Ponto de Controle 4-Sistemas Embarcados

Sistema de Monitoramento Inteligente

Derick Horrana
Matrícula: 10/0009891
Universidade de Brasília, Campus Gama.
E-mail:derickhorrana12@gmail.com
Brasília, Brasil

Eduardo Henrique
Matrícula: 11/0148011
Universidade de Brasília, Campus Gama.
E-mail: eduardoons@gmail.com
Brasília, Brasil

***Abstract*— This work aims to implement and develop an intelligent residential monitoring system, using Raspberry Pi. For example, when motion is detected, the cameras automatically initiate recording and the Raspberry Pi device alerts the owner of the possible intrusion having a smart phone.**

***Keywords*— Raspberry Pi; monitoring; system; smartphone;**

I. INTRODUÇÃO

Este trabalho tem como objetivo apresentar um sistema de monitoramento residencial que busca minimizar o tempo entre invasão/roubo e o conhecimento do usuário/proprietário. Para o desenvolvimento do projeto, serão empregados uma Raspberry Pi modelo B, o modulo oficial da câmera Raspberry Pi, um sensor de presença PIR. Os componentes serão configurados e testados de modo que, ao ser percebida movimentação no ambiente monitorado, o sistema seja capaz de avisar o usuário com o envio de e-mail com imagem em anexo mostrando o local onde o movimento foi percebido pelo sensor PIR.

A. Estrutura do Relatório

O conteúdo deste trabalho será estruturado em cinco partes onde cada um delas terá um propósito, conforme a descrição a seguir: Na Justificativa é apresentada a motivação e o propósito sobre o tema a ser tratado nesse projeto. Hardware e Software apresenta a proposta em concordância com o estudo realizado, ilustrando o modelo desenvolvido para a resolução do problema. Traz também a aplicação prática do modelo proposto, os custos do projeto e os resultados obtidos com ele.. A quarta parte apresenta os testes e seus resultados, realizados para garantir o funcionamento do sistema. E por fim, a quinta parte apresenta as conclusões finais e as possíveis sugestões para trabalhos futuros.

II. JUSTIFICATIVA

A ausência de um sistema de segurança, nas residências em geral, pode acarretar um tempo de espera muito grande até que o crime seja percebido pela vítima, trazendo ainda mais prejuízos e atrasando a intervenção das autoridades competentes. Dessa forma, a implementação de um sistema de segurança que proporcione os benefícios de uma resposta rápida para a vítima, pode ajudar no reconhecimento do criminoso e servir como material para a ação policial, oferecendo mecanismo adicional na proteção do patrimônio. Se a pessoa estiver em casa ou fora da cidade, a ideia por trás de um sistema de segurança é que ele impede intrusões ao notificar potenciais criminosos e que um sistema de alarme está em uso alertando o usuário ou a empresa de segurança quando o sistema de segurança é violado.

Neste ponto de Controle, será escrito o código principal, para leitura do sensor PIR, temperatura e sensor.

III. HARDWARE E SOFTWARE

1. Materiais Utilizados

Esta seção é reservada para a descrição dos materiais utilizados no projeto, sua função, integração com outros componentes e suas justificativas de uso. Os materiais utilizados foram: sensor de presença, plataforma Raspberry e módulo de câmera com infravermelho, que são apresentados com mais detalhamento na sequência.

2. Sensor *de presença PIR*

Será utilizado também um sensor de presença PIR. O Sensor de Movimento PIR DYP-ME003 consegue detectar o movimento de objetos que estejam em uma área de até 7 metros. Caso algo ou alguém se movimentar nesta área o pino de alarme é ativo. É possível ajustar a duração do tempo de espera para estabilização do PIR através do potenciômetro amarelo em baixo do sensor bem como sua sensibilidade. A estabilização pode variar entre 5-200 seg [3].

Figura 1-Sensor de Movimento Presença PIR.

Os sensores PIR (Passive infrared sensor) captam como o próprio nome diz, luz infravermelha. Eles utilizam a variação dessa radiação no ambiente para identificar quando há movimentos. Por serem passivos eles não emitem nenhum tipo de energia para detecção.

Os sensores PIR são construídos com materiais chamados cristais piroelétricos e contém inclusive uma lente geralmente de plástico (que alguns devem ter pensado ser somente uma proteção). Conforme figura 2.

