CEC Continuing Education Center

DesignNews

Arduino Pro Primer

Day 5:

The Multi-Talented Arduino Giga Display Shield

Sponsored by

DigiKey

informa markets

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.

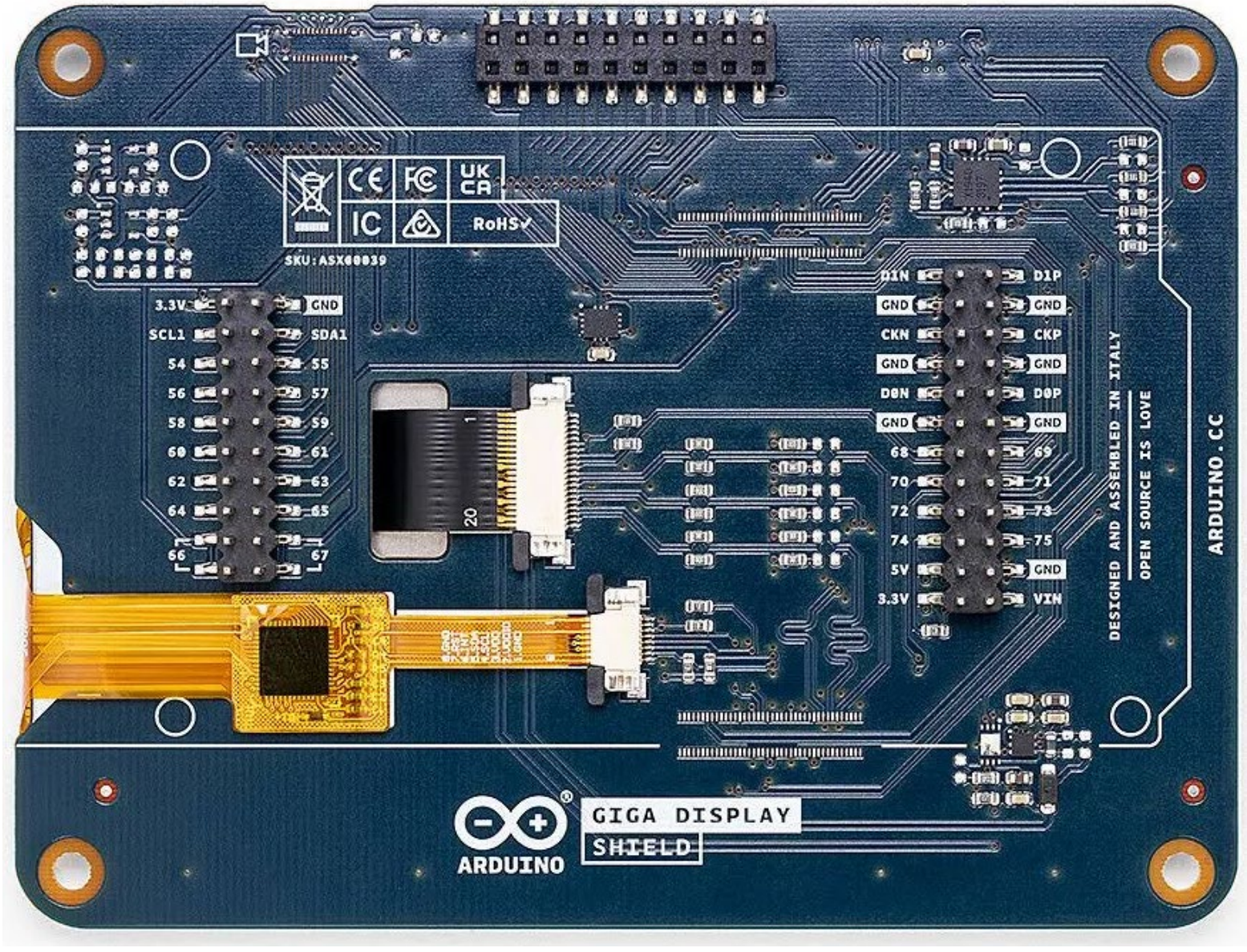
Fred Eady

Visit ‘Lecturer Profile’ in your console for more details.

