Customized Plastic Extruder

For mounting on franka emika panda and use with ROS2-Control

Author/Creator:	Andreas Moltumyr
E-mail:	andreas.h.moltumyr@gmail.com
Last modified:	December 1, 2021

1 Intro

The purpose of this document is to give useful information about the extruder in the picture and explain how to use it with ROS2. It is meant to be mounted to the fingers of a franka emika panda robot. Links to hardware, software and CAD resources can be found below. Usage is described below.

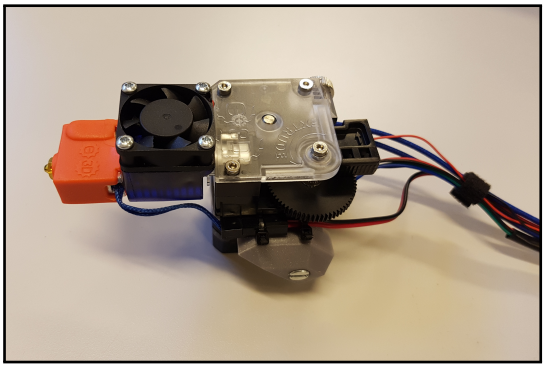

Figure 1: Extruder

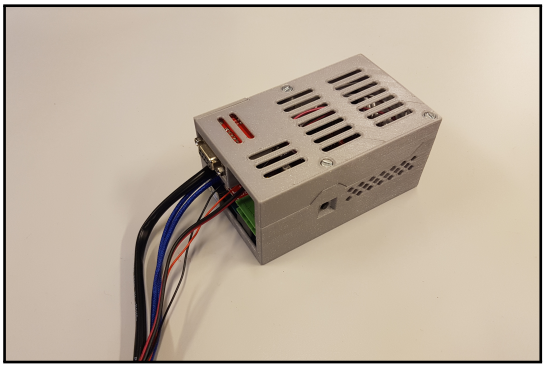

Figure 2: Control box

Table 1: Miscellaneous extruder data

Parameter	Value
Filament diameter	2.85 mm
Nozzle diameter	0.8 mm
Stepper motor steps per revolution	400
Heater cartridge power rating	30 Watts
Input voltage to extruder (ramp/arduino control box)	12 Volts
Current draw at 12 Volts	$0.5 \sim 3$ Amperes
Current draw (only stepper)	~ 0.5 Amperes
Current draw (only heater)	$0 \sim 2.5$ Amperes

2 Resources

Table 2: Part list

Parts	Notes
Titan extruder	2.85mm filament diameter, direct drive, with 'compact but powerful motor' (datasheet).
Volcano HotEnd	2.85mm filament diameter, direct drive, 12V. with 0.8mm brass nozzle.
Arduino Mega 2560 Rev3	Docs/Resources, arduino C++ reference
Ramps 1.4	Docs. Supports control of five stepper motors, fans, nozzle heating and bed heating. However, the extruder are only using one motor, nozzle heating and a fan.
DRV8825, Stepper motor driver	Datasheet.
RS-232 til TTL nivåomformer	Docs.
PCIe RS232 computer card	
RS232 cable	
PEI sheet	To print on top of. Should improve adhesion of first layer.
PLA filament	2.85mm diameter.

Table 3: Software resources

Project name	Links	Notes
`ROS2 foxy`	Docs. github	
`ROS2 Control (foxy)`	Docs. github	
`(Franka ROS)`	Docs. github	
`extruderCtrl`	github	Arduino code for control box. Maintained by author.
`am_extruder_tool`	github	ROS2 control hardware resource, descriptions and launch script. Maintained by author.
`am_extruder_msg`	github	Custom message types. Maintained by author.
`am_extruder_simple_ui`	github	RQT-gui plugin to set and read nozzle temperature and filament speed. Maintained by author.
`Teuniz/RS-232`	Docs. gitlab	By Teunis van Beelen.

