



Laboratorio 10:

Conversión analógico-digital

entrada por un touchscreen

Programación de sistemas y dispositivos

José Manuel Mendías Cuadros

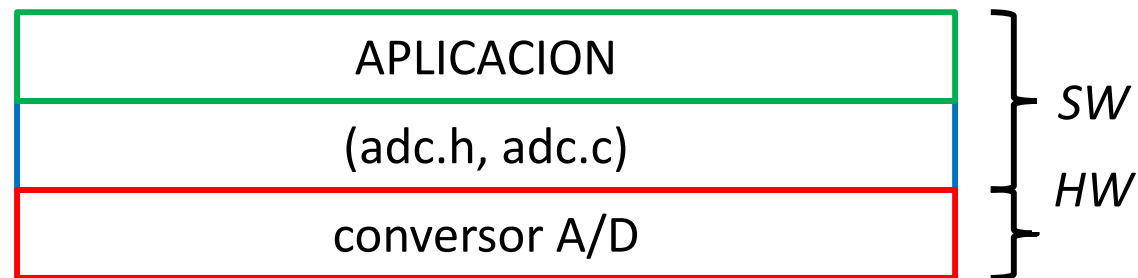
*Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid*



Presentación



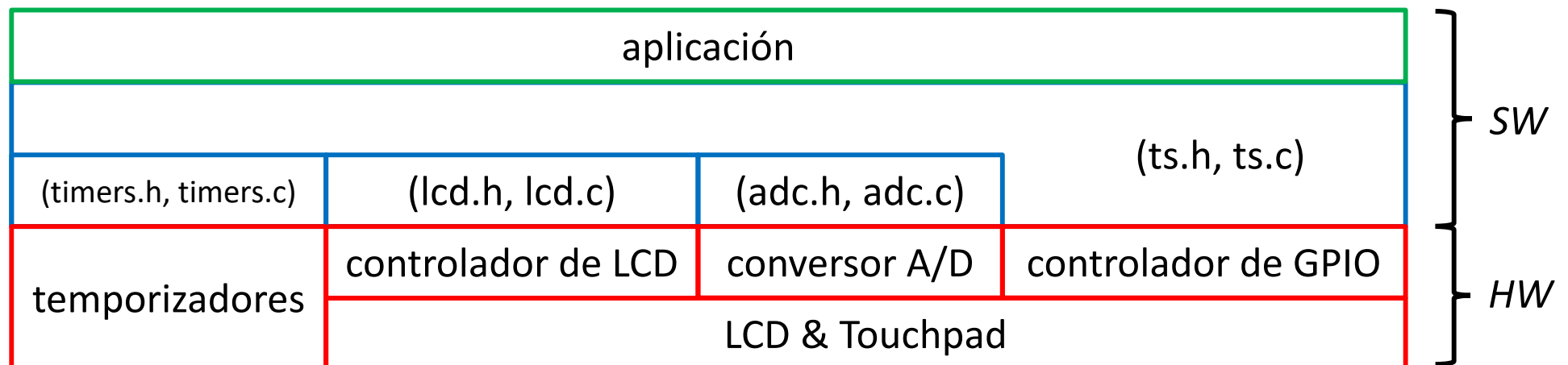
- Desarrollar una capa de firmware para leer señales analógicas
 - Implementaremos 7 funciones :
 - Inicialización del conversor A/D: `adc_init`
 - Encendido/apagado del conversor A/D: `adc_on` / `adc_off`
 - Consulta de estado del conversor A/D: `adc_status`
 - Lectura de una entrada analógica: `adc_getSample`
 - Activación/desactivación de interrupciones por fin de conversión A/D, así como instalación de la RTI que las atenderá: `adc_open` / `adc_close`



Presentación



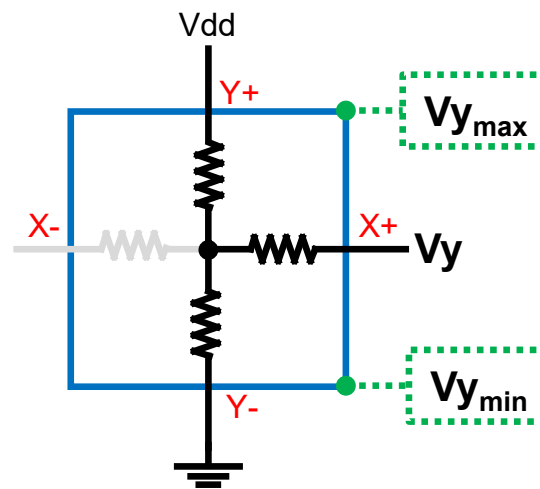
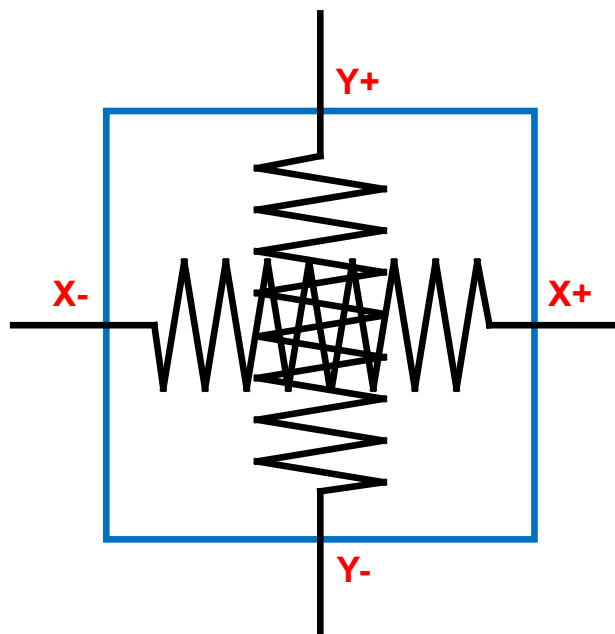
- Desarrollar una capa de firmware para leer datos de una touchscreen
 - Implementaremos 8 funciones sin gestión del tiempo:
 - Inicialización y calibrado de la touchscreen: `ts_init`
 - Lectura del estado de la touchscreen: `ts_pressed`
 - Encendido/apagado/consulta de estado de la touchscreen: `ts_on` / `ts_off` / `ts_status`
 - Espera por presión y lectura de la touchscreen : `ts_getpos`
 - Activación/desactivación de interrupciones por presión de la touchscreen, así como instalación de la RTI que las atenderá: `ts_open` / `ts_close`
 - Implementaremos 2 funciones con gestión del tiempo:
 - Espera por presión, lectura de la touchscreen y medida del tiempo: `ts_getpostime`
 - Espera con timeout por presión y lectura de la touchscreen: `ts_timeout_getpos`



