Smart Plant Life Monitoring System

Jeremy Maxey-Vesperman
Electrical and Computer Engineering Department
Kettering University
Flint, MI 48504

Zachary Goldasich
Electrical and Computer Engineering Department
Kettering University
Flint, MI 48504

Girma Tewolde
Electrical and Computer Engineering Department
Kettering University
Flint, MI 48504

Abstract— **In this age of technology, people want to be always connected to all things in their lives, which for many people include their house plants. The growing adoption of internet-of-things (IoT) in consumers, commercial, industrial, and services applications means the cost and accessibility of such technology is fast improving. This paper presents the design and development of a monitoring system that is capable of reporting various metrics of a common houseplant. These metrics include light intensity, temperature, and soil moisture. In order to facilitate the reporting process, the device implements an SMTP-based system to communicate with its owner remotely. Furthermore, in order to extend battery life, many power-saving features starting with a hardware level control and working up to the software level are implemented.**

Keywords—light sensor, temperature sensor, moisture sensor, low power operation, internet of things (IoT), mobile App

I. Introduction

Taking care of houseplants is a popular hobby among many people, but actually handling their watering schedules and the like is often demanding—more so with some species of plants over others. In order to handle this in a manner friendly to home automation, this project implements a “plant life sensor” that monitors the soil moisture, light, and temperature levels of the plant it’s attached to. This is accomplished through the use of a Light-Dependent Resistor (LDR), also referred to as a photoresistor, a Negative Temperature Coefficient (NTC) thermistor, and soil moisture sensor that measures the resistivity of the soil to determine soil electrical conductivity (EC).

LDRs vary their resistance in accordance with light levels in the room [1] . Usually, LDRs are only utilized as a cheap way to differentiate between light and dark. This is mainly due to the large tolerance range that they are manufactured with; A light intensity of 10 lux could result in as low as 50kΩ or as high as 100kΩ, in the case of the LDR utilized in this project [2]. However, by taking resistance measurements at varying light levels compared against a lux meter and utilizing statistics, a calibrated light intensity sensor can be created. This correlation between resistance and light intensity can be expressed as:

$$L = R^{m} * 10^{b} \tag{1}$$

Where, L represents light intensity in lux, R is resistance of the LDR, and m and b are the slope and intercept of a best fit line equation obtained from the log-log (Lux vs. R) graph of the calibration data [3].

Similarly, NTC thermistors vary their resistance in response to temperature. This response has inverse relationship, meaning that the resistance decreases as temperature increases. There are a few various approaches to correlating temperature to this resistance. One such method utilizes the simplified Steinhart-Hart equation:

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{\beta} * ln\left(\frac{R}{R_0}\right) \tag{2}$$

Where, T is current temperature in Kelvin (K), T_0 is room temperature (K), R is the measured resistance, and R_0 is the resistance measured at T_0. This also utilizes a constant, β, that is provided by the sensor manufacturer [4] .

The driving theory behind soil EC measurement is that there is a strong correlation between how well the soil conducts and its moisture content. Additionally, this measurement can also be used to determine soil particle size as sandier soils have higher resistances [5] .

The following describes the major goals of the Smart Plant Life Monitoring System design:

- It must have a relatively small footprint in order to be unobtrusive
- It must implement as many power saving strategies as is reasonably possible without compromising functionality, as a plant monitor that constantly needs recharging would defeat the purpose
- It must have a relatively low price point. The justification for this is twofold:
  - A lower total expense makes the prototyping stage affordable to work with at a low budget
  - Smart plant monitoring system is something of a niche market, but one that remains untapped for one major reason: cost. Most commercial devices out on the market today tend to abuse the fact that the IoT is a fad and drive up their prices accordingly. By keeping the pricing of the project’s components relatively low, we believe that the device stands a real chance at satisfying consumer demand by undercutting the (overpriced) competition.

