# Pololu 3pi Firmware Overview

*Rick Coogle*

## Introduction

The firmware developed for the Pololu 3pi robots exposes a command set similar to that of the K-Team Khepera robots. The intent is for environments that already have drivers developed for the Khepera (such as the Player/Stage project) to work with Pololu robots loaded with this firmware. This document describes how to program the robots with the firmware, and documents the complete command set, in addition to other important pieces of information.

The source code archive has the following directory structure:

pololu/
Root directory

> firmware/
> Primary firmware directory
>
> pololu_driver/
> Graphical user interface for controlling a robot.
>
> writeseed/
> Used to write the unique ID, random seed, and processor fuses to a robot.

## Firmware and Robot ID

This section describes how to build and program the robot with the firmware, as well as how to set unique identifiers on each robot.

### Firmware Build and Programming

Building the firmware is relatively straightforward. First off, the avr-gcc packages including avrdude for device programming must be installed on your system. Follow the directions for the appropriate Linux distribution to install the cross-compiler and other tools (this will also build in Cygwin under Windows).

Once that is taken care of, building the firmware is as simple as issuing the following command in the firmware source directory:

```
make
```

To program the device, connect the Pololu programmer between the computer and the robot you wish to program, turn on the robot by pressing the Power button, and issue the make command with the "program" target:

```
make program
```

If there are errors during programming, this is likely the result of a connection issue. These can usually be resolved by resetting the connections and the robot and trying again.

When the robot restarts, it will be listening on a serial device for commands; the serial speed is 9600 bps.

### Setting the Robot ID

If setting the unique identifier for the robot is necessary, a different firmware program must be loaded first and used to set the ID. Additionally, the AVR processor fuses must be set appropriately to ensure that the EEPROM storing the ID is not cleared each time the firmware is reprogrammed.

First, set the fuses. Make sure that the robot is connected to the computer via the programmer as discussed in the previous section. Change to the directory where the configuration program is stored (writeseed), and issue the command

```
make writefuses
```

to set the fuses. Next, build the driver program with `make`, and program the robot with it using `make program`. The robot's LCD should show a prompt: "oid=" is the old robot ID, and "id=" is the new ID. Use the A/B buttons to cycle through the numbers, and press C to set the new ID.

## Command Set

The following commands may be issued to the robot over the serial interface. Note that each command must be terminated by a carriage-return/line-feed combination to be recognized, and that each command may be passed an optional *id* parameter. If the robot identifier does not match the *id* parameter, then the command is ignored. Once a command completes, the robot sends a response back to the sender.

- `D,<left wheel speed>,<right wheel speed>[,id]` – Set the wheel speeds, in the range 0-255.
  - Response: `d,<id>`
- `E[,id]` – Read the wheel speeds.
  - Response: `e,<left wheel speed>,<right wheel speed>,<id>`
- `G,<left count>,<right count>[,id]` – Set the odometry counts for each wheel.
  - Response: `g,<id>`
- `H[,id]` – Read odometry counts.
  - Response: `h,<left count>,<right count>,<id>`
- `N[,id]` – Read IR light intensity sensor status.
  - Response: `n,<sensor 1>,<sensor 2>,<sensor 3>,<sensor 4>,<sensor 5>,<id>`
- `S,<fov>[,id]` – Set virtual sensor field-of-view radius.
  - Response: `s,<id>`

- `P,<target id>,<target x>,<target y>,<robot truth x>,<robot truth y>[,id]` – A virtual target position sent to the robot. It also includes the ground truth position of the robot for relative positioning.
    - Response: `p,<id>`
- `Q[,id]` – Return last target measurement (fuzzed target position).
    - Response: `q,<meas. validity>,<target id>,<meas. x>,<meas. y>,<id>`
- `V[,id]` – Return current battery voltage.
    - Response: `v,<voltage>,<id>`

## Control Interface

The pololu_driver directory contains the source for a Qt application for remotely controlling a Pololu 3pi that is programmed with the aforementioned firmware. To build it, Qt4 and the Qt Creator IDE are required. Operating the interface is straightforward; the current version uses a straight serial interface to the robot. It is intended for Linux systems.

Previously, Xbee radios were used for communication between the computer and the robots; when setting up the control interface, use the appropriate tty device. For example, /dev/ttyUSB1.

## Player Configurations

There are two config files that may be used with the Player server:

- pololu.cfg - Example configuration for a single robot.
- pololu-js.cfg – Example configuration that allows for control of a robot using a joystick.