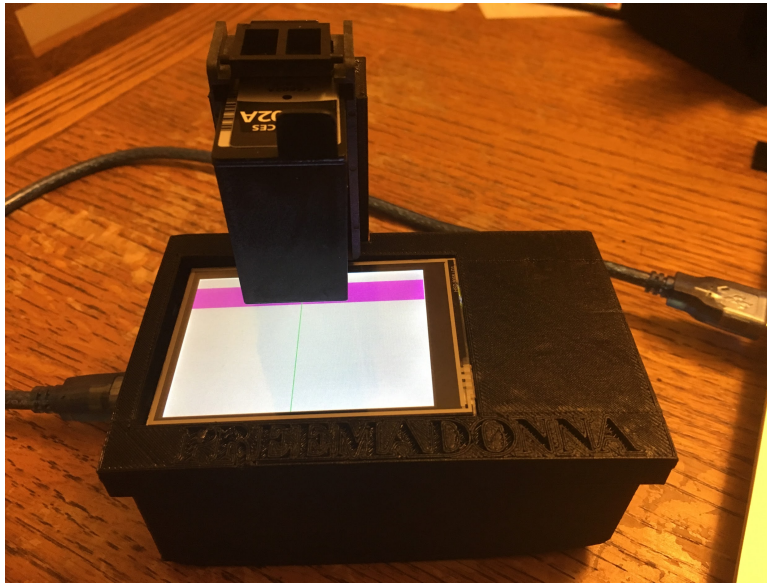




PREEMADONNA

Preemadonna Maker Kit Instructions



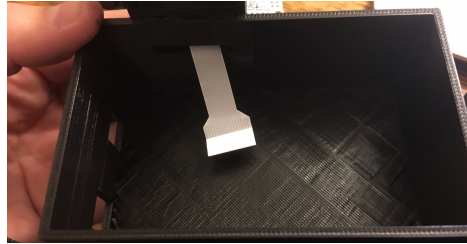
Assembly:

1. Plug the inkshield into the arduino.
2. Plug the shield stacking headers into the inkshield.
3. Plug the touchscreen into the stacking headers.

The results should look like this.



4. Load the inkjet cartridge into the carrier.
5. Feed the flat cable from the cartridge carrier into the bottom of the housing.



6. Plug the flat cable into the side of the inkshield board with the metal contacts pointed down.
7. Load the whole assembly into the bottom of the housing.
8. Put the top onto the housing.
9. Lower the cartridge assembly into place, feeding the flat cable into the housing.

Understanding the code

The code includes a number of sections. The first section loads all necessary libraries, initializes various components, and is where the design to be printed is coded. The 2nd section sets up the arduino. The 3rd section is the main loop which continuously runs. The 4th section is a subroutine that actually commands the printing.

If you are unfamiliar with arduino programming, please visit <https://www.arduino.cc/en/Tutorial/Foundations> to learn more.

This code uses a variety of libraries (provided by Preemadonna), which must be loaded onto your computer. Drag the library folders into your libraries folder. Under Windows, it will likely be called "My Documents\Arduino\libraries". For Mac users, it will likely be called "Documents/Arduino/libraries". On Linux, it will be the "libraries" folder in your sketchbook.

Preemadonna provides the initial sample code MakerKit.ino. This allows you to focus on adding your flair to the screen graphics and printed designs.

1st Section: Loading Libraries, Initializations, and Design

Loading libraries:

```
//Include all necessary libraries
#include <Adafruit_GFX.h>    // Core graphics library
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_ILI9341.h> //Graphics hardware library
#include <Adafruit_STMPE610.h> //Touchscreen hardware library
#include <InkShield.h> //inkshield library
```

Touchscreen Initializations

```
// This is calibration data for the raw touch data to the screen coordinates
#define TS_MINX 150
#define TS_MINY 130
#define TS_MAXX 3800
#define TS_MAXY 4000

// The STMPE610 uses hardware SPI on the shield, and #8
#define STMPE_CS 8
Adafruit_STMPE610 ts = Adafruit_STMPE610(STMPE_CS);

// The display also uses hardware SPI, plus #9 & #10
#define TFT_CS 10
#define TFT_DC 9
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);
```

Inkshield and Design Initializations

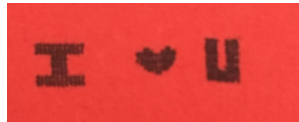
```
//initialize shield on pin 2
InkShieldAOA3 MyInkShield(2);
int enable = 1; //flag to only print when complete swipe

//initialize constants for designs
const int CharRows = 12; //how many rows each character/letter is (must be the same for all characters)
const int numOfChar = 5; //this is the number of letters/characters (including spaces) printed each time it is triggered. Change this as needed
const int fontSize = numOfChar*CharRows;
```

For this section, the CharRows and numOfChar will change depending on your design. For more details on how the design is implemented, see below.

