1. Tổng quan về LCD ILI9341

Màn hình cảm ứng LCD TFT Touch Screen 2.8 inch ILI9341 SPI Interface được sử dụng trong các ứng dụng điều khiển cảm ứng và hiển thị, màn hình sử dụng giao tiếp SPI nên rất dễ giao tiếp, giúp bạn xây dựng giao diện điều kiển cảm ứng

Thông số kỹ thuật:

- Điện áp sử dụng: 3.3~5VDC
- Điện áp giao tiếp: TTL 3.3~5VDC
- IC Driver hiển thị: ILI9341 giao tiếp SPI.
- Cỡ màn hình: 2.8 inch
- Độ phân giải: 240 x 320 pixels
- IC Driver cảm ứng: XPT2046 giao tiếp SPI
- Tích hợp khe thẻ nhớ SD giao tiếp SPI.
- Kích thước hiển thị: 46(W) X 65(H)mm
- Kích thước toàn màn hình: 86 x 50 mm

2. Porting LCD ILI9341 cho BeagleBone Black(BBB)

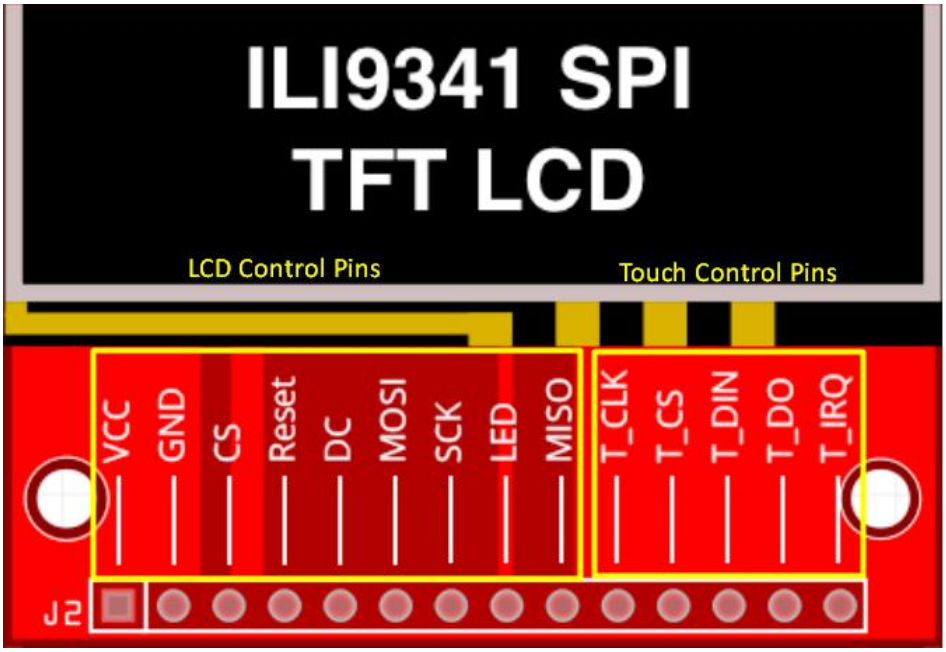
Phần giao diện người dùng (UI) của dự án sẽ được phát triển bằng framework QT. Để màn hình LCD ILI9341 hoạt động được với QT, chúng ta cần porting cho LCD hỗ trợ FrameBuffer của Linux. Ngoài

ra, LCD ILI9341 sử dụng giao thức SPI để gửi nhận dữ liệu nên việc đầu tiên trong quá trình porting và xác định bus SPI và các chân của BBB sẽ sử dụng.

2.1. Xác định bus SPI và các chân của BBB sẽ sử dụng

Mục tiêu của bước này là chọn ra bus SPI và chân có thể sử dụng. Để làm được điều này, chúng ta cần sử dụng tài liệu sơ đồ chân của LCD ILI9431, pinmuxing table và schematic của BBB.

