Ministerul Educației și Cercetării al Republicii Moldova Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică și Microelectronică Departamentul Ingineria Software și Automatică

RAPORT

**Lucrarea de laborator nr. 5**

**Disciplina: Internetul lucrurilor**

A efectuat:

st. gr.TI-215 Josan Artiom

A verificat:

Asistent Universitar Litra Dinu

Chişinău 2024

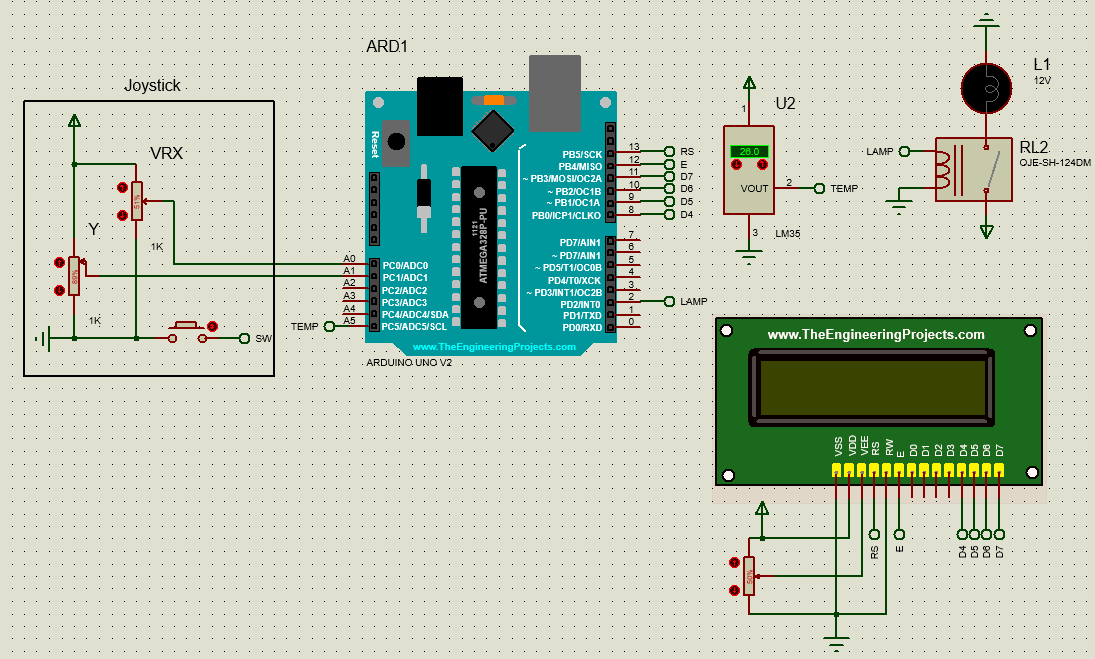
**Descrierea problemei**

Să se realizeze o aplicație in baza de MCU care va implementa sisteme de control pentru:

* + - Control temperatura sau umiditate cu aplicarea metodei de control On-Off cu histerezis cu acționare prin releu.
  + Control turații motor cu aplicarea metodei PID cu un encoder în calitate de senzor, și driver L298 pentru aplicarea puterii la motor.  
    NOTA: in p (b) se poate alege și la alt parametru de control, cu constrângerea ca acționarea va fi cu o rezoluție de min 8 biți.
* Set point (valoarea de referința pentru control) se va seta de la una din surse, la alegere:
  + Un potențiometru;
  + Doua butoane pentru UP/Down;
  + Sensor encoder;
  + Keypad;
  + Interfața serial;
* Valoarea de Setpoint si cea Curenta se vor afișa la LCD.

