Hephaestus

HEROTECH

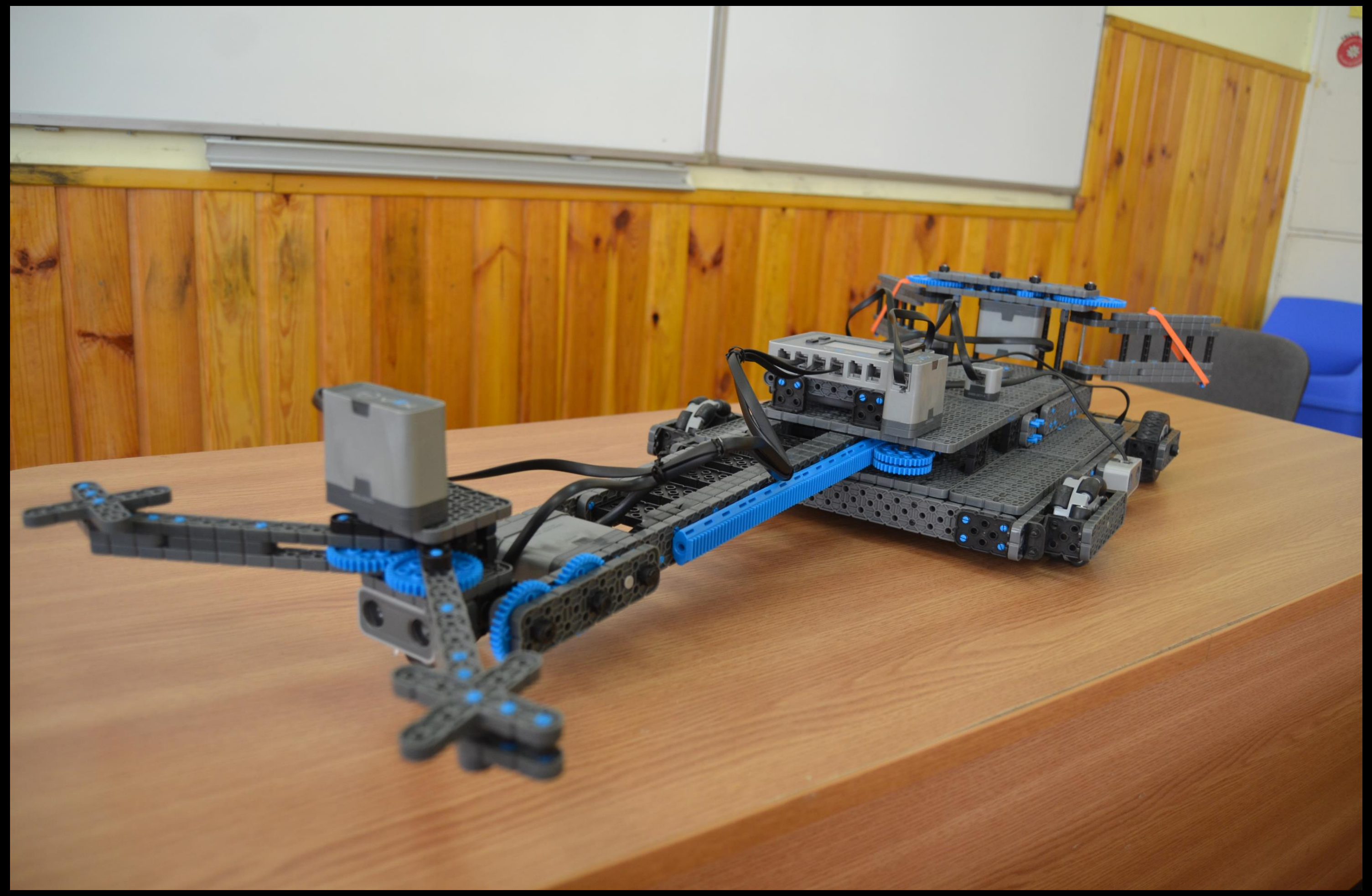
Alexandru-Ovidiu Rădac, Daniel Marpozan, Codrin Muntean | National College “Octavian Goga” | Sibiu, Romania

INTRODUCTION

Our team has worked together for an extended period of time to design a robot that would complete the tasks that were given and would be worthy of GENIUS Olympiad. As so, we worked with VEX IQ pieces to assemble the robot and we used C++ to program it.

