**ECEN5823 Final Project Team Proposal**

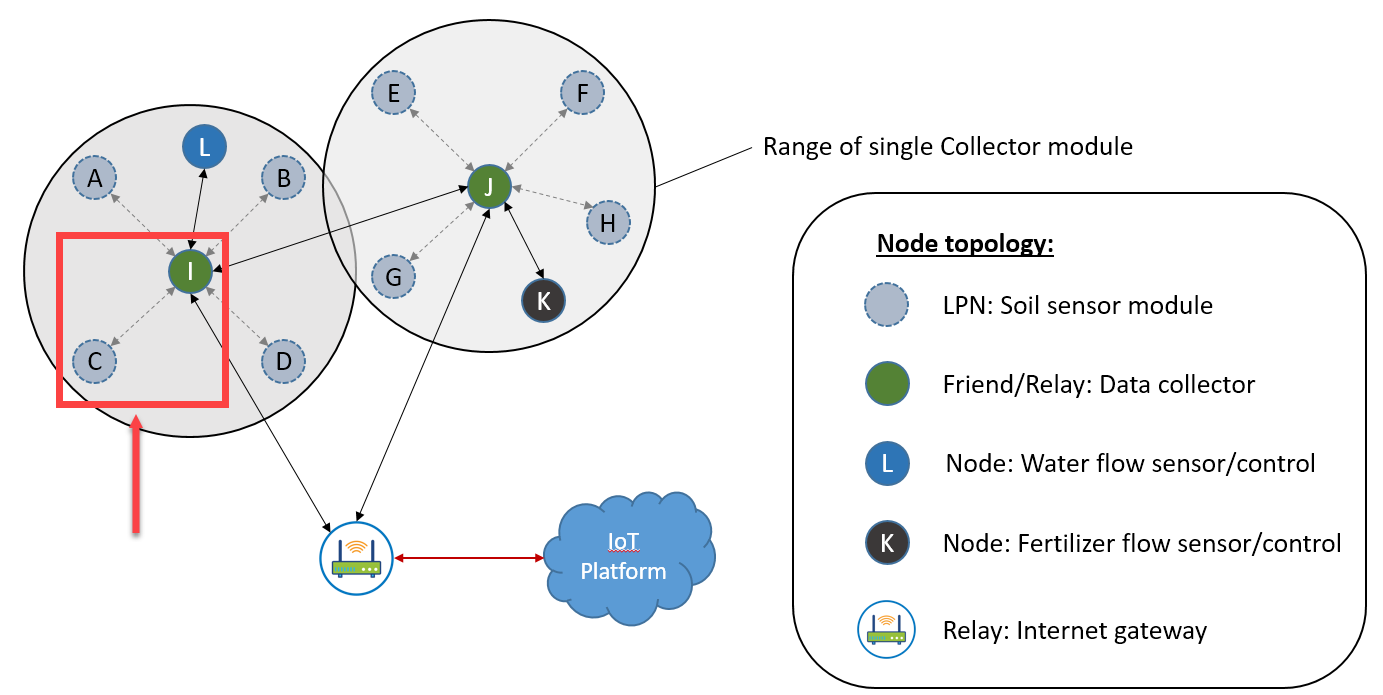
**Project Name:** Intelligent Agriculture Control System

**Members:** Victor Kronberg

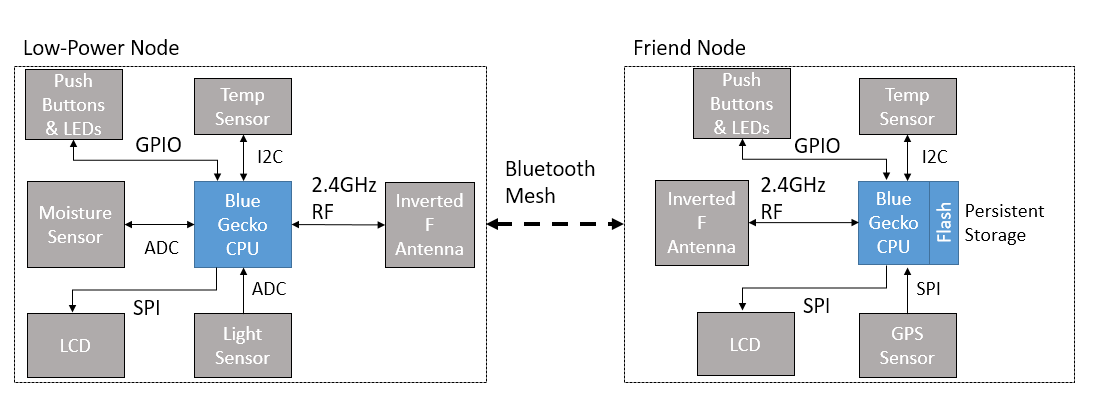
1. **Project Status Updates 1 & 2**
   1. 11/21 – Both sensors have been integrated into the implementation. They can both be read from successfully and accurately. A scheduler has been set up to poll the two sensors at a predefined interval (this would be a sleep interval) for the LPN. Framework for the LPN and Friend node pairing has been implemented, but not tested. Reset functionality has been implemented and tested. Next up is to complete the LPN/Friend pairing and define structure for transmission of data over BT Mesh.
   2. 11/12 - Thus far, the sensor breakout boards have been acquired from SparkFun, header pins have been soldered on and they are connected to a breadboard for testing and firmware development. The initial skeleton codebase has been checked out (btmesh) and functionality such as timer, gpio, and display have been fully ported over from other projects. Initial ADC initialization code has been implemented, but not fully tested. Pins have been selected for ADC input.
   3. Challenges so far have been finding time. I have not hit the meat of the technical portion, so no challenges there so far.
2. **Subsystem Overview**
   1. With a rapidly increasing population and limited fresh water resources for agriculture, utilizing our resources as efficiently as possible become ever more critical in the coming years. On top of that, high water prices and global agricultural competition mean that U.S. farmers need every advantage that they can get in order to stay financially competitive. Without detailed understanding of how the environment is impacting the soil and how the soil responds to treatments (water/chemical), it becomes ever more challenging to stay competitive.
3. **Subsystem High Level Design**
   1. The idea behind this project is to provide a modular, scalable wireless solution for monitoring and controlling soil conditions in an agricultural setting in order to optimize watering and fertilization to give the farmer the greatest yield per dollar spent on resources. This includes soil moisture level, soil temperature, sun exposure, and mineral content. There would also be sensors for monitoring water flow for irrigation and mineral content in fertilizer applications. This information would be fed into an intelligent platform that utilizes AI to help a farmer to optimize their water usage and soil treatments based on the soils’ response to watering and fertilizer applications.

