Proiect Arduino

programare orientată pe obiecte

Sistem ECG

Studenți:

Bălănescu Adrian-Gabriel,

Bîzdoc Vasile-Gabriel,

Chirap Andrei,

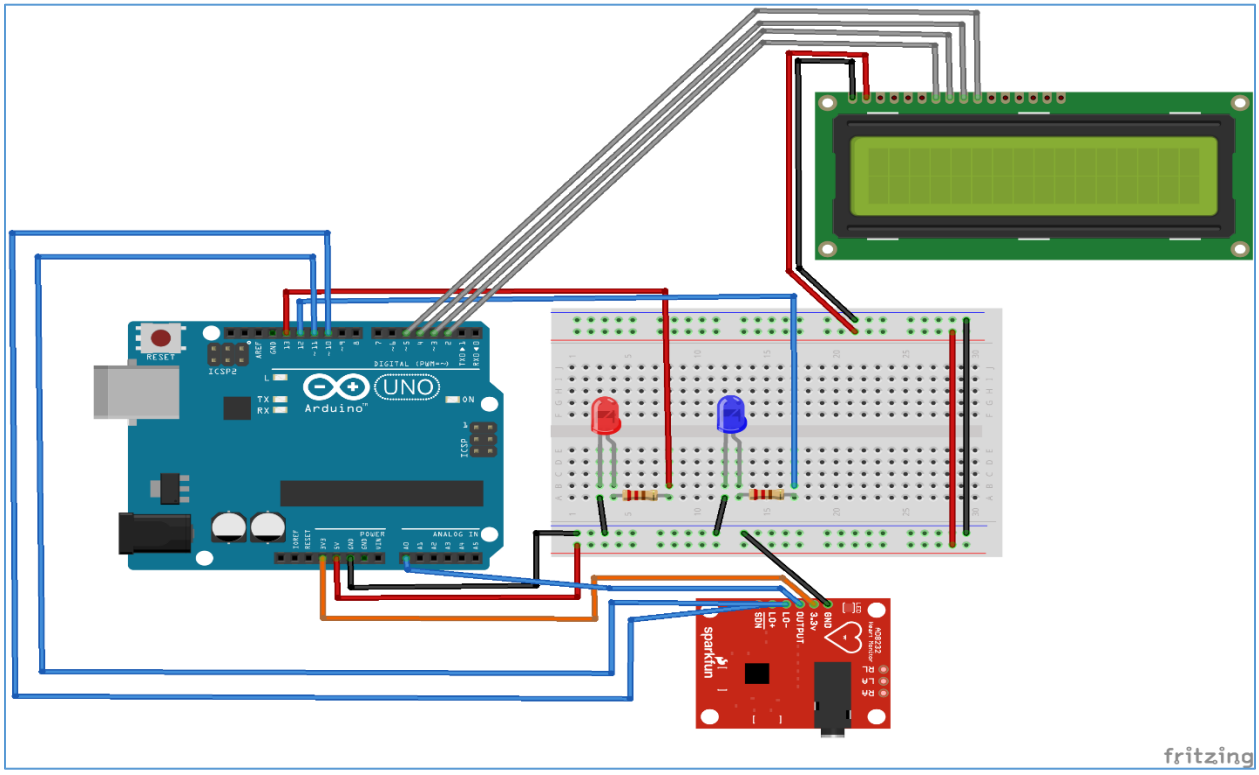
An II, Grupa 2.1

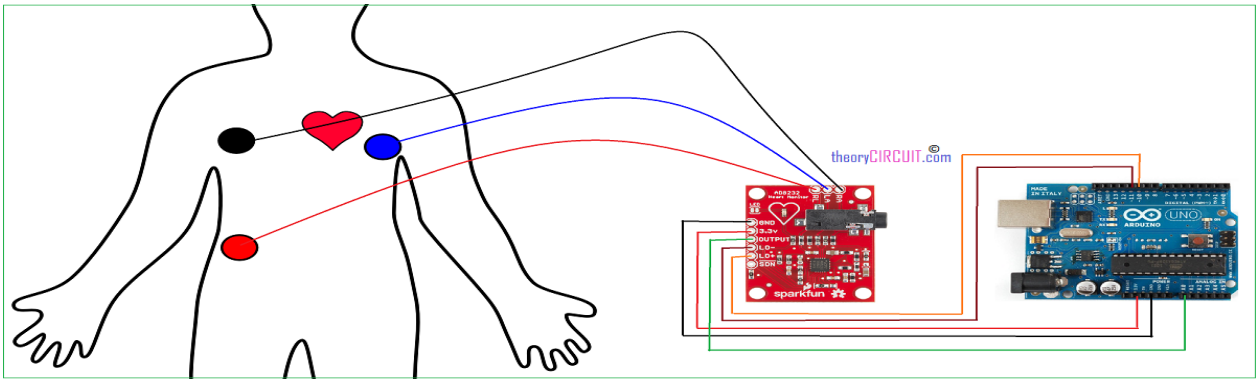
Timișoara, 2017

Cerințe inițiale:

1. Dacă sistemul este oprit se va aprinde ledul roșu, iar dacă este pornit ledul albastru. De asemenea se va afișa pe LCD un mesaj corespunzător.
