Dot Matrix Notifier

Daniel Hipp
Faculty of Computer Science
HTWG Konstanz
Konstanz, Germany
dahipp@htwg-konstanz.de

Etienne Gramlich
Faculty of Computer Science
HTWG Konstanz
Gailingen, Germany
etgramli@htwg-konstanz.de

Thomas Gnädig
Faculty of Computer Science
HTWG Konstanz
Singen, Germany
thomas.gnaedig@htwg-konstanz.de

***Abstract*—In this paper we introduce our approach to a Raspberry Pi controlled dot matrix used for displaying text and images and notifying in case of incoming messages. Our proposed approach will be explained, with also the difficulties with the matrix and the final result.**

The Raspberry Pi will be connected to the router via LAN for internet access and to the dot matrix it controlls via I^{2}C. The user's smartphone will be connected to the Raspberry Pi' WiFi interface that acts as an access point. From now on the smartphone can send HTTP requests to the Raspberry Pi, that prints the containing text and images on the matrix and is also able to the Internet as usual.

***Index Terms*—LED, dot matrix, LED matrix, Android, smartphone, WiFi, notification, client server**

I. Introduction

The goal of this project is to provide a screen that shows notifications, text and images from a smartphone so that they are better visible. It should stand alone somewhere in the room and is not intended as an extension to smartphone itself. It can also be used as WiFi accesspoint if any LAN connection is avaliable. Without any connection it can be used portable with a powerbank, provided there is enought power output for the dot matrix.

An inexpensive replacement for a screen is a dot matrix, that consists of a rectangular array of LEDs. Such a dot matrix can be easily connected to an Arduino or Raspberry Pi. The I^{2}C interface is supported by both devices. Due to the networking requirement of the task a Raspberry Pi version 3 seems to be the better choice over an Arduino. The latest Raspberry Pi has multiple network interfaces, such as LAN, WiFi and Bluetooth.

II. State of the Art

In this chapter we name three diffrent products with similar behavior as our matrix.

First of all there is the TEBE [1], this matrix has only one row for letters wich can be infinite long. Our matrix got only one row too, but the length is limited by 16 LEDs. In contrast to our matrix, it is possibile to use multiple colors on the LEDs, so colored text can be shown if needed, but no support for colored images is avaliable. To print some letters on the TEBE Matrix, you need to use a windows application and set your text. Special characters like stars are supported, but no image can be shown. Compared to our matrix we just need a smartphone to change the currently displaying message. So the TEBE is more suitable for static outputs, wich have only to be changed a few times a day. There is a timemanaging system, were you can define, how often your message should change, but you need to know this message at initialisation time. Therefore this module can be used inside and outside. Our matrix on the other hand is just suitable for dry interior environments. An extension to the TEBE matrix is the Proline [2]. This series supports multiple lines above, in theory there is no limitation for the number of rows to be used. But there is still no image support as in our matrix avaliable. It is possible to print out different characters on different rows with individual color, so you could try to do some ASCII-Arts but no real image can be shown.

Another comparable product is the Eurolite ESN [3], which also just supports one single row of characters. It also can change each pixel's color differently. The text can be displayed in 14 different fonts, is able to blink if necessary and can move to any direction. Out matrix just supports one direction and only one font with only capital letters and a few special characters. Another nice Feature is that the current date and time can be displayed in different formats (DD.MM.YYYY, US/UK or MM.DD.YYYY). This feature could be implemented to our matrix as well in the future. Even though there is the possibility to color each LED separately, there is still no image support avaliable.

III. Proposed approach

We have to choose a matrix, that is compatible with at least the Raspberry Pi and must not exceed the power that the Raspberry Pi can deliver. There are popular model with the MAX7219 chips from AZ-Delivery and Adafruit.

Since our dot matrix should sit somewhere in the room, it must connect wirelessly to the user's smartphone. As smartphones usually have WiFi and Bluetooth interfaces built-in and do not contain a LAN interface, WiFi and Bluetooth are the intersection of both devices.

As with web servers threre are already well-established technologies, we decided to transfer our application's data over HTTP from the client to the server. We will programm an Android app, so that the user can easily interact with our matrix. The implementation approach is discussed in the following sections.

A. Coose the matrix

The matrix must be compatible with the Rasbperry Pi, but for testing purposes the compatibility to Arduino is desirable. So the protocol must be either SPI or I^2C, because both are supported on Raspberry Pi and Arduino.