The base idea behind this 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.

ROBOT OVERVIEW

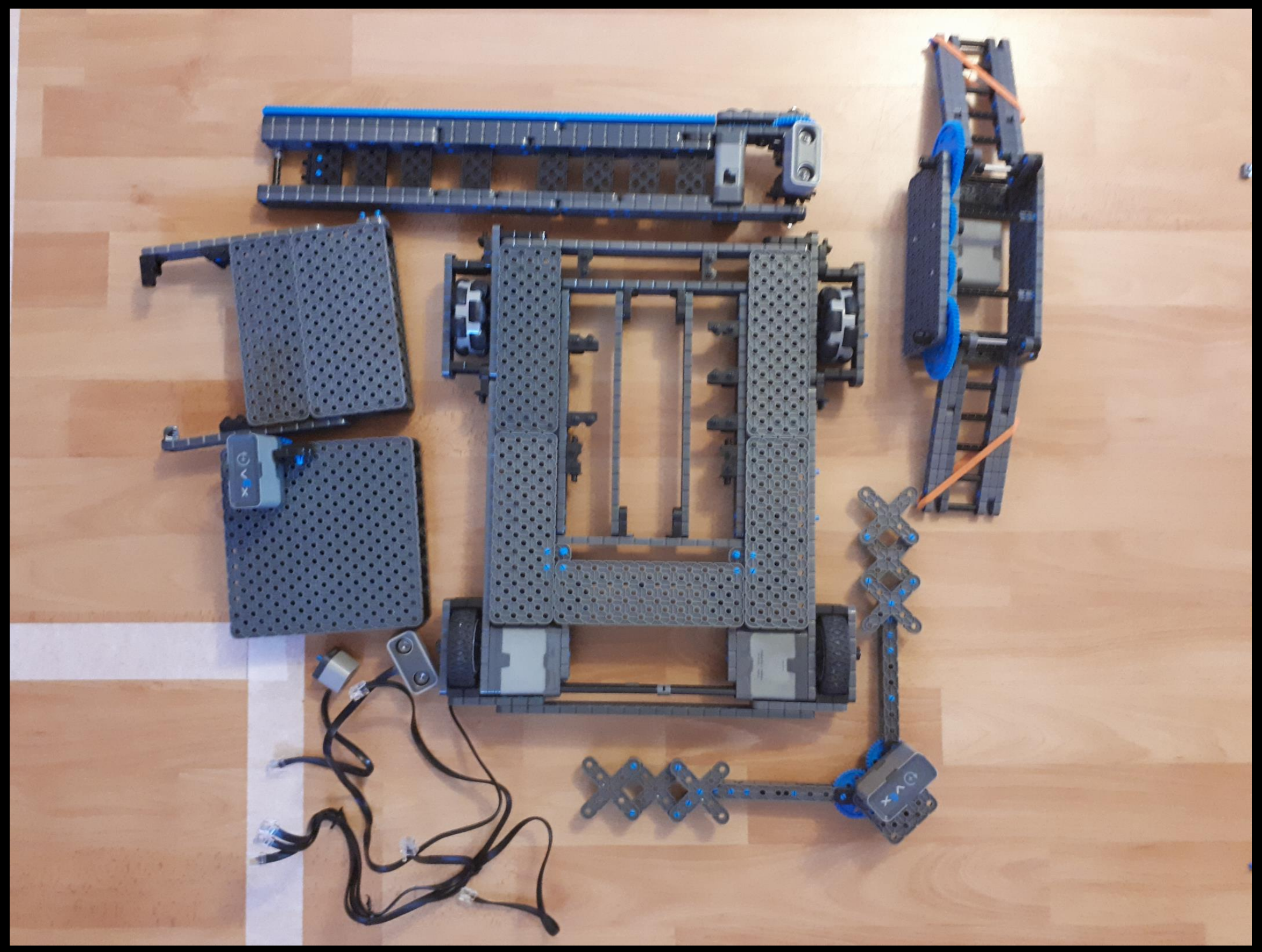


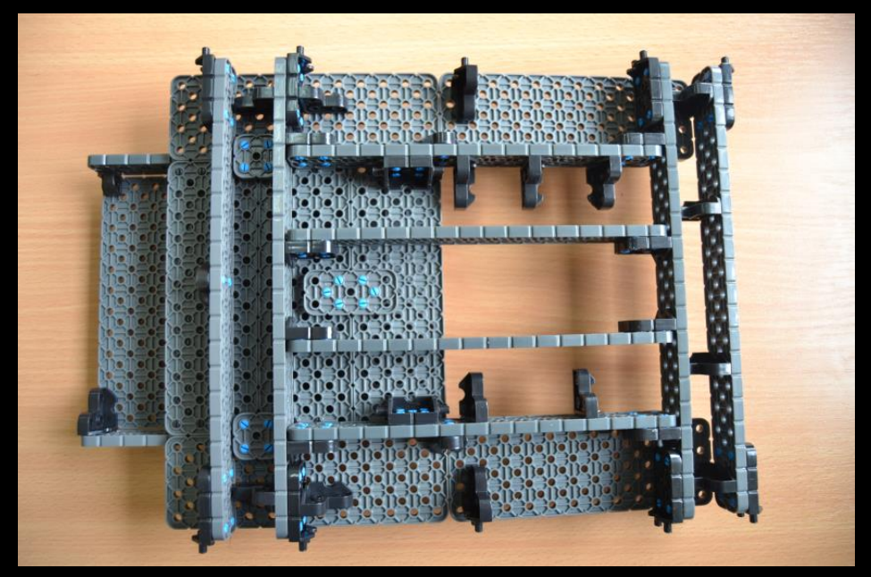
Hephaestus has been designed to be as modular as possible, allowing us to transport it easily overseas and to troubleshoot problems that may occur, as fast as possible.

The modules were designed by Alex to be able to complete the necessary tasks as fast as possible and they were built by Daniel according to the schematics. For us, the designing / building phase was the longest.

The code for Hephaestus has been written to have a low error rate for the first task. This was one of