Team 10 Documentation
Automated Hydroponics Garden

Nakhul Kisan, Fernando Vazquez, and John Paul Bautista

Table of Contents

Introduction

The concept behind this project was to build a greenhouse that can self-regulate and provide urban dwellers a way to produce food within their own home. By allowing consumers access to their own gardens in crowded, polluted cities, we are not only allowing consumers to produce their own food, we are also tackling city pollution by decreasing the environmental impact of erosion of soil and methane release from manure fertilizers. In this project, we have shown the steps to produce a basic greenhouse automated hydroponics garden in which the user will be able to plug in water hose and plug the project into a wall outlet such that it will self sustain and water itself whenever it gets to dry.

Materials

Components

- Arduino Uno
- DHT11 humidity and temperature sensor
- LCD display
- Solenoid Valve
- TIP120 Darlington transistor
- 1N40001 Diode
- Potentiometer
- 1K ohm resistor
- 220-ohm resistor
- Hookup wires (male/male and female to male)
- Solderless Breadboard
- 13 mm FNPT x 19mm FHT
- 10 ft. hose
- 1/2x72'' Wooden Dowel
- 1/4x48'' Wooden Dowel
- Cardboard
- Misting Nozzle (Not Pictured)

Tools

- Wood Glue
- Hacksaw
- Clamps
- 9V Power Supply
- Laptop
- Goggles
- Masking Tape

Software

- Arduino IDE

Instructions

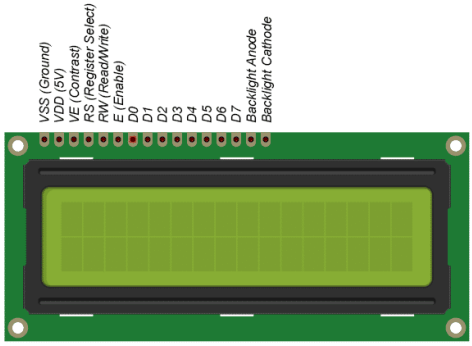
1. Wiring up the components

   LCD Screen circuit

   - Materials needed: 15 hookup wires, 220-ohm resistor, potentiometer, LCD screen breadboard, and the Arduino Uno.

   To start off, we want to connect a wire from the 5V pin of the Arduino to the positive rail of the breadboard, and a wire from the GND pin in the Arduino to the negative rail of the breadboard. Then we want to plug in female to male wires on the LCD pins onto the breadboard so that we can elevate the screen and place it onto on the

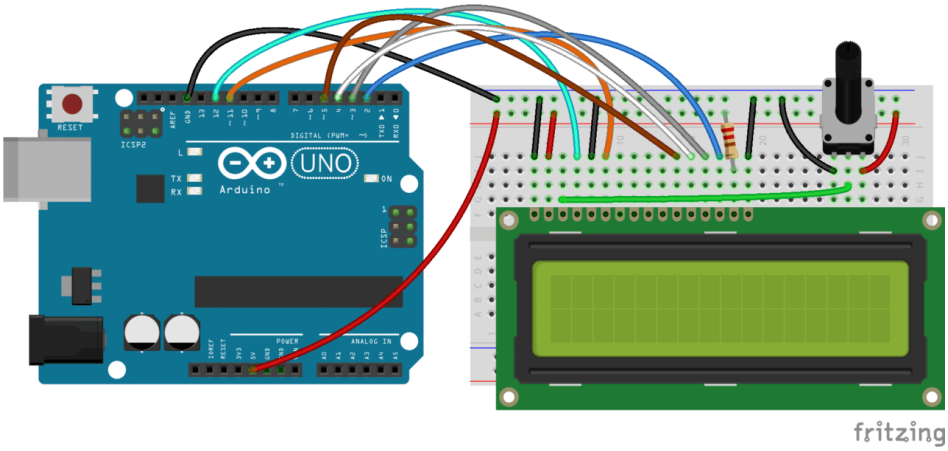
face of its enclosure box. Since not all pins will be used, we included a chart with details on which pins will be used as well as where to route them. We also want to connect the potentiometer just to the side of the wires since we will be routing wires to it. The diagram of the LCD goes as follows:



Starting from left to right up to pin E, and then jumping ahead to pin 11 we connect that given pin to the corresponding places based on the following chart:

From (Pin #, Name)	To
1. VSS	Ground rail
2. VDD	5V rail
3. VE (contrast)	Middle pin of potentiometer
4. RS (Register Select)	Arduino pin 12
5. RW (Read/Write)	Ground rail
6. E (Enable)	Arduino pin 11
11. D4	Arduino pin 5
12. D5	Arduino pin 4
13. D6	Arduino pin 3
14. D7	Arduino pin 2
15. Backlight Anode	Resistor 5V rail
16. Backlight Cathode	Ground rail
Potentiometer left most pin	Ground rail
Potentiometer right most pin	5V rail

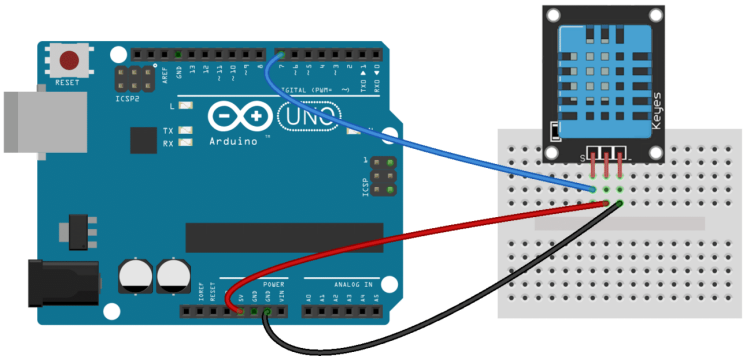
The final result should reference the following image, however instead of the LCD pins being connected to the breadboard, it will be the extended cables that we attached earlier to the pins of the LCD.



DHT11 Circuit

- Materials needed: DHT11 sensor, 3 hookup wires and wire extenders (female to female)

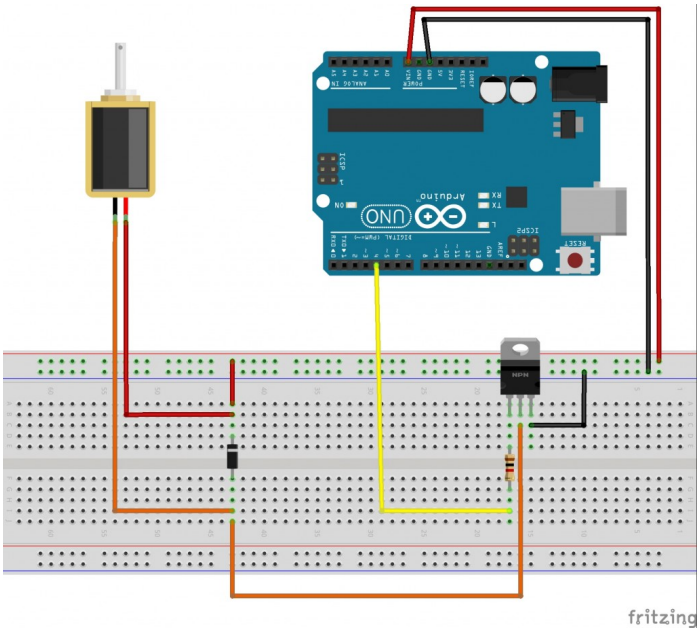
  For this part, we will be connecting the DHT11 Sensor to the Arduino. For this, we will be connecting one wire from the VCC pin of the sensor to the 5V rail of the breadboard with the LCD on it. The middle pin of the sensor is signal pin which will be outputting our data so this will be attached to pin 7 of the Arduino. The final pin is the ground pin, which can be connected to the ground rail of the LCD breadboard. All of these wire connections however should be extended with the wire extenders since the DHT11 will be placed within the structure of the project far from the other elements. If done correctly, it should look like similar to the following diagram, except that being connected to the breadboard, it will be connected with directly with wires and extenders so it has enough room to be placed far from the Arduino.



Solenoid Circuit

- Materials needed: Solenoid Valve, 1N40001 Diode, 1K ohm resistor, TIP120 Darlington transistor, 5 hookup wires

To begin with, we first must connect a wire from the Vin pin of the Arduino to the positive side of the rail on the opposite end of the breadboard with the LCD. This way, we will have a 5V rail in one side, and the 9V rail on the opposite side to power the solenoid. We then place the diode such that the white strip is facing the positive 9V rail. We want to make sure this is oriented the correct way, so that there is no damage done to the Arduino board or any other components in our circuit. We we then place a wire from the positive 9V rail to the diode with the white side facing towards it. In that same node, we place a wire that will be attached to the Solenoid valve. On the opposite side of the diode, we place the other wire that will connect to the solenoid, followed by another wire which will be connected to the middle in of the transistor. The leftmost pin of the transistor will be connected in series with the 1K resistor followed by a wire that will be attached to pin 8 of the Arduino. If done correctly, it should look similar to the following:



The solenoid valve wires will be connected via alligator clips so that we can place route the wires to where the solenoid will be placed.

