Codificação de Sinais

Alunos: Pedro Victor e Renan Cunha

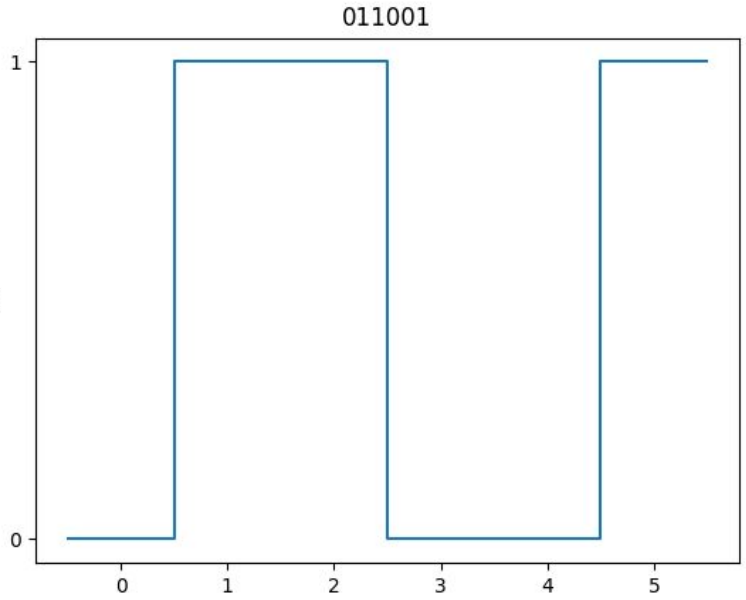
Agenda

- Ferramentas
- Funções Auxiliares
- Manchester
- Differential Manchester
- B8ZS
- HDB3

Ferramentas

Ferramentas

```
import matplotlib.pyplot as plt
x = [0,1,2,3,4,5]
y = [0,1,1,0,0,1]
plt.step(x,y)
```

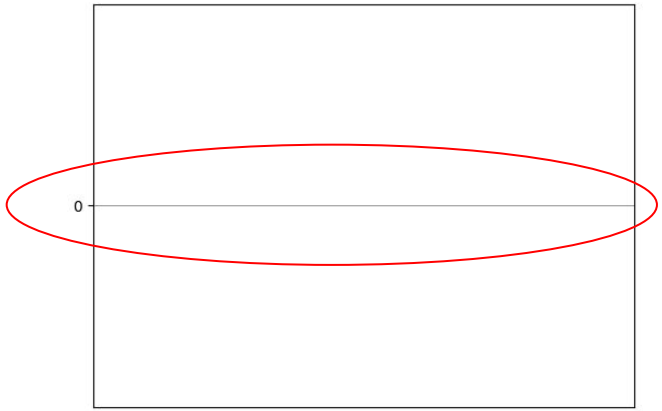


Funções Auxiliares

```
def bits_to_int_list(bits: str) -> List[int]:
    """
    Ex:
        ‘1010100’ -> [1, 0, 1, 0, 1, 0, 0]
    """
    bits = list(map(int, bits))
    return bits
```

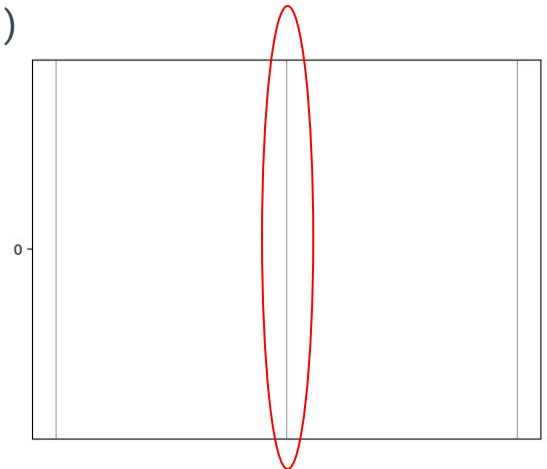
Funções Auxiliares

```
import matplotlib.pyplot as plt

def make_graph(signal: List[int], bits: List[int], title: str) -> None:

    features = {'color': 'gray', 'linewidth': 0.4}
    # horizontal line
    plt.axhline(y=0, **features)

    ...
```



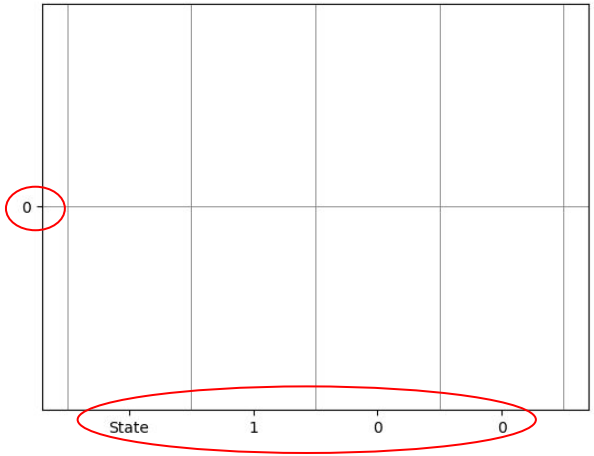
Funções Auxiliares

```
import matplotlib.pyplot as plt

def make_graph(signal: List[int], bits: List[int], title: str) -> None:

    ...
    # vertical lines
    bits = ["State"] + bits
    ratio = len(signal) // len(bits)
    lines_coordinates = list(range(0, len(signal), ratio))
    for x_pos in lines_coordinates:
        plt.axvline(x=x_pos, **features)

    ...
```



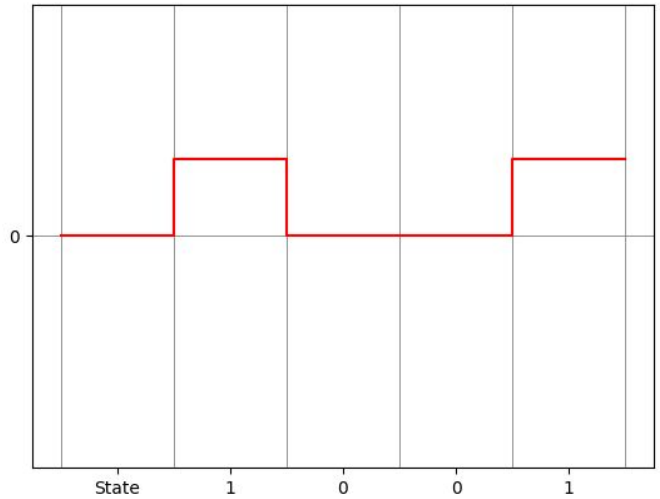
Funções Auxiliares

```
import matplotlib.pyplot as plt

def make_graph(signal: List[int], bits: List[int], title: str) -> None:

    ...
    # axis
    plt.xticks(vertical_lines_coordinates[:-1], bits)
    plt.yticks([0], [0])

    ...
```



