Ag Sensors for Measurement of Soil Water Contents

DESIGN DOCUMENT

Team Number: 38
Client: Iowa State University - Research Department
Advisers: Liang Dong
Team Members/Roles:
Colin Cox – Software group
Jarrod Droll – Sensor group
Rachel Hoke – Sensor group
Wage Miller – Control Box group
Scott Rowekamp – Software group
Tyler Thumma – Control Box group
Team Email: sdmay18-38@iastate.edu
Team Website: http://sdmay18-38.sd.ece.iastate.edu
Revised: 04/23/18/V3

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1 PROBLEM AND PROJECT STATEMENT

The research department is working towards improving the crop yields within the agriculture field. In order to increase yields, it is important to research and understand the differences between various crops and the nutrients contained in the soil.

This can be done with the development of electrochemical sensors. Our goals is to improve these sensors, it will allow for enhanced data collection and interpretation. The research team can then find out what would be the best type of crops to plant in certain areas or how much fertilizer will be needed throughout the season. Ultimately this research can be used to help farmers grow better crops increasing their yields allowing for more food and less waste.

1.2 OPERATIONAL ENVIROMENT

Our soil sensors will have an operating environment within a corn or soybean field or a greenhouse. With that in mind, they need to be dust/dirt proof to be able to keep the electronics clean from dirt debris. Additionally, the sensors will be out in the field for anywhere from a few hours to a few days, weeks or months, so sustainability is a key component that we intend to address with a rechargeable battery.

1.3 INTENDED USERS AND USES

The intended users for our sensors in the near future are those researching crop yields and corn/soybean farmers. This will begin with researchers in the College of Agriculture at Iowa State University and then be outsourced to other agricultural companies.

The plan is to then have our product used by farmers in order to give them the technology necessary to conduct soil sample testing on their own, rather than having to ship soil out for testing to a third-party company.

We need to make sure that our sensors meet the standards of both intended users, we plan on achieving this by upkeeping and improving the accuracy of the sensors and data collection while improving the durability and longevity of the control boxes. Also, we are allowing for easy instillation into the fields.

1.4 ASSUMPTIONS AND LIMITATIONS

Assumptions:

1. Each control box will have one sensor.
2. Soil water sensors can be modified to measure nitrogen, phosphorous, and potassium.

Limitations:

1. The system must operate under moderate temperature changes (-20 Fahrenheit to 120 Fahrenheit)
2. Project budget cannot exceed $2,000.00

1.5 EXPECTED END PRODUCT AND DELIVERABLES

Soil Sensor PCB

A new sensor PCB design/product that improves the fabrication process and accuracy of the sensor.

Software

A simple to use mobile app that will collect the data, display it, and upload it to a remote server for storage. Server sided software to accommodate uploading and viewing of sensor data.

Control Box

A new PCB design that is smaller in size than the current one as well as a physical box that is easy to manufacture.

2 Specifications and Analysis

Functional requirements:

- Can withstand temperatures ranging from –20 Fahrenheit to 120 Fahrenheit
- Portable
- Takes accurate measurements of nitrogen concentrations
- Low maintenance

Nonfunctional requirements:

- Easy to navigate software interface
- Efficient networking
- Cheap for consumers

2.1 DESIGN

Sensors

Our original proposed design in shown below in Figure 1. This design eliminates corrosion on the sides of the copper by adding a shadow mask around the contact pads. Errors can occur on when silver is deposited on our PCB due to the corrosion of the copper. After discussing with our faculty advisor and graduate TA's we decided on the design shown below in Figure 2.

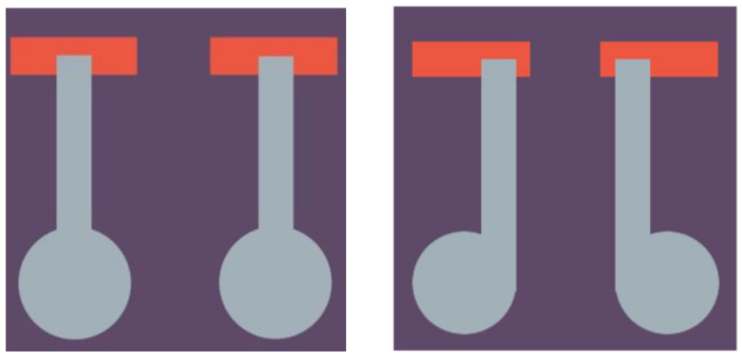


Figure 1: Original Proposed Design.

