HEROTECH

herotech team, Romania
herotech.ro

ENGINEERING BOOK

Title: **Hephaestus**

Student Names: **Alexandru Radac, CodrinMuntean, Daniel Marpozan**

School: **National College "Octavian Goga"**

City and Country: **Sibiu, Romania**

Choosing the kit

November 8th 2018

We searched the web for information about the kits that were recommended by GENIUS Olympiad . After a few hours of comparing the price / quality and quantity of the kits, we decided to use one of the three VEX kits.

November 9th 2018

Based on the reviews other people left for the VEX kits and the budget we had, we decided to buy the VEX IQ pieces that we needed. We bought multiple small kits with different pieces to have all the different kinds of gears, shafts, plates, etc… We also bought a bigger robot kit that included multiple sets of pieces, the robot controller and the robot brain.

December 3rd 2018

We had to wait a long time before the pieces finally arrived in Romania, as they were shipped from the United States. We unboxed everything to check if everything has arrived. After this Alex sorted all the pieces based on what they were for. This took way longer than we initially expected.

herotech team, Romania
herotech.ro

Learning to build

December 7th 2018

All three of us met at Alex's place where we started to learn how to use the new pieces. Alex already had some experience with building robots from LEGO pieces, but VEX IQ were a totally new experience. We started by watching VEX IQ tutorials from their website and some videos on YouTube posted by other people who owned VEX components.

December 8th 2018

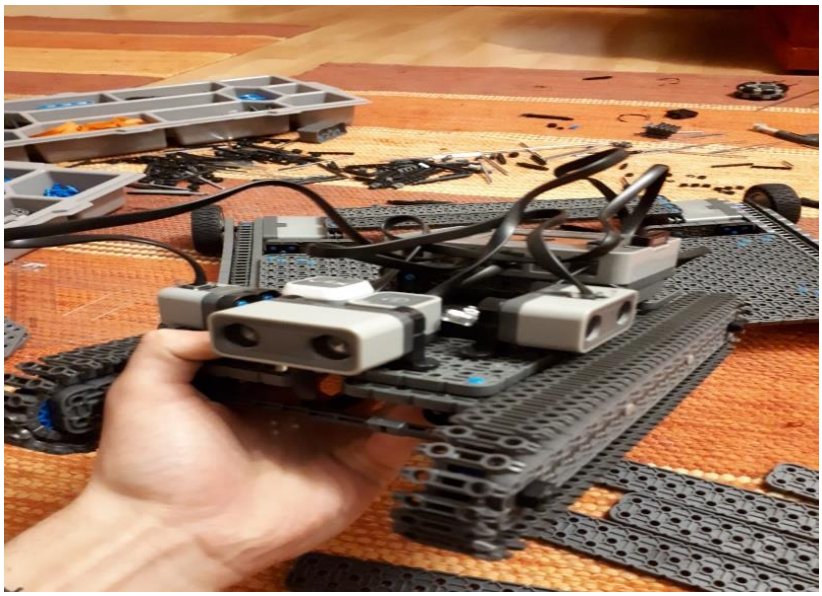
Alex and Codrin met again, and started testing the electrical components of the kits. Firstly we had to update the firmware for the sensors, motors and the controller. From the start we had some problems because the interface was not easy to use and we were stuck for a few hours at this. After finally figuring out the problem, Alex built a small, basic robot, which we used to test the sensors and learn how to program the robot. Codrin was the one who wrote the code for the robot as he already had experience with the C++ coding language.

We tested the 3 ultrasound distance sensors to see if they were accurate. Two of them worked perfectly, but one was giving us errors at times so we decided not to use it as it was flawed.

We also tested the gyroscope and the motors and they were all working fine.

December 10th 2018

This day Alex tried to change the wheels with tracks to see if they would be better. He decided not to use them as the friction between the tracks and the floor was way worse than that of the rubber wheels. He also tried to configure the color sensors but didn't manage to make them work at all.



This was our first try: sensors integrated into a simple robot model with tracks.

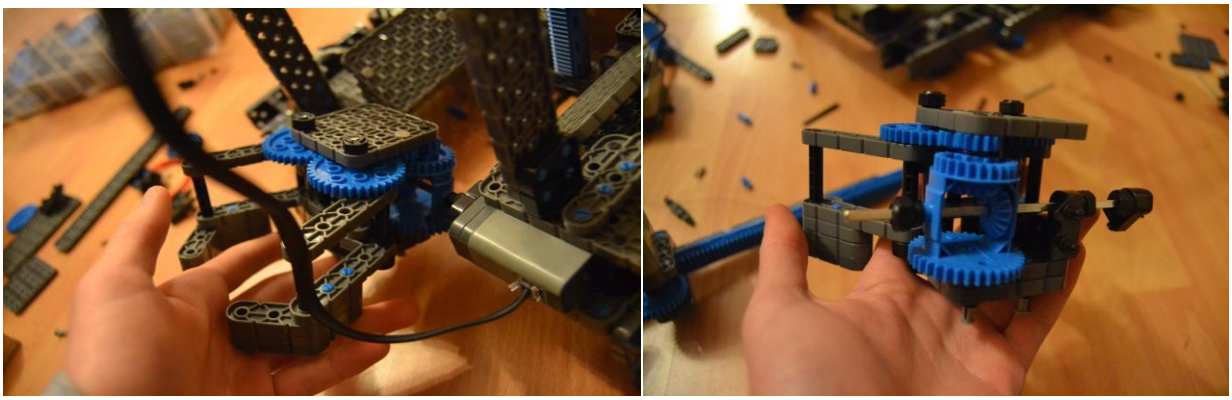
December 11th 2018

We met again to make a plan and to disassemble the first robot. We decided to try to build the robot for the precedent year to gather more experience. Also, as two of us were out of town during the winter break, we chose to meet again after the 1st day of 2019.

January 3rd 2019

As we all were back in town, we started building the second robot.

We went with Alex's design and built an arm that uses only one motor to move on 2 axis.



The claw as seen placed in the completed robot and as a simple module

It was created purely from mind and it worked better than expected.

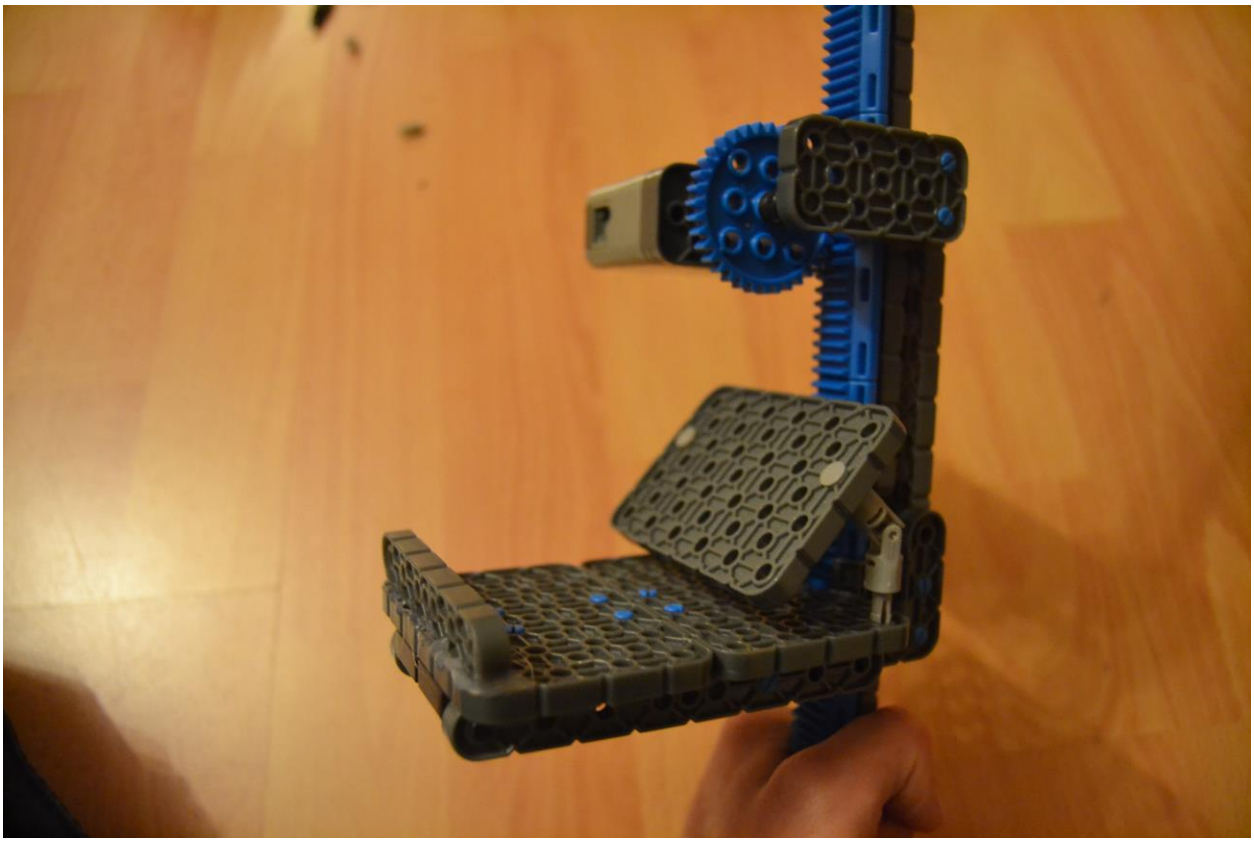
The motor is connected through a shaft to the other side of the base for better stability. The claw was designed to hold cubes, and when one cube was inside, the motor would turn, closing the claw. When the claw was fully closed, the system inside would lock and make the whole claw rotate upside. The claw would rotate 90 degrees, and when the cube was above it would drop to the other side.

After this, the system would work backwards, opening the claw to the maximum, and after that the whole claw would rotate downside until it reached the ground.

January 4th 2019

Alex built a system for the robot that looks like a lift for cubes. It was used to bring the cubes from the claw to the sorting system on the top of the robot. A long track of blue pieces were used to make the

base go up when the motor was rotating the gear.



We used a panel placed at an angle so the cubes wouldn't get stuck. For the same reason we glued with hot plastic a small piece at the front. This also prevents the cube from falling.

