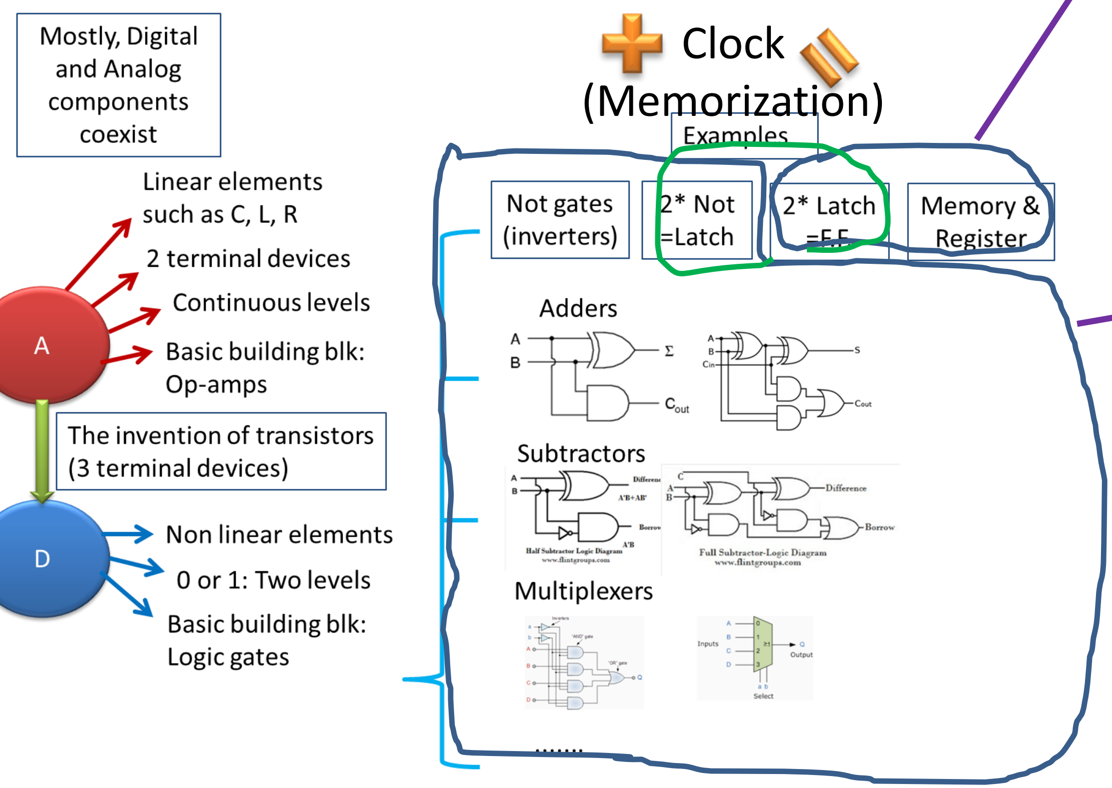
CET 241: Day 10
FSM and Stepper motors

Dr. Noori Kim

Let's apply this concept into MCU programming



Sequential Logics (SL)

- Things are in-order
- Tools to be used to understand SL:
  - FSM (Mealy, Moore)
  - Synchronous/Asynchronous concepts
  - SL Delay analysis

Combinational Logics (CL)

- Tools to be used to understand both CL & SL:
  - Truth table, K-map, SOP, POS, &
  - Analyzing them in transistor levels (i.e., CMOS)
  - CL Delay analysis

Finite state machine abstraction

Remember that it

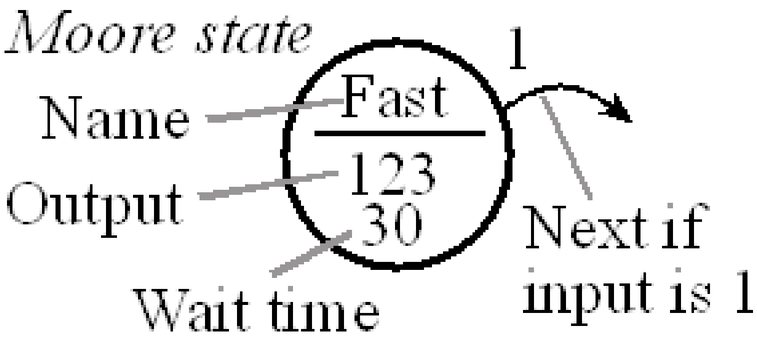
is a “FINITE” state machine

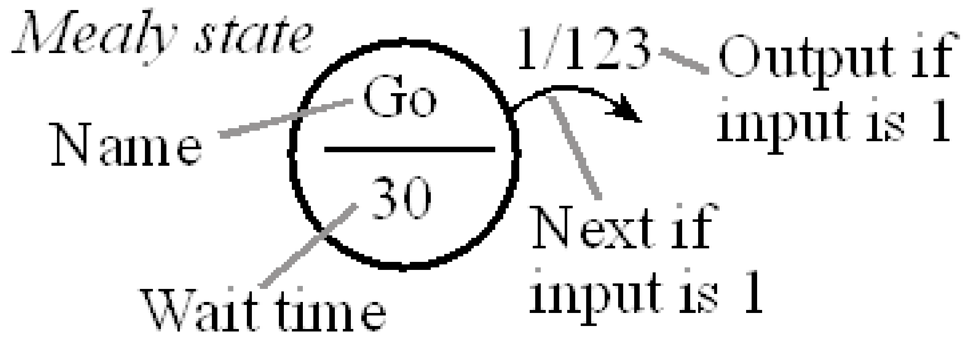
is not an INFINITE state machine

We are in the Digital world,
think about number theories

FSM five essential elements

1. A finite set of inputs
2. A finite set of outputs
3. A finite set of states
4. State transitions
   – Graph (STG)
   – Matrix
5. Output determination: how do you generate outputs? Under what conditions?

