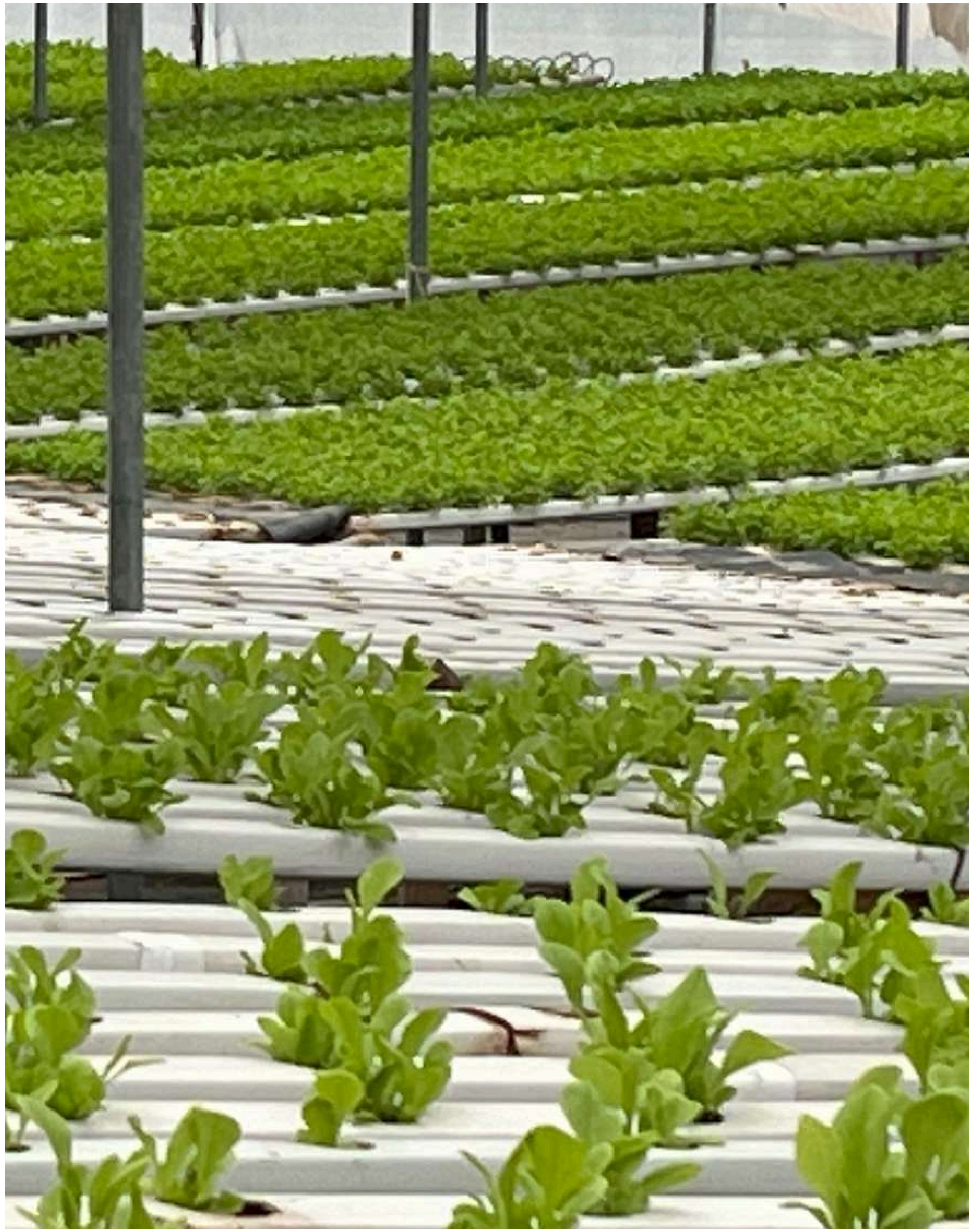


Development of an embedded system for the prediction of humidity in hydroponic germination phenolic sponge based on RNN/LSTM.

(Agronomic Industry).

Case study

Ing. Gustavo Castro (Argentina)

Hydroponic Agriculture

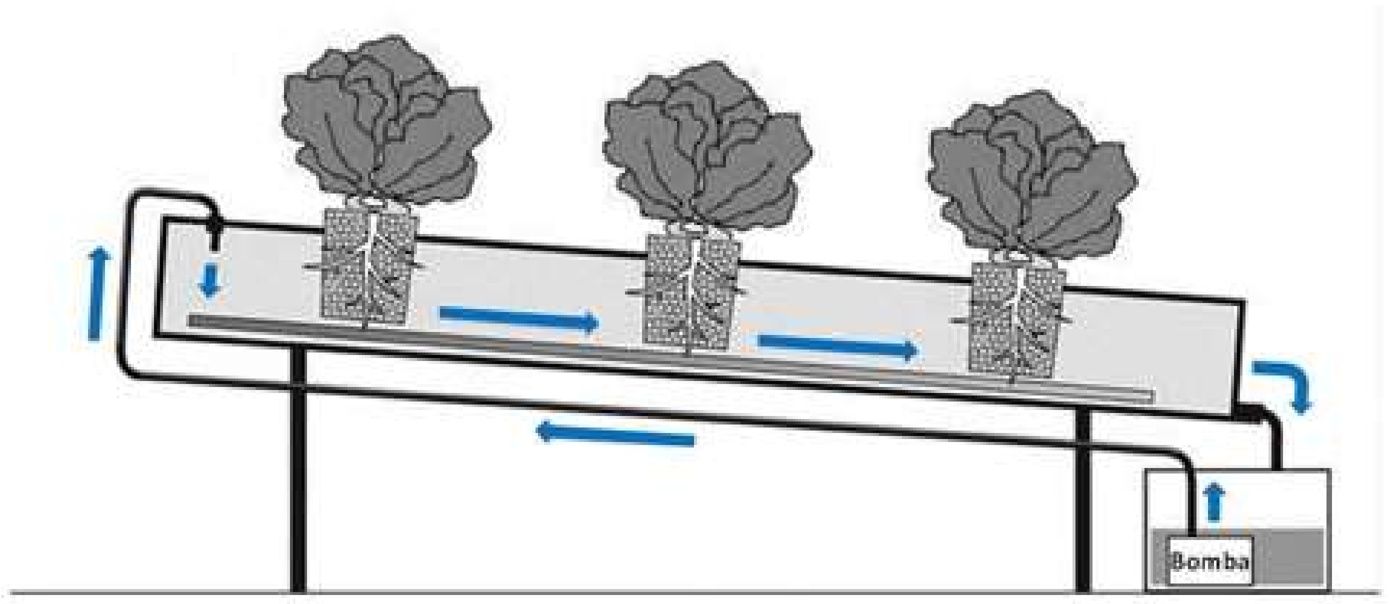
Hydroponics is the soilless production or cultivation technique, in which water and nutrients are supplied through a complete nutrient solution, providing the necessary conditions for better growth and development of the plant.

