JAI HOO SATNARYAN.

SIX WEEKS INTERNSHIP AT IIT JAMMU

INTERNSHIP REPORT

3rd WEEK

Day 15:

1. Perceptron Trick and Training

On this day, I delved into the foundational concept of the perceptron, a fundamental building block in neural networks. The focus was on understanding the "Perceptron Trick," which is a technique used to train a perceptron. This involves adjusting the weights based on the error in prediction to minimize this error over time. Specifically, I learned the following key points:

- **Weight Adjustment Rule:** The weights are updated using the formula:

$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = \eta(t - o)x_i$$

Where:

- $t = c(\vec{x})$ is target value
- o is perceptron output
- η is small constant (e.g., 0.1) called *learning rate*

- **Iterative Process:** Training a perceptron involves an iterative process of feeding input data, calculating the output, comparing it with the actual output, and updating the weights accordingly.

2. Perceptron Loss Function

A crucial aspect of training any machine learning model is defining a loss function that quantifies the error of the model. For the perceptron, the loss function used is typically the hinge loss or a variant thereof. This function helps in determining how well the perceptron is performing by measuring the difference between the predicted and actual outputs. Key insights include:

- **Hinge Loss :** The hinge loss function is defined as

$$L = \max(0, -y \cdot (\mathbf{w} \cdot \mathbf{x})),$$

- **Optimization Goal:** The objective is to minimize this loss function during training to improve the accuracy of the perceptron.

3. Problems with Perceptron

While the perceptron is a powerful tool, it is not without its limitations. I explored some of the common issues associated with perceptrons:

- **Linearly Separable Data:** A perceptron can only correctly classify linearly separable data. If the data is not linearly separable, the perceptron will fail to converge to a solution.
- **Limited Complexity:** The perceptron is limited in its capacity to solve complex problems due to its linear nature. It cannot model non-linear relationships in data, which restricts its applicability to simple classification tasks.

4. MLP Notation

Building upon the basics of perceptrons, I transitioned to understanding the notation and structure of Multi-Layer Perceptrons (MLPs). MLPs are an extension of the perceptron and form the basis of more complex neural networks. Key components include:

- **Layer Notation:** MLPs consist of an input layer, one or more hidden layers, and an output layer. Each layer contains a certain number of neurons, and the connections between these layers are represented by weight matrices.

- **Activation Functions:** Activation functions such as sigmoid, tanh, or ReLU are applied to the output of each neuron to introduce non-linearity into the model, enabling it to learn more complex patterns.

5. Multi-Layer Perceptron (MLP) Intuition

Finally, I focused on developing an intuitive understanding of how MLPs work. Unlike a single-layer perceptron, an MLP can solve more complex problems due to its layered structure. Important insights include:

- **Forward Propagation:** During forward propagation, the input data passes through each layer, with each neuron applying a weighted sum of inputs followed by an activation function. This process continues until the final output layer is reached.
- **Backpropagation:** The training process involves backpropagation, where the error is propagated back through the network, and weights are adjusted using gradient descent to minimize the loss function.

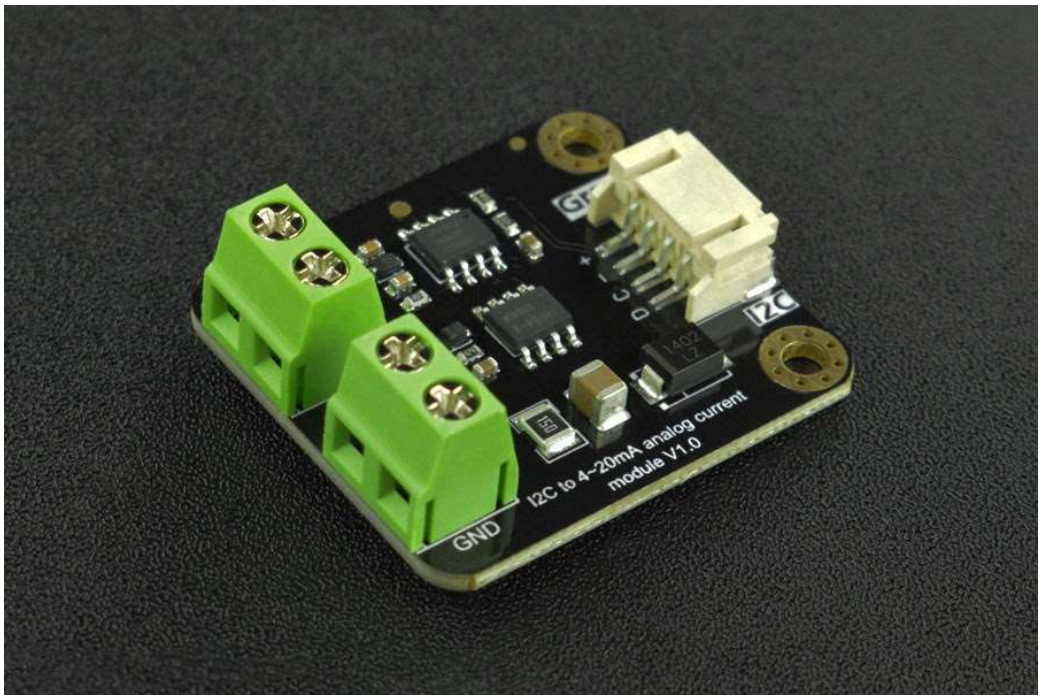
Day 16:

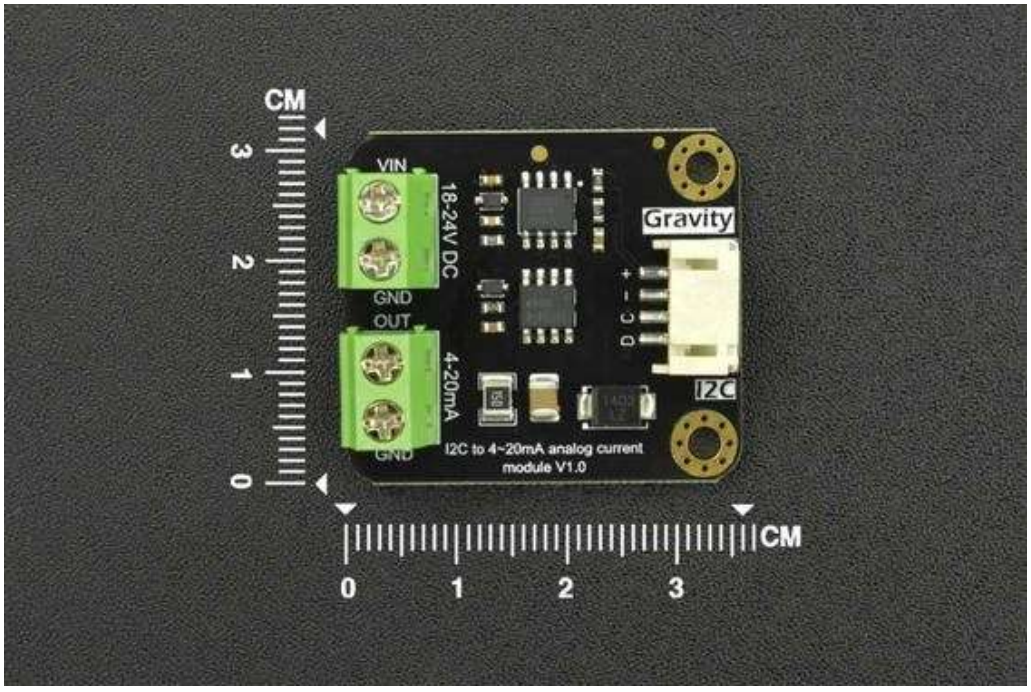
1. Understanding DAC DFR0972

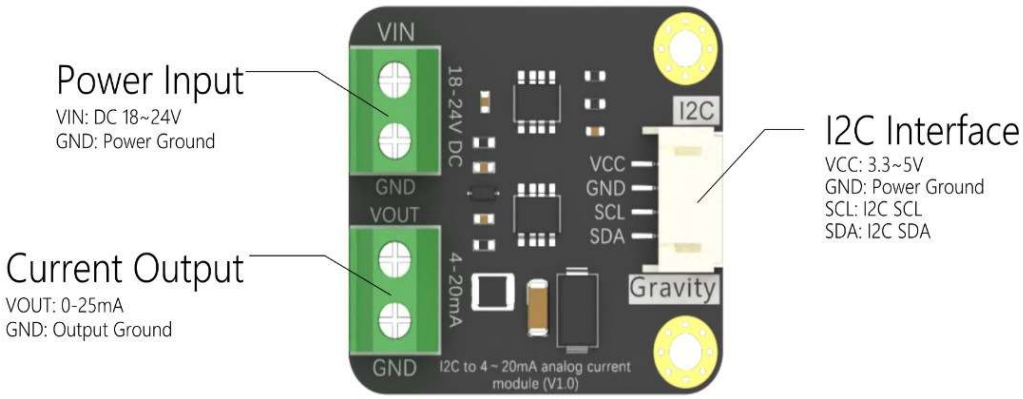
On the sixteenth day, I focused on gaining a comprehensive understanding of the Digital-to-Analog Converter (DAC) module, specifically the DFR0972. The DAC is an essential component in converting digital signals into analog signals, which can then be used to drive various analog devices. Key learnings include:

- **Functionality:** The DAC DFR0972 converts discrete digital data into continuous analog signals. This is crucial for applications that require precise control of analog outputs, such as audio signal generation, waveform synthesis, and motor control.

- **Specifications:** The DFR0972 is a 12-bit DAC, providing a high resolution of 4096 discrete levels. This allows for fine-grained control over the output signal, resulting in smoother and more accurate analog outputs.







2. DAC DFR0972 and Arduino Integration

Understanding how to integrate the DAC DFR0972 with an Arduino microcontroller was a critical part of the day's learning. This integration involves both hardware connections and software programming to control the DAC. Key points covered include:

- **Hardware Connections:**
  - **Pin Configuration:** The DAC DFR0972 has several pins that need to be correctly connected to the Arduino. These include power supply pins (VCC and

GND), communication pins (SDA and SCL for I2C communication), and output pins (AOUT).

- **Power Supply:** Ensuring that the DAC is supplied with the appropriate voltage (typically 3.3V or 5V) from the Arduino is crucial for its operation.
- **I2C Communication:** The DAC communicates with the Arduino using the I2C protocol, which requires connecting the SDA and SCL pins of the DAC to the corresponding pins on the Arduino.

- **Software Programming:**
  - **Library Utilization:** Using the appropriate Arduino libraries, such as the Wire library for I2C communication, simplifies the process of programming the DAC.
  - **Initialization:** Initializing the DAC in the Arduino code involves setting up the I2C communication and configuring the DAC's initial state.
  - **Signal Generation:** Writing code to send digital values to the DAC, which then converts these values into corresponding analog voltages. This involves writing to specific registers within the DAC to control its output.

3. Practical Applications

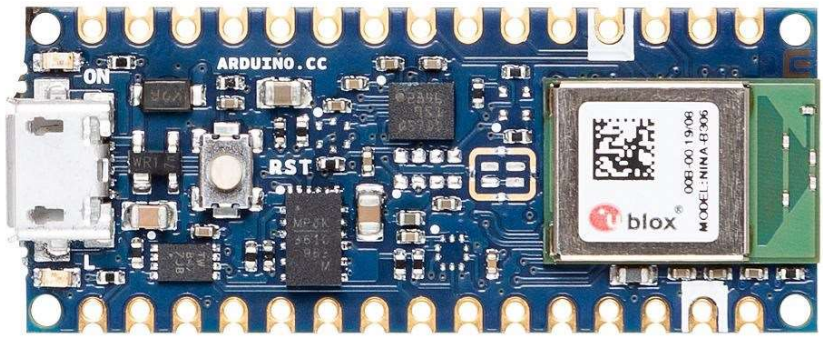
To solidify my understanding, I explored various practical applications of the DAC DFR0972 when used with an Arduino. These applications highlight the versatility and importance of the DAC in different scenarios. Key applications include:

- **Waveform Generation:** By sending a series of digital values representing a sine wave or other waveform types to the DAC, I can generate corresponding analog waveforms. This is useful in audio synthesis and signal processing.
- **Analog Signal Control:** The DAC can be used to control analog devices such as variable resistors, LED brightness, and motor speeds. This enables fine-tuned control in various electronic projects.
- **Sensor Interfacing:** In scenarios where analog sensors are used, the DAC can help in calibrating and fine-tuning the sensor outputs by providing precise reference voltages.

