A white paper on the

Wardian Case Terrarium a Technological Update

**Author: Joel Berg, Joshua Howard, Lexus Lineman, Spencer Peck**

**Date: April 25, 2023**

**Version 1.0**

**Table of Content**

**Abstract.......................................................................................................................................... 3**

**Introduction................................................................................................................................... 3**

**Problem Statement........................................................................................................................ 4**

**Solution.......................................................................................................................................... 6**

**Materials and Methods................................................................................................................. 7**

**Explanation of the Code…......................................................................................................... 10**

**Conclusion…............................................................................................................................... 12**

**Recommendations ...................................................................................................................... 12**

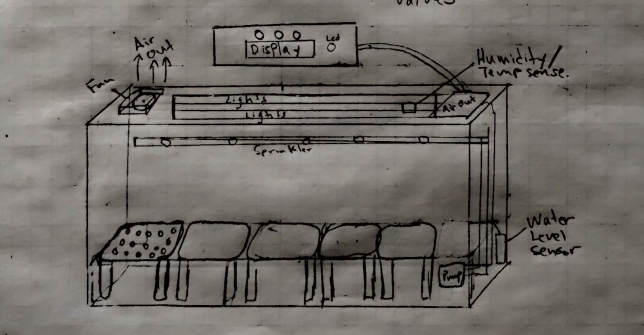
**Code ............................................................................................................................................. 14**

**Work Cited.................................................................................................................................. 23**

# Abstract

In 1829, Dr. Nathaniel Bagshaw Ward invented the Wardian case, a device designed to transport live plants around the world. Although the invention revolutionized the way plants were transported, it has become outdated due to modern transportation methods. However, the concept of the Wardian case can still be relevant in today's society, where many people lack the space, don't have backyards, live in small apartments, and are unable to have gardens. We propose creating a modern version of the Wardian case that is portable, and suitable for use in apartments and homes. Our design will incorporate a glass fish tank and technology to create the optimal environment for plant growth. The case will have sensors to measure humidity, temperature, and automatically water the plants. It will also provide sufficient light for plant growth. By creating this modern Wardian case, we aim to provide people with limited space the opportunity to experience the benefits of indoor gardening.

# Introduction

The purpose of Doctor Wards’ Invention was to be able to transport live plants all over the world. It was created using wood and glass. It was often in the shape of a house and was used as a portable greenhouse. It was a life-changing invention and is responsible for many ecosystems changing around the world. There are many people out there wanting to plant a garden but have no place to properly take care of a plant. In the past, people wanted to transport plants all over the world and share their discoveries. The second solution was solved by an invention called the Wardian Case. We plan on using the foundation of the Wardian Case Terrarium as a way to solve problem one. With our understanding of plants, we know that each plant's needs are different, meaning that we will need to create our Wardian Case to be more adaptable than the original.  
 Our plan will use a microcontroller known as an Arduino Uno to control humidity, a watering schedule, and a water pump. The microcontroller will also be connected to an LED display that will display conditions such as temperature, humidity, and what the watering cycle is set to. The user will be able to set a desired humidity allowance and watering schedule to keep the plants inside the case alive. The case will also include an LED plant light that will produce the necessary lighting requirements so this case can be placed practically anywhere. The case should be placed at room temperature indoors. Our goal is to make an inexpensive and simple control system with 3D printed hardware that can be joined to any standard glass aquarium.

# Problem Statement

Our project aims to solve common problems which many people face when attempting to grow houseplants and keep them alive. Some of those problems could be, but are not limited to, homes that do not have good natural lighting, exotic plants which require specific climate conditions, and maintaining the watering schedule required by the plants in question.

Natural lighting can be a problem for a variety of reasons. While houses usually have a yard or garden in which people may plant, people may also have the desire to grow plants inside, and if the house does not have many windows, specifically south-facing windows it becomes difficult to give the plants the light they need. Apartments/condos tend to have even more problems with natural light as they don’t have yards or gardens, and sometimes don’t even have a balcony on which they may grow their plants, meaning they have no other option than to grow their plants inside, and if they also don’t have many windows or at least a few south-facing windows, growing houseplants can become difficult.

