Garden Automated Rain/Daylight Executed by Near-Infrared Sensing

Nicholas Chitty, Brendan College, Scott Pierce, Justin Pham-Trinh

University of Central Florida, Dept. of Electrical and Computer Engineering, Orlando, Florida, 32816-2450

***Abstract*—This paper will show the application of near infrared spectroscopy and how it can measure electromagnetic waves from the emission of soil. Near infrared spectroscopy is an absorption spectroscopy method that can help determine the chemical composition of a substance through the radiation the substance gives off. Soil itself is a mixture of organic and inorganic substances that all together directly contribute to a garden's environment. We are starting with soil with unknown qualities, so comparisons will be made between our soil and soil of known qualities to match and ensure that our plant is in a healthy and suitable environment.**

I. Introduction

In the past year, we have seen a great increase in remote sensing, wireless communication, API integration, and so much more. All of which have been made more available and economical. The internet has also seen its share of "DIY" projects and its continuing growth.

In the agriculture industry, there have been new advancements in technology with high performance water distribution, network communication, and remote sensing. This research is intended to advance the field by producing a system that can maintain a suitable environment for a plant to grow. In the environment, there will be sensors that will help modify the conditions within the environment. In addition to this system, it will feature a web interface for notifications and the ability for the user to set settings.

Monitoring soil isn't always the most fun or the easiest task, because things could get dirty or we might forget about our plant. This project will feature an on-the-rise technology in the form of a spectrophotometer. Smart systems nowadays are big learning machines that are constantly aware of its surroundings. For a smart agricultural system, it would need to determine variables such as moisture levels, nutrient content, pH levels, and so much more. This paper will introduce near infrared spectroscopy as another method to soil sensing that may prove to be more beneficial over traditional products or techniques.

This project will also present other fields such as system controls, power, and web, all to provide a "set it and forget it" home gardening experience. A home gardening experience where the garden bed can communicate with the user and the user can provide instruction of what to do, but also at the core, this is a microcontroller project that anybody can do with just slight knowledge and understanding of digital communications. Like any smart device too, the internet plays an important role in making this as hands-off as it can be. There are many different communication protocols such as Bluetooth, Zigbee, Thread, and even short-range/long-range protocol. For this project we decided to go with WiFi because of its decreased bandwidth and that we aren't expecting to produce or receive large amounts of data.

An important aspect to a lot of devices as well is data storage and web usage. This is important so that we can look back on previous information and make observations and conclusions. With our microcontroller, the web system is going to communicate with it through transmission control protocol. Transmission control protocol is a standard on how to establish and maintain a network connection to exchange data. The web system will also have 16gb database to store data and have the ability to support multiple garden beds in a scaled solution. As mentioned before, the web system will have a feature to allow the user to set settings but also read the data that is stored. Lastly, an important factor to any garden is the weather and knowing when it might be cold, hot, or even rain outside. As a part of our web system, it will communicate using HTTP requests with a weather service to receive updates on the weather.

Solar power has been a growing source of energy for the past years and still continues to be with new developments and breakthroughs with solar technology. A lot of systems nowadays are solar powered but these products are not constantly in use, they have to turn of eventually. When the product is off the solar panel can still collect energy which is stored in a battery for conservation. In this paper, we will demonstrate our system running independently, battery powered but charged through solar panels.

In all, this paper will present how near infrared spectroscopy can be used to monitor a plant's environment and notify users with information of its health. Generating and guiding electromagnetic waves into a housing, scanning the substance, and then converting this optical power into a voltage that can be read and analyzed. It is then the microcontroller will communicate with the web system, storing the information and deciding on if anything needs to be done and notifying the user.

II. Controller Subsystem

In order for our system to be as self and power efficient as possible from an end-user perspective, our team decided to use a low-power, Internet-of-Things (IoT)-focused wireless microcontroller. To make the process of operating our product as hands-off as possible to end-users, the microcontroller will operate in Access Point (AP) mode to serve information to the user's mobile device. Our product will not produce or receive large amounts of data, or need complicated control schema for our various subsystems—and with power budget being a major concern of ours, our team needed to choose a product that would fit the bill.

A. CC3220 Overview

The Texas Instruments CC3220-series (hence referred to as the "CC3220", the "MCU", or the "microcontroller") of microcontrollers are WiFi-enabled chips with an ARM Cortex-M4 central processor and a WiFi network processor, along

with many useful peripherals and power management modules. This series of processors is delivered alongside a software development kit (SDK) provided by Texas Instruments to ease the development of IoT applications. The CC3220 is capable of running on bare metal, or with a Real-time Operating System (RTOS), allowing us to organize and schedule our various tasks.

The CC3220's WiFi network processor (NWP) supports 802.11b/g/n, SmartConfig provisioning, IPv4 and IPv6. The NWP also as the ability to host an internal HTTP/HTTPS server, and contains its own filesystem.

