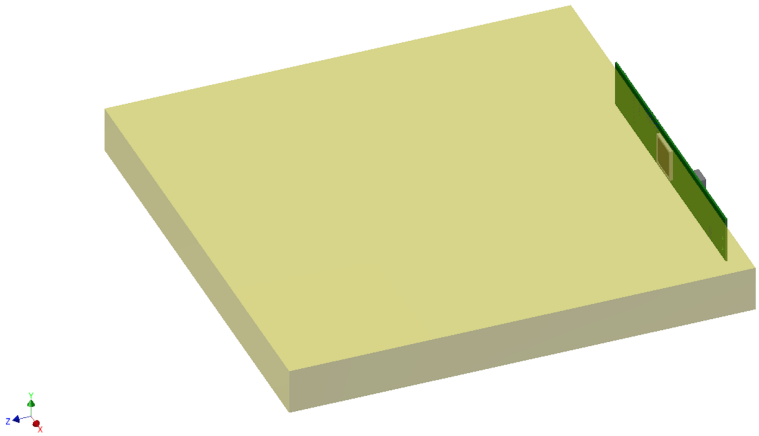
4.9 AMU-2: a cosmic muon detector

Fernando Barão, last update: 14 Fev 2024

4.9.1 Introdução

O projecto AMU-2 visa construir um telescópio de muões baseado em dois detectores de cintilação (cintiladores de plástico) com a dimensão $100 \times 100 \times 10$ mm.



A luz emitida no cintilador é lida por um photomultiplicador de estado sólido também conhecido por SiPM, colocado numa das paredes laterais do cintilador. O modelo do SiPM utilizado é o MICROFC-60035-SMT-TR1 cujas especificações podem ser consultadas no site da Farnell link O princío de funcionamento do SiPM está bem explicado neste sítio da hamamatsu: https://hub.hamamatsu.com/us/en/technical-notes/mppc-sipms/what-is-an-SiPM-and-how-does-it-work.html

Junto ao SiPM foi adicionada a electrónica de front-end [1] que visa a amplificação do sinal por um factor ~ 10.

O esquema completo da electrónica apresenta-se de seguida:

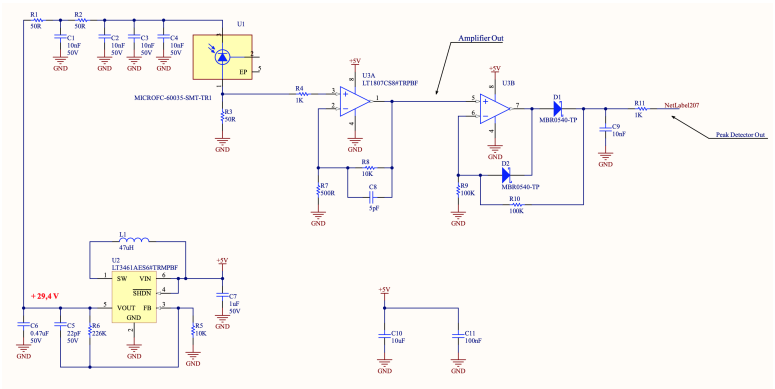


Figure 4.1: front-end electronics do SiPM para o AMU-2

A alimentação geral é feita por um transformador de $\sim 220V \rightarrow +5V$, que é utilizada para alimentar o conversor DC/DC que disponibiliza a tensão necessária

[1] LIP/Coimbra, Nuno Carolino, Luís Lopes

ao funcionamento do SiPM ($+29.4$ V) e também a tensão de alimentação do circuito electrónico, isto é, dos seus diferentes componentes.

O valor da tensão de *bias* do SiPM é controlada pela resistência R6. A expressão para calcular a tensão de saida é a seguinte: $V_{out} = 1.255(1 + R6/R5)$ V. Mantendo fixa R5 e variando R6 é suficiente para percorrer toda gama de tensões onde o SiPM tem eficiência.

O sinal de tensão à saída do SiPM é positivo e é amplificado com um factor R8/R7=20 e é extraído como *Output* por um cabo com uma ficha LEMO. A presença do condensador C8=5pF permite mais estabilidade no circuito limitando a resposta do amplificador nas altas frequências, reduzindo no entanto factor de amplificação para cerca de 10.

