Title : 8391W's Entry

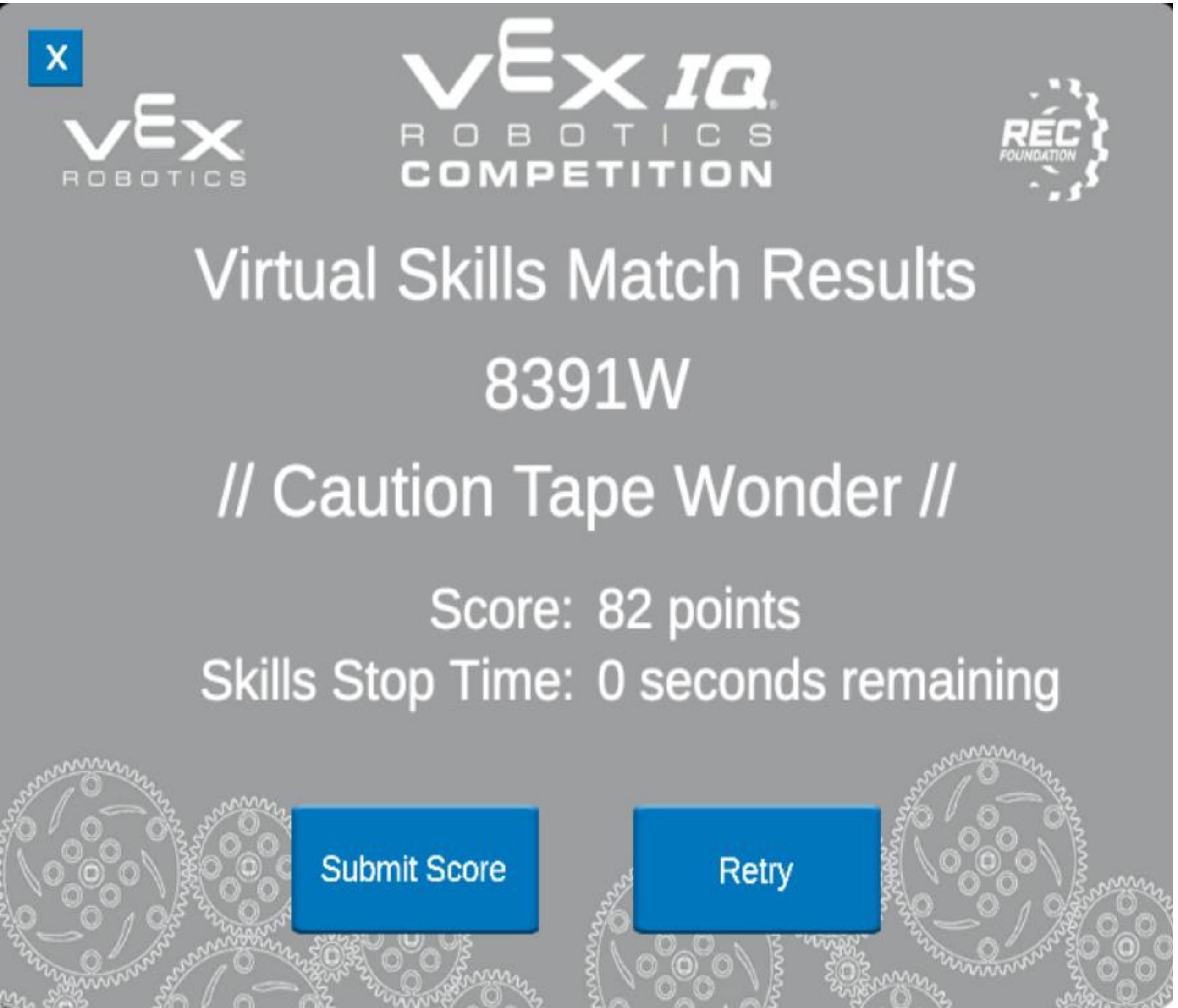
VEX® CODE VR

By Student : Ethan Wang

Team Number: 8391W

Location of team : Markham, Ontario

What Can My Code Do?/ What Does It Include?



Virtual Skills Match Results

8391W

// Caution Tape Wonder //

Score: 82 points

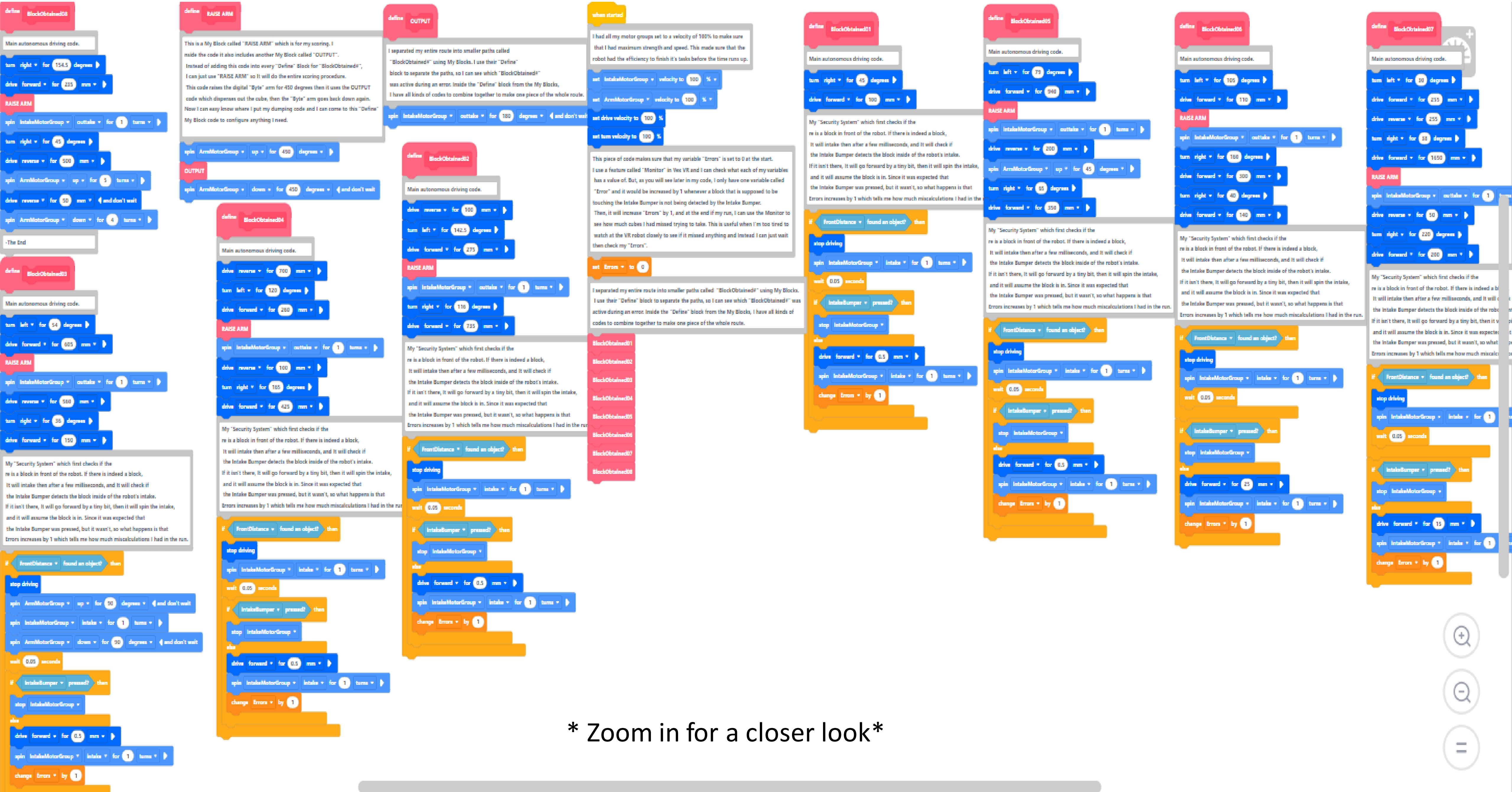
Skills Stop Time: 0 seconds remaining

- In my code, It contains 2 different sensor detection: Front Distance sensor and Intake Bumper sensor. But, In total I use a total of 6 sensors, when you include the sensors I use in the monitor. So, technically I am using 6 sensors.
- My code is completely controlled by “My Blocks”. I have a “Main Path”, where I connect all “My Blocks” then I put my codes into My Blocks. How my path works is that the codes run in each “My Block” and whenever one’s action is done, the next “My Block” does what it is supposed to do. The whole purpose of “My Blocks” are supposed to separate the path into smaller sections to be more easily edited.
- Inside of my code, I have a “Security System”, where I check if I have a block when I’m supposed to, and If I do not have it, I will go forward by a little bit and intake again. This makes sure that I won’t miss a block and I also increase my variable called “Error” by 1.

How Did VEX VR Help Improve My Skills?