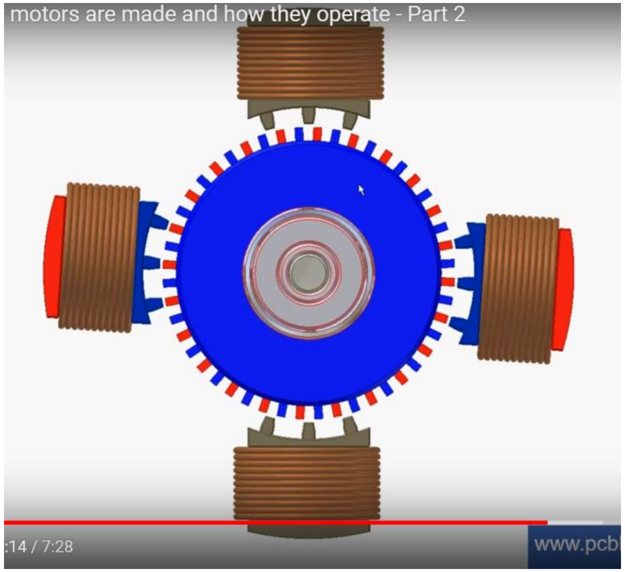


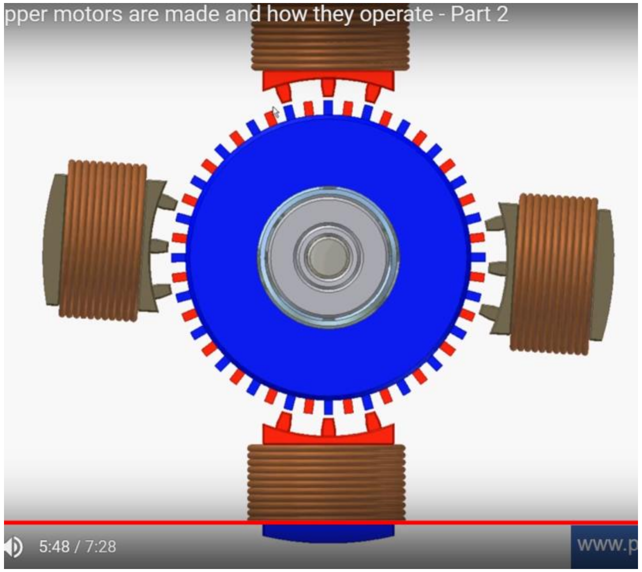


- *NextState = f(Input, CurrentState)*
- *Moore: Output = g(CurrentState)*
- *Mealy: Output = h(Input, CurrentState)*
- Both machine descriptions are equivalent
  - Any system that can be described using a Mealy machine can also be expressed equivalently as a Moore machine and vice versa.

Videos (How Stepper motors are made and how they operate)

- Part 1 (5 minutes): http://www.youtube.com/watch?v=MHdz3c6KLrg
- Part 2 (8 minutes): http://www.youtube.com/watch?v=t-3VnLadIbc





The stepper motor described in this video is not the same kind that we are using. However in terms of operational principle, this video explains more than we need to know.

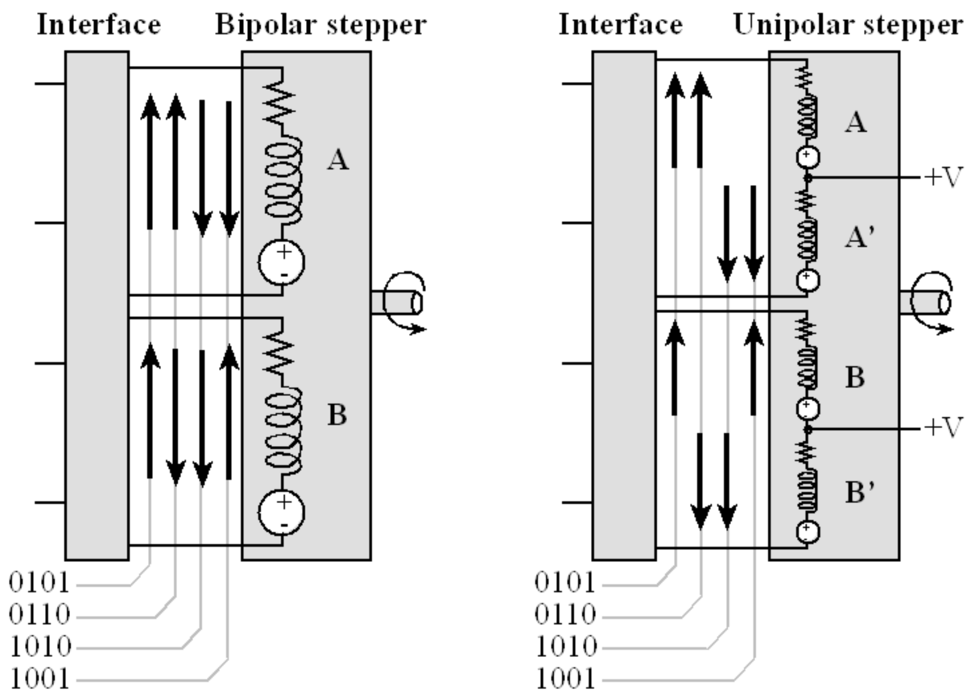
A stepper motor

- A typical motor evaluation: its maximum speed (RPM), its torque (N-m), and the efficiency (electrical power ←→mechanical power)
- Stepper motor evaluation: the rotational position (ϑ=motor shaft angle) rather than to control the rotational speed *($\omega=d\vartheta/dt$)*.

- Stepper motor applications: precise positioning is more important than high RPM, high torque, or high efficiency.
  - Microcontroller-based embedded systems
  - printers to move paper and print heads, tapes/disks to position read/write heads, and high-precision robots.
- The cost: typically higher than an equivalent DC permanent magnetic field motor.
  - But the overall system cost is reduced because stepper motors usually do not require feedback sensors.