Dựa vào tài liệu sơ đồ chân của LCD ILI9341, để điều khiển được LCD, ngoài 4 chân SPI (SCL, MISO, MOSI, CS), chúng ta còn cần thêm 2 chân GPIO: Reset và DC



Dựa vào pinmuxing table, chúng ta thấy rằng 4 chân P9_28, P9_29, P9_30, và P9_31 có thể sử dụng cho bus SPI1 nếu cấu hình ở mode 3, và 2 chân P9_25, và P9_27 có thể sử dụng cho GPIO nếu cấu hình ở mode 7. Tham chiếu đến schematic, ta cũng thấy rằng những chân này không được kết nối vào bất kỳ thiết bị nào.

Sơ đồ kết nối chân LCD ILI9341 và BBB

BBB	LCD ILI9341
VCC 3.3v	VCC
GND	GND
P9_28	CS

P9_25	Reset
P9_27	DC
P9_30	MOSI
P9_31	SCK
VCC 3.3v	LED
P9_29	MISO

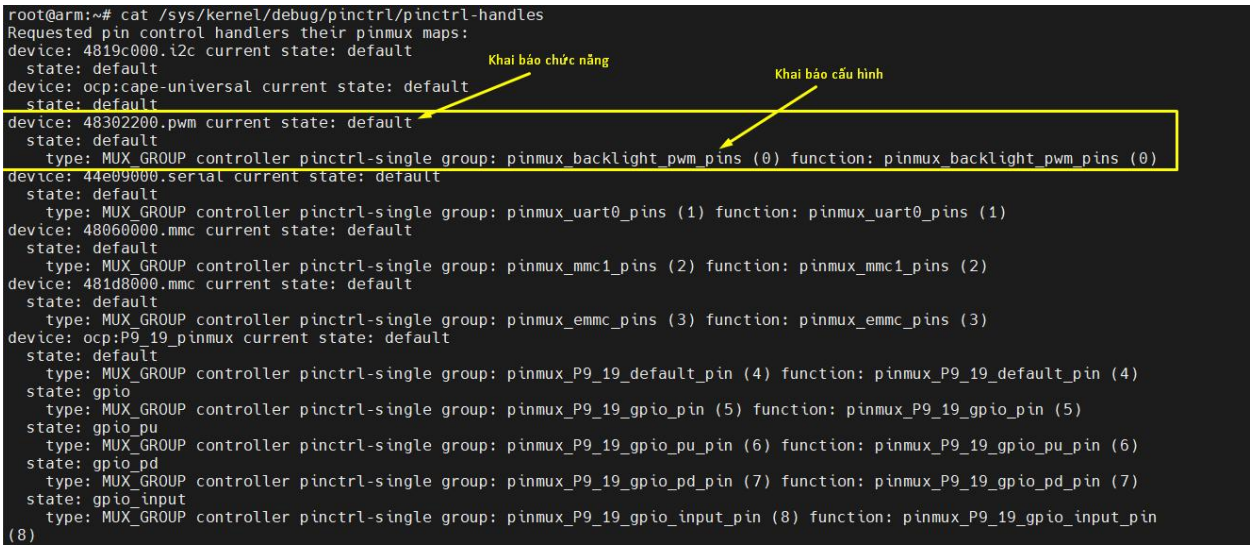
Tiếp theo, ta cần cấu hình pinctrl cho những chân này trong device tree.

2.2. Cấu hình pinctrl trong device tree

Mục tiêu của bước này là cấu hình 4 chân SPI1: P9_28, P9_29, P9_30, và P9_31 hoạt động ở mode 3, và 2 chân GPIO: P9_25, và P9_27 hoạt động ở mode 7

Trước khi thực hiện cấu hình, chúng ta cần kiểm tra xem 6 chân này đã được cấu hình chưa bằng cách đọc nội dung của file `pinctrl-handles` bằng command

