Adriann Liceralde and Jacob Larkin
Dr. Brian McPherson
Carbon Science Research at the University of Utah
Wireless Sensor Network Project
Last Updated March 18, 2022

Overview of the Sensor Node: Arduino Version

I. Introduction

Carbon sequestration is the process of storing CO_2 deep underground to reduce atmospheric CO_2 levels. This process is effective if the stored carbon does not leak into the atmosphere; therefore, monitoring systems are implemented over a sequestration field to detect signs of carbon leakage. The *Wireless Sensor Network Project (WSN)* aims to develop a low-cost system that accurately detects potential carbon leakage within an area.

The *Sensor Node*, or the *Node* for short, is a prototype instrument for locating carbon sources. The *WSN* project has two variants of this instrument: the *XBee Version* and the *Arduino Version*. This document discusses the *Arduino Version*, which consists of electronic hardware assembled onto a breadboard and controlled by an Arduino Microcontroller. When in operation, the *Node* reads CO_2 concentration, ambient temperature, wind speed, wind direction, and the UTC Unix timestamp. The information gets stored on a microSD card and can be manually retrieved.

The structure of this document is as follows: 1) in-depth discussion of the hardware, 2) procedure for assembly, and 3) explanation of the code.

II. Individual Components

The *Arduino Version* of the *Node* consists of the following major components: 1) Arduino Microcontroller, 2) CO_2 sensor, 3) wind sensor, 4) real-time clock, and 5) microSD card module. Additional parts include a switch, an LED, logic converters, wirings, resistors, and capacitors.

a. Arduino

The brain of the *Node* is the Arduino Microcontroller which contains the software to run the system. There are many different Arduino boards, but the most commonly used one is the Arduino Uno. A smaller version is the Arduino Nano, which has the same features and capabilities as the Uno. The Nano is used for the *Arduino Version* because its small size easily fits onto a breadboard.

The Arduino Nano continuously runs any software uploaded into it, as long as there is a power supply. If power is disconnected but reconnected, the software automatically runs again.

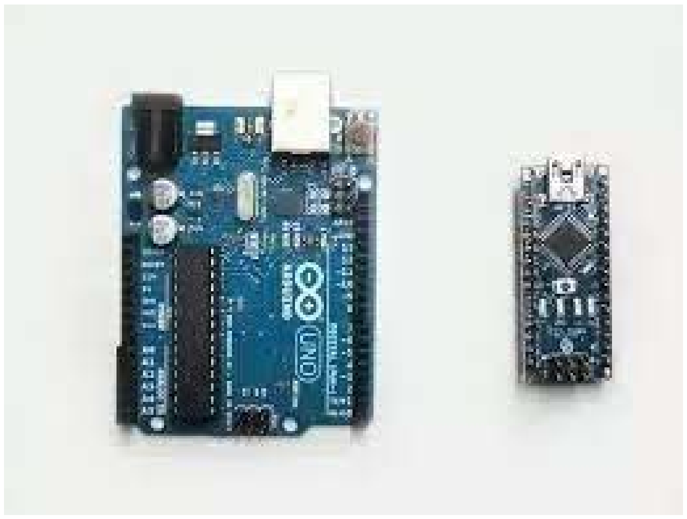
Figure 1. The Arduino Uno and the Arduino Nano[1].

b. CO_2 Sensor

One of the sensor units of the *Node* is a low-cost CO_2 sensor called the Sensirion SCD30. This module uses NDIR technology to passively measure the ambient CO_2 level without the need for pumps or fans. The fastest frequency the device can take readings is about 2.1 seconds. For consistent data timestamps, the collection frequency of the *Node* is set to 3 seconds. Other specifications are in Table 1.

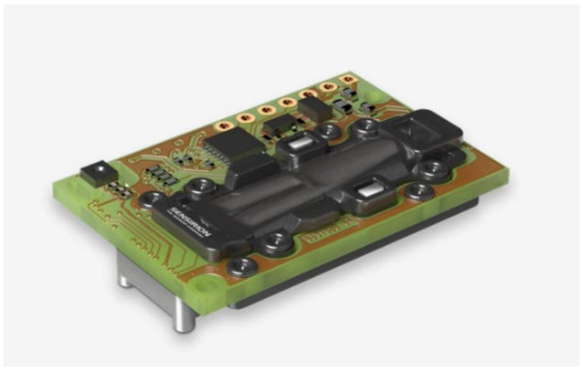
Figure 2. The Sensirion SCD30 CO_2 Sensor[2].

