2024 VEXcode VR Skills Challenge - Elementary School by Team 97793A

Team Number: 97793A

Team Name: Hicks Team A - Guardians of the Bot

Team Members: Carter, Colin, Markus, Rohanna, Yura

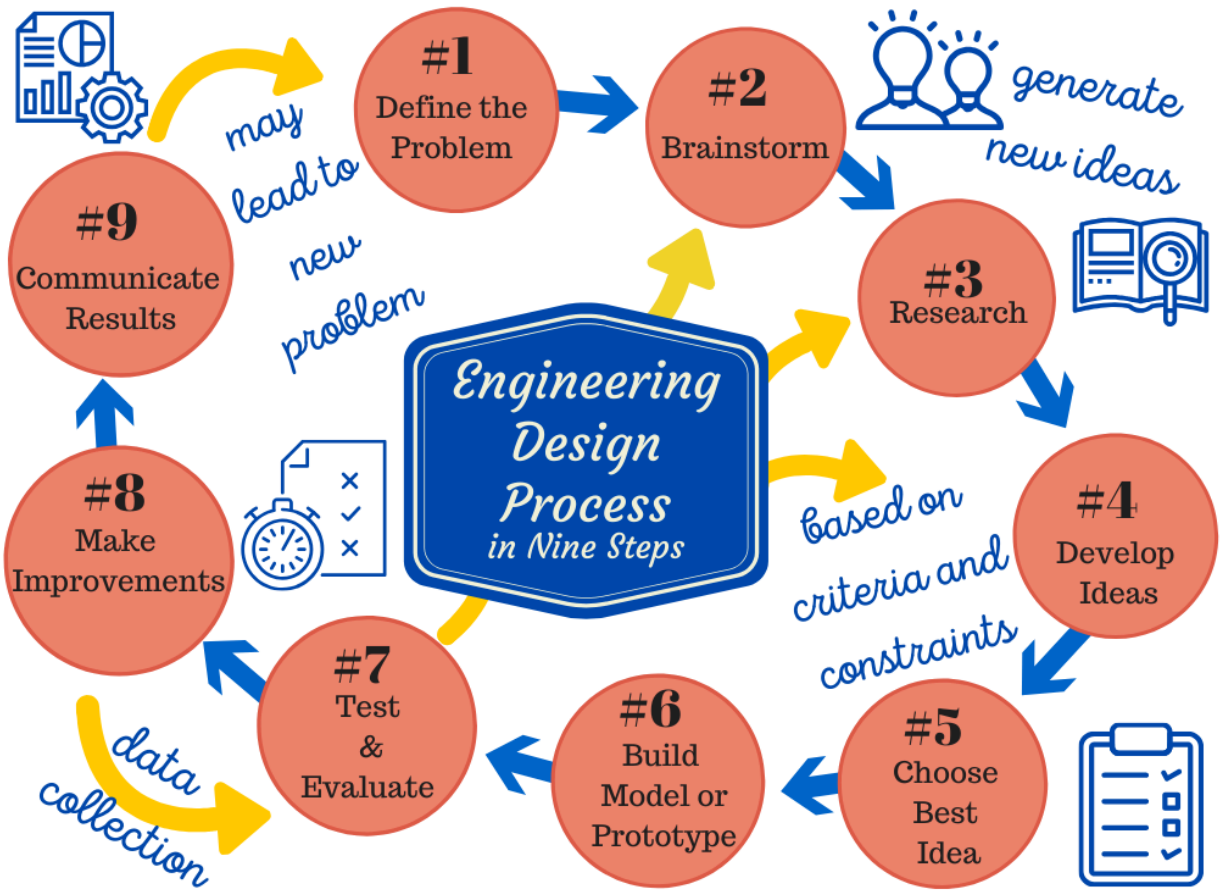
Programmers: Colin, Markus

School: Hicks Canyon Elementary

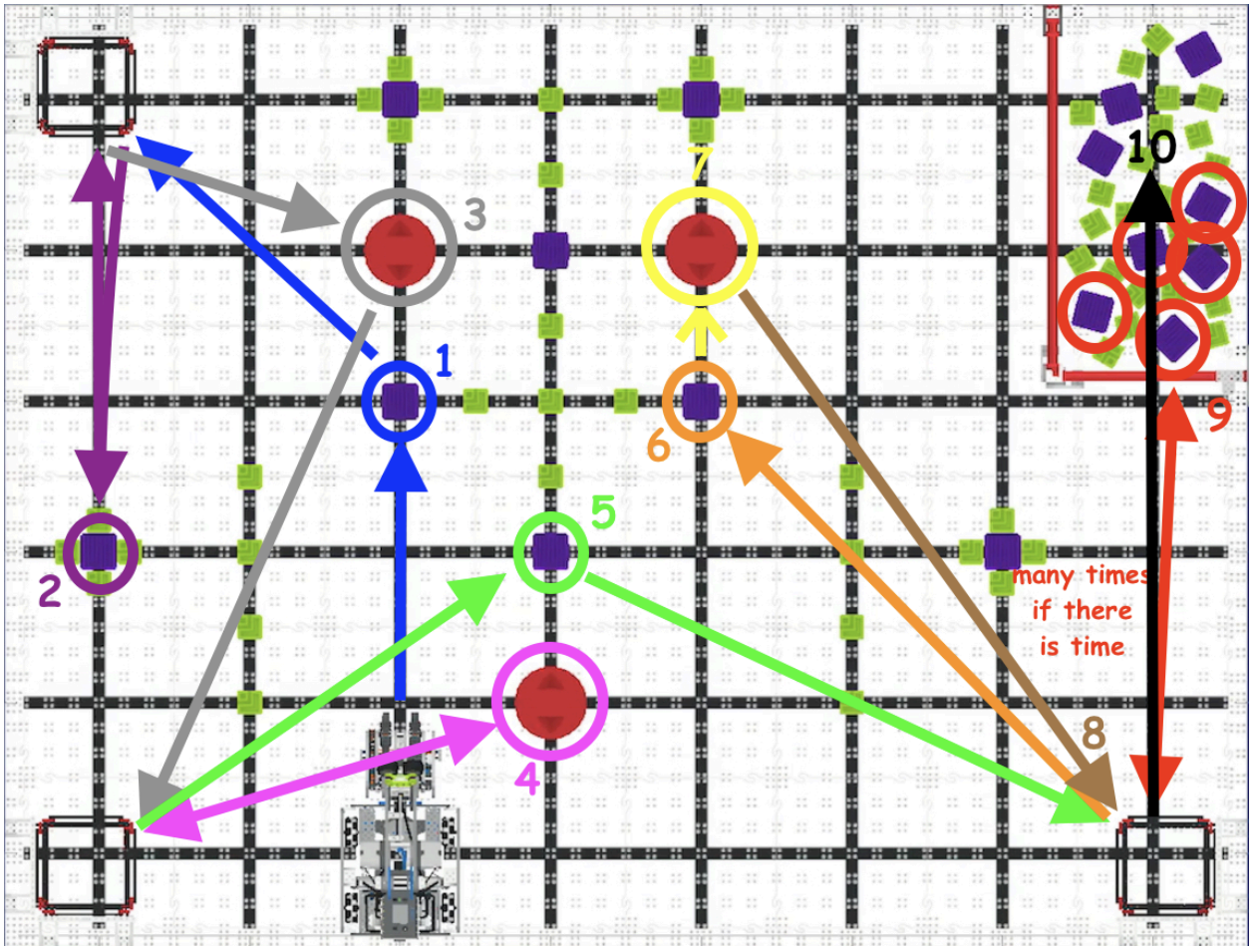
Location: Irvine, California

Our Process:

We followed **ALL the steps #1 to 9** of the **Engineering Design Process** to do this online challenge:

