adafruit learning system

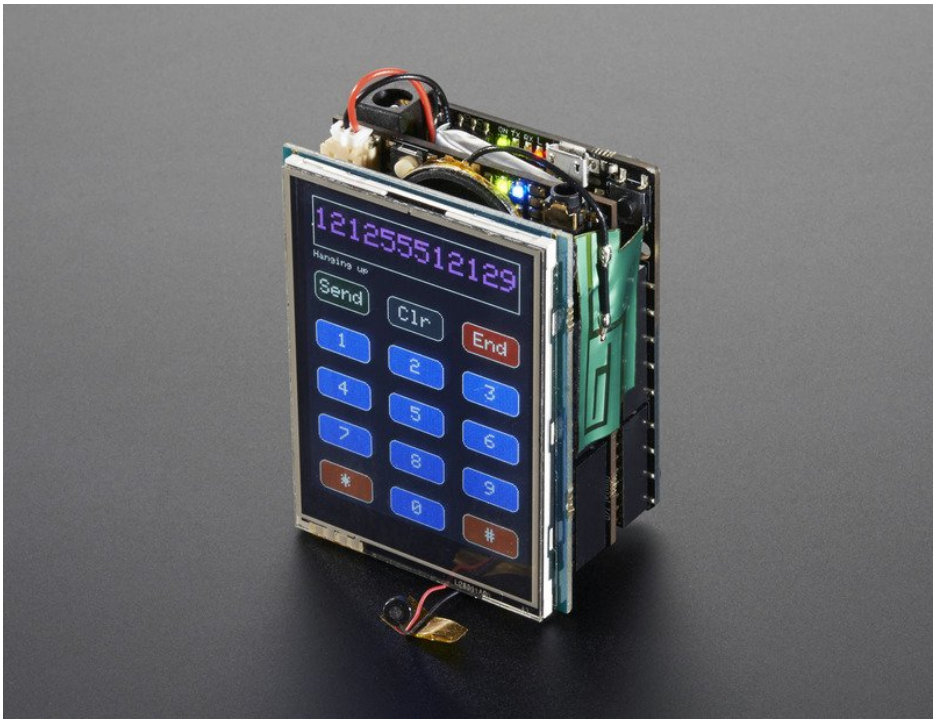
Arduin-o-Phone

Created by lady ada

Last updated on 2019-05-01 09:33:46 PM UTC

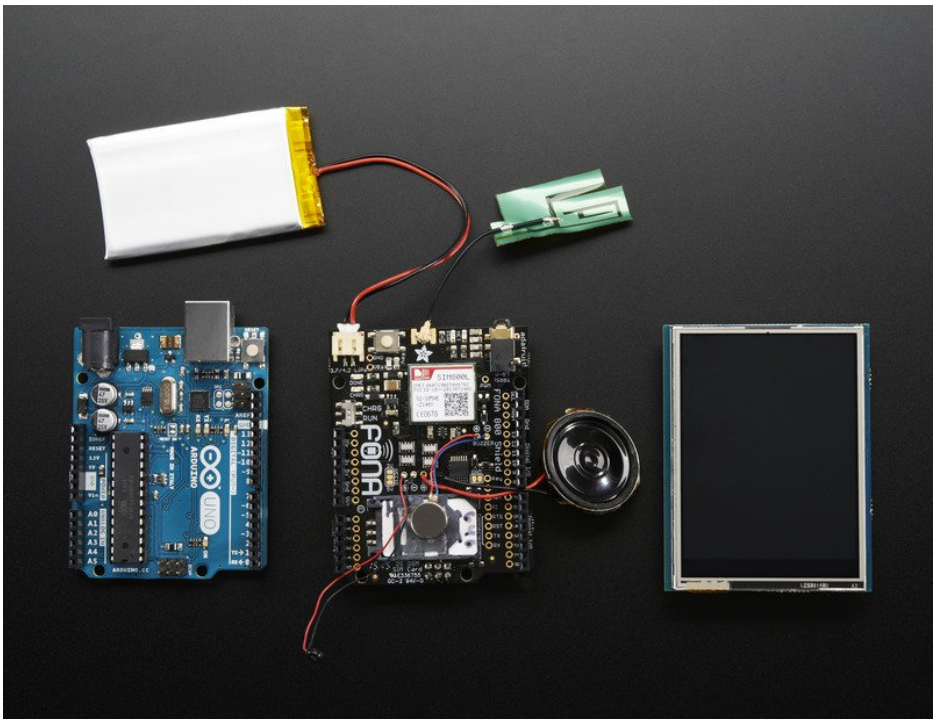
Overview

Apple, Schmapple! Make your *own* phone with an Arduino & Adafruit FONA shield. This project will show you how you can use the FONA shield and a TFT shield stacked on top to make a touch-screen phone that you can program yourself.

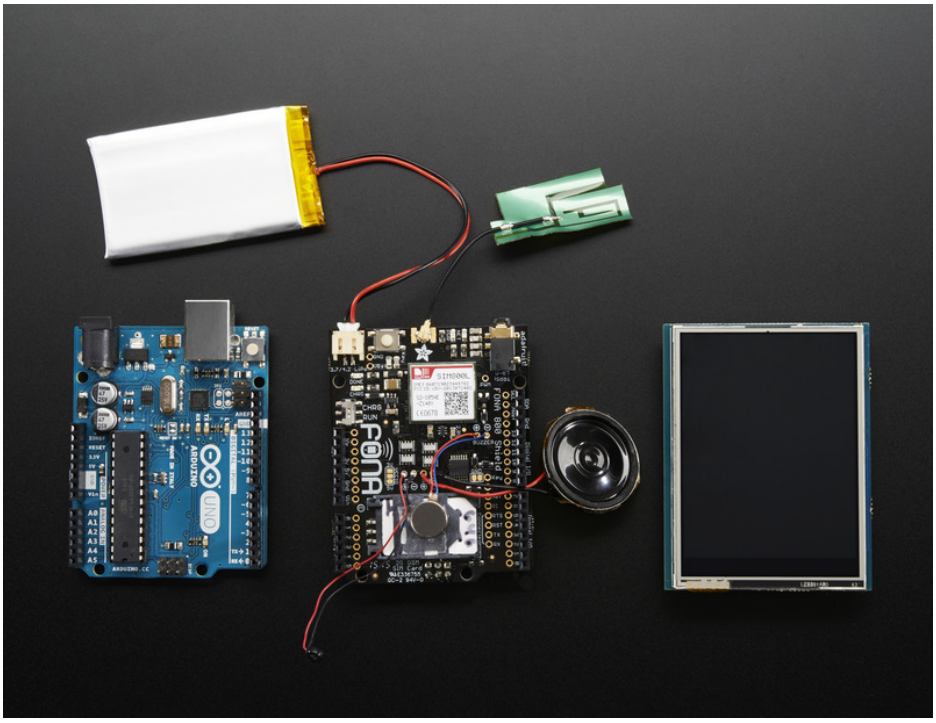


Using Adafruit's great libraries, you can make your own touch-screen dialer in 200 lines of code. Extend it yourself, or keep it simple. Design your own interface or code up a custom app, right from the Arduino IDE

OK sure, you can't check your facebook on the Arduin-o-Phone, but maybe that's not such a bad idea?



Parts and Prep



This project doesn't need a ton of parts, and there's some flexibility based on whether you want to use a headset or speaker&mic.

Start off with the basics:

- Adafruit FONA 800 shield (https://adafru.it/fdQ)
- Adafruit TFT Shield w/Resistive touch (https://adafru.it/fdR)
- Stacking headers (https://adafru.it/dsu)
- GSM SIM card - we suggest a TING SIM (https://adafru.it/fbW), they're a great mobile provider
- 1200mAh LiPoly Battery (http://adafru.it/258), or you could use a 500mAh (https://adafru.it/drL) but 1200 is recommended.
- GSM Antenna, this uFL sticker-type is quite nice (https://adafru.it/fbL)
- Arduino UNO (we later changed our prototype to a Metro so that there was space in the 'shield sandwich' for the lipoly battery (https://adafru.it/fdS))

