

Projeto Final: Sistema Automatizado de Ponto Eletrônico

Arnoldo Thiago Monteiro Lima¹ – 14/0016660
Dario Descartes Amaral Moreira¹ – 14/0018875

Programa de Graduação em Engenharia Eletrônica, Faculdade Gama
Universidade de Brasília
Gama, DF, Brasil
email: arnoldo.thiago@gmail.br, dario.des96@gmail.com

Resumo—Trabalho final da disciplina de Sistemas Embarcados que consiste na implementação de um ponto eletrônico *NFC* com suporte a câmera para suprir a dificuldade que gestores de empresas pequenas como *StartUps* e empresas juniores possuem. Para tal foi utilizado a *Raspberry Pi 3* juntamente com o sensor *NFC* PN532 e o módulo de câmera da *Raspberry Pi* Câmera. Tal sistema obteve os resultados esperados para a situação proposta; conseguindo ler o cartão *NFC*, subindo o código do cartão para uma planilha do Google Forms, tirando uma foto do usuário e realizando o *update* para o Google Drive.

Index Terms—Sistemas Embarcados, Sistemas Operacionais Embarcados, Raspberry Pi, NFC, I2C, PiCâmera.

I. INTRODUÇÃO

Com o intuito de suprir a necessidade de pequenas empresas no que toca gerir dados de assiduidade de seus membros a equipe implementou uma solução barata e eficiente de ponto eletrônico.

Para auxiliar os gestores dessas empresas, assim como permitir que sejam feitas inferências a cerca da assiduidade de seus funcionários, é necessário uma forma simples e intuitiva de visualização dos dados. Por isso é importante o processamento correto dos dados de forma a permitir a melhor extração de informações, para um melhor entendimento do contexto observado.

O objetivo do projeto final consiste na elaboração de um ponto eletrônico totalmente automatizado que coleta as entradas e saídas de cada funcionário; utilizando tags *NFC* personalizadas; realiza o upload automático dos dados para um banco de dados online, junto com uma foto de quem utilizou a tag tirada por uma câmera; que podem ser acessados posteriormente pelo usuário.

II. DESENVOLVIMENTO

A. Descrição do Hardware

1) **Hardware do SO:** O hardware do Sistema Operacional utilizado para o projeto foi a *Raspberry Pi 3 Model B*. Essa placa possui processador Broadcom BCM2837 de 64 bits e *clock* de 1,2GHz. Já tem integrado em sua placa WiFi 802.11n e Bluetooth 4.1, sem que haja necessidade de adaptadores adicionais. Possui 1G de memória RAM, adaptador para cartão microSD e GPU Videocore IV 3D.

Além disso possui um conector de vídeo HDMI, 4 portas USB 2.0, um conector Ethernet, uma interface para câmera (CSI), uma interface para display (DSI), conector de áudio e vídeo e GPIO¹ de 40 pinos.

2) **Sensor NFC:** O sensor *NFC* utilizado foi o Módulo Leitor *RFID NFC* PN532, que lê até uma distância de 7cm. Esse leitor suporta cartões Mifare 1K, 4K, CD97BX, IRT5001, RCS_860 e outros. Possui os padrões de conexões I2C, SPI ou HSU, com conversor lógico embutido na placa, sendo compatível com Arduino, Raspberry Pi e *NFC* Android. Sua tensão de operação é de 5V.

3) **Protocolo de Comunicação:** O protocolo de comunicação entre o Leitor *NFC* e a *Raspberry Pi* utilizado foi o I2C. Os pinos GND, VCC, SDA e SCL do PN532 foram conectados nos pinos GPIO da Raspberry 6, 4, 3 e 5, respectivamente. A comunicação I2C utiliza duas linhas, uma de dados (SDA), onde trafegam os endereços e as informações compartilhadas entre o dispositivo mestre e o escravo, e uma conexão de clock (SCL), em que a *Raspberry* fornece o *clock* para o PN532.

4) **Raspberry Pi Câmera:** A câmera utilizada foi o módulo *Raspberry Pi 5MP* Câmera, essa câmera utiliza a interface CSI da *Raspberry Pi* diretamente com o cabo flat de 15 vias e possui um ângulo de visão de 65°C.