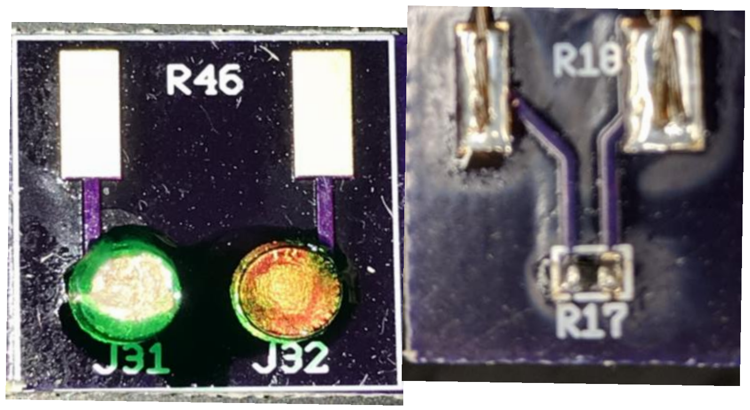


Figure 2: Current Sensor Design

The current design incorporated our idea for a circular contact pad for the working and reference electrode. The current design was also sent out for fabrication and had a solder mask added, this eliminated the corrosion on the sides of the copper contact pads. We also added a thermistor on the back of the PCB in order to measure the temperature of the soil during testing.

Control Box

For our circuit design we are proposing making the PCB voltage differential circuit two-sided by moving half of the op-amps, capacitors, and resistors to the other side. In addition, we want to add the voltage charger, which is currently a separate PCB, to the voltage differential circuit in order to only have one PCB total in our control box.

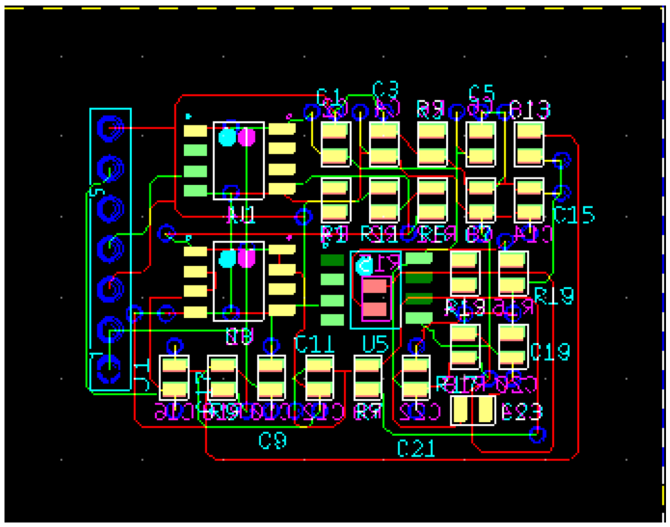


Figure 3: Proposed circuit PCB design

Server

The server was designed using NodeJS for the code, and MongoDB as the backend database. The server has two parts to it. The first part is the API endpoints. When the server starts, it creates an http server there is a different endpoint for each function of the API like adding a new user account for example. The server checks the request URL and either servers a static file or it will process the input if it is one of the endpoints. The second part of the server is the static content. This is a collection of HTML and JavaScript documents that a client's web browser may load. The following figure shows what the login page looks like:

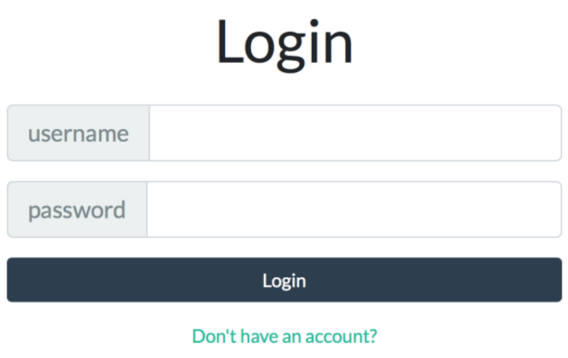


Figure 4: Login Screen

Microcontroller

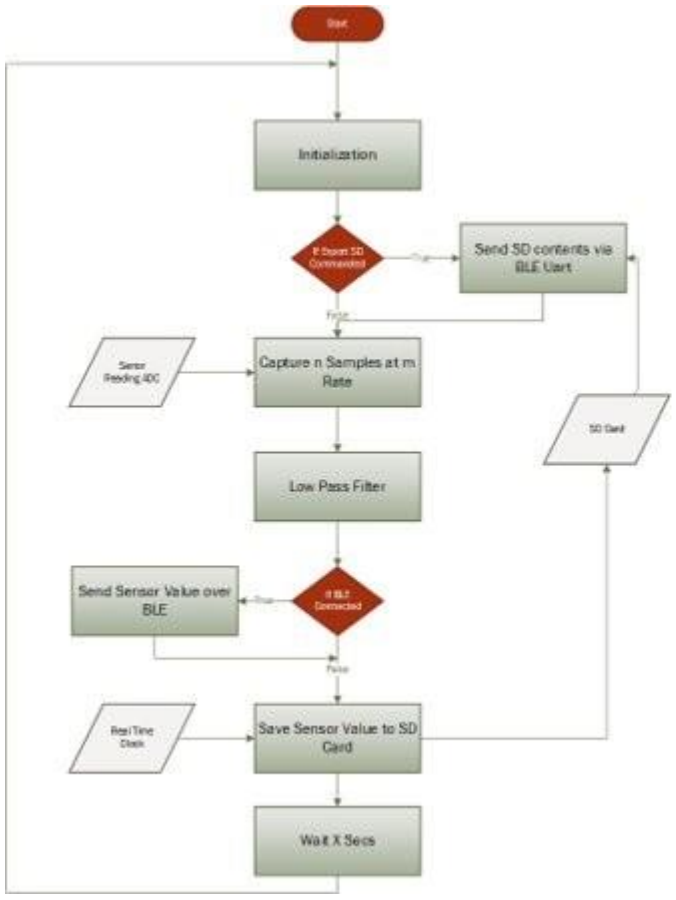


Figure 5: Microcontroller Operating Sequence

The microcontroller software was written in C++. The microcontroller has two modes the first mode being single measurement mode. In single measurement mode the microcontroller reads the voltage potential of the sensor with an external ADC and pass through a low pass filter before forwarding the information on to the Android app via Bluetooth LE. The second mode of the app is a continuous data logger which logs values to an internally SD card at a rate defined in the

MCU software. This process can take place even without an Android phone present. The end user can collect the data from the SD card with the app as well using Bluetooth Low Energy. The microcontroller software is written as a cyclic process meaning there are no delays commands anywhere. The software can easily be adjusted to collect samples at different rates and with different filter constants. We defined a simple communication protocol between the two apps consisting of 'op-codes' the Android app can send the MCU as seen below:

OP Codes:

O – Get a Single Measurement
C – Start Data Logging
S – Stop Data Logging
D – Clear Internal SD card
X – Disconnect

3 Testing and Implementation

Each of the main areas of improvement underwent vigorous testing and multiple iterations. A summary of the testing process for each area is as follows:

Sensors

In order to test our sensors, we first had to acquire a nitrogen solution with a high part per million (PPM). We then diluted the nitrogen solution until we had concentration from 1 PPM to 100 PPM. We then inserted the working and reference electrode into the nitrogen solution for 2 minutes and recorded the voltage using a multimeter. Next, we cleaned our sensors in diluted water between tests. All of the tests were performed at room temperature, there is a temperature sensor on the sensor PCB, however due to time constraints we were unable to perform tests at multiple temperatures. Our results from the sensor testing is shown below in Figure 4.

