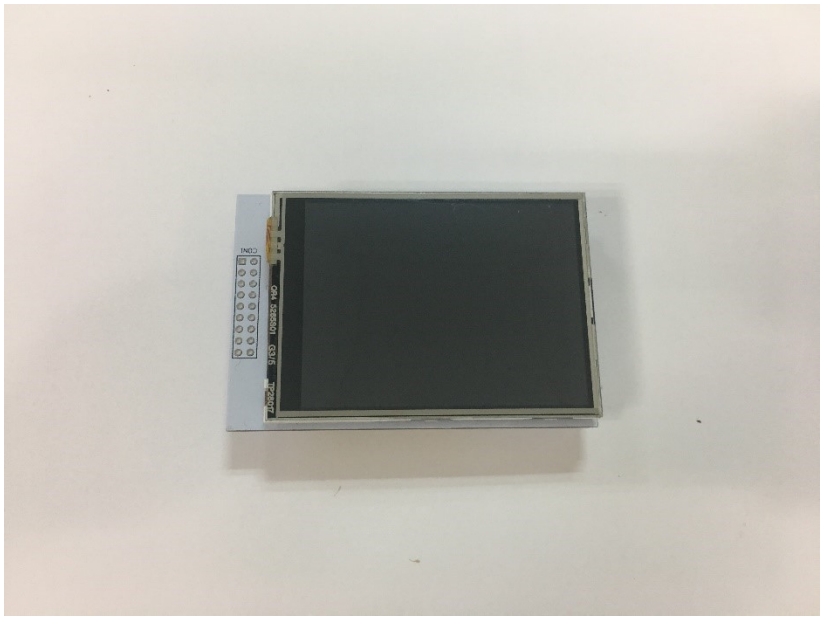
User Manual

For

2.8" TFT Touch Shield for Arduino with Resistive Touch Screen (TF028)



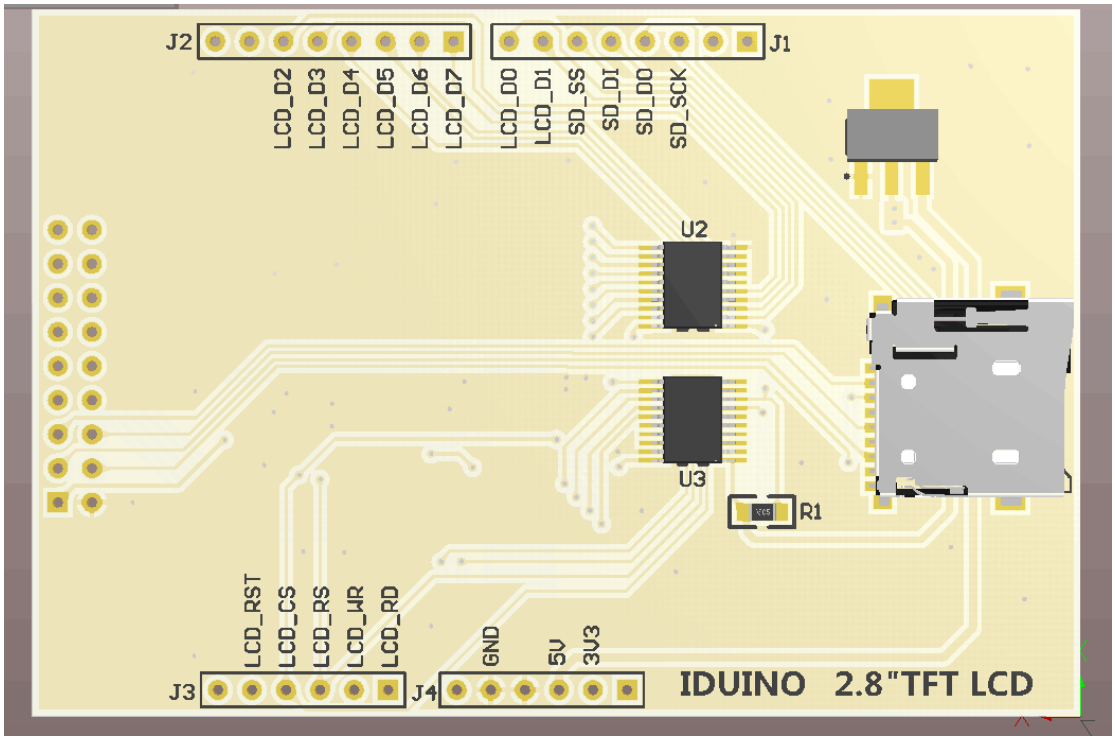
Description:

This 2.8" TFT Touchscreen is designed to suitable for Arduino UNO/Mega2560. It can directly plug inoto the UNO/Mega2560 board without any wiring and soldering. Library is compatible with Adafruit TFT touchscreen shield, which is easy to use . It has way more resolution than a black and white 128x64 display. As a bonus, this display has a resistive touchscreen attached to it already, so you can detect finger presses anywhere on the screen. 240x320 pixels with individual pixel control. Meanwhile, this module supports mini SD card to expand storage.