Did you know, In fact I started VEX 3 months ago? Yes, not start competing, but introduced to VEX 3 months ago. I started VEX in my school's VEX club. I first started to build the hero bot for this year (Byte) and I got to learn some decent coding in VEXCODE IQ. Then, I was introduced to VEX VR. I had a lot of experience in other coding languages (Lua, Python, Java, and Scratch) so I was really interested in VEX VR. I started practicing it day by day and eventually once I understood the logic, variables, and the unique "My Blocks" of VEXVR, I soon became the best coder in my club. After a month, my friend invited me to a team he just created in Caution Tape. When I first went, it was just us two on the Caution Tape team. He knew I was good at coding, so he decided to add me on the team. A few days after he joined, he contributed greatly in assisting with the teachers to teach me advanced turns and driving straight with Brain Inertial/Gyro. I learned a lot in VEX VR and I put all my knowledge into VEX IQ. Now, I have the knowledge to code complicated equations and logic systems in VEX IQ thanks to the online practice I had in VR when I did not have a physical robot yet. VR is great for newbies, as there are many features that IQ does not have and many features that IQ has and VR does not have. Thanks to the coding practice VR gave me, I was successful in getting higher scores autonomous in qualifiers.

Full View Of Code



```
define BlockObtained08
  Main autonomous driving code.
  turn right for 154.5 degrees
  drive forward for 235 mm
  RAISE ARM
  spin IntakeMotorGroup outtake for 1 turns
  turn right for 45 degrees
  drive reverse for 500 mm
  spin ArmMotorGroup up for 5 turns
  drive reverse for 50 mm and don't wait
  spin ArmMotorGroup down for 4 turns
  -The End
```

```
define BlockObtained03
  Main autonomous driving code.
  turn left for 54 degrees
  drive forward for 605 mm
  RAISE ARM
  spin IntakeMotorGroup outtake for 1 turns
  drive reverse for 580 mm
  turn right for 38 degrees
  drive forward for 150 mm
  My "Security System" which first checks if the
  re is a block in front of the robot. If there is indeed a block,
   It will intake then after a few milliseconds, and It will check if
   the Intake Bumper detects the block inside of the robot's intake.
  If it isn't there, It will go forward by a tiny bit, then it will spin the intake,
   and it will assume the block is in. Since it was expected that
   the Intake Bumper was pressed, but it wasn't, so what happens is that
  Errors increases by 1 which tells me how much miscalculations I had in the run.
  if FrontDistance found an object? then
    stop driving
    spin ArmMotorGroup up for 90 degrees and don't wait
    spin IntakeMotorGroup intake for 1 turns
    spin ArmMotorGroup down for 90 degrees and don't wait
    wait 0.05 seconds
    if IntakeBumper pressed? then
      stop IntakeMotorGroup
    else
      drive forward for 0.5 mm
      spin IntakeMotorGroup intake for 1 turns
      change Errors by 1
```

```
define RAISE ARM
  This is a My Block called "RAISE ARM" which is for my scoring. I
  nside the code it also includes another My Block called "OUTPUT".
   Instead of adding this code into every "Define" Block for "BlockObtained#",
   I can just use "RAISE ARM" so It will do the entire scoring procedure.
   This code raises the digital "Byte" arm for 450 degrees then it uses the OUTPUT
   code which dispenses out the cube, then the "Byte" arm goes back down again.
  Now I can easy know where I put my dumping code and I can come to this "Define"
  My Block code to configure anything I need.
  spin ArmMotorGroup up for 450 degrees
  OUTPUT
  spin ArmMotorGroup down for 450 degrees and don't wait
```

```
define BlockObtained04
  Main autonomous driving code.
  drive reverse for 700 mm
  turn left for 120 degrees
  drive forward for 260 mm
  RAISE ARM
  spin IntakeMotorGroup outtake for 1 turns
  drive reverse for 100 mm
  turn right for 165 degrees
  drive forward for 425 mm
  My "Security System" which first checks if the
  re is a block in front of the robot. If there is indeed a block,
   It will intake then after a few milliseconds, and It will check if
   the Intake Bumper detects the block inside of the robot's intake.
  If it isn't there, It will go forward by a tiny bit, then it will spin the intake,
   and it will assume the block is in. Since it was expected that
   the Intake Bumper was pressed, but it wasn't, so what happens is that
  Errors increases by 1 which tells me how much miscalculations I had in the rur
  if FrontDistance found an object? then
    stop driving
    spin IntakeMotorGroup intake for 1 turns
    wait 0.05 seconds
    if IntakeBumper pressed? then
      stop IntakeMotorGroup
    else
      drive forward for 0.5 mm
      spin IntakeMotorGroup intake for 1 turns
      change Errors by 1
```

```
define OUTPUT
  I separated my entire route into smaller paths called
   "BlockObtained#" using My Blocks. I use their "Define"
   block to separate the paths, so I can see which "BlockObtained#"
   was active during an error. Inside the "Define" block from the My Blocks,
   I have all kinds of codes to combine together to make one piece of the whole route.
  spin IntakeMotorGroup outtake for 180 degrees and don't wai
```

```
define BlockObtained02
  Main autonomous driving code.
  drive reverse for 100 mm
  turn left for 142.5 degrees
  drive forward for 275 mm
  RAISE ARM
  spin IntakeMotorGroup outtake for 1 turns
  turn right for 116 degrees
  drive forward for 735 mm
  My "Security System" which first checks if the
  re is a block in front of the robot. If there is indeed a block,
   It will intake then after a few milliseconds, and It will check if
   the Intake Bumper detects the block inside of the robot's intake.
  If it isn't there, It will go forward by a tiny bit, then it will spin the intake,
   and it will assume the block is in. Since it was expected that
   the Intake Bumper was pressed, but it wasn't, so what happens is that
  Errors increases by 1 which tells me how much miscalculations I had in the rur
  if FrontDistance found an object? then
    stop driving
    spin IntakeMotorGroup intake for 1 turns
    wait 0.05 seconds
    if IntakeBumper pressed? then
      stop IntakeMotorGroup
    else
      drive forward for 0.5 mm
      spin IntakeMotorGroup intake for 1 turns
      change Errors by 1
```

```
when started
  I had all my motor groups set to a velocity of 100% to make sure
   that I had maximum strength and speed. This made sure that the
  robot had the efficiency to finish it's tasks before the time runs up.
  set IntakeMotorGroup velocity to 100 %
  set ArmMotorGroup velocity to 100 %
  set drive velocity to 100 %
  set turn velocity to 100 %
  This piece of code makes sure that my variable "Errors" is set to 0 at the start.
  I use a feature called "Monitor" in Vex VR and I can check what each of my variables
   has a value of. But, as you will see later in my code, I only have one variable called
   "Error" and it would be increased by 1 whenever a block that is supposed to be
  touching the Intake Bumper is not being detected by the Intake Bumper.
   Then, it will increase "Errors" by 1, and at the end if my run, I can use the Monitor to
   see how much cubes I had missed trying to take. This is useful when I'm too tired to
   watch at the VR robot closely to see if it missed anything and instead I can just wait
  then check my "Errors".
  set Errors to 0
  I separated my entire route into smaller paths called  "BlockObtained#" using My Blocks.
   I use their "Define" block to separate the paths, so I can see which "BlockObtained#" was
  active during an error. Inside the "Define" block from the My Blocks, I have all kinds of
  codes to combine together to make one piece of the whole route.
  BlockObtained01
  BlockObtained02
  BlockObtained03
  BlockObtained04
  BlockObtained05
  BlockObtained06
  BlockObtained07
  BlockObtained08
```

