StackerBot Whiteboard Brainstorming Challenge

Your time is limited, so do the basics first and then add on if you have time.

There is not a single right answer!

Robot Specifications:

- West coast drive (3 wheels one each side connected by chains like our 2020 robot)
- Elevator that extends 6 feet up controlled by a motor
- Arm that pivots from horizontal to the floor 180 degrees over the top of the robot to horizontal on the other side of the robot. The pivot is controlled using a motor.
- Horizontal claw that opens and closes using a single solenoid
- The claw is on the end of an arm
- The arm is connected to a pivot on the elevator
- The arm can swing over the top of the elevator and go down the backside

Game Pieces: 12 inch solid cubes

Challenge:

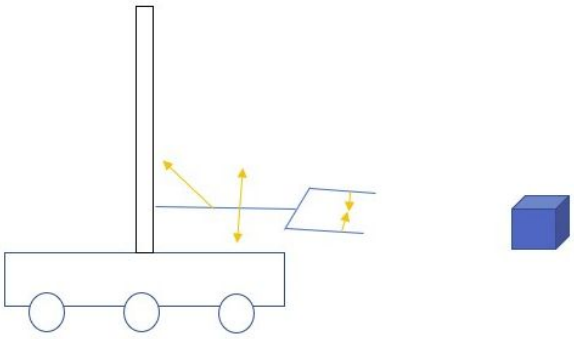
- Pick up cubes off a platform that is 1 foot high
- Drive to another 1 foot high platform and stack the cubes on top of each other
- You can have multiple stacks. Taller stacks get more points.
- You are only allowed to carry one cube at a time

Your Challenge:

As a team, work together at the whiteboard to define the software requirements

- What are the subsystems?
- What components does each subsystem have? (motors, sensors, encoders, etc)
- What actions does each subsystem perform?
- What states does each subsystem have?
- What limitations are there? (i.e. can one subsystem only accomplish a certain task if another subsystem is out of the way?)
- What actions do the drivers take?
- What buttons do the drivers use to perform those actions?
- Build sequences and parallels to complete the driver's actions

Yellow arrows indicate the direction of motion



Solution

Subsystems:
Drivetrain – motor group and encoder on each side
Elevator – 1 motor, 1 encoder
Arm - 1 motor, 1 encoder
Claw - 1 double solenoid

Actions:
Drivetrain – skid steer drive
Elevator – move up/down, go to set heights
Arm – pivot arm up/down, pivot over the top to face the other direction
Claw - open / close

Behaviors:
Drivetrain – zero, percent, velocity
Elevator – zero, states
Arm– zero, states
Claw - states

STATES:
Drivetrain:
st_drive_zero
st_drive_percent
st_drive_veleocity

Elevator:
st_elevator_zero
st_elevator_height1 (.5 feet)
st_elevator_height2 (1.5 feet) (default height)
st_elevator_height3 (2.5 feet)
st_elevator_height4 (3.5 feet)
st_elevator_height5 (4.5 feet)
st_elevator_height6 (5.5 feet)

Arm:
st_arm_zero
st_arm_out_front (0 degrees)
st_arm_protect_front (45 degrees)
st_arm_protect_back (135 degrees)
st_arm_out_back (180 degrees)

Claw
st_open
st_closed

DRIVER ACTIONS:
Drive - left and right joysticks
Velocity mode - right bumper
Open/Close Claw - right trigger

OPERATOR ACTIONS:
Arm out front - right trigger
Arm protect front - right bumper
Arm protect back - left bumper
Arm out back - left trigger
Elevator increase height - dpad up
Elevator decrease height - dpad down
Elevator minor adjustment up/down - left joystick
Arm minor adjustment up/down - right joystick

LOGIC:

Arm_Zero - Run arm motor backwards at -0.1 until 5000ms timer expires. Declare it has be zeroed

Elevator_Zero - Run arm motor backwards at -0.1 until 5000ms timer expires. Declare it has be zeroed

When the arm is in either the front or back protect position and the operator increments or decrements the height, the elevator does not move. Instead the value is stored and the elevator moves when the operator commands the arm to the front or back out position.

When moving from front to back or back to front side, The arm will move into the protect position. Then travel up to the top elevator position. Then switch to the opposite protect position. Then travel to the correct height.

