Team #8 Documentation Autonomous Plant Care

Table of Contents

Introduction

What is the concept behind your project? What problem are you solving, or what product are you creating? Tell anyone reading this or using the product what they're getting into.

This project is a device for a plant that can regulate many parameters of its environment including soil saturation levels and shade based on the imputed needs of the plant. This project saves water as it can sometimes be hard to know how much your plants need, and provides water to your plants when you are not present to manually tend to them. The basis of this device is so farmers and gardeners alike can have an autonomous system that cares for their plants when they cannot. Strict regulation of parameters such as shade and soil moistness with this device could also open up possibilities for plants to be grown in regions previously uninhabitable for them, such as tomatoes in the desert or mangos in the chaparral.

Materials

- Soil Moisture Sensor
- 1/2" Solenoid Valve
- Servo Motor
- Photoresistor
- Tip120 transistor
- 1n4001 diode
- 9V battery
- Arduino Uno
- Wires
- Protoboard
- Soldering Iron
- ½" Faucet Connector
- Computer
- 3D printer

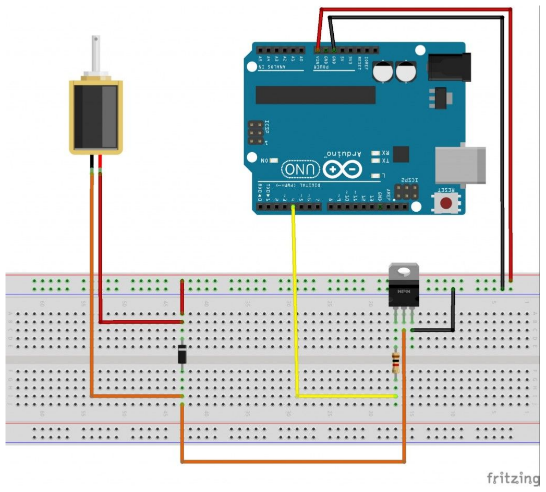
- Lantana camara plant
- Clay Pot
- Dirt
- Empty liter bottle

Instructions

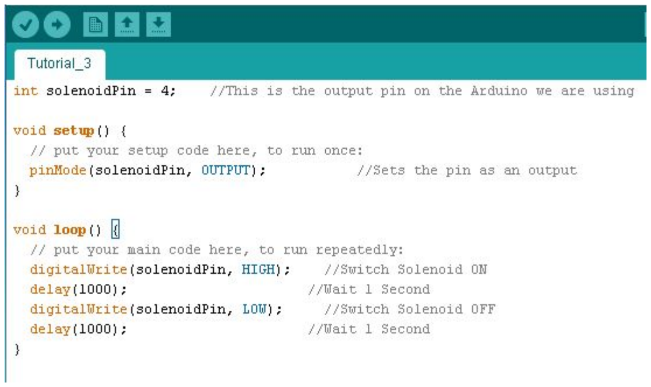
Testing Components

Solenoid Circuit:

Copy this schematic onto a breadboard



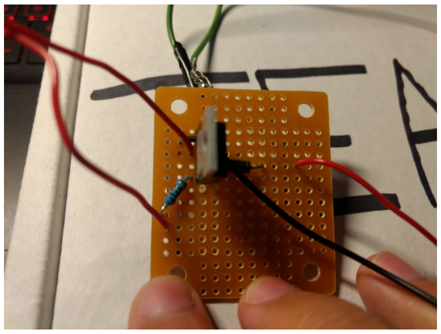
Upload this code onto an Arduino UNO for testing



