*********************Default*********************

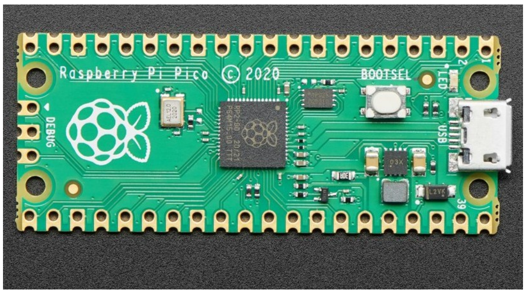
Raspberry Pi Pico RP2040 with TensorFLow Lite
Programming done on a Raspberry Pi4 and
Google Colaboratory
08/21/22

*********************Default*********************

Goals:

Goal 1.

To program a Raspberry Pi Pico RP2040 with TensorFLow Lite.



Additional information on process of compiling pico-tflmicro and testing can be found at

https://github.com/develone/my-projects-docs/blob/master/pico/tensorflow.txt

Steps to get the pico executable s
hello_world.elf, hello_world_test.elf & output_handler_test.elf

```
git clone git@github.com:develone/pico-tflmicro.git

cd pico-tflmicro

git clone git@github.com:develone/pico-sdk.git

cd pico-sdk/

git submodule update --init

cd ../

mkdir build

cd build

export PICO_SDK_PATH=../pico-sdk/

cmake -DPICO_BOARD=pico ..

make
```

Goal 2. To convert a TensorFLow model to a TensorFlow Lite model.

The command used to create model.cc was “xxd -i model.tflite > model.cc”

model.cc

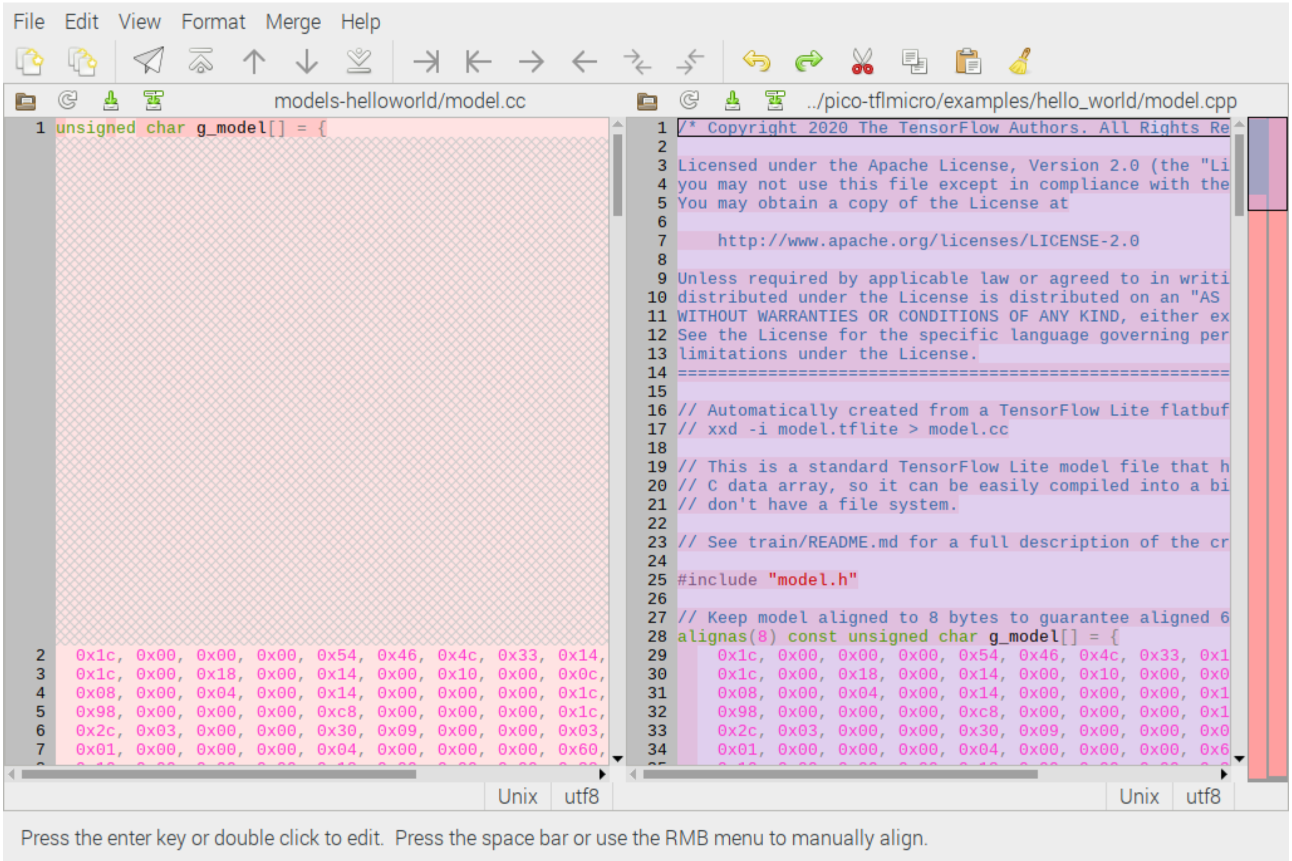
unsigned char g_model[] = {

model.cpp

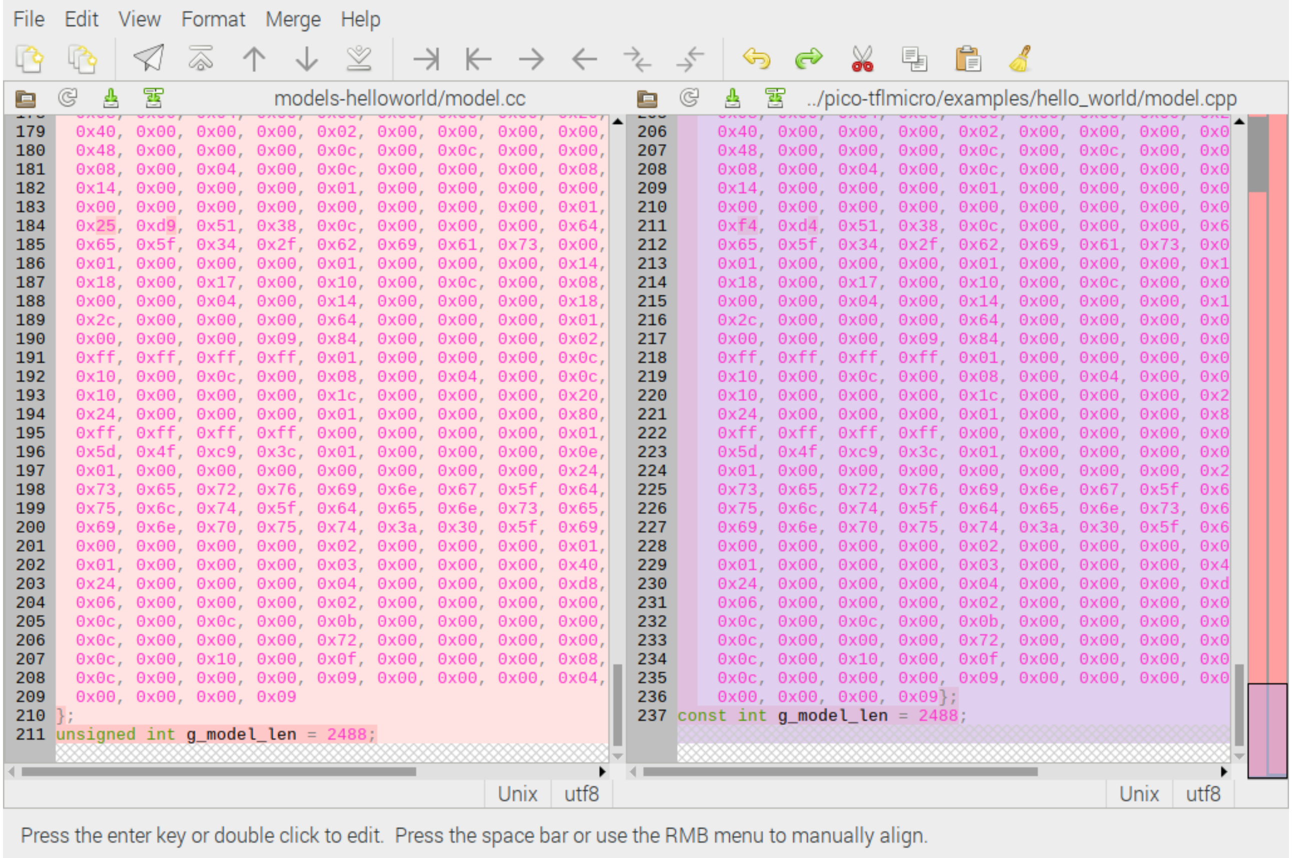
#include "model.h"