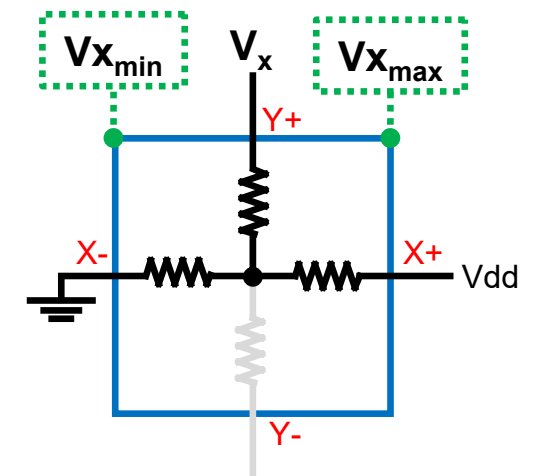
Touchscreen



- El LCD de la placa **S3CEV40** es una **touchscreen resistiva de 4 terminales**:
 - Tiene 2 capas de material transparente resistivo separadas por una capa de aire.
 - Cuando se presiona un punto, ambas capas resistivas hacen contacto.
 - Si se aplica tensión en los terminales de una capa se crea un divisor de tensión.
 - El voltaje en el punto de contacto es proporcional a su posición en el touchpad.
 - El voltaje será mayor si el punto está más próximo al terminal de mayor tensión.



poniendo tensión en la capa Y,
la posición del punto de contacto
se calcula según el valor de V_y

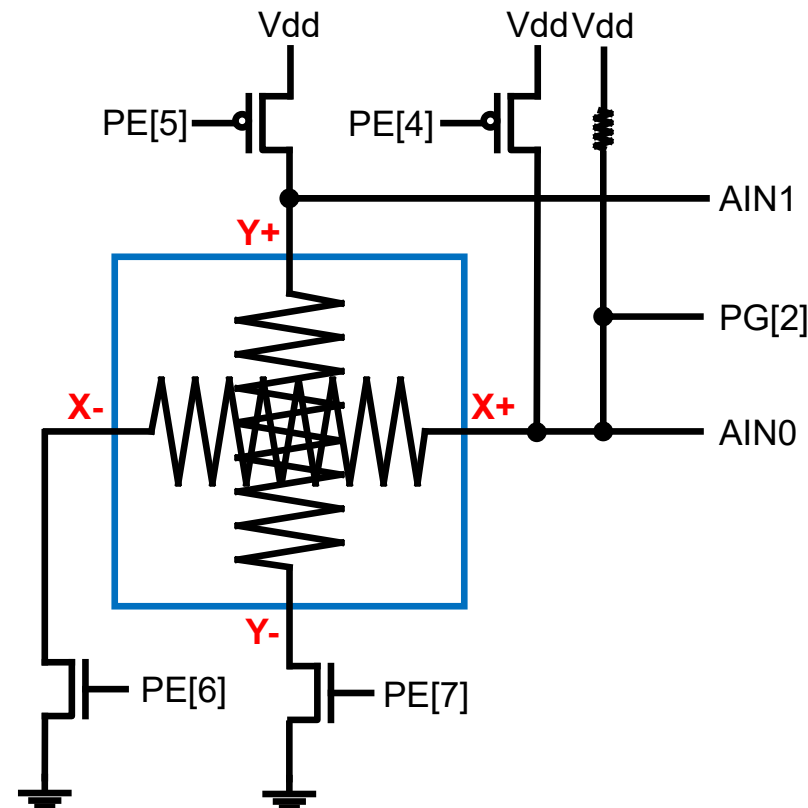


poniendo tensión en la capa X,
la posición x del punto de contacto
se calcula según el valor de V_x

Touchscreen



- Los **terminales** de la touchscreen **están conectados** en la placa a:
 - Pares de transistores MOSFET que permiten dar selectivamente tensión a cada capa.
 - Controlados por los **bits 7..4 del puerto E**
 - A las **entradas AIN0 y AIN1 del conversor A/D** para medir voltajes.
 - Al **bit 2 del puerto G** (configurable como fuente externa de interrupción).

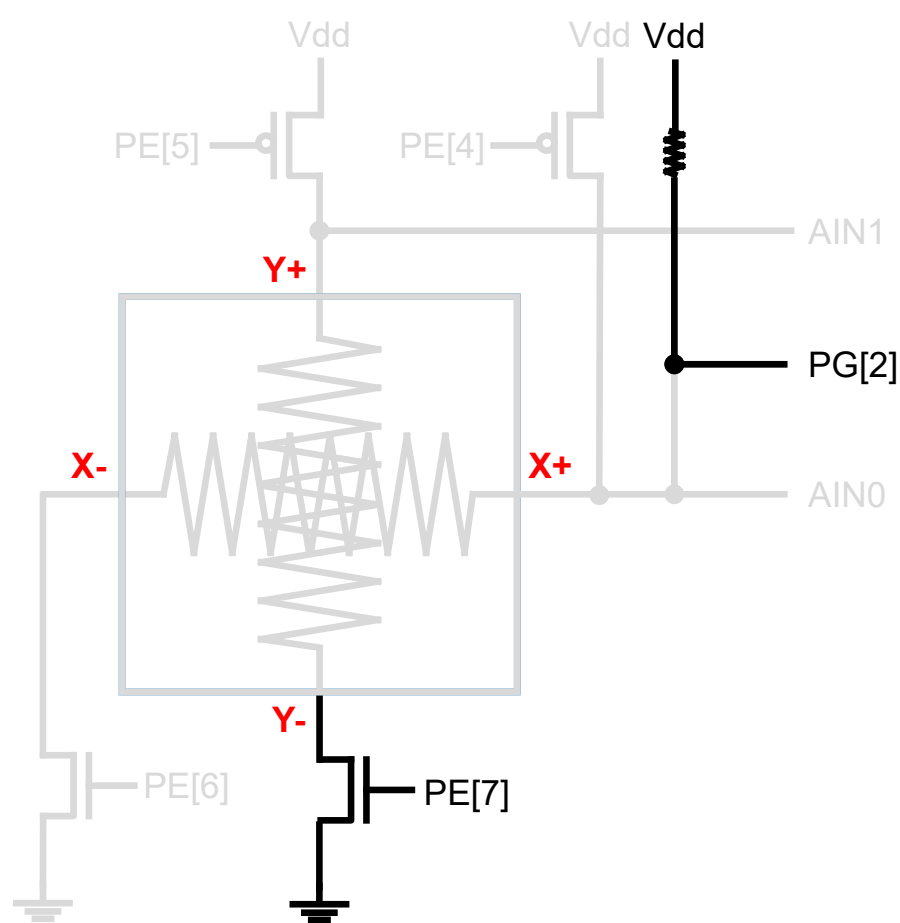


Touchscreen

lectura de voltajes



- El proceso de scan de la touchscreen supone:
 - Dar tensión por separado a cada una de las capas
 - Hacer lecturas del voltaje en los terminales usando un conversor A/D



