Relatório Final

Espaço de Coworking inteligente IoT

Hércules Ismael de Abreu Santos
Faculdade do Gama
Universidade de Brasilia
Gama – DF, Brasil
ismael-456@hotmail.com

Pedro Henrique Brito Checchia
Faculdade do Gama
Universidade de Brasília
Gama – DF, Brasil
Pedrobcbr2@gmail.com

***Abstract*— Aiming a better organization coworking spaces, one can use a system with IoT(Internet of Things) features to organize the tables, so that it can control the people that enter in the space by an identification, and schedule the rooms by a system that can be accessed by a smartphone. For this it is used the Raspberry PI development board for this project, to control the processes in this project.**

Keywords—Smart table; coworking; Raspberry PI; IoT; Internet of Things

***Resumo:* Visando a melhor organização de espaços de coworking, pode-se utilizar um sistema com recursos IoT(Internet das coisas) para organizar as salas, de modo a controlar as pessoas que entram por meio de uma identificação, e agendar as salas por meio de um sistema que pode ser acessado no smartphone. Para isso a placa de desenvolvimento Raspberry PI será usada neste projeto, para controlar os processos neste projeto.**

Palavras-chave: Bancada Inteligente; coworking; Raspberry PI; IoT; Internet das Coisas

I. Justificativa

Este projeto visa auxiliar pessoas a encontrarem vagas nas salas de coworking em horários que melhor lhes favorecer, além de monitorar as pessoas que entram na sala por meio de uma identificação gerada pelo estabelecimento que abriga os espaços de coworking.

Para melhorar a eficiência do projeto, ele contará com um bot no telegram onde os usuários do espaço de coworking terão a possibilidade de visualizar as salas disponíveis, e agendar a utilização destas salas, de modo a poder organizar com mais eficiência a utilização destes espaços.

II. Requisitos

Hardware:

- Raspberry pi 3
- Câmera para identificar o Qr code
- Módulo Relé 3.3 V
- Trava para porta

Software:

- Programação python que possa interagir com bots do Telegram
- Programação em C para identificar o QR code na Raspberry pi

III. Objetivos

- Auxiliar na organização dos espaços de coworking;
- Identificar cada um dos usuários do ambiente.
-Controlar os recursos disponíveis na sala para serem disponibilizados apenas para quem agendar o lugar;

IV. Benefícios

Os benefícios deste projeto incluem a maior praticidade de alugar os espaços de coworking, gerando uma maior comodidade do usuário que pode acessar as salas disponíveis e agendar estas conforme a necessidade em qualquer lugar em que ele possa acessar internet através do celular. Também gerará uma maior segurança e robustez para o ambiente de coworking, pois será automatizada a identificação das pessoas que entram nas salas e os acessos aos recursos deste ambiente.

V. Revisão Bibliográfica

Para verificar a viabilidade deste projeto, foi consultada a bibliografia para analisar projetos com aplicações semelhantes, como a utilização da raspberry em conjunto com o telegram como uma forma de IoT, ou mesmo a utilização desta placa para ler códigos de barra e códigos QR.

Com esta análise, encontrou-se vários projetos que utilizam dos recursos aqui propostos que obtiveram sucesso, como exemplo pode-se citar a utilização da Raspberry PI para automatização residencial utilizando telegram como uma forma de IoT. Este projeto da Índia tem como principal

objetivo demonstrar a capacidade da raspberry como um recurso para automatizar casas. Para isso utiliza-se o telegram como um meio de se comunicar com o controlador por meio de um BOT, que responde a comandos do usuário,e a partir destes a placa controla suas portas GPIO, abrindo uma gama de possibilidades.

Outro projeto que traz uma aplicação da raspberry com o telegram é o projeto de uma fazenda hidropônica inteligente. Neste projeto do Instituto de Tecnologia de Bandung na Indonésia, a ideia é montar um sistema que forneça informações da plantação remotamente aos fazendeiros, como umidade, luminosidade, temperatura e pH do solo. Este projeto tem ainda o diferencial de utilizar o telegram como sistema social que não só faz a comunicação entre o fazendeiro e o BOT do telegram para controlar a plantação, como também promove a interação entre os fazendeiros por meio de um grupo criado no telegram que é composto dos fazendeiros e do BOT do telegram.