It also must not consume more energy than the Raspbery Pi and Arduiono can provide, that makes it more easy to assemble and run. So it must work at a voltage of 3.3V or 5V and must draw at most 50mA and 300mA repectively [4].

Since there are no square, pre-assembled matrices of a size of 16x16 or larger, we have to build one out of smaller dot matrices. There are 8x8 or 8x16 Modules avaliable, wich can be used for this. The individual matrix must be addressable individually. So that a wrapper library can be built that acts as a single panel and outputs part of the image on the respective sub-matrix.

There are models with various sizes using the MAX7219 chip, which is an IC display driver for dot matrices and 7-segment LED displays. It makes programming in Python very easy and is compatible with both Arduino and Raspberry Pi.

B. Communication between Raspberry Pi and the matrix

The communication interface between the Raspberry Pi and the matrix must support multiple devices over one connection, so that multiple matrices can be combined to one larger matrix. For example SPI supports multiple slaves (matrices) that would be connected to one master (the Raspberry Pi). Also I^2C supports multiple slaves by different addresses and even multiple masters, the latter is not important for our use case. Both interfaces fulfill the conditions and are avaliable in combination with the MAX7219 chip.

In our setup, we use two 8x16 LED-Matrices, wich are created by two combined 8x8 matrices. The comminuication is completely handeled by the MAX7219 and we can see them as one 8x16 matrix. To connect both 8x16 matrices in parallel to the same output at the Pi, they need to get different adresses. After the basic setup is done, we created a python module wich handles the two different matrices as one 16x16 matrix. This can be done by splitting the input into half and send the upper informations to the first- and the other part to the secound adress. This comminication is done by the Adafruit_LED_Backpack [8] library, wich takes a 8x16 shaped array and sends it to the defined adress at the I^2C bus.

To display messages, the incoming string has to be converted into something wich fits on our 16x16 matrix. Therefor we choose a font to convert each incoming char to a 8xn array, the n is depending on the char value. The n lies betwen two, for a dot, and six for the uppercase letter M. The complete representation of an incoming string is a concatenation of all arrays from the converted chars, so the resulting shape is 8xn. To display this representation on our 16x16 matrix, the shape has to be adjusted. This can be done by adding zero padding above and below the representation, so the Result is a 16xn array, with letters in rows five to twelve and zeros in the other rows. Now we can display a 16x16 shaped submatrix of the full representation by indexing. The selected columns are increased sequentially with a short delay, in our case 0.125 sec, and displayed to the LED matrix. To get a smooth transition between the incoming text and the empty matrix, we also concatenate a 16x16 zero array before and after the representation. To display an binary image, we just convert the incoming string to an binary 16x16 array.

C. Communication between the Client and Server

Since WiFi and Bluetooth is the intersection of the Raspberry Pi's and smartphone's network interfaces, the communication must use one of those. Because the user may already have the smartphone connected to a headphone or headset, we chose to use the smartphone's WiFi interface. But to maintain normal smartphone usage, the internet connection, that was previously possible by the user's WiFi, must be piped through the Raspberry Pi. It must be connected to the internet via LAN and act as an access point for the smartpone. This way we can ensure that the Raspberry Pi has an static IP address, making the connection easier. Figure 1 shows an overview of the network connection.

The communication between the smartphone app and the Pi is done by simple running a HTTP server on the Pi. To send data to the matrix, the app just has to send an HTTP-GET request. The requested page is not a real page, it has to be the message or picture wich should be shown on the matrix. To distinguish between a text and a image we defined a capital letter for both types. To send a text to the Pi, the get request has to look like this : GET /S*text_to_display* for an image the letter I has to be used. Because the matrix supports only on and off and no colors or brightnesses the image data is just made of ones and zeros. To representate the image as string it has to be converted to a 16x16 binary image. This has to be converted to a 16x16 binary array by using a threshold. Now the array has to be flattened and the result should be a vector with 256 entries, each can either be one or zero. to display this flattened image the get request should look like this : GET /I*011010110...*. Whenever the server on the Pi gets an HTTP-GET request, a method is called, to check if its a legal request and then interprets the payload as image or string, depending on the first letter.

D. WiFi access point

To function as an access point, the Raspberry Pi needs packages for the access point itself, a DHCP server and for network address translation (NAT). Therefore two daemons are needed, one for using the WiFi interface as an host for the access point (hostapd) and one DHCP daemon.

