Proiectare cu Microprocesoare

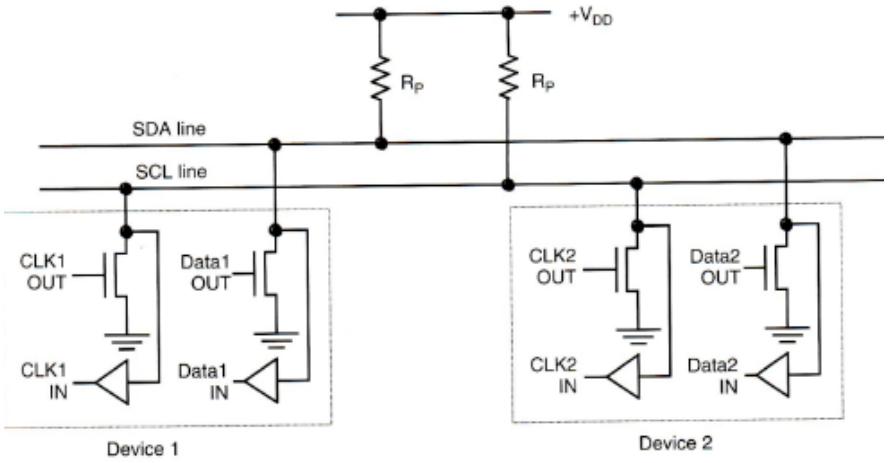
Curs 7

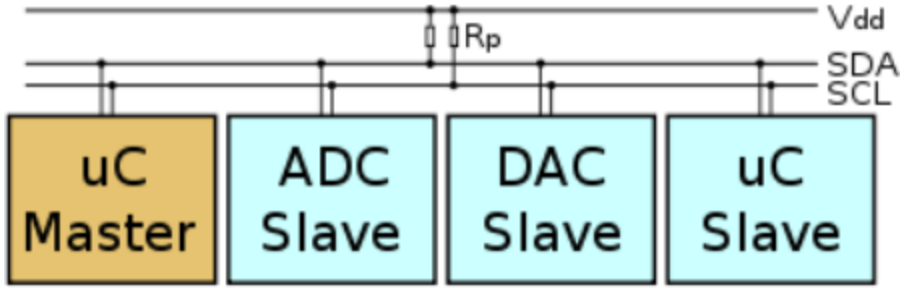
Interfata I2C. Interfete seriale la Arduino

An 3 CTI
Semestrul I

Lector: Răzvan Itu

I²C (Inter-Integrated Circuit)





Structura

- *SDA, SCL* – date si clock, bidirectionale, conectate “open collector” sau “open drain”
- Rezistente “pull up” tin linia la Vcc
- O linie poate fi trasa la ‘0’ de orice dispozitiv – ea va deveni ‘0’ daca cel putin un dispozitiv scrie ‘0’, altfel e ‘1’
- Fiecare dispozitiv are o adresa
- 7 biti de adresa
- 16 adrese rezervate
- 112 adrese disponibile – 112 dispozitive pe un bus

I^2C (Inter-Integrated Circuit)

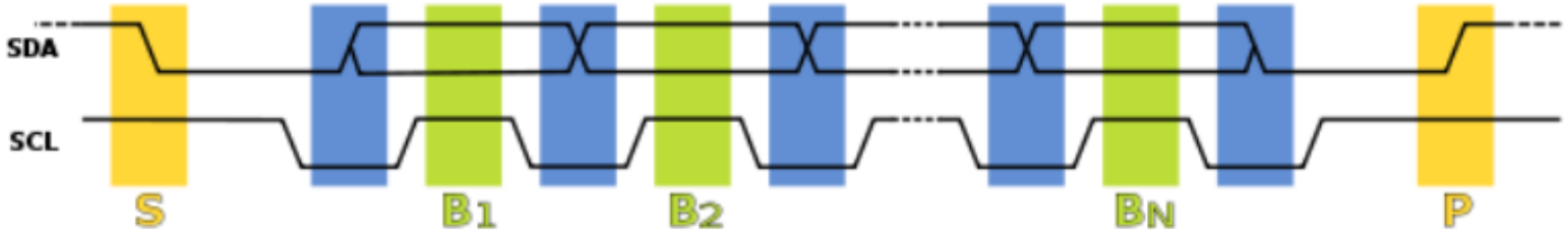
Tipuri de dispozitive (noduri)
- *Master – genereaza semnalul de ceas, si adresele*
- *Slave – primeste semnal de ceas, si adresa*

Rolul de master si slave se poate schimba pentru acelasi dispozitiv

Moduri de operare
- *master transmit* — nodul master trimite date la slave
- *master receive* — nodul master receptioneaza date de la slave
- *slave transmit* — nodul slave trimite date la master
- *slave receive* — nodul slave primeste date de la master

I²C (Inter-Integrated Circuit)

