Mini Projet

LE BRACELET MÉDICAL CONNECTÉ

Revue de projet n°2

Chloé Josse – BTS SN 1

SOMMAIRE

ENVIRONNEMENT ARDUINO

	Arduino Uno ver R3	Arduino Mega 2560	Arduino Nano
Microcontrôleur	ATmega328	ATmega2560	ATmega168/ATmega328
Dimension	69mm*54mm	101mm*53mm	45mm*18mm
Tension de fonctionnement	5V	5V	5V
Tension d'alimentation	7-12V	7-12V	7-12V
Tension d'alimentation	6-20V	6-20V	5V
Broches E/S numériques	14/6 PWM	54/14 PWM	14 /6 PWM
Broches E/S analogiques	6	16	8
Intensité des broches E/S (5V)	40 mA	40 mA	40 mA
Mémoire Programme Flash	32 KB	256 KB	16KB/32 KB
Mémoire SRAM	2 KB (ATmega328)	8 KB	1KB/2 KB (ATmega328)
Mémoire EEPROM	1 KB (ATmega328)	4 KB	512 bytes/KB
Vitesse d'horloge	16 MHz	16 MHz	16 MHz
SPI/I2C	DISPONIBLE	DISPONIBLE	DISPONIBLE

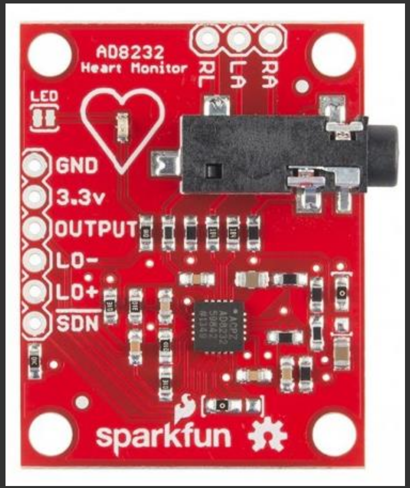
Suffisamment simple pour être accessible à n'importe qui, mais suffisamment riche et puissant pour piloter des expériences scientifiques complexes.

Nb entre sortie

Ram

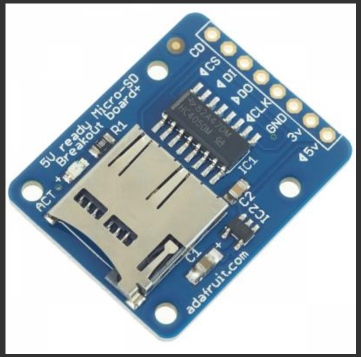
Courants delivre par l'arduino

COMPOSANT MIS EN ŒUVRE

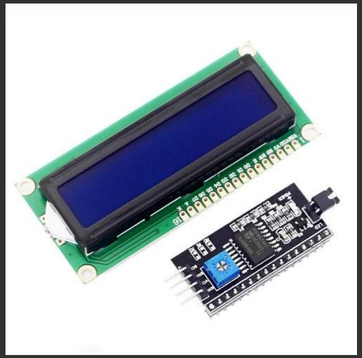


Moniteur Cardiaque (Kit ECG AD8232)

