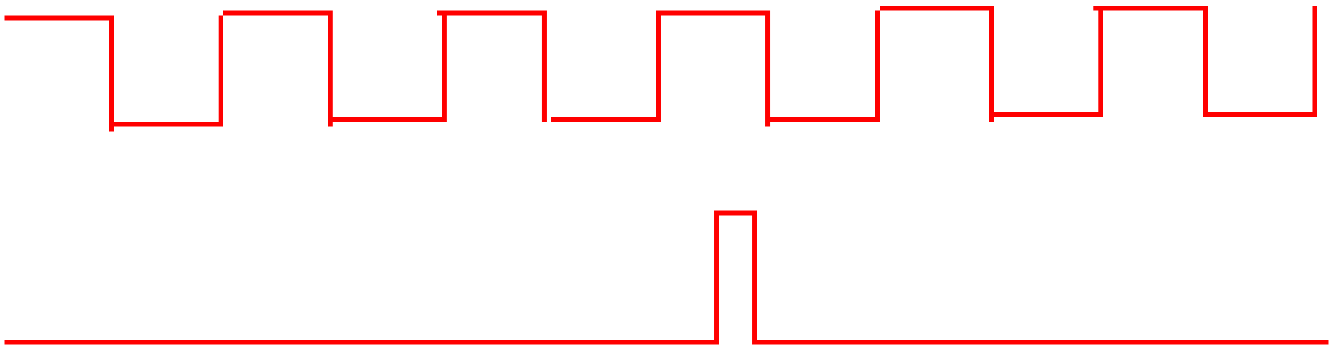
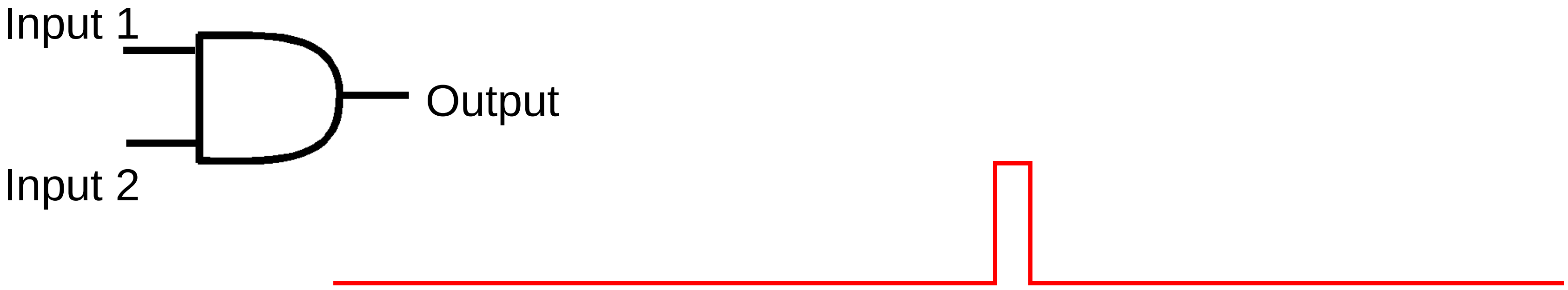
Lab 12: Timing and Control