**Problema 1**

1. **Schema**



**Figura 1.1 – Schema circuitului**

1. **Codul sursă:**

#define THERMOSTAT\_PIN 2

#define VRX\_PIN A0

#define VRY\_PIN A1

#define TEMP\_PIN A5

LiquidCrystal lcd(13, 12, 8, 9, 10, 11);

Joystick \*joystick;

TempSensor tempSensor(TEMP\_PIN);

Relay thermostat(THERMOSTAT\_PIN, false);

uint8\_t minTemp = 25;

uint8\_t lastMinTepm = -1;

uint8\_t lastTemp = -1;

void controlMinTemp();

bool lastThermostatState;

bool shouldThermostatBeActive(float tempC);

void controlThermostat();

void clearLcdLine(int line, int startIndex = 0, int endIndex = 15);

void displayCurrentTemp();

void displayMinTemp();

void controlMinTemp() {

Joystick::Move move = joystick->singleRead();

switch (move)

{

case Joystick::Move::UP:

if (minTemp < 50) {

minTemp++;

}

displayMinTemp();

break;

case Joystick::Move::DOWN:

if(minTemp > 1) {

minTemp--;

}

displayMinTemp();

break;

}

}

bool shouldThermostatBeActive(float tempC) {

if (tempC < minTemp) return true;

return false;

}

void controlThermostat() {

bool thermostatState = shouldThermostatBeActive(tempSensor.getTempCelsius());

if (thermostatState && !lastThermostatState) {

lastThermostatState = true;

thermostat.turnOn();

} else if (!thermostatState && lastThermostatState) {

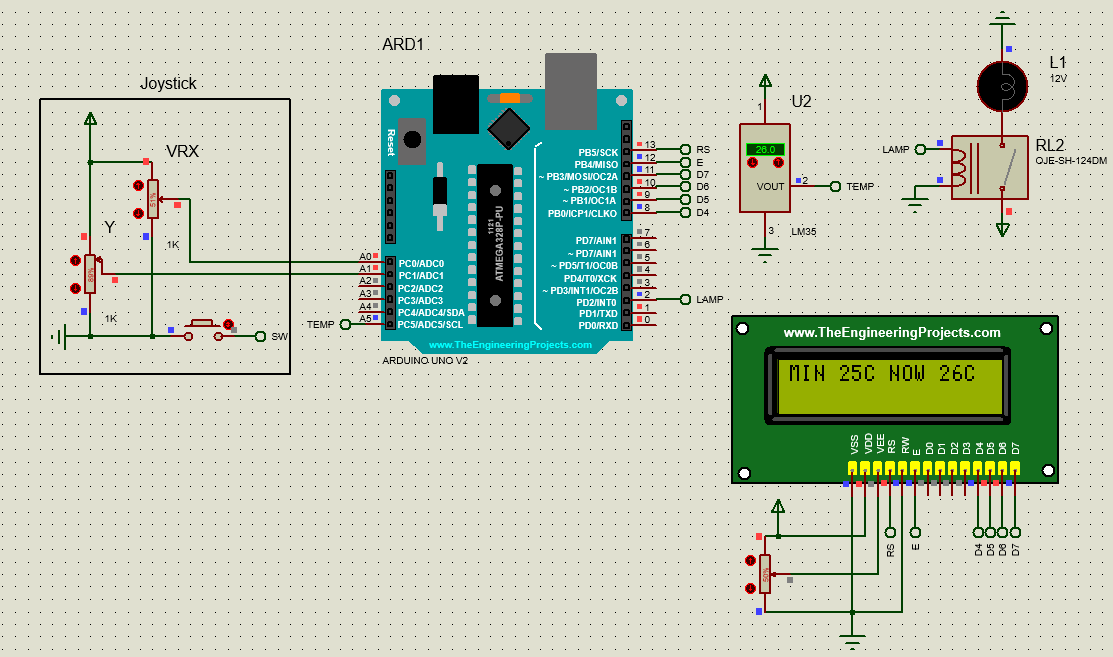
lastThermostatState = false;

thermostat.turnOff();

