

```

1 ## TeamCode Module
2
3 Welcome!
4
5 This module, TeamCode, is the place where you will write/
6 paste the code for your team's
7 robot controller App. This module is currently empty (a
8 clean slate) but the
9 process for adding OpModes is straightforward.
10
11 ## Creating your own OpModes
12
13 The easiest way to create your own OpMode is to copy a
14 Sample OpMode and make it your own.
15
16 Sample opmodes exist in the FtcRobotController module.
17 To locate these samples, find the FtcRobotController
18 module in the "Project/Android" tab.
19
20 Expand the following tree elements:
21 FtcRobotController / java / org.firstinspires.ftc.
22   robotcontroller / external / samples
23
24 A range of different samples classes can be seen in this
25 folder.
26 The class names follow a naming convention which indicates
27 the purpose of each class.
28 The full description of this convention is found in the
29 samples/sample_convention.md file.
30
31 A brief synopsis of the naming convention is given here:
32 The prefix of the name will be one of the following:
33
34 * Basic: This is a minimally functional OpMode used to
35   illustrate the skeleton/structure
36   of a particular style of OpMode. These are
37   bare bones examples.
38 * Sensor: This is a Sample OpMode that shows how to use
39   a specific sensor.
40   It is not intended as a functioning robot, it
41   is simply showing the minimal code
42   required to read and display the sensor values
43   .
44 * Hardware: This is not an actual OpMode, but a helper
45   class that is used to describe

```

32 one particular robot's hardware devices: eg:
33 for a Pushbot. Look at any Pushbot sample to see how this can be used in
34 an OpMode.
35 *** Pushbot:** This is a Sample OpMode that uses the Pushbot robot structure as a base.
36 *** Concept:** This is a sample OpMode that illustrates performing a specific function or concept.
37 These may be complex, but their operation should be explained clearly in the comments,
38 or the header should reference an external doc guide or tutorial.
39 *** Library:** This is a class, or set of classes used to implement some strategy.
40 These will typically NOT implement a full OpMode. Instead they will be included by an OpMode to provide some stand-alone capability.
42
43 Once you are familiar with the range of samples available, you can choose one to be the basis for your own robot. In all cases, the desired sample(s) needs to be copied into your TeamCode module to be used.
46
47 This is done inside Android Studio directly, using the following steps:
48
49 **1)** Locate the desired sample class in the Project/Android tree.
50
51 **2)** Right click on the sample class and select "Copy"
52
53 **3)** Expand the TeamCode / java folder
54
55 **4)** Right click on the org.firstinspires.ftc.teamcode folder and select "Paste"
56
57 **5)** You will be prompted for a class name for the copy. Choose something meaningful based on the purpose of this class.
58 Start with a capital letter, and remember that there may be more similar classes later.

```
60
61 Once your copy has been created, you should prepare it
for use on your robot.
62 This is done by adjusting the OpMode's name, and enabling
it to be displayed on the
63 Driver Station's OpMode list.
64
65 Each OpMode sample class begins with several lines of
code like the ones shown below:
66
67 ````
68 @TeleOp(name="Template: Linear OpMode", group="Linear
Opmode")
69 @Disabled
70 ````
71
72 The name that will appear on the driver station's "opmode
list" is defined by the code:
73 ` `name="Template: Linear OpMode" ````

74 You can change what appears between the quotes to better
describe your opmode.

75 The "group=" portion of the code can be used to help
organize your list of OpModes.

76
77 As shown, the current OpMode will NOT appear on the
driver station's OpMode list because of the
78 ` `@Disabled` ` annotation which has been included.
79 This line can simply be deleted, or commented out, to
make the OpMode visible.

80
81
82
83 ## ADVANCED Multi-Team App management: Cloning the
TeamCode Module

84
85 In some situations, you have multiple teams in your club
and you want them to all share
86 a common code organization, with each being able to *see*
the others code but each having
87 their own team module with their own code that they
maintain themselves.

88
89 In this situation, you might wish to clone the TeamCode
module, once for each of these teams.

90 Each of the clones would then appear along side each
```

90 other in the Android Studio module list,
91 together with the FtcRobotController module (and the
original TeamCode module).
92
93 Selective Team phones can then be programmed by selecting
the desired Module from the pulldown list
94 prior to clicking to the green Run arrow.
95
96 **Warning:** This is not for the inexperienced Software
developer.
97 You will need to be comfortable with File manipulations
and managing Android Studio Modules.
98 These changes are performed OUTSIDE of Android Studios,
so close Android Studios before you do this.
99
100 Also... Make a full project backup before you start this :
)
101
102 To clone TeamCode, do the following:
103
104 **Note:** Some names start with "Team" and others start with
"team". This is intentional.
105
106 **1)** Using your operating system file management tools,
copy the whole "TeamCode"
107 folder to a sibling folder with a corresponding new
name, eg: "Team0417".
108
109 **2)** In the new Team0417 folder, delete the TeamCode.iml
file.
110
111 **3)** In the new Team0417 folder, rename the "src/main/java/
org/firstinspires/ftc/teamcode" folder
112 to a matching name with a lowercase 'team' eg: "
team0417".
113
114 **4)** In the new Team0417/src/main folder, edit the "
AndroidManifest.xml" file, change the line that contains
115 package="org.firstinspires.ftc.teamcode"
116 to be
117 package="org.firstinspires.ftc.team0417"
118
119 **5)** Add: include ':Team0417' to the "/settings.gradle"
file.
120

121 **6)** Open up Android Studios and clean out any old files by using the menu to "Build/Clean Project""

```
1 package org.firstinspires.ftc.teamcode.Pixy;
2
3
4
5 import android.support.annotation.NonNull;
6
7 import com.qualcomm.robotcore.hardware.I2cAddr;
8 import com.qualcomm.robotcore.hardware.I2cDeviceSynch;
9 import com.qualcomm.robotcore.hardware.
10 I2cDeviceSynchDeviceWithParameters;
11 import com.qualcomm.robotcore.hardware.configuration.
12 I2cSensor;
13 import com.qualcomm.robotcore.hardware.configuration.
14 annotations.DeviceProperties;
15 import com.qualcomm.robotcore.hardware.configuration.
16 annotations.I2cDeviceType;
17 import com.qualcomm.robotcore.util.TypeConversion;
18
19 @I2cDeviceType()
20 @DeviceProperties(name = "PixyCam", description = "PixyCam"
21 , xmlTag = "PixyCam")
22 public class PixyCam extends
23 I2cDeviceSynchDeviceWithParameters<I2cDeviceSynch, PixyCam
24 .PixyCamParams> {
25
26     static final I2cAddr ADDRESS_I2C_DEFAULT = I2cAddr.
27 create7bit(0x54);
28
29     private static final int LEGO_QUERY_BASE = 0x50;
30     private static final int LEGO_QUERY_CC = 0x58;
31     private static final int EXTENDED_QUERY_BASE = 0x70;
32     private static final int EXTENDED_QUERY_CC = 0x78;
33     private static final int LEGO_QUERY_COUNT = 6;
34     private static final int LEGO_QUERY_SIG_COUNT = 5;
35     private static final int LEGO_QUERY_CC_COUNT = 6;
36     private static final int EXTENDED_QUERY_COUNT = 26;
37     private static final int EXTENDED_QUERY_BLOCK_SIZE = 5
38 ;
39     private static final int EXTENDED_QUERY_SIG_COUNT = 25
40 ;
41     private static final int EXTENDED_QUERY_SIG_BLOCK_SIZE
42 = 4;
43     private static final int EXTENDED_QUERY_CC_COUNT = 25;
44     private static final int EXTENDED_QUERY_CC_BLOCK_SIZE
45 = 6;
```

```
34     private static final int MIN_SIGNATURE = 1;
35     private static final int MAX_SIGNATURE = 7;
36
37     public static class PixyCamParams implements Cloneable
38     {
39         I2cAddr i2cAddr = ADDRESS_I2C_DEFAULT;
40
41         public PixyCamParams clone() {
42             try {
43                 return (PixyCamParams) super.clone();
44             } catch (CloneNotSupportedException e) {
45                 throw new RuntimeException("Internal Error
46 : Parameters not cloneable");
47             }
48         }
49
50         /**
51          * The ReadWindow used to do a PixyCam LEGO protocol
52          GeneralQuery
53          */
54         private I2cDeviceSynch.ReadWindow
55         legoProtocolGeneralQueryReadWindow;
56         /**
57          * The ReadWindows used to do the PixyCam LEGO
58          protocol SignatureQuery.
59          */
60         private I2cDeviceSynch.ReadWindow[]
61         legoProtocolSignatureQueryReadWindows;
62
63         /*
64          * The ReadWindow used to do the extneded query for
65          the enhanced FTC firmware
66          */
67         private I2cDeviceSynch.ReadWindow
68         extendedGeneralQueryReadWindow;
69         /**
70          * The ReadWindows used to do the PixyCam LEGO
71          protocol SignatureQuery.
72          */
73         private I2cDeviceSynch.ReadWindow[]
74         extnededSignatureQueryReadWindows;
75
76         public PixyCam(I2cDeviceSynch deviceSynch) {
```

```

69         super(deviceSynch, true, new PixyCamParams());
70
71         this.legoProtocolGeneralQueryReadWindow = new
72             I2cDeviceSynch.ReadWindow(LEGO_QUERY_BASE,
73                 LEGO_QUERY_COUNT, I2cDeviceSynch.ReadMode.ONLY_ONCE);
74         this.legoProtocolSignatureQueryReadWindows = new
75             I2cDeviceSynch.ReadWindow[MAX_SIGNATURE];
76         for (int i = MIN_SIGNATURE; i <= MAX_SIGNATURE; i
77             ++) {
78             this.legoProtocolSignatureQueryReadWindows[i
79                 - 1] = NewLegoProtocolSignatureQueryReadWindow(
80                 LEGO_QUERY_BASE, LEGO_QUERY_SIG_COUNT, i);
81         }
82
83         this.extendedGeneralQueryReadWindow = new
84             I2cDeviceSynch.ReadWindow(EXTENDED_QUERY_BASE,
85                 EXTENDED_QUERY_COUNT, I2cDeviceSynch.ReadMode.ONLY_ONCE);
86         this.extnededSignatureQueryReadWindows = new
87             I2cDeviceSynch.ReadWindow[MAX_SIGNATURE];
88         for (int i = MIN_SIGNATURE; i <= MAX_SIGNATURE; i
89             ++) {
90             this.extnededSignatureQueryReadWindows[i - 1]
91                 = NewLegoProtocolSignatureQueryReadWindow(
92                 EXTENDED_QUERY_BASE, EXTENDED_QUERY_SIG_COUNT, i);
93         }
94
95         private I2cDeviceSynch.ReadWindow
96             NewLegoProtocolSignatureQueryReadWindow(int base, int
97                 count, int signature) {
98             return new I2cDeviceSynch.ReadWindow(base +
99                 signature, count, I2cDeviceSynch.ReadMode.ONLY_ONCE);
100         }
101
102         private byte[] ReadEntireWindow(I2cDeviceSynch.
103             ReadWindow readWindow) {
104             this.deviceClient.setReadWindow(readWindow);
105             return this.deviceClient.read(readWindow.
106                 getRegisterFirst(), readWindow.getRegisterCount());
107         }

```

```
96      }
97
98      /**
99      *
100     * @return a Block object containing details about
101    the location of the largest detected block
102   */
103  public PixyBlock getBiggestBlock() {
104      byte[] buffer = ReadEntireWindow(this.
105      legoProtocolGeneralQueryReadWindow);
106
107      int signature = buffer[1] << 8 | buffer[0];
108
109
110     /**
111      * @param signature is a value between 1 and 7
112      corresponding to the signature trained into the PixyCam.
113      * @return a Block object containing details about
114      the location of the largest detected block for the
115      specified signature.
116      */
117      public PixyBlock getBiggestBlock(int signature) {
118          if (signature < MIN_SIGNATURE || signature >
119              MAX_SIGNATURE) {
120              throw new IllegalArgumentException("signature
121              must be between 1 and 7");
122          }
123
124          byte[] buffer = ReadEntireWindow(this.
125          legoProtocolSignatureQueryReadWindows[signature - 1]);
126
127          return new PixyBlock(signature, buffer[0], buffer
128          [1], buffer[2], buffer[3], buffer[4]);
129      }
130
131      public PixyBlockList getBiggestBlocks() {
132          byte[] buffer = ReadEntireWindow(this.
133          extendedGeneralQueryReadWindow);
134
135          PixyBlockList list = new PixyBlockList(buffer[0])
136          ;
137
138      }
```

```

129         for (int i = 1; i < EXTENDED_QUERY_COUNT; i +=
130             EXTENDED_QUERY_BLOCK_SIZE) {
131             int signature = TypeConversion.
132                 unsignedByteToInt(buffer[i]);
133             }
134
135             return list;
136         }
137
138     public PixyBlockList getBiggestBlocks(int signature)
139     {
140         if (signature < MIN_SIGNATURE || signature >
141             MAX_SIGNATURE) {
142             throw new IllegalArgumentException("signature
143             must be between 1 and 7");
144         }
145
146         byte[] buffer = ReadEntireWindow(this.
147             extnededSignatureQueryReadWindows[signature - 1]);
148
149         PixyBlockList list = new PixyBlockList(buffer[0])
150 ;
151
152         for (int i = 1; i < EXTENDED_QUERY_SIG_COUNT; i
153             += EXTENDED_QUERY_SIG_BLOCK_SIZE) {
154             list.add(new PixyBlock(signature, -1, buffer[
155                 i], buffer[i + 1], buffer[i + 2], buffer[i + 3]));
156         }
157
158         return list;
159     }
160
161     @Override
162     protected boolean doInitialize() {
163         return true;
164     }
165
166     @Override
167     protected boolean internalInitialize(@NonNull
168         PixyCamParams pixyCamParams) {
169         this.parameters = pixyCamParams.clone();
170         deviceClient.setI2cAddress(pixyCamParams.i2cAddr)

```

```
163 ;
164
165     return true;
166 }
167
168 @Override
169 public Manufacturer getManufacturer() {
170     return Manufacturer.Other;
171 }
172
173 @Override
174 public String getDeviceName() {
175     return "PixyCam";
176 }
177 }
178
```

```

1 package org.firstinspires.ftc.teamcode.Pixy;
2
3 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
4 import com.qualcomm.robotcore.eventloop.opmode.
5     LinearOpMode;
6 import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
7 import com.qualcomm.robotcore.hardware.I2cAddr;
8 import com.qualcomm.robotcore.hardware.I2cDevice;
9 import com.qualcomm.robotcore.hardware.I2cDeviceSynch;
10 import com.qualcomm.robotcore.hardware.
11     I2cDeviceSynchDevice;
12 import com.qualcomm.robotcore.hardware.I2cDeviceSynchImpl;
13 import com.qualcomm.robotcore.hardware.Servo;
14
15 /**
16 * Created by Sarthak on 7/3/2017.
17 */
18
19 @TeleOp(name="Test Pixy Cam", group = "Pixy Cam")
20 @Disabled
21 public class pixyTest extends LinearOpMode {
22     I2cDevice pixyCam;
23     I2cDeviceSynchImpl pixyCamReader;
24     I2cAddr pixyCamAddress = I2cAddr.create8bit(0x01);
25     double x, y, width, height, numObjs;
26     byte[] pixyData;
27
28     @Override
29     public void runOpMode() throws InterruptedException {
30         pixyCam = hardwareMap.i2cDevice.get("pixy");
31         pixyCamReader = new I2cDeviceSynchImpl(pixyCam,
32             pixyCamAddress, false);
33         pixyCamReader.engage();
34
35         while(opModeIsActive()) {
36             pixyCamReader.engage();
37             pixyData = pixyCamReader.read(0x51, 5);
38
39             x = pixyData[1];
40             y = pixyData[2];
41             numObjs = pixyData[0];
42
43             telemetry.addData("0", pixyData[0]);
44
45         }
46     }
47 }
```

```
43         telemetry.addData("1", pixyData[1]);
44         telemetry.addData("2", pixyData[2]);
45         telemetry.addData("3", pixyData[3]);
46         telemetry.addData("4", pixyData[4]);
47         telemetry.addData("Length", pixyData.length);
48         telemetry.update();
49         pixyCam.readI2cCacheFromController();
50     }
51
52 }
53
54 }
```

```
1 package org.firstinspires.ftc.teamcode.Pixy;
2
3 import com.qualcomm.robotcore.util.TypeConversion;
4
5 /**
6  * Block describes the signature, location, and size of a
7  * detected block.
8 */
9 public class PixyBlock {
10     /**
11      * A number from 1 through 7 corresponding to the
12      * color trained into the PixyCam,
13      * or a sequence of octal digits corresponding to the
14      * multiple colors of a color code.
15     */
16     public final int signature;
17
18     /**
19      * The x, y location of the center of the detected
20      * block.
21      * x is in the range (0, 255)
22      * 0 is the left side of the field of view and 255 is
23      * the right.
24      * y is in the range (0, 199)
25      * 0 is the top of the field of view and 199 is the
26      * bottom.
27     */
28     public final int x, y;
29
30     /**
31      * The size of the detected block.
32      * width or height of zero indicates no block detected
33      *
34      * maximum width is 255.
35      * maximum height is 199.
36     */
37     public final int width, height;
38
39     /**
40      * The count of blocks detected for the given
41      * signature
42      * Will be -1 if this is a block from the general
43      * query
44     */
45     public final int blockCount;
```

```
37
38     public PixyBlock(int signature, byte blockCount, byte
39         x, byte y, byte width, byte height) {
40         this(signature, TypeConversion.unsignedByteToInt(
41             blockCount), x, y, width, height);
42     }
43
44     public PixyBlock(int signature, int blockCount, byte x
45         , byte y, byte width, byte height) {
46         this.signature = signature;
47         this.x = TypeConversion.unsignedByteToInt(x);
48         this.y = TypeConversion.unsignedByteToInt(y);
49         this.width = TypeConversion.unsignedByteToInt(
50             width);
51         this.height = TypeConversion.unsignedByteToInt(
52             height);
53         this.blockCount = blockCount;
54     }
55
56     @Override
57     public String toString() {
58         return String.format("x: %d, y: %d, w: %d, h: %d
59         cnt: %d", this.x, this.y, this.width, this.height, this.
60         blockCount);
61     }
62 }
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28  */
29
30 package org.firstinspires.ftc.teamcode.Pixy;
31
32 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
33 import com.qualcomm.robotcore.eventloop.opmode.OpMode;
34 import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
35 import com.qualcomm.robotcore.hardware.I2cAddr;
36 import com.qualcomm.robotcore.hardware.I2cDeviceSynch;
37 import com.qualcomm.robotcore.util.ElapsedTime;
38
39 /**
40  * Demonstrates empty OpMode
41 */
42 @TeleOp(name = "Pixy simple test", group = "pixy")
43 @Disabled
44 public class PixyOpMode extends OpMode {
45
46     private ElapsedTime runtime = new ElapsedTime();
47
48     I2cDeviceSynch pixy;
49
50     @Override
51     public void init() {
52         //setting up Pixy to the hardware map
53         pixy = hardwareMap.i2cDeviceSynch.get("pixy");
54
55         //setting Pixy's I2C Address
56         pixy.setI2cAddress(I2cAddr.create7bit(0x54));
57
58         //setting Pixy's read window. You'll want these
59         //exact parameters, and you can reference the
60         //SDK Documentation to learn more
61         I2cDeviceSynch.ReadWindow readWindow = new
62         I2cDeviceSynch.ReadWindow (1, 26, I2cDeviceSynch.ReadMode.
63         REPEAT);
64         pixy.setReadWindow(readWindow);
65
66         //required to "turn on" the device
67         pixy.engage();
68         telemetry.addData("Status", "Initialized");
69     }
70
71
72     @Override
```

```
70     public void start() {
71         runtime.reset();
72     }
73
74
75     @Override
76     public void loop() {
77         telemetry.addData("Status", "Run Time: " +
    runtime.toString());
78
79         telemetry.addData("Byte 0", pixy.read8(0));
80         telemetry.addData("Byte 1", pixy.read8(1));
81         telemetry.addData("Byte 2", pixy.read8(2));
82         telemetry.addData("Byte 3", pixy.read8(3));
83         telemetry.addData("Byte 4", pixy.read8(4));
84         telemetry.addData("Byte 5", pixy.read8(5));
85         telemetry.addData("Byte 6", pixy.read8(6));
86         telemetry.addData("Byte 7", pixy.read8(7));
87         telemetry.addData("Byte 8", pixy.read8(8));
88         telemetry.addData("Byte 9", pixy.read8(9));
89         telemetry.addData("Byte 10", pixy.read8(10));
90         telemetry.addData("Byte 11", pixy.read8(11));
91         telemetry.addData("Byte 12", pixy.read8(12));
92         telemetry.addData("Byte 13", pixy.read8(13));
93         telemetry.update();
94     }
95 }
```

```
1 package org.firstinspires.ftc.teamcode.Pixy;
2
3 import com.qualcomm.robotcore.util.TypeConversion;
4
5 import java.util.Vector;
6
7 public class PixyBlockList extends Vector<PixyBlock> {
8     public int totalCount;
9
10    public PixyBlockList(byte totalCount) {
11        this.totalCount = TypeConversion.unsignedByteToInt
12        (totalCount);
13    }
14 }
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

File - C:\Users\Black Tigers #11192\Desktop\robot-code-3.0\robot code\TeamCode\src\main\java\org\firstinspires\ftc\teamcode

```
28  */
29
30 package org.firstinspires.ftc.teamcode.Pixy;
31
32 import android.os.Environment;
33
34 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
35 import com.qualcomm.robotcore.eventloop.opmode.OpMode;
36 import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
37 import com.qualcomm.robotcore.util.ElapsedTime;
38
39 import org.firstinspires.ftc.teamcode.Pixy.PixyBlock;
40 import org.firstinspires.ftc.teamcode.Pixy.PixyBlockList;
41 import org.firstinspires.ftc.teamcode.Pixy.PixyCam;
42
43 import java.io.File;
44 import java.io.FileNotFoundException;
45 import java.io.PrintWriter;
46 import java.io.UnsupportedEncodingException;
47 /**
48 * Demonstrates empty OpMode
49 */
50 @TeleOp(name = "PixyBetterOpMode", group = "pixy")
51 @Disabled
52
53 public class PixyBetterOpMode extends OpMode {
54
55     PixyCam pixyCam;
56     PixyBlockList blocks1;
57     ElapsedTime elapsedTime = new ElapsedTime();
58     ElapsedTime elapsedTime2 = new ElapsedTime();
59
60     PrintWriter file;
61
62     /*
63      * Code to run when the op mode is first enabled goes here
64      * @see com.qualcomm.robotcore.eventloop.opmode.OpMode
65      #start()
66      */
67     @Override
68     public void init() {
69         pixyCam = hardwareMap.get(PixyCam.class, "pixycam");
70     }
71     try {
72         file = new PrintWriter(new File(Environment.getExternalStorageDirectory().getAbsolutePath() + "/pixycam.log"));
73     } catch (FileNotFoundException e) {
74         e.printStackTrace();
75     }
76     file.println("OpMode Init");
77
78     #opModePeriodic()
79     /*
80      * Code to run every 10ms while the op mode is enabled goes here
81      */
82
83     #start()
84     /*
85      * Code to run when the op mode is disabled goes here
86      */
87
88     #stop()
89     /*
90      * Code to run when the op mode is destroyed goes here
91      */
92 }
```

```
70         file = new PrintWriter("/sdcard/pixyResults.
    txt", "UTF-8");
71     } catch (FileNotFoundException e) {
72         e.printStackTrace();
73     } catch (UnsupportedEncodingException e) {
74         e.printStackTrace();
75     }
76 }
77
78 /**
79 * This method will be called repeatedly in a loop
80 * @see com.qualcomm.robotcore.eventloop.opmode.
OpMode#loop()
81 */
82 @Override
83 public void loop() {
84     // Update every tenth of a second.
85     if (elapsedTime.milliseconds() > 100) {
86         elapsedTime.reset();
87         blocks1 = pixyCam.getBiggestBlocks(1);
88         telemetry.addData("Counts", "%d", blocks1.
totalCount);
89         file.println("-----");
90         file.format("Elapsed: %s Counts: %d\n",
elapsedTime2.toString(), blocks1.totalCount);
91         for (int i = 0; i < blocks1.size(); i++) {
92             PixyBlock block = blocks1.get(i);
93             if (!block.isEmpty()) {
94                 telemetry.addData("Block 1[" + i +
"]", block.toString());
95             }
96         }
97         telemetry.update();
98     }
99 }
100
101 @Override
102 public void stop() {
103     file.close();
104 }
105 }
```

