EE/CprE/SE 491 WEEKLY REPORT 5

10/11/2024 – 10/17/2024

Group number: sdmay25-04

Project title: Wireless Mesh Network for Pesticide Spray Monitoring and Mapping

Client: Claussen Lab- Iowa State University

Advisor: Nathan Niehart

Team Members/Role:
Software Side -
Ashley Falcon: IDEs and Microcontrollers, Group Communicator
Drew Scheidler: Mesh Networking; Note Taker
Henry Hingst: Mesh Networking; Group Leader
Hardware Side -
Hector Perez Prieto: Microcontroller; Circuit Design and Testing
Yok Quan Ong: Circuit Design and testing; Microcontroller
Wesley Smith: Circuit Design/Simulation; Microcontrollers; Note Taker

---

- **Weekly Summary**
  - This week, we split into two groups to get sections of our project started
    - Hardware Team
      - Designed simulations and built a circuit that will output set voltages to the microcontroller based on the resistances provided by the Claussen Lab sensor
      - Decided what components could be used based on the requirements of the project
    - Software Team
      - Installed and configured the necessary software to compile code and program it onto the ESP32 board
      - Created a circuit and software to program the ESP32 to blink an LED on and off
  - We had an advisor meeting as well, where we determined the teams' next steps
    - Recounted software and hardware teams' progress
    - Received advice on current progress

- ***o*** **<u>Past week accomplishments</u>**
    - ***Ashley Falcon***:
        - Collaborated with Software team to make an LED blink
        - Initialized Linux terminal
            - Determined which platform to use - Ubuntu
            - Set up freeRTOS
                - Link to Espressif's git repository where tool paths are stored
            - Used terminal to set folder location for EE491 code
        - Connected terminal to microcontroller
            - Once freeRTOS was loaded in the file, I had to run commands to install and export
            - Allowed me to start a new file
                - Blink LED project with main and build folder
            - Used PowerShell to share and attach USB port
        - Worked on blink LED code
            - Used basic tasks provided in freeRTOS and while loop
            - Flashed code to microcontroller
            - Success!
    - ***Drew Scheidler***:
        - Collaborated with the software team
        - Environment setup
            - Initialized the Linux terminal for development
            - Selected Ubuntu as the platform for development
        - FreeRTOS setup
            - Installed and configured FreeRTOS
            - Accessed Espressif's Git repository to configure tool paths
        - Microcontroller interaction
            - Connected the terminal to the microcontroller
            - Used PowerShell to share and attach the USB port
            - Flashed the LED blink code to the microcontroller
        - LED blink project
            - Developed the blink code using basic FreeRTOS tasks and a while loop
            - Successfully verified the functionality of the blink project
    - ***Hector Perez Prieto***:
        - Designed and tested a voltage divider that we built, this circuit will help with finding the resistance that is being detected from the pesticide that

lands on our IDEs
        - These tests were done with voltage sources and multimeters from the labs
        - The Hardware team was able to accurately measure voltages that would be used to help find the data that would be shared with the master node
        - Calculated theoretical values and then tested to verify my calculations
        - Collaborated with the Hardware team which includes Wesley Smith and Yok Quan Ong, to test and brainstorm solutions to small issues we were running into
- ***Henry Hingst:***
    - Collaborated with Drew and Ashley to install FreeRTOS (the software provided by the makers of the ESP32 used to program the board)
        - Helped Drew and Ashley get started with the installation
        - Diagnosed and fixed issues with accessing the USB port used for flashing
    - Created the software and circuit used to program an ESP32 to turn an LED on and off repeatedly
    - Created a GitHub repository to be used to store the code written by our team
- ***Yok Quan Ong:***
    - Testing the voltage divider circuit that will be implemented in the design
        - Measured the fixed value resistance so that the input voltage is within the range that ESP32 can handle
        - Simulated on LTspice and tested on a breadboard
        - Measured the output voltage and the IDE (resistance voltage)
            - Voltage ranges between 0.1-1.1V
        - Choose the Op Amp (LMC660CM for the testing stage), which might change to (TLV9104 on the actual design)
- ***Wesley Smith:***
    - Design and test of a voltage divider circuit that will be implemented in our initial design
        - Created and tested LTspice simulations before the hardware group met in the lab
        - Created an initial circuit to test on a breadboard
            - The goal of this circuit is to send voltages ranging from 0.1V-1.1V to the ADC based on the changing

resistance ranges of Claussen Labs' provided sensor
- Recorded data along the way
- Revised and complicated circuit to meet the needs of the project
- Presented findings to Advisor, gained ideas for next week

- ***o*** **Individual contributions**

NAME	Individual Contributions	Hours this week	HOURS cumulative
Ashley Falcon	Linux setup, LED blink test, FreeRTOS	**7**	**25**
Drew Scheidler	Set up Linux environment, completed LED blink test, integrated FreeRTOS into project	**8**	**28**
Hector Perez Prieto	Designed and tested a voltage divider circuit	**8**	**26**
Henry Hingst	FreeRTOS setup, LED Blinking Project, and GitHub Repository Creation	**8**	**28**
Yok Quan Ong	Simulated and tested voltage divider	**8**	**26**
Wesley Smith	Simulated and tested a functional voltage divider circuit to provide resistances to the ADC	**8**	**29**

- ***o*** **Plans for the upcoming week**
  - Hardware
    - Design wheatstone bridge circuit addition to current circuit that adds precision to our current circuit. Goal of 0.1-1.1V output
    - Create a back to back diode circuit to clamp the output voltage to not damage the ESP if we were to unplug the Claussen Lab sensor
    - Meet on Sunday to test the wheatstone bridge circuit that each member simulated and test on breadboard.
    - Fit bit solution for the range that we have tested from last week
    - Find the input offset of the op amp
    - Looked into the power supply, voltage regulator, and on off switch additions to current circuit
  - Software
    - Follow tutorials to make several small projects to learn the basic code functionality of the ESP32
    - Begin designing the code to read the voltage of the circuit the hardware team is designing and turn it into the sensor reading
    - Begin designing the code to transmit data from one ESP32 to another using 802.11 LR
    - Create bash and .bat files to streamline the coding, compiling, and flashing process for the ESP32

- *o* **Summary of weekly advisor meeting**
    - Professor Neihart provided feedback on both the hardware and software teams.
    - Hardware:
        - Reviewed the circuit and the data we designed and measured
        - Provided feedback: wheatstone bridge for a stable voltage output
        - Look for details for the input offset of op-amp and how that would affect the output voltage
        - Look up some future-use components: voltage regulator, rechargeable battery, and on-off switch
    - Software:
        - Discussed success with Blink Test
        - Will reach out to researchers to see how to flash microcontroller
            - Our process requires repetitive steps to get it to run, such as attaching the ports
        - Assigned next steps
            - Will work on communicating between microcontrollers
            - Will determine how to read input voltage data from ADC and convert it to digital data
            - Will type out the steps currently being used to set up and flash
                - Steps are slightly tedious and good documentation will be essential to staying on top of the project