DC Motor Guide

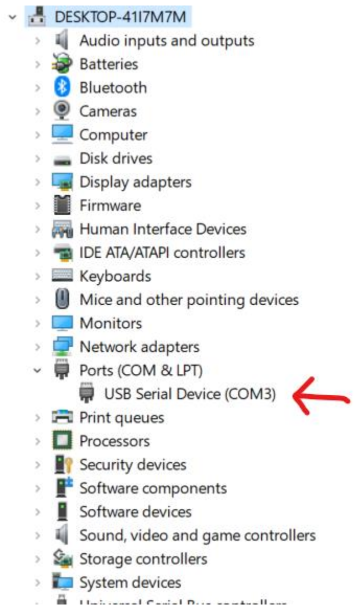
Setup

Plug in the Arduino to the PC with a USB cable. Plug the power adaptor into the 12V barrel (not the one on the Arduino board).

Finding COM Port

- On Windows

With the Arduino connected to the computer, open Device Manager (from search bar or Windows settings), and look for the USB Serial Device under the Ports tab. This will show you the COM port that the Arduino is currently assigned.



DESKTOP-41I7M7M
- Audio inputs and outputs
- Batteries
- Bluetooth
- Cameras
- Computer
- Disk drives
- Display adapters
- Firmware
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
  - USB Serial Device (COM3)
- Print queues
- Processors
- Security devices
- Software components
- Software devices
- Sound, video and game controllers
- Storage controllers
- System devices

- On Linux

Open Matlab and run the ‘seriallist’ command. This will return a list of available serial ports. The Arduino is usually assigned to /dev/ttyACM0 on Linux.

Using The run Function

Calling the run function will run the DC Motor system. It takes up to 4 parameters:

- u – The input vector, containing an input value between -1 and 1 (-100% and 100% of max torque) for each sample.
- “port” – (Optional) String containing the port the motor is connected on. Set by default to the last active port.
- “Ts” – (Optional) The sampling time (in seconds) to run the system with. Set by default to 5ms.
- “type” – (Optional) DC Motor interface type. Can be one of the following
  - “auto” – Will automatically choose the optimal interface type. Set by default.
  - “native” – Uses the native Matlab serial interface implementation

      - “windows” – Only works on windows. Is generally faster than the “native” option, but only compatible with port numbers up to “COM7”

The function returns 3 vectors:

- vel – A vector containing the velocity of the motor in RPM.
- alpha – A vector containing the angle of the motor shaft in degrees.
- t – A time vector

These 3 vectors have the same length as u.

Here’s an example of how the function is used:

```
u = [zeros(10, 1); 0.5*ones(1000, 1)];

[vel, alpha, t] = DCMRun.run(u, ‘port’, ‘COM4’, ‘Ts’, 1e-3, ‘type’, ‘windows’);

plot(t, vel);
```

Occasionally, the system will not run when calling the run function. If this occurs, a second attempt at running the system will usually function correctly. If the second attempt fails, ask a teaching assistant for help.

Online Mode

Some labs may require running the motor in online mode. This can be done using the following steps:

1. Initialize the motor by calling DCMRun.start() and assigning the result to a variable (e.g. ‘motor’).
2. Apply inputs to the motor by calling motor.step(u).
3. Synchronize the sampling period by calling motor.wait().
4. End the online session by calling motor.stop()

An example of this can be seen below:

```
u = ones(1, 200);

y = zeros(1, length(u));
motor = DCMRun.start();

for k = 1:length(u)
      y(k) = motor.step(u(k));
      motor.wait();
end

motor.stop();
```

Considerations for u and Ts

The run function clears the USB read buffer when it first runs, so in order to let it do so, it’s best to prepend your u vector with about 10 zeros.

I added a few helper functions for generating input vectors:

ustep(n) – generates a step signal with n – 10 samples. The first 10 samples are 0.

urand(n) – generates a random staircase input vector. Contains steps of random values with a width of 100 samples each.

Ex:

```
u1 = 0.5*ustep(1000);

u2 = urand(1000);
```

The output vectors (vel and alpha) are only as long as u, so if you want to run an impulse, you need to append enough zeros to the end of u to see the entire impulse response.

The minimum Ts (for the windows version) is 2.5ms. Any lower than that and the true sampling time won't be stable. Here's an example of setting a 10ms sampling time:

```
[vel, alpha, t] = DCMRun.run(u, “Ts”, 10e-3);
```