// Keep model aligned to 8 bytes to guarantee aligned 64-bit accesses.

alignas(8) const unsigned char g_model[] = {



Last of difference



Testing using the xxd command to create the model.cc

devel@pi4-27:~/xx/TF-CL/models-helloworld $ xxd -i model.tflite > tt.cc

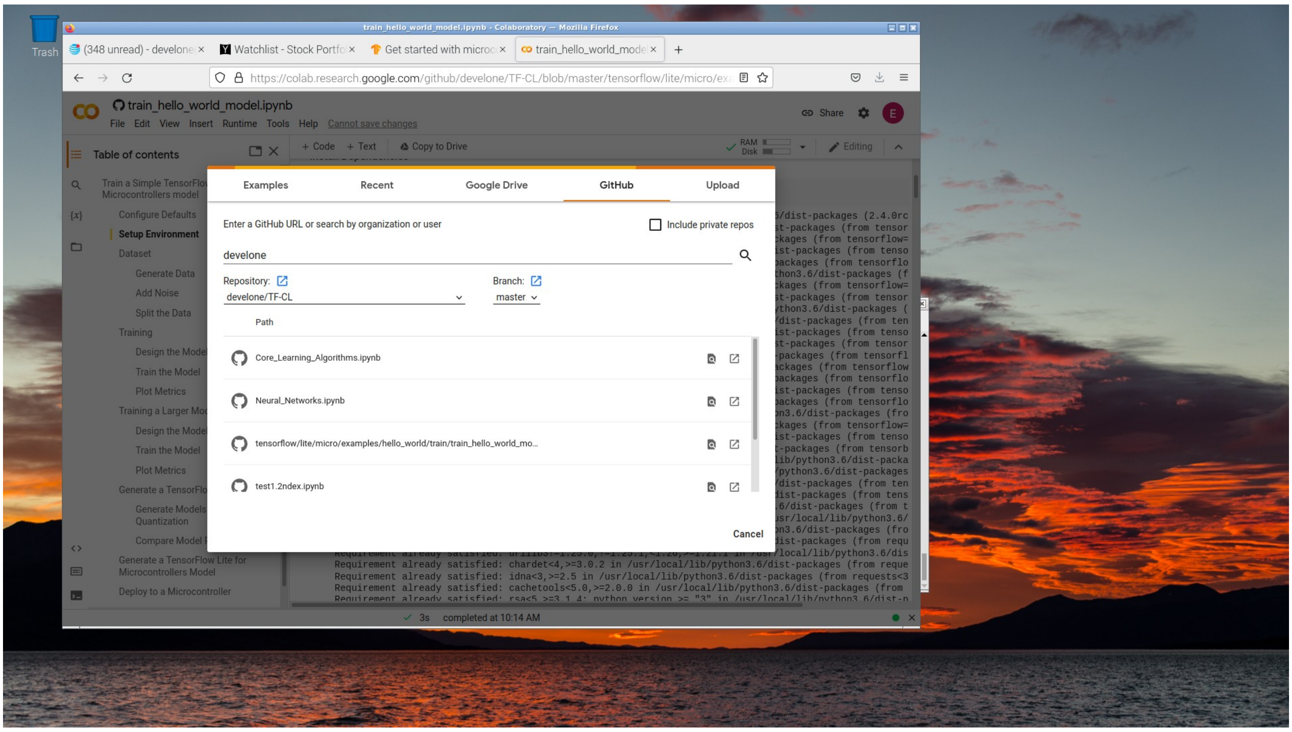
diff tt.cc model.cc
1c1
< unsigned char model_tflite[] = {

> unsigned char g_model[] = {
211c211
< unsigned int model_tflite_len = 2488;

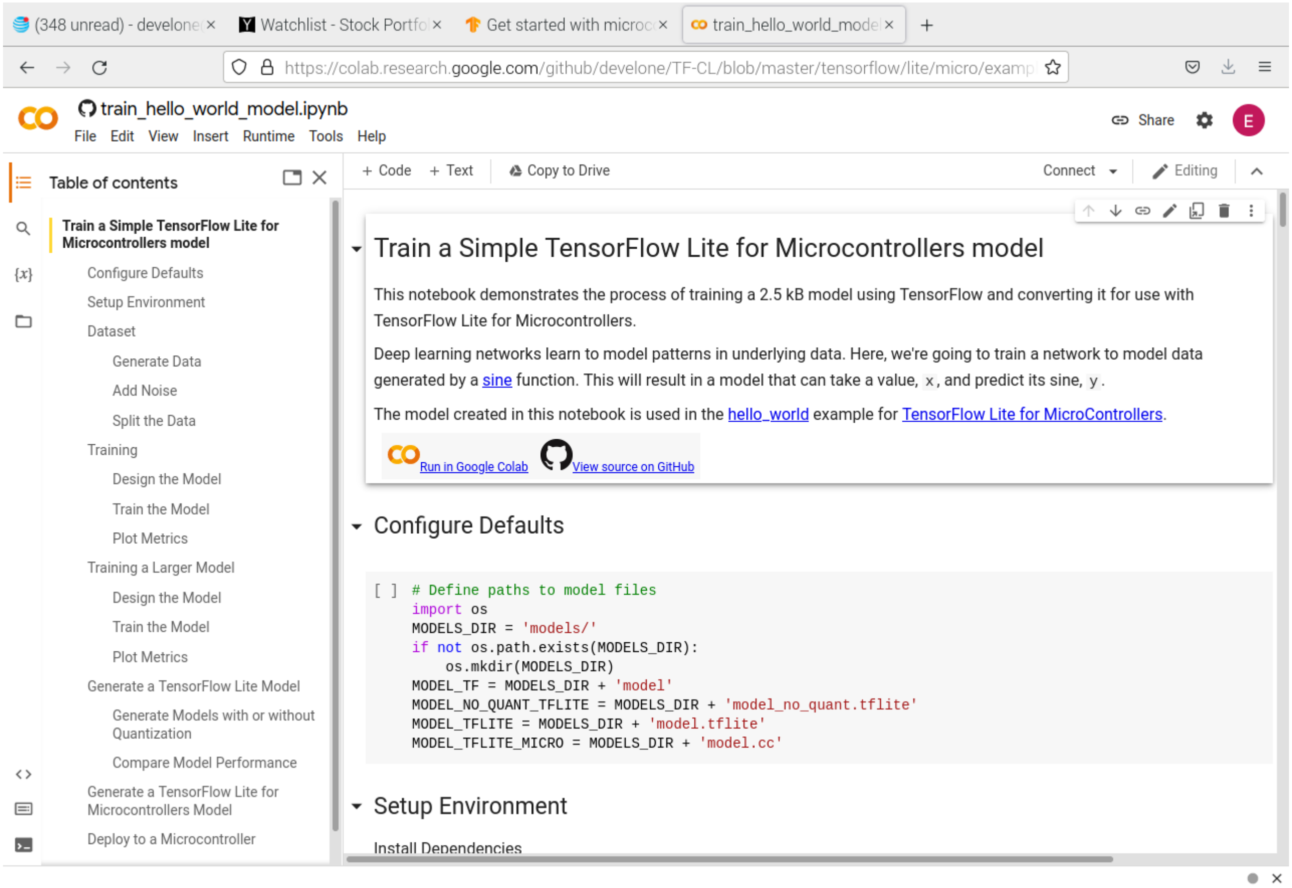
> unsigned int g_model_len = 2488;

