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

```
32              one particular robot's hardware devices: eg:
    for a Pushbot.  Look at any
33              Pushbot sample to see how this can be used in
    an OpMode.
34              Teams can copy one of these to create their
    own robot definition.
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
41              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
44  basis for your own robot.  In all cases, the desired
    sample(s) needs to be copied into
45  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.
58      Choose something meaningful based on the purpose of
    this class.
59      Start with a capital letter, and remember that there
    may be more similar classes later.
```

Once your copy has been created, you should prepare it for use on your robot.
This is done by adjusting the OpMode's name, and enabling it to be displayed on the
Driver Station's OpMode list.

Each OpMode sample class begins with several lines of code like the ones shown below:

```
 @TeleOp(name="Template: Linear OpMode", group="Linear Opmode")
 @Disabled
```

The name that will appear on the driver station's "opmode list" is defined by the code:
 ``name="Template: Linear OpMode"``
You can change what appears between the quotes to better describe your opmode.
The "group=" portion of the code can be used to help organize your list of OpModes.

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

ADVANCED Multi-Team App management: Cloning the TeamCode Module

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

In this situation, you might wish to clone the TeamCode module, once for each of these teams.
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)  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""
```

```java
package org.firstinspires.ftc.teamcode.Pixy;



import android.support.annotation.NonNull;

import com.qualcomm.robotcore.hardware.I2cAddr;
import com.qualcomm.robotcore.hardware.I2cDeviceSynch;
import com.qualcomm.robotcore.hardware.
I2cDeviceSynchDeviceWithParameters;
import com.qualcomm.robotcore.hardware.configuration.
I2cSensor;
import com.qualcomm.robotcore.hardware.configuration.
annotations.DeviceProperties;
import com.qualcomm.robotcore.hardware.configuration.
annotations.I2cDeviceType;
import com.qualcomm.robotcore.util.TypeConversion;

@I2cDeviceType()
@DeviceProperties(name = "PixyCam",description = "PixyCam"
,  xmlTag = "PixyCam")
public class PixyCam extends
I2cDeviceSynchDeviceWithParameters<I2cDeviceSynch, PixyCam
.PixyCamParams> {

    static final I2cAddr ADDRESS_I2C_DEFAULT = I2cAddr.
create7bit(0x54);

    private static final int LEGO_QUERY_BASE = 0x50;
    private static final int LEGO_QUERY_CC = 0x58;
    private static final int EXTENDED_QUERY_BASE = 0x70;
    private static final int EXTENDED_QUERY_CC = 0x78;
    private static final int LEGO_QUERY_COUNT = 6;
    private static final int LEGO_QUERY_SIG_COUNT = 5;
    private static final int LEGO_QUERY_CC_COUNT = 6;
    private static final int EXTENDED_QUERY_COUNT = 26;
    private static final int EXTENDED_QUERY_BLOCK_SIZE = 5
;
    private static final int EXTENDED_QUERY_SIG_COUNT = 25
;
    private static final int EXTENDED_QUERY_SIG_BLOCK_SIZE
 = 4;
    private static final int EXTENDED_QUERY_CC_COUNT = 25;
    private static final int EXTENDED_QUERY_CC_BLOCK_SIZE
= 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
    : Parameters not cloneable");
46              }
47          }
48      }
49
50      /**
51       * The ReadWindow used to do a PixyCam LEGO protocol
    GeneralQuery
52       */
53      private I2cDeviceSynch.ReadWindow
    legoProtocolGeneralQueryReadWindow;
54      /**
55       * The ReadWindows used to do the PixyCam LEGO
    protocol SignatureQuery.
56       */
57      private I2cDeviceSynch.ReadWindow[]
    legoProtocolSignatureQueryReadWindows;
58
59      /*
60       * The ReadWindow used to do the extneded query for
    the enhanced FTC firmware
61       */
62      private I2cDeviceSynch.ReadWindow
    extendedGeneralQueryReadWindow;
63      /**
64       * The ReadWindows used to do the PixyCam LEGO
    protocol SignatureQuery.
65       */
66      private I2cDeviceSynch.ReadWindow[]
    extnededSignatureQueryReadWindows;
67
68      public PixyCam(I2cDeviceSynch deviceSynch) {
```

```
69          super(deviceSynch, true, new PixyCamParams());
70
71          this.legoProtocolGeneralQueryReadWindow = new
    I2cDeviceSynch.ReadWindow(LEGO_QUERY_BASE,
    LEGO_QUERY_COUNT, I2cDeviceSynch.ReadMode.ONLY_ONCE);
72          this.legoProtocolSignatureQueryReadWindows = new
    I2cDeviceSynch.ReadWindow[MAX_SIGNATURE];
73          for (int i = MIN_SIGNATURE; i <= MAX_SIGNATURE; i
    ++) {
74              this.legoProtocolSignatureQueryReadWindows[i
    - 1] = NewLegoProtocolSignatureQueryReadWindow(
    LEGO_QUERY_BASE, LEGO_QUERY_SIG_COUNT, i);
75          }
76
77          this.extendedGeneralQueryReadWindow = new
    I2cDeviceSynch.ReadWindow(EXTENDED_QUERY_BASE,
    EXTENDED_QUERY_COUNT, I2cDeviceSynch.ReadMode.ONLY_ONCE);
78          this.extnededSignatureQueryReadWindows = new
    I2cDeviceSynch.ReadWindow[MAX_SIGNATURE];
79          for (int i = MIN_SIGNATURE; i <= MAX_SIGNATURE; i
    ++) {
80              this.extnededSignatureQueryReadWindows[i - 1]
     = NewLegoProtocolSignatureQueryReadWindow(
    EXTENDED_QUERY_BASE, EXTENDED_QUERY_SIG_COUNT, i);
81          }
82
83          this.deviceClient.setI2cAddress(
    ADDRESS_I2C_DEFAULT);
84
85          super.registerArmingStateCallback(false);
86          this.deviceClient.engage();
87      }
88
89      private I2cDeviceSynch.ReadWindow
    NewLegoProtocolSignatureQueryReadWindow(int base, int
    count, int signature) {
90          return new I2cDeviceSynch.ReadWindow(base +
    signature, count, I2cDeviceSynch.ReadMode.ONLY_ONCE);
91      }
92
93      private byte[] ReadEntireWindow(I2cDeviceSynch.
    ReadWindow readWindow) {
94          this.deviceClient.setReadWindow(readWindow);
95          return this.deviceClient.read(readWindow.
    getRegisterFirst(), readWindow.getRegisterCount());
```

```
 96      }
 97
 98      /***
 99       *
100       * @return a Block object containing details about
    the location of the largest detected block
101       */
102      public PixyBlock getBiggestBlock() {
103          byte[] buffer = ReadEntireWindow(this.
    legoProtocolGeneralQueryReadWindow);
104
105          int signature = buffer[1] << 8 | buffer[0];
106
107          return new PixyBlock(signature, -1, buffer[2],
    buffer[3], buffer[4], buffer[5]);
108      }
109
110      /**
111       * @param signature is a value between 1 and 7
    corresponding to the signature trained into the PixyCam.
112       * @return a Block object containing details about
    the location of the largest detected block for the
    specified signature.
113       */
114      public PixyBlock getBiggestBlock(int signature) {
115          if (signature < MIN_SIGNATURE || signature >
    MAX_SIGNATURE) {
116              throw new IllegalArgumentException("signature
     must be between 1 and 7");
117          }
118
119          byte[] buffer = ReadEntireWindow(this.
    legoProtocolSignatureQueryReadWindows[signature - 1]);
120
121          return new PixyBlock(signature, buffer[0], buffer
    [1], buffer[2], buffer[3], buffer[4]);
122      }
123
124      public PixyBlockList getBiggestBlocks() {
125          byte[] buffer = ReadEntireWindow(this.
    extendedGeneralQueryReadWindow);
126
127          PixyBlockList list = new PixyBlockList(buffer[0])
    ;
128
```

```
129         for (int i = 1; i < EXTENDED_QUERY_COUNT; i += EXTENDED_QUERY_BLOCK_SIZE) {
130             int signature = TypeConversion.unsignedByteToInt(buffer[i]);
131
132             list.add(new PixyBlock(signature, -1, buffer[i + 1], buffer[i + 2], buffer[i + 3], buffer[i + 4]));
133         }
134
135         return list;
136     }
137
138     public PixyBlockList getBiggestBlocks(int signature)
139     {
140         if (signature < MIN_SIGNATURE || signature > MAX_SIGNATURE) {
141             throw new IllegalArgumentException("signature must be between 1 and 7");
142         }
143
144         byte[] buffer = ReadEntireWindow(this.extnededSignatureQueryReadWindows[signature - 1]);
145
146         PixyBlockList list = new PixyBlockList(buffer[0]);
147
148         for (int i = 1; i < EXTENDED_QUERY_SIG_COUNT; i += EXTENDED_QUERY_SIG_BLOCK_SIZE) {
149             list.add(new PixyBlock(signature, -1, buffer[i], buffer[i + 1], buffer[i + 2], buffer[i + 3]));
150         }
151
152         return list;
153     }
154
155     @Override
156     protected boolean doInitialize() {
157         return true;
158     }
159
160     @Override
161     protected boolean internalInitialize(@NonNull PixyCamParams pixyCamParams) {
162         this.parameters = pixyCamParams.clone();
163         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
```

```java
package org.firstinspires.ftc.teamcode.Pixy;

import com.qualcomm.robotcore.eventloop.opmode.Disabled;
import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.I2cAddr;
import com.qualcomm.robotcore.hardware.I2cDevice;
import com.qualcomm.robotcore.hardware.I2cDeviceSynch;
import com.qualcomm.robotcore.hardware.I2cDeviceSynchDevice;
import com.qualcomm.robotcore.hardware.I2cDeviceSynchImpl;
import com.qualcomm.robotcore.hardware.Servo;

/**
 * Created by Sarthak on 7/3/2017.
 */

@TeleOp(name="Test Pixy Cam", group = "Pixy Cam")
@Disabled
public class pixyTest extends LinearOpMode {
    I2cDevice pixyCam;
    I2cDeviceSynchImpl pixyCamReader;
    I2cAddr pixyCamAddress = I2cAddr.create8bit(0x01);
    double x, y, width, height, numObjs;
    byte[] pixyData;

    @Override
    public void runOpMode() throws InterruptedException {
        pixyCam = hardwareMap.i2cDevice.get("pixy");
        pixyCamReader = new I2cDeviceSynchImpl(pixyCam, pixyCamAddress, false);
        pixyCamReader.engage();

        waitForStart();

        while(opModeIsActive()){
            pixyCamReader.engage();
            pixyData = pixyCamReader.read(0x51, 5);

            x = pixyData[1];
            y = pixyData[2];
            numObjs = pixyData[0];

            telemetry.addData("0", pixyData[0]);
```

```
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
```

```java
package org.firstinspires.ftc.teamcode.Pixy;

import com.qualcomm.robotcore.util.TypeConversion;

