10 Ianuarie 2018	//de completat	10:30 - 20:15

Participanţi:

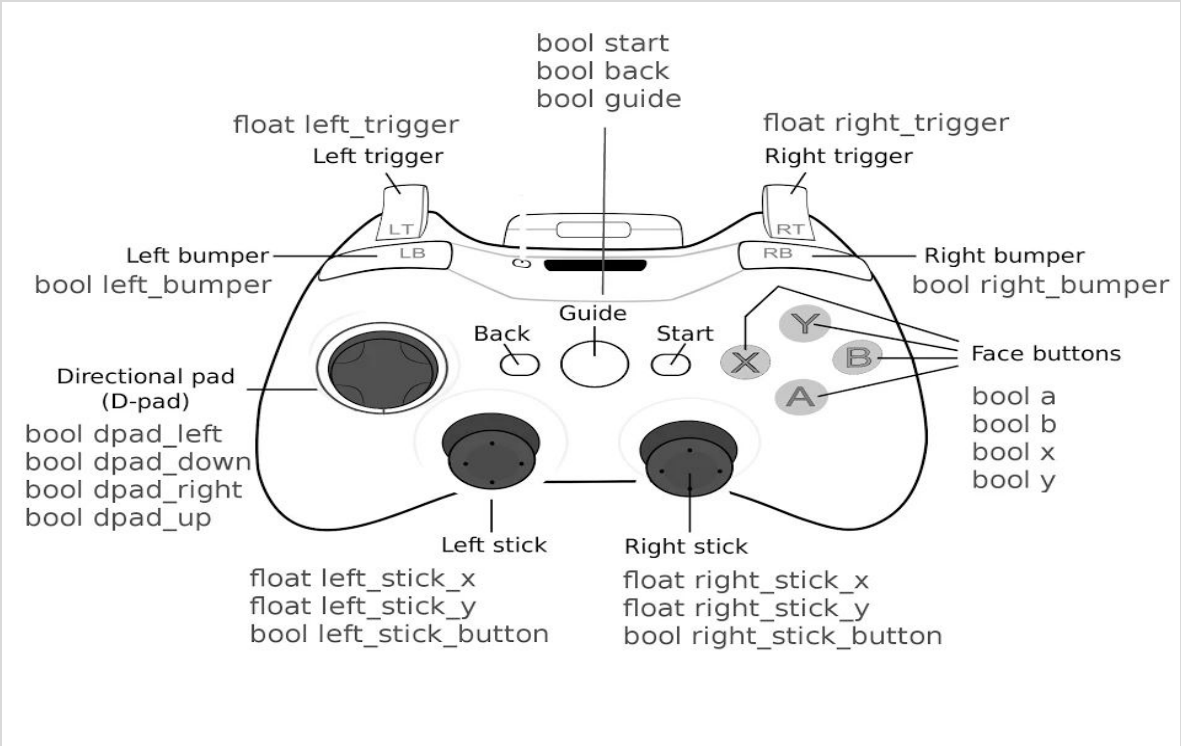
Coroian Sebastian, Costin Cristiana, Mathe Cezar;

Descriere:

Astăzi am decis să creăm un OpMode de tip TeleOp care să satisfacă toate capabilităţile robotului nostru. Acest OpMode va fi cel final folosit în competiţie, urmând să-l optimizăm şi să-l facem cât mai performant şi mai convenabil de folosit.

Explicaţii:

Pentru început, pentru a ne uşura munca, am căutat pe internet care sunt valorile pe care le putem citi de pe gamepad. Am găsit această ilustraţie pe reddit (https://goo.gl/hFEp7D):



Aici am declarat toate variabilele de care avem nevoie:

```java
// RUN TIME
private ElapsedTime runtime = new ElapsedTime();

//VARIABILE
private boolean x = false;

// MOVEMENT DcMotor
private DcMotor left_front = null;
private DcMotor left_back = null;
private DcMotor right_front = null;
private DcMotor right_back = null;
// QUBE INTAKE DcMotor
private DcMotor left_qubes = null;
private DcMotor right_qubes = null;
// LIFT DcMotor
private DcMotor lift = null;
// ROBOTIC ARM DcMotor
private DcMotor extender = null;

// RAMP SERVOS
private Servo ramp1 = null;
private Servo ramp2 = null;
// SENSOR SERVO
private Servo color_servo = null;
// QUBE INTAKE SERVO
private Servo qube_servo = null;
```

Printr-o funcţie scrisă de Cristiana am atribuit fiecărei variabile de hardware câte o componentă a robotului prin intermediului configuraţiei:

```java
// HARDWARE CONFIGURATION - Criss
private void hardwareCfg() {
    left_front = hardwareMap.get(DcMotor.class, "left_front");
    left_back = hardwareMap.get(DcMotor.class, "left_back");
    right_front = hardwareMap.get(DcMotor.class, "right_front");
    right_back = hardwareMap.get(DcMotor.class, "right_back");
    left_qubes = hardwareMap.get(DcMotor.class, "left_qubes");
    right_qubes = hardwareMap.get(DcMotor.class, "right_qubes");
    lift = hardwareMap.get(DcMotor.class, "lift");
    extender = hardwareMap.get(DcMotor.class, "extender");
    ramp1 = hardwareMap.get(Servo.class, "ramp1");
    ramp2 = hardwareMap.get(Servo.class, "ramp2");
    color_servo = hardwareMap.get(Servo.class, "color_servo");
    qube_servo = hardwareMap.get(Servo.class, "qube_servo");
}
```

Pentru mişcarea robotului omni, am optat în faza iniţială pentru un algoritm simplu, care permite mişcarea şi rotaţia, dar nu în acelaşi timp. Mişcarea robotului se realizează de pe gamepad-ul 1. Pentru mişcarea propriu-zisă, vom folosi stick-ul din stânga, iar pentru rotaţie vom folosi doar coordonata X a stick-ului din dreapta.

