# Chapter 1 Introduction

## Introduction

As the world’s primary means of providing food for its ever-increasing population, agriculture is one of the most crucial parts of the global economy. As the world’s population grows, so does its demand for food. This, in turn, creates a need for a faster and more efficient food production. There are several agricultural practices that have been developed to meet this need. One of those practices is hydroponics, which present quite a few benefits over traditional agriculture. Hydroponics offers faster growth rates, higher overall yields, and reduced water usage (CITATION NEEDED). However, the use of hydroponics introduces an increase in difficulty. Various environmental factors need to be carefully monitored to ensure optimal growth, such as temperature, humidity, CO2 levels, pH levels, and nutrient PPM (CITATION NEEDED). Fortunately, internet of things (IoT) technology has advanced considerably, which has made it easier and cheaper to implement than ever. Utilizing a sensor array utilizing IoT technology will allow farmers to monitor the environmental conditions in real-time. Not only that, but advancements in cloud computing technology have also similarly made it easier and cheaper to implement than ever. Combining the reach provided by IoT and the flexibility provided by cloud can lead to the creation of a truly powerful solution. Thus, I propose the creation of an IoT-enabled hydroponic farm monitoring system using Arduino and cloud.

## Background

Hydroponics technology is not new, the earliest recorded example of a hydroponic farm is the Hanging Gardens of Babylon. Unfortunately, the technology of the past rendered hydroponics a non-viable agricultural practice when compared to traditional agriculture. Due to advancements in technology and the need for more sustainable agricultural practices, hydroponics has seen a rise in popularity and viability (CITATION NEEDED).

Due to the sensitivity of hydroponics to changes in the environment, hydroponics requires careful monitoring to ensure optimal plant growth. When combined with the fact that the relevant environmental factors usually fluctuate relatively frequently, manual methods for monitoring and inspection become time-consuming and can produce inconsistencies (CITATION NEEDED).

The emergency of IoT technology as potent tools for monitoring hydroponics in real-time allows farmers to adjust conditions as needed. One of the most popular platforms for IoT due to its versatility, low cost, and ease of use is Arduino (Datta et al., 2020). Cloud-based technologies like Google Cloud Platform (GCP) and Amazon Web Services (AWS) can serve as a store for data and a platform on which a support system can be deployed on, allowing unparalleled ease of access to the system, which can provide farmers with invaluable insight into the conditions of their hydroponics (CITATION NEEDED).

In conclusion, although difficult, advancements in technology have made hydroponics a viable and desirable alternative to traditional agriculture. Two of those technologies are IoT and cloud computing. By combining both technologies to create a hydroponic farm monitoring system, farmers can obtain invaluable information in a consistent, reliable, and timely manner. This availability can allow farmers to make quick adjustments to ensure optimal plant growth that would have otherwise been impossible to do with a meaningful degree of consistency.

## Research Questions

1. Is it possible to devise an Arduino-based and cloud powered solution to track changes in a hydroponic plantation’s environmental factors?
2. Is it possible for the Arduino device to communicate the data it collected into a database without a need for direct involvement from the staff during day-to-day operations?
3. Is it possible to create an automated website that can compile the information collected from the Arduino and present it this information in an easy-to-understand manner?

## 1.4 Scope

The research and development of this project is defined and limited to the following scope:

1. The creation of an IoT-enabled hydroponic environment monitoring device powered by Arduino.
   1. Due to the vastly increased complexity otherwise and the limited time and resources allocated to the project, the capability of the device will be strictly limited to monitoring environmental variables. Its function is strictly to provide information, it will not be able to perform any actions that can directly influence the hydroponic farm.
   2. The environmental variables stated in point (a) will be limited to five:
      1. Temperature
      2. Humidity
      3. CO2 Levels
      4. pH Levels
      5. Nutrient PPM
2. The creation of a web-based application to act as a terminal for the user to access the system and as a receiver for data from the Arduino.
   1. In its role as a terminal for users, the web application will have a user interface.
      1. The user interface will be able to display data collected from the Arduino.
      2. The user interface will have a feature that allows the user to select specific hydroponic “houses” and the “trays” or “rows” inside of them.
      3. The user interface will also have a feature that allows the user to filter data based on a range of time.
   2. In its role as a receiver for data from the Arduino, the web application will include a URL-based API for the Arduino to access.
      1. The data will be passed on via PHP’s $\_GET super global variable passed by URL.
      2. The data will then be processed by the API code and be automatically inserted into the database.
   3. As a rudimentary website that features neither valuable data nor features that can be a security risk to the hydroponic farm, the security feature of the website will be limited to a simple password. Individual user accounts are deemed to be unnecessary, as the website will not feature differing access levels/tiers.
   4. Due to the simple nature of the web application, for ease of development due to the limited time and resources, it will be developed using the base PHP 8. It will not utilize any frameworks.

## Aims & Benefits

The aim of this case study is to create a hydroponic farm environment monitoring system using an Arduino-based sensor and a web-based application to be utilized for the day-to-day monitoring activities of JUST HYDROPONICS’s owners and employees with the following objectives:

1. Improving the ease, speed, efficiency, and effectiveness of the hydroponics environment data collection in JUST HYDROPONICS.
2. Migrating the hydroponics environment data collection from a manual process using single-use test kits to an automated process using sensors that tirelessly monitor and collect data nonstop.
3. Migrating the data storage means from non-secure whiteboards to a database that can store multiple years’ worth of data with relatively small storage requirements.

By utilizing this hydroponics monitoring system, JUST HYDROPONICS will be able to make the following improvements:

1. Increased operational efficiency, which has the potential to reduce costs.
2. Achieve consistency through a systemized and automated process instead of the individual interpretations of each of their staff members by manual testing.
3. Increasing the security and longevity of their data.
4. Make it a lot easier to read the data in a centralized system rather than individual whiteboards in each hydroponic “block”.
5. Save costs by making it no longer necessary to purchase test kits.
6. Increased ability to make timely decisions due to more consistent and frequent data collection.

## Report Structure

### Chapter 1 – Introduction

This chapter delivers the introduction and background of the topic and solution to this case study. It also contains other information that provides an overview of this case study, including the scope, aims & benefits, and this short explanation on the contents of each chapter.

### Chapter 2 – Theoretical Foundation

This chapter will cover the explain theoretical concepts, principles, and literature relevant to the case study. It includes a variety of documentations and sources both academic and non-academic that serve as the conceptual basis of this case study, both for the web application and the IoT sensor. As the web application utilizes PHP for the primary application, MySQL as the database, and Apache HTTP Server as the web server, this chapter contains explanations, documentations, and sources of various comprehensiveness for all of them. As the IoT sensor utilizes Arduino and several Arduino-compatible sensors and modules, the explanations, documentations, and sources for these is likewise included.

### Chapter 3 – Problem Analysis

This chapter covers the analysis of the problems and challenges faced by JUST HYDROPONICS’s owners in their day-to-day operations. It discusses the limitations currently faced by the hydroponic owners and discusses in short, the possible directions that the solution can take.

By understanding the limitations of the existing system, we can ensure that the new Arduino-based automated solution is able to address the issues faced by the farm with their current processes.

### Chapter 4 – System Design

This chapter describes how the devised solution can address the problems identified in the previous chapter. It describes the design of the proposed solution, including the required hardware components, software, and the integrated system architecture.

### Chapter 5 – System Implementation

This chapter describes the process of implementation and testing of the system. It discusses challenges that I encountered during implementation and testing, the solutions I adopted to solve those challenges, and the results of my implementation and testing.

### Chapter 6 – Discussion & Evaluation

This chapter provides an analysis of the performance of the system, including any room for improvement and its current limitations.

