Universidade Federal do ABC

Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas

PPG EBM107 – Tópicos Avançados em Instrumentação Biomédica
2º quadrimestre de 2019

Roteiro 07:
Criando interfaces gráficas na RaspberryPi

1 Objetivo

Neste roteiro iremos criar interfaces gráficas na RaspberryPi, e utilizá-las para mostrar, em tempo real, medidas de amplitude de um sinal senoidal utilizando um Arduino conectado a uma porta serial (via USB) da RaspberryPi.

2 Materiais necessários

Neste roteiro você irá precisar de:

- 1 placa RaspberryPi com Raspbian Desktop instalado;
- 1 placa Arduino Uno com cabo USB;
- 1 gerador de onda;
- 1 placa de condicionamento de sinal;
- 1 protoboard;
- 1 fonte de alimentação simétrica.

3 Introdução

A RaspberryPi possui um processador poderoso que, quando combinado com o sistema operacional Linux, é capaz de fornecer interfaces gráficas sofisticadas. Interfaces gráficas permitem diversas interações com o usuário que não são possíveis com interfaces por linha de comando. Existem diversas estruturas de desenvolvimento de aplicações gráficas disponíveis na RaspberryPi, como GTK+ e Qt. Essas estruturas fornecem bibliotecas de componentes visuais que podem ser combinadas para criar aplicações com considerável nível de interação com o usuário.

A RaspberryPi pode ser conectada diretamente a um display (LCD touchscreen, monitor, televisão etc), ou pode ser acessada remotamente para permitir interfaces gráficas com o usuário. Existem basicamente quatro arquiteturas de hardware de interface gráfica com o usuário que podem ser implementadas com uma RaspberryPi:

Computação de uso geral: conectando a RaspberryPi a um monitor pelo conector HDMI, e usando um teclado e mouse USB, ela pode ser usada com um computador de uso geral. Assim temos uma plataforma de baixo custo e baixo consumo. Entretanto, essa abordagem requer um monitor dedicado. Além disso, a RaspberryPi não possui a capacidade de processamento de um computador pessoal;

Display LCD touchscreen: conectando um LCD touchscreen ao conector GPIO da RaspberryPi, ela pode ser usada como um dispositivo *stand-alone*. O sistema todo pode ser alimentado por bateria, e existem diversos tamanhos de display disponíveis no mercado. Entretanto, esses displays são caros e de resolução modesta;

***Virtual network computing* (VNC):** através de um acesso remoto e software de controle, é possível controlar interfaces gráficas em um display virtual. Também permite a alimentação por bateria, mas requer um computador ou tablet e uma conexão de rede, o que pode deixar a interface lenta;

Aplicações remotas *fat-client*: usando programação customizada cliente/servidor com uma RaspberryPi conectada a uma rede, é possível criar interfaces remotas que trocam mensagens com a RaspberryPi (servidor), enquanto a maior parte do processamento fica com o computador remoto (cliente). A RaspberryPi pode ser alimentada por bateria, e apresenta um baixo consumo, já que parte do processamento é feito remotamente. Entretanto, requer desenvolvimento de uma aplicação específica, usando, por exemplo, soquetes TCP, e de um dispositivo remoto para rodar a aplicação do lado do cliente.

4 Qt

O Qt é uma estrutura de desenvolvimento multi-plataforma que usa a linguagem de programação C++. Ele fornece bibliotecas de códigos C++ para desenvolvimento de interfaces gráficas e acesso a banco de dados, tratamento de *threads*, rede, entre outros. O código desenvolvido em Qt pode ser executado em Windows, Linux, Mac OS X, Android, iOS, e em muitas plataformas embarcadas, como a RaspberryPi.

Uma interface gráfica construída no Qt é formada por widgets. Um widget é a unidade básica de um programa com interface gráfica: ele pode ser uma janela, um botão, um menu, um ícone, uma barra de rolagem, um gráfico etc. Trabalhar com vários widgets é a principal diferença entre um programa gráfico e um programa normal. No último caso, o programador procura sempre saber qual ponto do programa está sendo executado; no programa gráfico, é o sistema gerenciador de janelas que escolhe qual parte do programa (ou seja, qual parte de qual widget) está sendo rodado.