the most challenging parts of the project. For the second task, writing the code was pretty straight forward.

MODULES





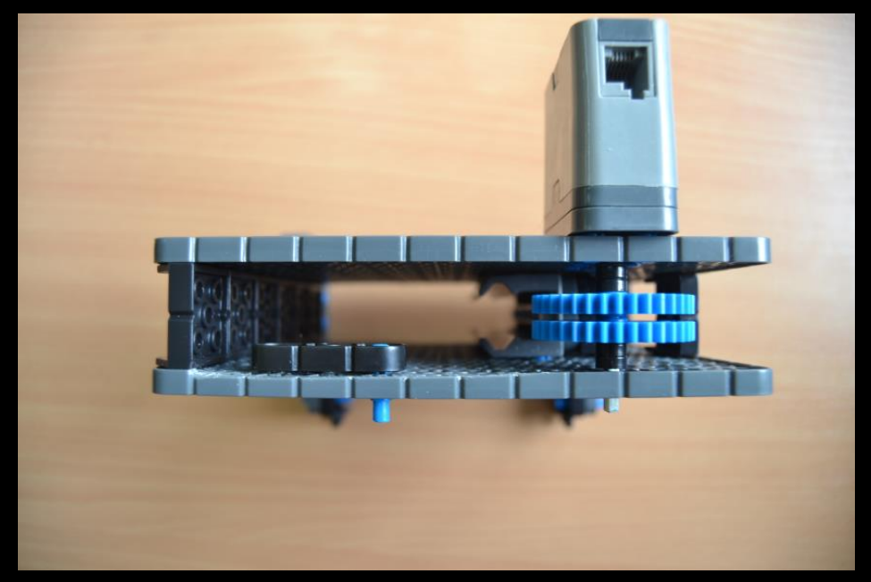
- Module #1: Base / resistance structure of the robot:

This module is the biggest one, and with it we connect all the other modules. As it can be seen in the picture, multiple long and flat VEX pieces have been placed at a 90° angle. This makes the robot feel more solid and "wiggle" less at sharp turns. Also this prevents the robot from collapsing under its own weight.

- Module #2: Big claw:

This module is composed of one motor, 4 gears that move the part that holds the cube, and some beams.