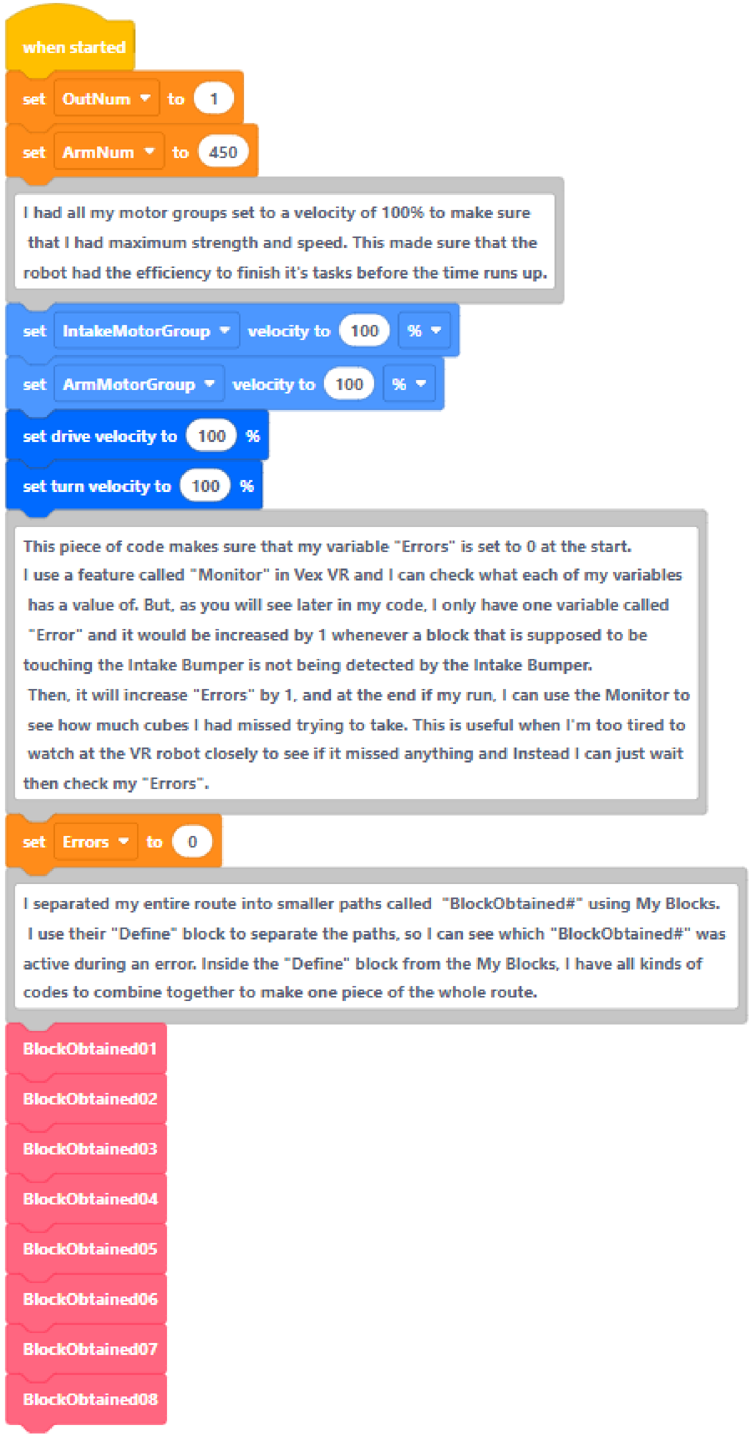
```
define BlockObtained01
  Main autonomous driving code.
  turn right for 45 degrees
  drive forward for 100 mm
  My "Security System" which first checks if the
  re is a block in front of the robot. If there is indeed a block,
   It will intake then after a few milliseconds, and It will check if
   the Intake Bumper detects the block inside of the robot's intake.
  If it isn't there, It will go forward by a tiny bit, then it will spin the intake,
   and it will assume the block is in. Since it was expected that
   the Intake Bumper was pressed, but it wasn't, so what happens is that
  Errors increases by 1 which tells me how much miscalculations I had in the
  if FrontDistance found an object? then
    stop driving
    spin IntakeMotorGroup intake for 1 turns
    wait 0.05 seconds
    if IntakeBumper pressed? then
      stop IntakeMotorGroup
    else
      drive forward for 0.5 mm
      spin IntakeMotorGroup intake for 1 turns
      change Errors by 1
```

```
define BlockObtained05
  Main autonomous driving code.
  turn left for 73 degrees
  drive forward for 940 mm
  RAISE ARM
  spin IntakeMotorGroup outtake for 1 turns
  drive reverse for 200 mm
  spin ArmMotorGroup up for 45 degrees
  turn right for 65 degrees
  drive forward for 350 mm
  My "Security System" which first checks if the
  re is a block in front of the robot. If there is indeed a block,
   It will intake then after a few milliseconds, and It will check if
   the Intake Bumper detects the block inside of the robot's intake.
  If it isn't there, It will go forward by a tiny bit, then it will spin the intake,
   and it will assume the block is in. Since it was expected that
   the Intake Bumper was pressed, but it wasn't, so what happens is that
  Errors increases by 1 which tells me how much miscalculations I had in the run.
  if FrontDistance found an object? then
    stop driving
    spin IntakeMotorGroup intake for 1 turns
    wait 0.05 seconds
    if IntakeBumper pressed? then
      stop IntakeMotorGroup
    else
      drive forward for 0.5 mm
      spin IntakeMotorGroup intake for 1 turns
      change Errors by 1
```

```
define BlockObtained06
  Main autonomous driving code.
  turn left for 105 degrees
  drive forward for 110 mm
  RAISE ARM
  spin IntakeMotorGroup outtake for 1 turns
  turn right for 160 degrees
  drive forward for 300 mm
  turn right for 40 degrees
  drive forward for 140 mm
  My "Security System" which first checks if the
  re is a block in front of the robot. If there is indeed a block,
   It will intake then after a few milliseconds, and It will check if
   the Intake Bumper detects the block inside of the robot's intake.
  If it isn't there, It will go forward by a tiny bit, then it will spin the intake,
   and it will assume the block is in. Since it was expected that
   the Intake Bumper was pressed, but it wasn't, so what happens is that
  Errors increases by 1 which tells me how much miscalculations I had in the run.
  if FrontDistance found an object? then
    stop driving
    spin IntakeMotorGroup intake for 1 turns
    wait 0.05 seconds
  if IntakeBumper pressed? then
    stop IntakeMotorGroup
  else
    drive forward for 25 mm
    spin IntakeMotorGroup intake for 1 turns
    change Errors by 1
```

```
define BlockObtained07
  Main autonomous driving code.
  turn left for 30 degrees
  drive forward for 255 mm
  drive reverse for 255 mm
  turn right for 38 degrees
  drive forward for 1650 mm
  RAISE ARM
  spin IntakeMotorGroup outtake for 1
  drive reverse for 50 mm
  turn right for 220 degrees
  drive forward for 200 mm
  My "Security System" which first checks if the
  re is a block in front of the robot. If there is indeed a b
   It will intake then after a few milliseconds, and It will
   the Intake Bumper detects the block inside of the robo
  If it isn't there, It will go forward by a tiny bit, then it w
   and it will assume the block is in. Since it was expecte
   the Intake Bumper was pressed, but it wasn't, so what
  Errors increases by 1 which tells me how much miscalc
  if FrontDistance found an object? then
    stop driving
    spin IntakeMotorGroup intake for 1
    wait 0.05 seconds
  if IntakeBumper pressed? then
    stop IntakeMotorGroup
  else
    drive forward for 15 mm
    spin IntakeMotorGroup intake for 1
    change Errors by 1
```

* Zoom in for a closer look*



```
when started
set OutNum ▾ to 1
set ArmNum ▾ to 450
```

I had all my motor groups set to a velocity of 100% to make sure that I had maximum strength and speed. This made sure that the robot had the efficiency to finish it's tasks before the time runs up.

```
set IntakeMotorGroup ▾ velocity to 100 % ▾
set ArmMotorGroup ▾ velocity to 100 % ▾
set drive velocity to 100 %
set turn velocity to 100 %
```

This piece of code makes sure that my variable "Errors" is set to 0 at the start. I use a feature called "Monitor" in Vex VR and I can check what each of my variables has a value of. But, as you will see later in my code, I only have one variable called "Error" and it would be increased by 1 whenever a block that is supposed to be touching the Intake Bumper is not being detected by the Intake Bumper. Then, it will increase "Errors" by 1, and at the end if my run, I can use the Monitor to see how much cubes I had missed trying to take. This is useful when I'm too tired to watch at the VR robot closely to see if it missed anything and Instead I can just wait then check my "Errors".

```
set Errors ▾ to 0
```

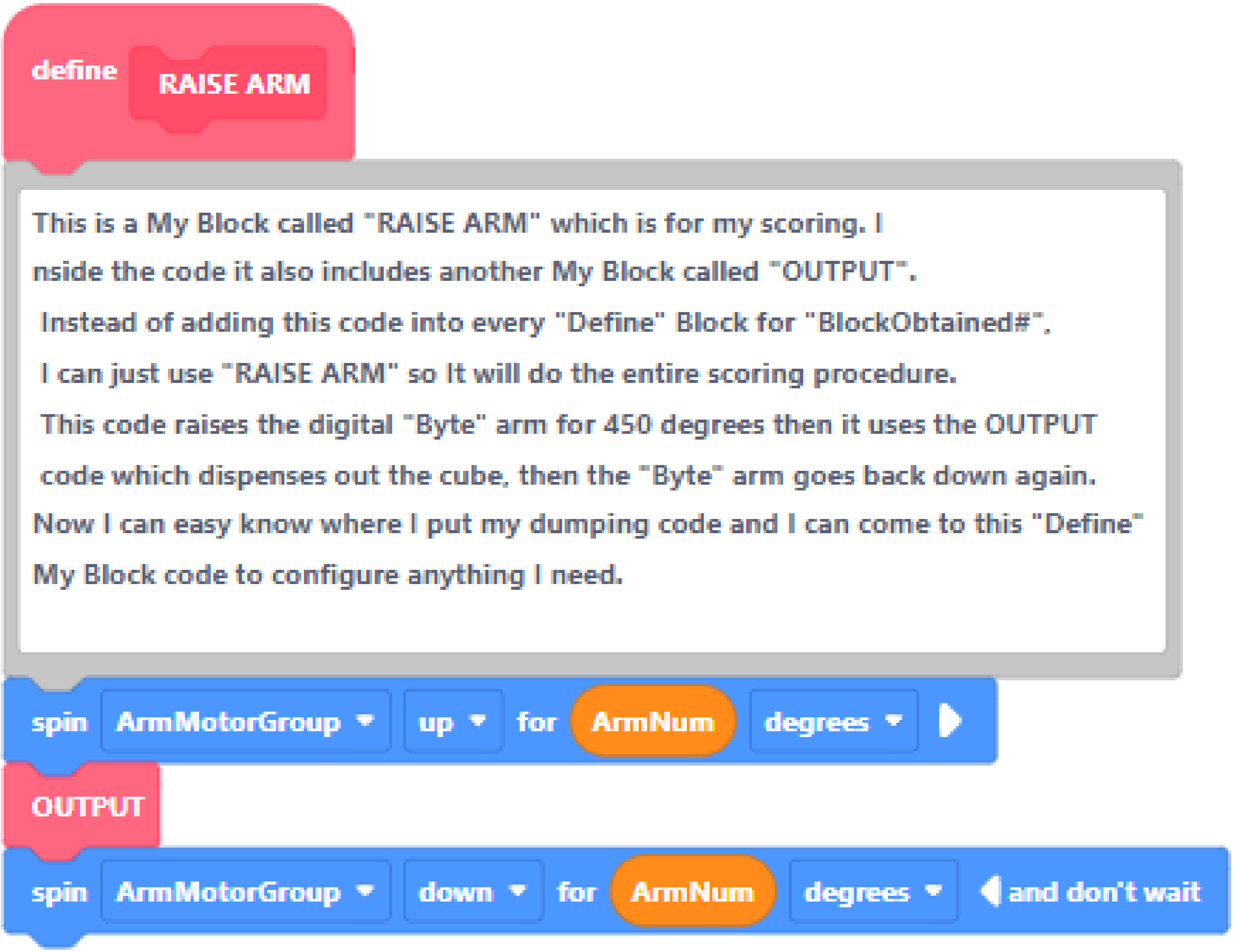
I separated my entire route into smaller paths called "BlockObtained#" using My Blocks. I use their "Define" block to separate the paths, so I can see which "BlockObtained#" was active during an error. Inside the "Define" block from the My Blocks, I have all kinds of codes to combine together to make one piece of the whole route.

```
BlockObtained01
BlockObtained02
BlockObtained03
BlockObtained04
BlockObtained05
BlockObtained06
BlockObtained07
BlockObtained08
```