Ainda outra característica que pode ser observada na bibliografia é o reconhecimento de códigos de barras e QR code, que pode se encontrar vários exemplos, desde a utilização de uma câmera para captura de QR code (Shyam, 2019), a reconhecimento de código de barras e QR no mesmo sistema. Um exemplo interessante que pode ser citado é a utilização da raspberry para reconhecimento de código de barras em imagens panorâmicas, de modo a se identificar o código sem precisar utilizar o reconhecimento tradicional unidimensional.

Desta forma temos vários projetos anteriores que sustentam a viabilidade deste projeto.

VI. Resultados

Para a primeira fase do projeto, foram testados os sensores a serem utilizados, de modo a conectá-los com o controlador e fazer a aquisição dos dados pertinentes ao projeto.

Em especial para este projeto, testou-se um leitor de código de barras e QR Code. Este leitor foi feito com uma webcam, e um algoritmo que processa a imagem de modo a obter o código referente ao código de barras ou ao QR Code. Para isso, utilizou-se um programa disponibilizado pela DataSymbol[6]. O programa de nome “dsreader” foi instalado na raspberry pi 3 , utilizada no projeto, com o sistema operacional raspbian instalado.

A partir deste programa conseguiu-se ler códigos de barras e QR Codes com sucesso, como ilustra a imagem:

Figura 1 - Execução do Programa dsreader

Nesta imagem vemos o terminal da raspberry, o programa é executado na primeira linha. Nas linhas subsequentes temos o reconhecimento de QR Codes e a partir do 7 reconhecimento temos códigos de barra. O QR Code é sinalizado pela palavra QRCode no início, e o código de barras pela palavra Interleaved. O código após os dois pontos é o código que identifica o QR Code, ou o código de barras. Já o número dentro do parênteses se refere ao tempo que levou para ler o cada um dos códigos.

Para a segunda fase do projeto, produziu-se um código para se utilizar do programa de reconhecimento de código de barras e QR Code para comparar os resultados com um arquivo txt (Anexos), assim verificando se o código reconhecido é de uma pessoa autorizada a utilizar a sala de coworking ou não, de modo que se for autorizada, a raspberry acionará uma de suas portas de gpio para acionar a trava da porta e liberar a energia do lugar.

Para a terceira fase do projeto, foram introduzidos relés de acionamento com a raspberry, definida a topologia necessária para fazer o controle da fechadura e da energia do espaço de coworking. Foi utilizado um módulo relé com alimentação de 5 V e sinal de 3.3 V conectado nas gpios da raspberry pi, sendo um canal para a lâmpada e o outro para a trava.

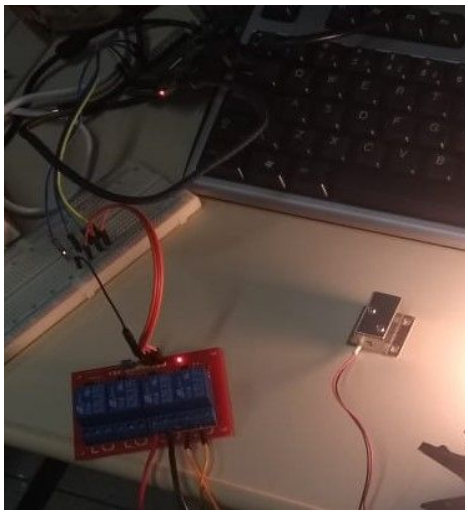

Figura 2 - Módulo Relé

A Quarta Fase do Projeto envolveu a remoção de bugs envolvendo o processo de abertura e fechamento do Dsreader, pois anteriormente o código não era capaz de encerrar a aplicação, fazendo com que o Dsreader rodasse infinitamente. Segue abaixo o trecho de código utilizado para encerrar a aplicação:

```
void mataDsreader(){
  pid_t pidDsreader;
  system("ps -a | grep dsreader > tmp.txt");
  pidDsreader=getPIDdsreader();
  kill(pidDsreader, SIGINT);
  kill(scanner, SIGKILL);
  waitpid(scanner, NULL, 0);
  system("rm tmp.txt");
}
```

Figura 3 - Função para encerrar o Dsreader

Essa função cria um arquivo txt temporário com o pid do dsreader. Feito isso, ela chama uma nova função para tratar a string salva no arquivo. Após adequar a string a um formato de pid e converter para um inteiro, o comando "kill" mata o dsreader, com seu pid e o processo pai. Ao terminar o arquivo temporário é apagado.

A última fase do projeto envolveu a criação de interface gráfica para que o usuário não precisasse utilizar o terminal para interagir com a aplicação.

