Utility Samples Documentation

`UtilityCameraFrameCapture.java`

Description:

This utility OpMode allows you to capture frames from a Webcam or the internal Control Hub camera and save them to the robot controller's storage. This is extremely useful for:

- Calibrating cameras for AprilTag detection.
- Collecting training data for Machine Learning models (e.g., TensorFlow or custom models).
- Debugging vision pipelines by analyzing real-world images seen by the robot.
- **Hardware Map Name:** `"Webcam 1"` (if using a webcam)
- **Output Location:** Saved to the Robot Controller's internal storage (typically `/sdcard/FIRST/data/`).

Code Breakdown:

1. Configuration Parameters:

At the top of the class, there are constants you should edit to match your setup.

```java
// Set to true if using an external webcam, false for internal phone/hub camera

final boolean USING_WEBCAM = false;



// If using internal camera, which one? (BACK or FRONT)

final BuiltinCameraDirection INTERNAL_CAM_DIR = BuiltinCameraDirection.BACK;



// Resolution to capture. Higher resolution = larger files.

final int RESOLUTION_WIDTH = 640;

final int RESOLUTION_HEIGHT = 480;
```

2. Initialization (VisionPortal):

The code uses the `VisionPortal` builder to set up the camera. It handles both Webcam and Internal Camera logic based on the `USING_WEBCAM` flag.

```
VisionPortal portal;



if (USING_WEBCAM) {

    // Initialize for Webcam

    portal = new VisionPortal.Builder()

            .setCamera(hardwareMap.get(WebcamName.class, "Webcam 1"))

            .setCameraResolution(new Size(RESOLUTION_WIDTH,
RESOLUTION_HEIGHT))

            .build();

} else {

    // Initialize for Internal Camera

    portal = new VisionPortal.Builder()

            .setCamera(INTERNAL_CAM_DIR)

            .setCameraResolution(new Size(RESOLUTION_WIDTH,
RESOLUTION_HEIGHT))

            .build();

}
```

3. Main Loop & Capture Logic:

The loop checks for the **X** button press. It uses a "rising edge detector" (`x && !lastX`) to ensure one press equals one photo, preventing a burst of hundreds of images if you hold the button.

```
boolean x = gamepad1.x;
```

```
// Check if X is pressed AND was not pressed last loop (Rising Edge)

if (x && !lastX) {

    // Save the frame with a unique name based on a counter

    // Files are saved as "CameraFrameCapture-000001.jpg", etc.

    portal.saveNextFrameRaw(String.format(Locale.US, "CameraFrameCapture-
%06d", frameCount++));

    // Record time for UI feedback

    capReqTime = System.currentTimeMillis();

}



// Update lastX for the next loop

lastX = x;
```

4. Telemetry Feedback:

Provides visual confirmation when a frame is captured.

```
// Show "Captured Frame!" for 1 second after capture

if (capReqTime != 0 && System.currentTimeMillis() - capReqTime > 1000) {

    capReqTime = 0; // Reset timer

}



if (capReqTime != 0) {

    telemetry.addLine("\nCaptured Frame!");

}
```

`UtilityOctoQuadConfigMenu.java`

Description:

This is a sophisticated **Configuration Tool** for the DigitalChickenLabs OctoQuad. Unlike a normal OpMode, this is an interactive menu system that runs on the Driver Station. It allows you to configure the internal settings of the OctoQuad module without writing code for every parameter.

Capabilities:

- **I2C Recovery Mode:** Configure how the OctoQuad handles I2C bus lockups.
- **Channel Bank Config:** Switch banks between Quadrature and Pulse-Width modes.
- **Encoder Directions:** Reverse individual encoders.
- **Velocity Intervals:** Tune the time window for velocity calculation.
- **Pulse Width Parameters:** Set min/max pulse widths for absolute encoders (e.g., REV Through Bore).
- **Flash Storage:** Save settings permanently to the OctoQuad's EEPROM.
- **Hardware Map Name:** The code searches for **all** configured OctoQuads, but primarily interacts with the first one found.