Stepper motors

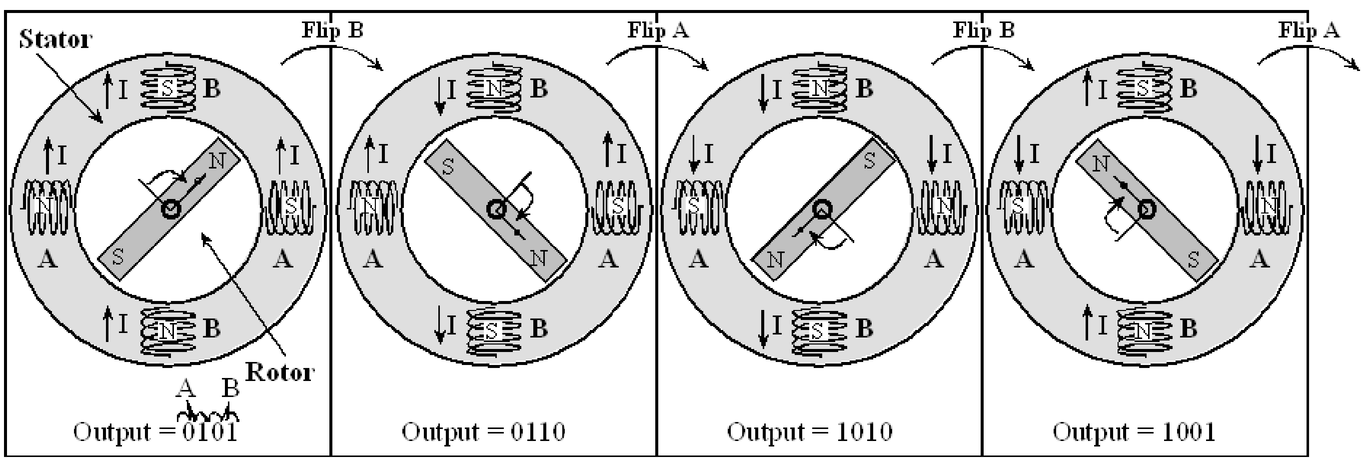
- Two coils, labeled **A** and **B**

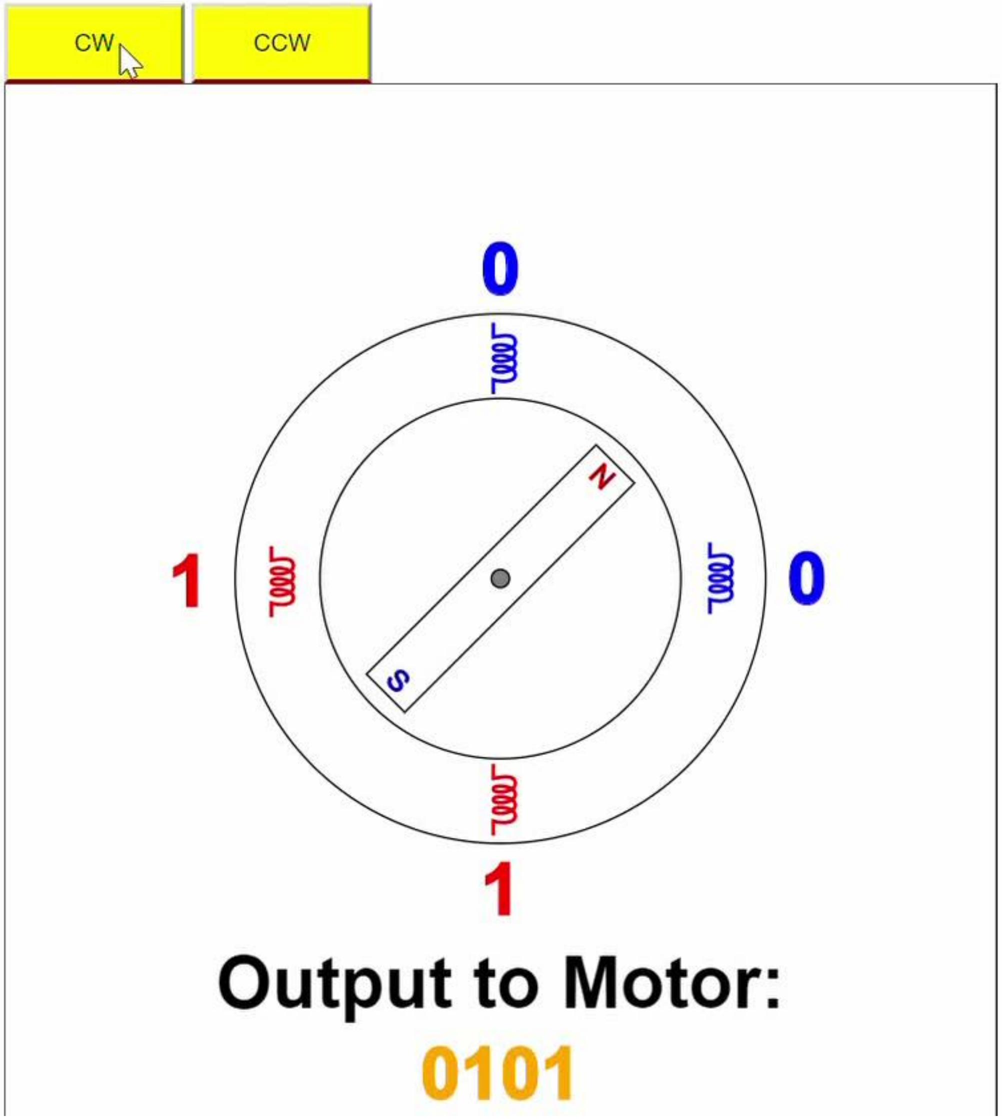


The unipolar stepper motor:

- Providing bi-directional currents, dividing each coil into two parts (center tap).
- The center tap is connected to the +V power
- Half of the electro-magnets are energized at one time →less torque
- Easier to interface.

- Current flows both coils.
  - Current go up: a binary 01 output to the interface
  - Current go down: a binary 10.
- 2 coils → four outputs (e.g., 0101_2 → up/up)
- Output the **sequence** to spin the motor,
  - 0101_2, 0110_2, 1010_2, 1001_2…
- **Reverse the sequence** to rotate the other direction
  - 0101_2, 1001_2, 1010_2, 0110_2…



