Desenvolvimento de uma Plataforma Aberta de Hardware e Software para a Construção de uma Rede de Detectores de Raios Cósmicos

G. Saito, M. Leite, M. Kuriyama, R. de Paula, R. Menegasso

Instituto de Física da USP

30 de Novembro de 2020 - I Encontro sobre Divulgação e Ensino de Física de Partículas

Proposta e objetivos

Criar uma rede de detectores de raios cósmicos em escolas pública e privadas envolvendo estudantes e professores na montagem e análise de dados, motivando o estudo de tópicos de física de partículas e tecnologias derivadas.

- Coletar dados de detectores construídos pelos alunos e
- Disponibilizar ferramentas para análise dos dados;
- Baixo custo, operação segura e flexibilidade;
- Proposta pedagógica de longo prazo;
- Cobertura de uma área extensa.

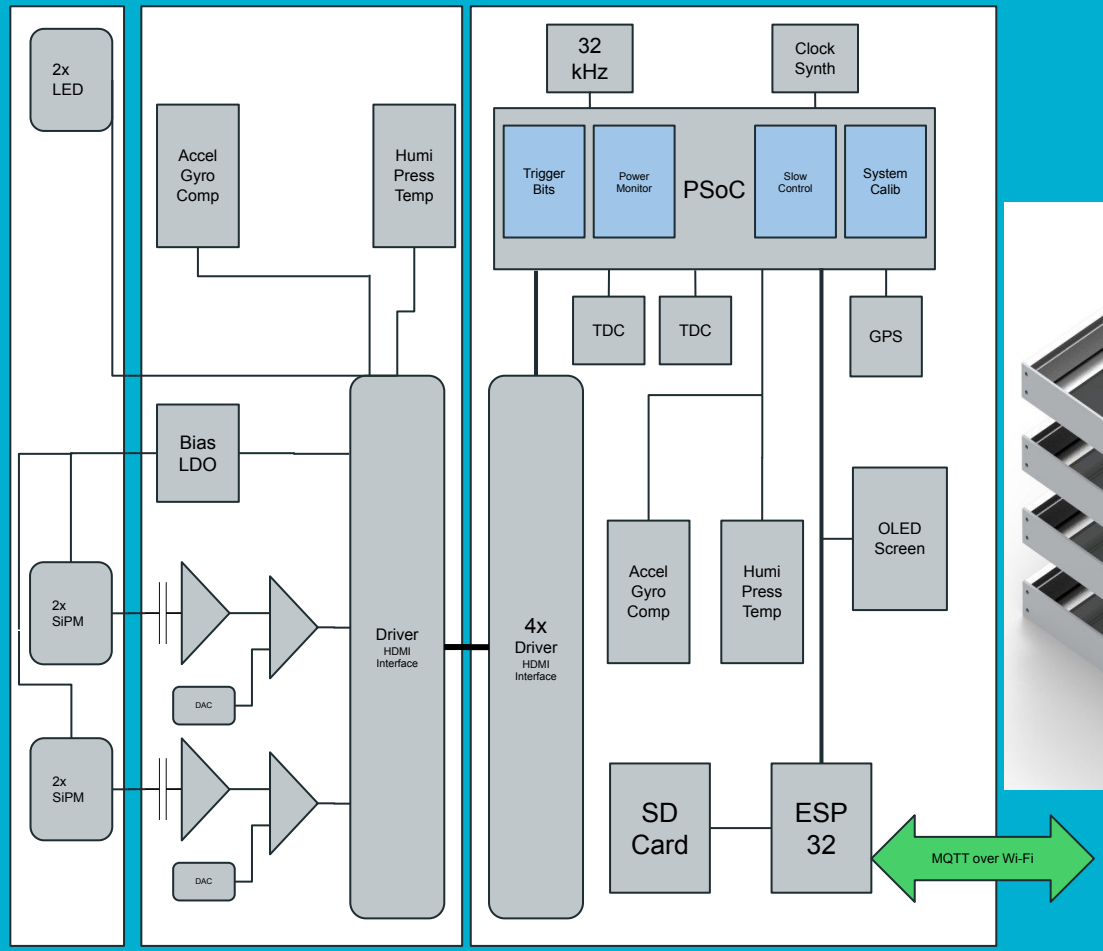
Características

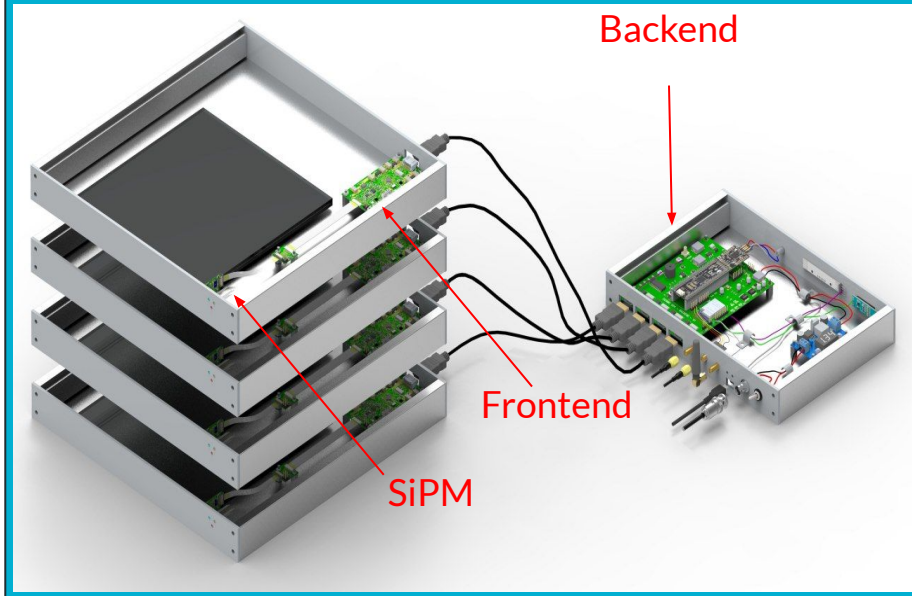
- 8 canais de detecção por estação
- Cintiladores Plásticos
- Fotomultiplicadoras de Silício (baixa tensão de operação)
- Marcação de tempo via GPS (dezenas de nanosegundos de acurácia)
- Detecção do plano de orientação (bússola, giroscópio e acelerômetro)
- Coordenadas da estação
- Conexão Wi-Fi
- Medida de temperatura, umidade, pressão
- Auto-calibração
- Baixo custo e simples de operar!