Hydroponic Agriculture in Argentina

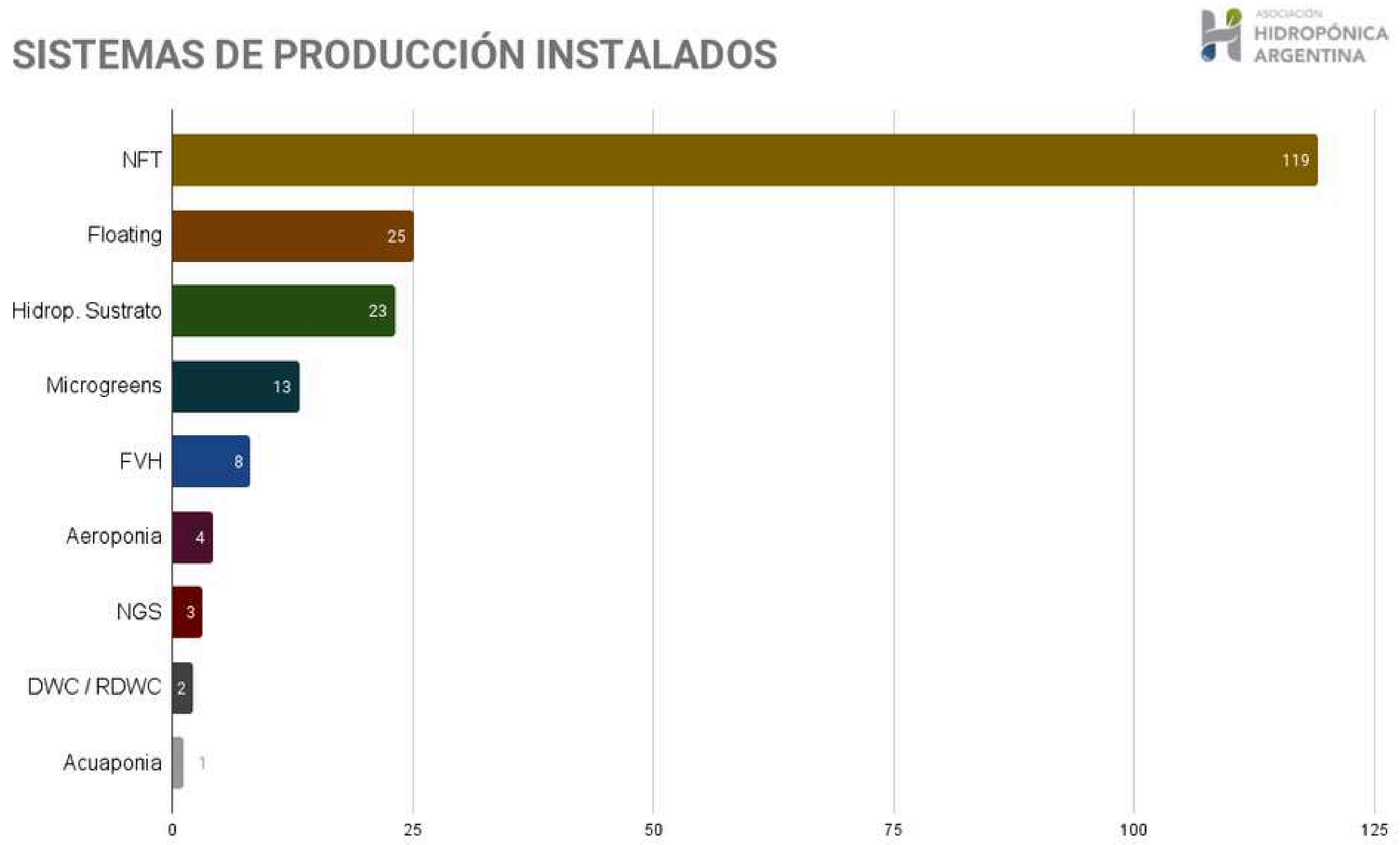
In Argentina, twenty of twenty-three provinces have producers who develop Hydroponic Agriculture, representing approximately 87% of the territory with the presence of this type of industry.

Hydroponic Agriculture in Argentina

The most used **production system** in this industry is the NFT (Nutrient Film Technique)



Esquema de un sistema NFT

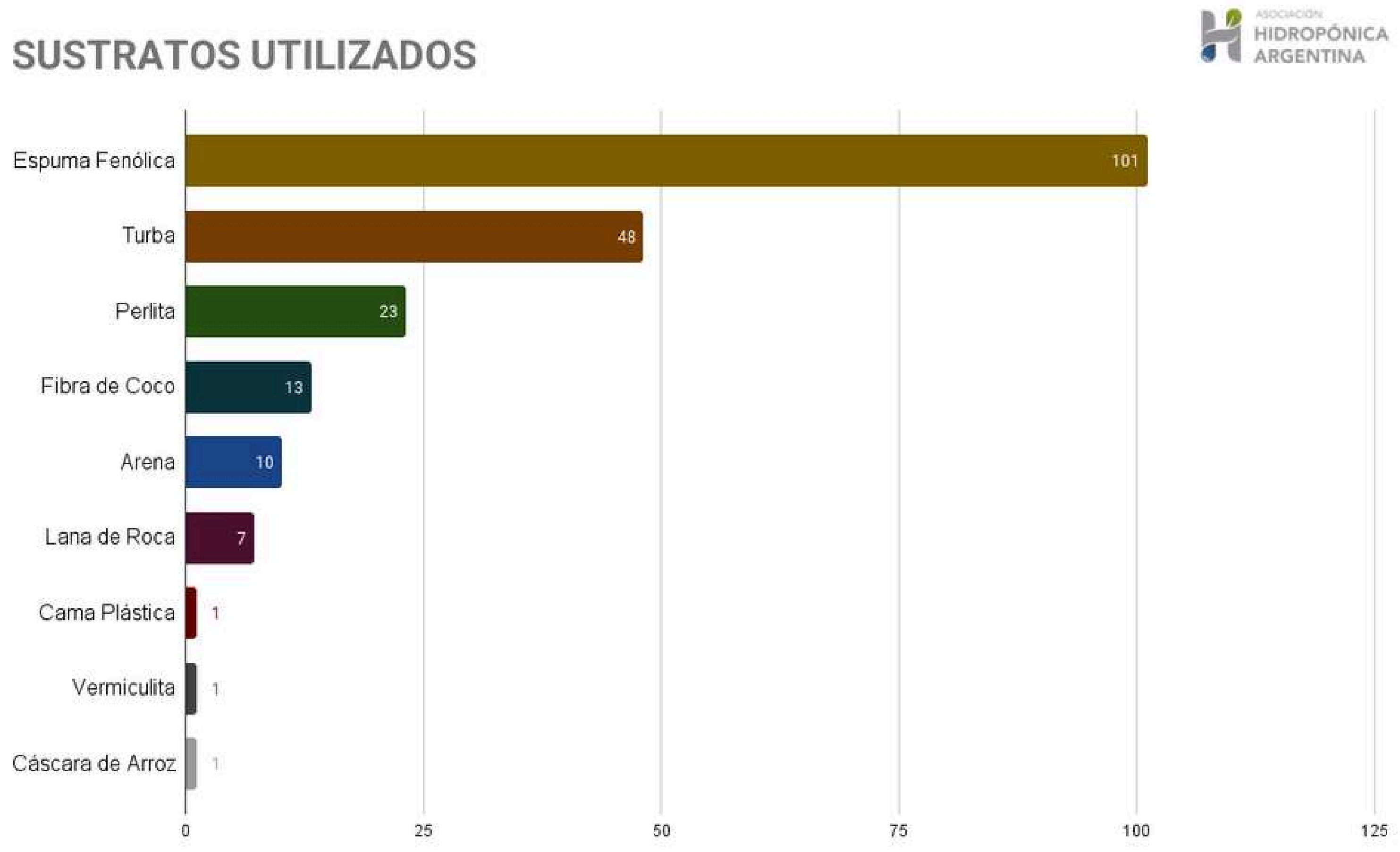


https://asociacionhidroponica.com.ar/encuesta/

Hydroponic Agriculture in Argentina

Of all producers, **87.3% carry out their own process of germination** from seeds to seedlings.

In 72.7% phenolic foam is used as a substrate



https://asociacionhidroponica.com.ar/encuesta/

Phenolic Foam

Mainly composed of Phenol, Formaldehyde, Catalysts, colorants and stabilizers, they have the following qualities:

- Density: 11 ~ 50 kg/m3
- pH: 6.5 ~ 7.4
- Electrical Conductivity: 0.55 ds/m
- Available water (% volume): 50 ~61%
- Water holding capacity (% volume): 50 ~ 91%
- Water retention time: 72 ~ 144 h

Espuma Fenóilica para germinación Hidropónica

Challenges nowaday

• **Manejo de la Humedad:** La gestión adecuada de la humedad durante el proceso de germinación es fundamental para garantizar un desarrollo óptimo de las plantines. Sin embargo, mantener niveles de humedad adecuados en el sustrato puede ser un desafío, especialmente en sistemas automatizados (Zhang et al., 2018).

• **Control de Enfermedades:** La germinación en sistemas hidropónicos puede estar asociada con un mayor riesgo de enfermedades, como la pudrición de la raíz, debido a la alta humedad y la ausencia de competencia microbiana del suelo. Abordar este desafío requiere estrategias efectivas de control de enfermedades (Wu et al., 2021).

