Automated Gardening System (AGS)

EE-CpE 4097

Dr. Robert Woodley

Advisors: Dr. Ian Ferguson & Dr. Zhou

AGS

Mohamed Mohamed	Nicole Voss	Alan Paaren	David Szatkowski	Samuel Ogunmolawa
xxxxxx@mst.edu	xxxxxx@mst.edu	xxxxxx@mst.edu	xxxxxx@mst.edu	xxxxxx@mst.edu
		Team Leader		

MISSOURI S&T

MINERS DIG DEEPER

OUTLINE

- Overview
  - Goals
  - Motivation
  - Value
- Progress
  - Entire System
  - Sound Frequency System
  - Ventilation System
  - Lighting System
  - Watering System
  - Nutrient Dosing System
  - User Interface
- Results
  - Plant Visuals
  - Graphs
- Lessons Learned
- References

OVERVIEW

AGS

MISSOURI S&T

MINERS DIG DEEPER

OVERVIEW - GOALS

- Automated care for indoor plants
- Make gardening more accessible
- Exceed basic features currently available
- Create a highly customizable system

OVERVIEW - MOTIVATION

- Increased demand for green-spaces

- Studies on stimulating plants through sound frequency
  - Increased plant health
  - Increased Growth
  - Quicker Crop yield

- No such product currently on the market

OVERVIEW - VALUE

- Provides an ideal growing environment for various species of plants
- Reduces upkeep and improves growth
- Modular design
- Relatively inexpensive

PROGRESS

AGS

MISSOURI S&T

MINERS DIG DEEPER

PROGRESS – ENTIRE SYSTEM

The Automated Garden consists of:

- Arduino Mega
  - Handles sound ventilation, watering, and the clock/UI
- Sound Frequency System
- Ventilation System
- Lighting System
  - Separate system
- Watering System
- Nutrient Dosing System
  - Integrated with watering system
- Clock Display/UI

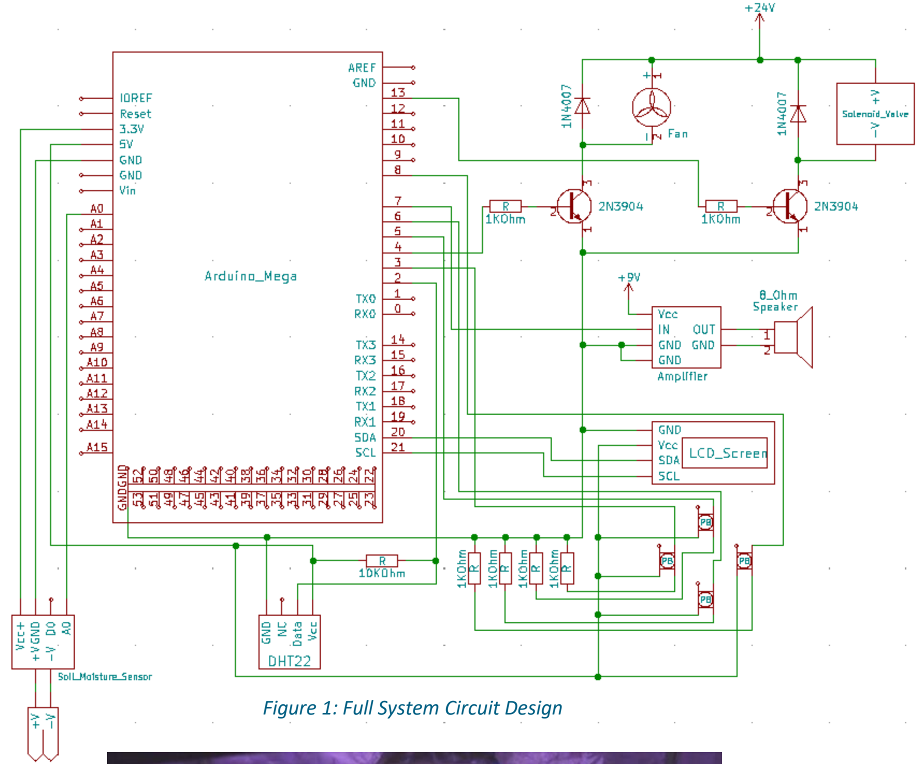

Figure 1: Full System Circuit Design

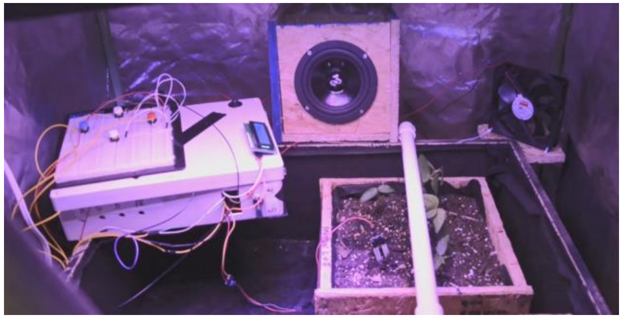

Figure 2: Full System - From Left to Right, UI/Logic, Frequency, Watering, Ventilation