2. Pe calculator, cu ajutorul programului *Processing* se va putea vizualiza semnalul de ECG, iar pe Serial Monitor se vor afișa valorile.

Diagramă hardware:





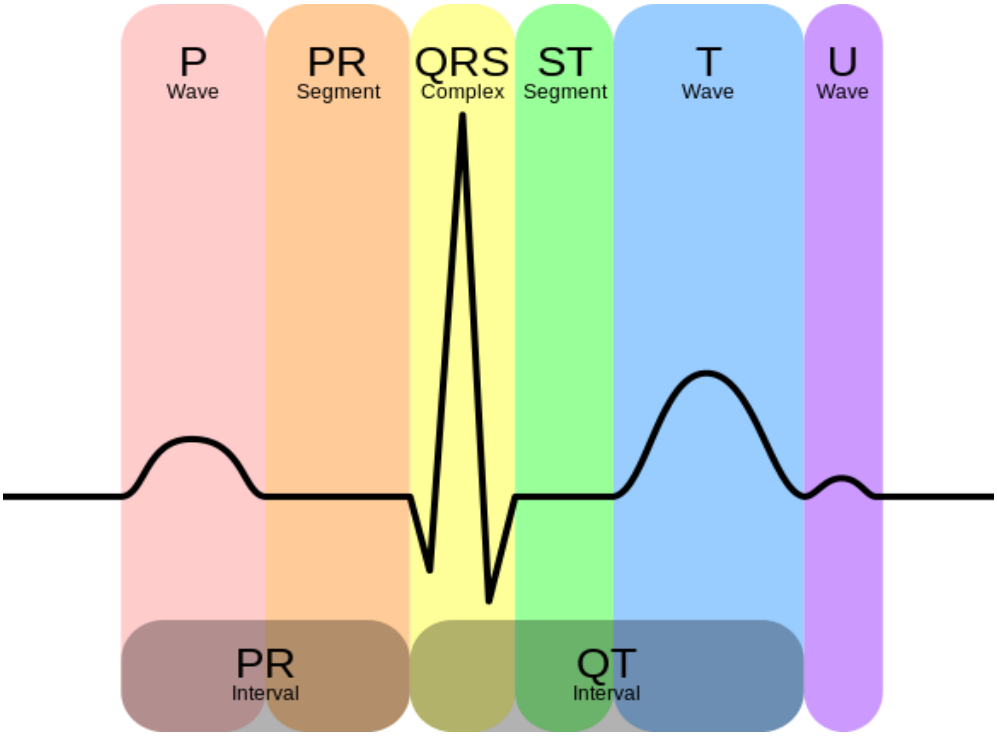
Componente sistem:

1. Placă de dezvoltare Arduino UNO v3. X1
2. Senzor ECG Sparkfun AD8232. X1
3. Led brick albastru. X1
4. Led brick roșu. X1
5. Shield LCD1602. X1

Obiectiv și mod de funcționare:

Acest proiect are ca scop înregistrarea și vizualizarea în timp real al unui semnal ECG (electrocardiogramă).