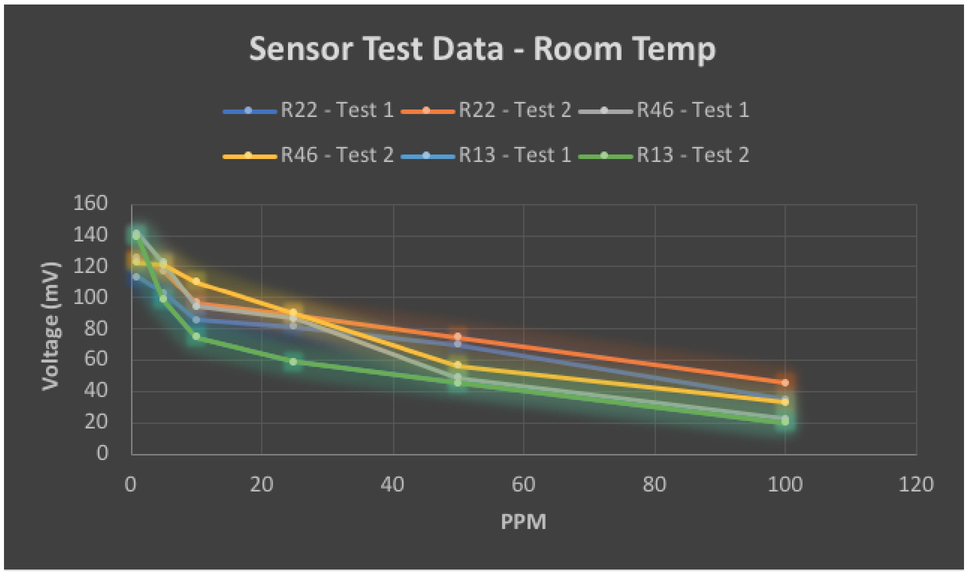


Figure 6: Sensor Test Results

Control Box

The testing process for the control box involved two main areas of focus, those being the printable circuit board and the physical box itself.

Starting with the printable circuit board, new designs needed to be created with the goal of reducing the size of the circuit board that Dr. Liang Dong's team was already using. These designs were approved by graduate assistants. They then ordered parts the parts from DigiKey, and boards were sent off to get manufactured. Once the components and PCBs came in, each board was soldered by our team and then tested by measuring voltages across various components of the circuit, most notably being the output. The final design that we came up with which reduced overall size was combining the voltage amplification circuit with the power charger, as well as utilizing both sides of the PCBs created using Multisim and Ultiboard software.

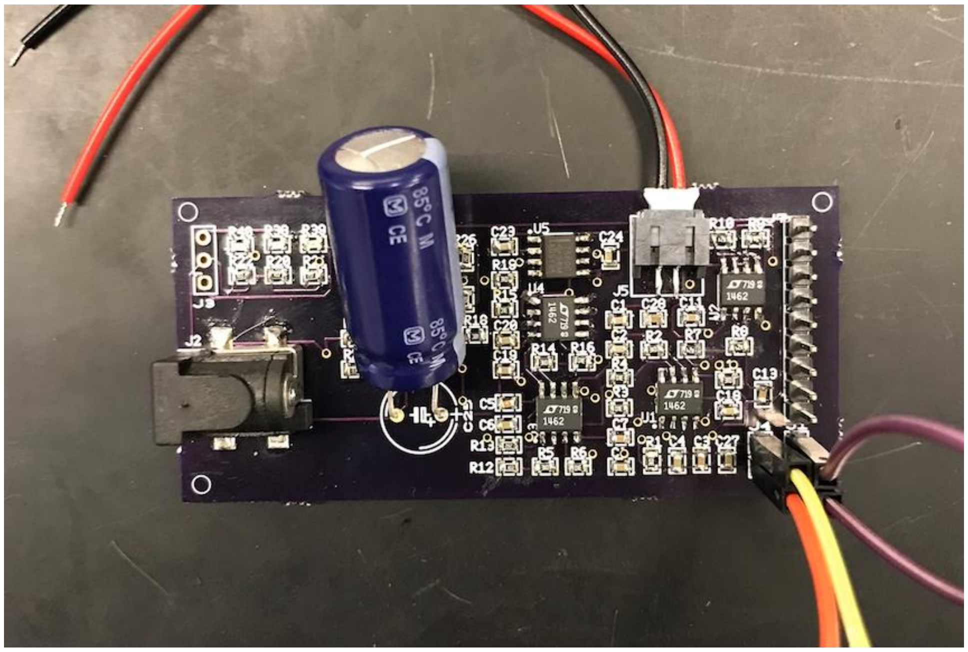


Figure 7: PCB with Amplifying Circuit and Power Booster

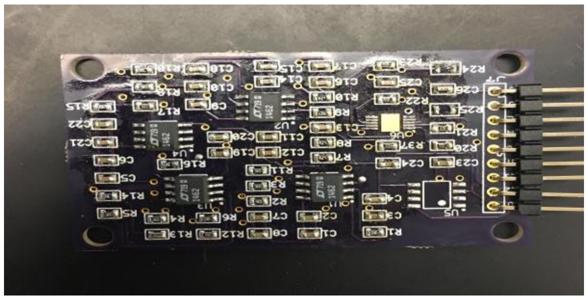


Figure 8: First iteration of PCB

As we design and tested our PCB we found that with our version it was reading in the values from the sensor slowly. This occurred with multiple boards we soldered together. We then went to the original design of implementing all three PCB's separately.