5) Bill Of Materials – BOM:

- Sensor NFC PN532;
- *Raspberry Pi 3 Model B*;
- Módulo *Raspberry Pi 5MP* Câmera;
- Cabos jumpers²;
- Cabo flat de 15 vias³;
- Cabo Ethernet⁴;
- Fonte de alimentação de 5V e 3A;
- Opcional: Cabo HDMI para ver a interface do sistema;

¹General Purpose Input/Output.

²Conexão do Sensor *NFC* com a *Raspberry Pi 3*

³Conexão entre a câmera e a *Raspberry Pi 3*

⁴Possibilita o desenvolvimento por SSH

- Opcional: Teclado e mouse para um desenvolvimento direto na *Raspberry Pi*.

6) Diagrama de Blocos do Sistema:

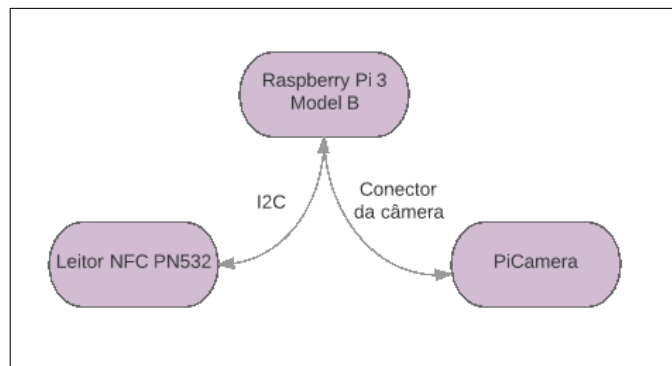


Figura 1. Diagrama de blocos do Sistema.

B. Descrição do Software

1) **SO Embarcado:** No projeto foi utilizado uma distribuição Linux denominada *Raspbian*⁵ que é uma variante da distribuição *Debian*, porém otimizada para o conjunto de instruções *ARM* que constitui o processador do *Raspberry Pi 3* assim como toda a família de *Raspberrys Pi* [1].

Assim como instruções otimizadas para processadores *ARM* está distribuição possui mais de 35.000 pacotes *.deb* que já estão pré-compilados e que podem ser facilmente instalados no hardware da *Raspberry Pi* e ambos estão otimizados para os processadores *ARM11* incluso na placa de desenvolvimento [1].

O *SO raspbian* possui duas variantes sendo uma delas uma versão do *SO* com interface gráfica de *desktop* e uma outra versão *Lite* que possui somente um terminal de comandos para a interação usuário e máquina. Inicialmente o projeto começou a ser desenvolvido em um na versão do *SO* que possui interface gráfica, uma vez que o desenvolvimento estava ocorrendo diretamente na placa. Porém, a eficiência estava baixa. Fazendo com que o desenvolvimento migrasse para um desenvolvimento via *SSH*. Entretanto, os alunos optaram por não modificar a versão do sistema para sua versão *Lite*, uma vez que, sua versão *desktop* servia para o propósito devido [1], [2].

2) **Bibliotecas Utilizadas:** O programa implementado contou com a utilização de algumas bibliotecas como a biblioteca para utilizar do leitor *NFC* [3], a biblioteca para utilização da *PiCâmera* da *Raspberry Pi* e a biblioteca *gdrive* [4] para realizar o *upload* de imagens para o Google Drive.

3) **Protocolo de Comunicação e Leitura do Sensor NFC:** O código conta com funções da biblioteca *libnfc* para realização do *polling*⁶ do leitor *NFC*, caso nenhum cartão magnético seja lido por um período de trinta segundos o programa se encerra.

⁵*Raspberry Pi + Debian*.

⁶Leitura do cartão do usuário por meio do sensor *NFC*.

A comunicação com o leitor *NFC* é realizado por meio do protocolo de comunicação *I2C*⁷. Em momentos onde não foi possível estabelecer a comunicação com o leitor *NFC* ou abrir a biblioteca *libnfc*, o programa se encerra com um erro, retornando -1.

A função para leitura do cartão magnético retorna uma *string* com diversas informações do cartão, entre elas o *UID*⁸. Realizou-se uma manipulação de *string* para obtenção do *UID* do cartão apenas.