Exotic plants and their picky climate conditions tend to bring a new set of problems, as they often do not adapt well and require the same conditions as their natural environment to survive. If you don’t live in their natural environment, maintaining the correct humidity levels, watering cycle, and light levels needed to mimic that environment can require a lot of work. Getting a small greenhouse can aid in growing these plants but that still brings us to our last problem, the watering cycle.

Even if people have ideal conditions to grow their plants, or they have a small greenhouse box of some kind to help obtain the needed growing conditions, people often struggle to maintain a proper watering schedule. We all have busy schedules, whether that is due to work, school, homework, family issues, dating life, or a combination of them all. People often forget to water their plants on time, which can leave the plants dry. Sometimes the opposite may occur as well. Out of fear, they are not watering their plant enough, perhaps because the plant has dried out, they begin to overwater the plant in hopes of saving them. This however can also lead to killing the plant.

Our project aims to resolve all these issues, so maintaining a plant may be possible regardless of where you live, what plant you hope to grow, and what watering schedule that plant may require.

# Solution

The frame of the appliance will take the shape of a Warden Case. These cases will allow for the humidity of the atmosphere the plant will grow in to match the climate where it would naturally occur. By attaching a light to the bottom of the lid that is controlled by a clock, the day and night periods of the plant’s natural habitat can be mimicked, ensuring a stable amount of energy to the plant, removing the need for a window or balcony to provide sunlight.

By providing a basin of water within the case the humidity will steadily rise as the air interacts with the basin. Using a sensor we can choose a humidity level where a fan should activate in order to move the humid air out and dry air in. The basin can then be used to water the plant. Employing a small pump that connects from the floor of the basin to a drip line will enable a powered watering system. By linking this pump to an internal timer, a consistent schedule can be set, avoiding the worry that users would normally feel about remembering to water the plant. With the basin both providing humidity and a supply to water the plants from, refills will be needed from the user.

# Materials and Methods



# 

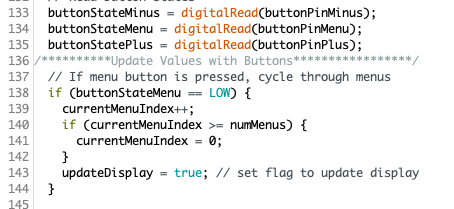
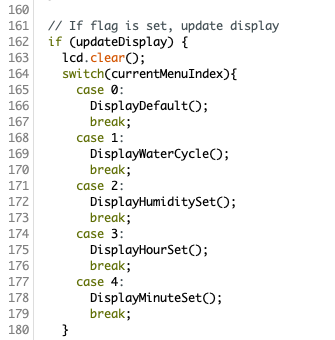
## Components and Parts