For the physical box, the main improvement that our client wanted was to make the manufacturing process easier, as several hundred of these boxes were going to have to be made in the future. In order to do this, we decided to use circular cutouts for the switch. Circular holes are much easier to do by machine than square cutouts, thus reducing manufacturing time required. The testing process for the physical box involved creating several designs through AutoCAD and Autodesk Inventor, and then sending that design to either the ETG or Boyd Lab. Once the box was finished, we tested sizing of the cutouts as well as how they influenced performance of our components.

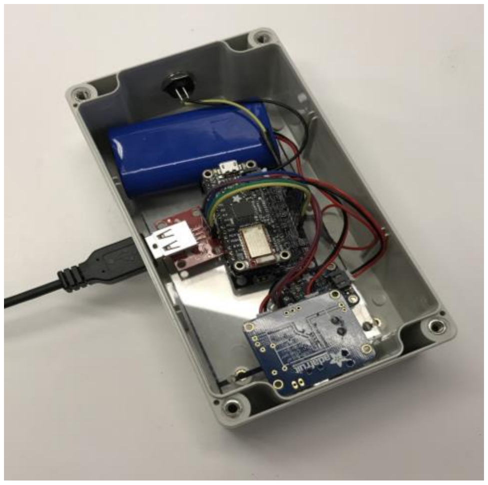
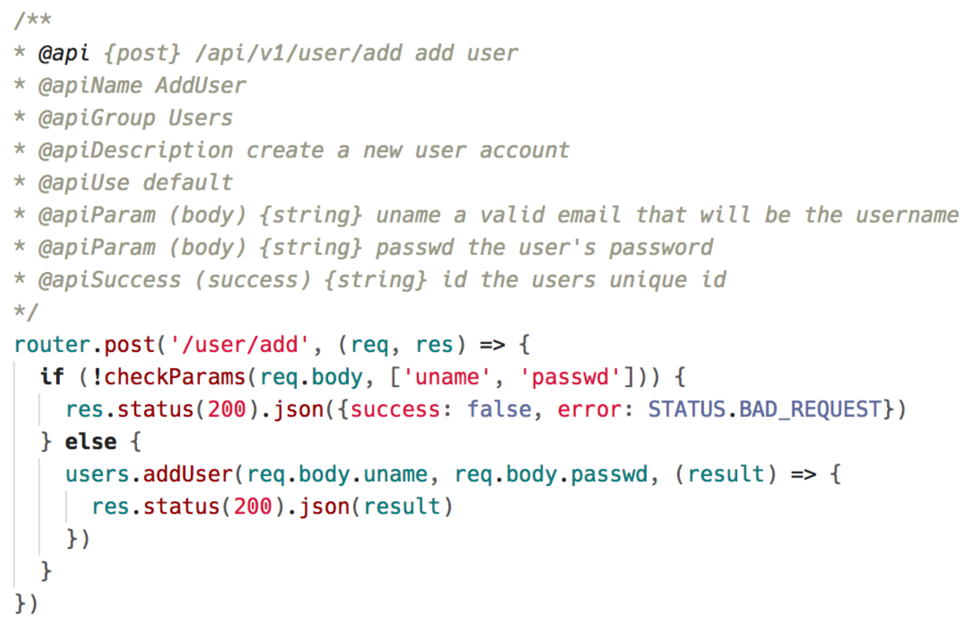