SiPM

1 a 4 Frontends

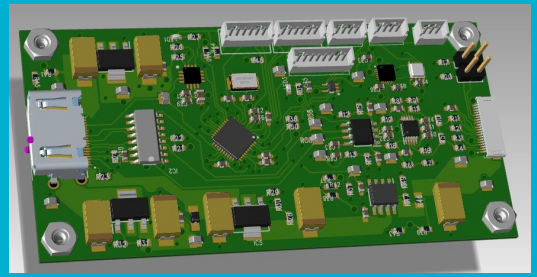
Backend



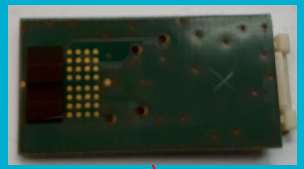


Placas desenvolvidas

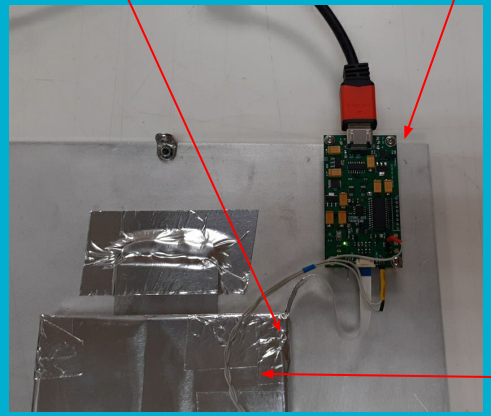
Frontend 2
Já em desenvolvimento



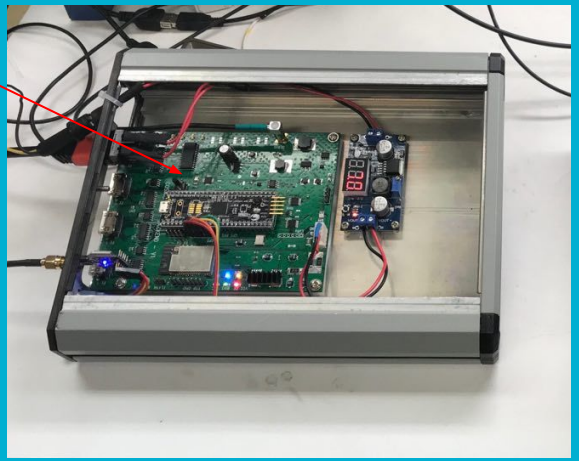
SiPM



Frontend

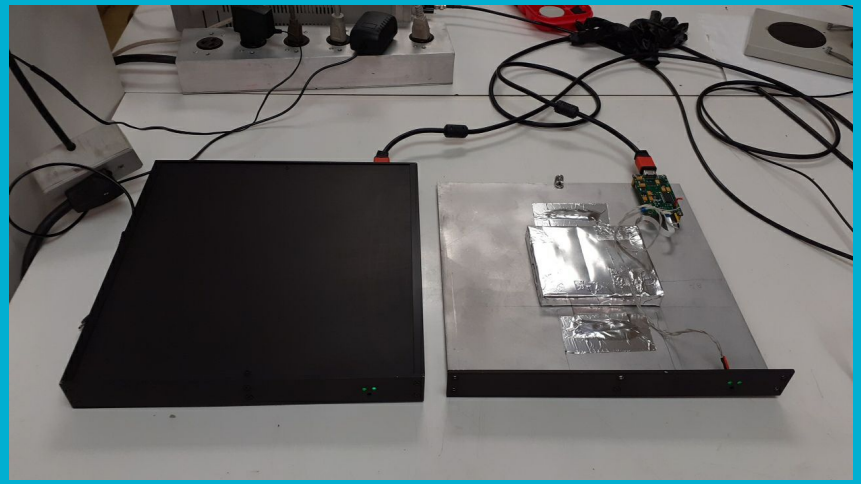


Backend



Cintilador

Protótipo Funcional



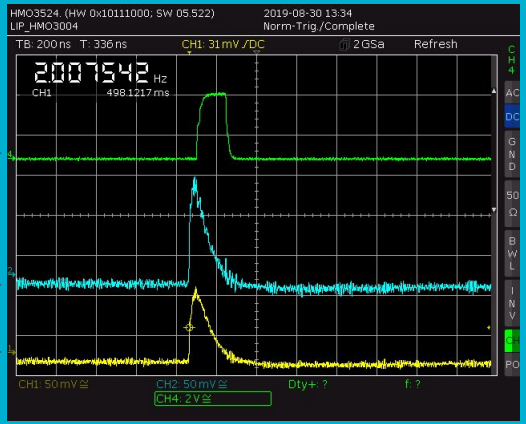
Totalmente projetado na USP

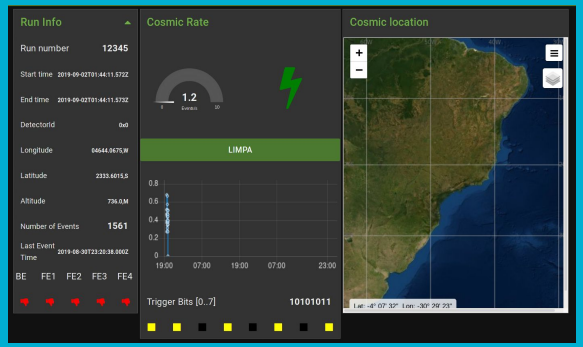
- Desenvolvimento do firmware
- Primeira coleta de dados

Coincidência dos Sinais

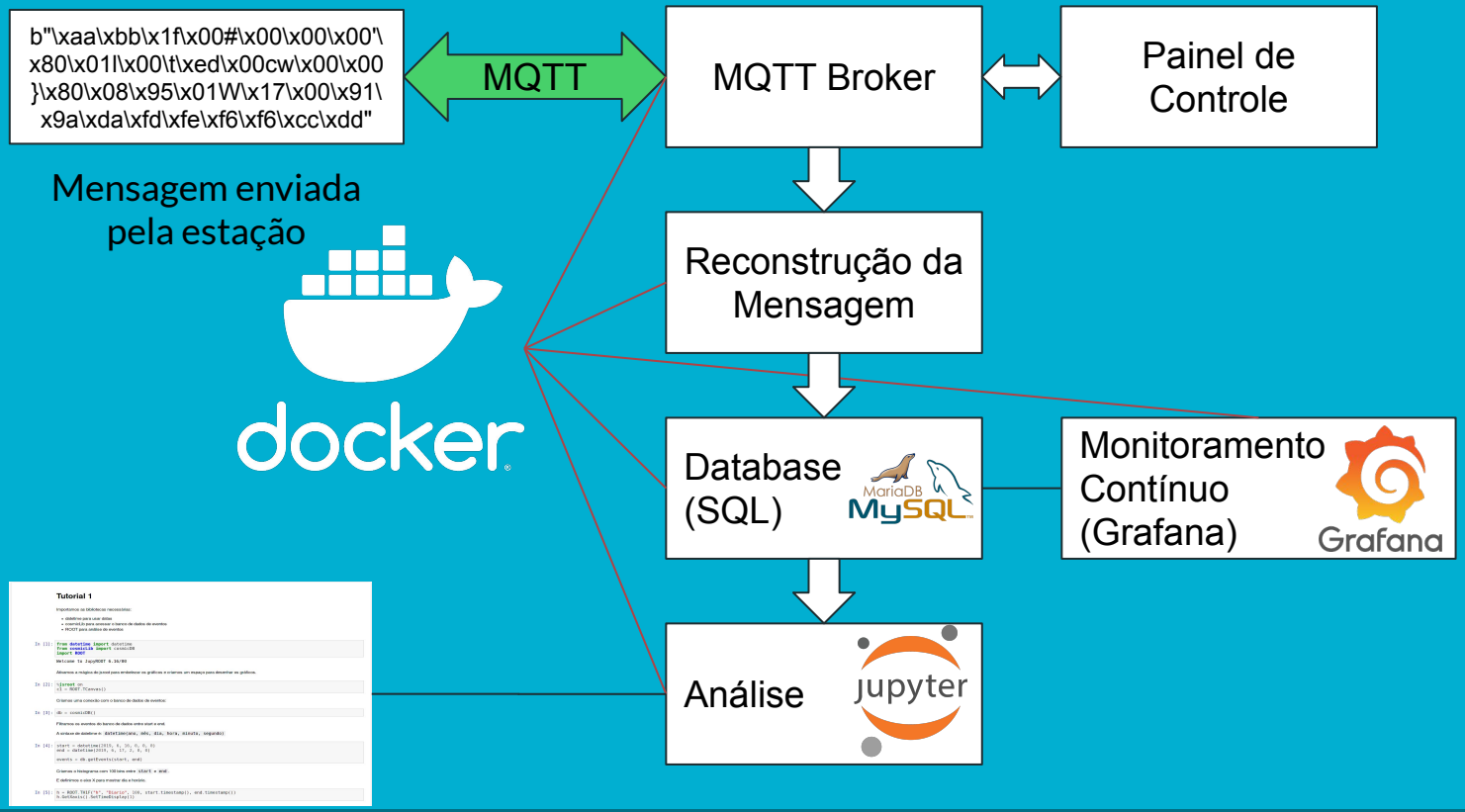
Sinal do SiPM

Sinal do SiPM





Software


b"\xaa\xbb\x1f\x00#\x00\x00\x00'\x80\x01l\x00\t\xed\x00cw\x00\x00}\x80\x08\x95\x01W\x17\x00\x91\x9a\xda\xfd\xfe\xf6\xf6\xcc\xdd"
MQTT
MQTT Broker
Painel de Controle
Mensagem enviada pela estação
docker
Reconstrução da Mensagem
Database (SQL)
MariaDB
MySQL
Monitoramento Contínuo (Grafana)
Grafana
Tutorial 1
Análise
jupyter

Dados

— Salvos no Banco de Dados:

- Instante em que ocorreu o trigger
- Sensores que detectaram o evento
- ID e posição da estação
- Largura do sinal
- RunNumber &CosmicBlock
- # Eventos detectados
- Tensão de bias e threshold dos detectores
- Sensores
- Calibração
- Informações extras do GPS

