noisR

Análise e Design de Circuitos: Projeto somador de bits

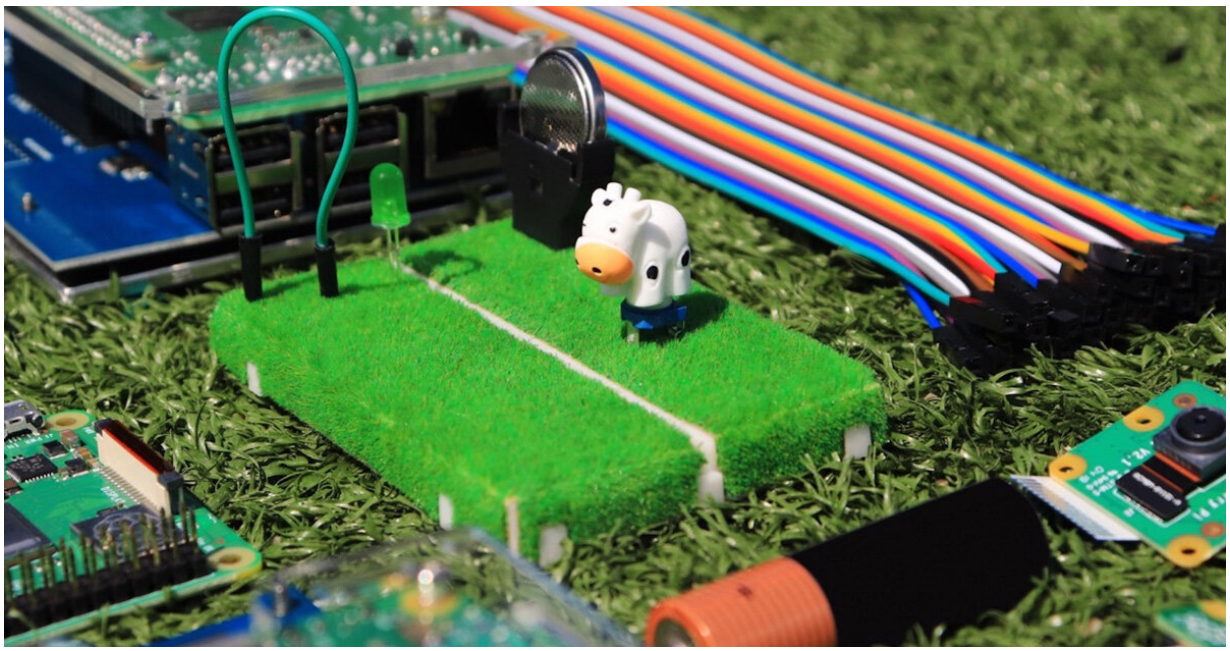

Fig. 1: Uma vaca feliz que te deseja boa sorte.

Departamento de Física
NoisR

2023

Grupos

WORKSHOPS 3, 4 + PROJETO 2			
GRUPO 1	**GRUPO 2**	**GRUPO 3**	**GRUPO 4**
Diogo Carvalho Gonçalo Girante Tomás Abreu	Gabriel Franco Beatriz Carvalho Gonçalo Andrade	Francisco Silva Beatriz Lima Bernardo Rodrigues	Ana Armada Apolo Gomes Eduardo Braz
GRUPO 5	**GRUPO 6**	**GRUPO 7**	
Miguel Moreira Simão Crispim Martim Leitão	Ricardo Ribeiro Guilherme Gonçalvez João Amaral	Rodrigo Silva Filipa Santiago Bernardo Monteiro	

Material

Componente	**#**	**Componente**	**#**	**Componente**	**#**	**Componente**	**#**
Pilha 9V	1	Díodo	3	Res 390R	3	Res 390R	3
Suporte Pilha	2	LED	8	Res 15R	3	Res 1MR	2
Fio	6	Botão	4	Res 1.2kR	3	Switch-2	1
Breaboard	1	Transistor	10	Res 470R	3	Terminal-2	1
Código Ohm	1	Regulador 5V	1	Res 330R	3	Rebuçados	3

Disclaimer: Se precisarem de mais material (provavelmente vai acontecer), falem connosco!

