# FDC-320 with Arduino
## Manual

How to operate the FDC-320 with Arduino controller

Reykjavík University

Torfi Þorgrímsson[*]

## QUICK START

### DOWNLOAD GIT

Run `winget install Git.Git` in cmd or powershell on Windows

Run `sudo apt install git` on Linux

### DOWNLOAD THE GITHUB REPOSITORY

```
git clone https://github.com/AwesomeQuest/fdc-320.git
```

The repo also has a copy of this manual in case you need another copy. Feel free to open an issue or send me an email if you have any problems.

### INSTALL JULIA

Second you must install the Julia language

### WINDOWS

Run the following command in powershell or cmd

```
winget install JuliaLang.Julia
```

### LINUX

Run the following command in your favorite shell

```
curl -fsSL https://install.julialang.org | sh
```

---

[*]torfit@gmail.com

## SETUP ENVIRONMENT

Then open a Julia REPL in the `FDC-320` folder

```
julia --project=.
```

And you should see something like this

```
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _    |  Type "?" for help, "]?" for Pkg help.
  | | | | | | | | |/ _` |  |
  | | |_| | | | | (_| |  |  Version 1.12.0-rc1 (2025-07-12)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org release
|__/                    |

julia>
```

Then include the file `FDC320lib.jl`

```
julia> include("FDC320lib.jl")
```

This file automatically puts the relevant libraries in your namespace.

## QUERYING THE FDC-320

### LIST AVAILABLE PORTS

Make sure the Arduino is plugged into your pc via USB. You check this with the following command.

```
julia> list_ports()
COM3
        Description:   Arduino Uno (COM3)
        Transport type: SP_TRANSPORT_USB
```

This command lists all the valid USB port names and their description. The example above is for a windows machine, on linux the name has a different format but is functionally equivalent.

### OPEN THE PORT

In order to use a USB port for serial communication you must "open" it first. This takes ownership of the port and makes sure no other processes can use it.

```julia
julia> port = LibSerialPort.open("COM3", 9600)
```

Here the first argument is the name of the port as shown by `list_ports()` and the second argument is the baud-rate. It is critical that the baud-rate be set the same as the Arduino, I have set it to 9600. The baud-rate of the FDC-320 is an internal setting and must be changed with the `setCommunicationBaudrate` function, then you must change the Arduino code to match.

**MAKE A REQUEST**

Now you can use the public API (Julia) to make requests to the FDC-320. Here is an example:

```julia
julia> get_ReadFlowrateActualFlowrate(port)
0.0f0
```

You can reference the FDC-300 manual pages 16-17 for the names of the available functions, each function either reads or writes to a register. In that table each register is marked either "Read only"(R), "Write only"(W), or "Read and write"(RW).

Each register gets one or two functions in the file `FDC320lib.jl` corresponding to if the register is R, W, or RW. If the register is marked R it gets a function of the same name prefixed with "get", if it is marked with W the function prefix is "set". If the register is RW then it gets both a "get" and a "set" function.

When calling a "set" function it is important to call it with the right type. This type is annotated in the table in the FDC-300 manual as `U16`, `U32`, or `Float`, these correspond to `UInt16`, `UInt32`, and `Float32` in Julia.

On a success the "set" function will return a `0`

Here are a few examples of dos and don'ts for a set function. (The function returning 0 means success)

```julia
julia> set_SetFlowrateActualFlowrate(port, 0.0f0)
0

julia> set_SetFlowrateActualFlowrate(port, 0.0e0)
ERROR: AssertionError: typeof(val) === regtypes[adr]
Stacktrace:
...

julia> set_SetFlowrateActualFlowrate(port, 0x01)
ERROR: AssertionError: typeof(val) === regtypes[adr]
```

```
Stacktrace:
 ...

julia> set_SetFlowrateActualFlowrate(port, Float32(0x01))
0


julia> set_SetFlowratePercentageMethod(port, 0x0001)
0

julia> set_SetFlowratePercentageMethod(port, 1)
ERROR: AssertionError: typeof(val) == regtypes[adr]
Stacktrace:
 ...

julia> set_SetFlowratePercentageMethod(port, 1.0)
ERROR: AssertionError: typeof(val) == regtypes[adr]
Stacktrace:
 ...

julia> set_SetFlowratePercentageMethod(port, UInt16(1.0))
0
```

As you can see, it is only valid to call the set function with that function's appropriate type as defined in the table.

## LOGGING DATA

In Julia it is most convenient to log data with the `CSV.jl` and `DataFrames.jl` packages. If you do not already have these installed you can do so as follows.

