

Guía de Implementación PCU-TCDS

****Puente Causal Universal - Teoría Cromodinámica Sincrónica****

🎯 Objetivo

Transformar tu ecosistema TCDS actual (disperso en múltiples repos) en un ****PCU ejecutable, falsable y auditable**** que demuestre el "desafío superado" según tu documento `Puente_Causal.pdf`.

📄 Estado Actual vs. Estado Objetivo

Estado Actual (Fragmentado)

...

Ecosistema TCDS actual:

- |— Reloj-Causal-Humano-TCDS/ → Demo JS (E-Veto visual)
- |— TCDS-SISTEMA-PREDICTIVO/ → Landing page + docs
- |— TCDS_CENAPRED_COLLAP_REPO/ → Datos sísmicos (?)
- |— Puentes Físicos (Zenodo) → Papers en PDF
- |— docs/ varios PDFs → Teoría + claims

...

****Problema:**** No hay código ejecutable que implemente el operador $Q-\Sigma-\varphi-\chi$ completo con trazabilidad forense.

Estado Objetivo (PCU Unificado)

...

TCDS-PCU-Universal/

- |— src/
 - | |— eveto.py GENERADO
 - | |— metrics_sigma.py GENERADO
 - | |— pcu_operator.py GENERADO
 - | |— runner.py GENERADO
- |— data/
 - | |— domain_sismo/ Migrar desde CENAPRED
 - | |— domain_humano/ Datos del Reloj Causal
 - | |— domain_fet/ Pendiente
 - | |— domain_trampa/ GENERADOR creado
- |— config/
 - | |— default.yaml GENERADO
- |— notebooks/
 - | |— demo_pcu.ipynb Pendiente
- |— docs/
 - | |— artefacto_axiomatico.pdf Tu PDF existente
 - | |— especificacion_pcu.pdf Tu Puente_Causal.pdf

...

🚀 Plan de Implementación (6 semanas)

Semana 1: Setup Inicial (10 horas)

Día 1-2: Crear repositorio base

```
```bash
En GitHub
git clone https://github.com/geozunac3536-jpg/TCDS-PCU-Universal.git
cd TCDS-PCU-Universal
```

#### # Estructura

```
mkdir -p {src,data,config,runs,outputs,docs,notebooks,tests}
...`
```

#### #### Día 3-4: Copiar archivos generados

##### 1. Copia los 4 scripts Python que generé:

- `eveto.py` → `src/`
- `metrics\_sigma.py` → `src/`
- `pcu\_operator.py` → `src/`
- `runner.py` → `src/`

##### 2. Copia archivos auxiliares:

- `generate\_trampa\_dataset.py` → `src/`
- `demo\_pcu\_completo.py` → raíz
- `config/default.yaml` → `config/`
- `requirements.txt` → raíz
- `README.md` → raíz

#### #### Día 5: Test inicial

```
```bash
python src/eveto.py
python src/metrics_sigma.py
python src/pcu_operator.py
...`
```

****Entregable:**** Repositorio funcional con tests básicos pasando.

Semana 2: Migración de Datos (12 horas)

Datos sísmicos

```
```bash
Si tienes CENAPRED_COLLAP_REPO
cp -r ../TCDS_CENAPRED_COLLAP_REPO/data/*.csv data/domain_sismo/
```

#### # Crear manifest

```
cat > data/domain_sismo/manifest.json << EOF
```

```
{
 "domain_name": "sismo",
 "observables": ["aceleracion_x", "aceleracion_y", "aceleracion_z"],
 "windowing": "p:q = 100:200 (ventanas pre/post evento)",
 "noise_model": "Ruido instrumental + ambiental",
 "coherence_proxy": "Σ-metrics (LI, R) sobre aceleración total",
 "forcing_proxy": "Nucleación sísmica (empuje Q de liberación de energía)",
 "substrate": "Litosfera terrestre (χ)",
 "assumptions": "Precusores coherenciales antes de eventos M≥4.0",
 "failure_modes": "Falsos positivos en tormentas, tráfico pesado"
}
EOF
...

```

```
Datos humanos (del Reloj Causal)
``bash
Si tienes logs del Reloj Causal
(probablemente no existan aún, usar sintéticos)
python -c "
import numpy as np
t = np.linspace(0, 300, 10000) # 5 min a 33Hz
accel = 9.8 + 0.5*np.sin(2*np.pi*1.2*t) + 0.1*np.random.randn(len(t))
np.save('data/domain_humano/acelerometro_ejemplo.npy', accel)
"
...

```

```
Dataset trampa
``bash
python src/generate_trampa_dataset.py
...

```

**\*\*Entregable:\*\*** Directorios `data/domain\_\*/` poblados con manifiestos.

---

### **\*\*Semana 3: Pipeline Ejecutable\*\*** (15 horas)

```
Integrar datos reales con runner
``python
test_pipeline_real.py
from runner import PCURunner
import numpy as np

Cargar datos sísmicos
sismo_data = np.load('data/domain_sismo/evento_001.npy')
humano_data = np.load('data/domain_humano/acelerometro_ejemplo.npy')
trampa_data = np.load('data/domain_trampa/gaussian_noise.npy')

```

```
data = {
 "sismo": [sismo_data],
 "humano": [humano_data],

```

```
"trampa": trampa_data.tolist()
```

```
}
```

```
runner = PCURunner()
```

```
verdicts = runner.run(data, seed=42)
```

```
...
```

```
Probar reproducibilidad
```

```
```bash
```

```
python demo_pcu_completo.py
```

```
```
```

**\*\*Entregable:\*\*** Pipeline end-to-end con datos reales pasando E-Veto.