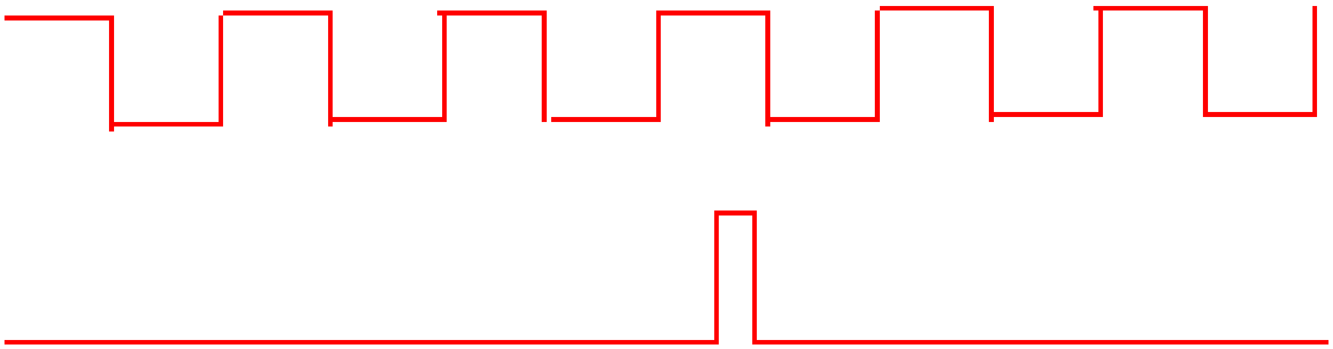
Digital Timing: True or False; 1 or 0



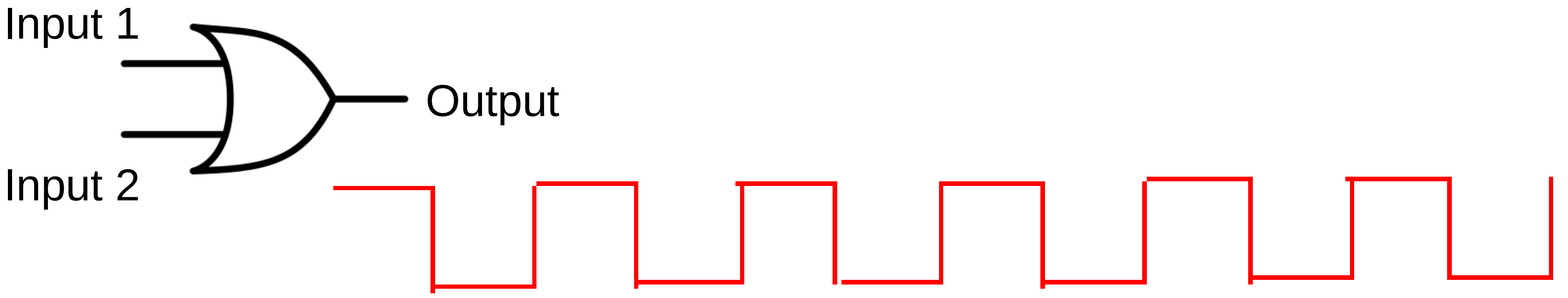
AND Gate



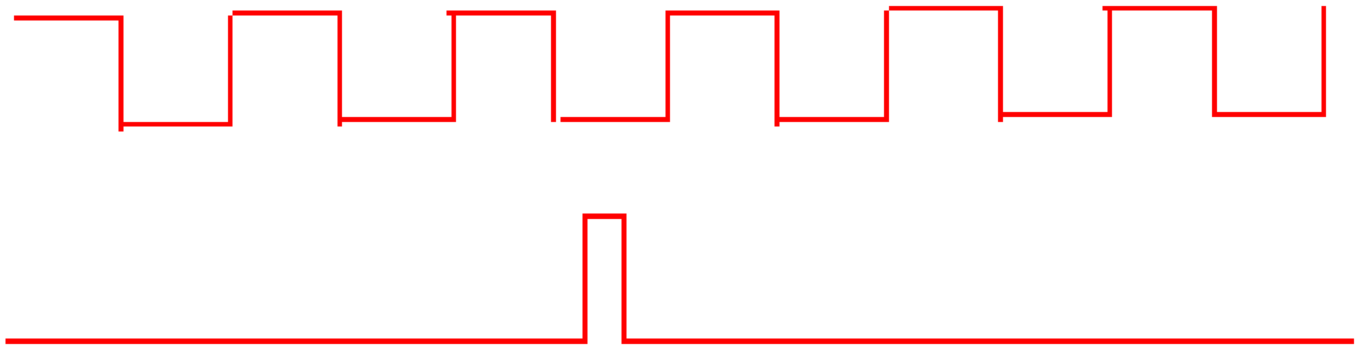
Digital Timing: True or False; 1 or 0



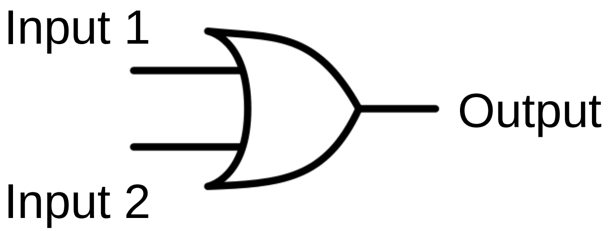
OR Gate

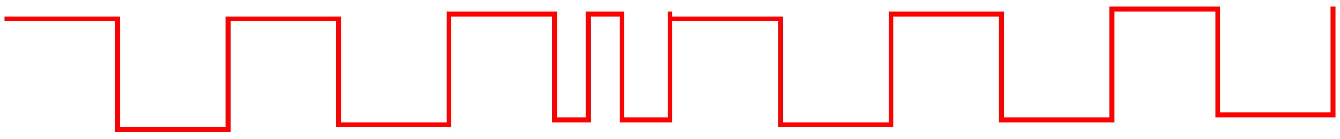


Digital Timing: True or False; 1 or 0

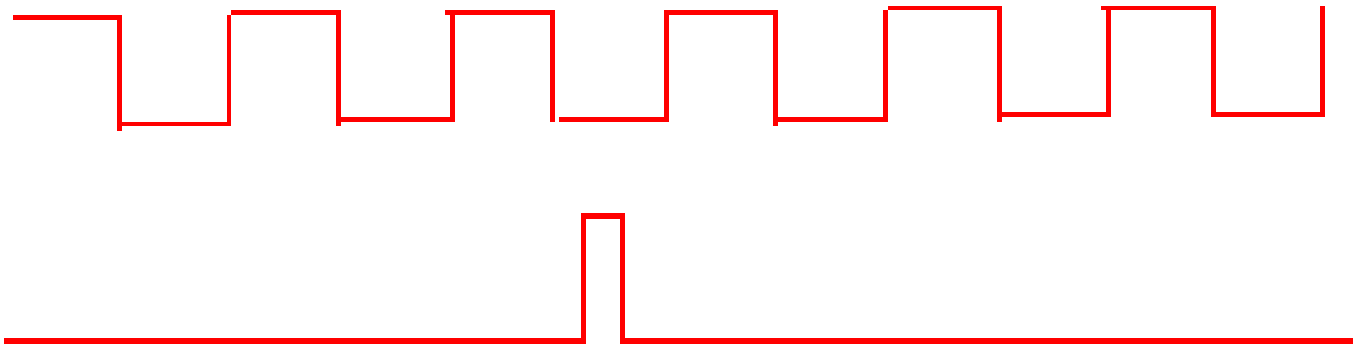


OR Gate

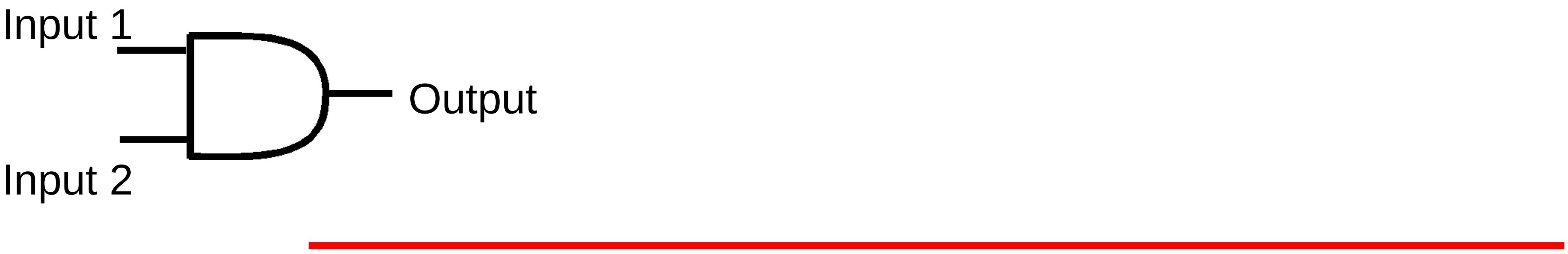




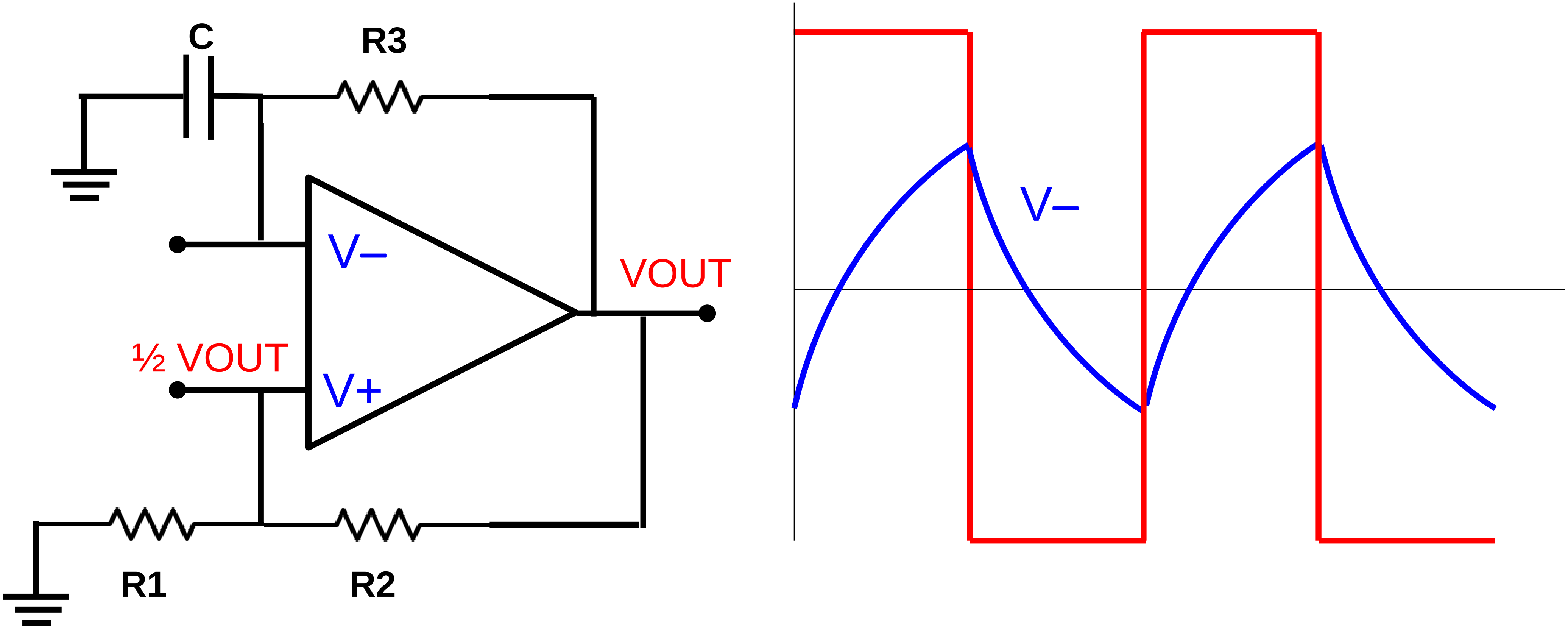
Digital Timing: True or False; 1 or 0



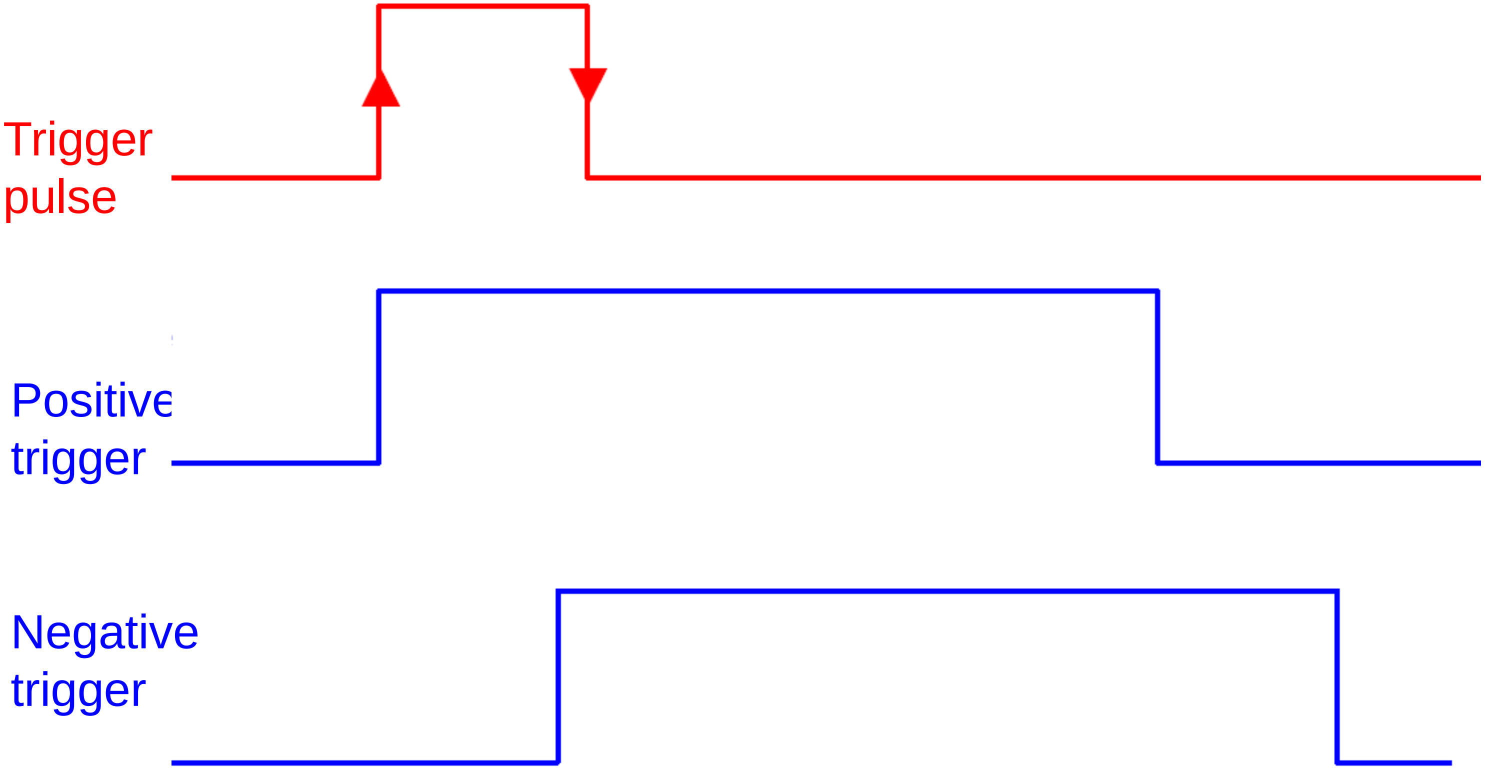
AND Gate



Relaxation Oscillator



Edge triggering of single-shot square pulse

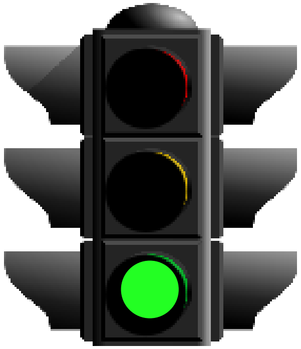


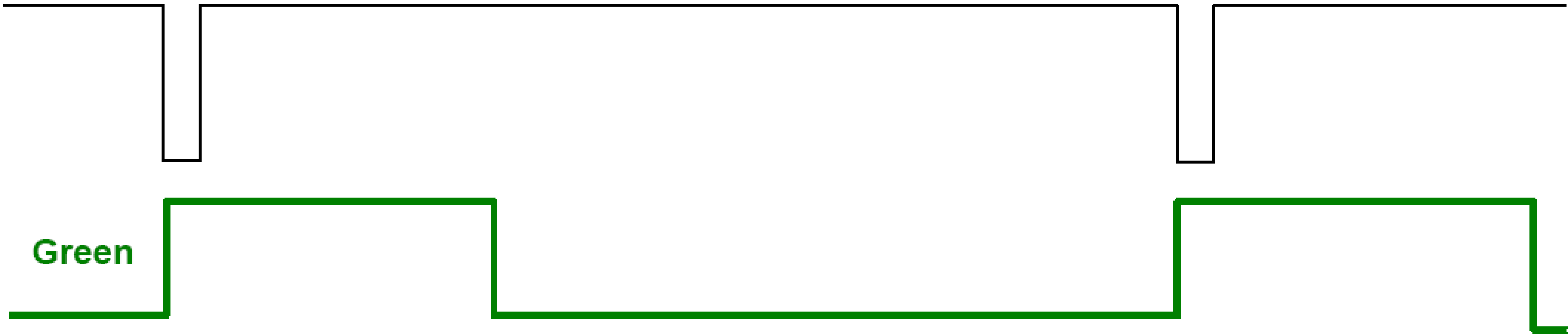
Trigger pulse

Positive trigger

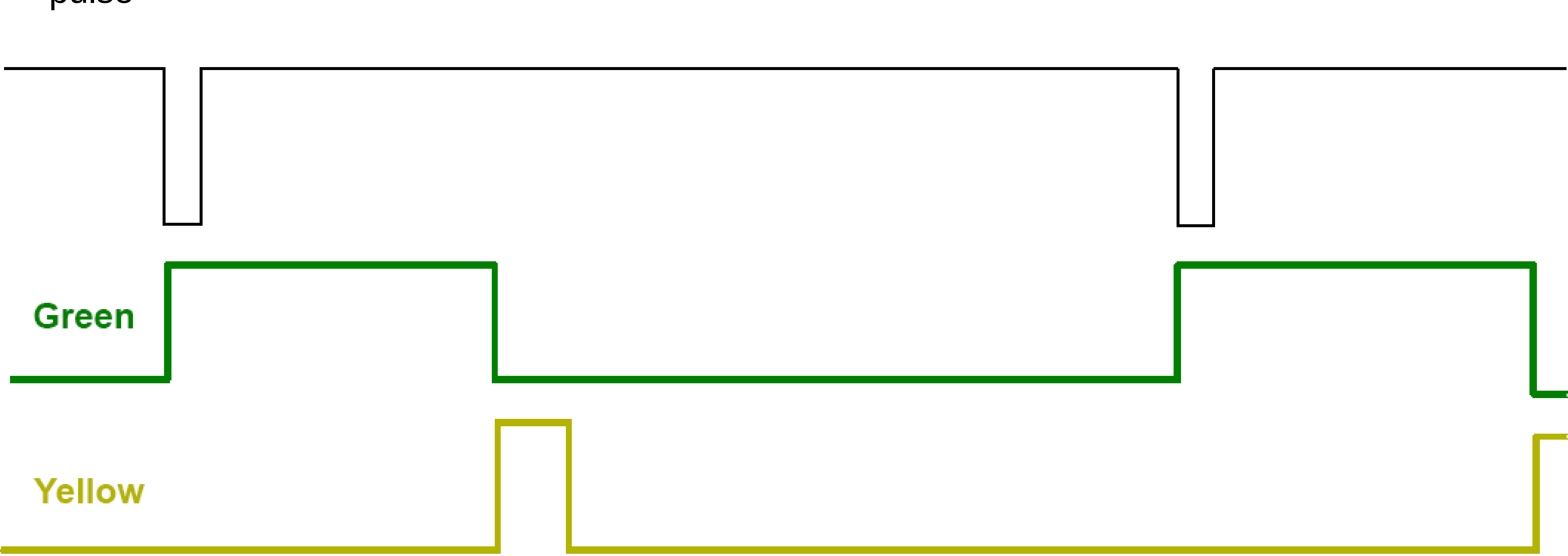
Negative trigger

Timing diagram: Traffic Light