When moving from protect to out at a new height, the arm will travel up to the new height first and then move from protect to out. This will reduce the chance of knocking over a stack of blocks.

TeleopModeLogic Variables:

```
private String mCurrentArmFacing = "front" or “back”;
private String mCurrentArmPosition = "protect" or “out”;
private int mCurrentElevatorHeight = 1 to 6;

private String mRequestedArmFacing = "front" or “back”;
private String mRequestedArmPosition = "protect" or “out”;
private int mRequestedElevatorHeight = 1 to 6;

private Boolean mClawOpen = true or false;
```

PARALLELS:

SEQUENCES:

```
#  Base sequences used in other sequences below
sq_front_to_back:
 sequence: [st_arm_protect_front, st_elevator_height6, st_arm_protect_back]
sq_back_to_front:
 sequence: [st_arm_protect_back, st_elevator_height6, st_arm_protect_front]

# Protect actions - always go to height 2
sq_front_to_h2_front_protect:
 sequence: [st_elevator_height2, st_arm_protect_front]
sq_front_to_h2_back_protect:
 sequence: [sq_front_to_back, st_elevator_height2]
sq_back_to_h2_back_protect:
 sequence: [st_elevator_height2, st_arm_protect_back]
sq_back_to_h2_front_protect:
 sequence: [sq_back_to_front, st_elevator_height2]

#  Front to front out actions
sq_front_to_h1_front_out:
 sequence: [st_elevator_height1, st_arm_out_front]
sq_front_to_h2_front_out:
 sequence: [st_elevator_height2, st_arm_out_front]
sq_front_to_h3_front_out:
 sequence: [st_elevator_height3, st_arm_out_front]
sq_front_to_h4_front_out:
 sequence: [st_elevator_height4, st_arm_out_front]
sq_front_to_h5_front_out:
 sequence: [st_elevator_height5, st_arm_out_front]
sq_front_to_h6_front_out:
 sequence: [st_elevator_height6, st_arm_out_front]

# Front to back out actions
sq_front_to_h1_back_out:
 sequence: [sq_front_to_back, st_elevator_height1, st_arm_out_back]
sq_front_to_h2_back_out:
 sequence: [sq_front_to_back, st_elevator_height2, st_arm_out_back]
sq_front_to_h3_back_out:
 sequence: [sq_front_to_back, st_elevator_height3, st_arm_out_back]
sq_front_to_h4_back_out:
 sequence: [sq_front_to_back, st_elevator_height4, st_arm_out_back]
sq_front_to_h5_back_out:
 sequence: [sq_front_to_back, st_elevator_height5, st_arm_out_back]
sq_front_to_h6_back_out:
 sequence: [sq_front_to_back, st_elevator_height6, st_arm_out_back]

# Back to back actions
sq_back_to_h1_back_out:
 sequence: [st_elevator_height1, st_arm_out_back]
sq_back_to_h2_back_out:
 sequence: [st_elevator_height2, st_arm_out_back]
sq_back_to_h3_back_out:
 sequence: [st_elevator_height3, st_arm_out_back]
```

```
sq_back_to_h4_back_out:
 sequence: [st_elevator_height4, st_arm_out_back]
sq_back_to_h5_back_out:
 sequence: [st_elevator_height5, st_arm_out_back]
sq_back_to_h6_back_out:
 sequence: [st_elevator_height6, st_arm_out_back]

# Back to front actions
sq_back_to_h1_front_out:
 sequence: [sq_back_to_front, st_elevator_height1, st_arm_out_front]
sq_back_to_h2_front_out:
 sequence: [sq_back_to_front, st_elevator_height2, st_arm_out_front]
sq_back_to_h3_front_out:
 sequence: [sq_back_to_front, st_elevator_height3, st_arm_out_front]
sq_back_to_h4_front_out:
 sequence: [sq_back_to_front, st_elevator_height4, st_arm_out_front]
sq_back_to_h5_front_out:
 sequence: [sq_back_to_front, st_elevator_height5, st_arm_out_front]
sq_back_to_h6_front_out:
 sequence: [sq_back_to_front, st_elevator_height6, st_arm_out_front]
```