Garden Automated Rain/Daylight Executed by Near-Infrared Sensing

Nicholas Chitty, Brendan College, Scott Peirce, 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.

A. Goals

The following are the goals the team set for themselves with this project. Starting with non- negotiable goals:

1) A spectrometer that operates in the 400nm to 1700nm band with a spectral resolution less than 50nm and signal-to-noise ratio greater than 2.
2) The ability to automatically feed water into the garden bed.
3) Serve data to the user.

Moving on, the team had goals they hoped to accomplish:

1) Power the garden bed completely from solar and battery power.
2) Serve data to the user via a website.

II. High-Level Design

Based on the goals listed in subsection I-A this section will discuss the high-level thought process the team undertook in achieving those goals.

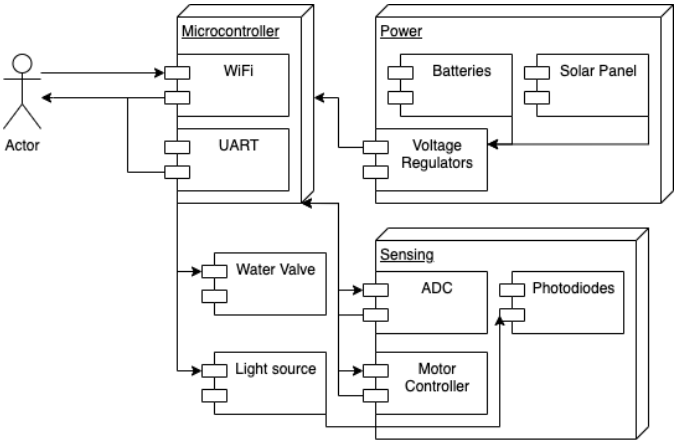

Fig. 1. The large systems and their interfaces and submodules

Figure II the actor is any user. The team labelled both UART and WiFi blocks as interfaces to the user because as of the time of writing, the web server is not serving live data. In the figure, sensing is used interchangeably with spectrometer.

A. Controller Subsystem

The controller subsystem would handle all of the I/O between each subsystem, control other ICs, record and store sensing data, as well as command and control the overall system. It would need to perform these actions while only sipping power from our battery, and have wireless capability. A real-time operating system (RTOS) would be needed to schedule and supervise the tasks needed to operate a spectrometer, tend to the user's flora, and communicate with the user and APIs wirelessly. All of this is relatively low-cost, this owed to the fact that no data processing will take place on-board the controller subsystem. The controller will command the sensing subsystem to send it data, and simply store this data to be interpreted at a later time.

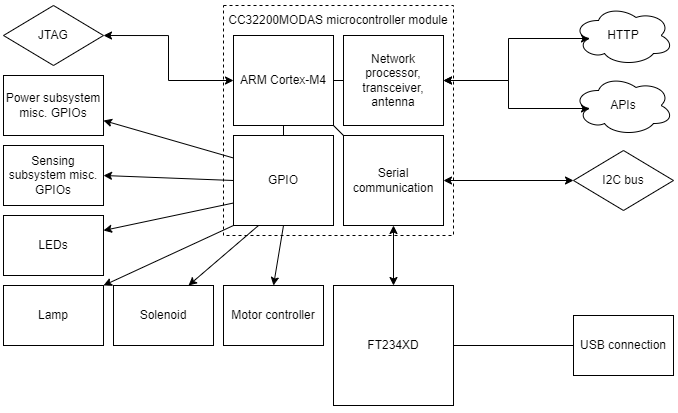

Fig. 2. Controller subsystem block diagram.

B. Spectrometer

In most instances, spectrometers are designed using charged couple devices in an array which some component such as a prism or diffraction grating will shine upon breaking the source into its components. This approach would be prohibitively expensive for such a consumer-oriented garden bed. The solution was to use either a photodiode or photoresistor to measure the magnitude of the of optical power in a given area[1]. This sensor could be moved in order to give the effect of shining the source on an array with the only major downside being the duration needed to scan. Find below a generalized block diagram for how this subsystem works.

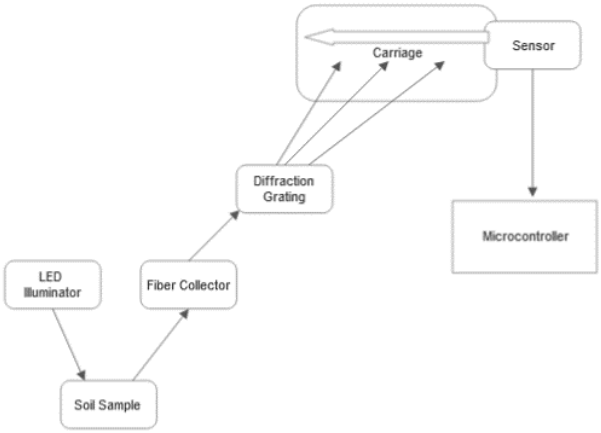

Fig. 3. Diagram showing the major components of the sensing subsystem.