2. Writing the code

Initial Setup

In order to start with the code, we must ensure the Arduino IDE is installed in our computer, which can be downloaded in the official Arduino website. Once that is installed, we go ahead and connect the Arduino to the computer using the USB cable. We then want to make sure that the right settings are checked in order to correctly upload the code into the Arduino's memory. For that, we want to go into Tools and under board, we want to make sure that we select the Arduino/Genuino Uno board.

We then want to make sure that the correct port is selected. Under the same “Tools” tab, we want to go to port, and make sure COM4 is selected.

Add the required library

An important step to have the components working is making sure that we have the DHT11Lib library installed. For that, we can download it in the Arduino playground website. Once the DHTLib.zip file is downloaded, in the Arduino IDE we go under the Sketch>include>Library>Add .ZIP Library and select the DHTLib.zip file. Once those settings have been applied, we are free to start the code.

Code and explanation

Since the Arduino works with the language C, at the top we want to include the library’s necessary for the components we will be using; those being the DHT11 library and the LCD screen library. Everything that follows will be the declaration of variables and the main code functions. The code goes as follows:

```
#include <dht.h>
#include <LiquidCrystal.h>

//declaring the pins for the LCD
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
dht DHT;

#define DHT11_PIN 7

int solenoidPin = 8; //This is the solenoid controlling pin on the Arduino

void setup(){
  Serial.begin(9600);
  lcd.begin(16, 2);
  pinMode(solenoidPin, OUTPUT);
}
//prints to monitor
void printScreen(int temp, int hum){
  Serial.print("Temperature = ");
  Serial.println(temp);
  Serial.print("Humidity = ");
  Serial.println(hum);
  return;
}
//prints to LCD screen
void printLCD(int temp, int hum){
  lcd.setCursor(0,0);
  lcd.print("Temp: ");
  lcd.print(temp);
  lcd.print((char)223);
  lcd.print("C");
  lcd.setCursor(0,1);
  lcd.print("Humidity: ");
  lcd.print(hum);
  lcd.print("%");
```

```
  return;
}
//main code loop
void loop(){
  //read the most recent data
  int chk = DHT.read11(DHT11_PIN);
  int humidity = DHT.humidity;
  int temperature = DHT.temperature;

  //print raw data to LCD and/or monitor
  //printScreen(temperature, humidity);
  printLCD(temperature, humidity);

  //irrigation system loop
  if(humidity < 40){ //the lower limit of what the plant requirement
    digitalWrite(solenoidPin, HIGH); //open the solenoid
    while(DHT.humidity < 60){ //upper limit of the plant
      delay(1200); //delay it for a second until water is enough
      chk = DHT.read11(DHT11_PIN); //read most recent data
      printScreen(temperature, DHT.humidity);
      printLCD(temperature, DHT.humidity);
     }
    digitalWrite(solenoidPin, LOW);//close the solenoid valve
  }
  delay(1000); //Clock timer, checks every second
}
```