Capteur de distance VL53LoX



Micro SD card Breakout board



Afficheur LCD en I2C

Régulateur

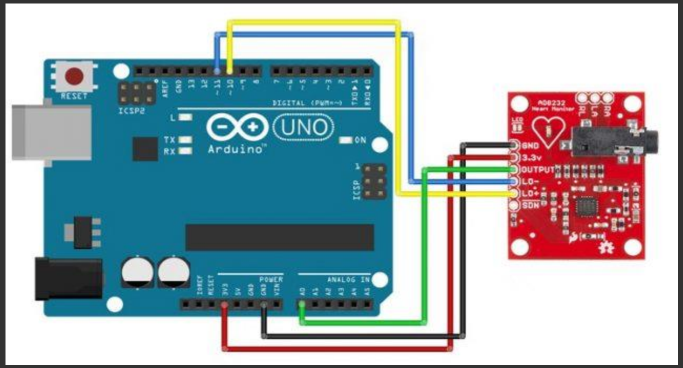
Arduino Nano

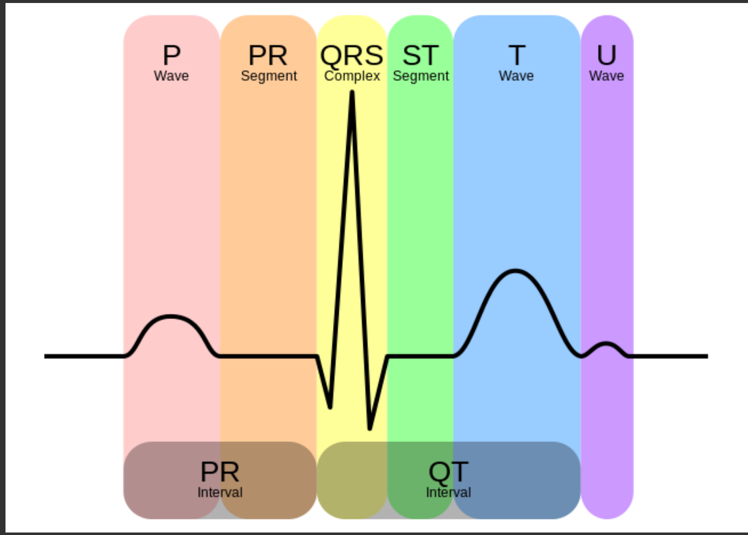
Batterie LiPo

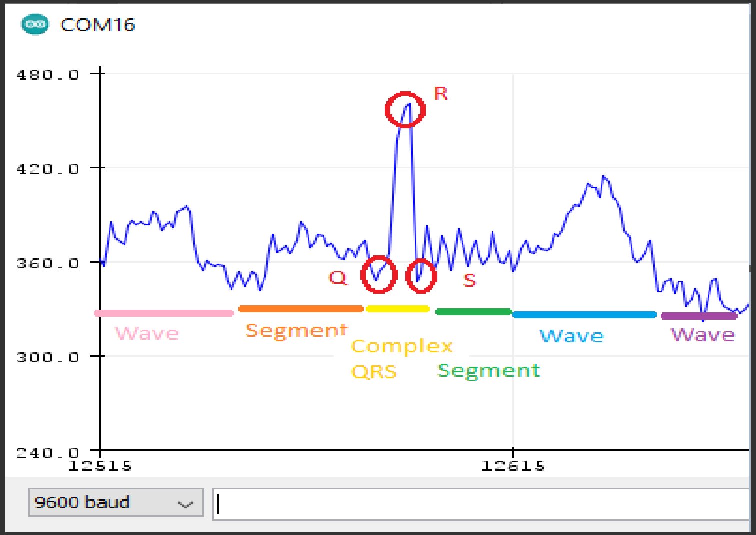
MONITEUR CARDIAQUE
(KIT ECG AD8232)