Introdução

Neste projeto vais aprender a construir um bloco **Half-Adder**. Um Half-Adder (descrito na Figura 2) é usado para somar 2 bits de entrada (A, B) e gerar dois bits de saída (a soma S e um carry-out C_{out}). O carry-out é o "vai um" das contas aritméticas que estamos habituados a fazer. Este bloco é uma pequena calculadora que soma números binários de forma semelhante ao que o processador do nosso computador faz.

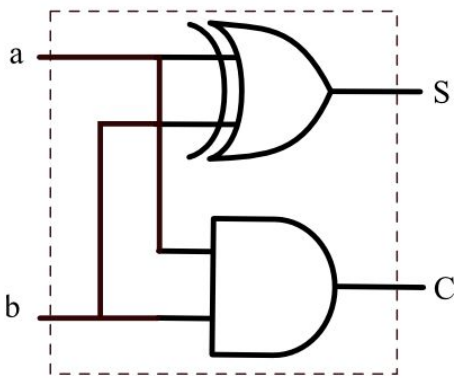

Fig. 2: Bloco Half-Adder

Para a criação deste bloco, terás de utilizar duas portas lógicas:

- Porta AND - que já sabes fazer do exercício 3 da primeira aula
- Porta XOR - composta por um OR e um NAND, que explico mais à frente

Terás de as implementar com transistors de forma semelhante ao que aprendeste na aula. Vamos neste enunciado acompanhar-vos na construção desse bloco Half-Adder.

Estas implementações vão ser feitas numa breadboard.

1 Montagem de portas lógicas

1.1 Anatomia de uma breadboard

Uma breadboard consiste numa placa de plástico com pequenos buracos onde é possível inserir componentes eletrónicos para prototipagem de circuitos. As conexões dos componentes à placa não são permanentes, pelo que é sempre possível removê-los caso o circuito implementado não tenha o comportamento desejado. Na Figura 3, apresenta-se o esquema de uma breadboard.

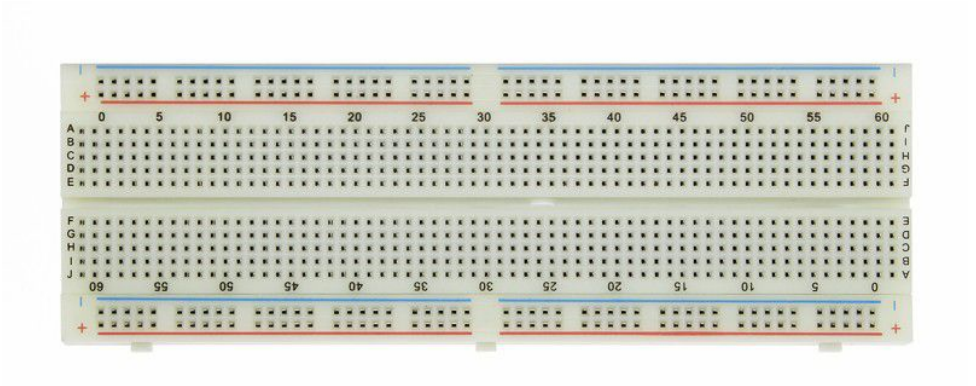

Fig. 3: Breadboard para prototipagem de circuitos

Os componentes eletrónicos (e.g., transístores, condensadores, resistências) têm pinos de metal que podem ser colocados nos buracos da breadboard. No interior dos buracos, existem filas de clips metálicos que prendem os pinos dos componentes, garantindo assim que não se soltam. As letras e números da breadboard permitem identificar buracos individuais da placa (e.g., o buraco C12 está na interseção entre a coluna C e a fila 12).

Quando se monta um circuito na breadboard, é necessário dar-lhe energia. Na placa, existem dois conjuntos de duas colunas com uma linha vermelha e uma linha azul onde é feita a alimentação da placa. Normalemente, na coluna da linha vermelha, liga-se a alimentação (+) e na coluna da linha azul liga-se o ground (-). Em circuitos eletrónicos, os valores típicos da alimentação e do ground são 5 V e 0 V, respetivamente, mas existem componentes que necessitam de mais energia (e.g., amplificadores operacionais).

