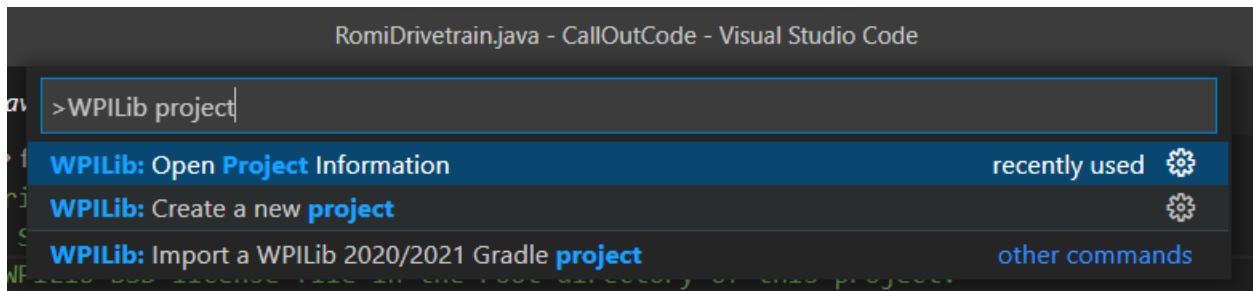
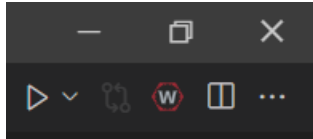


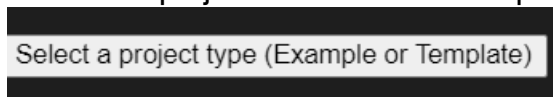
## Setting up the project

In WPILIB VS Code create a new project, press the hexagon in the top right and type project in the search bar

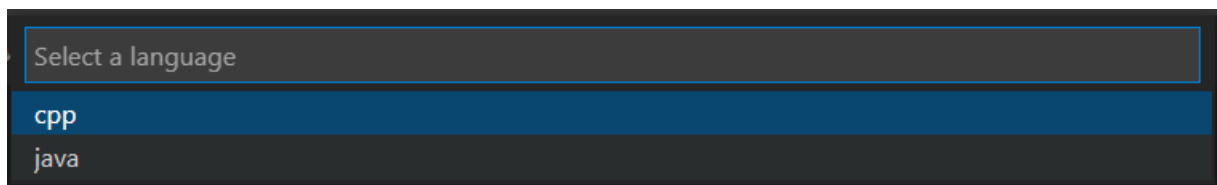


Click Create a new project

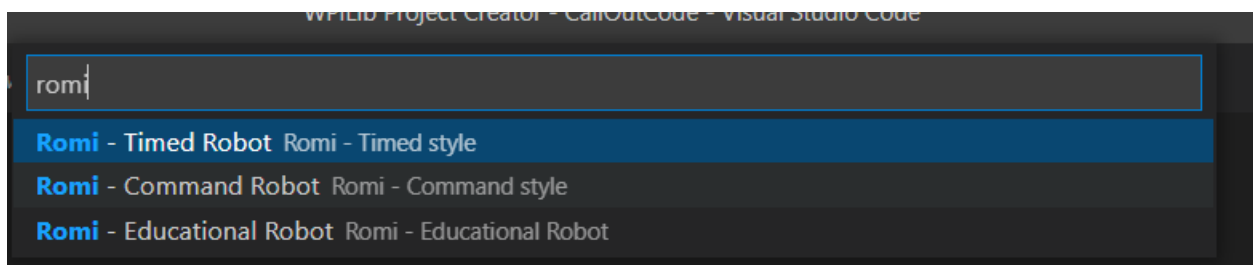
Then in the project screen click Select project type



