Nathan Phan - nphan23 - nphan23@uic.edu

Mohab Mustafa - mmustaf2 - mmustaf2@uic.edu

Ali S Baig - abaig28 - abaig28@uic.edu

EzPlant

Abstract:

We will be using multiple Arduinos to control a plant care system which monitors the plant's environmental conditions, soil moisture, and controls a light. Three Arduinos will communicate with each other and toggle the light based on time and send sensor readings and plant parameters, which will be stored in a master Arduino and will be set by the user. These sensors will measure humidity, temperature, and soil moisture. The environment conditions are used to create a game, where points are received if the plant is being properly cared for and points are subtracted when conditions are not met.

Description:

The project idea is to create an automated system that takes evaluates plant ownership as points. The system would automatically check the soil's moisture level, control lighting, and check humidity and temperature levels to ensure that the plant has the proper conditions to grow. The system would give a warning on an LCD screen to the user if proper conditions were not met. Lighting would be controlled on a timer. Every 10 minutes, the system evaluates the current conditions by comparing them to the inputted parameters and will add/subtract points based on whether the current conditions are acceptable. The system will work for multiple species of plants, as the system will be able to take input to change conditions and light timing.

Arduino Use Plan:

The plan is to use three Arduinos. One Arduino would be the master controller that takes input from another Arduino. Another Arduino would then be sent input from the master Arduino, which will tell it what operations to perform: turn the light on or off. To be concise, one Arduino will read from sensors and send this info to the master Arduino, which will then send that info to another Arduino to turn a light on or off. The master will also be used to allow users to set the light timer and plant conditions so that any plant may be used.

Multiple Arduinos Plan:

The Arduinos will work together to take care of the plant, where one Arduino controls, one analyzes, and one performs. The method of communication to be used between the Arduinos is the I2C serial protocol. The controller Arduino will be the master, while the other two will be slave. The master will request the readings from the analyzer Arduino, which will then send data from the requested sensor back to the master. The master will use this info to determine if all conditions are met. The master will also send set parameters to the analyzer slave. The master will also keep track of time. When the time matches the light-on time or the light-off time, the master will send a command to the performer Arduino to turn on or off the light.

Expected I/O:

The analyzer Arduino will have sensors as input. It will have the soil moisture sensor and a DHT11 sensor for both humidity and temperature. The performer Arduino will have one input and one output. The output will be a 5v relay that controls an outlet with a light plugged into it. The input will be a button that will enable/disable the light (will stay off even when signal to turn on the light is sent). The master will have the other two Arduinos as I/O devices, as well as a button as an input to reset parameters and an LCD screen to output the current time and points. It will also be connected to a computer, which will be used to set the conditions of the plant.

Original Work:

The original work includes the inclusion of an outlet controller, light timer, a plant condition alert, and the ability to set the conditions to a user specified value, allowing the system to be used by any plant. Also included is a simple point system where points are awarded/taken away depending on whether or not the plant's proper conditions are met. The system is essentially a game based on physical plant care.

How to Build

Outlet Controller Setup (MAKE SURE CORD IS NOT PLUGGED IN)

1. Cut extension cord’s wire near the base of the power block using wire cutters
2. Use razor to cut away 8 inches of the cord’s shielding
3. Cut a 4in piece of the hot wire (usually black, attached to small prong of outlet)
4. Strip ¼ in off one end of the piece of hot wire and ¾ in off the other end
5. Strip ¼ in off the end of the hot wire attached to cord plug
6. Strip ¾ in off the other two wires attached to the cord plug
7. Attach NM/SE connector to outlet box
8. Put the three wires attached to the cord plug through the NM/SE connector
9. Place the ¼ in stripped end of the piece of hot wire into the 5v relay’s NO terminal and tighten using a screwdriver
10. Place the ¾ in stripped end of the piece of hot wire around the positive terminal of the outlet and tighten using a screwdriver
11. Place the stripped end of the hot wire attached to the cord into the C terminal of the relay and tighten using a screwdriver
12. Place the stripped end of the neutral wire (usually white, attached to large prong of outlet) around the neutral terminal of the outlet and tighten using a screwdriver
13. Place the stripped end of the ground wire (usually green, attached to round prong of outlet) around the ground terminal of the outlet and tighten using a screwdriver
14. Use the jumper cables to connect the relay to the positive rail of the analyzer’s Arduino.

15. Use the jumper cables to connect the relay to the negative rail of the analyzer's Arduino.
16. Use the jumper cables to connect the relay to one of the analyzer's digital pins.
17. Use the razor and cut a groove in the outlet box for the cables to run through.
18. Tighten the NM/SE connecter with the screwdriver.
19. Place relay inside outlet box and align cables with the cut groove.
20. Place outlet on outlet box and screw outlet onto box with the screwdriver.
21. Place outlet cover onto outlet box.
22. Plug plant light into outlet and turn light switch to ON position.

Plant Setup

1. Buy a plant, appropriate soil, appropriately sized pot, and pot saucer.
2. Look up the plant's care requirements and make notes of the required temperature range, humidity range, recommended photoperiod, and watering requirements.
3. Place the saucer under the pot.
4. Fill the pot with soil.
5. Make a hole about the size of the plant's current pot
6. Take plant out of the pot and place in hole
7. Fill the hole with soil
8. Place Arduino moisture sensor into soil by the plant's roots

Master Arduino Setup

1. Connect a jumper cable from the GND pin of the master controller arduino to the negative rail of the breadboard.
2. Connect another jumper cable from the 5V pin of the arduino to the positive rail of the breadboard.
3. Place a button somewhere on the master controller breadboard.
4. Connect a jumper cable from one of the legs of the button to the positive rail.
5. Connect a 10k ohm resistor to the other leg of the button.
6. Connect a jumper cable to the 10k ohm resistor and to the negative power rail.
7. Connect a jumper cable from another leg of the button to a digital pin of the Arduino.

