“Smart Plant Pot”

On the final week of the module, I focussed most of my time on the assembly of the prototype. Since most of the wooden components were manufactured in week 3, the only task left is to glue them together. Below is the pot’s design that I decided at the end of the designing process:

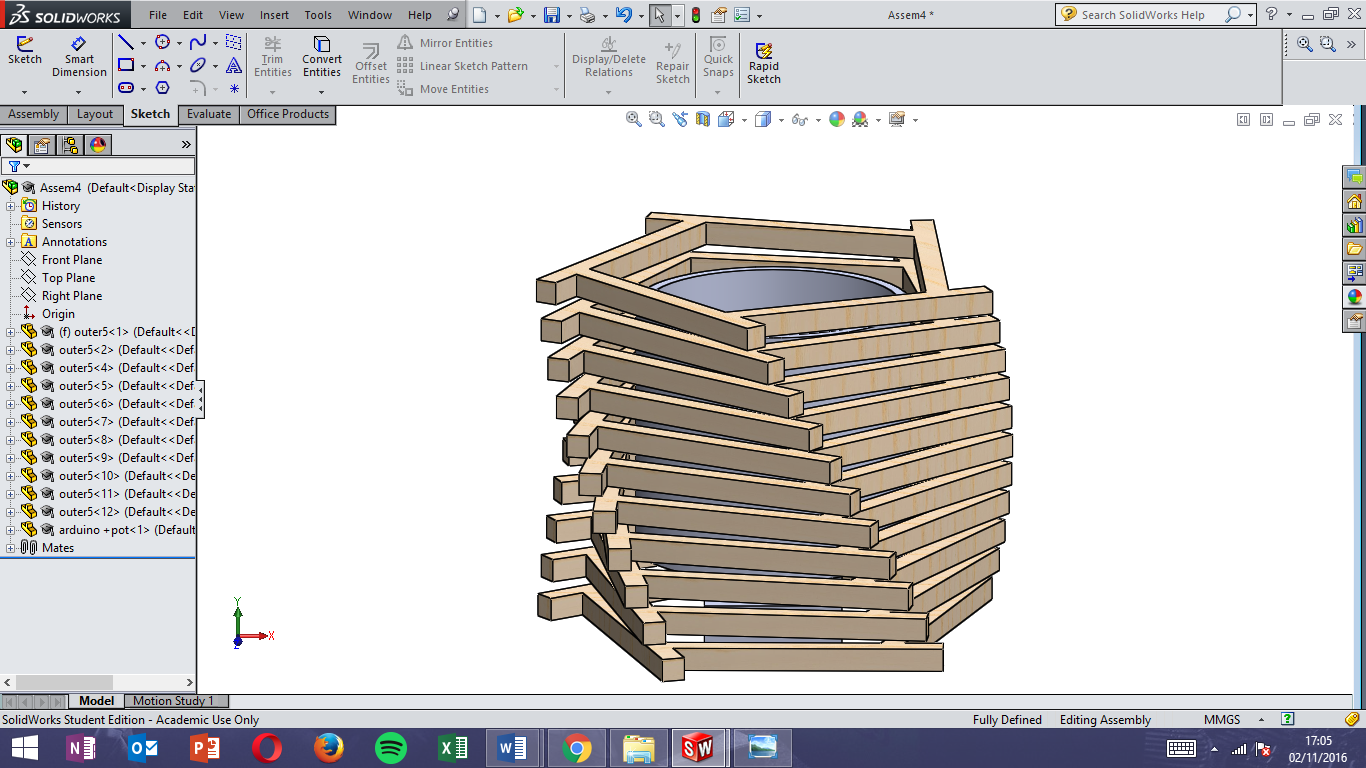


Figure 1: Prototype and CAD design

As the picture above shows, it’s like the CAD model developed from week 3. The difficult part in the manufacturing process of this model was the construction of the spiral shape. In the CAD model, the spiral’s twist is two degrees. However, in the actual prototype, the spiral’s twist was not followed due to the time required to rotate it at exactly two degrees. Instead, the angle was made through visual adjustments and estimations.



Figure 2: Paper Prototype

On the final week, I changed the Arduino program for my project to accommodate the 8x8 Neopixel RGB LED I received for my final prototype. Initially I was only using a single RGB LED to demonstrate my first prototype (shown above). The single RGB LED was bright enough to illuminate the paper model and put across the idea to other people. However, this is not enough to light up the final prototype, so more LED was required for the final prototype.