Table 1. Specifications of the SCD30.

Model	SCD30
Brand	Sensirion
Range of CO_2	0 – 40, 000 ppm
Range of Temperature	-40 – 70 °C
Accuracy of CO_2	± (30 ppm + 3%MV)
Voltage	3.3 – 5.5 V
Current	19 mA (avg); 75 mA (max)
Recent Price	$70

c. Wind Sensor

The second sensor unit of the *Node* is a low-cost anemometer called the Davis 6410. The anemometer consists of a wind vane and wind cups. The wind vane measures the direction, while the wind cups emits a pulse after a full rotation. Converting from the number of pulses to wind speed is performed with the following formula[3]:

$$V = P\left(\frac{2.25}{T}\right) / \, 2.237$$

Where:

V = speed (meters per second)
P = number of pulses per sample period
T = sample period (seconds)

The anemometer uses an RJ11 plug that has four wires. An RJ-11 port[4] is used to form a connection between the anemometer and the Arduino board.

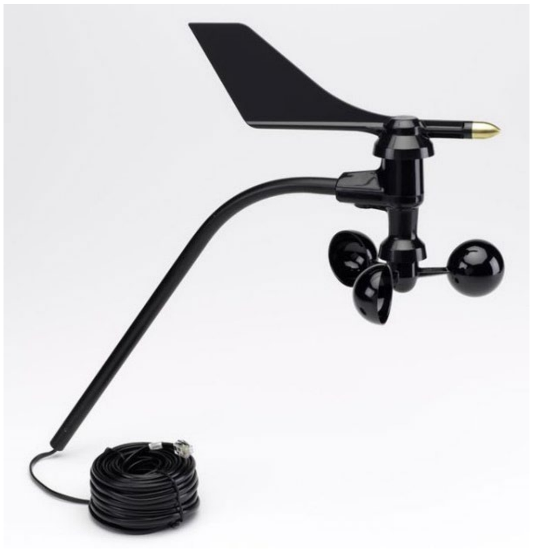

Figure 3. The Davis 6410 anemometer[5].

Table 2. Specifications of the Davis 6410.

Model	6410
Brand	Davis
Range of Direction	1 to 360 °
Range of Speed	0.5 to 89 m/s
Accuracy of Direction	± 3°
Accuracy of Speed	±1 m/s or ± 5%
Recent Price	$170

d. Real-Time Clock

A real-time clock (RTC) is an electronic module that keeps track of the current time; this is crucial for data-logging applications. For example, the *Sensor Node* must have accurate time to complement the collected data. A popular module is the ZS-042 board, which easily fits into a breadboard, and it contains two crucial components: the DS3231 chip and a CR2032 backup battery. The board also contains an EEPROM storage chip, but it is inactive in the *Node*.

The DS3231 chip is the actual RTC, which keeps track of time as long as there is power. Without power, the clock resets; this is where the backup battery serves its purpose. If the main power supply disconnects (e.g., the Arduino powers down), the clock still keeps the current time because of the backup battery. However, if both the main power supply and the backup battery disconnect, the clock resets and must be manually set to the current time.

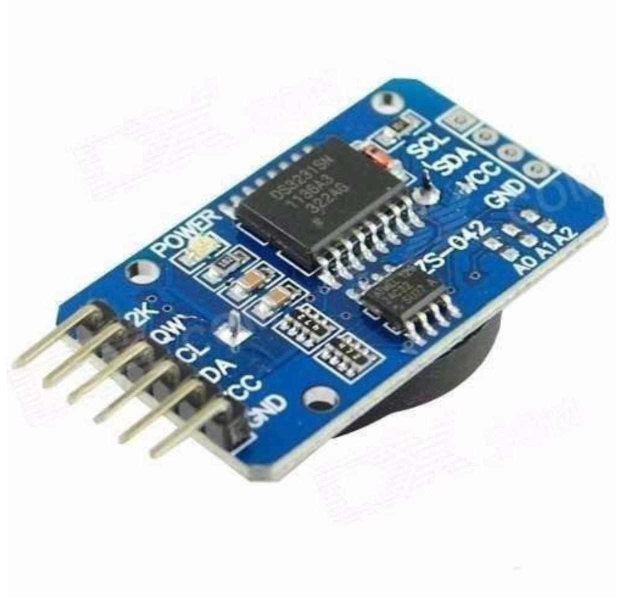

Figure 4. The ZS-042 real-time clock module[6].

Table 3. Specifications of the ZS-042 module.