II. Hardware Design

The schematic diagram in Fig. 1 provides an overview of the hardware design for the project. The primary component controlling the entire device is an Adafruit HUZZAH ESP8266--a small, inexpensive 80MHz microcontroller unit. This MCU has a total of 80kB Data RAM, 35kB Instruction RAM, and 4MB SPI flash. It also contains an RTC module with a small

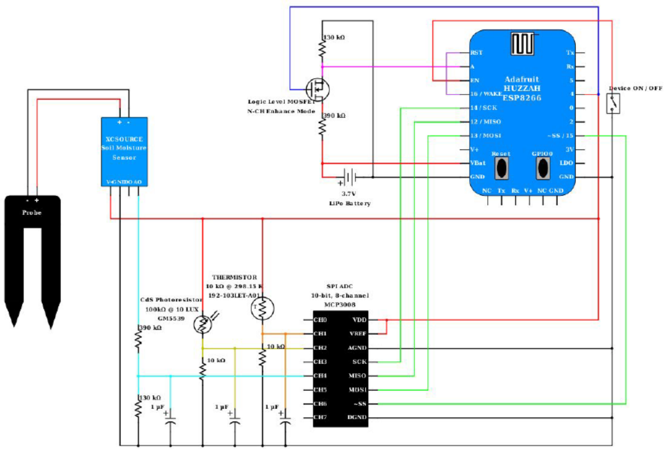


Fig. 1. Hardware Schematic Diagram of the Smart Plant Monitoring System

amount of RAM to hold WiFi connection information and other variables that the programmer would like to store between Deep Sleep cycles, as this remains powered during Deep Sleep. Furthermore, the device has 9 GPIO pins and 1 analog pin.

The most important feature for this project is a built-in Wifi, supporting 802.11n with TCP/IP stack, which permits for the implementation of an "email service" to send updates about plant status without increasing the cost of the device further by adding a clunky Wifi shield onto the list of parts.

All sensors are powered from a digital I/O pin. This allows the MCU to turn the sensor circuits on/off to reduce power consumption as these would constantly consume power if connected directly to the battery. The ESP8266 also has an internal 1.1V LDO that its ADC chip uses for voltage reference which allows for battery voltage to be read. Finally, a switch between EN pin and GND allows the user to turn the entire device on/off in case the device is not in use. When closed, this pulls the EN pin low and disables the 3.3V LDO voltage regulator that powers the entire system.

The sensor components attached to the system are simple resistive devices—a photoresistor, thermistor, and soil moisture sensor. All sensors are configured in a voltage divider circuit to allow for easy resistance calculations. Additionally, a voltage divider circuit is constructed on the battery input and connected to the analog pin of the ESP8266 to allow for battery voltage measurements. This is particularly important as LiPo-based battery (which is what this project intends to use as a power source) cells will become damaged if they are discharged below a threshold voltage. A logic-level enhancement-mode NMOS is placed in the middle of this voltage divider circuit. Its gate is connected to digital pin 4 to allow the MCU to turn the circuit on/off to reduce power consumption when battery readings are unnecessary.

An SPI-based dedicated analog to digital converter chip, MCP3008, handles reading of all the relevant sensor values. It references the output voltage of digital pin 4, which all the sensors are powered from, to make measurements.

The following calculations were obtained for maximum current through each sensor circuit, assuming that the LDR and NTC thermistor have a value of 0Ω (conservative approach as this is unlikely to be the case) and V_{in} of 3.3V:

- MCP3008 ADC I_{max} : 500 µA
- Photoresistor I_{max} : 330 µA
- Thermistor I_{max} : 330 µA
- Moisture sensor I_{max} : 6.346 µA
- Battery voltage sensing circuit I_{max} : 6.346 µA

As the sensors are connected in parallel to the same GPIO pin (aside from the battery sensing circuit), their total current was calculated by summing the individual currents:

- Total Sensor I_{max} : 1.173 mA

The ESP8266 has a max current draw of 120 mA while transmitting data and a max current draw of 20 µA while in Deep Sleep Mode. Assuming that the device is configured for hourly updates with sensor readings taking approximately ~1sec, transmission taking approximately ~60 sec, and the device entering Deep Sleep Mode for the rest of the hour, an average current draw of 2mA was obtained. With a 2200mAh LiPo battery, the theoretical lifespan of the device (before requiring recharge) was determined to be 32 days [6]. The authors feel that this meets the requirement set forth for a low-power system. Furthermore, the lifespan could be greatly increased by decreasing the message update frequency.