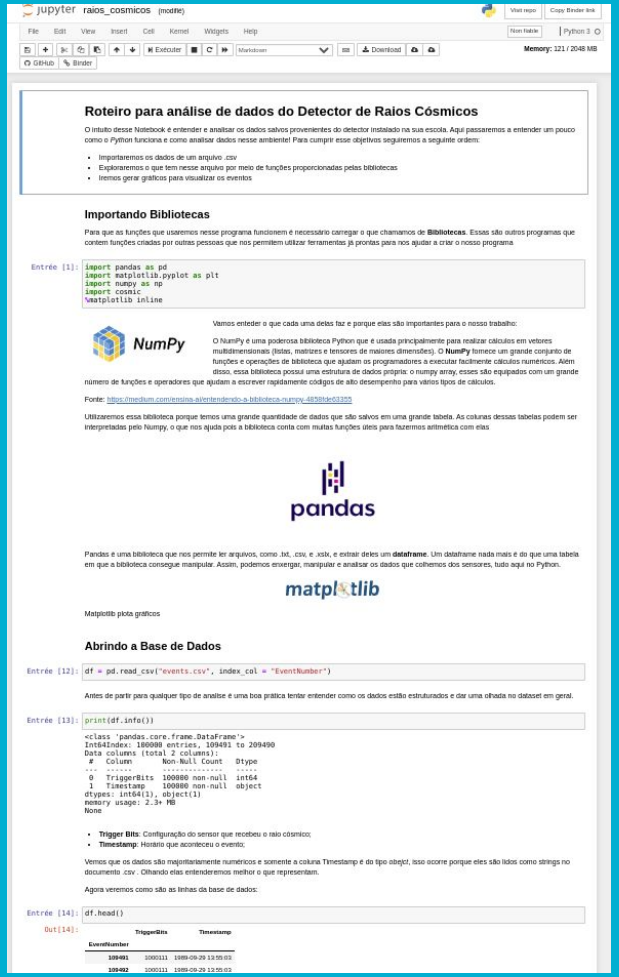
Bits de Trigger

Exemplo de dado salvo: quais sensores detectaram um sinal naquele evento

Provisoriamente disponível em: https://mybinder.org/v2/gh/Materloki/Cosmic-Binder/HEAD

Interface de Análise

- Acesso persistente via JupyterHub
- Ambiente Python via Jupyter Notebook
- Bibliotecas de análise de dados
- Dados públicos disponíveis no banco de dados



jupyter raios_cosmicos (modifié)

File Edit View Insert Cell Kernel Widgets Help

Non fiable | Python 3

Memory: 121 / 2048 MB

GitHub | Binder

Roteiro para análise de dados do Detector de Raios Cósmicos

O intuito desse Notebook é entender e analisar os dados salvos provenientes do detector instalado na sua escola. Aqui passaremos a entender um pouco como o Python funciona e como analisar dados nesse ambiente! Para cumprir esse objetivos seguiremos a seguinte ordem:

- Importaremos os dados de um arquivo .csv
- Exploraremos o que tem nesse arquivo por meio de funções proporcionadas pelas bibliotecas
- Iremos gerar gráficos para visualizar os eventos

Importando Bibliotecas

Para que as funções que usaremos nesse programa funcionem é necessário carregar o que chamamos de **Bibliotecas**. Essas são outros programas que contem funções criadas por outras pessoas que nos permitem utilizar ferramentas já prontas para nos ajudar a criar o nosso programa.

Entrée [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import cosmic
%matplotlib inline
```

NumPy

Vamos enteder o que cada uma delas faz e porque elas são importantes para o nosso trabalho:

O NumPy é uma poderosa biblioteca Python que é usada principalmente para realizar cálculos em vetores multidimensionais (listas, matrizes e tensores de maiores dimensões). O **NumPy** fornece um grande conjunto de funções e operações de biblioteca que ajudam os programadores a executar facilmente cálculos numéricos. Além disso, essa biblioteca possui uma estrutura de dados própria: o numpy array, esses são equipados com um grande número de funções e operadores que ajudam a escrever rapidamente códigos de alto desempenho para vários tipos de cálculos.

Fonte: https://medium.com/ensina-ai/entendendo-a-biblioteca-numpy-4858fde63355

Utilizaremos essa biblioteca porque temos uma grande quantidade de dados que são salvos em uma grande tabela. As colunas dessas tabelas podem ser interpretadas pelo Numpy, o que nos ajuda pois a biblioteca conta com muitas funções úteis para fazermos aritmética com elas

pandas

Pandas é uma biblioteca que nos permite ler arquivos, como .txt, .csv, e .xslx, e extrair deles um **dataframe**. Um dataframe nada mais é do que uma tabela em que a biblioteca consegue manipular. Assim, podemos envergar, manipular e analisar os dados que colhemos dos sensores, tudo aqui no Python.

matplotlib

Matplotlib plota graficos

Abrindo a Base de Dados

Entrée [12]:

```
df = pd.read_csv("events.csv", index_col = "EventNumber")
```

Antes de partir para qualquer tipo de analise é uma boa pratica tentar entender como os dados estão estruturados e dar uma olhada no dataset em geral.

Entrée [13]:

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100000 entries, 109491 to 209490
Data columns (total 2 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   TriggerBits  100000 non-null  int64
 1   Timestamp    100000 non-null  object
dtypes: int64(1), object(1)
memory usage: 2.3+ MB
None
```

- **Trigger Bits**: Configuração do sensor que recebeu o raio cósmico;
- **Timestamp**: Horário que aconteceu o evento;

Vemos que os dados são majoritariamente numéricos e somente a coluna Timestamp é do tipo *object*, isso ocorre porque eles são lidos como strings no documento .csv . Olhando elas entenderemos melhor o que representam.

Agora veremos como são as linhas da base de dados:

Entrée [14]:

```
df.head()
```

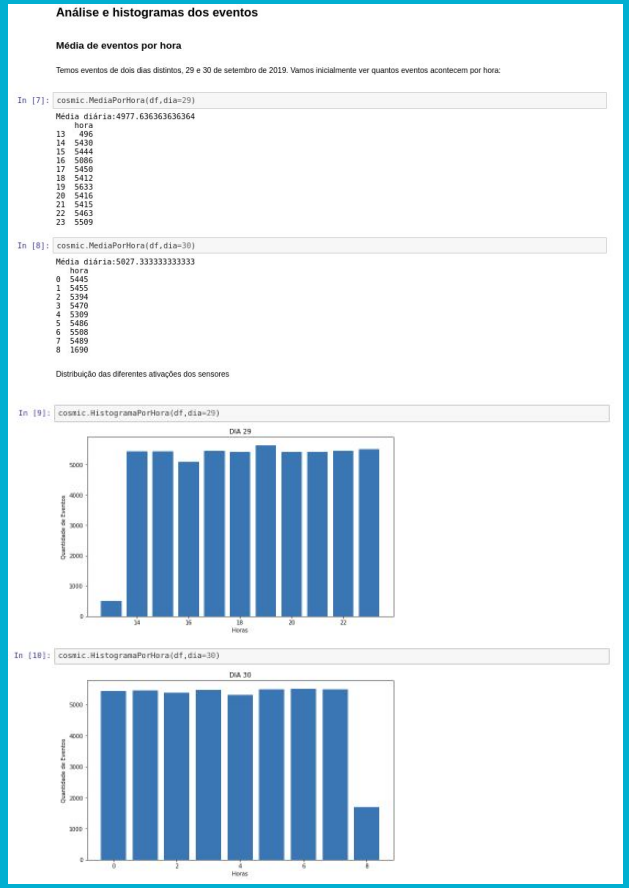
Out[14]:

EventNumber	TriggerBits	Timestamp
109491	1000111	1989-09-29 13:55:03
109492	1000111	1989-09-29 13:55:03

Provisoriamente disponível em: https://mybinder.org/v2/gh/Materloki/Cosmic-Binder/HEAD

Análise

- Filtragem de eventos
  - Intervalo de horário
  - Posição
  - Sensores
  - Bits de trigger
  - Configuração da estação
- Construção de histogramas e gráficos
  - Frequência de eventos
- Média, desvio-padrão, correlação...
- Simultaneidade de eventos entre estações



Análise e histogramas dos eventos

Média de eventos por hora

Temos eventos de dois dias distintos, 29 e 30 de setembro de 2019. Vamos inicialmente ver quantos eventos acontecem por hora:

```
In [7]: cosmic.MediaPorHora(df,dia=29)
```

```
Média diária:4977.636363636364
      hora
13   496
14  5430
15  5444
16  5086
17  5450
18  5412
19  5633
20  5416
21  5415
22  5463
23  5509
```

```
In [8]: cosmic.MediaPorHora(df,dia=30)
```

```
Média diária:5027.333333333333
      hora
0   5445
1   5455
2   5394
3   5470
4   5309
5   5486
6   5588
7   5489
8   1690
```