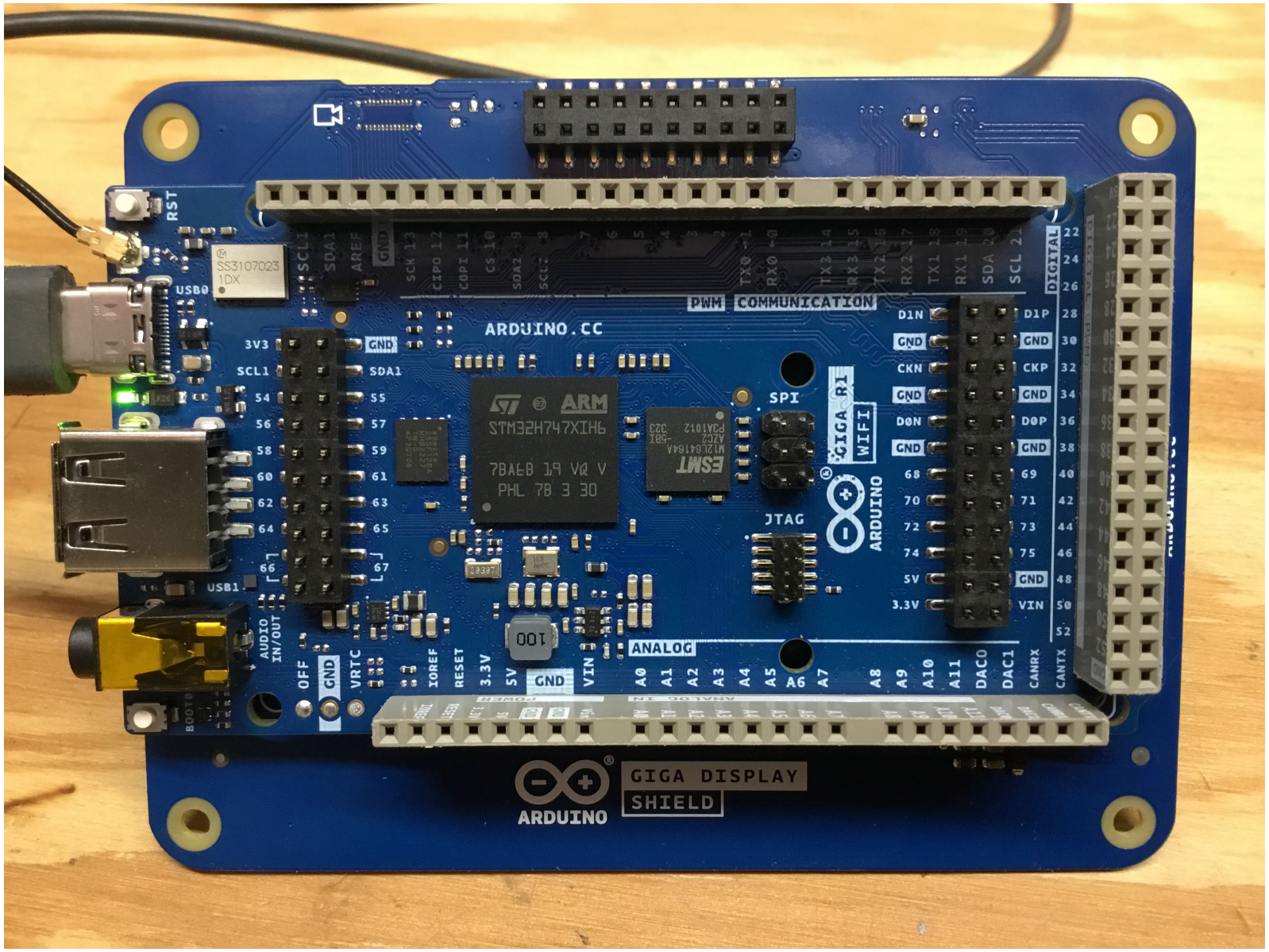
AGENDA

- **Hardware Meld**
- **GFX**
- **LVGL (Light and Versatile Graphic Library)**

Mount the Arduino Giga Display Shield

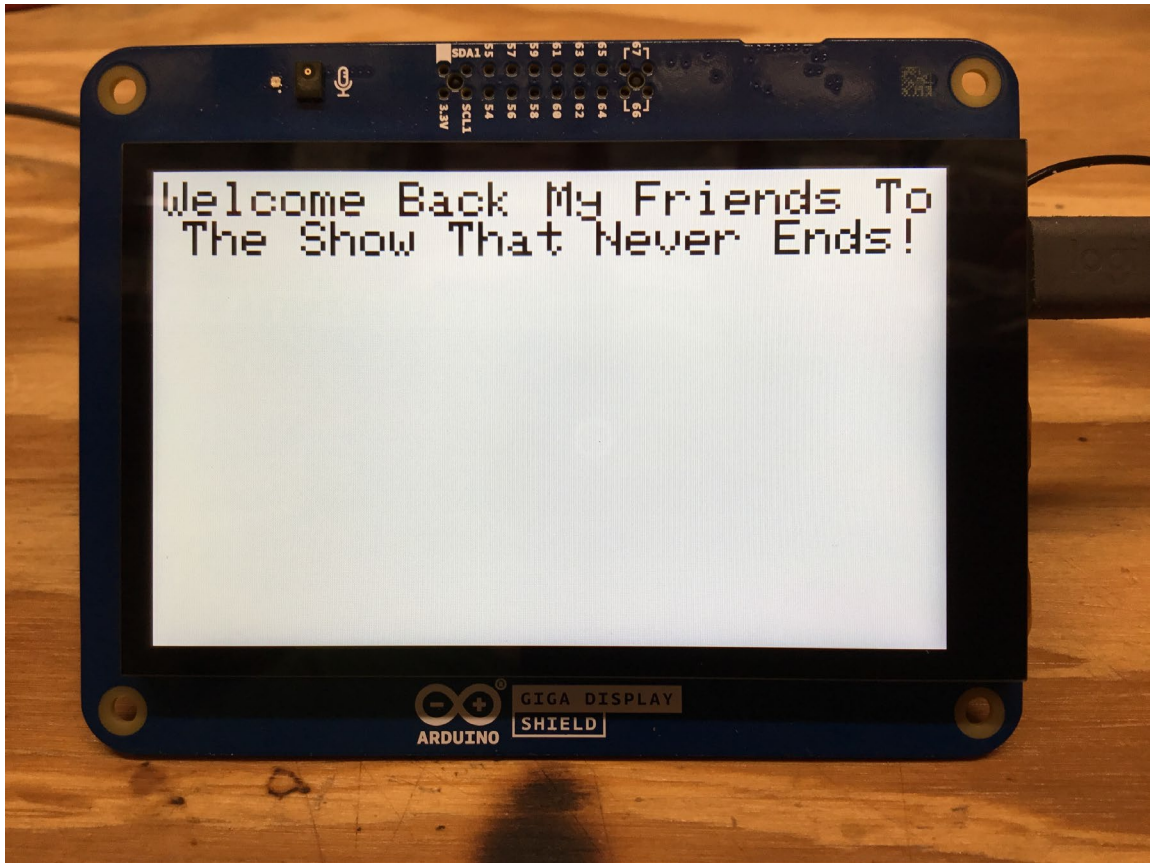


Mount the Arduino Giga Display Shield



Opening Shot (Courtesy of Greg Lake and ELP)

```
#include "Arduino_GigaDisplay_GFX.h"

GigaDisplay_GFX gfxDisplay; // Spawn the Display Object

// Common Color Definitions
#define RED      0xF800
#define GREEN    0x07E0
#define BLUE     0x001F
#define CYAN     0x07FF
#define MAGENTA  0xF81F
#define YELLOW   0xFFE0
#define BLACK    0x0000
#define WHITE    0xFFFF

void setup() {
  gfxDisplay.begin();
  gfxDisplay.setRotation(1);      // Landscape
  gfxDisplay.fillScreen(WHITE);   // Background Color
  gfxDisplay.setTextColor(BLACK); // Text Color
  gfxDisplay.setCursor(10,10);    // Initial x,y Position
  gfxDisplay.setTextSize(5);      // Text Size
  gfxDisplay.print("Welcome Back My Friends To The Show That Never Ends!");
}
void loop(){}
```

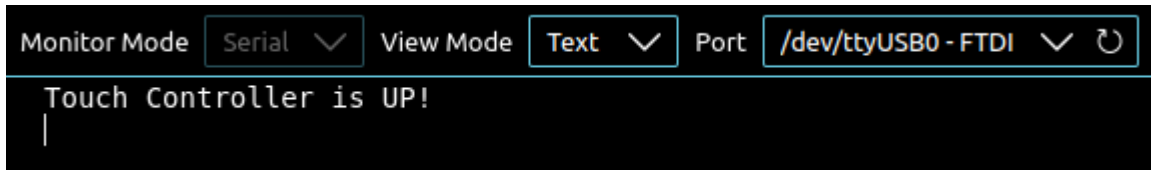


U Can Touch This!


```
6   #include "Arduino_GigaDisplay_GFX.h"
7   #include "Arduino_GigaDisplayTouch.h"
8
9   // Spawn the touch and display instances
10  Arduino_GigaDisplayTouch touchSensor;
11  GigaDisplay_GFX gfxDisplay;
12
13  // Common Color Definitions
14  #define RED      0xF800
15  #define GREEN    0x07E0
16  #define BLUE     0x001F
17  #define CYAN     0x07FF
18  #define MAGENTA  0xF81F
19  #define YELLOW   0xFFE0
20  #define BLACK    0x0000
21  #define WHITE    0xFFFF
22
23  uint8_t touch_contacts;     // number of touch points (fingers) detected
24  GDTpoint_t touch_points[5]; // max number of possible touch points (fingers)
25
26  int touch_x;  // x touch coordinate value
27  int touch_y;  // y touch coordinate value
28
29  int lastTouch;
30  int touch_threshhold_ms;  // touch debounce/delay time
```

