9th International scientific conference
Technics and Informatics in Education – TIE 2022
16-18 September 2022

Session: *Engineering Education and Practice*
Professional paper
DOI: 10.46793/TIE22.159O

Construction and programming of the platform for spatial imaging with sensors

Vasilije Ojdanić*, Stefan Čubonović, Vojislav Vujičić and Branko Koprivica
Faculty of Technical Sciences, University of Kragujevac, Čačak, Serbia
* ojdanic.vasilije98@gmail.com

Abstract: *In this paper, the construction solution and the method of programming the platform for spatial imaging with sensors are presented. The mechanical construction of the platform is described in detail, as well as the electrical components needed for movement of the sensor carrier in the horizontal plane. Movement is achieved by two stepper motors, and movement control is obtained by connecting the corresponding Arduino and LabVIEW programs. The movement of the carrier from the initial position, through four arbitrary points, until it is placed again in the initial position was realised. The paper provides relevant technical data about the platform, connection diagrams, parts of the program code and accompanying discussion.*

Keywords: *platform spatial imaging with sensors; mechanical construction; electrical components; Arduino; LabVIEW*

1. INTRODUCTION

Spatial imaging with sensors is done by moving into space, in a plane (two-dimensional space) or volume (three-dimensional space), a sensor or a system whose parameters are measured [1, 2]. This imaging can be realised for different purposes, just by replacing a sensor or the measuring system. Depending on the type of sensor, visible or invisible phenomena can be displayed by the human eye. For example, when current flows across a conductor, the only „visible" appearance that indicates the existence of a magnetic field in the vicinity of the conductor is a mechanical force or torque, which can act on a nearby object that is sensitive to the magnetic field. Real image of the magnetic field can be seen by the sensors and this is exactly what is achieved by the application of the platform [1].

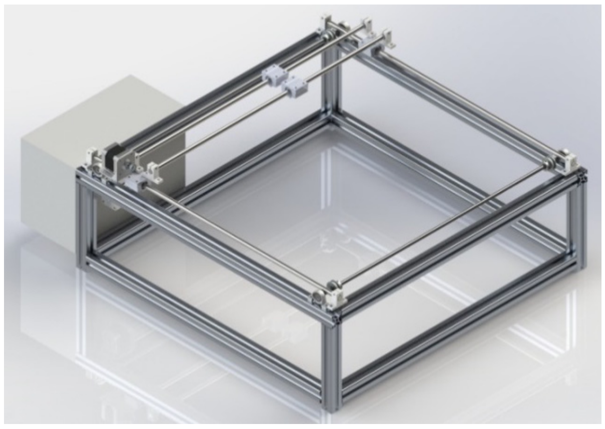
In a similarly way, by using other sensors, other fields, like electric or temperature, can be recorded, or other parameters, dimensions, friction or roughness can be measured.

This paper shows the process of making the platform, its dimensions 700x700x300 mm, mechanical parts, electrical components and parts of the program code for motion control. Also, the paper provides an appropriate discussion of the realized platform and directions for further development.

The platform will be used for teaching at master's studies at FTN in Čačak in the subjects Virtual Instrumentation (VI) and Electrical Measurements of Non-Electric Quantities (EMNEQ), as well as for research that will be carried out by students and teachers of the Faculty in the future.

2. PLATFORM MODEL

Before beginning of mounting mechanics and electrical components, it is necessary to create a model in the 3D CAD design software. A model of a platform for spatial imaging with sensors is created using SolidWorks software and is shown in Figure 1. This is necessary to do before the actual creation of the platform, in order to achieve the desired form on the model itself, through its development, and to spot potential errors that may affect the functionality of the platform in time.



The next step, after developing the model, is to form a list of required parts, which (for the realized platform) are given in Table 1. The table shows the component name and quantity.

A platform is designed to move in the horizontal plane (in two coordinate axes, *X* and *Y*).

3. PARTS OF THE PLATFORM

Table 1 shows the basic mechanical and electrical parts used to build a platform.

