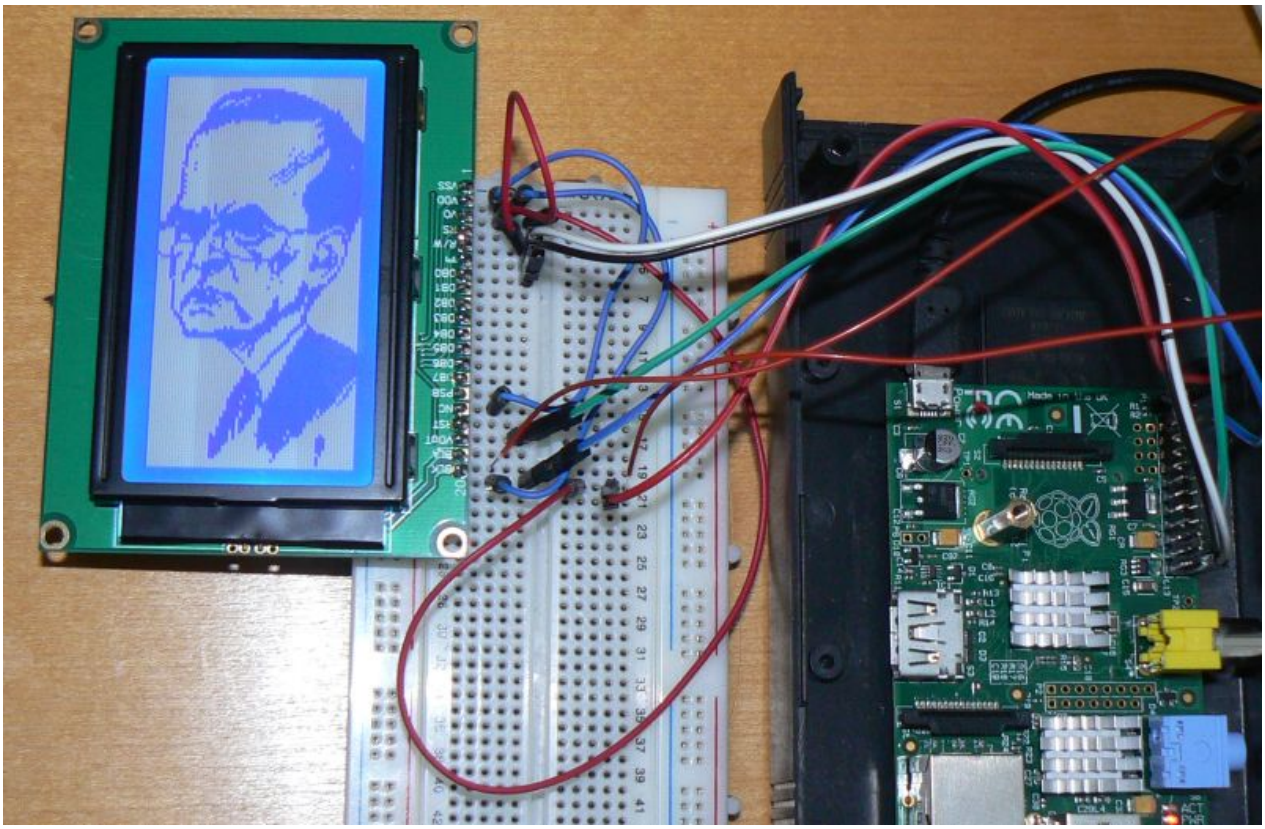
Raspberry Pi

42) Control of graphic displays

Part 4: 128x64 dot display with ST7920 controller (12864ZW designation)



I got another graphical display to get started.
This time it is a commercially available display with the ST7920 controller, which is designated 12864ZW.
I'm not quite sure, but it's probably this display: http://www.ebay.com/itm/330809490475