Uma ótima referência para aprender o básico da programar em Qt pode ser encontrada em https://pt.wikibooks.org/wiki/Qt.

5 Coletando dados do Arduino

Podemos escrever um programa que apresente dados coletados na porta A0 do Arduino. Para isso, vamos criar um programa no Arduino que leia 8 pontos de um sinal senoidal de $19225Hz$ a uma taxa de $153,8kHz$ (um ciclo) na porta A0 e transmita a amplitude desse sinal pela porta serial:

Listing 1: roteiro07/le_ampl_serial/le_ampl_serial.ino

```
1  /* Programa que mede um sinal senoidal a 19225Hz, calcula a amplitude
2   *  pico-a-pico do sinal e envia o valor calculado pela porta serial.
3   */
4  typedef union {
5      float floatingPoint;
6      byte binary[4];
7  } binaryFloat;
8
9  float piE[] = {0.000000, 0.176777, 0.250000, 0.176777, -0.000000, -0.176777, -0.250000, ...
       ↪ -0.176777, 0.250000, 0.176777, -0.000000, -0.176777, -0.250000, -0.176777, ...
       ↪ 0.000000, 0.176777};
10 int numSamples = 0;
11 byte medida[8];
12
13 void setup(){
14    // Configura a serial: baud rate de 115200, 8-bit, sem paridade, 1 stop bit
15    Serial.begin(115200, SERIAL_8N1);
16    ADCSRA = 0;             // clear ADCSRA register
17    ADCSRB = 0;             // clear ADCSRB register
18    ADMUX |= (0 & 0x07);    // set A0 analog input pin
19    ADMUX |= (1 << REFS0);  // set reference voltage
20    ADMUX |= (1 << ADLAR);  // left align ADC value to 8 bits from ADCH register
21    ADCSRA |= (1 << ADPS1) | (1 << ADPS0);    // 8 prescaler for 153.8 KHz
22    ADCSRA |= (1 << ADATE); // enable auto trigger
23    ADCSRA |= (1 << ADIE);  // enable interrupts when measurement complete
```

```
24     ADCSRA |= (1 << ADEN);  // enable ADC
25     ADCSRA |= (1 << ADSC);  // start ADC measurements
26  }
27
28  ISR(ADC_vect)
29  {
30    medida[numSamples] = ADCH;  // read 8 bit value from ADC
31    numSamples++;
32    if( numSamples == 8 ) {
33      ADCSRA &= ~(1 << ADSC);  // stop ADC measurements
34      ADCSRA &= ~(1 << ADEN);  // disable ADC
35    }
36  }
37
38  void loop(){
39    int i;
40    float seno, cosseno;
41    binaryFloat amplitude;
42    if( numSamples == 8 ) {    // fim da coleta
43      numSamples = 0;
44      seno = 0;
45      cosseno = 0;
46      for (i = 0; i < 8; i++){
47        seno += (float)medida[i]*(float)piE[i];
48        cosseno += (float)medida[i]*(float)piE[i+8];
49      }
50      amplitude.floatingPoint = 2*sqrt(sq(seno)+sq(cosseno))*5.0/255;
51      Serial.write(amplitude.binary,4);
52      delay(100); // aguarda 100ms e inicia nova coleta
53      ADCSRA |= (1 << ADEN);  // enable ADC
54      ADCSRA |= (1 << ADSC);  // start ADC measurements
55    }
56  }
```

O programa está escrito de modo que serão enviados aproximadamente 10 valores de amplitude pico a pico por segundo, uma vez que, ao final da função `loop()`, existe um atraso de $100ms$.