CAD-models

2.1 Connections

1. 12V power supply connector supplying ramps board with and arduino with power. Brown wire is +12V, white is ground. An internal step-down converter takes +12V and supplies +5V to the arduino board.
2. Connector port for serial/RS-232 communication. The extruder controller box uses this serial port to receive commands from the computer that is running ROS and to send sensor data to the computer.
3. USB interface to the arduino. Used for programming the microcontroller. It can also supply the arduino board with +5V, which is enough for it to run and communicate over serial, but it will not power the ramps card (or any of the other devices needing 12V, like the motor, heater or fan). Powering the box/arduino from both the USB (3.) and 12V power supply connector (1.) at the same time is/should be no problem.
4. Connector for a 12V heater cartridge.
5. Connector for a 12V heatsink fan. Polarity matter.

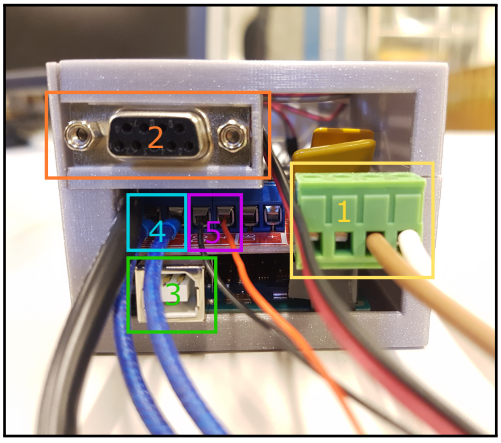

Figure 3: Side of control box

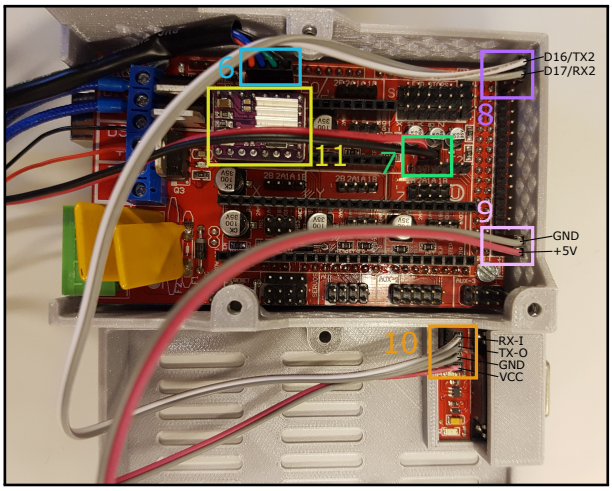

Figure 4: Top of control box

6. Connector for the extruder stepper motor. If connected the wrong way, the motor should move in the opposite and wrong direction.
7. Connector for thermistor.
8. Serial (TTL) TX and RX pins. Top pin in figure is D16/TX2 pin on arduino mega, while the second top pin is D17/RX2 pin on arduino.
9. Power to the RS232 level shifter. Bottom pin in figure is +5V and second bottom pin is ground.
10. Connections on RS232 level shifter board. From bottom in picture, VCC, GND, TX-O and RX-I. Connect VCC to +5V pin (9.), connect GND to Ground pin (9.), connect TX-O to D17/RX2 pin (8.), and connect RX-I to D16/TX2 pin (8.).
11. DRV8825, stepper motor driver mounted on top of the ramps 1.4 board.

3 Instructions on use

3.1 Hook up the extruder

Connect a serial cable from the computer to the serial port (2.) on the extruder control box. Connect a 12V power supply (1.) to the control box. Make sure the power supply can supply 3 Amperes or more to ensure the extruder will operate correctly. A current rating between 0.6 - 3 Amperes can work, but the force produced by the stepper motor and/or the thermal energy produced by the heater cartridge can be lower than what is necessary.

3.2 Setup

If you do not already have `ROS2 foxy`, `ros2_control` and `ros2_controllers` installed, install them.

Next, create a ros2 workspace, if you have not already done so.

Navigate to your ros2 workspace and clone the necessary repositories:

```
cd [your_ros2_workspace]/src
git clone https://github.com/mltmyr/am_extruder_tool
git clone https://github.com/mltmyr/am_extruder_simple_ui
git clone https://github.com/mltmyr/am_extruder_msg
```

The `am_extruder_tool` package has a git submodule, the RS-232 driver by Teunis van Beelen. We need to fetch that:

```
cd am_extruder_tool
git submodule init
git submodule update
cd ..
```

3.3 Build

```
source /opt/ros/foxy/setup.bash
cd [your_ros2_workspace]
colcon build
source install/setup.bash
```

If building is successful, move on to launching the example in Sec. 3.4. If it fails, you will have to resolve the issue(s) based on the feedback from colcon.