/**
 * Block describes the signature, location, and size of a detected block.
 */
public class PixyBlock {
    /**
     * A number from 1 through 7 corresponding to the color trained into the PixyCam,
     * or a sequence of octal digits corresponding to the multiple colors of a color code.
     */
    public final int signature;

    /**
     * The x, y location of the center of the detected block.
     * x is in the range (0, 255)
     * 0 is the left side of the field of view and 255 is the right.
     * y is in the range (0, 199)
     * 0 is the top of the field of view and 199 is the bottom.
     */
    public final int x, y;

    /**
     * The size of the detected block.
     * width or height of zero indicates no block detected.
     * maximum width is 255.
     * maximum height is 199.
     */
    public final int width, height;

    /**
     * The count of blocks detected for the given signature
     * Will be -1 if this is a block from the general query
     */
    public final int blockCount;
```

```
37
38      public PixyBlock(int signature, byte blockCount, byte
    x, byte y, byte width, byte height) {
39          this(signature, TypeConversion.unsignedByteToInt(
    blockCount), x, y, width, height);
40      }
41
42      public PixyBlock(int signature, int blockCount, byte x
    , byte y, byte width, byte height) {
43          this.signature = signature;
44          this.x = TypeConversion.unsignedByteToInt(x);
45          this.y = TypeConversion.unsignedByteToInt(y);
46          this.width = TypeConversion.unsignedByteToInt(
    width);
47          this.height = TypeConversion.unsignedByteToInt(
    height);
48          this.blockCount = blockCount;
49      }
50
51      public boolean isEmpty()
52      {
53          return x == 0 && y == 0 && width == 0 && height ==
     0 && blockCount == -1;
54      }
55
56      @Override
57      public String toString() {
58          return String.format("x: %d, y: %d, w: %d, h: %d
    cnt: %d", this.x, this.y, this.width, this.height, this.
    blockCount);
59      }
60  }
61
```

```
 1 /* Copyright (c) 2017 FIRST. All rights reserved.
 2  *
 3  * Redistribution and use in source and binary forms, with
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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
   exact parameters, and you can reference the
59         //SDK Documentation to learn more
60         I2cDeviceSynch.ReadWindow readWindow = new
   I2cDeviceSynch.ReadWindow (1, 26, I2cDeviceSynch.ReadMode.
   REPEAT);
61         pixy.setReadWindow(readWindow);
62
63         //required to "turn on" the device
64         pixy.engage();
65         telemetry.addData("Status", "Initialized");
66     }
67
68
69     @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  }
96
97
```

```java
package org.firstinspires.ftc.teamcode.Pixy;

import com.qualcomm.robotcore.util.TypeConversion;

import java.util.Vector;

public class PixyBlockList extends Vector<PixyBlock> {
    public int totalCount;

    public PixyBlockList(byte totalCount) {
        this.totalCount = TypeConversion.unsignedByteToInt(totalCount);
    }
}
```

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

```
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
   #start()
65      */
66     @Override
67     public void init() {
68         pixyCam = hardwareMap.get(PixyCam.class, "pixycam"
   );
69         try {
```

```
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 }
106
107
```

```java
package org.firstinspires.ftc.teamcode.Pixy;

import com.qualcomm.robotcore.eventloop.opmode.Autonomous;
import com.qualcomm.robotcore.eventloop.opmode.Disabled;
import com.qualcomm.robotcore.eventloop.opmode.
LinearOpMode;
import com.qualcomm.robotcore.hardware.I2cDeviceSynch;

@Autonomous(name = "ConceptSensorPixyCam")
@Disabled
public class ConceptSensorPixyCam extends LinearOpMode {
    I2cDeviceSynch pixyCam;

    double x, y, width, height, numObjects;

    byte[] pixyData;

    @Override
    public void runOpMode() throws InterruptedException {

        pixyCam = hardwareMap.i2cDeviceSynch.get("pixy");

        waitForStart();
        pixyCam.engage();
        while(opModeIsActive()){
            telemetry.addLine(pixyCam.getReadWindow().
toString());
            pixyData = pixyCam.read(0x51, 5);

            x = pixyData[1];
            y = pixyData[2];
            width = pixyData[3];
            height = pixyData[4];
            numObjects = pixyData[0];


            telemetry.addData("0", 0xff&pixyData[0]);
            telemetry.addData("1", 0xff&pixyData[1]);
            telemetry.addData("2", 0xff&pixyData[2]);
            telemetry.addData("3", 0xff&pixyData[3]);
            telemetry.addData("4", 0xff&pixyData[4]);
            telemetry.addData("Length", pixyData.length);
            telemetry.update();
            sleep (500);
        }
```

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

```java
package org.firstinspires.ftc.teamcode.Util;

import android.graphics.Path;

import com.qualcomm.robotcore.eventloop.opmode.OpMode;
import com.qualcomm.robotcore.hardware.ColorSensor;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.Servo;
import com.qualcomm.robotcore.wifi.NetworkConnection;
import com.qualcomm.robotcore.wifi.WifiAssistant;
import com.qualcomm.robotcore.wifi.WifiDirectAssistant;

import org.firstinspires.ftc.robotcontroller.internal.
FtcRobotControllerActivity;
import org.firstinspires.ftc.robotcore.internal.network.
WifiDirectAgent;
import org.firstinspires.ftc.robotcore.internal.network.
WifiDirectChannelManager;
import org.firstinspires.ftc.robotcore.internal.network.
WifiState;
import org.firstinspires.ftc.teamcode.Prototyping.
SensorColor;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintStream;
import java.sql.Time;
import java.util.Calendar;
import java.util.Date;
import java.util.Timer;
import java.util.Date;
import java.text.SimpleDateFormat;

public class LogCreater {
    private File log;
    private String PATH = "/sdcard/FIRST/Log-";
    private FileOutputStream outputStream;
    private PrintStream printStream;
    private String fileName;
    private OpMode opMode;

    public LogCreater(String fileName){
        this.fileName = fileName;
```

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

```java
package org.firstinspires.ftc.teamcode.Util;

import com.qualcomm.robotcore.eventloop.opmode.OpMode;
import com.qualcomm.robotcore.util.Range;

public class PIDController {

    protected double setpoint, tolerance, currentError;
    protected double kP, kI, kD;
    protected OpMode opMode;

    protected double prevError, sumError, prevTime;
    protected boolean running = false;

    /**
     * Instantiate a new PIDController.
     * @param kP
     * @param kI
     * @param kD
     * @param tolerance
     */
    public PIDController(double kP, double kI, double kD, double tolerance, OpMode opmode) {
        this.kP = kP;
        this.kI = kI;
        this.kD = kD;
        this.tolerance = tolerance;
        this.opMode = opmode;
    }

    /**
     * Get the output from the PID controller.
     * @param currentPosition the current position of the mechanism
     * @return the output to be sent to the motor(s)
     */
    public double getOutput(double currentPosition) {
        currentError = setpoint - currentPosition;
        sumError += currentError;

        double currentTime = opMode.getRuntime();
        double rateOfChange = (currentError - prevError) / (currentTime - prevTime);

        prevError = currentError;
```

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

```java
package org.firstinspires.ftc.teamcode.Util;

public class GlobalVariebels {

    public static int liftPosEndAuto = 0;

}

```

```java
package org.firstinspires.ftc.teamcode.Util;
import android.support.annotation.Nullable;

import com.qualcomm.robotcore.eventloop.opmode.Disabled;
import com.qualcomm.robotcore.eventloop.opmode.
LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.OpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.HardwareMap;

import org.firstinspires.ftc.robotcore.external.
ClassFactory;
import org.firstinspires.ftc.robotcore.external.navigation
.AngleUnit;
import org.firstinspires.ftc.robotcore.external.navigation
.VuforiaLocalizer;
import org.firstinspires.ftc.robotcore.external.navigation
.VuforiaLocalizer.CameraDirection;
import org.firstinspires.ftc.robotcore.external.tfod.
Recognition;
import org.firstinspires.ftc.robotcore.external.tfod.
TFObjectDetector;

import java.util.List;

import javax.net.ssl.HandshakeCompletedEvent;


public class GoldRecognation {
    public enum MineralPos{
        LEFT,
        CENTER,
        RIGHT,
        UNKNOWN;
    }

    private OpMode opMode;
    private HardwareMap hardwareMap;
    private DcMotor leds;
    /**
     * this parameters discribe our labels that we want to
  find
     */
    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+
   o7VusfYmY7FPvnXuhn8obCeB5x0hhjwjsBuOz2wnx1us6N5y6on0rdc1DO
   zC2gI767QLVXAvPyJvMfgtZRGcfzFk0evuSVxrIPCRQBjQYK2s5SsBLZ4s
   EO4HelibKK5kg0lgT9P1uHSNa8SWX+
   4wpJm76dFw1nSL7YElieByOTXxycmOdhWaae5qb1lvYYmDiBNFiwfHRDkB
   RD";
43
44     /**
45      * {@link #vuforia} is the variable we will use to
   store our instance of the Vuforia
46      * localization engine.
47      */
48     private VuforiaLocalizer vuforia;
49
50     /**
51      * {@link #tfod} is the variable we will use to store
   our instance of the Tensor Flow Object
52      * Detection engine.
53      */
54     public TFObjectDetector tfod;
55
56     public GoldRecognation(HardwareMap hardwareMap, OpMode
    opMode){
57         this.hardwareMap = hardwareMap;
58         this.opMode = opMode;
59         this.leds = hardwareMap.get(DcMotor.class, "leds")
   ;
60         initVuforia();
61         if (ClassFactory.getInstance().
   canCreateTFObjectDetector()) {
62             initTfod();
63         } else {
64             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
    object, and passing it to the Vuforia engine.
71           */
72          VuforiaLocalizer.Parameters parameters = new
    VuforiaLocalizer.Parameters();
73
74          parameters.vuforiaLicenseKey = VUFORIA_KEY;
75          parameters.cameraDirection = CameraDirection.
    FRONT;
76
77          //  Instantiate the Vuforia engine
78          vuforia = ClassFactory.getInstance().
    createVuforia(parameters);
79
80          // Loading trackables is not necessary for the
    Tensor Flow Object Detection engine.
81      }
82
83      /**
84       * Initialize the Tensor Flow Object Detection engine
    .
85       */
86      private void initTfod() {
87          int tfodMonitorViewId = hardwareMap.appContext.
    getResources().getIdentifier(
88                  "tfodMonitorViewId", "id", hardwareMap.
    appContext.getPackageName());
89          TFObjectDetector.Parameters tfodParameters = new
    TFObjectDetector.Parameters(tfodMonitorViewId);
90          tfod = ClassFactory.getInstance().
    createTFObjectDetector(tfodParameters, vuforia);
91          tfod.loadModelFromAsset(TFOD_MODEL_ASSET,
    LABEL_GOLD_MINERAL, LABEL_SILVER_MINERAL);
92      }
93      public void stopTfod(){
94          tfod.shutdown();
95      }
96
97      public void turnOnLeds() {
98          leds.setPower(light);
```

