Romi Workshop Volunteer Details

Thank you for helping us support growing the interest and knowledge of our younger generation in STEM. For the Romi Workshop, your main duties will be helping the kids/teams with their programming (Java based). Below is some information that should help guide you through that process. If you are ever in doubt, please reach out to one of the head mentors (Zhiquan Yeo or Joe Pokorny) and they can provide next steps.

**Get to know the Hardware:**  
We are using a Romi board (<https://www.pololu.com/product/4022>). This comes with a chassis, wheels, motors (and encoders for them), and a microcontroller with 5 I/O pins we can use (black bank of male pins). We have added a raspberry pi (green board) that sits on top of it and acts as the communication channel between the laptop and robot via a wireless interface.  
  
**Steps for setting up and connecting to the Romi:**One of the biggest challenges is getting the Romi fully initialized, setting up your laptop, connecting to the Romi, and finally running a test program. It is a big milestone getting everything fully operational and probably the place where we will spend most of our time. Below are a summarized list of the full documentation found here: <https://docs.wpilib.org/en/stable/docs/romi-robot/index.html>

1. Note - the hardware for the Romi has already been assembled and the microSD cards imaged.
2. **Turn the Romi on** - (push button in lower left corner). It should light up and make a noise when it first turns on.
3. **Connect to Romi** - Wait until the **access point WPILibPi-xxxxx** (where xxxxx is the value associated with what is written on tape on the Romi). Connect to this access point with **password WPILib2021!**
4. **Setup Firmware** - in a browser window, go to [**http://10.0.0.2**](http://10.0.0.2).
   1. If you do not see a UI show, it means you are not connected to the Romi.
   2. Click on **Romi** in the left tab.
   3. If you see “Firmware Compatible” as “Yes”, you do not need to do anything more for this step.
   4. If not, you need to connect the USB cable from the raspberry pi (any port but must be the USB-A, bigger plug) to the Romi micro USB-B (Romi is the black board, the port is on the front of the board).
   5. Scroll down in the browser to the “**Update Firmware**” button. If the wire was set up correctly on the robot this should be enabled. Click on it.
   6. If it works you should see in the log:  
      avrdude xxxxx bytes of flash verified  
      avrdude done. Thank you.
   7. If you see a failure about ttyACM0, please see troubleshooting
5. Launch **2022 WPILib VS Code** - If it is not on the computer, please download here:  
   <https://github.com/wpilibsuite/allwpilib/releases/tag/v2022.4.1>
6. **Leverage the romireference example to test**
   1. Follow directions here or the summary steps below: <https://docs.wpilib.org/en/stable/docs/romi-robot/programming-romi.html>
   2. On the far upper right corner, click on the wpilib icon (w with a red hexagon around it)
   3. Type in “**Create a new project**”
   4. Project Type: Example
   5. Language: Java
   6. Project Base: RomiReference
   7. Select a folder
   8. Name the project
   9. Give it any team number
   10. Generate the new project
   11. On the far upper right corner, click on the wpilib icon (w with a red hexagon around it)
   12. Type “**Simulate Robot Code on Desktop**”
7. **Make sure Robot is on the ground!**
8. When the Simulator UI launches, switch it to **Autonomous mode** (upper left of UI). Romi should move.
   1. If Simulator UI doesn’t launch or robot doesn’t move see Troubleshooting

**Programming**  
We will be using Java and a style called command-based programming, see here:  
<https://docs.wpilib.org/en/2022/docs/software/commandbased/index.html>

The RomiReference example is in this style already and we plan to build upon it. Other examples can be found at:

1. Basic Telop Driving for Romi Example:  
   <https://github.com/bb-frc-workshops/wpilib-ws-romi-command>
2. Servo Arm example for Romi Example:

<https://github.com/bb-frc-workshops/romi-examples/tree/main/romi-simple-servo>

There are several classes in wpilib that you can leverage. You can find the documentation on them here:  
<https://first.wpi.edu/wpilib/allwpilib/docs/release/java/edu/wpi/first/wpilibj/package-summary.html>

And the complete list here:  
<https://first.wpi.edu/wpilib/allwpilib/docs/release/java/index.html>

Suggestion is to leverage the examples as references or starting points of code and go from there.

**RobotContainer.java** - Where you will instantiate the subsystem and specify the default command(s).

**Robot.java** - skeleton of the different calls based on the Mode you are in (Teleoperated, Autonomous, or Test). There is an init and periodic for each. You can specify the command(s) you want to run for each mode (note, cancelling a command restores the default to that subsystem).

**subsystem folder** - where all your “hardware” components will be. For example, Drivetrain.java represents the interfacing with the hardware for your wheels. It is recommended that the students create a new subsystem here if they decided to add an actuator (Arm.java, Lifter.java, Intake.java, etc)

**commands folder** - where all the commands you want to provide to the robot are. These commands can be independent or done as a grouping. For example, AutonomousDistance.java shows how you can create a set of commands to be run in sequence.

**Troubleshooting:**

1. **When in doubt, try changing the batteries** - (6 AAs on bottom of Romi). If the Romi’s making a sound, things are moving that used to, or the wireless connection is dropping, most likely it’s due to power. New batteries solve a lot of issues that just don’t make sense.
2. **Firmware ttyACM0 error** - This could mean that you have a bad cable, or bad cable connection, or the microSD didn’t initialize correctly and doesn’t recognize the USB ports on the raspberry pi. First try reconnecting the cable, then try a different cable, and if that fails, ask for a freshly imaged microSD card from one of the head mentors.
3. **Simulation build error** - “WPIHaljni” dependency path not found  
   We are still working on this. If you see it, please call a head mentor over to capture the details. In the meantime, the workaround is to not simulate from the option in the drop-down, but rather after the failure, in the terminal in VS Code, type:  
   ./gradlew simulateJavaRelease
4. **Simulation build error** - All others  
   Something is wrong with the Java code. Please investigate the error outputs.
5. **Robot not moving -** 
   1. First check to see if the robot is connected (in the terminal in VS Code you should see a log output that it has “**HALSimWS: WebSocket Connected**”. If you see a bunch of “**Connection Attempt x**”, it is not connected to your robot.
   2. In Simulator UI, check that you are in “**Teleoperated**” mode
   3. In Simulator UI, check that your Joystick is mapped to **Joystick[0]** . If it says “Unassigned”, drag-and-drop the Logitech/Xbox from the “System Joysticks” on the left.
   4. In Simulator UI, verify that pressing buttons and joystick knobs do in fact light up in your Joystick[0] area. If not, try unplugging, and switching the game interface mode switch on the back of the controller from “X” to “D” or “D” to “X”. This switch changes the button/axis mappings for the controller.
   5. Verify the code. Did you map the buttons correctly in code (most likely RobotContainer or a command file)? Did you make sure that the command is executed (either directly or via a defaultCommand)? If it is to run in a loop continuously, did you ensure your command has an override isFinished returning false (like ArcadeDrive)?
   6. See point #1 again
6. Servo not moving
   1. Make sure servo brown wire is connected to the GND pin on the board. It is easy to accidentally plug this in backwards.
   2. Make sure the port specified matches the port mapping between the configuration on the web UI (<http://10.0.0.2>) and what's used in the servo initialization in the Java code "new Servo(<port>)"
   3. Make sure to set the default command of the servo/arm subsystem is set in RobotContainer initialization, or the configureButtonBindings in the simpleServo example.
   4. Make sure the servo/arm command isFinished method returns false so that it's always running.
   5. Make sure to set the values between 0 and 1.
   6. Try a new servo (they can burn out)
   7. See point #1 again