Diagrame de timp

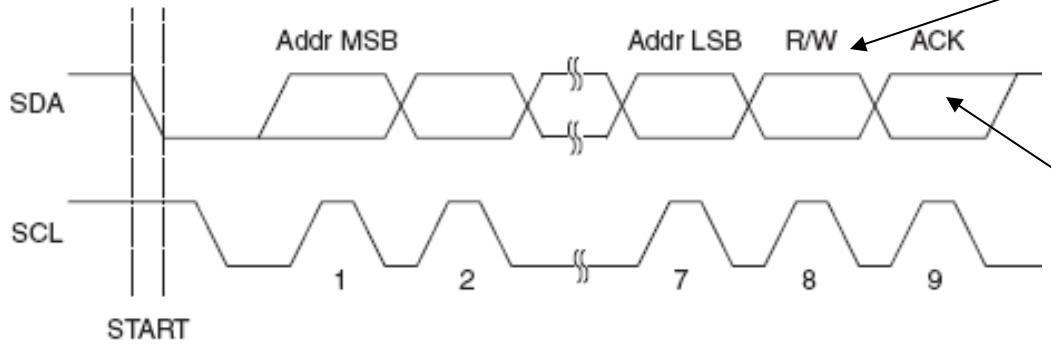


Evenimente

- *Start* - o tranzitie a SDA din ‘1’ in ‘0’, cu SCL mentinut pe ‘1’
- *Transfer biti* – valoarea bitului in SDA se schimba cand SCL e ‘0’, si este mentinuta stabila pentru preluare cand SCL e ‘1’
- *Stop* – o tranzitie a SDA din ‘0’ in ‘1’ cand SCL e ‘1’

I²C (Inter-Integrated Circuit)

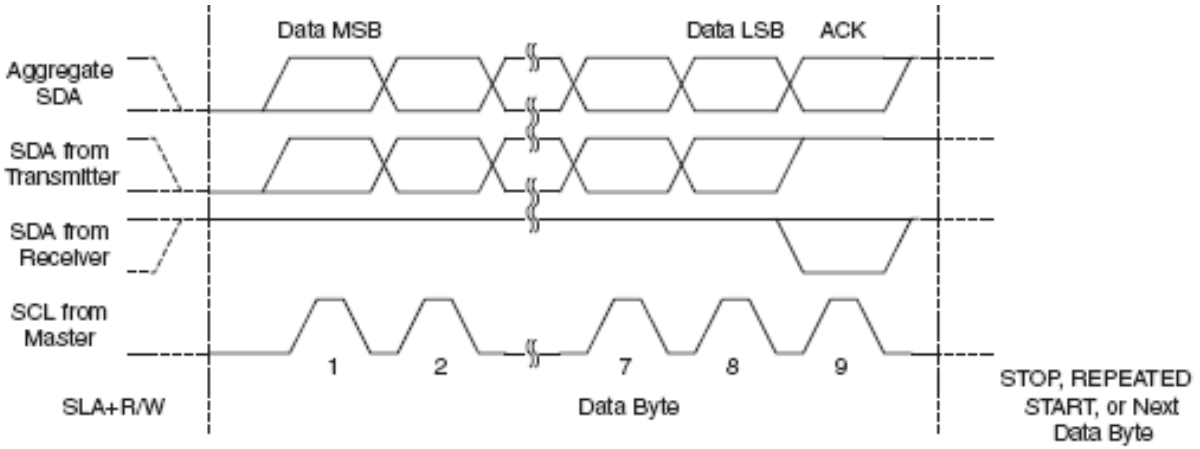
Formatul pachetelor - adresa



‘0’ – write
‘1’ – read

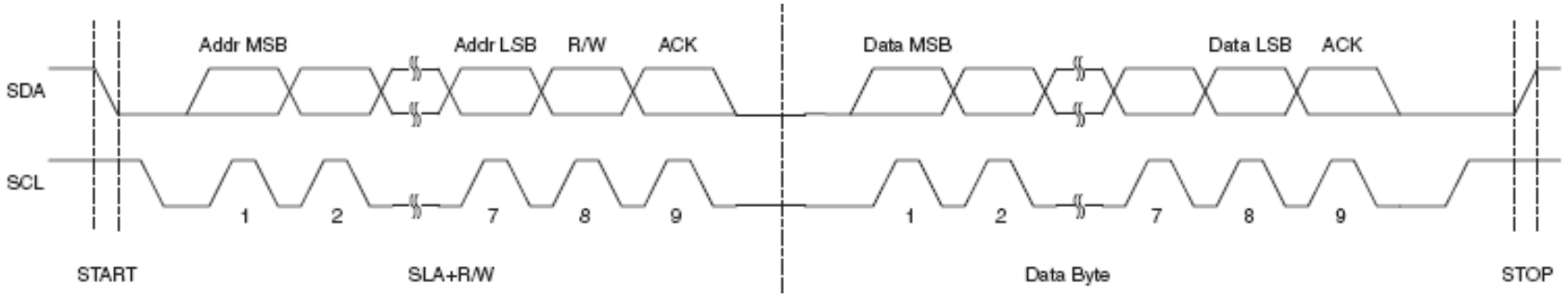
Scris de slave
‘0’ – ACK
‘1’ – NACK

Formatul pachetelor - date



I²C (Inter-Integrated Circuit)

Transmisie tipica



Arbitrare

- Fiecare master monitorizeaza semnalele de START si STOP, si nu initiaza un mesaj cat timp alt master tine bus-ul ocupat.
- Daca doi master initiaza transmisie in acelasi timp, se produce arbitrarea, pe baza liniei SDA.
    - Fiecare master verifica linia SDA, si daca nivelul acesteia nu este cel asteptat (cel scris), acel master pierde arbitrarea
    - Primul master care pune pe linie ‘1’ cand altul pune ‘0’ pierde.
    - Master-ul care pierde asteapta un semnal de stop, apoi incearca din nou

I^2C (Inter-Integrated Circuit)

Reglarea vitezei de transmisie

- ‘Clock stretching’
- Un dispozitiv slave poate tine linia de ceas (SCL) la ‘0’ mai mult timp, indicand necesitatea unui timp mai lung pentru procesarea datelor
- Dispozitivul master va incerca sa puna linia SCL pe ‘1’, dar va esua din cauza configuratiei ‘open collector’
- Dispozitivul master va verifica daca linia SCL a fost pusa pe ‘1’ si va continua transmisia cand acest lucru se intampla

