Redesigning of the gripping system on KUKA KR5 by utilizing 3 dimension printing technologies and Arduino

Viktor Varga[1], *Timotei István* Erdei[1,*], and *Géza* Husi[1]

[1]University of Debrecen, Mechatronics Department, 4028 Ótemető street 2. Debrecen, Hungary

Abstract. During this short summary of the project, we'd like to go through all the steps of the project, from the very first planning stages, up to the 3D modelling, and finishing up with the actual printing itself. We've also had the chance to install our finished tool onto the KUKA KR5 itself, and evaluate the quality of our tool using sensors, measurement, and by analysing our Arduino code.

1 Introduction

During Robotics course, the chance was given to study the KUKA KR5 robot, with many other robots [1] and automated systems [2], both theoretically, and practically, from the basic toolset to the robot's inner code and structure. At the moment, the robot uses a two finger pneumatic gripper as its tool. This gripper can hold objects ranging for 30 to 45 [mm] in diameter, and currently is used to grip on a tool used for educational purposes. This method is sufficient, but far from optimal, considering its small dynamic range, two-state (open or closed) nature only enables small objects to work with, and moreover they cannot be fragile as a result of either maximum or zero gripping strength. Because the Laboratory does not possess any more gripping tools, it was an obvious initiative to create more varied gripper both to increase the variety of tools, and to challenge us with this endeavour. We choose a 3 finger parallel gripper. As a result of the spread of 3D printing technologies, we could physically manifest the tool in real life as well. The supervision and control has been done with a Bluetooth module using an Arduino. Because the fingers are under constant surveillance by the sensor, we can get accurate measurements on the object held onto on the fly. We used a FSR (Force Sensitive Resistor) dynamometer to get the current clamping force on the fingers. After integrating the tool into the test environment, we proceeded to carry out some measurements and testing, to make sure our project was successful.

2 Presenting the mechanism

As our first step, before we started the 3D modelling, we had to specify the method of moving the fingers. The fingers are 120° form each other, and are being moved by a rotary component (*Fig. 1., Fig. 2.*).

Fig. 1. Presenting the mechanism

This enabled us to convert rotational movement to linear. Same method was used in steam engines, or being used in modern reciprocating (piston) engines, compressors or pumps.

Next step was to define the greatest angle which grants us safe and accurate, collision free movement. After the measurements, we got the precise angle of 106° to be used as the maximum range, which enables us to have a scope of 50 [mm] for gripping, which proved to be adequate. By knowing these values, we can use a sketch drawn in Fusion 360 [3] to accurately measure the upcoming components' sizes, and also to keep track of the angles, and mechanical model of our system. On the left side of the figure, we can see the motor at 0°, thus it's fully open position, while on the right we can see maximal 106°, approximately 50 [mm] long contraction.

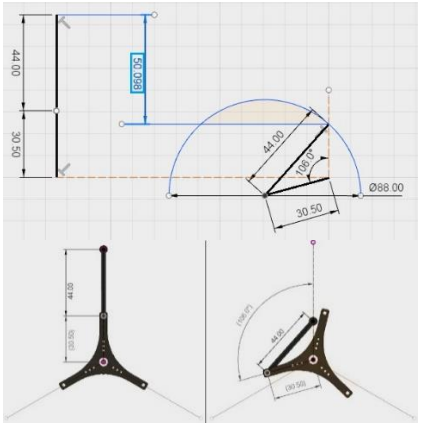

Fig. 2. Presenting the mechanism

* Corresponding author: timoteierdei@gmail.com

3 Presenting the 3D model

After we successfully drawn out our parts, and measured them correctly, we can get to 3D modelling the parts of our gripper. We used Autodesk Fusion 360 to create our 3D models. In the upcoming figures, we can see the parts of our gripper separately. On *Fig. 3.* we can see the frame of the gripper, which houses our motor, and the circular sliders placed 120° apart, to enable the fingers to move.