Distribuição das diferentes ativações dos sensores

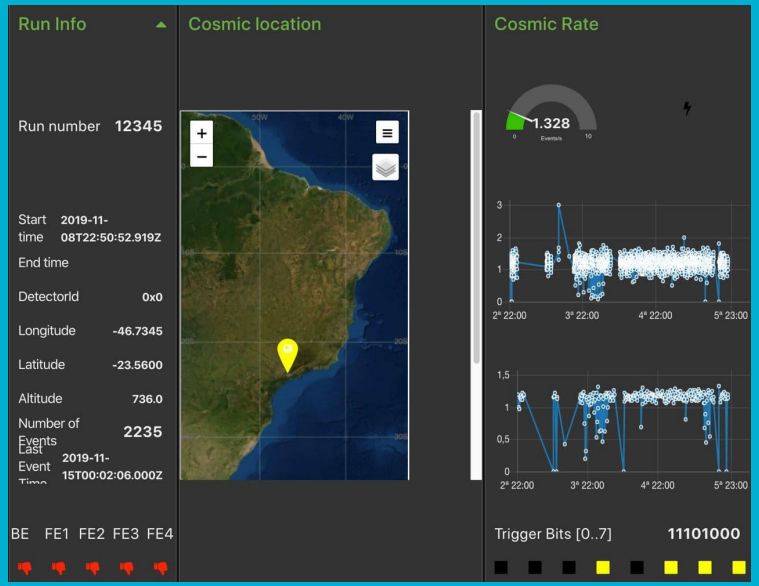
```
In [9]: cosmic.HistogramaPorHora(df,dia=29)
```



```
In [10]: cosmic.HistogramaPorHora(df,dia=30)
```

Monitoramento

- Coleta de dados
  - Frequência de eventos
  - Posição da estação
  - Sensores
- Ajuste de parâmetros
  - Tensão de polarização dos SiPM
  - Tensão limiar de detecção dos sinais
  - Seleção dos bits de trigger
- Situação da infraestrutura de servidores
  - CPU, memória, rede
  - Usuários conectados

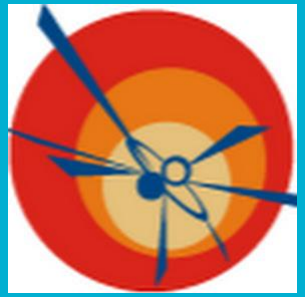


Painel de monitoramento com NodeRed

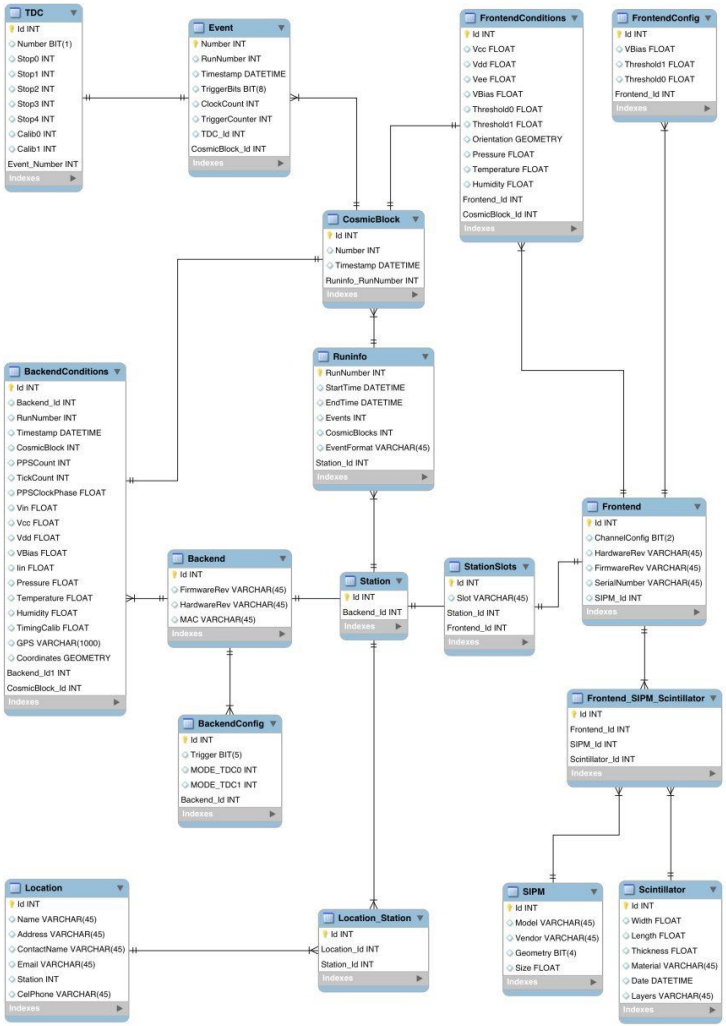
Comentários Finais

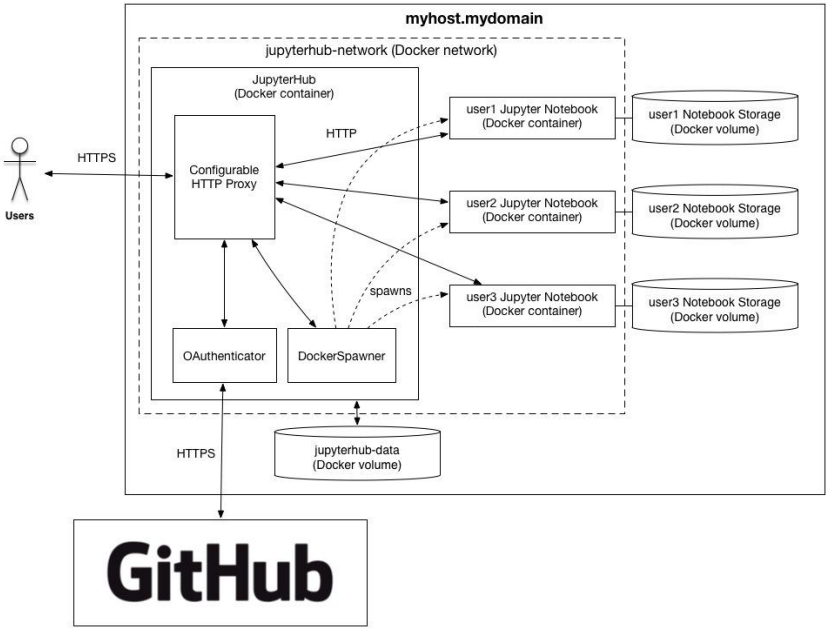
- Protótipo Funcional
- Desenvolvendo a segunda iteração da estação
- Infraestrutura de software em servidores na USP

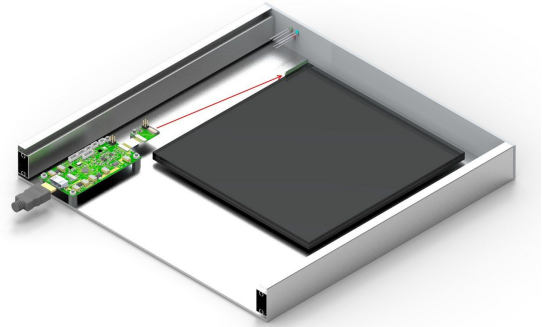
Agradecemos a RENAFAE pelo apoio na construção do protótipo.

