# 7600061 - Introdução a Microcontroladores

Projeto 05 - Espectrômetro e display VictorVision

Vinícius de Souza Miralhas, 10728289

Este documento deve ser utilizado em conjunto com o código Arduino e um display da Victor Vision (família V, classe C, $800 \times 480$ px.; estes dados podem ser alterados no software *UnicViewAD*) e tem por objetivo detalhar o funcionamento básico do sistema, suprir uma visão geral da montagem experimental e servir de referência geral para o uso do monocromador.

## 1 O espectrógrafo

O equipamento utilizado consiste em uma lâmpada e um conjunto de sensores que detectam luz na faixa de $300 - 1000$ nm por meio de uma grade de difração, espelhos côncavos para focalizar a luz e um motor de passos para ajustar o ângulo do monocromador da leitura da intensidade do comprimento de onda selecionado.

O conjunto dispõe de duas amostras e três configurações pelas quais passa a luz da lâmpada:

1. Uma cubeta vazia/ar;
2. Uma cubeta com água e
3. Uma cubeta com neodímio diluído em água.

Em cada caso, a intensidade de *transmissão* é lida pelo Arduino. Com o auxílio de uma *linha de base* $I_0$, é possível encontrar as raias de absorção da intensidade $I$ medida por uma outra cubeta pela relação simples

$$T(\lambda) = \frac{I(\lambda)}{I_0(\lambda)} \tag{1}$$

Por exemplo, se a linha de base ($I_0$) medida for a da cubeta com água e a próxima intensidade ($I$) for a da cubeta com neodímio, a curva de transmissão permite avaliar com eficácia as raias de absorção da amostra: as quedas na transmissão correspondem aos comprimentos de onda que a amostra preferencialmente absorve.

A limitação dos comprimentos de onda de $300 - 1000$ nm decorre tanto da faixa de operação do sensor acoplado ao espectrômetro e lido pelo Arduino como das restrições mecânicas impostas pelo motor de passos. O usuário poderá selecionar qualquer faixa de comprimentos de onda contanto que os limites impostos não sejam ultrapassados.

Os dados de transmissão são salvos no cartão SD do Arduino em `.csv` e podem eventualmente ser tratados em qualquer software de plotagem/manipulação, como *RStudio* ou *OriginPro*.

Para mais informações, o canal *Oficiência* do Prof. Luiz Antônio contém vídeos de **outros** tipos de espectrógrafos usando variados sensores e monocromadores:

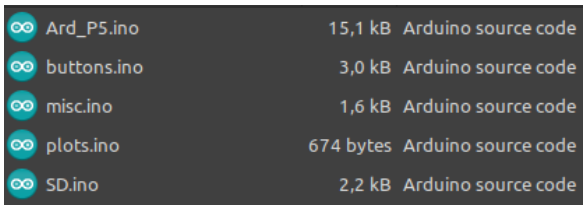
- Espectrógrafo com Sensor Linear TSL1402R
- 2_Espectroscopia de Luminescência e Excitação

# 2 Manual de uso

## 2.1 Upload preliminar

Antes de utilizar o espectrógrafo, certifique-se de fazer o upload do projeto `uvadproj` ao display da VictorVision e do código-fonte `.ino` à placa Arduino, como indicado pela fig. 1.

(a)




(b)


Figura 1: Arquivos necessários para inicializar o espectrógrafo. a) Caso esteja utilizando a Arduino IDE, inicialize qualquer um dos arquivos `.ino`: o programa é composto por todas as abas indicadas e o upload à placa automaticamente compila todos esses módulos. b) O arquivo `.uvadproj` deve ser aberto pelo UnicViewAD, compilado, transferido a qualquer dispositivo de armazenamento portátil e então injetado no display.

**Display VictorVision:**

1. Insira o dispositivo USB (pendrive, etc.) no computador e abra o arquivo `Equipamento.uvadproj`.
2. Clique no martelo que indica *compile all*.
3. Clique no botão que tem um cartão SD e um pendrive após selecionar o drive correto na caixa imediatamente à direita.
4. Copiados os arquivos, remova o dispositivo do computador. Em seguida, ligue um adaptador AC de 5V ao display. (**atenção:** o display não tolera tensões superiores a 5V.)
5. Insira o dispositivo do item 3 na entrada USB do display, ligue-o e aguarde a transferência dos arquivos. O buzzer do aparelho será acionado quando esta transferência for bem sucedida. Remova o pendrive; o display será reiniciado em seguida com o programa correto.