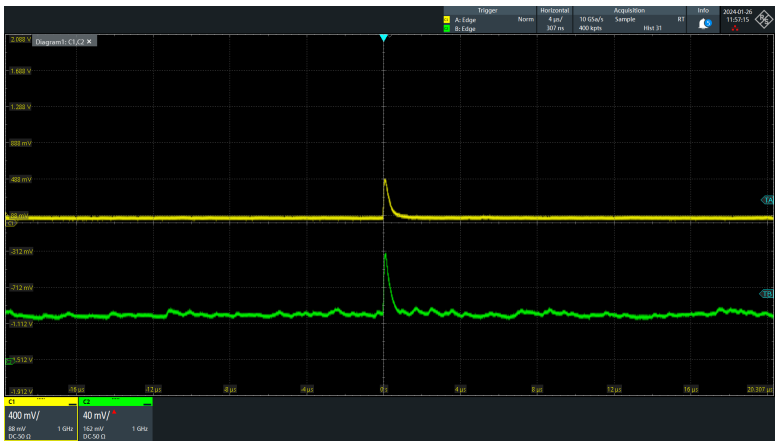
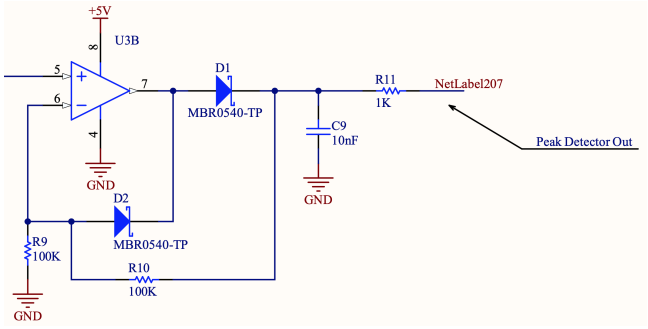


Figure 4.2: Sinal do SiPM tal como obtido após a electrónica de front-end (amplificação ~ 10)

Na electrónica de front-end existe ainda um detector de pico ("peak" detector) que visa prolongar no tempo o sinal amplificado, permitindo assim a sua amostragem por um ADC comercial, por exemplo presente no micro-controlador do tipo arduino. O sinal analógico produzido possui um tempo de decaimento dado por $\tau = (R9 + R10)C9 = 2$ msec. Este sinal não é no entanto extraído actualmente do circuito.



4.9.2 Project guidelines for student's project

O projecto a desenvolver pelos estudantes visa desenvolver um detector de muões com base nos detcetores de cintilação existentes no LabRC e descritos acima.

Etapas do projecto:

1. Desenvolver a electrónica e o sistema de aquisição do detector
2. Fazer medidas da taxa de muões ao longo do tempo
3. Fazer a análise espectral da taxa de muões
4. Fazer a medida das amplitudes dos sinais de ambos os detectores (dependendo da implementação do bloco de electrónica `peak&hold`)

Electrónica e tratamento do sinal

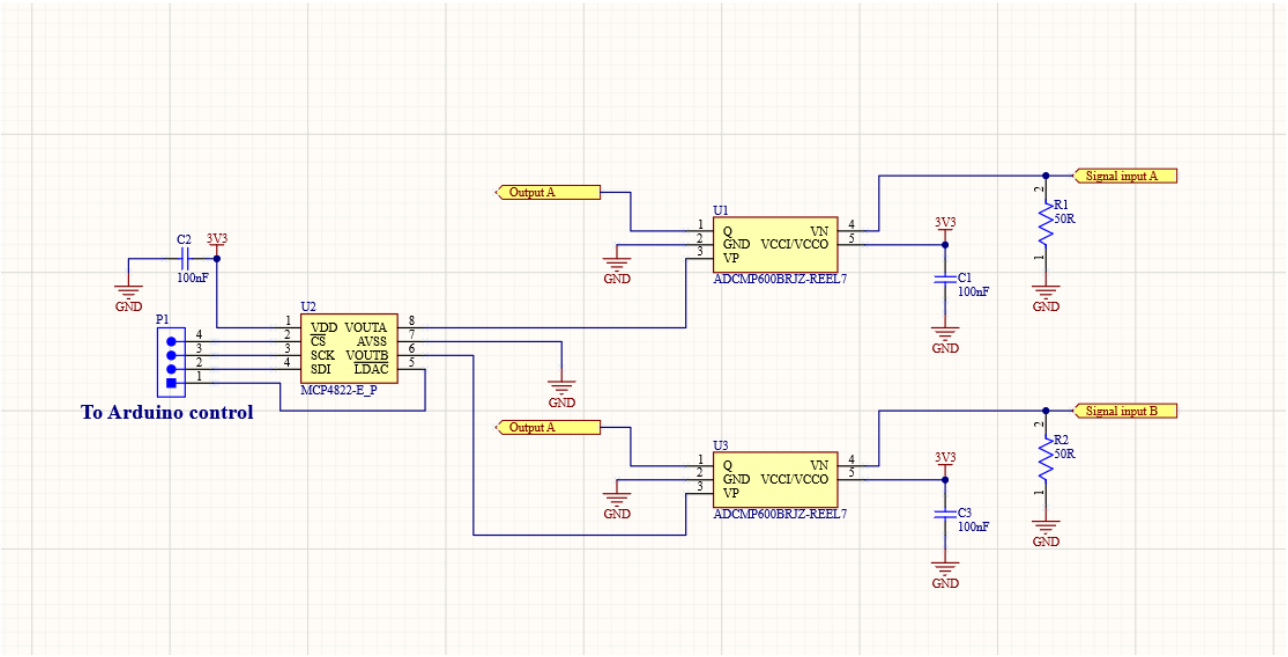
O detector AMU-2 será composto de dois detectores de cintilação que permitirão assim através de um sistema de coincidência temporal, somente adquirir os sinais correlacionados temporalmente nos dois detectores.