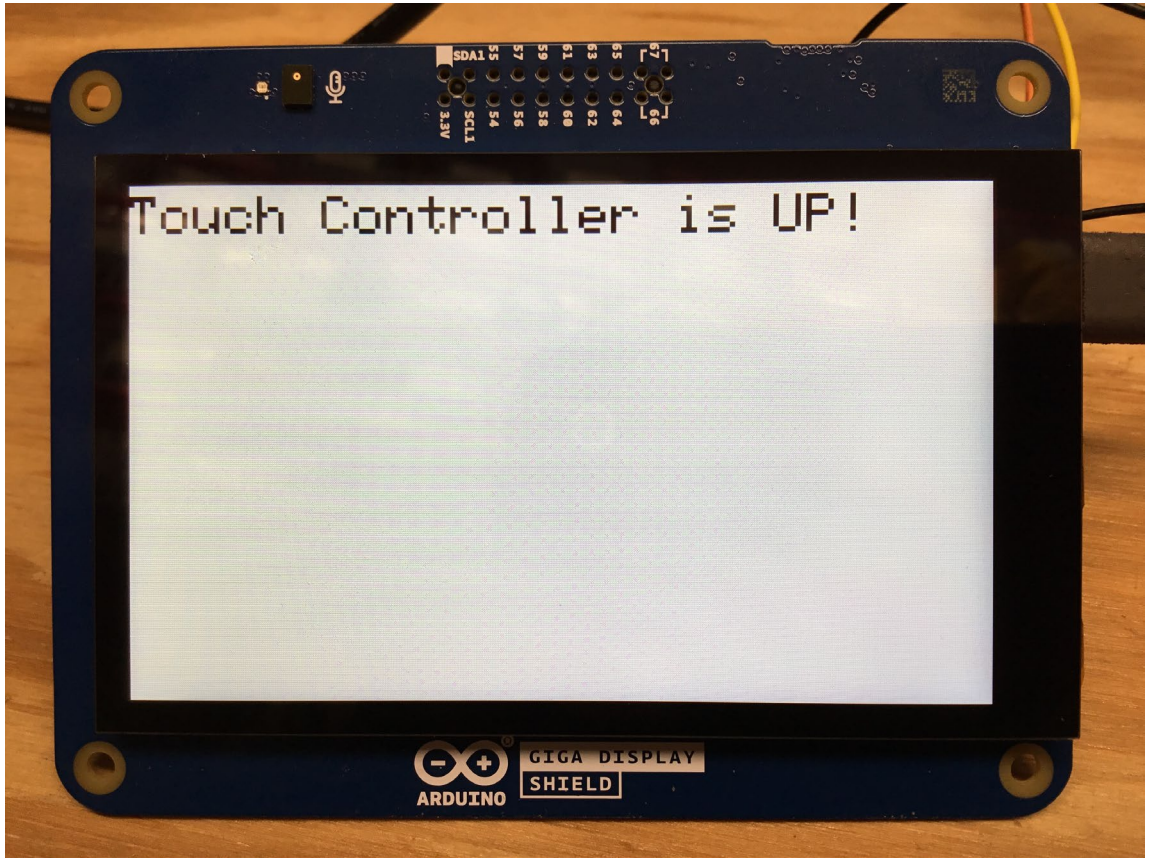
U Can Touch This!

```
32  void setup() {
33    Serial1.begin(115200);
34    gfxDisplay.begin();
35    gfxDisplay.setRotation(1);       // landscape
36    gfxDisplay.fillScreen(WHITE);    // white background
37    gfxDisplay.setTextSize(5);       // text size
38    gfxDisplay.setTextColor(BLACK); // text color
39    lastTouch = 0;                   // Initialize millis variable
40    touch_threshhold_ms = 250;       // Initialize the touch delay threshhold
41    if (touchSensor.begin()) {
42      Serial1.println("Touch Controller is UP!");
43      gfxDisplay.setCursor(0,10);
44      gfxDisplay.println("Touch Controller is UP!");
45    } else {
46      Serial1.println("FATAL ERROR: Touch Controller Failed to Initialize.");
47      gfxDisplay.println("FATAL ERROR: Touch Controller Failed to Initialize.");
48      while (1);
49    }
50  }
```



U Can Touch This!

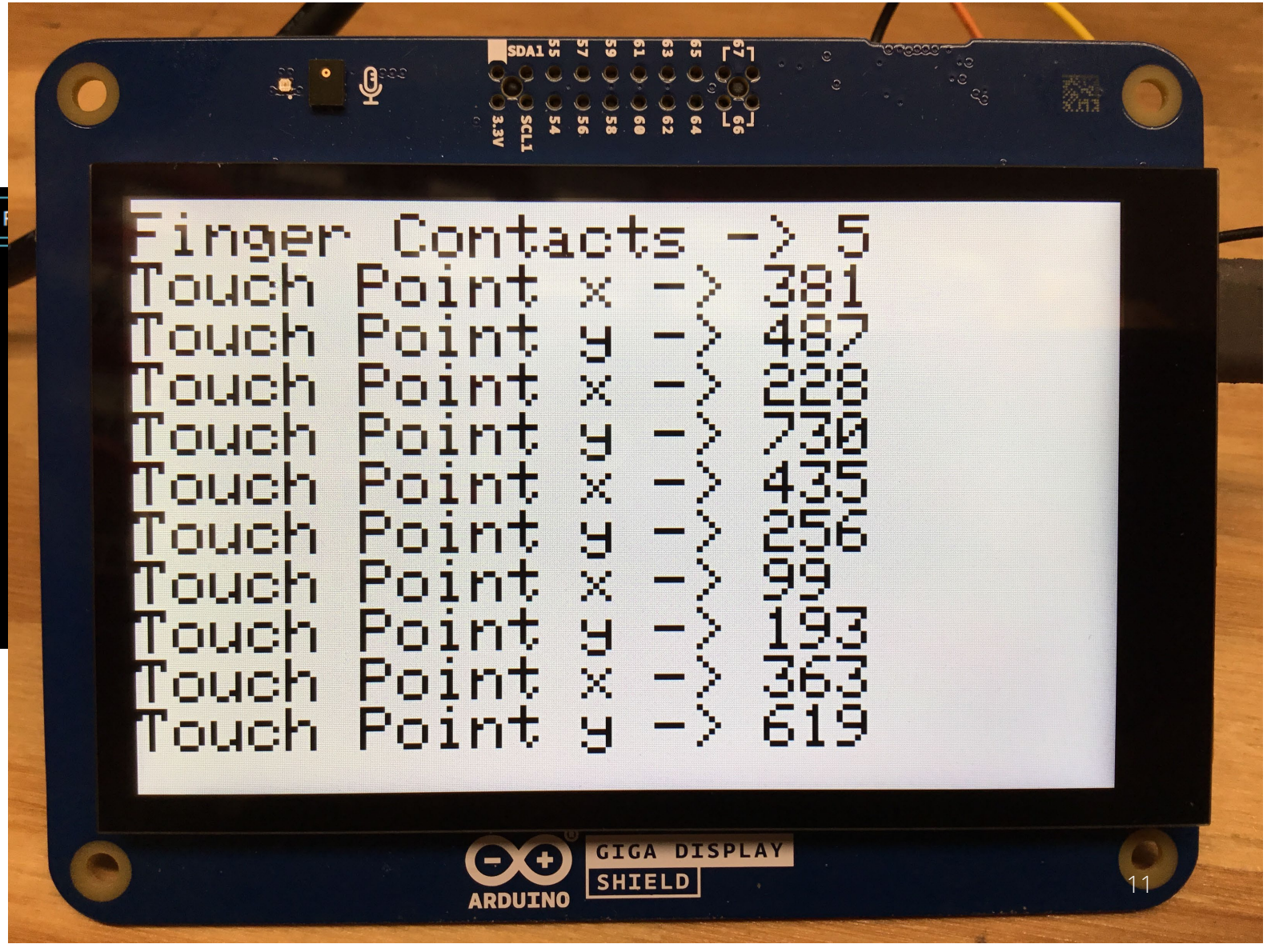
```
52  void loop() {
53    touch_contacts = touchSensor.getTouchPoints(touch_points);
54
55    if (touch_contacts > 0 && (millis() - lastTouch > touch_threshhold_ms)) {
56      // Display and Print the Number of Contacts (Fingers)
57      gfxDisplay.fillScreen(WHITE);
58      gfxDisplay.setCursor(0,10);
59      Serial1.print("Number of Touch (Finger) Contacts -> ");
60      Serial1.println(touch_contacts);
61      gfxDisplay.print("Finger Contacts -> ");
62      gfxDisplay.println(touch_contacts);
63      // Display and Print the Captured x,y Coordinates
64      for (uint8_t i = 0; i < touch_contacts; i++) {
65        touch_x = touch_points[i].x;
66        touch_y = touch_points[i].y;
67        Serial1.print("Touch Point x -> ");
68        Serial1.println(touch_points[i].x);
69        Serial1.print("Touch Point y -> ");
70        Serial1.println(touch_points[i].y);
71        gfxDisplay.print("Touch Point x -> ");
72        gfxDisplay.println(touch_points[i].x);
73        gfxDisplay.print("Touch Point y -> ");
74        gfxDisplay.println(touch_points[i].y);
75      }
76      lastTouch = millis();
77    }
78  }
```