Subsystem proof-of-concept will implement a low power node and friend node. This subsystem is responsible for sensing the properties of the soil and transmitting that information back to a “collector”. The “collector” stores the sensor data in persistent memory in case of network outage or system/power failure. In a full deployment, the “collector” would also transmit and relay sensor data towards the Internet gateway.

* 1. Subsystem Block Diagram:



**Figure 1: Subsystem of network to implement**



**Figure 2: Block diagram of subsystem and sensors to implement**

* 1. Sensors for Low Power Node:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sensor** | **Data type** | **Property Measured** | **Sparkfun P/N** |
| Moisture sensor | Analog (ADC) | Soil moisture content | <https://www.sparkfun.com/products/13637> |
| Ambient light sensor | Analog (ADC) | Ambient light | <https://www.sparkfun.com/products/8688> |
| Humidity & Temperature | I2C | Soil/surface temperature | Si7021 – Included on Blue Gecko |

Sensors for Friend/Relay Node:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sensor** | **Data type** | **Property Measured** | **Sparkfun P/N** |
| GPS | SPI | GPS Location | <https://www.sparkfun.com/products/15193> |
| Humidity & Temperature | I2C | Soil/surface temperature | Si7021 – Included on Blue Gecko |

* 1. *Sparkfun part numbers above*
  2. Profiles and exposed Services to implement:
     1. **Sensor Profile**
        1. Current Ambient Illuminance
        2. Current Temperature Service
        3. Current Voltage Input (Convert to Soil moisture content)
     2. (optional) **Location and Navigation** **Profile**
        1. Location Service
  3. **Implementation Plan:**
     1. Moisture sensor
        1. ADC Input: Expansion pin 14
           1. Port A, Pin 1
        2. Power (3.3V): Expansion pin 20
        3. Ground (GND): Expansion pin 19
     2. Ambient Light sensor
        1. ADC Input: Expansion pin 12
           1. Port A, Pin 0
        2. Power (3.3V): Expansion pin 20
        3. Ground (GND): Expansion pin 19
     3. Temp/Humidity sensor
        1. Utilizing Si7021 code from previous assignments for temperature
        2. ~~Add code for also reading off humidity measurements~~
     4. ~~GPS~~
        1. ~~SPI~~
           1. ~~SPI\_CS: Expansion Pin 10~~

