**032-x-22**

**Teaching Data Acquisition Through the Arduino Driven Home Weather Station Project**

Sheryl Dutton

Old Dominion University, Norfolk, VA

[sdutt005@odu.edu](mailto:sdutt005@odu.edu)

Kurt Galderisi

Old Dominion University, Norfolk, VA

[kgald001@odu.edu](mailto:kgald001@odu.edu)

Murat Kuzlu

Old Dominion University, Norfolk, VA

[mkuzlu@odu.edu](mailto:mkuzlu@odu.edu)

Otilia Popescu

Old Dominion University, Norfolk, VA

[opopescu@odu.edu](mailto:opopescu@odu.edu)

Vukica Jovanovic

Old Dominion University, Norfolk, VA

[v2jovano@odu.edu](mailto:v2jovano@odu.edu)

**Abstract**

The main objective of this paper is to present one possible way to engage undergraduate students in the design of a system that uses Internet of Things strategy for data acquisition and management. The MATLAB Home Weather Station project presented in this paper, was developed by a team of students for the senior design course in the Electrical Engineering Technology undergraduate program at Old Dominion University. The main purpose of this project was for the undergraduate students to learn how to create a localized, compact, and precise weather station. Utilizing various sensors, both homemade and sourced online, this weather station is capable of reading and displaying many useful weather details. Since a large variety of home weather stations are already available commercially, students start this project investigating the existing designs, comparing the commercially available solutions, and then they start developing their own solution, tailored to their own expectations for a weather station system.. The paper also includes the steps of the design process done by the team of undergraduate students who were focusing on the real-life aspects and were coming up with a new model of a home weather station. Following the steps of the project presented in this paper, one would gain the tools to create, calibrate, and customize their own Arduino Home Weather Station. This is an all-in-one self-sustaining station that collects outside weather details from the area where the user needs localized weather data and transmits that data wirelessly over Bluetooth to the point where the user is located, so they can collect and visualize the data from the comfort of their own home. The data is collected outdoors using various sensors connected to an Arduino UNO. Using a Bluetooth transmitter, the Arduino UNO wirelessly communicates sensor data to a receiver connected to the USB slot of an indoor computer. The computer runs a MATLAB program that takes the data received through the USB and graphically displays plots of the temperature, humidity, air pressure, precipitation, wind speed, wind direction, and cloud cover. With such a project, various teaching objectives can be accomplished, and students can learn about a broad variety of topics including sensors, microcontrollers, communication system, as well as data acquisition and processing.

**Introduction**

There are various commercially available solutions for home weather stations that can be easily integrated through the Internet of Things (IoT) technologies to home automation systems. They usually integrate humidity sensor, temperature sensor and barometric pressure sensor (Bäfverfeldt, 2016, Mao, 2018, Molnar, 2020). These integrated solutions usually focus on measuring temperature, relative humidity, and air pressure (Bäfverfeldt, 2016) and once coupled with some form of data logging and data streaming, they provide a great resource for students to build their skills related to the understanding of cloud-based instrumentation. Students can also build skills related to coding and creating the relationships among measured data and given streaming outputs (Caccamo, 2016). Involvement in such projects is especially beneficial for undergraduate students who can learn more about IoT technology, which can be better understood through a hands-on project that the students can easily relate to (Molnar, 2020). Being engaged in the process of designing a physical mockup of the system that can be used for the purpose of better understanding of the environmental parameters in their surroundings in a cost effective and flexible way can be done with the home weather project (Majumdar, 2018). Engaging more students in IoT related hands-on projects we are also trying to address the shortage problem of workforce capable of working on IoT projects that are highly needed in the industry. The project showcased in this paper is the work done by a team of undergraduate students for their senior design project course in the Electrical Engineering Technology program at ODU.

The curriculum of the Electrical Engineering Technology (EET) program at Old Dominion University culminates with a senior design project that students must complete for graduation. This project is completed through a sequence of two courses, a 1 credit hour course, during the first senior semester, in which students select a topic and make a project proposal, and a 3 credit hours course, in the second senior semester, in which the project is completed. This second course is also a writing intensive course, and it requires the students to submit a written report, in which they have to demonstrate their ability to write technical reports and incorporate technical references. The project ends with a presentation of the project with live demo. The grade for the senior project course is calculated based on a rubric that includes the technical implementation, the writing part, and the presentation. This paper is mainly developed on the senior project report prepared by a student team in fall 2021.

