Touch Screen

ECE 476 Advanced Embedded Systems

Lecture #28

Please visit Bison Academy for corresponding
lecture notes, homework sets, and solutions

Introduction

The ST7796S display includes both a graphics TFT display as well as a resistive touch input. With the touch input, you can set it up so the (x,y) coordinates of the touch screen match up with the graphics screen:

- (0,0) is the upper left corner
- (479,319) is the lower right corner

Multiple touch-points are supported. In theory, interrupts can be set up for when you touch the screen. In the following code, polling is used:

- Each time you call *touch.read_points()*, the number of touch points is returned along with the (x,y) coordinates of each touch point.

Topics for This Lecture

This lecture goes over using the touch-screen along with some basic program:

- Displaying the (x,y) location of up to 3 touch points (base code)
- Using slider bars to change the RGB color of the display
- Playing draw poker, allowing you to select which cards to discard, and
- Programming a numeric keypad with the touch screen

The base code comes from MicroPython Libraries

- https://docs.openmv.io/library/omv.gt911.html#module-gt911

gt911.py library

In order for the following routines to work, *gt911.py* must be downloaded to your Pi-Pico board. This is located on Bison Academy and comes from MicroPython Libraries

Base Code

Initializes the touch screen

Sets it for up to 3 touch points

If you touch the screen

- Displays (x,y) location of each touch point is then displayed in the shell window

```
from time import sleep_ms
from machine import Pin, I2C
from gt911 import GT911

rst_pin = Pin(10, Pin.OUT)
irq_pin = Pin(11)
sda_pin = Pin(8)
scl_pin = Pin(9)

touch = GT911(
    I2C(0, scl=scl_pin, sda=sda_pin, freq=100_000),
    rst_pin,
    irq_pin
)

touch.init(touch_points=3, refresh_rate=50)

while(1):
    num_points, points_data = touch.read_points()
    msg = ''
    for i in range(num_points):
        msg = msg + str(i) + ': ('  + str(points_data[i][0])
        msg = msg + ',' + str(points_data[i][1]) + ')  '
    if(num_points > 0):
        print(msg)
```

Sample Results

Shell Window

- Up to 3 touch points are accepted
- Each time you touch another point, another column is added
- The (x,y) location of each touch-point is displayed
- You get multiple reads while you are touching the display

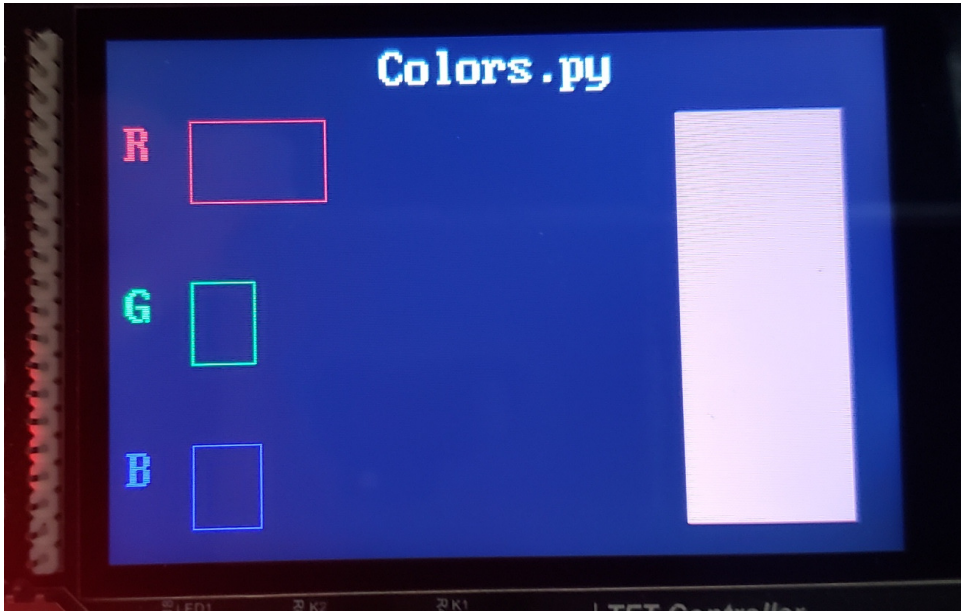
With this code, you can use the touch screen as an input to the Pi-Pico

```
0: (263,77)
0: (263,77)
0: (263,77)
0: (263,77)  1: (92,185)
0: (263,77)  1: (92,185)
0: (263,77)  1: (91,185)
0: (263,77)  1: (473,229)  2: (92,185)
0: (263,78)  1: (473,229)  2: (92,185)
0: (264,79)  1: (468,232)  2: (94,187)
0: (265,83)  1: (455,237)  2: (112,195)
0: (269,87)  1: (447,241)  2: (135,200)
0: (272,91)  1: (434,245)  2: (150,200)
0: (427,246)  1: (157,199)
0: (413,248)  1: (168,195)
0: (409,249)  1: (171,194)
0: (405,251)  1: (176,191)
0: (404,252)  1: (181,186)
0: (404,252)
0: (142,165)
0: (142,165)
```