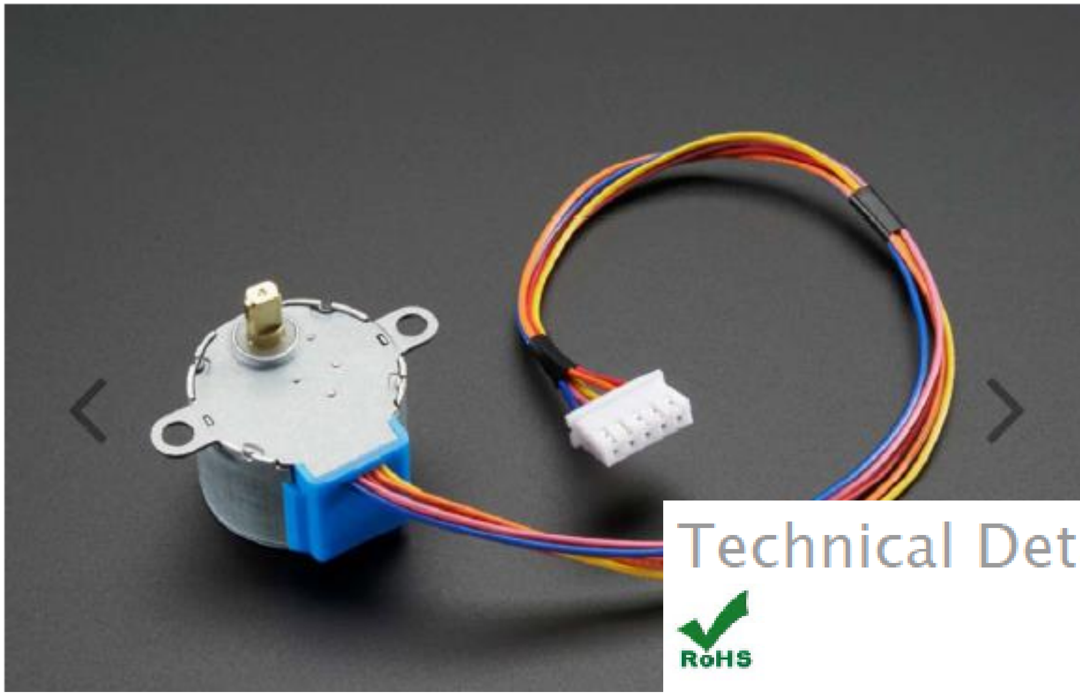


CW: 0101_2, 0110_2, 1010_2, 1001_2

CCW: 0101_2, 1001_2, 1010_2, 0110_2

Small Reduction Stepper Motor - 5VDC 32-Step 1/16 Gearing

PRODUCT ID: 858

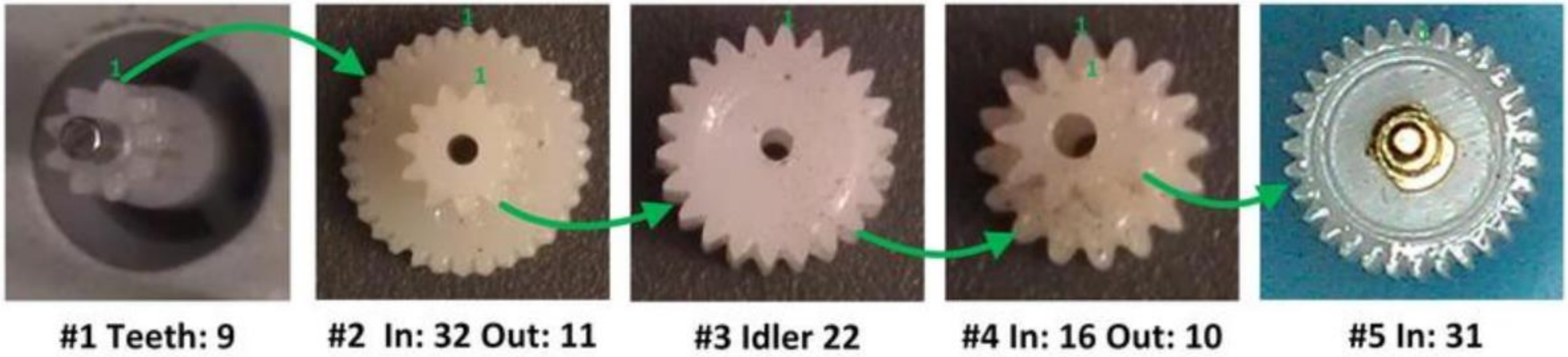


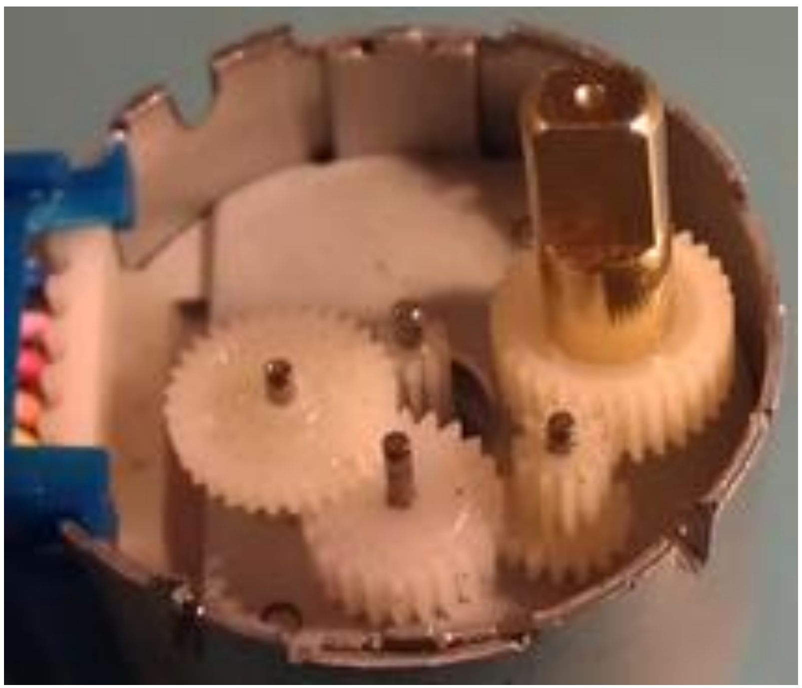
Technical Details

RoHS

- Unipolar stepper with 0.1" spaced 5-pin cable connector
- 513 steps per revolution
- 1/16.032 geared down reduction
- 5V DC suggested operation
- Weight: 37 g.
- Dimensions: 28mm diameter, 20mm tall not including 9mm shaft with 5mm diameter
- 9" / 23 cm long cable
- Holding Torque: 150 gram-force*cm, 15 N*mm/ 2 oz-force*in
- Shaft: 5mm diameter flattened
- Approx 42 ohm DC impedence per winding