III. Major Components

The components found in this section make up the logical boundaries between systems and their subsystems.

A. Microcontroller

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 operates 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.

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. Photosensitive device

The first major decision regarding components comes down to the choice between using a photodiode versus a photoresistor. The main advantage to photodiodes is their increased sensitivity. These means that with a lower optical power on the photosensitive surface, the electrical output should be higher by comparison to the photoresistor. This is a large consideration when understanding that soil will have a very low albedo (fraction of light a surface reflects) which means that every photon that hits the detector needs to have an effect. Thus the team chose two different photodiodes, the BPX 61 for the visual spectrum, and an InGaAs photodiode from Thor labs for near-infrared light.

C. Light source

The selection of the light source is important in that the light source needs to emit the wavelengths of light to be measured in roughly equal quantities. Based on the team's research tungsten lamps are a good source of broad spectrum light especially at optical power levels with respect to the necessary electrical input.

D. Linear Rail

Referring to our high-level design, the team needed a linear rail with a sled that the sensors could be mounted to. The team found such a device on Amazon.

E. Water valve

As part of the goals, the team wanted to be able to control the flow of water into the plant bed. To accomplish this, they chose a solenoid valve whose rated pressure was well specificied for use with home water pressures.

IV. Hardware Detail

In section III, a high level overview of components and features were given. This section will discuss the implementation details of the chosen components.

A. Spectrometer

The spectrometer is comprised of 5 major components, the photosensitive device, the linear rail and motor controller, the analog-to-digital converter, and the light source. The light source is connected via a Darlington transistor as it is driven with a 12V supply @ 1.7A. The darlington transistor will keep separate this electrical supply from the digital logic to control the lamp. The light source is fed into a fiber optic cable and pointed at a patch of soil which will reflect light into a secondary fiber optic cable. This project involves a large diffuse source, the illumined soil. The Fiber collimator will be attached to the fiber cable via their SMA connectors, then the collimator will be set in an acrylic block so that it rests 8.06mm above the soil. The block will also have a slot for the Light source to illuminate the soil at an angle, similar to the arrangement of the source and sensor in a computer mouse above a mousepad. The signal will travel through the fiber and up into the spectrometer housing, where the other connector of the cable will be mounted in place. Another collimator will take the output beam and collimate it so that it propagates through free space into the housing. The collimator has an output beam diameter of 1.7mm. This planar wavefront will strike the diffraction grating. We will call to the first optic cable as the source, and the second cable as the signal. From the signal cable, the light is transmitted onto a diffraction grating. This diffraction grating will split all the wavelengths of light across a rotational area of 47°. At this point, the team can do simple trigonometry to translate a linear position to a range of wavelengths being scanned. The team chose for the sensor to be at a distance of 30mm from the diffraction grating.

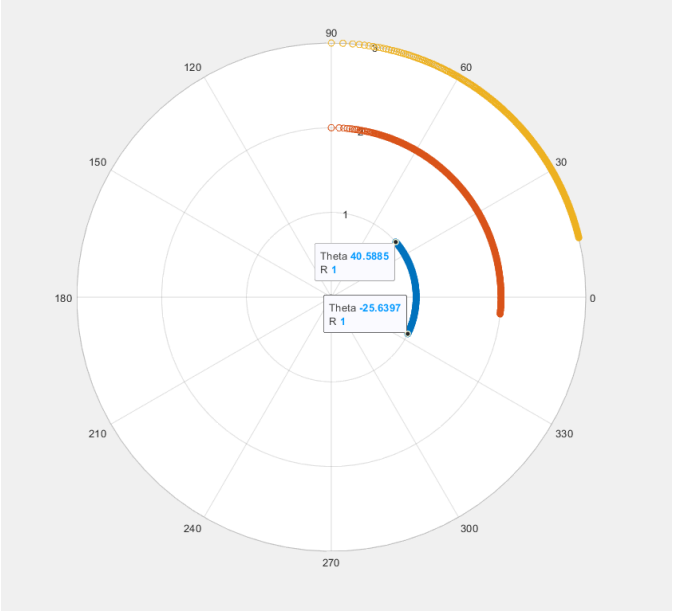

Fig. 4. The diffraction angle of 1st through 3rd order diffractions.

It is at this point that the spectrometer design enters the electrical domain. Due to the photosensitive device being a photodiode where the current is a result of the optical power on the sensitive area, the team built a current-to-voltage converter which would be fed into the analog-to-digital converter. The primary issue at this point is just how small the optical power of the light is. Given that the sensors have a translation of 1nA/1mW of power, we needed to design a high gain, low noise converter. We chose 100e6 for the gain meaning that 1nA would translate to 100mV which is over double the resolution of our 16-bit ADC.