Semnalul ECG este separat în două intervale de bază – PR – și -QT- reprezentate mai jos:

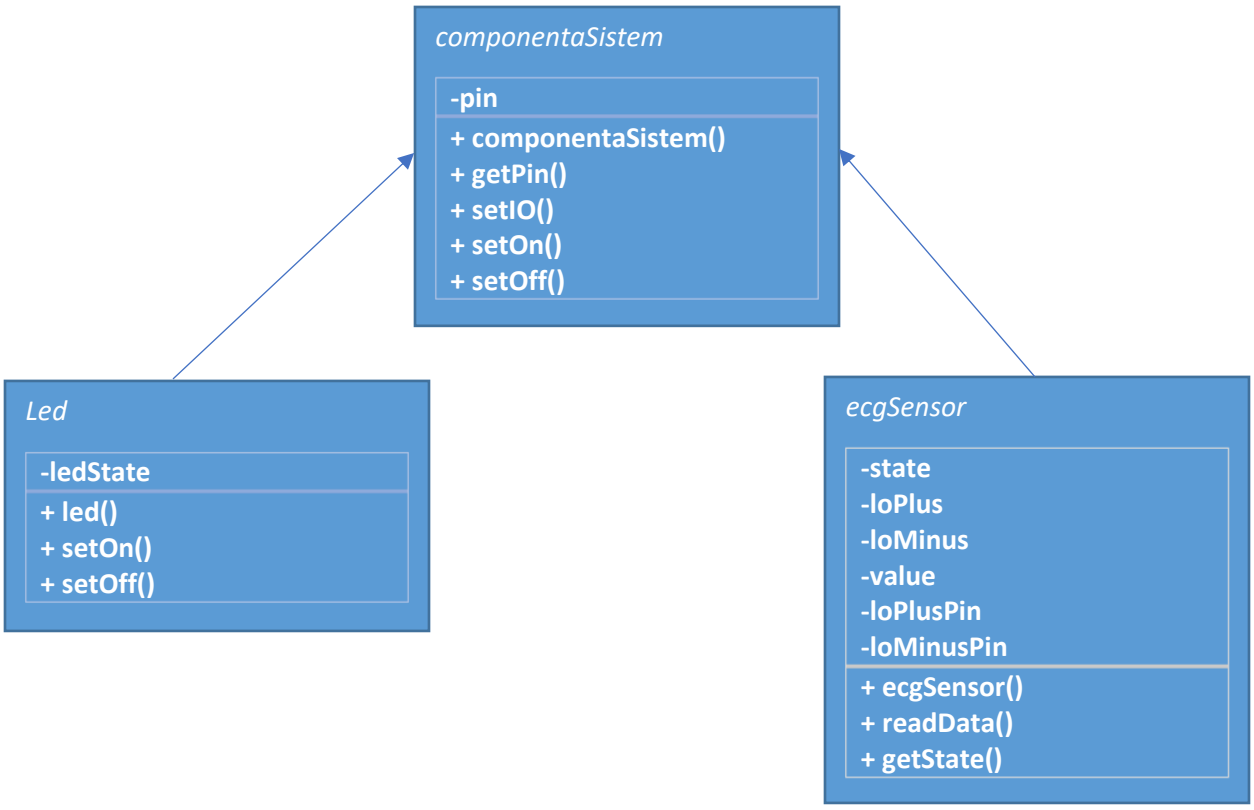


Semnalul este înregistrat sub formă de impulsuri electromagnetice, stabilizate și amplificate de senzorul ECG și este vizualizat prin intermediul unui program în Processing care preia datele de la ieșirea Serială a plăcii Arduino.