- Nu toate dispozitivele suporta ‘clock stretching’
- Exista limite pentru intervalul de stretching

I^2C (Inter-Integrated Circuit)

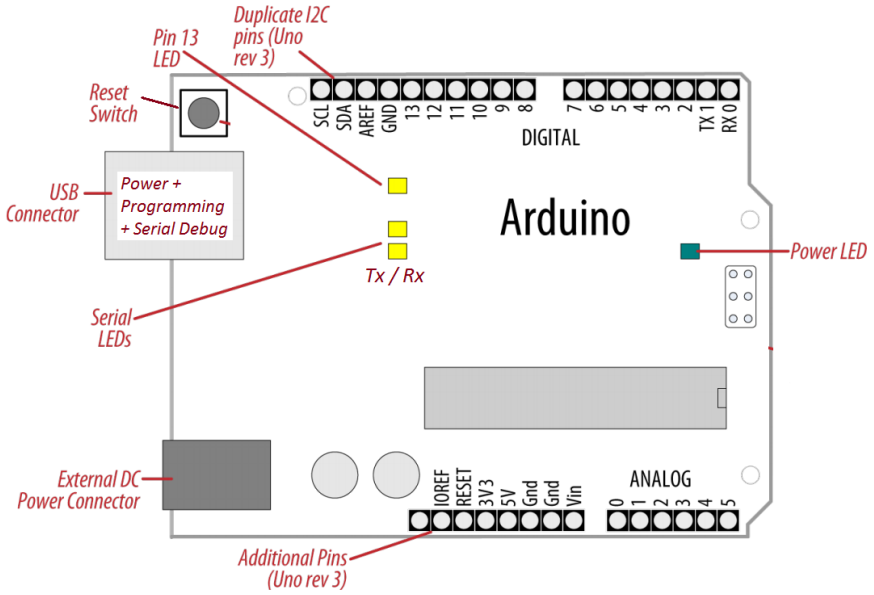
Aplicatii

- Citirea datelor de configutare din SPD EEPROM-ul din SDRAM, DDR SDRAM, DDR2 SDRAM
- Interfatare senzori digitali
- Accesarea convertoarelor DAC si ADC.
- Schimbarea setarilor in monitoare video (Display Data Channel).
- Schimbarea volumului in boxele inteligente.
-Citirea starii senzorilor de diagnostic hardware, precum termostatul CPU si viteza ventilatorului.

Avantaje

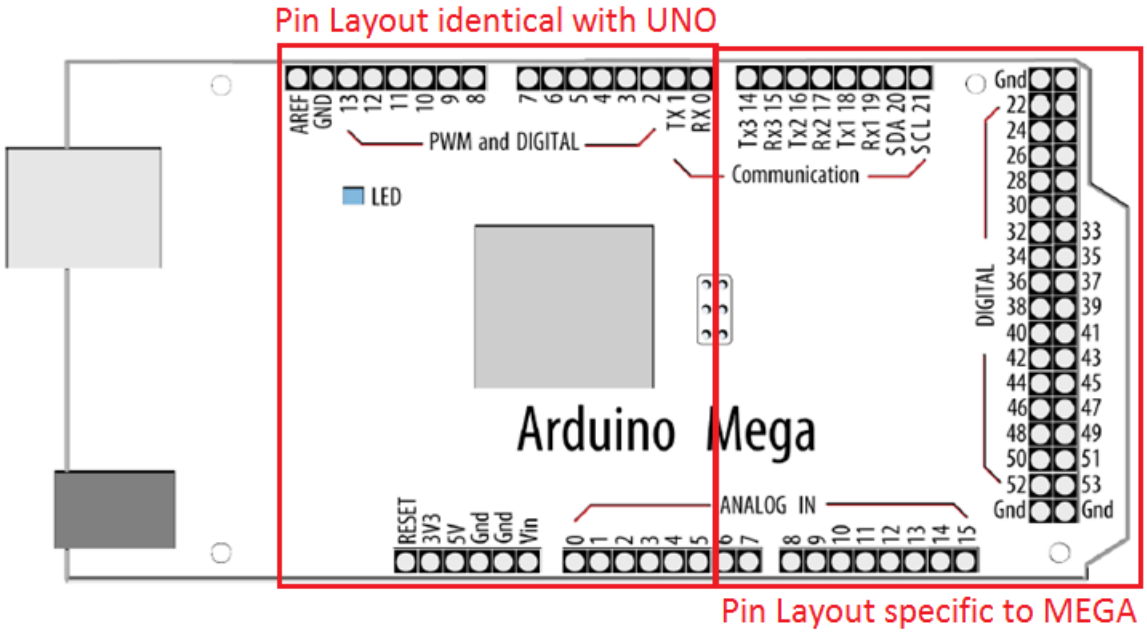
- Posibilitatea ca un microcontroller sa controleze multiple dispozitive atasate prin doar doua fire.
- Posibilitatea ca dispozitive sa fie atasate sau eliminate de pe bus in timpul functionarii

Interfete seriale la Arduino



Arduino UNO (rev. 3)

- **UART**: pin 0 (RX) , pin 1 (TX) – Utilizaţi pentru programare!
- **SPI**: pin 10 (SS), pin 11 (MOSI), pin 12 (MISO), pin 13 (SCK).
- **TWI(I2C)**: A4 sau pin SDA, A5 sau pin SCL