Analyzer Arduino Setup

1. Connect a jumper cable from the GND pin of the analyzer Arduino to the negative rail of the breadboard.
2. Connect another jumper cable from the 5V pin of the Arduino to the positive rail of the breadboard.
3. Place a DHT11 somewhere on the analyzer breadboard.
4. Connect a jumper cable from an analog pin (A0-A3) to one of the legs of the DHT11.
5. Connect a jumper cable from the positive rail to one of the legs.
6. Connect a wire from the negative rail to one of the legs.

7. Connect a jumper cable from an Arduino analog pin (A0-A3) to the SIG pin of the soil moisture sensor.
8. Connect a jumper cable from the negative rail of the breadboard to the GND pin of the soil moisture sensor.
9. Connect a jumper cable from the positive rail of the breadboard to the VCC pin of the soil moisture sensor.

Performer Arduino Setup

1. Attach wire from 5v pin to the breadboard’s positive power rail
2. Attach wire from GND pin to breadboard’s negative power rail
3. Place a button somewhere on the master controller breadboard.
4. Connect a jumper cable from one of the legs of the button to the positive rail.
5. Connect a 10k ohm resistor to the other leg of the button.
6. Connect a jumper cable to the 10k ohm resistor and to the negative power rail.
7. Connect a jumper cable from another leg of the button to a digital pin of the Arduino.
8. Connect a jumper cable from the positive power rail to the VCC pin of the 5v relay .
9. Connect a jumper cable from the negative power rail to the GND pin of the relay.
10. Connect a jumper cable from an analog pin (A0-A3) to the SIG pin on the relay.

Wire Communication Setup

1. Attach wires to pins A4 and A5 to the master, analyzer, and performer Arduinos.
2. Attach all the wires attached to A4 to the same terminal strips on the master's breadboard.
3. Attach all the wires attached to A5 to the same terminal strips on master's breadboard
4. Attach wires to the negative power rail on the corresponding breadboards of the analyzer and performer Arduinos.
5. Attach the wires from step 4 to the negative power rail on the master's breadboard so that all 3 Arduinos share the same ground.

Connect Potentiometer to Breadboard (do for master and analyzer Arduinos)

1. Attach potentiometer to breadboard
2. Connect potentiometer to positive power rail (5v)
3. Connect potentiometer to negative power rail (GND)

Connect LCD Screen to Breadboard and Arduino (do for master and analyzer Arduinos)

1. Attach LCD screen to breadboard.
2. LCD pin 1 to negative power rail on breadboard (GND)
3. LCD pin 2 to positive power rail on breadboard (5v)
4. LCD pin 3 to wiper (output) of potentiometer to control LCD contrast
5. LCD pin 4 to Arduino digital pin 12
6. LCD pin 5 to negative power rail on breadboard (GND)
7. LCD pin 6 to Arduino digital pin 11
8. LCD pin 11 to Arduino digital pin 5-LCD pin 12 to Arduino digital pin 4
9. LCD pin 13 to Arduino digital pin 3
10. LCD pin 14 to Arduino digital pin 2
11. LCD pin 15 to positive power rail on breadboard (5v) with 220 ohm resistor
12. LCD pin 16 to negative power rail on breadboard (GND)

How to Use

1. Research and record your plant’s parameters for the proper temperature, humidity, photoperiod, and watering routine.
2. Power all 3 Arduinos
3. Plug in outlet controller and make sure light switch is in the on position
4. Enter parameters.
    a. To get soil range, water plant and read soil readings. Add 10 to get the maximum soil moisture parameter and subtract 10 to get the minimum soil moisture parameter.
5. Input current time.
6. Water the plant when necessary to prevent point loss.

There will be a warning when parameters are not met. This will be displayed on the LCD screen attached to the analyzer’s breadboard. The warning will tell you if the moisture, temperature, or humidity is too high or low. Only one warning will be displayed at a time, with the current value displayed under the warning for more accurate solutions.

<u>Completed Timeline:</u>

Week 1: Write code for communication between 3 Arduinos

Week 2: Create outlet controlled by Arduino

Week 3: Figure out proper sensor reading methods

Week 4: Implement the I/O devices in the individual arduinos

Week 5: Combine code for the arduinos and their I/O devices to work together, start water pump building

Week 6: Work on the original idea and add additional implementations, finish water pump

Week 7 : 11/25/19: Design Presentation, implemented game system, abandoned water pump (did not work, all 3 DC motors broke)

Week 8: 12/2/19 : Project demonstration

Materials:

3x Arduinos

3x Arduino Power Sources (laptop, battery, or wall plug)

1x Soil Moisture Sensor

1x DHT11 Sensor

1x 5v Relay

1x Outlet

1x Outlet Box and Cover

1x Extension Cord

1x NM/SE Connector

1x Wire Strippers/Cutters

1x Phillips Head Screwdriver

1x Flat Head Screwdriver

1x Computer

1x Plant Light

2x Button

2x LCD Screen

2x Potentiometers

2x 220 Ohm Resistor

2x 10k Ohm Resistor

Jumper Wires

References:

Button:

https://www.arduino.cc/en/Tutorial/Button

I2C Communication with Wire:

https://www.arduino.cc/en/Tutorial/MasterWriter

Base Plant Watering System

https://create.arduino.cc/projecthub/neetithakur/automatic-plant-watering-system-
using-arduino-uno-8764ba