Model	ZS-042
Brand	*Variety of brands*
Range of Operating Temp.	-40 – 85 °C
Accuracy	±3.5 ppm
Voltage	2.3 – 5.5 V
Current	300 µA
Recent Price	$5

A fatal design flaw of the ZS-042 is the battery charging circuit. The board recharges the backup battery when the main power supply is connected. If the battery is a rechargeable CR2032, this system is not problematic. However, if the battery is NOT rechargeable, this system is hazardous. Therefore, the charging circuit should be removed to prevent any hazards.

There are two ways to disable the circuit: cut the PCB trace or desolder the diode. Below is a close-up of the module. Circled in red are two lines of PCB tracing. Using a knife to cut through those lines disables the circuit. Above that red circle is a red-orange diode. Desoldering that diode guarantees that the risky charging circuit is offline.

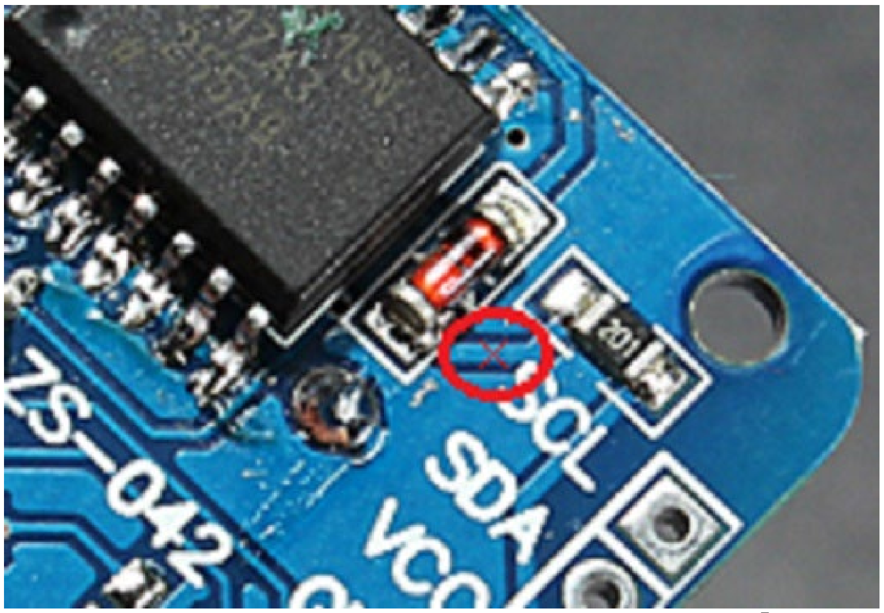

Figure 5. Close-up of the battery recharging circuit[7].

e. MicroSD Card Module

The MicroSD Card Module is an electronic device that stores data from a microcontroller into a microSD card. Long-term data loggers need a reliable way of storing collected data. Implementing this module onto the *Sensor Node* allows field deployments at long durations. The more storage space the card has, the longer the deployment can last before the card must be manually retrieved.

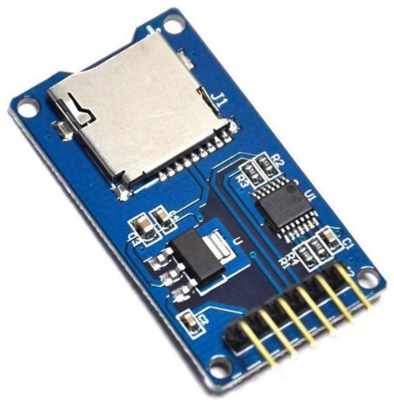

Figure 6. MicroSD Card Module[8].

Table 4. Specifications of the MicroSD Card Module.

Brand	*Variety of brands*
Voltage	4.5 to 5.5 V
Recent Price	$5

f. Logic Converters

The *Node* uses numerous communication protocols to perform its various functions, such as the I^2C and SPI methods. For example, the SCD30 CO_2 sensor and the ZS-042 RTC module communicate with the Arduino using the I^2C method. There are two pins dedicated to I^2C: the SDA pin and the SCL pin. These pins correspond to Arduino's analog pins: A4 and A5.

The microSD card module uses the SPI method, which uses the SCK, MISO, MOSI, and SS pins. These pins correspond to Arduino's digital pins: D13, D12, D11, and D10.

The datasheets recommend that the I^2C and SPI pins of the modules are at a voltage level of 3.3 V. This is problematic because the Arduino's I^2C and SPI pins are at a voltage level of 5.0 V and are unchangeable without an external device. Although the modules prefer 3.3 V for their communication pins, they can still operate when they are at 5.0 V. However, operating under this condition may increase the modules' degradation. Therefore, it is highly recommended that their communication pins are at 3.3 V.

