Apuntes Arduino II

Receptor IR

```
#include <IRremote.h>  // Librería para el receptor IR

const int RECV_PIN = 8; // Pin al que conectamos la salida del receptor IR
IRrecv irrecv(RECV_PIN); // Objeto de tipo IRrecv
decode_results results; // Datos que recibiremos

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Activamos el receptor IR
}

void loop()
{
  if(irrecv.decode(&results)) // Devuelve 0 si no hay datos disponibles, 1 en caso contrario
  {
    Serial.print("Codigo recibido: ");
    Serial.println(results.value, HEX); // Imprime el código recibido. Ver referencia:
                  // https://www.arduino.cc/reference/en/language/functions/communication/serial/println/
    irrecv.resume(); // Reinicia el estado del receptor IR y queda listo esperando el siguiente valor
    if(results.value == 0xFD00FF) { // Botón 'Power'
      Serial.println("Power");
    }
  }
}
```

- Con IRrecv irrecv(RECV_PIN) creamos la variable u objeto para el receptor IR en el pin especificado.
- Creamos la variable results, que es una estructura de datos en donde se guardaran todos los datos recibidos por el sensor.
- En Setup() inicializamos la recepción de datos con irrecv.enableIRIn()
- En loop() comprobamos si llega un dato al receptor mediante if(irrecv.decode(&results)). En este ejemplo, si se pulsa el botón "Power" lo muestra en el monitor serie.

Protocolo I2c

VENTAJAS Y DESVENTAJAS DEL I2C

Ventajas

- Requiere pocos cables
- Dispone de mecanismos para verificar que la señal hay llegado

Desventajas

- Baja velocidad
- No es full duplex
- No hay verificación de que el contenido del mensaje es correcto

I2C en Arduino

Arduino dispone de **soporte I2C por hardware** vinculado físicamente a ciertos pines. Estos pines varían de un modelo de Arduino a otro. En los que nos interesan particularmente:

- Arduino Uno: pines A4 (SDA) y A5 (SCL)
- Arduino Mega 2560: 20 (SDA) y 21 (SCL)

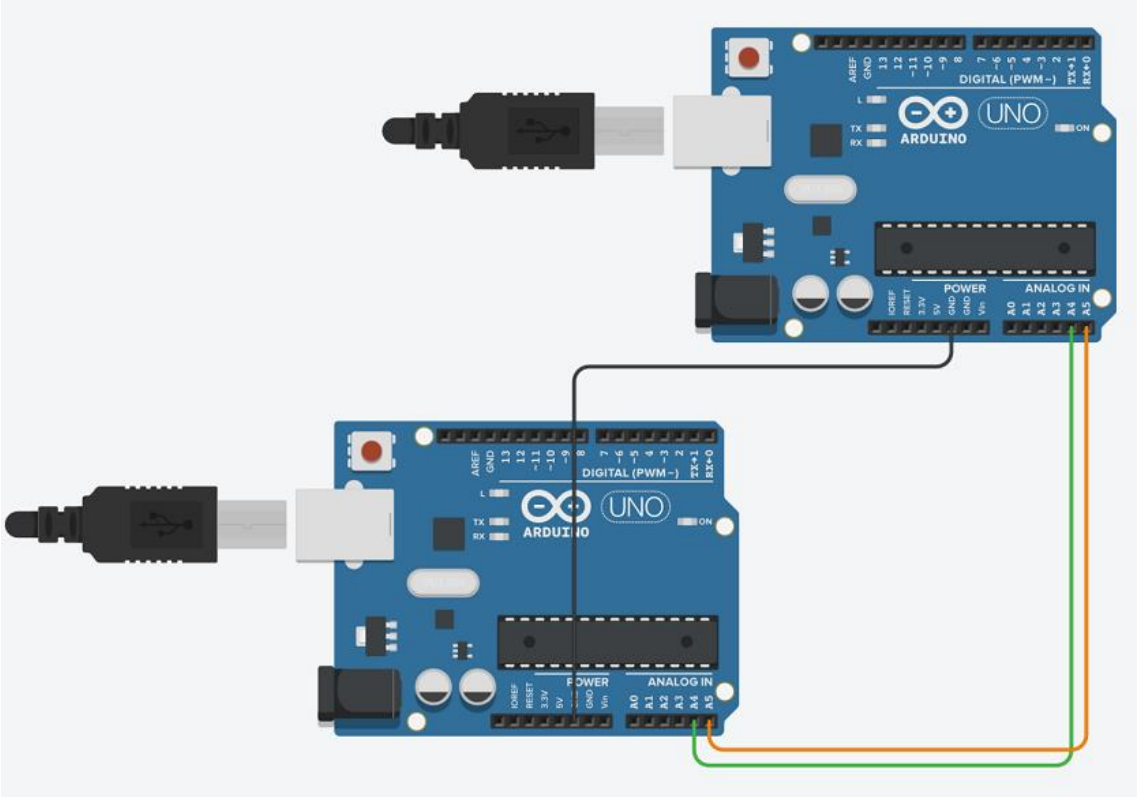
Para usar el bus I2C en Arduino, el IDE nos proporciona la librería "Wire.h", que contiene las funciones necesarias para controlar el hardware. Algunas de las funciones principales son:

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

Escáner de I2C

En ocasiones **el fabricante del hardware no nos facilita la dirección del dispositivo** o lo hace de forma incorrecta. Es bastante común, pero es sencillo realizar un escáner que realice un barrido por todas las posibles direcciones del bus, y muestre la dirección del dispositivo o dispositivos conectados.

Como ejemplo, nos dirigiremos a Tinkercad y realizaremos un circuito que conectará dos placas de Arduino, y simplemente mostrará en el monitor serie la dirección de la que establezcamos como esclavo.



Código maestro:

```
/*  MASTER  */

#include <Wire.h>

void setup() {
  Serial.begin(9600);
  Serial.println ("Escaneando dispositivos I2C: ");
  byte count = 0;

  Wire.begin();
  for(byte i = 1; i < 120; i++) {
    Wire.beginTransmission(i);
    // https://www.arduino.cc/en/Reference/WireEndTransmission
    if (Wire.endTransmission() == 0) {
      Serial.print("Encontrado dispositivo en la direccion: ");
      Serial.print(i, DEC);
      Serial.print(" (0x");
      if(i<16) Serial.print("0");
      Serial.print(i, HEX);
      Serial.println(")");
      count++;
      delay (1);
    }
  }
```

```
  Serial.print ("Encontrados ");
  Serial.print (count, DEC);
  Serial.println (" dispositivo(s).");
}

void loop() {

}
```

Código esclavo:

```
/* SLAVE */

#include <Wire.h>

void setup() {
  Wire.begin(15);
}

void loop() {

}
```