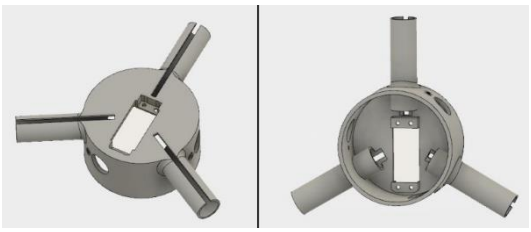

Fig. 3. Frame

The images down below (*Fig. 4.*) are the already presented 44 [mm] long pushrods (left), and the rotary component mounted to the motor later (right).

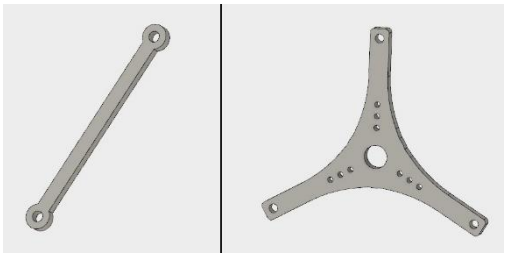

Fig. 4. Pushrod and rotary component

Fig. 5. shows our 3D modelled finger, which is made so that it can slide in the circular slider of the frame effortlessly.

Fig. 5. Pushrod and rotary component

Another task was to create correct a fitting mechanism in between the robot and the tool itself. The 6th axis of the KUKA KR5 possesses 4 mounting points where we can attach tools. By knowing the exact position and size of these bores, we made an adapter (*Fig. 6.*) so we can fit the tool onto the robot easily.

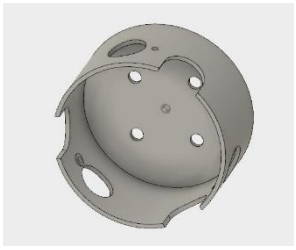

Fig. 6. The adapter

4 Assembling the parts in Fusion 360

As the first step of the assembly, we connected the TowerPro MG995 servo to our frame, then we mounted the rotary component onto it. The chosen servo has great range and great torque compared to its size. The table below shows some technical data about it:

Table 1. Specifications of the servo [4]

Servo type	Torque [Nm]	Speed [s/60°]	Weight [g]
TowerPro MG995	1	0,16	55

Coming up, we added the fingers, sled in from the side into the frame's sliders, connected with the pushrods of which thereafter connected to the rotary component. This concludes our 3D model of the gripper (*Fig. 7.*).

Fig. 7. Assembled 3D model

The grip range is the difference in between the fully open and closed positions, which lets us grab objects ranging from 9 [mm] to 109 [mm]. The adapter and the gripper itself can be sled together, and then secured together with screws (*Fig. 8.*).

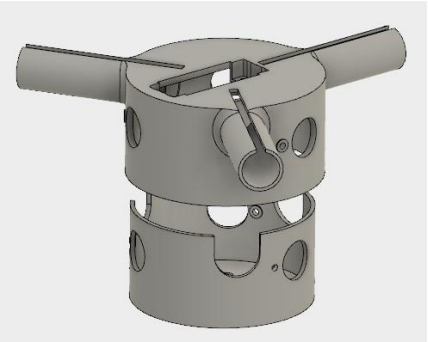

Fig. 8. Joining the adapter and the gripper together

5 Printing out the components in 3D

The printing has been done by a third party, after we agreed in the terms and materials of the print (*Fig. 9.*). Generally, ABS is more ductile, heat resistant and less prone to shattering, but since we won't need these advantages, since we won't move anything heavy or hot, PLA was a cheaper alternative. It's also preferred by hobbyists and designers also endorse it for its ease of use and larger colour variation [5].

