2025 TEEP Progress Report Week-15

Lab1 – On ESP32, complete the following tasks:

This project focuses on optimizing the single-cell battery charging monitoring system completed last week, with the goal of improving the overall stability of measurement and display. Each step of the experimental process must be thoroughly documented, compiled into the project report, and also presented in video format to ensure completeness and traceability of the research process.

1. Description

This week's primary focus was on deepening we understanding of the working principles of the **ADS1115** and **INA219** modules used in our **3-cell (3S) Li-ion battery monitoring system**. The system, built using an **ESP32 microcontroller**, successfully measures individual cell voltages, total pack voltage, and current flow. Although the project was completed last week and operated successfully, we noticed minor fluctuations in the readings during extended testing. Therefore, this week was dedicated to studying the underlying causes of those variations and exploring methods to further **enhance measurement accuracy and system stability**.

2. Work Done

Throughout this week, we conducted a detailed theoretical review and several focused experiments to refine the system's performance. our main objectives were to understand the exact operation of the ADC and current sensor modules, evaluate their limitations, and optimize the hardware–software interaction. The key activities included:

- Reviewing **I2C communication behavior** between ESP32, ADS1115, and INA219.
- Studying **datasheets, application notes, and calibration procedures** from Texas Instruments and Adafruit.
- Experimenting with different **ADS1115 input configurations** and programmable gain settings.
- Performing **recalibration of INA219** for improved current and voltage accuracy.
- Inspecting **hardware connections, grounding, and resistor tolerances** to minimize analog noise and offset errors.

The system now operates more smoothly, with improved consistency in the readings. This week's in-depth research provided a much clearer understanding of how these precision modules function internally and how small adjustments can lead to significant performance improvements.

3. Detailed Description of Study

3.1 Overview of ADS1115

The **ADS1115** is a **16-bit precision delta-sigma (ΔΣ) analog-to-digital converter (ADC)**. It is designed for accurate low-frequency measurements and communicates via the **I2C bus**, which makes it ideal for interfacing with microcontrollers like the **ESP32**.

The device includes:

- An internal **voltage reference** and **clock oscillator**.
- A **programmable gain amplifier (PGA)** for scaling input voltages.
- A **digital comparator** for threshold-based monitoring applications.

The ADS1115 measures the differential input signal **VIN = V(AINP) – V(AINN)**. The differential, switched-capacitor ΔΣ modulator attenuates common-mode noise, providing high accuracy and stability.

Conversion Modes

The ADS1115 operates in two modes:

- **Single-Shot Mode:** Performs one conversion and powers down to save energy.
- **Continuous Mode:** Continuously samples the input signal and updates the output at the programmed data rate.

This flexibility allows users to balance **power consumption** and **data update rate** according to their application.

Noise Performance

The **noise performance** of ΔΣ ADCs depends heavily on the **oversampling ratio (OSR)**—the ratio of the modulator frequency to the output data rate. Increasing the OSR improves noise performance since more samples are averaged to produce one output.

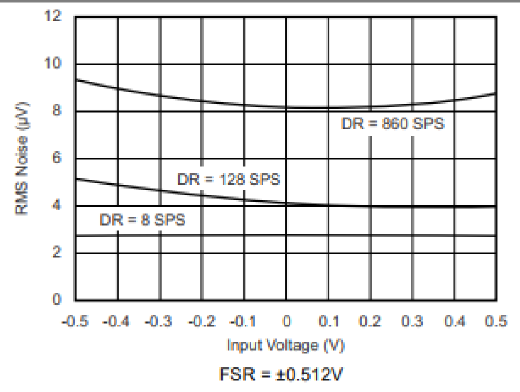


Figure 5-14. Noise vs Input Signal

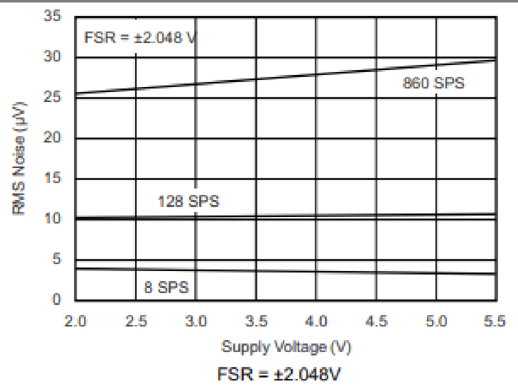


Figure 5-15. Noise vs Supply Voltage

According to the TI datasheet:

- At lower data rates (e.g., 8 to 128 samples per second), the ADS1115 achieves near **16-bit effective resolution** with minimal input-referred noise.
- Higher data rates (e.g., 860 SPS) lead to slightly lower resolution due to reduced averaging.