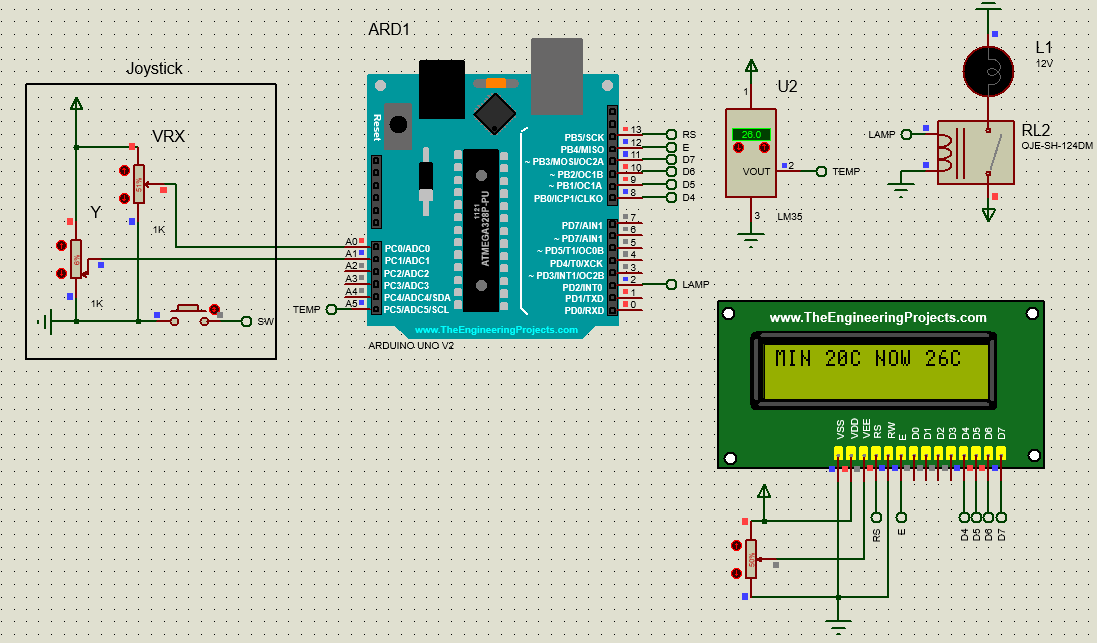
}

}

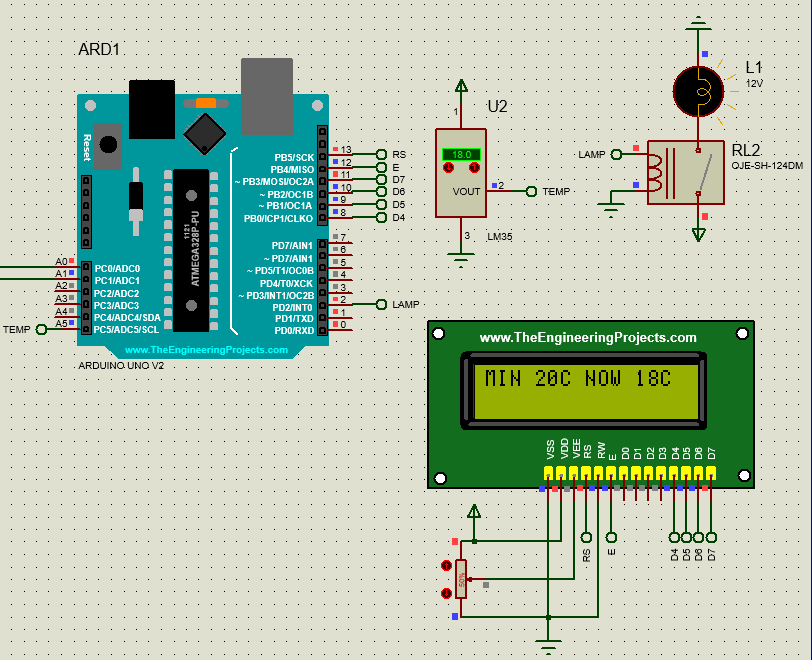
1. **Rezultatul**



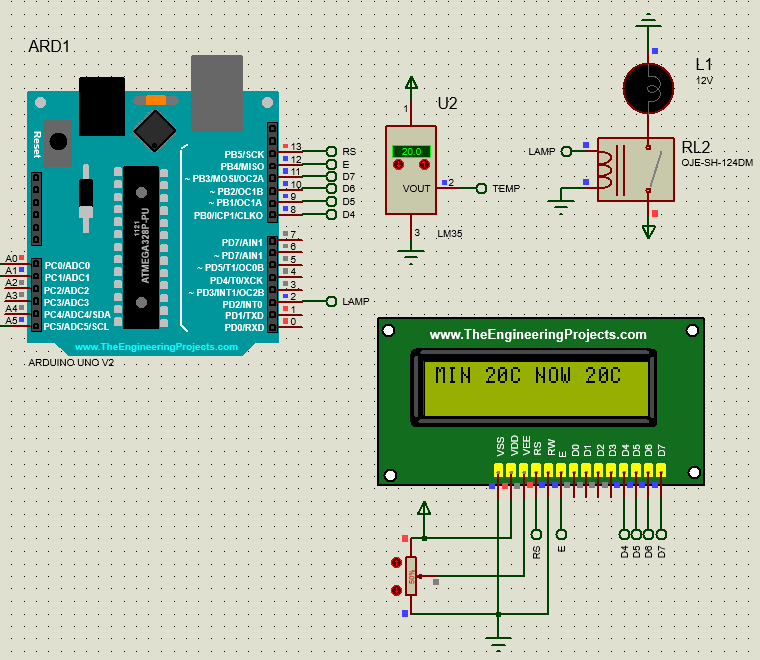
