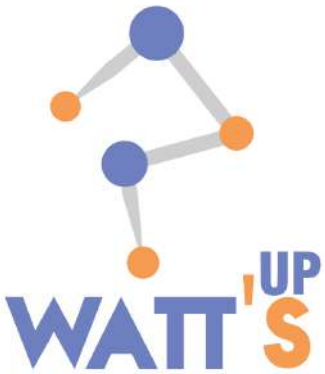
VERSION 2.0

2017-2018



ENGINEERING NOTEBOOK

PRESENTED BY: TEAM RO023

WATT'S UP

TABLE OF CONTENT

NOTEBOOK GUIDELINES

Our Notebook is used to show the progress or connect, communications, social media and fundraising programs. This Notebook contains our business plan and team bios.

Using the Notebook to record ideas, inventions, experimentation records, observations and all work details is a vital part of any engineering process. Careful attentions to how to keep your Notebook can have a positive impact on the patent outcome of a pending discovery or inventions.

We took a lot of time developing goals and our design process began after we learned what challenges we would facing in BRD FIRST TECH CHALLENGE SEANSON 2. Although this is not our first year in the competition, considering that there were some major changes, we could have been a little discouraged by the new tasks ahead of us. But on the contrary, this notebook is the material proof that we are able to put up to whatever change we might have to face and also an invitation to check out our work through all its phases.

1.WATT'S UP TEAM RO023

• TEAM DEVELOPMENT

Our top goal this year was to field our strongest team and best robot to achieve more than in our previous season. We put significant effort into determining not only each student's strengths, but also how to use those various strengths to develop the best team.

To accomplish this goal and to properly organize the work we have to do we divided our team in 4 sub-teams:

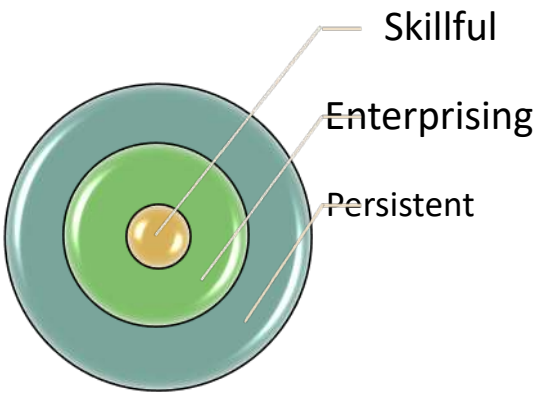
1) Programming team
2) Building team
3) Design team ("Creo team")
4) Social media & promotion team

All 4 teams worked together and took advantage of the experience that had gained in the past season in order to be able to design, build, promote, and compete with a robot.

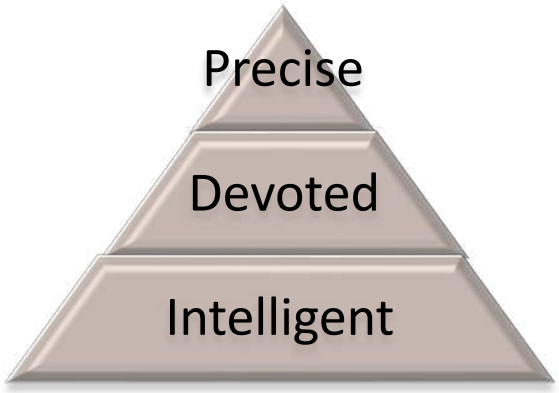
We also hope that our effort will be appreciated, like the year before, by the American Ambassador Hans Klem.

• TEAM MEMBERS

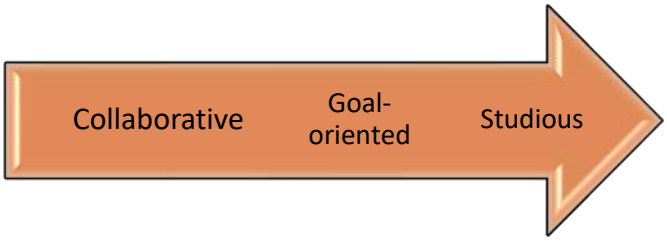
This is Bogdan Sitaru and he is an important part of the team. He was our coach in the competition that took place last year and his hard work has paid off. He is in charge of the programming team, but he also makes a great work at keeping us all focused and implicated. His native passion for computer science has determined him to join this competition and now, he has the chance to be part of a hardworking team and a tight friendship. This is one of the faces behind the program that makes our robot one of the best.



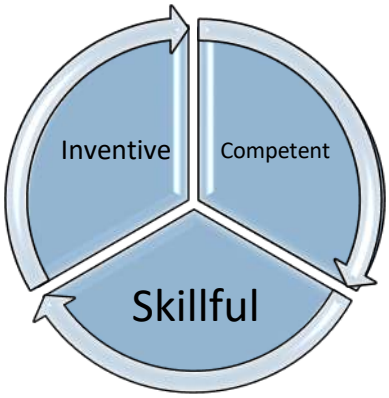
He is Costin Andrei Oncescu and you might have heard of him. He is one of the most diligent people, so his work is always exemplary and his contribution to this team project is of major importance. Given his award winning passion for computer science, he is part of the programming team. Despite all of the contests he has to attend, Costin still manages to make time for some more than useful teamwork sessions. He has shown his skills in the last FTC competition so we count on him to help us, once again, to build one of the best robots.



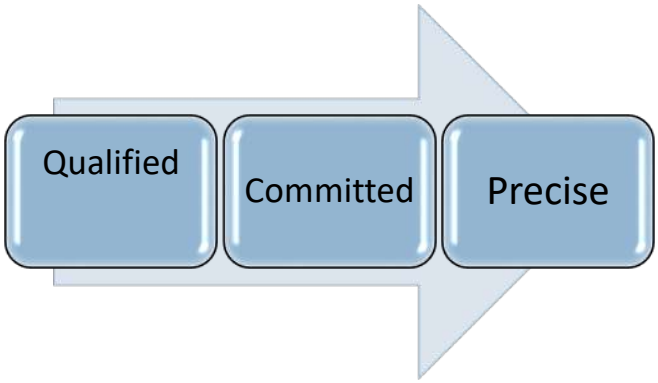
This is Andreea Dutulescu and she works at the construction of the robot and helps the “girls team” to handle all the social aspects of the competition. The reason she entered this competition is, first of all, to fulfill one of her childhood dreams. Since she was a little kid, she has enjoyed reading SF books and she was always wondering if the human race could be capable of building such things. Nowadays, not only we are capable, but she is a part of this amazing project in which we combine everything we had learned at physics, math and computer science classes and also develop teamwork skills.



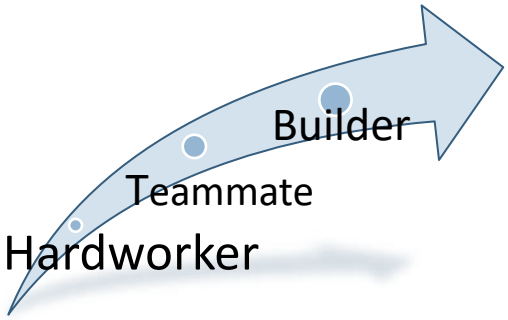
This is Nicolae Binica, and he is also in his second year in the competition. He is part both of the programming team and the building team and he has already shown his devotion in an extreme situation that occurred last year at FTC, when he chose work over sleep. He has chosen to participate in this project because of his passion for physics and computer science, fact shown by his long history with school competitions. He claims that his last year of high school should be the most memorable one, so he is ready to face with a positive attitude any challenges in this year's FTC competition.



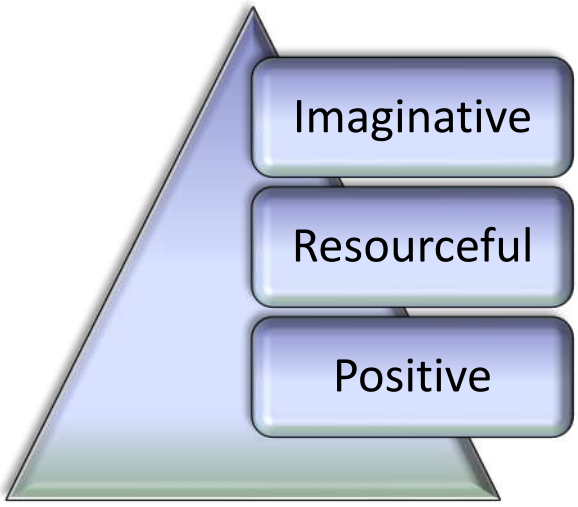
He is **Leonard Necula**, one of our CREO specialists. We can call him an expert in this domain because of his successful application that created a stir in last year's competition. He has promised himself to outdo his previous performance, but still, he manages to make time for working with the 3D printer. Leo has become really handy with it, so now he helps all the other teammates when it comes to making new pieces for the robot.



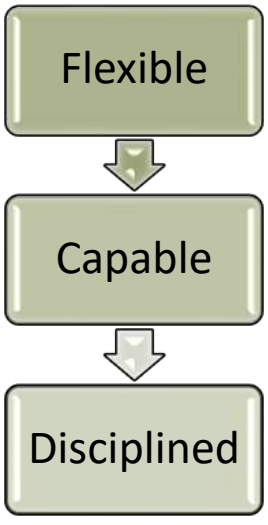
This is Andrei Popescu, passionate about robots but especially about their construction. Although he is just in the 10th grade he has always wanted to see how a robot works and how much effort is put into creating one. This is his second year in the competition and his enthusiasm has grown into a passion. He is part of the building process and he enjoys every second of teamwork he takes part of.



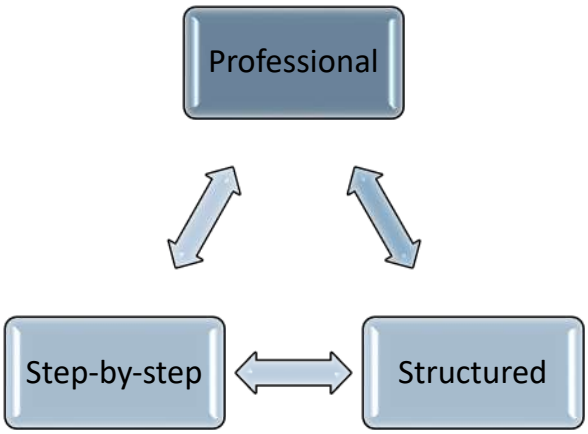
This is Daria Broscoteanu and she is part of the social media & promotion team. This is her first year in the competition and, as a newbie, she has to keep up with the team's Facebook page (FTC WATT's UP). She took all the pictures for this notebook and also helped at the general organization of our working lab. She was promised an unforgettable experience and she is more than eager to be part of this year's actual event, that takes place in Bucharest.



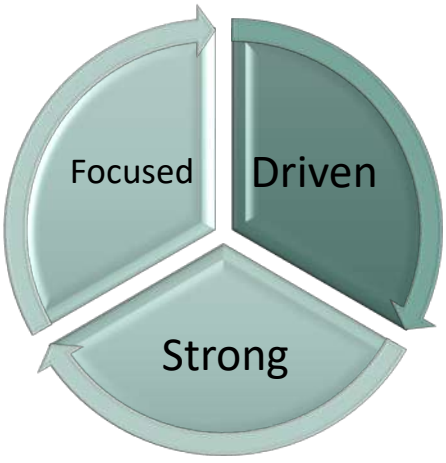
He is Radu Chivereanu and he is part of the Design team (“Creo team “). This is not his first year in the competition but it is his first one in this branch. He started by making the CREO components of the robot one by one, and than tried to make them a singular object. The final program is of the most importance, since we can install it on our phones and easily show our robot in a 3D format to anyone interested. He is a devoted member of the team and works hard to accomplish our common goal in this year’s FTC Challenge.



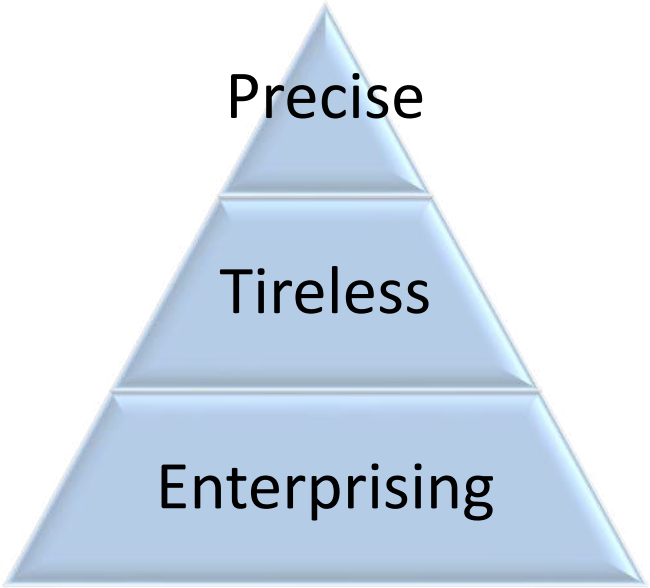
This is Andrei Duta and he is one of our precious drivers. He plays one of the most important roles in the actual competition, as he is in the first line of the battle. He has to synchronize with the other driver and the coach in order to control the robot properly, and that implicates a lot of hours spent in the lab, training with each other. Besides that, he is an active member of the team and always present in the working lab in case his skills are needed, always more than willing to help with anything.

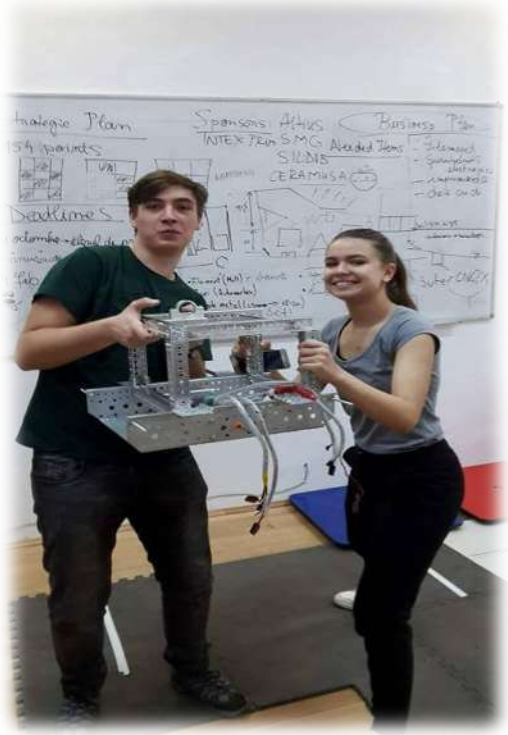


He is Octavian Milea and he is one of the brilliant minds in the programming team. He is a low key veteran in this sort of IT events, and not only because this is his second year in the competition, but also because it has always been his passion that pushed him to try this sort of experiences. He invested a huge amount of time into our robot but his efforts shall be rewarded hundredfold.

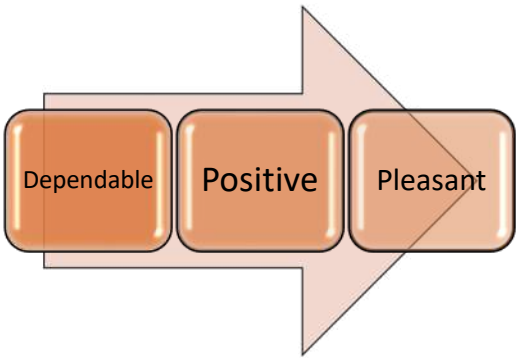


This is Alexandru Tesileanu and he is an important part of the building team and also one of the drivers. He has a native passion for crafting, manifested, at first, through the time he had put into working on his bike, and now, through the dedication he puts into this robot. Since he knows this little machine better than anyone, it felt normal for him to control it through a joystick. He is active both behind the scenes and in the front row of the competition and we are grateful for such a devoted team member.





This is Luiza Achim and she is one of the last year's volunteers and now, an active member of the team. During the competition she finds a way to be handy with everything she has to do, from being open to those visiting the stand, to charging the batteries. This year, she helped with the notebook and with the marketing part, eager for the competition to begin, so she could witness another memorable experience.



This is Iuliana Balica and she is writing the Engineering Notebook and handle the marketing part. She is full committed to her tasks and she is successfully succeeding to do all the work is needed.

• COACH

This is Dumitrascu Octavian and he is our coach. He is the so cold "leader of the pack" and we trust him completely since he has been the best at his job since last year's competition. He deals with the sponsors and makes sure that everything is on track, and also coordinates the programming team since he is one of the best computer since teachers in our town. He is one of the people without whom none of this would have been possible.

• MENTORS

She is Tatulea Maria and she is, besides an incredible mentor, one of the teachers involved in our team project. She helps everybody in need, mostly the programming team, but also supervises the entire creative process. Last year, Mrs. Tatulea has proven her devotion to this activity by being there for every little incident we faced and helping us get through with it. We are really grateful for her true implication.