The Route

These blocks are the very center of all my code. Without them, my code would not work at all. This code combines the smaller paths (BlockObtained_) together and it also sets my variable to 0 and sets my velocities to 100%. On the top, I set “Outnum” to 1 and “ArmNum” to 450.

RAISE ARM (Scoring Code)



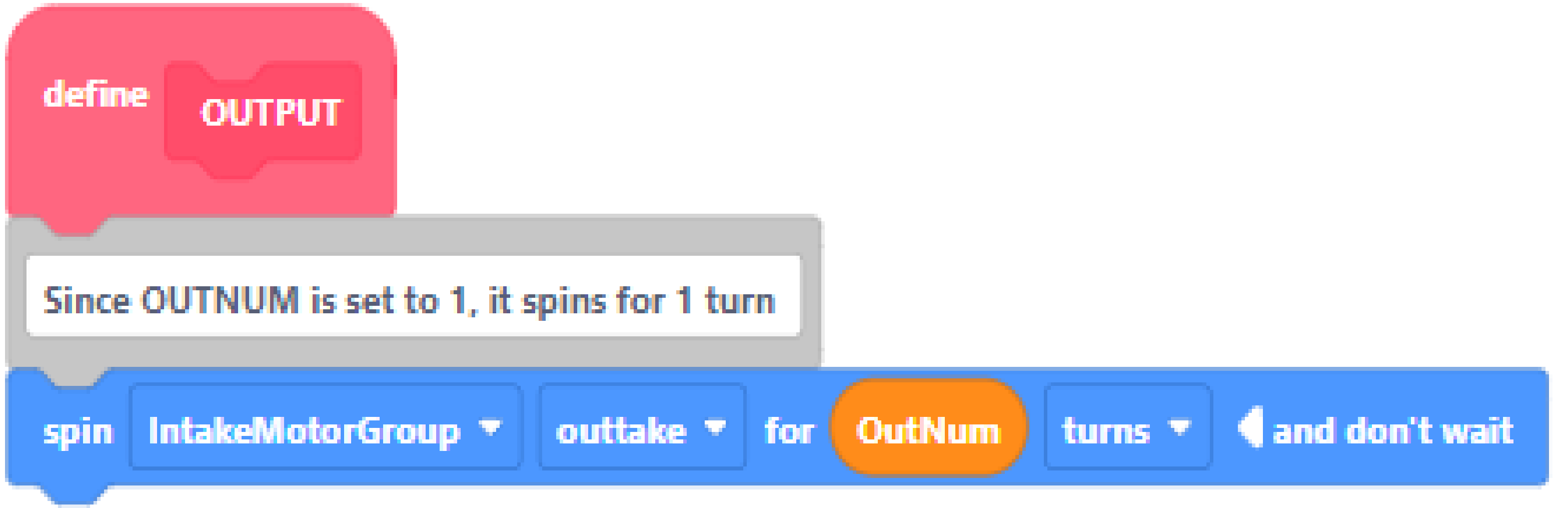
```
define RAISE ARM
```

This is a My Block called "RAISE ARM" which is for my scoring. I
nside the code it also includes another My Block called "OUTPUT".
Instead of adding this code into every "Define" Block for "BlockObtained#".
I can just use "RAISE ARM" so It will do the entire scoring procedure.
This code raises the digital "Byte" arm for 450 degrees then it uses the OUTPUT
code which dispenses out the cube, then the "Byte" arm goes back down again.
Now I can easy know where I put my dumping code and I can come to this "Define"
My Block code to configure anything I need.

```
spin ArmMotorGroup up for ArmNum degrees
OUTPUT
spin ArmMotorGroup down for ArmNum degrees and don't wait
```

This code is a My Block scripted for the online Byte to raise his arm, spin his outtake, then drop his arm.

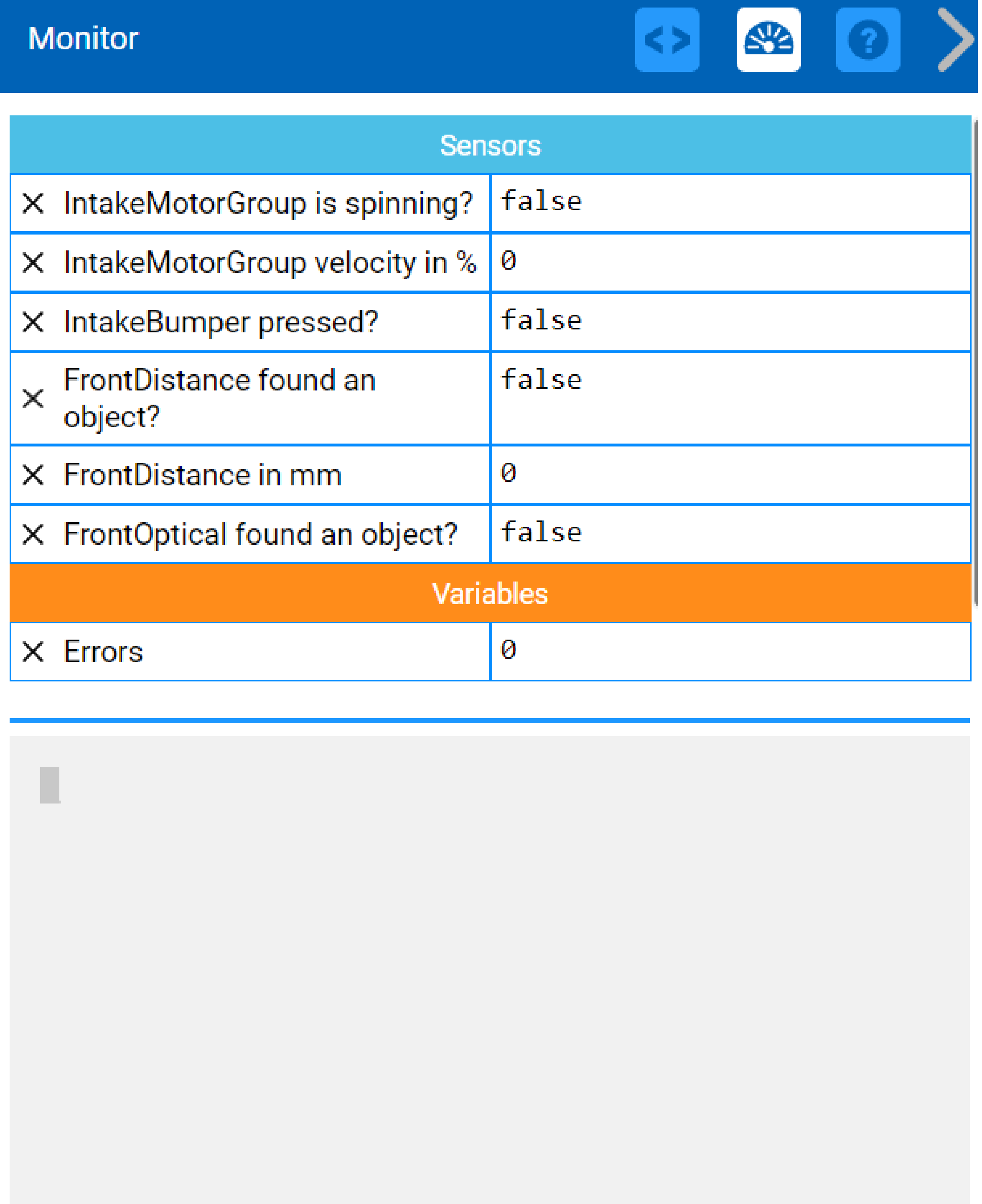
OUTPUT



```
define OUTPUT
Since OUTNUM is set to 1, it spins for 1 turn
spin IntakeMotorGroup ▾ outtake ▾ for OutNum turns ▾ and don't wait
```