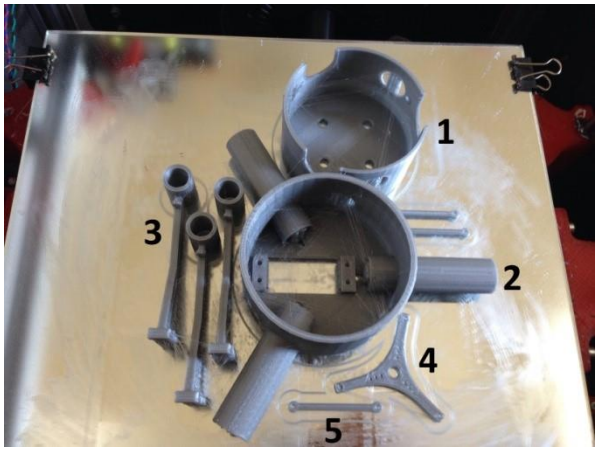

Fig. 9. The components printed from PLA

Ultimaker Cura takes the burden of setting all the parameters for each print from us, since it automatically sets it for us, saving a lot of time. The details can be found in the next table [6].

Table 2. Printing details per components

Component	Fill factor [%]	Total print time [hour:min]	Weight [g]	Cost [Ft]
1	30	2:48	31,1	165
2	30	6:54	82,1	435,1
3 (3 pieces)	30	2:54	23,7	126,6
4	30	00:10	1,6	8,5
5 (3 pieces)	30	00:06	1,2	6,3
Σ	N/A	12:52	139,7	741,5

The total weight of the gripper without the screws is only 194,7 [g], which takes up the 3,89% of the maximum of 5 [kg] load on the robot's last axis. After we manufactured the parts, we assembled in the same way as we planned it out in the virtual world (*Fig. 10.*). To further increase the grip strength, we used rubber ends on the fingertips.

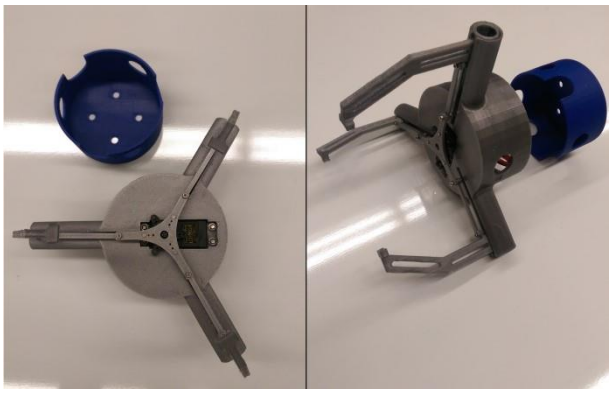

Fig. 10. The assembled gripper

6 Controlling the gripper with Arduino

As the basic methodology of servos, even if they hit an obstacle while turning, they won't stop, quite the contrary, they will try to continuously turn until the potentiometer inside reaches the given angle [7]. This can cause serious overheating, and later the failure of the servo. To project this issue onto our design, its not preferable to drive the servo for 0°to 106° as a method of gripping, since it will damage the servo in a short time. Thus, we tried to achieve the optimal state of gripping on the object, then locking the servo at that angle, preventing further movements, but also gripping on it steadily.

The issue was solved by analysing the potentiometer inside the servo. After we soldered a pin from the Arduino to the signal line of the potentiometer, we could easily, and accurately measure the state of the servo at any time in a scale of 0 to 1024 units. Then we used command "Map" to map out the potentiometer values for each angle, 122 for 20° and 302 for 126° (this +20 shift will be beneficial later).We saved this to the val variable, which will now always show +20 from the real value, which should be accounted for.