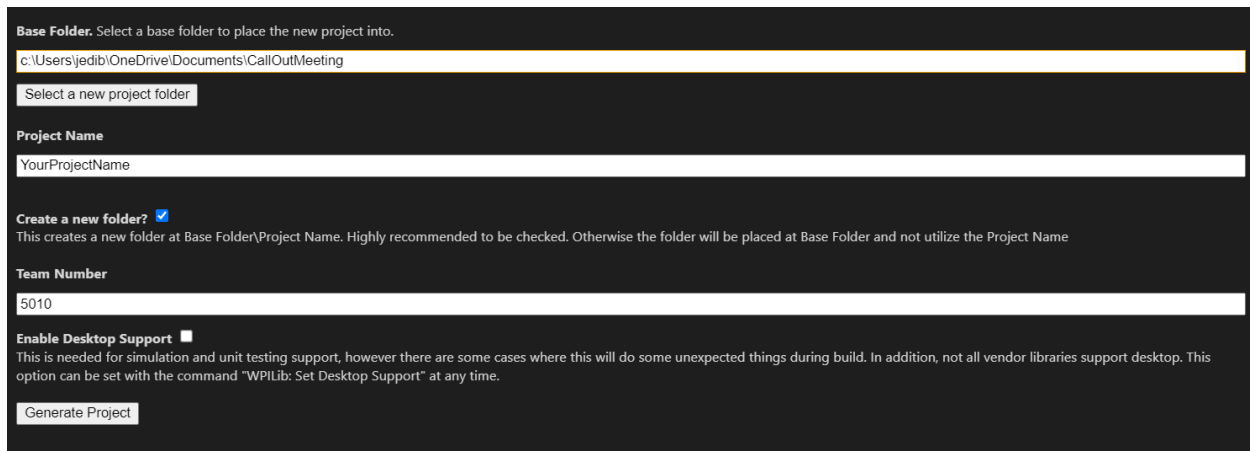
Click Template then java



Search Romi and select the Command Robot



Then put all this information in the project making sure to choose the CallOutMeeting folder in documents for the code, name the project whatever you like (Preferable not “CallOutCode” or “CallOutMeeting”)

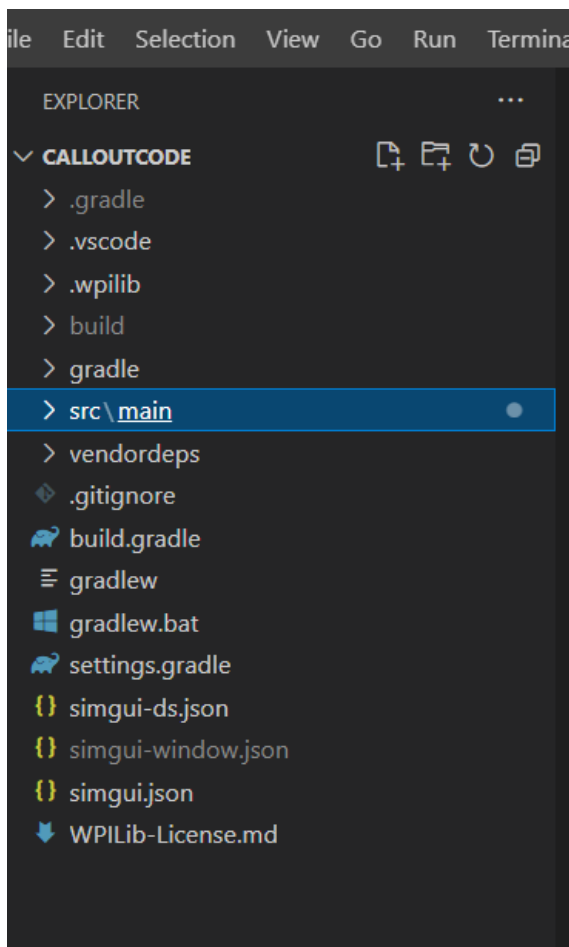


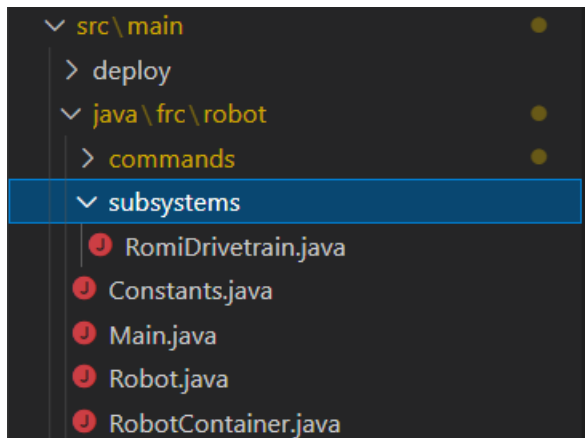
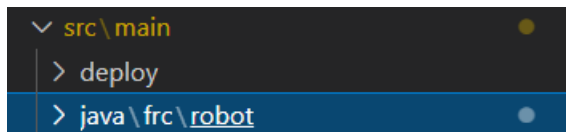
The screenshot shows a 'Generate Project' dialog box with a dark theme. It contains the following fields and options:

- Base Folder:** A text field with the value 'c:\Users\jedib\OneDrive\Documents\CallOutMeeting'. Below it is a button labeled 'Select a new project folder'.
- Project Name:** A text field with the value 'YourProjectName'.
- Create a new folder?** A checkbox that is checked. Below it is a note: 'This creates a new folder at Base Folder\Project Name. Highly recommended to be checked. Otherwise the folder will be placed at Base Folder and not utilize the Project Name'.
- Team Number:** A text field with the value '5010'.
- Enable Desktop Support:** A checkbox that is unchecked. Below it is a note: 'This is needed for simulation and unit testing support, however there are some cases where this will do some unexpected things during build. In addition, not all vendor libraries support desktop. This option can be set with the command "WPILib: Set Desktop Support" at any time.'
- Generate Project:** A button at the bottom.