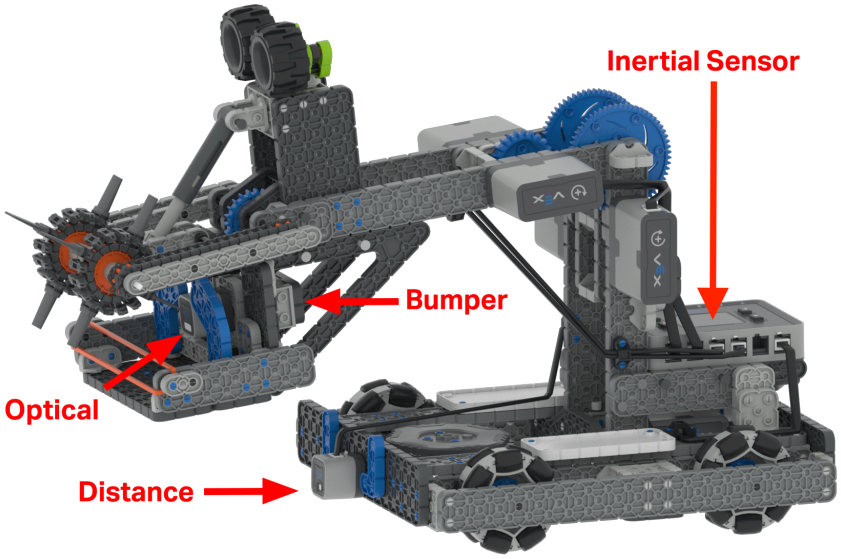


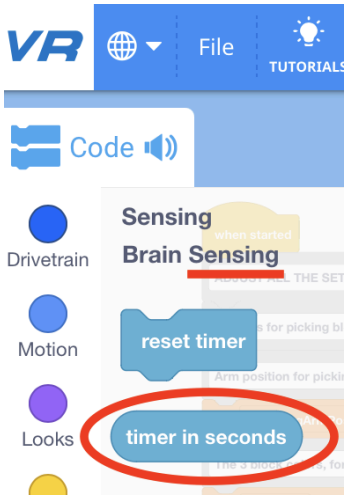
Our Strategy:



Criteria	**Points**
Score 6 blocks or more	6 or more
Move 3 red blocks	15
3 Uniform Bonus	30
Fill Level 2	20
Full Parking	10
TOTAL	**81 or more**

Our Sensors Use:





1. **Distance sensor:**
    - To tell us how far the object in front of the distance sensor is, to help us **calculate** how far to drive to reach our destination **accurately instead of hardcoding** the exact distance to drive
    - To put the bot in the correct position from the **goal** to score consistently

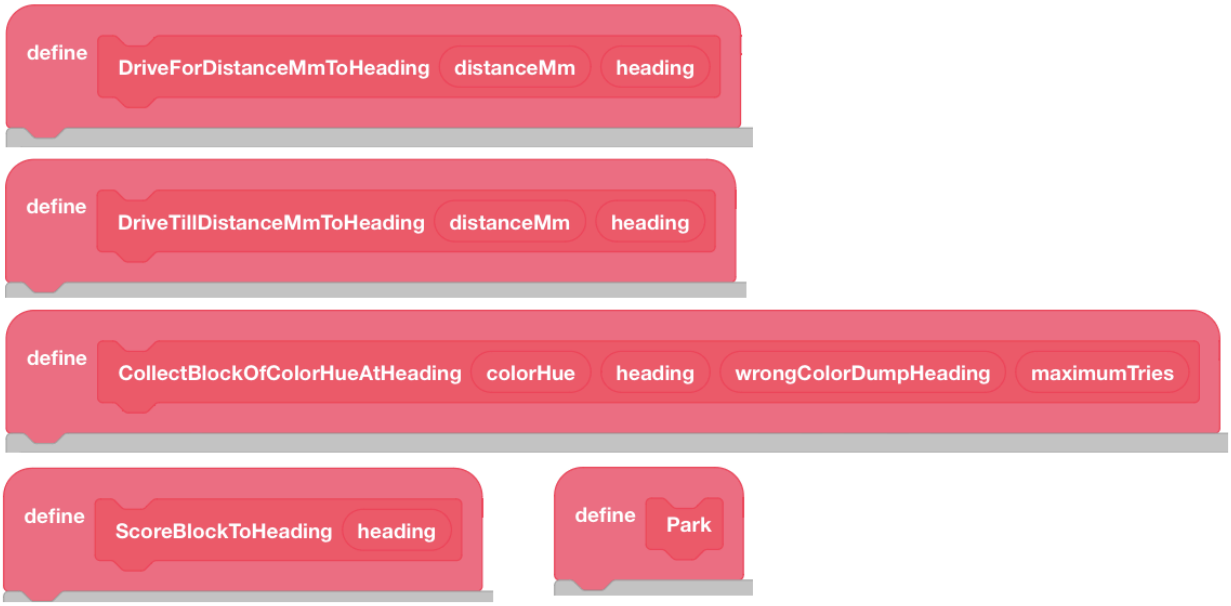
2. **Bumper switch:**
    - to see whether a block has been picked up successfully

3. **Optical sensor:**
    - to see what color the block in the intake is
    - To see whether the block that's picked up is the right color

4. **Inertial sensor:**
    - to make accurate and precise turns using the drivetrain heading

5. **Timer on brain:**
    - to see how much match time we have used up, to see how much time we have left to do things

Our Functions Use:



- We used functions for **common operations that are used more than once** in the program, for example picking up a block, **instead of making copies of the same code many times all over** the program

- This way, we will **not need to change many copies of the code** every time we need to make a change. Doing that **takes time** and is **easy to make mistakes**

- Even if an operation is not needed more than once, functions also helped us **organize** our code into groups, for example parking.

- The makes our code **more organized and easier to read** than hundreds of lines of code altogether in one place

Our functions:

1. **DriveForDistanceMmToHeading** - Turn and drive forward for the input distance

2. **DriveTillDistanceMmtoHeading -** Turn and drive until the robot reaches the input distance from the object in front

3. **CollectBlockOfColorHueAtHeading -** Collect a block

4. **ScoreBlockToHeading -** Turn and drive to the goal and score a block into the goal

5. **Park -** Fully park the bot.

- Functions 1-4 were **used many times** in the code

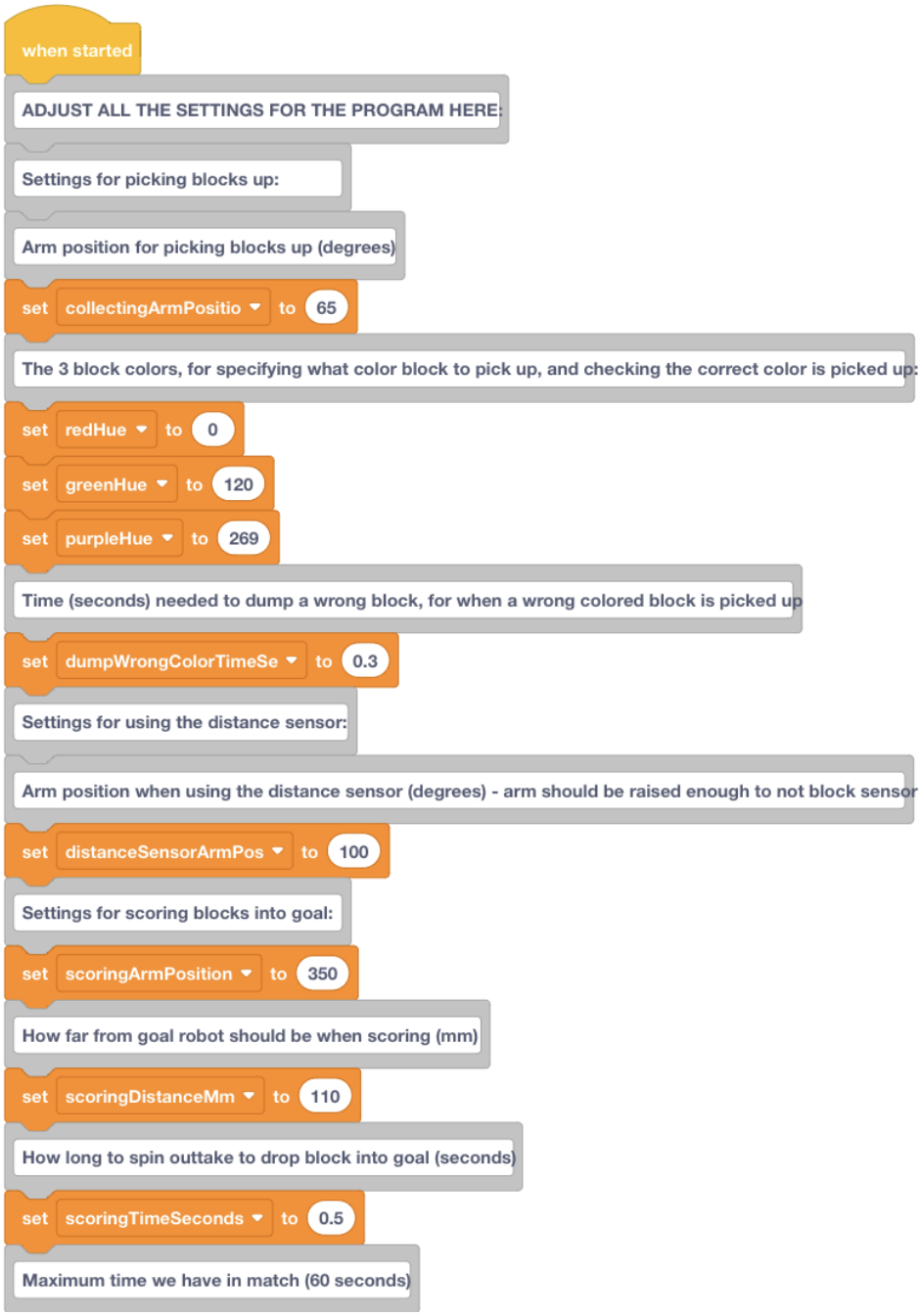
- We used inputs to **input different information for the functions to use** from different places in the code, for example whether to collect a purple or red block

- Function 5 was only used once, but it has almost 20 lines of code with conditional logic, so it was good to organize it into its own function that is easier to read.