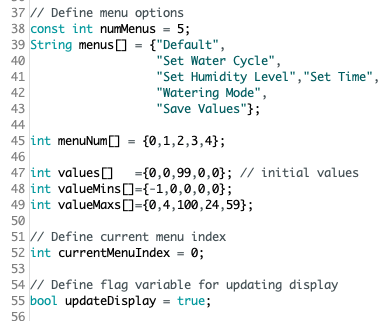
* 1 - I2C 1602 LCD Display Module 16X2 [Link](https://www.amazon.com/dp/B07S7PJYM6?psc=1&ref=ppx_yo2ov_dt_b_product_details)
* 1 Arduino Uno Microcontroller
* 1 Standard breadboard
* 1 Real-Time Clock Module [Link](https://www.amazon.com/AOICRIE-Precision-AT24C32-Arduino-Raspberry/dp/B09STHQ4TL/ref=sr_1_11?crid=E9PTUHUJZJ5Q&keywords=ds3231%2Brtc%2Bmodule%2Bfor%2Barduino&qid=1681059362&s=electronics&sprefix=RTC%2Bmodule%2Celectronics%2C132&sr=1-11&th=1)
* Jumper Wires
* 1 IRF520 Power MOSFET Driver Module [Link](https://www.amazon.com/Acxico-IRF520-MOSFET-Arduino-Raspberry/dp/B081YS3G58/ref=sr_1_6?keywords=mosfet+module&qid=1681058966&s=electronics&sr=1-6)
* 3 6x6x4.3mm TACT Switch Push Button
* 1 DC 3V 5V Micro Submersible Mini Water Pump [Link](https://www.amazon.com/Gikfun-Submersible-Fountain-Aquarium-EK1893/dp/B07BHD6KXS/ref=sr_1_4_sspa?crid=1WSXOSYL0PHJ3&keywords=arduino+water+pump&qid=1681059094&sprefix=arduino+water+pum%2Caps%2C152&sr=8-4-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUFWWkVUNko5MzZQNUUmZW5jcnlwdGVkSWQ9QTAwMzM4MTIxS0o5NDQ4NFo1QkVPJmVuY3J5cHRlZEFkSWQ9QTAxODYwNDgzMFIxUVJDQ041QkNHJndpZGdldE5hbWU9c3BfYXRmJmFjdGlvbj1jbGlja1JlZGlyZWN0JmRvTm90TG9nQ2xpY2s9dHJ1ZQ==)
* 1 DHT11 Temperature Humidity Sensor Module Digital Temperature Humidity [Link](https://www.amazon.com/dp/B01DKC2GQ0/ref=sspa_dk_detail_1?psc=1&pd_rd_i=B01DKC2GQ0&pd_rd_w=X6OxV&content-id=amzn1.sym.a8bfa847-39f1-4f54-bb95-ee052e9f7910&pf_rd_p=a8bfa847-39f1-4f54-bb95-ee052e9f7910&pf_rd_r=PWRXFTNM2SF6WFWKTSBQ&pd_rd_wg=XJEAe&pd_rd_r=3e7e0e92-2115-4391-a38f-525c6f1111ce&s=appliances&sp_csd=d2lkZ2V0TmFtZT1zcF9kZXRhaWxfdGhlbWF0aWM&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUExODhQUTZMQVNNWEIyJmVuY3J5cHRlZElkPUEwMzQyMTQwM1VBNEVPRksxWlhWQSZlbmNyeXB0ZWRBZElkPUEwNzc2MDY1MTcxNUpDUVdEMVZNTyZ3aWRnZXROYW1lPXNwX2RldGFpbF90aGVtYXRpYyZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU=)
* 1 Prototype PCB Solderable Breadboard for Electronics 88.9mm X 52.1mm [Link](https://www.amazon.com/dp/B07ZYNWJ1S?ref=ppx_yo2ov_dt_b_product_details&th=1)
* 1 DC 50mm Fan 5V [Link](https://www.amazon.com/dp/B08R9KN46J?ref=ppx_yo2ov_dt_b_product_details&th=1)
* Soldering iron and solder (optional)

**3D Printed Components**

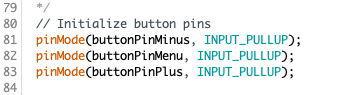
# 

# Explanation of the Code

The menu is displayed on a standard 16x2 LCD display. There are three buttons, a minus (“-”), plus (“+”), and a menu button. When you press the “menu” button it cycles through different menus where the user can press the plus or minus button to increase or decrease values. In the “loop” on the arduino IDE, you can have the program clear and print on the LCD display with some kind of delay. This constant updating causes many issues and electrical interferences which make the display flicker and possibly lose its data stream connection. To solve this problem we created a flag variable called “updateDisplay”. Which was set to true each time a button is pressed. When it’s set to true, the program will cycle through the loop and go through a switch statement that will print the display based on what the currentMenuIndex value is. Because we made this change in the code, the display does not constantly clear and update the display and instead only does one time when one of the three buttons are pressed. 

All of the menu displays hold a single value that is stored in an array called value[]. This value[] is incremented or decremented in the main loop if the plus or minus button is pressed. The values are then printed within one of the methods like DisplayDefault(), DisplayWaterCycle(), or DisplayHumiditySet(). Overall, the purpose of the values[] array is to provide a convenient way to store and manipulate a set of related values in the program.

The code initializes the three button pins used in the program, which are buttonPinMinus, buttonPinMenu, and buttonPinPlus, as input pins with internal pull-up resistors using the pinMode() function.