```
1 package org.firstinspires.ftc.teamcode.Pixy;
2
3 import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
4 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
5 import com.qualcomm.robotcore.eventloop.opmode.
6     LinearOpMode;
7 import com.qualcomm.robotcore.hardware.I2cDeviceSynch;
8
9 @Autonomous(name = "ConceptSensorPixyCam")
10 @Disabled
11 public class ConceptSensorPixyCam extends LinearOpMode {
12     I2cDeviceSynch pixyCam;
13
14     double x, y, width, height, numObjects;
15
16     byte[] pixyData;
17
18     @Override
19     public void runOpMode() throws InterruptedException {
20
21         pixyCam = hardwareMap.i2cDeviceSynch.get("pixy");
22
23         waitForStart();
24         pixyCam.engage();
25         while(opModeIsActive()){
26             telemetry.addLine(pixyCam.getReadWindow().
27                 toString());
28             pixyData = pixyCam.read(0x51, 5);
29
30             x = pixyData[1];
31             y = pixyData[2];
32             width = pixyData[3];
33             height = pixyData[4];
34             numObjects = pixyData[0];
35
36             telemetry.addData("0", 0xff&pixyData[0]);
37             telemetry.addData("1", 0xff&pixyData[1]);
38             telemetry.addData("2", 0xff&pixyData[2]);
39             telemetry.addData("3", 0xff&pixyData[3]);
40             telemetry.addData("4", 0xff&pixyData[4]);
41             telemetry.addData("Length", pixyData.length);
42             telemetry.update();
43             sleep (500);
44         }
45     }
46 }
```

```
44
45      }
46  }
47
```

```
1 package org.firstinspires.ftc.teamcode.Util;
2
3 import android.graphics.Path;
4
5 import com.qualcomm.robotcore.eventloop.opmode.OpMode;
6 import com.qualcomm.robotcore.hardware.ColorSensor;
7 import com.qualcomm.robotcore.hardware.DcMotor;
8 import com.qualcomm.robotcore.hardware.Servo;
9 import com.qualcomm.robotcore.wifi.NetworkConnection;
10 import com.qualcomm.robotcore.wifi.WifiAssistant;
11 import com.qualcomm.robotcore.wifi.WifiDirectAssistant;
12
13 import org.firstinspires.ftc.robotcontroller.internal.
14     FtcRobotControllerActivity;
14 import org.firstinspires.ftc.robotcore.internal.network.
15     WifiDirectAgent;
15 import org.firstinspires.ftc.robotcore.internal.network.
16     WifiDirectChannelManager;
16 import org.firstinspires.ftc.robotcore.internal.network.
17     WifiState;
17 import org.firstinspires.ftc.teamcode.Prototyping.
18     SensorColor;
18
19 import java.io.File;
20 import java.io.FileOutputStream;
21 import java.io.IOException;
22 import java.io.OutputStream;
23 import java.io.PrintStream;
24 import java.sql.Time;
25 import java.util.Calendar;
26 import java.util.Date;
27 import java.util.Timer;
28 import java.util.Date;
29 import java.text.SimpleDateFormat;
30
31 public class LogCreator {
32     private File log;
33     private String PATH = "/sdcard/FIRST/Log-";
34     private FileOutputStream outputStream;
35     private PrintStream printStream;
36     private String fileName;
37     private OpMode opMode;
38
39     public LogCreator(String fileName) {
40         this.fileName = fileName;
```

```
41      }
42
43      public void init(OpMode opMode) {
44          try {
45              SimpleDateFormat formatter = new
46                  SimpleDateFormat("_dd.MM.yyyy_HH:mm:ss");
47              Date date = new Date();
48              log = new File(PATH + fileName + formatter.
49                  format(date) + ".txt");
50              outputStream = new FileOutputStream(log);
51              printStream = new PrintStream(outputStream);
52              printStream.write(" Time, Object, Value,
53                  Comments ".getBytes());
54              newLine();
55              this.opMode = opMode;
56          } catch (IOException E) {
57          }
58      }
59
60      public void writeLog(String object, double value,
61          String comments) {
62          try {
63              printStream.write((opMode.getRuntime() + "," +
64                  object + "," + value + "," + comments).getBytes());
65              newLine();
66          } catch (IOException E) { }
67      }
68  }
69
```

```

1 package org.firstinspires.ftc.teamcode.Util;
2
3 import com.qualcomm.robotcore.eventloop.opmode.OpMode;
4 import com.qualcomm.robotcore.util.Range;
5
6 public class PIDController {
7
8     protected double setpoint, tolerance, currentError;
9     protected double kP, kI, kD;
10    protected OpMode opMode;
11
12    protected double prevError, sumError, prevTime;
13    protected boolean running = false;
14
15    /**
16     * Instantiate a new PIDController.
17     * @param kP
18     * @param kI
19     * @param kD
20     * @param tolerance
21     */
22    public PIDController(double kP, double kI, double kD,
23        double tolerance, OpMode opmode) {
24        this.kP = kP;
25        this.kI = kI;
26        this.kD = kD;
27        this.tolerance = tolerance;
28        this.opMode = opmode;
29    }
30
31    /**
32     * Get the output from the PID controller.
33     * @param currentPosition the current position of the
34     * mechanism
35     * @return the output to be sent to the motor(s)
36     */
37    public double getOutput(double currentPosition) {
38        currentError = setpoint - currentPosition;
39        sumError += currentError;
40
41        double currentTime = opMode.getRuntime();
42        double rateOfChange = (currentError - prevError) /
43            (currentTime - prevTime);
44
45        prevError = currentError;
46    }
47
48    /**
49     * Set the setpoint for the PID controller.
50     * @param setpoint the target value for the mechanism
51     */
52    public void setSetpoint(double setpoint) {
53        this.setpoint = setpoint;
54    }
55
56    /**
57     * Set the tolerance for the PID controller.
58     * @param tolerance the acceptable range of error
59     */
60    public void setTolerance(double tolerance) {
61        this.tolerance = tolerance;
62    }
63
64    /**
65     * Set the OpMode for the PID controller.
66     * @param opMode the OpMode object
67     */
68    public void setOpMode(OpMode opMode) {
69        this.opMode = opMode;
70    }
71
72    /**
73     * Get the current error from the PID controller.
74     * @return the current error value
75     */
76    public double getCurrentError() {
77        return currentError;
78    }
79
80    /**
81     * Get the sum of errors from the PID controller.
82     * @return the sum of errors value
83     */
84    public double getSumError() {
85        return sumError;
86    }
87
88    /**
89     * Get the previous error from the PID controller.
90     * @return the previous error value
91     */
92    public double getPrevError() {
93        return prevError;
94    }
95
96    /**
97     * Get the previous time from the PID controller.
98     * @return the previous time value
99     */
100    public double getPrevTime() {
101        return prevTime;
102    }
103
104    /**
105     * Get the current time from the PID controller.
106     * @return the current time value
107     */
108    public double getCurrentTime() {
109        return opMode.getRuntime();
110    }
111
112    /**
113     * Get the rate of change from the PID controller.
114     * @return the rate of change value
115     */
116    public double getRateOfChange() {
117        return rateOfChange;
118    }
119
120    /**
121     * Get the tolerance from the PID controller.
122     * @return the tolerance value
123     */
124    public double getTolerance() {
125        return tolerance;
126    }
127
128    /**
129     * Get the current opMode from the PID controller.
130     * @return the current opMode value
131     */
132    public OpMode getOpMode() {
133        return opMode;
134    }
135
136    /**
137     * Get the current kP from the PID controller.
138     * @return the current kP value
139     */
140    public double getkP() {
141        return kP;
142    }
143
144    /**
145     * Get the current kI from the PID controller.
146     * @return the current kI value
147     */
148    public double getkI() {
149        return kI;
150    }
151
152    /**
153     * Get the current kD from the PID controller.
154     * @return the current kD value
155     */
156    public double getkD() {
157        return kD;
158    }
159
160    /**
161     * Set the current kP for the PID controller.
162     * @param kP the new kP value
163     */
164    public void setkP(double kP) {
165        this.kP = kP;
166    }
167
168    /**
169     * Set the current kI for the PID controller.
170     * @param kI the new kI value
171     */
172    public void setkI(double kI) {
173        this.kI = kI;
174    }
175
176    /**
177     * Set the current kD for the PID controller.
178     * @param kD the new kD value
179     */
180    public void setkD(double kD) {
181        this.kD = kD;
182    }
183
184    /**
185     * Set the current tolerance for the PID controller.
186     * @param tolerance the new tolerance value
187     */
188    public void setTolerance(double tolerance) {
189        this.tolerance = tolerance;
190    }
191
192    /**
193     * Set the current opMode for the PID controller.
194     * @param opMode the new opMode value
195     */
196    public void setOpMode(OpMode opMode) {
197        this.opMode = opMode;
198    }
199
200    /**
201     * Set the current setpoint for the PID controller.
202     * @param setpoint the new setpoint value
203     */
204    public void setSetpoint(double setpoint) {
205        this.setpoint = setpoint;
206    }
207
208    /**
209     * Set the current sum of errors for the PID controller.
210     * @param sumError the new sum of errors value
211     */
212    public void setSumError(double sumError) {
213        this.sumError = sumError;
214    }
215
216    /**
217     * Set the current previous error for the PID controller.
218     * @param prevError the new previous error value
219     */
220    public void setPrevError(double prevError) {
221        this.prevError = prevError;
222    }
223
224    /**
225     * Set the current previous time for the PID controller.
226     * @param prevTime the new previous time value
227     */
228    public void setPrevTime(double prevTime) {
229        this.prevTime = prevTime;
230    }
231
232    /**
233     * Set the current current time for the PID controller.
234     * @param currentTime the new current time value
235     */
236    public void setCurrentTime(double currentTime) {
237        this.currentTime = currentTime;
238    }
239
240    /**
241     * Set the current rate of change for the PID controller.
242     * @param rateOfChange the new rate of change value
243     */
244    public void setRateOfChange(double rateOfChange) {
245        this.rateOfChange = rateOfChange;
246    }
247
248    /**
249     * Set the current current error for the PID controller.
250     * @param currentError the new current error value
251     */
252    public void setCurrentError(double currentError) {
253        this.currentError = currentError;
254    }
255
256    /**
257     * Set the current previous error for the PID controller.
258     * @param prevError the new previous error value
259     */
260    public void setPrevError(double prevError) {
261        this.prevError = prevError;
262    }
263
264    /**
265     * Set the current current time for the PID controller.
266     * @param currentTime the new current time value
267     */
268    public void setCurrentTime(double currentTime) {
269        this.currentTime = currentTime;
270    }
271
272    /**
273     * Set the current rate of change for the PID controller.
274     * @param rateOfChange the new rate of change value
275     */
276    public void setRateOfChange(double rateOfChange) {
277        this.rateOfChange = rateOfChange;
278    }
279
280    /**
281     * Set the current current error for the PID controller.
282     * @param currentError the new current error value
283     */
284    public void setCurrentError(double currentError) {
285        this.currentError = currentError;
286    }
287
288    /**
289     * Set the current previous error for the PID controller.
290     * @param prevError the new previous error value
291     */
292    public void setPrevError(double prevError) {
293        this.prevError = prevError;
294    }
295
296    /**
297     * Set the current current time for the PID controller.
298     * @param currentTime the new current time value
299     */
300    public void setCurrentTime(double currentTime) {
301        this.currentTime = currentTime;
302    }
303
304    /**
305     * Set the current rate of change for the PID controller.
306     * @param rateOfChange the new rate of change value
307     */
308    public void setRateOfChange(double rateOfChange) {
309        this.rateOfChange = rateOfChange;
310    }
311
312    /**
313     * Set the current current error for the PID controller.
314     * @param currentError the new current error value
315     */
316    public void setCurrentError(double currentError) {
317        this.currentError = currentError;
318    }
319
320    /**
321     * Set the current previous error for the PID controller.
322     * @param prevError the new previous error value
323     */
324    public void setPrevError(double prevError) {
325        this.prevError = prevError;
326    }
327
328    /**
329     * Set the current current time for the PID controller.
330     * @param currentTime the new current time value
331     */
332    public void setCurrentTime(double currentTime) {
333        this.currentTime = currentTime;
334    }
335
336    /**
337     * Set the current rate of change for the PID controller.
338     * @param rateOfChange the new rate of change value
339     */
340    public void setRateOfChange(double rateOfChange) {
341        this.rateOfChange = rateOfChange;
342    }
343
344    /**
345     * Set the current current error for the PID controller.
346     * @param currentError the new current error value
347     */
348    public void setCurrentError(double currentError) {
349        this.currentError = currentError;
350    }
351
352    /**
353     * Set the current previous error for the PID controller.
354     * @param prevError the new previous error value
355     */
356    public void setPrevError(double prevError) {
357        this.prevError = prevError;
358    }
359
360    /**
361     * Set the current current time for the PID controller.
362     * @param currentTime the new current time value
363     */
364    public void setCurrentTime(double currentTime) {
365        this.currentTime = currentTime;
366    }
367
368    /**
369     * Set the current rate of change for the PID controller.
370     * @param rateOfChange the new rate of change value
371     */
372    public void setRateOfChange(double rateOfChange) {
373        this.rateOfChange = rateOfChange;
374    }
375
376    /**
377     * Set the current current error for the PID controller.
378     * @param currentError the new current error value
379     */
380    public void setCurrentError(double currentError) {
381        this.currentError = currentError;
382    }
383
384    /**
385     * Set the current previous error for the PID controller.
386     * @param prevError the new previous error value
387     */
388    public void setPrevError(double prevError) {
389        this.prevError = prevError;
390    }
391
392    /**
393     * Set the current current time for the PID controller.
394     * @param currentTime the new current time value
395     */
396    public void setCurrentTime(double currentTime) {
397        this.currentTime = currentTime;
398    }
399
400    /**
401     * Set the current rate of change for the PID controller.
402     * @param rateOfChange the new rate of change value
403     */
404    public void setRateOfChange(double rateOfChange) {
405        this.rateOfChange = rateOfChange;
406    }
407
408    /**
409     * Set the current current error for the PID controller.
410     * @param currentError the new current error value
411     */
412    public void setCurrentError(double currentError) {
413        this.currentError = currentError;
414    }
415
416    /**
417     * Set the current previous error for the PID controller.
418     * @param prevError the new previous error value
419     */
420    public void setPrevError(double prevError) {
421        this.prevError = prevError;
422    }
423
424    /**
425     * Set the current current time for the PID controller.
426     * @param currentTime the new current time value
427     */
428    public void setCurrentTime(double currentTime) {
429        this.currentTime = currentTime;
430    }
431
432    /**
433     * Set the current rate of change for the PID controller.
434     * @param rateOfChange the new rate of change value
435     */
436    public void setRateOfChange(double rateOfChange) {
437        this.rateOfChange = rateOfChange;
438    }
439
440    /**
441     * Set the current current error for the PID controller.
442     * @param currentError the new current error value
443     */
444    public void setCurrentError(double currentError) {
445        this.currentError = currentError;
446    }
447
448    /**
449     * Set the current previous error for the PID controller.
450     * @param prevError the new previous error value
451     */
452    public void setPrevError(double prevError) {
453        this.prevError = prevError;
454    }
455
456    /**
457     * Set the current current time for the PID controller.
458     * @param currentTime the new current time value
459     */
460    public void setCurrentTime(double currentTime) {
461        this.currentTime = currentTime;
462    }
463
464    /**
465     * Set the current rate of change for the PID controller.
466     * @param rateOfChange the new rate of change value
467     */
468    public void setRateOfChange(double rateOfChange) {
469        this.rateOfChange = rateOfChange;
470    }
471
472    /**
473     * Set the current current error for the PID controller.
474     * @param currentError the new current error value
475     */
476    public void setCurrentError(double currentError) {
477        this.currentError = currentError;
478    }
479
480    /**
481     * Set the current previous error for the PID controller.
482     * @param prevError the new previous error value
483     */
484    public void setPrevError(double prevError) {
485        this.prevError = prevError;
486    }
487
488    /**
489     * Set the current current time for the PID controller.
490     * @param currentTime the new current time value
491     */
492    public void setCurrentTime(double currentTime) {
493        this.currentTime = currentTime;
494    }
495
496    /**
497     * Set the current rate of change for the PID controller.
498     * @param rateOfChange the new rate of change value
499     */
500    public void setRateOfChange(double rateOfChange) {
501        this.rateOfChange = rateOfChange;
502    }
503
504    /**
505     * Set the current current error for the PID controller.
506     * @param currentError the new current error value
507     */
508    public void setCurrentError(double currentError) {
509        this.currentError = currentError;
510    }
511
512    /**
513     * Set the current previous error for the PID controller.
514     * @param prevError the new previous error value
515     */
516    public void setPrevError(double prevError) {
517        this.prevError = prevError;
518    }
519
520    /**
521     * Set the current current time for the PID controller.
522     * @param currentTime the new current time value
523     */
524    public void setCurrentTime(double currentTime) {
525        this.currentTime = currentTime;
526    }
527
528    /**
529     * Set the current rate of change for the PID controller.
530     * @param rateOfChange the new rate of change value
531     */
532    public void setRateOfChange(double rateOfChange) {
533        this.rateOfChange = rateOfChange;
534    }
535
536    /**
537     * Set the current current error for the PID controller.
538     * @param currentError the new current error value
539     */
540    public void setCurrentError(double currentError) {
541        this.currentError = currentError;
542    }
543
544    /**
545     * Set the current previous error for the PID controller.
546     * @param prevError the new previous error value
547     */
548    public void setPrevError(double prevError) {
549        this.prevError = prevError;
550    }
551
552    /**
553     * Set the current current time for the PID controller.
554     * @param currentTime the new current time value
555     */
556    public void setCurrentTime(double currentTime) {
557        this.currentTime = currentTime;
558    }
559
560    /**
561     * Set the current rate of change for the PID controller.
562     * @param rateOfChange the new rate of change value
563     */
564    public void setRateOfChange(double rateOfChange) {
565        this.rateOfChange = rateOfChange;
566    }
567
568    /**
569     * Set the current current error for the PID controller.
570     * @param currentError the new current error value
571     */
572    public void setCurrentError(double currentError) {
573        this.currentError = currentError;
574    }
575
576    /**
577     * Set the current previous error for the PID controller.
578     * @param prevError the new previous error value
579     */
580    public void setPrevError(double prevError) {
581        this.prevError = prevError;
582    }
583
584    /**
585     * Set the current current time for the PID controller.
586     * @param currentTime the new current time value
587     */
588    public void setCurrentTime(double currentTime) {
589        this.currentTime = currentTime;
590    }
591
592    /**
593     * Set the current rate of change for the PID controller.
594     * @param rateOfChange the new rate of change value
595     */
596    public void setRateOfChange(double rateOfChange) {
597        this.rateOfChange = rateOfChange;
598    }
599
600    /**
601     * Set the current current error for the PID controller.
602     * @param currentError the new current error value
603     */
604    public void setCurrentError(double currentError) {
605        this.currentError = currentError;
606    }
607
608    /**
609     * Set the current previous error for the PID controller.
610     * @param prevError the new previous error value
611     */
612    public void setPrevError(double prevError) {
613        this.prevError = prevError;
614    }
615
616    /**
617     * Set the current current time for the PID controller.
618     * @param currentTime the new current time value
619     */
620    public void setCurrentTime(double currentTime) {
621        this.currentTime = currentTime;
622    }
623
624    /**
625     * Set the current rate of change for the PID controller.
626     * @param rateOfChange the new rate of change value
627     */
628    public void setRateOfChange(double rateOfChange) {
629        this.rateOfChange = rateOfChange;
630    }
631
632    /**
633     * Set the current current error for the PID controller.
634     * @param currentError the new current error value
635     */
636    public void setCurrentError(double currentError) {
637        this.currentError = currentError;
638    }
639
640    /**
641     * Set the current previous error for the PID controller.
642     * @param prevError the new previous error value
643     */
644    public void setPrevError(double prevError) {
645        this.prevError = prevError;
646    }
647
648    /**
649     * Set the current current time for the PID controller.
650     * @param currentTime the new current time value
651     */
652    public void setCurrentTime(double currentTime) {
653        this.currentTime = currentTime;
654    }
655
656    /**
657     * Set the current rate of change for the PID controller.
658     * @param rateOfChange the new rate of change value
659     */
660    public void setRateOfChange(double rateOfChange) {
661        this.rateOfChange = rateOfChange;
662    }
663
664    /**
665     * Set the current current error for the PID controller.
666     * @param currentError the new current error value
667     */
668    public void setCurrentError(double currentError) {
669        this.currentError = currentError;
670    }
671
672    /**
673     * Set the current previous error for the PID controller.
674     * @param prevError the new previous error value
675     */
676    public void setPrevError(double prevError) {
677        this.prevError = prevError;
678    }
679
680    /**
681     * Set the current current time for the PID controller.
682     * @param currentTime the new current time value
683     */
684    public void setCurrentTime(double currentTime) {
685        this.currentTime = currentTime;
686    }
687
688    /**
689     * Set the current rate of change for the PID controller.
690     * @param rateOfChange the new rate of change value
691     */
692    public void setRateOfChange(double rateOfChange) {
693        this.rateOfChange = rateOfChange;
694    }
695
696    /**
697     * Set the current current error for the PID controller.
698     * @param currentError the new current error value
699     */
700    public void setCurrentError(double currentError) {
701        this.currentError = currentError;
702    }
703
704    /**
705     * Set the current previous error for the PID controller.
706     * @param prevError the new previous error value
707     */
708    public void setPrevError(double prevError) {
709        this.prevError = prevError;
710    }
711
712    /**
713     * Set the current current time for the PID controller.
714     * @param currentTime the new current time value
715     */
716    public void setCurrentTime(double currentTime) {
717        this.currentTime = currentTime;
718    }
719
720    /**
721     * Set the current rate of change for the PID controller.
722     * @param rateOfChange the new rate of change value
723     */
724    public void setRateOfChange(double rateOfChange) {
725        this.rateOfChange = rateOfChange;
726    }
727
728    /**
729     * Set the current current error for the PID controller.
730     * @param currentError the new current error value
731     */
732    public void setCurrentError(double currentError) {
733        this.currentError = currentError;
734    }
735
736    /**
737     * Set the current previous error for the PID controller.
738     * @param prevError the new previous error value
739     */
740    public void setPrevError(double prevError) {
741        this.prevError = prevError;
742    }
743
744    /**
745     * Set the current current time for the PID controller.
746     * @param currentTime the new current time value
747     */
748    public void setCurrentTime(double currentTime) {
749        this.currentTime = currentTime;
750    }
751
752    /**
753     * Set the current rate of change for the PID controller.
754     * @param rateOfChange the new rate of change value
755     */
756    public void setRateOfChange(double rateOfChange) {
757        this.rateOfChange = rateOfChange;
758    }
759
760    /**
761     * Set the current current error for the PID controller.
762     * @param currentError the new current error value
763     */
764    public void setCurrentError(double currentError) {
765        this.currentError = currentError;
766    }
767
768    /**
769     * Set the current previous error for the PID controller.
770     * @param prevError the new previous error value
771     */
772    public void setPrevError(double prevError) {
773        this.prevError = prevError;
774    }
775
776    /**
777     * Set the current current time for the PID controller.
778     * @param currentTime the new current time value
779     */
780    public void setCurrentTime(double currentTime) {
781        this.currentTime = currentTime;
782    }
783
784    /**
785     * Set the current rate of change for the PID controller.
786     * @param rateOfChange the new rate of change value
787     */
788    public void setRateOfChange(double rateOfChange) {
789        this.rateOfChange = rateOfChange;
790    }
791
792    /**
793     * Set the current current error for the PID controller.
794     * @param currentError the new current error value
795     */
796    public void setCurrentError(double currentError) {
797        this.currentError = currentError;
798    }
799
800    /**
801     * Set the current previous error for the PID controller.
802     * @param prevError the new previous error value
803     */
804    public void setPrevError(double prevError) {
805        this.prevError = prevError;
806    }
807
808    /**
809     * Set the current current time for the PID controller.
810     * @param currentTime the new current time value
811     */
812    public void setCurrentTime(double currentTime) {
813        this.currentTime = currentTime;
814    }
815
816    /**
817     * Set the current rate of change for the PID controller.
818     * @param rateOfChange the new rate of change value
819     */
820    public void setRateOfChange(double rateOfChange) {
821        this.rateOfChange = rateOfChange;
822    }
823
824    /**
825     * Set the current current error for the PID controller.
826     * @param currentError the new current error value
827     */
828    public void setCurrentError(double currentError) {
829        this.currentError = currentError;
830    }
831
832    /**
833     * Set the current previous error for the PID controller.
834     * @param prevError the new previous error value
835     */
836    public void setPrevError(double prevError) {
837        this.prevError = prevError;
838    }
839
840    /**
841     * Set the current current time for the PID controller.
842     * @param currentTime the new current time value
843     */
844    public void setCurrentTime(double currentTime) {
845        this.currentTime = currentTime;
846    }
847
848    /**
849     * Set the current rate of change for the PID controller.
850     * @param rateOfChange the new rate of change value
851     */
852    public void setRateOfChange(double rateOfChange) {
853        this.rateOfChange = rateOfChange;
854    }
855
856    /**
857     * Set the current current error for the PID controller.
858     * @param currentError the new current error value
859     */
860    public void setCurrentError(double currentError) {
861        this.currentError = currentError;
862    }
863
864    /**
865     * Set the current previous error for the PID controller.
866     * @param prevError the new previous error value
867     */
868    public void setPrevError(double prevError) {
869        this.prevError = prevError;
870    }
871
872    /**
873     * Set the current current time for the PID controller.
874     * @param currentTime the new current time value
875     */
876    public void setCurrentTime(double currentTime) {
877        this.currentTime = currentTime;
878    }
879
880    /**
881     * Set the current rate of change for the PID controller.
882     * @param rateOfChange the new rate of change value
883     */
884    public void setRateOfChange(double rateOfChange) {
885        this.rateOfChange = rateOfChange;
886    }
887
888    /**
889     * Set the current current error for the PID controller.
890     * @param currentError the new current error value
891     */
892    public void setCurrentError(double currentError) {
893        this.currentError = currentError;
894    }
895
896    /**
897     * Set the current previous error for the PID controller.
898     * @param prevError the new previous error value
899     */
900    public void setPrevError(double prevError) {
901        this.prevError = prevError;
902    }
903
904    /**
905     * Set the current current time for the PID controller.
906     * @param currentTime the new current time value
907     */
908    public void setCurrentTime(double currentTime) {
909        this.currentTime = currentTime;
910    }
911
912    /**
913     * Set the current rate of change for the PID controller.
914     * @param rateOfChange the new rate of change value
915     */
916    public void setRateOfChange(double rateOfChange) {
917        this.rateOfChange = rateOfChange;
918    }
919
920    /**
921     * Set the current current error for the PID controller.
922     * @param currentError the new current error value
923     */
924    public void setCurrentError(double currentError) {
925        this.currentError = currentError;
926    }
927
928    /**
929     * Set the current previous error for the PID controller.
930     * @param prevError the new previous error value
931     */
932    public void setPrevError(double prevError) {
933        this.prevError = prevError;
934    }
935
936    /**
937     * Set the current current time for the PID controller.
938     * @param currentTime the new current time value
939     */
940    public void setCurrentTime(double currentTime) {
941        this.currentTime = currentTime;
942    }
943
944    /**
945     * Set the current rate of change for the PID controller.
946     * @param rateOfChange the new rate of change value
947     */
948    public void setRateOfChange(double rateOfChange) {
949        this.rateOfChange = rateOfChange;
950    }
951
952    /**
953     * Set the current current error for the PID controller.
954     * @param currentError the new current error value
955     */
956    public void setCurrentError(double currentError) {
957        this.currentError = currentError;
958    }
959
960    /**
961     * Set the current previous error for the PID controller.
962     * @param prevError the new previous error value
963     */
964    public void setPrevError(double prevError) {
965        this.prevError = prevError;
966    }
967
968    /**
969     * Set the current current time for the PID controller.
970     * @param currentTime the new current time value
971     */
972    public void setCurrentTime(double currentTime) {
973        this.currentTime = currentTime;
974    }
975
976    /**
977     * Set the current rate of change for the PID controller.
978     * @param rateOfChange the new rate of change value
979     */
980    public void setRateOfChange(double rateOfChange) {
981        this.rateOfChange = rateOfChange;
982    }
983
984    /**
985     * Set the current current error for the PID controller.
986     * @param currentError the new current error value
987     */
988    public void setCurrentError(double currentError) {
989        this.currentError = currentError;
990    }
991
992    /**
993     * Set the current previous error for the PID controller.
994     * @param prevError the new previous error value
995     */
996    public void setPrevError(double prevError) {
997        this.prevError = prevError;
998    }
999
1000   /**
1001    * Set the current current time for the PID controller.
1002    * @param currentTime the new current time value
1003    */
1004   public void setCurrentTime(double currentTime) {
1005       this.currentTime = currentTime;
1006   }
1007
1008   /**
1009    * Set the current rate of change for the PID controller.
1010    * @param rateOfChange the new rate of change value
1011    */
1012   public void setRateOfChange(double rateOfChange) {
1013       this.rateOfChange = rateOfChange;
1014   }
1015
1016   /**
1017    * Set the current current error for the PID controller.
1018    * @param currentError the new current error value
1019    */
1020   public void setCurrentError(double currentError) {
1021       this.currentError = currentError;
1022   }
1023
1024   /**
1025    * Set the current previous error for the PID controller.
1026    * @param prevError the new previous error value
1027    */
1028   public void setPrevError(double prevError) {
1029       this.prevError = prevError;
1030   }
1031
1032   /**
1033    * Set the current current time for the PID controller.
1034    * @param currentTime the new current time value
1035    */
1036   public void setCurrentTime(double currentTime) {
1037       this.currentTime = currentTime;
1038   }
1039
1040   /**
1041    * Set the current rate of change for the PID controller.
1042    * @param rateOfChange the new rate of change value
1043    */
1044   public void setRateOfChange(double rateOfChange) {
1045       this.rateOfChange = rateOfChange;
1046   }
1047
1048   /**
1049    * Set the current current error for the PID controller.
1050    * @param currentError the new current error value
1051    */
1052   public void setCurrentError(double currentError) {
1053       this.currentError = currentError;
1054   }
1055
1056   /**
1057    * Set the current previous error for the PID controller.
1058    * @param prevError the new previous error value
1059    */
1060   public void setPrevError(double prevError) {
1061       this.prevError = prevError;
1062   }
1063
1064   /**
1065    * Set the current current time for the PID controller.
1066    * @param currentTime the new current time value
1067    */
1068   public void setCurrentTime(double currentTime) {
1069       this.currentTime = currentTime;
1070   }
1071
1072   /**
1073    * Set the current rate of change for the PID controller.
1074    * @param rateOfChange the new rate of change value
1075    */
1076   public void setRateOfChange(double rateOfChange) {
1077       this.rateOfChange = rateOfChange;
1078   }
1079
1080   /**
1081    * Set the current current error for the PID controller.
1082    * @param currentError the new current error value
1083    */
1084   public void setCurrentError(double currentError) {
1085       this.currentError = currentError;
1086   }
1087
1088   /**
1089    * Set the current previous error for the PID controller.
1090    * @param prevError the new previous error value
1091    */
1092   public void setPrevError(double prevError) {
1093       this.prevError = prevError;
1094   }
1095
1096   /**
1097    * Set the current current time for the PID controller.
1098    * @param currentTime the new current time value
1099    */
1100   public void setCurrentTime(double currentTime) {
1101       this.currentTime = currentTime;
1102   }
1103
1104   /**
1105    * Set the current rate of change for the PID controller.
1106    * @param rateOfChange the new rate of change value
1107    */
1108   public void setRateOfChange(double rateOfChange) {
1109       this.rateOfChange = rateOfChange;
1110   }
1111
1112   /**
1113    * Set the current current error for the PID controller.
1114    * @param currentError the new current error value
1115    */
1116   public void setCurrentError(double currentError) {
1117       this.currentError = currentError;
1118   }
1119
1120   /**
1121    * Set the current previous error for the PID controller.
1122    * @param prevError the new previous error value
1123    */
1124   public void setPrevError(double prevError) {
1125       this.prevError = prevError;
1126   }
1127
1128   /**
1129    * Set the current current time for the PID controller.
1130    * @param currentTime the new current time value
1131    */
1132   public void setCurrentTime(double currentTime) {
1133       this.currentTime = currentTime;
1134   }
1135
1136   /**
1137    * Set the current rate of change for the PID controller.
1138    * @param rateOfChange the new rate of change value
1139    */
1140   public void setRateOfChange(double rateOfChange) {
1141       this.rateOfChange = rateOfChange;
1142   }
1143
1144   /**
1145    * Set the current current error for the PID controller.
1146    * @param currentError the new current error value
1147    */
1148   public void setCurrentError(double currentError) {
1149       this.currentError = currentError;
1150   }
1151
1152   /**
1153    * Set the current previous error for the PID controller.
1154    * @param prevError the new previous error value
1155    */
1156   public void setPrevError(double prevError) {
1157       this.prevError = prevError;
1158   }
1159
1160   /**
1161    * Set the current current time for the PID controller.
1162    * @param currentTime the new current time value
1163    */
1164   public void setCurrentTime(double currentTime) {
1165       this.currentTime = currentTime;
1166   }
1167
1168   /**
1169    * Set the current rate of change for the PID controller.
1170    * @param rateOfChange the new rate of change value
1171    */
1172   public void setRateOfChange(double rateOfChange) {
1173       this.rateOfChange = rateOfChange;
1174   }
1175
1176   /**
1177    * Set the current current error for the PID controller.
1178    * @param currentError the new current error value
1179    */
1180   public void setCurrentError(double currentError) {
1181       this.currentError = currentError;
1182   }
1183
1184   /**
1185    * Set the current previous error for the PID controller.
1186    * @param prevError the new previous error value
1187    */
1188   public void setPrevError(double prevError) {
1189       this.prevError = prevError;
1190   }
1191
1192   /**
1193    * Set the current current time for the PID controller.
1194    * @param currentTime the new current time value
1195   
```