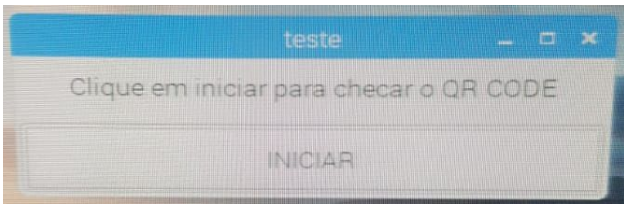

Figura 4 - Tela inicial da aplicação

Ao iniciar o usuário deve clicar no botão "iniciar" e posicionar o QR code de frente para a câmera. Ao verificar o código, se ele estiver cadastrado no arquivo txt, o programa manda a mensagem "Porta Aberta" e a raspberry Pi abre as gpios para o relé da trava por 5 segundos, e da lâmpada até que o usuário não deseje mais utilizar. As figuras abaixo mostram a trava e a lâmpada acionadas e a resposta do programa:

Figura 5 - Acionamento da lâmpada

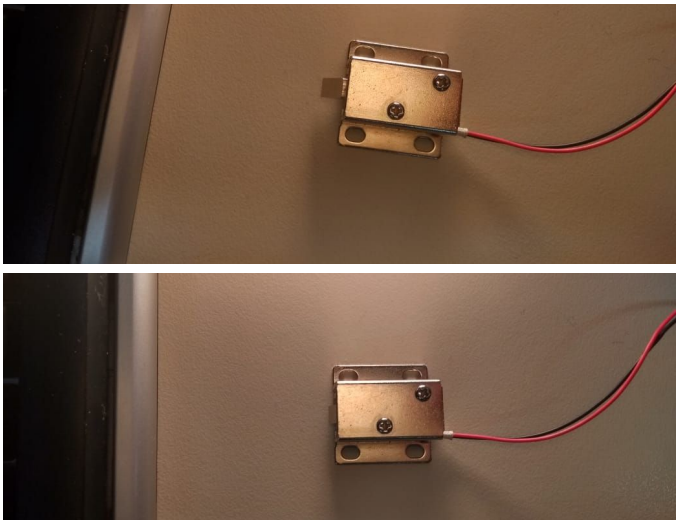

Figura 6 - Trava antes e depois do acionamento

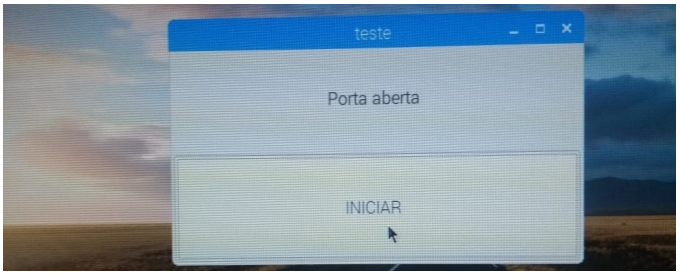

Figura 7 - QR code reconhecido

Se o QR code não for reconhecido o programa manda a mensagem “Porta Fechada”, sendo assim, nenhuma gpio é aberta.

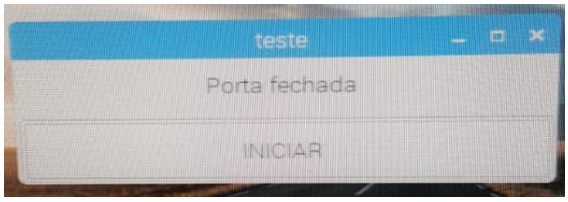

Figura 8 - QR code não reconhecido

Ao receber o resultado, pode- se realizar uma nova tentativa caso por algum motivo, um QR code válido não for reconhecido.

VII. Conclusão

Com isso conclui-se o relatório do projeto de um espaço de co-working inteligente. Ao problema do espaço de co-working conseguiu-se atingir avanços na área de identificação dos usuários, e foi alcançada uma interface para se utilizar o espaço com facilidade pelos usuários do espaço. Para projetos futuros, pode-se melhorar o projeto no aspecto da facilidade de utilização dos funcionários do local e na utilização dos consumidores, aplicando-se uma interface para o cadastro do QR code dos usuários do espaço, um gerador automático de códigos QR aleatórios, e a criação de um bot no telegram para atender os clientes.

Referências

[1] B. Wang, “Smart Table IoT” fevereiro 2019. Disponível em: <https://www.hackster.io/bruce-wang/smart-table-iot-c89a7b> Acesso em: 28 mar. 2019.