The program seen on *Fig. 11.* basic premise is to drive to motor from 0° to 106° inside a loop, incrementing the angle in each loop. As stated above, the val integer will keep increasing with the loop, as the motor is turning about, but if it gets stuck, it will cease to increase. As the grip happened, the loop will continue to run, and as It equals with the val integer, we can safely say that the grip has happened, so we set the servo to the loop's local variable x-17. The subtraction is important, since as val equals to x we are already overturning by 20°, so we step that down to only 3°which produces a firm grip, yet its light on the servo.

```
void loop() {
  if (Serial.available()) {       // is serial communication available?
    bluetooth = Serial.read();    // reading bluetooth signal
    Serial.println(bluetooth);    // writing bluetooth signal

    if (bluetooth == '+') {             // if the received signal from bluetooth is '+'
      for (int x = 0; x < 107; x++) {
        myservo.write(x);               // move the servo from 0 degrees to 106 degrees
        int val = analogRead(A5);       // read the value of the servo potentiometer
        val = map(val, 122, 302, 20, 126); // mapping the 'val' variable between 12 and 118 degrees
        Serial.print(x);                // writing the value of 'x'
        Serial.print("    ");
        Serial.println(val);            // writing the value of 'val'

        if (x == val)   // if the servo gets stuck, the 'x' value reaches the 'val' value
        {
          myservo.write(x - 17);    // stop the servo at the stuck point
          Serial.println(x - 17);
          break;
        }
        delay(35);
      }
    }
    else {              // if the bluetooth signal is '-'
      myservo.write(0); // servo goes to 0 degrees, the gripper opens
    }
  }
}
```

Fig. 11. A snippet from the Arduino code

The gripper can be controlled via Bluetooth, with the Module HC-06, along with a free Android app downloadable online [8]. The App can send two commands to the Module, on the first press, it starts the contraction, by sending a + signal, and then we can press it again to reverse the contraction, loosening it, with the - signal. We've drawn the wiring diagram in Fritzing, shown on *Fig. 12*. [9][10].

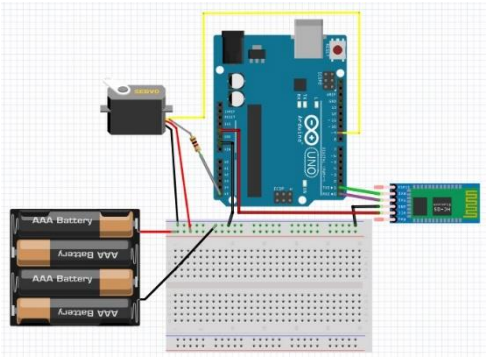

Fig. 12. The Arduino wiring diagram

The power for the motor is supplied by external source, since the Arduino cannot give it enough current to run smoothly. We also had to wire in a 1 [kΩ] resistor, so the signals would not fluctuate so much, and we grounded the whole contraption to a common ground so we achieve the desired results.

7 Calculating the diameter of the held object

By stopping the servo on the grip position, we can precisely calculate the diameter of the object held onto, using the position of the servo. The code presented it is obvious, that we can use the motor's present angle stored in x to calculate the diameter of said object. We will go through the calculation in a parametric fashion. On the figure below (*Fig. 13.*) we can see the symbols, and for added methodical approach, we added the indications for both left and right figures.

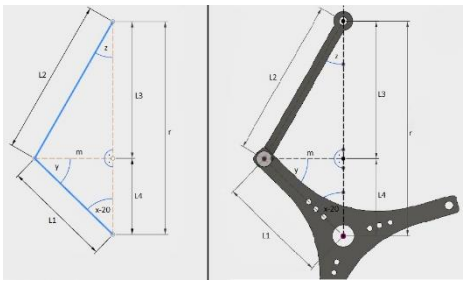

Fig. 13. Notable points and edges for the mechanical model

$$sin(x-20) = \frac{m}{L1} \tag{1}$$

As we've already stated x is always shows 20° more than the real value, so we subtract it.

$$m = sin(x-20) * L1 \tag{2}$$

$$sin(z) = \frac{m}{L2} \tag{3}$$

$$z = arcsin\left(\frac{m}{L2}\right) \tag{4}$$

$$L3 = cos(z) * L2 \tag{5}$$

$$L4 = cos(x-20) * L1 \tag{6}$$

$$r = L3 + L4 \tag{7}$$

$$d = (L3 + L4 - 20) * 2 \tag{8}$$

L1 and L2 are constants.

