OKY3420-6: MODULO SENSOR TACTIL CAPACITIVO MPR121.

Descripción:

El modulo tactil MPR121 es un controlador de sensor táctil capacitivo controlado por una interfaz I2C. El chip puede controlar hasta doce electrodos individuales, así como un decimotercero simulado. El MPR121 teclado touch también cuenta con ocho pines de conducción LED, esto quiere decir que cuando estos pines no están configurados como electrodos, se pueden usar para controlar los LED.

Características:

• Voltaje de operación: 2.5V a 3.6 V
• Interfaz: I2C
• Dirección I2C por defecto: 0x5A
• Dimensiones: 30mm x 20mm
• Cantidad de electrodos táctiles: 12 y 1 simulado
• Cantidad de LEDs que puede controlar: 8

El módulo MPR121 puede controlar 12 electrodos, cada electrodo se comporta como un sensor táctil capacitivo, la comunicación es con interfaz I2C, por lo que solo utiliza dos pines y es fácil de implementar con cualquier microcontrolador. En el reverso del módulo MPR121 tiene 4 puentes, los cuales hay que romper si no se necesita usar las resistencias pullup que tiene el modulo. También se puede romper el puente de ADD para cambiar la dirección en caso se necesite conectar otro MPR121 al bus I2C. El electrodo puede ser cualquier superficie conductora, como metal, papel metálico, cobre; para activar el electrodo no necesariamente tiene que haber contacto, la sensibilidad depende de cómo se calibre, esto se lo hace con umbrales el cual el módulo utiliza como referencia para que considere Touch o no. Para cada electrodo se le asigna un umbral de activación. En caso no se utilicen todos los electrodos, también se los puede configurar como salidas para activar LEDs

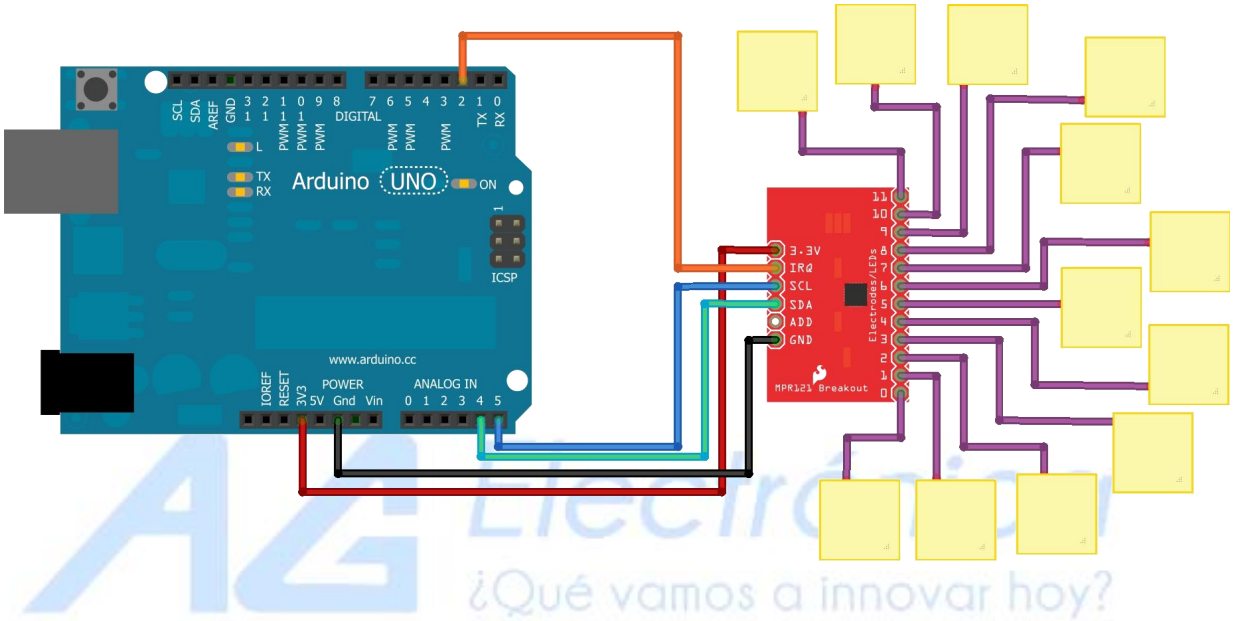
Conexión:

La alimentación es de 3.3V, la comunicación es a través del I2C del Arduino cuyos pines varía según la placa que se utilice, el pin IRQ generalmente va un pin de interrupción externa que trae el Arduino, a continuación se muestra la conexión del MPR121

MPR 121	Arduino uno, Nano, Mini Pro	Arduino mega 2560	Arduino Due
3.3V	PIN 3.3V (*)	PIN 3.3V (*)	PIN 3.3V (*)
IRQ	PIN 2 (**)	PIN 2 (**)	PIN 2 (**)
SCL	PIN A5	PIN 21	PIN 21
SDA	PIN A4	PIN 20	PIN 20
GND	PIN GND	PIN GND	PIN GND

*EL módulo PMR121 no tiene regulador de voltaje interno, por esto se utiliza la salida de voltaje 3.3V de Arduino, pero se puede usar una fuente externa siempre y cuando no exceda el voltaje de 3.3V.

**Para el pin IRQ se puede usar cualquier pin digital, en este caso no se usa interrupción, pero de ser el caso conectar al pin de interrupción externa correspondiente.



Código muesta:

Trabajaremos la programación sin usar librerías, de esta forma podrás migrar el código a cualquier otra plataforma o micro controlador, como los PICs de microchip

Empezamos declarando la librería para la comunicación I2C y el pin IRQ.

```
#include <Wire.h>
int  Pin_IRQ = 2;  // Pin IRQ en el pin digital 2
boolean EstadoAnterior[12]; //Para Alamacenar el ultimo estado del valor del touch
int address_mpr=0x5A; //dirección de nuestro MPR121
```

Notar que también creamos una variable para guardar los estados anteriores de los electrodos, para más adelante ver si han cambiado de estado. Además se declara una variable para especificar la dirección i2C de nuestro modulo PMR121

El siguiente paso es en configurar todos nuestros periféricos: los pines, la comunicación serial, el bus i2c y configurar el módulo MPR121.}

```
void setup(){
  pinMode(Pin_IRQ, INPUT);
  digitalWrite(Pin_IRQ, HIGH); //Habilitamos la resistencia pullup
  Serial.begin(9600);
  Wire.begin();
  mpr121_Config(address_mpr);
}
```

Como se observa para configurar el modulo usamos la funcion mpr121_Config(address_mpr), dicha función se muestra a continuación:

```
void mpr121_Config(int address){

  set_register(address,0x5E,0x00); //ELE_CFG

  // Section A - Controls filtering when data is > baseline.
  set_register(address, 0x2B, 0x01); //MHD_R
  set_register(address, 0x2C, 0x01); //NHD_R
  set_register(address, 0x2D, 0x00); //NCL_R
  set_register(address, 0x2E, 0x00); //FDL_R

  // Section B - Controls filtering when data is < baseline.
  set_register(address, 0x2F, 0x01); //MHD_F
  set_register(address, 0x30, 0x01); //NHD_F
  set_register(address, 0x31, 0xFF); //NCL_F
  set_register(address, 0x32, 0x02); //FDL_F

  // Section C - Sets touch and release thresholds for each electrode
  int umbral_Touch=0x44;
  int umbral_Release=0x40;
  set_register(address, 0x41, umbral_Touch); //Umbral Touch 0
  set_register(address, 0x42, umbral_Release); //Umbral Release 0

  set_register(address, 0x43, umbral_Touch); //Umbral Touch 1
  set_register(address, 0x44, umbral_Release); //Umbral Release 1

  set_register(0x5A, 0x45, umbral_Touch); //Umbral Touch 2
  set_register(0x5A, 0x46, umbral_Release); //Umbral Release 2

  set_register(address, 0x47, umbral_Touch); //Umbral Touch 3
  set_register(address, 0x48, umbral_Release); //Umbral Release 3

  set_register(address, 0x49, umbral_Touch); //Umbral Touch 4
  set_register(address, 0x4A, umbral_Release); //Umbral Release 4

  set_register(address, 0x4B, umbral_Touch); //Umbral Touch 5
  set_register(address, 0x4C, umbral_Release); //Umbral Release 5

  set_register(address, 0x4D, umbral_Touch); //Umbral Touch 6
  set_register(address, 0x4E, umbral_Release); //Umbral Release 6

  set_register(address, 0x4F, umbral_Touch); //Umbral Touch 7
  set_register(address, 0x50, umbral_Release); //Umbral Release 7

  set_register(address, 0x51, umbral_Touch); //Umbral Touch 8
  set_register(address, 0x52, umbral_Release); //Umbral Release 8

  set_register(address, 0x53, umbral_Touch); //Umbral Touch 9
  set_register(address, 0x54, umbral_Release); //Umbral Release 9

  set_register(address, 0x55, umbral_Touch); //Umbral Touch 10
  set_register(address, 0x56, umbral_Release); //Umbral Release 10

  set_register(address, 0x57, umbral_Touch); //Umbral Touch 11
  set_register(address, 0x58, umbral_Release); //Umbral Release 11

  // Section D
  // Set the Filter Configuration
  // Set ESI2
  set_register(address, 0x5D, 0x04); //FIL_CFG

  // Section E
  // Electrode Configuration
  // Set ELE_CFG to 0x00 to return to standby mode
  set_register(address, 0x5E, 0x0C);  // ELE_CFG  Enables all 12 Electrodes

}
```