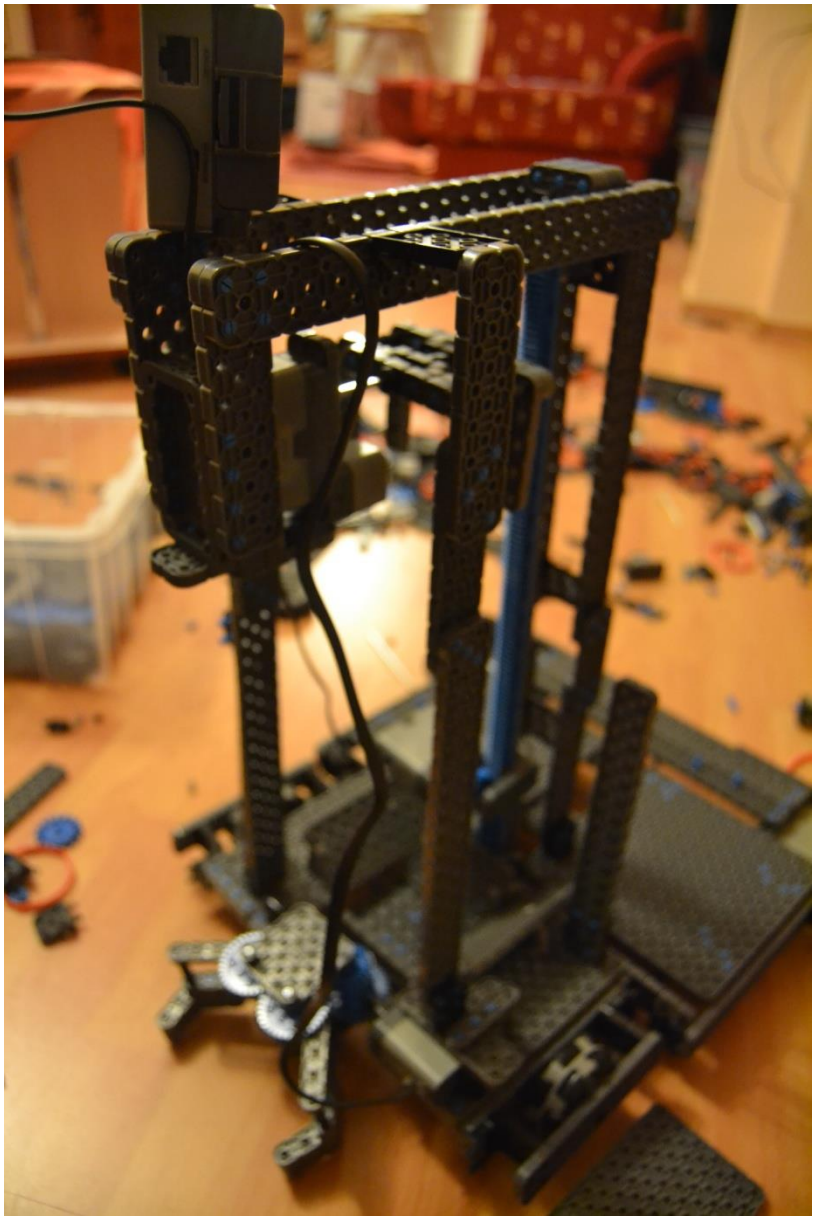
We had to grind a small bit of the flat piece at the bottom because it was getting stuck when moving along the blue rails.

January 5th 2019

We finished the robot and used the controller to move the motors. We didn't write a specific code for this robot as we didn't need to, to use it at its full potential.

HEROTECH

herotech team, Romania
herotech.ro



Here the robot can be seen completed with all the modules we built for it. We didn't create it to be fully able to do the last year's task, we built the parts that we felt would be the most challenging to create.

At the superior part of the robot, a motor with cube housing exists. It is connected to a color sensor that would have determined the color of the cubes and sort them accordingly.

HEROTECH

herotech team, Romania
herotech.ro

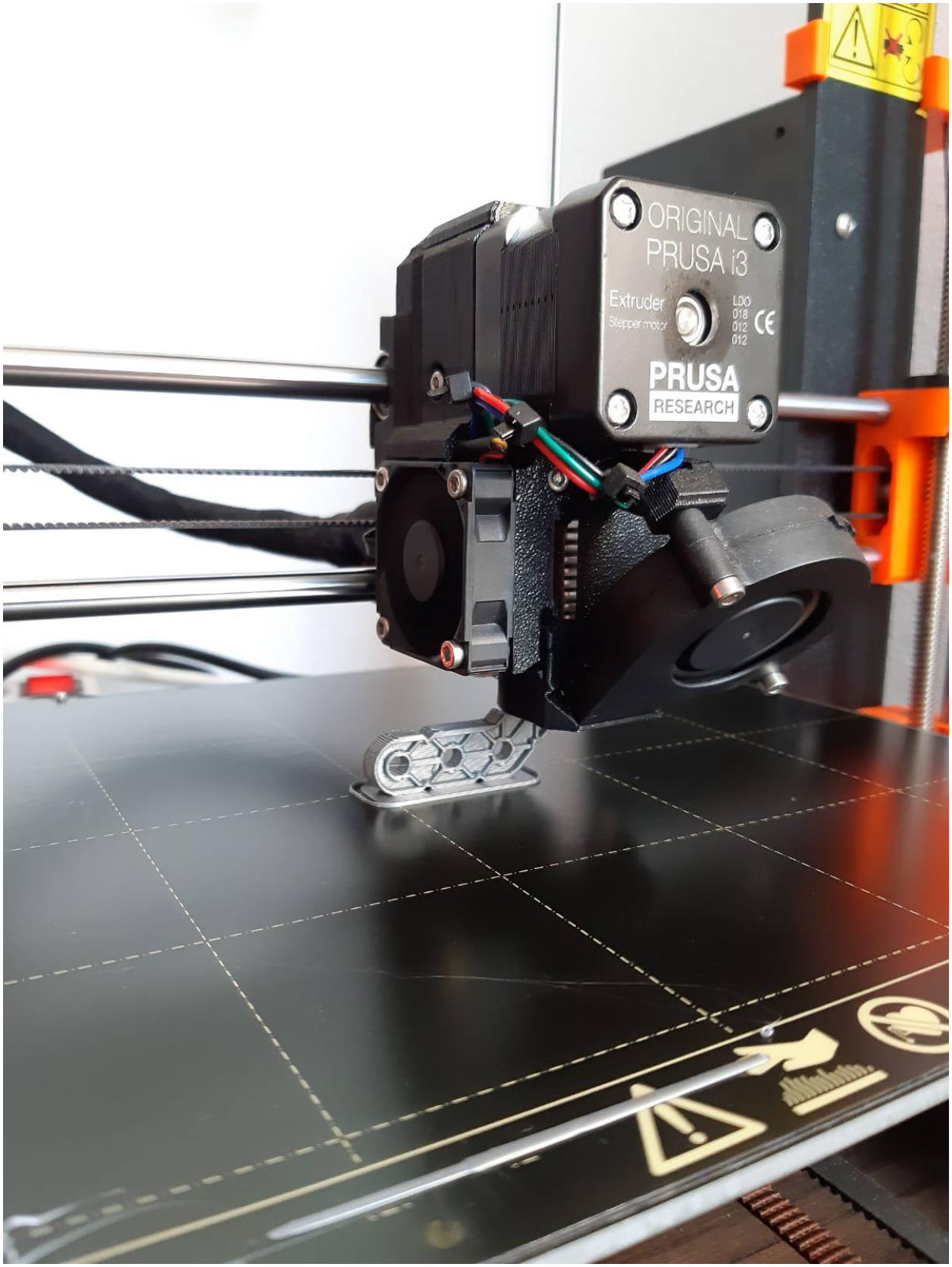
January 6th 2019

herotech team, Romania
herotech.ro

We thought that it would be a great idea to 3D print some VEX pieces that we may need later.

HEROTECH

herotech team, Romania
herotech.ro



herotech team, Romania
herotech.ro

Building Hephaestus

January 12th 2019

After nearly a week, the three of us met at our school to talk about how we would build the robot. We thought that it would be the most efficient if we split the work. As such, we decided that Alex and Daniel would build it based on the instructions from Alex and Codrin would write the code for the robot.

January 13th 2019

The base idea we was to design is to be as reliable as possible such that it will always manage to complete the first task within limits and be easy to control in the second task.

The first step was to create a base for the robot on which we could place all the things we needed to complete the tasks.

Secondly, we built the long arm of the robot that has the purpose to extend over the hazard zone and to hold the wood figurine with a claw that we mounted later.

At the end of the arm, a joint controlled by a motor exists to hold the claw and the distance sensor that we use in the first task to position the robot.

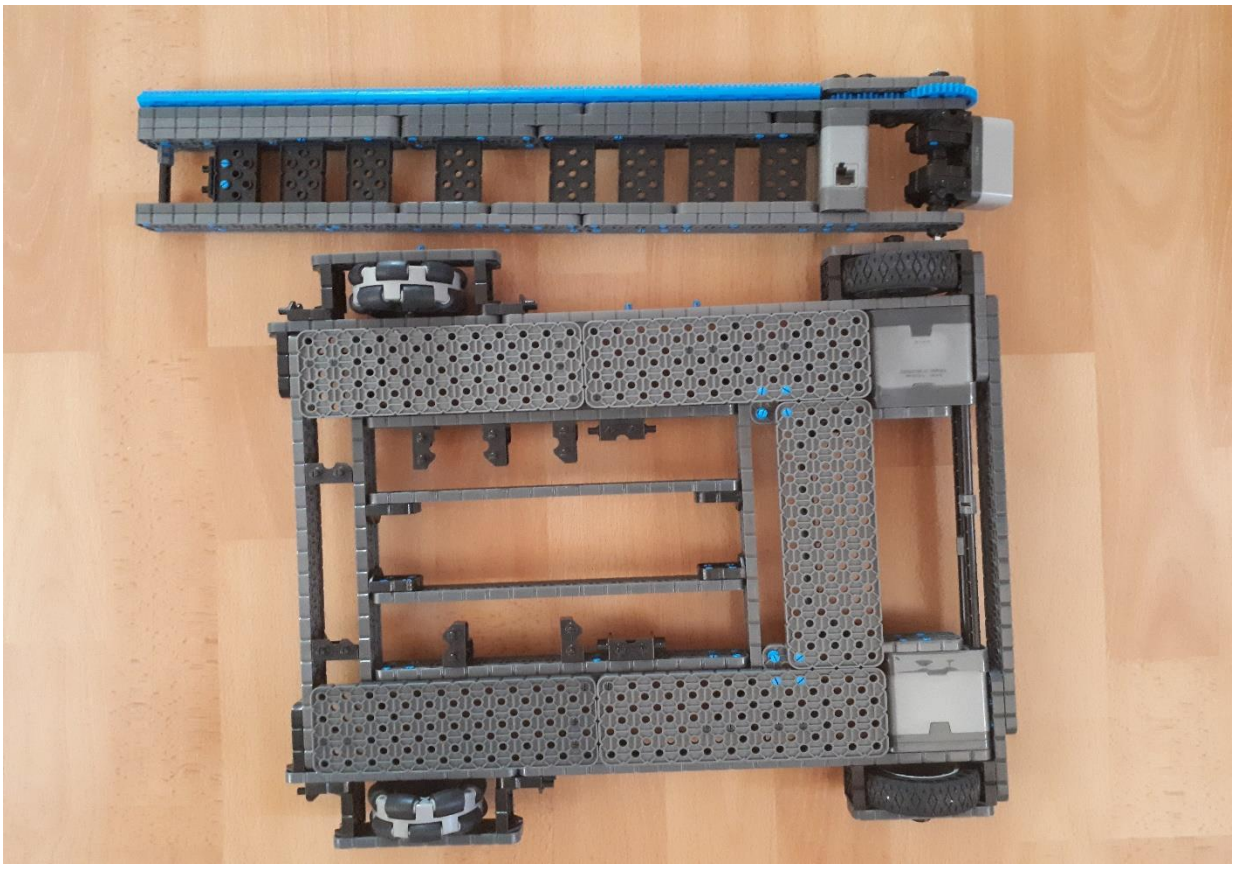
Codrin started to write the code for the first task that the robot hat to do. We had multiple ideas, but because the sensors were not as reliable as we though they were, we had to use as less of them as possible. We postponed writing the code until after the robot was completely built.