Como visto anteriormente, em cada detector existe já uma saída amplificada do sinal do SiPM. Necessitamos assim de implementar em breadboard ou em circuito perfurado a soldar os seguintes blocos de electrónica subsequentes:

- **bloco discriminador**

  Implementar um bloco discriminador do sinal em cada canal, que dará uma saída TTL (3.3V ou 5V) no caso do sinal estar acima de um sinal de tensão limiar. Utilizaremos o arduino para programação da tensão de limiar (*threshold*).

  O esquema de electrónica que se mostra (Miguel Ferreira / LIP), possui o bloco discriminador e as ligações ao arduino

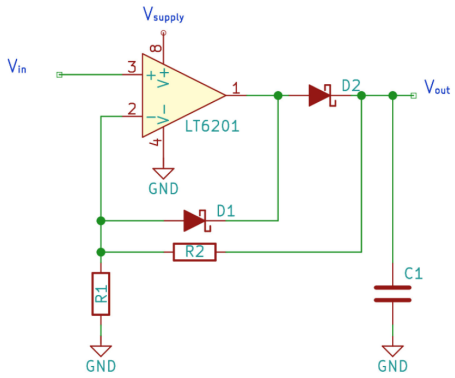


- **bloco peak&hold**

  Implementar um bloco `peak&hold` capaz de detectar o valor máximo do sinal e prolongá-lo num tempo suficiente que permita ao arduino, digitizá-lo. Necessitamos no entanto de construir o bloco de electrónica "peak & hold", cujo esquema pode ser inspirado a partir das seguintes fontes:

  - reproduzir o esquema já implementado na electrónica de front-end do detector de cintilação (esquema pode ser visto acima).
  - ver o esquema implementado no detector CosmicWatch e que pode ser consultado na página 17 e seguintes do artigo Axani, Spencer N., Janet

M. Conrad, and Conor Kirby. “The desktop muon detector: A simple, physics-motivated machine- and electronics-shop project for university students,” 2016. 85https://api.semanticscholar.org/CorpusID:92995929.



$$\tau = R \times C$$

FIG. 10: The peak detector circuit. We have selected R1 = R2 = 100kΩ and C1 = 1 nF.

- from the site of the `physicsopenlab` link, there is a peak and hold circuit

Aquisição de dados

O Arduino tem de ser programado em C/C++ para realizar as seguintes tarefas:

1. digitalizar o sinal dos dois sinais de pico dos detectores, por exemplo, nas entradas A0 e A1 com um prescale factor de 16

   determinar o valor de pico dos sinais, e definir um valor de discriminação para os aceitar como sinais candidatos de muões

2. definir a coincidência temporal dos dois sinais de forma a validar o sinal do muão

3. registar o timestamp do evento

   enviar a o valor dos picos dos sinais, timestamp, ... para o computador via USB, porta série, ...

A opção 1. está dependente da implementação do bloco `peak&hold`.