This means that for stable battery voltage measurements, **a lower data rate with continuous conversion** provides more reliable readings.

Practical Findings

During my testing, I noticed that incorrect voltage readings often resulted from:

- **Mismatched voltage dividers** used to scale battery voltage into the ADC input range.
- **Floating grounds** or poor wiring that introduced common-mode noise.
- **Incorrect PGA settings** that caused signal clipping.

By adjusting these parameters and using lower sample rates, I was able to improve reading stability.

3.2 Overview of INA219

The **INA219** is a **high-side current and power monitoring sensor** that uses a precision **shunt resistor** to measure current and voltage simultaneously. It communicates through the **I2C interface** and provides three key parameters:

1. **Bus voltage** (voltage across the load)
2. **Shunt voltage** (voltage drop across the resistor)
3. **Calculated current and power**

It is ideal for battery monitoring because it measures current **on the high side** of the circuit, meaning the load can share a common ground with the system.

Working Principle

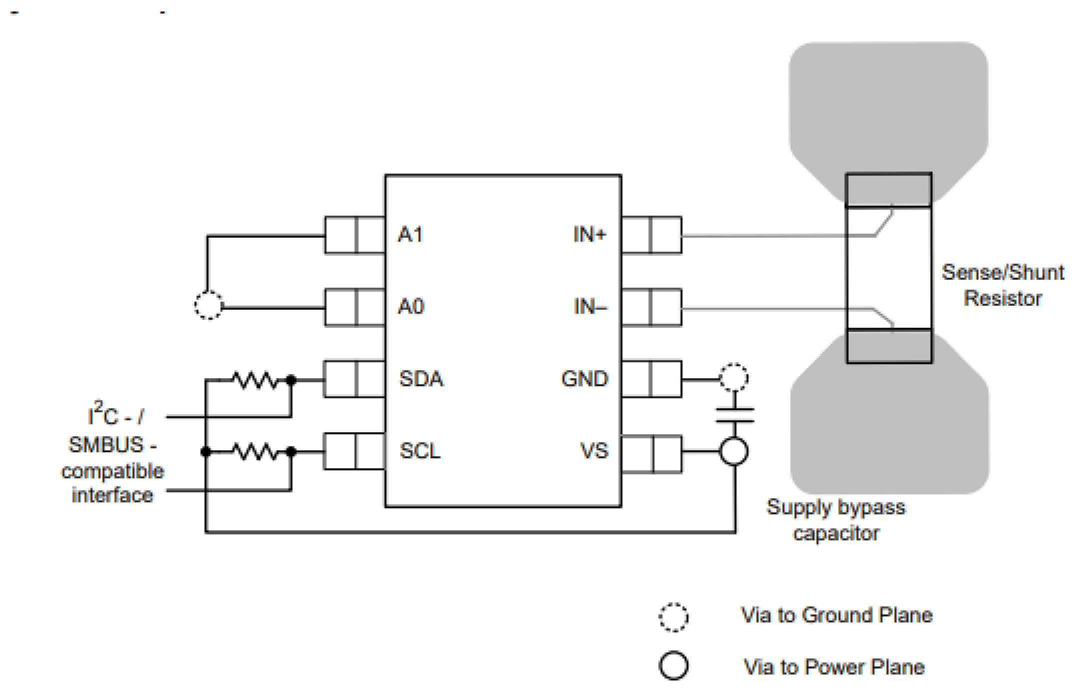
The INA219 measures the tiny voltage drop (in millivolts) across its internal shunt resistor and uses a **high-precision amplifier** to calculate the current. Using the formula:

I=VshuntRshuntI = \frac{V_{shunt}}{R_{shunt}}I=RshuntVshunt

and multiplying by the measured bus voltage, it can also calculate **power consumption (P = V × I)**.

