The Constant Gardener

Jose Jayma
The Constant Gardener
EEL5666: Intelligent Machine Design Laboratory
A. Antonio Arroyo, PhD
Eric M. Schwartz, PhD

Abstract: This document presents the design specifications and implementation of my robot, The Constant Gardener, a robot that roams about on a table, avoiding the table edge and finding plant pots. Once a pot is found, the moisture level of the soil is sampled and The Constant Gardener determines whether to water the plant.

Table of Contents

Executive Summary

The Constant Gardener is a robot that, in simple terms, waters potted plants on top of a table. The behaviors it exhibits includes preventing itself from falling off of the table, detecting a pot and centering about it, actuating an arm which mounts a moisture sensor and grabbing the value and determining whether to water the plant based on the output of that moisture sensor.

In order to prevent itself from falling off of the table, whether round or rectangular/square, The Constant Gardener uses four IR sensors mounted on the left, right, front-left and front-right. The left and right IR sensors help to keep the platform from approaching a table edge and the two front-mounted IRs prevent the platform from falling off of a table in direct, forward moving approach.

The two front-mounted IRs are also used to detect when a pot has been found. They use an edge-detection algorithm to center about the pot, which makes the next behavior simpler. After the pot has been centered, the moisture sensor arm is dipped into the soil of the pot to determine whether the soil is moist or not.

Once moisture level has been determined, there are two possible behaviors. The first happens when the moisture sensor determines that the soil is dry/has not been watered. In this case, the windshield washer pump blasts water into the plot for a period of 1.25 seconds, and then the moisture sensor arm moves out of the way to allow the robot to drive away. The other possible behavior is that the soil moisture sensor determines the plant to have already been watered. In this case, The Constant Gardener simply flashes its LEDs red and drives away.

Introduction:

There have been plant watering robots in previous semesters of EEL 5666. WaterBot, a line-following robot, is an impressive example which used RFID tags to recognize plant pots and a pump watering system for water delivery. I am going to attempt to let my plant watering robot, The Constant Gardener, roam freely and find the plant pots and deliver water using a windshield washer pump. The Constant Gardener will also be able to determine whether a plant needs watering based on the output of a soil moisture sensor.

Integrated System:

The robot is a rectangular shaped, two wheeled robot driven by two servos. The servos are placed along the long sides near the center of each side. This allows for ease of control of turning. The robot has two platforms – a lower platform for mounting control electronics (the microcontroller) and batteries and an upper platform which will mount the water reservoir and the pump, as well as the arm for the moisture sensor. In simpler terms, there is a wet platform and a dry platform, for safety of electronics.

IR sensors are placed along the side walls and on the front of the platform for the purpose of avoiding obstacles and prevent falling off of a table, a probable consequence of free roaming. The front mounted IRs sense when a pot is near and uses edge detection to center about that pot.

Once a pot is found and centered, the moisture sensor arm is activated and dips the end of the moisture sensor into the soil. Based on the moisture level of the soil, the windshield washer pump is either activated if the soil is dry and waters the plant for approximately 1.25 seconds, or the robot displays that the soil is moist (via the LCD and LEDs) and drives away.

Mobile Platform:

The driving system will consist of two modified servos. Two casters are placed at the center-front and center-rear of the platform in order to stabilize the platform and provide smooth driving operation. The servos are placed at the two long sides of the rectangular platform and are at the center of both sides. The result is predictable turning that behaves slightly circular. Servos are being used because they are relatively inexpensive and simple to deploy. They are slower than motors, which is a positive aspect for a robot that carries water.

The lower platform houses the microcontroller board as well as the batteries. This shields them from any water that is accidentally sprayed back onto the robot. The upper platform can slide out to allow access for servicing the microcontroller and associated electronics.

