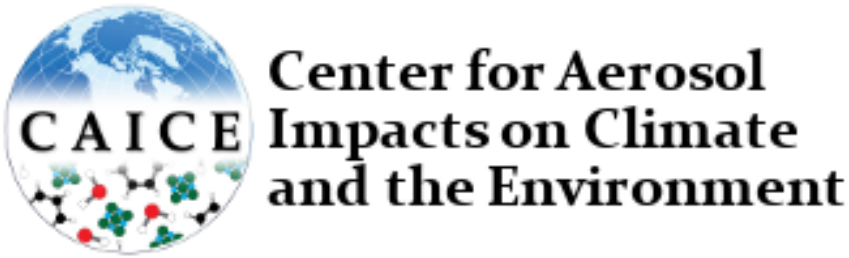
Project D - Constant Current Source for Coulometry

Mentors: Skip Pomeroy & Frank Cardone (Calit2)
Members: Nick Ben, Yuan Liu, Louis Muldrow, Nayyara Shaik, Bastien Levadoux

December 11, 2015



Executive Summary

Our team's goal for this project was to build a coulometer device from parts given to us by our mentor, Dr. Skip Pomeroy, in an already existing kit. The coulometer device is built using these kit components and our group's role was to replace an analog switch and multimeter, in the coulometer device setup, with a digital switch and mobile application respectively. The coulometer device is used to assist students in Dr. Pomeroy's chemistry class, to measure current and voltage readings from a base + acid titrated solution, to establish the concentration of electrons. Voltage and current measurements being read from the solution will assist students determine charge, which is used to determine the concentration of electrons. The improved coulometer system will streamline this data collection process for students as it enables students to turn On/Off the device as well as collect voltage, current, and charge data using our mobile application. The improved design also successfully reduces costs for Dr. Pomeroy.

We began our approach by first reading through the technical papers on the coulometer device experiment, provided to us by our mentors, and figuring out how the device worked and was built. Using a schematic from the technical paper, we built the physical device, and decided we needed to replace the multimeter in the system. The multimeter was being used to display voltage and current readings collected from the titrated solution. To be able to display these readings on a mobile application, we needed equipment to communicate between our physical device and the mobile application. A microprocessor board, the Arduino Uno, was best for this with it's ADC (Analog to Digital Converter) capability and additionally a Bluetooth Module was required to enable the coulometer device to communicate with the mobile application.

After the Arduino Uno microprocessor and Adafruit Bluetooth Low Energy Module (BLE) were decided, our group programmed the microprocessor using Arduino open source software, in which many libraries were made available to us. Simultaneously the mobile app was also being developed on the android platform, and we were fortunate enough to have some source code, provided to us by Adafruit (the manufacturer of the Bluetooth module), that enabled our mobile application to communicate with the BLE. Once programming the microprocessor, developing the android application and figuring out the wiring between the microprocessor, coulometer, and BLE were finalized we were able to successfully record virtually the same current and voltage readings that were read from the multimeter when it was a part of the system. There was only a slight discrepancy of ~0.05%, that was considered negligible.

Furthermore, we needed to establish how to enable the coulometer to be turned on from the mobile application. The original system had an analog switch, and we wanted to remove this so the user could just press an On/Off button in the application. An NPN BJT transistor was decided upon and was hooked up between the microprocessor and the coulometer device.

Finally, after replacing the switch and removing the multimeter from the system, we tested the system with a load resistor that simulated the solution that was to be measured from during the class experiment. The device was tested in a controlled environment and data was recorded to ensure that the system was working appropriately.

Table of Contents

List of Figures:

1. Introduction

1.1 Background

One of the mentors for the project, Dr. Skip Pomeroy, instructs a chemistry course, in which a number of experiments throughout the quarter are conducted. One such experiment is the titration of ~0.1 Molar HCL/NaCl solution by coulometry. A kit containing parts to build a coulometer device, for the purpose of collecting current and voltage readings from the titrated solution, is given to each student. The present setup for the coulometer device + solution experiment entails using an analog switch and multimeter for turning On/Off the system and displaying data readings respectively. Our improved design will enable students to turn On/Off the coulometer device from their smart phone and have current & voltage readings displayed on their application. By replacing the analog switch and the multimeter, with a digital switch and such that readings are now displayed on the Android application respectively, we have streamlined the process of data collection for the students and have cut back on costs to run the experiment.

1.2 Statement Of Work (SOW)

Over the course of the ten weeks we are to develop an android application, program our Arduino Uno microprocessor, and replace our analog switch with a digital switch. The end product should be able to be turned on from a person's smartphone and real time data should be sent directly to the mobile application for the user to see. The device should be tested in a controlled environment with a load resistor used to simulate the titrated solution.

The circuit should be setup on a breadboard for testing purposes, and we will use our Arduino board to build a circuit that will have the Bluetooth Module (Adafruit BLE), and coulometer device all wired such that we can communicate between the physical device and software (mobile application). When the circuit is assembled, the analog switch will have been replaced with a NPN BJT as a digital switch. The user will then be able to turn on the device from his or her phone and collect real time data on the application. The voltage, current, and charge data should then enable the user to determine the concentration of electrons within the titrated solution, as per the purpose of the 'Titration of the Solution by Coulometry' experiment.

1.3 Project Approach

When reflecting on the constant current source for coulometry device, we knew we had to first figure out how to build the device. We also knew from the SOW that we had to more importantly determine what we wanted to use to replace the analog switch and multimeter in the system. We needed to decide what component we would use in place of the analog switch and also how our physical coulometer device would have the ability to communicate with the Android application to which data will now be displayed on. We decided that an NPN BJT would be the most optimal replacement of the analog switch in which we can drive a voltage/current at the base of the transistor to switch the device On or cut the voltage/current from the base to turn Off the the device.

