Procedure to Configure Team 1895, Windows 11 Laptops for Python in 2023

Purpose:

This document describes how to set up our Windows 11 laptops for programming the robots in python. This is a very simple configuration just to get a team started in python.

This document also describes how to create and run a very simple Python program to verify the operation of the wheel motors, Joystick, shaft encoder, gyro and range finder.

Robot and Laptop Configuration

The Robotics team laptop is initially connected to the Robot RoboRIO using a USB cable.

The Robotics team software practice robot is an old AndyMark frame configured with a V1 RoboRIO, a single Falcon 500 motor on each side of the robot, a NAVX Gyro mounted on the RoboRIO and an analog range finder sensor on the front of the robot. See the constants.py for the interface numbers.

A simple "Logitech Attack 3" joystick is connected to the laptop. (Old style Joystick)

Reference Documents:

https://robotpy.readthedocs.io/en/stable/getting_started.html
https://github.com/robotpy/robotpy-ctre/blob/main/examples/basic/robot.py

Overview of Process:

1. Set up the laptop with the FIRST Robotics basic tools (VSCode and FRC Driver Station)
2. Set up the laptop to support Python
3. [One time per RoboRIO] Configure the RoboRIO to support Python
4. Verify operation by deploying a simple python program verify robot operation

[Hard] Lessons Learned:

1. Do NOT select [all] when selecting library modules to download into RoboRIO v1. This will fill the file system of a V1 RoboRIO. Recover by SSH'ing onto the RoboRIO, removing excess files and re-imaging the RoboRIO

Troubleshooting:

1. SSH into to RoboRIO to see the log under /home/lvuser/FRC-?????? Or view the 'Console' on the Drivers Station. [USB IP address of "roboRIO-1895-frc.local", login name of "admin" and no password.]

==========

Set up the laptop with the basic FIRST Robotics support tools *(vscode and driverstation)*

Reference Document: https://docs.wpilib.org/en/latest/docs/zero-to-robot/step-2/index.html

1. Locate the installation files at: "c:_robotics\FRC_and_Vendor_Software_for_Dev". [Specific for Fall 2023]
   [This folder contains the files ni-frc-2023-game-tools_23.1.0_offline.iso and WPILib_Windows-2023.4.3.iso]

2. Install "2023 FRC Game Tools". Open **ni-frc-2023-game-tools_23.1.0_offline.iso** and run **install.exe**. (Use the defaults) SKIP ACTIVATION BY SELECTING "cancel", **Reboot when done**

3. Install "Java/C++ WPILib Installer". Open "**WPIlib_Windows_20234.3.iso**". Browse to the D: drive and run "**WPILibInstaller.exe**". (SELECT "MORE INFO" then "RUN ANYWAY")

   Select "**Everything**" and "**Install for all users**", Select "**Download for this computer only**"

Set up the laptop to support Python

Reference: https://robotpy.readthedocs.io/en/stable/install/computer.html#install-computer

4. Download python from https://www.python.org/downloads/windows/

5. Install Windows Python on the laptop

Install RobotPy on the Laptop

6. Install robotpy using the command:

   py -3 -m pip install robotpy

7. Install third-party robotpy extensions using the command:

   py -3 -m pip install robotpy[ctre,navx,commands2]

Configure the RoboRIO to support Python [This shaded section only needs to be done one time on a RoboRIO]

Reference: https://docs.wpilib.org/en/latest/docs/zero-to-robot/step-3/imaging-your-roborio.html

8. Connect the laptop to the RoboRIO using a USB-A cable

9. Start the RoboRIO imaging tool

10. Verify the RoboRIO image is at 2023_v3.2. **IF NOT, UPDATE THE ROBORIO IMAGE**

11. Open a Command Prompt (select Win-R => CMD)

12. Download Python onto the Windows Laptop for deployment to the RoboRIO using the command within the command Window: ***(See the note following this command)***

    py -3 -m robotpy_installer download-python

    The MCPS Network "Content Filter" will cause the installation to fail since the "Content Filter" installs a self-signed X.509 certificate.

