2020 Tech Binder

Boba Bots 253

Tea-3PO

Table of Contents

Game Summary

Auto

The autonomous phase of play consists of robots attempting to score points solely through code and without driver control. Robots start on the side of the field opposite from the alliance driver station, and gain points by leaving the initiation line or scoring power cells into ports. During this 15-second phase, all points are doubled in value.

Teleop

The teleop phase of play has the longest duration among the phrases, it lasts for a total of 2 minutes and 15 seconds, during which driver-controlled robots score points. During this phase, drivers remotely operate robots to retrieve and score power cells in goals and manipulate control panels to activate different stages of the "shield generator".

There are three stages:

- Stage 1: Robots must score 9 power cells
- Stage 2: Robots must score 20 power cells in Stage 2 and complete rotational control.
- Stage 3: Robots must score 20 power cells and complete position control.

Endgame

The end game phase of play consists of the last 30 seconds of the match. During this phase, robots attempt to "energize" the "shield generator" by hanging onto a bar, which functions similar to a scale, at the center of the field. Points can be earned through a successful climb or a balance of multiple alliance robots.

Design Priorities

Our goal for the 2020 season was to make it to playoffs at a regional level, and our designs were influenced by that goal. During the design process, we considered the capabilities of each member of a playoff alliance (Captain, 1st Pick, 2nd Pick) as well as what was achievable for our team to accomplish, they are as follows:

Alliance Captain:	First Pick:	Second Pick:
Climb power switch with leveling/even out capability/buddy climb	Climb power switch and stay there	Climb power switch and stay there
Spin control panel with both positional and rotational control	Spin control panel with rotational control	
Score high goal nearly every time	Score high goal consistently	Play defense or score on low goal
Score three or more balls in high goal during autonomous period	Score three balls in high goal during autonomous period	Score on low goal during autonomous
Pick up and store five balls at a time	Pick up and store five balls at a time	
Intake balls from the floor	Intake balls from the floor	
Be able to go through the trench	Preferably go through trench	

As a team, we decided that it would be achievable for us to build a robot that would compliment an alliance captain. Here's a breakdown of the criteria for a first pick robot and our thought process behind its implementation and importance:

1. Climbing Mechanism - The endgame climb is the highest single-action scoring opportunity in a match. Also, in order to gain the endgame ranking point, each alliance needed a minimum of two robots hanging on the shield generator. These two factors led us to make the climbing mechanism a top priority for our robot.
2. No Control Panel Mechanism - After discussion regarding how the game would be played, we determined that a spinner mechanism for the Control Panel was of low priority, and only beneficial for gaining ranking-points, which isn't important in playoffs. Therefore, we chose the capability of shooting power cells over a spinner-mechanism during the design process.
3. High Goal Shooter - After a slight shift in our priorities, we chose to design a shooter mechanism that could score high-goal as quickly as possible. Rule G11 in Section 7.2.3 of the 2020 game manual states that any robot within it's target scoring zone is protected from other robots. Because of this rule we decided that the safest option to maximize our scoring potential was to manufacture a shooter optimized for shooting within a target zone.
4. Autonomous Shooting - At the start of each match, each robot can hold up to three balls. We decided that it was extremely valuable to take advantage of the autonomous double point scoring period. For that reason, we initially chose autonomous code over the designing of a ground intake. However, the two ended up not being mutually exclusive.
5. Ground Intake - During our initial game discussion we put a ground intake as a lower priority, but later in the season it was determined that a ground intake gave us significant strategic advantages, and would allow us to intake balls anywhere on the field

Mechanical

Drivetrain

Chassis and Wheels

The robot has a 31” x 28” West Coast drive base with six 6” VEXpro traction wheels. We added a 1/16” center drop to decrease the traction in our front and back wheels to allow for smoother pivoting. We found that 6” wheels would give us sufficient ground clearance over the 1” metal tubes in the rendezvous point, allowing us to drive through the rendezvous point without difficulty.

We decided on our chassis dimensions based on our conveyer prototypes, as we realized in order to fit five balls within the robot and fit underneath the trench, we needed to maximize the drive base area. To reduce the distance that the power cells have to travel while maintaining the structural integrity of our chassis, we slotted a 7” section on our front rail that allows for power cells to pass through. The slot allows for power cells to be taken into the robot while still maintaining contact with either the ground intake and conveyor belt at all times without having an open chassis.

