Intermediate Project

Embedded system for sewing machines activity monitoring using WiFi protocol

Author: Mateusz Rado

Department of Electronics, Embedded Robotics
Wrocław University of Science and Technology

Supervisor: Ph. D. Witold Paluszyński, K-7

February 2nd, 2017

Abstract

Main goal of the project was to accomplish embedded system, which could monitor industrial sewing machine activity (engine speed, number of stitches, amount of thread, etc.). All elements of the system should work in one WiFi network and send information from sewing machine to the server. In the first part of this report, it was presented problem. Afterwards it was described way of configuration of the FTP server at Raspberry Pi computer and implementation WiFi client on the embedded module (ESP8266). The results of the project are discussed in the last section of document.

1 Introduction

Sewing industry is gaining in importance. Sewing machines are used in more and more complicated tasks, level of its complication increase fast. For this reason monitoring of sewing process is really important. Monitoring of sewing machine parameters such as number of stitches during sewing a single material and related to it speed of motor could helps in optimization of production and increasing quality of the products. Fast and precise data transfer from sewing machine straight into the server might be used as one of the method to achieve these results.

2 Raspberry PI

Server for receiving data from sewing machine was set on the Raspberry Pi. First of all, operating system was installed on the SD card. Choosen system was Raspbian Jessie (release date: 23.09.2016).

2.1 Connection via SSH

If there is physical access to the router, Raspberry could be connect to it via Ethernet cable. Then the router automatically gives Rasperry IP and there are ability to connect to it via SSH protocol.

First connection to Raspberry Pi (without knowledge about Raspberry's IP in the network) could be also realize via editing file *cmdline.txt*, located at SD card, by adding at the end of line with configuration of the Linux kernel, IP which we want to be used by Raspberry. Assigned IP should be same type as network in which it is using. For example if network address is type of 192.168.X.0, then the Raspberry IP could be analogous with changed value of X (in range 1-255). It is important to not overwrite existing IP in the network.

Before one of this actions, connection to Raspberry could be realized.

2.2 Wireless USB network adapter

In this project, Raspberry is connected to WiFi network using wireless USB adapter - Tp-Link TL-WN725. To use this adapter, drivers for it have to be installed. To do it, the version of installed system has to be known. Commend *uname -a* typed in the command line will return the version of the system. Then, at the page: `https://www.raspberrypi.org/forums/viewtopic.php?p=462982#p462982`, proper version of the system and related to it link for download drivers should be find. After that, to get the package with the drivers were used command *wget* with proper link as argument. Next, downloaded package were unpack with *tar xzf* and name of the unpacked file, then installed *sudo ./install.sh*. At the end system was rebooted.

2.3 Automatic connection to WiFi network

To set the automatic connection with WiFi network for Raspberry Pi, file with configuration of the networks should be edited. Searched file - *interfaces.d* is localized in `\etc\network` directory. Proper configuration of the file is shown below:

```
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

iface eth0 inet manual

allow-hotplug wlan0
iface wlan0 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

allow-hotplug wlan1
```

```
iface wlan1 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

2.4 Remote access

Remote access signficantly simplifies work with Raspberry Pi. VNC Server is one of the available solutions for remote access tools. Installation were made using command *sudo apt-get install tightvncserver*. Next application were turn on with *tightvncserver* and password for connecting with Raspberry were set. After than, it was neccesary to edit file `\etc\init.d\tightvncserver` to allow access to Raspberry. Source of this file is shown below:

```
export USER='pi'

eval cd ~$USER

case "$1" in
  start)
    su $USER -c '/usr/bin/tightvncserver :1'
    echo "Starting TightVNC server for $USER "
    ;;
  stop)
    pkill Xtightvnc
    echo "Tightvncserver stopped"
    ;;
  *)
    echo "Usage: /etc/init.d/tightvncserver {start|stop}"
    exit 1
    ;;
esac
exit 0
```

Next proper rights were set to this file using *sudo chmod 755* `\etc\init.d\tightvncserver`. At the end process were configurated to start with the system: *sudo update-rc.d tightvncserver defaults*.

After installation of VNC Server apllication at Windows, remote access is realized by typing IP of Raspberry and number of port (for example *192.168.1.18*) and password set in file `\etc\init.d\tightvncserver`.

3 FTP server

Server used in this project was set on the Raspberry Pi 2 computer. There was two main types of the FTP servers available on Unix system: *ptoftpd* and *vsftpd*. During this project it was used *vstpd* server.

Process of installation were started by typing *sudo get-apt install vsftpd*. During the installation, user is asked about mode of the server (`inetd\standalone`). In this project were chosen standalone mode. File with configuration of the server is located at . Source of the file:

```
listen=YES                      #running as a seperate process
listen_ipv6=NO
anonymous_enable=YES            #anonymous login
anon_root=/home/trym/ftp        #catalog for anonmous users
local_enable=YES                #local users can login
write_enable=YES                #writing is allowed in directories
local_umask=077                 #rights for created files
anon_upload_enable=NO           #anonymous upload to the server
anon_mkdir_write_enable=NO      #anonymous creating directories
dirmessage_enable=NO            #messages during changing the directory
xferlog_enable=NO               #log transfers
connect_from_port_20=YES        #from which port connections will start
xferlog_file=/var/log/vsftpd.log
xferlog_std_format=YES
idle_session_timeout=600
data_connection_timeout=120
```

```
ftpd_banner=Hello!                    #message before connection
chroot_local_user=YES                 #prohibition of changin home directory
#list of users enabled to changing
#home directory(if chroot_local_user=YES)
#chroot_list_enable=NO
#chroot_list_file=/etc/vsftpd.chroot_list
secure_chroot_dir=/var/run/vsftpd
pam_service_name=vsftpd
```

After configuration of FTP server, user can connect to it using dedicatied applications e. g. FileZilla or typying in web browser addres of Raspberry Pi preceded by *ftp://* (e. g. *ftp://192.168.1.18*).

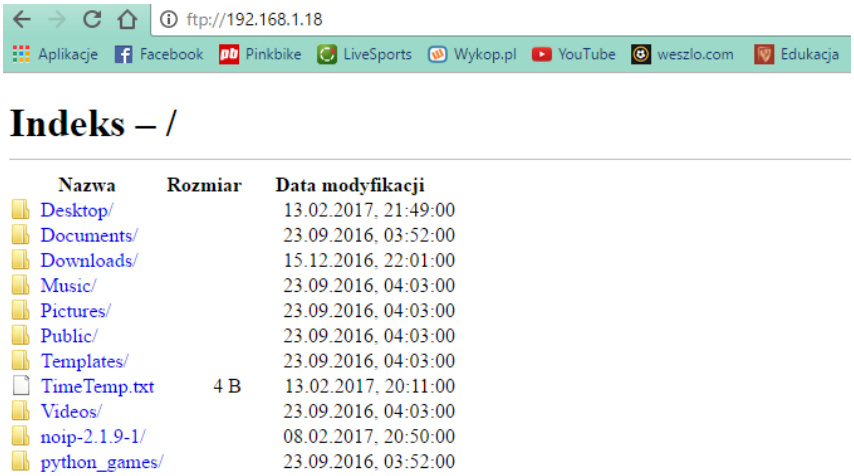

Figure 1: Home directory of FTP server after connection

4 Configuration and programming of ESP8266 module

4.1 ESP8266 configuration

Module is very sensitive and if it has insufficient current efficiency power, it could restarts or work unstable. Power should be filtered by proper capacitors. Correctly way of connection ESP8266 module (including pull-up/pull-down 10k resistors and filtering 100nF capacitor) are shown below. It also might be used 470μF capacitor close to module, which should prevent it from unwanted restarts and ensure stable work.

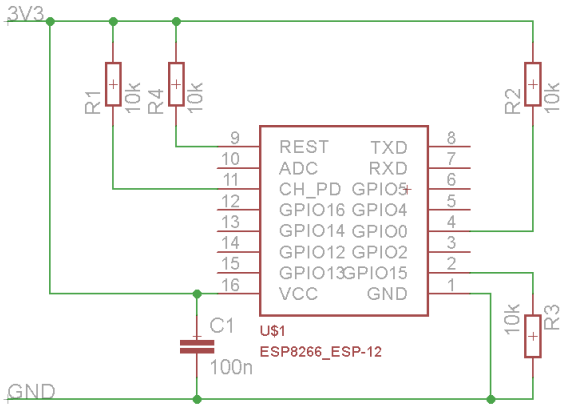

Figure 2: Connection of ESP8266-12E

4.2 Arduino IDE

