9721
Robot Presentation
Cailin Fitzgerald,
Arielle Greenblat
Madeline Greenblat

About team 9721

- We were team 3958 for FLL and we won the “Champions” award for the 2022 season!
- This is Cailin’s 5th year of FIRST (2 years of Jr FLL and 3 years of FLL) and her first season of FTC.
- Madeline and Arielle are in their first year of FIRST.
- Our team was founded in 2015 when it included Cailin’s older brother, Luke. He was a Dean’s List Finalist in 2021 and the team won second place for the Connections Award and the Think Award. The team has won the “Best of Vermont” award twice and the Think Award (2020). The team was an Alliance Captain for the Semi-Finals in 2019.

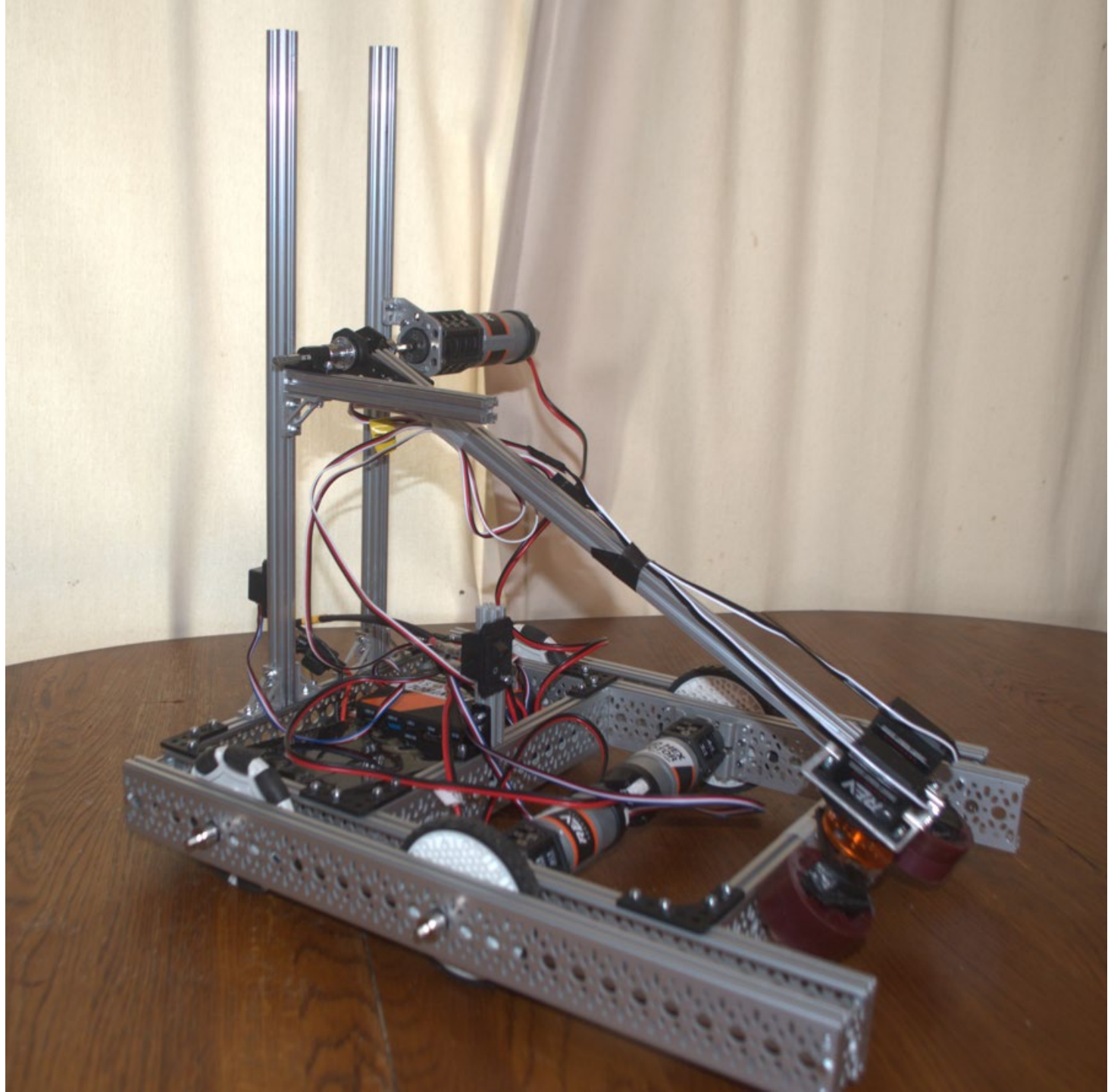
Season Plan