The model was saved to my github from

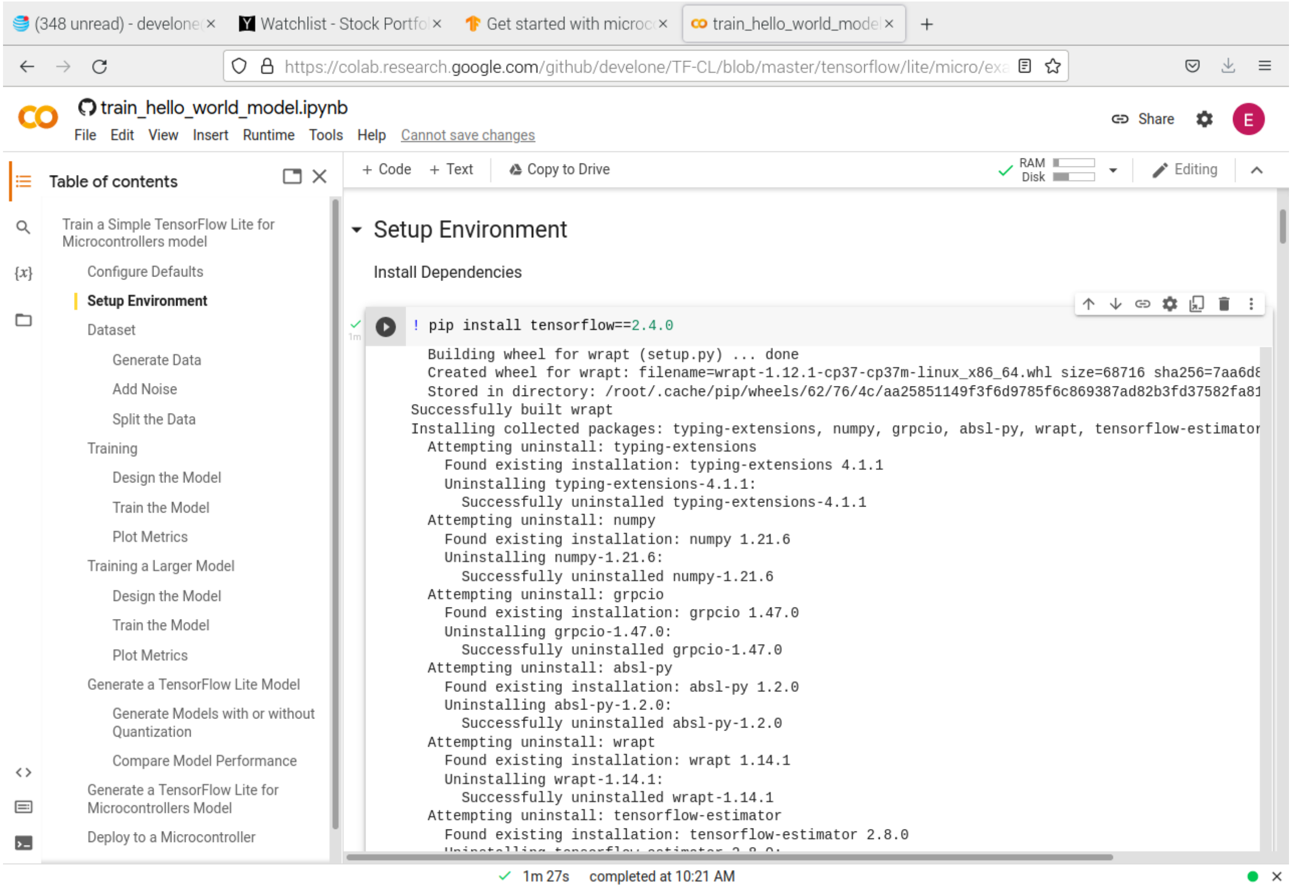
https://www.tensorflow.org/lite/microcontrollers/get_started_low_level