Our Variables Use:

- We used variables **instead of coding hard numbers**

- We **put all variables together at the top of the program** instead of scattered everywhere in the code. It is **very easy and convenient to see and adjust all their values** from the top of the program.



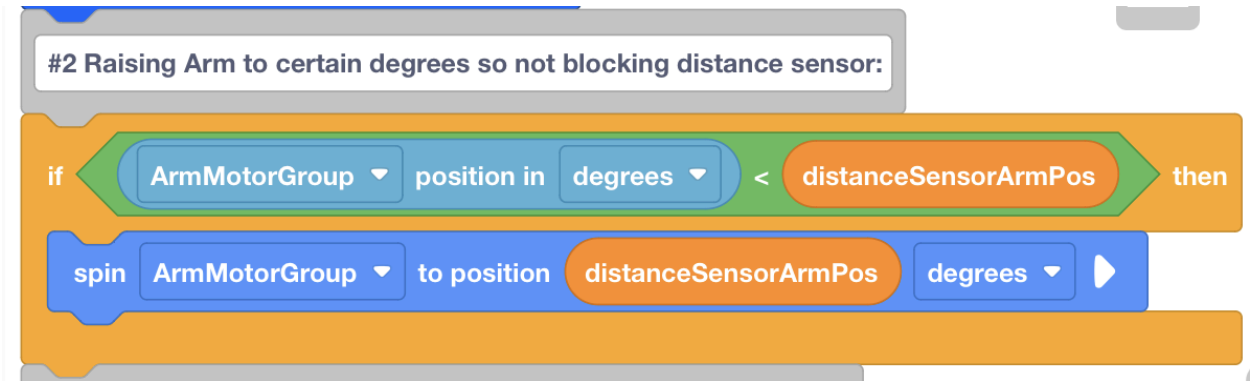

set maximumMatchTimeSeco to 60
How much time (seconds) is needed to grab and score another block before parking, to decide whether we should go for another
set timeNeededForOneMore to 15
Settings for full parking:
set parkingHeading to 0
How long (seconds) to speed towards the supply zone at the beginning of full parking:
set initialFullParkTimeS to 1.5
How much to jiggle the robot left and right to push it into the supply zone (degrees)
set fullParkJiggleAngle to 30
How much time (seconds) to pause after jiggling the robot
set fullParkJiggleTimeSe to 0.01
How long (seconds) to drive to climb into the supply zone after jiggling the robot
set continueFullParkTime to 0.5
Arm position after full parking (degrees) - point up to not stick out
set fullParkingArmPositi to 575
Minimum drivetrain heading (0 degrees)
set minimumHeading to 0
Drivetrain heading limit (360 degrees)
set headingLimit to 359.9
Set all velocities to the fastest possible:
set drive velocity to 100 %

Our Advanced Programming Structures Use:

Advanced programming structures and conditional logic we used:

1. **If blocks**

   Example:



   The arm position for using the distance sensor is set in the distanceSensorArmPos variable. The arm will not be blocking the distance sensor from there.

   When trying to use the distance sensor:

   If arm is currently lower than the distanceSensorArmPos variable setting **then**

   Raise arm to the distanceSensorArmPos variable position

   That means:

   If arm is too low and is blocking the distance sensor **then**

   Raise arm to non blocking position for the distance sensor

2. If Else blocks

Example - If Else block inside If block:

When trying to drive the robot to the input distance from the object in front:

If the distance sensor is not at the input distance from the object in front **then**

If the distance sensor is too far from the object **then**

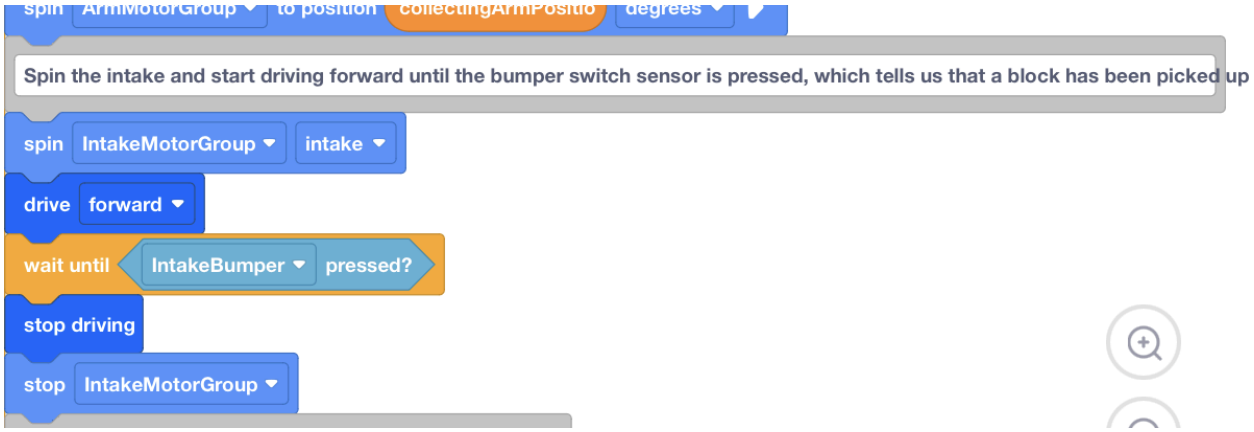
Drive forward for how much it is too far by

Else

Otherwise, drive backward for how far it is too close by

3. Wait Until blocks

Example:



When trying to pick up a block:

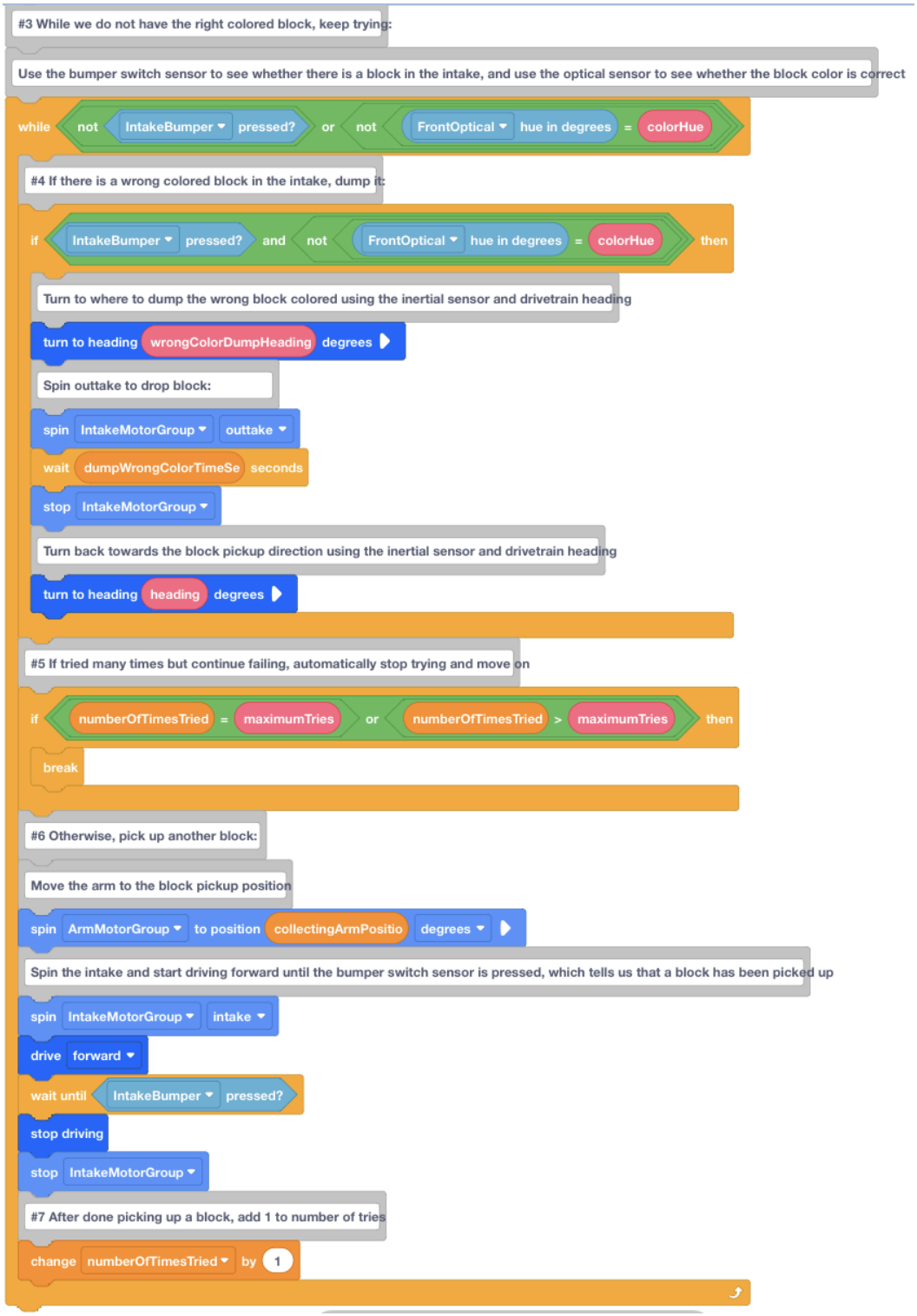
Start spinning the intake and driving forward

Wait until the bumper switch sensor is pressed, which tells us the sensor detects a block in the intake

Then stop driving forward and spinning the intake

4. While blocks

Example - While block with If blocks and Wait Until blocks inside it:



When trying to pick up a block, which has to be the right color:

While the intake is empty, or the block in the intake is not of the right color

If there is a block in the intake, but it is not of the right color **then**

Dump the wrong colored block and turn back to the block pick up direction

If we have reached, or exceeded the maximum number of tries to pick up the right colored block **then**

quit

Move the arm to the block pickup position, and start spinning the intake and driving forward to pick up a block

Wait until the bumper switch sensor is pressed, which tells us the sensor detects a block in the intake

Then stop driving forward and spinning the intake

Increase the number of tries counter

Lessons Learned:

How has VR Skills improved our coding skills and helped us with our competition?

1. We have gotten a lot more familiar with coding a robot using Vexcode Blocks

2. We have learned a lot about sensors, functions, and variables.

3. We have changed our **competition** autonomous code to use a lot **more variables** instead of coding hard numbers. This made **changing** the code **and testing** a lot **easier**.

4. We have changed our **competition** autonomous code to use **functions for common operations instead of repeating the same code** many times in our program, for example to score blocks into a goal. This has made **changing** our code a lot **easier and quicker**.

   We have also changed our **competition** autonomous code by **organizing** different sections of our code **into functions**. This has made our code **more organized and easier to understand**.

5. We have changed our **competition** autonomous code to use the **Inertial Sensor and drivetrain heading** to make accurate and precise turns **instead of hardcoding** the exact angles to turn, for example:

   From:

   Turn right for 65 degrees

   To:

   Turn to heading 165 degrees

This has helped us face our robot in the right direction no matter whether the robot was facing the right direction before. This has helped our **robot run a lot more smoothly and accurately in** our **competition autonomous skill runs**.

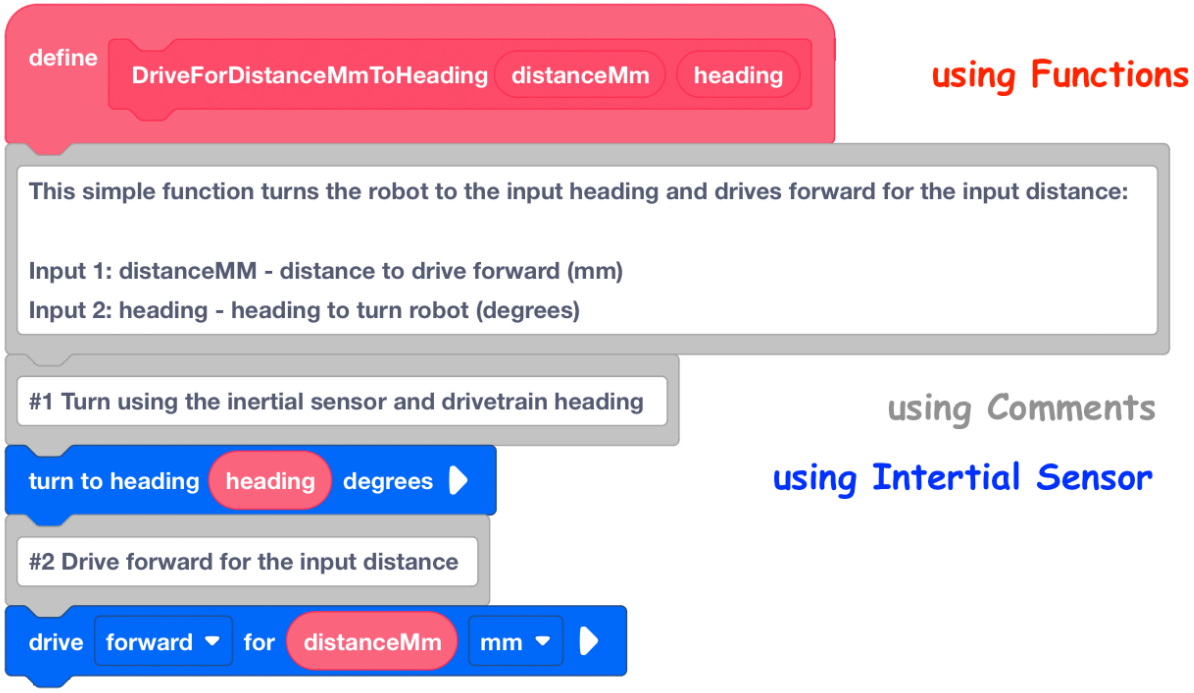
6. We are going to modify our **competition** robot to add a **distance sensor** to the front of our chassis:

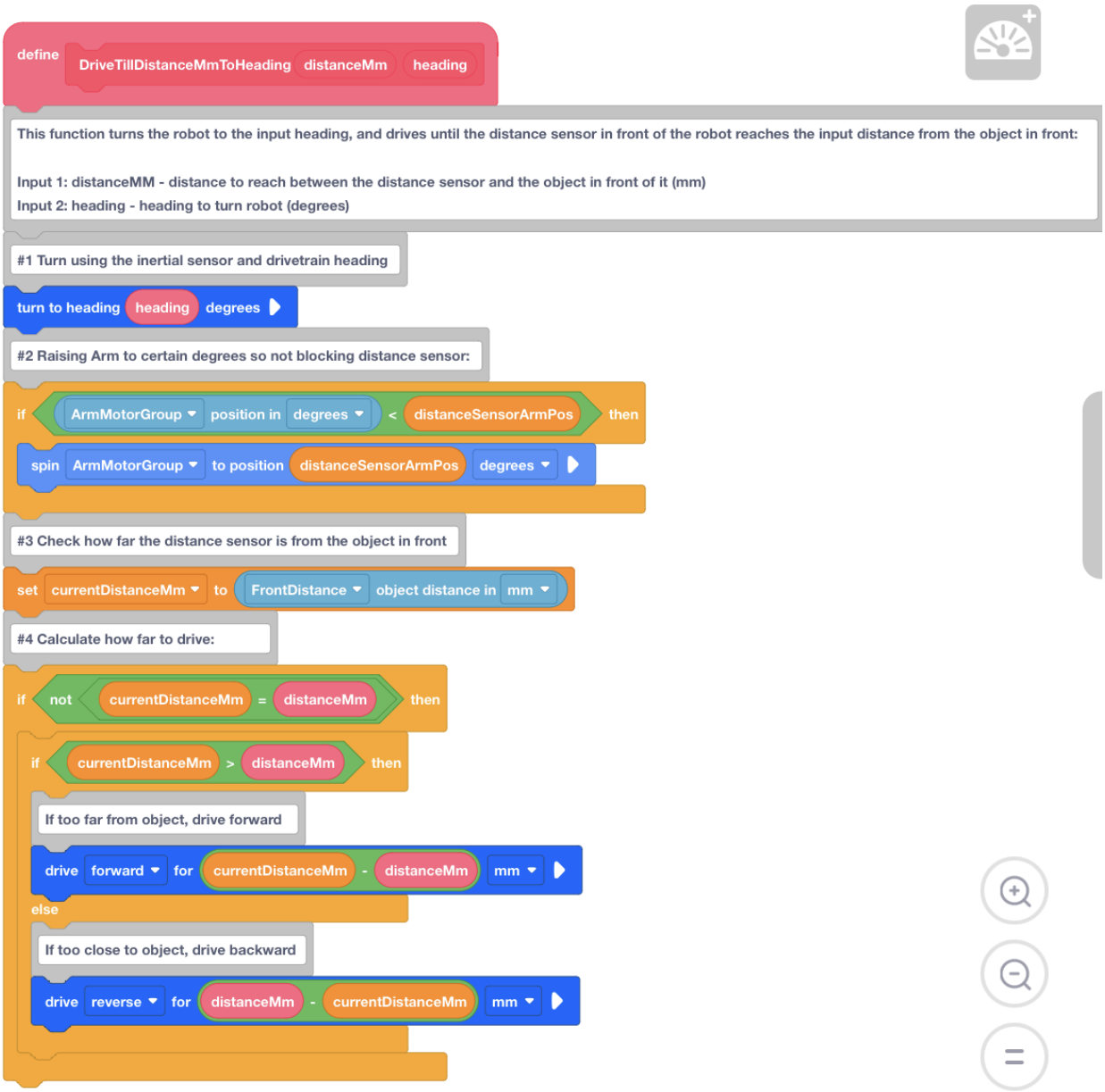
    - To help us **calculate and make accurate drives** to reach our destination **instead of hardcoding** the exact distances to drive which does not always work, for example when the robot has drifted which changes the next driving start position

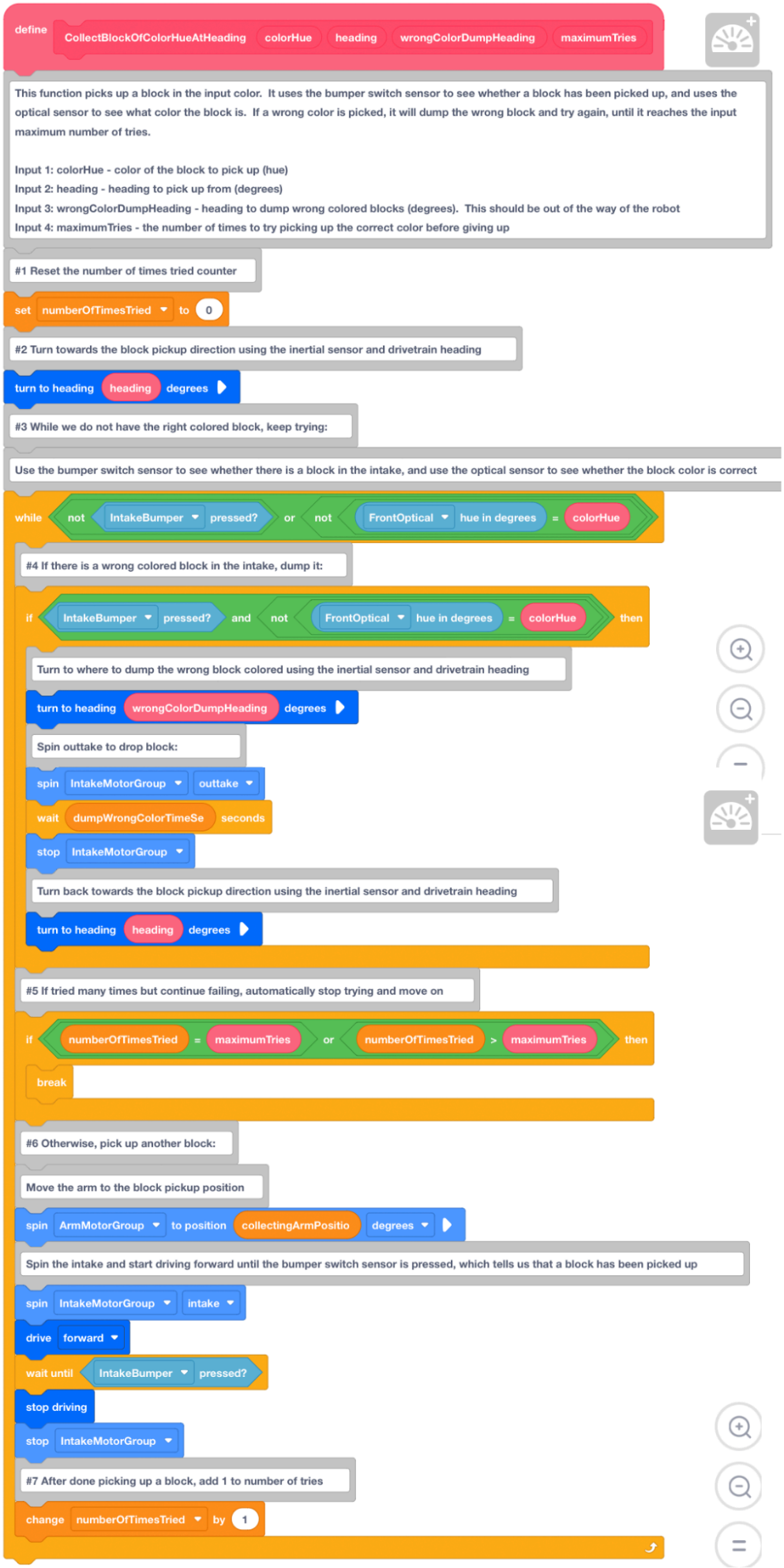
    - To help us always drive the robot to the correct distance from the **goal** to score consistently instead of hardcoding the exact distance to travel which does not always work

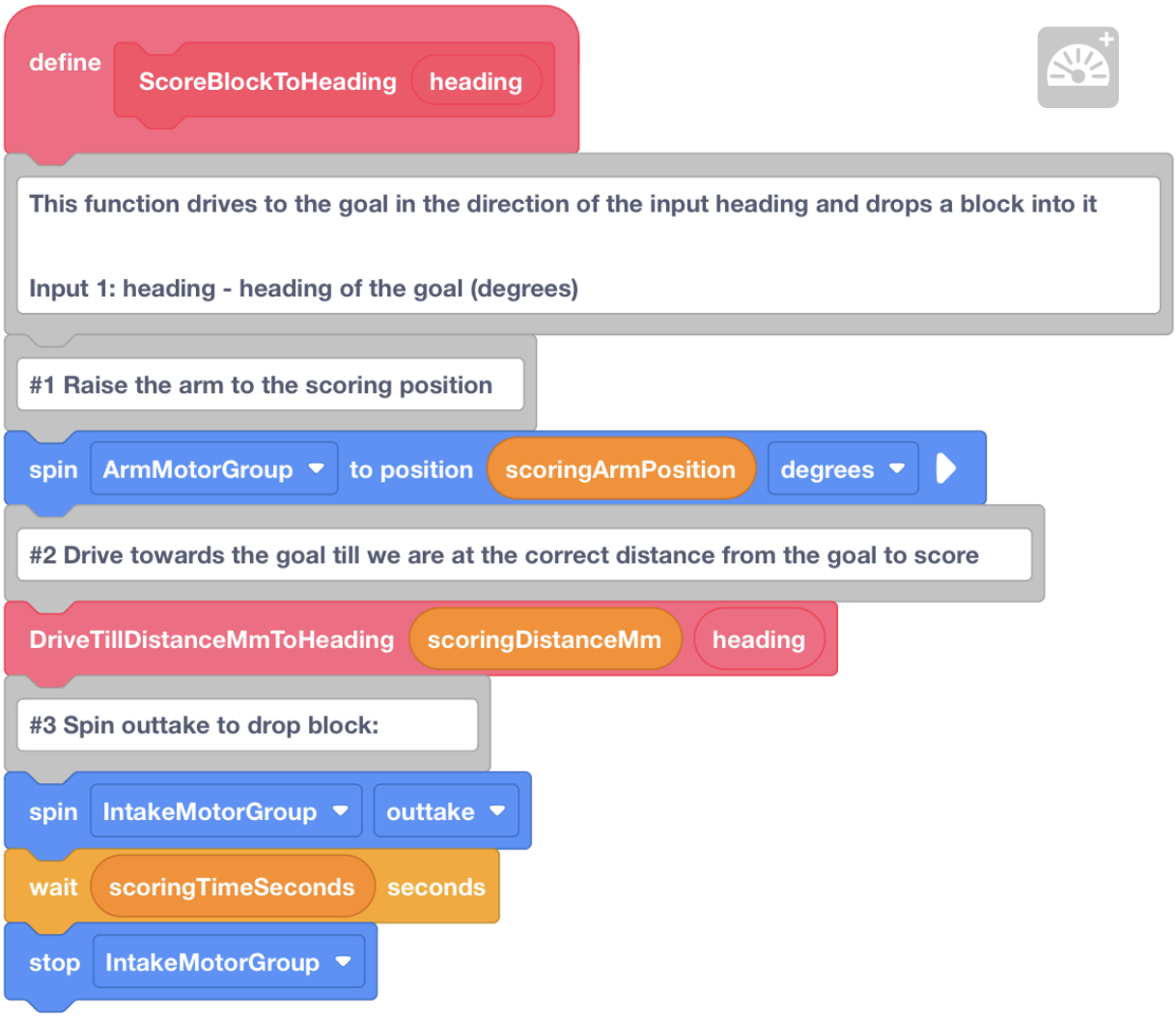
Our Code:

This simple function turns the robot to the input heading and drives forward for the input distance

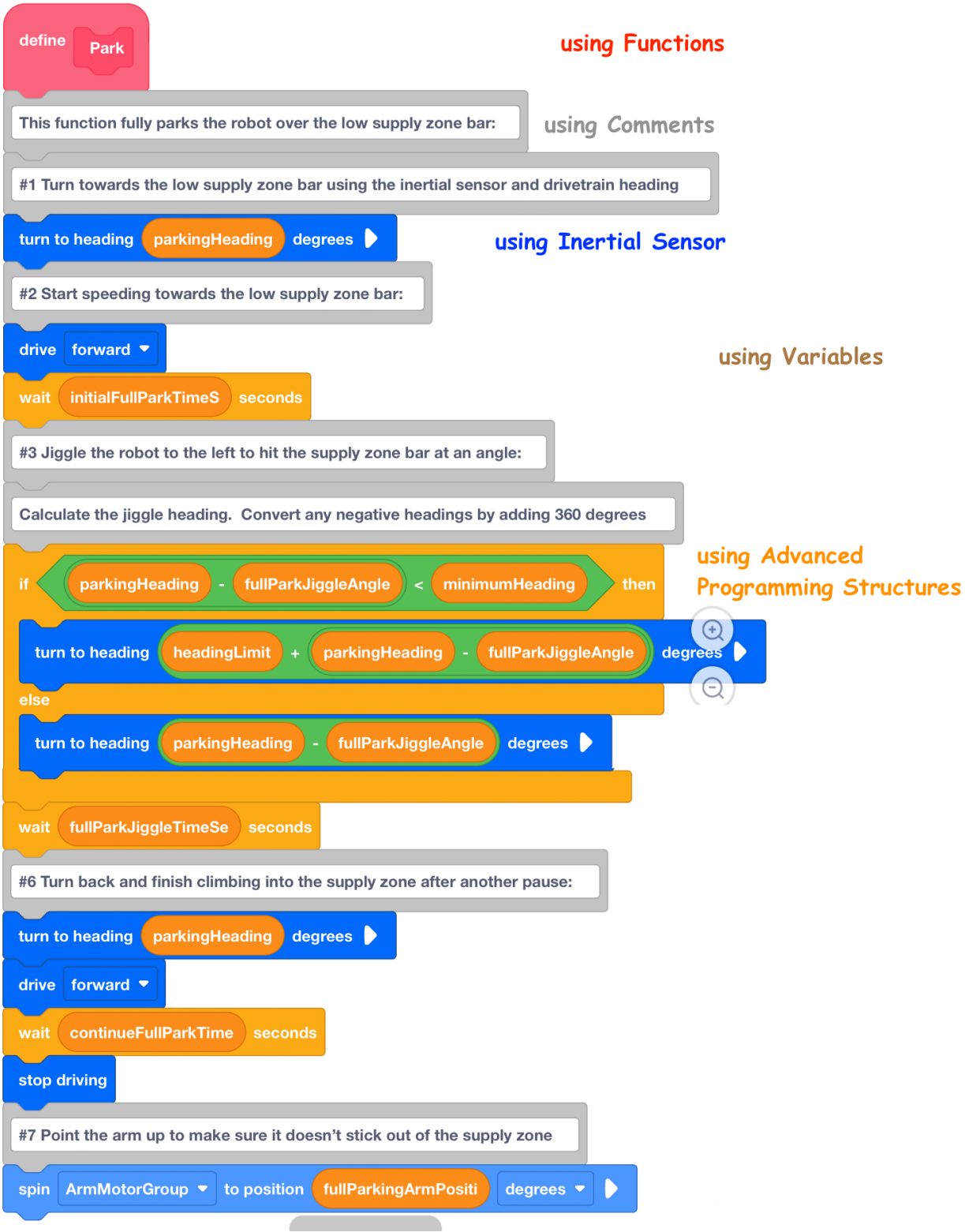


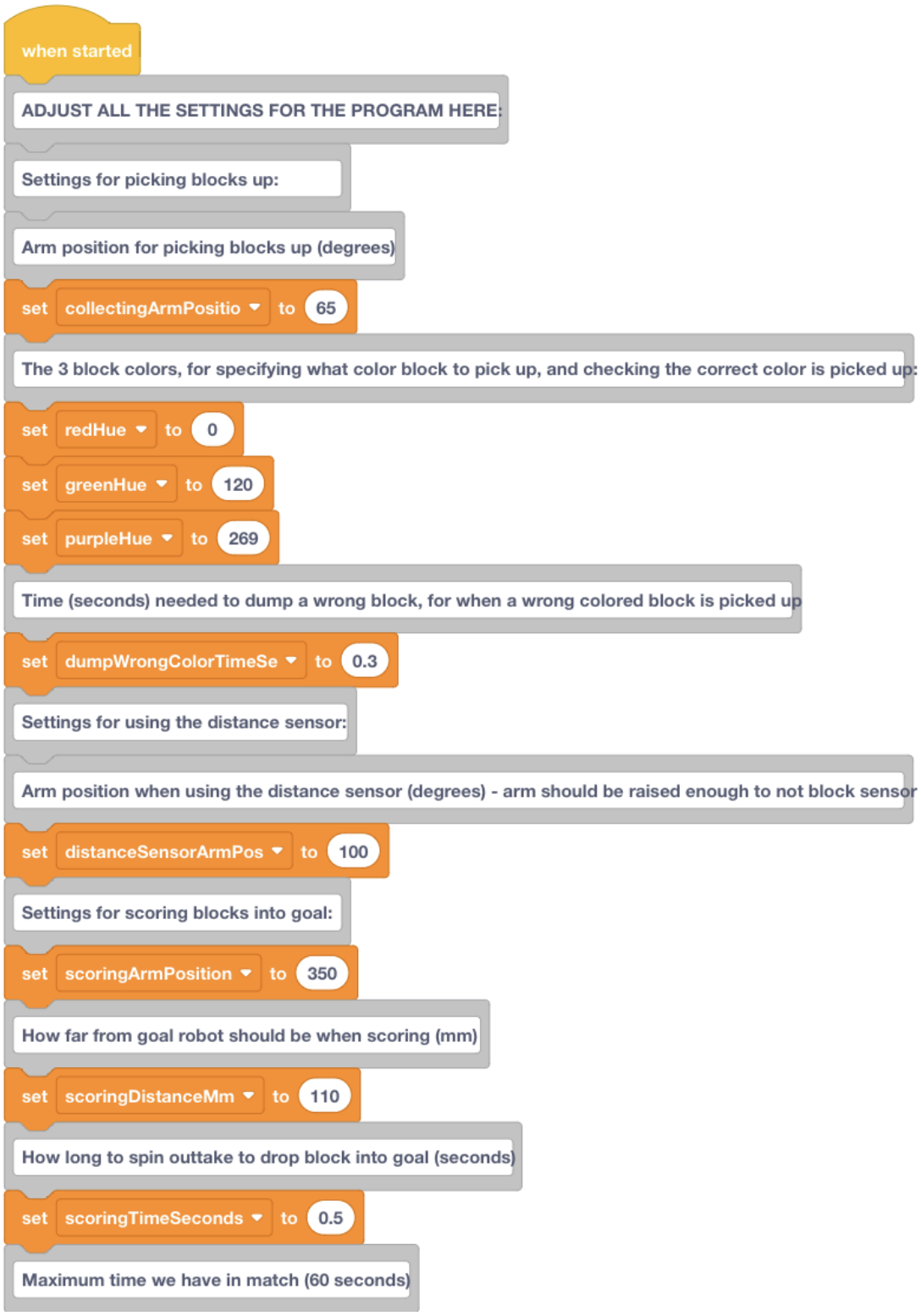

define DriveTillDistanceMmToHeading distanceMm heading
This function turns the robot to the input heading, and drives until the distance sensor in front of the robot reaches the input distance from the object in front:
Input 1: distanceMM - distance to reach between the distance sensor and the object in front of it (mm)
Input 2: heading - heading to turn robot (degrees)
#1 Turn using the inertial sensor and drivetrain heading
turn to heading heading degrees
#2 Raising Arm to certain degrees so not blocking distance sensor:
if ArmMotorGroup position in degrees < distanceSensorArmPos then
spin ArmMotorGroup to position distanceSensorArmPos degrees
#3 Check how far the distance sensor is from the object in front
set currentDistanceMm to FrontDistance object distance in mm
#4 Calculate how far to drive:
if not currentDistanceMm = distanceMm then
if currentDistanceMm > distanceMm then
If too far from object, drive forward
drive forward for currentDistanceMm - distanceMm mm
else
If too close to object, drive backward
drive reverse for distanceMm - currentDistanceMm mm


define
CollectBlockOfColorHueAtHeading
colorHue
heading
wrongColorDumpHeading
maximumTries
This function picks up a block in the input color. It uses the bumper switch sensor to see whether a block has been picked up, and uses the optical sensor to see what color the block is. If a wrong color is picked, it will dump the wrong block and try again, until it reaches the input maximum number of tries.
Input 1: colorHue - color of the block to pick up (hue)
Input 2: heading - heading to pick up from (degrees)
Input 3: wrongColorDumpHeading - heading to dump wrong colored blocks (degrees). This should be out of the way of the robot
Input 4: maximumTries - the number of times to try picking up the correct color before giving up
#1 Reset the number of times tried counter
set numberOfTimesTried to 0