- We are in our rookie season.
- We planned to re-use equipment from last year’s teams.
- We planned to be mentored by team 16295, which is all-female.

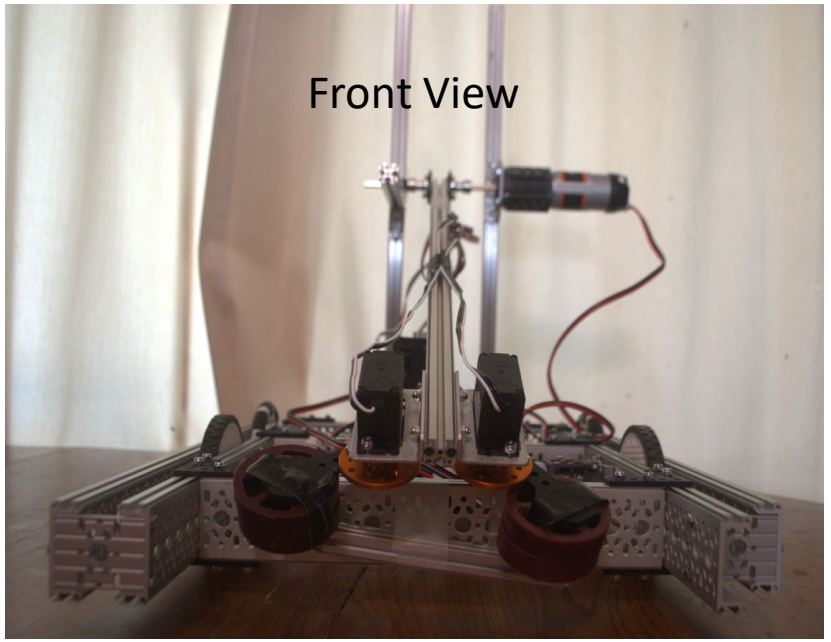
Freight Frenzie Goals

- First Plan(43)
- End Game
  - Cap the alliance tower (place it instead of cube at the start) (15)
- TeleOp
  - Place 3 cubes (from start) on the alliance tower top level (18)
- Autonomous
  - Drive and park to storage area (6)
  - Deliver 2 cubes to storage area(4)

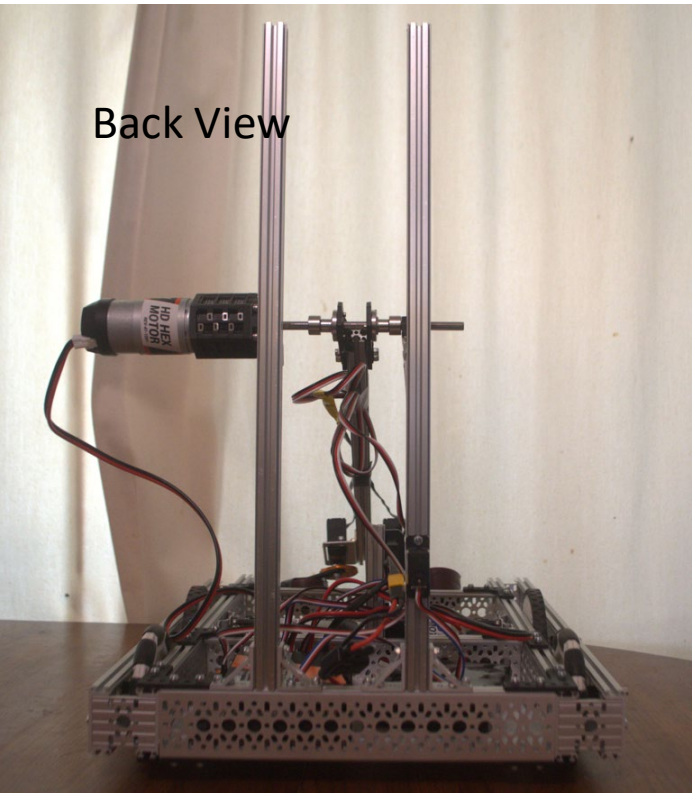
- Alternate Plan (36)
  - Tip the shared tower (20)
- TeleOp
  - Place cubes on the shared tower (6)
- Autonomous
  - Drive and park to storage area(6)
  - Deliver 2 cubes to storage area(4)