Y- a GND, resto abiertos

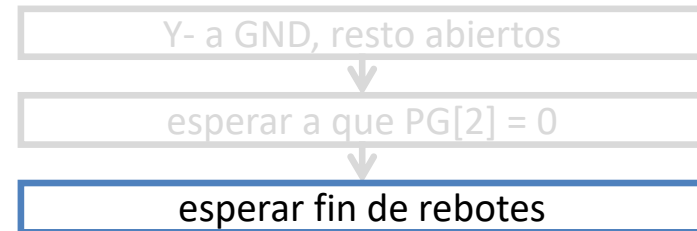
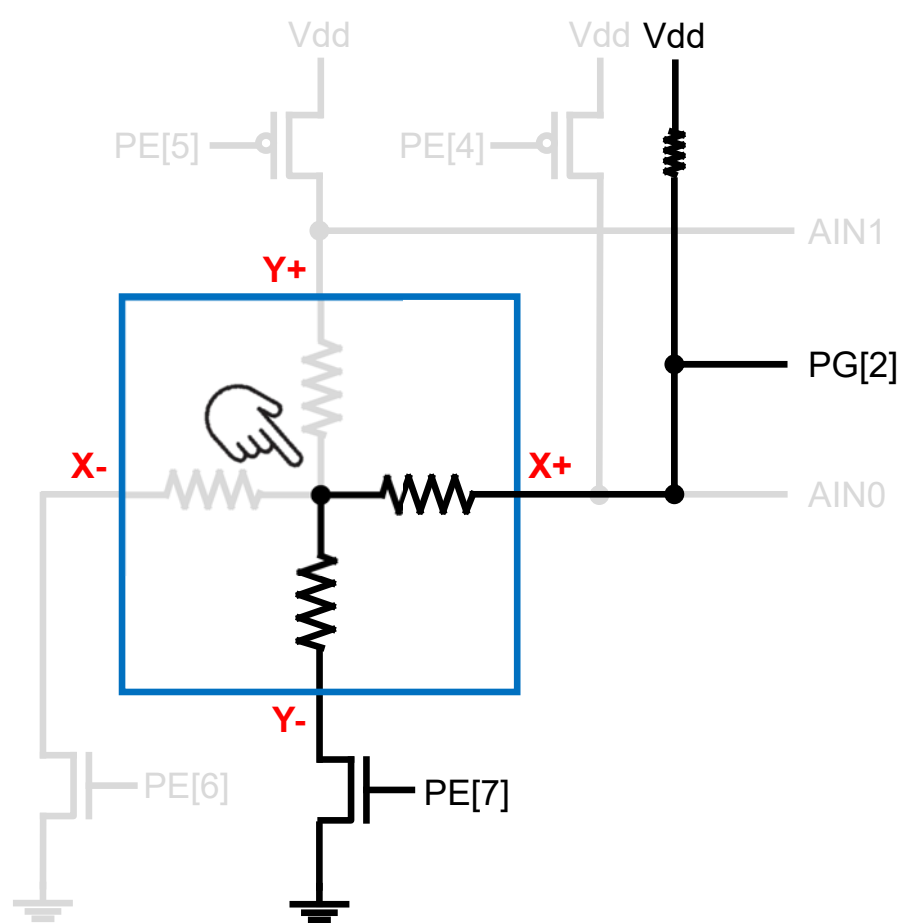
esperar a que PG[2] = 0

Touchscreen

lectura de voltajes



- El proceso de scan de la touchscreen supone:
 - Dar tensión por separado a cada una de las capas
 - Hacer lecturas del voltaje en los terminales usando un conversor A/D

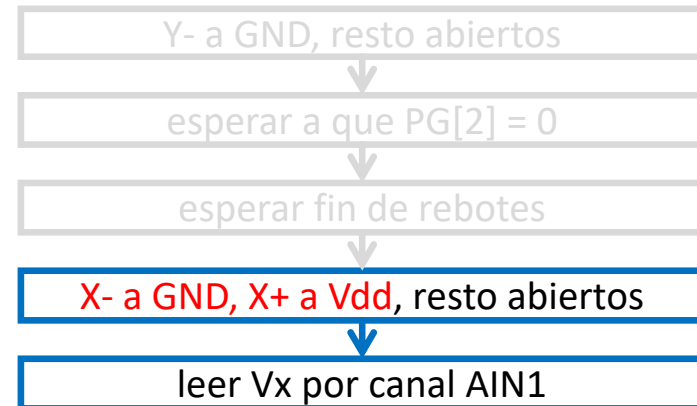
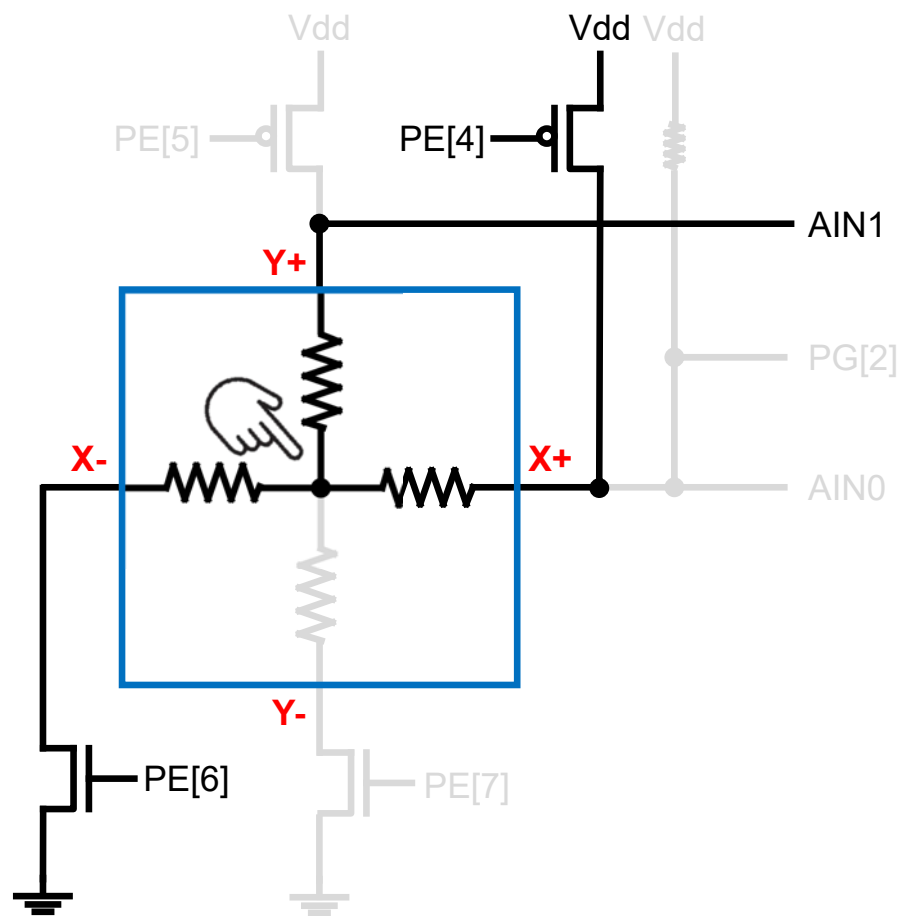


Touchscreen

lectura de voltajes



- El proceso de scan de la touchscreen supone:
 - o Dar tensión por separado a cada una de las capas
 - o Hacer lecturas del voltaje en los terminales usando un conversor A/D