Na Figura 4, mostra-se como os buracos da breadboard estão ligados. **Quando se coloca uma tensão num buraco da breadboard, os restantes buracos ligados a este encontram-se à**

mesma tensão. Para efetuar ligações entre componentes ao longo da breadboard, usam-se jumpers (fios com um pino de metal que pode ser colocado nos buracos da placa), ou ligam-se, quando possível, os terminais dos próprios componentes - reduzindo a poluição visual da nossa montagem.

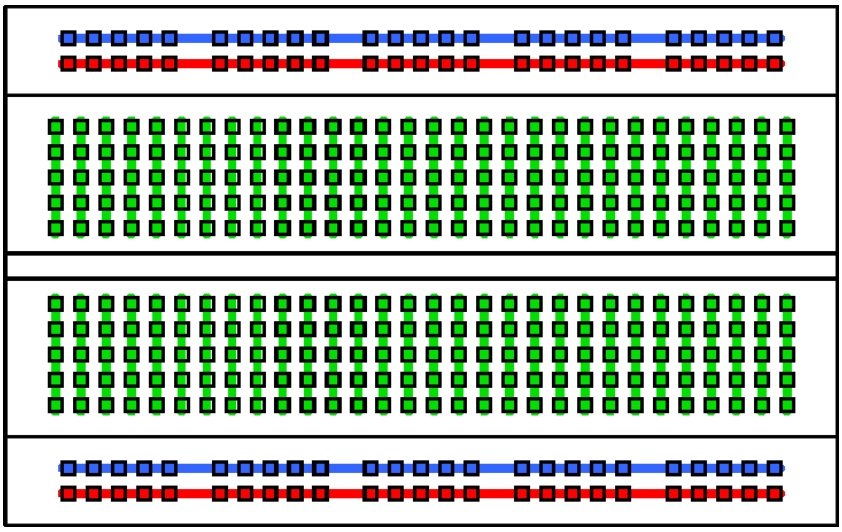

Fig. 4: Conexões numa breadboard

Agora que sabes como funciona uma breadboard, podes começar a implementar o teu primeiro circuito.

1.2 HIGH's E LOW's no nosso circuito

É diferente desenhar um **diagrama lógico** e um **esquema elétrico**.

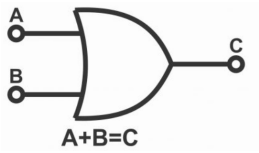

Fig. 5: Diagrama lógico de um OR

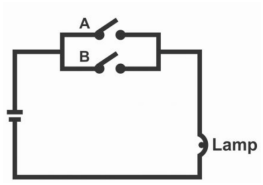

Fig. 6: Esquema elétrico de um OR

- Um **diagrama lógico** representa um bloco que produz uma operação lógica. Essa operação lógica tem um resultado lógico: 1 ou 0. Posso também chamar-lhes HIGH (H) ou LOW (L). Ou TRUE (T) ou False (F). Independentemente de como os chame, significam a mesma coisa para efeitos do nosso projeto.

- um **esquema elétrico** é o circuito com os respetivos componentes, com valores de tensão, resistência e de corrente que já sabemos calcular. Podemos definir um diagrama lógico (que produz "1"/"0") pode ter um correspondente circuito elétrico (que produz $5V/0V$).

No nosso projeto vamos considerar "1" uma tensão maior do que $2V$ e "0" uma tensão $\approx 0V$. Para ser de visualização mais fácil, **no nosso projeto vamos utilizar um LED ligado/desligado para servir de 1/0 respetivamente.**

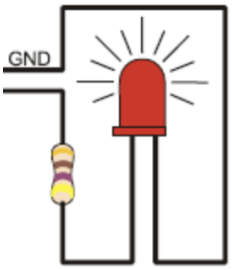

Fig. 7: LED a "1" (HIGH)

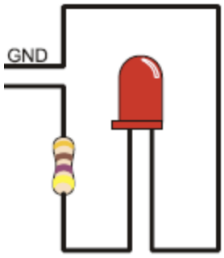

Fig. 8: LED a "0" (LOW)

1.3 Montagem da Porta AND

Dados dois inputs, A e B, e uma saída Y, a porta AND resulta em "1" **se e só se** ambas as entradas forem "1"; os demais resultados são "0". Essa operação é analiticamente descrita por $Y = A \cdot B$.