The 8th formula represents the 20 [mm] shift from the calculated to the real position. The multiplication of the radius with 2 equates to the diameter of the object. After we declared the necessary variables, we simply implemented the formula into the Arduino code (*Fig. 14.*), sometime consolidated together for saving space (and memory).

```
m = sin(radians (x-20))*L1;
w = m/L2;
z = asin(w);
d = (((cos(radians(x-20)) * L1) +( cos(z) * L2)) - 20) * 2;
Serial.print("The diameter of the gripped object = ");
Serial.println(d);
```

Fig. 14. Calculating the diameter of the held object

8 Determining the clamping force using the FSR 400 dynamometer

We've chose the FSR 400 (Force Sensitive Resistor) dynamometer for this task. The sensor is basically a strain gauging resistor increasing its resistance with pressure in between 0,2 [N] and 20 [N] [11]. The measurement can be carried out easily, visible on this simple wiring diagram:

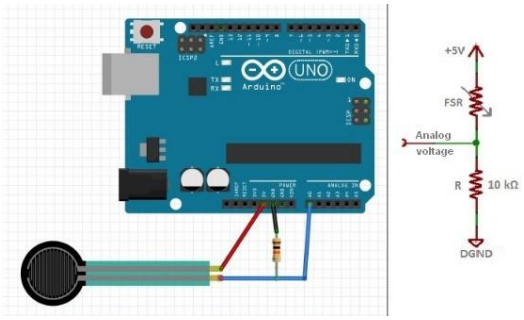

Fig. 15. Wiring diagram of the FSR 400 measurement [12]

One pin of the sensor goes to the 5 [V] power supply, the other goes to the common ground after a scraper resistance. Additionally, we connected the analog pin in between the scraper resistance and the FSR 400 (*Fig. 15.*). After overcoming the minimal threshold load (pressure), the FSR acts like a resistor of 100 [kΩ]. Proportionally with the pressure, the resistance drops from 100 [kΩ] to a 10 [kΩ] minimum. This means an increasing voltage on the line, equating to data for us on the analog pin. After the wiring had been done, we created a program to acquire the results from raw voltage (*Fig. 16.*).

```
int fsrAnalog; int fsrVolt; unsigned long fsrResistance;

void setup() {
  Serial.begin(9600);
}

void loop() {
  fsrAnalog = analogRead(A0);
  Serial.print("Analog value = ");
  Serial.println(fsrAnalog);

  fsrVolt = map(fsrAnalog, 0, 1023, 0, 5000);
  Serial.print("fsrVolt value in [mV] = ");
  Serial.println(fsrVolt);

  if(fsrVolt == 0) {
    Serial.println("No pressure");
  }
  else {
    // FSR = ((Vcc - V) * R) / V
    fsrResistance = 5000 - fsrVolt;
    fsrResistance *= 10000;
    fsrResistance /= fsrVolt;
    Serial.print("FSR resistance in ohm = ");
    Serial.println(fsrResistance);
  }
  Serial.println("------------------------");
  delay(1000);
}
```

Fig. 16. The code for the FSR 400 dynamometer

Firstly, as usual, we declared our variables. The main loop first prints the raw value of the analog pin onto the serial plotter. This equates from 0 [V] to 5 [V] proportional of the pressure, to increase the resolution, we used the command map to convert this from 0 [mV] to 5000 [mV]. We printed out the voltage to the serial monitor as well, in millivolts. Without any load, the program simply writes out: "No Pressure". We can calculate the voltage on the scraper resistance with the following formula:

$$Ur = U\left(\frac{R}{R + FSR}\right) \tag{9}$$

By sorting the equation, we can get the formula:

$$FSR = \frac{(U - Ur) * R}{U} \tag{10}$$

U = 5 [V], R = 10 [kΩ]

We implemented this formula into our Arduino code, resulting the FSR sensor's resistance in [Ω] (*Fig. 17.*). To measure the clamping for of the gripper, we use a 30 [mm], a 60 [mm] and a 90 [mm] diameter test prop, printed out in 3D [13]. We secured each prop with the gripper, while having the FSR 400 on one of the fingers. We can see the results of the 60 [mm] object on the image below.