As for the microprocessor our mentor suggested we use a 'Blue Giga.' We eventually decided that an Arduino Uno board would be best because it was made readily available to us, the Analog to Digital Converter (ADC) capabilities, the hardware modules that were easy to remove and add between the board and also it's ability to communicate with a BLE we decided upon. Furthermore some members of the group were familiar with coding in an Arduino environment and no one had any scripting experience that the Blue Giga required.

Concerning the bluetooth module that we also knew that was required of our system to enable communication between phone and device we decided on a BLE (bluetooth low energy) manufactured by Adafruit, for a number of reasons. It was able to communicate with our Arduino Uno microprocessor and android application, and had some readily available code that assisted us in being able to connect to bluetooth via our mobile application.

While the microprocessor was being coded we simultaneously developed an Android application. We first began by using the source code that enabled our application to connect to bluetooth. From there we were able to add buttons to Start/Stop/Resume/Pause the data collection. Furthermore we made a list view on the application interface to enable the user to see current, voltage and charge readings. Dr. Skip Pomeroy asked that we make the application such that it will automatically stop after running for 300 seconds. In turn data collection must happen every second, and the user has the option to Start and Stop the experiment at any point within the 300 seconds allotted. Furthermore the Pause/Resume buttons enabled the user to cut off the voltage being applied to NPN BJT or drive it respectively, if the user wanted to turn On/Off the coulometer device.

For wireless communication, we decided on BLE because we weren't storing a significant amount of data and the power consumption was much less than your standard bluetooth module. Furthermore the range of BLE was similar to the standard bluetooth as well.

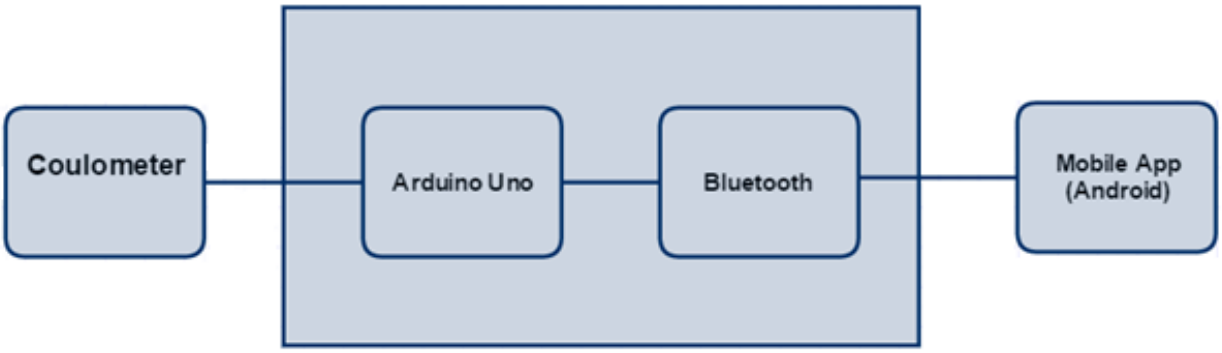


Figure 1: System's block diagram

2. Technical Background

2.1 Constant Current Source

The constant current source was a circuit we were required to build given a list of several different electrical components. The electrical components we used were: 9-V AC adapter, an analog switch, 4.7µ capacitor, diode, 10 ohm resistor, LM317T regulator, Heatsink, and a terminal strip. The constant current source is used to provide a constant current to a system of unknown resistance. The reason why this particular circuit is called a constant current source is because the current of this circuit will not change if we add a finite random resistance to the circuit of the constant current source. This is a very important characteristic of the constant current source because the resistance of the chemical solution is unknown. We based our design of the constant current source on figure 2a, a diagram given to us by our mentors. The letters in the diagram correspond to the electrical components we listed earlier in this paragraph.

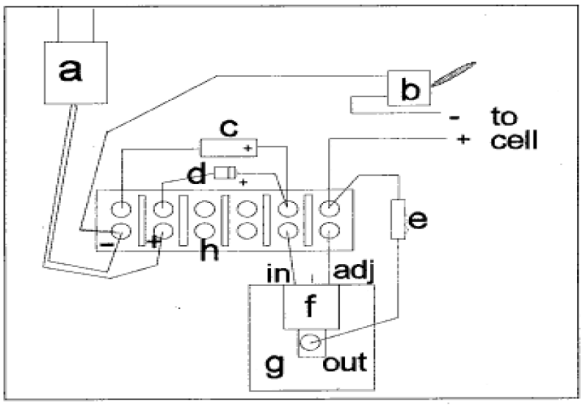


Figure 2.a: Diagram of constant current source

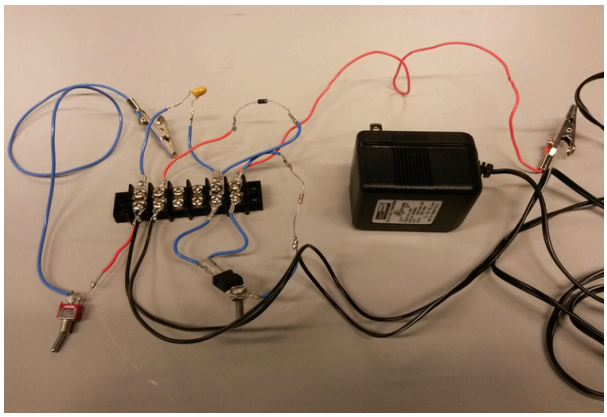


Figure 2.b: Constant current source we built

The initial problem with the constant current source was that for chemical solutions that were 40 ohms or less, the current created by the constant current source wouldn't be 16 mA consistently, it would vary depending on the resistance of the solution. This is because the combined resistance of the 10 ohm resistor and the unknown resistance of the solution (unknown, but less than 40 ohms) was not enough to keep the current constant. We made a diagram of the constant current source in Pspice, as seen in figure 3, to simulate the circuit and see how the current changed as the resistance of the solution varies.

