Curs_01_The_Emperor

Curs 1: GitHub, Java și Android Studio pentru FTC

Bun venit!

Acest curs te va introduce în instrumentele esențiale pentru programarea robotului FTC. La final, vei ști:

- Cum să lucrezi în echipă folosind GitHub
- Bazele limbajului Java
- Cum să navighezi în Android Studio

---

1. GitHub - Lucrul în Echipă

Ce este GitHub?

GitHub este o platformă care ne ajută să:

- **Salvăm** codul robotului în cloud (nu se pierde dacă se strică laptopul)
- **Lucrăm împreună** - mai mulți programatori pot lucra simultan
- **Vedem istoricul** - putem reveni la versiuni anterioare ale codului
- **Înțelegem modificările** - vedem cine a schimbat ce și când

Concepte Esențiale

Repository (Repo)

- Este "folderul" cu tot codul echipei
- Conține toate fișierele proiectului FTC
- Are un istoric complet al modificărilor

Commit

- O "salvare" a modificărilor tale
- Fiecare commit are un mesaj descriptiv
- Exemplu: `"Adaugat cod pentru mecanum drive"`

Push

- Trimiți commit-urile tale pe GitHub (în cloud)
- Acum toată echipa poate vedea modificările tale

Pull

- Descarci ultimele modificări de pe GitHub
- **Important:** Fă ÎNTOTDEAUNA pull înainte să începi să lucrezi!

Branch

- O "ramură" separată unde poți experimenta
- Branch-ul principal se numește `main` sau `master`
- Creezi un branch nou pentru funcții noi: `feature/sistem-intake`

Workflow-ul de Bază în Echipă

```
1. Pull (descarci ultimele modificări)
   ↓
2. Lucrezi pe codul tău
   ↓
3. Commit (salvezi local cu mesaj descriptiv)
   ↓
4. Pull din nou (verifici dacă cineva a făcut modificări între timp)
   ↓
5. Push (trimiți modificările tale pe GitHub)
```

Comenzi Git Esențiale (în terminal)

```bash
# Clonează repository-ul echipei (prima dată)
git clone https://github.com/echipa-ta/ftc-robot-2025.git

# Verifică statusul fișierelor
git status

# Adaugă fișierele modificate
git add .

# Salvează modificările cu un mesaj
git commit -m "Mesajul tau descriptiv"

# Descarcă ultimele modificări
git pull

# Trimite modificările pe GitHub
git push
```

Github Desktop (mai usor decat lucratul in terminal)

Puteti sa descarcati aceasta aplicatie, e mult mai vizuala si mai intuitiva decat terminalul https://desktop.github.com/download/

Mesaje de Commit Bune

✅ **Bune:**

- `"Adaugat sistem de control pentru intake"`
- `"Rezolvat bug la rotire autonomă"`
- `"Optimizat viteza motorului launcher"`

❌ **Rele:**

- `"Update"`
- `"asdf"`
- `"am modificat ceva"`

---

2. Introducere în Java

De ce Java pentru FTC?

FTC folosește Android phones pentru controlul robotului, iar Android se programează în Java (sau Kotlin). SDK-ul FTC este construit în Java.

Structura unui Program Java

```
public class MecanumDrive {
    // Aici declarăm variabilele
    private DcMotor frontLeft;
    private int vitezaMaxima = 100;

    // Constructor - se rulează când creăm obiectul
    public MecanumDrive() {
        System.out.println("Sistem de drive inițializat!");
    }

    // Metodă - o "funcție" care face ceva
    public void mergiInainte(double putere) {
        frontLeft.setPower(putere);
    }
}
```

Variabile și Tipuri de Date

```
// Numere întregi
int numarMotoare = 4;
int scor = 0;

// Numere cu zecimale
double putere = 0.75;
double distanta = 45.5;

// Text
String numeEchipa = "RoboLions";
String culoare = "albastru";

// Valori boolean (adevărat/fals)
boolean robotPornit = true;
boolean inAutonomous = false;

// Constant (nu se poate modifica)
final double VITEZA_MAX = 1.0;
```

Operatori

```
// Aritmetici
int suma = 5 + 3;         // 8
int diferenta = 10 - 4;   // 6
int produs = 6 * 7;       // 42
double impartire = 9.0 / 2.0;  // 4.5

// Comparare
boolean egal = (5 == 5);           // true
boolean diferit = (3 != 7);        // true
boolean maiMare = (10 > 5);        // true
boolean maiMicSauEgal = (4 <= 4);  // true

// Logici
boolean ambeleAdevarate = true && true;   // true (AND)
boolean celPutinUna = false || true;      // true (OR)
boolean opus = !true;                     // false (NOT)
```

Structuri de Control

If-Else

```
if (distantaSenzor < 10) {
    // Robotul este aproape de perete
    opreste();
} else if (distantaSenzor < 30) {
    // Robotul este la distanță medie
```

```
    mergiIncet();
} else {
    // Robotul are spațiu
    mergiNormal();
}
```