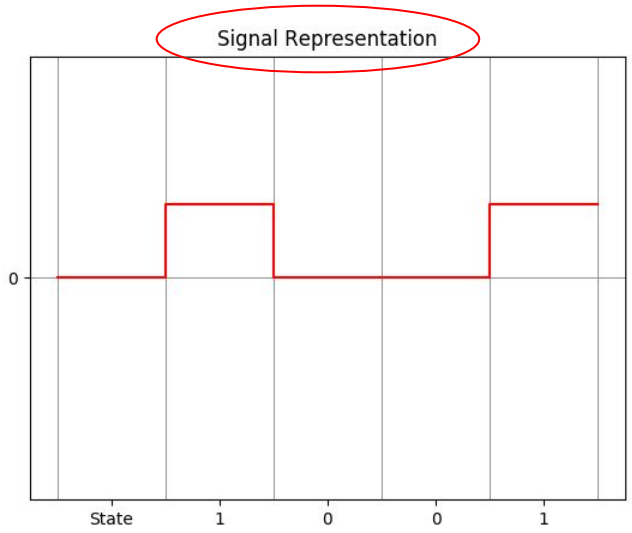
Funções Auxiliares

```
import matplotlib.pyplot as plt

def make_graph(signal: List[int], bits: List[int], title: str) -> None:

    ...
    # plot
    x_coordinates = list(range(len(signal)))
    plt.step(x_coordinates, signal, color='red')

    ...
```

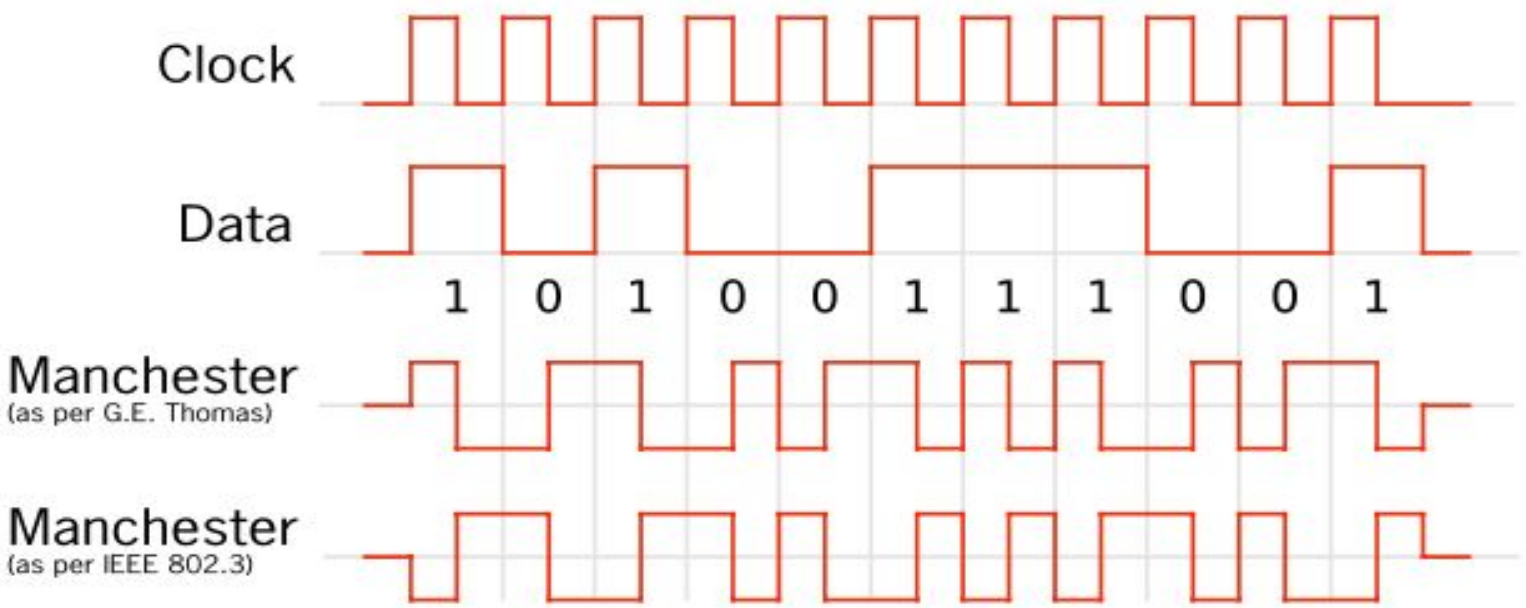


Funções Auxiliares

```
import matplotlib.pyplot as plt

def make_graph(signal: List[int], bits: List[int], title: str) -> None:

    ...
    plt.title(title)

    plt.show()
```

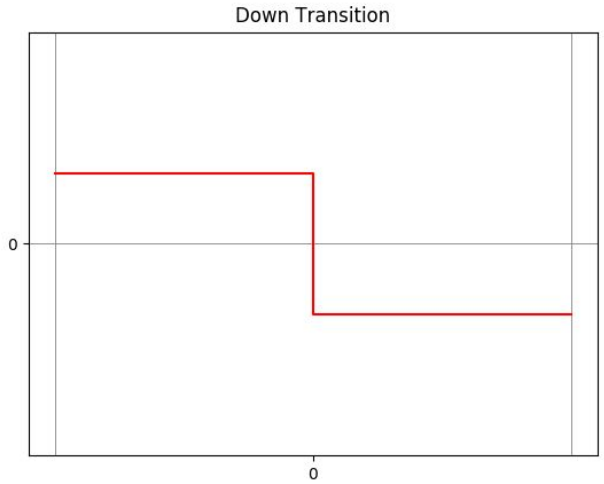


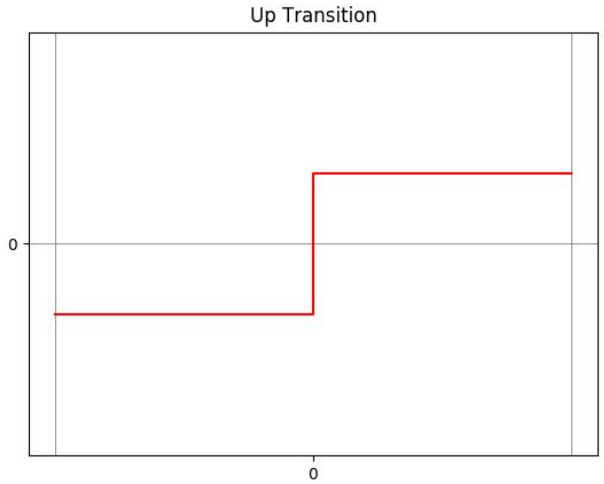
Manchester


Clock
Data
1 0 1 0 0 1 1 1 0 0 1
Manchester
(as per G.E. Thomas)
Manchester
(as per IEEE 802.3)

Manchester

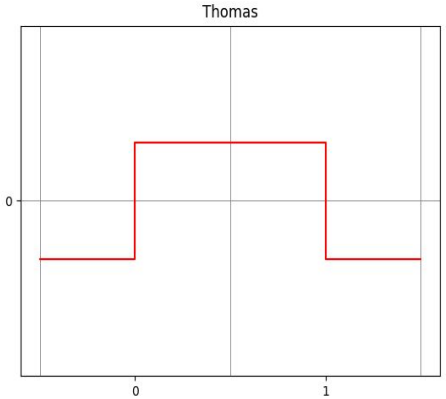
```
def manchester(bits: str, convention: str = "thomas") -> None:
    bits = bits_to_int_list(bits)
    down_transition, up_transition = [1, -1], [-1, 1]
    ...
```