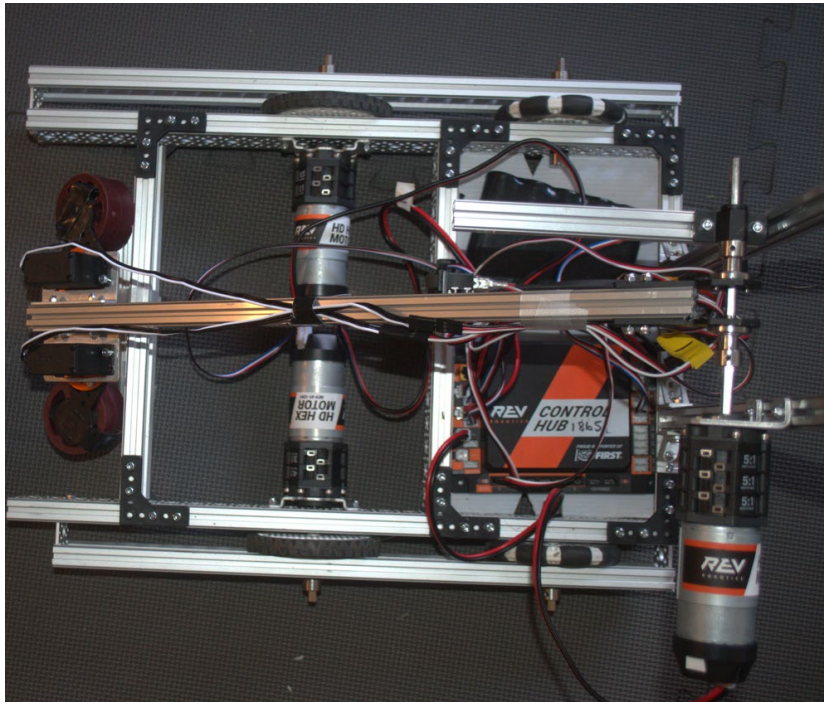
Front View



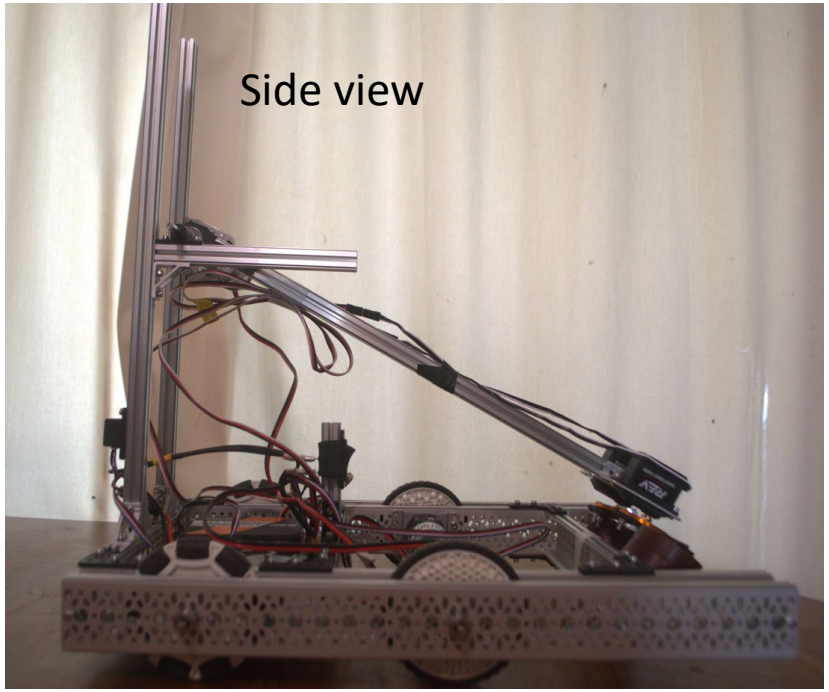
Back View



Top View



Side view



Robot Design-Chassis

- Robot Chassis was based on 2021 REV prototype (no longer available) that was used by team 18651 last year.
- We modified the chassis by removing the front omni wheels. We made this change to allow the robot to push freight around the game field.
- We used two ultra-planetary motors. We had planned to use the older 40:1 gear motors but the axle was not long enough to reach to the second C channel of the robot design.
- The motors were geared using a 12:1 ratio (3:1 and 4:1 cartridges). A longer axle was placed using the set screw so that the axle could span the distance across the channels to keep the wheels stable.
- The front bumper was moved forward to put the cubes in the correct position for collection.
- The side bumpers prevent material from getting stuck under the robot.

Robot Design-Electronics

- A floor was built by placing plastic corrugated board between the back C Channels. This creates a safe space for the control hub and battery
- The motors are attached left to right, 0-2 ports.
- The servos are attached left to right, 0-1 ports.
- A switch was mounted to turn the robot off and on.

Driver Control of Chassis

- The drive code was used both teams 16295 and 18651 last year.
- The method to drive the robot places the controls on the dpad and bumpers. This was simple and intuitive for the operators.
- The control pattern using a series of if else statements to tell the robot what to do.

Robot Design-Arm and Gripper

- The arm design was taken from the design of the arm on team 16295 from last year. The arm used three 5:1 cartridges along with setting the default 0 behavior to brake to operate the arm.