Arduino MEGA (rev. 3)

- **UART0**: pin 0 (RX) si pin 1 (TX) - programare; **UART1**: pin 19 (RX) si pin 18 (TX); **UART2**: pin 17 (RX) si 16 (TX); **UART3**: pin 15 (RX) si 14 (TX).
- **SPI:** pin 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS)
- **TWI(I2C)**: pin 20 (SDA) si 21 (SCL)

UART la Arduino

Toate placile Arduino au cel putin un port serial (port UART sau USART), accesibil prin obiectul C++ **Serial**

- Comunicare C ↔ PC prin conexiunea USB, folosind adaptorul USB-UART integrat pe placa – comunicare folosită şi pentru programarea plăcii !!
- Comunicare între plăci, folosind pinii digitali 0 (RX) si 1 (TX) - nerecomandat.

Pentru a putea folosi aceste tipuri de comunicatie, nu folositi pinii 0 si 1 pentru operatii generale de I/O digital !!!

Placa Arduino MEGA are trei porturi seriale suplimentare: **Serial1** foloseste pinii 19 (RX) si 18 (TX), **Serial2** pinii 17 (RX) si 16 (TX), **Serial3** pinii 15 (RX) si 14 (TX).

- Pentru a comunica cu un PC, este nevoie de un adaptor USB-UART extern, sau un adaptor UART-RS232, deoarece aceste interfete nu folosesc adaptorul integrat pe placa.
- Pentru a comunica cu un dispozitiv care foloseste interfata seriala cu nivele logice TTL, se conecteaza pinul TX al placii la pinul RX al dispozitivului, pinul RX al placii la pinul TX al dispozitivului, si pinii de masa (GND) impreuna.

UART la Arduino

Biblioteca **Serial** integrata in mediul de dezvoltare Arduino (http://arduino.cc/en/Reference/Serial) – folosita pentru facilitarea comunicatiei prin interfetele seriale disponibile.

Metodele clasei **Serial** (selectie):

- **Serial.begin(speed)** – configureaza viteza de transmisie (speed) si formatul implicit de date (8 data bits, no parity, one stop bit)
- **Serial.begin(speed, config)** – configureaza viteza (speed) + selecteaza un alt format al datelor (config):

  Exemple pentru *config*: SERIAL_8N1 (implicit), SERIAL_7E2, SERIAL_5O1 …
- **Serial.print(val)** – trimite valoarea val ca un sir de caractere citibil de catre utilizator (ex: Serial.print(20) va trimite caracterele ‘2’ si ‘0’)
- **Serial.print(val, format)** – format specifica baza de numeratie folosita (BIN, OCT , DEC, HEX. Pentru numere in virgula mobila, format specifica numarul de zecimale folosit.
- **Serial.println** – Trimite datele, plus (ASCII 13, sau '\r') + (ASCII 10, sau '\n')

Example:

Serial.print(78) transmite "78“ Serial.print(78, BIN) transmite "1001110"
Serial.print(1.23456) transmite "1.23“ Serial.println(1.23456, 4) transmite "1.2346"
Serial.print("Hello") transmite "Hello"

UART la Arduino

Metodele clasei **Serial** (selectie):

- byte IncomingByte = **Serial.read()** - citeste un byte prin interfata seriala
- int NoOfBytesSent = **Serial.write(data)** – scrie date in format binar prin interfata seriala. Datele se pot scrie ca un byte (val) sau ca un sir de octeti specificat ca un string (str) sau ca un sir buf, de lungime specificata len (buf, len)
- **Serial.flush**() – asteapta pana cand transmisia datelor pe interfata seriala este completa.
- int NoOfBytes = **Serial.available()** – Returneaza numarul de octeti disponibili pentru a fi cititi prin interfata seriala. Datele sunt deja primite si stocate intr-o zona de memorie buffer (capacitate maxima 64 octeti)
- **serialEvent()** – functie definita de utilizator, care este apelata automat cand exista date disponibile in zona buffer. Folositi **Serial.read()** in aceasta functie, pentru a citi aceste date.
- serialEvent1(), serialEvent2(), serialEvent3() – Pentru Arduino Mega, functii care se apeleaza automat pentru celelalte interfete seriale.

UART la Arduino

Exemplu program 1:

```
void setup() {
    Serial.begin(9600); // deschide portul serial, configureaza viteza la 9600 bps
    Serial.println(“Hello”);
}

void loop() {}
```

Exemplu program 2 (doar pentru Arduino Mega):

```
// Arduino Mega foloseste patru porturi seriale
// (Serial, Serial1, Serial2, Serial3),
// se pot configura cu viteze diferite:

void setup(){
  Serial.begin(9600);
  Serial1.begin(38400);
  Serial2.begin(19200);
  Serial3.begin(4800);

  Serial.println("Hello Computer");
  Serial1.println("Hello Serial 1");
  Serial2.println("Hello Serial 2");
  Serial3.println("Hello Serial 3");
}

void loop() {}
```

UART la Arduino

Comunicatie □C ↔ PC: receptionarea datelor seriale in Arduino – receptionarea datelor de la un PC sau un alt dispozitiv serial, si reactionarea la comenzi:

Examplu program 3 – receptioneaza o cifra (caracter de la ‘0’ la ‘9’) si modifica starea unui LED proportional cu cifra citita.

```
const int ledPin = 13;              // pin LED
int blinkRate=0;                    // rata de modificare a starii
void setup()
{
  Serial.begin(9600); // initializare port serial
  pinMode(ledPin, OUTPUT); // configurare pin LED ca iesire
}
void loop() {
  if ( Serial.available())          // Verifica data avem date de citit
  {
        char ch = Serial.read(); // citeste caracterul receptionat
        If( isDigit(ch) )           // Este cifra ?
        {
            blinkRate = (ch - '0');             // Se converteste in valoare numerica
            blinkRate = blinkRate * 100; // Rata = 100ms * cifra citita
        }
  }
  blink();
}
```

UART la Arduino

Exemplu program 3 – cont.

```
// modifica stare LED pe baza ratei calculate
void blink()
{
  digitalWrite(ledPin,HIGH);
  delay(blinkRate); // intarzierea calculata
  digitalWrite(ledPin,LOW);
  delay(blinkRate);
}
```

Pentru a utiliza acest exemplu:

- Folositi Serial Monitor, inclus in mediul Arduino, activandu-l din meniul Tools sau apasand <CTRL+SHIFT+M>)
- Selectati aceeasi viteza cu cea pe care ati configurat-o apeland Serial.begin()
- Tastati cifre, si apasati butonul “Send”.

UART la Arduino

Exemplu program 4 – exemplul 3 modificat sa foloseasca **serialEvent()**

```
void loop()
{
  blink();
}

void serialEvent() // functia se apeleaza automat, cand exista date de citit
{
  while(Serial.available()) // cat timp exista date disponibile
  {
     char ch = Serial.read(); // acestea se citesc
     Serial.write(ch); // echo – trimitem datele inapoi pentru verificare
     if( isDigit(ch) ) // e cifra ?
     {
        blinkRate = (ch - '0'); // convertire la valoare numerica
        blinkRate = blinkRate * 100; // calcul timp de intarziere
     }
   }
}
```

Software UART la Arduino

- In afara interfetelor seriale hardware, Arduino permite comunicarea seriala si pe alti pini digitali, realizand procesul de serializare / de-serializare a datelor prin program

- Se foloseste biblioteca SoftwareSerial, inclusa in pachetul software Arduino (necesita includerea header-ului softwareserial.h)

- Pinul de receptie **(RX) trebuie conectat la un pin digital care suporta intreruperi externe** declansate prin schimbarea starii acestuia:

    - La Arduino Mega, acesti pini sunt: 10, 11, 12, 13, 14, 15, 50, 51, 52, 53, A8 (62), A9 (63), A10 (64), A11 (65), A12 (66), A13 (67), A14 (68), A15 (69)

- Se pot crea mai multe obiecte C++ de tip SoftwareSerial, dar numai unul poate fi activ la un moment dat

- Crearea unui obiect – trebuie specificati pinii de receptie si transmisie:

```
SoftwareSerial mySerial =  SoftwareSerial(rxPin, txPin)
```

- Sunt implementate functiile **begin**, **read**, **write**, **print**, **println**, care se folosesc in acelasi mod ca in cazul interfetelor seriale hardware

Software UART la Arduino

Exemplu: comunicarea folosind doua interfete seriale, una hardware, conectata la PC, si una software, conectata la un alt dispozitiv compatibil UART

-Arduino joaca rolul de releu de comunicatie: ce primeste pe o interfata, transmite pe cealalta.

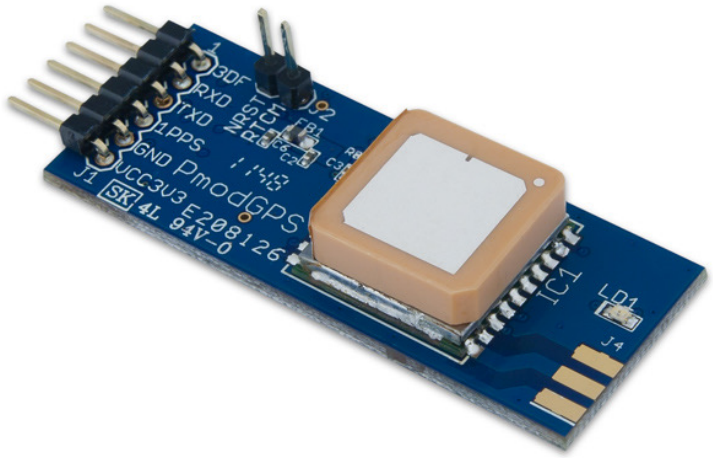
```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // Interfata software foloseste pin 10 pt RX, pin 11 pt TX

void setup()
{
 Serial.begin(9600); // Configurare interfata hardware
 mySerial.begin(9600); // Configurare interfata software
}

void loop()
{
  if (mySerial.available())
    Serial.write(mySerial.read());      // citire de pe interfata software, transmisie prin cea hardware
  if (Serial.available())
    mySerial.write(Serial.read());       // citire de pe interfata hardware, transmisie prin cea software
}
```

Software UART la Arduino

Exemplu: comunicarea folosind doua interfete seriale, una hardware, conectata la PC, si una software, conectata la un alt dispozitiv compatibil UART

- Posibila utilizare a programului anterior: vizualizare a datelor transmise de catre un dispozitiv compatibil UART pe terminalul Serial Monitor

- Exemplu: receptor GPS Digilent PMod GPS:




```
$GPGGA,064951.000,2307.1256,N,12016.4438,E,1,8,0.
95,39.9,M,17.8,M,,*65
$GPGSA,A,3,29,21,26,15,18,09,06,10,,,,,2.32,0.95,
2.11*00
$GPGSV,3,1,09,29,36,029,42,21,46,314,43,26,44,020
,43,15,21,321,39*7D
$GPRMC,064951.000,A,2307.1256,N,12016.4438,E,0.03
,165.48,260406,3.05,W,A*55
$GPVTG,165.48,T,,M,0.03,N,0.06,K,A*37
```

- Alte dispozitive UART care pot fi verificate/depanate in acest mod: adaptoare Bluetooth, adaptoare WiFi, etc.