• **Optimización de Nutrientes:** Garantizar un suministro adecuado de nutrientes sin provocar fitotoxicidad puede ser un desafío en sistemas hidropónicos (Sánchez-García et al., 2019)

Research **Objective**

Develop an embedded system for the prediction of humidity in phenolic sponge for hydroponic germination in order to improve efficiency during this production process

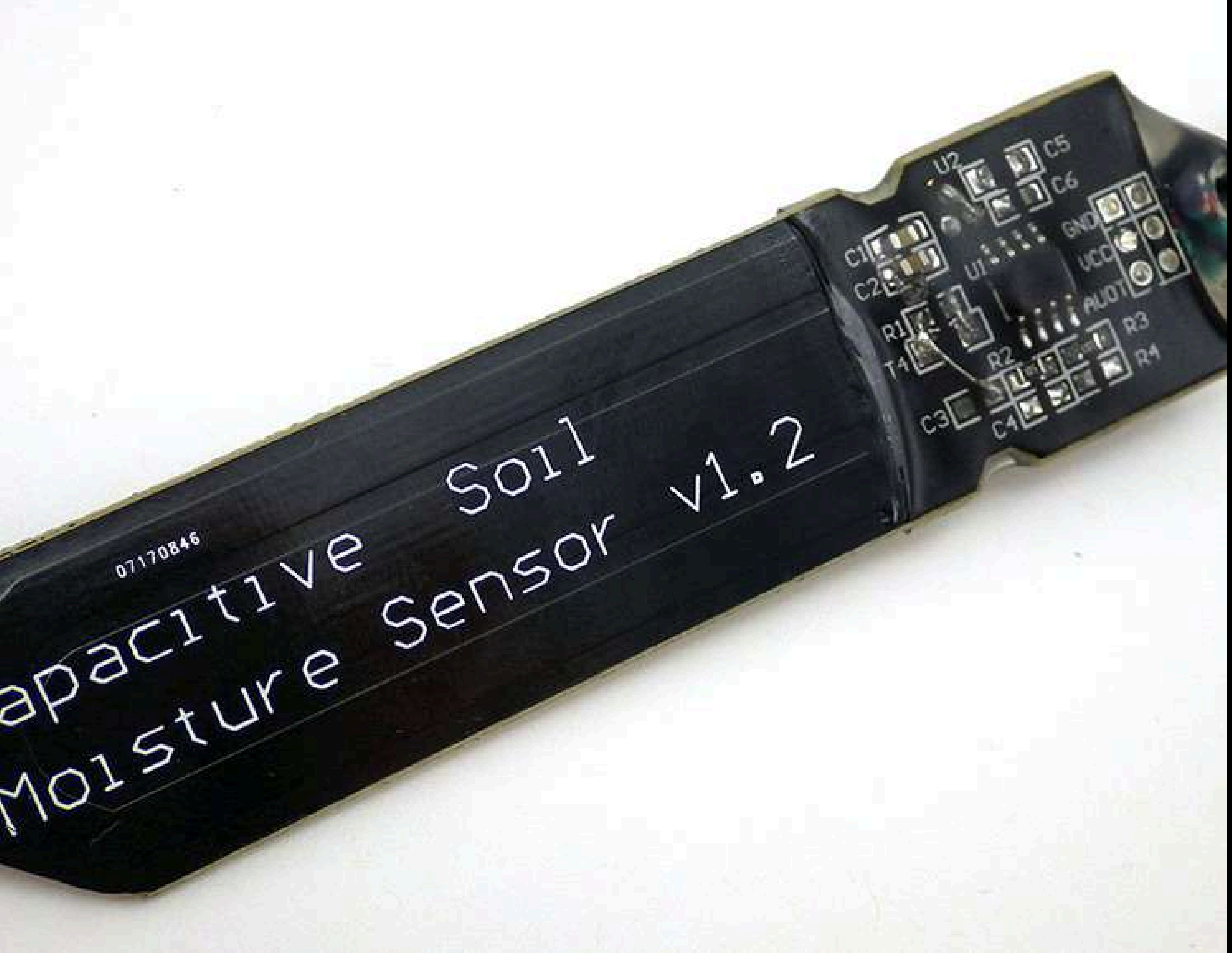
Methodology

Soil Moisture Sensors

Soil moisture sensors use a variety of physical principles and measurement techniques to determine the amount of water present in the soil.

The types of soil moisture sensors especially viable for application in hydroponic systems are Capacitance Sensors and Tensiometry Sensors.

Methodology



Soil Moisture Sensors

For the research, the Capacitive Sensor manufactured globally as "Capacitive Soil Moisture Sensor (v1.2)" was chosen and tested.

These sensors use coplanar traces to filter the high-frequency output of an oscillator, thus linearly translating the sensed relative humidity into a voltage difference (0~VDD).

Methodology

Soil Moisture Sensors

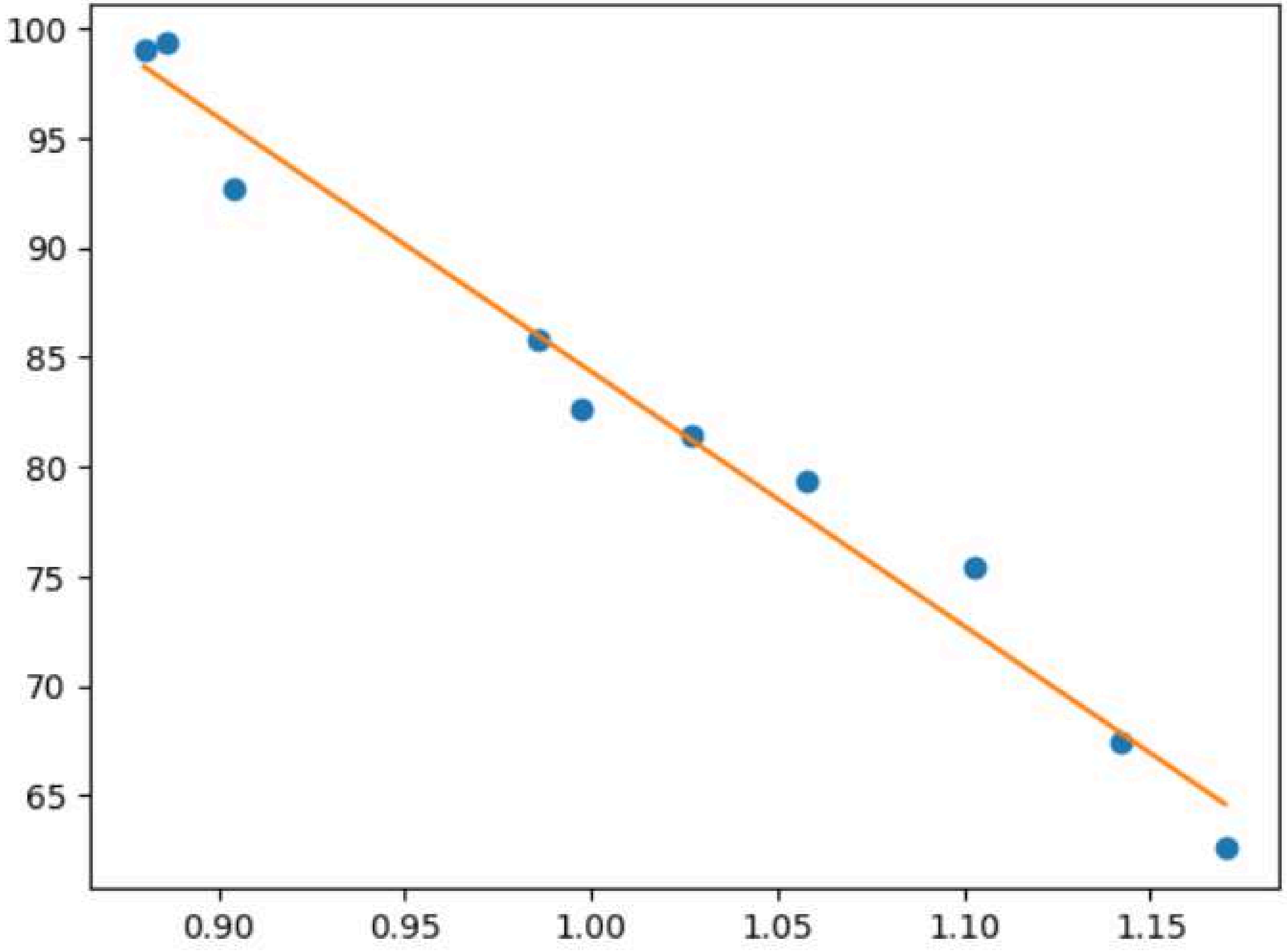
Rehearsal

observe and record the behavior of water evaporation of the phenolic sponge (decrease in the mass of retained water) versus the voltage variation at the output of the humidity sensor

Methodology

```
Parameters of the regression line:

m=  -116.04687806958512
b=  200.3753953040497
RMSE =  1.8011675021423856
```



Soil Moisture Sensors

With the survey of the mass values of water in the phenolic foam [RH%] and the values delivered by the sensor [Volts] in each measurement, using Python, the characteristic regression line was obtained to be able to model the sensor. As well as calculate its RMSE.