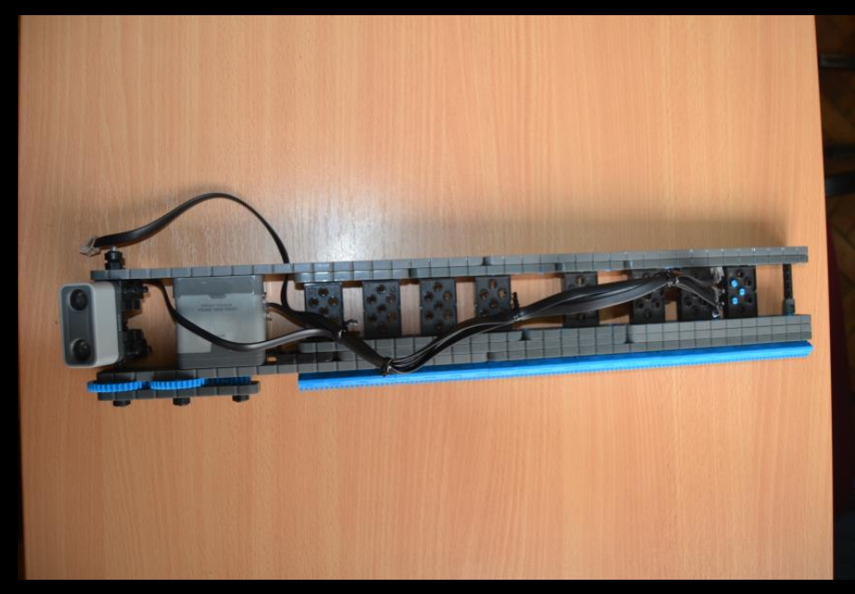
The orange elastic bands are used to create more adhesion between the arms and the cube, to make it easier for us to transport it.



- Module #3: Long arm housing:

This module is quite a simple one, it has two big and flat pieces, a set of rails that direct the long arm and a motor connected directly to two gears that move the arm.

It was important for us to make this as strong as possible because it has to support the weight of the whole long arm.



- Module #4: Long arm:

The long arm is the most important module of the robot. We use it both for the first and second task. It is essentially an extensible arm that is connected to module #3 and module #5.

Also, at the end we have placed a distance sensor that we use in the first task.

- Module #5: Small claw:

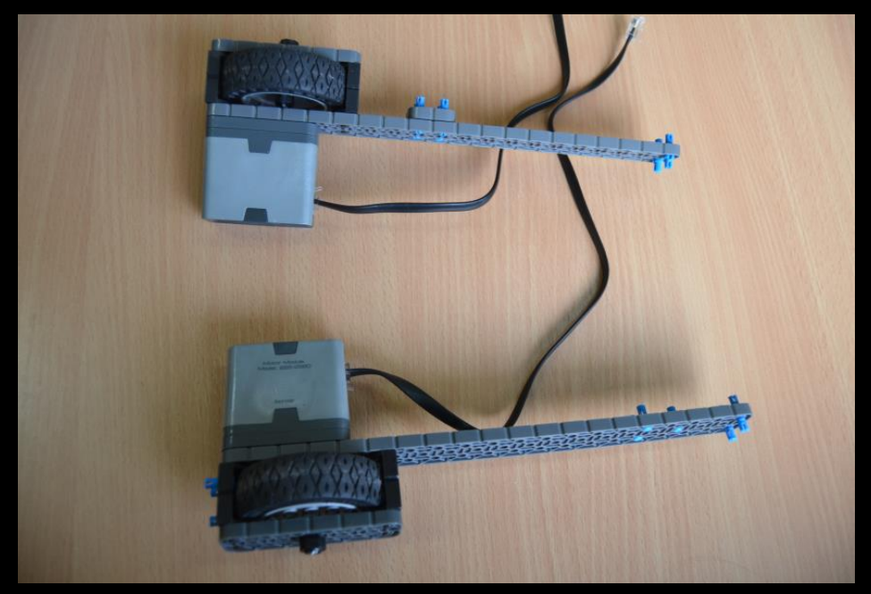
This claw went through a lot of modifying as we had a some problems with the first designs.