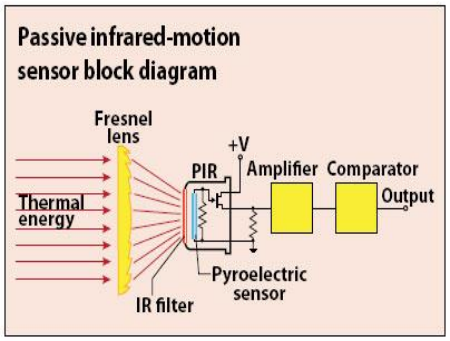

Figura 2-Funcionamento do sensor de presença PIR

A ligação elétrica do sensor PIR é composta dos pinos VCC, GND e OUT. Eles são ligados ao 5V, GND e GPIO17 do Raspberry Pi, como o modelo do Raspeberry é 3.3v conecta diretamente ao Raspberry Pi.

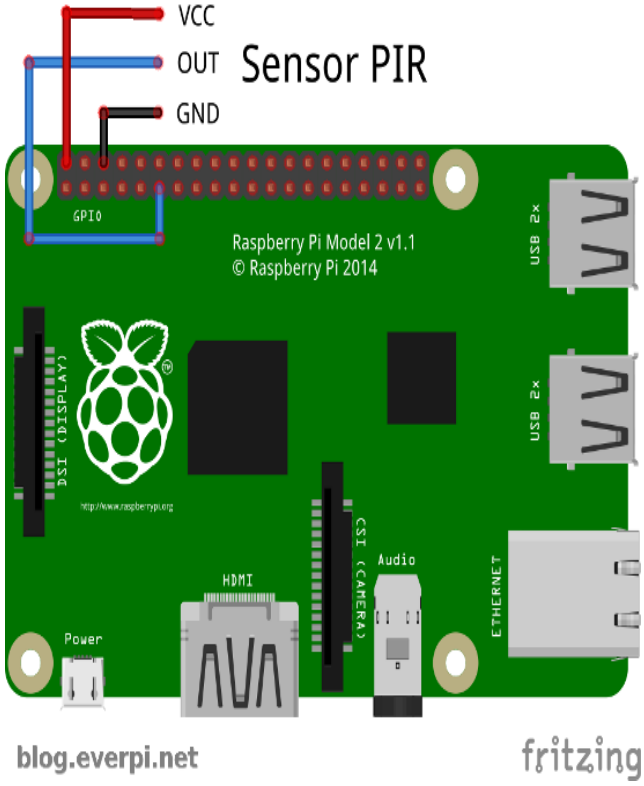

Figura 3 Esquema de ligação dos terminais do Raspberry PI e o sensor PIR

Software para comunicação entre o sensor PIR e o Raspberry Pi utiliza a biblioteca wiringPi e assim identificar quando o pino OUT estiver em HIGH.

A biblioteca WiringPi é uma biblioteca de acesso GPIO baseada em PIN escrita em C para o BCM2835 usado no Raspberry Pi . É lançado sob a licença e é utilizável de C, C ++ e RTB (BASIC), bem como muitos outros idiomas com invólucros adequados. WiringPi inclui um utilitário de linha de comando gpio que pode ser usado para programar e configurar os pinos GPIO.

WiringPiSetup (void): Inicializa o sistema wiringPi e assume que o programa de chamada estará usando o esquema de numeração de pinos wiringPi. Este é um esquema de numeração simplificado que fornece um mapeamento de números de pinos virtuais de 0 a 16 para os números de pino GPIO de Broadcom subjacentes reais. Vericando a figura 4, pode notar a configuração dos pinos.

Void pinMode (pino int, modo int): Define o modo de um pino em INPUT , OUTPUT ou PWM_OUTPUT .

Int digitalRead (int pin): Esta função retorna o valor lido no pino dado. Será ALTO ou BAIXO (1 ou 0) dependendo do nível lógico no pino.