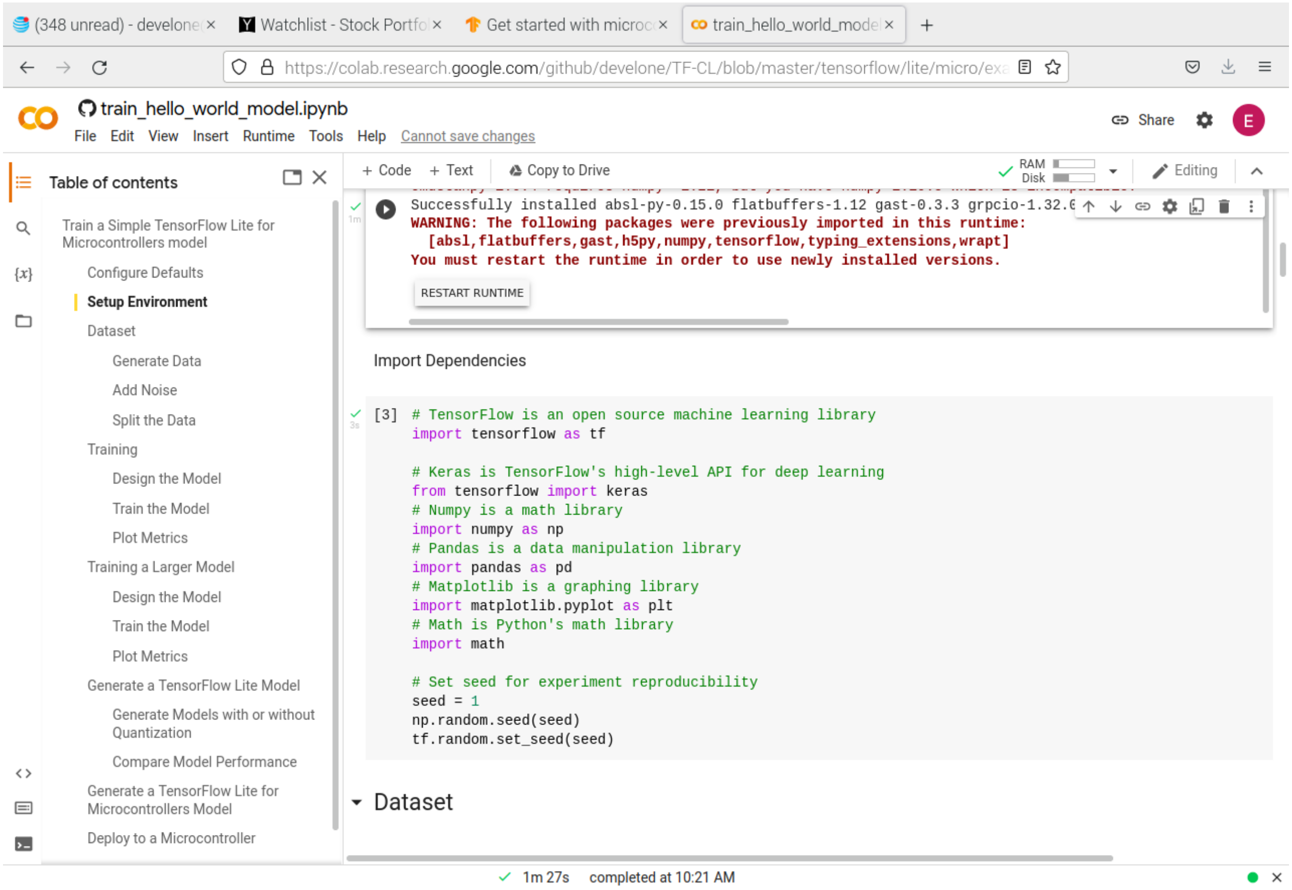
Loading the TensorFlow hello_world



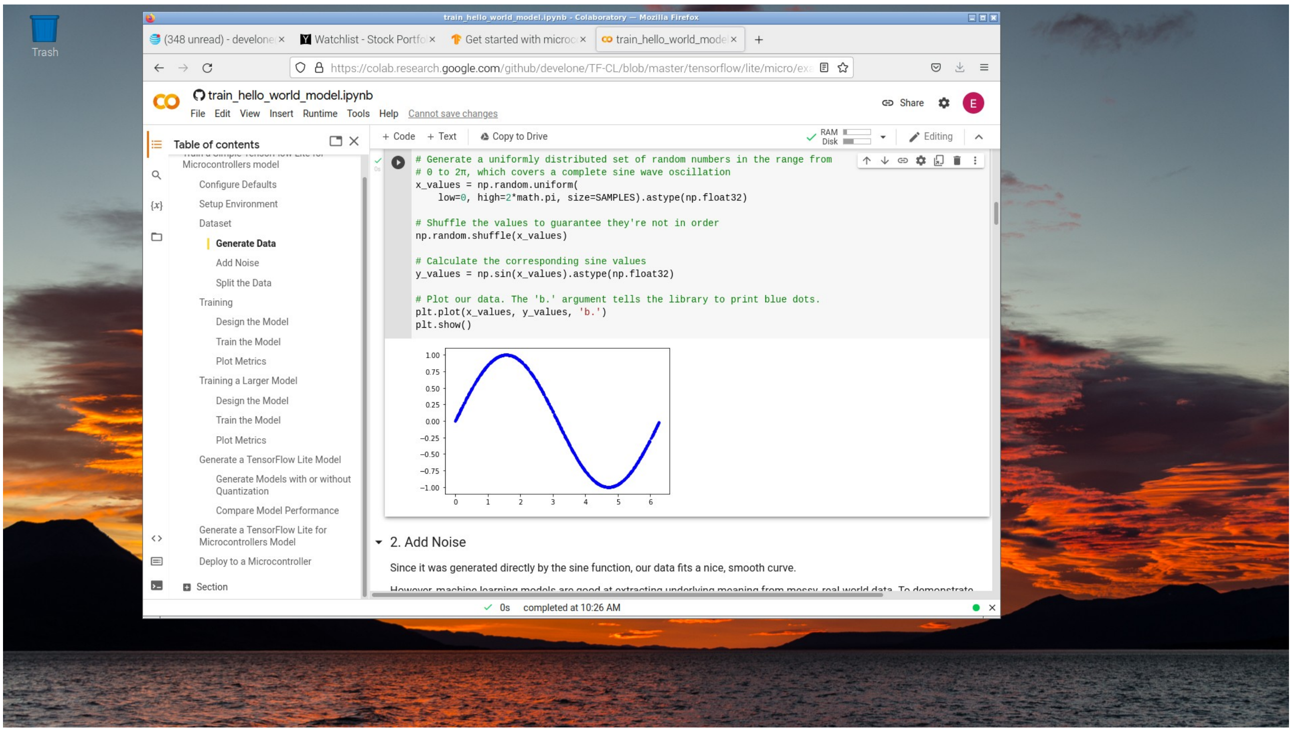
Setup Environment.