**Figura 1.2 – Stare inițială**



**Figura 1.3 – Modificare temperatură minimă**



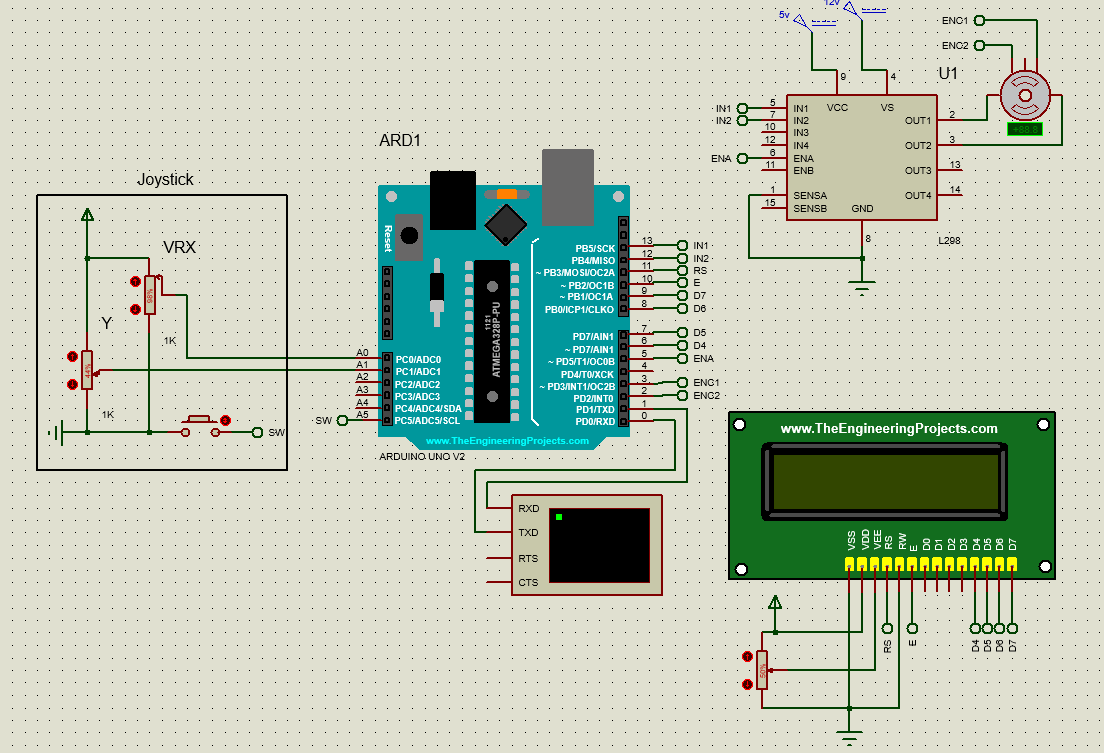
**Figura 1.4 – Temperatură mică, termostat aprins**