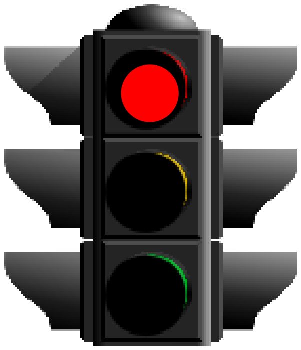


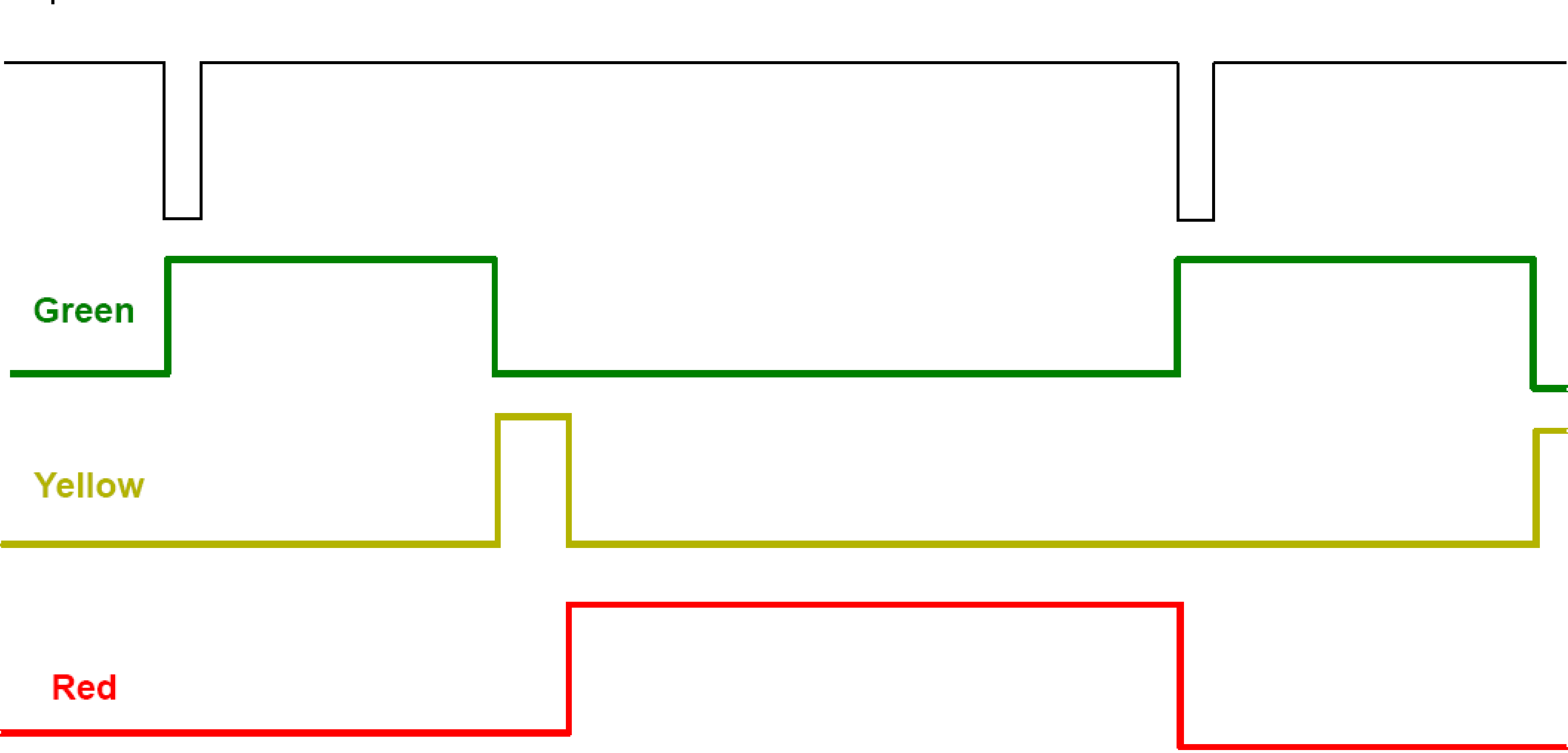


Timing diagram: Traffic Light

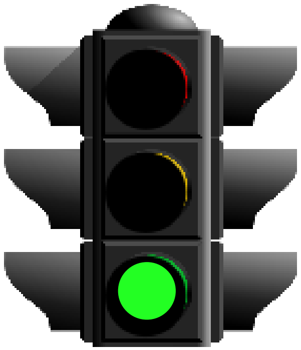


Timing diagram: Traffic Light

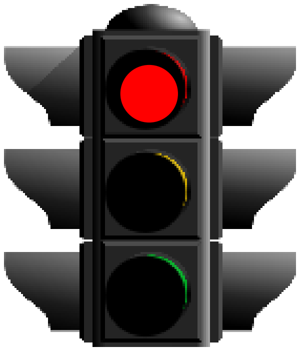




Timing: Traffic Light (State 1)

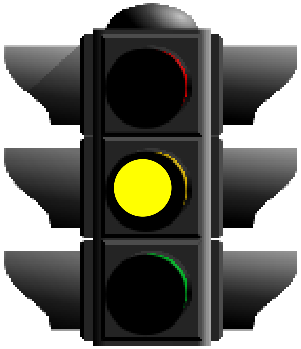


North-South bound

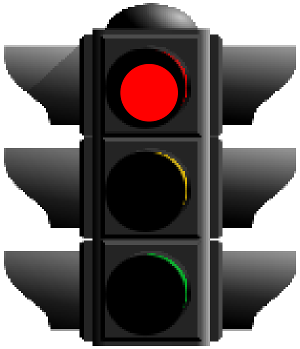


East-West bound

Timing: Traffic Light (State 2)

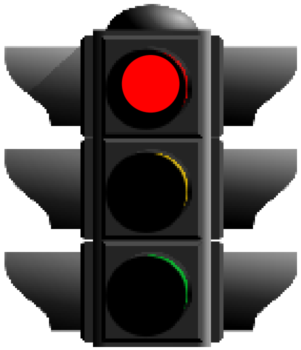


North-South bound

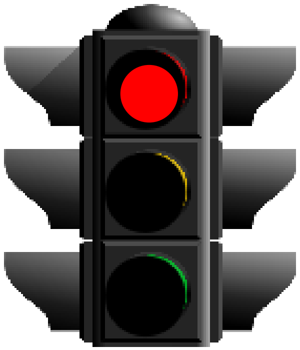


East-West bound

Timing: Traffic Light (State 3)

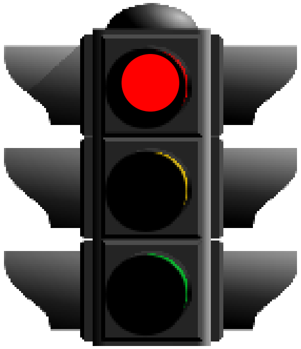


North-South bound

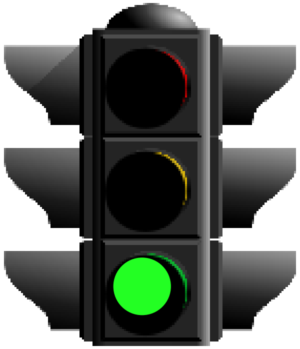


East-West bound

Timing: Traffic Light (State 4)

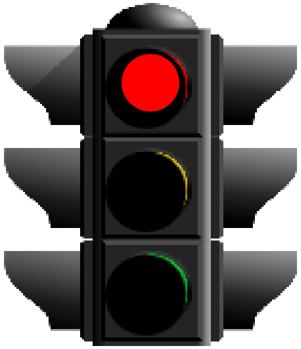


North-South bound

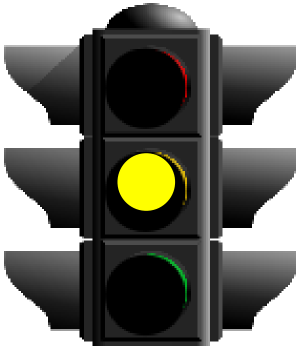


East-West bound

Timing: Traffic Light (State 5)