III. Software Design

Fig. 2 shows the general overview of how the entire system is designed to function. The software design for the ESP8266 MCU is relatively straightforward; all sensor readings can be measured via Analog-to-Digital Conversion (ADC) Modules, converted to appropriate units, and compacted into a single array of sensor values. As a result, the actual software logic can be broken down into a few important elements:

- Initialization
- Collect ADC Measurements
- Convert to appropriate units

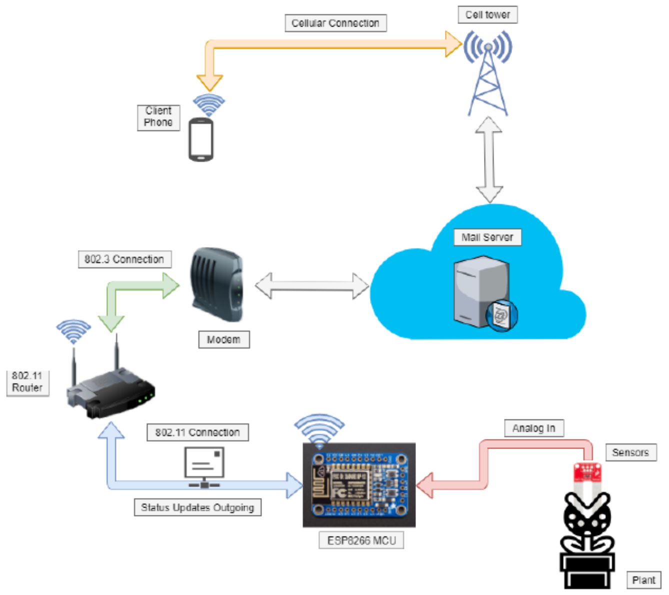


Fig. 2. Smart Plant Monitoring System overview

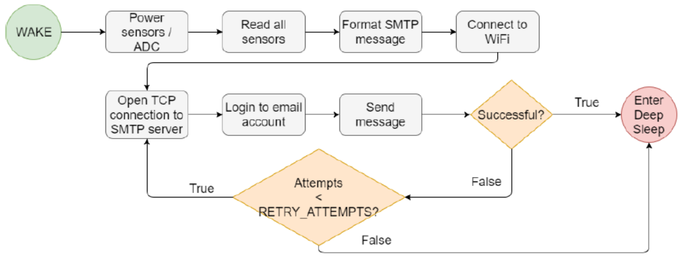


Fig. 3. ESP8266 software flowchart

- Format message
- Connect to WiFi router
- Send Email message
- Configure Deep Sleep With Timeout

The software design for the ESP8266 microcontroller is shown in Fig. 3. The Initialization phase covers the basic startup tasks; UART for diagnostics purposes is initialized and pins are configured. The Deep Sleep functionality for this particular microcontroller completely resets the system, such that it will return back to the initial Setup/Initialization stage starting point when it wakes up. The only way to store variables between Deep Sleep cycles is to either store in Flash memory, EEPROM, or the RTC module's RAM.

Communications with each of the sensors is handled through the separate ADC module; a library has been written for this purpose in order to enhance code readability and encapsulate much of the extra steps in wrapper functions. The unit conversion functions for each of the sensor's ADC measurements are also contained within the library.

After sensor measurements have been taken, the system waits until a connection has been established with a preprogrammed Wi-Fi network. Once this occurs, it begins attempting to send an email to the specified address and will repeat this operation until it succeeds or reaches the maximum retry attempts. Retry attempts increase the timeout to wait between TCP packets to attempt to resolve any communication issues.

The Deep Sleep phase exists as a termination point for the software logic; as mentioned, it resets almost the entire processor when going to sleep and restarts at the Setup phase when waking. A delay is configured using the onboard Realtime Clock and Calendar module, which currently causes the device to wake up once an hour. Due to some engineering flaws with this particular microcontroller, there is some drift involved with doing so (which is elaborated on further in the Conclusions section).