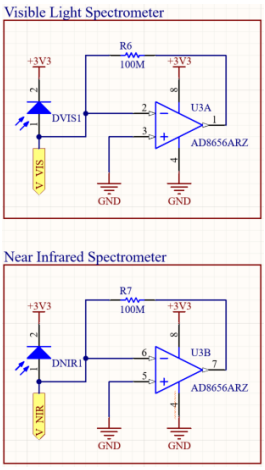

Fig. 5. The schematics for the current-to-voltage converter.

The net labelled `V_NIR` and `V_VIS` are sandwiched between ground planes on their track to the ADC to reduce noise caused by electrostatic discharge or other analog signals (such as the motor current). Before being read by the ADC the team designed a low pass filter to try to attenuate the noise above 60Hz. The value of 60Hz was chosen as the cut off frequency because during testing the oscilloscope showed a signal frequency in that range that would be detrimental to the readings.

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 is able to measure between 0 and 3.3 V from the output of the sensing subsystem's photodiode op-amp circuit.

Each photodiode's circuit produces a voltage 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:

$$\frac{(3.3-0)\,\mathrm{V}}{2^{16}\,\mathrm{steps}} = 50.35\,\mu\mathrm{V/step} \tag{1}$$

These steps are used to measure OH composition and nutrients in the soil. The MCU directly controls 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.

B. 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 IV-B). 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.

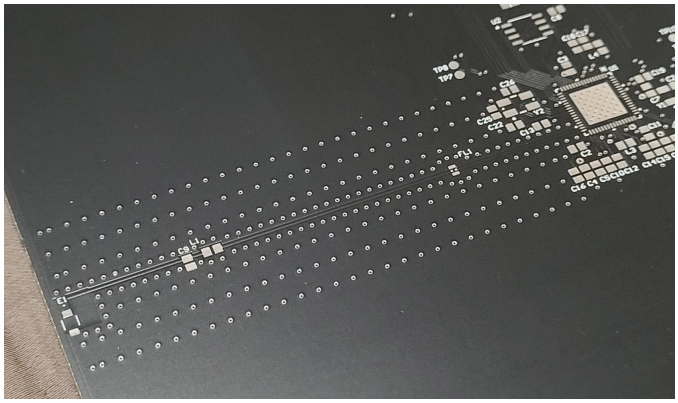

Fig. 6. The coplanar waveguide designed to carry a 2.4 GHz signal surrounded by via fencing.

A revised PCB (seen in IV-B) 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. All other components found on the first version of the board were kept (e.g. molex connectors, LEDs, JTAG connector, FT234XD, etc.). In addition, pinouts were added for GPIO, I2C, and power to aid in debugging.

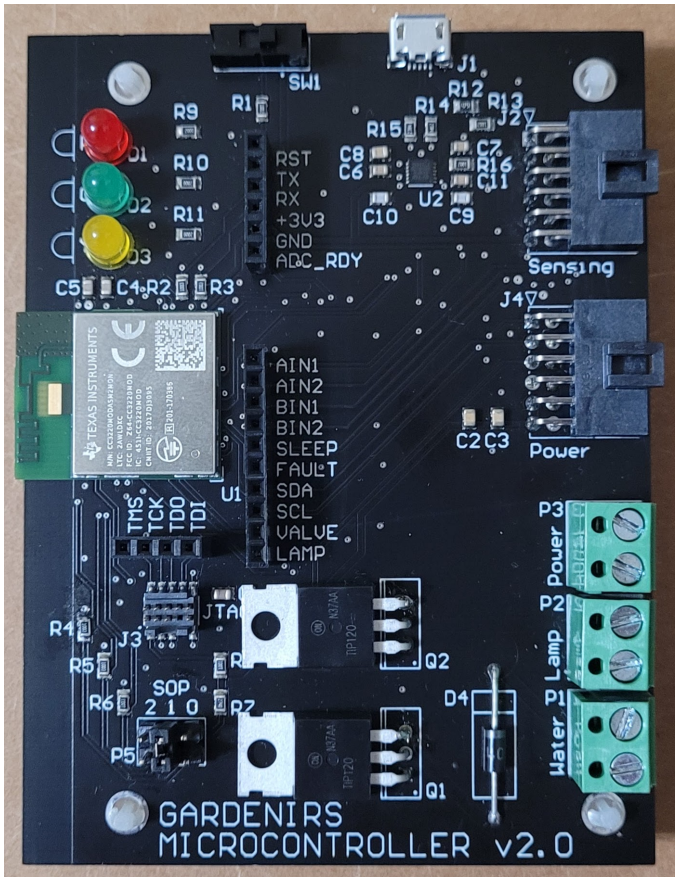

Fig. 7. Version 2 of the MCU PCB.

```
class Water {
    public:
        static Water& instance();

        // Disallow copying
        Water& operator = (const Water&) =
delete;
        Water(const Water&) = delete;

        // Disallow moving
        Water& operator = (Water&&) = delete;
        Water(Water&&) = delete;

        // Class functions

    private:
        Water();
        ~Water();

        // Class variables
};
```

Fig. 8. An example of the Meyers' Singleton pattern implementation within the project.