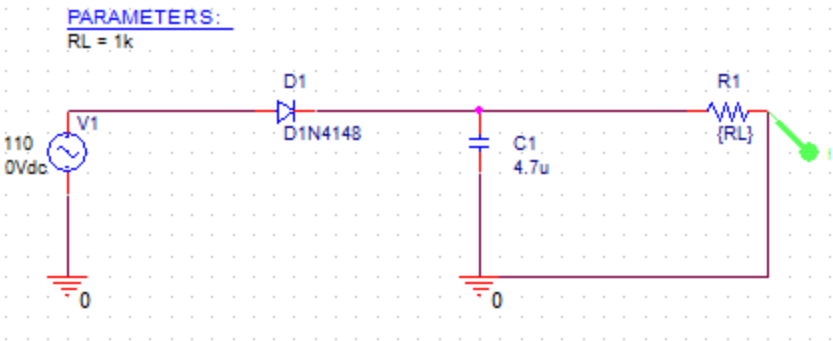


Figure 3: Constant current Pspice circuit

The resistor R1 in figure 3 is the combined resistance of both the 10 ohm resistor given to us by our mentors and the resistor of the unknown solution. We varied this resistor from 1 ohm to 210 ohms and watched the current change as the resistor varied in the graph below.

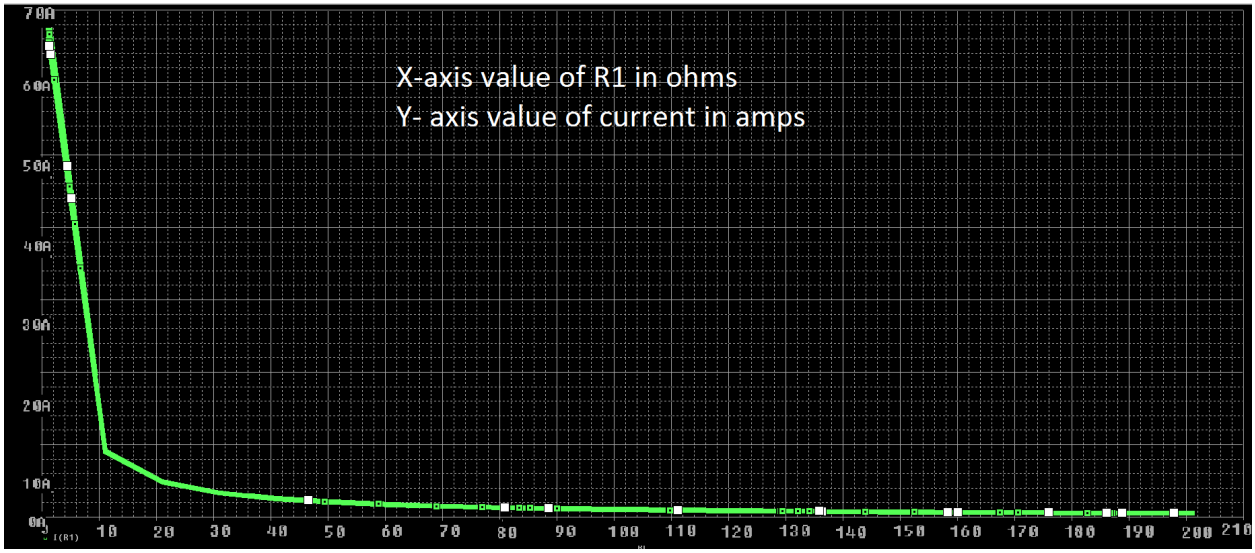


Figure 4: Graph of current with varying resistor values

From the graph in figure 4 you can see the current becomes approximately constant, with less than 1% variation, beyond 50 ohms, this met the requirement provided to us by our mentors. This meant that any resistor 50 ohms or greater will give us a constant current for any finite resistance placed in the same circuit with the constant current source.

2.2 Electrical Components

Power Source

The power source of our circuit is a 9-V AC adapter given to us by our mentors. The power source has a DC maximum voltage of 9 volts and a maximum AC voltage of 110V.The power source of the Arduino UNO is different from the 9-V AC adapter of the constant current source, the Arduino UNO power source is the computer or laptop it is connected to. The 9-V AC adapter is also used to bias the npn bjt.

Heat Sink

In the circuit we also have a passive heat sink. A passive heat sink is a device in the that keeps a certain component of your circuit from overheating. A passive heat sink is not connected to a fan or a Peltier device to keep the components from overheating. The heat sink also reduces noise caused by arbitrary factors in the circuit. In our circuit the heat sink is used to keep the regulator from overheating, the overheating of the regulator can be caused by the charge delivered to the regulator from the 75 ohm resistor.

LM317T Regulator

The voltage regulator is used to maintain a constant voltage on the circuit. The regulator stabilizes the DC voltages in our circuit by balancing the voltage fluctuations in the circuit. The regulator draws in the maximum amount of current in our circuit and multiplies it by the voltage difference between input and output pin of the regulator. The regulator acts like an op amp by multiply the voltage difference by some gain value, in our case the gain value is current.

Diode

The diode provides a minimum voltage for the circuit to work. If the input voltage is too small the diode becomes reverse biased, restricting current flow completely throughout the whole circuit. If the current flow is zero then the circuit will not operate. This benefits our circuit by not allowing negative voltages to influence our circuit, this can be seen in figure 5 below. This changes the sinusoidal input our circuit sees into a signal with a minimum of zero or greater.