Figure 9: Final Control Box

Software

Server

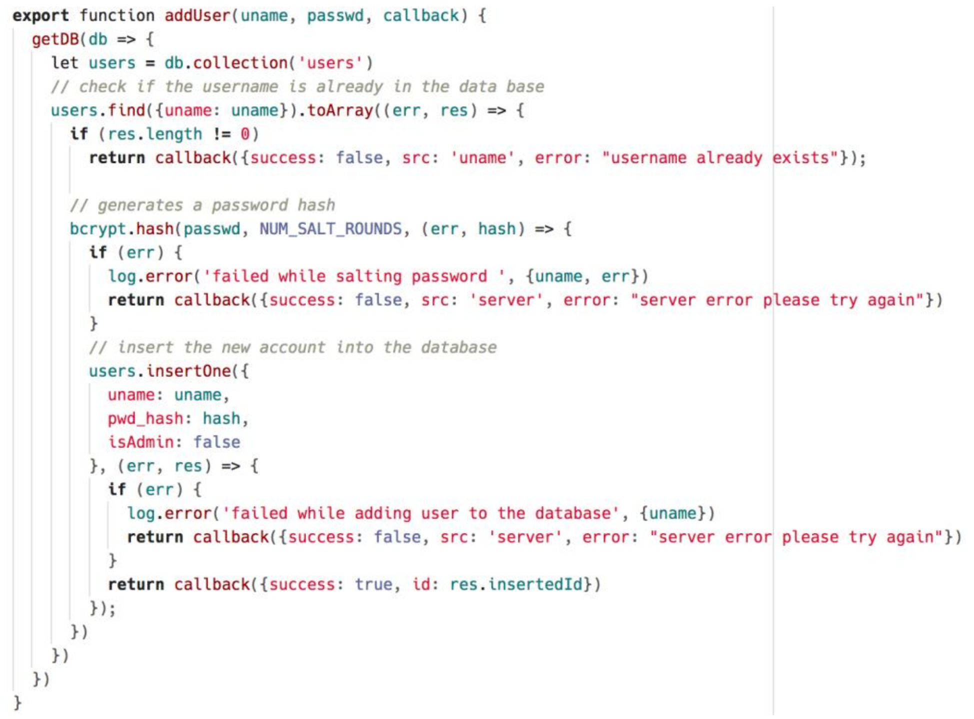
Testing for the server was done by checking and working with the following information below.



```
/**
* @api {post} /api/v1/user/add add user
* @apiName AddUser
* @apiGroup Users
* @apiDescription create a new user account
* @apiUse default
* @apiParam (body) {string} uname a valid email that will be the username
* @apiParam (body) {string} passwd the user's password
* @apiSuccess (success) {string} id the users unique id
*/
router.post('/user/add', (req, res) => {
  if (!checkParams(req.body, ['uname', 'passwd'])) {
    res.status(200).json({success: false, error: STATUS.BAD_REQUEST})
  } else {
    users.addUser(req.body.uname, req.body.passwd, (result) => {
      res.status(200).json(result)
    })
  }
})
```

Figure 10: Add User Endpoint

This is the code for the endpoint that can be called to add a user. It works by first checking the correct parameters were specified, in this case a username and password. The other endpoint functions are set up in a similar way and all of them are documented.

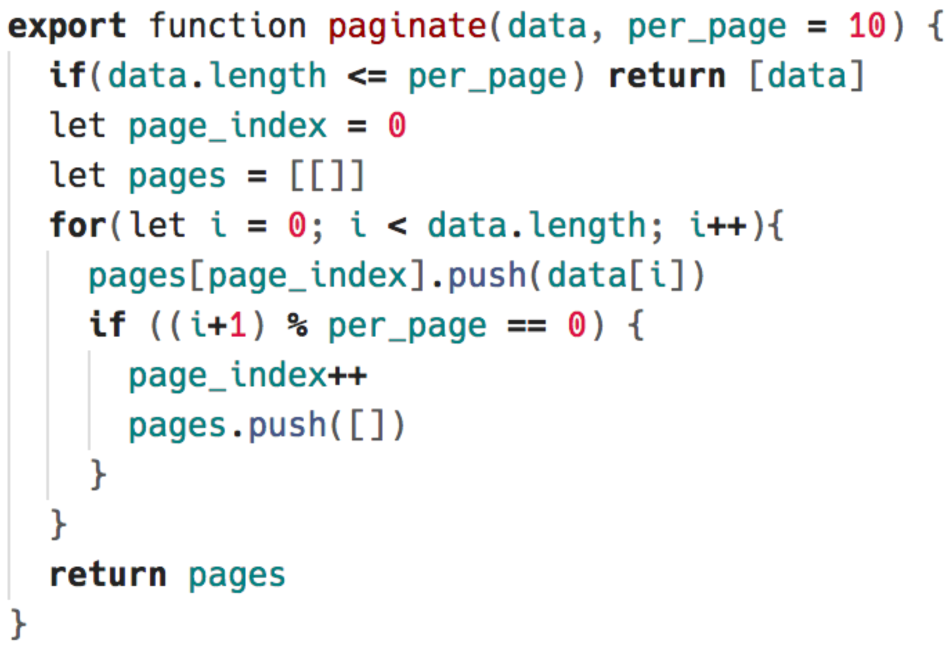


```
export function addUser(uname, passwd, callback) {
  getDB(db => {
    let users = db.collection('users')
    // check if the username is already in the data base
    users.find({uname: uname}).toArray((err, res) => {
      if (res.length != 0)
        return callback({success: false, src: 'uname', error: "username already exists"});

      // generates a password hash
      bcrypt.hash(passwd, NUM_SALT_ROUNDS, (err, hash) => {
        if (err) {
          log.error('failed while salting password ', {uname, err})
          return callback({success: false, src: 'server', error: "server error please try again"})
        }
        // insert the new account into the database
        users.insertOne({
          uname: uname,
          pwd_hash: hash,
          isAdmin: false
        }, (err, res) => {
          if (err) {
            log.error('failed while adding user to the database', {uname})
            return callback({success: false, src: 'server', error: "server error please try again"})
          }
          return callback({success: true, id: res.insertedId})
        });
      })
    })
  })
}
```

