UC Berkeley, ME100, Spring 2021
Lab 2: Solar cell characterization

Plan to complete these tasks during the week of February 8th
Checkoff due on Gradescope by **Monday February 15th**, 11:59pm

---

Introduction and objectives

In this lab you will use your ESP32 to interface with the physical world for the first time, by logging the voltage and current output from a solar cell under a range of loading conditions. Being able to determine the power available from a photovoltaic source in real time could be useful for IoT applications — for example, it could guide decisions about the best times for a remote IoT device to transmit or receive data, when additional power would be needed for a radio transmitter/receiver.

Specifically, in the lab you will:

- Learn about a current- and voltage-sensing module, the INA219, which connects to the ESP32.
- Learn about the *I2C* protocol which enables the INA219 to communicate with the ESP32.
- Design a simple circuit to connect the ESP32, the INA219, a solar cell, and a variable resistor that will provide a load to the solar cell.
- Modify Python code to log data from the INA219.
- Implement the circuit and code, and test it.

Step 1: Measure I_{sc} and V_{oc} for solar cell

You have been supplied with a solar cell that has two wires soldered onto the back of it. First, you will practice using your multimeter to determine the open-circuit voltage and short-circuit current of the cell under a particular light intensity.

Place the solar cell somewhere where it is receiving reasonably bright light — for example, next to a sunny window or under a desk lamp. Then set your multimeter to the 20 V DC range, ensure the probes are plugged into the “COM” and voltage-measuring ports, and hold the probe tips against the bare metal ends of the solar cell wires. When the reading is reasonably stable, record it: this is the open-circuit voltage, V_{oc} (recall that a voltmeter has a very high input resistance, so the meter is effectively operating as an open circuit).

Next — without changing the position or illumination conditions of the solar cell — set the multimeter to the 20 mA or 200 mA DC range, and again hold the probe tips against the wires of the solar cell. Record the current. Choose the current range that gives the largest number of significant figures for the magnitude of the current you are recording. Recall that an ammeter has very low resistance, so your meter is now approximating a short circuit and you are recording I_{sc}.

Record your I_{sc} and V_{oc} values in the Gradescope assignment for this Lab. Optionally, for interest you could try repeating the I_{sc} and V_{oc} measurements under one or more different illumination conditions.

Step 2: Build measurement circuit

Before building a circuit it's often a good idea to make a diagram of how you will connect the components so that you have its function clear in your mind. Draw a circuit in which the electrical energy supplied by the solar cell is dissipated by a 1 kΩ potentiometer, and in which the INA219 measures the current through the potentiometer and the voltage across it. Upload an image of your design to Gradescope.

Then assemble your circuit on your breadboard by combining the ESP32, solar cell, INA219 board, and potentiometer.

Hints:

- Make sure you are connecting to two pins of the potentiometer between which the resistance will actually vary as you turn the knob.
- The headers and terminal block of your INA219 board have been pre-soldered for you. The screw terminals and the header pins labeled "Vin+" and "Vin–" do the same thing.

Step 3: Prepare code

Transfer ina219.py to ESP32

The functionality needed for the ESP32 to interact with the INA219 is not built into the micropython firmware that you've already flashed to your board. Instead, we need to provide an additional module.

We are going to use a pre-written module from this source. To do this, download `ina219.py` from bCourses (Files > Labs > Lab2) and, in `shell49` with your board connected, copy it over to the board as follows:

```
cp ina219.py /flash/ina219.py
```

Then start a REPL and confirm that you can import the module — i.e. check that the following command does not return an error:

```
import ina219
```

Now open `ina219.py` in your text editor and study it. You do not need to understand its operation in detail to complete the lab, but it is worthwhile spending a little time appreciating its main features, and the parameters that can be controlled, including the voltage range, the *shunt* (i.e. current-sensing) resistance, and the number of bits to be used in the analog-to-digital convertor.

You may also wish to spend a little time reviewing the INA219 datasheet here.

Here are some questions to consider about the INA219 board, and the I2C protocol that is used by the INA219 board to communicate with the ESP32:

- Why do I2C-connected devices need addresses?
- How does the ESP32 identify the INA219?
- What would happen if you connected multiple INA219s to the same I2C bus?
- What does the `configure()` method in `ina219.py` do?

Modify measure_solar.py and test

Download the skeleton file `measure_solar.py` from bCourses (Files > Labs > Lab2). It contains most of the commands you need to connect to the INA219 and log measurements from it, but you need to complete the code. Modify the `while` loop to print one line to the terminal every ~0.5 seconds, in which that line reports the current being delivered to the load, the voltage across it, the resistance of the potentiometer at that moment, and the power being absorbed by the potentiometer. Make sure that your modified `print()` command formats its output so that the variables and their units are clear to the reader.

Then connect to your board and use a `run` command in `shell49` to test your modified `measure_solar.py`.

As you run your code, slowly turn the potentiometer knob and confirm that the computed resistance changes. Look at how the voltage, current and dissipated power change as well.

For a couple of positions of the potentiometer knob, check the voltage values being logged by the INA219 by placing your voltmeter across the potentiometer terminals.

- What is the percentage discrepancy between the voltage recorded by the voltmeter and the INA219?

Finally, insert the ammeter inline between the "Vin–" terminal of the INA219 and the potentiometer. Now, for a couple of positions of the potentiometer knob, compare the current value logged by the INA219 with that measured on the ammeter.

- What is the percentage discrepancy between the current recorded by the ammeter and the INA219?

Also, consider the resolution of the voltage and current measurements you are able to make with the INA219. If necessary, refer to `ina219.py` or to the documentation.

- What are the voltage and current resolutions of the INA219 board in the configuration in which you are using it?

Hints:

- If you run `measure_solar.py`, then stop it (e.g. by pressing Ctrl-C twice) and attempt to run the code again, you will probably see an error saying that the I2C bus is already used. To rectify this it will usually be necessary to exit `shell49` and restart it, or possibly to reset the ESP32 board.
- Note that the `current()` method in `ina219.py` returns current in mA, not amps. You will need to remember this when computing the load resistance and power absorbed.

Conclusion, and looking ahead

You will notice that the power absorbed by the resistor changes very substantially as you vary its value. If you wanted to extract the maximum possible power from the cell, you would need to select the optimal load resistance value. With your system as it stands, it can be hard to find that value from the purely text-based output. You could manually plot a graph of power absorbed against resistance, but this would be laborious. A way of automatically graphing power against resistance would be beneficial. In Lab 4, you will extend this application to transmit the current and voltage readings over the Internet using the MQTT protocol, and then use your computer to plot them.