**Figura 1.5 – Temperatură normală, termostat stins**

**Problema 2**

1. **Schema**



**Figura 2.1 – Schema circuitului**

1. **Codul sursă:**

#include <Arduino.h>

#include <AxisJoystick.h>

#include <LiquidCrystal.h>

#include "MotorDriver.h"

extern "C"

{

#include "pid.h"

}

#define IN1 13

#define IN2 12

#define ENA1 5

#define ENC1 3

#define ENC2 2

#define PULSES\_PER\_REVOLUTION 24

#define BAUD\_RATE 9600

#define VRX\_PIN A0

#define VRY\_PIN A1

double setpoint = -200;

double kp = 3;

double ki = 0.5;

double kd = 0.5;

double minPwm = 0;

double maxPwm = 255;

pid\_t pid;

volatile double actualSpeed;

LiquidCrystal lcd(11, 10, 6, 7, 8, 9);

Joystick \*joystick;

MotorDriver motor(ENA1, IN1, IN2);

void readEncoder();

double convertSpeedToRPM();

void motorControl();

void setPointControl();

void clearLcdLine(int line, int startIndex = 0, int endIndex = 15);

void displaySetPoint();

void displayMotorStats();

void setPointControl() {

Joystick::Move move = joystick->singleRead();

switch (move)

{

case Joystick::Move::UP:

if (setpoint < 255) {

setpoint += 10;

}

displaySetPoint();

pid\_set\_setpoint(&pid, setpoint);

break;

case Joystick::Move::DOWN:

if(setpoint > -255) {

setpoint -= 10;

}

displaySetPoint();

pid\_set\_setpoint(&pid, setpoint);

break;

case Joystick::Move::RIGHT:

if (setpoint < 255) {

setpoint += 50;

}

displaySetPoint();

pid\_set\_setpoint(&pid, setpoint);

break;

case Joystick::Move::LEFT:

if(setpoint > -255) {

setpoint -= 50;

}

displaySetPoint();

pid\_set\_setpoint(&pid, setpoint);

break;

}

}

void motorControl()

{

double rpmSpeed = speedToRpm(actualSpeed);

double computedSpeed = pid\_compute(&pid, abs(rpmSpeed));

displayMotorStats();

if (setpoint < 0) {

motor.reverse(abs(computedSpeed));

} else if (setpoint > 0) {

motor.forward(computedSpeed);

} else {

motor.stop();

}

}

double speedToRpm(double speed)

{

return speed / PULSES\_PER\_REVOLUTION \* 60.0;

}

attachInterrupt(digitalPinToInterrupt(ENC1), readEncoder, RISING);

pid\_init(&pid, abs(setpoint), kp, ki, kd, minPwm, maxPwm);

void readEncoder()

{

static uint32\_t prevTime = 0;

uint32\_t currentTime = micros();

double deltaTime = ((double)(currentTime - prevTime)) \* 0.000001;

int8\_t direction = (digitalRead(ENC2)) ? 1 : -1;

actualSpeed = direction / deltaTime;

prevTime = currentTime;