**Arduino Mega:**

1. Certifique-se de ajustar a placa e o microcontrolador selecionados são o Arduino Mega 2560 (fig. 2).

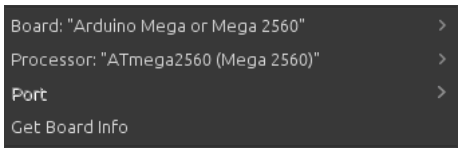



Figura 2: Configurações necessárias para o upload à placa.

2. Certifique-se de que a biblioteca `SdFat 1.1.4` está instalada no seu computador. Caso não esteja, baixe-a aqui e a instale conforme estas instruções.

3. Ligue o espectrógrafo em uma tomada de 110V. (**deve** ser de 110V.) Em seguida, faça o upload do programa à placa. Não faz diferença qual das abas está atualmente selecionada (`Ard_P5.ino`, `SD.ino`, `plots.ino`, `buttons.ino` ou `misc.ino`).

4. Feito o upload, desligue o equipamento e ligue os pinos do display na Serial TX3/RX3, seguindo as orientações dos fios.

5. Insira um cartão SD no shield acoplado ao Mega. O programa não inicializará se não houver um SD no dispositivo.

Para inicializar o programa, é suficiente ligar primeiro o display e em seguida espectrógrafo. O programa `.ino` automaticamente reiniciará o display e fará a "sincronização" handshake entre ambos. Quando tiver de reiniciar o programa, simplesmente aperte o botão de reset do Arduino; não há a necessidade de retirar e reinserir a alimentação do display, a não ser que aconteça alguma exceção não documentada.

## 2.2 Uso geral

1. Este programa pode ser utilizado sem a necessidade de conectar o cabo serial a um computador após o upload inicial. No entanto, se um Serial Monitor estiver ligado, funcionalidades extras são liberadas.

2. Quando inicializado, o programa pedirá a faixa de valores para o comprimento de onda $\lambda$. Toque nas caixas retangulares para alterar o valor de cada limite.

3. Em seguida, o motor de passos calibrará e localizará o comprimento de onda mínimo da faixa anterior. Este processo é lento.

4. Puxe ou empurre a alavanca até selecionar a cubeta necessária para a linha de base. Tipicamente, ou é escolhido o ar ou a cubeta com água. Nenhuma das outras funções estará disponível antes de medir $I_0$. Pressione o botão para medir. (fig. 3)

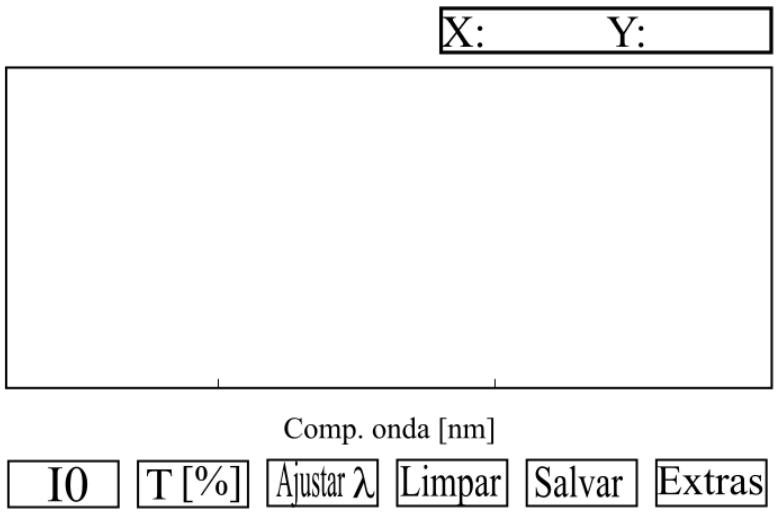



Figura 3: Funções principais do programa.

5. Enquanto o programa mede o espectro da cubeta selecionada, os valores de "x" ($\lambda$ da medição) e "y" (intensidade em unidades arbitrárias, variando de $0-1023$ *ou* transmissão percentual, variando de $0-100$) serão indicados no canto superior direito

6. Medido I0, sinta-se livre para imprimir os valores desse vetor no serial: pressione `Extras` e `Print I0`. (fig. 4)

7. Em seguida, ajuste a alavanca até a cubeta desejada e meça a curva de transmissão $T$ (eq. (1)) pressionando `T[%]`.

8. Ao final da medição, o usuário será perguntado se deseja salvar os dados. Em caso positivo, insira um nome (ou use o padrão) clicando na caixa `Nome:`. Para confirmar um nome, clique em `OK`no teclado. Por fim, salve clicando em `Salvar`.