In addition to the ESP8266 software, a simple Android app was developed for testing purposes using MIT's App Inventor [7]. This essentially wraps the SMTP server's mailbox web page and displays it within an app, as shown in Fig. 4. Additionally, the app wraps a plant reference site to provide the user a convenient method to search for various plant lighting, temperature, and moisture level metrics.

IV. Conclusions

Overall, the project was a success. The core features of the project are functioning well as expected. The primary goal was to create an inexpensive and smart plant life monitoring system that was capable of maintaining relatively long battery life via power saving techniques. This has largely been accomplished; it would be a relatively simple matter to integrate additional features on top of the existing logic, as the current program logic is largely designed as a modular sequence of tasks repeated between every deep sleep cycle.

As far as project difficulties go, the primary issue with this project resided with the Realtime Clock & Calendar module. Though the microcontroller we chose to use does have this functionality, we discovered later that the module has serious drifting issues when in deep sleep mode. In a project such as this, the drift is fairly inconsequential on the actual operation of the device; however in a production model it would be necessary to source another microcontroller capable of maintaining the precise time intervals necessary for the application.

References

[1] L. Ada, 'Measuring Light', 2012. [Online]. Available: https://learn.adafruit.com/photocells/measuring-light

[2] WODEYIJIA TECHNOLOGY CO.,LTD, 'GM55 Series Datasheet - CdS Photoresistor Manual'. [PDF]. Available: https://www.sparkinter.com/pdf/120300-0056590.pdf

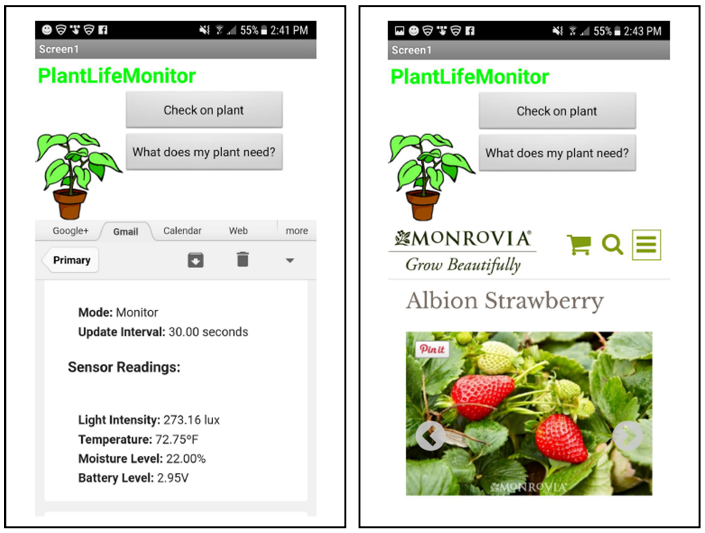
[3] D. Williams, 'Design a Luxmeter Using a Light Dependent Resistor', 2015. [Online]. Available: https://www.allaboutcircuits.com/projects/design-a-luxmeter-using-a-light-dependent-resistor/

[4] J. Corletto, 'Measuring Temperature with an NTC Thermistor', 2016. [Online]. Available: https://www.allaboutcircuits.com/projects/measuring-temperature-with-an-ntc-thermistor/

[5] R. Grisso, M. Alley, D. Holshouser, et. all, 'Precision Farming Tools: Soil Electrical Conductivity', 2009. [PDF]. Available: https://vtechworks.lib.vt.edu/handle/10919/51377

[6] 'Battery Life Calculator', 2017. [Online]. Available: https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-battery-life

[7] MIT App Inventor, 2019. [Online]. Available: https://appinventor.mit.edu/



Mode: Monitor
Update Interval: 30.00 seconds

Sensor Readings:

Light Intensity: 273.16 lux
Temperature: 72.75°F
Moisture Level: 22.00%
Battery Level: 2.95V

Fig. 4. SMTP wrapper page