An input pin is used to read the state of a button or a sensor. A pull-up resistor is used to ensure that the input pin is pulled high to the voltage level of the microcontroller when the button is not pressed, preventing the pin from floating or picking up random noise. When the button is pressed, the input pin is connected to ground, so the voltage level at the input pin becomes low.

The INPUT\_PULLUP argument passed to the pinMode() function allows the internal pull-up resistor for the corresponding pin. By using the internal pull-up resistor, we don't need to connect an external resistor to the button pins.

# 

# Conclusion

Based on our research and experimentation, we can conclude that it is indeed possible to create a modern Wardian Case that is accessible and practical for everyday people. With the addition of specialized accessories, such as lighting and humidity control systems, we have created a Wardian Case that can support a variety of plants in a range of indoor environments.

We believe that our Wardian Case design will not only provide an opportunity for people without access to a backyard or garden to grow plants but also encourage more people to incorporate plants into their homes. Plants not only add beauty and a sense of tranquility to indoor spaces but also improve air quality and provide numerous health benefits. We are excited about the potential of our modern Wardian Case design and hope that it will inspire more people to incorporate plants into their daily lives.

# Recommendations

After creating our modern Wardian Case, we have realized that there are many ways to improve functionality. For those interested in recreating this project, we would highly recommend adding additional accessories to make it more user-friendly and provide a better overall experience.

One accessory we suggest adding is a Grove moisture sensor. This will allow users to keep track of the soil moisture levels of each individual plant, as each plant requires its own specific level of moisture. This will replace the water timer so the plant will be watered only when the soil indicates that it needs it.

Another recommendation would be to upgrade the lighting system to include the ability to adjust the intensity of UV light. This will make the Wardian Case able to accommodate different types of plants that require different light levels.

We had originally planned on including doors which would lay flat over the exhaust holes, so while the fan was off the humidity could be trapped completely within the case, allowing it to get to 100% humidity. Then when the humidity increased passed the set level by the user, the exhaust fan would turn on, and a servo motor would open the doors allowing full air flow, and bring the humidity back down. However we found that even without the doors the humidity was able to get quite high, as high as 85%. Seeing as the humidity was able to get that high without the doors, and due to time constraints in completing the design, it was decided to not include them in the end. However even though the humidity maintenance system works well enough without the doors, we would suggest including the doors as it does allow for a greater range of humidity to be achieved.

Lastly, we suggest creating an app for the user to check the status of the plants and set specific requirements from their phone. This will make it easier for users to manage their plants and ensure they are getting the proper care they need, even when they’re not home. Overall, these recommendations will enhance the functionality of the modern Wardian Case and provide a more enjoyable experience for the user.

**Code**

#include <Wire.h>

/\*

\* Project: Wardian Case

\* Author: Joel Berg

\* Date 03/21/2023

\*/

**libraries**

#include <Wire.h>

#include <RTClib.h>

#include <LiquidCrystal\_I2C.h>

#include <EEPROM.h>

#include <dht11.h>

**LCD and RTC**

LiquidCrystal\_I2C lcd(0x27,16,2); // Display I2C 16 x 2

RTC\_DS1307 RTC;

**Hum and Temp sens**

#define DHT11PIN 2

dht11 DHT11;

**Fan** #define PWM 3 // connect hot wire of fan to pin 3.

**pump**

#define PUMP 11 // connect signal of relay to pin 11.

int pumpTime =5000; // time that the pump is on.