9. Sinta-se para verificar os dados dos vetores de $I$ e $T$ na tela `Extras` (fig. 4)

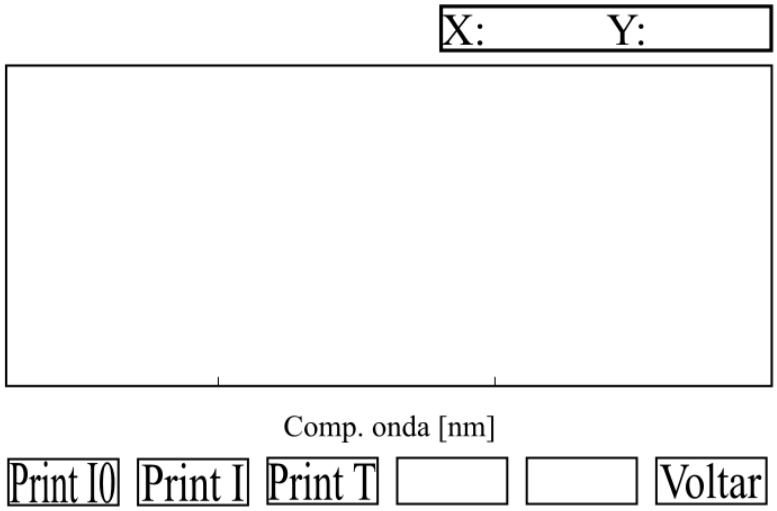



Figura 4: Funções extras do programa. Exigem que o Serial Monitor esteja aberto na Arduino IDE.

10. O programa suporta até 3 plots de $T$ simultaneamente em três cores: vermelho, azul e verde. No 4º plot, o 1º será automaticamente substituído.

11. Caso queira limpar o gráfico, pressione `Limpar` (fig. 3). Os dados atuais de $I_0$, $I$ e $T$ não serão perdido.

12. Caso tenha perdido a oportunidade de salvar a medição $T$ atual, pressione `Salvar` (fig. 4) para salvar manualmente.

13. Finalmente, para utilizar uma nova faixa de valores para o comprimento de onda, clique em `Ajustar` $\lambda$ (fig. 3). Ao redefinir essa faixa, o cursor do equipamento será navegado até o novo $\lambda$min e os valores anteriores de $I_0$, $I$ e $T$ serão imediatamente esvaziados. Portanto, salve os seus dados antes de reajustar $\lambda$.

14. Quando finalizar o uso do equipamento, desligue o espectrógrafo e remova o cabo serial (se estiver conectado) antes de remover o cartão SD para a análise dos dados.

Caso tenha de realizar alterações pontuais ao código-fonte `.ino`, sugiro que mantenha sempre um backup do programa original.

# 3 Documentação

A documentação geral pode ser consultada diretamente pelo código-fonte. Serão comentados aqui apenas os pontos principais do programa.

**Recursos extras:**

- Documentação biblioteca UVAD.ino. Baixe também o repositório de exemplos. *Referência Geral* e *Leitura de Sensores* são especialmente úteis.
- Manual de usuário UnicView AD e Projetos Demo. Utilize os projetos demo para ter uma noção de como usar o Unic View AD diretamente.
- Tutoriais em vídeo:
  - Tutoriais Victor Vision
  - Treinamento Unic View AD

## 3.1 Forçando o reinício do Display

```
1  void setup()
2  {
3    Serial.begin(9600);
4    Serial3.begin(115200);
5    Lcm.begin();
6    t_i = millis();
7    //*** Garantir o reinício completo do display
8    // Por um total de 1s, força o reinício do display.
9    // Isto contorna o problema do Lcm.write recebendo dados indevidos.
10   while (millis() - t_i < 1000) {
11     Lcm.resetLCM();
12     Lcm.changePicId(0);
13   }
14   //***
15 ...
```

Definindo uma variável `t_i` do tipo `unisgned long`, é possível travar o início do display por um total de 1s. Esse intervalo foi estabelecido por testes. A função `Lcm.resetLCM()` é constantemente acionada até que o display finalmente receba a função. `Lcm.changePicId(0)` tenta trocar a tela do display para a tela inicial imediatamente após o reset.