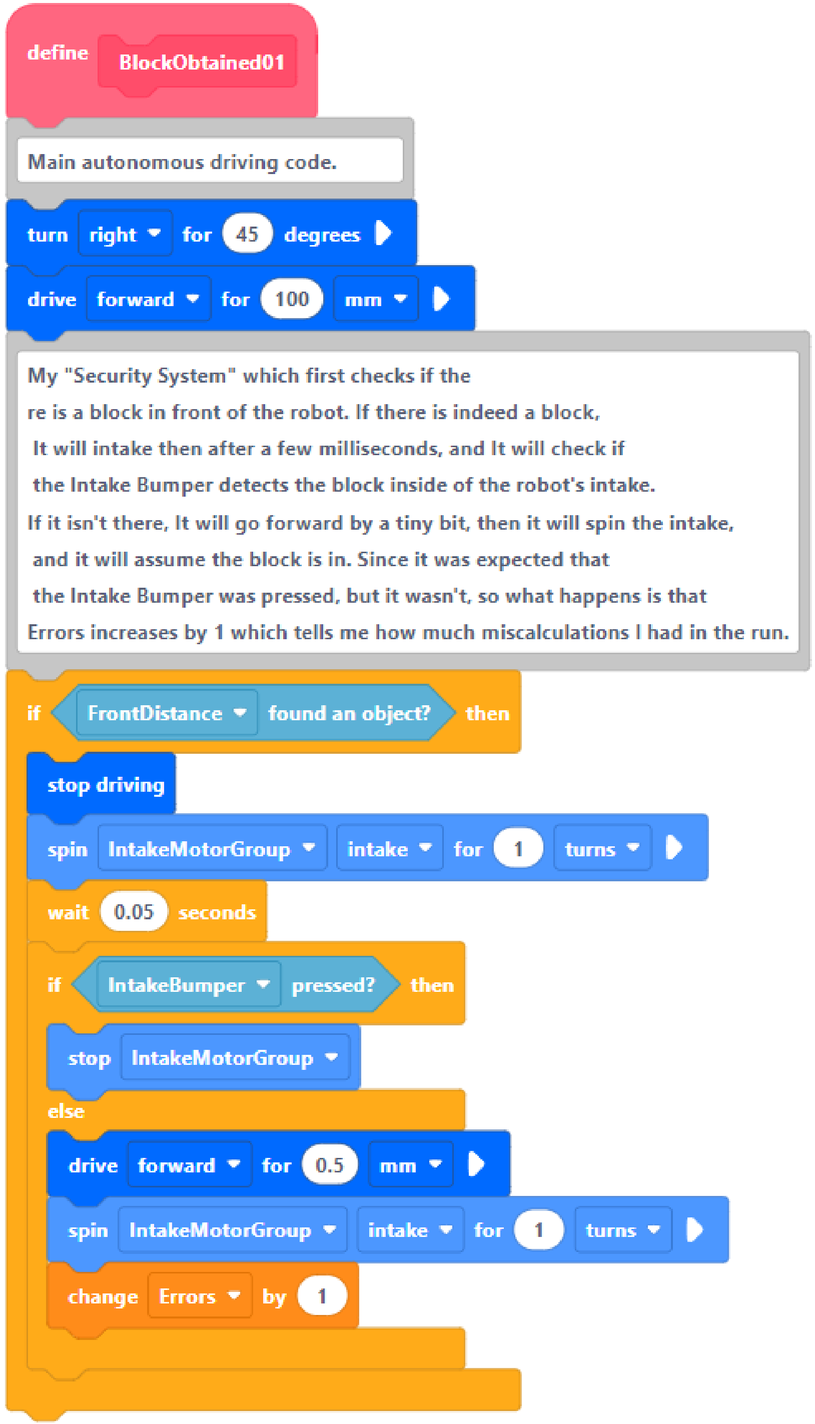
Pretty self explanatory. Spins the outtake of the online Byte for 1 turn.

The Monitor



Sensors	
× IntakeMotorGroup is spinning?	false
× IntakeMotorGroup velocity in %	0
× IntakeBumper pressed?	false
× FrontDistance found an object?	false
× FrontDistance in mm	0
× FrontOptical found an object?	false
Variables	
× Errors	0

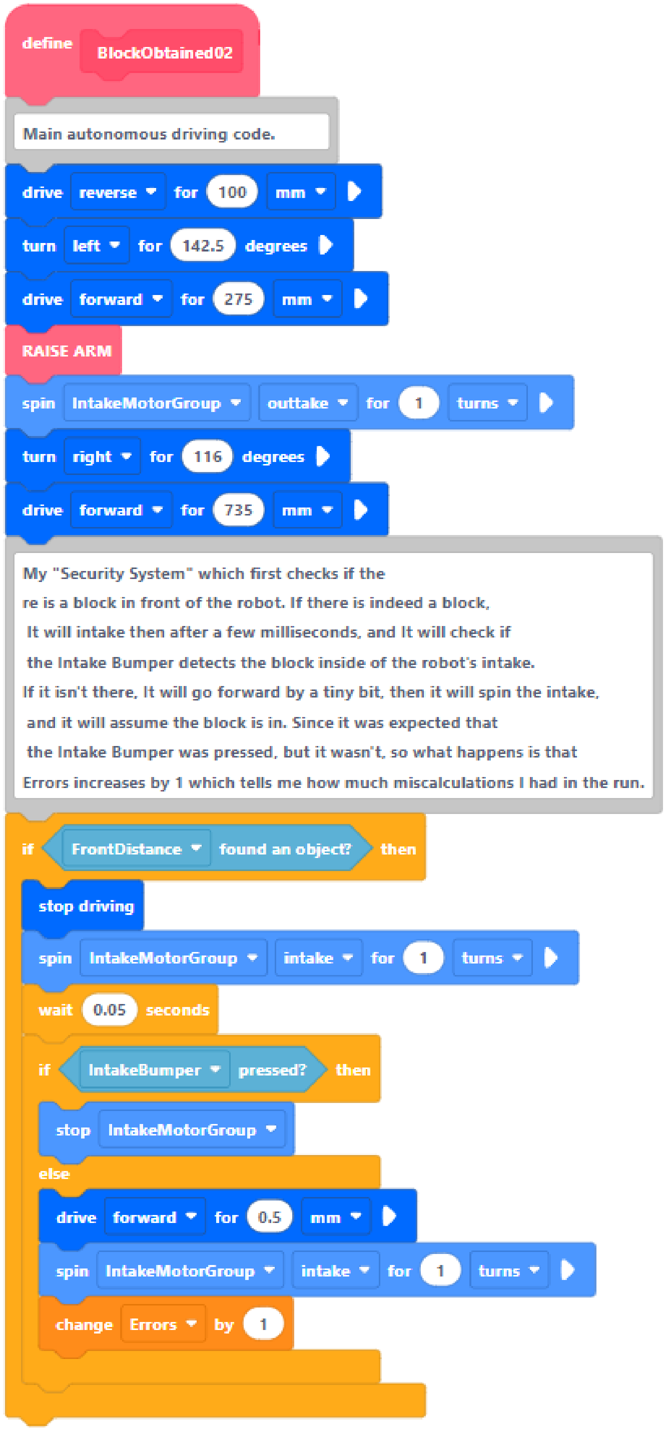
Inside the monitor, I get to see values and true/false of Sensors and Variables. At the end of every run, I check my custom variable “Errors”. What it does is that I have a security system that checks if I had any miscalculations about the positions of the cubes on the map. I can see the number of cubes that I would’ve missed if I did not have a “Security System” so I often fix my code thanks to the Monitor and it’s amazing functions. I use a total of one variable from my “Security System”, which I will show later, and I also use 6 different sensors in my Monitor.



```
define BlockObtained01
Main autonomous driving code.
turn right for 45 degrees
drive forward for 100 mm
My "Security System" which first checks if the
re is a block in front of the robot. If there is indeed a block,
 It will intake then after a few milliseconds, and It will check if
 the Intake Bumper detects the block inside of the robot's intake.
If it isn't there, It will go forward by a tiny bit, then it will spin the intake,
 and it will assume the block is in. Since it was expected that
 the Intake Bumper was pressed, but it wasn't, so what happens is that
Errors increases by 1 which tells me how much miscalculations I had in the run.
if FrontDistance found an object? then
    stop driving
    spin IntakeMotorGroup intake for 1 turns
    wait 0.05 seconds
    if IntakeBumper pressed? then
        stop IntakeMotorGroup
    else
        drive forward for 0.5 mm
        spin IntakeMotorGroup intake for 1 turns
        change Errors by 1
```

Path #1 (BlockObtained01)

