Projeto Final: Sistema Automatizado de Ponto Eletrônico

Arnoldo Thiago Monteiro Lima[1] – 14/0016660
Dario Descartes Amaral Moreira1[1] -– 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* PiCâmera. 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 BroadcomBCM2837 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 GPIO[1] 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, Raspbery 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 jumpers[2];
- Cabo flat de 15 vias[3];
- Cabo Ethernet[4];
- Fonte de alimentação de $5V$ e $3A$;
- Opcional: Cabo HDMI para ver a interface do sistema;

[1]General Purpose Input/Output.
[2]Conexão do Sensor *NFC* com a *Raspberry Pi 3*
[3]Conexão entre a câmera e a *Raspberry Pi 3*
[4]Possibilita o desenvolvimento por SSH

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

6) ***Diagrama de Blocos do Sistema****:*

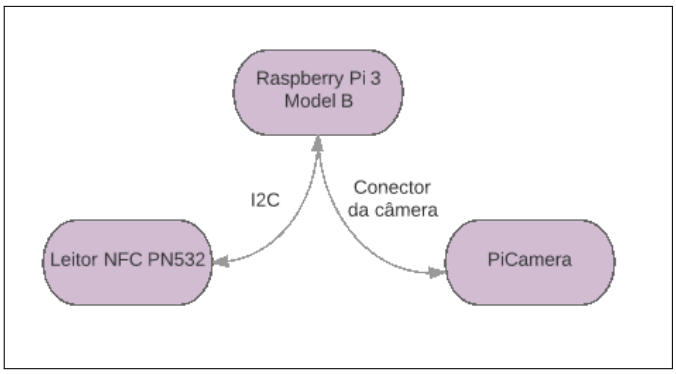

Figura 1. Diagrama de blocos do Sistema.

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.

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/1tfhcdKmafJVPrrW994O3eSQbdox4bUbeljjHI3thFUg/formResponse. Por fim, mandou-se a string com o UID do cartão através do comando chamado pela função system(): sudo curl[9] -k –data "entry.859460154=%s[10]"https://docs.google.com/forms/d/1tfhcdKmafJVPrrW994O3eSQbdox4bUbeljjHI3thFUg/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 obtêm imagens por cinco segundos e salva a última imagem obtida. Essa imagem é então enviada para o Google Drive.

6) ***Fluxograma de Dados****:*

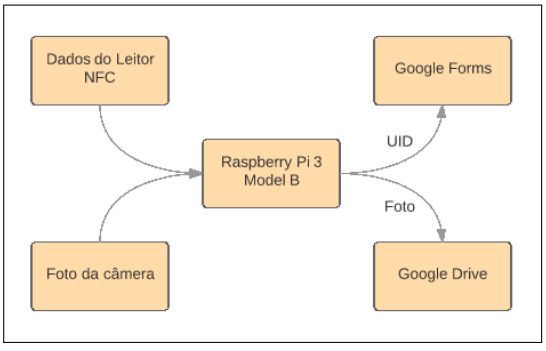

Figura 2. Fluxograma dos sinais do sistema.