`'cat /sys/kernel/debug/pinctrl/pinctrl-handles'`




```
root@arm:~# cat /sys/kernel/debug/pinctrl/pinctrl-handles
Requested pin control handlers their pinmux maps:
device: 4819c000.i2c current state: default
  state: default
device: ocp:cape-universal current state: default
  state: default
device: 48302200.pwm current state: default
  state: default
    type: MUX_GROUP controller pinctrl-single group: pinmux_backlight_pwm_pins (0) function: pinmux_backlight_pwm_pins (0)
device: 44e09000.serial current state: default
  state: default
    type: MUX_GROUP controller pinctrl-single group: pinmux_uart0_pins (1) function: pinmux_uart0_pins (1)
device: 48060000.mmc current state: default
  state: default
    type: MUX_GROUP controller pinctrl-single group: pinmux_mmc1_pins (2) function: pinmux_mmc1_pins (2)
device: 481d8000.mmc current state: default
  state: default
    type: MUX_GROUP controller pinctrl-single group: pinmux_emmc_pins (3) function: pinmux_emmc_pins (3)
device: ocp:P9_19_pinmux current state: default
  state: default
    type: MUX_GROUP controller pinctrl-single group: pinmux_P9_19_default_pin (4) function: pinmux_P9_19_default_pin (4)
  state: gpio
    type: MUX_GROUP controller pinctrl-single group: pinmux_P9_19_gpio_pin (5) function: pinmux_P9_19_gpio_pin (5)
  state: gpio_pu
    type: MUX_GROUP controller pinctrl-single group: pinmux_P9_19_gpio_pu_pin (6) function: pinmux_P9_19_gpio_pu_pin (6)
  state: gpio_pd
    type: MUX_GROUP controller pinctrl-single group: pinmux_P9_19_gpio_pd_pin (7) function: pinmux_P9_19_gpio_pd_pin (7)
  state: gpio_input
    type: MUX_GROUP controller pinctrl-single group: pinmux_P9_19_gpio_input_pin (8) function: pinmux_P9_19_gpio_input_pin
(8)
```

Trong cấu hình pinctrl, mỗi node cấu hình sẽ được liên kết với một node chức năng và điều này có thể được quan sát khi đọc nội dung của file `pinctrl-handles`. Do đó, các node cấu hình xuất hiện trong file này cho thấy chúng đang được áp dụng trên BeagleBone Black (BBB). Dựa trên các thông tin đó, ta có thể đối chiếu với device tree để kiểm tra trạng thái cấu hình của 4 chân SPI1: P9_28, P9_29,

P9_30, và P9_31 và 2 chân GPIO: P9_25, và P9_27 . Trong trường hợp hiện tại, những chân trên đang được sử dụng bởi node `mcasp0_pins` nằm trong `am335x-boneblack-hdmi.dtsi`. Vì vậy chúng ta sẽ cần disable node `mcasp0` là node chức năng đang sử dụng node `mcasp0_pins` nằm trong `am335x-boneblack-hdmi.dtsi`

```
97 &mcasp0 {
98         #sound-dai-cells = <0>;
99         pinctrl-names = "default";
100        pinctrl-0 = <&mcasp0_pins>;
101 //     status = "okay";
102        status = "disable";
103        op-mode = <0>;  /* MCASP_IIS_MODE */
104        tdm-slots = <2>;
105        serial-dir = <  /* 0: INACTIVE, 1: TX, 2: RX */
106                        0 0 1 0
107                >;
108        tx-num-evt = <32>;
109        rx-num-evt = <32>;
110 };
```

