Relatório de Implementação e Análise de Viabilidade Econômica da IA para Regulação de Fornos de Cimento

---

Acadêmico: Luigi Francesco Fedele

RU: 4517608

---

1. Arquitetura Neural e Racionais de Codificação

1.1 Arquitetura Proposta

A arquitetura selecionada para resolver o problema de previsibilidade de parada dos fornos rotativos é um **Perceptron de Camada Única**. Esta escolha minimalista é crítica e intencional para atender às severas limitações de hardware do acionador IoT.

- **Entradas ($\mathbf{x}$):** 6 (Níveis de 4 sensores, Temperatura e Densidade da Graxa).
- **Bias ($\mathbf{w_0}$):** 1.
- **Saída ($\mathbf{o}$):** 1 (Decisão Binária: 1 para Manter Produção, -1 para Parada).

1.2 Racionais de Codificação e Linguagem

A linguagem de programação escolhida para a implementação no microcontrolador é **C**.

- **Eficiência (Memória/Processamento):** O compilador C para Assembly apresenta uma das melhores taxas de conversão (1 comando C para 4,1 comandos Assembly), superando Python (1 para 12,7). Isso resulta em um código de máquina muito mais enxuto, essencial para o limite de 512 KBs.
- **Portabilidade:** O C é amplamente utilizado em sistemas embarcados e IoTs, garantindo que o código seja facilmente mantido e atualizado no ambiente industrial.
- **Otimização:** Os pesos sinápticos finais (W_0 a W_6) são definidos como constantes (`#define`), otimizando o uso da memória e evitando o *overhead* de variáveis globais.

2. Código Fonte Funcional (Linguagem C)

O código a seguir implementa o neurônio treinado para tomar a decisão de parada ou continuidade, imprimindo a decisão conforme a solicitação.

```c
#include <stdio.h>

#define W0 -1.0000
#define W1 0.2467
#define W2 0.1900
#define W3 0.2733
#define W4 0.9600
#define W5 -0.3400
#define W6 0.0800

/**
* @brief Simula a decisão do neurônio Perceptron.
```

```
* @param x1 Nível Sensor 1 (normalizado)
* @param x2 Nível Sensor 2 (normalizado)
* @param x3 Nível Sensor 3 (normalizado)
* @param x4 Nível Sensor 4 (normalizado)
* @param x5 Temperatura da Graxa (normalizada)
* @param x6 Densidade da Graxa (normalizada)
* @return int Retorna 1 (Produção Ativa) ou -1 (Parada Necessária)
*/

int tomar_decisao(float x1, float x2, float x3, float x4, float x5, float x6) {
    float f = (x1 * W1) + (x2 * W2) + (x3 * W3) + (x4 * W4) + (x5 * W5) + (x6 * W6) +
W0;
    if (f > 0.0) {
        return 1;
    } else {
        return -1;
    }
}



int main() {
    float entrada_sensores[6];
    int decisao;

    entrada_sensores[0] = 0.1;
    entrada_sensores[1] = 0.2;
    entrada_sensores[2] = 0.1;
    entrada_sensores[3] = 0.2;
    entrada_sensores[4] = 0.3;
    entrada_sensores[5] = 0.4;

    decisao = tomar_decisao(entrada_sensores[0], entrada_sensores[1],
entrada_sensores[2], entrada_sensores[3], entrada_sensores[4], entrada_sensores[5]);


    printf("Condição 1: Leitura dos Sensores: {%.2f, %.2f, %.2f, %.2f, %.2f, %.2f}\n",
entrada_sensores[0], entrada_sensores[1], entrada_sensores[2], entrada_sensores[3],
entrada_sensores[4], entrada_sensores[5]);

    printf("Decisão do Neurônio (Saída): %d\n", decisao);
    printf("Ação Sugerida: %s\n\n", (decisao == 1) ? "MANTER PRODUÇÃO ATIVA" :
"DESLIGAR FORNO/ESTEIRAS");

    entrada_sensores[0] = 0.8;
    entrada_sensores[1] = 0.9;
    entrada_sensores[2] = 0.7;
    entrada_sensores[3] = 0.8;
    entrada_sensores[4] = 0.9;
    entrada_sensores[5] = 0.8;

    decisao = tomar_decisao(entrada_sensores[0], entrada_sensores[1],
entrada_sensores[2], entrada_sensores[3], entrada_sensores[4], entrada_sensores[5]);



    printf("Condição 2: Leitura dos Sensores: {%.2f, %.2f, %.2f, %.2f, %.2f, %.2f}\n",
entrada_sensores[0], entrada_sensores[1], entrada_sensores[2], entrada_sensores[3],
entrada_sensores[4], entrada_sensores[5]);

    printf("Decisão do Neurônio (Saída): %d\n", decisao);
```

```
    printf("Ação Sugerida: %s\n", (decisao == 1) ? "MANTER PRODUÇÃO ATIVA" : "DESLIGAR
FORNO/ESTEIRAS");
    return 0;
}
```