Table 1. *Specification of used mechanical and electrical parts*

Name of component	Quantity/ Length
1. Arduino MEGA	1pcs
2. RAMPS 1.4	1 pcs
3. Structural profile 20x20 mm	9 m
4. 10mm aluminium round bar	4 m
5. Aluminium tile	2 pcs
6. Stepper motor	2 pcs
7. 3M Timing belt pulley, tooth pitch 3 mm, width 11 mm/ 16 mm for stepper motor	6 pcs
8. Screw M5	15 pcs
9. Flat washer for screw M5	30 pcs
10. T-slot nut sliding block swivel-in with spring Slot 8 - Type B - M5	100 pcs
11. Mounting enclosure (400x300 mm)	1 pcs
12. Power supply 12V DC (Power supply has huge capacity (30 A) for future upgrades)	1 pcs
13. Timing belt (width 10 mm, pitch 3 mm)	1 pcs
14. Corner connector for profile mounting	25 pcs
15. Optical sensor	2 pcs
16. USB cable, B type	1 pcs
17. Motor driver	2 pcs

To assemble the mentioned parts so that they form a whole that will perform the desired task, it is necessary to use the following tools and measuring instruments:

- electric drill,
- electric grinder,
- screwdriver set,
- hex key set,
- hand saw for metal,
- tape measure and
- digital multimeter.

The supporting structure of the platform is made of aluminum structural profiles 20x20 mm, connected with corner connectors and screws. Mechanical energy is transmitted via timing belts, while stepper motors are used as actuators. Used stepper motors have a two-phase winding and middle derivative leads on the phases. A middle leads are not connected anywhere, because in this way the entire winding of a motor is covered. Limit switches are pass-through optical sensors with three ends. A sensor consists of a photodiode as a transmitter and a phototransistor as a receiver. When a piece of opaque plastic cuts the beam of light, a voltage signal of 5 V is obtained at the ends of the sensor.

After mounting of the mechanical part of the platform, it is necessary to connect the electrical parts in a mounting enclosure according to the wiring diagram shown in Figure 2.

The appearance of the inside of a mounting enclosure after connecting the electrical components is shown in Figure 3. It can be noticed that most of a mounting enclosure is empty, which was the intention, with the aim of later upgrading the platform as needed, that is, the addition of electrical components.