PROGRESS – ENTIRE SYSTEM CODE

```
void setup() {
 lcd.init();
 lcd.backlight();
 setTime(16,00,0,10,30,19);//set time for 4pm October 30th, 2019
 Alarm.alarmRepeat(17,0,05,FreqOn);
 Alarm.alarmRepeat(17,0,35,FreqOff); // 4:00pm every day
 Alarm.alarmRepeat(17,0,10,WaterOn); // 10:00am every day
 Alarm.timerRepeat(3600, FanOn);    //Repeats every hour 3600 secs
 Alarm.timerRepeat(60, FanOff); //Checks fan if X minutes have passed
 Alarm.timerRepeat(60, PrintLoop); //Prints every min
 pinMode(fan, OUTPUT);
 pinMode(13, OUTPUT);
 pinMode(spkr, OUTPUT);
 pinMode(rightButton, INPUT);
 pinMode(leftButton, INPUT);
 pinMode(upButton, INPUT);
 pinMode(downButton, INPUT);
 Serial.begin(9600);
 dht.begin();
}

void loop() {
 ButtonCheck();
 Alarm.delay(100);

}
```

Figure 3: Entire system code

- We use several libraries
  - Handle DHT sensor, RTC setup, Alarm setup, LCD screen
- Initialize a lot of variables, and define several pins before and during void setup()
  - setTime(16,0,0,10,30,19) sets the clock to 4PM Oct 30, 2019
- Once void setup() has run once, void loop() repeatedly calls
  - ButtonCheck()
    - Looks for button inputs every .1 sec
  - Alarm.delay()
    - Constantly checks if an alarm has expired

PROGRESS – SOUND FREQUENCY SYSTEM

Figure 4: Speaker close-up

- Automation
  - Adjustable frequency
  - Varies in decibels
  - Daily timer with adjustable start and end times
- Benefits
  - Reduced germination time
  - Reduce in crop growth time
  - User set frequency & noise level (80-110bB)

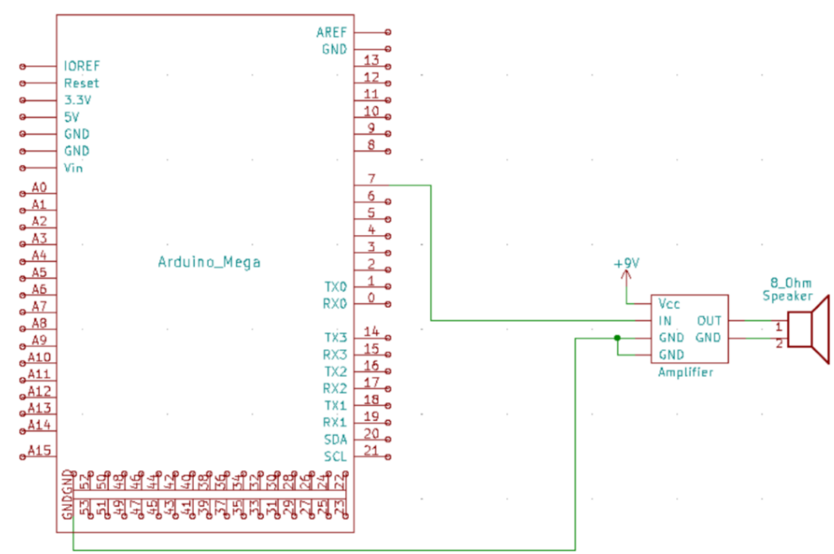

Figure 5: Amplifier/Speaker circuit

PROGRESS – SOUND FREQUENCY SYSTEM CODE

```
const int freq = 2700;
setTime(16,0,0,10,30,19);//set time for 4pm October 30th, 2019
 Alarm.alarmRepeat(13,0,0,FreqOn);
 Alarm.alarmRepeat(13,0,30,FreqOff);
 pinMode(spkr, OUTPUT);

//Timer for speaker
void FreqOn() {
  // write here the task to perform every morning
  Serial.println("Sound on");

  tone(spkr, freq);
}

void FreqOff() {
  //write here the task to perform every evening
  Serial.println("Sound off");
  noTone(spkr);
}
```