```

43         prevTime = currentTime;
44         return Range.clip(kP * currentError + kI *
45             sumError + kD * rateOfChange, -1.0, 1.0);
46     }
47
48     /**
49      * reset the PIDController
50      * @param setpoint the target of the PIDController
51      * @param startingPos the starting position of the
52      * mechanism
53      */
54     public void reset(double setpoint, double startingPos)
55     {
56         this.setpoint = setpoint;
57         running = true;
58
59         sumError = 0;
60         currentError = setpoint - startingPos;
61         prevError = currentError;
62         prevTime = opMode.getRuntime();
63     }
64
65
66     public void stop() {
67         running = false;
68     }
69
70     /**
71      *
72      * @return true if the current position is within the
73      * given tolerance of the setpoint
74      */
75     public boolean onTarget() {
76         return Math.abs(currentError) < tolerance;
77     }
78
79     /**
80      * Set the P, I and D gains.
81      * @param kP the kP to set
82      * @param kI the kI to set
83      * @param kD the kD to set
84      */
85     public void setPID(double kP, double kI, double kD) {

```

```
84         this.kP = kP;
85         this.kI = kI;
86         this.kD = kD;
87     }
88
89     /**
90      * @return the kP
91      */
92     public double getkP() {
93         return kP;
94     }
95
96     /**
97      * @param kP the kP to set
98      */
99     public void setkP(double kP) {
100        this.kP = kP;
101    }
102
103    /**
104     * @return the kI
105     */
106    public double getkI() {
107        return kI;
108    }
109
110   /**
111    * @param kI the kI to set
112    */
113    public void setkI(double kI) {
114        this.kI = kI;
115    }
116
117   /**
118    * @return the kD
119    */
120    public double getkD() {
121        return kD;
122    }
123
124   /**
125    * @param kD the kD to set
126    */
127    public void setkD(double kD) {
128        this.kD = kD;
```

```
129     }
130
131     /**
132      * @param setpoint the setpoint to set
133      */
134     public void setSetpoint(double setpoint) {
135         this.setpoint = setpoint;
136     }
137
138     public double getCurrentError() {
139         return this.currentError;
140     }
141
142     public void updateError(double currentPos) {
143         currentError = setpoint - currentPos;
144     }
145     public double getSetpoint() {
146         return setpoint;
147     }
148 }
149
```

```
1 package org.firstinspires.ftc.teamcode.Util;  
2  
3 public class GlobalVariebels {  
4  
5     public static int liftPosEndAuto = 0;  
6  
7 }  
8
```

```
1 package org.firstinspires.ftc.teamcode.Util;
2 import android.support.annotation.Nullable;
3
4 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
5 import com.qualcomm.robotcore.eventloop.opmode.
6     LinearOpMode;
7 import com.qualcomm.robotcore.eventloop.opmode.OpMode;
8 import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
9 import com.qualcomm.robotcore.hardware.DcMotor;
10 import com.qualcomm.robotcore.hardware.HardwareMap;
11
12 import org.firstinspires.ftc.robotcore.external.
13     ClassFactory;
14 import org.firstinspires.ftc.robotcore.external.navigation
15     .AngleUnit;
16 import org.firstinspires.ftc.robotcore.external.navigation
17     .VuforiaLocalizer;
18 import org.firstinspires.ftc.robotcore.external.navigation
19     .VuforiaLocalizer.CameraDirection;
20 import org.firstinspires.ftc.robotcore.external.tfod.
21     Recognition;
22 import org.firstinspires.ftc.robotcore.external.tfod.
23     TFOBJECTDetector;
24
25 import java.util.List;
26
27 import javax.net.ssl.HandshakeCompletedEvent;
28
29
30
31 public class GoldRecognition {
32     public enum MineralPos{
33         LEFT,
34         CENTER,
35         RIGHT,
36         UNKNOWN;
37     }
38
39     private OpMode opMode;
40     private HardwareMap hardwareMap;
41     private DcMotor leds;
42     /**
43      * this parameters describe our labels that we want to
44      * find
45      */
46     private static final String TFOD_MODEL_ASSET = "
```

```

37 RoverRuckus.tflite";
38     private static final String LABEL_GOLD_MINERAL = "Gold
Mineral";
39     private static final String LABEL_SILVER_MINERAL = "
Silver Mineral";
40     private final double light = 0.3;
41
42     private static final String VUFORIA_KEY = "AW/F0cP
////AAABmUpR4+
dbt0Negw2nqaCH9Cw2gV4ZxuUmpeJMm7XOTdQVumthQcOeoS9qktHy4EvX
tMAFoh7n5KeMiLMdqtKvd1TrbYNUy3f9ST3TMkH2hFYKB6oJMJPB8oell9
Bst/2XJBz0ycMMCkMSSiwyoqwoUHamAlwfT+
o7VusfYmY7FPvnXuhn8obCeB5x0hhjwjsBuOz2wnxlus6N5y6on0rdc1DO
zC2gI767QLVXAvPyJvMfgtZRGcfzfK0evuSVxrIPCRQBjQYK2s5SsBLZ4s
EO4HelibKK5kg0lgT9PluHSNa8SWX+
4wpJm76dFwlnSL7YElieByOTXxycmOdhWaae5qb11vYYmDiBNFiwfHRDkB
RD";
43
44 /**
45     * @link #vuforia} is the variable we will use to
46     store our instance of the Vuforia
47     * localization engine.
48     */
49     private VuforiaLocalizer vuforia;
50
51 /**
52     * @link #tfod} is the variable we will use to store
53     our instance of the Tensor Flow Object
54     * Detection engine.
55     */
56     public TFOBJECTDetector tfod;
57
58     public GoldRecognition(HardwareMap hardwareMap, OpMode
opMode) {
59         this.hardwareMap = hardwareMap;
60         this.opMode = opMode;
61         this.leds = hardwareMap.get(DcMotor.class, "leds")
62         ;
63         initVuforia();
64         if (ClassFactory.getInstance().
canCreateTFOBJECTDetector()) {
65             initTfod();
66         } else {
67             opMode.telemetry.addData("Sorry!", "This
device is not compatible with TFOD");

```

```
65      }
66  }
67
68  private void initVuforia() {
69      /*
70       * Configure Vuforia by creating a Parameter
71       * object, and passing it to the Vuforia engine.
72      */
73      VuforiaLocalizer.Parameters parameters = new
74          VuforiaLocalizer.Parameters();
75
76      parameters.vuforiaLicenseKey = VUFORIA_KEY;
77      parameters.cameraDirection = CameraDirection.
78          FRONT;
79
80      // Instantiate the Vuforia engine
81      vuforia = ClassFactory.getInstance().
82          createVuforia(parameters);
83
84      /**
85       * Initialize the Tensor Flow Object Detection engine
86       *
87      */
88  private void initTfod() {
89      int tfodMonitorViewId = hardwareMap.appContext.
90          getResources().getIdentifier(
91              "tfodMonitorViewId", "id", hardwareMap.
92          appContext.getPackageName());
93      TFObjectDetector.Parameters tfodParameters = new
94          TFObjectDetector.Parameters(tfodMonitorViewId);
95      tfod = ClassFactory.getInstance().
96          createTFObjectDetector(tfodParameters, vuforia);
97      tfod.loadModelFromAsset(TFOD_MODEL_ASSET,
98          LABEL_GOLD_MINERAL, LABEL_SILVER_MINERAL);
99  }
100 public void stopTfod() {
101     tfod.shutdown();
102 }
103
104 public void turnOnLeds() {
105     leds.setPower(light);
106 }
```

```

99      }
100
101     public void turnOffLeds() {
102         leds.setPower(0);
103     }
104
105     public MineralPos getGoldPos(LogCreator log) {
106         turnOnLeds();
107         if (((LinearOpMode) opMode).opModeIsActive()) {
108             /** Activate Tensor Flow Object Detection. */
109             if (tfod != null) {
110                 tfod.activate();
111             }
112             double time = opMode.getRuntime();
113             while (((LinearOpMode) opMode).opModeIsActive()
114                   && opMode.getRuntime() < time + 5) {
115                 List<Recognition> recognitionList = tfod.
116                 getUpdatedRecognitions();
117                 if (recognitionList != null) {
118                     opMode.telemetry.addData("size",
119                     recognitionList.size());
120                     if (log != null) {
121                         log.writeLog("SamplingSize",
122                         recognitionList.size(), "");
123                         for (Recognition recognition :
124                             recognitionList) {
125                             log.writeLog("SamplingObject"
126                             , recognition.getTop(),
127                             recognition.toString());
128                             if (recognition.getHeight() + ", w: " +
129                             recognition.getWidth());
130                         }
131                     }
132                     if (recognitionList.size() == 2) {
133                         double goldTop = -1;
134                         double silverTop = -1;
135                         for (Recognition recognition :
136                             recognitionList) {
137                             if (recognition.getLabel() ==
138                             "Gold Mineral") {
139                                 goldTop = recognition.
140                                 getTop();
141                             } else if (silverTop == -1) {
142                                 silverTop = recognition.
143                                 getTop();
144                             }
145                         }
146                         if (goldTop > silverTop) {
147                             log.writeLog("Gold Top: " + goldTop);
148                         } else {
149                             log.writeLog("Silver Top: " + silverTop);
150                         }
151                     }
152                 }
153             }
154         }
155     }

```

```
132 } else {  
133     return MineralPos.LEFT;  
134 }  
135 }  
136 opMode.telemetry.addData("goldPos  
", goldTop + ",silverPos", silverTop);  
137 if (goldTop != -1 && silverTop !=  
-1) {  
138     if (goldTop > silverTop) {  
139         return MineralPos.CENTER;  
140     } else return MineralPos.  
RIGHT;  
141 }  
142 }  
143 }  
144 opMode.telemetry.update();  
145 }  
146 }  
147 return MineralPos.UNKNOWN;  
148 }  
149 }  
150 }
```

```

1 package org.firstinspires.ftc.teamcode.Util;
2
3 import com.qualcomm.robotcore.eventloop.opmode.
4 LinearOpMode;
5 import com.qualcomm.robotcore.eventloop.opmode.OpMode;
6 import com.qualcomm.robotcore.util.Range;
7
8 public class TurnPIDController extends PIDController {
9     /**
10      * Instantiate a new PIDController.
11      *
12      * @param kP
13      * @param kI
14      * @param kD
15      * @param tolerance
16      * @param opmode
17      */
18     public TurnPIDController(double kP, double kI, double
19     kD, double tolerance, OpMode opmode) {
20         super(kP, kI, kD, tolerance, opmode);
21     }
22
23     @Override
24     public double getOutput(double currentPosition) {
25         currentError = setpoint - currentPosition;
26         while (currentError > 180 && ((LinearOpMode) opMode
27             ).opModeIsActive()) currentError -= 360;
28         while (currentError <= -179 && ((LinearOpMode)
29             opMode).opModeIsActive()) currentError += 360;
30         sumError += currentError;
31
32         double currentTime = opMode.getRuntime();
33         double rateOfChange = (currentError - prevError) /
34             (currentTime - prevTime);
35
36         prevError = currentError;
37         prevTime = currentTime;
38         return Range.clip(kP * currentError + kI *
39             sumError + kD * rateOfChange, -1.0, 1.0);
40     }
41
42     @Override
43     public void reset(double setpoint, double startingPos)
44     {
45         this.setpoint = setpoint;
46     }
47 }
```

```
39         running = true;
40
41         sumError = 0;
42         currentError = setpoint - startingPos;
43         while (currentError > 180 && ((LinearOpMode) opMode
44             .opModeIsActive())) currentError -= 360;
45         while (currentError <= -179 && ((LinearOpMode)
46             opMode).opModeIsActive()) currentError += 360;
45         prevError = currentError;
46         prevTime = opMode.getRuntime();
47     }
48 }
49
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28  */
29
30 package org.firstinspires.ftc.teamcode.Telemetry;
31
32 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
33 import com.qualcomm.robotcore.eventloop.opmode.OpMode;
34 import com.qualcomm.robotcore.hardware.DcMotor;
35 import com.qualcomm.robotcore.hardware.Gamepad;
36 import com.qualcomm.robotcore.hardware.HardwareMap;
37 import com.qualcomm.robotcore.util.ElapsedTime;
38
39 import org.firstinspires.ftc.teamcode.RobotSystems.Robot;
40 import org.firstinspires.ftc.teamcode.Util.LogCreator;
41
42 /**
43  * Demonstrates empty OpMode
44 */
45 @com.qualcomm.robotcore.eventloop.opmode.TeleOp(name = "TeleOp", group = "TeleOp")
46 public class TeleOp extends OpMode {
47
48     private ElapsedTime runtime = new ElapsedTime();
49     protected Robot robot = new Robot();
50     private LogCreator log = new LogCreator("TeleOp");
51
52     @Override
53     public void init() {
54         log.init(this);
55         robot.init(hardwareMap, this, log);
56         telemetry.addData("Status", "Initialized");
57         telemetry.update();
58     }
59
60     @Override
61     public void init_loop() {
62     }
63
64     @Override
65     public void start() {
66         runtime.reset();
67     }
68
69     @Override
70     public void loop() {
71         robot.telemetry(gamepad1, gamepad2);
```

```
72     telemetry.update();  
73  
74 }  
75 }  
76
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28  */
29
30 package org.firstinspires.ftc.teamcode.Telemetry;
31
32 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
33 import com.qualcomm.robotcore.eventloop.opmode.OpMode;
34 import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
35 import com.qualcomm.robotcore.hardware.DcMotor;
36 import com.qualcomm.robotcore.util.ElapsedTime;
37
38 import org.firstinspires.ftc.teamcode.Util.GoldRecognition;
39 ;
40 /**
41  * Demonstrates empty OpMode
42 */
43 @TeleOp(name = "Led_Test", group = "Concept")
44 @Disabled
45 public class Led_Test extends OpMode {
46
47     private ElapsedTime runtime = new ElapsedTime();
48     private GoldRecognition goldRecognition;
49
50     @Override
51     public void init() {
52         goldRecognition = new GoldRecognition(hardwareMap,
53         this);
54         telemetry.addData("Status", "Initialized");
55     }
56
57     @Override
58     public void init_loop() {
59     }
60
61     @Override
62     public void start() {
63         runtime.reset();
64     }
65
66     @Override
67     public void loop() {
68         telemetry.addData("Status", "Run Time: " + runtime.
69         toString());
69         telemetry.addLine("turnOnLeds: a");
69         telemetry.addLine("turnOffLeds: b");
```