Arduino Controlled Power Outlet:

http://www.circuitbasics.com/build-an-arduino-controlled-power-outlet/

Humidity and Temperature Sensor:

https://create.arduino.cc/projecthub/Arca_Ege/using-dht11-b0f365

LiquidCrystal

https://www.arduino.cc/en/Reference/LiquidCrystal

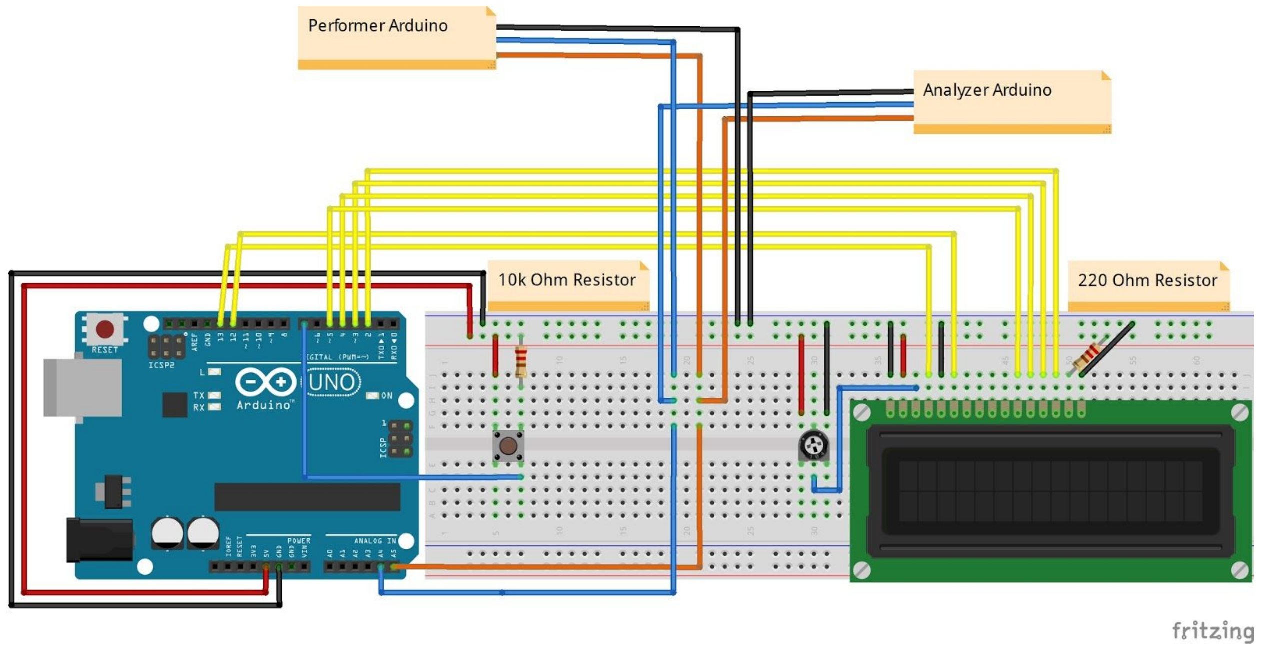
Time

https://playground.arduino.cc/Code/Time/

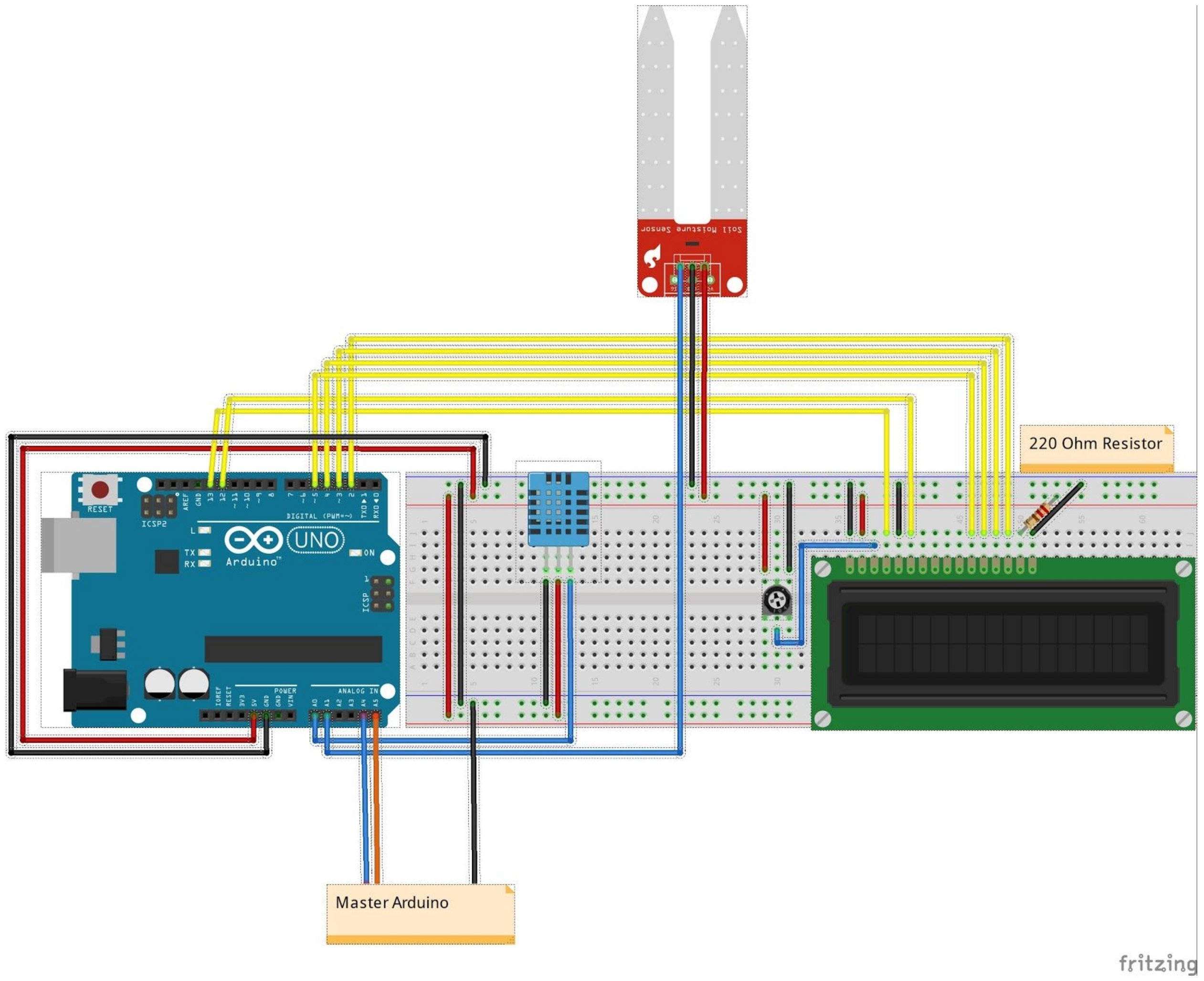
DHT Sensor Library

https://github.com/adafruit/DHT-sensor-library

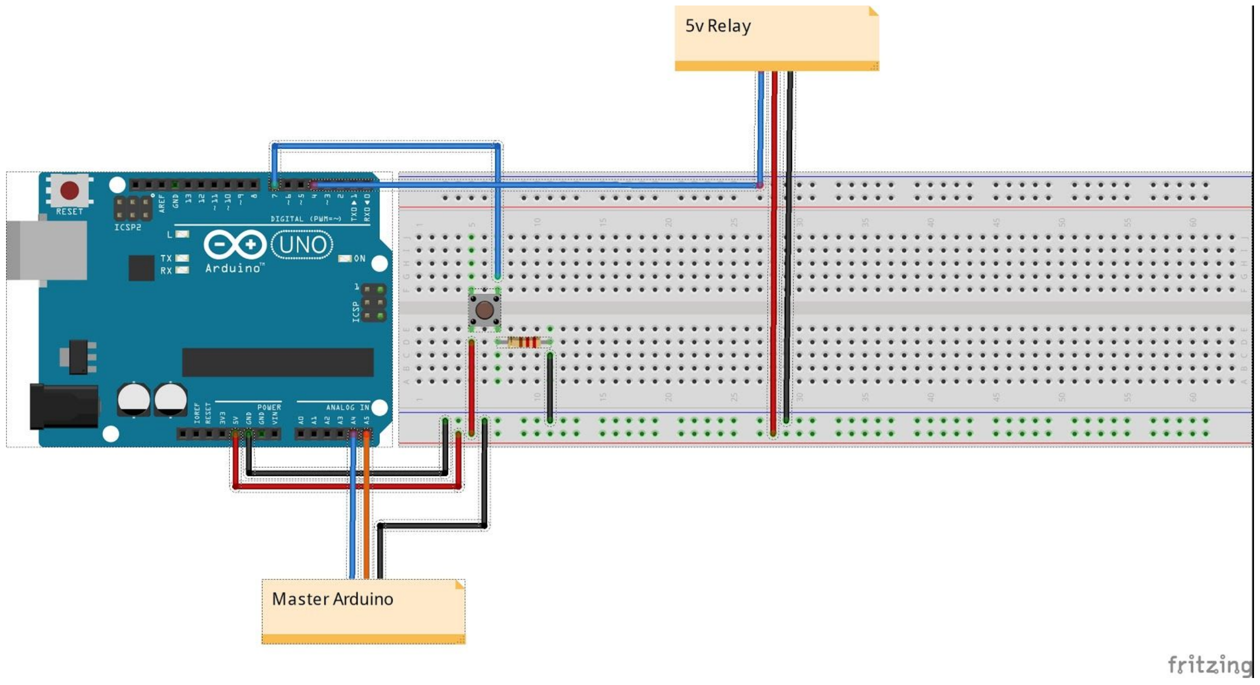
Master Diagram



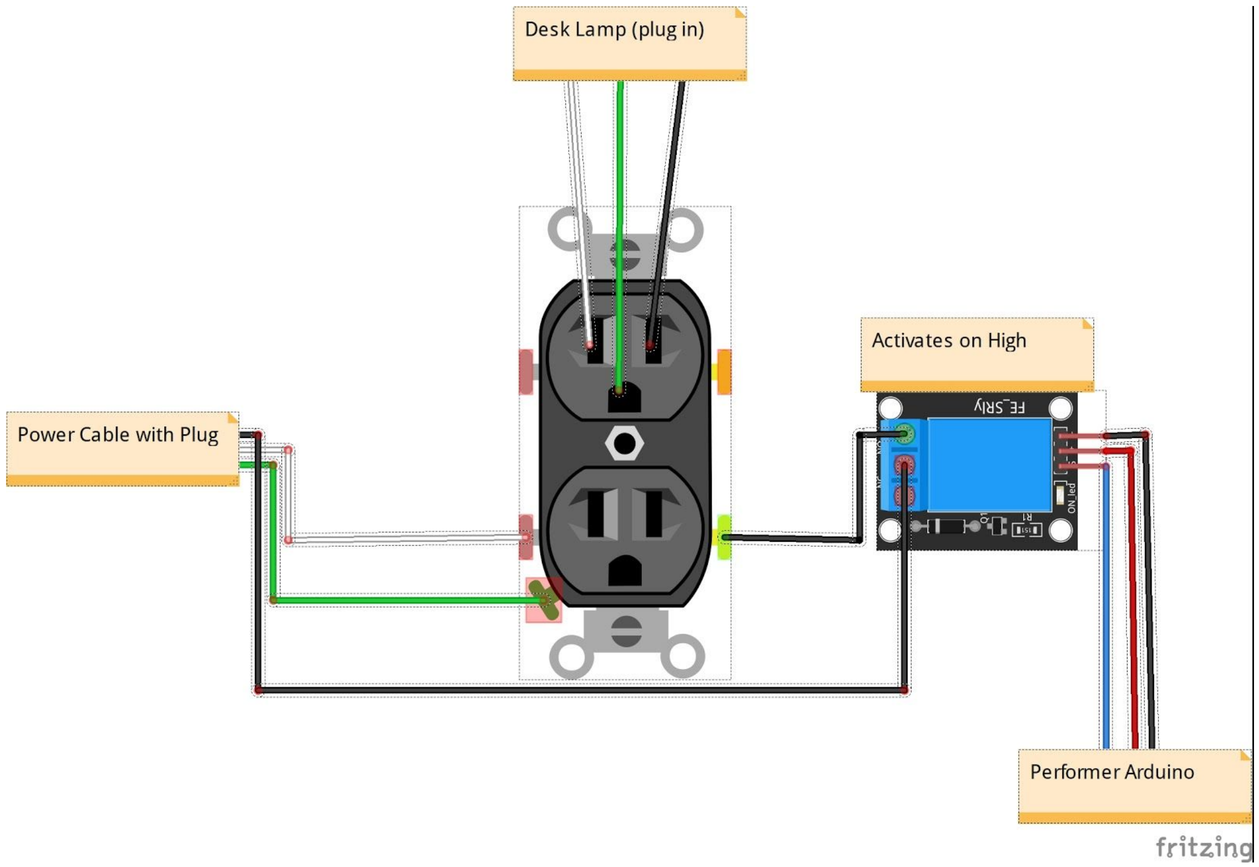
Analyzer Diagram



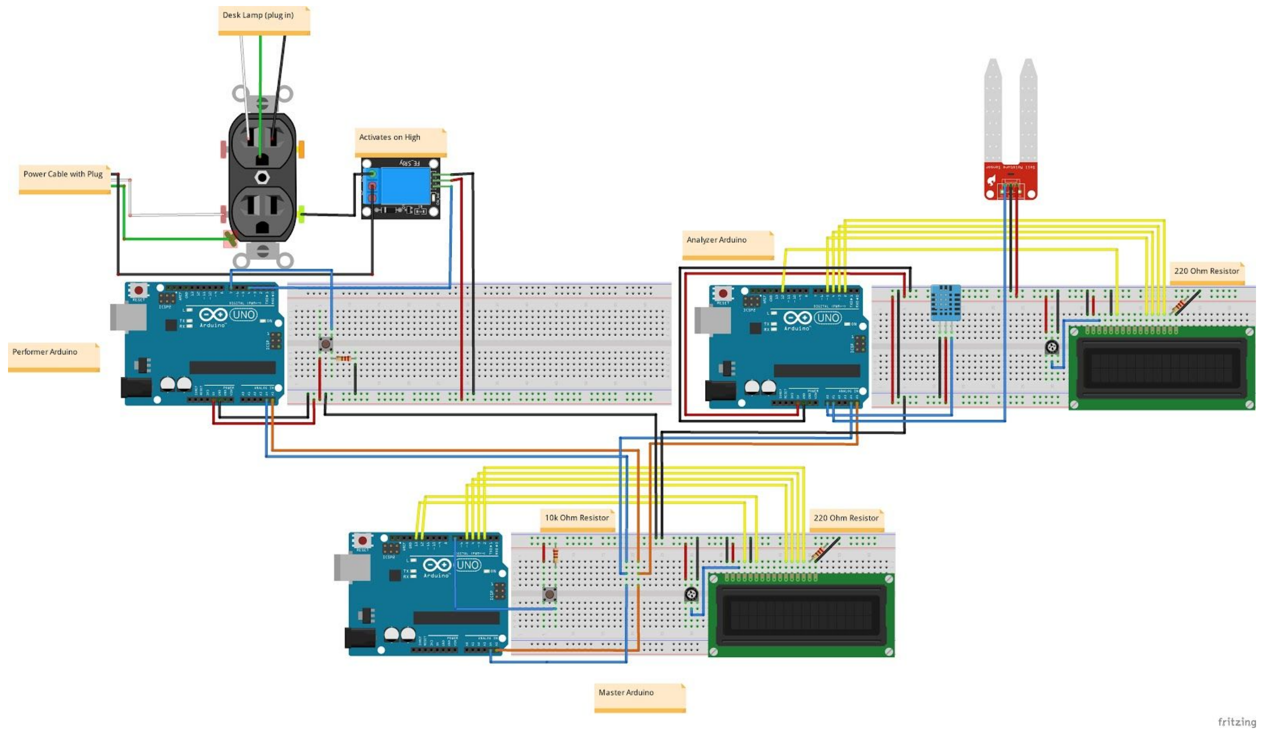
Performer Diagram



Outlet Diagram



All Together Diagram



<u>Sample Code:</u>

Master Controller

```
/*  master arduino
 *  This Arduino is the master Arduino and controls the other 2 Arduinos.
 *  Parameters are set in Serial by the user in this Arduino. Those parameters
 *  are sent to the analyzer Arduino for condition checking. The current values
 *  from sensors are sent from the analyzer to this Arduino, where the current
 *  values are checked. If the values are not within the proper range, points are
 *  subtracted. If they are in proper range, points are added. The current time and
 *  points are displayed on an LCD screen attached to this Arduino. This Arduino
 *  also sends commands to the performer Arduino. These commands turn on or off the
 *  light attached to the performer Arduino. There is also a button on this Arduino
 *  that allows the resetting of the parameters.
 *
 *  Team 34
 *  Nathan Phan - nphan23
 *  Ali S Baig - abaig28
 *  Mohab Mustafa - mmustaf2
 *  EzPlant
 We will be using multiple Arduinos to control a plant care system which monitors the
plant's environmental conditions, soil moisture,
 and controls a light. Three Arduinos will communicate with each other and toggle the
light based on time and send sensor readings and
 plant parameters, which will be stored in a master Arduino and will be set by the user.
These sensors will measure humidity, temperature,
 and soil moisture. The environment conditions are used to create a game, where points
are received if the plant is being properly cared for and
 points are detracted when conditions are not met.
*/

#include <LiquidCrystal.h>
#include <Wire.h>
#include <Time.h>
#include <TimeLib.h>

// button for parameter reset
int buttonPin = 7;

// initialize pins for LCD
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// initialize variables
```

```
int curHour, curMin, curSec, lastSec;
int minMoist, maxMoist, curMoist;
int minTemp, maxTemp, curTemp;
int minHum, maxHum, curHum;
int lastRead, curRead = 0;
int buttonState, checkOne, checkTwo;
int onHour, offHour;
int points = 0;

int analyzeVals[3]; // array for receiving current values from the analyzer
int minMaxes[6];      // array for sending min/max parameters to the analyzer

// commands for operator
char lightOn = 'O';
char lightOff = 'F';

// command for analyzer
char getMoisture = 'M';

// address of arduinos
int master = 0;