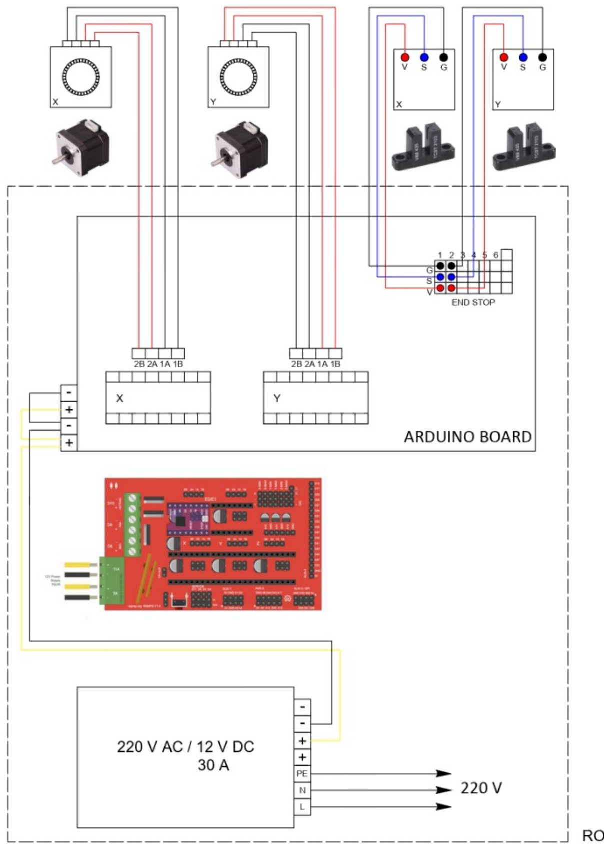


Figure 2. *Wiring diagram of electrical parts*

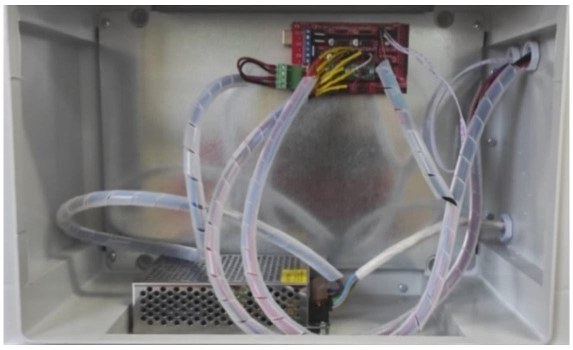


Figure 3. *Inside a mounting enclosure*

To start the stepper motors, the appropriate drivers are needed, which are connected to the RAMPS 1.4 (an additional board for working with motors and sensors), which is previously connected to the Arduino MEGA control board via the appropriate array of pins. The RAMPS 1.4 development board offers advantages when used with already made libraries for use with 3D printers or mini CNC machines. For this platform, the use of already made libraries makes only starting the machine easier, but not its further development. Generally, this project does not require RAMPS 1.4, but the motor drivers themselves require an external power supply which is easiest to connect to the Arduino MEGA board, and the board then routes that power to all the drivers placed on the RAMPS 1.4 board.

Figure 4 shows the stepper motor driver, with the connection ends marked and the connection to the controller and motor.

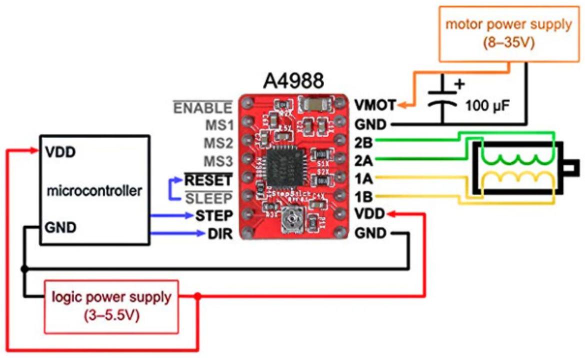


Figure 4. *Driver connection diagram with motor, power supply and* microcontroller

To start the motor, it is necessary to transfer three signals from the Arduino board:

- the first is the ENABLE signal which turns on the driver,
- the second is the DIRECTION signal that defines the direction of movement and
- the third is the STEP signal (an impulse that directly causes the motor to step).

So the platform has two stepper motors and therefore two drivers. It is necessary to determine the six signals that are sent from the Arduino board to the RAMPS 1.4. Picture 5 shows the RAMPS 1.4 with drivers, in a mounting enclosure of the platform.

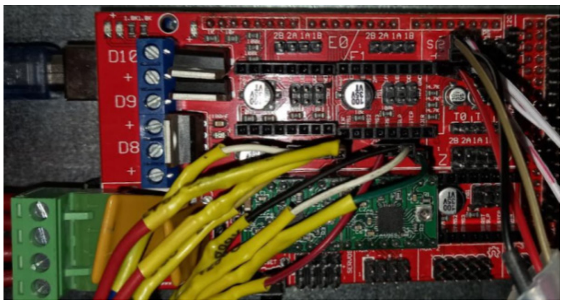


Figure 4. *Layout of the RAMPS 1.4 board in the machine cabinet*

In the cabinet of the machine, the drivers are placed on the bottom two RAMPS 1.4 connectors. It is necessary to remove the drivers from the RAMPS 1.4, and remove the RAMPS 1.4 from the Arduino board in order to examine with a multimeter the connection ends to which the drivers are connected and their connection to the Arduino board. When the RAMPS 1.4 top and bottom connections are found, the pins on the Arduino that the driver is connected to are found. Figure 6 shows the known electrical connection diagram obtained after measuring with a multimeter. Knowledge of these connections is necessary for writing the program code, because it initially defines the connections on which the necessary electrical signals are generated or measured.