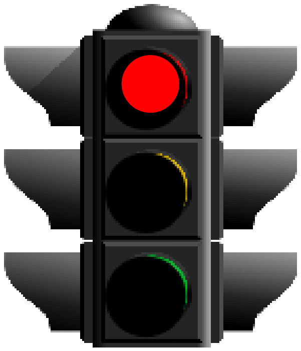


North-South bound

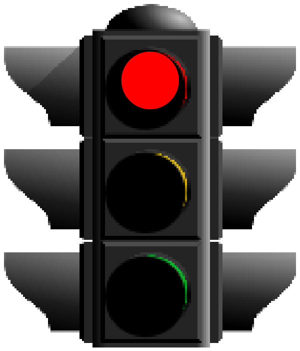


East-West bound

Timing: Traffic Light (State 6)

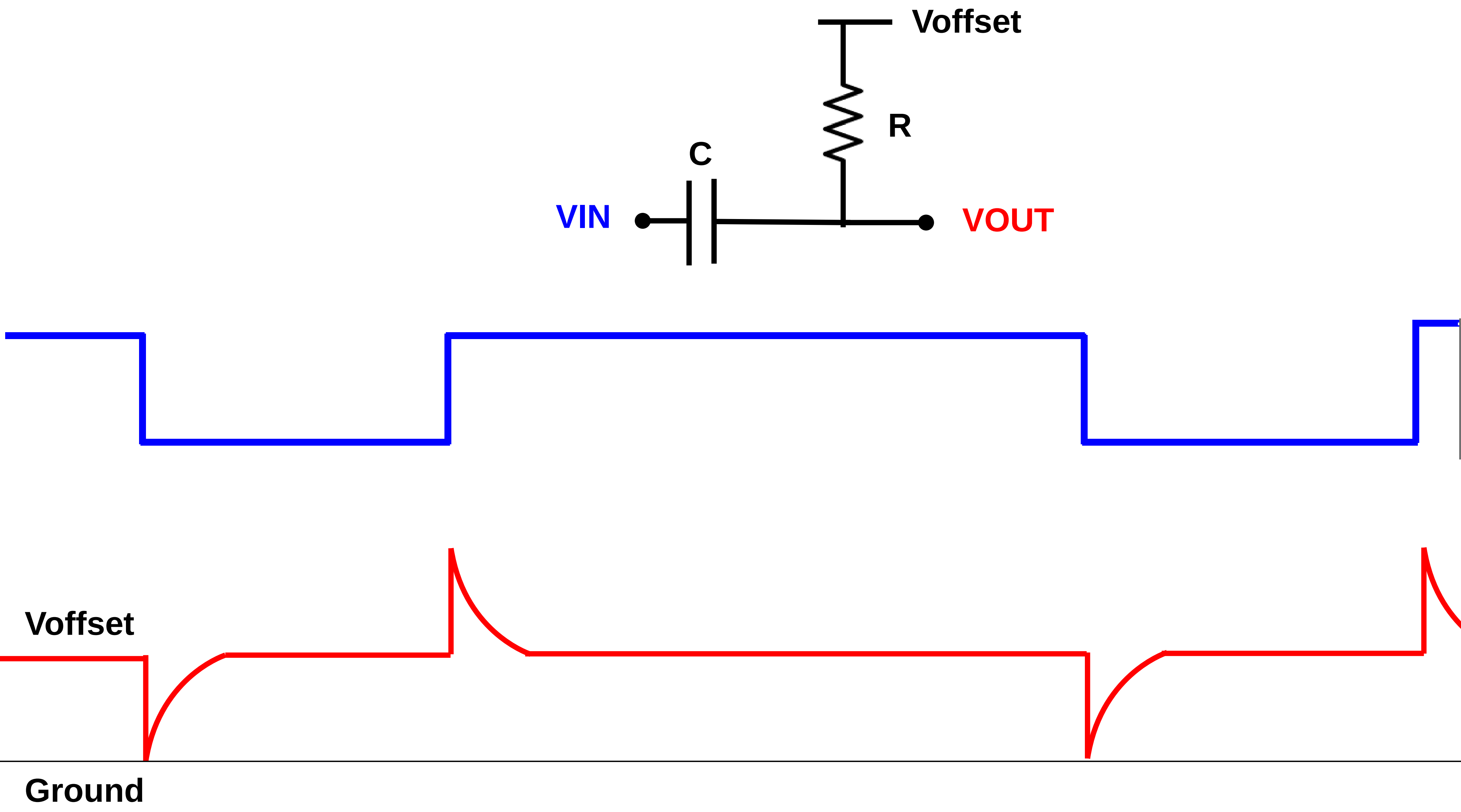


North-South bound



East-West bound

Differentiating pulses with RC circuit



100% Hardware

- Cheaper
    - No PC
    - No proprietary software/OS
- More reliable
    - No PC to crash
- More compact
    - Depends on complexity
- Inherently faster
    - Interface I/O limits speed
- Less power

LabView + Hardware

- Easier to setup/troubleshoot
- Easier to expand/adapt new versions
- LabView code more transparent than circuit schematic
- Probably better approach for research lab environment

EMBEDDED SYSTEMS

Dedicated computer hardware replaces the general purpose PC

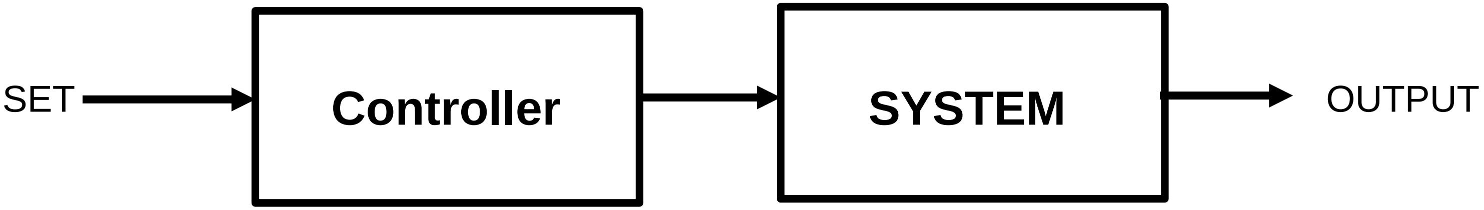
Inexpensive, low-power micro-controllers (RAM, Flash, I/O, etc)

Optimized to solve a specific problem or task

Examples

- Digital watch
- MP3 player
- Smoke detector
- Game console
- PDA
- Digital camera
- Cellphone
- GPS
- Microwave oven

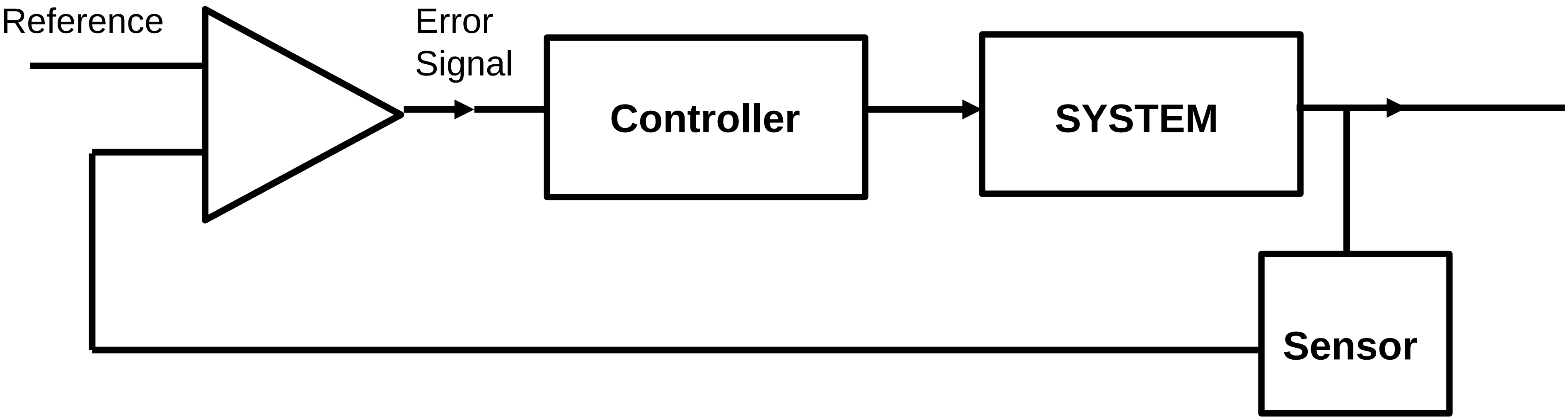
CONTROL: OPEN LOOP



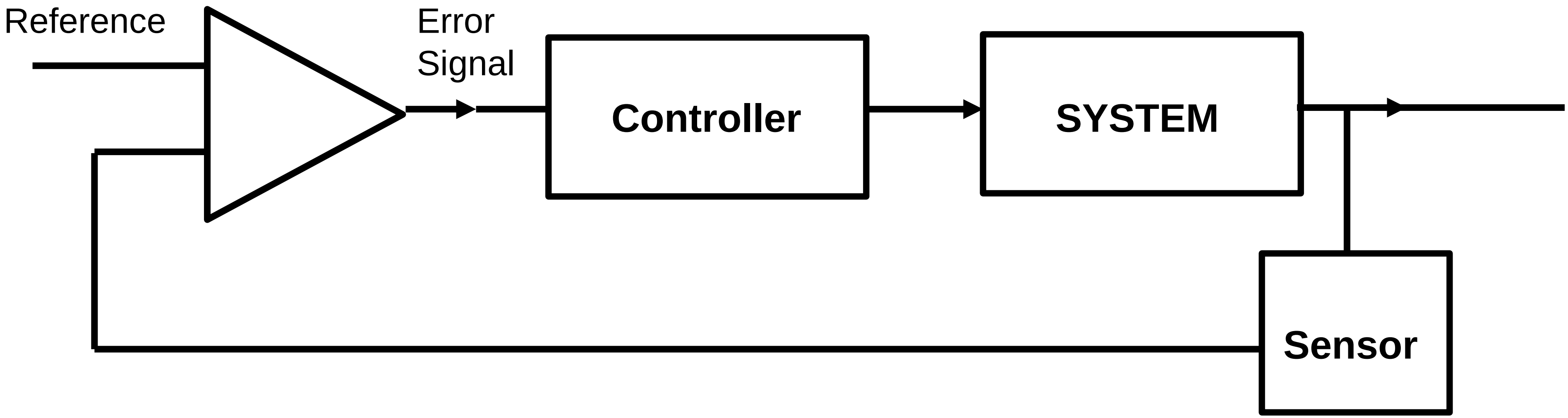
EXAMPLES:

- Washing machine
- Lawn sprinkler system

CONTROL: CLOSED LOOP

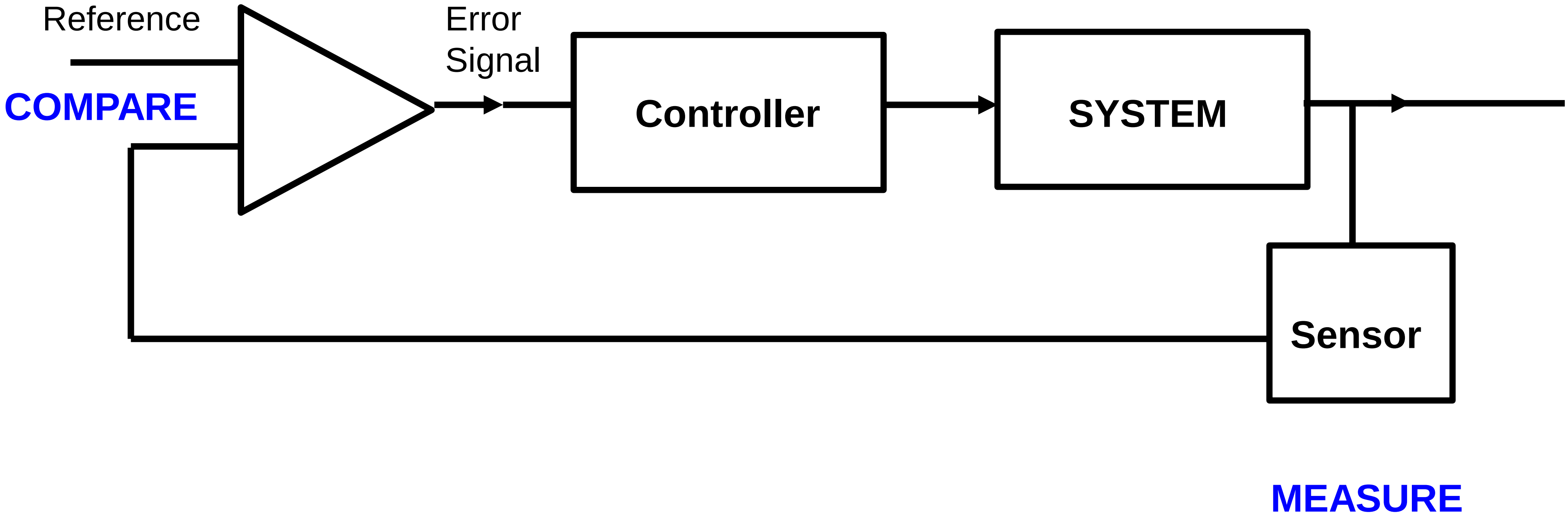


CLOSED LOOP

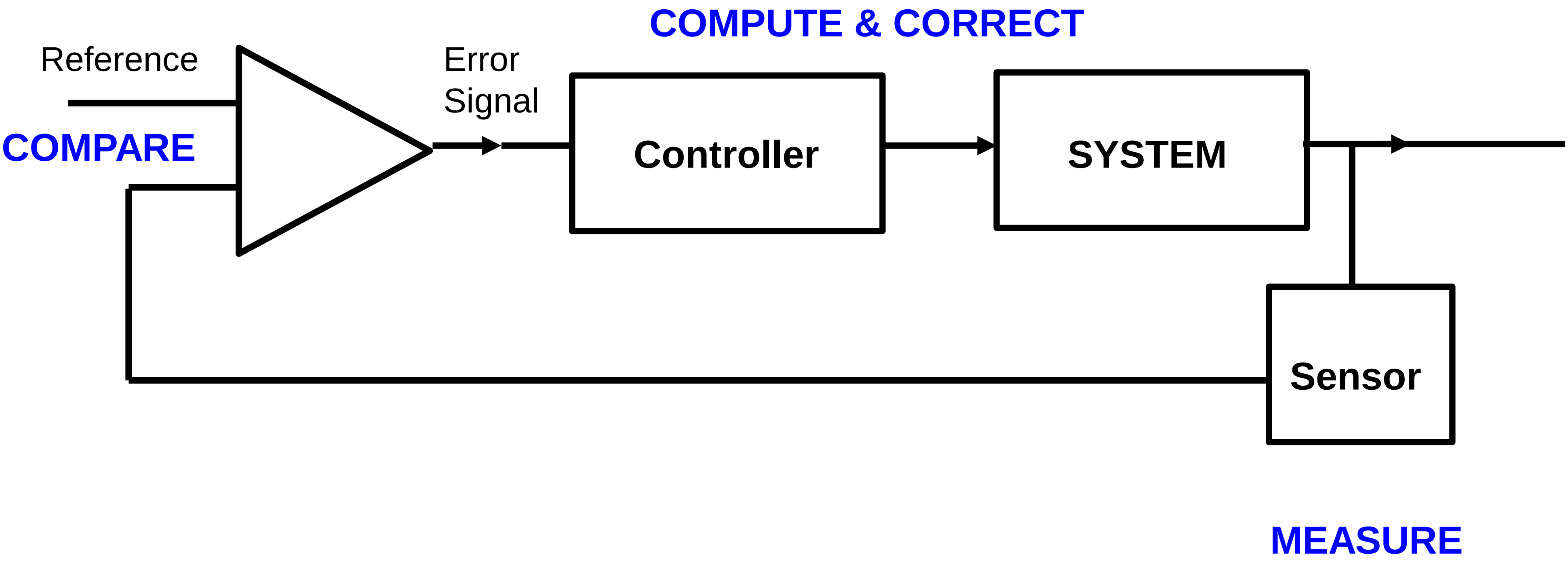


MEASURE

CLOSED LOOP


Reference
COMPARE
Error
Signal
Controller
SYSTEM
Sensor
MEASURE

CLOSED LOOP


COMPUTE & CORRECT
Reference
COMPARE
Error
Signal
Controller
SYSTEM
Sensor
MEASURE

Examples of closed loop controllers

Cruise control on car

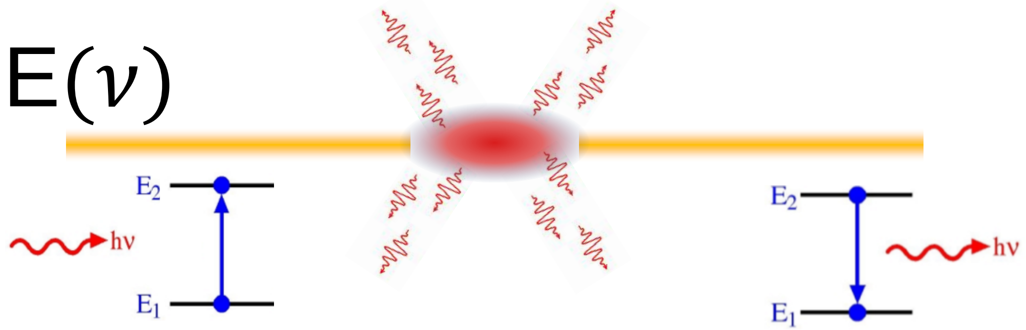
Thermostat on furnace

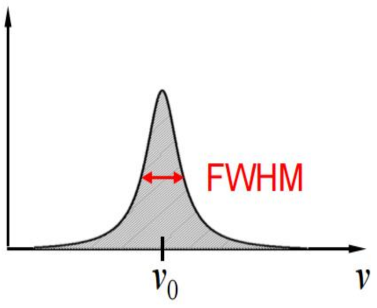
Water level in hot water heater or swamp cooler