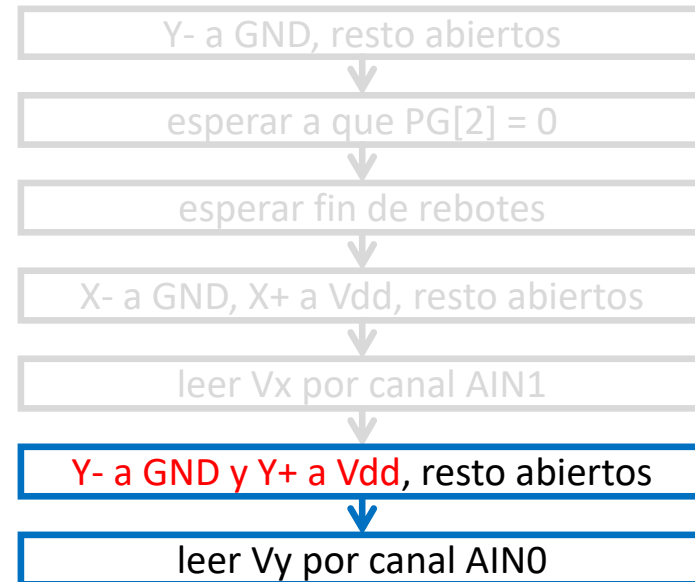
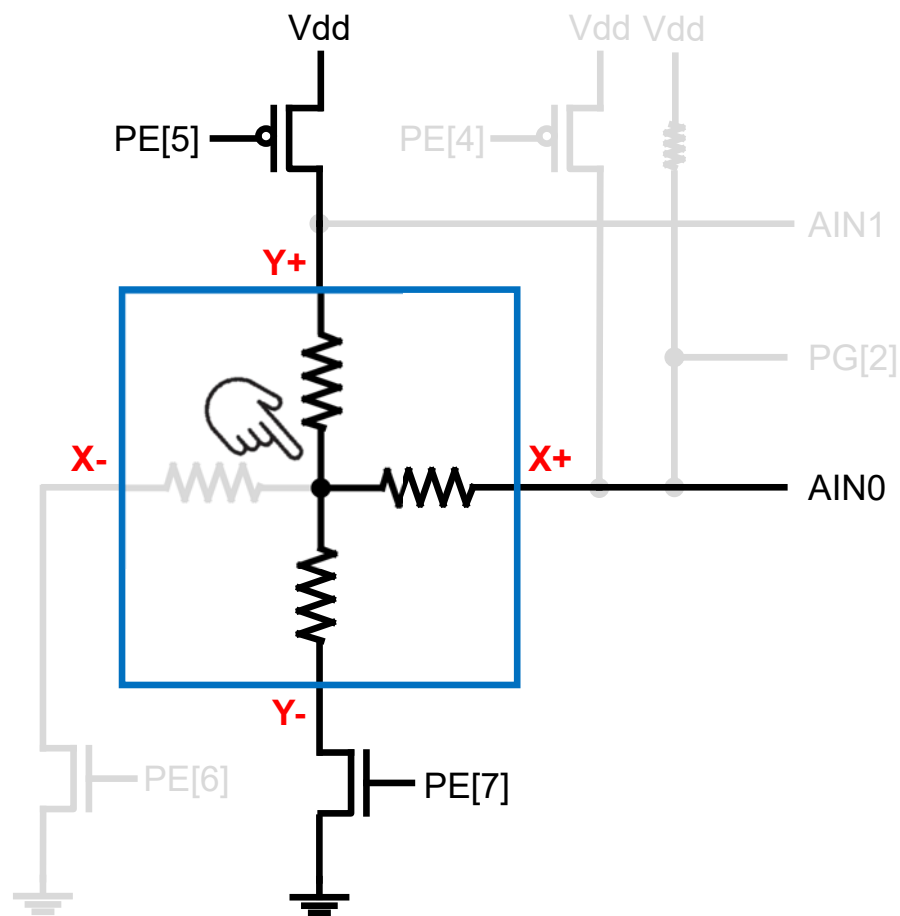


Touchscreen

lectura de voltajes



- El proceso de scan de la touchscreen supone:
 - o Dar tensión por separado a cada una de las capas
 - o Hacer lecturas del voltaje en los terminales usando un conversor A/D

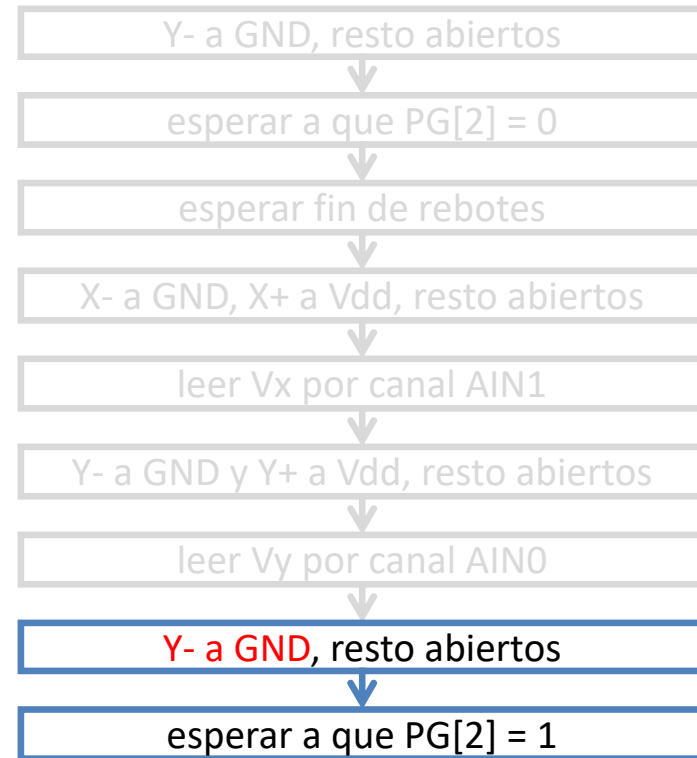
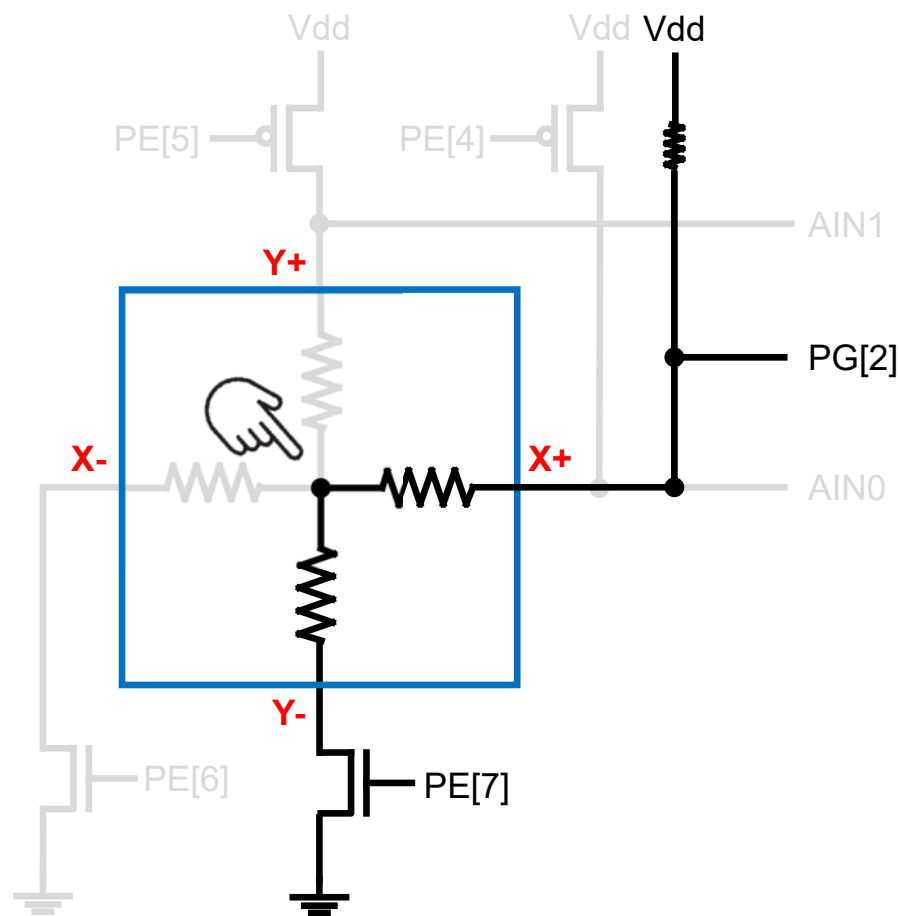