Give the Arduino Giga Display Shield a Hand

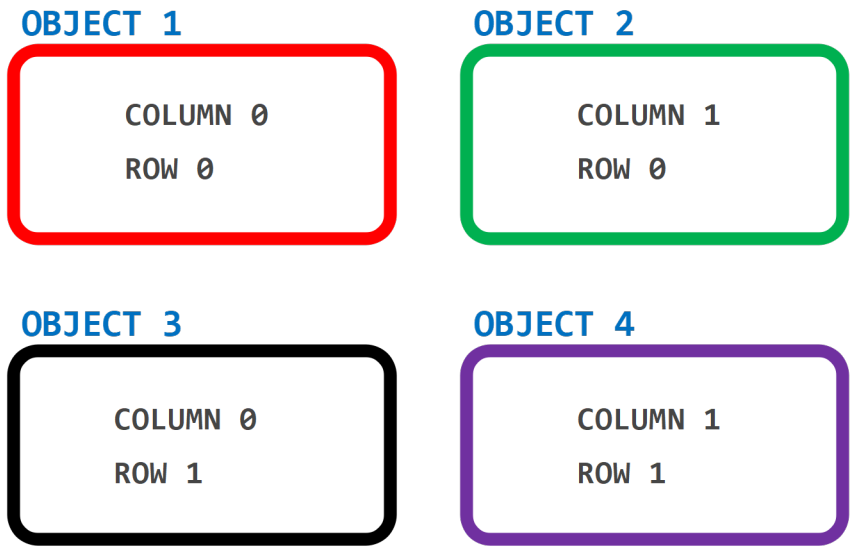
Monitor Mode: Serial | View Mode: Text | Port: /dev/ttyUSB0 -

```
Number of Touch (Finger) Contacts -> 1
Touch Point x -> 342
Touch Point y -> 640
Number of Touch (Finger) Contacts -> 5
Touch Point x -> 381
Touch Point y -> 487
Touch Point x -> 228
Touch Point y -> 730
Touch Point x -> 435
Touch Point y -> 256
Touch Point x -> 99
Touch Point y -> 193
Touch Point x -> 363
Touch Point y -> 619
```



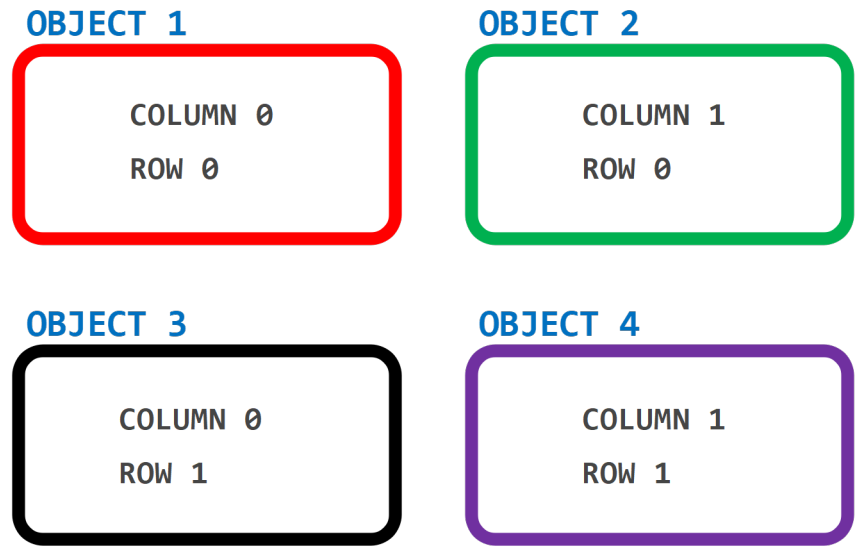
Create LVGL Objects and Callback Functions

```
#include "Arduino_H7_Video.h"
#include "Arduino_GigaDisplayTouch.h"
#include "lvgl.h"

// Create display and touch objects
Arduino_H7_Video          lvglDisplay(800, 480, GigaDisplayShield);
Arduino_GigaDisplayTouch  touchSensor;

// Button Callback Functions
static void btn00_event_cb(lv_event_t *e) {
  lv_obj_t *btn00 = (lv_obj_t *)lv_event_get_target(e);
  lv_obj_t *label00 = lv_obj_get_child(btn00, 0);
  lv_label_set_text_fmt(label00, "Clicked Btn 00");
}
static void btn10_event_cb(lv_event_t *e) {
  lv_obj_t *btn10 = (lv_obj_t *)lv_event_get_target(e);
  lv_obj_t *label10 = lv_obj_get_child(btn10, 0);
  lv_label_set_text_fmt(label10, "Clicked Btn 10");
}
static void btn01_event_cb(lv_event_t *e) {
  lv_obj_t *btn01 = (lv_obj_t *)lv_event_get_target(e);
  lv_obj_t *label01 = lv_obj_get_child(btn01, 0);
  lv_label_set_text_fmt(label01, "Clicked Btn 01");
}
static void btn11_event_cb(lv_event_t *e) {
  lv_obj_t *btn11 = (lv_obj_t *)lv_event_get_target(e);
  lv_obj_t *label11 = lv_obj_get_child(btn11, 0);
  lv_label_set_text_fmt(label11, "Clicked Btn 11");
}
```



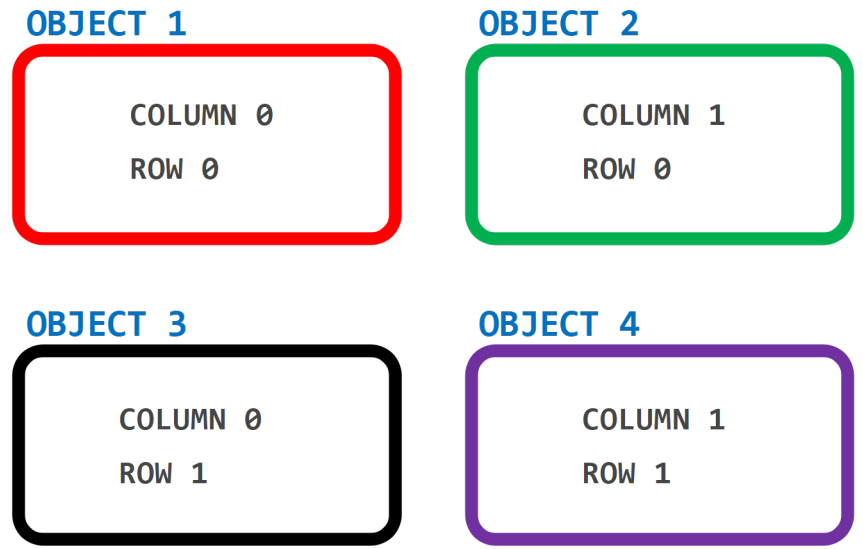
Configure the Screen and Define the Grid

```
31  void setup() {
32    lvglDisplay.begin();
33    touchSensor.begin();
34
35    // LVGL Screen Configuration
36    // Create a pointer to the active screen space and set the screen space size
37    lv_obj_t *lvglScreen = lv_obj_create(lv_scr_act());
38    lv_obj_set_size(lvglScreen, lvglDisplay.width(), lvglDisplay.height());
39
40    // Create Grid Layout ->  2 Columns - 2 Rows
41    static lv_coord_t col_dsc[] = {300, 300, LV_GRID_TEMPLATE_LAST};
42    static lv_coord_t row_dsc[] = {200, 200, LV_GRID_TEMPLATE_LAST};
43
44    lv_obj_t *lvglGrid = lv_obj_create(lv_scr_act());
45
46    lv_obj_set_grid_dsc_array(lvglGrid, col_dsc, row_dsc);
47
48    lv_obj_set_size(lvglGrid, lvglDisplay.width(), lvglDisplay.height());
49
50    lv_obj_center(lvglGrid);
```