Now we use it in both task as the piece that holds the wood figurine. It has an elongated design so we can reach with further figurines in the second task

- Module #6: Robot brain;

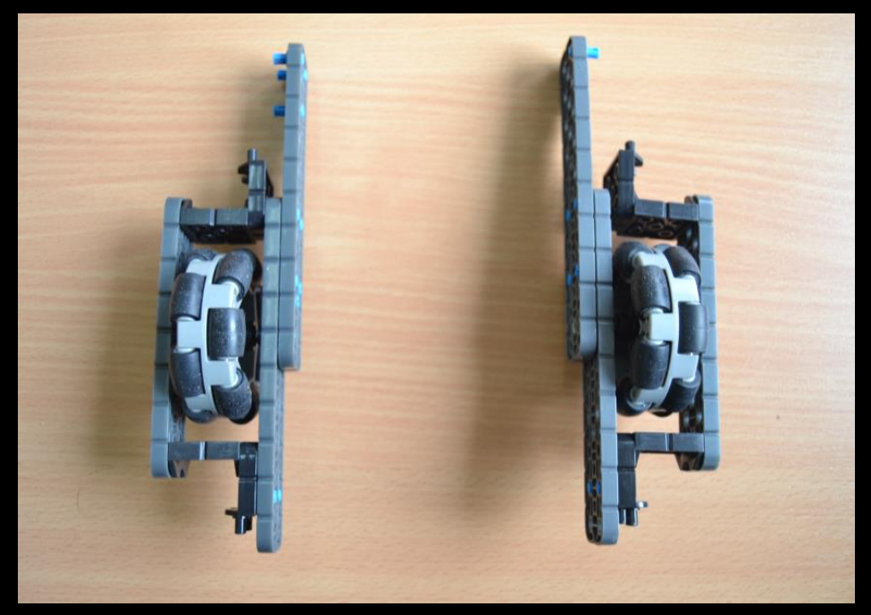
This one is not actually a module built by us, but one "customized" by us. It is composed from the Robot Brain, one gyroscope and two distance sensors.

The Robot Brain is also the place where we uploaded the code for the robot, and where all the sensors and motors are connected to.



- Module #7: Motor wheels:

This module is a simple one. We connected the wheels to the motors with shafts, built an enclosure for them to have better stability and we connected everything to two long beams that are connected to module #1.