3.4 Use the ROS2 modules to move and heat filament

Make sure the control box is connected to power and that it is connected to the computer via RS232 cable. Also, ensure that the arduino project `extruderCtrl` is loaded and running on the arduino. The stepper motor should make a low buzzing sound and you should not be able to turn the gears because of the motor torque.

To launch the extruder tool demo, run the following command in the terminal:

```
ros2 launch am_extruder_tool_example.launch.py
```

Open another terminal and echo the `joint_states`. Remember to load/source the ros2 workspace:

```
cd [your_ros2_workspace]
source install/setup.bash
ros2 topic echo /joint_states
```

If everything is set up correctly, you should now be able to read the nozzle temperature and filament speed from the data structure being written in the terminal.

To set a reference or target value for the nozzle temperature to 40°C, enter the following into a new terminal (remember to source the workspace):

```
ros2 topic pub /filament_heater_controller/commands std_msgs/msg/Float64MultiArray
    'data: {40.0}'
```

Heating is relatively slow, but after some time, a rise in temperature should be visible on the `/joint_states` topic. Set target value to 0.0 to turn off the heating.

To set a reference or target value for the filament speed to 5mm/s, enter the following into a new terminal:

```
ros2 topic pub /filament_mover_controller/commands std_msgs/Float64MultiArray '
    data: {5.0}'
```

The stepper motor and gears should start to rotate at a speed that will make inserted filament move at 5mm/s. To stop the motor from rotating, set target value to 0.0. To make the motor rotate the other way, use negative values.

3.5 Simple RQT plugin

Setting target values with the help of the `ros2 topic pub` command is cumbersome and a bit slow. To make it easier to manually control heating and speed of filament a simple RQT plugin was written.

The 'Simple extruder gui' RQT plugin can be run from terminal in three different ways:

1. `rqt`. Under plugins, in the top menu, choose `am_extruder -> Simple extruder GUI`.
2. `rqt -standalone am_extruder_simple_ui`
3. `ros2 run am_extruder_simple_ui am_extruder_simple_ui`

Running one of the options, a gui should open. In gui, the current nozzle temperature and filament speed is shown. By entering values in the two fields, the temperature target and speed of the motor can be controlled.

3.6 Extruder urdf definition

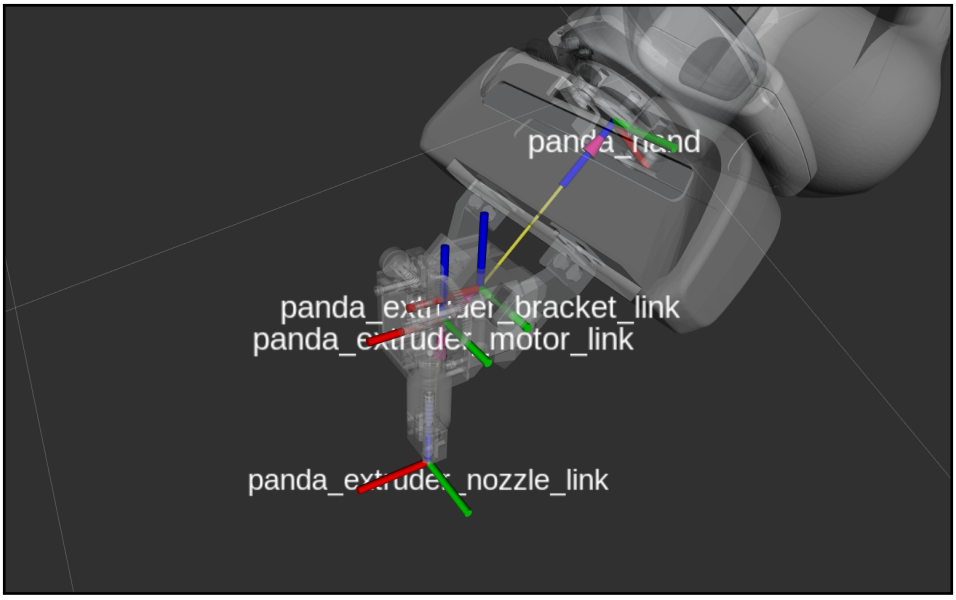

Figure 5: Extruder model and urdf coordinate systems shown with gazebo.

