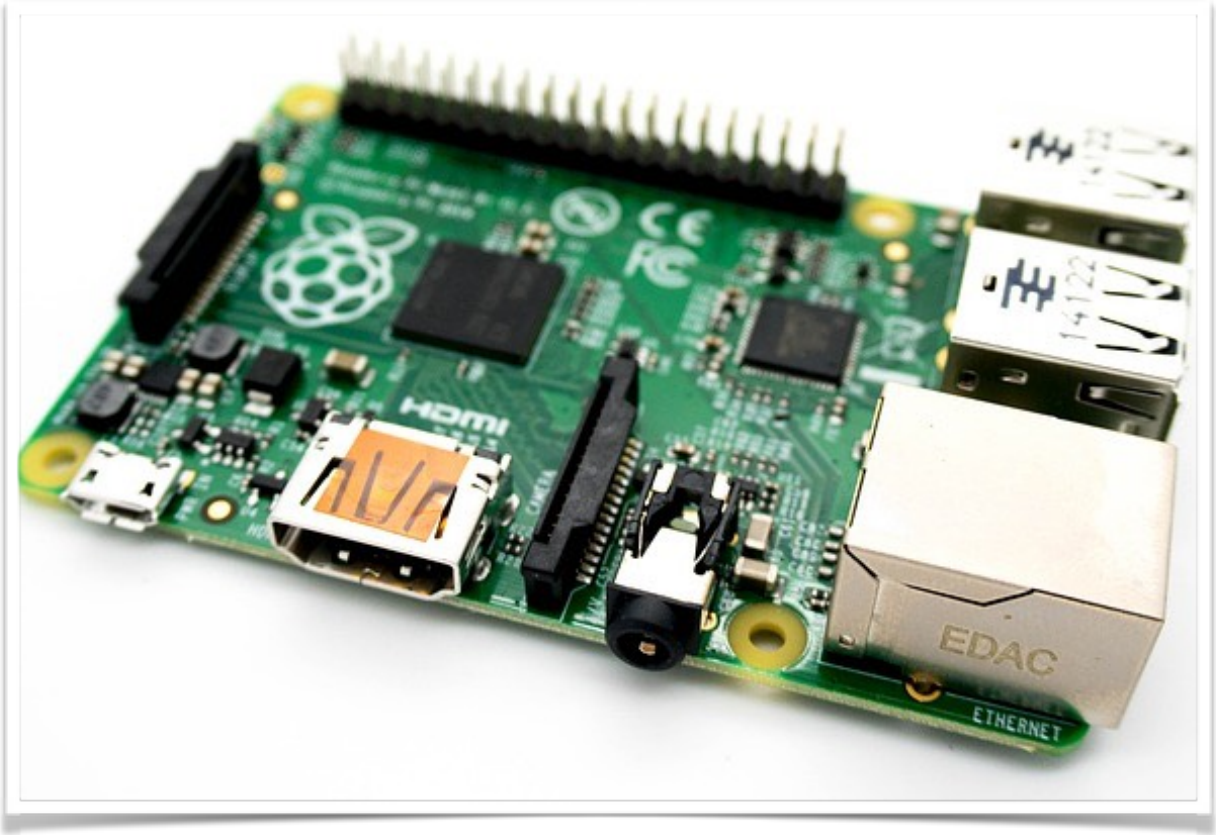
Linux w Systemach Wbudowanych



Laboratorium 4

Mateusz Sadowski

Lato 2018

1. Opis ćwiczenia

Ćwiczenie obejmował zbudowanie przy pomocy płytki RPi urządzenia przetwarzającego dźwięk lub obraz. System powinien być wyposażony w dwa interfejsy użytkownika:

1. Przyciski i diody LED
2. Interfejs WWW

3. Rozwiązanie

3.1. Opis systemu

Zostanie zbudowany system do zdalnego robienia specjalnych filmów w przyspieszonym tempie (z ang. "timelapse"). Tworzy się je w taki sposób, że kamerę montuje się w jednej pozycji lub na specjalnym suwaku, który płynnie będzie zmieniał jej położenie, a następnie robi się zdjęcie co określony interwał czasowy. Można w ten sposób otrzymać interesujący efekt dla takich zjawisk jak ruch chmur lub ciał niebieski po niebie, rosnącego drzewa itp.