January 14th 2019

We modified the base a little because it was not resistant enough, so we added more piece on the bottom to support the weight of the robot.

Also, we placed the motor and wheels on the robot. A housing for the wheels was created to better support them and prevent them from falling off.

Instead of using normal wheels for the frontal slots, we used omnidirectional ones to help us take easier turns. This helped us a lot but also created a big problem for us that we will explain later.



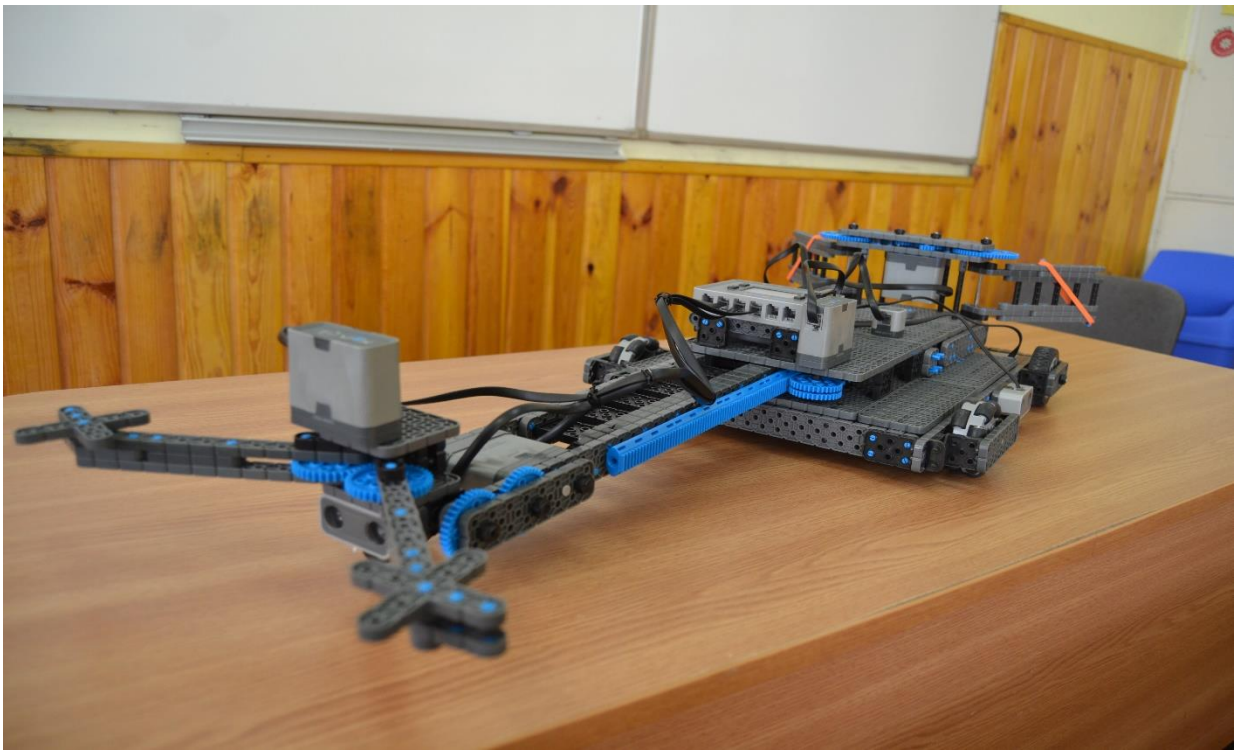
The base of the robot together with the long arm

January 17th 2019

We created the housing of the long arm and the motor that moves it. This system is made to be as strong as possible to be able to withstand the weight of the arm when it is fully extended.

We also built the small claw and the big claw. The big claw was quite a simple module to build, but the smaller one was tricky. We knew that it still needed to be upgraded, but we left it for some other time,

after we had sent our application.



This model is the first complete robot we had built

We connected the robot brain to all the motors and sensors. Codrin wrote a simple program that allowed us to control the robot.

After everything was set, we created a replica of the game table and we filmed our robot completing the tasks.

January 18th 2019

We connected the robot brain to all the motors and sensors. Codrin wrote a simple program that allowed us to control the robot.

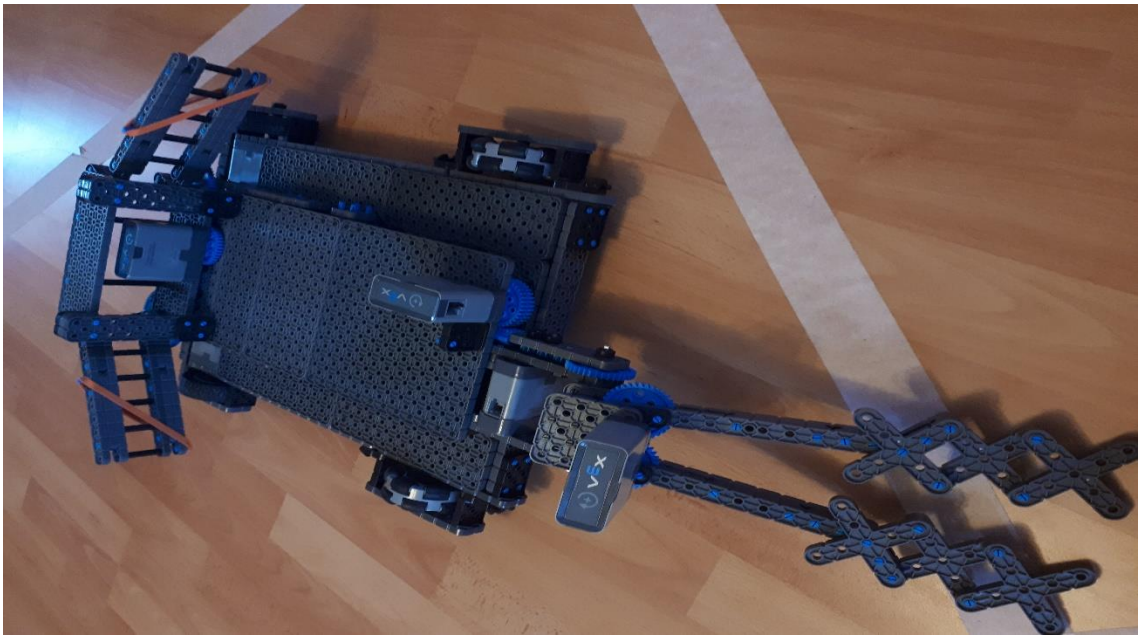
After everything was set, we created a replica of the game table and we filmed our robot completing the tasks.

The application was sent.

April 20th 2019

A long time has passed since we last worked on the robot, but now we started again. We made two major changes to the robot.

Firstly we changed the big claw, making it more stable by lowering the center of gravity and we modified the small claw to hold more wood pieces at a time



April 21th 2019

This was one of our hardest days. We created the program for the second task that was fairly easy, but the first one was hard. We went with Alex's algorithm that finally allowed us to have satisfactory results.

For the second task we programmed the controller to move the motors with variable speeds. We also made two modes of use for the controller because we didn't have enough buttons on it to control everything.

This marks the end of the developing and building part of the robot.