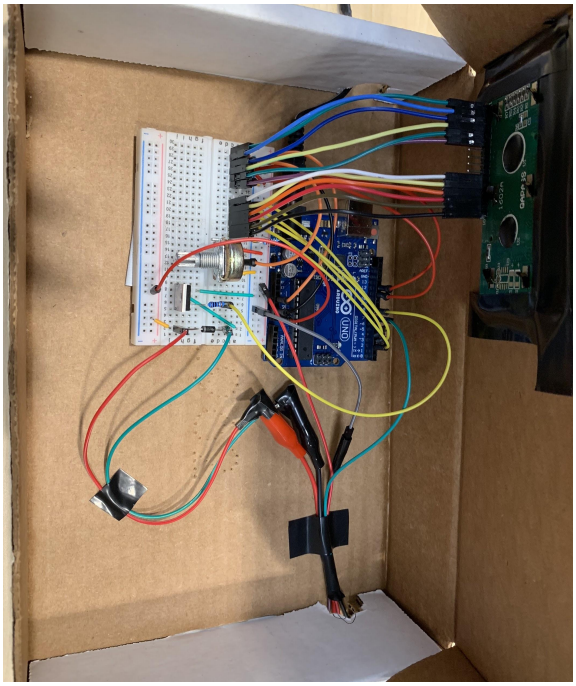
We can see that the code breaks down to four main functions. The first function is the setup function, which sets up the serial print screen used to print data to the computer monitor, initiates the LCD to print window, and assigns our solenoid pin to have an output voltage. The two functions that follow are used for printing purposes to either the monitor of the computer or to the LCD screen we have implemented given the values as parameters. The loop function is where we will be writing the heart of the code, which not only reports the measurements of the temperature and humidity, but also shuts the solenoid on and off based on the measured values. It starts by reading in the most recent data. And based on those readings, it either falls under the irrigation loop or it waits a few seconds to update its data readings to see if there are any significant changes in temperature and humidity. If at any given moment, if the humidity is less than what we desire, the solenoid will open as seeing in the source code given by “digitalWrite(solenoidPin, HIGH);”. The valve will then remain open until we break out of the loop that checks if we hit the highest percentage of humidity desired. The loop will keep running and check if we are in the bounds that we declare them to be.

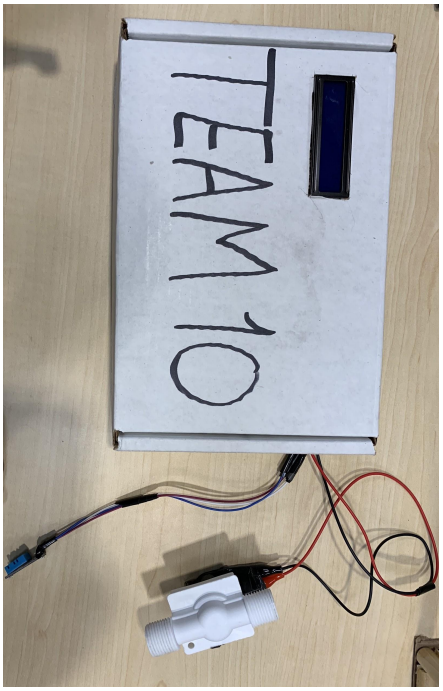
3. Building the frame

For the frame, we used the Envision tools to cut out four 20” rods and four 12” rods from the 72” dowels we purchased. The 20’’ rods were used to create the vertical axes; the 12’’ rods were used to create the horizontal axes.

Next, we built the plant suspension rods from the thinner, 48’’ dowel we bought. We cut out two rods equal to the length of the diagonal on the horizontal plane, sanded-down a semicircular hole in one of the rods, placed the other rod into that hole in order to form an “X” shape, and then glued the suspension in the middle of the vertical rods. We sanded down the top of the suspender rods to create a semi-flat surface. The whole frame was covered in plastic to maintain the humidity in the greenhouse

Cardboard was added as the base of the frame, to provide stability. Also, we created an external box to hold and protect the circuitry from water as seen on the following images.





4. Finally, we assembled our whole project by attaching the breadboard circuits, the arduino, and the hose with solenoid attachment to the sides of the dowel rods. The circuitry was placed on the outside of the system, above, and on the same dowell as the misting nozzle. This was done in order to prevent splashback from the nozzle.

   The final product:


Note: Because we were unsure on the certainty of having access to a water faucet and electric source in the presentation room, we recorded our product working beforehand and have included it for our presentation.

Conclusion

In conclusion, our project is a resource non-intensive, portable, and compact greenhouse that has a variety of uses. We are hoping that it can mainly help ease urban pollution and provide access to clean, healthy foods.

If we had more resources (time, money, etc), we would have created a more sturdy frame (thicker wood for the frame, acrylic paneling rather than plastic, wood paneling rather than cardboard, etc). Also, we would have fitted a misting nozzle onto the solenoid so that the DHT11 could function directly from the water source.

Our proudest moment was upon the completion of assembly. The whole project was finally coming together and looked amazing. Our greatest obstacle, on the other hand, was testing the system. Without the misting nozzle, we had to test it by other means. We triggered the water system by using a wet paper towel to reach the shutoff humidity on the DHT11.

References

1. DHT11 Circuit and Code
http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/

2. Solenoid Valve Circuit and Code
https://www.bc-robotics.com/tutorials/controlling-a-solenoid-valve-with-arduino/