int analyzer = 1;
int performer = 2;

void setup() {
  Serial.begin(9600);

  // set master address
  Wire.begin(master);

  // set up lcd screen
  lcd.begin(16, 2);

  // set up button
  pinMode(buttonPin, INPUT);
  buttonState = digitalRead(buttonPin);

  // set up plant parameters
  setParameters();
}

void loop() {
  // displays current time
  lcdDisplay();
  // checks for button bounce
  checkOne = digitalRead(buttonPin);
  delay(10);
  checkTwo = digitalRead(buttonPin);
  if (checkOne == checkTwo)
  {
        if (checkOne != buttonState)
        {
        // if button was pressed, allow reset of parameters
        if (checkOne == HIGH)
        {
        setParameters();
        }
        }
  }
  // assign buttonState to last state to prevent counting button holds
  buttonState = checkOne;

  // every ten minutes, request moisture, temperature, and humidity values from analyzer
slave
  // give or detract points depending on current values
  // repeat every 10 minutes
  curRead = minute();

  if (lastRead > curRead)
  {
        curRead += 60;
  }
  curRead -= lastRead;

  // if 10 minutes has passed since last read
  if (curRead == 10)
  {
        // set time to current time to reset counter
        lastRead = minute();

        // request current moisture readings from analyzer
        requestData(analyzeVals, analyzer);

        // set current values to values read from the analyzer
        curTemp = analyzeVals[0];
        curHum = analyzeVals[1];
        curMoist = analyzeVals[2];

        // if the plant's moisture is not within range, subtract 10 points
		if (curMoist < minMoist || curMoist > maxMoist)
		{
		points -= 10;
		}
		// if plant is watered properly, add 50 points
		else
		{
		points += 50;
		}

		// if environment's temparture is not within range, subtract 5 points
		if (curTemp < minTemp || curTemp > maxTemp)
		{
		points -= 5;
		}
		// if the temperature is acceptable, add 25 points
		else
		{
		points += 25;
		}

		// if environment's humidity is not within range, subtract 5 points
		if (curHum < minHum || curHum > maxHum)
		{
		points -= 5;
		}
		// if humidity is acceptable, add 25 points
		else
		{
		points += 25;
		}
	}

	// if current time is between time to turn on and time to turn off
	if (hour() >= onHour && hour() < offHour)
	{
		Serial.println("TURN ON");
		// send command to performer to turn light on
		sendCommand(lightOn, performer);
	}

	// if current time is after time to turn off or before time to turn on
	if (hour() >= offHour || hour() < onHour)
	{
		Serial.println("TURN OFF");
		// send command to performer to turn light off
		sendCommand(lightOff, performer);
	}

	lcd.clear();
}

// displays the current time on the lcd screen
void lcdDisplay()
{
	// print current time
	lcd.setCursor(0, 0);
	lcd.print(hour());
	lcd.print(":");
	lcd.print(minute());
	lcd.print(":");
	if (second() < 10)
	{
		lcd.print("0");
	}
	lcd.print(second());

	lcd.setCursor(0, 1);
	lcd.print("Points: ");
	lcd.print(points);
}

// clears Wire buffer
void clearWire()