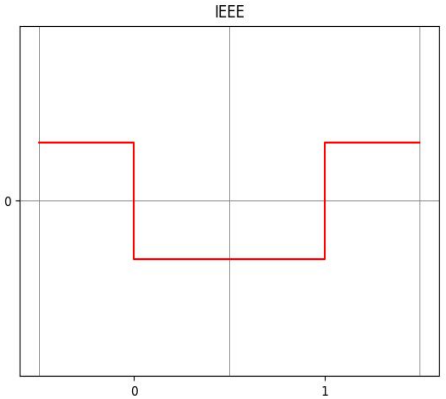




Manchester

```
def manchester(bits: str, convention: str = "thomas") -> None:
    ...
    if convention == 'thomas':
        zero, one = up_transition, down_transition
    elif convention == "ieee":
        zero, one = down_transition, up_transition
    ...
```

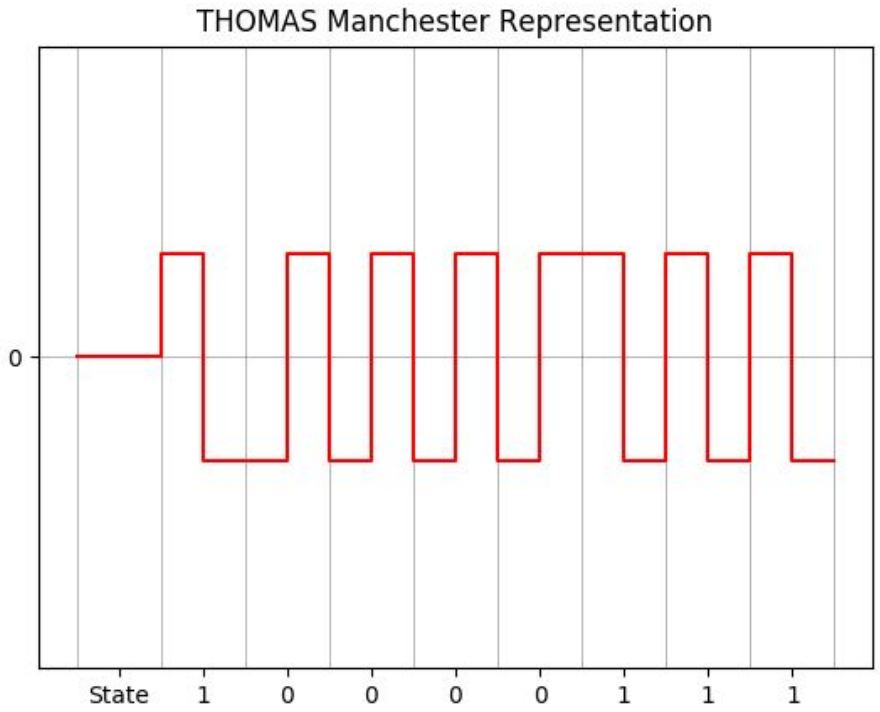




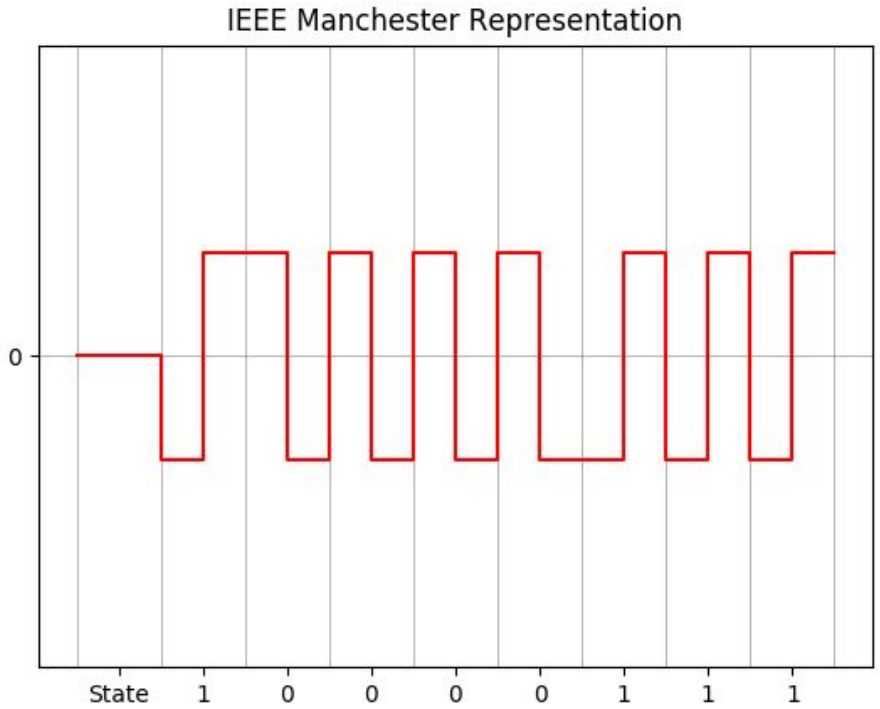
Manchester

```
def manchester(bits: str, convention: str = "thomas") -> None:
    ...
    signal = []
    for bit in bits:
        if bit == 1:
            signal += one
        elif bit == 0:
            signal += zero

    signal = [0]*3 + signal
    title = f"{convention.upper()} Manchester Representation"
    make_graph(signal, bits, title)
```