Tiếp tục, để thực hiện cấu hình pinctrl chúng ta truy cập vào file ‘`build/tmp/work-shared/beaglebone-yocto/kernel-source/arch/arm/boot/dts/am335x-boneblack.dts`’và thêm đoạn code sau:

```
&am33xx_pinmux {
        bb_spi1_pins: pinmux_bb_spi1_pins {
                pinctrl-single,pins = <
                        AM33XX_PADCONF(AM335X_PIN_MCASP0_ACLKX, PIN_INPUT_PULLUP,
MUX_MODE3)     /* mcasp0_aclkx.spi1_sclk,  INPUT_PULLUP | MODE3,   */
                        AM33XX_PADCONF(AM335X_PIN_MCASP0_FSX, PIN_INPUT_PULLUP,
MUX_MODE3)          /* mcasp0_fsx.spi1_d0,      INPUT_PULLUP | MODE3,   */
                        AM33XX_PADCONF(AM335X_PIN_MCASP0_AXR0, PIN_OUTPUT_PULLUP,
MUX_MODE3)     /* mcasp0_axr0.spi1_d1,    OUTPUT_PULLUP | MODE3,   */
                        AM33XX_PADCONF(AM335X_PIN_MCASP0_AHCLKR, PIN_OUTPUT_PULLUP,
MUX_MODE3)     /* mcasp0_ahclkr.spi1_cs0, OUTPUT_PULLUP | MODE3,   */
                >;
        };

        bb_lcd_pins: pinmux_bb_lcd_pins {
                pinctrl-single,pins = <
```

```
                        AM33XX_PADCONF(AM335X_PIN_MCASP0_FSR, PIN_OUTPUT_PULLUP,
MUX_MODE7)          /* gpio, dc,    OUTPUT_PULLUP | MODE07       */
                        AM33XX_PADCONF(AM335X_PIN_MCASP0_AHCLKX, PIN_OUTPUT_PULLUP,
MUX_MODE7)    /* gpio, reset,       OUTPUT_PULLUP | MODE07       */
                    >;
            };
        };
        &spi1 {
            #address-cells = <1>;
            #size-cells = <0>;
            status = "okay";
            pinctrl-names = "default";
            pinctrl-0 = <&bb_spi1_pins>;
        };
```

Tiếp theo, chúng ta sẽ porting LCD ILI9341 để hỗ trợ FrameBuffer.

2.3. Porting LCD ILI9341 để hỗ trợ FrameBuffer

Mục tiêu của bước này là enable tinydrm_ili9341 driver và thêm node tương ứng vào device tree để LCD ILI9341 có thể hoạt động với FrameBuffer.

Trước tiên, chúng ta thêm node vào device tree bằng cách truy cập vào file 'build/tmp/work-shared/beaglebone-yocto/kernel-source/arch/arm/boot/dts/am335x-boneblack.dts'. Và thêm đoạn code sau vào node &spi1.

```
        &spi1 {
            #address-cells = <1>;
            #size-cells = <0>;

            status = "okay";
            pinctrl-names = "default";
            pinctrl-0 = <&bb_spi1_pins>;

            lcd@0{
                compatible = "adafruit,yx240qv29";
```

```
                reg = <0>;
                pinctrl-names = "default";
                pinctrl-0 = <&bb_lcd_pins>;
                spi-max-frequency = <24000000>;
                rotation = <90>;
                dc-gpios     = <&gpio3 19 0>;    // lcd dc    P9.27 gpio3[19]
                reset-gpios = <&gpio3 21 0>;    // lcd reset P9.25 gpio3[21]
                status = "okay";
        };
};
```