Calibration and Accuracy

The module requires calibration to set the correct current and power scaling factors. I learned that incorrect calibration constants can lead to drastically wrong readings, especially when using external shunt resistors or varying supply voltages.



I studied the Adafruit INA219 guide in detail, understanding how to adjust the calibration register and conversion timing for better accuracy in my circuit.

Practical Findings

While experimenting, I noticed that:

- The **I2C address conflict** between ADS1115 and INA219 must be carefully managed.
- **Noise and ground loops** can affect shunt voltage readings.
- Using **shielded cables** and **decoupling capacitors** near the sensor improved signal stability.

This study helped me better understand the importance of proper hardware layout and accurate software calibration when using multiple I2C sensors.

4. Next Steps

- Redesign the voltage divider for balanced cell measurement.
- Implement **moving average filtering** in code to smooth noisy data.
- Calibrate the **INA219** for precise current measurement.
- Begin integrating the readings into a **Flask-based dashboard** for visualization once stability is achieved

5. Resources Studied

During this week, we studied several technical documents and guides to understand the internal operation of the modules:

1. **Texas Instruments Application Report:**
   Noise and Resolution in Delta-Sigma ADCs
   https://www.ti.com/lit/an/sbaa535a/sbaa535a.pdf

2. **Texas Instruments Datasheet for ADS1114/ADS1115:**
   https://www.ti.com/lit/ds/symlink/ads1114.pdf

3. **Adafruit INA219 Current Sensor Guide:**

   https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ina219-current-sensor-breakout.pdf

These resources helped us understand concepts like oversampling, ADC noise, programmable gain control, and shunt-based current measurement in depth.

Lab2- Answer questions in detail.

1. The name "TensorFlow" comes from the fact that its data operations are mainly based on:

Answer: (B) **Tensors**
TensorFlow is named after "tensors," which are multi-dimensional arrays that flow through a computational graph. A tensor is a generalization of vectors and matrices to higher dimensions, and TensorFlow performs mathematical operations on these tensors. The name reflects how data (tensors) "flows" through layers of a neural network during computation.

---

2. Which programming language is mainly used to implement the core of PyTorch?

Answer: (C) **C++**
Although PyTorch provides a Python interface for ease of use, its underlying core — including tensor computations and automatic differentiation engine — is written in **C++** for performance optimization. This allows PyTorch to execute high-speed numerical operations while still being user-friendly for Python developers.

---

3. Which of the following is an integrated development environment (IDE) included in Anaconda?

Answer: (C) **Spyder**
Spyder (Scientific Python Development Environment) is bundled with Anaconda and is designed for data science and scientific computing. It includes features like an editor, variable explorer, and integrated IPython console, making it convenient for developing machine learning and deep learning scripts.

---

4. Which command is used to create an Anaconda virtual environment named keras_tf?

Answer: (B) **conda create --name keras_tf anaconda**
This command creates a new environment called `keras_tf` and installs the default Anaconda packages. Each environment isolates libraries and dependencies to prevent version conflicts when working with multiple deep learning frameworks.

5. Which command is used to set Keras to use PyTorch as the backend?

Answer: (A) **set KERAS_BACKEND=pytorch**
This command (in Windows Command Prompt) modifies the environment variable `KERAS_BACKEND`, telling Keras which backend (TensorFlow, Theano, or PyTorch) to use for computations.

---

6. Briefly explain one difference between TensorFlow and PyTorch in terms of usage or development style.

Answer:
One key difference is that **TensorFlow** traditionally used **static computational graphs**, while **PyTorch** uses **dynamic computational graphs**.

- In TensorFlow, the graph is defined first and executed later, which is efficient but less intuitive for debugging.
- PyTorch builds the graph dynamically during runtime, allowing easier debugging and more flexible model design. This makes PyTorch more "Pythonic" and preferred for research and experimentation.