A team of two students built this MATLAB Home Weather Station project with the main goal of learning how to create a localized, compact, and precise weather station to allow individuals to appropriately prepare for the specific weather conditions in their area. When watching the weather on the news, the information is given for a wide geographic area, usually the overall region. This information is not always pertinent to an exact location. If an individual keeps outside animals or equipment that require specific temperature and humidity and the generalized weather report from the news calls for conditions that should be favorable, but in that individual’s exact location they are not, the individual could potentially lose valuable livestock or equipment. TheArduino Home Weather Station project presented in this paper would provide accurate data such that the users can better understand specific local weather conditions (Bäfverfeldt, 2016)

The MATLAB Home Weather Station is an all-in-one self-sustaining station that collects outside weather details where the individual needs localized weather data and transmits that data wirelessly over Bluetooth. This way, the individual can collect and visualize the data from the comfort of their own home. The data is collected outdoors using various sensors connected to an Arduino UNO, , an open-source microcontroller boarded based on Microchip ATmega328P microcontroller and developed by Arduino.cc and initially released in 2010 (Arduino, 2022, Blum, 2013). Using a Bluetooth transmitter, the Arduino UNO wirelessly communicates sensor data to a receiver connected to the USB slot of an indoor computer. The computer runs a MATLAB program that takes the data received through the USB and graphically displays plots of the temperature, humidity, air pressure, precipitation, wind speed, wind direction, and cloud cover.

The temperature data will be presented in both degrees Fahrenheit and Celsius on individual line graphs. Humidity data will be presented on a line graph in terms of relative humidity as a percentage. Air pressure data will be presented on a line graph in Pascals (Pa). Precipitation data will be presented on a line graph in millimeters of water covering the water level sensor. Wind speed data will be presented on a line graph in miles per hour (mph). Wind direction data will be presented on a circular plot with the cardinal directions. Cloud cover data will be presented on a line graph as a percentage, with 0% equated as dark/night and 100% meaning bright/clear skies.

**Main Considerations**

The design of the MATLAB Home Weather Station includes the various sensors, an Arduino UNO, a Bluetooth dongle, a home-made anemometer, a home-made wind vane, a solar powered power source, and the station itself which includes PVC pipe and a base of a plastic bin for containment and stability.

The temperature, humidity, air pressure, and precipitation sensors were purchased online and were wired to the Arduino UNO. The wind speed sensor was built by the students using an anemometer with a stationary Reed switch and a small magnet mounted to a spinning shaft, where it will “close” the sensor once every revolution. With this, the number of revolutions can be determined during a set amount of time and in turn calculate wind speed. The wind direction sensor will utilize a wind vane connected to a continuous single-turn potentiometer connected to a 5 Volt source. Using Analog to Digital Conversion, the value read from the potentiometer can then be converted to degrees of revolution to where the wind direction can be derived. The cloud cover sensor uses a photoresistor that decreases in resistance as bright light is exposed. Using an ADC conversion, the voltage read on this resistor can be attributed to a 0-1023 digital value. This can then be mapped in Arduino to scale between 0-100 to display a brightness as a percentage. The following diagram shows the different parts of the station, each of the four arms, what sensors they include, as well as schematics of how each sensor connects to the Arduino UNO.

Diagram

Description automatically generated

Figure 1. MATLAB Home Weather Station physical setup.