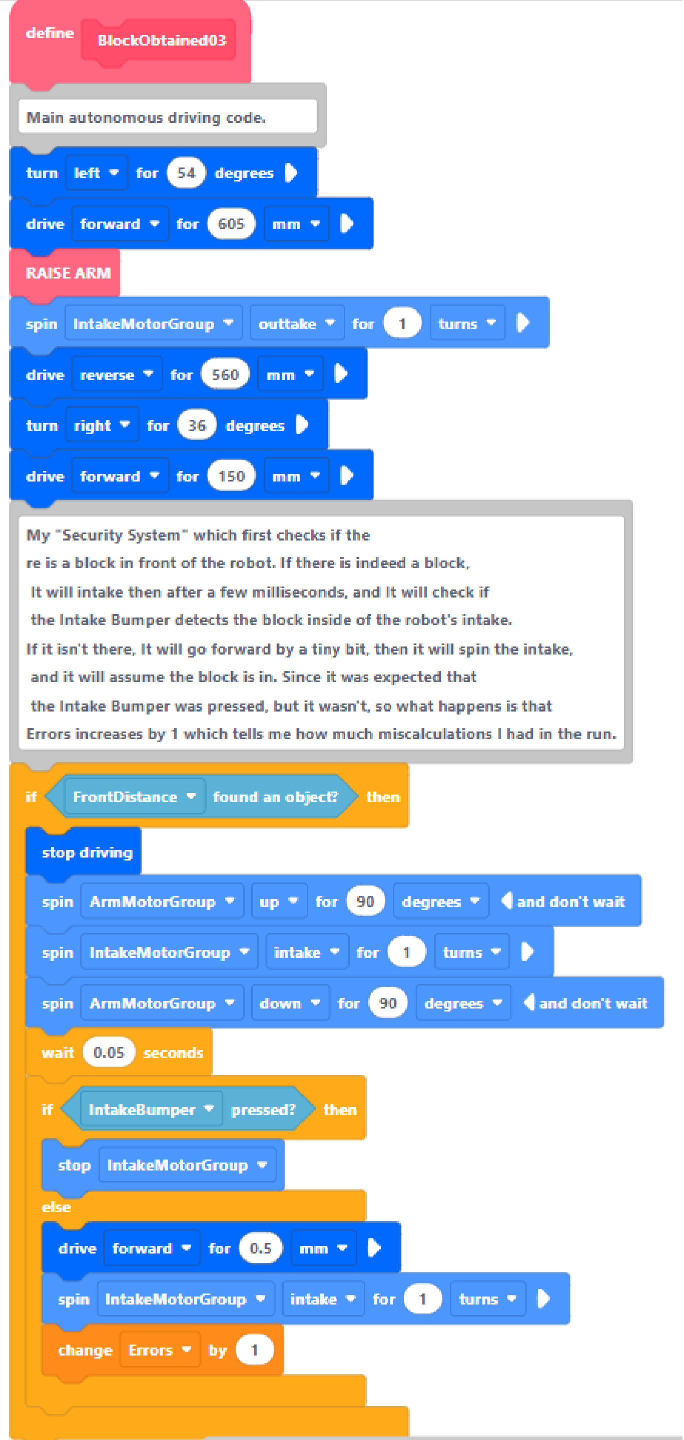
The first path in my route. Inside are the motion/drivetrain blocks for the path and my “security system”. I explained my “Security System” in the picture.



```
define BlockObtained02
Main autonomous driving code.
drive reverse ▾ for 100 mm ▾ ▶
turn left ▾ for 142.5 degrees ▶
drive forward ▾ for 275 mm ▾ ▶
RAISE ARM
spin IntakeMotorGroup ▾ outtake ▾ for 1 turns ▾ ▶
turn right ▾ for 116 degrees ▶
drive forward ▾ for 735 mm ▾ ▶
My "Security System" which first checks if the
re is a block in front of the robot. If there is indeed a block,
 It will intake then after a few milliseconds, and It will check if
 the Intake Bumper detects the block inside of the robot's intake.
If it isn't there, It will go forward by a tiny bit, then it will spin the intake,
 and it will assume the block is in. Since it was expected that
 the Intake Bumper was pressed, but it wasn't, so what happens is that
Errors increases by 1 which tells me how much miscalculations I had in the run.
if FrontDistance ▾ found an object? then
    stop driving
    spin IntakeMotorGroup ▾ intake ▾ for 1 turns ▾ ▶
    wait 0.05 seconds
    if IntakeBumper ▾ pressed? then
        stop IntakeMotorGroup ▾
    else
        drive forward ▾ for 0.5 mm ▾ ▶
        spin IntakeMotorGroup ▾ intake ▾ for 1 turns ▾ ▶
        change Errors ▾ by 1
```

Path #2 (BlockObtained02)

The second path in my route. Inside are the motion/drivetrain blocks for the path and my “Security System”, which I have explained in the picture



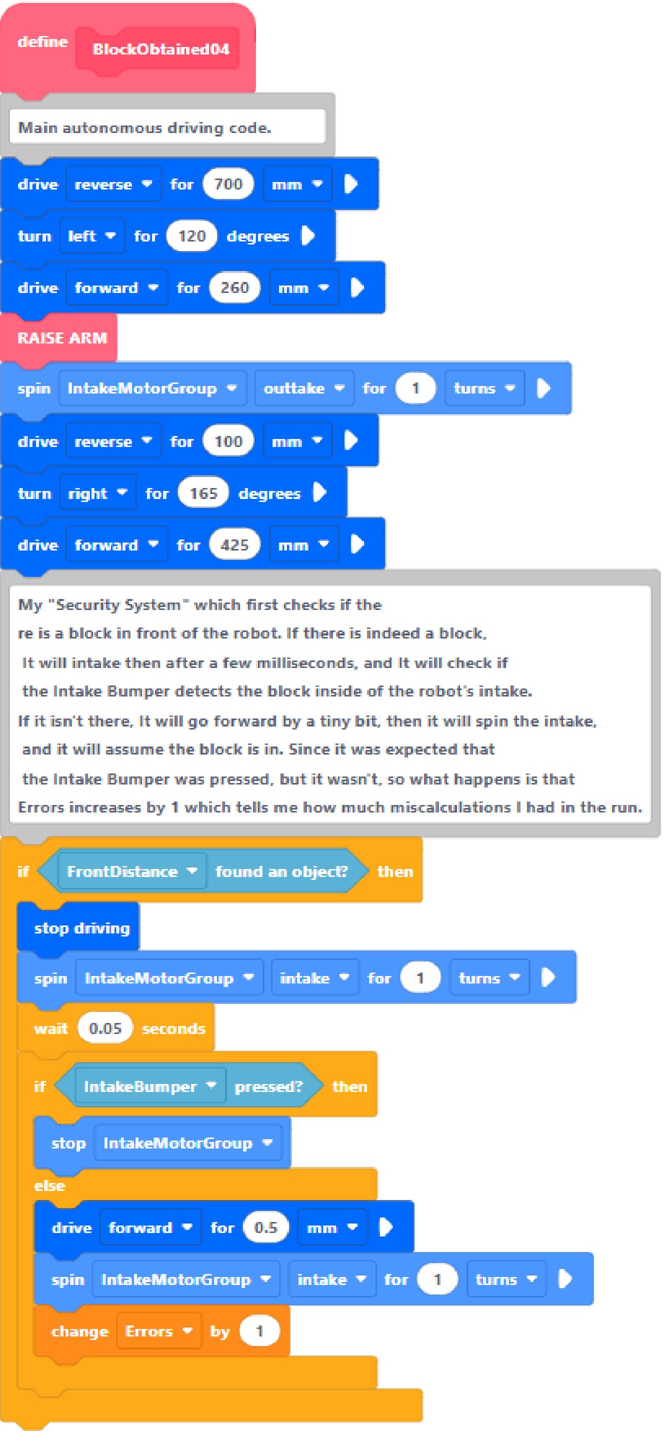
```
define BlockObtained03
Main autonomous driving code.
turn left for 54 degrees
drive forward for 605 mm
RAISE ARM
spin IntakeMotorGroup outtake for 1 turns
drive reverse for 560 mm
turn right for 36 degrees
drive forward for 150 mm
My "Security System" which first checks if the
re is a block in front of the robot. If there is indeed a block,
 It will intake then after a few milliseconds, and It will check if
 the Intake Bumper detects the block inside of the robot's intake.
If it isn't there, It will go forward by a tiny bit, then it will spin the intake,
 and it will assume the block is in. Since it was expected that
 the Intake Bumper was pressed, but it wasn't, so what happens is that
Errors increases by 1 which tells me how much miscalculations I had in the run.
if FrontDistance found an object? then
    stop driving
    spin ArmMotorGroup up for 90 degrees and don't wait
    spin IntakeMotorGroup intake for 1 turns
    spin ArmMotorGroup down for 90 degrees and don't wait
    wait 0.05 seconds
    if IntakeBumper pressed? then
        stop IntakeMotorGroup
    else
        drive forward for 0.5 mm
        spin IntakeMotorGroup intake for 1 turns
        change Errors by 1
```

Path #3 (BlockObtained03)