Specification:

- 2.8" diagonal LCD TFT display
- 240x320 resolution
- ILI9341 controller with built in video RAM buffer
- Works with any Arduino '328 or Mega
- 5V compatible! Use with 3.3V or 5V logic
- With white LED backlight.
- Suport Mini SD card extension
- Size: 78*52*16mm
- Weight: 38g

Pinout

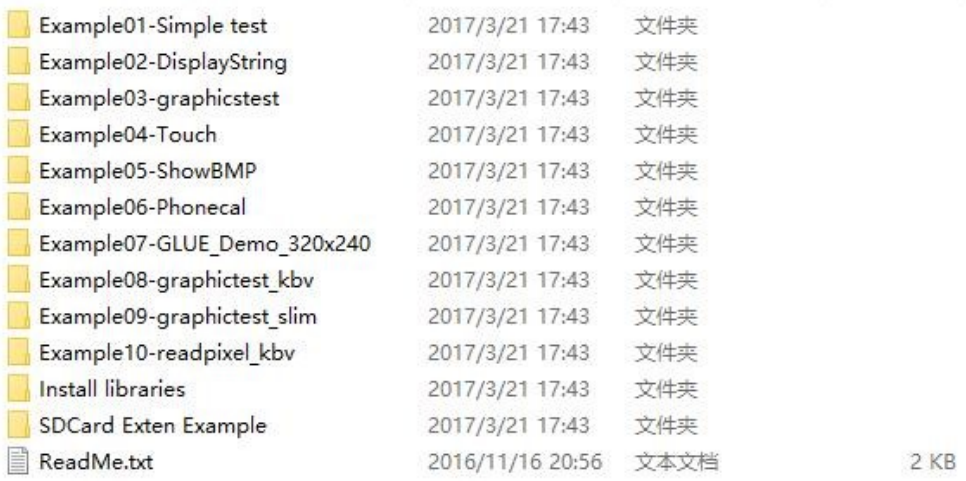


PIN	Description
LCD_RST	LCD Reset pin
LCD_CS	LCD Chip select
LCD_RS	LCD Register select
LCD_WR	LCD Write
LCD_RD	LCD Read
GND	Ground
5V	5V
3V3	3.3V
LCD_D0	LCD data bit 0
LCD_D1	LCD data bit 1
LCD_D2	LCD data bit 2
LCD_D3	LCD data bit 3
LCD_D4	LCD data bit 4
LCD_D5	LCD data bit 5
LCD_D6	LCD data bit 6
LCD_D7	LCD data bit 7

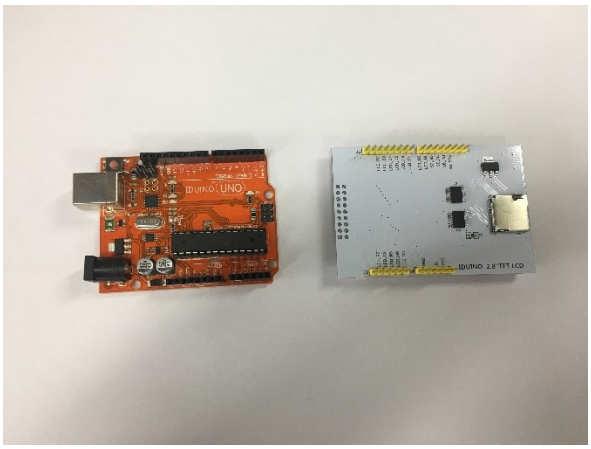
SD_SS	SD card	Slave select
SD_DI	SD card	Serial data In
SD_DO	SD card	Serial data Out
SD_SCK	SD card	Serial clock