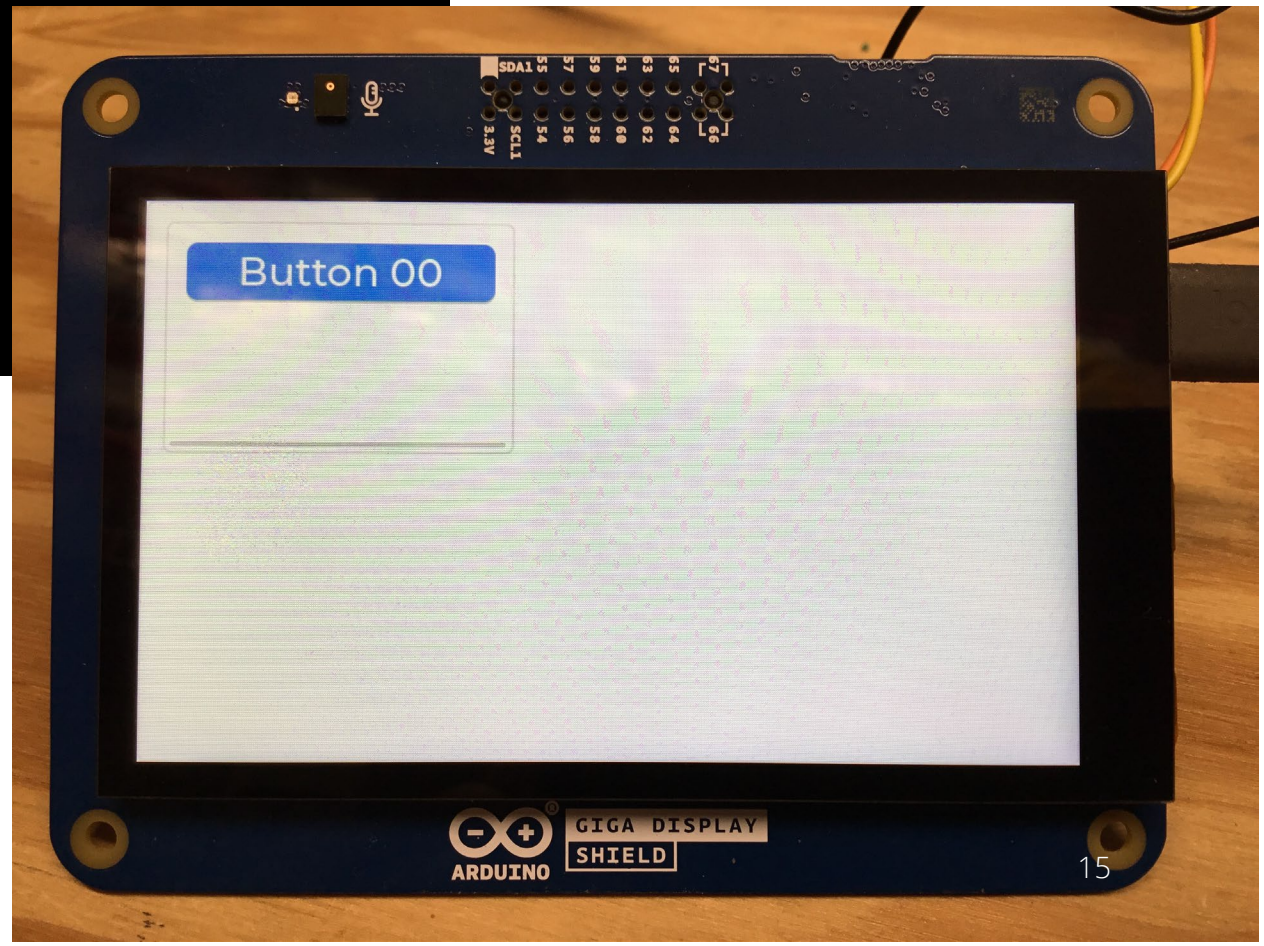
Configure the Screen and Define the Grid

```
31  void setup() {
32    lvglDisplay.begin();
33    touchSensor.begin();
34
35    // LVGL Screen Configuration
36    // Create a pointer to the active screen space and set the screen space size
37    lv_obj_t *lvglScreen = lv_obj_create(lv_scr_act());
38    lv_obj_set_size(lvglScreen, lvglDisplay.width(), lvglDisplay.height());
39
40    // Create Grid Layout ->  2 Columns - 2 Rows
41    static lv_coord_t col_dsc[] = {300, 300, LV_GRID_TEMPLATE_LAST};
42    static lv_coord_t row_dsc[] = {200, 200, LV_GRID_TEMPLATE_LAST};
43
44    lv_obj_t *lvglGrid = lv_obj_create(lv_scr_act());
45
46    lv_obj_set_grid_dsc_array(lvglGrid, col_dsc, row_dsc);
47
48    lv_obj_set_size(lvglGrid, lvglDisplay.width(), lvglDisplay.height());
49
50    lv_obj_center(lvglGrid);
```



Populate Grid Location [0;0]

```
52  // Create and Place Elements at Grid Location [0;0]
53  lv_obj_t *label00;
54  lv_obj_t *obj;
55  obj = lv_obj_create(lvglGrid);
56  lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, 0, 1,   // 0 = column align - 1 = column span
57  | | | | | | | | | | | | LV_GRID_ALIGN_STRETCH, 0, 1);  // 0 = row align - 1 = row span
58  lv_obj_set_flex_flow(obj, LV_FLEX_FLOW_COLUMN);
59  // Create Button and Link to Event Callback
60  lv_obj_t *btn00 = lv_btn_create(obj);
61  lv_obj_set_size(btn00, 265, 50);
62  lv_obj_center(btn00);
63  lv_obj_add_event_cb(btn00, btn00_event_cb, LV_EVENT_CLICKED, NULL);
64  // Create the Button Label
65  label00 = lv_label_create(btn00);
66  lv_label_set_text(label00, "Button 00");
67  lv_obj_center(label00);
```