Methodology

Embedded System

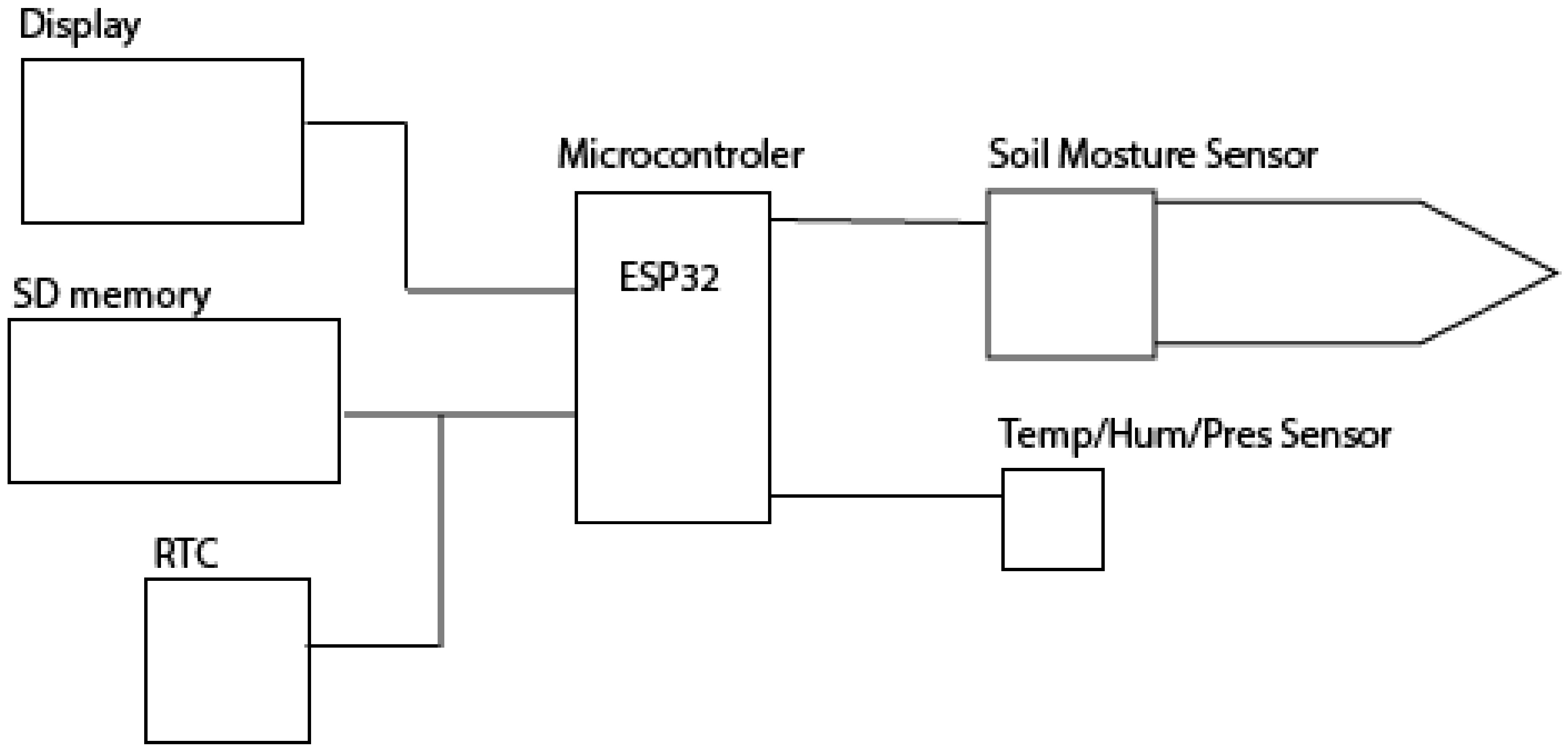
As a main function, the embedded system must measure environmental values such as Temperature, Pressure and Environmental Humidity, and also the Relative Humidity of the phenolic sponge, and predict its humidity in real time about 30 minutes in the future to be able to launch an alarm in case the future values are not within the optimum for seed germination.

Methodology

Embedded System

HARDWARE

Microprocessor: *ESP32*
RTC: *DS3231*
SD Reader: *microSD*
Display: *OLED SSD1306*
Temperature, Humidity, Pressure Sensor: *BMP280*

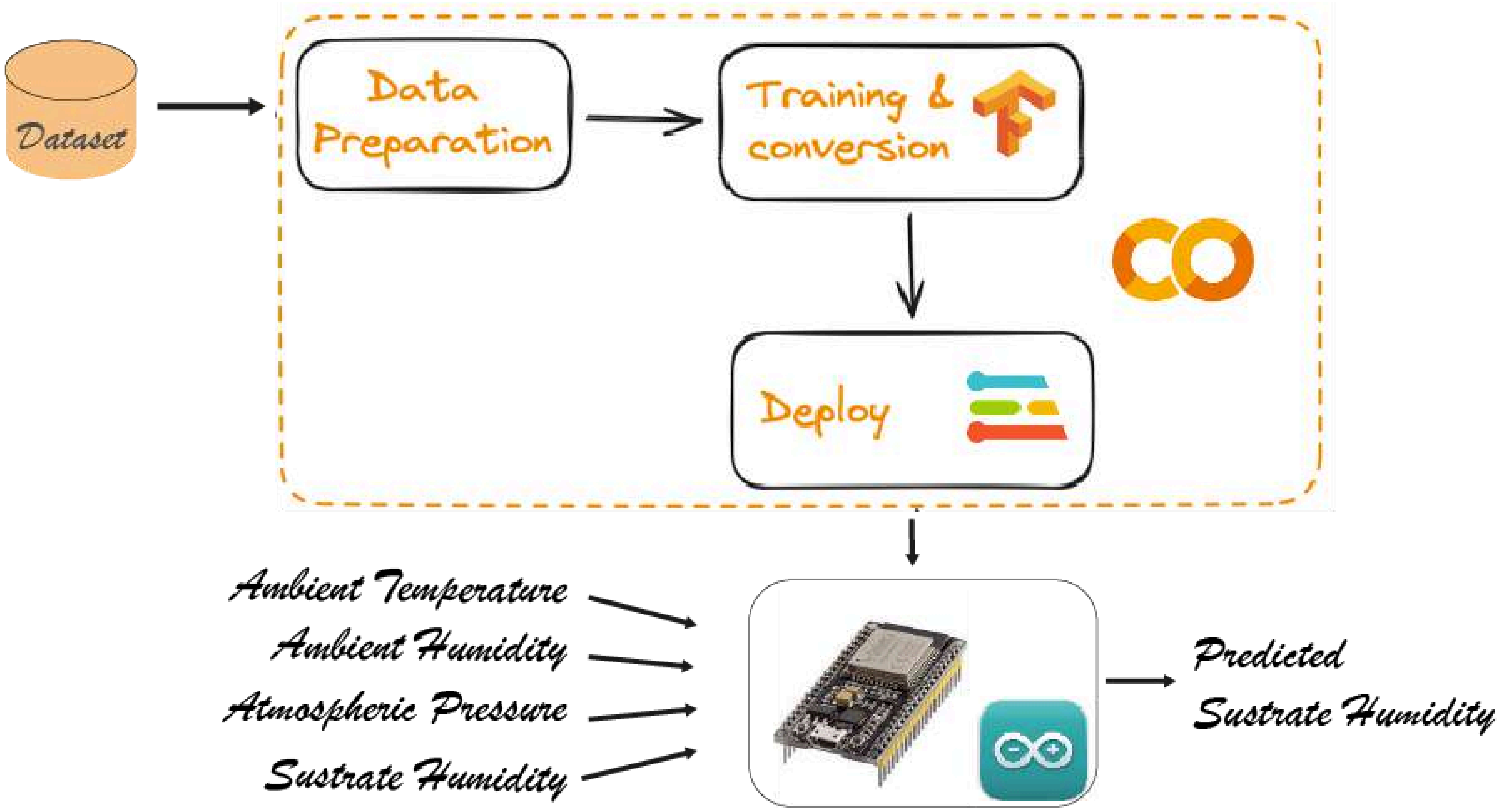


Methodology

Embedded System

THE PREDICTION
In order to achieve a prediction of humidity of the phenolic substrate, it is planned to **design a model** that at its input will enter the covariates: ambient temperature, ambient humidity, atmospheric pressure and humidity of the germination substrate, and that at its output will deliver the humidity prediction of substrate 30 minutes in the future.