3.OUTREACH

• TEAM OUTREACH

At the beginning of the BRD FIRST TECH CHALLENGE SEASON 2 we had to organize, mobilize and set the priorities. The first thing that we have done it was a meeting with all the members of the 1st season team to determine the following points:

- 2nd season theme
- Game rules
- Add new people in the team or find more volunteers

But we didn't realize that we had to do a lot of things in order to get all the tools that we needed.

The next time we met we did a brainstorm where all the members who knew the rules of the contest talked about the sponsors, strategy, constructions, design and how to arrange the lab to have more space and too maintain it clean.

• SHORT TIMELINE

27.09.2017 - We cleaned the lab and we came up with some ideas for the promotion video.

30.09.2017 - At the brainstorming meeting we talked about game rules, strategy, construction and design.

02.10.2017 - We have begun to design our tile runner in PTC Creo and we were thinking about the arena and the tools we need for building.

3.10.2017 – To make more room in the lab we put some shelves and organize our things.

12.10.2017 – Sponsors talk, parts order

16.10.2017 – Tile runner assembly, printing some gears

19.10.201- Ghlyphs grip system

26.10.2017- Singin up for the contest

31.10.2017- Disassembly of the 1st season robot

01.11.2017- Presentation video

05.11.2017- Registration form

07.01.2018- Singin up for the regionals

18.01.2018- Deadlines

05.02.2018- Promotional and advertise product

11.03.2018- Regional competition

• VOLUNTEER OPPORTUNITIES

As a volunteer in our team people need to express a willingness to undertaken a service. Volunteers are working together helping our team to stay organized and keep up with the latest events in our domain. They are a part of the team and without them things would have been harder. Beside them we learned how to listen to each other to be able to handling things in a smooth way and how to work like a team.

Every volunteer has unique opportunities like working with our coding team, learn how to use CAD or PTC Creo or even to be creative and crafty.

3.2017-2018 BUSINESS PLAN

• SPONSORSHIPS

To finance the effort of our team to build a strong and functional robot we needed some support from sponsors. With the intention of attracting active and serious sponsors who are willing to invest in our work, we had to promote our team in every possible way: we made an promotional video, a Facebook page where we are constantly active and a Youtube channel, (https://youtu.be/DiA8uzYEwXc).

After multiple attempts and mails we found or main sponsor Altius.

Altius is a company that distributes veterinary products that have an extraordinary team that helps not just animals but also our team development.

Other sponsors:

But we did not get help just from sponsors. Two of our team members have made our idea possible offering money which they have earned through the effort they have made to multiple international olympics and contest. We are talking about Bogdan Sitaru and Costin Oncescu special students without whom we would not have succeeded in this competition. We also need to say “thank you” to Anda Tatulea for her implication with the promotional materials.

„Profesorul de genii“, despre olimpicii care aleg Occidentul: „Nu trebuie să le îngrădim dorința de a se perfecționa“

ADEVARUL.RO

• FTC BUDGET PLANNED VS ACTUAL

ITEM	BUDGET (LEI)	ACTUAL COST (LEI)	CATEGORY
EXPENSES			
KIT OF PARTS: COMPETITON SET	2700	2,653	Robot Supplies
KIT OF PARTS: CONGTROL & COMUNICATION SET	2100	2000	Robot Supplies
KIT PARTS: ELECTRONICS MODULES & SENSORS	1000	1865	Robot Supplies
TOOLS	1200	1400	Robot Supplies
TEAM T-SHIRTS	1000	1000	Team Supplies
PROMOTIONAL MATERIALS	1500	1500	Team Supplies
TRANSPORT	1000	1032	Team Supplies
FOOD	1200	1300	Travel
ACCOMMODATION	2300	2300	Travel
PRINTING	500	600	Team Supplies
SUB-TOTAL	16000		
TOTAL		15650	

• TEAM CONTACT INFORMATION

MAIN CONTACTS:

Head Coach Dumitrascu Octavian Dan

Email: ftcwattsupp@gmail.com

Phone: 0722551305

SPONSORSHIP INFORMATION:

Checks should be made payable to Colegiul National Dinicu Golescu

MAILING ADRESS:

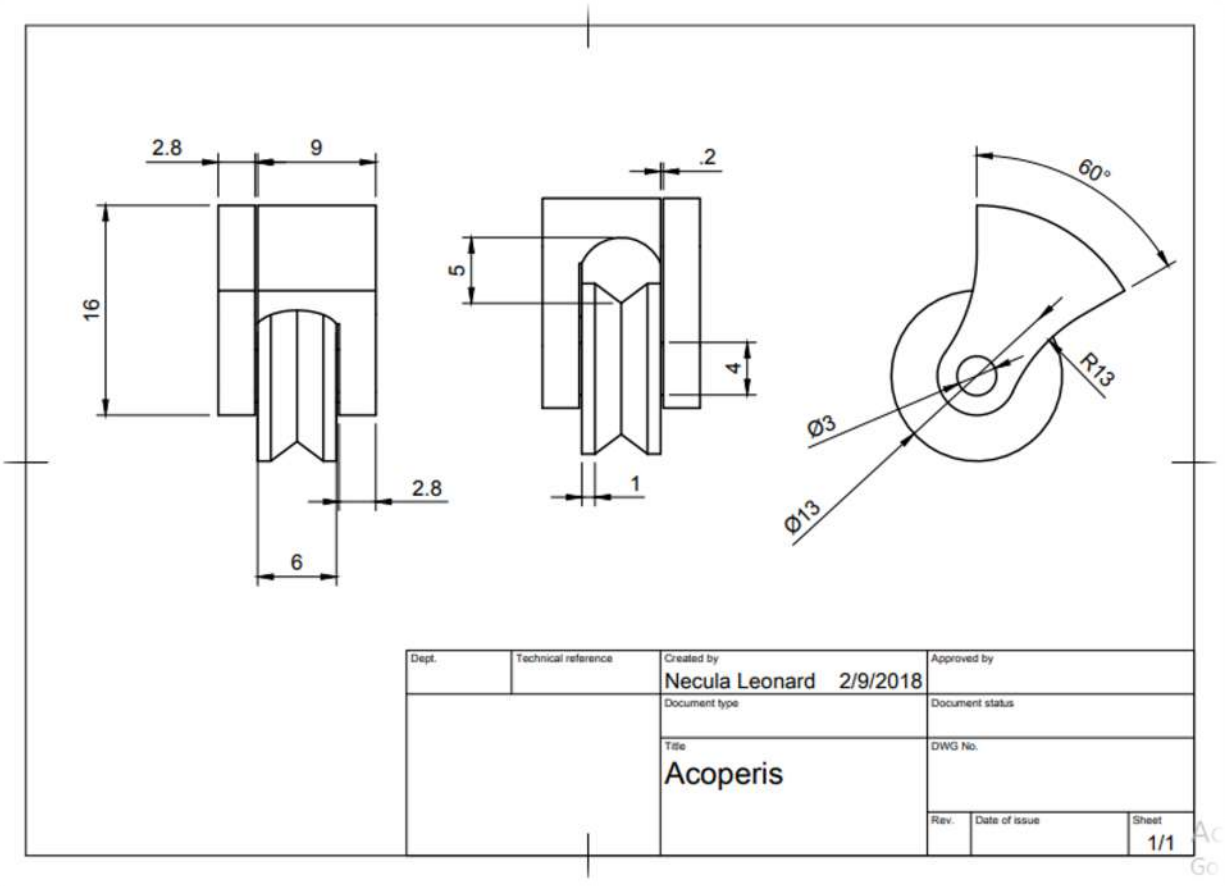
Strada Negru Voda 66, Campulung Muscel, Arges, 115100, Romania

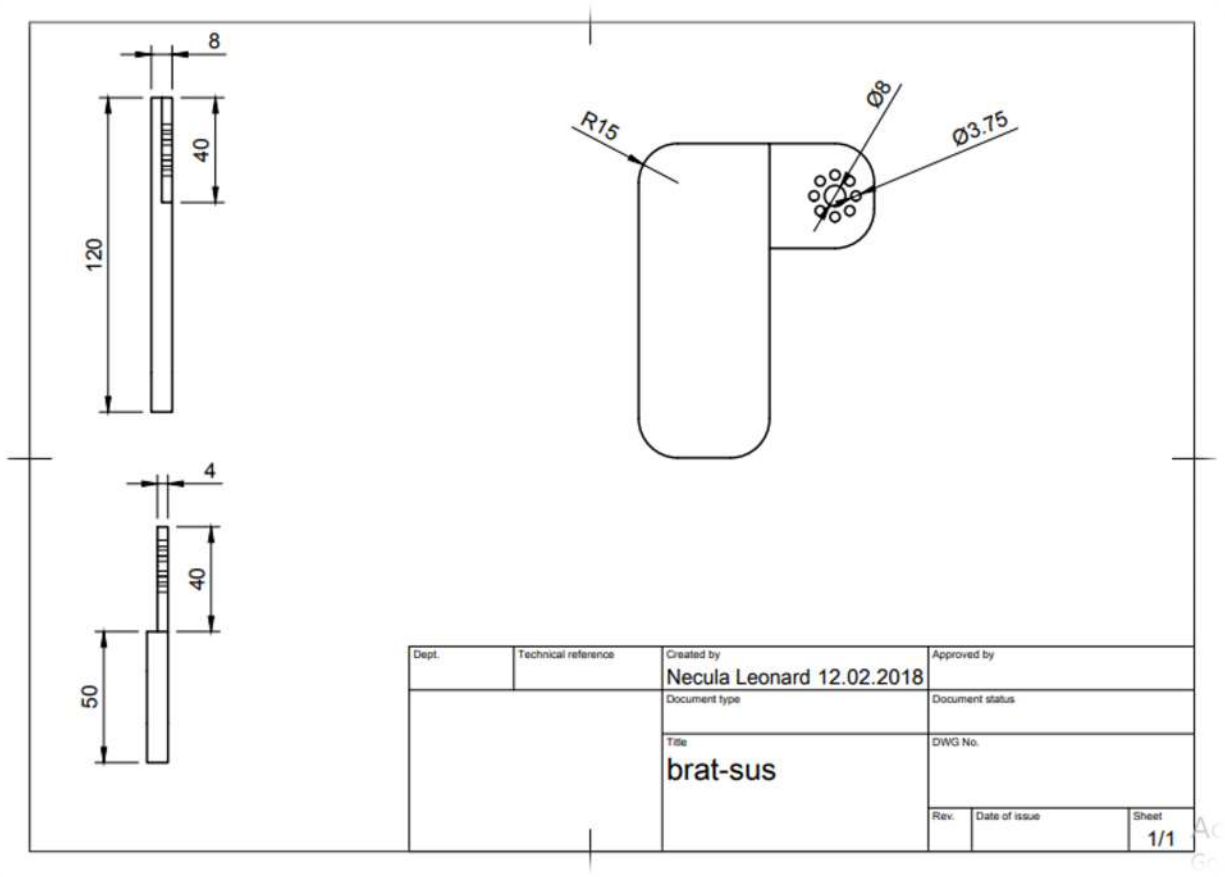
4.DESING

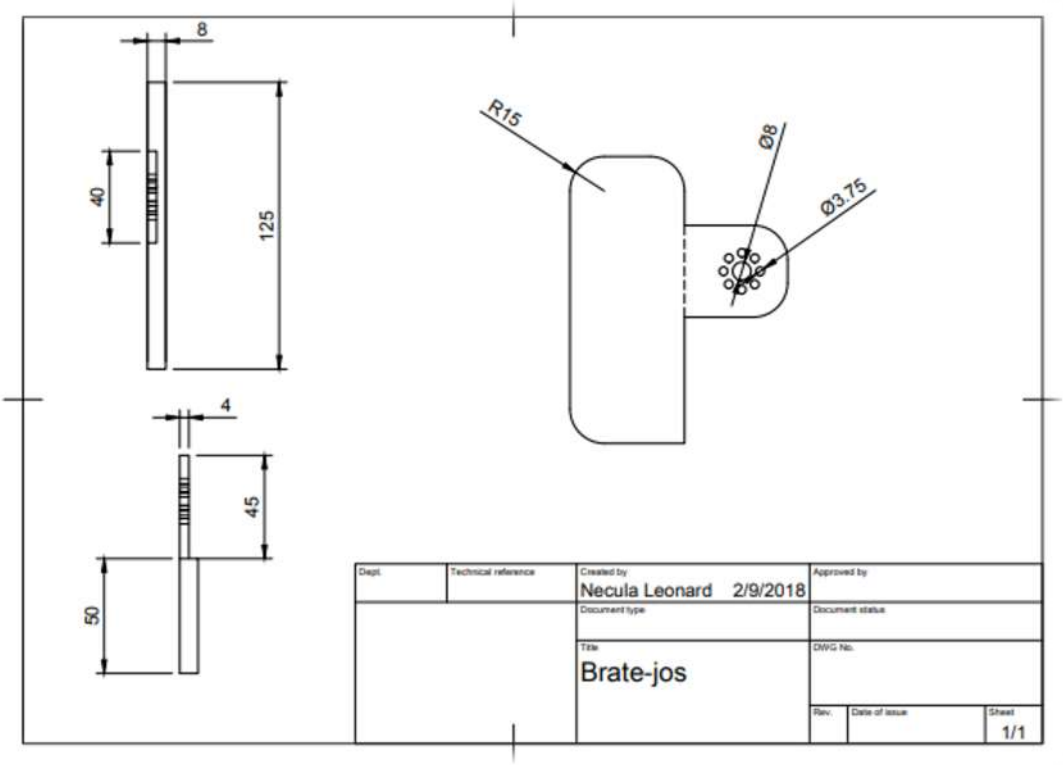
• CAD SKETCHES

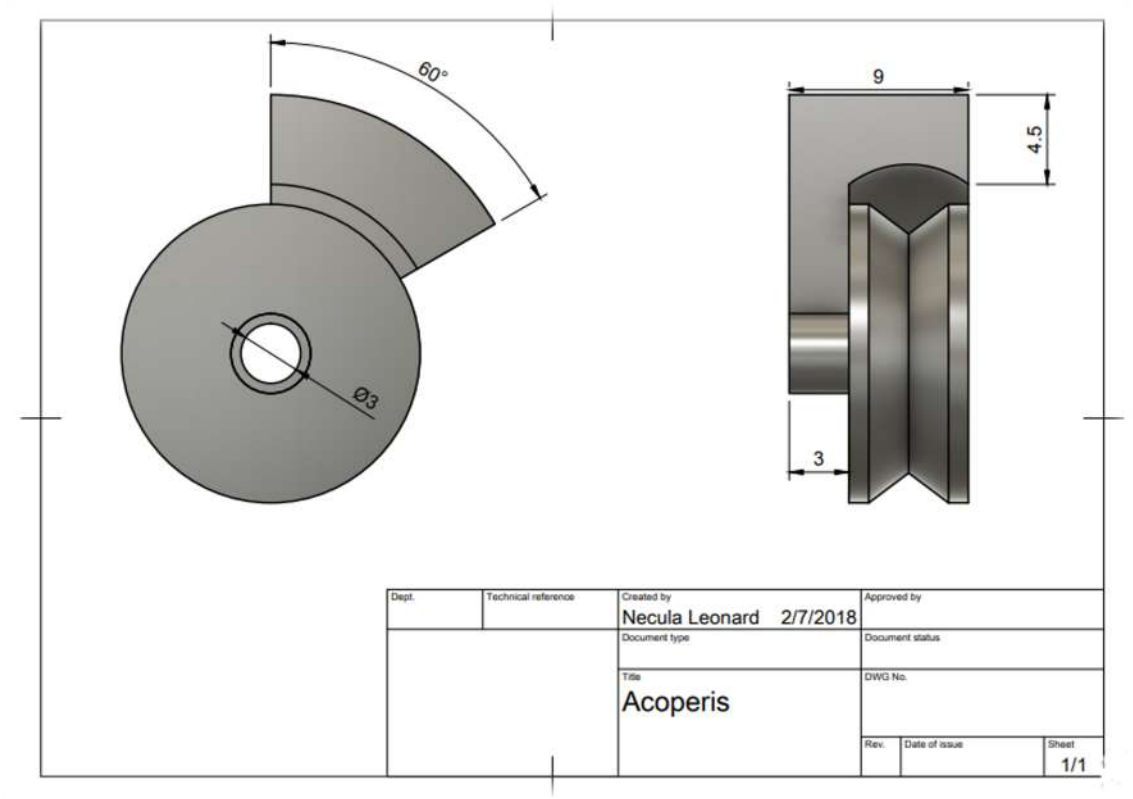
After a very long brainstorming session we came up with a rough design for the robot. We asked one of our sponsors to provide us with a tile runner style chassis. For the glyph grabber we use a system made with 4 servo motors, we collect one glyph with the first 2 servo motors then we lift it and with the other 2 servos we collect another glyph. For the relic delivering system we use a cascading slider for a faster delivery (we are working on retracting the slider). One of the problems of our robot is its speed, it's a slow poke.