Thus, the solution to this is the use of bi-directional logic converters. These converters shift the Arduino's 5.0 V level pins to the 3.3 V level. The CO_2 sensor and the microSD card module are still powered by 5.0 V, but their communication pins now operate at 3.3 V.

In short, the logic converters ensure a safer and more reliable operation of the CO_2 sensor and the microSD card module.

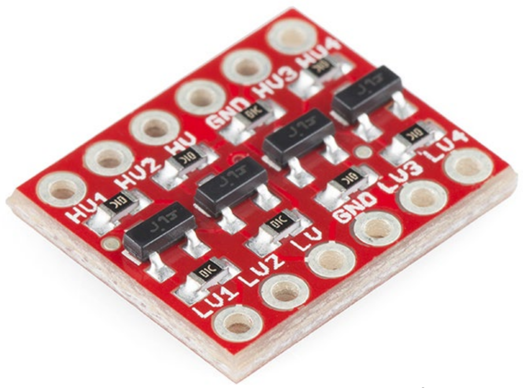

Figure 7. Logic Converter[9].

III. Assembly

Assembling the *Sensor Node* requires the components mentioned above, a solderless breadboard, many wires, and other additional parts. Figures 8 and 9 give a visual representation and a schematic of how the wires should be connected. It is recommended that when assembling the *Node*, excellent care and focus is taken to ensure that every wire and component is connected correctly. Improper or loose connections could result in serious malfunction. In addition, the Arduino board must NOT be connected to a power source while the assembly is taking place.

The modules and wires can be connected in any order. However, the following procedure is recommended for ease. If this procedure and the associated figures are followed, then the modules and wires must be connected EXACTLY like the one in the graphics. The graphics for each step are found in the external folder called "AssemblySteps."