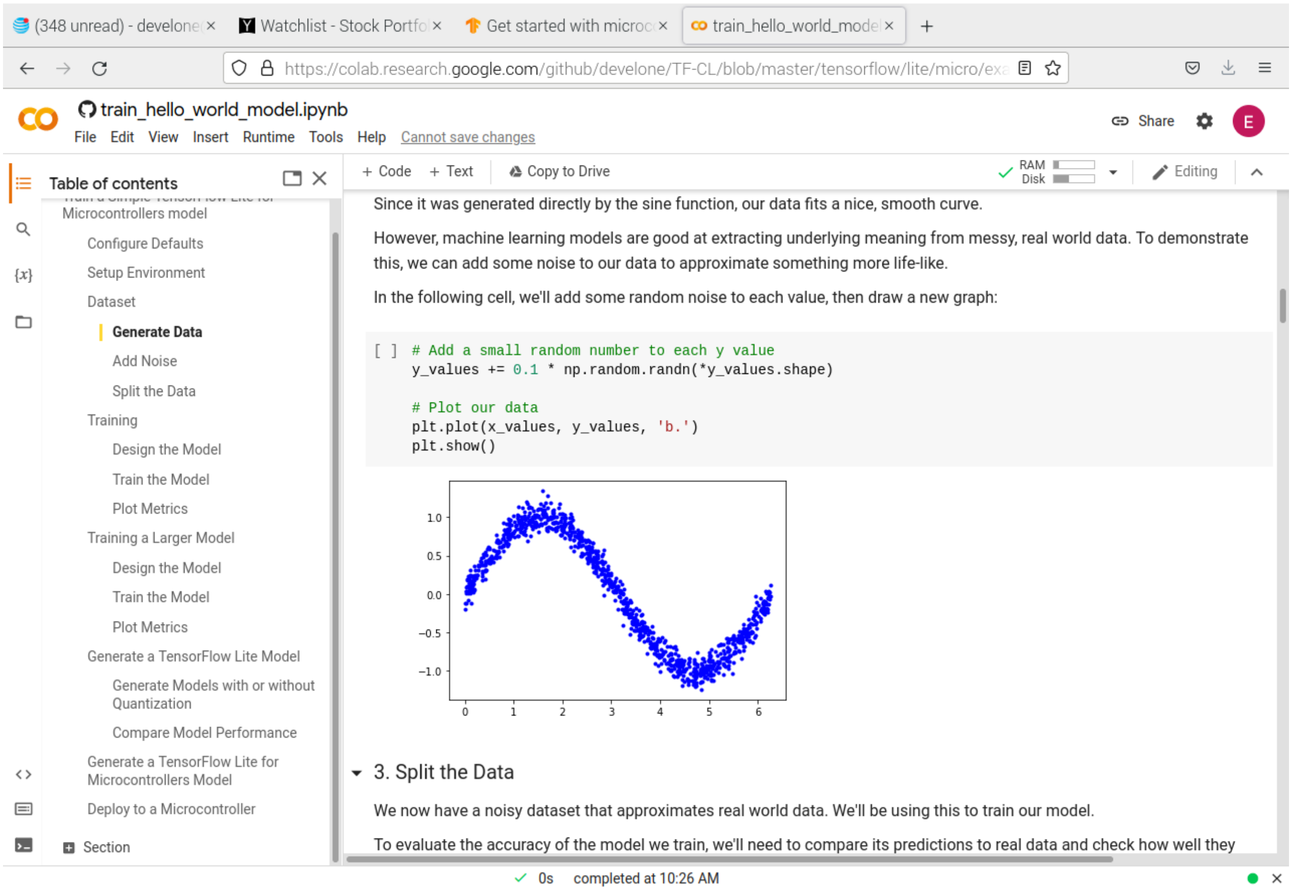
Import Dependencies



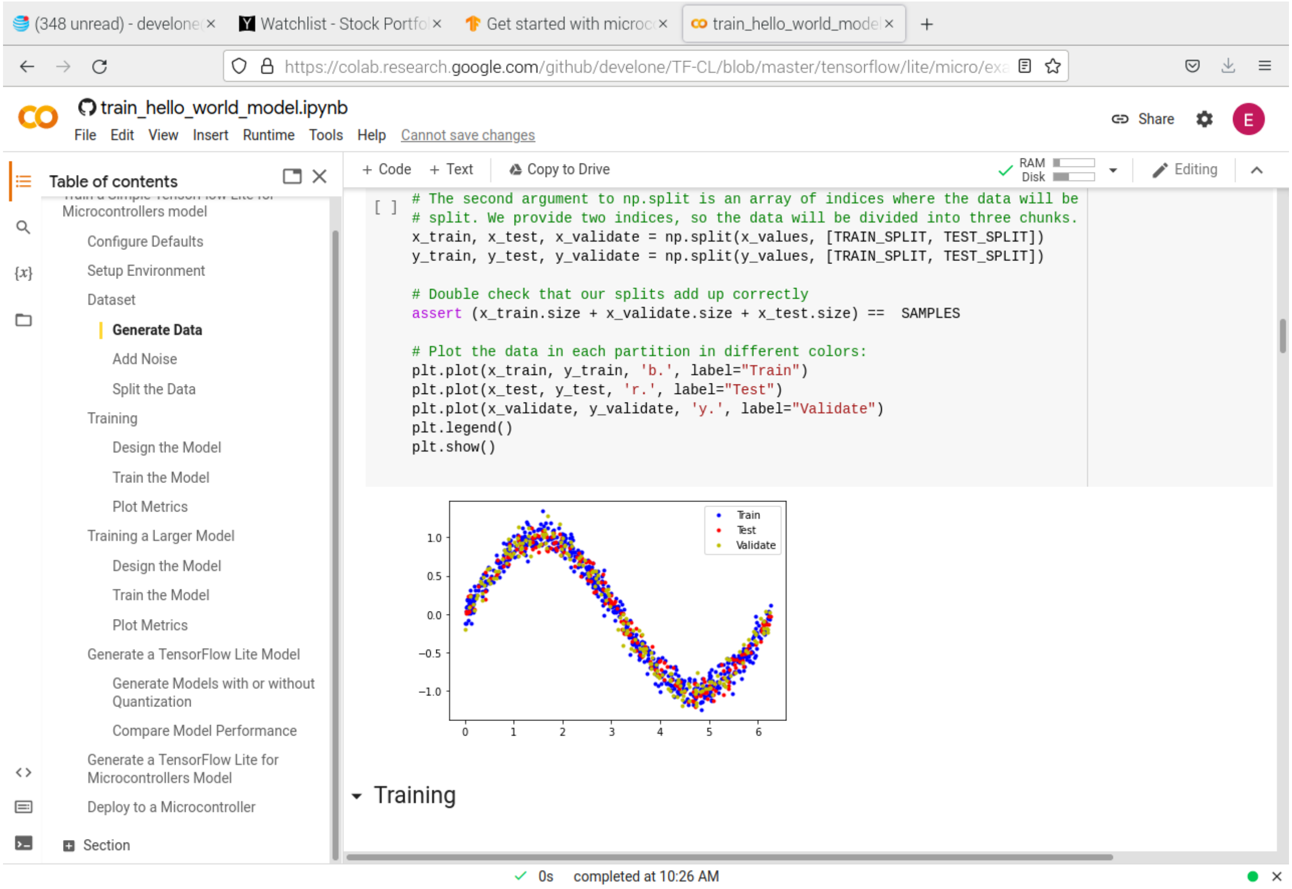
Generate Data



Add Noise



Split Data



Design the model

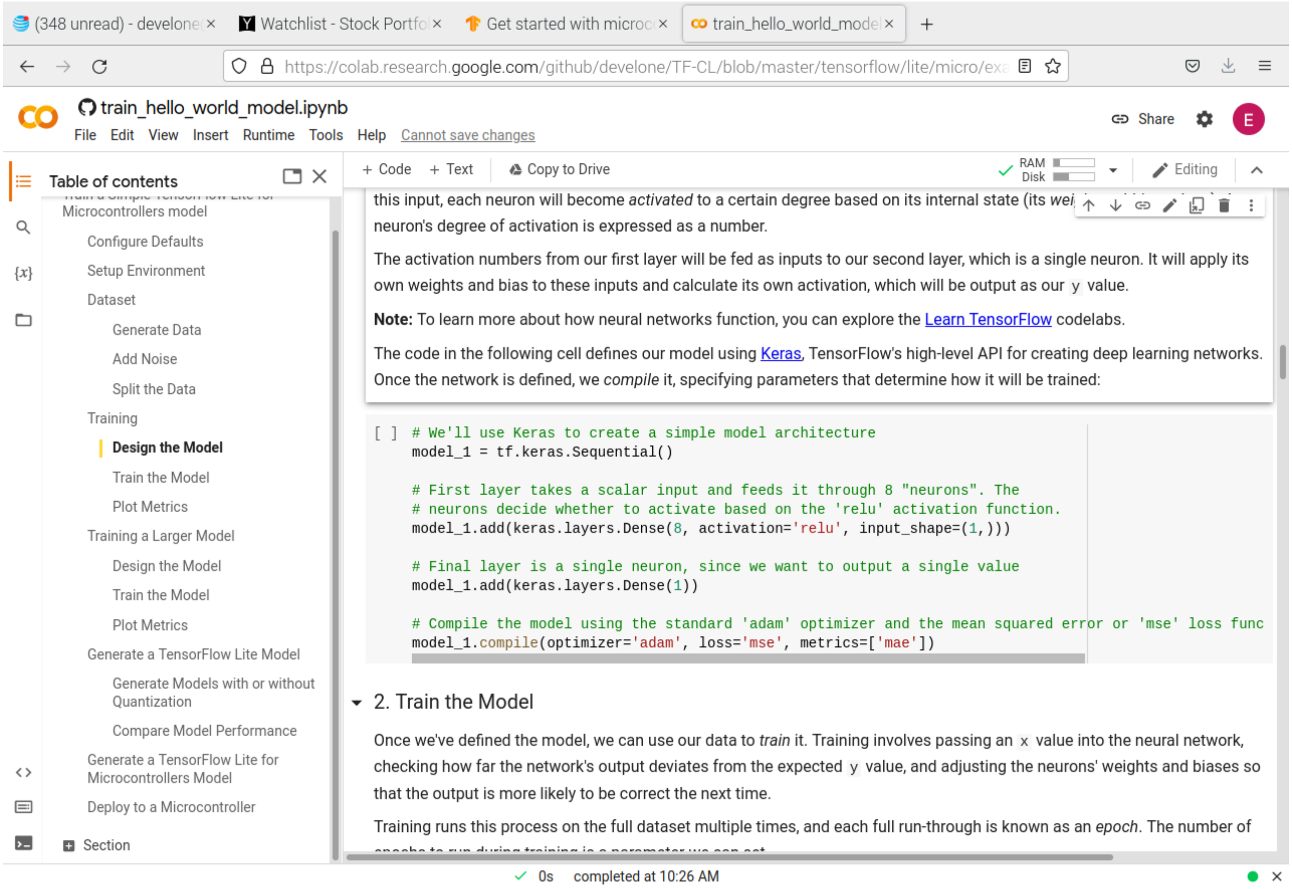
1. Design the Model
We're going to build a simple neural network model that will take an input value (in this case, `x`) and use it to predict a numeric output value (the sine of `x`). This type of problem is called a _regression_. It will use _layers_ of _neurons_ to attempt to learn any patterns underlying the training data, so it can make predictions.

To begin with, we'll define two layers. The first layer takes a single input (our `x` value) and runs it through 