## **Wireless Communication via Bluetooth Construction Guide**

This section of the design involves the USB Bluetooth Dongle 4.0 and the HC-05 Wireless Bluetooth RF Transceiver. The goal of this section is to get the PC to communicate with the Arduino UNO via the HC-05 wirelessly over Bluetooth. First, the Bluetooth Dongle 4.0 is inserted into a USB slot on a PC with Windows 10 installed. Once the program automatically installs all necessary drivers, all available Bluetooth devices should show up. Next, the Bluetooth device “HC-05” is chosen and the password “1234” is used to connect. The Bluetooth Dongle assigns itself to a serial COM port where data transmitted from the HC-05 is found. To connect the HC-05 to the Arduino UNO, the Vcc pin connects to 5V, the ground pin connects to the ground pin, and the Rx data pin on the HC-05 connects to pin D1 on the Arduino UNO. Note that the Tx pin on the HC-05 is not used as no data is being sent to the Arduino UNO. The HC-05 connects to the Arduino UNO as follows:

Diagram

Description automatically generated

Figure 2. HC-05 and Arduino UNO Diagram.

## **Outdoor Temperature including Wind Chill Guide:**

This section of the design involves the DS18B20 Temperature Sensor and the Arduino UNO. The goal of this section is to get the DS18B20 temperature sensor connected to the Arduino UNO and to test the sensor for accuracy. First, the sensor is connected to the Arduino UNO by connecting a yellow data wire to digital pin 13, a red wire to 5V, and a black wire to ground. Next, a simple test code (see Appendix D) is input into the Arduino program to display data read from the sensor. The serial monitor should display temperature in degrees Fahrenheit and degrees Celsius. Accuracy is tested by comparing the data to a known good thermometer. The DS18B20 connects to the Arduino UNO as follows:

Graphical user interface

Description automatically generated with medium confidence

Figure 3. DS18B20 Temp and Arduino UNO Diagram.

## **Outdoor Temperature and Humidity Guide**

This section of the design involves the DHT22 Digital Temperature and Humidity Sensor and the Arduino UNO. The goal of this section is to get the DHT22 temperature and humidity sensor connected to the Arduino UNO and to test the sensor for accuracy.  First, the sensor is connected to the Arduino UNO by connecting the ground pin of the DHT22 to ground, the VCC to 5V, and the middle “signal” pin to digital pin 12 on the UNO. Next, a simple test code (see Appendix C) is input into the Arduino program to display data read from the sensor. Accuracy is tested by comparing the results with a known good thermometer and hygrometer. The DHT22 connects to the Arduino UNO as follows:

Diagram

Description automatically generated with low confidence

Figure 4. DHT22 Temp & Humidity and Arduino UNO Diagram.

## **Precipitation Sensor Guide**

This section of the design involves the Water Depth Sensor and the Arduino UNO. The goal of this section is to get the water depth sensor connected to the Arduino UNO and to test the sensor for accuracy. The water depth sensor functions by treating water as a variable resistor; the higher the water level, the more energy flows and the sensor outputs a higher voltage to then be converted using ADC. By performing the ADC conversion on the values viewed on the serial monitor from pin A3, the water level can be determined and graphed. This is done by first noting the initial values viewed from the serial monitor; these initial values will be between 0-1023, increasing with depth. To find the ADC values to use for each increment of 5mm, the ADC values at each increment of 5mm is read and then the converted values on the serial monitor are displayed as the sensor is placed deeper in the water. The ADC values at these increments will be used to make an else if loop in the Arduino programming to display depth in mm. In construction, first the sensor is marked in increments of 5mm with a waterproof marker.

Next, a simple test code is input into the Arduino program to display data read from the sensor.

A picture containing text, electronics, screenshot

Description automatically generated

Figure 5. Rain Sensor and Arduino UNO Diagram.

Next, the sensor is connected to the Arduino UNO by connecting the positive terminal to 5V, the negative terminal to ground, and the “S” terminal to analog pin A3 on the UNO.

//Note this code works for finding average ADC values and need to be recorded to create depth ranges

int DepthValue; //create integer storage for ADC values

void setup()

{

//Begin serial communication

Serial.begin(9600);

}

void loop()

{

DepthValue = analogRead(A3); //read data from analog pin and store it to value variable

Serial.println(DepthValue); //display the ADC value

delay(4000); //long delay to observe values at depth increments

getDepth(); //create this once average ADC is observed and subroutine written

}

//subroutine created to display depth in millimeters from average ADC values at that depth

void getDepth()

{

if (DepthValue<=300){ //always empty below this ADC value

Serial.print("00");