```
 99      }
100
101      public void turnOffLeds() {
102          leds.setPower(0);
103      }
104
105      public MineralPos getGoldPos(LogCreater 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
    () && opMode.getRuntime() < time + 5) {
114                  List<Recognition> recognitionList = tfod.
    getUpdatedRecognitions();
115                  if (recognitionList != null) {
116                      opMode.telemetry.addData("size",
    recognitionList.size());
117                      if (log != null) {
118                          log.writeLog("SamplingSize",
    recognitionList.size(), "");
119                          for (Recognition recognition :
    recognitionList) {
120                              log.writeLog("SamplingObject"
    , recognition.getTop(),
121                                      recognition.toString(
    ) + ", h: " + recognition.getHeight() + ", w: " +
    recognition.getWidth());
122                          }
123                      }
124                      if (recognitionList.size() == 2) {
125                          double goldTop = -1;
126                          double silverTop = -1;
127                          for (Recognition recognition :
    recognitionList) {
128                              if (recognition.getLabel() ==
     "Gold Mineral") {
129                                  goldTop = recognition.
    getTop();
130                              } else if (silverTop == -1) {
131                                  silverTop = recognition.
    getTop();
```

```
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
```

```java
package org.firstinspires.ftc.teamcode.Util;

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.OpMode;
import com.qualcomm.robotcore.util.Range;

public class TurnPIDController extends PIDController {
    /**
     * Instantiate a new PIDController.
     *
     * @param kP
     * @param kI
     * @param kD
     * @param tolerance
     * @param opmode
     */
    public TurnPIDController(double kP, double kI, double kD, double tolerance, OpMode opmode) {
        super(kP, kI, kD, tolerance, opmode);
    }

    @Override
    public double getOutput(double currentPosition) {
        currentError = setpoint - currentPosition;
        while (currentError > 180 && ((LinearOpMode)opMode).opModeIsActive()) currentError -= 360;
        while (currentError <= -179 && ((LinearOpMode)opMode).opModeIsActive()) currentError += 360;
        sumError += currentError;

        double currentTime = opMode.getRuntime();
        double rateOfChange = (currentError - prevError) / (currentTime - prevTime);

        prevError = currentError;
        prevTime = currentTime;
        return Range.clip(kP * currentError + kI * sumError + kD * rateOfChange, -1.0, 1.0);
    }

    @Override
    public void reset(double setpoint, double startingPos) {
        this.setpoint = setpoint;
```

```
39          running = true;
40
41          sumError = 0;
42          currentError = setpoint - startingPos;
43          while (currentError > 180 && ((LinearOpMode)opMode
    ).opModeIsActive()) currentError -= 360;
44          while (currentError <= -179 && ((LinearOpMode)
    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
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    SUCH DAMAGE.
```

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

```
28  */
29
30 package org.firstinspires.ftc.teamcode.TeleOps;
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.GoldRecognation
   ;
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 GoldRecognation goldRecognation;
49
50   @Override
51   public void init() {
52     goldRecognation = new GoldRecognation(hardwareMap,
   this);
53     telemetry.addData("Status", "Initialized");
54   }
55
56   @Override
57   public void init_loop() {
58   }
59
60   @Override
61   public void start() {
62     runtime.reset();
63   }
64
65   @Override
66   public void loop() {
67     telemetry.addData("Status", "Run Time: " + runtime.
   toString());
68     telemetry.addLine("turnOnLeds: a");
69     telemetry.addLine("turnOffLeds: b");
```

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

```
 1 /* Copyright (c) 2017 FIRST. All rights reserved.
 2  *
 3  * Redistribution and use in source and binary forms, with
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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 import org.firstinspires.ftc.teamcode.Util.LogCreater;
41
42 @Autonomous(name = "TurnTest", group = "Tests")
43 public class AutoTest extends LinearOpMode {
44
45
46   private Robot robot = new Robot();
47   private ElapsedTime runtime = new ElapsedTime();
48
49
50
51
52   @Override
53   public void runOpMode() throws InterruptedException {
54     robot.init(hardwareMap , this);
55     waitForStart();
56     robot.drive.turnByGyroAbsolut(100,5);
57       robot.drive.turnByGyroAbsolut(-170,5);
58   }
59 }
```

```
 1 /* Copyright (c) 2017 FIRST. All rights reserved.
 2  *
 3  * Redistribution and use in source and binary forms, with
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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.Drive;
38 import org.firstinspires.ftc.teamcode.RobotSystems.Robot;
39 import org.firstinspires.ftc.teamcode.Util.LogCreater;
40 import java.util.Date;
41 import java.text.SimpleDateFormat;
42
43 import java.util.Calendar;
44 @Autonomous(name = "TimeTest", group = "Auto")
45 @Disabled
46 public class TimeTest extends LinearOpMode {
47   SimpleDateFormat formatter = new SimpleDateFormat("_dd.
   MM.yyyy_HH:mm:ss");
48   @Override
49   public void runOpMode() throws InterruptedException {
50     waitForStart();
51     while (opModeIsActive()) {
52       Date date = new Date();
53       telemetry.addData("time: ", formatter.format(date));
54       telemetry.update();
55     }
56   }
57 }
58
```

```
 1 /* Copyright (c) 2017 FIRST. All rights reserved.
 2  *
 3  * Redistribution and use in source and binary forms, with
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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.robotcore.external.tfod.
   Recognition;
38 import org.firstinspires.ftc.teamcode.Util.GoldRecognation
   ;
39 import org.firstinspires.ftc.teamcode.RobotSystems.Robot;
40
41 import java.util.List;
42
43 @Autonomous(name = "AutoPhoneCamera", group = "Tests")
44
45 public class AutoPhoneCamera extends LinearOpMode {
46   protected Robot robot = new Robot();
47   private GoldRecognation recognation = null;
48   private ElapsedTime runtime = new ElapsedTime();
49
50   @Override
51   public void runOpMode() throws InterruptedException {
52     recognation = new GoldRecognation(hardwareMap, this);
53     recognation.tfod.activate();
54     waitForStart();
55     recognation.turnOnLeds();
56     while (opModeIsActive()) {
57       GoldRecognation.MineralPos mineralPos = recognation.
   getGoldPos(null);
58       telemetry.addData(" mineral", mineralPos);
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
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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.
   LinearOpMode;
34 import com.qualcomm.robotcore.util.ElapsedTime;
35
36 import org.firstinspires.ftc.teamcode.RobotSystems.
   Climbing;
37 import org.firstinspires.ftc.teamcode.RobotSystems.Drive;
38 import org.firstinspires.ftc.teamcode.RobotSystems.Robot;
39 import org.firstinspires.ftc.teamcode.Util.GlobalVariebels
   ;
40 import org.firstinspires.ftc.teamcode.Util.GoldRecognation
   ;
41 import org.firstinspires.ftc.teamcode.Util.LogCreater;
42
43 /**
44  * doing Landing,Sampling and Team Marker
45  * PTS = 70
46  * Starting from Depot On The Lander
47  * FIRST Autonomous
48  */
49 @Autonomous(name = "Depot", group = "Auto")
50 public class Depot extends LinearOpMode {
51
52     protected Robot robot = new Robot();
53     protected ElapsedTime runtime = new ElapsedTime();
54     protected LogCreater log = new LogCreater("auto");
55
56     public void endAuto(GoldRecognation.MineralPos goldPos
   ) {
57         if (goldPos == GoldRecognation.MineralPos.LEFT) {
58             robot.drive.turnByGyroAbsolut(-45, 10);
59             robot.drive.driveByEncoder(100, 0.5, Drive.
   Direction.FORWARD, 3000);
60             robot.climbing.moveAngleAuto(Climbing.Angle.
   COLLECT);
61             robot.intake.collect();
62             robot.climbing.moveLiftAuto(Climbing.Height.
   PUT);
63             robot.climbing.moveLiftAuto(Climbing.Height.
   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.turnByGyroAbsolut(-9, 10);
77          GoldRecognation.MineralPos goldPos = robot.drive.
   Sampling(Drive.Side.DEPOT);
78          robot.intake.injackt();
79          sleep(2000);
80          robot.intake.stop();
81          endAuto(goldPos);
82          GlobalVariebels.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
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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.
   LinearOpMode;
34 import com.qualcomm.robotcore.util.ElapsedTime;
35
36 import org.firstinspires.ftc.teamcode.RobotSystems.
   Climbing;
37 import org.firstinspires.ftc.teamcode.RobotSystems.Drive;
38 import org.firstinspires.ftc.teamcode.RobotSystems.Robot;
39 import org.firstinspires.ftc.teamcode.Util.GoldRecognation
   ;
40 import org.firstinspires.ftc.teamcode.Util.LogCreater;
41
42 /**
43  * doing Landing, Team Marker and Parking
44  * PTS = 50
45  * In Case of no Sampling
46  * Starting from Depot On The Lander
47  * FIRST Autonomous
48  */
49 @Autonomous(name = "DepotToCreater", group = "Auto")
50 public class DepotToCreater extends Depot {
51
52   @Override
53   public void endAuto(GoldRecognation.MineralPos goldPos)
   {
54     double driveDis = 0, turnAngle = 0;
55     switch (goldPos) {
56       case RIGHT:
57         driveDis = 130;
58         turnAngle = -60;
59         robot.drive.driveByEncoder(15,0.3, Drive.Direction
   .BACKWARD, 2);
60         break;
61       case UNKNOWN:
62       case CENTER:
63         driveDis = 110;
64         turnAngle = -60;
65         robot.drive.driveByEncoder(8,0.3, Drive.Direction.
   BACKWARD, 2);
66         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
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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.
   LinearOpMode;
34 import com.qualcomm.robotcore.util.ElapsedTime;
35
36 import org.firstinspires.ftc.teamcode.RobotSystems.
   Climbing;
37 import org.firstinspires.ftc.teamcode.RobotSystems.Drive;
38 import org.firstinspires.ftc.teamcode.RobotSystems.Robot;
39 import org.firstinspires.ftc.teamcode.Util.GlobalVariebels
   ;
40 import org.firstinspires.ftc.teamcode.Util.GoldRecognation
   ;
41 import org.firstinspires.ftc.teamcode.Util.LogCreater;
42
43 /**
44  * doing Landing, Sampling and Parking.
45  * PTS=65
46  * Starting from Creater
47  */
48 @Autonomous(name = "Creater", group = "Tests")
49 public class Creater extends LinearOpMode {
50   protected Robot robot = new Robot();
51   protected ElapsedTime runtime = new ElapsedTime();
52   protected LogCreater log = new LogCreater("auto");
53
54   @Override
55   public void runOpMode() throws InterruptedException {
56     log.init(this);
57     robot.init(hardwareMap , this, log);
58     waitForStart();
59     robot.climbing.land();
60     robot.drive.turnByGyroAbsolut(-9, 10);
61     GoldRecognation.MineralPos goldPos = robot.drive.
   Sampling(Drive.Side.CREATER);
62     goToDepot(goldPos);
63     goToCreater(goldPos);
64     GlobalVariebels.liftPosEndAuto = robot.climbing.
   liftMotor.getCurrentPosition();
65
66   }
```

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