Example 1: RGB Sliders

Set the RGB level

- Sliders for each color
- Vary from 0 to 255



Code:

- Single touch-point
- X location determines value
- Y location determines R / G / B
- Polling used to read touch-pad

RGB Slider Code

Read (x,y) location

0 < y < 100

- Red

101 < y < 200

- Green

y > 200

- blue

```
while(1):
    num_points, points_data = touch.read_points()
    if(num_points > 0):
        tx = points_data[0][0]
        ty = points_data[0][1]
        print(tx, ty)
        if(ty < 100):
            LCD.Box(50, 50, R+50, 100, Black)
            R = max(0, min(250, tx - 50))
            LCD.Box(50, 50, R+50, 100, Red)
        elif(ty < 200):
            LCD.Box(50, 150, G+50, 200, Black)
            G = max(0, min(250, tx - 50))
            LCD.Box(50, 150, G+50, 200, Green)
        else:
            LCD.Box(50, 250, B+50, 300, Black)
            B = max(0, min(250, tx - 50))
            LCD.Box(50, 250, B+50, 300, Blue)

        Color = LCD.RGB(R, G, B)
        LCD.Solid_Box(350, 50, 450, 300, Color)
```

Example 2: Draw Poker

Play a variation of draw poker

Start of the game:

- Shuffle the deck
- Draw 5 cards

Player can select which card to discard

- Tap the card
- Tap again to deselect

Draw

- Replace the selected cards
- Draw from the top of the deck

Draw Poker Code

Start the game by shuffling the deck

Draw 5 cards

Display the cards on the LCD

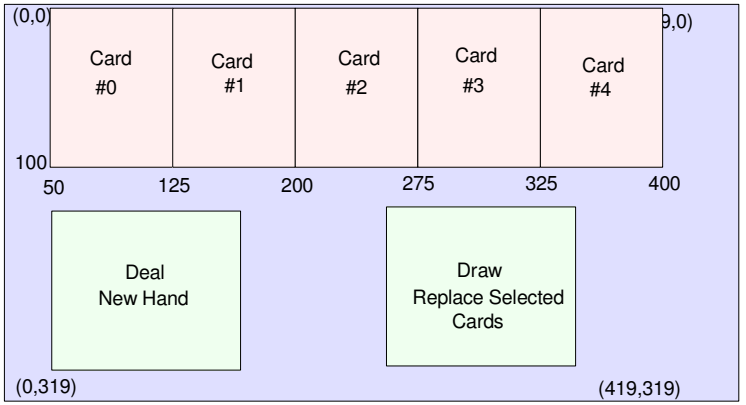
```
Deck = LCD.Shuffle()
Hand = [0]*5;
Value = [0]*5
Suit = [0]*5


for i in range(0,5):
    Hand[i] = Deck[i]
    Value[i] = (Hand[i] % 13) + 1
    Suit[i] = int(Hand[i] / 13) + 1
    LCD.Card(Value[i], Suit[i], 50+i*75, 50)

    LCD.Text2('Deal', 100, 200, White, Black)
    LCD.Text2('Draw', 300, 200, White, Black)

Draw = [0]*5
ptr = 5;
```