2.1 Evidências do Funcionamento

```
cd "/home/luigiffedele/Projetos/ia_forno_cimento/output"
./"ia-forno-cimenteiro"
➜  ia_forno_cimento cd "/home/luigiffedele/Projetos/ia_forno_cimento/output"
➜  output ./"ia-forno-cimenteiro"
Condição 1: Leitura dos Sensores: {0.10, 0.20, 0.10, 0.20, 0.30, 0.40}
Decisão do Neurônio (Saída): -1
Ação Sugerida: DESLIGAR FORNO/ESTEIRAS

Condição 2: Leitura dos Sensores: {0.80, 0.90, 0.70, 0.80, 0.90, 0.80}
Decisão do Neurônio (Saída): 1
Ação Sugerida: MANTER PRODUÇÃO ATIVA
➜  output
```

Resultado:

```
Condição 1: Leitura dos Sensores: {0.10, 0.20, 0.10, 0.20, 0.30, 0.40}
Decisão do Neurônio (Saída): -1
Ação Sugerida: DESLIGAR FORNO/ESTEIRAS

Condição 2: Leitura dos Sensores: {0.80, 0.90, 0.70, 0.80, 0.90, 0.80}
Decisão do Neurônio (Saída): 1
Ação Sugerida: MANTER PRODUÇÃO ATIVA
```

Exemplo de Saída Esperada:

- Decisão Neurônio (Amostra Normal): 1
- Decisão Neurônio (Amostra Crítica): -1

3. Cálculo de Viabilidade para o IoT (Memória Ocupada)

O cálculo visa demonstrar que o código de 68 linhas (o núcleo lógico da IA) cabe na memória de 512 KBs do acionador IoT.

3.1 Parâmetros e Conversão de Linguagem

Parâmetro	Valor
Linhas de Código C (Base)	68 linhas
Fator de Conversão C → Assembly	4,1
Fator de Conversão Assembly → Opcode	5,1
Custo do Opcode (Média)	4,3 bytes/operação
Memória Máxima Disponível	512 KBs (524.288 bytes)

3.2 Memória de Cálculo

1. Comandos Assembly Equivalentes:

$$\text{Comandos Assembly} = 68 \times 4,1 = 278,8$$

2. Comandos Opcode Equivalentes:

$$\text{Comandos Opcode} = 278,8 \times 5,1 = 1.421,88$$

3. Memória Ocupada (Bytes):

$$\text{Memória Ocupada} = 1.421,88 \times 4,3 \text{ bytes/operação} \approx \mathbf{6.114,08} \text{ bytes}$$

3.3 Conclusão de Viabilidade

A memória ocupada pela aplicação é de $\mathbf{6.114,08}$ bytes, o que corresponde a apenas **1,17%** da memória total disponível (524.288 bytes), confirmando que a implementação é **extremamente viável** e robusta em relação à capacidade limitada do microcontrolador IoT.

4. Cálculo de Custo Equivalente de Manutenção

O cálculo compara o custo mensal de manutenção dos fornos no cenário anterior (com automação, mas sem a IA) com o custo no cenário proposto (com a IA), considerando uma operação 24×7 (720 horas/mês).

