RUMWaiter Mk2 - Git Cheat Sheet

■ Simple Workflow (main branch)

```
git status                    # See what’s changed
git add .                     # Stage all changes
git commit -m "Message"       # Commit with message
git push origin main          # Push to GitHub
git pull origin main          # Pull from GitHub
```

■ Branching Workflow (safer experiments)

```
git checkout -b feature/avr8js   # Create & switch branch
git add .
git commit -m "Initial avr8js integration"
git push -u origin feature/avr8js

# After first push:
git push
```

■ Keeping branch up to date

```
git checkout main
git pull origin main          # Update main
git checkout feature/avr8js
git merge main                # Merge main into feature
```

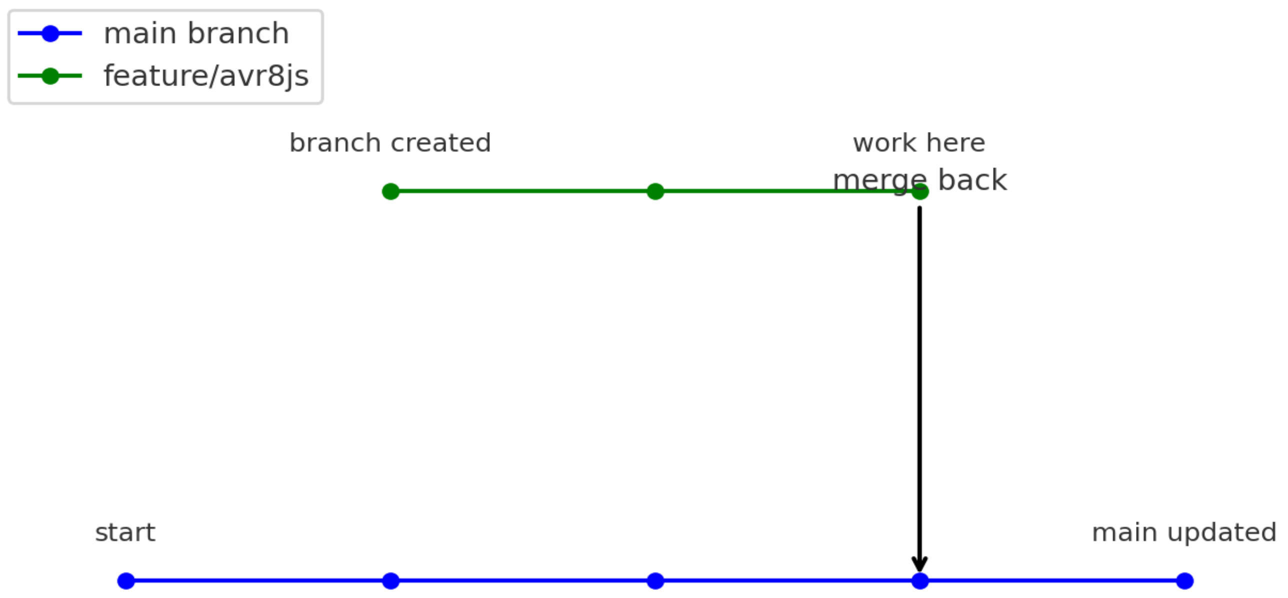
■ Merging feature branch back to main

```
git checkout main
git pull origin main
git merge feature/avr8js
git push origin main
```

■ Handy Extras

```
git log --oneline --graph --decorate  # Pretty history view
git diff                      # Show changes
git reset --soft HEAD~1       # Undo last commit (keep changes staged)
```

Branching Workflow Diagram



PNPM Install / Build / Run Guide

■ Setup

```
pnpm install         # Install all dependencies
pnpm -v              # Check pnpm version (should match spec)
```

■■ Build

```
pnpm build           # Build all packages in the monorepo
pnpm --filter ui dev  # Start the UI dev server (Vite)
pnpm --filter emu test  # Run emulator unit tests
```

■■ Run

```
pnpm start           # Start default app (if defined)
pnpm --filter ui dev  # Run UI simulator in dev mode
pnpm --filter physics test  # Run physics test suite
```

■■ Tips

```
pnpm workspace list   # Show packages in the monorepo
pnpm --filter <pkg> <cmd>   # Run cmd only in <pkg>
pnpm build --parallel       # Build all packages in parallel
```

Arduino CLI - Compile / Upload / HEX

■ Setup

```
arduino-cli version                # Check version
arduino-cli board list             # Detect connected boards
arduino-cli core install arduino:avr # Install AVR core (Uno/Mega)
```

■■ Compile

```
arduino-cli compile --fqbn arduino:avr:uno firmware/blink
arduino-cli compile --fqbn arduino:avr:mega firmware/rumwaiter
```

■■ Upload to Board

```
arduino-cli upload -p COM3 --fqbn arduino:avr:uno firmware/blink
arduino-cli upload -p COM4 --fqbn arduino:avr:mega firmware/rumwaiter
```

■ Export HEX (for emulator)

```
arduino-cli compile --fqbn arduino:avr:uno firmware/rumwaiter --output-dir ./build
Result: ./build/rumwaiter.ino.hex
```

■■ Tips

Use `--clean` with compile to force rebuild
Use `arduino-cli config init` to create a global config file
HEX files in ./build are what avr8js will load

Emulator (avr8js) Workflow

■ Setup

pnpm --filter emu add avr8js   # Install avr8js in emulator package
Ensure firmware HEX files are exported via Arduino CLI

■■ Run Emulator

pnpm --filter emu dev   # Start emulator dev mode
Load firmware HEX into avr8js instance
Tie Arduino pins -> physics & sensor adapters

■ Development Loop

1. Edit Arduino sketch in firmware/
2. Compile & export HEX with arduino-cli
3. Reload HEX in emulator (no code changes needed in sim)
4. Observe motor physics, sensors, OLED UI in sim

■■ Tips

Use fixed 1 ms step mode for deterministic runs
Use real-time mode for interactive UI testing
Log outputs to CSV/JSON for regression tests