Selecting Cards

Tapping the screen determines action

Different fields are used

- Tapping on a card selects / deselects
- Tapping on Deal reshuffles the deck and draws 5 new cards
- Draw replaces selected cards with the top cards of the deck



ST7796S Display

Selecting / Deselecting Cards

Toggle Draw[card]

- 0: keep card
- 1: disard card

```
while(1):
    num_points, points_data = touch.read_points()
    if(num_points > 0):
        tx = points_data[0][0]
        ty = points_data[0][1]
        card = round( (tx-50) / 75)
        card = max(0, min(4, card))
        if(ty < 150):
            Draw[card] = 1 - Draw[card]
```

Drawing Cards

Tap (250,200) to (350,250)

- Discard sekected cards
- Replace with the top cards of the deck
- Multiple draw steps are allowed

```
#Draw
if( (tx > 250)*(tx < 350)*(ty > 200)*(ty < 250) ):
    for i in range(0,5):
        if(Draw[i] == 1):
            Hand[i] = Deck[ptr]
            ptr = ptr + 1
        Draw[i] = 0
        Beeper.value(1)
        sleep_ms(10)
        Beeper.value(0)
```

Tap (50,200) to (150,250)

- Reshuffle the deck
- Draw 5 new cards

```
# Deal
if( (tx > 50)*(tx < 150)*(ty > 200)*(ty < 250) ):
    Deck = LCD.Shuffle()
    for i in range(0,5):
        Hand[i] = Deck[i];
    ptr = 5
    Draw = [0]*5
    for i in range(0,2):
        Beeper.value(1)
        sleep_ms(10)
        Beeper.value(0)
        sleep_ms(100)
```

Numeric Keypad

Create a 4x3 numeric keypad

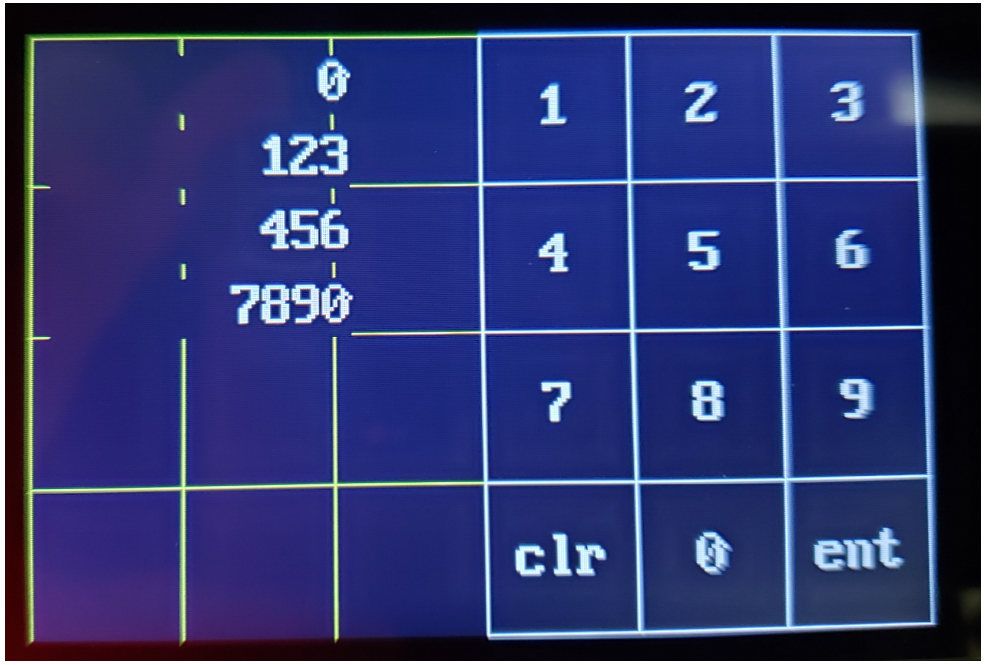
Use it to input numbers

- 0 to
- 9999999

0-9 inputs a number

clr removes the right-most number

enter pushes numbers onto a stack



Place_Button()