Touchscreen

lectura de voltajes



- El proceso de scan de la touchscreen supone:
 - Dar tensión por separado a cada una de las capas
 - Hacer lecturas del voltaje en los terminales usando un conversor A/D

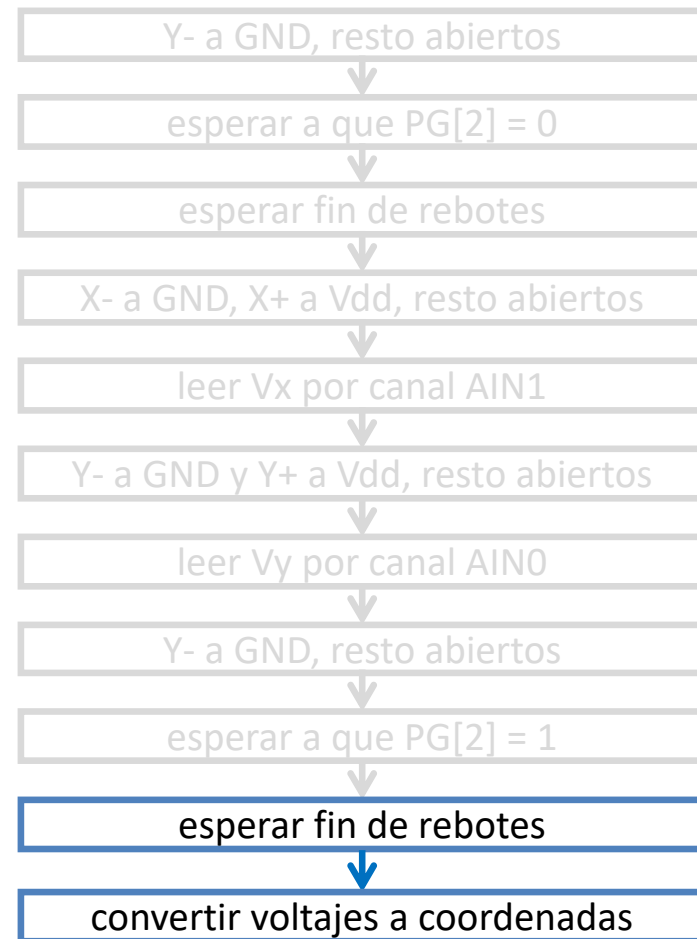
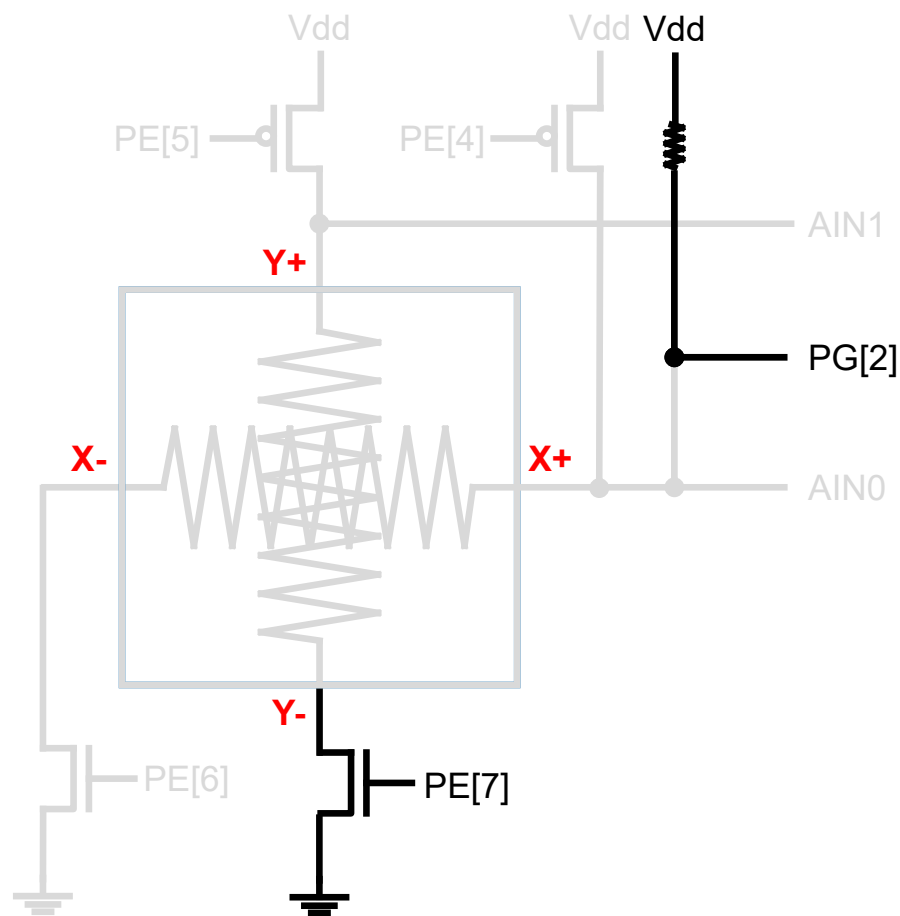


Touchscreen

lectura de voltajes



- El proceso de scan de la touchscreen supone:
 - Dar tensión por separado a cada una de las capas
 - Hacer lecturas del voltaje en los terminales usando un conversor A/D

