Portable Plant Monitoring System

Dept. of Electrical Engineering and Computer Science, University of Central Florida

Syed Bukhari Alexander Meade Nicholas Poidomani Kareem Rahounji

***Abstract*—One aspect that is often overlooked when it comes to cultivation of plant life for either commercial, academic, or hobbyist growth is data measurements. Expanding on this when it comes to a plants growth cycles there is an abundance of potentially untapped data resources. It is our groups belief that if we can capture trends in data about the plant's environmental metrics while it is growing these could in turn be used to formulate better growth patterns for future generations or provide insight as to why previous plant life failed in a specific environment. Our project's goal is to create a system which would trivialize the recording of data into one robust system and show using a bundled software interface a way the end user could help visualize and represent these trends.**

***Index Terms*—MCU (Micro-controller Unit) , RTC (Real-Time Clock), SPP (Serial Port Profile), UART, SPI, I^2C**

I. Introduction and Background

In this section we set to establish the goal of the project as well as the design principles we establish.

When we set out with this project we wanted to come up with the design that made capturing a plant's environmental metrics in an efficient manner. Once we had accomplished this we wanted to create a software tool that would showcase the information one could glean from this data collection and elaborate on how this information could potentially be used to aid in future generations of horticultural projects. Before we could establish the system, we first needed to come to a consensus on what environmental metrics would be most relevant to the average plant's growth cycles to establish clear guidelines for us to adhere to in the design and fabrication of the system.

For simplicity purpose we decided to go with some universally common environmental metrics shared by plants as our first design principle, these include soil moisture, light levels, temperature, and air humidity. While there remains a host of other potential selections, as a group we felt these choices would provide a nearly complete picture about the plant's development during its growth in a general case without incorporating measurement types that might be too specific and clutter the data sets, for instance a barometer was considered for higher altitude plants however in the average case we felt this didn't provide much of a benefit to data collection.

The final aspect of our project we considered crucial is the portability and power efficiency of the system. These were kept with a higher priority in mind due to the nature of horticultural projects and sciences as these endeavours are often in situations where they might not have a continuous source, such as AC outlet, supplying the power needed to drive the device. This led to our second design principle to not only make our system battery operated while being as low-power draw as possible to accomplish the given tasks. Tying into the previous consideration is the fact the plant under observation might be transported. This constant displacement could cause issues to the plant's growth if the system was not designed with being trivially relocatable in mind, so as our third design principle we went with an emphasis on portability .

II. System Design and Components

In the following section we seek to establish our core components and the outlying design of the system.

The cornerstone of our system is the environmental sensors divided by which metric they can record. The first is the soil moisture sensors which was the SEN-13322. This product by SparkFun is a capacitive volumetric sensor meaning it uses an electrical signal to check the resistance of the soil between it's two gold plated leads then based on an analog threshold comes up with a percentage ratio of soil moisture [6]. This sensor was chosen because it was economical and highly accurate, within ±5% and it's ability to provide a quick and easily accessible data reads.

Our second sensor is the light sensor which was used to see if the plant is receiving a sustainable amount of light for the photosynthesis process. For this sensor we had two major categories of sensor to consider ambient light sensor and photocell diodes. While ambient light provided an exact environmental metric reading in either lumens or lux, they broke one of our constraints of being able to provide a low-power environment whereas the counterpart of photocells did not. A photocell is a light controlled variable resistor which based on the amount of light the sensor currently receives it increases the resistance creating an analog signal which can be converted to a digital threshold. [1] For our selection of photocell we went with the Everlight EAALST05RDMA0 which doubled as both a low cost and low-power selection while providing wavelength support up to 630nm making it ideal for the types of light a plant would be exposed to.

Our next two environmental metrics are temperature and humidity which are perhaps the most basic of metrics in order to record a plant's growth. Due to their similarity in design for how they are measured via a sensor the next set of potential sensors we looked at were bundled as dual feature of being temperature and humidity sensors. Similar to the light sensors prior there existed many options however the matter was finding one that did the job with a low-power draw while still maintaining a specific level of accuracy and performance we required. This was found in the DHT11 senor which is a line of ultra-low cost dual packaged sensors often found in HVAC or Automotive applications. [2] Providing us with a digital sampling rate of 1Hz and an accuracy rating of ±5% same as the soil moisture sensor it proved to be the most logical choice.