### Chapter 7 – Conclusion & Recommendation

The final chapter summarizes the key findings of this case study, restates the research questions, and answers them based on the findings of the case study. It will also provide recommendations for future research in this field and the practical applications of the system.

# Chapter 2 Theoretical Foundation

## 2.1 Theoretical Foundation of the Hydroponic Sensor

### 2.1.1 Arduino Uno

Arduino is an open-source hardware and software company based in Italy that designs and manufactures easy-to-use electronics (*About Arduino*, 2021). Arduino’s open-source nature allows it to be easily modified and reproduced to suit the needs of each individual project that involves Arduino. Arduino’s easy-to-use nature makes it a viable platform to serve as the basis of this project’s hydroponic sensor.

A close-up of a blue circuit board

Description automatically generated with medium confidence  
Figure 2.x – Arduino Uno R3 16U2

There are many types of Arduino boards available in the commercial market, each with their own microcontroller. The Arduino board chosen for this project is the Arduino Uno R3 16U2, and according to freeCodeCamp.org (2021), it has the following specifications:

1. Microcontroller: ATMEGA328P
2. Operating Voltage: 5V
3. Input Voltage (Recommended): 7-12V.
4. Input Voltage (Limits): 6-20V
5. Digital I/O Pins: 14 (6 PWM)
6. Analog Input Pins: 6
7. DC Current On I/O Pins: 40mA
8. DC Current On 3.3V Pin: 50mA
9. Flash Memory: 32KB (0.5KB Occupied by Bootloader)
10. SRAM: 2KB (ATMEGA328P)
11. EEPROM: 1KB (ATMEGA328P)
12. Clock Speed: 16 MHz

Graphical user interface

Description automatically generated  
Figure 2.x – Arduino Schematic *(freeCodeCamp.org, 2021)*

As described by freeCodeCamp.org (2021), the Arduino Uno board can be divided into these 10 typical components or component groups:

1. The first group contains the digital pins.
   1. They are numbered 0 to 13.
   2. These digital pins are usually operated using the *digitalRead()* and *digitalWrite()* functions.
   3. Some of the pins have a tilde (~) symbol, which indicate that they support PWM.
   4. The PWM-capable pins can use the *analogWrite()* function.
2. The second group contains a built-in LED.
   1. By default, this LED is connected to pin 13.
   2. When the pin is set to *HIGH*, the LED will turn on.
   3. When the pin is set to *LOW*, the LED will turn off.
3. The third group contains the Power LED. This LED indicates whether the Arduino board is powered (LED ON) or not (LED OFF).
4. The fourth group contains the ATMEGA328P microcontroller, which controls everything that happens on the Arduino board.
5. The fifth group contains the Analog Input pins.
   1. They are numbered from A0 to A5.
   2. Due to them being input pins, they are used with the *analogRead()* function.
6. The sixth group contains the built-in Power Pins.
   1. These pins on the Arduino board supports 3.3V, 5V, and grounding.
   2. These pins should not be used for high-current modules.
7. The seventh group contains the Power Connector.
   1. This connector allows the Arduino to be powered when not connected to a computer.
   2. The connector accepts 7V-12V DC.
8. The eighth group contains the RX and TX LEDs.
   1. These LEDs indicate when there is communication between the Arduino board and another device.
   2. The RX LED blinks when the Arduino board is receiving data.
   3. The TX LED blinks when the Arduino board is transmitting data.
9. The ninth group contains the USB jack.
10. The tenth group contains the Reset Button. This button resets the ATMEGA328P microcontroller but does not clear the Arduino’s memory (the code that is uploaded to it).

The Arduino board is most typically controlled by using the Arduino IDE, which can be downloaded from Arduino’s official website. At the time of this case study’s writing, the Arduino IDE’s latest release is version 2.0.4. The Arduino IDE uses the C++ language, though the support library used is a subset (limited version) of the standard C library due to Arduino’s very small RAM capacity (Shinde, 2023).

### 2.1.2 Wi-Fi Module

To allow the Arduino to communicate via Wi-Fi, I used an ESP8266-01 Wi-Fi module. According to Barela (2023), the ESP8266 is an inexpensive and powerful programmable Wi-Fi breakout board. The cost is magnitudes lower compared to previous solutions such as the Wi-Fi Shield and Arduino Yun

A picture containing electronics, circuit component, electronic engineering, electronic component

Description automatically generated  
Figure 2.x – The ESP8266 Module

The ESP8266 uses 3.3V and draws over 300mA at peak operations. Using 5V on the ESP8266 runs the risk of damaging it. The ESP8266 chip itself contains 64 KiB of instruction RAM, 96 KiB of data RAM, and 4 MiB of QIO FLASH (Ada, 2015).

A close-up of a circuit board

Description automatically generated with medium confidence  
Figure 2.x – A Schematic of the ESP8266-01 Module *(Electronoobs, 2019)*

The schematic above indicates the purpose of each of the 8 individual pins on the ESP8266-01 module, which is slightly different from the ESP8266 module.

A picture containing text, circuit, screenshot, electronic engineering

Description automatically generated  
<MIGHT NEED REPLACEMENT WITH OWN DESIGN? MIGHT NOT! SEEING AS IT’S JUST AN EXAMPLE>  
Figure 2.x – An Example Schematic of How the ESP8266-01 In Use *(Electronoobs, 2019)*

<TO DO: Example Code for Basic Functionalities.>

<TO DO: A LOT! Need extensive theory documentation of the ESP8266 module and commands. Seeing as there are not any readily available and easy-to-use libraries out there. Commands must be sent manually using print() and println().>

### 2.1.3 Temperature Sensor

To measure temperature, I used a DS18B20 sensor. The documentation for this sensor is provided by Maxim Integrated, however, the model that I’ve bought is a third-party wired long probe, which is the one on the left on the figure below.

Diagram

Description automatically generated with medium confidence  
Figure 2.x – Two Models of the DS18B20 Sensor

According to Maxim (2019), the DS18B20 is a digital thermometer capable of providing 9 to 12-bit Celsius temperature measurements. It’s capable of operating in either external power mode or parasitic power mode. Each DS18B20 unit has a unique 64-bit serial code, allowing multiple DS18B20s to use the same wire bus.

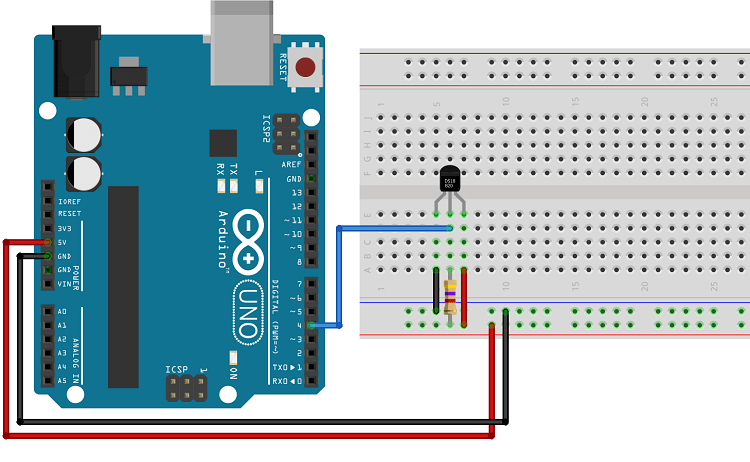
In more detail, according to Maxim (2019), the DS18B20 has the following features and specifications:

1. Capable of measuring temperature from -55 oC to +125 oC.
2. Has a ± 0.5 oC accuracy from -10 oC to + 85 oC.
3. Has a programmable resolution from 9-bit to 12-bit.
4. Capable of operating in parasitic power mode, which requires only the data and GND pins.