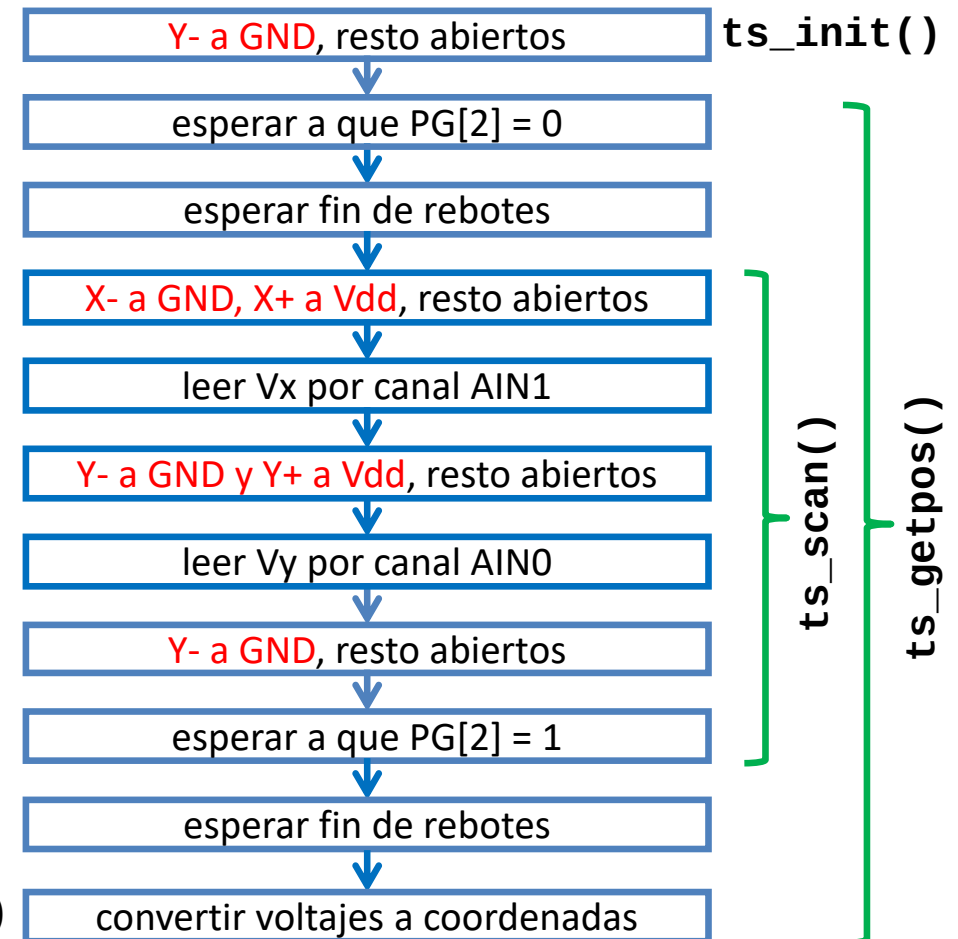
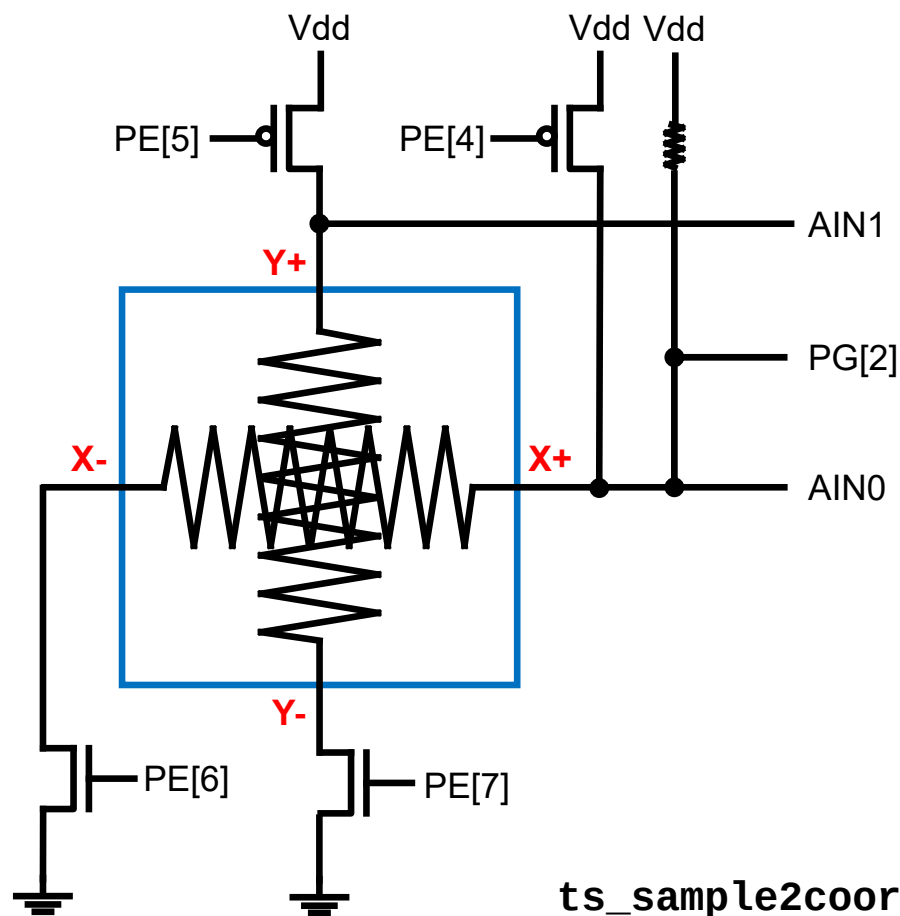


Touchscreen

lectura de voltajes



- El proceso de scan de la touchscreen supone:
 - o Dar tensión por separado a cada una de las capas
 - o Hacer lecturas del voltaje en los terminales usando un conversor A/D