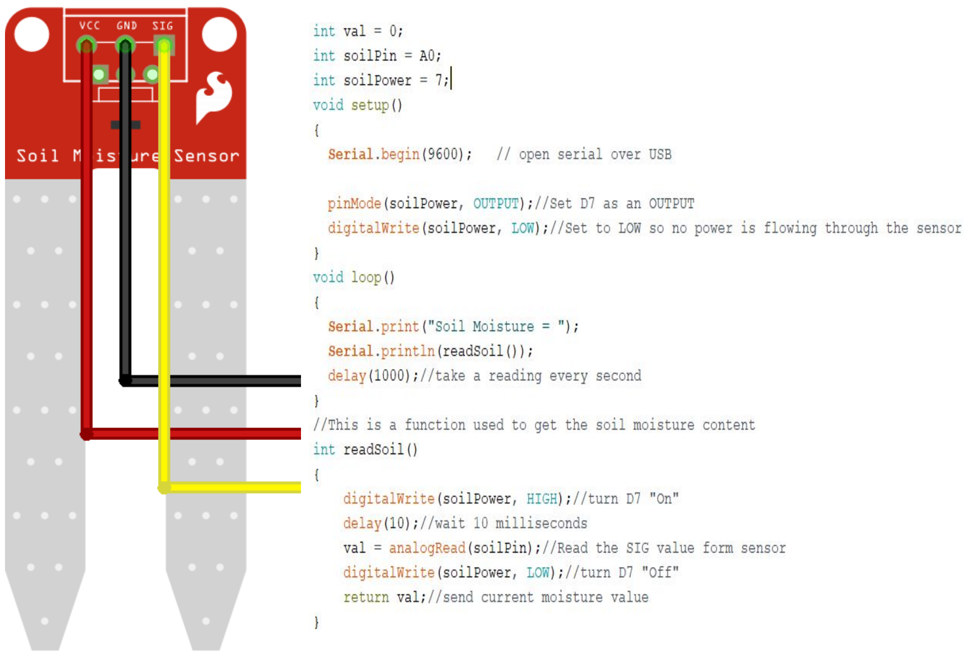
```
int solenoidPin = 4;    //This is the output pin on the Arduino we are using

void setup() {
  // put your setup code here, to run once:
  pinMode(solenoidPin, OUTPUT);           //Sets the pin as an output
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(solenoidPin, HIGH);    //Switch Solenoid ON
  delay(1000);                      //Wait 1 Second
  digitalWrite(solenoidPin, LOW);     //Switch Solenoid OFF
  delay(1000);                      //Wait 1 Second
}
```

Notes: *Make sure the negative side (silver stripe) of the diode faces the power side of the circuit.
*The solenoid is non-polarized so it doesn't matter how you wire it.
*A 9V battery will need to be attached in addition to this schematic to properly power the solenoid as the voltage of the Arduino is not enough.
*Simply attach the wire going from the emitter pin on the transistor to both the negative end of the battery and the ground of the Arduino, and attach the positive battery wire to the negative side of the diode.

*After testing the components for the solenoid, solder the components onto a protoboard, as shown on image below. Make sure to clip excess wires and remove extra solder.



Soil Moisture Sensor:



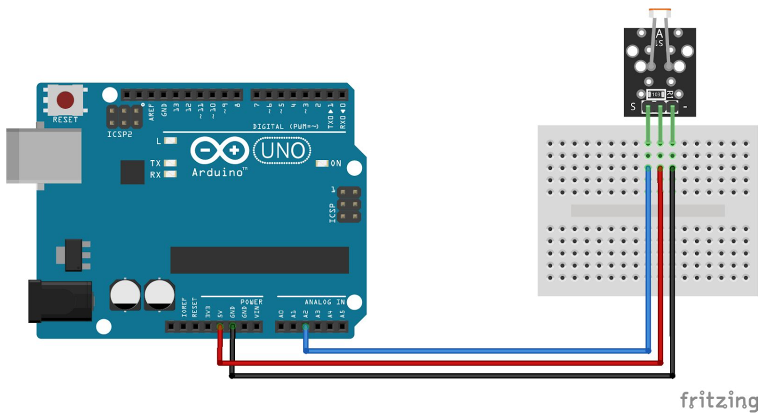
```
int val = 0;
int soilPin = A0;
int soilPower = 7;
void setup()
{
  Serial.begin(9600);   // open serial over USB

  pinMode(soilPower, OUTPUT);//Set D7 as an OUTPUT
  digitalWrite(soilPower, LOW);//Set to LOW so no power is flowing through the sensor
}
void loop()
{
  Serial.print("Soil Moisture = ");
  Serial.println(readSoil());
  delay(1000);//take a reading every second
}
//This is a function used to get the soil moisture content
int readSoil()
{
    digitalWrite(soilPower, HIGH);//turn D7 "On"
    delay(10);//wait 10 milliseconds
    val = analogRead(soilPin);//Read the SIG value form sensor
    digitalWrite(soilPower, LOW);//turn D7 "Off"
    return val;//send current moisture value
}
```

*Wire the VCC or power pin of the sensor to a digital pin of your choosing and wire the GND pin to ground.