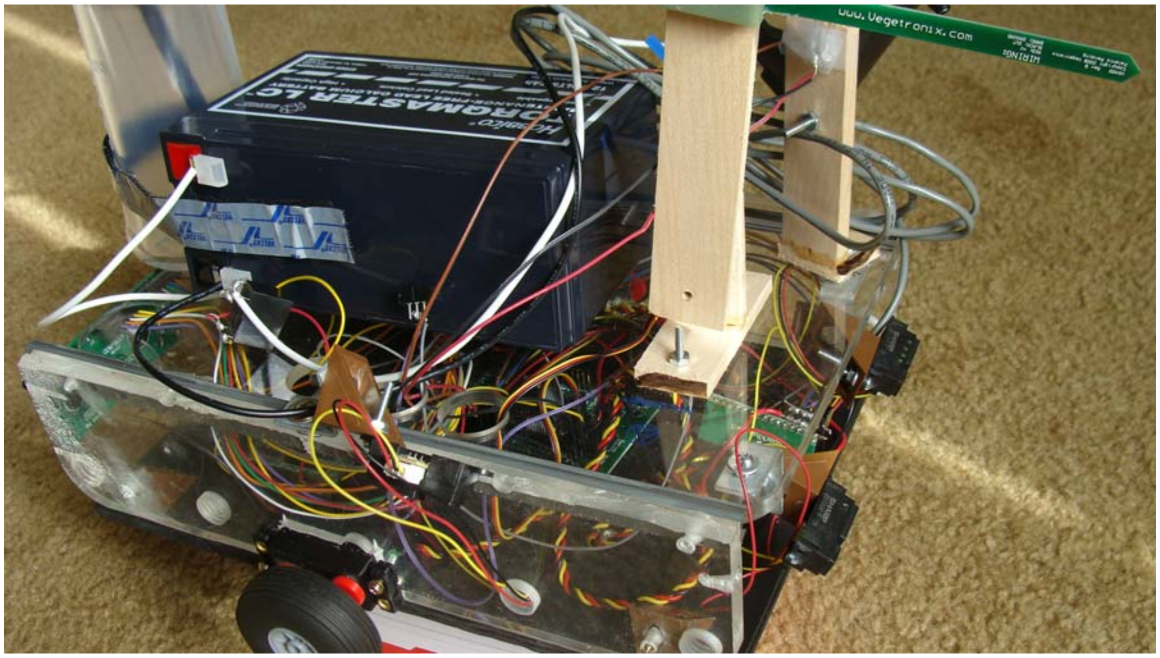


Figure 1 - A 3/4 view of the platform from the right side.

Actuation:

There are two mechanical features of The Constant Gardener – the watering mechanism and the soil moisture sensor arm.

The watering mechanism consists of a water reservoir and a windshield washer pump. The windshield washer pump is a 12 V, fairly typical washer pump commonly found in automobiles. The water reservoir is an RC fuel tank that holds enough water for approximately eight blasts of water from the windshield washer pump. The blast period of the windshield washer pump lasts for 1.25 seconds, but can be easily changed within the code.