}

else if (DepthValue>300 && DepthValue<=350){

Serial.print("05");

}

else if (DepthValue>351 && DepthValue<=400){

Serial.print("10");

}

else if (DepthValue>401 && DepthValue<=450){

Serial.print("15");

}

else if (DepthValue>451 && DepthValue<=500){

Serial.print("20");

}

else if (DepthValue>501 && DepthValue<=550){

Serial.print("25");

}

else if (DepthValue>551 && DepthValue<=600){

Serial.print("30");

}

else if (DepthValue>601 && DepthValue<=650){

Serial.print("35");

}

else if (DepthValue>651){ //always full at this ADC value

Serial.print("40");

## **Air Pressure Sensor Guide**

This section of the design involves the BMP180 GY-68 Digital Barometric Pressure Sensor and the Arduino UNO. The goal of this section is to get the BPM180 barometric pressure sensor connected to the Arduino UNO and to test the sensor for accuracy. First, the sensor is connected to the Arduino UNO by connecting Vin on the BMP180 to 3.3V on the UNO, Gnd to ground, SCL to A5, and SDA to A4. This sensor needs to be connected at 3.3V instead of the typical 5V due to the sensor overloading at 5V. Next, a simple test code (see Appendix A) is input into the Arduino program to display data read from the sensor. Accuracy is tested by comparing the results with a known good barometer. The BMP180 connects to the Arduino UNO as follows:

A picture containing timeline

Description automatically generated

Figure 6. BMP180 Air Pressure and Arduino UNO Diagram.

## **Cloud Cover Sensor Guide**

This section of the design involves the Photoresistor Photo Light Sensitive Resistor and the Arduino UNO. The goal of this section is to get the photoresistor connected to the Arduino UNO. First, the sensor is connected to the Arduino UNO by connecting the photoresistor with one lead connected to ground and the other lead connected to a 10k-ohm resistor, which is then connected to 5V. At the 10k-ohm and photoresistor junction, a lead connects to pin A3 of the Arduino UNO. A simple test code (See Appendix B) is used to verify and test the photoresistor photo light sensitive resistor. The photoresistor is connected to the Arduino UNO as follows:

A picture containing text, electronics, screenshot

Description automatically generated

Figure 7. Photoresistor and Arduino UNO Diagram.

## **Wind Direction Potentiometer Guide:**

This section of the design involves the 20k-ohm Potentiometer and the Arduino UNO. The goal of this section is to get the potentiometer connected to the Arduino UNO. This sensor works by performing an analog read on pin A0 of the UNO to generate a value between 0-1023 based on the position of a single turn of the potentiometer. This value can then be “mapped” to correlate to a direction value of 0 to 360 degrees, corresponding to the eight cardinal directions (N, NE, NW, S, SW, SE, E, W) when processed in an else if subroutine in the Arduino programming. To calibrate, use a known good compass to determine true North. The test program (see Appendix F) can then be run to find where the ADC conversion results in the values 1023, 0, or 1. Mark the potentiometer where these values read as these are the North values and the assembled device needs to be placed in this direction when installed.

Diagram

Description automatically generated

Figure 8. Potentiometer and Arduino UNO Diagram.

In construction, first the sensor is connected to the Arduino UNO by connecting a red lead to the CW pin on the potentiometer and connecting that to 5V. Next, a black lead is connected to the CCW pin and then connected to ground. Then, a yellow lead is connected to the center pin and that is connected to analog pin A0 on the Arduino UNO. The potentiometer is connected to the Arduino UNO as follows:

## 

## **Wind Vane Construction Guide**

Once the potentiometer is connected to the Arduino UNO, the rest of the wind vane needs to be constructed. This section of construction involves a PVC end cap, a ½-inch 90-degree L-shaped PVC connector, a ⅜-inch wooden dowel, a metal washer, wind vane tails cut out from a Sterlite bin lid, an aluminum spacer, and the potentiometer. First, three long 22 AWG wires are measured and cut to reach the Arduino UNO at the base of the station. These wires are soldered to the potentiometer contacts to ensure a stable connection. Next, a one-foot length of the ½-inch PVC pipe is connected to the PVC L-shaped connector, which will then be connected to the main 1-½-inch PVC vertical of the station. The potentiometer wires are fed through the elbow and the ½-inch pipe and then down the center pipe.  The vane consists of an aluminum spacer with an inner diameter that fits snugly to the turnscrew of the potentiometer. A 3/16-inch hole is drilled through the top of the spacer, which allows a one-foot length 3/16-inch diameter wooden dowel to be pressed through. Two wind vane tails are cut from a Sterlite bin lid and hot glued on either side of one end of the wooden dowel. On the other end of the dowel, a nose weight is made from a PVC end cap and metal washer glued together. Because the wind vane works most efficiently if properly balanced, the MATLAB Home Weather Station should be installed on a flat level surface. If the vane tilts to one side, the dowel can be pushed more to one side so that it stands up without tilting. The wind vane will only measure properly if first aligned to true North; to do this, make sure the station is installed with the marked line of the potentiometer facing true North. The following photo shows the constructed wind vane.

A close-up of a syringe

Description automatically generated with low confidence

Figure 9. Photo of Completed Wind Vane.

## **Wind Speed Guide:**

This section of construction involves a Reed Switch, Magnet, two 47-ohm resistors and the Arduino UNO. The goal of this section is to get the Reed switch, magnet, and 47-ohm resistors connected to the Arduino UNO. This sensor works by creating a closed loop via the Reed switch and the magnet each time the anemometer cups make a revolution about the wooden dowel. Each time the magnet passes the normally open Reed switch, the switch closes, completing the circuit and allowing the Arduino UNO to see one revolution. In order to calculate the wind speed from this information, calculations on the spinning radius, circumference, and revolutions per hour are needed.

The assembled anemometer has a spinning radius of 4 inches. This gives a spinning circumference of 2.093 feet.

(eq. 1)

When exposed to a 1 mph wind, the anemometer should turn 2,522 revolutions per hour. This is calculated by dividing the number of feet in a mile (5,280) by the circumference in feet (2.093). To reduce this value into seconds per revolution, divide the number of seconds in an hour (3,600) by the revolutions per hour (2,522). This results in a seconds per revolution value of 1.427.

(eq. 2)

(eq. 3)

These values will be used in the programming to correctly calculate wind speed.

When creating the code, the equation

(eq. 4)

is used, where V = velocity in mph, P = pulses sensed by the Arduino interrupt, and T = time period of the sampling. This equation was sourced from the spec sheet of a popular anemometer from Davis Instruments (see Appendix M).

Because this value is only truly accurate in a frictionless environment, a calibration is needed to adjust the seconds per revolution value to that of a known good store-bought anemometer. This is done by comparing the wind speed of a constant source, such as that from a free-standing home fan to both the store-bought anemometer and the home-made anemometer. The difference in wind speed measurement from the home-made anemometer will be plugged into the Arduino velocity equation to compute a more accurate wind speed measurement. The test fan output is about 5.5mph on the store-bought anemometer. By using the guess-and-check method of altering the seconds per revolution value in the Arduino program and comparing the mph values read, the new seconds per revolution value that most closely matched the 5.5mph of the store-bought anemometer was 2.5. Maximum speed this wind vane can withstand was not determined. Therefore, the final equation plugged into the Arduino programming is:

In construction, first the magnet is placed so that it closes only at one point of the complete anemometer rotation. Then, two 47-ohm resistors are placed on both sides of the Reed switch to prevent overload. One lead of the switch goes to ground, and the other lead connects to digital pin 11 of the Arduino UNO with a 4700-ohm pull up resistor also connected to pin 11 where the switch closes. Next, a simple test code (see Appendix G) is input into the Arduino program to display data read from the sensor. The anemometer is connected to the Arduino UNO as follows:

Diagram

Description automatically generated with medium confidence

Figure 10. Anemometer and Arduino UNO Diagram.

