Team 3176 White Paper

Title: Programming Pneumatics	Authors: Jacob Aldridge, Elijah Furuness, Ben McCarty, Anna Nierzwick
Subteam: Programming	Subject: Pneumatics
Season: 2018/19	White Paper Number 2

Abstract

This paper documents the process for programming pneumatics to be used in FRC robots. This paper is written for the general FIRST community interested in learning the basics of programming pneumatics for actuating pistons.

Definitions

Pneumatics - Refers to the use of compressed air to create mechanical motion.

Solenoids - Refers to internal valves used to control a cylinder using air pressure.

Compressor - Refers to a device that regulates air pressure in a pneumatics system.

Pistons - Refers to a device that is able to extend and retract that, in this case, is controlled by pneumatics.

Pneumatics Control Module (PCM) - The board that communicates with the Solenoids via wires.

Process

One should always start the coding process with research. This is important for many reasons because it allows you to understand what you should do, and so you can learn from other people's experiences and mistakes.

Research

Considering pneumatics (the code portion, at least) is easy to set up; research was relatively quick. Once the necessary imports and methods were found we had sufficient enough info to begin programming. Our primary source was the official WPILIB Pneumatics guide.

Issues

A common issue popped up after the Electrical team rewired the pneumatics testing board. Suddenly, the pistons did not operate as programmed. In fact, they did the opposite as they were programmed to do. It turned out the Electrical team plugged in the Solenoids to different ports, which jeopardized the code that was written. There were multiple ways to fix the issue, written below in order from easiest to most difficult:

1. Change the ports in the code.
2. Change the pneumatic tubing connected to the solenoids
3. Change the ports on the PCM

Pneumatics Coding

After you have obtained all the knowledge about what pneumatics and are aware of the possible problems, you are ready to start coding.

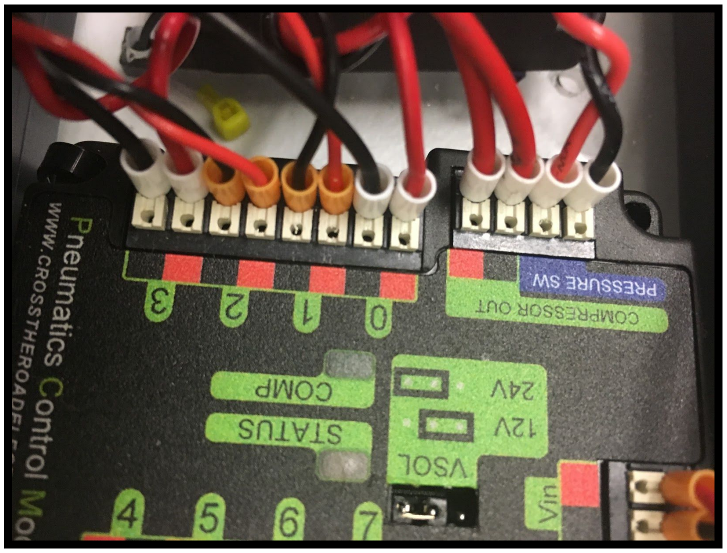
Getting Started

There are a few classes that are used when programming pneumatics.

1. Solenoid: This class controls the pistons by setting the Solenoids either true or false.
2. Compressor: Manages the functions of the compressor. In some cases, controlling the compressor is not necessary, so this class in not always needed when programming.
3. Joystick: A joystick is required for communicating with the Solenoids and the Compressor.

Instantiating Solenoids

Whilst instantiating Solenoids, you must first know the port that the Solenoids are plugged into. Look at the COMP side of the PCM and find the corresponding ports within which the solenoids are connected.



In the image, all four ports are being used, so you must instantiate four Solenoids, two for each piston. For example:

```
Solenoid blue1 = new Solenoid(0);
```

You should also instantiate your Compressor and Joystick. The Compressor does not need anything in the constructor, so it's pretty straightforward.

Controlling Solenoids

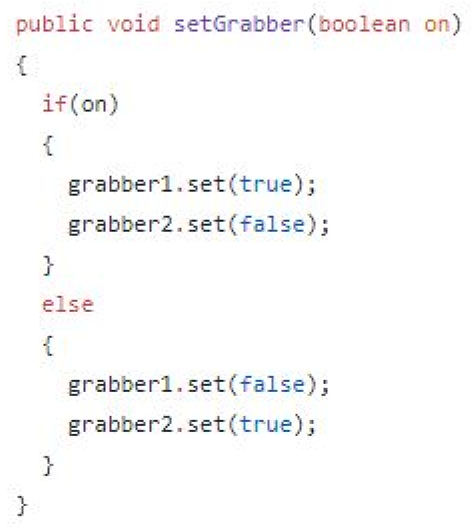
Considering pneumatics operate under simple boolean values, it should come as no surprise that controlling them is quite easy. The method `set(boolean)` is used to actuate the solenoids. Remember, as each piston utilizes two separate solenoids each must be set to operate it. Also, both connected solenoids must be set to opposite values(so if the first is set to true the second must be false).

Controlling The Compressor

The compressor is also simple to use. They do not appear often in the code, and when used, their methods are simple and self-explanatory. To turn on the compressor, use the `start()` method and to turn it off, use the `stop()` method.

Tips

Because you have two Solenoids per piston, it can be tedious to change them both. So, you can create a method that does it for you.



```
public void setGrabber(boolean on)
{
  if(on)
  {
    grabber1.set(true);
    grabber2.set(false);
  }
  else
  {
    grabber1.set(false);
    grabber2.set(true);
  }
}
```

The purpose of this method is to activate or deactivate the pistons in one line of code, instead of two. This is achieved by setting the Solenoids of a piston opposite to each other.