Básicamente en la función anterior se configura los registros necesarios para su funcionamiento, doce de los registros se utilizan para establecer los umbrales de “tocar” y otros doce para los umbrales de “soltar”, estos valores de registros hay q aumentar o disminuir de acuerdo al tipo de electrodo que usemos, nosotros hemos creado dos variables para establecer los umbrales de Touch y reléase (presionar y soltar) de los 12 electrodos por igual, pero puedes calibrarlo de forma independiente si usas electrodos diferentes

```
// Section C - Sets touch and release thresholds for each electrode
int umbral_Touch=0x44;
int umbral_Release=0x40;

set_register(address, 0x41, umbral_Touch); //Umbral Touch 0
set_register(address, 0x42, umbral_Release); //Umbral Release 0

set_register(address, 0x43, umbral_Touch); //Umbral Touch 1
set_register(address, 0x44, umbral_Release); //Umbral Release 1
```

Ambos umbrales generalmente son valores cercanos, pero se puede trabajar con un ciclo de histéresis si se pone el umbral de Release por debajo del de Touch.

Una vez configurado el módulo MPR121, el siguiente paso es leer si se ha presionado o tocado algún electrodo, para esto hacemos uso del pin IRQ, que se pone a 0 cada vez que hay una cambio en alguno de los electrodos.

Después de detectar el cambio a través del pin IRQ, leemos el estado de los electrodos, de la siguiente forma:

```
if(!digitalRead(Pin_IRQ)){

  // Leemos los estados de los electrodos del MPR121
  Wire.requestFrom(address_mpr,2);

  byte LSB = Wire.read();
  byte MSB = Wire.read();

  uint16_t touched = ((MSB << 8) | LSB); // 12 de los 16 bits corresponden a los estados táctiles
```

Como se observa el valor de los electrodos se almacena en una variable de 16bits, de los cuales los 12 menos significativos almacenan los valores del estado de los electrodos.

El siguiente paso es analizar el estado de cada uno de ellos de forma individual, el estado actual se compara con el estado anterior y si existe alguna cambio se realiza la acción correspondiente, que en nuestro caso simplemente es notificar por el puerto serie.

```
  for (int i=0; i < 12; i++)
  {
    boolean EstadoActual=bitRead(touched, i);
    if(!(EstadoAnterior[i])&&EstadoActual)
    {
        Serial.print("Electrodo ");
        Serial.print(i);
        Serial.println(" se acaba de tocar");
    }

    if(EstadoAnterior[i]&&(!EstadoActual))
    {
        Serial.print("Electrodo ");
        Serial.print(i);
        Serial.println(" ya no está siendo tocado");
    }
    EstadoAnterior[i]=EstadoActual;
  }
```