Em circuitos lógicos, a porta AND é representada pelo símbolo que se apresenta na Figura 10. Já no esquema elétrico, a porta AND pode ser alcançada pelo esquemático desenhado no **quarto exercício** dos problemas da aula.

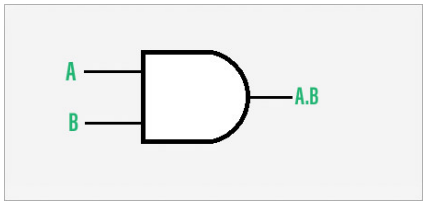

Fig. 9: Porta lógica AND

A	B	A AND B
0	0	
0	1	
1	0	
1	1	

Tab. 1: Tabela de verdade da porta AND

.1. (T) Preenche a tabela de verdade para uma porta lógica AND com 2 bits.

.2. (E) Ocupando o mínimo de espaço na breadboard, faz uma porta lógica AND. Utiliza transístores como switches, conforme mostrado na figura baixo. Deixa fios soltos em A e B para poderes curto-circuitá-los mais tarde com nós específicos do circuito para servir de switch.

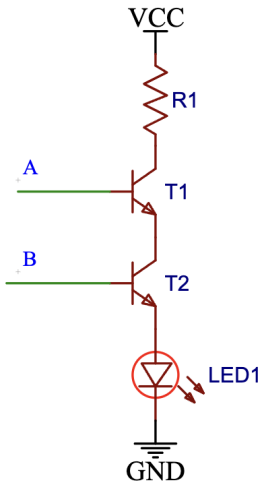

Fig. 10: Circuito da lógica AND

.3. (E) Curto-circuita (liga) os fios soltos nas bases dos transístors a cada um dos nós. No fundo, o que estás a fazer é a impor a tensão na base dos transístores conforme onde ligas esses fios. Se essa tensão for superior a $2V$, terás a passagem de corrente por entre o Coletor e o Emissor do respetivo transístor.

- **Para obteres "1"**: circuito-circuita ambos os fios soltos ao nó entre R_1 e o coletor do T1 (HIGH)
- **Para obteres "0"**: basta curto-circuitares um dos fios da base de um dos transístores ao ground (LOW)

Verifica que o LED acende e desliga conforme a tabela de verdade que escreveste. Se o LED não estiver a acender verifica a montagem, a resistência, a polaridade do LED, ou se o mesmo está fundido.

.4. (E) Melhora o circuito introduzindo dois botões nas bases dos transístors, em vez de ter fios soltos, e ligando-os de forma a que sirvam de ON (quando pressionado) e OFF (quando não pressionado). *Dica: é só deixar passar corrente ou não para a base do transístor.*

1.4 Montagem da Porta XOR

A porta lógica XOR (também conhecida como OR exclusivo, com o símbolo $\oplus$) é mais complexa que as portas lógicas AND e OR. O output do XOR de 2 bits é "1" quando **apenas um** dos bits de entrada é "1". No diagrama lógico, a porta XOR é representada pelo símbolo que se apresenta na Figura 11. A correspondente tabela de verdade está também descrita.

Fig. 11: Porta lógica XOR

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Tab. 2: Tabela de verdade da porta XOR

A implementação desta porta lógica pode ser feita com portas AND, OR, e a porta NAND (a negação da porta AND). A ligação das portas em si é feita de acordo com a Figura 12.

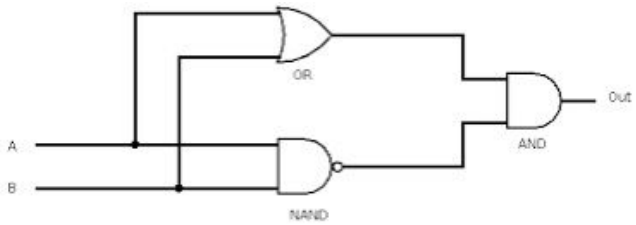

Fig. 12: Circuito lógico da porta lógica XOR

Ou seja, para criarmos esta porta XOR precisamos de outras três portas lógicas ligadas, e o seu output é que é o output do XOR.

Já montaste a porta AND na breadboard. Agora é altura de montares a porta OR, que para não tornar tudo demasiado fácil vamos descrever apenas brevemente[1].