ESP8266 module brings user a lot of benefits. It can be programmed in many languages such as LUA, python or Arduino. In this project ESP8266 module were programmed in Arduino Integrated Development Environment. Used version of Arduino IDE was 1.8.1. To perform an environment for programming ESP8266, it was installed the newest version of software, which can be downloaded from `https://github.com/esp8266/Arduino`[5].

To install the software, user has to choose option preferences from Arduino menu and provide link to the software located on *github* in proper window, as shown below.

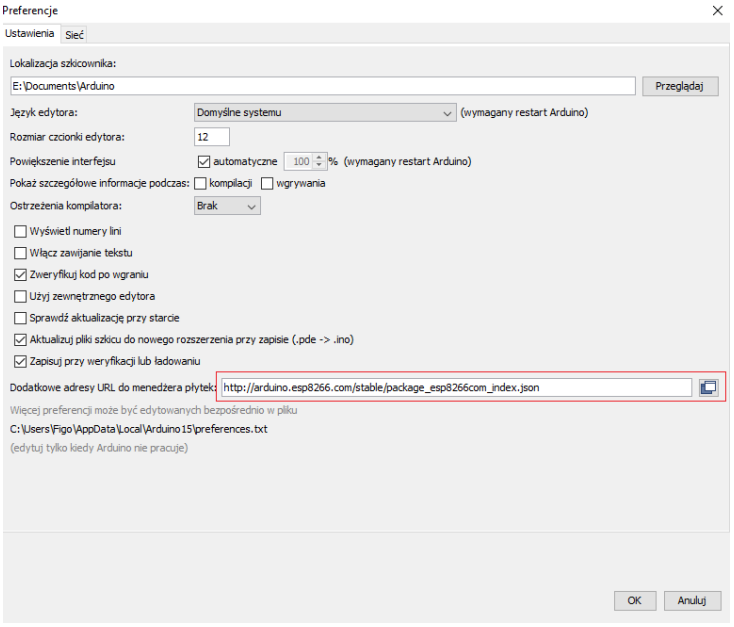

Figure 3: Arduino preferences menu

Then it has to be choose option *Board Manager* from *Tools* menu. In this window, user has to find ESP8266 software and choose proper version of softare. In this case, it was choosen version 2.3.0, because it provides libraries for handling with SPI Flash FileSystem, which has been used in this project.

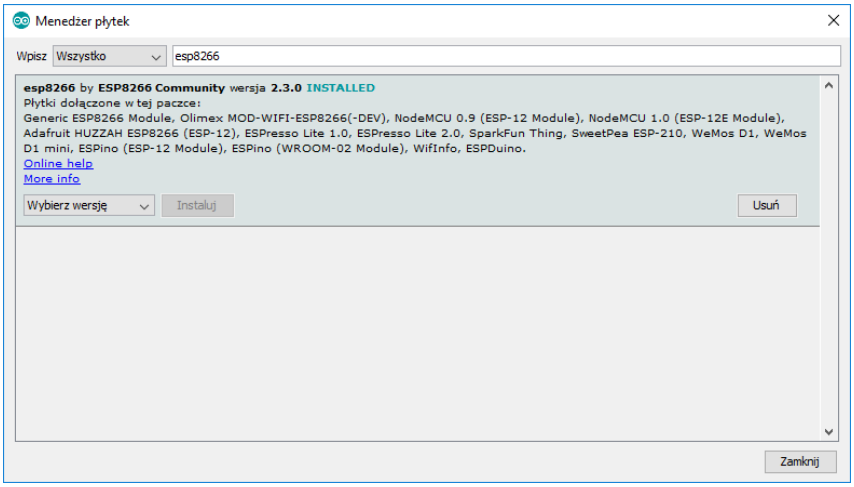

Figure 4: Arduino board manager

To program the module, user has to set few options according to version of his ESP8266 board. In this project it was used ESP8266-12E board and below is shown configuration for programming this type of board.

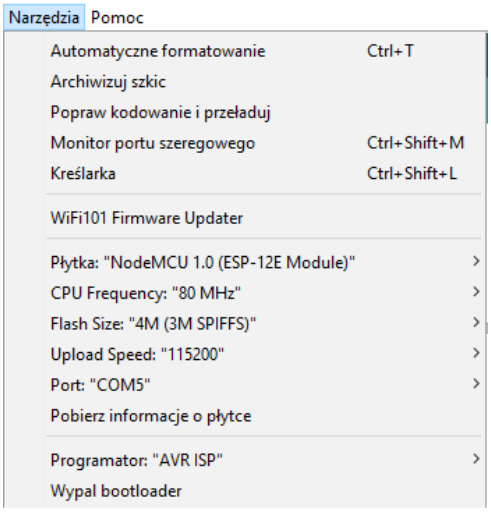

Figure 5: Programming preferences

5 FTP client

5.1 SPIFFS

ESP8266 has own file system, which helps handling with data storaged on the board. SPIFFS (SPI Flash File System) is stored on the same flash chip as the program, but programming new sketch will not modify file system contents. This allows user to use SPIFFS to store sketch data on the module.[6]

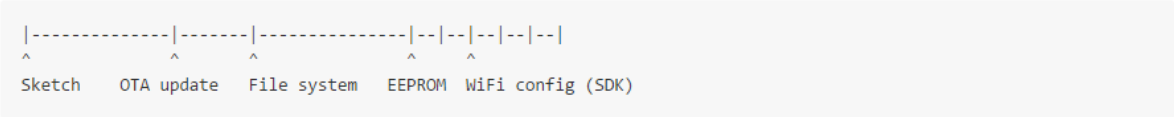

Figure 6: Flash layout used in Arduino environment
[6]

Data could be saved on the ESP8266 from Arduino IDE level (using *ESP8266 Sketch Data Upload tool*) or by creating file directly in the code.

5.2 Client

As main part of the project it was developed FTP client, which handle connection with the network and transfer files between module and server set on Raspberry Pi. Program allows to download SPIFFS file from server to board, upload SPIFFS file from board to server and show SPIFFS files storaged on the board. Every time module is turning on, program scan network (IP and password for both server and WiFi network are provided in the code), connects to the network, connects to the server and allows user to do proper action via serial port monitor:

- -u upload
- -d downlaod
- -r show SPIFFS files

Program were developed using libraries for Arduino cited in the references. Results of working program are shown below. During the test it was succeeded to download files from the server into the flash memory of ESP8266 module and upload files from memory to FTP server.

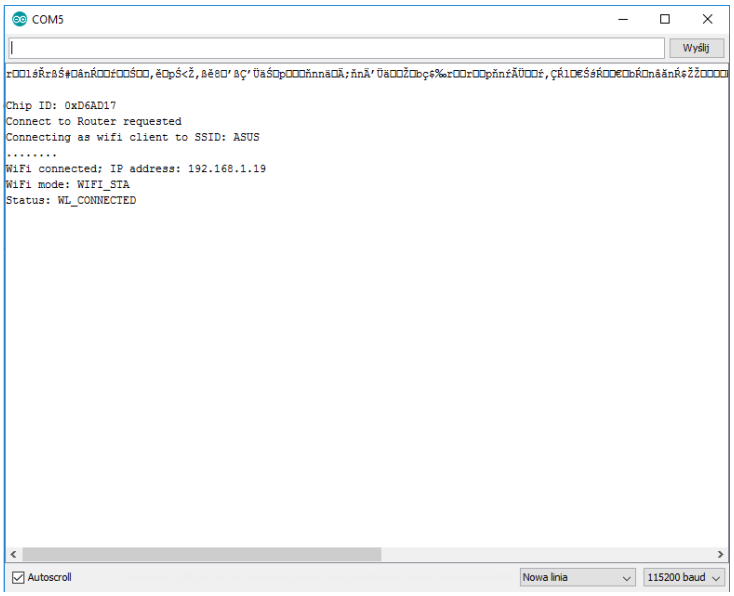

Figure 7: Serial port monitor

6 Results

During the formation of this project it was created a prototype of module, which provide proper powering of ESP8266 and connecting of signals from sewing machine. It was also developed program, which allows user to download files from external FTP server and upload files into it (client of the WiFi network), which was implemented using Arduino IDE and libraries available at *ESP8266 Community Forum*[4]. It was succeeded to transfer files between server and client in WiFi network.

Taking into account software, it has to be prepared implementation of handling with signals incoming from controller of sewing machine and saving it in file at ESP8266 module. Next this file will be send to a FTP server using already developed program.

References

[1] Simon Monk, *Raspberry Pi: Przewodnik dla programistów pythona*. Helion, 2014.

[2] M. Evans, J. Noble, J. Hochenbaum, *Arduino w akcji*. Helion, 2014.

[3] *ESP8266 SDK Getting Started Guide ver. 2.5*, 2016.

[4] ESP8166 Community Forum
`https://github.com/esp8266`.

[5] ESP8266 core for Arduino
`https://github.com/esp8266/Arduino`.

[6] ESP8266 SPIFFS library at *github.com*
`https://github.com/esp8266/Arduino/blob/master/doc/filesystem.md#file-system-object-spiffs`.

[7] Arduino WiFi library
`https://www.arduino.cc/en/Reference/WiFi`.

[8] Arduino serial library
`https://www.arduino.cc/en/reference/serial`.

[9] *Efka DC1550 instruction manual.*

[10] *Efka Compiler C200 user manual.*

[11] *Efka compiler C200 for advanced learner.*