Um delay entre `Serial3.begin(115200)` e `Lcm.begin()` não foi suficiente para resolver o problema, mas a rotina descrita sim.

## 3.2 "Interpolação" de ordem zero

Talvez o leitor esteja familiarizado com o conceito de interpolação linear ou similar. Defino *interpolação de ordem zero* uma aproximação simples de índices quando o total de medições é diferente do total de pontos exigidos pelo plot. O objeto *Trend Curve Display* do UnicView AD tem um incremento horizontal (em px.) fixo: o total de pontos a serem plotados é fixo, portanto. No código-fonte, dado que o passo mínimo é fixo, a quantidade de medições cresce com a faixa de $\lambda_{min}$ a $\lambda_{max}$.

Ou seja, é possível que o TrendCurveDisplay aguarde por 341 pontos enquanto tenham sido medidos apenas 43, por exemplo. Como contornar este problema?

Suponha uma situação mais simples: 10 pontos medidos, 18 pontos plotados. Imaginei a seguinte forma de simplificar o problema: se $i = 0, \ldots 9$ é o índice do array que armazena os pontos medidos e $j = 0 \ldots 17$ é o índice do ponto a ser plotado, é nítido que a $i-$ésima medição

talvez tenha de ser plotada mais de uma vez para atingir os 18 pontos exigidos. Impondo relação linear entre $i$ e $j$, chega-se a:

$$\frac{i}{j} = \frac{\Delta i}{\Delta j} = \frac{9}{17} = \frac{N_{\text{med}} - 1}{N_{\text{plot}} - 1} \tag{2}$$

Por exemplo, o $j = 13$ corresponde a $i = 6.88$, que pode ser arredondado para 7. Portanto, o $14^{\text{o}}$ $(13 + 1)$ ponto do plot é aproximado pela $8^{\text{o}}$ $(7 + 1)$ medição. Isto é o que defino de interpolação de ordem zero, por ser essencialmente um algoritmo de busca. A relação completa entre os índices pode ser consultada pela tabela 1.

| $j$ | $i_{\text{eff}}$ | $i_{\text{round}}$ |
|---|---|---|
| 0 | 0,00 | 0 |
| 1 | 0,53 | 1 |
| 2 | 1,06 | 1 |
| 3 | 1,59 | 2 |
| 4 | 2,12 | 2 |
| 5 | 2,65 | 3 |
| 6 | 3,18 | 3 |
| 7 | 3,71 | 4 |
| 8 | 4,24 | 4 |
| 9 | 4,76 | 5 |
| 10 | 5,29 | 5 |
| 11 | 5,82 | 6 |
| 12 | 6,35 | 6 |
| 13 | 6,88 | 7 |
| 14 | 7,41 | 7 |
| 15 | 7,94 | 8 |
| 16 | 8,47 | 8 |
| 17 | 9,00 | 9 |

Tabela 1: Colunas para o $j-$ésimo ponto do plot, a correspondente $i-$ésima medida puxada (exata) e a aproximada, respectivamente.

Portanto, se o vetor contendo as 10 medidas já for conhecida, é suficiente varrer um loop por $j$ e optar por plotar o $i_{\text{eff}}-$ésimo elemento medido. Mas e se o objetivo for plotar enquanto os pontos são medidos? Para isso, deve-se saber quantas vezes um dado $i$ deve ser plotado. Uma forma óbvia de fazer isso seria varrer $j = 0, \ldots, N_{\text{plot}} - 1$, multiplicar cada $j$ pelo fator de equivalência $\frac{N_{\text{med}}-1}{N_{\text{plot}}-1}$ e contar quantas vezes o $i$ atual foi mapeado nesse processo. Em seguida, é suficiente plotar o ponto $i$ atual essa mesma quantidade de vezes.

Código Arduino:

```
int contagem = 0;
for (int j = 0; j < plotPoints; j++) {
  if (round(j * conversion) == imed) contagem++;
}

for (int j = 0; j < contagem; j++) {
  int x, y;
  ...
  switch (currentChannel) {
    case 0:
      Lcm.writeTrendCurve0(y);
      break;
```

```
  ...
```

`Lcm.writeTrendCurve0(y)` plota o y atual (ou seja, a medição atual, a menos de fatores multiplicativos para determinar se é I0 ou T) pela quantidade de vezes que o $i$ atual é contabilizado ao varrer $j = 0, \ldots, N_{plot}$. Comentários adicionais estão no código-fonte.