Adafruit gear train

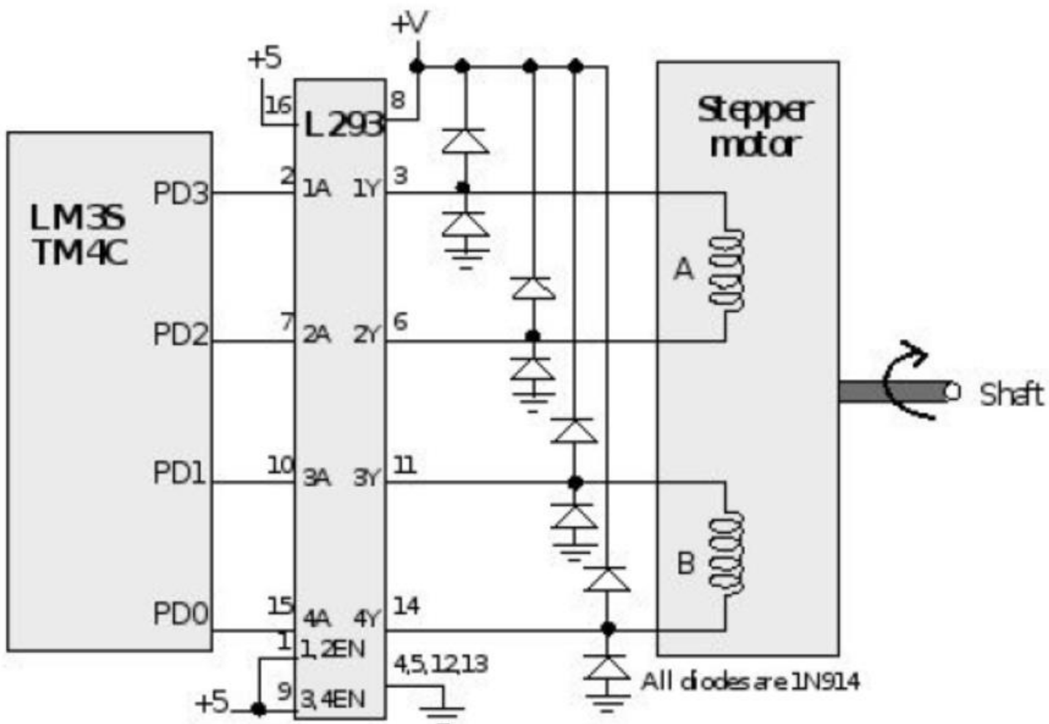


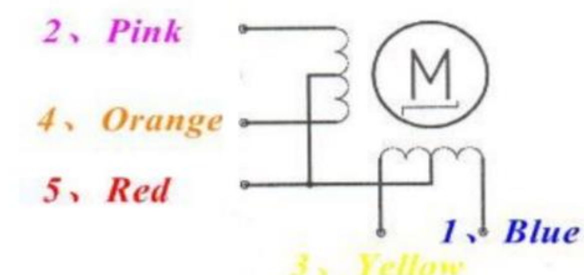


- The gear ratio is: $\frac{32\times16\times31}{9\times11\times10} = 16.032323..$
- **Full-stepping**
  - Internal motor: approx. 32 steps per motor revolution
  - Gear reduction ratio: approximately 1/16
  - So it takes 32×16=512 (513) steps per revolution for the output shaft

Interfacing a stepper motor

- Current supplied by MCU board is not enough to drive a motor
- We need an external power to drive motor, and have to separate two current paths (MCU current path vs Motor current path) ➔ need a motor driver
- Interfacing the stepper motor with a motor driver (i.e., L293 or SN754410) requires four pins: we chose **PD0-3** to control the stepper motor