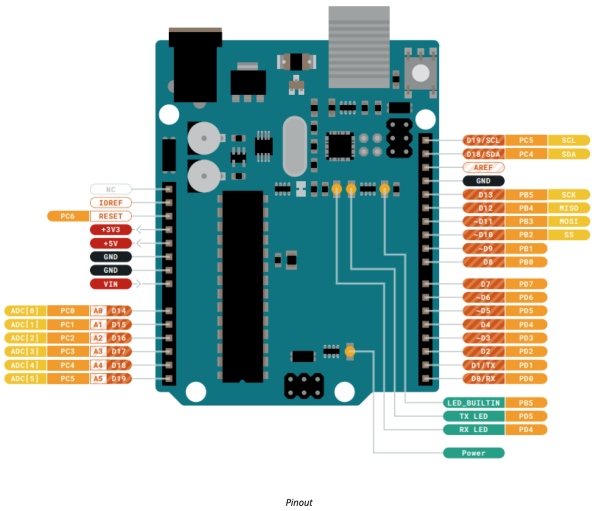
Deteção de coincidências

As saídas dos blocos discriminadores dos dois canais são canalizados para as entradas do arduino e o software pode determinar quando existe coincidência temporal, gerando assim um *timestamp* para o acontecimento e registando-o num ficheiro.

Digitalização do sinal

De seguida necessitamos digitalizar o sinal de pico a ainda de construir o sistema de coicidência temporal. Para isso podemos usar um micro-controlador como por exemplo a placa de aquisição Arduino UNO R3 que possui um processador ATMega328P. Link para a datasheet.

O pinout é o seguinte:



Pinout

O Arduino possui capacidade de digitalizar sinais analógicos (ADC) em seis entradas disponíveis: A0, A1, . . . , A5. A digitalização é realizada com o recurso a um dispositivo *Analog to Digital Converter* (ADC) de 10 bits que converte sinais analógicos no intervalo [0, +5V] num número entre [0, 2ˆ{10}-1=1023].

5.1 JANALOG

Pin	Function	Type	Description
1	NC	NC	Not connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog/GPIO	Analog input 0 /GPIO
10	A1	Analog/GPIO	Analog input 1 /GPIO
11	A2	Analog/GPIO	Analog input 2 /GPIO
12	A3	Analog/GPIO	Analog input 3 /GPIO
13	A4/SDA	Analog input/I2C	Analog input 4/I2C Data line
14	A5/SCL	Analog input/I2C	Analog input 5/I2C Clock line

O clock do Arduino é de 16 MHz. No entanto a conversão no ADC é feita a uma frequência ditada pelo factor de *prescale* que por defeito está definido no valor de 128 (`wiring.c`). Assim a frequência de conversão no ADC é: $f_{ADC} = 16\,MHZ/128 = 125\,KHz$. Há no entanto que ter ainda em conta que a conversão no ADC demora 13 ciclos de clock (confimar!), resultando por isso numa frequência máxima de amostragem de $f_{ADC} = 16\,MHZ/128/13 \simeq 9600\,Hz$. O que corresponde a um intervalo de tempo de $\Delta t = 1/9600 = 0.104\,sec$ (too long, when compared to peak and hold output time width).

Para melhorarmos este tempo de amostragem (para a ordem de 10 μsec) podemos reduzir o *prescale factor* sem grande implicação na performance da ADC. Podemos assim calcular qual o *prescale factor* (p) necessário:

$$f_{ADC}^{sample} = \frac{\text{CPU clock}}{p} \cdot \frac{1}{13} \Rightarrow \Delta t_{ADC}^{sample} = 10\,\mu sec = \frac{13\,p}{16 \cdot 10^6} \Rightarrow p = \frac{160}{13} \simeq 12$$

How to configure the prescaler?

ADPS2	ADPS1	ADPS0	prescale
0	0	0	2
0	0	1	2
0	1	0	4

ADPS2	ADPS1	ADPS0	prescale
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

The C++ code to set the prescaler:

```
// defines for setting and clearing register bits
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

void setup() {
  int start ;
  int i ;

#if FASTADC
  // set prescale to 16
  sbi(ADCSRA,ADPS2) ;
  cbi(ADCSRA,ADPS1) ;
  cbi(ADCSRA,ADPS0) ;
#endif

  Serial.begin(9600);
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);

  // test
  Serial.print("ADCTEST: ") ;
  start = millis() ;
  for (i = 0 ; i < 1000 ; i++)
    analogRead(0) ;
  Serial.print(millis() - start) ;
  Serial.println(" msec (1000 calls)") ;
}

void loop() {
}
```