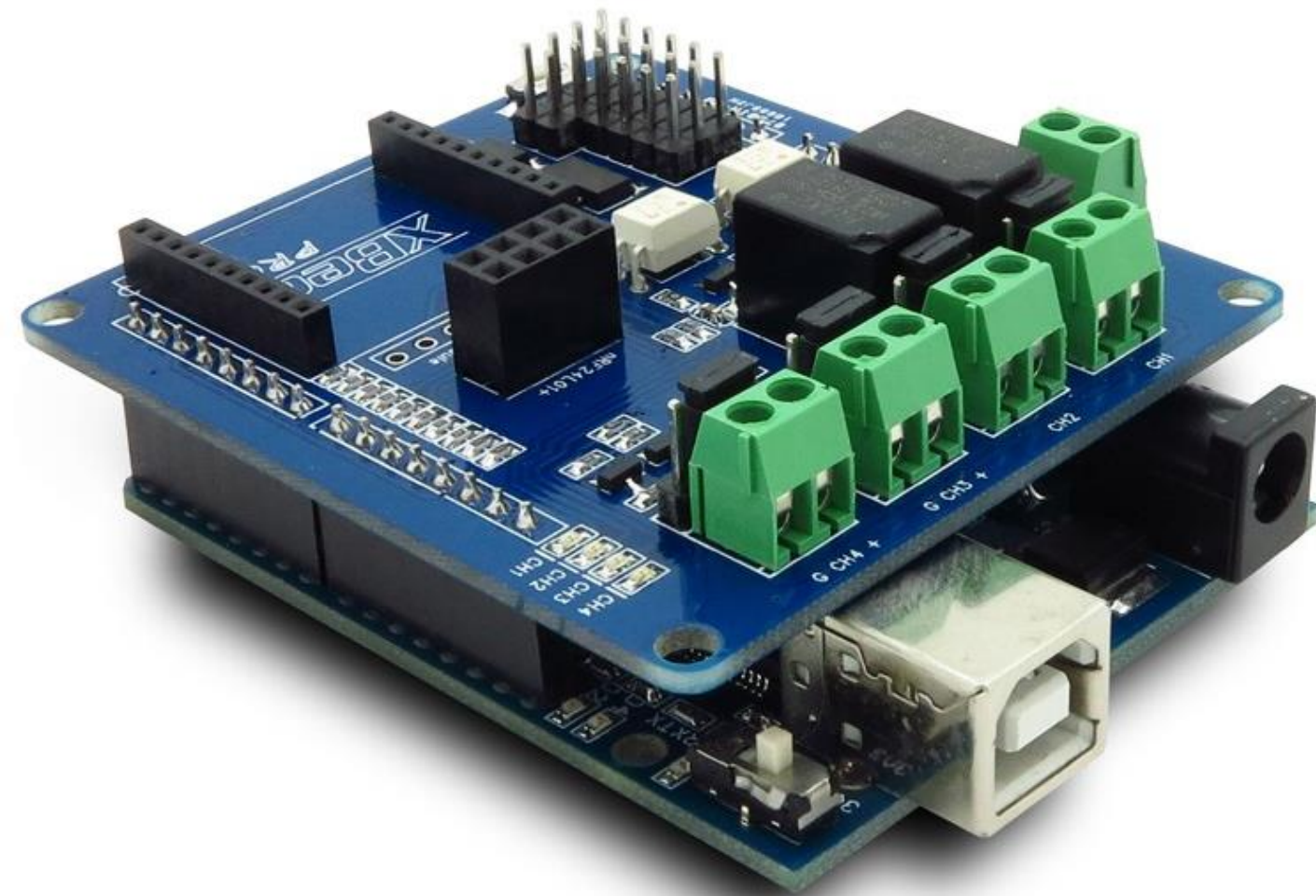


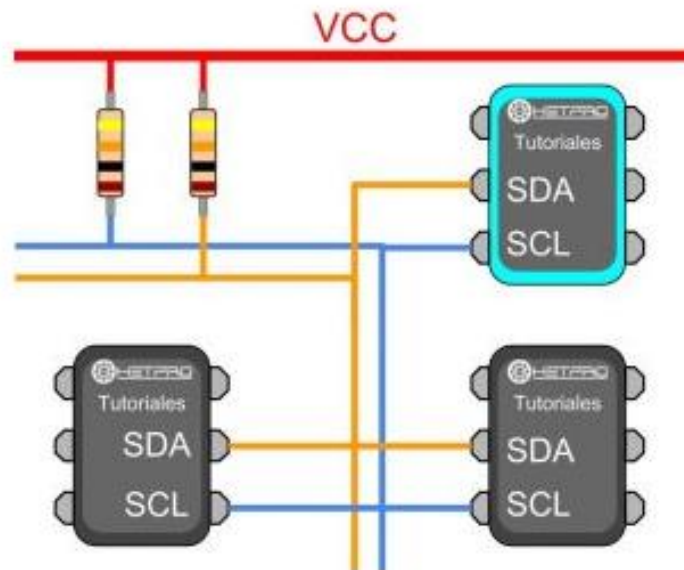


MÓDULOS DE COMUNICACIÓN I2C

SISTEMAS EMBEBIDOS

PROTOCOLO I2C

I2C es un puerto y protocolo de comunicación serial, define la trama de datos y las conexiones físicas para transferir bits entre 2 dispositivos digitales. El puerto incluye dos cables de comunicación, SDA y SCL. Además, el protocolo permite conectar hasta 127 dispositivos esclavos con esas dos líneas, con hasta velocidades de 100, 400 y 1000 kbits/s. En el caso de I2C se diferencian dos elementos básicos, un MAESTRO y un ESCLAVO



PROTOCOLO I2C

El **MAESTRO** I2C se encarga de controlar al cable de reloj, por sus siglas en inglés llamada SCL – Serial CLock. Además el MAESTRO se encarga de iniciar y parar la comunicación. La información binaria serial se envía sólo por la línea o cable de datos seriales, en inglés se llama SDA – Serial Data. Dos Maestros no pueden hacer uso de un mismo puerto I2C.

El **ESCLAVO** I2C, generalmente suele ser un sensor. Este elemento suministra de la información de interés al MAESTRO. Puede actuar de dos formas: esclavo-transmisor ó esclavo-receptor. Un dispositivo I2C esclavo, no puede generar a la señal SCL.

Trama de comunicación:

- Inicio ó Start – S
- Parada – P
- Confirmación – ACK
- No Confirmación – NACK
- Lectura-/Escritura – L/W
- 7 bits para la dirección del dispositivo esclavo/maestro
- 8 bits de dirección (para algunos sensores pueden ser 16 bits)
- 8 bits de datos

wire.begin() wire.available()

MASTER/SLAVE:

```
Wire.begin() // Inicializa el hardware del bus
Wire.beginTransmission(address); //Comienza la transmisión
Wire.endTransmission(); // Finaliza la transmisión