B. Connection

Our product shall be able to broadcast a WLAN in AP mode. This will allow the user to connect to the product's SSID. The user would then be directed to a web portal hosted by the MCU's internal HTTP server, where they will be able to view information and telemetry related to the product. Hosting a web interface would allow unanimous adaptation of our product for home users. The system shall be able to broadcast a WLAN in AP mode with a nominal signal strength of 6 dBm measured at the antenna. The network shall adhere to the standards of 802.11b, g, or n.

C. Weather API Implementation

As a stretch goal, the user will be able to configure the MCU in Station mode to allow it to connect to the user's home WLAN's SSID as a client. The user would still be able to access the MCU's web portal and see the same information as they had before. In addition, the MCU would be able to access an API to receive weather information and inform the user of recommended actions pertaining to their garden bed (e.g. if the system determines there will be freezing temperatures overnight, it may suggest to the user to cover the garden bed with a sheet or towel).

D. Over-the-Air Updates

At this time, our team does not intend to provide a method for over-the-air updates (OTA), however, this is a stretch goal that may be completed in the future.

E. Startup and Shutdown

Startup and shutdown will occur in a timely manner (within seconds). Abrupt shutdown shall not damage any components of the system, and the system will be able to restart to its previous state without any input from the user. If the system freezes or crashes, its internal watchdog timer will automatically reboot the device.

F. Data

As well as being able to see and configure the network settings of the product, the user will be able to see data relating to the OH-levels and common plant nutrient levels as determined by the spectral analysis of the product. For one measurement, our team will need to store 16 bits of data per position from 130 unique positions, per sensor.

$$\frac{16\,\mathrm{b}}{8\,\mathrm{b/B}} \times 130\,\mathrm{positions} \times 2\,\mathrm{sensors} = 520\,\mathrm{B} \tag{1}$$

With 8 Mb of the 32 Mb (1 MiB of the 4 MiB) shared serial flash dedicated to storing results, this allows us to store 2016 measurements.

$$\frac{1\,\mathrm{MiB}}{520\,\mathrm{B/measurement}} = 2016\,\mathrm{measurements} \tag{2}$$

Stretching out measurements to be recorded every 15 minutes, this will allow a user to see individual measurements stretching back exactly three weeks.

$$2016\,\mathrm{measurements} \times \frac{1\,\mathrm{measurement}}{15\,\mathrm{min}} = 30240\,\mathrm{min} \tag{3}$$

As a stretch goal these measurements can be aggregated and analyzed, and allow us to serve trends and predictions to the user.

G. Low Power Modes

The system shall run in active mode between sunrise and sunset, while the battery has 40-100% calculated charge remaining. The system shall run in low power mode between sunrise and sunset while the battery has 10-40% calculated charge remaining, and between sunset and sunrise while the battery has 10-100% calculated charge remaining. The system shall run in critical power mode while the battery has 5-10% calculated charge remaining. The system shall be in shutdown mode while the battery has 0-5% calculated charge remaining. The system shall be able to switch power modes within 5 seconds of the interrupt being triggered.

H. Libraries

It should be expected that the C++ standard library (as defined in C++14) will be used, as well as the POSIX library, along with the Texas Instruments SimpleLink CC32xx SDK.

I. Sensing

The system shall monitor and perform analog-to-digital conversion of the spectral values sensor values at least every 15 minutes while in active mode. The system will use I2C to talk to the analog-to-digital converter (ADC) located in the sensing subsystem.

J. Motor Control

Four GPIO lines will be used to control the stepper motor holding the sensing PCB (which includes the photodiodes, op-amp circuitry, ADC, motor controller, and connector). Depending on the digital values passed to the motor controller, the motor controller is able to control the movement direction and speed. Our team found that using the SimpleLink SDK's high-level hardware abstraction layer (HAL) provides too long of a delay when changing values in the four GPIO lines. Nanosecond-levels of delay were needed, rather than the microsecond levels of delay our team was seeing. Instead

Fig. 1. Toggling GPIO using driverlib results in a very short (350 ns) delay.

of using highly-abstracted APIs for control of the motor, our team got as close to hardware as is possible with C/C++ and used TI's driverlib driver library specifically for the CC3220. Instead of a simple GPIO number being passed to the function, our team had to determine the GPIO port and port-specific pin of each input of our motor controller. The driverlib function checks the port for correctness, and then performs a system call to write a value directly to memory (in our case, the address of the GPIO port). Using driverlib also allows us to bit stuff our GPIO pins if they are located on the same port—in our second iteration of the board, the motor controller GPIOs were relocated to the same GPIO port and bitmasks used so we can change all four GPIO pins with one driverlib write function. This allows us to toggle a single GPIO pin within a period of 350 ns (seen in II-J). With a speed of 80 MHz in active mode, this would mean we're able to write to GPIO in only 28 processor cycles.

$$350\,\text{ns} \times (80 \times 10^6 \text{cycles/s}) = 28\,\text{cycles} \tag{4}$$

This is a huge accomplishment for our team, especially with a program written in C++ and utilizing HAL.