1.4.1 Montagem da Porta OR

Dados dois inputs, A e B, e uma saída Y, a porta OR efetua a operação $Y = A + B$. O output do OR é "1" se alguma das entradas for "1". Em circuitos lógicos, a porta OR é representada pelo símbolo que se apresenta na Figura 13.

Fig. 13: Porta lógica OR

A	B	A OR B
0	0	
0	1	
1	0	
1	1	

Tab. 3: Tabela de verdade da porta OR

.1. (T) Preenche a tabela de verdade para uma porta lógica OR com 2 bits.

.2. (T) Desenha o que seria um circuito elétrico da porta OR com transístors antes de montares na breaboard. **Dica**: a partir do esquemático para a porta AND, pensa nas ligações que necessitarias de mudar para implementar a porta OR.

.3. (E) Ocupando o mínimo de espaço possível na breadboard, monta a porta lógica OR que desenhaste. Verifica, usando LED's para ver os sinais High (1) e Low (0), que a porta lógica se

[1] Na verdade, se leste este enunciado todo até aqui já sabes montar a porta OR :)

comporta de acordo com a tabela de verdade preenchida anteriormente.

1.4.2 Porta lógica NAND

Falta-nos a porta lógica NAND (NOT AND) para concretizar o XOR necessário para o Half-Adder. O NAND é a **negação** da porta lógica AND. Ou seja, o output do NAND é "1" sempre que o output do AND é "0", e "0" sempre que o output do AND é "1".

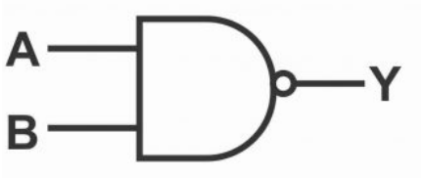

Fig. 14: Porta lógica NAND

A	B	A NAND B
0	0	
0	1	
1	0	
1	1	

Tab. 4: Tabela de verdade da porta NAND

DESCRIÇÃO DA MONTAGEM NAND: Recorrendo ao esquemático da construção que fizeste na aula para o NOT (exercício 3) percebes que a porta NOT, misturada com a porta AND, consiste em apenas trocar a posição do LED no AND que já fizeste para o colocar no nó entre R_1 e o coletor do transístor T1 (como ficaria numa porta NOT). Claro está, tem em atenção à polaridade do LED.

.1. (T) Preenche a tabela de verdade para uma porta lógica NAND com 2 bits.

.2. (T) Desenha o esquemático do NAND, de acordo com a descrição acima.

.3. (E) Ocupando o mínimo de espaço possível na breadboard, faz uma porta lógica NAND. Verifica no LED que obténs a tabela de verdade para essa porta lógica.

2 Construção do Half-Adder

Agora que tens as portas lógicas implementadas, tens tudo o que necessitas para implementar o bloco Half-Adder. No entanto, para percebermos onde queremos chegar com o nosso somador de bits, importa percebermos... como se soma bits.

2.1 Soma binária

Um bit (*binary digit*) é um placeholder para apenas dois estados possíveis. Numeração binária é um sistema numérico ponderado cujo alfabeto é $\{0, 1\}$, tendo por isso palavras de bits. Binário é a base do processamento das unidades lógicas e aritméticas de um microprocessador, onde dados são armazendados em memórias e processados no formato de palavras binárias.

Como o sistema numérico é ponderado, a posição de cada bit num número binário representa uma potência de 2. Por exemplo, em $1101_{(2)}$, o dígito mais à direita tem o peso 2^0, o próximo (à sua esquerda) tem peso 2^1, o próximo 2^2, e o mais à esquerda 2^3. Somando os valores de acordo com o seu peso relativo temos

$$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 \times 1 \times 2^0 = 13$$

A soma entre números binários é semelhante à soma entre números decimais, mas para um placeholder que só suporta 2 dígitos (0, 1) em vez de 10. Para somar $17 + 3$ faríamos a soma individualmente por dígitos de igual peso. Ou seja

$$\begin{aligned} 17 + 3 &= 1 \times 10^1 + 7 \times 10^0 + 3 \times 10^0 \\ &= 1 \times 10^1 + (7 + 3) \times 10^0 \\ &= 1 \times 10^1 + 10 \times 10^0 \\ &= 1 \times 10^1 + 1 \times 10^1 + 0 \times 10^0 \\ &= (1 + 1) \times 10^1 + 0 \times 10^0 \\ &= 2 \times 10^1 + 0 \times 10^0 \\ &= 20 \end{aligned}$$

Tudo o que aconteceu nestes passos foi a transmissão de um *carry* para o *placeholder* de peso acima.