**Define Button Pins**

// Define button pins

const int buttonPinMinus = 6;

const int buttonPinMenu = 7;

const int buttonPinPlus = 8;

// Define button states

int buttonStateMinus = 0;

int buttonStateMenu = 0;

int buttonStatePlus = 0;

// Define menu options

const int numMenus = 5;

String menus[] = {"Default",

"Set Water Cycle",

"Set Humidity Level","Set Time",

"Watering Mode",

"Save Values"};

int menuNum[] = {0,1,2,3,4};

int values[] ={0,0,99,0,0}; // initial values

int valueMins[]={-1,0,0,0,0};

int valueMaxs[]={0,4,100,24,59};

// Define current menu index

int currentMenuIndex = 0;

// Define flag variable for updating display

bool updateDisplay = true;

**Variables**

int waterCycle = 0; // time cycle set by user to define when to water.

int humidity = 99; // Humidity level set by user

int currentMinute;

void setup() {

// Initialize LCD

lcd.begin();

/\* the 'INPUT\_PULLUP' argument passed to the 'pinMode()' function

enables the interal pull-up resistor for the corresponding pin. By using

the internal pull-up resistor, we don't need to connect an external

resistor to the buttons pin.

An input pin is used to read the state of a button or a sensor.

A pull-up resistor is used to ensure that the input pin is pulled

high to the voltage level of the microcontroller when the button

is not pressed, preventing the pin from floating or picking up

random noise. When the button is pressed, the input pin is connected

to ground, so the voltage level at the input pin becomes low.

\*/

// Initialize button pins

pinMode(buttonPinMinus, INPUT\_PULLUP);

pinMode(buttonPinMenu, INPUT\_PULLUP);

pinMode(buttonPinPlus, INPUT\_PULLUP);

pinMode(2, OUTPUT);//Pin for the humidity/temp sensor

//pinMode(fanPin, OUTPUT); // PIN FOR FAN

pinMode(11, OUTPUT); // pin for water pump

digitalWrite(11, LOW); //initialize to low just in case.

Serial.begin(9600); ///YOU NEED THIS!!!

Wire.begin();

RTC.begin();

DateTime now = RTC.now();

values[3] = now.hour();

values[4] = now.minute();

if (! RTC.isrunning()) {

Serial.println("RTC is NOT running!");

// Set the date and time at compile time

RTC.adjust(DateTime(\_\_DATE\_\_, \_\_TIME\_\_));

}

currentMinute=now.minute();

**Welcome Message**

lcd.clear();

lcd.setCursor(0,0);

lcd.print(" Hello! :)");

Serial.print("Program Started \n");

delay (1500);

}

void loop() {

DateTime now = RTC.now();

if(currentMinute < now.minute()){

updateDisplay=true;

currentMinute = now.minute();

}

if(values[0]==-1){

startPump();

values[0]=0;

delay(2000);

}

startPumpCycle(values[1]);

if (DHT11.humidity > values[2]) {

analogWrite(PWM,150);

} else {

analogWrite(PWM,0);

}

DHT11.read(DHT11PIN);

// Read button states

buttonStateMinus = digitalRead(buttonPinMinus);

buttonStateMenu = digitalRead(buttonPinMenu);

buttonStatePlus = digitalRead(buttonPinPlus);

**Update Values with Buttons**

// If menu button is pressed, cycle through menus

if (buttonStateMenu == LOW) {

currentMenuIndex++;

if (currentMenuIndex >= numMenus) {

currentMenuIndex = 0;

}

updateDisplay = true; // set flag to update display

}

if(buttonStateMinus == LOW){

values[currentMenuIndex]--;

if(values[currentMenuIndex] < valueMins[currentMenuIndex]){

values[currentMenuIndex] = valueMaxs[currentMenuIndex];

}

updateDisplay=true;

}

if(buttonStatePlus==LOW){

values[currentMenuIndex]++;

if(values[currentMenuIndex] > valueMaxs[currentMenuIndex]){

values[currentMenuIndex] = valueMins[currentMenuIndex];

}

updateDisplay=true;

}

// If flag is set, update display

if (updateDisplay) {

lcd.clear();

switch(currentMenuIndex){

case 0:

DisplayDefault();

break;

case 1:

DisplayWaterCycle();

break;

case 2:

DisplayHumiditySet();

break;

case 3:

DisplayHourSet();

break;

case 4:

DisplayMinuteSet();

break;

}

/\*lcd.print(menus[currentMenuIndex]);

lcd.setCursor(0, 1);

lcd.print(values[currentMenuIndex]);\*/

updateDisplay = false; // reset flag

}

delay(100);