Wire.requestFrom(address,nBytes); //solicita un numero de bytes al esclavo en la dirección address
```

```
Wire.available(); // Detecta si hay datos pendientes por ser leídos
Wire.write(); // Envía un byte
Wire.read(); // Recibe un byte
Wire.setClock(clock); // Modifica la frecuencia de trabajo del puerto de comunicación. Valores son: 100000 (standard mode) and 400000 (fast mode).
```

```
Wire.onReceive(handler); // Registra una función de callback al recibir un dato
Wire.onRequest(handler); // Registra una función de callback al solicitar un dato
```

Código MASTER.ino

```
#include <Wire.h>
int cont=0;
void setup() {
  Wire.begin();
}

void loop() {
  cont++;
  Wire.beginTransaction(4);
  Wire.write(cont);
  Wire.endTransmission();
  delay(2000);
}
```

Librería de comunicación i2c.

Inicializa la comunicación i2c

Inicia la transmisión al dispositivo configurado como esclavo con dirección 4

Envía mensaje por puerto de comunicación

Termina la comunicación i2c

Código MASTER.ino

```
#include <Wire.h>
```

```
void setup()
```

```
{
```

```
  Wire.begin(4);
```

```
  Wire.onReceive(receiveEvent);
```

```
  Serial.begin(9600);
```

```
}
```

```
void loop()
```

```
{
```

```
  delay(100);
```

```
}
```

```
void receiveEvent(int bytes){
```

```
  while(Wire.available())
```

```
  {
```

```
    int c = Wire.read();
```

```
    Serial.print(c);
```

```
  }
```

```
}
```

Librería de comunicación i2c.

Evento de recepción de datos por comunicación i2c

Programación del evento de comunicación i2c

Condición si se encuentran datos

Lee los dato por comunicación i2c

PROGRAMACIÓN



CÓDIGO 12: ENVÍO Y
RECEPCIÓN DE MENSAJES I2C



CÓDIGO 13: MENSAJES
COMPLEJOS POR I2C