- The arm design was modified to add a second post to stabilize the arm.
- The second post was modified to align the axle on both posts.
- The gripper uses two servos to grab cubes. Cubes were selected because they are more numerous in the game and they do not roll away when you try to grab them.
- The arm collects cubes on one side of the robot and drops them on the other side of the robot. This means that the robot does not need to turn to collect items.

Driver Control Code

Code is taken from samples and modified to suit our needs.

Most of the driver code was mapped to gamepad 1 and was developed last year, including the following methods: DriveAxialForward, DriveAxialBackward, DriveRotateLeft,DriveRotateRight

The arm is controlled with direct commands, which are mapped to gamepad2 and gamepad1 for solo operation.

Autonomous Methods

Autonomous methods were developed
By collecting data on the distance the robot traveled
And the number of encoder clicks.

These values were graphed using excel.
A trend line was added to determine an equation
Between clicks and distance.

The equation was solved in reverse, to calculate the number of
Clicks for a specific distance.

The equation was entered into the program.

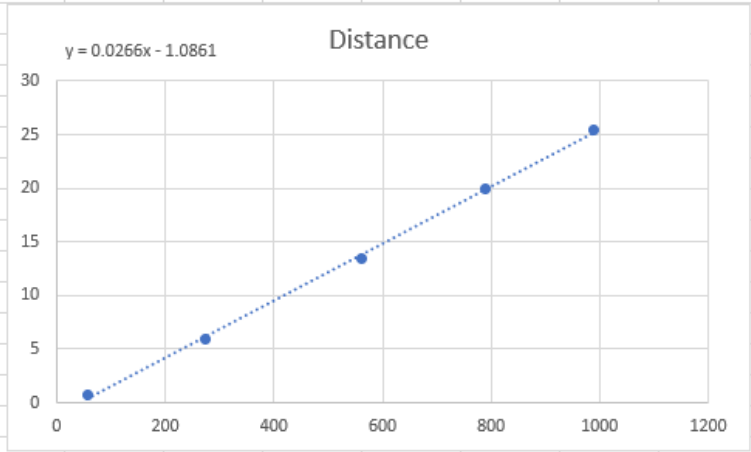
We used trial and error once the method was created to fine tune performance.

```
public void encoderDrive(double distance, double power){
    double start = driveLeft.getCurrentPosition();
    //7% error
    //double encoderAmount = (-100/4)*(distance-1.3);

    //double encoderAmount = (25.8)*(distance+1.3325);

    double encoderAmount = distance/0.0279*10/16;
    double end = start+encoderAmount;
    while(driveLeft.getCurrentPosition() < end){
        driveAxialBackward(power);
    }
    driveAllStop();
}
```