Figure 6: Frequency code

- In setup, two daily alarms are set to call functions that start and stop toggling the speaker
  - FreqOn() / FreqOff()
- tone(spkr, freq) toggles the spkr pin on/off at a rate of 2700 Hz until noTone(spkr) is called
  - Adjustable via the freq variable
- During testing, we stimulated plants from 1PM to 5PM
- This example runs for 30 sec after 1PM

PROGRESS – VENTILATION SYSTEM

Figure 7: Circulation Fan Close-Up

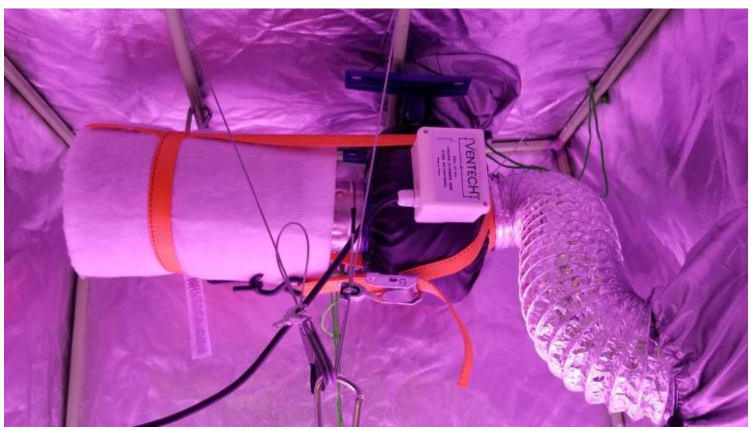

Figure 8: Carbon air filter and pump

- Provides proper air ventilation
- Maintains optimal temperature
- Prevents mold from accumulating
- Reduces heat impact
- Helps develop stronger stems

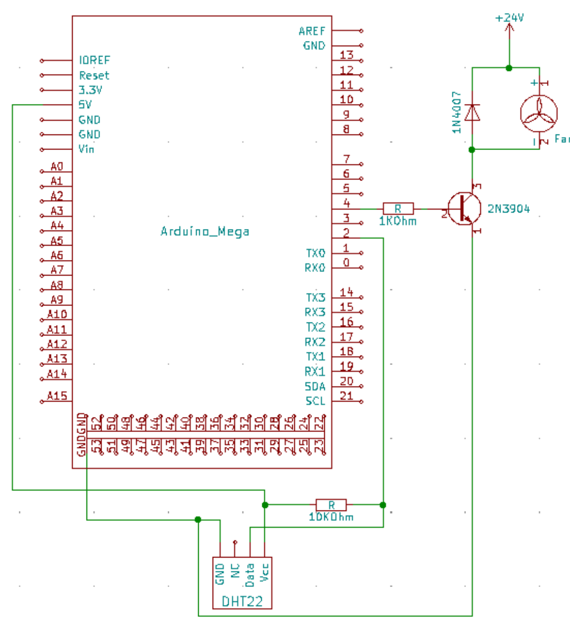

Figure 9: Fan/DHT sensor circuit

Figure 10: DHT22 Temperature-Humidity sensor

PROGRESS – VENTILATION SYSTEM CODE

```
#include <TimeAlarms.h>
#include <Time.h>
#include <TimeLib.h>
#include <DHT.h>
#define DHTPIN 2     // what pin we're connected to
#define DHTTYPE DHT22   // DHT 22  (AM2302)
#define fan 4
int fanCount = 11;
int maxHum = 60;
int maxTemp = 74;
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  setTime(16,0,0,11,21,19);//set time for 4pm October 30th, 2019
  Alarm.timerRepeat(3600, FanOn);    //Repeats every hour 3600 secs
  Alarm.timerRepeat(60, FanOff); //Checks every minute
  Alarm.timerRepeat(60, PrintLoop);
  pinMode(fan, OUTPUT);
  Serial.begin(9600);
  dht.begin();
}
```