Heart_Monitor_2

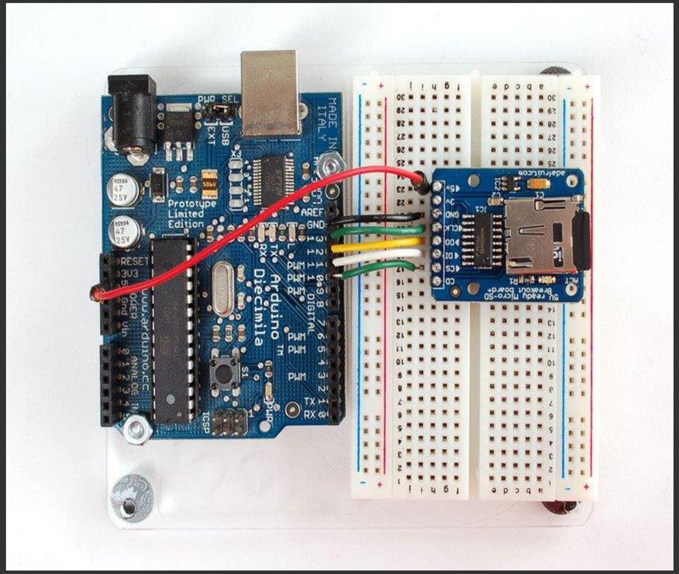
```
void setup() {
// initialize the serial communication:
Serial.begin(9600);
pinMode(10, INPUT); // Setup for leads off detection LO +
pinMode(11, INPUT); // Setup for leads off detection LO -

}

void loop() {

if((digitalRead(10) == 1)||(digitalRead(11) == 1)){
Serial.println('!');
}
else{
// send the value of analog input 0:
Serial.println(analogRead(A0));
}
//Wait for a bit to keep serial data from saturating
delay(1);
}
```