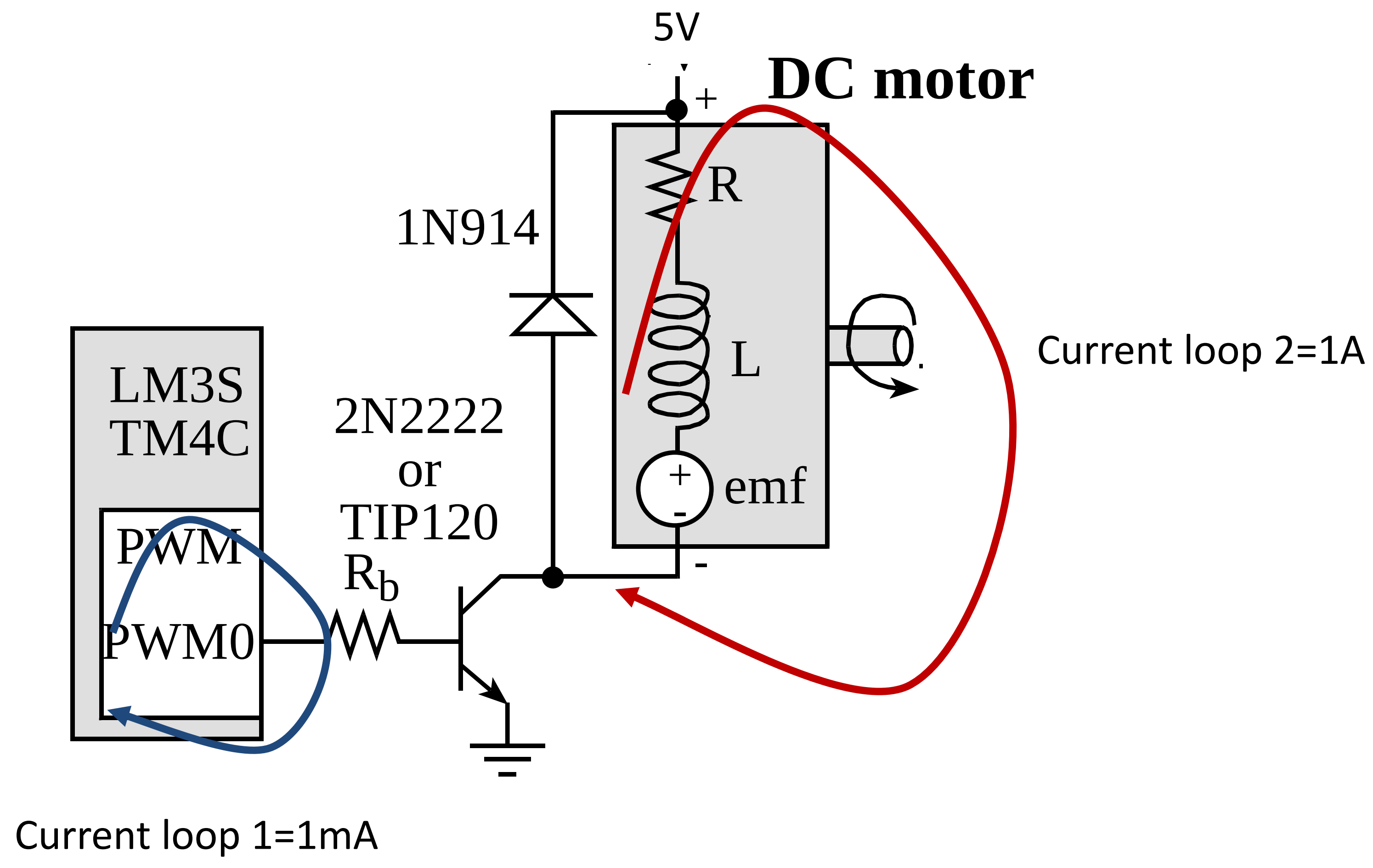




For unipolar

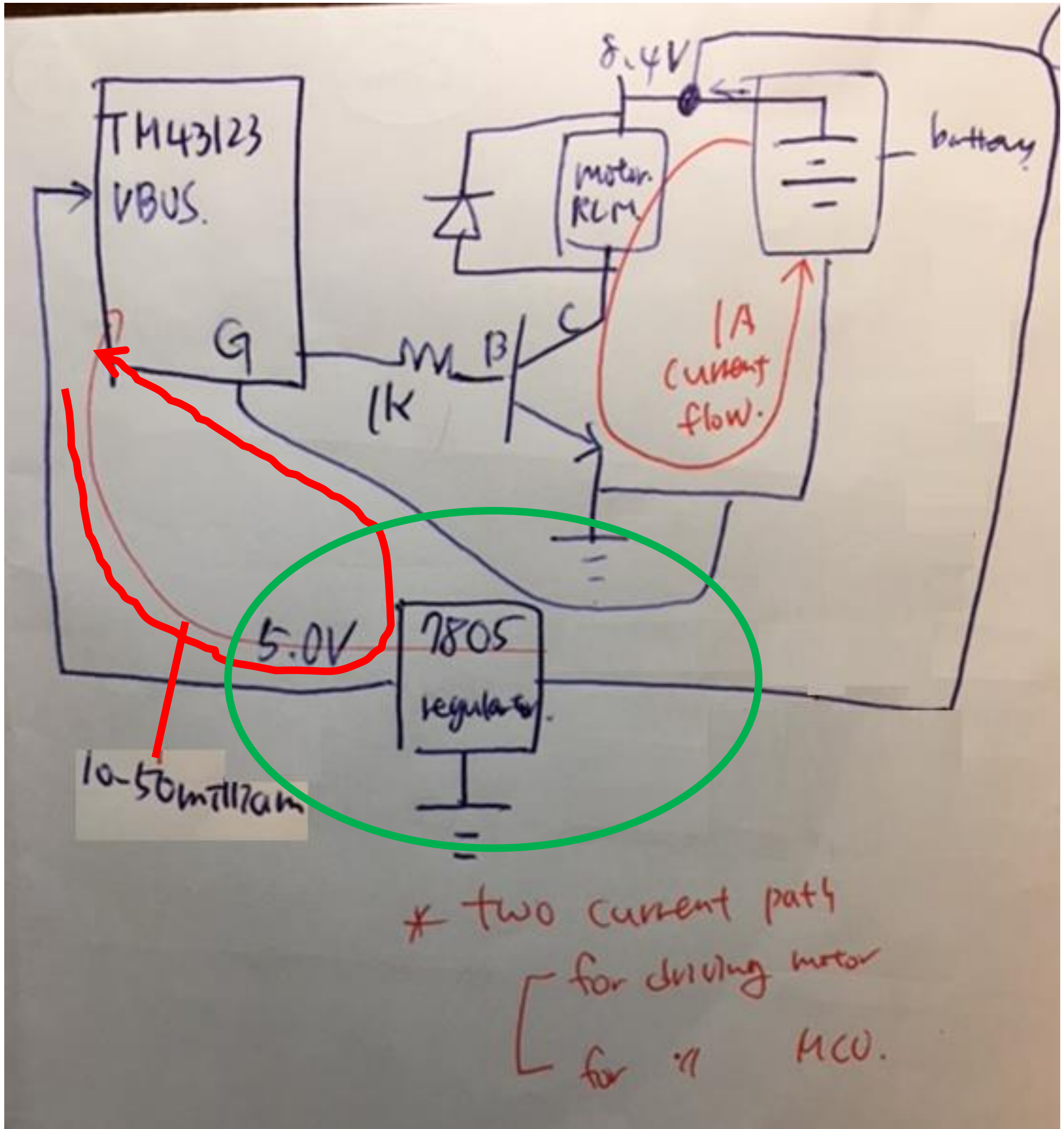
Figure 4.26. Bipolar stepper motor interface using a L293 driver.