V. Software Detail

The microntroller serves as the glue holding this design together. In this section the means of integrating the various subsystems via software will be discussed.

A. 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 V-A). 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[2].

B. 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.

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

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.

$$\frac{1\,\mathrm{MiB}}{520\,\mathrm{B/measurement}} = 2016\,\mathrm{measurements} \tag{3}$$

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

$$2016\,\mathrm{measurements} \times \frac{1\,\mathrm{measurement}}{15\,\mathrm{min}} = 30240\,\mathrm{min} \tag{4}$$

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

C. Software Logic Flow

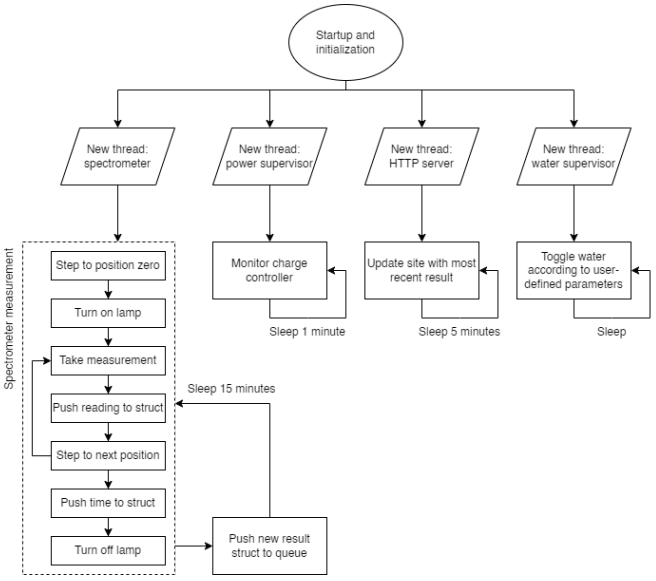

Fig. 9. Software logic flow diagram

The RTOS monitors 4 threads, each dealing with vital functions of the product. The first thread deals with the sensing subsytem and operating the spectrometer, as well as recording results. When a measurement is needed, the lamp is turned on and the spectrometer is stepped through every position to take a voltage reading that is placed in a struct. When the last measurement is taken, the light is turned off, the time is recorded in the struct, and the measurement struct

is pushed to a queue. The HTTP server thread reads the results from the queue and posts them for the user to see. The power supervisor thread reads from the charge controller every minute. If it encouters an unsafe situation, it is able to command the system's power state to resolve the situation. Lastly, the water supervisor thread controls the water solenoid, turning it on and off according to parameters input by the user.

D. Connection

Our product broadcasts a WLAN in AP mode that allows the user to connect to the product. The user is then directed to a web portal hosted by the MCU's internal HTTP server, where they will be able to view information, sensing data, and telemetry related to the product. Hosting a web interface would allow unanimous adaptation of our product for home users.

E. Libraries

The C++ standard library (as defined in C++14), the POSIX library, and the Texas Instruments SimpleLink CC32xx SDK were used for this project. No 3rd party libraries were used.

F. 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).

G. Motor Controller

Four GPIO lines are 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 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 V-G). With a speed of 80 MHz in active mode, this would mean we're able to write to GPIO in only 28 processor cycles.

$$350\,\text{ns} \times (80 \times 10^6 \text{cycles/s}) = 28\,\text{cycles} \tag{5}$$

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

Fig. 10. Toggling GPIO using driverlib results in a very short (350 ns) delay.

To drive a DC motor we similarly needed a device to separate the digital logic from the back emf of the motor. This is done through the use of the motor driver. The timing of the signals is the most important part as the driver is just using the current and voltage source designed specifically for the motor in the timings given by the input pins. Wrong timing leads to the ability to skip tests leading to indeterministic behavior. The issue is the the CC32xx SDK abstracts away a lot of the fine tunability of setting hardware registers. To resolve this, the team used the specific board's hardware driver implementation to set the output on the pins with a delay of only 28 cycles.

H. 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.

REFERENCES

[1] Y. Cao, "A $500 diy near-ir spectrometer that would sell for $10,000.".
[2] S. Meyers and A. Alexandrescu, "C++ and the perils of double-checked locking," *Dr. Dobb's Journal*, July/August 2004.