Comunicare SPI la Arduino

Se foloseste biblioteca SPI (http://arduino.cc/en/Reference/SPI) [3]

Functii oferite:

- **SPI.setBitOrder(order)** – ordinea bitilor: LSBFIRST sau MSBFIRST
- **SPI.setDataMode(mode)** – faza si polaritatea transmisiei: SPI_MODE0, SPI_MODE1, SPI_MODE2 sau SPI_MODE3 (combinatiile CPHA si CPOL)
- **SPI.setClockDivider()** – divizorul de frecventa: SPI_CLOCK_DIV(2 .. 128)
- **SPI.begin()** – initializare interfata SPI, configurand pinii SCK, MOSI, si SS ca iesire, punand SCK si MOSI pe LOW, si SS pe HIGH.
- **SPI.end()** – dezactiveaza SPI, dar lasa pinii in modul in care au fost setati la initializare
- ReturnByte **SPI.transfer(val)** - transvera un byte (val) pe magistrala SPI, receptionand in acelasi timp octetul ReturnByte.

- Biblioteca SPI suporta doar modul master
- Orice pin poate fi folosit ca Slave Select (SS).

Comunicare SPI la Arduino

Exemplu (SPI) – Controlul unui potentiometru digital folosind SPI [4]

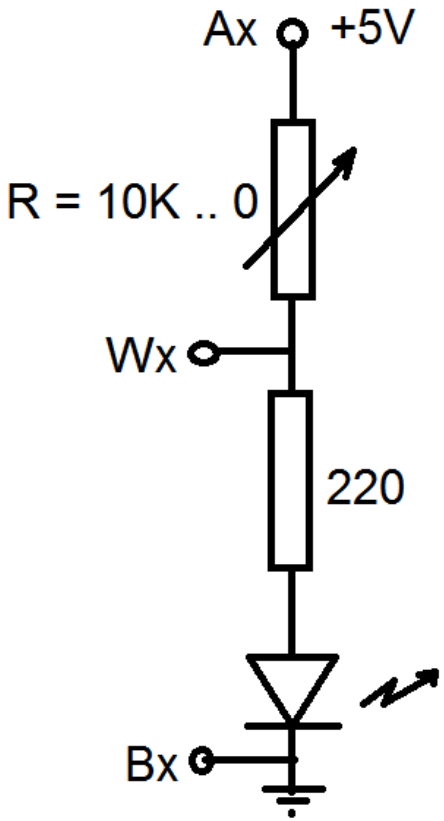
http://arduino.cc/en/Tutorial/SPIDigitalPot

http://www.youtube.com/watch?v=1nO2SSExEnQ

AD5206 datasheet: http://datasheet.octopart.com/AD5206BRU10-Analog-Devices-datasheet-8405.pdf

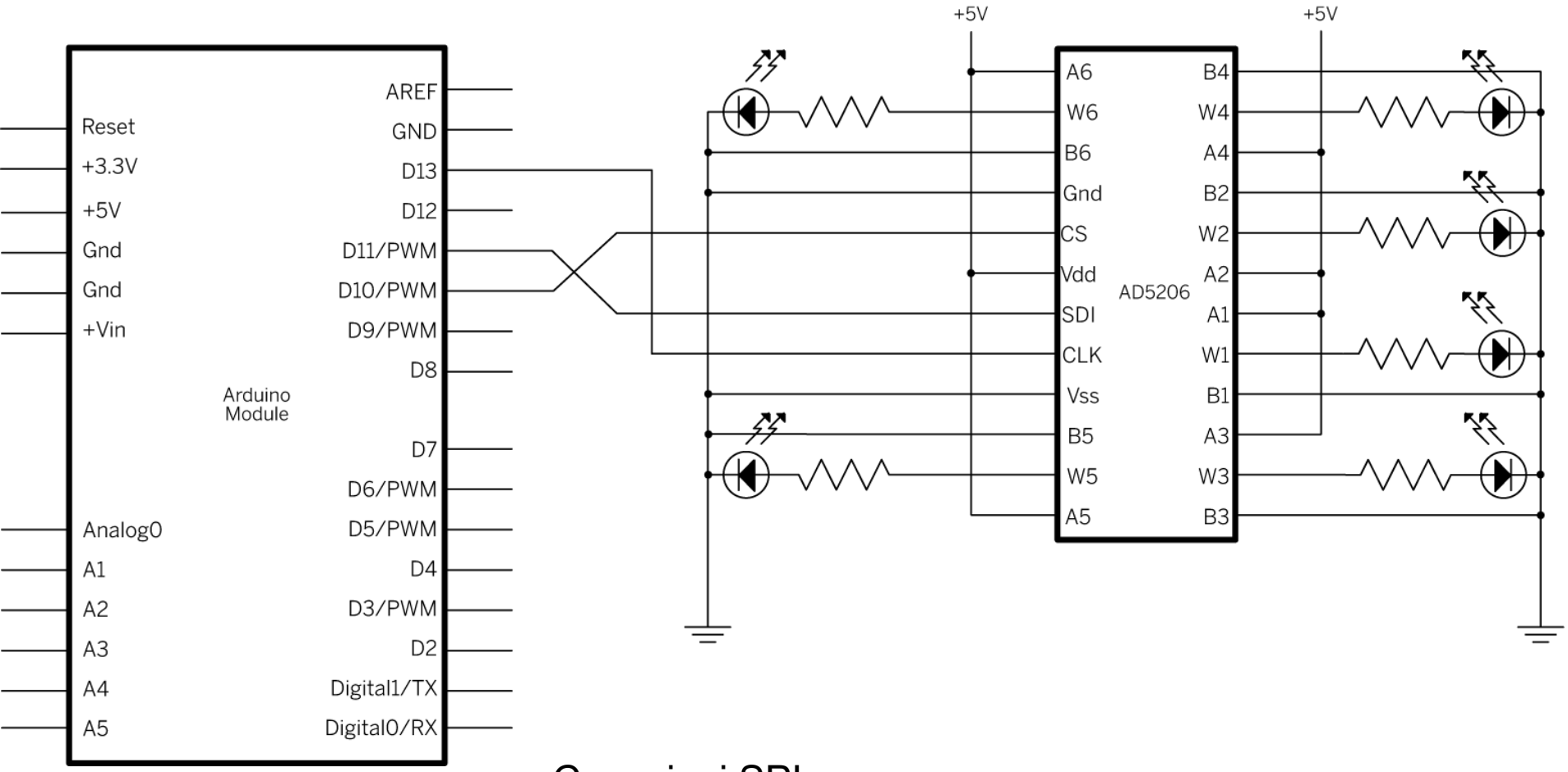
AD5206 este un potentiometru digital cu 6 canale (echivalent cu sase potentiometre individuale).