Micro SD card Breakout board

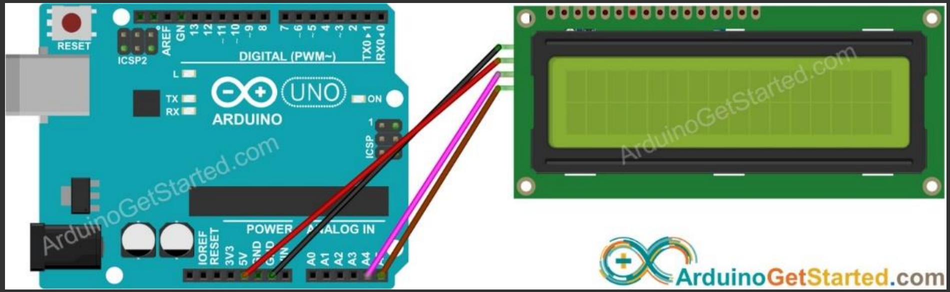


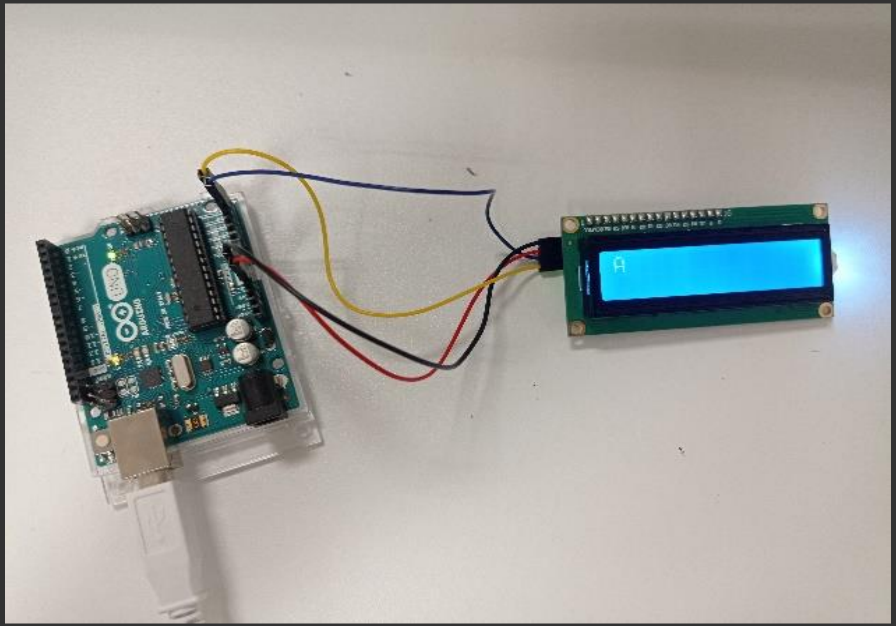
CardInfo §

```
// include the SD library:
#include <SPI.h>
#include <SD.h>

// set up variables using the SD utility library functions:
Sd2Card card;
SdVolume volume;
SdFile root;

// change this to match your SD shield or module;
// Arduino Ethernet shield: pin 4
// Adafruit SD shields and modules: pin 10
// Sparkfun SD shield: pin 8
// MKRZero SD: SDCARD_SS_PIN
const int chipSelect = 10;

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }


  Serial.print("\nInitializing SD card...");

  // we'll use the initialization code from the utility libraries
  // since we're just testing if the card is working!
  if (!card.init(SPI_HALF_SPEED, chipSelect)) {
    Serial.println("initialization failed. Things to check:");
    Serial.println("* is a card inserted?");
    Serial.println("* is your wiring correct?");
    Serial.println("* did you change the chipSelect pin to match your shield or module?");
    while (1);
  } else {
    Serial.println("Wiring is correct and a card is present.");
  }

  // print the type of card
  Serial.println();
  Serial.print("Card type:         ");
  switch (card.type()) {
    case SD_CARD_TYPE_SD1:
      Serial.println("SD1");
      break;
    case SD_CARD_TYPE_SD2:
      Serial.println("SD2");
```

```
    case SD_CARD_TYPE_SDHC:
      Serial.println("SDHC");
      break;
    default:
      Serial.println("Unknown");
  }

  // Now we will try to open the 'volume'/'partition' - it should be FAT16 or FAT32
  if (!volume.init(card)) {
    Serial.println("Could not find FAT16/FAT32 partition.\nMake sure you've formatted the card");
    while (1);
  }

  Serial.print("Clusters:          ");
  Serial.println(volume.clusterCount());
  Serial.print("Blocks x Cluster:  ");
  Serial.println(volume.blocksPerCluster());

  Serial.print("Total Blocks:      ");
  Serial.println(volume.blocksPerCluster() * volume.clusterCount());
  Serial.println();

  // print the type and size of the first FAT-type volume
  uint32_t volumesize;
  Serial.print("Volume type is:    FAT");
  Serial.println(volume.fatType(), DEC);

  volumesize = volume.blocksPerCluster();    // clusters are collections of blocks
  volumesize *= volume.clusterCount();       // we'll have a lot of clusters
  volumesize /= 2;                           // SD card blocks are always 512 bytes (2 blocks are
  Serial.print("Volume size (Kb):  ");
  Serial.println(volumesize);
  Serial.print("Volume size (Mb):  ");
  volumesize /= 1024;
  Serial.println(volumesize);
  Serial.print("Volume size (Gb):  ");
  Serial.println((float)volumesize / 1024.0);

  Serial.println("\nFiles found on the card (name, date and size in bytes): ");
  root.openRoot(volume);

  // list all files in the card with date and size
  root.ls(LS_R | LS_DATE | LS_SIZE);
}

void loop(void) {
}
```

Afficheur LCD en I2C

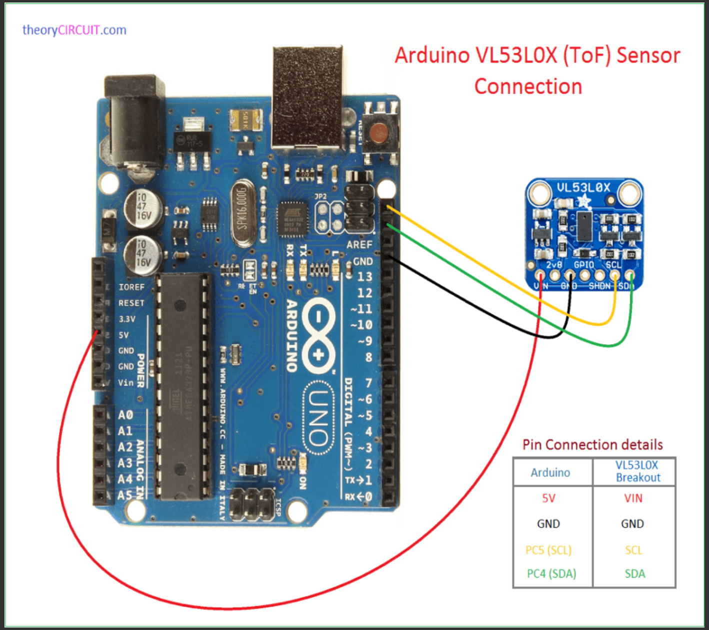




Afficheur_LCD_I2C

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address 0x27, 16 column and 2 rows