*Wire the Signal pin to a analog pin of your choosing

*Copy and paste the test code shown above, replacing soilPin and soilPower variables to the analog and digital pins you used respectively. Run the code and try to figure out the value that represents the how dry you want the soil when you water it.

Photoresistor:

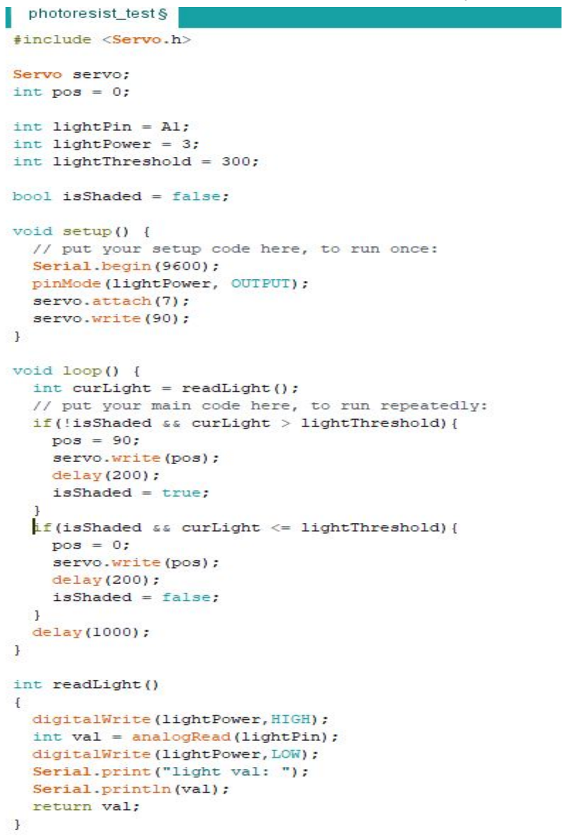


*Wire the Signal pin to an analog pin.
*Wire the positive (middle) pin to a digital pin of your choosing, instead of 5V pin as shown in the diagram.
*Wire the negative pin to the ground.

Servo Motor:

Wire the signal (orange) wire to a digital pin of your choosing and the positive (red) wire to the 5V pin. Wire the negative (brown) wire to the ground.

This code can be used to test both the photoresistor and the servo motor.