## 3.3 Ler números e textos do display

Vídeo auxiliar para esta subseção.

Para ler números e strings do display para o Arduino, crie um Numeric/Text Input com um determinado ValuePointer no software UnicView AD. Altamente recomendo que essas variáveis sejam espaçadas de 10 em 10 no mínimo em caso de Numeric Input e de pelo menos o comprimento da string no caso de Text Input.

Exemplo: em TextInput, se uma caixa de TextInput tiver Text Length de 10, o total de caracteres armazenados será o dobro disso (vide vídeo): 20. Se o VP deste objeto for 300, inicie o VP do próximo objeto (Numeric Input, Text Input, etc.) em pelo menos 320. Se não fizer isso, as strings serão escritas em um só objeto ao invés de dois objetos separados.

**Numeric Input:**

```
// Na definição de vars globais:
LcmVar lcmLambdaMinInput(502); // VP: 502
int lambda_min;
...
// Em adjustLambda():
if (lcmLambdaMinInput.available()) {
  lambda_min = lcmLambdaMinInput.getData();
  ...
```

**Text Input:**

```
// Na definição de vars globais:
LcmString fileNameInput(402, 16); //VP: 402

// Em saveToSDScreen():
String fname = "trans.csv";
...
if (fileNameInput.available()) {
  fname = "";
  while (fileNameInput.available()) {
    fname += (char)fileNameInput.getData();
  }
...
```

Para usar Text/Numeric input, uma tela de teclado é necessária. Use o exemplo *Referência Geral* para extrair os teclados necessários para cada um.

## 3.4 Usando SetValue

Similar a Text/Numeric Input, mas o toque simplesmente fixa um valor para a variável, não surge uma tela de teclado. É ideal para mapear funções de botões. A discussão de VP da subseção anterior continua válida.

```
...
// Nas definições de vars globais:
LcmVar plotButton(100);
...
// Em loop():
  if (plotButton.available()) {

```

```
    byte opcao = plotButton.getData();
    Serial.println(opcao);
    switch (opcao)
    {
      // Botão 1: Lê e plota I0
      // Limpa o TrendCurve. Insere os limites corretos. Mede e plota
   simultaneamente.
      case 1:
      ...
```

## 3.5 TrendCurveDisplay

Vídeo auxiliar para o TrendCurveDisplay Cada objeto TrendCurveDisplay admite uma única curva. Para dar a impressão de três curvas simultâneas, sobrepus três objetos TrendCurveDisplay sobre as mesmas coordenadas no UnicView AD. Cada TrendCurveDisplay tem um canal, então usei o Arduino para fazer um ciclo sobre o canal atual, incrementando uma unidade sempre que `T[%]` for chamada. TrendCurveDisplays admitem apenas valores inteiros, floats são arredondados para int.

**Total de pontos do plot:**

Meça a extensão horizontal do TrendCurveDisplay, em pixels. Definido o seu HorizontalIncrement, o total de pontos plotados para ocupar todo o plot é dado por

$$\Delta(\text{pixels})/\text{HorizontalIncrement} + 1.$$

Por exemplo, neste projeto, todas as TrendCurveDisplays têm extensão horizontal de 680 pixels e HorizontalIncrement de 2 pixels. Total de pontos: $340 + 1 = 341$.

**Escrevendo pontos no plot:** Defina uma escala prévia para os valores que o plot receberá. De preferência, use uma faixa fixa geral para todas as aplicações, como $0 \sim 1000$. Defina no Arduino a conversão entre a quantidade real e aquela que será plotada.

Exemplo: a faixa real de valores para a quantidade é $-680 \sim 100$. O valor $-340$ é mapeado para $435.897 = 436$.

Neste exemplo, use VerticalZoom como $1000 - 0 = 1000$ ao invés de $100 - (-680) = 780$.

Quando for acionar a função `Lcm.writeTrendCurve0(y)`, `Lcm.writeTrendCurve1(y)`, etc., certifique-se de que o seu $y \neq 0$. Se for $y = 0$, o ponto simplesmente não será plotado, será ignorado. Portanto, se 30 dos 341 pontos a serem plotados forem nulos, apenas 311 pontos serão plotados. Para evitar isso, certifique-se de que $y = 0$ é redefinido para $y = 1$, uma diferença imperceptível no plot se o seu VerticalZoom for alto.