Nízkoúrovňové programování

Práce s periferiemi I

Petr Krajča

Katedra informatiky
Univerzita Palackého v Olomouci

Sense HAT

- Sense HAT – rozšiřující modul (Hardware Attached on Top)
- připojeno na GPIO (General Purpose IO), viz `https://pinout.xyz/pinout/sense_hat`
- původně navrženo pro mise Astro PI (školní experimenty na ISS)
- řada senzorů (měření teploty, vlhkosti, atmosférickho tlaku, magnetometr, acelerometr, gyroskop)
- barevný LCD display 8x8, joystick pro ovládání
- část periferií přístupná přes jádro operačního systému (display, joystick)
- senzory přístupné přes sběrnici I2C (teoreticky vše přes I2C)

LCD display

- matice 8x8, barvy reprezentovány ve formátu 5:6:5
- jádro zpřístupňuje jako framebuffer
- zařízení `/dev/fb0`
- způsob komunikace
  - namapování do paměti sys. voláním `mmap` (preferovaná varianta)
  - alternativně můžeme pracovat jako s každým jiným souborem
  - zjištění stavu a informací o zařízení přes sys. volání `ioctl`

```
int fd = open("/dev/fb0", O_RDWR);
uint16_t *fb = mmap(NULL, HEIGHT * WIDTH * 2, PROT_READ | PROT_WRITE,
                    MAP_SHARED, fd, 0);

void fb_put_pixel(uint16_t *fb, int x, int y, uint16_t color) {
    fb[y * WIDTH + x] = color;
}
```

Výpis textu

- musíme vyřešit ve vlastní režii
- bitmapový font (velikost záleží na aplikaci)
- způsob uložení
  - (hardcoded) jako součást zdrojových kódů
  - jedna velká bitmapa s jednotlivými znaky v externím souboru
- vykreslení:

```
char font8x8_basic[128][8] = { ...,
    { 0x0C, 0x1E, 0x33, 0x33, 0x3F, 0x33, 0x33, 0x00} /* A */, ... }

void fb_draw_char(uint16_t *buf, uint16_t color, char c) {
    for (int y = 0; y < 8; y++) {
        for (int x = 0; x < 8; x++) {
            if (font8x8_basic[(int)c][y] & (1 << x))
                fb_put_pixel(buf, x, y, color);
        }
    }
}
```

Výpis rolujícího textu

- sestavíme bitmapu pro sousední dva znaky a vybereme odpovídajících 8 bitů
- pokud se vykresluje právě jeden znak převedeme na jednodušší řešení

```
void fb_draw_text_scrolled(uint16_t *buf, uint16_t color, char *s, int position)
    if ((position % 8) == 0) {
        fb_draw_char(buf, color, s[position / 8]);
        return;
    }
    char c1 = s[position / 8];
    char c2 = s[(position / 8) + 1];
    for (int y = 0; y < 8; y++) {
        for (int x = 0; x < 8; x++) {
            uint8_t mask_1 = font8x8_basic[(int)c1][y];
            uint8_t mask_2 = font8x8_basic[(int)c2][y];
            uint16_t mask = (mask_2 << 8) | mask_1;
            int xc = x + position % 8;
            if (mask & (1 << xc)) fb_put_pixel(buf, x, y, color);
        }
    }
}
```

Bitmapové obrázky

1. získat bitmapu
2. přeškálovat barvy (viz kód)

čtení bitmapového obrázku

- PNG, JPEG – knihovny libpng nebo libjpeg
- PBM, PGM, PPM – jednoduché formáty pro bitmapy, šedotónové a barevné obrázky
- příklad formátu PBM (ASCII):

```
P1   # format souboru
# komentar
6 4 # rozmery obrazku
0 1 0 0 1 0
0 0 1 0 1 0
0 0 0 1 1 0
```

- v RAW variantě hlavička (první 3 řádky) stejná, data v binární podobě
- nepříliš úsporný formát, podpora v grafických nástrojích, snadná implementace (klidně i na MCU)

Práce s obrázky typu PBM s pomocí knihovny „netbpm”

- balík netpbm (sada nástrojů pro práci s formáty P[BGP]M)
- nutné doinstalovat vývojový balíček `libnetpbm11-dev`
- linkujeme knihovnu `-lnetpbm`

```
#include <netpbm/ppm.h>
int cols, rows;
pixval max_val;

FILE *f = fopen("image.ppm", "r");
pixel **pixels = ppm_readppm(f, &cols, &rows, &max_val);
for (int r = 0; r < HEIGHT; r++) {
    for (int c = 0; c < HEIGHT; c++) {
        pixel p = pixels[r][c];
        fb_put_pixel(fb, c, r, scale_RGB_to_16b(p.r, p.g, p.b, max_val));
    }
}
fclose(f);
```

Práce s joystickem

- přístupný jako zařízení typu *input device*
- jako soubor `/dev/input/eventX`
- možná kolize s dalšími zařízeními
- nutné projít jednotlivá zařízení a zjistit podle identifikátoru, které je joystick
- použití sys. volání `ioctl`
- data v souboru uložena jako struktura

```
struct input_event {
    struct timeval time; // cas udalosti
    __u16 type;  // typ; napr EV_KEY (stisk klavesy), EV_REL (posun mysi)
    __u16 code;  // napr. kod klavesy; indikace osy, kde se mys posunuje
    __s32 value; // hodnota spojena s udalosti (napr. u klaves stisk/opakovani)
};
```

- joystick se chová jako klávesnice
- pro čtení je na místě použít systémové volání `poll` (viz předchozí seminář)

Hodnocené úkoly

(1) spojte kódy tak, aby šlo joystickem ovládat směr posunu textu

(2a) upravte hru z předchozího semináře, aby jako rozhraní používala Sense HAT

(2b) nebo vytvořte vlastní aplikaci, kde se bude používat LCD display a joystick