Then you can either go with speaker + mic for 'hold it up and talk' style

- Wired electret microphone (https://adafru.it/dDa)
- Small metal speaker (https://adafru.it/dDb)

or

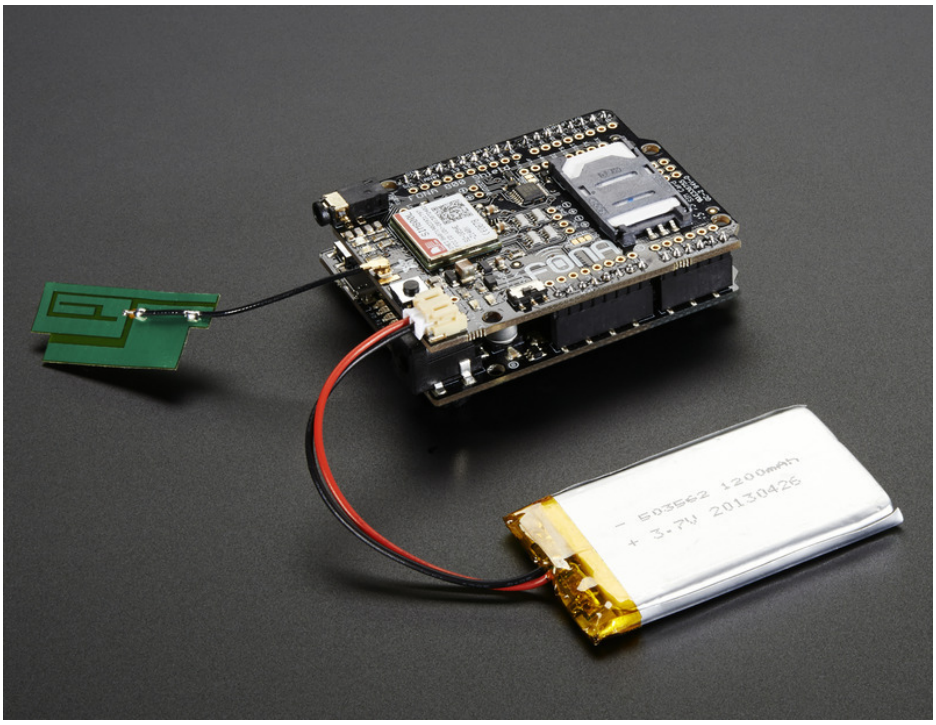
- A plug-in headset (https://adafru.it/fbS)

You'll also need a microUSB cable, soldering iron and solder, and possibly some wire or other basic electronic components!

Learn all about it!

Before you put together the Arduin-o-Phone you'll want to read up a bunch and be familiar with the:

- Arduino and Arduino IDE (https://adafru.it/BQZ)
- 2.8" TFT Shield (https://adafru.it/dpJ) - make sure you get this working nicely first
- FONA 800 Shield (https://adafru.it/CfE) - **solder this one up with the stacking headers so you can stack the TFT shield on top!** Then go through the example and make sure you have it working first



Wire & Test FONA shield

OK now you're ready to put it together. Most of the soldering happens on the FONA shield. Don't forget to solder it with stacking headers!