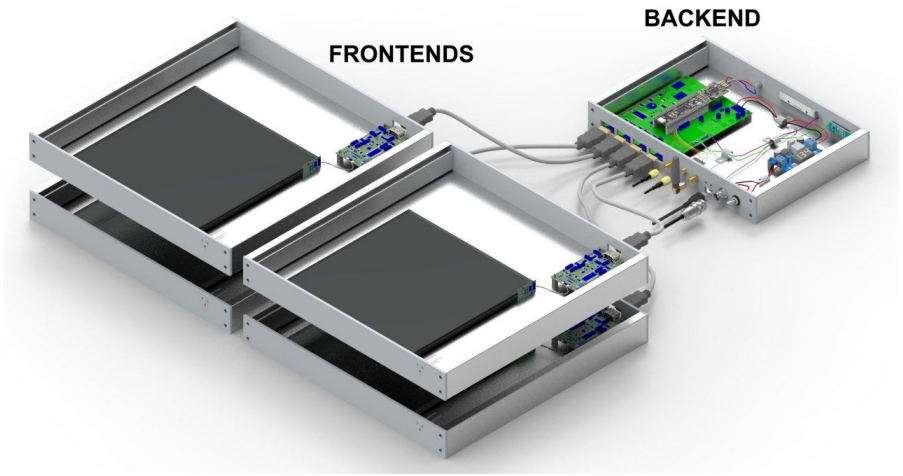


Backup

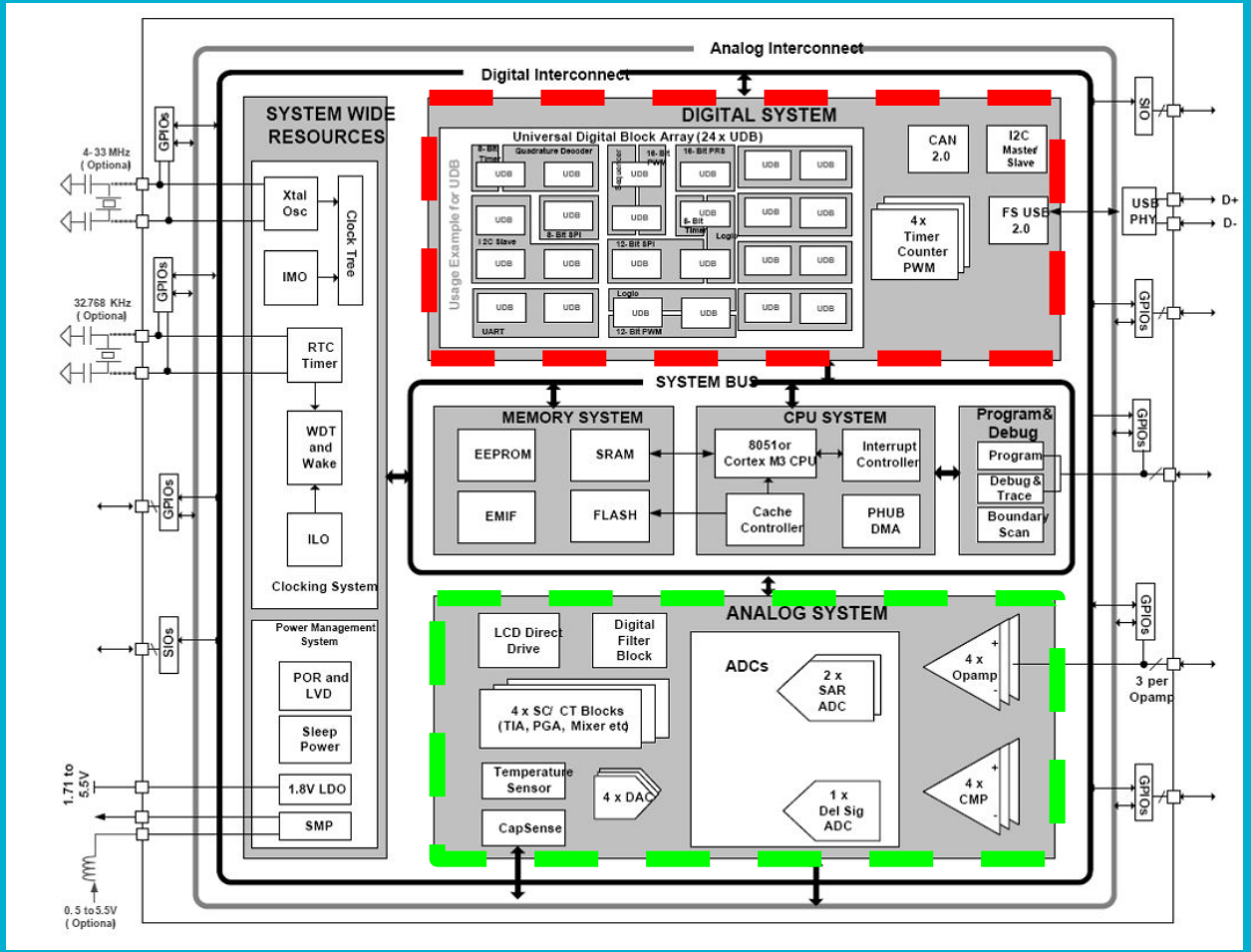

TDC
Id INT
Number BIT(1)
Stop0 INT
Stop1 INT
Stop2 INT
Stop3 INT
Stop4 INT
Calib0 INT
Calib1 INT
Event_Number INT
Indexes
Event
Number INT
RunNumber INT
Timestamp DATETIME
TriggerBits BIT(8)
ClockCount INT
TriggerCounter INT
TDC_Id INT
CosmicBlock_Id INT
Indexes
FrontendConditions
Id INT
Vcc FLOAT
Vdd FLOAT
Vee FLOAT
VBias FLOAT
Threshold0 FLOAT
Threshold1 FLOAT
Orientation GEOMETRY
Pressure FLOAT
Temperature FLOAT
Humidity FLOAT
Frontend_Id INT
CosmicBlock_Id INT
Indexes
FrontendConfig
Id INT
VBias FLOAT
Threshold1 FLOAT
Threshold0 FLOAT
Frontend_Id INT
Indexes
CosmicBlock
Id INT
Number INT
Timestamp DATETIME
RunInfo_RunNumber INT
Indexes
RunInfo
RunNumber INT
StartTime DATETIME
EndTime DATETIME
Events INT
CosmicBlocks INT
EventFormat VARCHAR(45)
Station_Id INT
Indexes
BackendConditions
Id INT
Backend_Id INT
RunNumber INT
Timestamp DATETIME
CosmicBlock INT
PPSCount INT
TickCount INT
PPSClockPhase FLOAT
Vin FLOAT
Vcc FLOAT
Vdd FLOAT
VBias FLOAT
Iin FLOAT
Pressure FLOAT
Temperature FLOAT
Humidity FLOAT
TimingCalib FLOAT
GPS VARCHAR(1000)
Coordinates GEOMETRY
Backend_Id1 INT
CosmicBlock_Id INT
Indexes
Backend
Id INT
FirmwareRev VARCHAR(45)
HardwareRev VARCHAR(45)
MAC VARCHAR(45)
Indexes
Station
Id INT
Backend_Id INT
Indexes
StationSlots
Id INT
Slot VARCHAR(45)
Station_Id INT
Frontend_Id INT
Indexes
Frontend
Id INT
ChannelConfig BIT(2)
HardwareRev VARCHAR(45)
FirmwareRev VARCHAR(45)
SerialNumber VARCHAR(45)
SIPM_Id INT
Indexes
BackendConfig
Id INT
Trigger BIT(5)
MODE_TDC0 INT
MODE_TDC1 INT
Backend_Id INT
Indexes
Frontend_SIPM_Scintillator
Id INT
Frontend_Id INT
SIPM_Id INT
Scintillator_Id INT
Indexes
Location
Id INT
Name VARCHAR(45)
Address VARCHAR(45)
ContactName VARCHAR(45)
Email VARCHAR(45)
Station INT
CellPhone VARCHAR(45)
Indexes
Location_Station
Id INT
Location_Id INT
Station_Id INT
Indexes
SIPM
Id INT
Model VARCHAR(45)
Vendor VARCHAR(45)
Geometry BIT(4)
Size FLOAT
Indexes
Scintillator
Id INT
Width FLOAT
Length FLOAT
Thickness FLOAT
Material VARCHAR(45)
Date DATETIME
Layers VARCHAR(45)
Indexes

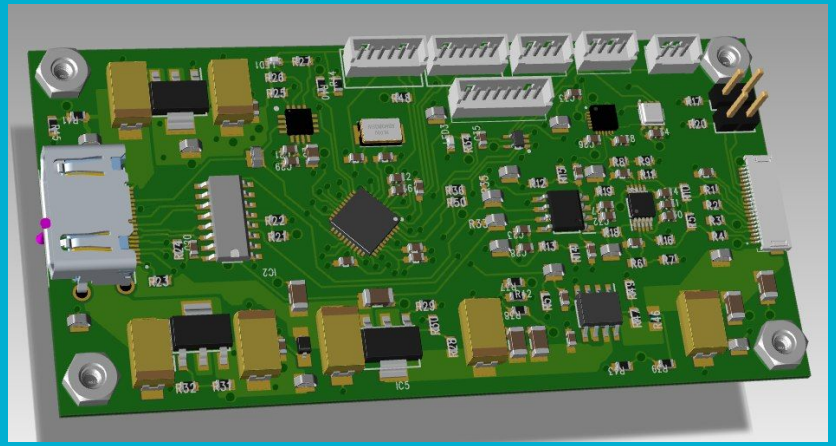

myhost.mydomain
jupyterhub-network (Docker network)
JupyterHub
(Docker container)
Configurable
HTTP Proxy
HTTP
HTTPS
Users
OAuthenticator
DockerSpawner
spawns
user1 Jupyter Notebook
(Docker container)
user2 Jupyter Notebook
(Docker container)
user3 Jupyter Notebook
(Docker container)
user1 Notebook Storage
(Docker volume)
user2 Notebook Storage
(Docker volume)
user3 Notebook Storage
(Docker volume)
HTTPS
jupyterhub-data
(Docker volume)
GitHub