Para calcular a amplitude do sinal, esse programa usa o método da demodulação por quadratura, visto na aula 1. Para isso, é necessário o cálculo de uma matriz pseudo-inversa, o que ser feito com o exemplo a seguir:

Listing 2: roteiro07/calcula_pseudo_inversa.m

```
1  clear all;
2  close all;
3  clc;
4  more off;
5
6  freqAmostragem = 153800       % frequencia de amostragem (PS_8 no Arduino)
7  freqSinal = 19225             % frequencia do sinal
8  numeroMaximoDeAmostras = 8;   % numero de pontos coletados
9
10 periodoAmostragem = 1.0/freqAmostragem;
11 periodoSinal = 1.0/freqSinal;
12 pontosPorCiclo = periodoSinal/periodoAmostragem
13 numeroDeCiclos = floor(numeroMaximoDeAmostras/pontosPorCiclo)
14 vetorTempos = 0:periodoAmostragem:periodoSinal*numeroDeCiclos;
15
16 seno = sin(2*pi*freqSinal*vetorTempos(1:end-1));
17 cosseno = cos(2*pi*freqSinal*vetorTempos(1:end-1));
18
19 E = [seno' cosseno' ones(length(vetorTempos(1:end-1)),1)];
20 piE = pinv(E)
```

5.1 PyQt (Python)

Também é possível utilizar o Qt em Python, utilizando PyQt, que é um empacotador da linguagem Python para a biblioteca Qt.

Para instalar os pacotes necessários, rode os seguintes comandos:

```
sudo apt install python3-pyqt5
sudo apt install pyqt5-dev-tools
```

```
sudo apt install python3-pyqtgraph
sudo apt install python3-serial
```

O programa em Python que conversa com o Arduino e mostra os valores enviados por ele na tela pode ser visto a seguir:

Listing 3: roteiro07/exemplo_pyqtgraph.py

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import sys
from pyqtgraph.Qt import QtGui, QtCore
import pyqtgraph as pg
import numpy as np
import serial
import atexit
import struct

win = pg.GraphicsWindow()
win.setWindowTitle('Amplitude medida')
p1 = win.addPlot()
data1 = np.zeros(300)
curve1 = p1.plot(data1, pen='r')
ptr1 = 0
ponto = 0

conexaoSerial = serial.Serial('/dev/ttyUSB0',115200)

def saindo():
    print('Saindo')

def update():
    global data1, curve1, ptr1, conexaoSerial, ponto
    if conexaoSerial.inWaiting() > 1:
        ieee754_data = conexaoSerial.read(4)     # le 4 bytes da serial
        dado = struct.unpack('f', ieee754_data)  # converte para float
        data1[ponto] = dado[0];                  # atualiza vetor do grafico
        ponto = ponto + 1
        if ponto > 299:
            ponto = 0
        curve1.setData(data1)                    # atualiza grafico

# inicia timer rodando o mais rápido possível
timer = QtCore.QTimer()
timer.timeout.connect(update)
timer.start(0)

# registra funcao de saida
atexit.register(saindo)

## Start Qt event loop
QtGui.QApplication.instance().exec_()
```

6 Atividades para a aula

Ligue seu Arduino na RaspberryPi usando um cabo USB e grave no Arduino o programa da listagem 1.

Em seguida, utilizando a placa de condicionamento de sinal montada pelo seu grupo, ligue o gerador de onda na entrada analógica `A0` do Arduino, da seguinte forma:

- **+12V**: deve ser ligado ao pino positivo da fonte de alimentação;
- **+5V**: deve ser ligado ao pino de 5V do Arduino;
- **GND**: deve ser ligado ao pino **GND** Arduino, ao terra da fonte de alimentação e ao terra do gerador de sinal;
- **-12V**: deve ser ligado ao pino negativo da fonte de alimentação;

- **Vout**: deve ser ligado ao pino de sinal do gerador de onda;
- **I**: deve ser deixado desconectado.

Agora, ajuste o gerador de onda para uma frequência de $19225Hz$, amplitude pico a pico de $1V$, e sem *offset*, e verifique, com o programa da listagem 3, o valor medido pelo arduino. Repita para diferentes valores de amplitude entre 0 e $5V$.

Referências Bibliográficas

- MOLLOY D., Exploring Raspberry Pi: Interfacing to the real world with embedded linux, Wiley,2016.
- Qt Serial Port - Qt Wiki (https://wiki.qt.io/Qt_Serial_Port).
- QCustomPlot (https://www.qcustomplot.com/).
- Plotting in Qt using QCustomPlot (https://www.youtube.com/watch?v=xWGEvlDWokQ)