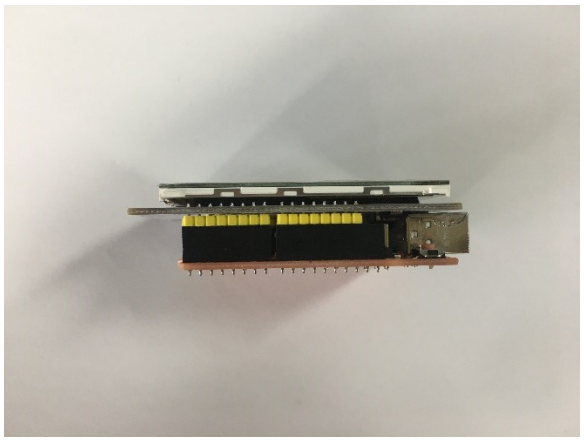
Example:

We have several cool projects as the example, such as text display, phone call and image display. Here we use the phone call to show.



Before uploading the code, you need install the library first. And no wire connection is needed, just plug the Screen module on the UNO/Mega board.





*******code begin*******

```
#include <Adafruit_GFX.h>    // Core graphics library
#include <Adafruit_TFTLCD.h> // Hardware-specific library
#include <TouchScreen.h>

// The control pins for the LCD can be assigned to any digital or
// analog pins...but we'll use the analog pins as this allows us to
// double up the pins with the touch screen (see the TFT paint example).
#define LCD_CS A3 // Chip Select goes to Analog 3
#define LCD_CD A2 // Command/Data goes to Analog 2
#define LCD_WR A1 // LCD Write goes to Analog 1
#define LCD_RD A0 // LCD Read goes to Analog 0

#define LCD_RESET A4 // Can alternately just connect to Arduino's reset pin

// When using the BREAKOUT BOARD only, use these 8 data lines to the LCD:
// For the Arduino Uno, Duemilanove, Diecimila, etc.:
//   D0 connects to digital pin 8  (Notice these are
//   D1 connects to digital pin 9   NOT in order!)
//   D2 connects to digital pin 2
//   D3 connects to digital pin 3
//   D4 connects to digital pin 4
//   D5 connects to digital pin 5
//   D6 connects to digital pin 6
//   D7 connects to digital pin 7
// For the Arduino Mega, use digital pins 22 through 29
// (on the 2-row header at the end of the board).

// Assign human-readable names to some common 16-bit color values:
#define  BLACK   0x0000
#define  BLUE    0x001F
```

```
#define  RED     0xF800
#define  GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF

// Color definitions
#define ILI9341_BLACK       0x0000      /*   0,   0,   0 */
#define ILI9341_NAVY        0x000F      /*   0,   0, 128 */
#define ILI9341_DARKGREEN   0x03E0      /*   0, 128,   0 */
#define ILI9341_DARKCYAN    0x03EF      /*   0, 128, 128 */
#define ILI9341_MAROON      0x7800      /* 128,   0,   0 */
#define ILI9341_PURPLE      0x780F      /* 128,   0, 128 */
#define ILI9341_OLIVE       0x7BE0      /* 128, 128,   0 */
#define ILI9341_LIGHTGREY   0xC618      /* 192, 192, 192 */
#define ILI9341_DARKGREY    0x7BEF      /* 128, 128, 128 */
#define ILI9341_BLUE        0x001F      /*   0,   0, 255 */
#define ILI9341_GREEN       0x07E0      /*   0, 255,   0 */
#define ILI9341_CYAN        0x07FF      /*   0, 255, 255 */
#define ILI9341_RED         0xF800      /* 255,   0,   0 */
#define ILI9341_MAGENTA     0xF81F      /* 255,   0, 255 */
#define ILI9341_YELLOW      0xFFE0      /* 255, 255,   0 */
#define ILI9341_WHITE       0xFFFF      /* 255, 255, 255 */
#define ILI9341_ORANGE      0xFD20      /* 255, 165,   0 */
#define ILI9341_GREENYELLOW 0xAFE5      /* 173, 255,  47 */
#define ILI9341_PINK        0xF81F

/****************** UI details */
#define BUTTON_X 40
#define BUTTON_Y 100
#define BUTTON_W 60
#define BUTTON_H 30
#define BUTTON_SPACING_X 20
#define BUTTON_SPACING_Y 20
#define BUTTON_TEXTSIZE 2

// text box where numbers go
#define TEXT_X 10
#define TEXT_Y 10
#define TEXT_W 220
#define TEXT_H 50
#define TEXT_TSIZE 3
#define TEXT_TCOLOR ILI9341_MAGENTA
```

```
// the data (phone #) we store in the textfield
#define TEXT_LEN 12
char textfield[TEXT_LEN+1] = "";
uint8_t textfield_i=0;

#define YP A3  // must be an analog pin, use "An" notation!
#define XM A2  // must be an analog pin, use "An" notation!
#define YM 9   // can be a digital pin
#define XP 8   // can be a digital pin