{
	while (Wire.available())
	{
		Wire.read();
	}
}



// clears Serial buffer
void clearSerial()
{
	while (Serial.available())
	{
		Serial.read();
	}
```

```
}


// reads input from Serial
void getInput(int& data)
{
  // wait for serial input
  while (!Serial.available()) {}

  // get input from Serial as an int
  data = Serial.parseInt();

  // clear Serial input buffer by reading the rest of the buffer
  clearSerial();
}



// request data from target
void requestData(int data[], int target)
{
  Wire.requestFrom(target, 3);

  for (int i = 0; i < 3; i++)
  {
        if (Wire.available())
        {
        analyzeVals[i] = Wire.read();
        }
  }
  clearWire();
}



// sends data to analyzer arduino
void sendCommand(char command, int target)
{
  Wire.beginTransmission(target); // transmit to target arduino
  Wire.write(command);      // sends five bytes
  Wire.endTransmission();  // stop transmitting
  clearWire(); // stop transmitting
}
```

```
// sends data to analyzer arduino
void sendAnalyze(int data[])
{
  Wire.beginTransmission(analyzer); // transmit to analyzer

  for (int i = 0; i < 6; i++)
  {
        Wire.write(data[i]);             // sends data
  }
  Wire.endTransmission();  // stop transmitting
  clearWire(); // stop transmitting
}



// allows setting of plant parameters
void setParameters()
{
  Serial.println("Set paramaters. Mins must be less than maxes.");

  // enter time to turn light on
  Serial.println("Enter Hour to Turn On Light (0-22): ");
  getInput(onHour);
  // checks if input is in correct interval
  while (onHour < 0 || onHour > 22)
  {
        Serial.println("Enter Hour to Turn On Light (0-22): ");

        // if input was wrong, repeat loop until correct
        getInput(onHour);
  }
  Serial.println(onHour);


  // enter time to turn light off
  Serial.print("Enter Hour to Turn Off Light (");
  Serial.print(onHour + 1);
  Serial.println("-23): ");
  getInput(offHour);
  // checks if input is in correct interval
  while (offHour <= onHour || offHour > 23)
  {
        Serial.print("Enter Hour to Turn Off Light (");
        Serial.print(onHour + 1);
        Serial.println("-23): ");

        // if input was wrong, repeat loop until correct
        getInput(offHour);
}
Serial.println(offHour);


// enter minimum temperature
Serial.println("Enter Minimum Temperature (0-99): ");
getInput(minTemp);
// checks if input is in correct interval
while (minTemp < 0 || minTemp > 99)
{
        Serial.println("Enter Minimum Temperature (0-99): ");

        // if input was wrong, repeat loop until correct
        getInput(minTemp);
}
Serial.println(minTemp);


// enter maximum temperature
Serial.print("Enter Maximum Temperature (");
Serial.print(minTemp + 1);
Serial.println("-100): ");
getInput(maxTemp);
// checks if input is in correct interval
while (maxTemp <= minTemp || maxTemp > 100)
{
        Serial.print("Enter Maximum Temperature (");
        Serial.print(minTemp + 1);
        Serial.println("-100): ");

        // if input was wrong, repeat loop until correct