Figure 11: Add User To Database Function

Above shows the code for adding a user to the database. This code is called by the add user endpoint. Designing functions for interfacing with the database what a programing challenge. Since JavaScript is asynchronous, the database functions also needed to be asynchronous. This is the reason why there are many nested functions. The nested functions are commonly referred to as callback and they are called at the end of an asynchronous function.



```
export function paginate(data, per_page = 10) {
  if(data.length <= per_page) return [data]
  let page_index = 0
  let pages = [[]]
  for(let i = 0; i < data.length; i++){
    pages[page_index].push(data[i])
    if ((i+1) % per_page == 0) {
      page_index++
      pages.push([])
    }
  }
  return pages
}
```

Figure 12: Paginate Function

Above is a snippet of code that paginates a table of data. Measuring the soil contents will generate a large amount of data, we needed a way to efficiently report it back to the user. The paginate function works by taking a large array of data and turning it into an array of smaller arrays. The user can the request a particular page to be returned. This makes it easier for a client application like the static HTML pages or the Android app to get data from the server and display it the user.

In order to ensure that all API endpoints worked correctly, the application Postman was used to test the same request after any additions were made to the server code. We then looked at the response and the data on the database to ensure everything worked correctly.

Android App

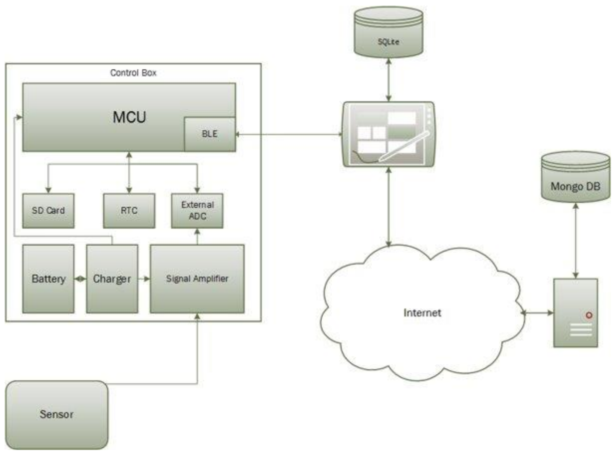


Figure 13: Top Level System Software Overview

The Android app is written entirely in Java. The main task of the Android app is to poll the control box for the sensor data, apply the correct calibration curve and then present the information to the user as well as pushing the data to a database. The Android app tries to account for almost everything the user and the operating environment can throw at it including but not limited to: no network connectivity, spotty network connectivity, Bluetooth disabled, user forgetting to save data, device rotation, gps disabled. The Android app makes use of fragments to help simplify the user flow. The Android app interfaces with the control box using Bluetooth Low Energy (BLE). The app is setup such that it will only connect to control boxes not any BLE device. After the app has successfully acquired a reading it will try to upload the result to the webserver if it can't it will cache the result in a local sqlite database until it can be uploaded. The app also lets you view past readings from both the webserver the those cached to the device itself that have yet to upload.

4 Closing Material

4.1 CONCLUSION

In conclusion, our team has accomplished quite a lot over the course of senior design. We were able to develop and test new sensors, create a new circuit and improve the control box, and develop a software application that can readily interface with the control box in order to see the concentration measurements in real time.

5 Appendices

5.1 APPENDIX I – OPERATION MANUAL

Hardware

First the sensor must be placed into the soil. Once the sensor is in position, the control box simply needs to be turned on. This initiates voltage measurement and amplification from the sensors and voltage amplification circuit. From here, the app simply needs to be opened and connected to the specific control box that is being used. A nitrate reading will then be displayed on your electronic device.