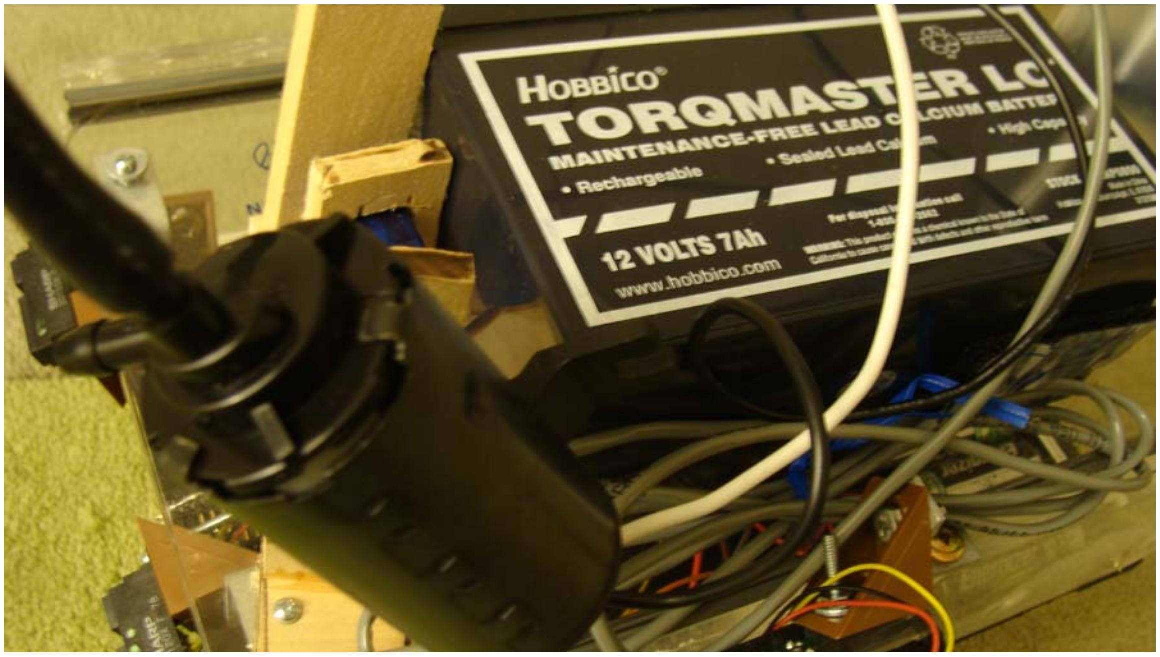
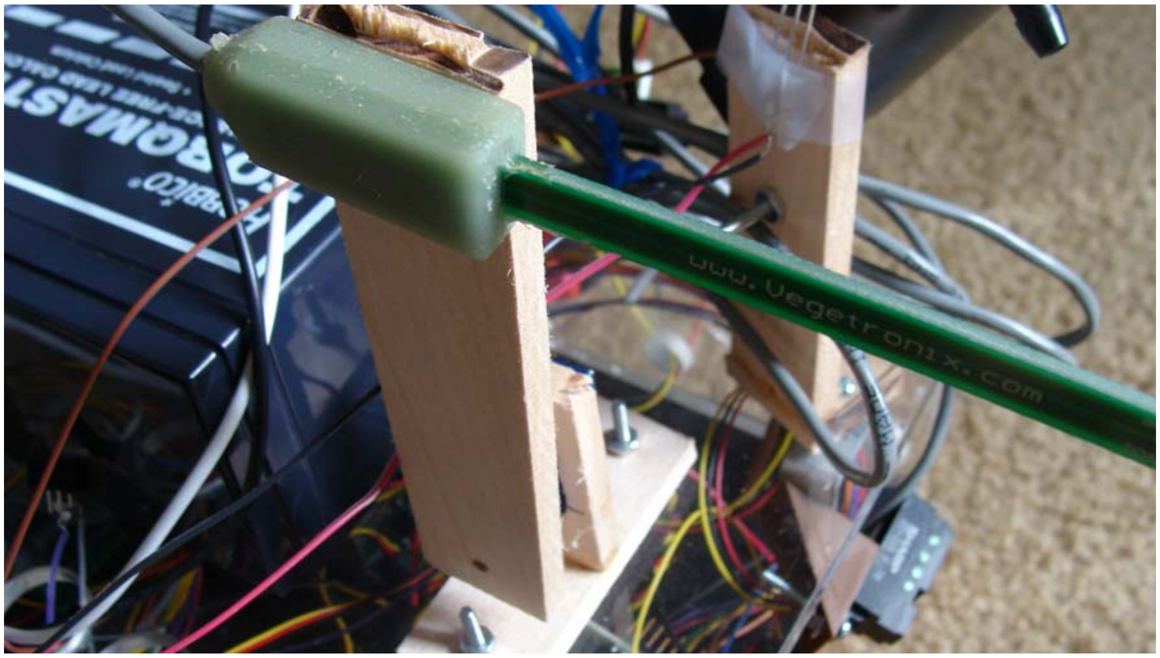


Figure 2 - The windshield washer pump is powered by a gigantic and heavy 12 V, 7 Ah battery. The battery was inexpensive and worked well for my platform.

The arm for the soil moisture sensor is a feather-servo mounted wooden arm that is approximately four inches in length. At the end of the arm is the moisture sensor, mounted perpendicular to the arm. The length was chosen based on the centering behavior of the front IR sensors.



Sensors:

IR sensors will be used for obstacle avoidance and for avoiding falling off of the table/platform during demonstration. The TCS230 light frequency sensor will be used to detect the color of the pots and IR sensors will be used to approach the pots for watering. A moisture sensor will determine whether a plant needs to be watered or not. The following are the data collected for the sensors so far.

IR sensor data

Table and Figure 1 represents the sensor output value for each IR sensor when an object (my wallet) is placed at a measured distance.

Distance (in.)	**Left IR**	**Right IR**	**Back IR**	**Front IR**
1	157	156	157	155
1.5	142	137	127	139
2	117	107	103	115
2.5	94	96	88	95
3	81	77	74	80
3.5	69	70	65	70
4	63	60	61	62
4.5	53	54	51	55
5	48	45	48	51
5.5	45	43	44	47
6	41	40	41	42
6.5	36	37	39	40
7	36	33	35	36

Figure 3 - The output of each IR sensor when measuring an object set at various distances.