        getInput(maxTemp);
}
Serial.println(maxTemp);


// enter minimum humidity
Serial.println("Enter Minimum Humidity (0-99): ");
getInput(minHum);
// checks if input is in correct interval
while (minHum < 0 || minHum > 99)
```

```
{
        Serial.println("Enter Minimum Humidity (0-99): ");

        // if input was wrong, repeat loop until correct
        getInput(minHum);
}
Serial.println(minHum);


// enter maximum humidity
Serial.print("Enter Maximum Humidity (");
Serial.print(minHum + 1);
Serial.println("-100): ");
getInput(maxHum);
// checks if input is in correct interval
while (maxHum <= minHum || maxHum > 100)
{
        Serial.print("Enter Maximum Humidity (");
        Serial.print(minHum + 1);
        Serial.println("-100): ");

        // if input was wrong, repeat loop until correct
        getInput(maxHum);
}
Serial.println(maxHum);


// enter minimum soil moisture
Serial.println("Enter Minimum Soil Moisture (0-99): ");
getInput(minMoist);
// checks if input is in correct interval
while (minMoist < 0 || minMoist > 99)
{
        Serial.println("Enter Minimum Soil Moisture (0-99): ");

        // if input was wrong, repeat loop until correct
        getInput(minMoist);
}
Serial.println(minMoist);


// enter maximum soil moisture
Serial.print("Enter Maximum Soil Moisture (");
Serial.print(minMoist + 1);
Serial.println("-100): ");
```

```
getInput(maxMoist);
// checks if input is in correct interval
while (maxMoist <= minMoist || maxMoist > 100)
{
	Serial.print("Enter Maximum Soil Moisture (");
	Serial.print(minMoist + 1);
	Serial.println("-100): ");

	// if input was wrong, repeat loop until correct
	getInput(maxMoist);
}
Serial.println(maxMoist);

// store the parameters into an array
minMaxes[0] = minTemp;
minMaxes[1] = maxTemp;

minMaxes[2] = minHum;
minMaxes[3] = maxHum;

minMaxes[4] = minMoist;
minMaxes[5] = maxMoist;

// send array of parameters to analyzer arduino
sendAnalyze(minMaxes);

// enter current hour
Serial.println("Enter Current Hour (0-23): ");
getInput(curHour);
// checks if input is in correct interval
while (curHour < 0 || curHour > 22)
{
	Serial.println("Enter Current Hour (0-23): ");

	// if input was wrong, repeat loop until correct
	getInput(curHour);
}
Serial.println(curHour);

// enter current minute
Serial.println("Enter Current Minute (0-59): ");
getInput(curMin);
// checks if input is in correct interval
while (curMin < 0 || curMin > 59)
{
        Serial.println("Enter Current Minute (0-59): ");

        // if input was wrong, repeat loop until correct
        getInput(curMin);
  }
  Serial.println(curMin);

  // enter current second
  Serial.println("Enter Current Second (0-59): ");
  getInput(curSec);
  // checks if input is in correct interval
  while (curSec < 0 || curSec > 59)
  {
        Serial.println("Enter Current Second (0-59): ");

        // if input was wrong, repeat loop until correct
        getInput(curSec);
  }
  Serial.println(curSec);

  setTime(curHour, curMin, curSec, 0, 0, 0);
  lastRead = minute();
}
```

