SS-9 Resistivity and Hall Effect in a Semiconductor

Saba Karimeddiny

November 13, 2018

1 Information

The password to the laptop is: SS-9

2 Aim

The aim of this manual is to provide you with enough information about the software I have written so that you may successfully complete the experiment. I will also (hopefully) explain some of the essential pieces of the software so that you may troubleshoot any minor bugs that may arise. As long as the code is not tampered with, the software should run smoothly. In the event of an unfixable[1] bug, please contact me via email at sk2992@cornell.edu. This manual is not intended to introduce any theory or give any hints towards completing the lab as there are plenty of other resources offered to you by the instructors.

3 Acknowledgements and Unsolicited Advice

The backend software that makes the automation of this lab possible is PyMeasure, a Python API developed here at Cornell by members of the Ralph Group – primarily: Colin Jermain, Neal Reynolds, Joseph Mittelstaedt and myself. PyMeasure is an amazing, modular Python package that makes it possible to automatically generate a customizable GUI that performs a prewritten procedure. The procedure can control any set of instruments that is used for a particular lab.

If you are an aspiring experimentalist, please consider learning and using PyMeasure as automating your experiments can save you countless hours in the lab. To get started, visit the PyMeasure github page

https://github.com/ralph-group/pymeasure

and read the documentation

https://pymeasure.readthedocs.io/en/latest/.

[1] to me, this means you have spent *at least* one whole lab period trying to fix it

4 Brief History

This is a great experiment. I myself completed this experiment in the Fall of 2018; it was fun, but to take the data I had to take (literally) hundreds of pictures of the multimeter screens with my iPhone and transcribe them into my computer, which is boring and unenlightening. Modern experimental physics is not done this way because people take time to automate things.

5 Things you will need to have (installed) on this PC

These things should be installed already, but in the event of a fatal error here is what you should have installed to run the code:

- Anaconda Python distribution
- Various Code from Measurement Computing Inc.
- PyMeasure
- My code: https://github.com/sabakarimeddiny/CornellPhys510_SS9

If a clean install is required, download my code and the version of PyMeasure from the github link above. Install the version of PyMeasure in my git repo by navigating to the PyMeasure folder in Anaconda prompt and running

>>> python setup.py install

6 How to break things

These are things that you don't want to do!

- Although PyMeasure is available on github, pip, and conda forge, DO NOT reinstall or replace the PyMeasure version that is currently installed on the computer UNLESS it is the version available on my personal github page. The version of PyMeasure on the computer actually contains an altered line of code that has not been committed to the git head as of Fall 2018. Leave the backend alone.
- DO NOT update the Anaconda/Python distribution. This software is verified to work with the current versions and updates might break things.
- DO NOT edit the original python scripts found in the "CornellPhys510_SS9" folder on the desktop. These are the files as I have written them. They work. If you feel confident in your Python abilities and want to mess around with the code, COPY it to another directory and work with that. You can then discuss with me/instructors about replacing my code with yours.

- DO NOT overwrite or use someone elses data file, your instructors *will* know if you plagiarize your data. Please create a new directory for yourself and input that save directory in my software.

7 The Apparatus

By now you have hopefully familiarized yourself with the experiment and instruments. The equipment relevant to this manual is:

- Laptop (the password is: SS-9)
- 4 × Keithley 2110 Digital Multimeter
- Copper Constantan ThermoCouple & Banana plug to USB DAQ

8 Communicating with the Instruments

I will discuss how you may communicate with each instrument in iPython. This section could prove very useful if you need to troubleshoot. You **will need** to get the digital addresses of the multimeters and figure out which multimeter corresponds to which address. To get started you will want to open an instance of Anaconda Prompt. Once open, type "python" and hit enter; this will open iPython. You should see lines terminated with >>>

8.1 Keithley 2110 Digital Multimeter

The Keithley 2110 digital multimeter is an excellent tool that happens to work with pre-existing PyMeasure drivers for the Keithley 2000. I will help you find the addresses of the Keithleys, but you should figure out which one corresponds to which address (use a battery or something). Once you have iPython open, type:

```
>>> import pymeasure
>>> from pymeasure.instruments.keithley import Keithley2000
```