[2] Shyam, “IOT: RaspberryPi + Camera Module – Scan For QR Codes Using Zxing in Java”, agosto 2016. Disponível em: <http://agilerule.blogspot.com/2016/08/iot-raspberrypi-camera-module-scan-for.html> Acesso em: 28 mar. 2019.

[3] VATSA, Vedang Ratan; SINGH, Gopal. Raspberry Pi based Implementation of Internet of Things using Mobile Messaging Application-‘Telegram’. **International Journal of Computer Applications**, v. 145, n. 14, 2016.

[4] DYACHOK, Roman; HRYTSYSHYN, Oleh; SALAMAHA, Sergiy. System of detection and scanning bar codes in panoramic images on raspberry PI. In: **Litteris et Artibus: матеріали**. Видавництво Львівської політехніки, 2017. p. 392-393.

[5] SISYANTO, Robert Eko Noegroho et al. Hydroponic smart farming using cyber physical social system with telegram messenger. In: **2017 International Conference on Information Technology Systems and Innovation (ICITSI)**. IEEE, 2017. p. 239-245.

[6] “Raspberry Pi Barcode Scanner” . Disponível em: <https://datasymbol.com/barcode-scanner/barcode-scanner-for-raspberry-pi/raspberry-pi-barcode-scanner.html > Acesso em : 03 maio 2019.

Apêndice

I) Arquivo txt contendo as leituras do dsreader:

```
 GNU nano 2.7.4              Arquivo: cadastro.txt                 Modificado

[1] QRCode: **dro Henriqu* (0.085 s)
[2] QRCode: **dro Henriqu* (0.081 s)
[3] QRCode: **dro Henriqu* (0.070 s)
[4] QRCode: **dro Henriqu* (0.069 s)
[5] QRCode: **dro Henriqu* (0.031 s)
[6] QRCode: **dro Henriqu* (0.075 s)
[7] QRCode: **dro Henriqu* (0.033 s)
[8] QRCode: **dro Henriqu* (0.068 s)
[9] QRCode: **dro Henriqu* (0.077 s)
[10] QRCode: **dro Henriqu* (0.032 s)
[11] QRCode: **dro Henriqu* (0.076 s)
[12] QRCode: **dro Henriqu* (0.034 s)
[13] QRCode: **dro Henriqu* (0.070 s)
[14] QRCode: **dro Henriqu* (0.031 s)
[15] QRCode: **dro Henriqu* (0.067 s)
[16] QRCode: **dro Henriqu* (0.031 s)
```

II) Arquivo txt contendo os códigos cadastrados:

III) Código Principal:

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>
#include <string.h>
#include <wiringPi.h>
#include <gtk/gtk.h>

pid_t scanner;
#define LedPin 0
#define Rele   2

void mataDsreader();
void le_strings(char *matriz_string);
void extraiCodigo(char *str, char *codigo);
int validaCodigo(char *codigos);
int verificaCodigo(char *codigo);
int getPIDdsreader();

void fecha_programa(GtkWidget *wid, gpointer ptr){
  gtk_main_quit();
}

void le_codigo(GtkWidget *wid, gpointer ptr){
  char codigo[10][100];
  char codigoReal[10][100];
  int sent=1;

  signal(SIGALRM, mataDsreader);
  scanner=fork();

  if(scanner==0){
    system("./dsreader > cadastro.txt");
  }else{
    while(sent){

    gtk_label_set_text(GTK_LABEL(ptr), "Posicione o código
de barras/QR CODE");
    alarm(15);
    pause();
    le_strings(codigo[0]);
    extraiCodigo(codigo[0], codigoReal[0]);
    sent=validaCodigo(codigoReal[0]);
    if(sent){
      gtk_label_set_text(GTK_LABEL(ptr), "Porta fechada");
      digitalWrite(LedPin,LOW);
      digitalWrite(Rele,LOW);


sleep(5);

      return;
}
    }

    if(verificaCodigo(codigoReal[0])){
      gtk_label_set_text(GTK_LABEL(ptr), "Porta fechada");
      digitalWrite(LedPin,LOW);
      digitalWrite(Rele,LOW);
      sleep(5);
    }else{
      gtk_label_set_text(GTK_LABEL(ptr), "Porta aberta");
      digitalWrite(LedPin,HIGH);
      digitalWrite(Rele,HIGH);
                delay(5000);
      digitalWrite(LedPin,LOW);
      delay(15000);
      digitalWrite(Rele,LOW);
    }
  }

  return;
}

int main(int argc, char *argv[]){
  gtk_init(&argc, &argv);