Sistemul funcționează astfel:

- La detectarea unui semnal ECG sistemul trece în starea Activ, se aprinde ledul albastru, se afișează pe LCD un mesaj și începe înregistrarea și transmiterea semnalului la interfața serială. (este detectat un semnal ECG atunci cand semnalele de la pinii LO+ și LO- sunt 0, adică atunci când nu există interferențe în semnal sau electrozi deconectați)
- În Processing semnalul este preluat sub forma unui șir de valori de amplitudine care sunt desenate cu ajutorul unor linii.

În figura de mai jos este reprezentată schema UML a softwareului ce preia datele de la senzorul ECG.

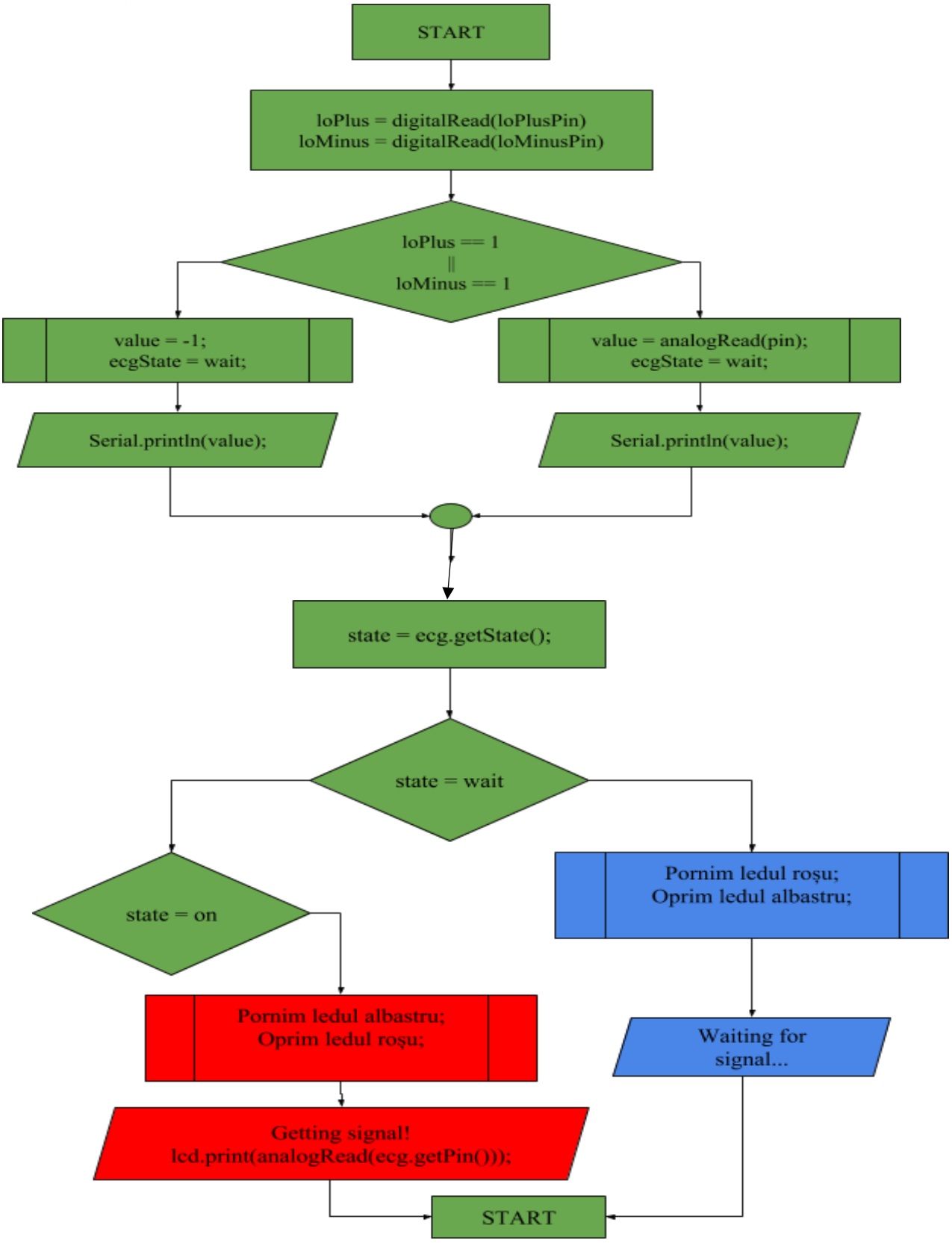


Din schema UML se observă clasa de bază *componentaSistem* care moștenește celelalte clase și conține funcții virtuale redefinite în celelalte clase.

Clasa *led* conține metode pentru pornirea și oprirea unui led.

Clasa *ecgSensor* conține metode pentru configurarea interfeței senzorului.

Schema logică:



Testarea aplicației:

Acțiune	*Răspuns așteptat*	*Răspuns real*
■ conectarea electrozilor în punctele specifice pentru detectarea semnalului ECG	■ aprinderea ledului albastru ■ stingerea ledului roșu ■ începere transmisiune date pe interfața serial ■ afișare mesaj pe LCD	■ Sistemul se comportă conform specificațiilor.
■ Deconectarea electrozilor	■ Aprinderea ledului roșu ■ Stingerea ledului albastru ■ Afișare mesaj pe LCD ■ Transmitere -1 pe interfața serială	■ Sistemul se comportă conform specificațiilor

Concluzii:

Pe parcursul acestui proiect interactiv am învățat să utilizăm diferite site-uri web pentru dezvoltarea tehnicilor de programare orientată pe obiecte (**ex:**https://paulmurraycbr.github.io/ArduinoTheOOWay.html**),** precum și să lucrăm cu plăcile de dezvoltare Arduino.

De asemenea am aprofundat noțiunile din limbajul de programare C++ și le-am concretizat în sursa codului de mai jos.