    This problem can be resolve by temporally using your cell phone as a "hot spot" to bypass the school filter on the laptop you are using to download the python file.

13. Install Robot python on the RoboRIO using the command:

    py -3 -m robotpy_installer install-python

14. Download RobotPY onto the Windows Laptop for deployment to the RoboRIO using the command:
    (See the note following this command)

    py -3 -m robotpy_installer download robotpy[ctre,navx,commands2]

    The MCPS Network "Content Filter" will cause the installation to fail since the "Content Filter" installs a self-signed X.509 certificate. This problem can be resolve by temporally using your cell phone as a "hot spot" to bypass the school filter on the laptop you are using to download the python file.

15. Install Robot python using the command:

    py -3 -m robotpy_installer install robotpy[ctre,navx,commands2]

Verify operation by deploying a simple python program

16. Open Windows File Explorer [Win-E] and browse to **c:_Robotics\2023**

17. Create a folder called **c:_Robotics\2023\Python_Checkout**

18. Start VSCode and select Folder Open and browse to **c:_Robotics\2023\Python_Checkout** and select "**Select Folder**".

19. Add the Python Extension by selecting the top menu option **View** => **Extensions**. Type in "**Python**" and select **Install**.

20. Within VSCode, create a **new file** called "robot.py" by selecting "**File**" => "**New File**" => "**Python File**".

21. Save the file by selecting "File" => "Save As" and entering **robot.py**.

22. Copy the code listed below and paste it into the robot.py.

```
#!/usr/bin/env python3

import wpilib
import wpilib.drive
import ctre
import constants
from navx import AHRS

class MyRobot(wpilib.TimedRobot):

    def robotInit(self):
        """
        This function is called upon program startup and
        should be used for any initialization code.

        """
        self.stick = wpilib.Joystick(constants.OIConstants.kDriverControllerPort)
        self.timer1 = wpilib.Timer()
        self.timer1.start()

##############
        # Instantiate the motors
        self.leftside_motor = ctre.WPI_TalonFX(constants.DriveConstants.LEFT_MOTOR_ID)
        self.rightside_motor = ctre.WPI_TalonFX(constants.DriveConstants.RIGHT_MOTOR_ID)

        # Reset all motor configuration to the default value
        self.leftside_motor.configFactoryDefault()
        self.rightside_motor.configFactoryDefault()

        # Reverse the direction of one motor since they are mounted in opposite direction
        self.leftside_motor.setInverted(False)
        self.rightside_motor.setInverted(True)

        # Set Coast versus brake (Forces the motor to stop when set zero)
        self.leftside_motor.setNeutralMode(ctre.NeutralMode.Coast)
        self.rightside_motor.setNeutralMode(ctre.NeutralMode.Coast)

        # Set up Motor groups
        # The motors on the left side of the drive.
        self.leftMotors = wpilib.MotorControllerGroup(self.leftside_motor)

        # The motors on the right side of the drive.
        self.rightMotors = wpilib.MotorControllerGroup(self.rightside_motor)

        # The robot's drive
        self.drive = wpilib.drive.DifferentialDrive(self.leftMotors, self.rightMotors)

#############################################################################################################
##      Enable the Falcon 500 built-in Shaft Encoders

        self.leftside_shaft_Encoder_Value = self.leftside_motor.getSensorCollection()
        self.leftside_shaft_Encoder_Value.setIntegratedSensorPosition(0)
        self.rightside_shaft_Encoder_Value = self.rightside_motor.getSensorCollection()
        self.rightside_shaft_Encoder_Value.setIntegratedSensorPosition(0)

        # How to read the encoder
        # self.leftshaft_Encoder_Values.getIntegratedSensorPosition()

#############################################################################################################
##      Enable the NAVX Gyro
        self.ahrs = AHRS.create_spi()
        #  self.ahrs.reset()
        #  self.ahrs.getYaw()

        self.front_Range_Finder = wpilib.AnalogInput(0)
        front_Range_Finder_Value = self.front_Range_Finder.getValue()

#############################################################################################################

    def autonomousInit(self):
        """This function is run once each time the robot enters autonomous mode."""
        self.timer.reset()
        self.timer.start()
```

```
    def autonomousPeriodic(self):
        """This function is called periodically during autonomous."""

        # Drive for two seconds
        if self.timer.get() < 2.0:
            self.drive.arcadeDrive(-0.5, 0)  # Drive forwards at half speed
        else:
            self.drive.arcadeDrive(0, 0)  # Stop robot

    def teleopInit(self) -> None:
        return super().teleopInit()
        self.ahrs.reset()
        print ("Reset Gyro")

    def teleopPeriodic(self):
        """This function is called periodically during operator control."""
        self.drive.arcadeDrive(-self.stick.getY(), self.stick.getX())

        if self.timer1.hasElapsed(0.5):
            right_side_Encoder_count = self.rightside_shaft_Encoder_Value.getIntegratedSensorPosition()
            left_side_Encoder_count = self.leftside_shaft_Encoder_Value.getIntegratedSensorPosition()
            heading_Angle = self.ahrs.getYaw()
            front_Range_Finder_Value = self.front_Range_Finder.getValue()
            print ("Left Encoder: %8d   Right Encoder: %8d    Heading: %5.1f   Range: %4d" %
                   (left_side_Encoder_count, right_side_Encoder_count, heading_Angle, front_Range_Finder_Value))
            self.timer1.reset()

if __name__ == "__main__":
    wpilib.run(MyRobot)
```