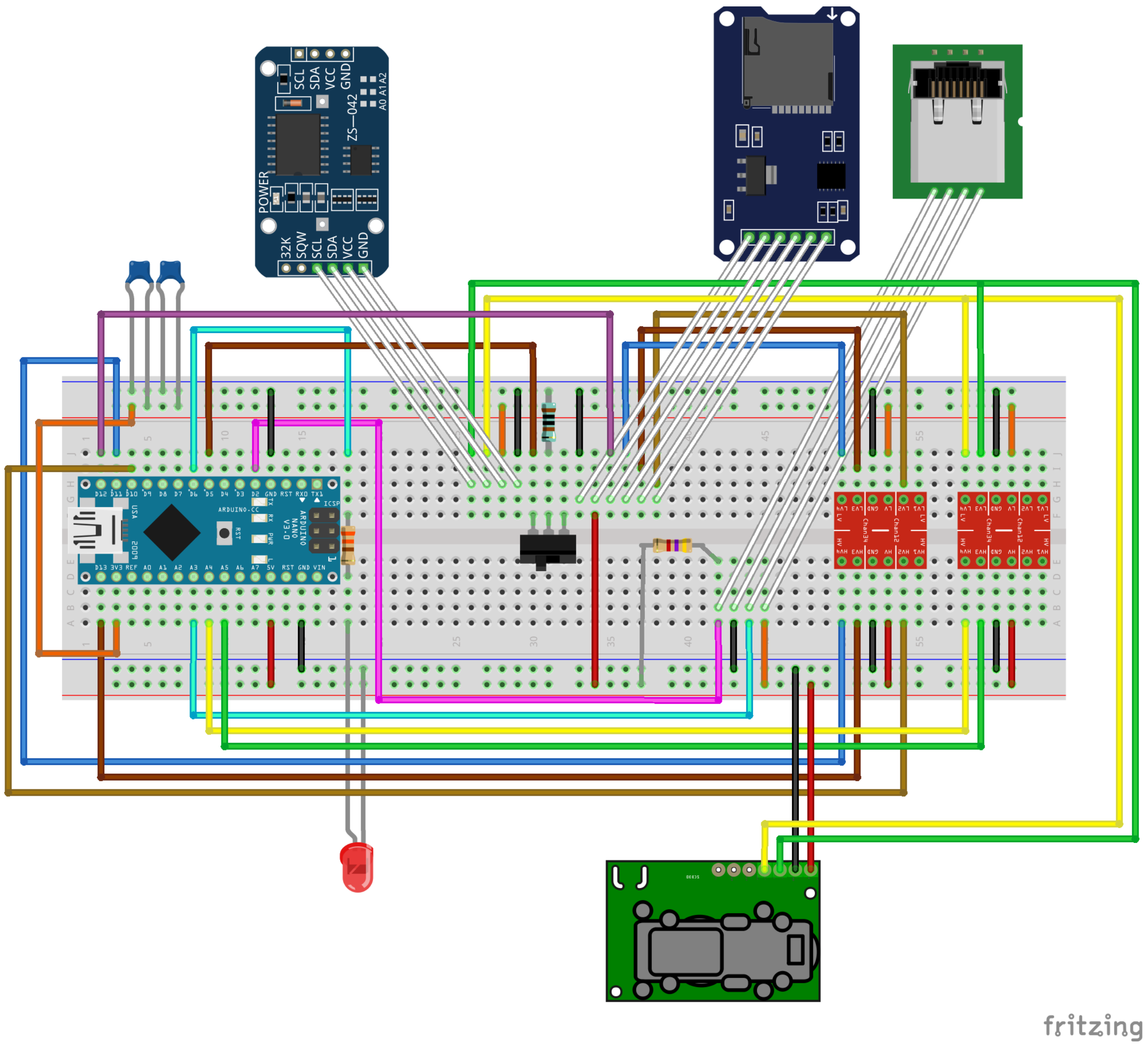

Figure 8. Breadboard Diagram of the Arduino Version.