void setup()
{
  lcd.init(); // initialize the lcd
  lcd.backlight();

  lcd.setCursor(0, 0);         // move cursor to   (0, 0)
  lcd.print("A");        // print message at (0, 0)
}

void loop()
{
}
```

Capteur de distance VL53L0X



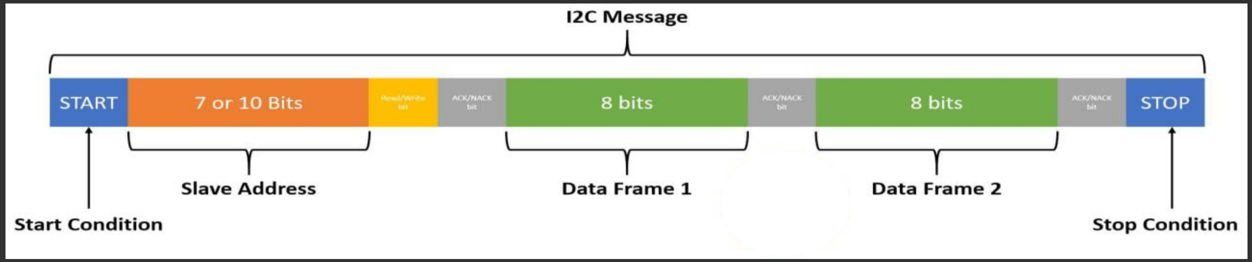
vl53l0x

```
#include "Adafruit_VL53L0X.h"

Adafruit_VL53L0X lox = Adafruit_VL53L0X();

void setup() {
  Serial.begin(115200);

  // wait until serial port opens for native USB devices
  while (! Serial) {
    delay(1);
  }

  Serial.println("Adafruit VL53L0X test");
  if (!lox.begin()) {
    Serial.println(F("Failed to boot VL53L0X"));
    while(1);
  }
  // power
  Serial.println(F("VL53L0X API Simple Ranging example\n\n"));
}


void loop() {
  VL53L0X_RangingMeasurementData_t measure;

  Serial.print("Reading a measurement... ");
  lox.rangingTest(&measure, false); // pass in 'true' to get debug data printout!

  if (measure.RangeStatus != 4) {  // phase failures have incorrect data
    Serial.print("Distance (mm): "); Serial.println(measure.RangeMilliMeter);
  } else {
    Serial.println(" out of range ");
  }