```
70     if (gamepad1.a)
71         goldRecognition.turnOnLeds() ;
72     if(gamepad1.b)
73         goldRecognition.turnOffLeds() ;
74 }
75 }
76
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28 */  
29  
30 package org.firstinspires.ftc.teamcode.Autonomous;  
31  
32 import com.qualcomm.robotcore.eventloop.opmode.Autonomous;  
33 import com.qualcomm.robotcore.eventloop.opmode.Disabled;  
34 import com.qualcomm.robotcore.eventloop.opmode.  
    LinearOpMode;  
35 import com.qualcomm.robotcore.util.ElapsedTime;  
36  
37 import org.firstinspires.ftc.teamcode.RobotSystems.  
    Climbing;  
38 import org.firstinspires.ftc.teamcode.RobotSystems.Drive;  
39 import org.firstinspires.ftc.teamcode.RobotSystems.Robot;  
40  
41 @Autonomous(name = "Land", group = "Tests")  
42 public class Land extends LinearOpMode {  
43  
44     protected Robot robot = new Robot();  
45     private ElapsedTime runtime = new ElapsedTime();  
46  
47     @Override  
48     public void runOpMode() throws InterruptedException {  
49         robot.init(hardwareMap, this);  
50         waitForStart();  
51         robot.climbing.land();  
52     }  
53 }
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28  */
29
30 package org.firstinspires.ftc.teamcode.Autonomous;
31
32 import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
33 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
34 import com.qualcomm.robotcore.eventloop.opmode.
35     LinearOpMode;
36 import com.qualcomm.robotcore.util.ElapsedTime;
37
38 import org.firstinspires.ftc.teamcode.RobotSystems.
39     Climbing;
40 import org.firstinspires.ftc.teamcode.RobotSystems.Drive;
41 import org.firstinspires.ftc.teamcode.RobotSystems.Robot;
42 import org.firstinspires.ftc.teamcode.Util.LogCreator;
43
44 @Autonomous(name = "TurnTest", group = "Tests")
45 public class AutoTest extends LinearOpMode {
46
47     private Robot robot = new Robot();
48     private ElapsedTime runtime = new ElapsedTime();
49
50
51     @Override
52     public void runOpMode() throws InterruptedException {
53         robot.init(hardwareMap, this);
54         waitForStart();
55         robot.drive.turnByGyroAbsolut(100, 5);
56         robot.drive.turnByGyroAbsolut(-170, 5);
57     }
58 }
59 }
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28  */
29
30 package org.firstinspires.ftc.teamcode.Autonomous;
31
32 import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
33 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
34 import com.qualcomm.robotcore.eventloop.opmode.
35     LinearOpMode;
36
37 import com.qualcomm.robotcore.util.ElapsedTime;
38
39 import org.firstinspires.ftc.teamcode.RobotSystems.Drive;
40 import org.firstinspires.ftc.teamcode.RobotSystems.Robot;
41 import org.firstinspires.ftc.teamcode.Util.LogCreator;
42 import java.util.Date;
43 import java.text.SimpleDateFormat;
44
45 @Autonomous(name = "TimeTest", group = "Auto")
46 @Disabled
47 public class TimeTest extends LinearOpMode {
48     SimpleDateFormat formatter = new SimpleDateFormat("_dd.
49     @Override
50     public void runOpMode() throws InterruptedException {
51         waitForStart();
52         while (opModeIsActive()) {
53             Date date = new Date();
54             telemetry.addData("time: ", formatter.format(date));
55             telemetry.update();
56         }
57     }
58 }
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28  */
29
30 package org.firstinspires.ftc.teamcode.Autonomous;
31
32 import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
33 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
34 import com.qualcomm.robotcore.eventloop.opmode.
35 LinearOpMode;
36
37 import com.qualcomm.robotcore.util.ElapsedTime;
38
39 import org.firstinspires.ftc.robotcore.external.tfod.
40 Recognition;
41 import org.firstinspires.ftc.teamcode.Util.GoldRecognition
42 ;
43 import org.firstinspires.ftc.teamcode.RobotSystems.Robot;
44
45 @Autonomous(name = "AutoPhoneCamera", group = "Tests")
46
47 public class AutoPhoneCamera extends LinearOpMode {
48     protected Robot robot = new Robot();
49     private GoldRecognition recognition = null;
50     private ElapsedTime runtime = new ElapsedTime();
51
52     @Override
53     public void runOpMode() throws InterruptedException {
54         recognition = new GoldRecognition(hardwareMap, this);
55         recognition.tfod.activate();
56         waitForStart();
57         recognition.turnOnLeds();
58         while (opModeIsActive()) {
59             GoldRecognition.MineralPos mineralPos = recognition.
60             getGoldPos(null);
61             telemetry.addData(" mineral", mineralPos);
62             telemetry.update();
63         }
64     }
65 }
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```

28  */
29
30 package org.firstinspires.ftc.teamcode.Autonomous.Depot;
31
32 import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
33 import com.qualcomm.robotcore.eventloop.opmode.
34     LinearOpMode;
35
36 import com.qualcomm.robotcore.util.ElapsedTime;
37
38 import org.firstinspires.ftc.teamcode.RobotSystems.
39     Climbing;
40 import org.firstinspires.ftc.teamcode.RobotSystems.Drive;
41 import org.firstinspires.ftc.teamcode.RobotSystems.Robot;
42 import org.firstinspires.ftc.teamcode.Util.GlobalVariebels
43 ;
44 import org.firstinspires.ftc.teamcode.Util.GoldRecognation
45 ;
46 import org.firstinspires.ftc.teamcode.Util.LogCreater;
47
48 /**
49  * doing Landing, Sampling and Team Marker
50  * PTS = 70
51  * Starting from Depot On The Lander
52  * FIRST Autonomous
53  */
54
55 @Autonomous(name = "Depot", group = "Auto")
56 public class Depot extends LinearOpMode {
57
58     protected Robot robot = new Robot();
59     protected ElapsedTime runtime = new ElapsedTime();
60     protected LogCreater log = new LogCreater("auto");
61
62     public void endAuto(GoldRecognation.MineralPos goldPos
63 ) {
64         if (goldPos == GoldRecognation.MineralPos.LEFT) {
65             robot.drive.turnByGyroAbsolut(-45, 10);
66             robot.drive.driveByEncoder(100, 0.5, Drive.
67                 Direction.FORWARD, 3000);
68             robot.climbing.moveAngleAuto(Climbing.Angle.
69                 COLLECT);
70             robot.intake.collect();
71             robot.climbing.moveLiftAuto(Climbing.Height.
72                 PUT);
73             robot.climbing.moveLiftAuto(Climbing.Height.
74                 COLLECT);

```

```
64             robot.climbing.moveLiftAuto(Climbing.Height.  
PUT);  
65         } else {  
66             robot.drive.driveByEncoder(40, 0.4, Drive.  
Direction.FORWARD, 3000);  
67         }  
68     }  
69  
70     @Override  
71     public void runOpMode() throws InterruptedException {  
72         log.init(this);  
73         robot.init(hardwareMap, this, log);  
74         waitForStart();  
75         robot.climbing.land();  
76         robot.drive.turnByGyroAbsolute(-9, 10);  
77         GoldRecognition.MineralPos goldPos = robot.drive.  
Sampling(Drive.Side.DEPOT);  
78         robot.intake.injacket();  
79         sleep(2000);  
80         robot.intake.stop();  
81         endAuto(goldPos);  
82         GlobalVariables.liftPosEndAuto = robot.climbing.  
liftMotor.getCurrentPosition();  
83     }  
84 }  
85
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28  */
29
30 package org.firstinspires.ftc.teamcode.Autonomous.Depot;
31
32 import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
33 import com.qualcomm.robotcore.eventloop.opmode.
34     LinearOpMode;
35
36 import com.qualcomm.robotcore.util.ElapsedTime;
37
38 import org.firstinspires.ftc.teamcode.RobotSystems.
39     Climbing;
40 import org.firstinspires.ftc.teamcode.RobotSystems.Drive;
41 import org.firstinspires.ftc.teamcode.RobotSystems.Robot;
42 import org.firstinspires.ftc.teamcode.Util.GoldRecognition
43 ;
44 import org.firstinspires.ftc.teamcode.Util.LogCreater;
45
46 /**
47 * doing Landing, Team Marker and Parking
48 * PTS = 50
49 * In Case of no Sampling
50 * Starting from Depot On The Lander
51 * FIRST Autonomous
52 */
53 @Autonomous(name = "DepotToCreater", group = "Auto")
54 public class DepotToCreater extends Depot {
55
56     @Override
57     public void endAuto(GoldRecognition.MineralPos goldPos)
58     {
59         double driveDis = 0, turnAngle = 0;
60         switch (goldPos) {
61             case RIGHT:
62                 driveDis = 130;
63                 turnAngle = -60;
64                 robot.drive.driveByEncoder(15, 0.3, Drive.Direction.
65                     .BACKWARD, 2);
66                 break;
67             case UNKNOWN:
68             case CENTER:
69                 driveDis = 110;
70                 turnAngle = -60;
71                 robot.drive.driveByEncoder(8, 0.3, Drive.Direction.
72                     BACKWARD, 2);
73                 break;
```

```
67     case LEFT:
68         driveDis = 100;
69         turnAngle = -45;
70         break;
71     }
72     robot.drive.turnByGyroAbsolut(turnAngle, 10);
73     robot.drive.driveByEncoder(driveDis, 0.5, Drive.
    Direction.FORWARD, 3000);
74     robot.climbing.moveAngleAuto(Climbing.Angle.COLLECT);
75     robot.intake.collect();
76     robot.climbing.moveLiftAuto(Climbing.Height.PUT);
77     robot.climbing.moveLiftAuto(Climbing.Height.COLLECT);
78     robot.climbing.moveLiftAuto(Climbing.Height.PUT);
79
80 }
81 }
82
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28  */
29
30 package org.firstinspires.ftc.teamcode.Autonomous.Creater;
31
32 import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
33 import com.qualcomm.robotcore.eventloop.opmode.
34     LinearOpMode;
35
36 import com.qualcomm.robotcore.util.ElapsedTime;
37
38
39 import org.firstinspires.ftc.teamcode.RobotSystems.
40     Climbing;
41 import org.firstinspires.ftc.teamcode.RobotSystems.Drive;
42 import org.firstinspires.ftc.teamcode.RobotSystems.Robot;
43 import org.firstinspires.ftc.teamcode.Util.GlobalVariebels
44 ;
45 import org.firstinspires.ftc.teamcode.Util.GoldRecognation
46 ;
47 import org.firstinspires.ftc.teamcode.Util.LogCreater;
48
49 /**
50  * doing Landing, Sampling and Parking.
51  * PTS=65
52  * Starting from Creater
53  */
54 @Autonomous(name = "Creater", group = "Tests")
55 public class Creater extends LinearOpMode {
56     protected Robot robot = new Robot();
57     protected ElapsedTime runtime = new ElapsedTime();
58     protected LogCreater log = new LogCreater("auto");
59
60     @Override
61     public void runOpMode() throws InterruptedException {
62         log.init(this);
63         robot.init(hardwareMap, this, log);
64         waitForStart();
65         robot.climbing.land();
66         robot.drive.turnByGyroAbsolut(-9, 10);
67         GoldRecognation.MineralPos goldPos = robot.drive.
68             Sampling(Drive.Side.CREATER);
69         goToDepot(goldPos);
70         goToCreater(goldPos);
71         GlobalVariebels.liftPosEndAuto = robot.climbing.
72             liftMotor.getCurrentPosition();
73     }
74 }
```

```
67
68     public void goToDepot(GoldRecognition.MineralPos
69         goldPos) {
70
71     public void goToCreator(GoldRecognition.MineralPos
72         goldPos) {
73         switch (goldPos) {
74             case LEFT:
75                 robot.drive.turnByGyroAbsolut(0, 10);
76                 robot.drive.driveByEncoder(15, 0.5, Drive.
77                     Direction.FORWARD, 5);
78                 robot.drive.turnByGyroAbsolut(-100, 3);
79                 robot.drive.turnByGyroAbsolut(-179, 5);
80                 robot.drive.driveByEncoder(25, 0.5, Drive.Direction
81                     .FORWARD, 3000);
82                 break;
83             case UNKNOWN:
84             case CENTER:
85                 robot.drive.turnByGyroAbsolut(0, 10);
86                 robot.drive.driveByEncoder(25, 0.5, Drive.
87                     Direction.FORWARD, 10);
88                 robot.drive.turnByGyroAbsolut(100, 5);
89                 robot.drive.turnByGyroAbsolut(180, 5);
90                 robot.drive.driveByEncoder(30, 0.5, Drive.
91                     Direction.FORWARD, 3000);
92                 break;
93             case RIGHT:
94                 robot.drive.turnByGyroAbsolut(0, 10);
95                 robot.drive.driveByEncoder(15, 0.5, Drive.
96                     Direction.FORWARD, 5);
97                 robot.drive.turnByGyroAbsolut(100, 5);
98                 robot.drive.turnByGyroAbsolut(180, 5);
99                 robot.drive.driveByEncoder(10, 0.5, Drive.
100                     Direction.FORWARD, 3000);
101                 break;
102         }
103     }
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28  */
29
30 package org.firstinspires.ftc.teamcode.Autonomous.Creater;
31
32 import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
33
34 import org.firstinspires.ftc.teamcode.RobotSystems.
35     Climbing;
36 import org.firstinspires.ftc.teamcode.RobotSystems.Drive;
37 import org.firstinspires.ftc.teamcode.RobotSystems.Robot;
38 import org.firstinspires.ftc.teamcode.Util.GoldRecognition
39 ;
40
41 @Autonomous(name = "CreaterToDepot", group = "Tests")
42 public class CreaterToDepot extends Creater {
43     @Override
44     public void goToDepot(GoldRecognition.MineralPos
45         goldPos) {
46         /*
47             if (goldPos == GoldRecognition.MineralPos.RIGHT
48                 || goldPos == GoldRecognition.MineralPos.LEFT) {
49                 robot.drive.driveByEncoder(70, 0.3, Drive.
50                     Direction.FORWARD, 3000);
51             }
52             else {
53                 robot.drive.driveByEncoder(55, 0.3, Drive.
54                     Direction.FORWARD, 3000);
55             }
56             robot.drive.turnByGyroAbsolut(55, 10);
57             robot.drive.driveByEncoder(135, 0.3, Drive.
58                 Direction.BACKWARD, 3000);
59             robot.drive.turnByGyroAbsolut(133, 10);
60             robot.drive.driveByEncoder(150, 0.5, Drive.
61                 Direction.BACKWARD, 3000);
62             //robot.climbing.moveLiftAuto(Climbing.Height.PUT
63         );
64         */
65         switch (goldPos) {
66             case LEFT:
67                 robot.drive.curvedDrive(60, 3, 0.5, Drive.
68                     Direction.FORWARD, Drive.CurvedDirection.LEFT);
69                 robot.drive.driveByEncoder(60, 0.4, Drive.
70                     Direction.BACKWARD, 5);
71                 robot.drive.turnByGyroAbsolut(130, 5);
72                 robot.drive.driveByEncoder(100, 0.5, Drive.
```

```

61 Direction.BACKWARD, 6);
62             break;
63         case UNKNOWN:
64         case CENTER:
65             robot.drive.curvedDrive(100, 10, 0.6, Drive
66             .Direction.FORWARD, Drive.CurvedDirection.LEFT);
67             robot.drive.turnByGyroAbsolut(90, 3);
68             robot.drive.driveByEncoder(100, 0.4, Drive
69             .Direction.BACKWARD, 5);
70             robot.drive.turnByGyroAbsolut(130, 5);
71             robot.drive.driveByEncoder(70, 0.5, Drive.
72             Direction.BACKWARD, 6);
73             break;
74         case RIGHT:
75             robot.drive.driveByEncoder(10, 0.5, Drive
76             .Direction.FORWARD, 5);
77             robot.drive.curvedDrive(100, 6, 0.5, Drive.
78             Direction.FORWARD, Drive.CurvedDirection.RIGHT);
79             robot.drive.driveByEncoder(60, 0.4, Drive.
80             Direction.FORWARD, 5);
81             robot.drive.turnByGyroAbsolut(140, 3);
82             robot.drive.driveByEncoder(80, 0.5, Drive.
83             Direction.BACKWARD, 6);
84             break;
85     }
86
87     @Override
88     public void goToCreator(GoldRecognition.MineralPos
goldPos) {
89         robot.drive.driveByEncoder(110, 1, Drive.
90         Direction.FORWARD, 3);
91         robot.climbing.moveAngleAuto(Climbing.Angle.
92         COLLECT);
93         robot.intake.collect();
94         robot.climbing.moveLiftAuto(Climbing.Height.PUT);
95         robot.climbing.moveLiftAuto(Climbing.Height.
96         COLLECT);
97         robot.climbing.moveLiftAuto(Climbing.Height.PUT);
98     }

```

```
95  
96 }
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28  */
29
30 package org.firstinspires.ftc.teamcode.Prototyping;
31
32 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
33 import com.qualcomm.robotcore.eventloop.opmode.OpMode;
34 import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
35 import com.qualcomm.robotcore.util.ElapsedTime;
36
37 import org.firstinspires.ftc.teamcode.Util.LogCreator;
38
39 /**
40  * Demonstrates empty OpMode
41  */
42 @TeleOp(name = "LogTest", group = "Concept")
43 @Disabled
44 public class LogTest extends OpMode {
45
46     private ElapsedTime runtime = new ElapsedTime();
47     private LogCreator log = new LogCreator("TeleOp");
48
49     @Override
50     public void init() {
51         telemetry.log().clear();
52         telemetry.addData("Status", "Initialized");
53         telemetry.log().add("init: " + runtime.seconds());
54         this.log.init(this);
55     }
56
57     /*
58      * Code to run when the op mode is first enabled goes
59      * here
60      * @see com.qualcomm.robotcore.eventloop.opmode.OpMode
61      #start()
62      */
63     @Override
64     public void init_loop() {
65
66     /*
67      * This method will be called ONCE when start is pressed
68      * @see com.qualcomm.robotcore.eventloop.opmode.OpMode#
69      loop()
70      */
71     @Override
```

```
70  public void start() {
71      runtime.reset();
72  }
73
74  /*
75   * This method will be called repeatedly in a loop
76   * @see com.qualcomm.robotcore.eventloop.opmode.OpMode#
77   *      loop()
78   */
79  @Override
80  public void loop() {
81      telemetry.addData("Status", "Run Time: " + runtime.
82      toString());
83      telemetry.log().add("loop: " + runtime.seconds());
84      log.writeLog("test", 0, "");
85  }
```

```
1  /* Copyright (c) 2018 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28 */  
29  
30 package org.firstinspires.ftc.teamcode.Prototyping;  
31  
32 import com.qualcomm.robotcore.hardware.HardwareMap;  
33  
34 import org.firstinspires.ftc.robotcore.external.  
    ClassFactory;  
35 import org.firstinspires.ftc.robotcore.external.matrices.  
    OpenGLMatrix;  
36 import org.firstinspires.ftc.robotcore.external.matrices.  
    VectorF;  
37 import org.firstinspires.ftc.robotcore.external.navigation.  
    Orientation;  
38 import org.firstinspires.ftc.robotcore.external.navigation.  
    VuforiaLocalizer;  
39 import org.firstinspires.ftc.robotcore.external.navigation.  
    VuforiaTrackable;  
40 import org.firstinspires.ftc.robotcore.external.navigation.  
    VuforiaTrackableDefaultListener;  
41 import org.firstinspires.ftc.robotcore.external.navigation.  
    VuforiaTrackables;  
42  
43 import static org.firstinspires.ftc.robotcore.external.  
    navigation.AngleUnit.DEGREES;  
44 import static org.firstinspires.ftc.robotcore.external.  
    navigation.AxesOrder.XYZ;  
45 import static org.firstinspires.ftc.robotcore.external.  
    navigation.AxesOrder.YZX;  
46 import static org.firstinspires.ftc.robotcore.external.  
    navigation.AxesReference.EXTRINSIC;  
47 import static org.firstinspires.ftc.robotcore.external.  
    navigation.VuforiaLocalizer.CameraDirection.FRONT;  
48  
49 import java.util.ArrayList;  
50 import java.util.List;  
51  
52 public class BTVuforia {  
53  
54     public static BTVuforia singalton = new BTVuforia();  
55  
56     private static final String VUFORIA_KEY = "private  
        static final VuforiaLocalizer.CameraDirection  
        CAMERA_CHOICE = FRONT";  
57
```

```

58     private static final float mmPerInch = 25.4f;
59     private static final float mmFTCFieldWidth = (12*6) *
60         mmPerInch;           // the width of the FTC field (from the
61         center point to the outer panels)
62     private static final float mmTargetHeight = (6) *
63         mmPerInch;           // the height of the center of the
64         target image above the floor
65
66     private static final float
67     CAMERA_FORWARD_DISPLACEMENT = 0;
68     private static final float
69     CAMERA_VERTICAL_DISPLACEMENT = mmTargetHeight;
70     private static final float CAMERA_LEFT_DISPLACEMENT
71     = 0;
72     private static final float CAMERA_LONG_AXIS_ROT = 90;
73     private static final float CAMERA_SHORT_AXIS_ROT = 0;
74     private static final float CAMERA_DEPTH_AXIS_ROT = -
75     90;
76
77     private VuforiaLocalizer vuforia;
78     private List<VuforiaTrackable> allTrackables = new
79     ArrayList<VuforiaTrackable>();
80     private OpenGLMatrix lastPosition = null;
81
82     public BTvuforia() {}
83
84     public void init(HardwareMap hardwareMap)
85     {
86         int cameraMonitorViewId = hardwareMap.appContext.
87             getResources().getIdentifier("cameraMonitorViewId", "id",
88             hardwareMap.appContext.getPackageName());
89         VuforiaLocalizer.Parameters parameters = new
90         VuforiaLocalizer.Parameters(cameraMonitorViewId);
91         //VuforiaLocalizer.Parameters parameters = new
92         VuforiaLocalizer.Parameters();
93         parameters.vuforiaLicenseKey = VUFORIA_KEY;
94         parameters.cameraDirection = FRONT;
95         vuforia = ClassFactory.getInstance().
96             createVuforia(parameters);
97
98         // Load the data sets that for the trackable
99         objects. These particular data
100        // sets are stored in the 'assets' part of our
101        application.
102        VuforiaTrackables targetsRoverRuckus = this.

```

```
86 vuforia.loadTrackablesFromAsset("RoverRuckus");
87         VuforiaTrackable blueRover = targetsRoverRuckus.
88             get(0);
89         blueRover.setName("Blue-Rover");
90         VuforiaTrackable redFootprint =
91             targetsRoverRuckus.get(1);
92         redFootprint.setName("Red-Footprint");
93         VuforiaTrackable frontCraters =
94             targetsRoverRuckus.get(2);
95         frontCraters.setName("Front-Craters");
96         VuforiaTrackable backSpace = targetsRoverRuckus.
97             get(3);
98         backSpace.setName("Back-Space");
99         allTrackables.addAll(targetsRoverRuckus);
100
101
102         OpenGLMatrix blueRoverLocationOnField =
103             OpenGLMatrix//blue rover position
104                 .translation(0, mmFTCFieldWidth,
105 mmTargetHeight)
106                 .multiplied(Orientation.getRotationMatrix
107 (EXTRINSIC, XYZ, DEGREES, 90, 0, 0));
108         blueRover.setLocation(blueRoverLocationOnField);
109
110         OpenGLMatrix redFootprintLocationOnField =
111             OpenGLMatrix//red footprint position
112                 .translation(0, -mmFTCFieldWidth,
113 mmTargetHeight)
114                 .multiplied(Orientation.getRotationMatrix
115 (EXTRINSIC, XYZ, DEGREES, 90, 0, 180));
116         redFootprint.setLocation(
117             redFootprintLocationOnField);
118
119         OpenGLMatrix frontCratersLocationOnField =
120             OpenGLMatrix//front crater posision
121                 .translation(-mmFTCFieldWidth, 0,
122 mmTargetHeight)
123                 .multiplied(Orientation.getRotationMatrix
124 (EXTRINSIC, XYZ, DEGREES, 90, 0 , 90));
125         frontCraters.setLocation(
126             frontCratersLocationOnField);
127
128         OpenGLMatrix backSpaceLocationOnField =
129             OpenGLMatrix//back space position
130                 .translation(mmFTCFieldWidth, 0,
131 mmTargetHeight)
```