Figure 11: Ventilation system code

- The ventilation system uses a repeating timer to handle the fan
- FanOn is called every hour and turns the fan on
- FanOff is called every minute and 10 minutes after FanOn was called turns the fan off
- PrintLoop() checks sensors, prints troubleshooting info to the serial monitor, and turns the fan on if the DHT sensor reads too hot or humid

PROGRESS – VENTILATION SYSTEM CODE

```
void PrintLoop()
{
 sensorValue = analogRead(sensorPin);
 Serial.println("Analog Value : ");
 Serial.println(sensorValue);
 float h = dht.readHumidity();
 // Read temperature as Celsius
 float t = dht.readTemperature();
 float TempF = (t*1.8)+32;
 // Check if any reads failed and exit early (to try again).
 if (isnan(h) || isnan(t)) {
  Serial.println("Failed to read from DHT sensor!");
  return;
 }
 if (h > maxHum || TempF > maxTemp) {
   digitalWrite(fan, HIGH);
   Serial.print("Turning on Fan it's hot! ");
 } else if(fanCount > 10) {
    digitalWrite(fan, LOW);
 }
 Serial.print("Day count: ");
 Serial.print(daycount);
 Serial.print(" \t");
 Serial.print("Humidity: ");
 Serial.print(h);
 Serial.print(" %\t");
 Serial.print("Temperature: ");
 Serial.print(TempF);
 Serial.println(" *F ");
}
```

Figure 12: Ventilation system code

- The code checks values from the DHT22 temperature-humidity sensor.
- Activates fan if the sensor readings exceed maxTemp or maxHum
- Temperature was set at 74˚ F
- Humidity was set at 60%
- Fan turns off once readings are back below those values.

PROGRESS – VENTILATION SYSTEM CODE

```
void FanOn() {
  float h = dht.readHumidity();
  // Read temperature as Celsius
  float t = dht.readTemperature();
  float TempF = (t*1.8)+32;
  Serial.print("Turning on Fan");
  digitalWrite(fan, HIGH);

  fanCount = 0;
}

void FanOff(){
  if(fanCount == 10){
    digitalWrite(fan, LOW);
    Serial.print("Fan Turned Off");
  }
  if(fanCount <= 11)
  {
    Serial.print("Fan on for ");
    Serial.print(fanCount);
    Serial.print("minutes.");
    fanCount++;
  }
```

Figure 13: FanOn() and FanOff() code

- FanOn()
  - Turns on fan, starts a counter for FanOff()
- FanOff()
  - Once counter reaches 10 min, turns off fan

PROGRESS – LIGHTING SYSTEM

Figure 14: LED Array Close-Up

- Full-Spectrum LEDs w/ spectrum control provide an optimum wavelength of light throughout the entire growth cycle
- System operates independently using values provided through an IR remote
- Programmable controls allow fully-customizable settings
- Daisy-chain feature provides simplicity and scalability

PROGRESS – WATERING SYSTEM

Figure 15: Watering plants close-up

Figure 16: Soil moisture sensor

- Controlled by soil moisture sensor value and timer
- Moisture values vary per plant
- Distributes water directly to plant