Bucle (Loops)

```
// For - când știi de câte ori să repeți
for (int i = 0; i < 5; i++) {
    System.out.println("Iterația " + i);
}

// While - cât timp o condiție e adevărată
while (robotPornit) {
    actualizeazaSenzori();
    if (butonOprireApasat) {
        robotPornit = false;
    }
}

// Bucla infinită (uzual în TeleOp)
while (opModeIsActive()) {
    // Codul se repetă continuu
    citestGamepad();
    actualizeazaMotoare();
}
```

Metode (Funcții)

```
// Metodă fără return
public void opreste() {
    motorStanga.setPower(0);
    motorDreapta.setPower(0);
}

// Metodă cu parametri
public void setPutere(double putereStanga, double putereDreapta) {
    motorStanga.setPower(putereStanga);
    motorDreapta.setPower(putereDreapta);
}

// Metodă care returnează o valoare
public double getDistanta() {
    return senzorDistanta.getDistance(DistanceUnit.CM);
}
```

```java
// Metodă cu parametri și return
public int calculeazaScor(int puncteAutonomous, int puncteTeleOp) {
    return puncteAutonomous + puncteTeleOp;
}
```

Clase și Obiecte (Introducere)

```java
// Clasa = șablon pentru un tip de obiect
public class Motor {
    private String nume;
    private double putere;

    // Constructor
    public Motor(String nume) {
        this.nume = nume;
        this.putere = 0.0;
    }

    // Getter
    public double getPutere() {
        return putere;
    }

    // Setter
    public void setPutere(double putere) {
        this.putere = putere;
    }
}

// Crearea unui obiect
Motor motorStanga = new Motor("Motor Stanga");
motorStanga.setPutere(0.8);
```

---

3. Android Studio pentru FTC

Ce este Android Studio?

Android Studio este **mediul de dezvoltare** (IDE) în care scriem codul pentru robotul FTC. Conține:

- **Editor de cod** - unde scriem Java
- **Debugger** - pentru găsirea erorilor
- **Gradle** - sistem de build care compilează codul
- **FTC SDK** - bibliotecile FTC pre-instalate

Instalare și Setup

1. **Descarcă Android Studio** de pe developer.android.com/studio
2. **Clonează FTC SDK-ul** de pe GitHub
3. **Deschide proiectul** în Android Studio
4. **Așteaptă Gradle Sync** - prima dată durează 5-10 minute

Structura Proiectului FTC

```
FtcRobotController/
├── TeamCode/
│   └── src/main/java/org/firstinspires/ftc/teamcode/
│       ├── AutonomieLaStanga.java      ← Programele tale Autonomous
│       ├── TeleOpPrincipal.java         ← Programele tale TeleOp
│       └── subsystems/                  ← Subsisteme (drive, intake, etc.)
│           ├── MecanumDrive.java
│           └── IntakeSystem.java
├── FtcRobotController/                  ← NU modifica aici
└── build.gradle                         ← Configurare Gradle
```

Anatomia unui OpMode FTC

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

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

@TeleOp(name="TeleOp Simplu", group="TeleOp")
public class TeleOpSimplu extends LinearOpMode {

    // Declarăm hardware-ul
    private DcMotor motorStanga;
    private DcMotor motorDreapta;

    @Override
    public void runOpMode() {
        // Inițializare - se rulează când apeși INIT
        motorStanga = hardwareMap.get(DcMotor.class, "motor_stanga");
        motorDreapta = hardwareMap.get(DcMotor.class, "motor_dreapta");

        telemetry.addData("Status", "Inițializat");
        telemetry.update();

        // Așteaptă start
        waitForStart();
```

```
        // Loop principal - se rulează continuu după START
        while (opModeIsActive()) {
            // Citește gamepad-ul
            double putereStanga = -gamepad1.left_stick_y;
            double putereDreapta = -gamepad1.right_stick_y;

            // Setează motoarele
            motorStanga.setPower(putereStanga);
            motorDreapta.setPower(putereDreapta);

            // Afișează informații pe Driver Station
            telemetry.addData("Stânga", putereStanga);
            telemetry.addData("Dreapta", putereDreapta);
            telemetry.update();
        }
    }
}
```

Navigare în Android Studio

Shortcuts Esențiale:

- `Ctrl + Space` - Auto-completare cod
- `Ctrl + /` - Comentează/decomentează linie
- `Ctrl + D` - Duplică linia curentă
- `Shift + F10` - Build și deploy pe robot
- `Alt + Enter` - Sugestii pentru rezolvarea erorilor

Paneluri Importante:

- **Project** (stânga) - structura fișierelor
- **Editor** (centru) - unde scrii cod
- **Logcat** (jos) - mesaje de debugging
- **Build** (jos) - erori de compilare

Debugging și Telemetry