- 3 pini pe chip pentru fiecare element: Ax, Bx si Wx (Wiper - cursorul).
- pin A = HIGH, pin B = LOW si pinul W = tensiune variabila. R are rezistenta maxima de 10 Kohm, impartita in 255 pasi.
- Pentru controlul rezistentei, se trimit pe SPI doi octeti: primul pentru selectia canalului (0 - 5) si al doilea cu valoarea rezistentei pentru fiecare canal (0 - 255) .



Comunicare SPI la Arduino

Exemplu (SPI) – Controlul unui potentiometru digital folosind SPI



Conexiuni SPI

* CS – la pinul 10 (SS)
* SDI – la pinul 11 (MOSI)
* CLK – la pinul 13 (SCK)

Comunicare SPI la Arduino

Exemplu (SPI) – Controlul unui potentiometru digital folosind SPI

```
#include <SPI.h>
// pin 10 ca SS
const int slaveSelectPin = 10;

void setup() {
  // SS trebuie configurat ca iesire
pinMode (slaveSelectPin, OUTPUT);
SPI.begin();   // activare SPI
}

void loop() {
  // se iau cele 6 canale la rand
  for (int channel = 0; channel < 6; channel++) {
    // se schimba rezistenta fiecarui canal de la min la max
    for (int level = 0; level < 255; level++) {
      digitalPotWrite(channel, level);
      delay(10);
    }
     delay(100); // asteptam 100 ms cu rezistenta maxima
    // se schimba rezistenta de la max la min
    for (int level = 0; level < 255; level++) {
      digitalPotWrite(channel, 255 - level);
      delay(10);
    }
  }
}
```

```
// functia care foloseste SPI pentru actualizarea
// rezistentelor
void digitalPotWrite(int address, int value) {
  // activeaza SS prin scriere LOW
  digitalWrite(slaveSelectPin, LOW);
  // se trimit pe rand canalul si valoarea:
  SPI.transfer(address);
  SPI.transfer(value);
  // inactiveaza SS prin scriere HIGH
  digitalWrite(slaveSelectPin, HIGH);
}
```

Comunicare I2C la Arduino

- Se utilizeaza biblioteca **Wire**, din pachetul software Arduino

- O placa Arduino poate fi I2C master sau I2C slave

- Metodele obiectului Wire:

- **Wire.begin(address)** – activeaza interfata I2C. Parametrul address indica adresa de 7 biti cu care Arduino se ataseaza la bus-ul I2C ca slave. Daca functia este apelata fara parametri, dispozitivul este master.

- **Wire.requestFrom(address, quantity)** – master cere o cantitate **quantity** date de la un slave identificat prin **address**. Functia poate fi apelata si ca **Wire.requestFrom(address, quantity, stop)** , stop fiind o valoare booleana specificand daca masterul va elibera bus-ul, sau va mentine conexiunea activa.

- **Wire.beginTransmission(address)** – incepe procesul de transmisie dinspre master catre un slave specificat de **address**. Dupa apelul acestei functii, datele pot fi scrise cu write() .

- **Wire.write(value)** – scrie un byte. Functia poate fi apelata de slave, daca a fost solicitat de catre master, sau de master, dupa ce a apelat beginTransmission. Alternative: Wire.write(string), sau Wire.write(data, length) .

- **Wire.endTransmission()** – apelat de master pentru a realiza efectiv transmisia inceputa cu beginTransmission, cu datele pregatite prin apelul **Wire.write()**.

Comunicare I2C la Arduino

- **Wire.available()** – returneaza numarul de bytes disponibili pentru a fi cititi.

- **Wire.read()** – citeste un byte, daca available() > 0. Apelabila si de master, si de slave.

