Программирование Teleop

Translated by PID 24670

Конфигурация

Прежде чем начать программировать, Вам нужно создать файл конфигурации. Ниже предоставлен обзор того, как должен быть настроен робот, чтобы код Teleop функционировал в должном образе:

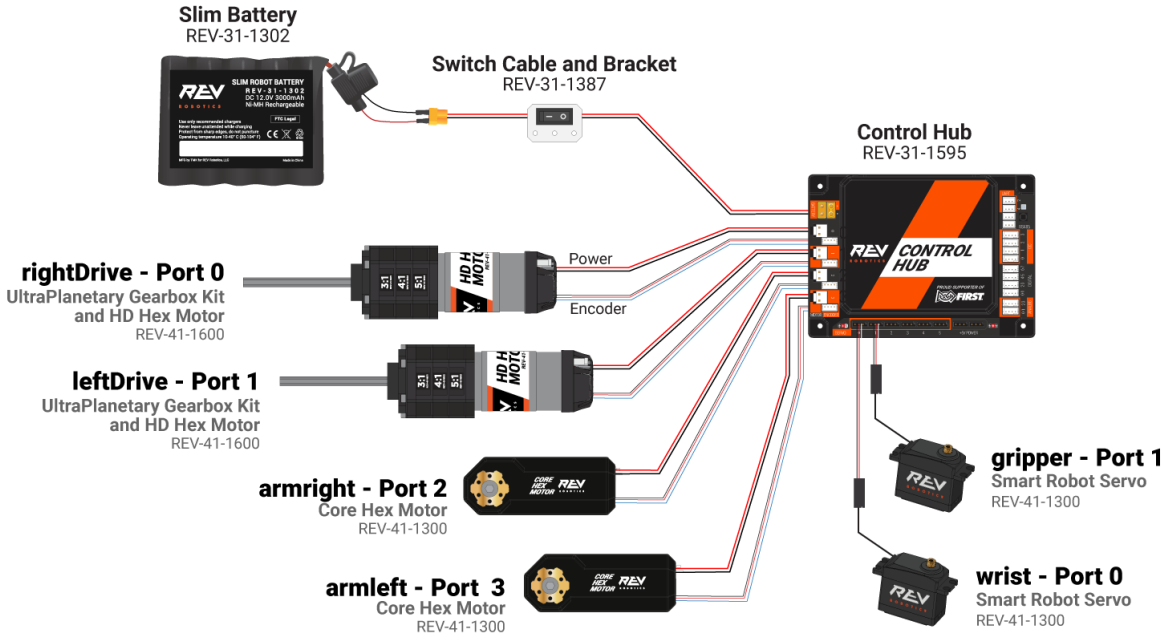
Тип порта	Номер порта	Тип устройства	Название
Двигатель	0	Шестигранный двигатель REV Robotics Ultraplanetary HD	rightDrive
Двигатель	1	Шестигранный двигатель REV Robotics Ultraplanetary HD	leftDrive
Двигатель	2	Шестигранный двигатель REV Robotics Core	rightarm
Двигатель	3	Шестигранный двигатель REV Robotics Core	leftarm
Сервопривод	0	Сервопривод	wrist

Тип порта	Номер порта	Тип устройства	Название
Сервопривод	1	Сервопривод	gripper

Загрузка файла конфигурации

Чтобы установить этот файл конфигурации на вашего робота, перетащите нижепредставленный файл в папку “FIRST” файловой системы вашего центра управления.
StarterBot2024
Для получения более подробной информации о процессе настройки ознакомьтесь с Hello Robot - Configuration!

Схема подключения



Название устройства	Вид устройства	Порт
rightDrive	UltraPlanetary Gearbox Kit and HD Hex Motor	Motor/Encoder Port 0
leftDrive	UltraPlanetary Gearbox Kit and HD Hex Motor	Motor/Encoder Port 1

Название устройства	Вид устройства	Порт
armright	Core Hex Motor	Motor/Encoder Port 2
armleft	Core Hex Motor	Motor/Encoder Port 3
wrist	Smart Robot Servo	Servo Port 0
gripper	Smart Robot Servo	Servo Port 1

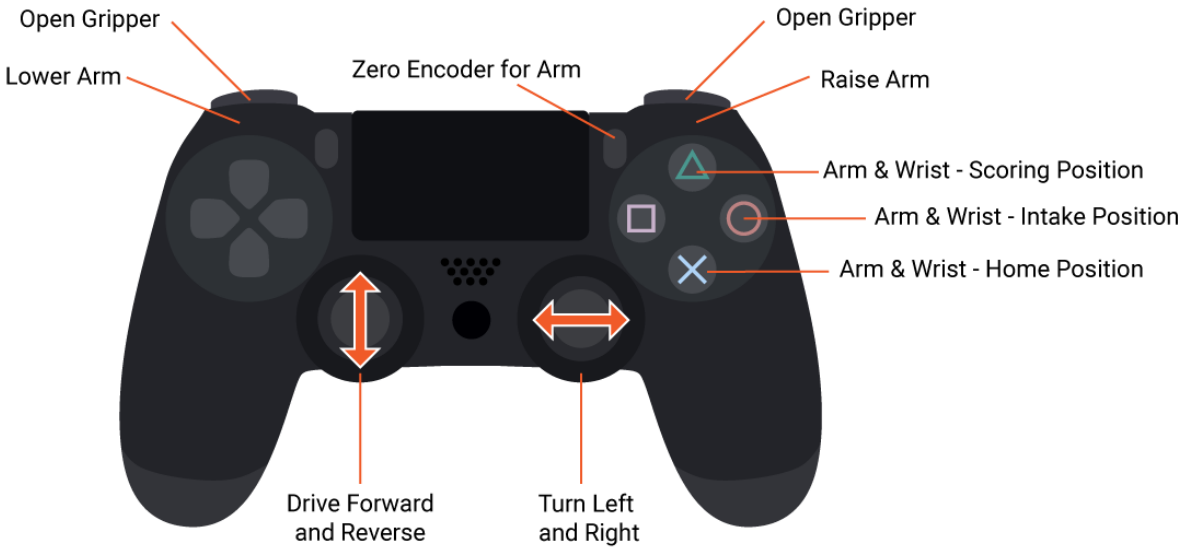
Настройки геймпада

Пункты, которые надо учитывать при настройке геймпада:

- Какие входные данные необходимы механизму?
  - **Джойстики и триггеры** вводят floating point data в свой код, позволяющий регулировать скорость двигателя в зависимости от давления, приложенного к спусковому крючку, или положения джойстика.
  - **Кнопки, бамперы и D-Pad** предоставляют логические данные для вашего кода и идеально подходят для запуска таких действий, как поворот двигателя в заданное положение.
- Какую трансмиссию вы используете и какой стиль вождения хотите использовать?
  - Мы решили, что наш робот для INTO THE DEEP будет управляться с разделенным аркадным приводом для повышения точности вождения.
- Какой ввод имеет наибольший смысл?
  - Для стартового бота этого года доступно несколько кнопок. Оба бампера контролируют способность клешни захватывать пиксели. Наручный сервопривод привязан к кнопкам "A/Крестик" и "B/Круг" для переключения из положения "Прием" в исходное положение.

Не на всех геймпадах кнопки обозначены одинаково. Точное отображение кнопок смотрите в документации производителя.

Макет геймпада для стартового бота REV DUO FTC 2024-25:



Кнопки геймпада	Функция
Правый джойстик	Поворот на лево и на право
Левый джойстик	Движение вперед и назад
Правый и/или левый бампер	Закрытие клешни
A/Крестик	Arm & Wrist перемещаются в исходное положение • Рука опущена, запястье поднято
B/Круг	Arm & Wrist меняются в позицию для захвата • Рука опущена, запястье опущено
Y/Треугольник	Arm & Wrist меняются в результативную позицию • Рука назад, запястье вверх
Опция	Нулевой кодер для Arm
Левый триггер	Спустить Arm

Кнопки геймпада	Функция
Правый триггер	Поднять Arm

Дополнительную информацию о программировании геймпадов для использования с вашим роботом можно найти на сайте Hello Robot - Using Gamepads.