```

114             .multiplied(Orientation.getRotationMatrix
115             (EXTRINSIC, XYZ, DEGREES, 90, 0, -90));
116             backSpace.setLocation(backSpaceLocationOnField);
117
118             OpenGLMatrix phoneLocationOnRobot = OpenGLMatrix
119             //phone position on the robot
120             .translation(CAMERA_FORWARD_DISPLACEMENT,
121             CAMERA_LEFT_DISPLACEMENT, CAMERA_VERTICAL_DISPLACEMENT)
122             .multiplied(Orientation.getRotationMatrix
123             (EXTRINSIC, YZX, DEGREES,
124             CAMERA_LONG_AXIS_ROT,
125             CAMERA_SHORT_AXIS_ROT, CAMERA_DEPTH_AXIS_ROT));
126
127             /** Let all the trackable listeners know where
128             the phone is. */
129             for (VuforiaTrackable trackable : allTrackables)
130             {
131                 ((VuforiaTrackableDefaultListener)trackable.
132                 getListener()).setPhoneInformation(phoneLocationOnRobot,
133                 parameters.cameraDirection);
134             }
135             targetsRoverRuckus.activate();
136         }
137
138         //is any picture is visible if so returns which one
139         private VuforiaTrackable getVisibleTarget() {
140             for (VuforiaTrackable trackable : allTrackables)
141             {
142                 if (((VuforiaTrackableDefaultListener)
143                     trackable.getListener()).isVisible()) {
144                     return trackable;
145                 }
146             }
147             return null;
148         }
149
150         //returns the full position and rotation of the robot
151         private OpenGLMatrix getRobotPosition() {
152             VuforiaTrackable visibalePicture =
153             getVisibleTarget();
154             if (visibalePicture != null) {
155                 OpenGLMatrix robotLocationTransform = ((
156                     VuforiaTrackableDefaultListener) visibalePicture.
157                     getListener()).getUpdatedRobotLocation();
158                 if (robotLocationTransform != null) {

```

```
146                     lastPosition = robotLocationTransform;
147                 }
148             }
149             return lastPosition;
150         }
151
152     //returns visible picture name
153     public String getVisibleTargetName()
154     {
155         VuforiaTrackable trackable = getVisibleTarget();
156         if(trackable != null)
157         {
158             return trackable.getName();
159         }
160         return null;
161     }
162
163     //returns the X value of the robot position
164     public Float getXpos()
165     {
166         OpenGLMatrix robotPos = getRobotPosition();
167         if(robotPos != null)
168         {
169             return robotPos.getTranslation().get(0);
170         }
171         return null;
172     }
173
174     //returns the Y value of the robot position
175     public Float getYpos()
176     {
177         OpenGLMatrix robotPos = getRobotPosition();
178         if(robotPos != null)
179         {
180             return robotPos.getTranslation().get(1);
181         }
182         return null;
183     }
184
185     //returns the rotation of the robot around the X axis
186     public Float getRotation()
187     {
188         Orientation rotation = Orientation.getOrientation
189         (getRobotPosition(), EXTRINSIC, XYZ, DEGREES);
190         if(rotation != null)
```

```

190         {
191             return rotation.thirdAngle;
192         }
193         return null;
194     }
195
196     //returns the distance from the visible picture wall
197     public Float getDistanceFromWall()
198     {
199         Float robotPos = null;
200         Float picPos = null;
201
202         VuforiaTrackable trackable = getVisibleTarget();
203         if(trackable != null) {
204             switch (trackable.getName()) {
205                 case "Red-Footprint":
206                 case "Blue-Rover":
207                     robotPos = getYpos();
208                     picPos = trackable.getLocation().
209                         getTranslation().get(1);
210                     break;
211
212                 case "Front-Craters":
213                 case "Back-Space":
214                     robotPos = getXpos();
215                     picPos = trackable.getLocation().
216                         getTranslation().get(0);
217                     break;
218             }
219             if (robotPos != null) {
220                 return Math.abs(robotPos - picPos);
221             }
222         }
223
224         @Override
225         public String toString() {
226             OpenGLMatrix position = getRobotPosition();
227             if(position != null)
228             {
229                 VectorF translation = position.getTranslation
230 () ;
231                 Orientation rotation = Orientation.
232                 getOrientation(position, EXTRINSIC, XYZ, DEGREES);

```

```
231
232         return "Visible Target: " +
233             getVisibleTargetName() + "\n" +
234                 "distance from visible wall: " +
235                     getDistanceFromWall() + "\n" +
236                         "Pos (mm) {X, Y, Z} = " + translation
237                             .get(0) + translation.get(1) + translation.get(2) + "\n" +
238
239                         "Rot (deg) {Roll, Pitch, Heading} = "
240                         + Math.round(rotation.firstAngle) + ", " + Math.round(
241                             rotation.secondAngle) + ", " + Math.round(rotation.
242                             thirdAngle);
243
244 }
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28  */
29
30 package org.firstinspires.ftc.teamcode.Prototyping;
31
32 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
33 import com.qualcomm.robotcore.eventloop.opmode.
34     LinearOpMode;
35 import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
36 import com.qualcomm.robotcore.hardware.DcMotor;
37 import com.qualcomm.robotcore.util.ElapsedTime;
38
39
40 @TeleOp(name="Tank Drive", group="Linear Opmode")
41 @Disabled
42 public class Tank_Drive extends LinearOpMode {
43
44     // Declare OpMode members.
45     private ElapsedTime runtime = new ElapsedTime();
46     private DcMotor frontLeftMotor = null;
47     private DcMotor backLeftMotor = null;
48
49     private DcMotor frontRightMotor = null;
50     private DcMotor backRightMotor = null;
51
52     @Override
53     public void runOpMode() {
54         telemetry.addData("Status", "Initialized");
55         //         telemetry.update();
56
57         frontLeftMotor = hardwareMap.get(DcMotor.class, "front_left_drive");
58         backLeftMotor = hardwareMap.get(DcMotor.class, "back_right_drive");
59         frontRightMotor = hardwareMap.get(DcMotor.class, "front_left_drive");
60         backRightMotor = hardwareMap.get(DcMotor.class, "back_right_drive");
61
62         // Most robots need the motor on one side to be
63         // reversed to drive forward
64         // Reverse the motor that runs backwards when
65         // connected directly to the battery
66         frontLeftMotor.setDirection(DcMotor.Direction.
FORWARD);
```

```
65         backLeftMotor.setDirection(DcMotor.Direction.  
    REVERSE);  
66         frontRightMotor.setDirection(DcMotor.Direction.  
    FORWARD);  
67         backRightMotor.setDirection(DcMotor.Direction.  
    REVERSE);  
68  
69         // Wait for the game to start (driver presses  
    PLAY)  
70         waitForStart();  
71         runtime.reset();  
72  
73         // run until the end of the match (driver presses  
    STOP)  
74     while (opModeIsActive()) {  
75  
76         double leftPower;  
77         double rightPower;  
78  
79         double drive = -gamepad1.left_stick_y;  
80         double turn = gamepad1.right_stick_x;  
81         leftPower = Range.clip(drive + turn, -1.0,  
    1.0);  
82         rightPower = Range.clip(drive - turn, -1.0,  
    1.0);  
83  
84         // Send calculated power to wheels  
85         frontLeftMotor.setPower(leftPower);  
86         backLeftMotor.setPower(leftPower);  
87         frontRightMotor.setPower(rightPower);  
88         backRightMotor.setPower(rightPower);  
89  
90         // Show the elapsed game time and wheel power  
    .  
91         telemetry.addData("Status", "Run Time: " +  
    runtime.toString());  
92         telemetry.addData("Motors", "left (%.2f),  
    right (%.2f)", leftPower, rightPower);  
93         telemetry.update();  
94     }  
95 }  
96 }  
97 }
```

```
1  /* Copyright (c) 2018 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```

28  */
29
30 package org.firstinspires.ftc.teamcode.Prototyping;
31
32 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
33 import com.qualcomm.robotcore.eventloop.opmode.
34     LinearOpMode;
35
36 import org.firstinspires.ftc.robotcore.external.
37     ClassFactory;
38 import org.firstinspires.ftc.robotcore.external.navigation
39     .VuforiaLocalizer;
40 import org.firstinspires.ftc.robotcore.external.navigation
41     .VuforiaLocalizer.CameraDirection;
42 import org.firstinspires.ftc.robotcore.external.tfod.
43     Recognition;
44 import org.firstinspires.ftc.robotcore.external.tfod.
45     TFOBJECTDetector;
46
47 /**
48 * This 2018-2019 OpMode illustrates the basics of using
49 * the TensorFlow Object Detection API to
50 * determine the position of the gold and silver minerals.
51 *
52 * Use Android Studio to Copy this Class, and Paste it
53 * into your team's code folder with a new name.
54 * Remove or comment out the @Disabled line to add this
55 * opmode to the Driver Station OpMode list.
56 *
57 * IMPORTANT: In order to use this OpMode, you need to
58 * obtain your own Vuforia license key as
59 * is explained below.
60 */
61
62 @TeleOp(name = "Concept: TensorFlow Object Detection",
63         group = "Concept")
64 @Disabled
65
66 public class UVC_Camera extends LinearOpMode {
67     private static final String TFOD_MODEL_ASSET =
68         "RoverRuckus.tflite";
69     private static final String LABEL_GOLD_MINERAL = "Gold
70         Mineral";
71     private static final String LABEL_SILVER_MINERAL = "
72         Silver
73         Mineral";
74 }

```

```
59 Silver Mineral";
60
61     /*
62      * IMPORTANT: You need to obtain your own license key
63      * to use Vuforia. The string below with which
64      * 'parameters.vuforiaLicenseKey' is initialized is
65      * for illustration only, and will not function.
66      * A Vuforia 'Development' license key, can be
67      * obtained free of charge from the Vuforia developer
68      * web site at https://developer.vuforia.com/license-
69      * manager.
70      *
71      * Vuforia license keys are always 380 characters
72      * long, and look as if they contain mostly
73      * random data. As an example, here is a example of a
74      * fragment of a valid key:
75      *
76      * ...
77      * yIgIzTqZ4mWjk9wd3cZO9T1axEqzuhxoGlfOOI2dRzKS4T0hQ8kT ...
78      *
79      * Once you've obtained a license key, copy the
80      * string from the Vuforia web site
81      *
82      * and paste it in to your code on the next line,
83      * between the double quotes.
84      */
85
86     private static final String VUFORIA_KEY = "AW/F0cP
87     //////////////////////////////////////////////////////////////////
88     dbt0Negw2nqaCH9Cw2gV4ZxuUmpeJMm7XOTdQVumthQcOeoS9qktHy4Ev
89     XtMAFoh7n5KeMiLMDqtKvd1TrbYNUy3f9ST3TMkH2hFYKB6oJMJPB8oel
90     L9Bst/2XJBz0ycMMcKmSsIwyOqwOuHamAlwfT+
91     o7VusfYmY7FPvnXuhn8obCeB5x0hhjwjsBuOz2wnx1us6N5y6on0rdc1D
92     OzC2gI767QLVXAvPyJvMfgtZRGcfzFk0evuSVxrIPCRQBjQYK2s5SsBLZ
93     4sEO4HelibKK5kg0lgT9P1uHSNa8SWX+
94     4wpJm76dFwlnSL7YElieByOTXxycmOdhWaae5qb11vYYmDiBNFiwfHRDK
95     BRD";
96
97     /**
98      * {@link #vuforia} is the variable we will use to
99      * store our instance of the Vuforia
100     * localization engine.
101     */
102
103    private VuforiaLocalizer vuforia;
104
105    /**
106     * {@link #tfod} is the variable we will use to store
107     * our instance of the Tensor Flow Object
108     * Detection engine.
109     */
```

```

84     */
85     private TFObjectDetector tfod;
86
87     @Override
88     public void runOpMode() {
89         // The TFObjectDetector uses the camera frames
90         // from the VuforiaLocalizer, so we create that
91         // first.
92         initVuforia();
93
94         if (ClassFactory.getInstance().
95             canCreateTFObjectDetector()) {
96             initTfod();
97         } else {
98             telemetry.addData("Sorry!", "This device is
99             not compatible with TFOD");
100
101         /** Wait for the game to begin */
102         telemetry.addData(">", "Press Play to start
103             tracking");
104         telemetry.update();
105         waitForStart();
106
107         if (opModeIsActive()) {
108             /**
109             * Activate Tensor Flow Object Detection. */
110             if (tfod != null) {
111                 tfod.activate();
112
113                 while (opModeIsActive()) {
114                     if (tfod != null) {
115                         // getUpdatedRecognitions() will
116                         // return null if no new information is available since
117                         // the last time that call was made.
118                         List<Recognition> updatedRecognitions
119                         = tfod.getUpdatedRecognitions();
120                         if (updatedRecognitions != null) {
121                             telemetry.addData("# Object
122                             Detected", updatedRecognitions.size());
123                             if (updatedRecognitions.size() == 3
124                             ) {
125                                 int goldMineralX = -1;
126                                 int silverMineral1X = -1;
127                                 int silverMineral2X = -1;

```

```

121                     for (Recognition recognition :
122                         updatedRecognitions) {
123                             if (recognition.getLabel() .
124                             equals(LABEL_GOLD_MINERAL)) {
125                                 goldMineralX = (int)
126                                 recognition.getLeft();
127                             } else if (silverMineral1X == -
128                                         1) {
129                                 silverMineral1X = (int)
130                                 recognition.getLeft();
131                             } else {
132                                 silverMineral2X = (int)
133                                 recognition.getLeft();
134                             }
135                         }
136                         telemetry.addData("Gold Mineral
Position", goldMineralX);
137                         if (goldMineralX != -1 &&
silverMineral1X != -1 && silverMineral2X != -1) {
138                             if (goldMineralX <
silverMineral1X && goldMineralX < silverMineral2X) {
139                                 telemetry.addData("Gold
Mineral Position", "Left");
140                             } else if (goldMineralX >
silverMineral1X && goldMineralX > silverMineral2X) {
141                                 telemetry.addData("Gold
Mineral Position", "Right");
142                             } else {
143                                 telemetry.addData("Gold
Mineral Position", "Center");
144                             }
145                         }
146                     }
147                     if (tfod != null) {
148                         tfod.shutdown();
149                     }
150                 }
151             /**

```

```
153     * Initialize the Vuforia localization engine.
154     */
155     private void initVuforia() {
156         /*
157             * Configure Vuforia by creating a Parameter
158             object, and passing it to the Vuforia engine.
159         */
160         VuforiaLocalizer.Parameters parameters = new
161             VuforiaLocalizer.Parameters();
162
163         parameters.vuforiaLicenseKey = VUFORIA_KEY;
164         parameters.cameraDirection = CameraDirection.
165             FRONT;
166
167         // Instantiate the Vuforia engine
168         vuforia = ClassFactory.getInstance().
169             createVuforia(parameters);
170
171         /**
172          * Initialize the Tensor Flow Object Detection engine
173          *
174          */
175         private void initTfod() {
176             int tfodMonitorViewId = hardwareMap.appContext.
177                 getResources().getIdentifier(
178                     "tfodMonitorViewId", "id", hardwareMap.
179                     appContext.getPackageName());
180             TFObjectDetector.Parameters tfodParameters = new
181                 TFObjectDetector.Parameters(tfodMonitorViewId);
182             tfod = ClassFactory.getInstance().
183                 createTFObjectDetector(tfodParameters, vuforia);
184             tfod.loadModelFromAsset(TFOD_MODEL_ASSET,
185                 LABEL_GOLD_MINERAL, LABEL_SILVER_MINERAL);
186         }
187     }
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28 */
29
30 package org.firstinspires.ftc.teamcode.Prototyping;
31
32
33 import android.app.Activity;
34 import android.graphics.Color;
35 import android.view.View;
36 import java.io.*;
37
38
39 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
40 import com.qualcomm.robotcore.eventloop.opmode.
41     LinearOpMode;
42 import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
43 import com.qualcomm.robotcore.hardware.ColorSensor;
44 import com.qualcomm.robotcore.hardware.DcMotor;
45 import com.qualcomm.robotcore.hardware.
46     NormalizedColorSensor;
47 import com.qualcomm.robotcore.hardware.NormalizedRGB;
48 import com.qualcomm.robotcore.hardware.Servo;
49 import com.qualcomm.robotcore.hardware.SwitchableLight;
50
51 import java.io.BufferedReader;
52 import java.io.FileNotFoundException;
53 import java.io.InputStreamReader;
54 import java.io.OutputStreamWriter;
55
56 @TeleOp(name = "Sensor: Color", group = "Sensor")
57 @Disabled
58 public class SensorColor extends LinearOpMode {
59
60     ColorSensor colorSensor;
61     NormalizedColorSensor normalizedColorSensor;
62     private final String FILE_PATH = "org/firstinspires/ftc/
63         teamcode/RGB_Test.txt";
64     final double SCALE_FACTOR = 255;
65     private java.io.File file = new java.io.File(FILE_PATH);
66     FileOutputStream fileOutputStream;
67     PrintStream printStream;
68     View relativeLayout;
69     private DcMotor collectMotor;
70     private Servo servo;
71     float hsvValues[] = {0F, 0F, 0F};
```

```
70
71     @Override
72
73     public void runOpMode() throws InterruptedException {
74
75         // Get a reference to the RelativeLayout so we can
76         // later change the background
77         // color of the Robot Controller app to match the hue
78         // detected by the RGB sensor.
79         int relativeLayoutId = hardwareMap.appContext.
80             getResources().getIdentifier("RelativeLayout", "id",
81             hardwareMap.appContext.getPackageName());
82         relativeLayout = ((Activity) hardwareMap.appContext).
83             findViewById(relativeLayoutId);
84         colorSensor = hardwareMap.get(ColorSensor.class, "color_sensor");
85         collectMotor = hardwareMap.get(DcMotor.class, "collectMotor");
86         collectMotor.setDirection(DcMotor.Direction.FORWARD);
87         normalizedColorSensor = hardwareMap.get(
88             NormalizedColorSensor.class, "color_sensor");
89         servo = hardwareMap.get(Servo.class, "Servo");
90         servo.setPosition(0.20);
91
92         waitForStart();
93         try {
94             File file = new File("/sdcard/FIRST/RGB_Test.txt");
95             fileOutputStream = new FileOutputStream(file);
96             printStream = new PrintStream(fileOutputStream);
97
98             printStream.write("Time | Hue | S_Value | Value \n");
99             printStream.write(new byte[10]);
100
101             while (opModeIsActive()) {
102                 collectMotor.setPower(0.8);
103                 NormalizedRGBA colors = normalizedColorSensor.
104                     getNormalizedColors();
105
106                 Color.RGBToHSV((int) (colorSensor.red() * SCALE_FACTOR),
107                               (int) (colorSensor.green() * SCALE_FACTOR
108                               ),
109                               (int) (colorSensor.blue() * SCALE_FACTOR)
110                               ,
111                               hsvValues);
112             }
113         }
```

```
102
103         printStream.write((((String) (time + "|" +
104             hsvValues[0] + "|" + hsvValues[1] + "|" + hsvValues[2]
105             ) + "\n")).getBytes());
106         telemetry.addLine(time + "|" + colors.red + "|" +
107             colors.green + "|" + colors.blue + "\n");
108         telemetry.addLine("Red : " + colorSensor.red() +
109             "| green : " + colorSensor.green() + "| blue : " +
110             colorSensor.blue());
111         telemetry.addLine("Hue : " + hsvValues[0] + "|" +
112             s : " + hsvValues[1] + "|" value : " + hsvValues[2]);
113         telemetry.update();
114
115
116     } catch (IOException e) {
117     } finally {
118
119
120         relativeLayout.post(new Runnable() {
121             public void run() {
122                 relativeLayout.setBackgroundColor(Color.WHITE);
123             }
124         });
125     }
126
127 }
128
129 protected void runSample() throws InterruptedException
{
130
131     // values is a reference to the hsvValues array.
132     float[] hsvValues = new float[3];
133     final float values[] = hsvValues;
134
135     // bPrevState and bCurrState keep track of the
136     // previous and current state of the button
137     boolean bPrevState = false;
138     boolean bCurrState = false;
```

```

139     // Get a reference to our sensor object.
140     //colorSensor = hardwareMap.get(NormalizedColorSensor
141     .class, "color_sensor");
142
143     // If possible, turn the light on in the beginning (
144     // it might already be on anyway,
145     // we just make sure it is if we can).
146     if (colorSensor instanceof SwitchableLight) {
147         ((SwitchableLight) colorSensor).enableLight(true);
148     }
149
150
151     // Wait for the start button to be pressed.
152     waitForStart();
153
154
155     // Loop until we are asked to stop
156     while (opModeIsActive()) {
157         // Check the status of the x button on the gamepad
158         bCurrState = gamepad1.x;
159
160         // If the button state is different than what it
161         // was, then act
162         if (bCurrState != bPrevState) {
163             // If the button is (now) down, then toggle the
164             // light
165             if (bCurrState) {
166                 if (colorSensor instanceof SwitchableLight) {
167                     SwitchableLight light = (SwitchableLight)
168                     colorSensor;
169                     light.enableLight(!light.isLightOn());
170                 }
171             }
172             bPrevState = bCurrState;
173
174             // Read the sensor
175             NormalizedRGBA colors = normalizedColorSensor.
176             getNormalizedColors();
177
178             /** Use telemetry to display feedback on the driver
179             station. We show the conversion
180             * of the colors to hue, saturation and value, and
181             * display the the normalized values
182             * as returned from the sensor.
183             * @see <u><a href="http://infohost.nmt.edu/tcc/help/
185             pubs/colortheory/web/hsv.html">HSV</a></u> */

```

```

175
176     Color.colorToHSV(colors.toColor(), hsvValues);
177     telemetry.addLine()
178         .addData("H", "%.3f", hsvValues[0])
179         .addData("S", "%.3f", hsvValues[1])
180         .addData("V", "%.3f", hsvValues[2]);
181     telemetry.addLine()
182         .addData("a", "%.3f", colors.alpha)
183         .addData("r", "%.3f", colors.red)
184         .addData("g", "%.3f", colors.green)
185         .addData("b", "%.3f", colors.blue);
186
187     /** We also display a conversion of the colors to
an equivalent Android color integer.
188     * @see Color */
189
190
191     /*telemetry.addLine("raw Android color: ")
192         .addData("a", "%02x", Color.alpha(color))
193         .addData("r", "%02x", Color.red(color))
194         .addData("g", "%02x", Color.green(color))
195         .addData("b", "%02x", Color.blue(color));*/
196
197     // Balance the colors. The values returned by
198     // getColors() are normalized relative to the
199     // maximum possible values that the sensor can
200     // measure. For example, a sensor might in a
201     // particular configuration be able to internally
202     // measure color intensity in a range of
203     // [0, 10240]. In such a case, the values returned
204     // by getColors() will be divided by 10240
205     // so as to return a value in the range [0,1].
206     // However, and this is the point, even so, the
207     // values we see here may not get close to 1.0 in,
208     // e.g., low light conditions where the
209     // sensor measurements don't approach their maximum
210     // limit. In such situations, the *relative*
211     // intensities of the colors are likely what is
212     // most interesting. Here, for example, we boost
213     // the signal on the colors while maintaining their
214     // relative balance so as to give more
215     // vibrant visual feedback on the robot controller
216     // visual display.
217     float max = Math.max(Math.max(Math.max(colors.red,
218         colors.green), colors.blue), colors.alpha);

```

```
208     colors.red    /= max;
209     colors.green  /= max;
210     colors.blue   /= max;
211     color = colors.toColor();
212
213     telemetry.addLine("normalized color:  ")
214         .addData("a", "%02x", Color.alpha(color))
215         .addData("r", "%02x", Color.red(color))
216         .addData("g", "%02x", Color.green(color))
217         .addData("b", "%02x", Color.blue(color));
218     telemetry.update();
219
220     // convert the RGB values to HSV values.
221     Color.RGBToHSV(Color.red(color), Color.green(color)
222     , Color.blue(color), hsvValues);
223
224     // change the background color to match the color
225     // detected by the RGB sensor.
226     // pass a reference to the hue, saturation, and
227     // value array as an argument
228     // to the HSVToColor method.
229     relativeLayout.post(new Runnable() {
230         public void run() {
231             relativeLayout.setBackgroundColor(Color.
232                 HSVToColor(0xff, values));
233         }
234     }
235
236
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28 */  
29  
30 package org.firstinspires.ftc.teamcode.Prototyping;  
31  
32 import com.qualcomm.robotcore.eventloop.opmode.Disabled;  
33 import com.qualcomm.robotcore.eventloop.opmode.  
    LinearOpMode;  
34 import com.qualcomm.robotcore.eventloop.opmode.TeleOp;  
35 import com.qualcomm.robotcore.hardware.DigitalChannel;  
36 @TeleOp(name = "Sensor: Digital touch", group = "Sensor")  
37 @Disabled  
38 public class SensorTouch extends LinearOpMode {  
39  
40     DigitalChannel digitalTouch;  
41  
42     @Override  
43     public void runOpMode() {  
44  
45         digitalTouch = hardwareMap.get(DigitalChannel.  
    class, "sensor_digital");  
46  
47         digitalTouch.setMode(DigitalChannel.Mode.INPUT);  
48  
49         waitForStart();  
50  
51         while (opModeIsActive()) {  
52  
53             if (digitalTouch.getState() == true) {  
54                 telemetry.addData("Digital Touch", "Is Not  
    Pressed");  
55             } else {  
56                 telemetry.addData("Digital Touch", "Is  
    Pressed");  
57             }  
58  
59             telemetry.update();  
60         }  
61     }  
62 }  
63
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28 */  
29  
30 package org.firstinspires.ftc.teamcode.Prototyping;  
31  
32 import com.qualcomm.robotcore.eventloop.opmode.Disabled;  
33 import com.qualcomm.robotcore.eventloop.opmode.OpMode;  
34 import com.qualcomm.robotcore.eventloop.opmode.TeleOp;  
35 import com.qualcomm.robotcore.util.ElapsedTime;  
36  
37 import java.io.File;  
38 import java.io.FileOutputStream;  
39 import java.io.IOException;  
40 import java.io.PrintStream;  
41 import java.util.Date;  
42  
43  
44 @TeleOp(name = "file trying", group = "try")  
45 @Disabled  
46  
47 public class FileInserting extends OpMode {  
48  
49     private ElapsedTime runtime = new ElapsedTime();  
50     private java.io.File file;  
51     private final String FILE_PATH = "/sdcard/FIRST/  
calibration.txt";  
52     private FileOutputStream fileOutputStream;  
53     private PrintStream printStream;  
54  
55  
56  
57     @Override  
58     public void init() {  
59         try {  
60             file = new File(FILE_PATH);  
61             fileOutputStream = new FileOutputStream(file);  
62             printStream = new PrintStream(fileOutputStream);  
63         } catch(IOException e) {}  
64         telemetry.addData("Status", "Initialized");  
65  
66     }  
67  
68     /*  
69      * Code to run when the op mode is first enabled goes  
here  
70      * @see com.qualcomm.robotcore.eventloop.opmode.OpMode
```