#### 2.1.3.1 External Power Mode

Diagram

Description automatically generated with medium confidence  
Figure 2.x – The DS18B20 Operating in External Power Mode *(Maxim, 2019)*

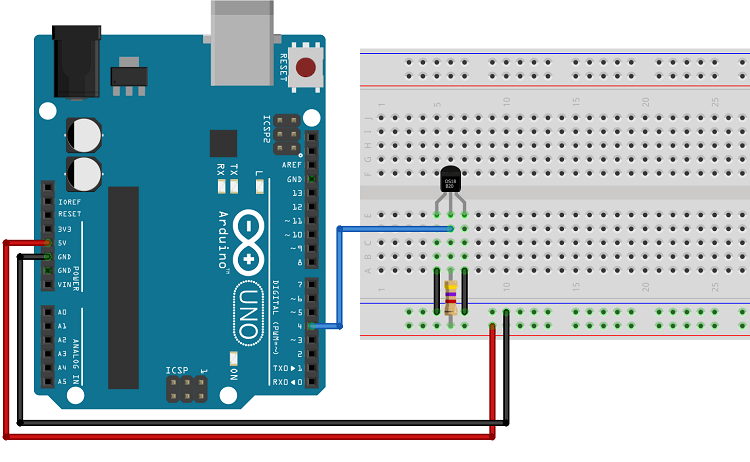
  
Figure 2.x – The DS18B20 Sensor Operating in Normal Mode *(Santos, 2016)*

In external power mode, the sensor requires 3 wire connections, with power being provided by the VDD pin (Santos, 2016).

#### 2.1.3.2 Parasitic Power Mode

A diagram of a bus

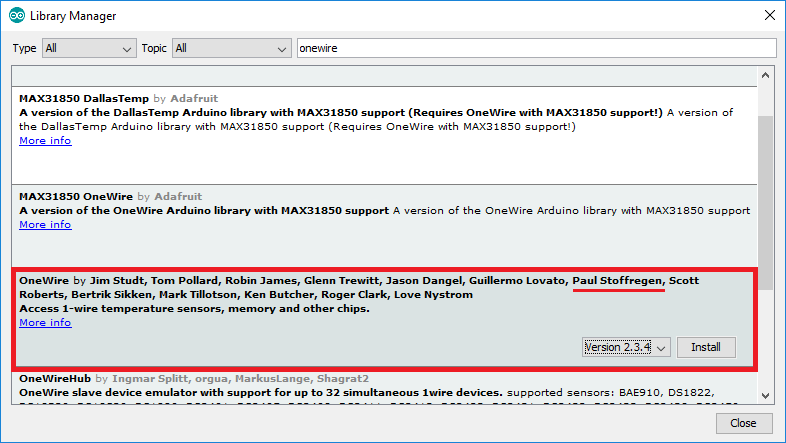
Description automatically generated with low confidence  
Figure 2.x – The DS18B20 Operating in Parasite Power Mode *(Maxim, 2019)*

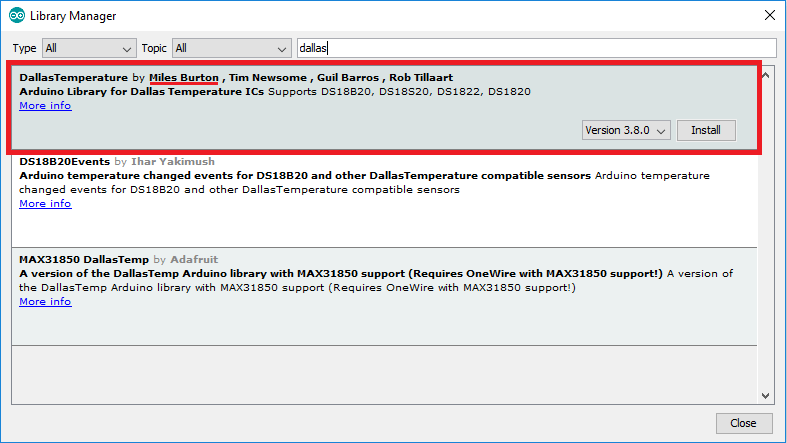
Figure 2.x – The DS18B20 Sensor Operating in Parasitic Power Mode *(Santos, 2016)*

In parasitic power mode, because the sensor draws power from the data line, the sensor only needs two pins, the data (DQ) and ground (GND) pins (Santos, 2016).

#### 2.1.3.3 Interfacing with the Sensor

Whichever power mode is chosen, to be able to successfully interface with the sensor, we need to install the OneWire and DallasTemperature libraries, which can be done from the “Manage Libraries” feature of the Arduino IDE (Santos, 2016).

  
Figure 2.x – The OneWire Library

  
Figure 2.x – The DallasTemperature Library

After arranging the components as shown in the figures above and installing the libraries, we can write a relatively simple code and upload it into our Arduino board to interface with the DS18B20 sensor.

#include “OneWire.h”

#include “DallasTemperature.h”

#define PIN\_TEMPERATURE 4

OneWire temperature(PIN\_TEMPERATURE);

DallasTemperature sensorTemperature(&temperature);

void setup() {

Serial.begin(115200);

sensorTemperature.begin();

}

void loop() {

sensorTemperature.requestTemperatures();

Serial.print(sensorTemperature.getTempCByIndex(0));

delay(5000);

}

### 2.1.4 Humidity Sensor

#### 2.1.4.1 Introduction

To measure humidity, I used a DHT11 sensor. According to Mouser Electronics (n.d.), the DHT11 is a sensor capable of measuring temperature and humidity with a calibrated digital signal output. The sensor’s humidity measurement component is a resistive-type component and an its temperature measurement component is NTC. The DHT11 sensor has 4 pins, however pin 3, as is represented in the figure below, is not connected (NC).

  
Figure 2.x – The DHT11 Sensor *(Mouser, n.d.)*

#### 2.1.4.2 Sensor Specifications

According to Mouser Electronics (n.d.), the DHT11 sensor has the following specifications:

1. Humidity Measurement Range: 20-90% RH
2. Humidity Accuracy: ± 5％ RH
3. Temperature Measurement Range: 0-50 oC
4. Temperature Accuracy: ± 2 oC

A picture containing diagram, text, line, font

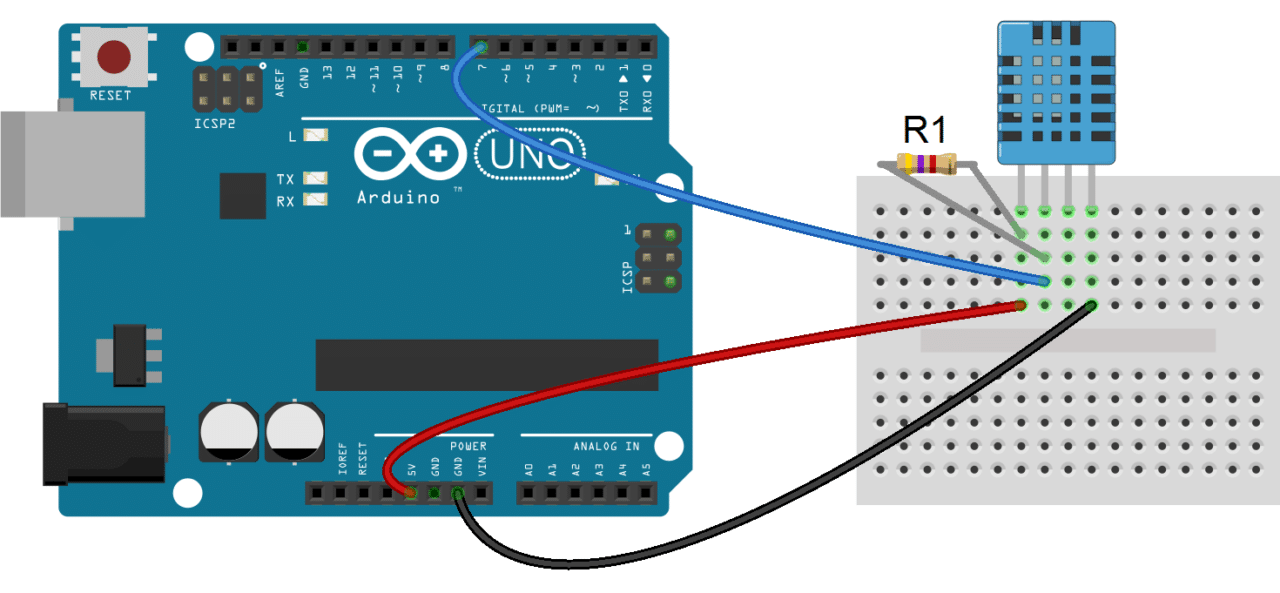
Description automatically generated  
Figure 2.x – Typical Application of the DHT11 Sensor *(Mouser, n.d.)*