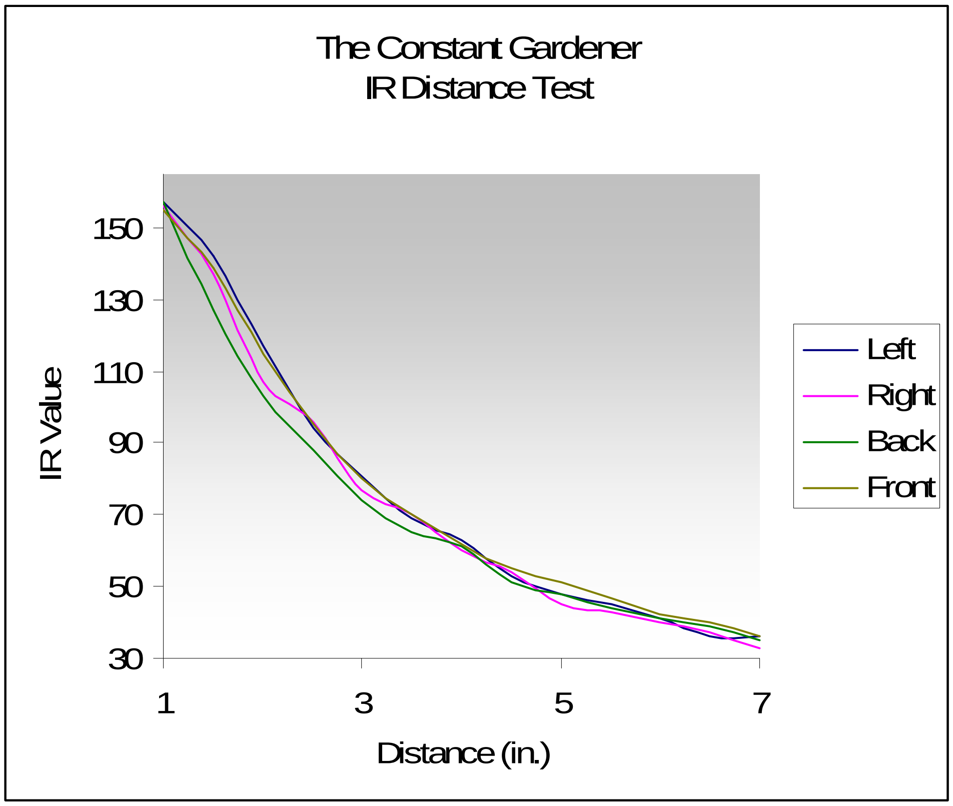


Figure 4- Under the lighting conditions of my apartment, this is the IR sensor output value when an object is placed at various distances.

The following tables represent the "safe" range for the robot, which means there is no obstacle and there is no table edge, under different lighting conditions in different locations. Table 1 is my apartment, Table 2 is the IMDL lab and Table 3 is the Harris Corp. Rotunda.

Position	**Safe Output Value**
Front-Left	140~152
Front-Right	127~140
Left	45~55
Right	65~73

Table 1 - The IR sensor values for the lighting conditions inside of my apartment, under fluorescent lighting.

Position	**Safe Output Value**
Front-Left	128~158
Front-Right	115~150
Left	35~70
Right	55~70

Table 2 - The IR sensor values for the lighting conditions inside of the IMDL lab.

Position	Safe Output Value
Front	TBD
Back	TBD
Left	35~105
Right	35~105

Table 3 - The IR sensor values for the lighting conditions on the circular table of the NEB Harris Corp. Rotunda.

Vegetronix Soil Moisture Sensor ***Special Sensor

The soil moisture sensor used is from a company called Vegetronix and the model number used is VG400. There exists a low voltage version for applications where the sensor will be left in the soil for extended periods of time. The application for my robot is an uneven sampling interval (it only samples when the pot is found), so the low voltage version is not necessary. The sensor is shown in the figure below.