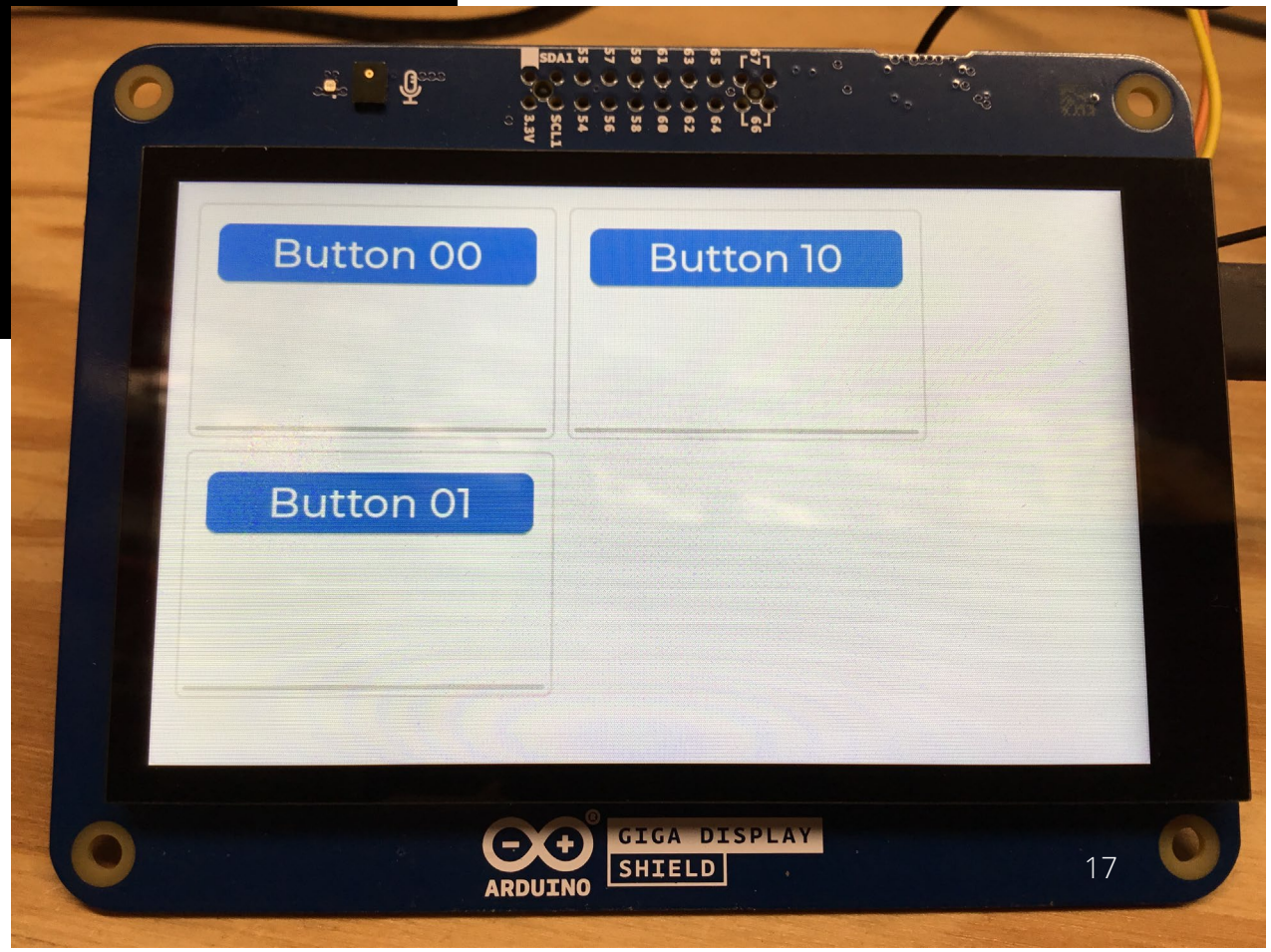


Populate Grid Location [1;0]

```
70  // Create and Place Elements at Grid Location [1;0]
71  lv_obj_t *label10;
72  obj = lv_obj_create(lvglGrid);
73  lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, 1, 1,  // 0 = column align - 1 = column span
74  | | | | | | | | | | | | LV_GRID_ALIGN_STRETCH, 0, 1); // 0 = row align - 1 = row span
75  lv_obj_set_flex_flow(obj, LV_FLEX_FLOW_COLUMN);
76  // Create Button and Link to Event Callback
77  lv_obj_t *btn10 = lv_btn_create(obj);
78  lv_obj_set_size(btn10, 265, 50);
79  lv_obj_center(btn10);
80  lv_obj_add_event_cb(btn10, btn10_event_cb, LV_EVENT_CLICKED, NULL);
81  // Create the Button Label
82  label10 = lv_label_create(btn10);
83  lv_label_set_text(label10, "Button 10");
84  lv_obj_center(label10);
```

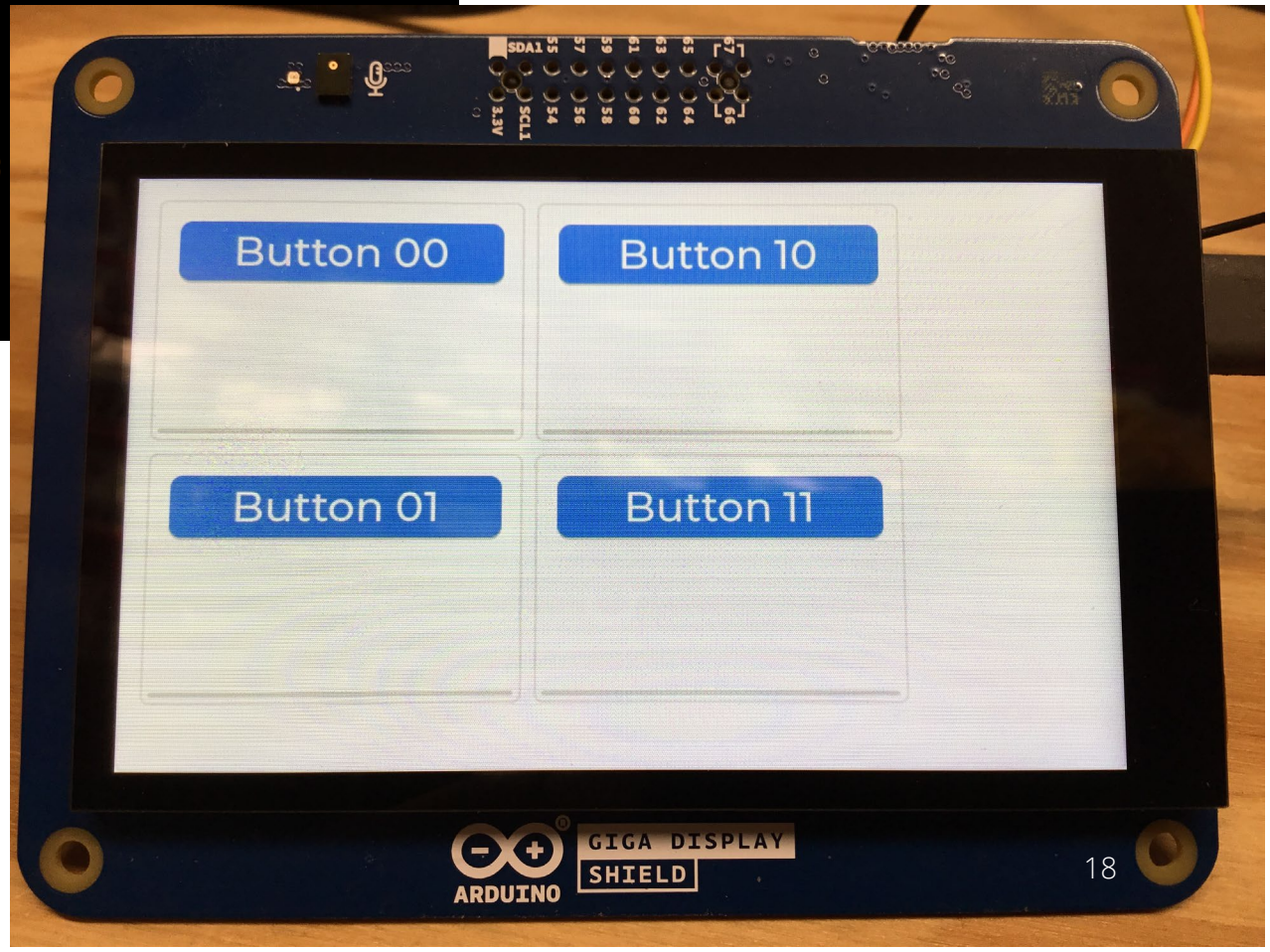
Populate Grid Location [0;1]

```
86   // Create and Place Elements at Grid Location [0;1]
87   lv_obj_t *label01;
88   obj = lv_obj_create(lvglGrid);
89   lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, 0, 1,  // 0 = column align - 1 = column span
90   | | | | | | | | | | | | | LV_GRID_ALIGN_STRETCH, 1, 1); // 0 = row align - 1 = row span
91   lv_obj_set_flex_flow(obj, LV_FLEX_FLOW_COLUMN);
92   // Create Button and Link to Event Callback
93   lv_obj_t *btn01 = lv_btn_create(obj);
94   lv_obj_set_size(btn01, 265, 50);
95   lv_obj_center(btn01);
96   lv_obj_add_event_cb(btn01, btn01_event_cb, LV_EVENT_CLICKED, NULL);
97   // Create the Button Label
98   label01 = lv_label_create(btn01);
99   lv_label_set_text(label01, "Button 01");
100  lv_obj_center(label01);
```



Populate Grid Location [1;1]