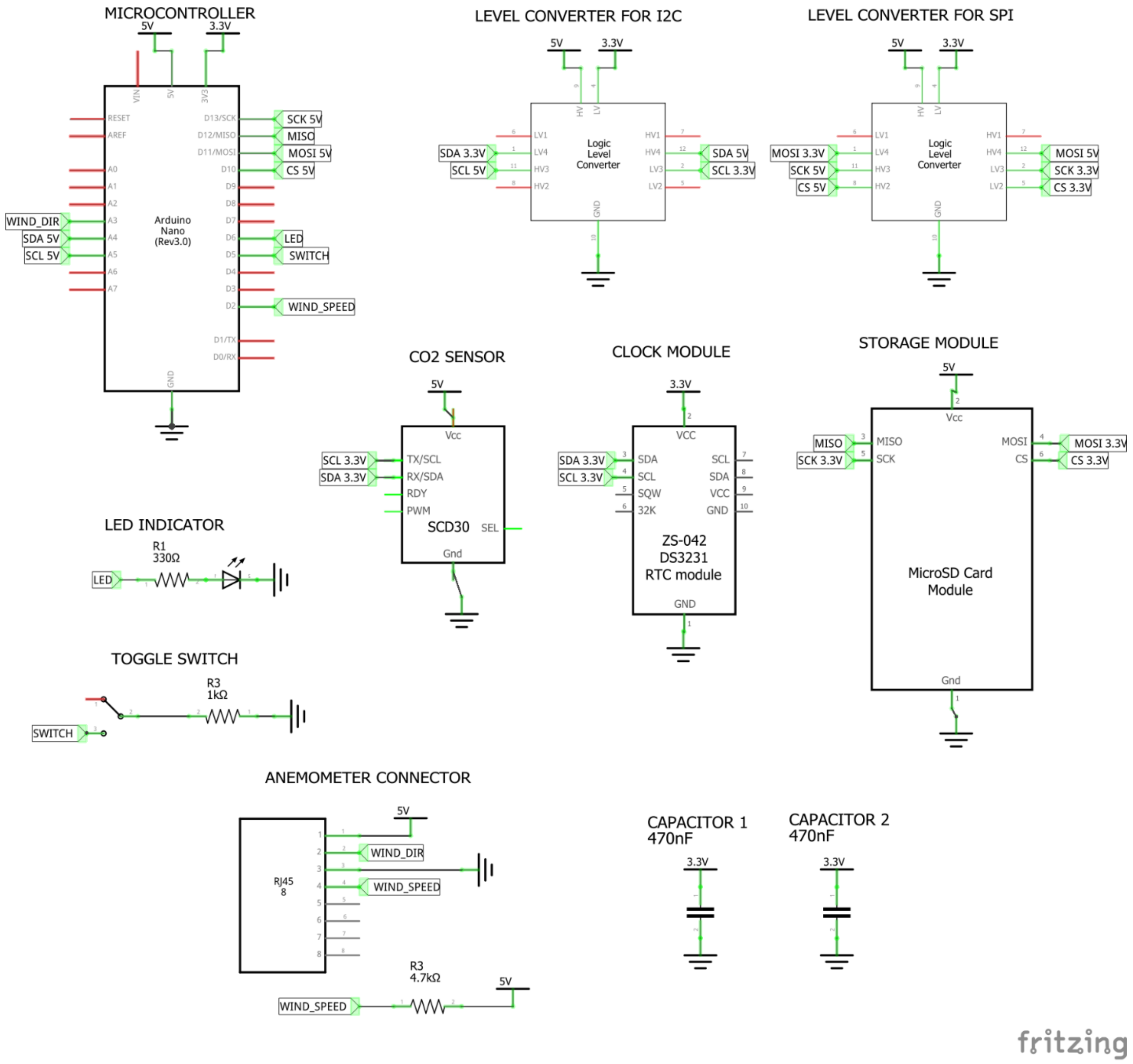

Figure 9. Electrical Schematic of the Arduino Version.

The step-by-step procedure for assembling the *Arduino Version* is as follows:

1. Orient a long solderless breadboard so that a blue line rail is at the top and a red line rail is at the bottom. Then, place the following components onto a long solderless breadboard:
    - Arduino Nano
    - Two Logic Converters
    - Two 470 nF Capacitors
    - Toggle Switch

2. Make the following connections using wires:
    - [-] 3V3 of the Nano to the top positive red rail
    - [-] Top GND of the Nano to the top negative blue rail
    - [-] LV pins of both Logic Converters to the top positive red rail
    - [-] Top GND pins of both Logic Converters to the top negative blue rail
    - [-] 5V of the Nano to the bottom positive red rail
    - [-] Bottom GND of the Nano to the bottom negative blue rail
    - [-] HV pins of both Logic Converters to the bottom positive red rail
    - [-] Bottom GND pins of both Logic Converters to the bottom negative blue rail
    - [-] HV4 of the left Logic Converter to D11 pin of the Nano
    - [-] HV3 of the left Logic Converter to D13 pin of the Nano
    - [-] HV2 of the left Logic Converter to D10 pin of the Nano
    - [-] HV4 of the right Logic Converter to A4 pin of the Nano
    - [-] HV3 of the right Logic Converter to A5 pin of the Nano

3. Place the 330-ohm resistor, 1000-ohm resistor, and LED onto the breadboard. Then make the following connections:
    - [-] D6 pin of the Nano to the 330-ohm resistor
    - [-] D5 pin of the Nano to the left pin of the Switch
    - Note: The 330-ohm resistor is associated with the LED, and the 1000-ohm resistor is associated with the Toggle Switch

4. Connect the CO2 sensor by making the following connections:
    - [-] VIN pin of the sensor to the 5V rail (bottom red rail)
    - [-] GND pin of the sensor to the GND rail (bottom blue rail)
    - [-] SCL pin of the sensor to the LV3 pin of the right Logic Converter and to the area left of the Switch
    - [-] SDA pin of the sensor to the LV4 pin of the right Logic Converter and to the area left of the Switch

5. a) Prepare the RTC module by making the following connections:
    - [-] GND rail (top blue rail) to the area left of the Switch
    - [-] 3V3 rail (top red rail) to the area left of the Switch

   b) Insert the RTC module into the breadboard upside down and orient in a way that ensures the following connections:
    - [-] GND pin of the RTC module to the GND line
    - [-] VCC pin of the RTC module to the 3V3 line
    - [-] SDA pin of the RTC module to the SDA line
    - [-] SCL pin of the RTC module to the SCL line

6. a) Prepare the SD-card module by making the following connections:
    - [-] GND rail (top blue rail) to the area right of the Switch
    - [-] 5V rail (bottom red rail) to the area right of the Switch
    - [-] D12 pin of the Nano to the area right of the Switch. This is the MISO line.
    - [-] LV2 pin of the left Logic Converter to the area right of the Switch. This will be the CS line
    - [-] LV3 pin of the left Logic Converter to the area right of the Switch. This will be the SCK line
    - [-] LV4 pin of the left Logic Converter to the area right of the Switch. This will be the MOSI line

   b) Insert the microSD-card module into the breadboard upside down and orient in a way that ensures the following connections:
    - [-] GND pin of the card module to the GND line
    - [-] VCC pin of the card module to the 5V line
    - [-] MISO pin of the card module to the MISO line
    - [-] MOSI pin of the card module to the MOSI line
    - [-] SCK pin of the card module to the SCK line
    - [-] CS pin of the card module to the CS line

7. a) Prepare the connector for the Davis Anemometer by connecting the following:
    - [-] D2 pin of the Nano to the area next to the left Logic Converter
    - [-] GND rail (bottom blue rail) to the area next to the left Logic Converter
    - [-] A3 pin of the Nano to the area next to the left Logic Converter
    - [-] 5V rail (bottom red rail) to the area next to the left Logic Converter

   b) Insert a 4.7 kilo-ohm resistor between the D2 line and the 5V rail

   c) Insert the RJ-11 connector into the breadboard and have the port face the 3V3 rail.

   d) Insert the Davis anemometer wire into the RJ-11 connector.