W tym celu zostanie użyta płytka RPi oraz kamera internetowa dostarczona na laboratorium. Będzie możliwość włączania i wyłączania kamery oraz zmiany interwału czasowego z poziomu przycisków jak i interfejsu www. Czerwona dioda LED będzie sygnalizować nagrywanie kiedy kamera jest włączona. Zrobione zdjęcia będą zapisywane na karcie pamięci na nieulotnym systemie plików oraz na bieżąco wysyłane przez sieć na zewnętrzny serwer.

3.2. Oprogramowanie systemu

Powyżej opisany system będzie zrealizowany na systemie użytkownika z laboratorium nr 3. Cała procedura wgrania obrazu systemu na pierwszą partycję karty, sformatowanie drugiej partycji na potrzebę wgrania systemu plików oraz wgranie systemu plików zostały opisane w sprawozdaniu do laboratorium nr. 3 i zostanie tu pominięta.

Ważne dodane paczki do systemu to:

- libv4l - kolekcja bibliotek do obsługi wideo na Linuxie
- fswebcam - prosta aplikacja konsolowa do robienia zdjęć za pomocą kamerki internetowej
- openssh - umożliwia obsługę protokołu SSH, będzie wykorzystana do nawiązania stałego uwierzytelnienia między płytką a zewnętrznym serwerem, co pozwoli na przesyłanie zdjęć bez wprowadzania hasła
- nodejs - środowisko uruchomieniowe zaprojektowane do tworzenia serwerów www napisanych w języku JavaScript
- express - prosty framework aplikacji internetowych Node.js

3.3. Konfiguracja kamery

Aby dostarczona kamera działała z płytką RPi należy załadować odpowiedni sterownik/moduł *gspca_ov519.ko*. Można to zrobić uruchamiając go bezpośrednio pod ścieżką:

```
./lib/modules/4.9.79/kernel/drivers/media/usb/gspca/gspca_ov519.ko
```

Lub korzystając z polecenia *modprobe*:

```
modprobe gspca_ov519
```

Następnie należy podłączyć kamerę za pomocą wejścia USB. Poczekać na wydruk informujący o rozpoznaniu kamery a później można przetestować działanie kamery za pomocą komendy:
`fswebcam`

W celu automatyzacji ładowania sterownika dla kamery zostanie dodany skrypt do */etc/init.d* o nazwie *S080Webcam*. Szablon skryptu został zaczerpnięty ze strony podanej w źródłach. Sam skrypt jest za długi, żeby go umieszczać w sprawozdaniu, ale został dołączony do plików z rozwiązaniem.

3.4. Ustanowienie połączenia z zewnętrznym serwerem

Na laboratorium jako serwer zostanie wykorzystana stacja robocza. Zostanie ustanowione stałe uwierzytelnienia za pomocą komendy *ssh-keygen* oraz kluczy uwierzytelniających. Później zdjęcia każdorazowo będą wysyłane za pomocą komendy *scp* na stację roboczą do wybranego miejsca.

Najpierw należy wykonać komendę:
`ssh-keygen –t rsa`
Program zapyta o podanie kilku opcji. Każdorazowo wystarczy wcisnąć klawisz “Enter”, aby wybrać opcję domyślną.

To spowoduje utworzenie dwóch kluczy w *~/.ssh/*: prywatnego oraz publicznego. Aby dodać klucz publiczny do kluczy rozpoznawanych przez serwer należy wykonać komendę:
`ssh user@server_ip_address "echo \"`cat ~/.ssh/id_rsa.pub`\" >> ~/.ssh/authorized_keys”`
lub
`ssh-copy-id user@server_ip_address`

Od tego momentu możliwe jest łączenie się przez SSH do serwera bez podawania hasła.