WiringPi Pin	BCM GPIO	Nome	Cabeçalho	Nome	BCM GPIO	WiringPi Pin
-	-	3.3v	1 | 2	5v	-	-
8	R1: 0 / R2: 2	SDA	3 | 4	5v	-	-
9	R1: 1 / R2: 3	SCL	5 | 6	0v	-	-
7	4	GPIO7	7 | 8	TxD	14	15
-	-	0v	9 | 10	RxD	15	16
0	17	GPIO0	11 | 12	GPIO1	18	1
2	R1: 21 / R2: 27	GPIO2	13 | 14	0v	-	-
3	22	GPIO3	15 | 16	GPIO4	23	4
-	-	3.3v	17 | 18	GPIO5	24	5
12	10	MOSI	19 | 20	0v	-	-
13	9	MISSÔ	21 | 22	GPIO6	25	6
14	11	SCLK	23 | 24	CE0	8	10
-	-	0v	25 | 26	CE1	7	11
WiringPi Pin	BCM GPIO	Nome	Cabeçalho	Nome	BCM GPIO	WiringPi Pin

Figura 4 Esquema de ligação dos terminais do Raspberry utilizando a biblioteca WiringPI

Os passos seguintes são para baixar e instalar as bibliotecas WiringPI, são:

Baixar a biblioteca no Raspberry:

$ git clone git://git.drogon.net/wiringPi

Instalar a biblioteca:

$ sudo apt-get install git-core

Executar a biblioteca como root ou sudo:

$ /build

Uma vez realizado, o código seguinte é necessário para mapear a porta do Raspberry PI. Para copilar o código o seguinte comando é executado.

$ gcc código_pir.c -o código_pir -lwiringPi

Para executar o código o seguinte comando é necessário:

./código_pir

```
//FUNCAO PARA REALIZAR A LEITURA DA PORTA do sensor de presença PIR
void sensor_PIR(void)
{
        pinMode(0,INPUT);
        int i=0;

        //Leitura do pino 17, segundo o mapa de pinos da biblioteca wiringpi

        i=digitalRead (0);   //porta 17 do RASPEBERRY PI

        if (i == 1)
        {
        /       /Sensor com movimento
                printf("Presença de Movimento detectado\n");}
    }
```

3. Sensor de Temperatura e Umidade

Este sensor inclui um componente medidor de umidade e um componente NTC para temperatura, ambos conectados a um controlador de 8-bits. O interessante neste componente é o protocolo usado para transferir dados entre o MCDU e DHT11, pois as leituras do sensor são enviadas usando apena um único fio de barramento.

Formato dos dados: 8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit check sum = 40 bits.

Modelo: DHT11:

– Alimentação: 3,0 a 5,0 VDC (5,5 Vdc máximo)

– Corrente: 200uA a 500mA, em stand by de 100uA a 150 uA

– Faixa de medição de umidade: 20 a 90% UR

– Faixa de medição de temperatura: 0º a 50ºC

– Precisão de umidade de medição: ± 5,0% UR

– Precisão de medição de temperatura: ± 2.0 ºC

– Tempo de resposta: < 5s

– Dimensões: 23mm x 12mm x 5mm (incluindo terminais)

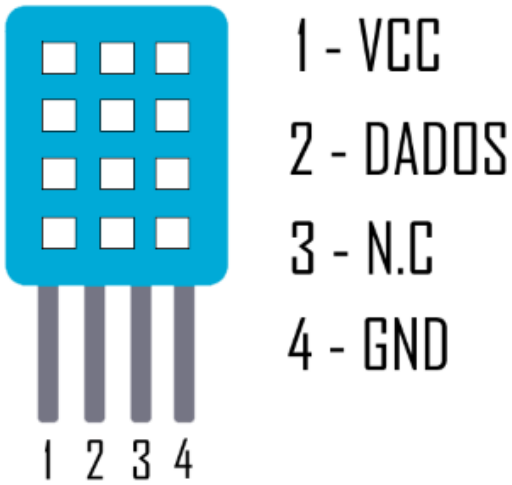

Figura 5 Esquema de ligação dos terminais do Raspberry utilizando a biblioteca WiringPI

A ligação física é composta na seguinte forma, conforme figura 5:

VCC (pino 1) -> 3.3v (pino 1)

GND (pino 4) -> GND (pino 6)

DATA (pino 2) -> GPIO7 (pino 7)