The DHT11 is supplied by a 3-5.5V DC. When the sensor is powered, instructions should not be sent with less than 1 second delay to pass the unstable status (Mouser, n.d.)

If the DHT11 is operated outside of its specifications, it can result in a shift in the sensor’s calibration. Chemical vapors may also interfere with the sensitive elements of the DHT11 sensor, resulting in miscalibration or permanent damage. Should these calibration shifts happen, and given that the sensor is not permanently damaged, the DHT11 can self-calibrate over-time if kept within its operating boundaries (Mouser, n.d.)

#### 2.1.4.3 How It Works

The DHT11 sensor measures humidity (water vapor) by measuring the electrical resistance between two electrodes. Higher humidity results in lower resistance, while lower humidity results in higher resistance between the electrodes (Campbell, 2015).

  
Figure 2.x – Basic Schematic of a DHT11 Sensor in Use *(Campbell, 2015)*

To communicate and interface with the DHT11 sensor, we need to install the DHTLib library, which is available for download from a variety of websites on the internet. I obtained mine through GitHub.

#### 2.1.4.4 The Code

After installing the DHT11 sensor according to the schematic and installing the DHTLib library, we can write and upload the relatively simple code below to interface with the DHT11 sensor.

#include <dht.h>

#define PIN\_HUMIDITY 7

#define DHT\_TYPE DHT11

DHT sensorHumidity(PIN\_HUMIDITY, DHT\_TYPE);

void setup() {

Serial.begin(115200);

sensorHumidity.begin();

}

void loop() {

Serial.println(sensorHumidity.temperature);

Serial.println(sensorHumidity.humidity);

delay(1000);

}

### 2.1.5 CO2 Sensor

To measure CO2, I used an MH-Z19B sensor by Winsen Electronics. According to Winsen (2016), the MH-Z19B is a small-sized sensor using non-dispersive infrared (NDIR) technology to detect the presence of CO2 in the air. It has a built-in temperature compensation, UART output, and PWM input. The MH-Z19B has a high sensitivity and resolution, low power consumption, built-in passive countermeasures against water vapor interference, and a relatively long lifespan.

A close-up of a device

Description automatically generated with low confidence A whiteboard with black text

Description automatically generated with low confidence  
Figure 2.x – MH-Z19B Sensor *(Winsen, 2016)* Figure 2.x – MH-Z19B Diagram *(Winsen, 2016)*

According to Winsen (2016), as can be seen in the figure above, the MH-Z19B has nine pins, two of them are not used. The remainder 7 are:

* + **Vin**, which is the power input pin.
  + **GND**, which is the ground pin.
  + **PWM**, which is the PWM pin.
  + **Vo**, which is the analog output pin.
  + **RX**, which is the UART input pin.
  + **TX**, which is the UART output pin.
  + **HD**, which is used for calibration purposes.

According to Winsen (2016), the MH-Z19B sensor by Zhengzhou Winsen Electronics has the following specifications:

|  |  |
| --- | --- |
| Target Gas | Carbon Dioxide (CO2) |
| Voltage | 4.5V ~ 5.5V DC |
| Current (Average & Peak) | < 60mA @ 5V & 150mA @ 5V |
| Interface Level | 3.3 V (Compatible with 5V) |
| Measurement Range | 0 ~ 2000 ppm |
| 0 – 5000 ppm |
| Accuracy | ± (50ppm + 3% measured value) |
| Output Signal | UART (3.3V) |
| PWM |
| DAC (0.4V – 2V) |
| Working Temperature | 0 ~ 50 oC |
| Working Humidity | 0 ~ 90% RH (No Condensation) |

Table 2.x – MH-Z19B Technical Specification

According to Winsen (2016), the MH-Z19B sensor by Zhengzhou Winsen Electronics has the following recommended software settings:

|  |  |
| --- | --- |
| Baud Rate | 9600 |
| Data Bits | 8 |
| Stop Bits | 1 |
| Parity (Check Bits) | 0 (NO) |

Table 2.x – MH-Z19B Recommended Software Settings

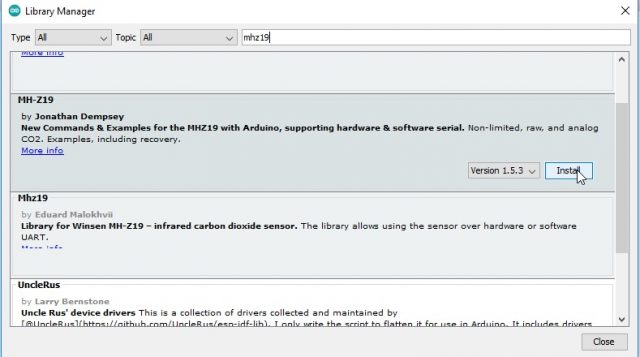
The sensor has an Automatic Baseline Correction (ABC) logic function, where the sensor will do a zero-point judgement and automatic calibration procedure every 24 hours after it is powered on. The automatic calibration has a zero-point of 400ppm. The sensor is also capable of manual and command calibration by utilizing the 0x87 and 0x88 commands. however, I’ve decided against using these manual methods, as my lack of expertise in this specific hardware might result in calibration inaccuracies that I might not be able to reverse (Winsen, 2016).

There are at least three ways to interface with the MH-Z19B sensor, which are digital, analog, and PWM.

#### 2.1.5.1 Digital Mode

The first method (digital) uses the UART (RX and TX) pins. This method requires the “MH-Z19” library. Though its theoretically possible to not use the library, it’s highly impractical to do so.

To digitally interface with the MH-Z19B sensor, we need to install the “MH-Z19B” library from the Arduino IDE’s “Manage Libraries” feature.

  
Feature 2.x – The MH-Z19B Library

In digital mode, the MH-Z19B sensor needs four connection points to the Arduino Uno board. The first is the Vin power pin, which needs to be connected to the 5V pin on the Arduino. The second is the GND pin, which needs to be connected to the GND pin on the Arduino. The third and fourth pins are the RX and TX pins, which needs to be connected to any of the Arduino’s digital pins.

A picture containing text, electronic engineering, electronic component, electronics

Description automatically generated  
Figure 2.x – MH-Z19B Schematic, Digital Mode *(Fahad, 2022)*

After installing the components as shown in the figure above and installing the MH-Z19B library, we can write and upload the relatively simple code below to communicate with the MH-Z19B sensor.