Analyzer Slave

```
/* analyzer Arduino
 * This Arduino utilizes the DHT11 sensor to read the room's temperature
 * and humidity. A soil moisture sensor is also used to read the moisture
 * in soil. These values are displayed on an LCD screen. The values are sent
 * to the master Arduino every 10 minutes. The master Arduino will send
 * minimum and maximum parameters for the temperature, humidity, and moisture
 * to this Arduino. This Arduino will use those parameters to determine
 * if the current values are within the proper range. If they are not,
 * then a warning will be displayed.
 */

#include <LiquidCrystal.h> // library for lcd
#include "DHT.h" // library to make the sensor work
#include <Wire.h> // communication library

#define DHTPIN A0 // dht11 pin
#define DHTTYPE DHT11 //type of our dht sensor

// set up LCD
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// set up DHT11 sensor
DHT dht(DHTPIN, DHTTYPE);

int moistPin = A1; // library for lcd

// initialize variables
int curHum, curTemp, curMoist;
int minTemp, maxTemp, minHum, maxHum, minMoist, maxMoist;
int started = 0;
int master = 0;
int analyzer = 1;

// array of min/max parameters sent from master
int minMaxes[6];

void setup() {
  Serial.begin(9600);          // start Serial for debugging
  Wire.begin(analyzer);        // start Wire address for analyzer (1)
  Wire.onRequest(sendInfo);  // when data is requested, call sendInfo
  Wire.onReceive(getInfo);   // when data is received, call getInfo
  lcd.begin(16, 2);            // setting up the lcd
  dht.begin();          // setting up the dht
```

```
}

void loop() {

  // get temperature and humidity
  curHum = (int) dht.readHumidity();
  curTemp = (int) dht.readTemperature(true);

  // display temp and humidity on LCD screen
  lcd.setCursor(0,0);
  lcd.print ("T: ");
  lcd.print (curTemp);
  lcd.print ("*F ");
  lcd.print ("H: ");
  lcd.print (curHum);
  lcd.print ("%");

  // get moisture values from sensor
  curMoist = analogRead(moistPin);

  // map moisture values to 0-100 from 320-620 for better readability
  curMoist = map(curMoist, 320, 620, 100, 0);

  // display moisture value on LCD screen
  lcd.setCursor(0, 1);
  lcd.print("M: ");
  lcd.print(curMoist);

  // if the min/max parameters have been received from the master
  if (started == 1)
  {
        // if curTemp is lower than minimum, display warning
        if (curTemp < minTemp)
        {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Freezing!");
        lcd.setCursor(0, 1);
        lcd.print(curTemp);
        }

        // if curTemp is higher than maximum, display warning
        else if (curTemp > maxTemp)
        {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Burning!");
        lcd.setCursor(0, 1);
        lcd.print(curTemp);
        }

        // if curHum is lower than minimum, display warning
        else if (curHum < minHum)
        {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Hum too Low!");
        lcd.setCursor(0, 1);
        lcd.print(curHum);
        }

        // if curHum is higher than maximum, display warning
        else if (curHum > maxHum)
        {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Hum too High!");
        lcd.setCursor(0, 1);
        lcd.print(curHum);
        }

        // if curMoist is lower than minimum, display warning
        else if (curMoist < minMoist)
        {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("More Water!");
        lcd.setCursor(0, 1);
        lcd.print(curMoist);
        }

        // if curTemp is higher than maximum, display warning
        else if (curMoist > maxMoist)
        {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Drowning!");
        lcd.setCursor(0, 1);
        lcd.print(curMoist);
        }
```

```
  }

  delay(500); // delay for LCD readability
  lcd.clear(); // clear LCD display
}

// called when a request is made from master, sends current sensor readings
void sendInfo()
{
  Wire.write(curTemp);
  Wire.write(curHum);
  Wire.write(curMoist);
}

// called when info is received from the master, sets the min/max variables stored in
master
void getInfo(int howMuch)
{
  // populate array with min/max parameters
  for (int i = 0; i < 6; i++)
  {
        if (Wire.available())
        {
        minMaxes[i] = Wire.read();
        }
  }

  // set local min/max variables to values sent from master
  minTemp = minMaxes[0];
  maxTemp = minMaxes[1];

  minHum = minMaxes[2];
  maxHum = minMaxes[3];

  minMoist = minMaxes[4];
  maxMoist = minMaxes[5];

  // Serial output for debugging
  Serial.println(minTemp);
  Serial.println(maxTemp);
  Serial.println(minHum);
  Serial.println(maxHum);
  Serial.println(minMoist);
  Serial.println(maxMoist);
  // set started to 1, will enable the parameter checking conditions
  started = 1;
}
```