#define TS_MINX 100
#define TS_MAXX 920

#define TS_MINY 70
#define TS_MAXY 900
// We have a status line for like, is FONA working
#define STATUS_X 10
#define STATUS_Y 65



#include <MCUFRIEND_kbv.h>
MCUFRIEND_kbv tft;
TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);
// If using the shield, all control and data lines are fixed, and
// a simpler declaration can optionally be used:
// Adafruit_TFTLCD tft;

Adafruit_GFX_Button buttons[15];
/* create 15 buttons, in classic candybar phone style */
char buttonlabels[15][5] = {"Send", "Clr", "End", "1", "2", "3", "4", "5", "6", "7",
"8", "9", "*", "0", "#" };
uint16_t buttoncolors[15] = {ILI9341_DARKGREEN, ILI9341_DARKGREY, ILI9341_RED,
                             ILI9341_BLUE, ILI9341_BLUE, ILI9341_BLUE,
                             ILI9341_BLUE, ILI9341_BLUE, ILI9341_BLUE,
                             ILI9341_BLUE, ILI9341_BLUE, ILI9341_BLUE,
                             ILI9341_ORANGE, ILI9341_BLUE, ILI9341_ORANGE};

void setup(void) {
  Serial.begin(9600);
  Serial.println(F("TFT LCD test"));

#ifdef USE_ADAFRUIT_SHIELD_PINOUT
  Serial.println(F("Using Adafruit 2.4\" TFT Arduino Shield Pinout"));
```

```
#else
  Serial.println(F("Using Adafruit 2.4\" TFT Breakout Board Pinout"));
#endif

  Serial.print("TFT size is "); Serial.print(tft.width()); Serial.print("x");
Serial.println(tft.height());

  tft.reset();

  uint16_t identifier = tft.readID();
  if(identifier == 0x9325) {
    Serial.println(F("Found ILI9325 LCD driver"));
  } else if(identifier == 0x9328) {
    Serial.println(F("Found ILI9328 LCD driver"));
  } else if(identifier == 0x4535) {
    Serial.println(F("Found LGDP4535 LCD driver"));
  }else if(identifier == 0x7575) {
    Serial.println(F("Found HX8347G LCD driver"));
  } else if(identifier == 0x9341) {
    Serial.println(F("Found ILI9341 LCD driver"));
  }else if(identifier == 0x7783) {
    Serial.println(F("Found ST7781 LCD driver"));
  }else if(identifier == 0x8230) {
    Serial.println(F("Found UC8230 LCD driver"));
  }
  else if(identifier == 0x8357) {
    Serial.println(F("Found HX8357D LCD driver"));
  } else if(identifier==0x0101)
  {
      identifier=0x9341;
       Serial.println(F("Found 0x9341 LCD driver"));
  }else {
    Serial.print(F("Unknown LCD driver chip: "));
    Serial.println(identifier, HEX);
    Serial.println(F("If using the Adafruit 2.8\" TFT Arduino shield, the line:"));
    Serial.println(F("  #define USE_ADAFRUIT_SHIELD_PINOUT"));
    Serial.println(F("should appear in the library header (Adafruit_TFT.h)."));
    Serial.println(F("If using the breakout board, it should NOT be #defined!"));
    Serial.println(F("Also if using the breakout, double-check that all wiring"));
    Serial.println(F("matches the tutorial."));
    identifier=0x9341;

  }
```

```
  tft.begin(identifier);
  tft.setRotation(0);
  tft.fillScreen(BLACK);

  // create buttons
  for (uint8_t row=0; row<5; row++) {
    for (uint8_t col=0; col<3; col++) {
      buttons[col + row*3].initButton(&tft, BUTTON_X+col*(BUTTON_W+BUTTON_SPACING_X),
                 BUTTON_Y+row*(BUTTON_H+BUTTON_SPACING_Y),    // x, y, w, h, outline,
fill, text
                  BUTTON_W,    BUTTON_H,    ILI9341_WHITE,    buttoncolors[col+row*3],
ILI9341_WHITE,
                  buttonlabels[col + row*3], BUTTON_TEXTSIZE);
      buttons[col + row*3].drawButton();
    }
  }

  // create 'text field'
  tft.drawRect(TEXT_X, TEXT_Y, TEXT_W, TEXT_H, ILI9341_WHITE);



}
// Print something in the mini status bar with either flashstring
void status(const __FlashStringHelper *msg) {