## 

## **Wind Speed Anemometer Construction Guide**

Once the Reed switch is connected to the Arduino UNO, the rest of the Anemometer needs to be constructed. This section of construction involves two PVC end caps, a ⅜-inch wooden dowel, a PVC SxSxS Tee, and four Betty Crocker measuring spoons. First, two long 22 AWG wires are measured and cut that will connect the Reed switch to the Arduino UNO at the base of the station. The wires are soldered to the ends of the Reed switch and pulled through a one-foot length of a ½-inch PVC pipe which connects to the main 1-½-inch vertical PVC pipe. The Reed switch is superglued to the PVC pipe so that it just barely sticks out the end of the pipe.  Next, a ball bearing is pressed inside one of the PVC end caps so that it fits snugly and level. In order to properly connect the two pieces together, the PVC cap with the bearing is then connected to the PVC SxSxS-Tee connector with a two-inch length of the ½-inch PVC. The other PVC end cap has four 3/16-inch holes drilled perpendicular from each other and a 3/16-inch diameter wooden dowel is cut into four 3-inch segments which are pushed through the holes of the PVC end cap. Four plastic Betty Crocker measuring spoons, in the tablespoon size, are cut and superglued to the dowels sticking out of the PVC end cap. The cups total length from the center of the end cap will be four inches which is needed in calculating wind speed.  Next, a 5-inch long ⅜-inch diameter wooden dowel is pressed into the inner diameter of the bearing of the PVC SxSxS Tee joint with ball bearing cap. Once the exposed top of the dowel to the PVC end cap with cups connected is made watertight, it is important to get the dowel and cup cap to seat as level as possible or it will not spin properly.  Next, a magnet is hot glued to the dowel shaft that goes through the middle of the SxSxS Tee connector. This way, the Reed switch glued to the inside of the SxSxS Tee connector will close every time the magnet connected to the dowel makes a full revolution. A diagram of the internals of the anemometer (the PVC spacer between the bottom end cap and the SxSxS Tee connector is not shown for internal clarity). Shown below is a photo of the completed anemometer. This project includes initial design of the weather station, limitations of each instrument build were not discussed in this study.

A close-up of a syringe with a red cap

Description automatically generated with low confidence

Figure 11. Photo of Completed Anemometer.

## **Main Housing Guide**

This section of construction involves the PVC pipes, a 12-qt plastic bin, super worm cups, the power supply, wiring, and the Arduino UNO. The goal of this section is to get all the sensors and power supply connected to the Arduino UNO inside the PVC pipes and plastic bin. The main housing is designed to be placed on the ground and maneuverable so that the wind vane can be aligned North, and tall enough so that wind can be caught in the anemometer cups and wind vane tail. A 12-qt plastic container is used as the base of the station, where the Arduino UNO and power supply are stored.

First, the 12-qt bin is turned upside down and a 1-⅞ inch hole is drilled in the center of the bottom of the container. Then, a four-foot-tall 1-½ inch PVC pipe is fit snugly into the hole and secured with hot glue. The PVC pipe acts as the center conduit for the sensor wires and the four arms of the MATLAB Home Weather Station. Next, four ⅞-inch holes are drilled into the 1-½-inch PVC; the first two drilled six inches from the top of the pipe and the other two holes twelve inches below those holes. These holes will be for the four, one-foot-long lengths of ½-inch PVC pipe that will act as the sensor arms. The top two sensor arms contain the home-made wind sensors, and the lower two arms contain the manufactured sensors. Because the lower two arms have sensors that need protection from the elements, modified super worm cups are placed to fit over the end of the sensor arms to provide protection. The top of the main PVC pipe contains the cloud sensor and is sealed over with clear plastic to keep the station watertight.

For the power source, a store-bought solar panel with attached rechargeable battery was purchased. The rechargeable battery is placed inside the plastic bin and the solar panel is placed outside. The bin is cut to allow the cord to pass through and sealed with hot glue to remain watertight. Because the rechargeable battery outputs a voltage of 12V and the Arduino UNO can only take 7-12V without overload, a Buck converter is installed in between the battery and UNO for safety. This Buck converted takes the battery output voltage of 12V and converts it into an input voltage to the UNO at 9V. Whereas the Arduino UNO can receive up to 12V, to make the MATLAB Home Weather Station more reliable, the Buck converter is implemented as a safety net.