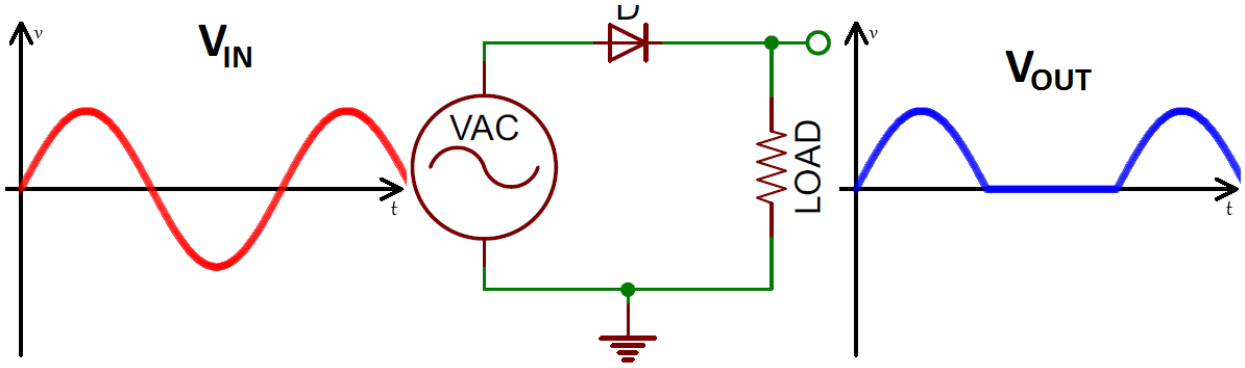


Figure 5: An example of the diode's effect on sinusoidal input voltage

4.7μ capacitor

The capacitor in our circuit is used as a voltage differentiator of the voltage output of our diode. This gives us a constant current when we differentiate the Vout sinusoidal signal seen in figure 5. Without the capacitor the current would vary with the voltage output of the diode.

2N3904 NPN BJT

The npn bjt is used as a switched in constant current source circuit. It controls the current flow of the circuit by a base voltage and a base current. We used a 47 ohm resistor to correctly bias our input current and voltage to the proper values. The npn bjt has a maximum base current of 100 mA and a minimum base voltage of 0.6 volts. This means the npn bjt will only turn on if the base voltage is above the 0.6 volts or the base current is 0. The maximum base current is the biggest base current the npn bjt can take without overloading. If the input voltage is below 0.6 volts or the base current is zero or less, the npn bjt with be reversed biased. This means that the npn bjt will restrict current flow and act like an open circuit.

75 ohm Resistor

The 75 ohm resistor is used to insure that the constant current source keeps a constant current. If the overall circuit resistance falls below 50 ohms the current value will fluctuate, we fixed this problem by attaching a resistor so big that current will remain approximately constant (less than 1% variation).

2.3 Arduino Code

The arduino in our project is designed to feature as a bridge between titration circuit and Android. It adopts the standard libraries of uART BLE. We developed our code on the basis of these libraries. According to the aim of our project, that is, to send the readings of the voltage, current and charge to Android, we have figured out the inputs and outputs of our arduino board. In order to make full use of the onboard resources, we added several features.

Titration Circuit's Voltage

We read the voltage across a sample resistor (65 Ω) in the titration circuit and then convert it to current. According to the relation:

$$I = U/R_{sample}$$

And as we have set the time interval for each sample, we have the charge for a sample:

Because o$Q = \int_{T_1}^{T_2} Idt$ur device will only read discrete voltage, thus we have converted the integral to sum:

$$Q_i = I \bullet T_{sample\ interval}$$

Then we sum it up to figure out the total charge:

$$Q_{total} = \sum_i Q_i$$

The voltage reading technique is quite simple. It uses two of the six analog pins to read the voltage of the sample resistor denote them as V_1andV_2, then it will be converted to a 10-bits digital value with the maximum of 5 volts. There is a little discrepancy in the reading, but it can be eliminated by using a two-way reference method or using a reference voltage.

Bluetooth Signal

Data conversion part executes in every loop if there is any data in the BLE buffer to send to the cell phone. Because we use a command-line-like protocol to communicate between Arduino and cell phone, the float type variable of the voltage needs to be converted to a length-specified string type. The length is limited to 20 bits.

Command reading part uses a extendable design. For each kind of command, a unique command ID is assigned to it in order to convert the string type command into an int type. Under this kind of design, there will not be any alias between commands. And we uses an upper-case X to denote the command sent from cell phone to distinguish it between data and commands.

Bluetooth controlling is used to control the data sending process. We designed the loop to be about 1 second per loop and the loop will run 300 seconds by default. Thus, in this part there are counters to control the loop iteration times. Apart from that, this part also have different subcases defined by the command ID. Under this kind of setup, 85% of the Arduino Uno memory is used and will decrease the CPU load of the cell phone.

Digital Switch

The code implement of digital switch is quite simple as well. Two pins of the arduino ports are used. The base and emission electrode are controlled by arduino. When the voltage between these two electrode is +5V, the BJT will let current pass from drain electrode to emission electrode.

2.4 Android App Development

During the development of the mobile application, one of our major concerns was to find a platform most compatible with our BLE, Adafruit. While both Android and iOS were compatible, we chose Android platform for a lot of important reasons:

1. Low barrier of entry for Android: In order to develop for iOS devices, a developer must use a Mac. However, Android app development can be done on Windows, Mac and Linux.
2. Knowledge of Java: Android development majorly requires Java for development, which is a proven and powerful programming language, used on a wide range of devices and operating systems. Developing for iOS, on the other hand, requires that you learn one of Apple's development languages (Objective C or Swift). Both of these languages are really only used for Apple-centric development (iOS and OS X), and the skills needed to develop in these languages cannot be carried over to other operating systems.
3. Unlimited Resources for BLE: We found a lot of valuable resources for Android App Development online that helped us jump-start our app.

We used Android Studio to develop communication between an Android phone and Arduino Uno. Android Studio is an IDE designed and developed specifically for Android app development. With this we were able to view live-layouts with real time app layout rendering and also preview a layout on multiple screen configurations while editing. Sending data wirelessly from Arduino Uno to the phone was relatively easy with just a few lines of code. We stimulate the UART device beneath the surface of the BLE, sending ASCII data back and forth between the devices, giving us the liberty to decide what data to send and what to do with the data on either end of the connection. We are able to send and receive data upto 10 meters away from our Android device. The following figures show the steps involved in successfully connecting the app and the Bluetooth module.