Motor interface ex1



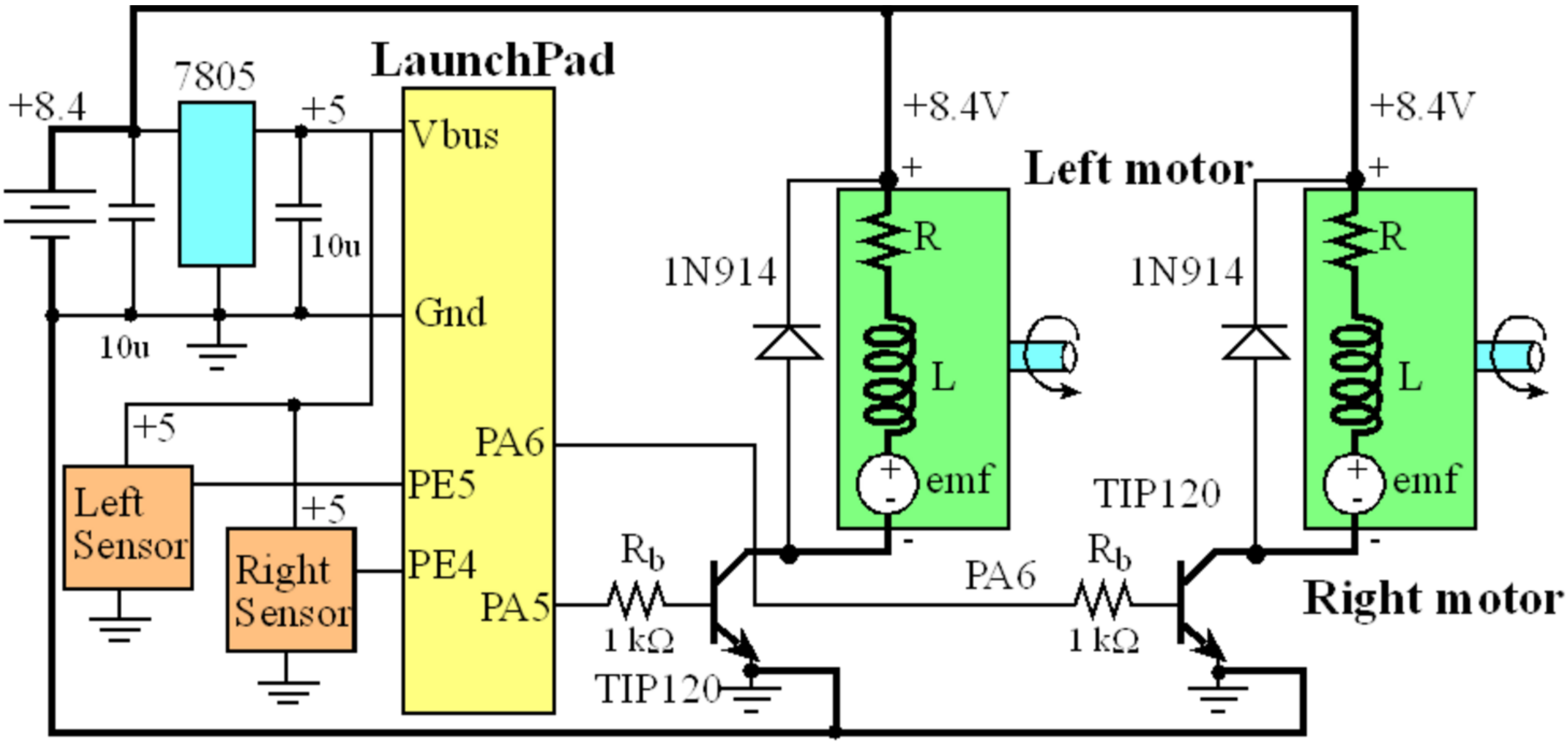
Motor interface ex2

Interfacing with a more than 5V required motor?



Motor interface ex3

Two motors



http://www.hobbytronics.co.uk/h-bridge-driver-sn754410

SN754410 is compatible and improved features compared to L293 driver

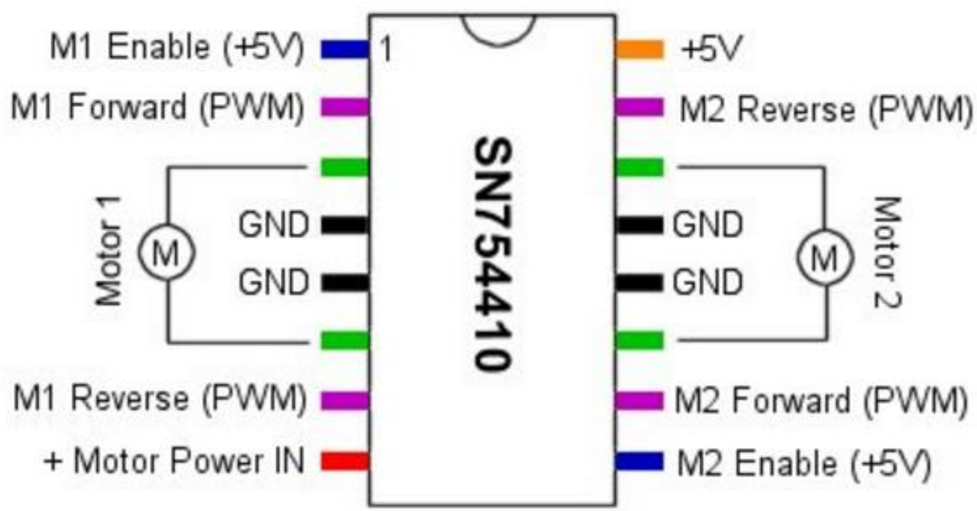
H-Bridge Motor Driver 1A - SN754410

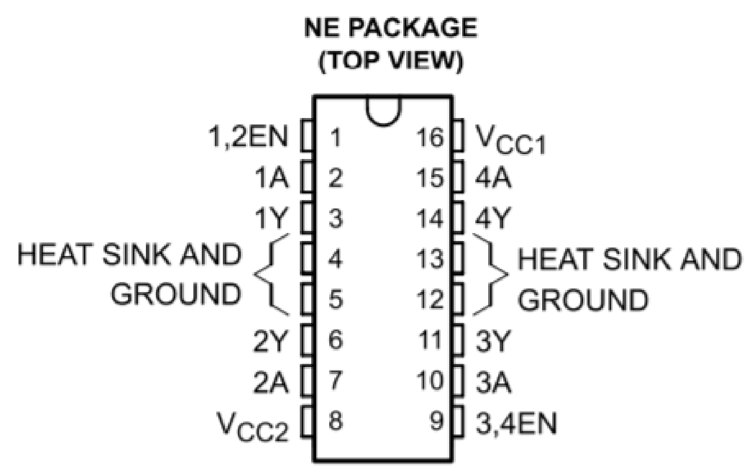
Faster, cheaper, smaller, better, right? The SN754410 Quad Half H-Bridge is just that. Capable of driving high voltage motors using TTL 5V logic levels, the SN754410 can drive 4.5V up to 36V at 1A continuous output current!

For even higher current applications, it is possible to physically stack two devices on top of each other to get almost 2 A of drive current.

The SN754410 is a quad half H-bridge IC. This allows the chip to either control 4 motors in one direction using the 4 half H-bridges or to control 2 motors in both directions using a full H-bridge for each motor.