K. Analog-to-digital Conversion

The sensing subsystem contains a Texas Instruments ADS 7142 ADC. The ADS 7142 is an IoT-focused nanowatt ADC with 2 external channels, an I2C interface, a sampling frequency of 140ksps, and an effective resolution of 16 bits in High-precision Mode. This ADC will be able to measure between 0 and 3.3 V from the output of the sensing subsystem's photodiode op-amp circuit.

It is expected that each photodiode's circuit will provide a voltage of between 0 and 3.3 V. The 16-bit ADC provides for an input of the same range. Therefore, the resolution of the ADC is calculated below:

$$\frac{(3.3-0)\,\text{V}}{2^{16}\,\text{steps}} = 50.35\,\mu\text{V/step} \tag{5}$$

These steps will be used to measure OH composition and nutrients in the soil. The MCU will directly control GPIO and bit-bang values to the stepper motor controlling the position of the photodiodes allowing them to measure the full range of spectral values.

```
1  class Water {
2      public:
3          static Water& instance();
4
5          // Disallow copying
6          Water& operator = (const Water&) =
   delete;
7          Water(const Water&) = delete;
8
9          // Disallow moving
10         Water& operator = (Water&&) = delete;
11         Water(Water&&) = delete;
12
13         // Class functions
14
15     private:
16         Water();
17         ~Water();
18
19         // Class variables
20 };
```

Fig. 2. An example of the Meyers' Singleton pattern implementation within the project.

L. Development Model

An Agile development model was used to program the control subsystem and its various modules. Code reviews were performed on an as-needed basis by a convening of members of the MCU subsystem and the web subsystem teams. Texas Instruments Code Composer Studio v12 was used to program, compile (via TI ARM compiler v20), and debug the C++-based project. GitHub was used as a repository for the project, using GNU Git for version control.

The Meyers' Singleton design pattern was used for most of the classes found in our project repository (e.g. for our water solenoid class as seen in II-L). Because our team is using a RTOS with a scheduler and threads, we are using that to our advantage to divide and schedule tasks for the different submodules of our project. However, one problem our team took into consideration is that there is was only one physical instance of each submodule. The normal Singleton pattern traditionally has been used when one wants only one instance of a class initialized at any one point, but they are not thread-safe. One variation of this pattern is the Meyers' Singleton, which guarantees that only one instance of a type is available at any time. Initialization of this pattern is thread-safe, and locks can be used to ensure thread safety when accessing members of the class. This is accomplished by using a static local variable inside a static member function to hold a single instance of the class. The Meyers' Singleton takes advantage of the fact that the initialization of static local variables inside functions is guaranteed to be thread-safe. The Meyers' Singleton disallows copying and moving of the class, and makes member constructors and destructors private.

M. Power

The MCU will receive regulated 3.3, 5, and 12 V power from from the power subsystem. This power will be routed to various modules and other subsystems.

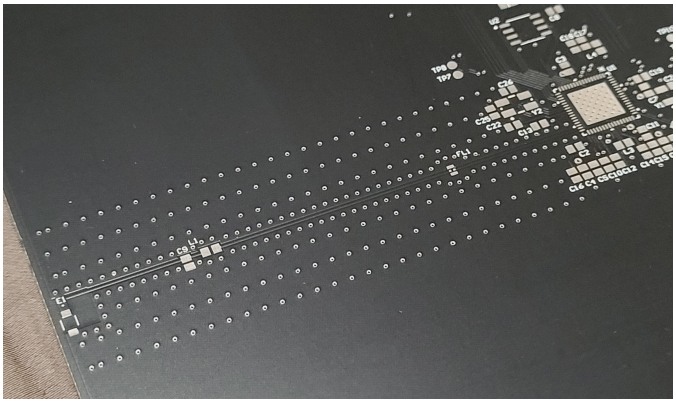

Fig. 3. The coplanar waveguide designed to carry a 2.4 GHz signal surrounded by via fencing.

N. LEDs

The system shall be able to make use of its onboard LEDs for notifying the user or developers of system states. When the system is in active mode, the green LED shall be solidly illuminated. When the system is in low power mode, the green LED shall flash 1 time for a period of 0.5 seconds, every 2 seconds. When the system is in critical power mode, the red LED shall flash 2 times for a period of 0.5 seconds per flash, seperated by 1 second between each flash, every 30 seconds. When the system is in shutdown mode, no LEDs shall be active. When the system is starting up, the green and red LED shall be solidly illuminated until the startup sequence is completed and the system transitions into a different power mode.

O. Printed Circuit Board

Our controller subsystem was developed and implemented on a Texas Instruments LaunchPad development board. An initial control printed circuit board (PCB) was designed, fabricated, and assembled (partially seen in II-O). The chip version of the CC3220 (CC3220S) was used requiring a 4-layer impedance-controller PCB, with waveguide and external antenna, as well as oscillators, numerous bypass capacitors, and inductors. A revised PCB was designed using the module version of the CC3220 (CC3220MODAS). The PCB design was simplified, requiring only a few bypass capacitors instead of the numerous other components to support the microcontroller found on the first revision of the board.