Coding the design

The design is coded in sections that define each character (including spaces). The designs are 12 columns long (the number of nozzles in the print head) and any number of rows. The number of rows in each character has to be the same. The example types out the design below



The designs are represented as 1s and 0s. A 1 means that the nozzle is fired and thus ink is dispensed, while a 0 means it is not. An inkjet expels a drop of ink by heating a resistor which causes a small bubble to form inside the nozzle and push a drop of ink out the nozzle. For information on how the inkshield sends the necessary signals to the print head, see <http://nicholasclewis.com/projects/inkshield/theory/>

Due to the orientation of the printhead and the direction of the swipe, the design is coded sideways. So in the code the design looks more like this



```
//Letters/characters are stored in an array. Expand this array as necessary  
const word font[fontSize] = {  
    //I  
    0b111000000111,  
    0b111000000111,  
    0b111000000111,  
    0b111000000111,  
    0b111111111111,  
    0b111111111111,  
    0b111111111111,  
    0b111111111111,  
    0b111000000111,  
    0b111000000111,  
    0b111000000111,  
    0b111000000111,  
  
    //space  
    0b000000000000,  
    0b000000000000,  
    0b000000000000,  
    0b000000000000,  
    0b000000000000,  
    0b000000000000,  
    0b000000000000,  
    0b000000000000,  
    0b000000000000,  
    0b000000000000,  
    0b000000000000,  
    0b000000000000,  
    0b000000000000,  
    0b000000000000,  
    0b000000000000,  
    0b000000000000,  
  
    //Heart  
    0b000000111000,  
    0b000011111000,  
    0b000011111100,  
    0b000111111100,  
    0b0001111111000,  
    0b001111110000,  
    0b001111110000,  
    0b001111110000,  
    0b001111111000,  
    0b001111111100,  
    0b000011111100,  
    0b000001111000,  
    0b000000111000,
```

2nd Section: Setup

```
void setup(void) {

  Serial.begin(9600);
  Serial.println(F("Premadonna, Inc"));

  tft.begin();

  if (!ts.begin()) {
    Serial.println("Couldn't start touchscreen controller");
    while (1);
  }

  //This makes a white screen with a horizontal magenta band, and a vertical green line
  //Feel free to explore other graphic primitives to design what the screen looks like
  tft.fillRect(ILI9341_WHITE);
  tft.fillRect(180, 0, 40, 319, ILI9341_MAGENTA);
  tft.drawLine(0,150,250,150,ILI9341_GREEN);

}
```

The first part of this section sets up serial communication. The 2nd part draws what will appear on the touch screen. In the example provided the screen is filled with white. Then a horizontal magenta band is drawn. Finally a green vertical line is drawn. The magenta band and green line indicate where the nozzles on the inkjet cartridge are.

This graphics setup section is a great place to add your own flair. Feel free to make creative designs to display during use. For more information on the types of graphical primitives you can design, see <https://learn.adafruit.com/adafruit-gfx-graphics-library/graphics-primitives>

3rd Section: Main Loop

```
void loop()
{
  // See if there's any touch data for us
  if (ts.bufferEmpty()) {
    return;
  }

  // Retrieve a point
  TS_Point p = ts.getPoint();

  //print the raw values
  Serial.print("X = "); Serial.print(p.x);
  Serial.print("\tY = "); Serial.print(p.y);
  Serial.print("\tPressure = "); Serial.println(p.z);


  // Scale from ~0->4000 to tft.width using the calibration #'s
  p.x = map(p.x, TS_MINX, TS_MAXX, 0, tft.width());
  p.y = map(p.y, TS_MINY, TS_MAXY, 0, tft.height());

  //print the scaled values
  Serial.print("("); Serial.print(p.x);
  Serial.print(", "); Serial.print(p.y);
  Serial.println(")");

  if (p.y>100 && p.y<160) {
    if(enable==1){
      for(int letter=numOfChar-1;letter>-1;letter--) {
        spray_letter(letter);
        enable = 0; //reset enable
      }
    }
  }

  if (p.y>200){enable=1;} //change enable only if the end of the swipe
}
```

This section will continuously loop while power is supplied to the board. First, the code checks to see if there is any data from the touchscreen. Then it reads the data from the touchscreen. The data it reads is the X,Y, and pressure. If the arduino is plugged into

the computer and the serial port is being monitored  then the computer will show the data. Next the code scales the raw data to fit the size and number of pixels in the screen (which was defined in the initializations). If a computer is connected to the arduino, it will also print the scaled X,Y location.

The magic of the code is the first if loop. It checks to see if the y location is in the trigger zone. If it is, then it prints the design. If it is not it goes back to the top of the main loop and goes again. The 2nd if loop is an enable function. This makes it so that you have to do a full swipe for the code to reset itself for the next print. If the enable function is not present then anytime the y location is in the trigger zone it will continuously spew the design and result in a puddle of ink.