Cabin air pressure in passenger plane

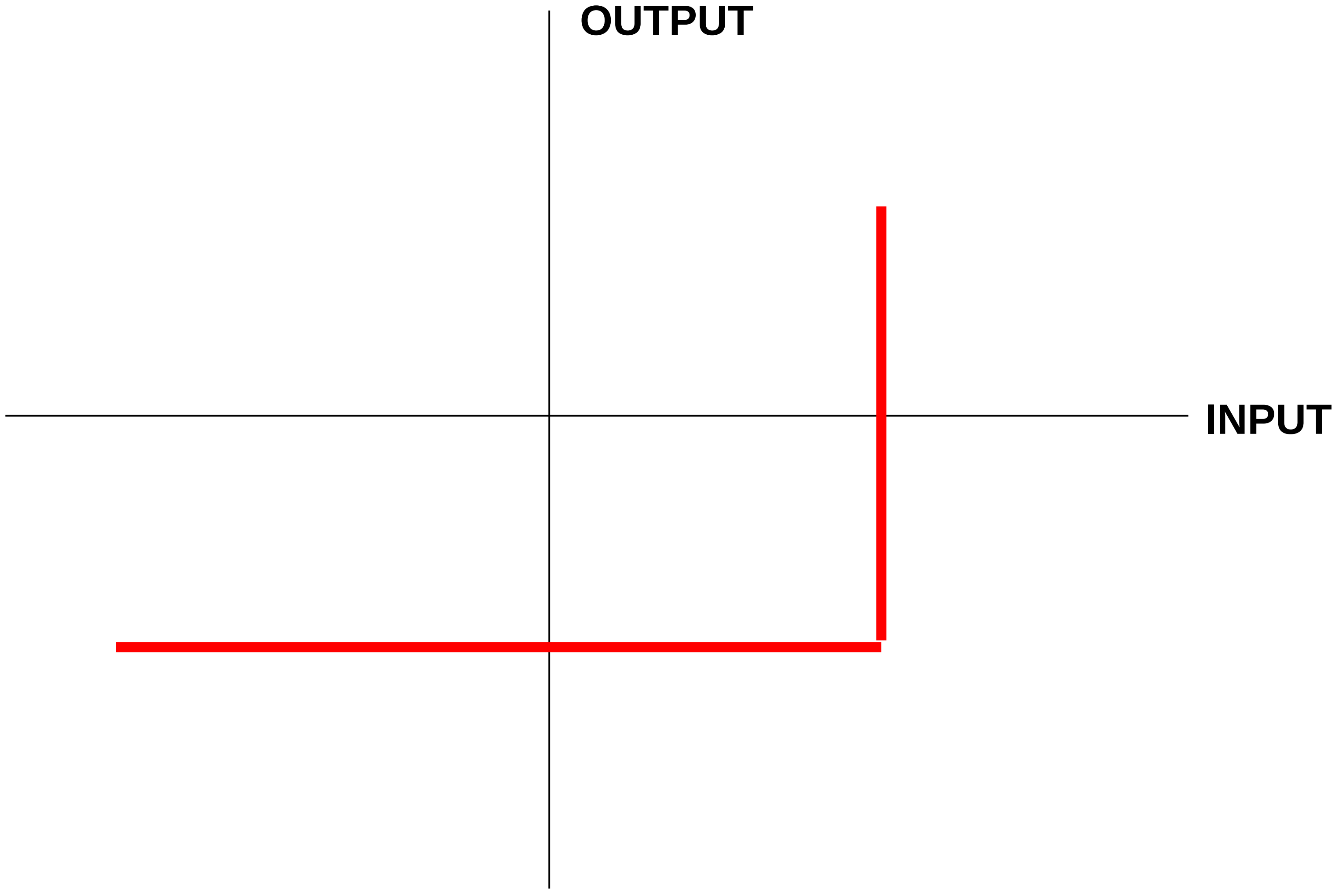
Clock on a PC

Optical clocks and atomic clocks

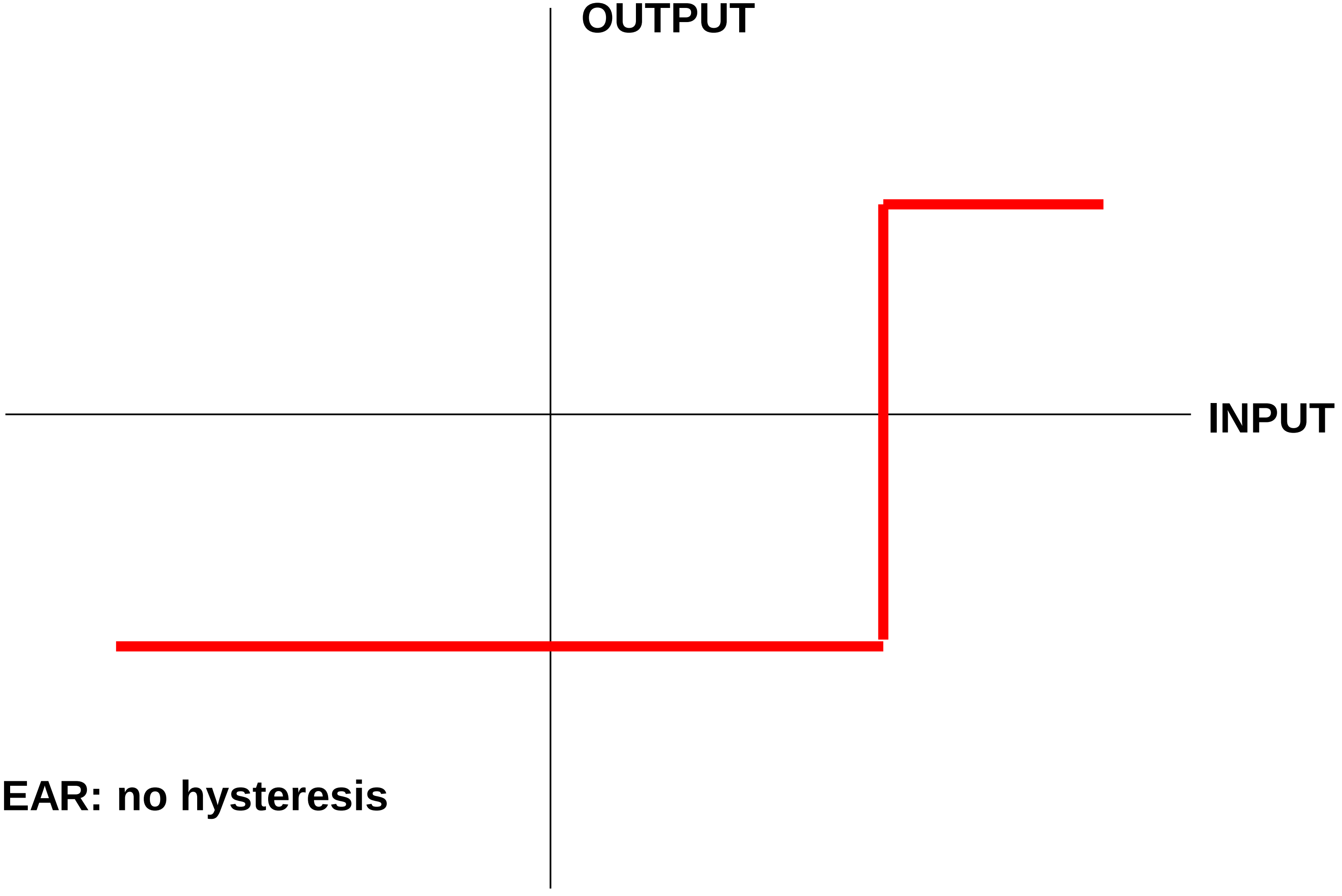




HYSTERETIC CLOSED LOOP CONTROLLER

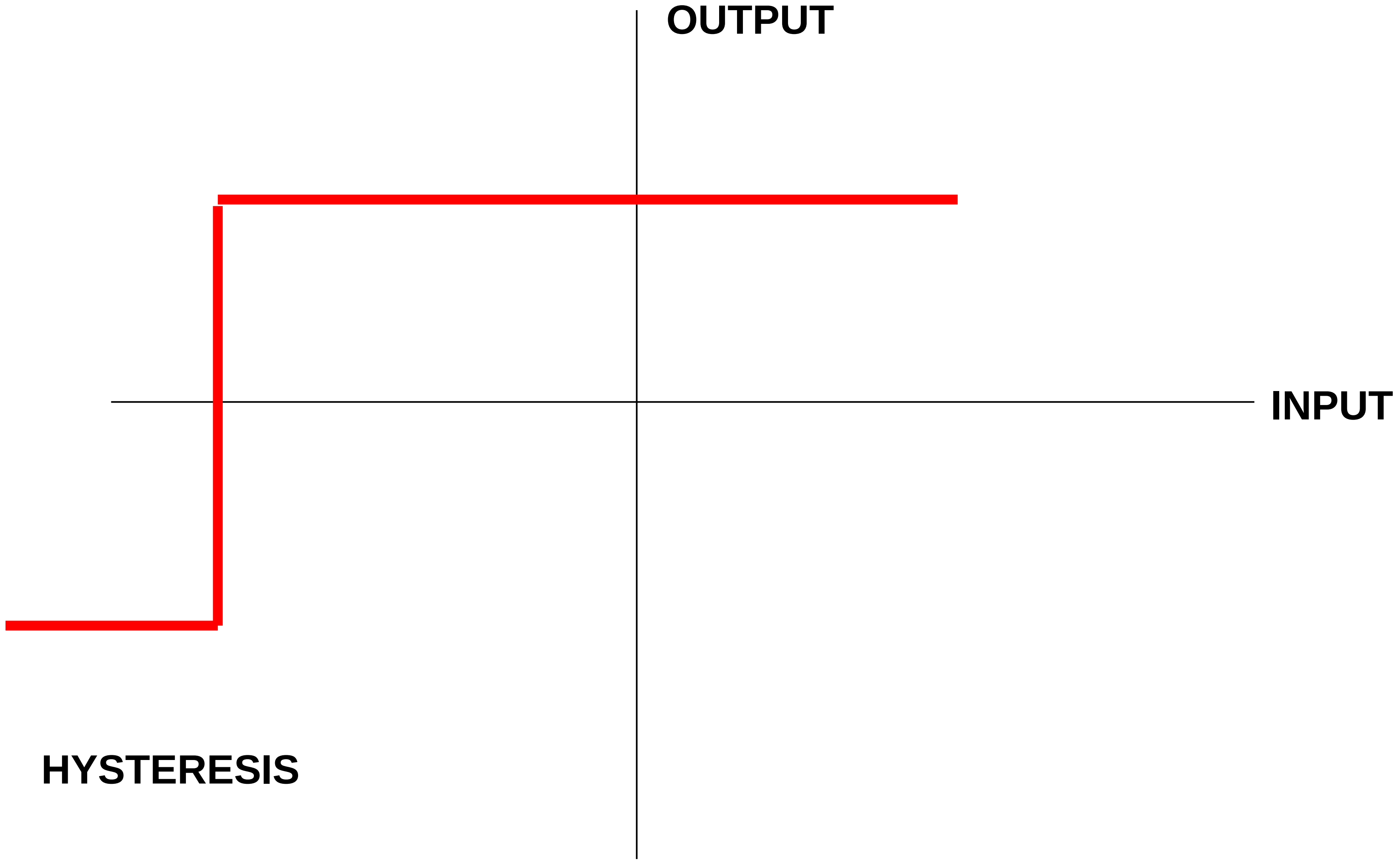


HYSTERETIC CLOSED LOOP CONTROLLER

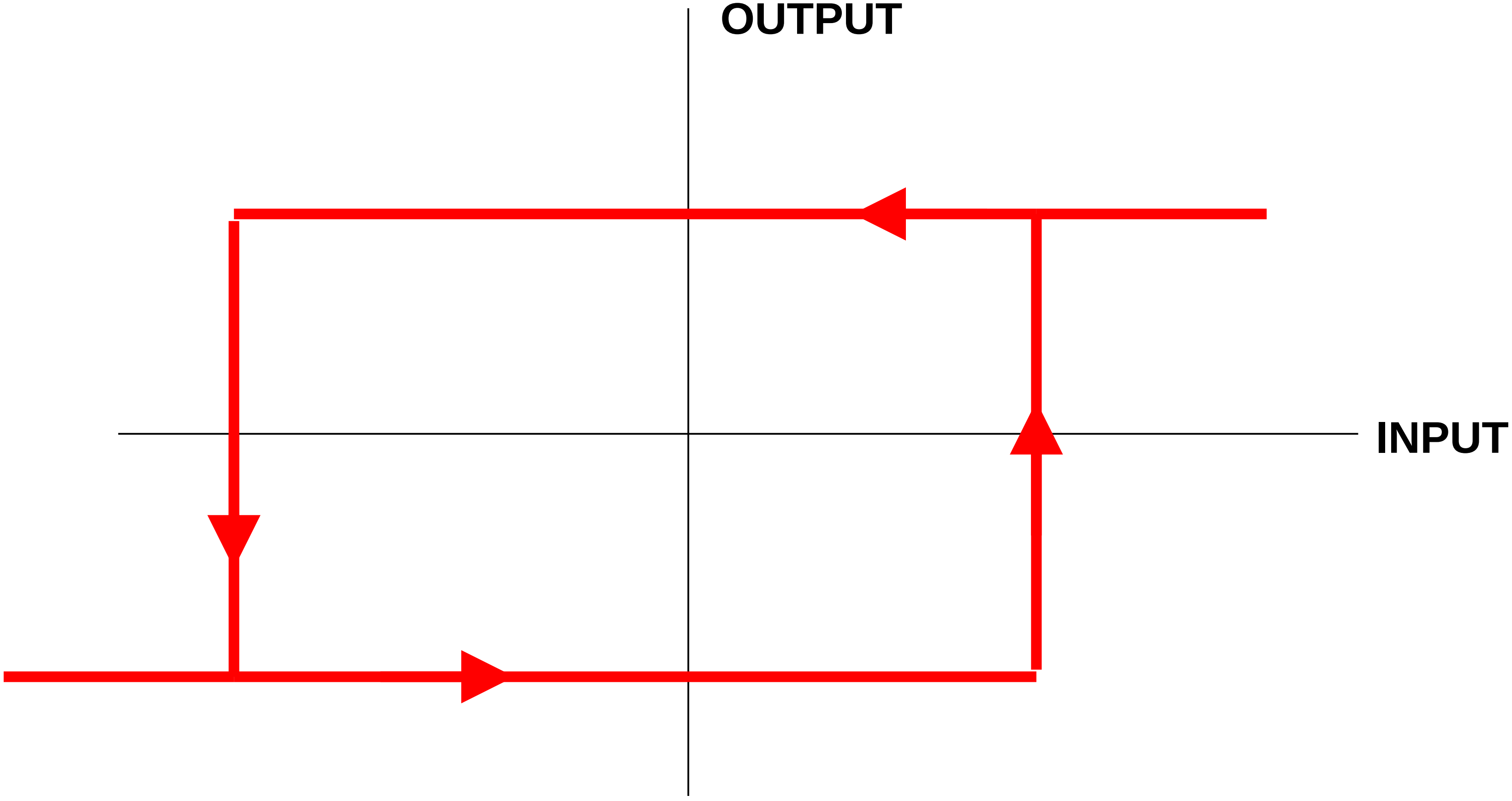