Em binário não é diferente:

	1	1	0	1
+	1	0	1	1
1	1	0	1	0
	4 bits			

Tab. 5: Soma binária com *carry*'s e overflow

Começa-se da coluna de menor peso. $1 + 1$ em binário é 0, com carry de 1. No placeholder de peso acima temos $1_{carry} + 1 + 0$, que resulta em 0 com carry 1. No terceiro peso, de 2^2, tem-se $1_{carry} + 1 + 1$, resultando em 1, com carry 1. Finalmente, na quarta coluna, temos $1_{carry} + 0 + 0$, resultando em 1, sem carry. O resultado final é $11000_{(2)}$ em binário. Notar que foi preciso adicionar um placeholder para podermos representar o número binário final.

.1. (T) Representa os números $101101_{(2)}$ e $1101110_{(2)}$ em decimal.

.2. (T) Soma dois números binários diretamente com as operações de soma binária: $101101_{(2)}$ + $1101110_{(2)}$. Confirma o resultado convertendo o número binário em decimal.

2.2 Construção do Half-Adder

.1. (T) Com base no conhecimento das operações da soma binária já podemos modelar um **Half-Adder** de um bit. Isto é, a soma de dois bits A, B que resulta em S e num carry C_{out}. Preenche a seguinte tabela de verdade:

A	B	C_{out}	S
0	0		
0	1		
1	0		
1	1		

Tab. 6: Soma de 2 bits

.2. (E) Em princípio, por esta altura, já deves ter montadas todas as portas lógicas necessárias para a construção do teu Half-Adder. O Diagrama Lógico do half-adder está definido na figura 2. Caso já tenhas as portas construídas, já fizeste este ponto. Yey!

.3. (E) Testa o teu Half-Adder. Faz um pequeno vídeo dele a funcionar e envia aos teus amigos, família e **principalmente** ao grupo dos Padawans para exibires o teu trabalho.

2.3 Full-Adder (EXTRA)

Se já fizeste tudo até aqui, já fizeste o projeto que te era pedido. Esta secção é apenas um extra para aprimorares o teu somador de bits. Repito, **esta parte é opcional, por isso não faço ideia porque é que ainda estás sequer a ler isto em vez de estares a estudar Cálculo.**

Em circuitos mais complexos, vários blocos somadores são ligados em cascata para realizar a soma de duas (e mais) palavras binárias, pelo que é necessário considerar, além do C_{out}, também o bit de *carry-in*, C_{in} que entrará na conta do placeholder seguinte.

O Full-Adder recebe 3 bits de entrada em vez de dois, sendo o terceiro bit o C_{in}. Lá está, a introdução deste terceiro input permite-nos colocar vários Full-Adder por sua vez em cascata e somar palavras de 3, 4, 5, N bits em vez de apenas 2. O bloco Full-Adder pode ser separado em dois blocos Half-Adder com uma porta lógica OR. A ideia desta secção é, para todos os que fizeram o bloco full-adder, poderem, no fim, criar um somador de N bits, sendo N o número de grupos que fizeram este extra em vez de estarem a estudar Cálculo.

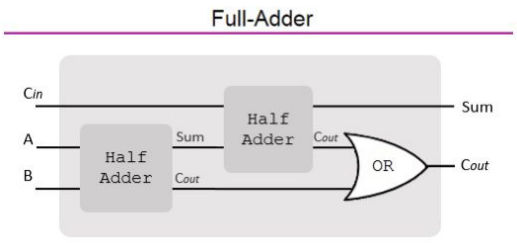

Fig. 15: Circuito em cascata do bloco Full-Adder

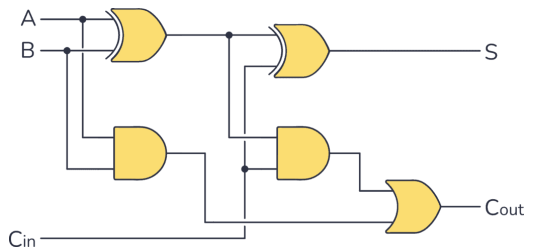

Fig. 16: Circuito lógico do bloco Full-Adder

.1. (E) Cria outro Half-Added na breadboard. Se precisarem de mais componentes (nomeadamente transístors) falem connosco.