Day 17:

1. Objective

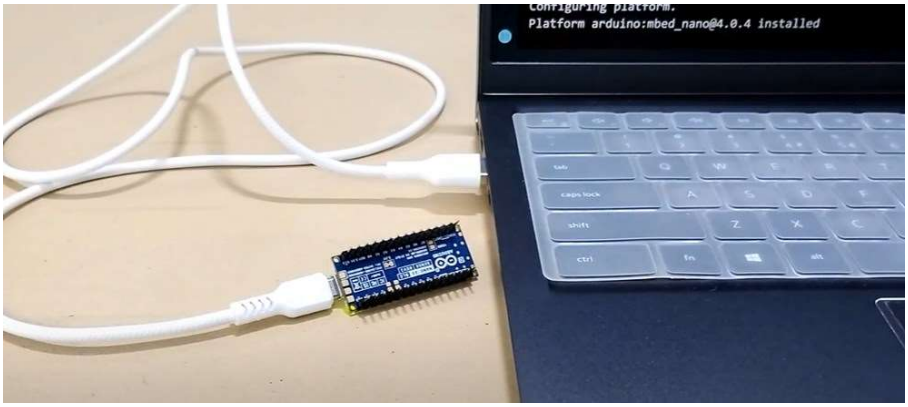
The primary objective for seventeenth was to check the validity of the Arduino Nano 33 BLE Sense Lite after connecting it to a laptop via USB. Additionally, I focused on installing various essential libraries, such as TensorFlow Lite and the Harvard "Hello World" library, and testing how to run programs using these libraries on the Arduino board.



2. Initial Setup and Connectivity Check

To start, I ensured that the Arduino Nano 33 BLE Sense Lite was properly connected to the laptop and recognized by the Arduino IDE. Key steps included:

- **Connecting the Board:**
    - **USB Connection:** Connected the Arduino Nano 33 BLE Sense Lite to the laptop using a USB cable. Confirmed that the power LED on the board lit up, indicating it was receiving power.



- **Arduino IDE Setup:**
    - **Software Installation:** Confirmed that the latest version of the Arduino IDE was installed on the laptop.
    - **Board Manager:** Opened the Arduino IDE and navigated to the Board Manager (Tools > Board > Boards Manager…). Installed the "Arduino mbed-enabled Boards" package, which includes support for the Arduino Nano 33 **3. Library Installation**

The next step involved installing various libraries that are essential for running advanced programs on the Arduino Nano 33 BLE Sense Lite:

- **TensorFlow Lite Library:**
    - **Installation:** Opened the Library Manager in the Arduino IDE (Tools > Manage Libraries…). Searched for "TensorFlow Lite" and installed the "Arduino_TensorFlowLite" library.
    - **Verification:** Checked the installation by opening an example sketch from the TensorFlow Lite library (File > Examples > Arduino_TensorFlowLite > magic_wand).
- **Harvard "Hello World" Library:**
    - **Installation:** Using the same Library Manager, searched for the "Harvard TinyML" library and installed it.
    - **Verification:** Verified the installation by opening an example sketch from the Harvard TinyML library (File > Examples > Harvard_TinyMLx > hello_world).

4. Running Example Programs

After installing the libraries, I tested running example programs to ensure they work correctly with the Arduino Nano 33 BLE Sense Lite:

- **Running Harvard "Hello World" Example:**
    - **Loading the Example Sketch:** Opened the "hello_world" example sketch from the Harvard TinyML library.

- **Uploading the Sketch:** Compiled and uploaded the sketch to the board. Monitored the upload process for any errors.
- **Execution:** Observed the output via the Serial Monitor. The "hello_world" example typically demonstrates a simple neural network model running on the Arduino, displaying results through serial output.

5. Conclusion

This day was successfully completed with the Arduino Nano 33 BLE Sense Lite being validated and various essential libraries installed. I ran example programs from the TensorFlow Lite and Harvard TinyML libraries, confirming the board's capability to execute advanced machine learning models.

Day 18:

Research

Research started on how to generate sine waves using the Arduino Nano 33 BLE Sense Lite with the DAC DFR0972 and visualize them on an oscilloscope.

Day 19:

Research

Continued research on generating sine waves using the Arduino Nano 33 BLE Sense Lite with the DAC DFR0972 and visualizing them on an oscilloscope.

Day 20 and Day 21

Weekend Break

No work was conducted on Day 20 (Saturday) and Day 21 (Sunday). These days were designated as non-working days, allowing for rest and relaxation.