```
 1 /* Copyright (c) 2017 FIRST. All rights reserved.
 2  *
 3  * Redistribution and use in source and binary forms, with
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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.
   Climbing;
35 import org.firstinspires.ftc.teamcode.RobotSystems.Drive;
36 import org.firstinspires.ftc.teamcode.RobotSystems.Robot;
37 import org.firstinspires.ftc.teamcode.Util.GoldRecognation
   ;
38
39 @Autonomous(name = "CreaterToDepot", group = "Tests")
40 public class CreaterToDepot extends Creater {
41     @Override
42     public void goToDepot(GoldRecognation.MineralPos
   goldPos) {
43         /*
44         if (goldPos == GoldRecognation.MineralPos.RIGHT
    || goldPos == GoldRecognation.MineralPos.LEFT) {
45             robot.drive.driveByEncoder(70, 0.3, Drive.
   Direction.FORWARD, 3000);
46         }
47         else {
48             robot.drive.driveByEncoder(55, 0.3, Drive.
   Direction.FORWARD, 3000);
49         }
50         robot.drive.turnByGyroAbsolut(55, 10);
51         robot.drive.driveByEncoder(135, 0.3, Drive.
   Direction.BACKWARD, 3000);
52         robot.drive.turnByGyroAbsolut(133, 10);
53         robot.drive.driveByEncoder(150, 0.5, Drive.
   Direction.BACKWARD, 3000);
54         //robot.climbing.moveLiftAuto(Climbing.Height.PUT
   );
55         */
56         switch (goldPos){
57             case LEFT:
58                 robot.drive.curvedDrive(60,3,0.5, Drive.
   Direction.FORWARD, Drive.CurvedDirection.LEFT);
59                 robot.drive.driveByEncoder(60,0.4, Drive.
   Direction.BACKWARD,5);
60                 robot.drive.turnByGyroAbsolut(130, 5);
61                 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
   .Direction.FORWARD, Drive.CurvedDirection.LEFT);
66                 robot.drive.turnByGyroAbsolut(90, 3);
67                 robot.drive.driveByEncoder(100,0.4, Drive
   .Direction.BACKWARD,5);
68                 robot.drive.turnByGyroAbsolut(130, 5);
69                 robot.drive.driveByEncoder(70,0.5, Drive.
   Direction.BACKWARD,6);
70                 break;
71             case RIGHT:
72                 robot.drive.driveByEncoder(10, 0.5, Drive
   .Direction.FORWARD, 5);
73                 robot.drive.curvedDrive(100,6,0.5, Drive.
   Direction.FORWARD, Drive.CurvedDirection.RIGHT);
74                 robot.drive.driveByEncoder(60,0.4, Drive.
   Direction.FORWARD,5);
75                 robot.drive.turnByGyroAbsolut(140, 3);
76                 robot.drive.driveByEncoder(80,0.5, Drive.
   Direction.BACKWARD,6);
77                 break;
78         }
79         robot.drive.turnByGyroRelative(10,3);
80         robot.intake.injackt();
81         sleep(2000);
82         robot.intake.stop();
83         robot.drive.turnByGyroAbsolut(160, 1);
84     }
85
86     @Override
87     public void goToCreater(GoldRecognation.MineralPos
   goldPos) {
88         robot.drive.driveByEncoder(110, 1, Drive.
   Direction.FORWARD, 3);
89         robot.climbing.moveAngleAuto(Climbing.Angle.
   COLLECT);
90         robot.intake.collect();
91         robot.climbing.moveLiftAuto(Climbing.Height.PUT);
92         robot.climbing.moveLiftAuto(Climbing.Height.
   COLLECT);
93         robot.climbing.moveLiftAuto(Climbing.Height.PUT);
94     }
```

```
95
96 }
```

```
 1 /* Copyright (c) 2017 FIRST. All rights reserved.
 2  *
 3  * Redistribution and use in source and binary forms, with
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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.LogCreater;
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 LogCreater log = new LogCreater("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
   here
59       * @see com.qualcomm.robotcore.eventloop.opmode.OpMode
   #start()
60       */
61   @Override
62   public void init_loop() {
63   }
64
65   /*
66    * This method will be called ONCE when start is pressed
67    * @see com.qualcomm.robotcore.eventloop.opmode.OpMode#
   loop()
68    */
69   @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#
   loop()
77     */
78    @Override
79    public void loop() {
80      telemetry.addData("Status", "Run Time: " + runtime.
   toString());
81      telemetry.log().add("loop: " + runtime.seconds());
82      log.writeLog("test", 0, "");
83    }
84  }
85
```

```
 1 /* Copyright (c) 2018 FIRST. All rights reserved.
 2  *
 3  * Redistribution and use in source and binary forms, with
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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) *
     mmPerInch;       // the width of the FTC field (from the
     center point to the outer panels)
60      private static final float mmTargetHeight = (6) *
    mmPerInch;          // the height of the center of the
    target image above the floor
61
62      private static final float
    CAMERA_FORWARD_DISPLACEMENT  = 0;
63      private static final float
    CAMERA_VERTICAL_DISPLACEMENT = mmTargetHeight;
64      private static final float CAMERA_LEFT_DISPLACEMENT
    = 0;
65      private static final float CAMERA_LONG_AXIS_ROT = 90;
66      private static final float CAMERA_SHORT_AXIS_ROT = 0;
67      private static final float CAMERA_DEPTH_AXIS_ROT = -
    90;
68
69      private VuforiaLocalizer vuforia;
70      private List<VuforiaTrackable> allTrackables = new
    ArrayList<VuforiaTrackable>();
71      private OpenGLMatrix lastPosition = null;
72
73      public BTVuforia() {}
74
75      public void init(HardwareMap hardwareMap)
76      {
77          int cameraMonitorViewId = hardwareMap.appContext.
    getResources().getIdentifier("cameraMonitorViewId", "id",
     hardwareMap.appContext.getPackageName());
78          VuforiaLocalizer.Parameters parameters = new
    VuforiaLocalizer.Parameters(cameraMonitorViewId);
79          //VuforiaLocalizer.Parameters parameters = new
    VuforiaLocalizer.Parameters();
80          parameters.vuforiaLicenseKey = VUFORIA_KEY;
81          parameters.cameraDirection = FRONT;
82          vuforia = ClassFactory.getInstance().
    createVuforia(parameters);
83
84          // Load the data sets that for the trackable
    objects. These particular data
85          // sets are stored in the 'assets' part of our
    application.
86          VuforiaTrackables targetsRoverRuckus = this.
```

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

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

```
146                     lastPosition = robotLocationTransform;
147                 }
148             }
149             return lastPosition;
150         }
151 
152         //returns visible picture name
153         public String getVisableTargetName()
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
    (getRobotPosition(), EXTRINSIC, XYZ, DEGREES);
189             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().
    getTranslation().get(1);
209                     break;
210
211                 case "Front-Craters":
212                 case "Back-Space":
213                     robotPos = getXpos();
214                     picPos = trackable.getLocation().
    getTranslation().get(0);
215                     break;
216             }
217             if (robotPos != null) {
218                 return Math.abs(robotPos - picPos);
219             }
220         }
221         return  null;
222     }
223
224     @Override
225     public String toString() {
226         OpenGLMatrix position = getRobotPosition();
227         if(position != null)
228         {
229             VectorF translation = position.getTranslation
    ();
230             Orientation rotation = Orientation.
    getOrientation(position, EXTRINSIC, XYZ, DEGREES);
```

```
231
232             return "Visible Target: " +
    getVisableTargetName() + "\n" +
233                     "distance from visible wall: " +
    getDistanceFromWall() + "\n" +
234                     "Pos (mm) {X, Y, Z} = " + translation
    .get(0) + translation.get(1) + translation.get(2) + "\n "
     +
235                     "Rot (deg) {Roll, Pitch, Heading} = "
     + Math.round(rotation.firstAngle) + ", " + Math.round(
    rotation.secondAngle) + ", " + Math.round(rotation.
    thirdAngle);
236         }
237         else
238         {
239             return "no target is visible";
240         }
241     }
242 }
243
```

```
 1 /* Copyright (c) 2017 FIRST. All rights reserved.
 2  *
 3  * Redistribution and use in source and binary forms, with
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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.DcMotor;
36 import com.qualcomm.robotcore.util.ElapsedTime;
37 import com.qualcomm.robotcore.util.Range;
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
   reversed to drive forward
63         // Reverse the motor that runs backwards when
   connected directly to the battery
64         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
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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
36 import org.firstinspires.ftc.robotcore.external.
   ClassFactory;
37 import org.firstinspires.ftc.robotcore.external.navigation
   .VuforiaLocalizer;
38 import org.firstinspires.ftc.robotcore.external.navigation
   .VuforiaLocalizer.CameraDirection;
39 import org.firstinspires.ftc.robotcore.external.tfod.
   Recognition;
40 import org.firstinspires.ftc.robotcore.external.tfod.
   TFObjectDetector;
41
42 import java.util.List;
43
44 /**
45  * This 2018-2019 OpMode illustrates the basics of using
   the TensorFlow Object Detection API to
46  * determine the position of the gold and silver minerals.
47  *
48  * Use Android Studio to Copy this Class, and Paste it
   into your team's code folder with a new name.
49  * Remove or comment out the @Disabled line to add this
   opmode to the Driver Station OpMode list.
50  *
51  * IMPORTANT: In order to use this OpMode, you need to
   obtain your own Vuforia license key as
52  * is explained below.
53  */
54 @TeleOp(name = "Concept: TensorFlow Object Detection",
   group = "Concept")
55 @Disabled
56 public class UVC_Camera extends LinearOpMode {
57     private static final String TFOD_MODEL_ASSET = "
   RoverRuckus.tflite";
58     private static final String LABEL_GOLD_MINERAL = "Gold
    Mineral";
59     private static final String LABEL_SILVER_MINERAL = "
```