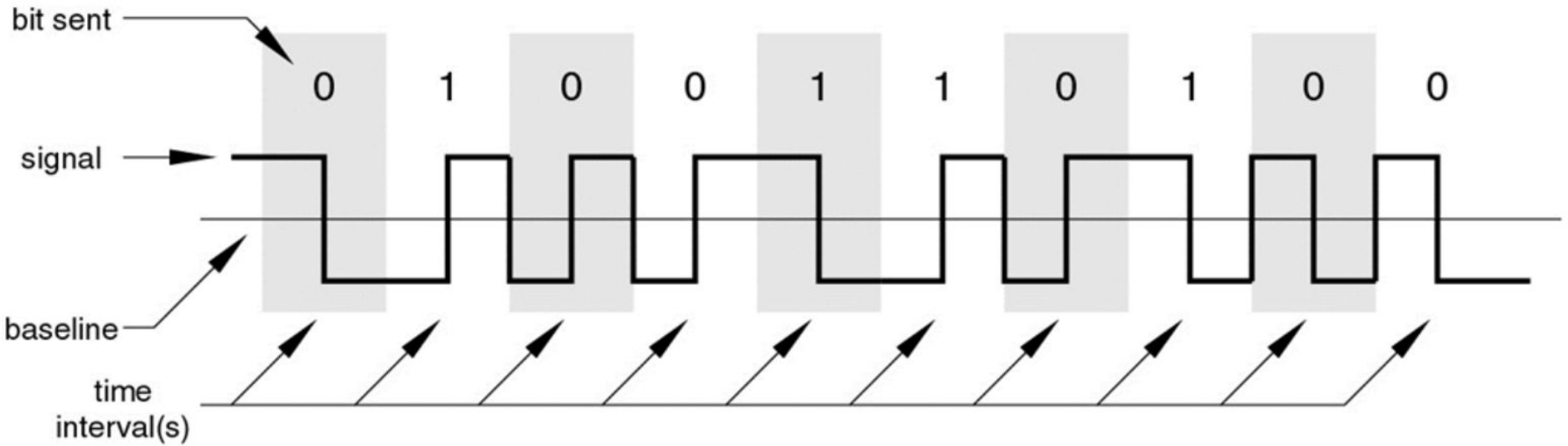
Thomas Manchester - Exemplo



IEEE Manchester - Exemplo



Differential Manchester

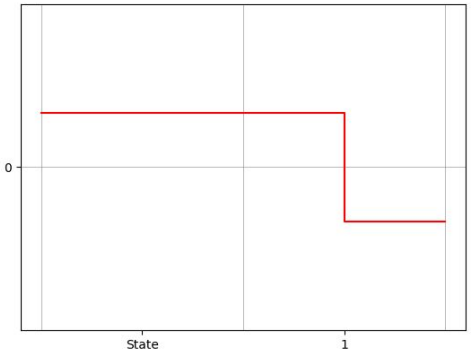


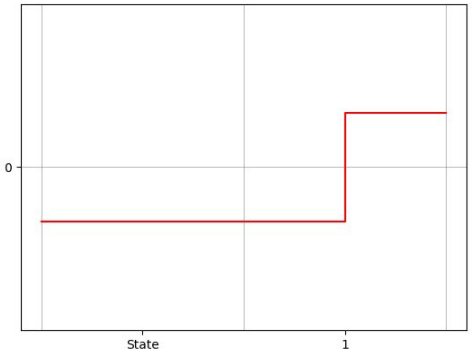
Differential Manchester

```
def d_manchester(bits: str, initial_phase: int) -> None:
    bits = bits_to_int_list(bits)

    signal = []
    phase = initial_phase
    ...
```

Differential Manchester

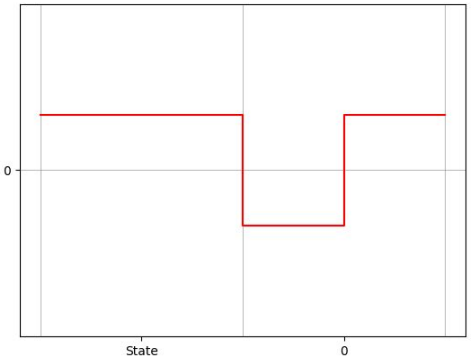
```
def d_manchester(bits: str, initial_phase: int) -> None:
    bits = bits_to_int_list(bits)

    signal = []
    phase = initial_phase

    for bit in bits:
        if bit == 1:
            signal.append(phase)
            phase *= -1
            signal.append(phase)
        elif bit == 0:
            signal.append(phase * -1)
            signal.append(phase)
    ...
```

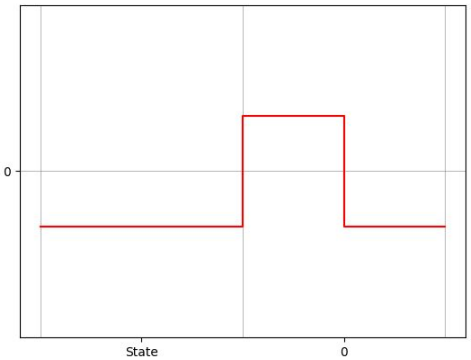




Differential Manchester

```
def d_manchester(bits: str, initial_phase: int) -> None:
    bits = bits_to_int_list(bits)

    signal = []
    phase = initial_phase

    for bit in bits:
        if bit == 1:
            signal.append(phase)
            phase *= -1
            signal.append(phase)
        elif bit == 0:
            signal.append(phase * -1)
            signal.append(phase)
    ...
```