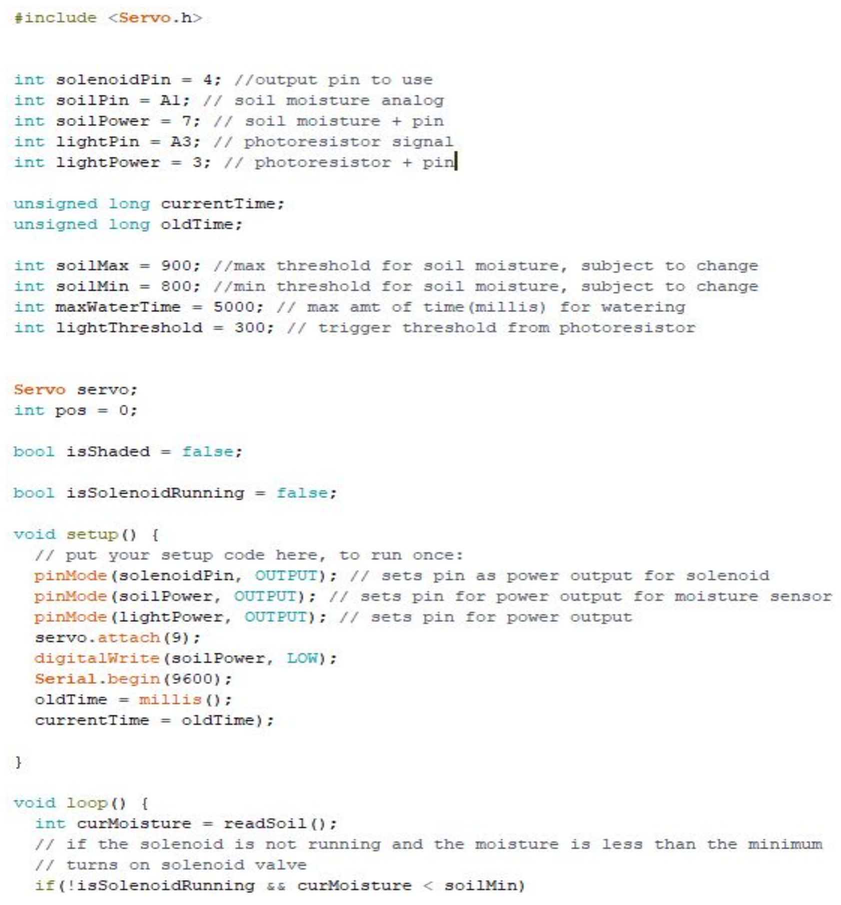
photoresist_test§

```
#include <Servo.h>

Servo servo;
int pos = 0;

int lightPin = A1;
int lightPower = 3;
int lightThreshold = 300;

bool isShaded = false;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(lightPower, OUTPUT);
  servo.attach(7);
  servo.write(90);
}

void loop() {
  int curLight = readLight();
  // put your main code here, to run repeatedly:
  if(!isShaded && curLight > lightThreshold){
    pos = 90;
    servo.write(pos);
    delay(200);
    isShaded = true;
  }
  if(isShaded && curLight <= lightThreshold){
    pos = 0;
    servo.write(pos);
    delay(200);
    isShaded = false;
  }
  delay(1000);
}

int readLight()
{
  digitalWrite(lightPower,HIGH);
  int val = analogRead(lightPin);
  digitalWrite(lightPower,LOW);
  Serial.print("light val: ");
  Serial.println(val);
  return val;
}
```

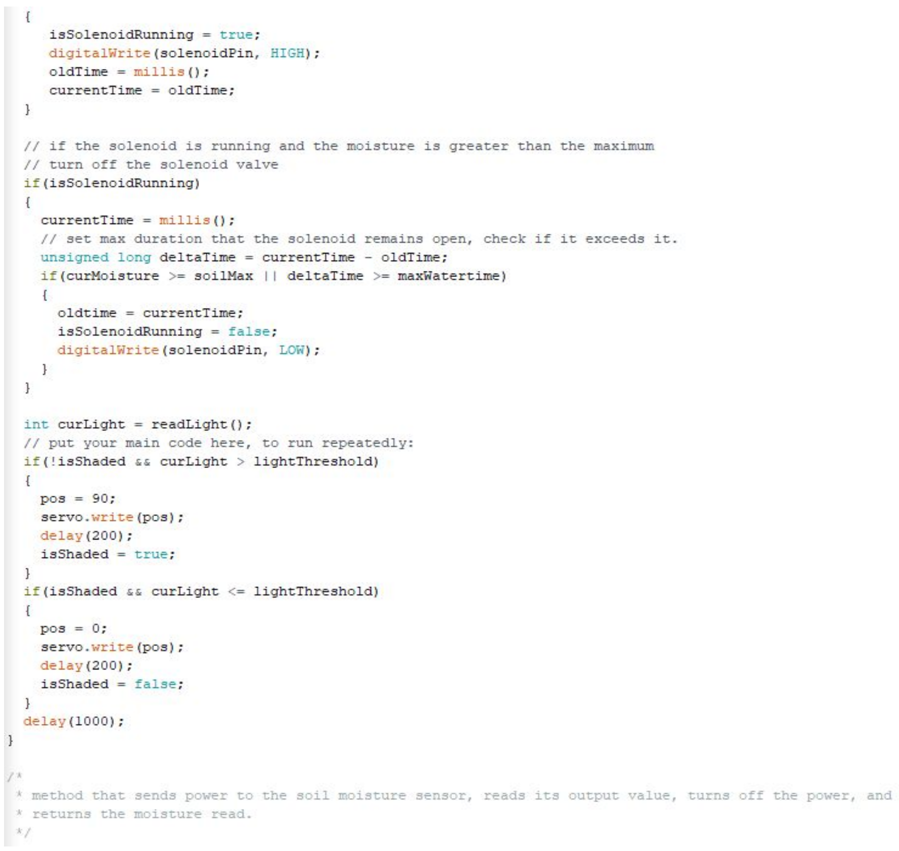
Where the variable lightPin is the analog pin you plugged the photoresistor in, lightPower is the digital pin you plugged the photoresistor in, and where servo.attach(#) is the pin you attached the signal of the servo motor. This test code spins the servo motor depending on whether light is touching the photoresistor or not.