Figure 10 showcases what the *Arduino Version* looks like when assembled correctly. Following the procedure above does not guarantee that the *Node* operates correctly. If the *Node* does not function properly after uploading the software, then the possible causes are:

- Loose wirings
- Poor pin connections
- Faulty modules
- Inadequate power supply
- Incorrect placements of modules
- Incorrect wiring connections

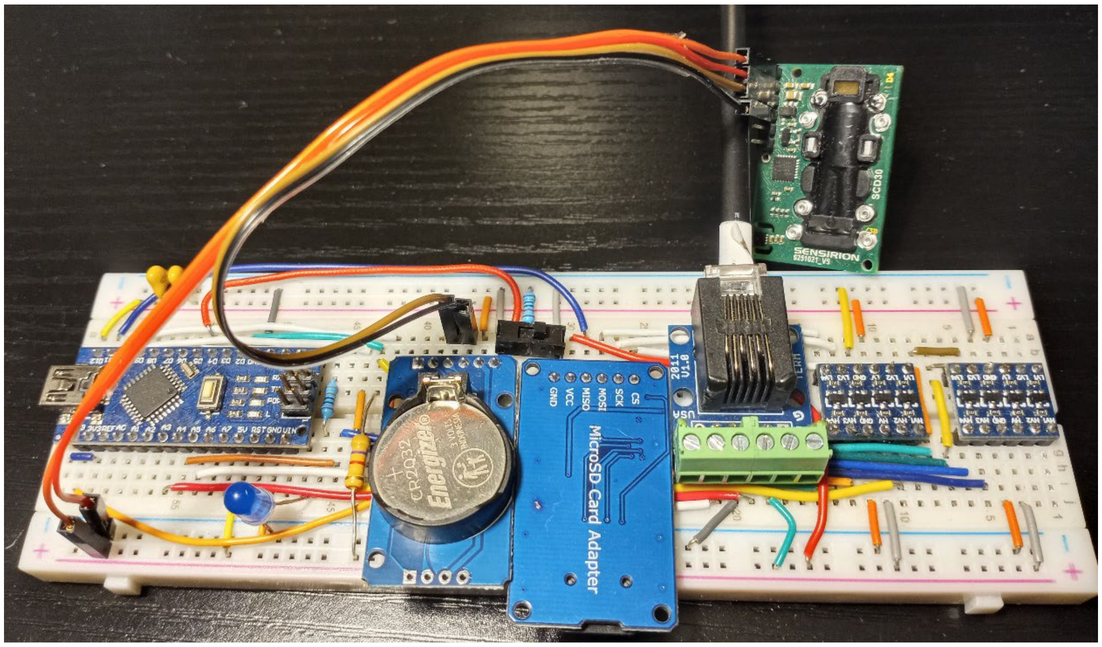

Figure 10. Arduino Version of the Sensor Node.

IV. Programming

The *Sensor Node* code is "ArduinoSensorNodeCode.ino", written using Arduino IDE. The code is divided into 5 different sections: *Libraries*, *Variables*, *Functions*, *Setup*, and *Main Loop*. The *Libraries* section loads packages containing necessary functions. Then, the *Variables* section defines most of the necessary variables and pre-allocates the necessary space. Next, the *Functions* part loads user-defined functions. Following is the *Setup* portion that activates the various components of the *Node*. Lastly, the *Main Loop* section is where the *Node* continuously records data in an endless loop.

a. Libraries

All libraries are either built-in or found in the Library Manager of the Arduino IDE. Below is the list of necessary libraries and the associated author if the library is found in the Library Manager.