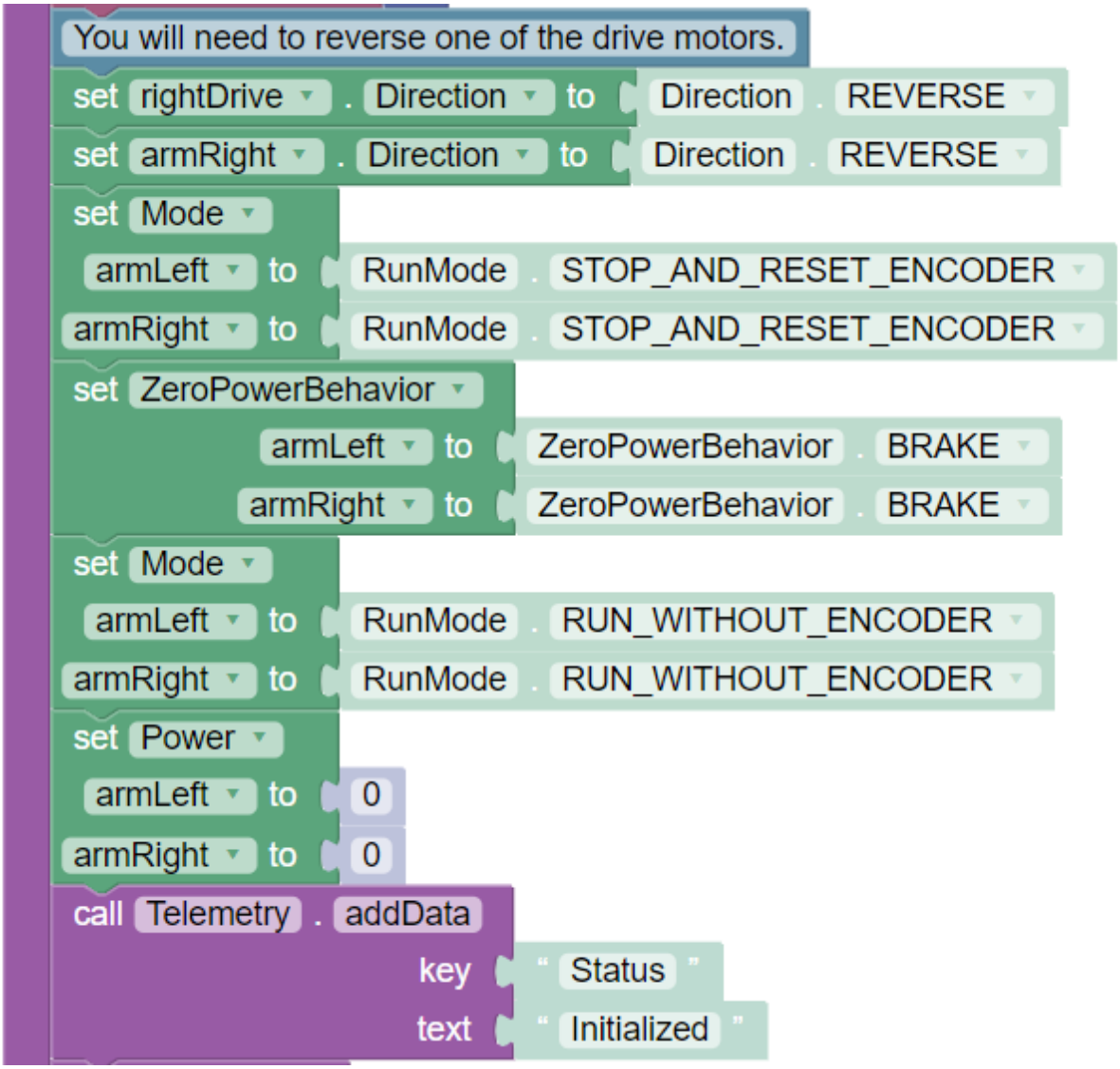
Программирование Teleop - Блоки

В этом разделе предполагается, что вы изучили некоторые основы программирования FTC, ознакомившись с руководством по Hello Robot. Если вы еще не ознакомились с этим руководством, пожалуйста, ознакомьтесь с ним, прежде чем продолжить.
В Hello Robot- основы программирования трансмиссий мы рассмотрели, как программировать аркадный привод с помощью одного джойстика. В этом примере мы будем программировать аркадный привод с помощью двух джойстиков. Этот тип привода называется "split arcade". В split arcade drive левый джойстик будет управлять движением робота вперед и назад, а правый джойстик - поворотом. Это похоже на то, как управляются некоторые радиоуправляемые машинки и видеоигры.

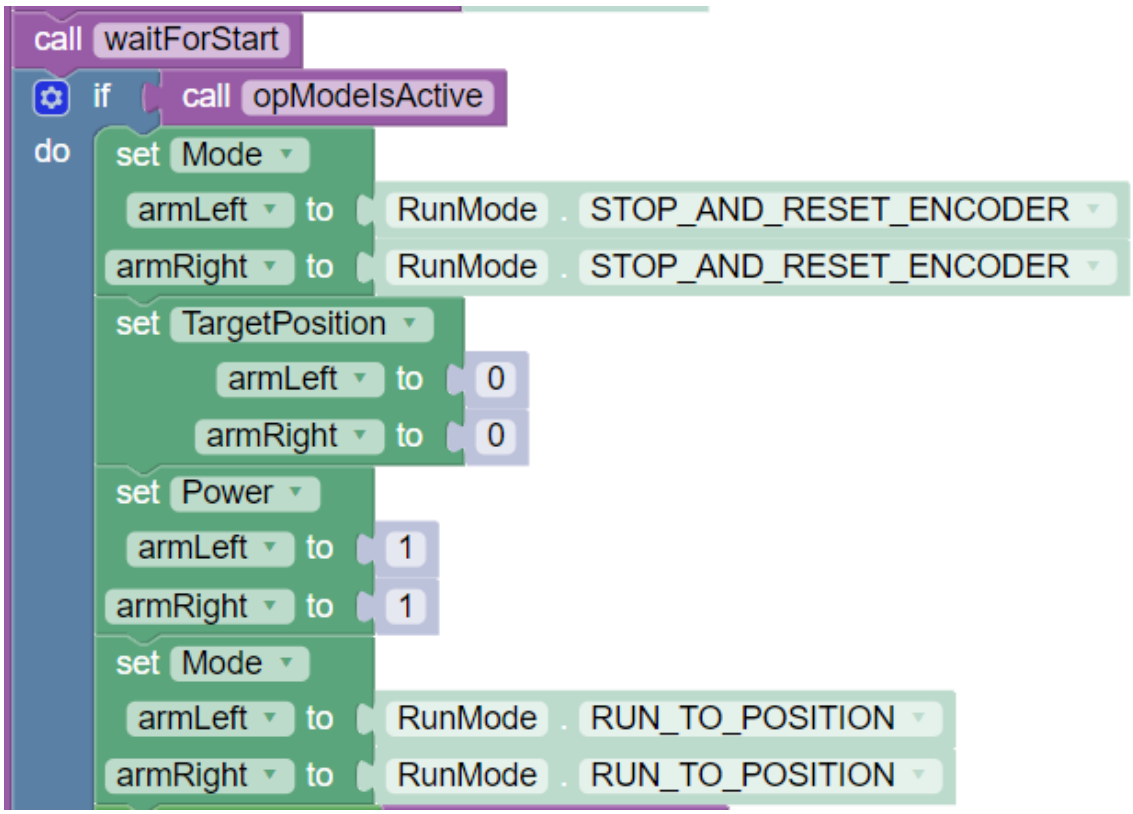
Переменные для стартового бота 2024-25
Для точного перемещения захвата и запястья вам нужно будет запрограммировать сервоприводы с помощью программатора SRS,входящего в комплект поставки!

Настройка кодеров:

* Этот код запускается ПОСЛЕ активации инициализации, но ДО нажатия кнопки воспроизведения.
Этот раздел позволяет обнулить кодировщик на шестигранных двигателях arm Core при запуске и устанавливает их поведение по умолчанию. Кроме того, это позволяет двум двигателям работать задним ходом, одному на трансмиссии и одному на рычаге управления.



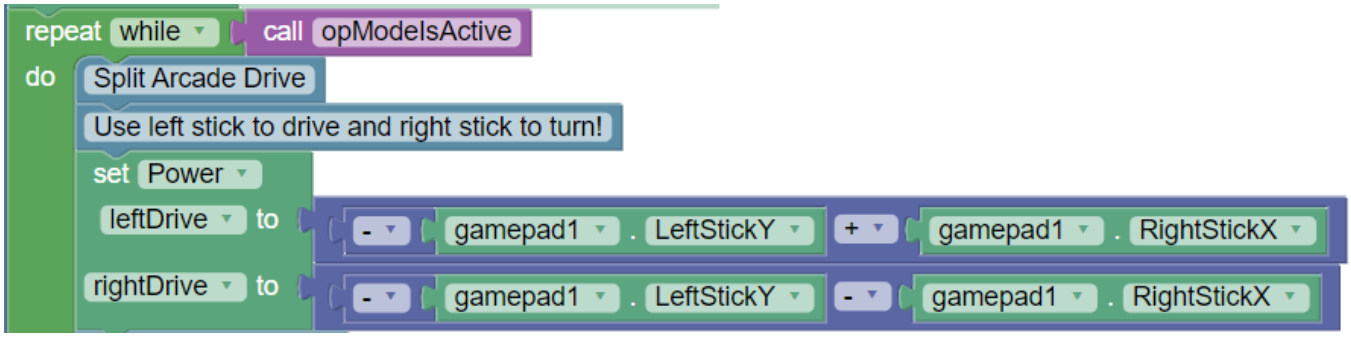
* Этот код будет запущен ОДИН раз после активации кнопки воспроизведения. Этот раздел служит дополнительной проверкой для сброса мощности и положения рычага, а также для установки двигателей Core Hex в режим "RUN_TO_POSITION" по умолчанию при нажатии кнопки воспроизведения.



