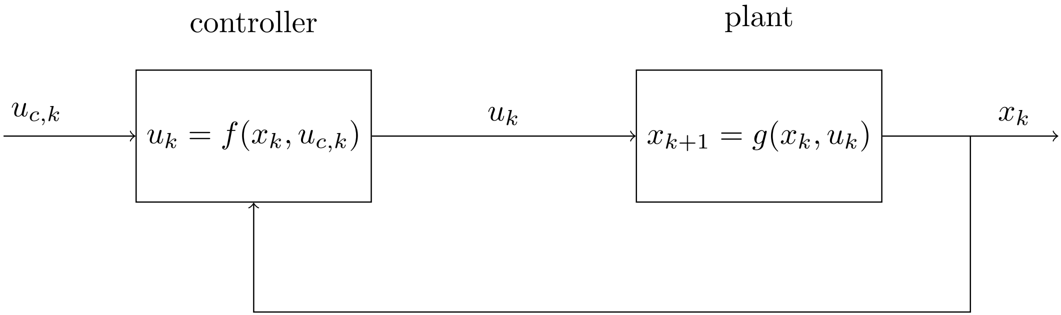
Logic control and boolean algebra

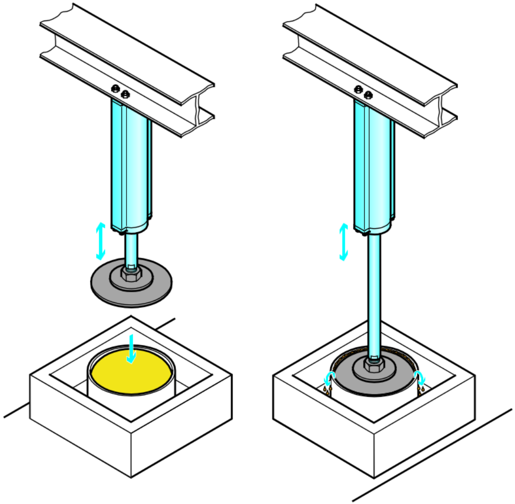
Kjartan Halvorsen

April 24, 2020

A logic control loop



Cheese pressing example



LATEX {From FESTO Didactic}

Cheese pressing example - Variables

Activating solenoid S1 extends the cylinder, activating solenoid S2 retracts the cylinder.

State variable

$$x_k = \begin{cases} 0 & \text{Cylinder retracted} \\ 1 & \text{Cylinder extended} \end{cases}$$

Control signal

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix},$$

with

$$u_1 = \begin{cases} 0 & \text{Don't activate S1} \\ 1 & \text{Activate S1} \end{cases}$$

$$u_2 = \begin{cases} 0 & \text{Don't activate S2} \\ 1 & \text{Activate S2} \end{cases}$$

Command signal

$$u_c = \begin{cases} 0 & \text{Button unpushed} \\ 1 & \text{Button pushed} \end{cases}.$$

Cheese pressing example - Plant dynamics and control law

Activating solenoid S1 extends the cylinder, activating solenoid S2 retracts the cylinder.

Plant dynamics

		state	
$u_{1,k}$	$u_{2,k}$	x_k	x_{k+1}
0	0	0	0
0	1	0	0
1	0	0	1
(1)	(1)	0	undefined
0	0	1	1
0	1	1	0
1	0	1	1
(1)	(1)	1	undefined

Control law

x	u_c	u_1	u_2
0	0	0	0
0	1	1	0
1	0	0	1
1	1	0	1

Boolean algebra

$X, Y \in \{0, 1\}$

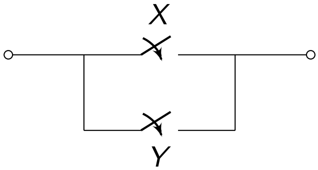
AND

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

Closed circuit $\Leftrightarrow$ 1
Open circuit $\Leftrightarrow$ 0

OR

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1



Boolean algebra, contd

$X, Y, Z \in \{0, 1\}$

	Property	Dual
Properties of 0 and 1	$X + 0 = X$	$X \cdot 0 = 0$
	$X + 1 = 1$	$X \cdot 1 = X$
Idempotent	$X + X = X$	$X \cdot X = X$
Complementarity	$X + \overline{X} = 1$	$X \cdot \overline{X} = 0$
Involution	$\overline{\overline{X}} = X$	
Commutative	$X + Y = Y + X$	$X \cdot Y = Y \cdot X$
Associative	$(X + Y) + Z = X + (Y + Z)$	$(XY)Z = Z(YZ)$
Distributive	$X \cdot (Y + Z) = XY + XZ$	$X + YZ = (X + Y)(X + Z)$