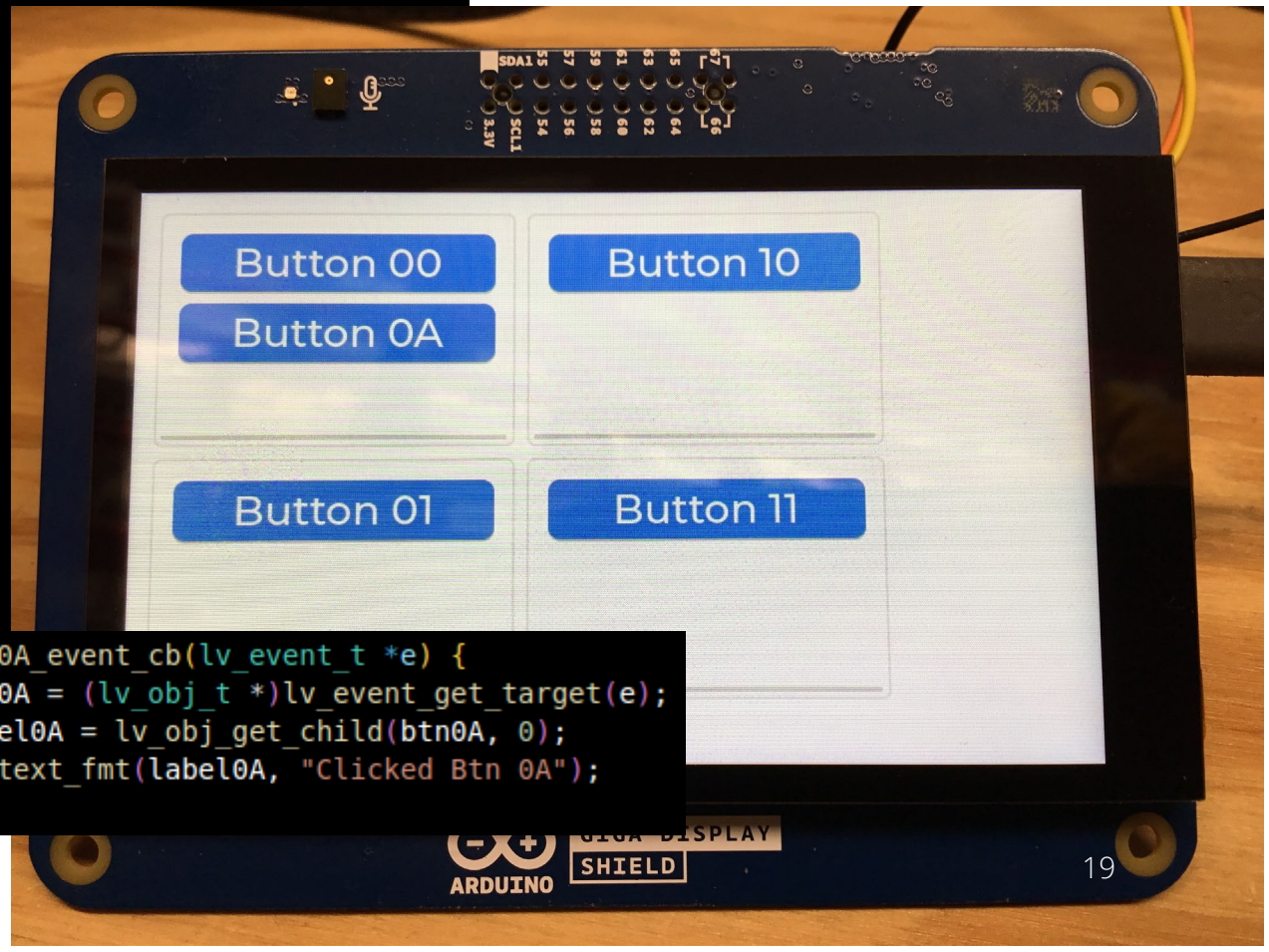
```
102   // Create and Place Elements at Grid Location [1;1]
103   lv_obj_t *label11;
104   obj = lv_obj_create(lvglGrid);
105   lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, 1, 1,  // 0 = column align - 1 = column span
106   | | | | | | | | | | | | | LV_GRID_ALIGN_STRETCH, 1, 1); // 0 = row align - 1 = row span
107   lv_obj_set_flex_flow(obj, LV_FLEX_FLOW_COLUMN);
108   // Create Button and Link to Event Callback
109   lv_obj_t *btn11 = lv_btn_create(obj);
110   lv_obj_set_size(btn11, 265, 50);
111   lv_obj_center(btn11);
112   lv_obj_add_event_cb(btn11, btn11_event_cb, LV_EVENT_CLICKED, NULL);
113   // Create the Button Label
114   label11 = lv_label_create(btn11);
115   lv_label_set_text(label11, "Button 11");
116   lv_obj_center(label11);
```



Add an Additional Button to Grid Location [0;0]

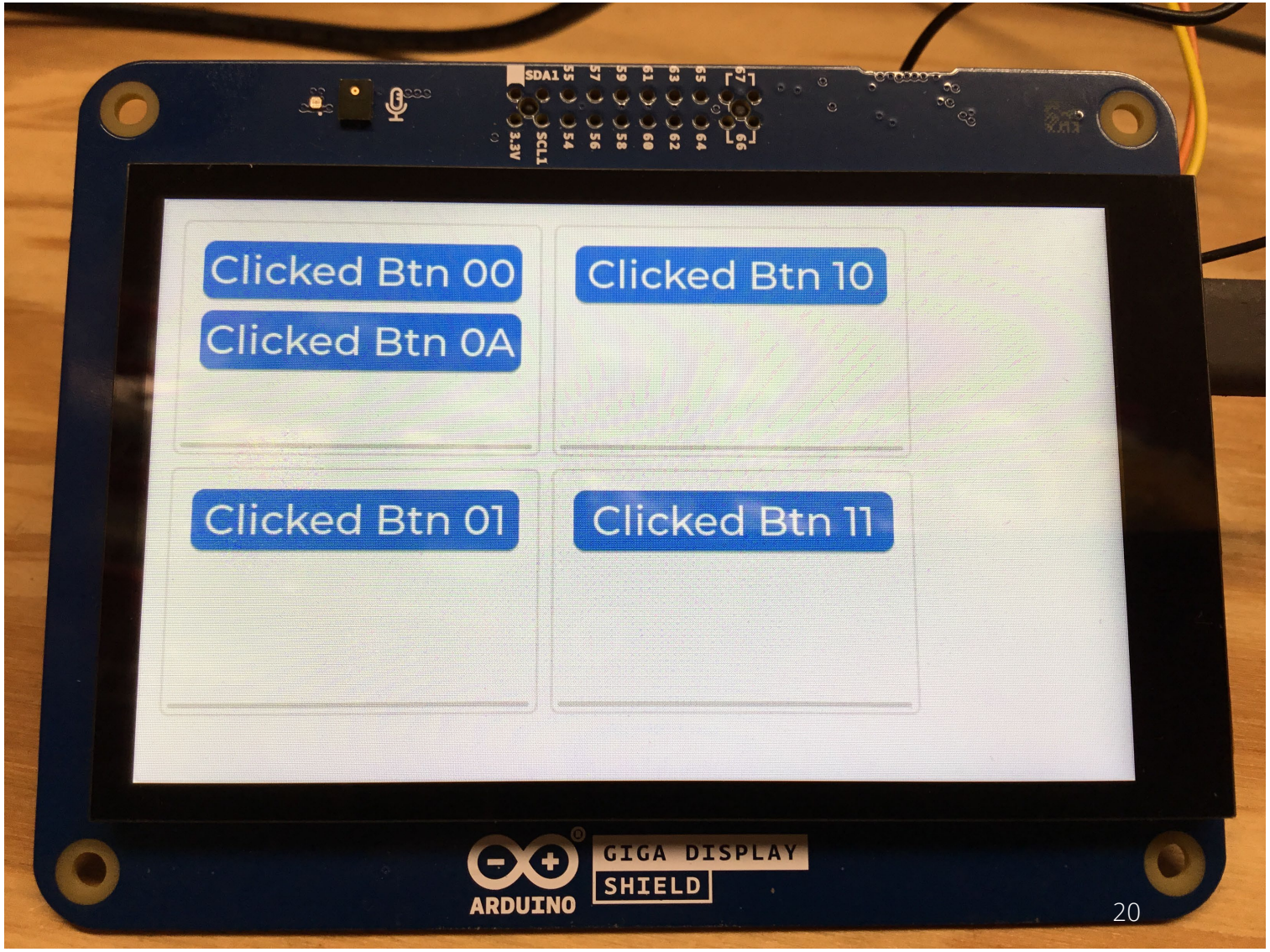
```
57  // Create and Place Elements at Grid Location [0;0]
58  lv_obj_t *label00;
59  lv_obj_t *obj;
60  obj = lv_obj_create(lvglGrid);
61  lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, 0, 1,   // 0 = column align - 1 = column span
62                       LV_GRID_ALIGN_STRETCH, 0, 1);  // 0 = row align - 1 = row span
63  lv_obj_set_flex_flow(obj, LV_FLEX_FLOW_COLUMN);
64  // Create Button and Link to Event Callback
65  lv_obj_t *btn00 = lv_btn_create(obj);
66  lv_obj_set_size(btn00, 265, 50);
67  lv_obj_center(btn00);
68  lv_obj_add_event_cb(btn00, btn00_event_cb, LV_EVENT_CLICKED, NULL);
69  // Create the Button Label
70  label00 = lv_label_create(btn00);
71  lv_label_set_text(label00, "Button 00");
72  lv_obj_center(label00);
73  // Put Another Button in Grid [0;0]
74  lv_obj_t *label0A;
75  lv_obj_t *btn0A = lv_btn_create(obj);
76  lv_obj_set_size(btn0A, 265, 50);
77  lv_obj_center(btn0A);
78  lv_obj_add_event_cb(btn0A, btn0A_event_cb, LV_EVENT_CLICKED, NULL);
79  // Create the Button Label
80  label0A = lv_label_create(btn0A);
81  lv_label_set_text(label0A, "Button 0A");
82  lv_obj_center(label0A);
```

```
31  static void btn0A_event_cb(lv_event_t *e) {
32    lv_obj_t *btn0A = (lv_obj_t *)lv_event_get_target(e);
33    lv_obj_t *label0A = lv_obj_get_child(btn0A, 0);
34    lv_label_set_text_fmt(label0A, "Clicked Btn 0A");
35  }
```