  GtkWidget *win =
gtk_window_new(GTK_WINDOW_TOPLEVEL);
  GtkWidget *btn = gtk_button_new_with_label("INICIAR");
  GtkWidget *label = gtk_label_new("Clique em iniciar para
checar o QR CODE");
  GtkWidget *box = gtk_vbox_new(FALSE, 5);

  g_signal_connect(btn, "clicked",
G_CALLBACK(le_codigo), label);
  g_signal_connect(win, "delete_event",
G_CALLBACK(fecha_programa), NULL);
  signal(SIGALRM, mataDsreader);

  wiringPiSetup();
  pinMode(LedPin,OUTPUT);
  pinMode(Rele,OUTPUT);
```

```
  gtk_box_pack_start(GTK_BOX(box), label, TRUE, TRUE,
0);
  gtk_box_pack_start(GTK_BOX(box), btn, TRUE, TRUE, 0);

  gtk_container_add(GTK_CONTAINER(win), box);

  gtk_widget_show_all(win);
  gtk_main();

  return 0;
}

// Objetivo: Procedimento para fechar Dsreader quando tempo
de execução acabar
// Parâmetro:
// Retorno:
void mataDsreader(){
  pid_t pidDsreader;
  system("ps -a | grep dsreader > tmp.txt");
  pidDsreader=getPIDdsreader();
  kill(pidDsreader, SIGINT);
  kill(scanner, SIGKILL);
  waitpid(scanner, NULL, 0);
  system("rm tmp.txt");
}


// Objetivo: Extrai apenas código da string vinda do arquivo
// Parâmetro: string do arquivo, ponteiro para string do código
// Retorno: código
void extraiCodigo(char *str, char *codigo){
  int linha, i, j;

  for(linha=0; linha<10; linha++){
    for(i=0; *(str + linha*100 + i)!=':' && *(str + linha*100 +
i)!='\0'; i++);

    for(i=i+2, j=0; *(str + linha*100 + i)!='(' && *(str +
linha*100 + i)!='\0'; i++, j++){
      *(codigo + linha*100 + j)=*(str + linha*100 + i);
    }
    *(codigo + linha*100 + j -1)='\0';
  }


}

// Objetivo: Lê 10 linhas do arquivo
// Parâmetro: ponteiro para strings
// Retorno: strings
void le_strings(char *matriz_string){
  FILE *cadastro;
  int i=0;

  cadastro=fopen("cadastro.txt", "r");

  while(!feof(cadastro) && i<10){
     fgets(matriz_string+100*i, 100, cadastro);
     i++;
  }

  fclose(cadastro);

  return;
}

// Objetivo: Compara 10 códigos para testar se eles são iguais
// Parâmetro: strings dos códigos
// Retorno: 0 quando for igual, 1 se for diferente
int validaCodigo(char *codigos){
  int i;
  int resultado=0;

  for(i=0; i<10; i++){
    if(strcmp(codigos, codigos+i*100)!=0){
      resultado=1;
    }
  }

  return resultado;
}

// Objetivo: Procura um código na base de dados
// Parâmetro: código
// Retorno: 0 para sucesso, 1 quando não encontrar
int verificaCodigo(char *codigo){
  FILE *dados;
  int i=0;
  char aux[100];
  int resultado=1;

  dados=fopen("codigo.txt", "r");

  while(!feof(dados)){
     fgets(aux, 100, dados);
     aux[strlen(aux)-1]='\0';
     if(strcmp(aux, codigo)==0){
       resultado=0;
     }
  }

  fclose(dados);

  return resultado;
}

int getPIDdsreader(){
  FILE *arq;
  char pidString[10];
  char aux=' ';
```

```
  int i=0;
  pid_t pid;

  arq=fopen("tmp.txt", "r");
  if(arq==NULL){
    fprintf(stderr, "Erro ao abrir tmp.txt");
    exit(-1);
  }

  while(aux==' '){
    aux=fgetc(arq);
  }


  while(aux!=' ' && !feof(arq)){
    pidString[i]=aux;
    aux=fgetc(arq);
    i++;
  }
  pidString[i]=0;

  pid=atoi(pidString);
  fclose(arq);

  return pid;
}
```