}

**Functions**

void DisplayDefault(){

DateTime now = RTC.now();

lcd.setCursor(0, 0);

lcd.print("Temp:");

lcd.print(DHT11.temperature);

lcd.print("C");

lcd.print(" Hum:");

lcd.print(DHT11.humidity);

lcd.print("%");

lcd.setCursor(0, 1);

lcd.print(now.hour());

lcd.print(":");

if(now.minute() < 10){

lcd.print("0");

}

lcd.print(now.minute());

lcd.print(" Hum:");

lcd.print(values[2]);

lcd.print("%");

}

void DisplayWaterCycle(){

lcd.setCursor(0, 0);

lcd.print("Water ");

lcd.setCursor(0,1);

lcd.print(" ");

lcd.print(values[currentMenuIndex]);

lcd.print(" times/day");

}

void DisplayHumiditySet(){

lcd.setCursor(0,0);

lcd.print("Humidity:");

lcd.print(DHT11.humidity);

lcd.setCursor(0,1);

lcd.print("SetHumidLvl:");

lcd.print(values[currentMenuIndex]);

lcd.print("%");

}

void DisplayHourSet(){

DateTime now = RTC.now();

lcd.setCursor(0,0);

lcd.print("Time: ");

lcd.print(now.hour());

lcd.print(":");

if(now.minute() < 10){

lcd.print("0");

}

lcd.print(now.minute());

lcd.setCursor(0,1);

lcd.print("Set Time: ");

lcd.print(now.hour());

lcd.print(":");

if(now.minute() < 10){

lcd.print("0");

}

lcd.print(now.minute());

RTC.adjust(DateTime(0,0,0,values[currentMenuIndex],now.minute(),0));

}

void DisplayMinuteSet(){

DateTime now = RTC.now();

lcd.setCursor(0,0);

lcd.print("Time: ");

lcd.print(now.hour());

lcd.print(":");

if(now.minute() < 10){

lcd.print("0");

}

lcd.print(now.minute());

lcd.setCursor(0,1);

lcd.print("Set Time: ");

lcd.print(now.hour());

lcd.print(":");

if(now.minute() < 10){

lcd.print("0");

}

lcd.print(now.minute());

//lcd.print(values[currentMenuIndex]);

RTC.adjust(DateTime(0,0,0, now.hour(), values[currentMenuIndex],0));

}

void startPumpCycle(int waterCycle) {

DateTime now = RTC.now();

bool start\_pump = false;

switch (waterCycle) {

case 1:

start\_pump = (now.hour() == 12 && now.minute() == 0 && now.second() == 0);

break;

case 2:

start\_pump = (now.hour() == 12 && now.minute() == 0 && now.second() == 0) ||

(now.hour() == 24 && now.minute() == 0 && now.second() == 0);

break;

case 3:

start\_pump = (now.hour() == 8 && now.minute() == 0 && now.second() == 0) ||

(now.hour() == 16 && now.minute() == 0 && now.second() == 0) ||

(now.hour() == 24 && now.minute() == 0 && now.second() == 0);

break;

case 4:

start\_pump = (now.hour() % 6 == 0 && now.minute() == 0 && now.second() == 0);

break;

default:

// Invalid water cycle setting

return;

}

if (start\_pump) {

startPump();

}

}

void startPump(){

digitalWrite(PUMP, HIGH);

Serial.print("Pump is on! \n");

delay(pumpTime);

digitalWrite(PUMP,LOW);

}

**References**

Keogh, L. (n.d.). *The Wardian case: How a simple box moved the plant kingdom*. The Wardian Case: How a Simple Box Moved the Plant Kingdom | Arnold Arboretum. Retrieved April 18, 2023, from https://arboretum.harvard.edu/stories/the-wardian-case-how-a-simple-box-moved-the-plant-kingdom/

*Read/write serial EEPROM via I2C*. Read/Write Serial EEPROM via I2C. (n.d.). Retrieved April 18, 2023, from https://fritzing.org/projects/readwrite-serial-eeprom-via-i2c

*The wardian case*. Historic UK. (n.d.). Retrieved April 18, 2023, from https://www.historic-uk.com/CultureUK/Wardian-Case/