Drive Gearboxes

The drivetrain is run on four Falcon 500 motors on two West Coast Products Single Speed Gearboxes geared at a 10.42:1 ratio.

Ground Intake

Prototypes

We spent time identifying the types of the wheels, height of the intake, and distance of the intake from the robot's bumpers. This was done by simply driving a hex shaft with a drill, and varying the wheels and relative distances until we found the combination that would allow the power cells to be popped up into the chassis.

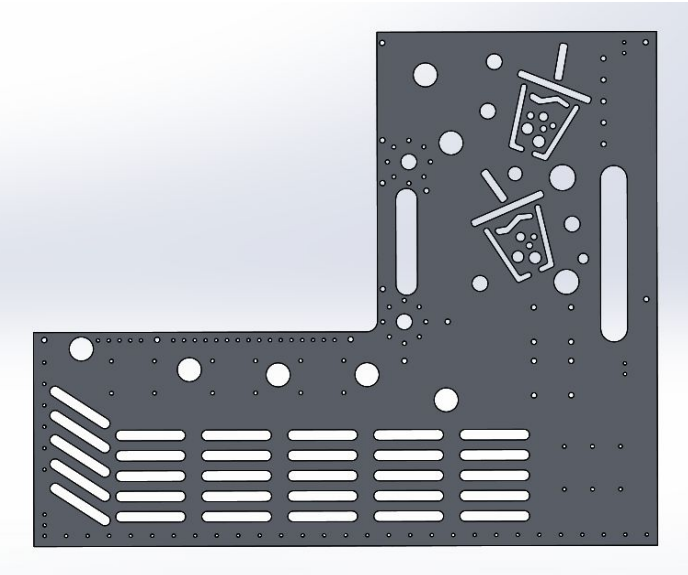
After some testing, we realized that our original intake design needed more support in order to allow both sides to retract into the robot. Because the intake is only powered by a motor on one side, the other side would not follow and retract into the robot, causing an asymmetrical intake. We solved this issue by a set of inner intake arms. With four intake arms, and the inner ones connected by a churro shaft, the intake was then able to move up and down without issues.

We also found that the power cells were jamming between the ground intake and the conveyor. We added driven brushes to the churro support of our intake to help guide the power cells smoothly into the conveyor.

Final Design

The ground intake has ten 3" mecanum wheels on the intake and one center omni wheel, which help center the power cells when they contact the intake. Four polycarbonate intake arms, one pair of inner arms, and one pair of outer arms, support the hex shaft running across the front of the robot. The retraction is powered by a BAG motor and the spinning of the intake wheels is powered by a 775pro. Poly Cord belts run from the driven thunderhex shaft to the churro support, driving pulleys that spin the intake brushes.

Conveyor



Prototypes

The team first discussed the different mechanisms that could both keep five power cells within the robot, be relatively short to allow the robot to drive underneath the trench, and minimize jams. After concerns that a hopper would cause jamming issues, we shifted to designing a conveyor. We tested a shallow gravity driven ramp, but decided that the speed at which the power cells feed into the shooter would be slower than we wanted. We build a wooden prototype of the conveyor, but found that the single NEO brushless motor isn't powerful enough to move the power cells through the system.

Later in the season, we tried to feed more than three power cells into the conveyor, the power cells immediately started jamming in the conveyor. We shifted the placements of our belts, but that only fixed the vertical section of the conveyor. We drilled new holes and shifted all of our driving shafts on the horizontal sections of the conveyor upwards, decreasing the compression. We then added brushes to the horizontal section of the conveyor, as we found that the downwards compression was the cause of the jams.

Final Design

The final design of the conveyor consists of two sections that are separately driven. The horizontal section of the conveyor is run off one NEO, and drives shafts with brushes attached to them. The brushes only lightly contact the power cells, preventing the power cells from jamming. The vertical section is also run off of one NEO, but the power cells are moved upwards by a belt. The conveyor is capable of holding five power cells and accelerating the conveyor to full speed, allowing all five power cells to exit the conveyor quickly.

Shooter

Prototypes

To simplify the power cell manipulating mechanism, we wanted to have our conveyor feed directly into the shooter, using a piston gate or a small feeder wheel to prevent the conveyor from prematurely feeding the power cells.