```
70  #start()
71      */
72  @Override
73  public void init_loop() {
74  }
75
76  /*
77   * This method will be called ONCE when start is
78   * pressed
79   * @see com.qualcomm.robotcore.eventloop.opmode.OpMode#
80   * loop()
81   */
80  @Override
81  public void start() {
82      try {
83          printStream.write(("HELLO WORLD!!").getBytes());
84      } catch (IOException e) {}
85      runtime.reset();
86  }
87
88  /*
89   * This method will be called repeatedly in a loop
90   * @see com.qualcomm.robotcore.eventloop.opmode.OpMode#
91   * loop()
91   */
92  @Override
93  public void loop() {
94      telemetry.addData("Status", "Run Time: " + runtime.
95      toString());
95  }
96 }
97
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```

28  */
29
30 package org.firstinspires.ftc.teamcode.Prototyping;
31
32 import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
33 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
34 import com.qualcomm.robotcore.eventloop.opmode.
35 LinearOpMode;
36 import com.qualcomm.robotcore.eventloop.opmode.OpMode;
37 import com.qualcomm.robotcore.hardware.DcMotor;
38 import com.qualcomm.robotcore.hardware.HardwareMap;
39 import com.qualcomm.robotcore.util.ElapsedTime;
40
41
42 @Autonomous(name="moveMotorByTicks", group="Pushbot")
43 @Disabled
44 public class MoveMotorByTicks extends LinearOpMode {
45
46     HardwarePushbot           robot      = new HardwarePushbot(
47 ) ;
48
49     private ElapsedTime        runtime   = new ElapsedTime();
50
51     static final double        COUNTS_PER_MOTOR_REV      = 1440
52     ;                      // eg: TETRIX Motor Encoder
53     static final double        DRIVE_GEAR_REDUCTION    = 2.0
54     ;                      // This is < 1.0 if geared UP
55     static final double        WHEEL_DIAMETER_INCHES   = 4.0
56     ;                      // For figuring circumference
57     static final double        COUNTS_PER_CM            = (
58         COUNTS_PER_MOTOR_REV * DRIVE_GEAR_REDUCTION) /
59         (WHEEL_DIAMETER_INCHES * 3.1415);
60     static final double        DRIVE_SPEED             = 0.6;
61     static final double        TURN_SPEED              = 0.5;
62
63
64     private DcMotor motor;
65
66
67     public void init(HardwareMap hardwareMap) {
68         motor = hardwareMap.get(DcMotor.class, "Drive");
69
70         motor.setDirection(DcMotor.Direction.FORWARD);
71
72         motor.setMode(DcMotor.RunMode.
73 STOP_AND_RESET_ENCODER);

```

```
65         motor.setMode(DcMotor.RunMode.RUN_USING_ENCODER);  
66  
67         motor.setPower(0);  
68     }  
69  
70     @Override  
71     public void runOpMode() {  
72         init(hardwareMap);  
73  
74         telemetry.addData("Status", "Resetting Encoders");  
75         //  
76         telemetry.update();  
77  
78         motor.setMode(DcMotor.RunMode.  
STOP_AND_RESET_ENCODER);  
79         motor.setMode(DcMotor.RunMode.RUN_USING_ENCODER);  
80  
81         telemetry.addLine("init complete");  
82         telemetry.update();  
83         waitForStart();  
84         telemetry.addLine("Start");  
85         telemetry.update();  
86         driveByTicks(30, 0.5);  
87  
88         sleep(1000); // pause for servos to move  
89         telemetry.addData("Path", "Complete");  
90         telemetry.update();  
91     }  
92     public void driveByTicks(int motorTicks, double power  
) {  
93         motor.setTargetPosition(motorTicks);  
94  
95         motor.setMode(DcMotor.RunMode.RUN_TO_POSITION);  
96         motor.setPower(power);  
97         while (motor.isBusy()) {  
98             telemetry.addLine("ticks " + motor.  
getCurrentPosition());  
99             telemetry.update();  
100        }  
101    }  
102  
103    motor.setPower(0);  
104    }  
105}
```

106

107

```
1  /* Copyright (c) 2018 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28  */
29
30 package org.firstinspires.ftc.teamcode.Prototyping;
31
32 import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
33 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
34 import com.qualcomm.robotcore.eventloop.opmode.
35     LinearOpMode;
36
37 import org.firstinspires.ftc.robotcore.external.
38     ClassFactory;
39 import org.firstinspires.ftc.robotcore.external.matrices.
40     OpenGLMatrix;
41 import org.firstinspires.ftc.robotcore.external.matrices.
42     VectorF;
43 import org.firstinspires.ftc.robotcore.external.navigation
44     .Orientation;
45 import org.firstinspires.ftc.robotcore.external.navigation
46     .VuforiaLocalizer;
47 import org.firstinspires.ftc.robotcore.external.navigation
48     .VuforiaTrackable;
49 import org.firstinspires.ftc.robotcore.external.navigation
50     .VuforiaTrackableDefaultListener;
51 import org.firstinspires.ftc.robotcore.external.navigation
52     .VuforiaTrackables;
53
54 import java.util.ArrayList;
55 import java.util.List;
56
57 import static org.firstinspires.ftc.robotcore.external.
58     navigation.AngleUnit.DEGREES;
59 import static org.firstinspires.ftc.robotcore.external.
60     navigation.AxesOrder.XYZ;
61 import static org.firstinspires.ftc.robotcore.external.
62     navigation.AxesOrder.YZX;
63 import static org.firstinspires.ftc.robotcore.external.
64     navigation.AxesReference.EXTRINSIC;
65 import static org.firstinspires.ftc.robotcore.external.
66     navigation.VuforiaLocalizer.CameraDirection.BACK;
67 import static org.firstinspires.ftc.robotcore.external.
68     navigation.VuforiaLocalizer.CameraDirection.FRONT;
69
70
71 /**
72 * This 2018-2019 OpMode illustrates the basics of using
```

```
57 * the Vuforia localizer to determine
58 * positioning and orientation of robot on the FTC field.
59 * The code is structured as a LinearOpMode
60 *
61 * Vuforia uses the phone's camera to inspect it's
62 * surroundings, and attempt to locate target images.
63 * When images are located, Vuforia is able to determine
64 * the position and orientation of the
65 * image relative to the camera. This sample code then
66 * combines that information with a
67 * knowledge of where the target images are on the field,
68 * to determine the location of the camera.
69 *
70 * This example assumes a "square" field configuration
71 * where the red and blue alliance stations
72 * are on opposite walls of each other.
73 * From the Audience perspective, the Red Alliance
74 * station is on the right and the
75 * Blue Alliance Station is on the left.
76 *
77 * The four vision targets are located in the center of
78 * each of the perimeter walls with
79 * the images facing inwards towards the robots:
80 *      - BlueRover is the Mars Rover image target on the
81 *        wall closest to the blue alliance
82 *      - RedFootprint is the Lunar Footprint target on
83 *        the wall closest to the red alliance
84 *      - FrontCraters is the Lunar Craters image target
85 *        on the wall closest to the audience
86 *      - BackSpace is the Deep Space image target on the
87 *        wall farthest from the audience
88 *
89 * A final calculation then uses the location of the
90 * camera on the robot to determine the
91 * robot's location and orientation on the field.
92 *
93 * @see VuforiaLocalizer
94 * @see VuforiaTrackableDefaultListener
95 * see ftc_app/doc/tutorial/
96 *      FTC_FieldCoordinateSystemDefinition.pdf
97 *
98 * Use Android Studio to Copy this Class, and Paste it
99 * into your team's code folder with a new name.
```

```

88 * Remove or comment out the @Disabled line to add this
89 * opmode to the Driver Station OpMode list.
90 *
91 * IMPORTANT: In order to use this OpMode, you need to
92 * obtain your own Vuforia license key as
93 * is explained below.
94 */
95
96 @Autonomous(name="vuforiaTest", group ="Concept")
97 @Disabled
98 public class vuforiaCameraTest extends LinearOpMode {
99
100     /*
101      * IMPORTANT: You need to obtain your own license key
102      * to use Vuforia. The string below with which
103      * 'parameters.vuforiaLicenseKey' is initialized is
104      * for illustration only, and will not function.
105      * A Vuforia 'Development' license key, can be
106      * obtained free of charge from the Vuforia developer
107      * web site at https://developer.vuforia.com/license-
108      * manager.
109      *
110      * Vuforia license keys are always 380 characters
111      * long, and look as if they contain mostly
112      * random data. As an example, here is a example of a
113      * fragment of a valid key:
114      *
115      * ...
116      yIgIzTqZ4mWjk9wd3cZ09T1axEqzuhxoGlfOOI2dRzKS4T0hQ8kT ...
117      *
118      * Once you've obtained a license key, copy the
119      * string from the Vuforia web site
120      *
121      * and paste it in to your code on the next line,
122      * between the double quotes.
123      */
124
125     private static final String VUFORIA_KEY = "AW/F0cP
126     ///////////////////////////////////////////////////
127     dbt0Negw2nqaCH9Cw2gV4ZxuUmpeJMm7XOTdQVumthQcOeoS9qktHy4Ev
128     XtMAFoh7n5KeMiLMDqtKvd1TrbYNUy3f9ST3TMkh2hFYKB6oJMJPB8oel
129     L9Bst/2XJBz0ycMMCkMSSIwy0qwOuHamAlwfT+
130     o7VusfYmY7FPvnXuhn8obCeB5x0hhjwjsBuOz2wnx1us6N5y6on0rdc1D
131     OzC2gI767QLVXAvPyJvMfgtzRGcfzFk0evuSVxrIPCRQBjQYK2s5SsBLZ
132     4sEO4HelibKK5kg0lgT9P1uHSNa8SWX+
133     4wpJm76dFw1nSL7YElieByOTXxycmOdhWaae5qb11vYYmDiBNFiwfHRDk
134     BRD";
135
136     // Since ImageTarget trackables use mm to specifiy

```

```
112 their dimensions, we must use mm for all the physical
113     // We will define some constants and conversions here
114     private static final float mmPerInch          = 25.4f;
115     private static final float mmFTCFieldWidth    = (12*6)
116     * mmPerInch;           // the width of the FTC field (from
117     // the center point to the outer panels)
118     private static final float mmTargetHeight     = (6) *
119     mmPerInch;           // the height of the center of the
120     target image above the floor
121
122     // Select which camera you want use.  The FRONT
123     // camera is the one on the same side as the screen.
124     // Valid choices are: BACK or FRONT
125     private static final VuforiaLocalizer.CameraDirection
126     CAMERA_CHOICE = BACK;
127
128     private OpenGLMatrix lastLocation = null;
129     private boolean targetVisible = false;
130
131     /**
132      * @link #vuforia} is the variable we will use to
133      // store our instance of the Vuforia
134      // localization engine.
135      */
136     VuforiaLocalizer vuforia;
137
138     @Override public void runOpMode() {
139
140         /*
141          * Configure Vuforia by creating a Parameter
142          object, and passing it to the Vuforia engine.
143          * We can pass Vuforia the handle to a camera
144          preview resource (on the RC phone);
145          * If no camera monitor is desired, use the
146          parameterless constructor instead (commented out below).
147          */
148
149         int cameraMonitorViewId = hardwareMap.appContext.
150             getResources().getIdentifier("cameraMonitorViewId", "id",
151             hardwareMap.appContext.getPackageName());
152
153         VuforiaLocalizer.Parameters parameters = new
154             VuforiaLocalizer.Parameters(cameraMonitorViewId);
155
156         // VuforiaLocalizer.Parameters parameters = new
157         // VuforiaLocalizer.Parameters();
158
159     }
```

```

142         parameters.vuforiaLicenseKey = VUFORIA_KEY ;
143         parameters.cameraDirection = CAMERA_CHOICE;
144
145         // Instantiate the Vuforia engine
146         vuforia = ClassFactory.getInstance().
147             createVuforia(parameters);
148
149         // Load the data sets that for the trackable
150         // objects. These particular data
151         // sets are stored in the 'assets' part of our
152         // application.
153
154         VuforiaTrackables targetsRoverRuckus = this.
155         vuforia.loadTrackablesFromAsset("RoverRuckus");
156
157         VuforiaTrackable blueRover = targetsRoverRuckus.
158             get(0);
159         blueRover.setName("Blue-Rover");
160
161         VuforiaTrackable redFootprint =
162             targetsRoverRuckus.get(1);
163         redFootprint.setName("Red-Footprint");
164
165         VuforiaTrackable frontCraters =
166             targetsRoverRuckus.get(2);
167         frontCraters.setName("Front-Craters");
168
169         VuforiaTrackable backSpace = targetsRoverRuckus.
170             get(3);
171         backSpace.setName("Back-Space");
172
173
174         // For convenience, gather together all the
175         // trackable objects in one easily-iterable collection */
176         List<VuforiaTrackable> allTrackables = new
177         ArrayList<VuforiaTrackable>();
178         allTrackables.addAll(targetsRoverRuckus);
179
180
181         /**
182          * In order for localization to work, we need to
183          * tell the system where each target is on the field, and
184          * where the phone resides on the robot. These
185          * specifications are in the form of <em>transformation
186          * matrices.</em>
187          * Transformation matrices are a central,
188          * important concept in the math here involved in
189          * localization.
190          * See <a href="https://en.wikipedia.org/wiki/
191          * Transformation_matrix">Transformation Matrix</a>
192          * for detailed information. Commonly, you'll
193          * encounter transformation matrices as instances

```

```

170             * of the @link OpenGLMatrix class.
171             *
172             * If you are standing in the Red Alliance
173             Station looking towards the center of the field,
174             *      - The X axis runs from your left to the
175             right. (positive from the center to the right)
176             *      - The Y axis runs from the Red Alliance
177             Station towards the other side of the field
178             *      where the Blue Alliance Station is. (Positive
179             is from the center, towards the BlueAlliance
180             station)
181             *      - The Z axis runs from the floor, upwards
182             towards the ceiling. (Positive is above the floor)
183             *
184             * This Rover Ruckus sample places a specific
185             target in the middle of each perimeter wall.
186             *
187             * Before being transformed, each target image is
188             conceptually located at the origin of the field's
189             * coordinate system (the center of the field),
190             facing up.
191             */
192
193
194             /**
195             * To place the BlueRover target in the middle of
196             the blue perimeter wall:
197             * - First we rotate it 90 around the field's X
198             axis to flip it upright.
199             * - Then, we translate it along the Y axis to
200             the blue perimeter wall.
201             */
202             OpenGLMatrix blueRoverLocationOnField =
203             OpenGLMatrix
204                 .translation(0, mmFTCFieldWidth,
205                 mmTargetHeight)
206                 .multiplied(Orientation.getRotationMatrix
207                 (EXTRINSIC, XYZ, DEGREES, 90, 0, 0));
208             blueRover.setLocation(blueRoverLocationOnField);
209
210
211             /**
212             * To place the RedFootprint target in the middle
213             of the red perimeter wall:
214             * - First we rotate it 90 around the field's X
215             axis to flip it upright.
216             * - Second, we rotate it 180 around the field's

```

```

197 Z axis so the image is flat against the red perimeter
wall
198         * and facing inwards to the center of the
field.
199         * - Then, we translate it along the negative Y
axis to the red perimeter wall.
200     */
201     OpenGLMatrix redFootprintLocationOnField =
OpenGLMatrix
202             .translation(0, -mmFTCFieldWidth,
mmTargetHeight)
203             .multiplied(Orientation.getRotationMatrix
(EXTRINSIC, XYZ, DEGREES, 90, 0, 180));
204     redFootprint.setLocation(
redFootprintLocationOnField);

205
206     /**
207         * To place the FrontCraters target in the middle
of the front perimeter wall:
208         * - First we rotate it 90 around the field's X
axis to flip it upright.
209         * - Second, we rotate it 90 around the field's Z
axis so the image is flat against the front wall
210         * and facing inwards to the center of the
field.
211         * - Then, we translate it along the negative X
axis to the front perimeter wall.
212     */
213     OpenGLMatrix frontCratersLocationOnField =
OpenGLMatrix
214             .translation(-mmFTCFieldWidth, 0,
mmTargetHeight)
215             .multiplied(Orientation.getRotationMatrix
(EXTRINSIC, XYZ, DEGREES, 90, 0, 90));
216     frontCraters.setLocation(
frontCratersLocationOnField);

217
218     /**
219         * To place the BackSpace target in the middle of
the back perimeter wall:
220         * - First we rotate it 90 around the field's X
axis to flip it upright.
221         * - Second, we rotate it -90 around the field's
Z axis so the image is flat against the back wall
222         * and facing inwards to the center of the

```

```

222 field.
223         * - Then, we translate it along the X axis to
224             the back perimeter wall.
224         */
225         OpenGLMatrix backSpaceLocationOnField =
226             OpenGLMatrix
227                 .translation(mmFTCFieldWidth, 0,
228 mmTargetHeight)
228                 .multiplied(Orientation.getRotationMatrix
229 (EXTRINSIC, XYZ, DEGREES, 90, 0, -90));
230         backSpace.setLocation(backSpaceLocationOnField);
230
231         /**
232          * Create a transformation matrix describing
233          where the phone is on the robot.
234          *
235          * The coordinate frame for the robot looks the
236          same as the field.
237          *
238          * The robot's "forward" direction is facing out
239          along X axis, with the LEFT side facing out along the Y
240          axis.
241          *
242          * Z is UP on the robot. This equates to a
243          bearing angle of Zero degrees.
244          *
245          * The phone starts out lying flat, with the
246          screen facing Up and with the physical top of the phone
247          * pointing to the LEFT side of the Robot. It's
248          very important when you test this code that the top of
249          the
250          *
251          * camera is pointing to the left side of the
252          robot. The rotation angles don't work if you flip the
253          phone.
254          *
255          * If using the rear (High Res) camera:
256          *
257          * We need to rotate the camera around it's long
258          axis to bring the rear camera forward.
259          *
260          * This requires a negative 90 degree rotation on
261          the Y axis
262          *
263          * If using the Front (Low Res) camera
264          *
265          * We need to rotate the camera around it's long
266          axis to bring the FRONT camera forward.
267          *
268          * This requires a Positive 90 degree rotation on
269          the Y axis
270          *

```

```

249         * Next, translate the camera lens to where it is
250             on the robot.
250         * In this example, it is centered (left to right
250 ), but 110 mm forward of the middle of the robot, and 200
250 mm above ground level.
251     */
252
253     final int CAMERA_FORWARD_DISPLACEMENT = 0; // eg: Camera is 110 mm in front of robot center
254     final int CAMERA_VERTICAL_DISPLACEMENT = 40
255             ; // eg: Camera is 200 mm above ground
256     final int CAMERA_LEFT_DISPLACEMENT = 0;
256 // eg: Camera is ON the robot's center line
257
258     OpenGLMatrix phoneLocationOnRobot = OpenGLMatrix
259             .translation(CAMERA_FORWARD_DISPLACEMENT,
259             CAMERA_LEFT_DISPLACEMENT, CAMERA_VERTICAL_DISPLACEMENT)
260             .multiplied(Orientation.getRotationMatrix
260             (EXTRINSIC, YZX, DEGREES,
261             CAMERA_CHOICE == FRONT ? 90 : -90
261             , 0, 0));
262
263     /** Let all the trackable listeners know where
263 the phone is. */
264     for (VuforiaTrackable trackable : allTrackables)
265     {
266         ((VuforiaTrackableDefaultListener)trackable.
266         getListener()).setPhoneInformation(phoneLocationOnRobot,
266         parameters.cameraDirection);
267     }
268
269     /** Wait for the game to begin */
270     telemetry.addData(">", "Press Play to start
270 tracking");
271     telemetry.update();
272     waitForStart();
273
274     /** Start tracking the data sets we care about.
274 */
275     targetsRoverRuckus.activate();
276     while (opModeIsActive()) {
277
278         // check all the trackable target to see
278 which one (if any) is visible.
279         targetVisible = false;

```

```

280             for (VuforiaTrackable trackable :
281                 allTrackables) {
282                 if (((VuforiaTrackableDefaultListener)
283                     trackable.getListener()).isVisible()) {
284                     telemetry.addData("Visible Target",
285                         trackable.getName());
286                     targetVisible = true;
287
288                     // getUpdatedRobotLocation() will
289                     // return null if no new information is available since
290                     // the last time that call was made,
291                     // or if the trackable is not currently visible.
292                     OpenGLMatrix robotLocationTransform =
293                         ((VuforiaTrackableDefaultListener)trackable.getListener(
294                             )).getUpdatedRobotLocation();
295                     if (robotLocationTransform != null) {
296                         lastLocation =
297                             robotLocationTransform;
298                     }
299                     break;
300                 }
301             }
302
303             // Provide feedback as to where the robot is
304             // located (if we know).
305             if (targetVisible) {
306                 // express position (translation) of
307                 // robot in inches.
308                 VectorF translation = lastLocation.
309                 getTranslation();
310                 telemetry.addData("Pos (in)", "{X, Y, Z
311 } = %.1f, %.1f, %.1f",
312                     translation.get(0) / mmPerInch,
313                     translation.get(1) / mmPerInch, translation.get(2) /
314                     mmPerInch);
315
316                 // express the rotation of the robot in
317                 // degrees.
318                 Orientation rotation = Orientation.
319                 getOrientation(lastLocation, EXTRINSIC, XYZ, DEGREES);
320                 telemetry.addData("Rot (deg)", "{Roll,
321                 Pitch, Heading} = %.0f, %.0f, %.0f",
322                     rotation.firstAngle,
323                     rotation.secondAngle, rotation.thirdAngle);
324             }
325             else {

```

```
307                     telemetry.addData("Visible Target", "none  
") ;  
308                 }  
309                 telemetry.update();  
310             }  
311         }  
312     }  
313
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28 */  
29  
30 package org.firstinspires.ftc.teamcode.Prototyping;  
31  
32 import com.qualcomm.robotcore.eventloop.opmode.Disabled;  
33 import com.qualcomm.robotcore.eventloop.opmode.OpMode;  
34 import com.qualcomm.robotcore.eventloop.opmode.TeleOp;  
35 import com.qualcomm.robotcore.hardware.Gamepad;  
36 import com.qualcomm.robotcore.hardware.Servo;  
37 import com.qualcomm.robotcore.util.ElapsedTime;  
38  
39 @TeleOp(name = "PrototypingMoveServo", group = "Concept")  
40 @Disabled  
41 public class PrototypingMoveServo extends OpMode {  
42  
43     private Servo servo;  
44     private double targetDownPos;  
45     private double targetUpPos;  
46  
47  
48     private ElapsedTime runtime = new ElapsedTime();  
49  
50     @Override  
51     public void init() {  
52         telemetry.addData("Status", "Initialized");  
53         servo = hardwareMap.get(Servo.class, "servo");  
54     }  
55  
56  
57     @Override  
58     public void init_loop() {  
59     }  
60  
61     @Override  
62     public void start() {  
63         runtime.reset();  
64     }  
65  
66  
67     @Override  
68     public void loop() {  
69         telemetry.addData("Status", "Run Time: " + runtime.  
70         toString());  
71         if (gamepad1.a) {  
72             targetDownPos = servo.getPosition() - 0.05;
```