Boolean algebra, contd

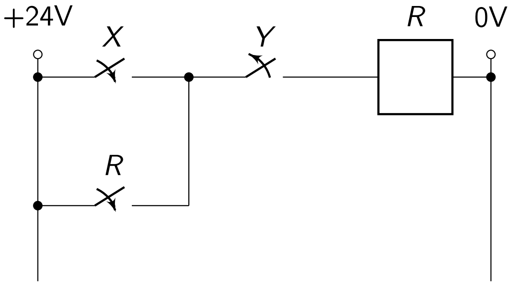
$X, Y, Z \in \{0, 1\}$

	Theorem	Dual
Absorption	$X + XY = X(1 + Y) = X$	$X(X + Y) = X$
Logic adjacency	$XY + X\overline{Y} = X(Y + \overline{Y}) = X$	$(X + Y)(X + \overline{Y}) = X$
De Morgan's	$\overline{X + Y} = \overline{X}\,\overline{Y}$	$\overline{XY} = \overline{X} + \overline{Y}$

An electrical circuit with memory

Latching circuit

LaTeX

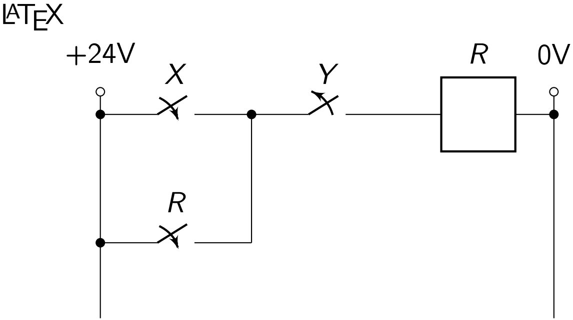


Truth table

X	Y	R_k	R_{k+1}
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

An electrical circuit with memory

Latching circuit



Truth table

X	Y	R_k	R_{k+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Cheese pressing example - Plant dynamics and control law revisited

Activating solenoid S1 extends the cylinder, activating solenoid S2 retracts the cylinder.

Plant dynamics

		state	
$u_{1,k}$	$u_{2,k}$	x_k	x_{k+1}
0	0	0	0
0	1	0	0
1	0	0	1
(1)	(1)	0	undefined
0	0	1	1
0	1	1	0
1	0	1	1
(1)	(1)	1	undefined

Control law

x	u_c	u_1	u_2
0	0	0	0
0	1	1	0
1	0	0	1
1	1	0	1

$$u_1 = \overline{x} u_c$$
$$u_2 = x\overline{u_c} + x u_c = x$$

$$\begin{aligned} x_{k+1} &= u_{1,k}\overline{u_{2,k}x_k} + \overline{u_{1,k}u_{2,k}}x_k + u_{1,k}\overline{u_{2,k}}x_k \\ &= \overline{u_{1,k}u_{2,k}}x_k + u_{1,k}\overline{u_{2,k}} \end{aligned}$$