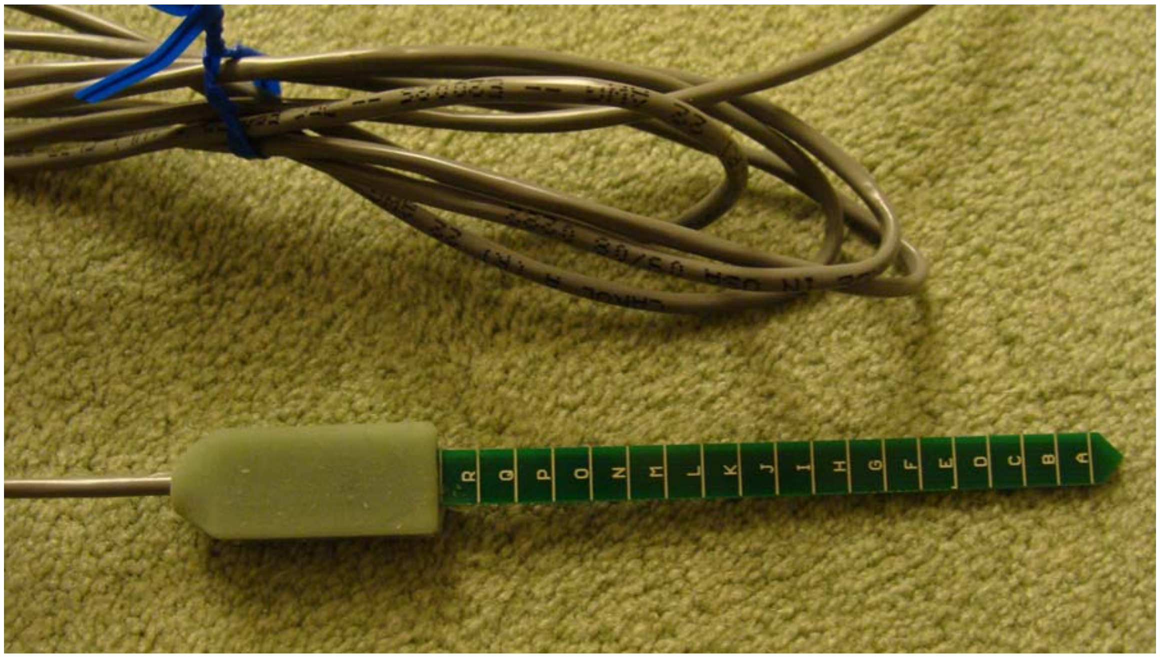


Figure 5 - The Vegetronix VG400 Soil Moisture Sensor.

The intended purpose of the VG400 is to output a voltage which can be converted into a Volumetric Water Content, which is the ratio of the volume of water to the volume of material (in this case, soil). Vegetronix provides an equation for converting the voltage output of the sensor to VWC, VWC= V*21.186 -10.381. Note that the equation is linear, which means that conversion to VWC is unnecessary if determination of "wetness" is all that is required. The following graph is provided by Vegetronix for the VG400.

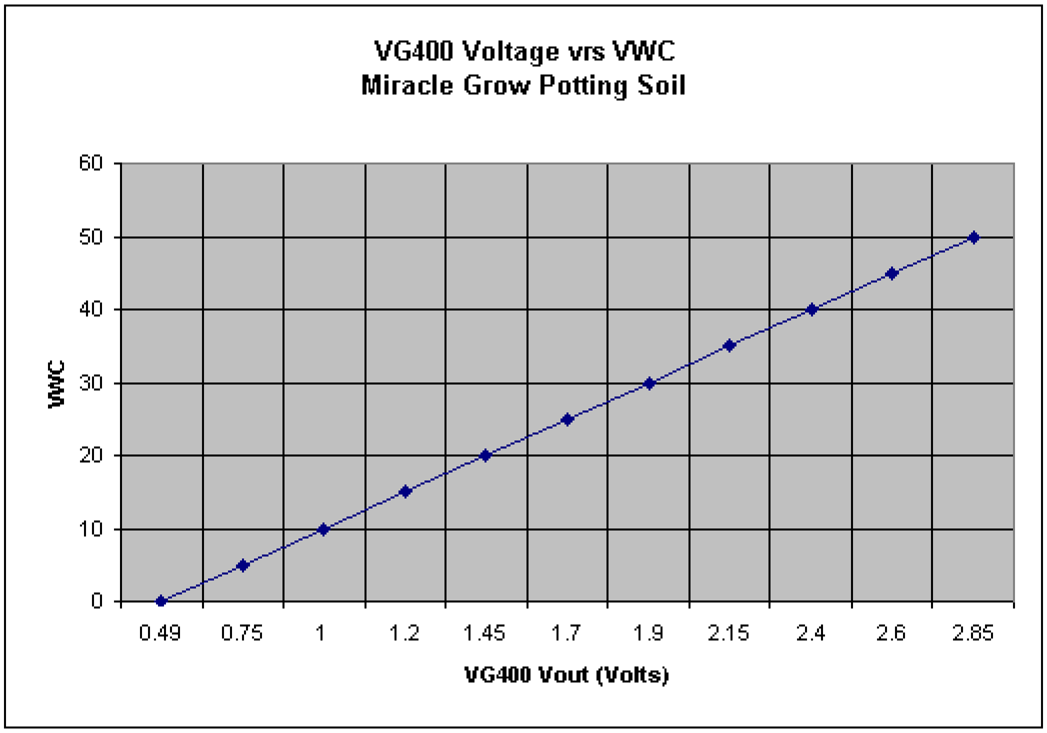


Figure 6 - Image courtesy of Vegetronix. http://www.vegetronix.com/Products/VG400/

Three tests were performed to determine the output from the VG400 Soil Moisture Sensor. First, the output when the probe is in various environments is tabulated. Second, the output when the probe is placed at various depths in *very* moist soil, soil where the upper layer is moist and a cup of water is tabulated and graphed. The probe has markings for various depths, starting at point A at 5 mm to point R at 90 mm of depth. The third test is designed to observe any difference in the output of the sensor in different water temperatures.