Разделенный аркадный драйв:

* Все следующие разделы ПОВТОРНО запускаются в "цикле while" после активации кнопки воспроизведения.

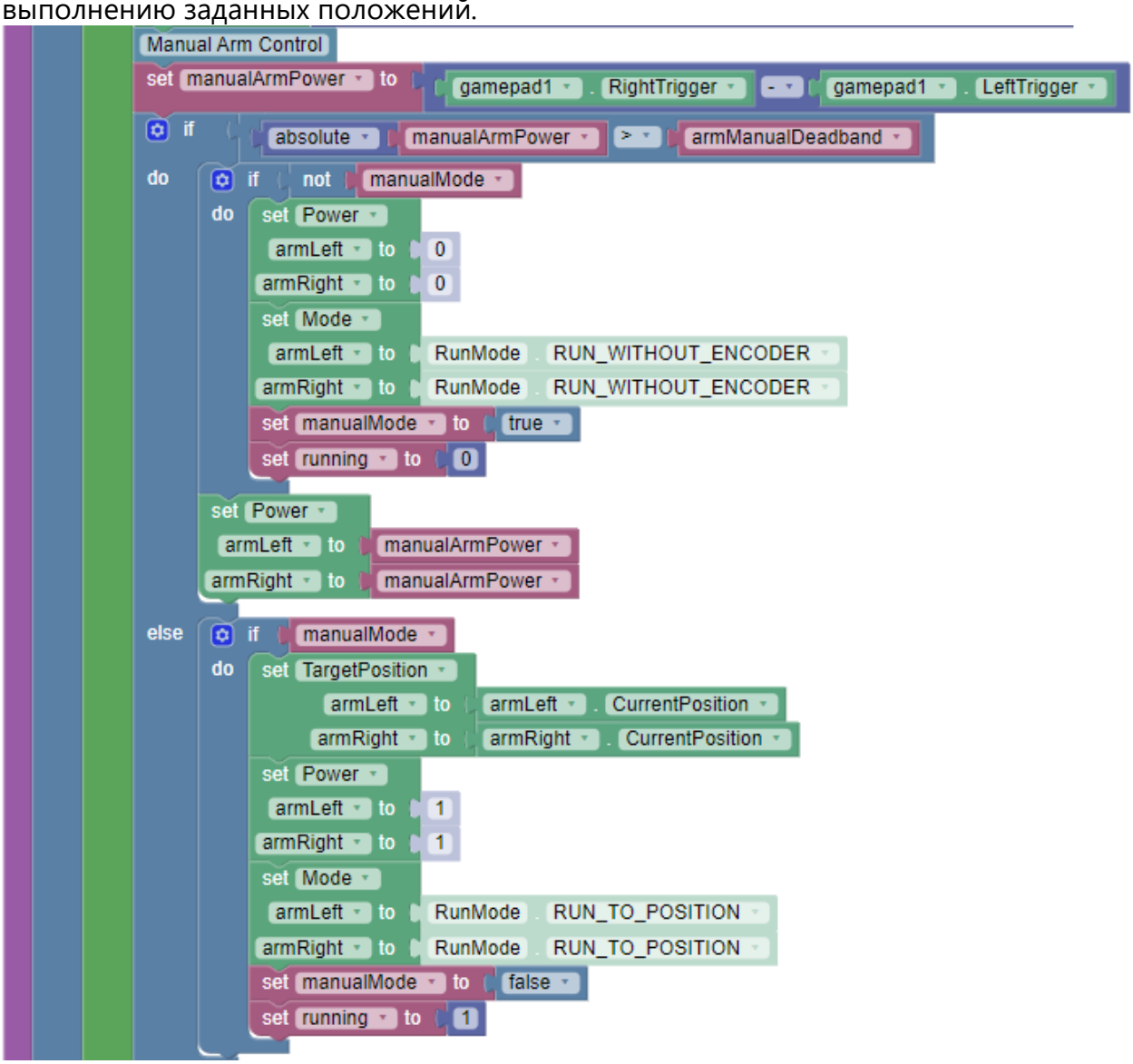
Давайте начнем с программирования нашего разделенного аркадного привода. Этот код будет принимать входные данные от правого и левого джойстиков для определения движения робота. Левый джойстик управляет движением вперед и назад, в то время как правый джойстик управляет левым и правым поворотами.