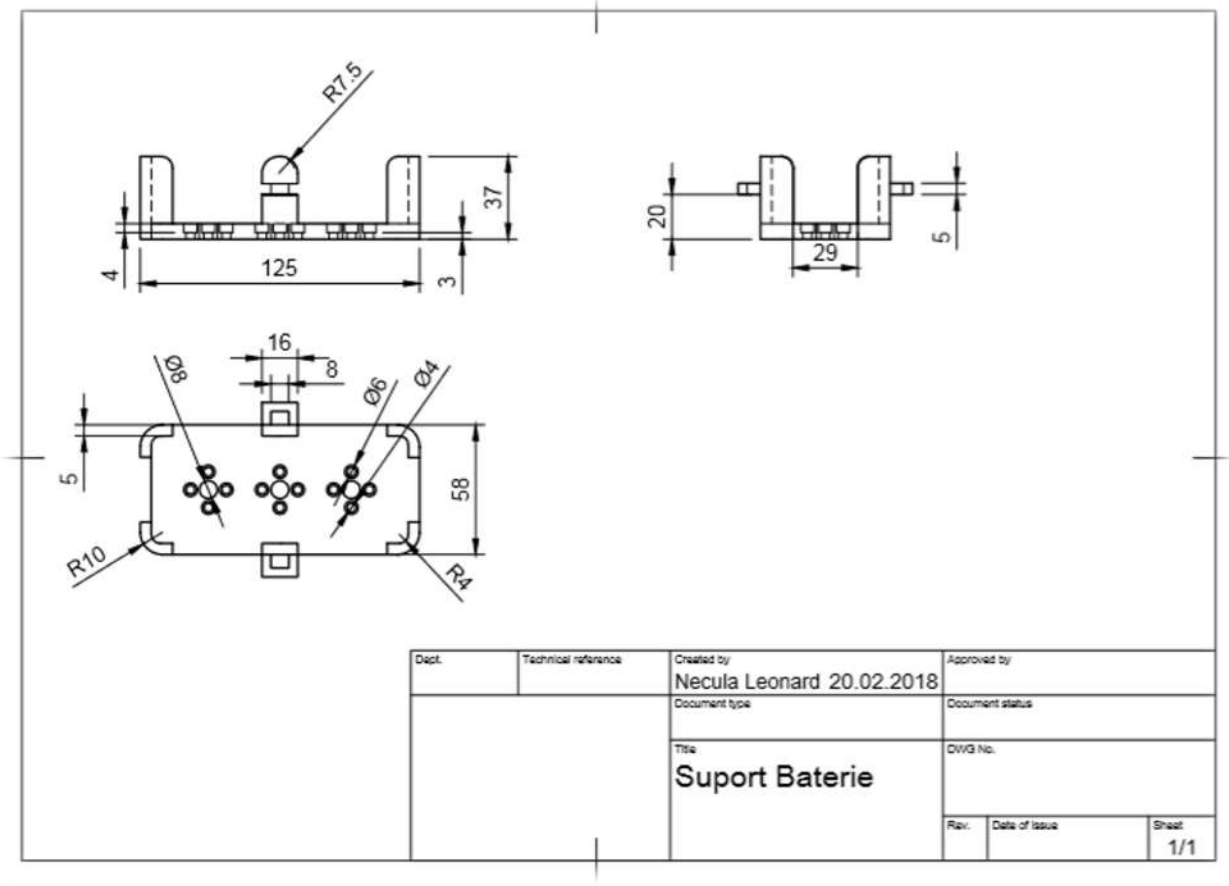
Here are some sketches:

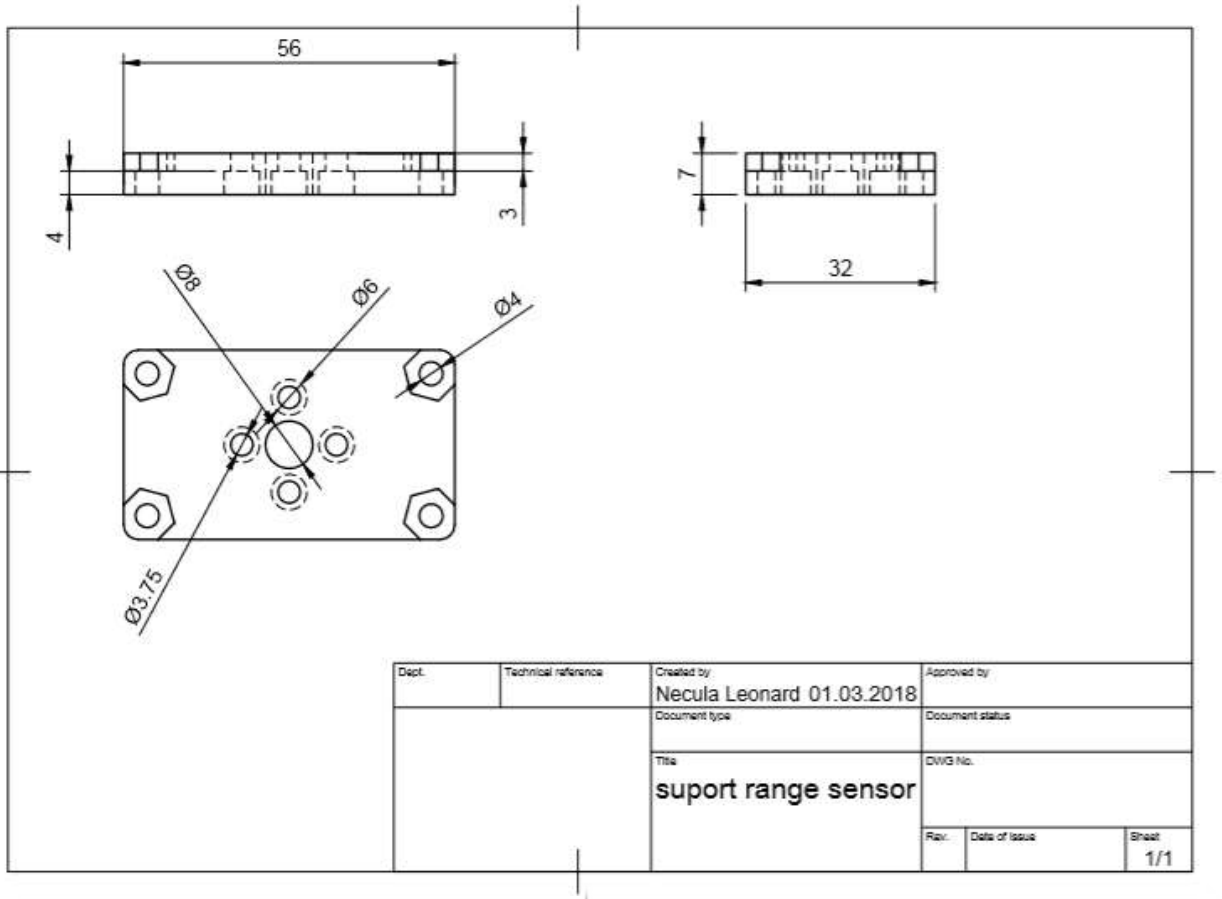


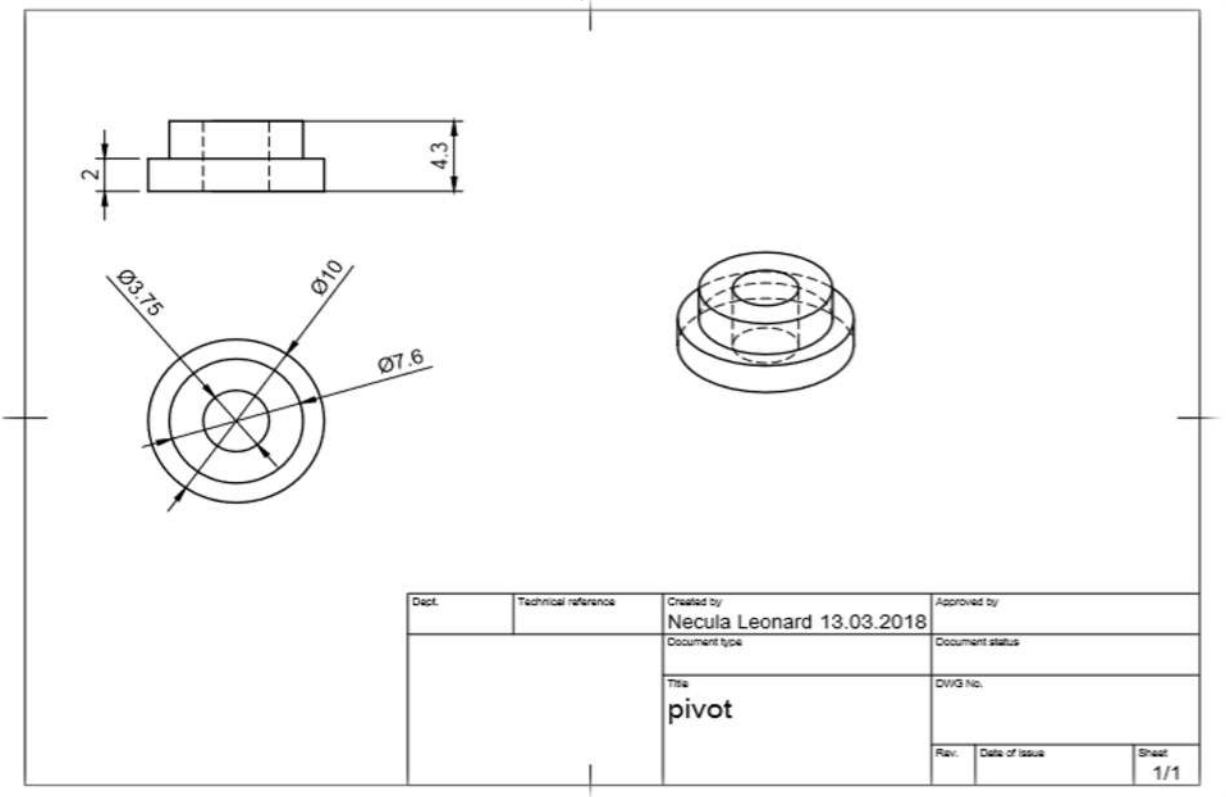

8
40
120
4
40
50
R15
Ø8
Ø3.75
Dept.
Technical reference
Created by
Necula Leonard 12.02.2018
Approved by
Document type
Document status
Title
brat-sus
DWG No.
Rev.
Date of issue
Sheet
1/1

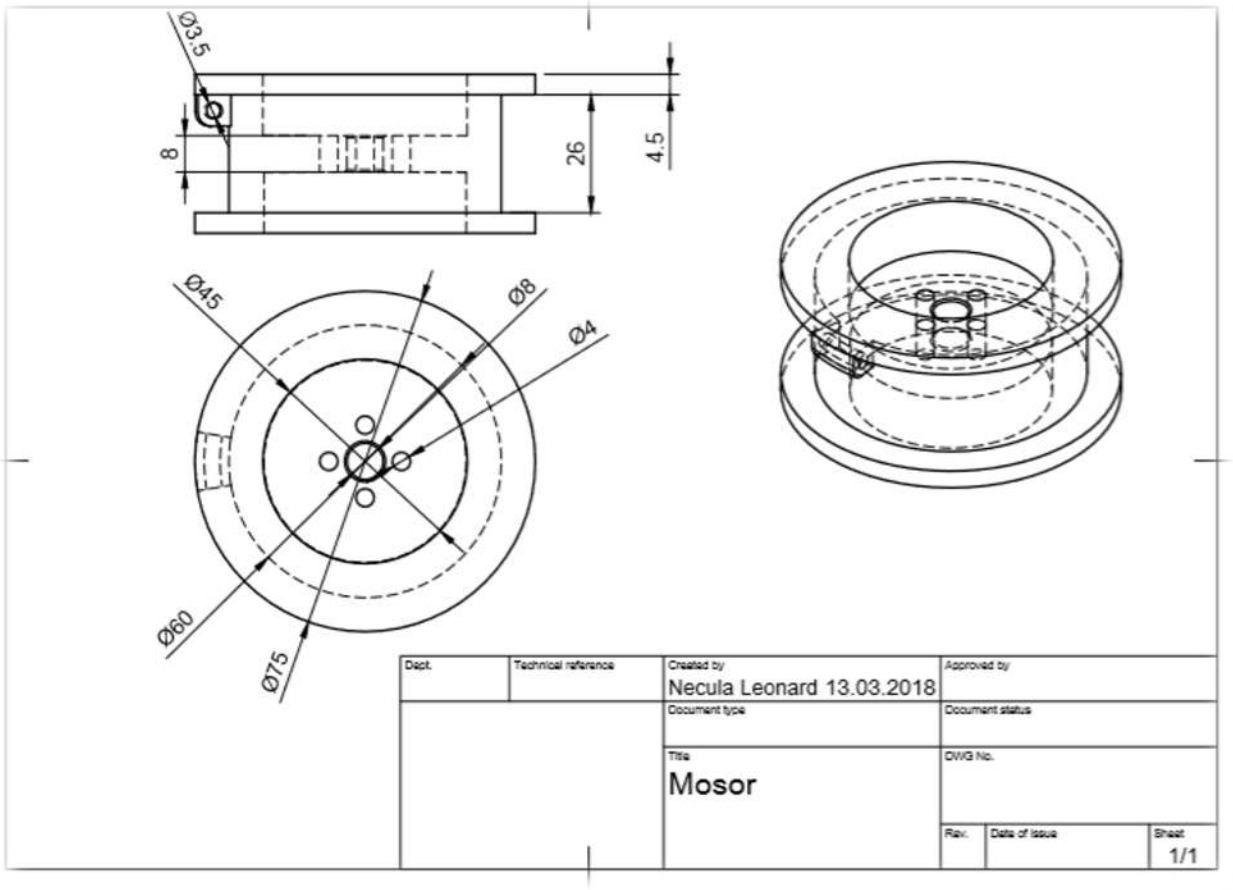

8
40
125
4
45
50
R15
Ø8
Ø3.75
Dept.
Technical reference
Created by
Necula Leonard 2/9/2018
Approved by
Document type
Document status
Title
Brate-jos
DWG No.
Rev.
Date of issue
Sheet
1/1

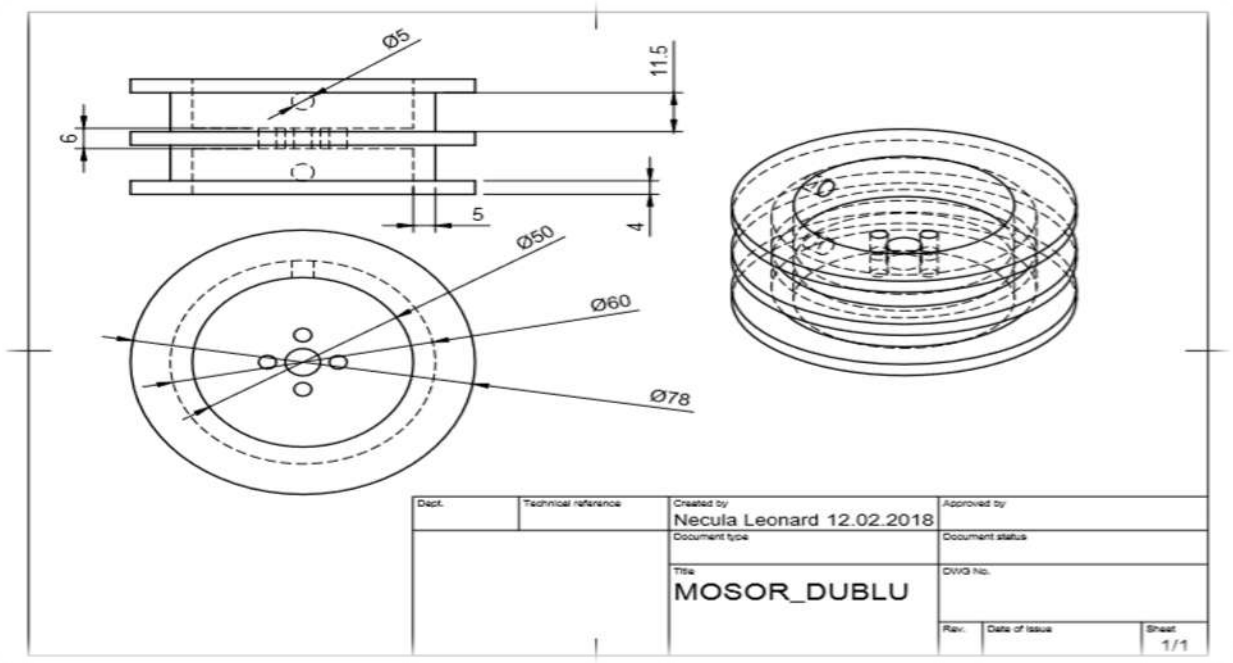

60°
Ø3
9
4.5
3
Dept.
Technical reference
Created by
Necula Leonard 2/7/2018
Approved by
Document type
Document status
Title
Acoperis
DWG No.
Rev.
Date of issue
Sheet
1/1


R7.5
37
4
125
3
20
5
29
16
8
Ø8
Ø6
Ø4
5
58
R10
R4
Dept.
Technical reference
Created by
Necula Leonard 20.02.2018
Approved by
Document type
Document status
Title
Suport Baterie
DWG No.
Rev.
Date of issue
Sheet
1/1


56
4
3
7
32
Ø8
Ø6
Ø4
Ø3.75
Dept.
Technical reference
Created by
Necula Leonard 01.03.2018
Approved by
Document type
Document status
Title
suport range sensor
DWG No.
Rev.
Date of issue
Sheet
1/1


2
4.3
Ø3.75
Ø10
Ø7.6
Dept.
Technical reference
Created by
Necula Leonard 13.03.2018
Approved by
Document type
Document status
Title
pivot
DWG No.
Rev.
Date of issue
Sheet
1/1


Ø3.5
8
26
4.5
Ø45
Ø8
Ø4
Ø60
Ø75
Dept.
Technical reference
Created by
Necula Leonard 13.03.2018
Approved by
Document type
Document status
Title
Mosor
DWG No.
Rev.
Date of issue
Sheet
1/1


Ø5
11.5
6
5
4
Ø50
Ø60
Ø78
Dept.
Technical reference
Created by
Necula Leonard 12.02.2018
Approved by
Document type
Document status
Title
MOSOR_DUBLU
DWG No.
Rev.
Date of issue
Sheet
1/1

• PTC CREO-ROBOT DESING

Creo is a family or suite of Computer-aided design (CAD) apps supporting product design for discrete manufacturers and is developed by PTC. The suite consists of apps, each delivering a distinct set of capabilities for a user role within product development.