4) **Comunicação com o Banco de Dados Online – Google Drive:** O Google Drive é utilizado como um banco de dados online, assim temos que o passo seguinte, ao ler o cartão *NFC*, é o envio do *UID* para o Google Forms. Realizou-se um questionário no Google Forms com um campo de entrada para inserir o *UID*. Descobriu-se o nome da entrada para esse campo (*entry.859460154* nesse caso). Criou-se uma planilha para se receber os dados e se obteve o link para essa página <https://docs.google.com/forms/d/1tfhcdKmafJVPrrW994O3eSQbdx4bUbeljjHI3thFUG/formResponse>. Por fim, mandou-se a *string* com o *UID* do cartão através do comando chamado pela função *system()*: `sudo curl9 -k -data "entry.859460154=%s"10https://docs.google.com/forms/d/1tfhcdKmafJVPrrW994O3eSQbdx4bUbeljjHI3thFUG/formResponse`. Dessa forma o *UID* do cartão lido é enviado para o Google Forms e organizado em uma planilha juntamente com seu horário de envio.

5) **Raspberry PiCâmera:** A etapa da câmera consiste na obtenção da imagem pela *PiCâmera*. Essa captação é realizada com o seguinte comando chamado pelo função *system()*: `sudo raspistill -t 5000 -o image.jpg`. Esse comando obtém imagens por cinco segundos e salva a última imagem obtida. Essa imagem é então enviada para o Google Drive.

6) Fluxograma de Dados:

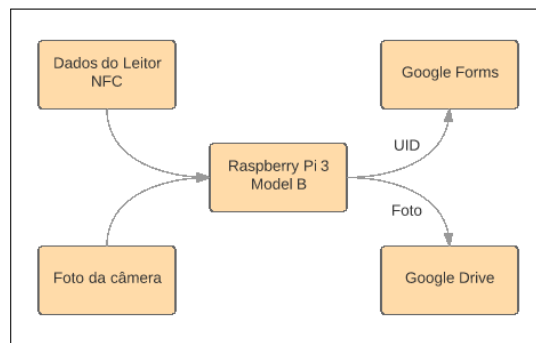


Figura 2. Fluxograma dos sinais do sistema.

⁷*Inter-Integrated Circuit*.

⁸*Unique Identity number - UID*

⁹*curl* é uma ferramenta para transferir dados de ou para um servidor, usando um dos protocolos suportados (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET e TFTP). O comando é projetado para funcionar sem a interação do usuário [5].

¹⁰*string* contendo *UID* no espaço referente a *%s*.

III. RESULTADOS

Após a montagem de todo o sistemas com as conexões entre a *Raspberry* com a câmera e o leitor *NFC* realizaram-se teste para aferir o funcionamento do sistema. Após o arquivo *trab_final.c* ter sido devidamente compilado, obteve-se o executável do programa.

Ao se iniciar o programa foram inicialmente obtidas as confirmações de que foi possível se conectar ao leitor *NFC* e abrir a biblioteca *libnfc*, imediatamente depois se observou a espera (*polling*) pela leitura de algum cartão *RFID*.

Observou-se que a aproximação do cartão *RFID* precisou ser menor do que *7cm* em alguns casos, sendo necessário encostar o cartão no leitor em alguns momentos. Após a leitura do cartão observou-se as informações impressas na tela do computador (UID e tipo de cartão magnético). Após essa leitura o programa fez o *upload* do UID para o Google Forms, sendo que o tempo de espera deste *upload* é pequeno. Ao se olhar as respostas no Google Forms foi possível averiguar que esta funcionalidade do programa estava correta, tendo sido gerados o horário de envio na planilha do Google e um gráfico com a porcentagem de vezes que cada cartão foi passado no leitor.

Logo após o envio para o Google Forms foi tirada uma foto com a câmera, tendo sido o tempo de espera para a foto de 5 segundos. Em duas ocasiões o programa não reconheceu a câmera, porém após um ajuste do cabo a conexão foi restabelecida. O *upload* da foto para o Google Drive foi o processo mais demorado dentro do funcionamento de todo o sistema. Foi possível averiguar o correto funcionamento desse *upload*, sendo que todas as imagens tiradas pela câmera foram encontradas na pasta do Google Drive.