Moving on from the sensor selection we now discuss the overall system architecture and how we plan to achieve our specific design goals. First and foremost we wanted our system to be designed with the ability to switch seamlessly between a primary and backup power source to ensure continuous uptime. Once this had been accomplished we rely on a singular low-powered MCU to drive the entire system. This MCU would drive the sensors and record the data readings take from the sensors. However we needed a way to control how often we should take readings from the sensors as leaving them in a constant polling state is very demanding on our power supply. In order to combat this situation we decide to implement a RTC module to provide us the ability to sleep the board peripherals during non-essential operational phases we would define the duration of as needed. We also considered if the user wanted to capture a data-set at time outside of these phases and thus provided a push button for which the user can press and instantly poll the current environmental readings. Another factor considered for the user end is the ability to view the current environmental readings without the need of loading a data-set to a program and our solution to this was to incorporate an LCD screen that would display the current readings on the previously mentioned user button press allowing both instantaneous data capture and display simultaneously.

The next component we required was a way to transmit the data-sets from the system itself to a external supporting software package that would allow the user to process the data-set and come to conclusions about trends and growth cycles with their observations. We decided for accessibility this would best be accomplished with a form of wireless communication as requiring external wiring from one system to the other would prove cumbersome and violate our portability requirement. Comparing several forms of wireless communication we eventually settled on Bluetooth SPP due to a good combination of energy efficiency, broadcast range, and data throughput. This system is represented in Figure 1 below to help visualize the overall system architecture.

Our final component decision rested with the

MCU itself. This was perhaps one of the most crucial selections as it is the driving force behind the entire system. Keeping our design requirements in mind we had several conditions we had to meet such as making sure the MCU was a low-power device, but also that it still had the ability to drive all the systems peripherals such as the sensors and communication methods. We also wanted to select a chip would had a fair amount of documentation for easy of integration into the PCB and additionally provided multiple ways for uploading the hex code onto the board so that we could dedicate more time to the construction of the algorithms and subroutines. These requirements drove us to select the ATMega 2560 which provide to be the best compromise of performance to requirements providing a trade-off in terms of hardware feature limitations to meeting all of the above conditions we set for the MCU.

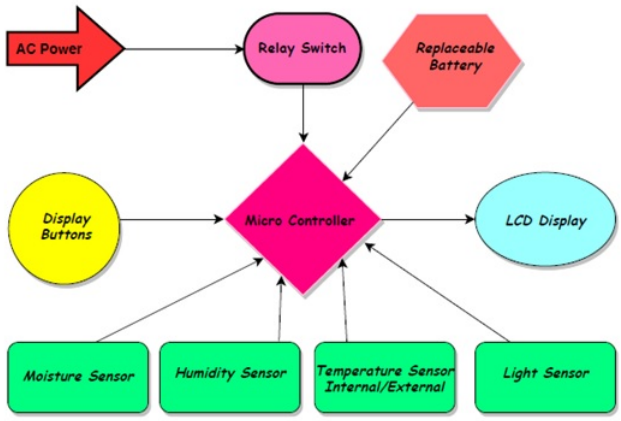


Fig. 1. System Diagram

III. Hardware Detail

Having established our design and component selection we now explore the electrical and physical construction of the system.

With regards to the hardware we broke the system design into three major sections the power delivery circuit, the central logic PCB, and the physical enclosure of the system. The first subsystem is the power delivery circuit listed in Figure 2 below. This circuit while not appearing overly complex serves multiple purposes in our project. First it offers protection while Figure 2 shows a 12V constant power source the circuit itself is rated to handle

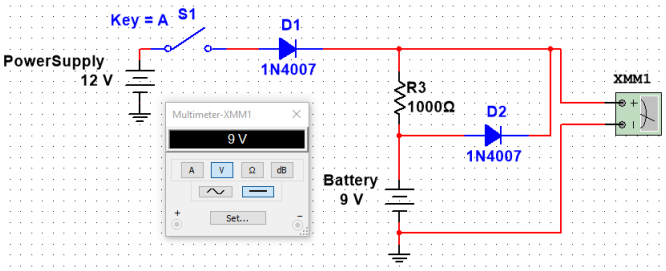


Fig. 2. Power Circuit

up to 30V input before sending the voltage to a step-down transformer in the PCB. Next it also acts as our automatic switching between power sources, when plugged in the switch S1 will shut and provide current to the rest of the board, but as soon as this power source is interrupted the battery backup will automatically take over ensuring a steady stream of power to the system so long as one of the two sources are present. It should additionally be noted that the particular diodes selected were chosen with the lowest power draw to current ratio keeping our low-power parameter in mind.