LINEAR: no hysteresis

HYSTERETIC CLOSED LOOP CONTROLLER

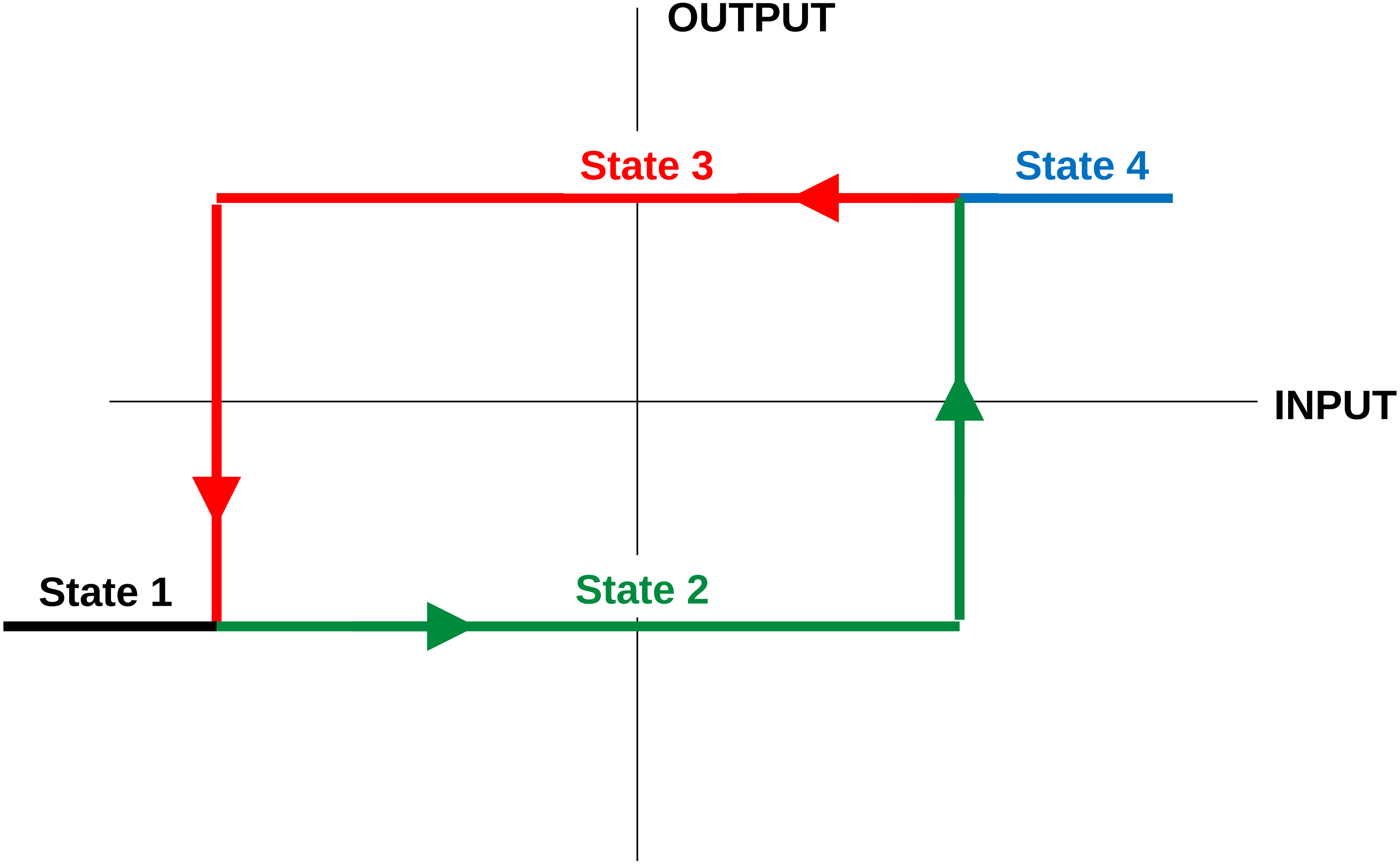


HYSTERETIC CLOSED LOOP CONTROLLER

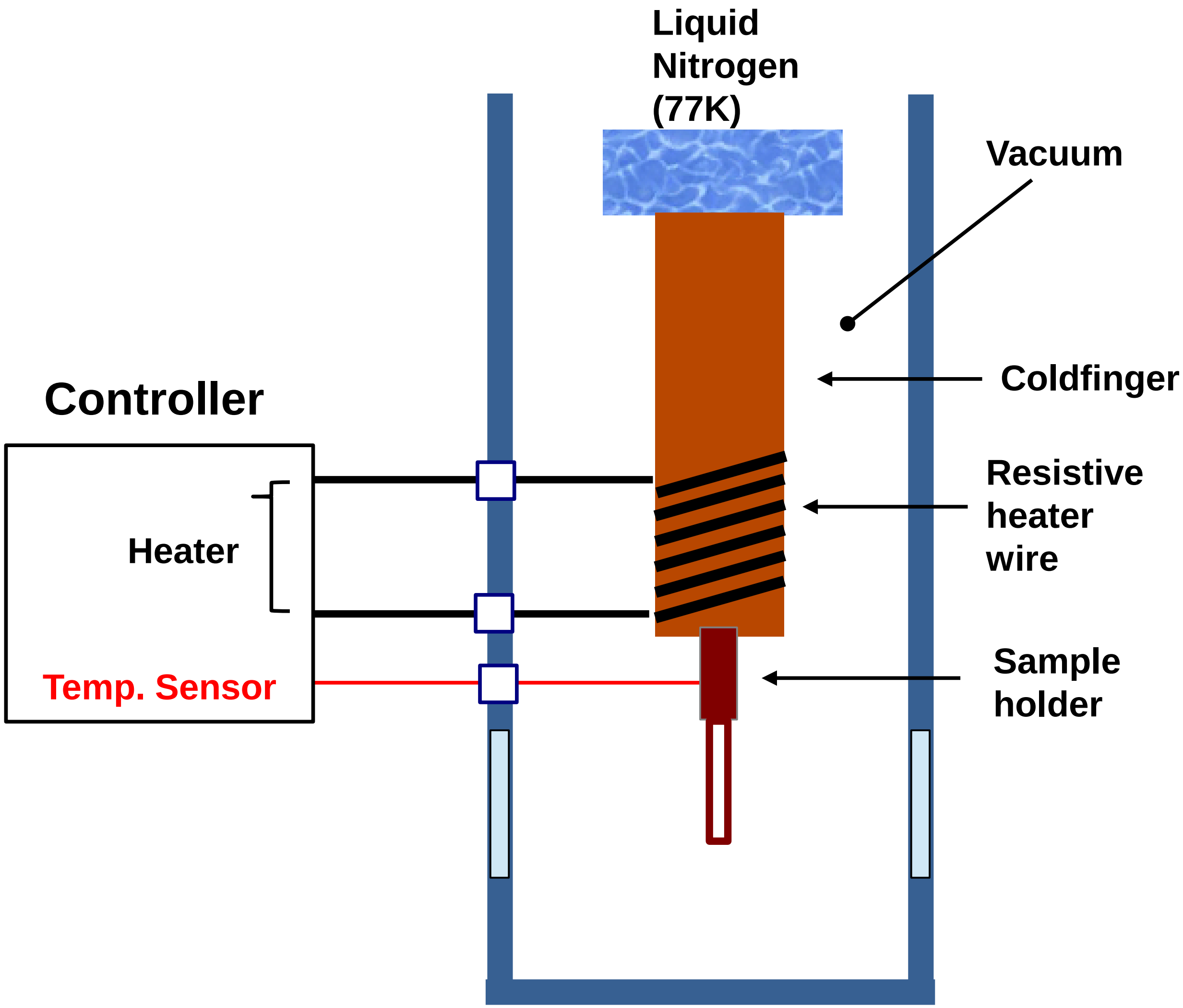


OUTPUT state depends on direction of INPUT state (its history)

HYSTERETIC CLOSED LOOP CONTROLLER



Cryostat Temperature Controller



Liquid Nitrogen (77K)