  delay(100);
}
```

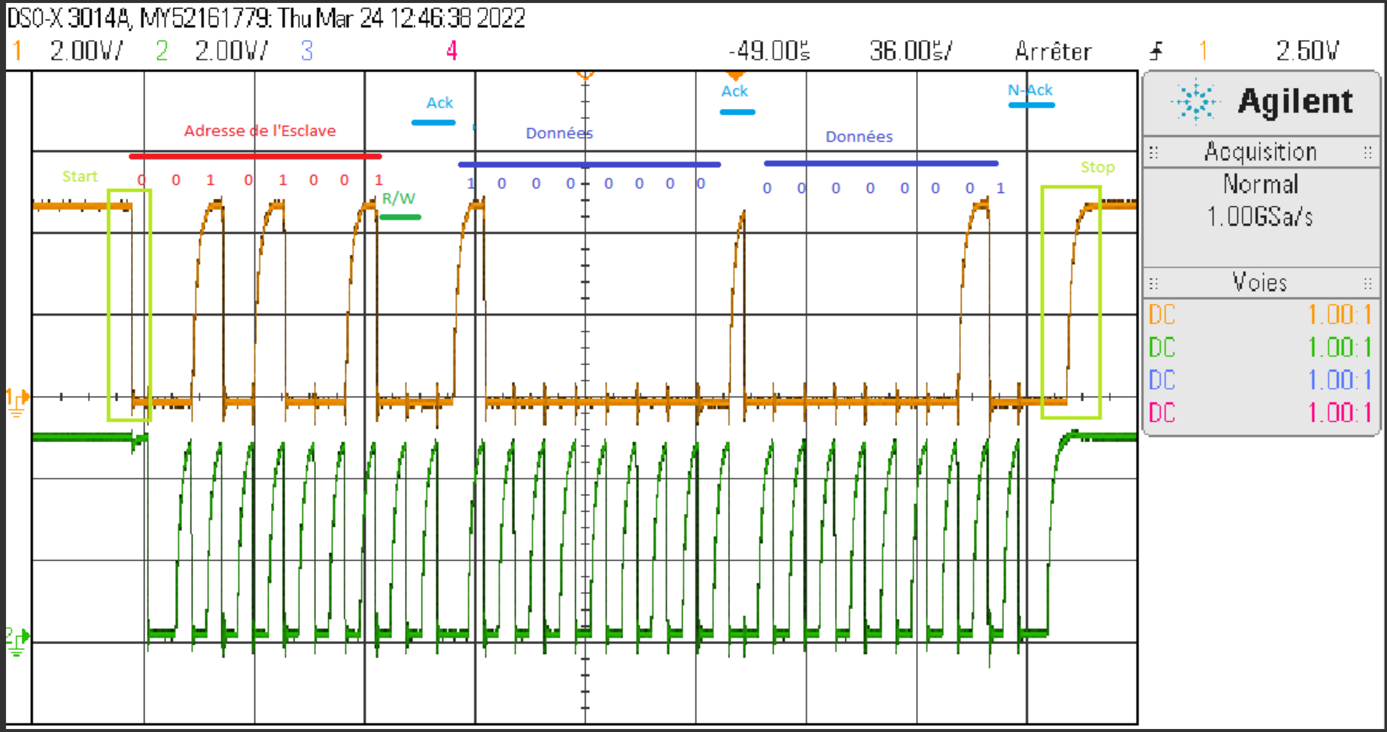
COM16

```
Reading a measurement... Distance (mm): 38
Reading a measurement... Distance (mm): 38
Reading a measurement... Distance (mm): 37
Reading a measurement... Distance (mm): 38
Reading a measurement... Distance (mm): 41
Reading a measurement... Distance (mm): 40
Reading a measurement... Distance (mm): 39
Reading a measurement... Distance (mm): 36
Reading a measurement... Distance (mm): 38
Reading a measurement... Distance (mm): 39
Reading a measurement... Distance (mm): 38
Reading a measurement... Distance (mm): 38
Reading a measurement... Distance (mm): 39
Reading a measurement... Distance (mm): 39
Reading a measurement... Distance (mm): 39
Reading a measurement... Distance (mm): 37
```

LIAISON I2C (INTER-INTERGRATED CIRCUIT)



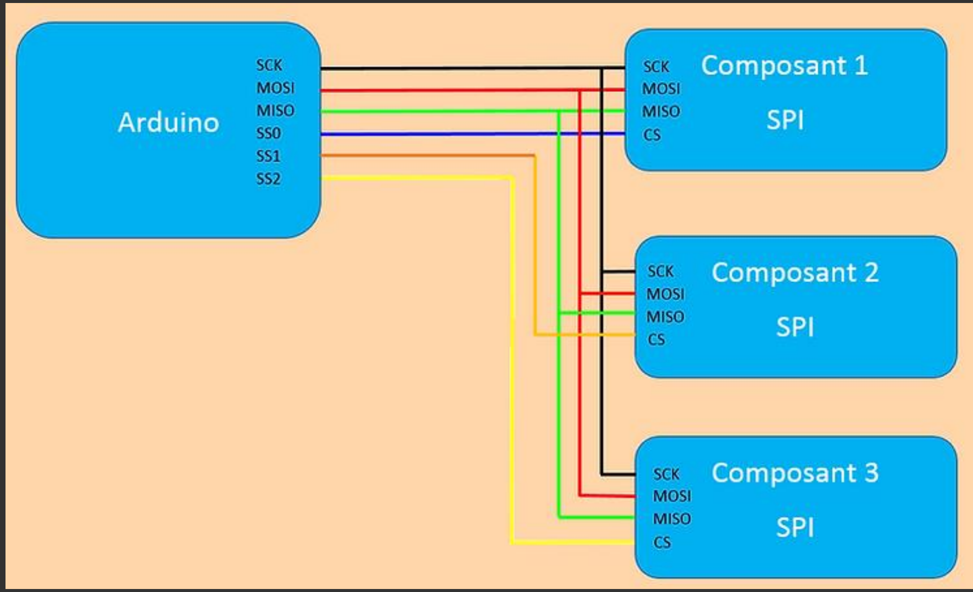
Le bus I2C permet de relier facilement à un microprocesseur à divers circuits intégrés (stockage et l'affichage de données, fonction numériques ou analogiques)



