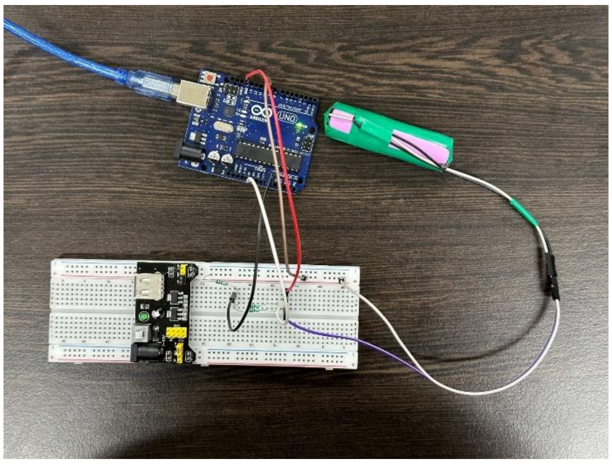
DEP Progress Report

1. Experimentation Setup and Execution

- A 3.6V Li-ion battery was tested by applying a 10mV sine wave perturbation superimposed on its nominal voltage.
- **Circuit Configuration:**
  - Arduino Uno generated a PWM-based sine wave signal (~50Hz).
  - Shunt Resistor (7.5Ω) was placed in series with the battery to measure current response.
  - Analog Input (A0, Arduino) recorded the voltage drop across the shunt resistor.
  - Data was logged via Serial Monitor for analysis.



- **Code**:

```
#include <Arduino.h>
#include <math.h>

const int outputPin = 9; // PWM output pin to apply sine wave voltage
const int inputPin = A0; // Analog input pin to measure voltage drop across shunt
const int sampleRate = 1000; // Sampling rate in Hz
const float amplitude = 10.0 / 1000.0; // 10 mV amplitude (converted to volts)
const float frequency = 50.0; // Frequency of sine wave in Hz
const long duration = 1000; // Duration of signal application in milliseconds
const float batteryVoltage = 3.6; // Battery voltage in volts
const float shuntResistor = 7.5; // Shunt resistor value in ohms

void setup() {
  Serial.begin(115200);
}

void loop() {
  unsigned long startTime = millis();
  while (millis() - startTime < duration) {
    float t = (millis() - startTime) / 1000.0; // Convert time to seconds
    float sineValue = amplitude * sin(2 * PI * frequency * t);
```

```
      float appliedVoltage = batteryVoltage + sineValue;

      int outputValue = map(appliedVoltage * 512 / 5.0, 0, 1023, 0, 255); // Scale to PWM range
      analogWrite(outputPin, outputValue);

      // Measure voltage drop across shunt resistor
      int inputValue = analogRead(inputPin);
      float shuntVoltage = (inputValue / 1023.0) * 5.0; // Convert ADC reading to voltage
      float current = shuntVoltage / shuntResistor; // Calculate current using Ohm's Law

      Serial.print(t, 6);
      Serial.print(",");
      Serial.print(appliedVoltage, 6);
      Serial.print(",");
      Serial.println(current, 6); // Print current value

      delayMicroseconds(1000000 / sampleRate); // Maintain sample rate
    }

    // Simulation (Python-style data visualization code for external use)
    Serial.println("SIMULATION DATA START");
    long totalSamples = (long)duration * sampleRate / 1000;
    for (long i = 0; i < totalSamples; i++) {
      float t = i / (float)sampleRate;
      float sineWave = amplitude * sin(2 * PI * frequency * t);
      float simulatedVoltage = batteryVoltage + sineWave;
      float simulatedCurrent = (simulatedVoltage / batteryVoltage) * 0.05; // Example relation
      Serial.print(t, 6);
      Serial.print(",");
      Serial.print(simulatedVoltage, 6);
      Serial.print(",");
      Serial.println(simulatedCurrent, 6);
    }
    Serial.println("SIMULATION DATA END");
    delay(5000);
  }
```

Data Sample from Serial Output: Reading-2
(Columns: Time (s), Applied Voltage (V), Measured Current (A))

2. Observations and Limitations

- **Voltage Response:** The applied voltage followed a partial sine wave, but PWM approximation led to distortion.

- **Current Measurement:** The current response followed the applied voltage changes, but small fluctuations were observed due to ADC limitations.
- **Limitations Identified:**
    - **PWM is not a true sine wave** – it introduces harmonic noise.
    - **Arduino's 10-bit ADC lacks precision** – small current variations are not captured accurately.
    - **Low sampling rate (~9.6kHz max)** limits impedance measurement at higher frequencies.
    - **Difficult to confirm if measured current is purely from the battery** – additional validation is required.

3. Evaluation of Circuit Approach

- **Why 3.6V + 10mV AC was used:**
    - Keeps the battery in its normal operating range without forcing charge/discharge.
    - Applying only a sine wave would make the battery voltage go negative, which is not valid.
- **Ensuring the Measured Current is from the Battery:**
    - Disconnect the battery and check if current is still detected.
    - Compare DC and AC current separately.
    - Use an oscilloscope to verify waveform shape.

4. Feasibility of Using STM32 with the Same Circuit

- Simply replacing Arduino with STM32G474RE will improve accuracy due to:
    - Built-in DAC for true sine wave generation.
    - Higher-resolution 16-bit ADC for precise current measurement.
    - Faster sampling rate (up to 4 MSPS).
    - Dual ADCs for simultaneous voltage and current capture.
- The existing circuit remains valid, but STM32 will eliminate major limitations**.**

5. Learnings and Next Steps

- The approach of applying a sine wave + measuring shunt current is correct, but hardware limitations affect accuracy.
- Using PWM for sine wave generation is inadequate; a DAC is needed.
- Arduino's ADC is not precise enough for impedance analysis.
- Using an offset voltage equal to battery nominal voltage prevents unwanted charge/discharge effects.

Next Steps:

- Validate this setup by switching to STM32.
- Conduct frequency response analysis for a complete impedance profile.