The datasheet is here: http://www.digole.com/images/file/Digole_12864_LCD.pdf

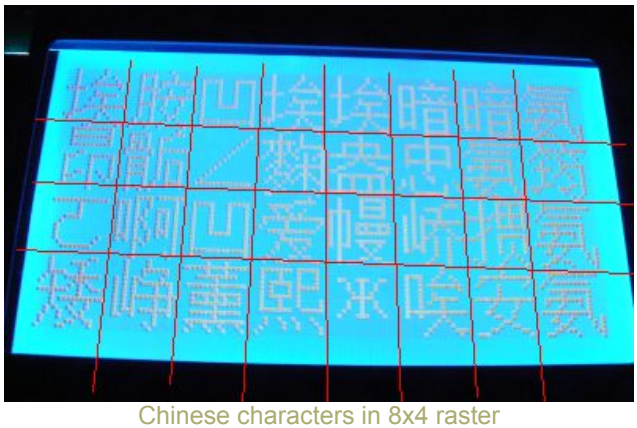
This display can be controlled by serial and parallel communication. In order to save GPIO pins in RasPi, I chose serial communication option.

Basic description of display modes

The display can work in two modes - **text** and **graphics** .

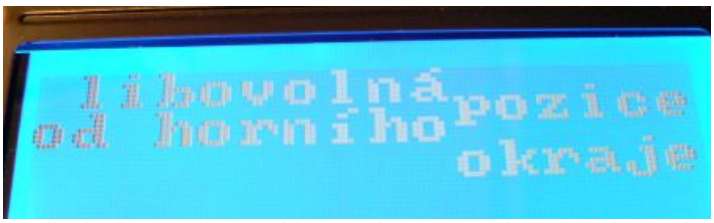
After switching to **text mode** , the display can show "8x16" European characters or 16x16 "Chinese" characters. In this mode it is also possible to define and display 4 custom "icons" of 16x16 points.
In text mode, the characters appear on the display at the specified positions, so that a maximum of 16 columns and 4 lines for "European" inscriptions can fit on the display, or 8 columns and 4 lines for "Chinese" inscriptions and custom icons. Cannot display text outside this bitmap.
In text mode it is not possible to print characters with Czech diacritics (hooks / commas) - they are not defined in the internal ROM display.

"European" characters in 16x4 raster



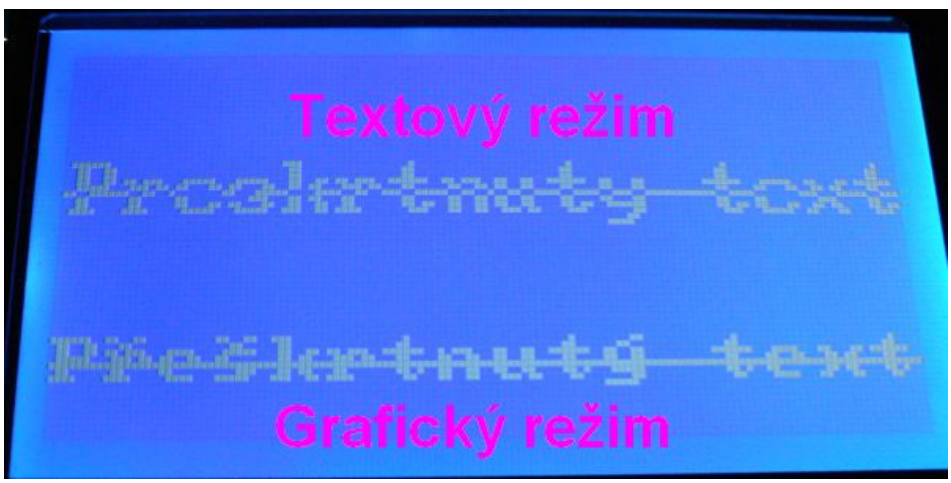
Chinese characters in 8x4 raster

After switching to **graphics mode** it is possible to control any point on the display.
If you define a font (graphic appearance of individual letters) in a file, you can
display text signs even in graphics mode . However, the disadvantage of this mode is slow speed.