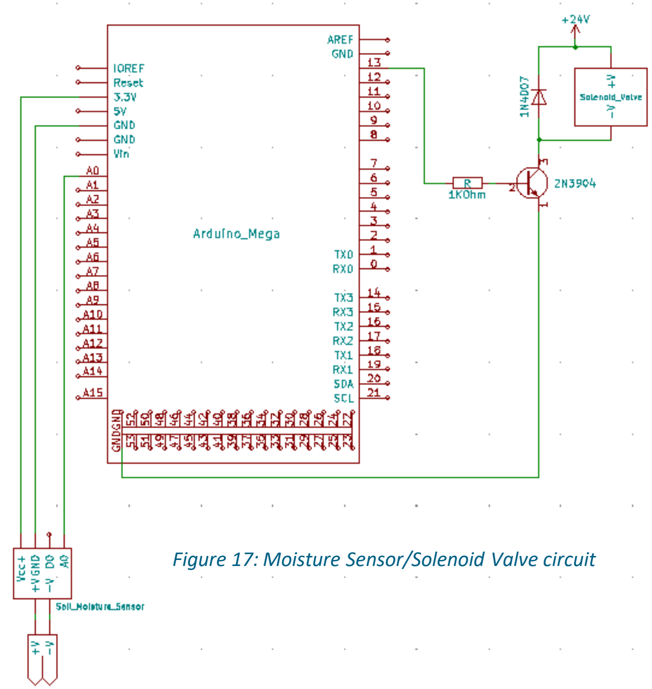

Figure 17: Moisture Sensor/Solenoid Valve circuit

PROGRESS – WATERING SYSTEM CODE

```
#include <TimeAlarms.h>
#include <Time.h>
#include <TimeLib.h>

int sensorPin = A0; //moisture sensor
int sensorValue; //moisture sensor
int limit = 600; //moisture sensor

void setup() {
  setTime(19,0,0,11,21,19);//set time for 4pm October 30th, 2019
  Alarm.alarmRepeat(10,0,0,WaterOn); //10:00am every day
  Serial.begin(9600);
}

//Timer for watering system
void WaterOn() {
  Serial.println("daycount: ");
  Serial.println(daycount);
  if (daycount== 1 or 4) {
   if (sensorValue>limit) {
     Serial.println("Alarm: - its Raining!");
     digitalWrite(13, HIGH);
     delay(30000);
     digitalWrite(13, LOW);
   }
   else {
     Serial.println("Plants have enough water");
   }
  }
  daycount=(daycount+1);
  if (daycount> 7){
  daycount=1;
  }
  Serial.println("daycount: ");
  Serial.println(daycount);
}
```

Figure 18: Soil moisture sensor code

- Reads values sent from the soil moisture sensor from the assigned analog pin.
- When the timer reaches a pre-set time, the timer calls WaterOn()
- WaterOn() uses a counter to check twice weekly if the soil moisture is below the desired value
- If the value is too low, the valve opens for 30 sec

PROGRESS – NUTRIENT DOSING SYSTEM

Figure 19: Liquid Nutrient Additives

- Dosage depends on the plant species
  - Optimal ratio for mung beans (1 - 0.5 - 1.59)
  - Ratio achieved was (1 - 0.5 - 1.7)
- Adjusted across each stage of the growing cycle
  - Recommended frequency is every other watering
- Uniformly distributed across the plot through the watering system

PROGRESS – USER INTERFACE

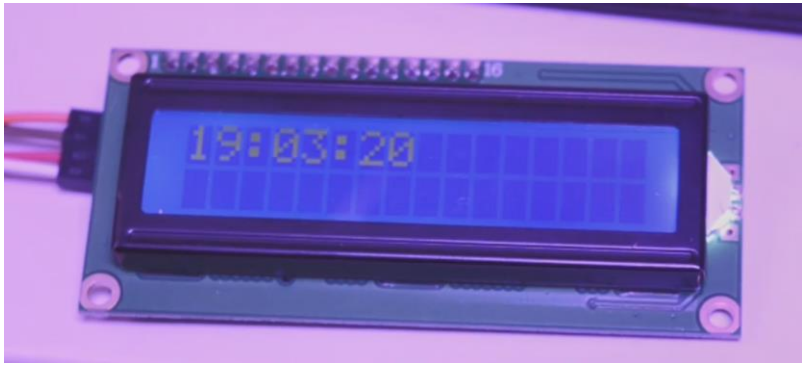

Figure 20: Clock LCD Display

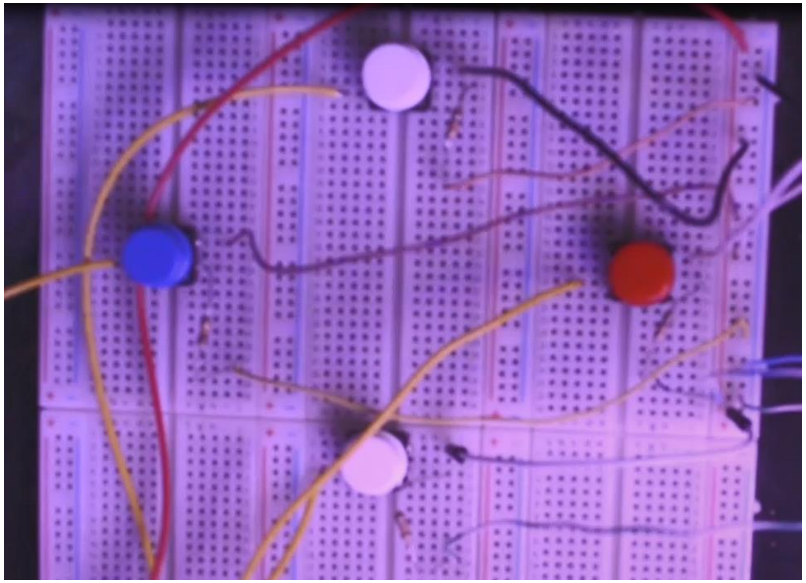

Figure 21: UI Button Array

- An LCD screen and button interface was installed to adjust the clock and easily troubleshoot system errors