Embedded System



Embedded System

The DATASET

After an investigation, no dataset was found with the covariates to be entered into the model, which is why an embedded system had to be manufactured to generate the dataset necessary for the training, validation and testing of the model.

To advance the development of the model, a dataset of environmental variables provided by the Planck Institute of Physics (Germany) was used.

Embedded System

Create a dataframe

The dataset is imported and created dataframe, and since the variable "substrate humidity" does not exist in this dataset, we choose "wind speed" as a covariate that replaces it in order to test the model until we can have the final dataset.

	p (mbar)	T (degC)	rh (%)	wv (m/s)
0	988.64	16.64	52.40	6.90
1	988.71	16.61	53.22	7.55
2	988.79	16.57	53.91	6.87
3	988.90	16.37	54.90	5.74
4	989.05	16.29	55.32	5.51
...	...	...	...	...
26059	985.13	15.51	97.70	0.88
26060	985.11	15.40	98.40	0.93
26061	985.07	15.27	99.90	1.02
26062	985.02	15.32	100.00	1.04
26063	984.96	15.29	99.90	1.20

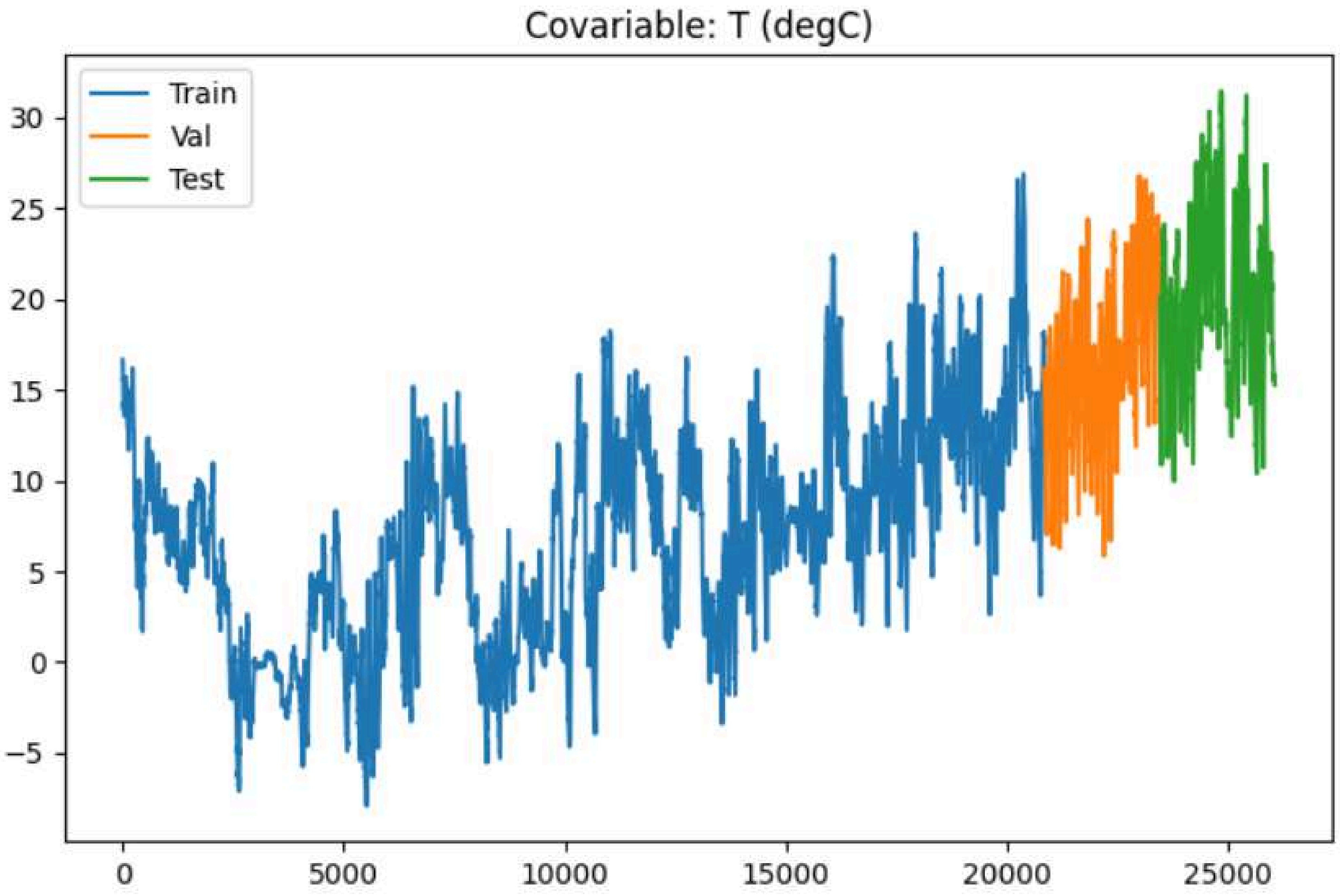
Embedded System

Preparing the data for Training

We should **split the dataset** into training, validation, and testing sets, with 80% for training, 10% for testing, and 10% for validation.

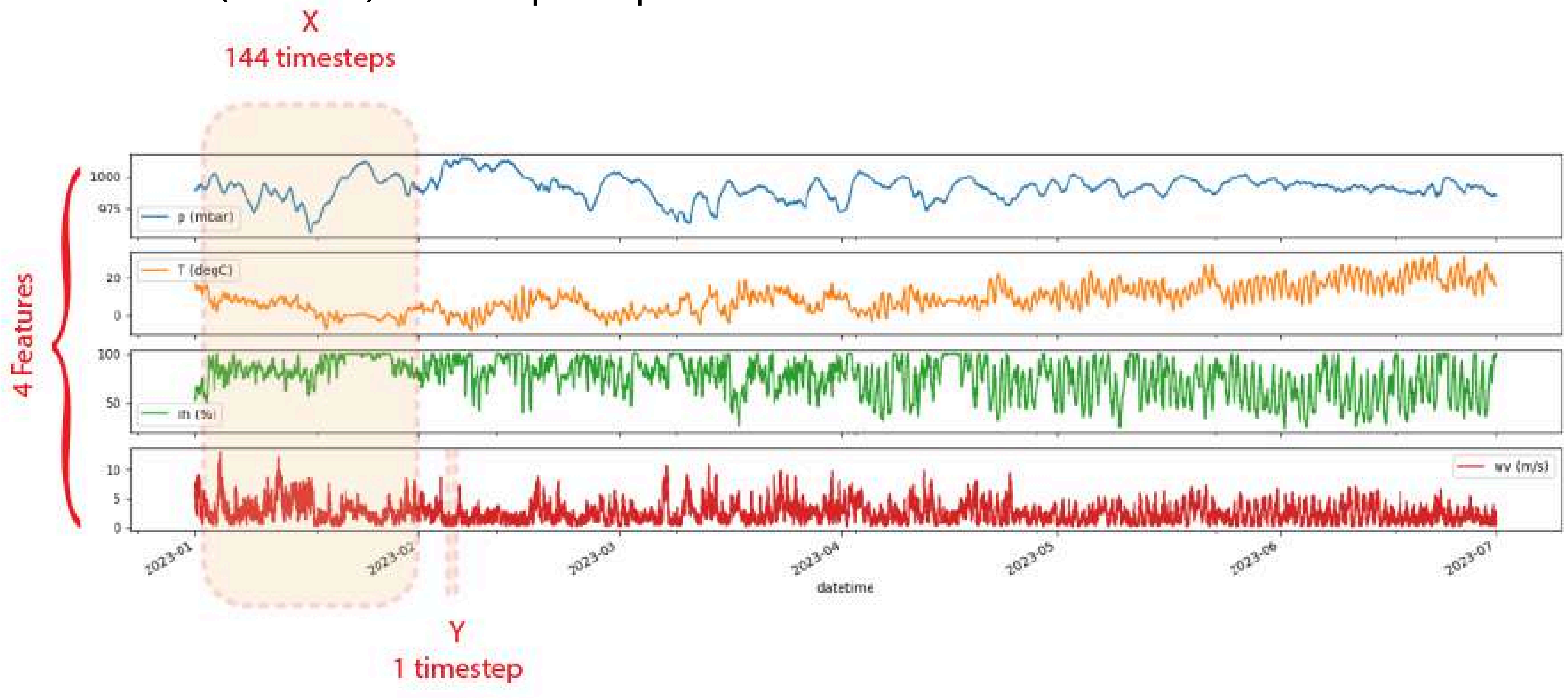
We verify that the data from each of the sets is not mixed and is consecutive.