La communication, une liaison série synchrone fonctionne avec trois fils :

- un signal de donnée (SDA).
- un signal d'horloge (SCL).
- la masse (oV).

LIAISON SPI (SERIAL PERIPHERAL INTERFACE)



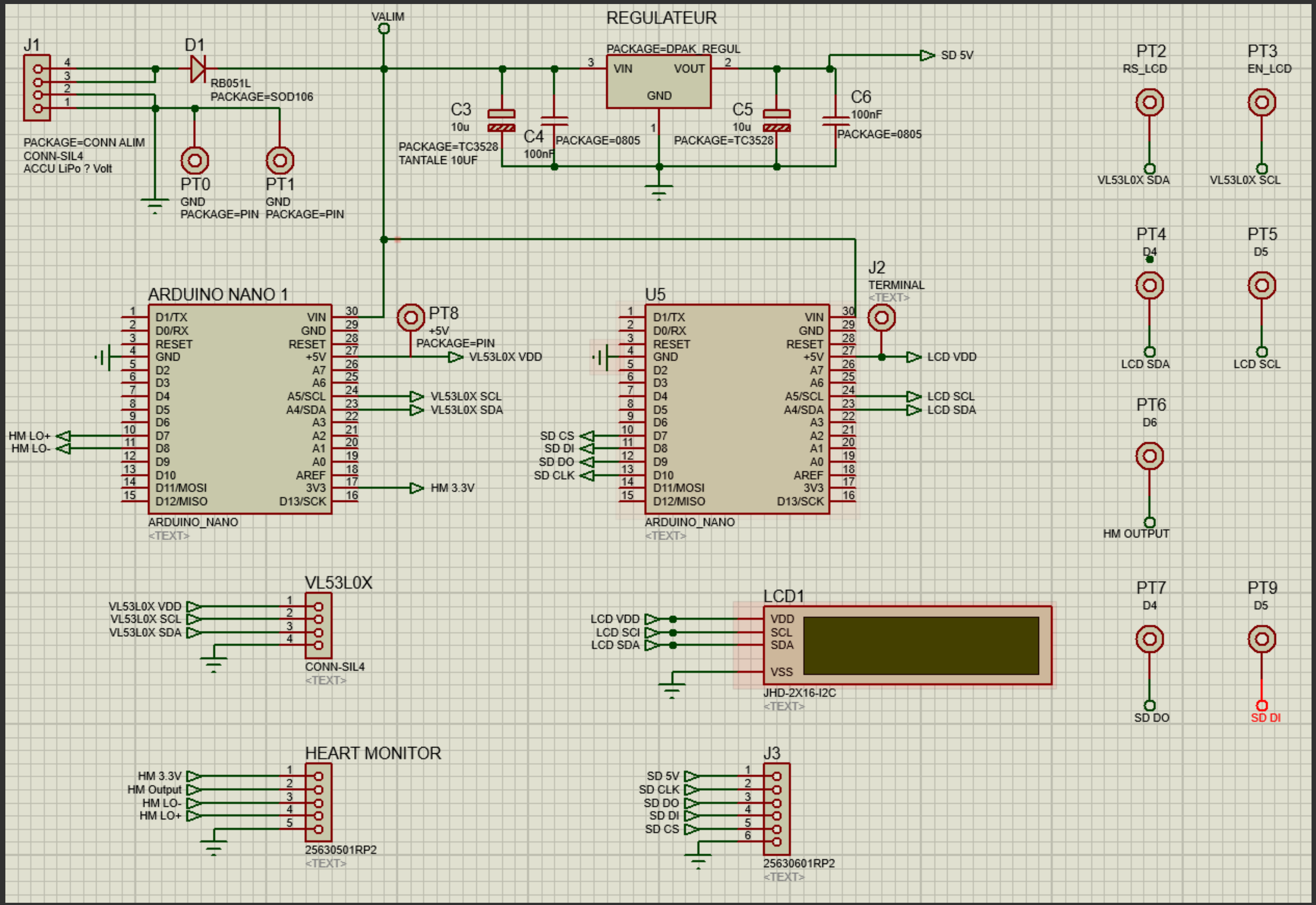
Liaison série synchrone qui fonctionne en full duplex. La communication est réalisée selon un schéma maître-esclaves, où le maître s'occupe totalement de la communication, et plusieurs esclaves peuvent être reliés au même bus .

Le bus SPI utilise quatre signaux logiques : SCK, MOSI, MISO et SS

La communication sur le bus se fait de la manière suivante :

- Le maître sélectionne l'esclave avec lequel il souhaite communiquer en mettant un niveau bas sur la ligne SS correspondante.
- Le maître génère le signal d'horloge en fonction des capacités de l'esclave
- A chaque coup d'horloge, le maître et l'esclave s'échangent un bit sur les lignes MOSI et MISO