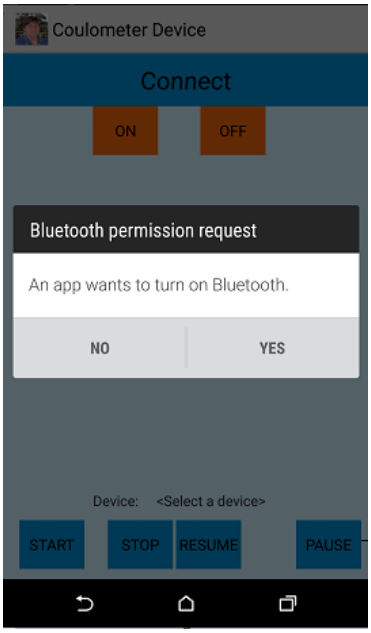


Figure 6.a: Request Permission

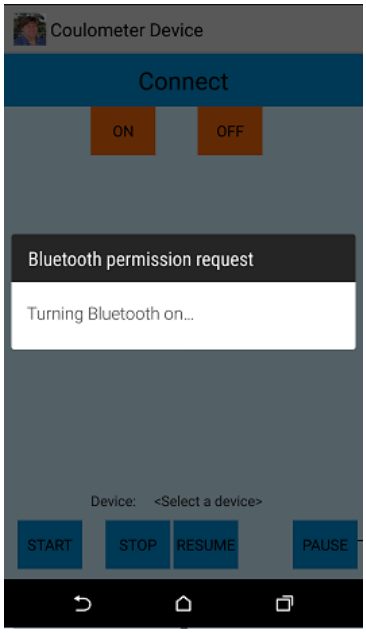


Figure 6.b: Turn Bluetooth ON

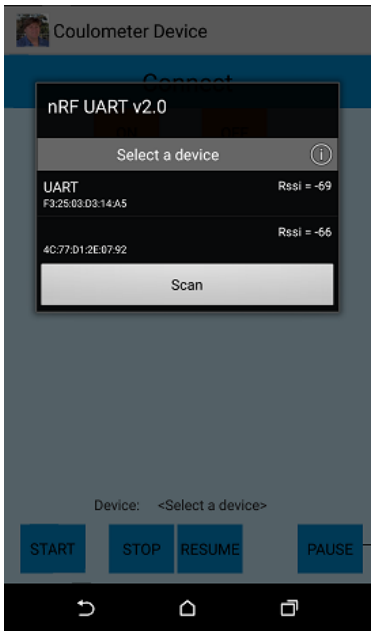


Figure 6.c: Connect to appropriate BLE

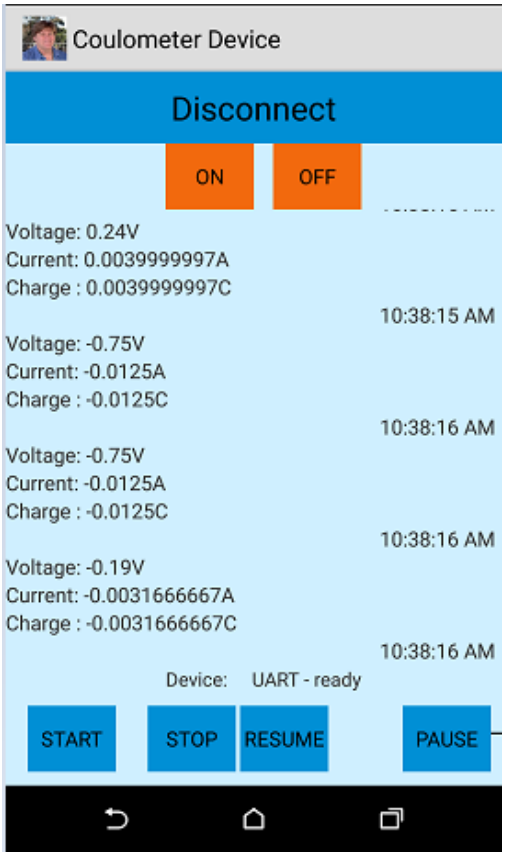


Figure 7: Display Readings

The app has been designed to primarily display Voltage, Current, and Charge of the solution. The app is able to connect to the appropriate Bluetooth device and notify the users of the connection status, turn the device ON or OFF completely with the touch of a button, Start button to start collecting the readings, Pause and Resume buttons that pause and resume the experiment, and Stop button that stops collecting readings.

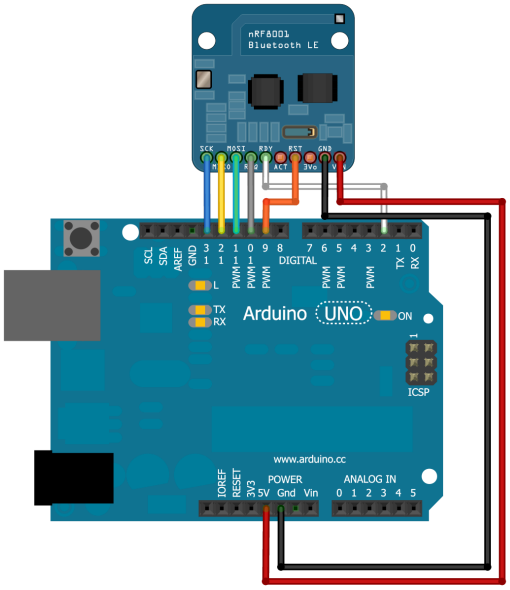
3. Current results

The project is currently at the state of a Breadboard, which can retrieve data generated by the constant current source and the solution in which the probes are plunged. This system generates a variable Voltage that is read by the Arduino.

Once the arduino module and the user's phone paired via bluetooth, the current source can be turned on and off from the phone application we developed. As soon as the user turns the data communication on, the readings are sent to the phone through the Bluetooth Low Energy module, which requires controlling in both of the Arduino and Android codes. Finally the user's phone application displays the different readings with Timestamps and keep them in memory so the user can analyse and use the collected datas.