Click Generate Project, then this window, and trust the project

When the project opens you are going to want to expand the SRC, then Java, then Subsystem drop downs





## Programming the ROMIs

There are two files you are going to want to add code into to make program the Romis

The first one is RobotContainer.java

You are going to need to add this function to the bottom of the code but still inside the class

```
public RomiDrivetrain getDrive(){  
    return m_romiDrivetrain;  
}  
} //End Of Class
```

You are doing this so that you can have the romiDrivetrain in RobotContainer in different classes

The next file you need to open is Robot.java and there is a bit more code to add to this one at the top you need to import the Joystick and RomiDrivetrain class next to the other import statements and just make sure all the imports here are there

```
//Imports  
import edu.wpi.first.wpilibj.TimedRobot;  
import edu.wpi.first.wpilibj2.command.Command;  
import edu.wpi.first.wpilibj2.command.CommandScheduler;  
import frc.robot.subsystems.RomiDrivetrain;  
import edu.wpi.first.wpilibj.Joystick;
```

then at the top of the Robot class you need to add the romiDrivetrain, driver, forward, and turn variables

```
public class Robot extends TimedRobot {
    private Command m_autonomousCommand;

    //variables
    private RobotContainer m_robotContainer;
    private RomiDrivetrain romiDrivetrain;
    private Joystick driver;
    private double forward;
    private double turn;
```

Then in robotInit() you must add two lines to initialize the variables romiDrivetrain and driver when the code starts

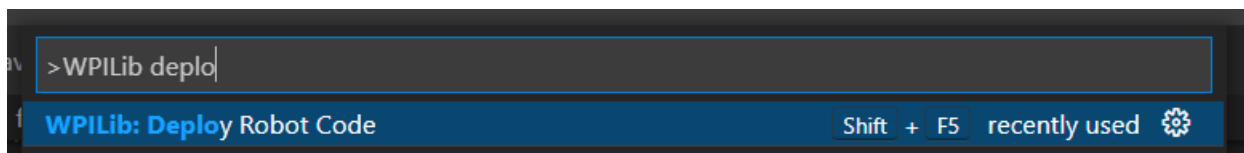
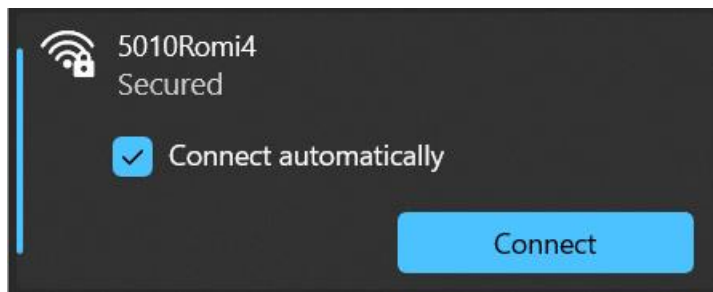
```
@Override
public void robotInit() {
    // Instantiate our RobotContainer. This will perform all our button
    bindings, and put our
    // autonomous chooser on the dashboard.
    m_robotContainer = new RobotContainer();
    //Initiallized functions
    romiDrivetrain = m_robotContainer.getDrive();
    driver = new Joystick(0);
}
```

Finally, in robotPeriodic() you set the values of forward and turn using the driver joystick values and setting them to the drivetrain using romiDrivetrain.arcadeDrive(forward, turn);

```
@Override
public void robotPeriodic() {
    //Getting axis values
    forward = driver.getRawAxis(1);
    turn = driver.getRawAxis(4);
    //Setting them to the driveTrain
    romiDrivetrain.arcadeDrive(forward, turn);

    CommandScheduler.getInstance().run();
}
```

Now all you should have to do is connect to the Romi through Wi-Fi(The password should be the name of the Wi-Fi, and the Romi number is written on the bottom), deploy the code and simulate (Both are done with the hexagon mentioned at the top) Don't forget to connect a controller!



Last thing you need to do is inside the simulate window you need to drag your controller to controller 0 then press Teleoperated