Environment	**Output**
Air	16~18
Palm of my hand (open)	40
Palm of my hand (closed)	115~122
Dry Soil (fully immersed)	28
Damp Paper Towel	63~151 (scale by pressure)
In Air after Damp Paper Towel	16~34

Table 4 - Test 1. The output in various environments is tabulated.

.

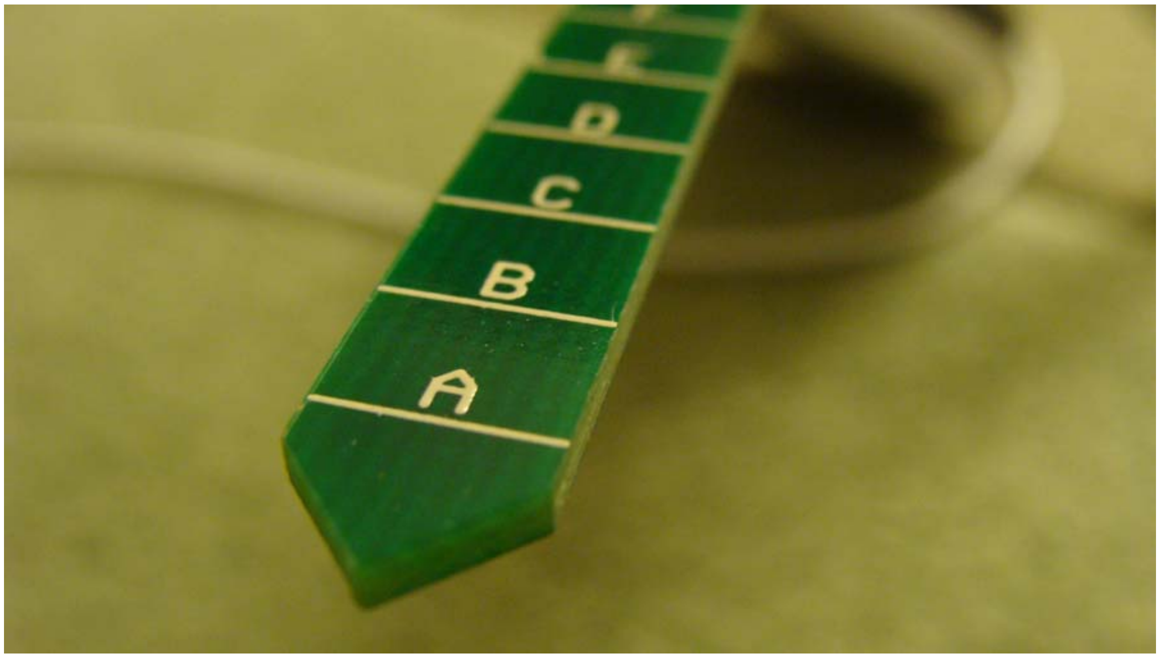


Figure 7 - Depth guidelines and letter markings from A to Z (5 mm to 90 mm)

Depth (mm)	Pure Water	Completely Moist Soil	Upper Layer Moist Soil
A – 5	21	27	21
B – 10	34	37	27
C – 15	40	46	35
D – 20	49	50	39
E – 25	52	54	42
F – 30	60	61	43
G – 35	68	68	46
H – 40	71	74	46
I – 45	80	81	47
J – 50	86	87	49
K – 55	90	91	51
L – 60	96	98	51
M – 65	99	99	57
N – 70	104	104	59
O – 75	110	110	60
P – 80	115	115	62
Q – 85	119	118	63
R - 90	127	124	65

Table 5 - Test 2. The outputs in three environments at various depths.