#include “Arduino.h”

#include "MHZ19.h"

#include “SoftwareSerial.h”

#define RXP 3

#define TXP 2

MHZ19 sensor;

SoftwareSerial mySerial(RXP, TXP);

void setup() {

Serial.begin(9600);

mySerial.begin(9600);

sensor.begin(mySerial);

sensor.autoCalibration();

}

void loop() {

int co2 = sensor.getCO2();

int8\_t temp = sensor.getTemperature();

Serial.print("CO2 Levels: ");

Serial.print(co2);

Serial.println(" ppm");

Serial.print("Temperature: ");

Serial.print(temp);

Serial.println(" Celcius");

delay(5000);

}

It’s important to note that the method above utilizes SoftwareSerial. While it is possible to have multiple instances of the SoftwareSerial class, only one can be used at a time, which might cause interference with other modules that requires SoftwareSerial (Arduino, 2023). Fortunately, interfacing with the MH-Z19B sensor using the analog method doesn’t have this problem.

#### 2.1.5.2 Analog Mode

The second method (analog) uses only the analog output (Vo) pin. This method does not require the “MH-Z19” library. In analog mode, the MH-Z19B sensor only needs three connection points. The Vin and GND pins need to be connected to the 5V and GND pins on the Arduino and the Vo pin needs to be connected to any one of the Arduino’s analog pins.

<image of MH-Z19 schematic in analog mode>  
Figure 2.x – MH-Z19 Schematic, Digital Mode

After arranging the components according to the figure and instructions above, we can upload the code below to communicate with the MH-Z19B sensor.

#include <Arduino.h>

#define PIN\_CO2 A0

void setup() {

Serial.begin(9600);

pinMode(PIN\_CO2, INPUT\_PULLUP);

}

void loop() {

float analogReading = analogRead(PIN\_CO2);

float co2Reading = 6.4995 \* analogReading - 590.53;

Serial.println(co2Reading);

}

#### 2.1.5.3 PWM Mode

The third method (PWM) uses only the PWM pin. This method also does not require the MH-Z19 library. Like Analog mode, the MH-Z19B sensor only needs three connection points with the Arduino. Again, the Vin and GND pins need to be connected to the 5V and GND pins on the Arduino board. The PWM pin needs to be connected to a PWM-capable digital pin on the Arduino board.

<image of MH-Z19 schematic in PWM mode>  
Figure 2.x – MH-Z19 Schematic, PWM Mode

After arranging the components according to the figure and instructions above, we can upload the code below to communicate with the MH-Z19B sensor.

#define PIN\_CO2 6

unsigned long pTime;

void setup() {

pinMode(DataPin, INPUT);

Serial.begin(152000);

}

void loop() {

pTime = pulseIn(DataPin, HIGH, 2000000) / 1000;

int co2ppm = 5000 \* pTime / 1004.0;

Serial.println(co2ppm);

delay(5000);

}

### 2.1.6 pH Sensor

To measure pH levels, I used a PH-4502C sensor kit. The sensor kit comes with the PH-4502C module and a pH probe.

#### 2.1.6.1 The PH-4502C Module

A close-up of a circuit board

Description automatically generated with medium confidence  
Figure 2.x – PH-4502C Module *(CimpleO, 2020)*

According to the figure above, the PH-4502C module has six pins on its left side, each with its own functions (CimpleO, 2020).

1. The TO pin is for temperature output.
2. The DO pin is for 3.3V output for pH limit.
3. The PO pin is the analog pH output.
4. The first GND is for the pH probe.
5. The second GND is for the board itself.
6. The VCC pin is the 5V power pin.

The PH-4502C board communicates pH value via voltage. The board defaults to pH 7 at 0 V, which means it will go minus when it reads acidic an pH value, which cannot be read by Arduino (CimpleO, 2020). This should be offset so that a pH value of 0 is represented by 0V, and a pH of 14 is represented by 5V.

#### 2.1.6.2 The Probe

<IMAGE OF THE PROBE, SELF TAKEN HERE>  
Figure 2.x – The pH Probe

<Brief explanation of the probe here>

#### 2.1.6.3 Setup

To setup the sensor, we need to connect 4 of the 6 available pins to the Arduino board. The PO pin should be connected to the Arduino A0 analog port. Both GND pins should be connected to the Arduino’s GND ports. The VCC pin should be connected to the Arduino’s 5V port.

A close-up of a circuit board

Description automatically generated with medium confidence  
<IMAGE TO BE REPLACED W/SELF TAKEN IMAGE. DIAGRAM FORMAT PLS>  
Figure 2.x – A Schematic of the PH-4502C with Arduino

#### 2.1.6.4 Calibration

To perform calibration, the potentiometer near the BNC connector needs to be turned to the correct offset. To set the offset, the BNC connector’s inner wire needs to be shorted with its outer shell, as is shown in the figure below. This will simulate a neutral pH value of 7. After this, the value of the PO pin should be measured using a multimeter, and the offset on the potentiometer should be adjusted until it reads 2.5 V (CimpleO, 2020).

A picture containing electronic engineering, electronics, cable, machine

Description automatically generated  
<IMAGE TO BE REPLACED W/SELF TAKEN IMAGE>  
Figure 2.x – PH-4502C Probe Offset Calibration

According to CimpleO (2020), it is important to note that a pH probe take some time to get to the correct value (at least two minutes). The readings may also fluctuate if the temperature is above 30 oC or below 10 oC.

#### 2.1.6.5 The Code

After arranging the parts as instructed, we can use the code below to interface and communicate with the PH-4502C module.

#include “Arduino.h”

#define PIN\_PH A0 // ANALOG

void setup() {

Serial.begin(115200);

}

float calcPH(float volts) {

return ((2.5 - volt) / 0.18) + 7;

}

void loop() {

int capture = 0;

for(int i = 0 ; i < 10 ; ++i) {

capture += analogRead(PIN\_PH);

}

float volts = 5 / 1023.0 \* measurings / 10;

Serial.println(calcPH(volts));

delay(5000);

}

### 2.1.7 PPM Sensor

#### 2.1.7.1 Introduction

To measure PPM (parts per million), I used an Analog TDS Sensor by DFRobot. According to DFRobot (n.d.), this TDS sensor provides an analog output that is Arduino compatible. It takes a wide 3.3V ~ 5.5V input and outputs 0V ~ 2.3V analog. The TDS probe is also waterproof and can be submerged in water for extended periods of time.

A glass jar with clear liquid and wires

Description automatically generated with low confidence  
Figure 2.x – DFRobot Analog TDS Sensor, Use Case Illustration *(DFRobot, n.d.)*

#### 2.1.7.2 Specifications

The Analog TDS Sensor kit includes two devices in a single set, which are the signal transmitter board and the TDS probe. These devices come with the following specifications (DFRobot, n.d.).

|  |  |
| --- | --- |
| Input Voltage | 3.3 ~ 5.5V |
| Output Voltage | 0 ~ 2.3V |
| Working Current | 3 ~ 6mA |
| Measurement Range | 0 ~ 1000ppm |
| Measurement Accuracy | ± 10% F.S. (25 oC) |
| Module Interface | PH2.0-3P |
| Electrode Interface | XH2.54-2P |

Table 2.x – Specifications of the TDS Signal Transmitter Board *(DFRobot, n.d.)*

|  |  |
| --- | --- |
| Number of Needles | 2 |
| Connection Interface | XH2.54-2P |
| Waterproof | Yes |

Table 2.x – Specifications of the TDS Probe *(DFRobot, n.d.)*

As is stated by the specifications above, the sensor kit’s Signal Transmitter Board connects to the probe via the XH2.54-2P interface, while it communicates with the Arduino board using the PH2.0-3P interface. The details of the two interfaces can be seen in the following image.