Left Moto	Distance
57	0.8
275	6
560	13.4
790	20
989	25.5



Autonomous Methods-Turning

- We used the sample program to determine which angle was the heading.
- We copied and pasted the important code from the sample to make a while loop that allowed the robot to turn right or left until it met or exceeded the target.
- Left turns are positive toward 90. Right turns are negative toward 90. This was discovered by trial and error with telemetry.

```
public void imuRotateRight(double angle, double power){
            angles   = imu.getAngularOrientation(AxesReference.INTRINSIC, AxesOrder.ZYX, AngleUnit.DEGREES);
            while(angles.firstAngle > angle){
                angles   = imu.getAngularOrientation(AxesReference.INTRINSIC, AxesOrder.ZYX, AngleUnit.DEGREES);
                driveRotateLeft(power);
            }
            driveAllStop();
            telemetry.addLine("The heading is..." + angles.firstAngle +"");
}

public void imuRotateLeft(double angle, double power){
            angles   = imu.getAngularOrientation(AxesReference.INTRINSIC, AxesOrder.ZYX, AngleUnit.DEGREES);
            while(angles.firstAngle < angle){
                angles   = imu.getAngularOrientation(AxesReference.INTRINSIC, AxesOrder.ZYX, AngleUnit.DEGREES);
                driveRotateRight(power);
            }
            driveAllStop();
            telemetry.addLine("The heading is..." + angles.firstAngle +"");

}
```

Autonomous Program

- Measured distances for initial values
- Trial and error to tune program

BlueAuto

```
encoderDrive(24, power);
imuRotateRight(-65, power)
encoderDrive(28, power);
```

```
//This organized the app and places the files in the teamcode section.
//This would be important if you wanted to do some more complicated
//programming and wanted access to a environment such as Android Studio.

package org.firstinspires.ftc.teamcode;

//These next few statements add specific coding to this class so that you
//can access the build-in methods of these other classes.

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
import org.firstinspires.ftc.robotcore.external.navigation.DistanceUnit;
import com.qualcomm.robotcore.hardware.DistanceSensor;
import org.firstinspires.ftc.robotcore.external.navigation.Position;
import com.qualcomm.robotcore.hardware.Servo;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;


//copied and pasted from sample imu program

import com.qualcomm.hardware.bosch.BNO055IMU;
import com.qualcomm.hardware.bosch.JustLoggingAccelerationIntegrator;

import org.firstinspires.ftc.robotcore.external.Func;
import org.firstinspires.ftc.robotcore.external.navigation.Acceleration;
import org.firstinspires.ftc.robotcore.external.navigation.AngleUnit;
import org.firstinspires.ftc.robotcore.external.navigation.AxesOrder;
```

Red Auto

```
encoderDrive(24, power);
imuRotateLeft(65, power);
encoderDrive(24, power);
```

```
//This organized the app and places the files in the teamcode section.
//This would be important if you wanted to do some more complicated
//programming and wanted access to a environment such as Android Studio.

package org.firstinspires.ftc.teamcode;

//These next few statements add specific coding to this class so that you
//can access the build-in methods of these other classes.

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
import org.firstinspires.ftc.robotcore.external.navigation.DistanceUnit;
import com.qualcomm.robotcore.hardware.DistanceSensor;
import org.firstinspires.ftc.robotcore.external.navigation.Position;
import com.qualcomm.robotcore.hardware.Servo;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;


//copied and pasted from sample imu program

import com.qualcomm.hardware.bosch.BNO055IMU;
import com.qualcomm.hardware.bosch.JustLoggingAccelerationIntegrator;

import org.firstinspires.ftc.robotcore.external.Func;
import org.firstinspires.ftc.robotcore.external.navigation.Acceleration;
import org.firstinspires.ftc.robotcore.external.navigation.AngleUnit;
import org.firstinspires.ftc.robotcore.external.navigation.AxesOrder;
```