The control is made to be quick and intuitive with a clear display and controls given to the system with a single click.

3.1 Arduino/Bluetooth module/Current source setup



The Bluetooth Low Energy setup as seen on the figure 8 is made on a breadboard to enable more modularity (such as the implementation of the digital switch).

The constant current source is set up as seen on figure 2.b and figure 9 shows the system of the whole experiment setup.

Figure 8: Arduino Uno and BLE module Pinout

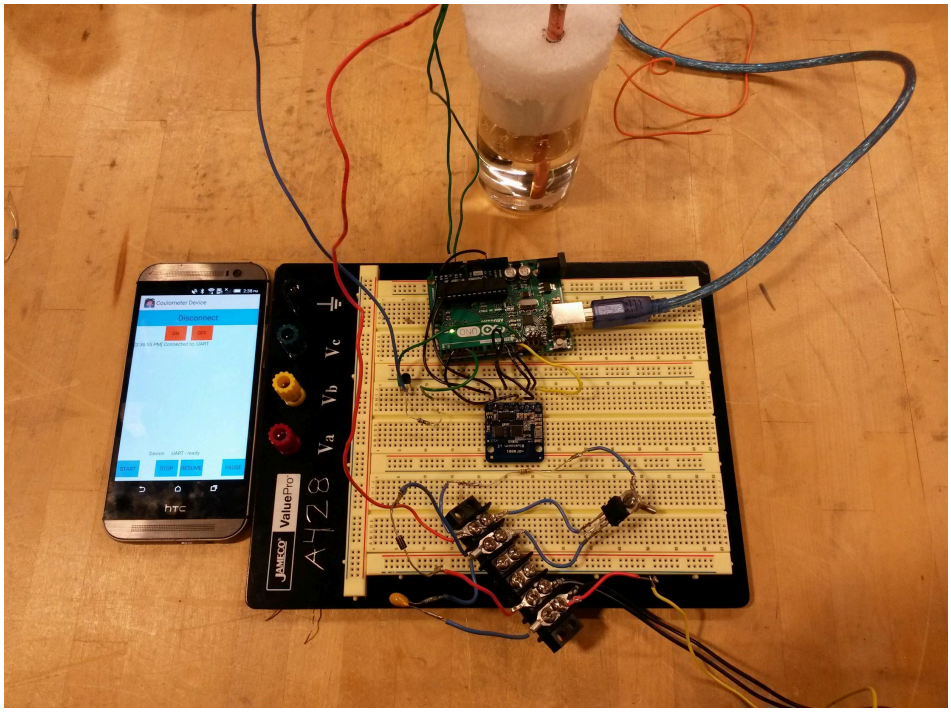


Figure 9: Experiment setup with smartphone on the left, solution on top of the image, the current source at the bottom and the BLE module is in the middle of the BreadBoard

3.2 Bluetooth module communication

The BLE module communicates with the smartphone through the mobile application that we developed. The different functions are explained in part 2.4 and the current display can be seen on figure 7.

3.3 Further developments

We thought about and aimed at different stretch goals that we could not reach mainly due to a lack of time. We think that the project could be improved and taken to a prototype by implementing a few features.

To improve the user experience and the collected results' exploitation, a function to send the data via email to the students could be useful. Another solution could be to save the data on text files that could be read on the smartphone or transferred to a computer.

A 3d printed case would be ideal to make the system compact and easily transportable. The case modelisation and realisation could be the main task to take the project to a prototype state. Ideally this case would allow to power the system on both a socket or a 9V battery.

We think that the system is currently power efficient and would not need major modifications in terms of components as the arduino and the bluetooth module does not require a lot of power, and the current source would still need to be powered.

4. Safety & Ethics

Safety has to be ensured while handling the device with liquids, especially acids and concentrated solutions as this may lead to the damage of the Arduino board and BLE. The wires connected must always remain intact in order to avoid disconnection of communication between the various parts used. If wires are out of place, please ensure that they are plugged back into their respective ports. Failure to do this could lead to malfunction and permanent damage to the device. Our app doesn't infringe on any fundamental human rights. Nor does it pose a threat to the health and safety of users, or to the environment. Just like usage of any electronic device, caution and safety must be ensured while using the coulometer device as well. The product developed has been intended for usage globally with fundamental resources such as electricity and a smartphone, apart from the experimental setup required to collect readings.

5. Budget

Budget Allocated:

- $750 for computational and I/O upgrade
- $250 for 3D printing and fabrication services

Parts Received: Arduino Uno, Heat Sink, Terminal Strip, Diode, Switch, AC Adapter, Resistor, Capacitor.

ITEM	VENDOR PART #	QUANTITY	COST PER ITEM	TOTAL COST W/O TAX
Hardware				
BlueFruit LE (BLE)	**nRF8001**	**1**	**19.29**	**19.29**
Arduino Uno-R3	**DEV 11021**	**1**	**19.99**	**19.99**
Breadboard	PRT 12002	1	0	0
Heat Sink		1	0	0
Terminal Strip		1	0	0
Diode		1	0	0
Switch		1	0	0
9V AC Adapter		1	0	0
Resistor		1	0	0
Capacitor		1	0	0
TOTAL				**$39.28**
Software				
Android Studio		1	0	0

Figure 10: Budget Table

6. Conclusion

Ultimately we were able to successfully accomplish all specifications set out by the Statement of Work (SOW), at the end of the 10 week duration given. The improved system design satisfied the needs laid out in the background. We were able to replace the analog switch, used to turn on the system, with a digital switch, remove the multimeter and use an android application as a means to relay current, voltage and charge data to the user, and also wrote the android application to record measurements at 1 second time intervals, capped off at 300 seconds. A number of smaller goals were met while working on the project, which entailed learning how to program an Arduino to enable our device to communicate with software (mobile application) and also learned the android development process, in which none of us had any experience with prior to the project. The most important thing that was achieved was the capability to work together as a group and be productive and professional during our meetings.