Utiliza a biblioteca WiringPI, o código é montado.

```
void sensor_dht11_dat()
{
uint8_t ultimo_estado   = HIGH;
uint8_t contador                = 0;
uint8_t j                       = 0, i;
float f; /* fahrenheit */


        dht11_dat[0] = dht11_dat[1] = dht11_dat[2] = dht11_dat[3] = dht11_dat[4] = 0;

    /* Pino de saida depois esperar 18 milliseconds */
        pinMode( DHT_PIN, OUTPUT );
        digitalWrite( DHT_PIN, LOW );
        delay( 18 );

        /* pino alto depois espera 40 microseconds */
        digitalWrite( DHT_PIN, HIGH );
        delayMicroseconds( 40 );
        /* Prepara o pino para entrada */
         pinMode( DHT_PIN, INPUT );
```

```
    /* Realizar a leitura */
        for ( i = 0; i < MAXTIMINGS; i++ )
        {
                contador = 0;
                while ( digitalRead( DHT_PIN ) == ultimo_estado )
        {
                contador++;
                delayMicroseconds( 1 );

                if ( contador == 255 )
                {
                        break;
                }
    }
        ultimo_estado = digitalRead( DHT_PIN );

        if ( contador == 255 ) {break;}
        if ( (i >= 4) && (i % 2 == 0) )
                {
                        dht11_dat[j / 8] <<= 1;
                        if ( contador > 16 )
                                dht11_dat[j / 8] |= 1;
                                j++;
                }
        }
        /* * check we read 40 bits (8bit x 5 ) + verify checksum in the last byte * print it out if data is good */

        if ( (j >= 40) && (dht11_dat[4] == ( (dht11_dat[0] + dht11_dat[1] + dht11_dat[2] + dht11_dat[3]) & 0xFF) ) )
        {
                f = dht11_dat[2] * 9. / 5. + 32;
                printf( "Umidade = %d.%d %% Temperatura = %d.%d *C (%.1f *F)\n",dht11_dat[0], dht11_dat[1], dht11_dat[2], dht11_dat[3], f );
        }else  {
                printf( "Sem bons dados\n" );
                }
    }
```

4. Envio de Email

Agora para instalar o software para enviar o e-mail usaremos ssmtp que é uma solução fácil e boa para enviar e-mail usando a linha de comando ou usando Python Script. São necessárias duas bibliotecas para enviar e-mails usando o SMTP:

- sudo apt-get install ssmtp
- sudo apt-get install mailutils

Depois de instalar as bibliotecas, precisa-se abrir o arquivo ssmtp.conf e editar este arquivo de configuração.

- sudo nano /etc/ssmtp/ssmtp.conf
- root=YourEmailAddress
- mailhub=smtp.gmail.com:587
- hostname=raspberrypi
- AuthUser=YourEmailAddress
- AuthPass=YourEmailPassword
- FromLineOverride=YES
- UseSTARTTLS=YES
- UseTLS=YES

Pode-se também testá-lo, enviando um e-mail de teste por meio do comando abaixo, se tudo está funcionando bem se deve:
Echo "Olá Derick e Eduardo" | Mail -s "Testando ..." derickhorrana12@gmail.com.

5. Linguagem Python

Criado por Guido Van Rossum a linguagem Python foi utilizada nesse projeto pela facilidade de integração com a plataforma Raspberry Pi, pela sua facilidade de codificação devido a sua estrutura orientada a objeto, e por ser uma linguagem com variáveis dinâmicas. Essas características da linguagem serão utilizadas amplamente no desenvolver do código de alerta a invasão, ou seja, o código em Python é responsável por ler o sinal do sensor e enviar o e-mail de alerta ao usuário cadastrado.