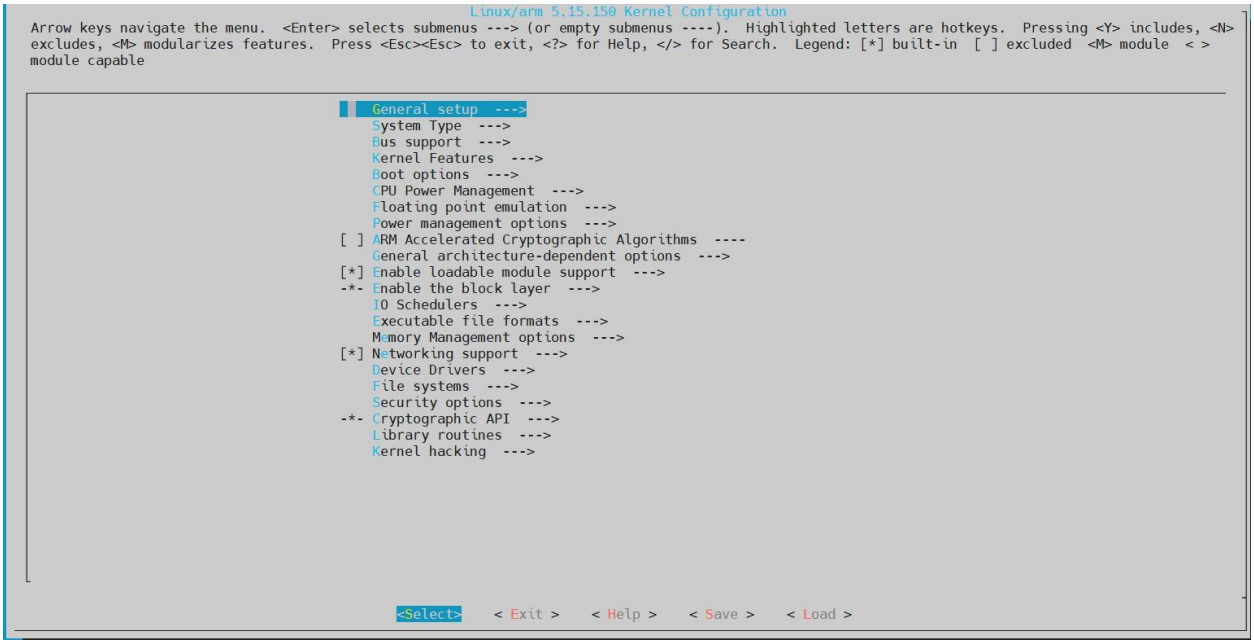
Trong đó:

- spi-max-frequency = <24000000> : Tốc độ truyền dữ liệu SPI, ở đây là 24MHz. Tốc độ cao giúp hiển thị nhanh hơn, nhưng phải nằm trong khả năng hỗ trợ của màn hình.
- rotatation = <90> : Xoay màn hình theo góc 90 độ để hiển thị đúng hướng.
- reset-gpios và dc-gpios :Khai báo các chân GPIO dùng để điều khiển:

Tiếp theo, chúng ta sẽ sử dụng kernel menuconfig để enable tinydrm_ili9341driver việc này có thể thực hiện bằng command sau:

```
$ bitbake -c menuconfig -f virtual/kernel
```

Lúc này trên màn hình console sẽ mở ra giao diện tương tác



Dùng phím mũi tên để di chuyển và nhấn Enter để chọn. Làm theo các bước sau để enable `tinydrm_ili9341` driver

-> Device Drivers

-> Device Drivers

-> Graphics support

-> DRM support for ILI9341 display panels (CONFIG_TINYDRM_ILI9341 [=m])

Thực hiện rebuild lại kernel và image bằng command sau:

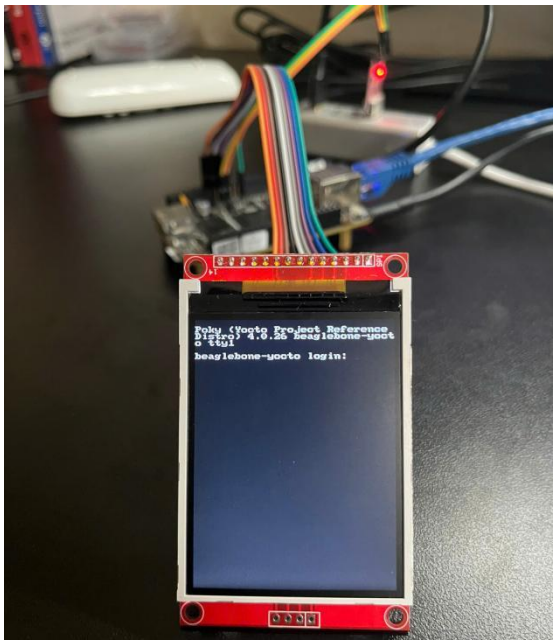
```
$ bitbake -f -c compile linux-yocto
$ bitbake -c deploy linux-yocto
$ bitbake core-image-full-cmdline
```