}

1. **Rezultatul**

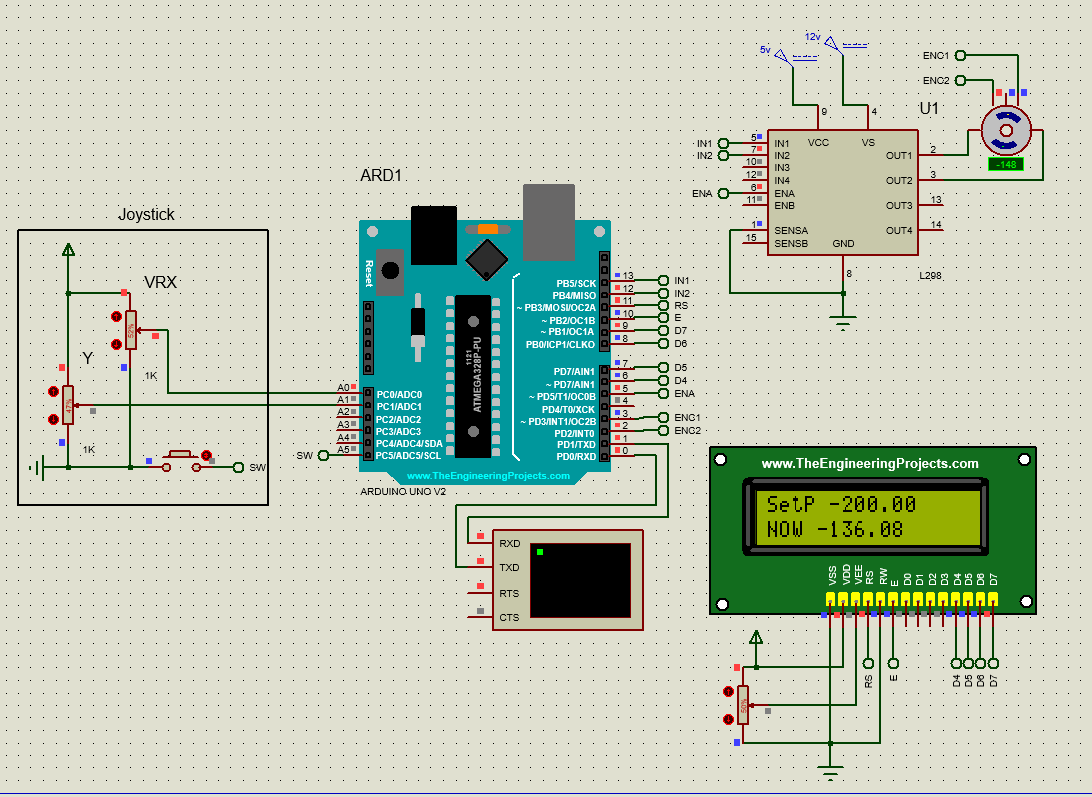


Figura 2.2 – Stare inițială

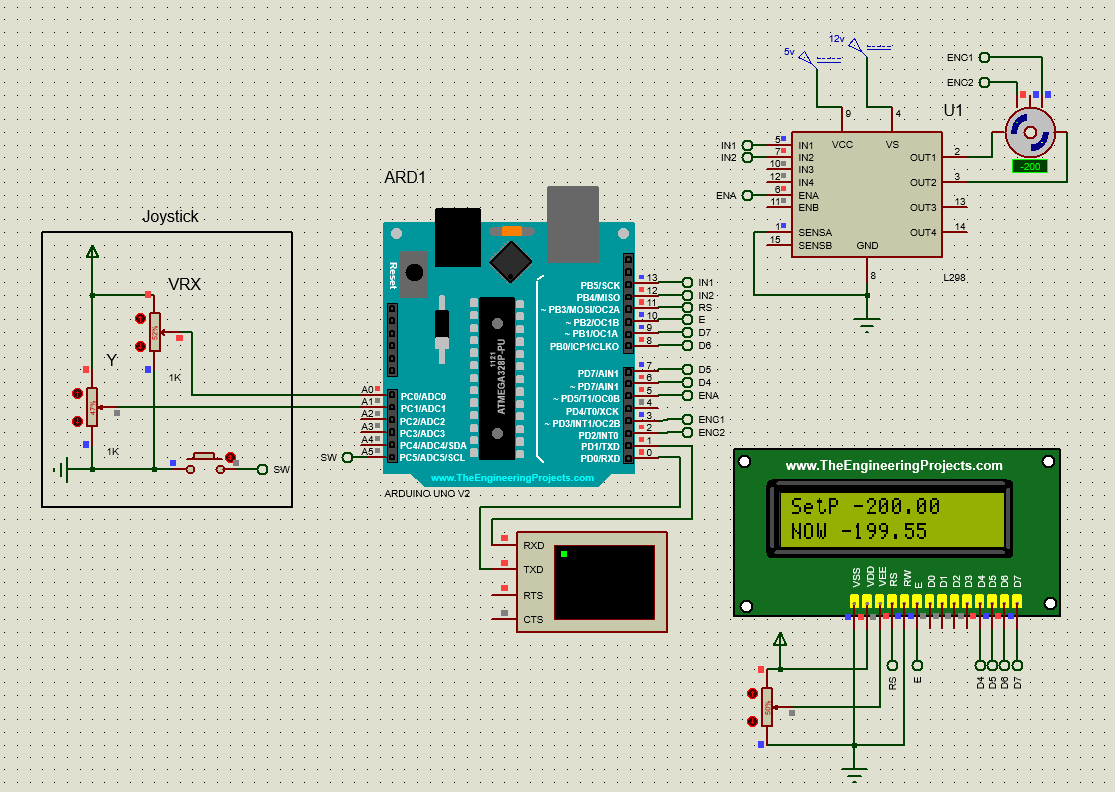