6. Código Python

O código a seguir é responsável pelo controle do sensor de presença. Ele é iniciado com a importação das bibliotecas que serão utilizadas no decorrer do algoritmo. Em seguida,são carregadas as variáveis com as informações dos e-mails do sistema de segurança e do cliente e, por fim, um laço de repetição é utilizado para verificação do estado atual do sensor e comparação com o estado anterior, de forma a disparar a notificação via e-mail caso o estado atual seja HIGH.

```
# -*- coding: utf-8 -*-
import RPi.GPIO as GPIO
import time
import numpy as np
```

```
from datetime import datetime
import os
import smtplib
from email.MIMEMultipart import MIMEMultipart
from email.MIMEBase import MIMEBase
from email.MIMEText import MIMEText
from email import Encoders
import socket
ip = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
ip.connect(("gmail.com",80))
gmail_user = "sensor.alerta@gmail.com"#E-mail do sistema
de seguraça
gmail_pwd = "********"#Senha do E-mail do sistema de
segurança
to = "derickhorrana12@gmail.com"#E-mail do cliente
subject = "ALERTA DE INTRUSO!"#Assunto do E-mail
text = "Violação na residencia detectado!\n"
+"Acesse o sistema de monitoramento pelo."
+"endereço: \n http://"+ip.getsockname()[0]+":"
+"8080/RPi_cam/ \n."# Corpo do E-mail a ser enviado
sensor = 4
GPIO.setmode(GPIO.BCM)
GPIO.setup(sensor, GPIO.IN, GPIO.PUD_DOWN)
previous_state = False # Estado inicial
current_state = False # Estado inicial
print "Iniciando..." # Log de inicio
print("1- GPIO pino %s é %s" % (sensor, current_state)) #
Log da variavel inicial
while True: # Laço de verificação
 time.sleep(0.1)
 previous_state = current_state
 current_state = GPIO.input(sensor)
 if current_state != previous_state: # Comparação com o
estado anterior
 new_state = "HIGH" if current_state
else "LOW" # Compara os dois estados, para escrever
HIGH ou LOW
56
 print("1- GPIO pino %s é %s" % (sensor, new_state)) #
Escreve a mensagem com
o estado
if current_state: # Compara se o estado é HIGH
print "2- Alerta de Movimentação!"
print "3- Capturando Imagem"
time.sleep(1)
picname = datetime.now().strftime("%d.%m.%y-%H:%M") #
Nome do
anexo
picname = '/tmp/alerta-'+picname+'.jpg' # Endereço de copia
para o
anexo
os.system("sudo cp /dev/shm/mjpeg/cam.jpg "+picname+"") #
Copia a
imagem para ser anexada
attach = picname # Adiciona o caminho da figura para ser
enviado
msg = MIMEMultipart() # Cria array para os endereços de
email
msg['From'] = gmail_user # E-mail do sistema
msg['To'] = to # E-mail do cliente
msg['Subject'] = subject # Assunto do E-mail
print "4- Enviando o E-mail" # Inicia o processo de Email
msg.attach(MIMEText(text))
part = MIMEBase('application', 'octet-stream')
part.set_payload(open(attach, 'rb').read())
Encoders.encode_base64(part)
part.add_header('Content-Disposition',
57
'attachment; filename="%s"' % os.path.basename(attach))
msg.attach(part)
mailServer = smtplib.SMTP('smtp.gmail.com:587') # inicia o
processo
de SMTP
mailServer.ehlo() # Identifica ao servidor o Protocolo de
SMTP
extendido
mailServer.starttls() # Inicia uma camada de transporte segura
mailServer.login(gmail_user, gmail_pwd) # Loga no serviço
de e-mail
mailServer.sendmail(gmail_user, to, msg.as_string()) # envia
o e-mail
mailServer.close() # Finaliza a Conexão com o servidor de e-
mail
print "5- E-mail Enviado"
os.remove(picname)
time.sleep(10)
```

IV. TRABALHOS FUTUROS

- Procurar diminuir a taxa de falha do sistema, para aumentar sua confiabilidade.
- Disponibilizar a página WEB para acesso via internet e não só uma rede local
- Diminuir a temperatura do protótipo, com pesquisa dos melhores materiais para manter o sistema em funcionamento e em segurança.
- Ligar o sistema a uma rede de segurança local, para que a força policial seja acionada com ainda mais rapidez.

V. REFERENCIAS

[1] Disponível em: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. Último acesso em Maio de 2017.

[2] Disponível em: https://www.raspberrypi.org/products/camera-module-v2/. Último acesso em Maio de 2017.

[3] Disponível em: http://www.filipeflop.com/pd-6b901-sensor-de-movimento-presenca-pir. Último acesso em Maio de 2017.

[4] Disponível em: http://blog.everpi.net/2014/07/raspberry-pi-sensor-temperatura-humidade-dht11.html. Último acesso em Maio de 2017.

[5] Disponível em: http://blog.everpi.net/2015/10/raspberry-pi-ligar-pir-sensor-movimento-hc-sr501.html/. Último acesso em Maio de 2017.