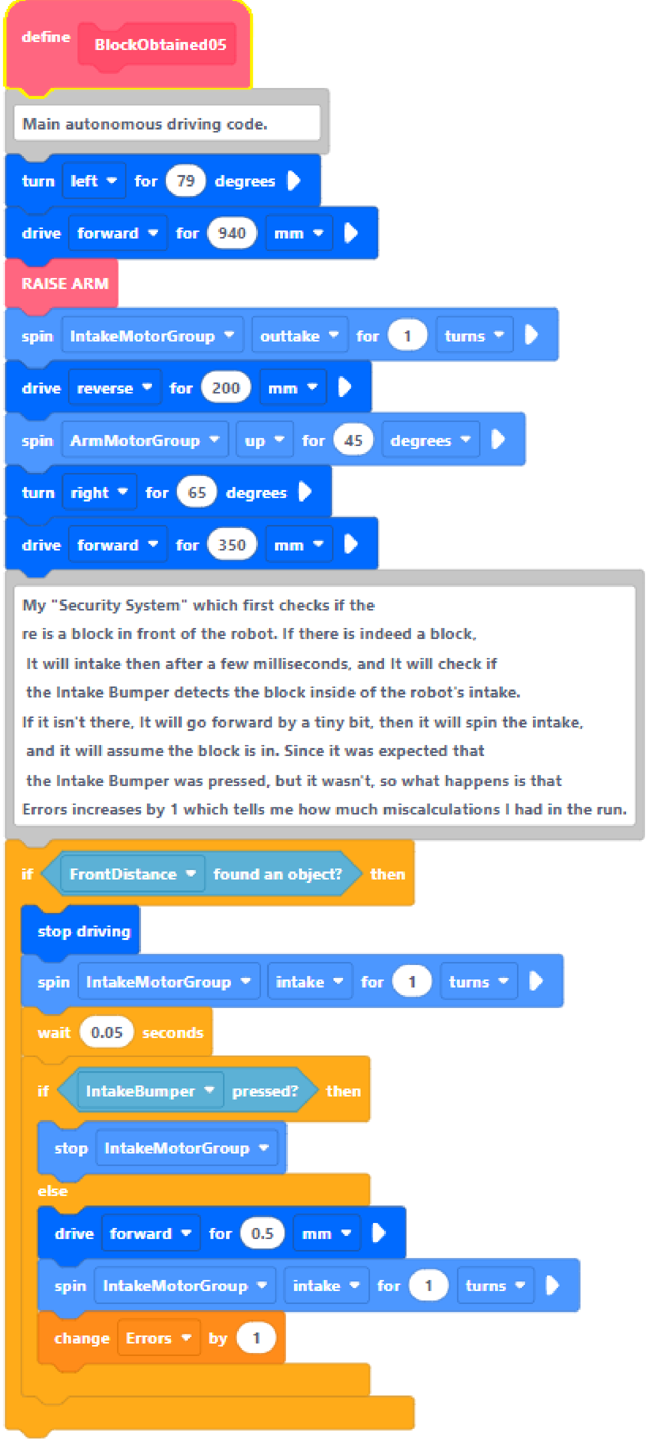
The third path in my route. Inside are the motion/drivetrain blocks for the path and my “Security System”, which I have explained in the picture

Path #4 (BlockObtained04)

The fourth path in my route. Inside are the motion/drivetrain blocks for the path and my “Security System”, which I have explained in the picture



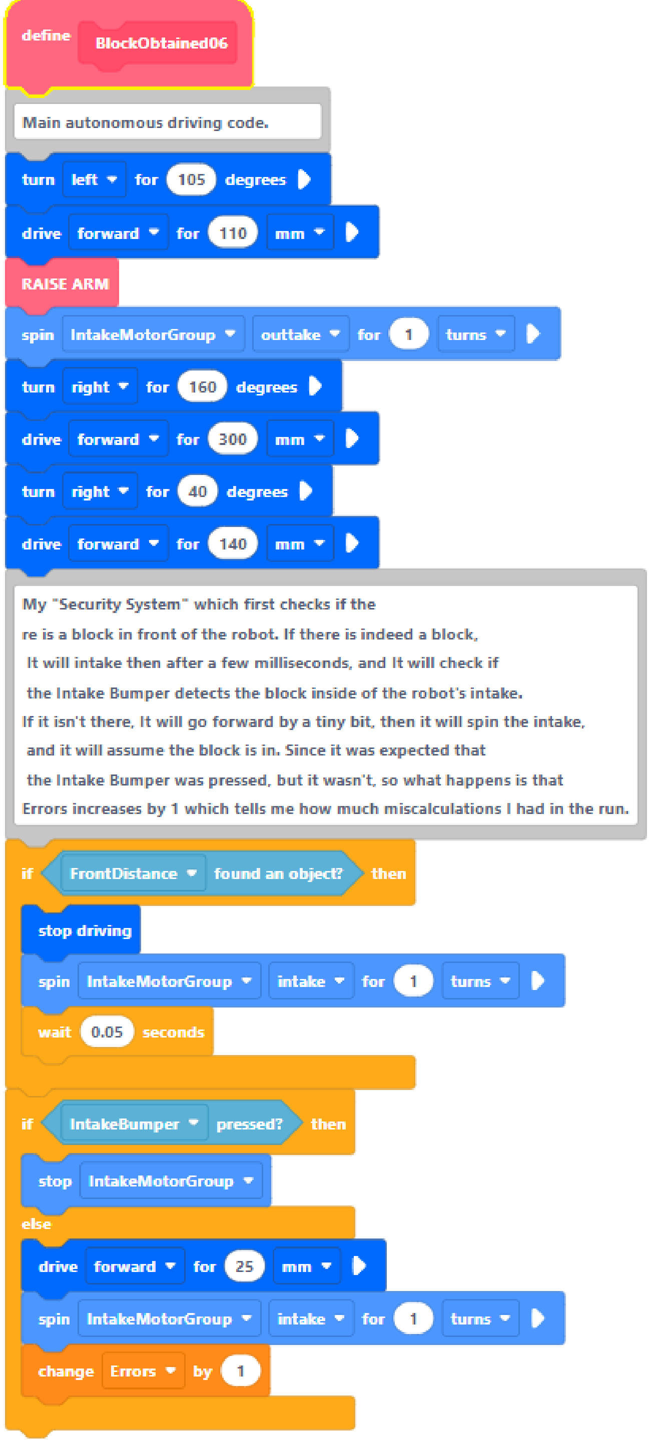
```
define BlockObtained04
Main autonomous driving code.
drive reverse for 700 mm
turn left for 120 degrees
drive forward for 260 mm
RAISE ARM
spin IntakeMotorGroup outtake for 1 turns
drive reverse for 100 mm
turn right for 165 degrees
drive forward for 425 mm
My "Security System" which first checks if the
re is a block in front of the robot. If there is indeed a block,
 It will intake then after a few milliseconds, and It will check if
 the Intake Bumper detects the block inside of the robot's intake.
If it isn't there, It will go forward by a tiny bit, then it will spin the intake,
 and it will assume the block is in. Since it was expected that
 the Intake Bumper was pressed, but it wasn't, so what happens is that
Errors increases by 1 which tells me how much miscalculations I had in the run.
if FrontDistance found an object? then
  stop driving
  spin IntakeMotorGroup intake for 1 turns
  wait 0.05 seconds
  if IntakeBumper pressed? then
    stop IntakeMotorGroup
  else
    drive forward for 0.5 mm
    spin IntakeMotorGroup intake for 1 turns
    change Errors by 1
```



```
define BlockObtained05
Main autonomous driving code.
turn left for 79 degrees
drive forward for 940 mm
RAISE ARM
spin IntakeMotorGroup outtake for 1 turns
drive reverse for 200 mm
spin ArmMotorGroup up for 45 degrees
turn right for 65 degrees
drive forward for 350 mm
My "Security System" which first checks if the
re is a block in front of the robot. If there is indeed a block,
 It will intake then after a few milliseconds, and It will check if
 the Intake Bumper detects the block inside of the robot's intake.
If it isn't there, It will go forward by a tiny bit, then it will spin the intake,
 and it will assume the block is in. Since it was expected that
 the Intake Bumper was pressed, but it wasn't, so what happens is that
Errors increases by 1 which tells me how much miscalculations I had in the run.
if FrontDistance found an object? then
    stop driving
    spin IntakeMotorGroup intake for 1 turns
    wait 0.05 seconds
    if IntakeBumper pressed? then
        stop IntakeMotorGroup
    else
        drive forward for 0.5 mm
        spin IntakeMotorGroup intake for 1 turns
        change Errors by 1
```

Path #5 (BlockObtained05)

The fifth path in my route. Inside are the motion/drivetrain blocks for the path and my “Security System”, which I have explained in the picture