The host access point daemon is configured to use the Raspberry Pi's built-in WiFi chip as the network interface and WPA2 encryption with a predefined password.

The DHCP server must assign an address to the smartphone connecting to the WiFi. It also communicates the Raspberry Pi as the Gateway so that the smartphone can connect to the internet.

Iptables is configured so that the smartphone in the Raspberry Pi's WiFi can access the internet through NAT at the Pi's

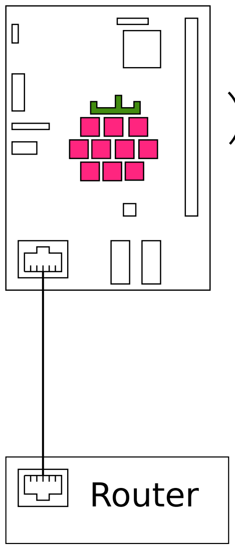

Fig. 1. Communication between Raspberry Pi, Smartphone and Router

LAN interface. This is solved that way, that new connections from the WiFi are accepted, but no new connections from the LAN NIC. Packages at the LAN NIC are only accepted, when they belong to an existing connection.

IV. Results

Finally we use a Raspberry Pi to drive two dot matrices and act as an access point for the smartphone. It routes normal internet traffic through its LAN interface to the existing router. The detailed description of our solutions follows in this chapter.

A. Difficulties

Difficulties arose from the first matrix that we used. It was a 8x32 controlled by a MAX7219 chip [5], that was supposed to work with both Arduino and Raspberry Pi. Not even the example code from the manufacturer's repository [5] worked on our Raspberry Pi 3, the matrix only printed random looking patterns. Neither a level shifter from 5V to 3.3V nor a separate power supply for the dot matrix did solve the problem. Therefore we moved to dot matrices from Adafruit [6]. That new matrix worked flawlessly with the Raspberry Pi, and has a "backpack" with an I^{2}C interface, instead of SPI.

B. Communication

The Raspberry Pi is our connection hub and main device. It has to connect the smartphone to the matrix and so has to maintain connection to both. The Raspberry Pi also delivers the power for the matrix from its own power supply.

C. Communication between the Client and Server

The communication between the server software on the Raspberry Pi and the client on the smartphone is done with HTTP requests. Only strings can be sent to the Raspberry Pi, so images gave to be encoded as strings.

If the received strings only contains zeros and is 256 (16x16) numbers long and an identifier letter, described in section III-C. It will be interpreted as an binary image and directly outputted on the dot matrix.

If the message contains an S as identifier, the following characters will be interpreted as a string to display. Special characters (i.e. apersand, space) must be escaped, as usual for HTTP, before sending to the Raspberry Pi. The characters are converted with a special font to images and outputted on the matrix.

D. Communication between Raspberry Pi and the matrix

As described earlyer, we use two 8x16 matrices above each other to form a large 16x16 matrix. Both are connected as slaves to the Raspberry Pi's I^{2}C interface. Therefore the two matrices need different addresses, so one of the matrices needs to be soldered to alter its default address.

After that step both matrices can be used at the same I^{2}C interface at the Raspberry Pi. But since there are only python libraries for one 8x16 matrix, we wrote a wrapper so that both matrices appear as a single 16x16 matrix. So we can print images on the matrix, to write text on the matrix one needs a special font for low resultion screen. With that we can render the glyphs of the text and display them on the matrix. For long texts we implemented scrolling from right to left.

E. The Android App

After the user installed the app he is able to send a text or an image to the dot matrix. Additionally the app sends the icons of incoming notifications to the matrix, if this feature is enabled. To achieve this, the app sends a http-request to the Rapsberry Pi with the text or image enbedded into the request. The requests are sended with the help from vollay, an Android library to send messages between multiple endpoints. The bitmaps of the images to be sent are scaled in size to 16x16 pixels and are converted to black-white binary images. At last they are changed into a string to make sendig over a HTTP request possible. The last point is the notification service which sends the notification's icon, which gets converted as normal images. The notification-service can be turned on or off by the user. Finally the user can use our app to show his favorit texts, images or notifications on a matrix. Figure 2 shows the home screen of our app, leading to the options to send a text or an image.

V. Conclusion and Future Work

This Project was done in a short period of time during a semester, we got a great idea of what to do and had to handle some complications. Especially the problems with SPI Matrix, took us a lot of time. In case, that no solution could be found quickly, we switched to I^{2}C. With the new Matrix we could implement our Ideas and got a well working Dot Matrix Notifier. There are still some parts in our project, that can be improved or further developed. The most important ideas will be discusst in the following sections.