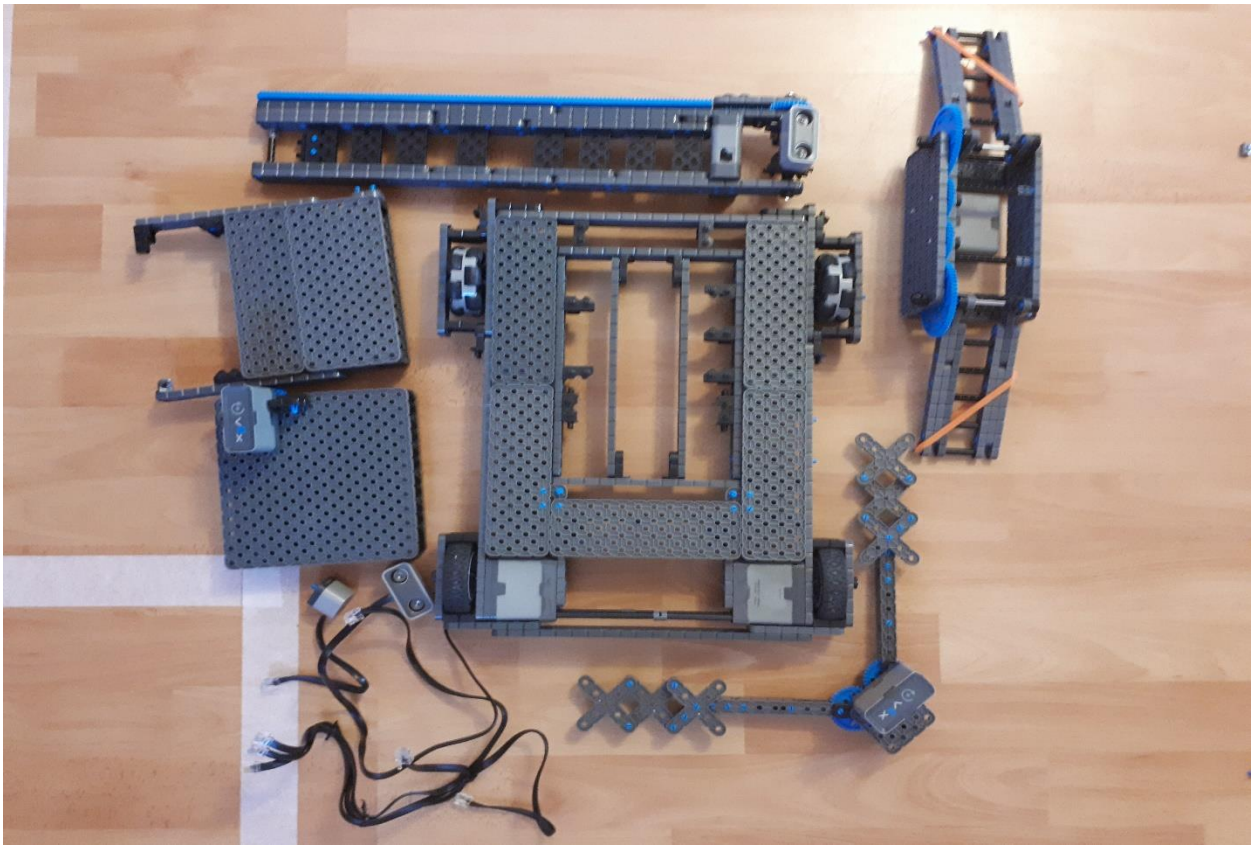
herotech team, Romania
herotech.ro

Design

Robot Modules

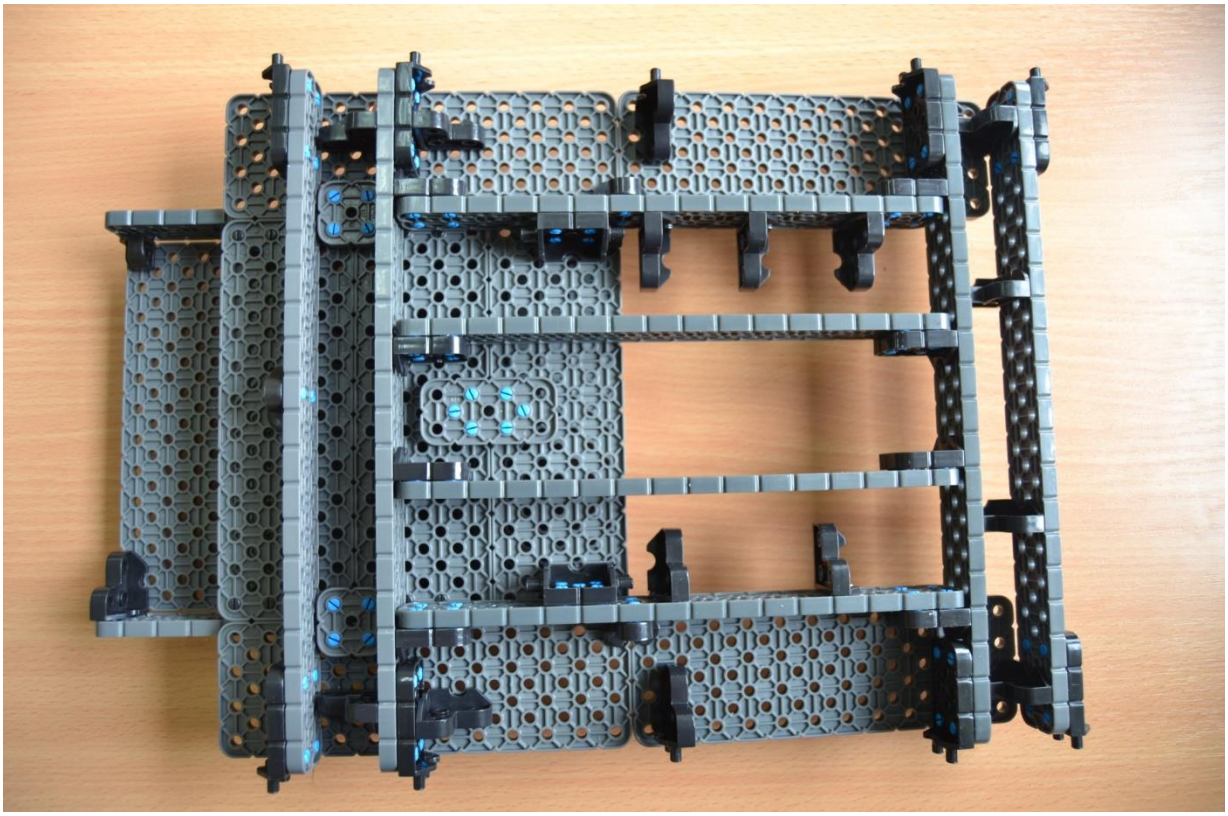
Hephaestus was created to be as modular as possible. Any big part of the robot can be separated from the rest easily, without affecting the rest of the modules. This allows us to disassemble it very fast, making it fit in a smaller package. This will be necessary because we have to transport it overseas. Also, by designing the robot to be modular, we can troubleshoot problems really fast in case something

happens.



The modules of the robot put side by side

- Module #1: The Base



This module is the biggest part of the robot, it being the resistance structure on which every other module is placed and fixed.

It was the first component of Hephaestus that we built, as such, all the other modules were created to fit.

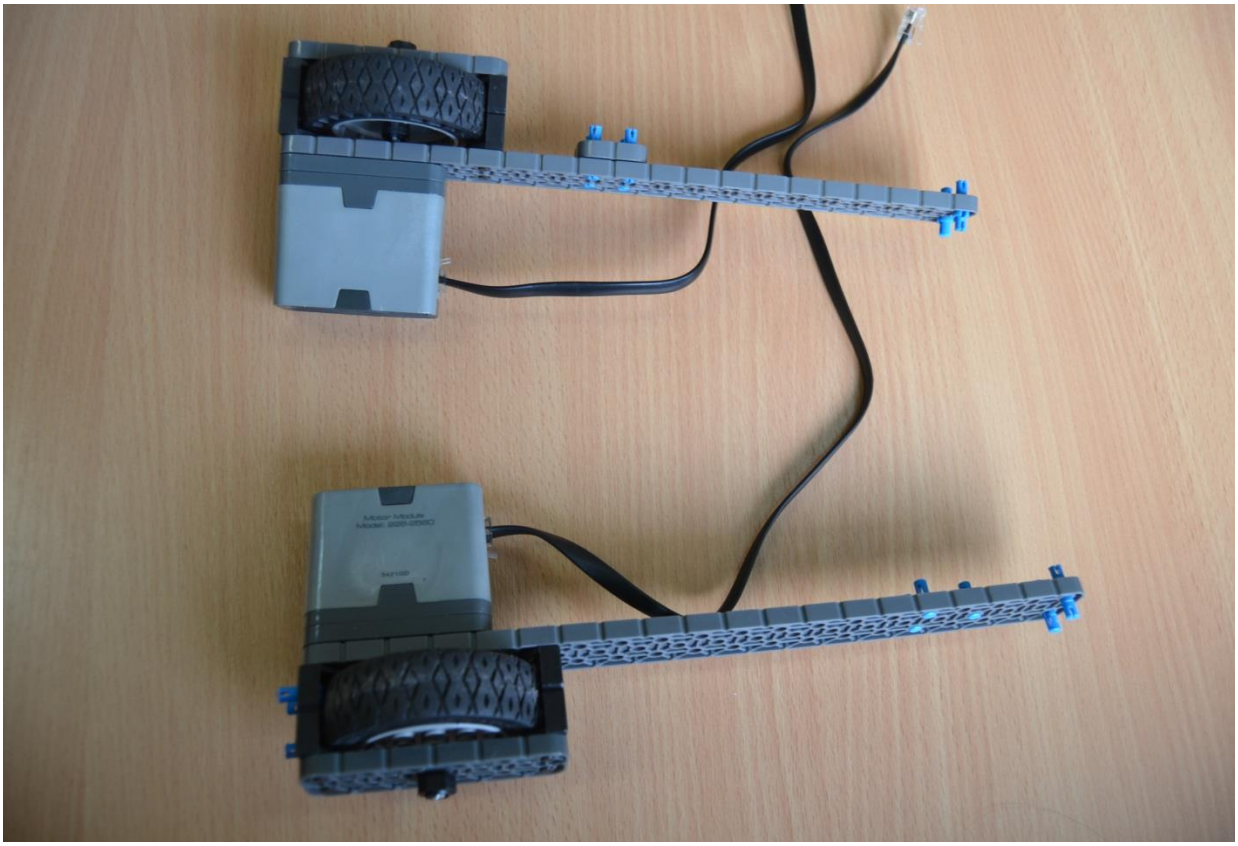
This part is made of plates of different sizes and lengths, black connectors that give the structure resistance and long beams that we used on the underside of the module.

The biggest problem we had in the beginning was that the bases we made were breaking and bending under the weight of the robot. To get over this problem we

decided that the upper part would be made out of plates, and the other one out of long beams. We did this because we observed that the thinner part of the pieces was the one that was failing and bending. We placed all the pieces at a 90 degrees angle to have the resistant part of the pieces on all three axis.

This action made the robot heavier, but helped much more because the modules that had motors were much more stable.

Module #2: Motor Wheels



This module is the one that allows the robot to move and turn.

It is made from 2 long beams, 2 motors, 2 rubber wheels, 2 shafts that connect the motor to the wheels directly and some smaller pieces that compose the housing of the wheels.

The two long beams allow us to strongly connect this module to Module #1 with blue pins.

The housing is a very important part because it prevents object to get stuck between the robot and the wheels, and even more important, it is the place where the shaft that is coming from the motor, going to the wheel, ends. This makes the wheel stay straight, not bending under the weight of the robot.

For better resistance and stability, the motors are not connected only to the base, but also to each other through a special piece, dedicated to motors, that is connected to the base.