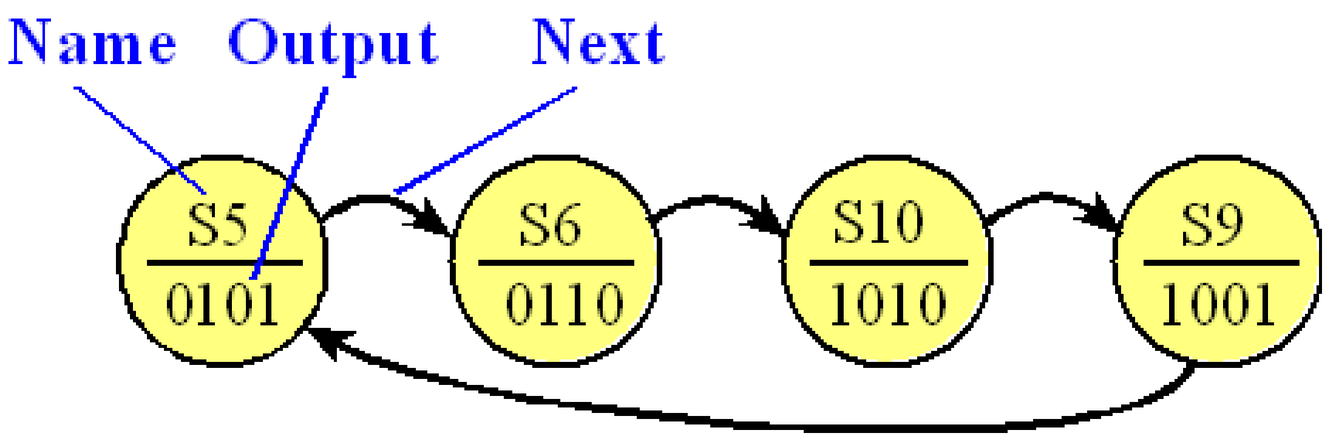
The following shows the connections for controlling 2 motors in either direction using 2 full H-bridges.





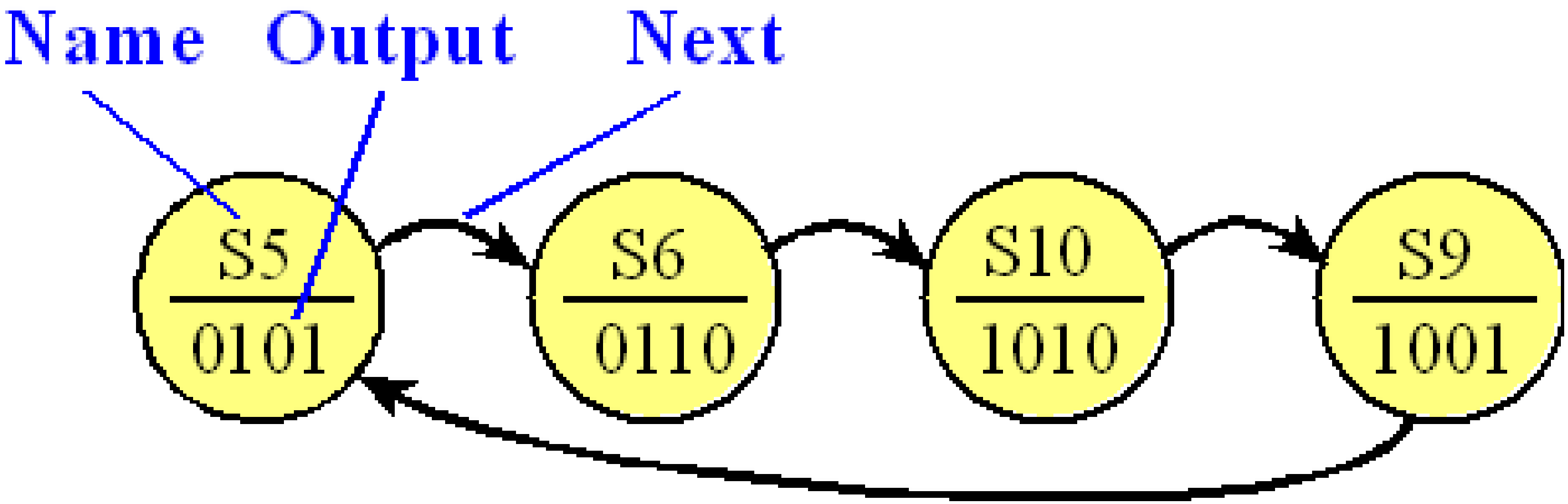
Merging FSM and Stepper motor!

- A simple Moore FSM has no inputs (like a counter), four output bits and four states
- Output pattern of 5,6,10,9... to spin the motor is a clockwise direction (CW).

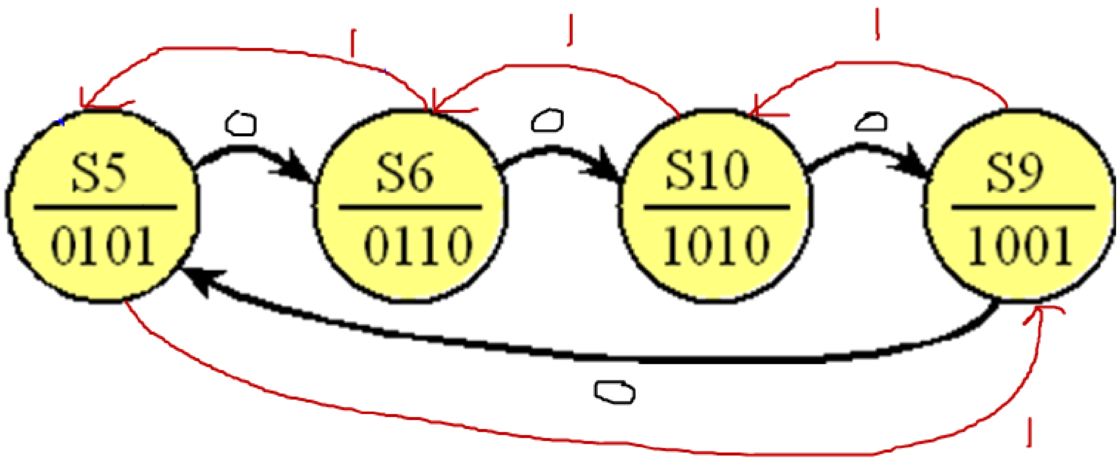


Adding CCW with CW as input options

- Input 0 → CW
- Input 1 →CCW



How to program this concept?



- Using a structure!

```
struct State{
   unsigned char Out;              // Output
   unsigned char next[2]; // 0 means CW
                          // 1 means CCW
};
```

Each state has one output and two inputs

Or

```
struct State{
   int Out;                // Output
   const struct State *Next[2]; // CW/CCW
};
```