We had two main prototypes: a hooded shooter and a linear shooter. The linear shooter used two wheels on each side, and created relatively consistent and accurate shots. The hooded shooter also resulted in similar results, and both proved to have enough power to achieve the goals that we had set. However, we decided to go with the hooded shooter, as the hood increased the number of contact points between the mechanism and the power cell. With the linear shooter, since the ball only contacts the mechanism at one point when it is shooting out, we were concerned that any variation on the power cells, such as a tear or hole in the power cell, could drastically decrease the accuracy of our shots.

Final Design

The final shooter design consists of two ¼" thick polycarbonate rails with arcs. Churros connect the rails to the conveyor and to each other for support, and a 3D printed hood attaches to the rails to increase the number of contact points. The shooter wheel is a 3lbs dense Neoprene drive roller, and is driven by 2 NEO motors at a 1:1 ratio. ,

Climber

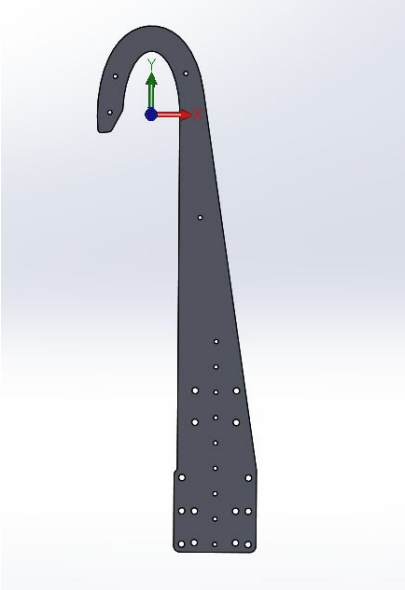
Prototypes

We had three main prototypes for climbing: telescoping cascade, 4-bar arm, and a one point pivot. The telescoping cascade was relatively simple, and was prototyped out of PVC and thin cable. It worked well, but would require us to either sacrifice not being able to fit under the trench, or build at least four stages to reach our target height. The four bar also worked effectively, but was eliminated for its complexity. Finally, the team settled on the one point pivot because it was simple, allowed us to reach our desired height, and was achievable by our team.

Final Design

Our final design is a one point pivoting climb. A vertical climber support is mounted at the corner of each side of the robot, with an aluminum 1x1 tube on each side to act as harms. We also designed a polycarbonate hook to give us extra reach, since with just the climber arms by themselves, we could not reach the level bar. The hooks are 22” long, and made of ¼” thick polycarbonate. 3D printed bearing carriages allow the hooks to slide up and down the climber arms, allowing the hook to be pushed up, and giving us a clearance of approximately 4 extra inches when we climb.

The climber arms are controlled by two 775pro motors, one for each arm. Gas shocks push on the climber arm upwards, and a constant force spring fulls the hooks up. A winch rope holds the arms down, and when the motors actuate, allowing the winch rope to release, both the climber hooks and the arms are raised. To retract the arms, the motor continues to drive in the same direction, and by limiting the length of the winch rope, allows us to drive the motors in one direction but allow the arms to both extend out and retract back to keep the robot level to the ground.



Electrical

Motors and Speed Controllers

For this year, we have decided to implement new components in our control system: the Falcon 500 motor, the REV Spark Max, and REV Neo motors. Our robot has four Falcon 500 motors on the drivetrain, two on each side. We have chosen these mainly for their small size, light weight, efficiency, and built-in encoders. Furthermore, we also have 4 REV NEO motors on our conveyor and flywheel shooter to provide enough torque and power to launch the power cells a significant distance. As for speed controllers, our 2020 robot contains 4 Talon SRXs—2 for climbers and 2 for intake—that allow for the control of motor speed and the toggling of braking and coasting our shooter on the robot.

Sensors

Of course, a robot cannot run without sensors. This year's robot uses magnetic encoders and hall effect sensors, built into the Falcons and NEOs, to regulate the drivetrain and shooter speeds and help with the positioning of our robot on the field. Also, an additional through bore encoder is attached to the arm of our intake, controlling the arm's upwards and downwards movements. In addition to magnetic encoders, photoelectric sensors in the middle of the conveyor and shooter help us detect incoming power cells and tell the conveyor to run. Their main function is to prevent the power cells from jamming inside the conveyor.