4.1 Parâmetros Financeiros e de Produção

Tipo de Parada	Custo/min/forno	% Tempo de Produção (Original)
Parada Prevista (Preventiva)	R$ 0,3 mil	2,7%
Parada Não Programada (Corretiva Imperativa)	R$ 3,1 mil	5,8%

Tempo Total de Produção Mensal:

$$30 \text{ dias} \times 24 \text{ horas/dia} \times 60 \text{ min/hora} = 43.200 \text{ minutos/mês/forno}$$

4.2 Cenário 1: Custo Anterior (Pós-Automação / Pré-IA)

A engenharia de automação existente elevou a previsibilidade para **73%** dos casos. Isso significa que 73% das paradas antes *não programadas* passam a ser *previstas* (custo mais baixo).

1. Paradas Previstas:

$$\text{Tempo Previsto Total} = (\text{Tempo Original Previsto}) + (73\% \text{ do Tempo Não Programado Original})$$

$$\text{Tempo Previsto Total} = (2,7\%) + (0,73 \times 5,8\%) = 2,7\% + 4,234\% = 6,934\%$$

$$\text{Custo Previsto} = 43.200 \text{ min} \times 0,06934 \times \text{R\$ } 0,3 \text{ mil/min} \approx \text{R\$ } \mathbf{899,5} \text{ mil}$$

2. Paradas Não Programadas (Remanescentes):

$$\text{Tempo Não Programado} = (100\% - 73\%) \text{ do Tempo Não Programado Original}$$

$$\text{Tempo Não Programado} = 0,27 \times 5,8\% = 1,566\%$$

$$\text{Custo Não Programado} = 43.200 \text{ min} \times 0,01566 \times \text{R\$ } 3,1 \text{ mil/min} \approx \text{R\$ } \mathbf{2.096,1} \text{ mil}$$

$$\text{Custo Mensal Total (Cenário 1)} = \text{R\$ } \mathbf{899,5} \text{ mil} + \text{R\$ } \mathbf{2.096,1} \text{ mil} = \text{R\$ } \mathbf{2.995,6} \text{ mil/forno}$$

4.3 Cenário 2: Custo com a Implementação da IA

O modelo de IA treinado com 100% de acerto transforma **100%** das paradas corretivas em paradas **previstas**, eliminando as paradas não programadas.

1. **Paradas Não Programadas:** 0% (Custo **R$ 0**)
2. Paradas Previstas:

$$\text{Tempo Previsto Total} = (\text{Tempo Original Previsto}) + (\text{Tempo Original Não Programado})$$

$$\text{Tempo Previsto Total} = 2,7\% + 5,8\% = 8,5\%$$

$$\text{Custo Previsto} = 43.200 \text{ min} \times 0,085 \times \text{R\$ } 0,3 \text{ mil/min} \approx \text{R\$ } \mathbf{1.101,6} \text{ mil}$$

$$\text{Custo Mensal Total (Cenário 2)} = \text{R\$ } \mathbf{1.101,6} \text{ mil/forno}$$

4.4 Comparação e Economia

A implementação da IA gera uma economia mensal significativa por forno:

$$\text{Economia Mensal} = \text{Custo Cenário 1} - \text{Custo Cenário 2}$$

$$\text{Economia Mensal} = \text{R\$ } 2.995,6 \text{ mil} - \text{R\$ } 1.101,6 \text{ mil} = \text{R\$ } \mathbf{1.894,0} \text{ mil/forno}$$

Conclusão de Custo: A solução de IA proposta, ao garantir a previsibilidade total das paradas necessárias, gera uma economia mensal de $\text{R\$ } \mathbf{1.894} \text{ mil}$ (aproximadamente R$ 1,9 milhão) por forno.

---

5. Justificativas Técnicas

- A escolha do Perceptron minimalista e da linguagem C é a única forma viável de garantir a decisão autônoma e residente no microcontrolador IoT de baixa capacidade. O treinamento com 100% de acerto elimina o problema das paradas imperativas não previstas, que é a principal causa do alto custo e das rupturas de contratos.