Код аркадного раздельного привода

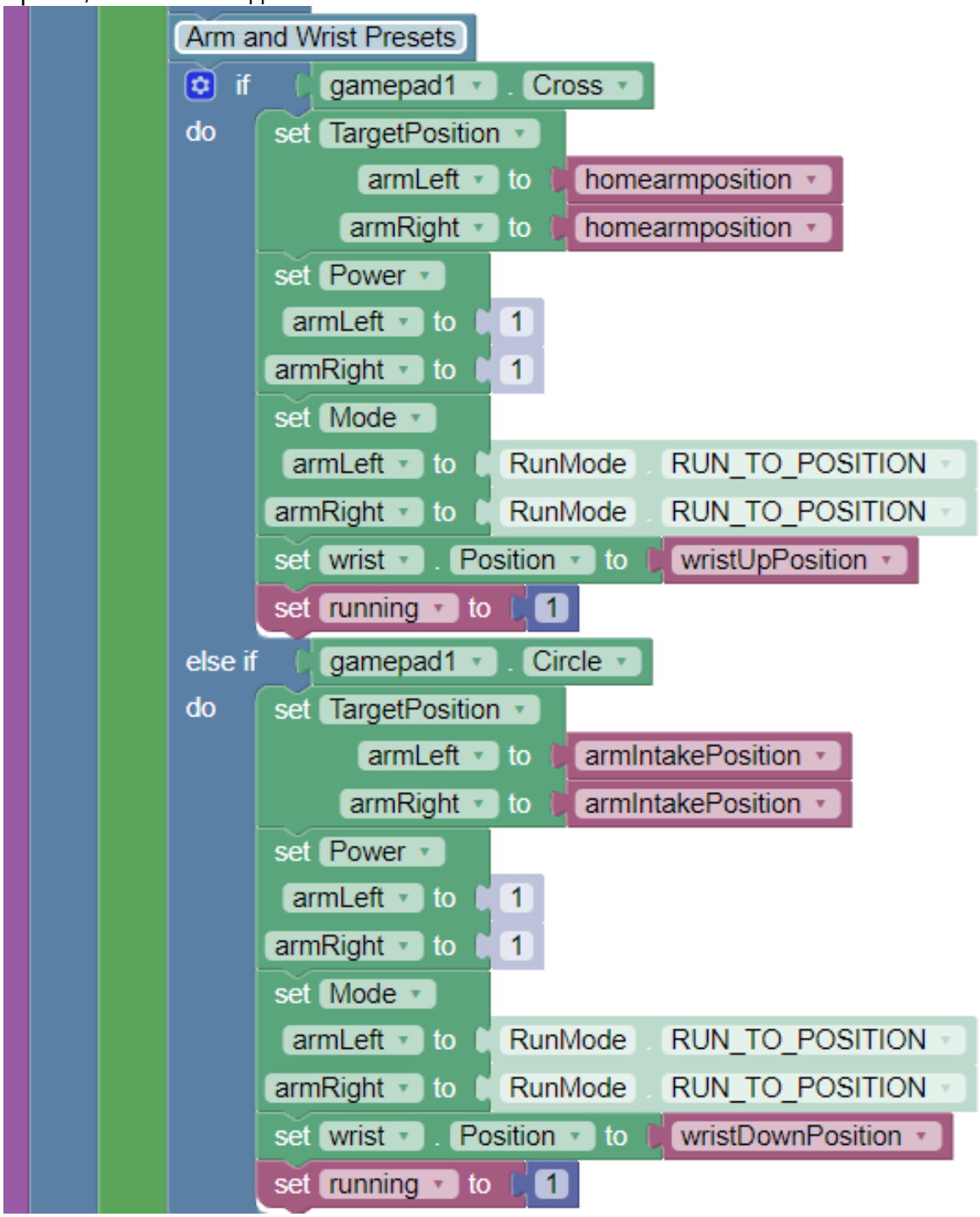
Ручное движение arm:

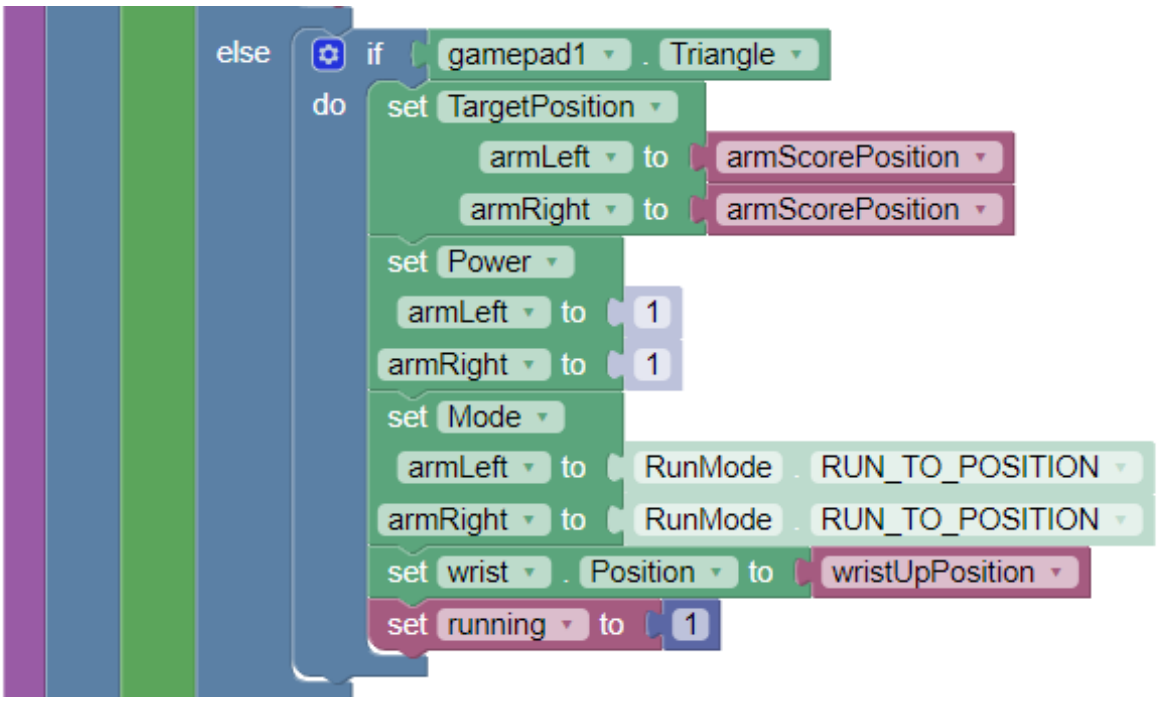
Триггеры на контроллере определяют направление вращения рычага. Мощность рычага определяется тем, насколько сильно нажаты оба триггера. Кроме того, оператор if / then переключает рычаг между ручным приводом и готовностью к выполнению заданных положений.