```
// Telemetry - afișează pe Driver Station
telemetry.addData("Nume", valoare);
telemetry.addLine("Text liber");
telemetry.update();  // Actualizează afișajul

// Exemple
telemetry.addData("Putere motor", motorStanga.getPower());
telemetry.addData("Poziție encoder", motorStanga.getCurrentPosition());
telemetry.addData("Gamepad Y", gamepad1.left_stick_y);
```

4. Exerciții Practice

Exercițiul 1: GitHub Basics

Obiectiv: Înțelege flow-ul git

1. Creează un cont pe GitHub (dacă nu ai)
2. Clonează repository-ul echipei
3. Creează un fișier `TEST_[numeleTau].txt`
4. Adaugă un mesaj în fișier
5. Fă commit cu mesajul "Test commit de la [numele tău]"
6. Fă push pe GitHub
7. Verifică pe GitHub că fișierul apare

Exercițiul 2: Primul Program Java

Obiectiv: Scrie un program simplu

Creează un fișier `Calculator.java`:

```
public class Calculator {
    public static void main(String[] args) {
        // Declară variabile
        int numar1 = 10;
        int numar2 = 5;

        // Calculează
        int suma = numar1 + numar2;
        int diferenta = numar1 - numar2;
        int produs = numar1 * numar2;
        double impartire = (double) numar1 / numar2;

        // Afișează rezultatele
        System.out.println("Suma: " + suma);
        System.out.println("Diferența: " + diferenta);
        System.out.println("Produsul: " + produs);
        System.out.println("Împărțirea: " + impartire);
    }
}
```

Sarcină: Modifică programul să calculeze și restul împărțirii (`%`)

Exercițiul 3: Primul OpMode

Obiectiv: Creează un OpMode funcțional în Android Studio

1. Deschide Android Studio cu proiectul FTC
2. Navighează la `TeamCode/src/.../teamcode/`
3. Creează fișier nou `TestMotor.java`
4. Copiază scheletul de OpMode de mai sus
5. Modifică să controleze un singur motor cu stick-ul gamepad-ului
6. Build proiectul (`Shift + F10`)
7. Deploy pe Robot Controller
8. Testează pe robot!

Exercițiul 4: Telemetry și Debugging

Obiectiv: Învață să afișezi informații utile

Modifică OpMode-ul anterior să afișeze:

- Puterea motorului
- Valoarea stick-ului
- Poziția encoder-ului
- Numărul de loop-uri executate

---

5. Concepte Importante pentru FTC

HardwareMap

```
// Accesează hardware-ul definit în configurația robotului
DcMotor motor = hardwareMap.get(DcMotor.class, "nume_motor");
Servo servo = hardwareMap.get(Servo.class, "nume_servo");
DistanceSensor senzor = hardwareMap.get(DistanceSensor.class,
"nume_senzor");
```

Important: Numele din cod trebuie să fie EXACT ca în configurația de pe Robot Controller!

Ciclul de Viață al unui OpMode

```
1. INIT - runOpMode() începe, inițializezi hardware
2. INIT_LOOP - (opțional) cod care se repetă până la START
3. START - waitForStart() se termină
4. LOOP - while(opModeIsActive()) se repetă
5. STOP - când oprești manual sau timpul expiră
```

TeleOp vs Autonomous

TeleOp:

- Controlat de șoferi cu gamepad-uri
- Loop continuu care citește input
- Durează 2 minute în meci

Autonomous:

- Robotul se mișcă singur
- Programat în avans
- Durează 30 secunde la început de meci

---

6. Resurse Suplimentare

Documentație Oficială

- FTC Docs - Documentație oficială FTC
- FTC SDK GitHub - Codul sursă
- Game Manual - Regulile sezonului

Tutoriale Video

- Game Manual 0 - Ghid complet FTC (LECTURA OBLIGATORIE!)
- FTC YouTube - Tutoriale oficiale

Comunitate

- FTC Discord
- r/FTC - Subreddit FTC

---

7. Întrebări Frecvente

Î: De câte ori fac push pe GitHub? R: După fiecare sesiune de lucru. Regula: "Commit des, push la final."

Î: Ce fac dacă am conflict la pull? R: Calm! Android Studio îți arată unde e conflictul. Discută cu colegul care a modificat același fișier și decideți ce cod păstrați.

Î: De ce nu compilează codul? R: Verifică:

- Ai făcut Gradle Sync?
- Ai toate parantezele închise?

- Ai pus `;` la final de instrucțiune?
- Numele din hardwareMap corespund cu configurația?

Î: Robotul nu răspunde la comenzi! R: Verifică:

- OpMode-ul rulează? (vezi pe Driver Station)
- Hardware-ul e configurat corect?
- Motoarele sunt conectate fizic?
- Puterea bateriei e suficientă?