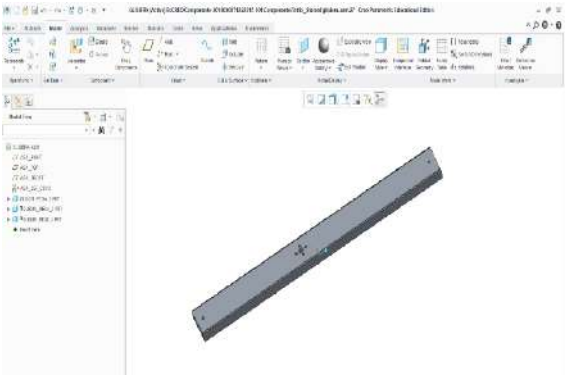
Last year, our team had a virtual model of the robot created in PTC Creo created after completing its construction. Due to the experience we've earned building our robot, we came to the conclusion that, in order to avoid potential mistakes in its' structure, the 3D Virtual design should be done first.

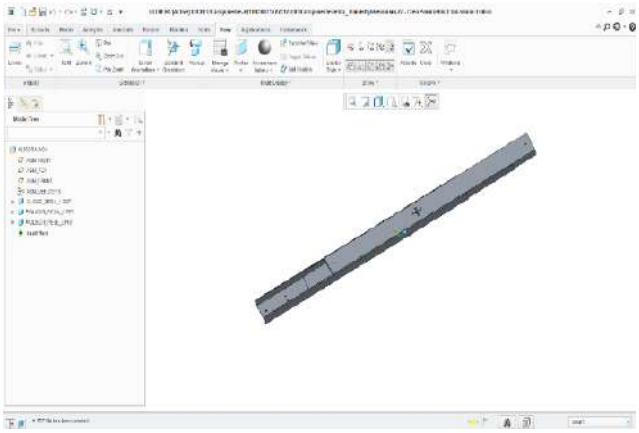
So, after a few days of intense brainstorming, we started developing the basic systems. In this process, the part of the team assigned with construction assisted and helped the part assigned with Creo by providing them with important informations.

- First, our top priority was:

THE WAY WE CAN MANIPULATE THE GLYPHS

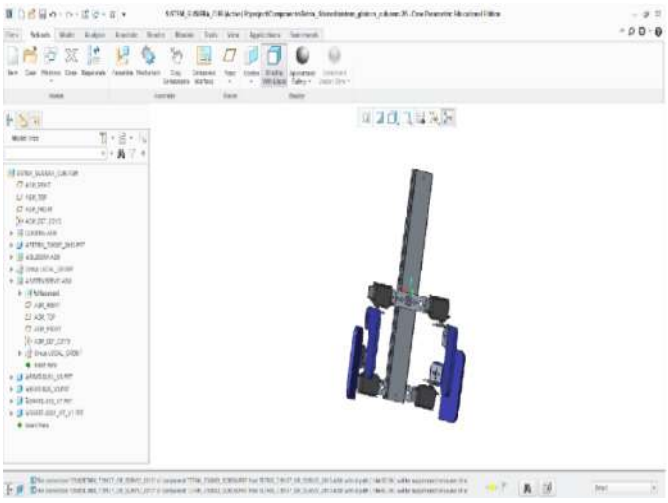
Our idea involved slides:



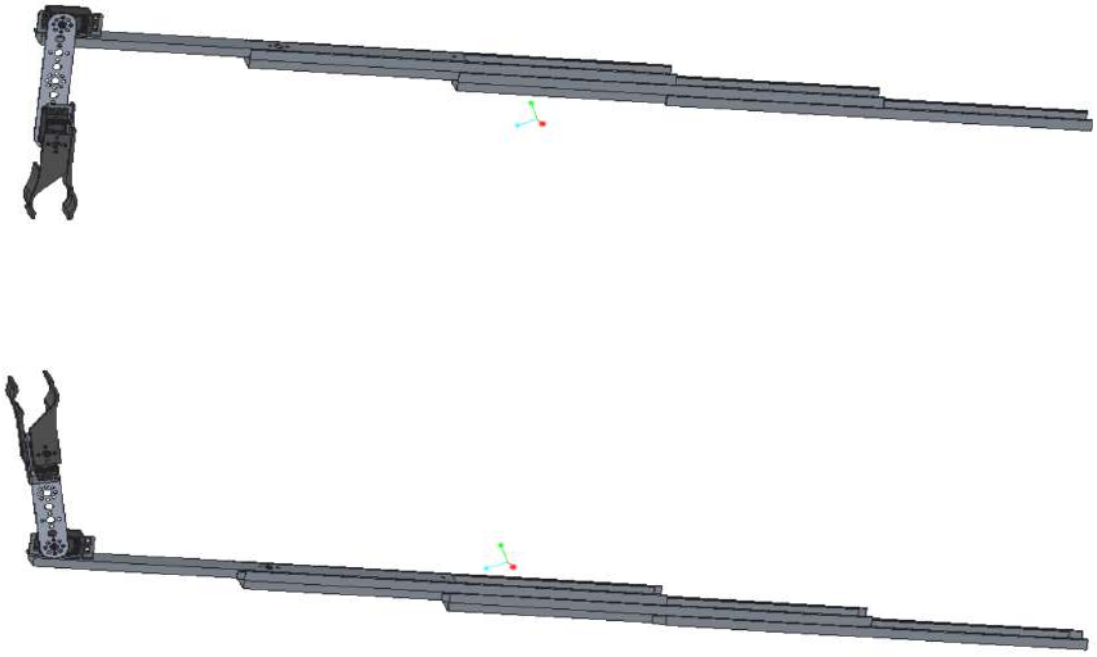


It was a simple mechanism: 2 lateral bars and 2 servo motors utilized to fix and lift the glyph.

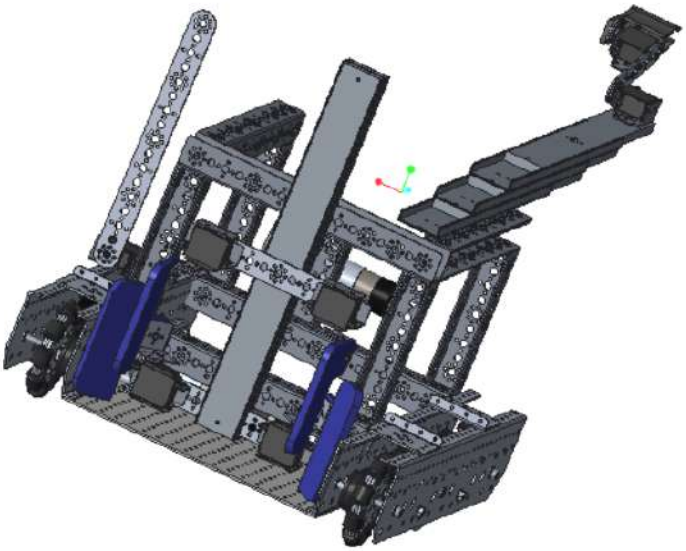
But we stumbled upon a problem: the time required was too long. So, we thought of something else, with almost the same principle, that allowed us to move 2 glyphs independently.



Second, we needed a system for the RELIC part:



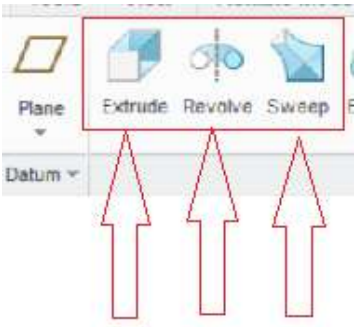
After these 2 important objectives were reached and after a few more touches, our robot began to look like this:



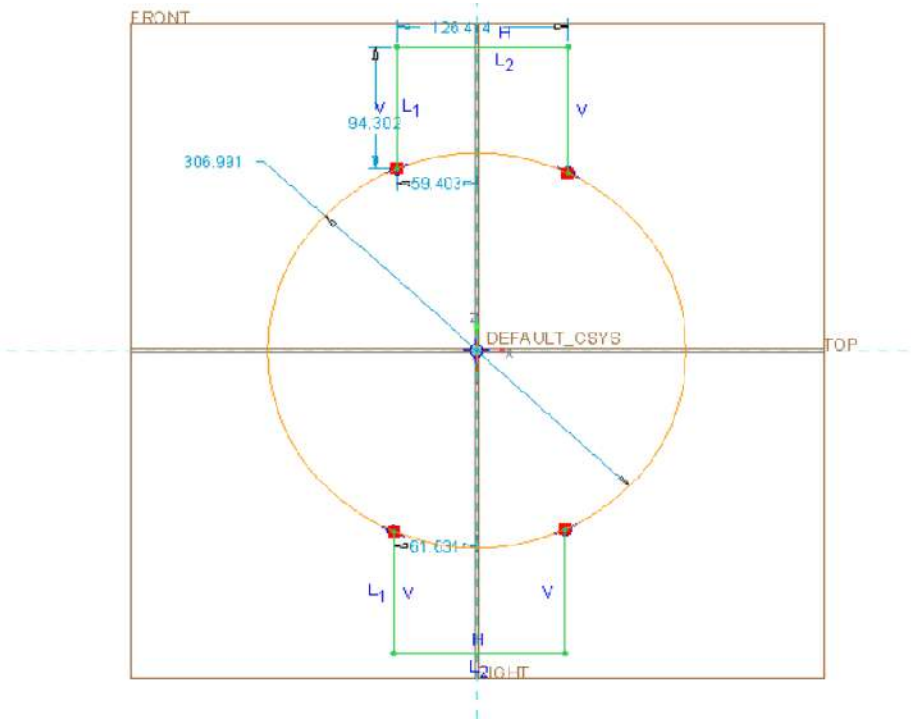
CREATING PARTS

Of course that our team had to design their own virtual parts, that were afterwards brought to reality with the 3D Printer.

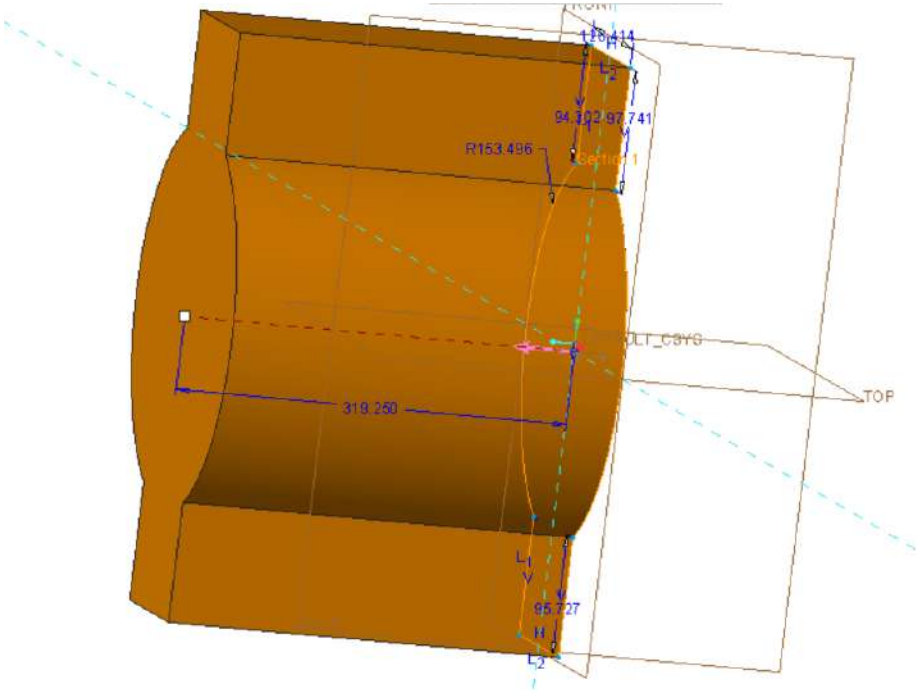
In this process, we utilized Creo's function: Extrude, Revolve and Sweep:



And, after the selection of a plane, Sketch was the tool that allowed us to create:

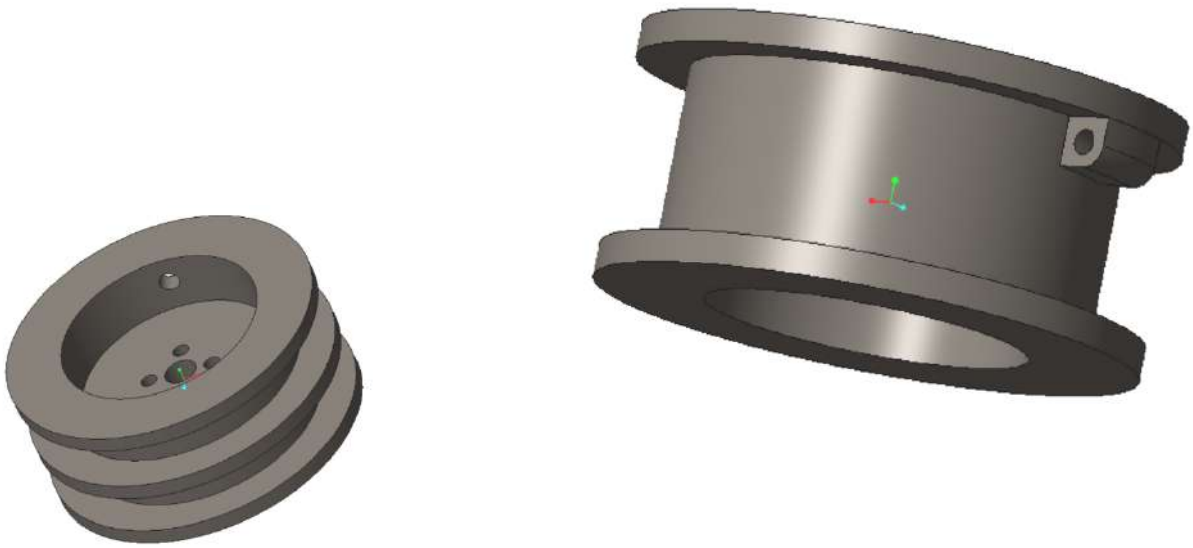


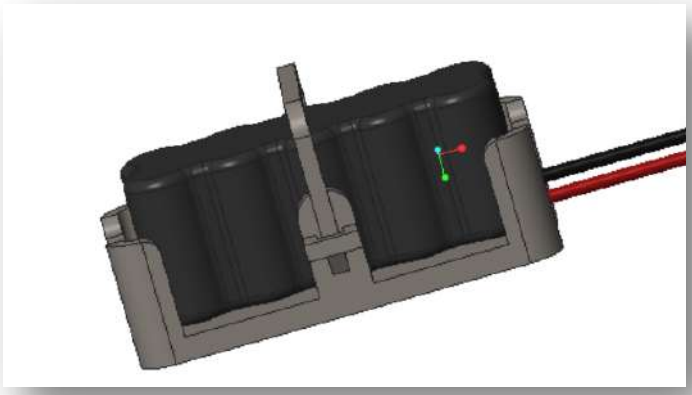
After the sketch was defined, the part received its 3-dimensional form:

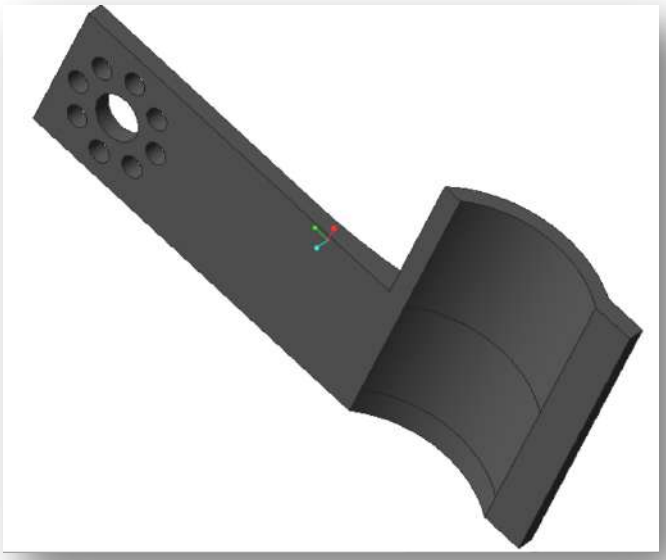


To define a smooth shape we used the function Round.