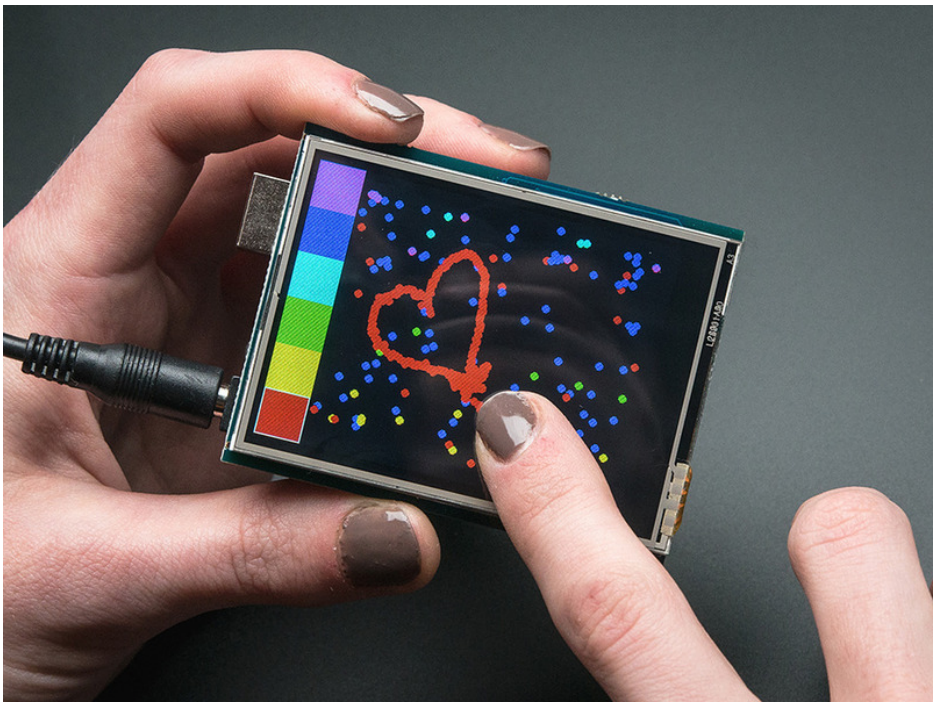
- Attach the Mini Metal Speaker to the **SPKR** pins by soldering the + wire to the + pad, and same with the - wire
- Attach the Wired Electret Microphone to the **MIC** pins by soldering the red wire to + and black wire to -
- Solder the Vibrating Motor Disc to the **Buzzer** pins by soldering the red wire to the + pad, and the black wire to -
- Plug the Lipoly battery into the JST connector, and switch the slide switch to **CHRG**
- Insert the SIM card
- Carefully plug the GSM antenna into the uFL connector, it should snap on nicely
- Plug the shield onto the Arduino and connect the Arduino to your computer

OK now go through the FONA test to make sure your FONA can get onto the cellular network, make phone calls, etc:

https://adafru.it/fdV

https://adafru.it/fdV

Test TFT Shield



Once that works, plug the 2.8" TFT shield on top and run the touch-paint tutorial to make sure you've got the display and touch capability working

https://adafru.it/dLS

https://adafru.it/dLS

All working? OK, *now* you can go to the next step

Arduin-o-Phone Sketch

Now you're ready to upload the Arduin-o-phone sketch. Visit the github repository (https://adafru.it/EGS)for the full code history and details, or simply download by clicking this link:

https://adafru.it/EGT

https://adafru.it/EGT

```
/*
does:
 * can make calls on the speaker & mic
todo:
 * status notification updates in loop()
 * dim screen when no touches in 1 minute
 * receive calls
 * receive texts
 * candy crush

*/


#include <SPI.h>
#include <Wire.h>      // this is needed even tho we aren't using it
#include "Adafruit_GFX.h"
#include "Adafruit_ILI9341.h"
#include <Adafruit_STMPE610.h>
#include <SoftwareSerial.h>
#include "Adafruit_FONA.h"

#define FONA_RX 2
#define FONA_TX 3
#define FONA_RST 4
```

```
#define FONA_RST 4

SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
Adafruit_FONA fona = Adafruit_FONA(FONA_RST);


// For the Adafruit TFT shield, these are the default.
#define TFT_DC 9
#define TFT_CS 10

// Use hardware SPI (on Uno, #13, #12, #11) and the above for CS/DC
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

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

// 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

/****************** 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;

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

/* 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};
Adafruit_GFX_Button buttons[15];

// 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);
}

void setup() {
  Serial.begin(9600);
  Serial.println("Arduin-o-Phone!");

  // clear the screen
  tft.begin();
  tft.fillScreen(ILI9341_BLACK);

  // eep touchscreen not found?
  if (!ts.begin()) {
    Serial.println("Couldn't start touchscreen controller");
    while (1);
  }
  Serial.println("Touchscreen started");

  // 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);

  status(F("Checking for FONA..."));
  // Check FONA is there
  fonaSS.begin(4800); // if you're using software serial

  // See if the FONA is responding
  if (! fona.begin(fonaSS)) {           // can also try fona.begin(Serial1)
    status(F("Couldn't find FONA :("));
    while (1);
  }
  status(F("FONA is OK!"));

  // Check we connect to the network
  while (fona.getNetworkStatus() != 1) {
    status(F("Looking for service..."));
    delay(100);
  }
  status(F("Connected to network!"));
```

```
  status(F("Connected to network!"));

  // set to external mic & headphone
  fona.setAudio(FONA_EXTAUDIO);
}


void loop(void) {
  TS_Point p;

  if (ts.bufferSize()) {
    p = ts.getPoint();
  } 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(") ");
  }

  // 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
    }
  }
}
```