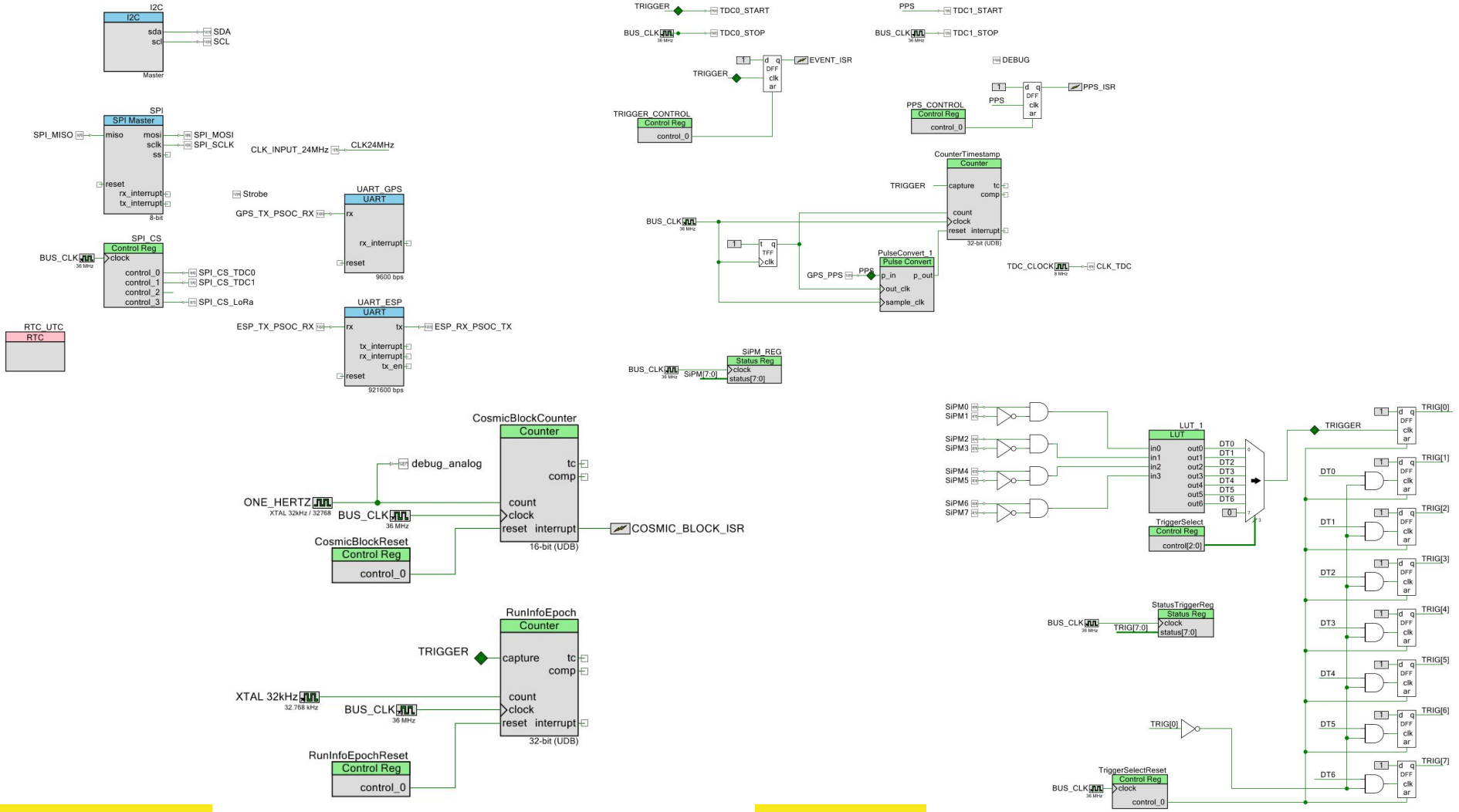


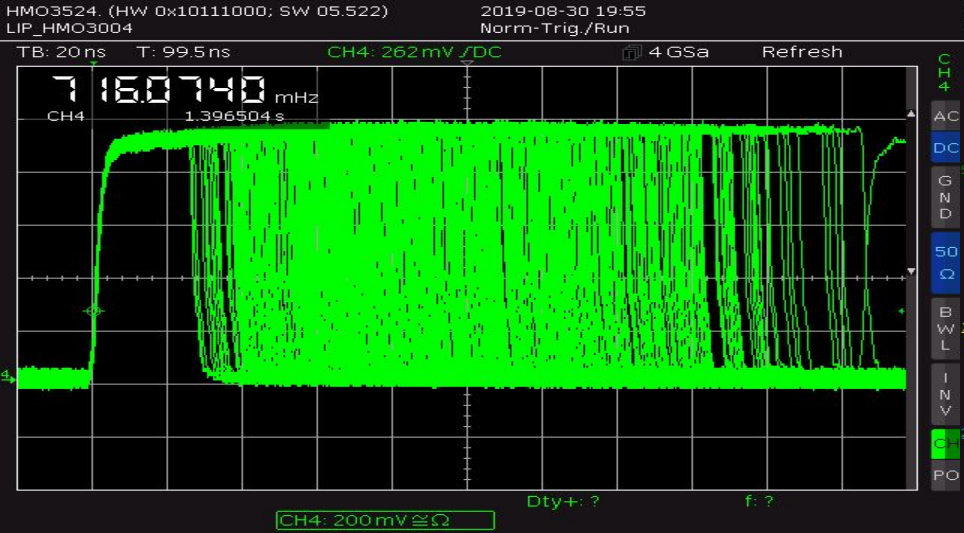

FRONTENDS
BACKEND

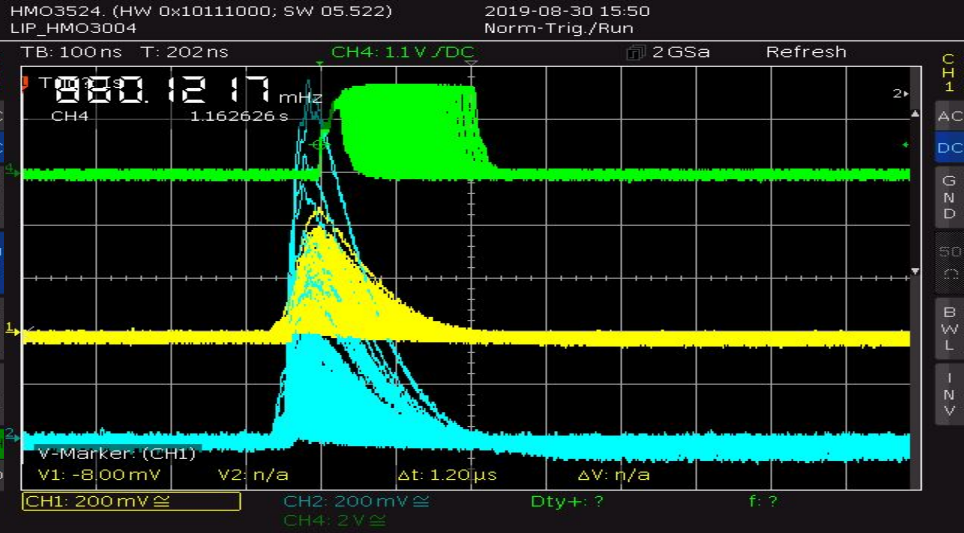
Baseado em um Cypress PSoC 5L

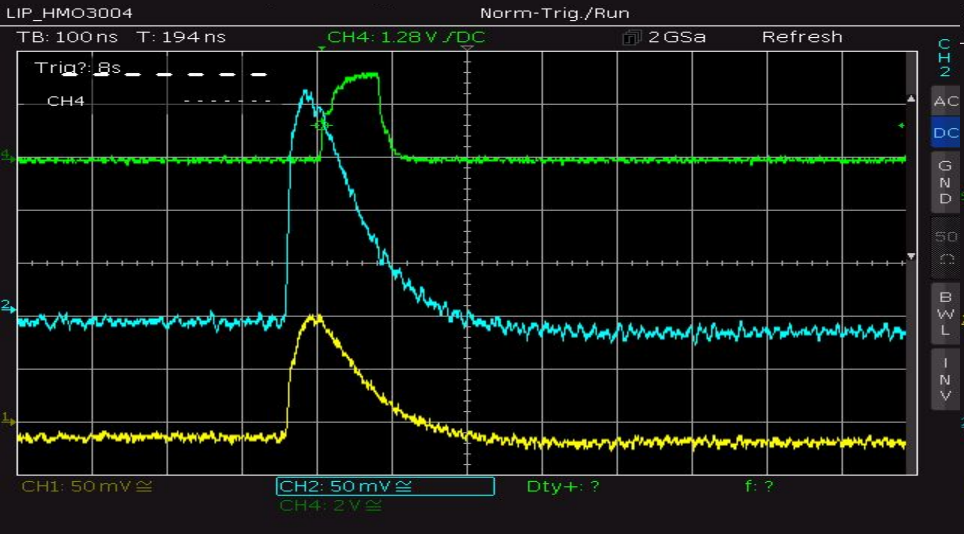


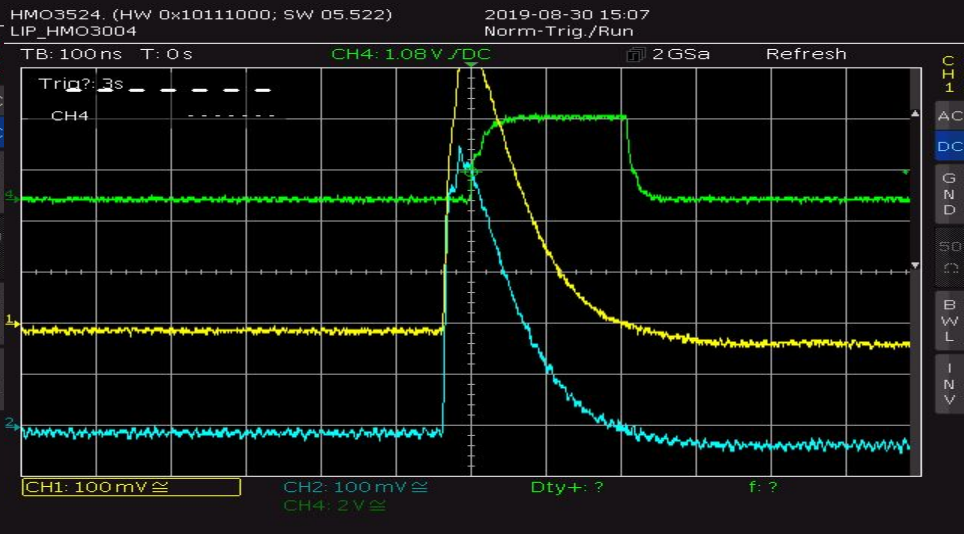



I2C
I2C
sda
scl
SDA
SCL
Master
SPI
SPI Master
SPI_MISO
miso
mosi
sclk
ss
reset
rx_interrupt
tx_interrupt
SPI_MOSI
SPI_SCLK
8-bit
CLK_INPUT_24MHz
CLK24MHz
SPI_CS
Control Reg
BUS_CLK
clock
control_0
control_1
control_2
control_3
SPI_CS_TDC0
SPI_CS_TDC1
SPI_CS_LoRa
RTC_UTC
RTC
Strobe
UART_GPS
UART
GPS_TX_PSOC_RX
rx
rx_interrupt
reset
9600 bps
UART_ESP
ESP_TX_PSOC_RX
tx
tx_interrupt
rx_interrupt
tx_en
ESP_RX_PSOC_TX
921600 bps
TRIGGER
TDC0_START
BUS_CLK
TDC0_STOP
PPS
TDC1_START
TDC1_STOP
EVENT_ISR
TRIGGER_CONTROL
DEBUG
PPS_ISR
PPS_CONTROL
CounterTimestamp
Counter
capture
tc
comp
count
interrupt
32-bit (UDB)
TFF
PulseConvert_1
Pulse Convert
p_in
p_out
out_clk
sample_clk
GPS_PPS
TDC_CLOCK
CLK_TDC
SiPM_REG
Status Reg
SiPM[7:0]
status[7:0]
CosmicBlockCounter
debug_analog
ONE_HERTZ
XTAL 32kHz / 32768
COSMIC_BLOCK_ISR
CosmicBlockReset
16-bit (UDB)
RunInfoEpoch
XTAL 32kHz
32.768 kHz
RunInfoEpochReset
SiPM0
SiPM1
SiPM2
SiPM3
SiPM4
SiPM5
SiPM6
SiPM7
LUT_1
LUT
in0
in1
in2
in3
out0
out1
out2
out3
out4
out5
out6
DT0
DT1
DT2
DT3
DT4
DT5
DT6
TriggerSelect
control[2:0]
TRIG[0]
TRIG[1]
TRIG[2]
TRIG[3]
TRIG[4]
TRIG[5]
TRIG[6]
TRIG[7]
StatusTriggerReg
TriggerSelectReset


HMO3524. (HW 0x10111000; SW 05.522)
LIP_HMO3004
2019-08-30 19:55
Norm-Trig./Run
TB: 20 ns T: 99.5 ns
CH4: 262 mV /DC
4 GSa
Refresh
716.0740 mHz
CH4
1.396504 s
CH4: 200 mV≅Ω
Dty+: ?
f: ?


HMO3524. (HW 0x10111000; SW 05.522)
LIP_HMO3004
2019-08-30 15:50
Norm-Trig./Run
TB: 100 ns T: 202 ns
CH4: 1.1 V /DC
2 GSa
Refresh
880.1217 mHz
CH4
1.162626 s
V-Marker: (CH1)
V1: -8.00 mV
V2: n/a
Δt: 1.20 µs
ΔV: n/a
CH1: 200 mV≅
CH2: 200 mV≅
CH4: 2 V≅
Dty+: ?
f: ?


LIP_HMO3004
Norm-Trig./Run
TB: 100 ns T: 194 ns
CH4: 1.28 V /DC
2 GSa
Refresh
Trig?: 8s
CH4
CH1: 50 mV≅
CH2: 50 mV≅
CH4: 2 V≅
Dty+: ?
f: ?


HMO3524. (HW 0x10111000; SW 05.522)
LIP_HMO3004
2019-08-30 15:07
Norm-Trig./Run
TB: 100 ns T: 0 s