Figure 3: Final Prototype, LED colour changes represent different scenarios

As the above image in figure 3 shows, using sixty-four LEDs is more than enough to illuminate the device (even with the indoor lights on). After completing the Smart Pot Prototype, I then took the remainder of the week to prepare the video and other deliverables.

For the video, the initial plan was to control the light colour using a potentiometer. An arrow was going to be attached to the potentiometer. As the arrow is turned, along with the potentiometer, it will point to different weather conditions to represent the information from the MET Office’s online forecast. The program will then change the LED’s light colour using “if” statements to show the visual signals to the user.



Figure 4: Arrow Points to weather to represent MET Office Forecast

However, this idea was scrapped during the filming of the prototype because it slowed down the story’s process. Showing the manual control of the potentiometer may also cause some confusion on the storytelling of the device.

To amend this, I decided to make another program which would automatically control the LED’s colour using a designated time interval. The use of this method enabled me to convey the story at a simpler way, whilst including all relevant information.

At the end of this module I have generated two physical “Smart Plant Pot” prototypes. One made of paper (designed and made with minimal resources to and the other from wood with a more complex design), three different Arduino programs and three electrical circuits. All of which are useful for progressing the idea step by step.

# Final Arduino Program

#include <Adafruit\_NeoPixel.h>

#ifdef \_\_AVR\_\_

#include <avr/power.h>

#endif

#define PIN 6

// Parameter 1 = number of pixels in strip

// Parameter 2 = Arduino pin number (most are valid)

// Parameter 3 = pixel type flags, add together as needed:

// NEO\_KHZ800 800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)

// NEO\_KHZ400 400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers)

// NEO\_GRB Pixels are wired for GRB bitstream (most NeoPixel products)

// NEO\_RGB Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)

// NEO\_RGBW Pixels are wired for RGBW bitstream (NeoPixel RGBW products)

Adafruit\_NeoPixel strip = Adafruit\_NeoPixel(60, PIN, NEO\_GRB + NEO\_KHZ800);

// IMPORTANT: To reduce NeoPixel burnout risk, add 1000 uF capacitor across

// pixel power leads, add 300 - 500 Ohm resistor on first pixel's data input

// and minimize distance between Arduino and first pixel. Avoid connecting

// on a live circuit...if you must, connect GND first.

void setup() {

// This is for Trinket 5V 16MHz, you can remove these three lines if you are not using a Trinket

#if defined (\_\_AVR\_ATtiny85\_\_)

if (F\_CPU == 16000000) clock\_prescale\_set(clock\_div\_1);

#endif

// End of trinket special code

strip.begin();

strip.show(); // Initialize all pixels to 'off'

}

void loop() {

// Some example procedures showing how to display to the pixels:

delay(10000);

colorWipe(strip.Color(0, 50, 0), 200); // Green

delay(8000);

colorWipe(strip.Color(50, 0, 0), 50); // Red

delay(8000);

colorWipe(strip.Color(0, 0, 50), 50); // Blue

delay(8000);

colorWipe(strip.Color(50, 50, 50), 50); // White

delay(8000);

colorWipe(strip.Color(50, 0, 0), 0); // Red

delay(1000);

colorWipe(strip.Color(50, 50, 50), 0); // White

delay(1000);

colorWipe(strip.Color(50, 0, 0), 0); // Red

delay(1000);

colorWipe(strip.Color(50, 50, 50), 0); // White

delay(1000);

colorWipe(strip.Color(50, 0, 0), 0); // Red

delay(1000);

colorWipe(strip.Color(50, 50, 50), 0); // White

delay(1000);

colorWipe(strip.Color(50, 0, 0), 0); // Red

delay(1000);

colorWipe(strip.Color(50, 50, 50), 0); // White

delay(1000);

colorWipe(strip.Color(0, 0, 0), 200); // White

delay(4000);

}

// Fill the dots one after the other with a color

void colorWipe(uint32\_t c, uint8\_t wait) {

for(uint16\_t i=0; i<strip.numPixels(); i++) {

strip.setPixelColor(i, c);

strip.show();

delay(wait);

}

}

// Input a value 0 to 255 to get a color value.

// The colours are a transition r - g - b - back to r.

uint32\_t Wheel(byte WheelPos) {

WheelPos = 255 - WheelPos;

if(WheelPos < 85) {

return strip.Color(255 - WheelPos \* 3, 0, WheelPos \* 3);

}

if(WheelPos < 170) {

WheelPos -= 85;

return strip.Color(0, WheelPos \* 3, 255 - WheelPos \* 3);

}

WheelPos -= 170;

return strip.Color(WheelPos \* 3, 255 - WheelPos \* 3, 0);