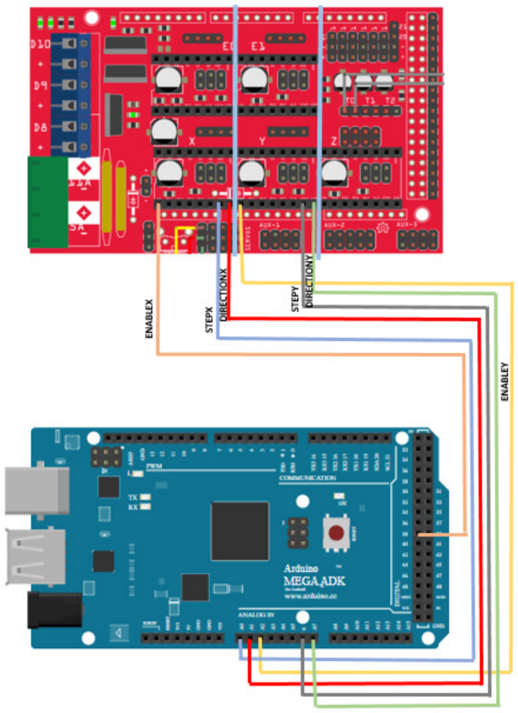
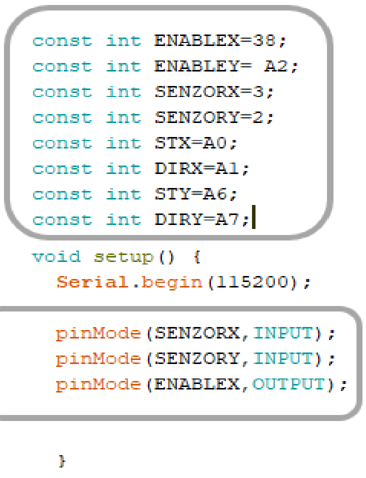


Figure 5. *Electrical connection and connections of the Arduino used by RAMPS 1.4*

4. PLATFORM PROGRAMMING

The next step in creating a platform is programming the platform itself. The goal was to connect the platform to a computer with the LabVIEW program (LV [5]) in order to give movement commands via the computer, from the LV program. Considering the advantages of the RAMPS 1.4 board, it was decided to control the operation of the motor using the Arduino program [6], which is written on the Arduino MEGA board, and to set only the desired dimensions and positions in movement that the platform should achieve from the LV program. The connection for sending data is realized by serial communication via the USB port, using the VISA driver for serial communication in LV. It is also necessary to manage the drivers at the lowest possible level of programming, which enables further development of the platform.

The first step in programming is to assign digital and analog inputs and outputs, Figure 7.



```
const int ENABLEX=38;
const int ENABLEY= A2;
const int SENZORX=3;
const int SENZORY=2;
const int STX=A0;
const int DIRX=A1;
const int STY=A6;
const int DIRY=A7;
void setup() {
  Serial.begin(115200);

  pinMode(SENZORX,INPUT);
  pinMode(SENZORY,INPUT);
  pinMode(ENABLEX,OUTPUT);


}
```

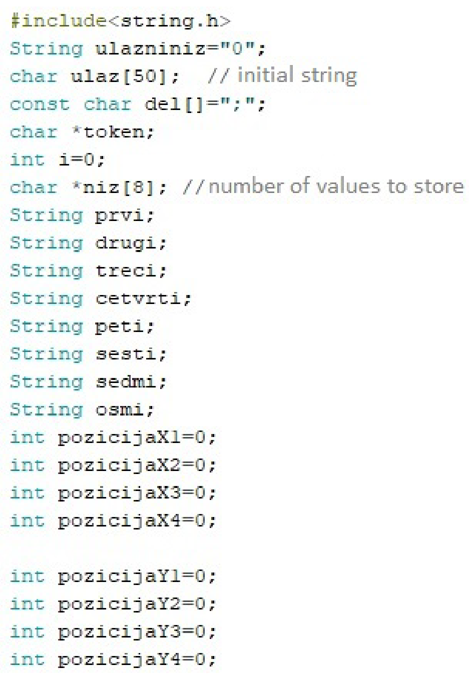
Figure 6. *Assigning inputs and outputs to the microcontroller*

As can be seen in Figure 7, it is necessary to define in the program, before the void setup loop, all the names of the variables to which analog and digital inputs and outputs are assigned.