Figura 2.3 – Creșterea vitezei până la setpoint

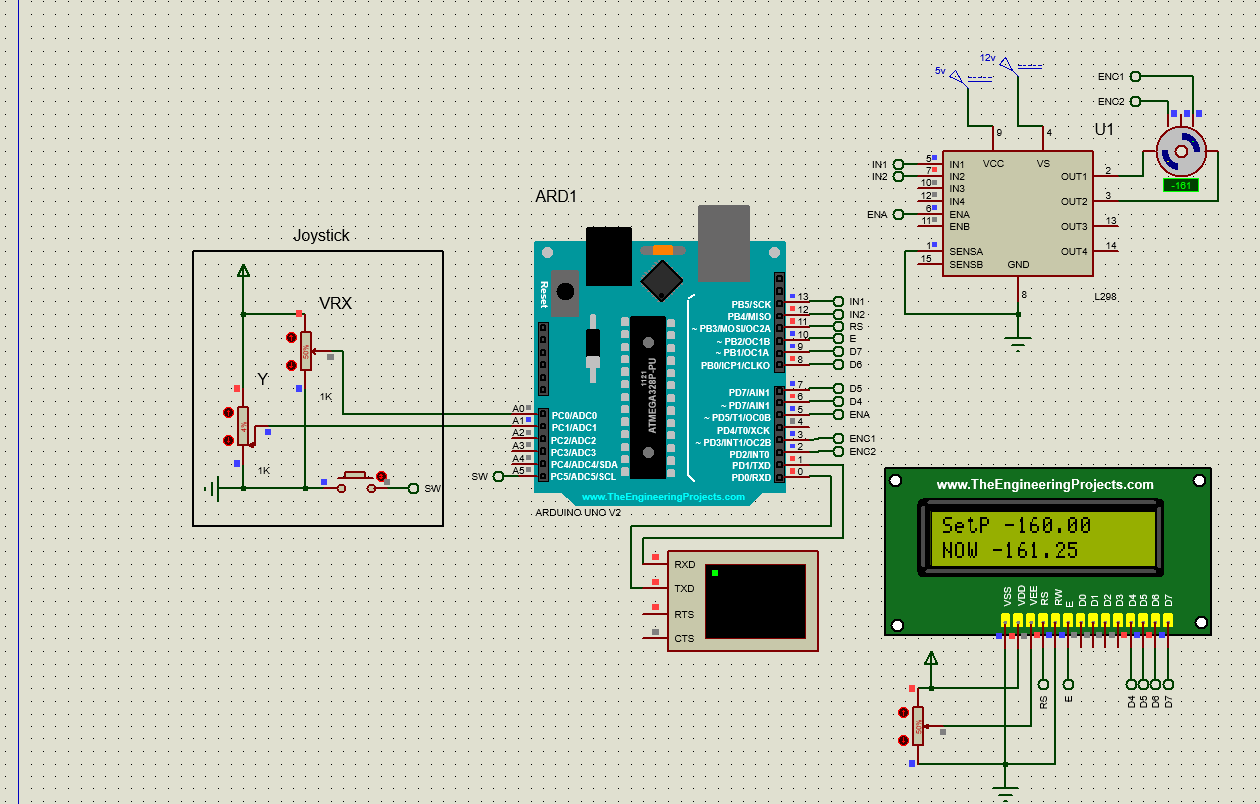