Tiếp theo, chúng ta sẽ cần flash image vào sdcard và kiểm tra hoạt động của LCD

2.4. Kiểm tra hoạt động của LCD

Mục tiêu của bước này là kiểm tra hoạt động của LCD ILI9341

Trong quá trình boot, nếu node lcd@0 trong device tree tương thích với tinydrm_ili9341driver thì trên LCD ILI9341 cũng sẽ hiển thị console



3. Porting touchscreen cho BeagleBone Black(BBB)

Phần giao diện người dùng (UI) của dự án ngoài việc hiện thị thì nó còn cần cho phép tương tác. Để màn hình LCD ILI9341 làm được điều này, chúng ta cần porting cho phần touching hỗ trợ Input Touchscreen của Linux. Phần touching của LCD cũng sử dụng giao thức SPI để gửi nhận dữ liệu nên việc đầu tiên trong quá trình porting và xác định bus SPI và các chân của BBB sẽ sử dụng.

3.1. Xác định bus SPI và các chân của BBB sẽ sử dụng

Để giao tiếp với touch , ngoài 4 chân dùng cho SPI, chúng ta cần thêm 1 chân GPIO cho tín hiệu ngắt (T_IRQ). Tương tự như bước 2.1, ta xác định rằng các chân P9_17, P9_18, P9_21 và P9_22 trên BeagleBone Black có thể được cấu hình ở mode 0 để sử dụng cho SPI0. Đồng thời, chân P9_23 có thể cấu hình ở mode 7 để hoạt động như một chân GPIO dùng cho T_IRQ.

Sơ đồ kết nối chân giữa touch và BBB

BBB	Touch ILI9341
P9_17	T_CS
P9_18	T_D0
P9_21	T_DIN
P9_22	T_CLK
P9_23	T_IRQ

Tiếp theo, ta cần cấu hình pinctrl cho những chân này trong device tree

3.2. Cấu hình pinctrl trong device tree

Chúng ta cần kiểm tra xem 5 chân này đã được cấu hình hay chưa, tương tự như đã thực hiện ở phần 2.2. Chúng ta sẽ thấy rằng các chân này chưa được cấu hình trong device tree, ta có thể sử dụng chúng mà không cần disable bất kỳ node nào sẵn có.

Để cấu hình các chân này thông qua pinctrl, hãy truy cập file: build/tmp/work-shared/beaglebone-yocto/kernel-source/arch/arm/boot/dts/am335x-boneblack.dts

Và thêm đoạn mã cấu hình sau vào phần phù hợp trong device tree.

```
&am33xx_pinmux {
        bb_spi0_pins: pinmux_bb_spi0_pins {
```

```
                pinctrl-single,pins = <
                        AM33XX_PADCONF(AM335X_PIN_SPI0_SCLK, PIN_INPUT_PULLUP,
MUX_MODE0)      /* spi0_sclk,   INPUT_PULLUP | MODE0,       P9_22 */
                        AM33XX_PADCONF(AM335X_PIN_SPI0_D0,   PIN_OUTPUT_PULLUP,
MUX_MODE0)      /* spi0_d0,     INPUT_PULLUP | MODE0,       P9_21 */
                        AM33XX_PADCONF(AM335X_PIN_SPI0_D1,   PIN_INPUT_PULLUP,
MUX_MODE0)      /* spi0_d1,     OUTPUT_PULLUP | MODE0,      P9_18 */
                        AM33XX_PADCONF(AM335X_PIN_SPI0_CS0,  PIN_OUTPUT_PULLUP,
MUX_MODE0)      /* spi0_cs0,    OUTPUT_PULLUP | MODE0,      P9_17 */
                        >;
                };
                bb_touchscreen_pins: pinmux_bb_touchscreen_pins {
                        pinctrl-single,pins = <
                                AM33XX_PADCONF(AM335X_PIN_GPMC_A1,  PIN_INPUT_PULLUP, MUX_MODE7)
        /* gpmc_a1,    OUTPUT_PULLUP | MODE0,       P9_23 */
                        >;
                };
        };
        &spi0 {
                #address-cells = <1>;
                #size-cells = <0>;

                status = "okay";
                pinctrl-names = "default";
                pinctrl-0 = <&bb_spi0_pins>;
                ti,pindir-d0-out-d1-in = <1>;
        };
```