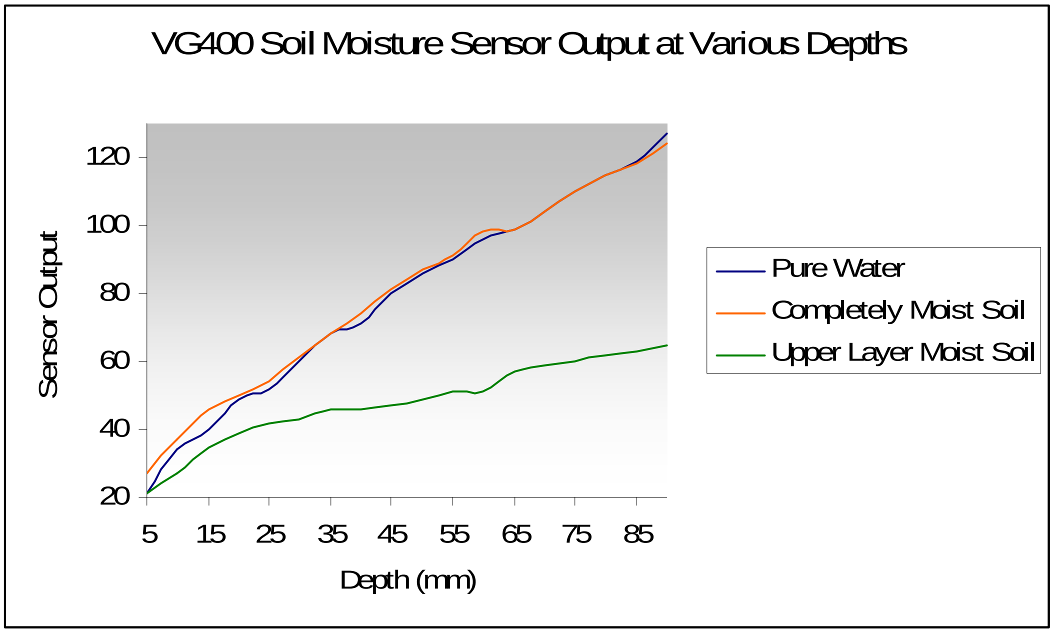


Figure 8 - The output of the VG400 at various depths. Note the similarities between pure water and completely moist soil.

Depth (mm)	Cold	Standard Tap	Hot
B – 10	31	32	33
Q – 85	113	113	113

Table 6 The output of the temperature test. The three temperature environments produced similar or exact output from the sensor. We can conclude that temperature does not need to be considered when taking readings.

Behaviors:

Obstacle avoidance and staying on top of a table are the basic behaviors. These two behaviors are implemented by the four IR sensors and the tabulated output for their "safe" distances. The left and front-left IRs detect the table edge falloff for the left of the robot and the right and front-right IRs do the same for the right of the platform. This behavior has been successfully implemented and it works very well on both circular and square/rectangular tables, as long as they are at least 15'' wide.

The next behavior is to detect the pot with either the front-left or front-right IRs and use both for edge-detection to center the pot. Once the pot is centered, the next behavior is to use the moisture sensor arm to dip the moisture sensor into the soil. This behavior spawns one of the two following behaviors: either a) water the plant if the soil moisture sensor output senses a low moisture or b) turn the LEDs red when the robot senses that the soil is already moist and turn and drive away.

The behaviors are implemented in the software included in the appendix of this document.

Conclusion:

For most demonstrations performed with The Constant Gardener, the robot performed very well and the audience was generally pleased with the watering mechanism. The first demo day for IMDL proved to be very difficult because The Constant Gardener began the day off by spinning in circles. Then, it began to show behaviors that I had corrected for during development, such as stabbing the plant base with the soil moisture sensor.

An improvement can be made by utilizing real pot detection. I had intended to use color sensors but they were not very cooperative. I believe the CMU cam can also work well, but at an increased cost.

Appendix:

The following is the code used to implement the behaviors of The Constant Gardener.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include "PVR_Servos.h"
#include "sleep.h"
#include "LCD.h"
#include "ADC.h"

//IR's are left=zero, right=one, two=frontLeft, three=frontRight
//Servo1 is right, Servo2 is left, Servo6 is arm...

int main(void) {

//Initialize everything
initServo();
initSleep();
lcdInit();
initADC();
moveServo6(0);

//Mux, Relay, LEDs Init
DDRB |= 0b00011111;
DDRD |= 0b10000000;
DDRA |= 0b11000011;

//Tell the user what you are doing...
lcdString("Constant");
lcdGoto(1,0);
lcdString("Gardener Proto");
PORTA |= 0b10000001;      //turn the LEDs red
PORTA &= ~(0b01111110);
```

```
ms_sleep(5000);
PORTA |= 0b01000010;      //turn the LEDs green
PORTA &= ~(0b10111101);
ms_sleep(5000);
PORTA |= 0b10000001;      //turn the LEDs red again...
PORTA &= ~(0b01111110);
lcdClear();

//FORMER COLOR SENSOR CODE.
//FOR THE COLOR SENSORS
//S0 = A7, S1 = A6, OE3 = A5
//S3 = B7, S2 = B6, OUT = B5

//For GREEN, B6 and B7 HIGH

//init data direction registers
//DDRA |= 0b11100000;
//DDRB |= 0b11000000;
//DDRB &= ~(0b00100000);
//set data
//PORTA |= 0b11000000;
//PORTA &= ~(0b00100000);
//PORTB |= 0b00000000;
//PORTB &= ~(0b11000000);