Предустановленные положения arm и wrist:

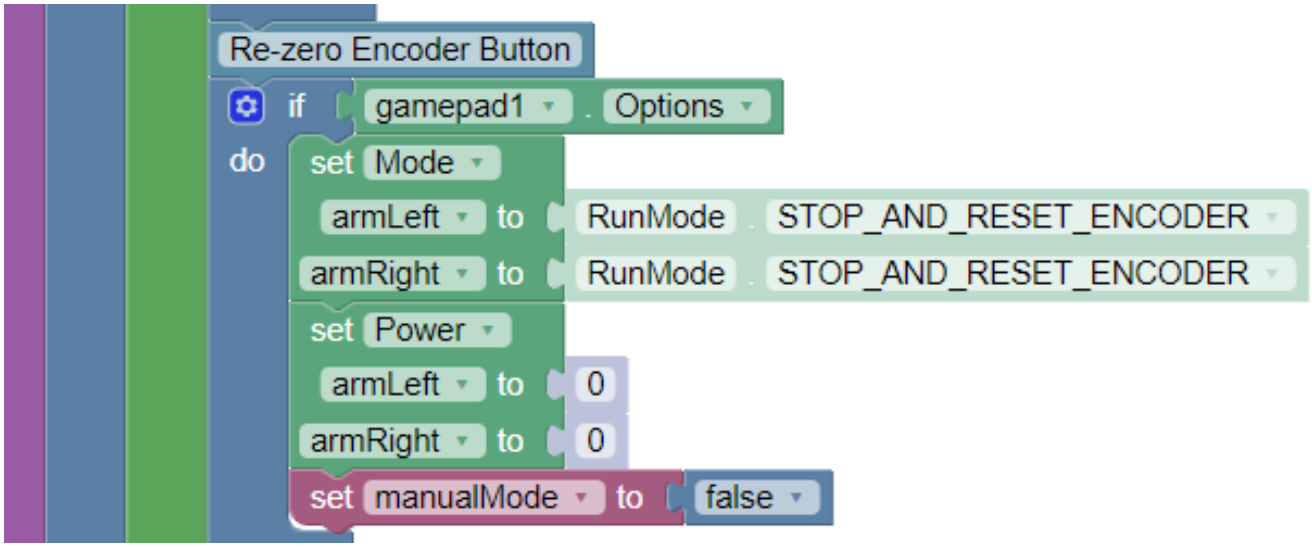
В стартовом роботе есть три предустановленных положения для руки и запястья - прием, начало и подсчет очков.





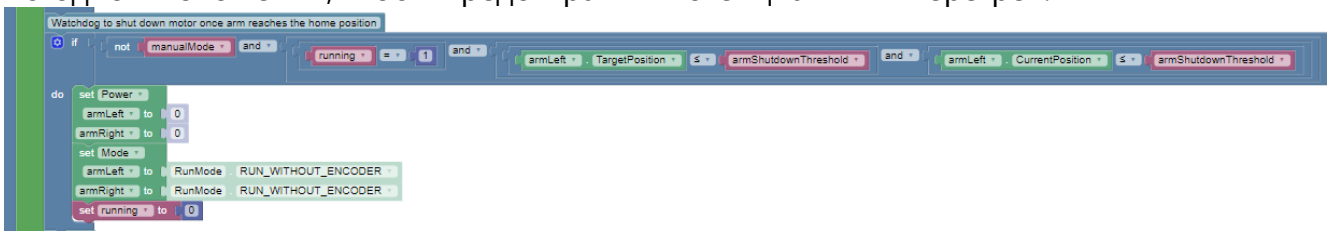
Кодировщик повторного обнуления:

При повторном обнулении датчика рука и запястье должны находиться в исходном положении.

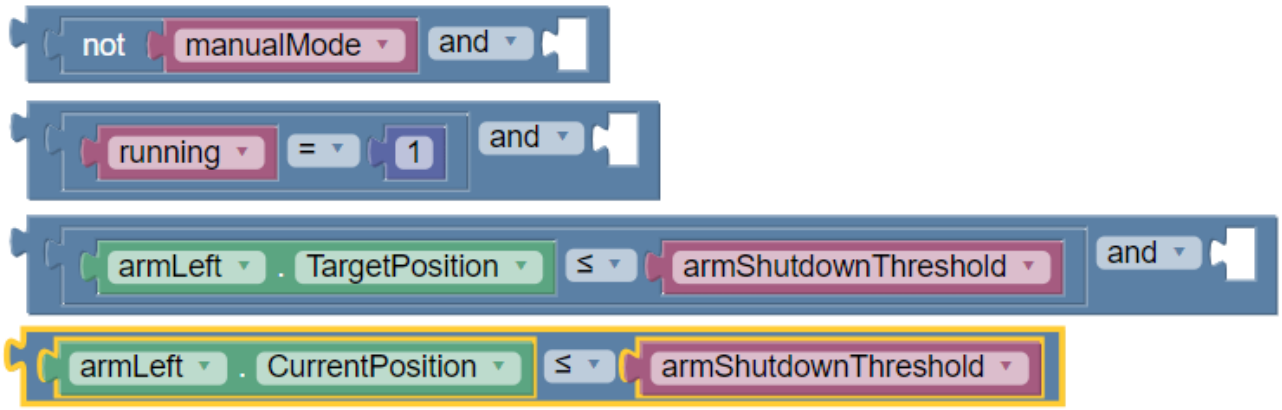


Контрольный таймер для отключения двигателя в исходном положении:

Эта проверка предотвращает продолжение работы двигателей Core Hex в исходном положении, чтобы предотвратить потенциальный перегрев.