---

7. What is a Python virtual environment, and why is it important for deep learning development?

Answer: A **Python virtual environment** is an isolated workspace that allows developers to create separate environments for different projects, each with its own set of installed packages and dependencies. This means that libraries, frameworks, and even Python versions installed inside one environment do not interfere with those in another.

In deep learning development, using a virtual environment is extremely important because different projects may require different versions of frameworks like TensorFlow, PyTorch, or Keras. For example, one project may use TensorFlow 2.10 while another uses TensorFlow 1.15. Without a virtual environment, installing both versions on the same system could cause dependency conflicts and make the setup unstable.

Virtual environments also help in **reproducibility** — meaning the same environment setup can be shared or recreated on another computer, ensuring consistent behavior and results across systems. Tools like **Anaconda** and **.venv** simplify the creation and management of these environments. By isolating project dependencies, developers can work more efficiently, prevent compatibility issues, and maintain cleaner system configurations.

8. What command is used in Anaconda to list all existing virtual environments?

Answer: In Anaconda, you can list all the virtual environments currently created on your system by using the following command in the Anaconda Prompt or terminal:

Bash : **conda env list**
This command displays a table of all existing environments, along with their paths on your computer. The active environment (the one currently in use) will be marked with an asterisk `(*)`.

---

9. State one important feature of Keras 3.0.

Answer: One of the most significant features introduced in **Keras 3.0** is its **multi-backend support**. Earlier versions of Keras were tightly coupled with TensorFlow as the backend engine, but Keras 3.0 was redesigned to work seamlessly with multiple deep learning frameworks such as **TensorFlow**, **PyTorch**, and **JAX**.

This means that developers can write a Keras model once and run it on any of these backends without changing the model code. It provides exceptional flexibility and portability — researchers can easily switch between frameworks depending on performance, hardware support, or experimentation needs.

Additionally, Keras 3.0 emphasizes simplicity, clean APIs, and better integration with modern tools. It also supports high-performance training on GPUs and TPUs, making it a truly universal front-end interface for deep learning. This innovation positions Keras 3.0 as a framework that bridges multiple ecosystems, enabling broader adoption across research and industry.

---

10. What are two differences between Google Colab and Jupyter Notebook running on a local computer?

Answer: Although both **Google Colab** and **Jupyter Notebook** provide an interactive notebook environment for writing and executing Python code, they differ in how they operate and in the resources they provide.

1. Hardware and Execution Environment:
Google Colab runs in the cloud and provides access to powerful hardware accelerators such as GPUs (Graphics Processing Units) and TPUs (Tensor Processing Units) for free. This is especially beneficial for deep learning tasks that require high computational power. Jupyter Notebook, on the other hand, runs locally on your computer and depends entirely on your

system's CPU or GPU. The performance of models in Jupyter is therefore limited by your local hardware.

2. Storage and Accessibility:
Answer: Colab notebooks are stored in Google Drive, allowing users to access their work from any device with an internet connection. It also supports real-time collaboration similar to Google Docs. Conversely, Jupyter Notebooks are stored locally on the computer where they are created, and users must manually back up or share them.

In summary, Google Colab offers cloud-based convenience, collaboration, and hardware acceleration, while Jupyter Notebook offers offline access, control over the environment, and privacy for local experimentation.

---

11. Write the complete command sequence to create a virtual environment named keras_torch → activate it → install Keras → install PyTorch.

Answer: Below is the step-by-step sequence of commands used in Anaconda Prompt or terminal:

Bash :
```
conda create --name keras_torch anaconda
conda activate keras_torch
pip install keras
pip install torch torchvision torchaudio
```

- The first command creates a new virtual environment named **keras_torch** with default Anaconda packages.
- The second command activates the newly created environment so that any subsequent installations are confined within it.
- The third and fourth commands install **Keras** and **PyTorch** respectively, ensuring both frameworks coexist in the same environment.

---

12. Explain how to modify the keras.json file in Windows to set TensorFlow as the backend for Keras.

Answer:

- **Locate the configuration file:** Open File Explorer and go to

  C:\Users\"UserName"\.keras

- **Open the file:**

  Right-click on `keras.json` and open it with a text editor such as Notepad.