3.5. Skrypt robiący zdjęcie

Do zrobienia pojedynczego zdjęcia oraz przesłania go na serwer wykorzystywany jest poniższy skrypt wywoływany w programie sterującym kamerą komendą *system*. Więcej opisane w podpunkcie 3.6.

webcam.sh

```
#!/bin/sh

DATE=$(date +"%Y-%m-%d_%H:%M:%S")
SAVE_PATH="/usr/webcam/timelapse"

fswebcam -r 1280x720 --no-banner $SAVE_PATH/$DATE.jpg

scp $SAVE_PATH/$DATE.jpg sadowskim@192.168.143.178:/home/samba/sadowskim/webcam/timelapse
```

3.6. Program sterujący kamerą

W celu automatyzacji i umożliwienia sterowania systemem został napisany program w języku C. Do obsługi przycisków i diod LED została wykorzystana biblioteka *wirinigPi*. Proces kompilacji programu na odpowiednią architekturę został opisany w sprawozdaniu do laboratorium nr 2.

Aby umożliwić jednoczesne sterowania systemem z poziomu przycisków jak i aplikacji webowej został wprowadzony plik konfiguracyjny. Program sczytuje aktualną konfigurację z pliku i odpowiednio do niej steruje kamerą. Program również zapisuje zmiany w konfiguracji wprowadzone z poziomu przycisków do tego pliku.

Program jest uruchamiany za pomocą skryptu startowego *timelapse.sh*. Ten skrypt przed uruchomieniem programu eksportuje odpowiednie piny, aby program mógł z nich korzystać.

timelapse.sh

```
#!/bin/sh

# Causes wiringPiSetup functions return error code on failure
export WIRINGPI_CODES=1

# Make GPIO available for regular user
# -g option enables BCM-GPIO pin numbering
# Export LEDs as output
gpio export 17 out
gpio export 18 out
gpio export 23 out
gpio export 24 out

# Export buttons as input
gpio export 27 in
gpio export 22 in
gpio export 10 in

# Enable button pins for edge interrupt triggering on both
gpio edge 27 both
gpio edge 22 both
gpio edge 10 both

# Run program
/usr/bin/timelapse

```

Obsługa przycisków, debouncingu oraz diod w programie została zrealizowana w taki sam sposób jak na laboratorium nr. 2. Kod źródłowy został załączony wraz ze sprawozdaniem. Kluczowe jego elementy zostały również załączone poniżej:

```c
int main(int argc, char *argv[])
{
    processArguments(argc, argv);
    setup();

    run();

    return EXIT_SUCCESS;
}
```