8 neurons. Based on this input, each neuron will become _activated_ to a certain degree based on its internal state (its _weight_ and _bias_ values). A neuron's degree of activation is expressed as a number.

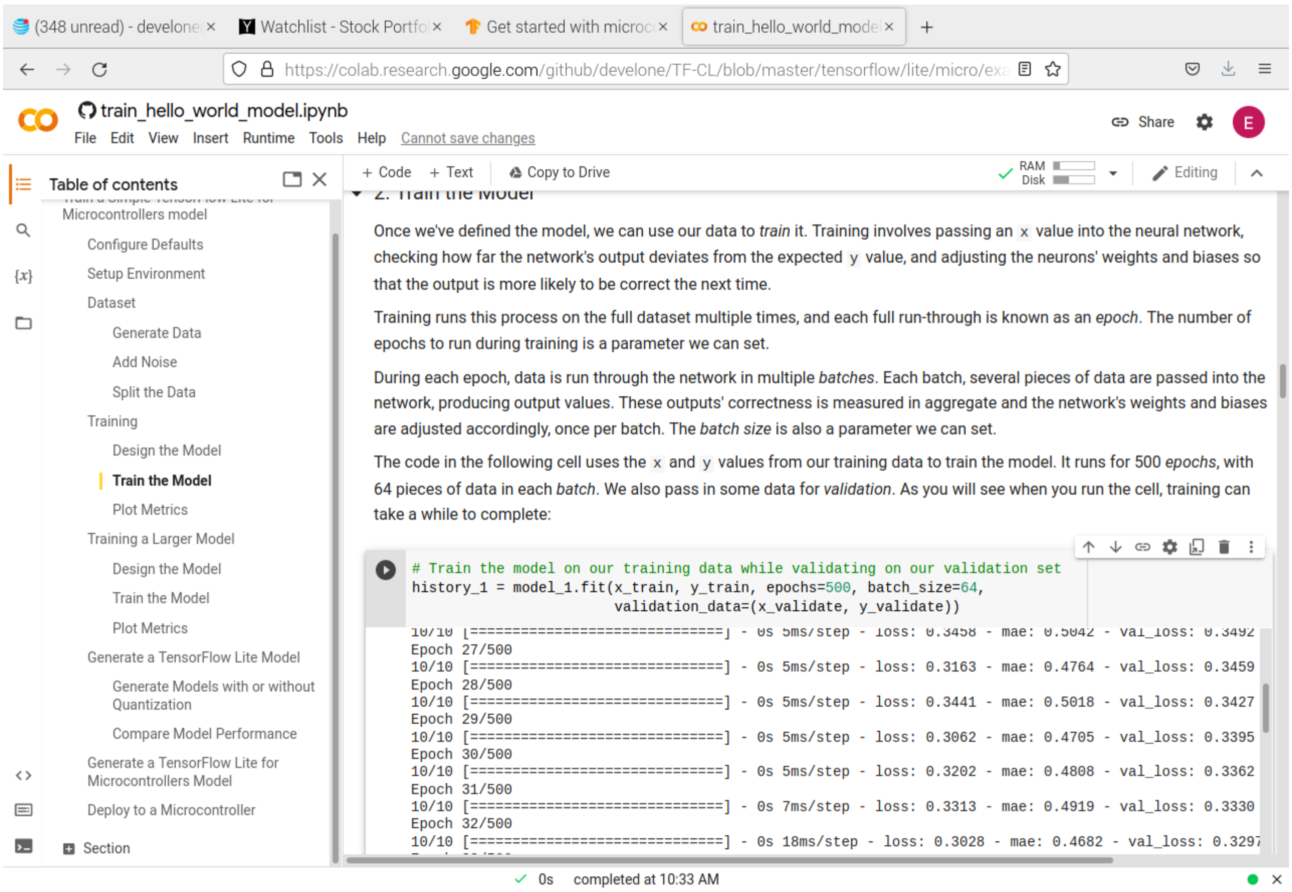
The activation numbers from our first layer will be fed as inputs to our second layer, which is a single neuron. It will apply its own weights and bias to these inputs and calculate its own activation, which will be output as our `y` value.

Note: To learn more about how neural networks function, you can explore the [Learn TensorFlow](https://codelabs.developers.google.com/codelabs/tensorflow-lab1-helloworld) codelabs.