Tiếp theo chúng ta cần porting touch để hỗ trợ Input TouchScreen.

3.3 Porting Touch ILI9341 để hỗ trợ Input TouchScreen

Mục tiêu của bước này là enable TOUCHSCREEN_ADS7846 driver và thêm node tương ứng vào device tree để touch có thể hoạt động với Input TouchScreen.

Trước tiên, chúng ta thêm node vào device tree bằng cách truy cập vào file 'build/tmp/work-shared/beaglebone-yocto/kernel-source/arch/arm/boot/dts/am335x-boneblack.dts' . Và thêm đoạn code sau vào node &spi0.

```
&spi0 {
        #address-cells = <1>;
        #size-cells = <0>;

        status = "okay";
        pinctrl-names = "default";
        pinctrl-0 = <&bb_spi0_pins>;
        ti,pindir-d0-out-d1-in = <1>;

        touchscreen@0 {
                compatible = "ti,ads7846";
                spi-max-frequency = <100000>;
                interrupts = <17 IRQ_TYPE_EDGE_FALLING>;
                interrupt-parent = <&gpio1>;
                pendown-gpio = <&gpio1 17 0>;
                reg = <0>;
                #addr-size = <2>;
                #page-size = <32>;
                pinctrl-names = "default";
                pinctrl-0 = <&bb_touchscreen_pins>;

                /* driver defaults */
                ti,x-min = /bits/ 16 <0xB0>;
                ti,y-min = /bits/ 16 <0x231>;
                ti,x-max = /bits/ 16 <0xF49>;
                ti,y-max = /bits/ 16 <0xF6B>;
                ti,pressure-min = /bits/ 16 <0>;
```

```
                ti,pressure-max = /bits/ 16 <0xFF>;
                ti,x-plate-ohms = /bits/ 16 <0x96>;
                ti,swap_xy = /bits/ 16 <0>;
                ti,keep_vref_on = /bits/ 16 <0>;
                ti,cs = /bits/ 16 <1>;
                linux,wakeup;
        };
};
```

Tiếp theo, chúng ta sẽ sử dụng kernel menuconfig để enable touchscreen_ads7846 driver bằng command sau:

```
$ bitbake -c menuconfig -f virtual/kernel
```

Dùng phím mũi tên để di chuyển và nhấn Enter để chọn. Làm theo các bước sau để enable touchscreen_ads7846 driver

```
-> Device Drivers
        -> Input device support
                -> Generic input layer (needed for keyboard, mouse, ...) (INPUT [=y])
                        -> Touchscreens (INPUT_TOUCHSCREEN [=y])
                                -> ADS7846/TSC2046/AD7873 and AD(S)7843 based
touchscreens (TOUCHSCREEN_ADS7846 = [y])
```

Tiếp theo, thêm evtest package để có thể sử dụng command line ‘evtest’ cho việc debug. Sau đó, rebuild lại image để update thay đổi.

```
$ bitbake -f -c compile linux-yocto
$ bitbake -c deploy linux-yocto
$ bitbake core-image-full-cmdline
```

Tiếp theo, chúng ta sẽ cần flash image vào sdcard và kiểm tra hoạt động của touch

3.3. Kiểm tra hoạt động của touch