Putting it All Together

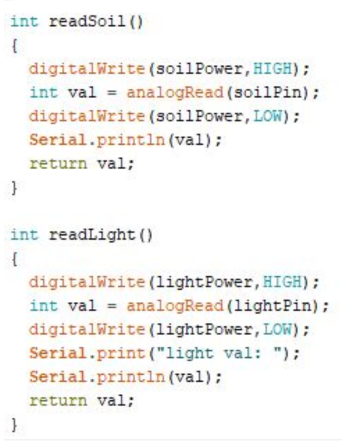
Now that each of the parts have been individually tested and made sure they worked, putting them all together is fairly simple. Solder the ground of the soil moisture sensor to the ground of the solenoid circuit on the protoboard. Then connect all the pins together to the Arduino. Note that any pins going to analog or digital pins can go to any of your choosing, just make sure to change the code accordingly.



```
#include <Servo.h>


int solenoidPin = 4; //output pin to use
int soilPin = A1; // soil moisture analog
int soilPower = 7; // soil moisture + pin
int lightPin = A3; // photoresistor signal
int lightPower = 3; // photoresistor + pin

unsigned long currentTime;
unsigned long oldTime;

int soilMax = 900; //max threshold for soil moisture, subject to change
int soilMin = 800; //min threshold for soil moisture, subject to change
int maxWaterTime = 5000; // max amt of time(millis) for watering
int lightThreshold = 300; // trigger threshold from photoresistor


Servo servo;
int pos = 0;

bool isShaded = false;

bool isSolenoidRunning = false;

void setup() {
  // put your setup code here, to run once:
  pinMode(solenoidPin, OUTPUT); // sets pin as power output for solenoid
  pinMode(soilPower, OUTPUT); // sets pin for power output for moisture sensor
  pinMode(lightPower, OUTPUT); // sets pin for power output
  servo.attach(9);
  digitalWrite(soilPower, LOW);
  Serial.begin(9600);
  oldTime = millis();
  currentTime = oldTime);

}

void loop() {
  int curMoisture = readSoil();
  // if the solenoid is not running and the moisture is less than the minimum
  // turns on solenoid valve
  if(!isSolenoidRunning && curMoisture < soilMin)
```