Place_Button() places a button along with its label at location (bx, by). The total size of the numeric keypad is (x,y). For example, the keypad shown above is a 6x4 keypad. Number '1' is at location (0,3)

- Top left corner is (0,0)
- Bottom right corner is (5,3)

```
def Place_Button(bx, by, x, y, Label):
    n = len(Label)
    Sx = 480//x
    Sy = 320//y
    x0 = bx*Sx
    y0 = by*Sy
    LCD.Box(x0, y0, min(479, x0 + Sx), min(319, y0 +  Sy), White)
    LCD.Text2(Label, x0+Sx//2-8*n, y0+Sy//2-16, White, Black)
```

Draw_Keyboard()

Draws the keyboard and labels each button.

```
def Draw_Keyboard(x,y):
    Sx = 480//x
    Sy = 320//y
    for i in range(0,y+1):
        LCD.Line(0,i*Sy,479,i*Sy, Grey)
    for i in range(0,x+1):
        LCD.Line(i*Sx,0,i*Sx,319, Grey)

    Place_Button(3, 0, x, y, '1')
    Place_Button(4, 0, x, y, '2')
    Place_Button(5, 0, x, y, '3')

    Place_Button(3, 1, x, y, '4')
    Place_Button(4, 1, x, y, '5')
    Place_Button(5, 1, x, y, '6')

    Place_Button(3, 2, x, y, '7')
    Place_Button(4, 2, x, y, '8')
    Place_Button(5, 2, x, y, '9')

    Place_Button(3, 3, x, y, 'clr')
    Place_Button(4, 3, x, y, '0')
    Place_Button(5, 3, x, y, 'ent')
```

Read_Keypad(x,y):

(x,y) keypad dimension

- 6x4 here

Wait for a press

- num_points > 0

Wait for release

- num_points == 0

Return the key location

- (tx, ty)

```
def Read_Keypad(x, y):
    Sx = 480//x
    Sy = 320//y
    num_points, points_data = touch.read_points()
    while(num_points == 0):
        num_points, points_data = touch.read_points()
        sleep_ms(30)
    tx = points_data[0][0] // Sx
    ty = points_data[0][1] // Sy
    while(num_points > 0):
        num_points, points_data = touch.read_points()
        sleep_ms(30)
    return(tx, ty)
```

Main Routine

Wait for a key press

- Read_Keypad

Interprit Key

- 0..9 = numbers
- clr = delete last digit
- enter = push onto stack

Display the stack

- Only update if changed

```
while(1):
    [x, y] = Read_Keypad(6,4)
    print(x, y)
    if(y==0):
        if(x==3):
            X = 10*X + 1
        elif(x==4):
            X = 10*X + 2
        elif(x==5):
            X = 10*X + 3
    :
    :
    if(flag == 1):
       LCD.Number2(T, 9, 0, 10, 10, White, Black)
       LCD.Number2(Z, 9, 0, 10, 50, White, Black)
       LCD.Number2(Y, 9, 0, 10, 90, White, Black)
       flag = 0
    LCD.Number2(X, 9, 0, 10,130, White, Black)
```

Once you have a keypad, you can use this for entering numbers, building a calculator, etc.

Summary

The touch screen is actually pretty easy to use with polling.

- Each time you call touch.read_points(), the number of touch points and their corresponding (x,y) locations is returned.
- Each time you call Read_Keypad(), the program waits until you press and release on the screen. The (x,y) location of the corresponding key is then returned.

What you do with this is kind of limited by the creativity of the programmer.