When displaying text in graphics mode, text can be printed anywhere.
In the example control program I used a font of 8x8 points and for simplicity I
left the X position in the original raster, so it is possible to change only the Y coordinate by individual pixels.

Both of these modes can be switched on simultaneously. So it is possible, for example, to create a frame using graphics mode and write text into it using text mode.
When using both modes at the same time, it is important to note that when the luminous point in text mode and the luminous point in graphic mode overlap, the result in the display goes out (it is XOR function). It is therefore not possible to type some text in text mode and then delete it with a graphic line.
Here's an example of how the strikethrough sign would look bad compared to the strikethrough sign displayed in graphics mode:

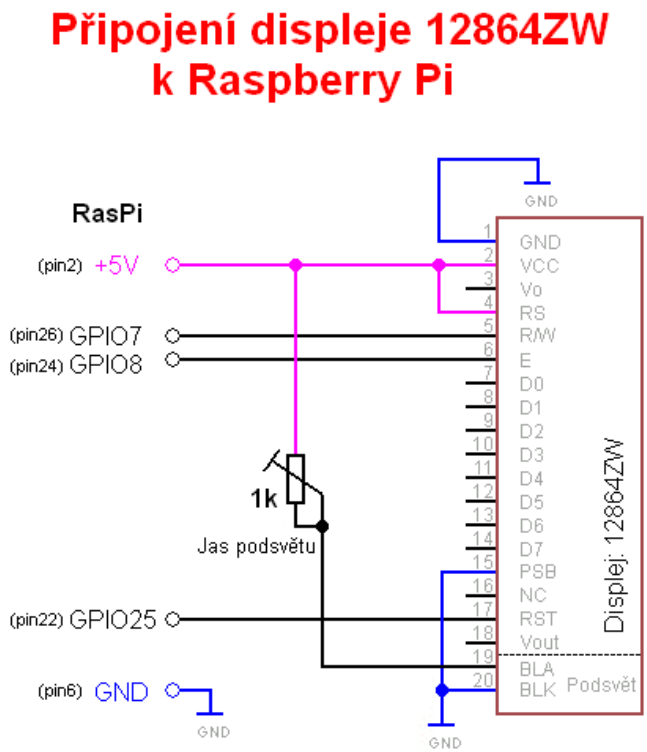


Connection of terminals according to data sheet:

Lead	Importance
1	Vss - GND
2	power supply 5V
3	VO - probably used to adjust the contrast, but the pin is unconnected
4	RS - for serial communication serves as "ChipSelect" - for "1" the display receives data
5	R / W - serves as data input (SID) for serial communication
6	E - serves as a clock input (SCLK) for serial communication

7	D0 - parallel data
8	D1 - parallel data
9	D2 - parallel data
10	D3 - parallel data
11	D4 - parallel data
12	D5 - parallel data
13	D6 - parallel data
14	D7 - parallel data
15 Dec	PSB - switching serial or parallel communication (for serial connect to GND)
16	NC - not connected
17	RST - external reset. After a brief "0", the display will reset.
18	Vout - Reportedly there should be some voltage to adjust the contrast, but there is nothing
19 Dec	Underground anode (LED consumption is about 60mA when powered directly from + 5V)
20 May	Underworld cathode - GND