În final vreau să felicit catedra de Programare orientată obiect a departamentului pentru frumosul exercițiu la care am fost supuși pentru a avea o mai bună înțelegere, aplicabilitate, asimilare și prelucrare concretă de date în vederea obținerii unui rezultat corect, performant și demn de profesia pe care am ales să o urmăm**.**

Cod sursă:

https://github.com/adrianB3/ArduinoECG

Program Arduino:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);

typedef enum {on, off, wait}sysState; //definire stari sistem

class sysComponent {
protected:
	uint8_t pin,IO;
public:
	sysComponent(uint8_t pin) {
		this->pin = pin; //definire pin folosit de componenta
	}

	virtual void setIO(uint8_t IO) {
		this->IO = IO; //definire tip pin intrare/iesire
	}

	virtual uint8_t getPin() {
		return pin; // returnare pin folosit de componenta
	}

	virtual void setOn() {};
	virtual void setOff() {};
};

class led:public sysComponent {
	sysState ledState;
public:
	led(uint8_t pin,sysState ledState) :sysComponent(pin) {
		this->ledState = ledState;
	}

	void setOn() {
		digitalWrite(pin, HIGH); // activarea unui led
		this->ledState = on;
	}

	void setOff() {
		digitalWrite(pin, LOW); // dezactivarea unui led
		this->ledState = off;
	}
};

class ecgSensor : public sysComponent {
	sysState ecgState;
	int value;
	int loPlus, loMinus;
	int loPlusPin, loMinusPin;
public:
	ecgSensor(uint8_t pin, int loPlusPin, int loMinusPin) :sysComponent(pin) {
		this->pin = pin;
		this->loPlusPin = loPlusPin;
		this->loMinusPin = loMinusPin; // definire pini folositi de senzorul ecg

		pinMode(loPlusPin, INPUT);
		pinMode(loMinusPin, INPUT);
	}

	void readData() {
		this->loPlus = digitalRead(loPlusPin);
		this->loMinus = digitalRead(loMinusPin);

		if (loPlus == 1 || loMinus == 1) {
			// daca exista interferente sau un conector este desprins
```