Vacuum

Coldfinger

Controller

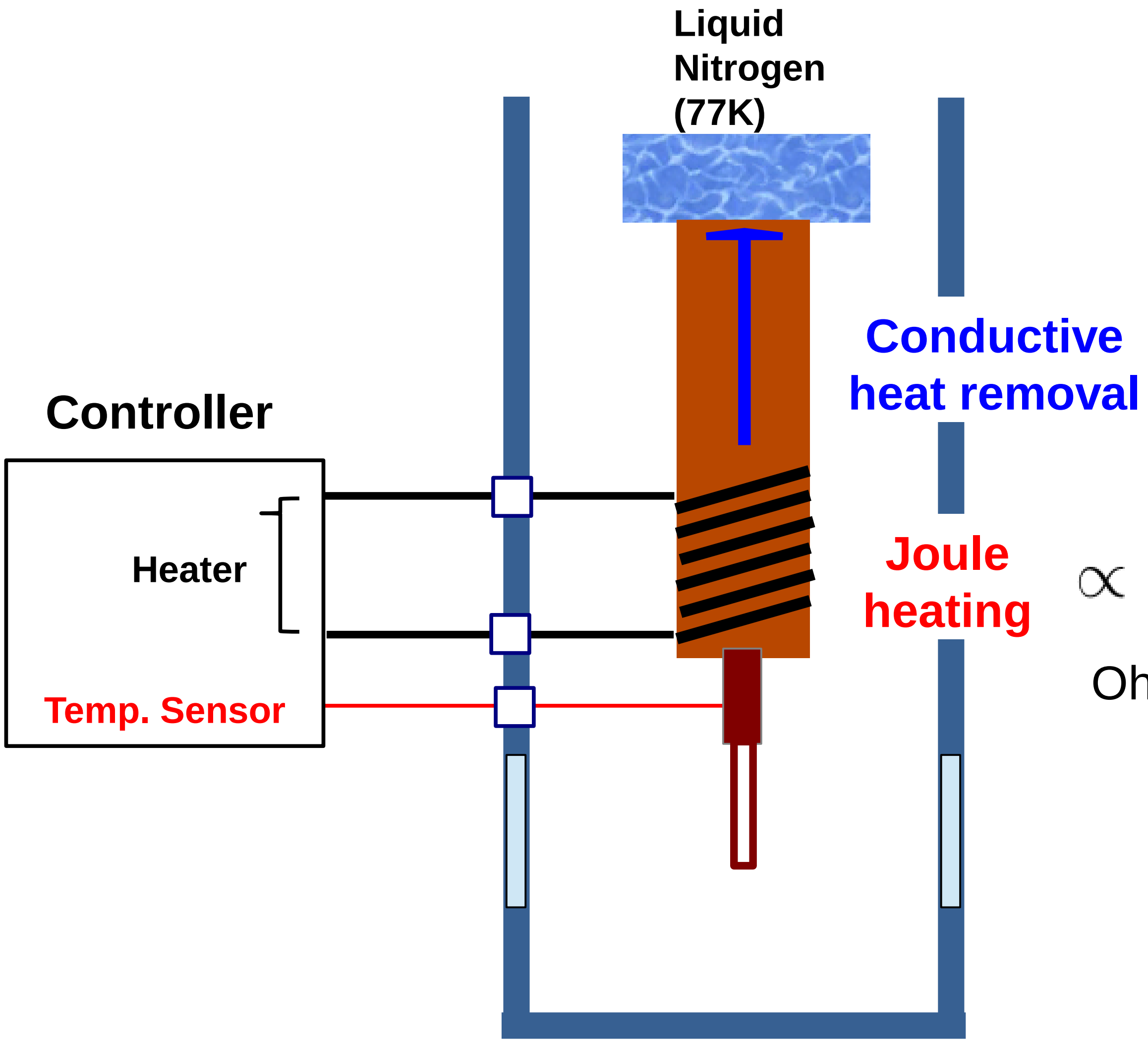
Heater

Resistive heater wire

Temp. Sensor

Sample holder

Cryostat Temperature Controller



Liquid Nitrogen (77K)

Controller

Heater

Temp. Sensor

Conductive heat removal

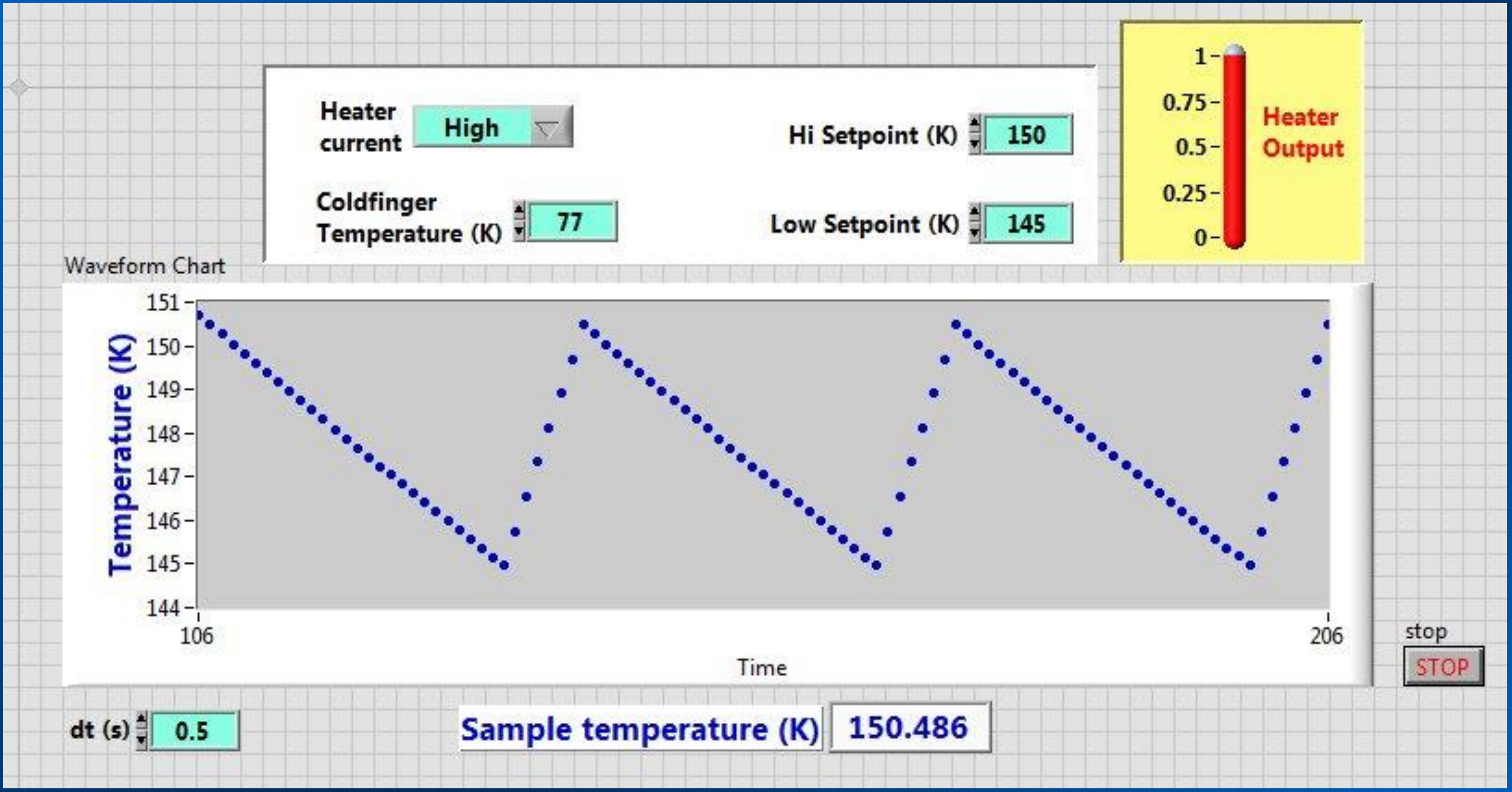
Joule heating

$$\propto \nabla T = \frac{dT}{dz} = k\Delta T$$

Fourier's Law

$$\propto I^2 R$$

Ohm's Law



Heater current: High

Hi Setpoint (K): 150

Coldfinger Temperature (K): 77

Low Setpoint (K): 145

dt (s): 0.5

Sample temperature (K): 150.486

Two-direction traffic light implemented with state-machine on a $1 TI micro-controller. Battery powered.

```
//Runs a 6 LEDS in sequence, simulating a two-direction traffic light.
//Implemented with timer interrupts using the 12 kHz VLO clock.
//MCU spends most of its time in LPM3.
//This is a state-machine with 6 states


#include <msp430g2253.h>
#ifndef TIMER0_A1_VECTOR
#define TIMER0_A1_VECTOR    TIMERA1_VECTOR
#define TIMER0_A0_VECTOR    TIMERA0_VECTOR
#endif


int main(void) {
   WDTCTL = WDTPW | WDTHOLD;                          // Stop watchdog timer
   P1DIR |= BIT0 + BIT4 + BIT6; // Direction 1: Red, yellow, green on P1.0, 1.4, & 1.6
   P1DIR |= BIT1 + BIT2 + BIT3; // Direction 2: Red, yellow, green on P1.1, 1.2, & 1.3
   P1OUT |= BIT0 + BIT1 + BIT2 + BIT3 + BIT4 + BIT6; //Turn all LEDs on
   BCSCTL3 |= LFXT1S_2;  //Set low frequency clock to the 12 kHz VLO
   //Divide the VLO (ACLK) as follows: DIVA_0,1,2,3 correspond to divide by 1,2,4,8
   //For DCO = 150 kHz, set RSELx = 1 and DCOx = 3
   BCSCTL1 = DIVA_2;
   TACCR0=6000;  //12000 counts for 1 second at DIVA_0;
  //Maximum count is 65535 (unsigned 16-bit)
   TACCTL0 |= CCIE;  //Enable timer interrupt
   //Configure TACTL last because it starts the timer
   TACTL |= TASSEL_1 + MC_1;  //Set Timer A to ACLK; MC_1 to count up to TACCR0.
   _BIS_SR(GIE); //Enable global interrupts.  Shouldn't be set until module is fully configured
   LPM3;
   P1OUT &= ~(BIT0 + BIT1 + BIT2 + BIT3 + BIT4 + BIT6); //Turn off all LEDs
```

```
for (;;) {                    // Endless loop
    //State 1
                   P1OUT &= ~BIT2; //Yellow 2 off
           P1OUT |= BIT0 + BIT1; //Both red LEDs on
                   TACCR0=12000; //2 second wait
            LPM3;
    //State 2
           P1OUT &= ~BIT0; //Red 1 off
           P1OUT |= BIT6; //Green 1 on
                   TACCR0=30000; //5 second wait
                   LPM3;
    //State 3
                   P1OUT &= ~BIT6; //Green 1 off
                   P1OUT |= BIT4; //Yellow 1 on
                   TACCR0=12000; //2 second wait
                   LPM3;
    //State 4
                   P1OUT &= ~BIT4; //Yellow 1 off
                   P1OUT |= BIT0 + BIT1; //Both red LEDs on
                   TACCR0=12000; //2 second wait
                   LPM3;
    //State 5
                   P1OUT &= ~BIT1; //Red 2 off
                   P1OUT |= BIT3; //Green 2 on
                   TACCR0=24000; //4 second wait
                   LPM3;
    //State 6
           P1OUT &= ~BIT3; //Green 2 off
                   P1OUT |= BIT2; //Yellow 2 on
                   TACCR0=12000; //2 second wait
                   LPM3;
                       }
}


#pragma vector=TIMER0_A0_VECTOR
 __interrupt void timerfoo (void)
{
                   LPM3_EXIT;
 }
```