Давайте подробнее рассмотрим все части этого оператора if / then:

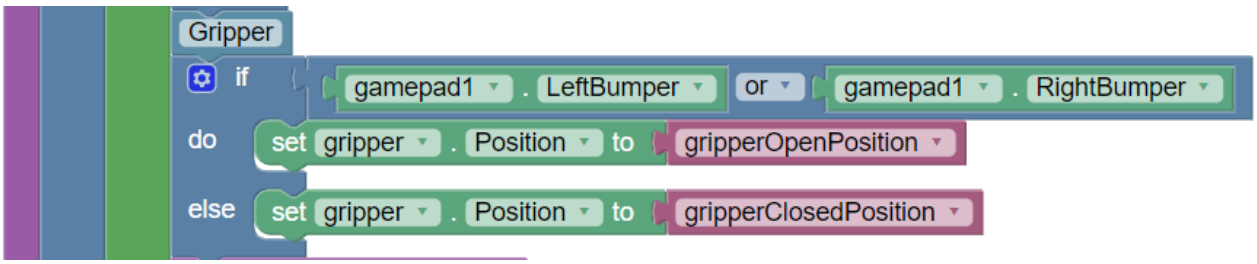


Этот оператор if / then будет выполняться только при соблюдении всех следующих условий:

- В настоящее время рука не движется в ручном режиме из-за нажатия триггеров
- Двигатели активно работают
- Текущее и целевое положение шестигранного двигателя с левым сердечником меньше или равно пороговому значению (в настоящее время установлено значение пять)
    - Значение начальной позиции установлено равным 0, в то время как Входное значение установлено равным 10

Элементы управления захватом:

Когда один или оба бампера на контроллере удерживаются нажатыми, захват полностью открывается. При отпускании он возвращается в закрытое положение, чтобы зафиксировать пальцы вокруг пикселя.



Программирование Teleop - OnBot Java

В этом разделе предполагается, что вы изучили некоторые основы программирования FTC, ознакомившись с руководством по Hello Robot. Если вы еще не ознакомились с этим руководством, пожалуйста, ознакомьтесь с ним, прежде чем продолжить.
В Hello Robot- основы программирования трансмиссий мы рассмотрели, как программировать аркадный привод с помощью одного джойстика. В этом примере мы будем программировать аркадный привод с помощью двух джойстиков. Этот тип привода называется "split arcade". В split arcade drive левый джойстик будет управлять движением робота вперед и назад, а правый джойстик - поворотом. Это похоже на то, как управляются некоторые радиоуправляемые машинки и видеоигры.