23. Within VSCode, create a new file called “contants.py” by selecting “**File**” => “**New File**” => “Python File”.

24. Save the file by selecting “File” => “Save As” and entering **constants.py**.

25. Copy the code listed below and paste it into the constants.py.

```
import math

class DriveConstants:
    LEFT_MOTOR_ID = 2
    RIGHT_MOTOR_ID = 1

    kLeftEncoderReversed = False
    kRightEncoderReversed = True

    kEncoderCPR = 1024
    kWheelDiameterInches = 6

    # Assumes the encoders are directly mounted on the wheel shafts
    kEncoderDistancePerPulse = (kWheelDiameterInches * math.pi) / kEncoderCPR

class OIConstants:
    kDriverControllerPort = 0
```

26. Save the update files by selecting “Save ALL” in VSCode.

Deploy the Python code to the robot

27. Open a command prompt on the Developers laptop using the command Win-R, enter CMD and select **OK**.

28. Within the command prompt, change directory to the folder where the code is using the command:

    cd c:_Robotics\2023\Python_Checkout

29. Within the command prompt, deploy the python code the Robot using the command:

    py -3 robot.py deploy

30. Start the Drivers Station application by double-clicking on the desktop icon: "FRC Drivers Station"

Safety Precautions:

1) VERIFY THE ROBOT'S WHEELS CAN TURN WITHOUT TOUCHING ANYTHING OR THE ROBOT MOVING
2) WARN YOUR TEAM MEMBERS BY STATING "ENABLING"

31. Select "**Teleoperated**"

32. Select "**Enable**"

33. Move the Joystick and verify the wheels move.

34. Open the Driver's Station Console by right-clicking on the "**Gear**" icon near top middle-right side and selecting "**View Console**".

35. On the console, verify the shaft encoder values are changing

36. On the console, verify the gyro heading value is changing when you change the robots heading

37. On the console, verify the range finder value is changing when you place your hand in front of the ultrasonic sensor on the front of the robot.

38. Disable the robot on the Driver's station.

Configure the laptop for a wireless operation to the Robot

39. Set the wired Ethernet IP address of the laptop to 10.18.95.5, Subnet mask to 255.0.0.0, gateway to 10.18.95.1 and DNS to 8.8.8.8.

40. Disconnect the USB cable.

41. On the laptop, select the wireless network associated with the robot. Inspect the radio for the number.

42. Verify the operation of the robot with a wireless connection

Going Further:

Now that the basic operation of the robot is verified, create "Command Based" robot code with a DriveTrain Subsystem and a few commands to move the robot in Teleoperated mode and Autonomous mode.