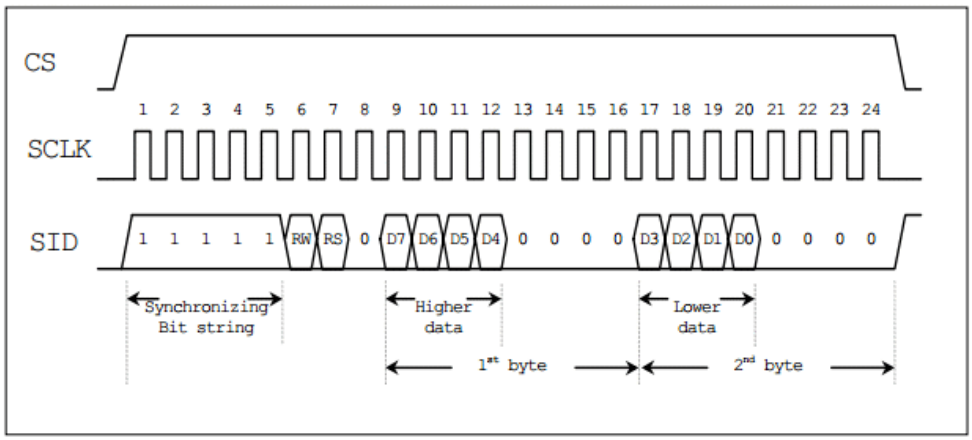
Connection scheme:



You can select GPIO pins as needed, but in that case you need to change their settings in the program.

Software:

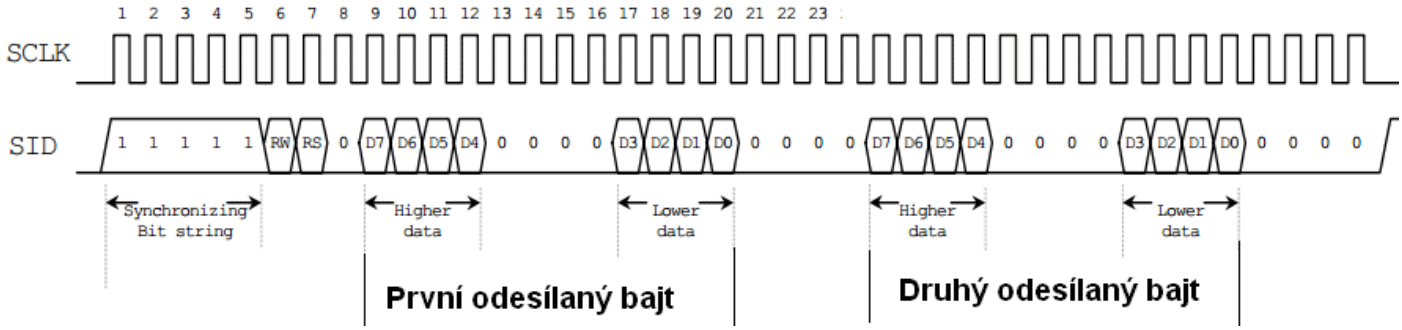
The basis of display management is the following communication description:



Timing Diagram of Serial Mode Data Transfer

This sends one byte to the display.

In some cases, you need to send two bytes immediately. I sent them like this:



Sample Python Script: glcd12864zw.py

To do this, you need a font (the same as in the previous graphical display control articles): font2.txt And another sample bitmap image: pokladnik.bmp (image taken from: www.karikaturist .estranky.cz / and reduced to 128x64 points)

Download all 3 files to **/ home / pi /** and run the script.

Here is an example of script activity:

Všechny podprogramy jsou jako obvykle vysvětleny přímo v kódu, tady je jen jejich krátký popis:

init()	Základní nastavení GPIO portů - stačí ho spustit jen 1x na začátku

		programu
init_text()		Přepne displej do textového režimu (obsah displeje nemaže)
init_grafika()		Přepne displej do grafického režimu (obsah displeje nemaže)
clr_text()		Smaže obsah textové části displeje (grafická část zůstává nezměněná)
clr_grafika(pattern)		Vyplní celý obsah grafické části displeje zadaným bajtem. (Při 0x00 displej smaže, při 0xFF ho celý vyplní bílými body, další hodnoty způsobí vyplnění displeje různými svislými čarami). Textová část displeje zůstává nezměněná.
disclear(pattern)		Provede oba předchozí způsoby smazání zároveň. Po návratu je displej přepnutý do textového režimu.
defikon(ikona,ikodata)		Nadefinuje jednu ze čtyř vlastních ikon ikona = číslo ikony 0 až 3 ikodata = proměnná, která obsahuje pole: 16x dvoubajtová hodnota
printiko(cislo , x , y)		Na souřadnice [x,y] vytiskne jednu ze 4 vlastních ikon. cislo = číslo ikony 0 až 3 x = sloupec 0 až 7 y= řádka 0 až 3
grafický font 8x8 bodů	**znak**(kod , x , supery , inverze)	Zobrazí jeden znak s ASCII kódem "kod" na souřadnice "x" (0 až 15), "supery" (0 až 63) - horní strana znaku je na mikrořádce "supery". Při "inverze" = True se zobrazí tmavý znak na světlém pozadí.
	slovo(text , x , supery, inverze)	Zobrazí text (více znaků) na souřadnicích "x" a "supery" (parametry stejné jako ve funkci "znak()").
Vnitřní font displeje	**velky_znak**(kod , x , y)	Zobrazí jeden znak v textovém režimu na souřadnicích [x,y]. **kod** je v rozsahu 1 až 127 (od 32 do 126 je to klasické ASCII) **X** je od 0 do 15 **Y** je řádka 0 až 3
	velky_napis(text , x , y)	Pomocí velkých znaků zobrazí na displeji text. Parametry jsou stejné, jako u funkce "velky_znak()" a platí pro první znak z textu.
plot(superx, supery, styl)		Na souřadnicích "superx" (0 až 127) a "supery" (0 až 63) zobrazí, smaže, nebo invertuje jeden bod. Jestliže **styl= 0**, provádí se mazání bodu, při **styl=1** se bod zobrazí a při **styl=2** se provádí změna stavu bodu na displeji.
mem_plot(superx, supery, styl)		Stejná funkce jako předchozí "**plot()**", akorát se body nezobrazují přímo na displeji, ale pouze v dočasném paměťovém prostoru. Pomocí této funkce je možné rychlejší kreslení. Po použití je ale nutné přenést obsah té dočasné paměti na displej pomocí funkce "**mem_dump()**".
mem_dump()		Přenesení obsahu paměti na displej po použití příkazu **mem_plot()**.
h_cara(supery, od , do, styl)		Vykreslení jednoduché horizontální čáry ve vzdálenosti "supery" od horního okraje s možností definování začátku a konce čáry (proměnné "od" a "do"). Parametr "styl" je stejný jako ve funkci "**plot()**".
h_cara2(supery , od , do , pattern)		Rychlejší varianta kreslení horizontální čáry.V tomto případě jsou ale parametry "od" a "do" v rozsahu 0 až 7 (jsou to šestnáctiny pixelů na displeji). Čára tedy může začínat a končit jen na souřadnicích, na které se tisknou ikony. Minimální délka takovéto čáry je 16 bodů. Parametr "pattern" udává styl čáry. Podle jednotlivých bitů toho parametru se dá nastavit čára plná, čárkovaná, tečkovaná, čerchovaná...
v_cara(superx, od, do, pattern)		Vertikální čára na libovolných souřadnicích. superx = X souřadnice čáry v rozsahu 0 až 127 od, do = y souřadnice začátku a konce čáry v rozsahu 0 až 63 Parametr "pattern" je stejný jako v předchozím případě.
load_bmp12864(jmeno_obrazku)		Načtení dvoubarevného obrázku ze souboru do displeje. Pozor na správný formát souboru!
posli_bajt1(rs, bajt)		Odešle do displeje 1 bajt. pomocí parametru "rs" se vybírá datový (1), nebo příkazový (0) registr.
posli_bajt2(rs, bajt1, bajt2)		Odešle do displeje 2 bajty najednou . pomocí parametru "rs" se vybírá datový (1), nebo příkazový (0) registr.