Pentru început preluăm valorile, după care, printr-un test logic, permitem fie rotația, fie mişcarea. De asemenea, funcţia de mişcare a robotului returnează statusul robotului pentru a putea adăuga în Telemetry dacă acesta se mişcă sau nu (vom aborda partea asta puţin mai târziu).

```java
// ROBOT MOVEMENT - Cezar
private boolean movementStatus() {
    double x = -gamepad1.left_stick_x;
    double y = gamepad1.left_stick_y;
    double rx = gamepad1.right_stick_x;

    if (rx > 0) {
        left_front.setPower(-rx);
        right_front.setPower(-rx);
        left_back.setPower(-rx);
        right_back.setPower(-rx);
    }
```

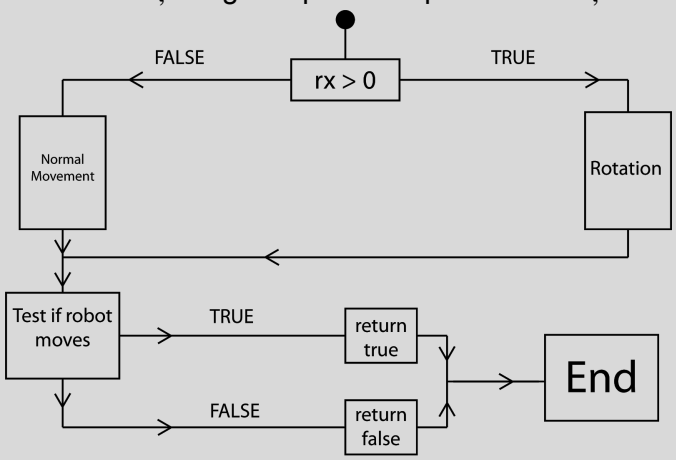
```java
    else {
        double fl = -y - x;
        double fr = y - x;
        double br = y + x;
        double bl = -y + x;

        fl = Range.clip(fl, -1, 1);
        bl = Range.clip(bl, -1, 1);
        fr = Range.clip(fr, -1, 1);
        br = Range.clip(br, -1, 1);

        left_front.setPower(fl);
        right_front.setPower(fr);
        left_back.setPower(bl);
        right_back.setPower(br);

    }
    if (x != 0 || y != 0 || rx != 0)
        return true;
    else
        return false;
}
```

Am realizat și un grafic pentru explicarea funcției de mișcare a robotului.



```
// QUBE INTAKE - Criss
private void qubeIntake() {

    if (gamepad1.x == true) {
    //de pe reddit
     x = !x;
    }
    if (x == true){
        left_qubes.setPower(1);
        right_qubes.setPower(1);
    }
    else {
        left_qubes.setPower(0);
        right_qubes.setPower(0);
    }


}
```

Deoarece am recurs la un model ce folosește două roți pentru a urca cuburile pe rampa (una de o parte, una de cealaltă), am creat funcția de mai sus.

- Primul “if” verifică dacă butonul **x** al gamepad-ului este apăsat, iar dacă acest lucru este adevărat, va atribui o altă valoare butonului pentru a îl menține oprit.
- Așadar, dacă butonul este apăsat, prin cel de al doilea “if”, vom trimite putere către motoare. Altfel, acestea nu vor primi putere.

```
// QUBE PLACING - Sebi
private void qubeLift() {
    while(gamepad2.right_trigger>0)
    {
        ramp1.setPosition(ramp1.getPosition()+15);
        ramp2.setPosition(ramp2.getPosition()+15);
    }
    while(gamepad2.right_trigger>0)
    {
        ramp1.setPosition(ramp1.getPosition()-15);
        ramp2.setPosition(ramp2.getPosition()-15);
    }
}
```

```
// RELIC - Sebi
private void relicLift() {
    extender.setPower(gamepad2.right_stick_y);

}
```

Aici pentru început vom configura variabilele ce conţin toate componentele hardware prin funcția hardwareCfg(), după care așteptam apăsarea butonului PLAY.

```
@Override
public void runOpMode() { // Cezar

    hardwareCfg();

    waitForStart();
    runtime.reset();
```

După aceea preluăm statusul robotului ca valoare returnată de funcţia de mişcare a acestuia. În continuare, vom apela fiecare funcţie a robotului(fiecare este explicată mai sus) după care finalizăm prin adăugarea valorilor în Telemetry. (Graficul se află pe pagina următoare)

```
while (opModeIsActive()) {

    boolean robot_status = movementStatus();

    qubeIntake();

    qubeLift();

    relicLift();

    String r_status;
    if (robot_status)
        r_status = "Moving";
    else
        r_status = "Stopped";

    telemetry.addData("Status", "Run Time:" + runtime.toString());
    telemetry.addData("Robot status", r_status);
    telemetry.update();
}
```

După terminarea codului am început să testăm funcţionarea servo-urilor ce se ocupă cu plasarea cuburilor de spumă în dulap. Prima impresie a fost bună, deşi după ataşarea rampei de metal ne-am dat seama că nu sunt suficient de puternice aşa că am asamblat o rampă nouă din aluminiu, mult mai uşoară.

După aceea am încercat să schimbăm unghiurile la care se învârt servo-urilor şi ne-am dat seama că ceva nu merge bine. Eu, Cezar, am căutat nişte videoclipuri pe YouTube, cu privire la programarea acestui tip de motor, însă Sebi a fost cel care a observat greşeala: servomotoarele acceptă unghiul în radiani de forma x*π , astfel încât x este cuprins în intervalul [0,1], şi nu în grade (ceea ce am făcut noi de fapt).