007150136561901322,-0.008981288981289052,-0.01079776609188381,0.001152057505243291,0,0,0,0,0,0,0,0,0,0,0,9.19117647058
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4.901960784313908e-05,0.0005442093677387889,-0.009144812091503479,-0.03570552491830115,-0.02560135825163434,-0.0005919117
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.0001247401247401258,-0.0003782968488850675,-0.00902409196526853,-0.03164944419683351,-0.0325677713913011,-0.005282295870531206,0.000821817292405535,0,0,0
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2.359569019275925e-18,0.0002731238025355716,-0.003809274886687166,-0.01981533569768882,-0.02580286168521484,-0.002244099302922839,0.0002311361134890466,0,0,0,0,0
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.0001792279411764727,-0.003247038398692819,-0.000532237200435729,3.880718954248165e-05,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-0.001663354531001606,0.03013467653173538,0.004939515307162349,-0.0003601565366271056,0
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7.745301862949011e-05,-9.923066448803112e-05,-0.01045411220043576,-0.02088048066448807,-0.02110038807189548,-0.01868723927547462,-0.004446827342047948,-0.0006678921568627461,0.0001102941176470591,0,0
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.011029411764741e-05,-6.505310457516457e-05,-0.000393739787581712,-0.001546512575924394,-0.0004398999183006679,-9.7528594771
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2.5
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5.106209150326462e-06,0.000246459694989107,-0.00228145424836601,-0.01491710920479308,-0.02137533461062879,-0.01935202205882358,-0.00411100898692811,0.0005882352941176484,0,0,0,0,0,0,0,0,0,0,0,0,0,0.000
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.043177882206199e-17,0.0006644653703477361,-0.01540754873272925,-0.04017936488524762,-0.05123884065060581,-0.04195874607639352,-0.02791732909379993,-0.00697214462
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.011128363317757e-19,2.859678422408842e-18,-4.729747367180836e-17,-5.095568423755518e-17,5.464265097270557e-18,0,0,0,0,0,0,0,0,0,0,0,0,-0.0002272125066242793,-0.006426035764271063,0.1062830194447845,0
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6.209392155989279e-18,3.95731
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.0006016876605111817,-0.008950715421303747,-0.02387387387387409,-0.04266764502058659,-0.03336695365556836,-0.02251151604092801,-0.007837022542904958,0.00065
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6.280637254902394e-06,1.628880718954313e-05,-0.001331444035947754,-0.002795343137255,-0.01025676694794364,-0.0238393075980396,-
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.0002371323529411763,-0.001382352941176469,-0.01109773284313725,
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.930147058823595e-05,-0.0001526245915032733,-0.0004271854575163545,-0.0006077004606416578,-0.0007963643790849968,-
};
```

SIZE OF DATA.

We have to compute the next function

$$h_\theta(x) = g(\theta^T x)$$

$$z = \theta^T x$$

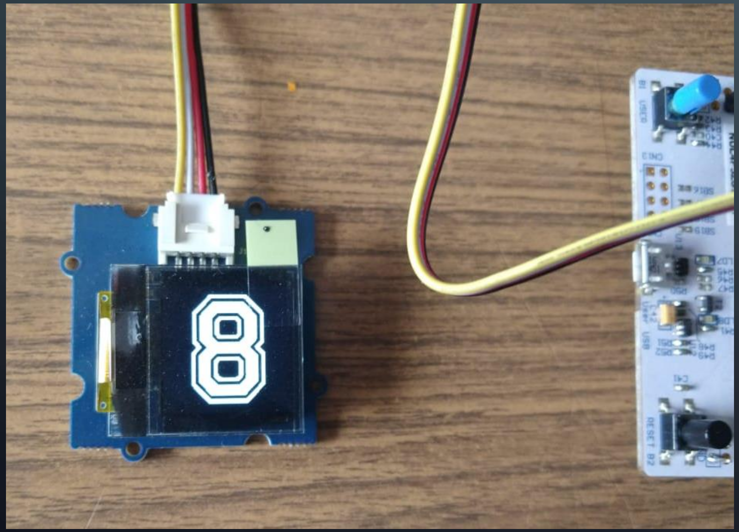
$$g(z) = \frac{1}{1 + e^{-z}}$$

Theta from data.h has 4010 elements of type float. We Will convert this data in a matrix with size 10x401

The array X_test has 12030 elements, to compute with theta we have to convert in a matrix with size 30x401.

After to calculate the theta transposed we have a matrix with size 401x10.

- To compute the product we have to compute X(30x401)*theta’(401x10) we will get a new matrix with size 30x10. after, we have to compute the sigmoid value of each matrix element.
- With this vector we will obtain the column number where is the largest value of each row (each row is a example) and we save the column number in a array p(prediction) finally this information is printed and compared with the real digit (y_test) and print a message to know if the prediction was correct or not.



SOURCES:

Machine Learning Course. Logistic Regresion https://www.coursera.org/learn/machine-learning