The next step is to **normalize the data** so all features (columns) will range from 0 to 1



Data preparation for LSTM model

The dataset samples are separated by 10 minutes in time. We used one day of data by defining a input vector of 144 (6x24x4) timesteps to predict the selected variable 30 minute in the future.



Embedded System

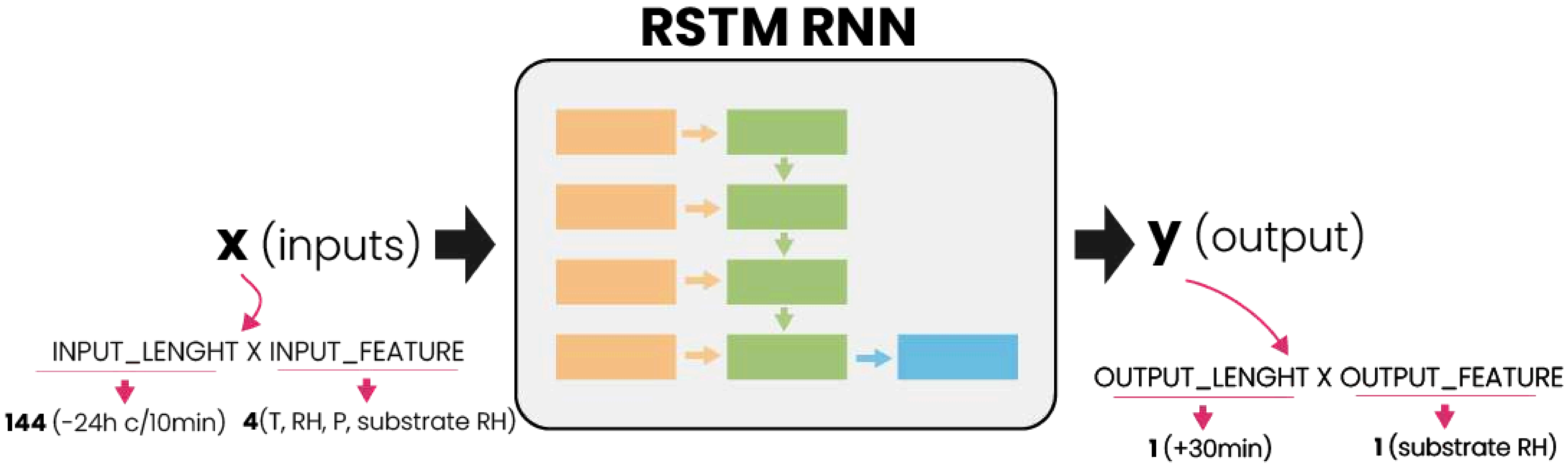
Design a Model

LSTM networks are particularly effective at modeling and predicting temporal sequences, making them suitable for working with time series data such as substrate humidity and the aforementioned environmental variables. The ability of LSTMs to capture long-term dependencies in data makes them ideal for prediction tasks where temporal relationships are important.

Embedded System

Design a Model

LSTM Models can be configured for different types of tasks, for this project the model will be used the **"many to one"** configuration, several input features, one output.



Design a Model

The LSTM architecture uses the Keras Sequential API to implement a many-to-one model.

```
# LSTM MODEL
N_UNITS = 128   #Size of hidden state (h) and memory cell (c)
INPUT_SHAPE = (x_tr_s.shape[1], x_tr_s.shape[2]) # 144 (samples, equivalent to 24 hours) x 4 (features)

modelo = Sequential()
modelo.add(LSTM(N_UNITS, input_shape=INPUT_SHAPE))
modelo.add(Dense(OUTPUT_LENGTH))

# Compile the model
modelo.compile(
    optimizer = 'adam',
    loss = 'mse',
)
```

Design a Model

The model was trained on the training data (X_train, y_train), while also being evaluated on a separate validation set (X_val, y_val). If the validation loss does not improve for five consecutive epochs, training will stop early and the model weights will return to those of the epoch with the lowest validation loss, effectively preventing overfitting and saving computational resources. The training process stopped in the 21th epoch.

Model evaluation calculated with RMSE (Root Mean Square Error), giving 0.021 on normalized data (maximum value is 1)

```
Comparative performances:
  RMSE train:    0.00011
  RMSE val:      0.00013
  RMSE test:     0.00019
```

Create TFLite LSTM Model - Float32

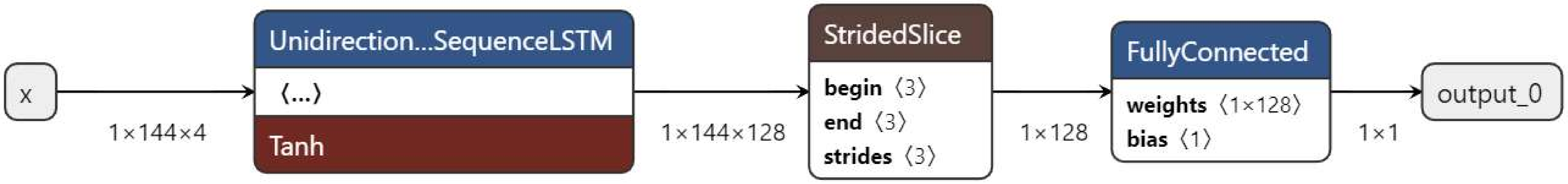
Converting a TensorFlow model to TensorFlow Lite (TFLite) for deployment on microcontrollers (TensorFlow
Lite Micro)

Operator Support: Only *UnidirectionalLSTM* is supported for LSTM operations in TensorFlow Lite Micro.

Quantization: is the process of reducing the precision of the numbers used to represent
a model's parameters, which is essential for running models on devices with limited precision and memory. Although float32 models are supported and tested, quantized models can sometimes present challenges, particularly with TensorFlow Lite Micro, which may not fully support quantization or may not have mature support for all operations in a quantized format. So, we will not use quantization in this project.

Create TFLite LSTM Model - Float32

Using *https://netro.app* we confirm that it only have unidirectional operators:



Deploying the Model with Edge Impulse Python SDK

For deploy this project, we will use the Edge Impulse Python SDK, a library to help us to develop machine learning (ML) applications for embedded systems.

The edgeimpulse Python SDK allows you to programmatically Bring Your Own Model (BYOM), developed and trained on any platform.

Deploying the Model with Edge Impulse Python SDK

Additionally, with it, we can estimate the RAM, ROM, and inference time for our model on the target hardware family.


```
Target results for float32:
===========================
{
    "device": "espressif-esp32",
    "tfliteFileSizeBytes": 276880,
    "isSupportedOnMcu": true,
    "memory": {
        "tflite": {
            "ram": 98891,
            "rom": 317992,
            "arenaSize": 98675
        },
        "eon": {
            "ram": 82264,
            "rom": 298776
        }
    },
    "timePerInferenceMs": 29720
}
```


The memory cost for TFLite micro use is estimated in 98 KB of RAM and 317 KB of ROM, what is OK with ESP32wroom.