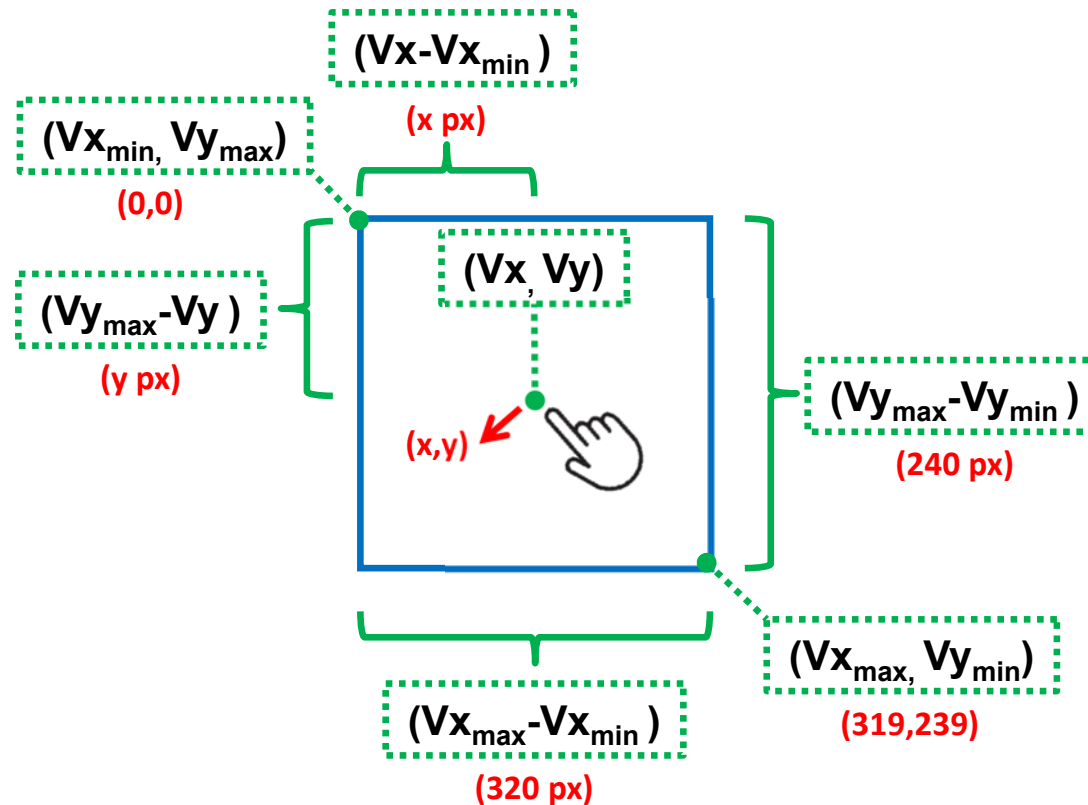




Touchscreen

conversión de voltajes a coordenadas

- El cálculo de las coordenadas a partir de voltajes leídos por el ADC:
 - Se realiza mediante una transformación lineal (regla de 3) que usa los factores de escala por calibrado.
 - En el caso de la placa S3CEV40, además, debe tenerse en cuenta que:
 - En la capa X, los voltajes crecen de izquierda a derecha, igual que las coordenadas del LCD
 - En la capa Y, los voltajes crecen de abajo a arriba, al revés que las coordenadas del LCD



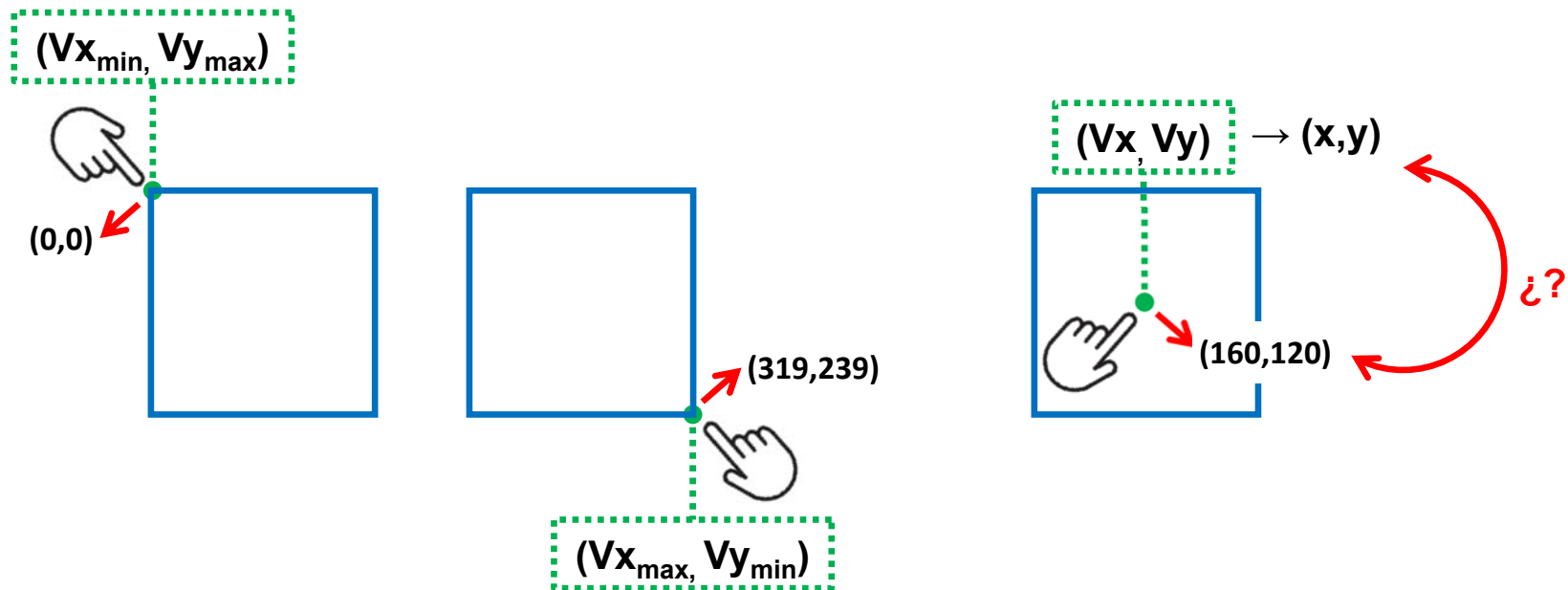
$$\begin{aligned} (V_{x_{max}} - V_{x_{min}}) &\rightarrow 320 \\ (V_x - V_{x_{min}}) &\rightarrow x \end{aligned}$$

$$\begin{aligned} (V_{y_{max}} - V_{y_{min}}) &\rightarrow 240 \\ (V_{y_{max}} - V_y) &\rightarrow y \end{aligned}$$

Touchscreen calibración



- La **touchscreen debe calibrarse** para:
 - Poder correlacionar los voltajes leídos con la resolución de la pantalla.
 - Compensar el desalineamiento entre coordenadas de visualización y de presión.
- El calibrado **se realiza una única vez** durante la inicialización del driver.
 - Leyendo los voltajes máximos/mínimos presionando en los extremos del touchpad.
 - Comparando la coordenada calculada tras presionar un punto de coordenada conocida.
 - El proceso se repite hasta que las coordenadas coincidan con un cierto margen de error.



Conversor analógico digital

configuración y operación (i)



- Tasa de conversión: máxima (100 KSPS, tiempo de conversión 10 μ s)
 - ADCPSR = 19 $100\text{KHz} = 64 \text{ MHz} / 2(n+1) \times 16 \Rightarrow n = 19$

- Para deshabilitar el conversor:
 - ADCCON[5] = 1 Sleep mode

- Para habilitar el conversor:
 - ADCCON[5] = 0 Normal mode
 - Esperar 10 ms antes de arrancar la conversión

- Para seleccionar un canal:
 - ADCCON[4:2] = X donde 0=AIN0, 1=AIN1...
 - Esperar 15 μ s antes de arrancar la conversión

Conversor analógico digital

configuración y operación (ii)



- Para realizar una conversión A/D individual por pooling:
 - o $ADCCON[1] = 0$ deshabilita arranque por lectura
 - o $ADCCON[0] = 1$ arranca manualmente la conversión
 - o Esperar hasta que la conversión comience (esperar mientras $ADCCON[0] == 1$)
 - o Esperar hasta que la conversión finalice (esperar mientras $ADCCON[6] == 0$)
 - o Leer ADCDAT
- Las lecturas de un conversor A/D son muy sensibles al ruido eléctrico
 - o Cualquier pequeño rebote en el voltaje dará una lectura errónea
- Para evitarlo, suele implementarse por SW algún tipo de filtrado
 - o Filtro de media: el valor de la muestra es la media de 5 conversiones consecutivas