While the goal of the project was for the most part a straightforward task, there was a lot of time and research put into ensuring that we could communicate with the physical device and what parts were required for us to be able to do that. As mentioned a number times, we hope that the improved system design will enable the user to be able to collect data much quicker and simpler that had they been working with the original test setup. We also see it as being a benefit in eliminating the need for equipment such as the multimeter and analog switch now that everything will be operated through a mobile phone application.

If this project were to be continued by another team there are some improvements that can still be made. The mobile application currently displays voltage, current, and charge data that are collected by the device. An additional save feature with timestamp can be made to enable the user to view previously collected data. Furthermore, if this data can be shared via e-mail that could also be of some use to the user. Finally, the system is still in it’s breadboard stage and can be made such that it is mounted onto a board/pcb to make the system more robust.

Appendix

Appendix A: Sample Arduino Code:

```
#include <SPI.h>
#include "Adafruit_BLE_UART.h"
#include "Timer.h"

// Connect CLK/MISO/MOSI to hardware SPI
// e.g. On UNO & compatible: CLK = 13, MISO = 12, MOSI = 11
#define ADAFRUITBLE_REQ 10
#define ADAFRUITBLE_RDY 2     // This should be an interrupt pin, on Uno thats #2 or #3
#define ADAFRUITBLE_RST 9

Adafruit_BLE_UART BTLEserial = Adafruit_BLE_UART(ADAFRUITBLE_REQ, ADAFRUITBLE_RDY,
ADAFRUITBLE_RST);
/**************************************************************************/
/*!
    Configure the Arduino and start advertising with the radio
*/
/**************************************************************************/
bool serialAvailable = false;
long loopNum = 0;
long loopSpeed = 1000;
long commandReadSpeed = 197;
long ADSpeed = 100;
long commandID = 0;
float commandData = 0.00;
float resistor = 65;
bool sendSum = false;
Timer loopTimer;
int loopTimerID = 0;
Timer ADTimer;
int ADTimerID = 0;
Timer commandReadTimer;
int commandReadTimerID = 0;
aci_evt_opcode_t laststatus = ACI_EVT_DISCONNECTED;
int sensorValue = analogRead(A1);
int sensorValue2 = analogRead(A2);
int sensorValueMinus = sensorValue - sensorValue2;
float voltage = sensorValueMinus * (5.00/1023.00);
float current = 1000*voltage/resistor;
float charge = 0;
void setup(void)
{
  Serial.begin(9600);
  while(!Serial); // Leonardo/Micro should wait for serial init
  Serial.println(F("Adafruit Bluefruit Low Energy nRF8001 Print echo demo"));

  // BTLEserial.setDeviceName("NEWNAME"); /* 7 characters max! */
  BTLEserial.begin();
  loopTimerID = loopTimer.every(loopSpeed,BLESending);
  ADTimerID = ADTimer.every(ADSpeed,takeReading);
```

```
  commandReadTimerID = commandReadTimer.every(commandReadSpeed, commandReading);
}
void commandReading() {
     String commandString ;
     String dataString;
     while (BTLEserial.available()) {
      char c = BTLEserial.read();
      Serial.print(c);
        if ( c =='X') {
        char cc;
        while (BTLEserial.available()) {
        cc = BTLEserial.read();

        Serial.print(cc);
        commandString += cc;
        }
      }
      if ( c == 'Z') {
        char cc;
        while (BTLEserial.available()) {
        cc = BTLEserial.read();
        dataString += cc;
        }
      }
    }
    commandData = dataString.toFloat();
    if (commandString == "start")
    commandID = 1;
    if (commandString == "pause")
    commandID = 2;
    if (commandString == "stop")
    commandID = 3;
    if (commandString == "resume")
    commandID = 4;
    if (commandString == "on")
    commandID = 5;
    if (commandString == "off")
    commandID = 6;
    if (commandString == "speedUp")
    commandID = 7;
    if (commandString == "speedDown")
    commandID = 8;
    if (commandString == "set9")
    commandID = 9;
}
void takeReading() {
      // AD
      sensorValue = analogRead(A1);
      sensorValue2 = analogRead(A2);
      sensorValueMinus = sensorValue - sensorValue2;
      voltage = sensorValueMinus * ( 5.00 / 1023.00);
      current = 1000*voltage / resistor;
       charge += current * ADSpeed * 0.001;
```

```
}
void BLESending(){
     BLESending1();
     BLESending2();
     BLESending3();
}
void BLESending1() {
     if (serialAvailable && loopNum > 0 ) {

      // We need to convert the line to bytes, no more than 20 at this time
      uint8_t sendbuffer[20];

      //  s.getBytes(sendbuffer, 20);

      String BLEString = "V"+String(voltage);
      //uint8_t volts[20];
      BLEString.getBytes(sendbuffer, 20);
      char sendbuffersize = min(20, BLEString.length());

      BTLEserial.write(sendbuffer, sendbuffersize);
      loopNum --;
    }
}
void BLESending2() {
     if (serialAvailable && loopNum > 0 ) {

      // We need to convert the line to bytes, no more than 20 at this time
      uint8_t sendbuffer[20];

      //  s.getBytes(sendbuffer, 20);

      String BLEString = "I"+String(current);
      //uint8_t volts[20];
      BLEString.getBytes(sendbuffer, 20);
      char sendbuffersize = min(20, BLEString.length());

      BTLEserial.write(sendbuffer, sendbuffersize);
      loopNum --;
    }
}
void BLESending3() {
     if (serialAvailable && loopNum > 0 ) {

      // We need to convert the line to bytes, no more than 20 at this time
      uint8_t sendbuffer[20];

      //  s.getBytes(sendbuffer, 20);

      String BLEString =  "C"+ String(charge);
      //uint8_t volts[20];
      BLEString.getBytes(sendbuffer, 20);
```

```
      char sendbuffersize = min(20, BLEString.length());

      BTLEserial.write(sendbuffer, sendbuffersize);
      loopNum --;
    }
}


void loop()
{

  // Tell the nRF8001 to do whatever it should be working on.

  BTLEserial.pollACI();

  // Ask what is our current status
  aci_evt_opcode_t status = BTLEserial.getState();
  // If the status changed....
  if (status != laststatus) {
    // print it out!
    if (status == ACI_EVT_DEVICE_STARTED) {
        Serial.println(F("* Advertising started"));
    }
    if (status == ACI_EVT_CONNECTED) {
        Serial.println(F("* Connected!"));
    }
    if (status == ACI_EVT_DISCONNECTED) {
        Serial.println(F("* Disconnected or advertising timed out"));
    }
    // OK set the last status change to this one
    laststatus = status;
  }

  if (status == ACI_EVT_CONNECTED) {
    // Lets see if there's any data for us!
    if (BTLEserial.available()) {
      Serial.print("* "); Serial.print(BTLEserial.available()); Serial.println(F(" bytes available
from BTLE"));
    }
    // OK while we still have something to read, get a character and print it out

   // int commandNum = readCommand();
    if (commandID == 1){   //start
      serialAvailable = true;
      loopNum = 300;
      charge = 0;
      commandID = 0;
      digitalWrite(4,HIGH);
    }
    if (commandID == 2){   //pause
      serialAvailable = false;
      commandID = 0;
      digitalWrite(4,LOW);
```

```
    }
    if (commandID == 3){   //stop
      serialAvailable = false;
      loopNum = 0;
      commandID = 0;
      charge = 0;
      digitalWrite(4,LOW);
    }
    if (commandID == 4){   //resume
      serialAvailable = true;
      digitalWrite(4,HIGH);
      commandID = 0;
    }
    if (commandID == 5){   //on
      digitalWrite(4,HIGH);
      commandID = 0;
    }
    if (commandID == 6){    //off
      digitalWrite(4,LOW);
      commandID = 0;
    }
    if (commandID == 7){
      loopTimer.stop(loopTimerID);
      loopSpeed -= 50;
      loopTimerID = loopTimer.every(loopSpeed,BLESending);
      commandID = 0;
    }
    if (commandID == 8){
      loopTimer.stop(loopTimerID);
      loopSpeed += 50;
      loopTimerID = loopTimer.every(loopSpeed,BLESending);
      commandID = 0;
    }
    if (commandID == 9){
      loopTimer.stop(loopTimerID);
      loopSpeed = round(commandData);
      loopTimerID = loopTimer.every(loopSpeed,BLESending);
      commandID = 0;
    }
    ADTimer.update();
    loopTimer.update();
    commandReadTimer.update();

  }
}
```