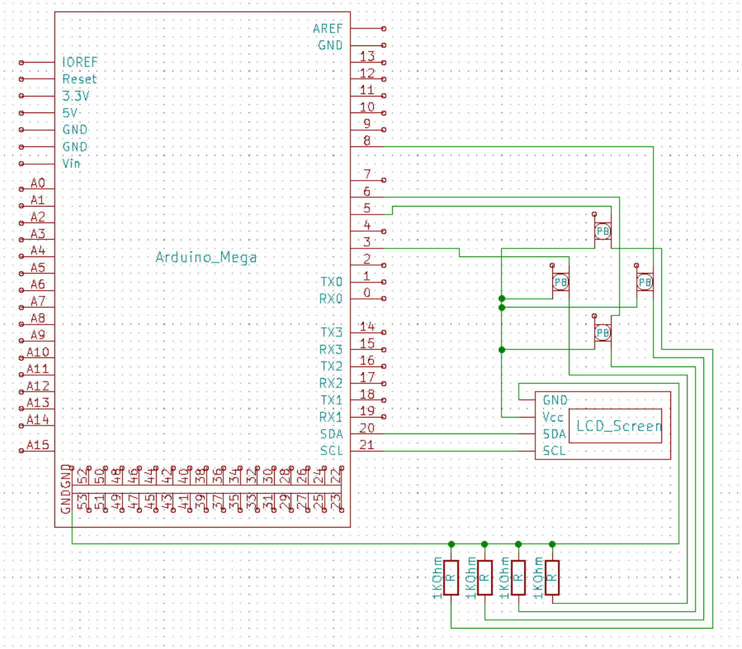

Figure 22: LCD/Button Array circuit

PROGRESS – USER INTERFACE

```
void ButtonCheck(){
 buttonStateRight = digitalRead(rightButton);
 buttonStateLeft = digitalRead(leftButton);
 buttonStateUp = digitalRead(upButton);
 buttonStateDown = digitalRead(downButton);
 if (buttonStateRight == HIGH) {
  RightButton();
 } else if (buttonStateRight == LOW) {
  right = 0;
 }
 if (buttonStateLeft == HIGH) {
  LeftButton();
 } else {
  left = 0;
 }
 if (buttonStateUp == HIGH) {
  UpButton();
 } else {
  up = 0;
 }
 if (buttonStateDown == HIGH) {
    DownButton();
 } else {
  down = 0;
 }
 digitalClockDisplay();
}
```

Figure 23: Checking for Button Presses

- ButtonCheck() is called every 100ms and checks for button presses
- If a button is pressed, it calls the related function
  - RightButton(), LeftButton(), etc.
- Also calls digitalClockDisplay() to print the clock to the LCD screen

PROGRESS – USER INTERFACE

- Functions scroll the cursor left and right or adjust the clock up or down

```
void RightButton(){
    if (right == 0) {
      if(clockCursor < 7){
        clockCursor++;
        if(clockCursor == 2 or clockCursor == 5)
          clockCursor++;
      }
      else
        clockCursor = -1;
    }
    right = 1;
}
void LeftButton(){
    if (left == 0) {
      if(clockCursor > -1){
        clockCursor--;
        if(clockCursor == 2 or clockCursor == 5)
          clockCursor--;
      }
      else
        clockCursor = 7;
    }
    left = 1;
}
```

Figure 24: Cursor Handling

```
void UpButton(){
    if (up == 0) {
      if(clockCursor == 0)
        adjustTime(36000);
      else if(clockCursor == 1)
        adjustTime(3600);
      else if(clockCursor == 3)
        adjustTime(600);
      else if(clockCursor == 4)
        adjustTime(60);
      else if(clockCursor == 6)
        adjustTime(10);
      else if(clockCursor == 7)
        adjustTime(1);
    }
    up = 1;
}
void DownButton(){
  if (down == 0) {
      if(clockCursor == 0)
        adjustTime(-36000);
      else if(clockCursor == 1)
        adjustTime(-3600);
      else if(clockCursor == 3)
        adjustTime(-600);
      else if(clockCursor == 4)
        adjustTime(-60);
      else if(clockCursor == 6)
        adjustTime(-10);
      else if(clockCursor == 7)
        adjustTime(-1);
    }
    down = 1;
```