The next section is the central logic of the PCB, which while still on the same physical board preforms an entirely separate set of functions from the previously mentioned power delivery circuit. The central logic part of the PCB is actually the one responsible for stepping down the voltage from the supplied power circuit to the various levels needed throughout the board such as 5V for the MCU and several modules and 3.3V for the various sensors and other miscellaneous components. This section of the board also contains all the communication lines needed for various aspects of the project such as the UART lines for the SPP module to transmit the data, the I^2C lines for the RTC module to provide accurate timing to the system as well as the various analog and digital communication lines to the MCU for the LCD and various sensors the system needs to communicate to.

The next final portion of the hardware design was the physical medium in which we would house the electronics which we referred to as the enclosure. One of the first things considered with the design of the enclosure was the type of material it would be constructed out of. We wanted to keep in line with design requirement of being lightweight, however we also had some concerns that due to the nature of the external environments the system

could encounter we wanted to be able to provide some level of water proofing to ensure the safety of the electrical systems. The solution to this was selecting underlayment plywood which was both lightweight and had a certain degree of protection against water which we would then increase by adding an additional layer of waterproof treatment coating.

Next was the layout of the enclosure which we decided to separate into two individual parts the base and the cover. The base would act as the primary housing section for the electronics and would be the primary point of protection of the system. It's design consists of a 1.2ft square with a 6in hole in the middle for setting the plant into. The connections are made by corrugated edges laser cut into the wood which were then further sealed by a wood adhesive to ensure further structural stability and water tightness. Surrounding this central hole we have 3 small bore holes which are the areas where the sensors are mounted and then the wiring is fed down into the PCB below.

Fig. 3. Prototype Enclosure

The next portion was a part derived to provide a level of protection for the plant itself and it is the cover of the base. This section would be a rectangular cover which would close around the central circle and the surrounding sensor mounts which would provide external protection for the plant. One thing taken into consideration was the ability of the plant to get oxygen naturally so for the design we decided that two walls in parallel would be mesh netting to allow natural airflow while the other two would be the same wood as the base to provide structural support additionally connecting the two wood walls would be a wooden truss for more stability and additionally acts a potential mounting point for a proposed light strip to allow the user to provide the plant with artificial light to supplement sun during transpiration. The top part of this figure would also be covered with mesh to allow for addition oxygen and light to reach the plant more efficiently. Finally we wanted this cover to be removable to trivialize access to the plant as opposed to being constantly mounted, in order to work this in we opted to have the wooden walls attached via Velcro strips mounted to the base which allows the cover to be placed and removed at the user's demand.

IV. Software Detail

Since the physical assembly had been outlined in this section we explore the software logic and driving force behind the system.

The first detail we should discuss when it comes to the software is the data itself and what type of information we are looking to store and transmit. Obviously we want to capture the sensor readings but what is the best way to encapsulate all the various readings we want. For the sensor readings we decided it best to keep the readings all as Int16 as this allows for proper data preservation across any modern system that supports little endian. Regarding the time-stamping of the readings we decided to use a C++ object referred to as DateTime, these objects allow us to capture dynamically measure clock dates or times and then store and represent them as string objects allowing for a smooth transfer via platforms.

Because of the design of our system we split the software into two primary parts the embedded section and the external supporting software. The first portion designed is was the embedded system which handles the core of the project from driving the system to collecting the data. Once the system is powered up it loads up an initialization state in which it creates all the variables needed for collection and then begins setting up the external devices to be driven and operated. Once all the initialization has been established we set the system into a standby low-power state in which it will only be awoken by one of two conditions, the first being the RTC senses a certain amount of time has elapsed and raises an interrupt flag to reflect this change in time while the second is the user presses a button to raise a separate interrupt flag. In both cases the same effect is achieved via different interrupts which powers on the sensors and other required

lines for a short duration in order to take several bursts of readings before storing them to be sent off to the external program. It should be noted that from this standby state there exists another option for the user to send the power down signal to the system via button press which will turn all devices attached and the system itself completely off save for the interrupt line the button is attached to.

Once data has been stored into the system memory there exist another two additional options for sending the signal to transmit the data-set over UART via the SPP. These two conditions are mirror counterparts to the first pair of decisions either letting the RTC signal an elapsed time interval predetermined to transmit at that point or allowing the user manual input to control the transmission on demand. Regardless of the decision taken to get to transmission state after the data-set has been sent off the system goes back into its standby state and awaits either another reading pulse trigger or for a power down signal we represent this state machine below in Figure 4.