Após o teste com algumas leituras foi realizada a saída do programa pressionando *Ctrl + C*, em nenhum momento a saída do programa demonstrou má funcionalidade. Como um todo todas as funcionalidades foram implementadas corretamente.

IV. CONCLUSÃO

O problema de verificar a presença de funcionários nos seus horários de trabalho pode ser diminuído com a verificação de seu cartão *RFID* no início e no fim de seu expediente, assim como a verificação da foto tirada nesse processo para confirmação de que o dono do cartão que o passou. Levando em consideração o problema abordado e a solução proposta, o projeto conseguiu atender aos requisitos propostos e permitir um maior controle dos horários dos funcionários de uma empresa guardadas as devidas limitações do sistema, que são a incapacidade de averiguar a presença entre o início e o fim do expediente e a incapacidade de reconhecer a face do funcionário automaticamente.

REFERÊNCIAS

- [1] M. Thompson and P. Green, "Raspbian," Nov. 2017. [Online]. Available: <https://www.raspbian.org/>
- [2] R. P. FOUNDATION, "Raspbian download," Nov. 2017. [Online]. Available: <https://www.raspberrypi.org/downloads/raspbian/>
- [3] P. Teuwen, "Platform independent near field communication (nfc) library," Sep. 2017. [Online]. Available: <https://github.com/nfc-tools/libnfc>
- [4] P. Rasmussen, "Google drive cli client," Sep. 2017. [Online]. Available: <https://github.com/prasmussen/gdrive>

- [5] D. Stenberg, "curl.1 the man page," Nov. 2017. [Online]. Available: <https://curl.haxx.se/docs/manpage.html>

APÊNDICE

CÓDIGO PRINCIPAL .C – TRAB_FINAL.C

```

1  #ifndef HAVE_CONFIG_H
2  #   include "config.h"
3  #endif // HAVE_CONFIG_H
4
5  #include <err.h>
6  #include <inttypes.h>
7  #include <signal.h>
8  #include <stdio.h>
9  #include <stddef.h>
10 #include <stdlib.h>
11 #include <string.h>
12
13 #include <nfc/nfc.h>
14 #include <nfc/nfc-types.h>
15
16 #include "utils/nfc-utils.h"
17
18 #define MAX_DEVICE_COUNT 16
19
20 static nfc_device *pnd = NULL;
21 static nfc_context *context;
22
23 // Funcoes pra Mandar para o Google Forms
24 char *sgets(char * str, int num, char **input){
25     char *next = input;
26     int numread = 0;
27
28     while (numread + 1 < num && *next) {
29         int isnewline = (*next == '\n');
30         *str++ = *next++;
31         numread++;
32         // newline terminates the line but is included
33         if (isnewline)
34             break;
35     }
36
37     if (numread == 0)
38         return NULL; // "eof"
39
40     // must have hit the null terminator or end of line
41     *str = '\0'; // null terminate this string
42     // set up input for next call
43     *input = next;
44     return str;
45 }
46
47 void remove_n(char* source){
48     char* i = source;
49     char* j = source;
50     int n;
51     while(*j != ':')
52         j++;
53     j+=2;
54     while(*j != 0)
55     {
56         *i = *j++;
57         if(*i != '\n')
58             i++;
59     }
60     *i = 0;
61 }
62
63 void get_uid(char * str, int num, char **input){
64     int i;
65     for(i=0; i<3; i++){
66         sgets(str, 200, input);
67     }
68     remove_n(str);
69 }
70
71 // Funcoes de Polling
72 static void stop_polling(int sig){
73     (void) sig;