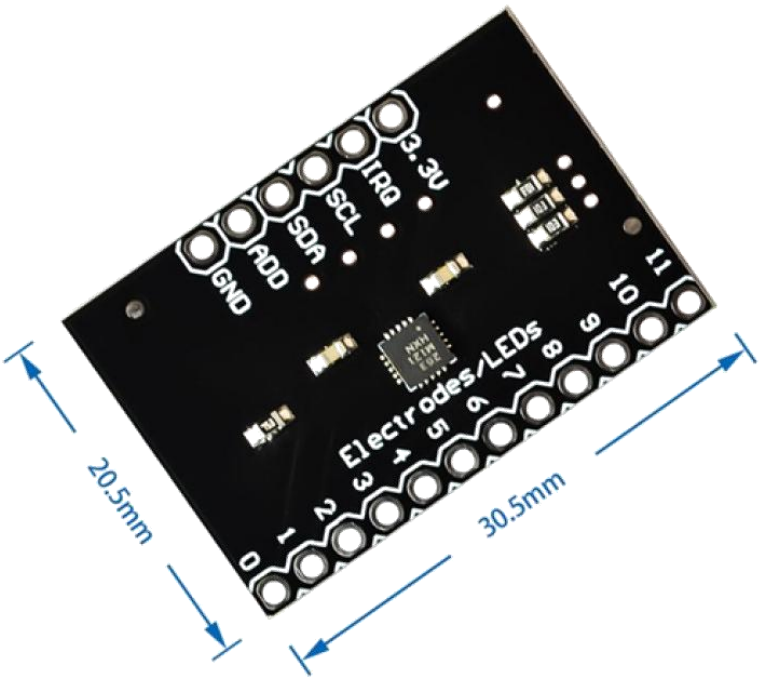
A continuación se muestra el código completo para leer los 12 electrodos.

```
#include <Wire.h>
int  Pin_IRQ = 2;  // Pin IRQ en el pin digital 2
```

```
boolean EstadoAnterior[12]; //Para Alamacenar el ultimo estado del valor del touch
int address_mpr=0x5A; //dirección de nuestro MPR121

void setup(){
  pinMode(Pin_IRQ, INPUT);
  digitalWrite(Pin_IRQ, HIGH); //Habilitamos la resistencia pullup
  Serial.begin(9600);
  Wire.begin();
  mpr121_Config(address_mpr);
}

void loop(){


    if(!digitalRead(Pin_IRQ)){

      // Leemos los estados de los electrodos del MPR121
      Wire.requestFrom(address_mpr,2);

      byte LSB = Wire.read();
      byte MSB = Wire.read();

      uint16_t touched = ((MSB << 8) | LSB); // 12 de los 16 bits corresponden a los estados
táctiles

      //Comprobamos si los electrodos se han presionado
      for (int i=0; i < 12; i++)
      {
        boolean EstadoActual=bitRead(touched, i);
        if(!(EstadoAnterior[i])&&EstadoActual)
        {
            Serial.print("Electrodo ");
            Serial.print(i);
            Serial.println(" se acaba de tocar");
        }

        if(EstadoAnterior[i]&&(!EstadoActual))
        {
            Serial.print("Electrodo ");
            Serial.print(i);
            Serial.println(" ya no está siendo tocado");
        }
        EstadoAnterior[i]=EstadoActual;
      }
    }
}


void mpr121_Config(int address){

  set_register(address,0x5E,0x00); //ELE_CFG

  // Section A - Controls filtering when data is > baseline.
  set_register(address, 0x2B, 0x01); //MHD_R
  set_register(address, 0x2C, 0x01); //NHD_R
  set_register(address, 0x2D, 0x00); //NCL_R
  set_register(address, 0x2E, 0x00); //FDL_R

  // Section B - Controls filtering when data is < baseline.
  set_register(address, 0x2F, 0x01); //MHD_F
  set_register(address, 0x30, 0x01); //NHD_F
  set_register(address, 0x31, 0xFF); //NCL_F
  set_register(address, 0x32, 0x02); //FDL_F

  // Section C - Sets touch and release thresholds for each electrode
  int umbral_Touch=0x44;
  int umbral_Release=0x40;
```