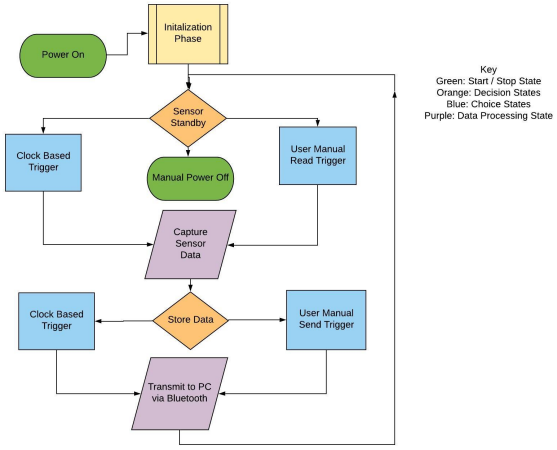
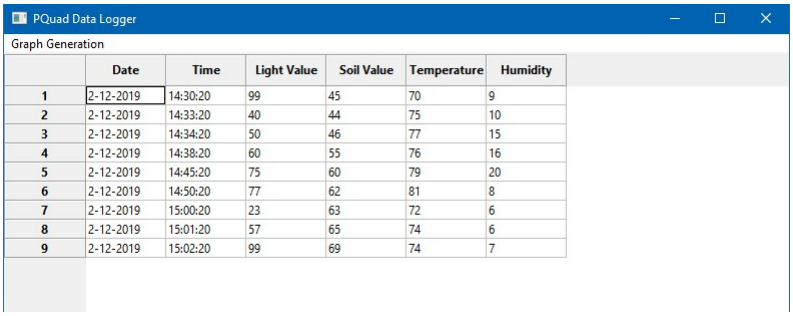


Fig. 4. Embedded Diagram

Transferring the data-sets over only encapsulates half of our software framework once the stream is sent to the user's computer via SPP there is another host of processes which happens in order for the user to gain information about the plant's growth trends. What happens first and simultaneously along side the embedded SPP is the computer receiving the data via a COM port connection. In order to parse the data for usability we utilized a framework called GoBetwino which allowed us to print the readings from the serial port directly into a formatted file combining both data storage and sanitation into one combined step and storing it in a predefined location on the file system [3]. Once the data log has been stored in the file system it is ready to be processed in the supporting software packaged with out system. The software is a compiled python script which can be run either as an executable or through Python's shell and only requires the user to have the Python interpreter installed. Upon execution of the script a GUI window will greet the user and from here the user must select the graph generation button which upon pressing will open up a file browser. The user navigates the file system to find the desired log file they wish to extrapolate data from and upon selecting it will be prompted to select the type of graphical data generation they which to construct. An example of the software running is shown in Figure 5.



	Date	Time	Light Value	Soil Value	Temperature	Humidity
1	2-12-2019	14:30:20	99	45	70	9
2	2-12-2019	14:33:20	40	44	75	10
3	2-12-2019	14:34:20	50	46	77	15
4	2-12-2019	14:38:20	60	55	76	16
5	2-12-2019	14:45:20	75	60	79	20
6	2-12-2019	14:50:20	77	62	81	8
7	2-12-2019	15:00:20	23	63	72	6
8	2-12-2019	15:01:20	57	65	74	6
9	2-12-2019	15:02:20	99	69	74	7

Fig. 5. User Interface

Should they wish the user may close this option and view the data in its tabular representation or continue the graph generation by selecting the type of graph they wish to display and then choose the X and Y axis parameters for the graph. After selecting the final parameter the graph will then generate based on the user's input specifications and from here the user is allowed a variety of options such as zooming in on the graph, adjusting the graphs scale, and saving an image format of the graph to the file system. After the graph generation is finished the user may also view the tabular data representation or go back to select another file for graph generation the flow of this logic is represented in Figure 6.