Driver del conversor A/D

adc.h / adc.c



```
#ifndef __ADC_H__
#define __ADC_H__

#include <common_types.h>

#define ADC_AIN0 (0)
#define ADC_AIN1 (1)
...
void adc_init( void );
void adc_on( void );
void adc_off( void );
uint8 adc_status( void );
uint16 adc_getSample( uint8 ch );
void adc_open( void (*isr)(void) );
void adc_close( void );

#endif
```

} Declara macros para identificar los canales de entrada analógica

```
void adc_init( void )
{
    ADCPSR = ...;
    adc_off();
}
```

Driver del conversor A/D

adc.c



```

void adc_on( void )
{
    ADCCON &= ...; ..... Habilita el conversor
    sw_delay_ms( 10 ); ..... Espera a que esté activo
    state = ON;
}

uint16 adc_getSample( uint8 ch )
{
    uint32 sample;
    uint8 i;

    ADCCON = ...; ..... Configura el conversor para lectura individual del canal indicado
    sw_delay_ms( 10 ); ..... Espera a que la configuración tenga efecto
    for( i=0, sample=0; i<5; i++ )
    {
        ADCCON |= ...; ..... Arranca la conversión
        while( ... ); ..... Espera que la conversión comience
        while( ... ); ..... Espera que la conversión finalice
        sample += ADCDAT & 0x3ff; ..... Acumula la muestra (10b)
    }
    return sample / 5; ..... Realiza la media de las 5 lecturas consecutivas
}
    
```

Driver de touchscreen

ts.h



```
#ifndef __TS_H__
#define __TS_H__
```

```
#include <common_types.h>
```

```
#define TS_OK (1)
```

```
#define TS_FAILURE (0xff)
```

```
#define TS_TIMEOUT (0xfe)
```

```
#define TS_OFF (1)
```

```
#define TS_ON (0)
```

Declara macros para identificar errores durante la lectura de la touchscreen

Declara macros para identificar el estado de la touchscreen

```
void ts_init( void );
```

```
void ts_on( void );
```

```
void ts_off( void );
```

```
uint8 ts_status( void );
```

```
void ts_pressed( void );
```

```
void ts_getpos( uint16 *x, uint16 *y );
```

```
void ts_getpostime( uint16 *x, uint16 *y, uint16 *ms );
```

```
uint8 ts_timeout_getpos( uint16 *x, uint16 *y, uint16 n );
```

```
void ts_open( void (*isr)(void) );
```

```
void ts_close( void );
```

```
#endif
```

Driver de touchscreen

ts.c



...

#define PX_ERROR (5)

Margen máximo de error entre las coordenadas de visualización y de presión para considerar que la touchscreen está correctamente calibrada

static uint16 Vxmin = 0;

static uint16 Vxmax = 0;

static uint16 Vymin = 0;

static uint16 Vymax = 0;

Valores máximos y mínimos legibles del conversor y correspondientes a las coordenadas de las esquinas del LCD (toman valores tras calibrar la touchscreen)

static uint8 state;

extern void isr_TS_dummy(void);

static void ts_scan(uint16 *x, uint16 *y);

static void ts_calibrate(void);

static void ts_sample2coord(uint16 Vx, uint16 Vy, uint16 *x, uint16 *y);

void ts_init(void)

{

timers_init();

lcd_init();

adc_init();

PDATE = ...; *Conecta Y- con GND dejando el resto de terminales abiertos*

sw_delay_ms(1); *Espera a la estabilización de los voltajes en el touchscreen*

ts_on();

ts_calibrate();

ts_off();

}

Driver de touchscreen

ts.c



```
static void ts_calibrate( void )
```

```
{
```

```
    uint16 x, y;
```

```
    ...
```

```
    do {
```

```
        ...
```

Borra pantalla y muestra algún mensaje introductorio de calibración sobre el LCD
Pinta un cuadrado de 5x5 px en la esquina superior izquierda (0,0) y solicita que se presione

```
        while( ... ); ..... Espera presión de la touchscreen
```

```
        sw_delay_ms( TS_DOWN_DELAY ); ..... Espera fin de rebotes
```

```
        ts_scan( &Vxmin, &Vymax ); ..... Lee Vxmin y Vymax correspondientes a la coordenada (0,0)
```

```
        while( ... ); ..... Espera depresión de la touchscreen
```

```
        sw_delay_ms( TS_UP_DELAY ); ..... Espera fin de rebotes
```

```
        ...
```

Pinta un cuadrado de 5x5 px en la esquina inferior derecha (319, 239) y solicita que se presione

```
        while( ... ); ..... Espera presión de la touchscreen
```

```
        sw_delay_ms( TS_DOWN_DELAY ); ..... Espera fin de rebotes
```

```
        ts_scan( &Vxmax, &Vymax ); ..... Lee Vxmax y Vymax correspondientes a la coordenada 319, 239)
```

```
        while( ... ); ..... Espera depresión de la touchscreen
```

```
        sw_delay_ms( TS_UP_DELAY ); ..... Espera fin de rebotes
```

```
        ...
```

Pinta un cuadrado de 5x5 px centrado en el punto central (160, 120) y solicita que se presione

```
        ts_getpos( &x, &y ); ..... Obtiene las coordenadas del punto central
```

```
    } while(
```

```
        (x > 160+PX_ERROR) || (x < 160-PX_ERROR)
```

```
        || (y > 120+PX_ERROR) || (y < 120-PX_ERROR)
```

```
    );
```

Repite el proceso de calibración mientras no se obtenga el nivel de precisión esperado

```
    lcd_clear();
```

```
}
```

Driver de touchscreen

ts.c



```
static void ts_scan( uint16 *Vx, uint16 *Vy )
{
    PDATE = ...; ..... Conecta X- con GND, X+ con Vdd y deja el resto de terminales abiertos
    *Vx = adc_getSample( ... ); ..... Lee Vx por el canal AIN1

    PDATE = ...; ..... Conecta Y- con GND, Y+ con Vdd y deja el resto de terminales abiertos
    *Vy = adc_getSample( ... ); ..... Lee Vy por el canal AIN0

    PDATE = ...; ..... Conecta Y- con GND dejando el resto de terminales abiertos
    sw_delay_ms( 1 ); ..... Espera a la estabilización de los voltajes en el touchscreen
}

static void ts_sample2coord( uint16 Vx, uint16 Vy, uint16 *x, uint16 *y )
{
    if( Vx < Vxmin )
        *x = 0; ..... Satura borde izquierdo
    else if( Vx > Vxmax )
        *x = 319; ..... Satura borde derecho
    else
        *x = 320*(Vx-Vxmin) / (Vxmax-Vxmin); ..... Regla de 3
    .....
}
```

Calcula x según Vx
(x crece si Vx crece)

Calcula y según Vy
(tener en cuenta que y crece si Vy decrece)



Acerca de *Creative Commons*



■ Licencia CC (*Creative Commons*)

o Ofrece algunos derechos a terceras personas bajo ciertas condiciones. Este documento tiene establecidas las siguientes:



Reconocimiento (*Attribution*):

En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.



No comercial (*Non commercial*):

La explotación de la obra queda limitada a usos no comerciales.



Compartir igual (*Share alike*):

La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Más información: <https://creativecommons.org/licenses/by-nc-sa/4.0/>