By these principles, we designed a lot of parts, like the ones bellow:



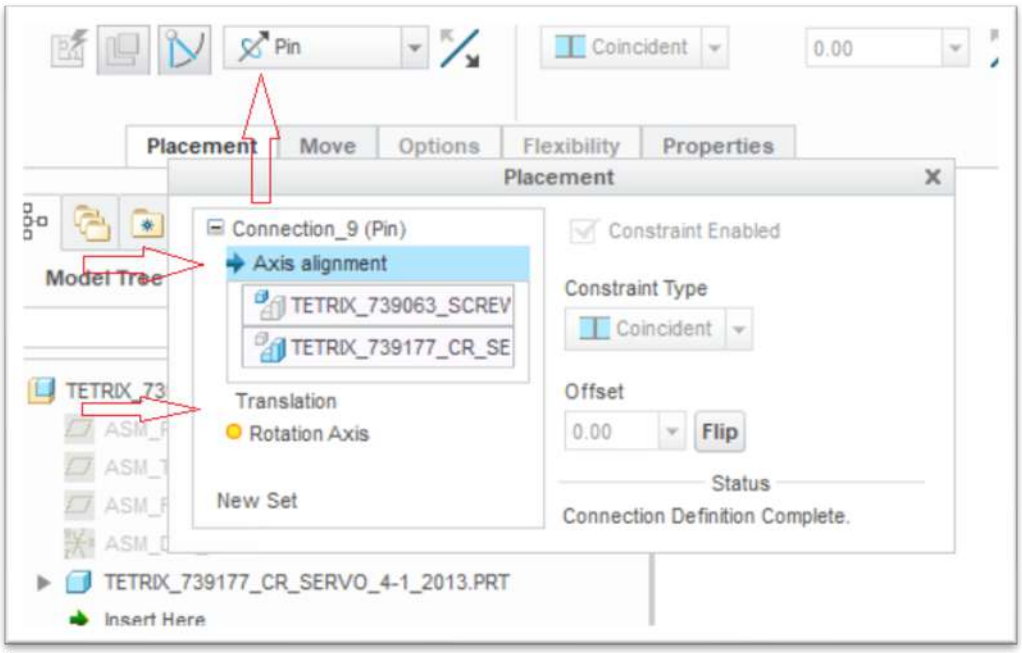


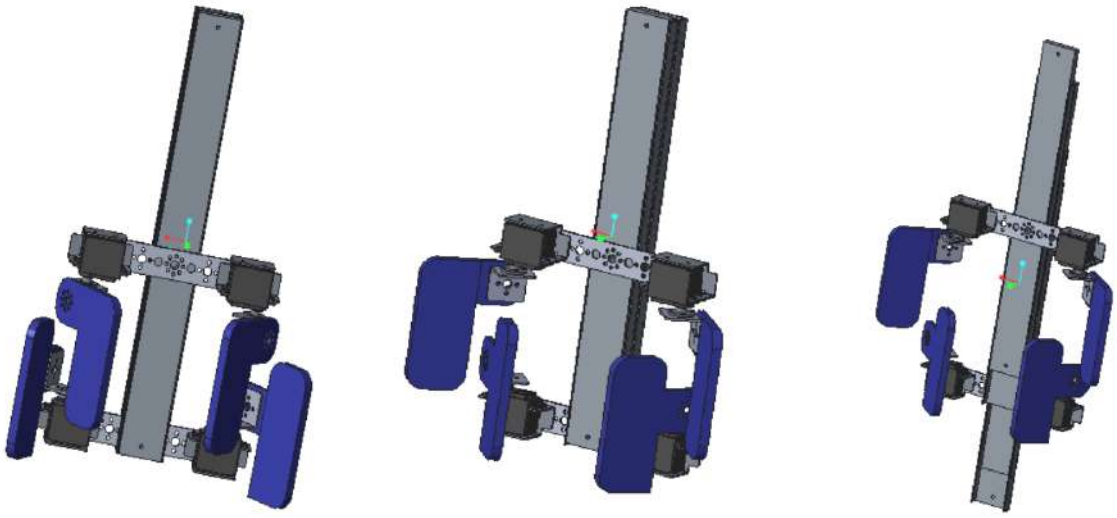


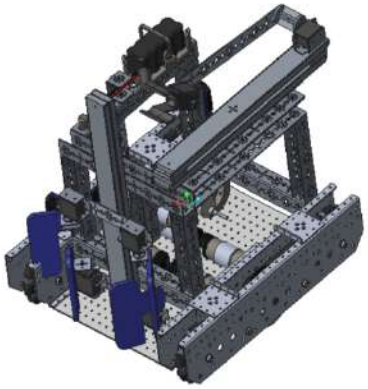
And a lot more.

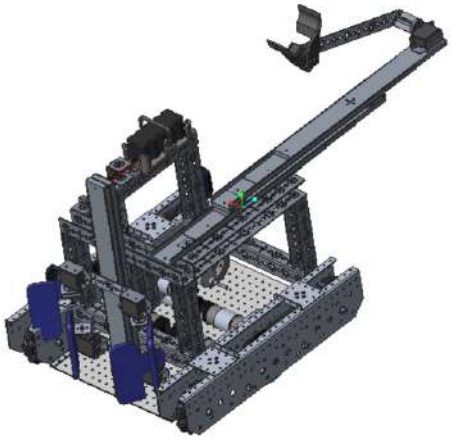
ASSEMBLING

To be able to simulate certain moves, we used motion constraints:







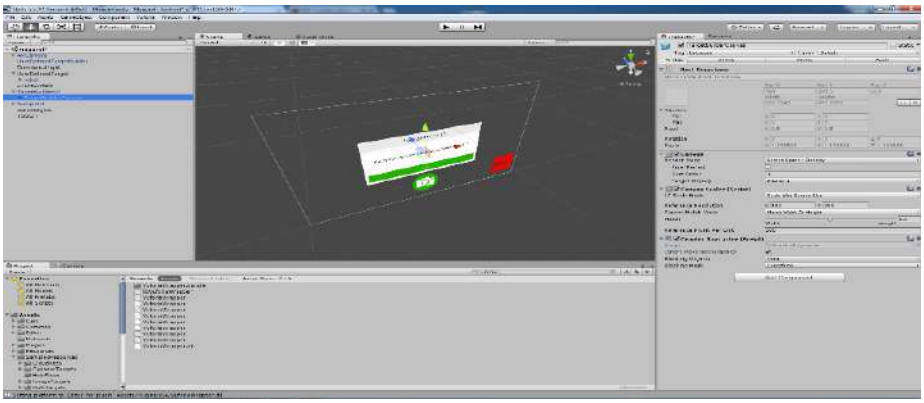


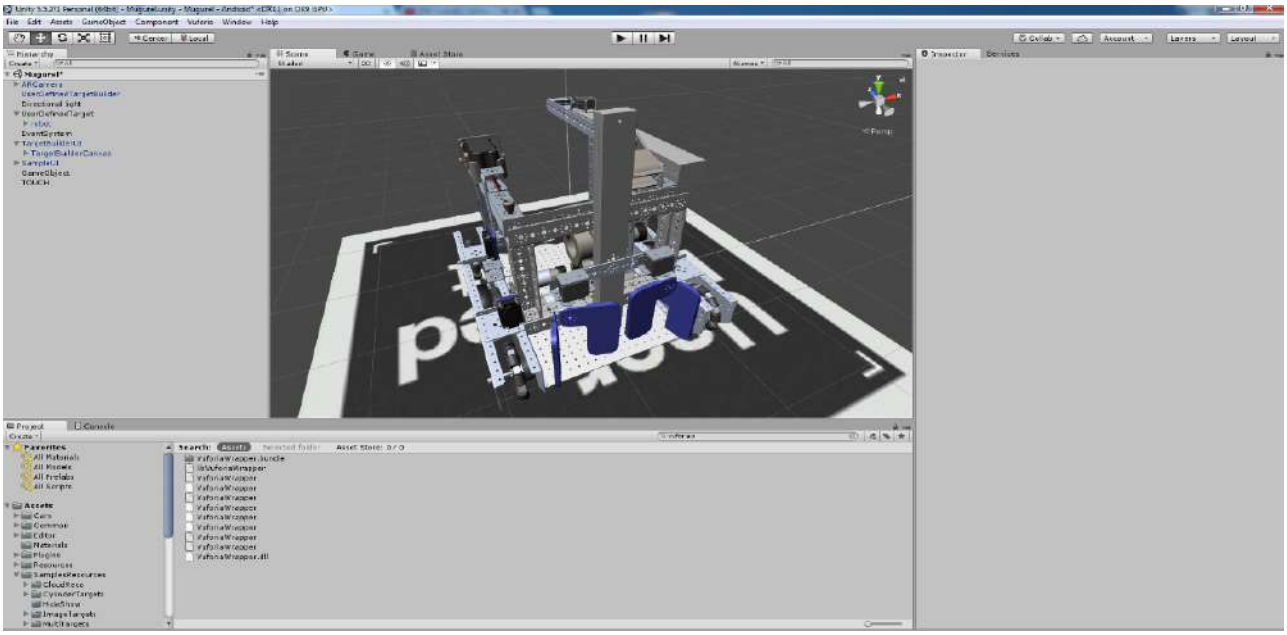
BRINGING OUR ROBOT INTO THE VIRTUAL REALITY

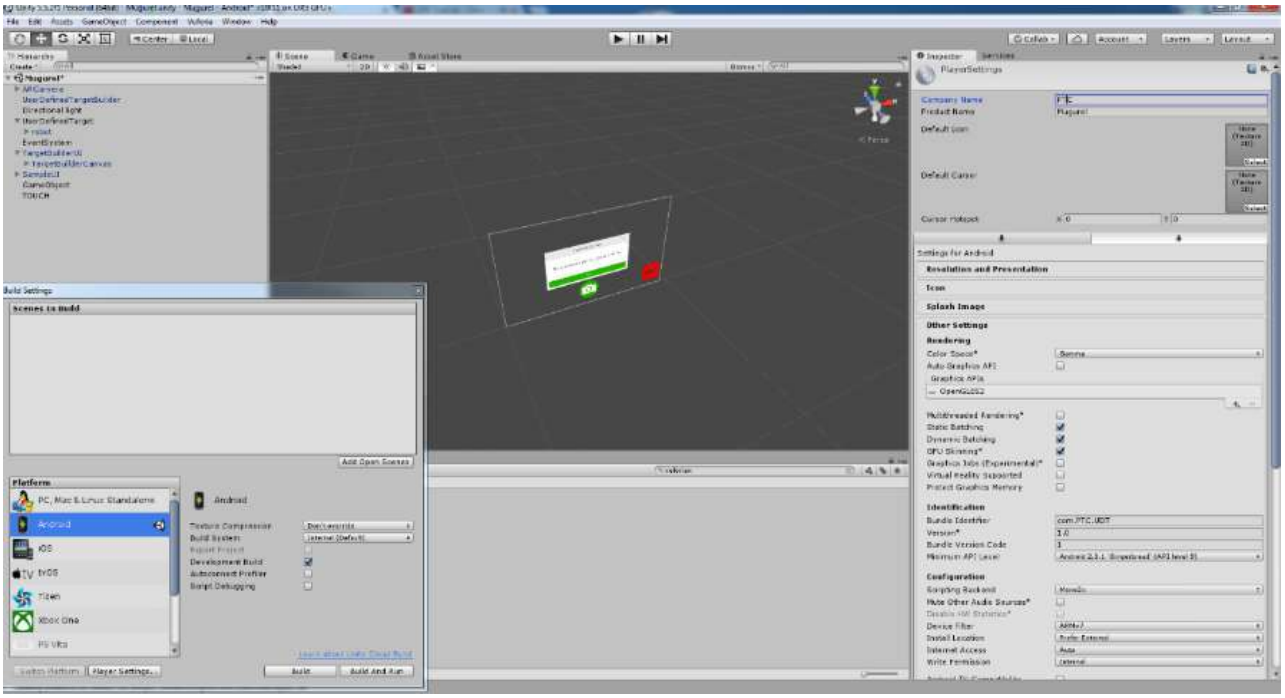
With a few tutorials, our team was able to create an app that can project the robot into the real world.

We used:

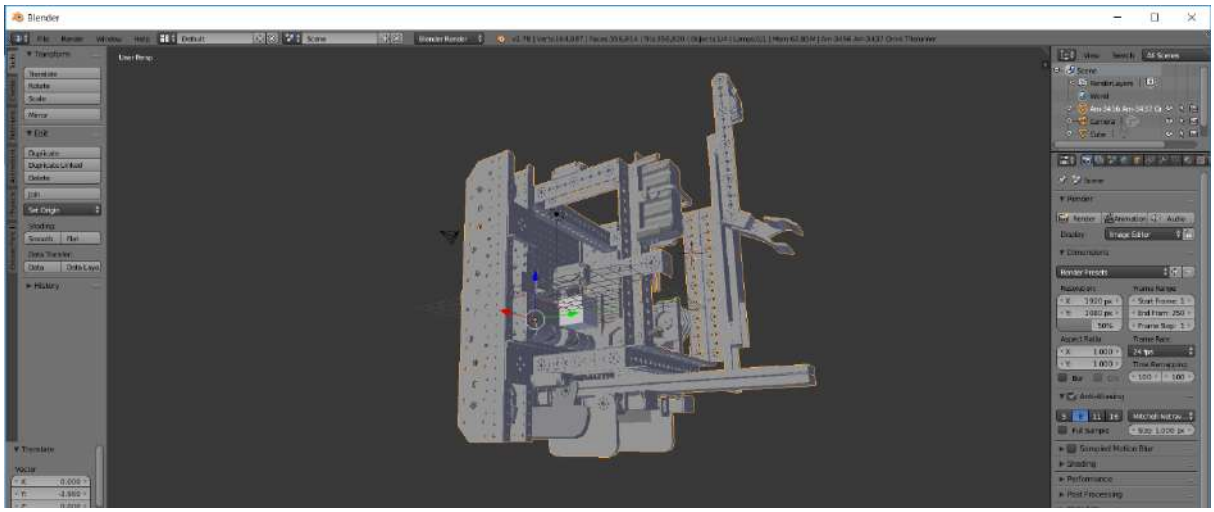
- Unity



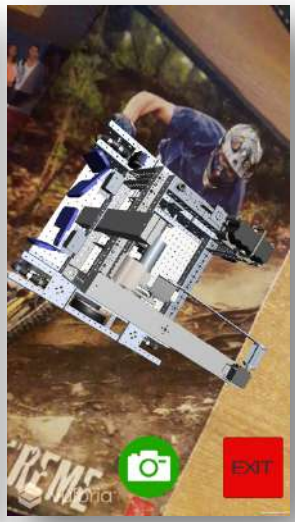


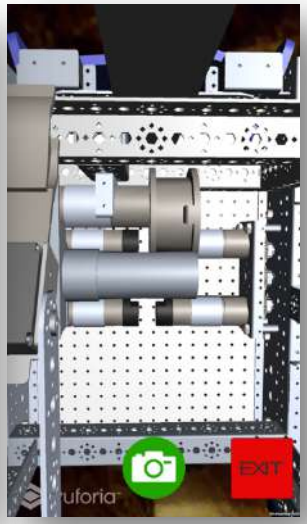


- Blender



We managed to do something amazing. These are the final results:





5.CONSTRUCTION DOCUMENTATION

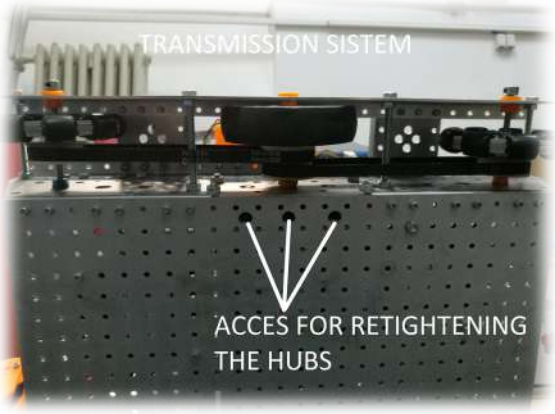
• TILE RUNNER

We had received our tile runner from our sponsors Setro Metal Group. We bought some screws that we needed (7X60 mm) and we printed the necessary adjustments for spindle and the bushings for Tetrix grip. We received belts and printed wheels for them. Then we started the proper construction of the tile runner and that was a very hard process due to the kit pieces which were not compatible with the tile runner.