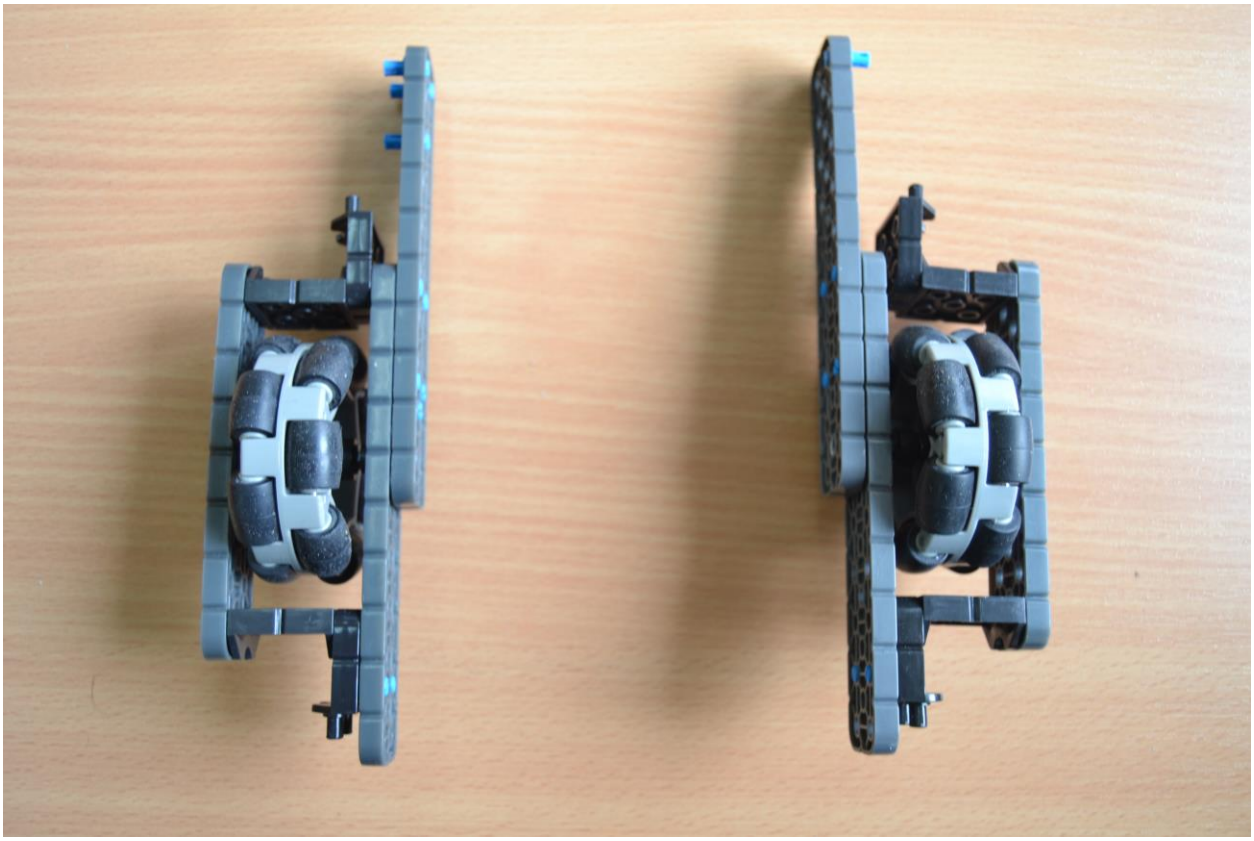
Alternative design:

At a point in the development of the robot we tried it to move faster. As we already were using the motors at their full speed, the only option was to make a gear system that would transform a part of the motor torque to rotation speed. As such, we used two shafts, one was connected to the motor and a big gear, and the other one was connected to the wheel and a small gear. The gears were interconnected and because of the difference of size, the translation was 1 to 4.

This made the robot much faster but it drastically reduced its force. We tested it heavily but we couldn't make it turn.

So, we returned to the base model that we used, which was simpler and stronger, but slower.

Module #3: Front Wheels



This module is made out of two omnidirectional wheels that are used to hold the weight of the robot. Similar to Module #2, they also have a housing that supports them and are connected to the base through a long beam.

Alternative design:

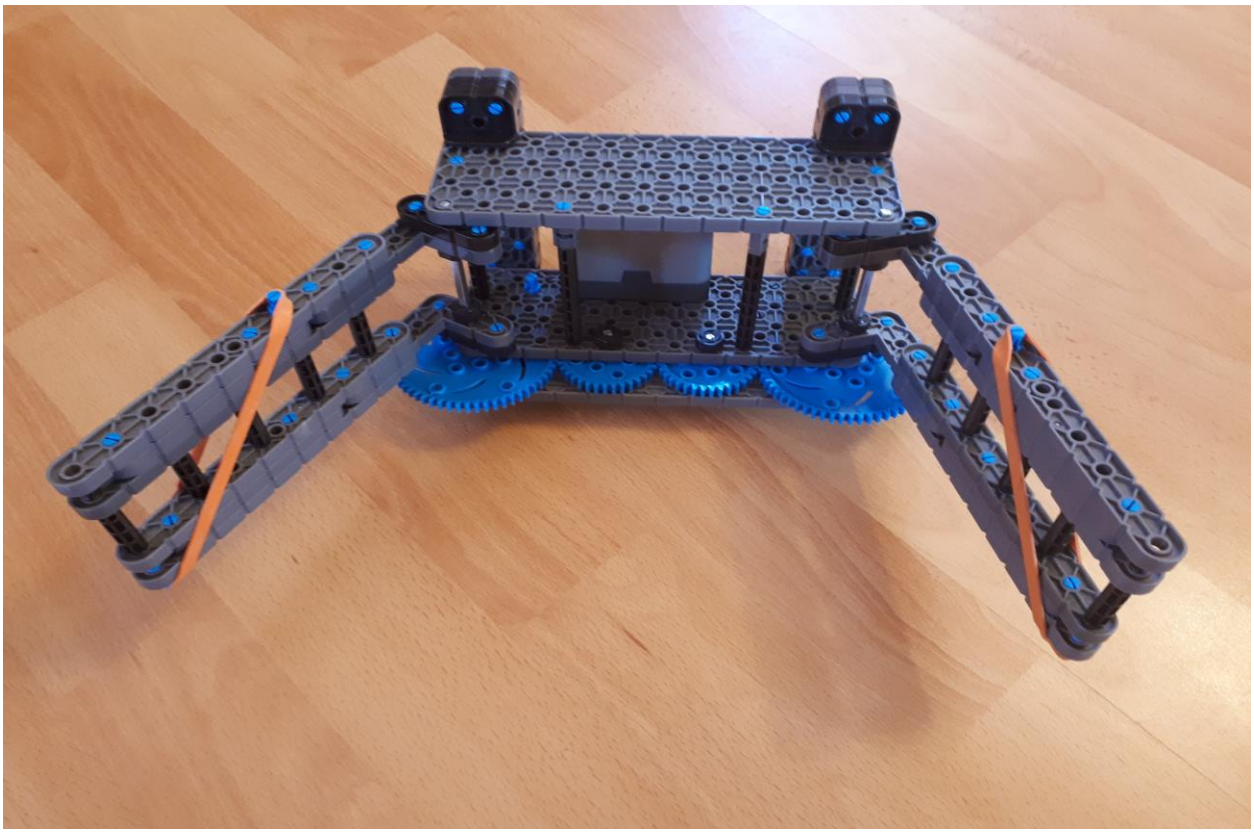
Our first design of this module was made with normal rubber wheels. They worked great when the robot had to move forward or backward, even when taking turns. But after more and more modules were placed on Hephaestus, the friction between the rubber wheels and the floor, created by the weight of the robot, made

the turns stop, because the motors didn't have enough power to rotate the whole robot.

As so, we moved to using omnidirectional wheels that weren't affected by the weight of the robot, but were a little bit bigger than the rubber ones. This made us to change the housing and the way we connected them to the base.

The omnidirectional wheels also created another problem, turns that were too wide at high speeds.

Module #4: Big Claw

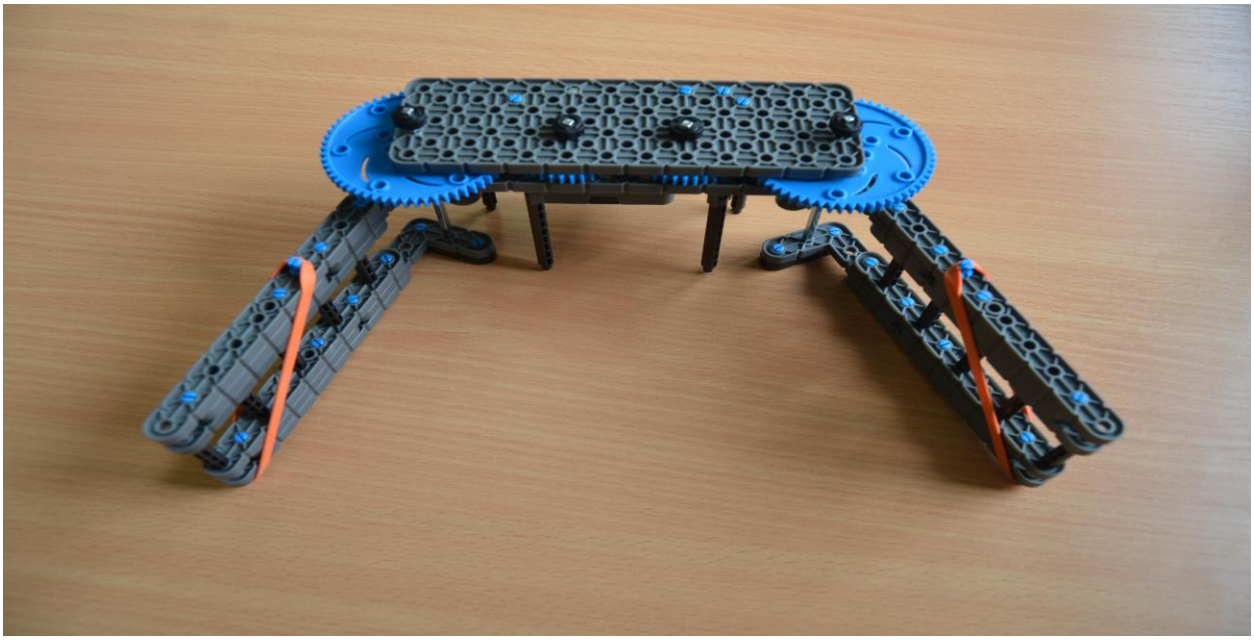


This is a module that has only one purpose, to carry the cubes from the second task to the fire zone. It is made from a lot of gears that conect the motor to the two arms. The arms are connected to the big gears directly. Through the medium to small gear mechanism, the pressing force of the arms is doubled as the ratio between the gears is 2 to 1.We used two orange rubber bands to create more adhesion between the cube and the big claw, to transport them more easily. It is mounted on the back of the