- **Modify the backend line**

  json: {

  "backend": "tensorflow",

  "epsilon": 1e-07,

  "floatx": "float32",

  "image_data_format": "channels_last"

  }

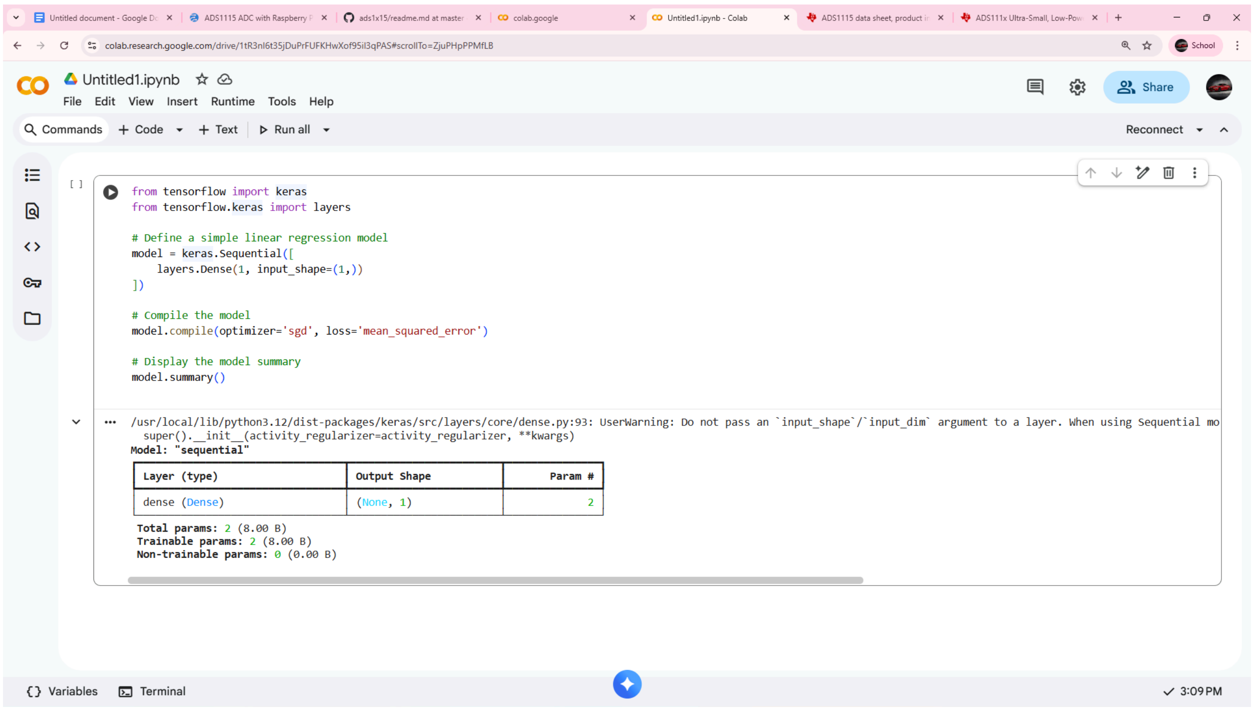
- **Save and close the file.**

  Now, whenever you import Keras in your Python script, it will automatically use **TensorFlow** as its computational backe

---

13. In Google Colab, build a simple linear regression model using Keras and display the total number of model parameters.

Answer:



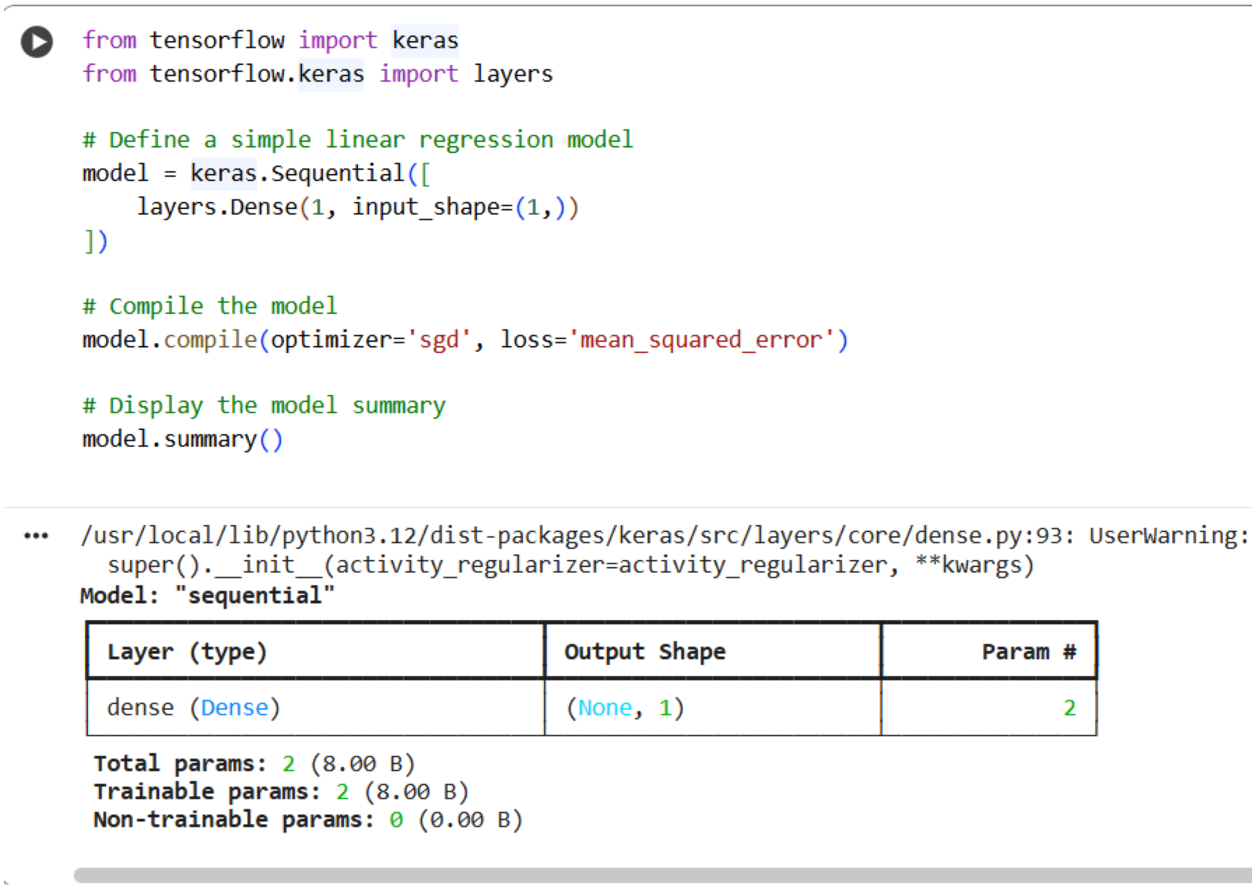
```python
from tensorflow import keras
from tensorflow.keras import layers

# Define a simple linear regression model
model = keras.Sequential([
    layers.Dense(1, input_shape=(1,))
])

# Compile the model
model.compile(optimizer='sgd', loss='mean_squared_error')

# Display the model summary
model.summary()
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential mo
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
┌──────────────────────────┬─────────────────────┬─────────────┐
│ Layer (type)             │ Output Shape        │     Param # │
├──────────────────────────┼─────────────────────┼─────────────┤
│ dense (Dense)            │ (None, 1)           │           2 │
└──────────────────────────┴─────────────────────┴─────────────┘
 Total params: 2 (8.00 B)
 Trainable params: 2 (8.00 B)
 Non-trainable params: 0 (0.00 B)
```

- The model consists of a single Dense layer with one neuron, representing a simple linear regression `y = wx + b`.
- The `input_shape=(1,)` specifies that the model accepts a single input feature.
- The optimizer used is **Stochastic Gradient Descent (SGD)**, and the loss function is **Mean Squared Error (MSE)**.
- The command `model.summary()` prints a detailed summary of the model, including the number of parameters.