Before we started the robot construction, the old robot has been disassembly in purpose of reusing the old pieces. We build the rest of the chassis respecting broadly the CREO plan.

At the regional competition we knew that the robot can't get on the balancing stone very well. In this case we used a glyph to put weigh on it, in order to climb it. But that is a problem that can be fixed.

We decided to make a brainstorm in order to get more points of view about our many problems. Then we realized that the 3 points through which the middle shaft must pass aren't collinear which results more friction that makes the robot to work harder.



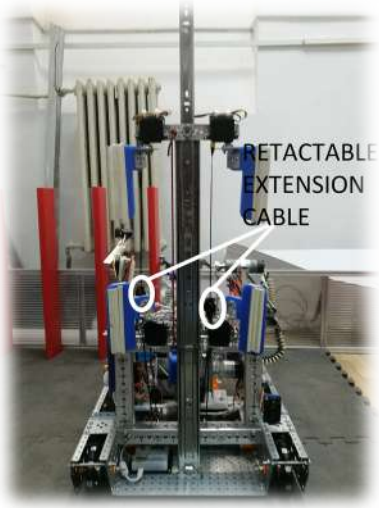
We disassembled the tile runner completely, reprinted gears, tightened every hub with Loctite. We gave access holes in the frame to access the hubs just in case of possible detachments.

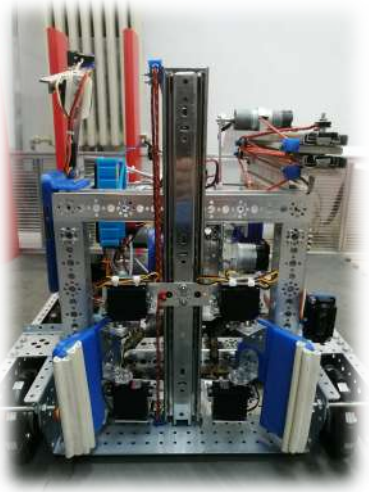
To be able to get on the balancing stone, we realized that we need to cut the tile runner edges such that it touches first the wheels not the chassis. To achieve this we needed help from Mr. Binica because we didn't have the equipment necessary. We retightened the belts and revised the frame of the tile runner.

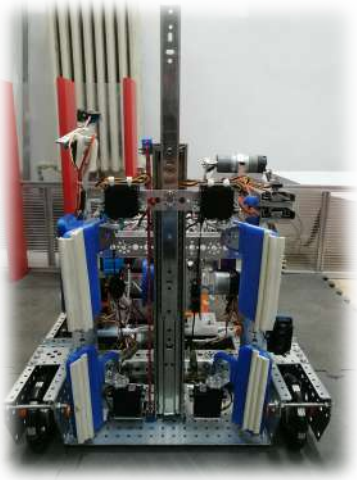
• GLIPHS GRABBER

We started with the first sliding system (linear slider with total extension of 400 mm). We searched and bought bearings (13X6 mm). The ropes that we used are put in a linear system to which we attached 4 servo motors with the 3D printed gliphs grabber. On the gliphs grabber we put cheder for adhesion.

A problem that we have noticed after the regional competition was that, if we bumped into the crypto box to hard, exists the high risk to bend the steel bar of which the servo motors are attached with the gliphs grabber. We didn't find a reasonable solution requiring minimal modification of the ensemble, so we managed to be more careful with the robot.



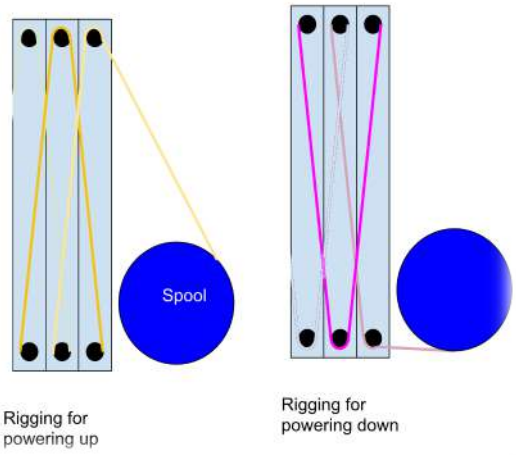




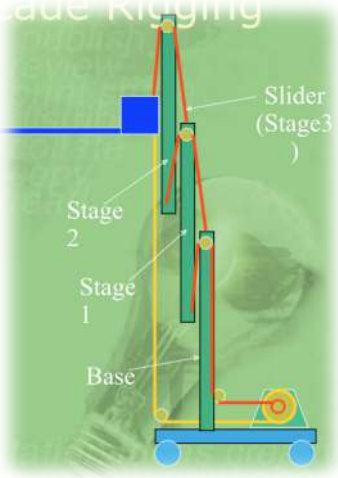
• RELIC GRABBER

At first we didn't know exactly how to proceed in order to retract the sliding system, because it was the first time we needed to build an horizontal sliding system where there isn't no gravitation to take them back in place after they were extended by the first reel.

The initial idea was to use some elastic bends, but we realized that these are not strong enough to keep them in place when the robot is moving and for long term they are not offering steadiness.



Another idea was to fix a rope at the end of the slide to pull it back using a small motor. Instantly we realized that we didn't need a motor since we could use 2 reels on needles, and the reels could have different size based on the position of the ropes. If we use a linear system, the reels must have the same size, but if we use a waterfall system, it must be N times larger, where N is the number of reels. After talking to Bogdan and doing some research on reddit, we found out that the double waterfall system that combines the advantages of the two previous projects. It uses 2 equal slides but is as fast as the normal waterfall system and offers stability to the robot during the game because it's reels keep still. We have decided to use this system for the slides that put on the relic. First we have put together 3 slides without spacers but it was too hard to fix the bearings. For the actual grip of the relic, we have decided to use 2 3D printed arms, a static one and one actioned by a servo. We call this a "claw" and it must be rotated by another servo witch we had bought from Andymark (cuplu 2N). We have also received from the Setro sponsor 4 horizontal slides (8 pieces:45 *30 *10 mm). We have put together 3 of this slides, this time using the spacers we had received and this has helped us a lot with the ropes. After a while wo thought about another ensemble, who has one less slider because we are using a steel bar at the end of which it has one 3D



printed hinge where the claw is attached, who is always coming down due to gravity.

One day before the regional competition, we find out the big servo motor that we had bought wasn't strong enough to pull up the claw, the moment of force being too strong. We tried to minimize it catching the steel bar in the center and adding some little weights in the opposite side of the claw to help the servo to get up. To make this happen we called Mr. Binica and asked if he can help us. When he arrived, we observed that the weights were too heavy. After we discussed we decided to replace the big servo with a motor 40:1 Andymark. We get Mr. Negus and Mr. Sitaru to give us some advices and to help us to expand the motor power wire using phone cable.

At the end of the day the motor had been a much better choice because it had enough force top pull up the claw, but it was too fast and too hard to control. If we put a much lower power in the code the force is small. Unfortunately is the end of the day and is nothing more we can do.

At the regional we tried to pick up the relic with the motor, but we observed that it can't stay still when it gets up the relic, the moment of the force being to big for it. Then we had an idea. We used the 60:1 motor which we used at the vertical slider system. We put the motor in the place of the one who rotates the claw system. Exactly how we thought, it has enough force to pull up the ensemble and it is easy to control because it has a low speed.

Now the ensemble is completely functional.

• JEWELS SYSTEM (FOR AUTONOMOUS PART)

Because we chose to use a tile runner, we can't use a single arm with an end-to-end color sensor that extends to knock down the jewels. So we decided to use 2 servos, one down the whole assembly and the other that acts like an extension of the arm where is attached the color sensor, which is also used to move balls without having to rotate the robot completely. we bought some very small servos to use them, but after we assembled them, we realized that the one who pulls down the ensemble is far too weak and easy so I replaced it with a servo motor from the Tetrix kit, which is a bit more resistant.

At the regional robot inspection, we were informed that we needed the documentation required for this type of servo to prove it was compatible.

After the regional, we decided to move the big servo that we used to raise the relic claw instead of the Tetrix that pull down the ensemble, and to use it in place of the small servo, to avoid any compatibility problems.

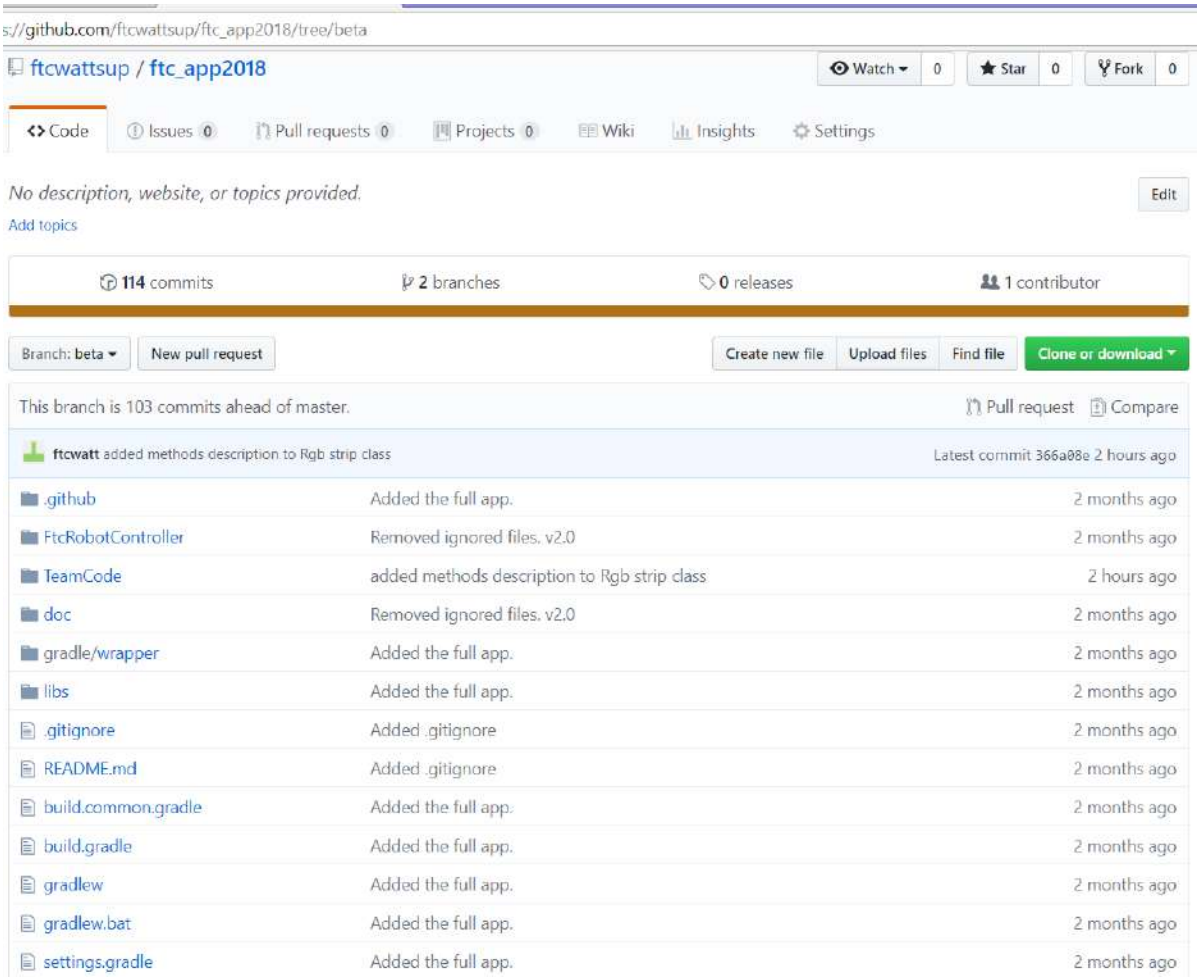
A final retouch was to stick a small sponge on the edge of the arm to reduce the impact with the floor caused by inertia.

6.PROGRAMMING

• PROGRAMMING ENTRIES

1) GITHUB ACCOUNT

Bogdan Sitaru created our Github account, and he also added the FTC Project. Then, our two programmers (Bogdan Sitaru an Milea Octavian) started syncing Android Studio with Github. This is how it looks now:



2) RUNNER

Runner class is where our programmers implemented all of the methods used for moving the robot on the field. By using a constructor they can pass all of the values when an object is created, so now they can call a move function from any other class.

Here can be seen some of the methods implementations in this class:

```
/**
 * Runner class is used for controlling the moving system of the robot #Mugurel
 */
public class Runner {
    /**
     * leftF = front left motor
     * leftB = back left motor
     * rightF = front right motor
     * rightB = back right motor
     */
    private DcMotor leftF, leftB, rightF, rightB;

    /**
     * telemetry = telemetry item assigned to the moving system
     * verbose = true if any telemetry item is assigned to this class
     */
    private Telemetry.Item telemetry;
    private boolean verbose = false;
```

```
/**
 * Checks for errors before initialization
 */
private void initialCheck()
{
    if( leftF.getController() != leftB.getController() )
    {
        if(verbose) telemetry.setValue("Different controller for LEFT motors!");
        error = 1;
    }

    if( rightF.getController() != rightB.getController() )
    {
        if(verbose) telemetry.setValue("Different controller for RIGHT motors!");
        error = 2;
    }

    if(error == 0 && verbose)  telemetry.setValue("OK!");
}
```

```
/**
 * Moves robot according to gamepad axis Y(forward-backward), X(rotation) and scale ratio R(power ratio)
 * @param y forward-backward axis
 * @param x rotation axis
 * @param r scale ratio
 */
public void move(double y, double x, double r)
{
    rnrX = x; rnrY = y; rnrR = r;

    if(x == 0)
        setPower(-y, y, r);
    else if(y == 0)
        setPower(-x, -x, r);
    else if(x < 0)
        setPower(-y - y * x, y, r);
    else if(x > 0)
        setPower(-y, y - y * x, r);
}
```

```
    /**
     * Moves robot cm centimeters forward (> 0) or backward (< 0)
     * @param cm number of centimeters
     */
    public void distanceMove(double cm, double power)
    {
        double addTicks = cm / wheelLength * ticksPerRevolution;
        int add = (int)Math.round(addTicks);

        DcMotor.RunMode oldMode = leftF.getMode();

        setMode(DcMotor.RunMode.STOP_AND_RESET_ENCODER);
        setMode(DcMotor.RunMode.RUN_TO_POSITION);

        leftF.setTargetPosition( -add );
        leftB.setTargetPosition( -add );
        rightF.setTargetPosition( add );
        rightB.setTargetPosition( add );

        power = Math.abs(power);
        setPower(power, power);

        while( leftF.getPower() > 0.0 || rightF.getPower() > 0.0 )
        {
            if(!leftF.isBusy() || !leftB.isBusy())
            {
                leftF.setPower(0.0);
                leftB.setPower(0.0);
            }
            if(!rightF.isBusy() || !rightB.isBusy())
            {
                rightF.setPower(0.0);
                rightB.setPower(0.0);
            }
        }

        setPower(0.0, 0.0);

        setMode(oldMode);
```

```
/**
 * Moves robot according to gamepad axis Y(forward-backward), X(rotation) and no scale ratio
 * @param y forward-backward axis
 * @param x rotation axis
 */
public void move(double y, double x)
{
    move(y, x, 1.0);
}
```

```
/**
 * Set power for all motors, for left and right sides
 * @param left left side power
 * @param right right side power
 * @param ratio scale ratio
 */
public void setPower(double left, double right, double ratio)
{
    left *= ratio; right *= ratio;
    leftF.setPower(left); leftB.setPower(left);
    rightF.setPower(right); rightB.setPower(right);
}
```

```
/**
 * Set the run mode for all motors
 * @param mode the new RunMode
 */
public void setMode(DcMotor.RunMode mode)
{
    leftF.setMode(mode);
    leftB.setMode(mode);
    rightB.setMode(mode);
    rightF.setMode(mode);
}
```

```
    * Set power for all motors, for left and right sides, without any scale ratio
    * @param left left side power
    * @param right right side power
    */
   public void setPower(double left, double right)
   {
       setPower(left, right, 1.0);
   }
```

3) RGB STRIP

RGB Strip class is used by our programmers to control the LED's mounted on our robot. Like the Runner class, methods in this class can be called, by creating an object, so these can be accessed from a different class too. Here you can find the implementation of these methods:

```
/**
 * RGBStrip class is used to control the RGB strip attached to robot. :) (#Duta)
 */
public class RGBStrip
{
    // Initialization of all objects

    private DcMotor  red, green, blue;
    private Telemetry.Item telemetry;
    private boolean verbose = false;
    private ModernRoboticsUsbDcMotorController controller;

    /**
     * Constructor
     * @param _red :  represents the Motor controller port where the RGB Strip cathode corresponding to green is connected.
     * @param _blue :  represents the Motor controller port where the RGB Strip cathode corresponding to blue is connected.
     * @param _green :  represents the Motor controller port where the RGB Strip cathode corresponding to green is connected.
     */
    RGBStrip(DcMotor _plus, DcMotor _red, DcMotor _green, DcMotor _blue, Object... _telemetry)
    {
        red = _red; green = _green; blue = _blue;
        red.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
        green.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
        blue.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
        controller = (ModernRoboticsUsbDcMotorController) red.getController();
        if(_telemetry.length > 0 && (_telemetry[0] instanceof Telemetry.Item))
        {
            verbose = true;
            telemetry = (Telemetry.Item) _telemetry[0];
        }
        else
            verbose = false;
    }
```

```
//Turns off the RGB Strip


public void off()
{
    red.setPower(-0.01);
    green.setPower(-0.01);
    blue.setPower(-0.01);
}


 //Set those parameters to the RGB values from the colors you need.

public void setColor(int r, int g, int b)
{
    if( (r == 0 && g == 0 && b == 0) || controller.getVoltage() < 11.0 )
    {
        off();
        return;
    }

    double rPower = (double)r / (double)255;
    double gPower = (double)g / (double)255;
    double bPower = (double)b / (double)255;

    if(r == 0)  rPower = -0.01;
    if(g == 0)  gPower = -0.01;
    if(b == 0)  bPower = -0.01;

    red.setPower(rPower);
    green.setPower(gPower);
    blue.setPower(bPower);
}



 // Displays Telemetry

 public void logInformation()
 {
     telemetry.setValue( String.format("%.3f", red.getPower()) + " " +
             String.format("%.3f",green.getPower()) + " " + String.format("%.3f", blue.getPower()) );
 }
```

4) RELIC GRABBER

Relic Grabber class is used by our programmers to control the relic grabbing and placing mechanism. Its methods are also accessible from any other class by passing values through constructor when the object is being initialized.