PROTEUS



REGULATEUR
VALIM
J1
D1
RB051L
PACKAGE=SOD106
PACKAGE=CONN ALIM
CONN-SIL4
ACCU LiPo ? Volt
PT0
GND
PACKAGE=PIN
PT1
GND
PACKAGE=PIN
PACKAGE=DPAK_REGUL
VIN
VOUT
GND
C3
10u
PACKAGE=TC3528
TANTALE 10UF
C4
100nF
PACKAGE=0805
C5
10u
PACKAGE=TC3528
C6
100nF
PACKAGE=0805
SD 5V
PT2
RS_LCD
VL53L0X SDA
PT3
EN_LCD
VL53L0X SCL
PT4
D4
LCD SDA
PT5
D5
LCD SCL
PT6
D6
HM OUTPUT
PT7
D4
SD DO
PT9
D5
SD DI
ARDUINO NANO 1
ARDUINO_NANO
PT8
+5V
PACKAGE=PIN
VL53L0X VDD
VL53L0X SCL
VL53L0X SDA
HM 3.3V
HM LO+
HM LO-
J2
TERMINAL
U5
LCD VDD
LCD SCL
LCD SDA
SD CS
SD DI
SD DO
SD CLK
VL53L0X
CONN-SIL4
LCD1
VDD
SCL
SDA
VSS
JHD-2X16-I2C
HEART MONITOR
HM 3.3V
HM Output
HM LO-
HM LO+
25630501RP2
J3
SD 5V
SD CLK
SD DO
SD DI
SD CS
25630601RP2

CONCLUSION

- Début de la mise en œuvre et de la création du projet
- Découverte de nouveaux composants, de nouvelles liaisons
- Environnement Arduino, et interface de programmation pour Arduino