The first part of the machine's main program is the HOME function, Figure 8. Given that the stepper motors were used, which do not provide information about the current position of the motor (unlike servo motors), it was necessary to define their initial position. The HOME process itself is performed by starting the motors in the direction of the coordinate origin, where the two optical sensors are located. They will move independently of each other until their corresponding sensor is activated, the sensor is a condition for stopping the motor.

The second part of the code is related to the LV program, which was created for the needs of the platform. The Arduino program allows data to be sent and read on the computer via USB. During the development of the platform, it is planned to realize the movement of the sensor carrier from the initial position, through four arbitrary points, until it is placed back in the initial position. Assignment of points would be done from the LV program. In order to achieve this, it is necessary to load the coordinates of those points from the LV program. The easiest way to transfer information from LV is to send the information as a string, Figure 9.

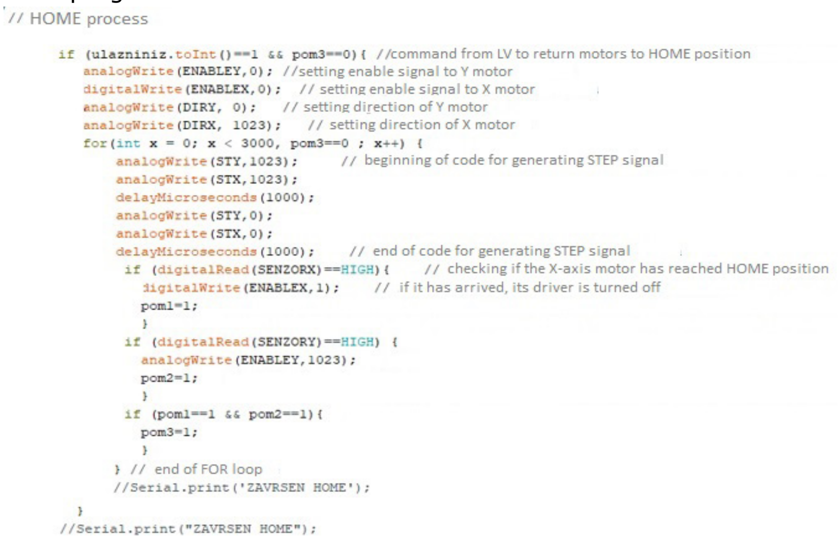
The written function reads the incoming string, divides it into a certain number of parts and finally converts those smaller strings into Int data type (Int is the only data type that can be used to specify the number of steps the motors should take).



```
#include<string.h>
String ulazniniz="0";
char ulaz[50];   // initial string
const char del[]=";";
char *token;
int i=0;
char *niz[8];  //number of values to store
String prvi;
String drugi;
String treci;
String cetvrti;
String peti;
String sesti;
String sedmi;
String osmi;
int pozicijaX1=0;
int pozicijaX2=0;
int pozicijaX3=0;
int pozicijaX4=0;

int pozicijaY1=0;
int pozicijaY2=0;
int pozicijaY3=0;
int pozicijaY4=0;
```