Code :

```
from tensorflow import keras

from tensorflow.keras import layers

# Define a simple linear regression model

model = keras.Sequential([

    layers.Dense(1, input_shape=(1,)) ])

# Compile the model

model.compile(optimizer='sgd', loss='mean_squared_error')

model.summary()
```



```
from tensorflow import keras
from tensorflow.keras import layers

# Define a simple linear regression model
model = keras.Sequential([
    layers.Dense(1, input_shape=(1,))
])

# Compile the model
model.compile(optimizer='sgd', loss='mean_squared_error')

# Display the model summary
model.summary()
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━┓
┃ Layer (type)                  ┃ Output Shape           ┃     Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━┩
│ dense (Dense)                 │ (None, 1)              │           2 │
└───────────────────────────────┴────────────────────────┴─────────────┘
 Total params: 2 (8.00 B)
 Trainable params: 2 (8.00 B)
 Non-trainable params: 0 (0.00 B)
```

14. Explain the concept of “Computational Graph” used in TensorFlow. Why is it beneficial?

Answer: A **Computational Graph** in TensorFlow is a structured representation of mathematical operations. It consists of nodes and edges, where **nodes** represent operations (like addition, multiplication, or matrix transformation) and **edges** represent the data — called **tensors** — flowing between these operations.

In essence, when you build a TensorFlow model, you are creating a computational graph that defines the flow of data from input to output through a series of operations.

Benefits of using a computational graph:

1. **Efficient Execution:** TensorFlow can optimize and execute the entire graph efficiently, distributing computations across CPUs, GPUs, or TPUs for better performance.
2. **Parallelism:** Since the graph structure is known beforehand, TensorFlow can identify independent operations and execute them simultaneously.
3. **Automatic Differentiation:** The graph allows TensorFlow to compute gradients automatically during backpropagation, simplifying the training process.
4. **Portability:** Once created, the graph can be saved, exported, and executed on different devices or even deployed in production systems.

---

15. If you need to use TensorFlow and PyTorch for different projects on the same computer, how can you prevent library version conflicts?

Answer: To prevent version conflicts when using TensorFlow and PyTorch on the same computer, the best practice is to create **separate virtual environments** for each framework.

Each environment acts as an isolated container that has its own Python interpreter and libraries.

Bash :
```
conda create --name tf_env tensorflow

conda create --name torch_env pytorch
```

Activate each environment separately

```
conda activate tf_env
```

export and share the environment configuration

```
conda env export > environment.yml
```

Developers can work on multiple deep learning frameworks simultaneously in a stable and organized way.

Lab3 learning reflection or comments :

For the past several weeks, I have been working continuously on the ESP32-based 12 V battery monitoring system, but the progress felt slower than expected. I was feeling a bit low because every time I tested the system, I got slightly different readings from the sensors. Despite many attempts, the data did not stabilize quickly, and it sometimes felt like the project would never be finished.

However, this week I decided to focus on understanding the problem rather than rushing the result. I carefully reviewed the ADS1115 and INA219 datasheets again, checked the wiring connections, and re-calibrated the sensors in Thonny. I also experimented with new resistor combinations for the voltage dividers and adjusted the sampling rate in my code to reduce noise. After several trials, I finally began to see consistent readings from all three Li-ion cells. Seeing the display update correctly for the first time felt very satisfying and gave me back some motivation. I also recorded a full demonstration video showing the system's stable operation and prepared the files for submission.

For **Lab 2**, the experience was quite different. Since my background is in **computer science**, I already have prior experience working with **TensorFlow, PyTorch, and Keras**. I've built small projects with these frameworks before, so revisiting them in this lab felt familiar and enjoyable. The questions helped me recall key concepts like computational graphs, virtual environments, and backend configuration.

Lab 4 Video Link

Note:

This week, I focused mainly on studying and understanding the technical aspects of my project in greater depth — particularly the working principles of the ADS1115, INA219, and ESP32 modules. Because most of my time was dedicated to research, testing, and troubleshooting rather than demonstrations, I was unable to record a video this week. I plan to compile all progress and record a full explanation video once the testing phase is completely stable.