This loads the drivers for the Keithley. Next we will have to help Python communicate through the USB busses. In the same iPython instance type:

```
>>> import pyvisa
>>> rm = pyvisa.ResourceManager()
>>> rm.list_resources()
```

This should show you a comma separated list of Keithley 2110 device addresses. There should be FOUR items. Sometimes only three appear. If this occurs unplug and replug the Keithley 2110 usb cables and repeat last two lines of code above until you see four devices. Once there are four devices, you can create a device object by typing:

```
>>> multimeter = Keithley2000("address")
```

where "address" is one of the addresses that appears after rm.list_resources() and the quotation marks are *required*. You can now run something like:

>>> multimeter.voltage

and something should happen. Ideally, a voltage is measured. All other commands can be found on the Keithley 2000 PyMeasure documentation. Use the above code to figure out which digital address corresponds to which multimeter, record this somewhere as you will need it later.

8.2 ThermoCouple and USB DAQ

Open Instacal and make sure that it detects the USB DAQ. Right click on the device, click test > analog and then measure a temperature. You may have to recalibrate the thermocouple; this I will leave up to you. Start iPython and type:

>>> import mcculw

>>> from mcculw import ul

>>> from mcculw.enums import TempScale

This loads the backend for the USB DAQ. To read the temperature, type:

>>> ul.t_in(0, 0, TempScale.KELVIN)

You should get the correct temperature reading in Kelvin.

9 How to use my software

Using my software is very easy if nothing is broken, and the data collection should be a breeze. To begin, click the icon on the desktop that says SS9Run. A Python GUI will appear. You will immediately notice the boxes on the left; most of these are self-explanatory, but I will explain them anyway.

- Save Directory – The data file will be output into a dated folder inside the save directory.
- Your Name – The output data file will be prefixed by your name, which will (redundantly) ensure unique filenames and no overwriting.
- field_strength – This is to note the strength of the magnetic field in T. It DOES NOT control the magnet, but is only intended to keep a record. Write this information in your lab notebook just to be safe.
- I_bias – This is to note which way the current (the switch) is biased forward (checked/True) or reverse (unchecked/False). It DOES NOT control the current switch, but is only intended to keep a record. Write this information in your lab notebook just to be safe.
- delay – This sets the delay between data point collection events. Once cooled down, the experiment takes a few hours to scan the temperature range that you would like to probe, if you set the delay too low (the minimum is 1s), then you will have a ton of data points; too high and you might have too few.

- T_max – This sets the maximum temperature reading in K that the thermocouple has to read to end the data collection. If you set it too high, you may never reach it; too low, and you might miss important phenomenon. Consult the lab information to choose an appropriate T_max. While the software is set up for you to be able to leave the apparatus during data collection, you *must* set a timer for yourself to return and shut off the heater. Letting your sample get too hot can degrade or even break it. I recommend you babysit the experiment the first time to have a rough idea of how long the warmup takes, then set a half-hour-shorter timer for yourself on the next runs.

- MMX Measurement – This sets the type of measurement you want the multimeter X to perform. You can measure voltage (V) or current (I). The MMX Address below it will control which multimeter you are selecting.

- MMX Range – This will set the range of the multimeter. For sensitive, small signals keep it low. For larger signals, keep it high to avoid railing. The MMX Address below it will control which multimeter you are selecting.

- MMX Address – This is where you will input the long digital address string that you hopefully found via iPython. Include the whole address *without* quotes.

The "queue" button queues a run. Once you are satisfied with your entries in the boxes, queue a run and the data collection will begin. I don't recommend queuing multiple runs for this experiment because I think it would be pointless.

After the first queue, the data collection will automatically begin and the data will be plotted in real time so you can monitor the progress. In the top right dropdown menus, you can select what quantity you want plotted on the x- and y-axes. You can choose which multimeter readings to see. If you queue up another run after clicking abort, make sure to click the "resume" button.

Have fun!