TeleOp

```
//This organized the app and places the files in the teamcode section.
//This would be important if you wanted to do some more complicated
//programming and wanted access to a environment such as Android Studio.

package org.firstinspires.ftc.teamcode;

//These next few statements add specific coding to this class so that you
//can access the build-in methods of these other classes.

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
import org.firstinspires.ftc.robotcore.external.navigation.DistanceUnit;
import com.qualcomm.robotcore.hardware.DistanceSensor;
import org.firstinspires.ftc.robotcore.external.navigation.Position;
import com.qualcomm.robotcore.hardware.Servo;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;


//copied and pasted from sample imu program

import com.qualcomm.hardware.bosch.BNO055IMU;
import com.qualcomm.hardware.bosch.JustLoggingAccelerationIntegrator;

import org.firstinspires.ftc.robotcore.external.Func;
import org.firstinspires.ftc.robotcore.external.navigation.Acceleration;
import org.firstinspires.ftc.robotcore.external.navigation.AngleUnit;
import org.firstinspires.ftc.robotcore.external.navigation.AxesOrder;
```

Arm Control

- The arm controls were placed on the buttons. This allows for a single gamepad to control the whole robot if necessary.
- The up/down controls are on the vertical buttons. The open close gripper controls are on the horizontal buttons.
- The main issue for operators is that the controls determine the movement of the motor. When the arm is past vertical, the controls that used to make the motor lift cause the motor to fall and vice versa. This confused the operators.

Cap Design

- The team design a cap using the cube as a template because the robot arm could already pick up a cube.
- The remaining shape resulted from experiments with recycled containers. The team wanted something that would easily work.
- The only issue with the cap design is that the attached cube makes the cap lean to one side. This could be a problem depending upon the cap designs of other teams. It makes sense for the robot to not double cap.

Chassis Testing

- The robot drives through obstacles
- The robot can drive inside the barriers (it was designed to be less than 13.5” wide) but it cannot drive over them.
- The robot needs at least .4 power to operate because of friction with the rubber floor.

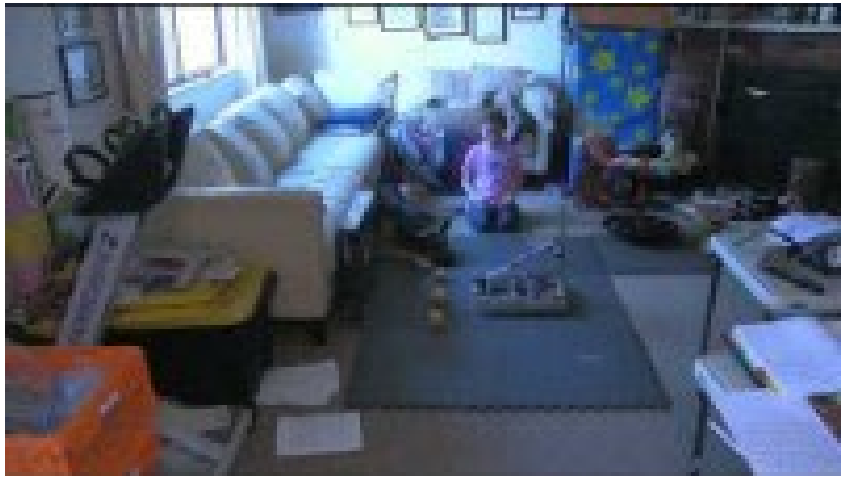
Arm Testing

- The arm can lift all of the cubes to the highest alliance tower
- The arm can lift the cap and place it on the towers
- The arm needs the tower to be leaning toward it to place it while drive forward
- The arm can place the cap on the tower from the backwards position it needed