```
72         servo.setPosition(targetDownPos);  
73     }  
74  
75     if (gamepad1.y) {  
76         targetUpPos = servo.getPosition() + 0.05;  
77         servo.setPosition(targetUpPos);  
78     }  
79  
80     telemetry.addData("servoPos: ", servo.getPosition());  
81 }  
82 }  
83 }
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```
28  */
29
30 package org.firstinspires.ftc.teamcode.Prototyping;
31
32 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
33 import com.qualcomm.robotcore.eventloop.opmode.OpMode;
34 import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
35 import com.qualcomm.robotcore.hardware.AnalogInput;
36 import com.qualcomm.robotcore.util.ElapsedTime;
37
38 /**
39  * Demonstrates empty OpMode
40 */
41 @TeleOp(name = "PotentometerPrototype", group = "Concept")
42 @Disabled
43 public class PotentometerPrototype extends OpMode {
44
45     private ElapsedTime runtime = new ElapsedTime();
46     private AnalogInput potentiometer;
47
48     @Override
49     public void init() {
50         telemetry.addData("Status", "Initialized");
51         potentiometer = hardwareMap.get(AnalogInput.class, "potentiometer");
52     }
53
54     /*
55      * Code to run when the op mode is first enabled goes here
56      * @see com.qualcomm.robotcore.eventloop.opmode.OpMode
57      #start()
58      */
59     @Override
60     public void init_loop() {
61
62
63     @Override
64     public void start() {
65         runtime.reset();
66     }
67
68     /*
69      * This method will be called repeatedly in a loop

```

```
70     * @see com.qualcomm.robotcore.eventloop.opmode.OpMode#  
71     */  
72     @Override  
73     public void loop() {  
74  
75         telemetry.addData("Status", "Run Time: " + runtime.  
    toString());  
76         telemetry.addData(" potentiometer: ", potentiometer.  
    getVoltage());  
77         telemetry.update();  
78     }  
79 }  
80
```

```
1 package org.firstinspires.ftc.teamcode.RobotSystems;
2
3 import com.qualcomm.hardware.bosch.BNO055IMU;
4 import com.qualcomm.robotcore.eventloop.opmode.
5     LinearOpMode;
6 import com.qualcomm.robotcore.eventloop.opmode.OpMode;
7 import com.qualcomm.robotcore.hardware.DcMotor;
8 import com.qualcomm.robotcore.hardware.Gamepad;
9 import com.qualcomm.robotcore.hardware.HardwareMap;
10 import com.qualcomm.robotcore.util.Range;
11
12 import org.firstinspires.ftc.teamcode.Pixy.PixyBlock;
13 import org.firstinspires.ftc.teamcode.Pixy.PixyCam;
14 import org.firstinspires.ftc.teamcode.Util.LogCreator;
15 import org.firstinspires.ftc.teamcode.Util.PIDController;
16 import org.firstinspires.ftc.teamcode.Util.GoldRecognition
17 ;
18 import org.firstinspires.ftc.teamcode.Util.
19     TurnPIDController;
20
21 public class Drive {
22     public enum Direction {
23         FORWARD, BACKWARD;
24     }
25     public enum CurvedDirection {
26         LEFT, RIGHT;
27     }
28     public enum Side {
29         DEPOT,
30         CREATER;
31     }
32
33     static final double COUNTS_PER_MOTOR_REV = 28;
34     static final double DRIVE_GEAR_REDUCTION = 40;
35     static final double WHEEL_DIAMETER_CM = 10.16;
36     static final double COUNTS_PER_CM = (
37         COUNTS_PER_MOTOR_REV * DRIVE_GEAR_REDUCTION) /
38             (WHEEL_DIAMETER_CM * 3.141592654);
39     private final double KP = 0.05, KI = 0.03, KD = 0.03,
40     TOLERANCE = 1;
41     private final double turnKP = 0.01, turnKI = 0.00009,
42     turnKD = 0.002, turnTOLERANCE = 7;
43
44     private OpMode opMode;
45     private LogCreator log;
```

```
40
41     private DcMotor leftDrive;
42     private DcMotor rightDrive;
43     private BT_Gyro gyro = new BT_Gyro();
44     private GoldRecognition recognition = null;
45
46     private PIDController forwardPID;
47     private TurnPIDController turnPID;
48
49
50     public void init(HardwareMap hardwareMap, OpMode
51         opMode, LogCreator log) {
52         this.log = log;
53         init(hardwareMap, opMode);
54     }
55
56     public void init(HardwareMap hardwareMap, OpMode
57         opMode) {
58         this.opMode = opMode;
59         leftDrive = hardwareMap.get(DcMotor.class, "leftDrive");
60         rightDrive = hardwareMap.get(DcMotor.class, "rightDrive");
61         gyro.init(hardwareMap);
62         if (opMode instanceof LinearOpMode) {
63             recognition = new GoldRecognition(hardwareMap,
64             opMode);
65         }
66
67         leftDrive.setDirection(DcMotor.Direction.FORWARD);
68         rightDrive.setDirection(DcMotor.Direction.REVERSE);
69
70         leftDrive.setPower(0);
71         rightDrive.setPower(0);
72
73         leftDrive.setMode(DcMotor.RunMode.
74             STOP_AND_RESET_ENCODER);
75         rightDrive.setMode(DcMotor.RunMode.
76             STOP_AND_RESET_ENCODER);
77
78         leftDrive.setMode(DcMotor.RunMode.
79             RUN_USING_ENCODER);
80         rightDrive.setMode(DcMotor.RunMode.
81             RUN_USING_ENCODER);
82
83
84
```

```

75         forwardPID = new PIDController(KP, KI, KD,
    TOLERANCE, opMode);
76         turnPID = new TurnPIDController(turnKP, turnKI,
    turnKD, turnTOLERANCE, opMode);
77
78     }
79     public void teleOpMotion(Gamepad driver) {
80         if (driver.dpad_up) {
81             leftDrive.setPower(0.2);
82             rightDrive.setPower(0.2);
83         } else if (driver.dpad_down) {
84             leftDrive.setPower(-0.2);
85             rightDrive.setPower(-0.2);
86         } else tankDrive(-driver.left_stick_y, -driver.
    right_stick_y);
87
88         //TODO: change telemetry
89         opMode.telemetry.addLine("Drive: ");
90             addData("left motor power: ", leftDrive.
    getPower())
91             .addData("right motor power: ",
    rightDrive.getPower())
92             .addData("left motor pos: ", leftDrive.
    getCurrentPosition())
93             .addData("right motor pos: ", rightDrive.
    getCurrentPosition())
94             .addData("\ndelta in encoders: ", Math.
    abs(leftDrive.getCurrentPosition() - rightDrive.
    getCurrentPosition()));
95     }
96
97     private void tankDrive(double powerLeftDrive, double
    powerRightDrive) {
98         leftDrive.setPower(powerLeftDrive * 0.5);
99         rightDrive.setPower(powerRightDrive * 0.5);
100    }
101
102    public void driveByEncoder(double distanceCm, double
    speed, Direction direction, double timeS) {
103        if (opMode instanceof LinearOpMode) {
104            int newLeftTarget = 0;
105            int newRightTarget = 0;
106
107            leftDrive.setMode(DcMotor.RunMode.
    STOP_AND_RESET_ENCODER);

```

```
108             rightDrive.setMode(DcMotor.RunMode.  
109             STOP_AND_RESET_ENCODER);  
110             leftDrive.setMode(DcMotor.RunMode.  
111             RUN_USING_ENCODER);  
112             rightDrive.setMode(DcMotor.RunMode.  
113             RUN_USING_ENCODER);  
114  
115             if (direction == Direction.FORWARD) {  
116                 newLeftTarget = (int) (distanceCm *  
117                 COUNTS_PER_CM);  
118                 newRightTarget = (int) (distanceCm *  
119                 COUNTS_PER_CM);  
120             } else if (direction == Direction.BACKWARD) {  
121                 newLeftTarget = (int) (distanceCm *  
122                 COUNTS_PER_CM * -1);  
123                 newRightTarget = (int) (distanceCm *  
124                 COUNTS_PER_CM * -1);  
125             }  
126  
127             leftDrive.setTargetPosition(newLeftTarget);  
128             rightDrive.setTargetPosition(newRightTarget);  
129  
130             leftDrive.setMode(DcMotor.RunMode.  
131             RUN_TO_POSITION);  
132             rightDrive.setMode(DcMotor.RunMode.  
133             RUN_TO_POSITION);  
134  
135             leftDrive.setPower(Math.abs(speed));  
136             rightDrive.setPower(Math.abs(speed));  
137  
138             double stopTime = opMode.getRuntime() + timeS;  
139         ;  
140         while (((LinearOpMode) opMode).opModeIsActive()  
141             () &&  
142                 (opMode.getRuntime() < stopTime) &&  
143                 (rightDrive.isBusy() && leftDrive.  
144                 isBusy())) {  
145                 opMode.telemetry.addData("leftPos",  
146                 leftDrive.getCurrentPosition());  
147                 opMode.telemetry.addData("rightPos",  
148                 rightDrive.getCurrentPosition());  
149                 opMode.telemetry.update();  
150                 if(log != null) {
```

```
139                     log.writeLog("leftDrive", leftDrive.  
140                         getCurrentPosition(), "target: " + newLeftTarget);  
140                     log.writeLog("rightDrive", rightDrive.  
141                         getCurrentPosition(), "target: " + newRightTarget);  
141                     }  
142                 }  
143                 leftDrive.setPower(0);  
144                 rightDrive.setPower(0);  
145  
146                 leftDrive.setMode(DcMotor.RunMode.  
146                   RUN_USING_ENCODER);  
147                 rightDrive.setMode(DcMotor.RunMode.  
147                   RUN_USING_ENCODER);  
148             }  
149         }  
150  
151     public void curvedDrive(double distanceCm, double  
151       angleFactor, double speed, Direction direction,  
151       CurvedDirection curvedDirection)  
152     {  
153         int newLeftTarget;  
154         int newRightTarget;  
155  
156         double rightSpeed;  
157         double leftSpeed;  
158  
159         if(direction == Direction.BACKWARD) {  
160             distanceCm *= -1;  
161         }  
162         if(curvedDirection == CurvedDirection.RIGHT) {  
163             newLeftTarget = (int) (distanceCm *  
163               COUNTS_PER_CM);  
164             newRightTarget = (int) (distanceCm *  
164               COUNTS_PER_CM / angleFactor);  
165  
166             rightSpeed = speed / angleFactor;  
167             leftSpeed = speed;  
168         }  
169         else {  
170             newLeftTarget = (int) (distanceCm *  
170               COUNTS_PER_CM / angleFactor);  
171             newRightTarget = (int) (distanceCm *  
171               COUNTS_PER_CM);  
172  
173             rightSpeed = speed;
```

```

174             leftSpeed = speed / angleFactor;
175         }
176
177         leftDrive.setMode(DcMotor.RunMode.
178             STOP_AND_RESET_ENCODER);
179         rightDrive.setMode(DcMotor.RunMode.
180             STOP_AND_RESET_ENCODER);
181
182         leftDrive.setMode(DcMotor.RunMode.
183             RUN_USING_ENCODER);
184         rightDrive.setMode(DcMotor.RunMode.
185             RUN_USING_ENCODER);
186
187         leftDrive.setTargetPosition(newLeftTarget);
188         rightDrive.setTargetPosition(newRightTarget);
189
190         leftDrive.setMode(DcMotor.RunMode.RUN_TO_POSITION
191 );
192         rightDrive.setMode(DcMotor.RunMode.
193             RUN_TO_POSITION);
194
195         leftDrive.setPower(Math.abs(leftSpeed));
196         rightDrive.setPower(Math.abs(rightSpeed));
197
198         while (((LinearOpMode) opMode).opModeIsActive()
199             &&
200             (rightDrive.isBusy() && leftDrive.isBusy(
201             ))) {
202             opMode.telemetry.addData("leftPos", leftDrive
203                 .getCurrentPosition());
204             opMode.telemetry.addData("rightPos",
205                 rightDrive.getCurrentPosition());
206             opMode.telemetry.update();
207             if(log != null) {
208                 log.writeLog("leftDrive", leftDrive.
209                     getCurrentPosition(), "target: " + newLeftTarget);
210                 log.writeLog("rightDrive", rightDrive.
211                     getCurrentPosition(), "target" + newRightTarget);
212             }
213         }
214         leftDrive.setPower(0);
215         rightDrive.setPower(0);
216     }
217
218     public void turnByGyroAbsolut(double targetDegree,

```

```

206 double timeS) {
207     while (targetDegree > 180 && ((LinearOpMode)
208 opMode).opModeIsActive()) targetDegree -= 360;
209     while (targetDegree <= -179 && ((LinearOpMode)
210 opMode).opModeIsActive()) targetDegree += 360;
211     turnPID.reset(targetDegree, getAngle());
212     timeS += opMode.getRuntime();
213
214     while (!turnPID.onTarget() && ((LinearOpMode)
215 opMode).opModeIsActive() && opMode.getRuntime() <= timeS)
216     {
217         while (!turnPID.onTarget() && ((LinearOpMode)
218 opMode).opModeIsActive() && opMode.getRuntime() <= timeS)
219         {
220             double angleCurrection = 0;
221             /* if(targetDegree>=170) {
222                 if (getAngle() < 0) {
223                     angleCurrection = 360;
224                 }
225             */
226             double output = turnPID.getOutput(
227                 getAngle() + angleCurrection);
228             leftDrive.setPower(-output);
229             rightDrive.setPower(output);
230             opMode.telemetry.addData("error: ",
231                         turnPID.getCurrentError())
232                         .addData("output: ", output)
233                         .addData("robot angle: ",
234                         getAngle());
235             opMode.telemetry.update();
236             if (log != null) {
237                 log.writeLog("gyro", getAngle(), "
238 target: " + targetDegree + ", output: " + output);
239             }
240             double time = opMode.getRuntime() + 0.3;
241             while ((opMode.getRuntime() < time) && ((LinearOpMode) opMode).opModeIsActive()) {
242                 turnPID.updateError(getAngle());

```

```

240         }
241     }
242
243     leftDrive.setPower(0);
244     rightDrive.setPower(0);
245   }
246   public void turnByGyroRelative(double degrees, double
247 timeS) {
247       turnByGyroAbsolut(this.gyro.getAngle() + degrees,
248 timeS);
248   }
249
250   public GoldRecognition.MineralPos Sampling(Side side)
251 {
251     GoldRecognition.MineralPos pos = recognition.
252     getGoldPos(log);
252     recognition.stopTfod();
253     recognition.turnOffLeds();
254     if (side == Side.DEPOT) {
255       switch (pos) {
256         case LEFT:
257           turnByGyroAbsolut(40,2);
258           driveByEncoder(10, 0.2, Direction.
259 BACKWARD, 5);
259           curvedDrive(140, 2, 0.5, Direction.
260 BACKWARD, CurvedDirection.LEFT);
260           //driveByEncoder(80, 0.5, Direction.
260 BACKWARD, 5);
261           //turnByGyroAbsolut(-30,2);
262           //driveByEncoder(45,0.5, Direction.
262 BACKWARD, 5);
263             break;
264         case CENTER:
265         case UNKNOWN:
266           turnByGyroAbsolut(5,3);
267           driveByEncoder(75,0.5, Direction.
267 BACKWARD, 10);
268             break;
269         case RIGHT:
270           turnByGyroAbsolut(-40,10);
271           //driveByEncoder(10, 0.2, Direction.
271 BACKWARD, 5);
272           curvedDrive(140, 2, 0.5, Direction.
272 BACKWARD, CurvedDirection.RIGHT);
273           //driveByEncoder(90,0.5, Direction.

```

```
273 BACKWARD, 10);  
274 //turnByGyroAbsolut(30,10);  
275 //driveByEncoder(45,0.5, Direction.  
276 BACKWARD, 10);  
277 break;  
278 }  
279 } else {  
280 switch (pos) {  
281 case LEFT:  
282 turnByGyroAbsolut(30,10);  
283 driveByEncoder(45,0.5, Direction.  
284 BACKWARD, 10);  
285 break;  
286 case CENTER:  
287 case UNKNOWN:  
288 turnByGyroAbsolut(5,10);  
289 driveByEncoder(40,0.5, Direction.  
290 BACKWARD, 10);  
291 break;  
292 case RIGHT:  
293 turnByGyroAbsolut(-30,10);  
294 driveByEncoder(45,0.5, Direction.  
295 BACKWARD, 10);  
296 break;  
297 }  
298 opMode.telemetry.addData("mineral", pos);  
299 return pos;  
300 }  
301  
302 public double getAngle(){  
303 return gyro.getAngle();  
304 }  
305  
306 public void setOpMode(OpMode opMode) {  
307 this.opMode = opMode;  
308 }  
309  
310 public void setLog(LogCreator log) {  
311 this.log = log;  
312 }  
313 }
```

```
314     /**
315      public void driveByEncoderUsingPID(double distance,
316          Direction direction){
317              if(direction == Direction.BACKWARD) {
318                  distance *= -1;
319              }
320              leftDrive.setMode(DcMotor.RunMode.
321                  STOP_AND_RESET_ENCODER);
322              rightDrive.setMode(DcMotor.RunMode.
323                  STOP_AND_RESET_ENCODER);
324              leftDrive.setMode(DcMotor.RunMode.RUN_USING_ENCODER
325 );
326              rightDrive.setMode(DcMotor.RunMode.RUN_USING_ENCODER
327 );
328              forwardPID.reset(distance, 0);
329
330              opMode.telemetry.addData("error: " , forwardPID.
331                  getCurrentError());
332              opMode.telemetry.update();
333              while (!forwardPID.onTarget() && ((LinearOpMode)
334                  opMode).opModeIsActive()){
335                  double output = forwardPID.getOutput(leftDrive.
336                      getCurrentPosition() / COUNTS_PER_CM);
337                  leftDrive.setPower(output);
338                  rightDrive.setPower(output);
339                  opMode.telemetry.addData("error: " , forwardPID.
340                      getCurrentError())
341                      .addData("output: " , output);
342                  opMode.telemetry.update();
343                  if (log != null) {
344                      log.writeLog("leftDrive", leftDrive.
345                          getCurrentPosition() / COUNTS_PER_CM, "target: " +
346                          distance);
347                      log.writeLog("rightDrive", rightDrive.
348                          getCurrentPosition() / COUNTS_PER_CM, "target" + distance
349 );
350                  }
351                  leftDrive.setPower(0);
352                  rightDrive.setPower(0);
353                  }
354              */
355 }
```

```
1 package org.firstinspires.ftc.teamcode.RobotSystems;
2
3 import android.graphics.Color;
4
5 import com.qualcomm.robotcore.eventloop.opmode.Disabled;
6 import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
7 import com.qualcomm.robotcore.eventloop.opmode.OpMode;
8 import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
9 import com.qualcomm.robotcore.hardware.DcMotor;
10 import com.qualcomm.robotcore.hardware.Gamepad;
11 import com.qualcomm.robotcore.hardware.HardwareMap;
12
13 import org.firstinspires.ftc.teamcode.Util.LogCreater;
14
15 import java.io.File;
16 import java.io.FileNotFoundException;
17 import java.io.FileOutputStream;
18 import java.io.IOException;
19 import java.io.PrintStream;
20 import java.util.Date;
21
22 public class Robot {
23
24     public Drive drive = new Drive();
25     public Intake intake = new Intake();
26     public Climbing climbing = new Climbing();
27     private LogCreater log;
28     private OpMode opMode;
29
30     public void init(HardwareMap hardwareMap, OpMode opMode, LogCreater log) {
31         this.opMode = opMode;
32         this.log = log;
33         drive.init(hardwareMap, opMode, log);
34         intake.init(hardwareMap, opMode, log);
35         climbing.init(hardwareMap, opMode, log);
36
37         opMode.telemetry.addLine("robot initialized");
38         opMode.telemetry.update();
39     }
40
41     public void init(HardwareMap hardwareMap, OpMode opMode) {
42         drive.init(hardwareMap, opMode);
```

```
43         intake.init(hardwareMap, opMode);
44         climbing.init(hardwareMap, opMode);
45         this.opMode = opMode;
46
47         opMode.telemetry.addLine("robot initialized");
48         opMode.telemetry.update();
49     }
50
51     public void teleop(Gamepad driver, Gamepad operator) {
52         drive.teleOpMotion(driver);
53         intake.teleOpMotion(driver, operator);
54         climbing.teleOpMotion(operator);
55     }
56
57     public void setOpMode(OpMode opMode) {
58         this.opMode = opMode;
59         this.climbing.setOpMode(opMode);
60         this.drive.setOpMode(opMode);
61         this.intake.setOpMode(opMode);
62     }
63
64     public void setLog(LogCreator log) {
65         this.log = log;
66         this.climbing.setLog(log);
67         this.drive.setLog(log);
68         this.intake.setLog(log);
69     }
70 }
71
```

```
1 package org.firstinspires.ftc.teamcode.RobotSystems;
2
3 import com.qualcomm.robotcore.eventloop.opmode.OpMode;
4 import com.qualcomm.robotcore.hardware.DcMotor;
5 import com.qualcomm.robotcore.hardware.Gamepad;
6 import com.qualcomm.robotcore.hardware.HardwareMap;
7 import com.qualcomm.robotcore.hardware.Servo;
8
9 import org.firstinspires.ftc.teamcode.Util.LogCreater;
10
11 public class Intake {
12
13     private final double COLLECTION_SPEED = 0.8;
14     private final double INJECT_SPEED = 0.5;
15     private final double RELEASE_SPEED = 0.5;
16     private final double LEFT_SERVO_OPEN_POS = 0.8;
17     private final double RIGHT_SERVO_OPEN_POS = 0;
18     private final double LEFT_SERVO_CLOSE_POS = 0.20;
19     private final double RIGHT_SERVO_CLOSE_POS = 0.7;
20
21     private OpMode opMode;
22     private LogCreater log;
23
24     private DcMotor collectMotor;
25     public Servo leftServo, rightServo;
26
27     private boolean collectModeIsPrevActive;
28     private boolean releaseModeIsPrevActive;
29     private boolean releaseModeIsActive;
30     private boolean collectModeIsActive;
31     private boolean injecktIsActive;
32     private boolean injecktIsPrevActive;
33
34     private double timeToOpenRight = -1;
35
36     public void init(HardwareMap hardwareMap, OpMode
37         opMode, LogCreater log) {
38         this.log = log;
39         init(hardwareMap, opMode);
40     }
41     public void init(HardwareMap hardwareMap, OpMode
42         opMode) {
43         this.opMode = opMode;
44         leftServo = hardwareMap.get(Servo.class, "
45         leftServo");
```

```
43         rightServo = hardwareMap.get(Servo.class, "rightServo");
44
45         collectMotor = hardwareMap.get(DcMotor.class, "collectMotor");
46         collectMotor.setDirection(DcMotor.Direction.FORWARD);
47         collectMotor.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
48
49         stop();
50     }
51
52     public void teleOpMotion(Gamepad driver, Gamepad operator) {
53
54         releaseModeIsActive = driver.left_bumper || operator.left_bumper;
55         collectModeIsActive = driver.right_bumper || operator.right_bumper;
56         injecktIsActive = driver.b;
57
58         if(collectModeIsActive) {
59             this.collect();
60         }
61         else if (releaseModeIsActive) {
62             this.release();
63         }
64
65         if(collectModeIsPrevActive && !collectModeIsActive)
66         {
67             closeRightGate();
68             closeLeftGate();
69             this.stop();
70         }
71
72         if(releaseModeIsPrevActive && !releaseModeIsActive)
73         {
74             this.stop();
75             closeRightGate();
76             closeLeftGate();
77             timeToOpenRight = -1;
78         }
79
80         if (injecktIsActive) {
```

```
79             this.injacket();
80         }
81
82         if(injecktIsPrevActive && !injecktIsActive) {
83             closeRightGate();
84             closeLeftGate();
85             this.stop();
86         }
87
88         opMode.telemetry.addLine("intake: \n")
89             .addData("collectMotorPower: ",
90             collectMotor.getPower()+"\n")
91             .addData("rightServoPos: ", rightServo.
92             getPosition())
93             .addData("leftServoPos: ", leftServo.
94             getPosition()+"\n")
95             .addData("releaseModeIsPrevActive: ",
96             releaseModeIsPrevActive)
97             .addData("collectModeIsPrevActive: ",
98             collectModeIsPrevActive +"\n");
99
100        releaseModeIsPrevActive = releaseModeIsActive;
101        collectModeIsPrevActive = collectModeIsActive;
102        injecktIsPrevActive = injecktIsActive;
103    }
104
105    private void openLeftGate() {
106        leftServo.setPosition(LEFT_SERVO_OPEN_POS);
107    }
108
109    private void openRightGate() {
110        rightServo.setPosition(RIGHT_SERVO_OPEN_POS);
111    }
112
113    private void closeLeftGate() {
114        leftServo.setPosition(LEFT_SERVO_CLOSE_POS);
115    }
116
117    public void collect() {
118        collectMotor.setPower(COLLECTION_SPEED);
```