Changes This Year

The increased complexity of this year's robot required us to make many changes. Weight limitations and design forced us to evaluate the importance of our components, and many of our past years' systems such as pneumatics were deemed unnecessary. Addressing a prominent issue in past years, we improved cable management by using cable sleeves and polycarbonate plastic plates, which streamlined both wiring and the organization process and protected our components from moving parts. In addition, we've implemented WAGO connectors, facilitating testing and design, for more simplified and easily changeable connections on the CAN bus. Lastly, in an effort to organize and increase efficiency of our wiring from years past, electrical components, like the speed controllers to their corresponding PDP ports, have been labeled and more focus has been put aside for organization and cable management.

Programming

Code Organization

In concordance with the changes in convention brought on by WPILib API 2020, we have changed the way we organize our code, creating a Constants class to store motor IDs and tuning variables for convenient and easy modification while testing, and a RobotContainer class to bind subsystem controls to controller inputs in a more unified and central way. Additionally, we have opted to create our own Units library in order to make the various unit conversions required on our robot more resilient to differences in mechanism design between seasons (such as changes in drivetrain wheel size).

Drivetrain

In order to make our driving more consistent, we chose to characterize our drivetrain, determining the voltages required to overcome static friction, achieve a particular velocity, and account for a particular acceleration. This model, known as a feedforward, was used in reverse to calculate the voltage required to achieve a particular drivetrain velocity. Combining this with knowledge about the robot's track width (distance between the wheels), we were able to achieve constant curvature turning regardless of speed, and thus make our driver practice more effective (as equal inputs always resulted in the same action performed by the drivetrain).

Shooter

This season was our first using the relatively new Spark MAX motor controllers and NEO brushless motors, which was initially an obstacle when we first worked on programming our robot's shooter. Over time, however, we learned the API and were able to implement closed-loop velocity control. This was done by characterizing our shooter (determining the voltages required to begin spinning and to reach any given RPM), and then using this model to determine the voltages required to reach a desired RPM. We also included a PID loop in order to account for the reduction in RPM caused by a power cell moving through the shooter, allowing us to send more voltage and return to shooting speed for the next power cell. We ultimately determined that this level of control was unnecessary, however, as our shooter had enough inertia in order to shoot five power cells consecutively into the goal without significant change in the cells' motion.

Climber

Due to the simple yet effective design of our climbing mechanism, the programming required to run it was minimal -- we simply needed to spin a motor in the correct direction in order to score the points allocated to climbing.

Intake

As our ground intake only has two desired positions, up and down, we initially wrote code that would stall the BAG motor running the intake into mechanical hardstops, holding them in place during rigorous defensive maneuvers. However, we discovered that the current draw created by this code caused the BAG to heat up significantly, an outcome that would reduce efficiency and eventually result in the destruction of the motor.

As a result, we added a REV Throughbore Encoder to measure the angle of the intake, and employed positional closed-loop control to move between positions. To be more specific, we used trapezoidal velocity profiles to translate the arm between the raised and lowered positions, allowing the arm to slowly ramp up and slow down at the beginning and end of the movement, respectively. This reduced the force at which the intake contacted the hard stops, and with the addition of a PID loop, effectively maintained the position of the arm without causing the same current draw issues we experienced prior to this change.

Conveyor

Intaking

Due to the inherent drawbacks of the conveyor belt system (specifically, the propensity to jam balls together beyond the torque limitation of the motors), we found it necessary to implement a power cell serializer. Various sensors such as photoelectrics, beam breaks, limit switches, and LIDARs were considered, but we ultimately decided on photoelectrics for their simplicity and various mounting options. With the change to brushes along the horizontal conveyor, we found that serializing was only necessary for the belt-driven vertical conveyor. By placing a sensor right before the vertical conveyor and one right before the shooter, we were able to avoid three main issues...

1. Jamming into the shooter wheels - The top photoelectric automatically stops the vertical conveyor before a power cell contacts the shooter wheels
2. Jamming within the conveyor - The photoelectric properly queues two power cells into the vertical conveyor with proper spacing, allowing all five power cells to fit into the horizontal and vertical conveyors.
3. Control simplicity - To make the job of driver easier, the queuing system runs on a default command, reducing the need for the driver to manually drive power cells into the vertical conveyor, also avoiding any driver error.

Positional control was considered, but was found to be unnecessary.

Exhausting

To further simplify driver controls, we integrated the conveyor with the shooter when shooting power cells. This feature runs the shooter to the proper speed, and runs the conveyor towards the shooter, all with the press of one button. The conveyor can also be run in the opposite direction, either for feeding other teams power cells or clearing jammed power cells.