Graphical user interface, application

Description automatically generated

Figure 12. Data acquisition dashboard in MATLAB.

The cloud cover and wind direction data will be represented separately in MATLAB. MATLAB is a programming platform, which is used to analyze and design systems by for engineers and scientists. It offers many features, such as plotting of functions and data, algorithms development, customize user interfaces, as well as interacting with other systems.

With the goal of creating a localized, compact, and precise weather station to allow individuals to prepare for their specific atmospheric conditions, the MATLAB Home Weather Station delivers. The MATLAB Home Weather Station could measure temperature in both degrees Celsius and degrees Fahrenheit, humidity, barometric pressure, wind speed, wind direction, cloud cover and even precipitation. On top of having all this sensing capability, the MATLAB Home Weather Station transmits the sensing data completely wirelessly over Bluetooth, all while the station itself is self-sustaining via a solar powered battery device. Using the powerful program MATLAB, the MATLAB Home Weather Station is able to take the data from the sensors and display it in easy-to-read graphs. This way, the user of a MATLAB Home Weather Station can keep track of their outside conditions from the comfort and safety of their own homes while the station withstands the elements with its rugged and watertight construction. The graphed results to show the range of the MATLAB Home Weather Station are included at the end of this report in Appendix L. It should be noted that these results were gathered in a test environment and do not necessarily represent realistic weather conditions. During the design and construction of this project, not all went smoothly and without issue. As this report outlines, several problems were encountered and had to be overcome, sometimes creatively. Some problems were small, such as poorly designed Reed switches purchased from Amazon, where even with the utmost care and precision of trying to bend the leads, the glass encased switch shattered, and some problems were larger, where the anemometer by design would not be completely accurate and a store-bought anemometer needed to be purchased to compare the two in order to calibrate the home-made anemometer correctly. At the completion of a project of this complexity, it is easy to look back and contemplate what could have been done differently along the way. More experience in cutting and manipulating PVC as well as more precise tools would have been helpful in getting the station to look more polished, along with using different colored PVC for each arm to make it easier to know where each sensor was placed. In conclusion, the MATLAB Home Weather Station project completed all the goals originally set and more.

# **Conclusion**

The project presented in this paper was completed by a group of two undergraduate students as part of their senior project requirement for the BS degree in Electrical Engineering Technology. To complete the project students had to learn how to properly set up and wire each sensor to the Arduino UNO unit, to solder wires where necessary, to program the Arduino processor and to write code in MATLAB. The project drew on knowledge learned through the courses of the Electrical Engineering Technology program and was also an opportunity to research and learn new knowledge and techniques. The project merged the hands-on work specific to electrical engineering with hardware design and construction, since the students had to work with PVC, hot glue, circular saws, and more, learning along the way. It is expected that the level of details and the step-by-step approach presented in this paper to help anyone who wishes to create and use their own MATLAB Home Weather Station.

# 

**References**

Arduino (2022). UNO R3, Arduino UNO R3, *Arduino*, Retrieved on October 1, 2022, from https://docs.arduino.cc/hardware/uno-rev3

Bäfverfeldt, F. (2016). Building an Arduino based weather station and connecting it as a slave to a control system.

Blum, J. (2019). Exploring Arduino: tools and techniques for engineering wizardry. John Wiley & Sons.

Caccamo, M. T., Calabrò, E., Cannuli, A., & Magazù, S. (2016). Wavelet study of meteorological data collected by Arduino-weather station: Impact on solar energy collection technology. *In MATEC Web of Conferences (Vol. 55, p. 02004)*. EDP Sciences.

Fritzing (2022). Fritzing source code, Retrieved on August 10, 2022, from: https://fritzing.org/

Hughes, J. M. (2016). *Arduino: A Technical Reference: A Handbook for Technicians, Engineers, and Makers (In a Nutshell) (1st ed.)*. Beijing, China: O'Reilly Media.

Khatri, P. (2018, October 31). *How to Plot Real Time Temperature Graph using MATLAB*. Retrieved November 09, 2020, from https://circuitdigest.com/microcontroller-projects/how-to-plot-real-time-temperature-graph-using-matlab

López, D. B., & Alfredo, B. V. (2019). *MATLAB Handbook with Applications to Mathematics*, Science, Engineering, and Finance (1st ed.). Boca Raton, FL: CRC Press Taylor & Francis.