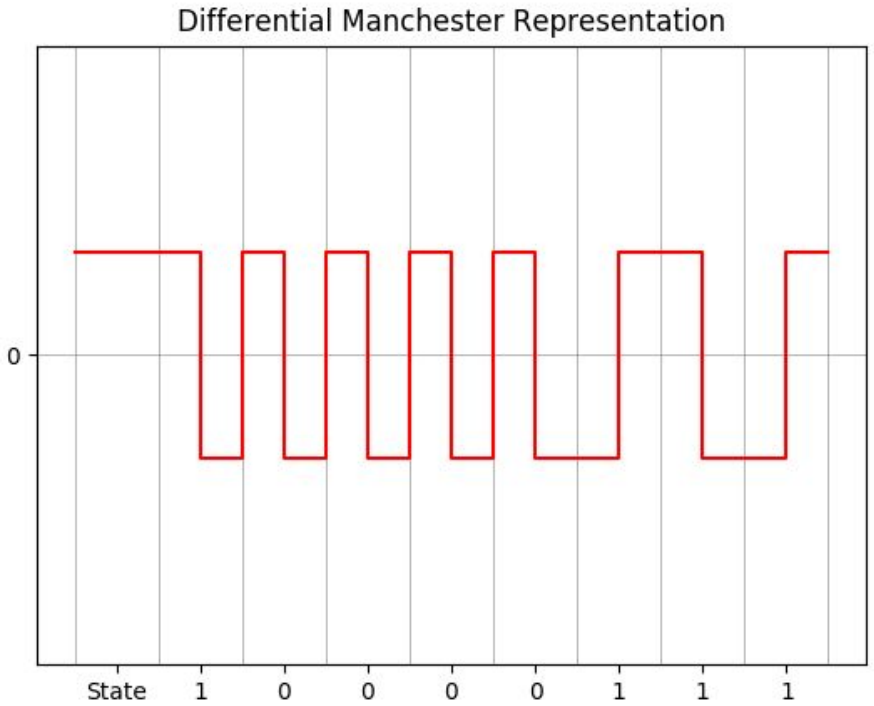




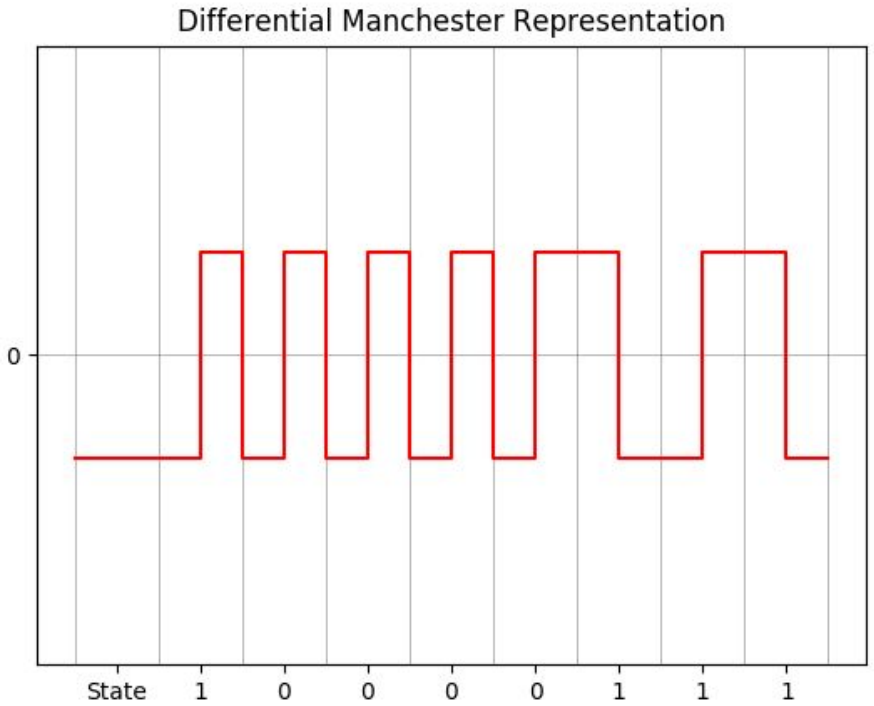
Differential Manchester

```
def d_manchester(bits: str, initial_phase: int) -> None:

    ...
    signal = [initial_phase]*3 + signal

    title = "Differential Manchester Representation"
    make_graph(signal, bits, title)
```

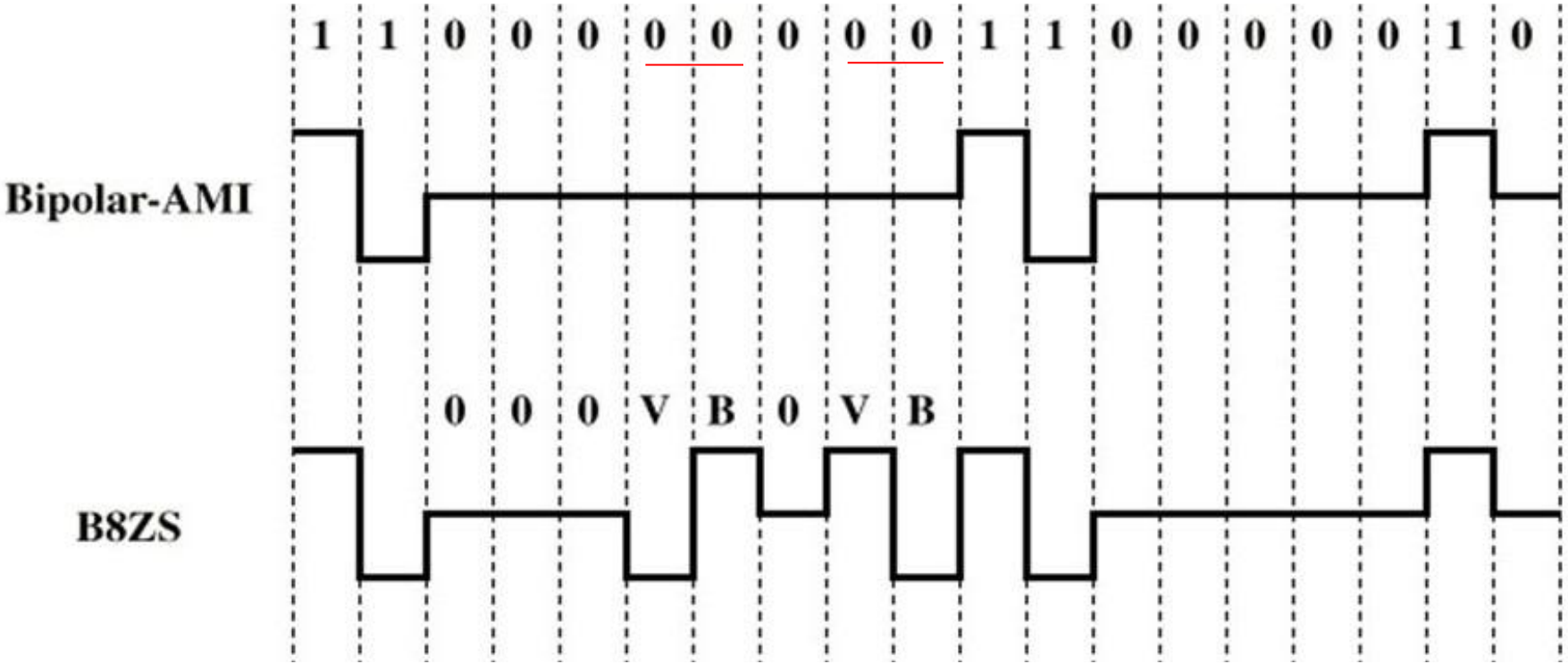
Differential Manchester - Exemplo



Differential Manchester - Exemplo



Bipolar 8-Zero Substitution (B8ZS)

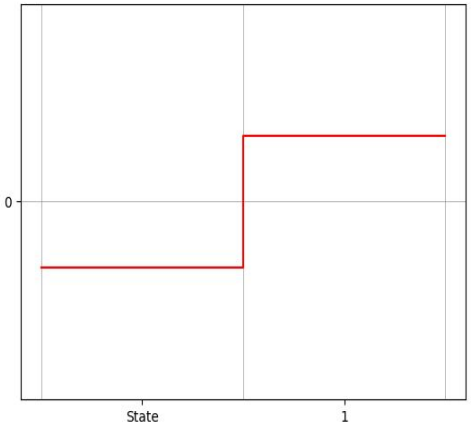


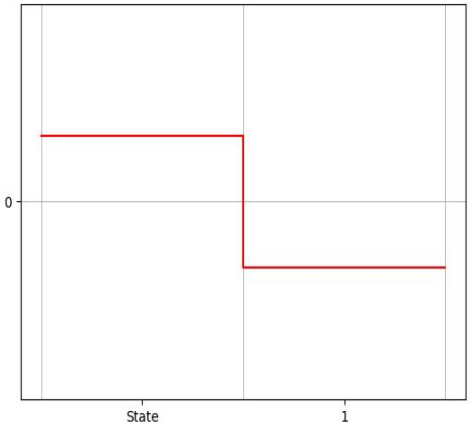
Bipolar 8-Zero Substitution (B8ZS)

```
def b8zs(bits: str, initial_phase: int) -> None:
    bits = bits_to_int_list(bits)

    signal = []
    phase = initial_phase

    count_zero = 0
    ...
```

Bipolar 8-Zero Substitution (B8ZS)