Performer Slave

```
/* performer Arduino
 * This Arduino controls a 5v relay that controls power flow to
 * an outlet. The Arduino receives commands (turn on or turn off)
 * from the master Arduino and performs operations based on those
 * commands. The Arduino also has a button that acts as an on/off
 * toggle for the 5v relay. When the button is pressed, the signal
 * written to the relay will be LOW, no matter what command is
 * received from the master.
*/

#include <Wire.h>

// initialize pins and variables
const int light = 4;
const int buttonPin = 7;
int buttonState;
char lightCom = ' ';
int lightState;

// for debounce check
int checkOne = 0;
int checkTwo = 0;

// light enable variable
bool isEnabled = true;

void setup() {
  Wire.begin(2); // start address as 2 (performer address)
  Wire.onReceive(receiveEvent); // when something is received from the master, call
receiveEvent

  Serial.begin(9600);
  pinMode(light, OUTPUT);   // initialize the digital pin as an output.
  pinMode(buttonPin, INPUT); // initialize the digital pin as an input.
}

void loop() {
  // debounce check for the button, prevents bounce and only counts clicks, not holds
  checkOne = digitalRead(buttonPin);
  delay(10);
  checkTwo = digitalRead(buttonPin);

  if (checkOne == checkTwo) {
        if (checkOne != buttonState) {
        if (checkOne == HIGH) {
        if (isEnabled == false) {
        isEnabled = true;

        Serial.println("Enable light");
        } else {
        isEnabled = false;
        digitalWrite(light, LOW);
        Serial.println("Disable light");
        }
        }
        }
  }
  buttonState = checkOne;

  lightState = digitalRead(light);

  // if the command received is to turn on the light and the light is enabled, turn on the
light if it is not already on
  if (isEnabled == true && lightCom == 'O') {
        if (lightState != HIGH)
        {
        digitalWrite(light, HIGH); // turn light on
        }
  }

  // if the command received is to turn off the light and the light is enabled, turn off the
light if it is not already off
  else if (isEnabled == true && lightCom == 'F')
  {
        if (lightState != LOW)
        {
        digitalWrite(light, LOW); // turn light off
        }
  }
}

// function that executes whenever data is received from master
// this function is registered as an event, see setup()
void receiveEvent(int howMany)
{
  while(Wire.available()) // read all bytes on the Wire buffer
  {
        lightCom = Wire.read(); // receive byte as a character
        Serial.print(lightCom);         // print the character (debugging)
  }
}
```