PORTB |= 0b00000000;
PORTB &= ~(0b00011111);
while (1) {
        if ( (adcZero()>35 && adcZero()<105) && (adcOne()>35 && adcOne()<105)
&& (adcTwo()>40 && adcTwo()<105) && (adcThree()>40 && adcThree()<105) ) {
                lcdString("Clear");
                ms_sleep(250);
                moveServo1(50);
                moveServo2(-50);
                lcdClear();
        }

        else if ( (adcZero()<35 || adcZero()>105) && (adcOne()>35 &&
adcOne()<105) && (adcTwo()>40 && adcTwo()<105) && (adcThree()>40 &&
adcThree()<105) ) {
                lcdString("Left danger!");
                moveServo1(-50);
                moveServo2(-50);
                ms_sleep(1500);
                lcdClear();
```

```
	}

	else if ( (adcZero()>35 && adcZero()<105) && (adcOne()<35 || adcOne()>105) && (adcTwo()>40 && adcTwo()<105) && (adcThree()>40 && adcThree()<105) ) {
		lcdString("Right danger!");
		moveServo1(50);
		moveServo2(50);
		ms_sleep(1500);
		lcdClear();
	}

	else if ( (adcZero()>35 && adcZero()<105) && (adcOne()>35 && adcOne()<105) && (adcTwo()<40) && (adcThree()>45 && adcThree()<105) ) {
		lcdString("FL danger!");
		moveServo1(-50);
		moveServo2(-50);
		ms_sleep(1500);
		lcdClear();
	}

	else if ( (adcZero()>35 && adcZero()<105) && (adcOne()>35 && adcOne()<105) && (adcTwo()>40 && adcTwo()<105) && (adcThree()<40) ) {
		lcdString("FR danger!");
		moveServo1(50);
		moveServo2(50);
		ms_sleep(1500);
		lcdClear();
	}

	else if ( (adcTwo()>107) ) {
		PORTA |= 0b01000010;
		PORTA &= ~(0b10111101);
		lcdString("FL Found!");
		moveServo1(0);
		moveServo2(0);
		ms_sleep(1500);
		lcdClear();
		//while (adcThree()<120) {
		//	moveServo1(10);
		//	moveServo2(10);
		//	ms_sleep(100);
		//}
		lcdString("Centered!");
		moveServo1(10);
		moveServo2(10);
		ms_sleep(400);
		moveServo1(0);
		moveServo2(0);
		moveServo6(80);
		ms_sleep(5000);
		lcdClear();
		PORTB |= 0b00001000;
		PORTB &= ~(0b00010111);
		ms_sleep(800);
		if (adcThree()<30) {
			PORTA |= 0b01000010;
			PORTA &= ~(0b10111101);
			PORTD |= 0b10000000;
			ms_sleep(1200);
			PORTD |= 0b00000000;
			PORTD &= ~(0b10000000);
		}
		else {
			lcdString("Not Watering");
			PORTA |= 0b10000000;
			PORTA &= ~(0b01111111);
			ms_sleep(1000);
		}
		ms_sleep(5000);
		lcdClear();
		moveServo6(0);
		ms_sleep(1000);
		PORTB |= 0b00000000;
		PORTB &= ~(0b00011111);
		lcdString("Moving");
		moveServo1(-100);
		moveServo2(-100);
		ms_sleep(2000);
	}

	else if ( (adcThree()>107) ) {
		PORTA |= 0b01000010;
		PORTA &= ~(0b10111101);
		lcdString("FR Found!");
		moveServo1(0);
		moveServo2(0);
		ms_sleep(1500);
		lcdClear();
		while (adcTwo()<105) {
			moveServo1(-10);
			moveServo2(-10);
```

```
                        ms_sleep(100);
                }
                lcdString("Centered!");
                moveServo1(0);
                moveServo2(0);
                moveServo6(80);
                ms_sleep(5000);
                lcdClear();
                PORTB |= 0b00001000;
                PORTB &= ~(0b00010111);
                ms_sleep(800);
                if (adcThree()<30) {
                        PORTA |= 0b01000010;
                        PORTA &= ~(0b10111101);
                        PORTD |= 0b10000000;
                        ms_sleep(1200);
                        PORTD |= 0b00000000;
                        PORTD &= ~(0b10000000);
                }
                else {
                        lcdString("Not Watering");
                        PORTA |= 0b10000001;
                        PORTA &= ~(0b01111110);
                        ms_sleep(1000);
                }
                ms_sleep(5000);
                lcdClear();
                moveServo6(0);
                ms_sleep(1000);
                PORTB |= 0b00000000;
                PORTB &= ~(0b00011111);
                lcdString("Moving");
                moveServo1(-100);
                moveServo2(-100);
                ms_sleep(2000);
        }

        else {
                moveServo1(-50);
                moveServo2(-50);
        }

}

return 0;
}
```