Figure 25: Manually Setting the RTC

PROGRESS – USER INTERFACE

- Formats and prints current time to LCD

```
void digitalClockDisplay() {
  // digital clock display of the time
  Serial.print(hour());
  printDigits(minute());
  printDigits(second());
  lcd.setCursor(0,0);
  if(hour() < 10)
  {
    lcd.print(0);
    lcd.setCursor(1,0);
  }
  lcd.print(hour());
  printLCDmin(minute());
  printLCDsec(second());
  Serial.println();
  lcd.setCursor(0, 1);
  lcd.print("                ");
  if(clockCursor >= 0)
  {
    lcd.setCursor(clockCursor, 1);
    lcd.print('^');
  }
}
```

Figure 26: Printing Clock to LCD and Serial Monitor

```
void printDigits(int digits) {
  Serial.print(":");
  if (digits < 10)
    Serial.print('0');
  Serial.print(digits);
}
void printLCDmin(int digits) {
  lcd.setCursor(2,0);
  lcd.print(":");
  lcd.setCursor(3,0);
  if (digits < 10)
  {
    lcd.print('0');
    lcd.setCursor(4,0);
  }
  lcd.print(digits);
}
void printLCDsec(int digits) {
  lcd.setCursor(5,0);
  lcd.print(":");
  lcd.setCursor(6,0);
  if (digits < 10)
  {
    lcd.print('0');
    lcd.setCursor(7,0);
  }
  lcd.print(digits);
}
```

Figure 27: Formatting Printing

PROGRESS – SYSTEM DEMO

RESULTS

AGS

MISSOURI S&T

MINERS DIG DEEPER

RESULTS – PLANT VISUALS

The batch of tests contained 3 groups including:

- 4 hours of 2.7kHz frequency at ~80dB & 18 hours of full spectrum lighting
- No frequency stimulation & 18 hours of full spectrum lighting
- Indoor control group

Control Group

Figure 28: Control System Testing Results

Lighting Only

Figure 29: Lighting System Stress Testing Results

Lighting & Frequency

Figure 30: Lighting and frequency System Stress Testing Results

RESULTS – GRAPHS

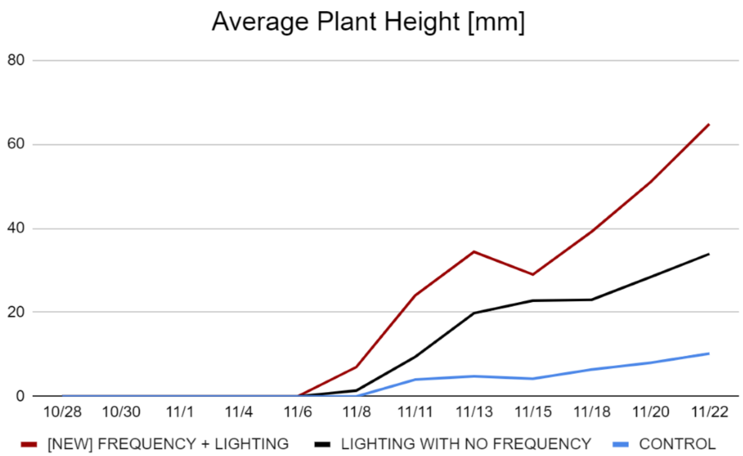

Figure 31: Plant Height Over Time

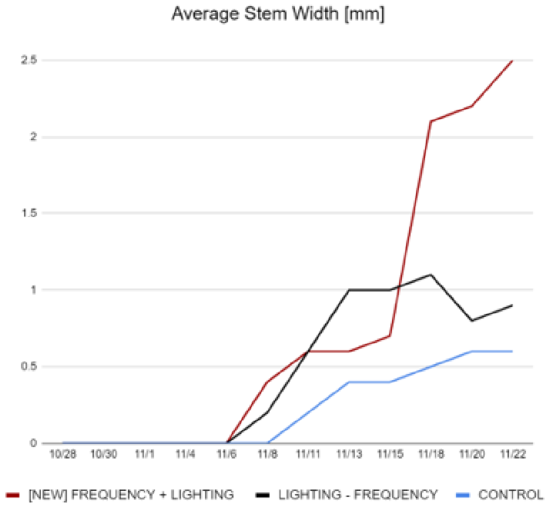

Figure 32: Stem Width Over Time

Mung beans that received 4 hours of 2.7kHz frequency stimulation and 18 hours of full spectrum lighting grew:

- 47.7% taller than the group that did not receive frequency stimulation
- 84.3% taller than the indoor control group

These mung beans had an average stem width that was:

- 64% ticker than the group that did not receive frequency stimulation
- 76% thicker than the indoor control group

RESULTS – GRAPHS

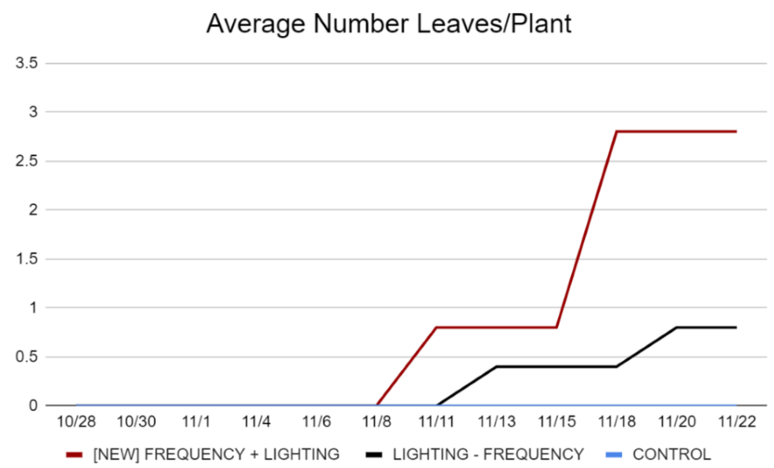

Figure 33: Average # Leaves Over Time

Mung beans that received 4 hours of 2.7kHz frequency stimulation and 18 hours of full spectrum lighting grew:

- 71.4% more leaves than the group that did not receive frequency stimulation
- 100% more leaves than the control group

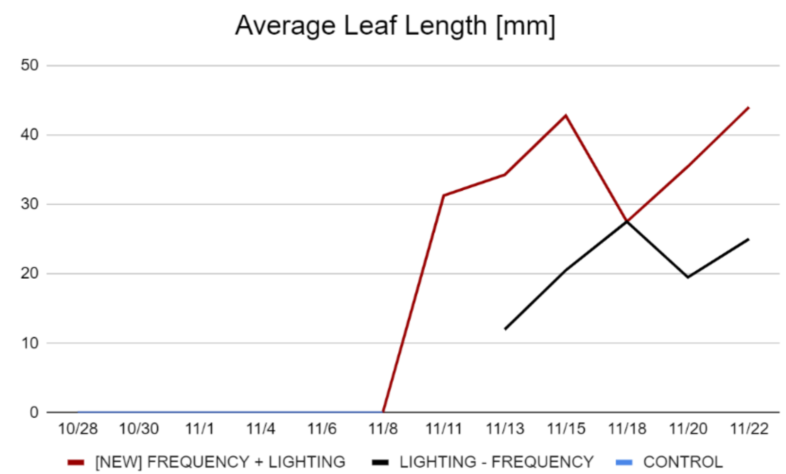

Figure 34: Average Leaf Length Over Time

These mung beans had average leaf length that was:

- 43% longer than the group that had only received the lighting treatment
- 100% longer than the control group

LESSONS LEARNED

AGS

MISSOURI S&T

MINERS DIG DEEPER

LESSONS LEARNED

- Plan ahead
  - Despite all the issues we weren't prepared for...
    - Components fried
    - Some parts were unable to handle the system stress required
    - Needed additional power source
- Enlist an expert
  - Most of our knowledge came from individual research of the systems
    - Our team was missing a background in the science of gardening
    - Bad gardening could lead to bad data
  - Get input from a horticulturalist or agriculturalist

REFERENCES

[1] J. McGrath, “Eight Devices to Help You Grow a Garden Indoors,” *Digital Trends*, Jan. 16, 2019. [Online]. Available: https://www.digitaltrends.com/home/indoor-garden-devices/. [Accessed: Feb. 10, 2019].

[2] W. Cai, H. He, S. Zhu, and N. Wang, “Biological Effect of Audible Sound Control on Mung Bean (Vigna radiate) Sprout,” *BioMed Research International*, vol. 2014, pp. 1–6, Aug. 2014

Automated Gardening System (AGS)

Thank you!

AGS

QUESTIONS?

MISSOURI S&T

MINERS DIG DEEPER