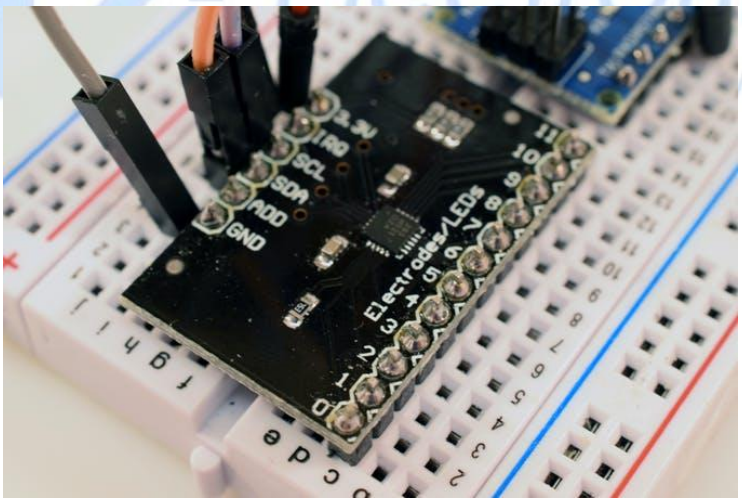
```
  set_register(address, 0x41, umbral_Touch); //Umbral Touch 0
  set_register(address, 0x42, umbral_Release); //Umbral Release 0

  set_register(address, 0x43, umbral_Touch); //Umbral Touch 1
  set_register(address, 0x44, umbral_Release); //Umbral Release 1

  set_register(0x5A, 0x45, umbral_Touch); //Umbral Touch 2
  set_register(0x5A, 0x46, umbral_Release); //Umbral Release 2

  set_register(address, 0x47, umbral_Touch); //Umbral Touch 3
  set_register(address, 0x48, umbral_Release); //Umbral Release 3

  set_register(address, 0x49, umbral_Touch); //Umbral Touch 4
  set_register(address, 0x4A, umbral_Release); //Umbral Release 4

  set_register(address, 0x4B, umbral_Touch); //Umbral Touch 5
  set_register(address, 0x4C, umbral_Release); //Umbral Release 5

  set_register(address, 0x4D, umbral_Touch); //Umbral Touch 6
  set_register(address, 0x4E, umbral_Release); //Umbral Release 6

  set_register(address, 0x4F, umbral_Touch); //Umbral Touch 7
  set_register(address, 0x50, umbral_Release); //Umbral Release 7

  set_register(address, 0x51, umbral_Touch); //Umbral Touch 8
  set_register(address, 0x52, umbral_Release); //Umbral Release 8

  set_register(address, 0x53, umbral_Touch); //Umbral Touch 9
  set_register(address, 0x54, umbral_Release); //Umbral Release 9

  set_register(address, 0x55, umbral_Touch); //Umbral Touch 10
  set_register(address, 0x56, umbral_Release); //Umbral Release 10

  set_register(address, 0x57, umbral_Touch); //Umbral Touch 11
  set_register(address, 0x58, umbral_Release); //Umbral Release 11

  // Section D
  // Set the Filter Configuration
  // Set ESI2
  set_register(address, 0x5D, 0x04); //FIL_CFG

  // Section E
  // Electrode Configuration
  // Set ELE_CFG to 0x00 to return to standby mode
  set_register(address, 0x5E, 0x0C);  // ELE_CFG  Enables all 12 Electrodes

}

void set_register(int address, unsigned char r, unsigned char v){
    Wire.beginTransmission(address);
    Wire.write(r);
    Wire.write(v);
    Wire.endTransmission();
}
```

Dimensiones:



Aplicaciones:

- Detección capacitiva
- Reemplazo de interruptores
- Botones táctiles
- Teclado táctil
- Rueda táctil
- Touchpad
- Detector de proximidad



AG Electrónica ¿Qué vamos a innovar hoy?	AG Electrónica S.A.P.I. de C.V. República del Salvador N° 20 Segundo Piso Teléfono: (01)55 5130 – 7210		
ACOTACIÓN: N/A	http://www.agelectronica.com	ESCALA: N/A	REALIZO: GAC REV: GAC
TOLERANCIA: N/A	MODULO SENSOR TACTIL CAPACITIVO MPR121		
TOLERANCIA: N/A	Fecha: 08/10/2021	**No. Parte: OKY3420-6**	

AG Electrónica ¿Qué vamos a innovar hoy?