[5]***Rasp****berry Pi +* ***Debian****.*

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

[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

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

```c
#ifdef HAVE_CONFIG_H
#  include "config.h"
#endif // HAVE_CONFIG_H

#include <err.h>
#include <inttypes.h>
#include <signal.h>
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <string.h>

#include <nfc/nfc.h>
#include <nfc/nfc-types.h>

#include "utils/nfc-utils.h"

#define MAX_DEVICE_COUNT 16

static nfc_device *pnd = NULL;
static nfc_context *context;

// Funcoes pra Mandar para o Google Forms
char *sgets(char * str, int num, char **input){
  char *next = *input;
  int  numread = 0;

  while (numread + 1 < num && *next) {
  int isnewline = (*next == '\n');
  *str++ = *next++;
  numread++;
  // newline terminates the line but is included
  if (isnewline)
  break;
  }

  if (numread == 0)
  return NULL;  // "eof"

  // must have hit the null terminator or end of line
  *str = '\0';  // null terminate this tring
  // set up input for next call
  *input = next;
  return str;
}

void remove_n(char* source){
  char* i = source;
  char* j = source;
  int n;
  while(*j!=':')
  j++;
  j+=2;
  while(*j != 0)
  {
  *i = *j++;
  if(*i != '\n')
  i++;
  }
  *i = 0;
}

void get_uid(char * str, int num, char **input){
int i;
for(i=0;i<3;i++){
sgets(str,200,input);
}
remove_n(str);
}

// Funcoes de Polling
static void stop_polling(int sig){
  (void) sig;
```

```
  if (pnd != NULL)
  nfc_abort_command(pnd);
  else {
  nfc_exit(context);
  exit(EXIT_FAILURE);
  }
}

static void
print_usage(const char *progname){
  printf("usage: %s [-v]\n", progname);
  printf("  -v\t verbose display\n");
}

int main(int argc, const char *argv[]){
bool verbose = false;

signal(SIGINT, stop_polling);

// Display libnfc version
const char *acLibnfcVersion = nfc_version();

printf("%s usa o libnfc %s\n", argv[0], acLibnfcVersion);
if (argc != 1) {
if ((argc == 2) && (0 == strcmp("-v", argv[1]))) {
verbose = true;
} else {
print_usage(argv[0]);
exit(EXIT_FAILURE);
}
}

const uint8_t uiPollNr = 20;
const uint8_t uiPeriod = 2;
const nfc_modulation nmModulations[5] = {
{ .nmt = NMT_ISO14443A, .nbr = NBR_106 },
{ .nmt = NMT_ISO14443B, .nbr = NBR_106 },
{ .nmt = NMT_FELICA, .nbr = NBR_212 },
{ .nmt = NMT_FELICA, .nbr = NBR_424 },
{ .nmt = NMT_JEWEL, .nbr = NBR_106 },
};
const size_t szModulations = 5;

nfc_target nt;
int res = 0;
char *s;
char uid[200],send[200];
int i=0;


while(1){
nfc_init(&context);
if (context == NULL) {
ERR("Nao foi possivel iniciar a biblioteca libnfc (malloc)");
exit(EXIT_FAILURE);
}

pnd = nfc_open(context, NULL);

if (pnd == NULL) {
ERR("%s", "Nao foi possivel abrir o dispositivo NFC.");
nfc_exit(context);
exit(EXIT_FAILURE);
}

if (nfc_initiator_init(pnd) < 0) {
nfc_perror(pnd, "nfc_initiator_init");
nfc_close(pnd);
nfc_exit(context);
exit(EXIT_FAILURE);
}

printf("Leitor NFC: %s aberto\n", nfc_device_get_name(pnd));
printf("Esperando leitura do cartao...\n");
if ((res = nfc_initiator_poll_target(pnd, nmModulations, szModulations, uiPollNr, uiPeriod, &nt))  < 0) {
nfc_perror(pnd, "nfc_initiator_poll_target");
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
    tfhcdKmafJVPrrW994O3eSQbdox4bUbeljjHI3thFUg/formResponse > log_out.txt",uid);
162 system(send);
163
164 printf("Sorria para a foto :)!\n");
165 sprintf(send,"sudo raspistill -t 5000 -o image%d.jpg",i);
166 system(send);
167
168 printf("Enviando foto para o Google Drive...\n");
169 sprintf(send,"gdrive upload -p 1MmsFA-qAY-Ff6pUxvZ7xo9m2IK7E7oUH image%d.jpg",i);
170 system(send);
171 i++;
172
173 printf("Esperando cartao ser removido...");
174 fflush(stdout);
175 while (0 == nfc_initiator_target_is_present(pnd, NULL)) {}
176 nfc_perror(pnd, "nfc_initiator_target_is_present");
177 printf("feito.\n");
178 } else {
179 printf("nenhum alvo encontrado.\n");
180 }
181
182 nfc_close(pnd);
183 nfc_exit(context);
184 system("clear");
185 }
186
187 exit(EXIT_SUCCESS);
188 }
```

BUILD.SH

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

CLEAR.SH

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