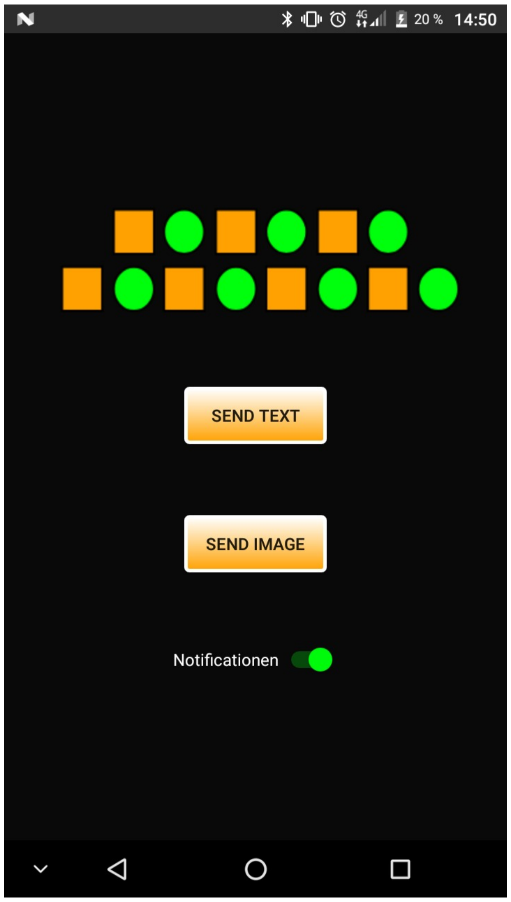

Fig. 2. Home screen of our Android app

A. Dot Matrix

Improvements can be made with better matrices, for example multi-color matrices or those that can be dimmed, to display a color image or a greyscale image respectively. A bigger matrix is possible, too. All this needs more computing power, but the Raspberry Pi is easily capable of this task, but an Arduino might not capable of that. The replacement of the matrix makes changes to the wrapper inevitable and one has to consider the power consumption of a larger matrix, a 32x32 dot matrix might draw up to 4A at 5V.

As new Feature a clock could be implemented, wich shows the current time, whenever no requests are incoming. So it could be used as simple clock even without a smartphone connected.

B. Server Security

Since the web server is not secured, multiple smartphones can connect to the Raspberry Pi and all of them can send messages to the Raspberry Pi simulateously. One possibility is that only one smartphone should be able to connect to the server. It can be achieved by a code, that is displayed on the matrix and entered on the smartphone. This makes sure that only the first smartphone is able to send messages. Optionally an encryptet HTTP connection could be used, to make the messages not readable for third parties.

C. Case and power supply

A relative simple improvement to the current status of our project is a nice case that contains the Rapberry Pi and provides an outlet for the matrix.

For the current setup with a Raspberry Pi 3 and a 16x16 matrix requires the power that a standard power supply can deliver. Our project should actually be powered in that way, becuase it also need a LAN cable is therefore stationary by design.

Another idea is to make the project mobile, which would requre a battery contained in the case. But this needs regular recharging. The real problem with this approach is, that we would need to connect to an existing WiFi network which usually use a passphrase for encryption and we do not have a possibility for the end user to enter the password.

References

[1] ACT GmbH, “Laufschrift Serie TEBE”.
[2] ACT GmbH, “Laufschrift Serie Proline”.
[3] eurolite, “ESN 7x80 5mm LED-Laufschrift”.
[4] Elinux.org, “RPi Low-level peripherals”, https://elinux.org/Rpi_Low-level_peripherals#General_Purpose_Input.2FOutput_.28GPIO.29, 2019.
[5] AZ-Delivery, “4 x 64er LED Matrix Display”, https://www.az-delivery.de/products/4-x-64er-led-matrix-display?ls=de, 12. 01. 2019.
[6] Adafruit, “Adafruit Mini 8x8 LED Matrix w/I2C Backpack - Yellow”, https://www.adafruit.com/product/870, 12. 01. 2019.
[7] Adafruit, “32x32 RGB LED Matrix Panel - 6mm pitch”, https://www.adafruit.com/product/1484, 15. 01. 2019.
[8] Adafruit, “Adafruit Python LED Backpack”, https://github.com/adafruit/Adafruit_Python_LED_Backpack, 14. 01. 2019.