****

Figura 2.4 – Modificarea vitezei

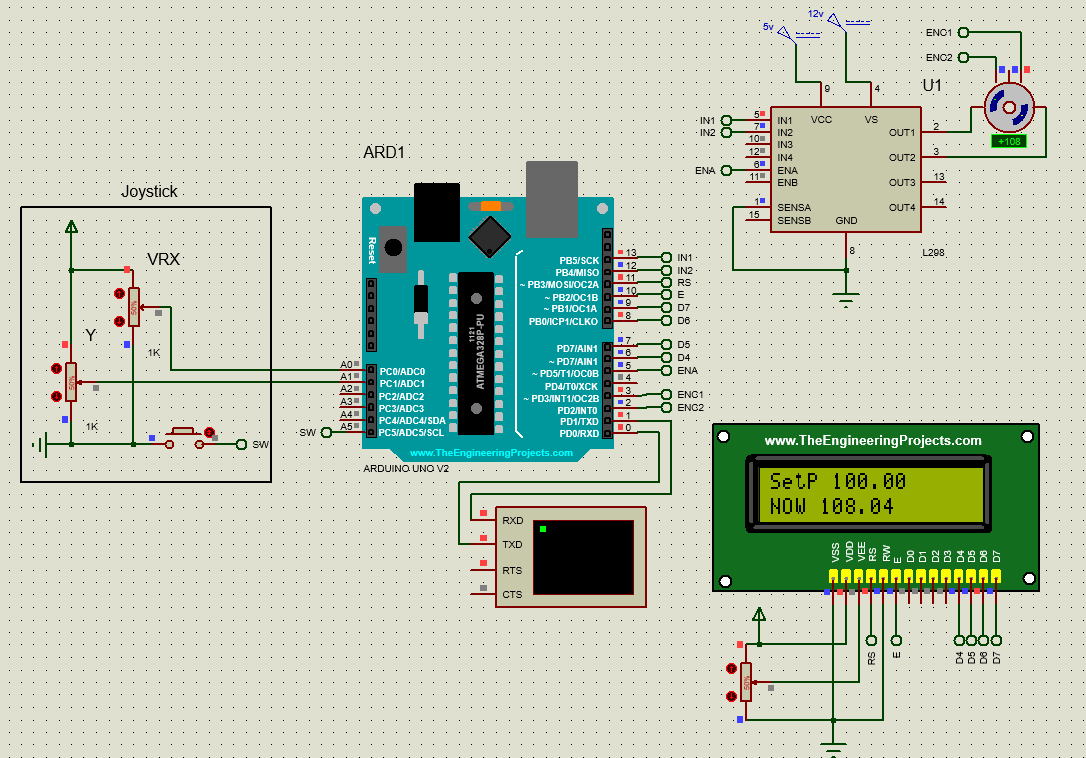


Figura 2.5 – Rotirea motorului în direcția opusă