The code in the following cell defines our model using [Keras](https://www.tensorflow.org/guide/keras), TensorFlow's high-level API for creating deep learning networks. Once the network is defined, we _compile_ it, specifying parameters that determine how it will be trained:



Train the model

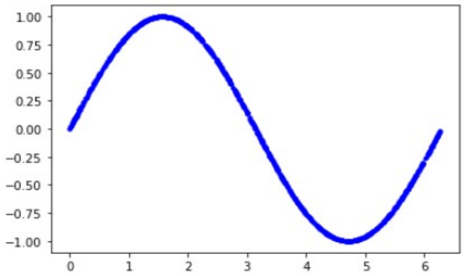


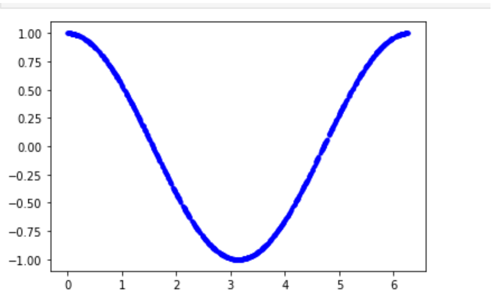
Wanted to determine if making a change in TensorFlow at Google Corroboratory would result in a TensorFlow Lite run on Raspberry Pi Pico.

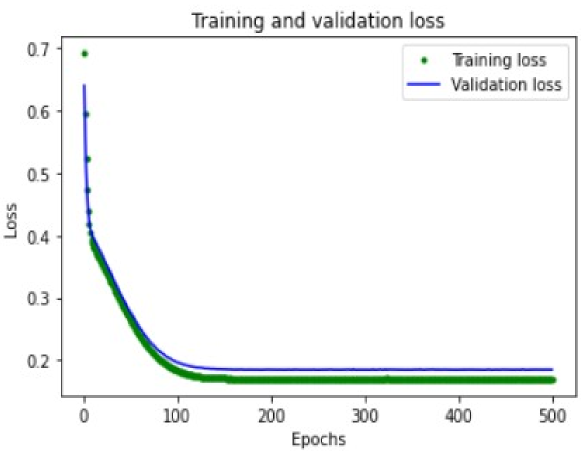
Changing the line y_values = np.sin(x_values).astype(np.float32) on left
to y_values = np.cos(x_values).astype(np.float32) image on the right.

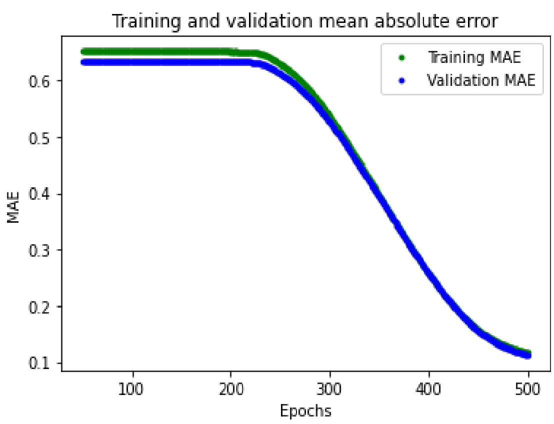
The original was saved in model-sin.cpp

This created a new model.cpp which was saved as model-cos.cpp



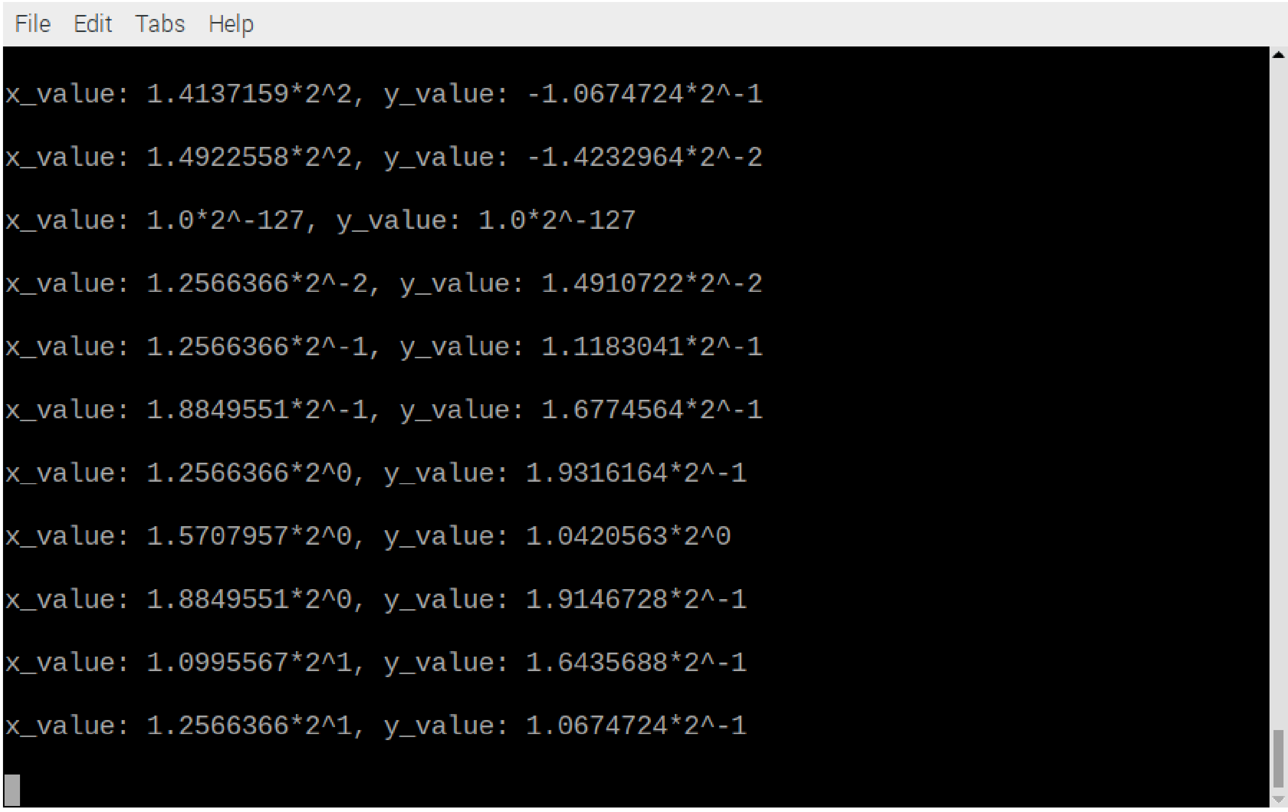






openocd -f interface/raspberrypi-swd.cfg -f target/rp2040.cfg -c "program examples/hello_world/hello_world.elf verify reset exit"