• "Wire"	already built-in
• "avr/wdt"	already built-in
• "RTClib"	by Adafruit
• "SdFat – Adafruit Fork"	by Bill Greiman
• "Sparkfun_SCD30_Arduino_Library"	by SparkFun Electronics

b. Variables

The pre-allocated variables set up important parameters and sizes for various program parts. The variables set up the following:

- SD module
- SD file
- CO2 sensor
- RTC module
- RTC time
- Anemometer pins
- LED pin
- CO2 data
- Wind speed
- Wind direction
- Wind direction offset

None of the variables are intended to be altered, except the "Offset" variable. This variable defines the direction offset of the wind anemometer relative to the true North. This number is set to 0 by default but can be changed accordingly.

c. Functions

Most of the functions in the software come from libraries. However, some custom functions are explicitly written for the *WSN* application. Below is a list of user-defined functions and their purpose:

- *CreateNewFile*: Creates a new CSV file in the microSD card, where the name is the timestamp from its time of creation.
- *WriteSample*: Saves collected data into a file in the microSD card. The data is stored in binary format, consuming less space than ASCII format.
- *CollectGas*: Collects data from the SCD30 sensor. If collection fails, a second attempt is performed. If the second attempt also fails, then zero data values are given for that timestamp.
- *WindRotation*: Increments the wind counter whenever a cup rotation occurs.
- *WindDirection*: Converts the raw data from the wind vane into a direction ranging from 1 to 360 degrees. The direction offset is applied to create a calibrated direction.
- *RTCBegin*: Boots up the RTC module. If boot-up fails, then attempts are continuously made until boot-up is successful.
- *SCD30Begin*: Boots up the SCD30 module. If boot-up fails, then attempts are continuously made until boot-up is successful.
- *SDBegin*: Boots up the SD module. If boot-up fails, then attempts are continuously made until boot-up is successful.

d. Setup

The *Setup* portion runs various actions to prepare the *Node* for continuous operation. Below is the order of operations:

1. Activate the Serial Monitor
2. Activate the I^2C Wire lines
3. Activate the LED, wind speed, and toggle switch pins
4. Boot-up the RTC module
5. Boot-up the SCD30 module
6. Boot-up the microSD card module
7. Activate the Watchdog Timer

The purpose of the Watchdog Timer is to force a system reset if the *Node* gets stuck for longer than 8 seconds. When the timer is activated, it must be "fed" within 8 seconds to reset the timer. If the timer exceeds 8 seconds, then a system reset occurs.

e. Main Loop

This portion of the code makes the *Node* run forever in a loop, as long as there is active power. The procedure of the loop is as follows:

1. Retrieve the timestamp
2. Convert the timestamp into Unix time
3. Collect the wind direction
4. Collect the CO_2 and temperature data
5. Wait until 3 seconds have passed according to the RTC module
6. Collect the number of wind cup rotations
7. Reset the rotation number to zero
8. Feed the Watchdog Timer to reset the timer

9. Repeat steps 1 to 8 four more times to create five timestamps in total
10. Store all five timestamps into the microSD card file at the same time
11. Check if an hour has passed since the file was created. If an hour has passed, then create a new CSV file
12. Repeat steps 1 to 11 indefinitely

Additional resources are contained in the Wireless Sensor Network repository[10] and the CSR Arduino Collection repository[11].

References and Useful Links

[1] Image of Arduino Uno and Arduino Nano
https://qph.fs.quoracdn.net/main-qimg-c751237a23c20be8d62be1bda7672747.webp

[2] Purchase Link for SCD30
https://sensirion.com/products/catalog/SCD30/

[3] Guide for Davis Anemometer
http://cactus.io/hookups/weather/anemometer

[4] Purchase Link for RJ-11 Connector
https://www.mouser.com/ProductDetail/Gravitech/RJ11-TERM?qs=WZeyYeqMOWd7pvPuxF1aQw%3D%3D

[5] Purchase Link for Davis Anemometer
https://www.davisinstruments.com/products/anemometer-for-vantage-pro2-vantage-pro

[6] Purchase Link for RTC
https://www.eitkw.com/product/zs-042-rtc-real-time-clock-module/

[7] Image of RTC charging circuit
https://forum.arduino.cc/t/zs-042-ds3231-rtc-module/268862

[8] Image of microSD card module
https://m.media-amazon.com/images/I/61CM7AI-HoL._AC_SX679_.jpg

[9] Purchase link for Logic Converter
https://www.sparkfun.com/products/12009

[10] Wireless Sensor Network Repository
https://github.com/jkub6/WirelessSensorNetwork

[11] CSR Arduino Collection Repository
https://github.com/RiceAllDay22/CSR_Arduino_Collection

Contact

For any questions or assistance, email Adriann Liceralde at adriann8399@gmail.com