```
59 Silver Mineral";
60
61     /*
62      * IMPORTANT: You need to obtain your own license key to use Vuforia. The string below with which
63      * 'parameters.vuforiaLicenseKey' is initialized is for illustration only, and will not function.
64      * A Vuforia 'Development' license key, can be obtained free of charge from the Vuforia developer
65      * web site at https://developer.vuforia.com/license-manager.
66      *
67      * Vuforia license keys are always 380 characters long, and look as if they contain mostly
68      * random data. As an example, here is a example of a fragment of a valid key:
69      *      ...
yIgIzTqZ4mWjk9wd3cZO9T1axEqzuhxoGlfOOI2dRzKS4T0hQ8kT ...
70      * Once you've obtained a license key, copy the string from the Vuforia web site
71      * and paste it in to your code on the next line, between the double quotes.
72      */
73     private static final String VUFORIA_KEY = "AW/F0cP
/////AAABmUpR4+
dbt0Negw2nqaCH9Cw2gV4ZxuUmpeJMm7XOTdQVumthQcOeoS9qktHy4Ev
XtMAFoh7n5KeMiLMDqtKvd1TrbYNUy3f9ST3TMkH2hFYKB6oJMJPB8oel
L9Bst/2XJBz0ycMMcKmSsIwyOqwOuHamAlwfT+
o7VusfYmY7FPvnXuhn8obCeB5x0hhjwjsBuOz2wnx1us6N5y6on0rdc1D
OzC2gI767QLVXAvPyJvMfgtZRGcfzFk0evuSVxrIPCRQBjQYK2s5SsBLZ
4sEO4HelibKK5kg0lgT9P1uHSNa8SWX+
4wpJm76dFw1nSL7YElieByOTXxycmOdhWaae5qb1lvYYmDiBNFiwfHRDk
BRD";
74
75     /**
76      * {@link #vuforia} is the variable we will use to store our instance of the Vuforia
77      * localization engine.
78      */
79     private VuforiaLocalizer vuforia;
80
81     /**
82      * {@link #tfod} is the variable we will use to store our instance of the Tensor Flow Object
83      * Detection engine.
```

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

```
121                         for (Recognition recognition :
    updatedRecognitions) {
122                           if (recognition.getLabel().
    equals(LABEL_GOLD_MINERAL)) {
123                             goldMineralX = (int)
    recognition.getLeft();
124                           } else if (silverMineral1X == -
    1) {
125                             silverMineral1X = (int)
    recognition.getLeft();
126                           } else {
127                             silverMineral2X = (int)
    recognition.getLeft();
128                           }
129                         }
130                           telemetry.addData("Gold Mineral
     Position", goldMineralX);
131                         if (goldMineralX != -1 &&
    silverMineral1X != -1 && silverMineral2X != -1) {
132                           if (goldMineralX <
    silverMineral1X && goldMineralX < silverMineral2X) {
133                             telemetry.addData("Gold
    Mineral Position", "Left");
134                           } else if (goldMineralX >
    silverMineral1X && goldMineralX > silverMineral2X) {
135                             telemetry.addData("Gold
    Mineral Position", "Right");
136                           } else {
137                             telemetry.addData("Gold
    Mineral Position", "Center");
138                           }
139                         }
140                       }
141                       telemetry.update();
142                     }
143                   }
144               }
145           }
146
147           if (tfod != null) {
148               tfod.shutdown();
149           }
150       }
151
152       /**
```

```
153      * Initialize the Vuforia localization engine.
154      */
155     private void initVuforia() {
156         /*
157          * Configure Vuforia by creating a Parameter
    object, and passing it to the Vuforia engine.
158          */
159         VuforiaLocalizer.Parameters parameters = new
    VuforiaLocalizer.Parameters();
160 
161         parameters.vuforiaLicenseKey = VUFORIA_KEY;
162         parameters.cameraDirection = CameraDirection.
    FRONT;
163 
164         //  Instantiate the Vuforia engine
165         vuforia = ClassFactory.getInstance().
    createVuforia(parameters);
166 
167         // Loading trackables is not necessary for the
    Tensor Flow Object Detection engine.
168     }
169 
170     /**
171      * Initialize the Tensor Flow Object Detection engine
    .
172      */
173     private void initTfod() {
174         int tfodMonitorViewId = hardwareMap.appContext.
    getResources().getIdentifier(
175             "tfodMonitorViewId", "id", hardwareMap.
    appContext.getPackageName());
176         TFObjectDetector.Parameters tfodParameters = new
    TFObjectDetector.Parameters(tfodMonitorViewId);
177         tfod = ClassFactory.getInstance().
    createTFObjectDetector(tfodParameters, vuforia);
178         tfod.loadModelFromAsset(TFOD_MODEL_ASSET,
    LABEL_GOLD_MINERAL, LABEL_SILVER_MINERAL);
179     }
180 }
181 
```

```
 1 /* Copyright (c) 2017 FIRST. All rights reserved.
 2  *
 3  * Redistribution and use in source and binary forms, with
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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.
   LinearOpMode;
41 import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
42 import com.qualcomm.robotcore.hardware.ColorSensor;
43 import com.qualcomm.robotcore.hardware.DcMotor;
44 import com.qualcomm.robotcore.hardware.
   NormalizedColorSensor;
45 import com.qualcomm.robotcore.hardware.NormalizedRGBA;
46 import com.qualcomm.robotcore.hardware.Servo;
47 import com.qualcomm.robotcore.hardware.SwitchableLight;
48
49 import java.io.BufferedReader;
50 import java.io.FileNotFoundException;
51 import java.io.InputStreamReader;
52 import java.io.OutputStreamWriter;
53
54
55 @TeleOp(name = "Sensor: Color", group = "Sensor")
56 @Disabled
57 public class SensorColor extends LinearOpMode {
58
59   ColorSensor colorSensor;
60   NormalizedColorSensor normalizedColorSensor;
61   private final String FILE_PATH = "org/firstinspires/ftc/
   teamcode/RGB_Test.txt";
62   final double SCALE_FACTOR = 255;
63   private java.io.File file = new java.io.File(FILE_PATH);
64   FileOutputStream fileOutputStream;
65   PrintStream printStream;
66   View relativeLayout;
67   private DcMotor collectMotor;
68   private Servo servo;
69   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
    later change the background
 76     // color of the Robot Controller app to match the hue
     detected by the RGB sensor.
 77     int relativeLayoutId = hardwareMap.appContext.
    getResources().getIdentifier("RelativeLayout", "id",
    hardwareMap.appContext.getPackageName());
 78     relativeLayout = ((Activity) hardwareMap.appContext).
    findViewById(relativeLayoutId);
 79     colorSensor = hardwareMap.get(ColorSensor.class, "
    color_sensor");
 80     collectMotor = hardwareMap.get(DcMotor.class, "
    collectMotor");
 81     collectMotor.setDirection(DcMotor.Direction.FORWARD);
 82     normalizedColorSensor = hardwareMap.get(
    NormalizedColorSensor.class, "color_sensor");
 83     servo = hardwareMap.get(Servo.class, "Servo");
 84     servo.setPosition(0.20);
 85
 86     waitForStart();
 87     try {
 88       File file = new File("/sdcard/FIRST/RGB_Test.txt");
 89       fileOutputStream = new FileOutputStream(file);
 90       printStream = new PrintStream(fileOutputStream);
 91
 92       printStream.write("Time | Hue | S_Value | Value \n"
    .getBytes());
 93
 94       while (opModeIsActive()) {
 95         collectMotor.setPower(0.8);
 96         NormalizedRGBA colors = normalizedColorSensor.
    getNormalizedColors();
 97
 98         Color.RGBToHSV((int) (colorSensor.red() *
    SCALE_FACTOR),
 99                 (int) (colorSensor.green() * SCALE_FACTOR
    ),
100                 (int) (colorSensor.blue() * SCALE_FACTOR)
    ,
101                 hsvValues);
```

```
102
103             printStream.write((((String) (time + "|" +
    hsvValues[0] + "| " + hsvValues[1] + "| "  + hsvValues[2]
    ) + "\n")).getBytes());
104             telemetry.addLine(time + "|" + colors.red + "|" +
     colors.green + "|" + colors.blue + "\n");
105             telemetry.addLine("Red : " + colorSensor.red() +
    "| green : " + colorSensor.green() + "| blue : "  +
    colorSensor.blue());
106             telemetry.addLine("Hue : " + hsvValues[0] + "| s
     : " + hsvValues[1] + "| value : "  + hsvValues[2]);
107             telemetry.update();
108
109            sleep(200);
110           }
111
112           collectMotor.setPower(0);
113           printStream.close();
114           fileOutputStream.close();
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
    previous and current state of the button
136         boolean bPrevState = false;
137         boolean bCurrState = false;
138
```

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

```
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
    getColors() are normalized relative to the
198         // maximum possible values that the sensor can
    measure. For example, a sensor might in a
199         // particular configuration be able to internally
    measure color intensity in a range of
200         // [0, 10240]. In such a case, the values returned
    by getColors() will be divided by 10240
201         // so as to return a value it the range [0,1].
    However, and this is the point, even so, the
202         // values we see here may not get close to 1.0 in,
    e.g., low light conditions where the
203         // sensor measurements don't approach their maximum
     limit. In such situations, the *relative*
204         // intensities of the colors are likely what is
    most interesting. Here, for example, we boost
205         // the signal on the colors while maintaining their
     relative balance so as to give more
206         // vibrant visual feedback on the robot controller
    visual display.
207         float max = Math.max(Math.max(Math.max(colors.red,
    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
    ), Color.blue(color), hsvValues);
222
223         // change the background color to match the color
    detected by the RGB sensor.
224         // pass a reference to the hue, saturation, and
    value array as an argument
225         // to the HSVToColor method.
226         relativeLayout.post(new Runnable() {
227           public void run() {
228             relativeLayout.setBackgroundColor(Color.
    HSVToColor(0xff, values));
229           }
230         });
231         */
232       }
233     }
234 }
235
236
```

```
 1 /* Copyright (c) 2017 FIRST. All rights reserved.
 2  *
 3  * Redistribution and use in source and binary forms, with
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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
   pressed
78    * @see com.qualcomm.robotcore.eventloop.opmode.OpMode#
   loop()
79    */
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#
   loop()
91    */
92   @Override
93   public void loop() {
94     telemetry.addData("Status", "Run Time: " + runtime.
   toString());
95   }
96 }
97
```