```

```

### **\*\*Semana 4: Trazabilidad & Auditoría\*\*** (12 horas)

```
Implementar role_checker.py
```

```
```python
```

```
# src/role_checker.py
```

```
"""
```

Verifica separación de roles en el grafo de ejecución.

Detecta si un módulo cruza límites de responsabilidad.

```
"""
```

```
import ast
```

```
import inspect
```

```
def check_role_separation():
```

```
    # Inspeccionar imports de cada módulo
```

```
    # Marcar como CORRUPTED_PIPELINE si:
```

```
    # - eveto.py importa pcu_operator.py
```

```
    # - metrics_sigma.py importa eveto.py
```

```
    # etc.
```

```
    pass
```

```
if __name__ == "__main__":
```

```
    check_role_separation()
```

```
...
```

```
#### Generar firmas SHA-256
```

```
```bash
```

```
Después de cada run
```

```
sha256sum runs/run_*/verdicts.jsonl > runs/run_*/checksums.txt
```

```
```
```

****Entregable:**** Auditoría automatizada + checksums verificables.

```
---
```

Semana 5: Jupyter Notebook Demo (8 horas)

Crear `notebooks/demo_pcu.ipynb`

```python

# Celdas:

# 1. Instalación

!pip install -r ../requirements.txt

# 2. Imports

from src.runner import PCURunner

import numpy as np

import matplotlib.pyplot as plt

# 3. Cargar datos

# ...

# 4. Ejecutar PCU

# ...

# 5. Visualizar resultados

# Gráficos de  $\Sigma$ -metrics, veredictos por dominio, etc.

# 6. Test reproducibilidad

# ...

# 7. Conclusión

# Checklist de "desafío superado"

```

****Entregable:**** Notebook ejecutable en Colab/Binder.

Semana 6: Publicación & DOI (10 horas)

Preparar para Zenodo

1. Asegurarse de que todos los tests pasen

2. Escribir `CITATION.cff`

3. Crear release en GitHub (v1.0.0)

4. Subir a Zenodo

Actualizar portal TCDS

```markdown

# En geozunac3536-jpg.github.io

## PCU - Puente Causal Universal

**\*\*DOI:\*\*** [10.5281/zenodo.XXXXX](https://doi.org/10.5281/zenodo.XXXXX)

**\*\*Repositorio:\*\*** [GitHub](https://github.com/geozunac3536-jpg/TCDS-PCU-Universal)

El PCU es el operador ejecutable que unifica todos los dominios TCDS

bajo criterios de validez reproducibles y falsables.

[Ver demo en Colab](https://colab.research.google.com/github/...)

...

**\*\*Entregable:\*\*** DOI publicado + portal actualizado.

---

## ## ✅ Checklist de Aceptación Final

Antes de publicar, verifica:

- ☐ Tests básicos pasan (`pytest tests/`)
- ☐ `demo\_pcu\_completo.py` ejecuta sin errores
- ☐ Reproducibilidad  $\geq 95\%$  en 3 corridas
- ☐ Dataset trampa RECHAZADO (0 ACCEPT)
- ☐ Al menos 1 dominio real (sismo) procesado
- ☐ Hashes SHA-256 generados y verificables
- ☐ README.md completo con instrucciones
- ☐ Notebook demo funcional
- ☐ Licencia DUAL clara (CC BY-NC-SA + comercial)
- ☐ DOI Zenodo obtenido

---

## ## 🛠 Comandos Rápidos

```
```bash
# Setup inicial
git clone https://github.com/geozunac3536-jpg/TCDS-PCU-Universal.git
cd TCDS-PCU-Universal
pip install -r requirements.txt

# Test rápido
python src/eveto.py
python src/metrics_sigma.py

# Generar trampa
python src/generate_trampa_dataset.py

# Demo completo
python demo_pcu_completo.py

# Tests
pytest tests/ -v

# Verificar reproducibilidad
python -c "
from runner import PCURunner
import numpy as np
```

```

data = {'test': [np.random.randn(500)]]
runner = PCURunner()

v1 = runner.run(data, seed=42)
v2 = runner.run(data, seed=42)

rep = runner.compute_reproducibility_between_runs(
    runner.run_id, runner.run_id
)
print(f'Rep: {rep[\"Rep\"]*100:.1f}%')
"
...

---

```

🎓 Próximos Pasos Después de Publicación

1. **Paper técnico**: "PCU: A Falsifiable Bridge Operator for Cross-Domain Causal Validation"
2. **Preprint en arXiv/viXra**
3. **Desafío público**: "Si alguien hace que el PCU acepte ruido sin modificar código, TCDS queda refutado"
4. **Integración con Reloj Causal**: Los datos del reloj web alimentan el PCU
5. **Dashboard en tiempo real**: Mostrar veredictos E-Veto de nodos activos

****Este documento es tu hoja de ruta ejecutable. Cada semana tiene entregables concretos.****

¿Empezamos con la Semana 1?