```
119     }
120
121     public void release() {
122         if (timeToOpenRight == -1) {
123             timeToOpenRight = opMode.getRuntime();
124         }
125         openLeftGate();
126         if (opMode.getRuntime() >= timeToOpenRight + 0.5)
127             openRightGate();
128         }
129         collectMotor.setPower(RELEASE_SPEED);
130     }
131
132     public void injackt() {
133         collectMotor.setPower(-INJECT_SPEED);
134         openRightGate();
135         openLeftGate();
136     }
137     public void stop() {
138         collectMotor.setPower(0);
139         closeRightGate();
140         closeLeftGate();
141     }
142
143     public void setOpMode(OpMode opMode) {
144         this.opMode = opMode;
145     }
146
147     public void setLog(LogCreator log) {
148         this.log = log;
149     }
150 }
```

```
1  /* Copyright (c) 2017 FIRST. All rights reserved.
2  *
3  * Redistribution and use in source and binary forms, with
4  * or without modification,
5  * are permitted (subject to the limitations in the
6  * disclaimer below) provided that
7  * the following conditions are met:
8  *
9  * Redistributions of source code must retain the above
10 * copyright notice, this list
11 * of conditions and the following disclaimer.
12 *
13 * Redistributions in binary form must reproduce the above
14 * copyright notice, this
15 * list of conditions and the following disclaimer in the
16 * documentation and/or
17 * other materials provided with the distribution.
18 *
19 * Neither the name of FIRST nor the names of its
20 * contributors may be used to endorse or
21 * promote products derived from this software without
22 * specific prior written permission.
23 *
24 * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
25 * RIGHTS ARE GRANTED BY THIS
26 * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
27 * HOLDERS AND CONTRIBUTORS
28 *
29 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
30 * INCLUDING, BUT NOT LIMITED TO,
31 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
32 * FOR A PARTICULAR PURPOSE
33 *
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
35 * OR CONTRIBUTORS BE LIABLE
36 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
37 * EXEMPLARY, OR CONSEQUENTIAL
38 *
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
40 * SUBSTITUTE GOODS OR
41 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
42 * INTERRUPTION) HOWEVER
43 *
44 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
45 * CONTRACT, STRICT LIABILITY,
46 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
47 * ANY WAY OUT OF THE USE
48 *
49 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
50 * SUCH DAMAGE.
```

```

28  */
29
30 package org.firstinspires.ftc.teamcode.RobotSystems;
31
32 import com.qualcomm.hardware.bosch.BNO055IMU;
33 import com.qualcomm.hardware.bosch.
34     JustLoggingAccelerationIntegrator;
35
36 import org.firstinspires.ftc.robotcore.external.navigation
37     .AngleUnit;
38 import org.firstinspires.ftc.robotcore.external.navigation
39     .AxesOrder;
40 import org.firstinspires.ftc.robotcore.external.navigation
41     .AxesReference;
42 import org.firstinspires.ftc.robotcore.external.navigation
43     .Orientation;
44 import org.firstinspires.ftc.robotcore.external.navigation
45     .Position;
46 import org.firstinspires.ftc.robotcore.external.navigation
47     .Velocity;
48
49 /**
50 * * {@link BT_Gyro} gives a short demo on how to use the
51 * BNO055 Inertial Motion Unit (IMU) from AdaFruit.
52 *
53 * * Use Android Studio to Copy this Class, and Paste it
54 * into your team's code folder with a new name.
55 * * Remove or comment out the @Disabled line to add this
56 * opmode to the Driver Station OpMode list
57 *
58 * * {@see <a href="http://www.adafruit.com/products/2472">
59 * Adafruit IMU</a>
60 */
61 public class BT_Gyro {
62     // The IMU sensor object
63     BNO055IMU imu;
64
65     public void init(HardwareMap hardwareMap) {
66         // Set up the parameters with which we will use
67         // our IMU. Note that integration
68         // algorithm here just reports accelerations to
69         // the logcat log; it doesn't actually
70         // provide positional information.
71         BNO055IMU.Parameters parameters = new BNO055IMU.
72

```

```
59 Parameters() {
60     parameters.angleUnit = BNO055IMU.AngleUnit.
61         DEGREES;
62     parameters.accelUnit = BNO055IMU.AccelUnit.
63         METERS_PERSEC_PERSEC;
64     parameters.calibrationDataFile = "
65         BNO055IMUCalibration.json"; // see the calibration sample
66         opmode
67
68         parameters.loggingEnabled = true;
69         parameters.loggingTag = "IMU";
70         parameters.accelerationIntegrationAlgorithm = new
71             JustLoggingAccelerationIntegrator();
72
73         // Retrieve and initialize the IMU. We expect the
74         // IMU to be attached to an I2C port
75         // on a Core Device Interface Module, configured
76         // to be a sensor of type "AdaFruit IMU",
77         // and named "The god of the sensors".
78         imu = hardwareMap.get(BNO055IMU.class, "imu");
79         imu.initialize(parameters);
80
81     }
82
83     public void start() {
84         // Start the logging of measured acceleration
85         imu.startAccelerationIntegration(new Position(),
86             new Velocity(), 1000);
87
88     }
89     public double getAngle() {
90         Orientation angles = imu.getAngularOrientation(
91             AxesReferenceINTRINSIC, AxesOrder.ZYX, AngleUnit.DEGREES
92         );
93
94         return angles.firstAngle;
95     }
96 }
```

```
1 package org.firstinspires.ftc.teamcode.RobotSystems;
2
3 import com.qualcomm.robotcore.eventloop.opmode.
4 LinearOpMode;
5 import com.qualcomm.robotcore.eventloop.opmode.OpMode;
6 import com.qualcomm.robotcore.hardware.AnalogInput;
7 import com.qualcomm.robotcore.hardware.DcMotor;
8 import com.qualcomm.robotcore.hardware.DigitalChannel;
9 import com.qualcomm.robotcore.hardware.Gamepad;
10 import com.qualcomm.robotcore.hardware.HardwareMap;
11
12 import org.firstinspires.ftc.teamcode.Util.GlobalVariebels
13 ;
14 import org.firstinspires.ftc.teamcode.Util.LogCreator;
15 import org.firstinspires.ftc.teamcode.Util.PIDController;
16
17 import static java.lang.Thread.sleep;
18
19 public class Climbing {
20     public enum Angle {
21         DOWN(10),
22         DRIVE_POS(30),
23         COLLECT(152),
24         LAND(52),
25         GO_TO_CLIMB(58),
26         CLIMB(30),
27         PUT(59);
28         float pos;
29         final double ticksPerDegree =93.588;
30
31         public int getTicks() {
32             return ((int) (ticksPerDegree * pos));
33         }
34
35         Angle(float ang) {
36             this.pos = ang;
37         }
38
39         public enum Height {
40             DRIVE_POS(0),
41             COLLECT(15),
42             LAND(31),
43             GO_TO_CLIMB(29),
```

```
44         CLIMB (10) ,  
45         PUT (36) ;  
46  
47         float pos;  
48         final double ticksPerCm = 134.2307;  
49  
50         public int getTicks() {  
51             return ((int) (ticksPerCm * pos));  
52         }  
53  
54         Height(float pos) {  
55             this.pos = pos;  
56         }  
57     }  
58  
59     private final double LIFT_SPEED = 1;  
60     private final double ANGLE_SPEED = 1;  
61  
62     private final double HANG_OPEN_POS = 0.8;  
63     private final double HANG_CLOSE_POS = 0;  
64  
65     private final double KP = 0.5, KI = 0, KD = 0,  
TOLERANCE = 4;  
66  
67     private OpMode opMode;  
68     private LogCreator log;  
69  
70     public DcMotor angleMotorLeft;  
71     public DcMotor angleMotorRight;  
72     public DcMotor liftMotor;  
73     private Servo hangServo;  
74     private DigitalChannel liftTouch;  
75     private DigitalChannel angleTouch;  
76     private AnalogInput potentiometer;  
77  
78     private boolean angleTouchIsPrevActive;  
79     private boolean angleTouchIsActive;  
80     private boolean liftTouchIsPrevActive;  
81     private boolean liftTouchIsActive;  
82  
83     private double angleJoystickValue = 0;  
84     private double angleJoystickPrevValue = 0;  
85     private double liftJoystickValue = 0;  
86     private double liftJoystickPrevValue = 0;  
87     private double stopPIDTime;
```

```
88
89     private PIDController anglePID;
90
91     public void init(HardwareMap hardwareMap, OpMode
92         opMode, LogCreator log) {
93         this.log = log;
94         init(hardwareMap, opMode);
95     }
96     public void init(HardwareMap hardwareMap, OpMode
97         opMode) {
98         this.opMode = opMode;
99         liftMotor = hardwareMap.get(DcMotor.class, "liftMotor");
100        angleMotorLeft = hardwareMap.get(DcMotor.class, "angleMotorLeft");
101        angleMotorRight = hardwareMap.get(DcMotor.class, "angleMotorRight");
102        hangServo = hardwareMap.get(Servo.class, "hangServo");
103        angleTouch = hardwareMap.get(DigitalChannel.class
104            , "angle_touch");
105        liftTouch = hardwareMap.get(DigitalChannel.class,
106            "lift_touch");
107        potentiometer = hardwareMap.get(AnalogInput.class
108            , "potentiometer");
109        anglePID = new PIDController(KP, KI, KD,
110            TOLERANCE, opMode);
111        angleMotorLeft.setZeroPowerBehavior(DcMotor.
112            ZeroPowerBehavior.BRAKE);
113        angleMotorRight.setZeroPowerBehavior(DcMotor.
114            ZeroPowerBehavior.BRAKE);
115        liftMotor.setZeroPowerBehavior(DcMotor.
116            ZeroPowerBehavior.BRAKE);
117
118        liftMotor.setPower(0);
119        angleMotorLeft.setPower(0);
120        angleMotorRight.setPower(0);
121
122        liftMotor.setDirection(DcMotor.Direction.REVERSE)
123        ;
124        angleMotorLeft.setDirection(DcMotor.Direction.
125            FORWARD);
126        angleMotorRight.setDirection(DcMotor.Direction.
127            REVERSE);
```

```
117
118         liftMotor.setMode(DcMotor.RunMode.
119             STOP_AND_RESET_ENCODER);
120
121         liftMotor.setMode(DcMotor.RunMode.
122             RUN_USING_ENCODER);
123         angleMotorLeft.setMode(DcMotor.RunMode.
124             RUN_WITHOUT_ENCODER);
125         angleMotorRight.setMode(DcMotor.RunMode.
126             RUN_WITHOUT_ENCODER);
127
128         liftTouch.setMode(DigitalChannel.Mode.INPUT);
129         angleTouch.setMode(DigitalChannel.Mode.INPUT);
130
131         hangServo.setDirection(Servo.Direction.REVERSE);
132     }
133
134     public void teleOpMotion(Gamepad operator) {
135         angleTouchIsActive = !angleTouch.getState();
136         liftTouchIsActive = !liftTouch.getState();
137
138         angleJoystickValue = -operator.left_stick_y;
139         liftJoystickValue = -operator.right_stick_y;
140
141         if (angleTouchIsActive && !angleTouchIsPrevActive)
142         {
143             angleMotorLeft.setPower(0);
144             angleMotorRight.setPower(0);
145         }
146
147         if (liftTouchIsActive && !liftTouchIsPrevActive)
148         {
149             liftMotor.setPower(0);
150             liftMotor.setMode(DcMotor.RunMode.
151                 STOP_AND_RESET_ENCODER);
152             liftMotor.setMode(DcMotor.RunMode.
153                 RUN_USING_ENCODER);
154             GlobalVariebels.liftPosEndAuto = 0;
155             if (log != null) {
156                 log.writeLog("liftMotor", liftMotor.
157                     getCurrentPosition(), "reset encoder");
158             }
159         }
160     }
```

```

153
154         if (!liftMotor.isBusy() && liftMotor.getMode() ==
155             DcMotor.RunMode.RUN_TO_POSITION) {
156             liftMotor.setPower(0);
157             liftMotor.setMode(DcMotor.RunMode.
158                 RUN_USING_ENCODER);
159             if (log != null) {
160                 log.writeLog("liftMotor", liftMotor.
161                     getCurrentPosition(), "arrived to target " + liftMotor.
162                     getTargetPosition());
163             }
164         }
165
166         if (anglePID.isRunning()) {
167             if (anglePID.onTarget()) {
168                 angleMotorRight.setPower(0);
169                 angleMotorLeft.setPower(0);
170                 if (opMode.getRuntime() >= stopPIDTime) {
171                     anglePID.stop();
172                     if (log != null) {
173                         log.writeLog("angleMotor",
174                             getAngle(), "arrived to target " + anglePID.getSetpoint()
175                         );
176                         anglePID.updateError(getAngle());
177                     } else {
178                         stopPIDTime = opMode.getRuntime() + 0.3;
179                         double output = anglePID.getOutput(
180                             getAngle());
181                         if (IsLiftInDeadZone()) {
182                             output *= 0.1;
183                             opMode.telemetry.addData("output: ", output);
184                             angleMotorLeft.setPower(output);
185                             angleMotorRight.setPower(output);
186                             if (log != null) {

```

```
187                     log.writeLog("angleMotor", getAngle())
188             , "target: " + anglePID.getSetpoint());
189         }
190     }
191
192     if (operator.dpad_down) {
193         moveLift(Height.DRIVE_POS);
194         moveAngle(Angle.DRIVE_POS);
195         if (log != null) {
196             log.writeLog("liftMotor", liftMotor.
197             getCurrentPosition(), "command to move to drive pos " +
198             Height.DRIVE_POS.getTicks());
199             log.writeLog("angleMotor", getAngle(), "command to move to drive pos " + Angle.DRIVE_POS.pos);
200         }
201     }
202
203     if (operator.dpad_left) {
204         moveLift(Height.CLIMB);
205         moveAngle(Angle.CLIMB);
206         if (log != null) {
207             log.writeLog("liftMotor", liftMotor.
208             getCurrentPosition(), "command to move to climb " +
209             Height.CLIMB.getTicks());
210             log.writeLog("angleMotor", getAngle(), "command to move to climb " + Angle.CLIMB.pos);
211     }
212
213     if (operator.dpad_up) {
214         moveLift(Height.GO_TO_CLIMB);
215         moveAngle(Angle.GO_TO_CLIMB);
216         openServo();
217         if (log != null) {
218             log.writeLog("liftMotor", liftMotor.
219             getCurrentPosition(), "command to move to go to climb " +
220             Height.GO_TO_CLIMB.getTicks());
221             log.writeLog("angleMotor", getAngle(), "command to move to go to climb " + Angle.GO_TO_CLIMB.pos)
222         }
223     }
224
225     if (operator.right_trigger > 0.7) {
```

```
221             moveLift(Height.COLLECT);  
222             moveAngle(Angle.COLLECT);  
223             if (log != null) {  
224                 log.writeLog("liftMotor", liftMotor.  
getCurrentPosition(), "command to move to collect " +  
Height.COLLECT.getTicks());  
225                     log.writeLog("angleMotor", getAngle(), "  
command to move to collect " + Angle.COLLECT.pos);  
226             }  
227         }  
228  
229         if (operator.left_trigger > 0.7) {  
230             moveLift(Height.PUT);  
231             moveAngle(Angle.PUT);  
232             if (log != null) {  
233                 log.writeLog("liftMotor", liftMotor.  
getCurrentPosition(), "command to move to put " + Height.  
PUT.getTicks());  
234                     log.writeLog("angleMotor", getAngle(), "  
command to move to put " + Angle.PUT.pos);  
235             }  
236         }  
237  
238         if (Math.abs(liftJoystickValue) > 0.1) {  
239             liftMoveManual(liftJoystickValue);  
240         } else if (Math.abs(liftJoystickPrevValue) > 0.1)  
{  
241             liftMoveManual(0);  
242         }  
243  
244         if (Math.abs(angleJoystickValue) > 0.1) {  
245             angleMoveManual(angleJoystickValue);  
246         } else if (Math.abs(angleJoystickPrevValue) > 0.1  
){  
247             angleMoveManual(0);  
248         }  
249  
250         if (operator.a) {  
251             openServo();  
252         }  
253  
254         if (operator.x) {  
255             lockServo();  
256         }  
257
```

```

258
259         opMode.telemetry.addLine("climbing: \n")
260                 .addData(" lift Position: ", liftMotor.
261             getCurrentPosition() + "\n")
262                 .addData(" Angle Right Position: ",
263             angleMotorRight.getCurrentPosition() + "\n")
264                 .addData(" Angle Left Position: ",
265             angleMotorLeft.getCurrentPosition() + "\n")
266                 .addData("Servo position: ", hangServo.
267             getPosition() + "\n")
268                 .addData("liftTouch: " ,
269             liftTouchIsActive)
270                 .addData(" angleTouch: " ,
271             angleTouchIsActive +"\n")
272                 .addData(" potentiometer: " ,
273             potentiometer.getVoltage())
274                 .addData("angle: ", getAngle())
275                 .addData("liftPosEndAuto",
276             GlobalVariebels.liftPosEndAuto);

277
278     angleTouchIsPrevActive = angleTouchIsActive;
279     liftTouchIsPrevActive = liftTouchIsActive;
280
281     angleJoystickPrevValue = angleJoystickValue;
282     liftJoystickPrevValue = liftJoystickValue;
283 }

284     public void moveLift(Height height) {
285         liftMotor.setTargetPosition(height.getTicks() -
286             GlobalVariebels.liftPosEndAuto);
287         liftMotor.setMode(DcMotor.RunMode.RUN_TO_POSITION
288     );
289         liftMotor.setPower(Math.abs(LIFT_SPEED));
290     }
291     angleMotorLeft.setPower(output);

```

```
292         angleMotorRight.setPower(output);  
293     }  
294  
295     private void liftMoveManual(double motorPower) {  
296         liftMotor.setMode(DcMotor.RunMode.  
RUN_USING_ENCODER);  
297         if (liftTouchIsActive && motorPower < 0) {  
298             return;  
299         }  
300         liftMotor.setPower(motorPower * LIFT_SPEED);  
301     }  
302  
303     private void angleMoveManual(double motorPower) {  
304         anglePID.stop();  
305  
306         if (angleTouchIsActive && motorPower < 0) {  
307             return;  
308         }  
309         angleMotorLeft.setPower(motorPower * ANGLE_SPEED)  
;  
310         angleMotorRight.setPower(motorPower * ANGLE_SPEED)  
);  
311     }  
312  
313     public void lockServo() {  
314         hangServo.setPosition(HANG_CLOSE_POS);  
315     }  
316  
317     public void openServo() {  
318         hangServo.setPosition(HANG_OPEN_POS);  
319     }  
320  
321  
322     //Auto methods  
323     public void moveLiftAuto(Height height) {  
324         moveLift(height);  
325         while(liftMotor.isBusy() && ((LinearOpMode)  
opMode).opModeIsActive()) {  
326             opMode.telemetry.addData("liftMotor: ",  
liftMotor.getCurrentPosition());  
327             opMode.telemetry.update();  
328             if (log != null) {  
329                 log.writeLog("lift", liftMotor.  
getCurrentPosition(), "target: " + height.getTicks());  
330             }  
331         }  
332     }  
333 }
```

```

331         }
332         liftMotor.setPower(0);
333     }
334
335     public void moveAngleAuto(Angle angle) {
336         anglePID.reset(angle.pos, getAngle());
337         while (!anglePID.onTarget() && ((LinearOpMode)
338             opMode).opModeIsActive()) {
339             double output = anglePID.getOutput(getAngle())
340         );
341             angleMotorRight.setPower(output);
342             angleMotorLeft.setPower(output);
343             opMode.telemetry.addData("error: ", anglePID.
344                 getCurrentError())
345                 .addData("output: ", output)
346                 .addData("angle", getAngle());
347             opMode.telemetry.update();
348             if (log != null) {
349                 log.writeLog("angle", getAngle(), "target
350 : " + angle.pos);
351             }
352         }
353         angleMotorRight.setPower(0);
354         angleMotorLeft.setPower(0);
355     }
356
357     public void moveAngleAndHeight(Angle angle, Height
358         height) {
359         anglePID.reset(angle.pos, getAngle());
360         liftMotor.setTargetPosition(height.getTicks());
361         liftMotor.setMode(DcMotor.RunMode.RUN_TO_POSITION
362 );
363         liftMotor.setPower(Math.abs(LIFT_SPEED));
364
365         while (!anglePID.onTarget() && ((LinearOpMode)
366             opMode).opModeIsActive()) {
367             double output = anglePID.getOutput(getAngle())
368         );
369             angleMotorRight.setPower(output);
370             angleMotorLeft.setPower(output);
371             opMode.telemetry.addLine("Angle \n")
372                 .addData("error: ", anglePID.
373                 getCurrentError())
374                 .addData("output: ", output)

```

```

367                     .addData("angle", getAngle() + "\n");
368
369             opMode.telemetry.addLine("Lift \n")
370             .addData("position: ", liftMotor.
371             getCurrentPosition());
372
373             if (!liftMotor.isBusy()) {
374                 liftMotor.setPower(0);
375                 liftMotor.setMode(DcMotor.RunMode.
376                 RUN_USING_ENCODER);
377             }
378             if (log != null) {
379                 log.writeLog("angle", getAngle(), "target
380 : " + angle.pos);
381                 log.writeLog("lift", liftMotor.
382                 getCurrentPosition(), "target: " + height.getTicks());
383             }
384
385             while(liftMotor.isBusy() && ((LinearOpMode)
386             opMode).opModeIsActive()) {
387                 opMode.telemetry.addLine("Lift \n")
388                 .addData("position: ", liftMotor.
389                 getCurrentPosition());
390                 opMode.telemetry.update();
391                 if (log != null) {
392                     log.writeLog("lift", liftMotor.
393                     getCurrentPosition(), "target: " + height.getTicks());
394                 }
395             }
396             liftMotor.setPower(0);
397             liftMotor.setMode(DcMotor.RunMode.
398             RUN_USING_ENCODER);
399         }
400
401     public void land() {
402         //moveAngleAuto(Climbing.Angle.LAND);
403         //moveLiftAuto(Climbing.Height.LAND);
404         //moveAngleAuto(Climbing.Angle.LAND);
405         moveAngleAndHeight(Climbing.Angle.LAND, Climbing.
406         Height.LAND);

```

```

403         ((LinearOpMode) opMode).sleep(600);
404         openServo();
405         ((LinearOpMode) opMode).sleep(700);
406         moveAngleAndHeight(Angle.DOWN, Height.CLIMB);
407
408         angleMotorLeft.setPower(0);
409         liftMotor.setPower(0);
410     }
411
412     public double getAngle() {
413         return potentiometer.getVoltage() / 3.347 *270;
414     }
415
416     public void setOpMode(OpMode opMode) {
417         this.opMode = opMode;
418     }
419
420     public void setLog(LogCreator log) {
421         this.log = log;
422     }
423     public boolean IsLiftInDeadZone() {
424         return getAngle()>49 && getAngle()<80;
425     }
426     public boolean IsliftInTarget(Height height) {
427         return liftMotor.getTargetPosition() >= height.
        .getTicks()+200 && liftMotor.getTargetPosition() <= height.
        .getTicks()-200;
428     }
429     /* TESTING CODE
430     public void moveAngleByPID(double target) {
431         anglePID.reset(target, getAngle());
432         while (!anglePID.onTarget() && ((LinearOpMode)
        opMode).opModeIsActive()) {
433             double output = anglePID.getOutput(getAngle
        ());
434             angleMotorRight.setPower(output);
435             angleMotorLeft.setPower(output);
436             opMode.telemetry.addData("error: " ,anglePID.
        getCurrentError())
437                 .addData("output: " , output)
438                 .addData("angle", getAngle());
439             opMode.telemetry.update();
440         }
441         angleMotorRight.setPower(0);
442         angleMotorLeft.setPower(0);

```

```
443         while (((LinearOpMode) opMode).opModeIsActive()
444             () ) {
444             opMode.telemetry.addData("error: " ,anglePID.
445             getCurrentError())
445             .addData("angle", getAngle());
446             opMode.telemetry.update();
447         }
448     }
449     */
450 }
451
```