Figure 7. *Calling and defining a string in a program*

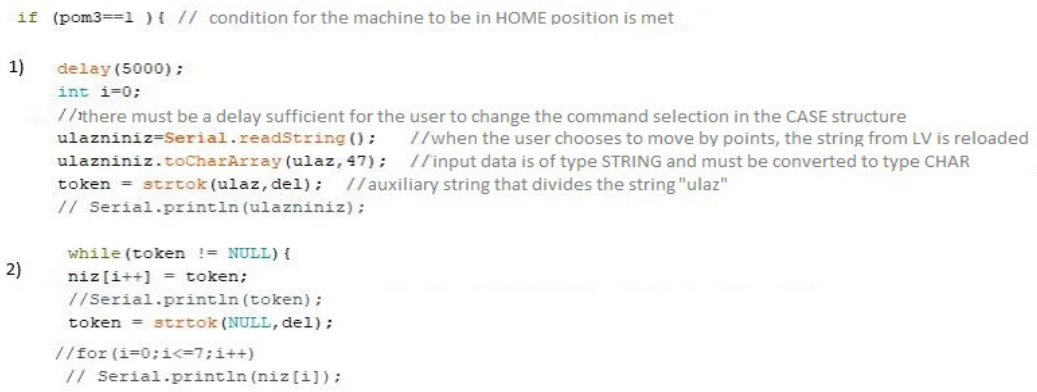


```
// HOME process

 if (ulazniniz.toInt()==1 && pom3==0){ //command from LV to return motors to HOME position
   analogWrite(ENABLEY,0); //setting enable signal to Y motor
   digitalWrite(ENABLEX,0);  // setting enable signal to X motor
   analogWrite(DIRY, 0);   // setting direction of Y motor
   analogWrite(DIRX, 1023);   // setting direction of X motor
   for(int x = 0; x < 3000, pom3==0 ; x++) {
       analogWrite(STY,1023);     // beginning of code for generating STEP signal
       analogWrite(STX,1023);
       delayMicroseconds(1000);
       analogWrite(STY,0);
       analogWrite(STX,0);
       delayMicroseconds(1000);    // end of code for generating STEP signal
        if (digitalRead(SENZORX)==HIGH){     // checking if the X-axis motor has reached HOME position
          digitalWrite(ENABLEX,1);    //  if it has arrived, its driver is turned off
          pom1=1;
          }
        if (digitalRead(SENZORY)==HIGH) {
          analogWrite(ENABLEY,1023);
          pom2=1;
          }
        if (pom1==1 && pom2==1){
          pom3=1;
          }
       } // end of FOR loop
       //Serial.print('ZAVRSEN HOME');

  }
 //Serial.print("ZAVRSEN HOME");
```

Figure 8. *Part of the program code for returning the platform to its initial position*



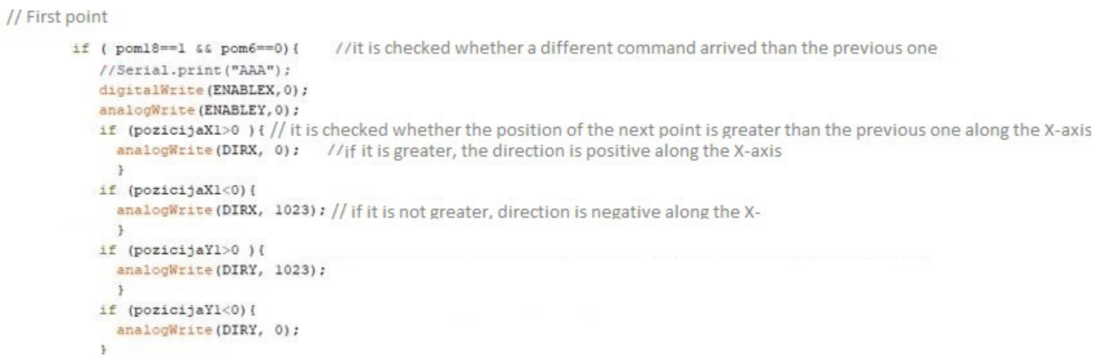
```
 if (pom3==1 ){ // condition for the machine to be in HOME position is met

1)    delay(5000);
      int i=0;
      //there must be a delay sufficient for the user to change the command selection in the CASE structure
      ulazniniz=Serial.readString();    //when the user chooses to move by points, the string from LV is reloaded
      ulazniniz.toCharArray(ulaz,47);   //input data is of type STRING and must be converted to type CHAR
      token = strtok(ulaz,del);    //auxiliary string that divides the string "ulaz"
      // Serial.println(ulazniniz);

       while(token != NULL){
2)     niz[i++] = token;
       //Serial.println(token);
       token = strtok(NULL,del);

      //for(i=0;i<=7;i++)
       // Serial.println(niz[i]);
```