Graphical user interface

Description automatically generated  
Figure 2.x – Signal Transmitter Board *(DFRobot, n.d.)*

Shown in the figure above in numbers 1 to 3 is the PH2.0-3P connection interface, which is just a compacted version of the individual connections that make it up. As is suggested by the writings on the figure above, number 1 is the Power GND, number 2 is the Power VCC (3.3 ~ 5.5V), and number 3 is the Analog Signal Output. On the left side of the figure is the XH2.54-2P connector, which is supposed to connect to the probe. On the top right corner of the figure is number 5, which is the LED power indicator.

#### 2.1.7.3 How It Works

A picture containing text, electronics

Description automatically generated  
Figure 2.x – Example Use of the Analog TDS Sensor Kit *(DFRobot, n.d.)*

Shown in the figure above is an example of how this sensor kit can be used with Arduino. The PH2.0-3P connection interface is split into three, with GND connecting to GND, VCC connecting to 5V, and Analog connecting to A0 ~ A5 on the Arduino board. The probe is also connected to the other side of the board and is dipped into the medium intended to be measured.

#### 2.1.7.4 Calculating PPM

Before we continue, there is the very important matter of calculating the TDS PPM, which is not as simple as one might think. This sensor by DFRobot measures TDS PPM by measuring electrical conductivity. In fact, most TDS sensors are just electrical conductivity sensors with automatic conversion to PPM.

According to Fernandez (2014), electrical conductivity is measured by passing a small amount of electricity between two electrodes. Pure distilled water has no electrical conductivity, so adding solids like hydroponic nutrients increases its conductivity. Electrical conductivity is measured using a unit called siemens, though its more common to use milisiemens (mS) or microsiemens (μS).

There are many methods used to measure TDS PPM from electrical conductivity, and this usually results in confusion for people that are inexperienced in measuring them. But according to DFRobot (n.d.), their sensor uses the NaCl conversion factor which is 0.47 rounded up to 0.5. This conversion factor is multiplied by the electrical conductivity in microsiemens to produce PPM.

Even though this lack of standard is very confusing, fortunately Grotek Canada has set the standard conversion factor to use 0.5 (Fernandez, 2014).

#### *2.1.7.5 The Code*

After arranging the parts and connecting them to the Arduino as instructed in the figures above, we can use the code below to communicate with the TDS sensor.

#include “EEPROM.h”

#include "GravityTDS.h"

#define PIN\_NUTRIENT A1

GravityTDS sensorNutrient;

void setup() {

Serial.begin(115200);

sensorNutrient.setPin(PIN\_NUTRIENT);

sensorNutrient.setAref(5);

sensorNutrient.setAdcRange(1024);

sensorNutrient.begin();

}

void loop() {

sensorNutrient.setTemperature(waterTemperature);

sensorNutrient.update();

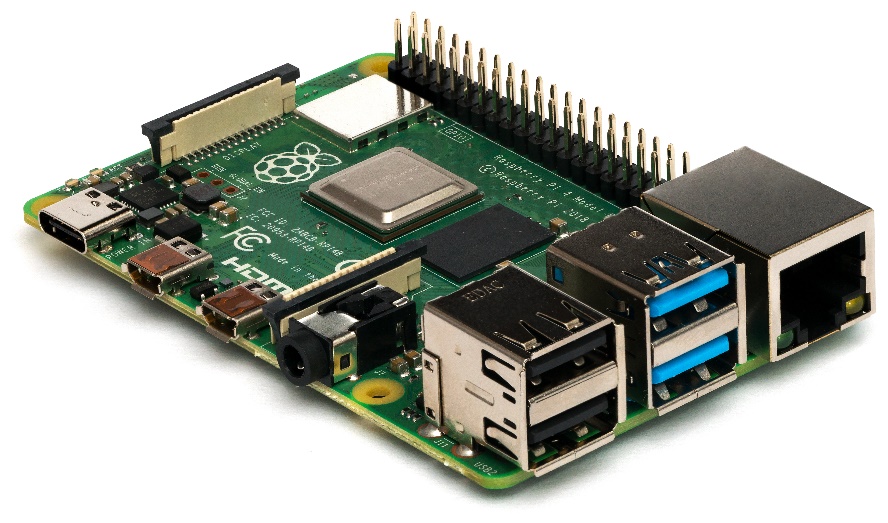
Serial.println(sensorNutrient.getTdsValue());

delay(5000);

}

### 2.1.8 Possible Alternatives

#### 2.1.8.1 Raspberry Pi

  
Figure 2.x – The Raspberry Pi

During the conception and ideation process of this case study, I considered using a Raspberry Pi instead of Arduino Uno but decided not to. There are several reasons for why I went with Arduino instead of Raspberry Pi.

To better understand my choice, first you need to understand what a Raspberry Pi is and how it’s different to an Arduino.

From my limited understanding of both platforms, Arduino is a small prototyping board with a small microcontroller, which is basically a very small and very slow computer. The amount of memory, storage, and processing power of an Arduino is very limited. An Arduino provides a flexible and very easy-to-use environment to design prototypes.

The Raspberry Pi on the other hand, though small, is a full-blown computer. Compared to the Arduino, the Raspberry Pi has a vastly faster processor, tremendously more memory, and supports a much larger storage in the form of SD cards. The Raspberry Pi can run full-blown operating systems like Linux.

For the purposes of designing the hardware of this case study, I could have gone either with Arduino or the Raspberry Pi, but I chose Arduino for several reasons:

* Arduino is much cheaper than the Pi, with a single Raspberry Pi costing over 20 times that of a single Arduino board.
* Sensor module availability is much greater with the Arduino, with almost any sensor that you can think of being available for the Arduino. Not necessarily true for the Raspberry Pi.
* Sensor modules for the Arduino are also much cheaper compared to the equivalents for Raspberry Pi.
* The capability of the Arduino Uno is sufficient for this case study. And should it be marginally insufficient, there are more powerful platforms such as the Arduino Mega, which are still much cheaper than a Raspberry Pi, which still costs 10 times as much as an Arduino Mega.
* Developing code for the Arduino is also much easier, as it has its own IDE and operating system. Developing software for the Raspberry Pi would be much more complex, as Raspberry Pis usually run Linux. Availability of software libraries also heavily favor the Arduino platform.

#### 2.1.8.2 …?

## 2.2 Theoretical Foundation of the Web Application

### 2.2.1 PHP

Hypertext Preprocessor (PHP) is a highly popular scripting language that is very commonly used in web development. It is a highly versatile and scalable, and features cross-platform compatibility, which makes it highly desirable for web developers around the globe.

One of PHP’s most significant traits is its ability to communicate with a variety of database management systems such as Oracle, PostgreSQL, and MySQL, which makes it easy for developers who wish to create websites that interact with databases (PHP, 2023).

PHP’s robust support for web protocols like HTTP, FTP, and many others makes it an adaptable and highly versatile language, which allows PHP to seamlessly integrate with other web application (PHP, 2023).

In addition, PHP is an already established and widely accepted programming language with a vast community of developers still actively using it. This large number of developers contribute to PHP’s development, making it evolve into ever more efficient forms. PHP’s open-source nature also makes it highly accessible and cost effective to customization, making it the de facto choice for many projects large and small.

This already established and widely recognized language makes it the ideal choice for this project, especially if the management of JUST HYDROPONICS wishes to pursue this solution further. The abundance of developers with expertise in PHP makes it easy to find developers who are willing to work on this project further, far after the conclusion of this case study.

### 2.2.2 HTML & CSS

Hypertext Markup Language (HTML) and Cascading Style Sheet (CSS) are the two quintessential languages in creating most modern web pages that are running today. HTML acts as the skeleton or structure of the website, which defines the layout of the website. CSS acts as the skin or appearance of the website, which defines the website’s visual styling.