CH4: 1.08 V /DC
2 GSa
Refresh
Trig?: 3s
CH4
CH1: 100 mV≅
CH2: 100 mV≅
CH4: 2 V≅
Dty+: ?
f: ?

Product Overview

Digi-Key Part Number	516-4280-ND
Quantity Available	**268** Can ship immediately
Manufacturer	Broadcom Limited
Manufacturer Part Number	AFBR-S4N44C013
Description	SENSOR OPT 420NM UV
Manufacturer Standard Lead Time	12 Weeks
Detailed Description	Optical Sensor Ultraviolet (UV) 420nm

Price & Procurement

Quantity 1

516-4280-ND

Customer Reference

Add to Cart

All prices are in USD.

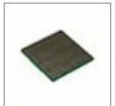
Price Break	Unit Price	Extended Price
1	26.41000	$26.41
10	23.33500	$233.35
100	17.80830	$1,780.83
500	16.33452	$8,167.26

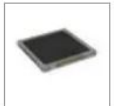
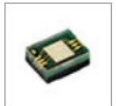
Submit a request for quotation on quantities greater than those displayed.

Documents & Media

Datasheets	AFBR-S4N44C013 Datasheet
Featured Product	AFBR-S4N44C013 NUV-HD Silicon Photomultiplier (SiPM)
Online Catalog	AFBR-S4N44C013

You May Also Be Interested In



TDC7200 (ACTIVE)

In English | Alert me

Time-to-digital converter for time-of-flight (ToF) applications for LIDAR and ultrasonic

DATASHEET

TDC7200 Time-to-Digital Converter for Time-of-Flight Applications in LIDAR, Magnetostrictive and Flow Meters datasheet (Rev. D)

View now | Download

Top purchased products for TDC7200

Part number	Buy from TI store	TI store inventory	Price | QTY	Package | Pins
TDC7200PW	Add to cart	3791	0.96 | 1ku	TSSOP (PW) | 14
TDC7200PWR	Add to cart	6236	0.80 | 1ku	TSSOP (PW) | 14

View all (2)

Description & parametrics | Technical documents | Design & development | Order now | Quality & packaging | Support & training

Description | Features | Diagram | Design resources for you | Parametrics | Complete your design

Description

The TDC7200 is a Time-to-Digital Converter (TDC) for ultrasonic sensing measurements such as water flow meter, gas flow meter, and heat flow meter. When paired with the TDC1000 (ultrasonic analog-front-end), the TDC7200 can be a part of a complete TI ultrasonic sensing solution that includes the MSP430, power, wireless, and source code.