4th Section: Print Loop

```
void spray_letter(int letter)
{
  //loop through the rows of the letter
  for(int row=0;row<CharRows;row++){
    word strip = font[(letter*CharRows)+row];
    //print the row (change the number of times it sprays to change the darkness)
    MyInkShield.spray_ink(strip);
    MyInkShield.spray_ink(strip);
    MyInkShield.spray_ink(strip);
    delay(8);
  }
}
```

This section uses the functions in the inkshield library to loop through each row of each character of your design to print the image. If you want a darker image you can copy the spray_ink command again. For a lighter image, reduce the number of times the spray_ink command is used.

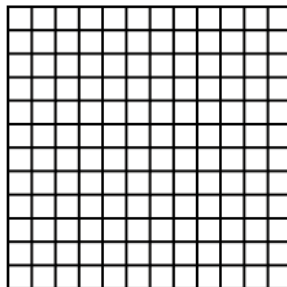
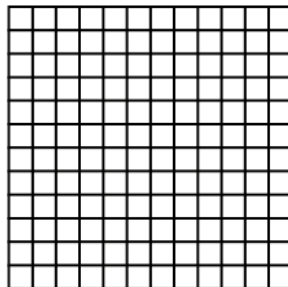
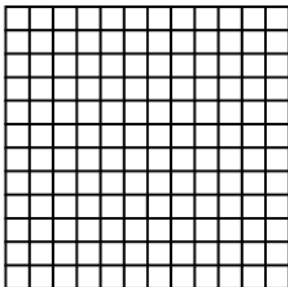
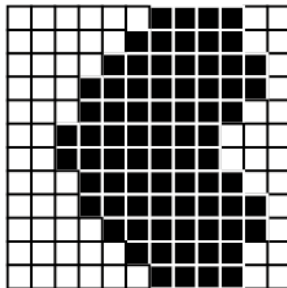
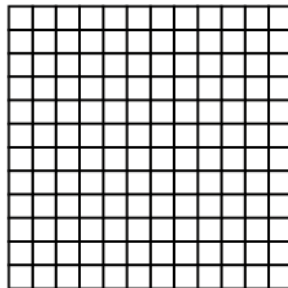
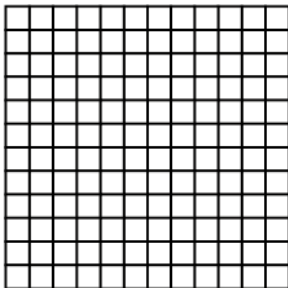
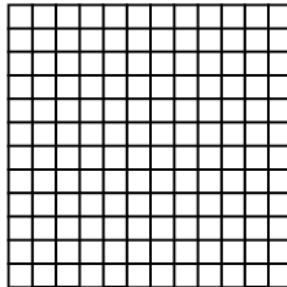
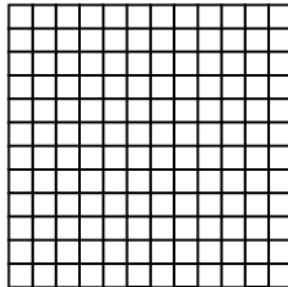
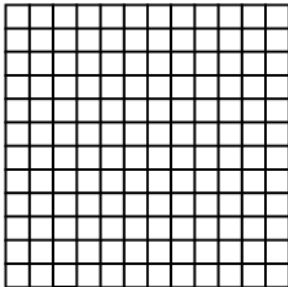
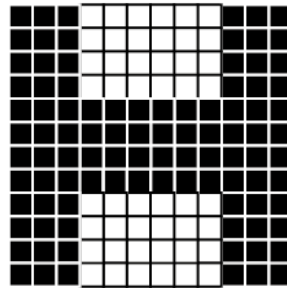
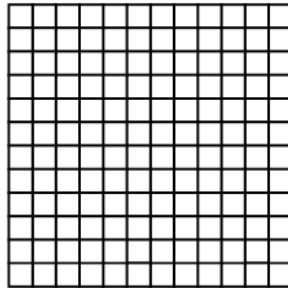
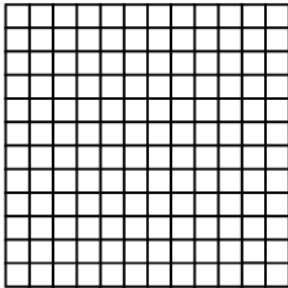
Cartridge Care

The ink in the nozzles can easy dry up. This causes the nozzle not to spray a drop of ink. If this happens, simply take a kleenex and wipe sideways across the nozzles until you get a consistent dark spot of ink when you wipe.

If the cartridge is not going to be used for a while, remove it from the printer and snap the orange cover back on. This protects the nozzles from air exposure and will ensure the cartridge lasts longer.

Design Template

Code This Side Up
(width is this orientation is ALWAYS 12--the number of nozzles)



Design This Side Up
(width at this orientation is the CharRows)