#2 Turn towards the block pickup direction using the inertial sensor and drivetrain heading
turn to heading heading degrees
#3 While we do not have the right colored block, keep trying:
Use the bumper switch sensor to see whether there is a block in the intake, and use the optical sensor to see whether the block color is correct
while not IntakeBumper pressed? or not FrontOptical hue in degrees = colorHue
#4 If there is a wrong colored block in the intake, dump it:
if IntakeBumper pressed? and not FrontOptical hue in degrees = colorHue then
Turn to where to dump the wrong block colored using the inertial sensor and drivetrain heading
turn to heading wrongColorDumpHeading degrees
Spin outtake to drop block:
spin IntakeMotorGroup outtake
wait dumpWrongColorTimeSe seconds
stop IntakeMotorGroup
Turn back towards the block pickup direction using the inertial sensor and drivetrain heading
turn to heading heading degrees
#5 If tried many times but continue failing, automatically stop trying and move on
if numberOfTimesTried = maximumTries or numberOfTimesTried > maximumTries then
break
#6 Otherwise, pick up another block:
Move the arm to the block pickup position
spin ArmMotorGroup to position collectingArmPositio degrees
Spin the intake and start driving forward until the bumper switch sensor is pressed, which tells us that a block has been picked up
spin IntakeMotorGroup intake
drive forward
wait until IntakeBumper pressed?
stop driving
stop IntakeMotorGroup
#7 After done picking up a block, add 1 to number of tries
change numberOfTimesTried by 1

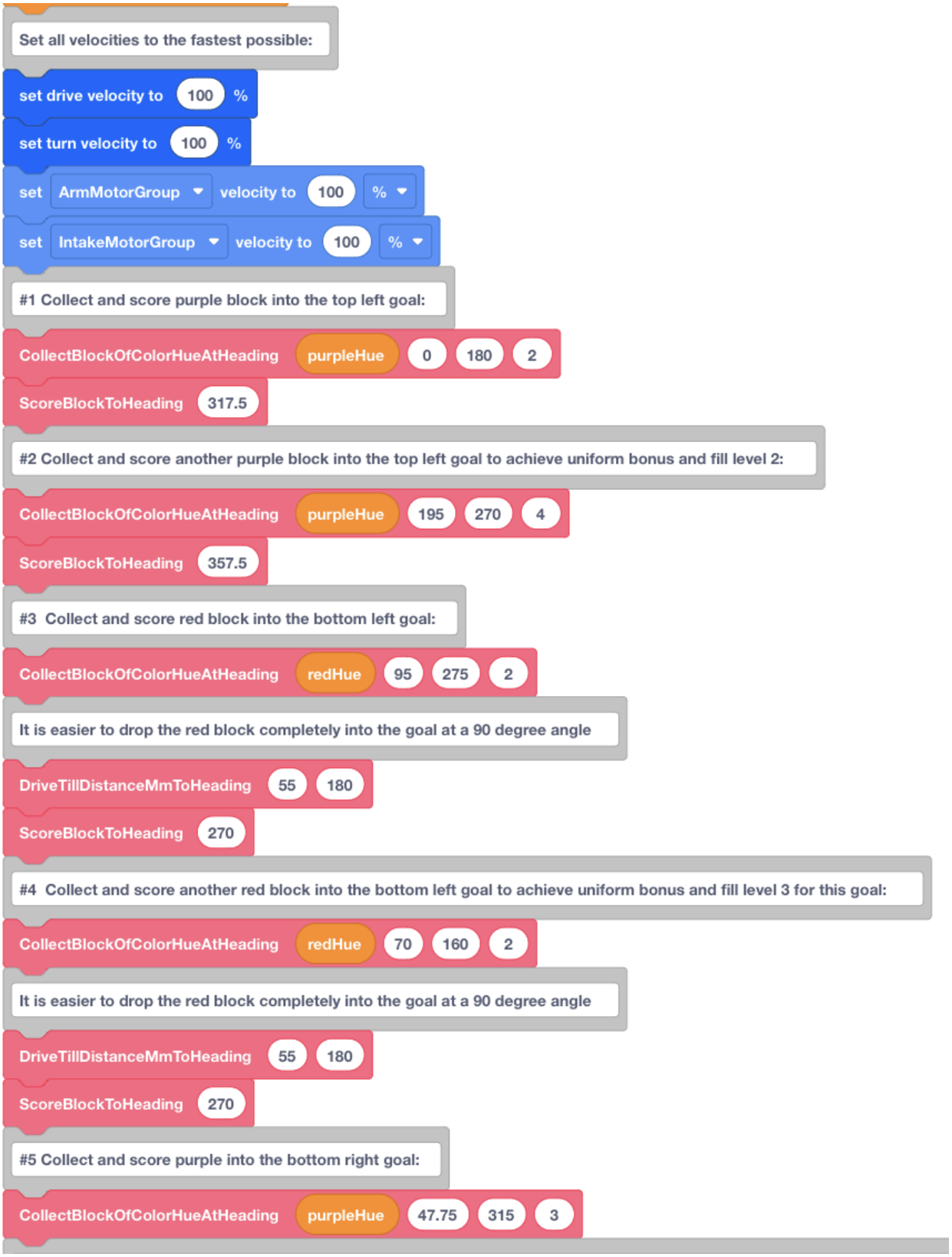

define
ScoreBlockToHeading
heading
This function drives to the goal in the direction of the input heading and drops a block into it
Input 1: heading - heading of the goal (degrees)
#1 Raise the arm to the scoring position
spin
ArmMotorGroup
to position
scoringArmPosition
degrees
#2 Drive towards the goal till we are at the correct distance from the goal to score
DriveTillDistanceMmToHeading
scoringDistanceMm
heading
#3 Spin outtake to drop block:
spin
IntakeMotorGroup
outtake
wait
scoringTimeSeconds
seconds
stop
IntakeMotorGroup