Deploying the Model with Edge Impulse Python SDK

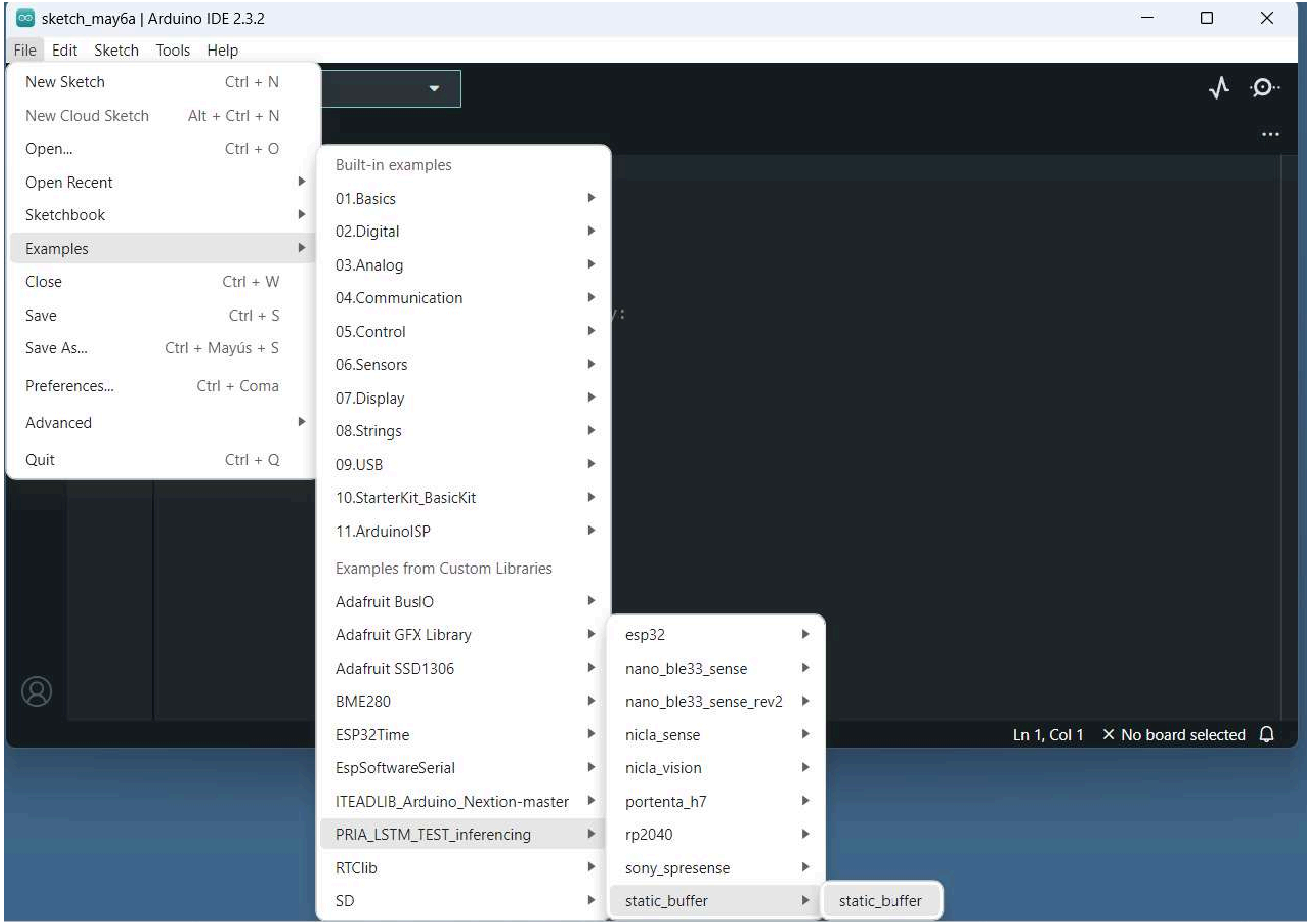
For deploy the model, we can call the *deploy()* function to convert the model from tflite to one of the **Edge Impulse supported outputs**. In this case, we will use ***arduino****, and* define the output type as ***Regression()***.

Having **the library created** *(lstm_float32_model.zip)*, it can be used through the **ARDUINO IDE**

Testing Inference

In Arduino IDE:

- go to Include Library and add.ZIP Library, select the library you create
- Select the sketch called “static buffer”



Testing Inference

For testing the model using the *static buffer sketch*, we will need a test datapoint to be loaded as “flat” input tensor in static const float features[] = { } . The datapoint with a shape as (144, 4), so input tensor should be (576,)

In Notebook:

```
reshaped_test = x_ts_s[0].reshape(-1)
reshaped_test.shape
```

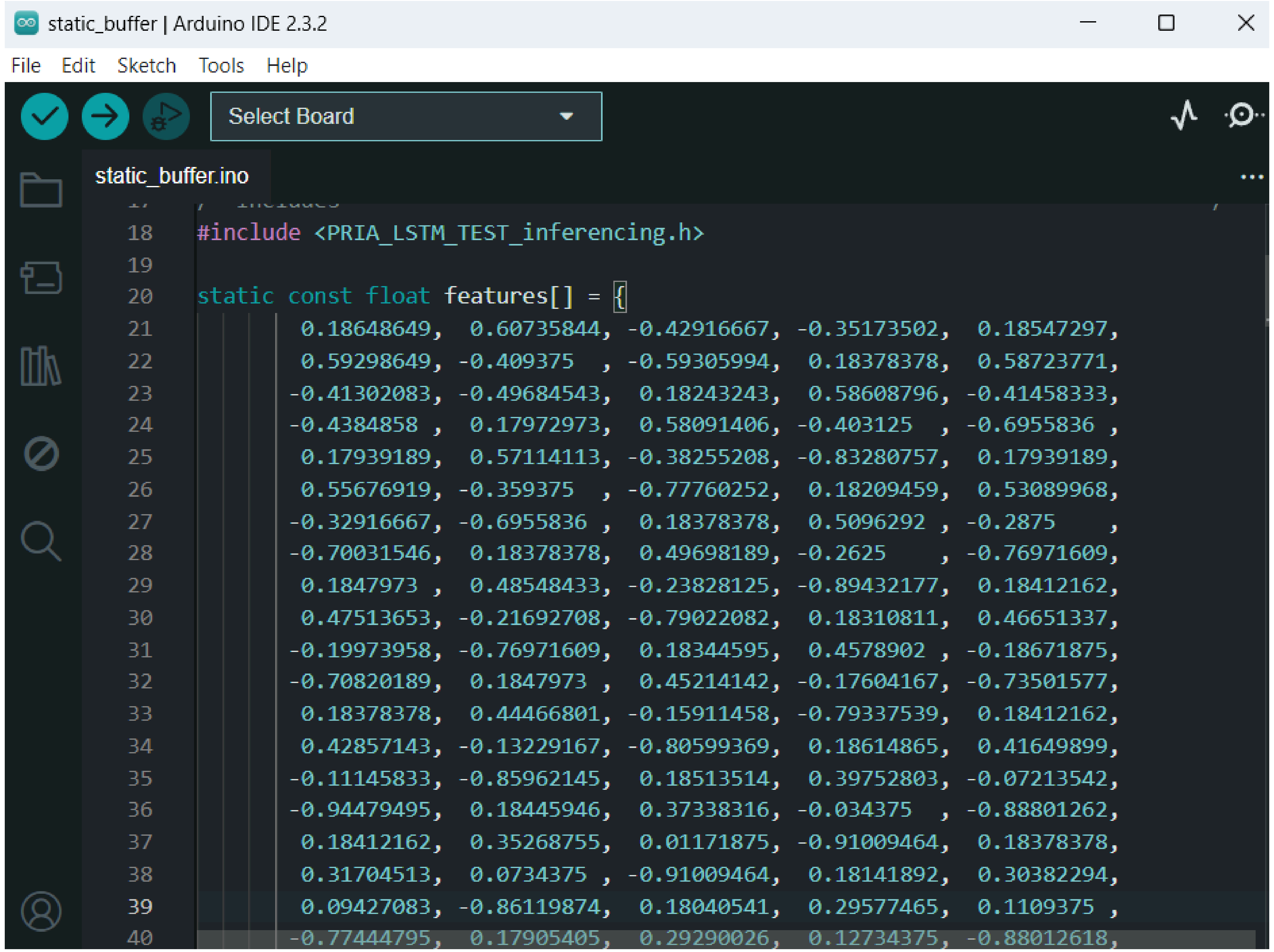
```
(576,)
```

```
reshaped_test
```

```
array([ 0.18648649,  0.60735844, -0.42916667, -0.35173502,  0.18547297,
        0.59298649, -0.409375  , -0.59305994,  0.18378378,  0.58723771,
       -0.41302083, -0.49684543,  0.18243243,  0.58608796, -0.41458333,
       -0.4384858 ,  0.17972973,  0.58091406, -0.403125  , -0.6955836 ,
        0.17939189,  0.57114113, -0.38255208, -0.83280757,  0.17939189,
        0.55676919, -0.359375  , -0.77760252,  0.18209459,  0.53089968,
       -0.32916667, -0.6955836 ,  0.18378378,  0.5096292 , -0.2875    ,
       -0.70031546,  0.18378378,  0.49698189, -0.2625    , -0.76971609,
        0.1847973 ,  0.48548433, -0.23828125, -0.89432177,  0.18412162,
```