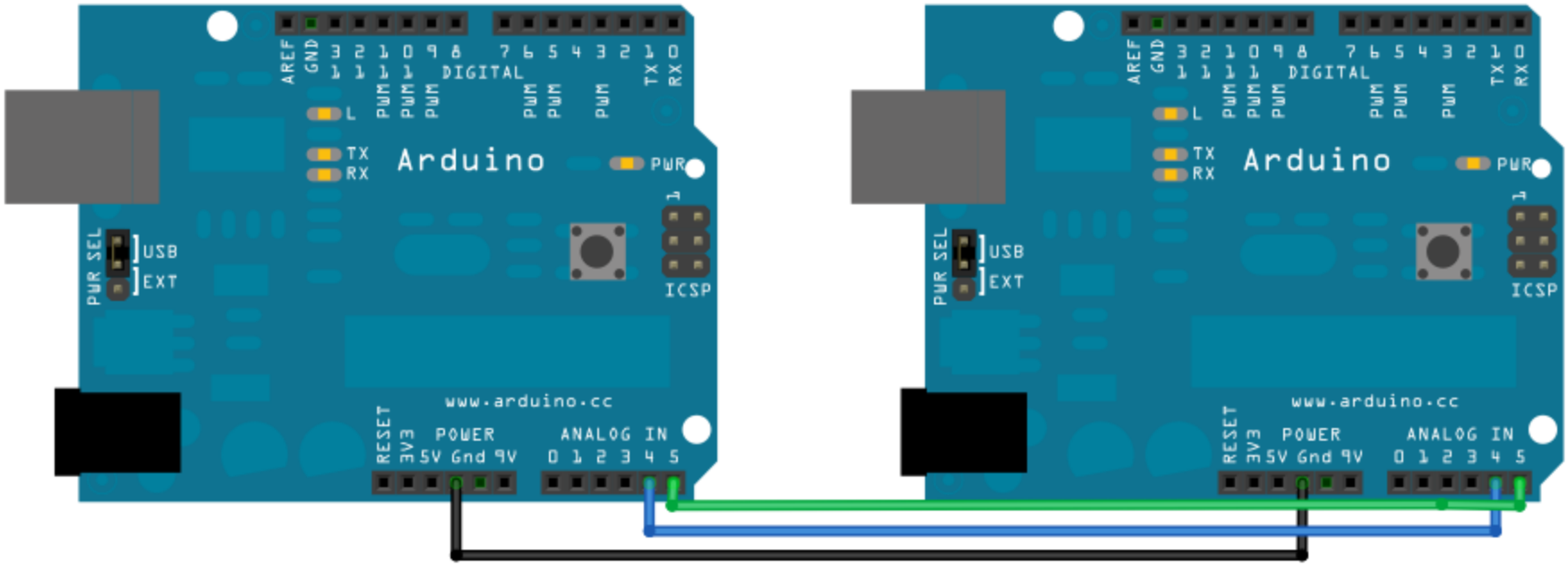
- **Wire.onReceive(handler)** – configureaza o **functie handler**, la dispozitivul slave, care va fi apelata automat la primirea datelor de la master.

- **Wire.onRequest(handler)** – configureaza o functie handler, la dispozitivul slave, care va fi apelata automat atunci cand masterul cere date.

Comunicare I2C la Arduino

Exemplu: conectarea a doua placi Arduino prin I2C, una avand rol de master, care va transmite datele, si una de slave, care le va receptiona.

Sursa: http://arduino.cc/en/Tutorial/MasterWriter



Comunicare I2C la Arduino

Exemplu: conectarea a doua placi Arduino prin I2C, una avand rol de master, care va transmite datele, si una de slave, care le va receptiona.

Cod master:

```
#include <Wire.h>
void setup()
{
  Wire.begin(); // activeaza interfata I2C ca master
}

byte x = 0; // valoarea de transmis, se va incrementa

void loop()
{
  Wire.beginTransmission(4);  // incepe un proces de transmisie, catre adresa slave 4
  Wire.write("x is ");        // scriere string
  Wire.write(x);              // scriere un byte
  Wire.endTransmission();    // finalizeaza tranzactia de scriere
  x++; // incrementare valoare
  delay(500);
}
```

Comunicare I2C la Arduino

Exemplu: conectarea a doua placi Arduino prin I2C, una avand rol de master, care va transmite datele, si una de slave, care le va receptiona.

Cod slave:

```
#include <Wire.h>
void setup()
{
  Wire.begin(4);                          // activeaza interfata i2c ca slave, cu adresa 4
  Wire.onReceive(receiveEvent);   // inregistreaza functia receiveEvent pentru a fi apelata la venirea datelor
  Serial.begin(9600);                    // activeaza interfata seriala, pentru a afisa pe PC datele primite
}

void loop()  // functia loop nu face nimic
{
  delay(100);
}

void receiveEvent(int howMany) // functie apelata cand slave primeste date
{
  while(Wire.available() > 1) // parcurge cantitatea de date, lasand ultimul octet separat
  {
    char c = Wire.read(); // citeste caracter cu caracter
    Serial.print(c);         // trimite caracterul la PC
  }
  int x = Wire.read();    // ultimul caracter este tratat ca o valoare numerica
  Serial.println(x);         // si va fi tiparit ca atare
}
```

Bibliografie

[1] Arduino Serial reference guide: http://arduino.cc/en/Reference/Serial

[2] Michael Margolis, Arduino Cookbook, 2-nd Edition, O'Reilly, 2012.

[3] Arduino SPI reference guide: http://arduino.cc/en/Reference/SPI

[4] Arduino SPI Tutorials: http://arduino.cc/en/Tutorial/SPIDigitalPot

[5] Arduino I2C Library: http://arduino.cc/en/reference/wire

[6] Arduino I2C tutorial: http://arduino.cc/en/Tutorial/MasterWriter