```
 1 /* Copyright (c) 2017 FIRST. All rights reserved.
 2  *
 3  * Redistribution and use in source and binary forms, with
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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.
   LinearOpMode;
35 import com.qualcomm.robotcore.eventloop.opmode.OpMode;
36 import com.qualcomm.robotcore.hardware.DcMotor;
37 import com.qualcomm.robotcore.hardware.HardwareMap;
38 import com.qualcomm.robotcore.util.ElapsedTime;
39
40 import org.firstinspires.ftc.robotcontroller.external.
   samples.HardwarePushbot;
41
42 @Autonomous(name="moveMotorByTicks", group="Pushbot")
43 @Disabled
44 public class MoveMotorByTicks extends LinearOpMode {
45
46     HardwarePushbot         robot   = new HardwarePushbot(
   );
47     private ElapsedTime     runtime = new ElapsedTime();
48
49     static final double     COUNTS_PER_MOTOR_REV    = 1440
    ;    // eg: TETRIX Motor Encoder
50     static final double     DRIVE_GEAR_REDUCTION    = 2.0
   ;     // This is < 1.0 if geared UP
51     static final double     WHEEL_DIAMETER_INCHES   = 4.0
   ;     // For figuring circumference
52     static final double     COUNTS_PER_CM         = (
   COUNTS_PER_MOTOR_REV * DRIVE_GEAR_REDUCTION) /
53             (WHEEL_DIAMETER_INCHES * 3.1415);
54     static final double     DRIVE_SPEED             = 0.6;
55     static final double     TURN_SPEED              = 0.5;
56
57     private DcMotor motor;
58
59     public void init(HardwareMap hardwareMap) {
60         motor = hardwareMap.get(DcMotor.class, "Drive");
61
62         motor.setDirection(DcMotor.Direction.FORWARD);
63
64         motor.setMode(DcMotor.RunMode.
   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          telemetry.update();
 76
 77          motor.setMode(DcMotor.RunMode.
    STOP_AND_RESET_ENCODER);
 78          motor.setMode(DcMotor.RunMode.RUN_USING_ENCODER);
 79
 80          telemetry.addLine("init complete");
 81          telemetry.update();
 82          waitForStart();
 83              telemetry.addLine("Start");
 84              telemetry.update();
 85              driveByTicks(30, 0.5);
 86
 87              sleep(1000);     // pause for servos to move
 88          telemetry.addData("Path", "Complete");
 89          telemetry.update();
 90      }
 91
 92      public void driveByTicks(int motorTicks, double power
    ) {
 93
 94          motor.setTargetPosition(motorTicks);
 95
 96          motor.setMode(DcMotor.RunMode.RUN_TO_POSITION);
 97          motor.setPower(power);
 98          while (motor.isBusy()) {
 99              telemetry.addLine("ticks " + motor.
    getCurrentPosition());
100              telemetry.update();
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
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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.
   LinearOpMode;
35
36 import org.firstinspires.ftc.robotcore.external.
   ClassFactory;
37 import org.firstinspires.ftc.robotcore.external.matrices.
   OpenGLMatrix;
38 import org.firstinspires.ftc.robotcore.external.matrices.
   VectorF;
39 import org.firstinspires.ftc.robotcore.external.navigation
   .Orientation;
40 import org.firstinspires.ftc.robotcore.external.navigation
   .VuforiaLocalizer;
41 import org.firstinspires.ftc.robotcore.external.navigation
   .VuforiaTrackable;
42 import org.firstinspires.ftc.robotcore.external.navigation
   .VuforiaTrackableDefaultListener;
43 import org.firstinspires.ftc.robotcore.external.navigation
   .VuforiaTrackables;
44
45 import java.util.ArrayList;
46 import java.util.List;
47
48 import static org.firstinspires.ftc.robotcore.external.
   navigation.AngleUnit.DEGREES;
49 import static org.firstinspires.ftc.robotcore.external.
   navigation.AxesOrder.XYZ;
50 import static org.firstinspires.ftc.robotcore.external.
   navigation.AxesOrder.YZX;
51 import static org.firstinspires.ftc.robotcore.external.
   navigation.AxesReference.EXTRINSIC;
52 import static org.firstinspires.ftc.robotcore.external.
   navigation.VuforiaLocalizer.CameraDirection.BACK;
53 import static org.firstinspires.ftc.robotcore.external.
   navigation.VuforiaLocalizer.CameraDirection.FRONT;
54
55
56 /**
57  * 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
   surroundings, and attempt to locate target images.
62  *
63  * When images are located, Vuforia is able to determine
   the position and orientation of the
64  * image relative to the camera.  This sample code than
   combines that information with a
65  * knowledge of where the target images are on the field,
    to determine the location of the camera.
66  *
67  * This example assumes a "square" field configuration
   where the red and blue alliance stations
68  * are on opposite walls of each other.
69  *
70  * From the Audience perspective, the Red Alliance
   station is on the right and the
71  * Blue Alliance Station is on the left.
72
73  * The four vision targets are located in the center of
   each of the perimeter walls with
74  * the images facing inwards towards the robots:
75  *     - BlueRover is the Mars Rover image target on the
   wall closest to the blue alliance
76  *     - RedFootprint is the Lunar Footprint target on
   the wall closest to the red alliance
77  *     - FrontCraters is the Lunar Craters image target
   on the wall closest to the audience
78  *     - BackSpace is the Deep Space image target on the
   wall farthest from the audience
79  *
80  * A final calculation then uses the location of the
   camera on the robot to determine the
81  * robot's location and orientation on the field.
82  *
83  * @see VuforiaLocalizer
84  * @see VuforiaTrackableDefaultListener
85  * see  ftc_app/doc/tutorial/
   FTC_FieldCoordinateSystemDefinition.pdf
86  *
87  * Use Android Studio to Copy this Class, and Paste it
   into your team's code folder with a new name.
```

```
 88  * Remove or comment out the @Disabled line to add this
    opmode to the Driver Station OpMode list.
 89  *
 90  * IMPORTANT: In order to use this OpMode, you need to
    obtain your own Vuforia license key as
 91  * is explained below.
 92  */
 93
 94 @Autonomous(name="vuforiaTest", group ="Concept")
 95 @Disabled
 96 public class vuforiaCameraTest extends LinearOpMode {
 97
 98     /*
 99      * IMPORTANT: You need to obtain your own license key
     to use Vuforia. The string below with which
100      * 'parameters.vuforiaLicenseKey' is initialized is
    for illustration only, and will not function.
101      * A Vuforia 'Development' license key, can be
    obtained free of charge from the Vuforia developer
102      * web site at https://developer.vuforia.com/license-
    manager.
103      *
104      * Vuforia license keys are always 380 characters
    long, and look as if they contain mostly
105      * random data. As an example, here is a example of a
     fragment of a valid key:
106      *      ...
    yIgIzTqZ4mWjk9wd3cZO9T1axEqzuhxoGlfOOI2dRzKS4T0hQ8kT ...
107      * Once you've obtained a license key, copy the
    string from the Vuforia web site
108      * and paste it in to your code on the next line,
    between the double quotes.
109      */
110     private static final String VUFORIA_KEY = "AW/F0cP
    /////AAABmUpR4+
    dbt0Negw2nqaCH9Cw2gV4ZxuUmpeJMm7XOTdQVumthQcOeoS9qktHy4Ev
    XtMAFoh7n5KeMiLMDqtKvd1TrbYNUy3f9ST3TMkH2hFYKB6oJMJPB8oel
    L9Bst/2XJBz0ycMMcKmSsIwyOqwOuHamAlwfT+
    o7VusfYmY7FPvnXuhn8obCeB5x0hhjwjsBuOz2wnx1us6N5y6on0rdc1D
    OzC2gI767QLVXAvPyJvMfgtZRGcfzFk0evuSVxrIPCRQBjQYK2s5SsBLZ
    4sEO4HelibKK5kg0lgT9P1uHSNa8SWX+
    4wpJm76dFw1nSL7YElieByOTXxycmOdhWaae5qb1lvYYmDiBNFiwfHRDk
    BRD";
111
112     // Since ImageTarget trackables use mm to specifiy
```

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

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

```
170           * of the {@link OpenGLMatrix} class.
171           *
172           * If you are standing in the Red Alliance
    Station looking towards the center of the field,
173           *     - The X axis runs from your left to the
    right. (positive from the center to the right)
174           *     - The Y axis runs from the Red Alliance
    Station towards the other side of the field
175           *       where the Blue Alliance Station is. (
    Positive is from the center, towards the BlueAlliance
    station)
176           *     - The Z axis runs from the floor, upwards
    towards the ceiling.  (Positive is above the floor)
177           *
178           * This Rover Ruckus sample places a specific
    target in the middle of each perimeter wall.
179           *
180           * Before being transformed, each target image is
     conceptually located at the origin of the field's
181           *  coordinate system (the center of the field),
    facing up.
182           */
183
184          /**
185           * To place the BlueRover target in the middle of
     the blue perimeter wall:
186           * - First we rotate it 90 around the field's X
    axis to flip it upright.
187           * - Then, we translate it along the Y axis to
    the blue perimeter wall.
188           */
189          OpenGLMatrix blueRoverLocationOnField =
    OpenGLMatrix
190                  .translation(0, mmFTCFieldWidth,
    mmTargetHeight)
191                  .multiplied(Orientation.getRotationMatrix
    (EXTRINSIC, XYZ, DEGREES, 90, 0, 0));
192          blueRover.setLocation(blueRoverLocationOnField);
193
194          /**
195           * To place the RedFootprint target in the middle
     of the red perimeter wall:
196           * - First we rotate it 90 around the field's X
    axis to flip it upright.
197           * - 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
    the back perimeter wall.
224          */
225         OpenGLMatrix backSpaceLocationOnField =
    OpenGLMatrix
226                 .translation(mmFTCFieldWidth, 0,
    mmTargetHeight)
227                 .multiplied(Orientation.getRotationMatrix
    (EXTRINSIC, XYZ, DEGREES, 90, 0, -90));
228         backSpace.setLocation(backSpaceLocationOnField);
229
230         /**
231          * Create a transformation matrix describing
    where the phone is on the robot.
232          *
233          * The coordinate frame for the robot looks the
    same as the field.
234          * The robot's "forward" direction is facing out
    along X axis, with the LEFT side facing out along the Y
    axis.
235          * Z is UP on the robot.  This equates to a
    bearing angle of Zero degrees.
236          *
237          * The phone starts out lying flat, with the
    screen facing Up and with the physical top of the phone
238          * pointing to the LEFT side of the Robot.  It's
    very important when you test this code that the top of
    the
239          * camera is pointing to the left side of the
    robot.  The rotation angles don't work if you flip the
    phone.
240          *
241          * If using the rear (High Res) camera:
242          * We need to rotate the camera around it's long
    axis to bring the rear camera forward.
243          * This requires a negative 90 degree rotation on
     the Y axis
244          *
245          * If using the Front (Low Res) camera
246          * We need to rotate the camera around it's long
    axis to bring the FRONT camera forward.
247          * This requires a Positive 90 degree rotation on
     the Y axis
248          *
```

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