  tft.fillRect(STATUS_X, STATUS_Y, 240, 8, ILI9341_BLACK);
  tft.setCursor(STATUS_X, STATUS_Y);
  tft.setTextColor(ILI9341_WHITE);
  tft.setTextSize(1);
  tft.print(msg);
}
// or charstring
void status(char *msg) {
  tft.fillRect(STATUS_X, STATUS_Y, 240, 8, ILI9341_BLACK);
  tft.setCursor(STATUS_X, STATUS_Y);
  tft.setTextColor(ILI9341_WHITE);
  tft.setTextSize(1);
  tft.print(msg);
}
#define MINPRESSURE 10
#define MAXPRESSURE 1000
void loop(void) {
  /*TSPoint p;
  p = ts.getPoint();
  */
```

```
 digitalWrite(13, HIGH);
 TSPoint p = ts.getPoint();
 digitalWrite(13, LOW);

 // if sharing pins, you'll need to fix the directions of the touchscreen pins
 //pinMode(XP, OUTPUT);
 pinMode(XM, OUTPUT);
 pinMode(YP, OUTPUT);
 //pinMode(YM, OUTPUT);

 // we have some minimum pressure we consider 'valid'
 // pressure of 0 means no pressing!

// p = ts.getPoint();
 /*
 if (ts.bufferSize()) {

 } else {
   // this is our way of tracking touch 'release'!
   p.x = p.y = p.z = -1;
 }*/

 // Scale from ~0->4000 to tft.width using the calibration #'s
 /*
 if (p.z != -1) {
   p.x = map(p.x, TS_MINX, TS_MAXX, 0, tft.width());
   p.y = map(p.y, TS_MINY, TS_MAXY, 0, tft.height());
   Serial.print("("); Serial.print(p.x); Serial.print(", ");
   Serial.print(p.y); Serial.print(", ");
   Serial.print(p.z); Serial.println(") ");
 }*/
  if (p.z > MINPRESSURE && p.z < MAXPRESSURE) {
   // scale from 0->1023 to tft.width
   p.x = map(p.x, TS_MINX, TS_MAXX, tft.width(), 0);
   p.y = (tft.height()-map(p.y, TS_MINY, TS_MAXY, tft.height(), 0));
  }

 // go thru all the buttons, checking if they were pressed
 for (uint8_t b=0; b<15; b++) {
   if (buttons[b].contains(p.x, p.y)) {
     //Serial.print("Pressing: "); Serial.println(b);
     buttons[b].press(true);  // tell the button it is pressed
   } else {
     buttons[b].press(false);  // tell the button it is NOT pressed
```

```
  }
}

// now we can ask the buttons if their state has changed
for (uint8_t b=0; b<15; b++) {
  if (buttons[b].justReleased()) {
    // Serial.print("Released: "); Serial.println(b);
    buttons[b].drawButton();  // draw normal
  }

  if (buttons[b].justPressed()) {
      buttons[b].drawButton(true);  // draw invert!

      // if a numberpad button, append the relevant # to the textfield
      if (b >= 3) {
        if (textfield_i < TEXT_LEN) {
          textfield[textfield_i] = buttonlabels[b][0];
          textfield_i++;
       textfield[textfield_i] = 0; // zero terminate

         // fona.playDTMF(buttonlabels[b][0]);
        }
      }

      // clr button! delete char
      if (b == 1) {

        textfield[textfield_i] = 0;
        if (textfield > 0) {
          textfield_i--;
          textfield[textfield_i] = ' ';
        }
      }

      // update the current text field
      Serial.println(textfield);
      tft.setCursor(TEXT_X + 2, TEXT_Y+10);
      tft.setTextColor(TEXT_TCOLOR, ILI9341_BLACK);
      tft.setTextSize(TEXT_TSIZE);
      tft.print(textfield);

      // its always OK to just hang up
      if (b == 2) {
        status(F("Hanging up"));
```

```
          //fona.hangUp();
        }
        // we dont really check that the text field makes sense
        // just try to call
        if (b == 0) {
          status(F("Calling"));
          Serial.print("Calling "); Serial.print(textfield);

          //fona.callPhone(textfield);
        }

      delay(100); // UI debouncing
    }
  }

}
```

*******code End*******

Upload the code, you'll see the display as follow:

Reference

https://learn.adafruit.com/arduin-o-phone-arduino-powered-diy-cellphone