App

The Android application walks you through the entire process after installing the app from the .apk simply open the app and follow the easy to use prompts the app even takes care of making sure you have Bluetooth enabled.

Server

A regular user will not need to do anything with the server. To start the server, first ssh into the server. Then run "cd soil_sensors" then run "npm run start" to start the server.

5.2 APPENDIX II – ALTERNATIVE DESIGNS

Sensors

This proposed design for the in-soil sensors changes the copper contact pads into a more cur design, the proposed solution is displayed in Figure 14.

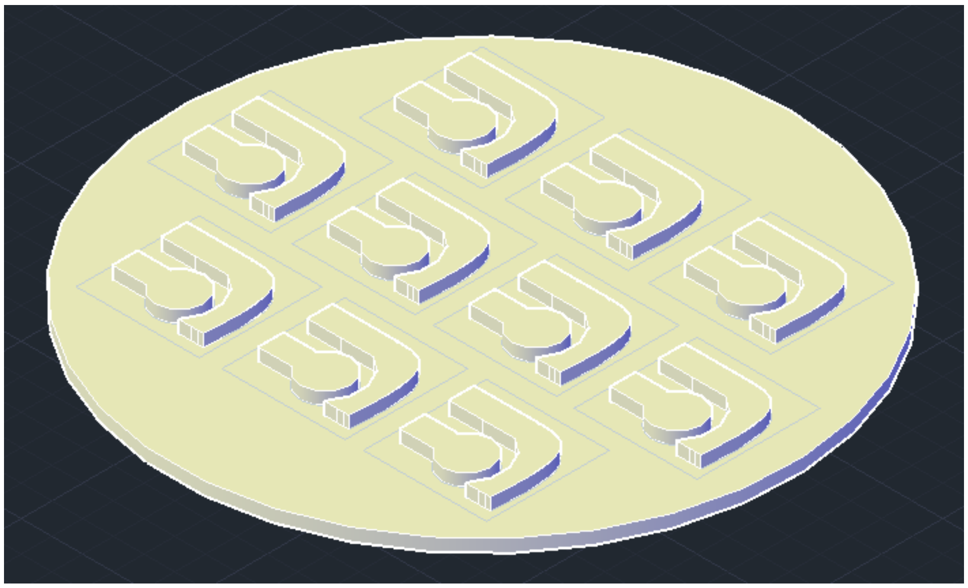


Figure 14: Proposed design for in-soil sensor.

From Figure 14, you can see that our proposed design in also wafer shaped rather than typical rectangular shape. The circular shape allows the PCB to fit better in our silver deposition machine. We plan on either applying a glue or shadow mask between the working and reference electrode to eliminate the corrosion problem.

Control Box

The first design that we tried was designing the amplifier circuit, power charger, and power booster on one board. After meticulous Ultiboard work, we were able to successfully simulate a working circuit. After receiving the physical components and boards for this design however, we continuously burnt out the power booster chip (TPS6109) and could not figure out why. Even when having our graduate assistant assisting us we burnt out a total of eight chips on three boards. This led us to try something less complex (the amplifying circuit and battery charger together without incorporating voltage booster) which worked successfully in testing and ended up being our final design. We decided finally to purchase the voltage booster from Adafruit (PowerBoost 1000 Charger).

The second PCB circuit had two main alternative designs that were tried and tested before we arrived at our final decision of adding the power charger to the voltage amplifier circuit. The first design was just the voltage amplification circuit itself but utilizing both sides of the PCB instead of one.

For the physical box, we generally used the same design throughout the project. The main challenge of this was getting the correct measurements for each component, as well as finding a machine that had the ability to create smooth, clean cuts. As a result, we had a few schematics that were off by a few millimeters, which resulted in us going through a few boxes before arriving at our final product.

Software

The initial plan for the software architecture consisted of a mesh network where all the devices would read from sensors continuously we decided to go away from this plan when our client requested the control box be optimized for one time reads of multiple samples in different locations.

We were also going to use the internal ADC of the microcontroller however we found it's 10-bit resolution was only accurate within +/-10% with the final design we used a 16-bit ADC that we could theoretically be accurate to within +/-0.1%

5.3 APPENDIX IV – CODE

https://git.ece.iastate.edu/sd/sdmay18-38