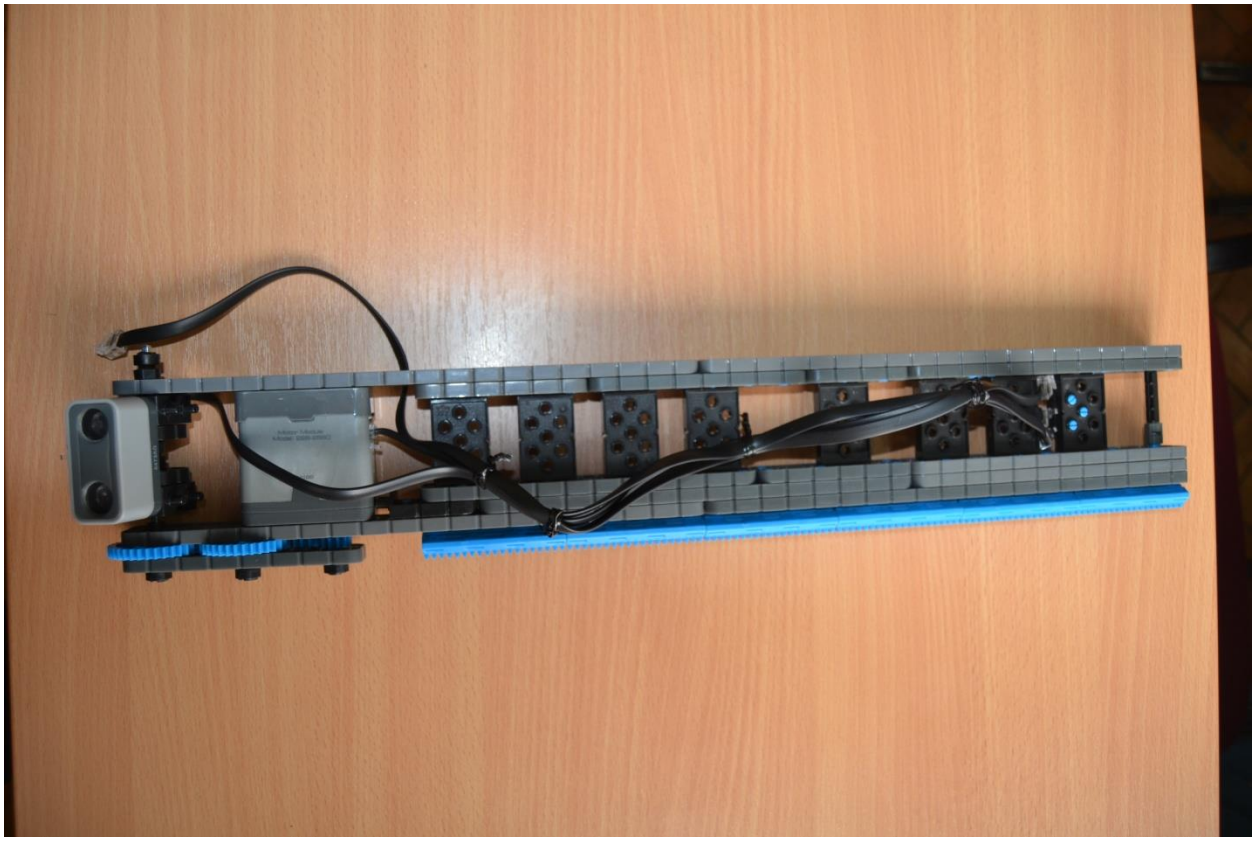
robot because there was the only space left available for such a big module and by it being placed that way, we can catch the cubes easier that it would have been if the module was on one of the sides of the robot.

Alternative design:

The first design of this module was very close to the current one, but had one big differance. All the gears and the motor were placed on the top. This made the module shake when it was in use and it was way more unstable when the robot was moving. The only reason we chose to have it that way was to prevent the cube from touching the moving gears, but after some testing we figured out is not a big deal and we switched to the later module.



Module #5: The Long Arm



The long arm is the most important module of the robot as it is used both in the first task and in the second one. It is composed of a long VEX structure made from beams, connected between them with the black pieces, an engine connected to 4 gears (2 medium, 2 small) that pass the movement an accentuate the force 3 times, and 6 blue “rails” that connect this module to its housing and make extension possible. It has a distance sensor that we use in the first task connected to a shaft that is rotated by the last gear and a support for the small claw.

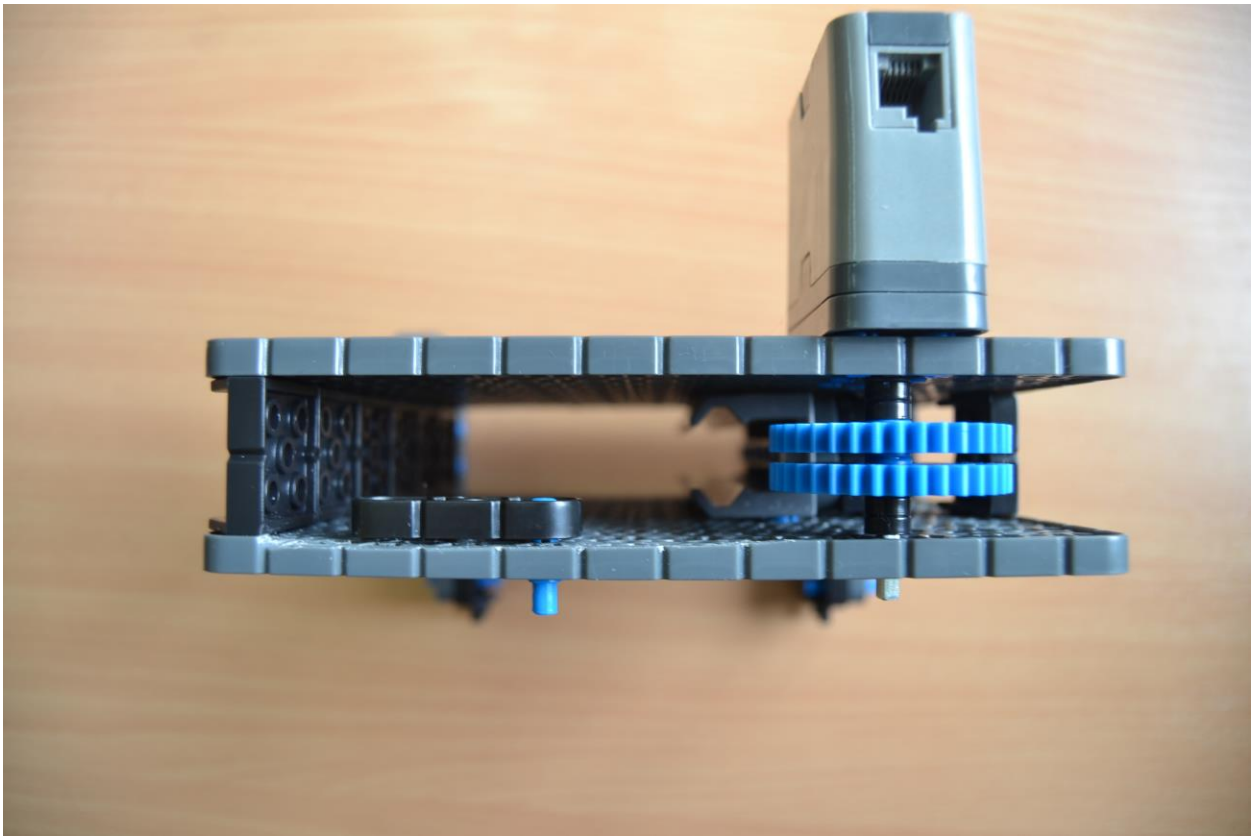
This module was the hardest one to build as we had a lot of restrictions for weight, length, resistance and the way the cables were placed. We spent a lot of time building it but we did a good job as we didn’t have to modify it later.

One problem we had was that the motor didn’t have enough force so we had to add the two small gears to triple the force but that also decreased the rotation speed.

Because this module is so long and heavy, we had to make it nearly impossible to bend, making it more stable on the robot base.

herotech team, Romania
herotech.ro

Module #6: The Long Arm Housing



This module is quite a simple one. It has 2 big flat components, a guidance piece for the arm, a stopper so the arm can't extend more than we want and 2 medium gears connected to a motor, stacked one on top of each other to move the arm.

This was also a module we used from the beginning without much modification.

It is very important for it to be solid and to be well fixed to the robot base, because when the long arm (module #6) is extended it must keep it as straight as possible, without letting it overextend and tilt.

We had quite a lot of trouble building this one, because the arm would get stuck in the edges of the big flat parts, so we had to polish them to be smooth.

One problem we encountered was that, when the arm was fully extended, the engine wouldn't have enough power to pull it back in, so we had to run it at

maximum speed.

Module #7: Small Claw

This claw went through quite a lot of modifying as we had to increase the maximum distance we could pick up a wood figurine. We decided that the claw should have 2 long arms. We also put at the end of the arms 6 + form pieces that were easy to fix and could hold the figurine properly “by the head”.

The module is made from 2 same size medium gears connected to the arms, an engine connected to one of the gears, respectively to the other, as the gears are interconnected.

This module has been designed to be placed at the end of the long arm.
Initially this module had 2 small gears, but we changed it to have 2 medium gears

because that way the figurine was easier to pick and it wouldn't fall at times from the grip.

One of the reasons why the arms are long is because after the wood figurine is caught, it has to be lifted to be placed in the bin, and as the claw doesn't have a motor to lift it, it uses the motor from the long arm to move it wholly.

Alternative design:

The first iteration of this design was one with only two + form pieces. This allowed us to hold the figurine but it took a long time to position it properly for the second task and it would sometimes drop the wood piece in the first task.

One part that was better with this module is that it was lighter because it was not so long.

Module #8: Robot Brain

This part is not actually a module made by us, but one “customized” by us. It

contains the main controller, a gyroscope sensor and a distance sensor. This

module also contains 9 cables of different lengths that connect everything.

Through two radio elements, the robot brain is connected to a controller which we use in the second task to pilot the robot.

It is placed on top of the robot base, connected in 4 points and has a space to

remove the battery without having to take everything out, for a faster and easier

charging.