ROS2_control and urdf example for integrating into a bigger system,

4 Configure

4.1 Configure microstepping level

Below the stepper motor driver, when it is mounted to the ramps 1.4 board, there are three pairs of pins used for configuring the microstepping level with jumper wires. In the current setup, a microstepping level of 16 microsteps/step is to be used. For the DRV8825, this is configured by adding a jumper wire over the two right most pins, given the orientation of the ramps board in fig. 4, while leaving the other two pairs of pins unconnected.

If wrong microstepping is used, the motor will either move faster or slower than intended. It will move half as fast with microstepping 32, twice as fast with microstepping 8, four times as fast with microstepping 4, eight times as fast with microstepping 2 and sixteen times as fast with no microstepping.

4.2 Filament speed and calibration

The speed at which the filament is moved by the stepper motor when sending a stepping-frequency message to the ramps/arduino board over serial is influenced by several parameters. If, for instance, the filament is to be moved at 1 mm/s, one would have to use these parameters to calculate the stepping-frequency value to be sent over serial. This calculation is done in the ROS2 hardware interface provided in the am_extruder_tool package and this calculation needs to be correct in order for the extruder to move the filament at the correct and desired rate.

The low level control of the stepper motor, i.e. voltage and current control of individual windings/phases in the motor, is performed by the DRV8825 stepper motor driver mounted on top of the ramps board. The interface between the stepper motor driver and the arduino board consists of three IO-pins, they are: 'enable', 'step' and 'direction'. The 'step'-pin should be pulsed once, each time the motor should move one step in the direction indicated by the 'direction'-pin.

To ensure that the 'step'-pin is pulsed regularly, and with the correct frequency, a hardware timer is used. A value is entered into the OCRxA register to control the period of the timer. When the timer expires, an interrupt routine is run that toggles the 'step'-pin.

5 Software details

5.1 Messages over serial

A simple and lightweight custom message format was created for the communication between the extruder control box and computer running ros.

Each message start with an operation code of one byte and a variable length data field (payload) where the length and format of the payload is implicitly given by the operation code.

There are no acknowledgements and no retransmissions of messages. Therefore, the serial communication is far from robust. But it is sufficient for the current use.

In order for the messages to be as short as possible, nozzle temperature data and stepper motor stepping frequency data is sent over serial as float32 (4 bytes) and not as a string of characters. This makes the data sent over serial unreadable with serial consoles (e.g. PuTTY) that interpret every byte as an ASCII character.

The messages that can be sent between the computer and the extruder control box are explained in Table 4 and Table 5.

Table 4: Messages from computer to extruder control box.

Opcode	Payload	Purpose
'H'	float32	Set nozzle temperature reference in °C for PID-controller. 5 bytes per message.
'X'	float32	Set stepper motor stepping frequency (Hz). 5 bytes per message.
'B'		Blink the debug LED on the arduino/ramps board once. For debugging. 1 byte per message.
'Y'		Purpose was to start periodic messaging of nozzle temperature and filament speed. This functionality might be disabled in the .ino file. 1 byte per message.
'N'		Same as above, but used to turn of periodic messaging. 1 byte per message.
'T'		Request the controller box to send nozzle temperature in °C once. Not necessary if nozzle temperature is periodically sent. 1 byte per message.
'E'		Request the controller box to send stepper motor stepping frequency (Hz) once. Not necessary if filament feed rate is periodically sent. 1 byte per message.

Table 5: Messages from extruder control box to computer.

Opcode	payload	Purpose
'T'	float32	Message containing nozzle temperature in °C. 5 bytes per message.
'E'	float32	Message containing stepper motor stepping frequency (Hz). 5 bytes per message.