Cheese pressing example - Control law

Solenoid S1

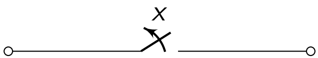
$$u_1 = \overline{x} u_c$$

LaTeX

Solenoid S2

$$u_2 = x$$

LaTeX

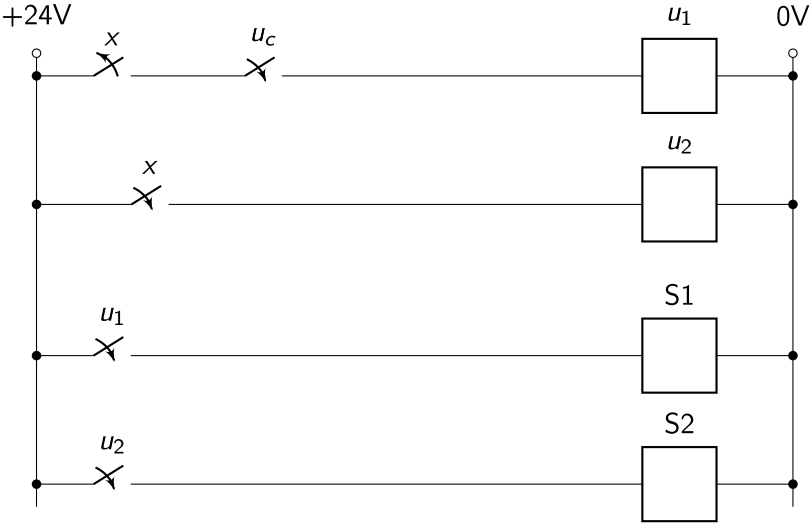


Cheese pressing example - Implementation of the control law

LaTeX

Cheese pressing example - Implementation of the control law, solution

LaTeX



Implementing the sequence A+B+B-A-

Implementing the sequence A+B+B-A-, control signal

Control signal

$$u = \begin{bmatrix} u_A+ & u_A- & u_B+ & u_B- \end{bmatrix}^T,$$

with

$$u_A+ = \begin{cases} 0 & \text{Solenoid extending A is not activated} \\ 1 & \text{Solenoid extending A is activated} \end{cases}$$

$$u_A- = \begin{cases} 0 & \text{Solenoid retracting A is not activated} \\ 1 & \text{Solenoid retracting A is activated} \end{cases}$$

Similar for B.

Implementing the sequence A+B+B-A-, state variables

State variables (naive)

$$x = \begin{bmatrix} x_A & x_B \end{bmatrix}^T,$$

with

$$x_{\{A,B\}} = \begin{cases} 0 & \text{Cylinder \{A,B\} retracted} \\ 1 & \text{Cylinder \{A,B\} extended} \end{cases}$$

Implementing the sequence A+B+B-A-, control law

Control law (problematic)

Ignoring input signal u_c. Movement should be cyclic

x_A	x_B	u_A+	u_A-	u_B+	u_B-
0	0				
0	1				
1	0				
1	1				

Implementing the sequence A+B+B-A-, control law

Control law (problematic)

Ignoring input signal u_c. Movement should be cyclic

x_A	x_B	u_A+	u_A-	u_B+	u_B-
0	0	1	0	0	0
(0)	(1)	0	0	0	1
1	0	0	1 or 0	1 or 0	0
1	1	0	0	0	1

Implementing the sequence A+B+B-A-, state variables

State variables (better)

$$x = \begin{bmatrix} x_A & x_B & x_P \end{bmatrix}^T,$$

with

$$x_{\{A,B\}} = \begin{cases} 0 & \text{Cylinder \{A,B\} retracted} \\ 1 & \text{Cylinder \{A,B\} extended} \end{cases}$$

$$x_P = \begin{cases} 0 & \text{Cheese not yet pressed} \\ 1 & \text{Cheese pressed} \end{cases}$$

State transitions

Implementing the sequence A+B+B-A-, control law

State transitions

Control law (better)

x_A	x_B	x_P	u_A+	u_A-	u_B+	u_B-
0	0	0				
1	0	0				
1	0	1				
1	1	1				

Implementing the sequence A+B+B-A-, control law

State transitions

Control law (better)

x_A	x_B	x_P	u_A+	u_A-	u_B+	u_B-
0	0	0	1	0	0	0
1	0	0	0	0	1	0
1	0	1	0	1	0	0
1	1	1	0	0	0	1

Implementing the sequence A+B+B-A-, control law

State transitions

Control law (better)

x_A	x_B	x_P	u_A+	u_A-	u_B+	u_B-
0	0	0	1	0	0	0
1	0	0	0	0	1	0
1	0	1	0	1	0	0
1	1	1	0	0	0	1

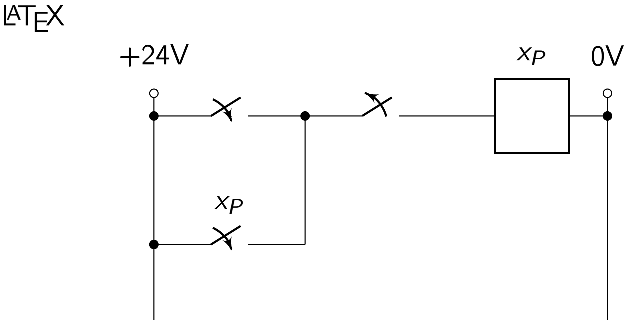
$$u_A+ = \overline{x_A}$$

$$u_A- = x_A\overline{x_B}x_P$$

$$u_B+ = x_A\overline{x_B x_P}$$

$$u_B- = x_B$$

Implementing the sequence A+B+B-A-, latching circuit for x_P



Implement the circuit!

For the report

- Truth table for the control law
- Control law as boolean expression
- Circuit diagram for the controller
- Screen shot and short video showing working solution in FluidSim
- Short video showing working solution in hardware