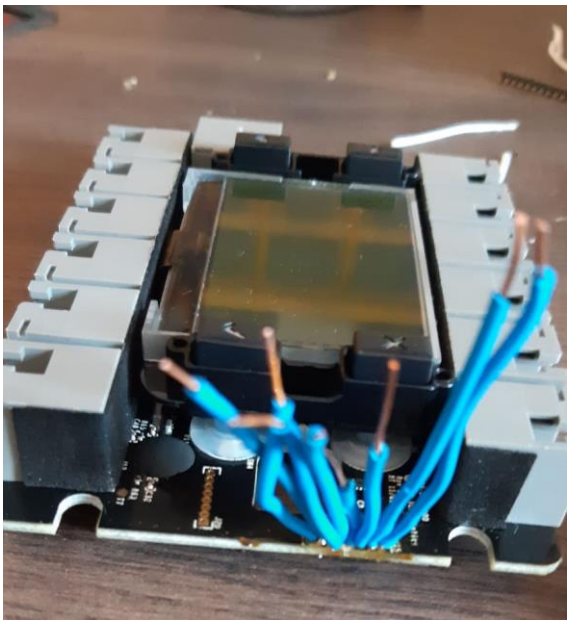
We decided to add a gyroscope to guide the robot to take perfect x degrees turns.

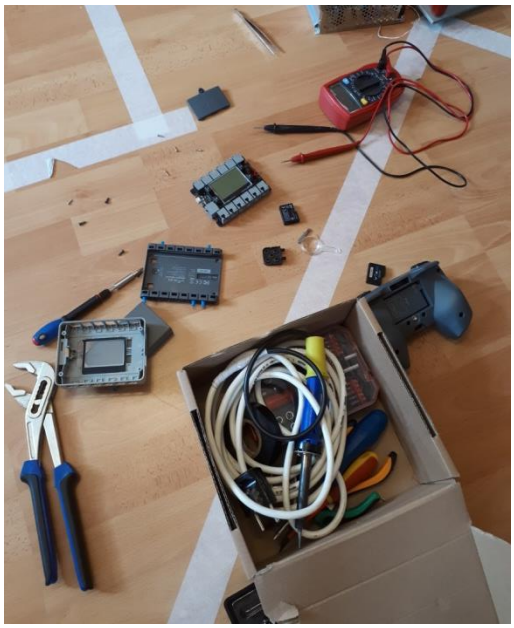
We also have a distance sensor that is placed on the side of the robot and is used

in the first task.

One of the problems we had with it was the length of the cables, they were not long enough to reach the end of the extended arm and so we had to limit the length of that module to the length of our longest cable.

Another problem we had and the biggest challenge we had/still have while writing this was breaking the Robot Brain. When we were uploading code to make the final tests for GENIUS Olympiad, the Micro USB port on the brain broke down. Alex and Daniel tried to repair it by soldering a new port in the place of the old one, but they were not successful. When they tried to repair it the radio port was also destroyed. Now we just ordered a full VEX kit (the brain was not in stock anywhere separately) to arrive at the hotel where we will sleep in the first night in New York. We pray it is delivered in time.



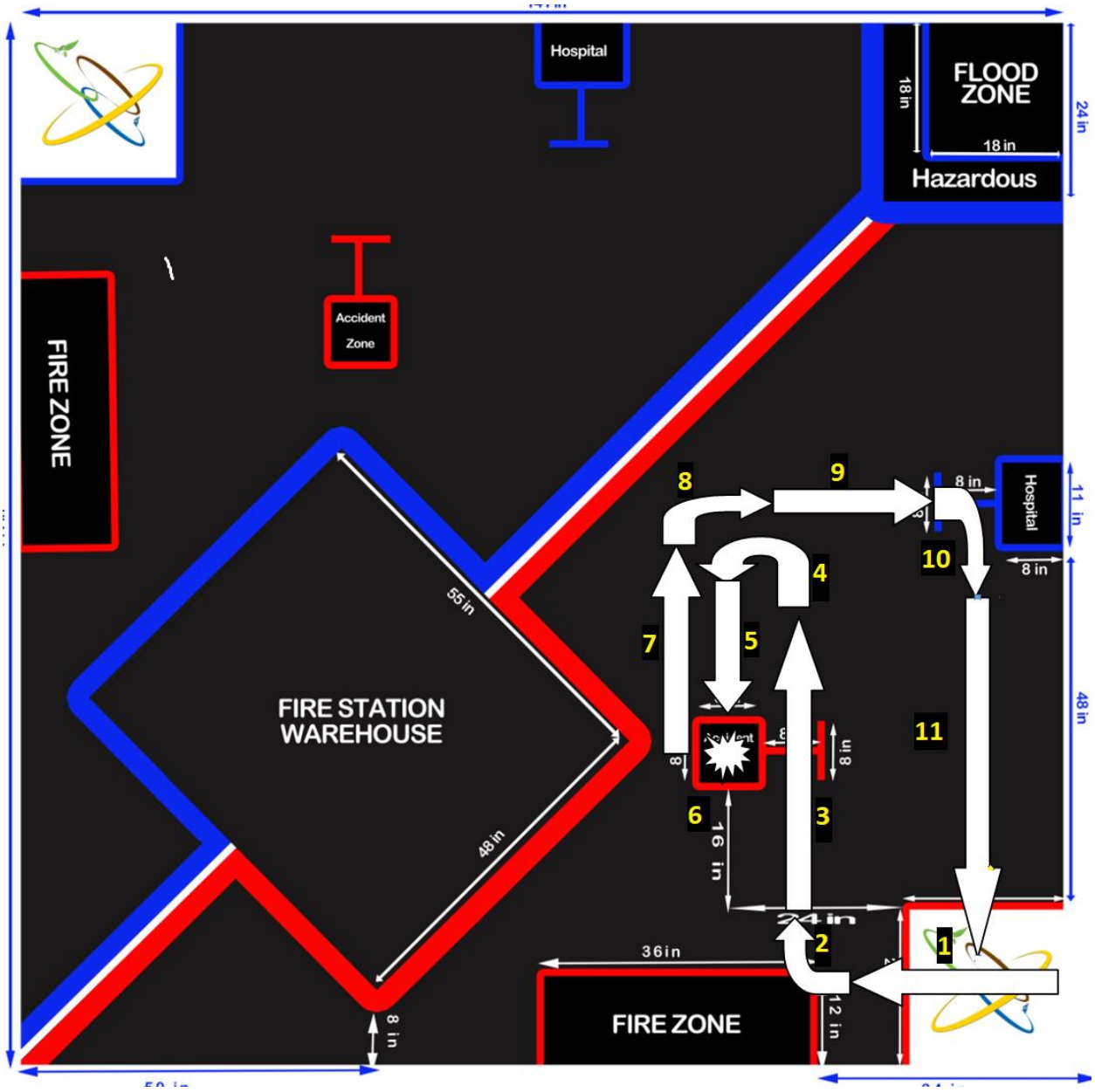


Robot code

First task

herotech team, Romania
herotech.ro

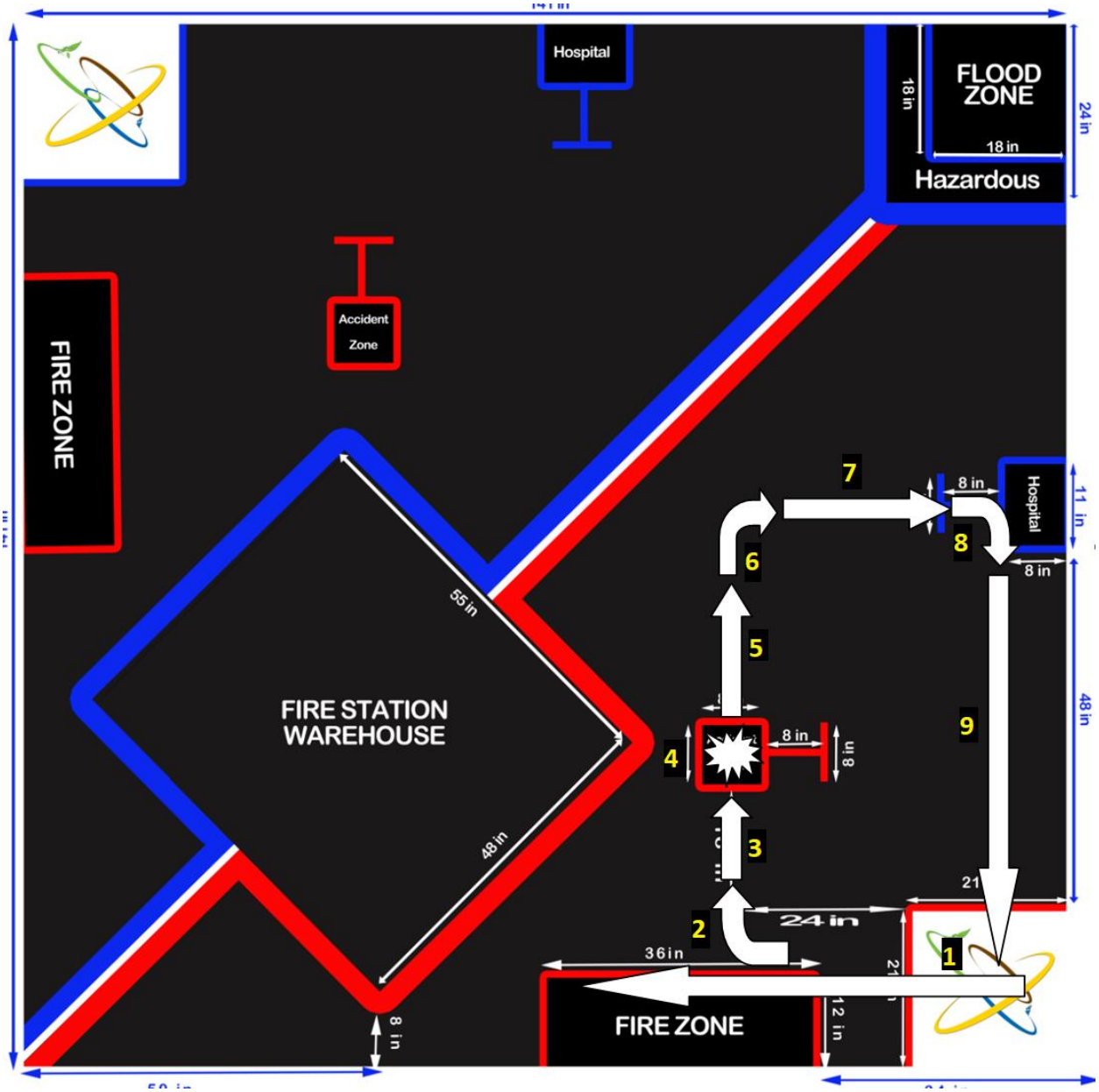
We wrote the code for the first in two different ways. In the first one, the robot moved on the path that is show in the picture.