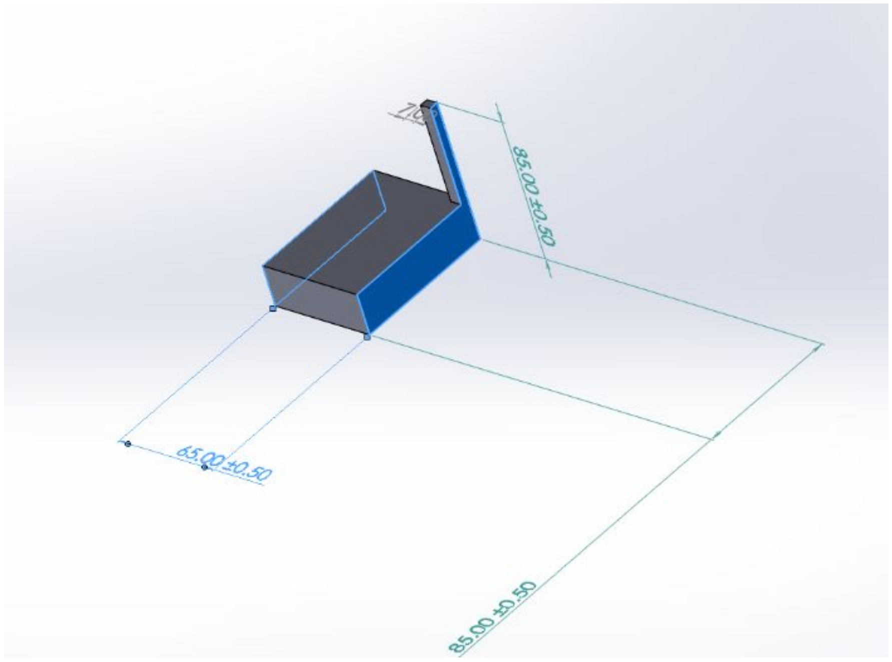


```
  {
     isSolenoidRunning = true;
     digitalWrite(solenoidPin, HIGH);
     oldTime = millis();
     currentTime = oldTime;
  }

  // if the solenoid is running and the moisture is greater than the maximum
  // turn off the solenoid valve
  if(isSolenoidRunning)
  {
    currentTime = millis();
    // set max duration that the solenoid remains open, check if it exceeds it.
    unsigned long deltaTime = currentTime - oldTime;
    if(curMoisture >= soilMax || deltaTime >= maxWatertime)
    {
      oldtime = currentTime;
      isSolenoidRunning = false;
      digitalWrite(solenoidPin, LOW);
    }
  }

  int curLight = readLight();
  // put your main code here, to run repeatedly:
  if(!isShaded && curLight > lightThreshold)
  {
    pos = 90;
    servo.write(pos);
    delay(200);
    isShaded = true;
  }
  if(isShaded && curLight <= lightThreshold)
  {
    pos = 0;
    servo.write(pos);
    delay(200);
    isShaded = false;
  }
  delay(1000);
}

/*
 * method that sends power to the soil moisture sensor, reads its output value, turns off the power, and
 * returns the moisture read.
 */
```



```
int readSoil()
{
  digitalWrite(soilPower,HIGH);
  int val = analogRead(soilPin);
  digitalWrite(soilPower,LOW);
  Serial.println(val);
  return val;
}

int readLight()
{
  digitalWrite(lightPower,HIGH);
  int val = analogRead(lightPin);
  digitalWrite(lightPower,LOW);
  Serial.print("light val: ");
  Serial.println(val);
  return val;
}
```

We designed a 3D model (shown below) for the shade to hot glue onto the servo motor. You can make your own design with any material as long as it shades the whole pot.



Conclusion

If you had more time, energy, funding, or resources, what would you change or improve upon? What was your greatest obstacle and proudest moment?

If we had more time, we could have developed a fully-fleshed out android app, since convenience and constant monitoring were the original goals of our project, regarding autonomous care. Also, we would have tried to devise a much more sophisticated method of specifying how much water to put in the soil if we had more time. At the moment, we did not have enough time to calibrate the soil moisture sensor and check what level of water would be considered “dry” and we didn’t know how to determine how much water to put in. So, we just set the arduino to put in a certain amount of water for a set amount of time maybe we shouldnt say this? we can try to do this today for the dryness threshold

Our shade device was considerably downgraded due to the lack of funds we were provided. Our original prototypes included solar powered shade outfitted with solar cells to power the shade. Since solar cells are considerably pricy, we opted out and decided to go with the more simple option of a simple servo motor. The movement of our solar powered shade would have been triggered in a similar fashion to the micro servo, with input from the photoresistor and a threshold of light per day allotted for each plant type.

With more time, we would have done long-term (5-8 month) trials on our device in a real gardening or crop setting, to truly gauge how successful our device is in tending to plants. This would also provide us with crucial data about the longevity of our product in an outside setting, for water damage and short-circuiting are common issues seen with electrical devices. After minor fixes on our device design and app work--and further exploration into plant nutrient distribution--we believe that our autonomous plant care device could be an efficient and conservational method for crop growth.

Our greatest obstacle was making the circuit for the solenoid valve. We ran into a surprising amount of issues trying to hook it up, especially with connecting it to an external battery as a power source.

References

This doesn't have to be fancy or follow any particular format. Give a shout-out to anything or anyone who helped you do it, and a link if possible. Make sure to indicate what they helped you do, what you used it for, etc.

https://www.bc-robotics.com/tutorials/controlling-a-solenoid-valve-with-arduino/

- This link gave us the schematic we used to make the solenoid circuit to test, and we stared at that schematic for quite a while