```
                        // este returnata valoarea -1 si sistemul este in asteptare
                        value = -1;
                        Serial.println(value);
                        ecgState = wait;
                }
                else {
                        // se trimit date in interfata seriala spre a fi prelucrate de Processing
                        value = analogRead(pin);
                        Serial.println(int(value));
                        ecgState = on;
                }
        }

        sysState getState() {
                return ecgState; // returneaza starea curenta a senzorului
        }
};

led ledRosu(13,off);
led ledAlbastru(12,off);
ecgSensor ecg(A0, 10, 11);

void setup()
{       //initializari
        Serial.begin(9600); //comunicare la 9600 baud
        lcd.begin(16, 2);
        lcd.clear();
        ledRosu.setIO(OUTPUT);
        ledAlbastru.setIO(OUTPUT);
        ecg.setIO(INPUT);
}

void loop()
{
        sysState state;
        ecg.readData(); // citire date
        state = ecg.getState();
        switch (state) // se efectueaza actiuni in functie de starea in care se afla sistemul
        {
        case wait:
                ledRosu.setOn();
                ledAlbastru.setOff();
                lcd.setCursor(0, 0);
                lcd.print("Waiting for");
                lcd.setCursor(0, 1);
                lcd.print("signal...");
                break;
        case on:
                lcd.setCursor(0, 0);
                lcd.write("Getting signal!");
                lcd.setCursor(0, 1);
                lcd.print(analogRead(ecg.getPin()));
                ledRosu.setOff();
                ledAlbastru.setOn();
                ecg.readData();
                break;
        }
}
```

Program Processing:

```
import processing.serial.*; //librarie folosita pentru a accesa interfata seriala
Serial myPort;
int xPos = 1;
float height_old = 0;
float height_new = 0;
float inByte = 0;
void setup () {
 fullScreen();
 frameRate(190);
 printArray(Serial.list());   // Afisarea porturilor seriale disponibile
 myPort = new Serial(this, Serial.list()[0], 9600); //initializare port
 myPort.bufferUntil('\n'); // asteptare port serial pana la intalnire newline
 background(0); //setarea backgroundului
 grid(); // desenare grid
}
void draw () {
  fill(0);
  stroke(0,255,0);
  rect(0,0,80,30);
  drawText(str(inByte)); // afisare valoare curenta
  String inString = myPort.readStringUntil('\n');
  if (inString != null) {
  inString = trim(inString); // eliminare spatii
  if (xPos >= width) {
    xPos = 0;
    background(0);
    grid(); // daca s-a ajuns la margina ferestrei se reia de la inceput
  }
  else {
    xPos++; // creste pozitia pe axa x
  }
  if (int(inString) == -1) {
    stroke(0, 0, 0xff); // setare culoare albastra (R, G, B)
    inByte = 512;  // middle of the ADC range (Flat Line)
  }
  else if(float(inString)!= -1){
    stroke(255, 150, 150); // setare culoare rosie ( R, G, B)
    inByte = int(inString);
```

```
  }
  float center = height/2;
  inByte = map(int(inByte), 0, 1023, center-200, center+200); // maparea valorilor analogice pentru a incapea in fereastra
  height_new = height - inByte;
  strokeWeight(2); // setare grosime drawer
  line(xPos - 1, height_old, xPos, height_new); // desenare linie pe axa x intre amplitudinea semnalului anterior si curent
  height_old = height_new; // semnalul anterior devine cel curent
 }

}


void grid() {
 // desenare grid
   for (int i=0; i<width; i++) {
     for (int j=0; j<height; j++) {
       strokeWeight(0);
       stroke(0,255,0);
       noFill();
       rect(i*width/16, j*height/16, width/16, height/16);
     }
   }
}
void drawText(String content) {
 //afisare text
  fill(255);
  text(int(content),10,20);
}
```