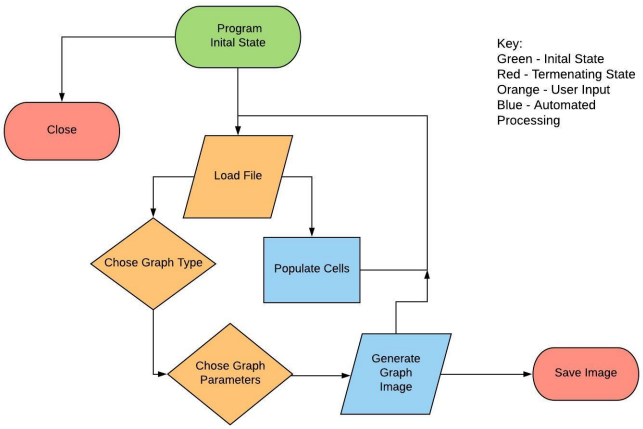


Fig. 6. Supporting Software Flow

Finally a late addition to the design was a semi-automated monitoring system that utilizes a python script as a sniffer of the selected COMPORT for signals from the embedded end. After the user has entered their G-mail and app password, a security feature implemented to protect Google authentication, the user can begin the monitoring of the COMPORT. They may set it to enter in the listen state in which it will listen for a predetermined duration and if during this point it picks up a distress code it will automatically match the code to a predetermined email which sends off to the user's entered email. This process is encapsulated in 7 below.

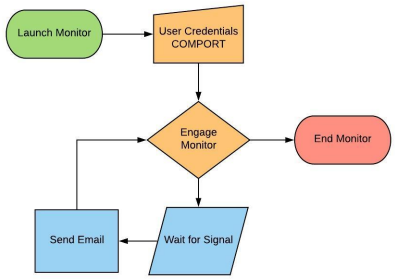


Fig. 7. Monitoring System Flow

V. Standards

It's pertinent to any project to establish the set of guidelines and regulations that your design must adhere to in order to assure quality.

During the course of our system's development cycle we had to take into consideration what sets of standards we adhered to ensure the most efficient design of the system. With regards to safety of user operation we had two main concerns which were the user becoming harmed via accessing parts of the system or the user becoming harmed due to water ingress into the system's electrical components. In order to preemptively combat this we looked to the *IEC 60529* which outlines protection for a plethora of sections. We focused on two major components of *IEC 60529* the degrees of protection against water and against access of hazardous parts. For the first we reviewed the guidelines and decided that we would provide a minimum protection level of two, which states that the system will be protected against vertically falling water drops, such as rain, when the enclosure is tilted up to 15°and provide a maximum level of protection of 4 which grants safety from splashing of water [4].

The next class of protection we offered again comes from *IEC 60529* and it regards hazardous part access by foreign objects which could range from screwdrivers to human appendages such as fingers. Due to the nature of accessibility in this project we had to take a little more lax selection in our degree of protection as we need the user to able to potentially handle the system's internal section we have to allow at least access for the user's hands. Because of this we decided to provide level two protection which stipulates protection of hazardous parts from a finger or a solid foreign object of $\leq$12.5 mm in diameter [4]. This degree of protection provides a good compromise as it assures that the delicate components of the system will not be harmed or cause hard to a user while still granting the user access to the system in order to preform maintenance or potentially change out components.

Another standard we have discussed prior but have no elaborated on is Bluetooth SPP which is a crucial point in the systems execution as it is the sole performer of data transfer. This protocol exists as a replacement to the existing RS-232 profile with the goal of emulating the connection of the serial cable while providing the same control signals. Due to it having a maximum payload capacity of only 128 bytes this standard's main usage is to relay small stream bursts of information between two devices making it perfect for our purposes in this system [5].

VI. Constraints and Testing

Perhaps more important than the systems design is the thoroughness of its execution which is outlined by our testing plan in the following section.