According to *HTML Introduction* (n.d.), HTML is a markup language whose role is to define the structure and layout of webpages. It has a set of elements that can be used to define things such as paragraphs, titles, headings, and many more. These elements are then structured into something called a Document Object Model (DOM). The DOM’s role is to render the web page itself.

In contrast, according to *CSS Introduction* (n.d.), CSS is a style sheet which allows website designers to define the visible appearance of the website, which covers things like font, color, margins, and so on.

When combined, HTML and CSS allows for the creation of tidy, visually engaging, and user-friendly websites. Their nature of being flexible and easy to use has virtually turned it into the world’s standard for frontend website development.

### 2.2.3 Bootstrap

According to *Bootstrap* (n.d.), Bootstrap is a commonly utilized frontend web development framework developed by twitter for HTML, CSS, and JS. Bootstrap offers a set of easily customizable and pre-designed website components that allows its users to save development time by not having to develop them from scratch.

Bootstrap also utilizes CSS preprocessors such as SASS, which enables web designers to be more efficient with their CSS code. SASS allows developers to modify key elements that govern the look and feel of a website quickly, without having to change the values on each individual element. The framework of bootstrap also ensures a consistent feeling throughout the website, even if there are multiple designers with different styles.

### 2.2.4 XAMPP

XAMPP is an open source packaged Apache PHP development environment distribution containing MariaDB, PHP, and some other modules. XAMPP enables developers to set up and run web servers on their local machine with relative ease. I chose XAMPP not only because it’s easy to use, but also because it contains everything I need for the development and deployment testing of the web application on this case study, which are PHP, a MariaDB server, and a Web Server.

A screenshot of a computer

Description automatically generated with medium confidence  
Figure 2.x – XAMPP Control Panel, Stripped Down, Only Apache & MySQL

MariaDB is an open-source relational database, made by the original developers of MySQL. Because MariaDB is based on the MySQL source code, it works well with PHP’s built in MySQL function, it even uses the same TCP 3306 port by default (MariaDB, 2023). MariaDB is chosen as the database management system (DBMS) for this project due to is simplicity and reliability.

The Apache Web Server is an open-source web server and application deployment platform. It serves as the test deployment platform during the web application’s development process. The Apache Web Server can be hosted locally, with a very simple one-click sever start procedure. You only need to put the web application that needs to be deployed in the *htdocs* folder (XAMPP Team, 2023).

### 2.2.5 Google Cloud Platform

According to Google (2023), Google Cloud Platform (GCP) is a cloud computing service provider that offers a wide range of services for organizations and individuals. GCP provides infrastructure services with high-availability and a global reach. GCP’s easy-to-use nature makes it an attractive solution for organizations and individuals alike.

The primary attraction to GCP in the scope of this project is the easily accessible free tier and relatively simple deployment process. When it comes to cost, due to the reputation and quality offered by GCP, the cost is admittedly high.

# Chapter 3 Problem Analysis

## 3.1 Current Processes

<to be added>

## 3.2 Problem Analysis

### 3.2.1 Data Vulnerability

The data being physically stored on whiteboards adds a factor of extreme data vulnerability. It does not take much to cause the valuable historical data stored on these whiteboards to be wiped, without any possibility of recovery or redundancy. All it takes is an employee wiping the whiteboards absent-mindedly or rain to fall on an unsheltered whiteboard to have the data be gone forever.

A white board with writing on it

Description automatically generated with low confidence  
Figure 3.x – A Whiteboard Used for Storing Data

My migrating the data storage method to a cloud-based solution, data security and availability is greatly enhanced, leaving a vastly larger room for error and increasing data resilience.

### 3.2.2 Arduous Data Management

Storing data on whiteboards makes it extremely laborious to track all the data, making the task of data management a nightmare. Manually copying or transcribing the data onto a data ledger also increases the farm’s labor workload.

A white paper with writing on it

Description automatically generated with low confidence  
Figure 3.x – Arduous Data Management

By having the ability to manage data on a cloud-based web application platform, the task of data management becomes much easier, allowing very few (if any at all) man-hours to be allocated to the task of data management.

### 3.2.3 Labor Intensive

The current process of data collection is done manually, using test kits for each individual hydroponic block. This is a labor-intensive process, requiring large amounts of man-hours and consequently, a larger workforce. Combine this with the additional task of data management, and the amount of time taken up just to collect and maintain the data essential to the optimal operations of the hydroponic farm stacks up to a staggering amount.

Most of these processes can be automated by introducing an Arduino-based monitoring system. Because the Arduino sensors communicate with the database automatically, the farm staff only needs to make sure that the sensors are operating correctly and do the occasional maintenance. The task of data collection would be rendered completely redundant. Data maintenance becomes far easier and takes only a miniscule portion of how much time it would have taken if done manually.

### 3.2.4 Reduced Accuracy & Inconsistency

The manual nature of the current data collection process adds inherent inaccuracies and inconsistencies in the collected data due to the differences of how each employee might interpret the data. There is also the added factor of human error, where an employee might make a mistake during data input and transcription.

Because of the highly important nature of the data being collected, any mistakes may cause improper adjustments to the hydroponic system, causing suboptimality and imbalances that negatively impact the hydroponic system.

By using automated electronic sensors, we can reduce or outright eliminate most of the factors that are causing inaccuracies and inconsistencies. These sensors can also be calibrated to a single standard and are not affected by human error which can improve accuracy and ensures that the data collected is consistent.

### 3.2.5 Delayed Response

Manual data collection, though done frequently, cannot be done frequently enough to account for minute changes in the hydroponic system. The data is usually collected on a schedule, which causes a response delay should anything go wrong in the hydroponic system. These delays in response may significantly reduce the optimality of the hydroponic system which could have been prevented by simply being notified of the problem sooner.

In periods of expected abnormalities, the data can be collected much more frequently, but this is not sustainable in the long term, as manual measuring utilizes a lot of single-use test kits and are physically laborious to do and keep track of.

On the other hand, an automated system can collect data with much higher frequency, require little to no human input, is reusable, and can even be configured to alert its users of any anomalies should it be desired. The data collection frequency can be changed by simply modifying a variable and does not incur extra costs or administrative overheads.

### 3.2.6 Inefficient Allocation of Financial Resources

All the problems above combined cause an inefficient allocation of financial resources, where money is being spent to maintain things that really shouldn’t require maintenance and to put out fires that shouldn’t have been allowed to light in the first place.

The currently difficult and laborious practices of data management and manual data collection take up manpower that could have been better spent somewhere else. The error-prone nature of processes that are done manually also take up resources to fix. Delays in response to undesirable environmental changes to the hydroponic system also take up resources to revert the unwanted changes.

All these issues are solvable by implementing an automated system. Automation lowers maintenance costs, increases reliability, vastly decreases the workload on the farm staff, are more reliable and consistent, and keep monitoring around the clock.

# Chapter 4 Solution Design

## 4.1 System Overview

### 4.1.1 Architecture

The solution is divided into two major components. The first component is the physical Arduino-based sensor. The second component is the web application which allows for the information collected by the sensor to be displayed to the user.

The Arduino-based sensor is responsible for collecting information via an array of five sensors, which are the DS18B20, DHT11, MH-Z19B, PH-4502C, and DFRobot TDS Meter, each of them collecting temperature, humidity, CO2, pH value, and nutrient ppm respectively.

A picture containing text, screenshot, circuit

Description automatically generated  
Figure 4.x - Arduino System Architecture

The data collected by the sensors will then be compiled and formatted into a string URL which adheres to the PHP $\_GET superglobal URL pattern. The data will then be sent into the internet via the ESP8266 Wi-Fi module.

  
Figure 4.x – Superglobal URL Pattern

This action is performed continuously with a configurable delay so long as the device is powered, and nothing goes wrong in the process cycle.