Figure 9. *Splitting an input string into multiple parts*



```
// First point

     if ( pom18==1 && pom6==0){     //it is checked whether a different command arrived than the previous one
       //Serial.print("AAA");
       digitalWrite(ENABLEX,0);
       analogWrite(ENABLEY,0);
       if (pozicijaX1>0 ){ // it is checked whether the position of the next point is greater than the previous one along the X-axis
         analogWrite(DIRX, 0);    //if it is greater, the direction is positive along the X-axis
         }
       if (pozicijaX1<0){
         analogWrite(DIRX, 1023); // if it is not greater, direction is negative along the X-
         }
       if (pozicijaY1>0 ){
         analogWrite(DIRY, 1023);
         }
       if (pozicijaY1<0){
         analogWrite(DIRY, 0);
       }
```

Figure 10. *Part of the function to move to the first point*

The variable String input string is the data that carries information about the coordinates of four points. Figure 10 shows the function that extracts data for each coordinate point. At the end of the function, the data related to the points is written into new variables that will later be used for movement.

The third part of the program is related to movement from point to point. Motion functions can start executing immediately after entering the coordinates. Figure 11 shows part of the code for moving to the first point. There are three main conditions that determine how many steps each motor should take, in which direction each of them can move, and that after reaching the first point, it can move to the second point in exactly the same way as for the first. The Arduino microcontroller executes this code cyclically. That is why certain variables are used to lock re-movement to the first point (or a subsequent one), if this has already been done previously.

5. LABVIEW PROGRAM

The LabVIEW Program was created that can receive and send data to the Arduino via the USB port. In order for communication to be possible, it is necessary to use VISA drivers. Figure 12 shows the front panel used to run the platform. In the LV program, the port used (number 1) and the communication speed must be set, which in the Arduino program must be the same as on the front panel (number 2). It is important that these two speeds are the same, otherwise communication will not be possible.

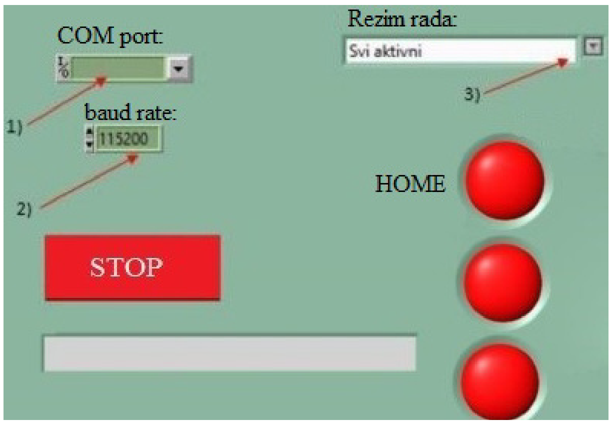


Figure 11. *LV display of front panel*

In the operating mode window (number 3), it is possible to choose several modes of movement on the machine. The first mode does not perform any movement, the second mode returns the machine to the starting position, and the third and fourth perform platform movement along four-point paths.

The block diagram in Figure 13 shows the use of the VISA driver. The Visa Write function (number 5) is used to send data via USB. This function sends data of type String. The field in which the String with point data (number 4) is entered is formatted as follows **X1**; **Y1**; **X2**; **Y2**; **X3**; **Y3**; **X4**; **Y4** with data for all 4 points in the Cartesian coordinate system.

The Visa Read function (number 6) reads the data sent from the Arduino, which is then displayed on the front panel of the LV program.