This is how the methods are implemented:

```
public class RelicGrabber
{
    /**
     * 0. monster = big servo
     * 1. claw = small servo attached to the big servo
     */
    private Servo claw;
    /**
     * pusher = push motor
     */
    private DcMotor monster, pusher;

    /**
     * telemetry = telemetry item assigned to the moving system
     * verbose = true if any telemetry item is assigned to this class
     */
    private Telemetry.Item telemetry;
    private boolean verbose = false;

    /**
     * monsterP = constant values of positions for the big servo
     * clawP = constant values of positions for the small servo
     * pushP = constant values to limit the actions for push motor
     */
    private double monsterP[] = {0.0, 1.0};
    private double clawP[] = {0.4, 0.15};
    private int pushP[] = {0, (int)(1e6)};
    /// TODO: Get constant values
```

```
 *  stateMonster = 0 -> down, 1 -> up
 *  stateClaw = 0 -> open, 1 -> closed
 */
private int stateMonster = 0, stateClaw = 0;

/**
 * Constructor
 * @param _monster big servo
 * @param _claw small servo
 * @param _pusher push motor
 * @param _telemetry OPTIONAL -> telemetry item
 */
RelicGrabber(DcMotor _monster, Servo _claw, DcMotor _pusher, Object... _telemetry)
{
    monster = _monster; claw = _claw; pusher = _pusher;
    monster.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);

    if(_telemetry.length > 0 && (_telemetry[0] instanceof Telemetry.Item))
    {
        telemetry = (Telemetry.Item) _telemetry[0];
        verbose = true;
        telemetry.setValue("OK!");
    }
    else
        verbose = false;

    pusher.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
    monster.setMode(DcMotor.RunMode.STOP_AND_RESET_ENCODER);
    monster.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
    //claw.setPosition(clawP[0]);
}
```

```
/**
 * PRIVATE
 * Changes position of a servo adding val to the current position
 * @param servo the changing servo
 * @param val added value
 */
private void addValue(Servo servo, double val)
{
    double p = servo.getPosition();
    p += val;
    p = Math.max(0.0, p);
    p = Math.min(1.0, p);
    servo.setPosition(p);
}

/**
 * Changes position of a servo adding val to the current position
 * @param servo id of the changing servo
 * @param val added value
 */
public void addValue(int servo, double val)
{
    if(servo == 1)  addValue(claw, val);
}
```

```
/**
 * Set power of push motor
 * @param power new power to set
 */
public void movePusher(double power)
{
    //if(pusher.getCurrentPosition() < pushP[0])   { pusher.setPower(0); return; }
    //if(pusher.getCurrentPosition() > pushP[1])   { pusher.setPower(0); return; }
    pusher.setPower(power);
}

    /**
     * Set run mode for push motor
     * @param rm new RunMode to set
     */
    public void setPusherMode(DcMotor.RunMode rm) { pusher.setMode(rm); }

    /**
     * Change state of claw
     */
    public void changeStateClaw()
    {
        stateClaw ^= 1;
        claw.setPosition(clawP[stateClaw]);
    }
/**
 * Enables/Disables power in all servos in this servo controller.
 * @param val 0 = disabled, 1 = enabled
 */
public void setPower(int val)
{
    if(val == 0)    claw.getController().pwmDisable();
    if(val == 1)    claw.getController().pwmEnable();
}

public void powerMonster(double power)
{
    monster.setPower(power);
}
```

5) AUTONOMOUS FUNCTION

This class was designed by our programmers as a place where all of the methods utilized in the autonomous program should be found. This class is abstract, so the methods can be accessible from anywhere.

```
public abstract class AutonomousFunctions extends LinearOpMode {

    protected ElapsedTime runtime = new ElapsedTime();
    /// Runner and Collector
    protected Runner rnr;
    protected CubeCollector collector;

    /// Sensors
    protected ModernRoboticsI2cGyro gyro;
    protected ModernRoboticsI2cRangeSensor dist_r,dist_f,dist_s;
    protected ModernRoboticsI2cCompassSensor compass;
    protected ModernRoboticsI2cColorSensor colorSensor;

    /// Servo
    protected Servo extender, rotor;


    /// Variables
    protected static int forward = 0;
    protected static int color = 0;    /// Red = 0, Blue = 1
    protected int nr=0;
    /// Vuforia
    protected VuforiaLocalizer vuforia;

    /// Compass + Accelerometer
    protected double fXg = 0;
    protected double fYg = 0;
    protected double fZg = 0;
```

```
protected void initialization()
{
    telemetry.setAutoClear(false);

    state = telemetry.addData("State", "init");

    Telemetry.Item timeTelemetry = telemetry.addData("Time", 0);

    Telemetry.Item runnerTelemetry = telemetry.addData("Runner", 0);
    rnr = new Runner(
            hardwareMap.get(DcMotor.class, "frontleft"),
            hardwareMap.get(DcMotor.class, "backleft"),
            hardwareMap.get(DcMotor.class, "frontright"),
            hardwareMap.get(DcMotor.class, "backright"),
            runnerTelemetry
    );

    Telemetry.Item collectorTelemetry = telemetry.addData("Collector", 0);
    collector = new CubeCollector(
            hardwareMap.get(Servo.class, "upleft"),
            hardwareMap.get(Servo.class, "upright"),
            hardwareMap.get(Servo.class, "downleft"),
            hardwareMap.get(Servo.class, "downright"),
            hardwareMap.get(DcMotor.class, "lifter"),
            collectorTelemetry
    );

    gyroTelemetry = telemetry.addData("Gyro", "Not initialized");
    gyro = hardwareMap.get(ModernRoboticsI2cGyro.class, "gyro");
    gyroTelemetry.setValue("Calibrating...");
    gyro.calibrate();
    while (!isStopRequested() && gyro.isCalibrating()) {
        gyroTelemetry.setValue("Calibrating...");
        telemetry.update();
        sleep(50);
    }
    gyroTelemetry.setValue("Calibrated!");
```

```
/**
 * Gets key drawer with Vuforia
 * LEFT = 1
 * CENTER = 2
 * RIGHT = 3
 * UNKNOWN = 2
 *
 * @return key drawer
 */
protected int getKeyDrawer() {

    int cameraMonitorViewId = hardwareMap.appContext.getResources().getIdentifier("cameraMonitorViewId", "id", hardwareMap.appContext
    VuforiaLocalizer.Parameters parameters = new VuforiaLocalizer.Parameters(cameraMonitorViewId);
    parameters.vuforiaLicenseKey = "AT4pxIn/////AAAAGSL31MQ2OkEAlE8fAGFaXLprsCGiwn5e81ZDhALBFUMK45pVGTrKvkFV9ZBC8LGo2fubVHcyc2JBpk2bs
    parameters.cameraDirection = VuforiaLocalizer.CameraDirection.FRONT;
    this.vuforia = ClassFactory.createVuforiaLocalizer(parameters);
    VuforiaTrackables relicTrackables = this.vuforia.loadTrackablesFromAsset("RelicVuMark");
    VuforiaTrackable relicTemplate = relicTrackables.get(0);
    relicTemplate.setName("relicVuMarkTemplate"); // can help in debugging; otherwise not necessary
    relicTrackables.activate();
    RelicRecoveryVuMark vuMark = RelicRecoveryVuMark.from(relicTemplate);

    if (vuMark == RelicRecoveryVuMark.LEFT) return 1;
    if (vuMark == RelicRecoveryVuMark.CENTER) return 2;
    if (vuMark == RelicRecoveryVuMark.RIGHT) return 3;
    return 2;
}
```

```
151     // CALL THIS METHOD FOR SCORING JEWELS
152     protected void scoreJewels(Servo extender)
153     {
154         rotor.setPosition(0.5);
155         sleep(750);
156         extender.setPosition(0.33);
157         sleep(1000);
158
159         colorSensor.enableLed(true);
160         int clr = 0;
161         if(colorSensor.red() > colorSensor.blue())  clr = 0;
162         else if(colorSensor.red() == colorSensor.blue())    clr = -1;
163         else    clr = 1;
164
165         if(clr == -1)
166         {
167             rotor.setPosition(0.45);
168             extender.setPosition(0.35);
169             sleep(200);
170             if(colorSensor.red() > colorSensor.blue())  clr = 0;
171             else if(colorSensor.red() == colorSensor.blue())    clr = -1;
172             else    clr = 1;
173         }
174
175         telemetry.addData("Red", colorSensor.red());
176         telemetry.addData("Blue", colorSensor.blue());
177         telemetry.update();
178
179         sleep(400);
180
181         if(clr == color)    rotor.setPosition(1.0);
182         else if(clr != -1)    rotor.setPosition(0.0);
183         //colorSensor.enableLed(false);
184
185         sleep(1000);
186         extender.setPosition(1);
187         sleep(500);
188         rotor.setPosition(0);
```

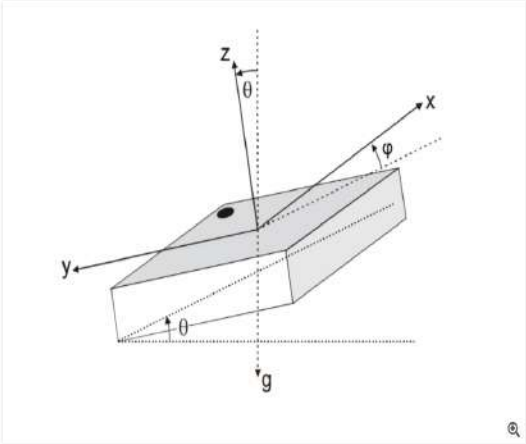
One of the challenging situation was represented by the way our robot should determine if it got down from the balancing stone, so or programmers did some research and found out how to convert x,y,z acceleration into pitch.
Here is the theory:
https://www.dfrobot.com/wiki/index.php/How_to_Use_a_Three-Axis_Accelerometer_for_Tilt_Sensing
And some related Geometry:

$$\tan\phi_{xyz} = \left(\frac{G_{py}}{G_{pz}}\right)$$

$$\tan\theta_{xyz} = \left(\frac{-G_{px}}{\sqrt{G_{py}^2 + G_{pz}^2}}\right)$$

Tilt Angle

We have already studied the Yaw-Pitch-Roll angle, they could tell how to rotate an object by its X - Y axis to a specified posture. Actually, there is another method to expree the exact spatial position, that is Tilt angle showing in the picture below. How to understand the relationship between them? You may came to think the Cartesian coordinates and Polar coordinates, that's it.



And the estimation of the angles is (From Bosch Sensortec):

```
acc_x = 1g * sinθ * cosϕ

acc_y = - 1g * sinθ * sinϕ

acc_z = 1g * cosθ

⇒ acc_y/acc_x = - tanϕ
```

And this is the implementation of the method:

```
// THIS METHOD RETURNS THE PITCH RELATIVE TO THE HORIZONT LINE, FROM THE ACCELEROMETER SENSOR
protected double getPitch()
{
    double alpha = 0.5;
    Acceleration acc = compass.getAcceleration();

    double X = acc.xAccel;
    double Y = acc.yAccel;
    double Z = acc.zAccel;

    double Xg, Yg, Zg;
    Xg = X;
    Yg = Y;
    Zg = Z;

    fXg = Xg * alpha + (fXg * (1.0 - alpha));
    fYg = Yg * alpha + (fYg * (1.0 - alpha));
    fZg = Zg * alpha + (fZg * (1.0 - alpha));

    double pitch = ( Math.atan2(-fXg, Math.sqrt(fYg * fYg + fZg * fZg)) * 180.0 ) / Math.PI;
    return pitch;
}
```

Ok, so know our accelerometer sensor (Modern Robotics compass sensor) can tell our robot's pitch, the only problem we had was to get the robot down relative to this information. What our programmers did was to scale the power. of the motors to (InitialPitch – ActualPitch)*KP(where kp is somewhere about 0.16).

Here is the implementation of the get down method which does all of these:

Here is the implementation of the get down method which does all of these:

Here is the implementation of the get down method which does all of these:

```
// CALL THIS METHOD TO GET DOWN FROM PLATFORM
protected void getDown()
{
    double power = 0.3;
    double heading = getPitch();
    double deg1 = 2.5;
    double okDegrees = 1;
    //rnr.setPower(-power * forward, power * forward);
    double last_dist = dist_r.getDistance(DistanceUnit.CM);
    int step = 0;

    while( nr == 0 )
    {

        double currentHeading = getPitch();
        compassTelemetry.setValue(currentHeading);
        telemetry.update();
        double dist = dist_r.getDistance(DistanceUnit.CM);
        //sleep(200);
        double dif = heading - currentHeading;
        // This lines makes a proportional gain for the motors power relative to the robot pitch( inclanation ),
        // relative to the horizont. Because the difference between initial value and the current value is dropping to a point an then climbing,
        //we need to check and see where we are in this interval, so that we can scale the motors power like an Monotonic descending function
        if(step == 0)
        {
            if(dif >= deg1)  step++;
            rnr.setPower(-power, power, forward);
        }
        else
        {
            if(dif <= okDegrees)
            {
                rnr.setPower(0.0, 0.0);
                Keep_Orientation(0);
                break;
            }
                    break;
                }
                //still looking for the drawer, just in case that robot gets stuck in this function
                if( last_dist - dist >=7 )
                    nr ++;
                double pw = power * dif * 0.16;
                rnr.setPower(-pw, pw, forward);
            }
            last_dist = dist;
            if(!opModeIsActive())   return;
        }
}
```

```
// THIS METHOD CONVERTS FROM INTEGRATED Z VALUE TO HEADING ( THE OANE RETURNED BY DEFAULT IS NOT CORRECT)
protected double getGyroHeading()
{
    double angle = gyroGetIntegratedZ();
    while(angle < 0)    angle += 360;
    while(angle >= 360)  angle -= 360;
    return angle;
}
```

```
// CALL THIS METHOD TO GRAB CUBE
protected void grab_cube()
{
    collector.closeArms(1);
    sleep(100);
    collector.moveLift(0.5);
    sleep(300);
    collector.moveLift(0.0);
}
/**
 *THIS METHOD ESTABLISHES HOW MUCH OF A DISTANCE DOES THE ROBOT NEEDS TO GO AFTER SEEING THE FIRST DRAWER,TO PLACE ITSELF
 IN FRONT OF THE NEEDED DRAWER
 * @PARAM nr: VALUE RETURNED FROM getKeyDrawer()
 */

protected void pick_drawer(int nr) {
    if (nr == 1)
        rnr.distanceMove(6 * forward, 0.3, this);
    if (nr == 2)
        rnr.distanceMove(25.5 * forward, 0.3, this);
    if (nr == 3)
        rnr.distanceMove(44 * forward, 0.3, this);
    if( !opModeIsActive() ) return;

}
```

```
/**
 * KEEPS THE ROBOT TO A DESIRED ORIENTATION
 * @PARAM OPTIMAL_POS : THE DESIRED ORIENTATION (IN DEGREES)
 */
protected void Keep_Orientation(double Optimal_pos)
{
    double okDegrees = 2;

    while (true)
    {
        double heading = getGyroHeading();

        gyroTelemetry.setValue(heading);
        telemetry.update();

        double left = 0;
        // checking to find the shortest way to travel to optimal pos , and determining the sense of the rotation(ccw/cw)
        if (Optimal_pos > heading) left = -heading + Optimal_pos;
        else left = -heading + 360 + Optimal_pos;

        double right = 0;
        if (Optimal_pos > heading) right = -Optimal_pos + heading + 360;
        else right = -Optimal_pos + heading;

        if( Math.min(left, right) <= okDegrees ) return;

        if (left < right)
            rnr.setPower(left, left, 0.006);//proportional turning
        else
            rnr.setPower(-right, -right, 0.006);//proportional turning

        if( !opModeIsActive() ) return;
    }
}
```

```
// CALL THIS METHOD FOR PLACING THE CUBE AFTER ROTATING IN FRONT OF THE CORRECT DRAWER, GUIDED BY ENCODERS
protected void place_cube_encoders(boolean middle)
{
    if(middle)
        rnr.distanceMove(20, 0.25, this);
    else
        rnr.distanceMove(30, 0.25, this);
    collector.openArms(3);
    sleep(500);
    rnr.distanceMove(-15, 0.25, this);
    sleep(500);

    if(runtime.seconds() > 28.0)    return;

    collector.closeArms(3);
    sleep(100);
    rnr.distanceMove(15, 0.25, this);
    sleep(100);
    rnr.distanceMove(-15, 0.25, this);
}
// THIS METHOD GETS THE REAL  INTEGRATED Z VALUE FROM THE GYRO SENSOR ( THE ONE RETURNED BY DEFAULT IS NOT CORRECT IN OUR CASE)
protected double gyroGetIntegratedZ()
{
    double angle = gyro.getIntegratedZValue();
    double scale = 360.0 / 346.0;
    angle *= scale;
    return angle;
}
```

```
// THIS METHOD STOPS THE ROBOT IN FRONT OF THE FIRST DRAWER , IT RECOGNIZES THE DRAWER U
protected void go_to_drawer ()
{
    double initPower = 0.2;

    double orientation = getGyroHeading();
    rnr.setPower(-1,1, initPower * forward);


    double last_dist = dist_r.getDistance(DistanceUnit.CM);

    rangeTelemetry = telemetry.addData("Range", String.format("%.3f", last_dist));
    nrTelemetry = telemetry.addData("Nr", nr);
    lft = telemetry.addData("Left", -1);
    rgt = telemetry.addData("Right", -1);
    telemetry.update();

    while ( nr < 1 )
    {
        //Keep_Orientation(orientation);
        double dist = dist_r.getDistance(DistanceUnit.CM);
        if( last_dist - dist >= 5 )
            nr ++;
        last_dist = dist;
        rnr.setPower(-1,1, initPower * forward);

        rangeTelemetry.setValue( String.format("%.3f", dist));
        nrTelemetry.setValue(nr);
        gyroTelemetry.setValue(getGyroHeading());
        telemetry.update();
        sleep(200);
        if( !opModeIsActive() ) return;
    }

    rnr.setPower(0.0,0.0);
}
```

6) AUTONOMOUS MIDDLE

This class is where we call all of our autonomous methods from Autonomous Functions, for placing the cube into the middle-placed Crypto Box.

```
import java.security.KeyPair;

@Autonomous(name="Autonomous Middle", group="Linear Opmode")
@Disabled
public class AutonomousMiddle extends AutonomousFunctions {
    @Override

    public void runOpMode() {
        Servo extender;
        extender=hardwareMap.get(Servo.class,"extender");
        extender.setPosition(1.0);
        initialization ();


        waitForStart();

        runtime.reset();


        /// Grab cube
        state.setValue("grab cube");
        telemetry.update();
        grab_cube();
        if (!opModeIsActive()) return;

        /// Gets key drawer
        state.setValue("get key drawer");
        int drawer = getKeyDrawer();
        if(forward == 1)    drawer = 4 - drawer;
        telemetry.update();

        /// Score jewels
        state.setValue("score jewels");
        telemetry.update();
        scoreJewels(extender);
        if(!opModeIsActive())   return;
```

```
/// Get down from platform
state.setValue("get down from platform");
telemetry.update();
getDown();
Keep_Orientation(0);
if(!opModeIsActive())   return;

/// Go in front of first drawer
state.setValue("go to drawer");
telemetry.update();
go_to_drawer();
if (!opModeIsActive()) return;

/// Go to needed drawer
state.setValue("pick drawer");
telemetry.update();
pick_drawer(1);

/// Rotate 90 degrees
state.setValue("rotate");
telemetry.update();
Keep_Orientation(270);
if( !opModeIsActive() ) return;


/// Place cube
state.setValue("place cube");
telemetry.update();
place_cube_encoders(true);
if( !opModeIsActive() ) return;
```

• PROGRAMS

1) AUTONOMOUS BLUE SIDE

Autonomous blue side is the class used for placing the glyph into the sided crypto box for blue alliance.

```
package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
import com.qualcomm.robotcore.eventloop.opmode.Disabled;
import com.qualcomm.robotcore.hardware.Servo;

@Autonomous(name="Blue Side", group="Linear Opmode")
//@Disabled
public class AutonomousBlueSide extends AutonomousFunctions {

    static
    {
        color = 1;
        forward = -1;
    }

    @Override
    public void runOpMode() {
```

```
// DECLARE THE EXTENDER SERVO ,CORESPONDING TO JEWEL DROPPING ARM SO THAT WE COULD SET THE INIT PO
Servo extender;
extender=hardwareMap.get(Servo.class,"extender");
extender.setPosition(1.0);
initialization ();

waitForStart();

runtime.reset();


/// Grab cube
state.setValue("grab cube");
telemetry.update();
grab_cube();
if (!opModeIsActive()) return;

/// Gets key drawer
state.setValue("get key drawer");
int drawer = getKeyDrawer();
if(forward == 1)    drawer = 4 - drawer;
telemetry.update();

/// Score jewels
state.setValue("score jewels");
telemetry.update();
scoreJewels(extender);
if(!opModeIsActive())   return;

/// Get down from platform
state.setValue("get down from platform");
telemetry.update();
getDown();
if(!opModeIsActive())   return;
```

```
    /// Rotate 90 degrees
    state.setValue("rotate 1st time");
    telemetry.update();
    Keep_Orientation(270);
    if(!opModeIsActive())   return;

    /// Get back some cm to be sure that we not miss the drawer
    state.setValue("get back");
    telemetry.update();
    rnr.distanceMove(forward * -10, 0.4, this);
    if(!opModeIsActive())   return;

    /// Go in front of first drawer
    state.setValue("go to drawer");
    telemetry.update();
    go_to_drawer();
    if (!opModeIsActive()) return;

    /// Go to needed drawer
    state.setValue("pick drawer");
    telemetry.update();
    pick_drawer(2);

    /// Rotate 90 degrees
    state.setValue("rotate");
    telemetry.update();
    Keep_Orientation(180);
    if( !opModeIsActive() ) return;

    /// Place cube
    state.setValue("place cube");
    telemetry.update();
    place_cube_encoders(false);
    if( !opModeIsActive() ) return;
}
```

2) AUTONOMOUS RED SIDE

Autonomous blue side is the class used for placing the glyph into the sided crypto box for red alliance.

```
//@Disabled
public class AutonomousRedSide extends AutonomousFunctions {

    static
    {
        color = 0;
        forward = 1;
    }

    @Override
    public void runOpMode() {
        Servo extender;
        extender=hardwareMap.get(Servo.class,"extender");
        extender.setPosition(1.0);
        initialization ();

        waitForStart();

        runtime.reset();


        /// Grab cube
        state.setValue("grab cube");
        telemetry.update();
        grab_cube();
        if (!opModeIsActive()) return;

        /// Gets key drawer
        state.setValue("get key drawer");
        int drawer = getKeyDrawer();
        if(forward == 1)    drawer = 4 - drawer;
        telemetry.update();

        /// Score jewels
        state.setValue("score jewels");
        telemetry.update();
        scoreJewels(extender);
        if(!opModeIsActive())   return;
```

```
/// Get down from platform
state.setValue("get down from platform");
telemetry.update();
getDown();
if(!opModeIsActive())   return;

/// Rotate 90 degrees
state.setValue("rotate 1st time");
telemetry.update();
Keep_Orientation(90);
if(!opModeIsActive())   return;

/// Get back some cm to be sure that we not miss the c
state.setValue("get back");
telemetry.update();
rnr.distanceMove(forward * -10, 0.4, this);
if(!opModeIsActive())   return;

/// Go in front of first drawer
state.setValue("go to drawer");
telemetry.update();
go_to_drawer();
if (!opModeIsActive()) return;

/// Go to needed drawer
state.setValue("pick drawer");
telemetry.update();
pick_drawer(3);

/// Rotate 90 degrees
state.setValue("rotate");
telemetry.update();
Keep_Orientation(0);
if( !opModeIsActive() ) return;


/// Place cube
state.setValue("place cube");
telemetry.update();
place_cube_encoders(false);
if( !opModeIsActive() ) return;
```

For scoring glyphs into the mid crypto in autonomous we use Autonomous Forward Blue, Backward Blue, Forward Red, Backward Red, which inherit autonomous middle class, and using variables forward and color, the autonomous middle can handle both the correct alliance color and the correct move of the robot (backward/forward).

16 lines (12 sloc) | 300 Bytes Raw Blame History

```
1  package org.firstinspires.ftc.teamcode;
2
3  import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
4
5  @Autonomous(name="Blue Backward", group="Linear Opmode")
6  //@Disabled
7
8  public class AutonomousBackwardBlue extends AutonomousMiddle
9  {
10     static
11     {
12         forward = -1;
13         color = 1;
14     }
15 }
```

3) DRIVER CONTROL

Driver control class is used for controlling the robot in TeleOp.

```
@TeleOp(name="DriverControl", group="Linear Opmode")
//@Disabled
public class DriverControl extends LinearOpMode
{

    //Declare OpMode components
    private ElapsedTime runtime = new ElapsedTime();
    private Runner runner;
    private CubeCollector collector;
    private RelicGrabber grabber;

    @Override
    public void runOpMode()
    {
        /// Initialize objects
        telemetry.setAutoClear(false);

        Telemetry.Item timeTelemetry = telemetry.addData("Time", 0);

        Telemetry.Item runnerTelemetry = telemetry.addData("Runner", 0);
        runner = new Runner(
                hardwareMap.get(DcMotor.class, "frontleft"),
                hardwareMap.get(DcMotor.class, "backleft"),
                hardwareMap.get(DcMotor.class, "frontright"),
                hardwareMap.get(DcMotor.class, "backright"),
                runnerTelemetry
        );

        Telemetry.Item collectorTelemetry = telemetry.addData("Collector", 0);
        collector = new CubeCollector(
                hardwareMap.get(Servo.class, "upleft"),
                hardwareMap.get(Servo.class, "upright"),
                hardwareMap.get(Servo.class, "downleft"),
                hardwareMap.get(Servo.class, "downright"),
                hardwareMap.get(DcMotor.class, "lifter"),
                collectorTelemetry
        );
```

```
grabber = new RelicGrabber(
        hardwareMap.get(DcMotor.class, "monster"),
        hardwareMap.get(Servo.class, "claw"),
        hardwareMap.get(DcMotor.class, "pusher")
);

Servo extender, rotor;
extender = hardwareMap.get(Servo.class, "extender");
rotor = hardwareMap.get(Servo.class, "rotor");
extender.setPosition(1.0);

telemetry.update();

waitForStart();
runtime.reset();

/// Initializations after start
runner.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
collector.setLiftMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
grabber.setPusherMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);

int[] last = {0, 0, 0, 0};
boolean lb = false, rb = false, trg = false;
int forward = 1;

while (opModeIsActive())
{
    /// gamepad1 functions
 // WHEN DPAD DOWN / UP BUTTON IS PRESSED DURIN DRIVER CONTROLLED PERIOD SWITCH ROBOT FORWARD AND BACKWARD CONTROL
    if(gamepad1.dpad_down)
    {
        if(forward == 1) runner.swapMotors();
        forward = -1;
    }
    if(gamepad1.dpad_up)
```

```
if(gamepad1.dpad_up)
{
    if(forward == -1)   runner.swapMotors();
    forward = 1;
}

double rnrY = -gamepad1.left_stick_y;
double rnrX = gamepad1.right_stick_x;
if(gamepad1.left_bumper) runner.move(rnrY, rnrX, 0.25);
else if(gamepad1.right_bumper) runner.move(rnrY, rnrX, 0.5);
else runner.move(rnrY, rnrX);

/// gamepad2 functions
if(gamepad2.a && last[0] == 0)
{
    collector.changeState(2);
    last[0] = 1;
}
else if(!gamepad2.a) last[0] = 0;

if(gamepad2.y && last[1] == 0)
{
    collector.changeState(1);
    last[1] = 1;
}
else if(!gamepad2.y) last[1] = 0;

if(gamepad2.x && last[2] == 0)
{
    collector.changeState(3);
    last[2] = 1;
}
else if(!gamepad2.x) last[2] = 0;
```

```
if(gamepad2.b && last[3] == 0)
{
    collector.changeState(3);
    last[3] = 1;
}
else if(!gamepad2.b) last[3] = 0;

collector.moveLift(-gamepad2.left_stick_y * 0.75);

/*if(gamepad2.left_bumper)
    grabber.powerMonster(-0.1);
else if(gamepad2.right_bumper)
    grabber.powerMonster(0.5);
else
    grabber.powerMonster(0.0);*/
//grabber.powerMonster(-gamepad2.right_stick_y);

if(gamepad2.right_bumper)
    grabber.movePusher(-0.2);
else if(gamepad2.left_bumper)
    grabber.movePusher(0.2);
else
    grabber.movePusher(0.0);

if(gamepad2.left_trigger > 0.5 || gamepad2.right_trigger > 0.5)
{
    if(!trg)
    {
        trg = true;
        grabber.changeStateClaw();
    }
}
else
    trg = false;
// Reversing gamepad x axis
double g2ry = -gamepad2.right_stick_y;
```

```
            if(g2ry < 0)
            {
                if(gamepad2.dpad_down)  grabber.powerMonster(g2ry * 0.5);
                else grabber.powerMonster(g2ry * 0.1);
            }
            else
            {
                if(gamepad2.dpad_up)    grabber.powerMonster(g2ry * 0.5);
                else    grabber.powerMonster(g2ry * 0.3);
            }
            //grabber.movePusher(g2ry * 0.25);
            /*if(g2ry == 0)   grabber.movePusher(0.0);
            else if(g2ry > 0)   grabber.movePusher(g2ry * 0.1);
            else if(g2ry < 0)   grabber.movePusher(g2ry * 0.05);*/

            /// Telemetry
            timeTelemetry.setValue(runtime.toString());
            runner.logInformation("Position");
            collector.logInformation("Positions");
            telemetry.update();
        }
    }
}
```

• STRATEGY

FIRST TECH CHALLENGE

Control Award Content Sheet

Updated 9.9.2017

****Please turn in this sheet during your Judge Interview along with your Engineering Notebook****

Team #	Team Name:

Autonomous objectives:

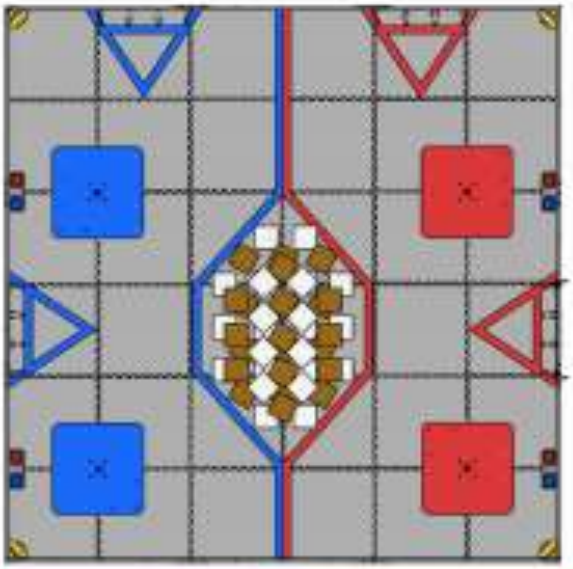
Sensors used:

Key algorithms:

Driver controlled enhancements:

Engineering notebook references:

Autonomous program diagrams:



THIS IS OUR JOURNEY THROUGH THE FTC 2017-2018 SEASON. THIS YEAR IT WAS A GREAT EXPERIENCE, WE LEARNED A LOT OF NEW THINGS AND WE SPENT SOME QUALITY TIME WITH OUR TEAMMATES AND OUR COLLEGUES. WE ALSO EXPECT A GOOD PERFORMANCE THIS YEAR. WISH US GOOD LUCK!