A tour of the code

The sketch will likely get updated with more capability but here's a tour of the code for the initial commit

Setup code

In the **setup()** code we start by initializing the screen first, before the FONA, so that we can display the status updates

```
void setup() {
  Serial.begin(9600);
  Serial.println("Arduin-o-Phone!");

  // clear the screen
  tft.begin();
  tft.fillScreen(ILI9341_BLACK);

  // eep touchscreen not found?
  if (!ts.begin()) {
    Serial.println("Couldn't start touchscreen controller");
    while (1);
  }
  Serial.println("Touchscreen started");
```

We then create 15 buttons using the Adafruit_GFX_Button helper. These make nice rounded rectangle buttons with the text centered. The buttons don't have any sort of touch 'knowledge' but its handy to have a shortcut to drawing a button!

We use the BUTTON_X, _Y, _SPACING #defines we made to define the size of the buttons and the spacing.

```
// 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();
  }
}
```

This just draws the 'outline' of the number text field

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

Now that the screen is drawn, we can check on the FONA, resetting it and turning on the external audio port. We print the *status* of the FONA module code to a little bar of text under the number textfield, using the **status()** helper function

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

status(F("Checking for FONA..."));
// Check FONA is there
fonaSS.begin(4800); // if you're using software serial

// See if the FONA is responding
if (! fona.begin(fonaSS)) {
  status(F("Couldn't find FONA :("));
  while (1);
}
status(F("FONA is OK!"));

// Check we connect to the network
while (fona.getNetworkStatus() != 1) {
  status(F("Looking for service..."));
  delay(100);
}
status(F("Connected to network!"));

// set to external mic & headphone
fona.setAudio(FONA_EXTAUDIO);
```

With that, the Arduin-o-Phone is set up

The main Loop()

Now we can perform a user-interface loop, where we check for button presses and respond

We start by checking for touch presses on the screen, and scaling the presses so they align with the 240x320 size of the display

```
void loop(void) {
  TS_Point p;

  if (ts.bufferSize()) {
    p = ts.getPoint();
  } 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(") ");
  }
```

next, we can use that press location to ask the buttons if that point is inside the bounds of the outline, if so we tell the button it is being pressed

```
  // 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
    }
  }
```

The Adafruit_GFX_Button object keeps track of whether it was *just* pressed or *just* released which is handy so we can only perform an action when it is first pressed or released. For example, if it was just pressed, redraw it so its 'inverted' colors, easy to tell it was 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!
```

Now we can perform actions based on the presses. If its button #3-15, its one of the number or * # buttons, we can add that to the text field array

```
// 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
          }
        }
```

In case we make a typo, the **CLR** button will let you delete a character

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

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

Now that those buttons are taken care of, we should redraw the text field, because it may have more or fewer characters

```
        // 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);
```

Button #2 is **END** and you can always just tell the FONA to hang up

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

Button #0 is **SEND** so you can tell the FONA to call whatever is in that text field

```
        // 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);
        }
```

Finally, we have a delay here to keep the UI button pressing handling from being too 'fast'

```
      delay(100); // UI debouncing
    }
  }
}
```