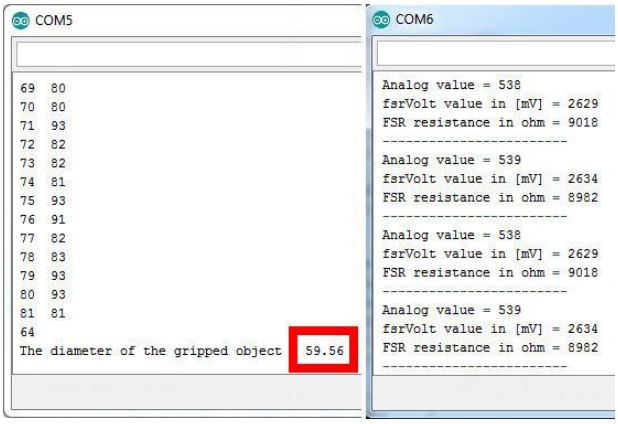

Fig. 17. Measurement results of the FSR 400

On the left side, the already presented diameter calculation is visible, and on the right, we can see the measurement results the FSR resistance's results are necessary to measure the force acting upon the sensor. Visible on the diagram below, where we display the results on the y axis, then intersecting that at 30,60 or 90 (diameter), then intersecting those points from the arc to the x axis, then divide these values by 100, we can get the correct force in [N].

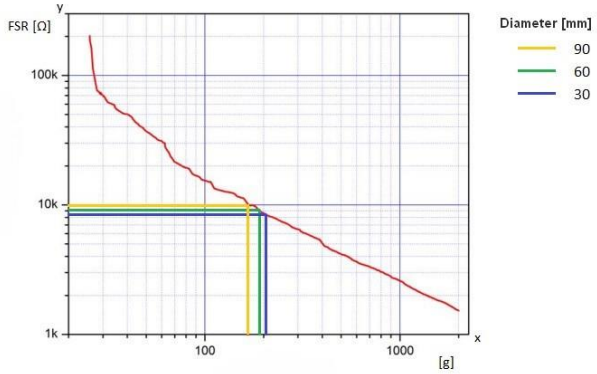

Fig. 18. Diagram of the FSR dynamometer

Interpreting the values, we can see that the finger acts with 1,7 [N], 1,9 [N] and 2,05 [N] on different diameters (*Fig. 18.*).

9 Reference work

After completing the setup, we installed the gripper with the adapter mentioned above to the robot (*Fig. 19.*), also, we rigged the Arduino onto the robot as well, for simplicity's sake.

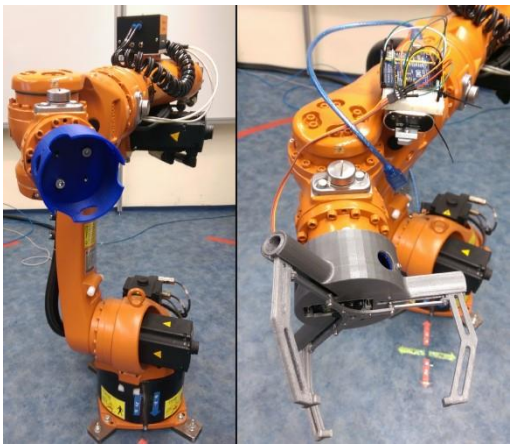

Fig. 19. Rigging the gripper onto the robot

After installing, we manipulated different objects, lifting them, or gripping them alike (*Fig. 20.*).