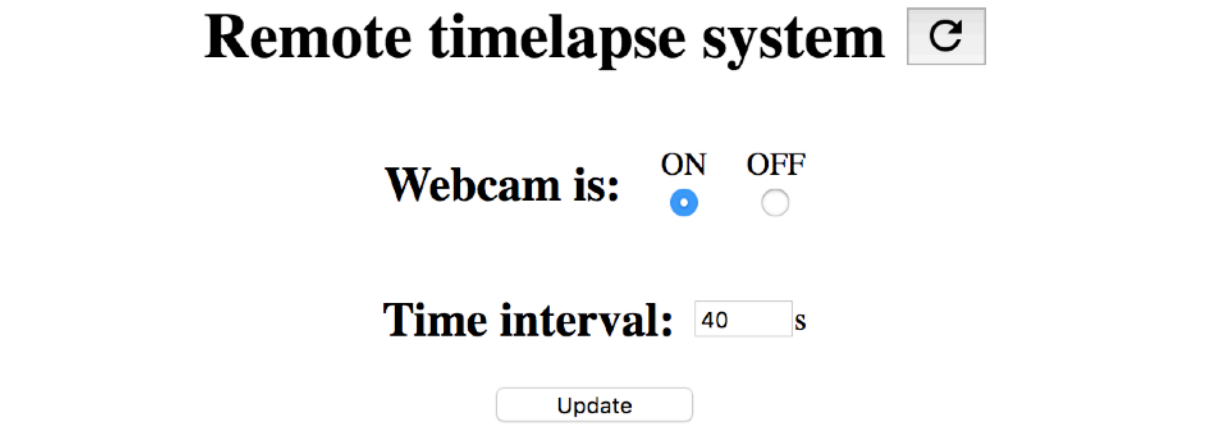
```c
void run()
{
    while (1)
    {
        printf("Waiting to start capturing...\n");
        capture();
        sleep(DELAY_TIMEOUT);
    }
}
```

```c
void capture()
{
    importConfigurationFromFile();
    while (isCapturing)
    {
        printf("Captured a photo. Time interval: %d\n", captureDelay);
        system("/usr/bin/webcam.sh > /dev/null 2>/dev/null");
        flashLED();
        sleep(captureDelay);
        importConfigurationFromFile();
    }
}
```

```c
void importConfigurationFromFile()
{
    char buff[255];
    FILE *config = openConfig();

    if (-1 == fseek(config, 0, SEEK_SET))
        perror("fseek");
    if (EOF == fscanf(config, "%s", buff))
        perror("fscanf");
    isCapturing = atoi(buff);

    if (EOF == fscanf(config, "%s", buff))
        perror("fscanf");
    captureDelay = atoi(buff);
    //printf("Status: %d, Time interval: %d\n", isCapturing, captureDelay);

    closeFile(config);
}
```

3.7. Aplikacja webowa

Aplikacja webowa została napisana w języku JavaScript przy użyciu środowiska Node.js i frameworka Express. Interfejs informuje użytkownika o obecnej konfiguracji (możliwość odświeżenia za pomocą przycisku) oraz pozwala na jej edycję, a w konsekwencji na sterowanie systemem z poziomu przeglądarki internetowej.



Kod źródłowy aplikacji został załączony wraz ze sprawozdaniem. Kluczowy element kodu serwera poniżej:

```
17  app.use(express.static(__dirname));
18  app.use(bodyParser.urlencoded({ extended: false }))
19  app.use(bodyParser.json());
20  app.get('/', (req, res, next) => {
21      res.redirect("home.html");
22      next();
23  })
24  app.get('/refresh', (req, res) => {
25      fs.readFile(filePath, { encoding: 'utf-8' }, function (err, data) {
26          if (!err) {
27              console.log('Received get config file request. Sent:  ' + data);
28              res.send(data);
29          } else {
30              console.log(err);
31          }
32      });
33  })
34  app.post('/update', (req, res) => {
35      var statusReq = req.body.status;
36      var timeReq = req.body.time;
37      fs.writeFile(filePath, statusReq + ' ' + timeReq, function(err) {
38          if(err) {
39              return console.log(err);
40          }
41
42          console.log("Config file updated. Status: " + statusReq + ", Time interval: " + timeReq);
43      });
44  })
45
46  app.listen(PORT, () => console.log('Timelapse webapp listening on port ' + PORT + "!"))
```

3. Odtworzenie projektu

Aby odtworzyć projekt należy w dużej mierze postępować według kroków opisanych powyżej. Obrazy systemów i system plików można wygenerować przy pomocy odpowiednich plików **.config** załączonych wraz z dokumentem. Pliki źródłowe programu oraz aplikacji można wgrać bezpośrednio na kartę pamięci do nieulotnego systemu plików. Program powinien zostać uruchomiony za pomocą skryptu startowego *timelapse.sh* a aplikacja internetowa za pomocą komendy *node server.js*. Plik konfiguracyjny powinien zostać umieszczony w folderze */usr/webcam/*.

4. Źrodła informacji

- https://github.com/fhd/init-script-template
- https://www.howtogeek.com/66776/how-to-remotely-copy-files-over-ssh-without-entering-your-password/
- https://stackoverflow.com/questions/23591083/how-to-append-authorized-keys-on-the-remote-server-with-id-rsa-pub-key
- https://www.raspberrypi.org/documentation/usage/webcams/README.md