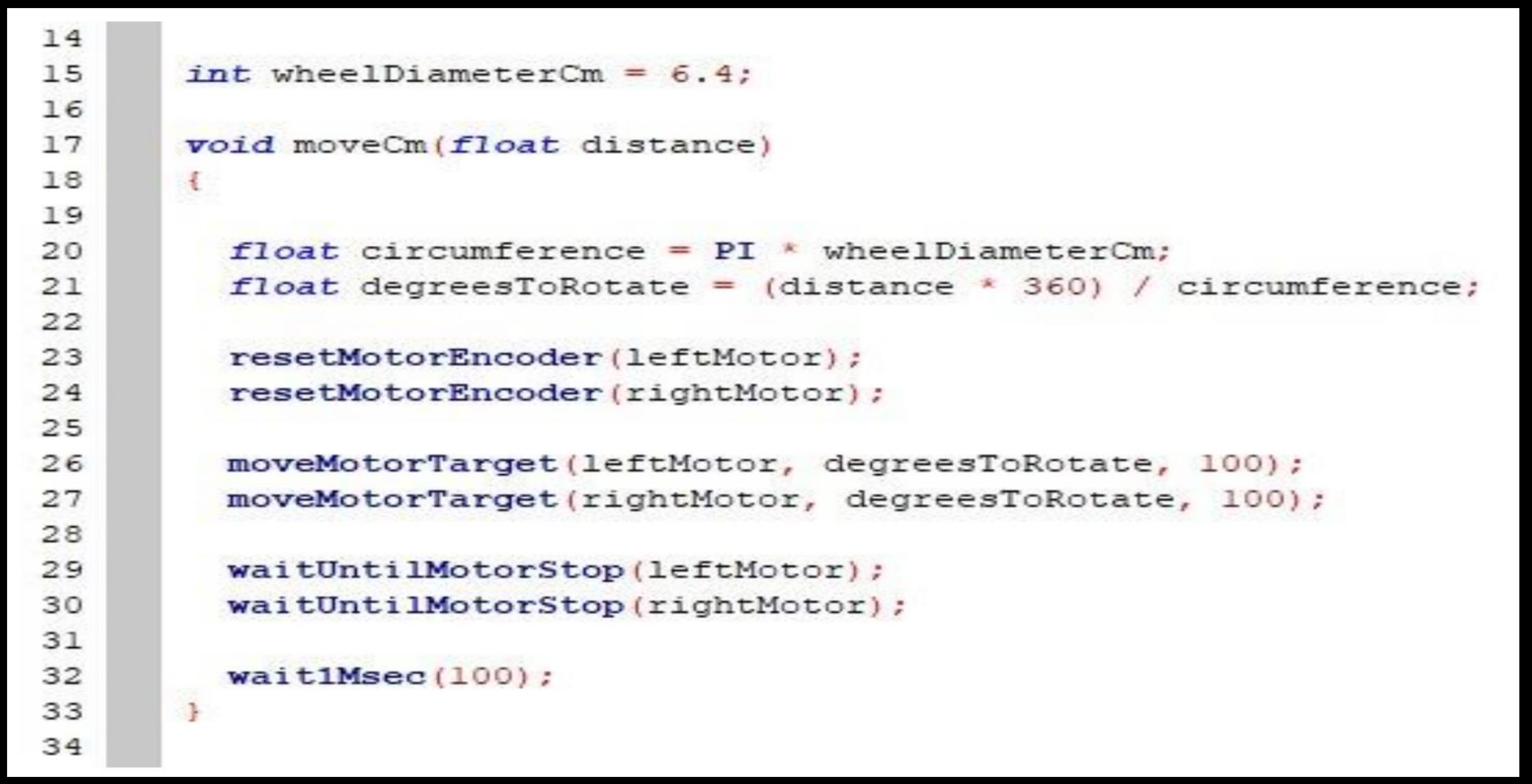
- Module #8: Omnidirectional wheels:

This module is similar to module #7, but here we have two omnidirectional wheels instead of the rubber ones. This allows the robot to take the turns more easily, as the friction is minimised when the wheels are moving laterally.

Also, this module doesn't have the two motors.