.2. (T) Preenche a seguinte tabela de verdade, agora para uma soma com 2 bits **com carry-in**.

C_{in}	A	B	C_{out}	S
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

.3. (E) Efetua as ligações necessárias entre as portas lógicas para implementar o bloco Full-Adder. Verifica, usando LED's para ver os sinais High (1) e Low (0), que o bloco Full-Adder funciona corretamente.

.4. (E) Celebrar o fim deste capítulo da vossa vida.

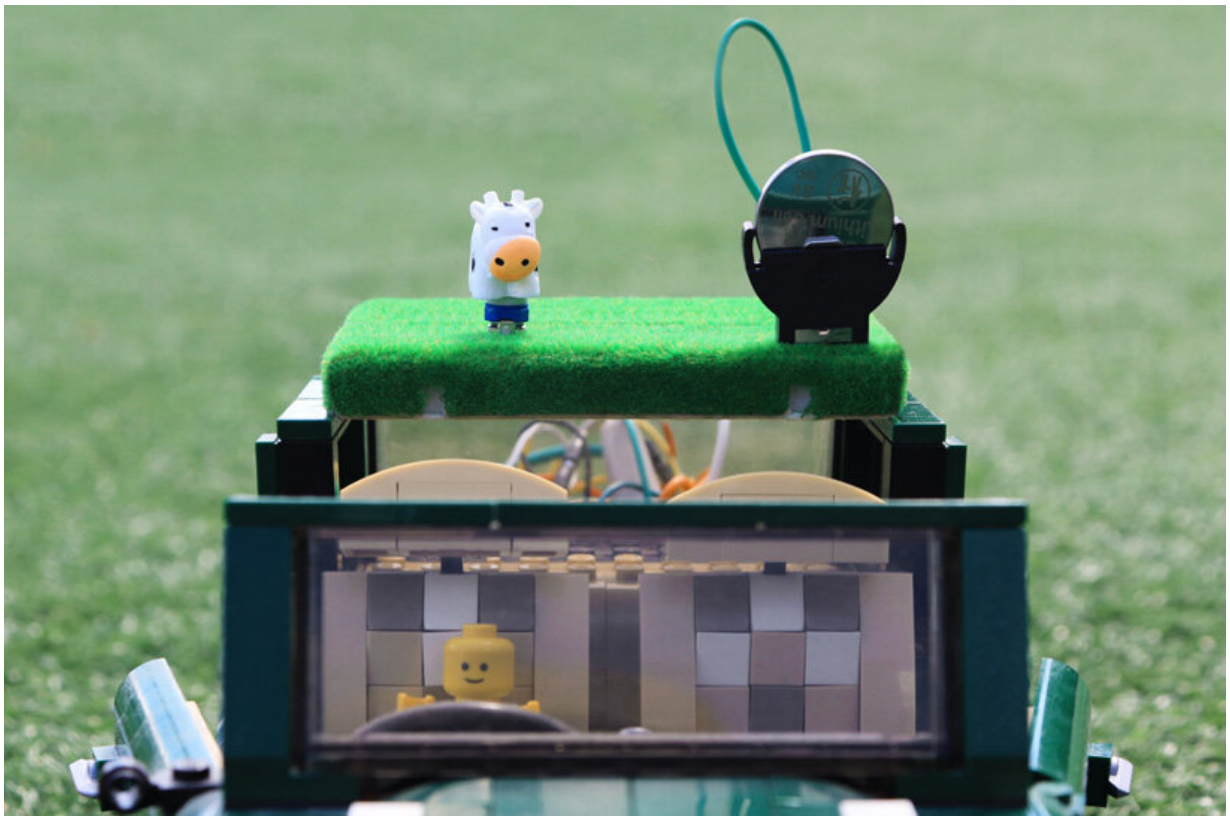

Fig. 17: Uma vaca feliz a vir dar-te os parabéns pelo teu trabalho.