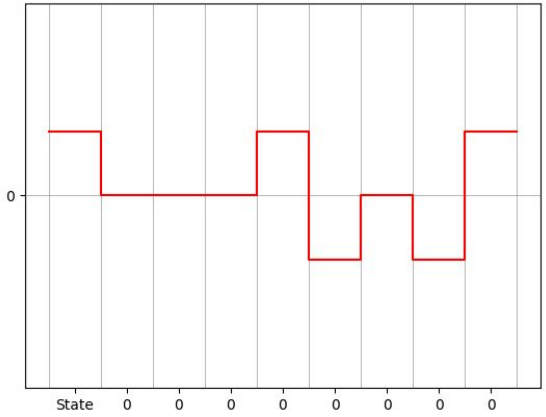
```
def b8zs(bits: str, initial_phase: int) -> None:
    bits = bits_to_int_list(bits)

    signal = []
    phase = initial_phase

    count_zero = 0
    for bit in bits:
        if bit == 1:
            phase *= -1
            signal.append(phase)
            count_zero = 0
    ...
```

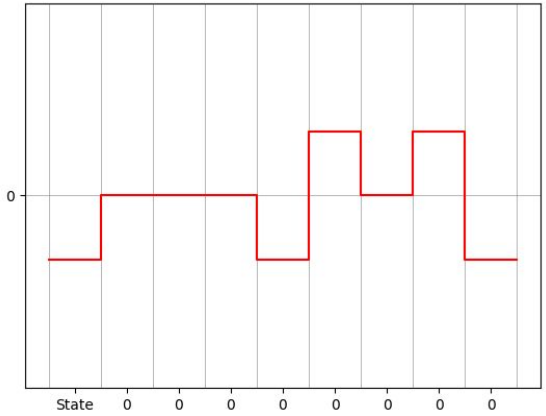




Bipolar 8-Zero Substitution (B8ZS)

```
def b8zs(bits: str, initial_phase: int) -> None:
    bits = bits_to_int_list(bits)

    signal = []
    phase = initial_phase

    count_zero = 0
    for bit in bits:
        ...
        elif bit == 0:
            signal.append(0)
            count_zero += 1
            if count_zero == 8:
                signal[-5:] = [phase, phase*-1, 0, phase*-1,
                               phase]
                count_zero = 0
        ...
```

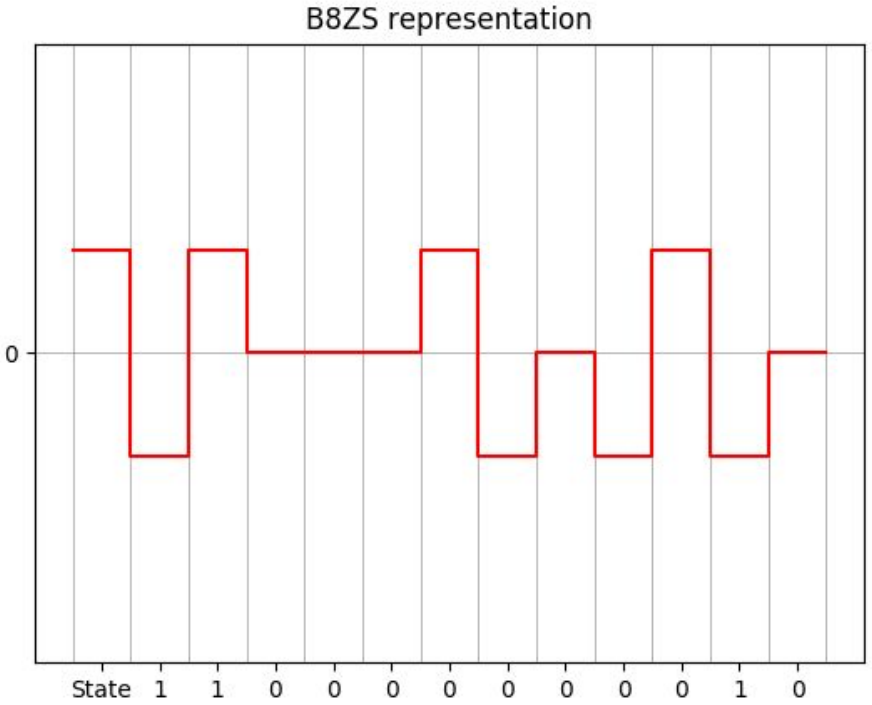




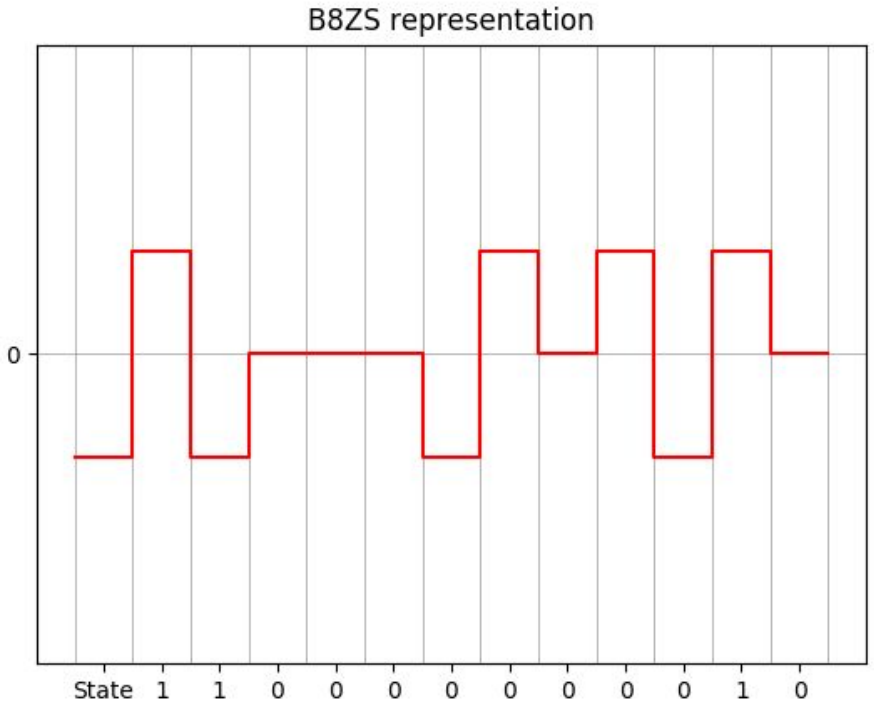
Bipolar 8-Zero Substitution (B8ZS)

```
def b8zs(bits: str, initial_phase: int) -> None:

    ...
    signal = [initial_phase]*2 + signal
    title = "B8ZS representation"
    make_graph(signal, bits, title)
```