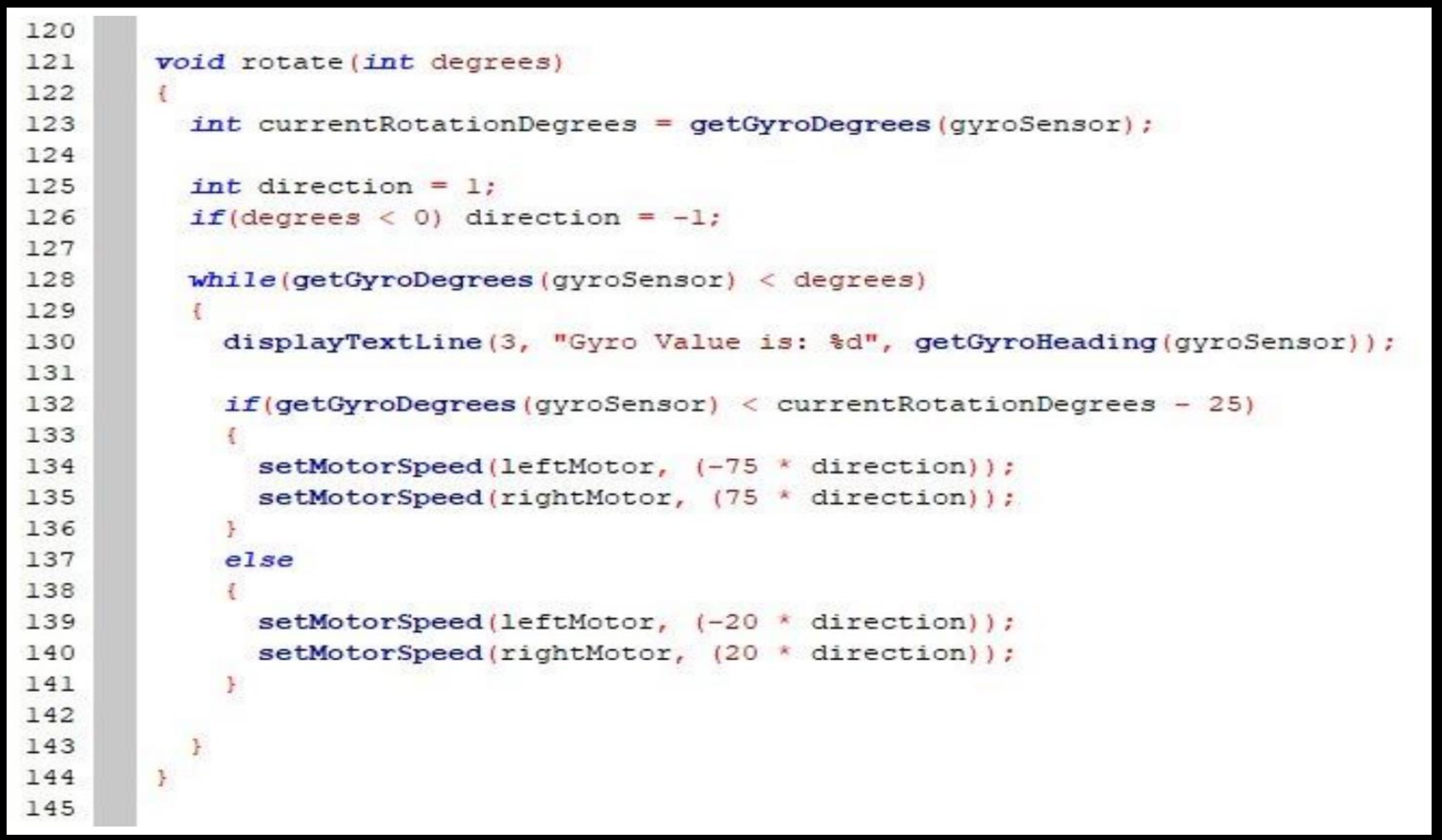
PROGRAMMING

We used the RobotC IDE provided on the VEX IQ website. Because we already had a little bit of experience in the programming domain, we opted to use the C++ language instead of the visual language. The coding was done in completely by Codrin and it was the last part of the project.



```
int wheelDiameterCm = 6.4;

void moveCm(float distance)
{

  float circumference = PI * wheelDiameterCm;
  float degreesToRotate = (distance * 360) / circumference;

  resetMotorEncoder(leftMotor);
  resetMotorEncoder(rightMotor);

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

  waitUntilMotorStop(leftMotor);
  waitUntilMotorStop(rightMotor);

  wait1Msec(100);
}

```

This function transforms an input distance from centimeters into rotation degrees, that the motor can use. The formula is *(distance_variable*360)/(PI*6.4)*.



```

void rotate(int degrees)
{
  int currentRotationDegrees = getGyroDegrees(gyroSensor);

  int direction = 1;
  if(degrees < 0) direction = -1;

  while(getGyroDegrees(gyroSensor) < degrees)
  {
    displayTextLine(3, "Gyro Value is: %d", getGyroHeading(gyroSensor));

    if(getGyroDegrees(gyroSensor) < currentRotationDegrees - 25)
    {
      setMotorSpeed(leftMotor, (-75 * direction));
      setMotorSpeed(rightMotor, (75 * direction));
    }
    else
    {
      setMotorSpeed(leftMotor, (-20 * direction));
      setMotorSpeed(rightMotor, (20 * direction));
    }


  }
}

```

This function makes the turns go slower when they reach the final 25 degrees, so the robot will loose its angular momentum and will not slide when the rotation is complete.

TASKS

1. FIRST TASK:

The algorithm we are using for this task was designed by Alex and coded by Codrin. After the timer stars the robot moves forward, constantly using the distance sensor mounted on its side to detect the wood figurine. When it detects it, Hephaestus takes a 90° turn to the right and moves forward until the figurine is detected by the frontal distance sensor at 14cm. After that, the small claw closes and lifts the figurine up. The robot moves to the hospital, where it places the figurine and then returns to the robot lot.

2. 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 reversed all the movement controls, and we can control the extensible arm, the lifting and the clutching of the claw.