During the course of the design and construction of the system our established requirements set many constraints both those set internally by the group others more market and unforeseen in nature. One major constraint we had to keep in mind that wasn't originally considered was the budgetary limitations of the project since this project was completely self-funded. Upon inspection the project does not appear to warrant a high cost and the estimated final bill of materials for production is only around $170 USD there were a lot of unforeseen developmental cost involved such as multiple revisions of the PCB being sent out to fix working issues, components being dead on arrival or dying during testing phases, and module incompatibilities for certain aspects of the system mainly the SPP's ability to communicate via a COMPORT to a user's file-system. However these various revisions and corrections to our system offered us ample opportunity to test functionality both individually and as a completed system. Regarding our testing methodologies we took the most thorough approach possible for the components and subsystems as possible. All initial testing was conducted on a development board for the ATMega 2560 produced by Arduino. Starting with the individual testing of each sensor and it's ability to write back to the integrated serial terminal provided by the Arduino IDE once individual functionality was confirmed next all three were integrated together and ran simultaneously displaying the necessity to instantiate the DHT object in a different scope to make sure it didn't block executing of the other sensors initialization. After this the RTC and Bluetooth were tested separately from the sensors and each other with the RTC being tasked to print to terminal based on a time interrupt and the Bluetooth to send a hexadecimal character to serial port upon command. Once both were confirmed working we then tested the Bluetooth to send a value after an RTC trigger, and then ultimately bundled these two together with the previously tested sensors to make sure that the sensors would read on an RTC interrupt and then could have their readings sent over through serial ports via Bluetooth. In between these stages testing of the LCD was conducted by changing the data from being displayed on a serial terminal to being displayed on the LCD. Once the first PCB revision was completed we began taking the same methodical approach of unit testing each component and system on the PCB to ensure that we covered any point of potential conflict or failure taking note of each undesired behavior and how it was rectified by comments in the code which was controlled using the centralized version control platform Git. With regard to the hardware testing we first tested the power circuit by creating a simulation to make sure the circuit could theoretically handle the loads provided then a vector board with the circuit was built and hooked up to an oscilloscope to monitor the levels and wave-forms outputted in the physical circuit in order to verify functionality. The PCB was tested by making sure each individual ground and power pin provided the expected output and all the communication pin such as UART, analog, or digital were tested for functionality by changing the pin assertion in the code to match the range of pins present on the board. After this the supporting software was test with both mock files used to simulate the format such as in Figure 5 previously and then with actual files generate by the board and transferred through the COMPORT. With the aforementioned testing we found the system sufficiently meeting our expectations.

VII. Acknowledgements

The design team would like to formally and sincerely thank the following people or groups for help contributed during the projects research, design, and construction. Dr. Lei Wei and Dr. Samuel Richie for being the leaders of the senior design class and providing the guidelines and advice during the general course of both sections of class. UCF professors, Dr. Zakhia Abichar for consultation on embedded debugging and oscillator design and Dr. Mark Llewellyn for consultation on python debugging. Additionally we would like to thank IEEE UCF branch particularly, Vice Chair, Gustavo Camero for consultation and guidance on the PCB design and implementation. Special thank you to the team at Quality Manufacturing Services, Inc. for their free-of-charge services in helping to mount tiny components on the PCB.

VIII. Design Engineers

The team of engineers the designed the Portable Plant Monitoring System were comprised of two duo teams one being electrical engineers the other being computer engineers who pooled their collective skill sets in order to come up with the idea, implantation, and execution of the system.

- Syed Bukhari
  Syed F Bukhari will be graduating from the University of Central Florida on May 2nd 2019 with a Bachelors of Science in Electrical Engineering. His primary interest are in Power, Transmission planning and Relay Protection. He is currently employed at Siemens Gas Power.
- Alexander Meade
  A computer engineer who originally wanted to work as a historian, his career path changed drastically after he received an apprenticeship in high school with Northrop Grumman. He began pursuing a degree in the field of engineering and quickly found a home in the EECS college due to his love of computers and electronics. While his planned career path is to eventually get into the field of computer networking or cyber-security he currently plans to work with Northrop Grumman after graduating as an embedded programmer while he seeks a higher education.
- Nicholas Poidomani
  Nicholas Poidamani is a B.S. in Computer Engineering who is currently a senior at University of Central Florida. He graduates May 2nd with the rest of the group and plans to pursue a career in embedded system design.
- Kareem Rahounji
  Kareem Rahounji is a B.S Electrical Engineering senior student who will be graduating in the semester of Spring 2019 on May 2nd. Field of study includes multiple disciplines of Electrical Engineering with a specialized focus on the different applications of Power Systems.

References

[1] Adafruit. Overview photocells. https://learn.adafruit.com/photocells/overview, July 2012. Accessed on 3-23-2019.

[2] Adafruit. Overview dht. https://learn.adafruit.com/dht/overview, November 2017. Accessed on 3-23-2019.

[3] Arduino. Gobetwino. https://playground.arduino.cc/Interfacing/GoBetwino/. Accessed on 3-23-2019.

[4] cvgstrategy. International iec 60529 ingress protection. https://cvgstrategy.com/iec-60529/, 2018. Accessed on 3-29-2019.

[5] SerialIO. Hat's the difference between bluetooths. https://www.serialio.com/faqs/whats-difference-between-bluetooth-le-and-bluetooth-spp-ble-vs-spp, 2018. Accessed on 3-29-2019.

[6] SparkFun. Sparkfun soil moisture sensor. https://www.sparkfun.com/products/13322. Accessed on 3-23-2019.