Appendix B: Sample Android Code:

```
 // Handle Disconnect & Connect button
       btnConnectDisconnect.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View v) {
               if (!mBtAdapter.isEnabled()) {
                   Log.i(TAG, "onClick - BT not enabled yet");
                   Intent enableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                   startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
               }
               else {
                  if (btnConnectDisconnect.getText().equals("Connect")){

                    //Connect button pressed, open DeviceListActivity class, with popup
windows that scan for devices

                     Intent newIntent = new Intent(MainActivity.this,
DeviceListActivity.class);
                      startActivityForResult(newIntent, REQUEST_SELECT_DEVICE);
                } else {
                   //Disconnect button pressed
                   if (mDevice!=null)
                   {
                      mService.disconnect();

                   }
                }
               }
           }
       });

                     //Collect Readings//
               if (action.equals(UartService.ACTION_DATA_AVAILABLE)) {

                   final byte[] txValue = intent.getByteArrayExtra(UartService.EXTRA_DATA);

                   runOnUiThread(new Runnable() {
                       public void run() {
                           try {
                               String text = new String(txValue, "UTF-8");

                               float voltage = Float.parseFloat(text);
                               float current = (voltage / 60);
                               float charge = current * 1;

                               String currentDateTimeString =
DateFormat.getTimeInstance().format(new Date());

                               listAdapter.add("
" + currentDateTimeString);
                               listAdapter.add("Voltage: " + text + "V");
                               listAdapter.add("Current: " + current + "A");
                               listAdapter.add("Charge : " + charge + "C");


messageListView.smoothScrollToPosition(listAdapter.getCount() - 1);

                           } catch (Exception e) {
```

```
                            Log.e(TAG, e.toString());
                        }
                    }
                });
            }
            //*********************//
            if (action.equals(UartService.DEVICE_DOES_NOT_SUPPORT_UART)) {
                showMessage("Device doesn't support UART. Disconnecting");
                mService.disconnect();
            }


        }
    };
}
```

Appendix C: Parts List

Item	Description	Vendor
9-V AC adapter	**Power Source**	**Dr. Skip Pomeroy**
Heat Sink	**Keeps electrical components from overheating**	**Dr. Skip Pomeroy**
LM317T Regulator	**Op Amp**	**Dr. Skip Pomeroy**
Diode	**low voltage filter**	**Dr. Skip Pomeroy**
4.7µ capacitor	**differentiator**	**Dr. Skip Pomeroy**
75 ohm Resistor	**Resistor**	**Dr. Skip Pomeroy**
2N3904 NPN BJT	**Switch**	**ECE Undergraduate Lab**
Arduino UNO	**Microprocessor**	**Jing Liu**
Bluetooth Low Energy Module	**Bluetooth Wireless Communication**	**Adafruit**

Appendix D: References

Adafruit tutorial for BLE module:
https://learn.adafruit.com/getting-started-with-the-nrf8001-bluefruit-le-breakout/introduction

Nordic Semiconductor Android Code link (given by Adafruit):
http://www.nordicsemi.com/eng/Products/nRFready-Demo-Apps/nRF-UART-App

Dr.Pomeroy's documents:
-Constant-Current Coulometric Titration of Hydrochloric Acid
-Protocol for the titration experiment