Clickity Click

```
9   // Button Callback Functions
10  static void btn00_event_cb(lv_event_t *e) {
11  |  lv_obj_t *btn00 = (lv_obj_t *)lv_event_get_target(e);
12  |  lv_obj_t *label00 = lv_obj_get_child(btn00, 0);
13  |  lv_label_set_text_fmt(label00, "Clicked Btn 00");
14  }
15  static void btn10_event_cb(lv_event_t *e) {
16  |  lv_obj_t *btn10 = (lv_obj_t *)lv_event_get_target(e);
17  |  lv_obj_t *label10 = lv_obj_get_child(btn10, 0);
18  |  lv_label_set_text_fmt(label10, "Clicked Btn 10");
19  }
20  static void btn01_event_cb(lv_event_t *e) {
21  |  lv_obj_t *btn01 = (lv_obj_t *)lv_event_get_target(e);
22  |  lv_obj_t *label01 = lv_obj_get_child(btn01, 0);
23  |  lv_label_set_text_fmt(label01, "Clicked Btn 01");
24  }
25  static void btn11_event_cb(lv_event_t *e) {
26  |  lv_obj_t *btn11 = (lv_obj_t *)lv_event_get_target(e);
27  |  lv_obj_t *label11 = lv_obj_get_child(btn11, 0);
28  |  lv_label_set_text_fmt(label11, "Clicked Btn 11");
29  }
30  static void btn0A_event_cb(lv_event_t *e) {
31  |  lv_obj_t *btn0A = (lv_obj_t *)lv_event_get_target(e);
32  |  lv_obj_t *label0A = lv_obj_get_child(btn0A, 0);
33  |  lv_label_set_text_fmt(label0A, "Clicked Btn 0A");
34  }
```

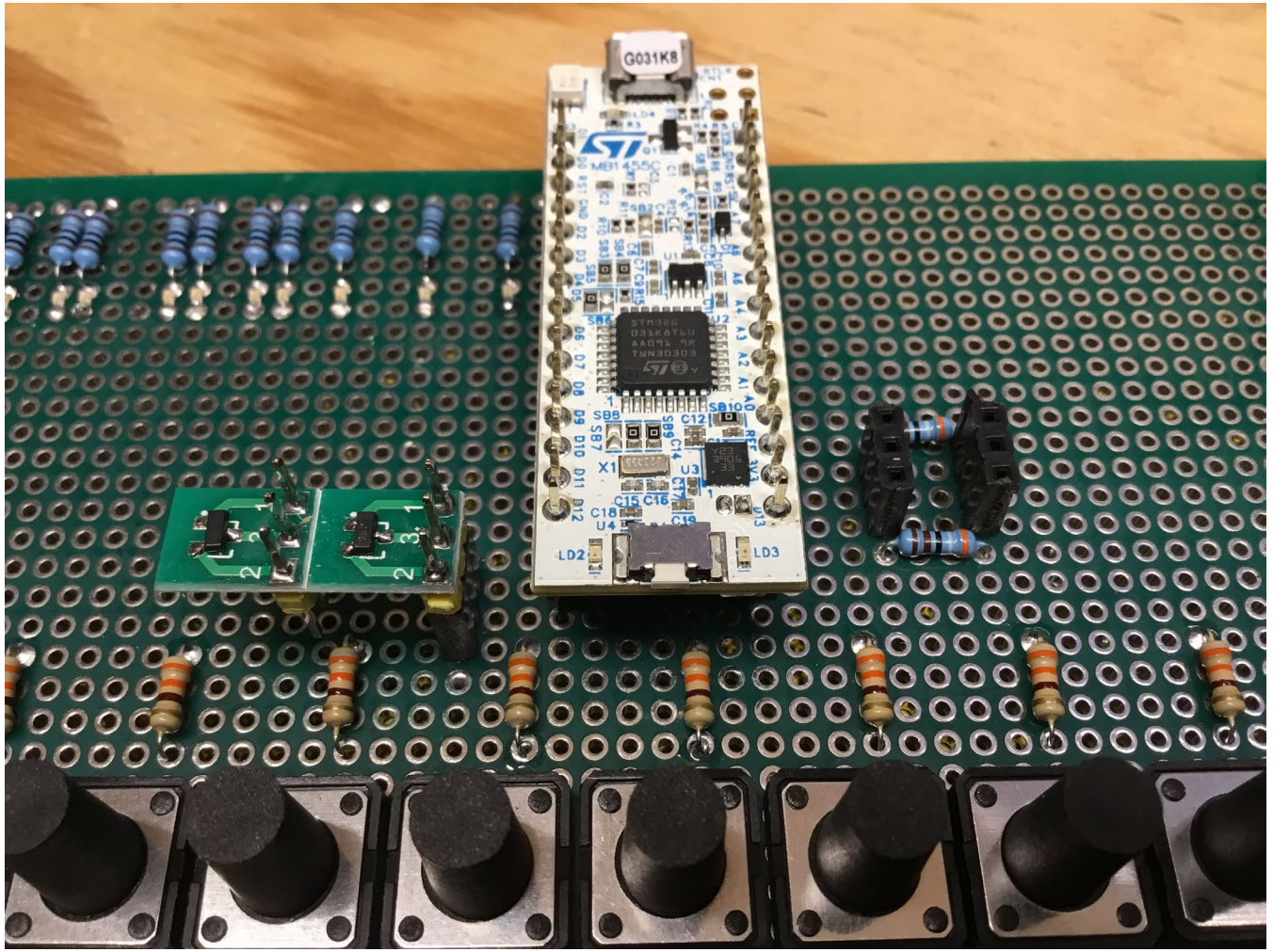


Thank you for attending!!!

Please consider the resources below:

- Today's Download Package
- arduino.cc

MORE TO COME..



CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

DigiKey

informa markets