A picture containing screenshot, graphics, darkness, black

Description automatically generated  
Figure 4.x – Process Cycle

The web application contains an API, which allows it to receive data transmissions from the Arduino via internet URL. The API acts as a passive listener, which is triggered only when the Arduino attempts to connect to the server to transmit data.

A picture containing screenshot, diagram, line, text

Description automatically generated  
Figure 4.x – Web Application System Architecture

The system requires an active internet connection. The internet connection used should be a basic, wireless capable connection with little to no firewall limitations or login requirements, as the Arduino device is incapable of independently connecting to a physical LAN port or fulfilling any additional requirements other than the basic SSID name and password.

The simple nature of the web application should allow for it to be deployed almost anywhere. The very-low bandwidth requirements allow for high connectivity tolerances. Theoretically, it should be able to be deployed on almost any third-party cloud service provider, or it could even be hosted locally, should the local internet connection have a dynamic IP address and is capable of port-forwarding. If the cloud service provider features additional security layers, it should be disabled as in it’s current form, the Arduino device’s design does not provide it with the capability for authentication and would likely be flagged as a spam.

## 4.2 Arduino

<Diagram image of the Arduino’s preliminary design. WIP>

Explain in detail about the design, about which ports (I.E. A0 A1 A2) are used by which sensor and for what. Explain the cable connections. Explain how it will work in semi-detail.

## 4.3 Web Application

### 4.3.1 Class Diagram

A screenshot of a computer

Description automatically generated with low confidence  
<The UML Class Diagram is a WIP. It’s done but there are more classes planned.>  
Figure 4.x – Class Diagram

In its current implementation, the Class Diagram has **<five>** classes. Due to the object-oriented nature of PHP, each of these classes serves a distinct purpose. The classes are:

* **Controller**
  + This is an abstract class. It serves as an abstraction layer for controller classes.
* **DashboardController**
  + As the name suggests, this class is a controller, which inherits the abstract *Controller* class. It controls the main page of the web application, which is the dashboard, which displays the data collected by the Arduino sensor device.
* **SensorData**
  + This class was created to classify, unify, and encapsulate the sensor data into a single object. This shortens the overall code length and improves code readability.
* **SensorDataArray**
  + This class was created specifically to retrieve an array of SensorData within a defined time frame.
  + This process would normally take approximately two dozen lines of code. By encapsulating this process behind a function within this class, overall code length is shortened, and readability is improved.
* **DatabaseConnection**
  + This class was created to handle most of the database connections and queries behind a layer of encapsulation.
  + This class also stores the connection object within its properties, so any queries can be performed by simply calling class’s *executeUpdate()* or *executeQuery()* functions.

### 4.3.2 Additional Features

### 4.3.3 Use Case

<Use Case Diagram image here!>

### 4.3.4 User Interface

A screenshot of a computer

Description automatically generated with medium confidence  
Figure 4.x – User Interface Main Page

Explain about the user interface design here.

## 4.4 Testing

Explain about the testing methods that will be used, what are expected etc. etc.

# Chapter 5 Solution Implementation

## 4.1 System Specification

## 4.2 Operational Procedures

## 4.3 Implementation Strategy

## 4.4 Test Plan

## 4.5 Module Testing

## 4.6 Integration Testing

# Chapter 6 Discussion & Evaluation

# Chapter 7 Conclusion & Recommendation

## 7.1 Conclusion

## 7.2 Recommendation

- Multi-device support.

- Casing design research.

- Economic viability.

# References

*About Arduino*. (2021, September 15). Arduino. <https://www.arduino.cc/en/about>

Ada, L. (2015). *Adafruit HUZZAH ESP8266 Breakout*. Adafruit. <https://learn.adafruit.com/adafruit-huzzah-esp8266-breakout/overview>

Arduino. (2023, June 26). *SoftwareSerial Library*. Arduino Documentation. <https://docs.arduino.cc/learn/built-in-libraries/software-serial>

Barela, A. (2023). *ESP8266 Temperature / Humidity Webserver*. Adafruit. <https://learn.adafruit.com/esp8266-temperature-slash-humidity-webserver/overview>

*Bootstrap – The most popular HTML, CSS, and JS library in the world.* (n.d.). Bootstrap. Retrieved April 17, 2023, from <https://getbootstrap.com/>

Campbell, S. (2015, October 1). *How to Set Up the DHT11 Humidity Sensor on an Arduino*. Circuit Basics. <https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/>

CimpleO, (2020, April 23). *Arduino pH-meter using PH-4502C*. CimpleO. <https://cimpleo.com/blog/simple-arduino-ph-meter/>

*CSS Introduction*. (n.d.). W3Schools. Retrieved April 17, 2023, from <https://www.w3schools.com/css/css_intro.asp>

DFRobot. (n.d.). *SEN0244 Gravity Analog TDS Sensor Meter for Arduino*. DFRobot Open-Source Hardware Electronics and Kits. Retrieved April 16, 2023, from <https://wiki.dfrobot.com/Gravity__Analog_TDS_Sensor___Meter_For_Arduino_SKU__SEN0244>

Electronoobs. (2019, August 4). ESP8266 + Arduino + database – Control Anything from Anywhere [Video]. YouTube. <https://www.youtube.com/watch?v=6hpIjx8d15s&t=491s>

Fahad, E. (2022, June 24). *MH-Z19B NDIR CO2 Sensor with Arduino*. Electronic Clinic. <https://www.electroniclinic.com/mh-z19b-ndir-co2-sensor-with-arduino-mhz19b/>

Fernandez, E. (2014, February 20). Hydroponic Nutrients TDS, PPMs and EC Explained! [Video]. YouTube. <https://www.youtube.com/watch?v=uI9D-ONNdHg>

freeCodeCamp.org. (2021, June 8). Arduino Course for Beginners – Open-Source Electronics Platform [Video]. YouTube. <https://www.youtube.com/watch?v=zJ-LqeX_fLU>

Google. (2023). *Google Cloud Documentation*. Google Cloud. <https://cloud.google.com/docs>

*HTML Introduction*. (n.d.). W3Schools. Retrieved April 17, 2023, from <https://www.w3schools.com/html/html_intro.asp>

MariaDB. (2023). *MariaDB Server: The open-source relational database*. MariaDB. <https://mariadb.org/>

Maxim Integrated. (2019). *DS18B20 – Programmable Resolution 1-Wire Digital Thermometer*. Maxim Integrated. <https://www.analog.com/media/en/technical-documentation/data-sheets/ds18b20.pdf>

Mouser Electronics. (n.d.). *DHT11 Humidity & Temperature Sensor*. Mouser Electronics. Retrieved April 16, 2023, from <https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf>

PHP. (2023). *History of PHP and Related Projects*. PHP. <https://www.php.net/manual/en/history.php>

Santos, R. (2016, August 24). *Guide for DS18B20 Temperature Sensor with Arduino*. Random Nerd Tutorials. <https://randomnerdtutorials.com/guide-for-ds18b20-temperature-sensor-with-arduino/>

Shinde, S. (2023, January 25). *What are the Key Pros and Cons of the Arduino Programming Language?* Emeritus. <https://emeritus.org/blog/coding-arduino-programming-language/>

Winsen. (2016). *Intelligent Infrared CO2 Module (Model: MH-Z19B)*. Zhengzhou Winsen Electronics Technology Co., Ltd. <https://www.winsen-sensor.com/d/files/infrared-gas-sensor/mh-z19b-co2-ver1_0.pdf>

XAMPP Team. (2023). *Windows Frequently Asked Questions*. Apache Friends. <https://www.apachefriends.org/faq_windows.html>

<asd>