```
280             for (VuforiaTrackable trackable :
    allTrackables) {
281                 if (((VuforiaTrackableDefaultListener)
    trackable.getListener()).isVisible()) {
282                     telemetry.addData("Visible Target",
    trackable.getName());
283                     targetVisible = true;
284 
285                     // getUpdatedRobotLocation() will
    return null if no new information is available since
286                     // the last time that call was made,
    or if the trackable is not currently visible.
287                     OpenGLMatrix robotLocationTransform =
     ((VuforiaTrackableDefaultListener)trackable.getListener(
    )).getUpdatedRobotLocation();
288                     if (robotLocationTransform != null) {
289                         lastLocation =
    robotLocationTransform;
290                     }
291                     break;
292                 }
293             }
294 
295             // Provide feedback as to where the robot is
    located (if we know).
296             if (targetVisible) {
297                 // express position (translation) of
    robot in inches.
298                 VectorF translation = lastLocation.
    getTranslation();
299                 telemetry.addData("Pos (in)", "{X, Y, Z
    } = %.1f, %.1f, %.1f",
300                         translation.get(0) / mmPerInch,
    translation.get(1) / mmPerInch, translation.get(2) /
    mmPerInch);
301 
302                 // express the rotation of the robot in
    degrees.
303                 Orientation rotation = Orientation.
    getOrientation(lastLocation, EXTRINSIC, XYZ, DEGREES);
304                 telemetry.addData("Rot (deg)", "{Roll,
    Pitch, Heading} = %.0f, %.0f, %.0f", rotation.firstAngle,
     rotation.secondAngle, rotation.thirdAngle);
305             }
306             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
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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.
   toString());
70     if (gamepad1.a) {
71       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
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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
   #start()
57      */
58   @Override
59   public void init_loop() {
60   }
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#
   loop()
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
```

```java
package org.firstinspires.ftc.teamcode.RobotSystems;

import com.qualcomm.hardware.bosch.BNO055IMU;
import com.qualcomm.robotcore.eventloop.opmode.
LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.OpMode;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.Gamepad;
import com.qualcomm.robotcore.hardware.HardwareMap;
import com.qualcomm.robotcore.util.Range;

import org.firstinspires.ftc.teamcode.Pixy.PixyBlock;
import org.firstinspires.ftc.teamcode.Pixy.PixyCam;
import org.firstinspires.ftc.teamcode.Util.LogCreater;
import org.firstinspires.ftc.teamcode.Util.PIDController;
import org.firstinspires.ftc.teamcode.Util.GoldRecognation
;
import org.firstinspires.ftc.teamcode.Util.
TurnPIDController;

public class Drive {
    public enum Direction {
        FORWARD, BACKWARD;
    }
    public enum CurvedDirection {
        LEFT, RIGHT;
    }
    public enum Side {
        DEPOT,
        CREATER;
    }

    static final double COUNTS_PER_MOTOR_REV = 28;
    static final double DRIVE_GEAR_REDUCTION = 40;
    static final double WHEEL_DIAMETER_CM = 10.16;
    static final double COUNTS_PER_CM = (
COUNTS_PER_MOTOR_REV * DRIVE_GEAR_REDUCTION) /
            (WHEEL_DIAMETER_CM * 3.141592654);
    private final double KP = 0.05, KI = 0.03, KD = 0.03,
TOLERANCE = 1;
    private final double turnKP = 0.01, turnKI = 0.00009,
turnKD = 0.002, turnTOLERANCE = 7;

    private OpMode opMode;
    private LogCreater log;
```

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

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

```
139                     log.writeLog("leftDrive", leftDrive.
    getCurrentPosition(), "target: " + newLeftTarget);
140                     log.writeLog("rightDrive", rightDrive
    .getCurrentPosition(), "target: " + newRightTarget);
141                 }
142             }
143             leftDrive.setPower(0);
144             rightDrive.setPower(0);
145 
146             leftDrive.setMode(DcMotor.RunMode.
    RUN_USING_ENCODER);
147             rightDrive.setMode(DcMotor.RunMode.
    RUN_USING_ENCODER);
148         }
149     }
150 
151     public  void curvedDrive(double distanceCm, double
    angleFactor, double speed, Direction direction,
    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 *
    COUNTS_PER_CM);
164             newRightTarget = (int) (distanceCm *
    COUNTS_PER_CM / angleFactor);
165 
166             rightSpeed = speed / angleFactor;
167             leftSpeed = speed;
168         }
169         else {
170             newLeftTarget = (int) (distanceCm *
    COUNTS_PER_CM  / angleFactor);
171             newRightTarget = (int) (distanceCm *
    COUNTS_PER_CM);
172 
173             rightSpeed = speed;
```

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

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

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

```
273 BACKWARD,10);
274                     //turnByGyroAbsolut(30,10);
275                     //driveByEncoder(45,0.5, Direction.
    BACKWARD,10);
276                     break;
277             }
278
279         } else {
280             switch (pos) {
281                 case LEFT:
282                     turnByGyroAbsolut(30,10);
283                     driveByEncoder(45,0.5, Direction.
    BACKWARD,10);
284                     break;
285                 case CENTER:
286                 case UNKNOWN:
287                     turnByGyroAbsolut(5,10);
288                     driveByEncoder(40,0.5, Direction.
    BACKWARD,10);
289                     break;
290                 case RIGHT:
291                     turnByGyroAbsolut(-30,10);
292                     driveByEncoder(45,0.5, Direction.
    BACKWARD,10);
293                     break;
294             }
295         }
296         opMode.telemetry.addData("mineral", pos);
297         return pos;
298     }
299
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(LogCreater log) {
311         this.log = log;
312     }
313
```

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

```java
package org.firstinspires.ftc.teamcode.RobotSystems;

import android.graphics.Color;

import com.qualcomm.robotcore.eventloop.opmode.Disabled;
import com.qualcomm.robotcore.eventloop.opmode.
LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.OpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.Gamepad;
import com.qualcomm.robotcore.hardware.HardwareMap;

import org.firstinspires.ftc.teamcode.Util.LogCreater;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.PrintStream;
import java.util.Date;

public class Robot {

    public Drive drive = new Drive();
    public Intake intake = new Intake();
    public Climbing climbing = new Climbing();
    private LogCreater log;
    private OpMode opMode;

    public void init(HardwareMap hardwareMap, OpMode
opMode, LogCreater log) {
        this.opMode = opMode;
        this.log = log;
        drive.init(hardwareMap, opMode, log);
        intake.init(hardwareMap, opMode, log);
        climbing.init(hardwareMap, opMode, log);

        opMode.telemetry.addLine("robot initialized");
        opMode.telemetry.update();
    }

    public void init(HardwareMap hardwareMap, OpMode
opMode) {
        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(LogCreater log) {
65          this.log = log;
66          this.climbing.setLog(log);
67          this.drive.setLog(log);
68          this.intake.setLog(log);
69      }
70  }
71
```

```java
package org.firstinspires.ftc.teamcode.RobotSystems;

import com.qualcomm.robotcore.eventloop.opmode.OpMode;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.Gamepad;
import com.qualcomm.robotcore.hardware.HardwareMap;
import com.qualcomm.robotcore.hardware.Servo;

import org.firstinspires.ftc.teamcode.Util.LogCreater;

public class Intake {

    private final double COLLECTION_SPEED = 0.8;
    private final double INJECT_SPEED = 0.5;
    private final double RELEASE_SPEED = 0.5;
    private final double LEFT_SERVO_OPEN_POS = 0.8;
    private final double RIGHT_SERVO_OPEN_POS = 0;
    private final double LEFT_SERVO_CLOSE_POS = 0.20;
    private final double RIGHT_SERVO_CLOSE_POS = 0.7;

    private OpMode opMode;
    private LogCreater log;

    private DcMotor collectMotor;
    public Servo leftServo, rightServo;

    private boolean collectModeIsPrevActive;
    private boolean releaseModeIsPrevActive;
    private boolean releaseModeIsActive;
    private boolean collectModeIsActive;
    private boolean injecktIsActive;
    private boolean injecktIsPrevActive;

    private double timeToOpenRight = -1;

    public void init(HardwareMap hardwareMap, OpMode opMode, LogCreater log){
        this.log = log;
        init(hardwareMap, opMode);
    }
    public void init(HardwareMap hardwareMap, OpMode opMode) {
        this.opMode = opMode;
        leftServo = hardwareMap.get(Servo.class, "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              closeRightGate();
67              closeLeftGate();
68              this.stop();
69          }
70
71          if(releaseModeIsPrevActive && !releaseModeIsActive
    ) {
72              this.stop();
73              closeRightGate();
74              closeLeftGate();
75              timeToOpenRight = -1;
76          }
77
78          if (injecktIsActive) {
```

```
 79              this.injackt();
 80          }
 81
 82          if(injecktIsPrevActive && !injecktIsActive) {
 83              closeRightGate();
 84              closeLeftGate();
 85              this.stop();
 86          }
 87
 88          opMode.telemetry.addLine("intake: \n" )
 89                  .addData("collectMotorPower: ",
    collectMotor.getPower()+"\n")
 90                  .addData("rightServoPos: ", rightServo.
    getPosition())
 91                  .addData("leftServoPos: ", leftServo.
    getPosition()+"\n")
 92                  .addData("releaseModeIsPrevActive: ",
    releaseModeIsPrevActive)
 93                  .addData("collectModeIsPrevActive: ",
    collectModeIsPrevActive +"\n");
 94
 95          releaseModeIsPrevActive = releaseModeIsActive;
 96          collectModeIsPrevActive = collectModeIsActive;
 97          injecktIsPrevActive = injecktIsActive;
 98      }
 99
100      private void openLeftGate() {
101          leftServo.setPosition(LEFT_SERVO_OPEN_POS);
102      }
103
104      private void openRightGate() {
105          rightServo.setPosition(RIGHT_SERVO_OPEN_POS);
106      }
107
108      private void closeLeftGate() {
109          leftServo.setPosition(LEFT_SERVO_CLOSE_POS);
110      }
111
112      private void closeRightGate() {
113          rightServo.setPosition(RIGHT_SERVO_CLOSE_POS);
114      }
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(LogCreater log) {
148         this.log = log;
149     }
150 }
151
152
153
154
```