Полный код робота

Чтобы использовать этот код с вашим начинающим ботом, скопируйте и вставьте его в новый режим работы OnBot Java!

```
/* Copyright (c) 2017 FIRST. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted (subject to the limitations in the disclaimer below) provided that
 * the following conditions are met:
 *
 * Redistributions of source code must retain the above copyright notice, this list
 * of conditions and the following disclaimer.
 *
 * Redistributions in binary form must reproduce the above copyright notice, this
 * list of conditions and the following disclaimer in the documentation and/or
 * other materials provided with the distribution.
 *
 * Neither the name of FIRST nor the names of its contributors may be used to endorse or
 * promote products derived from this software without specific prior written permission.
```

```
*
* NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT RIGHTS ARE GRANTED BY THIS
* LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
* CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
* OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.Disabled;
import com.qualcomm.robotcore.hardware.Servo;
import com.qualcomm.robotcore.eventloop.opmode.OpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.util.ElapsedTime;
import com.qualcomm.robotcore.util.Range;

@TeleOp(name="Starter Bot 2024", group="Iterative Opmode")

public class StarterBot2024Teleop extends OpMode
{
    // Declare OpMode members.
    private ElapsedTime runtime = new ElapsedTime();
    private DcMotor leftDrive = null;
    private DcMotor rightDrive = null;
    private DcMotor armLeft = null;
    private DcMotor armRight = null;
    private Servo gripper = null;
    private Servo wrist = null;

    private boolean manualMode = false;
    private double armSetpoint = 0.0;

    private final double armManualDeadband = 0.03;

    private final double gripperClosedPosition = 1.0;
    private final double gripperOpenPosition = 0.5;
```

```
private final double wristUpPosition = 1.0;
private final double wristDownPosition = 0.0;

private final int armHomePosition = 0;
private final int armIntakePosition = 10;
private final int armScorePosition = 600;
private final int armShutdownThreshold = 5;

/*
 * Code to run ONCE when the driver hits INIT
 */
@Override
public void init() {
    telemetry.addData("Status", "Initialized");

    leftDrive  = hardwareMap.get(DcMotor.class, "leftDrive");
    rightDrive = hardwareMap.get(DcMotor.class, "rightDrive");
    armLeft  = hardwareMap.get(DcMotor.class, "armLeft");
    armRight = hardwareMap.get(DcMotor.class, "armRight");
    gripper = hardwareMap.get(Servo.class, "gripper");
    wrist = hardwareMap.get(Servo.class, "wrist");

    leftDrive.setDirection(DcMotor.Direction.FORWARD);
    rightDrive.setDirection(DcMotor.Direction.REVERSE);
    leftDrive.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.FLOAT);
    rightDrive.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.FLOAT);

    armLeft.setDirection(DcMotor.Direction.FORWARD);
    armRight.setDirection(DcMotor.Direction.REVERSE);
    armLeft.setMode(DcMotor.RunMode.STOP_AND_RESET_ENCODER);
    armRight.setMode(DcMotor.RunMode.STOP_AND_RESET_ENCODER);
    armLeft.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);
    armRight.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);
    armLeft.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
    armRight.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
    armLeft.setPower(0.0);
    armRight.setPower(0.0);

    telemetry.addData("Status", "Initialized");
}

/*
 * Code to run REPEATEDLY after the driver hits INIT, but before they hit PLAY
 */
```

```
@Override
public void init_loop() {
}

/*
 * Code to run ONCE when the driver hits PLAY
 */
@Override
public void start() {
    runtime.reset();

    armLeft.setMode(DcMotor.RunMode.STOP_AND_RESET_ENCODER);
    armRight.setMode(DcMotor.RunMode.STOP_AND_RESET_ENCODER);
    armLeft.setTargetPosition(armHomePosition);
    armRight.setTargetPosition(armHomePosition);
    armLeft.setPower(1.0);
    armRight.setPower(1.0);
    armLeft.setMode(DcMotor.RunMode.RUN_TO_POSITION);
    armRight.setMode(DcMotor.RunMode.RUN_TO_POSITION);
}

/*
 * Code to run REPEATEDLY after the driver hits PLAY but before they hit STOP
 */
@Override
public void loop() {
    double leftPower;
    double rightPower;
    double manualArmPower;

    //DRIVE
    double drive = -gamepad1.left_stick_y;
    double turn  =  gamepad1.right_stick_x;
    leftPower    = Range.clip(drive + turn, -1.0, 1.0) ;
    rightPower   = Range.clip(drive - turn, -1.0, 1.0) ;

    leftDrive.setPower(leftPower);
    rightDrive.setPower(rightPower);

    //ARM & WRIST
    manualArmPower = gamepad1.right_trigger - gamepad1.left_trigger;
    if (Math.abs(manualArmPower) > armManualDeadband) {
        if (!manualMode) {
            armLeft.setPower(0.0);
```

```
            armRight.setPower(0.0);
            armLeft.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
            armRight.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
            manualMode = true;
        }
        armLeft.setPower(manualArmPower);
        armRight.setPower(manualArmPower);
    }
    else {
        if (manualMode) {
            armLeft.setTargetPosition(armLeft.getCurrentPosition());
            armRight.setTargetPosition(armRight.getCurrentPosition());
            armLeft.setPower(1.0);
            armRight.setPower(1.0);
            armLeft.setMode(DcMotor.RunMode.RUN_TO_POSITION);
            armRight.setMode(DcMotor.RunMode.RUN_TO_POSITION);
            manualMode = false;
        }

        //preset buttons
        if (gamepad1.a) {
            armLeft.setTargetPosition(armHomePosition);
            armRight.setTargetPosition(armHomePosition);
            armLeft.setPower(1.0);
            armRight.setPower(1.0);
            armLeft.setMode(DcMotor.RunMode.RUN_TO_POSITION);
            armRight.setMode(DcMotor.RunMode.RUN_TO_POSITION);
            wrist.setPosition(wristUpPosition);
        }
        else if (gamepad1.b) {
            armLeft.setTargetPosition(armIntakePosition);
            armRight.setTargetPosition(armIntakePosition);
            armLeft.setPower(1.0);
            armRight.setPower(1.0);
            armLeft.setMode(DcMotor.RunMode.RUN_TO_POSITION);
            armRight.setMode(DcMotor.RunMode.RUN_TO_POSITION);
            wrist.setPosition(wristDownPosition);
        }
        else if (gamepad1.y) {
            armLeft.setTargetPosition(armScorePosition);
            armRight.setTargetPosition(armScorePosition);
            armLeft.setPower(1.0);
            armRight.setPower(1.0);
            armLeft.setMode(DcMotor.RunMode.RUN_TO_POSITION);
```

```
            armRight.setMode(DcMotor.RunMode.RUN_TO_POSITION);
            wrist.setPosition(wristUpPosition);
        }
    }

    //Re-zero encoder button
    if (gamepad1.start) {
        armLeft.setMode(DcMotor.RunMode.STOP_AND_RESET_ENCODER);
        armRight.setMode(DcMotor.RunMode.STOP_AND_RESET_ENCODER);
        armLeft.setPower(0.0);
        armRight.setPower(0.0);
        manualMode = false;
    }

    //Watchdog to shut down motor once the arm reaches the home position
    if (!manualMode &&
    armLeft.getMode() == DcMotor.RunMode.RUN_TO_POSITION &&
    armLeft.getTargetPosition() <= armShutdownThreshold &&
    armLeft.getCurrentPosition() <= armShutdownThreshold
    ) {
        armLeft.setPower(0.0);
        armRight.setPower(0.0);
        armLeft.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
        armRight.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
    }

    //GRIPPER
    if (gamepad1.left_bumper || gamepad1.right_bumper) {
        gripper.setPosition(gripperOpenPosition);
    }
    else {
        gripper.setPosition(gripperClosedPosition);
    }

    telemetry.addData("Status", "Run Time: " + runtime.toString());
    telemetry.addData("Gamepad", "drive (%.2f), turn (%.2f)", drive, turn);
    telemetry.addData("Motors", "left (%.2f), right (%.2f)", leftPower, rightPower);
    telemetry.addData("Manual Power", manualArmPower);
    telemetry.addData("Arm Pos:",
        "left = " +
        ((Integer)armLeft.getCurrentPosition()).toString() +
        ", right = " +
        ((Integer)armRight.getCurrentPosition()).toString());
    telemetry.addData("Arm Pos:",
```

```
            "left = " +
            ((Integer)armLeft.getTargetPosition()).toString() +
            ", right = " +
            ((Integer)armRight.getTargetPosition()).toString());
    }

    /*
     * Code to run ONCE after the driver hits STOP
     */
    @Override
    public void stop() {
    }

}
```