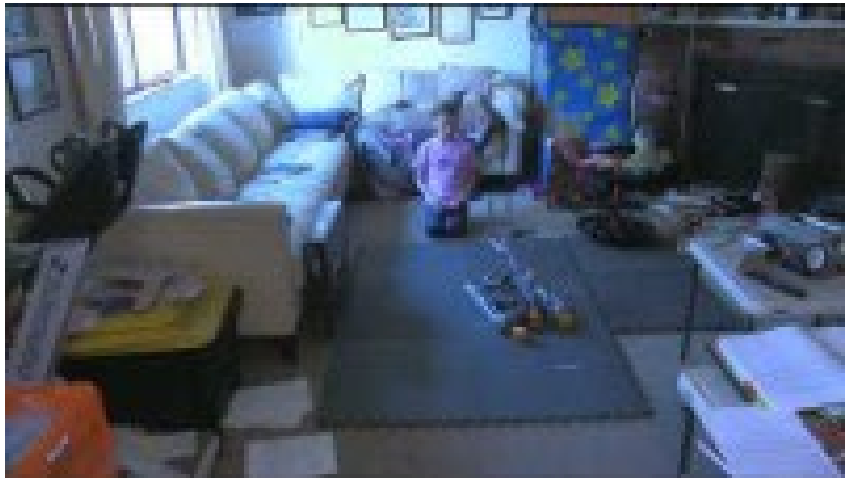
Autonomous (12 pts)

- The robot can drive and park from the blue position, but the team has to coordinate with the other alliance team because it can only do his from a single starting position. The team only needs ()seconds, so it could be delayed.
- The autonomous also delivers cubes as it moves because of the chassis design, which collects them like a dump truck along the way.

https://youtu.be/5j20gT4b33A Blue Autonomous



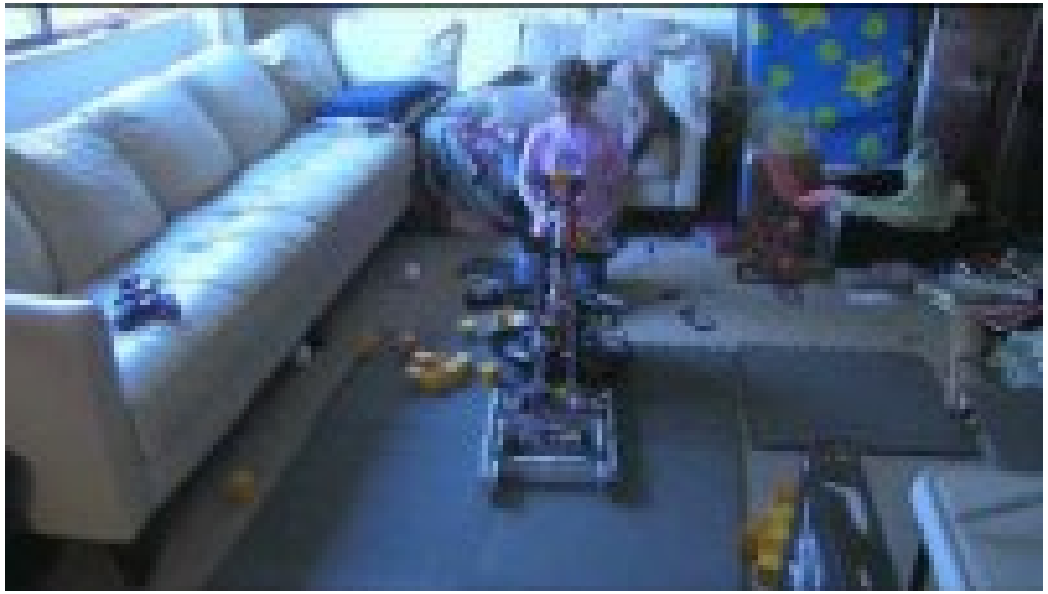
https://youtu.be/Dmbx2xaWvK8 Red Autonomous



TeleOp testing(18 pts)

A single operator can place (3) cubes on the Highest tower for 18 points

https://youtu.be/nPe_Eajas7E TeleOp Testing



End Game and Cap

Robot places the team designed Element on the alliance tower

https://youtu.be/E2h96wfccPM