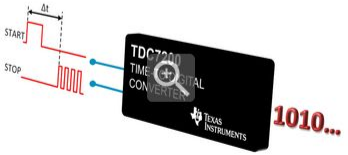
The Time to Digital Converter (TDC) performs the function of a stopwatch and measures the elapsed time (time-of-flight or TOF) between a START pulse and up to five STOP pulses. The ability to measure from START to multiple STOPs gives users the flexibility to select which STOP pulse yields the best echo performance.

View more

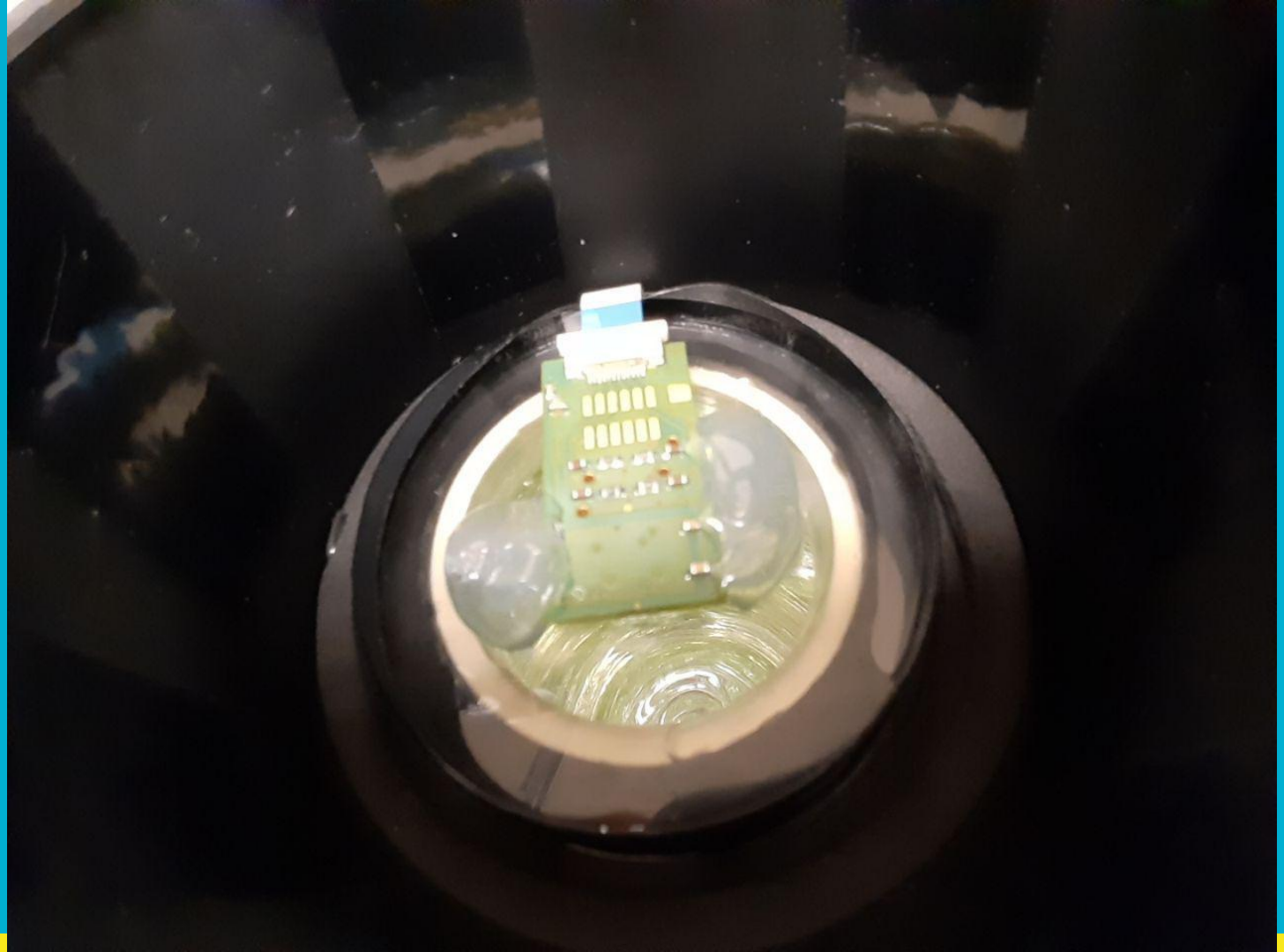
Features

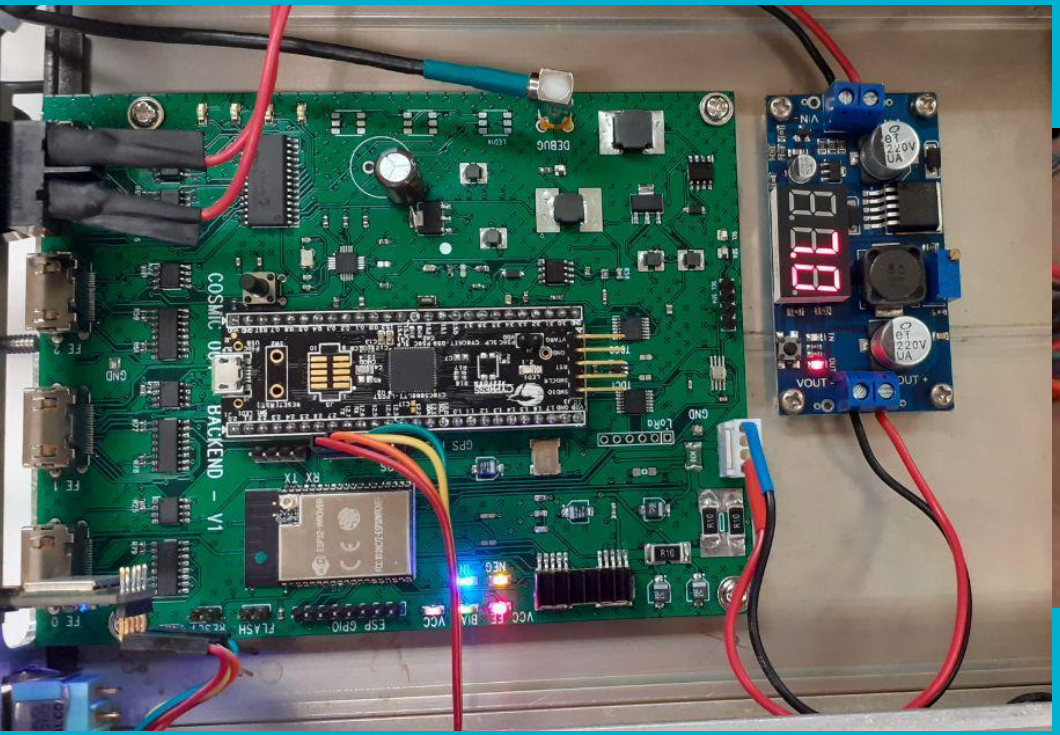
- Resolution: 55 ps
- Standard Deviation: 35 ps
- Measurement Range:
  - Mode 1: 12 ns to 500 ns
  - Mode 2: 250 ns to 8 ms
- Low Power Consumption: 0.5 µA (2 SPS)
- Supports up to 5 STOP Signals
- Autonomous Multi-Cycle Averaging Mode for Low

Diagram



TDC7200 - Functional Diagram




COSMIC UG
BACKEND – V1
DEBUG
VIN-
VOUT-
OUT+
GND
LORA
GPS
RX TX
NEG
VCC
FLASH
ESP GPIO


BK F1 F2 F3 F4 Environment Trend
Temp
22.1
Temperature
Humidity
75.4
Humidity
PPS Lost
Pressure
700.3
Pressure
L.E.T. 2000-01-01T00:00:00.000Z
SAMSUNG
DEWALT
DEWALT