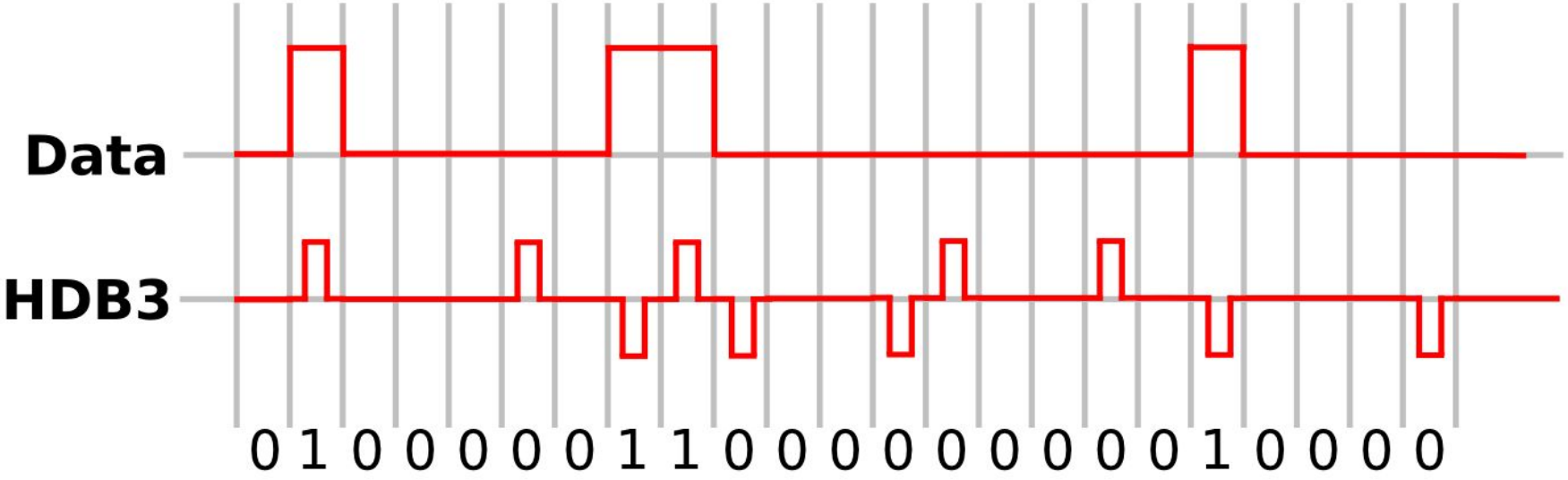
Bipolar 8-Zero Substitution (B8ZS) - Exemplo



Bipolar 8-Zero Substitution (B8ZS) - Exemplo



High Density Bipolar Order 3 (HDB3)



High Density Bipolar Order 3 (HDB3)

Nº de pulsos desde a última substituição

Polaridade do Ultimo Pulso	Ímpar	Par
-	000-	+00+
+	000+	-00-

Função Auxiliar (HDB3)

```
def four_zeros_substitution(count_pulses: int, phase: int) -> List[int]:

    count_pulses_even = (count_pulses % 2 == 0)
    phase_positive = (phase == 1)

    if count_pulses_even and phase_positive:
        modified = [-1, 0, 0, -1]
    elif count_pulses_even and not phase_positive:
        modified = [1, 0, 0, 1]
    elif not count_pulses_even and phase_positive:
        modified = [0, 0, 0, 1]
    else:
        modified = [0, 0, 0, -1]

    return modified
```

Nº de pulsos desde a última substituição

Polaridade do Ultimo Pulso	Ímpar	Par
-	000-	+00+
+	000+	-00-

High Density Bipolar Order 3 (HDB3)

```
def four_zeros_substitution(count_pulses: int, phase: int) -> List[int]:

    count_pulses_even = (count_pulses % 2 == 0)
    phase_positive = (phase == 1)

    if count_pulses_even and phase_positive:
        modified = [-1, 0, 0, -1]
    elif count_pulses_even and not phase_positive:
        modified = [1, 0, 0, 1]
    elif not count_pulses_even and phase_positive:
        modified = [0, 0, 0, 1]
    else:
        modified = [0, 0, 0, -1]

    return modified
```

Nº de pulsos desde a última substituição

Polaridade do Ultimo Pulso	Ímpar	Par
-	000-	+00+
+	000+	-00-

High Density Bipolar Order 3 (HDB3)

```
def four_zeros_substitution(count_pulses: int, phase: int) -> List[int]:

    count_pulses_even = (count_pulses % 2 == 0)
    phase_positive = (phase == 1)

    if count_pulses_even and phase_positive:
        modified = [-1, 0, 0, -1]
    elif count_pulses_even and not phase_positive:
        modified = [1, 0, 0, 1]
    elif not count_pulses_even and phase_positive:
        modified = [0, 0, 0, 1]
    else:
        modified = [0, 0, 0, -1]

    return modified
```

Nº de pulsos desde a última substituição

Polaridade do Ultimo Pulso	Ímpar	Par
-	000-	+00+
+	000+	-00-

High Density Bipolar Order 3 (HDB3)

```
def four_zeros_substitution(count_pulses: int, phase: int) -> List[int]:

    count_pulses_even = (count_pulses % 2 == 0)
    phase_positive = (phase == 1)

    if count_pulses_even and phase_positive:
        modified = [-1, 0, 0, -1]
    elif count_pulses_even and not phase_positive:
        modified = [1, 0, 0, 1]
    elif not count_pulses_even and phase_positive:
        modified = [0, 0, 0, 1]
    else:
        modified = [0, 0, 0, -1]

    return modified
```

Nº de pulsos desde a última substituição

Polaridade do Ultimo Pulso	Ímpar	Par
-	000-	+00+
+	000+	-00-

High Density Bipolar Order 3 (HDB3)

```
def four_zeros_substitution(count_pulses: int, phase: int) -> List[int]:

    count_pulses_even = (count_pulses % 2 == 0)
    phase_positive = (phase == 1)

    if count_pulses_even and phase_positive:
        modified = [-1, 0, 0, -1]
    elif count_pulses_even and not phase_positive:
        modified = [1, 0, 0, 1]
    elif not count_pulses_even and phase_positive:
        modified = [0, 0, 0, 1]
    else:
        modified = [0, 0, 0, -1]

    return modified
```

Nº de pulsos desde a última substituição

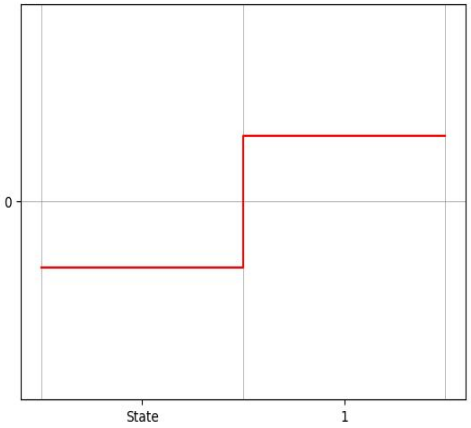
Polaridade do Ultimo Pulso	Ímpar	Par
-	000-	+00+
+	000+	-00-

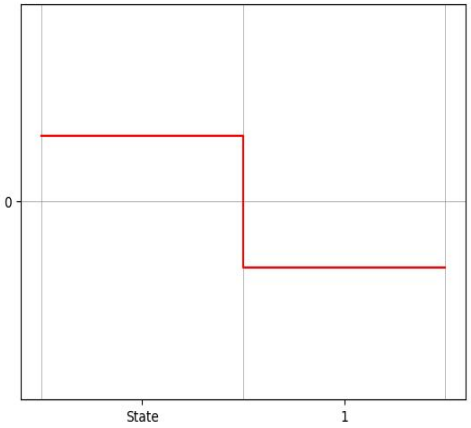
High Density Bipolar Order 3 (HDB3)

```
def hdb3(bits: str, initial_phase: int) -> None:

    bits = bits_to_int_list(bits)

    signal = []
    phase = initial_phase

    count_zero = 0
    count_pulses = 1
    ...
```