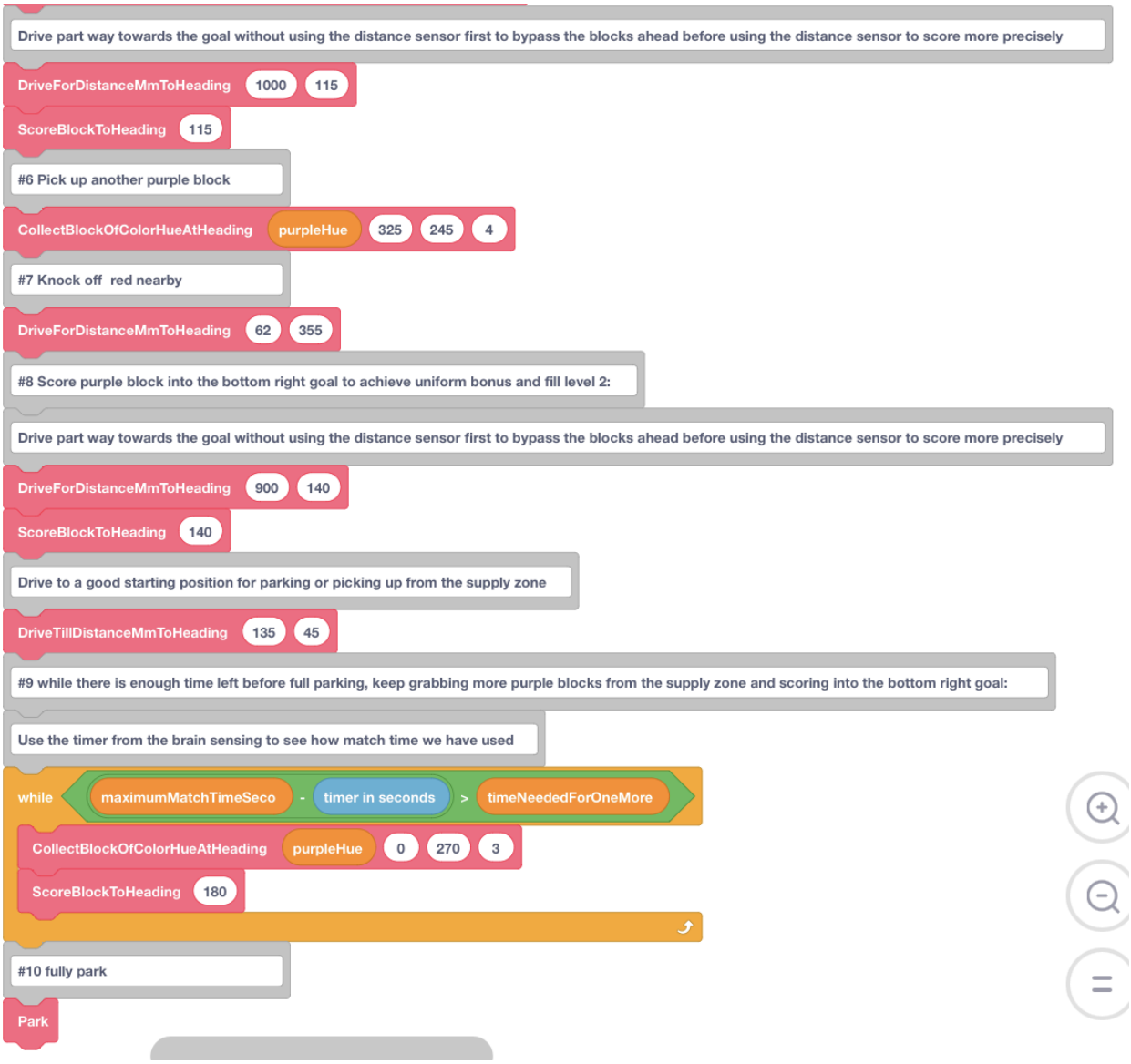
This function fully parks the robot over the low supply zone bar




when started
ADJUST ALL THE SETTINGS FOR THE PROGRAM HERE:
Settings for picking blocks up:
Arm position for picking blocks up (degrees)
set collectingArmPositio to 65
The 3 block colors, for specifying what color block to pick up, and checking the correct color is picked up:
set redHue to 0
set greenHue to 120
set purpleHue to 269
Time (seconds) needed to dump a wrong block, for when a wrong colored block is picked up
set dumpWrongColorTimeSe to 0.3
Settings for using the distance sensor:
Arm position when using the distance sensor (degrees) - arm should be raised enough to not block sensor
set distanceSensorArmPos to 100
Settings for scoring blocks into goal:
set scoringArmPosition to 350
How far from goal robot should be when scoring (mm)
set scoringDistanceMm to 110
How long to spin outtake to drop block into goal (seconds)
set scoringTimeSeconds to 0.5
Maximum time we have in match (60 seconds)


set maximumMatchTimeSeco to 60
How much time (seconds) is needed to grab and score another block before parking, to decide whether we should go for another
set timeNeededForOneMore to 15
Settings for full parking:
set parkingHeading to 0
How long (seconds) to speed towards the supply zone at the beginning of full parking:
set initialFullParkTimeS to 1.5
How much to jiggle the robot left and right to push it into the supply zone (degrees)
set fullParkJiggleAngle to 30
How much time (seconds) to pause after jiggling the robot
set fullParkJiggleTimeSe to 0.01
How long (seconds) to drive to climb into the supply zone after jiggling the robot
set continueFullParkTime to 0.5
Arm position after full parking (degrees) - point up to not stick out
set fullParkingArmPositi to 575
Minimum drivetrain heading (0 degrees)
set minimumHeading to 0
Drivetrain heading limit (360 degrees)
set headingLimit to 359.9
Set all velocities to the fastest possible:


Set all velocities to the fastest possible:
set drive velocity to 100 %
set turn velocity to 100 %
set ArmMotorGroup velocity to 100 %
set IntakeMotorGroup velocity to 100 %
#1 Collect and score purple block into the top left goal:
CollectBlockOfColorHueAtHeading purpleHue 0 180 2
ScoreBlockToHeading 317.5
#2 Collect and score another purple block into the top left goal to achieve uniform bonus and fill level 2:
CollectBlockOfColorHueAtHeading purpleHue 195 270 4
ScoreBlockToHeading 357.5
#3 Collect and score red block into the bottom left goal:
CollectBlockOfColorHueAtHeading redHue 95 275 2
It is easier to drop the red block completely into the goal at a 90 degree angle
DriveTillDistanceMmToHeading 55 180
ScoreBlockToHeading 270
#4 Collect and score another red block into the bottom left goal to achieve uniform bonus and fill level 3 for this goal:
CollectBlockOfColorHueAtHeading redHue 70 160 2
It is easier to drop the red block completely into the goal at a 90 degree angle
DriveTillDistanceMmToHeading 55 180
ScoreBlockToHeading 270
#5 Collect and score purple into the bottom right goal:
CollectBlockOfColorHueAtHeading purpleHue 47.75 315 3


Drive part way towards the goal without using the distance sensor first to bypass the blocks ahead before using the distance sensor to score more precisely
DriveForDistanceMmToHeading 1000 115
ScoreBlockToHeading 115
#6 Pick up another purple block
CollectBlockOfColorHueAtHeading purpleHue 325 245 4
#7 Knock off red nearby
DriveForDistanceMmToHeading 62 355
#8 Score purple block into the bottom right goal to achieve uniform bonus and fill level 2:
Drive part way towards the goal without using the distance sensor first to bypass the blocks ahead before using the distance sensor to score more precisely
DriveForDistanceMmToHeading 900 140
ScoreBlockToHeading 140
Drive to a good starting position for parking or picking up from the supply zone
DriveTillDistanceMmToHeading 135 45
#9 while there is enough time left before full parking, keep grabbing more purple blocks from the supply zone and scoring into the bottom right goal:
Use the timer from the brain sensing to see how match time we have used
while maximumMatchTimeSeco - timer in seconds > timeNeededForOneMore
CollectBlockOfColorHueAtHeading purpleHue 0 270 3
ScoreBlockToHeading 180
#10 fully park
Park