Go into Package mode by entering the ] character.

```
julia> ]

(fdc-320) pkg> add CSV, DataFrames
```

Then you include them in your namespace with a `using` statement.

```
julia> using CSV, DataFrames
```

The following is a simple example of logging and saving data

```julia
julia> using Dates

julia> times = []; datas = [];

julia> for _ in 1:100
           push!(datas, get_ReadFlowrateActualFlowrate(port))
           push!(times, now())
       end

julia> CSV.write("test.csv", DataFrame(timescolumn=times,
datascolumn=datas))
"test.csv"

julia> CSV.read("test.csv", DataFrame)
100×2 DataFrame
 Row │ timescolumn              datascolumn
     │ DateTime                 Float64
─────┼──────────────────────────────────────
   1 │ 2025-08-07T12:16:43.107          0.0
   2 │ 2025-08-07T12:16:43.328          0.0
  ⋮  │           ⋮                       ⋮
  99 │ 2025-08-07T12:17:05.101          0.0
 100 │ 2025-08-07T12:17:05.316          0.0
                              96 rows omitted
```

# READING TEMPERATURE AND HUMIDITY

The file `FDC320lib.jl` also contains a function to read the Temperature and Humidity detected by a DHT22. It is used as follows.

```julia
julia> T,H = TH = readTandH(port)
(temperature = 24.1f0, humidity = 48.100002f0)

julia> T
24.1f0

julia> H
48.100002f0

julia> (T,H) === (TH.temperature, TH.humidity)
true
```

The function returns a `NamedTuple` which can either indexed into like an array or via field like a struct.

# ARDUINO CODE AND WIRING

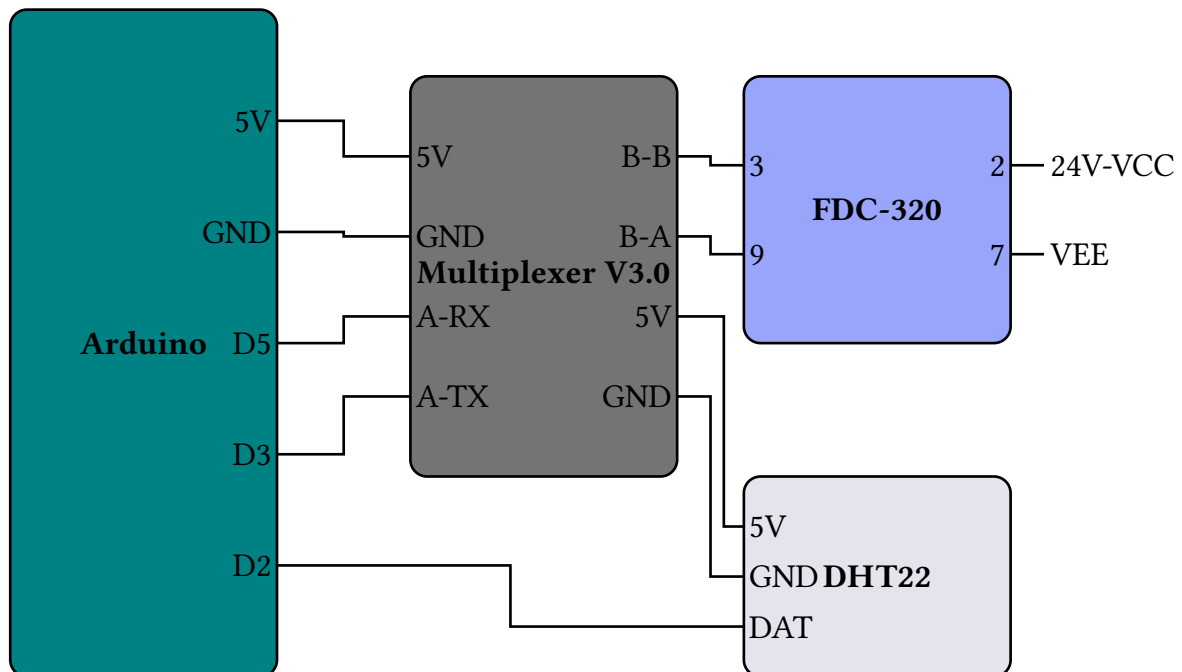The Arduino is running the file `reader/reader.ino` and can be uploaded directly to the Arduino via the ArduinoIDE.



Figure 1: A wiring diagram of the Arduino, FDC, Multiplexer, and DHT22

If you need to replace the multiplexer board it is called "Multi USB RS232 RS485 TTL Converter SKU TEL0070".