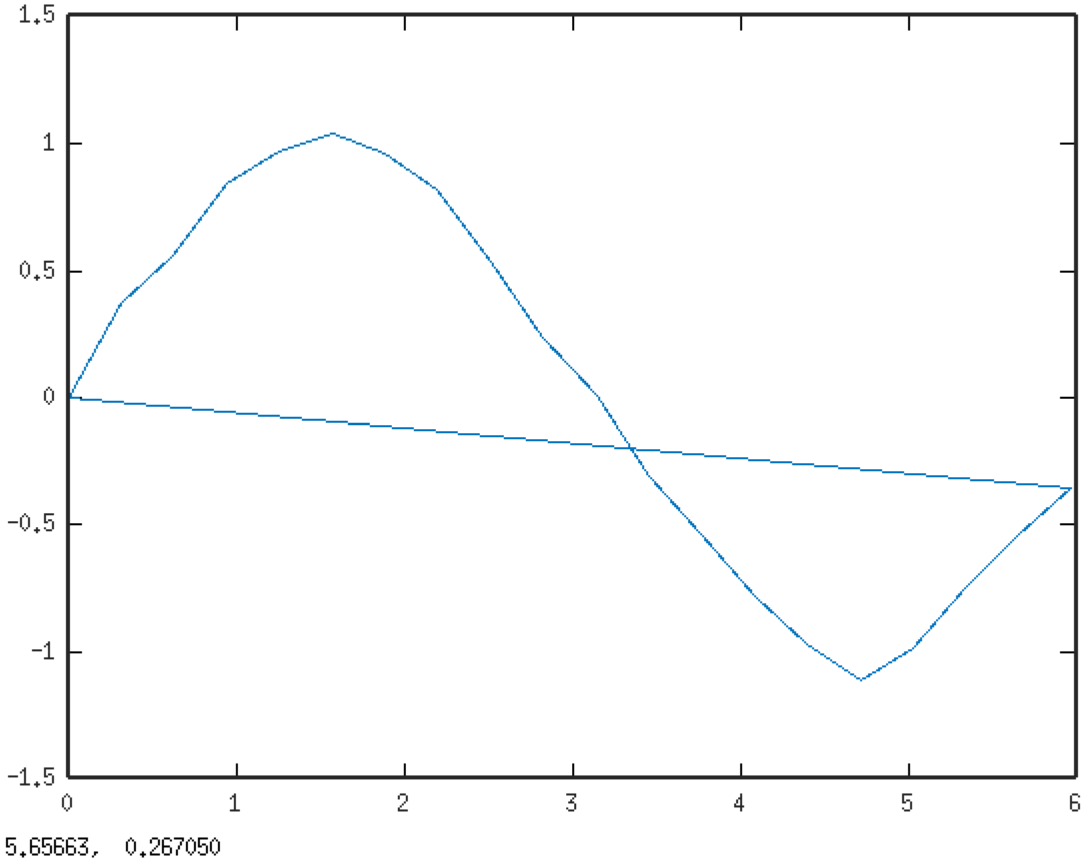
minicom myusb0



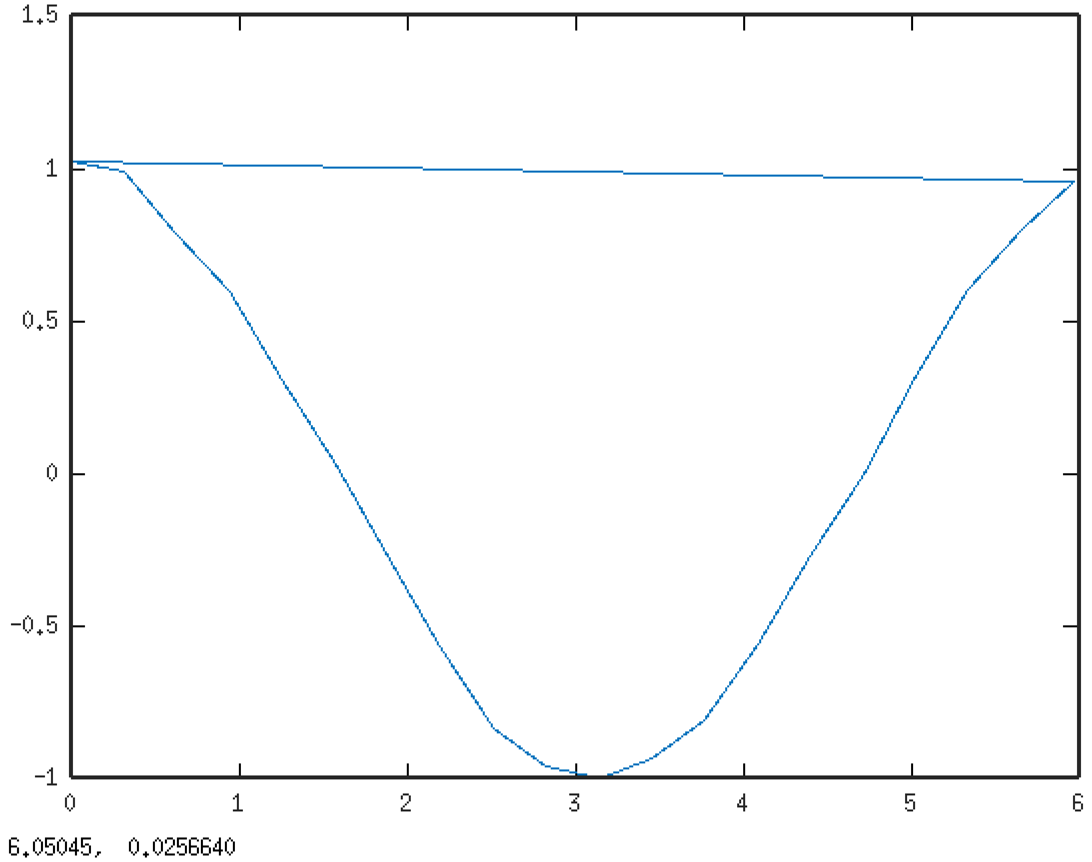

```
clear
close all
x=[1.0*2^-127,1.2566366*2^-2,1.2566366*2^-1,1.8849551*2^-1,1.2566366*2^0,1.5707957*2^0,1.8849551*2^0,   1.0995567*2^1,  1.2566366*2^1,   1.4137159*2^1,   1.5707957*2^1,   1.7278753*2^1,   1.8849551*2^1,    1.0210171*2^2,   1.0995567*2^2,   1.1780966*2^2,    1.2566366*2^2,    1.3351763*2^2,    1.4137159*2^2,   1.4922558*2^2,     1.0*2^-127];
y=[1.0230449*2^0,   1.9795505*2^-1,1.5803134*2^-1,   1.197711*2^-1,   1.2309807*2^-2,   1.0646323*2^-5,   -1.0646323*2^-2,   -1.1311719*2^-1,   -1.6634879*2^-1,   -1.913011*2^-1,   -1.9961854*2^-1,   -1.8631063*2^-1,   -1.6135832*2^-1,   -1.1145368*2^-1,   -1.0646323*2^-2,   1.0*2^-127,   1.2975204*2^-2,   1.2143458*2^-1,   1.5969484*2^-1,   1.913011*2^-1,   1.0230449*2^0];

 yy=[ 1.0*2^-127,   1.4910722*2^-2,   1.1183041*2^-1,   1.6774564*2^-1,   1.9316164*2^-1,   1.0420563*2^0,   1.9146728*2^-1,   1.6435688*2^-1,   1.0674724*2^-1,   1.8977287*2^-3,   1.0844163*2^-7,   -1.2199684*2^-2,   -1.0674724*2^-1,   -1.5588485*2^-1,   -1.9316164*2^-1,   -1.1098324*2^0,   -1.9655047*2^-1,   -1.4910722*2^-1,   -1.0674724*2^-1,   -1.4232964*2^-2,   1.0*2^-127];
figure
```

plot(x,yy)

figure
plot(x,y)



sin



xx
cos