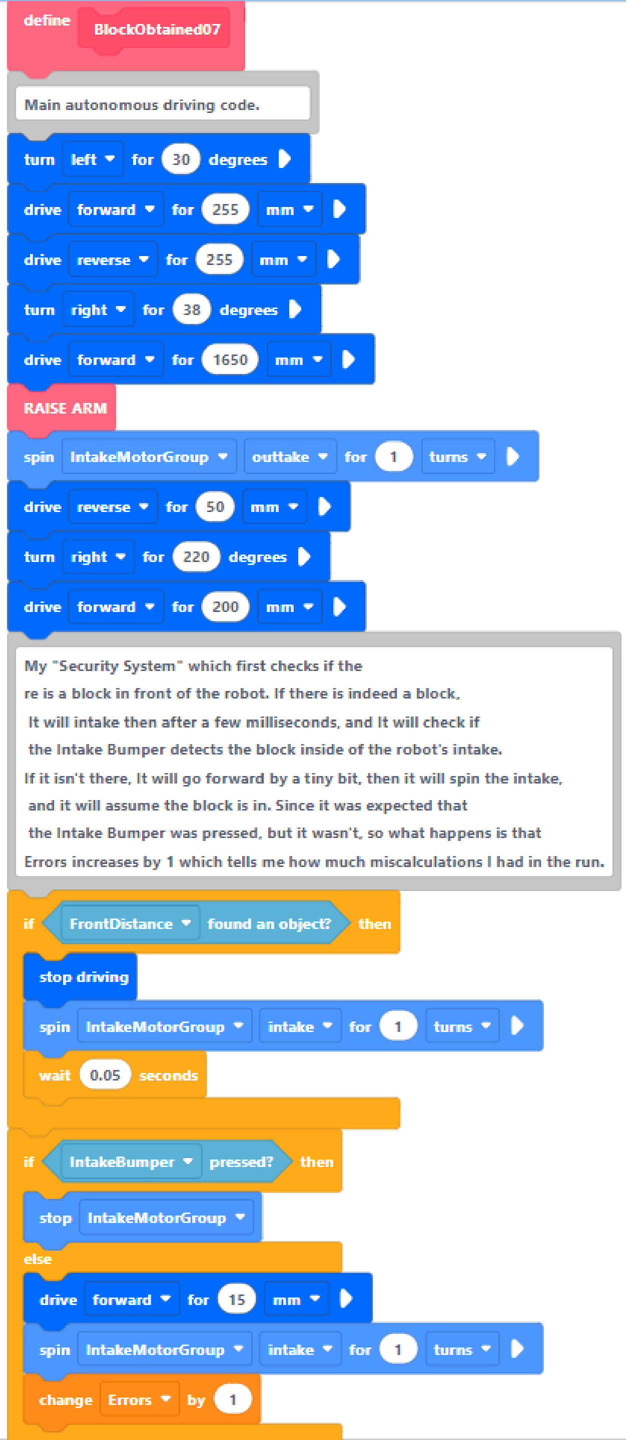
```
define BlockObtained06
Main autonomous driving code.
turn left ▾ for 105 degrees ▶
drive forward ▾ for 110 mm ▾ ▶
RAISE ARM
spin IntakeMotorGroup ▾ outtake ▾ for 1 turns ▾ ▶
turn right ▾ for 160 degrees ▶
drive forward ▾ for 300 mm ▾ ▶
turn right ▾ for 40 degrees ▶
drive forward ▾ for 140 mm ▾ ▶
My "Security System" which first checks if the
re is a block in front of the robot. If there is indeed a block,
 It will intake then after a few milliseconds, and It will check if
 the Intake Bumper detects the block inside of the robot's intake.
If it isn't there, It will go forward by a tiny bit, then it will spin the intake,
 and it will assume the block is in. Since it was expected that
 the Intake Bumper was pressed, but it wasn't, so what happens is that
Errors increases by 1 which tells me how much miscalculations I had in the run.
if FrontDistance ▾ found an object? then
    stop driving
    spin IntakeMotorGroup ▾ intake ▾ for 1 turns ▾ ▶
    wait 0.05 seconds
if IntakeBumper ▾ pressed? then
    stop IntakeMotorGroup ▾
else
    drive forward ▾ for 25 mm ▾ ▶
    spin IntakeMotorGroup ▾ intake ▾ for 1 turns ▾ ▶
    change Errors ▾ by 1
```

Path #6 (BlockObtained06)

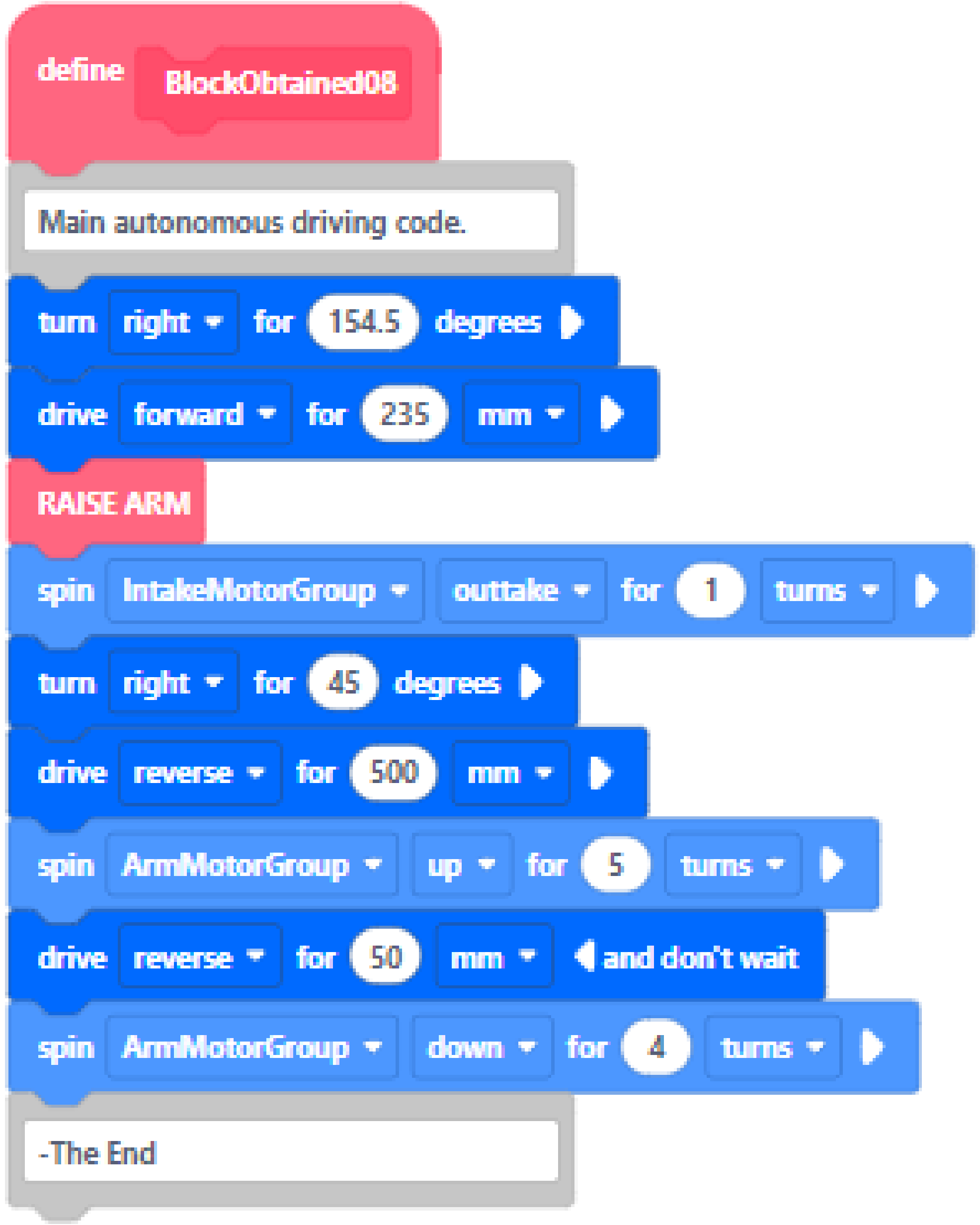
The sixth path in my route. Inside are the motion/drivetrain blocks for the path and my “Security System”, which I have explained in the picture

Path #7 (BlockObtained07)

The seventh path in my route. Inside are the motion/drivetrain blocks for the path and my “Security System”, which I have explained in the picture



```
define BlockObtained07
Main autonomous driving code.
turn left for 30 degrees
drive forward for 255 mm
drive reverse for 255 mm
turn right for 38 degrees
drive forward for 1650 mm
RAISE ARM
spin IntakeMotorGroup outtake for 1 turns
drive reverse for 50 mm
turn right for 220 degrees
drive forward for 200 mm
My "Security System" which first checks if the
re is a block in front of the robot. If there is indeed a block,
 It will intake then after a few milliseconds, and It will check if
 the Intake Bumper detects the block inside of the robot's intake.
If it isn't there, It will go forward by a tiny bit, then it will spin the intake,
 and it will assume the block is in. Since it was expected that
 the Intake Bumper was pressed, but it wasn't, so what happens is that
Errors increases by 1 which tells me how much miscalculations I had in the run.
if FrontDistance found an object? then
    stop driving
    spin IntakeMotorGroup intake for 1 turns
    wait 0.05 seconds
if IntakeBumper pressed? then
    stop IntakeMotorGroup
else
    drive forward for 15 mm
    spin IntakeMotorGroup intake for 1 turns
    change Errors by 1
```



```
define BlockObtained08
Main autonomous driving code.
turn right for 154.5 degrees
drive forward for 235 mm
RAISE ARM
spin IntakeMotorGroup outtake for 1 turns
turn right for 45 degrees
drive reverse for 500 mm
spin ArmMotorGroup up for 5 turns
drive reverse for 50 mm and don't wait
spin ArmMotorGroup down for 4 turns
-The End
```

Path #8 (BlockObtained08)

The last path in my route. Inside are the motion/drivetrain blocks for the path and my “Security System” is not needed, as I am not picking up any blocks.

Thanks for reading!

This took me a bunch of time and it spent a lot of my homework time. Thank you so much for reading and I hope you understand well my code, intentions, and improvements. For your information, this code is heavily relied on My Blocks, The Monitor, and Sensors/Variables.