1. The robot moves straight ahead for 70 cm;
2. The robot takes a 90 degree turn to the right;

3. Measuring the distance with the ultrasonic sensor placed on its side, the robot moves forward for 110 cm. We also measure how much the robot moves with the encoder integrated in the smart motors;
4. After turning 90 degree to the left, based on the minimum distance detected, the robot calculates how much it has to go forward to be aligned with the human figurine. It then takes another 90 degree turn to the left;
5. Using the frontal distance sensor, the robot moves forward until it detects the wood figurine at 14 cm;
6. The robot closes the small claw and then lifts it up;
7. The robot goes backwards based on the distance it traveled at point number 5 when it was moving towards the figurine;
8. The robot takes a 90 degree turn to the left;
9. The robot goes forward and leaves the figurine in the bin;
10. The robot takes a 90 degree turn to the right;
11. The robot moves forward until it reaches the robot lot.

This way of solving the first task was successful. We didn't use color sensor at all because they were not safe enough to trust them, this is why this code is quite a bit more complex. Another reason we didn't use the color sensors was that we don't know the exact dimensions of the game field, so we didn't want to risk anything. This took a little too long to complete. As we have to switch the program between the first and second task we decided to write another program that could do better at this task.



This code uses a similar principle as the anterior one, but it can easily be seen which one is simpler from the images.

This one also doesn't use the color sensors.

1. Measuring the distance with the ultrasonic sensor placed on its side, the robot moves forward for 110 cm;
2. The robot goes backwards to the place where it measured the smallest distance at point 1 and takes a 90 degree turn to the right;
3. Using the frontal distance sensor, the robot moves forward until it detects the wood figurine at 14 cm;
4. The robot closes the small claw and then lifts it up;
5. The robot moves forward a distance we set – how much it moved forward at point 3;
6. The robot takes a 90 degree turn to the right;
7. The robot moves forward and places the figurine in the bin;
8. The robot takes a 90 degree turn to the right;
9. The robot moves forward until it reaches the robot lot.

Using this algorithm we shave 7 seconds off the time we were getting from the anterior design. This allows us to have enough time to switch the programs between tasks.

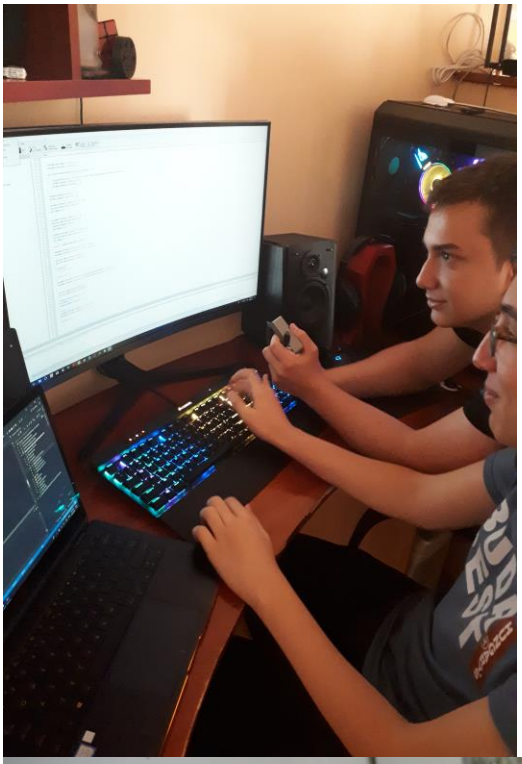
This way of solving the problem also grants us a higher chance of successfully rescuing the human trapped in the fire zone, as Hephaestus doesn't have to take as many turns as it has to with the other algorithm.

The best time we got was 27 seconds.

The success ratio is 18/23.

After we arrive at GENIUS Olympiad we will have to test this program on the game field, because we will have to make small adjustments to the variables of the code.

herotech team, Romania
herotech.ro



Codrin and Daniel working on the robot code.

herotech team, Romania
herotech.ro

Second task

We basically just programmed the robot to move at the controller action. We have 2 modes that we can change by pressing the Right-Down button on the controller. We needed 2 modes because we didn't have enough buttons on the

controller to do everything we wanted. For the first mode, we consider the front

of the robot where the big-claw module is. Using the A-B joystick we can make

the robot move and turn. Using the C-D joystick we control the big claw. In the second mode, we consider reverse all the movement controls, and we can control

the extensible arm, the lifting and the clutching of the figurine.

Motors and sensors configuration

```
#pragma config(Sensor, port4,           distanceSensor,        sensorVexIQ_Distance)
#pragma config(Sensor, port12,          distanceSensor2,       sensorVexIQ_Distance)
#pragma config(Sensor, port5,           gyroSensor,            sensorVexIQ_Gyro)
#pragma config(Sensor, port10,          distanceSensorArm,     sensorVexIQ_Distance)
#pragma config(Motor,  motor1,          armMotor1,             tmotorVexIQ, openLoop, encoder)
#pragma config(Motor,  motor2,          leftMotor,             tmotorVexIQ, openLoop, encoder)
#pragma config(Motor,  motor3,          rightMotor,            tmotorVexIQ, openLoop, reversed, encoder)
#pragma config(Motor,  motor6,          armMotor2,             tmotorVexIQ, openLoop, encoder)
#pragma config(Motor,  motor8,          armMotor3,             tmotorVexIQ, openLoop, encoder)
#pragma config(Motor,  motor9,          armMotor4,             tmotorVexIQ, openLoop, encoder)
```

This is the first part of our code, the part where we create a connection between the Robot Brain ports and the actual electric components, also here is the place where we name them.

Robot movement

```
float wheelDiameter = 6.4;

void moveCm(float distance)
{

    float circumference = PI * wheelDiameter;
    float degrees = (distance * 360) / circumference;

    resetMotorEncoder(leftMotor);
    resetMotorEncoder(rightMotor);

    moveMotorTarget(leftMotor, degrees, 100);
    moveMotorTarget(rightMotor, degrees, 100);

    waitUntilMotorStop(leftMotor);
    waitUntilMotorStop(rightMotor);
    wait1Msec(100);


}
```

Because there isn't any function that can tell the robot to move by x centimeters or by y inches, we wrote our own so we can reuse it as many times as we would like in our program. By using the provided *moveMotorTarget*method we can rotate a motor. Its parameters are: the reference to a motor, the number of degrees we want to rotate the motor and the speed of the rotation which is between 1 and 100. So before calling this method in our *moveCm*function we need to figure out how many degrees the motors should rotate based on the distance we want to move the robot with. We used the following formula to calculate the number of degrees:

$$degrees = \frac{\text{distance} * 360}{\text{PI} * \text{wheelDiameter}}$$

Where distance is the *distance* we want to move the robot in centimetres and *wheelDiameter* is the diameter of our robot's wheels which we measured and is 6.4 centimetres.

Robot Rotation

```
void rotate(int toRotateDegrees)
{
    resetGyro(gyroSensor);

    int direction = 1;

    if(toRotateDegrees < 0)
    {
        direction = -1;
    }

    if(direction == 1)
    {
        int degrees = getGyroDegrees(gyroSensor);

        while(degrees <= toRotateDegrees)
        {
            degrees = getGyroDegrees(gyroSensor);

            if(degrees <= toRotateDegrees - 20)
            {
                setMotorSpeed(leftMotor, (-80 * direction));
                setMotorSpeed(rightMotor, (80 * direction));
            }
            else
            {
                setMotorSpeed(leftMotor, (-10 * direction));
                setMotorSpeed(rightMotor, (10 * direction));
            }

        }

    }
    else if(direction == -1)
    {
        int degrees = getGyroDegrees(gyroSensor);

        while(toRotateDegrees <= degrees)
        {
            degrees = getGyroDegrees(gyroSensor);

            if(toRotateDegrees + 20 <= degrees)
            {
                setMotorSpeed(leftMotor, (-80 * direction));
                setMotorSpeed(rightMotor, (80 * direction));
            }
            else
            {
                setMotorSpeed(leftMotor, (-10 * direction));
                setMotorSpeed(rightMotor, (10 * direction));
            }

        }

    }

    setMotorSpeed(leftMotor, 0);
    setMotorSpeed(rightMotor, 0);


    wait1Msec(100);
}
```

Same for the *rotate* function. We wrote it so we can easily call it from our main program. This function rotates the robot by moving the wheels in opposite directions, with the same speed.

This function takes a single parameter that is the number of degrees we want to rotate the robot with. It can be a positive or a negative number between (-359, 359). The direction of the rotation is given by the number's sign(+/-). The robot will turn to the left if the number is positive and to right if the number is negative. We use the gyroscope sensor for this function. In a while loop we set the motors rotation speed as long as the gyro value isn't equal to the number of degrees we want to rotate with. Also, for the last 20 degrees of the rotation the speed is lowered from 80 to 10 because the omnidirectional wheels that we use have 0 friction with the ground, so when the turn ends, the robot slides some more. By lowering the rotation speed of the last 20 degrees, the rotational momentum is low enough that the robot will stop right when the turn ends. If by any cause the robot turns more than the set degree, it will correct the position after checking the gyroscope data. Also, if the robot doesn't move in a straight line when it should, with the data gathered from the gyroscope, the robot will correct its position when the moving stops.

Task 1 multitasking

```
bool detectedSmth = false;
startTask(Move130Cm);

int encoderValueMinDistance = 0;

while(true)
{
    int distance = getDistanceValue(distanceSensor);
    if(distance < minDistance)
    {
        encoderValueMinDistance = getMotorEncoder(leftMotor);
        minDistance = distance;

        detectedSmth = true;
    }

    if(detectedSmth && stopedTask)
        break;

}
```

For the first task we need to do two things simultaneously: move the robot and read values from the distance sensor. To do this, we used the integrated multitasking option in robotC. In our main program we start a task that moves to robot 130 cm in front and after the movement is done changes a global boolean value from false to true, so we know that the task is done.

Variables

Because we will have to do some last second adjustments, the code for the first task was created to be able to change the value of variables all in one place, changing nearly all the parameters of the way the robot moves.