Code Breakdown:

1. Initialization & Sanity Checks:

The code first attempts to talk to the OctoQuad to verify it's connected and running compatible firmware.

```java
// Get the first OctoQuad from the HardwareMap

octoquad = hardwareMap.getAll(OctoQuad.class).get(0);



// Check Chip ID to verify connection

if(octoquad.getChipId() != OctoQuad.OCTOQUAD_CHIP_ID) {

    telemetry.addLine("Error: cannot communicate with OctoQuad...");

    error = true;

}
```

```
// Check Firmware Version

if(octoquad.getFirmwareVersion().maj != OctoQuad.SUPPORTED_FW_VERSION_MAJ) {

    telemetry.addLine("Error: Firmware version mismatch...");

    error = true;

}
```

2. Menu System Structure:

The OpMode uses a custom `TelemetryMenu` class (nested at the bottom of the file) to create a hierarchical menu.

```
// Root Menu

TelemetryMenu.MenuElement rootMenu = new TelemetryMenu.MenuElement("OctoQuad Config Menu", true);



// Sub-menus

TelemetryMenu.MenuElement menuHwInfo = new TelemetryMenu.MenuElement("Hardware Information", false);

TelemetryMenu.MenuElement menuEncoderDirections = new TelemetryMenu.MenuElement("Set Encoder Directions", false);

// ... other sub-menus ...



// Add sub-menus to root

rootMenu.addChild(menuHwInfo);

rootMenu.addChild(menuEncoderDirections);

// ...
```

3. Populating Menu Options:

It reads the *current* settings from the OctoQuad to populate the menu defaults.

```java
// Example: Populating Encoder Directions

for(int i = 0; i < OctoQuad.NUM_ENCODERS; i++) {

    optionsEncoderDirections[i] = new TelemetryMenu.BooleanOption(

            String.format("Encoder %d direction", i),

            // Read current setting: Is it REVERSE?

            octoquad.getSingleEncoderDirection(i) ==
OctoQuad.EncoderDirection.REVERSE,

            "-", // Display for False (Forward)

            "+"  // Display for True (Reverse)

    );

}
```

4. Saving Settings (RAM vs. Flash):

- **Send to RAM:** Applies settings immediately but they are lost on power cycle. Useful for testing.
- **Program to FLASH:** Saves settings permanently.

```java
// The 'onClick' handler for the "Program to FLASH" button

optionProgramToFlash = new TelemetryMenu.OptionElement() {

    @Override

    void onClick() {

        sendSettingsToRam(); // First apply to RAM

        octoquad.saveParametersToFlash(); // Then commit to Flash

    }

};
```

5. `sendSettingsToRam()` Method:

This helper function iterates through all the menu options and sends the values to the OctoQuad.

```
void sendSettingsToRam() {

    for(int i = 0; i < OctoQuad.NUM_ENCODERS; i++) {

        // Update Direction

        octoquad.setSingleEncoderDirection(i,
optionsEncoderDirections[i].getValue() ? OctoQuad.EncoderDirection.REVERSE :
OctoQuad.EncoderDirection.FORWARD);

        // Update Velocity Interval

        octoquad.setSingleVelocitySampleInterval(i,
optionsVelocityIntervals[i].getValue());

        // Update Pulse Width Params

        OctoQuad.ChannelPulseWidthParams params = new
OctoQuad.ChannelPulseWidthParams();

        params.max_length_us = optionsAbsParamsMax[i].getValue();

        params.min_length_us = optionsAbsParamsMin[i].getValue();

        octoquad.setSingleChannelPulseWidthParams(i, params);

    }

    // ... update other global settings ...

}
```

6. Menu Navigation Controls:

The sample includes a custom UI. The controls are:

- **D-Pad Up/Down:** Navigate items.
- **X (or Square):** Select an item or enter a sub-menu.
- **D-Pad Left/Right:** Edit the selected option value.
- **Left Bumper:** Go up one level (Back).

7. The `TelemetryMenu` Helper Class:

The file includes a reusable `TelemetryMenu` class (lines 305+) that handles the UI logic.

- **Structure:** It uses a Tree structure where `MenuElement`s can have children.
- **Stack:** It uses a `Stack<Integer>` to remember your cursor position when you enter/exit sub-menus.
- **Rendering:** It builds an HTML string using `StringBuilder` and sends it to `telemetry`.

```java
// Example of the rendering loop

for (int i = 0; i < children.size(); i++) {

    if (selectedIdx == i) {

        builder.append("[<font color=green face=monospace>•</font>] "); // Cursor

    } else {

        builder.append("[ ] ");

    }

    // ... append name ...

}

telemetry.addLine(builder.toString());
```