```
 1 /* Copyright (c) 2017 FIRST. All rights reserved.
 2  *
 3  * Redistribution and use in source and binary forms, with
    or without modification,
 4  * are permitted (subject to the limitations in the
   disclaimer below) provided that
 5  * the following conditions are met:
 6  *
 7  * Redistributions of source code must retain the above
   copyright notice, this list
 8  * of conditions and the following disclaimer.
 9  *
10  * Redistributions in binary form must reproduce the above
    copyright notice, this
11  * list of conditions and the following disclaimer in the
   documentation and/or
12  * other materials provided with the distribution.
13  *
14  * Neither the name of FIRST nor the names of its
   contributors may be used to endorse or
15  * promote products derived from this software without
   specific prior written permission.
16  *
17  * NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT
   RIGHTS ARE GRANTED BY THIS
18  * LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
   HOLDERS AND CONTRIBUTORS
19  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
   INCLUDING, BUT NOT LIMITED TO,
20  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   FOR A PARTICULAR PURPOSE
21  * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
   OR CONTRIBUTORS BE LIABLE
22  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   EXEMPLARY, OR CONSEQUENTIAL
23  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR
24  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER
25  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
   CONTRACT, STRICT LIABILITY,
26  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
   ANY WAY OUT OF THE USE
27  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    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.
   JustLoggingAccelerationIntegrator;
34 import com.qualcomm.robotcore.hardware.HardwareMap;
35
36 import org.firstinspires.ftc.robotcore.external.navigation
   .AngleUnit;
37 import org.firstinspires.ftc.robotcore.external.navigation
   .AxesOrder;
38 import org.firstinspires.ftc.robotcore.external.navigation
   .AxesReference;
39 import org.firstinspires.ftc.robotcore.external.navigation
   .Orientation;
40 import org.firstinspires.ftc.robotcore.external.navigation
   .Position;
41 import org.firstinspires.ftc.robotcore.external.navigation
   .Velocity;
42
43 /**
44  * {@link BT_Gyro} gives a short demo on how to use the
   BNO055 Inertial Motion Unit (IMU) from AdaFruit.
45  *
46  * Use Android Studio to Copy this Class, and Paste it
   into your team's code folder with a new name.
47  * Remove or comment out the @Disabled line to add this
   opmode to the Driver Station OpMode list
48  *
49  * @see <a href="http://www.adafruit.com/products/2472">
   Adafruit IMU</a>
50  */
51 public class BT_Gyro {
52     // The IMU sensor object
53     BNO055IMU imu;
54
55     public void init(HardwareMap hardwareMap) {
56         // Set up the parameters with which we will use
   our IMU. Note that integration
57         // algorithm here just reports accelerations to
   the logcat log; it doesn't actually
58         // provide positional information.
59         BNO055IMU.Parameters parameters = new BNO055IMU.
```

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

```java
package org.firstinspires.ftc.teamcode.RobotSystems;

import com.qualcomm.robotcore.eventloop.opmode.
LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.OpMode;
import com.qualcomm.robotcore.hardware.AnalogInput;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.DigitalChannel;
import com.qualcomm.robotcore.hardware.Gamepad;
import com.qualcomm.robotcore.hardware.HardwareMap;
import com.qualcomm.robotcore.hardware.Servo;

import org.firstinspires.ftc.teamcode.Util.GlobalVariebels
;
import org.firstinspires.ftc.teamcode.Util.LogCreater;
import org.firstinspires.ftc.teamcode.Util.PIDController;

import static java.lang.Thread.sleep;

public class Climbing {
    public enum Angle {
        DOWN(10),
        DRIVE_POS(30),
        COLLECT(152),
        LAND(52),
        GO_TO_CLIMB(58),
        CLIMB(30),
        PUT(59);
        float pos;
        final double ticksPerDegree =93.588;

        public int getTicks() {
            return ((int) (ticksPerDegree * pos));
        }

        Angle(float ang) {
            this.pos = ang;
        }
    }

    public enum Height {
        DRIVE_POS(0),
        COLLECT(15),
        LAND(31),
        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 LogCreater 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
    opMode, LogCreater log) {
 92          this.log = log;
 93          init(hardwareMap, opMode);
 94      }
 95      public void init(HardwareMap hardwareMap, OpMode
    opMode) {
 96          this.opMode = opMode;
 97          liftMotor = hardwareMap.get(DcMotor.class, "
    liftMotor");
 98          angleMotorLeft = hardwareMap.get(DcMotor.class, "
    angleMotorLeft");
 99          angleMotorRight = hardwareMap.get(DcMotor.class,
    "angleMotorRight");
100          hangServo = hardwareMap.get(Servo.class, "
    hangServo");
101          angleTouch = hardwareMap.get(DigitalChannel.class
    , "angle_touch");
102          liftTouch = hardwareMap.get(DigitalChannel.class,
     "lift_touch");
103          potentiometer = hardwareMap.get(AnalogInput.class
    , "potentiometer");
104          anglePID = new PIDController(KP, KI, KD,
    TOLERANCE, opMode);
105
106          angleMotorLeft.setZeroPowerBehavior(DcMotor.
    ZeroPowerBehavior.BRAKE);
107          angleMotorRight.setZeroPowerBehavior(DcMotor.
    ZeroPowerBehavior.BRAKE);
108          liftMotor.setZeroPowerBehavior(DcMotor.
    ZeroPowerBehavior.BRAKE);
109
110          liftMotor.setPower(0);
111          angleMotorLeft.setPower(0);
112          angleMotorRight.setPower(0);
113
114          liftMotor.setDirection(DcMotor.Direction.REVERSE)
    ;
115          angleMotorLeft.setDirection(DcMotor.Direction.
    FORWARD);
116          angleMotorRight.setDirection(DcMotor.Direction.
    REVERSE);
```

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

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

```
187                     log.writeLog("angleMotor", getAngle()
    , "target: " + anglePID.getSetpoint());
188                 }
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.
    getCurrentPosition(), "command to move to drive pos " +
    Height.DRIVE_POS.getTicks());
197                 log.writeLog("angleMotor", getAngle(), "
    command to move to drive pos " + Angle.DRIVE_POS.pos);
198             }
199         }
200
201         if (operator.dpad_left) {
202             moveLift(Height.CLIMB);
203             moveAngle(Angle.CLIMB);
204             if (log != null) {
205                 log.writeLog("liftMotor", liftMotor.
    getCurrentPosition(), "command to move to climb " +
    Height.CLIMB.getTicks());
206                 log.writeLog("angleMotor", getAngle(), "
    command to move to climb " + Angle.CLIMB.pos);
207             }
208         }
209
210         if (operator.dpad_up) {
211             moveLift(Height.GO_TO_CLIMB);
212             moveAngle(Angle.GO_TO_CLIMB);
213             openServo();
214             if (log != null) {
215                 log.writeLog("liftMotor", liftMotor.
    getCurrentPosition(), "command to move to go to climb " +
     Height.GO_TO_CLIMB.getTicks());
216                 log.writeLog("angleMotor", getAngle(), "
    command to move to go to climb " + Angle.GO_TO_CLIMB.pos)
    ;
217             }
218         }
219
220         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.
    getCurrentPosition() + "\n")
261                 .addData(" Angle Right Position: ",
    angleMotorRight.getCurrentPosition() + "\n")
262                 .addData(" Angle Left Position: ",
    angleMotorLeft.getCurrentPosition() + "\n")
263                 .addData("Servo position: ", hangServo.
    getPosition() + "\n")
264                 .addData("liftTouch: " ,
    liftTouchIsActive)
265                 .addData(" angleTouch: ",
    angleTouchIsActive +"\n")
266                 .addData(" potentiometer: ",
    potentiometer.getVoltage())
267                 .addData("angle: ", getAngle())
268                 .addData("liftPosEndAuto",
    GlobalVariebels.liftPosEndAuto);
269
270
271         angleTouchIsPrevActive = angleTouchIsActive;
272         liftTouchIsPrevActive = liftTouchIsActive;
273
274         angleJoystickPrevValue = angleJoystickValue;
275         liftJoystickPrevValue = liftJoystickValue;
276     }
277
278     public void moveLift(Height height) {
279         liftMotor.setTargetPosition(height.getTicks() -
    GlobalVariebels.liftPosEndAuto);
280         liftMotor.setMode(DcMotor.RunMode.RUN_TO_POSITION
    );
281         liftMotor.setPower(Math.abs(LIFT_SPEED));
282     }
283
284     public void moveAngle(Angle angle) {
285         anglePID.reset(angle.pos, getAngle());
286         double output = anglePID.getOutput(getAngle());
287         if(IsLiftInDeadZone()){
288             output *= 0.1;
289             opMode.telemetry.addData("output: " , output)
    ;
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         liftMotor.setPower(0);
333     }
334
335     public void moveAngleAuto(Angle angle) {
336         anglePID.reset(angle.pos, getAngle());
337         while (!anglePID.onTarget() && ((LinearOpMode)
    opMode).opModeIsActive()) {
338             double output = anglePID.getOutput(getAngle()
    );
339             angleMotorRight.setPower(output);
340             angleMotorLeft.setPower(output);
341             opMode.telemetry.addData("error: ", anglePID.
    getCurrentError())
342                     .addData("output: ", output)
343                     .addData("angle", getAngle());
344             opMode.telemetry.update();
345             if (log != null) {
346                 log.writeLog("angle", getAngle(), "target
    : " + angle.pos);
347             }
348         }
349         angleMotorRight.setPower(0);
350         angleMotorLeft.setPower(0);
351     }
352
353     public  void moveAngleAndHeight(Angle angle, Height
    height){
354         anglePID.reset(angle.pos, getAngle());
355
356         liftMotor.setTargetPosition(height.getTicks());
357         liftMotor.setMode(DcMotor.RunMode.RUN_TO_POSITION
    );
358         liftMotor.setPower(Math.abs(LIFT_SPEED));
359
360         while (!anglePID.onTarget()  && ((LinearOpMode)
    opMode).opModeIsActive()){
361             double output = anglePID.getOutput(getAngle()
    );
362             angleMotorRight.setPower(output);
363             angleMotorLeft.setPower(output);
364             opMode.telemetry.addLine("Angle \n")
365                     .addData("error: ", anglePID.
    getCurrentError())
366                     .addData("output: ", output)
```

```
367                     .addData("angle", getAngle() + "\n");
368
369             opMode.telemetry.addLine("Lift \n")
370                     .addData("position: ", liftMotor.
    getCurrentPosition());
371             opMode.telemetry.update();
372
373             if(!liftMotor.isBusy()){
374                 liftMotor.setPower(0);
375                 liftMotor.setMode(DcMotor.RunMode.
    RUN_USING_ENCODER);
376             }
377             if (log != null) {
378                 log.writeLog("angle", getAngle(), "target
    : " + angle.pos);
379                 log.writeLog("lift", liftMotor.
    getCurrentPosition(), "target: " + height.getTicks());
380             }
381         }
382         angleMotorLeft.setPower(0);
383         angleMotorRight.setPower(0);
384
385         while(liftMotor.isBusy()  && ((LinearOpMode)
    opMode).opModeIsActive()){
386             opMode.telemetry.addLine("Lift \n")
387                     .addData("position: ", liftMotor.
    getCurrentPosition());
388             opMode.telemetry.update();
389             if (log != null) {
390                 log.writeLog("lift", liftMotor.
    getCurrentPosition(), "target: " + height.getTicks());
391             }
392         }
393
394         liftMotor.setPower(0);
395         liftMotor.setMode(DcMotor.RunMode.
    RUN_USING_ENCODER);
396     }
397
398     public void land() {
399         //moveAngleAuto(Climbing.Angle.LAND);
400         //moveLiftAuto(Climbing.Height.LAND);
401         //moveAngleAuto(Climbing.Angle.LAND);
402         moveAngleAndHeight(Climbing.Angle.LAND, Climbing.
    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(LogCreater 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             opMode.telemetry.addData("error: " ,anglePID.
    getCurrentError())
445                     .addData("angle", getAngle());
446             opMode.telemetry.update();
447         }
448     }
449     */
450 }
451
```