Three actuator linear slider

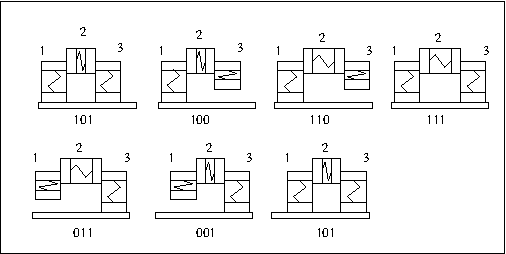
# Objective

To facilitate motion along linear path with three linear actuators which can be electromagnetic or piezo type.

# Physical principle

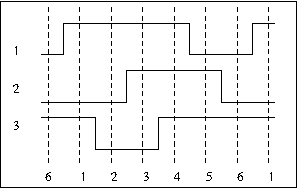
If two of the actuators provide, depending on their state, friction or free sliding against the surface, while the third one is capable of changing the distance between the first two, it is possible, by alternating contractions and extensions of thtee actuators, to achieve linear movements.

If extended actuator is coded as `1’ and contracted as `0’, the sequence for six phases is shown below. The seventh phase is the same as the first one. Free sliding is depicted as raised element, cohesion as lowered.



Note that only one of three binary positions changes from one phase to the next making this sequence similar to binary Gray code.

Three control signals, provided high level extends the actuator and low level contracts it, would look like below



# Implementation

The implementation consists of

* Signal Generator
* High Voltage (HV) side
* Mechanical part

# Signal Generator

## Overview

Signal generator is implemented on [ESP32 WROOM 32D](https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf) System on Module (SoM) from Espressif Systems. Even though it’s NRND (Not Recommended for New Designs), its low cost and upward compatibility with Espressif newer products makes it a feasible choice for prototyping.



Another advantage of ESP32 SoM is the presence of WiFi module which allows control over WiFi from smartphone or tablet.

SoM runs [MicroPython](https://micropython.org/) – a powerful and versatile development tool ideal for fast and efficient prototype building.

## MicroPython

### Installation

To install MicroPython on SoM you will need a Windows, Linux or MacOS host with the following software installed:

* python version 3.7 or later
* esptool
* ampy

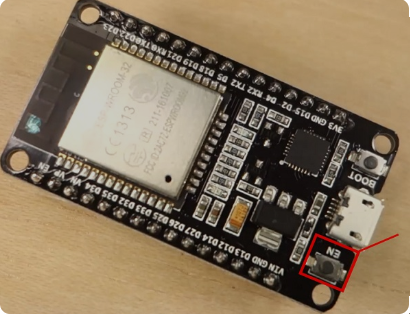
For esptool and ampy utilities installation instructions please refer to ["Esptool Installation"](https://docs.espressif.com/projects/esptool/en/latest/esp32/installation.html) and ["Adafruit Ampy Installation"](https://learn.adafruit.com/micropython-basics-load-files-and-run-code/install-ampy).

After these are in place, refer to [MicroPython Installation page](https://micropython.org/download/ESP32_GENERIC/) for instructions

Note that:

1. If the Virtual COM port on ESP32 USB connection (COM12 or /dev/ttyUSB0) was used by the host program, like minicom or putty, the program must exit and free the COM port before it can be used by esptool utility
2. The firmware must be written to ESP32 flash with offset 0x1000

After the firmware has been flashed, please start a terminal emulator ( minicom or putty) on your host with baudrate 115200 8 bits no parity and press RESET button on ESP32.



You should see something like this:

MicroPython v1.23.0 on 2024-06-02; Generic ESP32 module with ESP32

Type "help()" for more information.

>>>

This is MicroPython console, also called REPL (Read-Eval-Print Loop) and it makes all MicroPython functionality available. For example, to get the desired signal pattern on pins 12,13 and 14, you can enter:

>>> from time import sleep

>>> from machine import Pin

>>>

>>> p1 = Pin(12, Pin.OUT)

>>> p2 = Pin(13, Pin.OUT)

>>> p3 = Pin(14, Pin.OUT)

>>>

>>> delay=.25

>>>

>>> while (True):

... p1.on()

... sleep(delay)

... p3.off()

... sleep(delay)

... p2.on()

... sleep(delay)

... p3.on()

... sleep(delay)

... p1.off()

... sleep(delay)

... p2.off()

... sleep(delay)

...

...

...

Using MicroPython filesystem storage, makes this, as well as other features, like WiFi connectivity, more user-friendly.