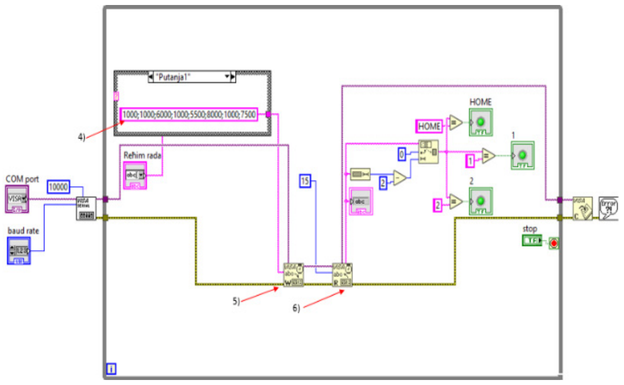


Figure 12. *Layout of the block diagram in the LV program*

6. FINAL APPEARANCE OF THE PLATFORM AND THE POSSIBILITY OF FURTHER IMPROVEMENT

Figure 13 shows the looks of the platform after construction.

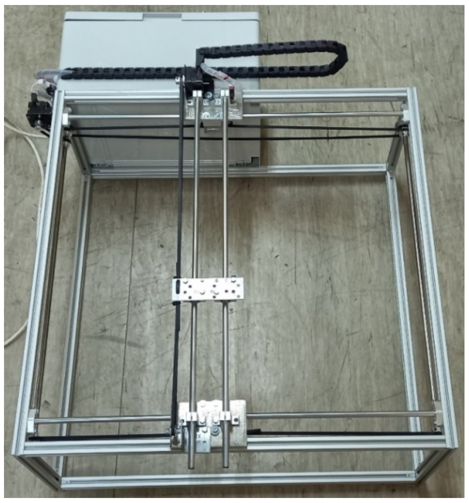


Figure 13. *The final looks of the platform*

The platform has been designed in such a way that numerous opportunities for improvement are left, and they are:

- placement of the sensor on the support and spatial recording on the given surface,
- mechanical finishing (if necessary),
- Improvement of the program for the Arduino board,
- simultaneous motor operation (currently, it is not possible to operate both motors at the same time in order to achieve movement along a given line, but first the operation of one motor to the given position along the corresponding axis is executed, and then the operation of the second motor along the other axis),
- improvement of the LV program so that the mapping of the given surface of an arbitrary shape is performed and the movement management and setting of the movement parameters (speed, resolution, etc.) is done from the LV program and
- speeding up of communication between the platform and the computer.

7. CONCLUSION

This paper shows the realization (construction and programming) of a platform for spatial imaging with sensors. The structural model of the platform implemented in the SolidWorks program is shown and the used mechanical and electrical parts are listed. The platform is made so that the sensor carrier moves in the horizontal plane (in two axes). The movement is performed using two stepper motors, and the Arduino MEGA board with the RAMPS 1.4 additional board and a personal computer, as well as the corresponding specially created Arduino and LabVIEW programs, were used to control the movement. During the testing of the platform, the movement of the carrier from the initial position, through four arbitrarily set points, to the return to the initial position was realized.

At the end text, the final looks of the platform is shown and possibilities for its improvement are listed. The next step in the development of the platform is to place the sensors on the support and test the operation of the entire system. This can be an assignment for students who will attend the VI and EMNEQ subjects in the master's studies.

ACKNOWLEDGMENT

This study was supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia, and these results are parts of the Grant No. 451-03-68/2022-14/200132 with University of Kragujevac – Faculty of Technical Sciences in Čačak.

LITERATURE

[1] Datasheet MMS-1A-RS/MMS-1X-RS,SENIS AG, Baar, Switzerland, May 2022.
[2] Domajnko J, Milanovič M. & Prosen N. (2022). 3D Platform for Coupling Coefficient Evaluation of an Inductive Power Transfer Systems. *Sensors, 22*(4), 1445.doi:10.3390/s22041445
[3] Product Reference Manual Arduino® MEGA 2560 Rev3, Arduino S.r.l., Monza, Italy, July, 2022.
[4] RAMPS 1.4 - https://reprap.org/wiki/RAMPS_1.4
[5] Milovanović, A., Bjekić, M., Koprivica, B. (2010). *Virtuelna instrumentacija*, Tehnički fakultet Čačak.
[6] Arduino IDE, Arduino S.r.l., Monza, Italy https://www.arduino.cc/en/software.