UserSpace IO via uLisp and Arduino Uno

Steak Electronics

1 UserSpace IO via uLisp and Arduino Uno

Overview: Goal is to get an easy way to echo "command" > /dev/ARDUINO from GnuLinux userspace.

Then I can 3D print a box, throw some LEDs in there, and get a hardware interface for anything from my distro.

1.1 Setup

First off, I installed the stable uLisp from ulisp.com (AVR version 3.4 in 2020/12). [1]

1.1.1 Trouble with interfacing directly to ttyACM0

The Arduino Uno uses an FTDI USB to Serial chip. It also has more than just RX/TX connected (I think). This means that when you echo "something" to arduino, this command will open the serial connection, send the command, then close the command and possibly reset the board via one of the UART reset pins (DTS perhaps). This won't work. First thing to do, get an FTDI board, and connect it to the computer, the connect it to digital pins 0,1 (for UART). Then

[1] It's a sketch. It installs from Arduino IDE (must be reasonably recent, so 1.8 from Arduino.com works, but not Beowulf/Buster's 1.0 Arduino).

test sending a command. This works. This will be the final setup. Computer ->FTDI ->Arduino.

It should be possible to get it to work without the FTDI chip, but not without some debugging. Have at it, if you wish. I don't have time to troubleshoot this now. I tried a few things (exec 3<>, and the stty) neither worked. You could also use Python, but then that's more dependencies. Bad idea. This should be simple.

Ref:

https://forum.arduino.cc/index.php?topic=61127.0

https://arduino.stackexchange.com/questions/16776/how-to-connect-arduino-device-to-linux

https://stackoverflow.com/questions/3918032/bash-serial-i-o-and-arduino

1.1.2 Troubleshooting Code

The first thing you will want to do is prove that you can echo or cat "code" >/dev/ARDUINO. This code works:

```
(defun b (&optional x)
  (pinmode 13 :output)
  (digitalwrite 13 x)
  (delay 1000)
  (b (not x)))
(b)
```

Save it as a text file, then cat the file into /dev/ttyUSB0 or /dev/ARDUINO. Problem is, the code runs indefinitely. You can't tell it to do anything else. Hit the reset button to restart the Uno back to square one. So instead, you want a function that runs and then stops. Here's one of those:

```
 (defun b ()
  (pinmode 13 :output)
  (digitalwrite 13 nil)
  (delay 500)
  (digitalwrite 13 t)
```

```
  (delay 200)
  (digitalwrite 13 nil)
  (delay 500)
  (digitalwrite 13 t)
  (delay 200)
  (digitalwrite 13 nil)
  (delay 500)
  (digitalwrite 13 t)
  (delay 200)
  (digitalwrite 13 nil)
  (delay 500)
  (digitalwrite 13 t)
  (delay 200))
(b)
```

This works. Save it as a .dat or .txt (whatever) and cat file.txt >/dev/ARDUINO. It will blink a few times, then stop. You can cat it again, if you wish. There, now we have an interface, and we have proven we can switch an IO high/low. With a couple of functions, it's trivial to build an interface.[2]

Tip: When building code, copy and paste into the Arduino Serial Monitor first. This way, if there's an error, it will tell you what's wrong. After you've tested the code, then you can put it in a text file. As we are keeping this simple, we don't have logging setup, or anything else. This is meant to be for basic IO from user space, not C compiled programs. Think lightweight. Fast. Fun.

1.2 An interface for basic GPIO

Now, let's build an API/interface so that we can flip some IO, and perhaps increment as mentioned.

```
let's ask this: what do I want to accomplish?
```

[2] i.e. have one function set IO high for a given pin. Have one function set IO low. Have one function blink IO some amount of times. This can be used for counters or shift registers, or in the former, just controlling GPIO.

answer: I want to be able to flash leds from user space.
possibly increment a counter, but
that's later. for now, just flash leds.

solution: build box / board for uno. connect everything up.
build initialization program
(that will enable outputs as high)
Then find a way to send commands from gnulinux. let's begin.

1. build initialization program
2. create enclosure, and wire up leds
3. run program live from gnuLinux.