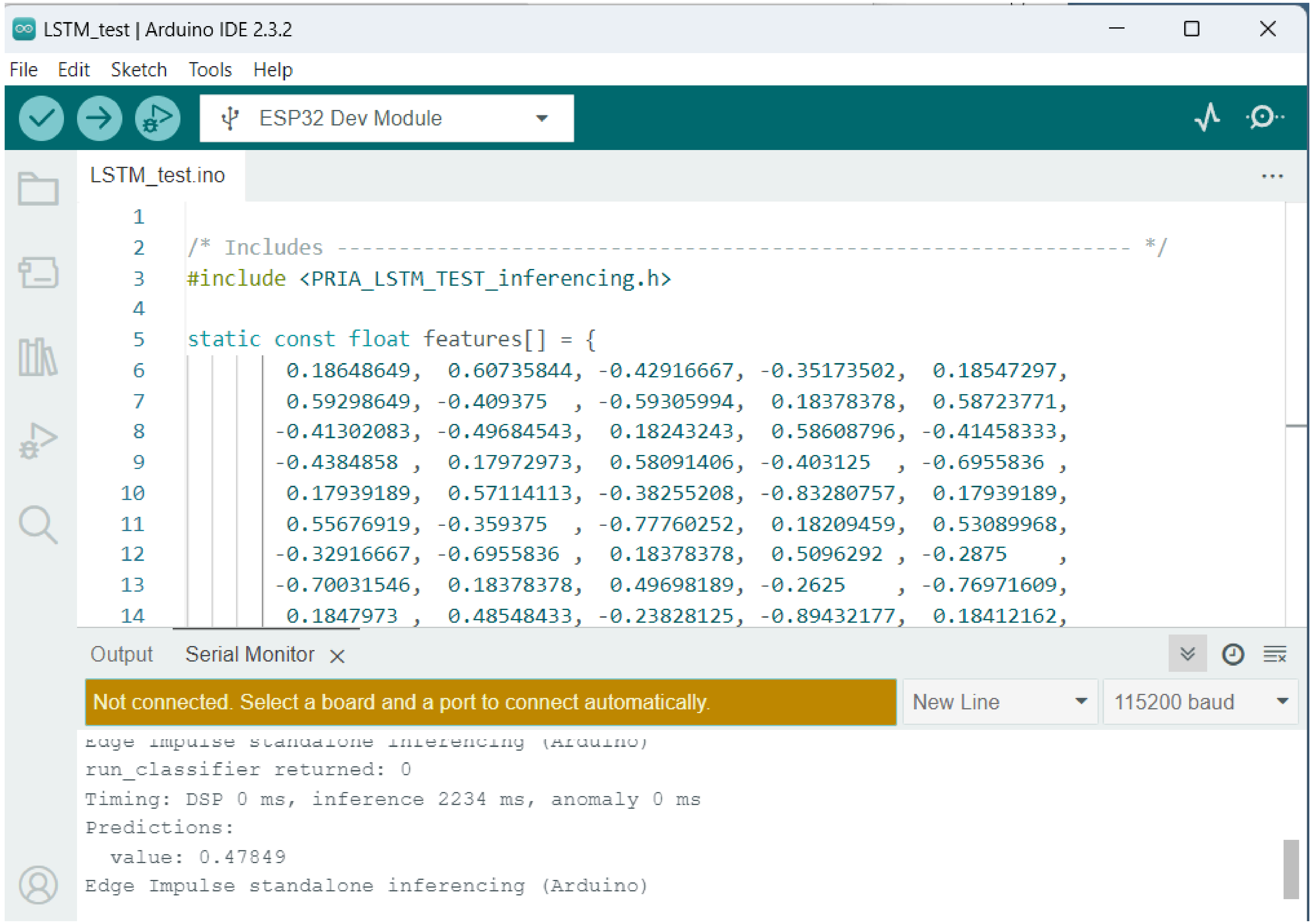
Testing Inference

Copy the values and past them on the Arduino Sketch.



Testing Inference

Then that connect our *ESP32wroom*, run the sketch, and we can see the **result of prediction** on the Serial Monitor:



y predicted = 0.47849

Testing Inference

If we look at the real value of y_test[0] we will get *0.4785858*

```
y_ts_s[0]
```

```
array([[0.4785858]])
```

We verified that the prediction value for this data point is 0.47849, which has an error of 0.0000957 from the actual value, and the latency was around 2.2 seconds, which is acceptable for this project (we will generate a prediction 30 minutes in the future).

Rescaling inference results in real values

Es importante reescalar el resultado de la inferencia para obtener el valor en la unidad original.
Durante la normalización, los parámetros de Max y Mín para cada cavariable se almacenan en un archivo de texto, que se puede utilizar para revertir el proceso de normalización y convertir las predicciones de nuestro modelo nuevamente a escala original.

Remembering, the min-max scaling formula is:

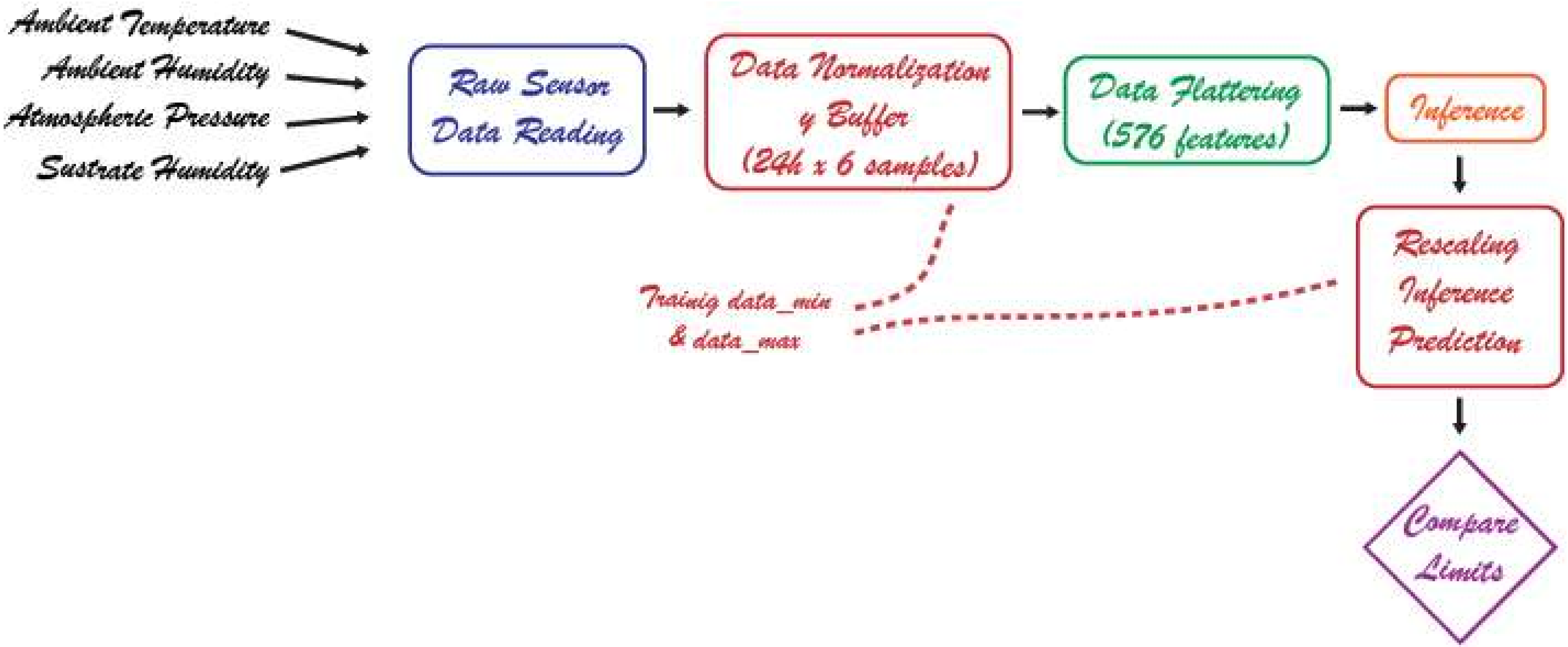
$$X_{\text{norm}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

And reverse normalization is:

$$X = X_{\text{norm}} \times (X_{\text{max}} - X_{\text{min}}) + X_{\text{min}}$$

Final Integration

Once the dataset obtained from the hydroponics field is obtained, the model will be re-trained and deployed again, now in a final embedded system.



Final Integration

The final embedded system will make the substrate humidity prediction 30 min in the future, and compare it with minimum and maximum humidity values established by the producer.

If the prediction is below the minimum value, a RED LED will light up, and if the prediction is located above the maximum humidity value, the system will turn on an AMBER LED.