~~Port C, Pin 9~~

* + - * 1. ~~SPI\_SCK: Expansion Pin 8~~

~~Port C, Pin 8~~

* + - * 1. ~~SPI\_MISO: Expansion Pin 6~~

~~Port C, Pin 7~~

* + - * 1. ~~SPI\_MOSI: Expansion Pin 4~~

~~Port C, Pin 6~~

* + - 1. Power (5V): Expansion pin 18
      2. Ground (GND): Expansion pin 1
  1. *Command table in “****documents****” folder*
  2. **Persistent data**
     1. Persistent data will be stored in a struct in the server. Struct will contain client state (temp, soil moisture level, and ambient light level) and a timestamp. Each piece of information with be type uint32, or 4 bytes. The 4 pieces of information will then be 16 bytes each and will all be stored in a single struct. GPS is assumed to be static, so there should not be a need to store that as well – if there is space, then it could be. Each struct will be 16 bytes + any persistent data overhead.
     2. Using the 128 unique keys that BGAPI allows for persistent store, we can create a FIFO that stores up to 64 elements at a time and performs a basic level of write leveling. This will consume 1024 of the 2048 bytes available in flash at most and ensure that there is enough persistent memory available for the mesh API (key storage, etc.). We can rotate through all of the keys to ensure that we minimize the number of write/erase cycles to the flash memory. For the demo, I am only using 16 keys to show how overflows are handled.
     3. In production, we would store the latest entry to persistent memory every half hour, or 2 per hour. This will ensure that we can log up to 1.33 days of coarse data in the case that the friend node is unable to transmit data at more regular intervals. That should give the end customer enough time to physically reach the unit and pull the data down. This is well within the 10 year retention time of the EFR32BG13 flash memory
     4. Each block (unique unit) of persistent memory is erased every 64 hours in this configuration. With 10,000 erase cycle before failure, the flash memory on the EFR32BG13 should last 73 years before failure.
  3. **User Interface**
     1. Client/Low-Power Node
        1. Display device name: “Low-power node”
        2. Display device BT address
        3. Display connection status: “Unprovisioned”, “Provisioned”
        4. Friend status: “No LPN”, “Friend Lost”, “LPN Established”
     2. Server/Friend & Relay
        1. Display device name: “Friend node”
        2. Display device BT address
        3. Display connection status: “Unprovisioned”, “Provisioned”,
        4. Display friendship status: “No LPN”, “Friend”
        5. Display LPN sensor state:
           1. Temp (degrees C)
           2. Soil moisture level (%)
           3. Ambient light (%)
        6. ~~Display own state:~~
           1. ~~Temp/humidity~~
           2. ~~GPS location~~

1. **Proposed Development Schedule (task – target implementation date)**
   1. Interface software to moisture and light sensor (ADC) – 11/14
      1. Develop ADC initialization routine –*complete*
      2. Develop IRQ-based ADC sensing routine – *completed 11/17*
      3. Develop ADC routine to switch channels – *completed 11/20*
   2. Interface software to temp/~~humidity sensor~~ (I2C) – 11/16 – *completed 12/2*
   3. Integrating sensors to application code – 11/19 - *complete*
   4. Integrating LCD to application code – 11/20 - *complete*
   5. ~~Load Power Management of sensors – 11/22~~ - *- not feasible with BRD4104A*
   6. Developing BLE LPN / Friend code – 11/26 - *complete*
   7. Develop persistent memory routine – 11/27 - *complete*
   8. ~~Firmware Update (DFU) – 11/30~~  *- not needed for Mesh project*
   9. Validation of the project – 12/1-12/10 - *complete*
2. **Low Power Design Verification**
   1. In order to test the low power requirements of the LPN, I will provision the devices and ensure the LPN/Friend friendship is established. The Friend node power consumption is not of concern as it will be powered to enable it to function as a relay. The low-power node will be testing utilizing the energy profiler. I will have the LPN programmed to publish at prescribed intervals as well as read sensor data/update LCD at intervals as well. I expect to see the power consumption between transmissions/sensor readings/display updates to be significantly less than other times. This will be confirmed with energy profiler where I will be able to measure out the pre-defined intervals and see where the transmissions occur.
3. **Lessons Learned**
   1. One lesson I learned was that in order for the EFR32BG13 to go into deep sleep while the ADC is active, the ADC must be configured to be in ASYNC mode, the ASYNC clock must be enabled, and this clock source must be AUXHFCRO.
   2. In order for the LPN to go into deep sleep, the mesh API’s gecko\_configuration\_t must have the sleep flags set to enable deep sleep (this is not set by default on some examples). Flags are set as “.sleep.flags = SLEEP\_FLAGS\_DEEP\_SLEEP\_ENABLE”. To make sure the LPN sleeps as deep as possible, the Proxy node feature should be disabled once the node is provisioned.
   3. The persistent storage routines for the Blue Gecko automatically perform write levelling. It shares the same memory space as the rest of the items stored in Flash (network keys, pairing info, etc.), and these are stored using the same routine. The different objects just use different keys to access objects through the persistent storage routine – 0x4000-0x407F are reserved for the user. When storing in flash using the persistent memory functions, there is some overhead added to each object stored – estimated to be 4-8 bytes, but that is unclear. There is also much less memory available for storage than originally anticipated – in my program, I ran into a limit storing 31 elements at 16-bytes each plus a single 2-byte element (plus overhead)
   4. Opinion: Bluetooth Mesh might be a good network in an environment where managed co-existence is required. Even though managed flooding with Bluetooth Mesh is very inefficient in networks with many nodes, the fact that Bluetooth Mesh only uses small frequency bands at 2402MHz, 2426MHz, and 2480MHz, means that a significant portion of the 2.4GHz band remains undisturbed by Bluetooth Mesh networking. This would free up quite a bit of the 2.4GHz band for protocols such as Thread or WiFi That being said, in a busy Bluetooth Mesh network, there might be challenges with coexistence of BLE devices as they would need to utilize the Bluetooth Mesh/Advertising channels for bonding.
4. **Final Status**
   1. The successfully implemented components were the LPN sensors (temperature, ambient light, and soil moisture). The LPN/Friend was successfully implemented and the LPN is able to operate in low power mode. On the Friend node, the GPS sensor implementation was not completed. Had this been completed successfully, I would have added an accurate timestamp from the GPS data to the persistent data, as well as a location tag.