```

```

74     if (pnd != NULL)
75         nfc_abort_command(pnd);
76     else {
77         nfc_exit(context);
78         exit(EXIT_FAILURE);
79     }
80 }
81
82 static void
83 print_usage(const char *progname){
84     printf("usage: %s [-v]\n", progname);
85     printf("    -v\t verbose display\n");
86 }
87
88 int main(int argc, const char *argv[]){
89     bool verbose = false;
90
91     signal(SIGINT, stop_polling);
92
93     // Display libnfc version
94     const char *acLibnfcVersion = nfc_version();
95
96     printf("%s usa o libnfc %s\n", argv[0], acLibnfcVersion);
97     if (argc != 1) {
98         if ((argc == 2) && (0 == strcmp("-v", argv[1]))) {
99             verbose = true;
100         } else {
101             print_usage(argv[0]);
102             exit(EXIT_FAILURE);
103         }
104     }
105
106     const uint8_t uiPollNr = 20;
107     const uint8_t uiPeriod = 2;
108     const nfc_modulation nmModulations[5] = {
109         { .nmt = NMT_ISO14443A, .nbr = NBR_106 },
110         { .nmt = NMT_ISO14443B, .nbr = NBR_106 },
111         { .nmt = NMT_FELICA, .nbr = NBR_212 },
112         { .nmt = NMT_FELICA, .nbr = NBR_424 },
113         { .nmt = NMT_JEWEL, .nbr = NBR_106 },
114     };
115     const size_t szModulations = 5;
116
117     nfc_target nt;
118     int res = 0;
119     char *s;
120     char uid[200], send[200];
121     int i=0;
122
123
124     while(1){
125         nfc_init(&context);
126         if (context == NULL) {
127             ERR("Nao foi possivel iniciar a biblioteca libnfc (malloc)");
128             exit(EXIT_FAILURE);
129         }
130
131         pnd = nfc_open(context, NULL);
132
133         if (pnd == NULL) {
134             ERR("%s", "Nao foi possivel abrir o dispositivo NFC.");
135             nfc_exit(context);
136             exit(EXIT_FAILURE);
137         }
138
139         if (nfc_initiator_init(pnd) < 0) {
140             nfc_perror(pnd, "nfc_initiator_init");
141             nfc_close(pnd);
142             nfc_exit(context);
143             exit(EXIT_FAILURE);
144         }
145
146         printf("Leitor NFC: %s aberto\n", nfc_device_get_name(pnd));
147         printf("Esperando leitura do cartao...\n");
148         if ((res = nfc_initiator_poll_target(pnd, nmModulations, szModulations, uiPollNr, uiPeriod, &nt)) < 0) {
149             nfc_perror(pnd, "nfc_initiator_poll_target");
150             nfc_close(pnd);

```

```
151 nfc_exit(context);
152 exit(EXIT_FAILURE);
153 }
154
155 if (res > 0) {
156     str_nfc_target(&s, &nt, verbose);
157     printf("%s", s);
158     get_uid(uid,200,&s);
159
160     printf("Enviando dados para o Google Forms...\n");
161     sprintf(send,"sudo curl -k --data \"entry.859460154=%s\" https://docs.google.com/forms/d/1
162         tfhdKmafJVPrW994O3eSQbdox4bUbeljjHI3thFUg/formResponse > log_out.txt",uid);
163     system(send);
164
165     printf("Sorria para a foto :)!\n");
166     sprintf(send,"sudo raspistill -t 5000 -o image%d.jpg",i);
167     system(send);
168
169     printf("Enviando foto para o Google Drive...\n");
170     sprintf(send,"gdrive upload -p 1MmsFA-qAY-Ff6pUxvZ7xo9m2IK7E7oUH image%d.jpg",i);
171     system(send);
172     i++;
173
174     printf("Esperando cartao ser removido...");
175     fflush(stdout);
176     while (0 == nfc_initiator_target_is_present(pnd, NULL)) {}
177     nfc_perror(pnd, "nfc_initiator_target_is_present");
178     printf("feito.\n");
179 } else {
180     printf("nenhum alvo encontrado.\n");
181 }
182
183 nfc_close(pnd);
184 nfc_exit(context);
185 system("clear");
186 }
187
188 exit(EXIT_SUCCESS);
189 }
```

BUILD.SH

```
1 #!/bin/sh
2 touch log_out.txt
3 sudo gcc -o trab_final -lnfc -I../libnfc trab_final.c ../libnfc/Utils/nfc-utils.o
4 ls -la
```

CLEAR.SH

```
1 #!/bin/sh
2 sudo rm -f *.jpg log_out.txt trab_final
3 ls -la
```