High Density Bipolar Order 3 (HDB3)

```
def hdb3(bits: str, initial_phase: int) -> None:

    ...
    for bit in bits:

        if bit == 1:
            phase *= -1
            signal.append(phase)
            count_pulses += 1
            count_zero = 0

        ...
```





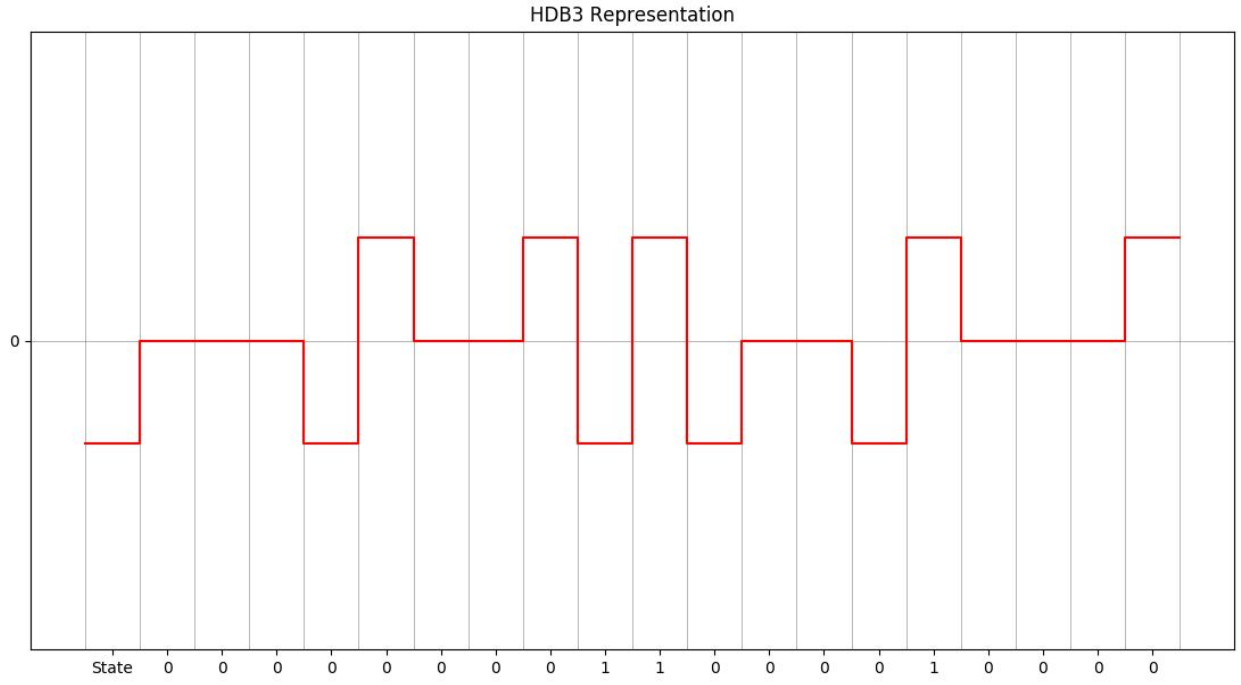
High Density Bipolar Order 3 (HDB3)

```
def hdb3(bits: str, initial_phase: int) -> None:

    ...
    for bit in bits:
        ...
        elif bit == 0:
            count_zero += 1
            signal.append(0)

            if count_zero == 4:
                signal[-4:] = four_zeros_substitution(count_one, phase)
                phase = signal[-1]
                count_pulses = 0
                count_zero = 0
```

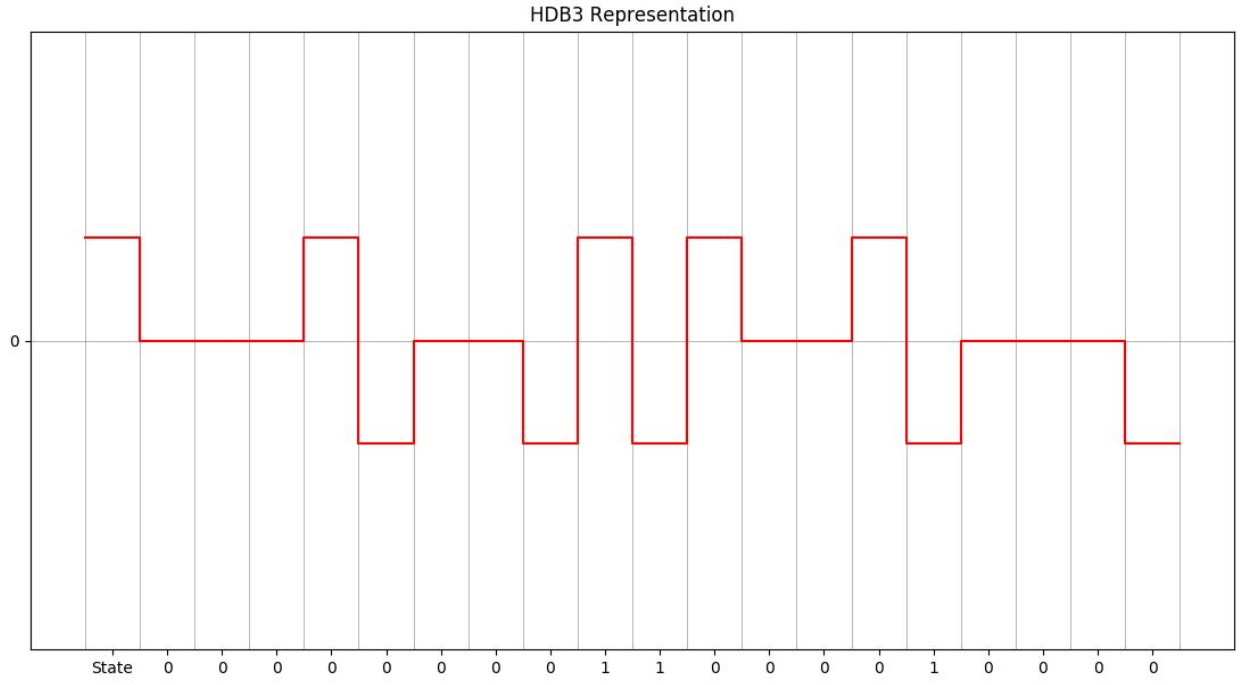
High Density Bipolar Order 3 (HDB3)

```
def hdb3(bits: str, initial_phase: int) -> None:

    ...
    signal = [initial_phase]*2 + signal
    title = "HDB3 Representation"
    make_graph(signal, bits, title)
```

High Density Bipolar Order 3 (HDB3) - Exemplo



High Density Bipolar Order 3 (HDB3) - Exemplo



Fim!

- ✅ Ferramentas
- ✅ Funções Auxiliares
- ✅ Manchester
- ✅ Differential Manchester
- ✅ B8ZS
- ✅ HDB3