Majumdar, I., Banerjee, B., Preeth, M. T., & Hota, M. K. (2018, July). Design of weather monitoring system and smart home automation. *In 2018 IEEE International Conference On System, Computation, Automation And Networking (icscan)* (pp. 1-5). IEEE.

Mao, H., Paul, O. K., Yang, N., & Li, L. (2018). Smart Arduino Sensor Integrated Drone for Weather Indices: Prototype. Drones-Applications; IntechOpen: London, UK, 31-41.

Molnár, J., Kiresova, S., Vince, T., Kovac, D., Jacko, P., Beres, M., & Hrabovský, P. (2020). Weather Station IoT Educational Model Using Cloud Services, *Journal of Universal Computer Science, 26(11)*, 1495-1512.

Moore, H., & Sanadhya, S. (2009). *MATLAB for Engineers* (p. 672). New York: Pearson Education International.

**Biographies**

**SHERYL DUTTON** was an undergraduate student during this project working on the senior design project and received his BS degree in Engineering Technology, majoring in Electrical Engineering Technology, from Old Dominion University, Norfolk, VA. They can be reached at [sdutt005@odu.edu](mailto:sdutt005@odu.edu)

**KURT GALDERISI** was an undergraduate student during this project working on the senior design project and received his BS degree in Engineering Technology, majoring in Electrical Engineering Technology, from Old Dominion University, Norfolk, VA. They can be reached at [kgald001@odu.edu](mailto:kgald001@odu.edu)

**MURAT KUZLU** is an assistant professor of engineering technology at Old Dominion University. He received his BSc, MSc, and PhD degrees in electronics and telecommunications engineering from Kocaeli University, Turkey, in 2001, 2004, and 2010, respectively. From 2005 to 2006, he worked as a Global Network Product Support Engineer at Nortel Networks, Turkey. In 2006, he joined the Energy Institute of TUBITAK-MAM (Scientific and Technological Research Council of Turkey – The Marmara Research Center), where he worked as a senior researcher. Before joining ODU, he worked as a research assistant professor at Virginia Tech’s Advanced Research Institute. His research interests include smart grid, demand response, smart metering systems (AMR, AMI, and AMM), home and building energy management systems, co-simulation, wireless communications, and embedded systems. Dr. Kuzlu may be reached at [mkuzlu@odu.edu](mailto:mkuzlu@odu.edu)

**OTILIA POPESCU** is an associate professor of engineering technology and the program director of the electrical engineering technology program at Old Dominion University. She received her Engineering Diploma and MS degree from the Polytechnic Institute of Bucharest, Romania, and PhD degree from Rutgers University, all in electrical and computer engineering. Her research interests include the general areas of communication systems, control theory, signal processing, and engineering education. In the past, she has worked for the University of Texas at Dallas, the University of Texas at San Antonio, Rutgers University, and the Politehnica University of Bucharest. Dr. Popescu may be reached at [opopescu@odu.edu](mailto:opopescu@odu.edu)

**VUKICA JOVANOVIC** is the Chair, Batten Endowed Fellow, and Associate Professor of Engineering Technology at Old Dominion University She holds a PhD from Purdue University, Magistar (PhD candidate) degree in industrial engineering and management, focused and dipl.ing. Degrees from the University of Novi Sad, Serbia. She has funded research in broadening participation efforts of underrepresented students in STEM funded by the U.S. Department of Education, focusing on computer science and cybersecurity pathways, and from the Office of Naval Research, focusing on mechatronic pathways. She is part of the ONR projects related to the additive manufacturing training of the active military. She is also part of the research team that has multiple projects funded by the NSF, focusing on veteran pathways and their success in engineering. She leads the team that delivers the summer program to ninth graders that focuses on broadening the participation of underrepresented students into STEM (ODU BLAST), funded by the Virginia Space Grant Consortium. Dr. Jovanovic may be reached at [v2jovano@odu.edu](mailto:v2jovano@odu.edu)