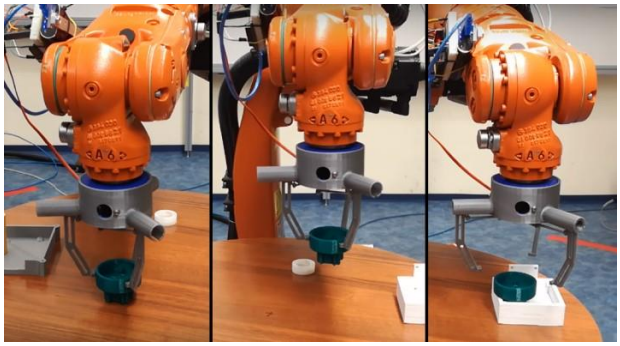

Fig. 20. The working robot

9 Conclusion

We can confidently say that we created a working gripping tool for the KUKA KR5, and acquired useful and practical knowledge on the way. The code and sensor working together can measure diameter and clamping force, adding another layer of complexity of our system. In this paper we went over the detailed creation of the whole project, and we are considering further improvements for the tool.

Acknowledgment

The publication was supported by the project EFOP-3.6.1-16-2016-00022. This project was co-funded by the European Union, from the European Social Fund.

References

1. Zs. Molnár, T. I. Erdei, N. C. Obinna, G. Husi, „*A novel design of an air-cushion vehicle and its implementation,*" MATEC Web Conf. Volume 126
2. T. I. Erdei, Zs. Molnár, N. C. Obinna, G. Husi, „*Surveillance and Security System in the Building Mechatronics Research Center,*" Electrical Engineering and Mechatronics Conference EEMC'16 - 7th-18th-19th March, 2016.
3. „*Autodesk Fusion 360*," [Online]. Available: https://www.autodesk.com/products/fusion-360/overview. [Access Date: 27 11 2017].
4. „*TowerPro MG995 Servo*," [Online]. Available: https://servodatabase.com/servo/towerpro/mg995. [Access Date: 27 11 2017].
5. „*ABS or PLA: Which 3D printing filament should you use?,*" [Online]. Available: https://www.digitaltrends.com/cool-tech/abs-vs-pla-3d-printing-materials-comparison/. [Access Date: 27 11 2017].
6. „*Ultimaker Cura software,*" [Online]. Available: https://ultimaker.com/en/products/ultimaker-cura-software. [Access Date: 27 11 2017].
7. F. Reed, „*How Do Servo Motors Work,*" [Online]. Available: https://www.jameco.com/jameco/workshop/howitworks/how-servo-motors-work.html. [Access Date: 27 11 2017].
8. „*Arduino Bluetooth controller,*" [Online]. Available: https://play.google.com/store/apps/details?id=com.giumig.apps.bluetoothserialmonitor. [Access Date: 27 11 2017].
9. „*Arduino,*" [Online]. Available: https://www.arduino.cc/. [Access Date: 27 11 2017].
10. „*Fritzing,*" [Online]. Available: http://fritzing.org/home/. [Access Date: 27 11 2017].
11. „*Force Sensitive Resistor (FSR), Overview,*" [Online]. Available: https://learn.adafruit.com/force-sensitive-resistor-fsr/overview. [Access Date: 27 11 2017].
12. „*Force Sensitive Resistor (FSR), Using an FSR,*" [Online]. Available: https://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr. [Access Date: 27 11 2017].
13. T. I. Erdei, Zs. Molnár, N. C. Obinna, G. Husi, „*A Novel Design of an Augmented Reality Based Navigation System & its Industrial Applications,*" 15th IMEKO TC10 – Technical Diagnostics in Cyber-Physical Era Budapest, Hungary, 6 – 7 June, 2017 - Organised by: MTA SZTAKI – Hungarian Academy of Sciences - Institute for Computer Science and Control.